CSS

Ángel Villalba Fdez-Paniagua

Table of Contents

1. Introducción

CSS nos permite crear una serie de reglas que especifican como debe de ser la apariencia de los elementos, como por ejemplo, el color de fondo de una página, el tamaño y color de las letras, el tipo de fuente, la posición de los elementos...

CSS le asigna una serie de reglas a una serie de elementos. Estas reglas van a definir como se tienen que mostrar los elementos a los que se les va a aplicar.

Una **regla** de CSS está compuesta de dos partes:

- **Selector**
- **Declaración**

El **selector** indica a que elementos del documento hay que aplicarles una regla. Las reglas se pueden aplicar a más de un elemento al mismo tiempo.

La **declaración** indica que estilos se tienen que aplicar a los elementos a los que apuntan los selectores. Estas declaraciones se encuentran entre llaves (`{}`) y se componen de dos partes:

- **Propiedad**: indica los aspectos que se quieren cambiar de un elemento (el color, el tamaño...)
- **Valor**: valor que se le va a dar a la propiedad (`red`, `13px`...)

```
p {
  color: blue;
  text-decoration: underline;
}
```

2. Uso de css

Para poder usar los estilos en una página de HTML, tenemos distintas formas de hacerlo:

2.1. Archivos externos

Si vamos a tener los estilos en un archivo externo a la página HTML, tendremos que importarlo de alguna forma. Para ello vamos a usar la etiqueta **link**.

A esta etiqueta le tenemos que añadir los siguientes atributos:

- `href`: la dirección donde se encuentra nuestra hoja de estilos (archivo con extensión `.css`), que debería de estar en una carpeta llamada *css* o *styles*.
- `rel`: especifica la relación entre la página HTML y el archivo que se está enlazando. En este caso el valor será `stylesheet` porque estamos enlazando una hoja de estilos.
- `type`: este atributo especifica el tipo de documento que estamos importando. En este caso es un documento de *css* por lo que su valor será `text/css`.

/styles/style.css

```
h1 {
  color: red;
}
```

/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" type="text/css" href="styles/style.css">
</head>
<body>
  <h1>Un título</h1>
  <p>Un párrafo largo...</p>
</body>
</html>
```

Dentro de una misma página HTML se pueden enlazar varios archivos de CSS, añadiendo más etiquetas **link** que apunten al resto de archivos.

Cuando tenemos múltiples páginas en un proyecto, esta es la mejor forma de crear los estilos, haciendolo en archivos externos, porque los podremos reutilizar entre las distintas páginas.

También estamos separando la estructura de la apariencia que se le da, y cualquier cambio en un estilo, se aplicará en todas aquellas páginas donde se haya enlazado el archivo.

2.2. Archivo HTML

También podemos añadir los estilos dentro de la propia página HTML, entre las etiquetas **style**.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <style>
    h1 {
      color: yellow;
    }
  </style>
</head>
<body>
  <h1>Un título</h1>
  <p>Un párrafo largo...</p>
</body>
</html>
```

Este método solo debería usarse cuando en una página vamos a tener muy pocos estilos que aplicar y que no afectan a ninguna otra página.

