Unidad Formativa UF2: Elaboración de hojas de estilo

Acción Formativa: Web Development Bootcamp

Autor: Javier Gil Motos - EOI

Índice

Contenido

Qué es una hoja de estilo

En las primeras versiones del HTML, el código fuente de una página web contenía tanto la información (el contenido) como la forma de representarse (el diseño o formato). Actualmente, estos dos aspectos se pueden separar. La página web (el documento html) sólo debe contener información, mientras que el formato se debe definir en las llamadas hojas de estilo (en inglés, CSS, Cascading Style Sheets, es decir, Hojas de Estilo en Cascada).

Las ventajas de utilizar hojas de estilo son muchas, sobre todo permiten hacer un diseño consistente y fácil de modificar. Si varias páginas web hacen referencia a la misma hoja de estilo, para cambiar la apariencia de un elemento de todas las páginas (por ejemplo, del encabezado <h1>) es suficiente con hacer los cambios en un único lugar, en la hoja de estilo.

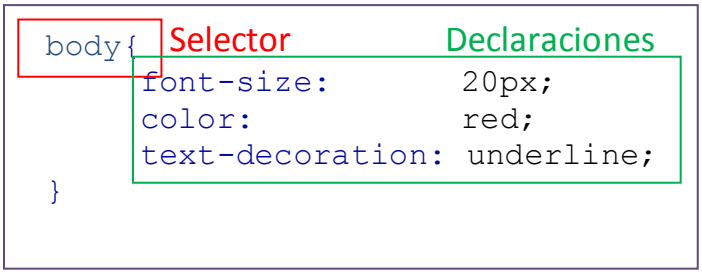
Sintaxis de las hojas de estilo

Una hoja de estilo es un fichero de texto plano (sin formato) en el que se define el aspecto de cada una de las etiquetas de una página web.
Una hoja de estilo está formada por una o varias sentencias. Existen dos tipos de sentencias: las reglas-arroba y las reglas.

Las reglas-arroba empiezan por el símbolo de arroba (@).

Las reglas están formadas por un selector y un bloque de declaración. El bloque de declaración empieza y acaba con llaves { }.

Regla



El bloque de declaraciones contiene en su interior una o más propiedades, las cuáles se descomponen, a su vez, en dos partes: nombre de la propiedad y su valor correspondiente:

Propiedad	Valor
font-size:	20px;

El símbolo de separación entre el nombre de la propiedad y su valor son los dos puntos (:)

Algunas reglas-arroba (@charset, @import o @namespace) deben aparecer al principio de la hoja de estilo, pero el resto (@font-face, @media, etc.) pueden escribirse en cualquier lugar de la hoja de estilo.

Cada bloque de declaración está formado por una o varias propiedades y su valor (o valores) correspondiente. Las propiedades van separadas entre sí por puntos y comas. El valor (o valores) van separados de las propiedades por dos puntos. En general, si una propiedad necesita varios valores, los valores van separados por espacios en blanco. En general, si una propiedad admite varios valores alternativos, los valores van separados por comas.

Ejemplo:

```
p {
  border-top: red 3px solid;
  border-bottom: red 3px solid;
}
```

Comentarios en las hojas de estilo

Una hoja de estilo puede contener comentarios. Los delimitadores del comentario son /* ... */. Los comentarios pueden extenderse varias líneas, como ilustra el siguiente ejemplo:

```
/* Autor: Javier Gil
   Fecha: 16 de octubre de 2015
*/
p {
  color : black ;              /* texto negro */
  background-color : white ;   /* sobre fondo blanco */
}
```

Formas de incluir estilos en una página

Existen tres formas de incluir estilos en nuestras páginas web:

1. Estilos en línea: las propiedades CSS se establecen dentro del atributo *style* de cualquier etiqueta HTML.

```
<p style="font-family: sans-serif;color: black;">
```

   Este método tiene el inconveniente de que en caso de que deseemos reutilizar estos estilos, no podremos, ya que aplican únicamente a la etiqueta en la que se encuentran incluidos, por lo que habría que copiarlos en todas las etiquetas deseadas.

   Es importante resaltar que los estilos aplicados dentro del atributo *style* de HTML, prevalecen sobre todos los demás que puedan ser de aplicación a esta etiqueta.

   Los estilos en línea sólo se utilizan en casos puntuales y muy concretos, donde queremos estilar un elemento y estamos absolutamente seguros de que dicho estilo no se repetirá ni aplicará en ningún otro lugar de nuestro sitio web.

2. Estilos a nivel de página: las reglas CSS se incluyen dentro de la etiqueta `<style>`, en cualquier lugar de nuestra página web (aunque se recomienda colocarla en el head) afectando así a la página entera.

```
<style type="text/css">
h2 {
      font-style: italic;
}

p {
      color: orange;
}
</style>
```

3. Hojas de estilo: las reglas CSS se incluyen en un fichero separado, con extensión .css
   Este fichero contendrá únicamente reglas CSS en su interior y podrá ser enlazado en nuestras páginas web mediante la etiqueta <link>. La ventaja de tener las reglas en un fichero aparte es evidente, ya que tenemos todo el CSS centralizado en un único lugar y los cambios que hagamos allí, se aplicarán automáticamente a todo nuestro sitio web. Un ejemplo sería éste:

index.htm

```
<!DOCTYPE html>
<html lang="es-es">
<head>
  <title>Hojas de estilo</title>
  <link rel="stylesheet" type="text/css" href="css/estilos.css" />
  ...
```

estilos.css

```
@charset "utf-8"

h2 {
     font-style: italic;
}

p {
     color: orange;
`
```

Consejos a seguir en las hojas de estilo

Cuando escribamos reglas en una hoja de estilo, es recomendable seguir una serie de pautas que nos ayudarán a evitar cometer errores. Algunas de las pautas más comunes son:

- Se recomienda utilizar únicamente caracteres "ingleses" en las hojas de estilo. Por ejemplo, se recomienda no utilizar acentos, ñ, ç, etc. en los nombres de las clases o ids.

- Escribir cada propiedad en una línea, sangrando cada línea con dos espacios en blanco y completando la línea con un punto y coma.
- No escribir espacios antes de los dos puntos que separan la propiedad y su valor. Escribir un espacio tras los dos puntos.
- No escribir espacios en blanco antes del punto y coma final de cada línea de propiedad.
- Escribir la llave de cierre en una línea, sin sangrar. En esa línea sólo estará la llave de cierre.
- Dejar una línea en blanco entre regla y regla.

```
h1 {
  background-color: black;
  color: white;
}

p {
  text-indent: 3em;
}
```

Orden de las sentencias en la hoja de estilo

Se pueden utilizar dos criterios en cuanto a la ordenación de las reglas en las hojas de estilo:

- Escribir las sentencias en el orden en que aparecen los elementos en la página web. Este criterio se suele utilizar cuando una hoja de estilo se va a aplicar a una sola página, lo que será habitual en este curso, en que cada ejercicio tiene su propia página web.

```
html {
  /* … */
}

body {
  /* … */
}

h1 {
  /* … */
}
```

- Agrupar las sentencias por categorías. Por ejemplo, escribir primero las relativas a los bloques y después las relativas a elementos en línea. Dentro de cada categoría se pueden escribir las sentencias por orden alfabético o por el orden en que suelen aparecer en la página. Este criterio se suele utilizar cuando una hoja de estilo se va a aplicar a muchas páginas distintas, lo que suele ser habitual en sitios web reales.

```
h1 {
  /* ... */
}

p {
  /* ... */
}

address {
  /* ... */
}
```

Orden de las propiedades en cada sentencia

Las propiedades se escribirán en el siguiente orden:

- Propiedades relacionadas con el posicionamiento (position, float, clear, width, height, top, right, bottom, left, overflow, etc.)
- Propiedades relacionadas con el modelo de caja (border, margin, padding, etc.)
- Propiedades relacionadas con el color (background-color, color)
- Resto de propiedades (en orden alfabético)

Es común ver los valores de la propiedad border escritos en el orden *color, grosor, estilo*.

