Tema 3: Desarrollo web en cliente: CSS.

(semana 1)

Bloque 1: Desarrollo web en cliente: HTML y CSS.

Módulo: Lenguajes de marcas y sistemas de gestión de la información.

IES San Vicente
Curso 2021-2022

Índice de contenidos

1 Introducción a CSS

Hasta el momento hemos visto cómo organizar un documento HTML, a lo largo de este tema vamos a ver qué son las **hojas de estilo en cascada** (*Cascading Style Sheets*) o **CSS**. Debemos tener claro que una página web en realidad es un documento de texto escrito en código HTML y que el código CSS es el encargado de dar forma, color, posición (entre otras características) a una página web.

El término CSS es el lenguaje de estilos utilizado para establecer cómo se van a presentar los documentos HTML o XML. Se denomina hoja de estilo en cascada porque las reglas se aplican de arriba a abajo.

1.1 Historia

CSS es una especificación desarrollada y mantenida por el W3C, comunidad que como vimos se encarga de desarrollar estándares para asegurar el crecimiento y la neutralidad de la web con independencia de las tecnologías propietarias. En dicho consorcio participan Apple, Adobe, Cisco, Facebook, Microsoft, Nokia, Twitter, etc. entre muchos otros.

La evolución de CSS ha ido evolucionando en diferentes versiones denominadas niveles. En el año 1996 salió CSS1 con propiedades de fuente, colores, alineación, etc. En 1998 CSS2 con propiedades de posicionamiento, tipos de medios, etc. En 2005 con CSS2.1 se corrigen errores de CSS2 y se modificaron ciertas propiedades. En 2011 CSS3 donde se separan las nuevas funcionalidades en pequeños módulos, favoreciendo así su implementación en navegadores.

2 Cómo usar CSS

Antes de comenzar a ver las reglas de CSS debemos conocer las diferentes formas que podemos utilizar para incluir estilos en nuestros HTML.

A continuación, vamos a ver las tres formas que hay para incluir nuestros estilos, siendo la más común (y recomendada) la primera, y la última la menos recomendada principalmente en proyectos grandes.

2.1 Enlace CSS externo (etiqueta link)

En la cabecera de nuestro HTML (dentro de la etiqueta `<head>`) debemos incluir una etiqueta `<link>` con la que haremos referencia a el archivo CSS que vamos a utilizar, en el atributo `href` indicaremos la ubicación de esa hoja de estilos. Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Mi primera hoja de estilo</title>
      <!-- Incluyo mi archivo CSS -->
      <link rel="stylesheet" href="estilo.css">
```

```
    </head>
    <body>
        <h1>Aprendiendo CSS</h1>
        <p>Hola este mensaje estará en rojo</p>
    </body>
</html>
```

Al incluir la etiqueta `link` con el `href` a nuestra hoja de estilos, estamos indicando al navegador que deben aplicar los estilos del archivo `estilo.css`. Antiguamente también se añadía el atributo `type="text/css"`, pero desde HTML5 ya no es necesario, y si se incluye es por el tema de compatibilidad con navegadores muy antiguos.

2.2 Enlace CSS interno (etiqueta style)

Otra forma habitual de incluir estilos en una página web es añadiéndolos directamente en el documento HTML mediante la etiqueta `<style>` dentro de la etiqueta `<head>`:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Mi primera hoja de estilo</title>
        <!-- Incluyo mi CSS -->
        <style>
            p {
                color: red;
            }
        </style>
    </head>
    <body>
        <h1>Aprendiendo CSS</h1>
        <p>Hola este mensaje estará en rojo</p>
    </body>
</html>
```

El ejemplo anterior estamos incluyendo nuestro CSS mediante la etiqueta `<style>`, pero con el objetivo de poder reutilizar nuestros estilos a posteriori es recomendable que se utilice mediante la etiqueta `<link>` como vimos en el punto anterior (Enlace CSS externo).

2.3 Estilos en linea (atributo style)

La tercera y última forma que tenemos para aplicar estilos en un documento HTML es hacerlo directamente mediante el atributo `style` a la etiqueta a la que le queremos aplicarle el estilo.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Mi primera hoja de estilo</title>
    </head>
    <body>
        <h1>Aprendiendo CSS</h1>
        <p style="color: red;">Hola este mensaje estará en rojo</p>
    </body>
</html>
```

Al igual que con el método anterior (etiqueta `<style>`), no se recomienda utilizar este método.

¡IMPORTANTE! Si modificamos una misma propiedad desde varios sitios (hoja de estilos / etiqueta style/ atributo style, ¿cuál se aplicará?. Se aplica el estilo de más prioridad, siendo de mayor a menor: atributo style → etiqueta style → Hoja CSS. Ejemplo:

Archivo index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Mi primera hoja de estilo</title>
        <style>
            p {
                color: green;
            }
        </style>
    </head>
    <body>
        <p style="color: red;">Hola este mensaje
estará en color...???</p>
    </body>
</html>
```

Archivo estilo.css

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

3 Estructura de CSS

Para definir un estilo CSS, se utilizan reglas. Una regla debe tener el siguiente formato:

selector { propiedad: valor; }

El **selector** es el elemento HTML al que vamos a aplicarle un estilo. Por ejemplo, si usamos el selector `p` estamos seleccionado todos los párrafos de nuestra página. Entre llaves, debemos poner las **declaraciones** de los estilos como puede ser el color de la letra, el borde de una tabla, etc. Cada declaración estará formada por una propiedad y un o varios valores asociados a esa propiedad, y cada declaración irá separada por punto y coma.

```
p {
    color: red; /* Con esta declaración todos los párrafos tendrán el color de letra rojo */
}
```

Nota: Un comentario en CSS comienzan con `/*` y finalizan con `*/` como en la mayoría de los lenguajes de programación.

