2 Uso de estilos

OBJETIVOS DEL CAPÍTULO

- ✓ Identificar las posibilidades de modificar las etiquetas HTML.
- ✓ Ser capaz de definir estilos de forma directa.
- ✓ Ser capaz de definir y asociar estilos globales en hojas externas.
- ✓ Ser capaz de definir hojas de estilos alternativas.
- ✓ Identificar las distintas propiedades de cada elemento.
- ✓ Ser capaz de crear clases de estilos.
- ✓ Ser capaz de utilizar herramientas de validación de hojas de estilos.

Las hojas de estilo en cascada o CSS (*Cascading Style Sheet*) son una parte muy importante dentro del diseño de aplicaciones web. CSS es un estándar del Consorcio WWW o W3C, ampliamente reconocido y utilizado debido a dos aspectos importantes que ofrece: ahorrar tiempo en el diseño de sitios web y conseguir efectos potentes, soportados por la mayoría de los navegadores.

Este capítulo se centra en los aspectos básicos de CSS. Se mostrarán los elementos más comunes y se ilustrarán con ejemplos prácticos. Todo ello para que un diseñador conozca este lenguaje y pueda desarrollar sus propias plantillas y/o interpretar plantillas CSS ya existentes. Sin embargo, CSS es un estándar muy extenso, y no es objetivo de este capítulo trabajarlo en su totalidad. Una vez el lector entienda el funcionamiento de CSS, podrá utilizar Internet para consultar el estándar en W3C, sitios web o bibliografía especializada en CSS para conocer aspectos avanzados (trucos y soluciones específicas).

2.1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET)

Entre todos los avances que en la última década se han producido en el campo del diseño web, la aparición de CSS (*Cascading Style Sheet*, Hojas de estilos en cascada) ha sido uno de los más significativos. Su principal ventaja es que permiten separar en el desarrollo de un sitio web lo que es el diseño (apariencia) de lo que son los contenidos (información que se quiere transmitir). Esto tiene como efecto inmediato un desarrollo y mantenimiento más eficiente de sitios web.

Más técnicamente, la filosofía de las CSS se puede resumir así: usar la etiqueta <body> de HTML para definir las estructuras de los contenidos que se muestran en el sitio (encabezados, párrafos, viñetas, etc.) y, luego, en otro archivo o en el <head> del HTML, se define la apariencia de cada página usando el lenguaje de CSS. Esto permite que se puedan cambiar los contenidos que se pongan en el <body> sin afectar a la apariencia definida en el archivo CSS (o en el <head> del HTML), o que se cambie la apariencia en el CSS sin que afecte a nada de los puesto en el <body> del HTML.

Otra de las grandes ventajas actuales de los CSS es la adaptación de los sitios web a los dispositivos con los que será visualizado. Un sitio web puede ser visto desde un iPad, un móvil con Android, un ordenador personal o cualquiera de los dispositivos disponibles actualmente. La combinación de CSS y HTML permite crear sitios web personalizados para cada dispositivo. Cuando el servidor detecta el dispositivo cliente (*Media Queries*), éste aplica una hoja de estilos para un mejor visionado. Evidentemente, cada una de las hojas de estilos para cada dispositivo debe haber sido diseñada a propósito por el diseñador, aquí no hay magias.

La organización W3C fue la encargada de estandarizar la versión CSS1. W3C definió CSS como hojas de estilo en cascada ya que es posible que un mismo contenido pueda ser regido por varios archivos CSS que controlen su apariencia. Lo que el estándar CSS1 de W3C pretendía era establecer los criterios por los que se da preferencia a un estilo respecto a otro (por eso lo de cascada). Un mismo elemento, como el encabezamiento <h1> de la página puede estar definido en un archivo CSS externo para aparecer como texto azul; en la propia página, definirlo como texto rojo. En este caso (muy común) el navegador tiene que optar por uno u otro estilo. Por lo tanto, la especificación W3C marca las pautas de preferencia que debe respetar un navegador compatible con CSS.

CSS1 alcanzó el status de recomendación por la W3C en el año 1996. Las reglas CSS1 tienen un soporte adecuado en prácticamente todos los navegadores modernos más conocidos. Las reglas CSS 2 alcanzaron el status de recomendación en el año 1998. En junio de 2011, el modulo de colore de CSS 3 Color ha sido publicado.

El objetivo de que W3C estandarice las CSS, HTML o cualquier otro lenguaje de Internet, es garantizar que el creador de un sitio web no tiene que hacer uno a propósito (ad hoc) para cada uno de los navegador web existente (IExplorer, Firefox, Chrome, Opera, etc.) y, además, que los usuarios no vean un sitio web con apariencia diferente dependiendo del navegador que empleen.

En las siguientes secciones se muestras los detalles de CSS lo que permitirá, que al final del capítulo, el lector sepa generar CSS para un sitio web y entender CSS de sitios web ya creados o creadas éstas con herramientas que automatizan el proceso.

2.2 SELECTORES: ESTILOS EN LÍNEA BASADOS EN ETIQUETAS, EN CLASES Y EN IDENTIFICADORES

Una hoja de estilo CSS está formada por *reglas* que indican la manera en la que se visualizará la página (reglas de estilo). Cada regla está compuesta de *selectores* lo cuales indican a qué elemento o parte de una página se aplica un determinado estilo.

2.2.1 SELECTORES BASADOS EN ETIQUETAS

Para dar los primeros pasos en la definición de reglas de estilo, la manera más sencilla es usar las propias etiquetas HTML como selectores. Esto consiste en asociar a cada etiqueta una *declaración* del estilo que se le aplica al selector (etiqueta HTML). Las declaraciones tienen esta forma:

```
selector { atributo:valor }
```

El *atributo* hace referencia a la característica que se quiere modificar de la etiqueta, por ejemplo color. El valor hace referencia a la instancia del atributo, por ejemplo, *blue*.

Así, por ejemplo, *h1* podría ser un selector para aplicar un estilo a la etiqueta <h1>. Para indicar entonces el estilo de <h1> se pondría:

```
h1 {color:blue}
```

Los selectores se escriben omitiendo las llaves < >, es decir, simplemente h1, h2, etc. La declaración {*atributo*:*valor*} ha de ir encerrada en llaves{ }.

A cualquier etiqueta HTML se le puede asignar un estilo pero la descripción de las reglas debe ajustarse a la sintaxis definida anteriormente (según la especificación CSS). Si un navegador encuentra un selector cuya sintaxis no comprende, éste ignorará la declaración entera y continúa con la siguiente, con independencia de si el error afecta a la propiedad, al valor o a toda la descripción.

CSS contempla sintaxis para definir atributos para varios selectores y selectores con varios atributos.

Selector1, Selector2, {atributo1:valor1; atributo2:valor2}

Así, por ejemplo, si el mismo estilo se le quiere aplicar a dos selectores distintos, la sintaxis sería:

```
h1 , h2 {color:blue}
```

y si al mismo selector se le quiere aplicar atributos diferentes:

```
h1 {color: blue; background-color:red }
```

La propiedad *blackground-color* hace referencia al color de fondo del texto que por defecto es como el de la página.

Una vez conocida la sintaxis de las reglas, la siguiente pregunta sería ¿dónde se debe poner esas declaraciones para que el navegador las lea y las interprete? Una respuesta válida (en la sección 2.8 se muestran más) es en la cabecera <head> </head>. Todas las declaraciones basadas en etiquetas se incluyen en el <head> del fichero HTML entre etiquetas <style></style> (trataremos estas etiquetas y su utilización en las Secciones 2.8 y 2.9 de este mismo capítulo). Así, por ejemplo, las declaraciones del ejemplo anterior quedarían así:

```
<head>
<title></title>
<style>
  h1 {color: blue; background-color:red }
</style>
</head>
```

Cuando en navegador lee la cabecera del HTML interpreta que todas las etiquetas <h1> incluidas en el <body> tendrán color azul y color de fondo rojo.

ACTIVIDADES 2.1

- Cree un fichero HTML con dos textos, uno entre etiquetas <h1></h1> y otro entre <h2></h2>. Luego incluya en el <head> un estilo que afecte al color de las etiquetas h1 (color) y al color de fondo (background-color) de los h2. De tal manera que quede como la Figura 2.1 (el texto entre <h1> es azul y el <h2> en negro con el *background* rojo):

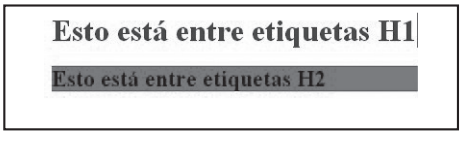

Figura 2.1. *Mi primer CSS*

2.2.2 SELECTORES BASADOS EN CLASES

Los selectores basados en etiquetas tienen un uso muy claro, lo que las hace fáciles de entender. Sin embargo, desde el punto de vista de la flexibilidad y la reutilización, esta alternativa no es muy ventajosa.

Un ejemplo que muestra esa falta de flexibilidad es si se desea establecer diferentes estilos según el tipo de párrafo que se ponga, de manera que se distinga el encabezado de la página del resto del texto. Con lo visto en la sección anterior, si se hace una declaración del tipo *p {color:blue}* siempre que aparezca un texto entre etiquetas éste se mostrará en color azul. Pero, si lo que se desea es que unos párrafos respondan a una regla de estilo y otros a otra esta alternativa no es válida. Es necesario usar *selectores basados en clases*.

Mediante las clases se pueden definir estilos abstractos, es decir, que no estén asociados directamente a una etiqueta HTML. Las clases permiten aplicar estilos a etiquetas HTML, con el mismo efecto que usando selectores de etiquetas, pero también a cualquier otro elemento de la página.

Hay diferentes tipos de clases: unas asociadas directamente a una etiqueta HTML (por ejemplo h1.verde {color:green}) y otras más genéricas que se pueden aplicar a cualquier etiqueta (por ejemplo, .citas {color:grey}).

Las clases asociadas a etiquetas HTML se definen con el

Nombreetiqueta.nombreclase {atributo:valor; atributo:valor; … }

Esta alternativa es más potente que la vista con selectores basados en etiquetas, ya que permite aplicar a las mismas etiquetas diferentes estilo.

```
<head>
<style>
h1.roja {color: red}
h1.verde {color: green}
h1.azul {color: blue}
</style>
</head>
```

Para indicar en cada etiqueta <h1> qué estilo se le quiere aplicar se usa el atributo *class* de la siguiente manera:

```
<body>
<h1 class=”roja”> Un encabezamiento rojo </h1>
<h1 class=”azul”> ahora azul </h1>
<h1 class=”verde”> Y ahora verde </h1>
</body>
```

El resultado quedaría como en la Figura 2.2: textos de tamaño grande pero de diferente color (el cambio de color se apreciará al ejecutar el código en el navegador).

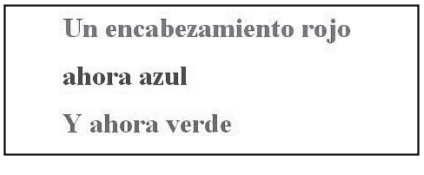

Figura 2.2. *Ejemplo clases basadas en etiquetas*

Las clases más genéricas no se aplican a ninguna etiqueta HTML, por lo que en su descripción se emite en el selector el nombre de ninguna etiqueta.

.nombreclase {atributo:valor; atributo:valor; … }

Por ejemplo:

```
.verde {color:green;}
```

Una clase así definida puede aplicarse a cualquier elemento de la página. Del ejemplo siguiente, la primera declaración se aplica a todo el párrafo, la segunda a todo el encabezado <h1> y la tercera a todo el bloque :

```
<p class="verde"> …
<h1 class="verde"> …
<div class="verde"> …
```

Con la siguiente declaración:

```
<head>
<style>
.roja {color: red}
.verde {color: green}
.azul {color: blue}
</style>
</head>
```

Y aplicando las clases a las diferentes etiquetas <h1>:

```
<body>
<h1 class="roja">Un encabezamiento rojo</h1>
<h1 class="azul">ahora azul </h1>
<h1 class="verde">Y ahora verde</h1>
</body>
```

se obtienen un resultado idéntico al de la Figura 2.2. Sin embargo, con estas solución, no solo se limita aplicar estas clases a las etiquetas <h1> sino que se pueden aplicara a cualquier otra, por ejemplo a una etiqueta <p>.

```
<p class="roja">Aquí está el párrafo en rojo </p>
```

Si esa línea HTML se incluye en el <body> del ejemplo anterior quedaría como muestra la Figura 2.3 (con la línea nueva insertada):

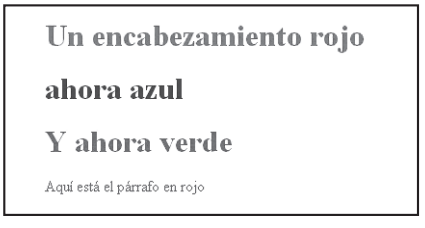

Figura 2.3. *Ejemplo de clases abstractas*

Una característica muy interesante del uso de clases es que se puede asignar más de una clase a una misma etiqueta, siempre y cuando no hay conflicto entre ellas. Por ejemplo, con la siguiente definición:

```
<head>
<style>
.textorojo { color: red }
.fondoazul { background-color: blue }
</style>
</head>
```

Se podría hacer el siguiente contenido en el <body>:

```
<h3 class="textorojo fondoazul">titulo en rojo, fondo azul</h3>
<p class="textorojo">color en rojo; fondo: el que herede de la pagina.</p>
```

El resultado sería el de la Figura 2.4 (los colores usados los explica el propio texto):

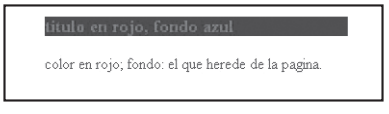

Figura 2.4. *Ejemplo de uso de varias clases en la misma etiqueta*

ACTIVIDADES 2.2

- Cree un fichero HTML con dos textos, uno entre etiquetas <h1></h1> y otro entre . Luego incluya en el <head> estilos con clases abstractas que muestren: el texto en <h1> en verde con fondo azul y el texto de en azul con fondo verde. Resuelva la actividad de dos maneras:
  a. La primera, usando una declaración de estilos con 4 etiquetas (2 para los colores y dos para los fondos).
  b. La segunda usando dos etiquetas: una que se puede llamar *.textoverdefondoazul* y otra que se puede llamar *.textoazulfondoverde*

2.2.3 SELECTORES BASADOS EN IDENTIFICADORES

A modo de resumen de esta sección anterior, las clases definen propiedades que pueden ser compartidas por uno o varios elementos. Esto hace que una clase, por ejemplo .roja vista anteriormente, puede ser usada en un elemento <h1 class="roja"></h1> y en un . Es decir, *las clases se pueden utilizar en varios elementos.*

Concluido esto, en esta sección se explicarán los selectores basados en identificadores, los cuales tienen una función y sintaxis muy parecida, pero que se diferencian en algo muy sutil: *los identificadores solo se pueden usar en un único elemento.*

Un selector basado en identificador se define dentro de las etiquetas <style></style> con la siguiente sintaxis:

Nombreetiqueta#nombreclase {atributo:valor; atributo:valor; … }
#nombreclase {atributo:valor; atributo:valor; … }

Como puede apreciarse, esta declaración es idéntica a la usada para definir selectores de clase. La única diferencia es que en vez de usar un punto "." se usa una almohadilla "#". Sin embargo, el significado, la semántica de ambas expresiones es muy parecida.