Nombres de las clases

Se intentará que los nombres de las clases hagan referencia a categorías más que a aspectos visuales. Por ejemplo, conviene evitar llamar a las clases con nombres de colores (azul, blanco, etc.) porque en una página web real los colores es uno de los elementos que más se suele cambiar con el paso del tiempo.
Lógicamente, esta recomendación es más importante en las webs reales que en ejercicios teóricos, ya que en las webs reales se modifica a menudo el aspecto

visual, mientras que en un curso cada ejercicio es independiente y sólo se realiza una vez.

Por otro lado, esta recomendación es una de las más ambiguas y por tanto difíciles de seguir. Por ejemplo, aunque se desaconseje utilizar nombres de colores, sería más admisible definir clases "izquierda" y "derecha" para imágenes flotantes, que también hacen referencia a un aspecto visual. El motivo es que es menos probable que decidamos cambiar a la derecha todas las imágenes flotantes que salían a la izquierda y viceversa, que decidamos cambiar el color de un elemento o cambiar la gama de colores de un sitio web.

En los casos en los que no sea fácil nombrar las categorías, se recomienda utilizar nombres genéricos, por ejemplo "color-1", "color-2", "color-3", sería preferible a "azul", "rojo", etc.
Cuando el nombre de la clase contiene varias palabras o números, separar las palabras con guiones (-): por ejemplo "estilo-1", "fecha-edicion", "autor-libro", etc.

Orden de las clases en la hoja de estilo

Si se definen clases, se agruparán todas las sentencias en la primera aparición de la etiqueta (aunque en la página web esas clases aparezcan más tarde).

```
h1 {
   /* ... */
}

p {
   /* ... */
}

p.importante {
   /* ... */
}

p.errata {
   /* ... */
}
```

Clases o id

Se utilizarán preferentemente clases en vez de id, aunque las clases se vayan a asignar a un único elemento.

Únicamente se aconseja el uso de id cuando asignar la misma clase a varios elementos provoque problemas en la página (por ejemplo, en el caso de posicionamiento absoluto de elementos).

Hojas de estilo alternativas

Una página web puede tener enlazadas varias hojas de estilo distintas para que el usuario elija cuál se aplica.

Tal y como se dijo anteriormente, en una página web, la etiqueta <link /> situada en la cabecera (<head>) indica la ubicación y nombre de la hoja de estilo enlazada mediante el atributo href:

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

Se pueden enlazar dos o más hojas de estilo, con lo que el código fuente contendrá tantas etiquetas <link /> como hojas enlazadas:

```
<link href="hoja1.css" rel="stylesheet" type="text/css" />
<link href="hoja2.css" rel="stylesheet" type="text/css" />
```

En este caso, el navegador aplica simultáneamente las dos hojas de estilo a la página web. Si un elemento está definido en las dos hojas, el navegador hará lo que diga la última hoja de estilo de la lista.

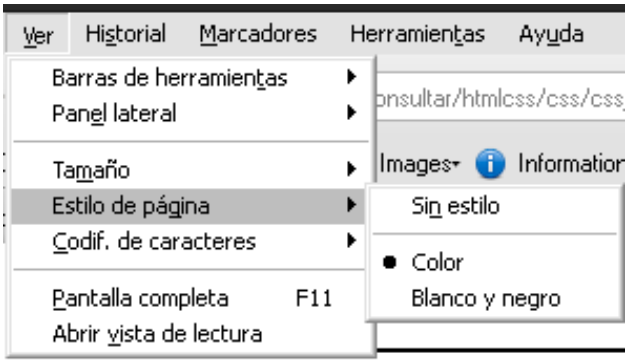
Para que sean hojas de estilo alternativas, es decir, para que el usuario pueda elegir en el navegador qué hoja de estilo quiere aplicar, es suficiente con que los enlaces a las hojas de estilo tengan el atributo title:

```
<link href="hoja1.css" rel="stylesheet" type="text/css" title="Hoja1" />
<link href="hoja2.css" rel="stylesheet" type="text/css" title="Hoja2" />
```

Nota: Si deseamos tener varias hojas de estilo combinadas en un único nombre, deberemos colocar el mismo atributo *title* a todas.

Hojas de estilo alternativas en navegadores

Cuando visitemos con nuestro navegador un sitio web que tenga hojas de estilo alternativas, podremos seleccionar la que queremos mostrar simplemente haciendo click en el menú *ver – estilo de página* en caso de que usemos Internet Explorer o Mozilla Firefox. Si usamos el navegador Chrome, deberemos descargar la extensión AltCSS para poder hacer uso de esta funcionalidad.



Estilos en cascada

Las hojas de estilo se llaman hojas de estilo "en cascada" porque:

- Las propiedades de estilo pueden estar escritas en varios sitios (en varios lugares de la página web o de la hoja de estilo) y dependiendo del sitio, afectan a más o menos elementos.
- Cuando un elemento está contenido en otro (por ejemplo, un párrafo <p> dentro de una división), al elemento de dentro se le aplican también las propiedades definidas para el elemento de fuera (al párrafo <p> se le aplicarían las propiedades definidas para la división).
- Dos reglas distintas pueden ser de aplicación a un mismo elemento (por ejemplo, a un párrafo <p> con clase “nombre”, le es de aplicación tanto el selector .nombre como el selector p).

Si las propiedades (escritas en diferentes sitios o para diferentes elementos) no entran en conflicto, el navegador aplica todas las propiedades. Por ejemplo, si el color de fondo de un elemento está definido en un sitio y el tamaño de letra en otro sitio, el navegador aplicará ambas propiedades al elemento.

Pero si las propiedades entran en conflicto (por ejemplo, el color del fondo del elemento está definido en varios sitios con colores distintos), existen reglas para decidir qué propiedad tiene preferencia.

Reglas de aplicación de los estilos

Si se define la misma propiedad para la misma etiqueta con el mismo selector en dos sitios distintos, las reglas de precedencia son las siguientes:

- Las propiedades definidas en un atributo style (estilos en línea) se imponen a las propiedades definidas en la etiqueta <style>.
- Las propiedades definidas en la etiqueta <style> se imponen a las propiedades definidas en una hoja de estilo enlazada (etiqueta link).

Además de estas propiedades definidas por el creador de la página web, hay que tener en cuenta que también se aplican las propiedades definidas en la hoja de estilo por defecto del navegador.

Si las propiedades se encuentran definidas en diferentes hojas de estilo, el navegador aplica el valor definido en la última hoja de estilo enlazada (es decir, en el último enlace <link /> del <head>).

Elementos dentro de otros

Cuando un elemento está contenido en otro (por ejemplo, un párrafo <p> dentro de una división), al elemento de dentro se le aplican también las propiedades definidas para el elemento de fuera.

