1 Capítulo 1. Introducción

1.1 ¿Qué es CSS?

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "*documentos semánticos*"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para *marcar* los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

1.2 Cómo incluir CSS en un documento HTML

Una de las principales características de CSS es su flexibilidad y las diferentes opciones que ofrece para realizar una misma tarea. De hecho, existen tres opciones para incluir CSS en un documento HTML.

1.2.1 Incluir CSS en el mismo documento HTML

Los estilos se definen en una zona específica del propio documento HTML. Se emplea la etiqueta `<style>` de HTML y solamente se pueden incluir en la cabecera del documento (sólo dentro de la sección `<head>`).

Ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Ejemplo de estilos CSS en el propio documento</title>
      <style type="text/css">
            p { color: black; font-family: Verdana; }
      </style>
```

```
</head>
<body>
    <p>Un párrafo de texto.</p>
</body>
</html>
```

Este método se emplea cuando se define un número pequeño de estilos o cuando se quieren incluir estilos específicos en una determinada página HTML que completen los estilos que se incluyen por defecto en todas las páginas del sitio web.

El principal inconveniente es que, si se quiere hacer una modificación en los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que se va a modificar.

Los ejemplos mostrados en este libro utilizan este método para aplicar CSS al contenido HTML de las páginas. De esta forma el código de los ejemplos es más conciso y se aprovecha mejor el espacio.

1.2.2 Definir CSS en un archivo externo

En este caso, todos los estilos CSS se incluyen en un archivo de tipo CSS que las páginas HTML enlazan mediante la etiqueta `<link>`. Un archivo de tipo CSS no es más que un archivo simple de texto cuya extensión es `.css` Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite.

Si se quieren incluir los estilos del ejemplo anterior en un archivo CSS externo, se deben seguir los siguientes pasos:

1) Se crea un archivo de texto y se le añade solamente el siguiente contenido:

```
p { color: black; font-family: Verdana; }
```

2) Se guarda el archivo de texto con el nombre `estilos.css` Se debe poner especial atención a que el archivo tenga extensión `.css` y no `.txt`

3) En la página HTML se enlaza el archivo CSS externo mediante la etiqueta `<link>`:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Ejemplo de estilos CSS en un archivo externo</title>
    <link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />
</head>
<body>
    <p>Un párrafo de texto.</p>
```

```
</body>

</html>
```

Cuando el navegador carga la página HTML anterior, antes de mostrar sus contenidos también descarga los archivos CSS externos enlazados mediante la etiqueta `<link>` y aplica los estilos a los contenidos de la página.

Normalmente, la etiqueta `<link>` incluye cuatro atributos cuando enlaza un archivo CSS:

- `rel`: indica el tipo de relación que existe entre el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor `stylesheet`
- `type`: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es `text/css`
- `href`: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.
- `media`: indica el medio en el que se van a aplicar los estilos del archivo CSS. Más adelante se explican en detalle los medios CSS y su funcionamiento.

De todas las formas de incluir CSS en las páginas HTML, esta **es la más utilizada con mucha diferencia**. La principal ventaja es que se puede incluir un mismo archivo CSS en multitud de páginas HTML, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

Con este método, el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo.

Aunque generalmente se emplea la etiqueta `<link>` para enlazar los archivos CSS externos, también se puede utilizar la etiqueta `<style>`. La forma alternativa de incluir un archivo CSS externo se muestra a continuación:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Ejemplo de estilos CSS en un archivo externo</title>
      <style type="text/css" media="screen">
            @import '/css/estilos.css';
      </style>
</head>
 <body>
      <p>Un párrafo de texto.</p>
</body>
```

```
</html>
```

En este caso, para incluir en la página HTML los estilos definidos en archivos CSS externos se utiliza una regla especial de tipo @import. Las reglas de tipo @import siempre preceden a cualquier otra regla CSS (con la única excepción de la regla @charset).

La URL del archivo CSS externo se indica mediante una cadena de texto encerrada con comillas simples o dobles o mediante la palabra reservada url(). De esta forma, las siguientes reglas @import son equivalentes:

```
@import '/css/estilos.css';

@import "/css/estilos.css";

@import url('/css/estilos.css');

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

El inconveniente de este modo de insertar hojas de estilos frente a la anterior es que algunos navegadores no cargan la página hasta que no está cargado completamente el fichero css.

