# TUTORIAL BÁSICO DEL PROGRAMADOR WEB: CSS DESDE CERO.

## Objetivos

CSS es un lenguaje de estilo que define la presentación de los documentos HTML. Por ejemplo, con CSS podemos cambiar fuentes, colores, márgenes, tamaños, imágenes de fondo, maquetación, crear animaciones y otros efectos y más. Este curso permite aprender los fundamentos de CSS, imprescindible para trabajar con páginas web hoy día.

## Destinatarios

Cualquier persona con interés en aprender fundamentos que le permitan crear páginas web usando CSS. Para realizar este curso, debes tener conocimientos previos de HTML. Se recomienda haber realizado el curso "Tutorial básico del programador web: HTML desde cero" de aprenderaprogramar.com antes de seguir este curso.

## Contenidos

- ✓ INTRODUCCIÓN A CSS. QUÉ ES Y PARA QUÉ SIRVE CSS. VERSIONES CSS. EL W3C.
- ✓ MODELO DE CAJAS CSS. ESTILOS EN LÍNEA, INTERNO Y EXTERNO. TIPOS DE ELEMENTOS.
- ✓ SELECTORES POR ID (#) Y POR CLASE(.). CONCEPTO DE HERENCIA Y DE CASCADA.
- ✓ SELECTORES AVANZADOS. PSEUDOCLASES. PSEUDOELEMENTOS.
- ✓ COLORES Y FONDOS. FUENTES Y TEXTOS. UNIDADES DE MEDIDA. PROPIEDADES CSS.
- ✓ BORDES CSS. MARGIN Y PADDING. PROPIEDADES BÁSICAS. EFECTOS AVANZADOS.
- ✓ ENLACES CON CSS. LISTAS CON CSS. TABLAS CON CSS. MENÚS CON CSS.
- ✓ POSICIONAMIENTO. FLOAT Y CLEAR. MAQUETACIÓN EN COLUMNAS. DISEÑO FLUIDO.
- ✓ PREFIJOS CSS DE NAVEGADOR: CHROME, FIREFOX, INTERNET EXPLORER, OPERA, SAFARI.
- ✓ EFECTOS CSS DE ÚLTIMA GENERACIÓN. USO DE SPRITES. ANIMACIONES.

## Duración

150 horas de dedicación efectiva, incluyendo lecturas, estudio y ejercicios.

## Dirección, modalidades y certificados

El curso está dirigido por César Krall, Responsable del Departamento de Producción de aprenderaprogramar.com del portal web aprenderaprogramar.com. Se oferta bajo la modalidad web (gratuito).

aprenderaprogramar.com

# INDICE DEL CURSO

**1. INTRODUCCIÓN A CSS. QUÉ ES Y PARA QUÉ SIRVE CSS. VERSIONES CSS. EL W3C.**

1.1. ¿Qué es CSS? HTML, conocimiento previo necesario.
1.2. ¿Es CSS un lenguaje de programación? ¿Para qué sirve? Diferencias de HTML, CSS, PHP, ASP...
1.3. Diferencias entre CSS y HTML. Frontera entre CSS, HTML y programación.
1.4. CSS en aplicaciones web Joomla, WordPress, Drupal, phpBB... Plantillas, Templates o Themes.
1.5. Empezar a usar CSS a partir de un documento HTML con estructura básica.
1.6. Versiones CSS. Algo de historia y perspectiva. ¿Qué es el W3C? Recomendaciones oficiales.
1.7. Documentación especificación oficial CSS. W3schools y W3fools. Validación CSS W3C validator.
1.8. ¿Qué necesito para escribir código CSS y crear páginas web?

**2. MODELO DE CAJAS CSS. ESTILOS EN LÍNEA, INTERNO Y EXTERNO. TIPOS DE ELEMENTOS.**

2.1. De la estructura HTML y su semántica al modelo de cajas CSS. Elementos block e inline CSS.
2.2. Estilos por defecto, en línea, interno y externo. Sobreescritura de estilos. link rel.
2.3. Archivos CSS. Comentarios CSS.

**3. SELECTORES POR ID (#) Y POR CLASE(.). CONCEPTO DE HERENCIA Y DE CASCADA.**

3.1. Selectores por id. Selectores por class. Ejemplos.
3.2. Concepto de herencia en hojas de estilo CSS. ¿Qué es? Forzar herencia con inherit.
3.3. Conceptos: cascada y herencia CSS. Estilos de usuario. !important. Ejemplos prácticos.

**4. SELECTORES AVANZADOS. PSEUDOCLASES. PSEUDOELEMENTOS. EJEMPLOS PRÁCTICOS.**

4.1. Selectores avanzados, pseudoclases y pseudoelementos CSS.
4.2. first-child, nth-child, last, not, -letter -line, after, before.
4.3. Selector universal. nth-of-type y nth-last-child.

**5. COLORES CSS.**

5.1. Colores HTML y CSS. RGB decimal o porcentual. Códigos de colores hexadecimales.
5.2. Web safe colors. Colores RGBA, HSL, HSLA. Transparencia CSS. Lista de colores HTML - CSS.

## 6. FONDOS.

6.1. Color de fondo. Propiedad background-color CSS.
6.2. Definición de fondo CSS. background-image. Efecto fondo página web. background-repeat.
6.3. CSS background-position, background-attachment, clip, origin y size. Shortand fondo. Ejemplos.
6.4. Definición de fondo CSS. background-image. Efecto fondo página web. background-repeat.

## 7. UNIDADES DE MEDIDA. PROPIEDADES WIDTH Y HEIGHT.

7.1. Unidades de medida CSS relativas o absolutas. in, cm, mm, pt, pc, pixel px, porcentaje, em, ex.
7.2. Propiedades CSS width y height. auto (automático). Ejemplos prácticos y ejercicios resueltos.

## 8. BORDES CSS. PROPIEDADES RELACIONADAS.

8.1. Tipos de borde CSS. border-style hidden, solid... Efectos 3D. border-top right bottom left.
8.2. Shortand: notación CSS abreviada. border-width (thin, thick) border-color (transparent).
8.3. Propiedad outline.

## 9. MARGEN (MARGIN) Y RELLENO (PADDING). PROPIEDADES RELACIONADAS.

9.1. Concepto de margen y relleno CSS. Diferencias entre margin y padding CSS (box model).
9.2. Padding y margin CSS. Top, right, bottom, left. Margin negativo y centrar con margin auto.

## 10. POSICIONAMIENTO, MAQUETACIÓN Y DISEÑO CON CSS. POSITION, FLOAT, CLEAR Y MÁS.

10.1. Propiedad position CSS: static, relative, absolute, fixed. Top, right, bottom, left. Ejemplos.
10.2. Propiedad display CSS. inline, block, none, list-item. Ejercicios de ejemplo resueltos.
10.3. Concepto float CSS. none, left, right y ¿centrar?. Colocar texto alrededor de una imagen.
10.4. clear CSS both ¿El texto no envuelve una imagen html? Explicación a problemas.
10.5. Diseño web CSS dos, tres, cuatro columnas con float ¿width en porcentajes no funciona?
10.6. overflow CSS. scroll. overflow-x, overflow-y. Propiedad visibility (visible, hidden, collapse).
10.7. z-index CSS ¿no funciona? Superposición de elementos tipo capas, objetos, div o imágenes.
10.8. CSS vertical-align middle Centrar verticalmente una imagen, texto, div, etc. Ejemplos.
10.9. Diseño líquido CSS (fluido) frente a responsive-design.
10.10. Propiedades CSS max-width, min-width, max-height, min-height.

## 11. FUENTES Y TEXTO.

11.1. CSS text-align, color, text-decoration, text-indent, white-space nowrap, pre, pre-wrap, pre-line.
11.2. CSS efecto sombra con text-shadow y blur, text-overflow clip, ellipsis.
11.3. Interlineado: propiedad CSS line-height.

11.4. CSS text-transform y first-letter, letter-spacing y word-spacing.
11.5. Cortar palabras largas con word-wrap.
11.6. CSS font-size (uso de larger, smaller).
11.7. font-weight o negrita (bolder, lighter).
11.8. font-style o cursiva italic.
11.9. CSS font-family. tipografía (tipos de letra). Lista de tipos.
11.10. font-variant (small-caps). Shortand font.
11.11. @font-face CSS. Fuentes web gratuitas. Problemas.
11.12. Convertir woff, eot, ttf, otf, svg. Font converters.

**12. ENLACES CSS.**

12.1. Pseudoclases CSS link, visited, focus, hover y active. Estilos y efectos en links.

**13. LISTAS CSS.**

13.1. Listas CSS. list-style-type, list-style-position (outside, inside), list-style-image.
13.2. Shortand list-style.
13.3. Menú horizontal CSS o vertical. Crear menús con efectos a partir de listas CSS. Ejemplos.
13.4. Menú desplegable CSS horizontal. Efecto dropdown. Ejemplo de código (float, display..).

**14. TABLAS CSS.**

14.1. Estilos y herencia CSS en tablas. Width, height, font-size y overflow en tablas. border-collapse.
14.2. Diseño de tablas CSS. border-spacing, caption-side, empty-cells.
14.3. Diseño de tablas CSS. Colores de filas intercalados alternos.

**15. COMPATIBILIDAD Y PREFIJOS DE NAVEGADOR: CHROME, FIREFOX, INTERNET EXPLORER, OPERA, SAFARI.**

15.1. Comentarios condicionales CSS. Problemas: compatibilidad entre navegadores.
15.2. Strict mode. Filtros.
15.3. Prefijos CSS de navegador Chrome, Firefox, Internet Explorer, Opera, Safari.
15.4. Prefijos -webkit -moz -ms -o Ejemplos.

**16. EFECTOS CSS DE ÚLTIMA GENERACIÓN. USO DE SPRITES. ANIMACIONES.**

16.1. Sprite CSS: concepto. ¿Qué es, cómo crear y usar un sprite? Ejemplos y código resuelto.
16.2. Columnas CSS: column-count, column-width, columns, colum-gap y colum-rule. Ejemplos.
16.3. Efecto CSS esquinas redondeadas: border-radius. border-left-top-radius. Círculo o elipse.
16.4. Efecto sombra CSS: box-shadow. Sombra interior inset. Difuminado blur, spread. Ejercicios.
16.5. Degradados CSS. Efecto lineal y radial. linear-gradient.

16.6. Angulos CSS: unidades deg, grad, turn, rad.
16.7. Efectos CSS. radial-gradient tipo circle o ellipse.
16.8. Efectos CSS. transform: rotate, scale, skew y translate. Rotar, escalar, sesgar, trasladar.
16.9. Animaciones CSS. transition-property, transition-duration, timing-function y efecto delay.
16.10. Animación CSS. Diferenciar transición y animación. @keyframes: fotogramas o estados clave.
16.11. Animation CSS. name, duration, delay, fill-mode, iteration-count.
16.12. Animation CSS. direction, timing-function, play-state.
16.13. cursor CSS. Tipos. Efectos hover: move, no-drop, resize, not-allowed, crosshair, progress, wait.

**17. MÁS SOBRE CSS.**

17.1. Otros aspectos de CSS.
17.2. ¿Qué hemos aprendido y qué no hemos aprendido con este curso de CSS?

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

## ORIENTACIÓN SOBRE EL CURSO PASO A PASO "TUTORIAL BÁSICO DEL PROGRAMADOR WEB: CSS DESDE CERO"

CSS es un lenguaje diseñado para separar el contenido de las páginas web de su presentación. Así, un mismo contenido podría verse de distintas maneras o con distintos aspectos aplicándole distintos estilos CSS.

CSS no es un lenguaje de programación propiamente dicho, aunque a veces se lo denomina lenguaje de programación coloquialmente. Esto es posiblemente debido a que muchas veces el lenguaje CSS se mezcla de alguna manera con el HTML o con lenguajes de programación como PHP. Esta mezcolanza implica que a veces el código CSS esté junto al HTML o PHP, o que a la hora de desarrollar una web o solucionar un problema en una web sean necesarios conocimientos tanto de HTML como de CSS como de un lenguaje de programación propiamente dicho.

CSS es un lenguaje descriptivo, formado por una serie de atributos, valores y etiquetas que el navegador interpretará de una u otra forma mostrando distintos aspectos para las diferentes etiquetas que forman la estructura de una página HTML. Es decir, podemos obtener muchos aspectos y diseños para una misma página HTML simplemente cambiando su hoja de estilos.

Este curso que estamos comenzando va dirigido a aquellas personas que quieran adquirir unos fundamentos básicos para crear hojas de estilo con vistas a poder desarrollar en el futuro páginas web atractivas y de cierta complejidad. No vamos a desarrollar un manual de referencia de CSS, sino un curso básico paso a paso. No vamos a contemplar todos los aspectos de las hojas de estilo CSS, sino aquellos que consideramos básicos desde el punto de vista didáctico, con vistas a que posteriormente la persona que lo desee amplíe sus conocimientos. Nuestro objetivo es ser **claros, sencillos y breves**, y para eso tenemos que centrarnos en determinadas cuestiones de CSS y dejar de lado otras.

Como conocimientos previos para iniciar este curso recomendamos (seguir la recomendación o no queda a criterio del alumno y/o profesor que vayan a seguir el curso) estos: Ofimática básica (saber copiar, pegar, mover y abrir archivos, etc.) y haber realizado el Curso básico de HTML que se ofrece en aprenderaprogramar.com (http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=69&Itemid=192).

Conocer algún lenguaje de programación sería un aspecto positivo.

Los conocimientos previos son, como hemos dicho, deseables pero no imprescindibles.

Aprender CSS requiere tiempo y esfuerzo. Para hacer ese recorrido más llevadero, te recomendamos que utilices los foros de aprenderaprogramar.com, herramienta a disposición de todos los usuarios de la web (http://www.aprenderaprogramar.com/foros/), y que te servirá para consultar dudas y recabar orientación sobre cómo enfrentarte a los contenidos. Entre los miembros del portal web y otros usuarios, trataremos de ayudarte para que el estudio te sea más llevadero y seas capaz de adquirir los conocimientos necesarios y avanzar como programador o diseñador web.

El tiempo necesario (orientativamente) para completar el curso incluyendo prácticas con ordenador, suponiendo que se cuenta con los conocimientos previos necesarios, se estima en 90 horas de

© aprenderaprogramar.com, 2006-2029

dedicación efectiva o aproximadamente un mes y medio con una dedicación de 3 horas diarias de lunes a viernes. Aprender a crear páginas web requiere dedicación y esfuerzo.

El curso ha sido generado paso a paso usando Windows como sistema operativo y por ello contiene algunas indicaciones específicas para usuarios de Windows, pero también puede ser utilizado en otros entornos (Linux, Macintosh, etc.).

Estamos seguros de que con tu esfuerzo y la ayuda que te podamos brindar este curso te resultará de gran utilidad.

**Próxima entrega:** CU01003D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**DEFINICIÓN O CONCEPTO DE LENGUAJE CSS**

CSS es un lenguaje utilizado en la presentación de documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Así, podemos decir que el lenguaje CSS sirve para dotar de presentación y aspecto, de "estilo", a una página web. Este lenguaje es principalmente utilizado por parte de diseñadores y programadores web para elegir multitud de opciones de presentación como colores, tipos y tamaños de letra, imágenes de fondo, bordes, etc.

La filosofía de CSS se basa en intentar separar lo que es la estructura o contenido y configuración básica del documento HTML de su presentación. Por decirlo de alguna manera: la página web sería lo que hay debajo (el contenido) y CSS sería "un cristal de color" que hace que el contenido se vea de una forma u otra. Usando esta filosofía, resulta muy fácil cambiarle el aspecto a una página web: basta con cambiar "el cristal" que tiene delante. Piensa por ejemplo qué ocurre si tienes un libro de papel y lo miras a través de un cristal de color azul: que ves el libro azul. En cambio, si lo miras a través de un cristal amarillo, verás el libro amarillo. El libro (el contenido) es el mismo, pero lo puedes ver de distintas maneras. CSS permite cambiar el estilo a los contenidos de las páginas web.

Algunas opciones básicas del lenguaje CSS por ejemplo pueden ser el poder cambiar el color de algunas típicas etiquetas HTML como <H1> (h1 es una etiqueta en el lenguaje HTML destinada a mostrar un texto como encabezado, en tamaño grande). Pero también hay funciones algo más complejas, como introducir espaciado entre elementos <DIV> (div es una etiqueta HTML para identificar una determinada región o división de contenido dentro de una página web) o establecer imágenes de fondo, bordes redondeados, etc..

CSS es muy intuitivo y sencillo una vez se llega a aprender, ya que para su definición siempre se hace uso de un identificador de etiqueta HTML (como por ejemplo <H1>), y luego indicamos con qué aspecto queremos que se muestren todas las etiquetas <H1> que aparezcan en un documento. Podemos definir cómo queremos que se muestren las distintas partes del documento HTML, pudiendo en cada caso definir sus propiedades (color, tipo de fuente, tamaño, espacio de márgenes, bordes, imagen de fondo, etc.) con algún determinado valor deseado.

Vamos a ver primero lo que sería un ejemplo muy sencillo de aplicación de CSS, que tratará de una página web o archivo HTML donde tan solo tendremos un párrafo de texto. El texto estará dentro de una etiqueta <p>. Distinguiremos con este ejemplo entre contenidos y presentación o aspecto.

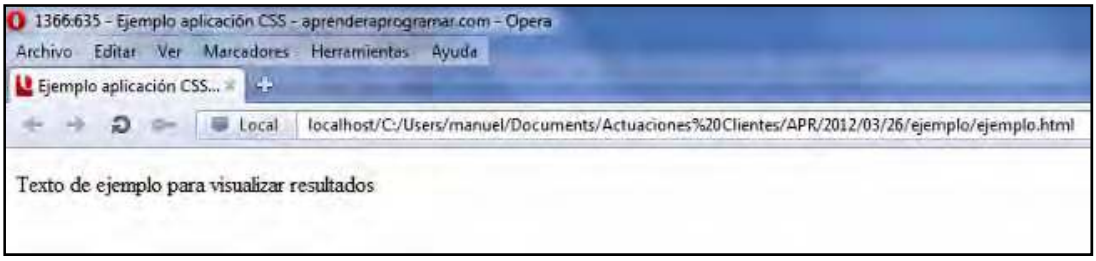
Para seguir este tutorial es necesario que tengas conocimientos básicos de HTML. Si no los tienes, te recomendamos que realices primero el curso básico de HTML disponible en la sección de cursos de aprenderaprogramar.com (o si prefieres acceder directamente, puedes hacerlo a través de este enlace: http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=69&Itemid=192).

© aprenderaprogramar.com, 2006-2029

Nuestro documento html contendrá el siguiente texto de partida (en este caso se ha llamado ejemplo.html):

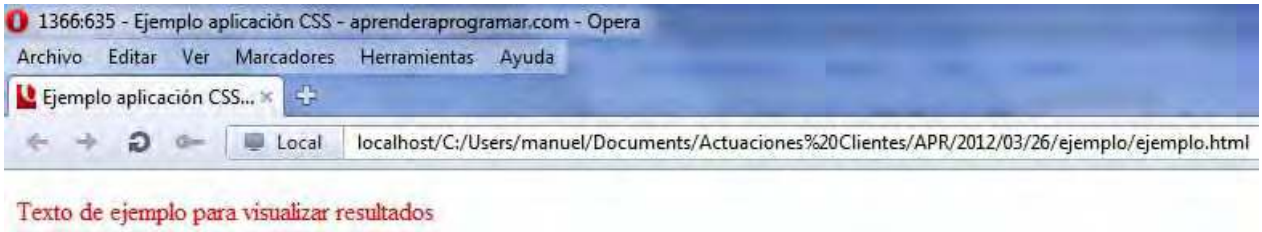
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
        <head>
                <title>Ejemplo aplicación CSS - aprenderaprogramar.com</title>
        </head>
        <body>
                <p>Texto de ejemplo para visualizar resultados </p>
        </body>
</html>
```

Una vez creado el documento ejemplo.html en nuestro ordenador, lo abriremos con un navegador web, obteniendo un resultado que será similar a éste:




Tenemos una página web que tan solo tiene un párrafo ("Texto de ejemplo para visualizar resultados"). Este aparece en color negro por defecto y nosotros, para ver la utilidad de CSS deseamos mostrar el texto en rojo. En realidad con CSS podemos hacer cosas mucho más complejas, pero ahora sólo queremos poner un ejemplo para mostrar la utilidad de CSS.

Una vez hayamos creado un archivo con el estilo para aplicar a los párrafos (durante el curso aprenderemos cómo hacer esto y muchas más cosas) podremos hacer que todos los párrafos de una página web se muestren de una misma manera sin tener que modificar párrafo a párrafo, sino únicamente modificando el estilo CSS que se debe aplicar a los párrafos. Como decíamos, introduciendo el código CSS adecuado podremos lograr que cambie el aspecto de nuestra página web:




¿Hace falta CSS para realizar esto? Antiguamente no. Podíamos hacerlo simplemente usando HTML:

© aprenderaprogramar.com, 2006-2029

```
<font color ="red"><p> Texto de ejemplo para visualizar resultados</p></font>
```

Sin embargo, con esta escritura sólo estábamos modificando el aspecto de un párrafo, mientras que con CSS podemos modificar el aspecto de todos los párrafos de una página web, repartidos en cientos de líneas y de archivos, "automáticamente". Con la evolución de HTML, algunas formas sintácticas que servían para dotar de aspecto a las páginas web han pasado a considerarse obsoletas o descatalogadas, dejando que la presentación quede controlada de forma independiente a través de CSS.

CSS hace que resulte fácil cambiar la presentación ya que si ahora quisiéramos cambiar el color, fuente, tamaño, etc, de todos los párrafos de nuestra página web, tan solo deberíamos de cambiar las propiedades en el archivo donde hayamos definido el estilo para los párrafos, sin alterar nada en la página web. Esto es mucho más fácil, rápido y legible que tener que estar modificando todas y cada una de las etiquetas <p> que aparecieran en nuestra página web.

Ahora bien, imaginemos que tenemos una página web con 3 etiquetas <p> pero no deseamos que todas tengan la misma presentación. CSS tiene previsto cómo poder aplicar distintos estilos a ciertos párrafos o etiquetas. Durante el curso veremos cómo. Esta imagen sería un ejemplo de uso de distintos estilos para la etiqueta <p> de párrafos.

La forma de definición de estilos CSS separa la presentación de la información en una página web. Es muy útil porque tenemos los estilos por un lado y los contenidos por otro. Si en un momento dado queremos cambiar la forma en que se ve la página pero no sus contenidos, únicamente tendríamos que modificar los archivos css. Pero también existen estas otras formas de aplicar estilos, lo veremos a lo largo del curso.

Quizás los ejemplos que hemos visto hasta el momento te resulten muy sencillos y poco atractivos. A continuación mostramos imágenes de lo que puede hacerse utilizando CSS a nivel avanzado, que como verás puede ser realmente espectacular.

Fíjate en las dos imágenes que mostramos a continuación y trata de identificar qué tienen en común y qué tienen distinto:

© aprenderaprogramar.com, 2006-2029

Podemos decir que tienen en común un mismo texto, un mismo menú y posición de menú. En cambio podemos decir que cambian los colores, tipos y tamaños de letra y las imágenes de fondo, tanto detalles decorativos como las flores o personas, como fondos que crean marcos para el texto o los menús. Con un conocimiento avanzado de CSS podremos lograr efectos similares a estos que estamos viendo y cambiar el aspecto de una página web con sólo un click de botón.

**Próxima entrega:** CU01004D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## PARA QUÉ SIRVE CSS

CSS es un lenguaje que sirve para dotar de presentación y aspecto, de "estilo", a páginas web (documentos HTML). CSS no es un lenguaje de programación. Podríamos decir que es un lenguaje que suele aparecer relacionado o próximo a un lenguaje de programación o que suele colaborar con un lenguaje de programación, pero no es un lenguaje de programación propiamente dicho.

A veces oirás hablar de "Lenguajes de programación HTML y CSS". Esta expresión es, desde el punto de vista formal, incorrecta, ya que ni HTML ni CSS son lenguajes de programación. No obstante, a veces coloquialmente se usa el término "programación CSS".

Un lenguaje de programación es un lenguaje que se usa para realizar procesos de interés a través de un ordenador o dispositivo electrónico, desde un cálculo para un estudiante o ingeniero, a una compra por internet, pasando por cualquier cosa que se te ocurra. Un lenguaje de programación tiene como características básicas el tener la capacidad para "tomar decisiones" o ejecutar un proceso u otro en función de las circunstancias (por ejemplo dependiendo del botón que pulse el usuario), así como el ser capaz de repetir procesos repetidas veces hasta que se cumpla una condición. CSS no es un lenguaje que permita cumplir estas funciones, por tanto no es un lenguaje de programación aunque se use junto a lenguajes de programación.

CSS es un lenguaje que apareció para hacer más fáciles y con mejor aspecto los desarrollos web. Un desarrollo web comprende múltiples áreas de conocimiento:

© aprenderaprogramar.com, 2006-2029

En la clasificación que hemos hecho, CSS estaría englobado dentro del área de diseño gráfico y maquetación.

Los desarrollos web tienen dimensiones muy variables. Podemos hablar desde una pequeña página web para una empresa local hasta un gran portal para una empresa de ámbito internacional. En ambos casos podríamos decir que interviene la programación web y el diseño web. Sin embargo, un pequeño desarrollo puede ser llevado a cabo por una sola persona que abarque tanto programación como diseño, mientras que un gran desarrollo requiere de un equipo de trabajo más o menos amplio y con distintos especialistas, ya que en torno a los desarrollos web hay diferentes áreas de conocimiento implicadas (análisis, diseño, programación, sistemas, integración, testing, etc.).

En un gran desarrollo existen personas especializadas en las distintas áreas, de modo que el programador no suele trabajar en el diseño (excepto para hacer algún retoque o cambio, o para solucionar problemas). No obstante, sí resulta conveniente que un programador web tenga los conocimientos básicos de CSS ya que le resultarán útiles y necesarios, por un lado para la solución de problemas y por otro para integrar cuestiones donde el diseño y la programación se entremezclan.

Si miramos a los lenguajes o tecnologías que hay en torno a los desarrollos web podríamos hacer una clasificación que comprende: HTML, CSS, Bases de datos, Servidores, Lenguajes de programación del lado del cliente (p.ej. Javascript) y Lenguajes de programación del lado del servidor (p.ej. PHP).

HTML y CSS son tecnologías (o metalenguajes, ya que no puede considerárselos lenguajes de programación) que intervienen en prácticamente todo desarrollo, grande o pequeño. Se encargan de dotar de una estructura y presentación agradables a aquello que ve el usuario de páginas web.

© aprenderaprogramar.com, 2006-2029

Los lenguajes de programación del lado del servidor realizan procesos en el servidor (computador remoto que se encarga de enviar las páginas web a través de internet): podemos citar entre estos lenguajes Java (JSP), ASP.NET, PHP, o Perl, entre los principales.

Los lenguajes de programación del lado del cliente realizan procesos en el ordenador personal del usuario (efectos visuales, cálculos, etc.): podemos citar entre estos lenguajes Javascript, Java (applets), o VBScript, entre los principales.

En cuanto a bases de datos podemos nombrar MySQL, SQLServer y Oracle, entre las principales.

Las tecnologías se combinan entre ellas de muy diversas maneras. Podemos citar algunas combinaciones bastante habituales entre lenguajes de programación y bases de datos: Java + Oracle, ASP.NET + SQLServer, PHP + MySQL. Sea cual sea la combinación utilizada, en un desarrollo web moderno siempre intervendrá HTML y CSS.

En resumen, CSS es un lenguaje para dotar de presentación y estilo a páginas web cuyos aspectos básicos deben ser conocidos tanto por programadores web como por diseñadores web o maquetadores web. En la práctica, muchas veces se entremezcla el código de programación con el código HTML y código CSS, de ahí que coloquialmente se hable de "programación web" para referirse a todo este conjunto, aunque formalmente ni HTML ni CSS son lenguajes de programación.

Fíjate que estamos tratando de dejar claro qué es y para qué sirve CSS antes de empezar a estudiar este lenguaje porque si tenemos los conceptos claros nos será mucho más sencillo el aprendizaje, ahorraremos tiempo y cometeremos menos errores.

**Próxima entrega:** CU01005D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## EFECTOS CSS. UBICAR LA FRONTERA

En el anterior epígrafe del curso hemos indicado que CSS no es un lenguaje de programación pero que en cierto sentido se entremezcla con éstos. A veces nos encontraremos que se puede lograr un mismo efecto usando HTML, usando CSS ó usando un lenguaje de programación. ¿Por qué tantas formas para hacer una misma cosa? ¿Dónde está la frontera entre cada lenguaje?

Esta pregunta no es de fácil respuesta. Vamos a ver con un ejemplo lo que puede ocurrir para algo tan sencillo como aplicar algunos efectos a un texto. No obstante, ten en cuenta que este ejemplo relativo a texto podría aplicarse a otros conceptos como bordes, márgenes, animaciones, etc.

El lenguaje HTML permite aplicar algunos efectos visuales al texto. Escribe o copia este código y guárdalo en un archivo de nombre ejemplo1.html.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
	<head>
		<title>Ejemplo HTML aprenderaprogramar.com</title>
		<meta name="tipo_contenido"  content="text/html;" http-equiv="content-type" charset="utf-8">
	</head>
	<body>
		<p>Negrita: <strong>Quiero aprender a programar</strong></p>
		<p>Itálica:  <i>Quiero aprender a programar</i></p>
		<p>Tachado:  <strike>Quiero aprender a programar</strike></p>
		<p>Color fuente:  <font color ="green">Quiero aprender a programar</font></p>
	</body>
</html>
```

Con Javascript podemos hacer algo parecido. Escribe o copia este código y guárdalo en un archivo de nombre ejemplo2.html:

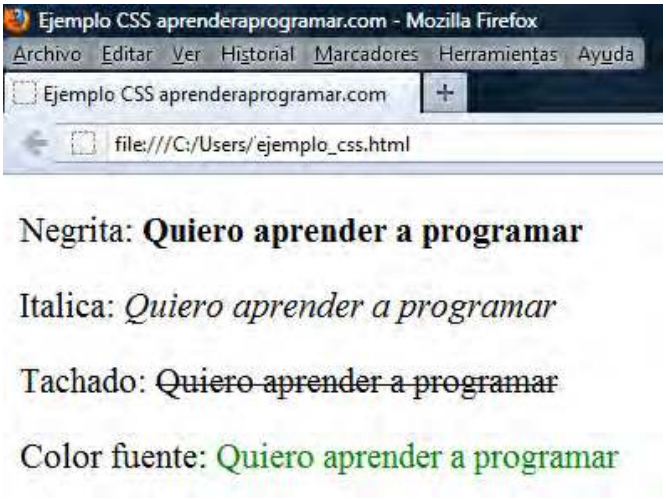
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
	<head>
		<title>Ejemplo Javascript aprenderaprogramar.com</title>
		<meta name="tipo_contenido"  content="text/html;" http-equiv="content-type" charset="utf-8">
	</head>
	<body>
		<script>
		var txt = "Quiero aprender a programar";
		document.write("<p>Negrita: " + txt.bold() + "</p>");
		document.write("<p>Itálica: " + txt.italics() + "</p>");
		document.write("<p>Tachado: " + txt.strike() + "</p>");
		document.write("<p>Color fuente: " + txt.fontcolor("green") + "</p>");
		</script>
	</body>
</html>
```

© aprenderaprogramar.com, 2006-2029

Y por último en vez de aplicar Javascript o simple HTML, podemos usar CSS. Escribe o copia este código y guárdalo en un archivo de nombre ejemplo3.html:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
	<head>
		<title>Ejemplo CSS aprenderaprogramar.com</title>
		<meta name="tipo_contenido"  content="text/html;" http-equiv="content-type" charset="utf-8">
	<style type="text/css">
		#negrita{font-weight:bold;}
		#italica{font-style:italic;}
		#tachado{text-decoration: line-through;}
		#verde{color:green;}
	</style>
	</head>
	<body>
		<p>Negrita: <span id="negrita">Quiero aprender a programar</span></p>
		<p>Italica: <span id="italica">Quiero aprender a programar</span></p>
		<p>Tachado: <span id="tachado">Quiero aprender a programar</span></p>
		<p>Color fuente:  <span id="verde">Quiero aprender a programar</span></p>
	</body>
</html>
```

Haz doble click sobre cada uno de los archivos para visualizar el resultado en un navegador. El resultado que obtenemos es algo similar a esto:




El único código que debemos entender por el momento es el correspondiente al ejemplo 1, ya que es código HTML que ya debemos conocer. El código del ejemplo 2 y ejemplo 3 no te preocupes si no lo entiendes ya que el objetivo ahora no es comprender ese código, sino simplemente ver cómo podemos alcanzar un mismo objetivo usando distintos lenguajes como HTML, Javascript ó CSS.

Además si nos fijamos, el código Javascript y el código CSS está dentro de un documento HTML (aunque podrían estar en archivos separados).

Todo esto nos puede llevar a preguntarnos: ¿Por qué se entremezclan unos lenguajes con otros? La respuesta sería histórica y técnica: HTML se convirtió en la forma de crear páginas web, pero tenía

© aprenderaprogramar.com, 2006-2029

muchas limitaciones. En un momento dado, se consideró que entremezclar (embeber) lenguajes entre sí podía ser una buena opción técnica para resolver problemas o hacer cosas que no era posible o conveniente hacer con HTML. Así, podemos embeber Javascript en HTML ó embeber CSS en HTML, o embeber HTML en PHP, etc. Por ello a veces ocurre que no hay una frontera clara entre lenguajes de programación, HTML y CSS. Esto, que puede resultar un tanto confuso inicialmente, se va convirtiendo en "comprensible" a medida que se trabaja y se aprende más sobre estos lenguajes.

Por otro lado, ¿por qué tantas vías distintas para hacer algo cuando quizás que solo hubiera una manera de poner el texto en negrita, o una sola manera de poner un color de fuente, sería más simple?

Para esto podemos citar varios motivos:

a) **Motivos históricos:** a veces las cosas se empezaron a hacer de una manera y luego se pensó que era mejor hacerlas de otra. Sin embargo, para evitar que las páginas web existentes dejaran de funcionar, se siguieron permitiendo formas de hacer las cosas "anticuadas". Por ejemplo la etiqueta <strike> ...</strike> en HTML se considera deprecated (obsoleta, de uso no recomendado) en HTML 4.01 y no está admitida en HTML 5. Sin embargo, se sigue usando. Muchas formas de hacer las cosas se admiten aunque no estén recomendadas.

b) **Motivos de independencia de tecnologías:** HTML es una cosa y Javascript es otra, aunque en la práctica nos encontremos con que Javascript se puede "entremezclar" (embeber) en HTML. Podríamos hacer cosas en Javascript y no querer usar otro lenguaje, es decir, podríamos tratar de hacer cosas independientes sin "entremezclar" tecnologías. Por ello lenguajes como Javascript ó PHP incorporan posibilidades para hacer cosas que ya se pueden hacer de otra manera. De esta forma tienen la potencialidad de ser más independientes.

c) **Motivos de políticas de desarrollo y estándares:** quizás no te lo hayas preguntado nunca, pero conviene tener al menos una orientación al respecto: ¿Quién define qué es un lenguaje como HTML, CSS, PHP, cómo se debe escribir, qué resultado debe generar cada etiqueta o instrucción, etc.? En general detrás de los lenguajes, aunque algunos fueron creados por personas individuales, hay empresas, comunidades de desarrollo, asociaciones, consorcios internacionales, comités, etc. En ocasiones un grupo de personas no está de acuerdo con la forma en que se está creando una especificación del lenguaje y forman grupos de trabajo alternativos que definen estándares alternativos. A veces triunfa un estándar y el otro se desecha, pero otras veces conviven distintos estándares que permiten hacer las cosas de distintas maneras. Para los creadores de páginas web esto resulta negativo, pero ¡así es la vida!

d) **Otros motivos:** podríamos abundar en el por qué de que las cosas sean como son, pero con tener una idea general nos basta.

Acostúmbrate a pensar que los desarrollos web no son matemáticas. Las cosas se pueden hacer de muchas maneras, y de hecho muchas veces se hacen "de mala manera" por desconocimiento, por prisas, o por ser más fácil.

Acostúmbrate a pensar que los desarrollos web usan distintos lenguajes que muchas veces se entremezclan entre sí hasta el punto de ser difícil distinguir en qué corresponde a un lenguaje y qué

© aprenderaprogramar.com, 2006-2029

corresponde a otro. Hay lenguajes comunes en los desarrollos web como HTML, pero por ejemplo en cuanto a lenguajes de programación no todos los programadores usan el mismo.

Acostúmbrate a encontrarte con que a veces las cosas no funcionan como una esperaría que lo hicieran, no debido a que se haya escrito mal el código o usado mal una instrucción, sino debido a que en el mundo de internet existen distintos estándares y distintas versiones. A veces aunque nos esforcemos porque todo se vea como nosotros esperamos en todos los navegadores o dispositivos, es difícil conseguirlo. Es un poco caótico, pero es así.

En este curso más que aprendernos todas las instrucciones, estándares, etc. vamos a tratar de ser capaces de razonar el por qué de las cosas, y a tratar de esforzarnos por saber cómo generar código limpio, bien estructurado y de calidad.

**Próxima entrega:** CU01006D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## CSS EN APLICACIONES WEB

Ya hemos dicho que CSS nos permite separar el contenido de una página web de su presentación o aspecto. Una de las aplicaciones más ampliamente extendida de CSS está en dotar de un aspecto atractivo a las aplicaciones web, dentro de las que destacan los Gestores de Contenidos o CMS (Content Management Systems). Un CMS es software que se instala en el servidor y sirve para publicar contenidos en una página web fácilmente.

El concepto de aplicación web (programa que se aloja en un servidor remoto o hosting y al que accedemos a través de internet) es muy amplio, de hecho con el paso de los años se ha hecho tan amplio como los programas de ordenador o las actividades que realiza el ser humano. Las aplicaciones web se han popularizado en los últimos años gracias a que buena parte de estas aplicaciones se comenzaron a distribuir y utilizar de forma gratuita, con una comunidad de usuarios y desarrolladores de software en torno a ellas.

Muchas de estas aplicaciones sirven para que personas que no tienen conocimientos de informática gestionen páginas web como tiendas de comercio electrónico, foros, portales de contenidos, periódicos digitales, etc.

Las aplicaciones web se podrían clasificar de varias maneras. De hecho es difícil realizar una clasificación debido a que los campos en que se utilizan las distintas aplicaciones muchas veces se solapan. Vamos a hacer una clasificación común, que es basándonos en el tipo de página web para el que son más habitualmente usados:

| CLASIFICACIÓN | EJEMPLOS | DESCRIPCIÓN |
|---|---|---|
| **Gestores de Contenidos** | Joomla, Drupal, OpenCMS, Plone, WordPress, b2evolution, Geeklog, Serendipity, Textpattern, CMS Made Simple, concrete5, Contao, ImpressPages, liveSite, Nucleus, PyroCMS, TYPO3, Chamilo, Moodle, phpMyFAQ, e107, Mahara, Mambo, ocPortal, PHP-Fusion, PHP-Nuke, Tiki Wiki, Xoops, Zikula | Orientados a crear portales web de muy diferentes temáticas, desde un periódico digital hasta una tienda on-line o un blog, página personal, etc. |
| **Foros y libros de visitas** | phpBB, SMF, fluxBB, MyBB, Vanilla Forums, XMB Forums, GBook, Lazarus GuestBook, | Pensados para la creación de sistemas de foros donde los usuarios participan intercambiándose mensajes o para libros de visitas |

© aprenderaprogramar.com, 2006-2029

| CLASIFICACIÓN | EJEMPLOS | DESCRIPCIÓN |
|---|---|---|
| **Wikis** | MediaWiki, DocuWiki, PmWiki, WikkaWiki, TikiWiki, PikiWiki | Pensados para mantener un sistema de información entre una comunidad de usuarios. Este sistema puede ser generalista como wikipedia o estar especializado en un área o campo de conocimiento concreto. |
| **Tiendas y comercio electrónico** | Magento, PrestaShop, CubeCart, OpenCart, osCommerce, TomatoCart, Zen Cart, | Pensadas para crear tiendas electrónicas y galerías de productos destinadas al comercio electrónico. |
| **Utilidades varias** | ExtCalendar, phpScheduleit, WebCalendar, phpFreeChat, phpMyChat, DadaMail, PHPList, SiteRecommender, OpenX, OSClass, QuickSell Classifieds, Help Center Live, Hesk, osTicket | Permiten crear calendarios, galerías de imágenes, Chats, Sistemas de envío de correo electrónico, sistemas de anuncios, sistemas de soporte a usuarios |

**MILLONES DE DESCARGAS, MILLONES DE WEBS DISTINTAS**

Vamos a centrarnos ahora en lo que permite mostrar una aplicación web a los usuarios. Por ejemplo, para un diario digital diremos que existe una parte denominada BackEnd donde escriben los articulistas y otra parte denominada FrontEnd que es la página web en sí del diario. Aplicaciones web que pueden servir para este propósito son Joomla, Drupal o WordPress. Si cientos de diarios digitales utilizan Drupal, por ejemplo, ¿Cómo consiguen tener un aspecto diferente unos de otros si el punto de partida es siempre el mismo?

La respuesta está en que estas herramientas incorporan código CSS avanzado mediante el que se crean aspectos distintos. La aplicación web suele contar con una parte para la gestión de contenidos mientras que otra parte denominada plantilla, template, theme, skin, etc. se encarga de controlar el aspecto.

El template o theme actúa como una piel sobre los contenidos. Fíjate cómo aplicando un filtro una fotografía puede cambiar:

© aprenderaprogramar.com, 2006-2029

Con CSS avanzado la idea es similar: aplicamos distintos tipos de fuentes, tamaños de fuentes, imágenes de fondo, colores, etc. para conseguir distintos aspectos. Fíjate en estas imágenes, que corresponden a themes o plantillas del gestor de contenidos Drupal.

Aquí vemos cómo usando CSS se pueden conseguir muy distintos aspectos. Esto ha permitido el éxito de gestores de contenidos como Joomla, Drupal o WordPress, con los que se puede crear desde una página dedicada al comercio electrónico hasta una web de un restaurante o un periódico digital. Gracias a los templates o themes también se puede cambiar el aspecto de páginas web cada cierto tiempo.

Si te fijas en las imágenes anteriores se puede argumentar que realmente no tienen el mismo contenido debajo. Efectivamente, en este caso no tienen el mismo contenido. Pero ten en cuenta que quizás el aspecto de una peluquería deba ser un poco diferente al aspecto de un restaurante.

En las siguientes imágenes te mostramos un mismo contenido con un cambio de theme:

© aprenderaprogramar.com, 2006-2029

© aprenderaprogramar.com, 2006-2029

En este caso el contenido sí es el mismo (excluyendo el espacio publicitario). Fíjate en los cambios: cambia la imagen de fondo que aparece en la cabecera de la web. Cambia el color de los elementos (letras, fondos), cambia la forma en que aparece el menú, cambia el espaciado entre líneas, etc. Pero la información que hay debajo es la misma.

Todo esto es posible gracias a CSS, la técnica y lenguaje que permite separar contenidos y presentación.

Realmente las páginas web tienen una gran variedad de aspectos no solo gracias a distintos colores e imágenes, sino también gracias a que las técnicas CSS permiten crear muy distintos aspectos.

Existen muchos estudios de diseño y programación donde se trabaja en la creación de templates o themes prediseñados. Hay muchos de distribución gratuita, pero la mayoría de los templates o themes de calidad son de pago (cosa lógica, ya que tienen un gran trabajo detrás).

Hemos querido con esta aproximación al uso de CSS en aplicaciones web remarcar la importancia que ha adquirido esta técnica en los desarrollos web. No vamos a entrar de momento en cuestiones relacionadas con templates o themes, sino a centrarnos en cuestiones básicas de CSS. La realidad en torno a los templates o themes de aplicaciones web es bastante compleja, ya que actualmente se tiende no sólo a permitir cambiar el aspecto de un mismo contenido, sino a permitir completamente la personalización de la presentación de páginas web (por ejemplo permitir usar dos módulos laterales y una columna central, o por el contrario dos columnas centrales sin módulos laterales, etc.). Esto ya supone el uso de programación al mismo tiempo que CSS, y también suele suponer la participación de distintos especialistas (diseñadores, maquetadores, expertos en CSS, programadores, etc.) para la creación de los templates o themes profesionales de las aplicaciones web.

Lo primero es lo primero, así que empecemos con los fundamentos de CSS.

**Próxima entrega:** CU01007D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## UN DOCUMENTO HTML BÁSICO

Para ver cómo CSS nos sirve para dotar de aspecto a un documento HTML vamos a partir de un código HTML que representa la estructura básica de una página web. En esta estructura la página web queda dividida en: cabecera con el título y mensaje breve, menú, texto con algunas imágenes, formulario de contacto y un pie de página con información sobre los autores o el copyryght.

Crea un documento HTML con un editor de texto como Notepad++ con el siguiente contenido:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!—Código base para el curso -->
<html>

<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta charset="utf-8">
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
</head>

<!-- Contenido de la página web -->
<body>

<!-- Cabecera de la página web -->
<div>
<h1>Portal web aprenderaprogramar.com</h1>
<h2>Didáctica y divulgación de la programación</h2>
</div>
<!-- Fin de la cabecera de la página web -->

<br />

<!-- Contenedor para la parte central -->
<div>

<!-- menu -->
<div>
<div>Menú</div>
<hr/>
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
</div>
<!-- fin menu -->
```

© aprenderaprogramar.com, 2006-2029

```
<!-- cuerpo -->
<div>

<!-- Texto con imágenes -->
<div>
<p>Aprender a programar es un objetivo que se plantea mucha gente y que no todos alcanzan.</p>
<p>Hay que tener claro que <a href="http://www.aprenderaprogramar.com">aprender programación</a> no es tarea de un día ni de una semana: aprender programación requiere al menos varios meses y, si hablamos de programación a nivel profesional, varios años. No queremos con esto desanimar a nadie: en un plazo de unos pocos días podemos estar haciendo nuestros primeros programas.</p>
<p>Puedes seguir uno de <a href="http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=57&Itemid=86">nuestros cursos</a> entre los varios disponibles. Cuando haya que utilizar un editor de textos recomendamos el uso de uno potente y sencillo como Notepad++, aunque son válidas otras alternativas como Crimson Editor.</p>
<a
href="http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=205&catid=57:herramientas-informaticas&Itemid=179">
<img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++, un útil editor de texto">
</a>
<img src="http://www.crimsoneditor.com/images/logo.jpg" alt="Crimson" title="Crimson, un editor de texto alternativo">

</div>
<!-- Fin del texto con imágenes -->
<br/>
<!-- Formulario de contacto -->
<form method="get" action="accion.html">
Si quieres contactar con nosotros envíanos este formulario relleno: <br /><br />
Nombre: <input type="text" name="nombre" /><br />
Apellidos: <input type="text" name="apellidos" /><br />
Dirección: <input type="text" name="direccion" /><br />
Correo electrónico: <input type="text" name="correo" /><br />
Mensaje: <textarea name="mensaje" cols=30 rows=2></textarea><br /><br />
<input type="submit" value="Enviar">
<input type="reset" value="Cancelar">
</form>
<!-- Fin del formulario de contacto -->

</div>
<!-- fin cuerpo -->
</div>
<!-- fin contenedor para la parte central -->
<br /> <br /> <br />
<!-- Pie de página o footer -->
<div>
<img src="http://lh5.ggpht.com/_PeVwghrmOec/TMkzEonRrcI/AAAAAAAAAHc/lxL8g0fTYtk/an_oliva_png.png" alt="logo apr" title="Logo aprenderaprogramar.com">
Copyright 2006-2038 aprenderaprogramar.com
</div>
<!-- Fin del pie de página o footer -->

</body>
<!-- Fin del contenido de la página web -->

</html>
```

© aprenderaprogramar.com, 2006-2029

El código anterior es HTML y lo usaremos a lo largo del curso para poner diferentes ejemplos, por lo que lo denominaremos "código base del curso". Para seguir este curso debes ser capaz de comprender todo el código HTML que hemos usado, su significado y sintaxis. Si no comprendes el código anterior no continues avanzando, dirígete a la web aprenderaprogramar.com y en la sección cursos busca el "Curso básico del programador web: HTML desde cero" y realízalo. Si no lo haces así no entenderás o no le sacarás todo el partido posible a este curso.

Visualiza el documento HTML en un navegador. Debes obtener un resultado similar a este (si te falla alguna imagen puedes cambiar las rutas y poner otra imagen que tú desees):

© aprenderaprogramar.com, 2006-2029

En este documento tenemos varios elementos como etiquetas de título h1 y h2, links, listas con elementos dentro de las listas, imágenes, formularios, botones, texto, etc. Todo ello nos va a servir para usar CSS y ver las posibilidades que nos ofrece CSS para dar formato a nuestras páginas web. En este curso nos vamos a centrar en tratar de aprender los aspectos más importantes de CSS y la lógica de CSS. El objetivo será saber hacer un buen diseño de CSS, un buen uso de CSS y comprender lo que hacemos. Por el contrario, no vamos a tratar de aprender o conocer todas las propiedades, posibilidades o instrucciones de CSS ya que si logramos comprender cómo funciona y su lógica, nos bastará con realizar búsquedas en internet para encontrar aquella propiedad o sintaxis que podamos necesitar en un momento dado. "Aprende a pescar, no te conformes con que te den peces".

**Próxima entrega:** CU01008D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## ORGANIZACIÓN CONCEPTUAL HTML

En la anterior entrega del curso vimos el código HTML correspondiente a una página web sencilla que constaba de cabecera con el título y mensaje breve, menú, texto con algunas imágenes, formulario de contacto y un pie de página con información sobre los autores o el copyryght. HTML podemos decir que define una organización conceptual: cada elemento está dentro de otro elemento o, en última instancia, dentro de la página web o documento HTML.

La organización conceptual que define HTML para nuestro ejemplo podríamos verla tal y como se refleja en el esquema 1.

Dentro de la organización que define HTML podemos señalar algunas cosas que resultan significativas. Por ejemplo, en distintas partes del documento HTML encontramos texto. No obstante, las etiquetas dentro de las que se encuentra el texto informan en algunos casos del significado de dicho texto. Por ejemplo el texto dentro de etiquetas <h1> ...</h1> sabemos que corresponde a un título prinicpal. El texto dentro de etiquetas <li> ...</li> sabemos que corresponde a un elemento de una lista. El texto dentro de elementos  ... sabemos que corresponde a un párrafo. Hay ocasiones en que el texto no aparece dentro de etiquetas con un significado explícito. Por ejemplo un texto dentro de  ...  es simplemente el texto dentro de un contenedor o espacio distinto de otros espacios, pero no tenemos más información acerca de su cometido.

Por otra parte en algunas partes aparece texto suelto sin estar delimitado por etiquetas, por ejemplo dentro del formulario aparece texto, sin estar incluido dentro de etiquetas específicas. Sabemos que dicho texto forma parte del formulario, pero nada más.

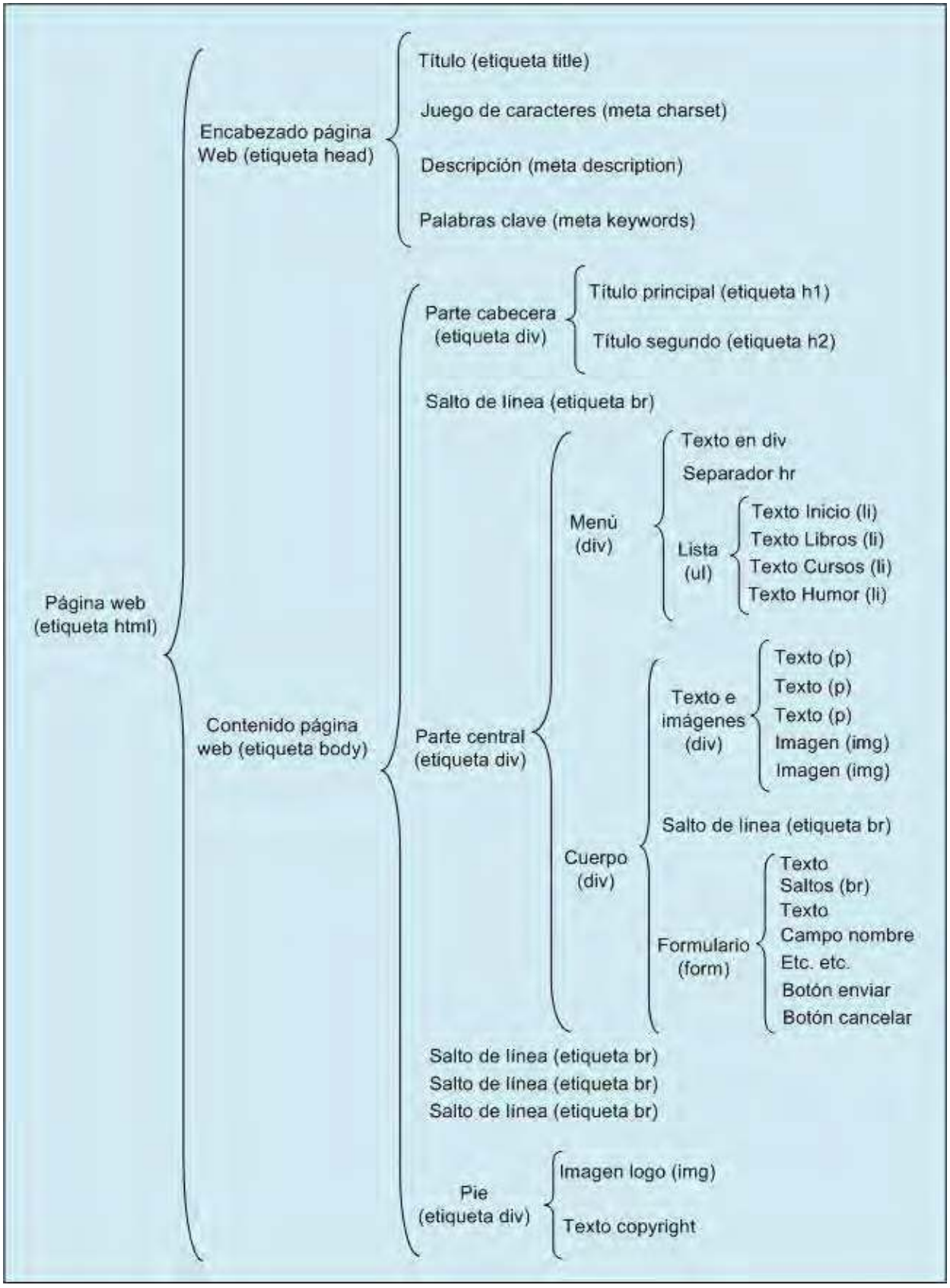
Otra cuestión que nos puede llamar la atención es que si bien la mayoría de las etiquetas tienen elemento de apertura y cierre (por ejemplo <h1> ...</h1>), algunas etiquetas se abren y cierran en un mismo elemento, como <br/>, mientras que otras se incluyen aparentemente como si solo tuvieran apertura, como es el caso de <img ..>. Todas estas particularidades de HTML debemos conocerlas.

## MODELO DE CAJAS

Ya hemos visto cómo conceptualmente HTML define una organización donde todos los elementos están dentro de un elemento matriz (el elemento body) y a su vez cada elemento puede tener otros elementos dentro de él y así sucesivamente.

A partir del HTML, los navegadores tienen que realizar la interpretación del código y mostrar en pantalla una página web. Ahora bien, a partir de un documento HTML habría distintas formas de presentar la información en pantalla. Por ejemplo, podrían mostrarse los sucesivos elementos de izquierda a derecha, o bien de arriba abajo, o bien de derecha a izquierda... ¿Qué hacen los navegadores? Los navegadores actúan según unas reglas predefinidas aceptadas por todos (o casi todos) según la cual cada elemento HTML se considera que está delimitado por un rectángulo o caja invisible. De ahí que se hable de "modelo de cajas" para la web.

© aprenderaprogramar.com, 2006-2029




Esquema 1. Organización conceptual basada en HTML

© aprenderaprogramar.com, 2006-2029

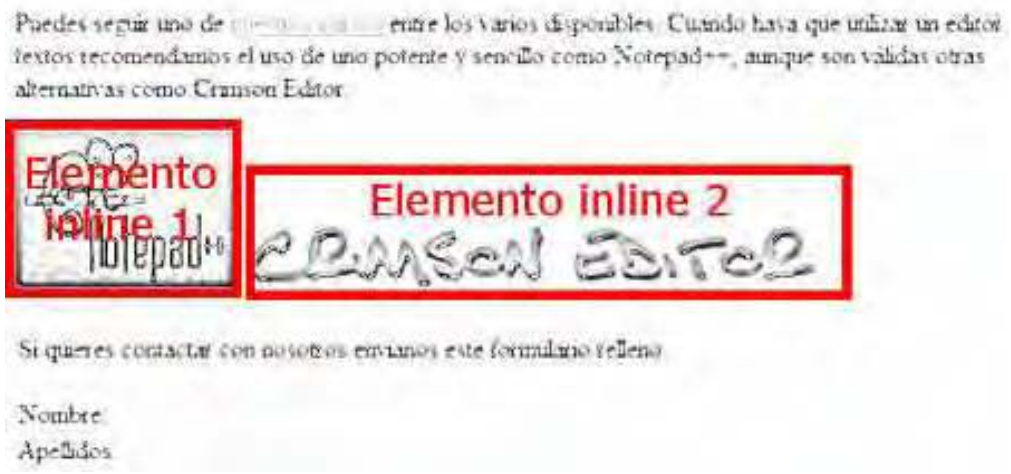
Cada caja puede ser o bien tipo <<block>> que podríamos considerar como "bloque a lo ancho" o bien tipo <<inline>> que podríamos considerar como "elemento dentro de una línea". Cada caja se coloca dentro de la pantalla de la siguiente manera:

- Una caja debajo de otra, si son elementos block del documento HTML del mismo rango o nivel en la jerarquía. Muchas de las etiquetas HTML se tratan por defecto como elementos block. Por ejemplo los elementos  ... son elementos block. Igualmente son elementos block los formularios <form> ...</form> o las listas <ul> ...</ul>.

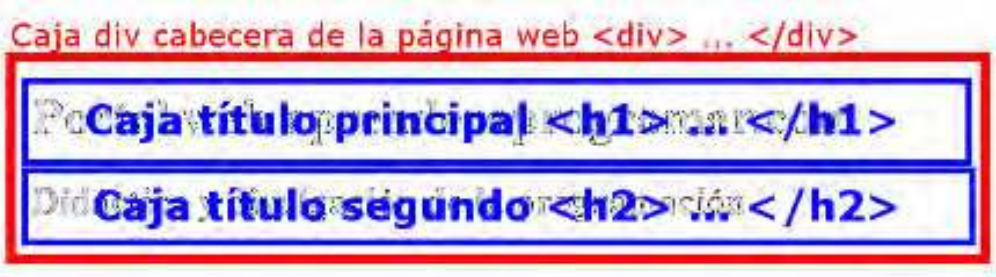
  En el siguiente esquema reflejamos algunos elementos block dentro de nuestro ejemplo.

Si te fijas, las cajas que hemos representado son las divisiones principales dentro de "la caja que lo engloba todo" que es la definida por <body> ...</body>.

- Una caja al lado de otra, de izquierda a derecha de acuerdo con el orden con que aparezcan en el documento HTML, si son elementos inline del documento HTML del mismo rango o nivel en la jerarquía. Los elementos se mantienen uno al lado de otro (excepto en el caso de que ya no haya espacio para ubicarlos, en cuyo caso pasan a la siguiente línea). Algunas de las etiquetas HTML se tratan por defecto como elementos inline. Por ejemplo los elementos <img> son elementos inline. Gráficamente en nuestro ejemplo esto se visualiza porque distintos elementos <img> se colocan uno al lado de otro y no uno debajo de otro como harían elementos block, como vemos a continuación.

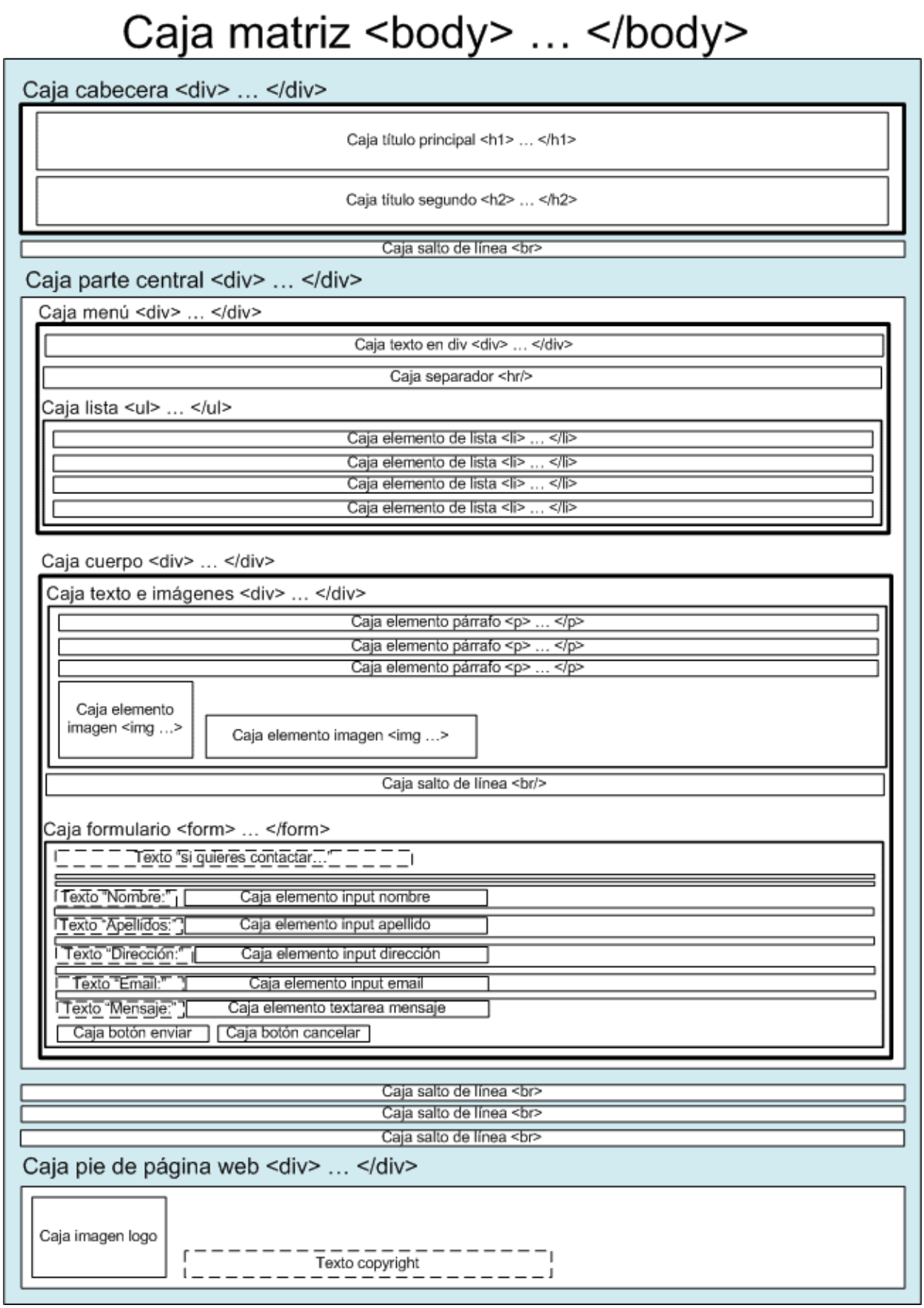



- Una caja dentro de otra, siendo la caja interior la de menor rango o jerarquía. Puede haber varias cajas dentro de cada caja, según se defina en el documento HTML. Por ejemplo dentro de la caja div que define la cabecera de nuestra web de ejemplo hay dos cajas, una correspondiente al título principal <h1>...</h1> y otra correspondiente al título segundo <h2> ...</h2>.




El siguiente esquema reflejaría el modelo de cajas del documento HTML de forma similar a como lo construye un navegador web como pueda ser Internet Explorer, Google Chrome o Mozilla Firefox, Safari, etc.

© aprenderaprogramar.com, 2006-2029




Esquema 2. Modelo de cajas CSS

© aprenderaprogramar.com, 2006-2029

En este esquema comprobamos que una página web puede tener gran complejidad en su organización estructural y en su representación con modelo de cajas.

Hemos representado las cajas en general delimitadas con una línea contínua (que en el navegador se vuelve invisible) excepto aquellos elementos de texto que no están delimitados por etiquetas específicas, que constituyen casos especiales que representamos con línea discontínua.

Como veremos a lo largo del curso, CSS nos servirá para aplicar estilos (bordes, fondos, tipos de letra, interlineado, etc.) a todas las cajas de un mismo tipo si lo deseamos, o también a cajas concretas para las que deseemos tener un tratamiento especial.

**EJERCICIO**

Analiza el siguiente código HTML y crea dos esquemas. Un esquema de organización utilizando llaves como hemos visto anteriormente, y otro esquema que refleje las cajas que intervienen en el documento HTML, siguiendo el ejemplo visto anteriormente.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
<title>Portal web - aprenderaprogramar.com</title>
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<meta charset="utf-8">
</head>
 <body>
 <p><a href="principal.html" title="Página principal" >Ir a la pagina principal</a></p>
<h1>Novedades</h1>
<p>Aquí presentamos las novedades del sitio.</p>
<h3>Lanzamos el producto X-FASHION</h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto T-MOTION</p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</body>
</html>
```

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01009D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

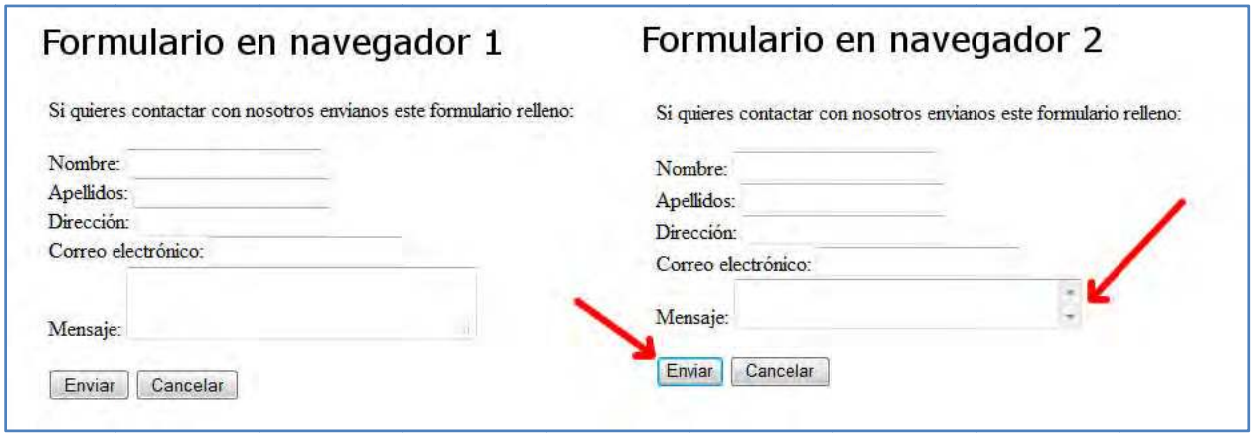
© aprenderaprogramar.com, 2006-2029

**ESTILOS POR DEFECTO**

Podemos dar formato a nuestros documentos HTML de varias maneras: incluyendo propiedades CSS en las propias líneas de HTML (aplicación de estilos en línea), en la parte inicial del documento HTML (aplicación de estilos interna) o en un archivo de extensión .css independiente del archivo HTML (aplicación de estilos externa).

Antes de escribir nuestro primer código CSS remarquemos una cuestión importante: al crear un documento HTML este ya posee un estilo. ¿Cuál? El estilo por defecto que aplican los navegadores. Este estilo por defecto suele comprender un tipo de letra, color negro para el texto y color blanco de fondo para el texto. El estilo por defecto para los enlaces (links) suele ser color azul y subrayado, aunque esto puede variar según el navegador que utilicemos. Es importante tener esto en cuenta porque nos podemos encontrar que ciertos elementos se visualicen de distinta manera en dos navegadores diferentes (por ejemplo Internet Explorer y Mozilla Firefox) debido a que un navegador aplique un diferente estilo por defecto.

Considera el formulario en el ejemplo que venimos usando para el desarrollo del curso (código base del curso). Fíjate en la siguiente imagen cómo se visualiza el mismo código HTML en dos navegadores distintos sin haber aplicado estilos:




Podemos señalar algunas diferencias:

a) En el navegador 2 el botón "Enviar" aparece remarcado con un borde azul y es un poco más pequeño que en el navegador 1.
b) En el navegador 2 la visualización del textarea correspondiente al mensaje incluye un scroll en el lateral derecho, que no existe en el navegador 1.
c) Otras: por ejemplo la altura de la caja del textarea es más pequeña en el navegador 2 que en el navegador 1.

© aprenderaprogramar.com, 2006-2029

Aquí nos encontramos con algo a lo que debemos acostumbrarnos como desarrolladores web. En general, no es posible (o quizás sí sea posible, pero llevaría demasiado tiempo y sería demasiado costoso) conseguir "exactamente" la misma visualización en distintos navegadores web.

Muchas personas pasan horas tratando de "cuadrar" con exactitud los elementos de una página web y muchas veces este trabajo, o parte de este trabajo, carece de sentido, ya que al cambiar de navegador (o de sistema operativo con el mismo navegador) todo lo que se había cuadrado puede aparecer descuadrado, o al menos no exactamente como se pensaba, dando lugar a grandes decepciones.

Nuestra recomendación es no obsesionarse con pequeños detalles y, cuando se trate de desarrollos importantes, probar las webs en distintos navegadores y ordenadores. Para este curso empezaremos trabajando con un solo navegador ya que para el aprendizaje nos resulta suficiente. Nosotros usaremos Mozilla Firefox, pero puedes usar otro si lo deseas. Cuando hayamos avanzado iremos explicando algunos detalles o características específicas de los distintos navegadores. Cuando trates de visualizar las páginas web que construiremos durante el curso ten en cuenta que puedes obtener visualizaciones distintas de aquellas que mostramos nosotros debido a que tu navegador no interprete exactamente de la misma manera el código. No te preocupes ahora por los pequeños detalles, trata de aprender los conceptos e ideas que hay en torno a CSS, más adelante ya habrá tiempo de definir cómo se debe manejar la problemática del distinto comportamiento entre navegadores.

**ESTILOS EN LÍNEA**

Una de las formas más simples e intuitivas de dotar de estilos al código HTML es usando el atributo style que admiten la mayoría de las etiquetas HTML.

Supón que sobre el código de ejemplo que estamos utilizando en el curso deseamos que el texto de los elementos del menú se muestre de color verde y el texto de los párrafos de color azul. Para aplicar un estilo en línea utilizaremos esta sintaxis:

<nombreDeEtiqueta style = "propiedadCSS: valorEstablecido;" > ...</nombreDeEtiqueta>

En nuestro caso para un párrafo usaríamos  ...

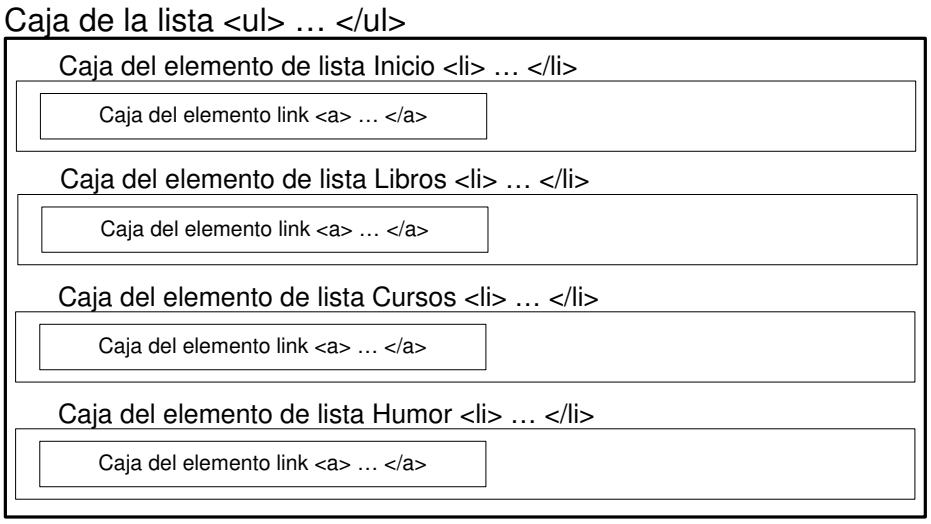
Para aplicar el color verde a los elementos del menú, que están en una lista, podemos probar a aplicarle el atributo *style* y la propiedad *color* a la etiqueta <ul> ...</ul>. El código quedaría como sigue:

```
<ul style = "color: green;">
<li><a href="#" >Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
```

© aprenderaprogramar.com, 2006-2029

El resultado obtenido lo vemos a continuación:

Podremos comprobar que no hemos obtenido el efecto deseado. Queríamos poner el texto de los elementos del menú en color verde y sin embargo continúa en color azul. Sí podemos observar, sin embargo, que las viñetas o iconos circulares que aparecen en el lateral izquierdo han cambiado de color negro a color verde. ¿Cómo explicamos este comportamiento? Tenemos que pensar en el modelo de cajas para comprender qué es lo que ocurre. El modelo de cajas para los elementos del menú sería el siguiente:




En este esquema tenemos cajas en tres niveles: la caja más exterior correspondiente a la lista, las cajas dentro de la lista correspondientes a cada uno de los elementos dentro de la lista, y las cajas más interiores correpondientes a las etiquetas <a> ...<a> que definen los link.

El navegador actúa aplicando estilos desde los niveles más exteriores hacia los niveles más interiores, de forma que el estilo que se ve cuando existen varios es el más interior entre todos los posibles estilos que afectan a un elemento.

En este caso, toda la lista se establece con color verde, lo cual afecta a las viñetas y al texto. En las etiquetas <li> no existe estilo propio que contradiga el color verde. Sin embargo, los elementos link tienen un estilo propio, un estilo que en este caso es incorporado por defecto por el navegador, según el cual estos elementos aparecen en color azul y subrayados. Este es el último estilo que lee el

© aprenderaprogramar.com, 2006-2029

navegador y el que aplica a la caja de los elementos <a> y esta caja al ser la más interna es la que se visualiza, mostrándose el texto en color azul al estar dentro de las etiquetas <a> ...</a>.

Para resolver este conflicto añadiremos estilos que modifican el color para los links:

```
<ul style = "color: green;">
<li><a href="#" style = "color: green;">Inicio</a></li>
<li> <a href="libros.html" style = "color: green;">Libros de programación</a> </li>
<li> <a href="cursos.html" style = "color: green;">Cursos de programación</a> </li>
<li> <a href="humor.html" style = "color: green;">Humor informático</a> </li>
</ul>
```

Ahora sí hemos conseguido el efecto deseado.

Fíjate en que hemos mantenido el estilo aplicado a la etiqueta <ul> ... </ul> par que el color de las viñetas se mantenga en verde. Si no aplicáramos ese estilo, el estilo aplicado sería el estilo por defecto según el cual las viñetas se mostrarían en negro.

Nos planteamos ahora dejar las viñetas en color rojo y el texto del menú en color verde pero sin subrayado. Para ello tendremos que añadir una propiedad al link que elimine el estilo subrayado que por defecto incorpora el navegador. Esta propiedad será text-decoration cuyos posibles valores son none (ninguno), underline (subrayado), overline (línea superior), line-through (tachado). El código será el siguiente:

```
<ul style="color: red;">
<li><a href="#" style="color:green; text-decoration: none;">Inicio</a></li>
<li><a href="libros.html" style="color:green; text-decoration: none;">Libros programación</a> </li>
<li><a href="cursos.html" style="color:green; text-decoration: none;">Cursos programación</a> </li>
<li><a href="humor.html" style = "color:green; text-decoration: none;">Humor informático</a> </li>
</ul>
```

© aprenderaprogramar.com, 2006-2029

Y el resultado:

Decimos que hemos sobreescrito una propiedad CSS, en este caso la propiedad de subrayado de los links, reemplazándola por una nueva propiedad.

Hay una cosa que llama la atención: vemos que el código CSS está "entremezclado" o "embebido" dentro del código HTML. Por ejemplo en <ul style="color: red;"> vemos que HTML y CSS están íntimamente relacionados, tanto que resulta difícil distinguir qué es HTML y qué es CSS. Esta es una característica a la que debes acostumbrarte, en los desarrollos web se entremezclan distintos lenguajes o metalengajes. Podríamos decir que HTML y CSS son sublenguajes de un supralenguaje: el lenguaje de los desarrollos web.

**EJERCICIO**

A partir del siguiente código:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
<title>Portal web - aprenderaprogramar.com</title>
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<meta charset="utf-8">
</head>
 <body>
 <p><a href="principal.html" title="Página principal" >Ir a la pagina principal</a></p>
<h1>Novedades</h1>
<p>Aquí presentamos las novedades del sitio.</p>
<h3>Lanzamos el producto X-FASHION</h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto T-MOTION</p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</body>
</html>
```

Modifica el código HTML anterior para cumplir con estos requisitos mediante la aplicación de estilos en línea:

© aprenderaprogramar.com, 2006-2029

a) La etiqueta h1 debe mostrar su texto en color rojo.
b) La etiqueta h3 con el texto relativo a X-FASHION debe mostrar su texto en color verde.
c) La etiqueta h3 con el texto relativo a X-MOTION debe mostrar su texto en color azul.
d) Todos los párrafos deben mostrar su texto en color brown (marrón).

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01010D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**FORMAS DE APLICAR CSS**

Ya hemos visto que existen estilos por defecto que aplican automáticamente los navegadores web cuando no hay estilos especificados, así como que podemos embeber código CSS en las propias etiquetas de HTML, "en línea". Vamos a ver ahora dos formas adicionales de aplicar estilos CSS: en la parte inicial del documento HTML (aplicación de estilos interna o CSS interno) o en un archivo de extensión .css independiente del archivo HTML (aplicación de estilos externa o CSS externo).

Volvemos a lo que hemos denominado código HTML base para el desarrollo del curso, código HTML donde no se aplican estilos.

En el epígrafe anterior vimos cómo aplicábamos estilos CSS en línea con este ejemplo:

```
<ul style="color: red;">
<li><a href="#" style="color:green; text-decoration: none;">Inicio</a></li>
<li><a href="libros.html" style="color:green; text-decoration: none;">Libros programación</a> </li>
<li><a href="cursos.html" style="color:green; text-decoration: none;">Cursos programación</a> </li>
<li><a href="humor.html" style = "color:green; text-decoration: none;">Humor informático</a> </li>
</ul>
```

**CSS INTERNO**

Vamos a ver ahora cómo podemos definir esos estilos en la cabecera del documento HTML usando la siguiente sintaxis:

```
<head>
...
<style type="text/css">
elementoAfectadoPorElEstilo {
propiedad1ParaEseTipoDeElementos:valor;
propiedad2ParaEseTipoDeElementos:valor;
propiedad3ParaEseTipoDeElementos:valor;
...
propiedadnParaEseTipoDeElementos:valor;
}
</style>
</head>
```

Las diferentes propiedades y valores se pueden poner en una misma línea o en distintas líneas según se prefiera (siempre separados mediante punto y coma).

© aprenderaprogramar.com, 2006-2029

Dentro de las etiquetas <head> ... </head> incluiremos una etiqueta de apertura de declaración de estilos <style type="text/css">, a continuación colocaremos la definición de tantos estilos como deseemos y cerraremos la declaración con </style>. En HTML 5 no es necesario especificar *type ="text/css"* pero de momento seguimos recomendando que se use esta sintaxis.

En nuestro código de ejemplo el cambio de los elementos del menú para que tengan el texto color verde y el icono o símbolo de viñeta de color rojo se haría de esta forma:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!—Código ejemplo para el curso -->
<html>

<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta charset="utf-8">
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<style type="text/css">
ul {color:red;}
a {color:green; text-decoration: none;}
</style>
</head>
</head>

<!-- Contenido de la página web -->
<body>

<!-- Cabecera de la página web -->
<div>
<h1>Portal web aprenderaprogramar.com</h1>
<h2>Didáctica y divulgación de la programación</h2>
</div>
<!-- Fin de la cabecera de la página web -->
<br />
<!-- Contenedor para la parte central -->
<div>
<!-- menu -->
<div>
<div>Menú</div>
<hr/>
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
</div>
<!-- fin menu -->
<!—Aquí el resto del código del ejemplo -->

</html>
```

© aprenderaprogramar.com, 2006-2029

Puedes comprobar que hemos usado las mismas propiedades CSS y los mismos valores que usábamos en línea.

Visualiza la página en tu navegador. El resultado para el menú será el mismo que cuando aplicamos CSS en línea:

No obstante podrás comprobar que hay otros elementos de la página web que también se ven afectados, por ejemplo los links presentes en el texto. Esto se debe a que hay una diferencia importante entre aplicar estilos CSS en línea y aplicarlos como CSS interno en la cabecera <head> ... </head> del documento HTML. Al aplicar estilos en línea, tenemos que repetir la aplicación de estilos en cada una de las líneas que queramos modificar y en cada ocasión afectamos a **únicamente una línea**. Con la aplicación de estilos interna nos basta con declarar una vez el estilo y el tipo de elemento al que se aplica, y automáticamente se aplicará **a todos** los elementos de ese tipo existentes dentro de la página web. Esto permite que en una página web de gran extensión nos ahorremos tener que escribir múltiples veces la definición de estilos (una en cada línea), ya que una sola declaración inicial nos bastará, lo cual es una ventaja bastante evidente.

Sin embargo ahora surge una cuestión adicional: es posible que tengamos dos o más listas de tipo <ul> y en algunos casos, por ejemplo menús, queramos aplicar un estilo, y en otros casos, por ejemplo listas de elementos dentro de un artículo periodístico, queramos aplicar otro estilo. También es posible que deseemos que los elementos del menú sean links con texto verde y sin subrayado, pero queramos mantener el resto de links como texto azul con subrayado. Tal y como hemos hecho la definición de estilos no conseguimos hacer esto, ya que estamos modificando **todas** las listas y **todos** los links presentes en el documento. CSS permite resolver satisfactoriamente esta problemática para aplicar estilos específicamente allí donde queremos. Veremos cómo próximamente.

**CSS EXTERNO**

Aunque el CSS interno nos permite unificar en una declaración todos los estilos para un archivo html, seguimos teniendo el problema de tener que repetir la definición de estilos en la cabecera de cada uno de los archivos html de nuestro desarrollo web. Si el desarrollo tiene pocos archivos quizás sea menos problemático, pero cuando el desarrollo tiene cientos o miles de archivos sí se convierte en un

© aprenderaprogramar.com, 2006-2029

verdadero problema, ya que cada vez que introdujéramos cambios habría que hacerlo en los cientos o miles de archivos de que constara el desarrollo.

Para solventar esta solución se ideó la declaración externa de CSS, basada en declarar los estilos CSS en un archivo independiente que puede servir como referente para dotar de estilos a decenas, cientos o miles de archivos html, que únicamente deberán invocar el nombre de archivo utilizando una sintaxis específica. De este modo un cambio en los estilos habrá que hacerlo únicamente en el archivo de estilos correspondiente, lo cual supone una gran ventaja. Incluso un cambio completo de los estilos de una página web se puede conseguir simplemente sustituyendo el archivo correspondiente.

Vamos a generar un archivo de estilos independiente. Para ello abre Notepad++ (o el editor de texto que estés usando) y crea un archivo con el siguiente contenido:

```
/* Comentario en CSS estilos aprenderaprogramar.com*/

ul {color:red;}
a {color:green; text-decoration: none;}
```

Seguimos usando las mismas propiedades CSS y los mismos valores que hemos usado en la forma CSS en línea y en la forma CSS interna.

Los contenidos que se encuentren entre los símbolos /* ... */ serán considerados comentarios y se pueden usar para introducir información del autor del archivo, versión, etc. así como para escribir aquellas aclaraciones sobre los estilos que se consideren necesarias. Un comentario puede comprender una o varias líneas según se desee.

Elige Guardar Como... y guarda el archivo con un nombre y extensión css, por ejemplo estilos.css. Asegúrate que la extensión del archivo sea css. No es válido si no tiene esta extensión.

En nuestro archivo HTML eliminaremos la definición de estilos interna y dejaremos sólo la invocación al archivo escrita con la siguiente sintaxis:

```
<head>
...

<link rel="stylesheet" type="text/css" href="rutaDelArchivo.css">
</head>
```

link es una etiqueta que se usa en HTML para establecer vínculos entre un documento HTML y otros recursos como una imagen de icono o una hoja de estilos CSS. En este caso se indica con la propiedad rel (relación o relationship) que el documento HTML debe usar los estilos definidos como texto/css en un archivo con el nombre indicado. Ten en cuenta que si como ruta de archivo indicas simplemente el

© aprenderaprogramar.com, 2006-2029

nombre del archivo, éste debe encontrarse en la misma carpeta en que se encuentre el documento HTML. Si el archivo CSS se encuentra en otra carpeta deberás indicarlo escribiendo una URL completa u bien una URL relativa que determine la ruta.