Si una misma propiedad está definida para el elemento inferior y para el superior, se aplica el valor establecido para el elemento inferior.

Reglas distintas que se aplican al mismo elemento

Dos reglas distintas se aplican a un mismo elemento cuando el elemento coincide con los selectores de ambas reglas. La regla que se aplica es la del selector de mayor especificidad. La especificidad de un selector se calcula atendiendo a los siguientes criterios:

1. Número de atributos id en el selector: la regla que tenga "más almohadillas" en el selector es la que tiene preferencia. Por ejemplo, el selector "`p#cabecera #elem1`" tiene preferencia sobre el selector "`#otroElemento`". En caso de empate entre las dos reglas, se aplica el siguiente punto.

2. Número de otros atributos y pseudo-clases en el selector (los pseudo-elementos se ignoran). Por ejemplo, el selector "`p.titulo .parrafo`" tiene preferencia sobre el selector "`.textoInferior`", ya que el número de IDs coincide en ambos casos (0), pero el primero tiene más clases (`.titulo` y `.parrafo`) que el segundo.

3. Número de elementos en el selector: si hay empate entre el número de IDs y el número de clases y pseudoclases, tendrá preferencia el selector con mayor número de elementos, sean los que sean. Por ejemplo, el selector "`a.titulo > p #elem1`" tendrá más preferencia que "`#elem1 > .parrafo`" ya que tienen igualdad de elementos ID, igualdad de clases, pero el primero tiene 3 elementos y el segundo, 2.

4. Posición en la hoja de estilo. Ante una igualdad entre selectores en número de IDs, número de clases y número de elementos, prevalecerá la regla que aparezca más tarde en la hoja de estilos.

Estos criterios se aplican en orden, es decir, primero se comparan el número de atributos id de cada selector. Si un selector tiene más que el otro, se aplica esa regla, si el número es el mismo, entonces se calcula el segundo criterio (número de otros atributos y pseudo-clases). Y así, sucesivamente.

Selectores

Los selectores de las reglas CSS se pueden escribir de varias formas distintas, de forma que su ámbito de aplicación varíe. Estas formas son las siguientes:

Selectores de tipo: E, E.e y E#e

Si se escribe una etiqueta (E), las propiedades afectan a todos los elementos con etiqueta E. Por ejemplo, una `p` hace que afecte a todos los párrafos <p>.

Ejemplo:

```
p{
  font-size: 20px;
}
```

Si se escribe una etiqueta seguida de un punto y un nombre de clase (E.e), las propiedades afectan a todos los elementos con etiqueta E cuyo atributo class tenga el valor e. Por ejemplo, p.aviso hace referencia a todos los párrafos de tipo `<p class="aviso">`

Ejemplo:

```
p.aviso{
  font-size: 20px;
}
```

Si se escribe una etiqueta seguida de una almohadilla y un nombre de id (E#e), las propiedades afectan al elemento con etiqueta E cuyo atributo id tenga el valor e. Por ejemplo, p#aviso se refiere al párrafo `<p id="aviso">`.

Ejemplo:

```
p#aviso{
  font-size: 20px;
}
```

Hay que tener en cuenta que el atributo id no se puede repetir, es decir, no puede haber dos elementos con el mismo valor del atributo id (independientemente de que los elementos tengan etiquetas iguales o distintas).

Selectores universales: *, . y

Si se escribe un asterisco (*), las propiedades afectan a todos los elementos de la página.

Ejemplo:

```
*{
  box-sizing: border-box;
}
```

Para definir una clase que afecte a cualquier etiqueta, se debe escribir un punto y el nombre de la clase. Esto sería equivalente a escribir *.clase (O, dicho de otro modo, "*.clase" es lo mismo que poner ".clase").

```
.aviso{
  font-size: 20px;
}
```

Para definir un id que afecte a cualquier etiqueta, se debe escribir una almohadilla y el nombre del id (#nombre) si bien, dado que el atributo id no se puede repetir en una página, esta regla haría referencia a un único elemento, el que tenga el atributo id="nombre".

Ejemplo:

```
#nombre{
  font-size: 20px;
}
```

Nota: Podemos colocar tantos selectores de este tipo como queramos, separándolos por comas. En el ejemplo siguiente, la regla aplicaría a todos los párrafos, a todos los elementos de la clase aviso y al elemento con ID "cabecera":

```
p,.aviso,#cabecera{
  font-size: 20px;
}
```

Selectores de descendientes: E F

Si se escriben dos etiquetas seguidas (E F), las propiedades afectan a los elementos con la segunda etiqueta (F) contenidos dentro de la primera etiqueta (E), aunque haya etiquetas intermedias.

En general, si se escriben varias etiquetas seguidas (E F G ...), las propiedades afectan a los elementos con la última etiqueta contenidos dentro de la etiqueta anterior, contenidos a su vez dentro de etiqueta anterior y así sucesivamente, aunque haya etiquetas intermedias.

Por ejemplo, si tenemos el selector "`section div p.aviso`", afectaría a todos los párrafos con la clase "aviso" que se encuentren dentro de un div y que a su vez el div esté dentro de un elemento section. No importa si entre estos elementos hay otros, como se ilustra en el siguiente ejemplo HTML:

```
...
<section>
  <h1>Cabecera</h1>
  <div>
    <a href="http://.....">
      <p class="aviso">Texto del párrafo</p>
    </a>
  </div>
</section>
...
```

El selector anterior aplica sin problemas, ya que aunque haya un <a> entre el div y el párrafo, el párrafo está dentro del div.

Selectores de hijos: E > F

Si se escriben dos etiquetas seguidas separadas por un signo "mayor que" (E > F), las propiedades afectan a los elementos con la segunda etiqueta (F) contenidos dentro de la primera etiqueta (E), pero no afecta si hay etiquetas intermedias entre ellos, es decir, aplicaría sólo a los F que sean hijos inmediatos de E.

Si tomamos el ejemplo del apartado anterior, el selector "`section > div > p.aviso`" no aplicaría, ya que p.aviso no es hijo directo de div. El siguiente selector sí aplicaría: "`section > div > a > p.aviso`"

Selectores de adyacentes: E+F

Si se escriben dos etiquetas seguidas separadas por un signo "más" (E + F), las propiedades afectan únicamente a los elementos con la segunda etiqueta (F) que van justo después de un elemento con la primera etiqueta (E), es decir, aplicaría a los F que sean hermanos inmediatos de un E.

Por ejemplo, el siguiente selector aplicaría al ejemplo anterior: "`h1+div`". Si observamos el ejemplo, vemos que hay un div que es hermano directo de un h1.

Selectores de atributo

Hay cuatro formas principales de seleccionar elementos con determinados atributos: `E[atributo]`, `E[atributo="valor"]`, `E[atributo~="valor"]`, `E[atributo|="valor"]`.

1. Si se escribe una etiqueta seguida del nombre de un atributo entre corchetes, E[atributo], las propiedades afectan a todos los elementos que tengan establecidos ese atributo.

Ejemplo:

```
video[autoplay]{
        ...
}
```

Este selector haría referencia a todas las etiquetas <video> que tengan establecido el atributo "autoplay", como por ejemplo ésta:

```
...
<video autoplay src="video.mp4">
</video>
...
```

---

2. Si se escribe una etiqueta seguida del nombre de un atributo igual a un valor entre corchetes, E[atributo="valor"], las propiedades afectan a todos los elementos que tengan establecidos ese atributo exactamente con ese valor.

