

1.1 Mi primera página web

En la lección de hoy vas a aprender todo lo que necesitas para crear una aplicación básica con HTML y CSS, que son la base de una página web: con HTML escribes el contenido de la página y con CSS modificas el aspecto que va a tener.

Hoy aprenderás a:

- Crear un proyecto desde cero.
- Trabajar con HTML.
- Trabajar con selectores y cascada en CSS.
- Hacer hojas reset y de normalización.
- Trabajar con imágenes, colores y fondos.

La lección de hoy es un poco larga; nuestro objetivo es que al finalizar estas dos lecciones puedas hacer páginas básicas con HTML y CSS, y así **puedas aprender con la práctica**. No queremos explicarte teoría durante un montón de días y que mientras tanto no puedas practicar.

Tu objetivo es entender a rasgos generales lo que explicamos en estas dos lecciones. **Durante el resto del módulo vas a asentar sobradamente lo aprendido.**

1.1.1 HTML

HTML

HTML es un lenguaje de marcado, una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de su estructura: qué es un título, un enlace o un párrafo, por ejemplo.

Un elemento HTML suele estar formado por dos etiquetas, una de apertura y una de cierre. Entre esas etiquetas colocaremos el contenido, que podrá ser texto u otra/s etiquetas HTML. Las etiquetas de apertura pueden incluir unos modificadores que se llaman atributos y que modifican el comportamiento por defecto del elemento o aportan información extra:

Con el atributo "lang" indicamos que este párrafo está en español:

```
<p lang="es">Párrafo</p>
```

Nota: Hay una serie de elementos HTML que no necesitan etiqueta de cierre, los veremos más adelante.

A los elementos HTML los vamos a llamar "etiquetas", para abbreviar. Podríamos decir que hay dos tipos de etiquetas: las que definen el documento y las que definen el contenido.

Puedes consultar más información sobre la [lista de elementos html en la página de la MDN](#)

Etiquetas de página

Una página web empieza con una etiqueta que indica que es una página HTML, `<html>`. Dentro va una cabecera o `<head>` (donde se definen aspectos relativos al contenido, metainformación como el título, la descripción o palabras clave) y un cuerpo o `<body>` (donde incluiremos el contenido de nuestra página).

Esto es una página HTML con cabecera y cuerpo:

```
<!DOCTYPE html>
<html>
  <head> </head>

  <body></body>
</html>
```

Justo antes de la etiqueta `<html>` se debe añadir una etiqueta especial que indica qué tipo de documento HTML es: [doctype](#).

En el siguiente ejemplo vemos la misma página, un poco más definida, con su doctype, un atributo en el `<html>` que indica que está en español y dos etiquetas en la cabecera: una que indica la codificación del texto y otra que indica el título del documento.

Nota: Es importante acostumbrarnos a usar el atributo "lang" en nuestras etiquetas `<html>` para indicar el idioma en el que está escrito nuestro contenido.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Mi página</title>
  </head>
  <body></body>
</html>
```

Nota: `<meta>` es una de esas etiquetas que no necesita cerrarse.

Etiquetas de contenido

El navegador lee las etiquetas en orden de escritura, de arriba a abajo, y va a intentar mostrarlas en ese orden.

El buen uso de estas etiquetas hace que se añada al contenido una valoración semántica. La semántica es importante para:

- Accesibilidad (abreviado, *a11y*): ayudará, por ejemplo, cuando se consulta la página usando algún sistema de ayuda, como lectores de pantalla.
- SEO: posicionamiento en buscadores o motores de búsqueda, como Google, Bing o DuckDuckGo; el SEO facilitará que nuestra página aparezca en las búsquedas.

EJEMPLO: Si marcamos un texto como encabezado le estamos asignando una importancia diferente a cuando lo marcamos como párrafo o como elemento de una lista.

Codificación de una página HTML

Vamos a detenernos un momento en este punto: podemos usar varios juegos de caracteres al crear nuestra página; cada juego tiene más o menos caracteres, así que podría pasar que nuestras tildes o caracteres especiales no estén disponibles. Por eso usaremos `utf-8`, que hoy en día es el más completo.

La codificación de un documento se indica en dos pasos:

1. El archivo se guarda usando una codificación.
2. En el `<head>` de la página se incluye una etiqueta `<meta charset="">` que indica al navegador qué juego de caracteres hemos usado al guardar el archivo.

Nota: VS Code (y la mayoría de editores de código) ya guardan los documentos en `utf-8` por defecto. Esto es más algo que debemos simplemente comprobar ;).

Elementos en HTML

Títulos o encabezados

Vamos a ver nuestros primeros elementos en HTML: los títulos o encabezados. Estos se indican con las etiquetas `<h1>` a `<h6>`, de más relevancia a menos.

```
<h1>Encabezado de nivel 1</h1>
<h2>Encabezado de nivel 2</h2>
<h3>Encabezado de nivel 3</h3>
<h4>Encabezado de nivel 4</h4>
<h5>Encabezado de nivel 5</h5>
<h6>Encabezado de nivel 6</h6>
```

Párrafo

Con la etiqueta `<p>` definiremos una etiqueta del tipo párrafo e indicaremos que su contenido va a ser un párrafo de texto.

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
```

Lista de elementos

En algún momento vamos a necesitar añadir una serie de elementos e indicar que están relacionados entre sí: para ello tenemos las listas ordenadas (*ordered*) `` y las no ordenadas (*unordered*) ``. En ambas usamos `` para marcar cada elemento de la lista.

```
<ol>
  <li>Elemento de la lista</li>
  <li>Elemento de la lista</li>
  <li>Elemento de la lista</li>
</ol>

<ul>
  <li>Elemento de la lista</li>
  <li>Elemento de la lista</li>
  <li>Elemento de la lista</li>
</ul>
```

La lista ordenada produce una lista numerando cada elemento por orden de escritura, y la no ordenada añade un símbolo (que por defecto es un círculo) delante de cada elemento de la lista, también por orden de escritura.

Ejercicios

1. Página web Receta de gazpacho

Nota: este ejercicio es importante, intenta hacerlo mientras leas la lección.

Con estos elementos vamos a crear nuestra primera página con una receta de gazpacho fresquito. Añade el texto de la receta al fichero `index.html` (dentro de la etiqueta `body`) y añade las etiquetas HTML que necesites para que se parezca al texto que hay a continuación:

GAZPACHO

El **gazpacho** es una sopa fría con ingredientes como aceite de oliva, vinagre y hortalizas crudas: generalmente tomate, pepino, pimiento, cebolla y ajo.

Suele servirse fresco en los meses calurosos de verano. Su color varía desde el anaranjado pálido al rojo, según se empleen tomates más o menos maduros (que aportan un colorante natural denominado licopeno). El origen del actual gazpacho es incierto, aunque tradicionalmente se lo ha considerado un plato del interior de Andalucía, donde el aceite de oliva y los productos de la huerta son abundantes, y los veranos muy secos y calurosos. Por esta razón se lo conoce comúnmente como **gazpacho andaluz**. A pesar de ello el origen del gazpacho como plato "desmigado" es anterior al uso de hortalizas en su elaboración y data de la época del al-Ándalus.

Índice

1. El gazpacho andaluz
2. Composición del gazpacho actual
 - a. Ingredientes

El gazpacho andaluz

El gazpacho, denominado andaluz popularmente, suele ser definido por algunos autores culinarios como una mezcla entre sopa y ensalada. En la actualidad se emplea como un refresco en la mayoría de las ocasiones, y se sirve por regla general en verano. Suele servirse fresco como una bebida o comida de aromas agradables y reconfortantes. Si bien no está demostrado, hay autores que sospechan que su origen es Sevilla. Se denomina andaluz por haber trascendido así al resto de regiones de España y del mundo, pero en Andalucía se toman gazpachos blancos (un ejemplo es el gazpacho blanco cordobés) que no contienen tomate, y gazpachos rojos que sí los tienen. Los gazpachos rojos se elaboran en Andalucía occidental, los blancos en Málaga, Córdoba y Granada y los verdes en Sierra Morena y la Sierra de Huelva.

Composición del gazpacho actual

El gazpacho dispone de unas características organolépticas peculiares de sabor, aroma y color debido a la variedad de verduras que lo componen. El gazpacho andaluz emplea como ingredientes un conjunto de cinco hortalizas, que pueden variar en proporción según los gustos de la localidad, del cocinero o de la familia.