Luego, para indicar en cada etiqueta <h1> qué estilo se le quiere aplicar se usa el atributo *id* en la etiqueta (de la misma manera que se usa *class* para las clases). Así, el siguiente ejemplo muestra una definición de estilos usando selectores de identificador.

```
<head>
<style>
#rojo { color:red; }
</style>
</head>

<body>
<p id="rojo"> el párrafo va en rojo </p>
</body>
```

Este código el navegador lo interpreta mostrando en color rojo "el párrafo va en rojo".

En el ejemplo anterior, si cambiamos *id* por *class* y "#" por "." en vez de aplicar selectores de identificador aplicamos selectores de clase. Aparentemente no hay diferencia entre identificadores y clases. Sin embargo, sí que las hay, muy sutiles, pero importantes. La diferencia entre identificadores y clases es la misma que entre clases y objetos en un modelo orientado a objetos. A grandes rasgos las clases definen un patrón que deben cumplir todos los objetos de la clase. Cada objeto se identifica con un identificador único que lo diferencia de los demás objetos de esa clase.

En CSS las clases se pueden usar en uno o varios elementos. Por ejemplo, en una etiqueta <h1> y en una <h2> y en otra <h1> diferente y en una <p>. Sin embargo, y esta es la diferencia, los identificador solo se deben usar en un único elemento. Esto es porque un identificador es como si hiciese referencia a un objeto de estilo y los objetos son únicos, por tanto solo un elemento puede "coger" ese objeto de estilo.

Un ejemplo que muestra la diferencia de uso es el siguiente: La clases son habituales usarlas de esta manera, que muestra cómo la clase *textorojo* se usa en dos elementos :

```
<div class="textorojo">
    El texto hereda las propiedades de una clase textorojo
</div>

<div class="textorojo">
    Se vuelve a heredar las propiedades de la misma clase textorojo
</div>

<div class="textonegrita">
    El texto hereda las propiedades de las clases textonegrita.
</div>
```

Sin embargo, lo identificadores se suelen usar en casos como el siguiente en donde los identificadores *cabecera*, *contenidos* y *piepagina* definen el estilo de esas tres partes de una página web. Los tres objetos son asociados a tres identificadores diferentes ya que no tiene sentido que esos identificadores se repitan en varios elementos (no tiene sentido que una página tenga por ejemplo dos o más elementos con un estilo tipo *cabecera* en una misma página web).

```
<div id=”cabecera”></div>
<div id=”contenido”></div>
<div id”=piepagina”></div>
```

En definitiva, las clases se usan cuando el estilo se quiere aplicar a más de un elemento, mientras que los identificadores se definen cuando lo que se busca es “exclusividad” ya que solo será aplicada a un elemento. Una práctica habitual de los identificadores es usarlos para nombrar los bloques o secciones principales de un sitio web. El resto de estilos se hacen con clases.

Realmente, en muchos de los navegadores actuales, si se usa un identificador en varios elementos estos se interpretan y devuelven un resultado idéntico al que se devuelve si se hiciese con clases. Sin embargo, son “*buenas prácticas*” *de diseñador usar los identificadores y las clases es su momento*, con vistas a seguir un estándar de uso y también para que el diseñador se pueda beneficiar de las ventajas de los identificadores. Por ejemplo, CSS permite dar como valor un identificador al atributo *href* de la etiqueta <a> de la siguiente manera:

```
<style>
#cabecera
{   background:#CCC;
    border:1px solid #093;
    margin:10px 12px 20px 15px;
 }
</style>
</head>

<body>

<div id=”cabecera”> Aquí está la cabecera </div>
…
…
<a href=”#cabecera”> Ir a la cabecera </a>
</body>
```

En este otro ejemplo la etiqueta <a> define un enlace al elemento que usa el identificador “cabecera” que es un . Obviamente, esto se puede hacer por la características deben tener los identificadores de usarse en un único elemento. Si se usan clases o se usa un identificador en varios elementos (no recomendable) el navegador no sabría a qué elemento definido con estilo “cabecera” saltar. Esa es solo una de las muchas ventajas que tiene usar identificadores según su se recomienda en la especificación CSS.

ACTIVIDADES 2.3

- Cree un fichero HTML que defina un selector de clase y un selector de identificador. Pruebe a usar ID y CLASS en uno y varios elementos HTML y compruebe su efecto. ¿Qué pasa si no se cumple la especificación y se repite un identificador en varios elementos? ¿Cómo responde el navegador? Y en otro navegador, ¿responde igual?

2.3 AGRUPACIÓN Y ANIDAMIENTO DE SELECTORES

Los selectores se pueden agrupar y anidar para conseguir estilos CSS, por un lado más concretos y definidos, y por otro lado para tener un fichero CSS más optimizado y fácil de entender por el equipo de desarrollo.

2.3.1 AGRUPAMIENTOS

El término agrupamiento hace referencia a la manera en la que se pueden escribir las reglas de estilo (selectores) para conseguir un CSS más claro y fácil de entender. De esta manera, los errores son más fáciles de detectar y corregir. El uso de agrupamientos, como ocurría con el uso de los *id* y *class* en la sección anterior, responden a buenas prácticas en la especificación de CSS. Como se verá, algunas de las reglas de agrupamiento han sido ya comentadas en la sección 2.2.

Cualquier selector se puede agrupar siguiendo esta sintaxis:

Selector1, Selector2, {atributo1:valor1; atributo2:valor2;…}

De esta manera se puede aplicar el mismo estilo a un conjunto de selectores al mismo tiempo. Por ejemplo, si se quiere aplicar un tipo de fuente *Arial* a etiquetas <h1>, <p> y <h3>, la regla sería:

```
h1,p,h3 {Font-family:arial}
```

La versión menos optimizada de esa misma regla sería:

```
h1 {font-family: arial;}
p {font-family: arial;}
h3 {font-family: arial;}
```

De la misma manera se podría hacer para selectores de clase, pero teniendo en cuenta que si son abstractos es necesario primero poner un ".". El siguiente ejemplo le asigna un color en hexadecimal (#0000FF) a tres clases: *mensaje, énfasis* y *subtítulo*.

```
.mensaje, .subtitulo, .enfasis {color:#0000FF}
```

Lo mismo se puede hacer para selectores de identificador, pero recordando que se usa la almohadilla. El siguiente ejemplo asigna a los identificadores un color verde (#00FF00):

```
#cabecera, #piepagina {color:#00FF00}
```

Como se ha comentado antes, las agrupaciones tienen como objetivo simplificar y dejar más claro un CSS. Por lo tanto, también se pueden combinar diferentes tipos de selectores para agrupar según estilos. Así, por ejemplo, la clase *.cabecera*, la etiqueta *h2* y el identificador *#micolor* comparten el mismo color en hexadecimal (#FF0000):

```
.cabecera, h2, #micolor {color:#FF0000}
```

2.3.2 ANIDAMIENTOS

Los selectores se pueden anidar con el fin de conseguir estilos más concretos y definidos. Este anidamiento es lo que se llama en CSS *selectores contextuales* que permiten aplicar un estilo a un elemento dependiendo de los elementos que tenga alrededor.

Selector anidado común

Se usa para crear reglas sobre elementos que están rodeados de otros elementos. La sintaxis general de este tipo de anidamiento para dos selectores es la siguiente:

SelectorX SelectorY {atributo1:valor1; atributo2:valor2;…}

Observar que entre ambos selectores hay un espacio en blanco.

Este tipo de anidamiento es útil cuando, por ejemplo, se desea ver en rojo un texto en negrita siempre y cuando esté incluido dentro de una etiqueta <h1> y en cursiva <i> (aunque entre medias haya otras etiquetas).

Si se definen los selectores de esta manera:

```
b {color:red}
```

El siguiente código HTML visualizará ambos textos en rojo, con independencia de la etiqueta <h1> y <i>:

```
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y rojo </b></i> </h1>
<b> Un texto en negrita y rojo </b>
</body>
```

Sin embargo, usando un anidamiento en la definición de la regla se consigue el efecto deseado:

```
<head>
<style>
h1 i b {color:red}
</style>
</head>
```

Las Figuras 2.5 y 2.6 muestran las salidas del HTML con los estilos anteriores (en la Figura 2.6 el anidamiento hace que "Un texto en negrita y rojo" aparezca en negro en el navegador).

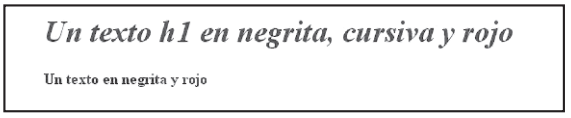

Figura 2.5. *Ejemplo sin anidamiento*

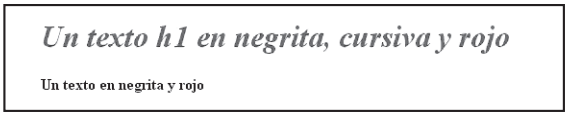

Figura 2.6. *Ejemplo con anidamiento*

En principio no hay límite en el número de anidamiento que se pueden hacer, sin embargo no es recomendable un anidamiento de más de 4 ó 5 por motivos de complejidad a la hora de interpretar el CSS, tanto por el navegador como por los diseñadores. Por otro lado, el anidamiento en los ejemplo de arriba se ha hecho con selectores basados en etiqueta, pero también pueden usarse con *class*, *id*, o combinaciones

Enlazando con los agrupamientos vistos anteriormente, este tipo de selectores se pueden agrupar. Por ejemplo, si suponemos que hemos definidos varias reglas con anidamiento h1 i b y h1 b, el agrupamiento sería así:

```
h1 i b, h1 b { color: red; },
```

que equivaldría a esto:

```
h1 i b {color:red}
h1 b {color:red}
```

Anidamiento de selectores hijos

El anidamiento común explicado anteriormente se comporta igual si las etiquetas están consecutivas o si hay etiquetas intermedias. Por ejemplo, el siguiente anidamiento mostraría los dos textos en verde.

```
<head>
<style>
h1 b {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y rojo </b></i> </h1>
<h1><b> Un texto en negrita y rojo </b></h1>
</body>
```

El motivo es que para el navegador, tanto <h1><i><b> como <h1><b> cumplen el anidamiento definido h1 b {color:red}.

Si lo que se desea es restringir que las etiquetas, además de estar en el mismo contexto, estén seguidas unas de otras, entonces la sintaxis que se tiene que usar en la definición es la siguiente (para anidamiento de dos selectores):

SelectorX > SelectorY {atributo1:valor1; atributo2:valor2;…}

De esta manera, el siguiente código mostrará en rojo solo el texto que tiene una etiqueta <b> dentro de <h1> sin ninguna entre medias tal y como muestra la Figura 2.7 (solo el segundo texto aparecerá en rojo al probarlo en un navegador).

```
<head>
<style>
h1>b {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y negro</b></i> </h1>
<h1><b> Un texto en h1,negrita y rojo </b></h1>
</body>
```

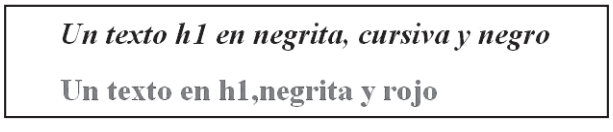

Figura 2.7. *Anidamiento de un hijo*

El siguiente código muestra un ejemplo que incluye a tres etiquetas, tal y como muestra la Figura 2.8 (solo el segundo texto aparecerá en rojo al probarlo en un navegador):

```
<head>
<style>
h1>b>i {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y negro</b></i> </h1>
<h1><b><i><u> Texto en h1,negrita, cursiva, rojo y subrayado </u></i></b></h1>
</body>
```

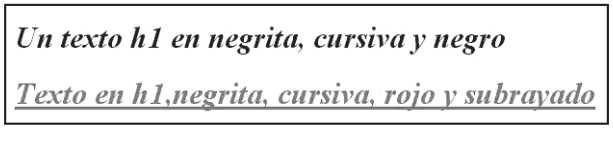

Figura 2.8. *Anidamiento de dos hijos*

Al igual que el resto de anidamientos, aunque los ejemplo se ha hecho con selectores basados en etiqueta también pueden usarse con *class*, *id*, o combinaciones.