2.3. Etiqueta HTML

En las anteriores versiones de HTML también se permitía añadir los estilos directamente en las propias etiquetas de HTML a través del atributo **style**. Pero en la última versión de HTML, se recomienda no hacerlo ya que lo que se persigue en esta última versión es separar la estructura de los estilos, aunque este método de aplicar estilos sigue funcionando.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <style>
    h1 {
      color: yellow;
    }
  </style>
</head>
<body>
  <h1 style="color:blue">Un título</h1>
  <p>Un párrafo largo...</p>
</body>
</html>
```

3. Selectores CSS

Los **selectores de CSS** nos permiten decidir a que elementos del DOM se les va a aplicar los estilos que se han definido. Hay varios tipos de selectores que se verán a continuación:

Table 1. Selectores

Selector	Nombre	Descripción
* {}	Selector global	Devuelve todos los elementos
p {}	Selector de etiqueta	Devuelve todos los elementos *p*
h1, h2 {}	Agrupación de selectores	Devuelve todos los elementos *h1* y *h2*
ul a {}	Selector de descendientes	Devuelve todos los elementos *a* que son descendientes de un elemento *ul*
ul > li {}	Selector de hijos	Devuelve todos los elementos *li* que son hijos directos de un elemento *ul*
#boton-imp {}	Selector de id	Devuelve todos los elementos cuyo *id* tiene el valor `boton-imp`
.enlace {}	Selector de clase	Devuelve todos los elementos que tienen la clase `enlace`
ul + p {}	Selector adyacente	Devuelve todos los elementos *p* que van justamente después de un elemento *ul*
ul ~ p {}	Selector hermano	Devuelve todos los elementos *p* hermanos a un elemento *ul*
div[class] {}	Selector de atributos	Indica que el elemento *div* tiene que tener un atributo *class*
a[href="www.google.com"]	Selector de atributos	Indica que el valor del atributo *href* tiene que ser exactamente `www.google.com`
a[href*="goo"]	Selector de atributos	Indica que el valor del atributo *href* tiene que contener la cadena `goo`
a[href^="http"]	Selector de atributos	Indica que el valor del atributo *href* tiene que empezar por `http`
a[href$=".es"]	Selector de atributos	Indica que el valor del atributo *href* tiene que acabar en `.es`

4. Orden en el que se aplican los estilos

En el caso en que haya varias reglas que afectan al mismo elemento, tendremos que tener en cuenta cual será la que se aplique al final.

Si estas reglas son iguales se va a aplicar la que aparezca en último lugar.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <style>
    h1 {
      color: yellow;
    }
  </style>
</head>
<body>
  <h1>Un título</h1>
  <p>Un párrafo largo...</p>
</body>
</html>
```

En ese caso, el color de la letra del elemento **h1** será *amarillo*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    h1 {
      color: yellow;
    }
  </style>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <h1>Un título</h1>
  <p>Un párrafo largo...</p>
</body>
</html>
```

Mientras que en este otro caso, el color que se aplicará, será el que se encuentra en el archivo de estilos (se aplica el color *rojo*).

También se mira la **especificidad de un selector** a la hora de aplicar los estilos. La especificidad de un selector se refiere a que el selector sea más específico o directo que otro.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <style>
    h1 {
      color: yellow;
    }
    body > p {
      color: orange;
    }
    p {
      color: blue;
    }
  </style>
</head>
<body>
  <h1>Un título</h1>
  <p>Un párrafo largo...</p>
</body>
</html>
```

En este caso, aunque el estilo que aplicaría el color *azul* al elemento **p** se carga después, el que se aplica es el del selector `body > p` que es más específico que el otro, por lo tanto va a aplicarse el color *naranja*.

Y por último, hay que tener en cuenta que el que predomina sobre todos, es aque al que se le añade la palabra `!important`, dando igual si es menos específico que otro selector o si se carga antes.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" type="text/css" href="style.css">
  <style>
    p {
      color: darkred !important;
    }
    h1 {
      color: yellow;
    }
    body > p {
      color: orange;
    }
    p {
      color: blue;
    }
  </style>
</head>
<body>
  <h1>Un título</h1>
  <p>Un párrafo largo...</p>
</body>
</html>
```

En este caso se le aplica el color *rojo oscuro*.

5. Colores

5.1. color

La propiedad **color** permite especificar el color del texto de un elemento.

Los colores se pueden especificar de tres formas distintas:

- Valores RGB: se especifica el color midiendo la cantidad de *rojo*, *verde* y *azul* en ese orden.
- Códigos Hexadecimal: estos códigos se componen de 3 o 6 dígitos que representan la cantidad de *rojo*, *verde* y *azul* precedidos de #.
- Colores predefinidos: en CSS hay 147 colores predefinidos que se usan poniendo el nombre que tienen asignados.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    h1 {
      color: darkorange;
    }
    h2 {
      color: #AA8B39;
    }
    p {
      color: rgb(170, 57, 57);
    }
  </style>
</head>
<body>
  <h1>Título</h1>
  <h2>Subtítulo</h2>
  <p>Un párrafo</p>
</body>
</html>
```

5.2. background-color

CSS trata cada elemento de HTML como si estuvieran dentro de una caja, y la propiedad **background-color** establece el color de fondo de esa caja.

El color del fondo se le puede especificar de las mismas tres formas que la propiedad **color**.

