MF0950_2-Contrucción de páginas web

UF1303: Elaboración de hojas de estilo

Contenido

Selectores básicos CSS

Selector universal

Se utiliza para seleccionar todos los elementos de la página. El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML (por ahora no es importante fijarse en la parte de la declaración de la regla CSS):

```
* {
margin: 0;
padding: 0;
}
```

Esta regla afecta a todos los elementos de la página, suele ser utilizado para determinar un tipo y tamaño de letra inicial, color, etc. También se suele usado para configurar los márgenes y paddings por defecto.

Selector de tipo o etiqueta

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {
…
}
```

Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres < y >) correspondiente a los elementos que se quieren seleccionar. El siguiente ejemplo aplica diferentes estilos a los titulares y a los párrafos de una página

HTML:

```
h1 {
color: red;
}
h2 {
```

```
color: blue;
}
p {
color: black;
}
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se pueden encadenar los selectores. En el siguiente ejemplo, los títulos de sección `h1`, `h2` y `h3` comparten los mismos estilos:

```
h1 {
color: #8A8E27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}
h2 {
color: #8A8E27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}

h3 {
color: #8A8E27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}
```

En este caso, CSS permite agrupar todas las reglas individuales en una sola regla con un selector múltiple. Para ello, se incluyen todos los selectores separados por una coma (,) y el resultado es que la siguiente regla CSS es equivalente a las tres reglas anteriores:

```
h1, h2, h3 {
color: #8A8E27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}
```

En las hojas de estilo complejas, es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas de esos mismos elementos. El siguiente ejemplo establece en primer lugar las propiedades comunes de los títulos de sección (color y tipo de letra) y a continuación, establece el tamaño de letra de cada uno de ellos:

```
h1, h2, h3 {
color: #8A8E27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}
h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h3 { font-size: 1.2em; }
```

Selector descendente

Permite seleccionar los elementos que son descendientes de otros elementos. Un elemento es descendiente de otro cuando su etiqueta se encuentra dentro de la etiqueta del otro.El selector del siguiente ejemplo selecciona todos los elementos de la página que se encuentren dentro de un elemento <p>:

```
p span { color: red; }
```

Si el código HTML de la página es el siguiente:

```
<p>
...
<span>texto1</span>
...
<a href="">...<span>texto2</span></a>
...
</p>
```

El selector `p span` selecciona tanto `texto1` como `texto2`. El motivo es que en el selector descendente, un elemento no tiene que ser *"hijo directo"* de otro. La única condición es que un elemento debe estar dentro de otro elemento, sin importar lo profundo que se encuentre.

Al resto de elementos de la página que no están dentro de un elemento <p>, no se les aplica la regla CSS anterior.

Los selectores descendentes permiten mejorar la precisión del selector de tipo o etiqueta. Así, utilizando el selector descendente es posible aplicar diferentes estilos a los elementos del mismo tipo. El siguiente ejemplo amplía el anterior y muestra de color azul todo el texto de los contenidos dentro de un <h1>:

```
p span { color: red; }
h1 span { color: blue; }
```

Con las reglas CSS anteriores:

- Los elementos que se encuentran dentro de un elemento <p> se muestran de color azul.
- Los elementos que se encuentran dentro de un elemento <h1> se muestran de color rojo.
- El resto de elementos de la página, se muestran con el color por defecto aplicado por el navegador.

La sintaxis formal del selector descendente se muestra a continuación:

```
elemento1 elemento2 elemento3 ... elementoN
```

Los selectores descendentes siempre están formados por dos o más partes separadas entre sí por espacios en blanco. La última parte indica el elemento sobre el que se aplican los estilos y todas las partes anteriores indican el lugar en el que se tiene que encontrar ese elemento para que los estilos se apliquen realmente.

En el siguiente ejemplo, el selector descendente se compone de cuatro partes:

```
p a span em { text-decoration: underline; }
```

Los estilos de la regla anterior se aplican a los elementos de tipo que se encuentren dentro de elementos de tipo , que a su vez se encuentren dentro de elementos de tipo <a> que se encuentren dentro de elementos de tipo <p>.

