Introducción a CSS

A la hora de afrontar el diseño web de un sitio, fundamentalmente vamos a acudir a dos tipos de documentos, que pueden formar uno solo o estar enlazados entre sí:

- Los documentos **HTML**, mediante los que dotaremos de estructura y contenido a cada página, es decir, indicaremos qué tipos de contenidos tendrán (imágenes, párrafos, títulos, subtítulos, etc), y en qué orden.
- Los documentos **CSS**, mediante los que dotaremos de estilo a las páginas, es decir, indicaremos aspectos como el tipo de letra, tamaño, colores, bordes, márgenes, etc.

Como veremos a lo largo de este tema, ambos conceptos son independientes, es decir, podemos tener páginas HTML que no tengan ningún estilo CSS (aunque no es lo habitual), y documentos de estilos CSS que no apliquemos a ninguna página (aunque serían inútiles). Así, estas dos herramientas de diseño pueden caminar en paralelo, vincularse la una a la otra en las páginas que nos interesen, o bien embeber los estilos en la propia página.

1. Historia

Para dotar de estilo a una página disponemos de lo que se llaman *hojas de estilo en cascada* (en inglés *Cascading Style Sheets*, abreviado **CSS**). Se trata de fragmentos de texto que definen estilos para los diferentes elementos de las páginas web, y que pueden ir tanto embebidos en la misma página web, como enlazados desde un fichero aparte. De esta forma se consigue separar por un lado el contenido (que irá en páginas HTML) del estilo (que irá en documentos CSS).

La primera versión de CSS (CSS1) fue publicada en 1996. En ella se definieron las directrices para ciertos estilos básicos, como tipos de letra, color de texto, alineación, márgenes, bordes...

Le siguió poco tiempo después CSS2, publicada en 1998, y que corregía y ampliaba las posibilidades de CSS1, añadiendo posicionamientos, sombras, textos bidireccionales, etc. Esta versión tuvo una revisión, la CSS2.1, que corregía algunos errores encontrados y eliminaba algunas funcionalidades que muchos navegadores no soportaban.

Finalmente, llegamos a la versión de CSS3. A diferencia de las anteriores, que son especificaciones globales de mucha información contenida, esta versión está formada por una serie de módulos, y cada uno añade un paquete de funcionalidades a lo que ya había en CSS2, por lo que se mantiene la compatibilidad con esta versión. De hecho, se empezó a trabajar con CSS3 en 1999, y cada módulo ha tenido una evolución diferente. Algunos se convirtieron en estándares pronto, y otros han tenido un camino más largo que recorrer.

2. Funcionamiento básico

Para definir un estilo sobre una página web en CSS, se utiliza lo que se llaman **reglas**. Una regla se compone de un **selector** (nombre del elemento sobre el que se va a aplicar el estilo) y, entre llaves y separados por punto y coma, las **declaraciones** o estilos que vamos a aplicar sobre ese elemento, como por ejemplo tipo de letra, tamaño, color, bordes... Cada declaración está formada por una **propiedad** y un **valor** (o valores) asociado a esa propiedad.

Así, si quisiéramos establecer que todos los párrafos de una página web fueran de color rojo y de tamaño 12 puntos tendríamos una regla como la siguiente:

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

Observemos que lo primero que ponemos es el selector, o nombre del elemento (en este caso, el nombre de la etiqueta HTML que hace referencia a los párrafos, `p`). Después, entre llaves, los diferentes estilos a aplicar, a base de declaraciones. En este ejemplo usamos dos declaraciones: una con la propiedad *color* para cambiar el color del texto a rojo, y otra con la propiedad *font-size* para cambiar el tamaño de letra a 12 puntos.

Si quisiéramos añadir otro estilo (otra regla) para que los encabezados de nivel 1 tengan color verde, podríamos añadirlo a continuación (o antes) del anterior:

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

h1
{
    color: green;
}
```

Veremos más adelante varios ejemplos de estilos que podemos incluir en estas llaves, divididos en categorías, pero primero vamos a ver cómo podemos aplicar estos estilos a una página HTML.

2.1. Aplicar los estilos a las páginas

Para aplicar un estilo o un conjunto de estilos CSS a una página web, tenemos varias alternativas:

Definir el estilo en un elemento concreto

Una opción es definir los estilos sobre un elemento concreto de la página en sí. Para ello empleamos un atributo `style` y en él colocamos las propiedades que queremos aplicar, junto con sus valores, finalizando cada declaración en punto y coma. Por ejemplo, si queremos que un párrafo concreto tenga el texto de color rojo y tamaño 10 puntos, podemos poner:

```
<p style="color: red;font-size: 10pt;">Párrafo de color rojo.</p>
```

Esta opción no se aconseja mucho, a no ser que sea un caso puntual de un elemento puntual con un estilo particular, ya que de lo contrario costaría mucho revisar todo el código HTML buscando estilos si hay que hacer algún cambio.

Definir los estilos en el bloque *head* de la página

Podemos también definir todo el conjunto de estilos de la página en su bloque `head`, empleando para ello una etiqueta `style` que englobe todas las reglas. Por ejemplo, para aplicar los dos estilos del ejemplo anterior (para párrafos y *h1*) a toda la página, pondríamos algo como esto en el head:

```
<html>
    <head>
        ...
        <style>
            p { color: red; font-size: 12pt; }
            h1 { color: green; }
        </style>
    </head>
...
```

Esta opción tampoco es muy recomendable si queremos aplicar el mismo estilo en diversas páginas, porque tendríamos que repetir el mismo bloque `style` en todas, con los consiguientes problemas de duplicidad que tendríamos si queremos hacer cualquier cambio.

Definir los estilos en un documento CSS aparte

Como tercera alternativa (y la más recomendable en la mayoría de casos), podemos definir todos los estilos en un documento CSS aparte, guardado con extensión *.css*, y después enlazar ese archivo con la página web, desde el `head` de la misma. Por ejemplo, si el archivo de estilos lo hemos llamado *estilos.css* y está en la misma carpeta que la página web donde aplicarlo, pondríamos algo así en el `head` (el atributo *type* es opcional):

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

Si el archivo estuviera dentro de una subcarpeta llamada *estilos*, la línea anterior cambiaría por esta otra (para indicar la subcarpeta):

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

Esta opción es la más recomendada porque queda todo centralizado en el archivo CSS, y cualquier cambio que haya que hacer se hace sobre el archivo y repercute a la vez en todas las páginas que lo enlacen. Además, de esta forma se puede dividir mejor el trabajo: una(s) persona(s) se encarga(n) del HTML, y otra(s) del CSS.

3. Estilos CSS básicos de texto

Veremos a continuación algunos estilos que podemos aplicar para modificar el formato del texto de una página HTML: tipos de letra, tamaños, alineaciones, colores...

3.1. Formato de carácter

Las siguientes propiedades afectan a cada uno de los caracteres del elemento HTML. Sería el equivalente a ir al menú *Formato* > *Fuente* o *Formato* > *Carácter* en editores como MS Word u OpenOffice/LibreOffice.

`font-family`

Para definir el tipo de letra utilizaremos la propiedad `font-family`.

En ella podemos poner un tipo de fuente concreto (entre comillas dobles si tiene espacios) como "Times" ó "Helvetica" o un tipo de fuente genérico. CSS define cinco fuentes genéricas: *serif*, *sans-serif*, *monospace*, *cursive* y *fantasy*.

Podemos indicar varios tipos de fuentes separados por comas. En este caso, el navegador elegirá la primera de las fuentes que tenga instaladas el sistema o bien aplicará la fuente que tenga establecida como predeterminada.

Es recomendable incluir al menos una fuente genérica al final de la lista de fuentes para asegurar una fuente aceptable y segura, ya que no hay garantía de que un tipo de fuente concreto esté disponible.

```
p { font-family: "Courier New", Arial, serif; }
```

`font-size`

Para definir el tamaño de la letra usamos la propiedad `font-size`, seguido del tamaño de la letra. Normalmente se expresa en puntos (pt), pero más adelante veremos que podemos utilizar otras unidades de medida.

```
h1 { font-family: Verdana; font-size: 30pt; }
```

`font-weight`

Para definir el grosor de la letra usamos la propiedad `font-weight`. Para simular un aspecto de negrita, se le da el valor de `bold`. Otros valores posibles son `normal`, `bolder` (aún más negrita de lo normal), `lighter` (más fina de lo normal), o un valor numérico (400 para normal, 700 para negrita...). Por ejemplo, si queremos que los *blockquotes* sean un poco más gruesos de lo normal, simulando negritas, podríamos poner:

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

`font-style`

Para definir el estilo de la letra usamos la propiedad `font-style`. Para simular un aspecto de cursiva, se le da el valor de `italic`. Otros valores posibles son `normal` u `oblique` (este último da un aspecto similar a *italic*). Por ejemplo, si quisiéramos que los *blockquotes* estuviesen en cursiva, podríamos poner:

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

`color`

Para definir el color de la letra se usa la propiedad `color`. Podemos poner un color simple en inglés (`blue`, `red`, `yellow` ...) como ejemplo. Más adelante veremos que existen otras formas de especificar muchos más colores.

