UD1: Aplicaciones web y lenguajes de marcas: HTML y CSS

ACTIVIDADE 3: Aplicando estilo con CSS.

Índice

1. CSS 3

1.1 Donde se escribe el código CSS 3
1.2 La sintaxis CSS 4
1.3 Indicación de URL 5
1.4 Comentarios 6
1.5 Orden de aplicación de los estilos 6
1.5.1 Herencias 6
1.6 Tipos de selectores..... 7
1.7 Propiedades a aplicar con las hojas de estilo 11
1.7.1 Propiedades de las fuentes..... 11
1.7.2 Propiedades de texto 14
1.7.3 Unidades e medidas 17
1.8 Aplicando estilos a listas..... 18
1.9 Enlaces..... 20

1. CSS

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).

La filosofía de las CSS (hojas de estilo en cascada) responde a la idea de separar *al máximo forma y contenido*. Las páginas tendrán únicamente etiquetas html con información de la estructura y contenido del documento (párrafos, cabeceras, listas, etc.) sin ninguna información sobre la apariencia del mismo. Esta apariencia (la forma en que ese documento debe ser visualizado), se indica con una serie de reglas que se definen separadamente (en una sección separada de la página o en un archivo separado), de forma que podemos cambiar la forma en que el documento se visualiza sin tocar ni una sola línea de contenido ni cambiar las etiquetas html, simplemente introduciendo unos pocos cambios en la definición de estilo.

Un único archivo css puede controlar la presentación de un número ilimitado de páginas, permitiendo mantener un aspecto homogéneo en todas ellas.

Las hojas de estilo en cascada responden a un standard definido por la organización W3C, y este standard debe funcionar correctamente con las versiones más recientes de los navegadores gráficos más populares. Cuando creamos nuestra página utilizando características de CSS3 podemos encontrarnos con que una persona que no utiliza una versión actualizada no pueda mostrarlas en su totalidad.

Por qué el nombre de Hojas de Estilo en Cascada?

La organización W3C, al usar la expresión de *hojas de estilo en cascada*, alude a la posibilidad de introducir en varios sitios *estilos* que controlan la apariencia de la página. Además, también debemos tener en cuenta que a un elemento que está contenido en otro (por ejemplo un párrafo contenido dentro de un elemento div) se le aplican las propiedades definidas para los dos elementos.

Todo esto hace que los estilos se pueden asignar para un mismo elemento desde distintos orígenes simultáneamente, por lo que estas reglas pueden entrar en *conflicto*. El standard CSS establece los criterios ('orden en cascada') que deciden que estilo se aplica preferentemente, lo que permite especificar estilos generales para el conjunto de páginas y subestilos de aplicación a elementos o secciones determinadas.

1.1 Donde se escribe el código CSS

Las propiedades pueden aparecer en tres lugares distintos:

- *Directamente en las etiquetas*: Añadiendo el atributo `style` al elemento html al que queremos dar formato. El valor del atributo consiste en dos partes separadas por dos puntos. La primera parte es la propiedad del estilo que queremos aplicar, mientras que la segunda es el valor que se le asigna a dicha propiedad. En el siguiente ejemplo vemos cómo aplicar estilo al color del texto de la etiqueta h1:

  ```
  <h1 style="color: blue">Texto</h1>
  ```

  Si aplicamos distintos estilos al mismo elemento, estos deben separarse usando punto y coma:

  ```
  <h1 style="color: red; background-color: black">Texto</h1>
  ```

- *En la cabecera* (`head`) del archivo HTML: Esto se conoce como hojas de estilos internas. La sintaxis consiste en insertar el código CSS directamente en una etiqueta `style` dentro del elemento `head` y aquí también debemos indicar a qué elementos queremos aplicar los estilos Así, cada componente estará compuesto de tres partes:
  - El nombre del tipo de elemento al que vamos a aplicar el estilo.
  - El nombre de la propiedad CSS que queremos cambiar
  - El valor que queremos aplicar a esta propiedad

  La propiedad y su valor aparecen entre llaves separadas por dos puntos como sigue:

  ```
  elemento {propiedade: valor}
  ```

  Si queremos aplicar más de un estilo a un tipo de elemento particular debemos separarlo por punto y coma. Por exemplo:

  ```
  <style type="text/css">
      h1 {
          color:red;
          text-align: center
         }
  </ style >
  ```

- *En un archivo distinto*, lo que se conoce como hojas de estilo externas. La mayor parte de las veces querremos aplicar los mismos estilos en distintas zonas de nuestra aplicación Web, para que así todas las páginas presenten el mismo aspecto.

  Una hoja de estilo externa se vincula al documento HTML utilizando o elemento `link` (en la del documento HTML). El nombre y la ubicación de la hoja de estilos se indica en el atributo `href`, y la relación entre la página HTML y la hoja de estilos mediante el atributo `rel` (el valor de este atributo debe ser `stylesheet`). También tenemos el atributo `type` para indicar el tipo de documento vinculado. Para una hoja de estilos el valor debe ser `text/css`.

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

1.2 La sintaxis CSS

CSS es un lenguaje declarativo, lo que hace que su sintaxis sea bastante fácil de usar y de comprender. Además, dispone de un potente sistema de recuperación de errores que permite cometer fallos sin que esto afecte a todo el código: simplemente, las declaraciones que no son comprendidas son ignoradas.

Vamos a ver más en detalle la sintaxis CSS. Una ***declaración CSS*** se compone de:

- *Propiedades*: que identifican la característica que se quiere cambiar (p.e: color de fondo)
- *Valores*: que determinan cómo se quiere cambiar la característica a la que se aplica (p.e. color verde)

Los pares propiedad y valor se separan por dos puntos. Hay más de 300 propiedades diferentes en CSS y casi infinitos valores diferentes. Cada propiedad tiene una lista específica de valores que puede adoptar.

Las declaraciones se agrupan en bloques que empiezan y terminan con llaves { }. Cada declaración contenida en un *bloque declarativo* deber estar separada por un punto y coma (;), sino el código no funcionará (o producirá resultados inesperados). La última declaración de un bloque no necesita llevar (;), aunque se considera buena práctica añadirlo, para prevenir olvidos cuando se añaden más declaraciones al bloque.