Anidamiento de selectores adyacentes

Este tipo de anidamiento se usa cuando se quiere aplicar un estilo a un elemento que tiene adyacente (al lado) a otro elemento en el mismo nivel de anidamiento en HTML. La sintaxis es la siguiente (para dos selectores):

SelectorX + SelectorY {propiedad1:valor1; propiedad2:valor2;…}

El siguiente ejemplo muestra este estilo de anidamiento sobre dos etiquetas <i> y <b>. Solo se aplicará el estilo sobre <b> si hay una etiqueta adyacente <i>. En este ejemplo en concreto, la palabra *advertencia* será la única que aparezca en rojo.

```
<head>
<style>
i+b {color:red}
</style>
</head>
<body>
<p> <i>Nota</i>, esto es una <b>advertencia</b> </p>
<b> Leer detenidamente </b>
</body>
```

El resultado se puede ver en la Figura 2.9 (la palabra “advertencia” aparece en rojo al probarlo en un navegador):

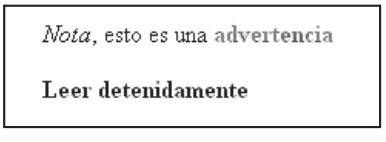

Figura 2.9. *Anidamiento de adyacentes*

Al igual que el resto de anidamientos, aunque los ejemplo se ha hecho con selectores basados en etiqueta también pueden usarse con *class*, *id*, o combinaciones.

ACTIVIDADES 2.4

Interprete qué hace el siguiente código. ¿De qué color aparecerán los elementos de la lista? ¿De qué tamaño? Una vez interpretado compruebe el resultado en el navegador.

```
<head>
<style>
i+b {color:red}
.nivel1 {color:green}
.nivel2 {color:blue}
ul ul .nivel2 { font-size: x-small }
ul ul li {font-size: x-small; color:red}
</style>
</head>

<body>
<h1><p> <i>Título</i>: <b> Anidamiento y agrupamiento </b> de selectores </p></h1>
<ul>
<li class=”nivel1”> Agrupamiento</li>
<li class=”nivel1”> Anidamiento </li>
<ul>
<li class=”nivel2”> Anidamiento de hijos</li>
<li class=”nivel2”> Anidamiento de adyacentes </li>
<li> Anidamiento común</li>
</ul>
</ul>
</body>
```

2.4 BUENAS PRÁCTICAS AL ESCRIBIR CSS

Hasta el momento los ejemplos usados en este capítulo eran muy básicos, por tanto, no necesitaban de un número elevado de reglas. Sin embargo, en un trabajo real, las reglas CSS se multiplican para cualquier cosa que se quiera hacer de manera profesional en una web. Si a eso se le suma que cada selector puede tener del orden de 5 o más atributos, el CSS resultante puede ser ilegible si no se estructura con cuidado.

Como se comprobará en ejemplos posteriores, el gran número de reglas que puede tener un CSS necesita que el desarrollador haga un esfuerzo a la hora de escribir los selectores con el fin de que sea más fácil hacer modificaciones posteriores. Las siguientes recomendaciones no están definidas en el estándar W3C, son solo *buenas prácticas* que la mayoría de los desarrolladores de CSS suelen tener en cuenta. Seguir estas prácticas no solo ayuda a un desarrollo propio más claro, sino que también ayudan a entender mejor desarrollo de otros autores.

En la Actividad 2.4 se ha usado como selector de clase el siguiente:*.nivel1 {color:green}*, pero también se podría haber usado este otro: *.ff8 {color:green}*. Ambas declaraciones tienen algo en común: no son muy descriptivas de lo que quieren definir. A continuación, se muestra una batería de propuesta de buenas prácticas:

- **Los selectores se nombran en minúsculas, nunca empezando por caracteres especiales o numéricos**: como en cualquier otro lenguaje de etiquetas, hacerlo así da más claridad al selector (además de que la especificación CSS no lo permite para nombre de selectores de clase e identificadores). CSS en principio no distingue entre mayúsculas y minúsculas, salvo en los nombres de selectores de clase e identificadores, pero se eliminan errores si todo se pone en minúsculas. Por último, aunque en las últimas versiones se permite usar "_" en los nombres, es su lugar (como luego veremos) es preferible usar "-", ya que no todos los navegadores soportan "_".

- **El nombre de los selectores debe ser específico y claro, para que tenga una mayor capacidad expresiva**: los ejemplos anteriores se podrían sustituir por *lista-nivel1{color:green}*, de esa manera se entiende mejor que lo que se pretender es dar un estilo para un elemento de una lista que está a *nivel1*.

- **El nombre de las clases e identificadores no debe describir una característica visual, como color, tamaño o posición**: es mejor no usar nombre asociado a color, tamaño o posición porque hace que el selector soporte peor los cambios. Si a una regla se le llama *.rojo{color:rojo}*, entonces si por cualquier motivo (que los hay) se cambiar el color de la clase, también se debería cambiar el del selector, con los problemas que eso lleva al tener que actualizar todas las referencias a esa clase en el HTML.

- **Los nombres deben seguir más una visión semántica que estructural**: por el mismo criterio de facilitar los cambios, no se deben usar nombre de selectores según la localización si no se dan otras alternativas en el mismo CSS. Por ejemplo, si se usa un selector de clase menú-izquierda{…} para referirse a un menú situado a la izquierda, si por cualquier motivo se quiere cambiar a la derecha, entonces hay que cambiar también el nombre del selector y de las referencias HTML. Es mejor usar menu-navegacion {…} para definir este selector y no asociarlo a la izquierda. Otra cosa, sería si se desea hacer un menú a la izquierda y otro a la derecha, y que sea el usuario el que seleccione el que más le gusta (como ocurre en CMS como Joomla!).

  Una alternativa semántica define los selectores según su función y no la estructura donde estará alojada. Por ejemplo, a un es preferible asociarle una clase llamada *.main* que una llamada *.left-content* (contenidos de la izquierda) ya que si por cualquier motivo se cambia de lugar se tienen que cambiar sus propiedades y las referencias en el HTML.

  Si en cualquier caso el desarrollador piensa que es necesario definir una clase única y exclusivamente para alinear una imagen o un párrafo y llamarla con esa acción, lo importante es documentarlo apropiadamente, para que él u otros desarrolladores no tengan problemas en el futuro. En CSS los comentarios se ponen como en Lenguaje C:

  */*texto del comentario */*

- **Separa las palabras mediante guiones o mayúsculas**: así se le da más claridad a los nombre de los selectores. Por ejemplo *menu-superior* en vez de *menú-navegacion* lo hace más legible.

- **No hacer uso excesivo de clases**: esto es útil ya que, a menudo, es más sencillo utilizar selectores contextuales o anidar selectores que no alteren el HTML, o incluso usar selectores de etiquetas para conseguir lo mismo. Los selectores de clase se suelen dejar para las partes más relevantes de la estructura.

Por ejemplo, en vez de usar:

```
<div class="main">
<div class="main-title">...</div>
<div class="main-paragraph">...</div>
</div>
```

Usar esta estructura ya que lo que se busca es el mismo efecto:

```
<div class="main">
<h1>...</h1>
<p>...</p>
</div>
```

- **Agrupar las reglas según su selector siempre que sea posible**: cuando hay varias reglas con el mismo selector (sea del tipo que sea) es interesante agruparlas unas debajo de otras. Por ejemplo:

```
[…]
table {border:double}
table.miembros {border:solid}
table.empleados {border:grrove}
[…]
```

- **Al principio de un CSS es aconsejable definir los selectores de etiquetas**: además se usan comentarios para dejar claro cuál es la parte que define los selectores de etiquetas y cuál es la parte que definen las clases y otros elementos. Por ejemplo:

/ Etiquetas HTML */*

```
html {font-family:arial, verdana, sans serif; font-size:13px;}
h1, h2, h3, h4, h5, h6, form, input, text-area{
border:0; padding:0; margin:0;
font-family:arial;}
h1{font-size:24px; color:#000000;}
h2{font-size:18px; color:#666666;}
```

/ FIN Etiquetas HTML */*

- **Estructurar visualmente los atributos**: si un elemento solo tiene tres atributos se pueden poner en la misma línea. Pero si hay más, se ponen en líneas diferentes sangrados con tabuladores. En el ejemplo anterior se puede ver este uso.

Estas son solo algunas propuestas de buenas prácticas. Sin embargo, conforme el diseñador profundiza en el desarrollo de CSS adquirirá muchas otras que darán más legibilidad a sus CSS.

2.5 ATRIBUTOS. MODELO DE CAJAS

Hasta el momento, todos los ejemplos anteriores estaban relacionados con la sintaxis de CSS. Por ello, en los ejemplos siempre se ha jugado con propiedades muy sencillas y fáciles de entender, como el color (*color*:), la fuente (*font-family*) o el tamaño de la fuente (*font-size*). Estas tres propiedades (*color*, *font-family* y *font-size*) se llaman *atributos*.

La potencia de los CSS radica principalmente en la gran cantidad de atributos que se pueden usar para dar estilo a las páginas web. Estos van desde la posición que puede ocupar un determinado elemento hasta la colocación de una imagen de fondo.

En esta sección se explicarán aspectos básicos de localización de elementos en las páginas y el significado de los atributos que definen esas distancias.

Para entender los estilos CSS lo más adecuado es pensar en cualquier elemento (<h1>,<p>, , etc.) como una caja. Para cada caja las dimensiones pueden ser controladas para producir una gran variedad de efectos.

Con esta definición, una página es una caja de cajas, y a esta simplificación se le llama *modelo de cajas*. La Figura 2.10 muestra los atributos de una caja. Este esquema es básico para conocer cómo afecta un determinado valor a un contenido. Es importante destacar, que en principio este esquema es seguido por todos los navegadores. Sin embargo, algunos navegadores como Internet Explorer lo interpretan de diferente manera. Por ejemplo, para Internet Explorer 6.0 el ancho del contenido va desde el *margin-left* hasta *margin-right*, sin tener en cuenta el ancho del borde ni del relleno (*padding*). En las siguientes explicaciones no haremos excepciones y se explicarán los atributos según índica el estándar W3C.

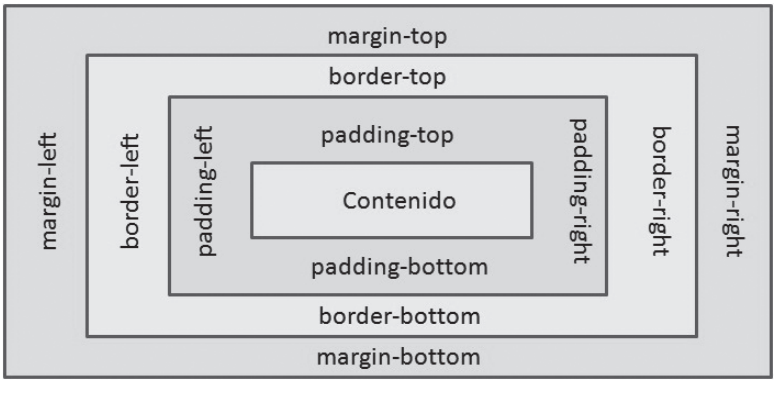

Figura 2.10. *Atributos de una caja*

Las cajas tienen varios atributos relacionados con los colores, imágenes, etc. Sin embargo esos atributos se verán en la siguiente sección.

Esta sección se describen los atributos que afectan a las dimensiones y posición de la caja:

- Atributos de posición de la caja.
- Los atributos de los márgenes, que asignan un borde externo a la caja.
- Los atributos de relleno (*padding*) asignan un espacio interno dentro de la caja para separar el contenido de los márgenes.
- Los atributos de los bordes, que definen las líneas gráficas alrededor de la caja.

2.5.1 UNIDADES DE MEDIDA

Los valores de estos atributos se pueden expresar en varias unidades absolutas: pulgadas, centímetros, milímetros, puntos, picas, y unidades relativas: em, ex y px.

- pulgadas (in). Una pulgada = 2,54 cm.
- centímetros (cm).
- milímetros (mm).
- puntos (pt). Un punto = 1/72 de pulgada.
- picas (pc). Una pica = 12 puntos.

El problema de las unidades absolutas es que siempre dependerán del entorno para el que fueron desarrolladas, y no siempre son portables a otros entornos. Por ejemplo, la medida punto depende de la resolución de la pantalla del usuario.

Por lo tanto, se recomienda el uso de unidades relativas. La menos aconsejable de estas son los píxeles ya que dependen de la resolución de pantalla y del tipo de ordenador: un sistema operativo Windows mantiene una equivalencia de 96px por pulgada y un Macintosh de 72 px por pulgada.