En caso de no especificar el color de fondo, por defecto se queda *transparente*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    h1 {
      color: darkorange;
      background-color: darkblue;
    }
    h2 {
      color: #AA8B39;
    }
    p {
      color: rgb(170, 57, 57);
    }
  </style>
</head>
<body>
  <h1>Título</h1>
  <h2>Subtítulo</h2>
  <p>Un párrafo</p>
</body>
</html>
```

5.3. opacity

La última versión de CSS3 introduce la propiedad **opacity** que permite especificar la opacidad de un elemento.

El valor de la opacidad es un número entre `0.0` (*0%*) y 1.0 (*100%*).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      background-color: orange;
    }
    h1 {
      color: darkorange;
      background-color: darkblue;
    }
    h2 {
      color: #AA8B39;
    }
    p {
      color: rgb(170, 57, 57);
      background-color: rgb(0, 0, 0);
      opacity: 0.4;
    }
  </style>
</head>
<body>
  <div>
    <h1>Título</h1>
    <h2>Subtítulo</h2>
    <p>Un párrafo</p>
  </div>
</body>
</html>
```

También podemos añadir la opacidad usando `rgba()`, donde el *a* es el cuarto valor (el canal **alpha**) que controla la opacidad del elemento.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      background-color: orange;
    }
    h1 {
      color: darkorange;
      background-color: darkblue;
    }
    h2 {
      color: #AA8B39;
      background-color: rgba(0, 0, 0, 0.3);
    }
    p {
      color: rgb(170, 57, 57);
      background-color: rgb(0, 0, 0);
      opacity: 0.4;
    }
  </style>
</head>
<body>
  <div>
    <h1>Título</h1>
    <h2>Subtítulo</h2>
    <p>Un párrafo</p>
  </div>
</body>
</html>
```

5.4. Herramientas para colores

- Paletton

6. Cajas

CSS trata cada elemento como si estubieran dentro de una caja, y con los estilos vamos a poder establecer unas cuantas propiedades que afectan a la apariencia de estas.

6.1. `width` y `height`

Con las propiedades de **width** y **height** podemos establecer el tamaño de las cajas, el ancho y el alto.

Normalmente las medidas de estas propiedades se suelen especificar en *pixeles*, aunque también se pueden poner las medidas expresadas en porcentajes.

Si se usan los porcentajes, el tamaño de la caja es relativo al tamaño de la caja que lo contiene.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    p {
      height: 50%;
      width: 50%;
      background-color: #999999;
    }
  </style>
</head>
<body>
  <div>
    <p>Minions ipsum tatata bala tu uuuhhh poopayee potatoooo uuuhhh hahaha potatoooo
bananaaaa tulaliloo aaaaaah. Bappleees hahaha ti aamoo! Butt poulet tikka masala. Jeje
poopayee daa tatata bala tu bee do bee do bee do pepete belloo! Belloo! Belloo! Bee do
bee do bee do aaaaaah. Uuuhhh bee do bee do bee do poopayee bappleees hahaha jiji
chasy bappleees uuuhhh jiji. Tank yuuu! belloo!</p>
  </div>
</body>
</html>
```

6.2. `overflow`

La propiedad **overflow** le dice al navegador que hacer cuando el contenido de una caja no entra porque es demasiado largo.

Esta propiedad puede tener dos valores:

- **hidden**: oculta el texto
- **scroll**: se añade una barra de scroll

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    p {
      height: 50%;
      width: 50%;
      background-color: #999999;
      overflow: scroll;
    }
  </style>
</head>
<body>
  <div>
    <p>Minions ipsum tatata bala tu uuuhhh poopayee potatoooo uuuhhh hahaha potatoooo
bananaaaa tulaliloo aaaaaah. Bappleees hahaha ti aamoo! Butt poulet tikka masala. Jeje
poopayee daa tatata bala tu bee do bee do bee do pepete belloo! Belloo! Belloo! Bee do
bee do bee do aaaaaah. Uuuhhh bee do bee do bee do poopayee bappleees hahaha jiji
chasy bappleees uuuhhh jiji. Tank yuuu! belloo!</p>
  </div>
</body>
</html>
```

Con esta propiedad podemos evitar que el contenido se solape con otros elementos.

6.3. `border`

Cada caja tiene una propiedad **border** que por defecto tiene un valor de `0px` por lo que no se ve. Con esta propiedad vamos a poder ver el borde de cada caja.

Podemos usar la propiedad `border-width` para controlar el ancho del borde. Esta propiedad puede recibir el valor en *pixeles* o como uno de los siguientes valores:

- thin
- medium
- thick

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    div#con-borde {
      border-top-width: 1px;
      border-bottom-width: 1px;
      border-left-width: 2px;
      border-right-width: 2px;
    }
  </style>
</head>
<body>
  <div id="con-borde">
  </div>
</body>
</html>
```