1.2.3 Incluir CSS en los elementos HTML

El último método para incluir estilos CSS en documentos HTML es el peor y el menos utilizado, ya que tiene los mismos problemas que la utilización de las etiquetas `<font>`.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Ejemplo de estilos CSS en el propio documento</title>
</head>
 <body>
      <p style="color: black; font-family: Verdana;">Un párrafo de texto.</p>
</body>
</html>
```

Esta forma de incluir CSS directamente en los elementos HTML solamente se utiliza en determinadas situaciones en las que se debe incluir un estilo muy específico para un solo elemento concreto.

1.3 Glosario básico

CSS define una serie de términos que permiten describir cada una de las partes que componen los estilos CSS. El siguiente esquema muestra las partes que forman un estilo CSS muy básico:

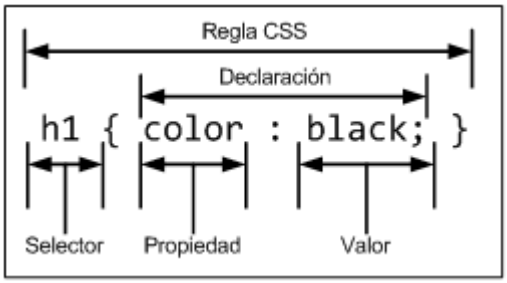

Figura 1.1 Componentes de un estilo CSS básico

Los diferentes términos se definen a continuación:

- **Regla**: cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de *"selectores"*, un símbolo de *"llave de apertura"* ({), otra parte denominada *"declaración"* y por último, un símbolo de *"llave de cierre"* (}).
- **Selector**: indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Declaración**: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
    - **Propiedad**: característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
    - **Valor**: establece el nuevo valor de la característica modificada en el elemento.

Un archivo CSS puede contener un número ilimitado de reglas CSS, cada regla se puede aplicar a varios selectores diferentes y cada declaración puede incluir tantos pares propiedad/valor como se desee.

El estándar CSS 2.1 define 115 propiedades, cada una con su propia lista de valores permitidos. Por su parte, los últimos borradores del estándar CSS 3 ya incluyen 239 propiedades.

1.4 Medios CSS

Una de las características más importantes de las hojas de estilos CSS es que permiten definir diferentes estilos para diferentes medios o dispositivos: pantallas, impresoras, móviles, proyectores, etc.

Además, CSS define algunas propiedades específicamente para determinados medios, como por ejemplo la paginación y los saltos de página para los medios impresos o el volumen y tipo de voz para los medios de audio. La siguiente tabla muestra el nombre que CSS utiliza para identificar cada medio y su descripción:

Medio	Descripción
`all`	Todos los medios definidos
`braille`	Dispositivos táctiles que emplean el sistema braille

Medio	Descripción
`embosed`	Impresoras braille
`handheld`	Dispositivos de mano: móviles, PDA, etc.
`print`	Impresoras y navegadores en el modo *"Vista Previa para Imprimir"*
`projection`	Proyectores y dispositivos para presentaciones
`screen`	Pantallas de ordenador
`speech`	Sintetizadores para navegadores de voz utilizados por personas discapacitadas
`tty`	Dispositivos textuales limitados como teletipos y terminales de texto
`tv`	Televisores y dispositivos con resolución baja

Los medios más utilizados actualmente son screen (para definir el aspecto de la página en pantalla) y print (para definir el aspecto de la página cuando se imprime), seguidos de handheld (que define el aspecto de la página cuando se visualiza mediante un dispositivo móvil).

Además, CSS clasifica a los medios en diferentes grupos según sus características. La siguiente tabla resume todos los grupos definidos en el estándar:

Medio	Continuo / Paginado	Visual / Auditivo / Táctil / Vocal	Mapa de bits / Caracteres	Interactivo / Estático
`braille`	continuo	táctil	caracteres	ambos
`embossed`	paginado	táctil	caracteres	estático
`handheld`	ambos	visual, auditivo, vocal	ambos	ambos
`print`	paginado	visual	mapa de bits	estático
`projection`	paginado	visual	mapa de bits	interactivo
`screen`	continuo	visual, auditivo	mapa de bits	ambos
`speech`	continuo	vocal	(no tiene sentido)	ambos

Medio	Continuo / Paginado	Visual / Auditivo / Táctil / Vocal	Mapa de bits / Caracteres	Interactivo / Estático
`tty`	continuo	visual	caracteres	ambos
`tv`	ambos	visual, auditivo	mapa de bits	ambos

La gran ventaja de CSS es que permite modificar los estilos de una página en función del medio en el que se visualiza. Existen distintas formas de indicar el medio en el que se deben aplicar los estilos CSS.

1.4.1 Medios definidos con las reglas de tipo @media

Las reglas @media son un tipo especial de regla CSS que permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla. Para especificar el medio en el que se aplican los estilos, se incluye su nombre después de @media. Si los estilos se aplican a varios medios, se incluyen los nombres de todos los medios separados por comas.

A continuación se muestra un ejemplo sencillo:

```
@media print {
  body { font-size: 10pt }
}
@media screen {
  body { font-size: 13px }
}
@media screen, print {
  body { line-height: 1.2 }
}
```

El ejemplo anterior establece que el tamaño de letra de la página cuando se visualiza en una pantalla debe ser 13 píxel. Sin embargo, cuando se imprimen los contenidos de la página, su tamaño de letra debe ser de 10 puntos. Por último, tanto cuando la página se visualiza en una pantalla como cuando se imprimen sus contenidos, el interlineado del texto debe ser de 1.2 veces el tamaño de letra del texto.

1.4.2 Medios definidos con las reglas de tipo @import

Cuando se utilizan reglas de tipo `@import` para enlazar archivos CSS externos, se puede especificar el medio en el que se aplican los estilos indicando el nombre del medio después de la URL del archivo CSS:

```
@import url("estilos_basicos.css") screen;
@import url("estilos_impresora.css") print;
```

Las reglas del ejemplo anterior establecen que cuando la página se visualiza por pantalla, se cargan los estilos definidos en el primer archivo CSS. Por otra parte, cuando la página se imprime, se tienen en cuenta los estilos que define el segundo archivo CSS.

Si los estilos del archivo CSS externo deben aplicarse en varios medios, se indican los nombres de todos los medios separados por comas. Si no se indica el medio en una regla de tipo `@import`, el navegador sobreentiende que el medio es `all`, es decir, que los estilos se aplican en todos los medios.

1.4.3 Medios definidos con la etiqueta <link>

Si se utiliza la etiqueta `<link>` para enlazar los archivos CSS externos, se puede utilizar el atributo `media`para indicar el medio o medios en los que se aplican los estilos de cada archivo:

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css" />

<link rel="stylesheet" type="text/css" media="print, handheld" href="especial.css" />
```

En este ejemplo, el primer archivo CSS se tiene en cuenta cuando la página se visualiza en la pantalla (`media="screen"`). Los estilos indicados en el segundo archivo CSS, se aplican al imprimir la página (`media="print"`) o al visualizarla en un dispositivo móvil (`media="handheld"`), como por ejemplo en un iPhone.

Si la etiqueta `<link>` no indica el medio CSS, se sobreentiende que los estilos se deben aplicar a todos los medios, por lo que es equivalente a indicar `media="all"`.

1.4.4 Medios definidos mezclando varios métodos

CSS también permite mezclar los tres métodos anteriores para indicar los medios en los que se aplica cada archivo CSS externo:

```
<link rel="stylesheet" type="text/css"  media="screen" href="basico.css" />

@import url("estilos_seccion.css") screen;

@media print {

 /* Estilos específicos para impresora */

}
```

Los estilos CSS que se aplican cuando se visualiza la página en una pantalla se obtienen mediante el recurso enlazado con la etiqueta `<link>` y mediante el archivo CSS externo incluido con la regla de tipo `@import`. Además, los estilos aplicados cuando se imprime la página se indican directamente en la página HTML mediante la regla de tipo `@media`.

1.5 Comentarios

CSS permite incluir comentarios entre sus reglas y estilos. Los comentarios son contenidos de texto que el diseñador incluye en el archivo CSS para su propia información y utilidad. Los navegadores ignoran por completo cualquier comentario de los archivos CSS, por lo que es común utilizarlos para estructurar de forma clara los archivos CSS complejos.

El comienzo de un comentario se indica mediante los caracteres `/*` y el final del comentario se indica mediante `*/`, tal y como se muestra en el siguiente ejemplo:

```
/* Este es un comentario en CSS */
```

Los comentarios pueden ocupar tantas líneas como sea necesario, pero no se puede incluir un comentario dentro de otro comentario:

```
/* Este es un

  comentario CSS de varias

  lineas */
```

Aunque los navegadores ignoran los comentarios, su contenido se envía junto con el resto de estilos, por lo que no se debe incluir en ellos ninguna información sensible o confidencial.

La sintaxis de los comentarios CSS es muy diferente a la de los comentarios HTML, por lo que no deben confundirse:

```
<!-- Este es un comentario en HTML -->


<!-- Este es un

   comentario HTML de varias

   lineas -->
```

1.6 Sintaxis de la definición de cada propiedad CSS

A lo largo del documento, se incluyen las definiciones formales de la mayoría de propiedades de CSS. La definición formal se basa en la información recogida en el estándar oficial y se muestra en forma de tabla.

Una de las principales informaciones de cada definición es la lista de posibles valores que admite la propiedad. Para definir la lista de valores permitidos se sigue un formato que es necesario detallar.

Si el valor permitido se indica como una sucesión de palabras sin ningún carácter que las separe (paréntesis, comas, barras, etc.) el valor de la propiedad se debe indicar tal y como se muestra y con esas palabras en el mismo orden.

Si el valor permitido se indica como una sucesión de valores separados por una barra simple (carácter `|`) el valor de la propiedad debe tomar uno y sólo uno de los valores indicados. Por ejemplo, la notación `<porcentaje> | <medida> | inherit` indica que la propiedad solamente puede tomar como valor la palabra reservada `inherit` o un porcentaje o una medida.

Si el valor permitido se indica como una sucesión de valores separados por una barra doble (símbolo `||`) el valor de la propiedad puede tomar uno o más valores de los indicados y en cualquier orden.

Por ejemplo, la notación `<color> || <estilo> || <medida>` indica que la propiedad puede tomar como valor cualquier combinación de los valores indicados y en cualquier orden. Se podría establecer un color y un estilo, solamente una medida o una

medida y un estilo. Además, el orden en el que se indican los valores es indiferente. Opcionalmente, se pueden utilizar paréntesis para agrupar diferentes valores.

Por último, en cada valor o agrupación de valores se puede indicar el tipo de valor: opcional, obligatorio, múltiple o restringido.