No debe confundirse el selector descendente con la combinación de selectores:

```
/* El estilo se aplica a todos los elementos "p", "a", "span" y "em" */
p, a, span, em { text-decoration: underline; }
/* El estilo se aplica solo a los elementos "em" que se encuentran dentro de "p a span" */
p a span em { text-decoration: underline; }
```

Si se emplea el selector descendente combinado con el selector universal, se puede restringir el alcance de un selector descendente. El siguiente ejemplo, muestra los dos enlaces de color rojo:

```
p a { color: red; }
<p><a href="#">Enlace</a></p>
<p><span><a href="#">Enlace</a></span></p>
```

Sin embargo, en el siguiente ejemplo solamente el segundo enlace se muestra de color rojo:

```
p* a { color: red; }
<p><a href="#">Enlace</a></p>
<p><span><a href="#">Enlace</a></span></p>
```

La razón es que el selector `p * a` se traduce como todos los elementos de tipo `<a>` que se encuentren dentro de cualquier elemento que, a su vez, se encuentre dentro de un elemento de tipo `<p>`. Como el primer elemento `<a>` se encuentra directamente bajo un elemento `<p>`, no se cumple la condición del selector `p * a`.

Selector de clase

Si se considera el siguiente código HTML de ejemplo:

```
<body>
<p>Lorem ipsum dolor sit amet...</p>
<p>Nunc sed lacus et est adipiscing accumsan...</p>
<p>Class aptent taciti sociosqu ad litora...</p>
</body>
```

¿Cómo se pueden aplicar estilos CSS sólo al primer párrafo? El selector universal (*) no se puede utilizar porque selecciona todos los elementos de la página. El selector de tipo o etiqueta (p) tampoco se puede utilizar porque seleccionaría todos los párrafos. Por último, el selector descendente (body p) tampoco se puede utilizar porque todos los párrafos se encuentran en el mismo sitio.

Una de las soluciones más sencillas consiste en indicar directamente en el elemento la regla CSS que se le va a aplicar. El atributo class de HTML permite indicar el nombre de la regla CSS que se aplica sobre el elemento:

```
<body>
<p class="destacado">Lorem ipsum dolor sit amet...</p>
<p>Nunc sed lacus et est adipiscing accumsan...</p>
<p>Class aptent taciti sociosqu ad litora...</p>
</body>
```

A continuación, se crea en el archivo CSS una nueva regla llamada destacado con todos los estilos que se van a aplicar al elemento. Para que el navegador no confunda este selector con los otros tipos de selectores, se prefija el valor del atributo class con un punto (.) tal y como muestra el siguiente ejemplo: `.destacado { color: red; }`

El selector `.destacado` se interpreta como *"cualquier elemento de la página cuyo atributo class sea igual a destacado"*, por lo que solamente el primer párrafo cumple esa condición. Este tipo de selectores se llaman selectores de clase y son los más utilizados junto con los selectores de ID que se verán a continuación. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden incluir el mismo valor en el atributo class:

```
<body>
<p class="destacado">Lorem ipsum dolor sit amet...</p>
<p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a> accumsan...</p>
<p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>
</body>
```

Los selectores de clase son imprescindibles para diseñar páginas web complejas, ya que permiten disponer de una precisión total al seleccionar los elementos. Además, estos selectores permiten reutilizar los mismos estilos para varios elementos diferentes. A continuación se muestra otro ejemplo de selectores de clase:

```
.aviso {
padding: 0.5em;
border: 1px solid #98be10;
background: #f6feda;
}
.error {
color: #930;
font-weight: bold;
}
<span class="error">...</span>
<div class="aviso">...</div>
```

El elemento `<span>` tiene un atributo `class="error"`, por lo que se le aplican las reglas CSS indicadas por el selector `.error`. Por su parte, el elemento `<div>` tiene un atributo `class="aviso"`, por lo que su estilo es el que definen las reglas CSS del selector `.aviso`.

En ocasiones, es necesario restringir el alcance del selector de clase. Si se considera de nuevo el ejemplo anterior:

```
<body>
<p class="destacado">Lorem ipsum dolor sit amet...</p>
<p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a> accumsan...</p>
<p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>
</body>
```

¿Cómo es posible aplicar estilos solamente al párrafo cuyo atributo `class` sea igual a `destacado`? Combinando el selector de tipo

y el selector de clase, se obtiene un selector mucho más específico:

```
p.destacado { color: red }
```

El selector `p.destacado` se interpreta como *"aquellos elementos de tipo <p> que dispongan de un atributo class con valor destacado"*. De la misma forma, el selector `a.destacado` solamente selecciona los enlaces cuyo atributo `class` sea igual a `destacado`.