Ingredientes

- **Tomates:** deben ser bien maduros para que aporten dulzura; antes eran solo posibles en otoño, pero en la actualidad con el desarrollo de invernaderos y del transporte desde otras latitudes, es posible tener tomates casi durante todo el año. Esta hortaliza es la que brinda el color rojo al gazpacho debido a su contenido en licopeno (colorante natural en la piel y carne del tomate). Su exclusión o inclusión como ingrediente hace que se hable de "gazpachos blancos" o de "gazpachos rojos", respectivamente. En algunos períodos de escasez se ha empleado pimentón en lugar del tomate para lograr el color rojo.
- **Pimientos** (rojos o verdes): el pimiento es una verdura que proporciona frescura y sabores con ligeros toques agrios. En España esta variedad de pimientos no es picante.
- **Ajo:** el ajo en pequeñas cantidades proporciona un aroma característico, dependiendo de los gustos se añade más o menos. Una de las funciones en el gazpacho es la emulgente entre el aceite de oliva y las hortalizas.

- **Pan:** se emplea para aumentar volumen o espesar, pero si se emplea se reduce el carácter de bebida refrescante. En su elaboración tradicional suele emplearse restos de pan duro que habitualmente se remojan en agua.

Hay dos ingredientes que aparecen en casi todas las recetas actuales, incluidas las de notables "chefs". Estos son el pepino y la cebolla. Que si bien no se incluyen en la sopa básica, se pueden aportar como acompañamiento en forma de picada.

- **Pepino:** el pepino se lleva bien con el vinagre. Su sabor es fuerte y su proporción es una medida a tener en cuenta por quien prepara el gazpacho. Cuando su elaboración no incluye pepino se suele tildar de "gazpacho suave". Tanto las propiedades del pepino, como las del vinagre y las del agua fría, aliviaban la sed de los segadores que trabajaban en largas jornadas, bajo condiciones de temperatura extremas y con el único alimento durante el día de su gazpacho.
- **Cebollas:** en las recetas suele ponerse una cierta cantidad. Proporciona saborizantes naturales. Aunque existe la posibilidad de emplear zanahorias, es una costumbre poco habitual.

Aceite de oliva, vinagre, agua y sal son el resto de ingredientes. Suelen emplearse los de mejor calidad; bien conocido es el refrán popular «con mal vinagre y peor aceite, buen gazpacho no puede hacerse».

1.1.2 CSS

CSS

La maquetación web tiene mucha relación con la maquetación en papel de toda la vida, donde se utilizan los estilos para definir la apariencia que tendrá un cierto contenido.

En la web tenemos CSS (del inglés "Cascading Style Sheets" o, en español, "hojas de estilo en cascada"), que es un lenguaje de estilos para definir la presentación visual de un documento escrito en un lenguaje de marcado, como HTML.

CSS tiene una sintaxis simple y usa un conjunto de palabras clave en inglés para especificar los nombres de varias propiedades de estilo.

¿Y lo de "en cascada"? Esto hace referencia al proceso de combinación y aplicación de estilos en CSS y cómo se resuelven los conflictos entre ellos, pero eso lo veremos más adelante.

Hoja de estilos CSS

Entonces, en una hoja de estilos CSS tendremos un conjunto de reglas que dirán cómo se tienen que mostrar nuestros elementos. Supongamos que partimos de nuestro ejercicio anterior, donde tenemos maquetado un documento simple, y queremos cambiar el color y el tamaño de letra o el color de fondo de nuestra página.

Para esto tenemos las propiedades:

- `color: green` → pone el color del texto a verde
- `font-size: 18px` → pone el tamaño de letra a 18px
- `background-color: yellow` → pone el color de fondo de color amarillo

Vamos a decirle a nuestro encabezado `<h1>` que se muestre de color azul y a 20px de tamaño.

```
h1 {  
    color: blue;  
    font-size: 20px;  
}
```

Para aplicar estilos a uno o varios elementos de nuestra web, la estructura que debemos utilizar es la siguiente:

- Escribiremos el nombre de un selector, que es un texto que hace referencia a un elemento HTML, como por ejemplo, `h1` (que hace referencia a la etiqueta `h1`).
- A continuación añadiremos unas llaves.
- Dentro de esas llaves escribiremos los atributos CSS, que son un conjunto de reglas formadas por una clave y un valor, separados por `:` y acabados en `;`. Estas reglas son las que definirán los estilos que aplicaremos al selector que hemos definido previamente.

Nota: Como norma general escribiremos un espacio después de los dos puntos `:` en cada atributo CSS para facilitar la lectura del código. Esta es una práctica muy típica y se lleva a cabo en muchas de las empresas de programación.

Para el selector se puede usar:

- La etiqueta del elemento HTML (`h1`, `p`, `ul`, etc.)
- Un atributo del elemento HTML. Hay dos especiales que se usan para esto: el *id* y la clase (los veremos más adelante).
- Otros selectores más avanzados que iremos viendo poco a poco.

Dentro de nuestra página esto quedaría así:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Mi página</title>
    <style>
      h1 {
        color: blue;
        font-family: Arial, sans-serif;
        font-size: 20px;
      }
    </style>
  </head>
  <body>
    <h1>Título de mi página</h1>
    <p>Contenido de prueba de mi página web.</p>
  </body>
</html>
```

Declaración de estilos

Una forma de añadir estilos a una página es a través de la etiqueta `<style>` que, como aplica ajustes visuales sobre la página, iría dentro de la cabecera de mi documento. Resulta cómoda para manejar un único archivo.

Desde hace unos años intentamos separar el contenido de la presentación guardando nuestros estilos siempre en un archivo aparte, que estará enlazado desde nuestra página HTML. Esta opción es mucho más limpia para el desarrollo web en general. Lo haremos con la etiqueta `<link>`, que es una de esas etiquetas que no necesita cerrarse. Esta etiqueta lleva dos atributos, uno que dice el tipo de archivo que va enlazado `rel="stylesheet"` y otro diciendo dónde está el archivo `href="style.css"`.

index.html

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Mi página</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1>Título de mi página</h1>
    <p>Contenido de prueba de mi página web.</p>
  </body>
</html>
```

style.css

```
h1 {
  color: blue;
  font-family: Arial, sans-serif;
  font-size: 20px;
}
```

Al abrir nuestra web en un navegador, la etiqueta link indica al navegador que debe buscar y aplicar los estilos de la hoja de estilos enlazada, en este caso `style.css`.

Existe una tercera opción que son los estilos "en línea" (*inline styles*): declarar el estilo dentro del propio elemento HTML dentro del atributo `style`. Puede ser útil en una emergencia (por ejemplo para depurar un problema), pero no debe usarse en código de producción.

```
<body>
  <h1>Título de mi página</h1>
  <p style="border: solid red">Contenido de prueba de mi página web.</p>
</body>
```

En este ejemplo le ponemos un borde rojo al párrafo interior, lo que nos puede ayudar a localizarlo en la página.

Ejercicios

1. Trabajando con estilos

Añade una hoja de estilos al ejercicio Gazpacho de la minilección anterior donde:

- El color de fondo de la página sea `#f3f4f5`.
- El título tenga asignado un tipo de letra `Arial` a `24px` de tamaño y color de texto `black`.
- Para el texto de los párrafos, tendremos que usar la fuente `Georgia` a `18px` de tamaño y color de texto `#757575`.

1.1.3 Estándar de HTML5

Estándar HTML

Nota: esta mini lección es menos importante.

El estándar de HTML5 define las propiedades y comportamientos del contenido de una web, como texto, imágenes, videos o juegos, entre otros. Los siguientes grupos son reconocidos por crear las recomendaciones y estándares que aseguran el crecimiento de la web:

- **[W3C World Wide Web Consortium](#)**: Fundado en 1994, el Consorcio WWW se ha encargado de generar recomendaciones y estándares que aseguran el crecimiento de la web.
- **[WHATWG \(Web Hypertext Application Technology Working Group\)](#)**: Creado en 2004 por miembros de Apple, Mozilla y Opera (más tarde se unirían Microsoft y Google) en respuesta al lento ritmo del W3C a la hora de desarrollar estándares modernos de HTML.

Estos grupos impulsaron y desarrollaron gran parte del estándar de HTML5 y colaboraron a lo largo de los años, aunque en muchos casos los navegadores incluían los estándares propuestos por WHATWG antes de que fueran aprobados por W3C, convirtiendo este paso en una mera formalidad.

En 2018 los intereses de ambas organizaciones se separaron generándose dos estándares divergentes. Recientemente se ha llegado a un nuevo acuerdo en el cual WHATWG es el responsable de mantener el estándar de HTML con la colaboración de W3C.

Validadores de HTML5

Para validar que nuestro código HTML5 cumple con el estándar, asegurando unos mínimos de calidad técnica, podemos utilizar diferentes validadores, como:

- <https://validator.w3.org/>
- <https://whatwg.org/validator/>

BONUS: Accesibilidad

Os dejamos un par de enlaces muy interesantes sobre accesibilidad web:

- [Introduction to Web Accessibility and W3C Standards \(vídeo: 4:07\)](#)
- [Introducción a la accesibilidad web \(charla\)](#)

Ejercicios

1. Validar la página del Gazpacho

Asegúrate de que el HTML del Gazpacho que hemos realizado en las mini lecciones anteriores cumple con el estándar utilizando un validador.