Si quisiéramos aplicar más estilos al párrafo como puede ser que el texto esté en cursiva, sería:

```
p {
    color: red; /* Con esta declaración todos los párrafos tendrán el color de letra rojo */
    font-style: italic; /* Debemos separar cada propiedad por punto y coma */
}
```

Nuestra hoja de estilos tendrá tantas reglas como queramos, si por ejemplo queremos además poner el heading `<h1>` de color azul sería:

```
p {
    color: red; /* Con esta declaración todos los párrafos tendrán el color de letra rojo */
    font-style: italic; /* Debemos separar cada propiedad por punto y coma */
}
h1 {
    color: blue; /* Pongo de azul el contenido de la etiqueta h1*/
}
```

3.1 Concepto de herencia

El término CSS es *Cascading Style Sheets* (Hoja de Estilos en Cascada). El concepto de herencia y de cascada son las dos características más importantes de CSS. En primer lugar, debemos saber que algunas propiedades de CSS se heredan desde los elementos padres a los hijos modificando el valor que tuvieran por defecto.

Por ejemplo, imaginemos que le aplicamos a la etiqueta `<body>` un color, haciendo que todos los textos que estén dentro de la etiqueta `<body>` de nuestro HTML se le aplique dicho color. Por ejemplo:

```
body {
    color: blue;
}
```

En este ejemplo estamos haciendo que todos los textos que estén dentro de la etiqueta `<body>` estén de color azul. Si tenemos una etiqueta `<div>` con texto en su interior y no tenemos ningún estilo de `color` para dicho elemento, el texto también aparecerá de color azul. Esto es debido a que la propiedad `color` en el caso de no darle ningún valor específico pasara a **heredar** el valor que tenga su elemento padre.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Aplicando estilos</title>
        <!-- Pongo el CSS aquí para que os resulte más fácil verlo en los apuntes,-→
            pero recordad que deberíamos ponerlo en un archivo externo para separar HTML de CSS  -->
```

```
        <style>
            body {
                color: blue;
            }
        </style>
    </head>
    <body>
        <h1>El título estará en azul</h1>
        <div id="container">
            <p>Este párrafo también estará en azul</p>
        </div>
    </body>
</html>
```

Debemos tener en cuenta que no todas las propiedades se heredan, hay propiedades como son los bordes que no se heredan. Veamos qué pasa si añadimos al ejemplo anterior un borde:

```
body {
    color: blue;
    border: 2px solid red;
}
```

Si la propiedad `border` se heredara todos los elementos que son hijos de `<h1>` (`<body>`, `<div>`, `<p>`) deberían tener un borde de color rojo, pero si nos fijamos no es así, sólo el `<body>` de nuestra web tiene esa propiedad. La herencia por tanto no ocurre con todas las propiedades CSS sino con algunas como `color` o `font`, a medida que vayamos trabajando con CSS nos iremos familiarizando con esto.

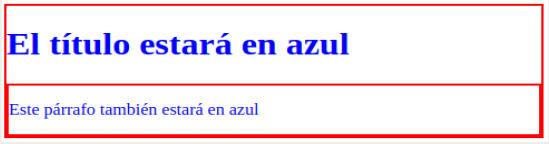
3.1.1 Valores especiales de herencia

Además de los valores de cada propiedad CSS, podemos aplicar unos valores especiales comunes a todas las propiedades existentes de forma que modifiquemos el comportamiento de la herencia para dicha propiedad. Estos valores son:

- **inherit**: hereda el valor que tiene la misma propiedad CSS que su elemento padre
- **initial**: establece el valor inicial que tenía la propiedad CSS inicialmente.
- **unset**: combinación de las dos anteriores, es decir, hereda la propiedad del elemento padre, y en el caso de no existir se le aplica su valor inicial.

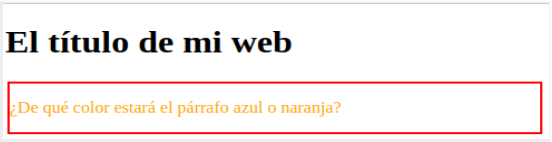
Por ejemplo, si queremos que en el ejemplo anterior al `<div>` se le aplique una propiedad que no se le aplica por defecto (propiedad `border` del elemento padre que es el `<body>`):

```
body {
    color: blue;
    border: 2px solid red;
}
div {
    border: inherit;
}
```



3.2 Concepto de cascada en CSS

Uno de los conceptos más importantes de CSS es el de **cascada**. Vamos a ver con un ejemplo qué significa el término de cascada en CSS. Imaginemos que tenemos dentro de nuestra hoja de estilos dos veces el mismo selector (en este caso el `<div>`)y para cada uno de ellos le hemos dado un valor diferente a la propiedad `color`.




