HTML y CSS

Para hacer páginas Web necesitas aprender dos lenguajes: **HTML y CSS.** Estos son los lenguajes que los navegadores (Firefox, Chrome, Internet Explorer) entienden e interpretan para mostrarte lo que ves cuando abres alguna página.

HTML se usa para definir **la estructura** de la página (títulos, párrafos, tablas, listas, vínculos, formularios, imágenes, etc.) y **CSS** se usa para **aplicarle el estilo** (color de fondo, márgenes entre los elementos, tamaño de las fuentes, bordes, etc.).

Etiquetas

HTML está compuesto de etiquetas. Una etiqueta es una palabra clave encerrada entre < y > . Por ejemplo, <hr> es una etiqueta que pinta una línea horizontal como la siguiente :

Algunas etiquetas necesitan una etiqueta de cierre porque pueden contener texto y otras etiquetas. La etiqueta de cierre lleva un slash (/) antes de la palabra clave. Por ejemplo, la etiqueta se utiliza para definir un párrafo y necesita una etiqueta de cierre cuando termina el párrafo:

```
<p>
  Este es un párrafo <em>importante</em>.
</p>
```

En este ejemplo hemos definido un párrafo que contiene texto y una etiqueta con más texto.

La estructura de una página Web

Una página Web (documento HTML) tiene dos grandes secciones: el encabezado (<head>) y el cuerpo (<body>) como se muestra en el siguiente ejemplo:

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

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

La primera línea <!DOCTYPE html> le dice al navegador que estamos usando la última versión de HTML,HTML5. Un documento HTML siempre empieza con la etiqueta <html>.

En el <head> va información que no es visible en la pantalla como el título del documento, referencia a otros archivos, etc. En el <body> van los elementos visibles en la pantalla.

Generalmente, el código HTML (Hyper Text Markup Language) viaja a través de HTTP (Hyper Text Transfer Protocol). Pero es posible crear un archivo con extensión .html y abrirlo en un navegador directamente.

La estructura con las nuevas etiquetas que soporta html5 serían las siguientes.

HEAD: Es para indicar la cabeza de tu documento. No debes confundir esto con header que se define más adelante.

- META. Con esto defines el tipo de codificación del documento por ejemplo utf-8
- TITLE: Sirve para especificar el título del documento. Este aparece en la barra del navegador. Esta es muy importante incluirla a los documentos creados, ya que la asociamos a una identidad por medio de su título.

HEADER: En esta sección va la cabecera de tu sitio web. El nombre, slogan y logo deben ir aquí.

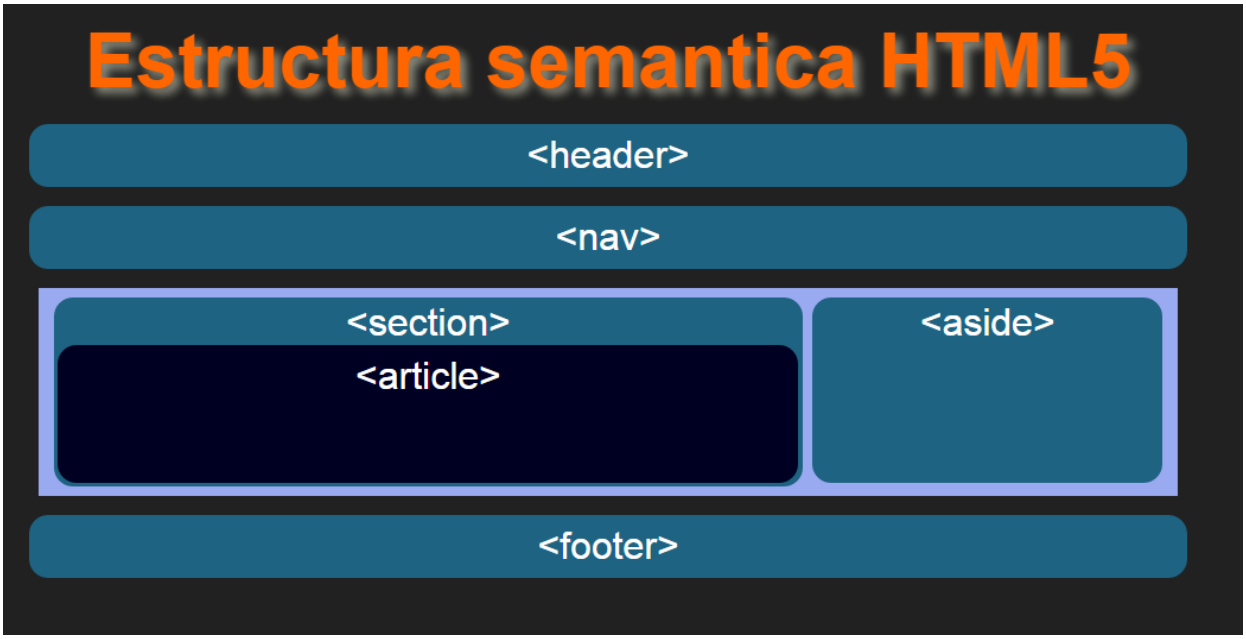
NAV: Es la principal barra de navegación o menú de navegación de tu sitio web. La posición y el estilo lo defines mediante CSS. Regularmente dentro de NAV estarán <ul> y <li>, que te sirven para dar forma y estilo a tu menú.

SECTION: Es la sección dónde va el contenido de tu sitio. Este a su vez puede agrupar articles

ARTICLE: Como su nombre lo indica, contiene un artículo. A su vez cada ARTICLE puede contener etiquetas <h2> para definir el título del artículo. Se recomienda que haya solo un <H1> por cada página que representa el título de tu página. A google no le gustan muchos <h1>, esto para tomarlo en cuenta en tu estrategia seo.

ASIDE: Contiene una barra lateral, como para poner enlaces a facebook, twitter y demás. Así divides el cuerpo de tu página en dos columnas. Una donde va el contenido, artículos, etc y otro donde van banners, ligas, etc.

FOOTER: Como su nombre lo indica este contiene el pie de página de tu sitio



Elementos

Un elemento HTML consiste generalmente de una **etiqueta de inicio** (p.e. `<p>`) y una **etiqueta de cierre** (p.e. `</p>`).
Las **etiquetas** se pueden anidar. Por ejemplo, para resaltar un texto en **negrilla** dentro de un párrafo se utiliza el **elemento** `strong`:

```
<p>Esto es un párrafo <strong>con un texto en negrilla</strong></p>
```

Párrafos

Los párrafos se definen con la etiqueta `<p>`:

```
<p>Esto es un párrafo</p>
```

```
<p>Esto es otro párrafo</p>
```

Títulos (encabezados)

Los títulos se definen con las etiquetas `<h1>` a `<h6>` siendo `<h1>` el título de más importancia y `<h6>` el de menos importancia:

```
<h1>Este es un título 1</h1>
<h2>Este es un título 2</h2>
<h3>Este es un título 3</h3>
```

Vínculos (links)

Los vínculos se definen con la etiqueta `<a>`:

```
<a href="http://makeitreal.camp/">Make it Real</a>
```

El destino del vínculo se define en el **atributo** `href`. Los atributos se utilizan para proveer información adicional a la etiqueta.