Si queremos ver el borde le tendremos que aplicar algún color, y para ello se usa la propiedad `border-color`.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    div#con-borde {
      border-top-width: 1px;
      border-bottom-width: 1px;
      border-left-width: 2px;
      border-right-width: 2px;
      border-style: dashed;
    }
  </style>
</head>
<body>
  <div id="con-borde">
  </div>
</body>
</html>
```

Y por último podemos indicar el estilo del borde con la propiedad `border-style`. Si queremos ver el borde como una línea (*solid*), con puntos (*dotted*), con líneas separadas por espacios (*dashed*)...

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    div#con-borde {
      border-top-width: 1px;
      border-bottom-width: 1px;
      border-left-width: 2px;
      border-right-width: 2px;
      border-style: dashed;
      border-color: yellowgreen;
    }
  </style>
</head>
<body>
  <div id="con-borde">
  </div>
</body>
</html>
```

También podemos usar la forma simplificada, que sería darle los tres valores a la propiedad `border` en el siguiente orden: *width*, *style* y *color*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    div#con-borde {
      border: 2px dashed yellowgreen;
    }
  </style>
</head>
<body>
  <div id="con-borde">
  </div>
</body>
</html>
```

6.4. margin

La propiedad **margin** se situa fuera del borde, y nos va a permitir añadir un espacio exterior entre los bordes de varias cajas.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 250px;
      width: 250px;
      background-color: #000000;
    }
    .caja {
      width: 60px;
      height: 60px;
      background-color: orange;
    }
    div#con-margin > .caja {
      margin: 10px;
    }
  </style>
</head>
<body>
  <div id="sin-margin">
    <div class="caja">Dato 1</div>
    <div class="caja">Dato 2</div>
    <div class="caja">Dato 3</div>
  </div>
  <div id="con-margin">
    <div class="caja">Dato 1</div>
    <div class="caja">Dato 2</div>
    <div class="caja">Dato 3</div>
  </div>
</body>
</html>
```

También se pueden usar las siguientes propiedades para añadir el *margen* solamente a los lados en que los necesitemos.

- `margin-top`
- `margin-right`
- `margin-bottom`
- `margin-left`

También es posible darle a la propiedad **margin** los cuatro valores anteriores, empezando desde el *top* y siguiendo las agujas del reloj (*top*, *right*, *bottom* y *left*).

6.5. padding

La propiedad **padding** se situa dentro del borde, y es el espacio que separa el borde del contenido que se encuentra en la caja.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 350px;
      width: 350px;
      background-color: #000000;
    }
    .caja {
      width: 60px;
      height: 60px;
      background-color: orange;
    }
    div#con-margin-padding > .caja {
      margin: 10px;
      padding: 15px;
    }
  </style>
</head>
<body>
  <div id="sin-margin-padding">
    <div class="caja">Dato 1</div>
    <div class="caja">Dato 2</div>
    <div class="caja">Dato 3</div>
  </div>
  <div id="con-margin-padding">
    <div class="caja">Dato 1</div>
    <div class="caja">Dato 2</div>
    <div class="caja">Dato 3</div>
  </div>
</body>
</html>
```

También se pueden usar las siguientes propiedades para añadir el *padding* solamente a los lados en que los necesitemos.

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

También es posible darle a la propiedad **padding** los cuatro valores anteriores, empezando desde el *top* y siguiendo las agujas del reloj (*top*, *right*, *bottom* y *left*).

6.6. Centrar caja

En caso de querer centrar la caja en la página o dentro de algún elemento, podemos establecer los valores de las propiedades `margin-left` y `margin-right` a **auto**. Para que funcione correctamente, hay que asignarle un *ancho* (`width`) a la caja, sino ocupará todo el ancho disponible.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    div#centrada, div#centrada > p {
      margin: 10px auto 10px auto;
    }
    p {
      height: 50%;
      width: 50%;
      background-color: #999999;
    }
  </style>
</head>
<body>
  <div id="centrada">
    <p>Un texto</p>
  </div>
</body>
</html>
```

6.7. `display`

La propiedad **display** permite convertir un elemento de línea en un elemento de bloque o viceversa, y también se puede usar para ocultar elementos.