Nosotros ubicaremos de momento el archivo CSS en la misma carpeta que el archivo HTML. En el archivo HTML tendremos lo siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!—Código ejemplo para el curso -->
<html>

<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta charset="utf-8">
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<link rel="stylesheet" type="text/css" href="estilos.css">
</head>

<!—Aquí el resto del código ... -->
...
```

Ahora procedemos a visualizar el archivo HTML en nuestro navegador. Si se está cargando la hoja de estilos correctamente deberemos obtener el mismo resultado que habíamos obtenido con la aplicación de estilos interna. Prueba a hacer distintos cambios en el archivo css y visualiza los resultados.

**¿QUÉ TIPO DE CSS USAR?**

En la siguiente tabla hacemos un resumen de las características de los distintos tipos de CSS que hemos visto hasta el momento:

| DECLARACIÓN CSS | ARCHIVOS AFECTADOS | CSS SE APLICA A |
|---|---|---|
| **En línea** | Uno, aquel donde se realiza la declaración | Una línea de código |
| **Interna** | Uno, aquel en cuya cabecera se realiza la declaración | Todos los elementos del archivo en los que resulte de aplicación el estilo |
| **Externa** | Todos los archivos que invoquen el archivo css | Todos los elementos de los archivos afectados en que resulte de aplicación el estilo |

© aprenderaprogramar.com, 2006-2029

¿Cuál de estas formas de aplicar CSS es mejor? ¿Cuál usar? En primer lugar cabe hacer una reflexión sobre qué forma de aplicar CSS hará nuestros desarrollos web más fácilmente mantenibles y aptos para ser modificados con poco esfuerzo. Parece claro que es la declaración externa la que mejor independiza los estilos del contenido y la que menor número de modificaciones implicará en general. Por ello es la forma de trabajar con CSS a la que debemos acostumbrarnos.
No obstante, cuando trabajes con desarrollos web comprobarás que también son de uso frecuente la aplicación de estilos en línea o interna. En algunos casos esto es debido a desconocimiento de la persona que realiza el desarrollo o a la forma de funcionamiento del programa con el que se ha creado la página web, pero en otros casos obedece a que se ha querido hacer así.

En ocasiones se opta por hacer modificaciones en línea porque se buscan efectos puntuales que sólo se quieren aplicar en un punto concreto y en ningún otro. En ocasiones se opta por hacer definiciones CSS internas porque se quiere afectar muy puntualmente a un archivo y a ningún otro. Y en otros casos, se usan los estilos en línea o internos simplemente "por las prisas" o "por ser lo más rápido".

A efectos del navegador, una declaración en línea tiene precedencia sobre una declaración interna, y a su vez una declaración interna tiene precedencia sobre una declaración externa. Por tanto podríamos tener CSS sobreescrito varias veces: la declaración externa puede ser sobreescrita por la interna, y ésta a su vez por la en línea. El navegador aplicará el orden:

**Declaración en línea > Declaración interna > Declaración externa**

Podría ocurrir que los estilos que visualicemos en el navegador estén definidos "a trozos" de modo que parte de lo que se visualiza se debe a las declaraciones CSS externas, parte a las declaraciones CSS internas y parte a declaraciones en línea. Esto en general será indeseable, ya que hará difícil de mantener el desarrollo web, no se sabrá con certeza cómo se generan los estilos y el desarrollo puede terminar convirtiéndose en un caos difícil de mantener. También puede dar lugar a visualizaciones incorrectas o erróneas.

A modo de resumen y como recomendación: mantén los estilos de forma externa, lo más ordenadamente posible y sin utilizar redefiniciones internas ni en línea. Usa este tipo de definiciones (interna o en línea) exclusivamente cuando resulte necesario y comenta adecuadamente el código. Trabajar ordenadamente será algo que a la larga te resultará ventajoso, tanto a ti mismo como a otras personas que tengan que trabajar en desarrollos donde tú hayas participado.

**EJERCICIO 1**

Modifica el código HTML mostrado a continuación para cumplir con estos requisitos mediante la aplicación de estilos internos:

a) Todas las etiquetas h1 deben mostrar su texto en color rojo.
b) Todas las etiquetas h3 deben mostrar su texto en color verde.
c) Todos los párrafos deben mostrar su texto en color brown (marrón).

© aprenderaprogramar.com, 2006-2029

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
<title>Portal web - aprenderaprogramar.com</title>
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<meta charset="utf-8">
</head>
 <body>
 <p><a href="principal.html" title="Página principal" >Ir a la pagina principal</a></p>
<h1>Novedades</h1>
<p>Aquí presentamos las novedades del sitio.</p>
<h3>Lanzamos el producto X-FASHION</h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto T-MOTION</p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</body>
</html>
```

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**EJERCICIO 2**

Modifica el código HTML del ejercicio anterior para cumplir con estos requisitos mediante la aplicación de CSS externo definido en un archivo independiente, donde debes incluir al menos un comentario:

a) Todas las etiquetas h1 deben mostrar su texto en color azul.
b) Todas las etiquetas h3 deben mostrar su texto en color orange (naranja).
c) Todos los párrafos deben mostrar su texto en color brown (marrón).

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01011D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**SELECTORES CSS**

Hemos estudiado cómo aplicar estilos CSS a todas las etiquetas HTML de un tipo, por ejemplo a todos los párrafos, todas las listas, todos los links, etc. Pero en numerosas ocasiones no queremos que todas las etiquetas de un tipo tengan un mismo estilo, sino aplicar distintos estilos a una etiqueta o a una parte de una página web. Para ello además del selector de etiqueta o palabra clave que ya conocemos existen otros selectores, entre los más usados tenemos los selectores por id y los selectores de clase.

Recordemos primero los selectores de etiqueta o palabra clave tipo etiqueta {propiedad:valor;}:

ul {color:red;} suponía aplicar color rojo al texto de todas las listas de tipo ul del documento HTML, es decir, a todas las etiquetas <ul> ...</ul>.

a {color:green; text-decoration: none;} suponía aplicar color verde y sin subrayado a todos los links del documento HTML, es decir, a todas las etiquetas <a> ...</a>.

De la misma forma se pueden usar como selector otras etiquetas HTML como div { }, span { }, p { }, etc. pero suponiendo que se apliquen a todos los elementos de ese tipo.

**SELECTOR POR ID**

Trataremos ahora de aplicar estilos a sólo una parte de una página web o documento HTML. Esto implica dos cosas:

a) En el documento HTML, habrá que identificar la parte del documento a la que queremos aplicar esos estilos incluyendo un atributo para la etiqueta que delimita el fragmento de código donde queremos aplicar estilos. Por ejemplo si queremos aplicar estilos dentro de un contenedor div la sintaxis sería  ... . Si quisiéramos aplicar estilos sólo a un párrafo la sintaxis sería  ...

b) En la declaración de estilos tendremos que declarar los estilos a aplicar a ese fragmento de código realizando la declaración usando esta sintaxis.

```
#nombreIdentificativoElegido {
propiedad1ParaEseTipoDeElementos:valor;
propiedad2ParaEseTipoDeElementos:valor;
propiedad3ParaEseTipoDeElementos:valor;
...
propiedadnParaEseTipoDeElementos:valor;
}
```

Las propiedades se pueden definir en una sola línea si se prefiere así, el único requisito es separarlas con punto y coma.

© aprenderaprogramar.com, 2006-2029

Con un ejemplo lo veremos más claro. Partimos del código base HTML que estamos usando para el curso, en el cual tenemos un menú. Vamos a elegir un nombre para identificar ese menú. Podríamos elegir como nombre <<menu>> (usaremos nombres sin tildes). Ahora mismo solo tenemos un menú, pero en el futuro es posible que existan otros, por eso vamos a preferir elegir como nombre <<menu1>>. En el documento HTML vamos a identificar el menú con el atributo id aplicado al contenedor div dentro del cual está el menú. El código es este:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!—Código ejemplo para el curso -->
<html>

<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta charset="utf-8">
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<link rel="stylesheet" type="text/css" href="rutaDelArchivo.css">
</head>

<!-- Contenido de la página web -->
<body>

<!-- Cabecera de la página web -->
<div>
<h1>Portal web aprenderaprogramar.com</h1>
<h2>Didáctica y divulgación de la programación</h2>
</div>
<!-- Fin de la cabecera de la página web -->
<br />
<!-- Contenedor para la parte central -->
<div>
<!-- menu -->
<div id="menu1">
<div>Menú</div>
<hr/>
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
</div>
<!-- fin menu -->
<!—Aquí el resto del código del ejemplo -->

</html>
```

Hemos incluido  en el contenedor div que delimita nuestro menú.

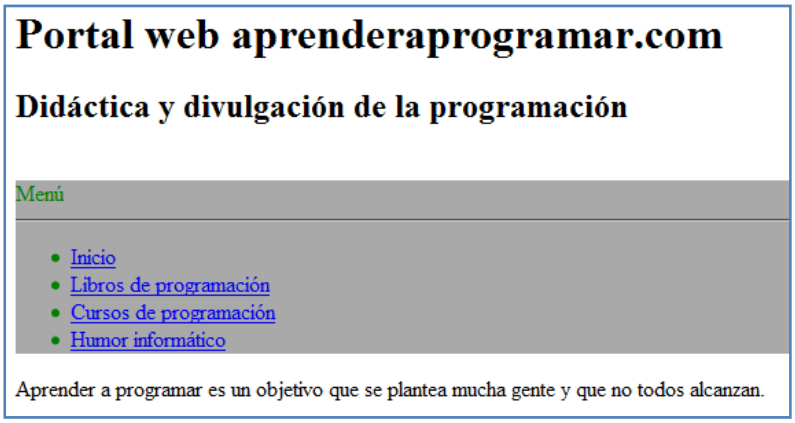
© aprenderaprogramar.com, 2006-2029

En el archivo css externo, en nuestro caso denominado estilos.css, vamos a definir que ese fragmento de página web (el menú) tenga el texto verde y fondo gris. Para ello escribiremos lo siguiente:

```
/* Curso CSS estilos aprenderaprogramar.com*/

#menu1 {
color:green;
background-color: DarkGray;
}
```

Si lo preferimos podemos escribir la declaración en una sola línea, obteniendo el mismo resultado. Es decir, podemos escribir #menu1 {color:green; background-color: DarkGray; }. Esto resulta más compacto y ahorra espacio, pero también puede ser menos claro a la hora de leer. Algunos programadores o diseñadores prefieren el estilo compacto y otros diferenciando líneas.

El resultado obtenido será similar a este:



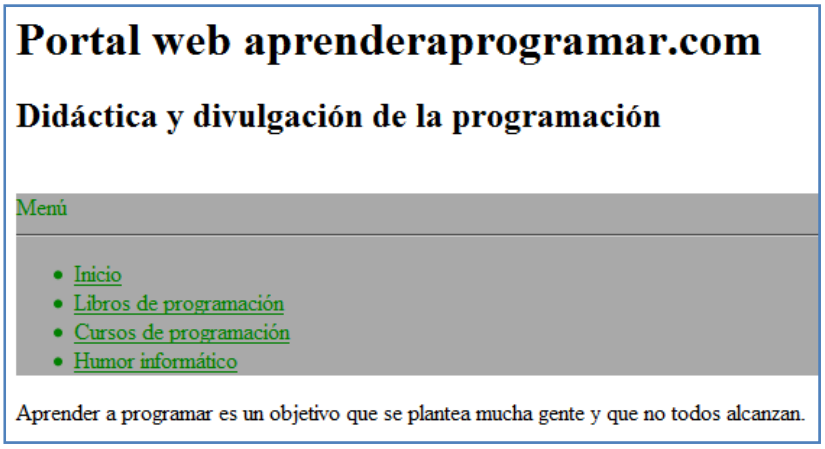

Analicemos ahora el resultado obtenido: se ha aplicado el fondo gris a la caja que define el contenedor div. Se ha aplicado color verde al texto y a las viñetas de la lista existente en el menú. Sin embargo, los links del menú mantienen su color azul y subrayado, a pesar de que hemos indicado que el color verde debería de aplicarse a todo el contenedor. ¿Por qué ocurre esto? CSS no aplica la misma importancia a todas las reglas o estilos que se especifican. En este caso la regla o estilo que hemos definido no supera en la escala de importancia CSS a la regla por defecto según la cual los links son de color azul y subrayado.

Para resolver este conflicto hemos de especificar que queremos que los elementos <a> ...</a> (links) dentro del bloque de código con identificador menu1 se muestren aplicando la regla de estilo definida para el menú. Modifica el archivo css escribiendo el siguiente código y visualiza la página web:

```
/* Curso CSS estilos aprenderaprogramar.com*/

#menu1, #menu1 a {color:green; background-color: DarkGray;}
```

© aprenderaprogramar.com, 2006-2029

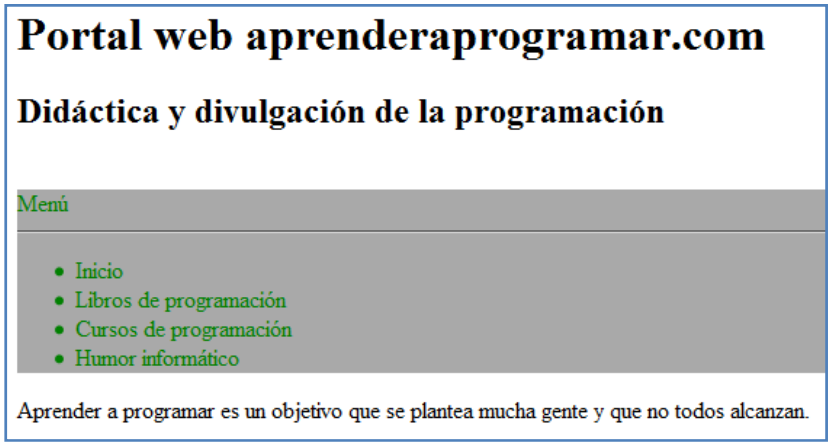
Mediante la sintaxis #nombreIdentificativoElegido elementoAfectadoPorElEstilo {..} estamos especificando "explícitamente" que deseamos aplicar esa regla de estilo al elemento indicado (en nuestro caso los links, <a> ...</a>) dentro de un bloque de código definido por un identificador. De esta forma el navegador le concede preferencia o mayor importancia al estilo definido que al estilo por defecto de los links. El resultado en este caso es que se muestra el texto de los links en color verde (pero se sigue manteniendo el subrayado porque no hemos sobreescrito esta propiedad de los links):




Vamos a especificar ahora que los links del menú no aparezcan subrayados. Para ello modificaremos nuestro archivo css dejándolo con esta línea:

```
#menu1, #menu1 a { color:green; background-color:DarkGray; text-decoration:none;}
```

Ahora sí tenemos el menú en color verde y sin subrayado, ya que hemos sobreescrito la propiedad text-decoration que estaba dando lugar a la aparición del subrayado:




Es interesante de lo visto hasta ahora razonar cómo CSS trabaja con unos órdenes de precedencia o jerarquía de reglas CSS. No te preocupes de conocerlos con exactitud, ya que a lo largo del curso y a medida que vayas cogiendo experiencia irás aprendiendo más al respecto. También ten en cuenta que los órdenes de precedencia pueden cambiar según los navegadores o según las versiones CSS. Por tanto más que saber con exactitud los órdenes de precedencia nos interesa el ser capaces de interpretar por qué ocurren las cosas y la forma de trabajar de CSS y su filosofía.

© aprenderaprogramar.com, 2006-2029

**APLICAR ESTILOS DIFERENCIADOS DENTRO DE UN SELECTOR ID**

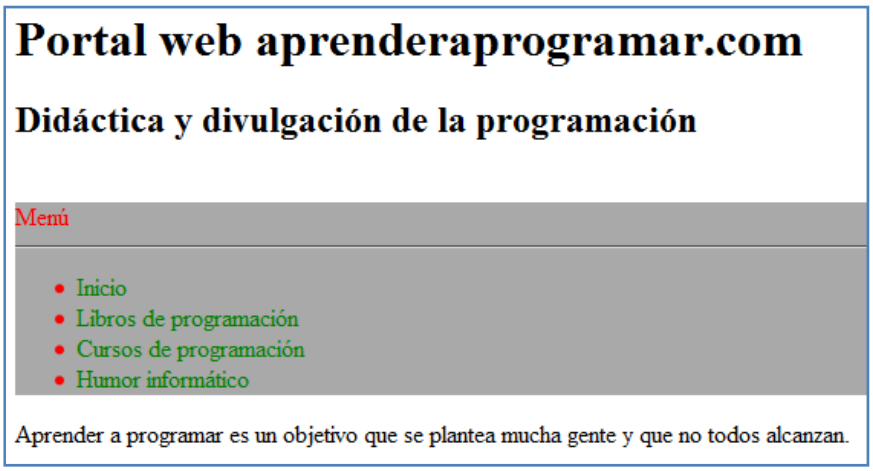
El código que hemos visto anteriormente se puede escribir también de esta manera:

```
/* Curso CSS estilos aprenderaprogramar.com*/

#menu1 {color:green; background-color: DarkGray; text-decoration:none; }

#menu1 a {color:green; background-color: DarkGray; text-decoration:none; }
```

El efecto con este código sería el mismo que con el anterior. Realmente en este caso no tendría interés escribir la especificación para menu1 {..} y para menu1 a {..} por separado ya que resulta más compacto escribirlo en una línea. Sin embargo sí tiene interés en el caso de que deseemos aplicar un estilo general al menú y sobreescribir o especificar estilos diferenciados para los elementos <a> ...</a> dentro del menú.

En nuestro caso vamos a hacer lo siguiente: estableceremos como color general para el menú rojo y como fondo gris. Como fondo para los links (elementos a) dentro del menú no especificaremos ninguno, el color lo estableceremos en verde y eliminaremos el subrayado. Escribe este código y visualiza el resultado:

```
/* Curso CSS estilos aprenderaprogramar.com*/

#menu1 {color:red; background-color: DarkGray; text-decoration:underline; }

#menu1 a {color:green; text-decoration:none; }
```




En este caso tenemos como color de los elementos del menú rojo, pero los links se muestran en verde porque para los elementos a el color ha sido sobreescrito y el navegador elige el estilo más interno.

© aprenderaprogramar.com, 2006-2029

Prueba a establecer distintas propiedades para #menu1 y para #menu1 a. Posiblemente te encuentres que en algunos casos los resultados no son tal y como esperas. Ten en cuenta que el navegador tiene que resolver sobre el orden de importancia de las reglas, las sobreescrituras, estilos por defecto, etc. lo cual hace que el proceso adquiera cierta complejidad. En ocasiones se producen conflictos en la definición de estilos que el navegador tratará de resolver, aunque en algunos casos quizás no pueda resolver el problema y simplemente ignore los estilos que entran en conflicto. Iremos hablando sobre estas situaciones a medida que vayamos viendo ejemplos.

**¿CUÁNDO UTILIZAR SELECTORES POR ID?**

El atributo id nos permite darle un nombre específico a una parte de un documento HTML definida por una etiqueta HTML, diferenciándola de las demás. En general este atributo se utiliza para poner nombres a distintas partes de un documento HTML, por ejemplo menu, articulo, formulario, footer, etc. de modo que cada parte de la web está identificada con un nombre único. No es recomendable aplicar el mismo nombre id a distintas partes de una web, ni siquiera aunque sean del mismo tipo. Un id debe ser único, por lo tanto ese nombre debe aparecer solo una vez en el documento HTML. Si queremos aplicar un mismo estilo en distintas partes de la web utilizaremos el atributo class, que explicaremos más adelante.

No es obligatorio identificar todas las partes de una web con ids. Puede haber partes del documento HTML con identificador y otras partes que carezcan de él. Incluso la totalidad de la web puede carecer de identificadores id.

El uso del símbolo # combinado con el atributo id permite aplicar CSS a partes muy concretas de una página web y se usa sobre todo cuando se quieren diferenciar partes estructurales del documento HTML como el menú o el footer, y en general aplicado a elementos  ... que actúan como contenedores. Aunque se podría aplicar a etiquetas más específicas como <p> esto no es habitual ya que supondría estarle poniendo un nombre único a un párrafo y un párrafo en general no tiene tanta importancia dentro de una web como para ponerle un nombre específico. No obstante, ten en cuenta que existe la posibilidad de usar este atributo sobre cualquier etiqueta HTML.

**EJERCICIO**

Analiza el siguiente código HTML. En él encontrarás tres etiquetas div. Haz lo siguiente:

a) Establece atributos id para cada una de ellas con valores <<novedades>> para la primera, <<xFashion>> para la segunda y <<tMotion>> para la tercera.

b) Usando CSS establece como color de texto el rojo (red) para los elementos h1 que se encuentren dentro del elemento con id <<novedades>>.

© aprenderaprogramar.com, 2006-2029

c) Usando CSS establece como color de texto el verde para los elementos h3 que se encuentren dentro de los elementos con id <<xFashion>> y <<tMotion>>.

d) Usando CSS establece como color de fondo para el elemento con id <<novedades>> el amarillo (yellow).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<meta charset="utf-8">
</head>
<body>
<div>
<h1>Novedades</h1>
<p>Aquí presentamos las novedades del sitio.</p>
</div>
<div>
<h3>Lanzamos el producto X-FASHION</h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
</div>
<div>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto T-MOTION</p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</div>
</body>
</html>
```

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01012D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**SELECTORES CSS**

Ya sabemos cómo aplicar estilos CSS a todas las etiquetas HTML de un tipo, o bien a una parte del documento HTML con un nombre único (id). Pero muchas veces querremos aplicar una misma definición de estilos en distintos lugares de una página web. Para ello además del selector de etiqueta o palabra clave que ya conocemos y del selector # por id, podemos definir estilos para aplicarlos donde deseemos mediante el atributo class.

**SELECTOR POR CLASS**

Trataremos ahora de definir estilos que podamos aplicar en diferentes lugares de una página o proyecto web, allá donde nosotros deseemos. Para ello haremos dos cosas:

a) En el documento HTML, habrá que identificar la parte del documento a la que queremos aplicar esos estilos incluyendo un atributo para la etiqueta que delimita el fragmento de código donde queremos aplicar estilos. Por ejemplo si queremos aplicar estilos dentro de un contenedor div la sintaxis sería  ... . Si quisiéramos aplicar estilos sólo a un párrafo la sintaxis sería  ... . Podemos comprobar que la sintaxis es casi igual a la empleada con el atributo id, pero ahora usando el término class.

b) En la declaración de estilos tendremos que declarar los estilos a aplicar a ese fragmento de código realizando la declaración usando esta sintaxis.

```
.nombreIdentificativoElegido {
propiedad1ParaEseTipoDeElementos:valor;
propiedad2ParaEseTipoDeElementos:valor;
propiedad3ParaEseTipoDeElementos:valor;
propiedad4ParaEseTipoDeElementos:valor;
propiedad5ParaEseTipoDeElementos:valor;
propiedad6ParaEseTipoDeElementos:valor;
...
propiedadnParaEseTipoDeElementos:valor;
}
```

Mientras que para el id el símbolo CSS que se empleaba era la almohadilla ó #, para los class el símbolo CSS que empleamos es el . (punto).

Las propiedades se pueden definir en una sola línea o agrupándolas como nosotros queramos, el único requisito es separarlas con punto y coma.

© aprenderaprogramar.com, 2006-2029

Con un ejemplo lo veremos más claro. Partimos del código base HTML que estamos usando para el curso, en el cual tenemos un menú, tres párrafos, dos imágenes y un formulario. Vamos a definir un estilo CSS consistente en poner fondo naranja, texto en negrita y el texto un 100 % del tamaño normal con el fin de aplicarlo en dos puntos de nuestra página web: al primer párrafo y al texto "Si quieres contactar..." del formulario. A este estilo vamos a llamarlo "destacado" y lo definimos en nuestro archivo css de esta forma:

```
/* Curso CSS estilos aprenderaprogramar.com*/
.destacado { background-color: orange; font-weight:bold; font-size:100%; }
```

Ahora tenemos que aplicar estilos CSS en las partes del documento HTML que nos interesan, el primer párrrafo:

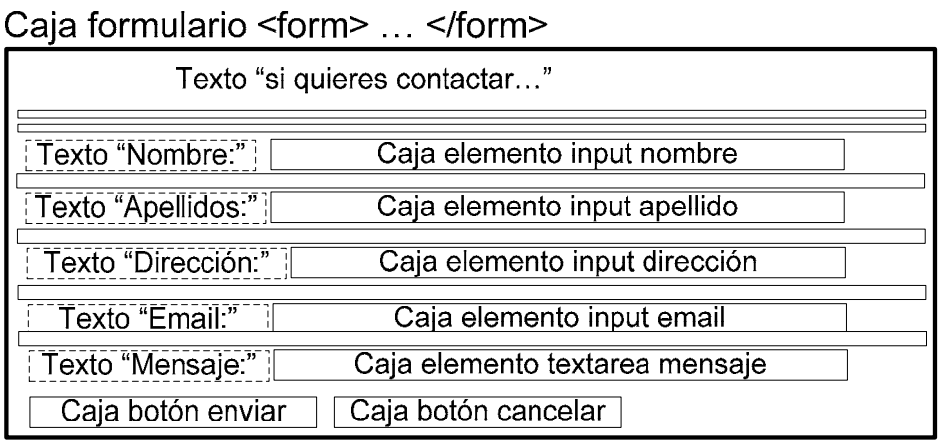
```
<p class="destacado">Aprender a programar es un objetivo que se plantea
mucha gente y que no todos alcanzan.</p>
```

Hemos indicado de esta forma que la caja contenedora definida por las etiquetas  ... del primer párrafo verá su estilo afectado por las reglas CSS que hemos definido.

Pero en el caso del formulario tenemos un problema. El código inicial es este:

```
<form  method="get" action="accion.html">
Si quieres contactar con nosotros envíanos este formulario relleno: <br /><br />
Nombre: <input type="text" name="nombre" /><br />
Apellidos: <input type="text" name="apellidos" /><br />
Dirección: <input type="text" name="direccion" /><br />
Correo electrónico: <input type="text" name="correo" /><br />
Mensaje: <textarea name="mensaje" cols=30 rows=2></textarea><br /><br />
<input type="submit" value="Enviar">
<input type="reset" value="Cancelar">
</form>
```

Y el texto al que queremos aplicar el estilo destacado es "Si quieres contactar con nosotros...". ¿Cuál es el problema? Que dicho texto no tiene una caja específica, sino que es un texto "suelto" dentro de otra caja, en este caso la caja inmediata que lo enmarca es la caja del formulario. Podemos verlo en este esquema:




© aprenderaprogramar.com, 2006-2029

Si queremos aplicar estilos tenemos que hacerlo sobre etiquetas delimitadoras (una caja) y ahora mismo solo disponemos de <form> ... </form>. Si escribimos <form class="destacado" method="get" action="accion.html"> el resultado sería este:

Obviamente el estilo se aplica a toda la caja del formulario, y no sólo al texto que nosotros deseamos.

¿Qué solución podemos darle? Hemos de crear una caja específica para el texto ya que es sólo al texto a quien queremos aplicarle el estilo "destacado". Esto podemos hacerlo de varias maneras: con un párrafo  ... , con un contenedor  ...  o con un divisor  ... . Las diferencias entre las opciones son que el párrafo es un elemento de tipo block que lleva asociados un estilo predeterminado por el navegador. div es un elemento que crea una división o caja y es de **tipo block**, también ocupará la página web a todo lo ancho. div en principio no lleva estilos predeterminados asociados. Por último span es un elemento que crea una **división inline**, por tanto no abarca todo el ancho de la página sino el espacio ocupado por aquello que contiene (en este caso el espacio ocupado por el texto) y tampoco tiene estilos predeterminados asociados. Fíjate en la diferencia entre aplicar un estilo a un elemento block o a un elemento inline.

© aprenderaprogramar.com, 2006-2029

Elegiremos aquel elemento divisor que nos parezca más adecuado en función de los criterios de diseño. En nuestro caso vamos a optar por un divisor  ...

El código del formulario en el documento HTML nos queda así:

```
<form method="get" action="accion.html">
<p class="destacado">Si quieres contactar con nosotros envíanos este formulario relleno:</p> <br /><br />Nombre: <input type="text" name="nombre" /><br />
Apellidos: <input type="text" name="apellidos" /><br />
Dirección: <input type="text" name="direccion" /><br />
Correo electrónico: <input type="text" name="correo" /><br />
Mensaje: <textarea name="mensaje" cols=30 rows=2></textarea><br /><br />
<input type="submit" value="Enviar">
<input type="reset" value="Cancelar">
</form>
```

Ejecuta el código. La visualización obtenida debe ser la tipo block que hemos indicado anteriormente.

**EJERCICIO**

Analiza el siguiente código HTML. En él encontrarás tres etiquetas div. Haz lo siguiente:

a) Establece atributos class para cada una de ellas con valores <<principal>> para la primera, y <<secundario>> para la segunda y la tercera.

b) Usando CSS establece como color de texto el rojo (red) y tamaño de fuente el 150% respecto de lo normal para los elementos cuyo valor class sea <<principal>>.

c) Usando CSS establece como color de texto el verde para los elementos y tamaño de fuente el 110% respecto de lo normal para los elementos cuyo valor class sea <<secundario>>.

d) Usando CSS establece como color de fondo para los párrafos dentro de elementos cuyo atributo class sea <<secundario>> el amarillo (yellow).

© aprenderaprogramar.com, 2006-2029

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<meta charset="utf-8">
</head>
<body>
<div>
<h1>Novedades</h1>
<p>Aquí presentamos las novedades del sitio.</p>
</div>
<div>
<h3>Lanzamos el producto X-FASHION</h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
</div>
<div>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto T-MOTION</p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</div>
</body>
</html>
```

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01013D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**HERENCIA EN CSS**

Hemos visto cómo aplicar estilos CSS a todas las etiquetas HTML de un tipo, o bien a una parte del documento HTML con un nombre único (id), o bien en distintas partes de un documento o proyecto web usando class. Una característica interesante de CSS es cómo nos permite definir estilos en clases superiores o clases padre que se van transmitiendo hacia las subclases o clases hijas.

**DEFINIR CLASES QUE SOLO SEAN APLICABLES A ETIQUETAS ESPECÍFICAS**

Partimos del código que hemos utiliado en la anterior entrega del curso. En concreto habíamos definido una clase CSS cuyo nombre era "destacado" y cuyo código era:

```
/* Curso CSS estilos aprenderaprogramar.com*/
.destacado { background-color: orange; font-weight:bold; font-size:100%; }
```

Podemos definir que una clase solo sea aplicable a una etiqueta específica. Por ejemplo si modificamos el código que define la clase destacado y escribimos:

```
/* Curso CSS estilos aprenderaprogramar.com*/
p.destacado{ background-color: orange; font-weight:bold; font-size:100%; }
```

Ahora el estilo destacado sólo surte efecto cuando va en una etiqueta p, en cambio en otras etiquetas como div, span, a, etc. no tendría efecto.

Esto nos permitiría usar el mismo nombre de clase y obtener distintos efectos según en qué etiqueta aplicáramos la clase. Por ejemplo:

```
/* Curso CSS estilos aprenderaprogramar.com*/
p.destacado { background-color: orange; font-weight:bold; font-size:100%; }
div.destacado {background-color: blue; font-weight:bold; font-size:100%; }
```

Ahora si definimos un párrafo con el atributo class = "destacado" su fondo será naranja. En cambio si es un divisor div con el atributo class = "destacado" su fondo será azul. Prueba a ejecutar código comprobando estas diferencias.

**DEFINIR SUBESTILOS DENTRO DE UNA CLASE**

Podemos definir que determinadas cajas dentro de otra principal tengan estilos diferenciados. Por ejemplo, podemos querer que los párrafos de clase "destacado" tengan fondo naranja, pero que si

© aprenderaprogramar.com, 2006-2029

dentro del párrafo hay un link (etiquetas <a> ...</a>) dicho link tenga un estilo específico, por ejemplo que tenga fondo amarillo y un tamaño un 20% superior a lo normal. Para ello escribiríamos:

```
/* Curso CSS estilos aprenderaprogramar.com*/
p.destacado{
background-color: orange;
font-weight:bold;
font-size:100%; }
p.destacado a { background-color: yellow;
 font-size:120%;
}
```

Con la declaración p.destacado a { ...} estamos indicando que los links dentro de párrafos con estilo destacado estarán en negrita (**por herencia** del estilo general aplicable a la clase destacado) y tendrán fondo amarillo y tamaño de letra un 20% superior a lo normal (ya que hemos sobreescrito las propiedades de color de fondo y tamaño de letra de la clase padre).

Si quisiéramos definir un estilo destacado para los párrafos, otro para los div, pero el mismo subestilo para los links que estén tanto dentro de párrafos como dentro de divs escribiríamos lo siguiente:

```
/* Curso CSS estilos aprenderaprogramar.com*/
p.destacado{ background-color: orange; font-weight:bold; font-size:100%; }
div.destacado { background-color: blue; font-weight:bold; font-size:100%; }
p.destacado a, div.destacado a { background-color: yellow; font-size:120%; }
```

Con la declaración p.destacado a, div.destacado a { ...} aplicamos el mismo estilo a los link dentro de párrafos o div que lleven el atributo class = "destacado".

Se observa una cosa: podemos definir estilos agrupando el código de distintas maneras. Podemos repetir código pero también podemos crear clases padre que agrupen las características comunes y definir en clases hijo las características específicas. Veamos el siguiente código:

```
/* Curso CSS estilos aprenderaprogramar.com*/
.destacado {font-weight:bold; font-size:100%;}
p.destacado {background-color: orange;}
div.destacado { background-color: blue; }
p.destacado a, div.destacado a {background-color: yellow; font-size:120%; font-weight:lighter; }
```

Este código define: una clase padre o superclase <<destacado>> según la cual la letra será negrita y el tamaño de letra normal (100 %). Las clases hijas <<p.destacado>> y <<div.destacado>> aportan las características de la clase padre (por herencia) más algunas otras características adicionales (añadidas) o distintas (sobreescritura de propiedades de la clase padre), y las clases hijas de las hijas <<p.destacado a>> y <<div.destacado a>> tienen las características de la clase padre y de la clase abuela con algunas características adicionales o distintas.

Observamos aquí una propiedad importante de CSS: la herencia o transmisión de características de clases antecesoras a clases sucesoras o de clases padre a clases hija.

© aprenderaprogramar.com, 2006-2029

**CONCEPTO DE HERENCIA CSS**

La herencia CSS permite evitar la repetición de código en todas las subclases que derivan a partir de una clase superior. La herencia se aplica tanto a estilos de etiquetas HTML (por ejemplo todas las etiquetas son subetiquetas de la etiqueta padre body, de modo que un estilo aplicado a body será heredable por el resto de etiquetas HTML) como a selectores id o class.

Esto nos da alguna pista sobre prácticas CSS que son habituales entre programadores o diseñadores CSS experimentados: si aplicamos algunos estilos básicos como un tipo de letra (propiedad CSS font-family) a la etiqueta body y todas las otras etiquetas son herederas del estilo de body, habremos conseguido definir el tipo de letra en toda la página web en una simple línea, sin necesidad de tener que estar repitiendo el estilo en múltiples ocasiones para párrafos, div, span, forms, etc.

Ejemplo de declaración de este tipo:

```
body {font-family: Arial, sans-serif; }
```

En un documento HTML todo tiene ascendiente excepto el elemento html. Así body hereda de html, div hereda de body y <<div a>> (link dentro de un contenedor div) hereda de div. En nuestro caso el estilo << p.destacado a>> hereda de <<p.destacado>> y <<p.destacado>> hereda de <<destacado>>.

Sin embargo no todas las propiedades se heredan. Por ejemplo la propiedad font-family es heredable, pero los márgenes (propiedad margin) no se heredan excepto si se indica explícitamente que deben heredarse. ¿Por qué unas propiedades se heredan y otras no? Los creadores de CSS pensaron que sería útil que algunas propiedades se heredaran porque suelen mantenerse constantes dentro de una caja. Por ejemplo lo más normal es usar un tipo de letra en un contenedor y no mezclar distintos tipos de letra. En cambio, otras propiedades era poco lógico que se heredaran, como los márgenes, ya que por ejemplo las fotografías dentro de un artículo normalmente tenían un margen distinto al que tenía el propio artículo respecto a su contenedor o los párrafos. Nos encontramos así con que la herencia de propiedades se rige en cierta manera por el "sentido común". No obstante, en la especificación oficial de CSS encontraremos que cuando se define una propiedad se establece un atributo denominado "inherit" que puede estar establecido en "yes" o en "no" según la propiedad de que se trate. Así font-family es una propiedad cuyo valor inherit es "yes": esta propiedad se hereda. En cambio la propiedad margin tiene un valor inherit "no", lo cual significa que no se hereda. Más adelante abordaremos las propiedades CSS, explicaremos cómo consultar la documentación oficial CSS y estudiaremos esta circunstancia con más detalle.

Hay algunas curiosidades en torno a la herencia. Por ejemplo no todos los navegadores responden exactamente igual en cuanto a la herencia. Podrías encontrarte algún navegador donde no se genera herencia de una propiedad mientras que en otro sí.

Otra curiosidad es algo que hemos comentado: hemos dicho que la propiedad background-color se transmite por herencia. Sin embargo la definición del estándar CSS dice que background-color tiene un valor inherit "no". ¿Cómo se explica esto? Lo veremos más adelante cuando hablemos de las propiedades CSS, pero baste decir que el color de fondo por defecto para todos los elementos es el transparente. Al tener el padre un color y superponer el hijo transparente el efecto visual es que se hereda el color del padre, aunque formalmente no es así.

© aprenderaprogramar.com, 2006-2029

## FORZAR LA HERENCIA CON INHERIT

Hay una manera explícita para indicar que la propiedad que se debe aplicar es la heredada del estilo padre. Considera el código con que venimos trabajando en el que tenemos un menú. Supón que el menú lo hemos identificado con :

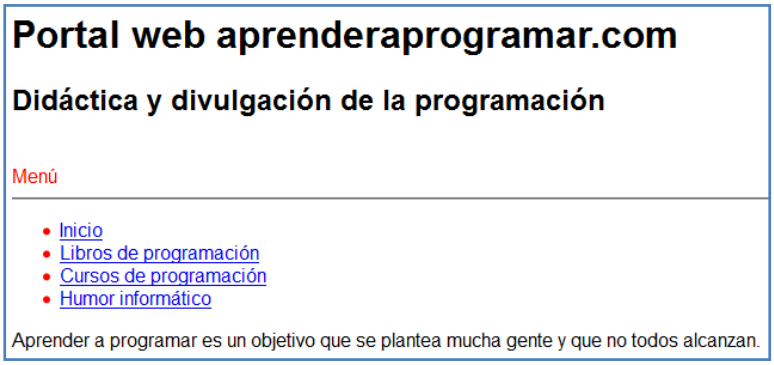
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<!—Código ejemplo para el curso -->
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta charset="utf-8">
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<link rel="stylesheet" type="text/css" href="estilos.css">
</head>
</head>
<!-- Contenido de la página web -->
<body>
<!-- Cabecera de la página web -->
<div>
<h1>Portal web aprenderaprogramar.com</h1>
<h2>Didáctica y divulgación de la programación</h2>
</div>
<!-- Fin de la cabecera de la página web -->
<br />
<!-- Contenedor para la parte central -->
<div>
<!-- menu -->
<div id="menu1">
<div>Menú</div>
<hr/>
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
</div>
<!-- fin menu -->
<!—Aquí el resto del código del ejemplo -->

</html>
```

Escribe esta definición en el archivo estilos.css y visualiza el resultado:

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial, sans-serif;}
#menu1{ color: red; text-decoration:none;}
```

Se obtiene algo similar a esto:

© aprenderaprogramar.com, 2006-2029




Vemos que el color rojo no está siendo aplicado por la etiqueta a (link) dentro del elemento con id menu1, y esto a pesar de que la propiedad color es heredable (su valor inherit = yes). La propiedad text-decoration tampoco está siendo aplicada. En este caso text-decoration no es heredable (inherit = no). En realidad que sea heredable o no no está teniendo efecto aquí ya que las propiedades por defecto del elemento <<a>> sobreescriben a las propiedades que hubiéramos definido en nuestro estilo menu1. Ya sabemos que podemos sobreescribir la propiedad de color y no subrayado del elemento <<a>> dentro de menu1 por ejemplo escribiendo este código (ejemplo A):

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial, sans-serif;}
#menu1{ color: red; text-decoration:none;}
#menu1 a { color: red; text-decoration:none;}
```

Que más sintéticamente podemos escribir así (ejemplo B):

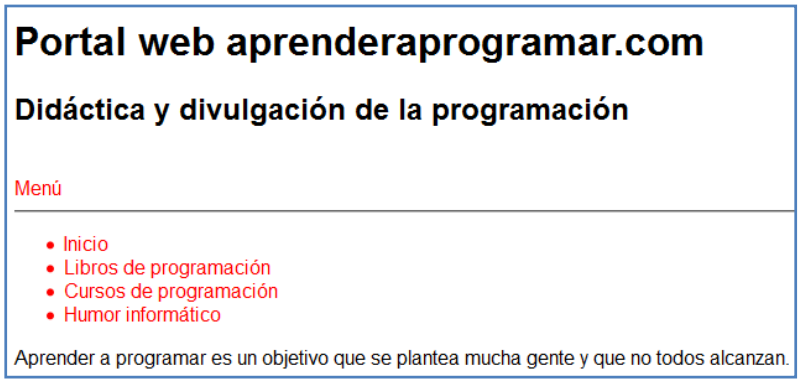
```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial, sans-serif;}
#menu1, #menu1 a { color: red; text-decoration:none;}
```

Conociendo la palabra clave inherit también podríamos obtener el mismo efecto así (ejemplo C):

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial, sans-serif;}
#menu1{ color: red; text-decoration:none;}
#menu1 a { color: inherit; text-decoration:inherit; }
```

En este caso hemos indicado que los links dentro de menu1, los hijos <<menu1 a>>, deben tener un color y un text-decoration tal como se haya definido en el padre, <<menu1>>.

El resultado con (A), (B) o (C) será algo similar a esto:




© aprenderaprogramar.com, 2006-2029

El uso de inherit puede tener algunas desventajas:

a) Algunos navegadores (en especial más antiguos) puede que no reconozcan la palabra clave inherit, con lo cual seguirían mostrando los links dentro del menú en color azul y subrayados.
b) Puede hacer el código menos claro o menos fácil de entender, en especial si tenemos que recorrer varios niveles de herencia hasta comprobar desde dónde viene una propiedad. En este caso la sobreescritura puede resultar más clara.
c) Podemos caer en la tentación, como se ha hecho en el ejemplo anterior, de aplicar inherit a una propiedad que en CSS tiene valor inherit = no. Es decir, aplicar herencia CSS a una propiedad que de acuerdo con la documentación oficial CSS no tiene herencia. Esto puede generar confusiones y hacer el código difícil de comprender y mantener. Sobreescribiendo la propiedad no habría lugar a dudas.

Por ello en general no usaremos la invocación de estilos de elementos padre con inherit excepto en circunstancias excepcionales. Esto no quita que debamos conocer su significado pues podemos enfrentarnos a tener que analizar o modificar hojas de estilo que hayan sido generadas por otras personas.

**RESUMEN**

Las clases CSS pueden ser aplicadas a etiquetas específicas (por ejemplo que la clase solo sea aplicable a  ... ) o a etiquetas de cualquier tipo, según hagamos la definición de reglas CSS.

También pueden definirse los estilos de modo que un mismo nombre de clase dé lugar a la aplicación de distintas reglas en función de en qué etiqueta se aplique, por ejemplo que una clase dé lugar a un fondo naranja si se aplica en una etiqueta  y a un fondo azul si se aplica en una etiqueta .

Las etiquetas HTML o estilos CSS definidos presentan herencia de estilos para aquellas propiedades que tienen un valor inherit "yes". La herencia puede no producirse si existe sobreescritura del estilo por parte de algún elemento, o si una propiedad tiene valor inherit no.

Puede forzarse la herencia desde un elemento padre escribiendo en el código CSS del elemento hijo nombreDeLaPropiedad: inherit, aunque en general será preferible sobreescribir la propiedad en lugar de indicar herencia con inherit por ser más claro.

**EJERCICIO**

Analiza el siguiente código HTML. En él encontrarás tres etiquetas div. Haz lo siguiente:

a) Establece atributos class para cada una de ellas con valores <<principal>> para la primera, y <<secundario>> para la segunda y la tercera.

b) Usando CSS establece como color de texto el rojo (red) y tamaño de fuente el 130% respecto de lo normal para los elementos h1 que se encuentren dentro de un elemento cuyo valor class sea <<principal>>.

© aprenderaprogramar.com, 2006-2029

c) Usando CSS establece como color de texto el verde y tamaño de fuente el 110% respecto de lo normal para los elementos h3 que se encuentren dentro de un elemento cuyo valor class sea <<secundario>>.

d) Usando CSS establece como color de fondo el amarillo (yellow) para los elementos span que se encuentren dentro de elementos h3 que se encuentren dentro de elementos cuyo atributo class sea <<secundario>>.

e) Establece como tipo de fuente para todo el documento HTML el tipo Arial.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title>
<meta name="description" content="Portal web aprenderaprogramar.com">
<meta name="keywords" content="aprender, programar, cursos, libros">
<meta charset="utf-8">
</head>
<body>
<div>
<h1>Novedades</h1>
<p>Aquí presentamos las novedades del sitio.</p>
</div>
<div>
<h3>Lanzamos el producto <span>X-FASHION</span></h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
</div>
<div>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto <span>T-MOTION</span></p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</div>
</body>
</html>
```

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01014D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## REFERENCIAR CON CUALQUIER ATRIBUTO

Sabemos que podemos aplicar estilos CSS a partir de una etiqueta HTML (referenciar por tag), o bien a partir de una parte del documento HTML con un nombre único (referenciar por id), o bien a partir del atributo class (referenciar por clase). A partir de CSS3 se introdujo la posibilidad de referenciar estilos mediante cualquier atributo de elementos HTML.

Los atributos son parámetros asociados a una etiqueta HTML y que tienen un valor concreto. Por ejemplo en  ...  el atributo class del contenedor div tiene como valor "destacado". Pero encontramos muchos otros atributos en el código HTML. Algunos ejemplos son:

| Ejemplo código | Etiqueta | Atributo | Valor del atributo |
|---|---|---|---|
| <a href="libros.html">Libros de programación</a> | <a> ...</a> | href | libros.html |
| <img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++, un útil editor de texto"> | <img ..> | src | http://i.imgur.com/afC0L.jpg |
| <img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++, un útil editor de texto"> | <img ..> | alt | Notepad++ |
| <img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++, un útil editor de texto"> | <img ..> | title | Notepad++, un útil editor de texto |
| <input type="text" name="nombre" /> | <input ..> | type | text |
| <input type="text" name="nombre" /> | <input ..> | name | nombre |
| <input type="submit" value="Enviar"> | <input ..> | type | submit |

Con CSS podemos aplicar estilos a todos los elementos de un tipo que cumplan tener cierto atributo. La sintaxis a emplear es:

```
NombreElementoHTML[selectorDeAtributo] {
propiedad1ParaEseTipoDeElementos:valor;
propiedad2ParaEseTipoDeElementos:valor;
...
propiedadnParaEseTipoDeElementos:valor;
}
```

© aprenderaprogramar.com, 2006-2029

Como selector de atributo tenemos distintas posibilidades. A continuación señalamos algunas:

| Selector de atributo | Ejemplo CSS | Observaciones |
|---|---|---|
| [nombreAtributo] | input[name] { background-color: red;}<br>p[title] {background-color: yellow;}<br>Nota: no puede haber espacio entre el nombre de elemento y el corchete. | Selecciona todos los elementos que tienen el atributo nombreAtributo, independientemente de su valor. En el ejemplo, todos los inputs que lleven name tendrán fondo rojo y todos los párrafos que lleven title tendrán fondo amarillo. |
| [nombreAtributo = "valor"] | input[name="correo"]{background-color: yellow; font-size:75%;} | Selecciona todos los elementos cuyo nombreAtributo es valor. En el ejemplo, se pone fondo amarillo y tamaño de letra 75% a todos los elementos input con atributo nombre = "correo". Si el input no tiene nombre ó este no es "correo", no se aplica. |
| [nombreAtributo^="valor"] | a[href^="http://www.aprender"] {background-color: yellow; font-size:75%;} | Selecciona todos los elementos cuyo nombreAtributo comienza por valor. En el ejemplo, se pone fondo amarillo y tamaño de letra 75 % a todos los links cuyo atributo href comienza por http://www.aprender |
| [nombreAtributo*="valor"] | a[href*="aprenderaprogramar"] {background-color: pink; font-size:125%;} | Selecciona todos los elementos cuyo nombreAtributo contiene a valor. En el ejemplo, se pone fondo rosado y tamaño de letra 125 % a todos los links cuyo atributo href contiene aprenderaprogramar. |
| [nombreAtributo$="valor"] | a[href$=".com"], a[href$="6"] {background-color: pink; font-size:125%;} | Selecciona todos los elementos cuyo nombreAtributo termina en valor. En el ejemplo, se pone fondo rosado y tamaño de letra 125 % a todos los links cuyo atributo href termina en .com ó en 6. |

Los selectores ^, * y $ tienen aquí un uso similar al que se da en otros lenguajes para crear lo que se denomina expresiones regulares: expresiones que siguen un determinado patrón (como empezar por.., terminar por.., contener.., etc.).

Escribe código CSS utilizando estos selectores y pruébalos sobre el documento HTML base que estamos usando para el curso. Recuerda no dejar espacios entre el nombre de elemento y los corchetes, es decir, a[href$="6"] es correcto pero a [ href$="6" ] no es correcto por contener espacios.

**EJERCICIO**

Crea una regla de estilos para aplicar la propiedad font-family: Arial a toda la página web. Crea otra regla que afecte a todas las etiquetas de imagen (<img ..>) que tengan atributo "title" y aplícales las siguientes propiedades CSS: border-style:solid, border-width: 5px y border-color: LightSalmon. Aplica esta regla al código HTML base que estamos usando para el curso a través del archivo estilos.css con el que estamos aplicando estilos a este documento HTML y visualiza los resultados.

© aprenderaprogramar.com, 2006-2029

**SOLUCIÓN**

El código a incluir en el archivo estilos.css sería el siguiente:

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial;}
/* Aplicamos un borde especial a las imágenes que tienen atributo title*/
img[title]{border-style:solid; border-width: 5px; border-color: LightSalmon;}
```

Nota: si tu navegador no reconoce LightSalmon escribe #FFA07A en lugar de LightSalmon. Es decir, escribiríamos border-color: #FFA07A;

El resultado al visualizar el documento HTML en el navegador debe ser similar a este, donde vemos las imágenes que tienen atributo title con un borde color salmón:

**SELECTORES BÁSICOS FRENTE A ESPECIALES**

La mayoría de las páginas web se pueden trabajar con los selectores básicos basados en etiquetas HTML, atributos id y class. En general otros selectores tienen menor uso, en parte porque pueden complicar la dificultad de comprensión de una hoja de estilos. Restringiremos por tanto su uso a casos en que sea especialmente necesario y comentaremos adecuadamente el código CSS cuando los utilicemos.

**Próxima entrega:** CU01015D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**SELECTORES AVANZADOS CSS**

Además de los selectores básicos (por etiqueta, por id o por class) y de los selectores basados en atributos HTML, disponemos de algunas formas más de establecer criterios de selección para la aplicación de estilos CSS: selectores especiales y pseudoclases CSS.

Hay selectores especiales que usan los símbolos >, + y ~ para especificar la relación entre elementos.

Como selector de atributo tenemos distintas posibilidades. A continuación señalamos algunas:

| Selector | Ejemplo CSS | Observaciones |
|---|---|---|
| elemPadre > elemHijo | div > ul { background-color: red;}<br>div > p.destacado {background-color: pink;}<br>div > img[src$=".png"] {background-color: blue; } | Selecciona todos los elementos de tipo elemHijo que están contenidos en un elemento elemPadre directamente (sin que existan otros elementos entre ambos). En el ejemplo, todos las listas tipo ul que estén dentro de un div tendrán fondo rojo, todos los párrafos cuya clase sea destacado y estén dentro de un div tendrán fondo rosa y todas las imágenes tipo png dentro de un div tendrán fondo azul . |
| elemAnterior + elemSiguiente | p.destacado + p { color: green;} | Selecciona todos los elementos de tipo elemSiguiente que estén exactamente después de un elemAnterior (sin otros elementos entre ellos) y que tengan al mismo elemento padre que éste. En el ejermplo, aquellos párrafos que estén justo después de otro párrafo cuya clase sea destacado tendrán el texto de color verde. |
| elemAnterior ~ elemPosterior | p.destacado ~ p{ color: green;} | Selecciona todos los elementos de tipo elemAcontinuacion que aparezcan después de un elemAnterior (pudiendo existir elementos intermedios entre ellos) y que tengan al mismo elemento padre que éste. En el ejemplo, aquellos párrafos que estén después de un párrafo cuya clase sea destacado y que tengan el mismo contenedor tendrán color verde. |

Es importante tener en cuenta que las reglas que usan > solo se aplican cuando existe una relación directa padre – hijo ó contenedor – contenido sin que existan cajas intermedias entre los elementos indicados. Por ejemplo div ul se aplica a una lista incluso si el elemento ul está dentro de un divisor span intermedio entre la lista y el div. En cambio div > ul no se aplica a una lista si está dentro de un span. Las reglas + sólo se aplican entre "hermanos" (elementos con un mismo padre, por ejemplo dos párrafos dentro de un div serían hermanos cuyo padre es el div) al hermano justo después del indicado. Por último las reglas ~ se aplican entre hermanos a todos los hermanos que se encuentren con posterioridad al indicado incluso aunque estén separados por elementos intermedios.

© aprenderaprogramar.com, 2006-2029

**PSEUDOCLASES CSS**

Existen algunos elementos especiales que se pueden seleccionar con lo que en CSS se denominan pseudoclases, que son identificadores especiales que van precedidos por dos puntos. Entre las pseudoclases disponibles tenemos las siguientes:

| Pseudoclase | Ejemplo CSS | Observaciones |
|---|---|---|
| tipoElem:first-child | li:first-child {font-size: 200%;} | Selecciona el primer elemento dentro de una serie de elementos del tipo tipoElem. En el ejemplo, allí donde haya una serie de elementos li, el primero de ellos tendrá el doble de tamaño de fuente normal. |
| tipoElem:last-child | li:last-child {font-size: 200%;} | Selecciona el último elemento dentro de una serie de elementos del tipo tipoElem. En el ejemplo, allí donde haya una serie de elementos li, el último de ellos tendrá el doble de tamaño de fuente normal. |
| tipoElem:only-child | div.destacado img:only-child{border-style:solid; border-width:10px;} | Selecciona un elemento si es hijo único, es decir, si es el único elemento dentro del contenedor tipoElem. En el ejemplo se seleccionan las imágenes que estén dentro de contenedores div cuya clase sea destacado y dentro de los cuales exista únicamente una imagen y le aplica un borde de anchura 10 píxeles. |
| tipoElem:nth-child(number) | li:nth-child(3) {font-size: 200%;}<br>p.destacado:nth-child(2) {font-size: 50%;} | Selecciona el elemento número number dentro de una serie de elementos de tipo tipoElem. No puede haber elementos intermedios de otro tipo en la serie. En el ejemplo, el tercer elemento de una serie de li tendría el doble del tamaño normal de fuente y el segundo elemento dentro de una serie de párrafos con clase destacado tendría el 50 % del tamaño normal. |
| Elem1:not(tipoElem2) | p:not(.destacado) {color: yellow;} | Selecciona todos los elementos de tipo Elem1 que no sean de tipo tipoElem2. En el ejemplo, se seleccionan todos los párrafos cuya clase no sea destacado. |
| tipoElem::first-letter | p::first-letter {font-size: 300%;} | Selecciona la primera letra de tipoElem |
| tipoElem::first-line | p::first-line {color:red} | Selecciona la primera línea de tipoElem |
| tipoElem::after | h1::after {content:"***"; background-color: yellow;} | Posiciona al final del elemento tipoElem permitiendo añadir contenido usando content:"contenidoDeseado" con las propiedades que se indiquen. |
| tipoElem::before | h1::before {content:"---"; color: blue;} | Posiciona al principio del elemento tipoElem permitiendo añadir contenido usando content:"contenidoDeseado" con las propiedades que se indiquen. |

Escribe código CSS utilizando estos selectores y pruébalos sobre el documento HTML base que estamos usando para el curso.

© aprenderaprogramar.com, 2006-2029

No deben existir espacios al escribir estas declaraciones. Por ejemplo li:nth-child(3) {font-size: 200%;} es válido pero li : nth-child (3) {font-size: 200%;} no es válido por existir espacios.

Si el número referenciado con nth-child no existe, la regla no tendrá efectos. En lugar de first-child se puede usar nth-child(1) alternativamente.

Existen más pseudoclases CSS que iremos estudiando a lo largo del curso.

**COSAS QUE PUEDEN OCURRIR...**

Quizás con el código que hemos ido probando en lo que llevamos de curso hayas encontrado algún selector o fragmento de código que no te ha funcionado como cabía esperar. Quizás hayas pensado que has hecho algo mal, o que hay algún error en los contenidos del curso. Muchas veces no será ni una cosa ni otra. El motivo de que muchas veces no se obtenga el resultado esperado es la distinta respuesta de los distintos navegadores (Internet Explorer, Mozilla Firefox, Chrome, Opera, Safari, etc.) ante determinado código. A lo largo del curso iremos estudiando cómo resolver, en la medida de lo posible, estos problemas. Pero como ya hemos indicado en alguna ocasión, en el mundo de los desarrollos web los cambios y novedades son constantes y no queda otra que tener paciencia, tratar de adquirir experiencia y mantenerse actualizado a través de cursos, internet, libros, revistas, etc.

Pondremos un ejemplo relacionado con lo anterior. Hay una pseudoclase que es ::selection cuya inclusión en el estándar CSS se ha discutido y es permitida por algunos navegadores pero por otros no. La sintaxis sería del tipo p::selection { color: gold; background-color: red; } y daría lugar a que cuando el usuario hace una selección dentro de un elemento p, aquello seleccionado aparezca de color oro y con fondo rojo. Al no haberse estandarizado, es posible que no se reconozca la sintaxis por el navegador y que no se pueda ver este efecto, o bien que los distintos navegadores tengan distintas sintaxis para aplicar estos efectos y haya que escribir una línea específica de distinta manera para cada navegador.

Al igual que con esta pseudoclase puede ocurrir algo similar con otros elementos CSS. Sería ideal que todos los navegadores funcionaran igual, pero en la práctica nos encontraremos pequeñas o grandes divergencias entre ellos.

Otra cosa que puede ocurrir con cierta frecuencia es que estemos analizando el código css de una web o aplicación web y encontremos código que no hayamos estudiado. En ese caso tenemos que acudir a nuestro conocimiento general sobre CSS para interpretar que por ejemplo div:empty seleccionará los elementos div que no tengan hijos o contenedores internos, mientras que input:checked {background-color: green;} aplicará color de fondo verde a aquellos elementos input con el atributo checked="checked". Sin embargo, buena parte de este código puede que no sea soportado por muchos navegadores.

Algunas personas dicen: ¡Esto de los desarrollos web es caótico! Quizás no sea para tanto, en realidad hay un grado de estandarización bastante amplio, pero sí es verdad que en algunos momentos puede parecer un tanto caótico.

**EJERCICIO**

Dado este fragmento de código HTML que estamos usando como base para el curso:

© aprenderaprogramar.com, 2006-2029

```
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
<!-- fin menu -->
```

Definir que esta lista es de clase (class) especial. La clase especial no tendrá estilos especificados. En cambio, deberá definirse que los elementos li dentro de la lista ul de clase especial tengan color de fondo gris para los elementos impares (primero, tercero, quinto, séptimo...) y color de fondo rosa para los elementos pares (segundo, cuarto, sexto, octavo...).

Indicar los cambios en el documento HTML y el código a incluir en el archivo css.

**SOLUCIÓN**

En el archivo HTML escribiremos: <ul class="especial">

El contenido del archivo css puede ser el siguiente:

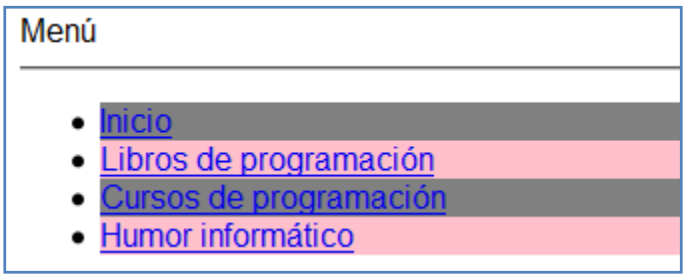
```
/* Curso CSS estilos aprenderaprogramar.com*/
ul.especial li:nth-child(1) {background-color: grey;}
ul.especial li:nth-child(2) {background-color: pink;}
ul.especial li:nth-child(3) {background-color: grey;}
ul.especial li:nth-child(4) {background-color: pink;}
```

Dado que solo tenemos 4 elementos en la lista solo definimos los cuatro primeros hijos.

Pero como queremos indicar que se apliquen los estilos a todos los elementos impares y pares sería más correcto usar esta definición:

```
/* Curso CSS estilos aprenderaprogramar.com*/
ul.especial li:nth-child(odd) {background-color: grey;}
ul.especial li:nth-child(even) {background-color: pink;}
```

El resultado será similar a este:




**Próxima entrega:** CU01016D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**EJEMPLOS SELECTORES CSS**

A lo largo del curso hemos ido estudiando distintos tipos de selectores que nos permiten indicar en un archivo css a qué elementos del código HTML se debe aplicar una regla. En la práctica los selectores pueden aparecer combinados unos con otros: vamos a ver ejemplos reales de selectores que nos podemos encontrar en páginas web.

**SELECTOR UNIVERSAL**

Se usa el símbolo * como selector universal. Este selector afecta a todos los elementos dentro de una página web. Suele usarse para anular los márgenes o espaciados predeterminados que utiliza el navegador. Un ejemplo sería el siguiente:

```
* {
margin: 0; padding: 0; color: green; text-decoration:none;
}
```

Esta regla establece las propiedades de margin y padding en 0 pixeles para todos los elementos, su color en verde y la propiedad text-decoration con valor none.

Un aspecto interesante del selector universal es que para las propiedades que se indiquen anula los valores por defecto que tengan los elementos HTML para esas propiedades. Por ejemplo los títulos h1, los párrafos, etc. tienen unos márgenes y colores por defecto. Usando el selector universal podemos anularlos (como en el ejemplo estableciendo su valor a cero) o establecerlos a un valor inicial concreto igual para todos (como en el ejemplo se establece el color verde o que no exista decoración para los textos). Este resultado no se consigue si esa definición se hace usando body { ...}

En la siguiente imagen vemos el resultado de aplicar esa definición del selector universal para el código HTML que estamos usando como base de ejemplo para el curso.

© aprenderaprogramar.com, 2006-2029

Vemos que al aplicar las propiedades margin y padding 0 usando el selector universal, todos los elementos aparecen "apelotonados" al carecer del espaciado que incorpora el navegador por defecto. Sin embargo, sí se aprecian algunos espacios. ¿Por qué? Porque tenemos definidos algunos saltos de línea con etiquetas <br/>. Estos saltos de línea siguen existiendo.

### MÁS SELECTORES

Algunos selectores adicionales respecto a los que hemos visto son estos:

| Selector | Ejemplo CSS | Observaciones |
|---|---|---|
| tipoElem:nth-last-child(num) | li:nth-last-child(2) {font-size: 200%;} | Selecciona los elementos de tipo tipoElem dentro de una serie contando desde el último hasta la posición indicada por num. En el ejemplo, los penúltimos elementos li dentro de una serie tendrán tamaño de fuente el doble de lo normal. |
| tipoElem:nth-of-type(sel) | li:nth-of-type(odd) {font-size: 200%; color: red;}<br>li:nth-of-type(even) {font-size: 100%;} | Selecciona los elementos de tipo tipoElem dentro de una serie y a los que son de tipo sel les aplica los estilos indicados. En el ejemplo los elementos li impares tendrán tamaño de fuente doble del normal y color rojo, mientras que los pares tendrán tamaño de fuente normal y el color que corresponda. |

Recuerda que en algunos navegadores es posible que no se obtengan los resultados deseados.

### EJEMPLOS DE CÓDIGO CSS Y SU INTERPRETACIÓN

Hasta ahora hemos trabajado con selectores normalmente de forma independiente: selectores por etiqueta, por id, por clase, por atributo, selectores avanzados específicos, etc. Cuando nos enfrentemos al desarrollo del código CSS de una web, o cuando tengamos que analizar una hoja de estilos desarrollada por otra persona, nos encontraremos que los selectores aparecen combinados de distintas maneras.

A continuación vamos a mostrar ejemplos reales de código CSS y a interpretar el significado de los selectores que se utilizan.

| Selector | Observaciones aprenderaprogramar.com |
|---|---|
| * { ...} | Selector universal. Afecta a todos los elementos de la web y sobreescribe los estilos por defecto. |
| html { ...} | Selector que afecta a toda la web pero no sobreescribe algunos estilos por defecto. |
| body.contentpane { ...} | Selector que afecta a cualquier elemento con atributo class = "contentpane" que esté dentro de <body> ...</body> |

© aprenderaprogramar.com, 2006-2029

| Selector | Observaciones aprenderaprogramar.com |
|---|---|
| #mainout { ...} | Selector que afecta a cualquier elemento con id = "mainout" |
| .padding { ..} | Selectores que afectan a elementos con class = "padding". |
| h3 #slo { ...} | Selector que afecta a elementos con id = "slo" situados dentro de elementos h3. Por ejemplo <h3 >Portal web para aprender programación  aprenderaprogramar.com</h3> |
| .find form .search .inputbox { ...} | Afecta a elementos cuya clase sea "inputbox" que se encuentren dentro de elementos cuya clase sea "search" que se encuentren dentro de elementos form que se encuentren dentro de elementos de clase "find". |
| .find, form, .search, .inputbox { ...} | Afecta a elementos cuya clase sea "inputbox" ó "search" ó "find" y a elementos <form> ... </form>. Nótese que la diferencia con el anterior selector es que en este caso los elementos están separados por comas. |
| .search label { ...} | Afecta a elementos <label> ... </label> que se encuentren dentro de un elemento con class = "search". |
| .find .searchintro { ...} | Afecta a elementos cuya clase sea searchintro que se encuentren dentro de elementos cuya clase sea find. |
| .syndicate-module span { ...} | Afecta a  ...  que se encuentren dentro de elementos con class = "syndicate-module" |
| fieldset a { ...} | Afecta a links (<a> ...</a>) dentro de <fieldset> ...</fieldset> |
| a img, fieldset, img { ...} | Afecta a elementos img dentro de links, a elementos fieldset y a elementos img. |
| fieldset { ...} | Afecta a elementos <fieldset> ...</fieldset> |
| #form-login fieldset { ...} | Afecta a elementos <fieldset> ... </fieldset> que estén dentro de elementos con id = "form-login" |
| #login ul, #form-login ul { ...} | Afecta a listas ul dentro de elementos con id = "login" ó listas ul dentro de elementos con id = "form-login" |
| .inputbox, .registration input, .login input, .contact-form input, #jform_contact_message, input { ..} | Afecta a elementos con class = "inputbox" ó elementos input que estén dentro de otro con clase "registration" ó elementos input que estén dentro de otro con clase "login" ó elementos input que estén dentro de otro con clase "contact-form" o elementos con id = "jform_contact_message" ó elementos input. |

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Indicar el ámbito de aplicación de los selectores que se indican:

| Selector | Ambito de aplicación |
|---|---|
| .header { ...} | |
| .find form .search .button { ...} | |
| .find, form, .search, .button { ...} | |
| .header label { ...} | |
| #cohe, #cobo { ...} | |
| .syndicate-module img { ...} | |
| #form-login p { ...} | |
| #login br { ...} | |
| #login .button { ...} | |
| label.invalid { ...} | |
| div.itemCommentsForm form input#submitCommentButton | |
| #s5_bottom_menu_wrap ul.menu li { ...} | |
| .module_round_box ul.menu ul a { ...} | |

© aprenderaprogramar.com, 2006-2029

**SOLUCIÓN**

| Selector | Ambito de aplicación |
|---|---|
| .header { ...} | Afecta a elementos cuya clase sea "header". |
| .find form .search .button { ...} | Afecta a elementos cuya clase sea "button" y estén dentro de elementos cuya clase sea "search" y estén dentro de un <form> ... </form> que esté dentro de un elemento cuya clase sea "find". |
| .find, form, .search, .button { ...} | Afecta a elementos cuya clase sea "button" ó "search" ó "find" ó estén dentro de un <form> ... </form>. |
| .header label { ...} | Afecta a elementos <label> ... </label> que se encuentren dentro de un elemento con class = "header". |
| #cohe, #cobo { ...} | Afecta a elementos con id = "cohe" ó elementos con id = "cobo" |
| .syndicate-module img { ...} | Afecta a <img ... > que se encuentren dentro de elementos con class = "syndicate-module" |
| #form-login p { ...} | Afecta a párrafos que estén dentro de elementos con id = "form-login" |
| #login br { ...} | Afecta a elementos <br/> que estén dentro de elementos con id = "form-login" |
| #login .button { ...} | Afecta a elementos con class = "button" que estén dentro de elementos con id = "login" |
| label.invalid { ...} | Afecta a elementos con class = "invalid" que estén dentro de <label> ... </label>. |
| div.itemCommentsForm form input#submitCommentButton | Afecta a elementos con id = "submitCommentButton" que estén dentro de elementos <input> ... </input> que estén dentro de elementos form que estén dentro de un div cuya clase sea "itemCommentsForm". |
| #s5_bottom_menu_wrap ul.menu li { ...} | Afecta a elementos li que se encuentren dentro de un elemento ul cuya clase sea "menu" y que se encuentren dentro de un elemento con id = "s5_bottom_menu_wrap" |
| .module_round_box ul.menu ul a { ...} | Afecta a links que se encuentren dentro de listas ul que se encuentren dentro de otras listas ul cuya clase sea "menu" y que se encuentren dentro de elementos cuya clase sea "module_round_box". |

**Próxima entrega:** CU01017D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**CASCADA EN CSS**

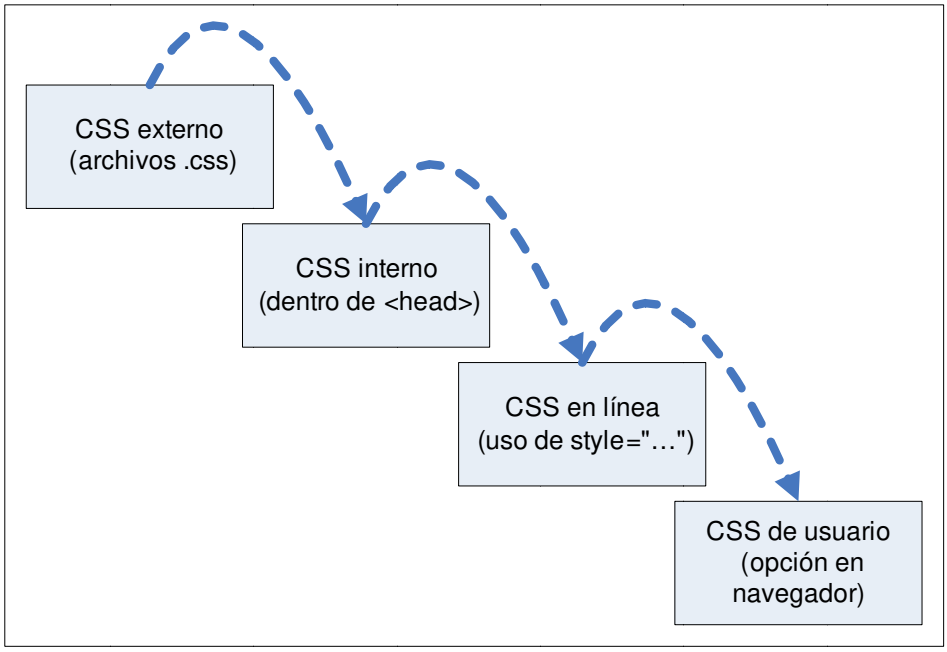
En epígrafes anteriores del curso hemos visto cómo funciona la herencia CSS y cómo los elementos de rango inferior heredan las propiedades CSS de elementos de rango superior. Por ejemplo un párrafo  ... hereda las propiedades de color definidos para <body> ...</body>. Sin embargo, algunas propiedades no se heredan y en algunas propiedades como el color de los link apreciábamos peculiaridades.

Una circunstancia con la que nos tendremos que familiarizar es la existencia de conflictos entre las declaraciones CSS. Por ejemplo si estamos trabajando con un archivo CSS externo podemos incluir en él una declaración p {font-color: blue;}. Pero podría suceder que como CSS interno entre las etiquetas <head> ...</head> se encontrara una declaración p {font-color: red;}. Y también podría suceder que en un párrafo concreto dentro del documento HTML apareciera una declaración como <p style = "font-color: yellow;". Estas declaraciones suponen un conflicto o "colisión de estilos" para el navegador, que ha de decidir qué estilo aplicar.

Ya comentamos que de forma general el navegador aplica el criterio de precedencia:

Declaración en línea > Declaración interna > Declaración externa

Esto podemos verlo como una cascada de forma que el navegador va recorriendo la cascada hasta llegar al nivel más bajo posible.




Hemos incluido en el esquema la opción "CSS de usuario" para reflejar que algunos navegadores permiten definir al usuario estilos propios.

© aprenderaprogramar.com, 2006-2029

**ESTILOS DE USUARIO CSS**

A través de una opción de menú de algunos navegadores, se puede definir por ejemplo que el texto se debe mostrar de un tamaño determinado o de un color determinado. Cuando el usuario elige estas opciones normalmente anula aquello que ha sido definido por el autor de la página web, de modo que sus opciones prevalecen respecto a cualquier otro tipo de declaración. Esta situación no es habitual, de hecho quizás la mayoría de los usuarios la desconocen o no la usan. Su empleo quizás se limita a personas con problemas de vista para poner tamaños de fuente grandes o colores específicos que les faciliten la visión, o a usuarios avanzados que tienen un gran dominio de las opciones de configuración. A pesar de su relativo poco uso, como programadores web debemos tener constancia de ello porque es posible que tengamos que resolver problemas donde esta configuración afecte a la solución del problema.

Aprovechando que hablamos de opciones de configuración de los navegadores, citaremos también que algunos navegadores permiten configurar opciones relativas a los estilos por defecto. Ya hemos dicho que cuando un documento HTML carece de estilos aplicados, en realidad sí tiene lo que podríamos denominar estilos básicos o estilos por defecto que aplica el navegador. Por ejemplo, una navegador recién instalado puede tener un tamaño de fuente por defecto de 12 pixeles, pero en aquellos navegadores que lo permiten, este parámetro podría cambiarse y establecerse en 16 pixeles a través de las opciones de configuración del navegador. También algunos navegadores permiten configurar otros parámetros adicionales. Este es uno de los motivos por los que es recomendable no confiar en los estilos por defecto cuando se crean páginas web, sino especificar todos los parámetros desde nuestra hoja de estilos para asegurarnos de que se aplican exactamente las reglas CSS que nosotros deseamos.

**ORIGEN, IMPORTANCIA Y ESPECIFICIDAD**

Nos encontramos con que existen distintos tipos de CSS como el predefinido por el navegador, el definido por el autor de la página web (que puede ser externo, interno o en línea) ó el definido por el usuario en la configuración de su navegador. A su vez, las reglas pueden entrar en conflicto (que una regla diga una cosa y otra regla una cosa distinta) y no sólo a nivel navegador – autor – usuario, sino en lo que para nosotros es la definición CSS principal, la que escribimos cuando creamos una web. Por ejemplo en un archivo css externo podemos tener una regla body {color: red;} y otra regla p { color: blue;} y otra regla #destacado {color: yellow;}. Si en el documento HTML nos encontramos algo así como  ... ¿Qué estilo aplica el navegador entre los tres posibles?

Todas estas situaciones se denominan situaciones de conflicto o colisión y CSS define una forma de resolución de conflictos o colisiones. A este mecanismo de resolución de conflictos se le denomina "cascada". La cascada se basa en determinar todas las reglas que son de aplicación a un elemento y que entran en conflicto, en ordenar esas reglas en base a unos criterios y mostrar como resultado en el navegador la regla que haya quedado en primer lugar, a la que denominamos "regla ganadora".

El mecanismo es similar a la asignación de "pesos" a cada regla CSS para lograr la ordenación. Supongamos que la regla body {color: red;} tiene peso 45 sobre 100 para ser aplicada a los párrafos que la regla p { color: blue;} tiene peso de 55 para ser aplicada a los párrafos y que la regla #destacado {color: yellow;} tiene peso 70 para ser aplicada a un párrafo con id = "destacado". Cuando el navegador

© aprenderaprogramar.com, 2006-2029

se encuentra un párrafo dentro del documento HTML ve las reglas que le son de aplicación y el peso de cada una, aplicando como regla ganadora la que mayor peso tenga.

¿Cómo se determinan los pesos y el orden que se asigna a las reglas, es decir, cómo se determina qué regla "gana" cuando hay varias reglas que entran en conflicto?

Las reglas de resolución de conflictos son complejas y no todos los navegadores responden de la misma manera. Por ello en cursos de aprendizaje de CSS no recomendamos un estudio en profundidad, sino conocer las reglas básicas y la filosofía del procedimiento de cascada.

El navegador sigue un proceso similar al siguiente para la resolución de colisiones de estilos:

1) Crea una lista con todas las reglas que son de aplicación directa o indirecta por herencia a un elemento concreto. Cuanto más directa o próxima esté la regla al elemento, antes se coloca. La proximidad viene dada por cuán próxima está la regla a una definición directa del elemento afectado. Por ejemplo si un elemento div con id="menu1" tiene un párrafo con class="items" entre #menu1 {..} y .items {..} la regla ganadora a la hora de aplicar estilo al párrafo es .items { ...} porque es la que más directamente define el elemento. En este ejemplo menu1 define al elemento indirectamente a través del div, mientras que items define directamente la clase del párrafo y por ello resulta ganadora.
   Otro ejemplo: para un párrafo dentro de un div, una regla que hereda de div está más próxima que una regla que hereda de body. Si existiera una regla directa relativa a los párrafos, esta iría en primer lugar antes que las demás debidas a herencia.
   Si existe una sola regla directa relativa al elemento que se está evaluando, se aplica y termina el proceso de cascada.

2) Si existe más de una regla con igual nivel de proximidad, se dividen las reglas por origen en base a si se trata de estilos de usuario, predefinidos del navegador, estilos de autor, y en caso de estilos de autor si son css externo, interno o en línea. Se ordenan en base al origen. Básicamente se colocan primero las declaraciones en línea, luego las internas y luego las externas.

3) Se dividen las declaraciones de reglas en declaraciones importantes y declaraciones normales. Las declaraciones importantes son las que están escritas como:

| nombreElemento {propiedad: valor **!important**;} |
|---|

Si dos declaraciones son iguales y una lleva !important y la otra no, prevalece la que lleve !important incluso aunque la otra aparezca con posterioridad. Por ejemplo si en un archivo css escribimos:

```
div {color:blue !important;}

div {color: pink; }
```

Podría pensarse que los elementos div tendrán color rosa por aparecer esta regla en último lugar y sobreescribir a la anterior. Sin embargo, no ocurre así porque la palabra clave !important hace que esa declaración prevalezca sobre otra igual que no lleve la palabra clave !important. El significado de !important en este caso sería similar a "esta regla sobreescribe a cualquier otra que defina la misma propiedad para el mismo elemento y no es sobreescribible".

© aprenderaprogramar.com, 2006-2029

Si en un archivo css externo tenemos la declaración p {color:blue !important;} y en el documento HTML tenemos en línea  ... la palabra clave !important anulará la precedencia del origen, de modo que el texto se verá de color azul. El significado de !important quedaría ampliado a "esta regla no es sobreescribible y anula la precedencia debida al origen".

Algunos navegadores antiguos no reconocen la palabra clave !important. Entre los navegadores modernos, no todos aplican exactamente el mismo significado a la presencia de esta palabra clave.

Teniendo en cuenta el origen y si existe o no la palabra clave !important, las reglas se ordenan situándose en primer lugar las declaraciones de estilos en línea, excepto cuando una regla de aplicación directa al elemento lleva la palabra clave !important. Si existe una regla ganadora, se aplica.

4) Si no existe una regla ganadora, para las reglas en conflicto se valora un parámetro denominado especificidad. Estaríamos en el caso de que existan reglas que definan directamente un elemento y que entren en conflicto. Por ejemplo estas dos reglas:

   body div.destacado p {color: cyan;}
   div.destacado p {color: yellow;}

Definen directamente el estilo para párrafos que se encuentren dentro de elementos div cuyo atributo class sea "destacado". No hay una regla entre las dos que defina más directamente el valor de la propiedad para ese tipo de elemento, por lo que hay que proceder a una resolución de conflicto. Esta resolución se hace con el procedimiento de cálculo de especificidad que explicaremos más adelante. La regla con mayor especificidad será la que se mostrará al visualizar la web.

Otro ejemplo de conflicto sería en un párrafo con id="atelier" y esta declaración en un archivo css:

```
p {color: red; text-decoration:none;}
#atelier {color: yellow;}
```

Ambas reglas son de aplicación directa al párrafo, por lo que la regla ganadora se determina mediante el cálculo de especificidad.

5) Si se llega a que dos reglas tienen igual especificidad, igual importancia e igual origen y se contradicen, prevalecerá la que esté escrita en último lugar (criterio de orden o de sobreescritura).

**EJERCICIO**

En el archivo CSS externo a un documento HTML encontramos esta declaración:

```
.magicTitle {
margin:0px;
width:20px;
height:30px;
color:green !important;
border-style: hidden !important;
}
```

© aprenderaprogramar.com, 2006-2029

Por otro lado dentro del código HTML tenemos el siguiente código:

```
<div class="magicTitle" style="border:1px solid blue; color: blue;">
Aprendiendo a programar
</div>
```

Responde a las siguientes preguntas:

a) ¿Se mostrará un borde para el div? ¿Por qué?

b) ¿Con qué color se mostrará el texto, con color verde o con color azul? ¿Por qué?

c) ¿Se respetarán los valores de width y height definidos en el archivo CSS externo o quedarán anulados? ¿Por qué?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01018D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**CÁLCULO DE LA ESPECIFICIDAD**

El mecanismo de cascada CSS determina que cuando diferentes reglas son de aplicación a un elemento éstas se ordenan en base a unos criterios. Si con el criterio de proximidad o de origen no se ha podido resolver el conflicto entre reglas se valora lo que se denomina especificidad. La especificidad para reglas que aplican igual de directamente a un elemento es un valor numérico que utiliza el navegador para ordenar reglas que entran en conflicto.

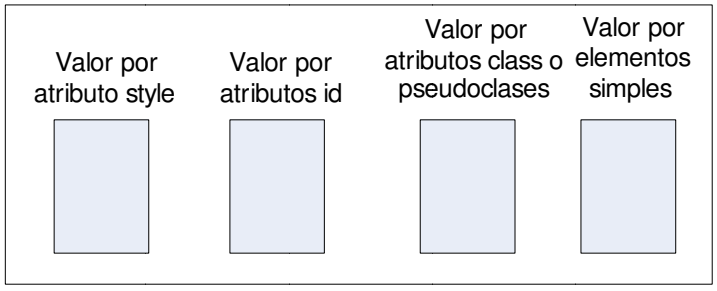
La especificidad se puede calcular como un número que consta de cuatro dígitos ABCD en el cual tenemos:

**A o primer dígito:** toma valor 1 cuando el estilo se declara en línea o cero en caso contrario.

**B o segundo dígito:** se calcula sumando 1 por cada identificador de tipo id que afecte a un elemento. Si una declaración es #menu1 #item1 {..} el valor del segundo dígito de especificidad para esta regla será 2, resultado de sumar 1+1, una unidad por cada id que afecte al elemento.

**C o tercer dígito:** se calcula sumando 1 por cada clase o pseudoclase que afecte a un elemento. Por ejemplo .destacado { ...} aporta un valor 1, mientras que .destacado .especial .suiter { ...} aporta un valor 3 resultado de sumar 1+1+1, una unidad por cada clase o pseudoclase.

**D o cuarto dígito:** se calcula sumando 1 por cada elemento HTML o pseudoelemento que aparezca en la declaración. Por ejemplo ul li a { ...} aporta un valor 3, resultado de sumar 1+1+1, una unidad por cada elemento HTML referenciado.




Una vez determinado cada dígito, se obtiene un valor numérico (por ejemplo 0112) que podemos ver como el peso de la regla. "Gana" la regla con mayor peso. Es interesante fijarse en que el uso de style siempre ganará a cualquier combinación de reglas, lo cual nos dice que habitualmente los estilos en línea ganarán. Una web bien construida debe prescindir en general del uso de estilos en línea, aunque comprobarás que por un motivo u otro es frecuente encontrarlos cuando se analizan desarrollos web existentes. Después de los estilos en línea, los estilos definidos para un id resultan ganadores respecto al resto. Finalmente, las clases o pseudoclases ganan a la definición de estilos para elementos simples HTML.

Como vimos anteriormente, la palabra clave !important puede introducir excepciones.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO RESUELTO**

Supongamos que tenemos el siguiente código HTML:

```
<div> <!—Ejemplo aprenderaprogramar.com-->
<div class="destacado">
<p> Aprender a programar es un objetivo que se plantea mucha gente y que no todos alcanzan.</p>
…</div> …</div>
```

Si en un archivo css tenemos las siguientes declaraciones, determinar cuál es la puntuación por especificidad cuando proceda, cuál es la regla ganadora, de qué color se visualizará el texto y por qué:

| Declaraciones | Puntuación especificidad | Regla ganadora, color del texto y por qué |
|---|---|---|
| body {color: grey;}<br>body div.destacado p {color: cyan;}<br>.destacado {color: green;}<br>div.destacado {color: blue;}<br>div.destacado p {color: yellow;}<br>div:first-child {color: magenta;} | | |
| body {color: grey;} | | |
| body {color: grey;}<br>.destacado {color: green;} | | |
| body div.destacado p {color: cyan;}<br>.destacado {color: green !important;} | | |
| body div.destacado p {color: cyan;}<br>.destacado {color: green;}<br>p {color: blue;} | | |
| body div.destacado p {color: cyan;}<br>.destacado {color: green;}<br>p {color: blue !important;} | | |

© aprenderaprogramar.com, 2006-2029

## SOLUCIÓN AL EJERCICIO PLANTEADO

Nota: no todos los navegadores responden igual a la palabra clave !important, por lo que podrías encontrar algunos navegadores en que la respuesta no fuera exactamente como hemos indicado.

| Declaraciones | Puntuación | Regla ganadora, color del texto y por qué |
|---|---|---|
| body {color: grey;}<br>body div.destacado p {color: cyan;}<br>.destacado {color: green;}<br>div.destacado {color: blue;}<br>div.destacado p {color: yellow;}<br>div:first-child {color: magenta;} | **No**<br>**0-0-1-3**<br>**No**<br>**No**<br>**0-0-1-2**<br>**0-0-1-1** | 1ª regla: no aplica directamente, si hiciéramos el cálculo body aporta 1 al cuarto dígito. Total 0001<br>2ª regla: body, div y p aportant 1+1+1 = 3 al cuarto dígito y .destacado aporta 1 al tercer dígito. Total 0013<br>3ª regla: no aplica directamente. Si lo calculáramos, 0010<br>4ª regla: no aplica directamente. Si lo calculáramos, 0011<br>5ª regla: div y p aportan 1+1=2 al cuarto dígito y .destacado aporta 1 al tercer dígito. Total 0012<br>6ª regla: la pseudoclase first-child aporta 1 al tercer dígito y div aporta 1 al cuarto dígito. Total 0011<br>Gana: la regla 2, el texto se vería de color cyan. |
| body {color: grey;} | **No** | Gana: la única regla que, por herencia, afecta. El texto se vería de color gris. Si calculáramos la especificidad sería 0001 |
| body {color: grey;}<br>.destacado {color: green;} | **No**<br>**0-0-1-0** | Gana: la segunda regla por ser la más próxima. El texto se vería de color verde. |
| body div.destacado p {color: cyan;}<br>.destacado {color: green !important;} | **0-0-1-3**<br>**No** | Gana: la primera regla. Aunque la segunda regla lleva la palabra clave !important es menos directa. La primera regla afecta directamente al elemento p por lo que tiene prevalencia. |
| body div.destacado p {color: cyan;}<br>.destacado {color: green;}<br>p {color: blue;} | **0-0-1-3**<br>**No**<br>**0-0-0-1** | Gana la primera regla, el texto se verá de color cyan. |
| body div.destacado p {color: cyan;}<br>.destacado {color: green;}<br>p {color: blue !important;} | **0-0-1-3**<br>**No**<br>**!important** | Gana la tercera regla: hay dos reglas que afectan directamente al elemento p, la primera y la tercera. Al tener la tercera la declaración !important significa que se sobreescriben el resto de reglas, independientemente de su valor de especificidad y origen. |

## CÓMO USAR !IMPORTANT EN CSS

Considera una declaración de este tipo. Pruébala sobre el código HTML de prueba que estamos usando para el curso:

```
body div.destacado p {color: cyan; text-decoration:underline;}
p {color: red !important;}
```

© aprenderaprogramar.com, 2006-2029

Aquí apreciamos dos cosas no del todo correctas. En primer lugar, por convenio los programadores y diseñadores web suelen poner las declaraciones más generales en primer lugar y las más específicas a continuación, tanto más abajo en el archivo o definición css cuanto más específicas sean. Esto facilita el análisis y comprensión de hojas de estilo. Por tanto cambiaríamos el orden de la declaración y pondríamos en primer lugar la declaración más general relativa a párrafos en general y en segundo lugar la otra declaración, más específica.

En segundo lugar, si queremos que los párrafos sean rojos: ¿Para qué declarar un color de párrafo cyan que luego anulamos con una declaración con la palabra clave !important? En un archivo CSS con cientos de líneas estas declaraciones generan confusión y dificultan el análisis del código. Cuando vemos cosas de este tipo analizando páginas web en general corresponden a que la persona que generó el código no tenía claros los conceptos de CSS ó a que se han realizado correcciones apresuradas en el código dejando inconsistencias. El problema está en que cuando una hoja de estilos se manipula múltiples veces añadiendo en cada ocasión más inconsistencias, se vuelve incoherente e inmanejable.

El código anterior queda más correcto así. Comprueba que obtienes el mismo resultado:

```
p {color: red;}
body div.destacado p {text-decoration:underline;}
```

Sólo debería usarse !important en casos concretos y en los que resulta estrictamente necesario. Usar la palabra clave !important con frecuencia anulando estilos repetidos es síntoma de un mal código CSS.

En general, para personas que se están iniciando con CSS recomendamos no usar !important en el código, posponiendo su uso para cuando se haya adquirido experiencia y un mayor nivel de destreza.

**CONCLUSIONES SOBRE LA CASCADA CSS**

El orden final con que se ordenan reglas en conflicto se basa en un proceso denominado “cascada”. El proceso es complejo, las reglas van cambiando en el tiempo y pueden existir diferencias entre navegadores. Por ello no resulta de interés aprender todas las reglas del proceso de cascada ni estar realizando continuamente cálculos para determinar precedencias. Sin embargo, sí resulta de interés comprender el concepto y las reglas básicas porque nos puede ayudar a resolver problemas que aparezcan en la visualización de páginas web. Con esto, más la aplicación de sentido común y la experiencia que iremos ganando a medida que trabajemos con CSS, será suficiente para crear páginas web con un código CSS de calidad.

**Próxima entrega:** CU01019D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## COLORES EN HTML Y CSS

Como es obvio, entre los aspectos importantes de las páginas web tenemos el color. El color interviene en el atractivo y facilidad de lectura que puede tener una página web. No vamos a hablar ahora de cómo se deben combinar los colores, sino de cómo se pueden especificar colores usando HTML y CSS. Hay distintas maneras de poner nombre a un color: notación RGB, códigos hexadecimales, uso de nombres específicos, etc.

## CÓDIGOS RGB DECIMALES

Los colores en los monitores de los ordendadores personales y dispositivos móviles como tablets, smartphones, etc. se forman por combinación de tres colores básicos: rojo (Red), verde (Green) y azul (Blue). La combinación de las letras de estos tres colores en inglés es RGB y así se denomina al sistema de construcción de colores basado en indicar la proporción de cada uno de estos tres colores. Hay otros sistemas como el CMYK que son de aplicación en el diseño y la industria gráfica y de edición, pero no en los desarrollos web.

Un color con el sistema RGB se escribe en CSS así:

```
rgb (cantidadDeRojo, cantidadDeVerde, CantidadDeAzul)
```

La cantidad de cada color se expresa entre un valor mínimo que es cero y un valor máximo que es 255. Si se indicara un número menor que cero se consideraría cero, o si se indica un número mayor que 255 se consideraría 255. El color rgb (0, 0, 0) es el negro y el color rgb (255, 255, 255) es el blanco. El color rojo será el rgb (255, 0, 0), el verde rgb (0, 255, 0) y el azul (0, 0, 255).

La mezcla completa de rojo y verde sin azul da lugar al amarillo: rgb (255, 255, 0).

La mezcla completa de verde y azul sin rojo da lugar al cyan: rgb (0, 255, 255).

El resto de colores se construyen mezclando estos colores básicos: por ejemplo el rgb (255, 100, 15) sabemos que será un color rojizo-amarillento porque hay mucho rojo, algo de verde y poco azul.

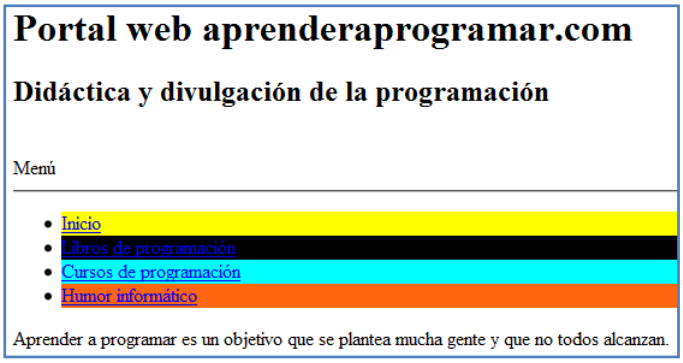
Sobre el código de prueba HTML que estamos usando para el curso, en concreto vamos a fijarnos en nuestra lista de elementos de menú:

```
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html" >Libros de programación</a> </li>
<li> <a href="cursos.html" >Cursos de programación</a> </li>
<li> <a href="humor.html" >Humor informático</a> </li>
</ul>
```

© aprenderaprogramar.com, 2006-2029

Prueba a aplicar estas reglas CSS y visualiza el resultado, que deberá ser similar al que mostramos en la imagen a continuación:

```
/* Curso CSS estilos aprenderaprogramar.com*/
li:first-child {background-color: rgb(255, 255, 0);}
li:nth-child(2){background-color: rgb(0, 0, 0);}
li:nth-child(3){background-color: rgb(0, 255, 255);}
li:nth-child(4){background-color: rgb(255, 100, 15);}
```




**CÓDIGOS RGB PORCENTUALES**

Los códigos RGB se pueden expresar en valores de tanto por ciento, correspondiendo el 0% a 0 y el 100% a 255 en la notación anterior. La sintaxis será:

```
rgb (cantidadDeRojo%, cantidadDeVerde%, CantidadDeAzul%)
```

El código del ejemplo anterior en notación porcentual se escribiría así:

```
/* Curso CSS estilos aprenderaprogramar.com*/
li:first-child {background-color: rgb(100%, 100%, 0%);}
li:nth-child(2){background-color: rgb(0%, 0%, 0%);}
li:nth-child(3){background-color: rgb(0%, 100%, 100%);}
li:nth-child(4){background-color: rgb(100%, 39.21%, 5.88%);}
```

Visualiza el resultado de este código y comprueba que sea igual que el anterior.

Nota: los valores de porcentajes decimales deben ir separados con un punto y no con una coma, por ejemplo rgb(100%, 39.21%, 5.88%) es válido pero rgb (100%, 39,21%, 5,88%) no es válido.

La equivalencia entre un valor numérico y el valor porcentual se puede calcular con una regla de tres "Si 255 equivale a 100, un valor equivale a ..." o lo que es lo mismo, usando la fórmula:

Porcentaje equivalente = Valor Numérico * (100/255) para pasar de valor numérico a porcentaje

Valor numérico = Porcentaje equivalente * (255/100) para pasar de valor numérico a porcentaje

Por ejemplo, el valor numérico 15 es un porcentaje de 15 x 100 / 255 = 5.88%

© aprenderaprogramar.com, 2006-2029

## CÓDIGOS HEXADECIMALES

Una de las notaciones soportada por las hojas de estilo CSS para colores, y hoy día la más usada, es la notación basada en códigos hexadecimales. Este sistema de códigos se basa en el alfabeto hexadecimal compuesto por 16 símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Ha de considerarse la equivalencia A = 10, B = 11, C = 12, D = 13, E = 14 y F = 15

Un código en base hexadecimal se traslada a numeración decimal utilizando potencias de 16. Por ejemplo el código 1B0DE equivale al número $1x16^4 + 11x16^3 + 0x16^2 + 13*16^1 + 14x16^0 = 65536 + 45056 + 0 + 208 + 14 = 110814$. El código A3 equivaldría a $10x16^1 + 3*16^0 = 160 + 3 = 163$

La notación hexadecimal sigue el sistema RGB: define la cantidad de color rojo, verde y azul que se aplica para formar un color, de ahí que también se le denomine notación RGB hexadecimal. Pero usa otra notación distinta a la que vimos para expresar colores RGB anteriormente, en concreto con notación hexadecimal un color se designa como:

#cantidadDeRojoHexadecimal**cantidadDeVerdeHexadecimal**CantidadDeAzulHexadecimal

Los códigos se expresan como el símbolo # seguido de 6 símbolos, correspondiendo los dos primeros a la cantidad de rojo expresada en notación hexadecimal, los dos intermedios a la cantidad de verde expresada en notación hexadecimal y los dos últimos a la cantidad de azul expresada en notación hexadecimal. Por ejemplo #FF4D21 es la notación de un color hexadecimal.

Los límites siguen siendo 0 como límite inferior, que en hexadecimal se expresa como 00, y 255 como límite superior, que en hexadecimal se expresa como FF (15x161 + 15x160 = 240 + 15 = 255).

La cantidad de cada color se expresa de la misma forma que en el sistema RGB: entre un valor mínimo que es cero y un valor máximo que es 255, pero los números se escriben en hexadecimal. Si se indicara un número hexadecimal menor que cero se consideraría cero, o si se indica un número hexadecimal mayor que 255 se consideraría 255. El color negro, rgb (0, 0, 0), se expresa como #000000 en hexadecimal. El color blanco, rgb (255, 255, 255), se expresa como #FFFFFF hexadecimal. El color rojo, rgb (255, 0, 0), se expresa como #FF0000 hexadecimal. El verde, rgb (0, 255, 0), es el #00FF00 hexadecimal y el azul rgb (0, 0, 255) es el #0000FF hexadecimal.

La mezcla completa de rojo y verde sin azul da lugar al amarillo, rgb (255, 255, 0) ó #FFFF00 hexadecimal.

Si tuviéramos que expresar un color intermedio como el rgb (255, 100, 15) tendremos que expresar 255, 100 y 15 en notación hexadecimal, resultando que 255 es FF en hexadecimal, 100 es 64 en hexadecimal y 15 es 0F en hexadecimal, por lo que el color es #FF150F.

Prueba este código CSS y comprueba que el resultado es el mismo que has obtenido con códigos rgb decimales y códigos rgb porcentuales anteriormente.