1.1.4 Etiquetas de HTML

Etiquetas con semántica

Las etiquetas HTML nos permiten estructurar nuestro contenido según su función o carga semántica. Vamos a ver más etiquetas:

- para **definir nuestra página**
- para agrupar en **secciones**
- para identificar semánticamente el **contenido**
- para crear **tablas de datos**

Nota: Todavía no lo hemos dicho expresamente, pero lo normal es anidar las etiquetas, o lo que es lo mismo, meter unas dentro de otras.

```
<html>
  <body>
    <header>
      <h1>Título</h1>
    </header>
    <main>
      <section>
        <h2>Subtítulo</h2>
        <p>Contenido y más contenido</p>
        <p>Contenido con <a href="">enlaces</a></p>
        <ul>
          <li>lista</li>
          <li>de</li>
          <li>cosas</li>
        </ul>
      </section>
    </main>
  </body>
</html>
```

Secciones

Normalmente no solo vamos a querer meter nuestro contenido en la página y ya está, sino que querremos darle una estructura y agruparlo en bloques. Para ello tenemos la etiqueta `<section>`. Usaremos una sección para agrupar contenidos por temática.

Por ejemplo, en la página de un producto agruparemos la descripción del producto por un lado y los comentarios de las compradoras/usuarias por otro.

Hay una serie de secciones especiales que tienen asignado un significado semántico predeterminado:

- `<header>` : cabecera o sección de presentación de un bloque.
- `<main>` : indica la sección principal de contenido.
- `<footer>` : un pie o sección final de un bloque.
- `<nav>` : un bloque de navegación, para un menú.
- `<aside>` : un bloque de contenido de menor importancia o con contenido relacionado.
- `<article>` : un artículo, que aunque elimináramos el resto de contenido seguiría teniendo sentido por sí mismo.

Estos bloques especiales se pueden usar unos dentro de otros según tenga sentido: por ejemplo, un `<article>` puede tener cabecera y pie, mientras que una cabecera no debería tener pie.

Nota: Si usamos mal estos elementos el navegador no va a dar error, pero estaremos haciendo un favor muy pobre a quienes necesiten este extra de semántica para navegar (por ejemplo, una usuaria ciega).

Contenido

Dentro de estas secciones querremos incluir nuestros contenidos. Además de los encabezados, párrafos y listas, tenemos un juego importante de etiquetas.

Enlaces

Uno de los conceptos básicos de HTML es el enlace que nos permite vincular páginas o partes de ellas de manera que la información no quede como algo aislado sino conectado.

Un ejemplo es Wikipedia, donde en cada artículo se añaden enlaces relacionados que hacen que puedas seguirse para completar la información a medida que se va consultando.

El enlace se escribe con la etiqueta `<a>` y con un atributo `href=""` que indica adónde enlaza.

Podemos enlazar a:

- una página o un archivo
- un punto de la misma o de otra página

El primer enlace es muy fácil, simplemente pondremos la dirección de nuestra página o archivo como valor del atributo `href`:

```
<a href="https://www.wikipedia.org">Wikipedia</a>
```

El segundo tipo de enlace necesita de un atributo especial que es el `id=""`. Cualquier elemento de nuestra página puede llevar este atributo, pero al tratarse de un identificador no debe haber dos elementos con el mismo id en la misma página.

En nuestra página, vamos a identificar la cabecera y el contenido principal:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8" />
        <title>Mi página</title>
    </head>
    <body>
        <header id="top">
            <h1>Título de mi página</h1>
        </header>
        <main id="main">
            <h2>Texto en latín</h2>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
                tempor incididunt ut labore et dolore magna aliqua.
            </p>
        </main>
    </body>
</html>
```

Ahora podría añadir un enlace debajo del todo para ahorrarle el *scroll* a las usuarias poniendo en el `href` el símbolo `#` seguido del id que quiero enlazar:

index.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8" />
        <title>Mi página</title>
    </head>
    <body>
        <header id="top">
            <h1>Título de mi página</h1>
        </header>
        <main id="main">
            <h2>Texto en latín</h2>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
                tempor incididunt ut labore et dolore magna aliqua.
            </p>
        </main>
        <footer>
            <a href="#top">Volver arriba</a>
        </footer>
    </body>
</html>
```

Si quisiese enlazar al contenido principal de mi página desde otra página usaría:

product.html

```
<a href="index.html#top">Volver al principio de la página principal</a>
```

En estos dos casos se dice que las rutas son **relativas**, porque apuntan dentro de nuestro proyecto. Si incluimos el dominio donde está alojada la página o archivo, aunque sea en nuestro dominio, diremos que la ruta es **absoluta**.

Si conocemos una página que use id, podemos enlazar directamente a esa parte del contenido. Vamos a enlazar a la Wikipedia, justo a la parte donde se habla de la piratería en la Edad Media:

```
<a href="https://es.wikipedia.org/wiki/Piratería#La_Edad_Media">
    La piratería en la Edad Media</a>
    >
```

Aquí el atributo `href` lleva la dirección de la página de la Wikipedia sobre la piratería y el `id` de la sección que se refiere a la Edad Media.

Nota: Si quieres ver los enlaces relativos que estamos utilizando en esta página, ve arriba del todo y pulsa en los enlaces de cada uno de los ejercicios.

La etiqueta `<a>` tiene otros atributos que debemos conocer:

- `title=""`: donde podemos añadir un texto complementario que el navegador mostrará en un pequeño *tooltip* cuando pongamos el cursor sobre el enlace. Nos interesaría usarlo cuando tengamos un enlace tipo "descargar" y queramos asociarle un texto explicativo, como "Descargar archivo PDF". **Ejemplo:**

[Descargar](#)

Descargar archivo PDF

- `target=""`: aquí podemos especificar dónde se abre el enlace. Por ejemplo, con el valor `_blank`, indicamos que se abra en una nueva pestaña, lo cual nos interesa cuando en nuestra página enlazamos a páginas externas y no queremos que la usuaria "pierda" nuestra página al hacer clic en esos enlaces.

Negritas, cursivas

Tradicionalmente se usaban las etiquetas `` e `<i>` para poner un texto en negrita (*bold*) o en cursivas o itálicas (*italic*) respectivamente. Estas etiquetas todavía se utilizan, aunque no tienen carga semántica, simplemente muestran el texto visualmente en negrita o cursiva.

Existen unas etiquetas más recientes, `` y ``, que aunque visualmente hacen lo mismo (*strong* muestra el texto en negrita y *em*, en cursiva) sí que tienen una carga semántica, que sirve para indicar el nivel de énfasis o de importancia. La etiqueta `` indica un primer nivel de importancia del texto, y `` indica una importancia mayor.

```
<p>
    Dentro de este párrafo tenemos <em>un texto importante</em> y
    <strong>otro más importante</strong>
</p>
```

Nota: El aspecto visual de estas etiquetas es una convención entre los diferentes navegadores y, como veremos, se puede cambiar.

Imágenes

Muchas veces querremos acompañar nuestro contenido con imágenes, ya sea con fines ilustrativos (p. ej., imágenes de una noticia), o como motivo principal (p. ej., una galería de ilustraciones).

Para ello tenemos la etiqueta ``, que tiene varios atributos:

- `src=""`: aquí indicamos la ruta de nuestro archivo de imagen.
- `alt=""`: el atributo *alt* es el texto que va a mostrar el navegador en caso de que la imagen no se pueda cargar. También es muy importante para la accesibilidad, ya que será el texto que lean los dispositivos de apoyo como lectores de pantalla. Cuando la imagen sea parte del contenido debemos usarlo añadiendo un texto descriptivo. Cuando la imagen sea meramente decorativa, se recomienda dejar el valor vacío, pero no omitirlo. **Ejemplo:**



Saltos de línea

Aunque es recomendable usarla con tiento, tenemos una etiqueta que fuerza un salto de línea: `
`. Desde que empezamos a separar el contenido y el diseño de las páginas web, esta etiqueta ha quedado relegada a un lugar muy secundario,

pero todavía hay veces en los que tendremos que forzar una línea nueva en mitad de un texto y está bien conocerla.

```
<h1>Mi título genial <br />en dos líneas</h1>
```

Contenedores generales

A parte de las secciones tenemos un par de contenedores sin propósito específico que nos sirven para hacer agrupaciones sin carga semántica. Son el `<div>` y el ``. Mientras que el **div** es para bloques de contenido, el **span** está indicado para partes del texto o elementos en línea. Los iremos viendo más adelante.