La unidad más aconsejable es em. Esta unidad es igual a la altura (*font-size*) de la letra del elemento en el que se usa. Por ejemplo, si para un párrafo especificamos una sangría de 2em el largo de la sangría será igual a dos veces el tamaño de la letra de ese párrafo. En el siguiente ejemplo la sangría es de 22 px (11px*2 del em).

```
p { font-size:11px;text-indent: 2em; }
```

Para el tamaño de letra. si el párrafo está contenido en un elemento , el tamaño de la letra sería de 18px (un 20% mayor que el especificado para dicho div). Si no estuviera contenido en un , un 20% mayor que el tamaño de letra del elemento del que descienda (por ejemplo, <body>).

```
div { font-size:15px;}
p { font-size:1.2em;}
```

Por último están los porcentajes que son muy utilizados. Un valor de porcentaje se forma por un número y el signo %. No hay espacios en un valor de porcentaje. Los valores de porcentaje se fijan en relación a otro. Generalmente, el valor de porcentaje es relativo al tamaño de fuente del elemento:

ACTIVIDADES 2.5

Compruebe en un navegador las diferencias de tamaño con los siguientes códigos. Observar que en este caso el estilo no está definido en el <head> del documento, sino que incorporado en la propia etiqueta (atributo *style*). Este método viene bien para poner ejemplos rápidos como en este caso, pero no para diseñar debido a su poca escalabilidad.

```
<body>
<div style="font-size:0.2in;">Texto de 0.2 pulgadas</div>
<div style="font-size:0.3in;">Texto de 0.3 pulgadas</div>
<div style="font-size:0.4in;">Texto de 0.4 pulgadas</div>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
<div style="font-size:10px;">Texto de 10 píxeles</div>
<div style="font-size:12px;">Texto de 12 píxeles</div>
<div style="font-size:14px;">Texto de 14 píxeles</div>
<div style="font-size:16px;">Texto de 16 píxeles</div>
<div style="font-size:18px;">Texto de 18 píxeles</div>
<div style="font-size:20px;">Texto de 20 píxeles</div>
<div style="font-size:8mm;">Texto de 8 milímetros</div>
<div style="font-size:9mm;">Texto de 9 milímetros</div>
<div style="font-size:10mm;">Texto de 10 milímetros</div>
<div style="font-size:11mm;">Texto de 11 milímetros</div>
<div style="font-size:1pc;">Texto de 1 picas</div>
<div style="font-size:2pc;">Texto de 2 picas</div>
<div style="font-size:3pc;">Texto de 3 picas</div>
<div style="font-size:4pc;">Texto de 4 picas</div>

</body>
```

2.5.2 ATRIBUTOS DE POSICIÓN

Los atributos de posición son principalmente *top*, *left*, *right* y *bottom*. *Top* (desde arriba) y *bottom* (desde abajo) indican la distancia en vertical donde se colocará la capa y *left* (desde la izquierda) y *right* (desde la derecha) la horizontal. Sin embargo, esta distancia está supeditada al atributo *position* que define el tipo de posición de la capa. Si el atributo position es *absolute* (o *fixed*), *top* indica la distancia del borde superior de la capa con respecto al borde superior de la página. Si el atributo position era *relative*, *top* indica la distancia desde donde se estaba escribiendo en ese momento en la página hasta el borde superior de la capa.

En la sección 2.7 se tratan más en profundidad el atributo *position* y sus posibilidades.

2.5.3 ATRIBUTOS MARGIN

Como se puede apreciar en la Figura 2.10, los atributos *margin-left*, *margin-right*, *margin-top*, *margin-bottom* marcan la separación entre otra caja y el borde de la que se representa. El siguiente ejemplo muestra estos atributos definidos en dos cajas: una para <body> y otra para . El ejemplo se hace para las dos porque de esa manera se puede ver que las medidas se hacen relativas a la caja que la contiene, en este caso las medidas de los márgenes del se hacen relativas al <body>. Para visualizar mejor el efecto se ha incluido un atributo *border* que, como veremos más adelante, dibuja el borde de 3 px en rojo para el y azul para el <body>.

```
<head>
<style>
body {
 margin-top: 100px;
margin-right:100px;
margin-bottom: 100px;
margin-left: 100px ;
border : 3px dotted blue ;}
div {
margin-top: 15px;
margin-right:15px;
margin-bottom: 15px;
margin-left: 15px;
border : 3px dotted red ; }
</style>


</head>
<body>
<div style=”font-size:0.5in;”>Texto de 0.5 pulgadas</div>
</body>
```

El resultado de este código es mostrado en la Figura 2.11:

Figura 2.11. *Márgenes entre body y div*

Al no haber relleno (*padding*) la distancia entre el borde del cuadro exterior (azul) y el más interno (rojo) es de unos 15 px. La distancia de 100 px es entre el borde exterior (azul) y los bordes de la vista del navegador. S i se incrementan los márgenes de 15 px a 150 px se puede observar que la distancia es mayor entre ambos bordes. También se puede poner una distancia de -15 px en vez de 15 px para conseguir que el borde interior (rojo) salga quede por fuera 15 px del exterior (azul). Sin embargo, no es muy aconsejable usar medidas negativas para las distancias.

Para simplificar también se puede usar un atributo *margin* que tiene 4 valores separados por espacios en blanco y cuyo orden es el siguiente (a favor de las agujas del reloj): el primer margen superior (*margin-top*), el derecho (*margin-right*), el inferior (*margin-bottom*) y el izquierdo (*margin-left*). Si se proporcionan solo dos o tres, los que faltan se asignan automáticamente según la relación con otras cajas. Para el ejemplo anterior, *body* se podía haber definido de esta manera:

```
body {
 margin: 100px;100px 100px 100px ;
border : 3px dotted blue ;
}
```

Los valores de estos atributos (y de cualquier de los siguientes que veremos) también pueden ser *auto o inherit*. Auto calcula automáticamente la mínima distancia según la relación con otros elementos. Esto es útil cuando la relaciones se hacen con medidas relativas. Por su lado, *inherit* hereda el valor del mismo atributo en la caja que lo contiene, es decir, hereda los valores del padre.

2.5.4 ATRIBUTOS PADDING

Padding se puede traducir como relleno. Como se puede apreciar en la Figura 2.10, estos atributos indican la distancia entre el borde y los elementos que se encuentras en el interior. Dicho de otro modo, tienen una función opuesta a la vista anteriormente para el atributo *margin*. Las medidas que se usan son las mismas que para *margin* y los nombres son muy parecidos: *padding-top* (relleno superior), *padding-right* (relleno derecho), *padding-bottom* (relleno inferior) y *padding-left* (relleno izquierdo).

Observar como en la Figura 2.11 la distancia entre la "T" de *Texto de 0.2 pulgadas* y el borde rojo es inexistente. Esto es debido a que el *padding-left* no ha sido definido, con lo que se coge el valor mínimo. El siguiente ejemplo asigna *padding-left* de 10 px para apreciar la diferencia.

```
<head>
<style>
body {
 margin: 100px 100px 100px 100px ;
border : 3px dotted blue ;}
div {
    margin-top:15px;
    margin-right:15px;
    margin-bottom: 15px;
    margin-left: 15px;
    padding-left: 10px;
    border : 3px dotted red ; }
</style>
```

```
</head>
<body>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
</body>
```

El resultado se puede ver en la Figura 2.12 (el borde del cuadro más exterior es azul y el más interior es rojo como podrá verse al probarlo en un navegador):

Figura 2.12. *div con padding-left de 15 px*

Como ocurre con el atributo *margin*, también hay una atributo *padding* que simplifica el escribir el nombre de los 4 atributos. Un ejemplo de uso es el siguiente:

```
body {
 padding: 100px;100px 100px 100px ;
border : 3px dotted blue ;
}
```

2.5.5 ATRIBUTOS BORDER-TOP, BORDER-BOTTOM, BORDER-RIGHT, BORDER-LEFT

Como se puede apreciar en la Figura 2.10, estos atributos definen el estilo y color del borde de la caja. Algunas de las palabras clave que tienen son: *none*, *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset* y *outset:*

En el ejemplo de la Figura 2.11 se ha usado un *atributo border: 3px dotted red ;* para indicar que todo el borde de la caja sea rojo, punteado con puntos de 3 píxeles.

Se puede separar la asignación de color y estilo usando dos atributos separados: *border-color y border-style.* Estos atributos aplican respectivamente un color y un estilo a todos los bordes de la caja.

También se puede especificar un color y estilo para cada uno de los bordes (*top*, *left*, *right*, *bottom*) por separado con *border-top, border-bottom, border-right, border-left.*

Los atributos mostrados no son todos los que definen CSS (versión 3) para manejar bordes. Existen muchos más, como por ejemplo, *border-radius* que se usa para hacer bordes redondeados. Este atributo se utiliza en el siguiente ejemplo.

Por último, también se puede definir por separado el ancho de un borde con la propiedad *border-width* para todos los bordes de la caja o con *border-top-width*, *border-bottom-width*, *border-right-width*, *border-left-width* para cada uno por separado. Los anchos (*width*) pueden tener como valor un tamaño en cualquier unidad como los visto anteriormente, o valores predefinidos: *thin* (estrecho), *medium* (mediano) o *thick* (ancho).

El siguiente ejemplo muestra el uso de muchos de los atributos visto para hacer una caja dentro de otra con borde redondeado.

```
<head>
<style>
body {
     margin: 100px 100px 100px 100px ;
     border-style:inset;
     border-color:blue; /* color del borde azul */
     border-radius: 15px; /*borde redondeado de radio 15px */
     border-width:thick;  /* Un borde grueso */
     padding: 15px 15px 15px 15px;
    }


div {
           margin-top:15px;
           margin-right:15px;
           margin-bottom: 15px;
           margin-left: 15px;
           padding-left: 10px;
           border-top : 3px dotted red ; /*estilo, tamaño y color incluidos en el mismo
atributo */
           border-right : 2px solid blue ;
           border-bottom : 3px double green ;
           border-left : 3px groove red ;
    }
</style>


</head>
<body>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
</body>
```

El resultado se puede ver en la Figura 2.13 mostrada en Chrome 14.0 (el cuadro más exterior es de color azul y el interior tiene el lado derecho azul, el inferior verde y los otros dos lado rojos, como se puede ver al probarlo en un navegador).

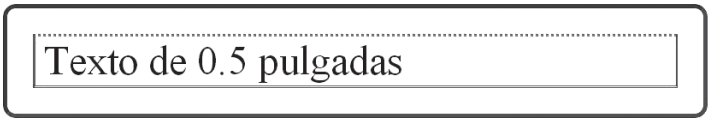

Figura 2.13. *Aplicación del atributo Border*

2.5.6 ATRIBUTOS DEL CONTENIDO

También se puede concretar el ancho y alto de un contenido (ver Figura 2.10) con el atributo *width* y *height*. *Width* establece la distancia entre el límite del *padding-left* y el *padding-right* (así es como viene definido en el CSS, aunque navegadores como Internet Explorer 6 no los considera así). *Height* igual pero entre el *padding-bottom* y el *padding-top*.

En esta a sección se ha pretendido hacer una introducción a los atributos más importantes que participan en el modelo de cajas, con la idea de que el desarrollador se forme un mapa mental de cómo pueden estar distribuidos los elementos en una web. Sin embargo, no se han incluido todas las posibilidades que ofrece CSS en este tipo de atributos. Se recomienda consultar las siguientes URL para conocer más sobre el tema.

Todos los atributos disponibles en CSS 2.1 de la W3C:

http://www.w3c.es/divulgacion/guiasreferencia/css21/#modeloCajas

Esta otra URL muestra las nuevas incorporaciones de CSS3, en la que se puede ver *border-radius*.

http://www.css3.info/preview/

ACTIVIDADES 2.6

El siguiente código muestra una estructura básica de página web con un encabezado una parte para el menú y otra parte para los contenidos. Este es el esqueleto básico de una web para una Restaurante. Como se puede observar, esta estructura está hecha con clases e identificador con el fin de dar mayor flexibilidad.

```
<head>
<style>

div#body  /*Define una identificador "body" asociado a un elemento <div>. Es la
posición del cuerpo de la página (barra lateral más contenido */
{
margin:auto auto auto auto;  /* top right bottom left: equivale a poner un solo
auto */
width:710px; /* El ancho del estilo lo coloca en 710px */
}
div#header /*Define una identificador "header" asociado a un elemento <div>. Es la
posición de la cabecera */
{
margin:10px auto 10px auto;
width:710px;
border:dotted 1px #ccc;
}
div#sidebar
```

```
{
display:none;
border:solid 1px #CCC;
}
body.main-sidebar div#main, body.sidebar-main div#main
{
width:490px;
}
body.main-sidebar div#sidebar, body.sidebar-main div#sidebar
{
display:block;
width:200px;
border:solid 1px #CCC;
}
body.main-sidebar div#main, body.main-sidebar div#sidebar  /*ms hace referencia a
main-sidebar. Este estilo coloca el sidebar a la derecha y el main a la izquierda */
{
float:left;
border:solid 1px #CCC;
}
body.sidebar-main div#main, body.sidebar-main div#sidebar /*sm hace referencia a
sidebar-main. Este estilo coloca el sidebar a la izquierda y el main a la derecha*/
{
float:right;
border:solid 1px #CCC;
}

</style>
</head>
<body class=”main-sidebar”>
<div id=”header”>Cabecera. RESTAURANTE: El muslito</div>
<div id=”body”>
<div id=”main”>Este restaurante es una de las referencias gastronómicas del esta
zona. [...] </div>
<div id=”sidebar”>Menú principal</div>
</div>
</body>
```

Analice el código. Se han definido dos clases asociadas al *body*: *body.main-sidebar* y *body.sidebar-main.* Dependiendo de qué clase se use en la etiqueta <body> el *div* asociado a la barra lateral (*sidebar*) se pondrá a la derecha o izquierda. Por lo tanto, se puede apreciar cómo de fácil es cambiar una distribución con solo cambiar <body class=”main-sidebar”> por <body class=”sidebar-main”>.