Imágenes

Las imágenes se definen con la etiqueta `<img>`:

```
<img src="imagen.jpg" alt="Una imagen">
```

La etiqueta `<img>` **no** necesita una etiqueta de cierre. El atributo `src` contiene la ruta al archivo con la imagen y el atributo `alt` contiene el texto que describe la imagen (útil para los buscadores y personas que utilizan lectores de pantallas).

Comentarios

Puedes agregar comentarios al código HTML utilizando la siguiente sintaxis:

```
<!-- Escribe acá tu comentario -->
```

Los comentarios no son visibles en el documento HTML.

Más elementos de HTML

En esta sección vamos a ver otros elementos de HTML.

Importancia y énfasis al texto

La etiqueta `<strong>` se utiliza para darle importancia al texto y la etiqueta `<em>` se utiliza hacer énfasis en el texto.
Generalmente `<strong>` se muestra en **negrilla** y `<em>` en *itálica*, pero este comportamiento se puede modificar a través de CSS.

```
<p>Esto es <strong>importante</strong> y quiero <em>hacer énfasis</em> en esto.</p>
```

Nota: También existen las etiquetas `<b>` e `<i>` que muestran el texto en **negrilla** e *itálica*respectivamente. Sin embargo, no se considera buena práctica usarlas porque es mejor aplicar el formato por CSS.

Saltos de línea y línea horizontal

La etiqueta `<br>` se utiliza para forzar un salto de línea (lo equivalente a oprimir la tecla `Enter` cuando estás escribiendo un documento)

```
<p>Hola<br>Amigos</p>
```

La etiqueta `<br>` no necesita una **etiqueta de cierre**.
La etiqueta `<hr>` muestra una línea horizonal en el documento.

Listas

Puedes crear listas ordenadas y no ordenadas.

Una **lista ordenada** está enumerada mientras que una **lista no ordenada** utiliza viñetas.

Un ejemplo de **lista ordenada** es la siguiente:

1. Item 1
2. Item 2
3. Item 3

Un ejemplo de **lista no ordenada** es la siguiente:

- Item 1
- Item 2
- Item 3

Una **lista ordenada** se crea con la etiqueta `<ol>` y los items con la etiqueta `<li>`:

```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ol>
```

Una **lista no ordenada** se crea con la etiqueta `<ul>` y los items con la etiqueta `<li>`:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

Nota: Por ningún motivo utilices la etiqueta `<li>` por fuera de un `<ul>` o `<ol>`.

Etiquetas invisibles

Existen dos etiquetas que no son visibles en el documento pero se utilizan para agrupar otros elementos y aplicarles estilos: `<div>` y `<span>`.
Veamos un ejemplo de `<div>` en donde estamos agrupando un título `<h1>`, una imagen `<img>` y un párrafo `<p>`:

```
<div class="main-section">
  <h1>Título</h1>
  <img src="imagen.jpg" alt="Imagen">
  <p></p>
</div>
```

`<span>` se utiliza para aplicarle estilos a un texto (generalmente dentro de un parrafo):

```
<p>Hola, tu saldo es <span class="saldo">$12000</span></p>
```

Elementos de bloque y en línea

Existen elementos que, independiente de su contenido, ocupan todo el ancho de la página como `p`, `ol`, `ul`, `li` y `div`. A estos elementos se les conoce como **elementos de bloque**.
Otros elementos como `span`, `strong` y `em` ocupan solo el espacio que ocupa su contenido interno. A estos elementos se les conoce como **elementos en línea**.

CSS

CSS **(Cascading Style Sheets)** , es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias a las CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como incluir márgenes, tipos de letra, fondos, colores. Por ejemplo, la siguiente regla cambia el color de la fuente de los párrafos a azul:

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

Introducción a CSS

CSS (Cascading Style Sheets) es un lenguaje que se utiliza para definir **el formato (los estilos)** de los elementos HTML.

Existen 3 formas para definir los estilos CSS:

- Inline (en línea): Utilizando el atributo `style` de los elementos HTML.
- Interno: Utilizando la etiqueta `<style>` dentro de `<head>`.
- Externo: Utilizando un archivo externo con extensión `.css`.

La forma más recomendada es utilizar un archivo externo.

Inline CSS

Se utiliza para aplicarle estilos a un único elemento HTML:

```
<h1 style="color: blue">Este es un título de color azul</h1>
```

Nota: esto no se considera una buena práctica porque estás mezclando la estructura del documento con el formato.

CSS Interno

Se utiliza para definir los estilos de una única página Web. El CSS se define en el `<head>` dentro de la etiqueta `<style>`:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
       h1 { color: blue; }
    </style>
  </head>
  <body>
    <h1>Esta es un título de color azul</h1>
  </body>
</html>
```

Para definir el CSS se utilizan **reglas CSS**:

```
h1 {
  color: blue;
  font-size: 24px;
}
```

Una **regla CSS** está compuesta de un **selector** (que define a qué elementos se le quieren aplicar esos estilos), y una o más **propiedades** (que definen los estilos que se le van a aplicar a los elementos seleccionados).

Una propiedad CSS está compuesta de un atributo y un valor. Al final se agrega un punto y coma (`;`).

CSS Externo

Es un archivo con extensión `.css` que se utiliza para definir los estilos de múltiples páginas Web.
Para referenciar el archivo CSS desde HTML se utiliza la etiqueta `<link>` en el `<head>` del HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
```

```
  <body>
    <h1>Esta es un título de color azul</h1>
  </body>
</html>
```

Reglas CSS

Cuando utilizas CSS interno o externo los estilos se definen utilizando **reglas CSS**.

Una **regla CSS** está compuesta de un **selector** (que define a qué elementos se le quieren aplicar esos estilos), y una o más **propiedades** (que definen los estilos que se le van a aplicar a los elementos seleccionados).

Por ejemplo:

```
h1 {
  color: blue;
  font-size: 24px;
}
```

`h1` es el **selector**. En este caso estamos "seleccionando" todas los elementos `h1` del documento.
Las propiedades se encierran entre llaves `{` y `}`.
Cada propiedad está compuesta de un atributo, dos puntos y un valor. Al final de cada propiedad va un punto y coma (`;`).

Selectores

La forma más fácil de seleccionar elementos es por etiqueta. Sin embargo ¿qué pasa si solo queremos seleccionar ciertos párrafos del documento? Para eso podemos utilizar clases y ids.

