

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 (IEexplorer, 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 muestran 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* los 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 *background-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):

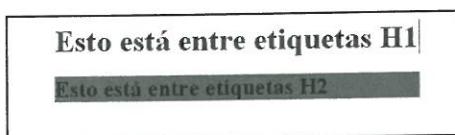


Figura 2.1. Mi primer CSS

## 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 `<p></p>` 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 estilos.

```
<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).

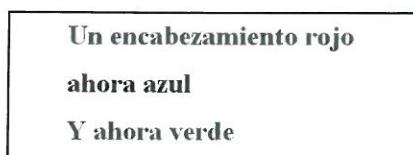


Figura 2.2. Ejemplo clases basadas en etiquetas

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.

```
.nombrecalse {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 `<div>`:

```
<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>
```

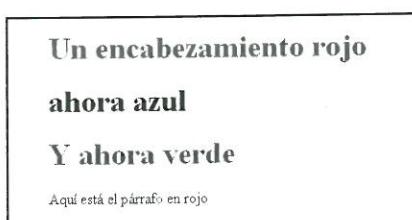
Y aplicando las clases a las diferentes etiquetas `<h1>`:

```
<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 aplicar 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**

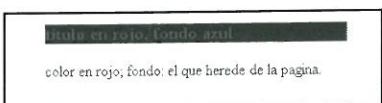
Una característica muy interesante del uso de clases es que se puede asignar más de una clase a una misma etiqueta, siempre y cuando no hay conflicto entre ellas. Por ejemplo, con la siguiente definición:

```
<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 <p></p>. Luego incluya en el <head> estilos con clases abstractas que muestren: el texto en <h1> en verde con fondo azul y el texto de <p> 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 *.textoaazulfondoverde*

### 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 <p class="roja"></p>. 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 identificadores 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>:

```
<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, los 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 `<div>` 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 `<div>` 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 `<div>`. 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 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).

*Un texto h1 en negrita, cursiva y rojo*

Un texto en negrita y rojo

Figura 2.5. Ejemplo sin anidamiento

*Un texto h1 en negrita, cursiva y rojo*

Un texto en negrita y rojo

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>
```

***Un texto h1 en negrita, cursiva y negro***

**Un texto en h1, negrita y rojo**

*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>
```

***Un texto h1 en negrita, cursiva y negro***

**Texto en h1, negrita, cursiva, rojo y subrayado**

*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

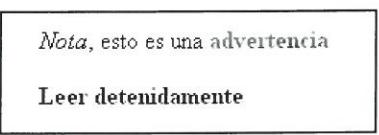
Este tipo de anidamiento se usa cuando se quiere aplicar un estilo a un elemento que tiene adyacente (al lado) a otro elemento en el mismo nivel de anidamiento en HTML. La sintaxis es la siguiente (para dos selectores):

*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):



*Nota, esto es una advertencia*  
**Leer detenidamente**

*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 `<div>` 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:groove}  
[...]
```

- **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.