```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Aplicando estilos</title>
        <!-- Pongo el CSS aquí para que os resulte más fácil verlo en los apuntes,-→
            pero recordad que deberíamos ponerlo en un archivo externo para separar HTML de CSS  -->
        <style>
            div {
                color: blue;
                border: 2px solid red;
            }

            div {
                color: orange;
                margin: 2px;
            }
        </style>
    </head>
    <body>
        <h1>El título de mi web</h1>
        <div id="container">
            <p>¿De qué color estará el párrafo azul o naranja?</p>
        </div>
    </body>
</html>
```

¿Que ha ocurrido? ¿Qué propiedad se va a aplicar? Si nos fijamos, tenemos dos propiedades color aplicadas al mismo elemento `<div>` y están al mismo nivel. En este caso la respuesta es fácil: siempre se va a aplicar la última regla definida, la cual va a mezclar las dos propiedades (`border` y `margin`) y sobreescribir aquella que esté repetida (`color`). Es como si en nuestra hoja de estilos tuviéramos lo siguiente:

```
div {
    color: orange; /* Se aplica el último valor que se encuentra en el caso de tener 2
                       propiedades con valores distintos */
    border: 2px solid red;
    margin: 2px;
}
```

Este no es el único caso donde podemos ver el concepto de cascada de CSS, más adelante seguiremos viendo cómo saber qué propiedad se aplicará a un mismo elemento

si accedemos a él por medio de la etiqueta, de su atributo `id` y/o de una `class`. Antes debemos saber cómo aplicar estilos mediante selectores.

4 Estilos CSS básicos

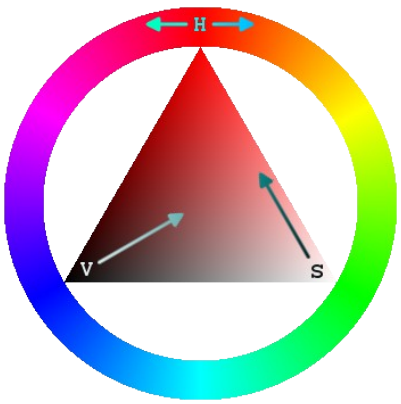
4.1 Tipografía

La elección de una tipografía de nuestro sitio web puede resultar bastante importante a la hora de que un usuario valore nuestro sitio web (color, tamaño letra, espaciado entre letras, interlineado, etc.). En este tipo vamos a ver cómo podemos modificar desde CSS estas propiedades.

4.1.1 Propiedad: color

La propiedad `color` nos permite modificar el color del texto. Los valores que puede tomar esta propiedad son:

- **Palabras clave de color**: Hay más de 140 palabras para indicar los colores como: `red, blue, orange, ...`
- **Esquema RGB**: Las siglas RGB (rojo, verde y azul) se expresan cada uno de los colores con un valor de 0 a 255. Antiguamente estos valores estaban entre paréntesis separados por comas. Ejemplo: `color: rgb (0 255 0);`, pero hoy en día se pueden separar por espacios. Ejemplo: `color: rgb (0 255 0);`
- **Esquema RGB hexadecimal**: es el más utilizado, consiste en dar el valor de cada color del RGB en hexadecimal comenzando con el carácter #. De modo que el ejemplo anterior sería: `color: #00FF00;` Podemos utilizar el formato abreviado cuando **los pares de cifras** son idénticos. En el caso anterior podríamos resumirlo en: `color: #0F0;`
- **Esquema HSL**: Las siglas HSL significan (color, saturación y brillo). La primera de ellas es el matiz del color (cifra de 0 a 360 grados), los siguientes valores son la saturación y el brillo indicados en porcentajes, estos valores como el esquema RGB puede ir separado por comas o espacios. Ejemplo: `color: hsl(180deg, 15%, 85%);`



Además de estos, podemos especificar también el canal alfa para establecer una transparencia parcial en determinados colores, el valor alfa se da en porcentajes de 0% a

100% o numérico de 0 al 1 con decimales. Este valor alfa se da si utilizamos rgb, hsl, o con el formato hexadecimal, y para ello pasaríamos a utilizar → rgba(...), hsla(...), ó #00FF0080;

Hay dos herramientas muy utilizadas para seleccionar el color HSL color picker y Adobe Color

4.1.2 Propiedad: font-family

Con esta propiedad podemos seleccionar el tipo de tipografía que queremos (Verdana, Arial, etc.) La forma de indicar qué familia tipográfica queremos será escribiendo su nombre, en el caso de ser un nombre compuesto por ejemplo 'Times New Roman', deberemos ponerlo entre comillas simples. Si el usuario final no tiene instalada la tipografía verá la letra con la tipografía por defecto. Una buena práctica es poner varias tipografías separadas por coma para que en el caso de no visualizar una que tenga otra opción como alternativa.

Veamos un ejemplo:

```
<!DOCTYPE html>
<html lang="es">
   <head>
      <meta charset="UTF-8">
      <title>Aplicando estilos</title>
      <!-- Pongo el CSS aquí para que os resulte más fácil verlo en los apuntes,-→
         pero recordad que deberíamos ponerlo en un archivo externo para separar HTML de CSS -->
      <style>
         p{
            font-family: Georgia, 'Times New Roman', Times, serif;
         }
      </style>
   </head>
   <body>
      <h1>Título de mi web</h1>
      <p>Mi perra me está dando con el hocico en el brazo</p>
</body>
</html>
```

Es importante elegir una fuente genérica al final del listado de nuestras fuentes para asegurarnos una fuente aceptable ya que no hay nada que nos asegure que el usuario tenga un tipo de fuente en concreto instalada. Las fuentes seguras son: serif, sans-serif, cursive, fantasy y monospace.

4.1.3 Propiedad: font-size

Con esta propiedad podemos especificar el tamaño que va a tener la fuente, los valores que puede tener son:

- **Medidas absolutas**: `xx-small, x-small, small, medium, large, x-large, xx-large`, por defecto el valor es `medium`.
- **Medidas relativas**: representan un tamaño más grande/pequeño que el actual: `larger, smaller`

- **Medida específica**: Indicamos los píxeles/porcentajes/pt/otra con el tamaño en concreto a usar. El punto (pt) es el que se recomienda para fijar el tamaño de la fuente.

Ejemplo:

```
p{
    font-size: 12pt;
}
```

4.1.4 Propiedad: font-style

Con esta propiedad podemos aplicar ciertos estilos muy utilizados para maquetar textos. Los valores que puede tomar son:

- **normal**: valor por defecto
- **italic**: Cursiva, estilo con ligera inclinación a la derecha, es la versión cursiva de la fuente creada por el diseñador
- **oblique**: oblícua, es igual al anterior salvo que esta inclinación se realiza de forma artificial. Muchas veces no se aprecia diferencia entre `italic` y `oblique`

4.1.5 Propiedad: font-weight

Con esta propiedad podemos indicar el grosor de la fuente. Los valores que podemos darle son:

- **Valores absolutos**: *normal* o *bold*. Por defecto el valor es normal.
- **Valores relativos**: `bolder` o `ligther`, dependiendo de si la queremos más gruesa o más fina a la actual.
- **Valor numérico**: un número de 100(menos gruesa) a 900(más gruesa), normalmente el valor es de 100 en 100.

4.1.6 Propiedad: font-variant

Permite convertir las letras minúsculas en mayúsculas dentro de un texto. La diferencia es que el tamaño de las que antes estaban en minúsculas es más pequeño aún siendo mayúsculas, y esto crea un efecto curioso. Valores:

- **normal**: Sin cambios.
- **small-caps**: Aplica el efecto mencionado anteriormente, así el texto “Me llamo Juan", quedaría como “ME LLAMO JUAN " más o menos.

Por ejemplo, a un elemento con la `id="destacado"` le vamos a aplicar una fuente `"Arial"`, de tamaño 24 puntos, en cursiva y negrita. Y además, que las letras minúsculas aparezcan como mayúsculas más pequeñas de lo normal.

```
#destacado{
    font-family: Arial;
    font-size: 24pt;
    font-style: italic;
    font-variant: small-caps;
    font-weight: bold;
}
```

ME LLAMO JUAN

4.1.6.1 fuentes externas

En lugar de depender de los tipos de letra instalados en el sistema del usuario, podemos cargar el/los archivo/s con nuestra propia fuente y obligar al navegador a utilizarla. Estos archivos deberán estar en formato True Type Font (ttf), Vectorial (svg), o OpenType Font (otf). En internet hay disponibles miles de fuentes así para descargar. Para ello, ponemos la regla `@font-face` para descargar una fuente de una web y cargarla en el navegador para utilizarla aunque no esté instalada en nuestro sistema. Esta regla se suele poner al principio del fichero CSS.

Para declarar un tipo de fuente a partir de un archivo debemos darle un nombre, e indicar el archivo (puede ser local o en una url), a partir del cual se obtiene la fuente:

```
@font-face {
    font-family: SuperFuente;
    src: url('fonts/superfnt.ttf');
}
```

Si disponemos de archivos para negrita o cursiva, por ejemplo, de la misma fuente, se pueden englobar siempre que los declaremos bajo el mismo nombre de fuente, e indiquemos el estilo con font-weight y/o font-style.

```
@font-face {
    /* Ahora le indicamos que debe buscar la versión en negrita de la fuente anterior*/
    font-family: SuperFuente;
    src: url('fonts/superfnt-bold.ttf');
    font-weight: bold;
}
```

Finalmente, para usar este fuente que hemos declarado simplemente, hay que aplicar la propiedad font-family con el nombre personalizado al elemento, por ejemplo:

```
p {
    font-family: SuperFuente;
}
```

Otra posibilidad, es utilizar las fuentes externas de Google. Cuando elegimos la fuente a utilizar nos indica el código html que debemos introducir dentro de nuestra etiqueta `<head>` y el nombre de la fuente a utilizar en la propiedad `font-family`.

Por ejemplo, imaginad que queremos utilizar la fuente Road Rage, debemos pulsar sobre “Select this style” y nos aparecerá qué debemos copiar en el head para poderla utilizar, en este caso es:

```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Road+Rage&display=swap" rel="stylesheet">
```

Además, nos pone el código CSS que debemos utilizar para seleccionar dicha fuente:

```
font-family: 'Road Rage', cursive;
```

Si esto lo aplicamos a un párrafo, el resultado sería:

Este párrafo tiene una fuente Road Rage

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Fuente externa</title>
        <link rel="preconnect" href="https://fonts.googleapis.com">
        <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
        <link href="https://fonts.googleapis.com/css2?family=Road+Rage&display=swap"
rel="stylesheet">
        <style>
            p {
                font-family: 'Road Rage', cursive;
            }
        </style>
    </head>
    <body>
        <p>Este párrafo tiene una fuente Road Rage</p>
    </body>
</html>
```

4.1.7 Textos y alineaciones

Con CSS los estilos que podemos dar en relación al espaciado, o interlineado son los siguientes:

4.1.7.1 propiedad: letter-spacing

Esta propiedad te permite establecer un espaciado **entre cada letra** de un texto(interletraje o tracking). Los valores que puede tener son: *normal* o un tamaño específico. Si el valor es un número negativo tendremos más unidas las letras, si el valor es positivo estarán más separadas.

4.1.7.2 propiedad: word-spacing

Esta propiedad te permite establecer un espacio que hay **entre una palabra y otra** en un texto. Los valores que puede tener son: *normal* o un tamaño específico.

4.1.7.3 propiedad: line-height

Esta propiedad te permite establecer el **interlineado** (altura entre líneas). Los valores que puede tener son: *normal* o un tamaño específico.

4.1.7.4 propiedad: text-indent

Esta propiedad te permite establecer las **sangrías** (indentación del texto). El valor sera un numero, por defecto es 0. Si el valor es negativo desplazará el texto a la izquierda, en el caso de ser positivo se desplaza a la derecha.