Ejemplo:

```
a[title="Enlace a la web de mi empresa"]{
        ...
}
```

---

3. Si se escribe una etiqueta seguida del nombre de un atributo tilde-igual a un valor entre corchetes, E[atributo~="valor"], las propiedades afectan a todos los elementos que tengan establecidos ese atributo con ese valor (entre otros).

Ejemplo:

```
a[title~="web"]{
        ...
}
```

Este selector aplicaría, por ejemplo, al siguiente enlace:

```
<a href="http://...." title="Enlace a la web de mi empresa">Mi empresa</a>
```

Esto es así, ya que el atributo "title" de la etiqueta "a", contiene la palabra "web".

---

4. Si se escribe una etiqueta seguida del nombre de un atributo |-igual a un valor entre corchetes, E[atributo|="valor"], las propiedades afectan a todos los elementos que tengan establecidos ese atributo con un valor que comience por ese valor seguido de un guión.

Ejemplo:

```
a[title|="Enlace"]{
        ...
}
```

```
<a href="http://...." title="Enlace-a-la-web-de-mi-empresa">Mi empresa</a>
```

El selector aplicaría, ya que el atributo "title" del elemento "a" comienza por el valor "Enlace-".

Pseudo-clases y pseudo-elementos

Qué son las pseudo-clases y los pseudo-elementos

Las hojas de estilo asocian características de estilo a los elementos basándose en las etiquetas de los elementos y en su posición relativa (jerarquía en el documento HTML). Las pseudo-clases y los pseudo-elementos permiten hacer referencia a determinados elementos sin basarse en la información contenida en la jerarquía HTML. **Por ejemplo, podemos hacer referencia al evento de "pasar el ratón por encima de un elemento".**

La sintaxis de las pseudo-clases y los pseudo-elementos es la misma: etiqueta:pseudo-elemento (el nombre de la etiqueta y del pseudo-elemento o de la pseudo-clase separado por dos puntos).

La pseudo-clase :first-child

La pseudo-clase :first-child hace referencia al primer elemento de un tipo contenido dentro de otro. Por ejemplo, `div :first-child` haría referencia a cualquier elemento HTML que sea el primer hijo de un DIV. Si pusiéramos "`div p:first-child`", haríamos referencia a un párrafo, que debe ser el primer hijo de un div.

Las pseudo-clases de enlace :link y :visited

La pseudo clase :link (o a:link) permite especificar el aspecto de los enlaces que todavía no han sido visitados.

La pseudo clase :visited (o a:visited) permite especificar el aspecto de los enlaces que ya han sido visitados alguna vez.

Nota: si queremos que el navegador "olvide" los enlaces que hemos visitado, tendríamos que borrar la caché del mismo.

Las pseudo-clases dinámicas :hover, :active y :focus

1. La pseudo-clase `:hover` permite especificar el aspecto del elemento sobre el que se encuentra el ratón. Si especificamos un selector tal que "`a:hover{…}`" hará referencia a aquellos enlaces sobre los que se

encuentra posado el ratón. En el momento en que el ratón deje de estar sobre un enlace, ese estilo **dejará de aplicarse.**

2. La pseudo-clase `:active` permite especificar el aspecto de un elemento cuando se hace clic sobre él (y mientras se mantiene el botón del ratón apretado). En el momento en que dejemos de pulsar el ratón sobre el elemento, este estilo dejará de aplicarse.

3. La pseudo-clase :focus permite especificar el aspecto de un elemento cuando este tiene el foco. Los elementos que admiten el foco en una página web son aquellos que reaccionan a entrada por teclado (por ejemplo, los elementos de los formularios o los enlaces). En este tema aún no hemos visto elementos que puedan recibir el foco del ratón, por lo que no podremos probar este estilo.

La pseudo-clase de idioma :lang()

La pseudo-clase de idioma :lang() permite especificar el aspecto de los elementos de un idioma determinado. En el ejemplo siguiente se cambia las comillas utilizadas en una cita <q> según el idioma indicado en el atributo lang.

```
<p>Ambrose Bierce dijo (más o menos) que
<q lang="es">una cita es una manera de repetir
erróneamente las palabras de otros</q>.</p>

<p>Ambrose Bierce dijo (más o menos) que
<q lang="en">a quotation is the act of repeating
erroneously the words of another</q>.</p>
```

```
q:lang(es) {
  quotes: "«" "»";
}

q:lang(en) {
  quotes: '"' '"';
}
```

El resultado sería el siguiente:

Ambrose Bierce dijo (más o menos) que «una cita es una manera de repetir erróneamente las palabras de otros».

Ambrose Bierce dijo (más o menos) que “a quotation is the act of repeating erroneously the words of another”.

Los pseudo-elementos :first-line y :first-letter

El pseudo-elemento :first-line permite especificar el aspecto de la primera línea de texto, mientras que :first-letter permite especificar el aspecto de la primera letra de texto. Si el primer carácter no es un carácter alfanumérico, el pseudo-elemento se aplica hasta el primer carácter alfanumérico.

Ejemplo:

```
p:first-letter {
  color: #FF0000;
}
```

```
...
<p>¿Cuántas letras se ven en rojo?</p>
...
```

Resultado:

¿Cuántas letras se ven en rojo?

Los pseudo-elementos :before y :after

Los pseudo-elementos :before y :after permiten añadir contenido a un elemento desde la hoja de estilo, al principio o al final del elemento. El contenido generado mediante la propiedad content puede incluir texto, o incluso una imagen indicando su URI (con la función URL()).
No es válido incluir etiquetas html en la propiedad content, es decir que no sólo los navegadores no hacen caso y lo incluyen como texto, no como código html, sino que al validar la hoja de estilo daría error.

Ejemplo:

```
p.cuidado:before {
  content: "Aviso: ";
  font-weight: bold;
  text-decoration: underline;
}
```

```
<p>Este párrafo es un párrafo &lt;p&gt; sin clase.</p>

<p class="cuidado">Este párrafo es un párrafo &lt;p&gt; con
clase "cuidado".</p>
```

Resultado:

Este párrafo es un párrafo <p> sin clase.

Aviso: Este párrafo es un párrafo <p> con clase "cuidado".

Tabla resumen de selectores vistos hasta ahora:

SELECTOR	EJEMPLO	DESCRIPCIÓN DEL EJEMPLO
etiqueta	div	Todos los
etiqueta, etiqueta	div,p	Todos los y todos los <p>
.class	.cabecera	Todos los elementos con class=”cabecera”
#id	#boton_uno	El elemento con id=”boton_uno”
elemento elemento	div p	Todos los <p> que estén dentro de un
elemento > elemento	div > p	Todos los <p> que sean hijos directos de un
elemento + elemento	div + p	Todos los <p> que sean hermanos directos de un
[atributo]	[title]	Todas las etiquetas que tengan el atributo title.
[atributo=valor]	[title=”Algo”]	Todas las etiquetas que tengan el atributo title=”Algo”.
[atributo~=valor]	[class=”titulo”]	Toda las etiquetas que tengan en el atributo “class” la palabra “titulo”.
[atributo|=valor]	[lang|=”es”]	Todas las etiquetas que tengan el atributo “lang” y cuyo valor comience por “es”.
:first-child	p:first-child	Todos los <p> que sean el primer hijo de otra etiqueta.
:link	a:link	Todos los enlaces no visitados aún.
:visited	a:visited	Todos los enlaces ya visitados.
:hover	div:hover	Todos los que tengan el ratón posado sobre ellos.
:active	div:active	Todos los sobre los que se tenga pulsado el ratón.
:focus	input:focus	Todos los <input> que tengan el foco.
:lang(idioma)	p:lang(es)	Todos los <p lang=”es”>
:first-line	p:first-line	La primera línea de todos los <p>
:first-letter	p:first-letter	La primera letra de todos los <p>
:before	p:before	Inserta contenido antes de cada <p>
:after	p:after	Inserta contenido después de cada <p>

Propiedades CSS más importantes

Aquí se resumen las propiedades CSS más importantes, vistas en clase. Se deja a discreción del alumno investigar nuevas propiedades (así como sus valores) en las numerosas referencias, tanto oficiales como no oficiales, disponibles online.