De lo anterior se deduce que el atributo `.destacado` es equivalente a `*.destacado`, por lo que todos los diseñadores obvian el símbolo `*` al escribir un selector de clase normal.No debe confundirse el selector de clase con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo class="aviso" */
p.aviso { ... }
/* Todos los elementos con atributo class="aviso" que estén dentro
de cualquier elemento de tipo "p" */
p .aviso { ... }
/* Todos los elementos "p" de la página y todos los elementos con
atributo class="aviso" de la página */
p, .aviso { ... }
```

Por último, es posible aplicar los estilos de varias clases CSS sobre un mismo elemento. La sintaxis es similar, pero los diferentes valores del atributo `class` se separan con espacios en blanco. En el siguiente ejemplo:

```
<p class="especial destacado error">Párrafo de texto...</p>
```

Al párrafo anterior se le aplican los estilos definidos en las reglas `.especial`, `.destacado` y `.error`, por lo que en el siguiente ejemplo, el texto del párrafo se vería de color rojo, en negrita y con un tamaño de letra de 15 píxel:

```
.error { color: red; }
.destacado { font-size: 15px; }
.especial { font-weight: bold; }
<p class="especial destacado error">Párrafo de texto...</p>
```

Si un elemento dispone de un atributo `class` con más de un valor, es posible utilizar un selector más avanzado:

```
.error { color: red; }
.error.destacado { color: blue; }
.destacado { font-size: 15px; }
.especial { font-weight: bold; }
<p class="especial destacado error">Párrafo de texto...</p>
```

En el ejemplo anterior, el color de la letra del texto es azul y no rojo. El motivo es que se ha utilizado un selector de clase múltiple `.error.destacado`, que se interpreta como *"aquellos elementos de la página que dispongan de un atributo `class` con al menos los valores `error` y `destacado`"*.

Selectores de ID

En ocasiones, es necesario aplicar estilos CSS a un único elemento de la página. Aunque puede utilizarse un selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso.

El selector de ID permite seleccionar un elemento de la página a través del valor de su atributo `id`. Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo `id` no se puede repetir en dos elementos diferentes de una misma página.

La sintaxis que utilizan los selectores de ID es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de la almohadilla (#) en vez del punto (.) como prefijo del nombre de la regla CSS:

```
#destacado { color: red; }
<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

En el ejemplo anterior, el selector `#destacado` solamente selecciona el segundo párrafo (cuyo atributo `id` es igual a `destacado`). La principal diferencia entre este tipo de selector y el selector de clase tiene que ver con HTML y no con CSS. Como se sabe, en una misma página, el valor del atributo `id` debe ser único, de forma que dos elementos diferentes no pueden tener el mismo valor de `id`. Sin embargo, el atributo `class` no es obligatorio que sea único, de forma que muchos elementos HTML diferentes pueden compartir el mismo valor para su atributo `class`.

De esta forma, la recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página y utilizar el selector de clase cuando se quiere aplicar un estulo a varios elementos diferentes de la página HTML. Al igual que los selectores de clase, en este caso también se puede restringir el alcance del selector mediante la combinación con otros selectores. El siguiente ejemplo aplica la regla CSS solamente al elemento de tipo `<p>` que tenga un atributo `id` igual al indicado:

```
p#aviso { color: blue; }
```

A primera vista, restringir el alcance de un selector de ID puede parecer absurdo. En realidad, un selector de tipo `p#aviso` sólo tiene sentido cuando el archivo CSS se aplica sobre muchas páginas HTML diferentes.

En este caso, algunas páginas pueden disponer de elementos con un atributo `id` igual a `aviso` y que no sean párrafos, por lo que la regla anterior no se aplica sobre esos elementos. No debe confundirse el selector de ID con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo id="aviso" */
p#aviso { ... }
/* Todos los elementos con atributo id="aviso" que estén dentro
de cualquier elemento de tipo "p" */
p #aviso { ... }
/* Todos los elementos "p" de la página y todos los elementos con
atributo id="aviso" de la página */
p, #aviso { ... }
```

Ejercicio 1

A partir del código HTML y CSS que se muestra, añadir los selectores CSS que faltan para aplicar los estilos deseados. Cada regla CSS incluye un comentario en el que se explica los elementos a los que debe aplicarse:

```
<!DOCTYPE html">
<head>

<title>Ejercicio de selectores</title>
<style>
/* Todos los elementos de la pagina */
{ font: 1em/1.3 Arial, Helvetica, sans-serif; }