El carácter * indica que el valor ocurre cero o más veces; el carácter + indica que el valor ocurre una o más veces; el carácter ? indica que el valor es opcional y por último, el carácter {número_1, número_2}indica que el valor ocurre al menos tantas veces como el valor indicado en número_1 y como máximo tantas veces como el valor indicado en número_2.

Por ejemplo, el valor [<family-name> ,]* indica que el valor de tipo <family_name> seguido por una coma se puede incluir cero o más veces. El valor <url>? <color> significa que la URL es opcional y el color obligatorio y en el orden indicado. Por último, el valor [<medida> | thick | thin] {1,4} indica que se pueden escribir entre 1 y 4 veces un valor que sea o una medida o la palabra thick o la palabra thin.

No obstante, la mejor forma de entender la notación formal para las propiedades de CSS es observar la definición de cada propiedad y volver a esta sección siempre que sea necesario.

1.7 Buenas prácticas CSS

A continuación, se enumeras algunas recomendaciones a tener en cuenta al crear las reglas css.

- /* hay que poner comentarios */
- Los selectores se nombran en minúsculas, nunca empezando por caracteres especiales o numéricos. En las últimas versiones se puede utilizar en el nombre "-" no "_" ya que no todos los navegadores lo soportan.
- El nombre de los selectores debe ser específico y claro, para que tenga una mayor capacidad expresiva.
- El nombre de las clases e identificadores no debe describir una característica visual, como color, tamaño o posición.
- Los nombres deben seguir más una **visión semántica que estructural**. Para facilitar los cambios.
- Separa las palabras mediante guiones o mayúsculas.
- No hacer uso excesivo de clases.
- **Agrupar las reglas según su selecto**r siempre que sea posible. Agruparlas unas debajo de otras:

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

- Al principio de un CSS es aconsejable definir los selectores de etiquetas. Además de usar **comentarios** para dejar claro cuál es la parte que definen las clases y otros elementos.

```
/* Etiquetas html */
```

```
html {font-family:Arial, verdana, sans serif; font-size:13px;}
h1,h2,h3,h4,h4,h6,form,input,text-area{
border:0; padding:0; margin:0;
font-family:arial;}
                    /* FIN de Etiquetas HTML
```

- **Estructurar visualmente los atributos**. Si un elemento solo tiene 3 atributos se pueden poner en la misma línea. Pero si hay más se ponen en líneas diferentes sangrados con tabuladores.

2 Capítulo 2. Selectores

Para crear diseños web profesionales, es imprescindible conocer y dominar los selectores de CSS. Una regla de CSS está formada por una parte llamada "selector" y otra parte llamada "declaración".

La declaración indica *"qué hay que hacer"* y el selector indica *"a quién hay que hacérselo"*. Por lo tanto, los selectores son imprescindibles para aplicar de forma correcta los estilos CSS en una página.

A un mismo elemento HTML se le pueden aplicar varias reglas CSS y cada regla CSS puede aplicarse a un número ilimitado de elementos. En otras palabras, una misma regla puede aplicarse sobre varios selectores y un mismo selector se puede utilizar en varias reglas.

El estándar de CSS incluye una docena de tipos diferentes de selectores, que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.

No obstante, la mayoría de las páginas de los sitios web se pueden diseñar utilizando solamente los cinco selectores básicos.

2.1 Selectores básicos

2.1.1 Selector universal

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

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

El selector universal se indica mediante un asterisco (`*`). A pesar de su sencillez, no se utiliza habitualmente, ya que es difícil que un mismo estilo se pueda aplicar a todos los elementos de una página.

No obstante, sí que se suele combinar con otros selectores y además, forma parte de algunos *hacks* muy utilizados, como se verá más adelante.

2.1.2 Selector de tipo o etiqueta

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

```
p {
  ...
}
```

Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres `<` y `>`) correspondiente a los elementos que se quieren seleccionar.

El siguiente ejemplo aplica diferentes estilos a los titulares y a los párrafos de una página HTML:

```
h1 {
 color: red;
}

h2 {
 color: blue;
}

p {
 color: black;
}
```

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

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

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

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

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

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

h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h3 { font-size: 1.2em; }
```

2.1.3 Selector descendente

Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

El selector del siguiente ejemplo selecciona todos los elementos `<span>` de la página que se encuentren dentro de un elemento `<p>`:

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

Si el código HTML de la página es el siguiente:

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

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

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

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

```
p span { color: red; }

h1 span { color: blue; }
```

Con las reglas CSS anteriores:

Los elementos `<span>` que se encuentran dentro de un elemento `<p>` se muestran de color rojo.

Los elementos `<span>` que se encuentran dentro de un elemento `<h1>` se muestran de color azul.

El resto de elementos `<span>` de la página, se muestran con el color por defecto aplicado por el navegador.

La sintaxis formal del selector descendente se muestra a continuación:

```
selector1 selector2 selector3 ... selectorN
```

Los selectores descendentes siempre están formados por dos o más selectores separados entre sí por espacios en blanco. El último selector indica el elemento sobre el que se aplican los estilos y todos los selectores anteriores indican el lugar en el que se debe encontrar ese elemento.

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

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

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

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