Los valores que puede obtener esta propiedad son:

- **inline**: hace que un elemento de bloque actue como un elemento de línea.
- **block**: hace que un elemento de línea actue como un elemento de bloque.
- **block-inline**: hace que un elemento de bloque actue como un elemento de línea, pero se queda

con alguna característica del elemento de bloque.

- **none**: se encarga de esconder un elemento en la página.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    li {
      display: inline;
      margin-right: 10px;
    }
  </style>
</head>
<body>
  <nav>
    <ul>
      <li><a href="#">Enlace 1</a></li>
      <li><a href="#">Enlace 2</a></li>
      <li><a href="#">Enlace 3</a></li>
    </ul>
  </nav>
</body>
</html>
```

6.8. box-shadow

La propiedad **border-shadow** nos permite añadir una sombra alrededor de la caja. Para ello la propiedad va a recibir mínimo tres valores y como máximo cinco. Estos valores son:

- Desplazamiento horizontal (obligatorio)
- Desplazamiento vertical (obligatorio)
- Distancia de difuminado (opcional)
- Propagación de la sombra (opcional)
- Color (obligatorio)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    div#sombra {
      box-shadow: 10px 10px 15px 5px blue;
    }
  </style>
</head>
<body>
  <div id="sombra">
  </div>
</body>
</html>
```

6.9. border-radius

La propiedad **border-radius** permite redondear los bordes de las cajas. El valor que recibe se expresa en *pixeles*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div {
      height: 300px;
      width: 300px;
      background-color: #000000;
    }
    div#borde-redondeado {
      border-radius: 15px;
    }
  </style>
</head>
<body>
  <div id="borde-redondeado">
  </div>
</body>
</html>
```

Si queremos que la caja tenga forma de circulo, se le puede dar como valor `50%` a esta propiedad.

7. Listas

Dentro de las listas podemos cambiar los símbolos que aparecen delante de cada elemento **li**. Para ello vamos a usar la propiedad **list-style-type** la cual puede recoger los siguientes valores dependiendo del tipo de lista en la que se encuentren los elementos *li*.

- Listas desordenadas:
  - none
  - disc
  - circle
  - square

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    ul > li {
      list-style-type: square;
    }
  </style>
</head>
<body>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ul>
</body>
</html>
```

- Listas ordenadas
  - decimal
  - decimal-leading-zero
  - lower-alpha
  - upper-alpha
  - lower-roman
  - upper-roman

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    ol > li {
      list-style-type: lower-alpha;
    }
  </style>
</head>
<body>
  <ol>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ol>
</body>
</html>
```

8. Fuentes

CSS nos proporciona unas propiedades que nos va a permitir modificar la apariencia del texto.

8.1. font-weight

La propiedad **font-weight** nos permite especificar el grosor que tiene que tener la fuente.

Los valores que puede obtener son:

- normal: el valor por defecto.
- bold: negrita.
- lighter: letra menos gruesa que el tipo del elemento padre.
- bolder: letra más gruesa que el tipo del elemento padre.
- 100, 200, 300, ..., 900: valor númerico que va desde *100* a *900*. Si se pone el valor *120*, se aplicará el más cercano, en este caso sería el *100*.

```
h1 {
  font-weight: bold;
}
```

8.2. font-style

La propiedad **font-style** permite definir el aspecto de una familia tipográfica entre los siguientes

valores:

- normal: letra normal.
- italic/oblique: letra cursiva (depende de la familia hay que usar un valor u otro).

```
h1 {
  font-style: italic;
}
```

8.3. font-family

La propiedad **font-family** nos permite definir una serie de fuentes que queremos usar para los elementos que cumplan con el selector. Se pueden poner varias fuentes una detrás de otra separadas por comas.

```
h1 {
  font-family: Helvetica, sans-serif, Courier;
}
```

8.4. text-shadow

La propiedad **text-shadow** nos permite aplicar una sombra a los textos.

Los valores que obtiene son los `desp_h desp_v difuminado color_rgb` y hay que ponerlos en el orden en que aparecen.

```
text-shadow: 4px 4px 3px #000;
```

9. Fondos

9.1. background-image

La propiedad **background-image** nos permite poner una imagen de fonde de la página HTML. Esta propiedad recibe como valor `url(urlImg)`.