/* Todos los parrafos de la pagina */
{ color: #555; }

/*Todos los párrafos contenidos en #primero */
{ color: #336699; }

/* Todos los enlaces la pagina */
{ color: #CC3300; }

/* Los elementos "em" contenidos en #primero */
{ background: #FFFFCC; padding: .1em; }

/* Todos los elementos "em" de clase "especial" en toda la pagina */
{ background: #FFCC99; border: 1px solid #FF9900; padding: .1em; }

/* Elementos "span" contenidos en .normal */
{ font-weight: bold; }
</style>
```

```
</head>


<body>

<div id="primero">

<p>Lorem ipsum dolor sit amet, <a href="#">consectetuer adipiscing elit</a>. Praesent blandit nibh at felis. Sed nec diam in dolor vestibulum aliquet. Duis ullamcorper, nisi non facilisis molestie, <em>lorem sem aliquam nulla</em>, id lacinia velit mi vestibulum enim.</p>

</div>

<div class="normal">

<p>Phasellus eu velit sed lorem sodales egestas. Ut feugiat. <span><a href="#">Donec porttitor</a>, magna eu varius luctus,</span> metus massa tristique massa, in imperdiet est velit vel magna. Phasellus erat. Duis risus. <a href="#">Maecenas dictum</a>, nibh vitae pellentesque auctor, tellus velit consectetuer tellus, tempor pretium felistellus at metus.</p>

<p>Cum sociis natoque <em class="especial">penatibus et magnis</em> dis parturient montes, nascetur ridiculus mus. Proin aliquam convallis ante. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc aliquet.Sed eu metus. Duis justo.</p>

<p>Donec facilisis blandit velit. Vestibulum nisi. Proin volutpat, <em class="especial">enim id iaculis congue</em>, orci justo ultrices tortor, <a href="#">quis lacinia eros libero in eros</a>. Sed malesuada dui vel quam. Integer at eros.</p>

</div>

</body>

</html>
```

Unidades de medida

Abosolutas

Una medida indicada mediante unidades absolutas está completamente definida, ya que su valor no depende de otro valor de referencia. A continuación se muestra la lista completa de unidades absolutas definidas por CSS y su significado:

- `in`, pulgadas (*"inches"*, en inglés). Una pulgada equivale a `2.54` centímetros.
- `cm`, centímetros.
- `mm`, milímetros.
- `pt`, puntos. Un punto equivale a `1 pulgada/72`, es decir, unos `0.35` milímetros.
- `pc`, picas. Una pica equivale a `12` puntos, es decir, unos `4.23` milímetros.

La principal ventaja de las unidades absolutas es que su valor es directamente el valor que se debe utilizar, sin necesidad de realizar cálculos intermedios. Su principal desventaja es que son muy poco flexibles y no se adaptan fácilmente a los diferentes medios.

De todas las unidades absolutas, la única que suele utilizarse es el punto (`pt`). Se trata de la unidad de medida preferida para establecer el tamaño del texto en los documentos que se van a imprimir, es decir, para el medio `print` de CSS, tal y como se verá más adelante.

Relativas

La unidades relativas, a diferencia de las absolutas, no están completamente definidas, ya que su valor siempre está referenciado respecto a otro valor. A pesar de su aparente dificultad, son las más utilizadas en el diseño web por la flexibilidad con la que se adaptan a los diferentes medios.

A continuación se muestran las tres unidades de medida relativas definidas por CSS y la referencia que toma cada una para determinar su valor real:

- `em`, (no confundir con la etiqueta `<em>` de HTML) relativa respecto del tamaño de letra del elemento.
- `ex`, relativa respecto de la altura de la letra `x` (*"equis minúscula"*) del tipo y tamaño de letra del elemento.

- px, (píxel) relativa respecto de la resolución de la pantalla del dispositivo en el que se visualiza la página HTML.

La unidad em hace referencia al tamaño en puntos de la letra que se está utilizando. Si se utiliza una tipografía de 12 puntos, 1em equivale a 12 puntos. El valor de 1ex se puede aproximar por 0.5 em.

La unidad de medida em siempre hace referencia al tamaño de letra del elemento. Por otra parte, todos los navegadores muestran por defecto el texto de los párrafos con un tamaño de letra de 16 píxel. Por tanto, en este caso el margen de 1em equivale a un margen de anchura 16px.

A continuación se modifica el ejemplo anterior para cambiar el tamaño de letra de los párrafos:

```
p { font-size: 32px; margin: 1em; }
```

El valor del margen sigue siendo el mismo en unidades relativas (1em) pero su valor real ha variado porque el tamaño de letra de los párrafos ha variado. En este caso, el margen tendrá una anchura de 32px, ya que 1em siempre equivale al tamaño de letra del elemento.

Si se quiere reducir la anchura del margen a 16px pero manteniendo el tamaño de letra de los párrafos en 32px, se debe utilizar la siguiente regla CSS:

```
p { font-size: 32px; margin: 0.5em; }
```

El valor 0.5em se interpreta como *"la mitad del tamaño de letra del elemento"*, ya que se debe multiplicar por 0.5 su tamaño de letra (32px x 0.5 = 16px). De la misma forma, si se quiere mostrar un margen de 8px de anchura, se debería utilizar el valor 0.25em, ya que 32px x 0.25 = 8px.

Porcentajes

El porcentaje también es una unidad de medida relativa, aunque por su importancia CSS la trata de forma separada a em, ex y px. Un porcentaje está formado por un valor numérico seguido del símbolo % y siempre está referenciado a otra medida. Cada una de las propiedades de CSS que permiten indicar como valor un porcentaje, define el valor al que hace referencia ese porcentaje.

Los porcentajes se pueden utilizar por ejemplo para establecer el valor del tamaño de letra de los elementos:

```
body { font-size: 1em; }
```

```
h1 { font-size: 200%; }
h2 { font-size: 150%; }
```

Los tamaños establecidos para los elementos `<h1>` y `<h2>` mediante las reglas anteriores, son equivalentes a `2em` y `1.5em` respectivamente, por lo que es más habitual definirlos mediante `em`.

Los porcentajes también se utilizan para establecer la anchura de los elementos:

```
div#contenido { width: 600px; }
div.principal { width: 80%; }

<div id="contenido">
  <div class="principal">
    ...
  </div>
</div>
```

En el ejemplo anterior, la referencia del valor `80%` es la anchura de su elemento padre. Por tanto, el elemento `<div>` cuyo atributo `class` vale `principal` tiene una anchura de `80% x 600px = 480px`

Recomendaciones

En general, se recomienda el uso de unidades relativas siempre que sea posible, ya que mejora la accesibilidad de la página y permite que los documentos se adapten fácilmente a cualquier medio y dispositivo.

Se recomienda el uso de la unidad `em` para indicar el tamaño del texto y para todas las medidas que sean posibles.

Normalmente se utilizan píxel y porcentajes para definir el layout del documento (básicamente, la anchura de las columnas y de los elementos de las páginas) y `em` y porcentajes para el tamaño de letra de los textos.

Diseño de Cajas

La manera en la que hoy en día se plantean los diseños es el modelo de cajas, para ello se usa siempre la etiqueta y mediante los CSS se le aplican una serie de propiedades que ayudan a modelar una página web.

Anchura y altura (widht & height)

Establecen un ancho y alto a la caja.

```
#lateral { width: 200px;
```

```
          Height: 200px }

<div id="lateral">
  ...
</div>
```

Recomendación importante

Todo diseño debe empezar con un div que enmarque todo el contenido, así, independientemente de la resolución que use el usuario, los contenidos no se mueven o desplazan.

Margen y relleno (margin & padding)

El margen es el espacio que se deja entre un elemento y otro y el relleno es el margen interior que se deja desde el borde hacia adentro,

Tanto en el margen como en el relleno se pueden especificar hasta 4 valores (arriba, derecha, abajo, izquierda).

- Si solo se indica un valor, todos los márgenes tienen ese valor.
- Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.
- Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.
- Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.

Ejemplos

```
div img {
  margin-top: .5em;
  margin-bottom: .5em;
  margin-left: 1em;
  margin-right: .5em;
}
```

Alternativa directa:

```
div img {
  margin: .5em .5em .5m 1em;
}
```

Otra alternativa:

```
div img {
  margin: .5em;
  margin-left: 1em;
}
```

Los Bordes

Con los CSS se pueden aplicar un sinfín de atributos para el borde los divs, cada lado se puede configurar con el atributo que le toca (top-right-bottom-left) si no se especifica un lado, lo toma para los 4.

Anchura (border-width)

Border-width:2 px →Los 4 lados del div se ponen a 2 píxeles

Border-top-width: 8px;→Sólo el borde superior se pone a 8píxeles

Color (border-color)

Tipo de borde (border-style)

Con los siguientes valores disponibles:

none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset |

Color de borde (border-color)

Se puede especificar el color de borde para los 4 o cada uno de los lados.

Todo en uno "Shorthand" (border)

Mediante el atributo border se puede especifar el ancho el tipo y el color del borde, para los 4 lados o para cada lado en una sola línea.

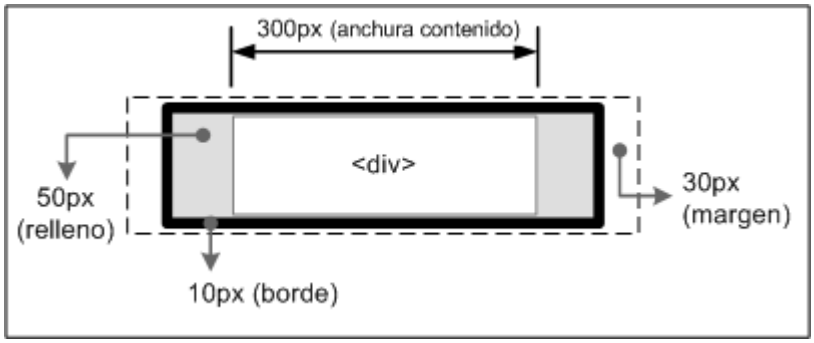
```
border: 3px groove #369;
```

Los tamaños verdaderos de las cajas

La anchura y altura de un elemento no solamente se calculan teniendo en cuenta sus propiedades `width` y `height`. El margen, el relleno y los bordes establecidos a un elemento determinan la anchura y altura final del elemento. En el siguiente ejemplo se muestran los estilos CSS de un elemento:

```
div {
  width: 300px;
  padding-left:  50px;
  padding-right: 50px;
  margin-left:   30px;
  margin-right:  30px;
  border: 10px solid black;
}
```

La anchura total con la que se muestra el elemento no son los `300` píxel indicados en la propiedad `width`, sino que también se añaden todos sus márgenes, rellenos y bordes:



De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:

`30px + 10px + 50px + 300px + 50px + 10px + 30px = 480 píxel`

Así, la anchura/altura establecida con CSS siempre hace referencia a la anchura/altura del contenido. La anchura/altura total del elemento debe tener en cuenta además los valores del resto de partes que componen la caja del *box model*.

Box-sizing

Existe una propiedad adicional incorporada en CSS3 relacionada con la estructura y el Modelo de Caja Tradicional. La propiedad box-sizing nos permite cambiar cómo el espacio total ocupado por un elemento en pantalla será calculado forzando a los navegadores a incluir en el ancho original los valores de las propiedades padding y border.

Por este motivo, si declaramos la propiedad width igual a 100 pixeles, margin en 20 pixeles, padding en 10 pixeles y border en 1 pixel, el área horizontal total ocupada por el elemento será: 100+40+20+2= 162 pixeles (note que tuvimos que duplicar los valores de margin, padding y border en la fórmula porque consideramos que los mismos fueron asignados tanto para el lado derecho como el izquierdo).

Esto significa que cada vez que declare el ancho de un elemento con la propiedad width, deberá recordar que el área real para ubicar el elemento en pantalla será seguramente más grande. Dependiendo de sus costumbres, a veces podría resultar útil forzar al navegador a incluir los valores de padding y border en el tamaño del elemento. En este caso la nueva fórmula sería simplemente: tamaño + márgenes.

```
div { width: 100px; margin: 20px; padding: 10px; border: 1px solid #000000; -moz-box-sizing: border-box; -webkit-box-sizing: border-box; box-sizing: border-box; }
```

La propiedad box-sizing puede tomar dos valores. Por defecto es configurada como content-box, lo que significa que los navegadores agregarán los valores de padding y border al tamaño especificado por width y height. Usando el valor border-box en su lugar, este comportamiento es cambiado de modo que padding y border son incluidos dentro del elemento.

IMPORTANTE:

Muchas de las reglas CSS3 requieren de ser declaradas para ser compatibles en todos los navegadores con variaciones según el navegador como se aprecia en el código anterior.

Border-radius

Por muchos años diseñadores han sufrido intentando lograr el efecto de esquinas redondeadas en las cajas de sus páginas web. El proceso era casi siempre frustrante y extenuante. Todos lo padecieron alguna vez. Si mira cualquier presentación en video de las nuevas características incorporadas en HTML5, cada vez que alguien habla sobre las propiedades de CSS3 que hacen posible generar fácilmente esquinas redondeadas, la audiencia enloquece. Esquinas redondeadas eran esa clase de cosas que nos hacían pensar: "debería ser fácil hacerlo". Sin embargo nunca lo fue.