```
/* Curso CSS estilos aprenderaprogramar.com*/
li:first-child {background-color: #FFFF00;}
li:nth-child(2) {background-color: #000000;}
li:nth-child(3) {background-color: #00FFFF;}
li:nth-child(4) {background-color: #FF640F;}
```

© aprenderaprogramar.com, 2006-2029

## VARIACIONES PERMITIDAS EN CÓDIGOS HEXADECIMALES

Se admiten algunas variantes a la hora de definir un color hexadecimal en CSS.

Un color se puede designar con solo tres símbolos en lugar de seis. En este caso se considera que es una abreviación de la repetición de esos símbolos, por ejemplo #FFF se considera abreviación de #FFFFFF y #05F se considera abreviación de #0055FF.

También se admite el uso de letras en minúsculas en lugar de letras mayúsculas. De este modo, #ffffff equivale a #FFFFFF ó #0dab4f equivale a #0DAB4F.

El sistema más extendido es usar seis símbolos y letras mayúsculas, pero en ocasiones te encontrarás hojas de estilo donde se usan con frecuencia tres símbolos u hojas de estilo donde usan minúsculas. También en muchos casos se encuentran "mezclas", por ejemplo algunos colores designados con mayúsculas y otros con minúsculas.

Para aquellas personas que se inician con CSS recomendamos usar siempre la notación de 6 símbolos (evitando las abreviaturas) y letras mayúsculas.

## CONOCER A QUÉ COLOR CORRESPONDE UN CÓDIGO HEXADECIMAL

En ocasiones estaremos analizando una hoja de estilos y veremos un color hexadecimal como #F477A2 y querremos saber a qué color, visualmente, corresponde. Para ello basta escribir en un buscador como bing, google o yahoo "colores hexadecimal" y pinchar en cualquiera de las páginas que ofrecen el servicio de mostrar un color a partir de su código. Una vez introduzcamos el código, se nos mostrará el color que representa. El resultado para el color de ejemplo que hemos puesto será similar a este:

El mismo resultado podemos obtener usando un programa de diseño gráfico como Gimp ó Photoshop, que en sus herramientas de color permiten introducir códigos hexadecimales y nos muestran el color asociado a ese código.

© aprenderaprogramar.com, 2006-2029

## CONOCER EL CÓDIGO DE UN COLOR

En ocasiones tendremos una imagen en formato jpg, png, tiff, etc. y querremos conocer el código de alguno de los colores que aparecen en la imagen. Para ello abriremos la imagen con un programa de diseño gráfico como Gimp ó Photoshop, escogeremos la herramienta "recoge-color" y pulsaremos sobre la parte de la imagen donde está el color deseado. Una vez hecho esto, con la herramienta de color del programa ya podremos ver el código hexadecimal, RGB, HSV, etc. correspondiente a ese color.

En otros casos veremos una página web y querremos conocer qué color es el que aparece como fondo, borde, relleno, etc. de un elemento. Para ello podemos proceder de varias maneras:

a) Usar una herramienta recoge-color web (como ColorZilla, disponible para Mozilla Firefox y Google Chrome) que nos indica los códigos de color simplemente pinchando con el ratón sobre aquella parte de la web de la que queremos conocer el color.
b) Usar una herramienta de análisis web (como Firebug para Mozilla Firefox o Firebug Lite para el resto de navegadores) que nos permite obtener múltiple información sobre la web.
c) Hacer una captura de pantalla (print screen) y pegar la imagen obtenida en un programa de diseño gráfico como Gimp ó Photoshop. Una vez hecho esto, obtendremos el color como hemos explicado para una imagen cualquiera.

Estas herramientas son en algunos casos complejas y no vamos a profundizar en ellas de momento porque no las consideramos ahora de importancia para el desarrollo del curso. No obstante, nos ha parecido oportuno citarlas para ir teniendo unas nociones generales sobre ellas y para que aquellas personas que tengan curiosidad puedan investigar por su cuenta si lo desean.

## EJERCICIO

En la siguiente tabla están mal ordenados los códigos de colores. Ordénalos de forma que en cada fila estén los códigos equivalentes de forma ordenada y crea un documento HTML donde se muestre un cuadrado donde aparezca como texto el color con la notación empleada, y como color de fondo el color:

| Nombre | Hexadecimal | RGB |
|---|---|---|
| DimGray | 1E90FF | 178,34,34 |
| IndianRed | 696969 | 205,92,92 |
| FireBrick | CD5C5C | 30,144,255 |
| DodgerBlue | B22222 | 75,0,130 |
| Indigo | 4B0082 | 105,105,105 |

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01020D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**COLORES EN LA WEB**

Aunque la forma más habitual de designar colores en CSS es la notación hexadecimal escogiendo el color deseado entre los varios millones posibles, hay algunas curiosidades que merece la pena nombrar. Por un lado, la existencia de un conjunto de colores al que se denominaron "web safe colors" que fue usado durante varios años debido a que los monitores en que se visualizaban las páginas web no tenían capacidad para mostrar todos los colores.

**LOS WEB SAFE COLORS**

El número total de colores RGB se puede calcular teniendo en cuenta que cada uno de los tres colores admite 256 valores (desde 0 hasta 255), de ahí que el total de colores disponible sea exageradamente grande, en concreto de 256 x 256 x 256 = 16.777.216 colores. Cuando surgieron los desarrollos web y el auge de internet, la mayoría de los monitores sólo eran capaces de mostrar 256 colores diferentes. Por ello se definió una gama de colores (en concreto 216, ya que algunos se dejaron reservados para elementos de sistema) a los que se llamó "colores seguros para la web" o "web safe colors". Estos colores eran colores que se mostraban correctamente en todos los monitores, de ahí que se los llamara colores seguros, frente a otros colores que podían no mostrarse correctamente que eran inseguros.

Hoy día los monitores han evolucionado y ya tienen capacidad para mostrar millones de colores, con lo cual ha perdido sentido la clasificación de colores entre seguros e inseguros. La mayoría de los diseñadores usan el color que les apetece entre los millones disponibles y todos son seguros. No obstante, los web safe colors siguen siendo utilizados con preferencia por algunos diseñadores a los que les resulta cómodo usar una gama de colores básica o les gusta usar una gama de colores simple. También son usados por algunos diseñadores que crean diseños para dispositivos con limitaciones en la gama de colores. Si quieres consultar la lista con los 216 web safe colors puedes hacerlo en la dirección websafecolors.info

**NOTACIÓN RGBA**

Existen otras notaciones o sistemas de colores menos utilizadas pero que vamos a nombrar porque están admitidas para su uso en CSS y conviene al menos conocerlas brevemente.

La notación RGBA incorpora un parámetro adicional denominado "Alpha" y que indica el grado de transparencia que se aplica al color siendo 0.0 el valor mínimo, que significa completamente transparente, y 1.0 el valor máximo, que significa completamente opaco.

© aprenderaprogramar.com, 2006-2029

Un color con el sistema RGBA se escribe en CSS así:

```
rgba (cantidadDeRojo, cantidadDeVerde, CantidadDeAzul, valorDeAlpha)
```

Sobre el código de prueba HTML que estamos usando para el curso, volvemos a fijarnos en nuestra lista de elementos de menú:

```
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html" >Libros de programación</a> </li>
<li> <a href="cursos.html" >Cursos de programación</a> </li>
<li> <a href="humor.html" >Humor informático</a> </li>
</ul>
```

Prueba a aplicar estas reglas CSS y visualiza el resultado, que deberá ser similar al que mostramos más adelante:

```
/* Curso CSS estilos aprenderaprogramar.com*/
li:first-child {background-color: rgba(255,0,0,1);}
li:nth-child(2){background-color: rgba(255,0,0,0.66);}
li:nth-child(3){background-color: rgba(255,0,0,0.33);}
li:nth-child(4){background-color: rgba(255,0,0,0);}
```

Cuando tenemos un fondo blanco y vamos añadiendo transparencia el efecto que observamos es que el color parece que se va haciendo más claro. En realidad no es que el color se haga más claro, sino que al tener más transparencia deja ver cada vez más el fondo blanco resultándonos a la vista más claro.

Escribe la declaración .destacado {background-color: rgb(0,0,0);} y añade en el HTML código para indicar que la lista sea de esa clase: <ul class="destacado">. Con esto, comprueba la diferencia entre añadir transparencia teniendo fondo blanco o añadir transparencia teniendo fondo negro. En el primer caso el color se va haciendo cada vez más claro, en el segundo se va haciendo cada vez más oscuro.

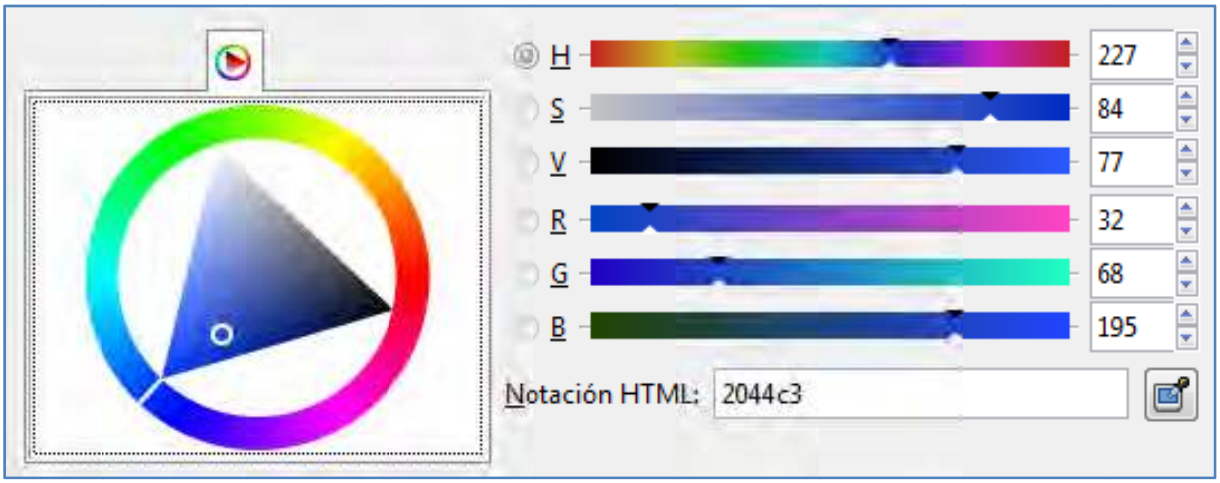
Nota: no todos los navegadores admiten la notación RGBA. En especial, navegadores antiguos no la soportan. El número de colores disponibles es más de 16 millones, añadiéndole la transparencia resulta un número de colores prácticamente infinito.

© aprenderaprogramar.com, 2006-2029

**NOTACIÓN HSL**

El sistema HSL usa una rueda de colores y tres parámetros, el primero un valor numérico y los dos siguientes un porcentaje, de modo que un color HSL se escribe:

```
hsl (anguloDeGiroColor, sombraGrisSaturación%, CantidadDeAzul, brillo%)
```




El primer parámetro o ángulo representa el color. En la rueda el ángulo de 0 grados (o 360 grados) representa el rojo, el ángulo de 120 grados el verde y el ángulo de 240 grados el azul.

El segundo parámetro o porcentaje de saturación representa la cantidad de color frente a sombra gris y el 100 % representa el color completo mientras que el 0% supone nada de color y todo de sombra gris. Todo color con saturación 0 es una sombra gris.

El tercer parámetro o porcentaje de brillo genera el negro si es 0% y color más claro (representaría la incidencia de luz sobre el color) cuanto más próximo está a 100% hasta llegar al blanco.

Prueba a visualizar el código HTML con esta definición de estilos:

```
/* Curso CSS estilos aprenderaprogramar.com*/
li:first-child {background-color: hsl(0,0%,50%);}
li:nth-child(2){background-color: hsl(0,50%,50%);}
li:nth-child(3){background-color: hsl(0,100%,50%);}
li:nth-child(4){background-color: hsl(58,100%,50%);}
```

El resultado debe ser la primera línea del menú gris, la segunda grisácea-rojiza, la tercera roja y la cuarta en un tono amarillo.

Nota: no todos los navegadores admiten la notación HSL. En especial, navegadores antiguos no la soportan.

© aprenderaprogramar.com, 2006-2029

**NOTACIÓN HSLA**

El sistema HSLA es respecto al HSL análogo a lo que es el sistema RGBA respecto al RGB, es decir, añade un parámetro adicional alpha que representa la transparencia y que toma un valor entre 0 y 1. La sintaxis es:

```
hsla (anguloDeGiroColor, sombraGrisSaturación%, CantidadDeAzul, brillo%, valorDeAlpha)
```

Los efectos que se generan son los mismos que hemos descrito para la notación RGBA. Recuerda que al añadir transparencia el efecto obtenido depende del color de fondo que tengamos. Si el fondo es blanco, la impresión que se genera es que el color se va aclarando. Si es oscuro, la impresión es que el color se va oscureciendo.

**NOMBRES DE COLORES ESTÁNDAR**

Los navegadores permiten el uso de nombres específicos para más de cien colores: además de los más conocidos (Red, Blue, Yellow, Brown, Pink, Green, etc.) existen nombres para muchos colores más. Por ejemplo BlueViolet es color azul violáceo, Tomato es color tomate, SkyBlue es color cielo. Se admite el uso de minúsculas y mayúsculas indistintamente: es válido tanto red como Red, ó BlueViolet como blueviolet.

No es un sistema muy extendido y es posible que algunos navegadores no reconozcan algunos nombres, pero algunos diseñadores lo usan y sobre todo se recurre a él cuando se quiere poner un color rápidamente y nos acordamos del nombre pero no del código hexadecimal. En general, se desaconseja su uso ya que es menos seguro que el uso de códigos de color.

A modo de curiosidad, ponemos a continuación una lista de nombres de colores.

aliceblue, antiquewhite, aqua, aquamarine, azure, beige, bisque, black, blanchedalmond, blue, blueviolet, brown, burlywood, cadetblue, chartreuse, chocolate, coral, cornflowerblue, cornsilk, crimson, cyan, darkblue, darkcyan, darkgoldenrod, darkgray, darkgreen, darkkhaki, darkmagenta, darkolivegreen, darkorange, darkorchid, darkred, darksalmon, darkseagreen, darkslateblue, darkslategray, darkturquoise, darkviolet, deeppink, deepskyblue, dimgray, dodgerblue, firebrick, floralwhite, forestgreen, fuchsia, gainsboro, ghostwhite, gold, goldenrod, gray, green, greenyellow, honeydew, hotpink, indianred, indigo, ivory, khaki, lavender, lavenderblush, lawngreen, lemonchiffon, lightblue, lightcoral, lightcyan, lightgoldenrodyellow, lightgray, lightgreen, lightpink, lightsalmon, lightseagreen, lightskyblue, lightslategray, lightsteelblue, lightyellow, lime, limegreen, linen, magenta, maroon, mediumaquamarine, mediumblue, mediumorchid, mediumpurple, mediumseagreen, mediumslateblue, mediumspringgreen, mediumturquoise, mediumvioletred, midnightblue, mintcream, mistyrose, moccasin, navajowhite, navy, oldlace, olive, olivedrab, orange, orangered, orchid, palegoldenrod, palegreen, paleturquoise, palevioletred, papayawhip, peachpuff, peru, pink, plum, powderblue, purple, red, rosybrown, royalblue, saddlebrown, salmon, sandybrown, seagreen, seashell, sienna, silver, skyblue, slateblue, slategray, snow, springgreen, steelblue, tan, teal, thistle, tomato, turquoise, violet, wheat, white, whitesmoke, yellow, yellowgreen

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Crea un documento HTML con 20 divisiones. En las diez primeras divisiones introduce el color RGB 178,34,34 con grados de transparencia desde 0.1 hasta 1.0 (en cada división un punto decimal más de opacidad). En las otras diez divisiones introduce el color RGB 218,165,32 con grados de transparencia desde 1.0 hasta 0.1 (en cada división un punto decimal más de transparencia). En cada división escribe el código de color y el grado de transparencia que muestra. Por ejemplo: <<RGB 178,34,34 con transparencia 0.6>>

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01021D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**BACKGROUND-COLOR CSS**

La propiedad background-color de CSS nos permite establecer un color de fondo en cualquier elemento delimitado dentro de una página web. Todo elemento tiene la propiedad color de fondo con valor por defecto "transparente" (trasnparent), por lo que muchas veces se piensa que no existe un color de fondo por defecto. En realidad sí existe, pero no se ve.

| PROPIEDAD CSS background-color | |
|---|---|
| **Valor por defecto** | transparent |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | Un color RGB ó RGBA p.ej. rgb (255, 0, 0) |
| | Un color hexadecimal p.ej. #FF0000 |
| | Un color HSL ó HSLA p.ej. hsl(0, 100%, 50%) |
| | Un color designado con nombre (p.ej. red) |
| | inherit (el color se heredará del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | h1 {background-color:#ffff00;}<br>.destacado {background-color: red;} |

La propiedad background-color es bastante simple. De hecho, ya la hemos utilizado para poner ejemplos básicos sobre selectores css y otras cuestiones a lo largo del curso, por lo que no vamos a extendernos en ella.

Un aspecto interesante de esta propiedad es que en ocasiones puede servirnos para visualizar las cajas que conforman una página web. Por ejemplo consideremos los elementos h1 y h2 que tenemos definidos en el código HTML que estamos usando para el curso:

```
<body>
<div>
<h1>Portal web aprenderaprogramar.com</h1>
<h2>Didáctica y divulgación de la programación</h2>
</div>
<br/>...
```

Si aplicamos estos estilos:

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {background-color: yellow;}
h1{background-color: blue;}
h2{background-color: red;}
```

© aprenderaprogramar.com, 2006-2029

Vemos con claridad cuáles son las cajas que intervienen. Si quisiéramos ver la caja div que envuelve a los títulos h1 y h2 bastaría con añadir div {background-color: pink;} con lo que veríamos algo así:

Al visualizar las cajas podemos entender mejor qué es lo que está ocurriendo en una página web. No siempre será necesario, de hecho cuando tengamos un poco de experiencia prácticamente las iremos visualizando mentalmente sin necesidad de aplicar un color de fondo, o usaremos una herramienta de análisis para estudiarlas, pero al comenzar a trabajar con CSS nos podrá servir de ayuda en algunas ocasiones.

Si analizamos la imagen y el resultado obtenido podemos ver algunas cosas curiosas:

a) A pesar de que en el código no tenemos ningún elemento que separe el div del body, el div aparece "separado" del body tanto por arriba como por la izquierda como por la derecha. Lo comprobamos porque se ve el fondo amarillo. ¿Por qué existe esa separación?
b) El h1 está perfectamente ajustado al div: no existe separación alguna ni por arriba ni por la izquierda ni por la derecha. El borde azul está perfectamente alineado con el borde rosado. En cambio, hacia abajo el h1 está separado del h2, si no existiera esa separación no se vería el fondo rosado. Sin embargo en el código no hemos especificado ningún elemento que dé lugar a esa separación entre h1 y h2. ¿Por qué aparece esta separación?

Aparentemente la única división "normal" entre elementos vendría dada por el <br/> que está debajo del div, y dado que el br es un elemento separador sería lógico que creara un espacio entre la caja del div y la siguiente caja dentro del documento HTML.

La razón para esos espacios que apreciamos se debe a los estilos por defecto que introduce el navegador para los elementos div, h1 y h2. Hablaremos de ello más adelante, de momento nuestro objetivo ha sido únicamente conocer esta propiedad y ver cómo nos puede ayudar a conocer el modelo de cajas CSS y el funcionamiento del navegador. Más adelante hablaremos de por qué aparecen esas separaciones entre elementos y de cómo podemos modificar o anular esos estilos que por defecto introducen los navegadores.

Recordar que se puede añadir un grado de transparencia a un color de fondo definiendo el color con notación rgba ó hsla como hemos explicado en apartados anteriores del curso.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Crea un documento HTML donde la ventana del navegador esté dividida en 6 partes, cada una con su borde. En cada división pon como color de fondo lo siguiente:

a) División 1: un color expresado con notación RGB.

b) División 2: un color expresado con notación RGBA.

c) División 3: un color expresado con notación hexadecimal.

d) División 3: un color expresado con notación HSL.

e) División 4: un color expresado con notación HSLA.

f) División 5: un color designado con un nombre.

En cada división incluye un texto con la notación y color empleado. Por ejemplo <<Notación RGB, color 218, 165, 32>>. Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01022D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## VERSIONES CSS

Comprobarás que a lo largo del curso no hemos dedicado apenas unas palabras a hablar de cuáles son las versiones de CSS. El motivo para ello es que con este curso pretendemos transmitir más la lógica de CSS y sus fundamentos que las especificidades concretas que tiene cada versión. No obstante, haremos ahora algunas reflexiones y comentarios sobre las versiones CSS.

## EL WORLD WIDE WEB CONSORTIUM O W3C

CSS es un lenguaje cuya definición ha sido creada por el World Wide Web Consortium, cuyas siglas son W3C. Se trata de un consorcio internacional que produce especificaciones, recomendaciones, manuales y herramientas en relación al desarrollo de internet. El W3C es un organismo que cuenta entre sus principales promotores al MIT (Massachussets Institute of Technology, USA), el ERCIM (European Research Consortium for Informatics and Mathematics, participado por numerosos países), la Keio University (Japón) y la Beihang University (China).

Uno de los objetivos principales del W3C es generar estándares: documentos donde se definen las sintaxis de lenguajes y protocolos que intervienen en el desarrollo de internet. El objetivo es promover que las empresas, instituciones y personas que participan o trabajan en desarrollos web utilicen un mismo lenguaje y se pongan de acuerdo a la hora de generar software y productos relacionados con internet. ¿Por qué decimos que la propiedad para dar color de fondo es background-color y no back_color ó bcol? Porque la forma de nombrar esta propiedad ha sido definida por el W3C de esta manera y todas las empresas, instituciones y personas lo han aceptado.

No siempre lo que propone el W3C es aceptado. El W3C emite propuestas, no leyes de obligado cumplimiento porque no tiene capacidad legal para ello. Hay otras instituciones o grandes empresas que también hacen propuestas o tienen criterios propios, y en ocasiones esas propuestas o criterios alternativos hacen que haya distintos grupos de trabajo y distintos estándares o forma de funcionamiento del software.

Por último, hay que tener en cuenta que el W3C está formado por un equipo de personas que también cometen errores y que "lo que dice el W3C" no siempre tiene por qué ser lo mejor ni lo más usado. No obstante, hoy día el W3C es la principal institución de referencia a la hora de crear y difundir estándares relacionados con los desarrollos web, entre ellos el CSS.

## LAS VERSIONES DE CSS

El W3C trabaja continuamente para mejorar el lenguaje CSS, corrigiendo errores e incorporando nuevas funcionalidades. Antes de llegar a una especificación o recomendación oficial se trabajan numerosos borradores que son sometidos a revisión y corrección. Cuando se alcanza un relativo grado de acuerdo entre los miembros del W3C se libera lo que se denomina una recomendación oficial de CSS ó versión a modo de propuesta para su uso y aplicación por todas las empresas, instituciones y personas.

© aprenderaprogramar.com, 2006-2029

Las versiones de CSS a lo largo de la historia han sido* :

CSS 1: publicada en 1996.

CSS 2: publicada en 1998.

CSS 2.1: publicada en 2004.

CSS 3: publicada en 2011.

CSS 4: se estima que pueda ser especificación oficial en 2019.

* Las fechas indicadas son sólo orientativas, la realidad es que una version no aparece un día, sino que tiene un largo proceso de desarrollo que a veces dura años. Con CSS 3 se introdujo una fuerte modularización o división por apartados de CSS, de modo que algunos módulos se encontraban en fase de borrador mientras otros se convertían en especificación oficial.

A la pregunta de ¿qué versión usar? damos la siguiente respuesta: la que sea de más amplia difusión en el momento en que estés haciendo un desarrollo web. Ten en cuenta que CSS en general va manteniendo su sintaxis y lógica con las diferentes versiones, y que en general una nueva versión mantiene las características de las anteriores y además introduce nuevas posibilidades. Por tanto tus conocimientos de CSS te seguirán valiendo aunque aparezcan nuevas versiones. Por ejemplo, algunas propiedades que aparecieron con CSS 1, entre ellas background-color, siguen usándose en CSS 3 y CSS 4. Otras propiedades aparecieron con CSS 2 ó 2.1, otras han aparecido con CSS 3 y otras con CSS 4.

Por otro lado, hay que tener en cuenta que "seguir con exactitud" una versión no significa que nuestra web vaya a funcionar perfectamente, debido a que no todos los navegadores reconocen todas las propiedades o sintaxis que se definen en una versión. También puede suceder que un navegador sí reconozca la sintaxis pero no ofrezca el mismo resultado que otro, lo cual da lugar a problemas en la visualización de páginas web.

Conseguir buenos resultados con CSS pasa por estar al día de la especificación del W3C pero también por seguir las novedades de la web, de los navegadores y siendo prácticos, por hacer muchas pruebas y comprobaciones con distintos navegadores o herramientas específicas para este fin.

**EJERCICIO**

Busca información en internet e indica para cada una de las siguientes propiedades CSS en qué versión de CSS fueron introducidas y si siguen estando vigentes en la actual versión de CSS: a) clip , b) font-weight c) overflow d) animation. Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01023D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## SERVICIOS DEL W3C

Hemos comentado que el estándar CSS es definido por el consorcio internacional W3C. El W3C ofrece algunos servicios de interés a través de internet, como la posibilidad de consultar la documentación oficial relacionada con CSS, una web divulgativa y con ejemplos sobre la aplicación de CSS en la práctica y la validación (comprobación de cumplimiento del estándar) CSS de páginas web.

## DOCUMENTACIÓN OFICIAL CSS

La web oficial del consorcio W3C es http://www.w3.org/. A través de esta web podemos acceder a distintos contenidos del trabajo de la W3C, que no solo tratan sobre CSS, sino en general sobre las tecnologías relacionadas con el desarrollo web como HTML, CSS, Javascript, gráficos, audio y video en entornos web, protocolos, etc.

Desde la página principal de W3C podemos navegar hasta llegar a la página oficial CSS (cuya URL será similar a esta: http://www.w3.org/Style/CSS/) donde podremos encontrar numerosa información sobre CSS (en inglés). Dentro de esta información están disponibles las especificaciones oficiales CSS, por ejemplo en http://www.w3.org/TR/selectors/ podemos consultar la especificación oficial de selectores para CSS. El principal problema de recurrir a la documentación oficial para el aprendizaje es que resulta demasiado extensa y demasiado técnica. También existen numerosos documentos técnicos, algunos de los cuales son recomendaciones, otros pre-recomendaciones (se denominan candidatos a recomendación) y otros borradores. Este exceso de información y su forma de presentación resulta poco útil de cara a adquirir una formación práctica y básica sobre CSS. No obstante, conviene conocer su existencia porque puede ser necesaria su consulta en algún momento.

## RECURSOS DIDÁCTICOS SOBRE CSS

Indicado ya que la documentación oficial CSS puede ser demasiado árida o extensa, ¿qué recursos existen que sean más didácticos y educativos? Existen numerosas alternativas.

En primer lugar señalaremos la página web http://www.w3schools.com/. Esta plataforma alberga tutoriales y documentación sobre numerosas tecnologías relacionadas con el desarrollo web como HTML, CSS, Javascript, AJAX, PHP, SQL y muchas otras. Contrariamente a lo que pueda parecer, esta web pertenece a una empresa privada y no está en absoluto relacionada con el consorcio W3C. La similitud del nombre de dominio resulta, cuando menos, confusa. Tampoco hay indicaciones explícitas en la plataforma de la no-relación con el W3C. Los resultados de w3schools suelen aparecer en los primeros lugares cuando se hacen consultas en buscadores, de ahí que muchos usuarios o profesionales relacionados con internet piensen que w3schools es un servicio de W3C, cuando la realidad es que no es así. Debido a estos hechos y a otros aspectos negativos que se pueden achacar a w3schools han surgido voces críticas que recomiendan no usar esta web, entre los que podemos destacar http://www.w3fools.com/.

© aprenderaprogramar.com, 2006-2029

Nosotros no recomendamos el uso de w3schools, pero tampoco lo desaconsejamos. Simplemente pensamos que debe conocerse qué es y cuáles son sus ventajas e inconvenientes. Como ventajas señalaremos estas:

- Se trata de una plataforma web con una buena estructuración de los contenidos y una buena presentación.
- Tiene un sistema de contenidos, ejemplos y pruebas basadas en la edición de código on-line que en general resultan bastante didácticos y facilitan el aprendizaje.
- Es accesible gratuitamente.

Como inconvenientes señaleremos los siguientes:

- Parecen aprovecharse de la confusión y similitud de nombre con el W3C, aparentando ofrecer información oficial cuando no la es.
- Ofrecen cursos y certificados previo pago que no están avalados por organismos o empresas con garantías suficientes. Su publicidad puede resultar engañosa.
- En general se trata de webs bastante cargadas de publicidad que se entremezcla con los contenidos, llegando a resultar molesta.
- Parte de la información que ofrece esta web puede ser no correcta o estar no actualizada.

La mayor parte de los desarrolladores web o diseñadores web han utilizado en algún momento w3schools. Nosotros simplemente te aconsejamos que si la usas, conozcas qué es, sus ventajas e inconvenientes.

A continuación vamos a ofrecer un listado con recursos didácticos sobre CSS:

| Recurso | URL |
|---|---|
| **Tutorial CSS de aprenderaprogramar.com** | http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203 |
| **Documentación CSS de Mozilla** | https://developer.mozilla.org/en-US/docs/Web/CSS |
| **Wiki de la W3C para CSS** | http://www.w3.org/wiki/CSS |
| **Documentación CSS de webplatform** | http://docs.webplatform.org/wiki/css |
| **Documentación CSS de sitepoint** | http://reference.sitepoint.com/css |
| **Documentación CSS de w3schools** | http://www.w3schools.com/css/ |

Nuestra recomendación es que para el aprendizaje inicial de CSS uses un solo curso y trates de seguirlo de forma completa y hasta el final, consultando otras fuentes sólo para cuestiones puntuales. Si tratas de adquirir una formación básica recurriendo a varias fuentes de información al mismo tiempo es posible que acabes perdiendo el tiempo y haciéndote un lío. Una vez tengas las bases de CSS y algo de experiencia, será más fácil moverte consultando la multitud de recursos e información que existen en libros, revistas y páginas web.

© aprenderaprogramar.com, 2006-2029

## VALIDACIÓN CSS CON W3C VALIDATOR

Hay un servicio de la W3C que puede resultar de relativo interés: el W3C CSS validator o validador CSS de la W3C. Se accede a través de w3.org escogiendo la opción validators: CSS. También se puede acceder directamente en la URL http://jigsaw.w3.org/css-validator/

Una vez en la página con el formulario de entrada, basta introducir una URL para que el contenido de esa página web sea sometido a análisis y aparezcan los resultados del mismo. Si el CSS de la web cumple con la especificación oficial aparece un mensaje informativo y se nos ofrece la posibilidad de insertar los logos de validación W3C en nuestra página web.

En caso de que la página web no cumpla la especificación, se nos indican cuántas faltas se han encontrado y cuáles son. Las faltas se dividen en faltas menores (warnings o advertencias) y faltas mayores o errores.

El servicio de validación CSS puede ser una herramienta útil para personas que están aprendiendo y para profesionales, al permitirles identificar en qué partes de su código no están cumpliendo con la especificación oficial. No obstante, te recomendamos que no pierdas demasiado tiempo tratando de que la página web que estés creando tenga cero errores por un motivo simple: no tener errores en el validador no garantiza que tu web se vaya a visualizar correctamente. Muchas personas pierden horas tratando de corregir el código para pasar la validación W3C cuando su web no tenía ningún problema de visualización. Entonces, ¿para qué perder el tiempo en ello? Sí, siempre es "bonito" pasar limpiamente un validador, la cuestión que debemos tener en mente es si merece la pena el tiempo que vayamos a invertir.

© aprenderaprogramar.com, 2006-2029

A modo de curiosidad, piensa en algunas páginas web importantes que conozcas (por ejemplo de algún diario, alguna televisión, alguna empresa importante) y pásale el validador CSS de W3C. Encontrarás que muchas de ellas no superan la validación, y sin embargo son webs de prestigio y con millones de visitas. Si estas grandes webs no se preocupan de tener una validación W3C perfecta ¿merece la pena obsesionarse con no tener errores de validación CSS en nuestras páginas web? A modo de curiosidad te diremos que en el momento de escribir este curso hemos pasado el validador CSS a facebook.com y hemos obtenido como resultado 24 errores. En la web oficial del gobierno de Estados Unidos, 102 errores, y así podríamos continuar una larga lista de webs importantes que no superan la validación. Pero repetimos: lo importante no es superar la validación, lo importante es que la web funcione y se vea correctamente.

Sobre los logos del W3C de validación, su presencia en una página web no asegura que la página web pase el validador, ya que se trata de simples imágenes que cualquiera podría insertar en su web. La forma de saber si una web cumple la validación es pasar su URL por el validador CSS de la W3C.

**EJERCICIO**

Crea un documento HTML que conste de los siguientes elementos: un título h1 con el texto "Aprendizaje de la programación". Una división div con id menú que contenga: un título h2 con el texto "Menú" y una lista de elementos no ordenados (ol) con los siguientes items: Programación básica, Programación intermedia y Programación avanzada. Finalmente, una división div con id footer con el texto "Curso aprenderaprogramar.com". Establece diferentes valores de las propiedades color, background-color y font-size para:

a) Los elementos h1

b) Los elementos h2

c) Los elementos ol y partes internas a este (elementos li de listas).

d) El elemento con id footer.

A continuación utiliza el W3C validator y comprueba si no te indica ningún error, o qué errores o warnings te indica y trata de identificar el por qué de ellos y corregirlos. Puedes comprobar si tus respuestas son correctas consultando en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01024D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## UNIDADES DE MEDIDA

Cuando trabajamos con CSS necesitaremos especificar con frecuencia anchura, altura, márgenes o tamaño de distintos elementos como cajas contenedoras de distintos tipos, imágenes, texto, etc. CSS permite la especificación de valores relacionados con el tamaño de distintas maneras para facilitar el trabajo a los programadores y diseñadores web.

Las unidades de medida CSS se dividen en dos grupos: unidades de medida relativas y unidades de medida absolutas. También se habla a veces de unidades híbridas.

Las unidades de medida relativas son aquellas que determinan un tamaño en función de otro tamaño. Por ejemplo si indicamos que el ancho de un elemento como 50%, dicho porcentaje tiene que estar referido al 50% de algo ¿de qué? En el caso de un ancho del 50% puede estar referido normalmente al 50% del ancho que tenga el contenedor del elemento. Las unidades relativas aluden a un elemento de referencia. Veremos que existen distintos elementos que se toman como referencia.

Las unidades de medida absolutas son aquellas que determinan el tamaño de algo de forma concreta, específica y medible. Por ejemplo si decimos que un tamaño de un div debe ser de 150 mm de alto, no hay duda al respecto. El div tomará exactamente ese valor (aunque puede haber pequeñas distorsiones en función del dispositivo en que se visualice) y si cogemos una regla escolar y lo medimos, el div tendrá esa medida (aproximadamente 150 mm).

Las unidades de medida se aplican a numerosas propiedades CSS de las que empezaremos destacando dos de ellas: width (ancho de un elemento) y height (alto de un elemento). La sintaxis a emplear es la siguiente:

```
selectorDeElemento {width: valorDePropiedadUnidades;
                    height: valorDePropiedadUnidades;}
```

## UNIDADES DE MEDIDA ABSOLUTAS

A continuación indicamos las unidades de medida absolutas disponibles en CSS:

| Unidad | Símbolo | Ejemplo aprenderaprogramar.com | Observaciones |
|---|---|---|---|
| **Pulgada** | in | div {background-color:pink; width: 7in;} | 1 pulgada son 25.4 mm |
| **Centímetro** | cm | div{background-color:pink; width: 20cm;} | |
| **Milímetro** | mm | div{background-color:pink; width: 1000mm;} | |

© aprenderaprogramar.com, 2006-2029

| Unidad | Símbolo | Ejemplo aprenderaprogramar.com | Observaciones |
|---|---|---|---|
| **Punto** | pt | div{background-color:pink; font-size: 24pt;} | 1 punto equivale a 1/72 pulgadas ó aprox. 0.35 mm |
| **Pica** | pc | div{background-color:pink; font-size: 2pc;} | 1 pica equivale a 12 puntos o aprox. 4.23 mm |
| **Pixel*** | px | div{background-color:pink; font-size: 24px;} | Tantas veces los puntos visibles que tenga la pantalla donde se visualice la página web como se especifique.<br>*Puede considerarse una unidad híbrida, ni absoluta ni relativa. |

Prueba a aplicar estas reglas sobre el código HTML de prueba que estamos usando en el curso.

Las unidades de medida absolutas en general podemos decir que se utilizan poco en desarrollos web, excepto el pixel que es una unidad especial y, en menor medida, el punto. Pulgadas, milímetros, picas, etc. tienen más sentido en trabajos de impresión sobre papel. El pixel tiene aplicaciones diversas mientras que la unidad pt se usa a veces para definir el tamaño de letra (tamaño de fuente).

El motivo para que las unidades absolutas se usen poco es que no permiten adaptarse al tamaño de la pantalla que utilice cada usuario. Hoy día, existen grandes pantallas (por ejemplo de 32 pulgadas de unos 480 x 730 mm) pero también pequeñas pantallas (por ejemplo smartphones con pantallas de 7 pulgadas de unos 90x170 mm de tamaño). Si queremos mostrar un contenido y lo definimos en unidades absolutas, por ejemplo con un ancho de 300 mm y un alto de 200 mm, cuando lo veamos en un monitor grande nos parecerá pequeño; por el contrario cuando lo veamos en un dispositivo pequeño como una tablet nos aparecerá cortado por ser excesivamente grande. Si en vez de unidades absolutas indicamos una unidad relativa como 100%, el contenido se adaptará al ancho disponible para cada dispositivo, lo cual resulta ventajoso. En realidad la gran variación de tamaño de los dispositivos supone un problema que no es nada fácil de solventar, ni siquiera utilizando unidades de medida relativas. Será uno de los campos en que la experiencia y la práctica resulten fundamentales en el trabajo con desarrollos web.

El pixel es una unidad de medida un tanto especial. Comprobaremos que se alude a esta medida como una unidad relativa, absoluta o híbrida según dónde consultemos. La realidad es que puede considerársela de cualquiera de estas tres maneras según dónde centremos nuestra atención. En nuestra opinión es más "justo" considerarla unidad relativa porque no tiene un valor absoluto y único, sino que presenta variaciones según los dispositivos. No obstante, estas variaciones son relativamente pequeñas o moderadas, de ahí que muchas veces se lo considere una unidad absoluta.

Una pantalla se divide en miles de pequeños puntos visibles denominados pixeles. Cuando distintos grupos de pixeles toman distintos colores se forman las letras, imágenes o formas que podemos ver en las pantallas. Por ejemplo si situamos pixeles negros uno junto a otro en horizontal sobre un fondo blanco, se acaba formando una línea negra horizontal. Si en vez de en horizontal cada pixel está ligeramente desplazado hacia la derecha y ligeramente por encima del anterior se formará una línea negra inclinada. Por el mismo principio se forman letras, imágenes, figuras, etc.

© aprenderaprogramar.com, 2006-2029

Las pantallas suelen tener una resolución expresada expresada en pixeles, por ejemplo 1280 x 800 px, donde el primer valor indica el número de puntos horizontales y el segundo valor el número de puntos verticales en que se divide la pantalla (que podemos ver como una cuadrícula de minúsculos puntitos).

El tamaño real del pixel considerado como "un cuadradito muy pequeño" puede variar entre dispositivos, por ejemplo podremos encontrar dispositivos donde un pixel mida 0.20 mm y otros donde mida 0.30 mm. Las diferencias entre dispositivos pueden ser pequeñas cuando hablamos de tamaños de pantalla similares, pero se hacen apreciables para tamaños de pantalla muy diferentes.

El análisis teórico de las resoluciones de los dispositivos y unidades de medida puede ser extenso, por lo que nosotros vamos a optar por considerar al pixel, desde el punto de vista práctico, una unidad de medida híbrida. Absoluta en cuanto a que define unas dimensiones más o menos exactas para dispositivos similares (como monitores de ordenadores), relativa en cuanto a que sus dimensiones varían cuando cambiamos el tipo de dispositivo (por ejemplo en un smartphone el pixel tendrá distinta dimensión que en un monitor de 32"). Esta definición puede ser poco exacta desde el punto de vista teórico, pero a efectos prácticos nos va a resultar suficiente para trabajar en desarrollos web. Consideraremos el pixel como un pequeño puntito visible de una pantalla.

El pixel es una unidad de medida no tan rígida como pueda ser el mm y no tan flexible como pueda ser el % u otras unidades relativas. Permite fijar tamaños de forma relativamente precisa, pero a su vez con un cierto grado de adaptación al dispositivo en que se visualiza una web. Se usa con bastante frecuencia en distintos aspectos del diseño como iremos comprobando a lo largo del curso.

**UNIDADES DE MEDIDA RELATIVAS**

A continuación indicamos las unidades de medida relativas disponibles en CSS:

| Unidad | Símbolo | Ejemplo | Observaciones aprenderaprogramar.com |
|---|---|---|---|
| **Porcentaje** | % | #menu1 {width: 50%;} | Porcentaje relativo al elemento contenedor. |
| **Relativa al tamaño de letra** | em | #menu1 {font-size: 2.65em;} | Tantas veces el tamaño que sea de aplicación como se indique. Por ejemplo si el tamaño de letra de aplicación es 12 pixeles, 1 em son 12px, 2 em son 24 px, 3 em son 36 px, etc. |
| **Relativo a la x minúscula** | ex | #menu1 {font-size: 2.65ex;} | Tantas veces la altura de la letra x minúscula como se especifique. Por ejemplo si la x minúscula que se debería mostrar mide 10mm, 1 ex son 10 mm, 1.5ex son 15mm, 2ex son 20mm, etc. |
| **Pixel*** | px | #menu1 {font-size: 24px;} | Tantas veces los puntos visibles que tenga la pantalla donde se visualice la página web como se especifique.<br>*Puede considerarse una unidad híbrida, ni absoluta ni relativa. |

© aprenderaprogramar.com, 2006-2029

**¿QUÉ UNIDAD USAR?**

Las unidades estrictamente absolutas como el mm son poco habituales en los diseños web y podemos decir que prácticamente no se usan.

Son de uso muy habitual unidades relativas como el % y el em porque permiten la adaptación flexible a los distintos tamaños de pantalla de los dispositivos. El % nos permite establecer proporciones adecuadas en relación a la pantalla y el em en relación al tamaño de texto que se esté usando.

Por último el pixel es también muy usado pues permite que determinados elementos de la web se visualicen con tamaños bastante precisos, evitando que la web tenga cambios drásticos de aspecto cuando se cambie de dispositivo.

Usa preferentemente %, em y px en aquellas hojas de estilo que tengas que crear. Usa otras unidades cuando razonadamente resulten necesarias.

**EJERCICIO**

Los navegadores aplican un tamaño por defecto a los títulos h1 a h6, aunque estos tamaños no son siempre los mismos sino que dependen del navegador que se emplee. Para el navegador que estés empleando, divide la página web en una cuadrícula con 5 espacios a lo ancho y 6 espacios a lo alto, tal y como se ve en la siguiente tabla. En cada espacio, muestra un texto y la forma en que se ha definido el tamaño. Por ejemplo: h1 (predefinido), h1 (px), h1 (em), h1 (pt), h1 (%). El resultado ha de ser que todo se vea del mismo tamaño y modo, pero sin embargo la forma de definir el estilo será diferente en cada caso. Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

Orientación: h1 suele ser 24px ó 2em en la mayoría de los casos, pero tendrás que comprobarlo en el navegador que estés empleando.

| Título | Tamaño px | Tamaño em | Tamaño pt | Tamaño % |
|---|---|---|---|---|
| h1 | | | | |
| h2 | | | | |
| h3 | | | | |
| h4 | | | | |
| h5 | | | | |
| h6 | | | | |

**Próxima entrega:** CU01025D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## WIDTH Y HEIGHT

Dos propiedades básicas en CSS son width (ancho) y height (alto). Las usaremos con frecuencia para determinar el tamaño de muchos elementos en nuestro diseño CSS. Estas propiedades serán designadas con alguna de las unidades de medida que hemos estudiado previamente, usando la que resulte más adecuada en función del elemento al que se vayan a aplicar.

| PROPIEDADES CSS width y height | |
|---|---|
| **Valor por defecto** | auto (en este caso, es determinado por el navegador) |
| **Aplicable a** | Elementos tipo block y elementos insertados en una posición que son reemplazados por un objeto (entre ellos img, input, textarea, select, object) |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | Una unidad de medida absoluta como in, cm, mm, pt, ó pc |
| | Una unidad de medida relativa como %, em ó ex |
| | Una unidad de medida híbrida como px (pixel) |
| | inherit (indica que se hereda del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | html {height: 100%;}<br>body {background-color: yellow; width: 900px;}<br>.logo-module {width: 16em;} |

Recordemos que la sintaxis a emplear es la siguiente:

```
selectorDeElemento {width: valorDePropiedadUnidades;
                    height: valorDePropiedadUnidades;}
```

Un aspecto importante es que las propiedades width y height no son aplicables a cualquier elemento. Por ejemplo elementos de tipo inline como  ...  no admiten estas propiedades. Si intentamos aplicar width o height a un elemento que no lo admite, el navegador simplemente lo ignorará.

Las propiedades width y height no siempre se especifican. Se puede especificar una de ellas, ambas o ninguna. Para aquella propiedad que no se especifica, el elemento tomará como anchura y altura el valor que por defecto o por herencia le sean de aplicación.

© aprenderaprogramar.com, 2006-2029

No debe existir espacio entre el valor establecido y las unidades cuando escribamos el código. Por ejemplo si establecemos #menu1 {width: 80%;} no debe existir espacio entre 80 y %. Si establecemos #menu1 {width: 600px;} no debe existir espacio entre 600 y px. En el caso de unidades que se expresen con decimales, la separación decimal debe indicarse con un punto en lugar de con una coma. Por ejemplo #menu1 {font-size: 12.75pt;} es correcto pero #menu1 {font-size: 12,75pt;} es incorrecto y no será interpretado por el navegador.

En el caso de decimales inferiores a uno, se admite omitir el 0 previo al punto decimal o el 0 de terminación. Por ejemplo sería tan válido escribir #margen1 {widht: 0.90%;} como #margen1 {widht: .9%;} como #margen1 {widht: .90%;}

En el caso de valores cero (nulo), no es necesario indicar unidades de medida. Por ejemplo #soundCode1 {widht: 0px;} es igual de válido que #soundCode1 {widht: 0;}

**EJEMPLO PRÁCTICO**

Define los siguientes estilos para el documento HTML que estamos usando como base de ejemplo para el curso y visualiza el resultado:

```
/* Curso CSS estilos aprenderaprogramar.com*/
html{background-color: pink;}
body {background-color: yellow; width: 800px;}
h1{background-color: blue; width: 50%;}
h2{background-color: red; width: 8em}
```

Con los conocimientos que hemos ido adquiriendo a lo largo del curso ya hemos de ser capaces de interpretar qué es lo que visualizamos. Vamos a hacerlo. El fondo se muestra rosa porque hemos definido que el background-color del elemento html sea rosado. En amarillo vemos el elemento body. El ancho del elemento body es de 800 px. El elemento body no tiene el mismo ancho que el elemento html por dos motivos: por un lado, la pantalla donde se ha visualizado la web para capturar la imagen que hemos mostrado tiene una resolución de 1024 x 768 pixeles. Como nosotros hemos definido que el

© aprenderaprogramar.com, 2006-2029

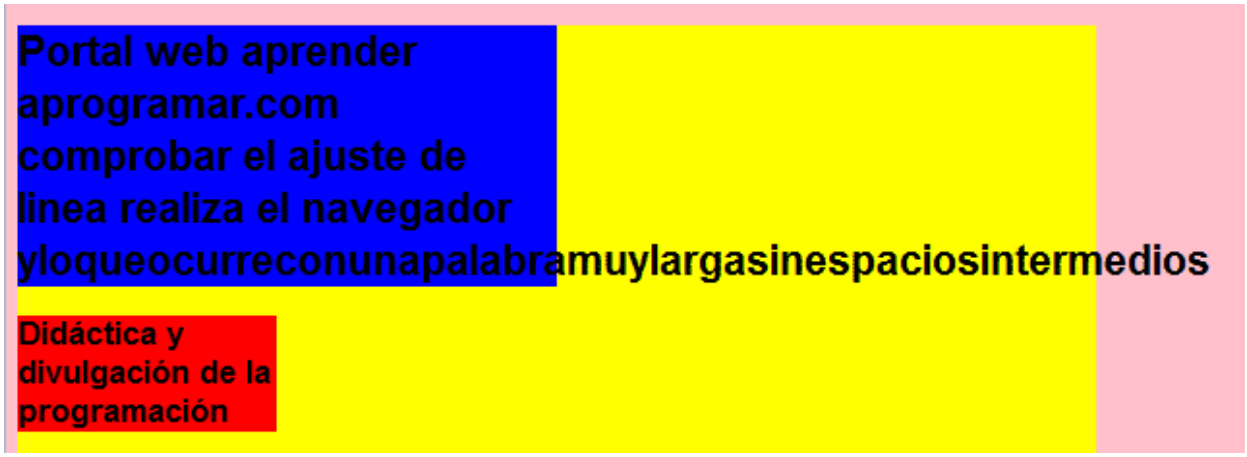
elemento body tendrá 800 px, el resto que falta hasta los 1024 px disponibles se ven con el color de fondo del elemento html. Otro motivo por el que el elemento body no tiene el mismo ancho que el elemento html es que el navegador introduce unos márgenes por defecto (ya veremos más adelante cómo se pueden anular). Prueba a cambiar el width de body y ponle distintos valores como 200 px y 2400 px. Para valores de ancho que exceden el tamaño máximo en pixeles de anchura de la pantalla el navegador incorporará automáticamente unas barras de desplazamiento o scrolls.

El elemento h1 con el texto "Portal web aprenderaprogramar.com" tiene un ancho del 50%. ¿Pero el 50% de qué? Pues el 50% de su elemento contenedor (elemento dentro del cual está o elemento padre). Como su elemento contenedor es body y éste tiene 800 px de ancho, el ancho de h1 es la mitad de este valor, es decir, 400 px.

Con el elemento h1 comprobamos una cosa: que el texto se muestra en dos líneas y que el tamaño del texto de la segunda línea excede en anchura al tamaño de su elemento contenedor, el h1 con anchura de 400 px. Por tanto ya sabemos que el texto ocupa algo más de 400 px y por ello se sale del espacio azul y ocupa parte del espacio amarillo. ¿No sería más lógico que el texto se cortara cuando llegara al límite de su contenedor? ¿O que siguiera en la siguiente línea en vez de salirse hacia la derecha? Quizás, pero el comportamiento que observamos está determinado por el navegador que estamos usando. Se está aplicando la siguiente regla: si el texto excede el ancho de su elemento contenedor, se crean nuevas líneas aumentando la altura (height) automáticamente siempre que existan espacios dentro del texto que permitan introducir el salto de línea. Si no existe espacio para introducir ese salto de línea, el texto se expande hacia la derecha sobrepasando el tamaño de su contenedor.

El elemento h2 tiene fijado un width de 8 em. El em es una medida relativa al tamaño de letra que se esté usando. ¿A qué tamaño de letra en este caso? Pues al tamaño de letra del h2, que como no lo hemos especificado, lo determina por defecto el navegador. El tamaño por defecto puede variar entre navegadores, pero de forma aproximada podemos decir que serán similares a estos: 16px (1em o 100%) como tamaño normal de fuente, h1 de tamaño 2.25em (36px) y h2 de 1.5em (24px). En este caso hemos comprobado que el navegador que usamos aplica 24px de tamaño al h2, entonces 8em se interpreta como 8 veces el tamaño de fuente del elemento, resultando 8x24 = 192 pixeles.

Para verificar que el navegador maneja la propiedad height automáticamente (a falta de especificación por nuestra parte su valor es **auto**) y lo que ocurre cuando el texto no se puede ajustar al espacio disponible fíjate en la siguiente imagen que refleja lo que ocurre al escribir un texto más largo:




© aprenderaprogramar.com, 2006-2029

Como vemos el texto se sale del elemento contenedor h1, superponiéndose sobre el elemento padre, body, y al tener más ancho aún se expande hasta ocupar espacio del elemento padre del padre o "abuelo", el elemento html. Si el texto se alargara aún más el navegador introduciría un scroll para permitir que se pueda leer.

Para entender mejor la diferencia entre que width y height estén especificados con un valor o que no lo estén y mantengan el valor por defecto "auto" fíjate en la siguiente imagen. Hemos escrito un texto muy largo en el h1 y en el lado izquierdo vemos lo que ocurre si definimos h1{background-color: blue; width: 50%;} mientras que en el lado derecho vemos el resultado para el código h1{background-color: blue; width: 50%; height: 150px;}

En el lado izquierdo vemos cómo el navegador adapta automáticamente el valor height del h1 para que el texto no se salga de él. Al no estar especificada la propiedad height tiene valor "auto". En el lado derecho, al tener restringida la altura del h1 a 150 pixeles, cuando el texto no cabe en su contenedor continúa expandiéndose hacia abajo.

Lo que nos interesa de todo esto es ser capaces de interpretar lo que ocurre cuando se obtiene una visualización errónea o distorsionada debido al código CSS. Hay que tener en cuenta, sin embargo, que los problemas de visualización en páginas web pueden tener otros orígenes distintos del CSS como el código de programación, configuraciones de servidor o navegador, etc.

**EJERCICIO RESUELTO**

Crea una hoja de estilos css de forma tal que el elemento body tenga el 100% del ancho de la pantalla, los elementos h1 un 80% y los elementos h2 un 60%. Escribe un CSS para obtener la misma visualización pero expresando los valores en pixeles.

© aprenderaprogramar.com, 2006-2029

**SOLUCIÓN**

A continuación proponemos una solución. Si tu respuesta no coincide con lo que indicamos o tienes dudas, puedes consultar en los foros aprenderaprogramar.com.

La definición de estilos en % sería la siguiente. Hemos añadido colores de fondo para que se visualice mejor:

```
/* Curso CSS estilos aprenderaprogramar.com*/
html {background-color: pink; }
body {background-color: yellow; width: 100%;}
h1 {background-color: blue; width: 80%;}
h2 {background-color: red; width: 60%;}
```

Comprueba el resultado al aplicar estos estilos al código HTML base que estamos usando para el curso. Si quisiéramos realizar una definición equivalente en pixeles, tendríamos que consultar la resolución en pixeles de la pantalla con que estamos trabajando (en Windows se puede ver yendo al botón Inicio, luego "Panel de Control" y luego "Ajustar resolución de pantalla")

Supongamos que estamos en un monitor de 1280 x 800 pixeles. El 100 % son los 1280 pixeles, el 80 % son 1280 x (80/100) = 1024 pixeles y el 60 % son 1280 x (60/100) = 768 pixeles. En este caso tendríamos que escribir:

```
/* Curso CSS estilos aprenderaprogramar.com*/
html {background-color: pink; }
body {background-color: yellow; width: 1280px;}
h1 {background-color: blue; width: 1024px;}
h2 {background-color: red; width: 768px;}
```

Supongamos que estamos en un monitor de 1366 x 768 pixeles. En este caso el 100 % son los 1366 pixeles, el 80 % 1092.8 pixeles y el 60 % son 819.60 pixeles. El código a escribir sería:

```
/* Curso CSS estilos aprenderaprogramar.com*/
html {background-color: pink; }
body {background-color: yellow; width: 1366px;}
h1 {background-color: blue; width: 1092.8px;}
h2 {background-color: red; width: 819.6px;}
```

Para otras resoluciones de pantalla tendríamos que hacer los mismos cálculos.

Fíjate que usando tanto por ciento la representación o aspecto será igual en todos los dispositivos o pantallas, mientras que usando pixeles el aspecto depende de la pantalla en que estemos viendo la web. De ahí que con frecuencia se aluda al pixel como una medida absoluta o híbrida.

**Próxima entrega:** CU01026D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

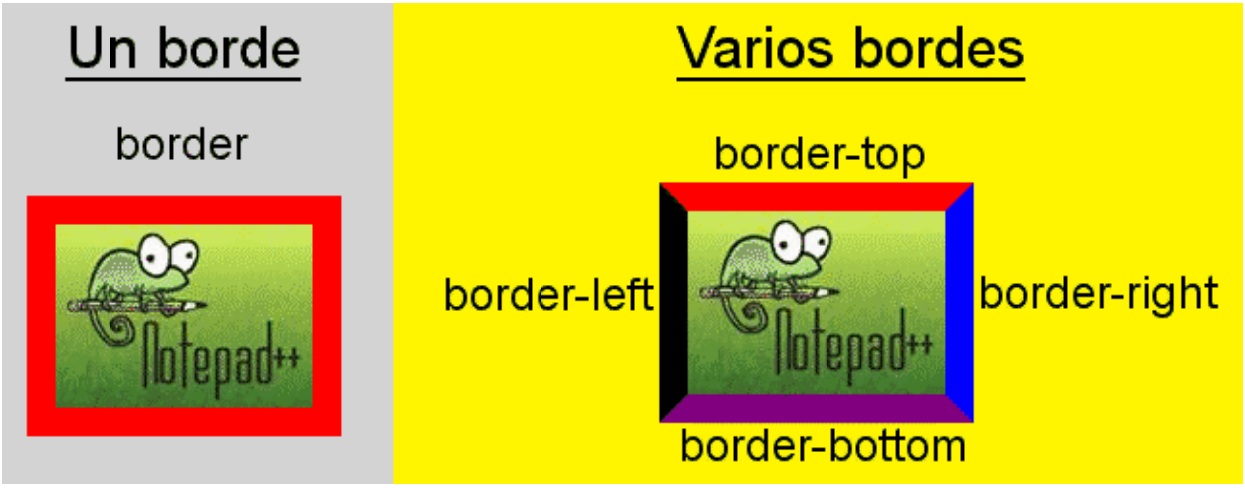
© aprenderaprogramar.com, 2006-2029

**CAJAS Y MODELO DE BORDES**

En apartados anteriores del curso hemos dicho que CSS se basa en el modelo de cajas. Cada división estructural en la página web, cada etiqueta o tag, genera una caja que puede ser un elemento tipo block (a todo lo ancho) o un elemento inline (se posiciona al lado de otros en una línea). Vamos a estudiar cómo aplicar bordes a las cajas y las propiedades básicas relacionadas con los bordes.

**BORDE ÚNICO O BORDES POR PARTES**

Lo primero que debemos tener en cuenta es que una caja podremos verla como un elemento con un borde o bien como un elemento con distintos bordes (hasta un total de cuatro bordes: superior o top, derecho o right, abajo o bottom, izquierdo o left).




Si sólo necesitamos definir un borde pensaremos como en la imagen de la izquierda: todo el elemento lo consideraremos rodeado por un solo borde. Cuando necesitemos definir de forma diferenciada uno o varios de los cuatro bordes posibles, pensaremos siguiendo el modelo mostrado en la imagen de la derecha. Si prestas atención a esta imagen, verás que cuando están establecidos de distintos colores los cuatro bordes, en la zona de unión entre bordes en las esquinas se produce un "choque" entre dos bordes. El navegador divide la esquina común entre los dos colores y esto en cierta manera puede generar un efecto similar a 3D (tridimensional). En el caso de que sólo se estableciera un borde como el borde superior (border-top), éste borde se mostraría como una línea rectangular de cierto grosor, con las esquinas terminando en ángulos rectos en lugar de en bisel.

Un borde se define utilizando propiedades de bordes para elementos CSS, de entre las cuales las fundamentales son:

a) El tipo de borde (border-style) mediante el cual definimos si es una línea normal, un borde de puntos, un borde de pequeñas líneas, etc.
b) El grosor del borde (border-width), mediante el cual definimos el grosor del borde.
c) El color del borde (border-color), medianTe el cual definimos el color del borde.

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS border-style | |
|---|---|
| **Función de la propiedad** | Define el tipo de borde que se aplica a un elemento. |
| **Valor por defecto** | none (no existe borde) |
| **Aplicable a** | Todos los elementos |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | none (no existe borde; no se ocultan otros bordes adyacentes) |
| | hidden (el borde existe pero está oculto y no es visible; oculta bordes contiguos) |
| | dotted (borde a base de puntos redondeados) |
| | dashed (borde a base de trazos o segmentos rectangulares) |
| | solid (borde como una línea normal formando un rectángulo) |
| | double (borde en forma de doble línea, exterior e interior) |
| | groove (efecto 3D con foco de luz arriba a la izquierda. En algunos colores y grosores de línea la visualización no es buena) |
| | ridge (efecto 3D con foco de luz abajo a la derecha. En algunos colores y grosores de línea la visualización no es buena) |
| | inset (efecto 3D de ventana con elemento al fondo. En algunos colores y grosores de línea la visualización no es buena) |
| | outset (efecto 3D de elevación. En algunos colores y grosores de línea la visualización no es buena) |
| **Ejemplos aprenderaprogramar.com** | img {border-style: solid;}<br>.imgVentana {border-style: inset; border-width:10px;} |

La propiedad border-style se puede aplicar a todo el borde del elemento, o bien a cualquiera de los cuatro bordes en que se puede subdividir un borde. De esta manera se generan cuatro propiedades cuyo funcionamiento es análogo al de border-style, pero que permiten dar un tratamiento diferenciado a cada uno de los bordes: **border-top-style**, **border-right-style**, **border-bottom-style** y **border-left-style**. La sintaxis a utilizar es esta:

```
selectorDeElemento {border-top-style: valor1; border-right-style: valor2;
border-bottom-style: valor3; border-left-style: valor4; }
```

No hay por qué especificar los cuatro bordes. Podemos especificar sólo uno de ellos, o dos, o tres, hasta el máximo de 4, en el orden que queramos. Por ejemplo podríamos usar img {border-top-style: solid; } con lo cual únicamente aparecería para los elementos img un borde: el borde superior. Hemos dicho que hay 3 propiedades básicas para definir un borde, relativas al tipo (style), grosor (width) y color (color). Si dejamos sin definir el border-style el resultado será que no se visualizará borde, ni siquiera

© aprenderaprogramar.com, 2006-2029

aunque esté definido un grosor y color, ya que por defecto el border-style tiene valor “none”, lo que significa que no se visualizará borde alguno.

¿Qué ocurriría si en el código CSS incluimos una declaración como img {border-style: solid; border-top-style: dashed; } ? Podemos ver el resultado en la siguiente imagen:

En primer lugar, se establece el borde para todos los elementos imagen como un borde sólido (línea normal). ¿Por qué se ve un borde si no hemos establecido un grosor (width) ni un color (color)? Tenemos que tratar de razonar sobre el por qué de las cosas que observamos. Ser buenos programadores o diseñadores CSS implica razonar y entender, no sólo conocer de memoria propiedades o sintaxis del lenguaje. Si sin haber establecido un grosor se ve una línea de determinado grosor es porque se está aplicando un valor de defecto o valor inicial para el grosor. Igualmente si el borde se ve de color azul sin haber establecido un color es porque se está aplicando un valor de defecto o valor inicial para el color. Estos valores de defecto serán los definidos por el estándar CSS o por el navegador que estemos utilizando. Si no deseamos que se apliquen estos valores por defecto debemos especificar unos valores para dichas propiedades.

Otra cuestión que podemos razonar con esta imagen es: ¿Por qué aparece el borde superior (top) en forma de segmentos y el resto de bordes (right, bottom, left) en forma de línea normal? Si leemos el código empleado veremos que en primer lugar se define que el borde completo de los elementos img será solid (línea normal); a continuación se define que el borde superior de los elementos img será en forma de trazos (segmentos). El navegador procede a sustituir el borde superior por el borde a trazos, mientras que mantiene el resto de bordes (right, bottom y left) como borde normal. Con este código hemos efectuado una declaración inicial que es contradicha por una declaración posterior. Quizás no tengamos problemas y la visualización sea correcta en todos los navegadores. Pero como “estilo de trabajo” recomendaríamos evitar siempre que sea posible las contradicciones porque a la larga pueden generar problemas. En este caso en vez de {border-style: solid; border-top-style: dashed; } podríamos usar la declaración {border-top-style: dashed; border-right-style: solid; border-bottom-style: solid; border-left-style: solid; }. Veremos más adelante que esta expresión puede escribirse también de forma abreviada o compacta. De esta manera hemos definido con claridad lo que queremos sin necesidad de incurrir en contradecir o sobreescribir parcialmente una propiedad. Si nuestras definiciones de código CSS son precisas, la probabilidad de que surjan problemas será menor. Por tanto y a modo de recomendación, no te conformes con que la visualización sea correcta: intenta que el código esté definido lo más correctamente posible.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Crea un documento HTML con 10 divisiones, cada una separada de la anterior por dos elementos <br/>. En cada división introduce un texto (p.ej. div 1, div 2, div 3..) y aplícale un estilo de borde diferente utilizando la propiedad border-style.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01027D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## SHORTAND NOTATION O NOTACIÓN ABREVIADA CSS

Con bastante frecuencia tendremos propiedades CSS donde es posible definir un valor global, como un borde único, o bien hasta cuatro valores específicos para el lado superior (top), derecho (right), inferior (bottom) o izquierdo (left). Para facilitar la escritura de código CSS permite el uso de diferentes formas abreviadas.

La forma abreviada (en inglés "shortand notation") básica para diferentes propiedades que usan los cuatro lados de las cajas CSS se basa en usar esta sintaxis: selectorElemento {nombrePropiedad: valorTop valorRight valorBottom valorLeft;}

Por ejemplo la forma abreviada img {border-style: solid dashed dotted double; } es equivalente a escribir de forma extendida img {border-top-style: solid; border-right-style: dashed; border-bottom-style: dotted; border-left-style: double; }. Tener en cuenta que en la forma abreviada los valores van separados simplemente con un espacio. En ambos casos el resultado sería similar al que vemos en la siguiente imagen.

Como se puede comprobar, usar la forma abreviada nos permite ahorrar en extensión de código que es necesario teclear y por tanto nos permite ahorrar tiempo cuando estemos escribiendo código CSS. Además, cuando los archivos CSS son muy extensos, usar la forma abreviada puede redundar en que nuestros archivos sean más ligeros y se carguen más rápidamente.

La notación abreviada o shortand notation se usa para muchas propiedades para establecer márgenes, rellenos, etc. por lo que nos la encontraremos repetidamente cuando trabajemos con CSS. Hay una cosa que debemos memorizar: el orden en que se escriben los valores y su correspondencia con los cuatro lados de la caja. Para ello podemos usar esta regla mnemotécnica: el día comienza a las 12:00 y el primer lado es TOP, las agujas del reloj avanzan hacia la derecha y el segundo lado el RIGHT. Las agujas siguen avanzando y el tercer lado es BOTTOM. El reloj sigue avanzando y el cuarto lado es LEFT. De ahí que cuando vemos una expresión de tipo img {border-style: solid dashed dotted double; } debamos pensar en solid --> TOP, dashed --> RIGHT, dotted --> BOTTOM, double --> LEFT.

Pero se pueden usar formas abreviadas de otras maneras, usando en vez de cuatro valores menos valores. En la siguiente tabla vemos el significado de las expresiones según el número de valores que usemos:

© aprenderaprogramar.com, 2006-2029

| Valores empleados | Ejemplo | Significado |
|---|---|---|
| 1 | img {border-style: dotted; } | Los cuatro lados de la caja se muestran con el valor indicado. |
| 2 | img {border-style: dotted solid; } | El primer valor se aplica a TOP y BOTTOM y el segundo valor se asigna a RIGHT y LEFT |
| 3 | img {border-style: dotted solid dashed; } | El primer valor se aplica a TOP, el segundo a RIGHT y LEFT y el tercero a BOTTOM |
| 4 | img {border-style: dotted solid dashed double;} | El primer valor se aplica a TOP, el segundo a RIGHT, el tercero a BOTTOM y el cuarto a LEFT |

Hay más formas de notaciones abreviadas en CSS como veremos más adelante. Algunos navegadores, en especial los más antiguos, podrían no reconocer las expresiones abreviadas, aunque prácticamente todos los navegadores actuales sí las reconocen.

**PROPIEDAD CSS BORDER-WIDTH**

Vistos ya los diferentes tipos de bordes que podemos aplicar con CSS, un aspecto importante será definir su grosor. Esto se hace con la propiedad border-width.

| PROPIEDAD CSS border-width | |
|---|---|
| **Función de la propiedad** | Define el grosor de borde que se aplica a un elemento. |
| **Valor por defecto** | medium (grosor por defecto medio) |
| **Aplicable a** | Todos los elementos |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | Una unidad de medida absoluta o relativa admitida y aplicable (en general cualquier unidad excepto porcentaje será válida) |
| | thin (fino) |
| | medium (medio) |
| | thick (grueso) |
| | inherit (heredado del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | img {border-style: solid; border-width: 15px;}<br>.imgVentana {border-style: inset; border-width:1.25em;} |

© aprenderaprogramar.com, 2006-2029

El grosor exacto en el caso de usar thin, medium y thick depende del navegador que estemos utilizando, de ahí que normalmente los diseñadores prefieran especificar el tamaño de los bordes con valores como pixeles o em, que les permiten un mayor control.

Recordar que para que el borde sea visible debemos establecer border-style, ya que sin esta propiedad establecida la aplicación de border-width no generará ningún resultado.

La propiedad border-width puede especificarse usando la shortand notation como hemos explicado anteriormente. Por ejemplo {border-width:0.25em 0.85em 0.45em;} establece el borde superior en 0.25em, los bordes laterales izquierdo y derecho en 0.85em y el borde inferior en 0.45em.

La propiedad border-width se puede aplicar a todo el borde del elemento, o bien a cualquiera de los cuatro bordes en que se puede subdividir un borde. De esta manera se generan cuatro propiedades cuyo funcionamiento es análogo al de border-widht, pero que permiten dar un tratamiento diferenciado a cada uno de los bordes: **border-top-width**, **border-right-width**, **border-bottom-width** y **border-left-width**. La sintaxis a utilizar y la forma de funcionamiento es la misma que hemos explicado anteriormente para border-style.

**PROPIEDAD CSS BORDER-COLOR**

La propiedad border-color nos permite aplicar colores de borde.

| PROPIEDAD CSS border-color | |
|---|---|
| **Función de la propiedad** | Define el color de borde que se aplica a un elemento. |
| **Valor por defecto** | Será el valor que tenga la propiedad color para el elemento, establecida en el código o la que tenga el elemento por defecto. |
| **Aplicable a** | Todos los elementos |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | Un color establecido de cualquiera de las maneras válidas en CSS |
| | transparent (el borde existe pero es transparente) |
| | inherit (heredado del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | img {border-style: solid; border-width: 15px; border-color: cyan}<br>.imgVentana {border-style: solid dotted; border-width:0.35em; border-color: red blue;} |

El valor transparent nos permite generar la apariencia de que existe un borde pero este no será visible por ser transparente.

Recordar que para que el borde sea visible debemos establecer border-style, ya que sin esta propiedad establecida la aplicación de border-color no generará ningún resultado.

© aprenderaprogramar.com, 2006-2029

La propiedad border-color puede especificarse usando la shortand notation como hemos explicado anteriormente. Por ejemplo {border-color:#ff0000 rgb(0,255,0) blue rgba(0,255,0,0.33);} establece los bordes de distintos colores siguiendo el orden top – right – bottom – left.

La propiedad border-color se puede aplicar a todo el borde del elemento, o bien a cualquiera de los cuatro bordes en que se puede subdividir un borde. De esta manera se generan cuatro propiedades cuyo funcionamiento es análogo al de border-widht, pero que permiten dar un tratamiento diferenciado a cada uno de los bordes: **border-top-color**, **border-right-color**, **border-bottom-color** y **border-left-color**. La sintaxis a utilizar y la forma de funcionamiento es la misma que hemos explicado anteriormente para border-style.

**PROPIEDAD CSS BORDER**

Hasta ahora hemos visto que para establecer un borde habíamos de definir al menos la propiedad border-style, y si queríamos especificar el grosor y el color adicionalmente teníamos que definir border-width y border-color.

Hay una propiedad que permite expresar las tres propiedades básicas de un borde de forma abreviada: la propiedad border. La sintaxis a emplear es:

border { valorBorderWidth **valorBorderStyle** valorBorderColor; }

| PROPIEDAD CSS border | |
|---|---|
| **Función de la propiedad** | Permite definir grosor, estilo y color de borde. |
| **Valor por defecto** | Los de las distintas sub-propiedades de que consta. |
| **Aplicable a** | Todos los elementos |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | Un valor (establecerá border-style) |
| | Dos valores (establecerá border-style y grosor o color) |
| | Tres valores (establecerá border-width, border-style y border-color) |
| | inherit (heredado del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | img {border: 15px solid #FFAA33; }<br>.imgVentana { border: dotted;} |

Nota: en general los navegadores permiten que las subpropiedades de border se expresen en cualquier orden, es decir, dará igual escribir img {border: 15px solid #FFAA33; } que img {border: solid #FFAA33 15px;}. El navegador interpretará en cada caso qué es el estilo, qué es el grosor y qué es el color.

La propiedad admite hasta tres valores (anchura, estilo y color) de los cuales es obligatorio el estilo y opcionales los otros dos.

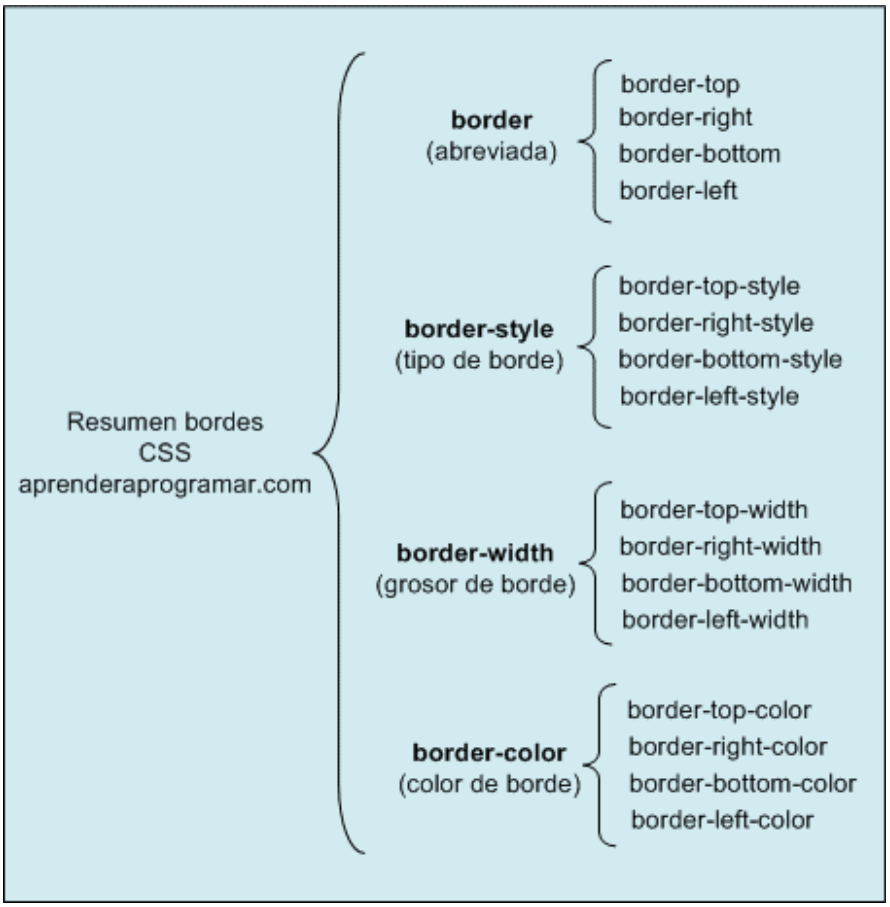
© aprenderaprogramar.com, 2006-2029

| Valores empleados | Ejemplo | Significado |
|---|---|---|
| 1 | img {border: dotted; } | Los cuatro lados de la caja se muestran con el estilo indicado. Grosor y color serán los de defecto. |
| 2 | img {border: 20px solid; }<br>img {border: solid red; } | Se aplica el estilo indicado y el grosor ó color indicado. Lo no especificado será por defecto. |
| 3 | img {border: 15px solid #FFAA33; } | Se aplica el grosor, estilo y color indicados. |

Este tipo de propiedades como border que permiten abreviar diferentes propiedades se denominan "shortand properties" o propiedades abreviadas.

La propiedad border se puede aplicar a cualquiera de los cuatro bordes en que se puede subdividir un borde. De esta manera se generan cuatro propiedades cuyo funcionamiento es análogo al de border, pero que permiten dar un tratamiento diferenciado a cada uno de los bordes: **border-top**, **border-right**, **border-bottom** y **border-left**. La sintaxis a utilizar y la forma de funcionamiento para estas propiedades son las que ya conocemos.