Un ejemplo de bloque declarativo sería:

```
{
  color: navy;
  text-decoration: underline;
}
```

Selectores y reglas CSS

Ahora toca indicar a qué elementos afecta nuestro bloque declarativo. Para ello ponemos un prefijo a modo de *selector*, que será el que identificará algunos elementos de la página. Un selector más un bloque declarativo reciben el nombre de ***regla CSS***.

Los selectores (filtros) pueden llegar a ser bastante complicados — se puede establecer una regla que afecte a múltiples elementos separándolos en el selector por comas, y estos pueden ser identificados en conjunto, por ejemplo: Seleccionar los elementos de la clase `p`, pero solo los que estén dentro de un `<article>`, y solo cuando pase por encima el puntero del ratón.

```
h1, div p, #cont:hover {
  background-color: #F0F0F0;
  background-style: none;
}
```

Además, un elemento puede estar afectado por varios selectores, de tal manera que una propiedad podría ser asignada más de una vez al mismo elemento. Más adelante veremos cómo se sabe cuál va a tener preferencia.

Reglas-@

Se usan para definir metadatos en CSS , información condicional u otra información descriptiva. Comienzan con el símbolo (`@`), seguido del identificador del tipo de regla, luego un bloque sintáctico correspondiente y terminará con un (;). Cada tipo de Regla-@ definido por un identificador dispondrá de su propia sintaxis y semántica internas. Algunas son:

`@charset y @import` → Metadatos

Por ejemplo: `@import 'estilos.css'` importará un archivo externo.

`@media` → sólo se aplicará si el dispositivo que ejecuta el navegador soporta la característica que se indica. Por ejemplo:

```
@media (min-width: 801px) {
  body {
    margin: 0 auto;
    width: 800px;
  }
}
```

Sólo se aplica si el ancho de página es mayor de 800 pixeles.

`@supports` → sólo se aplica si el navegador soporta la característica mencionada

1.3 Indicación de URL

Muchas propiedades de CSS necesitan poder indicar direcciones URL a recursos necesarios para las páginas (direcciones de enlaces, imágenes, etc.). Para ello se utiliza una función llamada URL a la que (entre paréntesis) se le indica la dirección URL. Por ejemplo, para cambiar la imagen de fondo:

```
body{
  background-image: url(imágenes/fondo1.jpg);
}
```

1.4 Comentarios

Al igual que en HTML, se aconseja incluir comentarios en el CSS para ayudar a entender el funcionamiento del código al tiempo de haberlo programado, además de ayudar a otros a entenderlo. Los comentarios son prácticos cuando se está probando ciertas partes del código, por ejemplo para averiguar que parte del código está causando el error encontrado.

Los comentarios en CSS comienzan con /* y acaban con */.

```
/* Esto es un comentario */
```

1.5 Orden de aplicación de los estilos

1. Primero se aplican los estilos del navegador. Es decir el formato predefinido del navegador. Todos los navegadores poseen un estilo predefinido, que dicta con que tamaño por defecto se muestra el texto, los colores, el tipo de letra,… Estos estilos son los que se ejecutan en primer lugar. Pero cualquier estilo definido fuera del navegador, tiene preferencia.
2. Después se aplican los estilos externos (los que se incorporan con la etiqueta `<link>`).
3. Después los que proceden de la etiqueta `<style>`.
4. Después los que se definan internamente en el elemento (mediante el atributo `style`).

En caso de dos estilos referidos al mismo elemento y definidos en el mismo ámbito (por ejemplo ambos procedentes de archivos externos e incluidos con el elemento link) tiene preferencia el último que se utilice en el código (es decir ganan los estilos del segundo link). No obstante se puede alterar la preferencia utilizando una palabra clave:

`!important` Los estilos marcados con ella tienen preferencia sobre cualquier otro.

Ejemplo:

```
p{
    color: green  !important;
}
```

El color verde para los párrafos tendrá preferencia sobre cualquier redefinición de estilos sobre el elemento `p`.

1.5.1 Herencias

Hay que tener en cuenta que hay etiquetas que son padre de otras, esto es, etiquetas que contienen a otras. Por ejemplo, en:

```
<p>Asignatura: <em>Lenguaje de marcas</em></p>
```

La etiqueta `p` es padre de la etiqueta `em` (`em` está dentro de `p`). Esto hace que `em` herede todo el estilo que posea `p` y además añada el suyo propio. Por ejemplo, si hemos definido:

```
p{ color:blue;
    font-size: 12pt
}
em{
    font-size: 14pt;
}
```

Lenguaje de marcas tendrá color azul pero tamaño 14.

1.6 Tipos de selectores

Existen los siguientes tipos de selectores:

- *Selectores simples*: Seleccionan los elementos por el nombre del tipo de elemento, class, o su id.
- *Selectores de atributos*: Seleccionan los elementos por los valores de sus atributos.
- *Pseudo-clases*: Seleccionan los elementos por el estado en que se encuentran, cómo por ejemplo, al pasar el ratón encima de un elemento, o por ser el primer hijo de su padre en el árbol DOM.
- *Pseudo-elementos*: Selecciona los elementos por su situación en relación a otro elemento, por ejemplo: la primera palabra de cada párrafo, o el contenido que se encuentra justo después de un elemento.
- *Combinaciones*: No son en sí mismos selectores, sino formas de combinar dos o más selectores de forma práctica para una selección especial. Por ejemplo, se pueden seleccionar párrafos que sean descendientes de divs, o párrafos situados justo después de títulos.
- *Selectores múltiples*: Tampoco son selectores en sí mismos; podemos agrupar múltiples selectores en la misma regla CSS separados por comas, para aplicarlos a una de las declaraciones o a todos los elementos seleccionados por estos selectores

Vamos a ver los más usados además del selector de elementos.

Selector universal

Es un selector que permite aplicar un estilo a todas las etiquetas. Se indica mediante un *****.

```
*{
    color: black;
}
```

No se suele utilizar de esa forma ya que es demasiado indiscriminada. Pero sí se utiliza combinada con otros tipos de elementos. Por ejemplo:

```
table * em {
    color: red;
}
```

Permite colorear en rojo el texto de los elementos `em` cuando esta etiqueta está dentro de una tabla.

Selectores de clases

Una de las primeras formas que tiene CSS para diferenciar elementos del mismo tipo (por ejemplo un párrafo de otro) son las clases, que se asignan usando el atributo HTML `class`.

El valor del atributo `class` (el nombre de la clase) sólo pueden contener caracteres ingleses, números, guiones y subrayados, aunque el primer carácter no puede ser ni un número ni un guion.