Clases

A las etiquetas HTML les puedes agregar un atributo especial llamado `class` que después puedes usar como selector:

```
<p>Párrafo 1</p>
<p class="special">Parrafo 2</p>
```

Si queremos que los elementos que tengan la clase `special` sean de color azul, podemos hacer lo siguiente:

```
.special {
  color: blue;
}
```

Para seleccionar elementos por clase se utiliza un punto seguido del nombre de la clase. Esto seleccionaría todos los elementos que tengan esa clase.

El atributo `class` en el HTML soporta más de una clase separándolas por espacio. Por ejemplo:

```
<p class="clase1 clase2 clase 3">Parrafo 2</p>
```

Ids

Otra forma de seleccionar elementos es por el atributo `id`. La diferencia entre clases y id's es que los ids no se deben repetir en varios elementos, las clases si se pueden repetir.

```
<p>Párrafo 1</p>
<p id="algun-id">Parrafo 2</p>
```

Ahora, si queremos que el elemento que tenga el id `algun-id` sea de color azul, podemos hacer lo siguiente:

```
#algun-id {
  color: blue;
}
```

Nota: En general (y en lo posible) se recomienda utilizar clases en vez de ids.

Propiedades CSS

Dentro de una regla de CSS se pueden definir una o más propiedades CSS. Las propiedades se componen de un atributo, seguido de dos puntos (`:`), un espacio y el valor:

```
color: blue;
```

Veamos algunos de los atributos que más se utilizan:

color

El atributo **color** define el color del texto. Las dos formas más comunes de definir colores en CSS son:

- El **nombre de un color** (`red`, `blue`, `white`, etc.). Ver lista de nombres
- El **valor hexadecimal** (p.e. negro es `#000000` y blanco `#ffffff`).

Utiliza esta herramienta para ver el **valor hexadecimal** de cualquier color.

font-size

El atributo **font-size** define el tamaño del texto. Las formas más comunes de definir el tamaño son:

- **Pixeles (px):** Un pixel es igual a un punto en la pantalla de tu computador. Ejemplos: `12px`, `24px`, `36px`, etc.
- **Ems (em):** Es una forma de definir el tamaño de forma relativa al elemento padre. Esta forma está tomando cada vez más fuerza para crear páginas que se adaptan a la pantalla. Ejemplos: `1em`, `0.8em`, `1.2em`.

font-family

El atributo **font-family** define la fuente que se quiere usar. Se pueden definir varias fuentes separadas por coma. Si la primera fuente no está instalada, se intenta con la segunda y así sucesivamente.

En este enlace puedes encontrar una lista de fuentes más comunes.

font-weight

El atributo **font-weight** define el peso de una fuente (su grosor). Puede ser:

- `normal`: este es el valor por defecto, muestra el texto normal.
- `bold`: muestra el texto en negrilla.

background-color

El atributo **background-color** define el color de fondo del elemento. Se utilizan las mismas formas de definir el color que describimos en el atributo **color** previamente.

border

El atributo **border** dibuja un borde alrededor del elemento, recibe tres argumentos:

- El **ancho** en `px` (pixeles).
- El **estilo de borde** (sólido, punteado, doble, etc.)
- El **color** del borde.

Ejemplos:

```
p { border: 1px solid grey; }
h1 { border: 5px dotted blue; }
h2 { border: 10px dashed #000; }
```

Tablas

Una tabla se define con la etiqueta `<table>`.
Las filas se definen con la etiqueta `<tr>`.
Las celdas se definen con la etiqueta `<td>`.
Ejemplo:

```
<table>
  <tr>
    <td>Pedro</td>
    <td>Perez</td>
  <tr>
  <tr>
```

```
    <td>Juan</td>
    <td>Gomez</td>
  <tr>
</table>
```

Nunca utilices `<tr>` por fuera de un `<table>`.
Nunca utilices `<td>` por fuera de un `<tr>`.

Agregando bordes

Por defecto las tablas no tienen bordes. Si deseas agregarles un borde puedes utilizar la propiedad `border` de CSS:

```
table, th, td {
  border: 1px solid black;
}
```

Uniendo los bordes

Si quieres que los bordes se unan en uno puedes utilizar la propiedad `border-collapse` de CSS:

```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```

Encabezados

Para crear encabezados utiliza `th` en vez de `td`:

```
<table>
  <tr>
    <th>Nombre</th>
    <th>Apellido</th>
  <tr>
  <tr>
    <td>Juan</td>
    <td>Gomez</td>
  <tr>
</table>
```

Agregar espacio interno en las celdas

Utiliza la propiedad `padding` de CSS para agregar espacio interno en las celdas:

```
th, td {
  padding: 15px;
}
```

Celdas que ocupan más de una columna o fila

Para que una celda (`td` o `th`) ocupe más de una columna utiliza el atributo `colspan` con el número de columnas que quieres que ocupe. Por ejemplo:

```
<table>
  <tr>
    <td>1</td>
    <td>2</td>
    <td>3</td>
  </tr>
  <tr>
    <td colspan="2">Ocupa 2 columnas</td>
    <td>4</td>
  </tr>
  <tr>
    <td colspan="3">Ocupa 3 columnas</td>
  </tr>
</table>
```

Para que una celda ocupe más de una fila utiliza el atributo `rowspan` con el número de filas que quieres que ocupe:

```
<table>
  <tr>
    <td rowspan="2">Ocupa 2 filas</td>
    <td>1</td>
  </tr>
  <tr>
    <td>2</td>
  </tr>
</table>
```

Formularios

Los formularios nos permiten pedirle información a los usuarios.

Un formulario se crea con la etiqueta `<form>`:

```
<form>

</form>
```

Un formulario puede tener muchos elementos de entrada como campos de texto, casillas de verificación (checkbox), listas desplegables, botones de envío y muchos más. Veamos algunos de ellos:

Campo de texto

Para definir un campo de texto utiliza la etiqueta `<input type="text">`.

```
<form>
```

```
  <input type="text">
</form>
```

Campo de contraseña

Los campos de contraseña son similares a los campos de texto pero el texto se reemplaza automáticamente por asteriscos (*) cuando la persona escribe.
Para definir un campo de contraseña utiliza la etiqueta `<input type="password">`.