Tablas

Hubo un tiempo en el que las tablas eran la base sobre la que se maquetaba cualquier página web. Hoy se utilizan para lo que son: presentar datos tabulados.

La tabla básica tiene una estructura bastante simple y tres etiquetas principales:

- una etiqueta que indica que se va a escribir una tabla
- una etiqueta para las filas
- una etiqueta para las celdas

En una imagen, una tabla de 3 filas y 3 columnas sería algo así:

```
<table>
  <tr> <td></td> <td></td> <td></td> </tr>
  <tr> <td></td> <td></td> <td></td> </tr>
  <tr> <td></td> <td></td> <td></td> </tr>
</table>
```

Y en código quedaría así:

```
<table>
  <tr>
    <td></td>
    <td></td>
    <td></td>
  </tr>
  <tr>
    <td></td>
    <td></td>
    <td></td>
  </tr>
  <tr>
    <td></td>
    <td></td>
    <td></td>
  </tr>
</table>
```

En principio, en las tablas siempre tiene que haber un número de celdas igual en cada fila.

Ejercicios

1. Organizando la semana

Nota: este ejercicio es importante, intenta hacerlo mientras leas la lección.

Crea una tabla con la comida de cada día de la semana usando `<th>` para las celdas que contienen los días.

2. Una página clásica

Nota: este ejercicio es importante, intenta hacerlo mientras leas la lección.

Construye una página semántica explicando algunos de los lenguajes de programación que vamos a aprender durante el curso, con:

- Un título principal.

- Contenido:
 - Tres párrafos de contenido.
 - Dos párrafos con anuncios secundarios.
 - Un listado de 4 lenguajes de programación.
- Un texto de copy (© 2020).

3. La MDN

Busca información sobre todos estos elementos en la [MDN](#) para ir conociéndolos y familiarizándote con ellos.

- Definición de página
 - doctype
 - html
 - head
 - title
 - link
 - meta * body
- Secciones de página
 - section
 - header, main, footer, nav, aside, article
- Contenido
 - div/span
 - h1-h6
 - p
 - blockquote
 - ol/ul, li
 - img
 - a
 - strong, em
 - small, abbr, sup, sub

- br, hr
- Tablas
 - table
 - tr
 - td, th
 - caption
 - thead
 - tbody
 - tfoot
 - col, colgroup

1.1.5 Selectores de CSS

Los navegadores ofrecen un cierto aspecto por defecto para todas las páginas. Nosotras lo podemos cambiar con CSS, creando estilos para definir la apariencia de nuestras páginas.

Para cambiar el aspecto de un elemento usamos un selector. Un selector hace referencia a las distintas formas que tenemos de especificar qué elemento vamos a seleccionar para cambiar su aspecto. Existen varios tipos de selectores:

- La propia etiqueta del elemento: `h1`, `a`, `p`, etc.
- Una clase que hayamos incluido con el atributo `class=""`.
- Un identificador en el atributo `id=""`.
- Una pseudo clase, que son unas palabras clave que, añadidas al selector, especifican un estado especial del elemento.
- Una mezcla de los anteriores.

Vamos a ver cada uno de los casos.

El propio elemento como selector

El estilo puede especificarse para cada etiqueta. Por ejemplo, si utilizamos el selector `p`, cambiaremos el estilo de todos los párrafos. No es lo ideal porque aplicaría a todos los elementos con esa etiqueta de la página, así que hay que usarlo con prudencia.

Podemos, por ejemplo, hacer que todos nuestros enlaces sean rojos.

```
a {  
    color: red;  
}
```

Clases como selectores

Las clases son palabras clave que asignamos a elementos HTML para agruparlos por función o apariencia, y diferenciarlos del resto de elementos de su mismo tipo. Por ejemplo: la clase "text-link" nos permite aplicar estilos particulares a los enlaces que lleven dicha clase sin afectar al resto de etiquetas.

```
<a href="#" class="text-link">Enlace de texto</a>
```

En CSS creamos clases para aplicarlas a grupos de elementos, como pueden ser todos los enlaces de texto, los elementos del listado de ingredientes o a los párrafos del pie de página. La manera de indicar en CSS que se trata de una clase es escribiendo primero un `.`:

```
.text-link {  
    color: red;  
}
```

id como selector

Ya habíamos visto que los *id* eran una palabra clave que usábamos como identificador para un único elemento de la página. En CSS también los podemos usar como selector, pero como no puede haber más de uno por página, no es recomendable usarlo salvo en casos especiales.

En una lista de acciones, por ejemplo, podemos tener unas clases para añadir estilos a los elementos del bloque y, además, añadir un identificador único para cada elemento.

```
<ul class="actions">  
    <li class="action">  
        <a id="add-user" href="" class="button">Nuevo usuario</a>  
    </li>  
    <li class="action">  
        <a id="rename-user" href="" class="button">Renombrar usuario</a>  
    </li>  
    <li class="action">  
        <a id="delete-user" href="" class="button">Eliminar usuario</a>  
    </li>  
</ul>
```

Y ahora podríamos usar el *id* para cambiar el tamaño del texto de uno de los elementos. Para ello, usamos la `#` seguida de la id como selector.

```
#add-user {  
    font-size: 24px;  
}
```

Pseudo clase como selector

Las pseudo clases son palabras clave que, añadidas a alguno de los selectores anteriores, especifican un estado concreto del elemento. El más usado es el estado de `hover`, que ocurre cuando colocamos el ratón encima del elemento.

Las pseudo clases se escriben usando el selector, seguido de `:` y la palabra clave para el estado.

Por ejemplo, como en uno de los ejemplos anteriores, tenemos un enlace al cual le vamos a poner el texto de color rojo, pero al pasar el cursor encima invertiremos los colores y lo mostraremos con fondo rojo y color blanco. Partimos del mismo `html` que anteriormente.

```
<a href="#" class="text-link">Enlace de texto</a>
```

Y el css sería:

```
.text-link {  
    color: red;  
}  
.text-link:hover {  
    background-color: red;  
    color: white;  
}
```

► [Codepen con ejemplo de hover](#)

Los selectores se pueden mezclar

En el ejemplo que acabamos de ver hay múltiples selectores. Mezclar selectores nos puede ayudar a contemplar casos particulares sin tener que usar identificadores.

Un elemento HTML puede tener tantas clases como queramos: las indicamos en su atributo `class` separadas cada una por un espacio.

Por ejemplo, si tenemos una lista de botones como la anterior:

```
<ul class="actions">
  <li class="action">
    <a href="" class="button button--new">Nuevo usuario</a>
  </li>
  <li class="action">
    <a href="" class="button button--rename">Renombrar usuario</a>
  </li>
  <li class="action">
    <a href="" class="button button--delete">Eliminar usuario</a>
  </li>
</ul>
```

Hemos eliminado el atributo `id` y hemos añadido una clase extra para cada tipo de botón. De esta manera tenemos por cada "botón" una clase general `.button` donde colocaremos los estilos comunes a todos los botones, y luego una particular (`.button--new`, `.button--rename` y `.button--delete`) donde solo pondremos los ajustes particulares.

Digamos que queremos que los botones tengan una caja con bordes redondeados, pero que el de Nuevo usuario tenga fondo verde, el de Renombrar azul y el de Eliminar, claramente, rojo muerte.

```
.button {  
    background-color: grey;  
    border-radius: 20px;  
    color: white;  
    display: inline-block;  
    margin-bottom: 1em;  
    padding: 10px 20px;  
    text-decoration: none;  
}  
  
.button--new {  
    background-color: green;  
}  
  
.button--rename {  
    background-color: blue;  
}  
  
.button--delete {  
    background-color: red;  
}
```

► [Codepen de ejemplo de composición de clases](#)

Selectores combinados

En un mismo selector podemos combinar varios elementos, lo que se conoce como "selector combinado" o "combinador" (*combinator*). En la práctica se suelen usar dos tipos de combinadores: madre (o hija) y descendiente.

El selector madre > se usa cuando una etiqueta es la madre de otra:

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

En este caso solo seleccionaremos los párrafos que están incluidos directamente en un artículo, pero no cuando aparezcan dentro de otros elementos intermedios:

```
<article>
  <p>Esta etiqueta es hija directa y por tanto aparece en azul.</p>
  <div>
    <p>Esta etiqueta no es hija directa y aparece en negro.</p>
  </div>
</article>
```

Cuando nos vale cualquier descendiente, sea directa o no, nos basta combinar con un espacio :

```
.action button {
  color: green;
}
```

En este caso, la regla se aplicará a cualquier etiqueta `button` que esté dentro de una etiqueta con clase `action`, aunque haya etiquetas intermedias.

```
<form class="action">
  <button>Botón hija directa, en verde</button>
  <div>
    Valores
    <button>Botón descendiente, también en verde</button>
  </div>
</form>
```

Puedes ver ambos ejemplos en [este Codepen](#). Podemos combinar selectores de clase, de etiqueta o incluso de identificador. Os preguntaréis: ¿qué pasa si varios selectores entran en conflicto? ¿Cuál se aplica en este caso? Lo veremos en la siguiente lección.