```
#logo {
    background-image: url(images/logo.png);
}
```

9.2. `background-repeat`

La propiedad **background-repeat** nos permite indicar si queremos que la imagen se repita en el fondo.

Los valores que puede obtener son:

- repeat: repite la imagen horizontal y verticalmente. Es el valor por defecto.
- no-repeat: la imagen de fondo no se va a repetir.
- repeat-x: la imagen se repite solo horizontalmente.
- repeat-y: la imagen se repite solo verticalmente.

```
div.main {
    background-image: url(images/fondo1.png);
    background-repeat: no-repeat;
}
```

10. Positions

La propiedad **position** nos permite indicar como se tienen que posicionar los elementos.

10.1. `static`

Los elementos se van a colocar en el orden según aparecen en el documento. Este es el valor que coge por defecto la propiedad `position`.

10.2. `relative`

Con el valor **relative** los elementos se van a colocar siguiendo el orden en el que aparecen, pero si modificamos su posición, por ejemplo usando las propiedades `left`, `top`... se va a mover partiendo de la posición en la que se encuentra.

El siguiente elemento, si tiene este valor como posición y lo movemos, se moverá a partir de la posición que le tocaba ocupar, la cual no se modifica aunque se haya movido el elemento anterior.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div>
    <div class="relative1"></div>
    <div class="relative2"></div>
  </div>
</body>
</html>
```

```css
div {
  width: 500px;
  height: 500px;
  border: 1px solid red;
  left: 100px;
  position: absolute;
}

div.relative1 {
  width: 150px;
  height: 150px;
  border: 1px solid blue;
  position: relative;
  top: 200px;
  /* top: 0px; */
  left: 200px;
}

div.relative2 {
  width: 150px;
  height: 150px;
  border: 1px solid yellow;
  position: relative;
  top: 0px;
  left: 110px;
}
```

Si probamos a cambiar el valor que se usa para la propiedad `top` en la primera caja por el que se encuentra comentado, podremos ver que el segundo hijo se mueve a partir de la posición que le correspondía y no de la nueva que estaba ocupando el primer hijo.

10.3. fixed

Con el valor **fixed** el elemento lo ponemos en la posición que queramos a partir de la esquina superior izquierda del navegador, dando igual si es hijo de otros elementos o no lo es.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div>
      <div class="fixed1"></div>
      <div class="fixed2"></div>
  </div>
</body>
</html>
```

```
div {
  width: 500px;
  height: 500px;
  border: 1px solid red;
  left: 400px;
  position: relative;
}

div.fixed1 {
  width: 150px;
  height: 150px;
  border: 1px solid blue;
  position: fixed;
  top: 200px;
  left: 200px;
}

div.fixed2 {
  width: 150px;
  height: 150px;
  border: 1px solid orange;
  position: fixed;
  top: 100px;
  left: 100px;
}
```

10.4. absolute

El valor de **absolute** es como el **fixed** pero en lugar de posicionarse teniendo en cuenta el navegador, se posiciona teniendo en cuenta la posición del elemento superior sitiene como valor alguno distinto a `static` para la propiedad `position`.

En caso de tener el elemento superior `static` como valor para la propiedad `position`, entonces se tiene en cuenta la posición del navegador a la hora de posicionar el elemento.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div>
    <div class="absolute"></div>
  </div>
</body>
</html>
```

```
div {
  width: 500px;
  height: 500px;
  border: 1px solid red;
  float: right;
  /* position: relative; */
}

div.absolute {
  width: 150px;
  height: 150px;
  border: 1px solid blue;
  position: absolute;
  top: 200px;
  left: 200px;
}
```

10.5. sticky

Con el valor **sticky** vamos a conseguir que un elemento permanezca en un sitio, aun cuando se realiza un scroll con el que tendría que desaparecer.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Cabecera de la página</h1>
  </header>
  <main>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit eos quae
cupiditate deserunt fugiat odio ut ab voluptatem quaerat rem modi commodi laborum et
voluptatibus quas omnis beatae, nostrum ipsum? Impedit magni eos at commodi tempore
nihil deserunt velit tenetur, corrupti iusto possimus neque cumque suscipit voluptates
facilis aut minus? Aperiam et officia maxime iure reiciendis dolore culpa, esse nemo!
Enim provident doloremque omnis aspernatur quis alias nesciunt corporis delectus
voluptatem, nostrum est molestias laudantium magnam modi quisquam! Nobis, laborum
doloremque alias commodi voluptatem vel a quam esse at consequuntur!</p>
  </main>
</body>
</html>
```

```
body {
  height: 1500px;
  margin: 0px;
}

header {
  height: 50px;
  background-color: orange;
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  position: sticky;
  top: 0;
}

header > h1 {
  font-family: Verdana, Geneva, Tahoma, sans-serif;
}

main {
  width: 60%;
  margin: 20px auto;
}
```

11. Degradados

Los **degradados** o **gradientes** nos van a permitir definir un color inicial y un color final, y lo que ocurrirá es que el color inicial se irá transformando poco a poco en uno o varios colores hasta llegar al color final.

Los gradientes requieren al menos dos colores (inicio y fin) y, sino se especifica la dirección, es de arriba a abajo.