4.1.7.5 propiedad: white-space

Esta propiedad te permite establecer un cierto comportamiento de los espaciados. Los valores más habituales que suele tener son: *normal | nowrap | pre | pre-line | pre-wrap*. Por defecto el valor es *normal* (transforma múltiples espacios en blanco en un solo espacio consecutivo), pero las otras opciones nos permite:

- **normal**: Los espacios se transforman en uno solo, se ajusta al contenedor.
- **nowrap**: Los espacios se transforman en uno solo, ignora los saltos de línea.
- **Pre**: Respeta los espacios, pero ignora los saltos de línea
- **pre-wrap**: Respeta los espacios y se ajusta al contenedor
- **pre-line**: Respeta literalmente los espacios y suprime los espacios del final. Se ajusta al contenedor.

La diferencia entre pre-wrap y pre-line es que este último sí respeta liteteralmente los espacios que están antes del texto, mientras que si sobran después del texto los elimina.

4.1.7.6 propiedad: tab-size

Esta propiedad te permite establecer el ancho de las tabulaciones (espacio o tamaño). Los valores que puede tener son: un número o un tamaño específico.

4.1.7.7 propiedad: direction

Esta propiedad te permite establecer la dirección del texto, es decir, si el contenido, texto y otros elementos fluyen de izquierda a derecha o viceversa. Los valores que puede tener son: *ltr* o *rtl*, es decir, de izquierda a derecha o de derecha a izquierda respectivamente.

4.1.7.8 propiedad: text-align

Esta propiedad te permite justificar el texto. Los valores que puede tener son: *left | center | right | justify,* es decir, alineamos el texto a la izquierda, centro, derecha o justificamos el texto respectivamente, de la misma forma que podemos hacer con el Writer/Word.

4.1.7.9 propiedad: text-justify

Esta propiedad te permite establecer el método con el que el navegador justificará los textos. Los valores que puede tener son: *auto | inter-word | inter-character | none,* es decir, automáticamente el navegador elige cómo justificar, que sea una justificación entre palabras, una justificación entre caracteres, o que no haya justificación.

4.1.7.10 propiedad: text-overflow

Esta propiedad te permite modificar el comportamiento que tiene por defecto el navegador cuando un texto no cabe y se desborda. Los valores que puede tener son: *clip | ellipsis | texto,* es decir, desbordar el contenedor (valor por defecto), muestra puntos suspensivos cuando el texto no cabe, o mostrar un texto cuando no cabe y no queremos que muestre los puntos suspensivos.

4.1.7.11 propiedad: vertical-align

Del mismo modo que la propiedad text-align, esta propiedad nos permite alinear un elemento de forma vertical. Los valores que puede tener entre otros valores son:

- **top**: el elemento se posiciona en la parte superior del elemento padre.
- **middle**: el elemento se posiciona en la mitad del elemento padre
- **bottom**: el elemento se posiciona en la parte inferior del elemento padre.
- **text-top**: el elemento se posiciona en la parte superior del texto padre.
- **text-bottom**: el elemento se posiciona en la parte inferior del texto padre.

Cuando queramos alinear bloques de contenido o crear estructuras de diseño utilizaremos **flexbox** como veremos más adelante.

4.1.7.12 propiedad: text-decoration

Esta propiedad te permite subrayar el texto (*underline*), subrayar por encima del texto(*overline*) y tachar el texto(*line-through*) si se utiliza el valor *none* se elimina cualquiera

de los formatos anteriores. Esta propiedad se utiliza mucho para eliminar por ejemplo el subrayado que por defecto aparece con la etiqueta de enlace/hipervínculo.

4.1.7.13 propiedad: text-transform

Esta propiedad te permite convertir textos a mayúsculas (*uppercase*) o minúsculas(*lowercase*). También nos permite capitalizar el texto, es decir poner sólo la primera letra en mayúsculas independientemente de cómo esté escrito(*capitalize*).

4.1.7.14 propiedad: text-shadow y box-shadow

Con estas dos propiedades podemos dar sombras a los textos (text-shadow) o a otros elementos como cajas o contenedores(box-shadow).

text-shadow aplica una sombra a un contenido de texto, los valores que puede tener son:

- Valores posiciónX y posiciónY: permite indicar las coordenadas X e Y donde pondremos la sombra con respecto al texto o contenedor original.
- Valor size, para el caso del text-shadow es el valor del blur, es decir el radio que de desenfoque de la sombra medido en px, em, etc. Cuanto más pequeño sea el valor menos difuminada estará la sombra, cuanto más alto sea más borrosa se verá. En el caso del box-shadow se le puede dar dos valores de size, el primero será como el de text-shadow y el segundo hace referencia al factor de crecimiento de la sombra. Aumentando este valor podemos hacer que la sombra crezca hacia los lados.
- Valor color: podemos indicarle el valor que queremos que tome la sombra.

Por otro lado, la propiedad box-shadow aplica una sombra a una caja o contenedor. Puede tener los mismos valores que text-shadow junto con *inset* que sirve para indicarle que la sombra la queremos interna en lugar de externa que es la que aparece por defecto.

Veamos un ejemplo:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Aplicando estilos</title>
      <!-- Pongo el CSS aquí para que os resulte más fácil verlo en los apuntes,-→
         pero recordad que deberíamos ponerlo en un archivo externo para separar HTML de CSS -->
      <style>
         p {
            text-shadow: 5px -5px 2px #444;
         }
         div {
            box-shadow: 2px 2px 10px #666;
         }
      </style>
   </head>
   <body>
```

```
        <h1>Título de mi web</h1>
        <p>Mi perra me está dando con el hocico en el brazo</p>
        <div>normal</div>
    </body>
</html>
```

4.2 Fondo (background)

A lo largo de este punto vamos a ver cómo podemos poner un color de fondo o una imagen.

4.2.1 Propiedad: background-color

La propiedad `color` establecía el color del texto, la propiedad `background-color` establece el color de fondo del elemento. Todas las propiedades CSS donde existen valores color, pueden tomar los mismos valores que vimos en el punto de la propiedad `color`.