```
<form>
  <input type="password">
</form>
```

Casilla de verificación (checkbox)

Para definir una casilla de verificación utiliza la etiqueta `<input type="checkbox">`:

```
<form>
  <input type="checkbox"> Acepto recibir información
</form>
```

Botón de envío

El botón de envío, como su nombre lo indica, se utiliza para enviar la información que ha ingresado el usuario a un servidor remoto.

Para definir un botón de envío utiliza la etiqueta `<input type="submit" value="Enviar">`:

```
<form>
  <input type="submit" value="Enviar">
</form>
```

El atributo `value` define el texto del botón.
Para utilizar un botón de envío necesitas también definir al menos:

1. La URL del servidor al que quieres enviar la información en el atributo `action` .
2. Agregar un atributo `name` a todos los campos de entrada que quieres enviar al servidor.

Por ejemplo:

```
<form action="http://mi-empresa.com/contacto">
  <div>
    Nombre: <input type="text" name="nombre">
  </div>
  <div>
    Mensaje: <input type="text" name="mensaje">
  </div>
  <input type="submit" value="Enviar">
</form>
```

Labels

Es buena práctica utilizar una etiqueta `<label>` para encerrar el texto que se va a mostrar para cada campo de entrada.
Para relacionar el `<label>` al campo de entrada debes:

- Agregar un atributo `id` al campo de entrada.
- Agregar un atributo `for` al `<label>` con el mismo valor del atributo `id` del campo de entrada.

Por ejemplo:

```
<form>
  <label for="nombre">Nombre:</label> <input type="text" id="nombre"
name="nombre">
</form>
```

La ventaja de utilizar un label es que si hacen click sobre el label se va a enfocar automáticamente el campo de entrada. Esto es especialmente útil en casillas de verificación y botones de radio.

Botones de radio

Para definir un botón de radio utiliza la etiqueta `<input type="radio">`:

```
<form>
  <input type="radio">
</form>
```

Los botones de radio son especialmente útiles para que el usuario escoja una única opción. Por ejemplo:

```
<form action="/accion">
  <div>
    <label for="masculino">Masculino</label>
    <input type="radio" name="genero" id="masculino" value="masculino">
  </div>
  <div>
    <label for="femenino">Femenino</label>
    <input type="radio" name="genero" id="femenino" value="femenino">
  </div>
  <div>
    <label for="otro">Otro</label>
    <input type="radio" name="genero" id="otro" value="otro">
  </div>
  <input type="submit" value="Enviar">
</form>
```

Fíjate en lo siguiente:

- El valor del atributo `name` es `genero` en todos los botones de radio.
- Cada botón de radio tiene un atributo `value` con un valor distinto.

Es decir, que al servidor va a llegar el `genero` con el valor del botón de radio que haya seleccionado el usuario (`masculino`, `femenino` u `otro`).

Áreas de texto

Las áreas de texto son similares a los campos de texto pero pueden ser de múltiples líneas.

Para crear un área de texto se utiliza la etiqueta `<textarea>`:

```
<form>
  <textarea rows="5"></textarea>
</form>
```

Nota: a diferencia de la etiqueta `<input>`, la etiqueta `<textarea>` necesita una etiqueta de cierre.

Lista de selección

Para crear una lista de selección utiliza la etiqueta `<select>`:

```
<form>
  <select name="genero">
    <option value="masculino">Masculino</option>
    <option value="femenino">Femenino</option>
  </select>
</form>
```

El modelo de caja en CSS

El modelo de caja se compone del margen externo (`margin`), borde (`border`) y margen interno (`padding`) de un elemento.

Márgenes

El margen se utiliza para definir el espacio alrededor del elemento.

Puedes utilizar las propiedades CSS `margin-top`, `margin-right`, `margin-bottom` y `margin-left` para definir el margen superior, derecho, inferior e izquierdo respectivamente:

```
p {
  margin-top: 5px;
  margin-right: 5px;
  margin-bottom: 5px;
  margin-left: 5px;
}
```

Atajo con `margin`

Para reducir el código puedes especificar todas las márgenes en una sola propiedad `margin`:

```
p {
  margin: 5px 5px 5px 5px;
}
```

El orden es: margen superior, derecho, inferior e izquierdo.

Si todos los márgenes son iguales puedes hacer lo siguiente:

```
p {
  margin: 5px;
}
```

`margin` también puede recibir dos valores: el valor para el margen superior e inferior, y el valor para el margen izquierdo y derecho.
Por ejemplo, la siguiente regla le aplicaría un margen de `10px` arriba y abajo, y `20px` a los lados.

```
p {
  margin: 10px 20px;
}
```

`margin` también puede recibir **tres** valores: el valor para el margen superior, y el margen de los lados, y el margen inferior.
Por ejemplo, la siguiente regla le aplicaría un margen de `10px` arriba, `20px` a los lados y `30px` abajo:

```
p {
  margin: 10px 20px 30px;
}
```

Bordes

Las propiedades CSS `border-width`, `border-style` y `border-color` nos permiten definir el ancho, el estilo y el color del borde de un elemento.
El siguiente ejemplo definiría un borde **sólido** de `1px` de color rojo:

```
p {
  border-width: 1px;
  border-style: solid;
  border-color: red;
}
```

Las opciones más comunes de `border-style` son `solid` (sólido), `dotted` (punteado), `dashed`(guiones) y `double` (doble).
Cada una de las propiedades puede recibir de uno a 4 valores, muy parecido a como funciona con los márgenes:

- Un valor: aplica a los 4 lados (p.e. `border-width: 5px`).
- Dos valores: el primero para arriba y abajo, el segundo a los lados. (p.e. `border-width: 5px 10px`)
- Tres valores: el primero para arriba, el segundo para los lados y el tercero para abajo (p.e. `border-width: 5px 10px 20px`).
- Cuatro valores: arriba, derecha, abajo, izquierda (p.e. `border-width: 1px 2px 3px 4px`).

Atajo con `border`

Puedes utilizar el atajo `border` para definir el ancho, estilo y color de todos los lados:

```
p {
  border: 1px solid blue;
}
```

También puedes utilizar el atajo pero para cada uno de los lados:

```
p {
  border-top: 1px solid blue;
  border-right: 2px dashed red;
  border-bottom: 3px dotted yellow;
  border-left: 4px double green;
}
```

Bordes redondeados

La propiedad `border-radius` se utiliza para agregar bordes redondeados a un elemento:

```
p {
  border: 2px solid red;
  border-radius: 5px;
}
```

Puedes asignar la propiedad `border-radius` a cada esquina individualmente con `border-top-left-radius`, `border-top-right-radius`, `border-bottom-left-radius` y `border-bottom-right-radius`.

Márgenes externas (padding)

El **padding** se utiliza para definir el margen interno de un elemento.

Al igual que con los márgenes externos, CSS tiene propiedades para definir el **padding** de cada lado del elemento: `padding-top`, `padding-right`, `padding-bottom` y `padding-left`.