```
/* El estilo se aplica a todos los elementos "p", "a", "span" y
"em" */

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


/* El estilo se aplica solo a los elementos "em" que se

   encuentran dentro de "p a span" */

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

Se puede restringir el alcance del selector descendente combinándolo con el selector universal. El siguiente ejemplo, muestra los dos enlaces de color rojo:

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

```
<p><a href="#">Enlace</a></p>

<p><span><a href="#">Enlace</a></span></p>
```

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

```
p * a { color: red; }
```

```
<p><a href="#">Enlace</a></p>

<p><span><a href="#">Enlace</a></span></p>
```

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

2.1.4 Selector de clase

Si se considera el siguiente código HTML de ejemplo:

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

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

Una de las soluciones más sencillas para aplicar estilos a un solo elemento de la página consiste en utilizar el atributo `class` de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar:

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

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

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

El selector `.destacado` se interpreta como *"cualquier elemento de la página cuyo atributo `class` sea igual a `destacado`"*, por lo que solamente el primer párrafo cumple esa condición.

Este tipo de selectores se llaman selectores de clase y son los más utilizados junto con los selectores de ID que se verán a continuación. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden utilizar el mismo valor en el atributo `class`:

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

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

A continuación, se muestra otro ejemplo de selectores de clase:

```
.aviso {
 padding: 0.5em;
 border: 1px solid #98be10;
 background: #f6feda;
}
.error {
 color: #930;
 font-weight: bold;
}
```

```
<span class="error">...</span>
```

```
<div class="aviso">...</div>
```

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

atributo `class="aviso"`, por lo que su estilo es el que definen las reglas CSS del selector `.aviso`.

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

```
<body>

    <p class="destacado">Lorem ipsum dolor sit amet...</p>

    <p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a>
    accumsan...</p>

    <p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>

</body>
```

¿Cómo es posible aplicar estilos solamente al párrafo cuyo atributo `class` sea igual a `destacado`? Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

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

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

De lo anterior se deduce que el atributo .destacado es equivalente a *.destacado, por lo que todos los diseñadores obvian el símbolo * al escribir un selector de clase normal.

No debe confundirse el selector de clase con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo class="aviso" */

p.aviso { ... }


/* Todos los elementos con atributo class="aviso" que estén dentro

   de cualquier elemento de tipo "p" */

p .aviso { ... }


/* Todos los elementos "p" de la página y todos los elementos con

   atributo class="aviso" de la página */

p, .aviso { ... }
```

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

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

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

```
.error { color: red; }

.destacado { font-size: 15px; }

.especial  { font-weight: bold; }
```

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

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

```
.error { color: red; }

.error.destacado { color: blue; }

.destacado { font-size: 15px; }

.especial  { font-weight: bold; }
```

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

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

2.1.5 Selectores de ID

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

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

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

```
#destacado { color: red; }
```

```
 <p>Primer párrafo</p>

<p id="destacado">Segundo párrafo</p>

<p>Tercer párrafo</p>
```

En el ejemplo anterior, el selector #destacado solamente selecciona el segundo párrafo (cuyo atributo id es igual a destacado).

La principal diferencia entre este tipo de selector y el selector de clase tiene que ver con HTML y no con CSS. Como se sabe, en una misma página, el valor del atributo `id` debe ser único, de forma que dos elementos diferentes no pueden tener el mismo valor de `id`. Sin embargo, el atributo `class` no es obligatorio que sea único, de forma que muchos elementos HTML diferentes pueden compartir el mismo valor para su atributo `class`.

De esta forma, la recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página y utilizar el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la página HTML.

Al igual que los selectores de clase, en este caso también se puede restringir el alcance del selector mediante la combinación con otros selectores. El siguiente ejemplo aplica la regla CSS solamente al elemento de tipo `<p>` que tenga un atributo `id` igual al indicado:

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

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

En este caso, algunas páginas pueden disponer de elementos con un atributo `id` igual a `aviso` y que no sean párrafos, por lo que la regla anterior no se aplica sobre esos elementos.

No debe confundirse el selector de ID con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo id="aviso" */
p#aviso { ... }


/* Todos los elementos con atributo id="aviso" que estén dentro
    de cualquier elemento de tipo "p" */
p #aviso { ... }


/* Todos los elementos "p" de la página y todos los elementos
con
    atributo id="aviso" de la página */
p, #aviso { ... }
```

2.1.6 Combinación de selectores básicos

CSS permite la combinación de uno o más tipos de selectores para restringir el alcance de las reglas CSS. A continuación se muestran algunos ejemplos habituales de combinación de selectores.

```
.aviso .especial { ... }
```

El anterior selector solamente selecciona aquellos elementos con un `class="especial"` que se encuentren dentro de cualquier elemento con un `class="aviso"`.

Si se modifica el anterior selector:

```
div.aviso span.especial { ... }
```

Ahora, el selector solamente selecciona aquellos elementos de tipo `<span>` con un atributo `class="especial"` que estén dentro de cualquier elemento de tipo `<div>` que tenga un atributo `class="aviso"`.