El siguiente esquema resume las propiedades de borde CSS:




© aprenderaprogramar.com, 2006-2029

**EJERCICIO 1**

Crea un documento HTML con 2 divisiones, cada una separada de la anterior por dos elementos <br/>. En cada división introduce un texto (p.ej. div 1, div 2) y aplícale los siguientes estilos de borde escribiendo de forma individual cada una de las siguientes propiedades CSS:

a) Para el div 1: la parte superior con borde de puntos redondeados, grosor 10 píxeles y color verde. La parte derecha con borde de trazos o segmentos rectangulares, grosor 20 píxeles y color azul. La parte inferior con borde de línea doble, grosor 10 píxeles y color #A52A2A. La parte izquierda con borde con efecto groove, grosor 30 píxeles y color #2F4F4F.

b) Para el div 2: la parte superior con borde con efecto inset, grosor 30 píxeles y color #B22222. La parte derecha con borde sólido, grosor 22 píxeles y color #DAA520. La parte inferior con borde de línea doble, grosor 25 píxeles y color #4B0082. La parte izquierda con borde de puntos redondeados, grosor 17 píxeles y color #808000.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**EJERCICIO 2**

Crea un documento HTML con el mismo aspecto y propiedades CSS que las descritas para el ejercicio 1, pero en este caso usando la notación shortand de bordes CSS y la propiedad border abreviada. Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01028D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## MARGEN Y RELLENO

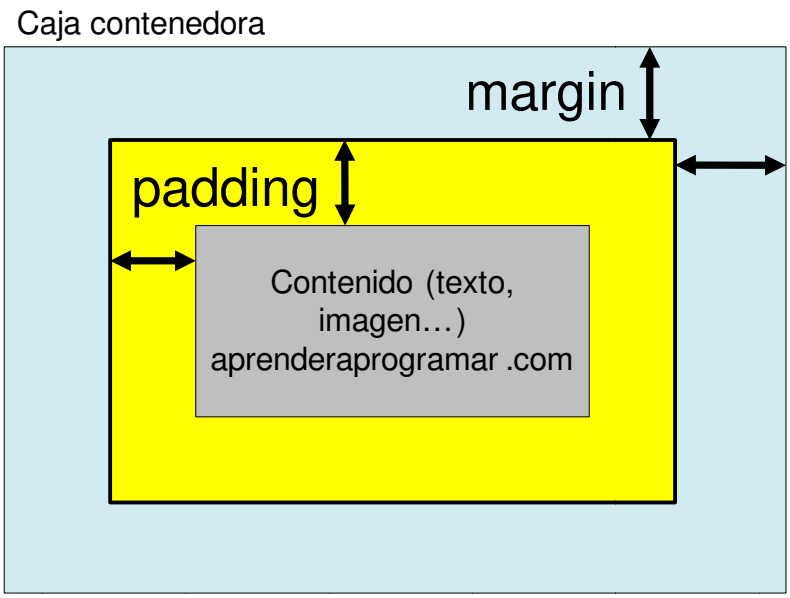
Para avanzar en la comprensión del funcionamiento de CSS debemos conocer, saber utilizar y diferenciar dos propiedades importantes: margin y padding. Al igual que con los bordes, tendremos distintas propiedades derivadas de las principales y también tendremos shortand notations o notaciones abreviadas para poder expresarlas.

## MARGIN Y PADDING

Empezaremos realizando una definición de conceptos:

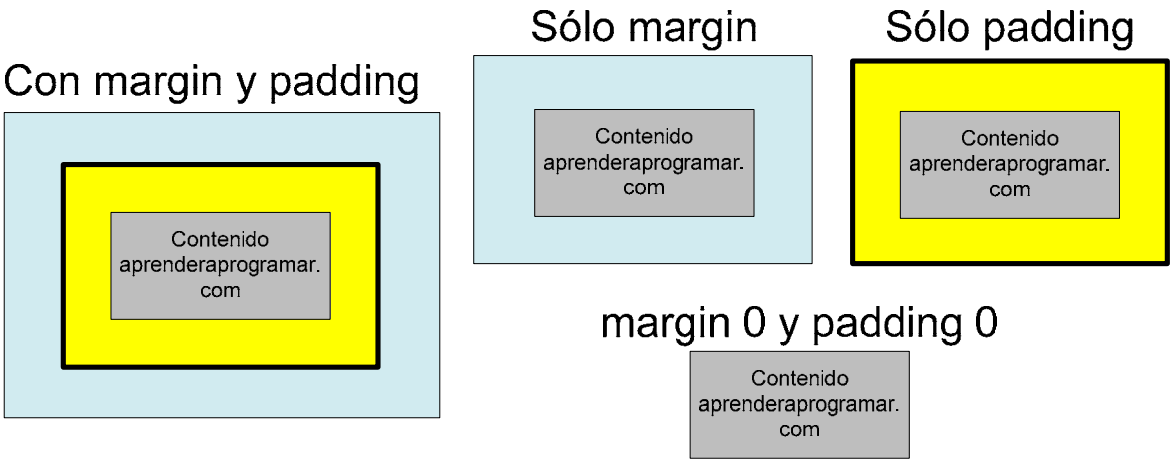
a) Contenido de un elemento HTML es aquello que se pretende mostrar al visualizar ese elemento: puede ser un texto, una imagen o quizás otros elementos como un objeto reproductor de sonido ó un mapa. El contenido en CSS tiene forma de caja rectangular.
b) Borde: es el perímetro que engloba el contenido y el posible relleno (en inglés padding) entre el borde y el contenido.
c) Relleno (padding): es el espacio transparente que se puede dejar, opcionalmente, entre el contenido y el borde del elemento. El relleno no tiene color (pero permite ver el color de fondo).
d) Margen: es el espacio transparente que se puede dejar, opcionalmente, entre el borde del elemento y el borde de otras cajas adyacentes. El margen no tiene color (pero permite ver el color de fondo).

Viendo esquemas gráficos y código se comprenderán con más facilidad estos conceptos:

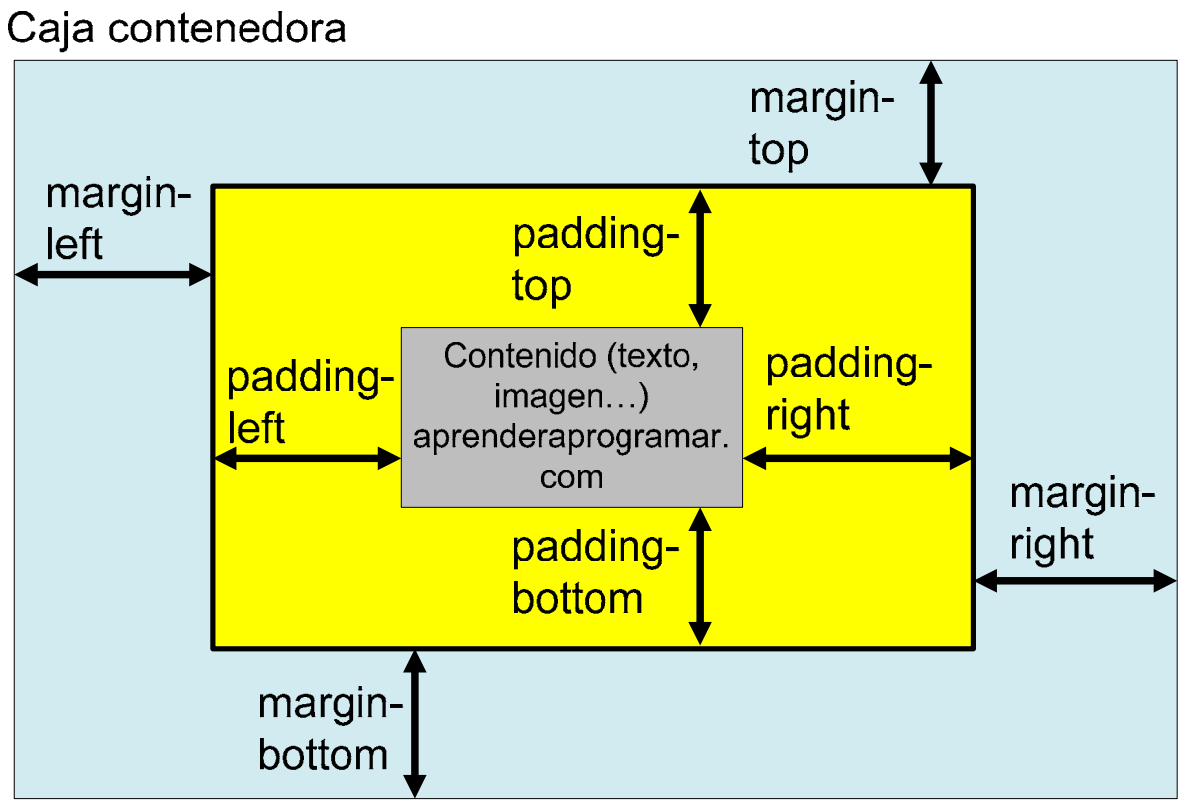



© aprenderaprogramar.com, 2006-2029

Un elemento HTML puede tener margin y padding, sólo tener margin, sólo tener padding, o no tener ni margin ni padding. En caso de no tener margin ni padding el elemento aparecerá "ajustado" exactamente a su caja contenedora.




Al igual que ocurría con las propiedades de borde, las propiedades de margen y relleno se pueden subdividir para aplicarlas a cada uno de los lados de la caja CSS por separado.



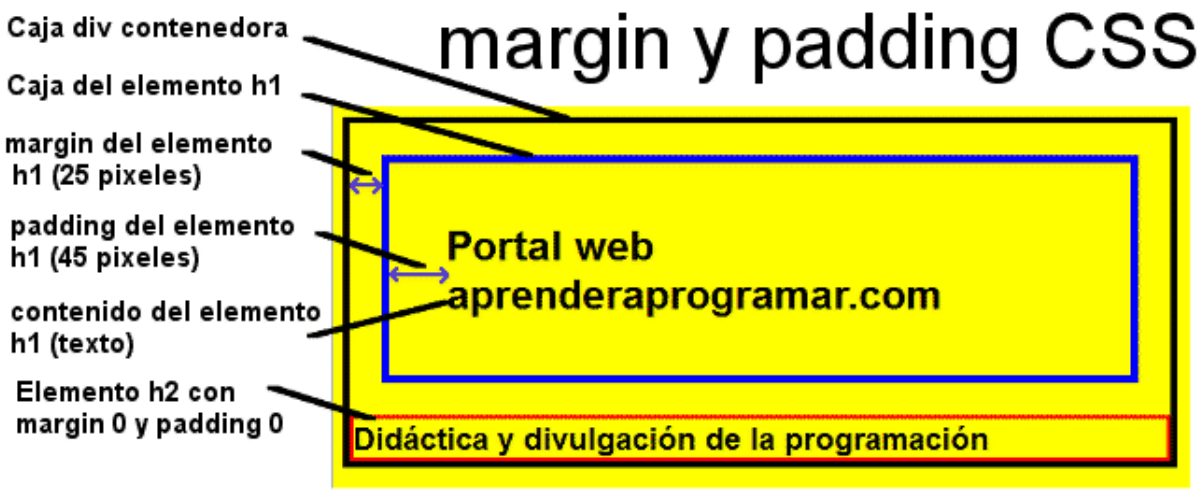

© aprenderaprogramar.com, 2006-2029

**EJEMPLO EN CÓDIGO**

Para ver la aplicación práctica de estos conceptos vamos a utilizar el código HTML de base que venimos empleando a lo largo del curso.

Define los siguientes estilos css y visualiza el resultado en tu navegador:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {background-color: yellow; width: 60%;}
h1{margin: 25px; padding: 45px; border: solid 6px blue; }
h2{border-style: solid; border-color:red; margin: 0; padding:0;}
div {border: solid thick black;}
div div {border: solid medium purple;}
div div div {border: dashed medium grey;}
img {border: solid #FF00FF 2px;}
```




Vamos a analizar línea a línea el código CSS y los resultados obtenidos. La primera línea es un comentario. En la segunda línea establecemos que las fuentes de texto a emplear en todos los elementos del documento sean tipo arial. En la tercera línea establecemos que el elemento body tenga color de fondo amarillo y ocupe el 60% del ancho de pantalla disponible.

En la cuarta línea definimos que el margen de los elementos h1 será de 25 píxeles. Al no especificar top, right, bottom o left, este margen se aplicará en las cuatro direcciones (mismo comportamiento que el que hemos estudiado previamente con los bordes). Igualmente definimos que el relleno para los elementos h1 será de 45 px en todas las direcciones y que el borde se muestre de color azul.

En la quinta línea definimos que los elementos h2 muestren un borde de color rojo y que no llevarán margen ni relleno.

En las siguientes líneas establecemos bordes para que se muestren las cajas contenedoras de los elementos HTML de distintos colores. Estas líneas no nos interesa comentarlas ahora.

En los resultados de visualización comprobamos lo siguiente: si los elementos h2 no tienen margen, se mostrarán pegados a las cajas adyacentes. Así vemos que ocurre con la caja del div contenedor, pero no

© aprenderaprogramar.com, 2006-2029

con la caja del h1. ¿Es esto correcto? Sí, porque el h1 tiene establecido un margen y por tanto la caja del elemento h2 no puede pegarse a la caja del elemento h1 porque hay un margen que lo impide.

Prueba a hacer el siguiente cambio en el código: h2{border-style: solid; border-color:red; margin: 25px; padding:0;}

Al visualizar el resultado comprobarás que los márgenes verticales (top y bottom) contiguos no son aditivos, es decir, si la caja h1 tiene de margen 25px y la caja de h2 tiene de margen 25px, el espacio vertical que quedará entre ambas cajas es 25px y no 50px. ¿Por qué? Porque la referencia para establecer un margen vertical es el borde de las cajas y no el límite del margen de una caja adyacente. Si la caja h1 tuviera de margen vertical 25px y la caja h2 tuviera de margen vertical 10px el espacio entre ambas cajas sería de 25px, el mayor entre los dos valores.

El comportamiento de los márgenes horizontales es diferente: en este caso, sí es aditivo. Supongamos que tenemos dos imágenes una al lado de otra y establemos img {margin:25px;}. En este caso el espacio que habrá entre ambas imágenes sí es de 50 px, es decir, los márgenes horizontales sí se suman, mientras que los verticales no se suman. Este comportamiento peculiar deberemos tenerlo en cuenta cuando trabajemos con CSS. Y como CSS no es matemáticas, habrá que realizar pruebas y comprobaciones para verificar que se cumple aquello que esperamos.

La idea clave con que debemos quedarnos es que los márgenes verticales tienen un comportamiento distinto al de los márgenes horizontales.

Continuamos con el análisis de los resultados. Si la caja h2 no tiene padding, la línea de borde quedará ceñida al contenido (en este caso al texto). Así vemos que ocurre por la izquierda, por arriba y por abajo. Sin embargo por la derecha el borde no está ajustado al texto. ¿Por qué? Porque los elementos h1 y h2 son elementos de tipo block, y por tanto tienden a ocupar todo el ancho de pantalla. Más adelante veremos que podemos cambiar el comportamiento para que se comporten como block ó como inline según nuestro criterio.

Finalmente queremos llamar la atención sobre el hecho de que se muestre un pequeño espacio entre el borde del div y el borde del elemento body, de ahí que se vea una pequeña franja amarilla por fuera del contenedor div. Esta situación se debe a los márgenes o rellenos que por defecto aplica el navegador. Podemos eliminar estos márgenes o rellenos por defecto incluyendo en el selector universal este código: *{margin:0; padding:0;} Prueba a aplicarlo y compruébalo. A prácticas de este tipo, consistentes en escribir código para anular el comportamiento por defecto que introducen los navegadores, se le suele denominar “reseteo”. Es una práctica que siguen muchos diseñadores y programadores, pero no todos. Algunos programadores están en contra de utilizar prácticas de reseteo.

El código de reseteo puede ser bastante más amplio que el que hemos indicado y hablaremos de ello más adelante. Nosotros iremos introduciendo algunas prácticas de reseteo cuando lo consideremos necesario para el desarrollo del curso.

**EJERCICIO**

Para cada una de las siguientes afirmaciones indica si la afirmación es verdadera o falsa y justifica brevemente tu respuesta:

© aprenderaprogramar.com, 2006-2029

a) CSS muestra los contenidos con tres formas básicas: rectangular, circular y elipsoidal.

b) Al crear un documento HTML, pueden aparecer márgenes y rellenos que no hayan sido explícitamente introducidos por nosotros como programadores, sino que hayan sido introducidos por el navegador que estemos empleando.

c) El reseteo CSS consiste en la recarga de la página para borrar la caché del navegador, de modo que los estilos se recarguen completamente.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01029D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## MARGIN Y PADDING

Ya debemos tener claro el concepto de margen y de relleno en CSS. Ambos son elementos muy usados en los diseños y que presentan ciertas similitudes y ciertas diferencias. En algunos casos el efecto visual al que llevan es muy parecido. Sin embargo, conviene tener clara la diferencia entre ambos conceptos y saber utilizarlos adecuadamente.

## PROPIEDAD CSS PADDING

La propiedad CSS padding nos permite fijar el relleno de un elemento. La sintaxis a emplear es del tipo:

```
selectorElemento { padding valor1Uds valor2Uds valor3Uds valor4Uds; }
```

| PROPIEDAD CSS padding | |
|---|---|
| **Función de la propiedad** | Permite definir un relleno entre el borde de caja y el contenido. |
| **Valor por defecto** | 0 (no existe relleno). |
| **Aplicable a** | Elementos tipo block y elementos insertados en una posición que son reemplazados por un objeto (entre ellos img, input, textarea, select, object). |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | Un valor absoluto o relativo (establece relleno en los cuatro sentidos) |
| | Dos valores (el primero de ellos aplica a top y bottom y el segundo a right y left) |
| | Tres valores (el primero para top, el segundo right / left y el tercero bottom). |
| | Cuatro valores (se aplican a top, right, bottom y left en este orden) |
| | inherit (heredado del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | #menu1 {padding: 55px 0.5em 177px 2px;}<br>.imgVentana { padding: 15px;}<br>h1 {padding: 10% 5%;}<br>.container1 {padding: 0 auto;} |

La propiedad padding se puede aplicar en todos los lados del elemento, o bien a cualquiera de los cuatro lados que conforman una caja CSS. Existen cuatro propiedades cuyo funcionamiento es análogo al de padding, pero que permiten dar un tratamiento diferenciado a cada uno de los lados de la caja CSS: **padding-top**, **padding-right**, **padding-bottom** y **padding-left**. La sintaxis a utilizar y la forma de funcionamiento es la misma que hemos explicado.

© aprenderaprogramar.com, 2006-2029

El valor de padding se puede establecer en %, realizándose el cálculo **respecto al ancho del elemento contenedor** (un valor 100% sería un relleno igual al ancho del elemento contenedor).

La propiedad padding no admite valores negativos.

Recordar que tanto margin como padding generan un espacio transparente. El color o imagen que se vean en dicho espacio serán el color de fondo (background-color) o la imagen de fondo (background-image) que estén definidos para el elemento afectado.

Tanto con margin como con padding pueden haber determinadas situaciones en que los navegadores, sobre todo los más antiguos, no respondan como sería de esperar, aunque la mayor parte de los navegadores actuales responden adecuadamente.

**PROPIEDAD CSS MARGIN**

La propiedad CSS margin nos permite fijar el margen de un elemento. La sintaxis a emplear es del tipo:

```
selectorElemento { margin valor1Uds valor2Uds valor3Uds valor4Uds; }
```

| PROPIEDAD CSS margin | |
|---|---|
| **Función de la propiedad** | Permite definir el margen entre el borde de la caja del elemento y el borde de las cajas adyacentes. |
| **Valor por defecto** | 0 (no existe margen). |
| **Aplicable a** | Elementos tipo block y elementos insertados en una posición que son reemplazados por un objeto (entre ellos img, input, textarea, select, object). |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | auto (el navegador aplicará el margen por defecto) |
| | Un valor absoluto o relativo (establece márgenes en los cuatro lados de la caja) |
| | Dos valores (el primero de ellos aplica a top y bottom y el segundo a right y left) |
| | Tres valores (el primero para top, el segundo right / left y el tercero bottom). |
| | Cuatro valores (se aplican a top, right, bottom y left en este orden) |
| | inherit (heredado del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | #menu1 {margin: 55px 0.5em 177px 2px;}<br>.imgVentana { margin: 15px;}<br>h1{margin: 10% 5%;}<br>.container1 {margin: 0 auto;} |

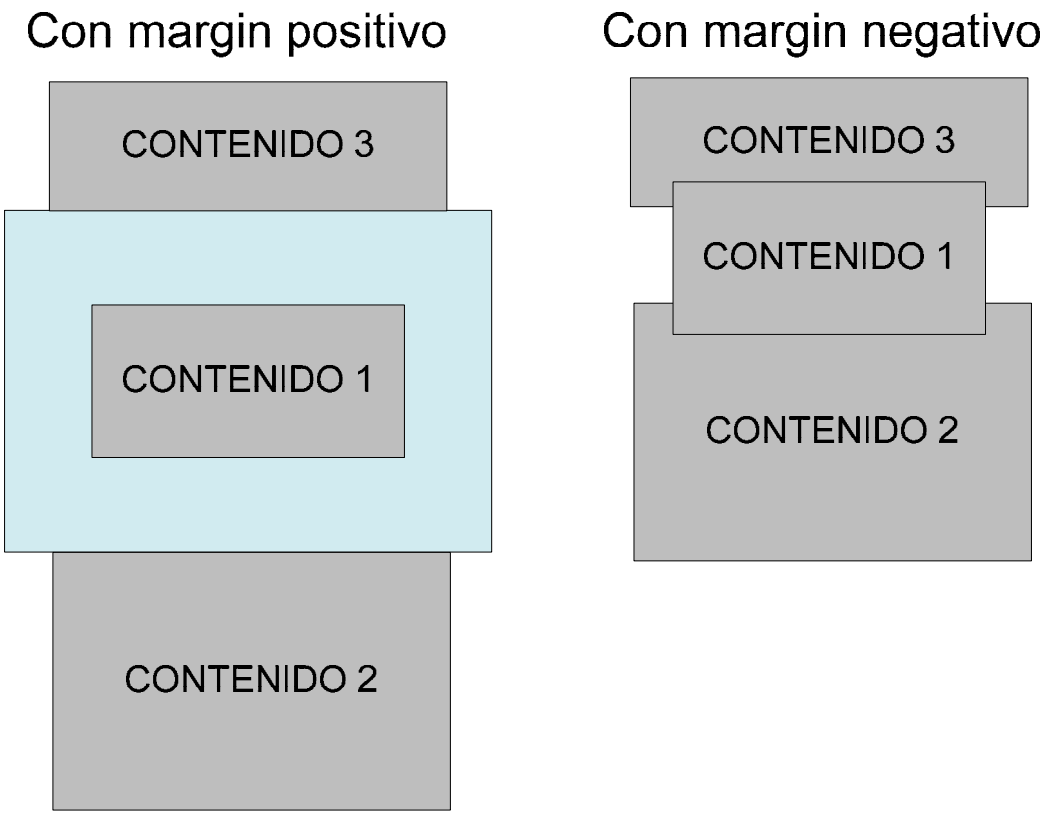
© aprenderaprogramar.com, 2006-2029

La propiedad margin se puede aplicar en todos los lados del elemento, o bien a cualquiera de los cuatro lados que conforman una caja CSS. Existen cuatro propiedades cuyo funcionamiento es análogo al de margin, pero que permiten dar un tratamiento diferenciado a cada uno de los lados de la caja CSS: **margin-top**, **margin-right**, **margin-bottom** y **margin-left**. La sintaxis a utilizar y la forma de funcionamiento es la misma que hemos explicado anteriormente.

El valor de margin se puede establecer en %, realizándose el cálculo **respecto al ancho del elemento contenedor** (un valor 100% sería un margen igual al ancho del elemento contenedor).

A diferencia de otras propiedades, se admite que un valor de margin pueda ser negativo (aunque esto no es usual y no debería usarse excepto cuando razonadamente es necesario y aconsejable. Un margen negativo puede dar lugar, en algunos casos, a visualizaciones incorrectas o extrañas). Para entender el efecto de valores de margen negativos podemos pensar en los márgenes como aviones que tienen un espacio básico definido por la anchura de sus alas y fuselaje, que definen un espacio rectangular que sería el margin 0 (no hay margin y el avión "empuja" a otros aviones colindantes). Ese espacio se puede aumentar hacia fuera estableciendo un margin positivo que sería como "establecer un perímetro de seguridad para el avión", lo que impide que otros aviones entren en el espacio reservado a nuestro avión. Si el margen se establece en un valor negativo, el avión no sólo no tiene un espacio extra hacia fuera, sino que tampoco tiene su espacio básico y se coloca invadiendo el espacio delimitado por otro avión contiguo. Margin positivo sería "empujar hacia fuera" y margin negativo sería "solaparse" con elementos adyacentes.

En la siguiente imagen vemos cómo afecta establecer un margen positivo o negativo a un elemento "Contenido 1" que tiene otros dos elementos vecinos.




© aprenderaprogramar.com, 2006-2029

La invasión del espacio colindante puede ser en todas direcciones o establecerse específicamente para alguna de las cuatro direcciones posibles (top, right, bottom o left) usando margin-top, margin-right, margin-bottom o margin-left. Un aspecto a tener en cuenta es que cuando existe superposición de un elemento con otros habrá que comprobar o establecer qué elemento debe quedar encima y qué elemento debe quedar debajo. En la imagen anterior hemos dibujado el elemento "Contenido 1" situándose encima de los otros elementos, pero en la práctica esto dependerá de diversos factores y no siempre será así. Existen propiedades CSS con las que podremos controlar el orden de apilamiento de elementos cuando existe superposición. Lo veremos más adelante.

Otro aspecto que merece ser comentado es el resultado que se obtendrá al establecer un valor de margin "auto". Elegir auto equivale a indicar que el margin debe ser establecido automáticamente por el navegador. En general, si un elemento es de tipo block y no tiene establecido una anchura específica, cuando establecemos el margin con valor "auto" el margen que aplicará el navegador cerá cero (sin margen). En cambio, si el elemento es de tipo block pero tiene un valor de ancho (width) definido, al establecer el margin auto el navegador establecerá un margen izquierdo y derecho iguales, de modo que el contenido estará equidistante respecto al borde del contenedor. El efecto que se aprecia, en este caso, es que el elemento aparece centrado.

Ejemplo:

a) Código que da lugar a un margen cero: h1{ border: solid; margin:auto; }, el elemento h1 se ve a todo lo ancho y sin márgenes.

b) Código que da lugar a el centrado horizontal del elemento: h1{border: solid 6px blue; width:65%; margin:auto;}. El elemento h1 se ve sin márgenes superior (top) ni inferior (bottom), pero se ve centrado (con márgenes right y left de igual valor).

Prueba a visualizar la página de pruebas en tu navegador con este código. En el primer caso h1 es un elemento block sin ancho definido, por lo que tiende a ocupar todo el ancho disponible. En este caso margin: auto; da lugar a que se aplique margen cero. En el segundo caso h1 es un elemento block con ancho el 65% del elemento contenedor, por tanto tiene un ancho definido. En este caso margin: auto; da lugar a que se apliquen márgenes iguales por ambos lados del elemento y a que visualmente el elemento aparezca centrado.

**EJERCICIO**

Crea un documento HTML con 2 elementos div de anchura 250 píxeles, con un margen de 20px en todas direcciones y uno junto al otro (en horizontal). En cada div introduce un texto (p.ej. div 1, div 2) y aplícale los siguientes estilos de borde y relleno a ambos elementos. Color de fondo #FFB6C1. La parte superior con borde de puntos redondeados, grosor 15 píxeles, color #DC143C y relleno de 30 píxeles. La parte derecha con borde de trazos o segmentos rectangulares, grosor 10 píxeles, color verde y relleno de 45 píxeles. La parte inferior con borde de línea doble, grosor 10 píxeles, color #FF00FF y relleno 0 píxeles. La parte izquierda con borde con efecto ridge, grosor 40 píxeles, color #2F4F4F y relleno 60 píxeles.

Responde a las siguientes preguntas:

© aprenderaprogramar.com, 2006-2029

a) ¿Cuál es el ancho total ocupado por cada div (incluyendo sus bordes y rellenos)?

b) ¿Cuál es el alto total ocupado por cada div (incluyendo sus bordes y rellenos)?

c) ¿Cuál es el ancho total desde el límite izquierdo del borde del div más a la izquierda hasta el límite derecho del borde del div más a la derecha (teniendo en cuenta márgenes, bordes y rellenos)?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01030D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

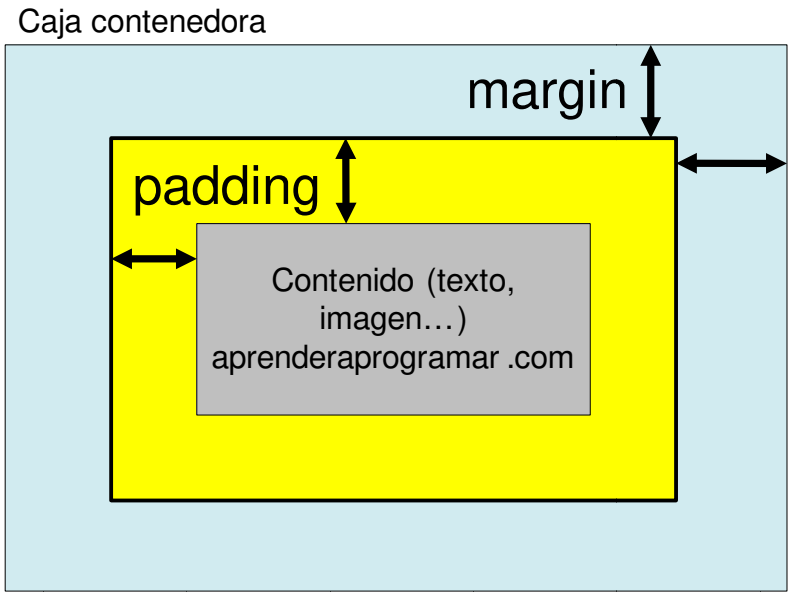
© aprenderaprogramar.com, 2006-2029

## FONDO EN CSS

CSS tiene diferente propiedades relacionadas con el fondo de las cajas del box-model o modelo de cajas. Ya conocemos la propiedad background-color. Vamos a ampliar el conocimiento sobre los fondos estudiando otras propiedades relacionadas: background-image, background-repeat, background-attachment, background-position y la propiedad shortand background.

## CONCEPTO DE FONDO DE UN ELEMENTO

El fondo de un elemento CSS comprende el espacio desde los bordes hacia el interior de la caja del box model, incluyendo el propio borde. Por tanto el fondo comprende el espacio de borde, el padding y el espacio del contenido. El margin no se ve afectados por el fondo del elemento. Si el borde es opaco, el fondo quedará oculto por el borde. En cambio si tiene transparencia o partes no opacas, dejará ver el fondo.




## PROPIEDAD BACKGROUND-COLOR

Es la propiedad que permite establecer el color de fondo, de la cual ya hemos hablado, por lo que no vamos a extendernos en ella. Recordar que el background-color por defecto para todos los elementos es transparent.

## PROPIEDAD BACKGROUND-IMAGE

Esta propiedad nos permite establecer una imagen de fondo para un elemento. Una imagen se coloca encima del background-color. Si la imagen es opaca, ocultará el background-color. Si la imagen tiene partes transparentes, se verá el color de fondo rellenando esas partes transparentes.

© aprenderaprogramar.com, 2006-2029

La sintaxis a emplear es del tipo:

```
selectorElemento {background-image: valorDefinido; }
```

El valor definido puede ser none para indicar que no existe imagen de fondo, o puede definirse escribiendo lo siguiente: url(rutaDeLaImagen). No debe haber espacio entre url y el paréntesis dentro del que se coloca la ruta. Es decir, url(ruta) es correcto pero url (ruta) es incorrecto.

Como ruta de la imagen puede especificarse un nombre de archivo que tengamos en el mismo directorio en el que se encuentre el archivo HTML, por ejemplo url(barco.jpg), o también una ruta relativa a otro directorio. Por ejemplo url(images/barco.jpg) haría referencia a un archivo de nombre barco.jpg que se encuentra dentro del directorio images.

También puede especificarse una url completa, por ejemplo podría escribirse lo siguiente:

url(http://www.crimsoneditor.com/images/logo.jpg)

El contenido de la ruta puede escribirse entre comillas simples o dobles. Es decir, resulta válida cualquiera de estas tres opciones:

a) url(http://www.crimsoneditor.com/images/logo.jpg)
b) url("http://www.crimsoneditor.com/images/logo.jpg")
c) url('http://www.crimsoneditor.com/images/logo.jpg')

En algunos casos las rutas pueden tener caracteres extraños y estos caracteres pueden dar lugar a problemas al no ser capaz el navegador de reconocer la ruta.

| PROPIEDAD CSS background-image | |
|---|---|
| **Función de la propiedad** | Permite definir una imagen de fondo para un elemento. |
| **Valor por defecto** | none (no existe imagen de fondo). |
| **Aplicable a** | Todos los elementos. |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | none (establece que no existe imagen de fondo) |
| | Una ruta relativa (al fichero de imagen) |
| | Una ruta absoluta (al fichero de imagen) |
| | inherit (heredado del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | #menu1 {background-image: url(barco.jpg);} .imgVentana { background-image: url(http://crimsoneditor.com/images/logo.jpg);}<br>h1 { background-image: url("bgforh1.png"); }<br>.container1 {background-image: none;} |

© aprenderaprogramar.com, 2006-2029

En caso de que el archivo de imagen a que se aluda no esté disponible, la visualización será la misma que si estuviera establecido que background-image fuera none. Por tanto si tenemos un color de fondo y una imagen de fondo, en caso de no encontrarse o no estar disponible la url con el archivo de la imagen de fondo, se vería el color de fondo. Este es el motivo por el que se recomienda establecer color de fondo incluso si se pone una imagen de fondo opaca: en caso de no estar la imagen disponible, el color de fondo aportaría la seguridad de que se visualiza algo como fondo.

Para ver la aplicación práctica de estos conceptos vamos a utilizar el código HTML de base que venimos empleando a lo largo del curso.

Define los siguientes estilos css y visualiza el resultado en tu navegador:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {background-color: yellow;}
h1{
border: dashed thick purple; width: 55%; color: blue; background-color:pink;
margin:20px; padding:50px;
background-image:
url(http://lh5.ggpht.com/_PeVwghrmOec/TMkzEonRrcI/AAAAAAAAAHc/IxL8g0fTYtk/an_oliva_png.png);
}
```

El resultado será similar a la imagen anterior y lo interpretamos de la siguiente manera: el borde tiene partes no opacas por estar establecido como dashed. Por ello allí donde la línea es discontínua se ve la imagen de fondo o, en caso de que la imagen de fondo tenga transparencia (como en este caso), se ve el color de fondo.

Por defecto, la imagen se coloca en la parte superior izquierda alineada o encajada con el borde superior izquierdo, y a partir de aquí se repite hacia arriba (de ahí que se vean algunas manchas en el borde superior, hacia la izquierda (de ahí que se vean manchas en el borde izquierdo), hacia la derecha y hacia abajo. Este comportamiento puede ser cambiado utilizando otras propiedades de background (background-position y background-repeat) como veremos más adelante.

En caso de que la imagen fuera más grande que el espacio disponible en el elemento, sólo se mostraría una parte de la imagen (la parte que quepa).

Se pueden poner varias imágenes de fondo (aunque sólo en navegadores modernos; algunos navegadores más antiguos pueden no funcionar con esta posibilidad) . En este caso, la visualización es el resultado de crear un collage o pila de imágenes, situándose encima la primera en la enumeración, más abajo la siguiente y así sucesivamente hasta llegar a la capa final que sería el color de fondo. Al hacer esto, se crea un efecto de superposición donde la transparencia de una imagen deja ver

© aprenderaprogramar.com, 2006-2029

parcialmente lo que hay debajo. En caso de que la primera imagen fuera opaca, ocultaría todo lo que hay debajo.

Ejemplo: background-image: url("barco.png", "mar.jpg") da lugar a que la imagen del mar se vea debajo de la imagen del barco.

Prueba a visualizar el resultado de establecer una imagen de fondo con transparencia encima de otra imagen de fondo (por ejemplo background-image: url(http://lh5.ggpht.com/_PeVwghrmOec/TMkzEonRrcI/AAAAAAAAAHc/IxL8g0fTYtk/an_oliva_png.png), url(http://www.crimsoneditor.com/images/logo.jpg);

Prueba también a establecer una imagen de fondo para el elemento body y comprueba los resultados. Podrás ver cómo si toda la página tiene una imagen de fondo oscura, el texto negro se hace difícil de leer.

Un aspecto de interés radica en que se pueden usar imágenes de pequeño tamaño que situadas en un fondo y repetidas creen el efecto de papel tapiz para una página web. Por ejemplo podemos coger imágenes simples de forma cuadrada:

Estas imágenes, si están bien diseñadas, al repetirse crean un fondo uniforme o "papel tapiz" pareciendo que fueran una sola cosa. Esto supone una gran ventaja respecto a usar una imagen de gran tamaño por varios motivos:

a) Usando una imagen de fondo que se repite la página puede ser tan grande a lo ancho y a lo alto como queramos, no tenemos que preocuparnos de si la imagen va a ser lo suficientemente grande para cubrir el espacio de la página web.

b) Usando una imagen pequeña que se repite facilitamos que la página web cargue más rápidamente al tener el navegador que recibir una pequeña imagen y repetirla, en lugar de tener que recibir una gran imagen. De este modo, podemos cargar por ejemplo una imagen de 4 Kb cuya carga es muy rápida en lugar de una imagen de 400 Kb cuya carga sería muy lenta.

**PROPIEDAD BACKGROUND-REPEAT**

Esta propiedad nos permite definir si una imagen de fondo de tamaño inferior al disponible debe repetirse. También se puede especificar si la repetición debe ser horizonal (eje x), vertical (eje y) o en ambos sentidos.

La sintaxis a emplear es del tipo:

```
selectorElemento { background-repeat: valorDeRepetición; }
```

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS background-repeat | |
|---|---|
| **Función de la propiedad** | Permite definir si una imagen de fondo debe repetirse y cómo. |
| **Valor por defecto** | repeat (la imagen se repite horizontal y verticalmente). |
| **Aplicable a** | Todos los elementos. |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | repeat (repetición horizontal y vertical) |
| | repeat-x (repetición sólo horizontal) |
| | repeat-y (repetición sólo vertical) |
| | no-repeat (la imagen se muestra sólo una vez, sin repeticiones) |
| | inherit (heredado del elemento padre) |
| **Ejemplos aprenderaprogramar.com** | #menu1 {background-image: url("barco.jpg"); background-repeat: no-repeat;}<br>.imgVentana { background-image: url(ut.jpg); background-repeat: repeat-x;}<br>h1{ background-image: url(ut.jpg); background-repeat: repeat-y;}<br>.container1 { background-image: url(ut.jpg); background-repeat: repeat;} |

Opcionalmente puede usarse otra sintaxis (aunque no la vemos por el momento recomendable porque en algunos navegadores puede no ser reconocida) basada en dos palabras clave, una para definir el comportamiento en el eje x u horizontal y otra para establecer el comportamiento en el eje y o vertical. La sintaxis es de tipo:

```
selectorElemento { background-repeat: valorParaEjeX valorParaEjeY; }
```

Escribir background-repeat: repeat-x; sería equivalente a escribir background-repeat: repeat-x no-repeat;

Hay otras palabras clave, aunque no son reconocidas por todos los navegadores. En concreto, **space** significaría que la imagen se repite hasta donde sea posible pero sin cortarse en ningún momento. La palabra **round** vendría a significar que la imagen se repite hasta donde sea posible y en caso de quedar un espacio sobrante, se produce un redimensionamiento del tamaño de la imagen hasta ajustarse al espacio disponible. Ten en cuenta que si estableces la propiedad background-repeat con una sintaxis que el navegador no reconoce, le aplicará su valor por defecto, que es repeat. Ten en cuenta también que no todos los navegadores responden igual.

Prueba a establecer la propiedad background-repeat sobre el código que vimos anteriormente y comprueba sus efectos.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO 1**

Crea un documento HTML con 4 elementos div de 250 píxeles de ancho y 250 píxeles de alto, todos ellos con un margin de 30 píxeles en todas direcciones y un padding de 30 píxeles en todas direcciones. En cada uno de los elementos div coloca una imagen de fondo diferente y un background-color diferente. Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**EJERCICIO 2**

Crea dos páginas web cumpliendo estos requisitos:

a) Una página web debe tener una única imagen de gran tamaño (por ejemplo 1024x768 píxeles) como imagen de fondo, sin repetición de la misma.

b) Una página web debe tener una imagen de pequeño tamaño (por ejemplo 135x135 píxeles) que mediante el uso de la propiedad repeat se expanda como fondo de la página web creando un efecto tapiz.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01031D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**PROPIEDADES DE FONDO**

CSS tiene diferente propiedades relacionadas con el fondo de las cajas del box-model o modelo de cajas. Ya conocemos algunas de ellas. Vamos a ampliar el conocimiento sobre los fondos estudiando otras propiedades: background-position, background-attachment, background-clip, background-origin, background-size y la propiedad shortand background.

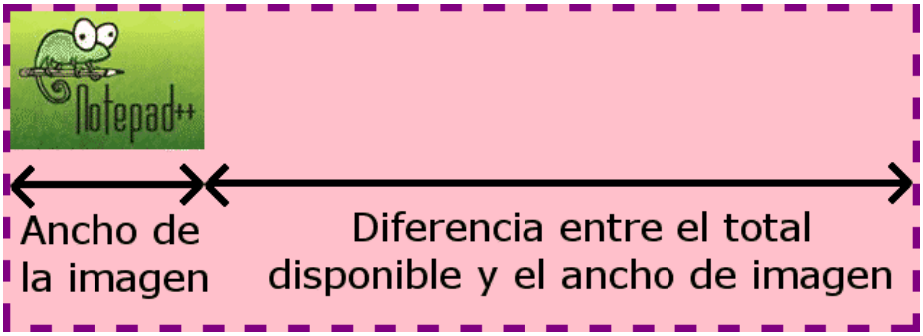
**PROPIEDAD BACKGROUND-POSITION**

Esta propiedad nos permite establecer la posición de una imagen de fondo. Por defecto hemos visto que la imagen de fondo se sitúa en la parte superior izquierda del elemento (justo en la esquina interior del borde), pero podremos especificar otra posición usando esta propiedad.

La sintaxis a emplear es del tipo:

```
selectorElemento {background-position: especificaciónDePosición; }
```

La especificación de posición puede hacerse de distintas maneras. Una de ellas es escribir dos porcentajes, representando el primero de ellos el desplazamiento x (horizontal) y el segundo el desplazamiento y (vertical). Ahora bien, ¿el % de qué medida es la que se usa? Se usa la medida del ancho y alto total disponible menos el ancho y alto de la imagen.




A efectos prácticos, los resultados aplicando porcentajes son estos:

0% 0%: la imagen se coloca en posición superior izquierda (posición por defecto, no hay desplazamiento).

100% 0%: la imagen se coloca en posición superior derecha (se desplaza el total del espacio diferencia en horizontal).

0% 100%: la imagen se coloca en posición inferior izquierda (se desplaza el total del espacio diferencia en vertical).

100% 100%: la imagen se coloca en posición inferior derecha (se desplaza el total del espacio diferencia en vertical y en horizontal).

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS background-position | |
|---|---|
| **Función de la propiedad** | Permite definir la posición de la imagen de fondo de un elemento. |
| **Valor por defecto** | 0% 0%. |
| **Aplicable a** | Todos los elementos. |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | Dos porcentajes (primer % indica horizontal, segundo % indica vertical) |
| | Un porcentaje (indica % horizontal, y en la vertical se produce el centrado automático de la imagen) |
| | Dos unidades de medida, relativas o absolutas (primera unidad indica el desplazamiento horizontal respecto al punto de inicio, segunda unidad indica el desplazamiento vertical respecto al punto de inicio). |
| | Una unidad relativa o absoluta (indica desplazamiento horizontal respecto al punto de inicio, y en la vertical se produce el centrado automático de la imagen) |
| | Dos palabras clave a elegir entre left, center y right en la horizontal y top, center y bottom en la vertical (primera palabra indica posición en la horizontal y segunda palabra clave indica posición en la vertical) |
| | Una palabra clave a elegir entre left, center y right (indica posición en la horizontal; en la vertical se produce el centrado automático de la imagen). |
| | inherit (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | background-position: 25% 45%;<br>background-position: 250px 2.55em;<br>background-position: center bottom;<br>background-position: right;<br>background-position: -10%;<br>background-position: -10px -10px; |

Se admiten valores negativos para especificar la posición de la imagen de fondo. Los valores negativos se llevarán la imagen hacia fuera del espacio disponible y posiblemente quede una parte de la imagen no visible. Una utilidad que puede tener el establecer un valor negativo es para posicionar la imagen encima del borde izquierdo o del borde superior.

Si usamos dos o más imágenes de fondo podríamos fijar sus posiciones separando la especificación de posición de cada una de ellas como en este ejemplo, (aunque algunos navegadores antiguos pueden no reconocer esta sintaxis):

```
background-image: url(http://i.imgur.com/afC0L.jpg), url(http://www.crimsoneditor.com/images/logo.jpg);
background-repeat: no-repeat;
background-position: -10px -10px, right bottom;
```

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD BACKGROUND-ATTACHMENT**

Esta propiedad nos permite establecer si la imagen se comporta como una imagen normal, y se muestra en una posición concreta, o si se desplaza a medida que el usuario se desplaza por la página web haciendo scroll de modo que aparece siempre fija como fondo de un elemento determinado (siempre que ese elemento esté visible en la pantalla).

El comportamiento normal se corresponde con la palabra clave **scroll**, mientras que el aparecer siempre fija como fondo cuando hay desplazamiento se corresponde con la palabra clave **fixed**. Hay otra nueva palabra clave, sólo soportada por las últimas versiones de los navegadores, que es **local**, cuyo significado es que si el elemento dentro del cual está la imagen tiene una barra de desplazamiento o scroll, la imagen permanece fija en ese elemento mientras se haga scroll en él.

La mejor forma de entender el comportamiento de background-attachment es realizar una prueba. Para ello vamos a utilizar el código HTML de base que venimos empleando a lo largo del curso.

Define los siguientes estilos css y visualiza el resultado en tu navegador:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {background-color: yellow;
background-image: url(http://i.imgur.com/afC0L.jpg);
background-repeat: no-repeat;
background-attachment: fixed;
}
h1 {height: 450px; width: 2000px;}
h2 {height: 450px;}
```

Comprueba cómo la imagen se mantiene en la parte superior izquierda de la página web aunque te desplaces verticalmente hacia abajo (hemos introducido una altura y anchura desproporcionada para los elementos h1 y h2 simplemente para forzar que la página se expanda en la vertical y en la horizontal y poder comprobar mejor el efecto que se genera).

Un aspecto importante a tener en cuenta es que si se establece background-attachment: fixed; la posición de la imagen de fondo que visualizaremos ya no es la esquina interior superior izquierda del elemento, sino que es la esquina superior izquierda de la ventana visible en el navegador. Esto puede hacer que la imagen de fondo no se vea (por haberse desplazado hasta la esquina superior izquierda de la ventana del navegador, quedando fuera del área de fondo del elemento), o que aparezca cortada aparentando ser una visualización errónea. Por eso muchos programadores y diseñadores web utilizan background-attachment preferentemente para poner imágenes de fondo al elemento body (cuya área de visualización coincide normalmente con la ventana del navegador). Si se usa background-attachment con otros elementos, hay que establecer la posición adecuadamente para que la visualización sea correcta.

Podrían especificarse varias imágenes de fondo y varios comportamientos de esta propiedad, de la misma forma que vimos para la propiedad background-position.

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS background-attachment | |
|---|---|
| **Función de la propiedad** | Permite fijar una imagen de fondo fija aunque haya desplazamientos. |
| **Valor por defecto** | scroll |
| **Aplicable a** | Todos los elementos. |
| **¿Se hereda directamente?** | No |
| **Valores posibles para esta propiedad** | scroll (comportamiento normal) |
| | fixed (la imagen se mantiene como fondo aún en desplazamientos) |
| | local (la imagen se mantiene como fondo si el elemento tiene barras de desplazamiento y el usuario se mueve dentro del elemento) |
| | inherit (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | body { background-image: url(http://i.imgur.com/afC0L.jpg);<br>background-repeat: no-repeat;<br>background-attachment: fixed; } |

**OTRAS PROPIEDADES DE FONDO**

Además de las propiedades de fondo que hemos visto, CSS contempla otras menos comunes que son las siguientes:

a) **background-clip**: determina la ubicación del área de fondo. Sus valores posibles son **border-box**, lo cual situa el área de fondo en la esquina exterior del borde en la parte superior izquierda, **padding-box**, lo cual sitúa el área de fondo en la esquina interior del borde en la parte superior izquierda del borde, o **content-box**, lo cual situa el área de fondo en la esquina superior izquierda del contenido quedando excluida el área de padding. El valor de defecto es **border-box**, lo que da lugar a que si hay un color de fondo se sitúe debajo del borde. La propiedad background-clip afecta tanto al color de fondo como a la imagen de fondo, sin embargo hay que tener en cuenta que la posición de la imagen de fondo es la que hemos indicado al hablar de background-image y no se ve afectada por border-clip. Es decir, la imagen se mantiene en su posición a no ser que especifiquemos otra. Con border-clip la imagen podría verse cortada (si reducimos el área de fondo). Esta propiedad puede no funcionar correctamente en navegadores antiguos.

b) **background-origin**: determina respecto a qué origen se ha de considerar la posición establecida mediante la propiedad background-position. Sus valores posibles son **border-box**, **padding-box** y **content-box**. Su valor de defecto es **padding-box**, de ahí que una imagen se sitúe en la esquina interior de la parte superior izquierda del borde. Si queremos situar la imagen en la parte exterior de la esquina (alineada con el borde) basta establecer el background-origin: border-box;

© aprenderaprogramar.com, 2006-2029

c) **background-size**: permite establecer el tamaño de las imágenes de fondo. Se pueden utilizar las palabras clave **cover** (indica que la imagen debe escalar su tamaño manteniendo sus proporciones hasta que encaje en una dimensión del fondo para cubrir completamente el área de fondo; la imagen puede verse cortada), **auto** (indica que la imagen se mantendrá con sus dimensiones originales si no se ha modificado ninguna de ellas, o sin deformarse en caso de haber especificado una de las dos dimensiones) ó **contain** (indica que la imagen debe escalarse manteniendo sus proporciones hasta alcanzar una de las dos dimensiones del área de fondo; la imagen no se cortará en ningún caso, pero el área de fondo puede quedar parcialmente descubierta). Esta propiedad puede no ser reconocida en navegadores antiguos.

La propiedad background-size también puede establecerse usando unidades de medida relativas o absolutas y especificando dos medidas o solo una medida.

Si se especifica solo una medida como background-size: 30%; ó background-size: 4.5em ó background-size: 50px, se entiende que esta medida establece el ancho (width) de la imagen. La altura automáticamente se trata como si tuviera valor auto.

Si se especifican dos medidas, la primera de ellas se considerará de aplicación a la dimensión horizontal y la segunda a la dimensión vertical. Por ejemplo en background-size: 4.5em 2.25em; la dimensión horizontal se establece en 4.5em y la vertical en 2.5em.

En el caso de usar porcentajes, el % se calcula respecto a las dimensiones del área de fondo de la imagen establecidas por background-origin. Por defecto coincide con el área de padding, pero puede cambiarse si se cambia el valor de la propiedad background-origin. También genera una alteración el uso de background-attachment: fixed; porque el área de fondo en este caso es el área visible de la ventana del navegador.

**PROPIEDAD SHORTAND BACKGROUND**

Esta propiedad nos permite establecer de forma conjunta las diferentes propiedades de fondo.

La sintaxis a emplear es del tipo:

```
background: { valorColorDeFondo valorImagenDeFondo valorRepeticiónImagen valorPosiciónImagen
valorSizeImagen valorOriginPosición ValorClipFondo valorAttachmentFondo } ;
```

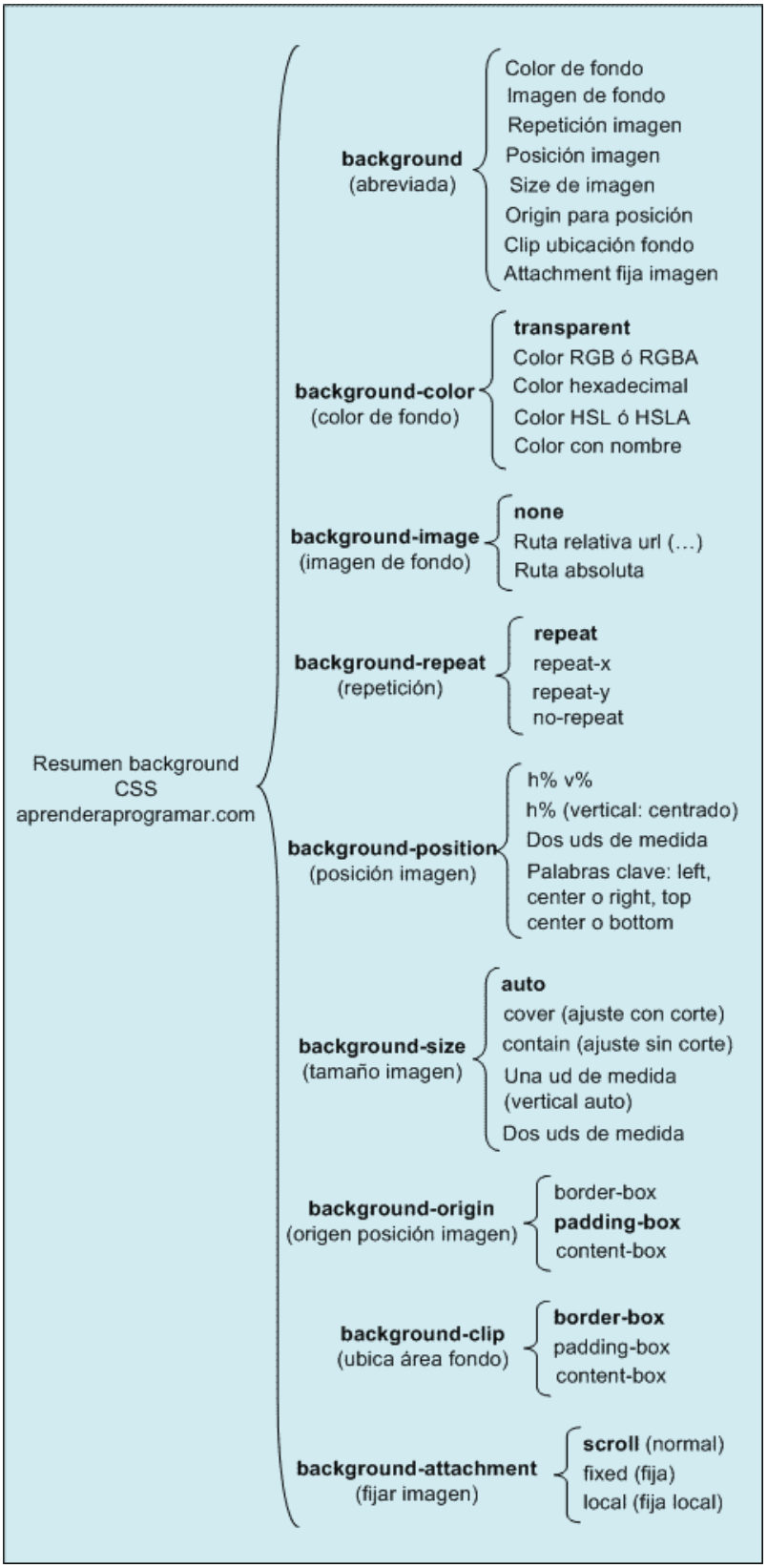
No es necesario seguir un orden concreto para escribir las propiedades. Si se dejan algunos valores sin especificar, el navegador aplicará los valores por defecto que tengan esas propiedades sin especificar. Por ejemplo si escribimos: background: { blue url(img1.jpg) } ; los valores correspondientes a la repetición, attachment, posición, etc. serán los valores por defecto para estas propiedades.

Un aspecto importante a tener en cuenta si se usa la propiedad shortand background, es que cualquier propiedad no especificada se sobreescribe a su valor por defecto anulando los posibles valores anteriores que se hubieran especificado. Por ejemplo si escribimos:

.content43 {background-repeat: no-repeat; background: url(http://i.imgur.com/afC0L.jpg); } podríamos pensar que la imagen no se a repetir porque hay indicado un valor no-repeat. Sin embargo, la imagen sí

© aprenderaprogramar.com, 2006-2029

se repetirá porque aunque no lo hemos indicado específicamente, al incluir una propiedad shortand con posterioridad a la especificación inicial de background-repeat, el valor de repetición se sobreescribe pasando a ser su valor de defecto (que es repeat, de ahí que la imagen sí se repetirá). Es un aspecto con el que conviene ser cuidadosos. En principio, si elegimos usar la propiedad shortand, no debemos realizar otras especificaciones independientes para no incurrir en contradicciones.




© aprenderaprogramar.com, 2006-2029

El esquema anterior resume las propiedades relacionadas con el fondo (background) CSS. Obviamente no debemos pretender recordar de memoria todas las propiedades ni todos sus valores posibles o palabras clave asociadas. No obstante, debemos tener conocimiento de las posibilidades que nos ofrece CSS para el manejo de fondos y saber buscar información y aplicar nuestro conocimiento de la lógica de CSS cada vez que nos sea necesario. Para ello es conveniente realizar pruebas y escribir código en nuestro ordenador.

**CAPAS DE VISUALIZACIÓN**

Hemos visto que en el box-model o modelo de cajas CSS intervienen diferentes elementos como un contenido, un borde, un relleno o padding, una imagen de fondo, un color de fondo y un margen. Cuando varios elementos se superponen unos tienen que visualizarse encima de otros, y de ahí que hablemos de modelo tridimensional o modelo de capas para las cajas CSS. El orden en que se muestran los elementos es:

a) El borde se superpone al resto de elementos (capa 1)
b) El contenido y el padding se sitúan debajo del borde (capa 2)
c) La imagen de fondo se sitúa debajo del contenido y el padding (capa 3)
d) El color de fondo y el margen se sitúan debajo de la imagen de fondo (capa 4)

**EJERCICIO**

Crea un documento HTML con 4 elementos div de 400 píxeles de ancho y 400 píxeles de alto, todos ellos con un margin de 40 píxeles en todas direcciones y un padding de 40 píxeles en todas direcciones. En cada uno de los elementos div crea un borde y coloca una imagen de fondo diferente y un background-color diferente. Usa la propiedad background-position para hacer que la imagen esté centrada tanto vertical como horizontalmente respecto al borde del div (por ejemplo, si una imagen mide 100x100 píxeles, deberá existir la misma distancia hasta el borde del div en las cuatro direcciones). Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01032D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**POSICIONAMIENTO CSS**

Hasta ahora conocemos que los elementos HTML se dividen en dos grandes tipos: elementos block que tienden a ocupar el espacio disponible a todo lo ancho y en caso de existir varios se sitúan unos debajo de otros y elementos inline que ocupan el espacio necesario dentro de una línea y en caso de existir varios se sitúan uno junto a otro en la misma línea (siempre que haya espacio).

Los principales elementos HTML tipo block son:

a) **Divisores**: div
b) **Párrafos**: p
c) **Formularios**: form
d) **Títulos**: h1, h2, h3, h4, h5, h6.
e) **Listas**: ul, ol
f) **Elementos de listas**: li
g) **Tablas**: table
h) **Otros**: center, pre, tbody, td, th, tr…

Los principales elementos HTML tipo inline son:

a) **Links**: a
b) **Divisores**: span
c) **Imágenes**: img (aunque este elemento tiene características especiales)
d) **Controles de formulario**: input, label
e) **Otros**: strong, u, select…

Queda fuera de la clasificación el elemento body, que engloba todo el espacio disponible en el dispositivo de visualización.

Cuando tenemos distintos elementos tipo block cada uno se coloca en una nueva línea distinta a la del elemento anterior, es decir, se colocan uno debajo de otro:

Bloque1 tiene este contenido de texto

Bloque2 tiene este contenido de texto

Bloque3 tiene este contenido de texto …

Cuando tenemos distintos elementos en línea se colocan uno junto a otro:

ElementoEnLínea1 ElementoEnLínea2 ElementoEnLínea3

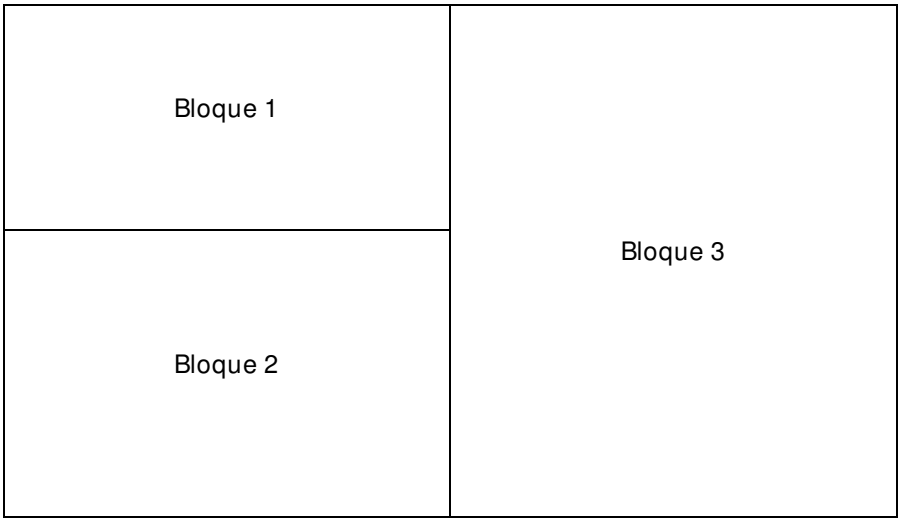
© aprenderaprogramar.com, 2006-2029

Además los elementos en línea que están dentro de un bloque se insertan en la misma línea que el contenido del bloque. Por ejemplo:

Este texto es un párrafo **y esto es un link inline** y aquí continúa el párrafo.

Recordar que las propiedades width y height sólo son aplicables a elementos tipo block y elementos insertados en una posición que son reemplazados por un objeto (entre ellos img, input, textarea, select, object).

Ahora bien, cuando creamos páginas web necesitaremos que elementos de dimensiones determinadas se sitúen en posiciones determinadas. Por ejemplo podemos querer tener tres elementos con dimensiones width y height definidas (por tanto bloques) situados en determinadas posiciones como vemos en este esquema:




Con el código simple que hemos visto hasta ahora esto no sería posible, ya que unos elementos se situarían debajo de otros, sin cumplir nuestro objetivo de situarlos en posiciones específicas.

CSS tiene distintas propiedades y posibilidades que nos van a permitir gestionar el posicionamiento de elementos para crear diseños atractivos y conforme a nuestras necesidades. Comenzaremos el estudio de estas posibilidades con la propiedad CSS position.

**PROPIEDAD POSITION**

Esta propiedad nos permite establecer la posición de un elemento.

La sintaxis a emplear es del tipo:

```
selectorElemento {position: especificaciónDePosición; }
```

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS position | |
|---|---|
| **Función de la propiedad** | Permite definir el tipo de posición de un elemento y su comportamiento. |
| **Valor por defecto** | static |
| **Aplicable a** | Elementos tipo block y algunos elementos inline especiales como img. |
| **Valores posibles para esta propiedad** | **static** (comportamiento normal o por defecto) |
| | **relative** (permite que un elemento se desplace respecto a lo que hubiera sido su posición normal; el resto de elementos continúan en su posición ignorando al que se desplaza, lo que puede crear superposiciones; el espacio libre que deja el elemento queda libre). |
| | **absolute** (permite que un elemento se desplace respecto al origen de coordenadas del primer elemento contenedor posicionado ó respecto a la esquina superior izquierda de la ventana de visualización; el resto de elementos actúan como si el desplazado no existiera, por lo que su espacio será ocupado por otros elementos; puede crear superposiciones) |
| | **fixed** (permite fijar un elemento en una posición respecto al origen de coordenadas del primer elemento contenedor posicionado ó respecto a la esquina superior de la ventana de visualización; el elemento se mantendrá en la ventana de visualización o viewport, siempre en una misma posición aunque el usuario se desplace por la web haciendo scroll) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {position: relative; }<br>.elementoFijado {position: fixed; } |

La posición absoluta de un elemento que no está dentro de ningún otro se determina respecto a la parte superior izquierda del elemento html (parte superior izquierda de la ventana donde se visualiza la página web; normalmente coincide con la parte superior izquierda del elemento body).

Si un elemento está dentro de varios contenedores (p.ej. dentro de varios div) el origen de coordenadas se establece respecto a la esquina superior izquierda del contenedor que tenga un valor de position establecido y distinto de static o, en caso de no existir ningún elemento contenedor posicionado, respecto a la esquina superior izquierda del elemento html (parte superior izquierda de la ventana donde se ve la página web). Esto se comprenderá con los ejemplos de código que veremos.

Los ejemplos de código sobre el uso de esta propiedad los veremos una vez hayamos estudiado algunas propiedades que se usan habitualmente junto a ésta (top, right, bottom y left).

Esta no es la única propiedad para establecer posiciones de elementos. Hay otras propiedades importantes como float, que veremos más adelante.

© aprenderaprogramar.com, 2006-2029

**PROPIEDADES TOP, RIGHT, BOTTOM Y LEFT**

Estas propiedades nos permiten definir el desplazamiento de la posición de un elemento respecto a un origen de coordenadas y el origen de coordenadas que se toma.

| PROPIEDADES CSS top, right, bottom y left | |
|---|---|
| **Función de la propiedad** | Permiten definir el desplazamiento de un elemento y el origen de coordenadas que se tomará para el mismo. |
| **Valor por defecto** | Auto |
| **Aplicable a** | Elementos que están posicionados (no aplica si la posición es static). |
| **Valores posibles para estas propiedades** | auto (no hay desplazamiento especificado y como origen de coordenadas se toma la esquina superior izquierda del elemento padre) |
| | Una unidad de medida relativa (se admiten porcentajes). |
| | Una unidad de medida absoluta. |
| | inherit (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #menu1 { position: relative; top: 40px; left: 2em; }<br>#moduloEGM { position: absolute; bottom: 5%; left: 30px; } |

Para elementos con posición relative estas propiedes definen cuánto se desplaza el elemento respecto a lo que sería su posición normal.

Para elementos con posición absolute o posición fixed estas propiedades definen cuánto se desplaza el elemento respecto a las coordenadas de origen.

Se admiten valores de desplazamientos negativos.

En el caso de usar porcentajes, éste se calcula:

a) Respecto al valor de altura (height) del elemento si se aplican con top ó bottom.
b) Respecto al valor de anchura (width) del elemento si se aplican con right ó left.

No tiene sentido utilizar top y bottom al mismo tiempo, porque sería decir que el elemento sube y baja. Hay que utilizar sólo una de estas dos propiedades. Lo mismo podemos decir para right y left.

**EJEMPLOS DE POSICIONAMIENTO RELATIVO**

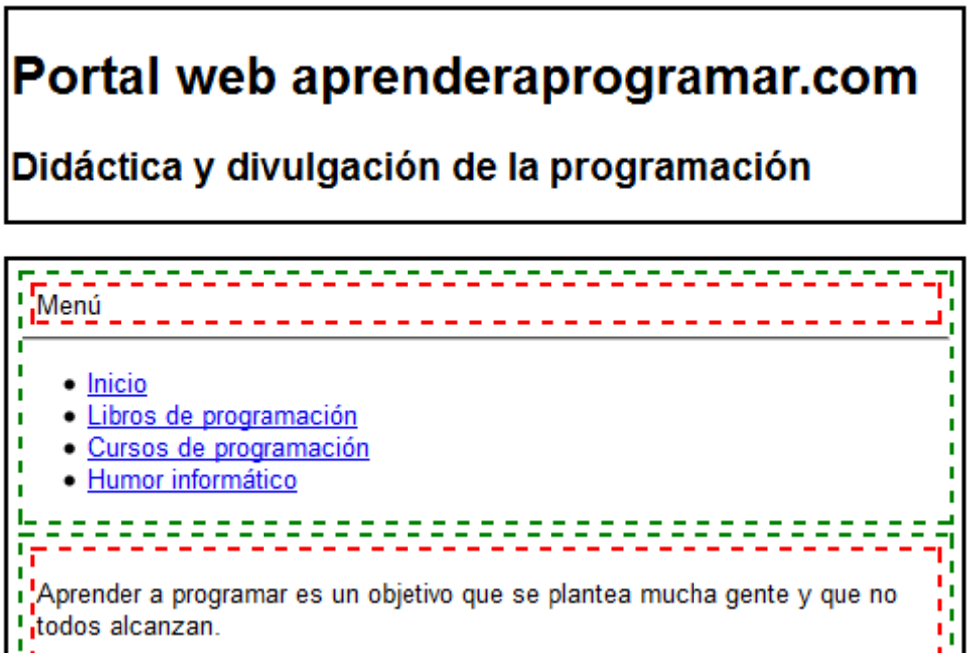
A continuación veremos algunos ejemplos de uso de las propiedades anteriormente especificadas. Para ello vamos a utilizar el código HTML de base que venimos empleando a lo largo del curso.

Define los siguientes estilos css y visualiza el resultado en tu navegador:

© aprenderaprogramar.com, 2006-2029

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
body {width: 600px;}
div {border-style: solid;}
div div {border-style: dashed; border-color: green; margin: 5px;}
div div div {border-style: dashed; border-color: red; margin: 5px;}
```

Con esto lo único que hemos hecho es poner bordes y márgenes para visualizar las cajas que conforman la página web. El resultado debe ser similar a este:




Vemos un div en color verde correspondiente al menú (dentro del cual hay otro div en color rojo con el texto "Menú", una línea generada por el <hr/> y una lista ul con los items de menú).

Accede al código HTML y establece como id de dicho div "menu1". El código HTML de ese fragmento de web quedará así:

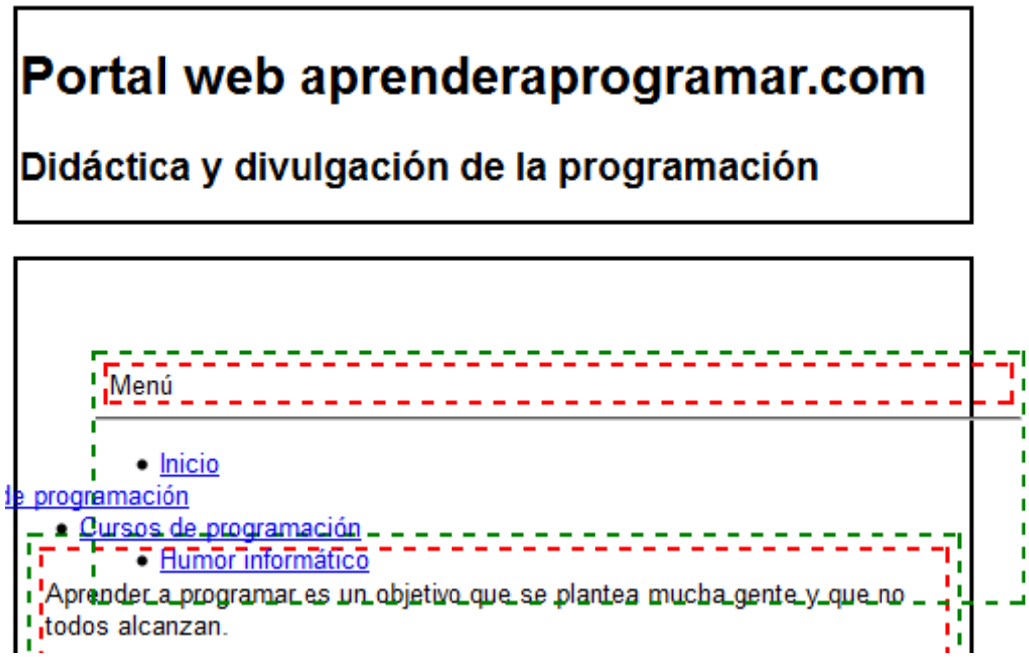
```
<!-- menu -->
<div id="menu1">
<div>Menú</div>
<hr/>
<ul>
<li><a href="#">Inicio</a></li>
<li><a href="libros.html" >Libros de programación</a> </li>
<li><a href="cursos.html" >Cursos de programación</a> </li>
<li> <a href="humor.html" >Humor informático</a> </li>
</ul>
</div>
<!-- fin menu -->
```

Ahora vamos a modificar la definición de estilos CSS para hacer lo siguiente: modificaremos la posición del elemento con id "menu1" para que se desplace 50 pixeles hacia abajo y 40 pixeles hacia la derecha respecto a lo que debía ser su posición original. Además vamos a hacer que el segundo elemento de la

© aprenderaprogramar.com, 2006-2029

lista de items de menú se desplace 150 pixeles hacia la izquierda respecto a lo que debiera ser su posición normal y el tercer elemento de la lista 50 pixeles a la izquierda respecto de lo que debiera ser su posición normal. Para ello usamos este código y visualizamos el resultado:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
body {width: 600px;}
div {border-style: solid;}
div div {border-style: dashed; border-color: green; margin: 5px;}
div div div {border-style: dashed; border-color: red; margin: 5px;}
#menu1 { position:relative; top:50px; left: 40px; }
li:nth-child(2) {position: relative; right: 150px;}
li:nth-child(3) {position: relative; right: 50px;}
```




Comprobamos que la caja del div se ha desplazado (con todo su contenido). Igualmente los elementos de lista indicados se han desplazado. Comprobamos algunas cosas curiosas: al desplazar la caja del div, ésta sale por fuera de las dimensiones del elemento body y **se ha superpuesto** con la caja que existía debajo. Al desplazar el segundo elemento de lista 150 px hacia la izquierda, el elemento ha quedado cortado y no se visualiza parte de él (si el desplazamiento hubiera sido todavía mayor, no se vería ninguna parte de este item de la lista aparentando haber "desaparecido". El tercer elemento de la lista se ve desplazado pero no se ha salido de la pantalla.

Para desplazar hacia abajo usamos top y para desplazar hacia la derecha usamos left. Puede parecer confuso porque left significa izquierda, pero usar left no significa "desplazar hacia la izquierda", sino "respecto a lo que sería la posición normal de su lateral izquierdo, desplazar lo indicado", correspondiendo un valor positivo a "desplazar hacia la derecha" y un valor negativo a "desplazar hacia la izquierda".

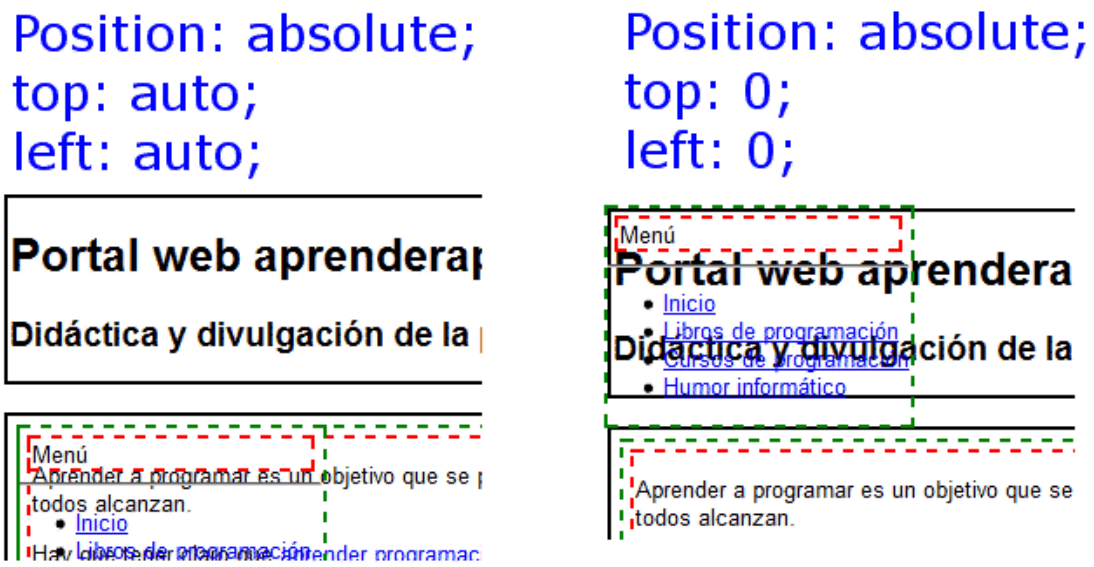
© aprenderaprogramar.com, 2006-2029

**EJEMPLOS DE POSICIONAMIENTO ABSOLUTO**

Seguimos trabajando sobre el HTML y hoja de estilos anterior. Define los siguientes estilos css y visualiza el resultado en tu navegador:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width: 600px;}
div {border-style: solid;}
div div {border-style: dashed; border-color: green; margin: 5px;}
div div div {border-style: dashed; border-color: red; margin: 5px;}
 #menu1 { position:absolute;}
```

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width: 600px;}
div {border-style: solid;}
div div {border-style: dashed; border-color: green; margin: 5px;}
div div div {border-style: dashed; border-color: red; margin: 5px;}
 #menu1 { position:absolute; left:0; top:0;}
```




El código #menu1 { position:absolute;} equivale a #menu1 { position:absolute; left: auto; top: auto;} lo que supone que se toma como origen de coordenadas el del elemento padre. De ahí que en la imagen de la izquierda se vea la caja en su posición natural, pero al estar posicionada de forma absoluta es ignorada por el resto de elementos y su espacio es ocupado como si no estuviera ahí.

El código #menu1 { position:absolute; left:0; top:0;} significa que el origen de coordenadas será el del primer contenedor posicionado o, si no existe, la esquina superior izquierda de la ventana de visualización. En este caso se toma la esquina superior izquierda de la ventana como origen de coordenadas.

Cuando se establece position en absolute o fixed el ancho del bloque se ajusta al contenido (excepto si está establecido específicamente mediante la propiedad width).

© aprenderaprogramar.com, 2006-2029

Prueba a cambiar los valores de left y top y comprueba sus resultados. Por ejemplo si estableces valores left: -50px; top: -50px; verás cómo la caja "se sale de la pantalla" por la esquina superior izquierda. Si estableces right:0; top:0; verás que la caja se sitúa en la parte superior derecha de la pantalla.

Supón ahora que quieres usar como origen de coordenadas absolutas la esquina superior izquierda del div contenedor del que estamos usando como ejemplo. Ese div contenedor tiene, aparte del menú, varias cosas más como texto con imágenes y un formulario.

Para hacer que el div del menú tome para origen de coordenadas el div contenedor en lugar de la esquina superior izquierda de la pantalla hemos de posicionar el contenedor. **Posicionarlo significa darle un valor de position distinto del valor por defecto (static).** Esto podemos hacerlo de varias maneras. Por ejemplo podríamos poner un id al div contenedor y aplicarle una regla nombreEscogido {position: relative; }. No es necesario establecer un desplazamiento ya que nosotros ahora lo único que queremos es que dicho elemento se identifique como un elemento posicionado, y para ello establecer su propiedad position como relative es suficiente.

También podemos hacerlo como vemos en este código:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
body {width: 600px; border-style:dotted;}
div {border-style: solid; position:relative;}
div div {border-style: dashed; border-color: green; margin: 5px;}
div div div {border-style: dashed; border-color: red; margin: 5px;}
#menu1 { position:absolute; left:150px; top:30px; }
```

Hemos establecido para todos los div su propiedad position como relative. Al ir a realizar el posicionamiento del elemento con id menu1 el navegador busca el primer div contenedor posicionado y realiza el desplazamiento respecto a su esquina superior izquierda. Comprueba la diferencia al mantener todos los div con position:relative; y visualiza cómo cambia el resultado respecto a eliminar ese posicionamiento para todos los div: el origen de coordenadas que se toma es distinto.

**EJEMPLOS DE POSICIONAMIENTO FIXED**

Seguimos trabajando sobre el HTML y hoja de estilos anterior. Escribe y comprueba los resultados con este código:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
body {width: 600px; height:2400px; border-style:dotted;}
div {border-style: solid;}
div div {border-style: dashed; border-color: green; margin: 5px;}
div div div {border-style: dashed; border-color: red; margin: 5px;}

 #menu1 { position:fixed; left:50; top:0;}
```

Hemos establecido un valor de height para body aleatoriamente grande para forzar que aparezcan barras de scroll vertical en la página web.

© aprenderaprogramar.com, 2006-2029

Al hacer scroll, comprobarás que la caja del elemento con id menu1 se mantiene fija en una posición de la página aunque nos desplacemos con el scroll.

El comportamiento del origen de coordenadas con fixed es igual al que hemos explicado con absolute.

**EJERCICIO 1**

Define un documento HTML con un div padre (divPadre), dentro del cual existan otras 3 cajas contenedoras div (div1, div2 y div3), cada una de ellas con unas dimensiones de 300x300px, 40 píxeles de margin en todas direcciones, 30 píxeles de padding en todas direcciones y un background color diferente. Usando posicionamiento relativo genera un desplazamiento de los div de la siguiente manera:

a) El div 1 deberá desplazarse 200 píxeles a la derecha y 100 píxeles hacia abajo respecto a lo que sería su posición normal.

b) El div 2 deberá desplazarse 50 píxeles a la izquierda y 50 píxeles hacia arriba respecto a lo que sería su posición normal.

c) El div 3 deberá desplazarse 450 píxeles a la derecha y 300 píxeles hacia arriba respecto a lo que sería su posición normal.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**EJERCICIO 2**

Define un documento HTML con 3 cajas contenedoras div (div1, div2 y div3), la primera con unas dimensiones de 600x600px y un background color amarillo. La segunda con dimensiones 400x400px y un background color verde. La tercera con dimensiones 100x100px y background color azul. Usando posicionamiento absoluto establece para el div2 y el div3 el mismo origen que para el div1, de modo que el efecto generado sea ver un cuadrado amarillo dentro del cual hay un cuadrado verde dentro del cual hay un cuadrado azul. Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**EJERCICIO 3**

Define un documento HTML con varios div que contengan suficiente texto como para que la página se muestre con scroll (barras de desplazamiento). El primero de los div debe contener el texto <<Esta página web utiliza cookies. Si continúa navegando acepta el uso de cookies.>>, un valor height (altura) de 100 píxeles y color de fondo amarillo. Usando posicionamiento fixed, fija este div en la parte superior de la página de modo que se continúe visualizando aún cuando hagamos scroll. Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01033D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## DISPLAY CSS

Los elementos HTML se dividen en dos grandes tipos: elementos block que tienden a ocupar el espacio disponible a todo lo ancho y en caso de existir varios se sitúan unos debajo de otros y elementos inline que ocupan el espacio necesario dentro de una línea y en caso de existir varios se sitúan uno junto a otro en la misma línea (siempre que haya espacio). La propiedad display nos permite alterar el tipo de caja con que se muestra un elemento.

Sabemos que por ejemplo los elementos tipo título como h1 son elementos block, y por tanto de forma "natural" tienden a ocupar todo el ancho de la página. Por el contrario elementos como los links a o las imágenes son elementos inline. Veamos cómo la propiedad display permite alterar estas cualidades.

## PROPIEDAD DISPLAY

Esta propiedad nos permite establecer el tipo de caja que el navegador empleará para visualizar un elemento, siendo los tipos más comunes inline y block, aunque existen bastantes otros.

La sintaxis a emplear es del tipo:

```
selectorElemento {display: especificaciónDeVisualización; }
```

| PROPIEDAD CSS display | |
|---|---|
| **Función de la propiedad** | Permite definir el tipo de posición de caja para visualizar un elemento. |
| **Valor por defecto** | Depende del elemento (inline para elementos inline y block para elementos tipo block) |
| **Aplicable a** | Todos los elementos. |
| **Valores posibles para esta propiedad** | **inline** (el elemento se muestra en una caja inline) |
| | **block** (el elemento se muestra en una caja block). |
| | **none** (el elemento no se muestra; el efecto es como si no existiera, por lo que su espacio será ocupado por otros elementos) |
| | **list-item** (el elemento se comporta como si fuera un elemento li) |
| | **inline-block** (el elemento genera una caja block pero que se comporta como si fuera inline admitiendo otros elementos en la misma línea; el comportamiento se asemeja al de los elementos img) |
| | Otros que llevan a que el elemento simule el comportamiento de otro (inline-table, table, table-caption, table-cell, table-column, table-column-group, table-footer-group, table-header-group, table-row, table-row-group) |

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS display | |
|---|---|
| | Otros avanzados (flex, inline-flex, grid, inline-grid, run-in) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {display: inline;}<br>.elementoMonter {display: block;} |

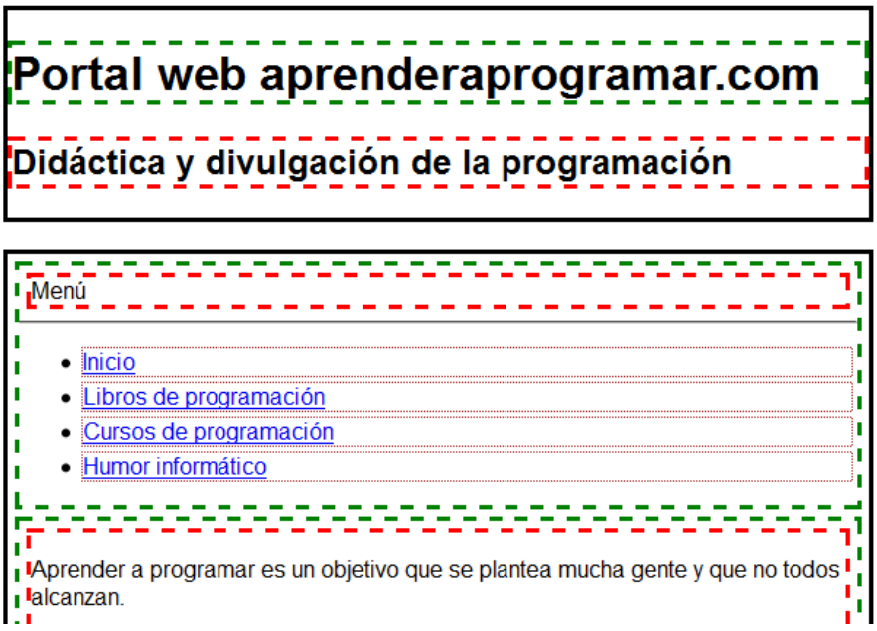
La propiedad display admite numerosos valores, pero los más usados son inline, block e inline-block. Son estos tres valores los que debemos aprender ahora. El resto los iremos conociendo progresivamente a medida que nos puedan resultar necesarios o cuando profundicemos en el conocimiento de CSS. Tener en cuenta también que estas otras propiedades a que nos referimos pueden no ser reconocidas por muchos navegadores.

A continuación veremos algunos ejemplos de uso de las propiedades anteriormente especificadas. Para ello vamos a utilizar el código HTML de base que venimos empleando a lo largo del curso.

Define los siguientes estilos css y visualiza el resultado en tu navegador:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width:600px;}
div {border-style: solid;}
div div {border-style: dashed; border-color: green; margin: 5px;}
div div div {border-style: dashed; border-color: red; margin: 5px;}
h1 {border-style: dashed; border-color: green;}
h2 {border-style: dashed; border-color: red; }
li {border-style: dotted; border-width:thin; border-color:brown; margin: 3px;}
```

Con esto lo único que hemos hecho es poner bordes y márgenes para visualizar las cajas que conforman la página web. El resultado será similar a este:




© aprenderaprogramar.com, 2006-2029

Vemos un div en color verde correspondiente al menú (dentro del cual hay otro div en color rojo con el texto "Menú", una línea generada por el <hr/> y una lista ul con los items de menú).

Accede al código HTML y establece como id de dicho div "menu1".