```
p {
  padding-top: 5px;
  padding-right: 5px;
  padding-bottom: 5px;
  padding-left: 5px;
}
```

Atajo con `padding`

Para reducir el código puedes especificar todos los **paddings** en una sola propiedad `padding`:

```
p {
  padding: 5px 5px 5px 5px;
}
```

`padding` puede recibir uno, dos, tres y cuatro valores como `margin`:

- Un valor: aplica a los 4 lados.
- Dos valores: el primero aplica arriba y abajo, el segundo a los lados.
- Tres valores: el primero arriba, el segundo a los lados y el tercero abajo.
- Cuatro valores: arriba, derecha, abajo, izquierda.

Ancho y alto

Puedes especificar el **ancho** y el **alto** de un elemento utilizando las propiedades `width` y `height`:

```
p {
  width: 400px;
  height: 300px;
}
```

El **padding** y el **borde** suman al ancho y alto del elemento.

Fondos

Para definir el color, la imagen, la posición, etc. de fondo de un elemento utiliza las siguientes propiedades CSS:

- `background-color`: define el color de fondo.
- `background-image`: define la imagen de fondo.
- `background-position`: define la ubicación de la imagen de fondo.
- `background-repeat`: define si se repite la imagen de fondo horizontal y verticalmente.
- `background-attachment`: define si la imagen se mantiene fija o se desplaza con la página.

`background-color`

Esta propiedad define el color de fondo del elemento. Por ejemplo, para que todos los párrafos tengan un color azul utiliza la siguiente regla:

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

También puedes usar un valor hexadecimal (p.e. `#0000FF`) o RGB (p.e. `rgb(255, 0, 0)`).

`background-image`

Define la imagen de fondo. Por ejemplo:

```
p {
  background-image: url('mi-imagen.jpg')
}
```

La ruta de la imagen puede ser relativa al documento o un URL a una imagen en Internet.

`background-repeat`

Por defecto, cuando utilizas una imagen de fondo, la imagen se repite tanto horizontal como verticalmente (cuando no ocupa el ancho o alto del elemento).

Para cambiar este comportamiento utiliza la propiedad `background-repeat` que recibe las opciones:

- `no-repeat`: no repetir horizontal ni verticalmente.

- `repeat-x`: repetir solo horizontalmente.
- `repeat-y`: repetir solo verticalmente.

`background-position`

Nos permite controlar la posición y el tamaño de la imagen cuando `background-repeat` es `no-repeat`.
`background-position` recibe dos valores: la posición vertical y horizontal.
Los valores de `background-position` pueden ser una posición en palabras (p.e. top left), un porcentaje o una posición en pixeles u otra unidad.
Por ejemplo, la siguiente regla centra la imagen en el componente:

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

`background-size`

`background-size` nos permite cambiar el tamaño de la imagen de fondo.
`background-size` puede recibir dos valores: el ancho y el alto en porcentaje, pixeles o alguna otra unidad. Por ejemplo:

```
p {
  background-image: url('imagen.jpg');
  background-repeat: no-repeat;
  background-size: 60px 40px;
}
```

`background-size` también puede recibir dos valores especiales:

- `cover`: la imagen se estira hasta que ocupe todo el área del elemento.
- `contain`: la imagen se estira hasta que el alto o el ancho ocupen el alto o ancho del contenedor.

`background-attachment`

Por defecto, cuando utilizas una imagen de fondo, la imagen se desplaza con la página.

`background-attachment` nos permite cambiar ese comportamiento y dejar la imagen fija cuando la página se desplace creando un efecto interesante:

```
  background-image: url('imagen.jpg');
  background-repeat: no-repeat;
  background-attachment: fixed;
```

El atajo `background`

Puedes especificar varias propiedades de fondo en una línea con `background`.
Por ejemplo:

```
background: #00ff00 url("smiley.gif") no-repeat fixed center;
```

El orden es: `background-color`, `background-image`, `background-position`/`background-size`, `background-repeat` y `background-attachment`.
No importa si omites algunas de las propiedades. Por ejemplo, las siguientes declaraciones son válidas:

```
background: red;
background: url('mi-imagen');
background: red url('mi-imagen');
background: red url('mi-imagen') fixed center;
```

Posicionamiento

La propiedad CSS `position` define el tipo de posicionamiento para el elemento. Los posibles valores son:

- `static`: el valor por defecto.
- `relative`: relativo a su ubicación normal.
- `fixed`: fijo en la ventana del navegador.
- `absolute`: relativa el ancestro más cercano con posición `relative` o a la ventana del navegador.

Los valores `relative`, `fixed` y `absolute` se utilizan generalmente en conjunto con las propiedades `top`, `left`, `bottom` y `right`.

`position: relative`

Nos permite mover el elemento a una ubicación **relativa** a su posición original utilizando las propiedades: `left`, `top`, `right`, `bottom`.

```
p {
  position: relative:
  top: 20px;
  left: 20px;
}
```

En lo posible se recomienda mejor utilizar `margin`.
`position: relative` se utiliza más para posicionar elementos de forma absoluta dentro del elemento.

`position: absolute`

Nos permite ubicar un elemento a una ubicación **relativa** al ancestro más cercano que esté posicionado con `position: relative`, o a la ventana del navegador en su defecto.
Se utilizan las propiedades `top`, `left`, `right` y `bottom` para ubicar el elemento.

```
p {
  position: absolute:
  left: 100px;
  top: 100px;
}
```

`position: fixed`

Esta propiedad nos permite ubicar un elemento de manera **fija** relativamente a la ventana del navegador. No importa si se utilizan las barras de desplazamiento, el elemento va a permanecer fijo en su posición.

```
div.fixed {
  position: fixed;
  bottom: 0;
  right: 20px;
}
```

Selectores CSS

Ya hemos visto los selectores por etiqueta (p.e. `table`), clase (p.e. `.alert`) e id (p.e. `#alert`). Veamos otros selectores.

Lista de selectores

Puedes definir más de un selector en una regla CSS separando los selectores por coma (,). Por ejemplo:

```
h1, h2, h3 {
  margin-bottom: 21px;
}
```

Esta regla CSS le aplicaría el margen inferior a las etiquetas `h1`, `h2` y `h3`.

Más específicos

Podemos ser más específicos mezclando los selector. Por ejemplo, si queremos aplicar un estilos a todos los **párrafos** que tengan la clase **importante** utilizaríamos el selector `p.importante`:

```
p.importante {
  font-size: 21;
}
```

Descendientes

Puedes definir selectores que estén **dentro** de otros selectores.

Por ejemplo, la siguiente regla CSS aplica a las etiquetas `span` que estén dentro de una etiqueta `a`con clase `link`, que a su vez estén dentro de una etiqueta `div`.