La combinación de selectores puede llegar a ser todo lo compleja que sea necesario:

```
ul#menuPrincipal li.destacado a#inicio { ... }
```

El anterior selector hace referencia al enlace con un atributo `id` igual a `inicio` que se encuentra dentro de un elemento de tipo `<li>` con un atributo `class` igual a `destacado`, que forma parte de una lista `<ul>`con un atributo `id` igual a `menuPrincipal`.

Práctica 1.Selectores básicos

2.2 Selectores avanzados

Utilizando solamente los selectores básicos de la sección anterior, es posible diseñar prácticamente cualquier página web. No obstante, CSS define otros selectores más avanzados que permiten simplificar las hojas de estilos.

Desafortunadamente, el navegador Internet Explorer 6 y sus versiones anteriores no soportaban este tipo de selectores avanzados, por lo que su uso no era común hasta hace poco tiempo. Si quieres consultar el soporte de los selectores en los distintos navegadores, puedes utilizar las siguientes referencias:

- Lista completa de los selectores que soporta cada versión de cada navegador
- Test online en el que puedes comprobar los selectores que soporta el navegador con el que realizas el test

2.2.1 Selector de hijos

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento. Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante **el "signo de mayor que" (>):**

```
p > span { color: blue; }
```

```
<p><span>Texto1</span></p>

<p><a href="#"><span>Texto2</span></a></p>
```

En el ejemplo anterior, el selector `p > span` se interpreta como *"cualquier elemento `<span>` que sea hijo directo de un elemento `<p>`"*, por lo que el primer elemento `<span>` cumple la condición del selector. Sin embargo, el segundo elemento `<span>` no la cumple porque es descendiente pero no es hijo directo de un elemento `<p>`.

El siguiente ejemplo muestra las diferencias entre el selector descendente y el selector de hijos:

```
p a { color: red; }

p > a { color: red; }

<p><a href="#">Enlace1</a></p>

<p><span><a href="#">Enlace2</a></span></p>
```

El primer selector es de tipo descendente y por tanto se aplica a todos los elementos `<a>` que se encuentran dentro de elementos `<p>`. En este caso, los estilos de este selector se aplican a los dos enlaces.

Por otra parte, el selector de hijos obliga a que el elemento `<a>` sea hijo directo de un elemento `<p>`. Por lo tanto, los estilos del selector `p > a` no se aplican al segundo enlace del ejemplo anterior.

2.2.2 Selector adyacente

El selector adyacente se emplea para seleccionar elementos que en el código HTML de la página se encuentran justo a continuación de otros elementos. Su sintaxis emplea el signo `+` para separar los dos elementos:

```
elemento1 + elemento2 { ... }
```

Si se considera el siguiente código HTML:

```
<body>

<h1>Titulo1</h1>

 <h2>Subtítulo</h2>

...

 <h2>Otro subtítulo</h2>

...

</body>
```

La página anterior dispone de dos elementos `<h2>`, pero sólo uno de ellos se encuentra inmediatamente después del elemento `<h1>`. Si se quiere aplicar diferentes colores en función de esta circunstancia, el selector adyacente es el más adecuado:

```
h2 { color: green; }

h1 + h2 { color: red }
```

Las reglas CSS anteriores hacen que todos los `<h2>` de la página se vean de color verde, salvo aquellos `<h2>` que se encuentran inmediatamente después de cualquier elemento `<h1>` y que se muestran de color rojo.

Técnicamente, los elementos que forman el selector adyacente deben cumplir las dos siguientes condiciones:

- `elemento1` y `elemento2` deben ser *elementos hermanos*, por lo que su elemento padre debe ser el mismo.

- `elemento2` debe aparecer inmediatamente después de `elemento1` en el código HTML de la página.

El siguiente ejemplo es muy útil para los textos que se muestran como libros:

```
p + p { text-indent: 1.5em; }
```

En muchos libros, suele ser habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. Con el selector `p + p`, se seleccionan todos los párrafos de la página que estén precedidos por otro párrafo, por lo que no se aplica al primer párrafo de la página.

2.2.3 Selector de atributos

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- `[nombre_atributo]`, selecciona los elementos que tienen establecido el atributo llamado `nombre_atributo`, independientemente de su valor.
- `[nombre_atributo=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` con un valor igual a `valor`.
- `[nombre_atributo^=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` y su valor comience por valor
- `[nombre_atributo$=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` y su valor finaliza por valor
- `[nombre_atributo~=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` y al menos uno de los valores del atributo es `valor`.

A continuación, se muestran algunos ejemplos de estos tipos de selectores:

```
/* Se muestran de color azul todos los enlaces que tengan
   un atributo "class", independientemente de su valor */
a[class] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
   un atributo "class" con el valor "externo" */
a[class="externo"] { color: blue; }

/* Se muestran de color azul todos los enlaces que apunten
   al sitio "http://www.ejemplo.com" */
a[href="http://www.ejemplo.com"] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
```

```
   un atributo "class" en el que al menos uno de sus valores
   sea "externo" */
a[class~="externo"] { color: blue; }

/* Selecciona todos los elementos de la página cuyo atributo
   "lang" sea igual a "en", es decir, todos los elementos en
inglés */
*[lang=en] { ... }
```

2.3 Agrupación de reglas

Cuando se crean archivos CSS complejos con decenas o cientos de reglas, es habitual que los estilos que se aplican a un mismo selector se definan en diferentes reglas:

```
h1 { color: red; }
...
h1 { font-size: 2em; }
...
h1 { font-family: Verdana; }
```

Las tres reglas anteriores establecen el valor de tres propiedades diferentes de los elementos `<h1>`. Antes de que el navegador muestre la página, procesa todas las reglas CSS de la página para tener en cuenta todos los estilos definidos para cada elemento.