La regla con el selector `class`, se construye con el carácter punto.

A un elemento con clase se le aplica tanto la regla de la clase como la regla de la etiqueta general (esta es una característica del mecanismo de cascada de las hojas de estilo). Por ejemplo:

```
<p>Aunque ahora es muy querida no fue profeta en su tierra. </p>
<p class="entrevista">¿Se siente valorada en su país?</p>
```

Con la siguiente hoja de estilos, se vería:

```
p {
  font-style: oblique;
}

p.entrevista {
  color: blue;
}
```

Aunque ahora es muy querida no fue profeta en su tierra.

¿Se siente valorada en su país?

Un elemento puede tener *varios nombres de clases separados por un espacio en blanco*. El orden en que se escriban los nombres de las clases en el atributo `class` no es importante, aunque sí que es importante el orden de las reglas en la hoja de estilo. Si la misma propiedad está definida en varias clases y a un elemento tiene asignado varias clases, se aplica la definición de la clase que aparece después en la hoja de estilo (esta es una característica del mecanismo de cascada de las hojas de estilo). Por ejemplo, si al siguiente html:

```
<ul>
  <li class="primero hecho">Crear una página HTML</li>
  <li class="segundo hecho">Crear la hoja de estilos CSS</li>
  <li class="tercero">Vincular la hoja de estilos al html</li>
</ul>
```

Le aplicamos la siguiente hoja de estilos:

```
/* El elemento con la clase "primero" está en negrita */
.primero {
  font-weight: bold;
}

/* Todos los elementos con la clase "hecho" están tachados */
.hecho {
  text-decoration: line-through;
}
```

Veríamos lo siguiente:

- ~~Crear una pÃ¡gina HTML~~
- ~~Crear la hoja de estilos CSS~~
- Vincular la hoja de estilos al html

En un mismo documento varios elementos pueden compartir el mismo valor de clase. En este caso podemos usar el selector universal cuando todos los elementos comparten todas las propiedades. Por ejemplo,

```
<p class="resaltado"> El autor presenta su nueva ópera sobre el
    artista </p>
<p> Una ópera sobre el pintor en la que le dibuja
    como <span class="resaltado">el destructor de la
    pintura</span>. </p>
```

con la siguiente hoja de estilo se vería

```
.resaltado {
  color: red;
  font-style: oblique;
}
```

El autor presenta su nueva ópera sobre el artista

Una ópera sobre el pintor en la que le dibuja como *el destructor de la pintura*.

Donde `.resaltado` es equivalente a usar la clase universal `*.resaltado`

Selectores ID

El atributo `id` define un identificador único (ID) el cual no debe repetirse en todo el documento. Su propósito es identificar el elemento al vincularlo (usando un identificador de fragmento), en scripts u hojas de estilo (con CSS).

Hay que tener en cuenta que el atributo `id` no se puede repetir, es decir, que no puede haber en una página web dos elementos con el mismo valor del atributo `id` (independientemente de que los elementos tengan etiquetas iguales o distintas). En el caso de que la misma hoja de estilo se enlace desde varias páginas web, en cada una de esas páginas sí que puede haber un elemento (del mismo tipo o distinto en cada página) con el mismo valor del atributo `id`.

La regla con el selector id, se construye con el carácter #.

Por ejemplo, podríamos usarlo para identificar un párrafo que es resumen del texto y así aplicarle un estilo diferenciado:

```
<p id="resumen">Este es el resumen del texto</p>
```

Y en `css`:

```
.#resumen { color: blue; }
```

Combinaciones

En CSS, podemos combinar varios selectores juntos y con ello seleccionar elementos contenidos en otros elementos, o elementos adyacentes a otros. Disponemos de cuatro tipos:

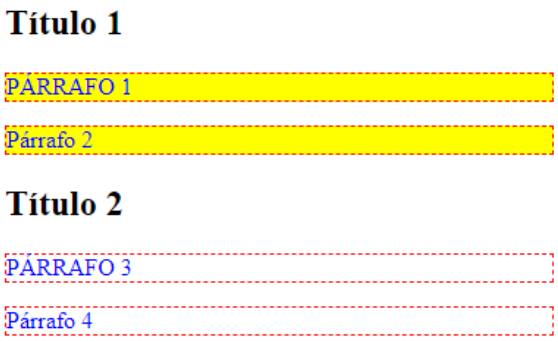
- El selector *descendiente*: __`(espacio)`__ permite seleccionar un elemento anidado en alguna parte dentro de otro elemento (no tiene por qué ser un hijo; puede ser un nieto, por ejemplo)
- El selector *hijo*: __>__ permite seleccionar un elemento que es hijo directo de otro elemento.
- El selector de *adyacentes*: __+__ permite seleccionar un elemento que va justo después de otro elemento en el mismo nivel jerárquico.
- El selector *hermano en general*: __~__ permite seleccionar cualquier elemento hermano de otro (por ejemplo en el mismo nivel jerárquico, pero no necesariamente adyacente a él).

Vamos a ver un ejemplo:

```
<section>
  <h1>Título 1</h1>
  <p>Párrafo 1</p>
  <p>Párrafo 2</p>
  <div>
    <h1>Título 2</h1>
    <p>Párrafo 3</p>
    <p>Párrafo 4</p>
  </div>
</section>
```

Con el siguiente css se vería:

```
section p {
  color: blue;
}
section > p {
  background-color: yellow;
}
h1 + p {
  text-transform: uppercase;
}
h1 ~ p {
  border: 1px dashed red;
}
```



Esto es así porque:

`section p` → Selecciona todos los párrafos descendientes del elemento `section`.

`section > p` → Selecciona sólo los dos primeros párrafos porque estos son los hijos directos del elemento `section`.

`h1 + p` → Selecciona sólo los párrafos que están justo después del elemento `h1` en el mismo nivel jerárquico (en este caso el primer y tercer párrafos).

`h1 ~ p` → Selecciona cualquier párrafo en el mismo nivel jerárquico que los elementos h1

Selectores de atributos

Los selectores de atributos realizan la selección de los elementos afectados por la declaración en función de los atributos y sus valores. Su sintaxis consiste en la inclusión del nombre del atributo entre corchetes seguido opcionalmente de la condición respecto del valor del atributo.