```
div a.link span {
  color: red;
}
```

Descendientes directos

Puedes definir selectores que sean **hijos directos** de otros selectores.

Por ejemplo, la siguiente regla aplica a todos los elementos con clase `link` que sean **hijos directos** de una etiqueta `div`:

```
div > .link {
  color: red;
}
```

Cualquier selector

Utiilza el asterísco (`*`) para referirte a cualquier selector.
Por ejemplo, la siguiente regla aplicaría a **todos los elementos** de la página:

```
* {
  box-sizing: border-box;
}
```

Pseudo clases

Nos permiten seleccionar elementos por su estado actual.

Por ejemplo los vínculos que ya han sido visitados, los checkboxes que están seleccionados, los elementos sobre los que está pasando el mouse, etc.

Por ejemplo, los vínculos pueden estar en varios estados:

```
/* no visitado */
a:link {
    color: #FF0000;
}

/* visitado */
a:visited {
    color: #00FF00;
}

/* mouse sobre el vínculo */
a:hover {
    color: #FF00FF;
}

/* seleccionado */
a:active {
    color: #0000FF;
}
```

`:first-child` y `:last-child`

Estas dos pseudo clases nos permiten definir el primer y último selector dentro de cualquier elemento.

Por ejemplo, la siguiente regla aplicaría al primer párrafo dentro de cualquier elemento:

```
p:first-child {
  color: red;
}
```

Bordes, sombras y gradientes

En este capítulo vamos a explorar algunas de las nuevas características de CSS3 como bordes redondeados, sombras para cajas y textos, y gradientes.

Antes se utilizaban trucos para lograr el mismo efecto con imágenes pero ahora con CSS3 tenemos una forma más fácil y estándar de hacerlo.

Bordes redondeados

Veamos primero un ejemplo de bordes redondeados (consulta el HTML y CSS en la pestaña respectiva):

Para crear bordes redondeados utiliza la propiedad `border-radius`. `border-radius` recibe un tamaño (puede ser en pixeles, rems, etc.) como valor. Por ejemplo, para aplicar un borde redondeado con radio de `5px` a un `div` con clase `redondeado`utilizarías la siguiente regla:

```
div.redondeado {
  border-radius: 5px;
}
```

También puedes aplicarle bordes redondeados a cada esquina individualmente utilizando las siguientes propiedades:

- `border-top-left-radius`
- `border-top-right-radius`
- `border-bottom-left-radius`
- `border-bottom-left-radius`

Sin embargo, también existen atajos que puedes utilizar con `border-radius`. Por ejemplo, puedes definir el tamaño de cada esquina separando los valores por espacio:

```
div.redondeado {
  border-radius: 5px 10px 10px 20px;
}
```

El primer valor aplica a la esquina superior izquierda, el segundo a la superior derecha, el tercero a la inferior izquierda y el cuarto a la inferior derecha.

Cuando los valores de la esquina superior derecha e inferior izquierda son iguales podemos utilizar otro atajo:

```
div.redondeado {
  border-radius: 5px 10px 20px;
}
```

El primer valor aplica a la esquina superior izquierda, el segundo a la superior derecha e inferior izquierda, y el tercero a la inferior derecha.

Si la esquina superior izquierda y la inferior derecha tuvieran el mismo valor (p.e. 5px) podríamos utilizar otro atajo:

```
div.redondeado {
```

```
  border-radius: 5px 10px;
}
```
El primer valor aplica a la esquina superior izquierda y a la inferior derecha. El segundo valor a la superior derecha y a la inferior izquierda.

Sombras

Utiliza la propiedad `box-shadow` para agregarle sombra a una caja y `text-shadow` para agregarle sombra al texto.

Sombra de caja

Veamos primero un ejemplo (consulta el HTML y CSS en la pestaña respectiva):

Para agregarle sombra a una caja utiliza la propiedad `box-shadow`.
`box-shadow` recibe los siguientes valores separados por espacio:

- **Posición horizontal** (requerido) - un valor positivo ubica la sombra a la derecha de la caja mientras que uno negativo la ubica a la izquierda.
- **Posición vertical** (requerido) - un valor positivo ubica la sombra debajo de la caja mientras que uno negativo la ubica encima de la caja.
- **Difuminación** (opcional) - el radio de difuminación, entre mayor el número más borrosa va a ser la sombra.
- **Propagación** (opcional) - el radio de propagación, un valor positivo incrementa el tamaño de la sombra mientras que uno negativo reduce el tamaño de la sombra.
- **Color** (opcional) - por defecto utiliza el color de la fuente.

Sombra de texto

Veamos primero un ejemplo (consulta el HTML y CSS en la pestaña respectiva):

Para agregarle sombra al texto utiliza la propiedad `text-shadow`.
`text-shadow` recibe los siguentes valores separados por espacio:

- **Posición horizontal** (requerido) - un valor positivo ubica la sombra a la derecha del texto, uno negativo a la izquierda.
- **Posición vertical** (requerido) - un valor positivo ubica la sombra debajo del texto, uno negativo encima.
- **Difuminación** (opcional) - el radio de difuminación, entre mayor el número, más borrosa la sombra.
- **Color** (opcional) - por defecto utiliza el color de la fuente.

Gradientes

Los gradientes te permiten realizar transiciones entre dos o más colores.

Existen dos tipos de gradientes:

- Lineales
- Radiales

Gradientes lineales

Para crear un gradiente lineal utiliza la función `linear-gradient` en la propiedad `background` o `background-image`:

```
.gradiente {
  background: linear-gradient(red, orange);
}
```

Por defecto la dirección es de arriba hacia abajo:

Dirección

Para cambiar la **dirección** puedes pasarle un primer argumento a `linear-gradient`. Los posibles valores son:

- `to bottom` - hacia abajo (el valor por defecto).
- `to top` - hacia arriba.
- `to right` - hacia la derecha.
- `to left` - hacia la izquierda.
- `to top left` - hacia la esquina superior izquierda.
- `to top right` - hacia la esquina superior derecha.
- `to bottom left` - hacia la esquina inferior izquierda.
- `to bottom right` - hacia la esquina inferior derecha.
- Un ángulo (p.e. `90deg`, `-45deg`, etc.)

Múltiples colores

Puedes utilizar más de dos colores en tu gradiente:

```
.gradiente {
  background: linear-gradient(to right, blue, red, yellow, green);
}
```

Veamos este ejemplo en Codepen:

También puedes cambiar la ubicación donde cambia cada color:

```
.gradiente {
  background: linear-gradient(to right, blue 20%, red 60%, yellow, green);
}
```

Transparencias

Los gradientes soportan transparencias que se pueden utilizar para crear efectos interesantes. El siguiente gradiente inicia completamente transparente y termina en un rojo total:

```
.gradiente {
  background: linear-gradient(to right, rgba(255, 0, 0, 0), rgba(255, 0, 0, 1));
}
```

Gradientes radiales

Para crear un gradiente radial utiliza la función `radial-gradient` en la propiedad `background` o `background-image`:

```
.gradiente {
  background: radial-gradient(red, yellow);
}
```

Por defecto el gradiente inicia en el centro de la caja y tiene forma de elipse:

Si quieres cambiar el centro del gradiente puedes pasarle un primer argumento a `radial-gradient`:

```
.gradiente {
  background: radial-gradient(at 20px 20px, red, yellow);
}
```

Veamos este ejemplo en Codepen:

Además de posiciones exactas puedes utilizar `top`, `right`, `bottom`, `left`, `top right`, etc.
También puedes cambiar la forma del gradiente a un círculo de la siguiente forma:

```
.gradiente {
  background: radial-gradient(circle at 20px 20px, red, yellow);
}
```

Bootstrap 3

Bootstrap es el framework CSS más popular de la actualidad y se utiliza para crear páginas Web con un diseño consistente, que se adapta a diferentes tipos de pantallas (responsive design), utilizando componentes prediseñados que pueden ser fácilmente modificados.

Aunque la versión más reciente de Bootstrap es la 4, aún vale la pena aprender Bootstrap 3 por varias razones. Primero, la mayoría de sitios y aplicaciones Web aún utilizan Bootstrap 3. Segundo, Bootstrap 4 ya no soporta Internet Explorer 8 y 9 (que sí son soportados en Bootstrap 3), así que en algunas ocasiones no

tendrás opción. Tercero, aprender Bootstrap 4 va a ser mucho más fácil si ya conoces Bootstrap 3.

Empezando con Bootstrap

Aunque existen varias formas de empezar con Bootstrap, quizá la forma más fácil es utilizar la siguiente plantilla HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap Starter</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></scrip
t>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></sc
ript>
  </head>
  <body>

  </body>
</html>
```

Bootstrap se compone de un archivo **CSS** y un archivo de **JavaScript**. Adicionalmente necesitamos **jQuery**.

Estamos usando un **CDN (Content Delivery Network)** que se encarga de alojar y publicar estos archivos en Internet de forma muy eficiente.

¿Qué nos ofrece Bootstrap?

- Estilos más modernos y consistentes para los elementos básicos de HTML: títulos, párrafos, listas, imágenes, tablas, formularios, etc.
- Componentes como barra de navegación, carrusel de imágenes, pestañas, fuente de íconos, paginación, ventanas emergentes (modales), botones y barras de progreso entre otros.
- Una grilla para organizar los elementos de nuestra página fácilmente.
- Algunas clases de CSS útiles para trabajar con texto y responsive design.

La documentación

Puedes encontrar la documentación de Bootstrap 3
en https://getbootstrap.com/docs/3.3/.

La documentación está en Inglés pero es muy completa. Te recomendamos usarla como referencia.

Elementos básicos de HTML

Bootstrap 3 le aplica estilos a los elementos básicos de HTML como encabezados, párrafos, formularios, tablas, etc. Con sólo incluir **Bootstrap** en tu página ya vas a ver cambios en la tipografía y otros elementos.

Encabezados

Utiliza los encabezados `<h1>` a `<h6>` normalmente en tus páginas.
También puedes utilizar la etiqueta `<small>` para agregar texto secundario como se muestra en el siguiente ejemplo:

Alineación

Utiliza las clases `text-left`, `text-center`, `text-right` y `text-justify` en párrafos para alinearlos a la izquierda, centro, derecha y justificados respectivamente.

Tablas

Para aplicarle los estilos de **Bootstrap** a las tablas debes agregar la clase `table`:

```
<table class="table">
  ...
</table>
```

Veamos un ejemplo que puedes editar para probar diferentes configuraciones:

Ver la documentación de Bootstrap.

Tablas con bandas

Agrega la clase `table-striped` para intercalar el fondo de las filas entre blanco y gris:

```
<table class="table table-striped">
  ...
</table>
```

Tablas con bordes

Agrega la clase `table-bordered` para agregar bordes a la tabla:

```
<table class="table table-bordered">
  ...
```

```
</table>
```

Resaltar la fila sobre la que pasa el mouse

Utiliza la clase `table-hover` para que cuando pases el mouse sobre una fila cambie el fondo a gris:

```
<table class="table table-hover">
  ...
</table>
```

Filas contextuales

Puedes utilizar las clases `active`, `success`, `info`, `warning` y `danger` sobre los `<tr>`, `<th>` o `<td>` para cambiar el color de las filas o celdas:

```
<!-- En filas -->
<tr class="active">...</tr>
<tr class="success">...</tr>
<tr class="warning">...</tr>
<tr class="danger">...</tr>
<tr class="info">...</tr>

<!-- En celdas (`td` or `th`) -->
<tr>
  <td class="active">...</td>
  <td class="success">...</td>
  <td class="warning">...</td>
  <td class="danger">...</td>
  <td class="info">...</td>
</tr>
```

Abrir en la documentación de Bootstrap

Formularios

Bootstrap le aplica algunos estilos a los formularios pero debes seguir la siguiente estructura:

```
<form>
  <div class="form-group">
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" class="form-control">
  </div>
  <button type="submit" class="btn btn-default">Enviar</button>
</form>
```

Para tener en cuenta:

- Encierra los campos de entrada con un `div` que tenga la clase `form-group`.
- Agrégale la clase `form-control` al `input`.
- Utiliza un `<button>` con `type="submit"`. En la siguiente sección vamos a hablar más de los botones.

Para los checkboxes utiliza la siguiente estructura:

```
<div class="checkbox">
  <label>
    <input type="checkbox"> Check me out
  </label>
</div>
```

Existen muchas variaciones sobre los formularios. Para más información abre la documentación en Bootstrap.

Botones

Agrega la clase `btn` sobre las etiquetas `<a>` y `<button>`, en conjunto con una de las siguientes clases para crear un botón más estilizado: `btn-default`, `btn-primary`, `btn-success`, `btn-info`, `btn-warning`, `btn-danger` o `btn-link`.

```
<button type="button" class="btn btn-default">Default</button>
```

Veamos un ejemplo con diferentes tipos de botones:

Abrir en la documentación de Bootstrap

Tamaños

Utiliza las clases `btn-lg`, `btn-sm` y `btn-xs` para cambiar el tamaño del botón.

Mostrar en bloque

Utiliza la clase `btn-block` para que el botón ocupe el 100% del elemento que lo contiene.

Botones deshabilitados

Agrega la propiedad `disabled` sobre los `<button>` y la clase `.disabled` sobre los `<a>` para que el botón parezca deshabilitado:

```
<a href="#" class="btn btn-primary disabled">Primary link</a>
```

Imágenes

Para hacer las imágenes **responsive** agrega la clase `img-responsive` a `<img>`:

```
<img src="..." class="img-responsive" alt="Responsive image">
```

Utiliza la clase `img-rounded` para aplicarle borders redondeados a la imagen:

```
<img src="..." alt="..." class="img-rounded">
```

Utiliza la clase `img-circle` para que la imagen aparezca en un círculo:

```
<img src="..." alt="..." class="img-circle">
```

Utiliza la clase `img-thumbnail` para agregarle un bordes redondeados y un borde adicional a la imagen:

```
<img src="..." alt="..." class="img-thumbnail">
```

La grilla

La grilla nos permite organizar nuestro contenido en la pantalla más facilmente utilizando una combinación de **contenedores**, **filas** y **secciones** que pueden ocupar de **1 a 12 columnas**.