Ejercicios

1. ¡A jugar!

Para practicar los selectores, desde los más sencillos a los más complejos, te recomendamos [practicar con este divertido juego](#).

1.1.6 Herencia de CSS

Cuando los estilos se heredan, sus propiedades se transmiten a las hijas de una etiqueta.

Estilos heredados

Hay una serie de propiedades de CSS que se heredan por defecto: se transmiten a las etiquetas que están dentro de una etiqueta. Un ejemplo es `color`: si aplicamos esta propiedad a una etiqueta, todas las etiquetas anidadas heredarán el mismo color.

Otros estilos no se heredan, por ejemplo `border`: el borde se aplica a un elemento determinado, no a los elementos que contiene.

Mira el siguiente bloque de HTML:

```
<article style="color: blue; border: solid">
  <h1>Mi primer artículo</h1>
  <p>Este artículo va sobre CSS.</p>
</article>
```



Mi primer artículo

Este artículo va sobre CSS.

Resultado: el borde azul no se hereda.

Fíjate cómo el atributo `color` declarado en `article` se transmite a sus hijas `h1` y `p`, mientras que el atributo `border` no. Generalmente podemos intuir qué elementos se heredan por defecto y cuáles no: no tiene mucho sentido heredar los bordes de un elemento.

Modificando la herencia

Podemos hacer que una etiqueta herede también el atributo `border`, cambiando su valor a `inherit`:

```
<h1 style="border: inherit">Mi primer artículo</h1>
```



El título también adquiere un borde azul.

Nota: recuerda que la declaración de estilos en línea debe usarse solo como última alternativa.

En este [Codepen](#) encontrarás una declaración más elegante para la misma presentación.

Para modificar la herencia podemos aplicar las siguientes palabras clave:

- `inherit`: activa la herencia, como acabamos de ver.
- `initial`: fija la propiedad a su valor por defecto. En esencia, "desactiva" la herencia.
- `unset`: devuelve la propiedad a su valor por defecto: si la propiedad normalmente se hereda actúa como `inherit`, y de otro modo como `initial`.

Hay otro par de palabras clave, `revert` y `revert-layer`, menos usadas, que resetean una propiedad al estilo por defecto del browser o de la capa, respectivamente.

Podemos cambiar la herencia de (casi) todas las propiedades de un elemento con la propiedad especial `all`:

```
h1 {  
  all: initial;  
}
```

Probad a descomentar esta propiedad en el [Codepen de arriba](#): veréis que el título `h1` aparece como texto normal, ya que reseteamos todas sus propiedades a las iniciales del documento (antes incluso de aplicar las de `h1`).

Cascada y especificidad de selectores

CSS significa *cascading style sheets*, o en español: "hojas de estilo en cascada". La "cascada" se refiere al proceso de combinación y aplicación de estilos en CSS y a cómo se resuelven los conflictos entre ellos.

Acabamos de ver que, a veces, varios selectores se aplican al mismo elemento. ¿Cuál se mostrará cuando esto ocurre? El algoritmo de cascada es quien decide en último lugar qué propiedades se aplican y cuáles se ignoran.

La cascada depende de 3 factores, de menos importante a más:

1. El **orden** en el archivo CSS: si varios selectores tienen el mismo "peso" ganarán los que estén más abajo, porque el CSS se aplica en orden de escritura.
2. La **especificidad**: cuanto más específico es un selector más peso tendrán sus reglas sobre las demás.
3. La **importancia**: hay una palabra clave (`!important`) que hace que nuestra propiedad se aplique siempre.

Vamos a ver cada uno en detalle.

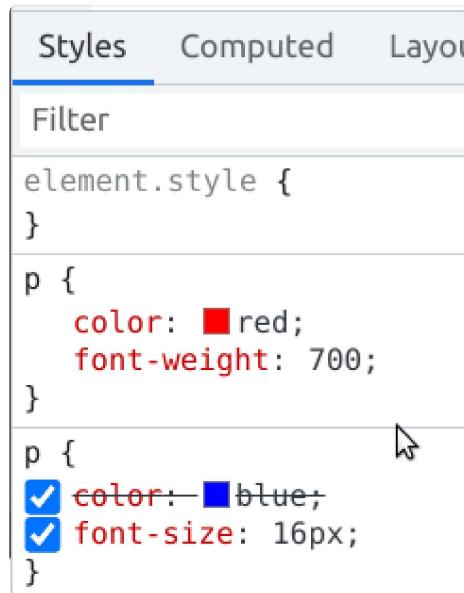
Orden de selectores

Según se recorre el archivo CSS las reglas que se encuentren van sobrescribiendo las anteriores, hasta que al final la última es la que cuenta. Para cada regla se

consideran los atributos por separado:

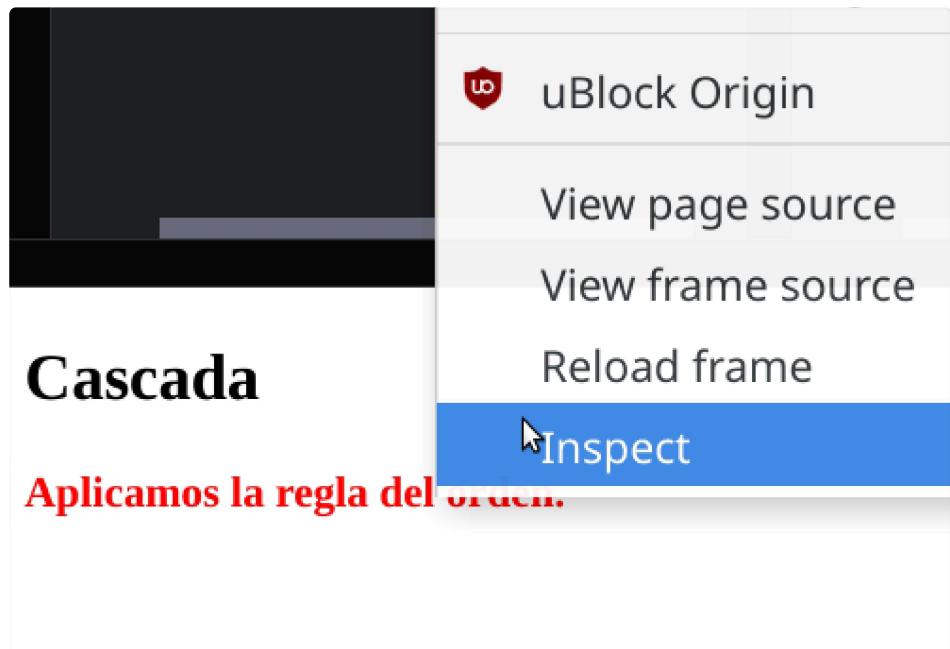
```
p {  
  color: blue;  
  font-size: 16px;  
}  
/* ... más abajo ... */  
p {  
  color: red;  
  font-weight: 700;  
}
```

La segunda regla sobrescribe la propiedad `color`, mientras que siguen aplicando el `font-size` de arriba y el `font-weight` de abajo. Como se puede ver en [este Codepen](#), el resultado será que se sobrescribe solo el color azul de la primera regla, que está repetido en la segunda, pero se sigue aplicando el `font-size` que no está sobrescrito:



Se ignora el color azul primero.

Puedes obtener esta vista en Chrome seleccionando "Inspect" con el botón derecho sobre un elemento visual y luego yendo a la pestaña "Styles".



Botón derecho sobre el párrafo.

Veremos el texto en color rojo, tamaño 16 puntos y negrita (`font-weight` a 700):

Cascada

Aplicamos la regla del orden.

Texto en rojo y negrita.

El orden se tiene en cuenta solo cuando la especificidad de las reglas es la misma, así que vamos al siguiente punto.

Especificidad

El siguiente aspecto que determina la aplicación de las propiedades es cómo de específico es cada selector: un selector más específico gana siempre a uno más genérico. El selector menos específico es el de etiqueta:

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

El siguiente en peso es el selector de clase:

```
.type {  
    /* ... */  
}
```

Un selector de clase tiene siempre más peso que uno de etiqueta, así que en caso de conflicto nos quedaremos con el selector de clase.