PROPIEDAD	DESCRIPCIÓN	VALORES
Fondos		
background-color	Color de fondo	#0000FF, rgb(0,0,255)
background-image	Imagen de fondo	url("imagen.jpg"), none
background-repeat	Repetición o no de la imagen de fondo	repeat, no-repeat, repeat-x…
background-attachment	Movimiento del fondo con el scroll	fixed, scroll…
Fuentes		
font-size	Tamaño de la fuente	small, medium, large, 20px, 5%...
font-family	Familia de la fuente	Arial, Verdana, serif…
font-style	Estilo de la fuente	normal, italic, oblique…
font-weight	Anchura de la fuente	normal, bold, lighter, 100-900…
color	Color del texto	#00FF00, rgb(0,255,0)
letter-spacing	Espacio entre letras del texto	normal, 5px, 2cm…
text-align	Alineación horizontal del texto	left, right, center, justify
text-decoration	Decoración añadida al texto	underline, overline, line-through, blink, none…
text-transform	Mayúsculas/minúsculas	none, capitalize, uppercase, lowercase
vertical-align	Alineación vertical	5px, 2pt, 5cm, 2%, top, middle, bottom…
Dimensiones		
height	Altura de un elemento	20px, 50%...
width	Anchura de un elemento	100px, 70%...
min-height	Altura mínima de un elemento	100px, 70%...
max-height	Altura máxima de un elemento	100px, 70%...
min-width	Anchura mínima de un elemento	100px, 70%...

max-width	Anchura máxima de un elemento	100px, 70%...
Márgenes y rellenos		
margin-top	Margen superior de un elemento	10px, 2%, auto…
margin-right	Margen derecho de un elemento	10px, 2%, auto…
padding-bottom	Relleno inferior de un elemento	10px, 2%, auto…
padding-left	Relleno izquierdo de un elemento	10px, 2%, auto…
Bordes		
border-top-width border-right-width border-bottom-width border-left-width	Anchura de un borde cualquiera (superior, derecho, inferior o izquierdo)	2px, thin, medium…
border-style	Estilo del borde	none, dotted, dashed, solid, groove…
border-color	Color del borde	#FF0000, rgb(255,0,0)
Listas		
list-style-image	Especificar una imagen como marcador de una lista	url("imagen.jpg"), none…
list-style-position	Especifica si los marcadores de la lista pertenecen al flujo del contenido	inside, outside
list-style-type	Tipo de marcador de los elementos de la lista	none, circle, disc, square, decimal…
Tablas		
border-collapse	Especifica si los bordes de la table deben estar colapsados	collapse, separate
border-spacing	Distancia entre las celdas	5px, 2%...
Posicionamiento		
top right bottom left	Especifica la posición de la parte superior/derecha/inferior/izquierda de un elemento posicionado	5px, 2cm, 7%, 1em…
clear	Especifica si a los lados de un elemento se permiten o no elementos flotantes	left, right, both, none
display	Especifica cómo se debe mostrar un elemento HTML	none, inline, block…

float	Especifica si una caja debe flotar	left, right, none
position	Especifica el tipo de método de posicionamiento utilizado para un elemento (estática, relativa, absoluta o fija)	static, absolute, relative, fixed
visibility	Especifica si un elemento es visible o no	visible, hidden, collapse
z-index	Establece el orden de aparición de las cajas que se solapen en pantalla (a número más alto, se verá sobre las demás)	auto, 1, 2, 999…

Las propiedades **visibility** y **display** tienen una peculiaridad que las diferencia en el aspecto de permitir mostrar y ocultar un elemento: mientras que "`visibility: hidden`" oculta el elemento y deja en blanco el espacio que ocupa el mismo, "`display: none`" elimina por completo el elemento de la página y su espacio lo ocupan los elementos circundantes.

Propiedades abreviadas (shorthands)

Algunas propiedades comparten un mismo prefijo, como habrá podido observarse con la anchura de los bordes (border-top-width, border-right-width, border-bottom-width y border-left-width). En estos casos se puede utilizar una abreviatura, como es el caso que hemos comentado, donde podemos usar la propiedad "border-width" para afectar a todos los bordes por igual:

```
#div1{
  border-width: 10px;
}
```

También podemos agrupar en una abreviatura, a su vez, los valores de las propiedades border-width, border-style, border-color…

```
#div1{
       border: 10px solid #0000FF;
}
```

El ejemplo anterior es equivalente a esto:

```
#div1{
  border-width: 10px;
  border-style: solid;
  border-color: #0000FF;
}
```

Hay que llevar cuidado con el uso de las abreviaturas, dado que si no especificamos todas las propiedades que engloba, éstas se establecerán a los valores por defecto que contemplen. Por ejemplo, "`border: 10px;`" equivaldría a "`border: 10px none`", con lo que no tendríamos borde alguno.

Nota: Cuando usamos abreviaturas, los distintos valores especificados se separan con espacios.

Fuentes externas

En ocasiones nos interesa utilizar en nuestra página web fuentes para el texto que no están incluidas en las familias que el navegador reconoce (Arial, serif, Verdana, Times, etc.). En estos casos, querremos poder utilizar una fuente de cuyo fichero disponemos, colocándolo en nuestro servidor web para que el navegador de los usuarios se lo descargue y lo use como cualquier otra fuente del sistema. Este comportamiento se consigue con el uso de la regla-arroba @font-face. En el siguiente ejemplo podemos observar cómo se incluye una fuente externa y se utiliza la misma posteriormente en la propiedad font-family, como una alternativa más:

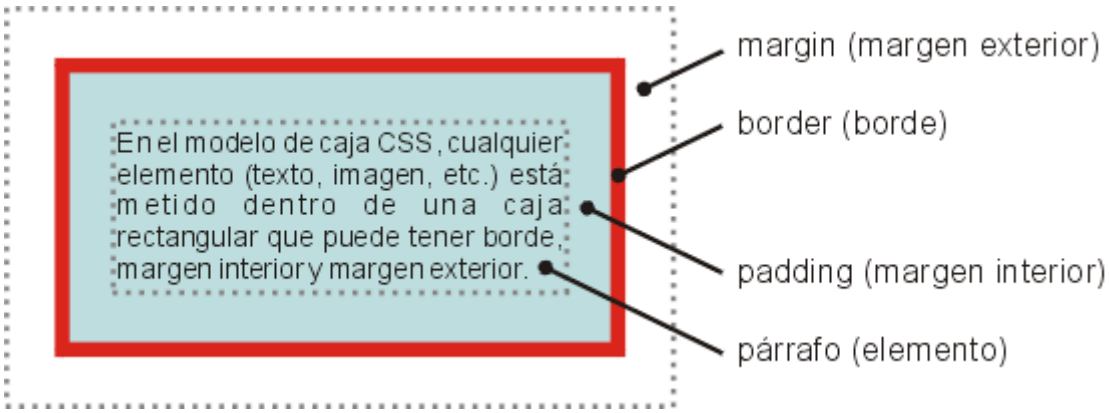
```
@font-face{
        font-family: "sefcarm";
        src:url("../fonts/2prot.ttf");
}

.fuenteSef{
        font-family: "sefcarm",Verdana;
}
```

Modelo de caja

Cualquier elemento de una página web está contenido en una "caja" rectangular. Esta caja puede tener:

- Un borde visible (que se define mediante la propiedad border).
- Un margen exterior transparente por fuera del borde (que se define mediante la propiedad margin).
- Un margen interior transparente entre el borde y el elemento (que se define mediante la propiedad padding).