El resultado aplicado <body class=”main-sidebar”> puede verse en la Figura 2.14.

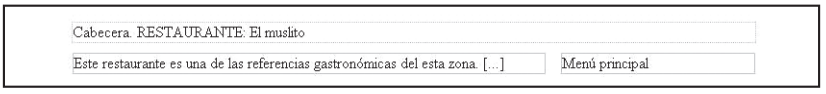

Figura 2.14. *Esqueleto de una web para una Restaurante*

- Analice el atributo *float*. El valor de *float:left* indica que la caja con este estilo se colocan a la izquierda y el resto de cajas que se crean posteriores en el código HTML, y que están en la misma posición, se colocan a la derecha de ésta. Esto es lo que le da el efecto de una caja *main* a la izquierda y el *sidebar* apoyada a su derecha. Cuando es *float:right* lo que se indica es que esa caja se coloca a la derecha. Esto da el efecto de una caja *main* a la derecha y el *sidebar* apoyada a su izquierda.

  El efecto de un *float* sigue hasta que se usa en otra caja un estilo *clear*.

  El problema surge si posteriormente se desea colocar una caja que ya no se apoye a la derecha (o izquierda) de la flotante sino que se coloque debajo como se hace por defecto (sin atributo *position*). En este caso, esa nueva caja se tiene que crear un el estilo *clear.* Si se pone *clear:left* se indica que ya no se desea colocar esa caja adyacente a una definida con *float:left*. Si pone *clear:right*, se indica que ya no se quiere colocar esa caja adyacente a una definida con *float:right.* El estilo *clear* elimina el efecto del *float*.

- Modifique este código para conseguir los siguientes efectos:

  a. Que el borde de la cabecera sea en azul, solido, redondeado con un ancho de 3 px.

  b. Que el borde del menú principal sea de 1 px, en verde y punteado (*dotted*) y con la palabra "Menú Principal" separada 15 px del borde de la caja (solo el borde izquierdo).

  c. ¿Qué ocurre si el borde es de tamaño 5 px?

  d. Define un pie de página igual de ancho y alto que el encabezado pero de color de borde verde (PISTA: Usar en ese nueva caja un estilo con *clear*).

  e. ¿Qué ocurre si no se usa un estilo *clear* en el pie de página?

  El resultado debe ser algo similar a la Figura 2.15 usando el navegador Chrome 14.0 (el cuadro de borde redondeado en el encabezado es azul y el del pie de página es verde). Internet Explorer 8.0 no muestra el efecto curvo, sino que usa cajas con ángulos rectos (es decir, no interpreta *border-radius*):

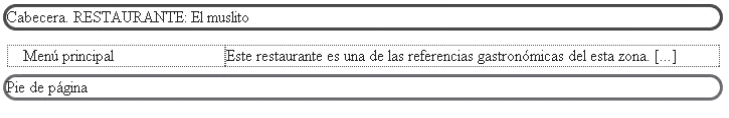

Figura 2.15. *Esqueleto con pie de página no flotante*

Solución

El siguiente código muestra la solución a la actividad.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<style>
```

```
div#body  /*Define una identificador "body" asociado a un elemento <div>. Es la
posición del cuerpo de la página (barra lateral más contenido) */
{
margin:auto auto auto auto;  /* top right bottom left: equivale a poner un solo
auto */
width:710px; /* El ancho del estilo lo coloca en 710px */
}
div#header /*Define una identificador "header" asociado a un elemento <div>. Es la
posición de la cabecera */
{
margin:10px auto 10px auto;
width:710px;
border:solid 3px blue;
border-radius: 15px; /*borde redondeado de radio 15px */
}
div#sidebar
{
display:none;
border:solid 1px #CFC;
padding-left:15px; /*Se modifica el ancho*/
}
body.sidebar-main div#footer  /*Define una identificador "footer" asociado a un
elemento <div>. Es la posición del pie de página */
{
margin:25px auto 25px auto;
width:710px;
border:solid 3px green;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:right;/* Deja sin utilidad la opción de float:right */
}

body.main-sidebar div#footer /*Define una identificador "footer" asociado a un
elemento <div>. Es la posición del pie de página */
{
margin:15px auto 15px auto;

width:710px;
border:solid 3px blue;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:left;/* Deja sin utilidad la opción de float:left*/
}
body.main-sidebar div#main, body.sidebar-main div#main
{
width:490px;
```

```
border:solid 4px green;

}
body.main-sidebar div#body, body.sidebar-main div#body
{
margin:auto auto auto auto;
width:710px;
border:solid 4px #FFF;
}
body.main-sidebar div#sidebar, body.sidebar-main div#sidebar
{
display:block;
width:200px;
border:solid 1px #CCC;
}
body.main-sidebar div#main, body.main-sidebar div#sidebar  /*Main-sidebar. Este
estilo coloca el sidebar a la derecha y el main a la izquierda */
{
float:left;
border:dotted 1px green;
}
body.sidebar-main div#main, body.sidebar-main div#sidebar /*Sidebar-main. Este
estilo coloca el sidebar a la izquierda y el main a la derecha*/
{
float:right;
border:dotted 1px green;
}

</style>
</head>
<body class="sidebar-main">
<div id="header">Cabecera. RESTAURANTE: El muslito</div>

<div id="body">
<div id="main">Este restaurante es una de las referencias gastronómicas del esta
zona. [...] </div>
<div id="sidebar">Menú principal </div>
</div>
<div id="footer">Pie de página</div>
</body>
</html>
```

2.6 ELEMENTOS: COLORES DE FONDO, TEXTOS, ENLACES, LISTAS, TABLAS, VISIBILIDAD, IMÁGENES

Una vez vistos los atributos asociados a las clases. En esta sección se muestran atributos adicionales de carácter más general y relacionados con la apariencia de textos, listas, tablas, enlaces e imágenes.

Atributos de fuentes

- **Color**: *RGB* o *nombre de color*. Sirve para indicar el color del texto. Lo admiten casi todas las etiquetas de HTML. Admite nombres de colores en inglés para algunos colores y valores RGB para todos.
- **font-size**: *unidades* | *xx-small* | *x-small* | *small* | *medium* | *large* | *x-large* | *xx-large*. Sirve para determinar el tamaño de una fuente.
- **font-family**: *serif* | *sans-serif* |*cursive* | *fantasy* | *monospace* . Con este atributo indicamos la familia de tipografía del texto. Los primeros valores son genéricos, es decir, los navegadores las comprenden y utilizan las fuentes que el usuario tenga en su sistema. También se pueden definir con tipografías normales, como ocurría en HTML. Si el nombre de una fuente tiene espacios se utilizan comillas para que se entienda bien.
- **font-weight**: *normal* | *bold* | *bolder* | *lighter* | *100* | *200* | *300* | *400* | *500* | *600* | *700* | *800* | *900*. Sirve para definir la anchura de los caracteres, es decir, efecto de negrita. Normal y 400 son el mismo valor, así como *bold* y 700.
- **font-style**: *normal* | *italic* | *oblique*. Es el estilo de la fuente. El estilo *oblique* es similar al italic.

Atributos de párrafos

- **line-height**: *normal* | *unidades*. El alto de una línea, y por tanto, el espaciado entre líneas. Es una de esas características que no se pueden modificar con HTML.
- **text-decoration**: *none* | *underline* | *overline* | *line-through*. Para establecer la decoración de un texto, es decir, si está subrayado, sobre-rayado o tachado.
- **text-align**: *left* | *right* | *center* |*justify*. Sirve para indicar la alineación del texto. Es interesante destacar que las hojas de estilo permiten el justificado de texto, aunque no funciona en todos los navegadores.
- **text-indent**: *Unidades*. Un atributo que sirve para hacer sangrado o márgenes en las páginas.
- **text-transform**: *capitalize* | *uppercase* |*lowercase* | *none text-transform*|*none*. Permite transformar el texto haciendo que tenga la primera letra en mayúsculas de todas las palabras, todo en mayúsculas o minúsculas.

Atributos de fondo

- **Background-color**: *RGB* o *nombre de color*. Sirve para indicar el color de fondo de un elemento de la página.
- **Background-image**: nombre de la imagen con su camino relativo o absoluto.

Atributos tablas

- **caption-side**: *valores top | bottom*. Posición del título.
- **table-layout**: *auto |fixed*. Control del algoritmo usado para el formato de las celdas, filas y columnas.
- **border-collapse**: *collapse| separate*. Selección del modelo de los bordes.
- **border-spacing**: *unidades*. Espaciado entre los bordes de celdas adyacentes.
- **empty-cells**: *show | hide*. Visibilidad de los bordes de celdas sin contenido (ocultar o mostrar).

Atributos visibilidad

- **Overflow**: *visible|hidden |scroll|auto*. Comportamiento del contenido si se desborda en la caja
- **Clip**: *rect (top,right,bottom, left) | auto*. Especifica la región visible del elemento mediante las dimensiones de un rectángulo que hace de ventana de visualización.
- **Visibility**: *visible|hidden|collapse*. Visibilidad de las cajas. No se reorganizan las cajas de alrededor, solo se oculta.
- **Display**: *inline | block | none | list-item | run-in | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | inherit*. Muestra una caja con diferentes estilos. El más común es *none*, que se diferencia de *visibility:hidden* en que en este caso las cajas de alrededor se reorganizan cuando se oculta.

Atributos de listas

- **list-style-type**: *disc|circle|square|decimal|decimal-leading-zero|lower-roman|upper-roman|lower-greek|lower-latin|upper-latin|armenian| georgian|lower-alpha|upper-alpha|none*. Estilo aplicable a los marcadores visuales de las listas.
- **list-style-image**: *url("http://...")|none*. Imagen aplicable a los elementos de las listas.
- **list-style-position**: *inside|outside*. Posición dentro de la lista de los elementos marcadores de las listas.

Atributos de enlaces

Los enlaces en CSS se pueden manejar con bastante libertad comparado con HTML. CSS permite definir estilos en los enlaces, quitando el subrayado o hacer enlaces en la misma página con distintos colores. Para aplicar estilo a los enlaces debemos definirlos para los distintos tipos de enlaces, que son:

- **Enlaces normales**: *A:link {atributos}*
- **Enlaces visitados**: *A:visited {atributos}*
- **Enlaces activos**: *A:active {atributos}* (Los enlaces están activos en el preciso momento en que se pulsa sobre ellos).
- **Enlaces hover**: *A:hover {atributos}* (Cuando el ratón está encima de ellos).

El atributo para definir enlaces sin subrayado es *text-decoration:none*, y para darles color es con el conocido atributo *color*.

El siguiente ejemplo muestra una definición personalizada de un menú de enlaces basados en los atributos vistos. Para los enlaces visitados se quita en subrayado y se pone en color. Si el ratón se coloca encima de un enlace aparece subrayado (*underline*), si se seleccionan todos pasan a color gris rodeados por un marco.

```
<html>
<head>
<style>
a.menus:link
{text-decoration:none;
color: #000000;
border:#FFFFFF 1px solid;
} /* Link no visitado*/


a.menus:visited {
text-decoration:none;
color:#cccccc;
} /*Link visitado*/


a.menus:active {
text-decoration:none;
color: #003399;
background: #green;
border:#FFFFFF 1px solid;
} /*Link activo*/


a.menus:hover {
text-decoration:underline;
color: #003399;
background: #red;
border:#FFFFFF 1px solid;}
/*Ratón sobre el link*/