A su vez, un selector de id tiene más peso que uno de clase:

```
#identifier {  
    /* ... */  
}
```

Pero hay un selector con más peso todavía, y que gana siempre en caso de conflicto: el del atributo `style=""`.

```
<p style="/* ... */>  
    Texto random  
</p>
```

Por tanto, el orden de especificidad es:

atributo > id > clase > etiqueta

Selectores combinados

¿Qué pasa cuando se combinan varios componentes en un selector? Por ejemplo:

```
.main > p .type {  
    /* ... */  
}
```

En este caso se tiene en cuenta la especificidad de la categoría con más peso:

- Si dentro de nuestro selector combinado hay un selector por id, ganará a uno que solo tenga selectores de clase.
- Si hay selectores por clase ganarán siempre a los selectores por etiqueta.
- Y los selectores por etiqueta ganan solo si no hay ni por clase ni por id.

Mira este ejemplo:

```
#identifier {
  color: red;
}
.type {
  color: blue;
}
p {
  color: green;
}
```

Lógicamente, primero aplicaremos `#identifier`, ya que estamos identificando una etiqueta directamente; luego `.type`, donde hemos asignado una clase manualmente; y por último `p`, que es una etiqueta genérica.

Si hay múltiples selectores combinados con la misma categoría, ganará el que tenga más selectores. Por ejemplo, en este caso:

```
article > div > p {
  color: red;
}
div > p {
  color: blue;
}
p {
  color: green;
}
```

El primer selector combinado se tendrá en cuenta porque especifica una secuencia de tres etiquetas; después el segundo con solo dos etiquetas; y por último el tercero donde solo estamos especificando una etiqueta.

En este [Codepen](#) podéis ver varios selectores combinados. ¿Por qué hay una línea que aparece en negro?

Nota: ten en cuenta que los selectores muy específicos son un arma de doble filo, por lo que es una buena práctica escribir los selectores lo más genéricos posible.

Importancia

El tercer factor que se considera en caso de conflicto es la importancia: una propiedad marcada como `!important` se aplicará siempre antes que todas las demás. Incluso aunque haya una regla con `id` que se le aplique directamente.

Alerta: úsala con prudencia. Cuando en un proyecto se abusa del `!important` siempre llega un momento en el que queremos cambiar un estilo y no podemos porque ya tiene esta palabra clave. Por ello se recomienda usar `!important` lo menos posible y en su lugar usar la especificidad. La mencionamos solo para que sepáis que existe, pero ni siquiera os vamos a enseñar a usarla 😅.

Recursos externos

- [Más info sobre herencia en la MDN](#)
- [Cascada y Herencia MDN.](#)
- [CSS Specificity Wars](#)

Ejercicios

1. Coloréame esos links

El color es una de las propiedades que se heredan, así que si tenemos esta estructura ([Codepen](#)):

```
<article>
  <h2>Título</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit</p>
  <p>Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</p>
  <aside class="links">
    <h3>Enlaces relacionados</h3>
    <ul>
      <li><a href="#">Enlace 1</a></li>
      <li><a href="#">Enlace 2</a></li>
      <li><a href="#">Enlace 3</a></li>
      <li><a href="#">Enlace 4</a></li>
    </ul>
  </aside>
</article>
```

y al `<aside>` con clase `.links` le aplicamos una regla que ponga el texto rojo, ¿qué quedará en rojo?

2. Herencia para todas

Nota: este ejercicio es importante, intenta hacerlo mientras leas la lección.

Partiendo del ejercicio 1 vamos a hacer uso de la herencia, y utilizando un solo selector de CSS vamos a hacer que:

- el tamaño de fuente de la página sea de `18px`
- el tipo de fuente sea `Tahoma`
- el color de fuente sea `rebeccapurple`
- el color de fondo sea `mediumpurple`
- el interlineado sea de 1,5

Pista: Tenemos que trabajar con la primera etiqueta del contenido de `html` que es `<body>`, y utilizar el selector de etiqueta para que todas sus hijas anidadas hereden de ella.

3. Conociendo la especificidad

Nota: este ejercicio es importante, intenta hacerlo mientras leas la lección.

Partiendo de este [Codepen de ejemplo](#):

1. ¿Por qué los enlaces son verdes y no rojos?
2. Haz que los enlaces sean rojos.
3. Rehaz el HTML usando `<div>` en lugar de `` y ``. ¿Qué pasa?
4. Comenta el CSS que no se puede tocar y reescríbelo usando solo selectores de clase, es decir, sin utilizar selectores de etiqueta.
5. BONUS: Cambia ahora entre `<div>` y ` / `.

1.1.7 Colores y fondos

Valores de los colores

Para empezar, vamos a ver los distintos formatos que podemos usar para indicar colores, por ejemplo como valor de nuestra querida propiedad CSS `color`.

Colores por nombre

La primera forma de indicar un color es mediante la palabra clave que indica el nombre del color. Podemos usar un montón de palabras clave para colores que podéis ver en [la última especificación del consorcio W3](#). Nosotras nos quedaremos por ahora con los [colores básicos](#).

Color names and sRGB values

Named	Numeric	Color name	Hex rgb	Decimal
		<i>black</i>	#000000	0,0,0
		<i>silver</i>	#C0C0C0	192,192,192
		<i>gray</i>	#808080	128,128,128
		<i>white</i>	#FFFFFF	255,255,255
		<i>maroon</i>	#800000	128,0,0
		<i>red</i>	#FF0000	255,0,0
		<i>purple</i>	#800080	128,0,128
		<i>fuchsia</i>	#FF00FF	255,0,255
		<i>green</i>	#008000	0,128,0
		<i>lime</i>	#00FF00	0,255,0
		<i>olive</i>	#808000	128,128,0
		<i>yellow</i>	#FFFF00	255,255,0
		<i>navy</i>	#000080	0,0,128
		<i>blue</i>	#0000FF	0,0,255
		<i>teal</i>	#008080	0,128,128
		<i>aqua</i>	#00FFFF	0,255,255

Colores básicos en la especificación del W3C.

Ejemplo: el [rosa fucsia](#).

```
p {
  color: fuchsia;
}
```

Colores en hexadecimal

¿Qué son esos valores misteriosos a la derecha de cada color de la tabla? De forma equivalente a las palabras clave, podemos expresar un color con formato hexadecimal. En este formato declaramos un color con una almohadilla `#` y sus 3 componentes RGB - R (rojo), G (verde), B (azul). Cada uno de los componentes se representa con 2 dígitos en hexadecimal, es decir, cada dígito puede tener 16 valores, entre 0 - 9 y A (10) - F (15). Podemos usar mayúsculas o minúsculas a nuestra elección. Por ejemplo, el color fucsia se compone de una componente máxima de rojo (ff), nada de verde (00) y máxima de azul (ff).

```
p {  
    color: #ff00ff;  
}
```

Suele ser habitual expresar algunos colores comunes de forma simplificada. Si los dígitos de cada componente son iguales (por ejemplo, `ff`) puede escribirse el color de una forma simplificada escribiendo solo una vez el dígito repetido. De esta forma el fucsia puede simplificarse porque todos los componentes tienen el dígito repetido.

```
p {  
    color: #f0f;  
}
```

RGB y RGBA

Como hemos visto podemos expresar los colores con sus componentes RGB (Red, Green, Blue). Si no nos queremos calentar la cabeza con valores hexadecimales, podemos también expresar el color usando el valor decimal de cada componente RGB, que tendrá un valor entre 0 y 255. Es el mismo rango que en hexadecimal, con 0 = `00` y 255 = `ff`.

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

Existe además la posibilidad de indicar un nivel de opacidad al color con el formato RGBA que añade transparencia (también llamada "canal alfa"). Este último

componente tiene valores decimales entre 0 (totalmente transparente) y 1 (totalmente opaco).

```
p {  
  color: rgba(255, 0, 255, 0.7);  
}
```

Notas: desde hace tiempo se puede usar `rgb()` o `rgba()` indistintamente para especificar un color con RGB, tenga o no canal alfa. También pueden usarse porcentajes para el valor de alfa, entre 0% y 100%. Las comas son opcionales: podemos usar `rgb(255 0 255)`.

Puedes comparar las diversas formas en [este Codepen](#).

HSL

Igual que el RGB nos permite expresar colores a partir de sus componentes de color rojo/verde/azul, existe otro sistema, HSL, que nos permite expresarlos a través de tres componentes diferentes:

- H: *hue* o matiz, expresado como valor numérico entre 0 y 360. Representa los grados en un círculo de una rueda de color, con el rojo en el 0, el verde en 120 y el azul en 240. Si recuerdas las matemáticas de la secundaria, también pueden usarse radianes con el sufijo `rad`.
- S: *saturation* o saturación, entre 0% y 100%.
- L: *lightness* o luminosidad, también entre 0% y 100%.