La imagen siguiente muestra un párrafo de texto con borde y márgenes exterior e interior. En la imagen, los límites del borde y de los márgenes se han indicado mediante líneas discontinuas, pero en realidad esos límites no se ven en los navegadores.



Tanto el margen exterior (margin) como el margen interior (padding) son transparentes. Por eso el margen interior se ve del color de fondo del elemento (en el ejemplo, del color de fondo del párrafo), mientras que el margen exterior se ve del color de fondo del elemento padre (en el ejemplo, del color de fondo de la página).

Cuando establecemos márgenes, bordes o rellenos, podemos utilizar una propiedad abreviada del tipo `padding: 1px 2px 3px 4px;` En estos casos, el orden de las propiedades es “top right bottom left”. Si sólo especificamos dos valores como en `padding: 1px 2px;` el primero (1px) hace referencia al top y al bottom, mientras que el segundo (2px) hace referencia al right y al left.

Posicionamiento de elementos

Los elementos de una página web están contenidos en una caja rectangular, de acuerdo con el modelo de caja. La manera en que los diferentes elementos de una página web se distribuyen en la pantalla dependen del esquema de posicionamiento elegido.
Existen tres esquemas de posicionamiento:

- Normal
- Flotante
- Absoluto

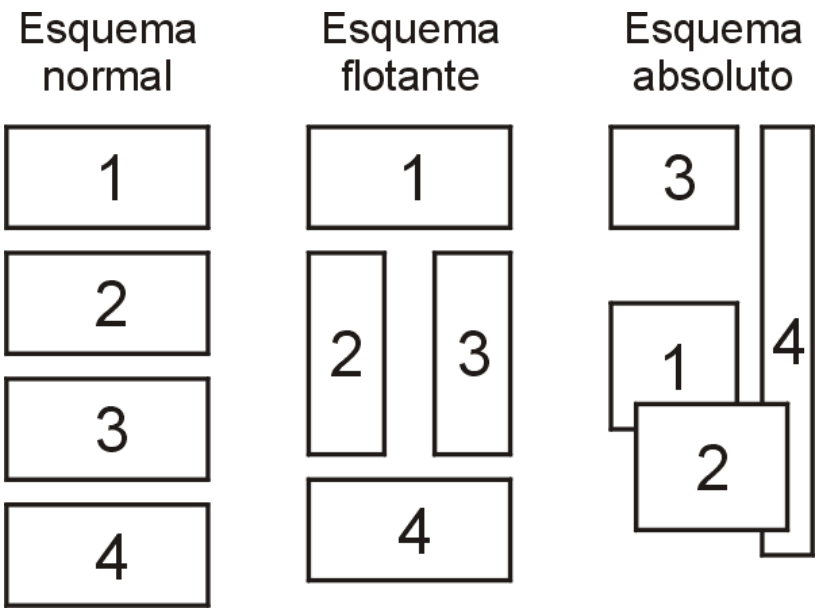
El esquema normal es el que se aplica por omisión. En el esquema de posicionamiento normal, los elementos se muestran en la pantalla en el mismo orden en que se encuentran en el código fuente de la página. Los elementos pueden ser elementos de bloque o elementos en línea. Los elementos en línea están contenidos dentro de elementos de bloque y los elementos de bloque ocupan todo el ancho de la página y la altura necesaria para mostrar todo su contenido.

El esquema flotante es el que permite que un elemento no ocupe todo el ancho de la pantalla y se muestre a la izquierda o a la derecha de la página con otros elementos a su lado. En el esquema de posicionamiento flotante, los elementos se muestran en la pantalla en el mismo orden en que se encuentran en el código fuente, pero pueden estar desplazados a derecha a izquierda. Para asignar el esquema de posicionamiento flotante a un elemento se utiliza la propiedad float.

El esquema absoluto permite asignar cualquier posición a un elemento en la página (no solamente a izquierda o derecha como en el esquema flotante). En el esquema de posicionamiento absoluto, los elementos pueden mostrarse en la pantalla en un orden diferente al que se tienen en el código fuente, e incluso superponerse. Para asignar el esquema de posicionamiento absoluto a un elemento se utiliza la propiedad position.

La imagen siguiente muestra cómo podrían mostrarse cuatro bloques de una página que en el código fuente estuvieran seguidos. En el esquema normal los bloques se muestran en el mismo orden que en el código fuente; en el esquema

flotante se puede colocar un bloque a la izquierda (el bloque 2) y el bloque siguiente se puede colocar a la derecha del flotante; en el esquema absoluto los bloques se pueden colocar en cualquier orden y posición.



La propiedad float

Las imágenes son elementos en línea, es decir, que se insertan como si fueran caracteres, formando parte del párrafo o del elemento de bloque en el que se insertan. La altura de la línea en la que está insertado el elemento aumenta lo necesario para poder alojar la imagen, como muestra el siguiente ejemplo, en el que la hoja de estilo no contiene ninguna propiedad relacionada con la imagen:

```
img {
}
```

La imagen se comporta igual que si fuera texto, ocupando el espacio de la primera línea y haciendo que ésta aumente en altura.

Podemos hacer uso de la propiedad `float` para conseguir que la imagen “flote” a la izquierda del texto:

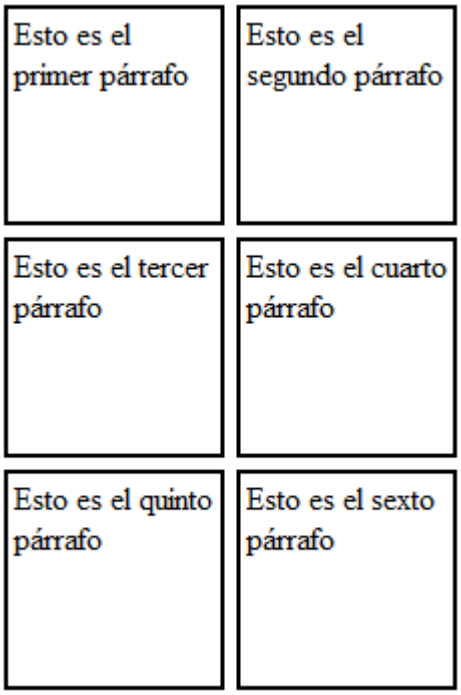
```
img {
  float: left;
}
```

Ahora el texto se comporta de forma normal, pero la imagen “flota” a la izquierda del mismo, consiguiendo este efecto. Podríamos hacer lo mismo con `float:right` y la imagen “flotaría” a la derecha, mientras que el texto ocuparía el espacio a su izquierda.