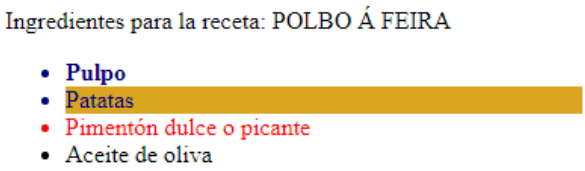
- `[attr]` : Selecciona todos los elementos que contengan el atributo `attr`, sin importar el valor que tenga.
- `elem[attr]` : Selecciona los elementos de tipo `elem` que tengan el atributo `attr`, sin importar el valor que tenga.
- `[attr=val]` : Selecciona los elementos con el atributo `attr`, pero solo aquellos cuyo valor coincida con `val`.
- `[attr~=val]`: Este selector afectará a los elementos con el atributo `attr`, pero solo si el valor `val` está contenido en la lista de valores (separados por espacios) incluidos en el valor de `attr`, por ejemplo una de las clases contenida en una lista de clases (separadas por espacios).
- `[attr^=val]`: Selecciona todos los elementos cuyo atributo `attr` comienza por el valor `val`.
- `[attr|=val]`: Selecciona todos los elementos con el atributo `attr` cuyo valor sea exactamente `val` o empieza por `val-` (el se usa para manejar códigos de lenguaje de programación).
- `[attr$=val]`: Selecciona los elementos cuyo atributo `attr` termina por `val`. Este selector podríamos usarlo por ejemplo para asignar una imagen determinada dependiendo de la extensión, pdf, rar, txt, doc, jpeg.
- `[attr*=val]`: Selecciona todos los elementos cuyo atributo `attr` contiene la cadena `val` (al contrario que `[attr~=val]`, este selector no considera los espacios como separador de valores sino como parte del valor del atributo).
- `[attr operator value i]`:Agregar una `i` (o `I`) antes del corchete de cierre hace que el valor sea comparado sin distinguir entre mayúsculas y minúsculas (para caracteres dentro del rango ASCII).

Vamos a ver un ejemplo. Tenemos el siguiente html:

```
<p>Ingredientes para la receta: <span lang="gl-GL">Polbo á
feira</span></p>
<ul>
  <li cantidad="2kg" tipo="pescado">Pulpo</li>
  <li cantidad="4 unidades" tipo="vegetal">Patatas</li>
  <li cantidad="al gusto">Pimentón dulce o picante</li>
  <li>Aceite de oliva</li>
</ul>
```

Si le aplicamos el siguiente css se vería:

```
[cantidad] {
  color: navy;
  }
[tipo="vegetal"] {
  background-color: golden;
 }
[cantidad*="gusto"] {
  color: red;
  }
[lang|=gl] {
  text-transform: uppercase;
  }
[cantidad$="kg"] {
  font-weight: bold;
}
```



1.7 Propiedades a aplicar con las hojas de estilo

1.7.1 Propiedades de las fuentes

CSS1 define las propiedades `font-family`, `font-style`, `font-variant`, `font-weight`, `font-size`, e `font`.

La posibilidad de manejar las fuentes es una de las más apreciadas en las hojas de estilo pero debemos tener en cuenta que existen diferencias en el modo en que los navegadores interpretan las reglas de estilo. Además, hay muchas diferencias entre los valores que admite un tipo de fuente u otro. Por ejemplo: veremos que se poden especificar hasta nove valores de negrita o bold, pero muchas fuentes están diseñadas de tal forma que sólo pueden ser visualizadas en dos valores: normal o negrita.

font-family

La propiedad `font-family` permite establecer el tipo de letra (fuente) del elemento. Se puede escribir el nombre concreto de una fuente (escribiéndolo entre comillas si el nombre contiene espacios), pero hay que tener en cuenta que si la persona que ve la página no tiene esa fuente instalada en su ordenador (o la tiene, pero con otro nombre), su navegador utilizará una fuente distinta. Para evitar este problema se puede utilizar uno de los nombres genéricos (serif, sans-serif, cursive, fantasy o monospace). Los navegadores tienen asociados a cada nombre genérico una fuente que sí que está instalada en el ordenador.

Se puede poner una lista de tipos de letra específicos o genéricos separados por comas. El navegador aplicará la primera fuente de la lista que esté instalada en el ordenador. Los nombres de las fuentes deben escribirse entre comillas si contienen espacios.

```
body { font-family: "Tahoma", "Corbel", sans-serif }
```

Lo normal es indicar en primer lugar los tipos de letra que elige el autor y dejar como úl-

tima alternativa una familia genérica. De este modo aseguras que el documento se visualizará con una fuente semejante (de la misma familia) que la designada.

Hay cinco familias genéricas de fuentes. A continuación se enumeran los nombres de estas familias, y entre paréntesis un ejemplo de fuente perteneciente a la misma: serif (Times New Roman), sans-serif (Arial, Helvetica), cursive (Comic Sans\), fantasy (Ransom) e monospace (Courier).

font-style

Una forma de variar el tipo de letra es inclinar las letras. En Tipografía se distinguen dos formas de inclinación:

- Oblicua: la forma de la letra no cambia, simplemente se inclina
- itálica: además de inclinarse, la forma de algunas letras cambia

La propiedad `font-style` permite elegir como valor `normal`, `italic` o `oblique`. La diferencia entre `italic` y `oblique` pasará desapercibida en la mayor parte de las fuentes.

```
h1, h2, h3 { font-style: oblique; }
```

font-variant (Versalitas)

Puede tomar los valores: `normal` o `small-caps`. Estas últimas son tipos en las que todos los caracteres están en mayúsculas, aunque el tamaño de las minúsculas tendrá la altura de la letra x del tipo (aunque no siempre es así). Si el tipo de fuente seleccionado non tiene versión `smalls-caps`, el navegador visualizará el texto en mayúsculas ordinarias.

```
h3 { font-variant: small-caps; }
```

que se vería:

ESTO DEBERÍA VERSE EN SMALL-CAPS e esto EN MAIÚSCULAS NORMALES.

font-weight

Puede tomar los valores: `normal`, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, `900`. El valor `font-weight` determina el grosor de la fuente. Los valores 100 a 900 indican una secuencia jerárquica, donde cada número indica un grosor secuencialmente más grueso que su predecesor. El valor `normal` equivale a 400. `Bold` sería 700.

```
p { font-weight: normal; } /* 400 */
h1 { font-weight: 700; } /* bold */
```

Este texto tiene un valor 200
Este es un texto normal;
Este es texto weight 700 o 'bold';
Este tiene un valor 900.