Curiosamente, usando estos tres valores tenemos acceso a la misma gama de colores que con los componentes RGB. También podemos añadir un valor A *alpha channel* o canal alfa, entre 0 y 1 (o entre 0% y 100%).

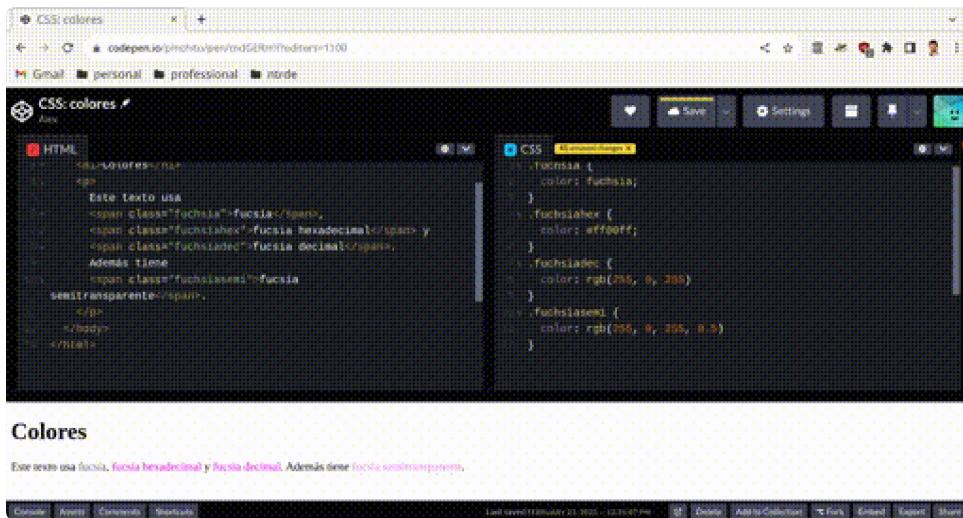
```
p {  
  color: hsl(300, 100%, 50%);  
}
```

Nota: también podemos usar la función `hsla()`, aunque oficialmente está desaconsejada o en desuso(*deprecated*).

Hay otros modos de color menos usados: HWB, lab, etc. No vamos a verlos en detalle pero está bien saber que existen.

Color picker

Para seleccionar un color en Chrome, o para jugar con la gama de colores, puedes inspeccionar un elemento con el botón derecho, y luego pinchar en cualquier color. No es el mejor selector de color del mundo, pero a menudo es el que tenemos más cerca.



Cómo abrir color picker en Chrome.

Puedes usar también el [Google color picker](#).

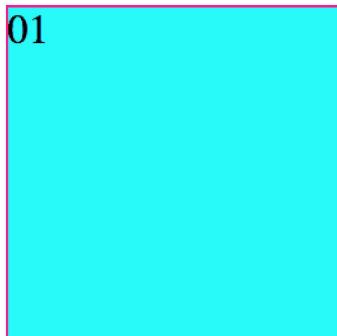
Fondos

Como veremos más adelante, cada elemento se puede ver como una caja. ¿Cómo le podemos añadir un fondo a esta "caja"?

Gracias a la propiedad `background` podemos llenar el fondo de nuestro contenedor con una imagen, con un color, o ambos:

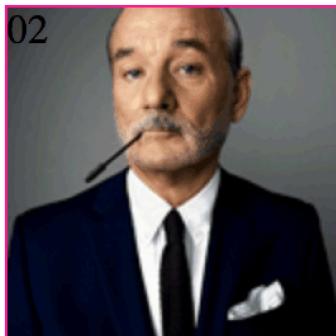
Fondo de color

01



Fondo de imagen

02



Fondo mixto

03



Ejemplos de background

La propiedad `background` puede contener estos valores:

- Url de una imagen
- Posición de la imagen dentro del contenedor (horizontal y vertical)
- Modo de repetición de la imagen
- Color de fondo

```
.box {  
  background: url('url-de-la-imagen') left top no-repeat #ffcc00;  
}
```

Nota: el orden no tiene que ser necesariamente el de arriba, pero os proponemos usarlo para tener una estructura uniforme.

Propiedades sueltas o combinadas

Realmente, la propiedad `background` es una versión acortada de estas propiedades, que se pueden usar por separado:

- `background-color` : fija el color del fondo, con los mismos valores que la propiedad `color` que hemos visto en detalle.

- `background-image` : carga una imagen para usar como fondo del elemento. Normalmente se usa con una URL:
`url('https://servidor.com/ruta/imagen.jpg')`.
- `background-position` : especifica la posición de la imagen en el fondo, relativa al origen de la capa. Valores posibles: `top`, `bottom`, `left`, `right`, `center`. También se pueden combinar los dos valores, vertical y horizontal: `top right`.
- `background-repeat` : especifica cómo se repite la imagen si hay espacio de sobra. Valores posibles: `repeat` repite la imagen, `no-repeat` no la repite, `space` deja espacio entre repeticiones.
- `background-size` : fija el tamaño de la imagen de fondo respecto al elemento contenedor. Valores posibles: `contain` fuerza a que la imagen esté dentro del elemento, `cover` estira la imagen hasta que esté completamente incluida en el elemento.
- `background-attachment` : especifica cómo se comporta la imagen de fondo cuando el texto no cabe en la ventana y hay que hacer *scroll*. Valores posibles: `scroll` hace que el fondo no se mueva, `fixed` hace que se quede clavado en la esquina, y `local` hace que el fondo se mueva con el resto del contenido.
- `background-clip` : especifica si el fondo se extiende por debajo del borde (`border-box`), del relleno (*padding*) (`padding-box`) o solo del contenido (`content-box`). El modo `text` nos permite también fijar el fondo únicamente en el texto.
- `background-origin` : fija el origen de la capa que puede tener los siguientes valores:
 - `border-box`: El fondo se coloca en relación con el borde del cuadro.
 - `padding-box`: El fondo está colocado en relación con el relleno del cuadro.
 - `content-box`: El fondo se coloca en relación con el contenido del cuadro.

En [este Codepen](#) puedes ver varios ejemplos de propiedades sueltas y combinadas.

Usar la propiedad combinada `background` o las propiedades sueltas puede producir el mismo resultado, pero en general no son lo mismo:

```
.box {  
    background-color: green;  
}  
.box {  
    background: green;  
}
```

Mientras que en el primer caso estamos diciendo solamente que el color de fondo sea `green`, en el segundo estamos diciendo eso y además que el resto de valores (`background-image`, `background-position` y demás) pasen a su valor por defecto.

¿Sabrías adelantar el resultado de aplicar esta clase?:

```
.box {  
    background-image: url("https://www.fillmurray.com/150/1500");  
    background: red;  
}
```

- Se verá a Bill Murray de fondo.
- Se verá a Bill Murray de fondo pero lo que no rellene la imagen será de color rojo.
- Solamente se verá un color rojo de fondo.
- Otra.

Cuando usamos un atajo estamos definiendo TODAS las opciones, aunque no las usemos; por eso siempre deberíamos especificar las propiedades que necesitemos, y solo usar los atajos cuando realmente estemos usando la mayoría de las propiedades :)

Si únicamente queremos cambiar el color de fondo deberíamos usar `background-color`. Si por el contrario queremos poner una imagen, en una posición y con una repetición, deberíamos usar el atajo `background`, ya que realmente se escribe menos:

```
.box {  
background-image: url("https://www.fillmurray.com/150/150");  
background-position: left top;  
background-repeat: no-repeat;  
}
```

```
.box {  
background: url("https://www.fillmurray.com/150/150") left top no-repea  
}
```

Composición de imágenes

Hay una propiedad que solo se puede usar por separado:

- `background-blend-mode` : modo de mezclar imágenes. Es posible cargar varias imágenes con un modo de mezcla diferente cada una.

Ya hemos visto en [este Codepen](#) cómo combinar imágenes. Es un concepto avanzado que no se usa mucho, pero está bien saber que existe.

Background-size

Desde hace un tiempo hay una propiedad nueva que nos permite redimensionar la imagen de fondo y hasta definir cómo se debe ajustar a nuestro contenedor:

`background-size`.

Nosotras vamos a ver cómo ajustar el fondo a nuestro contenedor, pero puedes consultar la [documentación completa](#).

El tamaño se puede especificar en píxeles (`px`), tanto por ciento `50%`, etc.

Podemos usar dos tamaños, horizontal y vertical: `100px 50%` ajustaría el tamaño de la imagen a 100 píxeles en horizontal y un 50% en vertical del tamaño del contenedor.

También hay dos valores especialmente interesantes ya que permiten definir cómo se ajustará nuestra imagen de fondo al contenedor: `contain` y `cover`.