</style>
</head>
<body>
<a href="#" class="menus">Enlace uno</a>
<a href="#" class="menus">Enlace dos </a>
<a href="#" class="menus">Enlace tres</a>
</body>
</html>
```

La Figura 2.16 muestra los enlaces ya visitados y uno de ellos con el ratón encima (*hover*). El color de fondo del último enlace es rojo.

Figura 2.16. *Ejemplo de menú*

Aunque se han mostrado en esta sección los atributos más usados, faltan muchos por conocer. Se recomienda consultar las siguientes URL para conocer más sobre el tema. Todos los atributos disponibles en CSS 2.1 de la W3C pueden consultarse en *http://www.w3c.es/divulgacion/guiasreferencia/css21* y en esta otra URL se presentan las nuevas incorporaciones de CSS3 *http://www.css3.info/preview/*.

ACTIVIDADES 2.7

Modifique el código del esqueleto mostrado en la Actividad 2.6 para incluir los siguientes elementos:

a. Dividir el encabezado *header* en dos partes. Colocar un logo de Superman en una de ellas y un texto <h1> a lado del logo.

b. Colocar una imagen Superman en el contenidos *main.*

c. Hacer un menú con 4 enlaces en el *sidebar* que no lleve a ningún enlace pero que si tenga atributos para sus enlaces: activo, *hover*, etc.

El resultado debe quedar algo como muestra la Figura 2.17 (usando el navegador Chrome 14.0).

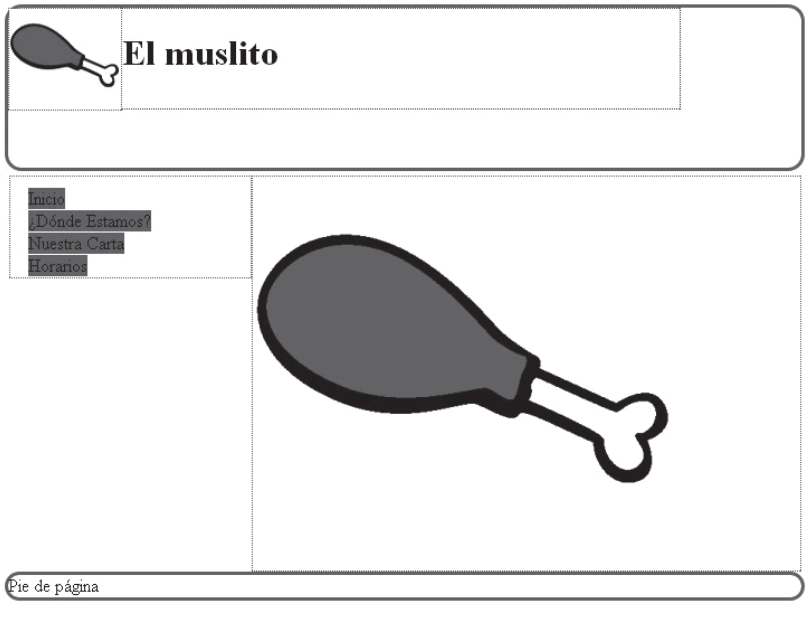

Figura 2.17. *Esqueleto mejorado con menú e imágenes*

Solución: (el siguiente código muestra la solución a la actividad)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<style>

div#body  /*Define una identificador "body" asociado a un elemento <div>. Es la posición del cuerpo de la página (barra lateral más contenido) */
{
margin:auto auto auto auto;  /* top right bottom left: equivale a poner un solo auto */
width:710px; /* El ancho del estilo lo coloca en 710px */
clear:left; /*En el body quito el flotante para que la siguiente caja quede debajo */
}
div#header /*Define una identificador "header" asociado a un elemento <div>. Es la posición de la cabecera */
{
margin:auto auto auto auto;
width:710px;
border:solid 3px #815608;
border-radius: 15px; /*borde redondeado de radio 15px */
}
div#sidebar
{
display:none; /*este div no lo hace visible */
border:solid 1px #CFC;
padding-left:15px; /*Se modifica el ancho*/
}
body.sidebar-main div#footer  /*Define una identificador "footer" asociado a un elemento <div>. Es la posición del pie de página */
{
margin:25px auto 25px auto;
width:710px;
border:solid 3px #815608;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:right;/* Ya que el div encima es tipo float:right, es necesario decirle ahora que coloque este div cuando no hay ningún div que este flotando a la derecha */
}

body.main-sidebar div#footer /*Define una identificador "footer" asociado a un elemento <div>. Es la posición del pie de página */
{
margin:15px auto 15px auto;
```

```
width:710px;
border:solid 3px #815608;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:left;/* Ya que el div encima es tipo float:left, es necesario decirle ahora
que coloque este div cuando no hay ningún div que este flotando a la izquierda*/
}

body.main-sidebar div#main, body.sidebar-main div#main
{
width:490px;
border:solid 4px green;

}
body.main-sidebar div#body, body.sidebar-main div#body
{
margin:auto auto auto auto;
width:710px;
border:solid 4px #FFF;
}

body.main-sidebar div#sidebar, body.sidebar-main div#sidebar
{
padding-top:10px;
display:block;
width:200px;
border:solid 1px #CCC;
}
body.main-sidebar div#main, body.main-sidebar div#sidebar  /*Main-sidebar. Este
estilo coloca el sidebar a la derecha y el main a la izquierda */
{
float:left;
border:dotted 1px green;
}
body.sidebar-main div#main, body.sidebar-main div#sidebar /*Sidebar-main. Este
estilo coloca el sidebar a la izquierda y el main a la derecha*/
{
float:right;
border:dotted 1px green;
}
div#header div#logo /*Se define el logo flotante a la izquierda*/
{
margin-bottom:inherit;
border:dotted 1px green;
float:left; /*Hago estas caja flotante, y las siguientes cajas que estén en la misma
```

```
posición también serám flotantes a la izquierda. No hace falta ponerle a div#header
div#main-title también eta propiedad*/
}
div#header div#main-title /*Es el título del encabezado. Al ser el logo flotante
este también lo será */
{
width:600px;
margin-bottom:inherit;
border:dotted 1px blue;
padding-bottom:10px;

/* DEFINE LOS ENLACES DE MEnU*/
}
a.menus:link
{text-decoration:none;
color:#000000;
background-color:#815608;
border:#FFFFFF 1px solid;
} /* Link no visitado*/

a.menus:visited {
text-decoration:none;
color:#cccccc;
} /*Link visitado*/

a.menus:active {
text-decoration:none;
color: #003399;
background: green;
border:#FFFFFF 1px solid;
} /*Link activo*/

a.menus:hover {
text-decoration:underline;
color: #003399;
background: red;
border:#FFFFFF 1px solid;}
/*Ratón sobre el link*/

</style>
</head>

<body class=”sidebar-main”>
<div id=”header”>
```

```
<div id="logo"> <img src="logo.jpg" width="100px" height="90px"></img> </div>
<div id="main-title"><h1> El muslito </h1></div>
<p> </p>
</div>

<div id="body">
<div id="main"><img src="portada.png" > </div>
<div id="sidebar">
<div><a href="#" class="menus">Inicio</a></div>
<div><a href="#" class="menus">¿Dónde Estamos?</a></div>
<div><a href="#" class="menus">Nuestra Carta</a></div>
<div><a href="#" class="menus">Horarios</a></div>

 </div>

</div>
<div id="footer">Pie de página</div>
</body>
</html>
```

2.7 SUPERPOSICIÓN Y PRECEDENCIA DE ESTILOS

Para terminar con el uso de CSS, en esta sección se tratarán dos importantes aspectos: la *superposición de cajas*, que está relacionada con lo visto en la sección 2.5 y la *precedencia de estilos*. Controlar estos aspectos supone entender bien el funcionamiento del CSS y conseguir resultados muy precisos y personales.

2.7.1 SUPERPOSICIÓN DE CAJAS

En el apartado 2.5 se mostró todo lo relacionado con el posicionamiento de las cajas, tanto vertical como horizontalmente- Sin embargo, además de estas dos dimensiones, CSS permite controlar la profundidad de las cajas determinando el orden de superposición de éstas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas, haciendo efectos de solapamiento.

El atributo *z-index* permite definir el nivel de profundidad de una caja. Su valor es un número entero. En principio el estándar W3C permite número negativos, pero generalmente, el valor 0 suele tomarse como en nivel más bajo. Cuanto más alto sea el valor, más *cerca* se mostrará la capa al usuario en la web, es decir, una caja con *z-index=10* se mostrará por encima de una con *z-index=9*).

El atributo *z-index* solo tiene efecto si va acompañado de otro muy importante para determinar la posición de una caja respecto a las otras. Este atributo se llama *position* y su funcionalidad está muy ligada con lo visto en la sección 2.5.

El atributo position puede tener como valores: *static*, *absolute*, *relative*, *fixed* o *inherit*. A continuación se describen cada uno de ellos:

- **Static**: es el valor predeterminado del atributo y el posicionamiento normal de los elementos en la página. Quiere decir que los elementos se colocarán según el flujo normal del HTML, es decir, según estén escritos en el propio código HTML. Por decirlo de otra manera, *static* no provoca ningún posicionamiento especial de los elementos y por tanto, los atributos *top*, *left*, *right* y *bottom* no se tendrán en cuenta.
- **Absolute**: el valor *absolute* permite posicionar cajas de manera absoluta, esto es de manera definida por valores de los atributos *top, left, bottom* y *right*. Las capas o elementos con posicionamiento absoluto quedan aparte del flujo normal del HTML no viéndose afectadas por el lugar en donde aparezcan dentro del HTML y tampoco afecta estas cajas a otras del flujo normal del HTML.

Es importante destacar que los valores *top*, *left*, *bottom* y *right* son una distancia con respecto al primer elemento contenedor que tenga un valor de position distinto de *static*. Si todos los contenedores donde esté la capa posicionada con estilo *absolute* (todos sus padres hasta llegar a <body>) son *static*, simplemente se posiciona con respecto al lado superior de la página, para el caso de *top*, el inferior para *bottom*, del lado izquierdo para *left* o el derecho, en el caso de utilizar *right*.

El siguiente código muestra el resultado de la Figura 2.18. Una modificación para comprobar el efecto de *static* podría ser la siguiente: poner las tres capas del ejemplo a *static*, eso colocaría una debajo de las otras según han sido definidas en el HTML, sin tener en cuenta los valores *top*, *right*, *bottom* y *left*.

```
<body>
<div style="position: absolute; width: 300px; height: 140px; top: 100px; left: 30px;
background-color: #ff8800; color: #fff; padding: 15px;z-index: 2;">
Esta capa tiene posicionamiento absoluto. Permite especificar top y left para colocarla
con respecto a la esquina superior izquierda.
</div>


<div style="position: static ; width: 820px; height: 30px; padding: 10px; background-
color: #ddf; top: 150px; left: 10px; z-index: 1;">Posicionamiento static. Las otras dos
capas absolutes se posicionan después de la static que se coloca arriba.</div>


<div style="position: absolute; width: 100px; height: 20px; padding: 10px; background-
color: #ddf; bottom: 10px; right: 10px;">Posicionamiento absoluto con atributos bottom
y right</div>
</body>
```

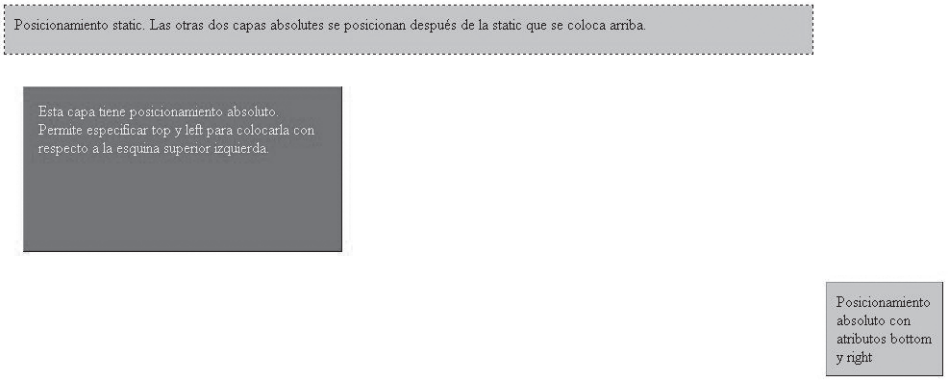

Figura 2.18. *Cajas static y absolute*

- **Relative**: indica que la capa sí forma parte del flujo normal de elementos de la página, por lo que su posición dependerá del lugar donde esté en el código y el flujo HTML. Además, las capas con posicionamiento *relative*, admiten los valores *top* y *left* para definir la distancia a la que se colocan con respecto al punto donde esté en ese momento el flujo normal del HTML. Como afectan al mencionado flujo del HTML, los elementos colocados después de las capas estilo *relative*, tendrán en cuenta sus dimensiones para continuar el flujo y saber dónde colocarse.

La Figura 2.19 muestra el resultado del siguiente ejemplo con diferencias entre *static* y *relative*.

```
<body>

<h1>NO define posición</h1>


<div style="background-color: #606; color:#ffc; padding:10px; text-align: center; width:
300px;">No define posicion, por lo que es static</div>


<div style="position: relative; width: 300px; padding: 10px; background-color: #066;
color:#ffc; top:100px; left: 30px;">Capa de posicionamiento relative<br>Se tiene en
cuenta esta capa para posicionar las siguientes.</div>


<h2>La última en ponerse</h2>


</body>
```

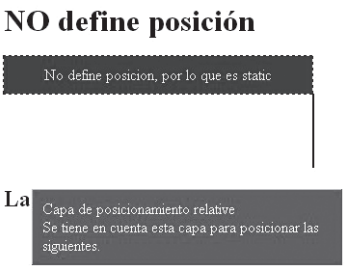

Figura 2.19. *Cajas static y relative*

Las etiquetas <h1> y <h2> respetan el flujo HTML y el que no establece posición (por defecto es *static*), por tanto, también es afectada por el flujo. Hay una capa estilo *relative*, en el segundo elemento , que también se posiciona con respecto al flujo normal. Como tiene un *top* y *left*, aparece un poco desplazada del lugar que le tocaría con respecto al flujo. El último <h2> que aparece se coloca teniendo en cuenta al flujo y tiene en cuenta la capa *relative*, por eso deja un espacio en blanco arriba, pero no atiende a la posición real de ésta, que se marcó con los atributos *top* y *left*.

- **Fixed**: este atributo sirve para posicionar una capa con posicionamiento absoluto, pero su posición final será siempre fija, es decir, aunque se desplace el documento con las barras de desplazamiento del navegador, siempre aparecerá en la misma posición. El lugar donde se "anclará" la capa siempre es relativo al cuerpo (el espacio disponible del navegador para la página). Si utilizamos *top* y *left*, estaremos marcando su posición con respecto a la esquina superior izquierda y si utilizamos *bottom* y *right* su posición será relativa a la esquina inferior derecha.

Las capas *fixed* son útiles para hacer zonas en la página web que no se muevan ni bajando la barra de desplazamiento.

- **Inherit**: indica que el valor de *position* tiene que heredarse del elemento padre. Este valor no es soportado por muchos navegadores por lo que no se utiliza mucho en la actualidad.