Los anteriores son valores absolutos; existen dos valores relativos: `bolder` y `lighter` que seleccionan grosores de fuente mayores o menores en relación al grosor "heredado" del elemento del que dependen.

No existe certeza sobre el grosor exacto con el que un navegador determinado puede visualizar una fuente. Además, algunas fuentes sólo admiten dos valores (`normal` y `bold`).

font-size

La propiedad `font-size` determina el tamaño de la fuente. Los valores pueden especificarse:

- de forma absoluta
- de forma relativa
- en porcentaje
- expresando su unidad de medida

de forma absoluta

Al especificar el tamaño de fuente en valores absolutos, el autor de la página se remite a la tabla de tamaños de fuente predeterminados en el ordenador del usuario. Son valores admisibles `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, e `xx-large`.

`Medium` es el tamaño de fuente por defecto, y cada uno de los valores absolutos supone un incremento o diminución de tamaño en un valor constante que depende del navegador o de la versión de CSS empleada; Pero en cualquier caso, los diferentes valores absolutos siempre respetarán sus proporciones entre ellos. Una fuente `x-large` será siempre más grande que una fuente `large`. Un exemplo:

xx-large x-large large médium small x-small xx-small

de forma relativa

La segunda forma de especificar el tamaño de la fuente es usando valores relativos. Los valores admitidos son `smaller` y `larger`. Estos valores se interpretan con respecto al elemento padre y se toman el valor siguiente o anterior de la lista de tamaños absolutos.

en unidades

También se puede indicar el tamaño de la fuente con su valor en unidades: cm, in, pt, px. Hace años (antes de 2010) se desaconsejaba utilizar estas unidades de forma indiscriminada porque los navegadores no permitían reducir ni ampliar los elementos definidos mediante unidades absolutas, pero actualmente no existe esa limitación.

```
p { font-size: 16pt;}
```

en porcentajes o unidades relativas

Se puede definir el tamaño de cada elemento mediante porcentajes (o em, teniendo en cuenta que 1em = 100%), que se interpretan con respecto al tamaño base.

```
p { font-size: 90%;}
```

font

La propiedad compuesta `font` permite definir simultáneamente las propiedades relacionadas con el tipo de letra: `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height` y `font-family`.

`line-height`, que en realidad es una propiedad de texto alude al espacio entre líneas. No es necesario especificar valores para todas las propiedades, sólo para aquellas que se quieran cambiar, pero el orden tiene que ser estrictamente ese para que funcione correctamente. Las únicas propiedades que deben aparecer obligatoriamente son `font-size` y

`font-family`. Las propiedades grosor (`weight`), estilo (`style`) y `variant` se pueden especificar en cualquier orden; seguidas por el tamaño (`size`), seguidas por una barra o slash "/" y la altura de línea (`line-height`), seguidas por la familia (`family`). Por ejemplo:

```
p { font: italic bold 12pt/14pt serif; }
```

especifica párrafos en negrita (`bold`) e itálica (`italic`), fuente `serif` con un tamaño de 12 puntos y una distancia entre líneas (o altura de línea) de 14 puntos.

@font-face

Uno de los problemas más importantes sobre la tipografía es querer utilizar un tipo de letra sobre la que no hay seguridad que exista en el dispositivo de la persona que está viendo la página HTML.

CSS3 ha introducido el uso de esta directiva en el código CSS para precargar el tipo de letra necesario. Para ello deberemos utilizar el archivo que contiene el tipo de letra en sí. Dicho archivo puede ser ttf (letras de tipo True Type) u otf (Open Type) (Explorer utiliza el formato de letra eot) (Embedded Open Type). Por ejemplo supongamos que queremos utilizar el tipo de letra Artistik y disponemos (teniendo en cuenta que los tipos de letra pueden tener derechos exclusivos de uso, por lo que habrá que asegurarse de que uso es libre o bien pagar los derechos para su uso) de archivo Artistik.ttf que contiene el tipo de letra en sí. El código para que todos los párrafos la usen (con la seguridad de que sí saldría) es:

```
@font-face{
  font-family: artistik;
  src: url('Artistik.ttf'), url('Artistik.eot');
}
p {
  font-family: artistik;
}
```

En la parte `@font-face` hay que indicar al menos dos propiedades:

- `font-family`. Para dar nombre de familia a la letra que estamos importando. Ese nombre es el que luego utilizarán los selectores para elegir la letra que hemos cargado (en el ejemplo es lo que usa p para dar formato con esa letra a los párrafos).
- `src`. En la que se indica una o más URL (separadas por comas) que apuntan a archivos de letras que se cargarán en la página para que se vea correctamente la letra.

  Además, opcionalmente se puede usar:

  - `font-weight`. La propiedad ya vista anteriormente, que nos permitiría indicar si el tipo de letra es para negrita (bold) o para texto normal.
  - `font-style`. Ya vista anteriormente para indicar si la letra es cursiva o no.
  - `font-stretch`. Indica el grado de condensación de la letra (por defecto se toma normal).

1.7.2 Propiedades de texto

color

La propiedad color permite establecer el color del texto. Hay distintas maneras de indicar el color:

- **Notación hexadecimal**: Se trata de la notación más utilizada en CSS. Consiste en un valor hexadecimal compuesto por seis valores alfanuméricos (0…F), precedida del símbolo #. Las dos primeras cifras hexadecimales indican el nivel de rojo, las dos siguientes el nivel de verde y las dos últimas de azul; por ejemplo #FF0000 es el código del rojo puro, que se podría representar en forma abreviada como #F00.