```
body {
    background-color: red;
}
```

4.2.2 Propiedad: background-image

Si queremos utilizar una imagen de fondo utilizaremos la propiedad `background-image` y en el valor será `url()` y dentro de los paréntesis la ubicación de la imagen.

```
body {
    background-image: url(imagen.jpg);
}
```

Una vez que hemos establecido la imagen que vamos a utilizar podremos personalizar cómo hacer que se repita (`background-repeat`), dónde ponerla (`background-position`), o cómo queremos que se comporte al hacer scroll(`background-attachment`).

4.2.3 Propiedad: background-repeat

Indica si la imagen se debe repetir o no hasta ocupar todo el ancho y alto del elemento (por defecto no se repite) sus valores pueden ser:

- **no-repeat**: No repetirá la imagen, la mostrará sólo 1 vez, aunque no ocupe todo el alto o el ancho del elemento.
- **repeat**: Repetirá la imagen, tanto horizontal como verticalmente, hasta ocupar todo el ancho y alto del elemento (valor por defecto).
- **repeat-x**: Repetirá la imagen, pero sólo en el eje x, hasta ocupar todo el ancho del elemento.
- **repeat-y**: Repetirá la imagen, pero sólo en el eje y, hasta ocupar todo el alto del elemento.

4.2.4 Propiedad: background-position

Indica la posición vertical y horizontal donde se debe situar la imagen dentro del elemento. Sus valores se pueden definir como:

- **posX posY**: Posición que ocupan en el eje vertical (y) y el horizontal (x) del elemento.
- **posX** puede tomar los valores left (izquierda), center (centrado), right (derecha), y % (siendo 0% el extremo derecho, y 100% el extremo izquierdo), coordX (un número de 0 hasta el ancho en píxeles del elemento).
- **posY** puede tomar los valores top (arriba), center (centrado), bottom (abajo), y % (siendo 0% la parte más alta, y 100% la más baja del elemento), coordY (un número de 0 hasta el alto en píxeles del elemento).

4.2.5 Propiedad: background-attachment

Indica si la imagen se debe mover al hacer scroll en la página o quedarse fija donde está. Sus valores son:

- **scroll**: La imagen se comportaría como esperamos, es decir, al hacer scroll se quedaría en su sitio junto con el resto de contenidos (opción por defecto si no ponemos esta propiedad).
- **fixed**: La imagen bajaría y subiría con nosotros al hacer el scroll (hasta que ya no fuera visible el elemento que la contiene).

Ejemplo de celda a la que se le asigna una imagen de fondo que no debe repetirse y que además se situará en la parte superior de la misma y centrada en el eje X:

```
td{
    background-image: url(imagen.jpg);
    background-repeat: no-repeat;
    background-position: top center;
}
```

Es posible establecer todas estas propiedades anteriores en una sola regla de CSS a modo de atajo, donde aunque no es estricto se recomienda utilizar el siguiente orden:

```
background: <color> <imagen> <repeat> <attachment> <position>;
```

5 Los selectores

Hasta el momento hemos visto cómo podemos aplicar un estilo a una etiqueta de nuestro código HTML. sin embargo, lo que estamos haciendo haciendo es seleccionando todos los elementos del documento que sean dicha etiqueta y aplicando el estilo a todos.

Por ejemplo, en el primer estilo que aplicamos al párrafo dándole el valor rojo (`color: red`):

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
```

```
        <title>Aplicando estilos</title>
        <!-- Pongo el CSS aquí para que os resulte más fácil verlo en los apuntes,-→
            pero recordad que deberíamos ponerlo en un archivo externo para separar HTML de CSS -->
        <style>
            p {
                color: red;
            }
        </style>
    </head>
    <body>
        <h1>El título de mi web</h1>
        <div id="container">
            <p>Primer párrafo de mi web</p>
            <p>Este es otro párrafo</p>
        </div>
</body>
</html>
```

En este ejemplo por tanto estamos diciéndole al navegador que todas las etiquetas `<p>` que encuentre en nuestra página le aplique el color rojo.

5.1 Seleccionar por ID (valor único)

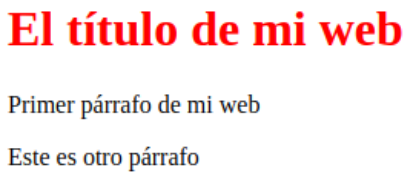
Hemos visto la forma más básica de seleccionar elementos en CSS utilizando como selector la etiqueta a la que le queremos aplicar el estilo. En el tema pasado, vimos que todas las etiquetas HTML pueden tener un atributo `id`,ese valor va a ser un **valor único** y por tanto no podremos encontrar dentro de nuestro HTML dos etiquetas con el mismo valor `id` (el `id` actúa como nuestro DNI, no podemos encontrar dos personas con el mismo DNI). Por ejemplo:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Aplicando estilos</title>
    </head>
    <body>
        <div id="header">
            <h1>El título de mi web</h1>
        </div>
        <div id="container">
            <p>Primer párrafo de mi web</p>
            <p>Este es otro párrafo</p>
        </div>
</body>
</html>
```

En el ejemplo anterior, tenemos dos elementos `<div>` cada uno con un `id` diferente, el primero de ellos tiene el valor `header`, y el segundo `container`. No podríamos poner otro `<div>` dentro de nuestro HTML con uno de esos dos valores ya que ya hay una etiqueta con ese valor.

A día de hoy no se suelen utilizar muchos ids (desde HTML5), pero son atributos válidos que se emplean para designar una zona que sabemos que no se va a repetir e identificarla como única. Si al código anterior HTML, le aplicamos el siguiente estilo:

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



En el anterior ejemplo, estamos dando un estilo accediendo con el selector `id`. Para ello utilizamos el símbolo `#`, es decir, estamos dándole el color rojo al texto que contenga el atributo: `id="header"`.

5.2 Seleccionar por clases

Cuando tenemos propiedades CSS que se repiten y no podemos asignarle a varias etiquetas distintas el mismo identificador utilizamos las clases de CSS. Toda etiqueta HTML puede tener el atributo `class`, la diferencia con respecto a los ids es que estos no tienen porqué ser únicas y por tanto podemos encontrarnos varias clases iguales en un mismo documento HTML.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Aplicando estilos</title>
    <style>
        .btn-page {
            background-color: aqua;
            color: white;
        }
        .btn-cancel {
            background-color: maroon;
            color: white;
        }
    </style>
</head>
<body>
    <form action="#" method="post">
        <button class="btn-page">Atrás</button>
        <button class="btn-page">Siguiente</button>
        <button class="btn-cancel">Cancelar</button>
    </form>
</body>
</html>
```