ACTIVIDADES 2.8

Modifique el código siguiente para colocar la capa delante del fondo.

```
<body>

<h1>NO define posición</h1>
<div style="z-index:9; position: absolute; background-color: #606; color:#ffc;
padding:10px; text-align: center; width: 300px;">No define posicion, por lo que es
static</div>
<div style="z-index:17; position: relative; width: 300px; padding: 10px; background-
color: #066; color:#ffc; left: 30px;">Capa de posicionamiento relative<br>Se tiene
en cuenta esta capa para posicionar las siguientes.</div>
<h2>La última en ponerse</h2>

</body>
```

ACTIVIDADES 2.9

Analice el siguiente código e interprete cómo quedarán las capas. Haga un bosquejo de la pantalla resultado.

```
<body>
<h1>NO define posición</h1>

<div style="background-color: #606; color:#ffc; padding:10px; text-align: center;
width: 300px;">No define posición, por lo que es static</div>

<div style="position: absolute; width: 300px; padding: 10px; background-color:
#066; text-align:right; color:#ffc; top:100px; left: 300px;">Capa Absoluta (solo
tiene como contenedor a "body") por lo tanto se relativa a body. </div>

<div style="position: relative; width: 300px; padding: 10px; background-color:
#033; color:#ffc; top:0px; left: 30px;">Capa de posicionamiento relative. Se coloca
según la posición que le tocaría en el flujo HTML.</div>

<div style="position: static; width: 300px; padding: 10px; background-color: #096;
color:#ffc; top:10px; left: 30px;">Capa Fija se coloca como una absoluta</div>

<div style="position: absolute; width: 350px; height:100px; padding: 10px;
background-color: #666; text-align:right; color:#ffc; top:300px; left: 500px;">
    <div style="position: absolute; width: 300px; padding: 10px; background-color:
#111; text-align:right; color:#ffc; top:0px;  left: 5px;"> Capa absoluta, se
referencia según a su contenedor porque no es un static </div>
</div>

<h2>La última en ponerse</h2>
</body>
```

2.7.2 PRECEDENCIA DE ESTILOS

La precedencia de estilos es una manera de indicar que un estilo definido prevalece por encima de otro definido en la misma o en CSS diferentes. Esto es necesario principalmente cuando hay dos o más estilos que actúan sobre los mismos atributos pero con diferente valor.

La precedencia de estilos va asociado con el concepto de *especificidad* de una regla. La especificidad se refiere al peso que toman cada uno de los elementos de una hoja de estilo. Cuanto más peso más especificidad. Cuanta más especificidad tenga un regla menos problemas a la hora de garantizar que será esa y no otra la regla que se aplique sobre un determinado contenido.

Un cálculo sencillo para calcular la especificidad de una regla es sumar los puntos según el tipo de selectores que contenga:

- Se da un valor de 1 punto a un *selector de etiqueta* (por ejemplo, <h1>, <p>,).
- A un *selector de clase* se le da el valor de 10 puntos.
- A un *selector de identificador* se le da un valor de 100 puntos.
- A un *atributo de estilo* a los que se les da un valor de 1.000 puntos. Este atributo es *style* y se han usado en la Actividad 2.5).

Con este cálculo, si se desea agregar un estilo a los párrafos <p> de una página agregando un selector *p{estilo}*, este será utilizado para dar estilo a todos los párrafos del HTML, pero solo se le dará el valor total de 1 punto. Un punto es muy poca especificidad. Cualquier regla con especificidad > 1 se aplicaría antes.

Si se define una clase *.parrafo {estilo}* que se aplique sobre los párrafos tendrá una especificidad de 10 puntos. Con lo cual, los párrafos <p> que usen el estilo de la clase *.parrafo* tendrán como estilo lo de la clase antes que los definidos con p{estilo}.

De la misma manera, si en vez de clase se define un selector de identificador *#id-parrafo{estilo}* esta tomará el valor de 100 puntos de especificidad, por lo que será la más importante que los de clase y el de etiqueta definidos antes.

Es importante destacar que, en el caso de que dos o más reglas en conflicto tengan el mismo valor de especificidad, se aplicará la última definida.

El siguiente código muestra las especificidades de los selectores definidos:

```
<style>
p{background: crimson;} /* Especificidad de 1 puntos */
.parrafo{background: pink;} /*Especificidad de 10 puntos*/
p.parrafo{background: maroon;} /*Especificidad de 11 puntos*/
#id-parrafo{background: orange;}/*Especificidad de 100 puntos*/
p#id-parrafo{background: red;}/*Especificidad de 101 puntos*/
p.parrafo#id-parrafo{background:green;}/*Especificidad de 111 puntos*/
</style>
```

El siguiente código HTML aplica estos estilos:

```
<body>
<p>El fondo de este párrafo será color carmesí</p>
<p class=”parrafo”>El fondo de este párrafo será color granate</p>
<div class=”parrafo”>Este div tendrá el fondo de color rosa</div>
<p id=”id-parrafo” class=”parrafo”>El fondo de este párrafo será de color verde</p>
<p id=”id-parrafo” style=”background: black;”>El fondo de este párrafo será negro porque
usa un atributo de estilo (con peso 1000) dentro de la etiqueta</p>
<p id=”id-parrafo”>El fondo de este párrafo será rojo</p>
</body>
```

Además de la especificidad, se de obligar a que un atributo de una regla se aplique por encima del resto de reglas. Esto se hace usando la declaración *!important*. Para utilizar *!important* en una regla de estilo, siempre se coloca en la parte del valor del atributo, antes del punto y coma ";". Por ejemplo:

```
p{
background: red !important;
background: crimson ;
}
```

Tenemos una declaración de estilos para los párrafos <p>, donde definimos dos veces el atributo *background*. En condiciones normales, se tendría en cuenta el valor definido en segundo lugar. Sin embargo, al estar el primer *background* definido como *!important* en realidad lo que ocurrirá es que prevalezca este atributo sobre el otro y se aplique un color de fondo rojo sobre todos los párrafos.

Si se sustituye esta declaración de p vista en el ejemplo anterior (*p{background: crimson;}*) por esta otra (*p{background: red !important; background: crimson ;}*) el resultado será que todos los párrafos tendrán color de fondo rojo (el *div* seguirá siendo rosa porque no es una párrafo y no se aplica sobre él esta regla).

2.8 CREAR Y VINCULAR HOJAS DE ESTILO

En este punto se describen las alternativas existentes para asociar un código HTML (contenido) con un estilo determinado y definido en CSS. Existen tres alternativas principales, y todas ellas han sido mostradas en los diferentes ejemplos del capítulo.

La primera alternativa es usando el atributo *style* dentro de las etiquetas HTML (reglas de estilos integradas). Por ejemplo:

```
<p style="background: black;">El fondo de este párrafo será negro </p>
```

Esta alternativa presenta alta prioridad tiene frente a otras reglas. Sin embargo esta aparente ventaja se convierte en desventaja cuando, al usarla, se pierde la posibilidad de reutilizar reglas entre elementos, teniendo que escribir todos los atributos de nuevo cuando se desea aplicar el mismo estilo a otros elementos. Además, otra desventaja añadida es que se dificulta el mantenimiento de las CSS haciendo más complicados los cambios y la localización de errores.

Otra alternativa es usando la etiqueta <style> dentro del mismo fichero (reglas de estilo incrustadas). En este caso la etiqueta <style> se coloca en la cabecera <head> del documento. La etiqueta <style> tiene varios atributos opcionales:

- **Type** (requerido): se usa para indicar que se aplica un estilo formato CSS.
- **Media**: atributo para indicar sobre qué dispositivo se aplicarán los estilos. Algunos de los valores posibles son: *handheld* (para dispositivos móviles), *print* (para salida por impresora), *projection* (para presentación en proyectores), *screen* (para pantallas), tv (para televisores), *braille* (para presentación en dispositivos braille) u *all* (para todos los dispositivos). Si se desean especificar varios valores se puede hacer separados por comas (,).
- **Title**: nombre que se le da al estilo. Útil cuando se vinculan CSS externas.

Un ejemplo de declaración es el siguiente:

```
<style type="text/css" media="screen,tv" title="Mi Estilo 1"  >
```

En la mayor parte de los ejemplos utilizados en este capítulo se ha sido la etiqueta <style> aplicada a un único HTML. Este método tiene sentido cuando un único documento tenga un único estilo. Si la misma hoja de estilo se usa en múltiples documentos o páginas web, entonces sería más apropiado disponer de una hoja de estilo externa tal y como se verá en la siguiente sección.

2.9 CREAR Y VINCULAR HOJAS DE ESTILO EN CASCADA EXTERNA

La ventaja de utilizar hojas de estilo externas (CSS externas) es que se pueden reutilizar en varios documentos HTML, permitiendo así, por ejemplo, que todo un sitio web se rija por las misma CSS. De hecho, esta alternativa es la más empleada profesionalmente.

Una hoja de estilo externa puede ser enlazada a un documento HTML mediante la etiqueta <link> que se coloca en el <head> de la página. El siguiente ejemplo muestra cuatro enlaces a hojas CSS. Los atributos *type* y media tienen la misma semántica que en la etiqueta <style> vista en la sección 2.8:

```
<link rel=stylesheet href="estilo.css" type="text/css" media=screen>
<link rel=stylesheet href="color-8b.css" type="text/css" title="estilo de color 8-bit"
media="screen, print">
<link rel="alternate stylesheet" href="color-24b.css" type="text/css" title="estilo de
color 24-bit" media="screen, print">
<link rel=stylesheet href="aural.css" type="text/css" media=screen>
```

Cuando se define una CSS externa ésta no debe contener ninguna etiqueta HTML como <head> o <style>. La hoja de estilo solo debería consistir de reglas de estilo o sentencias. Un archivo que solo consista de la siguiente línea podría utilizarse como hoja de estilo externa.

```
p { margin: 2em }
```

El atributo *rel* se usa para definir la relación entre el archivo enlazado y el documento HTML. *rel=stylesheet* especifica un estilo persistente o preferido. Un estilo persistente es aquel que siempre se aplica si están activas las hojas de estilo. Un estilo preferido es uno que se aplica automáticamente, como en la segunda etiqueta <link> en el ejemplo. La combinación de *rel=stylesheet* y un atributo *title* especifica un estilo preferido. Los autores no pueden especificar más de un estilo preferido.

Por otro lado *rel="alternate stylesheet"* define un estilo alternativo. Un estilo alterno se indica por *rel="alternate stylesheet"*. La tercera etiqueta <link> en el ejemplo define un estilo alternativo, que el usuario podría elegir para reemplazar la hoja de estilo preferido. Debe tenerse en cuenta que algunos navegadores carecen de la capacidad de elegir estilos alternativos.

Un estilo simple también puede ser dado mediante múltiples hojas de estilo. Todas ellas contribuyen a dar el estilo al HTML que las importa:

```
<link rel=stylesheet href="basico.css" title="miestilo">
<link rel=stylesheet href="tablas.css" title="miestilo">
<link rel=stylesheet href="formas.css" title="miestilo">
```

En este ejemplo, tres hojas de estilo son combinadas en un estilo llamado "miestilo" que se aplica como una hoja de estilo preferido. Para combinar múltiples hojas de estilo en un estilo único, se debe usar el mismo *title* con cada hoja de estilo.

Como se ha comentado al principio, una hoja de estilo externa es ideal cuando el estilo se aplica a muchas páginas. Un autor podrá cambiar la apariencia de un sitio completo mediante el cambio de un solo archivo. Además, la mayoría de navegadores guardan en caché las hojas de estilo externas, evitando así una demora en la presentación una vez que la hoja de estilo se ha guardado en caché.

Otra alternativa para enlazar CSS externos es usar la regla *@import*. Esta regla va incluida dentro de las etiquetas <style>. Un ejemplo de de uso de esta regla es el siguiente:

```
<style>@import url("estilos.css");</style>
```

El funcionamiento es igual que usar <link>. La diferencia es que *@import* no es soportada por todos los navegadores y, además, <link> ofrece mejores alternativas si se desea usar hojas de estilo preferentes, alternativas o preferidas. Aún así, *@import* permite hacer ciertas cosas como elegir cuál importar dependiendo del medio (media) al que se vaya a aplicar.

```
<style>
@import url("impresora.css")print;
@import url("normal.css")screen;
</style>
```

ACTIVIDADES 2.10

- Cree un fichero de texto con extensión "css" que contenga el interior de las etiquetas <style> de la Actividad 2.7. Guárdelo con el nombre "miestilo.css".
- Cree otro fichero HTML llamado "micontenido.html" que tenga el código HTML restante de la Actividad 2.7. En ese HTML importa los estilos de "miestilo.css": Primero usando la regla *@import* y otra con la etiqueta <link>.

2.10 HERRAMIENTAS Y TEST DE VERIFICACIÓN

Como se comentó al principio del capítulo CSS es un estándar de W3C. Para que los desarrolladores puedan comprobar que los estilos que definen cumplen ese estándar, el consorcio de estándares web W3C (*World Wide Web Consortium*) proporciona herramientas para validar tanto el código HTML como las hojas de estilo CSS, comprobando si éstas son correctas según las gramáticas publicadas.

A este servicio de validación se puede acceder a través de la página web: *http://jigsaw.w3.org/css-validator/*. Su uso es muy sencillo, el usuario solo tiene que introducir la URL del sitio web que pretenda evaluar y seleccionar las opciones que desea considerar.