Si queremos evitar que haya elementos a la izquierda y/o derecha de un elemento, debemos usar la propiedad `clear`, que viene a ser “la contraria a float”.

Cuando usamos `float: left`, los elementos intentan posicionarse lo más arriba y a la izquierda posible, mientras quede espacio horizontal suficiente para colocarse. Si no tienen espacio, se irán hacia abajo hasta que encuentren un espacio libre.

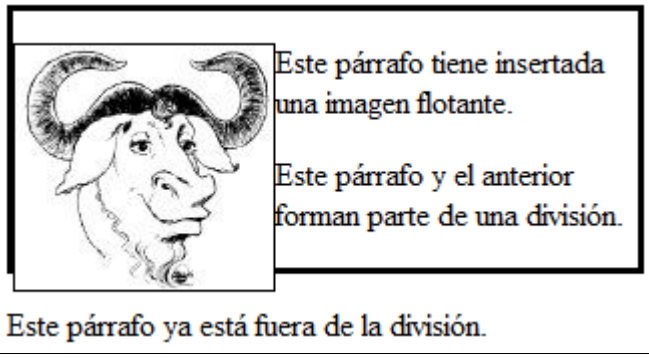
```
p {
 border: black 2px solid;
 float: left;
 height: 100px;
 margin: 3px;
 padding: 3px;
 width: 100px;
}
```



Tamaño de los elementos que contienen elementos flotantes

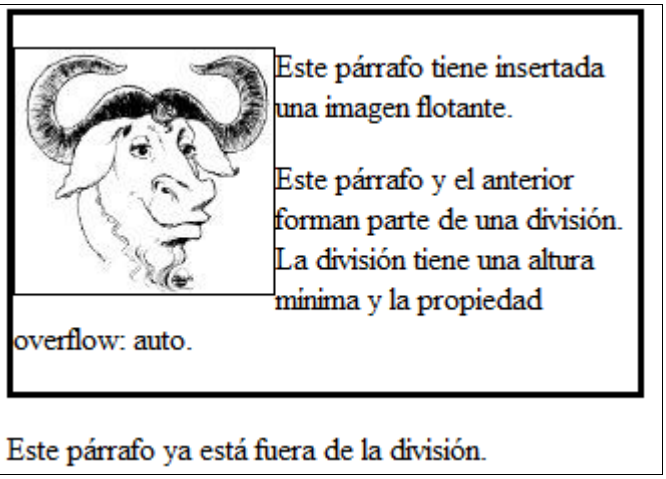
Los elementos flotantes no se tienen en cuenta al calcular el tamaño de los elementos que los contienen. Por ejemplo, si una imagen flotante forma parte de una división con borde, la imagen puede "salirse" del borde, como se ve en siguiente ejemplo:

```
div {
  border: black 3px solid;
}

img {
  float: left;
}
```



Si se quiere que la división incluya la imagen, la solución más sencilla que funciona en versiones actuales de Firefox, Chrome e Internet Explorer sería establecer la propiedad: `overflow: auto` en la división.

```
div {
  border: black 3px solid;
  overflow: auto;
}

img {
  float: left;
}
```



Posicionamiento estático

Cuando se establece la propiedad position como static, el elemento se sitúa en la posición que le corresponde de acuerdo con el flujo normal del documento. Es decir, el posicionamiento absoluto estático es equivalente al esquema normal.

```
#div1{
}

#div2{
}

#div3{
}
```

Posicionamiento relativo

Cuando se establece la propiedad position como relative, el navegador reserva el espacio en la página que tendría el elemento de acuerdo con el flujo normal del documento, pero el elemento se puede desplazar con respecto a esa posición. Es decir, el "hueco" de la posición original se mantiene, aunque el elemento se haya desplazado, como muestran los siguientes ejemplos.

El desplazamiento relativo se establece mediante las propiedades top, bottom, left y right.

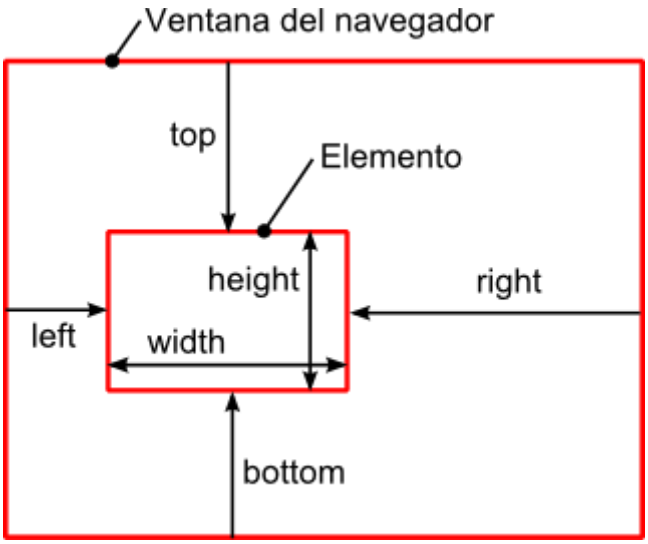
El desplazamiento vertical se puede establecer tanto con la propiedad top como con la propiedad bottom.

```
#div1{
}

#div2{
        position: relative;
        left: 10px;
        bottom: 15px;
}

#div3{
}
```

Posicionamiento fijo y absoluto

Cuando se establece la propiedad position como fixed o como absolute, el elemento se puede situar en cualquier posición de la pantalla.

La posición del elemento se establece mediante las propiedades top, bottom, left y right (aunque también se pueden utilizar las propiedades width y height), como indica la imagen siguiente:



Cuando se establece la propiedad position como fixed, el elemento se sitúa en la posición indicada y no se puede desplazar. Aunque la página sea larga y se pueda desplazar verticalmente, el elemento fijo estará siempre en la misma posición de la pantalla.

Cuando se establece la propiedad position como absolute, el elemento se sitúa en la posición exacta que se le indique. Pero si la página es larga y se puede desplazar verticalmente, el elemento absoluto se desplazará con el resto de elementos.

El navegador no tiene en cuenta la posición de los elementos con posicionamiento fijo o absoluto al mostrar el resto de elementos de la página, es decir, que los elementos con este posicionamiento no reservan espacio.

Profundidad: z-index

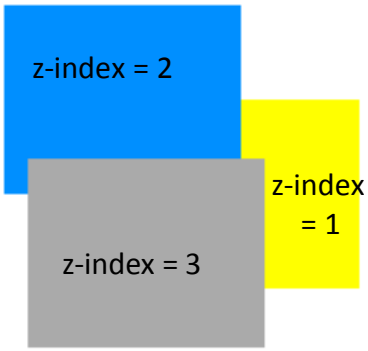
Al utilizar posicionamiento relativo, fijo o absoluto, dos o más elementos se pueden solapar. La propiedad z-index, que debe ser un número entero (positivo, cero o negativo), permite establecer qué elemento se muestra encima, pero también influye si los elementos tienen o no posicionamiento y el orden en que aparecen los elementos en el código fuente.