```
h2 { color: blue; }
```

`text-decoration`

Para establecer la decoración del texto (es decir, otros efectos como subrayados, tachados, etc.), se usa la propiedad `text-decoration`. Si queremos hacer un efecto de subrayado, le daremos el valor `underline`. Si queremos un aspecto de tachado, le daremos el valor `line-through`. Podemos especificar otros valores, como `none` (ninguna decoración, valor por defecto para la mayoría de etiquetas HTML) u `overline` (subrayado por encima o sobrerrayado). Por ejemplo, si queremos poner un subrayado inferior en un *h1*:

```
h1 { text-decoration: underline; }
```

`text-transform`

Para definir ciertas transformaciones sobre el texto. Puede tomar los valores `none` (ninguna), `uppercase` (pasar el texto a mayúsculas), `lowercase` (minúsculas), `capitalize` (primera letra de cada palabra en mayúscula) o `full-width` (todas las letras de igual anchura). Así pondríamos todas las negritas en mayúscula:

```
strong { text-transform: uppercase; }
```

`text-shadow`

Define una sombra bajo el texto. Debemos indicar cuatro parámetros:

- Desplazamiento horizontal de la sombra (valores positivos para desplazar a la derecha, negativos a la izquierda)
- Desplazamiento vertical de la sombra (valores positivos para desplazar hacia abajo, negativos hacia arriba)
- Difuminación (*blur*) de la sombra
- Color de la sombra

Por ejemplo, así definimos una sombra sobre los párrafos con desplazamiento derecha de *2px*, inferior de *4px*, difuminación de *1px* y color rojo:

```
p { text-shadow: 2px 4px 1px red; }
```

> **Ejercicio 1:**
>
> Crea una página llamada **fuentes.html** con un encabezado h1 y un párrafo. Aplica los siguientes estilos al párrafo:
>
> - Tipos de fuentes: Trebuchet MS, Arial, sans-serif
> - Tamaño de fuente: 16pt
> - Grosor de letra: 900

> - Cursiva
> - Subrayado
> - Mayúsculas
>
> Y los siguientes estilos al encabezado:
>
> - Tamaño de la fuente: 22pt
> - El sombreado que quieras

3.1.1. Más sobre tipos de letra

Como hemos visto, la propiedad `font-family` nos permite aplicar un tipo de letra a un conjunto de elementos de la página. Dicha propiedad admite un listado de fuentes, de modo que se aplicará la primera que el sistema tenga instalada.

Web safe fonts

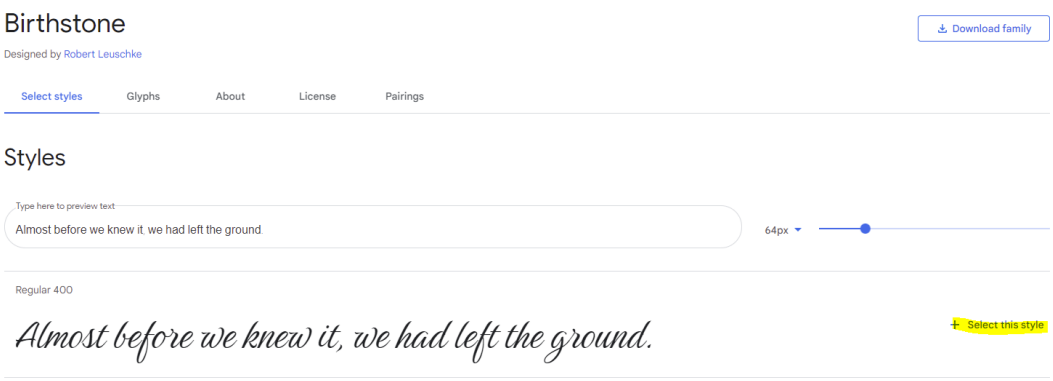
Esto nos plantea un problema: nuestra web puede verse diferente dependiendo de si un sistema u otro tienen instaladas unas u otras fuentes. Para intentar paliar este problema, existe un conjunto reducido de fuentes que están presentes en (casi) todos los sistemas, y que se conocen como *web safe fonts*. Algunos ejemplos de estas fuentes son *Arial*, *Courier New*, *Times New Roman* o *Verdana*. En algunas webs como esta se recoge un listado de *web safe fonts* actual, para poderlo consultar.

Utilizando fuentes externas

Además de utilizar *web safe fonts*, otra alternativa que tenemos es utilizar una fuente externa al sistema, de forma que, independientemente del sistema o dispositivo donde estemos visualizando la web, se intente cargar dicha fuente desde Internet.

Uno de los recursos más utilizados para buscar estas fuentes externas es Google Fonts. Podemos buscar distintas fuentes en base a categorías o palabras clave, y luego hacer clic en la que nos guste. Habrá diferentes estilos derivados de esa fuente, y podemos elegir el que queramos. Finalmente, se nos indicará el código HTML que tenemos que añadir en la cabecera (*head*) de nuestras páginas, y el nombre de fuente que deberemos incorporar en la propiedad `font-family`.

Por ejemplo, imaginemos que queremos aplicar la fuente *Birthstone*. Buscamos esta fuente en el buscador superior, y la elegimos de los resultados de búsqueda. Las fuentes pueden venir en diferentes grosores, así que elegimos el que más nos guste y pulsamos en el enlace *Select this style* de la derecha.



Aparecerá un panel donde se nos indicará qué tenemos que hacer para incorporar esta fuente a nuestras páginas. En este caso, deberemos incluir estas líneas en el encabezado (*head*) de cada página donde queramos incorporar la fuente, y deberemos incluirlas ANTES de incluir el estilo CSS que queramos aplicar:

```
<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=Birthstone&display=swap"
  rel="stylesheet">

<!-- Aquí incluimos nuestros estilos -->
<link rel="stylesheet" href="estilos.css">
```

Después, en la hoja de estilos, añadimos la propiedad `font-family` con el tipo de fuente que nos indican, para los elementos que queramos. Por ejemplo, para los párrafos:

```
p
{
    font-family: 'Birthstone', cursive;
    font-size: 14pt;
}
```

> **Ejercicio 2:**
>
> Elige una fuente que te guste de Google Fonts. Crea una página llamada **fuenteGoogle.html**, y carga esa fuente para aplicarla a los párrafos de la página. Comprueba que la fuente se visualiza correctamente.

3.2. Formato de párrafo

Las siguientes opciones afectan al comportamiento del texto del elemento como conjunto: su alineación, interlineado, etc. Sería el equivalente a las opciones del menú *Formato* > *Párrafo* en editores de texto como MS Word u OpenOffice/LibreOffice.

`text-align`

Para modificar la alineación horizontal del elemento, usamos la propiedad `text-align`, que puede tomar los típicos valores de cualquier procesador de textos: `left`, `right`, `center` o `justify`.

```
p { text-align: justify; }
```

`vertical-align`

Para modificar la alineación vertical del elemento (y alinearlo así verticalmente dentro de otro que lo contenga) está la propiedad `vertical-align`, que puede valer `top` (arriba), `bottom` (abajo) o `middle` (en medio), entre otros valores. Sin embargo, esta propiedad no suele funcionar muy bien en ciertos casos, por ejemplo, cuando queremos centrar un texto en el medio de un *div*. Dependerá en gran parte de si las dimensiones del elemento que lo contiene están bien definidas.

```
p { vertical-align: middle; }
```

`background-color`

Esta propiedad modifica el color de fondo del elemento en cuestión. Puede tomar los mismos valores que la propiedad `color` para el texto.

```
div { background-color: green; }
```

`text-indent`

Para modificar la tabulación o indentación del texto (el equivalente a una sangría izquierda), usamos la propiedad `text-indent`, seguida de una unidad (veremos más adelante diferentes tipos de unidades). Por ejemplo, para dejar una tabulación de 2 cm:

```
p { text-indent: 2cm; }
```

`line-height`

Para establecer el interlineado o separación de las líneas dentro del mismo elemento, como por ejemplo un párrafo, se usa la propiedad `line-height`, seguida de una medida. Por ejemplo, para un interlineado de 1,5 líneas usaríamos algo como:

```
p { line-height: 1.5em; }
```

`box-shadow`

Define una sombra bajo el bloque. Funciona similar a la propiedad *text-shadow* vista antes para el texto, pero añadiendo un nuevo elemento antes del color, que indica el grado de dispersión (*spread*). Un valor positivo incrementa el tamaño de la sombra, y uno negativo lo decrementa. Así definiríamos una sombra bajo un párrafo con desplazamiento horizontal de 2px (a la derecha), vertical de 4px (hacia abajo), difuminación de 1px y dispersión de -2px., de color rojo.