**EJERCICIO RESUELTO**

Sin ejecutar código, indica cuáles deberían ser los resultados obtenidos al añadir las líneas que se indican en la tabla al archivo CSS. Completa la tabla primero sin ejecutar el código. Luego, compara tu solución con la expuesta en la solución.

| Fragmento de código añadido | Resultado |
|---|---|
| **li {display: inline;}** | |
| **img {display: block;}** | |
| **#menu1 {display:none;}** | |
| **h2 {margin-left:30px; display:list-item;}** | |
| **li {display: inline-block;}** | |
| **ul {display: table; }**<br>**li {display: table-cell; padding:10px;}** | |

© aprenderaprogramar.com, 2006-2029

**SOLUCIÓN**

Prueba a visualizar los resultados en tu navegador al añadir estos fragmentos de código y comprueba que obtienes los mismos resultados que se indican a continuación:

| Fragmento de código añadido | Resultado |
|---|---|
| **li {display: inline;}** | Los elementos de item de menú que son li y que ocupaban cada uno una línea, se ponen en la misma línea. Sólo cuando no caben en una línea continuan en la siguiente. |
| **img {display: block;}** | Las imágenes que estaban en una sola línea unas junto a otras se sitúan cada una en una línea comportándose como elementos block |
| **#menu1 {display:none;}** | El menú (con sus cajas interiores, items de menú, etc.) desaparece completamente y su espacio es ocupado por el texto "Aprender a programar es..." |
| **h2 {margin-left:30px; display:list-item;}** | El título h2 aparece con una viñeta como si fuera un elemento li de una lista |
| **li {display: inline-block;}** | A diferencia de establecer sólo inline, si un elemento salta de línea lo hace completamente por ser un bloque. Su texto no puede cortarse entre líneas como ocurriría si fuera inline. |
| **ul {display: table; }**<br>**li {display: table-cell; padding:10px;}** | Hace que la lista se comporte como una tabla donde cada item es una celda. |

Hay una curiosidad. En la imagen donde vimos las cajas podemos observar que el elemento h2 tiene un espacio libre por arriba y por debajo, que corresponde a un margin-top y margin-bottom automático que aplica el navegador por defecto para este tipo de elementos. Si escribimos h2 {display: inline;} comprobamos no sólo que la caja deja de ocupar todo el ancho disponible sino que los márgenes desparecen ¿Por qué? Porque en los elementos inline no son aplicables (o se ignoran si están establecidas) ciertas propiedades como width, height, margin y float. De este modo, al cambiar el formato de caja no sólo estamos afectando a que los elementos ocupen una línea a todo lo ancho o no, sino también otras propiedades que pueden quedar desactivadas cuando se cambia el tipo de caja usando la propiedad display. Tener en cuenta también que los elementos inline sólo pueden contener otros elementos inline o texto en su interior.

Otra curiosidad. Posiblemente no obtengas nada coherente, pero comprueba qué ocurre si como css utilizas esta única declaración *{font-family: arial; display: inline-block;}

Si te surgen dudas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01034D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**POSICIONAMIENTO FLOTANTE**

Aunque ya conocemos la propiedad position, que nos permite organizar la posición de elementos dentro de una página web, esta propiedad no es lo suficientemente flexible como para lograr diseños atractivos y que se visualicen correctamente en todo tipo de dispositivos. Vamos a estudiar el denominado "posicionamiento flotante", que amplía las posibilidades para organizar los elementos en la web.

El posicionamiento flotante puede resultar un poco desconcertante o confuso cuando se empieza a trabajar con él. Comprender su funcionamiento y posibilidades requiere tiempo y bastantes horas de estudio y ejercicio. Hay que tener paciencia y practicar para llegar a comprenderlo. Sin embargo, a medida que se conoce y se adquiere experiencia resultará fácil trabajar con él y se comprenderán las ventajas que supone, de ahí que sea la forma de posicionamiento más utilizado.

Lo primero que trataremos de abordar es la definición de "flotar". El concepto de posicionamiento flotante se creó inicialmente para lograr que el texto se situara alrededor de las imágenes y no necesariamente debajo de ellas.

Crea un archivo HTML con este contenido:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01034DA.css">
</head>
<body>
<div>
<p>Aquí un párrafo de texto situado antes de la imagen, dentro de un div contenedor</p>
<img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++, un útil editor de texto">
<p>Aquí otro párrafo de texto. CSS es un lenguaje utilizado en la presentación de documentos HTML.
Un documento HTML viene siendo coloquialmente "una página web". Así, podemos decir que el lenguaje
CSS sirve para dotar de presentación y aspecto, de "estilo", a una página web.
</p>
</div>
</body>
</html>
```

Y un archivo de hoja de estilos (nosotros estamos usando el nombre de archivo estilosCU01034DA.css) con este código que nos permite ver el borde de la caja contenedora:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
body {width: 410px; border-style: dotted; }
div {border-style: solid; margin: 20px; padding: 5px; background-color: yellow;}
p {text-align: justify; margin:15px;}
img {margin:10px;}
```

© aprenderaprogramar.com, 2006-2029

Se llama "flotar" un elemento a establecer un comportamiento especial para él definiendo un valor para la propiedad CSS float.

La sintaxis a emplear es del tipo:

```
selectorElemento {float: valorFloat; }
```

La propiedad float sólo admite 3 valores: none (el elemento no flota), right y left.

El valor none significa que el elemento se comportará de la forma habitual dentro del flujo del documento HTML, y es el valor por defecto. Un elemento con float: none; decimos que no es flotante.

El valor right hará que el elemento se desplace hacia la derecha. Se desplazará dentro de su línea de posición y dentro de su contenedor tanto como sea posible, y que los elementos que vienen a continuación se situarán envolviéndolo (en este caso lo rodearán por la izquierda, ya que el elemento estará completamente a la derecha). Los elementos anteriores no cambian su comportamiento.

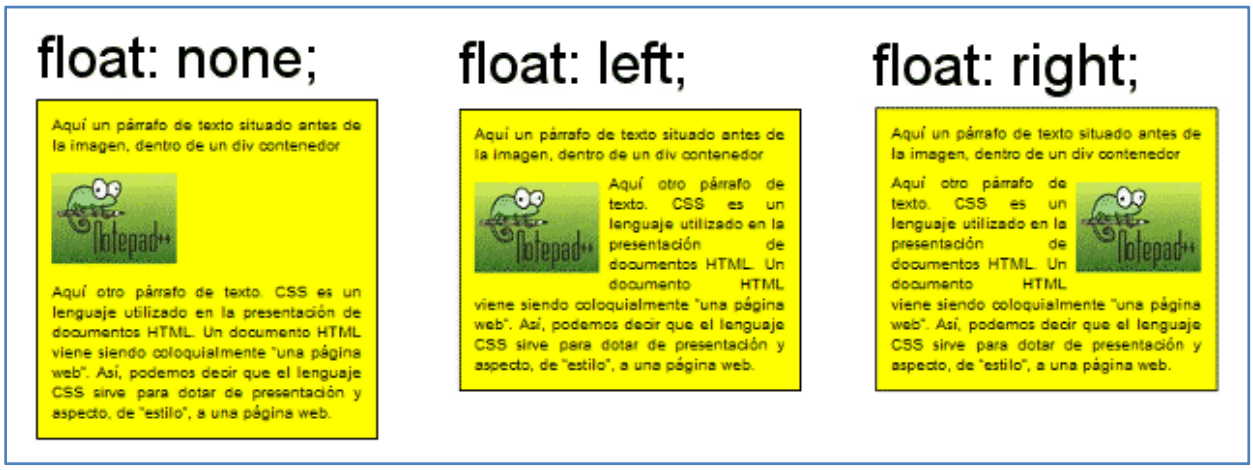
El valor left hará que el elemento se desplace hacia la izquierda. Se desplazará dentro de su línea de posición y dentro de su contenedor tanto como sea posible y que los elementos que vienen a continuación se situarán rodeándolo (en este caso lo rodearán por la derecha, ya que el elemento estará completamente a la izquierda). Los elementos anteriores no cambian su comportamiento.

**La propiedad float no se puede usar para centrar elementos en una página web.** No existe un valor center para esta propiedad. Unicamente sirve para "arrastrar" elementos hacia la derecha o hacia la izquierda, tanto como sea posible, generando un cambio en el flujo del documento HTML.

Ahora visualiza el resultado modificando la regla css del código anterior para elementos img de las siguientes maneras:

1) img {margin:10px; float:none;} /* Elemento img no es flotante */
2) img {margin:10px; float:right;} /* Elemento img es flotante */
3) img {margin:10px; float:left;} /* Elemento img es flotante */

Visualiza los resultados, que deberán ser similares a estos:




© aprenderaprogramar.com, 2006-2029

Comprobamos cómo con float:none; para la imagen, el comportamiento es el normal. Con float: left; el párrafo anterior a la imagen no cambia su comportamiento. Sin embargo, el párrafo posterior se sitúa envolviendo (wrapping) al elemento flotante. Con float: right; ocurre lo mismo, pero en este caso la imagen está lo más a la derecha posible dentro de su contenedor y el texto del párrafo que viene debajo envuelve la imagen por su lado izquierdo.

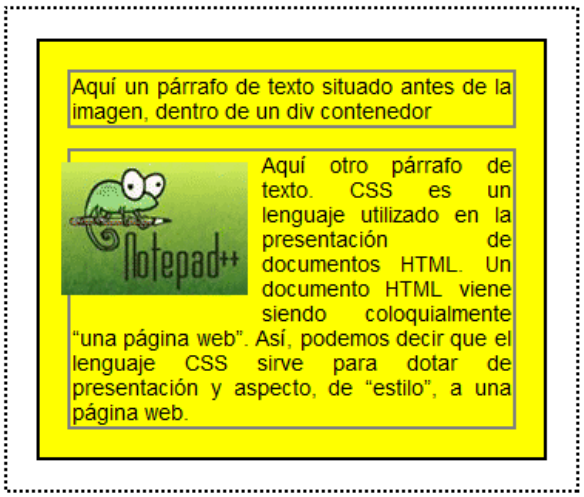
Un aspecto importante es que la propiedad float sólo debe usarse con elementos tipo block, o quizás sería preferible decir sobre elementos que tengan una anchura definida (valor width establecido. Un valor width establecido significa que lo habremos especificado a través de código sobre un elemento block, o que el elemento es un elemento img sin width especificado pero que tiene un valor width "implícito", el ancho de la propia imagen).

Ahora bien, si decimos que sólo debe usarse con elementos tipo block ¿cómo es que hemos aplicado con éxito la propiedad a un elemento img que es un elemento inline? La explicación es que cuando se aplica la propiedad float sobre un elemento, éste automáticamente pasa a comportarse como una caja tipo block. Dicho esto podríamos pensar que también podríamos aplicarle la propiedad float a un párrafo en el ejemplo anterior. La respuesta es: no. No podemos, o al menos no debemos, porque los párrafos del ejemplo anterior aún siendo block no tienen un ancho especificado. **La propiedad float sólo debe aplicarse a elementos con un ancho (width) definido explícita o implícitamente.** Si se aplica la propiedad float a un elemento sin ancho especificado podemos tener resultados impredecibles, o diferentes según el navegador empleado. Por tanto no debemos hacerlo.

### CAJAS ENVOLVENTES EN TORNO A UN ELEMENTO FLOTANTE

Hemos comprobado cómo un elemento como un párrafo envuelve a un elemento flotante como una imagen. La pregunta que nos planteamos ahora es: ¿cuál es la caja del párrafo envolvente? Lo podemos visualizar si en el anterior código CSS usamos estas reglas:

```
p {text-align: justify; margin:15px; border-style:solid; border-color:grey;}
img {margin:10px; float:left;}
```




En la imagen vemos la respuesta: el elemento flotante ha salido del flujo normal del documento. La caja del elemento envolvente tiene su forma normal y se encuentra por debajo del elemento flotante.

**Los bordes, imágenes de fondo y colores de fondo de los elementos envolventes se sitúan debajo de los elementos flotantes a los que envuelven.**

© aprenderaprogramar.com, 2006-2029

### ¿CUÁNTOS ELEMENTOS ENVUELVEN A UN ELEMENTO FLOTANTE?

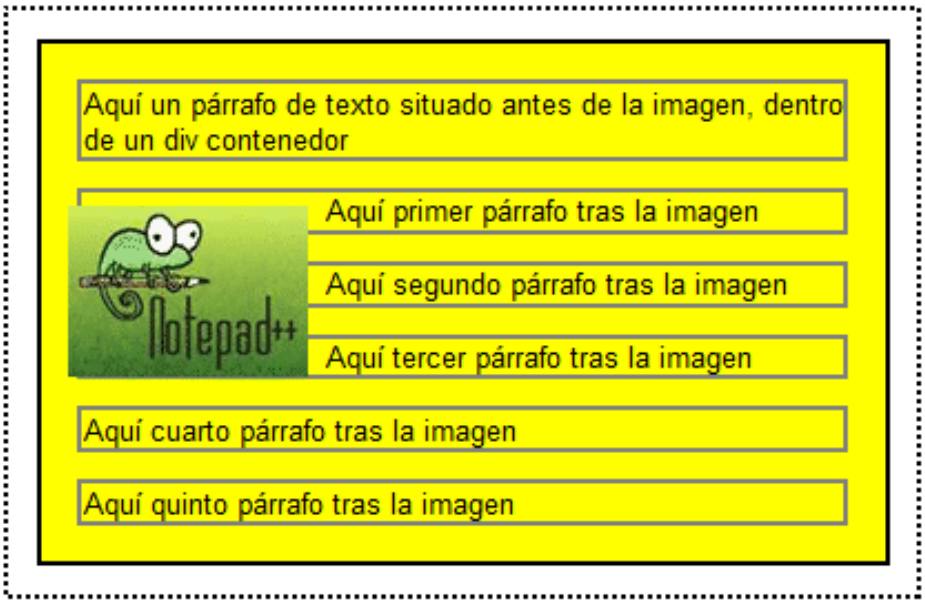
En el ejemplo que venimos viendo un párrafo envuelve a la imagen que ha tomado posicionamiento flotante. ¿Pero qué ocurriría si tenemos varios párrafos de pequeño tamaño? ¿El primero envuelve y los demás siguen su flujo normal o todos los siguientes elementos van envolviendo?

Vamos a modificar el fragmento de código HTML de la etiqueta body así:

```
<body>
<div>
<p>Aquí un párrafo de texto situado antes de la imagen, dentro de un div contenedor</p>
<img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++, un útil editor de texto">
<p>Aquí primer párrafo tras la imagen</p> <p>Aquí segundo párrafo tras la imagen</p>
<p>Aquí tercer párrafo tras la imagen</p> <p>Aquí cuarto párrafo tras la imagen</p>
<p>Aquí quinto párrafo tras la imagen</p>
</div>
</body>
```

Y el código CSS lo dejaremos así:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
body {width: 510px; border-style: dotted; }
div {border-style: solid; margin: 15px; padding: 5px; background-color: yellow;}
p {text-align: justify; margin:15px; border-style:solid; border-color:grey;}
img {margin:10px; float:left;}
```




Al comprobar el resultado, vemos que todos los elementos a continuación de un elemento flotante proceden a envolverlo por un lateral mientras haya espacio para ello. Podemos hacer que los elementos debajo de un elemento flotante dejen de envolverlo a partir de un elemento dado usando la propiedad clear, que explicaremos a continuación.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Para cada una de las siguientes afirmaciones indica si es verdadera o falsa, y justifica brevemente tu respuesta:

a) La propiedad float puede tomar cuatro valores: top, right, bottom y left.

b) float es una propiedad que nos permite maquetar páginas web (documentos HTML).

c) Usando float: center; podemos centrar el contenido de un elemento respecto de su caja contenedora.

d) Un elemento flotante sale del flujo normal de posicionamiento de elementos en una página web.

e) Todos los elementos a continuación de un elemento flotante lo envuelven, a no ser que especifiquemos lo contrario usando la propiedad clear.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01035D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

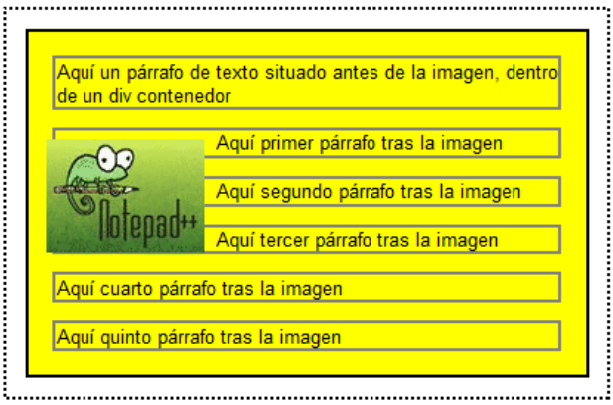
## POSICIONAMIENTO FLOTANTE

Hay varias cuestiones sobre el posicionamiento flotante que aún debemos resolver. Por ejemplo, veremos cómo usando la propiedad clear podemos definir que un elemento y aquellos que vengan después de este deben dejar de envolver al elemento flotante. También veremos qué ocurre cuando varios elementos consecutivos se flotan.

Recordaremos en primer lugar la propiedad float:

| PROPIEDAD CSS float | |
|---|---|
| **Función de la propiedad** | Permite establecer un comportamiento especial para un elemento, que es desplazado tan a la derecha o izquierda como sea posible y admite ser envuelto por otros elementos. |
| **Valor por defecto** | none |
| **Aplicable a** | Elementos con un ancho, especificado o implícito como en imágenes. |
| **Valores posibles para esta propiedad** | **none** (el elemento tendrá comportamiento normal) |
| | **right** (el elemento se desplaza a la derecha tanto como es posible y admite ser envuelto). |
| | **left** (el elemento se desplaza a la izquierda tanto como es posible y admite ser envuelto). |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {float: left;}<br>.elementoMonter {float: right;} |

En la entrega anterior vimos cómo si a un elemento como una imagen se le aplicaba un valor de float distinto de none, los elementos a continuación de él (en nuestro ejemplo párrafos) lo envolvían.




© aprenderaprogramar.com, 2006-2029

**PROPIEDAD CLEAR**

La propiedad clear indica si un elemento y los que le siguen pueden envolver a un elemento flotante precedente y cómo (por la izquierda, por la derecha o por ninguno de los dos lados).

La sintaxis a emplear es del tipo:

```
selectorElemento {clear: valorAsignado; }
```

| PROPIEDAD CSS clear | |
|---|---|
| **Función de la propiedad** | Permite establecer si un elemento y los que le siguen debe envolver (wrap) a un elemento flotante precedente y cómo. |
| **Valor por defecto** | none |
| **Aplicable a** | Elementos tipo block. |
| **Valores posibles para esta propiedad** | **none** (el elemento y los que le siguen envolverá al elemento flotante anterior) |
| | **left** (el elemento y los que le siguen no envolverán a un elemento flotante anterior cuya propiedad float sea left; no habrá elementos a la derecha del flotante). |
| | **right** (el elemento y los que le siguen no envolverán a un elemento flotante anterior cuya propiedad float sea right; no habrá elementos a la izquierda del flotante). |
| | **both** (el elemento y los que le siguen no envolverán a un elemento flotante anterior por ninguno de los dos lados; reestablece el flujo "normal") |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {clear: left;}<br>.elementoMonter {float: right;} |

Usaremos este código para probar la propiedad:

```
<html>
<head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01035DA.css">
</head>
<body> <div>
<p>Aquí un párrafo de texto situado antes de la imagen, dentro de un div contenedor</p>
<img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++, un útil editor de texto">
<p>Aquí primer párrafo tras la imagen</p> <p>Aquí segundo párrafo tras la imagen</p>
<p id="tercero">Aquí tercer párrafo tras la imagen</p>
<p>Aquí cuarto párrafo tras la imagen</p> <p>Aquí quinto párrafo tras la imagen</p>
</div> </body> </html>
```

Y el código CSS lo dejaremos así (pon el nombre adecuado al archivo css):

© aprenderaprogramar.com, 2006-2029

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
body {width: 510px; border-style: dotted; }
div {border-style: solid; margin: 15px; padding: 5px; background-color: yellow;}
p {text-align: justify; margin:15px; border-style:solid; border-color:grey;}
img {margin:10px; float:left;}
```

Ahora probaremos a añadir estas reglas y comprobar el resultado:

a) #tercero {clear: both;} : comprobaremos cómo el párrado con id="tercero" que corresponde al párrafo cuyo texto es "Aquí tercer párrafo..." ya no flota a la derecha de la imagen, sino que se sitúa debajo de ella restableciendo el comportamiento normal.
b) #tercero {clear: right;} : comprobaremos cómo el párrafo con id="tercero" se mantiene flotando en torno a la imagen. ¿Por qué? Porque clear: right; da lugar a que los elementos no floten a la izquierda de un elemento con valor de float right. Como en este caso no tenemos un elemento con valor de float right, este código no genera ningún efecto.
c) #tercero {clear: left; } : comprobaremos cómo el párrado con id="tercero" que corresponde al párrafo cuyo texto es "Aquí tercer párrafo..." ya no flota a la derecha de la imagen, sino que se sitúa debajo de ella. ¿Por qué? Porque clear: left; da lugar a que un elemento y los que le siguen dejen de envolver a un elemento anterior cuyo valor de float sea left, como es el caso que nos ocupa.

Es fácil intuir que clear es una propiedad que se usa en combinación con la propiedad float para crear diseños atractivos.

**¿QUÉ ENVUELVE A UN ELEMENTO FLOTANTE Y CUÁNDO LO ENVUELVE?**

El posicionamiento flotante tiene diversas peculiaridades que debemos ir conociendo poco a poco.

Vamos a emplear este código HTML:

```
<html> <head>
<title>Portal web - aprenderaprogramar.com</title>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01035DB.css">
</head>
<body>
<div>
<p>Un texto antes de las cajas</p>
<div id="caja1">Caja 1</div>
<div id="caja2">Caja 2</div>
<div id="caja3">Caja 3</div>
<div id ="caja4">Caja 4</div>
<p>Un texto después de las cajas</p>
<div class="limpiador"></div>
</div> </body>
</html>
```

Y el siguiente código css de partida (ponle el nombre de archivo adecuado):

© aprenderaprogramar.com, 2006-2029

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width:410px; border-style: dotted;}
div {border-style: solid; margin:7px; background-color: #FFEFD5;  }
div div {padding: 5px; width: 60px;}
p {margin:5px;}
.limpiador{padding:0; border-style:none;}
#caja1{ border-color:red; } #caja2{ border-color:blue;} #caja3{ border-color:green;} #caja4{border-color:orange;}
```

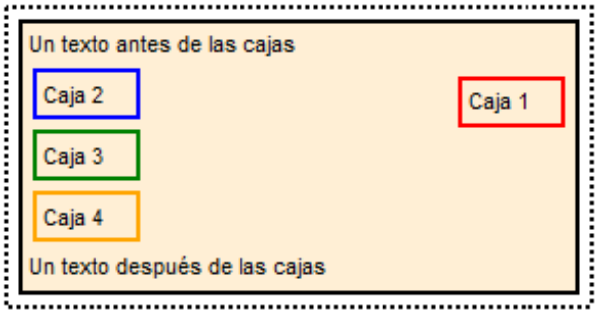
La situación de partida es esta:

Ahora vamos a realizar este cambio en el código CSS: #caja1{ border-color:red; float:right; }

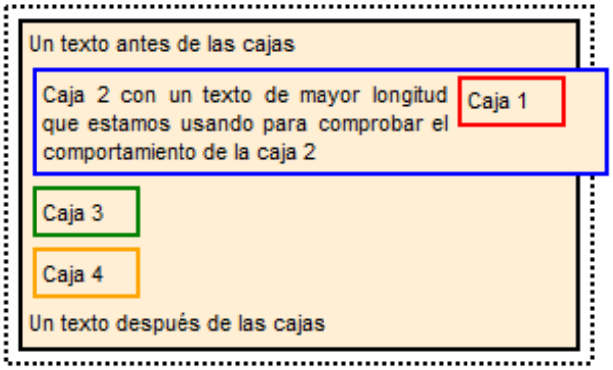
Al establecer el valor de float right comprobamos que la caja 1 se va hacia la derecha tanto como puede saliendo del flujo normal del documento. La caja 2 se sitúa en la línea en que se encontraba la caja 1, ocupando la posición "natural" que le correspondería si no existiera la caja 1, de hecho pasa algo similar a esto porque la caja 1 ha salido del flujo normal y la caja 2 "ignora a la caja 1". Ahora bien, la caja 2 no se ve que envuelva a la caja 1 ¿Por qué? En realidad la caja 2 sí es envolvente de la caja 1, lo que ocurre es que no hay texto para apreciar el efecto. Alarga el texto correspondiente al div de la caja 2 para comprobar el efecto. ¿Qué ocurre? Que el texto se alarga hacia abajo tratándose de adaptar al width definido para la caja, con lo cual tampoco comprobamos que haya comportamiento envolvente. Modifica así el código: #caja2{ border-color:blue; width:390px;}

Ahora sí podemos comprobar que el texto de la caja 2 que llega a confluir con la caja 1 tiene un comportamiento envolvente.

¿La caja 2 no envuelve a la caja 1?




Sí, pero para verlo hace falta texto y anchura




© aprenderaprogramar.com, 2006-2029

Conclusión: el comportamiento envolvente no se aprecia si no existe texto y anchura suficiente para que se pueda apreciar. En caso de no existir texto y anchura suficiente, el documento se muestra como si el flujo normal continuara ignorando al elemento flotante.

A la pregunta ¿Qué envuelve a un elemento flotante? Podemos responder que **lo que envuelve a un elemento flotante es el texto** de los elementos que vengan a continuación de él. Repetimos: lo que envuelve es el texto, no envuelven las cajas ni imágenes ni otro tipo de elementos, sólo texto.

¿Cuándo envuelve el texto de un elemento posterior a un elemento flotante? Cuando existe ancho (width) suficiente para que el siguiente elemento alcance el borde del elemento flotante y cuando el texto tiene longitud suficiente como para poder apreciar el efecto.

**¿QUÉ OCURRE SI NO HAY ESPACIO SUFIENTE PARA ENVOLVER?**

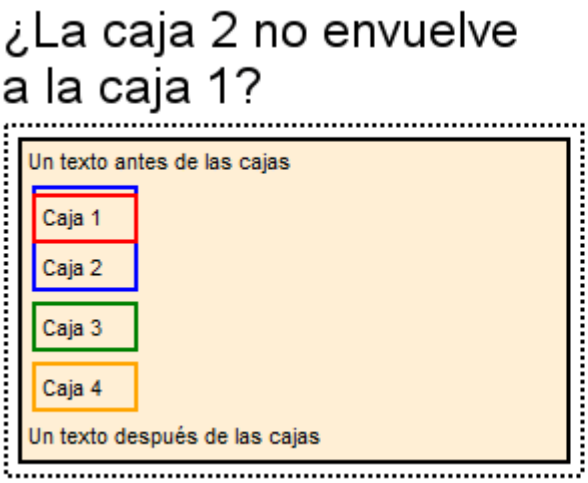
Sobre el código inicial que habíamos planteado, vamos ahora a realizar este cambio en el código CSS: #caja1{ border-color:red; float:left; }

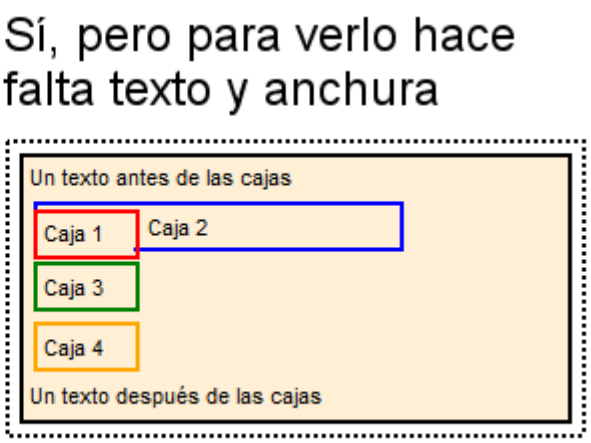
Observaremos algo extraño: la caja 2 parece que no envuelve a la caja 1. El texto de la caja 2 aparece debajo del texto de la caja 1. ¿Qué ocurre? Que el texto de la caja 2 para poder situarse a la derecha del texto de la caja 1 necesita disponer de un ancho suficiente. Si no dispone de ancho, pasa a la siguiente línea (agrandando el tamaño de su caja). Basta darle el ancho suficiente para que se vea cómo el texto de la caja 2 envuelve al texto de la caja 1 por su derecha.

Modificaremos el código así: #caja2{ border-color:blue; width:250px;}

Con ello comprobamos cómo el texto de la caja 2 envuelve a la caja 1.

Muchas veces un programador o diseñador web nos ha planteado por qué el texto no envuelve a una imagen HTML u otro tipo de elemento. Aquí está la respuesta. **Cuando no hay espacio suficiente para envolver el texto se sitúa debajo del elemento flotante.**



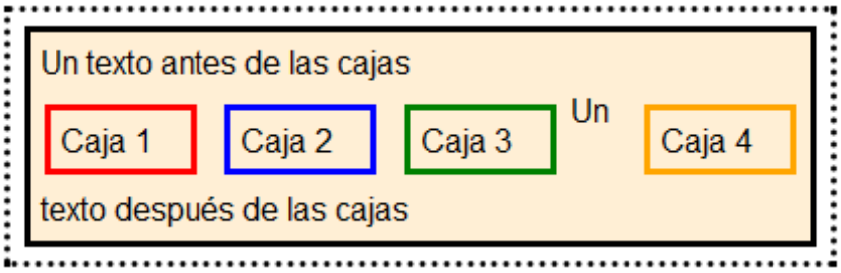




© aprenderaprogramar.com, 2006-2029

**¿QUÉ OCURRE SI APLICAMOS FLOAT A VARIOS ELEMENTOS CONSECUTIVOS?**

Si aplicamos float a varios elementos consecutivos, los elementos se van arrimando unos a otros mientras exista espacio suficiente. Cuando ya no queda espacio, pasarán a la siguiente línea.

Modifica el código inicial que hemos usado así: #caja1{ border-color:red; float:left; } #caja2{ border-color:blue; float: left;} #caja3{ border-color:green; float: left;} #caja4{border-color:orange; float: right;}

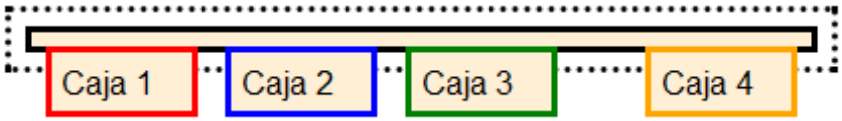



El resultado lo interpretamos de la siguiente manera. La caja 1 se sitúa tan a la izquierda como le es posible, es decir, pegada al límite del contenedor. La caja 2 trata de situarse tan a la izquierda como le es posible, pero al encontrar un elemento flotante ya posicionado no puede pegarse al borde del contenedor, sino que se alinea junto al elemento flotante que se ha encontrado. La caja 3 se comporta igual. La caja 4, en cambio, se sitúa tan a la derecha como le es posible.

El párrafo "Un texto después de las cajas" trata de envolver al último elemento flotante (la caja 4). Su texto comienza en la vertical línea que sería su posición natural si ninguna de las 4 cajas existieran (ya que han salido del flujo normal) y en la horizontal comienza desde que un elemento float deja espacio suficiente para poder insertarse. Trata de avanzar para envolver a la caja 4, pero al no haber espacio disponible salta a la siguiente línea.

Elimina ahora los párrafos "Un texto antes de las cajas" y "Un texto después de las cajas" y comprueba el resultado (mantiendo el posicionamiento flotante de las cuatro cajas, tres a la izquierda y la cuarta a la derecha).

Te podrás encontrar con algo así:

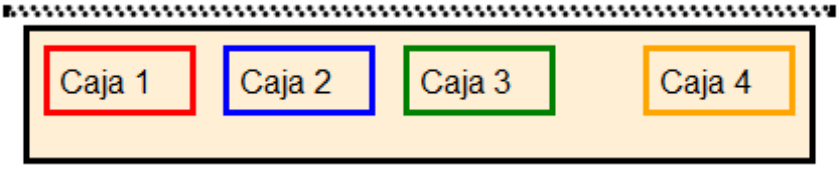



¿Qué está ocurriendo? Que al salir del flujo normal los elementos flotantes dejan de ocupar espacio en su caja contenedora de forma que el valor height de la caja contenedora puede llegar a hacerse nulo. Esto en alguna ocasión puede que ni siquiera lo notemos. No obstante, en algunos casos como aquellos donde tenemos un borde, color de fondo o imagen de fondo para la caja contenedora puede ser un problema.

¿Cómo evitar que la caja contenedora no incluya a las cajas flotantes hijas y hacer que se dibujen sus fondos correctamente englobando a los elementos flotantes? Hay varias maneras para hacer esto.

Una forma de hacerlo es convertir en flotante también a la caja madre. Una caja flotante amplía su tamaño para incluir a sus cajas hijas, de forma que declarando a la madre como flotante ya se ampliará automáticamente. Si hacemos esta modificación: div {border-style: solid; margin:7px; background-color: #FFEFD5; float:left; } el resultado será similar a este:

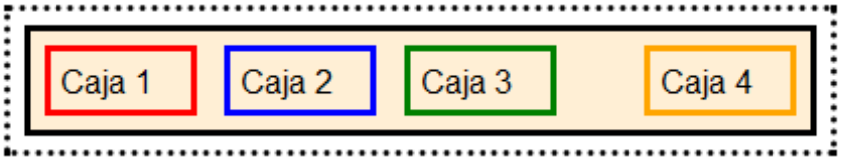
© aprenderaprogramar.com, 2006-2029




Vemos cómo el elemento body aparece "comprimido" sin altura en la parte superior, porque al ser tanto la caja madre como las hijas flotantes, están fuera del flujo del documento y el elemento body no adquiere altura. Tener en cuenta que en este caso estamos viendo el elemento body porque hemos aplicado la propiedad border-style:dotted; pero esto lo hacemos en el marco de un curso de aprendizaje, en la práctica profesional el elemento body no lleva borde prácticamente nunca.

Ahora veamos otra alternativa distinta a convertir en flotante a la caja madre. Vamos a dejar la caja madre como div {border-style: solid; margin:7px; background-color: #FFEFD5;} y por el contrario vamos a hacer esta modificación: .limpiador{padding:0; border-style:none; clear:both; }

El resultado será similar a este:




¿Por qué ocurre esto? Porque al incluir un elemento, en este caso un div, con valor clear:both; que indica que debe mostrarse sin flotar, la caja se comporta como si debajo de los elementos flotantes estuviera el elemento con flujo normal, de modo que se muestra conteniendo tanto a los elementos flotantes como a este elemento. Si el elemento "limpiador" está vacío realmente no se muestra, pero como el navegador interpreta que hay algo, adapta las dimensiones de la caja.

Hay más formas de lograr un resultado similar a este (por ejemplo aplicando la propiedad overflow: hidden; a la caja madre), pero no vamos a entrar a explicarlas ahora.

**EJERCICIO**

Define un documento HTML donde a través del uso de las propiedades float y clear y de las anteriores propiedades que hemos visto a lo largo del curso crees un diseño con este aspecto:

a) En primer lugar se deben mostrar 8 cajas div de 50x50 píxeles, con margin-right de 5 píxeles para cada una de ellas, y cada una de ellas con distinto color de fondo, alineadas en horizontal hacia la izquierda gracias al uso de float left.

b) En segundo lugar se debe mostrar un div con un texto y color de fondo amarillo, con margen superior e inferior de 20 píxeles, abarcando todo el ancho disponible.

c) En tercer lugar se deben mostrar 3 cajas div de 200x50 píxeles, con margin-right de 5 píxeles para cada una de ellas, y cada una de ellas con distinto color de fondo, alineadas en horizontal hacia la derecha gracias al uso de float right.

© aprenderaprogramar.com, 2006-2029

d) En cuarto lugar un div de fondo de color #DEB887 que ocupe todo el ancho disponible y contenga el texto <<Curso CSS aprenderaprogramar.com>>

Nota: si es necesario, usa los div contenedores auxiliares que te sean necesarios.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01036D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## DISEÑO EN COLUMNAS

Uno de los usos habituales de float y clear es crear diseños donde las cajas tienen distintos tamaños y se crean una cabecera de página web, una parte central, una o varias columnas laterales y un espacio en la parte inferior denominado pie o footer. Veamos cómo podemos generar este tipo de diseños usando CSS.

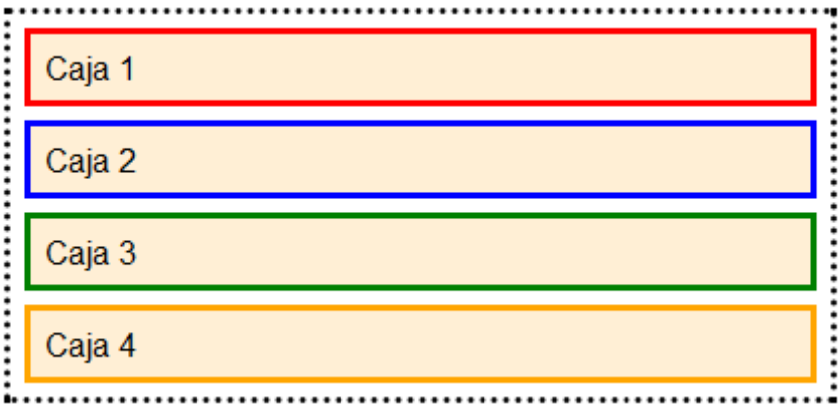
Vamos a partir de este código HTML:

```
<html> <head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01036DA.css">
</head>
<body>
<div id="caja1">Caja 1</div> <div id="caja2">Caja 2</div>
<div id="caja3">Caja 3</div> <div id ="caja4">Caja 4</div>
</body>
</html>
```

Y el siguiente código css de partida (ponle el nombre de archivo adecuado):

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width:410px; border-style: dotted; border-width: 2px;}
div {border-style: solid; border-width:2px; margin:7px; padding:7px; background-color: #FFEFD5; }
#caja1{ border-color:red;}
#caja2{ border-color:blue; }
#caja3{ border-color:green;}
#caja4{border-color:orange;}
```

La situación de partida es que simplemente tenemos cuatro cajas (dentro de las cuales se supone que vamos a organizar los diferentes contenidos de la página web):




© aprenderaprogramar.com, 2006-2029

**ANCHURA Y ALTURA ESPECIFICADA Y TOTAL DE UN ELEMENTO ¿WIDTH NO FUNCIONA?**

¿Cuál es el valor de la propiedad width de la caja 1? Viendo el dibujo y el código podríamos pensar que el cálculo para obtener este ancho sería: 410 px totales – 7px * 2 lados = 396 píxeles

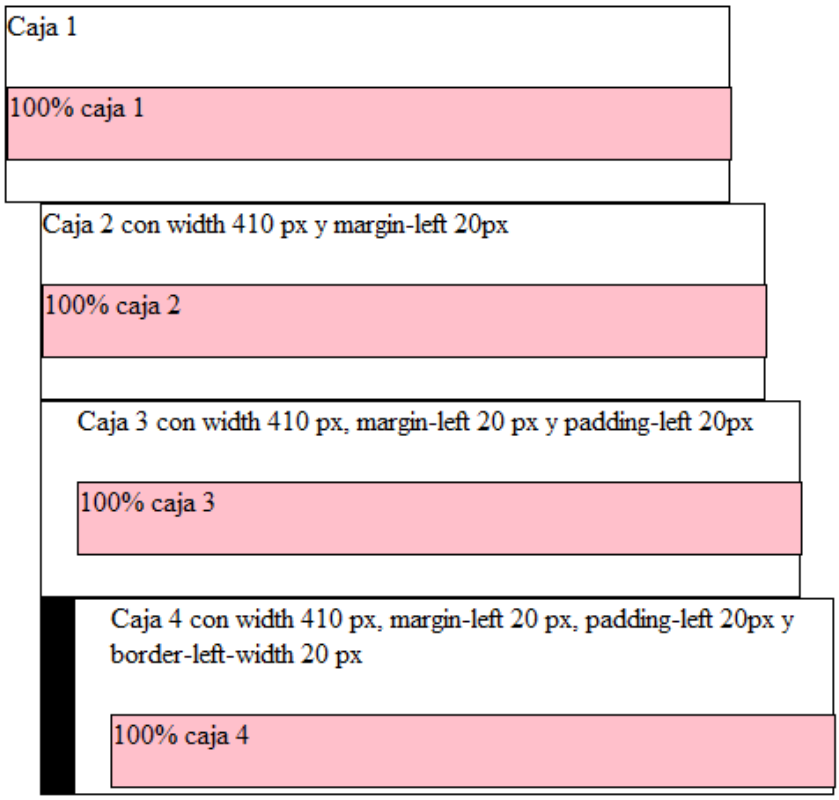
Sin embargo, el valor de la propiedad width se refiere al ancho del espacio que ocupa el contenido del elemento excluido el margen, borde y padding. Esto nos lleva a que el width de la caja 1 se calcule como 410 px totales – 7 px * 2 márgenes – 7 px * 2 paddings – 2px * 2 bordes = 378 píxeles ¿Por qué es esto así?

Podemos pensar en distintos tipos de anchura (nos referiremos a anchura, aunque para altura es igualmente válido lo que expondremos). La anchura "declarada" es la anchura que especificamos al dar un valor a la propiedad width. La anchura "aparente en pantalla" es la anchura que aparenta tener el elemento en la pantalla (hasta el borde visible, si existe). La anchura total asociada al elemento se obtiene como **margin + border + padding + width** (en horizontal para widht, o de la misma forma en vertical para height)

Al declarar un valor width para un elemento hijo en porcentajes, se toma como valor de referencia el valor de la propiedad width de su elemento padre. Esto puede generar efectos extraños y hace que mucha gente piensa que "width en porcentajes no funciona". En realidad sí funciona, pero hay que saber cómo. Escribe este código y visualiza los resultados:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{margin:0; padding:0;}
div{ border-style:solid; border-width:1px; height: 115px;}
#caja1{width:410px; }
#caja2{width:410px;  margin-left:20px;}
#caja3{width:410px;  margin-left:20px; padding-left:20px;}
#caja4{width:410px;  margin-left:20px; padding-left:20px; border-left-width:20px;}
.interior {margin-top: 25px; width:100%; height:40px; background-color:pink;}
</style>
</head>
<body>
<div id="caja1">Caja 1 <div class="interior">100% caja 1</div></div>
<div id="caja2">Caja 2 con width 410 px y margin-left 20px<div class="interior">100% caja 2</div></div>
<div id="caja3">Caja 3 con width 410 px, margin-left 20 px y padding-left 20px<div class="interior">100% caja 3</div></div>
<div id ="caja4">Caja 4 con width 410 px, margin-left 20 px,
 padding-left 20px y border-left-width 20 px<div class="interior">100% caja 4</div></div>
</body>
</html>
```

© aprenderaprogramar.com, 2006-2029




Observamos algunas cosas "extrañas":

a) En la caja 1 vemos cómo la caja interior en color rosado sobresale "ligeramente" por el lado derecho ¿Por qué? Esta caja rosada interior se posiciona justo en la esquina interior del borde superior izquierdo de la caja contenedora (caja 1). Luego es desplazada hacia abajo 25px debido a la propiedad margin-top que tiene establecida. A continuación se dibuja ocupando un espacio de 1px como ancho del borde izquierdo + 410 px de contenido + 1 px de borde derecho. El ancho visible en pantalla (delimitado por los bordes) es de 412 px, igual que el de su caja padre, pero está ligeramente desplazada hacia la derecha, de ahí que sobresalga ligeramente por la derecha.

b) En la caja 3 vemos que la caja interior con un ancho del 100% no toma las dimensiones de la caja 3, sino que es más pequeña. ¿Por qué? Porque el borde que vemos de la caja 3 incluye el padding aplicado a la caja 3 de 20 px de modo que la caja 3 "se ve en la pantalla" con un ancho de (410 px de contenido + 20 px de padding-left + 2 px de bordes) = 432 px. Sin embargo la caja interior tiene el 100 % del valor width de su caja contenedora, es decir, 410 px, más 2 px de borde, con lo que se ve con un ancho de 412 px.

En resumen, hay que tener siempre presente que los valores width y height son los relativos al contenido de un elemento excluido el padding, borde y margen. El ancho apreciable en pantalla de un elemento puede ser superior al establecido con width debido al padding y al borde. El ancho total asociado a un elemento puede ser superior al establecido con width debido al padding, al borde y al margin.

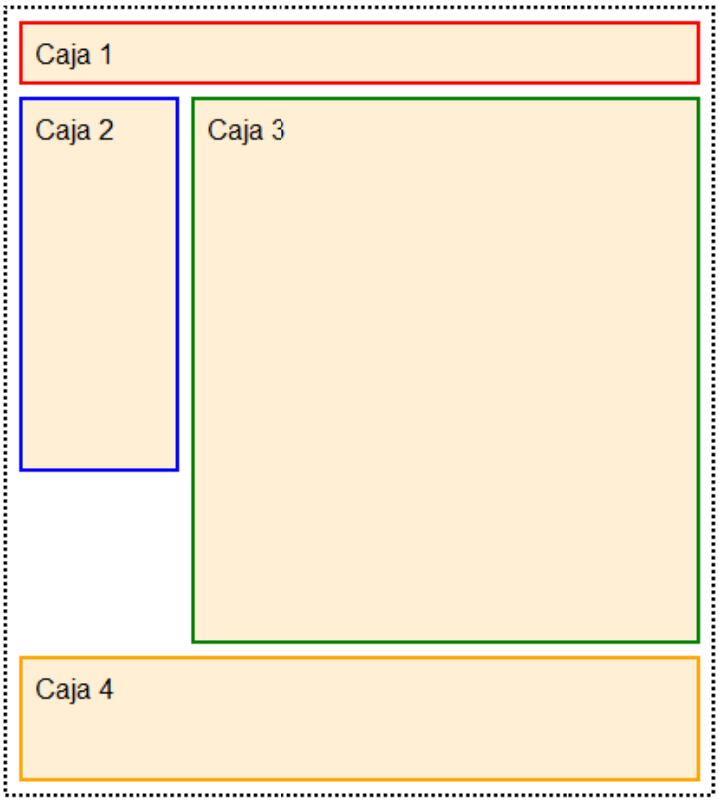
© aprenderaprogramar.com, 2006-2029

**CREAR UN DISEÑO**

Ahora vamos a utilizar el código CSS para crear un diseño con cabecera, una columna lateral izquierda, un espacio principal y un pie de página o footer.

Escribe el siguiente código CSS y comprueba los resultados sobre el código HTML de las 4 cajas:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width:410px; border-style: dotted; border-width: 2px;}
div {border-style: solid; border-width:2px;
margin:7px; padding:7px;
 background-color: #FFEFD5; }
/*La anchura del contenido de la caja 1 es:
 410 - 14 de margin - 4 de border - 14 de padding, resulta 378 px*/
#caja1{ border-color:red;}
#caja2{ margin:0 7px 7px 7px; border-color:blue;
 width:75px; float:left; height:200px; }
/*Ancho total 410 px - 8px bordes - 21 px de margin - 28 de padding -75px contenido caja 2
 obtenemos 278 px*/
#caja3{ margin:0 7px 7px 0px; border-color:green;
 width:278px; float: left; height: 300px;}
#caja4{border-color:orange; clear:both; height:55px;}
```




Hemos aplicado unos valores de height para poder visualizar el diseño. En una web real height puede quedar sin especificar, de modo que las cajas irán agrandando su tamaño en vertical en función de la cantidad de contenido que alberguen.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO RESUELTO**

Escribir el código CSS para crear un diseño similar al anterior pero con cabecera, una columna lateral izquierda, un espacio principal y una columna lateral derecha.

**SOLUCIÓN**

Añadiremos esta línea como última línea del código HTML, antes del cierre de body, para restablecer el flujo normal (realmente no tenemos elementos debajo y no sería necesario hacerlo, pero podríamos tenerlos):

```
<div class="limpiador"></div>
```

El código CSS sería:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width:410px; border-style: dotted; border-width: 2px;}
div {border-style: solid; border-width:2px;
margin:7px; padding:7px;
 background-color: #FFEFD5; }
/* La anchura del contenido de la caja 1 es:
 410 - 14 de margin - 4 de border - 14 de padding, resulta 378 px*/
#caja1{ border-color:red;}
#caja2{ margin:0 7px 7px 7px; border-color:blue;
 width:65px; float:left; height:200px; }
/* Ancho total 410 px - 12px bordes - 28 px de margin - 42 de padding -65px contenido caja 2
 -65 px contenido caja 4 obtenemos 198 px*/
#caja3{ margin:0 7px 7px 0px; border-color:green;
 width:198px; float: left; height: 300px;}
#caja4{ margin:0; border-color:orange; float: left; width:65px; height:200px;}
.limpiador {clear:both; padding:0; border-style:none; }
```

Comprueba los resultados en tu navegador.

Ten en cuenta que este es un posible código para obtener un diseño como el propuesto. Quizás a ti se te haya ocurrido otro código y será igualmente válido (o incluso mejor que este). Por ejemplo, no hay por qué usar px para fijar las dimensiones, podríamos usar % ó em u otras unidades. Ten en cuenta que CSS y el diseño web no son matemáticas. Lo importante es obtener el resultado deseado y generar un código lo mejor posible, pero no el "código perfecto" porque este no existe (o requeriría demasiado tiempo trabajar para conseguirlo, sin obtener ventajas a cambio).

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Define un documento HTML donde a través del uso de la propiedad float y de las anteriores propiedades que hemos visto a lo largo del curso crees un diseño con este aspecto:

<table>
<tr><td colspan="3">Bienvenidos a aprenderaprogramar.com (color de fondo: #DEB887)</td></tr>
<tr><td rowspan="2">Menú<br><br>-Cursos<br><br>-Humor<br><br>-Divulgación</td><td>Conoce las últimas novedades del lenguaje JavaScript. (color de fondo: #ADD8E6)</td><td rowspan="2">Espacio reservado para publicidad</td></tr>
<tr><td>Artículo sobre Gimp, un programa de software libre para el diseño gráfico. (color de fondo: #90EE90)</td></tr>
<tr><td colspan="2">Contacta con nosotros (color de fondo: #DDA0DD)</td><td>Aviso legal</td></tr>
</table>

Nota: los anchos de los elementos serán del 100 % disponible si es todo el ancho (por ejemplo <<Bienvenidos>>), del 25 % si es una columna simple (por ejemplo <<Menú>>) y del 50 % si es una columna doble (por ejemplo <<Artículo sobre Gimp>>).

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01037D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## EFICIENCIA Y VELOCIDAD DE CARGA

Una página web compleja puede tener que cargar decenas o cientos de imágenes, tanto imágenes de fondo como imágenes de contenido. Hemos visto que las imágenes de fondo se manejan con las propiedades tipo background y que entre estas propiedades tenemos algunas como background-position que nos permiten establecer dónde debe situarse la imagen.

Uno de los problemas con las que se enfrentan las páginas web es que son cada vez más complejas y con frecuencia incluyen un gran número de imágenes. Cargar una imagen conlleva una petición al servidor, una respuesta del servidor, carga en el navegador, etc. e incluso si las imágenes son de pequeño tamaño todas estas peticiones consumen tiempo y hacen que la página cargue con lentitud.

Gracias a las facilidades que ofrece CSS para el manejo de imágenes de fondo podemos mejorar la eficiencia del procesado de imágenes y la velocidad de carga de la página.

## CONCEPTO DE SPRITE CSS

En programación el concepto de sprite puede tener distintos significados. En CSS nos referimos a sprite para hacer alusión a un conjunto de imágenes diferentes agrupadas todas ellas en una misma imagen. Por ejemplo:

Esto que vemos es un ejemplo típico: diferentes iconos se agrupan en una imagen siguiendo una "cuadrícula". La cuadrícula es un elemento que usan los diseñadores gráficos para saber que cada icono ocupa exactamente el mismo espacio (por ejemplo 64 pixeles de ancho y 64 pixeles de alto), pero una vez terminan de crear el diseño la imagen se suele guardar sin cuadrícula (es decir, no se ven líneas de división entre las distintas imágenes). El fondo tipo tablero de ajedrez en los programas de diseño como Gimp ó Photoshop indica que la imagen es una transparencia, no forma parte de la imagen en sí.

© aprenderaprogramar.com, 2006-2029

En el caso de agrupar iconos en una sola imagen, el número de iconos puede llegar a ser muy grande, en algunos casos pueden ser decenas o cientos.

En la imagen anterior tenemos 7x7 iconos en una sola imagen. Supongamos que en cargar una imagen en la página web se tardara 0,1 segundos. Si cargáramos los 49 iconos por separado requeriría 4.9 segundos mientras que cargando el sprite únicamente necesitamos 0.1 segundos (esto es una simplificación, pero nos vale como ejemplo). Este es el motivo por el que se usan los sprites en diseño web y en CSS.

Hasta ahora hemos visto las imágenes dentro del sprite como imágenes todas iguales, cada uno en un espacio de la cuadrícula, pero esto no tiene por qué ser así. Por ejemplo se puede usar algo de este tipo:

Aquí vemos cómo las distintas imágenes que componen el sprite siguen una cuadrícula, pero cada cuadro es de distintas dimensiones. Por ejemplo, el icono de la estrella tiene distinto tamaño que el icono del pulpo o el icono del escáner.

También puede crearse el sprite considerando que una imagen ocupa varios espacios de la cuadrícula como vemos aquí:

© aprenderaprogramar.com, 2006-2029

En la imagen anterior vemos algunos iconos que ocupan un cuadro completo, otros de menor tamaño que solo ocupan parte de un cuadro y por otro lado una imagen de una cabeza de pájaro de mayores dimensiones que ocupa varios espacios de la cuadrícula.

Un sprite no sólo puede contener iconos, también puede tener otro tipo de imágenes como logos, botones, imágenes que representen bordes o esquinas de contenedores, fondos, etc.

### GENERAR O CREAR LOS SPRITES CSS

Normalmente se parte de las imágenes que componen un sprite por separado para luego reunirlas en el sprite. Esto se puede hacer de varias maneras:

a) Usando un programa de diseño gráfico (como Photoshop, Gimp, etc.).
b) Usando herramientas on-line (sprite generators) que a través de páginas web nos permiten subir las imágenes y nos devuelven el sprite creado.

No vamos a profundizar en la creación de sprites porque no es nuestro objetivo ahora entrar en este tipo de detalles. Con tener el concepto o idea general nos basta.

### CÓMO USAR LOS SPRITES CSS

En un sprite reunimos distintas imágenes que van a aparecer en la página web como si fueran imágenes independientes, aunque realmente sean una sola imagen. ¿Cómo se consigue esto? Tendremos que jugar con el área visible de un elemento y con la posición de la imagen. Con lo que hemos visto en apartados anteriores del curso ya conocemos cómo funcionan los fondos CSS, cómo se determina su área visible y cómo se puede jugar con la posición de las imágenes de fondo. Por ello ya debemos ser capaces de aplicar estos conceptos a la resolución de un ejercicio.

### EJERCICIO RESUELTO

Usando el código HTML de base que venimos empleando a lo largo del curso, y en concreto el fragmento siguiente:

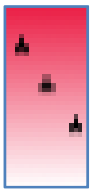
```
<!-- menu -->
<div>
<div>Menú</div>
<hr/>
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
</div>
<!-- fin menu -->
```

© aprenderaprogramar.com, 2006-2029

Se desea realizar lo siguiente:

a) Reemplazar el elemento <hr/> con un elemento que tenga como fondo una imagen de 18x40 pixeles de anchoxalto **que se repita** a lo largo de todo el ancho disponible en la web, creando la apariencia de ser una barra de separación.

La imagen a utilizar debe ser la que ponemos a continuación (o cualquier otra con las dimensiones indicadas):



b) Partiendo de un sprite con cuatro iconos, cada uno de los cuales ocupa 40x40 pixeles, colocar como imagen de fondo en un elemento anexo a cada item de menú cada uno de los iconos. La imagen a utilizar debe ser la que ponemos a continuación (o cualquier otra con las dimensiones indicadas):

Como cada icono tiene 40 pixeles de ancho y 40 de alto, el tamaño total del sprite es de 160 pixeles de ancho por 40 pixeles de alto. Los 160 pixeles de ancho resultan de multiplicar el ancho de 40 pixeles de cada icono por 4.

La idea a utilizar es la siguiente: cada elemento anexo a un item de menú tendrá un área visible de 40x40 pixeles, de modo que al colocar el sprite sólo se vea un icono (los otros quedarán ocultos). En los cuatro elementos anexos a los item de menú colocaremos la misma imagen, pero en distinta posición según el elemento, dando lugar a que en cada caso en el área visible se vea algo distinto. La siguiente imagen refleja la idea:

© aprenderaprogramar.com, 2006-2029

Hemos dejado como transparentados los iconos que no deben aparecer en la web (aquí los mostramos transparentados para que se aprecie la idea de que siempre estamos utilizando la misma imagen, pero colocándola de distinta manera para que en el área visible aparezca justo el trozo de imagen que a nosotros nos interese).

Nota: eliminar las viñetas de la lista que conforma el menú usando esta propiedad: li {list-style-type: none;}

**SOLUCIÓN AL EJERCICIO**

Para el apartado a) modificaremos el código HTML de la siguiente manera:

<!--<hr/>-->

Hemos comentado el hr (con lo cual lo eliminamos), y a cambio hemos introducido un contenedor div sin texto y con id="barra".

En el archivo de estilos css introduciremos el siguiente código:

#barra {width: auto; height: 40px; background-image: url("barra.png")}

Donde barra.png es el nombre de archivo con la imagen, que debe estar localizado en el mismo directorio que el archivo html. También se podría usar una URL externa como https://lh3.googleusercontent.com/-_6GcdriGAEI/UkkjeJoZlOI/AAAAAAAAAb0/5B8NPiEqYsM/s40/CU01037D_5.png

Para el apartado b) crearemos contenedores de división donde alojar las imágenes de fondo.

No podemos usar  para crear los contenedores porque span es un elemento inline y necesitamos aplicar width y height, y hay que recordar lo siguiente: "width y height son sólo aplicables a elementos tipo block y elementos insertados en una posición que son reemplazados por un objeto (entre ellos img, input, textarea, select, object)". Usaremos por tanto elementos div para crear las divisiones.

```
<ul id="itemsMenu">
<li><div></div><a href="#">Inicio</a></li>
<li><div></div> <a href="libros.html" >Libros de programación</a> </li>
<li><div></div> <a href="cursos.html" >Cursos de programación</a> </li>
<li><div></div> <a href="humor.html" >Humor informático</a> </li>
</ul>
```

El código que proponemos como solución es el siguiente (ten en cuenta que tú puedes haber creado otro código diferente. Lo importante es que el código esté bien construido y proporcione los resultados deseados):

© aprenderaprogramar.com, 2006-2029

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; }
#barra {height: 40px; background-image: url("barra.png"); margin-bottom:30px;}
li {list-style-type: none;}
#itemsMenu li {height: 45px; line-height:2em; margin-top: 5px; }
#itemsMenu li div { height: 40px; width:40px; margin-right:15px; background-image: url(cuatroIconos.png); float:left;}

#itemsMenu li:nth-child(1) div {background-position: 0px;}
#itemsMenu li:nth-child(2) div {background-position: 40px;}
#itemsMenu li:nth-child(3) div {background-position: 80px;}
#itemsMenu li:nth-child(4) div {background-position: 120px;}
```

Donde cuatroIconos.png es el nombre de archivo con la imagen, que debe estar localizado en el mismo directorio que el archivo html. También se podría usar una URL externa como https://lh6.googleusercontent.com/-KahySaHLhww/UkkjeX4jh4I/AAAAAAAAAcA/iEIIJs6NvVw/s160/CU01037D_6.png

Con los conocimientos que hemos adquirido a lo largo del curso debemos ser capaces de interpretar este código y sus resultados. Si tienes dudas, consulta en los foros (aprenderaprogramar.com/foros).

El resultado a obtener será similar a este:

Comprobamos cómo podemos usar una sola imagen y crear la apariencia de estar usando cuatro imágenes jugando con el área visible y posición de la imagen.

**Próxima entrega:** CU01038D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## VISIBILITY CSS

En apartados anteriores hemos visto cómo a través de la propiedad display podíamos hacer que un elemento "desapareciera" de una página web estableciendo su valor en "none" (el elemento no se muestra; el efecto es como si no existiera, por lo que su espacio es ocupado por otros elementos). La propiedad visibility permite crear un efecto similar, pero en este caso el elemento permanece ocupando su espacio, aunque no se muestra.

| PROPIEDAD CSS visibility | |
|---|---|
| **Función de la propiedad** | Permite ocultar elementos sin que su espacio sea ocupado por otros y también colapsar filas o columnas de una tabla. |
| **Valor por defecto** | visible |
| **Aplicable a** | Todos los elementos (excepto valor collapse, sólo para filas o columnas de tablas). |
| **Valores posibles para esta propiedad** | **visible** (el elemento es visible) |
| | **hidden** (el elemento es transparente, no es visible). |
| | **collapse** (aplicable a filas o columnas de tablas las colapsa haciendo que sean no visibles). |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {visibility: hidden;}<br>tr.filaColapsada {visibility: collapse;} |

Un aspecto importante a tener en cuenta es que esta propiedad se hereda directamente por parte de todas las cajas hijas de una dada, de modo que ninguna caja hija se mostrará (excepto si tiene establecida la propiedad como visible). Esto tiene cierto interés, pues nos permite ocultar completamente una parte de una página web excepto algunos elementos específicamente señalados.

Nos podemos preguntar: ¿para qué vamos a querer tener en nuestra web un elemento que no se muestra? Pueden existir varios motivos. Uno de ellos, que queramos reservar un espacio que no sea ocupado por ningún otro elemento. Otro motivo puede ser que a través de un lenguaje de programación como PHP, Javascript, etc. queramos alterar el valor de la propiedad visibility para que el elemento en algunos casos se muestre y en otros no.

El valor collapse puede no responder correctamente en algunos navegadores.

Veremos código para aplicar esta propiedad después de explicar la propiedad overflow.

© aprenderaprogramar.com, 2006-2029

## PROPIEDAD OVERFLOW CSS

Una caja dentro del box-model de una página web puede tener unas dimensiones limitadas y el contenido exceder esas dimensiones. A eso se le llama "desbordamiento" (overflow en inglés). La propiedad overflow sirve para controlar cómo debe actuar el navegador en caso de que el contenido (por ejemplo texto) de una caja exceda las dimensiones de la misma.

| PROPIEDAD CSS overflow | |
|---|---|
| **Función de la propiedad** | Permite definir cómo debe comportarse el navegador en caso de que el contenido de una caja exceda su tamaño. |
| **Valor por defecto** | visible |
| **Aplicable a** | Elementos tipo block |
| **Valores posibles para esta propiedad** | **visible** (el contenido no se corta, se sale de su contenedor y se superpone con los elementos adyacentos que puedan existir) |
| | **hidden** (el contenido que se muestra es sólo el que cabe en la caja contenedora y el resto queda no visible). |
| | **scroll** (el contenido que se muestra es sólo el que cabe en la caja contenedora y el navegador muestra barras de scroll que permiten acceder al contenido no visible; el scroll se muestra incluso si el contenido no excede la caja). |
| | **auto** (el comportamiento es el que establezca el navegador por defecto para casos de desbordamiento, normalmente mostrar scroll) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {overflow: hidden;}<br>.elementoExpress {overflow: scroll;} |

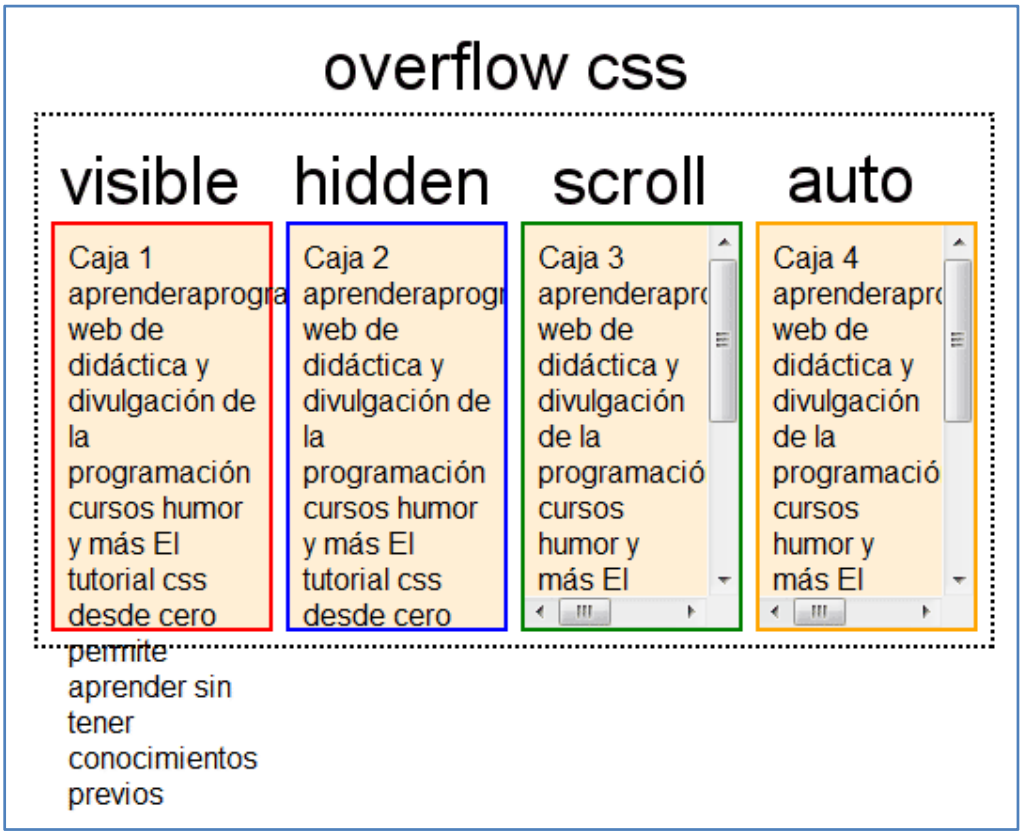
Crea un documento HTML con este contenido:

```
<html> <head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01038DA.css">
</head>
<body>
<div id="caja1">Caja 1 aprenderaprogramar.com web de didáctica y divulgación de la programación cursos humor y más
El tutorial css desde cero permite aprender sin tener conocimientos previos</div>
<div id="caja2">Caja 2 aprenderaprogramar.com web de didáctica y divulgación de la programación cursos humor y más
El tutorial css desde cero permite aprender sin tener conocimientos previos</div>
<div id="caja3">Caja 3 aprenderaprogramar.com web de didáctica y divulgación de la programación cursos humor y más
El tutorial css desde cero permite aprender sin tener conocimientos previos</div>
<div id ="caja4">Caja 4 aprenderaprogramar.com web de didáctica y divulgación de la programación cursos humor y más
El tutorial css desde cero permite aprender sin tener conocimientos previos</div>
<div class="limpiador"></div>
</body>
</html>
```

© aprenderaprogramar.com, 2006-2029

Y un archivo de hoja de estilos con estas reglas (pon el nombre de archivo adecuado) que nos permitirán ver el resultado de aplicar los distintos valores de overflow:

```
/* Curso CSS estilos aprenderaprogramar.com*/

*{font-family: arial;}
body {width:507px; border-style: dotted; border-width: 2px;}

div {border-style: solid; border-width:2px; margin:56px 0px 7px 7px;
padding:7px; background-color: #FFEFD5; width:100px; height:200px; }

#caja1{ border-color:red; float:left; overflow: visible;}
#caja2{  border-color:blue;  float:left; overflow: hidden;}
#caja3{  border-color:green;  float: left; overflow: scroll;}
#caja4{ border-color:orange; float: left; overflow: auto;}

.limpiador {clear:both; padding:0; width:0; height:0; border-style:none; }
```




En la caja 1 con la propiedad overflow: visible; comprobamos cómo el contenido se sale de la caja tanto hacia la derecha como hacia abajo. No obstante hay una diferencia: hacia la derecha el texto se encuentra otro elemento y queda "tapado", mientras que hacia abajo no encuentra otros elementos, y a pesar de salirse de la caja contenedora (incluso se sale del elemento body) se muestra.

Prueba ahora a establecer la propiedad visibility para distintas cajas con distintos valores, e incluso para el elemento body, y comprueba los resultados.

© aprenderaprogramar.com, 2006-2029

**PROPIEDADES OVERFLOW-X Y OVERFLOW-Y CSS**

Las propiedades overflow-x y overflow-y son completamente análogas a la propiedad overflow, sólo que en vez de controlar tanto el desbordamiento horizontal como vertical, únicamente controlan el horizontal (overflow-x) o el vertical (overflow-y). De esta manera, se pueden especificar distintos comportamientos en horizontal y en vertical.

Estas propiedades pueden no funcionar correctamente en algunos navegadores por lo que son menos usadas que la propiedad overflow. También tienen definidos algunos valores como no-display y no-content que no son reconocidos por algunos navegadores.

En algunos casos, al establecer valores visible o hidden para estas propiedades en sentido horizontal o vertical el navegador introduce barras de scroll en el otro sentido, aún no habiéndose especificado nada al respecto.

**EJERCICIO**

Crea un contenedor div con las siguientes características definidas a través de CSS: ancho y alto 200 píxeles, color de fondo amarilo, borde color azul de 2 píxeles de ancho tipo sólido, un texto de tamaño 30 píxeles y con un largo suficiente para exceder la capacidad del div contenedor, y mediante la propiedad overflow haz que aparezcan scrolls horizontal y vertical que permitan visualizar todo el texto. Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01039D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## PROPIEDAD Z-INDEX

Hasta ahora hemos hablado del posicionamiento de elementos en el plano de la pantalla de visualización y en dos direcciones: horizontal (x) y vertical (y). CSS permite, aunque con algunas limitaciones, gestionar la posición en la vertical (eje z) de modo que dados dos elementos superpuestos se pueda indicar cuál debe mostrarse encima y cuál debajo.

| PROPIEDAD CSS z-index | |
|---|---|
| **Función de la propiedad** | Permite definir cómo se colocan unos elementos encima de otros. |
| **Valor por defecto** | auto |
| **Aplicable a** | Elementos con posición establecida explícitamente. |
| **Valores posibles para esta propiedad** | **auto** (no hay z-index especificado) |
| | **Un número entero** (el elemento está más próximo al usuario cuanto mayor sea el valor de su z-index; se admiten números negativos). |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {z-indez: 22;}<br>.elementoJukeBox {z-index:0;} |

Es importante destacar que z-index no vale para superponer contenidos a nuestro antojo, ya que sólo funciona cuando se cumplen determinadas condiciones.

En el caso de un elemento "A" cuyo contenido (texto) se sale de su contenedor y se superpone con otro elemento "B" situado con posterioridad en el código HTML, el texto quedará por debajo del elemento "B" debido al orden del código. La propiedad z-index no funcionará ya que ni siquiera se trata de una superposición entre elementos, sino del texto desbordado de uno con el contenedor de otro.

En el caso de un elemento "A" posicionado y otro "B" no posicionado, no podrá establecerse el orden en el eje z usando z index ya que ambos elementos tendrían que estar posicionados para responder a z-index.

En el caso de elementos situados dentro de cajas contenedoras, su valor de z-index se establece en relación a la caja contenedora. Supongamos que una caja "A" tiene posición y z-index 10 y dentro de ella hay una caja "B" con posición y z-index "20". Otra caja "C" tiene posición y z-index 15. Si las cajas A, B y C se superponen ¿En qué orden se muestran? Se mostrará la caja C en posición superior (por tener z-index 15 frente al valor 10 de la otra caja en su mismo nivel) y debajo la caja A (con la caja B en su interior). La caja B es hija de la caja A por lo que su z-index es relativo a su padre, no puede compararse con otras cajas que no estén en su mismo nivel.

© aprenderaprogramar.com, 2006-2029

Crea un documento HTML con este contenido:

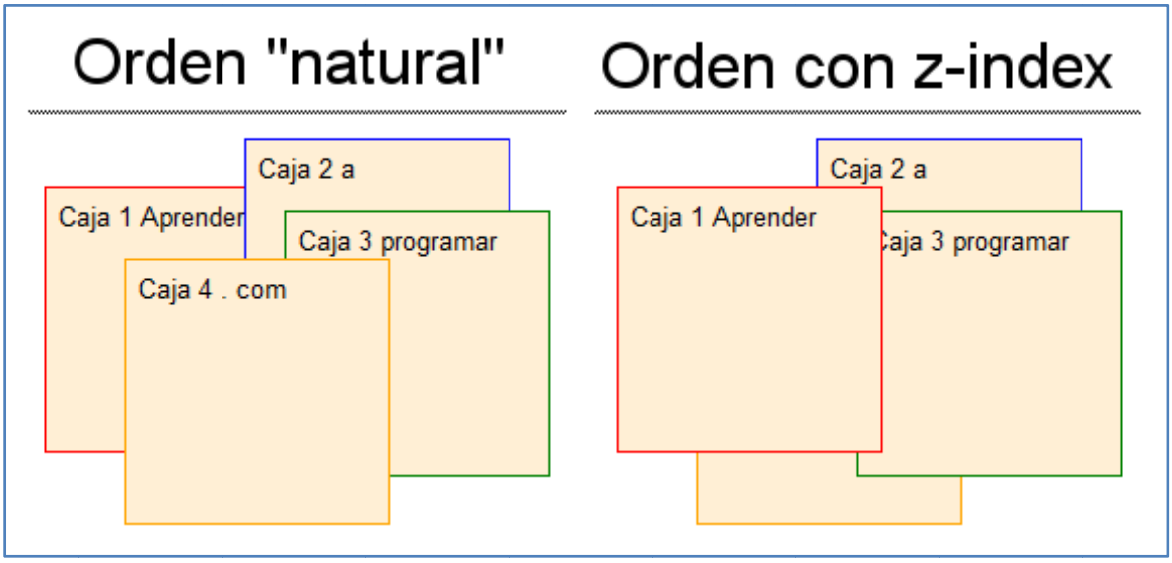
```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01039DA.css">
</head>
<body>
<div id="caja1">Caja 1 Aprender</div>
<div id="caja2">Caja 2 a</div>
<div id="caja3">Caja 3 programar</div>
<div id ="caja4">Caja 4 . com</div>
</body>
</html>
```

Y un archivo de hoja de estilos con estas reglas (pon el nombre de archivo adecuado) que nos permitirán ver el resultado antes de aplicar z-index:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width:507px; border-style: dotted; border-width: 2px;}
div {border-style: solid; border-width:2px;
padding:7px; background-color: #FFEFD5; width:150px; height:150px; position:absolute; }
#caja1{ border-color:red; top: 55px; left: 50px;  }
#caja2{ border-color:blue; top: 25px; left: 175px; }
#caja3{ border-color:green; top: 70px; left: 200px; }
#caja4{ border-color:orange; top: 100px; left: 100px; }
```

Ahora añade propiedades z-index de la siguiente manera y compara los resultados de una forma y otra:

```
#caja1{ border-color:red; top: 55px; left: 50px; z-index:40;  }
#caja2{ border-color:blue; top: 25px; left: 175px; z-index:10;}
#caja3{ border-color:green; top: 70px; left: 200px; z-index:20 }
#caja4{ border-color:orange; top: 100px; left: 100px; z-index:0;}
```




© aprenderaprogramar.com, 2006-2029

Elimina la propiedad position:absolute; para todos los div, manteniendo el resto del código igual. Si visualizas el resultado comprobarás que las propiedades top y left son ignoradas y que el posicionamiento sigue el flujo natural de elementos block en un documento. Declara ahora posición absoluta únicamente para la caja 4 con #caja4{ border-color:orange; top: 100px; left: 100px; z-index:0; position:absolute;} El resultado será que la caja 4 se superpone con las otras cajas. Sin embargo, el hecho de que la caja 4 tenga un z-index 0 y las otras cajas un z-index superior no lleva a que la caja 4 se vaya al fondo. ¿Por qué? Porque z-index sólo funciona entre elementos con posición establecida. En este caso la caja 4 tiene una posición absoluta establecida y está fuera del flujo normal del documento, mientras que las otras cajas no tienen posición establecida y están dentro del flujo normal.

Las cajas con posición absoluta se sitúan por encima de las cajas dentro del flujo normal del documento, es como si estuvieran en dos planos distintos, un plano superior para elementos con posición absoluta y un plano inferior para elementos dentro del flujo normal.

Con frecuencia se trata de aplicar z-index para elementos no posicionados, o entre elementos posicionados y no posicionados, y el resultado es que aparentemente z-index "no funciona". z-index sí funciona, pero de acuerdo con unas reglas y limitaciones que debemos conocer para no tener quebraderos de cabeza innecesarios.

**EJERCICIO**

Analiza el siguiente código, visualiza su resultado y responde a las preguntas:

```
<html>
<head>
<title>Ejemplo aprenderaprogramar.com</title>
<meta charset="utf-8">
<style type="text/css">
*{font-family: sans-serif;}
#cajaGris { width: 225px; height: 225px; border: solid 1px #ccc; background: #ddd; margin-top:20px; }
#cajaAzul {width: 225px; height: 225px; border: solid 3px #4a7497;
background: #8daac3; margin-top: -50px; margin-left: 50px;}
#cajaOcre { width: 225px; height: 225px; border: solid 2px #8b6125;
background: #ba945d; margin-top: -50px; margin-left: 100px; margin-bottom: 20px;}
</head>
</style>
<body>
<div id ="cajaGris">Caja gris</div><div id ="cajaAzul">Caja azul</div><div id ="cajaOcre">Caja ocre</div>
</body>
</html>
```

a) ¿Cuántas cajas contenedoras hay? ¿Están identificadas por id o por class?

b) ¿Cuál es el grosor del borde de la caja azul y en qué unidades está expresado?

c) ¿Por qué se superponen unas cajas encima de otras? ¿En qué orden aparecen las cajas superpuestas (es decir, cuál está arriba, cuál está en posición intermedia y cuál está en el fondo)? ¿Por qué aparecen con ese orden y no otro?

© aprenderaprogramar.com, 2006-2029

d) Queremos que la caja gris se sitúe por encima del resto de cajas. ¿Qué modificaciones en el código hemos de hacer para lograr este objetivo?

e) Modifica el código para que la caja ocre quede en el fondo, la caja azul en posición intermedia y la caja gris encima.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01040D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## PROPIEDADES PARA CONTROL DEL TEXTO CSS

CSS dispone de una serie de propiedades que permiten controlar la forma en que se muestra el texto. Entre ellas vamos a estudiar algunas como text-align, color, text-decoration, text-indent, white-space. Algunas de ellas son muy usadas como text-align. Otras como white-space, aún siendo menos usadas tienen un gran potencial.

### PROPIEDAD TEXT-ALIGN

| PROPIEDAD CSS text-align | |
|---|---|
| **Función de la propiedad** | Permite definir cómo se alinea una unidad de texto (párrafo o similar). |
| **Valor por defecto** | left |
| **Aplicable a** | Contenido de elementos tipo block y contenido de tablas |
| **Valores posibles para esta propiedad** | **left, right, center, justify** (el texto se alinea según se indique) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {text-align: center;}<br>.elementoJukeBox {text-align: justify;} |

Los elementos inline dentro del texto como imágenes o spans quedan también afectados por esta propiedad. La propiedad actúa siempre en relación a la alineación horizontal del texto. La alineación vertical habrá que realizarla usando otras propiedades.

| | | | |
|---|---|---|---|
| Alineación del texto left | Alineación del texto right | Alineación del texto center | Alineación del texto justify el texto queda alineado tanto por la derecha como por la izquierda mediante el ajuste del espacio. |

Esta propiedad se hereda directamente por parte de los elementos hijos.

### PROPIEDAD COLOR

Para establecer el color de un texto se usa la propiedad color. En realidad sería más razonable que se llamara text-color, pero por motivos históricos se ha mantenido la denominación color. Ya hemos hablado de la forma de expresar colores y grados de transparencia para colores usando CSS en anteriores apartados del curso. Recordar que es posible usar notación RGB decimal o porcentual, notación hexadecimal o notaciones RGBA, HSL y HSLA. Ejemplo: contentJBox {color: # FF0000;} Esta propiedad se hereda directamente por parte de los elementos hijos con la única excepción de los enlaces <a> ...</a>, cuyo color tiene que establecerse directamente si quiere ser cambiado.

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD TEXT-DECORATION**

| PROPIEDAD CSS text-decoration | |
|---|---|
| **Función de la propiedad** | Permite definir algunos elementos decorativos para texto como el subrayado o el efecto de tachado. |
| **Valor por defecto** | none |
| **Aplicable a** | Textos |
| **Valores posibles para esta propiedad** | **none** (no hay decoración), **underline** (subrayado normal)**, overline** (subrayado por encima)**, line-through** (efecto de tachado) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {text-decoration: none;}<br>.elementoJukeBox {text-decoration: line-trough;} |

Esta propiedad sirve para evitar el uso de las antiguas etiquetas HTML de subrayado <u> …</u> y tachado <strike> …</strike>. ¿Por qué no deben usarse estas etiquetas HTML? Porque como ya hemos comentado con anterioridad en los desarrollos web actuales HTML debe usarse exclusivamente para definir la estructura y contenedores del documento, mientras que los estilos deben estar definidos por separado mediante CSS.

Un valor adicional para esta propieadad era blink, que servía para indicar que el texto debía parpadear. No obstante, este valor se ha eliminado del estándar por lo que no debe usarse.

El valor none puede usarse para anular el subrayado que por defecto introducen los navegadores para los links (elementos a) si establecemos a {text-decoration: none;}.

Aunque text-decoration es una propiedad sencilla, tiene un aspecto que puede resultar bastante problemático. Supongamos que queremos establecer que todo el texto dentro de un div vaya subrayado, pero que en el elemento con id=" contentMbox" el texto no lleve subrayado. Podríamos pensar en usar este código:

```
div {text-decoration: underline;}
#contentMbox {text-decoration: none;}
```

El resultado es que aún dentro del elemento para el que hemos especificado text-decoration:none; el texto sigue estando subrayado. ¿Por qué? Porque text-decoration es una propiedad un tanto especial que se dibuja acumulativamente transmitiéndose de padres a hijos sin ser anulable por los hijos. En este caso lo que ocurre es que a todos los div se le aplica el subrayado y luego, acumulativamente, al elemento contentMbox se le aplica text-decoration: none;. El resultado es underline + none = underline, el texto aparece subrayado. Supongamos ahora que escribimos esto:

```
div {text-decoration: underline;}
#contentMbox {text-decoration: overline;}
```

El texto dentro del elemento contentMbox aparecerá con una línea por abajo (underline) y una línea por arriba (overline) debido al carácter acumulativo de esta propiedad.

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD TEXT-INDENT**

| PROPIEDAD CSS text-indent | |
|---|---|
| **Función de la propiedad** | Permite definir el indentado (desplazamiento hacia dentro) de la primera línea de un párrafo de texto. |
| **Valor por defecto** | 0 |
| **Aplicable a** | Textos |
| **Valores posibles para esta propiedad** | Una unidad de medida absoluta o relativa (incluido %) permitida en CSS |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {text-indent: 20px;}<br>.elementoJukeBox {text-indent: 0.6em;} |

En el caso de utilizarse un valor en %, el porcentaje se calcula como porcentaje del valor width del elemento padre.

**PROPIEDAD WHITE-SPACE**

| PROPIEDAD CSS white-space | |
|---|---|
| **Función de la propiedad** | Determina el tratamiento que le da el navegador a los espacios en blanco. |
| **Valor por defecto** | normal |
| **Aplicable a** | Todos los elementos |
| **Valores posibles para esta propiedad** | **normal** (si aparece una secuencia de varios espacios en blanco dentro de un texto en el documento HTML, el navegador los colapsa dejando un solo espacio; el espacio en blanco se usa para generar saltos de línea si se excede el tamaño del contenedor. Los saltos de línea en el documento HTML se ignoran) |
| | **nowrap** (el comportamiento es como el normal, pero el espacio en blanco no se usa para generar saltos de línea, por lo que todo el texto queda en una sola línea) |
| | **pre** (el texto se conserva tal y como esté en el documento HTML, conservando las secuencias de espacio y saltos de línea como estén allí definidos) |
| | **pre-wrap** (el comportamiento es como con wrap pero los espacios en blanco se usan para generar nuevas líneas si no hay espacio sufiente disponible) |
| | **pre-line** (se mantienen los saltos de línea definidos en el documento HTML, se colapsan secuencias de espacios en blanco y se usan espacios en blanco para generar nuevas líneas si no hay espacio suficiente disponible) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {white-space: pre;}<br>.elementoJukeBox {white-space: nowrap;} |

© aprenderaprogramar.com, 2006-2029

Esta propiedad evita el uso de la etiqueta HTML <pre> cuyo uso es no recomendado.

Puede resultar de gran interés para realizar ciertas maquetaciones de ciertas partes de páginas webs donde queremos que el texto se comporte de una manera específica en relación a los espacios en blanco, tabuladores y saltos de línea.

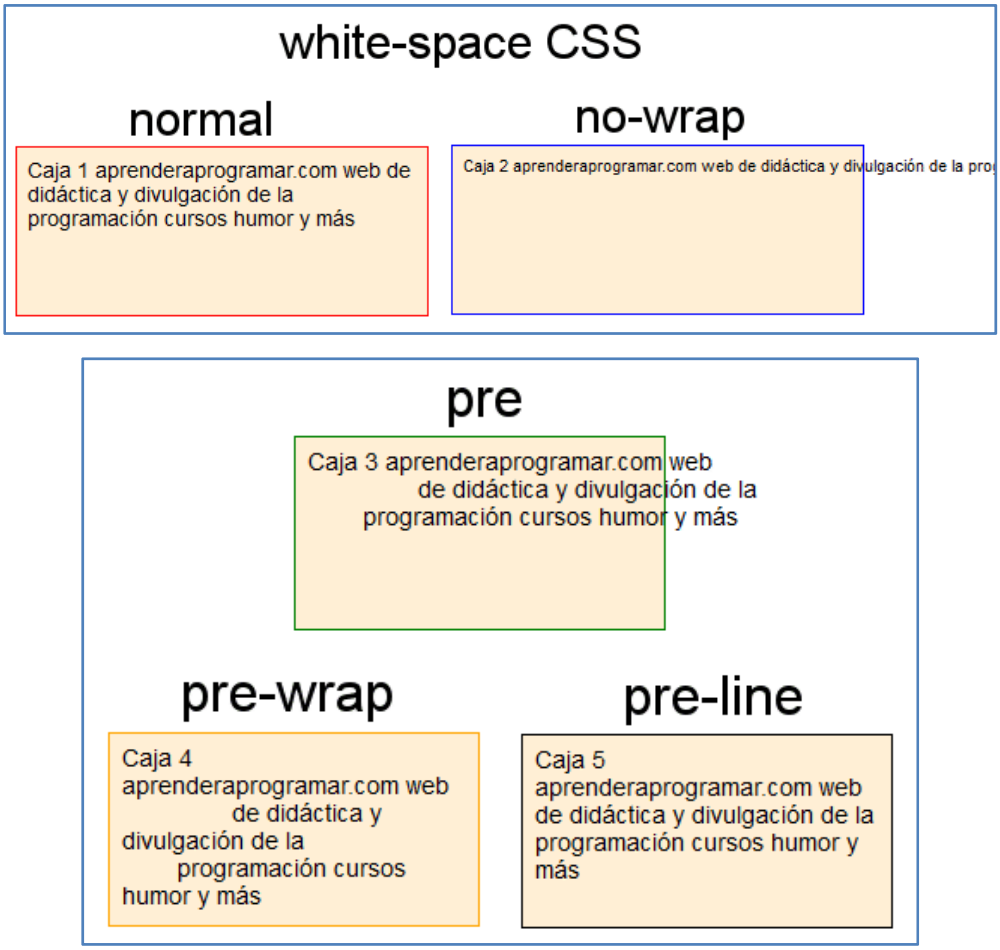
Crea un documento HTML con este contenido (respetando los espacios y saltos de línea):

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01040DA.css">
</head>
<body>
<div id="caja1">Caja 1 aprenderaprogramar.com web
          de didáctica y divulgación de la
           programación cursos humor y más
</div>
<div id="caja2">Caja 2 aprenderaprogramar.com web
          de didáctica y divulgación de la
           programación cursos humor y más</div>
<div id="caja3">Caja 3 aprenderaprogramar.com web
          de didáctica y divulgación de la
           programación cursos humor y más</div>
<div id ="caja4">Caja 4 aprenderaprogramar.com web
          de didáctica y divulgación de la
           programación cursos humor y más</div>
<div id ="caja5">Caja 5 aprenderaprogramar.com web
          de didáctica y divulgación de la
           programación cursos humor y más</div>
<div class="limpiador"></div>
</body>
</html>
```