Las reglas a aplicar son las siguientes:

- Los elementos sin propiedad z-index se tratan como si tuviera un z-index igual a 0.
- Los elementos sin posicionamiento se tratan como si tuvieran un z-index igual a 0 (aunque tengan propiedad z-index).
- Los elementos con mayor z-index se colocan sobre los elementos con menor z-index,
- Si los z-index son iguales, los elementos con posicionamiento se colocan sobre los elementos sin posicionamiento
- Si los z-index son iguales y los elementos tienen posicionamiento, los elementos que aparecen después en el código fuente se colocan encima de los elementos que aparecen antes en el código fuente

En este ejemplo podemos ver el uso de la propiedad z-index:

```
#div1{
        position: relative;
        z-index: 2;
}

#div2{
        position: relative;
        left: 10px;
        bottom: 15px;
        z-index: 3;
}

#div3{
        position: relative;
        left: 50px;
        bottom: 120px;
        z-index: 1;
}
```



Los elementos con z-index mayor se ven sobre los que tienen un z-index menor.

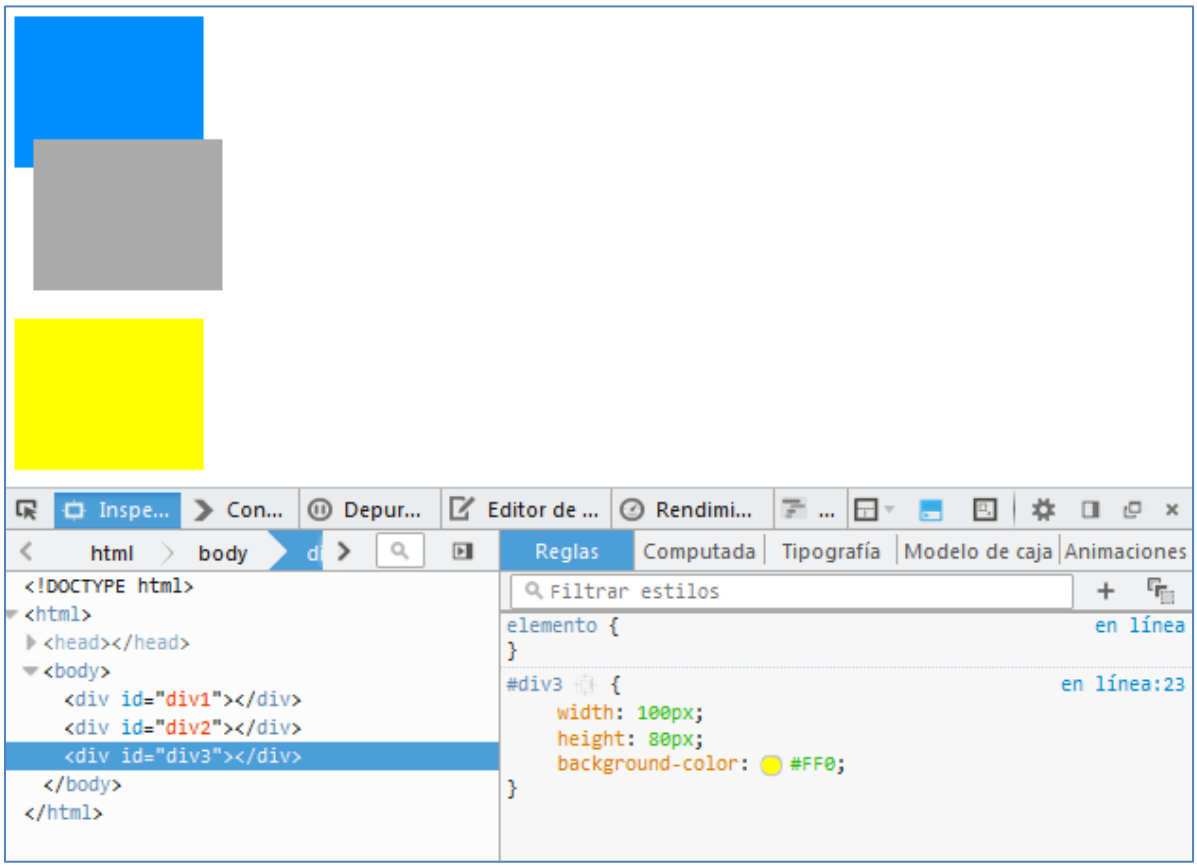
Inspector CSS

Cuando probamos la web que estamos diseñando, es común encontrar errores o cosas que no se comportan como esperamos. En estos casos, podemos hacer uso de una herramienta que incorporan la mayoría de navegadores modernos: el inspector web. Pulsando la tecla F12 del teclado, se nos mostrará una herramienta con una serie de opciones: por un lado, tenemos el código HTML de nuestra página, donde podremos seleccionar elementos de forma individual y los veremos resaltados en la vista del navegador. También podemos hacer click derecho sobre un elemento de la página y marcar la opción “Inspeccionar elemento”. En la parte derecha encontramos las reglas CSS que se aplican al elemento seleccionado, así como las que hereda de otros elementos superiores y de la hoja de estilos por defecto del navegador.

Esta herramienta es muy útil ya que nos permite, no sólo ver el código HTML y CSS de la página, sino realizar modificaciones en vivo y desactivar selectivamente

reglas CSS. Estos cambios son temporales y, cuando cerremos el navegador o recarguemos la página, se perderán.

Dentro del inspector disponemos de algunas opciones interesantes, como por ejemplo la "Vista de diseño adaptable", que nos permite probar cómo se vería nuestra página en diferentes resoluciones de distintos dispositivos (teléfonos móviles, tablets, etc.)



Validación de CSS

Al igual que ocurría con HTML, nuestro código CSS debe ser validado antes de ser publicado en un sitio web de Internet, para garantizar que no se han cometido errores sintácticos y que las propiedades usadas se ajustan al estándar. Si bien existen multitud de validadores no oficiales de CSS, en nuestro caso haremos uso del oficial, presente en la web del consorcio W3C:

https://jigsaw.w3.org/css-validator/

Optimización de CSS

Conforme vamos construyendo el CSS de nuestra página web, es recomendable seguir algunos consejos de cara a optimizar la cantidad de código producida. Algunos consejos que se suelen seguir son los siguientes:

- **Agrupar estilos similares**, incluyendo en un único selector todas las propiedades que tengan en común varios elementos. Ejemplo:

```
body{
  background-color: #CCCCCC;
}

.titulares{
  background-color: #CCCCCC;
  font-size: 20px;
}

#cabecera{
  background-color: #CCCCCC;
}
```

El bloque de CSS anterior, podría agruparse de la siguiente manera:

```
body,.titulares,#cabecera{
  background-color: #CCCCCC;
}

.titulares{
  font-size: 20px;
}
```

- **Omitir ceros y unidades de medidas innecesarias**, como cuando decimos "`width: 0;`" en lugar de "`width: 0px;`" o poner "`top: .12em;`" en lugar de "`top: 0.12em;`".

- **Eliminar clases e identificadores que no se utilicen**. Es muy común crear reglas CSS usando identificadores y clases que luego finalmente no se acaban usando en el HTML y quedan huérfanas, por lo que podrían ser eliminadas. Esta labor se puede realizar manualmente o con la ayuda de programas y extensiones de navegador apropiadas, como por ejemplo los complementos para Firefox "CSS Usage" o "Dust-Me Selectors".

Bibliografía

Este documento es obra de Javier Gil Motos y ha sido construido, en parte, a partir de las siguientes fuentes de documentación:

- Referencia oficial del W3C - http://www.w3.org/Style/CSS/
- Portal MCLIBRE – Material Curricular Libre - http://www.mclibre.org/