Color	Código hexadecimal	Código mediante función RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

- **Mediante función RGB**. Consiste en usar el formato rgb(r,g,b) donde r es el nivel de rojo (entre 0 y 255), r es el nivel de rojo y b el de azul. `rgb(255,0,0)` → es el código del rojo puro.
- **Mediante RGB con porcentaje**: Funciona como la anterior, pero el nivel de rojo, verde y azul se indican con tanto por ciento. Ejemplo:

  `rgb(50%,25%,12%)` → el color es un rojo anaranjado
- **Mediante función RGB y transparencia**: Permite utilizar en la función RGB un cuarto parámetro que es un valor de transparencia (conocido como parámetro alpha). Este parámetro puede tener un valor entre 0 (transparencia total) y 1.0 (opacidad total). Es parte de CSS3 y sólo está disponible en versiones recientes del navegador. Por ejemplo

  `color: rgba(255,0,0,1)` → para el rojo.
- **Mediante función HSL:** Permite seleccionar colores utilizando tres valores:
  - -Tono (Hue). Un valor de 0 a 360 que indica el giro en la rueda de colores. Por ejemplo el cero es el rojo, el 120 el verde y el 240 el azul.
  - -Saturación. Un número del 0% al 100%, que indica el nivel de saturación. Más saturación indica un color más vivo. Una saturación del 0% significa pasar el color a escala de grises.
  - -Luminosidad. Un número del 0% al 100%. Indica el nivel de luminosidad del color: más luminoso significa más claridad para el color (0% significa negro).

  Este modo está disponible en los últimos navegadores compatibles con CS3. Es el modelo más parecido a la forma que tiene el ser humano de percibir el color. Ejemplo: `color: hsl(0%, 100%, 50%)` → para el rojo.
- **Mediante función HSLA**: Añade al modelo anterior un cuarto valor (un número decimal entre 0.0 y 1.0) que sirve para indicar transparencia (nivel alfa). Por ejemplo

  `color: hsla(0%, 100%, 50%, 0.5)`
- **Por el nombre**: Permite indicar el color por su nombre estándar (Nombres de colores SVG/X11)

background-color

Permite establecer el color de fondo del elemento.

El color de fondo de la página se establece en la etiqueta `body`.

El color de fondo de una etiqueta no se aplica solamente al texto, sino a la caja que contiene el texto (por eso el color de fondo se extiende hasta el extremo derecho de la línea).

espazado de palabras (word-spacing) e espazado de letras (letter-spacing)

Las propiedades `word-spacing` y `letter-spacing` pueden ser usadas para añadir espacio extra entre palabras o letras. Los valores que admiten las dos propiedades son `normal` o unidades de lonxitud. Los valores pueden ser negativos.

text-decoration

La propiedad `text-decoration` permite añadir a un bloque de texto distintos tipos de rayado (aunque si el elemento carece de texto, la propiedad no tiene efecto). Sus posibles valores son: subrayado (`underline`), sobrerayado (`overline`), tachado (`line-through`), o ninguno (`none`). Este último se usa para quitar el subrayado a los links.

vertical-align

Esta propiedad permite controlar la posición vertical del texto en relación al elemento que lo contiene. Es muy versátil porque permite tanto alinear en vertical un texto respecto, por ejemplo, a la celda de la tabla en la que se encuentre; como indicar superíndices y subíndices. Los posibles valores son:

- `top`: Arriba respecto al elemento más alto de la línea.
- `bottom`: Abajo respecto al elemento más alto de la línea.
- `text-top`: En la línea superior del texto.
- `text-bottom`: En la línea inferior del texto.
- `baseline`: coloca en liña o elemento co elemento parent.
- `middle`: Punto medio respecto a la altura del texto o contenedor en el que estemos.
- `Sub`: Subíndice.
- `super`: Superíndice.

text-transform

La propiedad `text-transform` convierte texto a mayúsculas (valor `uppercase`), minúsculas (valor `lowercase`), o convierte a mayúscula la primera letra de cada palabra (`capitalize`). El valor `none` visualiza el texto normalmente, ignorando cualquier otro valor `text-transform` heredado.

text-align

La propiedad `text-align` permite establecer la alineación horizontal de un bloque de texto. Los valores son alinear con margen izquierdo, derecho, justificado o centrado (`left, right, justify, center`).

text-indent (Sangría)

La propiedad `text-indent` se usa para tabular la primera línea dentro de un bloque. Su valor se puede establecer como cualquier otra longitud. Admite valores negativos para hacer una sangría francesa.

line-height

La propiedad `line-height` (que podía ir especificada dentro de la propiedad `font`) indica el interlineado, medida entre las bases de cada línea. La base de una línea es la señalada por la parte inferior de un tipo de letra "no descendente"; por ejemplo, una "a", y no una "p". Los valores son números (si indicamos `line-height`: 2 decimos al navegador que la distancia entre líneas es el valor `font-size` multiplicado por 2). También se puede establecer como cualquier otra longitud.

White-space (Espacios en blanco)

La propiedad `white-space` permite el control de los espacios dentro del elemento. En la siguiente tabla vemos cómo se comportan los distintos valores que puede tomar:

	normal	pre	nowrap	pre-wrap	pre-line
Contrae múltiples espacios en uno	SI	NO	SI	NO	SI
Ignorar los saltos de línea	SI	NO	SI	NO	NO
Hace ajuste de línea (pasa a la siguiente línea si el texto no cabe en una sola línea)	SI	NO	NO	SI	SI

direction

La propiedad direction permite establecer la dirección de escritura. Los posibles valores de direction son `ltr` (de izquierda a derecha) y `rtl` (de derecha a izquierda).

1.7.3 Unidades e medidas

Ya hemos visto que nos podemos referir a las medidas en lonxitud y porcentaje. En ocasiones, el valor de una propiedad se puede expresar en unidades de longitud. Por ejemplo, los valores de `font-size` y `text-indent`. Las unidades de longitud consisten en un número seguido de una unidad de medida (`cm`, `em`, `rem`, `in`, `pt`, `px`). Existen dos tipos de unidades: absolutas e relativas.

Unidades absolutas:

- polgadas (in). Unha polgada=2.54 cm.
- centímetros (cm).
- milímetros (mm)
- puntos (pt). Un punto=1/72 de polgada.
- picas (pc). Una pica=12 puntos

Las unidades absolutas no son las ideales para mostrar texto en pantalla, aunque son perfectas para definir fuentes de impresión

Unidades relativas:

- `em` : La unidad `em` es igual a la altura (font-size) de la letra del elemento en que se usa. Por ejemplo, si para un párrafo especificamos un indentado de `2em`, la sangría será igual a dos veces el tamaño de la letra de ese párrafo. Cuando empleamos la unidad de

medida `'em'` para especificar el tamaño de letra (font-size) entonces el valor de `'em'` viene dado por el tamaño de fuente de su elemento padre.