Y un archivo de hoja de estilos con estas reglas (pon el nombre de archivo adecuado) que nos permitirán ver el resultado del uso de white-space:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
body {width:650px; border-style: dotted; border-width: 2px;}
div {border-style: solid; border-width:2px; margin:56px 0px 7px 7px;
padding:7px; background-color: #FFEFD5; width:290px; height:110px; }
#caja1{ border-color:red; white-space:normal;}
#caja2{  border-color:blue; white-space: nowrap; font-size:12px;}
#caja3{  width: 225px; border-color:green; white-space: pre;}
#caja4{ width: 225px; border-color:orange; white-space: pre-wrap;}
#caja5{ width: 225px; border-color:black; white-space: pre-line}
.limpiador {clear:both; padding:0; width:0; height:0; border-style:none; }
```

© aprenderaprogramar.com, 2006-2029




En la caja 1 vemos el comportamiento normal, según el cual varios espacios en blanco seguidos son ignorados y el texto se va desarrollando sobre la caja y aprovechando un espacio en blanco para crear el salto de línea cuando es necesario debido a que el tamaño del contenedor es insuficiente.

En la caja 2 vemos el comportamiento no-wrap. El texto no salta de línea y se mantiene en una única línea, excediendo el tamaño del contendor sin adaptarse a él. Si quisiéramos evitar que el texto salga fuera del contenedor podríamos usar la propiedad overflow.

En la caja 3 vemos el comportamiento con pre. El texto se mantiene tal cual ha sido definido en el documento HTML, incluyendo secuencias de espacios, tabuladores, saltos de línea, etc.

En la caja 4 vemos el comportamiento con pre-wrap. En este caso se mantiene el texto tal cual ha sido definido en el documento HTML, pero allí donde no hay espacio suficiente el texto crea nuevas líneas aprovechando espacios en blanco para hacerlo.

En la caja 5 vemos el comportamiento con pre-line. En este caso se mantienen los saltos de línea del documento HTML, pero los espacios en blanco se colapsan y se generan saltos de línea automáticos cuando el contenido excede el tamaño del contenedor.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Crea un documento HTML y un archivo con la hoja de estilos CSS que cumpla con estos requisitos:

a) Deben exisitir tres contenedores (div1, div2 y div3) situados en horizontal, cada uno con margin 20px en todas direcciones, relleno de 10 píxeles en todas direcciones, ancho de 200 píxeles, altura de 400 píxeles y borde sólido de 4 píxeles de anchura.

b) El div 1 debe tener alineación del texto centrada, color de texto #FF6347. Debe contener un texto suficientemente largo, con algunas palabras subrayadas y el primer párrafo indentado un 10%.

c) El div 2 debe tener alineación del texto a la derecha, color de texto #008080. Debe contener un texto suficientemente largo, con algunas palabras subrayadas por encima y el primer párrafo indentado en 50 píxeles. Este div estará definido en el HTML como un texto con saltos de línea definidos en ciertos puntos. Al mostrarse por pantalla, deben usarse propiedades CSS para se mantengan los saltos de línea definidos en el documento HTML, se colapsen secuencias de espacios en blanco y se usen espacios en blanco para generar nuevas líneas si es necesario

d) El div 3 debe tener alineación del texto justificada, color de texto #8B4513. Debe contener un texto suficientemente largo, con algunas palabras tachadas, y el primer párrafo indentado en un 20%. Además el texto contendrá una frase donde existan 15 espacios en blanco seguidos, y dichos espacios deben mostrarse cuando se visualice el texto (si coincide con un borde del contenedor deberán continuar en una nueva línea).

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01041D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## CONTROL DEL TEXTO CSS

Ya conocemos algunas propiedades para dar formato a textos, como text-align, color, text-decoration, text-indent, white-space, etc. Vamos a continuar repasando propiedades que resultan de interés para darle formato a textos como text-overflow, line-height y text-shadow. Con text-shadow podremos crear sombras con distintos efectos lo cual nos resultará útil para mejorar nuestros diseños web.

## PROPIEDAD TEXT-OVERFLOW

Ya conocemos la propiedad overflow, que permite definir cómo debe comportarse el navegador en caso de que el contenido de una caja exceda su tamaño, por ejemplo si el texto que excede el tamaño de su caja contenedora debe permanecer visible, oculto, o accesible mediante barras de scroll.

La propiedad text-overflow se concibió para dar un mayor control sobre cómo debería comportarse un texto que se sale de su contenedor, permitiendo el reemplazo del final del texto por unos puntos suspensivos (ellipsis) o por una cadena definida por nosotros, por ejemplo podría ser "Leer más...".

Para aplicar esta propiedad tenemos que establecer antes la propiedad overflow con valor hidden, ya que si no lo hacemos no observaremos efecto alguno (prevalecerá lo indicado por overflow). Además lo indicado se aplicará al texto que no quepa en sentido horizontal pero en cualquier punto del texto, no sólo al final. Esto puede generar efectos extraños (por ejemplo que unos puntos suspensivos aparezcan en un punto intermedio, cuando realmente sólo deberían aparecer al final). Para evitar esto habremos de forzar que el texto no haga saltos de línea usando white-space: nowrap; gracias a lo que el text-overflow se aplicará exactamente a la parte final del texto.

| PROPIEDAD CSS text-overflow | |
|---|---|
| **Función de la propiedad** | Permite generar un tratamiento específico para texto que excede el tamaño de su contenedor. |
| **Valor por defecto** | clip |
| **Aplicable a** | Elementos tipo block con texto |
| **Valores posibles para esta propiedad** | **clip** (el texto se corta al llegar al borde del contenedor) |
| | **ellipsis** (la parte final del texto se reemplaza con ... antes de salirse del contenedor) |
| | Una cadena, escrita entre comillas simples, por ejemplo 'Leer más' (no admitido por algunos navegadores) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {overflow: hidden; white-space: nowrap; text-overflow: ellipsis;} |

© aprenderaprogramar.com, 2006-2029

Supongamos que | representa el límite del contenedor. Supongamos este texto:

aprenderaprogramar.com es una w| eb para aprender programación.

Los resultados que obtendríamos para los distintos casos serían:

**clip**: aprenderaprogramar.com es una w|

**ellipsis**: aprenderaprogramar.com es una ..|

'...continúa' : aprenderaprogramar.com ...continúa |

Nota: debido a las limitaciones y problemas de compatibilidad entre navegadores no recomendamos el uso de esta propiedad. Usa preferiblemente overflow en lugar de text-overflow.

**PROPIEDAD LINE-HEIGHT**

Esta propiedad es de amplio uso para establecer la separación entre líneas cuando se muestra un texto.

| PROPIEDAD CSS line-height | |
|---|---|
| **Función de la propiedad** | Permite fijar la altura ocupada por las líneas (interlineado). |
| **Valor por defecto** | normal |
| **Aplicable a** | Todos los elementos |
| **Valores posibles para esta propiedad** | **normal** (el interlineado será el predeterminado por el navegador o el existente por herencia) |
| | **Un número sin especificar unidades** (el interlineado será tantas veces el interlineado normal como indique el número) |
| | **Una unidad de medida relativa o absoluta** (se admiten porcentajes) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {line-height: 1.3;}<br>.elementoJukeBox {line-height: 130%;} |

Se recomienda no especificar unidades para line-height porque puede entrar en conflicto con el tamaño del texto. Por ejemplo si el texto es de 14px y especificamos line-height 10px el texto no cabría en la línea y se generaría una superposición. Otro caso problemático sería tener un elemento h1 dentro de un div y aplicar line-height: 1.1em. Al hacer esto se fuerza al h1 a que tome como interlineado 1.1 veces el tamaño de letra definido para el div, lo que puede generar distorsiones. En cambio especificando simplemente 1.1 sin unidades todos los elementos tomarán como interlineado 1.1 veces lo que sería su interlineado normal, lo cual proporciona un resultado más seguro que otra especificación basada en unidades.

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD TEXT-SHADOW**

Esta propiedad sirve para añadir efectos de sombra a un texto, lo cual permite generar diseños atractivos.

La sintaxis a emplear es la siguiente:

```
selectorElemento {text-shadow: valorX valorY blurOpcional colorOpcional; }
```

Donde valorX indica el tamaño o desplazamiento de la sombra respecto al texto hacia la derecha (valores positivos) o hacia la izquierda (valores negativos), expresado con una unidad de medida válida en CSS.

ValorY indica el tamaño o desplazamiento de la sombra respecto al texto hacia abajo (valores positivos) o hacia arriba (valores negativos), expresado con una unidad de medida válida en CSS.

blur es un parámetro opcional que indica un efecto de desenfoque o difuminado de la sombra, expresado con una unidad de medida válida en CSS. A mayor valor, la sombra tiene un mayor difuminado y se vuelve más clara y brillante.

El color es un parámetro opcional. Si se indica, especifica el color de la sombra. Si no se indica, el color de la sombra será el mismo que el color que tenga el texto.

| PROPIEDAD CSS text-shadow | |
|---|---|
| **Función de la propiedad** | Permite definir un efecto de sombra para el texto. |
| **Valor por defecto** | none |
| **Aplicable a** | Todos los elementos |
| **Valores posibles para esta propiedad** | **none** (no hay efecto de sombra) |
| | valorX valorY blurOpcional colorOpcional (genera una sombra de acuerdo a los parámetros especificados; se permiten especificar 2 ó más sombras, separando su especificación mediante comas) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 { text-shadow: 2px 2px red;}<br>.elementoJukeBox { text-shadow: -6px -6px 12px blue ;} |

Esta propiedad está disponible en todos los navegadores recientes, pero algunos más antiguos pueden no reconocerla.

© aprenderaprogramar.com, 2006-2029

Crea un documento HTML con este contenido para probar esta propiedad y visualiza los resultados:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{font-family: arial;}
h1 {padding:10px 20px;}
</style>
</head>
<body>
<h1 style="color: black; text-shadow: 2px 2px red;">aprenderaprogramar.com</h1>
<h1 style="color: black; text-shadow: 2px 2px 4px red;">aprenderaprogramar.com</h1>
<h1 style="color: black; text-shadow: 2px 2px 12px red;">aprenderaprogramar.com</h1>
<h1 style="color: black; text-shadow: 2px 2px 12px red, -6px -6px blue ;">aprenderaprogramar.com</h1>
<h1 style="color: black; text-shadow: 2px 2px 12px red, -6px -6px 12px blue ;">aprenderaprogramar.com</h1>
</html>
```

En la imagen anterior observamos en primer lugar una sombra sólida de color rojo con desplazamiento hacia la derecha y abajo. En segundo lugar tenemos esa misma sombra con un ligero difuminado (blur) y en tercer lugar la misma sobra con un mayor difuminado aún. En cuarto lugar tenemos dos sombras, una hacia abajo a la derecha roja y difuminada y otra hacia arriba a la izquierda azul y sólida. En quinto lugar la sombra azul anterior se ha difuminado, con lo cual observamos el efecto de dos sombras difuminadas, una roja hacia abajo a la derecha y otra azul hacia arriba a la izquierda.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Crea un documento HTML y un archivo con la hoja de estilos CSS que cumpla con estos requisitos:

a) Deben exisitir tres contenedores (div1, div2 y div3) situados en horizontal, cada uno con margin 18px en todas direcciones, relleno de 8 píxeles en todas direcciones, ancho de 240 píxeles, altura de 300 píxeles y borde sólido de 3 píxeles de anchura con color de borde azul.

b) El div 1 debe contener un texto suficientemente largo como para exceder el tamaño del contenedor, y el excedente de texto no debe mostrarse apareciendo en el punto final unos puntos .... La altura de línea debe ser un 5 % superior a lo normal. Un fragmento del texto (delimitarlo con span) debe tener tamaño de fuente 18 píxeles y una sombra sólida de color rojo con desplazamiento hacia la derecha y abajo.

c) El div 2 debe contener un texto suficientemente largo como para exceder el tamaño del contenedor, y el excedente de texto no debe mostrarse aunque rebase al contenedor, y no deben aparecer puntos en el lugar donde se corte el texto. La altura de línea debe ser un 10 % superior a lo normal. Un fragmento del texto (delimitarlo con span) debe tener tamaño de fuente 18 píxeles y una sombra sólida de color azul con desplazamiento hacia la izquierda y abajo.

d) El div 3 debe tener alineación del texto justificada y color de texto #8B4513. Debe contener un texto suficientemente largo como para exceder el tamaño del contenedor. La altura de línea debe ser un 20 % superior a lo normal. Un fragmento del texto (delimitarlo con span) debe tener tamaño de fuente 18 píxeles y una sombra con efecto de difuminado de color verde de modo que parezca que existe vapor verde detrás del texto.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01042D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## MÁS CONTROL DEL TEXTO CSS

Continuamos repasando propiedades que resultan de interés para darle formato a textos como text-transform, que nos permitirá entre otras posibilidades transfomar un texto a mayúsculas o minúsculas. Además veremos como word-spacing y letter-spacing nos permiten variar el espacio de separación entre palabras o letras y word-wrap nos permitirá romper palabras largas que no caben en su contenedor y no tienen espacios intermedios que permitan crear saltos de línea.

## PROPIEDAD TEXT-TRANSFORM

| PROPIEDAD CSS text-transform | |
|---|---|
| **Función de la propiedad** | Permite transformar la apariencia de un texto. |
| **Valor por defecto** | none |
| **Aplicable a** | Todos los elementos |
| **Valores posibles para esta propiedad** | **none** (no hay transformación del texto) |
| | **uppercase** (transforma el texto a mayúsculas) |
| | **lowercase** (transforma el texto a minúsculas) |
| | **capitalize** (convierte la primera letra de cada palabra en mayúsculas) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {text-transform: uppercase;} |

Supongamos que tenemos el texto: "La mejor web para APRENDER a programar". Los efectos serían:

**uppercase**: LA MEJORA WEB PARA APRENDER A PROGRAMAR

**lowercase**: la mejor web para aprender a programar

**capitalize**: La Mejor Web Para Aprender A Programar

Recordar que hay un selector que nos permite definir estilos para la primera letra de texto de un párrafo o contenedor que es first-letter. Podemos combinar este selector con text-transform para poner la primera letra en un formato especial. Por ejemplo p::first-letter {font-size: 300%; text-transform: uppercase;} nos permite poner la primera letra en mayúsculas y con un formato extragrande.

© aprenderaprogramar.com, 2006-2029

**PROPIEDADES LETTER-SPACING Y WORD-SPACING**

Se trata de dos propiedades de uso no habitual que permiten establecer una separación entre letras específica o una separación entre palabras específica.

Su valor por defecto es normal y se establecen indicando una unidad de medida válida en CSS como valores en píxeles o en em.

Ejemplos: content1 { letter-spacing: 9px;}

#jukBox { word-spacing: 1.3em;}

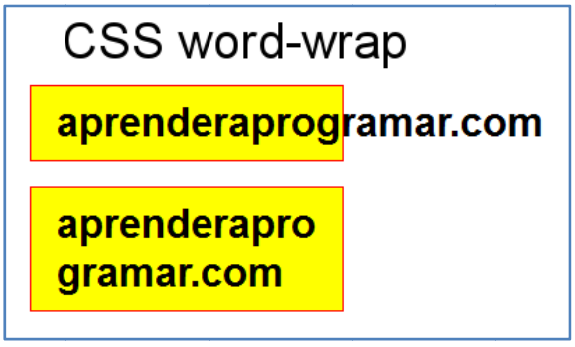
Se admiten valores negativos, pero en algunos casos pueden generar efectos extraños indeseados (por ejemplo superposición de palabras o letras unas encima de otras).

**PROPIEDAD WORD-WRAP**

Ya sabemos que a no ser que hayamos especificado lo contrario con la propiedad white-space, las palabras aprovecharán espacios en blanco para situarse en una nueva línea cuando el espacio a lo ancho es insuficiente en el contenedor. No obstante, existe el problema de que una palabra puede ser tan larga que ocupe todo el ancho disponible y se salga del contenedor sin posibilidad de crear una nueva línea por la falta de espacios en blanco (esto se podría denominar "palabra irrompible"). Usando la propiedad word-wrap cuyo valor por defecto es normal y cuyo único valor posible es break-word podemos especificar que la palabra deberá continuar en una nueva línea incluso cuando no exista un espacio en blanco para generar el salto de línea.

Crea un documento HTML con este contenido para probar esta propiedad y visualiza los resultados:

```
<html> <head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{font-family: arial; }
h1 {margin: 20px; padding:10px 20px; border-style: solid; border-color: red; border-width: thin;}
</style> </head>
<body> <h1 style="width:210px; background-color:yellow;">aprenderaprogramar.com</h1>
<h1 style="width:210px; background-color:yellow; word-wrap: break-word;">aprenderaprogramar.com</h1>
</body></html>
```




© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Crea un documento HTML y un archivo con la hoja de estilos CSS que cumpla con estos requisitos:

a) Deben exisitir tres contenedores (div1, div2 y div3) situados en horizontal, cada uno con margin 33px en todas direcciones, sin relleno, ancho de 180 píxeles, altura de 300 píxeles y borde sólido de 6 píxeles de anchura con color de borde rojo.

b) El div 1 debe contener un texto suficientemente largo, con numerosos párrafos, como para exceder el tamaño del contenedor. El texto del html debe transformarse completamente a mayúsculas mediante el uso de propiedades CSS. La separación entre letras debe ser un 5% superior a lo normal. La primera letra de cada párrafo debe tener un tamaño un 250% lo normal.

c) El div 2 debe contener un texto suficientemente largo como para exceder el tamaño del contenedor. El texto del html debe transformarse completamente a minúsculas mediante el uso de propiedades CSS. La separación entre letras debe ser un 10% superior a lo normal. Debe contener una palabra (cadena de texto cualquiera) de gran longitud, de modo que no quepa en el contenedor, y "romperse" para no exceder la capacidad del contenedor usando la propiedad word-wrap.

d) El div 3 debe contener un texto suficientemente largo como para exceder el tamaño del contenedor. El texto del html debe transformarse para que toda palabra comience con una letra mayúscula mediante el uso de propiedades CSS. La separación **entre palabras** debe ser un 10% superior a lo normal. Debe contener una palabra (cadena de texto cualquiera) de gran longitud, de modo que no quepa en el contenedor, y no romperse.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01043D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## CENTRADO VERTICAL DE ELEMENTOS

Hemos visto un número relativamente amplio de propiedades CSS que nos permiten trabajar aspectos de alineación en la horizontal y, sin embargo, las posibilidades de trabajo en la vertical son menores. Vamos a estudiar la propiedad vertical-align que permite el posicionamiento y centrado de elementos en la vertical, aunque su funcionamiento tiene ciertas complicaciones.

## PROPIEDAD VERTICAL-ALIGN

Es muy frecuente oir la expresión "vertical-align no funciona bien" a personas que están comenzando con CSS o que tienen un conocimiento superficial de CSS. La realidad es que hay que conocer para qué sirve esta propiedad y comprenderla para poder valorar los resultados que se obtienen con ella.

Es una propiedad que al principio resulta difícil de entender, vamos a tratar de hacerlo poco a poco.

| PROPIEDAD CSS vertical-align | |
|---|---|
| **Función de la propiedad** | Permite posicionar elementos inline en distintas alineaciones verticales respecto a una línea. También permite alinear verticalmente cualquier elemento dentro de una celda de una tabla. |
| **Valor por defecto** | baseline |
| **Aplicable a** | Elementos inline y elementos dentro de celdas de tablas |
| **Valores posibles para esta propiedad** | **baseline** (el elemento se alinea como lo haría normalmente) |
| | **middle** (elementos inline se alinean de modo que se centran verticalmente respecto a la línea del elemento padre en la que están; elementos en celdas de tablas se centran verticalmente en la celda de la tabla) |
| | **top y bottom** (elementos inline se alinean de modo que se su parte superior se alinea con la línea en la que están; elementos en celdas de tablas se colocan en la parte superior en la celda de la tabla) |
| | **text-top y text-bottom** (elementos inline se alinean de modo que su parte superior se alinea con la parte superior de la línea del elemento padre; su efecto suele ser muy similar al de top y bottom) |
| | **sub y super** (elementos inline se alinean de modo que se su parte superior se alinea con las líneas de subíndice o superíndice del elemento padre) |
| | Una unidad de medida relativa o absoluta válida en CSS (se admiten porcentajes. Desplaza la línea de base del elemento hacia arriba respecto a su baseline en la medida indicada. Si se indica 0 equivale a baseline; si es % se calcula respecto al valor de line-height. Se admiten valores negativos.) |
| | **inherit** (se heredan las características del elemento padre). |

© aprenderaprogramar.com, 2006-2029

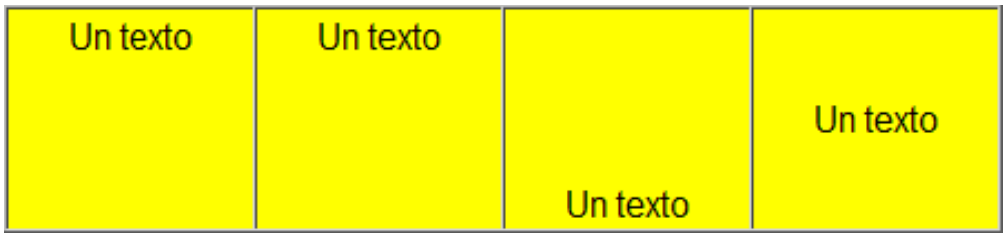
| PROPIEDAD CSS vertical-align | |
|---|---|
| **Ejemplos aprenderaprogramar.com** | img {vertical-align:middle;} |

Como vemos la propiedad vertical-align tiene cierta complejidad, entre otras cosas porque muestra distinto comportamiento según se aplique a elementos en contenedores "normales" o a elementos dentro de celdas de tablas.

Dentro de una tabla el comportamiento de vertical-align podemos decir que es el que espera la mayoría de la gente: permite centrar verticalmente.

Crea un documento HTML con este contenido para probarlo y visualiza los resultados:

```
<html> <head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css"> *{font-family: arial; } table {margin:20px; background-color:yellow;}
td {height: 100px; text-align:center; padding:2px;}
 </style> </head>
<body>
<table border="1" cellpadding="0" cellspacing="0" style="width: 450px;">
<tr> <td style="vertical-align:baseline;" ><p>Un texto</p></td>
<td style="vertical-align:top;" ><p>Un texto</p></td>
<td style="vertical-align:bottom;" ><p>Un texto</p></td>
<td style="vertical-align:middle;" ><p>Un texto</p></td>
</tr> </table> </body> </html>
```




Con las celdas de una tabla el resultado es el esperado: con top el texto se coloca arriba, con bottom se coloca abajo y con middle se centra verticalmente.

Ahora vamos a intentar emular este comportamiento con contenedores "normales" como div. Crea un documento HTML con este contenido para probarlo y visualiza los resultados:

```
<html> <head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css"> *{font-family: arial; }
div {height: 100px; width:112px; text-align:center; padding:2px; float:left; background-color:yellow; border-style:solid;}
 </style> </head>
<body>
<div style="vertical-align:baseline;" ><p>Un texto</p></div>
<div style="vertical-align:top;" ><p>Un texto</p></div>
<div style="vertical-align:bottom;" ><p>Un texto</p></div>
<div style="vertical-align:middle;" ><p>Un texto</p></div>
</body> </html>
```

© aprenderaprogramar.com, 2006-2029

¿Por qué no se alinea el texto? Pues porque la propiedad vertical-align funciona de una manera en tablas y de otra manera fuera de las tablas. Fuera de tablas, esta propiedad no puede aplicarse a elementos tipo block. Por tanto no puede aplicarse a un contenedor esperando que lo que haya dentro de él se centre verticalmente. El código anterior no tiene sentido (aunque mucha gente intenta realizar alineaciones verticales así, de ahí que se oiga con tanta frecuencia que "no funciona").

Fuera de tablas, esta propiedad se aplica a elementos inline para centrarlos respecto a la línea en la que se insertan, no puede ser aplicada a un contenedor para que los elementos dentro de él se alineen verticalmente. El concepto es completamente distinto en tablas que fuera de tablas (lo cual es curioso y habría que preguntarle a los creadores de CSS por qué han hecho esto así). Para comprender el concepto de vertical-align fuera de tablas fíjate en esta imagen porque supone un resumen didáctico del uso de vertical-align fuera de tablas.

En el recuadro superior izquierdo vemos lo que sería una línea dentro de un contenedor div en su posición natural: se coloca en la parte superior del div. En el recuadro inferior izquierdo vemos cómo aparecería la imagen si se inserta dentro de una línea sin especificar vertical-align (o lo que es lo mismo, aplicándole vertical-align: baseline;). En el recuadro superior derecho se ha aplicado vertical-align: top; a la imagen y ésta lo que hace es poner su parte superior alineada con la línea dentro de la cual está la imagen. En el recuadro inferior derecho se ha aplicado vertical-align: middle; a la imagen y ésta se centra verticalmente respecto a la línea dentro de la cual está.

© aprenderaprogramar.com, 2006-2029

Podemos destacar lo siguiente:

a) vertical-align se aplica al elemento inline para alinearlo respecto a la línea dentro de la cual está (por el contrario, text-align se aplica al contenedor para alinear horizontalmente su contenido, lo cual es un concepto distinto).
b) vertical-align no alinea respecto al contenedor, sino respecto a la línea.

Mostramos ahora el código HTML. Escríbelo, visualiza los resultados y trata de comprenderlo:

```
<html> <head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{font-family: arial; }
img { margin-left: 5px; margin-right:5px; }
div { width: 360px; height: 210px; margin: 10px; padding:5px 20px; border-style: solid; border-color: red;
border-width: thin; background-color: yellow; text-align: center;}
 </style> </head>
<body>
<div style="float:left;">Línea de texto sin una imagen</div>
<div style="float:left;">
<p> Línea de texto<img style="vertical-align:baseline;" src="http://i.imgur.com/afC0L.jpg">con una imagen</p>
</div>
<div style="clear:both; float:left;" >
<p> Línea de texto<img style="vertical-align:top;" src="http://i.imgur.com/afC0L.jpg">con una imagen</p>
</div>
<div style="float:left;">
<p> Línea de texto<img style="vertical-align:middle;" src="http://i.imgur.com/afC0L.jpg">con una imagen</p>
</div> </body> </html>
```

**¿Y CÓMO ALINEAMOS ENTONCES VERTICALMENTE?**

Ya hemos visto que vertical-align no nos va a servir para realizar un centrado vertical de un elemento dentro de un contenedor, pero esto nos interesará hacerlo con frecuencia. ¿Cómo hacerlo entonces?

Esta es una cuestión que suele generar problemas a los programadores y diseñadores web porque no existe un único método, sino que hay varias maneras de hacerlo y cada una tiene sus ventajas y sus inconvenientes.

Existen aproximadamente una docena de métodos más o menos estándar descritos para centrar verticalmente con CSS. Sin embargo, ninguno de ellos es la solución perfecta. Aquí vamos a poner un ejemplo de un método sin que esto signifique que sea el mejor ni que lo recomendemos. Simplemente lo ponemos a modo de ejemplo.

En nuestro caso vamos a tratar de centrar verticalmente un div interior situado dentro de un div exterior. A su vez el div interior tiene un texto y vamos a tratar de centrarlo verticalmente.

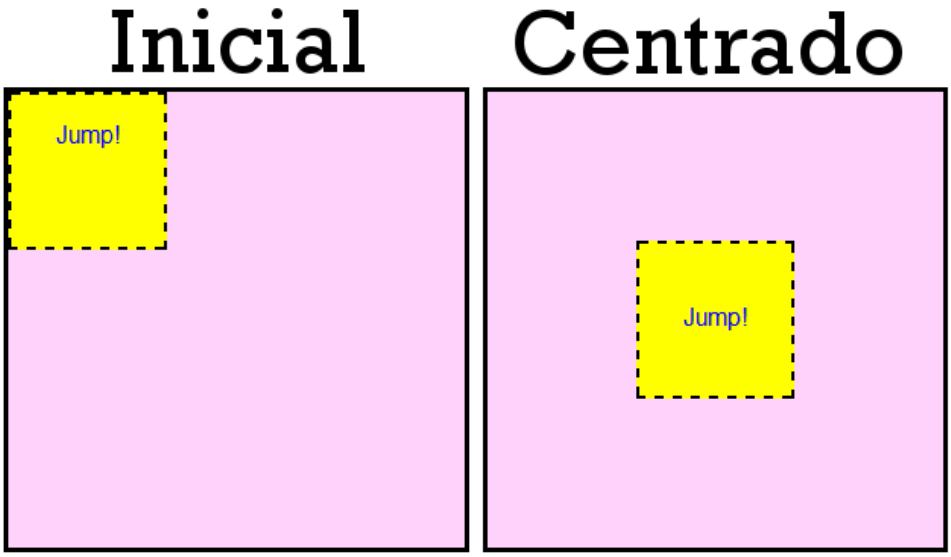
La situación de partida es esta:

© aprenderaprogramar.com, 2006-2029

```
<html> <head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{font-family: arial; }
img { margin-left: 5px; margin-right:5px; }
div { width: 360px; height: 210px; margin: 10px; padding:5px 20px; border-style: solid; border-color: red;
border-width: thin; background-color: yellow; text-align: center;}
 </style> </head>
<body>
<div style="float:left;">Línea de texto sin una imagen</div>
<div style="float:left;">
<p> Línea de texto<img style="vertical-align:baseline;" src="http://i.imgur.com/afC0L.jpg">con una imagen</p>
</div>
<div style="clear:both; float:left;" >
<p> Línea de texto<img style="vertical-align:top;" src="http://i.imgur.com/afC0L.jpg">con una imagen</p>
</div>
<div style="float:left;">
<p> Línea de texto<img style="vertical-align:middle;" src="http://i.imgur.com/afC0L.jpg">con una imagen</p>
</div> </body> </html>
```

Y el código que nos va a servir como solución, basado en posicionamiento absoluto del div interior respecto al exterior, es este:

```
<html> <head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{font-family: arial; }
div {width: 300px; height: 300px; border-style:solid; background-color:#FED2FB; position:relative;}
div div {width:100px; height: 100px;  text-align:center;  background-color:yellow; border-style:dashed;
border-width:2px; top:50%; margin-top:-52px; left: 50%; margin-left:-52px;
position:absolute;  line-height:4em;}
div div p {color:blue; }
 </style> </head>
<body>
<div> <div ><p>Jump!</p></div> </div>
</body> </html>
```




© aprenderaprogramar.com, 2006-2029

Escribe ambos códigos, visualiza el resultado y compara una situación y otra. Trata de comprender todo lo que hemos hecho (con los conocimientos adquiridos a lo largo del curso ya debemos ser capaces de interpretar este código). En un caso hemos posicionado el div interior respecto al interior y en el caso del texto hemos aplicado line-height para forzar al texto a desplazarse verticalmente. Si algo no te queda claro escribe una consulta en los foros (http://aprenderaprogramar.com/foros).

A modo de resumen: no hay un método universal para centrar verticalmente en CSS. Cada programador o diseñador usa uno o varios métodos y trata de buscar soluciones adaptadas a cada situación que se le plantea.

**EJERCICIO**

Analiza el siguiente código, visualiza su resultado y responde a las preguntas:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{font-family: arial; }
div { width: 360px; height: 210px; margin: 10px; padding:5px 20px;
border-style: solid; border-color: red; border-width: thin;
background-color: yellow; text-align: center; float: left;}
#vcent { display: table;  }
#vcent span { display: table-cell; vertical-align:middle;}
 </style>
</head>
<body>
<div><span>Línea de texto contenedor 1</span></div>
<div id="vcent"><span>Línea de texto contenedor 2</span></div>
</body>
</html>
```

a) ¿Cuántas cajas contenedoras hay? ¿Están identificadas por id o por class o por ninguno de ellos?

b) Visualiza el resultado en al menos dos navegadores distintos. ¿Quédiferencias observas entre ambos? ¿A qué crees que se deben?

c) ¿Por qué el texto <<Línea de texto contenedor 2>> se muestra centrado verticalmente?

e) Modifica el código para que el texto <<Línea de texto contenedor 1>> se muestre centrado verticalmente.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01044D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**TIPOGRAFÍA CSS**

Continuamos viendo posibilidades que ofrece CSS en relación a los textos. Un aspecto importante de los textos es el tipo de letra que usan y cómo se configura la letra. Hay distintas propiedades para configurar la tipografía, entre ellas font-size, font-weight y font-style.

**PROPIEDAD FONT-SIZE**

| PROPIEDAD CSS font-size | |
|---|---|
| **Función de la propiedad** | Permite establecer el tamaño de letra. |
| **Valor por defecto** | medium |
| **Aplicable a** | Todos los elementos |
| **Valores posibles para esta propiedad** | Una unidad de medida absoluta o relativa (incluido %, en cuyo caso el cálculo se hace respecto al tamaño de letra del elemento padre) permitida en CSS |
| | Una palabra clave: xx-small, x-small, small, medium, large, x-large, xx-large donde medium es el tamaño por defecto que aplica el navegador. |
| | Una palabra clave: larger ó smaller, que significa más grande o más pequeño que el tamaño de letra del elemento padre. |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {font-size: 20px;}<br>.elementoJukeBox {font-size: small;} |

Cuando se usan las unidades em y ex para fijar el tamaño de letra, el cálculo de éste se hace respecto al tamaño de letra del elemento padre. En cambio cuando se usa em y ex en otro contexto, el tamaño de letra que se toma como referencia es el del propio elemento.

Hay que tener en cuenta que al aplicar font-size se pueden estar modificando dimensiones de otros elementos de la página web debido a que hay unidades de medida como em y ex que toman como referencia el tamaño de letra.

Si establecemos como tamaño de letra en el elemento body un valor como 12 píxeles, entonces 1 em serán 12 píxeles, 2 em serán 24 píxeles y así sucesivamente. Si no se ha establecido un valor de tamaño de letra, entonces 1 em corresponde al valor del tamaño de letra por defecto que use el navegador (que normalmente es de 16 píxeles).

Usar tamaños de letra relativos basados en em ó en palabras clave puede ser interesante, puesto que nos permitirán cambiar la apariencia de toda la página web variando el tamaño de fuente de referencia a través de su definición en el elemento body.

© aprenderaprogramar.com, 2006-2029

Algunos programadores y diseñadores web usan una técnica consistente en establecer el tamaño de fuente en body en el 62.5% (de 16 px) que son 10 px de modo que 1 em serán 10 px. En el resto de elementos usan valores en em cuya correspondencia en píxeles es directa, ya que 1 em serán 10px, 1.2 em serán 12px, 1.6em serán 16px y así sucesivamente.

Los títulos suelen tener una equivalencia similar a la que mostramos a continuación.

| Tipo de título | Palabra clave | Píxeles | em (*) |
|---|---|---|---|
| **h1** | xx-large | 24 | 1.5 |
| **h2** | x-large | 22 | 1.375 |
| **h3** | large | 18 | 1.125 |
| **h4** | medium | 16 | 1 |
| **h5** | small | 12 | 0.75 |
| **h6** | xx-small | 10 | 0.625 |

(*) Tener en cuenta que el valor de un em es cambiante si se modifica el tamaño de fuente de referencia.

**PROPIEDAD FONT-WEIGHT**

| PROPIEDAD CSS font-weight | |
|---|---|
| **Función de la propiedad** | Permite establecer el grosor de letra. |
| **Valor por defecto** | normal |
| **Aplicable a** | Todos los elementos |
| **Valores posibles para esta propiedad** | Un valor numérico múltiplo de 100 comprendido entre 100 y 900 |
| | Una palabra clave: normal (equivale a 400), bold (equivale a 700), bolder (indica "más grueso que el grosor de fuente del elemento padre"), lighter (indica "menos grueso que el grosor de fuente del elemento padre"). |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {font-weight: bold;}<br>.elementoJukeBox {font-weight: 300;} |

Algunas fuentes sólo permiten los valores normal y bold. En este caso cualquier valor entre 100 y 500 se consideraría normal y cualquier valor entre 600 y 900 se consideraría bold.

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD FONT-STYLE**

Esta propiedad es una propiedad sencilla que permite poner una fuente en cursiva (italic) o en oblícuo (oblique), siempre que la fuente que esté en uso lo permita (no todas las fuentes permiten la cursiva por ejemplo).

La cursiva y la oblícua son muy similares, de hecho a veces ni se distinguen. La diferencia entre ambas está en que la cursiva usa caracteres parecidos a los de la fuente original, pero distintos, mientras que la oblícua utiliza los mismos caracteres que la fuente original pero ligeramente inclinados.

El valor por defecto para esta propiedad es 'normal' y sus valores posibles 'normal', 'italic' y 'oblique'. Un ejemplo de aplicación sería este: p.italic { font-style:italic }

**EJERCICIO**

Elige dos navegadores distintos y rellena la siguiente tabla para cada uno de ellos.

| Tipo de título | Píxeles | em |
|---|---|---|
| **h1** | | |
| **h2** | | |
| **h3** | | |
| **h4** | | |
| **h5** | | |
| **h6** | | |

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01045D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**TIPOGRAFÍA CSS**

Continuamos viendo posibilidades que ofrece CSS en relación a los textos. Un aspecto importante de los textos es el tipo de letra que usan y cómo se configura la letra. Hay distintas propiedades para configurar la tipografía, entre ellas font-family, font-variant y la propiedad shortand font.

**PROPIEDAD FONT-FAMILY**

Fuente (font) suele usarse como sinónimo de letra. Así, tamaño de fuente significa tamaño de letra y tipo de fuente significa tipo de letra. Hay tipos de fuentes muy conocidos como Arial, Times New Roman, Calibri…pero la realidad es que existen cientos o miles de tipos de letras.

Las fuentes disponibles en un computador se corresponden con archivos donde se encuentra la información de dicha fuente. Por ejemplo en Windows podemos usar la fuente arial porque existe un archivo denominado arial.ttf en el sistema. El tipo (formato o extensión) del archivo puede variar de unos sistemas a otros: extensiones habituales son ttf, woff, svg, etc.

El tipo de letra que usa una página web ha sido históricamente un problema debido a que los intentos de usar determinados tipos de fuentes chocaban con que los navegadores empleaban fuentes que el usuario tuviera disponible en su ordenador o dispositivo de visualización. Si el diseñador o programador creaba la página web pensando en un tipo de fuente "X" y el usuario no disponía de dicha fuente en su ordenador, la página no se visualizaba tal y como se había previsto (y en muchas ocasiones los resultados eran desastrosos). La propiedad font-family trató de definir una vía que evitara este tipo de problemas y se ha usado y sigue usando mucho, aunque hoy día se combina con otras soluciones (como @font-face, que veremos más adelante) que tratan de ser la solución definitiva al problema de las fuentes en los desarrollos web.

En desarrollos web se usan los siguientes conceptos:

a) Lista de prioridad: permite definir varias posibilidades respecto a las fuentes a emplear en una página web, de modo que la fuente que se usará es la primera en la lista de prioridad siempre que esté disponible. En caso de no estar disponible la primera fuente en la lista de prioridad, se pasaría a la segunda y así sucesivamente. Si no hubiera disponible ninguna fuente dentro de las definidas en la lista de prioridad, se usaría la fuente por defecto.

b) Familia tipográfica o fuente concreta: alude a una fuente concreta y todas sus variantes (negrita, cursiva, etc.). Una familia tipográfica es por ejemplo "Times New Roman", otra familia tipográfica es "Times" y otra familia tipográfica es "Arial". Para escribir una familia tipográfica en CSS debemos hacerlo entre comillas simples o dobles siempre que el nombre contenga espacios en blanco.

© aprenderaprogramar.com, 2006-2029

c) Familia genérica: alude a un grupo de fuentes de características similares y que se distinguen sólo por pequeñas variaciones entre ellas. CSS permite indicar familias genéricas como mecanismo de seguridad para poder buscar alguna fuente dentro del sistema del usuario que tenga características parecidas a las que desea el diseñador o programador. Sirve por tanto como "mecanismo de seguridad" para poder usar una fuente lo más parecida posible a una deseada. Hay 5 familias genéricas que se usan en CSS: serif (ejemplo: Times), sans-serif (ejemplo: Arial), monospace (ejemplo: Courier) , cursive (ejemplo: Zapfino ) y fantasy (ejemplo: Comic Sans MS).

Normalmente en una lista de prioridad se escriben primero las familias tipográficas y en último lugar las familias genéricas. Ejemplo: Arial, Helvetica, sans-serif; es una lista de prioridad donde los dos primeros nombres aluden a tipos de letra concretos y el último a una familia genérica a usar en caso de no estar disponible ninguna de los tipos concretos indicados.

| PROPIEDAD CSS font-family | |
|---|---|
| **Función de la propiedad** | Permite indicar el tipo de fuente que se debe usar y establecer un orden de prioridad para el caso de no disponibilidad de algunas fuentes. |
| **Valor por defecto** | No existe un valor por defecto, depende del navegador |
| **Aplicable a** | Todos los elementos |
| **Valores posibles para esta propiedad** | Un tipo de fuente concreto |
| | Una familia genérica de fuentes |
| | Una lista de prioridad que puede incluir tipos de fuente concretas o familias genéricas, separadas por comas. |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 { "Trebuchet MS", Helvetica, sans-serif;}<br>.elementoJukeBox { font-family: 'Courier New', Courier, monospace;} |

Aunque se admite especificar sólo un tipo de fuente específico, se recomienda indicar siempre un tipo de fuente genérico para el caso de que no pueda cargarse la fuente específica deseada.

Cuando se definen prioridades, el hecho de que sea posible aplicar el primer criterio especificado no significa que no se vaya a usar el segundo. Por ejemplo, podría existir un símbolo no disponible en la fuente deseada, y en ese caso se recurre a la segunda posibilidad en la lista de prioridades.

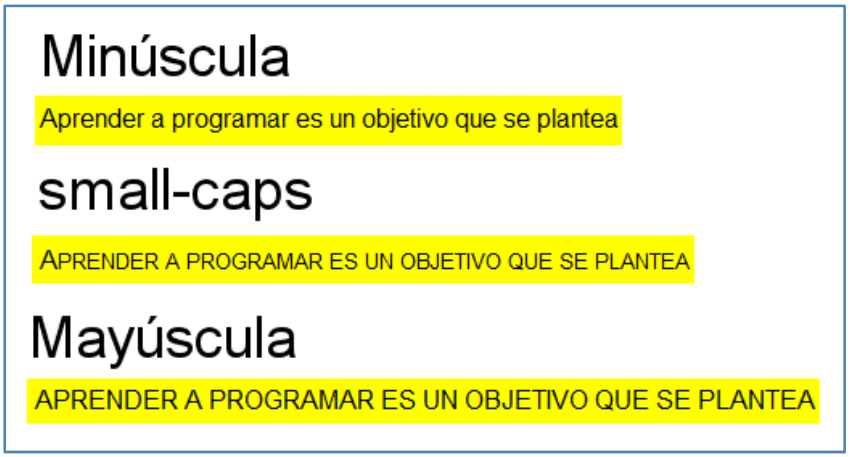
Para crear diseños con tipos de letra que no sean los más comunes habremos de recurrir a una regla especial CSS, @font-face, que estudiaremos más adelante.

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD FONT-VARIANT**

Esta propiedad es una propiedad sencilla que permite poner una fuente en un modo especial denominado de "pequeñas mayúsculas", "versales" o "small-caps". Si usamos text-transform: uppercase; el texto se pondrá en mayúsculas normales. Si usamos font-variant: small-caps; el texto se pondrá en una mayúscula condensada, de un tamaño inferior a la mayúscula normal correspondiente a esa letra.

El valor por defecto para esta propiedad es 'normal' y el único valor alternativo que admite es 'small-caps'. Por ejemplo: p { font-variant:small-caps;}




**PROPIEDAD SHORTAND FONT**

Al igual que con otras propiedades, CSS permite el uso de una propiedad shortand denominada font donde se pueden declarar abreviadamente distintas propiedades relacionadas con las fuentes.

Recordar que hay que tener una precaución con las propiedades shortand: un valor no declarado en la propiedad será sobreescrito a su valor por defecto. Si no tenemos esto en cuenta, podemos estar sobreescribiendo una propiedad sin darnos cuenta.

La sintaxis a emplear es la siguiente:

selectorElemento {Valor-font-styleOpcional Valor-font-variantOpcional Valor-font-weightOpcional ValorFont-sizeObligatorio/ValorLine-heightOpcional valoresListaPrioridad-font-family }

| Ejemplo | Significado |
|---|---|
| **p { font: 12px/ 2em sans-serif }** | El tamaño de letra se establece en 12 pixeles, el interlineado se establece en 2em y el tipo de letra es la familia genérica sans-serif. El resto de propiedades implicadas quedan establecidas a sus valores por defecto. |

© aprenderaprogramar.com, 2006-2029

| Ejemplo | Significado |
|---|---|
| **#content1 { font: bold 1.3em "Trebuchet MS",Arial,Sans-Serif; }** | font-style será negrita (bold), el tamaño de letra 1.3em y se usará prioritariamente la fuente Trebuchet MS. Si no estuviera disponible se usaría Arial. Si no estuviera disponible se usuaría cualquier fuente de tipo Sans-Serif. El resto de propiedades implicadas quedan establecidas a sus valores por defecto. |

**EJERCICIO**

Crea un documento HTML y un archivo con la hoja de estilos CSS que cumpla con estos requisitos:

a) Deben exisitir tres contenedores (div1, div2 y div3) situados uno debajo de otro, cada uno con margin 33px en todas direcciones, con relleno 25 px en todas direcciones, ancho 100% del disponible, y borde sólido de 5 píxeles de anchura con color de borde rojo.

b) El div 1 debe contener un texto suficientemente largo (varios párrafos), tipo de fuente genérica serif, tamaño de fuente 18 píxeles y al menos un párrafo de varias líneas con la fuente establecida como mayúscula condensada (small caps).

c) El div 2 debe contener un texto suficientemente largo (varios párrafos), tipo de fuente genérica sans serif, tamaño de fuente 14 píxeles y color del texto #B22222.

d) El div 3 debe contener un texto suficientemente largo (varios párrafos), y las propiedades de texto establecidas con la propiedad CSS font (shortand). El tipo de fuente será genérica fantasy y el tamaño del texto 2 veces lo normal .

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01046D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**@FONT-FACE CSS**

Hemos comentado las dificultades que históricamente han existido en relación al uso de tipos de letra en páginas web. Esto se debía principalmente a que los navegadores se venían basando en un archivo de fuente instalada en el propio ordenador del usuario y no se podía garantizar que el usuario tuviera disponible la fuente que especificaba el programador o diseñador.

**REGLA @FONT-FACE**

Para superar la limitación de tener que depender de las fuentes disponibles por parte de cada usuario se creó la regla @font-face que permite especificar un nombre y la localización (url) de un archivo con la fuente que se desea usar en una página web. De esta manera, cambia el concepto tradicional, ya que con @font-face podremos usar fuentes que hemos definido y localizado nosotros mismos, sin depender de las fuentes de que disponga el usuario. Ten en cuenta que @font-face está disponible en todos los navegadores modernos, pero que versiones antiguas de navegadores no admiten el uso de esta regla, de ahí que siempre debamos especificar una familia tipográfica genérica como alternativa de seguridad para el caso de que una fuente especificada no pueda cargarse.

Una fuente es un archivo informático con la información necesaria para que un sistema operativo o programa puede mostrar un tipo de letra determinado. Por ejemplo, en Windows se dispone de la fuente "Arial" gracias a que dentro del sistema de archivos existe un archivo denominado arial.ttf. Si borramos ese archivo, la fuente deja de estar disponible. El tipo (formato o extensión) del archivo puede variar de unos sistemas a otros: extensiones habituales son ttf, otf, woff, svg, etc.

La extensión woff (Web Open Font Format) ha sido creada específicamente como tipo de fuente para desarrollos web y aspira a convertirse en un estándar. Sin embargo, la realidad es que no existe todavía un formato de archivo estándar que se haya impuesto para todos los navegadores.

El uso básico de la regla @font-face conlleva definir dos parámetros: por un lado, el "alias" o nombre con el que vamos a designar a la fuente cuando la empleemos en nuestras reglas CSS. Por otro lado, la ruta donde se localiza el archivo que contiene la fuente.

La sintaxis básica a emplear es de este tipo:

```
@font-face {font-family: "nombreDeFuente";    src: url(nombreDeArchivo.extension); }
```

Con esta regla definimos un nombre de fuente p.ej. "burlesque" e indicamos una ruta del archivo donde está disponible esa fuente p.ej. burlesque.woff. En este caso hemos indicado simplemente el

© aprenderaprogramar.com, 2006-2029

nombre del archivo, por lo que la ruta debe ser la misma donde se ubique el archivo css de estilos. Ahora este nombre de fuente puede utilizarse como se haría normalmente con los nombres de fuente en la propiedad font-family, por ejemplo h1 {font-family: burlesque, sans-serif; }

También es posible definir rutas de archivos alojados externamente indicando su url. Por ejemplo:

```
@font-face { font-family: "Bitstream Vera Serif Bold";
src: url(https://mdn.mozillademos.org/files/2468/VeraSeBd.ttf); }
```

Y este nombre de fuente se podría aplicar por ejemplo en body { font-family: "Bitstream Vera Serif Bold", serif }

Tener en cuenta que siempre es recomendable dar al menos una familia genérica para indicar el tipo de fuente que se debería usar en el caso de que no se pueda cargar la fuente deseada.

Debido a que la regla @font-face se ha ido introduciendo en los últimos años y los diferentes navegadores han ido admitiendo formatos de fuentes distintos poco a poco, un "truco" al que recurren muchos diseñadores y programadores para garantizar que la mayor parte de navegadores usen la fuente deseada es especificar las rutas de los archivos fuente en distintos formatos, de modo que se usará el primero en ser reconocido por el navegador. Por ejemplo:

```
@font-face { font-family: Sansat;
src: url('Sansation_Light.woff'), url('Sansation_Light.ttf'), url('Sansation_Light.eot') ; }
```

La regla @font-face permite que se especifique si se debe buscar la fuente primero en el ordenador del usuario y usarla en caso de estar disponible, antes que usar una fuente especificada mediante un archivo. Para ello se especifica el origen como local("nombreDeFuenteEnSistemaUsuario"). Esto es lo que se hace en este ejemplo:

```
@font-face { font-family: Sansat; src: local("Sansation Light"),
url('Sansation_Light.woff'), url('Sansation_Light.ttf'), url('Sansation_Light.eot') ; }
```

Se podrían especificar varios nombres de fuentes locales separados por comas si se desea.

**LAS MEJORES FUENTES WEB GRATUITAS**

Existen diferentes páginas web donde se ofrecen archivos de fuentes gratuitos. Es difícil decir cuál es la página que ofrece las mejores fuentes, entre otras cosas porque en estas páginas hay cambios contínuos y se van incorporando nuevas fuentes y nuevos formatos continuamente. A continuación

© aprenderaprogramar.com, 2006-2029

damos una relación de sitios que permiten el uso de fuentes gratuitas, sin recomendar ninguno en concreto. Pruébalos y usa el que más te guste.

| Nombre | URL | Comentarios |
|---|---|---|
| **Font Squirrel** | http://www.fontsquirrel.com/ | Colección de fuentes no sólo para desarrollos web. Archivos de descarga disponibles en determinados formatos. |
| **Da Font** | http://www.dafont.com/ | Colección de fuentes con clasificación por temáticas. |
| **Google fonts** | http://www.google.com/fonts | Servicio de google que ofrece fuentes gratuitas. Google ofrece una url que permite enlazar al archivo de fuente alojado en sus servidores, o también la descarga de la fuente. |

**CONVERTIDORES DE FUENTES ONLINE**

Muchas veces las fuentes que tenemos disponibles para descarga están en un solo formato como ttf y para asegurarnos de que se visualicen bien deseamos disponer de ellas en varios formatos. Los formatos más usados son: WOFF, EOT, TTF, OTF y SVG.

En principio usando TTF y EOT nos garantizamos que nuestra fuente sea reconocida por la mayoría de los navegadores, aunque si buscamos el máximo de seguridad convendría incluir los cinco formatos.

Existen distintos convertidores on-line, entre los cuales el de Font Squirrel es quizás el más usado y el que mejores críticas recibe.

| Nombre | URL | Comentarios |
|---|---|---|
| **Font Squirrel converter** | http://www.fontsquirrel.com/tools/webfont-generator | Permite subir un archivo en un formato y descargar un archivo zip con la fuente en diferentes formatos incluyendo woff, eot, ttf y svg. |
| **Online Font Converter** | http://onlinefontconverter.com/ | Admite la conversión entre distintos formatos. |
| **Files conversion** | http://www.files-conversion.com/font-converter.php | Admite la conversión entre distintos formatos. |
| **Font converter** | http://www.fontconverter.org/ | Admite la conversión entre distintos formatos. |

© aprenderaprogramar.com, 2006-2029

Ten en cuenta que al realizar la conversión la fuente convertida será muy similar a la original, pero quizás no exactamente igual (pueden aparecer pequeñas diferencias en el espacio entre letras por ejemplo). Recomendación: no trates de cuadrar una página web "al milímetro". Considera siempre que puede ocurrir que la fuente que se cargue no sea exactamente la que tú habías previsto.

**PROBLEMAS CON @FONT-FACE**

La regla @font-face se ha introducido en los últimos años y su estandarización todavía no es suficiente, aunque su uso es cada vez más amplio. Te podrás encontrar con situaciones como que:

a) Un determinado navegador no reconozca la regla @font-face, con lo cual recurrirá a la fuente estándar que hayas definido, o si no la has definido, a la fuente por defecto.
b) Un determinado navegador no reconozca el formato de la fuente que hayas indicado.
c) Un determinado navegador no responda bien a la sintaxis que hemos descrito y sea necesario aplicar "un parche" o código específico para ese navegador.
d) Algunas reglas CSS no se aplican bien con un determinado formato de fuente. Por ejemplo, quizás te encuentres que al aplicar text-transform: uppercase; con algunas fuentes no te funcione.

Para muchos de los problemas que genera font-face existen soluciones específicas o parches de los que no vamos a hablar porque supondría entrar en detalles muy concretos y no es ese nuestro objetivo. Recuerda que el objetivo de este curso es conocer la lógica de CSS y sus aspectos más importantes, sin entrar en detalles relativos a navegadores concretos.

Algunos programadores o diseñadores trabajan con fórmulas alternativas a font-face (normalmente scripts escritos en algún lenguaje de programación) buscando evitar los problemas que les surgen.

**EJERCICIO EJEMPLO RESUELTO DE USO DE @FONT-FACE**

Usando el código HTML de base que venimos empleando a lo largo del curso, y usando la regla @font-face, definir tres tipos de letra distintos (obtenidos desde alguna de las fuentes web gratuitas que hemos indicado) para las etiquetas h1, h2 y p. Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**SOLUCIÓN**

Nosotros vamos a usar las siguientes fuentes:

Títulos h1: fuente "Bubblegum Sans". Nos hemos descargado la fuente en formato otf (BubblegumSans-Regular.otf) desde fontsquirrel.com

© aprenderaprogramar.com, 2006-2029

Títulos h2: fuente "Just Gotta Smile". Nos hemos descargado la fuente en formato ttf (Just Gotta Smile.ttf) desde dafont.com.

Párrafos p: fuente "Alef". Nos hemos descargado la fuente en formato ttf (Alef-Regular.ttf) desde google.com/fonts

Si quieres usar otras fuentes no hay problema.

Junto al archivo css de definición de estilos debemos colocar los archivos de las fuentes (los tres archivos que hemos citado, en nuestro caso de extensión otf, ttf y ttf). El código básico que resuelve el ejercicio y el resultado a obtener sería el siguiente (escribe el código y visualiza el resultado; si no obtienes el resultado previsto sigue leyendo):

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial; background-color:yellow;}
@font-face { font-family: "bubblegum"; src: url('BubblegumSans-Regular.otf'); }
@font-face { font-family: "Just Gotta Smile"; src: url('Just Gotta Smile.ttf'); }
@font-face { font-family: "Alef-Regular"; src: url('Alef-Regular.ttf'); }
h1 {font-family: "bubblegum", arial; font-size: 42px; }
h2 {font-family: "Just Gotta Smile", arial; font-size: 30px; }
p {font-family: "Alef-Regular", arial;  }
p a {font-family: "Alef-Regular", arial;}
```

Nota: hemos puesto un fondo amarillo y letra arial para los elementos sin letra definida; además hemos dotado de un tamaño de letra adecuado a los elementos h1 y h2. También hemos definido el tipo de letra específicamente para los elementos a (links) dentro de párrafos porque éstos no heredan el tipo de letra definido.

¿Qué problema más común nos podemos encontrar con este código? Que algunos navegadores no reconozcan el formato de la fuente. Para tener mayor seguridad en cuanto a que los navegadores apliquen la fuente deseada vamos a generar las fuentes en los formatos más comunes.

© aprenderaprogramar.com, 2006-2029

En nuestro caso accederemos a un servicio web gratuito para convertir fuentes, en concreto vamos a usar http://www.fontsquirrel.com/tools/webfont-generator y subimos los ficheros fuente que tenemos (BubblegumSans-Regular.otf, Just Gotta Smile.ttf y Alef-Regular.ttf) y nos descargamos el archivo zip con las fuentes convertidas a distintos formatos.

Ahora para cada fuente tendremos cuatro archivos, cada uno de los cuales corresponde a una extensión. Por ejemplo para BubblegumSans-Regular ahora tendremos cuatro archivos como bubblegumsans-regular-webfont.woff, bubblegumsans-regular-webfont.ttf, bubblegumsans-regular-webfont.eot, bubblegumsans-regular-webfont.svg. Para las otras fuentes lo mismo, con lo cual en total tendremos 12 archivos.

Ahora escribimos el código indicando que en caso de no reconocerse un formato, se utilice el siguiente disponible. Ten en cuenta que debes colocar los 12 archivos con las fuentes en la misma ubicación en la que tengas el archivo css de estilos. El código quedaría así:

```
/* Curso CSS estilos aprenderaprogramar.com*/

*{font-family: arial; background-color:yellow;}

@font-face { font-family: "bubblegum"; src: url('bubblegumsans-regular-webfont.woff'),
url('bubblegumsans-regular-webfont.ttf'), url('bubblegumsans-regular-webfont.eot'), url('bubblegumsans-regular-
webfont.svg'); }

@font-face { font-family: "Just Gotta Smile"; src: url('just_gotta_smile-webfont.woff'),
url('just_gotta_smile-webfont.ttf'), url('just_gotta_smile-webfont.eot'), url('just_gotta_smile-webfont.svg'); }

@font-face { font-family: "Alef-Regular"; src: url('alef-regular-webfont.woff'),
url('alef-regular-webfont.ttf'), url('alef-regular-webfont.eot'), url('alef-regular-webfont.svg'); }

h1 {font-family: "bubblegum", arial; font-size: 42px; }

h2 {font-family: "Just Gotta Smile", arial; font-size: 30px; }

p {font-family: "Alef-Regular", arial; }

p a {font-family: "Alef-Regular", arial;}
```

Con este código tenemos mayores garantías de que la visualización será correcta en la mayor parte de los navegadores. Pero la garantía no es del 100%, de ahí que siempre sea conveniente especificar qué fuente debe mostrarse en caso de no poder cargar la deseada. En nuestro caso hemos indicado “arial”.

**Próxima entrega:** CU01047D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**PSEUDOCLASES PARA LINKS**

Cuando hablamos de selectores nombramos las pseudoclases CSS y dijimos que las pseudoclases permiten identificar ciertos elementos o situaciones dentro de una página web para aplicar un estilo específico a dichos elementos o situaciones. Por ejemplo, la pseudoclase :first-child permite definir el estilo a aplicar al primer elemento de una serie.

CSS define otra serie de selectores denominados pseudoelementos que van precedidos de :: y cuya función es análoga a la de las pseudoclases. También hemos hablado de ellos anteriormente en el curso, por ejemplo ::first-letter nos permite definir un estilo para la primera letra del texto de un elemento.

Vamos a hablar ahora de cómo dotar de estilos a los links en CSS. Para ello es necesario referirnos a algunas pseudoclases que son las que permiten hacer esto.

| Pseudoclase | Aplicable a | Significado |
|---|---|---|
| **:link** | Los links | Se usa para definir estilos para links que no han sido pinchados por el usuario, aunque si no está en el orden adecuado puede sobreescribir otros estilos de links |
| **:visited** | Sólo links que ya han sido pinchados por el usuario | Se usa para definir estilos para links que han sido pinchados por el usuario y así distinguirlos de los no pinchados |
| **:focus** | Cualquier elemento que tiene el foco por estar seleccionado con el ratón o por uso del tabulador | Se usa para definir estilos para elementos que tienen el foco. Un link puede tener el foco. |
| **:hover** | Cualquier elemento que tiene el puntero del ratón encima de él | Se usa para definir estilos específicos para elementos por los que el usuario pasa el ratón por encima, aún sin pinchar en ellos |
| **:active** | Cualquier elemento en el momento de ser activado | Se usar para definir estilos específicos para links <a> o botones <button> en el justo momento en que son pulsados. |

La sintaxis a emplear es la que ya conocemos: tipoElemento:nombrePseudoclase { ...}

© aprenderaprogramar.com, 2006-2029

A la hora de aplicar estas pseudoclases es importante el orden en que se aplican, ya que si no se hace correctamente se pueden anular estilos definidos. Por ejemplo:

a:hover {color: orange;}

a:link {text-decoration:none; font-weight: bold; color:blue;}

Podríamos pensar que los links sobre los que se situara el ratón se mostrarían naranjas, pero no va a ser así. Se mostrarán azules porque a:link sobreescribe lo definido en a:hover porque :link afecta a todos los links (incluso los que tienen el ratón encima de ellos) mientras que :hover sólo afecta a determinados links. Por ello si situamos :link después de :hover estaremos anulando lo definido en :hover.

Para evitar esto se usa una regla mnemotécnica: LVHA (link-visited, hover, active; es conveniente recordar y aplicar este orden para evitar sobreescribir indebidamente. Podemos usar una frase como "Llegamos vivos hasta amanecer" o "Luego vuelvo hasta allí para acordarnos de este orden.

La pseudoclase :focus se suele colocar justo antes de :hover.

Veremos ejemplos de uso de estas propiedades a continuación.

## PROPIEDAD OUTLINE CSS

La propiedad outline CSS es una propiedad shortand análoga a la propiedad border. De hecho, lo que hace es crear un contorno similar a un borde, con la diferencia de que no ocupa espacio. Su uso suele ser para marcar los contornos de un elemento (muchas veces lo usan los diseñadores y programadores para ver las cajas del modelo de cajas mientras están realizando el diseño).

Al igual que con los bordes existen propiedades individuales que son: outline-style, outline-widht y outline-color, pero en general será preferible usar la propiedad shortand.

Dado que esta propiedad es completamente análoga a la propiedad border, te remitimos a leer el apartado de bordes si quieres profundizar en su manejo. En el código del ejemplo que vemos a continuación puedes ver un ejemplo de aplicación de esta propiedad.

## EJEMPLO DE APLICACIÓN

Usando el código HTML de base que venimos empleando a lo largo del curso, escribe un archivo de estilos css con este código:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: sans-serif; }
a:link {text-decoration:none; font-weight: bold; color:blue; }
a:visited {color: #6D006D;}
a:focus {outline: dashed thin red;}
a:hover {color: orange;}
a:active {font-style: italic;}
li {margin:5px;}
li a:link{border-bottom:solid #B22222 2px; font-weight: bold; color:#B22222;}
```

© aprenderaprogramar.com, 2006-2029

En la imagen anterior vemos el resultado de aplicar el código css indicado. En naranja el aspecto de un link sobre el que tuviéramos situado el ratón encima. Abajo en azul, la línea punteada roja indica que el link tiene el foco. El color azul con fuente en negrita es el que hemos definido para todos los links excepto los que estén dentro de elementos li, que se muestran en la parte superior de otro color.

Hemos eliminado el subrayado por defecto de los links y a cambio hemos usado la propiedad border-bottom. Esto es una práctica que usan muchos diseñadores porque les permite un mayor control sobre los links (evita que queden cortadas las partes inferiores de las letras, permite establecer grosores personalizados, etc.).

Nota: en algunos navegadores puede haber comportamientos extraños o particulares. Por ejemplo que los links visitados definidos con a:visited no respondan y no muestren los estilos definidos a través de la regla css correspondiente. Esto ha sido introducido como opción de seguridad en algunos navegadores para evitar que se pueda rastrear lo que ha hecho el usuario y qué links ha visitado durante su navegación. Podemos modificar las opciones de seguridad en nuestro navegador (en algunos navegadores esto se hace accediendo a Opciones, Privacidad y eligiendo "Recordar mi historial de descargas y navegación"), pero esto sólo afectará a nuestro navegador.

Consejo: acostúmbrate a aceptar que los navegadores tengan distintos comportamientos. Es algo con lo que tendrás que convivir. En algunos casos se podrán aplicar soluciones específicas o "parches" específicos para cada navegador y en otros casos simplemente habrá que aceptarlo así.

**EJERCICIO RESUELTO**

Usando el código HTML de base que venimos empleando a lo largo del curso, se desea realizar lo siguiente:

Para los links dentro de elementos li: establecer su color como #B22222. Estos links deben aparecer sin subrayado y en negrita.Poner la siguiente imagen (que mide 38x38 px) a la izquierda de los links dentro de listas:

© aprenderaprogramar.com, 2006-2029

Para los links dentro de párrafos: establecer su color como azul, formato negrita, sin subrayado. Color para cuando han sido visitados: #6D006D. Color al pasar el ratón encima de ellos: naranja. Poner la siguiente imagen (que mide 38x38 px) a la derecha de los links dentro de párrafos:

Adicionalmente: cuando el usuario ponga el ratón encima de un elemento que lleva el icono rosado la imagen de fondo deberá cambiar y pasar a ser la imagen con el icono azul. Del mismo modo, cuando ponga el ratón encima de un elemento que lleva el icono azul la imagen de fondo deberá cambiar y pasar a ser la imagen con el icono rosado. Es decir, el color del icono debe cambiar al pasar el ratón por encima del link generando un "efecto de cambio".

Nota: eliminar las viñetas de las listas usando esta propiedad: li {list-style-type: none;}

**SOLUCIÓN**

Aquí te planteamos una posible solución (en tu caso tendrás que poner las rutas de imágenes que hayas utilizado). Ten en cuenta que se puede llegar a un resultado de distintas maneras, por tanto tu código podría ser distinto a este código. Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: sans-serif; }

li {list-style-type: none; }

a:link {text-decoration:none; font-weight: bold; color:blue;
background: url('CU01047D_2.png') no-repeat right;
padding-right:38px; display:inline-block; line-height:2.5em;}
a:visited {color: #6D006D;}
a:focus {outline: dashed thin red;}
a:hover {color: orange; background: url('CU01047D_3.png') no-repeat right;}
a:active {font-style: italic;}
li {margin:5px;}
li a:link{text-decoration:none; font-weight: bold; color:#B22222;
background: url('CU01047D_3.png') no-repeat left;
padding-left:46px; line-height:2.5em; display:inline-block; }
li a:hover{background: url('CU01047D_2.png') no-repeat left;}
```

En este código hemos introducido el icono como imagen de fondo y hemos creado tamaño para él introduciendo un padding lateral (right o left según nos interese). A su vez hemos hecho las modificaciones necesarias (márgenes, display, line-height) para conseguir una buena visualización. El resultado que debes obtener debe ser similar a el que mostramos a continuación.

Cuando el usuario sitúe el ratón encima de un link, el icono correspondiente debe cambiar de color.

© aprenderaprogramar.com, 2006-2029

**Próxima entrega:** CU01048D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## APLICAR ESTILOS A LISTAS

Al trabajar con documentos HTML es habitual el uso de listas para realizar enumeraciones. Las listas se clasifican en dos grandes tipos. Por un lado las listas desordenadas o ul (unordered lists) en los cuales el orden no se considera importante. Por otro lado las listas ordenadas, etiqueta <ol> (ordered lists).

Una lista desordenada podría ser por ejemplo de "personas asistentes a la conferencia", donde no hay un primer asistente y un segundo asistente, sino simplemente asistentes. Estas listas es habitual representarlas usando viñetas (elemento gráfico que puede ser un pequeño círculo u otro tipo de elemento como un cuadrado, una estrella, etc.). Ejemplo:

- Juan
- Pedro
- Ismael

Una lista ordenada podría ser por ejemplo "Puntuación obtenida por los alumnos en el examen" donde el alumno con mayor puntuación es el elemento 1º de la lista, el siguiente el elemento 2º, y así sucesivamente. Estas listas es habitual representarlas usando números. Ejemplo:

1. Juan
2. Pedro
3. Ismael

Los navegadores suelen introducir márgenes o paddings por defecto a las listas. En caso de querer eliminar estos valores de defecto deberemos establecer una regla de este tipo:

```
ul {  margin: 0;  padding: 0;  }
```

Sin embargo esto puede hacer que los elementos de la lista se peguen demasiado al borde izquierdo del contenedor. Para hacer que se alineen aproximadamente con los párrafos tal y como los muestra el navegador se puede usar una regla del tipo ul li {  margin-left: 1em;  }

## PROPIEDAD LIST-STYLE-TYPE

| PROPIEDAD CSS list-style-type | |
|---|---|
| **Función de la propiedad** | Permite establecer el tipo de símbolo a mostrar precediendo a los elementos de una lista. |
| **Valor por defecto** | disc (muchos navegadores usan disc para listas desordenadas ul y decimal para listas ordenadas ol) |
| **Aplicable a** | Elementos li de una lista desordenada u ordenada o elementos a los que se haya aplicado la propiedad display: list-item; |

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS list-style-type | |
|---|---|
| **Valores posibles para esta propiedad** | **disc** (se mostrará un círculo relleno, opción por defecto), **circle** (se mostrará un círculo no relleno) ó **square** (mostrará un cuadradito relleno). |
| | **none**: no se mostrará ningún símbolo |
| | **decimal** (números empezando en 1), **decimal-leading-zero** (números de dos dígitos desde 01 hasta 99) |
| | **lower-roman** (números romanos en minúsculas, i, ii, iii, iv...) ó **upper-roman** (números romanos en mayúsculas) |
| | **lower-alpha** (letras minúsculas empezando por la a) ó **upper-alpha** (letras mayúsculas empezando por la A). Mismo efecto con **lower-latin** y **upper-latin**. |
| | Otros valores menos usados: **lower-greek** (letras griegas), **armenian** (numeración armenia), **georgian** (numeración georgiana), etc. |
| | Algunos navegadores incluyen soporte para numeraciones en otros idiomas |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | ul li {list-style-type: square;}<br>li {list-style-type: upper-latin;} |

El color de la viñeta será el mismo que sea aplicable a la lista.

Muchas veces se crean menús de navegación que se basan en listas. En este caso normalmente se prescinde de las viñetas o numeraciones para lo cual se establece esta propiedad con el valor none.

**PROPIEDAD LIST-STYLE-POSITION**

| PROPIEDAD CSS list-style-position | |
|---|---|
| **Función de la propiedad** | Permite controlar la posición de las viñetas. |
| **Valor por defecto** | outside |
| **Aplicable a** | Elementos li de una lista desordenada u ordenada o elementos a los que se haya aplicado la propiedad display: list-item; |
| **Valores posibles para esta propiedad** | outside (la viñeta o numeración aparecerá a la izquierda fuera del bloque de texto que queda a la derecha de la viñeta) |
| | inside (la viñeta o numeración aparecerá como si fuera parte del texto al comienzo de la primera línea) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myList li {list-style-position: inside;}<br>.sandList li {list-style-position: outside;} |

© aprenderaprogramar.com, 2006-2029

Si hay poco texto puede que no se aprecie la diferencia en esta propiedad. Sólo cuando tenemos varias líneas se puede comprobar el distinto efecto que se genera. Ejemplo:

- En este ejemplo tenemos la viñeta outside, la viñeta está a la izquierda y el bloque de texto a la derecha. En este ejemplo tenemos la viñeta outside, la viñeta está a la izquierda y el bloque de texto a la derecha.

- En este ejemplo tenemos la viñeta inside, la viñeta está a la izquierda y el bloque de texto se alinea con la viñeta como si ésta fuera texto. En este ejemplo tenemos la viñeta inside, la viñeta está a la izquierda y el bloque de texto se alinea con la viñeta como si ésta fuera texto.

**PROPIEDAD LIST-STYLE-IMAGE**

| PROPIEDAD CSS list-style-image | |
|---|---|
| **Función de la propiedad** | Permite definir una imagen específica para ser usada en lugar de un símbolo tipo viñeta de entre los predefinidos para listas. |
| **Valor por defecto** | none |
| **Aplicable a** | Elementos li de una lista desordenada u ordenada o elementos a los que se haya aplicado la propiedad display: list-item; |
| **Valores posibles para esta propiedad** | none (no se aplica una imagen a modo de viñeta) |
| | url (rutaDeLaImagenDeseada) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myList li {list-style-image: url(CU01048D_1.png);}<br>.sandList li { list-style-image: url(CU01048D_1.png);} |

Esta propiedad da poco juego a la hora de crear efectos adicionales, modificar tamaños, posiciones, etc. por lo que muchos programadores y diseñadores prefieren colocar una imagen a modo de viñeta usando una imagen de fondo y anulando la viñeta propiamente dicha en lugar de usando esta propiedad.

**PROPIEDAD SHORTAND LIST-STYLE**

Al igual que con muchas otras propiedades, CSS permite definir una propiedad shortand o abreviada para aplicar estilos a listas. La sintaxis básica a emplear es de este tipo:

```
selectorElemento {Valor-list-style-type valor-list-style-position valor-list-style-image }
```

© aprenderaprogramar.com, 2006-2029

Las propiedades se pueden indicar en cualquier orden. Ejemplos:

ul li {list-style: square inside; }

ul li {list-style: square outside url(http://i.imgur.com/afC0L.jpg) ; }

¿Para qué definir un tipo de viñeta si definimos una imagen? Al definir una imagen, esta será la que se muestre siempre que sea posible cargarla. Si no es posible cargarla, se mostrará el tipo de viñeta que hayamos especificado (o si no hemos especificado ninguna, la viñeta que aplique el navegador por defecto).

**EJERCICIO**

Crea una lista ul con 27 elementos li (puedes partir del menú que tenemos en el código HTML de base que venimos empleando a lo largo del curso si lo deseas). Sobre dicha lista aplica los siguientes estilos a los elementos de la lista: los tres primeros elementos tipo disc y outside sin usar la propiedad shortand, los tres siguientes tipo circle e inside sin usar la propiedad shortand, los tres siguientes tipo square e inside, los tres siguientes none, los tres siguientes decimal y outside, los tres siguientes decimal-leading-zero y outside, los tres siguientes lower-roman e inside, los tres siguientes upper-alpha e inside, y los tres últimos con una imagen mediante list-style-image. Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01049D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## MENÚ VERTICAL Y MENÚ HORIZONTAL CSS

A partir de listas de tipo ul es habitual crear menús de navegación en páginas web donde cada elemento del menú se corresponde con un elemento de la lista. Para ello hemos de crear reglas css por un lado para eliminar los estilos por defecto de la lista y por otro para generar la apariencia de menú.

Para ver un ejemplo de aplicación usaremos el código HTML de base que venimos empleando a lo largo del curso, y en concreto el fragmento siguiente:

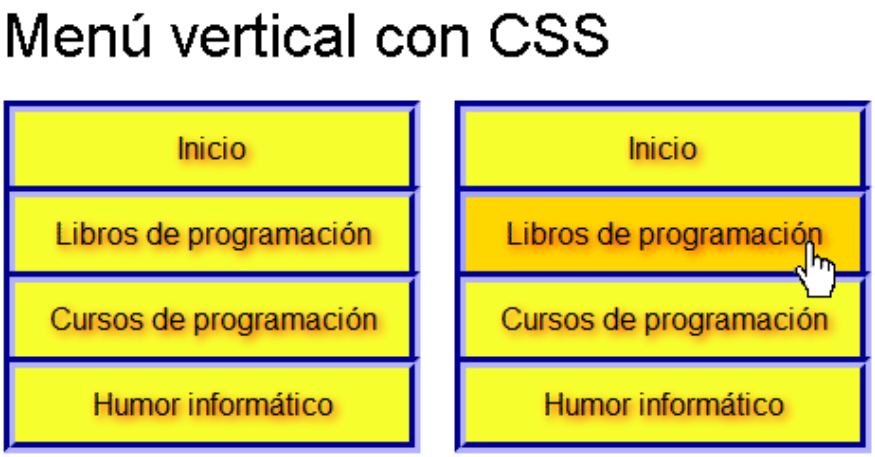
```
<!-- menu -->
<div>
<div>Menú</div>
<hr/>
<ul>
<li><a href="#">Inicio</a></li>
<li> <a href="libros.html">Libros de programación</a> </li>
<li> <a href="cursos.html">Cursos de programación</a> </li>
<li> <a href="humor.html">Humor informático</a> </li>
</ul>
</div>
<!-- fin menu -->
```

Dotaremos de definición de clase al menú así: <ul class="menuVert1">

Y crearemos los estilos necesarios para visualizar la lista como un menú vertical. Escribe este código y comprueba los resultados:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: sans-serif; }

hr{display:none;}

ul.menuVert1 {width: 220px; list-style: none;
 text-align: center; text-shadow: 2px 2px 4px red; padding:0;}

ul.menuVert1 li {padding: 10px 0px; border: groove blue;
border-width: 6px 6px 0px 6px; background-color: #F7FE2E;}

ul.menuVert1 li:last-child { border-bottom: groove blue; border-bottom-width: 6px;}
ul.menuVert1 li a:link, ul.menuVert1 li a:visited { text-decoration: none; color:black;}
ul.menuVert1 li:hover {background-color: #FFD700;}
```

© aprenderaprogramar.com, 2006-2029




Todos los efectos que hemos aplicado en este código han sido explicados durante el desarrollo del curso, por lo que deberías ser capaz de interpretar todas las reglas que hemos definido.

**MENÚ HORIZONTAL CSS**

Vamos a crear un menú horizontal a partir del anterior de la siguiente manera.

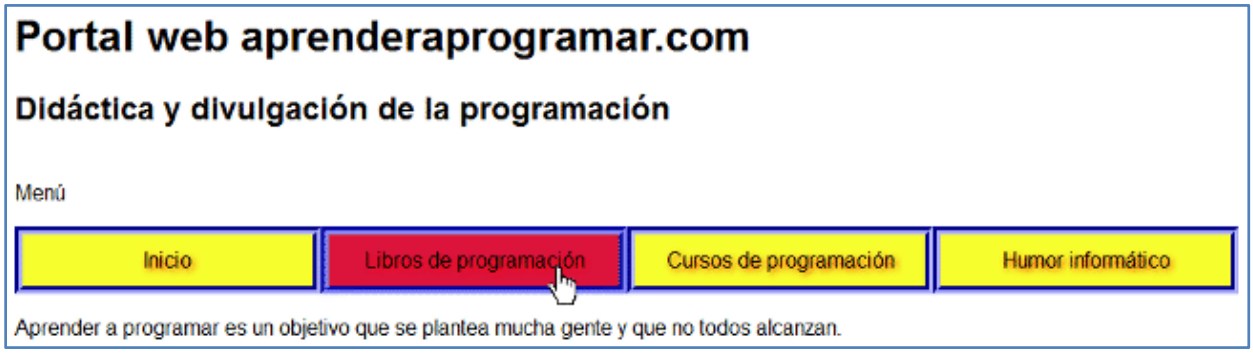
a) En el código HTML introducimos esta línea después del cierre del elemento ul del menú: 

b) Definimos el código CSS de la siguiente manera:

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: sans-serif; }

hr{display:none;}

ul.menuVert1 { list-style: none;  text-align: center; text-shadow: 2px 2px 4px red; padding:0; }

ul.menuVert1 li {width: 220px; padding: 10px 0px; border: groove blue;
border-width: 6px 6px 6px 6px; background-color: #F7FE2E; float:left;}

ul.menuVert1 li a:link, ul.menuVert1 li a:visited { text-decoration: none; color:black;}
ul.menuVert1 li:hover {background-color: #DC143C;}

.limpiador{padding:0; border-style:none; clear:both; }
```

El motivo para introducir un elemento limpiador (y la existencia de alternativas a este procedimiento) ya la hemos discutido anteriormente. El resultado a obtener será similar a este.

© aprenderaprogramar.com, 2006-2029




**USAR MENÚS HORIZONTALES PREDISEÑADOS**

Cuando estemos creando una web es posible que nos den un diseño en formato de "imagen" creado por un diseñador gráfico que tengamos que transformar en un diseño web. También es posible que tengamos que crear una página web construyendo nosotros mismos un diseño.

En el área de CSS existe una gran cantidad de código que ya ha sido creada por otros programadores y diseñadores y que tenemos a nuestra disposición en internet. En general será más útil usar algo ya disponible que crear un diseño desde cero (aunque depende de la situación y de cada caso). Para hacer uso de código predefinido únicamente debemos elegir páginas web de diseñadores que ofrecen su código gratuitamente y que nos resulten útiles o nos gusten.

Vamos a mostrar un código que hemos adaptado desde un código ofrecido por un diseñador a través de internet para crear un menú horizontal con aspecto de solapas y vamos a tratar de interpretarlo.

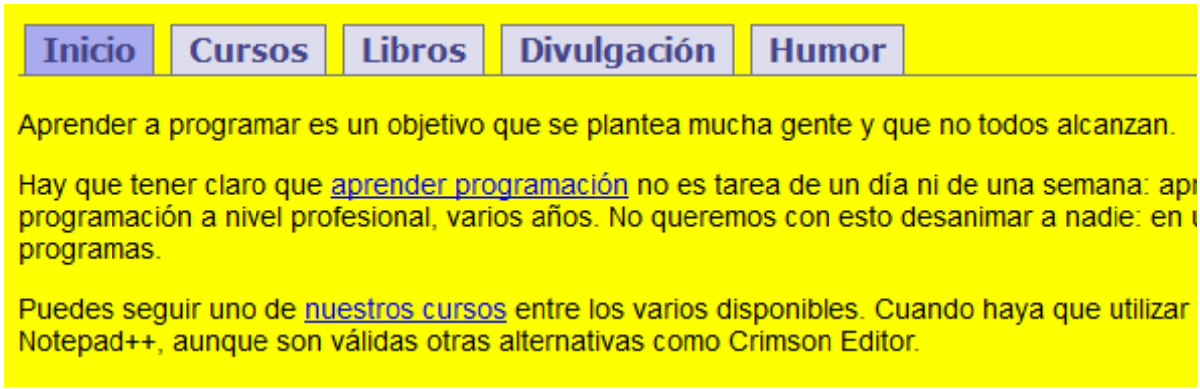
En primer lugar el código HTML:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilos2.css">
</head>
<body>
<div id="navcontainer">
<ul id="navlist">
<li id="active"><a href="http://aprenderaprogramar.com" id="current">Inicio</a></li>
<li><a href="http://aprenderaprogramar.com">Cursos</a></li>
<li><a href="http://aprenderaprogramar.com">Libros</a></li>
<li><a href="http://aprenderaprogramar.com">Divulgación</a></li>
<li><a href="http://aprenderaprogramar.com">Humor</a></li>
</ul>
</div>
</body>
</html>
```

Y a continuación el código CSS (define correctamente el nombre de archivo) y el resultado de visualización:

© aprenderaprogramar.com, 2006-2029

```
body {font-family: sans-serif; background-color: yellow; font-size: 14px;}

#navlist { padding: 3px 0px;
border-bottom: 1px solid #778;
font: bold 16px Verdana, sans-serif; }

#navlist li { list-style: none; display: inline; margin-left:10px;}

#navlist li a { padding: 3px 1em;
border: 1px solid #778; border-bottom: none;
background: #DDE; text-decoration: none; }

#navlist li a:link { color: #448; }
#navlist li a:hover {color: #000; background: #AAE; border-color: #227; }
#navlist li a#current { background: #AAE; }
```




Vamos a proceder a interpretar el código. El código HTML de la parte del menú, esquematizadamente, es el siguiente:

: define el contenedor para alojar el menú como con id “navcontainer”

<ul id="navlist">: define la lista que constituye el menú como con id “navlist”

<li><a href="..." id="current">..</a></li>: define un elemento a con id “current

<li>..</li>: se crean varios elementos de lista dentro de ul y dentro de navcontainer

</ul> : se cierran los elementos

En el código CSS encontramos lo siguiente:

body {..}: define el color de fondo, tipo de fuente y tamaño de fuente

#navlist { padding: 3px 0; margin-left: 0; border-bottom: 1px solid #778; font: bold 16px Verdana, sans-serif; }: lo más importante es el uso de border-bottom para crear una línea horizontal inferior al contenedor a todo lo ancho de la página. El padding top y bottom se establece en 3px lo cual desplaza el borde hacia abajo.

© aprenderaprogramar.com, 2006-2029

#navlist li { list-style: none; margin: 0; display: inline; margin-left:10px;}: elimina los símbolos de viñeta de lista y hace que los elementos de lista se muestren inline (uno a continuación de otro en la misma línea) en lugar de como elementos block, con una separación entre ellos.

#navlist li a { padding: 3px 0.5em; border: 1px solid #778; border-bottom: none; background: #DDE; text-decoration: none; }: cada link dentro de un elemento de lista lleva un relleno (al ser de 3px hacia abajo, la parte inferior de este relleno coincide con la línea de borde que hemos colocado anteriormente. También se crean los bordes eliminando el inferior (que ya está construido mediante la línea de borde a todo lo ancho), se pone un color de fondo y se elimina el subrayado por defecto de los link.

#navlist li a:link { color: #448; }: establece el color de texto para los links dentro de elementos li del menú.

#navlist li a:hover {color: #000; background: #AAE; border-color: #227;}: establece que cuando el ratón se sitúa sobre un elemento de lista el texto se muestre negro, cambie su color de fondo y su color de borde.

#navlist li a#current { background: #AAE; }: establece un color de fondo distinto para el elemento que está identificado con el id "current".

Nota: en un ejemplo de menú horizontal anterior hemos usado float:left; mientras que en este ejemplo hemos usado display:inline; ¿Por qué? No es obligatorio hacer las cosas de una determinada manera, por eso usamos una u otra forma para ver distintos ejemplos de formas de trabajar. Con los conocimientos que ya tenemos, debemos pensar que hay una diferencia importante entre usar float left y usar display inline: cuando los elementos float no caben en el espacio disponible se colocan debajo creando una nueva línea. En cambio los elementos inline se siguen manteniendo siempre en una línea. En muchos casos usar float será más ventajoso para poder añadir tantos elementos de menú como se deseen sin preocuparnos de si caben en el espacio disponible o no.

**EJERCICIO**

Analiza el código HTML y CSS que mostramos a continuación y realiza una interpretación descriptiva del código, explicando el significado de cada una de sus partes (Nota: como referencia, puedes ver cómo se ha hecho la interpretación descriptiva del código de ejemplo que hemos visto anteriormente).

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

© aprenderaprogramar.com, 2006-2029

```
<html><head><title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
/* Curso CSS aprenderaprogramar.com */
#nav-menu ul {font-family: sans-serif; list-style: none;
padding: 0;margin: 0; }
#nav-menu li {float: left;margin: 0 0.15em; border: 5px groove #FFAA33 ;}
#nav-menu li a { background-color: #FFC0CB; height: 2em; line-height: 2em; float: left;
width: 9em; display: block; border: 0.1em solid #dcdce9; color: #0d2474; text-decoration: none;
text-align: center;}
#nav-menu li a:hover {background-color: #FF6347; }
</style></head>
<body><div id="nav-menu">
<ul id="navlist"><li id="active"><a href="http://aprenderaprogramar.com" id="current">Inicio</a></li>
<li><a href="http://aprenderaprogramar.com">Cursos</a></li>
<li><a href="http://aprenderaprogramar.com">Libros</a></li>
<li><a href="http://aprenderaprogramar.com">Divulgación</a></li>
<li><a href="http://aprenderaprogramar.com">Humor</a></li>
</ul></div></body></html>
```

**Próxima entrega:** CU01050D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## MENÚ DESPLEGABLE CSS

Aunque en muchas ocasiones la generación de efectos visuales relativamente complejos se hace mediante un lenguaje como javascript, vamos a comprobar como usando HTML y CSS podemos generar un menú desplegable atractivo y con un efecto visual muy interesante.

Vamos a diseñar un menú con 4 items principales. Cada item principal tiene subitems o subelementos. El esquema que define cuáles son los items principales y cuáles los subitems lo definimos a continuación.

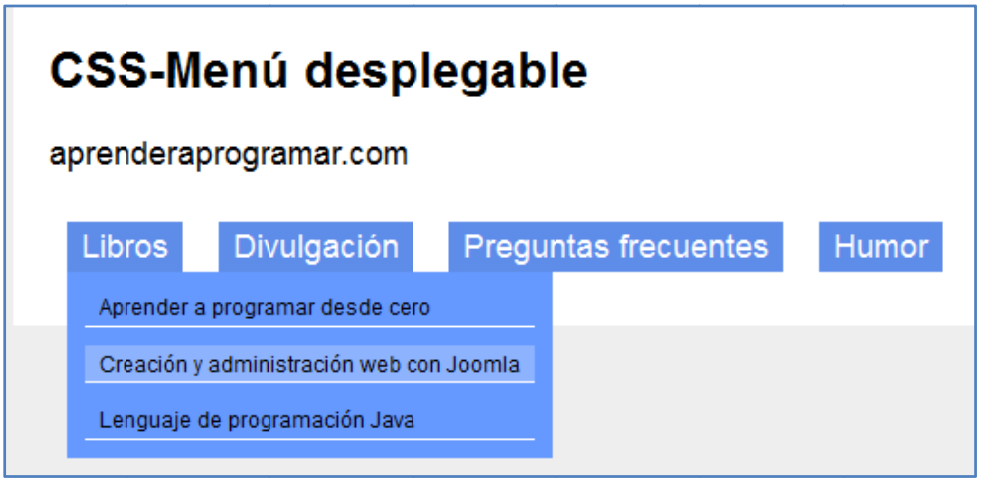
| Item principal | Subitems |
|---|---|
| **Libros** | Aprender a programar desde cero<br>Creación y administración web con Joomla<br>Lenguaje de programación Java |
| **Divulgación** | Los 100 trucos de CSS |
| **Preguntas frecuentes** | |
| **Humor** | Humor informático<br>Humor bases de datos<br>Humor programación<br>Humor universidad |

Podemos comprobar que el primer item de menú o apartado tendrá tres subitems o subapartados. El segundo item tendrá un solo subitem. El tercer item no tendrá ningún subitem. El cuarto item tendrá cuatro subitems.

Cuando la página web cargue inicialmente mostraremos sólo los items principales del menú:

© aprenderaprogramar.com, 2006-2029

Cuando el usuario posicione el puntero del ratón sobre un elemento principal, aparecerá el menú desplegable con los subitems correspondientes. Por ejemplo al posicionarnos sobre el item "Libros" se verán los tres subitems correspondientes y se podrá elegir aquel que se desee:




La idea general es la siguiente. En HTML se definen todos los items y subitems. Los items serán elementos li dentro de una lista ul. Los menús que se despliegan serán elementos ul que contienen elementos li y que a su vez están dentro de los elementos li del menú principal. Puede parecer un poco complicado, pero estudiando el código llegarás a entenderlo fácilmente.

Escribe este código HTML y comprueba su estructura. Hazte un esquema donde indiques qué elementos van quedando dentro de otros. Fíjate en los comentarios que hemos incluido, que te servirán de guía. Ten en cuenta que en el documento HTML no se define qué será visible al cargar la página y qué no será visible, ya que esto lo haremos a través de CSS.