En el ejemplo anterior tenemos 3 botones, los dos primeros son botones para navegar en una página, y para establecerle el mismo estilo le hemos puesto una clase llamada `btn-page`. Por otro lado tenemos el botón Cancelar que le hemos puesto un estilo diferente (clase `btn-cancel`). Al usar `class` nos ha permitido con un sólo selector utilizar un mismo estilo en dos botones y podremos reutilizarlos si queremos aplicar ese estilo en más elementos de nuestro HTML.

En CSS si queremos acceder a una determinada clase, se accede mediante el carácter punto(.): Ejemplo: `.btn-page` , de esa forma estamos accediendo a todos los elementos de nuestro HTML que tengan el atributo: `class="btn-page"`

Además, en las clases podemos indicar en nuestro CSS qué tipo de elemento se trata, pudiendo utilizar ese mismo nombre de la clase en otros elementos HTML y darle un estilo diferente. Por ejemplo:

```
button.btn-page { /* Asignamos estilo sólo a los elementos button que tengan la clase btn-page */
    background-color: aqua;
    color: white;
}
p.btn-page{ /* Asignamos estilo sólo a los elementos p que tengan la clase btn-page */
    background-color: maroon;
    color: white;
}
```

5.3 Prioridad de selectores

¡IMPORTANTE! ¿Qué pasa si aplicamos varias veces la misma propiedad a un mismo elemento?. Hay un orden de prioridad establecido de mayor a menor: id, clase, elemento), ejemplo:

```
<p id="par1" class="cont">Soy un texto de color</p>
```

Si en el documento CSS pusiéramos:

```
p {
    color: red;
}
#par1 {
    color: green;
}
.cont {
    color: blue;
}
```

Los 3 afectan al elemento <p> de arriba, pero el color de su texto sería verde (green), ya que el identificador por id es el que mayor prioridad tiene. Si este se eliminase, se le aplicaría el color azul (blue), ya que el selector de clase es el siguiente que más prioridad tiene.

Si se modifica más de 1 vez la misma propiedad para un mismo elemento, pero ambas modificaciones tienen la misma prioridad, se le aplicará la última especificada (de ahí el nombre de Hojas de Estilo en Cascada). Por ejemplo:

```
.cont {
    color: red;
    color: blue;
}
```

Se aplicaría un color azul porque es la última modificación y ambas tienen la misma preferencia (clase).

5.4 Selecciones mixtas

Es posible utilizar varias clases en un mismo elemento HTML, para ello cada clase irá separada por espacios dentro del atributo `class`. De esta forma, a dicho elemento se le aplicará los estilos de cada una de las clases que tenga asignada. Por ejemplo:

```
<button class="btn blue btn-page">Atrás</button>
```

Por último, indicar que si queremos ser más específicos y aplicar un cierto estilo sólo a los elementos HTML que tengan todas las clases indicadas se realiza poniendo en nuestro CSS cada clase de forma consecutiva:

```
button.btn.blue.btn-page {
    ... /* Este estilo se aplicará sólo a aquellos elementos HTML button que tengan una clase btn,
           una clase blue y una clase btn-page. Si falla alguna de ellas no se aplicará */
}
```

5.5 Selectores CSS avanzados

Además de los selectores visto en los puntos hay una serie de métodos para seleccionar elementos dependiendo de la estructura del documento HTML, son los denominados combinadores CSS:

Nombre	Símbolo	Ejemplo	Significado
Agrupación de selectores	,	p, a, div {...}	Se aplican estilos a **varios elementos**
Selector descendiente		#page div { ... }	Se aplican estilos a elementos **dentro de otros**
Selector hijo	>	#page > div { ... }	Se aplican estilos a elementos **hijos directos**
Selector hermano adyacente	+	div + div { ... }	Se aplican estilos a elementos que **siguen a otros**
Selector hermano general	~	div ~ div { ... }	Se aplican estilos a elementos al **mismo nivel**
Selector universal	*	#page * { ... }	Se aplican estilos a **todos** los elementos

5.5.1 Agrupación de selectores

Si queremos asignar el mismo estilo CSS a varios selectores distintos, en lugar de crear el mismo bloque para cada selector, una buena práctica es simplificar nuestro CSS agrupandolo mediante una coma (,).

```
button.btn-page {
    background-color: aqua;
```

```
    color: white;
}
p {
    background-color: aqua;
    color: white;
}
```