- `rem`: Igual que el anterior pero toma como referencia el tamaño del elemento raiz
- `ex`: A unidade ex é igual á altura da letra x minúscula
- `px`: En CSS3 se trabaja con el llamado pixel de referencia del dispositivo de visionado , que equivale a 1/96 de pulgada y que mejora su funcionamiento con respecto a versiones anteriores de CSS
- `vw`: 1% de la anchura del viewport (dispositivo de visionado).
- `vh`: 1% de la altura del viewport.
- `vmin`: Entre `vw` y `vh` toma el que tenga menor valor.
- `vmax`: Entre `vw` y `vh` toma el que tenga mayor valor.

Los ems (em) y exes (ex), tanto en uno como en otro caso, nos encontramos con referencias relativas, especialmente adecuadas para la asignación de fuentes en pantalla; los ems han ganado mucha popularidad y son una unidad muy recomendable. La unidad ex no está plenamente soportada por todos los navegadores, muchos de los cuales, simplemente, hacen una valoración aproximada de su tamaño.

Las unidades relativas al viewport son muy importantes y se deben tomar en cuenta cuando realizamos una web para varios tipos de resolución de pantalla.

Unidades de porcentaje:

El porcentaje (%), permite el redimensionamiento sencillo de las fuentes en pantalla. Un valor del 100% significa el tamaño de la fuente actual, típicamente 1em. No es muy recomendable para impresión.

1.8 Establecer fondos

La propiedad `background` permite asignar como fondo un color o una imagen. También permite determinar el posicionamiento de la imagen; si se va a repetir y como; y si se desplazará con el texto en caso de scroll o permanecerá fija.

background-image

Permite especificar una imagen de fondo, indicando la url. Cuando se indica una imagen de fondo, esta se superpone al color de fondo si lo hay. Es conveniente especificar también un color de fondo alternativo ya que si no encuentra la imagen se verá este color de fondo.

Por defecto, la imagen se repite tantas veces como sea necesario en horizontal y vertical hasta que se rellena el elemento.

background-repeat

Usando la propiedad `background-repeat` y sus valores: `repeat`, `repeat-x`, `repeat-y`, y `no-repeat` podemos determinar si la imagen de fondo debe repetirse (si no ocupa toda la pantalla), y el modo de hacerlo. Con `repeat`, la imagen se repetirá horizontal y verticalmente. Los valores `repeat-x` e `repeat-y` hacen que la imagen se repita horizontal o verticalmente respectivamente, creando una banda de imágenes de un lado a otro. Con el valor `no-repeat`, la imaxe no se repite.

```
body { background: red; url(fondo.gif);
background-repeat:repeat-y; }
```

background-attachment

Puede tomar los valores: `scroll` y `fixed`. Si especificamos una imagen de fondo, este parámetro indica si permanecerá fija o acompañará al contenido si deslizamos la página con la barra de scroll.

```
body {
  background: navy url(fondo.gif);
  background-repeat: repeat-y;
  background-attachment: fixed;
}
```

background-position

Por defecto la imagen se coloca desde la esquina superior izquierda de la página (posición 0,0), pero esta posición inicial se puede cambiar utilizando esta propiedad. La forma de hacerlo es:

```
background-position: posicionHorizontal posicionVertical;
```

Para indicar las posiciones podemos usas coordenadas concretas (p.e. 10px) o relativas al tamaño del elemento (10%). Además también disponemos de las palabras clave `left`, `center` y `right` para establecer la posición horizontal y `top`, `center y bottom` para la vertical.

Si solo se da un valor, el otro valor será `center`. Las palabras clave son interpretadas como sigue:

```
top left = left top = 0% 0%
top = top center = center top = 50% 0%
right top = top right = 100% 0%
left = left center = center left = 0% 50%
center = center center = 50% 50%
right = right center = center right = 100% 50%
bottom left = left bottom = 0% 100%
bottom = bottom center = center bottom = 50% 100%
bottom right = right bottom = 100% 100%
```

1.9 Aplicando estilos a listas

Existen ciertos estilos que se pueden aplicar a las listas. Por defecto, los navegadores muestran los elementos de las listas no ordenadas con una viñeta de un pequeño círculo de color negro y los elementos de las listas ordenadas con numeración decimal.

list-style-type

Permite controlar el tipo de viñeta que muestran las listas. Algunos valores son:

- `list-style-type: disc` → muestra como viñeta un círculo relleno.
- `list-style-type: circle` → muestra como viñeta un círculo vacío.
- `list-style-type: square` → muestra como viñeta un cuadrado relleno.
- `list-style-type: decimal` → 1, 2, 3...
- `list-style-type: decimal-leading-zero` → 01, 02, 03…

- `list-style-type: lower-roman` → i, ii, iii, iv...
- `list-style-type: upper-roman` → I, II, III,…
- `list-style-type: lower-alpha` → a, b, c, …
- `list-style-type: upper-alpha` → A, B, C, …
- `list-style-type: lower-greek` → α, β, …
- `list-style-type: none` → Elimina las viñetas

list-style-image

Se usa para personalizar el aspect de las viñetas. Esto permite mostrar una imagen propia en lugar de una viñeta automática. Las imágenes personalizadas se indican mediante la URL a la imagen. Si no encuentra la imagen o no se puede cargar se muestra la imagen correspondiente.

```
ul { list-style-image: url("smiley.png"); }
```

list-style-position

La propiedade `list-style-position` permite controlar la colocación de las viñetas. Los posibles valores son: `inside` y `outside`, indicando si con salto de línea el texto va tabulado o en la línea de la viñeta.

list-style

La propiedade `list-style` permite establecer todas las propiedades de una lista de una vez. El orden de los valores es indiferente. Además, ninguno de ellos es obligatorio.

Cuando se usa una viñeta personalizada, es conveniente indicar la viñeta automática que se mostrará si no se puede cargar la imaxe:

1.10 Enlaces

Además de poder cambiar el estilo de un enlace modificando propiedades como su tipo de letra, color o fondo, CSS permite aplicar distintos estilos a un mismo enlace en función del estado en que se encuentre. Los estados posibles son:

- `a:link` → enlace normal, no visitado por el usuario
- `a:visited` → enlace que el usuario ya ha visitado
- `a:hover` → enlace sobre el que el usuario tiene posicionado el puntero del ratón
- `a:active` → enlace en el momento en que se hace clic.