```
<html>
<head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01050D.css">
</head>

<body>
<div id="wrap"> <!--Caja contenedora-->
<h2>CSS-Menú desplegable</h2>
<p>aprenderaprogramar.com</p>
<ul id="navbar"> <!--Lista que define los elementos principales que se muestran siempre-->
<li><a href="http://aprenderaprogramar.com">Libros</a> <!--Item principal de menú-->
<ul> <!--Lista que define subitems de menú que se mostrarán sólo al posicionar el ratón sobre el item principal-->
<li><a href="http://aprenderaprogramar.com">Aprender a programar desde cero</a></li> <!--SubItem de menú-->
<li><a href="http://aprenderaprogramar.com">Creación y administración web con Joomla</a></li> <!--SubItem de menú-->
<li><a href="http://aprenderaprogramar.com">Lenguaje de programación Java</a></li> <!--SubItem de menú-->
</ul>
</li>
<li><a href="http://aprenderaprogramar.com">Divulgación</a> <!--Item principal de menú-->
<ul> <!--Lista que define subitems de menú que se mostrarán sólo al posicionar el ratón sobre el item principal-->
<li><a href="http://aprenderaprogramar.com">Los 100 trucos de CSS</a></li> <!--SubItem de menú-->
</ul>
</li>
<li><a href="http://aprenderaprogramar.com">Preguntas frecuentes</a></li><!--Item principal de menú-->
```

© aprenderaprogramar.com, 2006-2029

```
<li><a href="http://aprenderaprogramar.com">Humor</a> <!--Item principal de menú-->
<ul> <!--Lista que define subitems de menú que se mostrarán sólo al posicionar el ratón sobre el item principal-->
<li><a href="http://aprenderaprogramar.com">Humor informático</a></li> <!--SubItem de menú-->
<li><a href="http://aprenderaprogramar.com">Humor bases de datos</a></li> <!--SubItem de menú-->
<li><a href="http://aprenderaprogramar.com">Humor programación</a></li> <!--SubItem de menú-->
<li><a href="http://aprenderaprogramar.com">Humor universidad</a></li> <!--SubItem de menú-->
</ul>
</li>
</ul>
<div class="limpiador"></div> <!--Explicado en apartados anteriores del curso-->
</div>
</body>
</html>
```

Escribe ahora el código CSS. Con los contenidos que hemos visto a lo largo del curso debes ser capaz de interpretar todas las instrucciones que aparecen en el mismo. Pon el nombre de archivo css adecuado. Ten en cuenta que en algunos navegadores, en especial en los más antiguos, es posible que no obtengas el resultado deseado.

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font: 62.5%/1.2 Arial, Helvetica, sans-serif; background-color: #eee; }
h2 {margin:0;}

/*Caja contenedora*/
#wrap { font-size: 1.8em; width: 520px; padding: 20px;
      margin: 0 auto; /*Da lugar al centrado de la caja en el elemento superior body*/
      background-color: #fff;}

/* Estilos que crean el menú desplegable */
/*Eliminamos padding y margin que introducen navegadores por defecto en listas*/
#navbar { padding:0; margin:0; }

/*Elementos items principales de menú que se muestran siempre*/
#navbar li { list-style: none; float: left; margin:10px;}

#navbar li a {
  display: block; /* Usamos display block para poder aplicar propiedades de caja a links */
  padding: 3px 8px; background-color: #5e8ce9; color: #fff;
  text-decoration: none; }

/*Listas de subitems de menú*/
#navbar li ul {
  display: none; /*La lista inicialmente no se muestra debido a display:none; */
  background-color: #69f;}

#navbar li:hover ul {
  font-size: 12px;
  display: block; /*Al situar el cursor sobre el item la lista de subitems pasa de display none a display block y se muestra*/
  position: absolute; /*Al desplegarse el submenú no ocupa espacio y no desplaza a otros elementos gracias a que establecemos position absolute*/
  margin: 0; padding: 0; /*Anulamos margin y padding que por defecto introducen navegadores*/
  }
```

© aprenderaprogramar.com, 2006-2029

```
#navbar li:hover li { float: none; } /* Anulamos el float left que define el elemento padre para que los subitems se muestren en vertical */

/* Creamos la apariencia de los subitems de menú, color de fondo, borde inferior, color de texto*/
#navbar li:hover li a { background-color: #69f; border-bottom: 1px solid #fff; color: #000; }

/* Creamos el efecto de cambio de color y aspecto cuando ponemos el puntero del ratón sobre un subitem de menú*/
#navbar li li a:hover { background-color: #8db3ff; }

.limpiador{padding:0; border-style:none; clear:both; } /* Forzamos a la caja a mostrarse aún conteniendo elementos flotantes*/
```

Es importante que comprendas todo el código HTML y todo el código CSS que hemos utilizado.

Señalaremos algunos aspectos principales que se usan para generar el efecto de menú desplegable:

a) Las listas de subitems no se muestran inicialmente porque tienen establecida la propiedad display como "none". Hacemos que se muestren cuando el usuario pone el ratón encima de un item principal indicando que la lista hija de ese elemento pase a tener su propiedad display como "block".

b) Las listas de subitems no desplazan a otros elementos porque establecemos su propiedad position como "absolute". El valor absolute permite que un elemento se desplace respecto al origen de coordenadas del primer elemento contenedor posicionado ó respecto a la esquina superior izquierda de la ventana de visualización; el resto de elementos actúan como si el elemento con position absolute no existiera, por lo que su espacio es ocupado por otros elementos. Nosotros no usamos absolute para crear un desplazamiento, pero sí nos aprovechamos de que da lugar a que no se ocupe espacio.

Además en este ejemplo utilizamos otras propiedades como float que ya hemos estudiado anteriormente en el curso y que aquí nos sirven para repasar y reafirmar conocimientos. Si estás siguiendo el curso y tienes dudas, consulta en los foros en http://aprenderaprogramar.com/foros.

Un aspecto que merece la pena comentar es cómo CSS introduce elementos que le permiten realizar tareas asemejables a las de un lenguaje de programación. En concreto, fíjate cómo la lógica del menú desplegable es de tipo condicional, una capacidad propia de los lenguajes de programación. En concreto, se trataría de ejecutar "Si el usuario tiene el ratón apuntando a un item de menú { Mostrar lista desplegable de subitems} sino { Mostrar sólo el item principal }". No lo hemos expresado así, pero la lógica se aproxima a esto. Comentamos en su momento que CSS no es un lenguaje de programación, no obstante se entremezcla con éstos al escribirse embebidos unos lenguajes con otros, o tiene algunas analogías como la que estamos comentando que se asemeja a una operación lógica.

**Próxima entrega:** CU01051D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**ESTILOS PARA TABLAS**

En el pasado las tablas fueron elementos que muchos diseñadores o programadores utilizaban para crear un diseño donde las celdas eran contenedores partes de la web. Hoy en día esto ya no se usa y los diseños se crean usando elementos flotantes o nuevas capacidades de CSS para crear columnas. Sin embargo, las tablas siguen siendo elementos importantes dentro de una web.

Debe quedar claro que no está recomendado el uso de tablas para crear divisiones de maquetación en la web. Por ejemplo un menú no es recomendable colocarlo dentro de una tabla, será preferible usar elementos flotantes para ello. Sin embargo, muchas veces estaremos trabajando en desarrollos web donde nos será necesario introducir tablas simplemente porque se trata de un contenido tabulado. Por contenido tabulado hacemos alusión a una serie de registros que describen los valores de ciertos parámetros que se indican en la cabecera de la tabla. Por ejemplo, una tabla puede servir para describir los nutrientes de los alimentos.

Contenido nutricional por cada 100 g de alimento.

| Alimento | Calorías (kCal) | Grasas (g) | Proteína (g) | Carbohidratos (g) |
|---|---|---|---|---|
| **Arándano** | 49 | 0.2 | 0.4 | 12.7 |
| **Plátano** | 90 | 0.3 | 1.0 | 23.5 |
| **Cereza** | 46 | 0.4 | 0.9 | 10.9 |
| **Fresa** | 37 | 0.5 | 0.8 | 8.3 |

Para representar este tipo de contenidos tabulados usamos tablas y para lograr que la tabla tenga un aspecto estético adecuado aplicamos estilos CSS.

Recordaremos ahora algunos conceptos básicos de HTML:

<table> ...</table> delimita la tabla.

<caption> ...</caption>: elemento opcional que permite poner un título de tabla. Muchas veces no se usa.

<thead> ...<thead>: delimita filas o celdas que no son celdas de datos, sino encabezados de la tabla.

<th> ...</th>: delimita una celda de encabezado (table header). A veces todas las celdas de una tabla se definen como celdas normales y se prescinde del uso de th.

<tbody> ...<tbody>: delimita las filas y celdas de datos de una tabla.

<tr> ...</tr>: delimita una fila (table row).

<td> ...</td>: delimita una celda normal de datos (table data cell).

<tfoot> ...</tfoot>: delimita el pie de tabla.

© aprenderaprogramar.com, 2006-2029

Vamos a utilizar el siguiente código HTML para trabajar los estilos de tablas. Escribe el código en un editor y guarda el documento HTML.

```
<html>
<head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01051D.css">
</head>
<body>
<table >
<caption>Contenido nutricional por cada 100 g de alimento.</caption>
<tr> <th>Alimento</th> <th>Calorías (kCal)</th> <th>Grasas (g)</th>
<th>Proteína (g)</th> <th>Carbohidratos (g)</th>
</tr>
<tr> <td>Arándano</td> <td>49</td> <td>0.2</td>
<td>0.4</td> <td>12.7</td>
</tr>
<tr> <td>Plátano</td> <td>90</td> <td>0.3</td>
<td>1.0</td> <td>23.5</td>
</tr>
<tr> <td>Cereza</td> <td>46</td> <td>0.4</td>
<td>0.9</td> <td>10.9</td>
</tr>
<tr> <td>Fresa</td> <td>37</td> <td>0.5</td>
<td>0.8</td> <td>8.3</td>
</tr>
</table>
</body>
</html>
```

Por tradición era (y en cierta medida sigue siendo) muy habitual usar atributos de la etiqueta HTML table para definir los bordes y espaciado en las tablas, por ejemplo <table border="1" cellpadding="14" cellspacing="0">

Estos atributos no los usaremos en nuestros desarrollos web, ya que como venimos indicando a lo largo de todo el curso debemos separar la estructura del documento (html) de su apariencia (css).

Si la tabla no tiene definido ningún estilo se mostrará como por defecto la muestre el navegador, en general sin bordes. Para introducir bordes tenemos que definir el borde de la tabla (borde externo) y el borde de las celdas (bordes internos). Escribe este código CSS, guárdalo en un archivo css con el nombre adecuado, y comprueba la diferencia entre visualizar la tabla sin estilos o con este estilo básico:

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial, Helvetica, sans-serif; }
table, th, td {border: 1px solid #000;}
```

Podemos comprobar que al dibujar la tabla se dibuja un borde externo para la tabla en su conjunto y luego un borde interno para cada una de las celdas. El efecto que se genera es un poco extraño al tener la tabla “líneas dobles” de separación. Este efecto se puede anular usando la propiedad border-collapse como veremos a continuación.

© aprenderaprogramar.com, 2006-2029

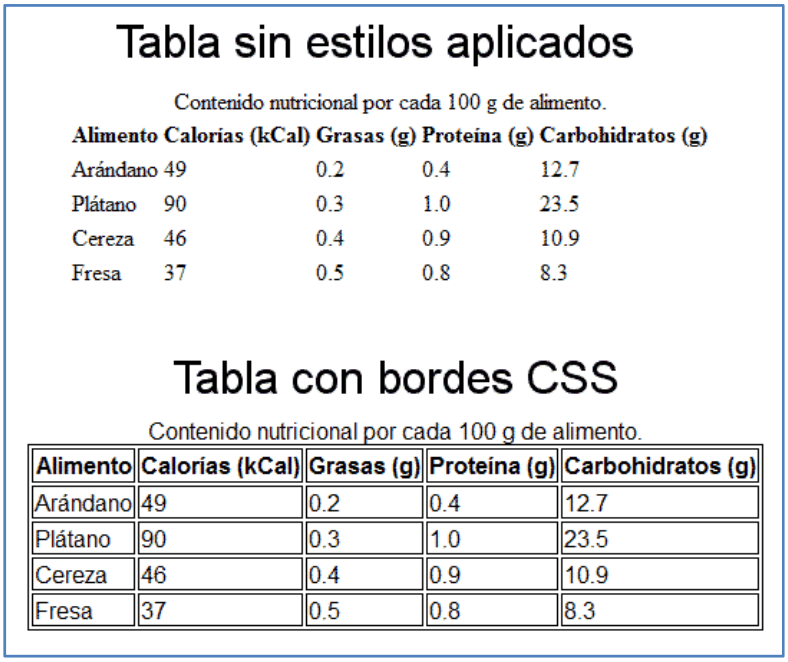


Tabla sin estilos aplicados

Contenido nutricional por cada 100 g de alimento.

| Alimento | Calorías (kCal) | Grasas (g) | Proteína (g) | Carbohidratos (g) |
|---|---|---|---|---|
| Arándano | 49 | 0.2 | 0.4 | 12.7 |
| Plátano | 90 | 0.3 | 1.0 | 23.5 |
| Cereza | 46 | 0.4 | 0.9 | 10.9 |
| Fresa | 37 | 0.5 | 0.8 | 8.3 |

Tabla con bordes CSS

Contenido nutricional por cada 100 g de alimento.

| Alimento | Calorías (kCal) | Grasas (g) | Proteína (g) | Carbohidratos (g) |
|---|---|---|---|---|
| Arándano | 49 | 0.2 | 0.4 | 12.7 |
| Plátano | 90 | 0.3 | 1.0 | 23.5 |
| Cereza | 46 | 0.4 | 0.9 | 10.9 |
| Fresa | 37 | 0.5 | 0.8 | 8.3 |

**ATRIBUTOS QUE NO SE HEREDAN EN TABLAS**

Algunas reglas que normalmente se heredan no son heredadas por las tablas. Por ejemplo si aplicamos una regla como body {font-size: 8px;} podremos comprobar que en algunos navegadores las tablas no heredan este tamaño. Tendremos entonces que especificar estos valores para tablas usando declaraciones como table {font-size: 8px;}. Aunque sean los mismos atributos que los definidos para la etiqueta body, será necesario para evitar que el navegador aplique el valor por defecto que tenga previsto para tablas.

Nota: otra vía por la que se puede conseguir la herencia de algunos atributos como font-size en tablas es añadiendo una declaración en cabecera del documento HTML del tipo <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">. Este tipo de declaraciones pueden dar lugar a que el navegador actúe de una forma u otra (se dice que el navegador utiliza un modo u otro), lo cual nos vuelve a referenciar al problema de las diferentes formas en que los distintos navegadores pueden mostrar un mismo código HTML. En general preferiremos reescribir el atributo y asegurarnos de que el navegador actuará conforme esperamos antes que fiarnos de que todos los navegadores respondan igual ante una declaración en la cabecera del documento HTML.

**ANCHURA Y ALTURA DE TABLAS Y ELEMENTOS DE TABLAS**

La anchura de una tabla será la necesaria para mostrar sus contenidos y cada celda tendrá el ancho necesario, excepto si establecemos indicaciones específicas. Una forma interesante de hacerlo es usando porcentajes. Por ejemplo si establecemos un ancho para la tabla de 550 px y tenemos 5 columnas, podemos indicar que todas las columnas tengan el mismo ancho dividiendo el 100% de ancho de la tabla entre 5 para obtener un 20 %. Aplicaríamos reglas de este tipo: table {width: 550px;} th, td {width: 20%;}

Si la tabla está dentro de otro contenedor y queremos que ocupe el 100% de espacio disponible podemos aplicar reglas del tipo table {width: 100%; }

© aprenderaprogramar.com, 2006-2029

**ATRIBUTOS NO APLICABLES A TABLAS**

Podemos establecer un valor de height para las celdas de tablas pero este valor no será respetado si el contenido de la celda es mayor que el espacio disponible. Si intentamos aplicar un atributo como th, td { overflow: hidden;} comprobaremos que probablemente no funcione. Si necesitamos aplicar un valor height y que éste no sea modificable de ninguna manera, podemos introducir el contenido de las celdas en el HTML dentro de elementos div. Por ejemplo:

```
<td><div>Arándano este texto es muy largo y va a exceder el espacio disponible</div></td>
```

Ahora estableceríamos el control de la altura de los elementos div y el valor de la propiedad overflow que sí es aplicable a los elementos div de forma similar a esta:

```
th, td {text-align: center; padding:0.5em; width:25%; height: 1em; overflow: hidden;}

th div, td div {height: 1em; overflow: hidden; color:red;}
```

Haz pruebas escribiendo código para verificar lo que hemos expuesto.

**PROPIEDAD BORDER-COLLAPSE**

| PROPIEDAD CSS border-collapse | |
|---|---|
| **Función de la propiedad** | Permite controlar si en una tabla cada elemento mantiene un borde propio generando un efecto de "bordes dobles" o si los bordes se fusionan generando un efecto de "borde único". |
| **Valor por defecto** | separate |
| **Aplicable a** | Tablas |
| **Valores posibles para esta propiedad** | separate (cada elemento conserva su borde) |
| | collapse (los bordes se fusionan en un borde único) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myTable {border-collapse: separate;}<br>.myTableSP {border-collapse: collapse;} |

Aunque el valor separate es el valor por defecto para esta propiedad, la mayoría de las veces se aplica el valor collapse porque genera una visualización más sencilla y acorde a las expectativas de los usuarios.

Modifica el código CSS del ejemplo anterior como indicamos a continuación y comprueba el efecto que se produce:

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial, Helvetica, sans-serif; }
table, th, td {border: 1px solid #000; border-collapse: collapse;}
th, td {text-align: center; padding:0.5em;}
```

Comprobarás que ahora los bordes se muestran como bordes únicos.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Crea un documento HTML con tres tablas, todas ellas iguales, con un título de tabla, cinco columnas y 7 filas (la primera ella, fila de encabezados que no son datos propiamente dichos). Aplícale los siguientes estilos y comprueba la visualización obtenida:

a) La tabla 1 tendrá una anchura de 600 píxeles y cada columna tendrá un 20% de la anchura total de la tabla. Existirán bordes únicos de color gris y tendrán un grosor de 8 píxeles.

b) La tabla 2 tendrá una anchura igual al 100 % disponible en la ventana y cada columna tendrá un 20 % de la anchura total de la tabla. Existirán bordes dobles de color marrón y tendrán un grosor de 2 píxeles.

c) La tabla 3 tendrá una anchura de 500 píxeles y cada columna tendrá 100 píxeles de anchura. Sólo existirán bordes en la parte inferior de las filas (es decir, no habrá bordes laterales ni superiores), tipo borde único, con un grosor de 6 píxeles y color azul.

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01052D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## DISEÑO CSS DE TABLAS

Hay una serie de propiedades que nos permiten definir estilos para tablas. Vamos a estudiar las propiedades border-spacing, caption-side y empty-cells. Además abordaremos cómo utilizar los fondos en las tablas, cómo crear filas con colores diferentes y veremos un ejemplo de cómo utilizar estilos creados por otros programadores o diseñadores.

## PROPIEDAD BORDER-SPACING

| PROPIEDAD CSS border-spacing | |
|---|---|
| **Función de la propiedad** | Permite controlar la separación entre bordes dentro de una tabla cuando los bordes no están fusionados con border-collapse. |
| **Valor por defecto** | 0 |
| **Aplicable a** | Tablas sin border-collapse establecido o con border-collapse: separate; |
| **Valores posibles para esta propiedad** | Una unidad de medida válida en CSS (indica separación vertical y horizontal) |
| | Dos unidades de medida separadas con un espacio (la primera unidad indicará separación horizontal y la segunda separación vertical) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myTable {border-spacing: 2em;}<br>.myTableSP {border-spacing: 15px 5px;} |

Esta propiedad es análoga al atributo cellspacing que se usaba tradicionalmente en HTML para establecer una separación entre celdas en las tablas. En el caso de la antigua propiedad cellpadding, se puede obtener un efecto análogo en tablas simplemente usando la propiedad padding.

Si se establece border-spacing: 0; el efecto es similar a definir border-collapse:collapse; aunque en realidad son situaciones distintas. Usando border-spacing:0; los bordes dobles siguen existiendo, lo que ocurre es que están tan juntos que visualmente sólo se aprecia una línea de borde. En este caso el grosor de las líneas de borde será doble del especificado (ya que se juntan dos líneas de borde). Usando border-collapse: collapse; no existen líneas de borde para cada celda, sino que las líneas colindantes se fusionan en una sola línea. En este caso el grosor de las líneas de borde será exactamente el especificado.

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD CAPTION-SIDE**

Es una propiedad que permite colocar el elemento caption en una posición concreta. Los valores permitidos son top, right, left, bottom y hacen que el título establecido en caption se coloque en el lugar especificado. Por defecto el título se coloca en posición top (encima de la tabla). En posiciones top y bottom (debajo de la tabla) el título aparece por defecto centrado horizontalmente. Si queremos establecer una alineación horizontal específica podemos usar text-align como en este ejemplo, que coloca el título abajo a la derecha:

```
table {width: 550px; caption-side:bottom;}

caption {text-align:right;}
```

En el caso de las posiciones left y right por defecto el título se coloca en el lateral, alineado con la parte superior de la tabla. Si queremos centrarlo verticalmente podemos hacerlo como en este ejemplo:

```
table {width: 550px; caption-side:left;}

caption {vertical-align:middle;}
```

Esta propiedad es una propiedad poco usada, aunque en algunos casos concretos puede resultar de interés.

**PROPIEDAD EMPTY-CELLS**

Normalmente las celdas tienen un contenido. Por ejemplo <td>9</td> es una celda con contenido 9. <td> </td> es una celda con un espacio en blanco. No obstante, en algunos casos las celdas pueden estar completamente vacías en forma <td></td>. Esto es bastante frecuente cuando no se tienen algunos datos y resulta imposible rellenar la tabla completamente.

La propiedad empty-cells sirve para definir cómo debe mostrar el navegador las celdas vacías. Tiene dos valores posibles: show (valor por defecto) y hide. Si usamos show (o simplemente si no especificamos esta propiedad) las celdas vacías se mostrarán de la misma manera que las celdas normales. Si usamos hide los bordes y fondo (background) de la celda no se mostrarán

Esta propiedad es una propiedad poco usada, aunque en algunos casos concretos puede resultar de interés.

**FONDOS (BACKGROUND EN TABLAS)**

Se pueden aplicar colores de fondo o imágenes de fondo a las tablas. La única peculiaridad a tener en cuenta es que en una tabla existen distintos elementos y a la hora de aplicar fondos cada elemento queda en una capa inferior o superior a la hora de visualizarse, de modo que un fondo en una capa inferior puede quedar oculto por un fondo establecido en una capa superior. Los fondos que pueden establecerse y el orden en que quedarían ordenados son los siguientes:

1. Fondo del elemento table, es el que se sitúa "más abajo" y será tapado total o parcialmente por otros fondos si se establecen.
2. Fondo de grupos de columnas
3. Fondo de columnas

© aprenderaprogramar.com, 2006-2029

4. Fondo de grupos de filas
5. Fondo de filas
6. Fondo de celdas, que son los que se sitúan "más arriba" y tapan total o parcialmente a los fondos que existan debajo.

## COLORES DE FILAS INTERCALADOS O ALTERNOS

Es frecuente presentar tablas donde los colores de filas estén intercalados o alternos. Esto se puede hacer de forma sencilla con CSS. Una forma de hacerlo sería establecer en el documento HTML una especificación de clase para cada fila. Por ejemplo <tr class="tr_type1">..</tr> alternando con <tr class="tr_type2">..</tr>.

Pero en general será mucho más práctico y sencillo usar los selectores avanzados :nth-child(odd) y :nth-child(even) que ya hemos explicado anteriormente en el curso para conseguir alternar los colores de fondo. Aquí mostramos un código de ejemplo donde aplicamos un color de fondo amarillo a las celdas de cabecera de la tabla y un color gris a las filas impares y un color rosado a las filas pares.

```
table th {background-color: yellow; }

table tr:nth-child(odd) {background-color: grey;}

table tr:nth-child(even) {background-color: pink;}
```

El resultado que se consigue es del tipo que mostramos a continuación.

Contenido nutricional por cada 100 g de alimento.

| Alimento | Calorías (kCal) | Grasas (g) | Proteína (g) | Carbohidratos (g) |
|---|---|---|---|---|
| Arándano | 49 | 0.2 | 0.4 | 12.7 |
| Plátano | 90 | 0.3 | 1.0 | 23.5 |
| Cereza | 46 | 0.4 | 0.9 | 10.9 |
| Fresa | 37 | 0.5 | 0.8 | 8.3 |
| Arándano | 49 | 0.2 | 0.4 | 12.7 |
| Plátano | 90 | 0.3 | 1.0 | 23.5 |
| Cereza | 46 | 0.4 | 0.9 | 10.9 |
| Fresa | 37 | 0.5 | 0.8 | 8.3 |

## USAR DISEÑOS EXISTENTES

Como ya hemos comentado en otras ocasiones, existen muchos diseños atractivos que han sido creados por diseñadores expertos que son puestos a disposición de todos por sus creadores a través de internet. Normalmente será interesante usar estos diseños cuando queramos ahorrar tiempo y nos resulte útil un diseño que se nos proponga. Para encontrar diseños para tablas basta con usar un buscador e introducir una búsqueda como "css table designs" ó "diseños css para tablas". En los resultados, podremos encontrar diferentes propuestas de diseño y de código.

© aprenderaprogramar.com, 2006-2029

A continuación te indicamos el código de un diseño de tabla propuesto por un diseñador. Escribe el código HTML y el código CSS y comprueba el resultado.

```
<html>
<head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="estilosCU01051D.css">
</head>

<body>
<table >
<caption>Contenido nutricional por cada 100 g de alimento.</caption>
<tr> <th>Alimento</th> <th>Calorías (kCal)</th> <th>Grasas (g)</th>
<th>Proteína (g)</th> <th>Carbohidratos (g)</th>
</tr>
<tr> <td>Arándano</td> <td>49</td> <td>0.2</td>
<td>0.4</td> <td>12.7</td>
</tr>
<tr> <td>Plátano</td> <td>90</td> <td>0.3</td>
<td>1.0</td> <td>23.5</td>
</tr>
<tr> <td>Cereza</td> <td>46</td> <td>0.4</td>
<td>0.9</td> <td>10.9</td>
</tr>
<tr> <td>Fresa</td> <td>37</td> <td>0.5</td>
<td>0.8</td> <td>8.3</td>
</tr>

</table>
</body>
</html>
```

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {font-family: Arial, Helvetica, sans-serif;}

table {    font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;
           font-size: 12px;     margin: 45px;       width: 480px;
           text-align: left;    border-collapse: collapse; }

th {       font-size: 13px;     font-weight: normal;          padding: 8px;       background: #b9c9fe;
           border-top: 4px solid #aabcfe;          border-bottom: 1px solid #fff;
           color: #039;
}

td {       padding: 8px;        background: #e8edff;          border-bottom: 1px solid #fff;
           color: #669;         border-top: 1px solid transparent;
}

tr:hover td { background: #d0dafd; color: #339; }
```

© aprenderaprogramar.com, 2006-2029

Contenido nutricional por cada 100 g de alimento.

| Alimento | Calorías (kCal) | Grasas (g) | Proteína (g) | Carbohidratos (g) |
|---|---|---|---|---|
| Arándano | 49 | 0.2 | 0.4 | 12.7 |
| Plátano | 90 | 0.3 | 1.0 | 23.5 |
| Cereza | 46 | 0.4 | 0.9 | 10.9 |
| Fresa | 37 | 0.5 | 0.8 | 8.3 |

Fíjate en el código CSS que crea este diseño de tabla. Con los contenidos que hemos estudiado a lo largo del curso ya debemos ser capaces de comprender todas las reglas CSS utilizadas.

Ten en cuenta que nosotros aquí por motivos didácticos estamos aplicando estilos directamente a elementos table, th, td, etc. lo cual supondría que son estilos para todas las tablas. Para definir varios estilos diferentes para tablas bastará definir clases o ids que apliquemos a tablas y luego definir los estilos para esos ids o clases. Por ejemplo #tablaTipo1 td { ...} ó .tablaTipo1 td { ...} serían reglas que nos permitirían aplicar estilos diferentes a tablas concretas a las que hayamos aplicado un id o clase concreta.

Con este ejemplo comprobamos otra cosa: un buen diseño no tiene por qué ser complicado a nivel de código. En la tabla anterior lo podemos comprobar, un buen diseño puede basarse simplemente en una buena combinación de colores de texto, colores de fondo y tipos de letra. El código es bastante sencillo y esto es bueno para acelerar la velocidad de carga de una página web y para evitar problemas debido a que distintos navegadores interpreten el código de forma distinta. Cuando estés trabajando en la creación de páginas web recuerda esta máxima: "lo sencillo puede ser bello y efectivo. Lo complicado tenderá a crearnos problemas".

Lo que hemos indicado aquí aplicado a tablas es válido también para el resto de diseño CSS: composiciones de páginas web, estilos para botones, estilos para formularios, etc. pueden encontrarse en internet y resultarnos útiles para nuestros diseños, y será preferible el código sencillo al complicado.

**EJERCICIO**

Crea el código HTML de una tabla con un título de tabla, cinco columnas y 7 filas (la primera ella, fila de encabezados que no son datos propiamente dichos). Aplícale los siguientes estilos, comprueba la visualización obtenida y responde a las siguientes cuestiones:

```
/* Curso CSS aprenderaprogramar.com */
table {  color: #333; font-family: Helvetica, Arial, sans-serif; width: 640px; border-collapse: collapse;}
td, th { border: 1px solid transparent; height: 30px; }
th { background: #D3D3D3; font-weight: bold; }
td { background: #FAFAFA; text-align: center; }
tr:nth-child(even) td { background: #F1F1F1; }
tr:nth-child(odd) td { background: #FEFEFE; }
tr td:hover { background: #666; color: #FFF; }
```

© aprenderaprogramar.com, 2006-2029

a) Describe cuál es el efecto que genera cada una de las especificaciones que hemos incluido en el código CSS (Ejemplo: color: #333 da lugar a que se muestren todos los textos dentro del elemento table con color #333333, que es un color gris oscuro. Font-family: Helvetica, Arial, sans-serif; da lugar a que ...).

b) Indica qué modificación habría que hacer en el código para que se muestren bordes dobles con grosor 2 píxeles.

c) Indica qué modificación habría que hacer en el código para que se muestren bordes simples de color naranja de grosor 3 píxeles.

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01053D

**Acceso al curso completo** en aprenderaprogramar.com --> Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## DISEÑO LÍQUIDO FRENTE A RESPONSIVE DESIGN

Con el auge del acceso a internet a través de tablets, smartphones, smartTVs, etc. ha surgido el problema de que las páginas web deben mostrarse en pantallas de muy diferente tamaño. Conseguir que la misma página web se vea bien en dispositivos muy distintos es algo que no es sencillo de resolver.

Una primera aproximación o intento de resolver este problema es el denominado diseño líquido o fluido. Se dice que una página web tiene diseño líquido cuando la anchura de sus elementos está definida utilizando porcentajes. Supongamos que tenemos una cabecera cuyo ancho fijamos en el 100 %, una columna lateral cuyo ancho fijamos en el 25 % y una parte principal de contenidos cuyo ancho fijamos en el 75%. El resultado será que al visualizarse en distintos dispositivos la anchura de los elementos será un porcentaje de la anchura disponible. Veamos algunos ejemplos:

| Dispositivo | Dimensiones pantalla en pixeles | Ancho que tomará la columna lateral (25 %) | Ancho que tomará la parte principal (75%) |
|---|---|---|---|
| **Un TV Full HD** | 1.920 x 1080 px | 480 px | 1440 px |
| **Un ordenador portátil** | 1280 x 800 px | 320 px | 960 px |
| **Tablet Samsung Galaxy Tab** | 800 x 1280 px | 200 px | 600 px |
| **iPhone** | 640 x 960 | 160 px | 480 px |

Supongamos que establecemos un ancho “fijo” para nuestra página web de 900 px y que para el contenedor que conforma la columna lateral definimos #columnaLateral {widht: 225px; } y para el contenedor del espacio principal definimos #espacioCentral {width: 675px;}. Con estas reglas decimos que el diseño no es fluido porque no se adaptarán al ancho disponible de pantalla. En la TV Full HD de nuestro ejemplo el ancho disponible es de 1920 px, mientras que el ancho total que ocuparemos con la definición indicada es de 900 px. Esto supone que 1920 – 920 = 1000px de la pantalla del televisor quedará libre, con lo que la sensación será que la página web se ve muy pequeña y se desaprovecha la pantalla.

Por el contrario, si consideramos el iPhone del ejemplo, su ancho disponible es de 640 px mientras que nuestra página web tiene 900 px de ancho. Como la página web no cabe en la pantalla, aparecerá cortada y para poder ver las partes no visibles aparecerán unas barras de scroll. La sensación será que la página web se ve muy grande y cortada.

Pasamos ahora al diseño fluido. Para ello definiríamos las anchuras en porcentajes. Fíjate que siempre hablamos de anchuras porque se entiende que las páginas web en vertical son muy largas y el usuario

© aprenderaprogramar.com, 2006-2029

ya está acostumbrado a tener que subir y bajar por ellas. En cambio, tener que moverse lateralmente es algo que suele considerarse "desagradable". En el diseño fluido definiríamos los anchos de esta manera: para la columna lateral #columnaLateral {widht: 25%; } y para el contenedor del espacio principal #espacioCentral {width: 75%;}.

Parece una solución ideal. Ahora el tamaño de la página web se ajustará a la pantalla y no se desaprovechará ninguna parte de las pantallas grandes ni aparecerán barras de scroll lateral en las pantallas pequeñas. ¿Qué ocurre en la práctica? En la práctica esta solución no es ideal ni mucho menos. Nos encontramos que ocurre lo siguiente: en las pantallas grandes, un párrafo que en una pantalla de un ordenador mediano se ve en 5 líneas se verá quizás en una sola línea ya que el ancho disponible es muy grande. El efecto es una línea muy larga que resulta desagradable de leer. Por el contrario, cuando un párrafo consta de una sola línea corta, en una pantalla muy grande se verá en el lateral izquierdo mientras que a su derecha queda un gran espacio en blanco. En general, el diseño fluido dará lugar a una deformación de contenidos que quedan muy a lo largo o demasiado cortos, y la impresión general es mala.

En las pantallas pequeñas como las de un smartphone, el diseño fluido hará que la página web "se comprima" para ocupar el ancho disponible que es muy poco. Cuando la pantalla es realmente pequeña, el efecto que se genera es de "miniaturización" o que todo se ve "diminuto", tan pequeño que prácticamente no se puede leer. Por ello en la práctica es más habitual establecer estas anchuras principales en pixeles antes que en porcentajes.

La solución a obtener una correcta visualización en páginas web puede pasar por distintas vías:

a) Utilizar lo que se denomina "responsive design" que se podría traducir como "diseño sensible al dispotivo" aunque muchas veces en español se habla simplemente de "diseño responsive". Esta técnica trata de usar un único HTML y CSS pero mediante la detección del dispositivo en que se va a visualizar la página, ofrecer un diseño adecuado para cada tipo de dispositivo (esto obliga a escribir diferentes reglas según el tipo de dispositivo en que se vaya a visualizar la web).

b) Utilizar diferentes dominios para el acceso con dispositivos pequeños o dispositivos grandes. Por ejemplo el dominio http://www.iberia.com/ está destinado a dispositivos de pantallas medianas o grandes mientras que http://iberia.mobi/ está destinado a tablets, smartphones y similares. Los diseños son distintos: más simples para los dispositos pequeños. Puede utilizarse la detección de dispositivo para redirigir del dominio principal al .mobi.

c) Definir ancho en pixeles, que como indicamos al hablar de medidas CSS podría considerarse una medida híbrida entre relativa y absoluta y es la que suele ofrecer un mejor resultado "intermedio".

d) Utilizar otras técnicas.

En general cuando hablamos de las diferentes técnicas que se pueden emplear hay algunas propiedades CSS que suelen resultar útiles y que son: max-widht, min-width, max-height y min-height.

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD MAX-WIDTH**

Esta propiedad permite definir la anchura máxima de un elemento, indicada en una unidad de medida válida con CSS (admitiéndose porcentajes). Su valor por defecto es none, que equivale a que esta propiedad no esté declarada. Se puede aplicar a cualquier elemento excepto elementos inline (aunque sí a elementos inline tipo imagen u objetos incrustados), filas de tablas o grupos de filas de tablas.

Si se usa un porcentaje, se refiere al ancho del elemento contenedor.

En el diseño fluido, nos puede servir para evitar que en las pantallas muy grandes la página web se expanda demasiado lateralmente. De este modo podemos permitir que se vea más grande, pero con un límite máximo. Ejemplo: #columnaLateral {width: 25%; max-widht: 420px; } #espacioCentral {width: 75%; max-widht: 1680px;}.

**PROPIEDAD MIN-WIDTH**

Esta propiedad permite definir la anchura mínima de un elemento, indicada en una unidad de medida válida con CSS (admitiéndose porcentajes). Su valor por defecto es none, que equivale a que esta propiedad no esté declarada. Se puede aplicar a cualquier elemento excepto elementos inline (aunque sí a elementos inline tipo imagen u objetos incrustados), filas de tablas o grupos de filas de tablas.

Si se usa un porcentaje, se refiere al ancho del elemento contenedor.

En el diseño fluido, nos puede servir para evitar que en las pantallas muy pequeñas la página web se miniaturize. De este modo podemos permitir que se vea más pequeña, pero con un límite mínimo. Ejemplo: #columnaLateral {widht: 25%; max-widht: 420px; min-widht: 180px; } #espacioCentral {width: 75%; max-widht: 1680px; min-width: 720px; }.

**PROPIEDAD MAX-HEIGHT**

Esta propiedad es análoga a max-widht, pero menos usada ya que como hemos indicado el sentido vertical suele tomarse menos en consideración. Se puede aplicar a cualquier elemento excepto elementos inline (aunque sí a elementos inline tipo imagen u objetos incrustados), columnas de tablas o grupos de columnas de tablas. En caso de usar porcentajes, se calculan respecto al valor height del contenedor. Si este valor no está establecido, aplicar esta regla puede no tener efecto alguno.

**PROPIEDAD MIN-HEIGHT**

Esta propiedad es análoga a min-widht, pero menos usada ya que como hemos indicado el sentido vertical suele tomarse menos en consideración. Se puede aplicar a cualquier elemento excepto elementos inline (aunque sí a elementos inline tipo imagen u objetos incrustados), columnas de tablas o grupos de columnas de tablas. En caso de usar porcentajes, se calculan respecto al valor height del contenedor. Si este valor no está establecido, aplicar esta regla puede no tener efecto alguno.

**Próxima entrega:** CU01054D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## CURSOR CSS

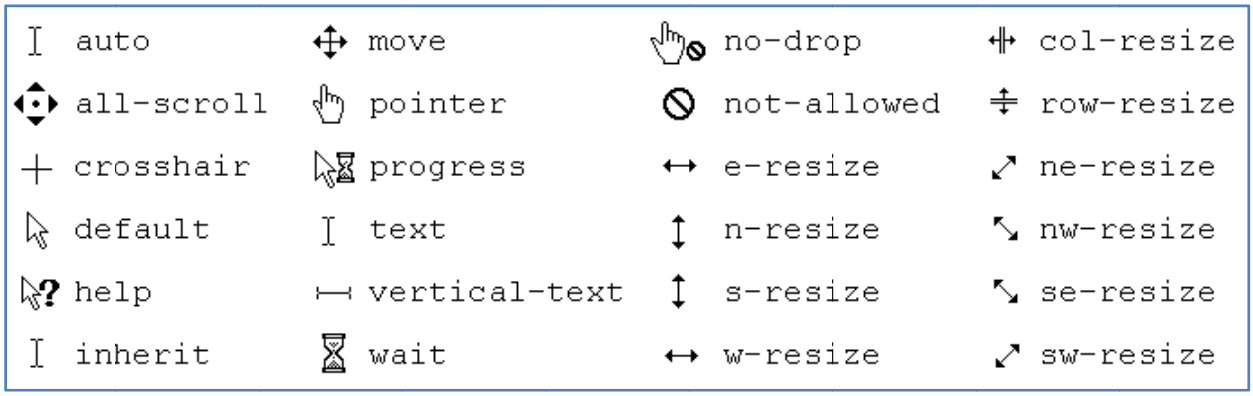
CSS permite modificar el símbolo que se muestra como puntero cuando se sitúa el puntero del ratón sobre una superficie mediante la propiedad cursor. Aunque no siempre se usa, esta propiedad permite generar algunos efectos visuales atractivos y hacer más amigable la navegación.

El motivo por el que resulta de mayor interés esta propiedad es porque combinada con técnicas de programación puede servir para generar efectos dinámicos indicativos de acciones del usuario como "elemento que se puede seleccionar o arrastrar" o "proceso en ejecución" cambiando la forma del puntero del ratón. También se puede usar sin combinar con programación, pero las posibilidades que tendremos para aplicar esta propiedad serán más limitadas.

La propiedad cursor es habitual aplicarla en combinación con hover para cambiar el aspecto del puntero cuando el usuario se posiciona en determinados lugares de la página web, aunque se puede aplicar a un elemento sin necesidad de hacerlo en combinación con hover.

## PROPIEDAD CURSOR

La forma más directa para entender esta propiedad es viendo una imagen con los tipos más habituales de punteros que se pueden mostrar.




| PROPIEDAD CSS cursor | |
|---|---|
| **Función de la propiedad** | Permite definir el aspecto del puntero que se muestra en pantalla. |
| **Valor por defecto** | Auto |
| **Aplicable a** | Todos los elementos |

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS cursor | |
|---|---|
| **Valores posibles para esta propiedad** | auto (el navegador determina el aspecto) |
| | url (rutaDeLaImagenDeseada) [Se pueden especificar varias urls separadas por comas por si no es posible cargar una se intenta cargar la siguiente y un valor predefinido final por si no es posible cargar ninguna url) |
| | Una palabra clave predefinida indicadora de un tipo general: auto, default (puntero básico de defecto, usualmente una flecha), none (no muestra puntero) |
| | Una palabra clave predefinida indicadora de un tipo de status: context-menu, help, pointer, progress, wait |
| | Una palabra clave predefinida indicadora de un tipo de selección: cell, crosshair, text, vertical-text |
| | Una palabra clave predefinida indicadora de un tipo de arrastrar y soltar: alias, copy, move, no-drop, not-allowed |
| | Una palabra clave predefinida indicadora de redimensionamiento o scrolling: all-scroll, col-resize, row-resize, n-resize, e-resize, s-resize, w-resize, ne-resize, nw-resize, ne-resize, nw-resize, se-resize, sw-resize, ew-resize, ns-resize, nesw-resize, nwse-resize |
| | Una palabra clave predefinida indicadora de zoom: zoom-in, zoom-out |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myBox {cursor: url(CU01048D_1.png), wait;}<br>.myBoxT2 { cursor: help;} |

Prueba el resultado de usar la propiedad cursor CSS escribiendo el código que mostramos a continuación y visualizándolo en tu navegador. Ten en cuenta que algunas palabras clave no son reconocidas por algunos navegadores (cuanto más antiguo sea el navegador menos palabras clave reconocerá):

```
<html> <head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{margin:0; padding:0; font-family: sans-serif;}
div{ border-style:solid; border-width:1px;
height: 95px; width: 240px;
margin:10px; background-color:yellow;
font-size: 24px; text-align:center;}
</style>
</head>
<body>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="cursor: url(http://cursors2.totallyfreecursors.com/thumbnails/mickey-mouse2.gif), wait;">Url (wait si falla) </div>
<div style="cursor:none;">none </div>
<div style="cursor:context-menu;">context-menu (falla en varios navegadores) </div>
<div style="cursor:help;">help </div>
<div style="cursor:pointer;">pointer </div>
<div style="cursor:progress;">progress </div>
<div style="cursor:wait;">wait </div>
```

© aprenderaprogramar.com, 2006-2029

```
<div style="cursor:cell;">cell </div>
<div style="cursor:crosshair;">crosshair </div>
<div style="cursor:text;">text </div>
<div style="cursor:row-resize;">row-resize </div>
</div>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="cursor:alias;">alias </div>
<div style="cursor:copy;">copy </div>
<div style="cursor:move;">move </div>
<div style="cursor:no-drop;">no-drop </div>
<div style="cursor:not-allowed;">not-allowed </div>
<div style="cursor:all-scroll;">all-scroll </div>
<div style="cursor:col-resize;">col-resize </div>
<div style="cursor:s-resize;">s-resize </div>
<div style="cursor:nwse-resize;">nwse-resize </div>
<div style="cursor:zoom-in;">zoom-in </div>
<div style="cursor:zoom-out;">zoom-out </div>
</div>
</body>
</html>
```

**EJERCICIO**

Crea un documento HTML y un archivo con la hoja de estilos CSS que cumpla con estos requisitos:

a) Deben exisitir dos contenedores (div1, div2) situados en horizontal, cada uno con margin 25px en todas direcciones, sin relleno, ancho de 200 píxeles, altura de 300 píxeles y borde sólido de 5 píxeles de anchura con color de borde violeta.

b) El div 1 debe contener un texto con varios links (etiqueta <a> de html). Al pasar el cursor sobre cualquiera de los links dentro del div1 el cursor deberá ponerse en modo help (es decir, se verá un pequeño interrogante junto al cursor).

c) El div 2 debe contener una imagen con anchura 300 píxeles y altura igual o superior a 200 píxeles. La imagen debe a su vez ser un link a otro documento HTML al que denominamos documento 2 y que debe abrirse en una nueva pestaña cuando se pulse sobre la imagen. Al pasar el cursor sobre la imagen el cursor deberá ponerse en modo zoom-in, es decir, mostrar una lupa con un pequeño + en su interior. En el documento 2 debemos tener la misma imagen pero con mayor tamaño, por ejemplo 600 píxeles de anchura. Antes de crear el código tendrás que preparar las imágenes: partir de una imagen con un tamaño más amplio, y con la ayuda de cualquier programa de edición de imágenes crear su gemela con menores dimensiones.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01055D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**COMENTARIOS CONDICIONALES CSS**

Con el paso de los años y la evolución de los lenguajes y tecnologías empleados en desarrollos web hay un factor que ha permanecido como un problema: las diferentes formas de interpretar o mostrar el código de los distintos navegadores. Una técnica creada por Microsoft basada en condicionales se ha venido usando para tratar de corregir problemas específicos de ciertos navegadores.

Las personas que trabajan en desarrollos web están acostumbrados a tener que lidiar con este tipo de problemas. No todos los navegadores responden igual. Algunos navegadores (al menos en sus versiones más modernas) cumplen o tratan de cumplir en todo lo posible el estándar definido por la W3C, organización encargada de definir el estándar CSS. Otros navegadores cumplen la mayor parte del estándar pero no responden bien con algunas propiedades o características. Además algunos navegadores incorporan características particulares propias, que no se encuentran dentro del estándar.

Cuando se está realizando un desarrollo web, es frecuente realizar pruebas en distintos navegadores y en distintas versiones. No se pueden probar todos los navegadores y todas las versiones porque sería imposible, pero los desarrolladores web suelen tener un conjunto de navegadores y versiones para test. Uno de los navegadores que históricamente ha venido dando más problemas es el Internet Explorer de Microsoft, aunque hay que decir que en las últimas versiones se han realizado mejoras y cada vez tiende a cumplir mejor los estándares.

Para permitir a los programadores y diseñadores solucionar los problemas o diferencias específicas de Internet Explorer, Microsoft creó un mecanismo de "filtro" que permite definir reglas específicas para navegadores concretos.

En primer lugar veamos en qué puede consistir un problema de distintas interpretación entre navegadores. Para ello hemos escrito este código HTML y el CSS en un archivo con el nombre adecuado y hemos realizado un test para comprobar la visualización en dos navegadores distintos:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="CU01055D_A.css">

</head>
<body>
<div id="txtCabecera"><h1>aprenderaprogramar.com</h1></div>
</body>
</html>
```

© aprenderaprogramar.com, 2006-2029

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}
#txtCabecera {width: 500px; margin: 0 auto; padding: 10px; border: solid 2px; text-align: center;
text-shadow: 2px 2px 12px red, -6px -6px 12px blue ;}
```

Esta prueba en concreto la hemos hecho con el navegador Firefox (versión actual) y con un navegador Internet Explorer de hace unos 12 años (esta es una versión antigua, pero aquí no queremos entrar a discutir sobre las versiones o sobre los navegadores, simplemente queremos poner un ejemplo de cómo con el mismo código se pueden obtener visualizaciones distintas y entender que esto puede ser un problema para los programadores y diseñadores web. Podríamos tener problemas de distintas visualizaciones también entre navegadores modernos). Al comprobar el resultado de visualización nos hemos encontrado con estas diferencias principales:

a) Un navegador muestra el efecto text-shadow mientras que el otro no.
b) Un navegador alinea la caja en el centro de la pantalla mientras que el otro lo alinea a la izquierda de la pantalla.
c) Un navegador muestra un mayor espacio superior de relleno sobre el texto dentro de la caja mientras que el otro muestra menos espacio de relleno.

Para analizar un problema como el de text-shadow un paso lógico es preguntarse: ¿soportan estos navegadores la propiedad text-shadow? Es decir, ¿están preparados para interpretarla correctamente? Da la impresión de que el segundo navegador no. De todas formas, para comprobarlo realizamos la búsqueda "text-shadow mozilla developer" en internet y comprobamos lo siguiente:

**Desktop** Mobile

| Feature | Chrome | Firefox (Gecko) | Internet Explorer | Opera | Safari (WebKit) |
|---|---|---|---|---|---|
| Basic support | 2.0.158.0 | 3.5 (1.9.1) | 10 | 9.5 | 1.1 (100) |

Aquí nos indica para la propiedad text-shadow la compatibilidad de los navegadores y nos dice que esta propiedad es admitida en Chrome desde la versión 2.0.158.0, en Firefox desde la versión 3.5, en

© aprenderaprogramar.com, 2006-2029

Internet Explorer desde la versión 10, en Opera desde la versión 9.5 y en Safari desde la versión 1.1. Esto nos confirma lo que sospechábamos: el navegador que no muestra el efecto text-shadow no lo hace porque no está preparado para interpretar esta propiedad, debido a su antiguedad.

Es conveniente realizar comprobaciones básicas de este tipo antes de dar por hecho que "la culpa es del navegador", ya que a veces la culpa no es del navegador, sino del programador o diseñador que ha creado un código inadecuado.

Una vez conocemos el problema podemos elegir entre dos opciones: corregirlo para que la visualización sea correcta en el navegador donde hemos detectado el problema, o ignorarlo. Si lo ignoramos la página web no se verá correctamente en ese navegador, pero como hemos comentado no se pueden comprobar ni corregir los problemas de visualización en todos los navegadores. Hay navegadores muy antiguos en los que no se suelen chequear ni corregir problemas de visualización porque se consideran obsoletos y que hay muy poca gente que pueda estar usándolos todavía. No tiene sentido aspirar a que nuestros desarrollos web se vean correctamente en todos los navegadores porque esto nos llevaría semanas o meses de trabajo y tendría poco interés práctico: lo normal es centrarse en que la página se vea bien en los principales navegadores en versiones relativamente modernas (hasta qué versión comprobar es algo que cada desarrollador o equipo de desarrollo tiene que decidir).

**VARIAS FORMAS DE SOLUCIONAR UN PROBLEMA**

Con frecuencia existen varias formas de solucionar un problema. Vamos a referirnos, a modo de ejemplo, al problema de que la caja no se muestre centrada en uno de los navegadores que hemos probado. A modo de comprobación, realizamos la búsqueda "margin css mozilla developer" en internet y nos vamos a la página de documentación de Mozilla correspondiente. En la parte denominada "Browser compatibility" aparece que para Internet Explorer el Basic support fue introducido en la versión 3.0 mientras que para auto value indica que la versión en que se introdujo fue 6.0 (strict mode). Esto nos da una pista sobre una forma de solucionar el problema: la indicación de strict mode quiere decir que Internet Explorer no realiza el centrado con margin: 0 auto; excepto si en la cabecera del documento HTML se hace una declaración de que el documento está definido en HTML strict. Para corregir el problema escribiríamos esta línea como primera línea del documento HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Una vez hecho esto comprobaríamos que la caja ya aparece centrada. Quizás esta solución pueda resultarnos adecuada. O quizás no, porque no queramos realizar este tipo de declaración.

También es posible que no nos demos cuenta de que el problema se podría resolver de esta manera.

En caso de que no nos resulte una solución adecuada o que se nos pase por alto la posibilidad de usar esta solución, normalmente dispondremos de varias alternativas para lograr un mismo objetivo. Existen en internet muchas webs donde se habla de técnicas CSS para resolver problemas o conseguir un objetivo de distintas maneras. No tiene sentido aspirar a conocerlo todo ni a saber todos los "trucos" o alternativas posibles. Sí será conveniente que nos vayamos construyendo nuestro pequeño repertorio de trucos o formas de solución y que nos acostumbremos a buscar en internet formas de solucionar problemas.

© aprenderaprogramar.com, 2006-2029

**FILTROS CSS**

Se habla de "filtro CSS" en alusión a alguna instrucción o mecanismo que permita conocer qué navegador, versión o tipo de dispositivo (tablet, smartphone, etc.) es en el que se va a visualizar la página web y aplicar unas reglas específicas para ese caso.

Lo ideal sería no tener que usar filtros, sino que unas solas reglas nos resolvieran todos los problemas, pero en la práctica a veces no queda más remedio que recurrir a diversas estrategias para lograr mejorar la visualización en navegadores o dispositivos que presentan problemas de visualización y no queremos dejar atrás. No obstante, ten presente una cosa: cuantos menos filtros, trucos o especificidades utilices mejor, el código será más sencillo y por tanto más fácil de mantener y de corregir.

Existen diversos tipos de filtros y diversas técnicas. Vamos a comentar ahora una de ellas, que se ha venido usando para resolver problemas de visualización específicos del navegador Internet Explorer de Microsoft, denominada técnica de "comentarios condicionales para Internet Explorer".

La forma de aplicar estos filtros se basa en indicar con un condicional unas reglas específicas a cargar y aplicar exclusivamente para ese navegador (o conjunto de navegadores). Las reglas se escriben dentro de las etiquetas <head> ...</head> del documento HTML y estos ejemplos facilitan su comprensión:

| Ejemplo | Significado |
|---|---|
| <!--[if IE 7]><br><link rel="stylesheet" type="text/css" href="ie7.css"><br><![endif]--> | Si el navegador es Internet Explorer 7, concretamente para este navegador aplicar las reglas definidas en el archivo indicado. Se puede especificar IE 7, IE 8, IE 9, IE 10, IE 12, IE 15, IE 20, IE 30... es decir, cualquier versión que queramos. |
| <!--[if IE]><br><link rel="stylesheet" type="text/css" href="ie.css"><br><![endif]--> | Si el navegador es Internet Explorer en cualquiera de sus versiones, concretamente para este navegador aplicar las reglas definidas en el archivo indicado. |
| <!--[if gte IE 15]><br><link rel="stylesheet" type="text/css" href="ie15h.css"><br><![endif]--> | Si el navegador es Internet Explorer 15 o superior, concretamente para esta versión o sus superiores aplicar las reglas definidas en el archivo indicado. gte = greater than or equal to |
| <!--[if gt IE 15]><br><link rel="stylesheet" type="text/css" href="ie15u.css"><br><![endif]--> | Si el navegador es Internet Explorer versión superior a la 15, concretamente para estas versiones o sus superiores aplicar las reglas definidas en el archivo indicado. gt = greater than |
| <!--[if lte IE 15]><br><!--[if lt IE 15]> | Mismo razonamiento pero para versiones inferiores o iguales, o inferiores. lt = lower than, inferior. lte = lower than or equal to, inferior o igual a. |
| <!-- [if (gt IE 6)&(lt IE 10)]><br><link rel="stylesheet" type="text/css" href="ieR.css"><br><![endif]--> | Operador lógico "and", si el navegador cumple el primer requisito y el segundo al mismo tiempo, concretamente para esas versiones aplicar las reglas definidas en el archivo indicado. |
| <!-- [if (IE 19)\|(IE 20)]><br><link rel="stylesheet" type="text/css" href="ieY.css"><br><![endif]--> | Operador lógico "or", si el navegador cumple el primer requisito, o el segundo, o ambos, concretamente para esas versiones aplicar las reglas definidas en el archivo indicado. |

© aprenderaprogramar.com, 2006-2029

Nota: no deben existir espacios sobrantes dentro de los tags porque en ese caso puede no interpretarse el código.

Ahora veamos cómo sería la aplicación concreta para resolver el problema que hemos descrito anteriormente.En el código HTML añadiríamos el fragmento condicional de la siguiente manera:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="CU01055D_A.css">
<!--[if(IE 7)]>
<link rel="stylesheet" type="text/css" href="ie7.css">
<![endif]-->
</head>
<body>
<div id="txtCabecera"><h1>aprenderaprogramar.com</h1></div>
</body>
</html>
```

Fíjate que aquí no hemos incluido la declaración de DOCTYPE que indicamos antes. Vamos a suponer que no queremos incluir esta declaración (o que no nos hemos dado cuenta de que podemos usar esto para resolver el problema de centrado de la caja).

Ahora en un archivo css independiente del principal, al que vamos a denominar ie7.css, incluiremos estas reglas:

```
/* Curso CSS estilos aprenderaprogramar.com*/
body {text-align:center;}
#txtCabecera { background-color: #FA8258; }
h1 { padding: 15px 5px 5px 5px;}
```

Ahora tenemos 3 archivos. El archivo html del documento html, el archivo CSS principal (en nuestro caso lo hemos llamado CU01055D_A.css) y el archivo CSS específico para los navegadores que cumplen el condicional indicado (en nuestro caso el condicional es que el navegador sea Internet Explorer7 y el archivo lo hemos denominado ie7.css). Las tres reglas definidas en el archivo ie7.css se aplicarán sólo cuando el navegador donde se vaya a mostrar la página web sea el que hemos indicado en el condicional. Las reglas que aplicamos son:

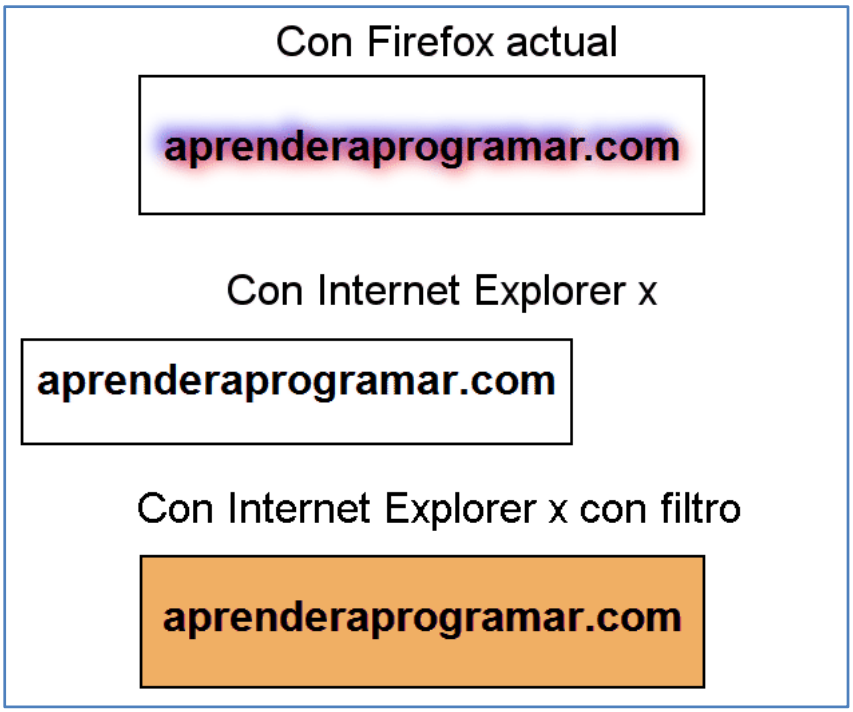
a) body {text-align:center;} , alineamos al centro los elementos dentro de body. Esta regla posiblemente no la usaríamos en un desarrollo web real porque en general no nos va a interesar alinear todos los elementos dentro de body al centro (si lo hacemos, tendríamos que establecer alineaciones específicas para el resto de elementos cuando no quisiéramos que estén centrados). Pero para este ejemplo concreto nos sirve.
b) #txtCabecera {background-color: #FA8258;} , a falta del efecto text-shadow aplicamos como sustituto un color de fondo. No logramos el mismo efecto que con text-shadow, pero supondremos que es la alternativa que elegimos entre las varias posibles. Otra alternativa sencilla sería usar una imagen de fondo. Podríamos asemejar más el aspecto al que genera text-shadow, pero no vamos a entretenernos en ello.

© aprenderaprogramar.com, 2006-2029

c) h1 { padding: 15px 5px 5px 5px;}, ampliamos el relleno para conseguir una visualización lo más parecida a lo que nosotros deseamos.

Ten en cuenta que hablamos de "visualización parecida", similar, etc. ya que no podemos o no queremos detenernos a buscar "la precisión absoluta" porque esto no tiene sentido.

El resultado ahora obtenido lo comprobamos a continuación. Gracias al filtro hemos centrado la caja, adecuado su relleno y aunque no hemos igualado el efecto de sobra al menos hemos puesto un color de fondo:




**MÁS FILTROS CSS**

Hemos comentado una técnica de filtrado concreta, pero hay otras técnicas. Algunas de esas otras técnicas las veremos más adelante dentro del curso. Ten en cuenta que las técnicas de filtrado van evolucionando con el tiempo: algunas técnicas dejan de ser útiles o válidas mientras que surgen otras nuevas. En este curso no nos interesa conocer las diferentes técnicas de filtrado para los distintos navegadores, sino comprender la problemática existente en torno a las diferencias entre navegadores y el concepto de filtrado como técnica que puede ayudar a solucionar problemas específicos.

**EJERCICIO**

Busca en alguna página web o blog de internet una técnica de filtrado, aplícala y explica su utilidad. Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01056D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## PREFIJOS CSS DE NAVEGADORES

Se llaman prefijos de navegador o prefijos comerciales (vendor prefixes) a un prefijo que se antepone a una regla CSS destinado a que dicha regla sea leída y aplicada exclusivamente por un navegador concreto (por ejemplo Chrome) pero no por el resto de navegadores. El uso de prefijos suele aplicarse a propiedades que se encuentran en fase experimental o que aún no se han convertido en un estándar.

Al igual que los comentarios condicionales que se han venido usando específicamente para Microsoft Internet Explorer, los prefijos son un tipo de filtro que permite que una instrucción CSS se aplique específicamente a un navegador o familia de navegadores pero no a los demás. Sin embargo, a diferencia de los comentarios condicionales, existen prefijos específicos para todos los tipos de navegador. A continuación se indican los prefijos más habituales y seguidamente pasaremos a ver un ejemplo.

| Prefijo | Familia de navegadores a los que aplica |
|---|---|
| **-webkit-** | Chrome, Safari, Android, iOs |
| **-moz-** | Firefox |
| **-o-** | Opera |
| **-ms-** | Microsoft Internet Explorer |

Veamos un ejemplo de uso que es quizás lo que mejor nos haga comprender para qué se usan los prefijos. Escribe este código HTML y CSS (estableciendo el nombre de archivo adecuado) que procedemos a explicar a continuación:

© aprenderaprogramar.com, 2006-2029

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="CU01056D_A.css">
</head>
<body>
<div class="fondoGradient"><h1>aprenderaprogramar.com</h1></div>
</body>
</html>
```

```
/* Curso CSS estilos aprenderaprogramar.com*/
*{font-family: arial;}

.fondoGradient {
 text-align:center; width: 500px; height: 9em;
 line-height: 4.5em; margin: 0 auto;
 border: solid 3px;
 background-color: #F4FA58; /*Navegadores que no aceptan el gradiente de fondo*/
 background-image: -webkit-gradient(linear, left top, left bottom, from(#F4FA58), to(#FF0000)); /* Chrome, Safari versiones
más antiguas */
 background-image: -webkit-linear-gradient(top, #F4FA58, #FF0000); /* Chrome, Safari versiones relativamente modernas */
 background-image:   -moz-linear-gradient(top, #F4FA58, #FF0000); /* Firefox versiones relativamente modernas*/
 background-image:    -o-linear-gradient(top, #F4FA58, #FF0000); /* Opera versiones relativamente modernas*/
 background-image:       linear-gradient(to bottom, #F4FA58, #FF0000); /* Chrome, Firefox, IE, Opera versiones actuales */
}
```

Vamos a comentar el código CSS que hemos utilizado y que a primera vista podremos calificar como "poco elegante", ya que como vemos hay una aparente repetición de instrucciones con distintos prefijos o variantes de sintaxis. La repetición de código nunca es deseable porque dificulta la comprensión y mantenimiento de nuestros desarrollos web. No obstante, tenemos que estar preparados para interpretar este tipo de situaciones cuando estemos analizando una página web. También podemos usar código de este tipo cuando consideremos que es razonadamente necesario, pero debemos evitar su uso continuado y sin reflexión.

El código CSS define una clase donde el texto del elemento se alinea al centro mediante la propiedad text-align, la anchura del elemento se establece en 500 pixeles, y su anchura en 9 veces la altura de letra de aplicación. Para centrar verticalmente el texto se ha usado line-height. Se ha establecido un borde de 3 pixeles y un color de fondo con código #F4FA58 que es un tono de amarillo mediante la propiedad background-color.

A continuación se escribe una propiedad background-image en distintas ocasiones. Recordar que la propiedad background-image coloca una imagen encima del background-color. Por tanto si la imagen ocupa todo el contenedor, el color de fondo quedará oculto. La primera línea que encontramos es:

background-image: -webkit-gradient(linear, left top, left bottom, from(#F4FA58), to(#FF0000)); Esta línea aplica el prefijo -webkit- a una propiedad denominada gradient que crea una imagen de fondo tipo "degradado de colores" a partir de la definición de los colores que deben intervenir y de otros parámetros. El prefijo webkit significa que esta línea está dirigida a navegadores Chrome, Safari,

© aprenderaprogramar.com, 2006-2029

Android, iOs y que el resto de navegadores ignorarán lo que aquí se indique. Sin embargo en la siguiente línea volvemos a encontrar el prefijo webkit de esta manera:

background-image: -webkit-linear-gradient(top, #F4FA58, #FF0000); ¿Por qué se repite dos veces el prefijo webkit?

Como hemos dicho, los prefijos se usan normalmente para propiedades experimentales, que están en fase de diseño o prueba. La sintaxis para la propiedad con el prefijo -webkit- la definen los desarrolladores de estos navegadores y a lo largo del tiempo es posible que cambien la forma en que debe escribirse una regla experimental. Ello da lugar a situaciones como esta: existe una sintaxis que se venía usando en algunas versiones de estos navegadores (que es la definida en la primera línea). Luego la sintaxis fue cambiada y se empezó a usar otra diferente (que es la definida en la segunda línea). Para que la página se visualice como esperamos tanto en navegadores Chrome más antiguos como en navegadores Chrome más modernos, se incluyen ambas formas de la sintaxis. El navegador ignorará aquella que no es adecuada y utilizará la que puede interpretar correctamente.

A continuación nos encontramos con líneas que son exclusivamente de aplicación a navegadores Firefox (prefijo -moz-) y Opera (prefijo -o-).

Finalmente nos encontramos con esta línea:

background-image: linear-gradient(to bottom, #F4FA58, #FF0000); Aquí podemos comprobar que no aparece ningún prefijo. ¿Por qué? Porque a partir de una determinada versión esta propiedad ha dejado de considerarse experimental y se ha convertido en una propiedad "estándar" (al menos para algunos navegadores). Ahora para que los usuarios que usen navegadores en los que esta propiedad se ha normalizado aplicamos esta línea.

Con estas repeticiones de la misma regla con diferentes prefijos o sintaxis buscamos lograr que la visualización sea correcta en:

- Las versiones de navegadores que introdujeron esta propiedad como experimental por primera vez.
- Las versiones de navegadores que mantuvieron la propiedad como experimental pero fueron introduciendo cambios en la sintaxis.
- Las versiones de navegadores que dejaron de considerar a esta propiedad como experimental y la empezaron a emplear como una propiedad normalizada.

De este modo tratamos de cubrir el mayor rango de navegadores posibles para que el mayor número de usuarios posible vean la página web como nosotros esperamos. Ahora bien, ¿qué ocurre con usuarios que empleen versiones de navegadores en los que esta propiedad no existía ni siquiera como experimental? Estos navegadores ignorarán la sintaxis incluso aunque exista un prefijo dirigido a ellos porque no está preparados para reconocer esas instrucciones. Al ignorar background-image aplicarán la propiedad background-color, que sí es reconocida por todos los navegadores. El efecto no será el mismo, pero es la alternativa que se ha dado aquí.

En los navegadores que sí reconozcan el gradiente el resultado será similar a este:

© aprenderaprogramar.com, 2006-2029

**BUEN USO Y MAL USO DE PREFIJOS**

A la hora de usar propiedades experimentales y aplicar prefijos conviene plantearse cuáles son las ventajas y los inconvenientes existentes para cada caso particular. Supongamos que el ejemplo anterior simplemente trataba de aplicar una imagen con gradiente de color al fondo de una caja en la página web. ¿Qué ventajas e inconvenientes presentaría el uso de la propiedad linear-gradient?

Ventajas:

- Posiblemente el tiempo de carga de la página sea más rápido usando esta propiedad que usando una imagen. Con esta propiedad el navegador simplemente tiene que renderizar (dibujar) a partir de una instrucción, mientras que con una imagen es necesario descargar el archivo y cada descarga de archivo implica un pequeño consumo de tiempo.

Inconvenientes:

- Al ser una propiedad con poca trayectoria histórica las diferentes versiones de navegadores requieren de distintas sintaxis, lo que obliga a la repetición de varias líneas de código con el mismo fin.
- Podemos tener dudas de que la visualización vaya a ser buena en la mayor parte de dispositivos y navegadores, ya que algunos de ellos (en especial los más antiguos) no reconocerán esta propiedad.

¿Qué será mejor entonces, aplicar o no aplicar este tipo de propiedades? En principio te recomendaríamos que siguieras estos criterios:

- Si dispones de una alternativa altamente estandarizada, simple, que permite alcanzar el objetivo planteado y que evita la repetición de código, úsala y prescinde de propiedades experimentales.
- Si consideras que te aporta más ventajas que inconvenientes y te decides a usarla, trata de cubrir el mayor rango posible de navegadores y versiones y establece una regla "de salvaguarda", es decir, una regla que permita que la página web se vea razonablemente bien si la propiedad que estás intentando aplicar no es admitida por un navegador concreto. En nuestro ejemplo anterior la regla de salvaguarda era background-color, alternativa en caso de que fallara la aplicación de background-image.

Respecto al uso de prefijos, úsalos sólo cuando sean realmente necesarios. Aquí hemos visto cómo se aplica el prefijo a una propiedad, pero también se aplican en ocasiones a selectores o valores. Antes de aplicar prefijos infórmate en páginas que den información de calidad como Mozilla Developer Network.

Otra cuestión a valorar es si merece escribir las distintas sintaxis para dar soporte a versiones anteriores de navegadores. Cuando la propiedad se ha convertido en un estándar, algunos programadores y diseñadores optan por escribir simplemente la sintaxis estándar junto a la salvaguarda y se olvidan de la repetición de distintas sintaxis para versiones anteriores de navegadores. Es decir, en el ejemplo anterior en lugar de las diversas líneas escribiríamos simplemente:

© aprenderaprogramar.com, 2006-2029

```
background-color: #F4FA58;
background-image: linear-gradient(to bottom, #F4FA58, #FF0000);
```

Es decir, se sintetiza en dos línea: la salvaguarda y la sintaxis estandarizada. En algunos casos ni siquiera será necesario la salvaguarda. Por ejemplo imagínate que pretendes aplicar un borde redondeado y usas una propiedad que ha sido experimental hasta hace poco sin aplicar salvaguarda. Los navegadores que reconozcan la regla mostrarán el borde redondeado, mientras que los que no la reconozcan lo mostrarán rectangular. Si este no es un aspecto clave y no te preocupa, puedes utilizar una sola línea y dejar que se muestre de una forma u otra según el navegador que esté empleando el usuario.

Como ves existen distintas alternativas y recursos para resolver problemas o conseguir objetivos, no hay un criterio único. Algunos equipos de trabajo siguen prefiriendo usar imágenes antes que propiedades avanzadas porque piensan que así se garantizan la misma representación en todos los navegadores, mientras que otros equipos optan por el uso de nuevas propiedades considerando que aportan más ventajas que inconvenientes. Con el tiempo irás formándote tus propios criterios y formas de actuar y razonar en lo relativo al uso de CSS. No pretendas hacer las cosas perfectas, simplemente ocúpate de hacer que las cosas funcionen bien aunque la codificación no sea la mejor posible.

**MÁS FILTROS CSS**

Hemos comentado ya dos técnicas de filtrado, una basada en comentarios condicionales y otra en prefijos, pero hay otras técnicas. Algunas de esas otras técnicas las veremos más adelante dentro del curso.

**EJERCICIO**

Busca en alguna página web o blog de internet una técnica de filtrado basada en prefijos para el navegador que estés utilizando. Aplícala y explica su utilidad. Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01057D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## COLUMNAS EN CSS

Aunque con las herramientas que ya conocemos somos capaces de crear columnas, CSS ha introducido una serie de propiedades que pretenden facilitar el diseño de elementos web en columnas. Vamos a estudiar las propiedades column-count, column-width, columns, colum-gap y column-rule.

Ten en cuenta que estas propiedades no son soportadas por muchas de las versiones de navegadores que no son recientes, e incluso en algunos de los recientes todavía se consideran experimentales, lo que obliga al uso de prefijos específicos de navegador, aunque algunos navegadores ya las han introducido como estándar y no requieren de prefijo.

## PROPIEDAD COLUMN-COUNT

| PROPIEDAD CSS colum-count | |
|---|---|
| **Función de la propiedad** | Permite definir el número de columnas con que se debe mostrar el contenido dentro de un elemento. |
| **Valor por defecto** | auto |
| **Aplicable a** | Contenedores como elementos block, table, table-cell, etc. |
| **Valores posibles para esta propiedad** | Un número entero igual o superior a 1 |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer {column-count: 3;}<br>.myContainerSP { column-count: 5;} |

## PROPIEDAD COLUMNS

Esta propiedad es un shortand que permite especificar el número de columnas bien en forma de número de columnas (lo que sería equivalente a usar column-count), bien indicando una unidad de medida (lo que sería equivalente a usar column-width).

Ejemplo: myBox {columns: 3; }

© aprenderaprogramar.com, 2006-2029

**PROPIEDAD COLUMN-WIDTH**

| PROPIEDAD CSS colum-width | |
|---|---|
| **Función de la propiedad** | Permite sugerir un ancho de columna deseado, aunque su aplicación no será estricta si existe column-count, que induce un ancho basado en el número de columnas especificado. |
| **Valor por defecto** | auto |
| **Aplicable a** | Contenedores como elementos block, table, table-cell, etc. |
| **Valores posibles para esta propiedad** | auto (el número de columnas derivará del establecido con column-count) |
| | Una unidad de medida válida en CSS |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer {column-width: 150px;}<br>.myContainerSP { column-width: 5em;} |

En presencia de column-count, puede omitirse esta propiedad ya que column-count induce un ancho de columna basado en el número de columnas indicado. Puede entrar en conflicto con column-count si los valores indicados son incompatibles entre sí. En ausencia de column-count, puede inducir un número de columnas en base al ancho especificado. Si nos fijamos, column-count y column-width vienen siendo dos formas de expresar lo mismo.

**PROPIEDAD COLUMN-GAP**

Esta propiedad sirve para definir un espacio de separación entre columnas. Ejemplo: column-gap: 20px;

En algunos navegadores es necesario el uso de prefijo específico de navegador.

**PROPIEDAD COLUMN-RULE-WIDTH, COLUM-RULE-STYLE, COLUM-RULE-COLOR, COLUMN-GAP**

Estas propiedades tienen por finalidad establecer el ancho, style y color de la línea de separación entre columnas.

column-rule-width funciona de forma análoga a border-width (valor por defecto medium, y valore posibles thin, medium, thick ó una unidad de medida válida CSS).

column-rule-style funciona de forma análoga a border-style.

column-rule-color funciona de forma análoga a border-color.

**PROPIEDAD SHORTAND COLUMN-RULE**

Esta propiedad permite establecer los valores de column-rule width, column-rule-style y column-rule-color en una sola línea. Ejemplo: colum-rule: 3px solid blue;

© aprenderaprogramar.com, 2006-2029

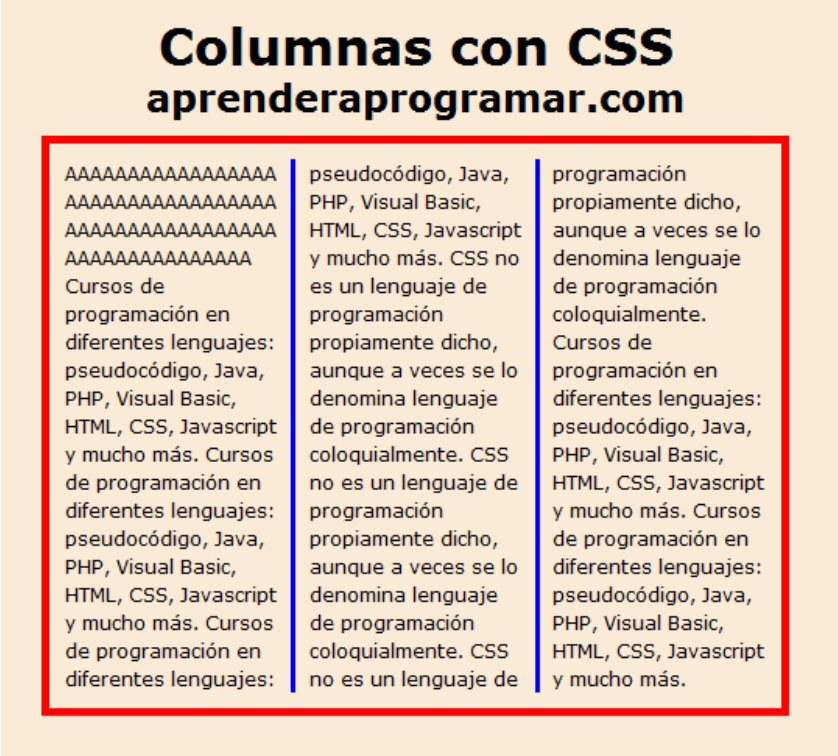
**EJEMPLO DE DISEÑO CON COLUMNAS**

Escribe este código y visualiza sus resultados. Con los contenidos que hemos explicado a lo largo del curso debes ser capaz de interpretar todo el código que hemos incluido, por ejemplo por qué resulta de interés aplicar la propiedad word-wrap, por qué usamos prefijos y por qué hemos dejado comentada la línea correspondiente a column-width. También debes ser capaz de valorar las ventajas y desventajas que puede tener usar este tipo de propiedades.

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">

<style type="text/css">
* {font-family: verdana, sans-serif; margin:0; background-color:#FAEBD7;}
h1, h2 {text-align:center;}
.container2{
        word-wrap: break-word;      font-size: 12px;
        width: 450px;       border: solid 5px red;
        margin: 10px auto 10px auto; padding: 10px; line-height: 1.5em;
         -moz-column-count: 3; -webkit-column-count: 3;          column-count: 3;
        /*-moz-column-width: 130px; -webkit-column-width: 130px; colum-width: 130px;*/
        -moz-column-gap: 20px; -webkit-column-gap: 20px; colum-gap: 20px;
        -moz-column-rule: 3px solid blue; -webkit-column-rule: 3px solid blue;
        colum-rule: 3px solid blue;}
</style> </head>

<body>
<h1>Columnas con CSS</h1>
<h2>aprenderaprogramar.com</h2>

  <div class="container2">
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Cursos de programación en diferentes lenguajes: pseudocódigo, Java, PHP, Visual Basic, HTML, CSS, Javascript y mucho más. Cursos de programación en diferentes lenguajes: pseudocódigo, Java, PHP, Visual Basic, HTML, CSS, Javascript y mucho más. Cursos de programación en diferentes lenguajes: pseudocódigo, Java, PHP, Visual Basic, HTML, CSS, Javascript y mucho más.

CSS no es un lenguaje de programación propiamente dicho, aunque a veces se lo denomina lenguaje de programación coloquialmente. CSS no es un lenguaje de programación propiamente dicho, aunque a veces se lo denomina lenguaje de programación coloquialmente. CSS no es un lenguaje de programación propiamente dicho, aunque a veces se lo denomina lenguaje de programación coloquialmente.

Cursos de programación en diferentes lenguajes: pseudocódigo, Java, PHP, Visual Basic, HTML, CSS, Javascript y mucho más. Cursos de programación en diferentes lenguajes: pseudocódigo, Java, PHP, Visual Basic, HTML, CSS, Javascript y mucho más.
  </div>

</body>
</html>
```

El resultado que se obtiene en un navegador que acepte estas propiedades será similar a este:

© aprenderaprogramar.com, 2006-2029




**EJERCICIO**

Analiza el código que mostramos a continuación, aplícalo a un documento HTML y visualiza los resultados, y responde a las preguntas.

```
.cols3 {
 -webkit-column-count: 3;  -webkit-column-gap: 20px;   -webkit-column-rule: 1px solid #000;
 -moz-column-count: 3;   -moz-column-gap: 20px;  -moz-column-rule: 1px solid #000;
 column-count: 3;  column-gap: 20px;  column-rule: 1px solid #000;
}

.cols3 h1 {   -webkit-column-span:all;   -moz-column-span:all;   column-span:all;
}
```

Explica paso a paso a qué da lugar cada instrucción o fragmento de código (ejemplo: .cols3 indica que se aplicarán los estilos definidos a todos los elementos html cuyo atributo class sea igual a cols3, -webkit-column-count: 3; se escribe para lograr que ...). ¿Qué utilidad tiene la propiedad column-span?

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01058D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## ESQUINAS REDONDEADAS EN CSS

Las cajas CSS son rectangulares y cuando aplicamos propiedades como un color o imagen de fondo, se aplican sobre la caja rectangular. Sin embargo, para hacer los diseños web más atractivos los diseñadores usan esquinas redondeadas. Este efecto antiguamente no era fácil de conseguir y había que recurrir a utilizar imágenes de fondo con transparencia y redondeadas u otras técnicas.