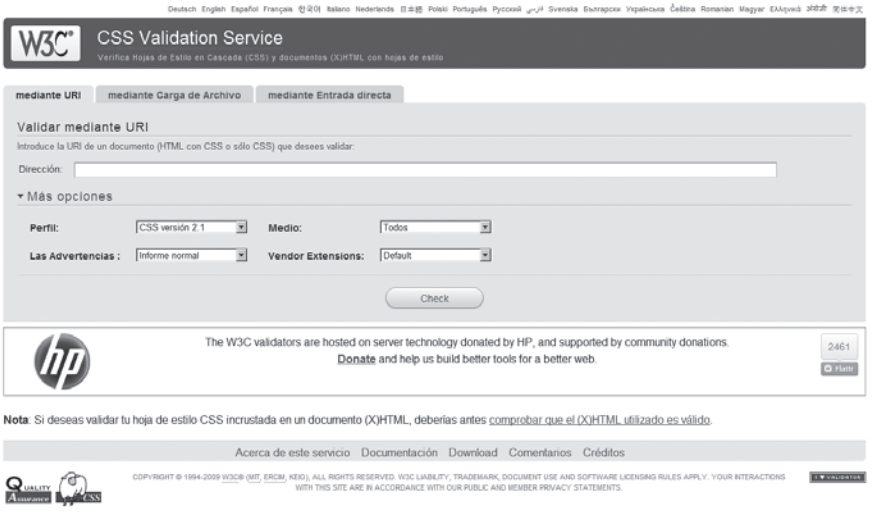

Figura 2.20. *Portal de la W3C donde se ofrece un servicio de validación de CSS*

Si el proceso de validación no encuentra errores, sus autores pueden incluir en la página web un icono como el siguiente. Con la inclusión de esta imagen los visitantes verán que los desarrolladores se han preocupado por crear un sitio web interoperable y acorde al estándar.

Además de la W3C las principales herramientas de desarrollo de sitios web también ofrecen facilidades para el desarrollo y validación de hojas de estilo.

Otras herramientas de validación que pueden utilizarse para comprobar nuestras hojas de estilo podemos encontrarlas ligadas a determinados navegadores; por ejemplo *Firefox* ofrece un complemento que puede instalarse y permite validar un sitio web. Dicho complemento puede descargarse en *https://addons.mozilla.org/es-es/firefox/addon/css-validator/*.

XHTML-CSS (*http://xhtml-css.com/*) es otra herramienta de validación que puede utilizarse indistintamente para comprobar la bondad (lo adecuado) de nuestro código XHTML y el CSS. Los informes ofrecidos por estas herramientas, en muchas ocasiones, están relacionados con distintas situaciones que deben ser comprobadas por el desarrollador o diseñador del sitio web, pero que no necesariamente son errores que haya que subsanar obligatoriamente. Dichas situaciones son identificadas como *warnings*, es decir, se trata de código cuyo bondad o no debe ser comprobada en última instancia por un humano, ya que automáticamente no hay evidencias totalmente objetivas y automáticas que permitan afirmar que hay una vulneración en el uso y construcción de la hoja de estilo.

Además de herramientas de validación, también hay otras ligadas a los navegadores (*plugins*) que permiten examinar CSS y detectar errores y editar código al instante. Algunos ejemplos son:

- *Firebug* (*http://getfirebug.com/*): es un *plugin* para Firefox y Chrome (Firebug Lite) entre otros navegadores, que permite, por ejemplo, analizar CSS en cascada, editar "en vivo" reglas y atributos (propiedades) y autocompletar valores de atributos con sugerencias de contexto. Esta herramienta no es solo para ayudar en el desarrollo de CSS, sino que es un asistente para todo lo que conlleva el desarrollo web. Por ejemplo, permite inspeccionar códigos HTML y Javascript.
- *Pendule* (*https://chrome.google.com/webstore/detail/gbkffbkamcejhkcaocmkdeiiccpmjfdi*): es un *plugin* de Chrome que está más especializado en CSS, permitiendo validar CSS pero también visualizar reglas, deshabilitarlas, mostrar los colores usados, etc. Es una herramienta simple para comprobar problemas en un sitio web (depurar).

Por otro lado, en Internet hay una gran cantidad de herramientas de uso gratutito que permiten generar (y ayudar a generar) código CSS automáticamente. Algunos ejemplos son:

- *List-o-matic* (*http://www.accessify.com/tools-and-wizards/developer-tools/list-o-matic/*): permite crear código CSS para menús. Se indica el texto que aparecerá en los enlaces, se selecciona el diseño preferido y la herramienta crear el CSS que lo representa, para que el desarrollador lo copie y incorpore en su web.
- *CSS Layout Generator* (*http://www.pagecolumn.com/*): permite crear CSS para páginas web con diferentes capas, distribuidas de la manera que el usuario desee (varias columnas, horizontales, verticales, etc.). Una vez seleccionado el diseño preferido, la herramienta crear el código para copiar y pegar. Se puede elegir solo el CSS o también el HTML que lo usa.
- *CSS Text Wrapper* (*http://www.csstextwrap.com/*): es una herramienta que permite colocar mediante CSS un texto dentro de la forma que se desee. Lo normal es colocar los textos en rectángulos. Pero esta herramienta genera código para colocarlo según la forma (circular, trapezoidal, triangular, etc.) que el usuario elija visualmente. Es una utilidad muy adecuada para hacer textos aparentes en un web.

Por último, la gran cantidad de sitios web que ofrecen de manera gratuita o de pago plantillas CSS y HTML ya creados son una gran ayuda para la creación de sitios web. Los desarrolladores, ante la idea de empezar un sitio web desde cero, pueden optar por usar una plantilla ya creada y modificarla para que obtenga toda la funcionalidad que un cliente quiera (personalización).

- ✓ *FreeCSSTemplates* (*http://www.freecsstemplates.org/*) es uno de los sitios web que ofrecen plantillas CSS3-HTML5 gratuitas. Un desarrollador puede elegir de este sitio la plantilla que más se asemeja a las especificaciones del cliente, descargarla, y modificarla para conseguir un sitio personalizado.

Los vistos son solo algunos ejemplos de herramientas gratuitas disponibles en Internet para ayudar en la creación de CSS. Conforme pasa el tiempo, la aparición de estas herramientas se incrementa, facilitando mucho a los diseñadores la generación de código, ahorrando tiempo de desarrollo, al mismo tiempo que se crean sitios aparentes y actuales.

2.11 CONCLUSIÓN Y PROPUESTAS PARA AMPLIAR

En este capítulo se han mostrado las características más destacables de CSS. Si lugar a duda, se podría concluir que los visto en las secciones 2.2, 2.5y 2.9 es la esencia de CSS. Comprender bien las actividades propuestas y los ejemplos de esas secciones ayuda a que el diseñador pueda controlar perfectamente todos los elementos incluidos en su web: desde el punto de vista del tamaño, posición y aspecto.

Terminado el capítulo, el lector debe entender que no ha sido posible sintetizar *todo CSS* en estas páginas. CSS tiene muchos aspectos avanzados que no han sido comentados, se han quedado en el tintero. Sin embargo, para facilitar que el lector profundice en algunos de esos aspectos no vistos, a continuación se enumeran los más destacados con el fin de facilitar al lector la búsqueda de información relativa a ellos:

- Selectores universales, que son aplicables a todas las etiquetas de un HTML.
- Atributos específicos de CSS3.
- Selectores aplicables solo a etiquetas HTML que tengan un determinado atributo. Por ejemplo:
  - Si se quiere definir un selector que solo se aplique a etiquetas <img> que tengan definidos un atributo *alt*, se podría: img [alt] { estilo que se quiera }.
  - Si se quiere afinar más y que se aplique a <img> con atributo *alt* que tenga un valor determinado (*vacaciones*) entonces sería: *img* [alt=”vacaciones”] { estilo que se quiera }.
- Nomenclatura *de facto* para las partes más reconocibles de una web. Por ejemplo, a las reglas que definen la cabecera se les llama *header*, a los pies de página *footer*, etc.

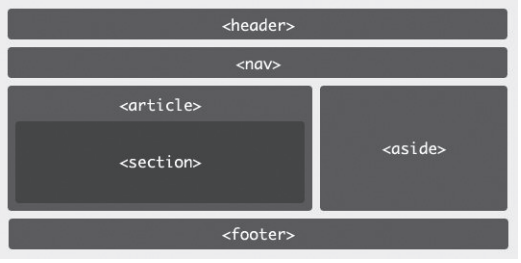

Figura 2.21. *Estructura con nomenclatura "de facto"*

- Encabezado: *header*; Pie:*footer*; Zona de navegación: *nav*; Secciones de la página: *section*; Artículos: *article*; Barra lateral: sidebar.
- Ya que esta estructura está muy extendida, HTML5 tiene etiquetas específicas para definirlas, simplificando así el trabajo con CSS: <header> <footer> <nav> <article> <section> y <aside> (barra lateral). La Figura 2.21 muestra este esquema.

- Estilos de autor y de lector. Diferentes ámbitos de personalización.

RESUMEN DEL CAPÍTULO

En este capítulo se ha hecho una introducción al manejo y funcionalidad básica de CSS. Se han mostrado todos los elementos principales que permiten al diseñador explotar sus características desde una buena base teórica. Se ha hecho especial hincapié en el modelo de cajas y el posicionamiento de las capas usando CSS.

Con lo visto en el capítulo se deduce que, hoy en día, en el diseño web con CSS, los desarrolladores no tienen que empezar desde cero. Hay muchas herramientas, libres y de pago, que ayudan a la generación automática de código CSS a partir de modelos visuales. Además, también hay plantillas creadas por otros autores que pueden servir como base para la creación de diseños personalizados.

Todas las herramientas y posibilidades de CSS no sirven para nada si los desarrolladores no adquieren desde el principio "buenas prácticas" a la hora de generar su propio código CSS, ya que esa es la base para hacer diseños fáciles de mantener, compartir y de reutilizar entre un proyecto y otro.

EJERCICIOS PROPUESTOS

1. Busque en Internet al menos cuatro herramientas gratuitas que permitan generar automáticamente código CSS a partir de diseños visuales. Éstas deben ser diferentes a las mostradas en este capítulo.

2. En parejas, desarrollad una hoja de estilos CSS para un sitio web de un grupo musical. Cada miembro debe utilizar individualmente, por lo menos, dos herramientas que generen código CSS automáticamente (de las encontradas en el ejercicio anterior o de las mostradas en el capítulo). La unificación de los códigos generados por cada miembro debe hacerse conjuntamente para obtener el resultado final.

3. En parejas, utilizad una herramienta libre para verificar la bondad del código CSS realizado e interpretar sus resultados. Además, comprobar que el resultado de la plantilla es el mismo en los principales navegadores: IEXplorer, Firefox, Chrome y Safari.

4. Busque en Internet algún sitio web, diferente a los mostrados en este capítulo, que ofrezcan plantillas CSS3 de pago. Seleccione al menos cuatro plantillas de ese sitio que te llamen la atención. ¿Enumere las características técnicas más importantes que destacaría de las plantillas seleccionadas? Reflexione sobre qué es lo que las hace atractivas para un diseñador como para pagar por ellas.

5. En parejas, descargad una plantilla basada en CSS disponible en Internet. Se pueden usar los enlaces mostrados en el capítulo o cualquier otro que los miembros conozcan. Modificad esa plantilla y personalizadla para que sea una buena propuesta para una web de un restaurante de comida tradicional. Se valorará el número de cambios de la plantilla final con respecto a la original, así como las diferencias visuales entre ambas.

TEST DE CONOCIMIENTOS

1 Se considera buena práctica para nombrar a los selectores:
- a) Que el nombre describa una característica visual como el color.
- b) Que el nombre no esté asociado a la localización de un elemento (salvo que se ofrezcan otros selectores con otras alternativas de localización).
- c) Que el nombre solo contenga minúsculas y no empiece por un carácter especial.

2 El *padding* en el modelo de cajas:
- a) Es la separación entre el borde de esa caja y otras cajas adyacentes.
- b) Es la separación entre el borde de esa caja y los elementos que hay en su interior.
- c) Se puede decir que es el ancho y alto de la caja.

3 Si se desea poner un elemento que no se oculte moviendo las barras de desplazamiento:
- **a)** Se puede definir como *position:absolute*.
- **b)** Se puede definir como *position:fixed*.
- **c)** Se puede definir como *relative:no_movement*.

4 En la definición *p{background: red; background: crimson ;}*:
- **a)** Los párrafos aparecerán con color fondo *crimson* por estar en segundo lugar.
- **b)** Los párrafos aparecerán con color *red* por estar en primer lugar.
- **c)** CSS dará un error por no repetir el *background* y no usar *important* en alguno de ellos.

5 A la hora de trabajar con CSS es un inconveniente:
- **a)** Que no haya plantillas en Internet ya hechas y disponibles para bajar y modificarlas.
- **b)** Que no haya herramientas visuales que permitan crear CSS automáticamente.
- **c)** Que cuando el código CSS es muy extenso es complicado de gestionar si no se lleva un orden adecuado y se respetan las buenas prácticas para su escritura.

6 Respecto a la posición de la cajas es falso que:
- **a)** *Clear* elimina el efecto del *float*.
- **b)** Los elementos de una página se posicionan por defecto como *static*.
- **c)** Los valores *top*, *left*, *bottom* y *right* son una distancia con respecto al primer elemento contenedor que tenga un valor de *position* distinto de *absolute*.