En el siguiente ejemplo vemos cómo ocultar el subrayado cuando el usuario pasa el ratón por encima de cualquier enlace de la página:

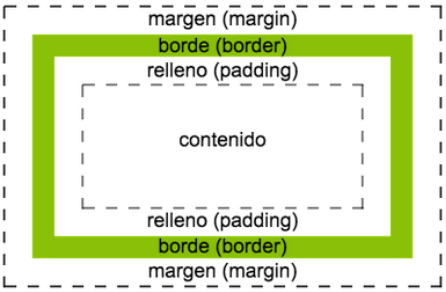
```
a:hover { text-decoration: none; }
```

Al configurar el estilo para varios estados de enlace, hay algunas reglas de orden:

- `a:hover` DEBE definirse después de `a:link` y `a:visited`
- `a:active` DEBE definirse después de `a:hover`

1.11 El modelo de caja

Todos los elementos de una página HTML (párrafos, enlaces, imágenes, etc.) son cajas rectangulares. Los navegadores sitúan estas cajas de la forma que nosotros les indiquemos para maquetar la página.



El modelo de caja tal y como fue especificado en el CSS2.1 funciona así:

```
width (anchura) + padding (relleno) + border (borde) = anchura real de la caja
```

```
height (altura) + padding (relleno) + border (borde) = altura real de la caja
```

Y este es el modelo que utilizan todos los navegadores por defecto excepto el IE6, donde:

```
width (anchura) = anchura real de la caja
```

```
height (altura) = altura real de la caja
```

La propiedad `box-sizing` permite escoger el modelo de caja que deseamos usar. Toma los valores:

- `content-box` → Se comporta como fue especificado en CSS2.1
- `border-box` → El `padding` y el borde del elemento no incrementan su ancho/alto.

Vamos a ver las propiedades que afectan a las cajas

Ancho y alto

La propiedad `width` representa el ancho de la caja y el alto con la propiedad `height`.

Si trabajamos con el modelo de caja de contenido (content-box), que es el establecido por defecto, se trata del ancho y alto interior de la caja, es decir, sin bordes, márgenes, ni `padding`.

Su valor por defecto es `auto`, lo que implica que el elemento se hará lo suficientemente ancho para que quepa su contenido. Podemos indicar este ancho/alto en medidas absolutas (normalmente píxeles) o relativas. En el caso de las unidades relativas, se toma como referencia lo que mide el contenedor; si indicamos un 50% a un elemento que está dentro de un elemento `div`, tomará la mitad de la anchura de ese `div`, si nuestro elemento cuelga directamente de `body`, tomará la mitad de la anchura de la página.

Además de las propiedades `width` y `height` podemos usar:

- `max-width` → Cuando especificamos la anchura del elemento en porcentajes, con esta propiedad indicamos el tamaño máximo que va a tener el elemento. Por ejemplo:

```
width:75%;
max-width:800px;
```

quiere decir que el elemento medirá las tres cuartas partes de su contenedor, pero sin superar 800 píxeles, que será su anchura máxima.

- `min-width` → Lo mismo pero en este caso indicamos el tamaño mínimo.
- `max-height` → Igual que max-width per aplicado a la altura del elemento. Con esta indicamos la altura máxima del elemento.

- `max-height` → Altura mínima del elemento.

Relleno

Con `padding` establecemos la distancia de "relleno" existente entre el contenido y el borde. Si queremos poner un `padding` de 20 píxeles para toda la caja, sería:

```
padding : 20px;
```

Podemos establecer un `padding` distinto para cada lado, usando los sufixos `-top` (superior), `-bottom` (inferior), `-left` (izquierda) y `-right` (derecha):

```
padding-top: 10px;
padding-bottom: 5px;
padding-left: 30px;
padding-right: 20px;
```

Podemos abreviar lo anterior en una sola línea, indicando primero el `padding` superior y continuar siguiendo el orden de las agujas del reloj. Es decir: arriba, derecha, abajo, izquierda. El ejemplo anterior quedaría así:

```
padding: 10px 20px 5px 30px;
```

Otro atajo útil es especificar sólo dos medidas: el `padding` superior e inferior por un lado, y el `padding` lateral. Si queremos que los `paddings` superior e inferior sean de 10 píxeles, y los laterales (izquierdo y dereche) de 20 píxeles, escribimos:

```
padding: 10px 20px;
```

Bordes

Si queremos que nuestra caja tenga bordes, tenemos que usar la propiedad `border`. Tiene la siguiente sintaxis:

```
border: width style color;
```

- `width` especifica el grosor del borde. Normalmente se expresa en píxeles, pero también se pueden usar las palabras `thin` (fino), `medium` (normal) y `thick` (grueso) que quedan a interpretación del navegador.
- `style` es el tipo de borde. Hay bastantes, pero los valores más comunes son: `solid` (línea continua), `dashed` (línea discontinua), `dotted` (línea de puntos), `double` (línea continua doble), `groove` (borde 3D con efecto de hundimiento) y ridge (borde 3D con efecto de elevación).
- `color` indica el color del borde.

Podemos poner un tipo de borde diferente para cada lado con los sufijos `-top`, `-bottom`, `-left` y `-right`. Por ejemplo, para que algo tenga un borde inferior de 1 píxel de puntitos rojos:

```
border-bottom: 1px dotted #f00;
```

Para eliminar el borde, simplemente ponemos que tiene de grosor 0 píxeles o que el estilo es `none`. Esto se usa con las imágenes, ya que si tenemos una imagen enlazando a algo, los navegadores le ponen un borde.

```
img {border: none;}
```

Podemos configurar por separado los bordes con las propiedades:

- `border-width`, para indicar el grosor y donde podemos usar los mismos atajos que con el padding para indicar el grosor de cada lado. Por ejemplo:

```
border-width: thick thin medium 0px;
```

- `border-style`, para indicar el para indicar el estilo y donde podemos usar los mismos atajos que con el padding para indicar el estilo de cada lado. Por ejemplo:

  ```
  border-style: solid dashed;
  ```

- `border-color`, para indicar el color. ejemplo:

  ```
  border-top-color: red;
  ```

Margen

La propiedad `margin`, establece el espacio entre el borde de la caja y los elementos que la rodean. Se usa igual que la propiedad `padding`, así que la forma de escribir y los atajos es exactamente la misma. Por ejemplo, si queremos márgenes superior e inferior de 20 píxeles, y laterales de 5 píxeles:

```
margin: 20px 5px;
```

Truco: Para centrar un elemento de bloque, podemos hacer uso de `auto`:

```
margin: 0px auto;
```