De esta manera, el código anterior lo podríamos simplificar e indicarle al navegador que estas propiedades CSS se las aplique al conjunto de selectores indicado.

```
button.btn-page, p {
    background-color: aqua;
    color: white;
}
```

Una buena práctica para mejorar la legibilidad del código CSS es poner cada selector en lugar de uno detrás de otro, ponerlo cada uno en una línea separada. Esto hacemos que en grandes documentos CSS sea más fácil de leer y por tanto de revisar/modificar/mantener nuestro CSS.

```
button.btn-page,
p {
    background-color: aqua;
    color: white;
}
```

5.5.2 Selector descendiente

El selector descendiente es una forma de denominar a ciertos elementos que están dentro de otros. La forma de reflejar esto en CSS es mediante un espacio, por ejemplo:

```
div.container div{
    background-color: gray;
}
```

En el ejemplo anterior, estamos aplicando un fondo a todos los elementos `<div>` que estén dentro de otro `<div>` con `class` `container` (sean hijos, nietos, etc.) De esta forma, si hay un `<div>` fuera con la clase `container` no se le aplicarán los estilos. Ejemplo:

```
...
<main>
    <div class="container">
        <div class="articulo"> <!-- Se aplica -->
            <h2>Economía</h2>
            <p>...</p>
        </div>
        <div class="articulo"><!-- Se aplica -->
            <h2>Deporte</h2>
            <p>...</p>
        </div>
        <article>
            <div> <!-- Se aplica aun siendo "nieto" u no hijo directo-->
                <p>Otro ejemplo</p>
```

```
        </div>
    </article>
</div><!-- Cerramos el div con class container -->
<div><!-- No se aplica -->
    <p>Aquí no se aplicará</p>
</div>
</main>
<footer>
<div> <!-- No aplica -->
    <p>Página realizada por...</p>
</div>
</footer>
```

5.5.3 Selector hijo

El selector hijo lo utilizaremos cuando en lugar de querer seleccionar todos los elementos descendientes como hacíamos antes, seleccionar sólo los descendientes directos (sólo hijos, no habrá nietos ni sucesivos) del primer elemento especificado. Para ello utilizaremos el símbolo `>`.

```
div.container > div{
    background-color: gray;
}
```

Utilizando este selector, en el HTML anterior estaremos seleccionado sólo los `<div>` que en este caso tienen la clase `articulo`.

5.5.4 Selector hermano adyacente

Si queremos acceder únicamente a aquellos elementos que están directamente a continuación del elemento especificado (elemento hermano), utilizaremos el símbolo +, con esto seleccionaremos los hermanos que están seguidos el uno del otro, es decir, al mismo nivel.

```
div.container div+div {
    background-color: gray;
}
```

Con el ejemplo anterior de HTML y este CSS estamos seleccionado el `div` con la clase articulo que contiene información de deporte, ya que estamos seleccionando todos los divs que están a continuación de `<div class="container">` `<div>`. Si tuviéramos a continuación otro div de tecnología también se seleccionaría.

```
<div class="container">
    <div class="articulo"> <!-- Se aplica -->
        <h2>Economía</h2>
        <p>...</p>
    </div>
    <div class="articulo"><!-- Se aplica -->
        <h2>Deporte</h2>
        <p>...</p>
```

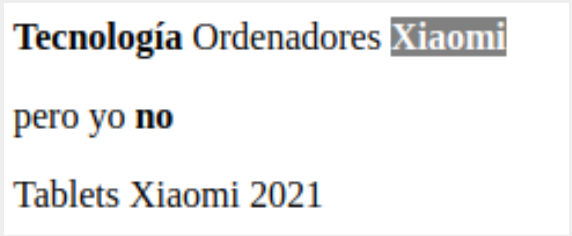
```
    </div>
    <div class="articulo"><!-- Se aplica -->
        <h2>Tecnología</h2>
        <p>...</p>
    </div>
    <article>
        <div> <!-- No se aplica al no ser hermano -->
            <p>Otro ejemplo</p>
        </div>
    </article>
</div>
```

Es importante tener en cuenta que el primer div (el de economía) no se selecciona al estar usándolo de base.

5.5.5 Selector hermano general

Si por lo contrario lo que queremos es seleccionar todos los hermanos en general sin tener en cuenta si son adyacentes o no, tendremos que utilizar el símbolo ~. Veamos un ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aplicando estilos</title>
    <style>
        div.container strong~strong {
            background-color: gray;
            color: white;
    }
    </style>
</head>
<body>
    <div class="container">
        <div class="articulo">
            <strong>Tecnología</strong>
            <span>Ordenadores</span>
            <strong>Xiaomi</strong> <!-- Yo soy su hermano -->
            <p>pero yo <strong>no</strong></p>
        </div>
        <div class="articulo">
            <span>Tablets</span>
            <span>Xiaomi</span>
            <span>2021</span>
        </div>
    </div>
</body>
</html>
```



Si nos fijamos, no es necesario que el elemento `<strong>` se encuentre adyacente al primero, sino que basta con que sean hermanos en el mismo nivel.

5.5.6 Selector universal

El selector universal se representa con un * y es la forma de aplicar ciertos estilos en todos y cada uno de los elementos HTML correspondientes. Por ejemplo, si utilizando el HTML anterior tenemos el siguiente código CSS:

```
div.container * {
    background-color: gray;
}
```

Estaríamos seleccionando todos los elementos que se encuentran dentro de la clase `.container`, que en este caso son los dos articulo.

El selector universal se suele utilizar para resetear propiedades de un documento como son los distintos márgenes que algunos navegadores ponen.

```
* { /* Eliminamos los márgenes internos y externos de todos los elementos del HTML */
    margin: 0;
    padding: 0;
}
```