```
#principal { display: block; width: 500px; margin: 50px auto; padding: 15px; text-align: center; border: 1px solid #999999;
background: #DDDDDD; -moz-border-radius: 20px; -webkit-border-radius: 20px; border-radius: 20px; }
```

Al igual que con margin o padding, border-radius puede también trabajar solo con dos valores. El primer valor será asignado a la primera y tercera equina (superior izquierda, inferior derecha), y el segundo valor a la segunda y cuarta esquina (superior derecha, inferior izquierda). También podemos dar forma a las esquinas declarando un segundo grupo de valores separados por una barra. Los valores a la izquierda de la barra representarán el radio horizontal mientras que los valores a la derecha representan el radio vertical. La combinación de estos valores genera una elipsis:

```
-moz-border-radius: 20px / 10px; -webkit-border-radius: 20px / 10px; border-radius: 20px / 10px;
```

Box-shadow

La propiedad box-shadow necesita al menos tres valores. El primero, que puede ver en la regla del Listado 3-6, es el color. Este valor fue construido aquí utilizando la función rgb() y números decimales, pero podemos escribirlo en números hexadecimales también, como hicimos previamente para otros parámetros en este libro. Los siguientes dos valores, expresados en pixeles, establecen el desplazamiento de la sombra. Este desplazamiento puede ser positivo o negativo. Los valores indican, respectivamente, la distancia horizontal y vertical desde la sombra al elemento. Valores negativos posicionarán la sombra a la izquierda y arriba del elemento, mientras que valores positivos

crearán la sombra a la derecha y debajo del elemento. Valores de 0 o nulos posicionarán la sombra exactamente detrás del elemento, permitiendo la posibilidad de crear un efecto difuminado a todo su alrededor.

```
-moz-box-shadow: rgb(150,150,150) 5px 5px;
-webkit-box-shadow: rgb(150,150,150) 5px 5px;
box-shadow: rgb(150,150,150) 5px 5px;
```

Agregando otro valor más en pixeles al final de la propiedad desparramará la sombra. Este efecto cambia un poco la naturaleza de la sombra expandiendo el área que cubre.

```
box-shadow: rgb(150,150,150) 5px 5px 10px;
```

Text-shadow

Los valores para text-shadow son similares a los usados para box-shadow. Podemos declarar el color de la sombra, la distancia horizontal y vertical de la sombra con respecto al objeto y el radio de difuminación.

```
text-shadow: rgb(0,0,150) 3px 3px 5px;
```

Gradiente lineal

Los gradientes son uno de los efectos más atractivos entre aquellos incorporados en CSS3. Este efecto era prácticamente imposible de implementar usando técnicas anteriores pero ahora es realmente fácil de hacer usando CSS. Una propiedad background con algunos pocos parámetros es suficiente para convertir su documento en una página web con aspecto profesional:

```
background: linear-gradient(30deg, #FFFFFF, #006699);
background: -webkit-linear-gradient(top, #FFFFFF, #006699);
background: -moz-linear-gradient(top, #FFFFFF, #006699);
```

Gradiente radial

La sintaxis estándar para los gradientes radiales solo difiere en unos pocos aspectos con respecto a la anterior. Debemos usar la función radial-gradient() y un nuevo atributo para la forma:

```
background: radial-gradient(center, circle, #FFFFFF 0%, #006699 200%);
```

RGBA

La función rgba() tiene cuatro atributos. Los primeros tres son similares a los usados en rgb() y simplemente declaran los valores para los colores rojo, verde y azul en números decimales del 0 al 255. El último, en cambio, corresponde a la nueva capacidad de opacidad. Este valor se debe encontrar dentro de un rango que va de 0 a 1, con 0 como totalmente transparente y 1 como totalmente opaco.

```
#titulo {  font: bold 36px MiNuevaFuente, verdana, sans-serif;  text-shadow: rgba(0,0,0,0.5) 3px 3px 5px; }
```