Cuando el selector de dos o más reglas CSS es idéntico, se pueden agrupar las declaraciones de las reglas para hacer las hojas de estilos más eficientes:

```
h1 {
 color: red;
 font-size: 2em;
 font-family: Verdana;
}
```

El ejemplo anterior tiene el mismo efecto que las tres reglas anteriores, pero es más eficiente y es más fácil de modificar y mantener por parte de los diseñadores. Como CSS ignora los espacios en blanco y las nuevas líneas, también se pueden agrupar las reglas de la siguiente forma:

```
h1 { color: red; font-size: 2em; font-family: Verdana; }
```

Si se quiere reducir al máximo el tamaño del archivo CSS para mejorar ligeramente el tiempo de carga de la página web, también es posible indicar la regla anterior de la siguiente forma:

```
h1 {color:red;font-size:2em;font-family:Verdana;}
```

2.4 Herencia

Una de las características principales de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad. Si se considera el siguiente ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de herencia de estilos</title>
<style type="text/css">
 body { color: blue; }
</style>
</head>

<body>
 <h1>Titular de la página</h1>
 <p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

En el ejemplo anterior, el selector `body` solamente establece el color de la letra para el elemento `<body>`. No obstante, la propiedad `color` es una de las que se heredan de forma automática, por lo que todos los elementos descendientes de `<body>` muestran ese mismo color de letra. Por tanto, establecer el color de la letra en el elemento `<body>` de la página implica cambiar el color de letra de todos los elementos de la página.

Aunque la herencia de estilos se aplica automáticamente, se puede anular su efecto estableciendo de forma explícita otro valor para la propiedad que se hereda, como se muestra en el siguiente ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
     <meta http-equiv="Content-Type" content="text/html;
     charset=iso-8859-1" />
```

```
        <title>Ejemplo de herencia de estilos</title>
        <style type="text/css">
                body { font-family: Arial; color: black; }
                h1 { font-family: Verdana; }
                p { color: red; }
        </style>
</head>

<body>
        <h1>Titular de la página</h1>
        <p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

En el ejemplo anterior, se establece en primer lugar el color y tipo de letra del elemento `<body>`, por lo que todos los elementos de la página se mostrarían con ese mismo color y tipo de letra. No obstante, las otras reglas CSS modifican alguno de los estilos heredados.

De esta forma, los elementos `<h1>` de la página se muestran con el tipo de letra `Verdana` establecido por el selector `h1` y se muestran de color negro que es el valor heredado del elemento `<body>`. Igualmente, los elementos `<p>` de la página se muestran del color rojo establecido por el selector `p` y con un tipo de letra `Arial` heredado del elemento `<body>`.

La mayoría de propiedades CSS aplican la herencia de estilos de forma automática. Además, para aquellas propiedades que no se heredan automáticamente, CSS incluye un mecanismo para forzar a que se hereden sus valores, tal y como se verá más adelante.

Por último, aunque la herencia automática de estilos puede parecer complicada, simplifica en gran medida la creación de hojas de estilos complejas. Como se ha visto en los ejemplos anteriores, si se quiere establecer por ejemplo la tipografía base de la página, simplemente se debe establecer en el elemento `<body>` de la página y el resto de elementos la heredarán de forma automática.

2.5 Colisiones de estilos

En las hojas de estilos complejas, es habitual que varias reglas CSS se apliquen a un mismo elemento HTML. El problema de estas reglas múltiples es que se pueden dar colisiones como la del siguiente ejemplo:

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

```
<p>...</p>
```

¿De qué color se muestra el párrafo anterior? CSS tiene un mecanismo de resolución de colisiones muy complejo y que tiene en cuenta el tipo de hoja de estilo que se trate (de navegador, de usuario o de diseñador), la importancia de cada regla y lo específico que sea el selector.

El método seguido por CSS para resolver las colisiones de estilos se muestra a continuación:

1. Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
2. Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su prioridad (palabra clave `!important`).
3. Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
4. Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó en último lugar.

Hasta que no se expliquen más adelante los conceptos de tipo de hoja de estilo y la prioridad, el mecanismo simplificado que se puede aplicar es el siguiente:

1. Cuanto más específico sea un selector, más importancia tiene su regla asociada.
2. A igual *especificidad*, se considera la última regla indicada.

Como en el ejemplo anterior los dos selectores son idénticos, las dos reglas tienen la misma prioridad y prevalece la que se indicó en último lugar, por lo que el párrafo se muestra de color azul.