`contain`

Aumenta o reduce la imagen proporcionalmente todo lo que pueda sin deformarla para que quepa en nuestro contenedor.

`cover`

Aumenta o reduce la imagen proporcionalmente para asegurarse que siempre cubre todo el área de nuestro contenedor, aunque eso signifique que parte de la imagen pueda quedar oculta.

[Ejemplos de uso de `contain` y `cover`](#)

Combinación de tamaño y posición

Ten en cuenta que para especificar `background-size` dentro de la propiedad global hay que usarlo combinado con un `background-position` como `position/size`, como en el [Codepen de arriba](#):

```
.second {  
  background: url('https://upload.wikimedia.org/wikipedia/commons/0/0a/Bi  
}
```

Recursos Externos

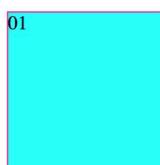
[Guía de colores de MDN](#).

[Guía de fondos de MDN](#).

Ejercicios

1. ¿Sabrías replicar los ejemplos de fondo usando [este Codepen](#)?

Fondo de color



Fondo de imagen



Fondo mixto



1.1.8 Hojas de Normalización y Reset

Todos los elementos HTML tienen una apariencia que comparte cada navegador, con pequeñas variaciones. Por defecto, el tamaño de texto es de 16px, con un interlineado de 1,15. Los encabezados y párrafos tienen un margen superior e inferior relacionado con el tamaño de texto: el `<h1>` se muestra con un tamaño de 32px y tiene 22px de margen. El fondo de la página es blanco y el color del texto es negro, etc.

Nota: la medida básica en web es el píxel o px; cada dispositivo tiene su pantalla con unas dimensiones definidas en píxeles. Por ejemplo, la pantalla de móvil más pequeña tiene 320×480px (si no se indica lo contrario siempre es "alto por ancho").

Hojas de reset

Debido a las pequeñas variaciones de la apariencia por defecto de los diferentes elementos HTML en cada navegador existen unas hojas de estilos más o menos estándar y más o menos completas que se llaman [hojas de reset o normalización de CSS](#).

Se trata de una hoja de estilos que intenta que todos los elementos se muestren igual en todos los navegadores. No hay un estándar para esta normalización.

Ahora mismo parte de la comunidad de desarrollo no considera que estas hojas de reset sean necesarias porque:

- Las páginas deben verse bien en todos los navegadores, pero no necesariamente igual.
- Las últimas versiones de los navegadores son bastante decentes y la época dura de los navegadores antiguos, ya pasó.
- No hay un método estándar de reseteo de CSS.

Nota: aún así, en algunos casos puede interesar usar una o incluso hacerse una propia, así que conocerlas es importante.

Spotify

Ejercicio para el pair programming

Hemos creado este ejercicio para que lo hagáis durante la hora de pair programming entre tu compañera y tú. Este ejercicio es incremental, es decir, cada día vamos a ir añadiendo las nuevas funcionalidades que hemos aprendido.

En estos materiales encontraréis el enunciado de las tareas que debéis hacer cada día. Es obligatorio que lo hagáis en la hora de pair programming porque:

- Aquí os enseñamos trucos y buenas prácticas.
- Cuando a mitad de módulo cambiéis de pareja tendréis que seguir trabajando sobre vuestro ejercicio o el de vuestra nueva pareja.

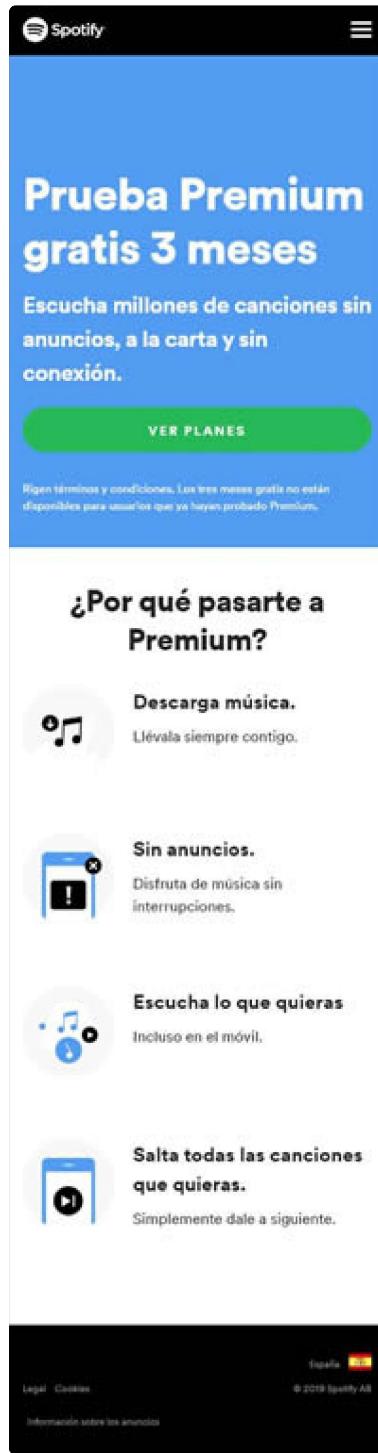
Enunciado

El objetivo de este ejercicio es aprender a estructurar y maquetar una página web desde cero. Por ello nos vamos a preocupar de que la distribución de elementos de la página esté perfecta.

Qué vamos a maquetar

Dados los siguientes diseños, maquetaremos la web aplicando las *media queries* necesarias. Vas a comenzar por el diseño en formato móvil primero.

Aspecto de la web en una pantalla de 480px



Pantalla completa

Aspecto de la web en una pantalla entre 480px y 1000px



Premium Ayuda Descargar | Registrarse Iniciar sesión

Prueba Premium gratis 3 meses

Escucha millones de canciones sin anuncios, a la carta y sin conexión.

[VER PLANES](#)

Rigen términos y condiciones. Los tres meses gratis no están disponibles para usuarios que ya hayan probado Premium.



¿Por qué pasarte a Premium?



Descarga música.

Llévala siempre contigo.



Sin anuncios.

Disfruta de música sin interrupciones.



Escucha lo que quieras

Incluso en el móvil.



Salta todas las canciones que quieras.

Simplemente dale a siguiente.

[Legal](#) [Cookies](#) [Información sobre los anuncios](#)

España

© 2019 Spotify AB

Pantalla completa

Aspecto de la web a pantalla completa (por encima de 1000px)



Premium Ayuda Descargar | Registrarse Iniciar sesión

Prueba Premium gratis 3 meses

Escucha millones de canciones sin anuncios, a la carta y sin conexión.

VER PLANES

Rigen términos y condiciones. Los tres meses gratis no están disponibles para usuarios que ya hayan probado Premium.



¿Por qué pasarte a Premium?



Descarga música.

Llévala siempre contigo.



Sin anuncios.

Disfruta de música sin interrupciones.



Escucha lo que quieras

Incluso en el móvil.



Salta todas las canciones que quieras.

Simplemente dale a siguiente.

[Legal](#) [Cookies](#) [Información sobre los anuncios](#)

España © 2019 Spotify AB

Pantalla completa

Para entender lo que vamos a hacer durante los próximos días debes empezar por estos pasos:

Ejercicios

1. Revisa la Guía de introducción a Git

Una de las herramientas fundamentales que utilizaremos durante el curso es Git para llevar el control de las versiones de nuestro código y para compartir el código con nuestras compañeras.

Por ello el primer paso que vamos hacer para optimizar el trabajo en pareja es preparar nuestro entorno de programación con Git, con lo cual te recomendamos que leas [la Guía de introducción a Git del módulo 0](#). Es fundamental que en este punto sepas:

1. Crear un proyecto en Github.
2. Clonar un repositorio.
3. Crear un token en Github.
4. Subir y bajar cambios de un repositorio.

2 Crea el proyecto en Github

Ya hemos aprendido los primeros pasos de Git, ahora vamos a crear el repositorio para nuestro proyecto.

1. Crea un repositorio nuevo en GitHub llamado `promo-X-module-1-pair-Y-spotify`. Este repositorio lo crearás desde tu cuenta de GitHub, e incluirá el código que desarrolles durante el módulo 1. **Sustituye X por el número de tu promoción y Y por el número de tu pareja.** Por ejemplo, si eres de la promoción U y tu pareja es la 2, el nombre de tu repositorio será `promo-1-module-1-pair-2-spotify`. Incluye en este repositorio a tu compañera de pair como colaboradora.

(Si aún no lo tienes claro, revisa la Guía Básica de Git, o espera a la clase de Git; este punto no es prioritario.)

2. [En este enlace se encuentra el diseño para las diferentes versiones de dispositivos.](#)
3. Crea la estructura de carpetas adecuada para tu proyecto con HTML y CSS.