```
#grad {
    background: linear-gradient(red, yellow);
}
```

Se puede indicar el sentido en que se va a aplicar el degradado con los siguientes valores:

- to top
- to right
- to bottom
- to left

```
#grad {
  background: linear-gradient(to right, red , yellow);
}
```

Ademas se puede indicar dos posiciones para lograr que el efecto salga en diagonal.

```
#grad {
  background: linear-gradient(to bottom right, red, yellow);
}
```

Podemos repetir un patrón mediante la propiedad `repeating-linear-gradient(color1 n%, color2 n%, color3 n%, ⋯)` y se puede indicar el tamaño que tiene que ocupar cada color añadiendole un porcentaje justo a continuación de cada color.

```
#grad {
  background: repeating-linear-gradient(red, yellow 10%, green 20%);
}
```

12. Transformaciones

Existen cuatro tipos de valores para transformar elementos HTML mediante CSS3:

12.1. Skew

Desplaza los ejes horizontales.

```
#skew {
  transform: skew(35deg);
}
```

12.2. Scale

Modifica la escala del elemento.

```
.scale {
  transform: scale(1,0.5);
}
```

12.3. Rotate

Cambia la rotación del elemento (en grados).

```
.rotate {
  transform: rotate(45deg);
}
```

12.4. Translate

Desplaza el elemento desde su posición original.

```
.translate {
  transform: translate(10px, 20px);
}
```

13. Transiciones

La propiedad **transition** es una animación hecha unicamente con CSS, su sintaxis consiste en seleccionar una propiedad CSS a animar y definir el tiempo de animación. Para que funcione tiene que estar asociada a un cambio de estado (*focus*, *hover*...) o evento.

```
#hola:hover {
  background-color:black;
  color:white;
}

#hola {
  transition: background-color 1s, color 1s;
}
```

14. Animaciones

Una animación en CSS3 permite que cualquier elemento cambie gradualmente de una propiedad CSS a otra, el número de veces que se quiera. Es una alternativa moderna a las animaciones que usan *Flash* o *JavaScript*.

Debemos especificar `keyframes` que son momentos puntuales dentro de la animación en los que indicamos que valor tiene que tener una propiedad del elemento que se está animando.

```
#animation1 {
  animation-name: animation1;
  animation-duration: 4s;
}

@keyframes animation1 {
  from {
    background-color: red;
  }
  to {
    background-color: yellow;
  }
}
```

Si no especificamos `animation-duration`, la animación no se reproducirá, porque por defecto es `0`.

Podemos usar porcentajes para especificar los puntos clave del *keyframe*, el uso de `from` y `to` es equivalente a usar los porcentajes `0%` y `100%`.

```
@keyframes animation1 {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
```

Podemos hacer que cualquier animación retrase su inicio mediante el uso de `animation-delay` y repetir la animación el número de veces que especifiquemos con `animation-duration-count` (si ponemos `infinite`, se reproduce eternamente).

También se puede indicar la dirección en que tiene lugar la animación con `animation-direction: reverse` o incluso alternarla con `animation-direction: alternate`.

Podemos especificar la curva que seguirá la animación con `animation-timing-function` cuyos valores pueden ser los siguientes:

- ease: empieza lento, luego acelera, y termina lento. Es el valor que tiene por defecto.
- linear: misma velocidad de principio a fin.
- ease-in: empieza lento.
- ease-out: termina lento.
- ease-in-out: empieza y termina lento.
- cubic-bezier(x, y, z, n): permite especificar nuestros parámetros siguiendo una función de Bezier.

Todos los parámetros se pueden resumir en una sola propiedad `animation` con el siguiente orden `name duration timing-function delay iteration-count direction`.

```
#div {
  animation: animation1 4s linear 5s infinite alternate;
}
```