```
p { box-shadow: 2px 4px 1px -2px red; }
```

Ejercicio 3:

Crea un documento HTML llamado **estilosTexto.html**. Añádele en el `head` una etiqueta `style` con los estilos añadidos dentro. Vamos a definir los siguientes estilos:

- Los encabezados de primer nivel (*h1*) deberán tener letra Arial de 30 puntos, color rojo y subrayado.
- Los encabezados de nivel 2 (*h2*) deberán ser de letra Arial, color azul, tamaño 20 puntos. Deberán tener una alineación horizontal derecha.
- Los párrafos deberán ser letra "Times New Roman", tamaño 12 puntos, color gris, alineación horizontal justificada e interlineado de 1,5 líneas.

En el body del documento, prueba a crear un elemento al menos de cada tipo (*h1*, *h2* y párrafo de 3 o más líneas), para comprobar el estilo de cada cosa.

4. Los selectores

Hemos visto que una regla CSS se compone del nombre del elemento sobre el que se aplica el estilo (el selector), y de los estilos a aplicar entre llaves (declaraciones). En cuanto a los selectores, pueden ser de distintos tipos. Veremos a continuación un primer conjunto básico de selectores que podemos aplicar.

4.1. El selector de etiqueta

El selector de etiqueta aplica el estilo a todas las etiquetas HTML de ese tipo que haya en la página (y a las subetiquetas que estén contenidas en ella). Por ejemplo, si aplicamos el color rojo a los párrafos, también serán rojas todas las negritas y cursivas que haya dentro de los párrafos. Para utilizar este selector, basta con poner el nombre de la etiqueta HTML sobre la que aplicar el estilo, como en los ejemplos que hemos visto hasta ahora:

```
p { ... }
h1 { ... }
```

Si utilizamos el selector `body`, estaremos definiendo estilos generales para toda la página (porque todas las demás etiquetas están contenidas dentro de body). Por tanto, este selector se utiliza fundamentalmente para definir tipos de letra generales (por ejemplo, que toda la página tenga fuente *Arial*, sea cual sea el tamaño), color o imagen de fondo de la página, etc.

```
body { font-family: Arial; background-color: yellow; }
```

Podemos también aplicar los mismos estilos a un conjunto de etiquetas, poniéndolas todas juntas separadas por comas en el selector y luego, en otros selectores, definir estilos particulares para alguna de ellas:

```
h1, h2, h3 { color: blue; }
h1 { font-size: 30pt; }
h2 { font-size: 24pt; }
h3 { font-size: 20pt; }
```

4.2. El selector descendente

El selector descendente se utiliza para aplicar un estilo a una etiqueta sólo cuando esté contenida dentro de otra. Por ejemplo, si queremos que las negritas contenidas dentro de párrafos (directa o indirectamente) sean de color verde, usaríamos este selector:

```
p strong { color: green; }
```

Notar que se pone primero el nombre de la etiqueta contenedora, y después, separado por espacio, el nombre de la etiqueta contenida.

4.3. El selector de clase

El selector de clase es muy útil cuando no queremos aplicar un estilo a todas las etiquetas de un tipo en un documento, sino sólo a algunas de ellas. En este caso, se "marcan" esas etiquetas en el archivo HTML con un atributo `class`, y un nombre que queramos. Por ejemplo, si queremos que algunos párrafos de un documento tengan letra Tahoma, primero marcamos en el documento HTML esos párrafos con un atributo *class* y el nombre que queramos (por ejemplo, *miEstilo*). Además, puede haber otros párrafos que no tengan esa clase, como en este ejemplo:

```
<html>
    ...
    <body>
        ...
        <p class="miEstilo">Este párrafo es del estilo miEstilo.</p>
        <p>Este otro párrafo es normal.</p>
        <p class="miEstilo">Este párrafo también es de estilo miEstilo.</p>
...
```

Después, en el documento CSS añadimos el selector de clase, que tiene el nombre que le hemos dado a la clase, precedido por un punto (en nuestro ejemplo sería `.miEstilo`)

```
.miEstilo { font-family: Tahoma; }
```

Este selector también se aplica a otros elementos de otro tipo que tengan la misma clase. Por ejemplo, si pusiéramos un h1 con la misma clase, también tendría este estilo, y por tanto letra Tahoma:

```
<html>
    ...
    <body>
        ...
        <h1 class="miEstilo">Encabezado con estilo miEstilo</h1>
        <p class="miEstilo">...
...
```

Si queremos evitar esto, y aplicar el estilo sólo a las clases de una cierta etiqueta, se puede anteponer el nombre de la etiqueta al de la clase. Por ejemplo, si queremos que el estilo sólo se aplique a los párrafos con clase *miEstilo*, ponemos:

```
p.miEstilo { font-family: Tahoma; }
```

También podemos definir más de una clase para un elemento HTML, separando las clases por espacios dentro del atributo class:

```
<p class="miEstilo resaltado">Este párrafo es...</p>
```

Así, podríamos definir un estilo `.miEstilo` y otro estilo `.resaltado`, y cada uno se aplicaría a las etiquetas que lo tuvieran (las que tengan los dos, recibirán los dos estilos).

4.4. El selector de "id"

En un documento HTML, podemos identificar cada elemento con un identificador diferente. Esto se hace mediante un atributo `id` que podemos poner en cualquier etiqueta. Por ejemplo, un párrafo concreto puede tener el identificador *resumen*:

```
<p id="resumen">En resumen, esta página trata de ... </p>
```

No puede haber en una misma página dos elementos (aunque sean etiquetas diferentes) con el mismo identificador. Esto será útil cuando tratemos el tema de los formularios, más adelante, para identificar unívocamente cada campo, y para acceder a esos elementos desde JavaScript.

Una vez tenemos puesto un identificador en una etiqueta, podemos definir un estilo que sea único para ese identificador. Por ejemplo, si queremos que el párrafo resumen del ejemplo anterior tenga un tipo de letra diferente del resto (por ejemplo, Verdana color gris), en el CSS ponemos el nombre del identificador precedido de una almohadilla (#):

```
#resumen { font-family: Verdana; color: gray; }
```

Al igual que ocurría con el selector de clase, podemos anteponer el nombre de la etiqueta al nombre del *id*. Como el *id* es único, esto no tiene mucho sentido hacerlo, salvo si aplicamos el mismo CSS en varias páginas, y en otras puede haber otros elementos con el mismo id.

```
p#resumen { font-family: Verdana; color: gray; }
```

> **Ejercicio 4:**
>
> Indica qué selector utilizarías para definir un estilo para cada uno de estos casos:
>
> 1. Todos los elementos de una página
> 2. Todos los párrafos de una página
> 3. Un encabezado de nivel 2 con id "especial"
> 4. Todas las negritas de clase "clase1"
> 5. Todos las cursivas contenidas en encabezados de nivel 2

4.5. Reglas y solapamiento

¿Qué ocurre cuando hay varias reglas CSS que afectan a un elemento? Los estilos se acumulan y, en el caso de que sean incompatibles (por ejemplo, que una regla aplique un color de letra y otra aplique otro diferente) prevalece la que más se ajuste al elemento en cuestión. Por ejemplo, veamos estas dos reglas:

```
.prueba
{
    background-color: yellow;
    color: red;
}
p
{
    color: blue;
}
```

Si las aplicamos sobre un párrafo como este:

```
<p class="prueba">Párrafo de prueba</p>
```

Las dos reglas son aplicables sobre el párrafo. De todas ellas, los estilos que no choquen entre sí (como el color de fondo amarillo) se aplican directamente. En aquellas propiedades donde haya conflicto (como el color de texto, en este caso) se aplica el de la regla que más se ajuste al elemento. En este caso, es más restrictivo el selector `.prueba`, por lo que prevalecería el color rojo. Sin embargo, si definimos los estilos así...

```
.prueba
{
    background-color: yellow;
    color: red;
}
p.prueba
{
    color: blue;
}
```

... entonces es más restrictivo un párrafo de clase prueba, y encaja mejor con el elemento donde lo estamos aplicando, por lo que prevalecería el color azul definido en esa regla.

> **Ejercicio 5:**
>
> Dados los siguientes selectores:

```
p { ... }
p.miClase { ... }
p .miClase { ... }
p, #miID { ... }
p span em { ... }
.miClase .miClase2 { ... }
```

> Indica a qué textos del siguiente fragmento HTML se aplicarían:

```
<p>blablablabla <span>blablablabla</span> blablablabla</p>
<p class="miClase">blabla <span>blablabla<em>blabla</em></span> </p>
<h1 class="miID">blablablabla <span class="miClase">aaaa</span> </h1>
<p>blablablabla <span class="miClase">blablabla</span> blabla</p>
<p class="miClase">blabla <span class="miClase2">blabla</span> </p>
```

5. Unidades de medida y colores

Ya hemos visto algunas propiedades básicas de CSS para modificar el estilo del texto. Algunas de estas propiedades hacen uso de unidades de medida, por ejemplo, para definir el tamaño de la fuente, o de colores, para definir el color del texto o de fondo. Veremos ahora qué tipo de valores podemos dar en estos tipos de propiedades.

5.1. Unidades de medida

En CSS podemos utilizar dos tipos de unidades de medida: relativas o absolutas. Las primeras dependerán de algún factor externo (como la resolución de la pantalla, o el tamaño de fuente establecido), y las segundas no, miden siempre lo mismo independientemente de otros factores externos.

Unidades de medida relativas

Podemos distinguir las siguientes unidades de medida relativas:

`em`

Se utiliza para definir medidas relativas al tamaño en puntos utilizado actualmente. Por ejemplo, si el tamaño (por defecto o por CSS) es de 12 puntos y especificamos una medida de `0.7em`, estaremos multiplicando la medida predefinida (los 12 puntos) por 0.7, lo que equivaldría a 8.4 puntos de tamaño real. Así, *1em* equivale al 100% del tamaño por defecto.

```
p { font-size: 0.8em; }
```

De esta forma, partiendo de una medida base, se pueden definir todas las demás medidas relativas a ésa, y todo se escala en proporción.

`rem`

La unidad `rem` tiene muchas similitudes con la unidad `em`, pero se aplica para ámbitos distintos. *rem* es la abreviatura de *root em*, y por tanto, hace referencia al tamaño del elemento raíz (*html*). Así, si los párrafos tienen un tamaño de fuente de 12px y queremos hacer que las negritas que hay dentro de esos párrafos sean un 10% más grandes, podríamos hacer esto:

```
p { font-size: 12px; }
p strong { font-size: 1.1em; }
```

Sin embargo, si en lugar de eso ponemos `1.1rem`, entonces el tamaño base que estamos tomando no son los 12px del párrafo, sino el tamaño por defecto definido a nivel global de todo el documento (no para los párrafos). En resumen, utilizaremos `em` para hacer cambios de tamaño relativos a un elemento contenedor determinado, y `rem` para cambios relativos a la configuración global de la página.

`px`

Se utiliza para definir medidas en píxeles. Es una medida relativa porque su tamaño dependerá de la resolución del monitor.

```
p { font-size: 15px; }
```

`%`

También podemos utilizar porcentajes, de forma que el elemento afectado tomará ese porcentaje de su tamaño máximo permitido.

```
p { width: 70%; }
```

Unidades de medida absolutas

Las unidades de medida absolutas que podemos emplear son `cm` (centímetros), `mm` (milímetros) y `pt` (para dar medidas en puntos, normalmente para fuentes).

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

> **Ejercicio 6:**

> Crea un documento HTML llamado **tamanosEM.html**. Dentro, en el `head` con una etiqueta `style`, define estos estilos:
>
> - Todo el documento (body) tendrá un tamaño de letra de 20 puntos.
> - Un selector de clase llamado `pequeno` que ponga el tamaño de letra a 0.8em
> - Otro selector de clase llamado `grande` que ponga el tamaño de letra en 1.2em Los estilos deben aplicarse a tres párrafos diferentes en el documento (uno pequeño, uno grande y uno normal) para ver cómo cambia el tamaño de letra en cada uno de ellos. Añade también un *h1* para ver qué tamaño toma.

5.2. Colores

Podemos indicar una propiedad de color (como por ejemplo, la propiedad `color` para el color del texto, o `background-color` para el color de fondo) de diferentes formas:

Nombres de colores simples

Una primera forma sencilla consiste en utilizar alguno de los colores predefinidos que admite CSS. Se trata de colores simples, con el nombre en inglés: `black`, `white`, `yellow`, `blue`, `red` ...

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

Sin embargo, esta forma limita mucho la paleta de colores que podemos utilizar.

Colores RGB en formato decimal

Todos los colores se pueden descomponer en una combinación de los colores primarios, que en informática son el rojo (*Red*), el verde (*Green*) y el azul (*Blue*). Cualquier color se puede expresar como una combinación de valores de estos tres, y estos valores van desde el 0 (nada de ese color) hasta el 255 (todo ese color). Por ejemplo, el mismo color rojo se puede expresar como (255, 0, 0). El morado (rojo + azul) se puede expresar como (255, 0, 255). El blanco (unión de todos los colores) sería (255, 255, 255), y el negro (ausencia de color) sería (0, 0, 0).

Para indicar un color con este formato, se pone la palabra `rgb`, seguida de los tres valores numéricos entre paréntesis, separados por comas. Así, el ejemplo anterior se podría poner también como:

```
p { color: rgb(0, 0, 255); }
```

Notar que de esta forma se amplía mucho la paleta de colores disponibles, pudiendo tener 255 x 255 x 255 posibilidades (más de 16 millones de colores).

Colores RGB en formato hexadecimal

Los colores RGB se pueden expresar de forma más abreviada utilizando una codificación hexadecimal. Se sigue la misma filosofía que el caso anterior, pero en este caso, en lugar de dar valores de 0 a 255 para cada color primario, se dan los valores en formato hexadecimal (desde el 00 hasta FF, que equivaldría al 255 decimal).

Este formato, a pesar de que puede parecer el más enrevesado, es en realidad el que más se utiliza, porque cada color se representa así con 6 dígitos hexadecimales, precedidos por una almohadilla, lo que hace fácil identificar los colores en la hoja de estilos. Así, el negro se representaría por #000000, el blanco por #FFFFFF, y el rojo por #FF0000. El ejemplo anterior quedaría así:

```
p { color: #0000FF; }
```

Podemos encontrar en Internet diversas páginas con códigos de color, de forma que haciendo clic en el color que queramos, nos aparecerá el código hexadecimal que debemos añadir a la CSS. Aquí tenéis un ejemplo de página para consultar colores.

Ejercicio 7:

Crea un documento HTML llamado **coloresCSS.html**. Define un archivo de estilos CSS aparte, llamado **estilos.css**, y enlázalo con el documento HTML usando la etiqueta `link` dentro del bloque `head`. En ese documento CSS, define tres estilos con selector de clase: uno llamado *amarillo* para definir un color de texto amarillo, otro llamado *rojo* para definir un color de texto rojo, y otro llamado *verde* para definir un color de texto verde. Utiliza para ello formato RGB hexadecimal.

Después, añade tres párrafos al documento, cada uno con un estilo (y por tanto, un color) diferente.

Ejercicio 8:

En este enlace tienes un ejemplo de un ejercicio llamado *curriculum.html* de secciones anteriores, donde se pedía que añadiéramos los encabezados, párrafos y negritas/cursivas necesarios para elaborar un breve currículum sobre nosotros.

El documento HTML no tiene ningún formato asociado. Se pide ahora que crees una hoja de estilos llamada **estilos.css** y la asocies al documento. Los estilos que deberás aplicar son:

- Toda la página deberá tener un color de fondo `#B6EEF0` y un tipo de letra *Gluten*, de Google Fonts, con grosor de 400 aproximadamente, tamaño 12 puntos. El texto en general deberá estar justificado a ambos lados, con interlineado de 1.2 líneas.
- El título principal (*h1*) deberá tener una fuente *Arial* de tamaño 24 puntos, color `#0728F6`.
- Los títulos segundarios (*h2*) serán *Arial* tamaño 20 puntos, color `#455CED`.
- Los títulos de tercer nivel (*h3*) serán *Arial* tamaño 16 puntos, color `#6272D6`.
- Las negritas se pondrán en color `#455CED`.

Aplicando estos estilos al documento descargado, deberá verse así:

Mi currículum

Datos personales

Mi nombre es **Juan Pérez**, vivo en *Alicante*, en la *Avenida Doctor Rico*. Soy desarrollador web, y entre mis aficiones más destacadas está la natación, y salir a hacer senderismo siempre que puedo los fines de semana. Intento que el trabajo no se lleve por delante todo mi tiempo libre, y procuro quedar con mis amigos siempre que puedo.

Formación

Idiomas

Actualmente hablo cuatro idiomas: el **castellano** y **valenciano** a nivel nativo, el **inglés** a nivel avanzado (*C1*) y el francés a un nivel más intermedio (*B1*). Me gustaría empezar a aprender alemán, pero de momento es un proyecto que he dejado aparcado.

Estudios

Estudié primaria en el *CEIP Jaime I*, en San Vicente del Raspeig, y después hice bachillerato en el *IES Miguel Hernández* de Alicante. De ahí pasé a estudiar el **Grado en Ingeniería Multimedia** en la Universidad de Alicante, junto con un **Máster en Desarrollo de Aplicaciones Móviles** en esa misma Universidad. Actualmente me dedico al diseño de páginas web (en la parte *frontend*) y al diseño de interfaces para aplicaciones móviles.