Con la propiedad border-radius se ha hecho posible redondear las esquinas de las cajas con facilidad. Ten en cuenta que esta propiedad no es soportada por muchas de las versiones de navegadores que no son recientes, e incluso en algunos de los recientes todavía es posible que se consideren experimentales, lo que obliga al uso de prefijos específicos de navegador, aunque la mayor parte de los navegadores actuales ya las han introducido como estándar y no requieren de prefijo.

## FORMAS DE ESPECIFICAR EL REDONDEO

El redondeo de una esquina se puede especificar con un parámetro "R" que indica el radio del círculo que se va a emplear para generar el redondeo (cuanto más grande mayor efecto de redondeo conseguiremos).

Aunque el uso de un fragmento de círculo es quizás lo más habitual, también podemos generar esquinas a partir de un fragmento de elipse como vemos en la siguiente imagen.

Cuando usamos un fragmento de elipse hemos de especificar dos parámetros, un valor horizontal R1 y un valor vertical R2. Si R1 y R2 son iguales equivale a usar un solo valor. Si R1 y R2 son distintos generamos distintos efectos de "achatamiento".

## PROPIEDAD BORDER-TOP-LEFT-RADIUS

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS border-top-left-radius | |
|---|---|
| **Función de la propiedad** | Permite redondear la esquina superior izquierda de una caja CSS y definir la forma en que se hace indicando parámetros que hacen la esquina más o menos redondeada. |
| **Valor por defecto** | 0 |
| **Aplicable a** | Todos los elementos. |
| **Valores posibles para esta propiedad** | Una unidad de medida válida CSS (indica radio para el círculo que genera el redondeo de la esquina superior izquierda, excepto si se usa un porcentaje) |
| | Dos unidades de medida válidas CSS (la primera medida indica radio horizontal de la elipse a emplear y la segunda medida radio vertical) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer {border-top-left-radius: 40px;}<br>.myContainerSP { border-top-left-radius: 40px 20px;} |

Si se especifican valores en forma de porcentajes, se calculan para el radio horizontal en función del valor width de la caja, y para el radio vertical en función del valor height de la caja. Si se especifica un solo porcentaje se calcularán los dos radios: el horizontal en función del ancho y el vertical en función del alto. Por tanto el uso de porcentaje implica que se aplica el redondeo por elipse.

### PROPIEDADES BORDER-TOP-RIGHT-RADIUS, BORDER-BOTTOM-LEFT-RADIUS Y BORDER-BOTTOM-RIGHT-RADIUS

Estas propiedades funcionan de la misma forma que hemos visto para border-left-top-radius. Nos permiten especificar el redondeo individual de cada una de las esquinas superior derecha, inferior izquierda o inferior derecha.

### PROPIEDAD BORDER-RADIUS (SHORTAND)

| PROPIEDAD CSS border-radius | |
|---|---|
| **Función de la propiedad** | Permite especificar el redondeo de todas las esquinas de una caja CSS y definir la forma en que se debe hacer. |
| **Valor por defecto** | 0 |
| **Aplicable a** | Todos los elementos. |
| **Valores posibles para** | Una unidad de medida válida CSS (indica radio para el círculo que genera el redondeo de todas las esquinas) |

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS border-radius | |
|---|---|
| **esta propiedad** | Dos unidades de medida válidas CSS (la primera medida indica radio superior izquierdo y radio inferior derecho; la segunda medida indica radio superior derecho e inferior izquierdo) |
| | Tres unidades de medida válidas CSS (la primera medida indica radio superior izquierdo, la segunda radio superior derecho e inferior izquierdo y la tercera radio inferior derecho) |
| | Cuatro unidades de medida válidas CSS (la primera medida indica radio superior izquierdo, la segunda radio superior derecho, la tercera radio inferior izquierdo y la cuarta radio inferior derecho) |
| | Una a cuatro unidades de medida / Una a cuatro unidades de medida (las unidades a la izquierda de la barra son radio horizontal para el formato de elipse y las unidades a la derecha de la barra son radio vertical) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer {border-radius: 40px;}<br>.myContainerSP { border-radius: 40px / 20px;} |

Si se especifican valores en forma de porcentajes, se calculan para el radio horizontal en función del valor width de la caja, y para el radio vertical en función del valor height de la caja. Si se especifica un solo porcentaje se calcularán los dos radios: el horizontal en función del ancho y el vertical en función del alto. Por tanto el uso de porcentaje implica que se aplica el redondeo por elipse.

### EJEMPLOS DE USO DE BORDER-RADIUS

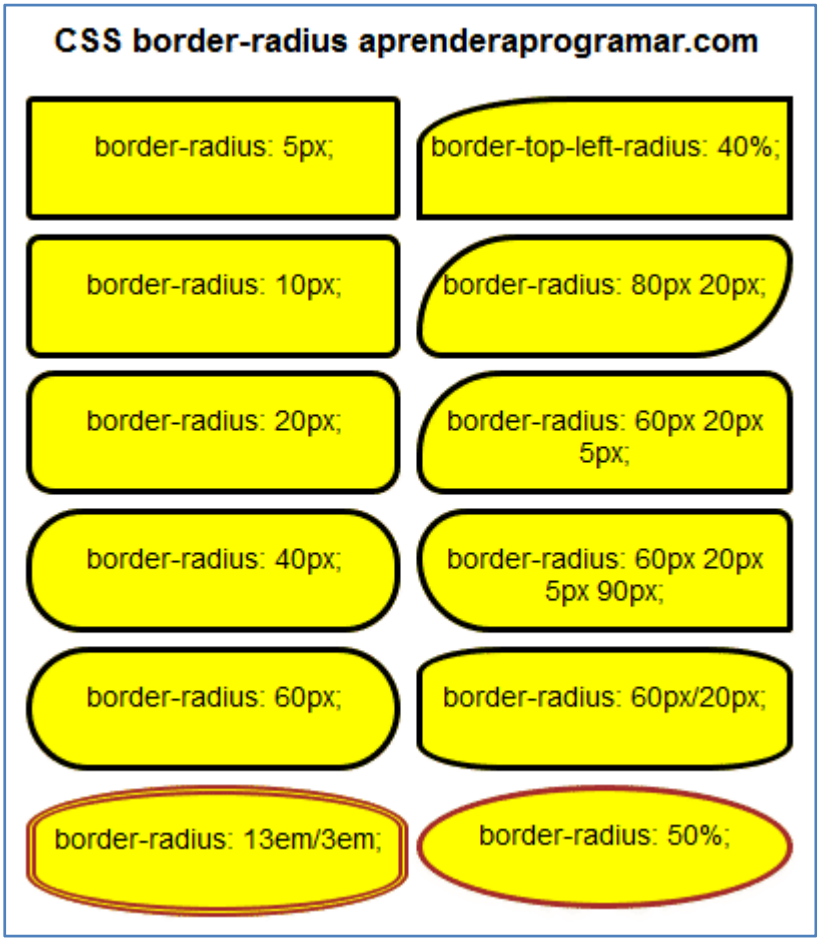
Escribe este código y visualiza sus resultados. Con los contenidos que hemos explicado a lo largo del curso debes ser capaz de interpretar todo el código que hemos incluido. También debes ser capaz de valorar las ventajas y desventajas que puede tener usar este tipo de propiedades.

```
<head> <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{margin:0; padding:0; font-family: sans-serif;}
div{ border: 5px solid; height: 60px; width: 260px; margin:10px; background-color:yellow;
font-size: 20px; text-align:center; padding-top: 20px; word-wrap:break-word; }
h2{margin: 15px 0 -15px 40px;}
</style>
</head>
<body>
<h2>CSS border-radius aprenderaprogramar.com</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 5px;">border-radius: 5px;</div>
<div style="border-radius: 10px;">border-radius: 10px; </div>
<div style="border-radius: 20px;">border-radius: 20px; </div>
<div style="border-radius: 40px;">border-radius: 40px; </div>
```

© aprenderaprogramar.com, 2006-2029

```
<div style="border-radius: 60px;">border-radius: 60px; </div>
<div style="border: double #A52A2A 8px; border-radius: 13em/3em;">border-radius: 13em/3em; </div>
</div>

<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-top-left-radius: 40%;">border-top-left-radius: 40%; </div>
<div style="border-radius: 80px 20px;">border-radius: 80px 20px; </div>
<div style="border-radius: 60px 20px 5px;">border-radius: 60px 20px 5px; </div>
<div style="border-radius: 60px 20px 5px 90px;">border-radius: 60px 20px 5px 90px; </div>
<div style="border-radius: 60px/20px;">border-radius: 60px/20px; </div>
<div style="border: solid #A52A2A 5px; border-radius: 50%;">border-radius: 50%; </div>
</div>
</body>
</html>
```

El resultado que se obtiene en un navegador que acepte estas propiedades será similar a este:




© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

En numerosas ocasiones queremos crear un diseño para una página web o aplicación parecido al que nos encontramos en otra página web. Supón que has encontrado el siguiente diseño. Crea el código HTML y CSS para lograr un resultado lo más parecido posible a lo que se ve en la siguiente imagen, donde tenemos una tabla con bordes redondeados.

| Standard | Professional | Enterprise |
|---|---|---|
| ✔ | ✔ | ✔ |
| ✔ | ✔ | ✔ |
| ✘ | ✔ | ✔ |
| ✘ | ✘ | ✔ |
| $99 | $199 | $399 |

Orientación:

Crea el código de la tabla en HTML. Define una clase para la tabla (p.ej. class="tabla-destacada"), define el elemento thead para poner ahí la fila y celdas de encabezado. Define el contenido de la tabla dentro de tbody. Define la última fila como tfoot.

Busca imágenes en internet ó para introducirlas como símbolo de check o aspa en las casillas correspondientes.

Vete aplicando propiedades CSS por partes hasta lograr el efecto deseado.

Nota: no tienes que conseguir exactamente el mismo aspecto de la imagen, sólo lo más aproximado posible.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01059D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**EFECTO SOMBRA EN CSS**

Ya conocemos la propiedad text-transform para aplicar sombras y efectos tridimensionales a texto, así como la propiedad border-style que permite generar algunos efectos tipo sombra pero en realidad bastante limitados. La propiedad box-shadow facilita la introducción de sombras y efectos 3D. Este efecto antiguamente no era fácil de conseguir y se solía recurrir al uso de imágenes u otras técnicas.

Con la propiedad box-shadow se ha hecho posible conseguir sombras y efectos 3D en las cajas contenedoras CSS con facilidad. Ten en cuenta que esta propiedad no es soportada por muchas de las versiones de navegadores que no son recientes, e incluso en algunos de los recientes todavía es posible que se considere experimental, lo que obliga al uso de prefijos específicos de navegador, aunque la mayor parte de los navegadores actuales ya las han introducido como estándar y no requieren de prefijo.

**PROPIEDAD BOX-SHADOW**

La sintaxis a emplear se basa en indicar uno o más efectos de sombra separados por comas. Cada efecto de sombra comprende hasta 5 parámetros:

```
selectorDeElemento { noEspecificado_none_o_inset_opcional
distancia_horizontal_requerida distancia_vertical_requerida
blurOpcional
spreadOpcional
colorOpcional}
```

Si no se especifica, la sombra será externa (hacia fuera de la caja). Si se escribe la palabra inset la sombra será interna (hacia dentro de la caja).

Blur crea un efecto de difuminado y brillo.

Spread crea un efecto de agrandamiento de la sombra, haciéndola más grande que la propia caja contenedora. Si spread es cero la sombra tiene el mismo tamaño que la caja, pero con un desplazamiento.

Si se especifica un color, la sombra tomará el color especificado. Si no se especifica color, la sombra tomará el valor que por defecto o explícitamente tenga la propiedad color de aplicación.

© aprenderaprogramar.com, 2006-2029

## PROPIEDAD BOX-SHADOW

| PROPIEDAD CSS box-shadow | |
|---|---|
| **Función de la propiedad** | Permite crear sombras hacia fuera o hacia dentro de una caja contenedora. Se pueden especificar varios efectos separando su especificación por comas. |
| **Valor por defecto** | none |
| **Aplicable a** | Todos los elementos. |
| **Valores posibles para esta propiedad** | none (elimina el efecto de sombra) |
| | Una especificación sintáctica con un desplazamiento horizontal (offset-x) y vertical (offset-y) en unidades de medida válidas (no se admiten %). |
| | Una especificación sintáctica que además de los desplazamientos puede incluir opcionalmente: el valor inset para indicar que la proyección de sombra debe ser hacia el interior, un valor de efecto blur en una unidad de medida válida (no se admiten porcentajes), un valor de efecto spread en una unidad de medida válida (no se admiten porcentajes) y un valor de color. |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer { box-shadow: -5px -5px; }<br>.myContainerSP { box-shadow: inset 0 0 15px 0 maroon; } |

Si el contenedor tiene un borde con forma distinta a la rectangular establecida con border-radius, la sombra establecida con box-shadow adopta la forma que tenga el borde.

Si los desplazamientos se establecen a cero, la sombra no se visualiza excepto si se establecen valores para el efecto blur o spread, en cuyo caso se crea una sombra uniforme en torno a toda la caja contenedora.

Los desplazamientos con valores positivos son hacia la derecha para el horizontal y hacia abajo para el vertical. Los desplazamientos con valores negativos son hacia la izquierda para el horizontal y hacia arriba para el vertical.

Supongamos que se especifican varias sombras: box-shadow: 0.5em -0.5em 0.4em red, 0.5em 0.5em 0.4em gold, -0.5em 0.5em 0.4em lime, -0.5em -0.5em 0.4em blue;

En este caso, la sombra relacionada en primer lugar se sitúa encima del resto de sombras (que pueden quedar ocultas por esta, parcial o totalmente). En este ejemplo la sombra roja estaría encima de la sombra oro, esta encima de la sombra lima y esta encima de la sombra azul.

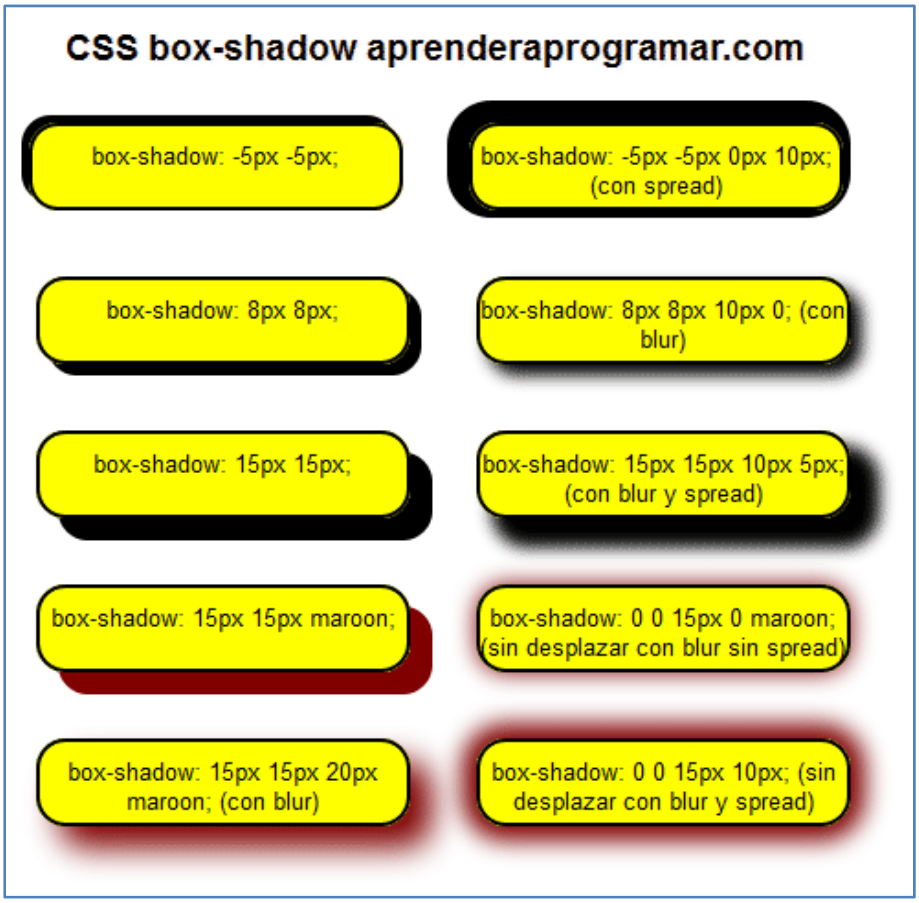
## EJEMPLO DE USO DE BOX-SHADOW

Escribe este código y visualiza sus resultados. Con los contenidos que hemos explicado a lo largo del curso debes ser capaz de interpretar todo el código que hemos incluido. También debes ser capaz de valorar las ventajas y desventajas que puede tener usar este tipo de propiedades.

© aprenderaprogramar.com, 2006-2029

```
<html> <head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{margin:0; padding:0; font-family: sans-serif;}
div{ border: 3px solid;  height: 45px; width: 250px; margin:45px 25px; background-color:yellow;
font-size: 16px; text-align:center; padding-top: 10px; word-wrap:break-word; }
div:first-child {margin:10px 20px 0 20px;} h2{margin: 15px 0 -45px 70px;}
</style> </head>
<body>
<h2>CSS box-shadow aprenderaprogramar.com</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 20px; box-shadow: -5px -5px;">box-shadow: -5px -5px;</div>
<div style="border-radius: 20px; box-shadow: 8px 8px;">box-shadow: 8px 8px;</div>
<div style="border-radius: 20px;box-shadow: 15px 15px;">box-shadow: 15px 15px; </div>
<div style="border-radius: 20px; box-shadow: 15px 15px maroon;">box-shadow: 15px 15px maroon; </div>
<div style="border-radius: 20px; box-shadow: 15px 15px 20px maroon;">box-shadow: 15px 15px 20px maroon; (con blur) </div>
</div>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 20px; box-shadow: -5px -5px 0px 10px;">box-shadow: -5px -5px 0px 10px; (con spread)</div>
<div style="border-radius: 20px; box-shadow: 8px 8px 10px 0;">box-shadow: 8px 8px 10px 0; (con blur)</div>
<div style="border-radius: 20px;box-shadow: 15px 15px 10px 5px;">box-shadow: 15px 15px 10px 5px; (con blur y spread) </div>
<div style="border-radius: 20px;box-shadow: 0 0 15px 0 maroon;">box-shadow: 0 0 15px 0 maroon; (sin desplazar con blur sin spread) </div>
<div style="border-radius: 20px;box-shadow: 0 0 15px 10px maroon;">box-shadow: 0 0 15px 10px; (sin desplazar con blur y spread) </div>
</div>
</body>
</html>
```

El resultado que se obtiene en un navegador que acepte estas propiedades será similar a este:




© aprenderaprogramar.com, 2006-2029

**EJERCICIO RESUELTO**

Invertir todas las sombras del ejemplo anterior para que se proyecten hacia dentro de las cajas contenedoras y no hacia fuera y comprobar los resultados comparándolos con los anteriores.

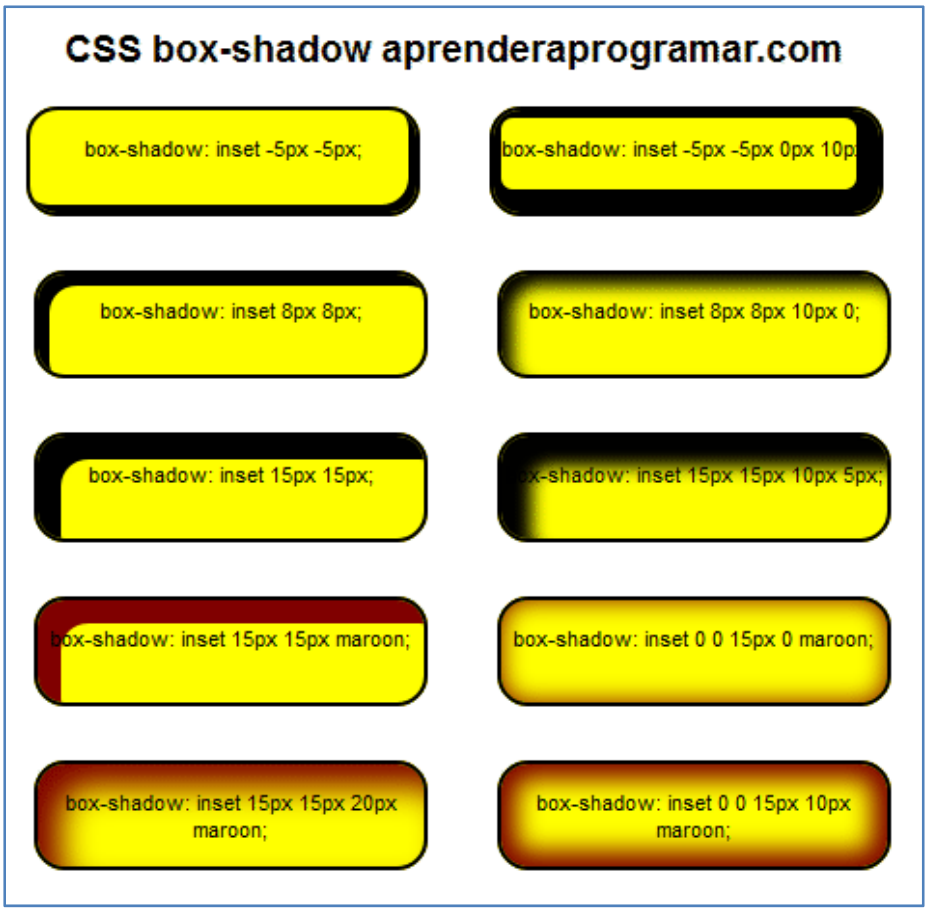
**SOLUCIÓN**

Tenemos que añadir la especificación inset para las sombras. El código sería el siguiente:

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">

<style type="text/css">

*{margin:0; padding:0; font-family: sans-serif;}

div{ border: 3px solid;
height: 50px; width: 250px;
margin:35px 25px; background-color:yellow;
font-size: 14px; text-align:center;
padding-top: 16px; word-wrap:break-word;
}

div:first-child {margin:0 20px 0 20px;}

h2{margin: 15px 0 -45px 70px;}

</style>

</head>
<body>
<h2>CSS box-shadow aprenderaprogramar.com</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 20px; box-shadow: inset -5px -5px;">box-shadow: inset -5px -5px;</div>
<div style="border-radius: 20px; box-shadow: inset 8px 8px;">box-shadow: inset 8px 8px;</div>
<div style="border-radius: 20px;box-shadow: inset 15px 15px;">box-shadow: inset 15px 15px; </div>
<div style="border-radius: 20px; box-shadow: inset 15px 15px maroon;">box-shadow: inset 15px 15px maroon; </div>
<div style="border-radius: 20px; box-shadow: inset 15px 15px 20px maroon;">box-shadow: inset 15px 15px 20px maroon;
</div>
</div>

<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 20px; box-shadow: inset -5px -5px 0px 10px;">box-shadow: inset -5px -5px 0px 10px;</div>
<div style="border-radius: 20px; box-shadow: inset 8px 8px 10px 0;">box-shadow: inset 8px 8px 10px 0;</div>
<div style="border-radius: 20px;box-shadow: inset 15px 15px 10px 5px;">box-shadow: inset 15px 15px 10px 5px; </div>
<div style="border-radius: 20px;box-shadow: inset 0 0 15px 0 maroon;">box-shadow: inset 0 0 15px 0 maroon; </div>
<div style="border-radius: 20px;box-shadow: inset 0 0 15px 10px maroon;">box-shadow: inset 0 0 15px 10px maroon; </div>
</div>
</body>
</html>
```

© aprenderaprogramar.com, 2006-2029

El resultado que se obtiene en un navegador que acepte estas propiedades será similar al mostrado a continuación.




**Próxima entrega:** CU01060D

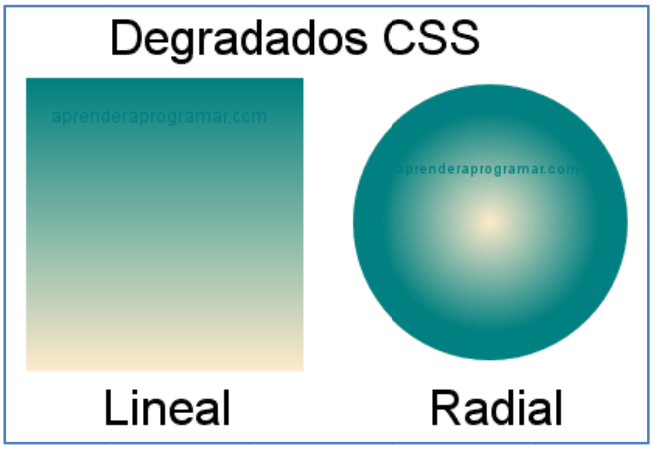
**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## DEGRADADOS EN CSS

Un color que se va degradando progresivamente hasta convertirse en otro es un efecto muy usado para fondos de elementos (o incluso como fondo web) ya que permite conseguir un efecto atractivo. Este efecto antiguamente no era fácil de conseguir y se solía recurrir al uso de imágenes de fondo. Actualmente disponemos de mecanismos que facilitan incluir degradados en páginas web.

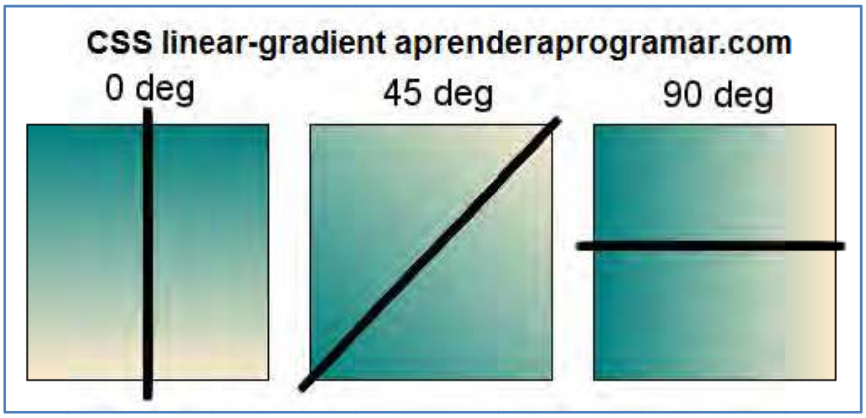
Los degradados CSS son de dos tipos: degradados lineales (linear-gradient) o degradados radiales (radial-gradient). En el primer caso la transformación de color va avanzando línea a línea, mientras que en el segundo caso la transformación de color se produce porque sucesivos círculos concéntricos van cambiando de color. Aquí vemos la diferencia entre ambos efectos.




Un degradado lineal es una imagen que genera CSS a través de la invocación de una función denominada linear-gradient.

## LÍNEA DE GRADIENTE

Un gradiente lineal se define a partir de una línea de gradiente. La imagen se genera creando líneas cuyo color va cambiando, perpendiculares a la línea de gradiente. Aquí vemos la línea de gradiente para distintos degradados:




© aprenderaprogramar.com, 2006-2029

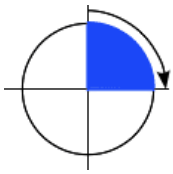
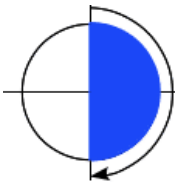
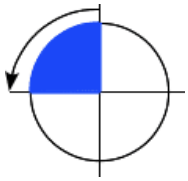
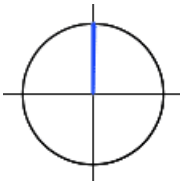
En el primer caso la línea de gradiente va de arriba hacia abajo, por lo que las líneas de color son horizontales. En el segundo caso la línea de gradiente forma un ángulo de 45 grados y esto hace que las líneas de color formen un ángulo respecto a la caja contenedora. En el tercer caso la línea de gradiente es horizontal y esto hace que las líneas de color sean verticales.

Como vemos, nos puede resultar de interés trabajar indicando un grado de inclinación específico para la línea de gradiente.

## UNIDADES ANGULARES CSS

CSS permite la especificación de un ángulo o inclinación en unidades denominadas unidades angulares. Las unidades admitidas son grados deg, grados grad, tantos por uno de circunferencia turn y radianes.

Como en CSS las designaciones de orden siguen la definición top, right, bottom, left, las unidades angulares consideran que el valor 0 se corresponde con la vertical (es decir, no se sigue el mismo criterio que en matemáticas donde 0 grados se corresponden con la horizontal). En la siguiente tabla vemos las unidades angulares y su significado gráfico.

| Visión gráfica | Equivalencia | Comentarios |
|---|---|---|
|  | 90deg = 100grad = 0.25turn ≈ 1.5708rad | Un valor positivo desplaza el ángulo en sentido horario. Para linear-gradient 90deg equivale a to right (degradar desde la izquierda hacia la derecha). |
|  | 180deg = 200grad = 0.5turn ≈ 3.1416rad | Un valor de 180deg para linear-gradient equivale a to bottom (degradar desde arriba hacia abajo). |
|  | -90deg = -100grad = -0.25turn ≈ -1.5708rad | Un valor negativo desplaza el ángulo en sentido antihorario. -90deg con linear-gradient equivale a to left (degradar desde la derecha hacia la izquierda) |
|  | 0deg = 0grad = 0turn = 0rad | Un valor 0deg equivale a la posición de partida. Con linear-gradient equivale a to top (degradar desde abajo hacia arriba). |

© aprenderaprogramar.com, 2006-2029

**FUNCIÓN LINEAR-GRADIENT**

Esta función nos permite generar una imagen con un degradado de colores con una dirección y colores especificados. Tener en cuenta que linear-gradient no genera un color de fondo, sino una imagen sin dimensiones especificadas, que se adaptará automáticamente para cubrir todo el espacio disponible.

Esta función no es soportada por muchas de las versiones de navegadores que no son recientes (o bien la soportan con sintaxis específicas para cada navegador o versión), e incluso en algunos de los recientes todavía es posible que se considere experimental. Esto obliga a que debamos pensar si conviene hacer uso de prefijos específicos de navegador. Nosotros trabajaremos aquí sin estos prefijos porque la mayor parte de los navegadores actuales ya la han introducido como estándar y no requieren de prefijo.

La función se suele invocar normalmente como fondo cuando se usa la propiedad background-image o cuando se usa la propiedad shortand background (aunque quizás a ti se te ocurran otros usos que podrías también probar). Habitualmente la sintaxis será de este tipo:

```
selectorDeElemento { background: linear-gradient
anguloOespecificaciónDeDirección  colorInicial  colorFinal}
```

Ejemplo: style="background: linear-gradient (45deg, #008080, #FFEBCD);

La especificación de dirección se puede hacer con una unidad angular o usando alguno de los siguientes valores:

a) **to bottom:** indica que se comienza con el color inicial arriba (top) y se progresa hasta el color final abajo (bottom). Equivale a un ángulo de 180deg.
b) **to top:** indica que se comienza con el color inicial abajo (bottom) y se progresa hasta el color final arriba (top). Equivale a un ángulo de 0deg.
c) **to right:** indica que se comienza con el color inicial a la izquierda (left) y se progresa hasta el color final a la derecha (right). Equivale a un ángulo de 90deg.
d) **to left:** indica que se comienza con el color inicial a la derecha (right) y se progresa hasta el color final a la izquierda (left). Equivale a un ángulo de -90deg ó 270deg.

Pero existen más posibilidades: se puede especificar una lista de colores separados por comas. Ejemplo: style="background: linear-gradient( 90deg, red, blue, yellow); ". El número de colores puede ser el que se desee. Con esta sintaxis cada color ocupa una fracción del espacio disponible. En el espacio anterior, se comienza con el rojo que se va degradando hasta convertirse en azul, luego el azul se va degradando hasta convertirse en rojo y cada color ocupa un tercio del espacio disponible por ser tres colores. Si fueran 5 colores, cada color ocuparía una quinta parte del espacio disponible.

Otra posibilidad es especificar puntos de localización (stop points) de los colores, de forma que no se distribuyan a partes iguales, sino como nosotros especifiquemos. La sintaxis es de tipo:

style="background: linear-gradient( 90deg, red, blue 20px, yellow); " signfica que se comienza en el color rojo que se va degradando hasta convertirse en azul a una distancia de 20px desde el inicio. A partir de esos 20px el azul se va transformando en amarillo hasta ser completamente amarillo al final del degradado.

© aprenderaprogramar.com, 2006-2029

Otro ejemplo:

style="background: linear-gradient( 90deg, blue, red 30px, yellow 120px, green); " significa que se comienza en el color azul que se degrada hasta convertirse en rojo a los 30px del inicio, desde ahí el rojo se degrada hasta convertirse en amarillo a los 120px del inicio y a partir de ahí el amarillo se degrada hasta convertirse en verde al final.

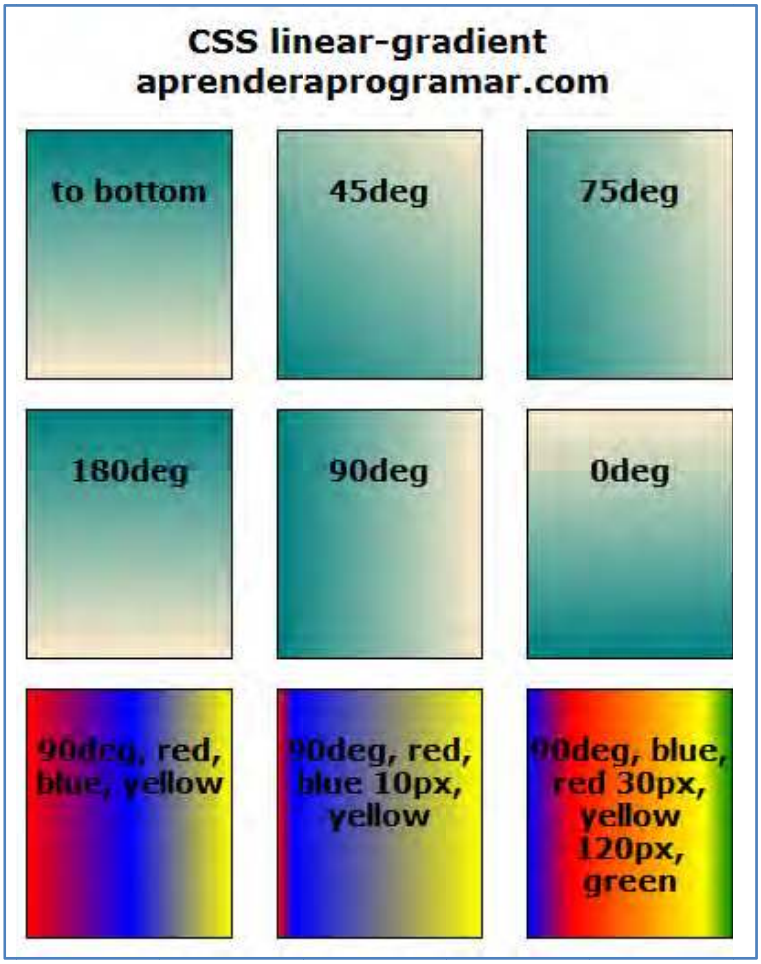
**EJEMPLOS DE USO DE LINEAR-GRADIENT**

Escribe este código y visualiza sus resultados. Con los contenidos que hemos explicado a lo largo del curso debes ser capaz de interpretar todo el código que hemos incluido. También debes ser capaz de valorar las ventajas y desventajas que puede tener usar esta función.

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{margin:0; padding:0; font-family: verdana, sans-serif;}
div{ border: 1px solid; float: left;
height: 140px; width: 140px;
margin:10px 15px;
font-size: 20px; font-weight:bold; text-align:center;
padding-top: 30px; word-wrap:break-word;
}
h2{margin: 55px 0 -30px 65px; font-size:22px; white-space:pre;
text-align:center; width: 400px;}
</style>
</head>
<body>
<h2>CSS linear-gradient
 aprenderaprogramar.com</h2>
<div style=" width:600px; border-style:none; border-width:0; background-color:white;">
<div style=" background: linear-gradient( to bottom, #008080, #ffebcd); ">to bottom</div>
<div style=" background: linear-gradient( 45deg, #008080, #ffebcd); ">45deg</div>
<div style="background: linear-gradient( 75deg, #008080, #ffebcd); ">75deg</div>
<div style="background: linear-gradient( 180deg, #008080, #ffebcd); ">180deg</div>
<div style="background: linear-gradient( 90deg, #008080, #ffebcd); ">90deg</div>
<div style="background: linear-gradient( 0deg, #008080, #ffebcd); ">0deg</div>
<div style="background: linear-gradient( 90deg, red, blue, yellow); ">90deg, red, blue, yellow</div>
<div style="background: linear-gradient( 90deg, red, blue 10px, yellow); ">90deg, red, blue 10px, yellow</div>
<div style="background: linear-gradient( 90deg, blue, red 30px, yellow 120px, green); ">90deg, blue, red 30px, yellow 120px,
green</div>
</div>
</body>
</html>
```

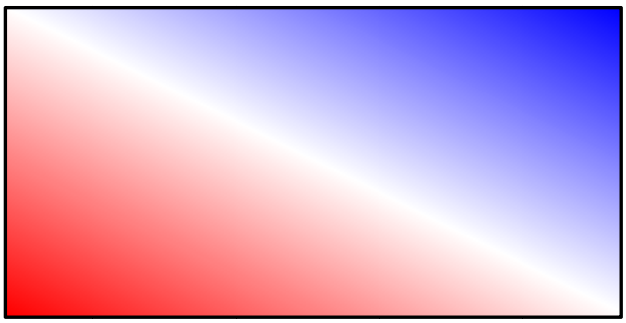
El resultado que se obtiene en un navegador que acepte esta función será similar a este:

© aprenderaprogramar.com, 2006-2029




**EJERCICIO**

Crea un documento HTML donde tengas un contenedor div centrado, con márgenes de 50 píxeles en todas direcciones y dimensiones ancho 400 píxeles y alto 200 píxeles. Crea el efecto que se ve en la siguiente imagen dentro de dicho contenedor usando la propiedad CSS linear-gradient.



Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01061D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**MÁS EFECTOS CSS**

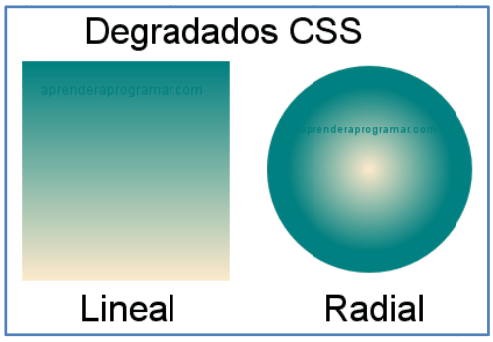
A lo largo del curso hemos ido viendo cómo CSS ha ido introduciendo progresivamente cada vez mayores posibilidades para generar efectos visuales atractivos que hace unos años resultaban complicados de generar. Además de las propiedades que ya hemos visto como text-shadow, box-shadow, etc. CSS está en constante evolución e incorpora cada vez más posibilidades.

Ya comentamos en su momento que el objetivo de este curso era aprender la lógica de CSS y tener un conocimiento sólido de sus fundamentos y no ser un manual de referencia donde se especificara toda la sintaxis y todas las posibilidades de esta tecnología. En esta entrega vamos a comentar algunas propiedades o efectos que son interesantes. Para conocer su sintaxis te recomendamos consultes alguna web de referencia CSS de entre las que indicamos en http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=727&catid=75:tutorial-basico-programador-web-css-desde-cero&Itemid=203

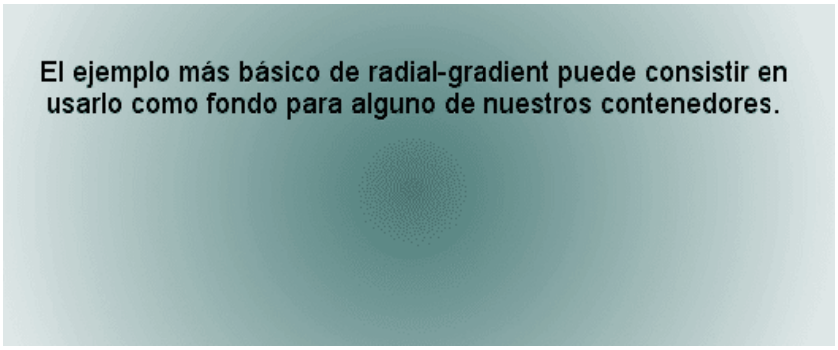
Nosotros usamos habitualmente Mozilla Developer Network como web de referencia para especificación y sintaxis, pero elige tú la que te parezca más adecuada.

**RADIAL-GRADIENT**

Al igual que linear-gradient servía para generar un degradado lineal, radial-gradient permite generar degradados circulares o elípticos. Aquí vemos la diferencia entre lineal y radial:




Y aquí algunas posibilidades que tendríamos usando CSS radial-gradient:

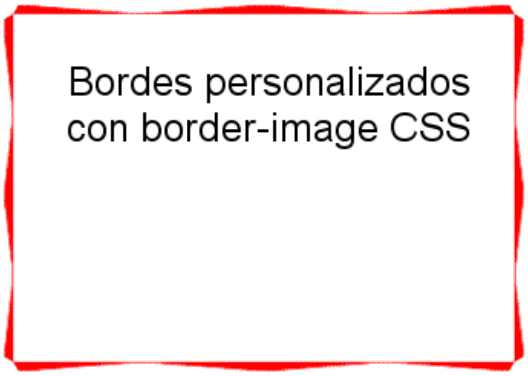



© aprenderaprogramar.com, 2006-2029

Ten en cuenta que estos efectos no son posibles en los navegadores más antiguos y que los navegadores actuales pueden requerir el uso de prefijos o tener distintas posibilidades o respuestas según la versión y navegador que se utilice. Ten en cuenta que aunque conviene conocer las posibilidades de CSS debemos valorar las ventajas y desventajas de todo aquello que usemos en nuestros desarrollos web. Como alternativa más estándar, tenemos el uso de background-image para mostrar las imágenes de degradado radial. Esto tiene sus ventajas y sus inconvenientes.

**BORDER-IMAGE CSS**

Los bordes en CSS se controlan principalmente mediante la propiedad border (y en menor medida con la propiedad outline). La propiedad border-image se ideó para tratar de facilitar el uso de imágenes para constituir bordes con diseños personalizados, de modo que cada desarrollo web pudiera tener los bordes que deseara a través de imágenes que definen un fragmento del borde y se repiten o colocan de una manera determinada especificada mediante código.




© aprenderaprogramar.com, 2006-2029

Ten en cuenta que estos efectos no son posibles en los navegadores más antiguos y que los navegadores actuales pueden requerir el uso de prefijos o tener distintas posibilidades o respuestas según la versión y navegador que se utilice. Ten en cuenta que aunque conviene conocer las posibilidades de CSS debemos valorar las ventajas y desventajas de todo aquello que usemos en nuestros desarrollos web. Como alternativa más estándar, tenemos el uso de background-image para generar los bordes. Esto tiene sus ventajas y sus inconvenientes.

**RECORDATORIO**

CSS está incorporando continuamente posibilidades y novedades. Muchas de ellas tratan de permitir a los desarrolladores web trabajar sin usar carga de imágenes localizadas en archivos para crear los efectos deseados. No obstante, ten en cuenta que las imágenes procedentes de archivos no están sujetas a interpretación por parte del navegador, mientras que los efectos generados mediante código sí, lo que puede dar lugar a que la visualización en distintos navegadores no sea coincidente o requiera de "parches". Por eso los efectos CSS no son utilizados por muchos desarrolladores, que prefieren ser más conservadores y seguir utilizando imágenes tradicionales. Por otro lado, otros desarrolladores utilizan todos los efectos CSS posibles en sustitución de las imágenes tradicionales. Nosotros te recomendamos que razones para cada ocasión qué es lo más conveniente, teniendo en cuenta factores como velocidad de carga de la página y compatibilidad entre navegadores entre otros aspectos.

**EJERCICIO**

Busca en internet (en un blog, página web, etc.) una imagen para crear un borde personalizado con border-image. Crea un documento HTML con un div de 400 píxeles de ancho y 200 píxeles de alto en cuyo interior muestre un degradado radial y aplícale el borde personalizado. Ten en cuenta las peculiaridades de tu navegador.

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01062D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

## MÁS EFECTOS CSS

La propiedad CSS transform permite generar efectos gráficos que antes no era posible generar con CSS, como efecto de rotación, escalado completo, escalado horizontal, escalado vertical, sesgado horizontal o vertical y traslación horizontal o vertical de elementos.

## PROPIEDAD TRANSFORM

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales.

| PROPIEDAD CSS transform | |
|---|---|
| **Función de la propiedad** | Permite rotar, escalar, sesgar o trasladar elementos. |
| **Valor por defecto** | none |
| **Aplicable a** | Elementos transformables (tipo block o equivalente). |
| **Valores posibles para esta propiedad** | none (indica que no hay transformación) |
| | rotate (udAng) (donde udAng es una unidad angular válida en CSS) |
| | scale (valorX) ó scale (valorX, valorY) ó scaleX(valorX) ó scaleY(valorY) (donde la especificación X fuerza el escalado en horizontal y la especificación Y lo fuerza en vertical, siendo ambas números enteros o decimales) |
| | skewX (udAng) ó skewY(udAnd) (donde udAng es una unidad angular válida en CSS. Se establece un sesgado del elemento en horizontal o en vertical) |
| | translate (valorX) ó translate (valorX, valorY) ó translateX(valorX) ó translateY(valorY) (donde la especificación X fuerza la traslación en horizontal y la especificación Y lo fuerza en vertical, siendo ambas unidades de medida válidas admitiéndose porcentajes) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer { skewX(-30deg);}<br>.myContainerSP { scale(0.5);} |

Esta propiedad permite generar efectos muy vistosos cuando se aprovecha para combinarla con hover, por ejemplo: #contentBox1:hover { transform: skewX(-20deg);} da lugar a que el efecto se produzca cuando el usuario pase el ratón por encima del elemento afectado, y mientras esto no ocurre el elemento se muestra como esté definido para una situación normal. Este tipo de efectos anteriormente no se podían generar con CSS y era necesario hacerlo mediante imágenes previamente preparadas con un programa de diseño gráfico, con Javascript u otras técnicas.

© aprenderaprogramar.com, 2006-2029

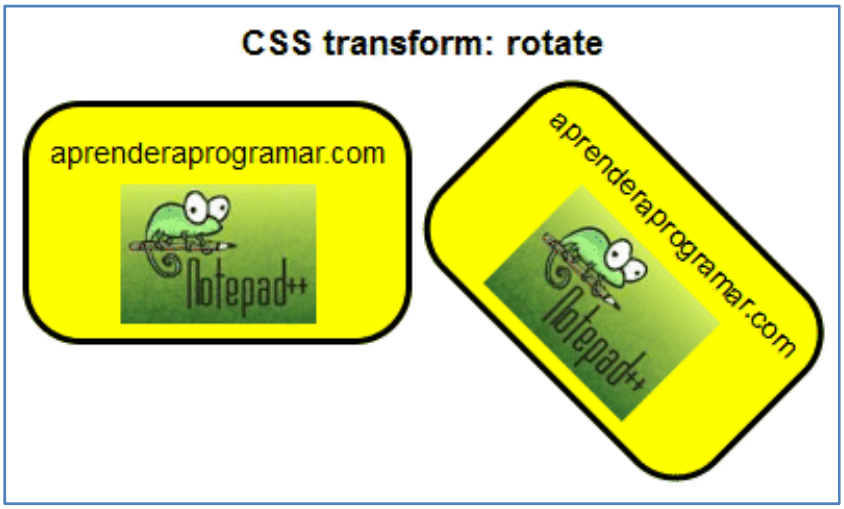
Escribe el siguiente código HTML y comprueba los resultados de esta propiedad (ten en cuenta que es posible que necesites añadir prefijos, depende del navegador que estés usando):

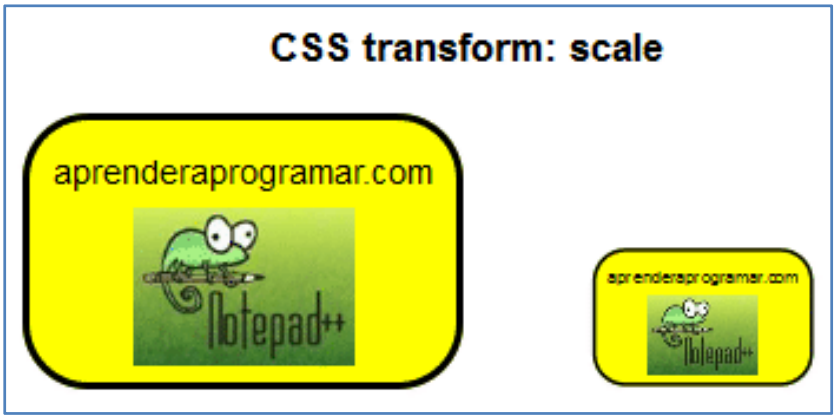
```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{margin:0; padding:0; font-family: sans-serif;}
div{ border: 5px solid; width: 260px; margin:10px; background-color:yellow; font-size: 20px; text-align:center;
padding-top: 20px; word-wrap:break-word; }
h2{margin: 15px 0 -15px 40px;} img {padding:10px; }
#contentBox1 {border-radius: 40px; margin-left:40px;}
#contentBox1:hover { transform:  skewX(-20deg);}
}
</style> </head>
<body>
<!--ROTATE-->
<h2>CSS transform: rotate</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px;">aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px; margin-top: 50px; transform: rotate(45deg);">aprenderaprogramar.com<img
src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<p style="clear: both; "></p>
<!--SCALE-->
<h2 style="margin-top:80px;">CSS transform: scale</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px;">aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px; margin-top: 50px; transform: scale(0.5);">aprenderaprogramar.com<img
src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<p style="clear: both; "></p>
<!--SKEWX-->
<h2>CSS transform: skewX</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px;">aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px; margin-left: 40px; transform: skewX(-30deg);">aprenderaprogramar.com<img
src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<p style="clear: both; "></p>
<!--SKEWY-->
<h2>CSS transform: skewY</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px;">aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px; margin-left: 40px; transform: skewY(-20deg);">aprenderaprogramar.com<img
src="http://i.imgur.com/afC0L.jpg" ></a></div>
```

© aprenderaprogramar.com, 2006-2029

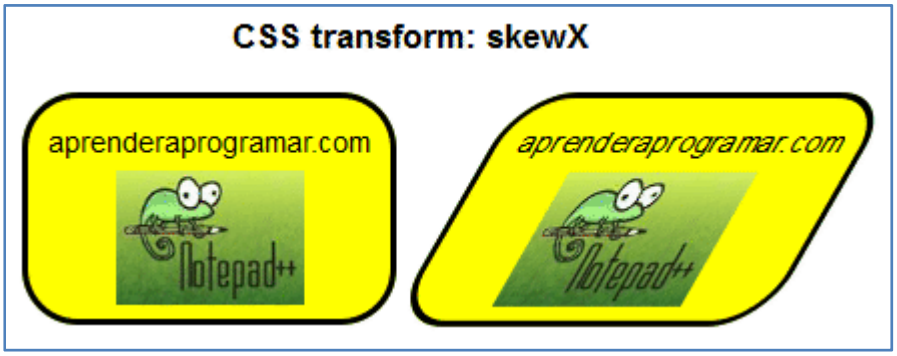
```
</div>
<p style="clear: both; "></p>
<!--TRANSLATE-->
<h2>CSS transform: translate</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px;">aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div style="border-radius: 40px; margin-left: 40px; transform: translate(70px, 70px);">aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<p style="clear: both; "></p>
<!--COMBINADO CON HOVER-->
<h2>CSS transform con hover</h2>
<div style="float:left; border-style:none; border-width:0; background-color:white;">
<div id="contentBox1" >aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg" ></a></div>
</div>
<p style="clear: both; "></p>
</body>
</html>
```

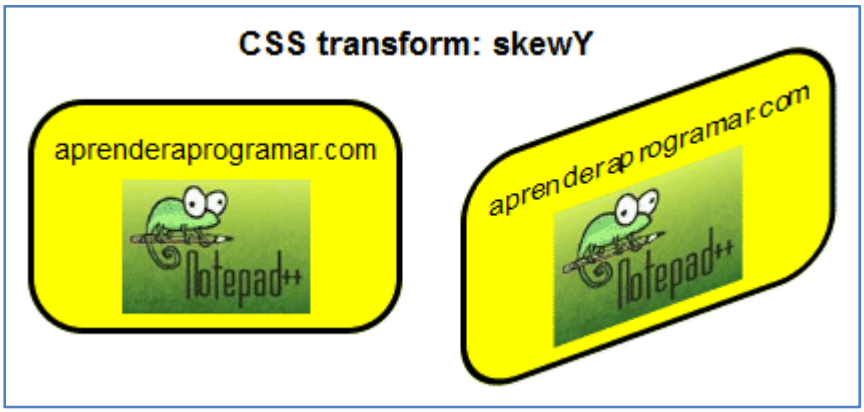
El resultado que se obtiene en un navegador que acepte estas propiedades será similar a este:

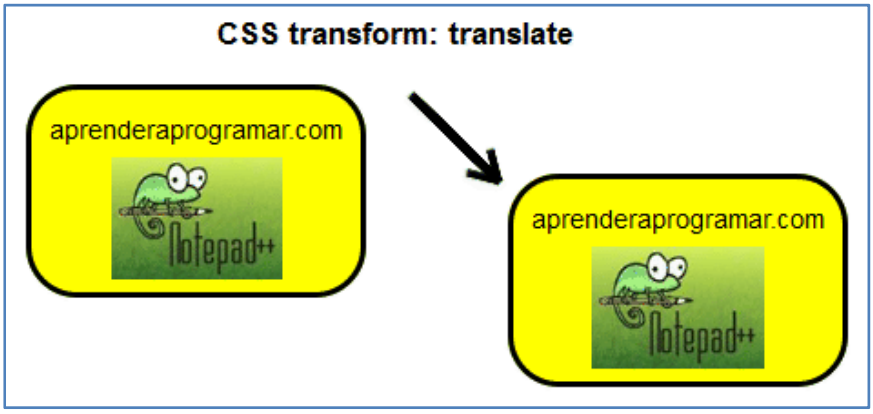



© aprenderaprogramar.com, 2006-2029




**EJERCICIO**

Estudia el siguiente código CSS y responde a las cuestiones planteadas:

```
#skew {   transform:skew(35deg); }
#scale {  transform:scale(1,0.5); }
#rotate {   transform:rotate(45deg); }
#translate {   transform:translate(10px, 20px); }
#rotate-skew-scale-translate {   transform:skew(30deg) scale(1.1,1.1) rotate(40deg) translate(10px, 20px); }
```

© aprenderaprogramar.com, 2006-2029

a) Crea un documento HTML donde se vean los estilos que tenemos en el código aplicados a distintos elementos.

b) Explica el significado de cada una de las partes del código (por ejemplo #skew indica el estilo que se aplicará a los elementos con atributo id="skew". Transform:skew(35deg) indica que ...)

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01063D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**MÁS EFECTOS CSS**

Combinando la propiedad transform con hover podemos crear efectos dinámicos sobre una web. Sin embargo, la animación que se puede conseguir con este efecto es bastante limitada. CSS ha introducido nuevas técnicas que permiten elaborar animaciones más sofisticadas que pueden colaborar a no tener que recurrir a tecnologías como Javascript ó Flash.

**CONCEPTO DE TRANSICIÓN CSS**

Una transición CSS es un efecto de cambio progresivo desde un estado inicial hasta un estado final de un elemento HTML. En el estado inicial el elemento tiene unas propiedades y en el estado final tiene una o varias propiedades que han cambiado. El navegador se encarga de generar los pasos intermedios que suponen cambiar de un estado a otro, de la forma que especifiquemos a través del código.

Una transición CSS empieza cuando se produce una situación de "disparo" del cambio. El disparo del cambio puede ser generado de distintas maneras, pero la forma más básica y la que vamos a ver en los ejemplos que pondremos a continuación es que el cambio empieza cuando el usuario pone el ratón encima del elemento HTML afectado por la transición, es decir, que usamos la pseudoclase :hover para definir la transición que empieza cuando el usuario pone el puntero del mouse encima del elemento.

Una transición empieza y termina y no se repite indefinidamente (a no ser que el usuario vuelva a poner el ratón encima del elemento otra vez), aunque veremos más adelante que CSS ofrece más posibilidades para generar animaciones.

Mediante transiciones CSS podemos generar efectos dinámicos que anteriormente sólo se podían generar recurriendo a otros lenguajes o tecnologías. Constituyen un recurso interesante, pero a la vez peligroso, en el sentido de que se puede caer en la tentación de crear páginas web con gran cantidad de efectos y movimientos pero que resulten demasiado sobrecargadas y poco útiles para los usuarios.

**PROPIEDAD TRANSITION-PROPERTY**

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

CSS permite cambiar las propiedades de un elemento HTML mediante una transición con efecto de animación progresiva. El primer parámetro a definir es qué propiedad es la que se va a cambiar durante la animación. Por ejemplo, podemos cambiar la altura ó la anchura de un elemento. También se pueden cambiar varias propiedades simultáneamente, por ejemplo la altura, la anchura y el color o la imagen de fondo, tamaño de texto, etc. Para definir qué propiedad o propiedades van a ser incluidas en la animación se usa la propiedad transition-property.

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS transition-property | |
|---|---|
| **Función de la propiedad** | Permite definir qué elementos van a ser afectados por una transición. |
| **Valor por defecto** | all |
| **Aplicable a** | Todos los elementos. |
| **Valores posibles para esta propiedad** | **all** (indica que se verán afectadas por la transición todas las propiedades que se pueda) |
| | **none** (indica que no se aplicará la transición a ninguna propiedad) |
| | **nombreDeLaPropiedad** (indicando un nombre de propiedad se especifica la propiedad que se va a ver afectada; si son varias, separarlas con comas.) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer { transition-property: color, transform; }<br>.myContainerSP { transition-property: font-size; } |

**PROPIEDAD TRANSITION-DURATION**

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos. transition-duration sirve para especificar cuál tiene que ser la duración de la transición en segundos (s) o en milisegundos (ms). Su valor por defecto es 0 (esto equivale a que no habrá transición). En caso de que haya varias propiedades afectadas, los tiempos se especifican separados por comas y se asocian a cada propiedad en el mismo orden en que los define la propiedad transition-property. Ejemplo: transition-property: color, transform; transition-duration: 10s, 4s;

En este caso la transición para la propiedad color durará 10 segundos, será más lenta. La transición para la propiedad transform durará 4 segundos, será más rápida.

Otro ejemplo con valores expresados en milisegundos: transition-duration: 120ms, 750ms;

**PROPIEDAD TRANSITION-DELAY**

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

Esta propiedad sirve para especificar cuál tiene que ser el retardo con el que empiece la transición en segundos (s) o en milisegundos (ms), a partir del momento en que se produzca la situación "de disparo". Su valor por defecto es 0 (esto equivale a que la transición comenzará inmediatamente cuando se produzca la situación de disparo). En caso de que haya varias propiedades afectadas, los tiempos se especifican separados por comas y se asocian a cada propiedad en el mismo orden en que los define la propiedad transition-property. Ejemplo:

© aprenderaprogramar.com, 2006-2029

transition-property: color, transform; transition-delay: 2s, 0s;

En este caso la transición para la propiedad empezará con un retardo de 2 segundos, no se hará patente inmediatamente. El retardo para la propiedad transform será de cero segundos, lo que significa que se empezarán a apreciar los efectos de la transición inmediatamente una vez se produzca la situación de disparo.

Otro ejemplo con valores expresados en milisegundos: transition-delay: 120ms, 750ms;

## PROPIEDAD TRANSITION-TIMING-FUNCTION

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

| PROPIEDAD CSS transition-timing-function | |
|---|---|
| **Función de la propiedad** | Permite definir la velocidad a la que transcurre la transición respetando la duración establecida. A efectos prácticos, permite que el usuario vea la animación como progresiva a velocidad constante o con cierta aceleración. |
| **Valor por defecto** | ease |
| **Aplicable a** | Todos los elementos. |
| **Valores posibles para esta propiedad** | **ease**: comienzo rápido, luego velocidad constante y final lento. |
| | **ease-in**: al revés que ease. |
| | **ease-out**: similar a ease con final a velocidad constante. |
| | **ease-in-out**: comienzo lento y progresión a velocidad constante. |
| | **linear**: progresión a velocidad constante. |
| | **step-start**: salto de la situación inicial a la final al comienzo. |
| | **step-end**: salto de la situación inicial a la final al final. |
| | **steps** (numeroDePasos, end) [El parámetro numeroDePasos establece el número de pasos intermedios entre la situación inicial y la final, lo que equivale a que la transición se vea como "pequeños saltos".] |
| | **cubic-bezier**(valor0a1, valor0a1, valor0a1, valor0a1) [Esta forma de definición permite especificar la curva temporal de forma matemática para definir formas de cambio personalizadas] |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer { transition-timing-function:ease;}<br>.myContainerSP { transition-timing-function:steps(6, end);} |

© aprenderaprogramar.com, 2006-2029

**EJEMPLO DE APLICACIÓN**

Escribe el siguiente código y comprueba los resultados (ten en cuenta que para algunos navegadores puede ser necesario que indiques prefijos). El efecto a apreciar es cómo un div con un texto e imagen va rotando y cambiando de color hasta volver a su posición inical pero con el color cambiado.

```
<html>
<head>
<title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
<style type="text/css">
*{margin:0; padding:0; font-family: sans-serif;}

.bloquePub{ border: 5px solid; width: 260px;
margin:60px 0 0 90px; background-color:yellow;
font-size: 20px; text-align:center;
padding-top: 20px; word-wrap:break-word;
transition-property: color, transform;
transition-duration: 6s, 4s;
transition-timing-function: ease;
transition-delay: 2s, 0s;
}

.bloquePub:hover {color: red; transform:rotate(360deg);}

h2{margin: 15px 0 0 140px;}
img {padding:10px; }
}
</style>
</head>
<body>
<h2>CSS transition</h2>
<div id="mainCont" style="border-style:none; border-width:0; background-color:white;">
<div class="bloquePub" style="border-radius: 40px;">aprenderaprogramar.com<img src="http://i.imgur.com/afC0L.jpg"
alt="Notepad++" ></a></div>
</div>
</body>
</html>
```

El mismo efecto se puede conseguir usando la propiedad shortand transition como veremos a continuación.

**PROPIEDAD SHORTAND TRANSITION**

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

Las funciones relacionadas con transiciones transition-property, transition-duration, transition-delay y transition-timing-function se pueden agrupar en una sola propiedad shortand denominada transition. La sintaxis a emplear es de este tipo:

© aprenderaprogramar.com, 2006-2029

```
selectorDeElemento { valorTransitionProperty   valorTransitionDuration
                     valorTransitionDelay      valorTransitionTimingFunction}
```

Para aplicar esta propiedad en el ejemplo visto anteriormente reemplazaríamos las cuatro propiedades relacionadas con transition por esta línea:

transition: color 6s ease 2s, transform 4s ease 0s;

El efecto lo vemos descrito en esta secuencia de imágenes:

**LO ELEGANTE Y LO EXCESIVO**

Con las propiedades que estamos viendo que permiten efectos dinámicos usando CSS muchos diseñadores y programadores son capaces de generar efectos elegantes y atractivos para páginas web. Sin embargo, muchos otros llenan sus páginas web de complejos efectos visuales que son muy espectaculares, pero luego resultan desagradables para el usuario debido a que en la página todo se está moviendo y dando vueltas, lo que puede llegar incluso a generar mareos. Por tanto, concluimos con una recomendación. Si usas estos efectos, hazlo con prudencia y moderación.

© aprenderaprogramar.com, 2006-2029

**EJERCICIO**

Estudia el siguiente código CSS y responde a las cuestiones planteadas:

```
#transEj1 div {  transition:all 2s ease-in-out;  perspective: 800px;  perspective-origin: 50% 100px; }
#transEj1:hover #rotateX {  transform:rotateX(180deg);}
#transEj1:hover #rotateY {  transform:rotateY(180deg);}
#transEj1:hover #rotateZ {  transform:rotateZ(180deg);}
```

a) Crea un documento HTML donde se vean los estilos que tenemos en el código aplicados a distintos elementos.

b) Explica el significado de cada una de las partes del código (por #transEj1 div indica los estilos que se aplicarán a div dentro de elementos con id="transEj1". transition: all indica que .., 2s indica que ..., ease-in-out indica que.., etc.)

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01064D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**ANIMATION CSS**

Ya hemos visto que usando las propiedades transform y transition con hover podemos crear efectos dinámicos sobre una web. Sin embargo, la animación está limitada a que se produzca al posicionar el usuario el ratón sobre un elemento o usar otros recursos. CSS ha introducido nuevas técnicas que permiten elaborar animaciones más sofisticadas que pueden colaborar a no tener que recurrir a tecnologías como Javascript ó Flash.

**CONCEPTO DE ANIMACIÓN CSS**

El concepto de animación es un concepto que tiene cierta similitud con el concepto de transición CSS, pero que es más amplio y ofrece más posibilidades. Una animación es un efecto dinámico que cambia las propiedades de un elemento en el tiempo y que puede tener varios estados intermedios especificados. Una animación requiere al menos de dos estados: el estado inicial (from) y el estado final (to). No obstante, puede tener otros estados intermedios adicionales.

A continuación vemos las diferencias entre transición y animación:

| | Transición | Animación |
|---|---|---|
| **Uso básico** | Cambio de estado inicial a final cuando usuario posiciona mouse encima de elemento | Cambio de estado inicial a otro final, pasando por varios estados intermedios diferentes si se desea y de ejecución espontánea si se desea |
| **Ejecución espontánea fácil de codificar** | No | Sí |
| **Repetición espontánea durante cierto tiempo fácil de codificar** | No | Sí |
| **Repetición continua fácil de codificar** | No | Sí |
| **Ejecución hacia delante y hacia detrás fácil de codificar** | No | Sí |
| **Puede requerir uso de prefijos en ciertos navegadores** | Sí | Sí |

© aprenderaprogramar.com, 2006-2029

**@KEYFRAMES**

Para definir una animación empezaremos definiendo las propiedades que va a tener el elemento en cada uno de los estados de la animación mediante el uso de la regla CSS @keyframes.

Para ello se usa la siguiente sintaxis:

```
@keyframes nombreDeLaAnimacion{
    from {propiedad1: valor1; propiedad2: valor2; ...propiedadN: valorN;  }
    valor%1 { propiedad1: valor; propiedad2: valor; ...propiedadN: valor;  }
    valor%2 { propiedad1: valor; propiedad: valor2; ...propiedad: valorN;  }
    ...
    valor%N { propiedad1: valor; propiedad: valor2; ...propiedad: valorN;  }
    to {propiedad1: valor1; propiedad2: valor2; ...propiedadN: valorN;  }
}
```

Las propiedades definidas en from { ...} definen el estado inicial o situación de partida. Las propiedades definidas en to { ...} definen el estado final o situación de terminación. Estos dos estados son de declaración obligatoria.

Entre los estados from y to pueden existir un número indeterminado de estados intermedios que representan la situación en momentos intermedios durante la ejecución de la animación. La animación va pasando así por los estados definidos hasta llegar al estado final.

El ejemplo más básico puede ser tener sólo un estado inicial y un estado final. Sería este caso:

```
@keyframes animacion{
  from { margin-left: 10%; }
  to { margin-left: 60%; color:red }
}
```

En este ejemplo estamos definiendo que el elemento HTML al que se le aplique la animación que tiene por nombre "animacion" pasará de tener un margin-left del 10%, es decir, estar al lado izquierdo, a tener un margin-left del 60%, es decir, estar al lado derecho de su contenedor. Además en el lado izquierdo tendrá su color por defecto mientras que en lado derecho tomará color rojo. Aquí hemos definidio básicamente un movimiento de izquierda a derecha.

Ahora otra definición con un punto clave intermedio:

```
@keyframes animacion{
  from { margin-left: 10%; }
                50% { margin-left:35%; margin-top:10%; }
                to { margin-left: 60%; }
}
```

© aprenderaprogramar.com, 2006-2029

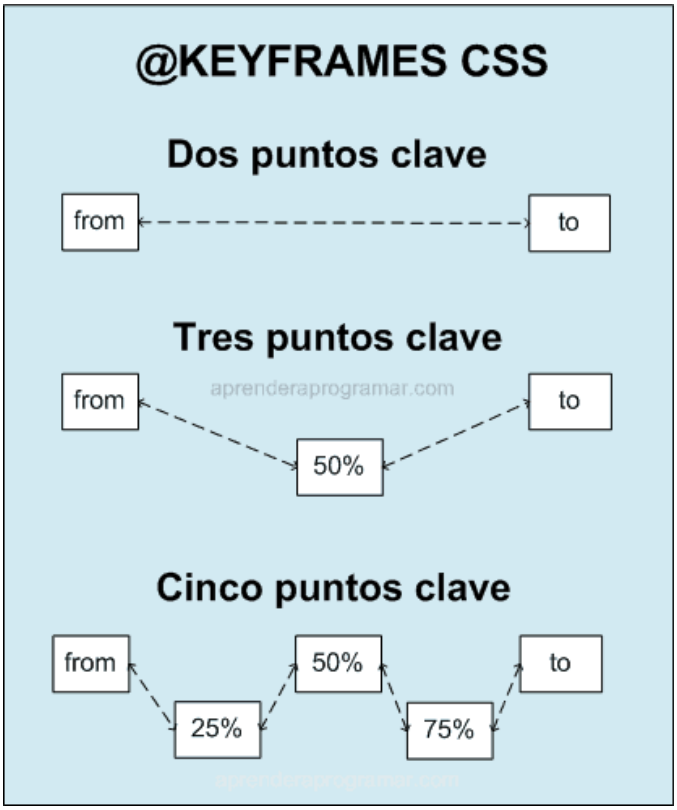
Aquí hemos indicado que cuando la transición esté al 50% de su ejecución el elemento debe estar a medio camino entre la izquierda y la derecha (entre el 10% y el 60% el punto medio resulta el 35%) y además que en ese momento debe tener un desplazamiento hacia abajo debido al margin-top. Hemos definido un movimiento de izquierda a derecha que va bajando hasta alcanzar su posición más baja en el centro y luego sube hasta llegar a su posición en el lado derecho.

Ahora otra definición con más puntos clave:

```
@keyframes animacion{
  from { margin-left: 10%; }
                25% { margin-left:22.5%; margin-top:10%; }
                50% { margin-left:35%; margin-top:0%;}
                75% { margin-left:47.5%; margin-top:10%; }
                to { margin-left: 60%; }
}
```

Con esta definición el elemento partirá de arriba a la izquierda, se irá desplazando hacia la derecha y bajará, luego subirá, luego volverá a bajar y terminará arriba a la derecha. Hemos definido un movimiento de "sube y baja" de arriba a derecha.

Para entender gráficamente lo que suponen estas definiciones podemos ver este esquema:




Nosotros hemos usado como ejemplo que un elemento se va moviendo, pero ten en cuenta que animación no tiene por qué implicar movimiento. Animación implica cambio de propiedades, con lo

© aprenderaprogramar.com, 2006-2029

cual un elemento HTML puede estar quieto y presentar una animación debido a que va cambiando su color, su tamaño de fuente, su fondo, etc. y no es obligatorio que exista movimiento. De hecho, un efecto bastante habitual es hacer brillar un texto, hacer que su color o efecto de gradiente de fondo vaya cambiando, etc.

La regla @keyframes puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

Una vez hemos definido los keyframes pasaremos a la definición de los parámetros que definen la animación mediante las propiedades animation-name, animation-duration, animation-delay, animation-fill-mode, animation-iteration-count, animation-direction y animation-timing-function.

**EJERCICIO**

Estudia el siguiente código CSS y responde a las cuestiones planteadas:

```
@keyframes anime{
   0%{ color: #f00;  font-size: 10px;  top: 10px;}
   25%{color: #A52A2A; font-size: 90px; left: 100px;}
   50%{ top: 56px;  color: #000; font-size: 90px; }
   100%{color: #A52A2A; font-size: 90px; left: 0; }
}
```

a) ¿Cuál es el nombre de la animación? ¿Cuántos puntos clave define?

b) ¿Qué ocurrirá (cuál será el cambio de propiedades) durante la animación?

c) ¿Es equivalente usar 0% en lugar de from? ¿Y 100% en lugar de to?

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01065D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**ANIMATION CSS**

Mediante la regla CSS @keyframes podemos definir una animación con un nombre y los diferentes estados por los que pasará el elemento HTML que va a ser animado durante el tiempo que dure la animación. Conocida esta regla, vamos a ver cómo aplicarla a un elemento o grupo de elementos mediante el uso de un selector tipo id o class y la aplicación de las propiedades relacionadas con animation CSS.

**PROPIEDAD ANIMATION-NAME**

Mediante esta propiedad indicamos el nombre de animación definida en una regla @keyframes que va a ser aplicada a un elemento seleccionado mediante un selector CSS. El valor por defecto para esta propiedad es none (implica que no se aplicará ninguna animación). Si se desean aplicar varias animaciones, se especificarán sus nombres separadas por comas.

Ejemplo: .pubMov{ animation-name: animacion1musk; } esta regla indica que a los elementos con class = "pubMov" se les aplicará la animación de nombre animacion1musk que habrá sido definida en una regla @keyframes.

Otro ejemplo: .pubMov{ animation-name: animacion1musk, animacion7kim; } esta regla indica que a los elementos con class = "pubMov" se le aplicarán dos animaciones: la que lleva por nombre animacion1musk y la que lleva por nombre animacion7kim. Ambas deberán haber sido definidas en reglas @keyframes.

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

**PROPIEDAD ANIMATION-DURATION**

Mediante esta propiedad indicamos el tiempo en segundos (s) o milisegundos (ms) que llevará completar un ciclo de la animación. La animación habrá sido definida en una regla @keyframes y va a ser aplicada a un elemento seleccionado mediante un selector CSS. El valor por defecto para esta propiedad es 0s (implica que no se aplicará ninguna animación ya que la duración es nula). Si se desean aplicar varias animaciones, se especificarán sus duraciones separadas por comas, siendo cada valor correspondiente a la animación cuyo nombre figura en el mismo orden en la propiedad animation-name.

© aprenderaprogramar.com, 2006-2029

Ejemplo: .pubMov{ animation-name: animacion1musk; animation-duration: 5s;} esta regla indica que a los elementos con class = "pubMov" se les aplicará la animación de nombre animacion1musk que habrá sido definida en una regla @keyframes y que esta animación deberá completarse en 5 segundos.

Otro ejemplo: .pubMov{ animation-name: animacion1musk, animacion7kim; animation-duration: 5s, 250ms;} esta regla indica que la primera animación deberá completarse en 5 segundos y la segunda en 250 milisegundos.

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

**PROPIEDAD ANIMATION-ITERATION-COUNT**

Mediante esta propiedad indicamos el número de veces que debe repetirse la animación. La animación habrá sido definida en una regla @keyframes y va a ser aplicada a un elemento seleccionado mediante un selector CSS. El valor por defecto para esta propiedad es 1 (implica que la animación se ejecutará una sola vez y se detendrá). Sus valores posibles son cualquier número igual o superior a cero, admitiéndose decimales (por ejemplo se admite, 1, 2, 3, 3.1, 3.2, 3.5, 10, 20, etc.) ó infinite. Si se usa la palabra clave infinite la animación se repetirá indefinidamente sin parar.

Si se desean aplicar varias animaciones, se especificarán sus repeticiones separadas por comas, siendo cada valor correspondiente a la animación cuyo nombre figura en el mismo orden en la propiedad animation-name.

Ejemplo: .pubMov{ animation-name: animacion1musk; animation-duration: 5s; animation-iteration-count:infinite;} esta regla indica que a los elementos con class = "pubMov" se les aplicará la animación de nombre animacion1musk que habrá sido definida en una regla @keyframes y que esta animación deberá completarse en 5 segundos y que una vez completada volverá a repetirse y así continuamente.

Otro ejemplo: .pubMov{ animation-name: animacion1musk, animacion7kim; animation-duration: 5s, 250ms; animation-iteration-count:infinite, 7;} esta regla indica que la primera animación se repetirá continuamente mientras que la segunda se ejecutará 7 veces y se detendrá.

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

**PROPIEDAD ANIMATION-DIRECTION**

Mediante esta propiedad indicamos cómo debe ejecutarse una animación: hacia delante (del principio al final), hacia atrás (del final al principio), una vez hacia delante y otra hacia detrás, etc. La animación habrá sido definida en una regla @keyframes y va a ser aplicada a un elemento seleccionado mediante

© aprenderaprogramar.com, 2006-2029

un selector CSS. El valor por defecto para esta propiedad es **normal** (implica que la animación se ejecutará hacia delante). Sus valores posibles son cualquiera de estas palabras clave:

- **normal:** la animación se ejecutará hacia delante. Si se repite, cuando vuelve a empezar parte de la situación inicial.
- **reverse:** la animación se ejecutará hacia detrás. Si se repite, cuando vuelve a empezar parte de la situación inicial.
- **alternate:** la animación se ejecutará una vez en un sentido y otra vez en otro, comenzando hacia delante, luego hacia detrás y así sucesivamente el número de repeticiones especificado.
- **alternate-reverse:** la animación se ejecutará una vez en un sentido y otra vez en otro, comenzando hacia detrás, luego hacia delante y así sucesivamente el número de repeticiones especificado.

Si se desean aplicar varias animaciones, se especificarán sus direcciones separadas por comas, siendo cada valor correspondiente a la animación cuyo nombre figura en el mismo orden en la propiedad animation-name.

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos o no ser reconocida por algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos o si la propiedad será reconocida por un navegador concreto.

**PROPIEDAD ANIMATION-DELAY**

Análoga a la propiedad transition-delay. Indica el tiempo en segundos (s) o milisegundos (ms) que debe retrasarse el inicio de la animación respecto a lo que sería el momento normal para el comienzo. Ejemplo: animation-delay: 3s;

**PROPIEDAD ANIMATION-TIMING-FUNCTION**

La animación habrá sido definida en una regla @keyframes y va a ser aplicada a un elemento seleccionado mediante un selector CSS. El valor por defecto para esta propiedad es **ease** (implica que la animación se ejecutará con un comienzo rápido, luego velocidad constante y final lento.).

Si se desean aplicar varias animaciones, se especificarán sus timing-functions separadas por comas, siendo cada valor correspondiente a la animación cuyo nombre figura en el mismo orden en la propiedad animation-name.

© aprenderaprogramar.com, 2006-2029

| PROPIEDAD CSS animation-timing-function | |
|---|---|
| **Función de la propiedad** | Permite definir la velocidad a la que transcurre la animación respetando la duración establecida. A efectos prácticos, permite que el usuario vea la animación como progresiva a velocidad constante o con cierta aceleración. |
| **Valor por defecto** | ease |
| **Aplicable a** | Todos los elementos. |
| **Valores posibles para esta propiedad** | **ease**: comienzo rápido, luego velocidad constante y final lento. |
| | **ease-in**: al revés que ease. |
| | **ease-out**: similar a ease con final a velocidad constante. |
| | **ease-in-out**: comienzo lento y progresión a velocidad constante. |
| | **linear**: progresión a velocidad constante. |
| | **step-start**: salto de la situación inicial a la final al comienzo. |
| | **step-end**: salto de la situación inicial a la final al final. |
| | **steps** (numeroDePasos, end) [El parámetro numeroDePasos establece el número de pasos intermedios entre la situación inicial y la final, lo que equivale a que la animación se vea como "pequeños saltos".] |
| | **cubic-bezier**(valor0a1, valor0a1, valor0a1, valor0a1) [Esta forma de definición permite especificar la curva temporal de forma matemática para definir formas de cambio personalizadas] |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | .myContainer { animation-timing-function:ease;}<br>.myContainerSP { animation-timing-function:steps(6, end);} |

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

**PROPIEDAD ANIMATION-PLAY-STATE**

Esta propiedad permite detener una animación que se está ejecutando (ponerla en pausa). Su valor por defecto es running y sus dos valores posibles son:

- **running:** la animación está en ejecución
- **paused:** la animación está en pausa

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

© aprenderaprogramar.com, 2006-2029

Ejemplo: .pubMov:hover{ animation-play-state:paused; } con esta regla damos lugar a que la animación se detenga si el usuario pone el puntero del ratón sobre el elemento animado.

**PROPIEDAD ANIMATION-FILL-MODE**

Esta propiedad permite detener definir cómo debe comenzar y cómo debe quedar un elemento que tiene una animación. Su valor por defecto es none (implica que el elemento comenzará y quedará en el estado que tenía antes de que comenzara la animación) y sus valores posibles son:

- **none:** el elemento comenzará y quedará en el estado previo a la animación.
- **forwards:** el elemento quedará en el estado final de la animación.
- **backwards:** el elemento se pondrá en el estado indicado para el comienzo de la animación inmediatamente y esperará en ese estado hasta que se cumpla el tiempo indicado en animation-delay. Una vez se cumpla ese tiempo la animación continuará la ejecución desde ese estado inicial.
- **both:** combinación de las dos opciones anteriores.

Esta propiedad puede no ser reconocida por los navegadores antiguos o requerir del uso de prefijos específicos para algunos navegadores actuales. Consulta en Mozilla Developer Network para conocer si debes aplicar prefijos.

Ejemplo: .pubMov:hover{ animation-fill-mode: forwards; } con esta regla damos lugar a que cuando la animación se detenga quede en el último estado definido (el correspondiente a to ó 100% dentro de la regla @keyframes).

**PROPIEDAD SHORTAND ANIMATION**

Esta propiedad permite definir las propiedades animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction y animation-fill-mode en una sola regla CSS.

Ejemplo: .pubMov{ animation: animacion 5s 2 alternate linear forwards; } equivale al conjunto de reglas:

animation-name:animacion; animation-duration: 5s; animation-iteration-count:2; animation-direction:alternate; animation-timing-function:linear; animation-fill-mode: forwards;

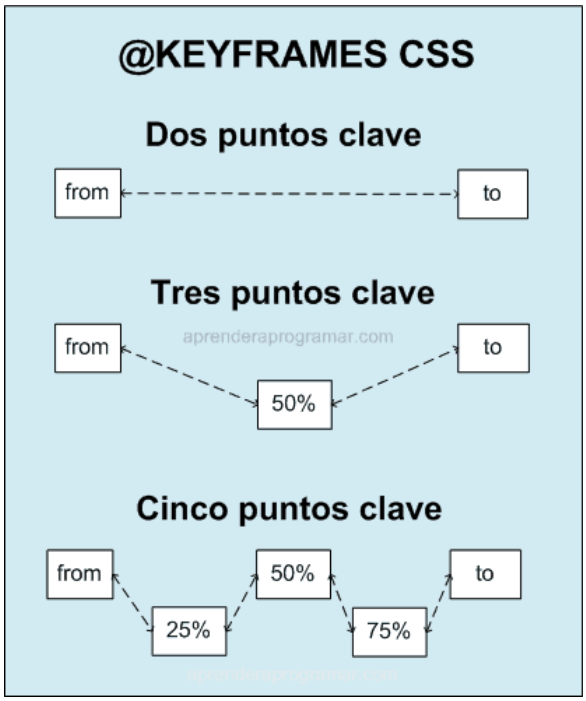
**EJEMPLOS DE APLICACIÓN**

Escribe este código y comprueba los resultados en tu navegador. Recuerda que hay navegadores que no soportan estas propiedades. Consulta en Mozilla Developer Network el soporte que ofrecen los navegadores y sus versiones.

© aprenderaprogramar.com, 2006-2029

```
<html>
<head>
  <title>Portal web - aprenderaprogramar.com</title> <meta charset="utf-8">
  <style type="text/css">

    @keyframes animacion{
      from { margin-left: 10%; color: red;}
                              25% { margin-left:22.5%; margin-top:10%; }
                              50% { margin-left:35%; margin-top:0%;}
                              75% { margin-left:47.5%; margin-top:10%; }
                              to { margin-left: 60%; }
    }

    #bloquePub{ text-align:center; position:relative; width:250px;
                padding: 15px; font-family:arial; font-size:22px;
                font-weight:bold; background-color: yellow; }
    img {padding: 10px;}
    .pubMov{ animation-name:animacion; animation-duration: 5s;
             animation-iteration-count:2; animation-direction:alternate;
             animation-timing-function:linear; animation-fill-mode: forwards;  }

     .pubMov:hover{ animation-play-state:paused; }
  </style>
</head>
<body>
        <div class="pubMov" id="bloquePub" style="border-radius: 40px;">aprenderaprogramar.com
        <img src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" ></a>
        </div>
</body>
</html>
```

El resultado a obtener debe ser similar al que se indica como “Cinco puntos clave” en esta figura:




© aprenderaprogramar.com, 2006-2029

El elemento con un texto y una imagen debe partir de la zona izquierda de la pantalla, ir progresivamente hacia la derecha bajando primero hasta la posición 25%, luego subiendo hasta 50%, luego bajando de nuevo hasta 75% y luego subiendo hasta 100%. A continuación, debe continuar invirtiendo el movimiento, es decir, realizar lo que hizo anteriormente pero al revés hasta volver a la posición inicial.

Prueba a cambiar el número de puntos clave y otros parámetros y observa las diferencias. Si no logras visualizar la animación es probable que se deba a un problema de compatibilidad del navegador o necesidad de uso de prefijos. Consulta en los foros aprenderaprogramar.com si no logras resolverlo.

**EJERCICIO**

Busca en internet (página web o blog) una animación CSS y ejecútala en tu navegador. Escribe por separado el código HTML y el código CSS. Describe paso a paso qué es lo que indica cada fragmento de código CSS. ¿Es necesario usar prefijos de navegador para lograr su ejecución en otros navegadores?

Para comprobar si tu código y respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01066D

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203

© aprenderaprogramar.com, 2006-2029

**FINAL DEL CURSO TUTORIAL**

Con esta entrega llegamos al final del curso "Tutorial básico del programador web: CSS desde cero". Esperamos que haya sido un curso útil y ameno para todas las personas que lo hayan seguido. Y como en todo final, cabe hacer algunas consideraciones especiales.

- Gracias al equipo humano de aprenderaprogramar.com que ha hecho posible su publicación, y en especial a Javier Roa, Jorge Maestro, Manuel Tello, Walter Sagástegui, Enrique González y Mario Rodríguez.

- Gracias a todas las personas que de una u otra forma han participado enviando propuestas de mejora, comentarios, avisos de erratas, etc. y a los alumnos que han seguido el curso colaborando a través de los foros.

- A todos los que no han participado pero han seguido el curso de forma gratuita a través de la web, desde aprenderaprogramar.com les agradeceríamos nos hicieran llegar una opinión o propuesta de mejora sobre el mismo, bien a través de correo electrónico a contacto@aprenderaprogramar.com , bien a través de los foros. Todas las opiniones son bienvenidas y nos sirven para mejorar.

- A quienes hayan seguido el curso de forma gratuita y piensen que los contenidos son de calidad y que merece dar un pequeño apoyo económico para que se puedan seguir ofreciendo más y mejores contenidos en este sitio web, les estaremos muy agradecidos si realizan una pequeña aportación económica en forma de donación pulsando sobre el enlace que aparece en la página principal de aprenderaprogramar.com.

Quienes tengan interés en proseguir formándose en el área de programación y programación web les animamos a que visiten la sección "Cursos de aprenderaprogramar.com" en la URL http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=57&Itemid=86

Quienes estén interesados en cursos tutorizados on-line pueden visitar http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=64&Itemid=87

A todos los que nos han leído y nos siguen, gracias. ¡Nos vemos en el próximo!

**El equipo de aprenderaprogramar.com**

© aprenderaprogramar.com, 2006-2029