El siguiente ejemplo muestra un contenedor con **2 filas**. La primera fila tiene **12 secciones** (cada una de **1 columna**) y la segunda fila tiene **2 secciones** (cada una de **6 columnas**).

Fíjate que entre las **secciones** hay un espacio, llamado gutter. Por defecto ese espacio (padding) es de `15px` a cada lado. Entre una fila y otra no existe ningún espacio.

Contenedores

Hasta ahora no lo hemos mostrado, pero las **filas** siempre deben estar envueltas en un **contenedor**.

Un **contenedor** puede contener muchas **filas**.

Existen dos tipos de contenedores: de **ancho fijo** y de **ancho variable**.

Contenedores de ancho fijo

Los contenedores de **ancho fijo** se crean utilizando una etiqueta `<div>` con clase `container`:

```
<div class="container">

</div>
```

Los contenedores de **ancho fijo** ocupan:

- El 100% de la pantalla en teléfonos móviles (`xs`).
- `768px` en tabletas (`sm`).
- `992px` en pantallas de escritorio (`md`).
- `1200px` en pantallas grandes (`lg`).

Contenedores de ancho variable

Los contenedores de **ancho variable** (o fluidos) ocupan siempre el 100% de la página y se crean utilizando una etiqueta `<div>` con clase `container-fluid`:

```
<div class="container-fluid">

</div>
```

Filas

Para definir una **fila** se utiliza una etiqueta `<div>` con clase `row`:

```
<div class="row">
  <!-- acá van las secciones -->
</div>
```

Cada **fila** se puede dividir en **secciones** que pueden ocupar de **1 hasta 12 columnas**.

Para definir una **sección** se utiliza un `<div>` con una clase que comienza con alguno de los prefijos `col-xs-`, `col-sm-`, `col-md-` o `col-lg-` seguido del número de columnas que uno quiere que ocupe esa **sección**.
Nota: más adelante entenderás qué significan los prefijos, por ahora siempre vamos a usar `col-sm-`.
Por ejemplo, la siguiente **fila** está dividida en dos **secciones**, cada una de **6 columnas**:

```
<div class="row">
  <div class="col-sm-6"></div>
  <div class="col-sm-6"></div>
</div>
```

El siguiente ejemplo tiene una fila dividida en **3 secciones**, cada una de **4 columnas**:

```
<div class="row">
  <div class="col-sm-4"></div>
  <div class="col-sm-4"></div>
  <div class="col-sm-4"></div>
</div>
```

Anidando filas

Dentro de una **sección** puedes anidar otra **fila** que nuevamente se va a dividir en **12 columnas** como se muestra a continuación:

El siguiente ejemplo tiene una **fila** que se divide en dos **secciones**, cada una de **6 columnas**. Dentro de la primera **sección** estamos definiendo otra **fila** con una **sección** de **12 columnas**:

```
<div class="row">
  <div class="col-sm-6">
    <div class="row">
      <div class="col-sm-12"></div>
    </div>
  </div>
  <div class="col-sm-6"></div>
</div>
```

Esto nos permite distribuir los elementos de la página en configuraciones bastante complejas.

Diferentes tipos de pantalla

En **Bootstrap**, `xs` (extra small) se refiere a teléfonos móviles (menos de `768px`).
`sm` (small) se refiere a tabletas (entre `768px` y `992px`).
`md` (medium) se refiere a pantallas de escritorio (entre `992px` y `1200px`).
`lg` (large) se refiere a pantallas de más de `1200px`.

Los prefijos de las secciones

Los prefijos `col-xs-`, `col-sm-`, `col-md-` y `col-lg-` determinan cuántas columnas ocupa la sección en cada dispositivo.
Por ejemplo, si queremos que en teléfonos móviles (`xs`) y tabletas (`sm`) una sección ocupe **12 columnas** y en escritorio (`md`) y pantallas más grandes (`lg`) ocupe **6 columnas** podemos utilizar las siguientes clases:

```
<div class="col-xs-12 col-sm-12 col-md-6 col-lg-6"></div>
```

Sin embargo, en este ejemplo podemos omitir la última clase (`col-lg-6`) porque las clases siempre aplican hacia arriba, entonces `col-md-6` va a aplicar también a `lg`.
También podemos omitir `col-xs-12` y `col-sm-12` porque ese es el valor por defecto. Por lo tanto, quedaría así:

```
<div class="col-md-6"></div>
```

Quizá el prefijo más común es `col-sm-` para que en teléfonos móviles las **secciones** se vean de forma vertical (una encima de otra) y en los demás dispositivos de forma horizontal (una al lado de la otra **siempre y cuando no superen las 12 columnas**), pero eso depende de cada caso.
No se recomienda mezclar prefijos en una misma fila. Por ejemplo, si quieres dividir la fila en 3 secciones de diferentes tamaños asegúrate que siempre utilicen los mismos prefijos. Por ejemplo:

```
<div class="row">
  <div class="col-sm-3"></div>
  <div class="col-sm-4"></div>
  <div class="col-sm-5"></div>
</div>
```

En este ejemplo las secciones se van a ver de forma vertical en teléfonos móviles y horizontal (ocupando 3, 4 y 5 columnas respectivamente) en los demás dispositivos.

Filas y secciones

En una **fila** la suma de las **columnas** que ocupan las **secciones** no necesariamente debe ser 12.

Si las secciones suman menos de 12, las columnas de la derecha quedan vacías. Por ejemplo, el siguiente código solo ocuparía **2 columnas**, las 10 restantes quedarían vacías.

```
<div class="row">
  <div class="col-sm-2"></div>
</div>
```

Si la suma de las secciones ocupa más de 12, las secciones que no quepan aparecen debajo. Por ejemplo, en el siguiente código la última sección aparecería debajo:

```
<div class="row">
  <div class="col-sm-6"></div>
  <div class="col-sm-6"></div>
  <div class="col-sm-6"></div>
</div>
```

Si quieres mover una sección a la derecha puedes utilizar `offsets`. Por ejemplo, para mover una sección 3 columnas a la derecha utilizarías `col-sm-offset-3` (recuerda que también puedes usar `xs`, `md` y `lg`):

```
<div class="row">
  <div class="col-sm-6 col-sm-offset-3"></div>
</div>
```

Ocultando secciones y elementos

Puedes ocultar secciones y elementos de la página para ciertos dispositivos utilizando las clases `hidden-xs`, `hidden-sm`, `hidden-md` y `hidden-lg`.
Por ejemplo, el siguiente div no va a estar visible en `xs` y `sm`:

```
<div class="hidden-xs hidden-sm">Solo es visible en md y lg</div>
```