En el siguiente ejemplo, la regla CSS que prevalece se decide por lo específico que es cada selector:

```
p { color: red; }

p#especial { color: green; }

* { color: blue; }
```

```
<p id="especial">...</p>
```

Al elemento `<p>` se le aplican las tres declaraciones. Como su origen y su importancia es la misma, decide la especificidad del selector. El selector `*` es el menos específico, ya que se refiere a *"todos los elementos de la página"*. El selector `p` es poco específico porque se refiere a *"todos los párrafos de la página"*. Por último, el selector `p#especial` sólo hace referencia a *"el párrafo de la página cuyo atributo `id` sea igual a especial"*. Como el selector `p#especial` es el más específico, su declaración es la que se tiene en cuenta y por tanto el párrafo se muestra de color verde.

La mayor prioridad la tienen los atributos de estilo (style). 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, div).
- 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 (style) a los que se les da un valor de 1000 puntos.

2.6 Pseudo-clases y pseudoelementos

Una **pseudoclase** es un selector que marca los elementos que están en un estado específico, por ejemplo, los que son el primer elemento de su tipo, o aquellos por los que el cursor les pasa por encima. Tienden a actuar como si hubieras aplicado una clase en una parte determinada del documento y, a menudo, ayudan a reducir el exceso de clases y proporcionan un marcado más flexible y fácil de mantener.

Las pseudo-clases son palabras clave que comienzan con dos puntos:.

La pseudo-clase **:first-child** selecciona el primer elemento hijo de un elemento. Si se considera el siguiente ejemplo que selecciona el primer elemento que es hijo de un elemento que se encuentra dentro de un elemento <p>:

```
p em:first-child {

 color: red;

}
```

El siguiente ejemplo selecciona los elementos que se encuentran dentro del primer párrafo que sea el primer hijo de un elemento.

```
p:first-child em {

 color: red;

}
```

Es válido escribir pseudo-clases y pseudoelementos sin que les preceda un selector de elemento. En el ejemplo anterior, podría escribirse :first-child y la regla se aplicaría a cualquier elemento que sea el primer hijo de un elemento , **:first-child equivale a *:first-child.** Pero normalmente se quiere más control y hay que ser más específico.

Todas las pseudoclases se comportan del mismo modo.

- :last-child para seleccionar el último elemento.
- :only-child selecciona elementos sin hermanos.

Algunas pseudoclases solo intervienen cuando el usuario interactúa con el documento de alguna manera. Estas pseudoclases de **acción de usuario**, que también reciben el nombre de **pseudoclases dinámicas**, actúan como si se añadiese una clase al elemento cuando el usuario interactúa con él. Algunos ejemplos son:

- :**link** enlace no visitado
- :**visited** enlace ya visitado
- **:hover** solo interviene si el usuario pasa el cursor sobre un elemento, normalmente un enlace.
- :**active** se activa cuando el usuario activa un elemento, por ejemplo cuando pulsa con el ratón sobre un elemento. El estilo se aplica durante un espacio de tiempo prácticamente imperceptible, ya que sólo dura desde que el usuario pulsa el botón del ratón hasta que lo suelta.

link y :visited son pseudo clases link y sólo se pueden asignar al elemento La pseudo clase link debe colocarse antes de las demás. Cuando se pulsa por ejemplo un enlace que fue visitado previamente, al enlace le afectan las pseudo-clases :visited, :hover y :active.

Las pseudo clases :link y :visited son pseudo clases link y sólo se pueden asignar a enlaces. Las pseudo clases :hover, :active y :active son pseudo clases dinámicas que, en teoría, se pueden asignar a cualquier elemento.

Para darle un estilo apropiado a los enlaces, hay que colocar la regla :visited después de la regla :link pero antes de las reglas :hover y :active, según lo definido por el orden **LVHA**: :link, :visited, :hover y :active.

Por último, también es posible aplicar estilos combinando varias pseudo-clases compatibles entre sí. La siguiente regla CSS por ejemplo sólo se aplica a aquellos enlaces que están seleccionados y en los que el usuario pasa el ratón por encima:

```
a:focus:hover { ... }
```

Los **pseudoelementos** se comportan de manera similar. Sin embargo, actúan como si hubieras añadido un elemento HTML totalmente nuevo en el marcado, en lugar de haber aplicado una clase nueva a los elementos presentes. Los pseudoelementos empiezan con un doble signo de dos puntos ::.

Algunos ejemplos de pseudo elementos:

::first-line hay que tener en cuenta que la longitud de la primera línea depende de muchos factores, incluido el ancho del elemento, el ancho del documento y el tamaño de fuente del texto. Este pseudoelemento permite seleccionar la primera línea de texto de un elemento.

Así, la siguiente regla CSS muestra en mayúsculas la primera línea de cada párrafo:

```
p::first-line { text-transform: uppercase; }
```

Este pseudo-elemento sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

::first-letter aplica estilos a la primera letra de la primera línea de un elemento a nivel de bloque.

Así, la siguiente regla CSS muestra en mayúsculas la primera letra de cada párrafo:

```
p::first-letter { text-transform: uppercase; }
```