



## Módulo 1 - Full Stack - Sprint 01

---

## Índice

<b>Sprint 1</b>	<b>4</b>
Introducción Full stack	4
¿Qué es una aplicación web?	5
¿Qué es FrontEnd ?	5
¿Qué es BackEnd?	6
HTML5	6
¿Qué es HTML?	6
¿Qué se necesita para trabajar con HTML?	6
Creando nuestro primer archivo .HTML	6
Historia de la evolución de HTML a HTML5	7
¿Qué es HTML5?	7
Etiquetas HTML5	8
Estructura de una página web HTML5	8
Estructura de la cabecera de un sitio web	9
Estructura del cuerpo de un sitio web	11
Etiqueta audio y video	12
Etiqueta audio	12
Etiqueta video	13
Etiqueta de medios <iframe>	14
Etiqueta div	15
Etiqueta formulario	15
CSS3	18
¿Qué es CSS?	18
Historia y evolución de CSS	18
Sintaxis	20
Propiedades de forma	24
Unidades de CSS	26
Propiedades de color	28
Propiedades de ubicación	30
Propiedades de texto	39
Otros recursos	42
Propiedades de lista	43
Propiedades de enlace o link	46
Box model	48
¿Qué es Box Model?	48
Contenido	48
Padding	50
Border	52

---

Margin	54
Outline	55
Flexbox	56
Conceptos	56
Los dos ejes de flexbox	57
El eje principal	57
El eje secundario	58
Líneas de inicio y de fin	59
El contenedor flex	60
Cambiar flex-direction	61
Contenedores flex Multi-línea con flex-wrap	62
La abreviatura flex-flow	63
Propiedades aplicadas a los ítems flex	64
La propiedad flex-basis	64
La propiedad flex-grow	65
La propiedad flex-shrink	65
Valores abreviados para las propiedades flex	65
Alineación, justificación y distribución del espacio libre entre ítems	67
Align-items	67
justify-content	67
Responsive	68
¿Qué es el diseño web responsive?	68
¿En qué consiste el diseño responsive?	68
Etiqueta Meta Viewport	69
Estableciendo el Viewport	69
Tamaño del contenido para el Viewport	70
Grid-view	71
Media Query	72
Aregar un punto de interrupción o breakpoint	73
Siempre diseñar siguiendo el concepto de Mobile First	74
Breakpoint típicos	74
Frameworks (bootstrap 4, foundation, material)	74
¿Qué es un framework web y qué ventajas aportan?	74
¿Qué es un framework web?	75
Bootstrap 4	76
Introducción a Bootstrap	76
Sistema de grilla y estructura	85
Header y navegación	92
Flexbox en bootstrap	97

---

Cards (Tarjetas)	101
Práctica Bootstrap 4	103
Glosario	103
Bibliografía	105

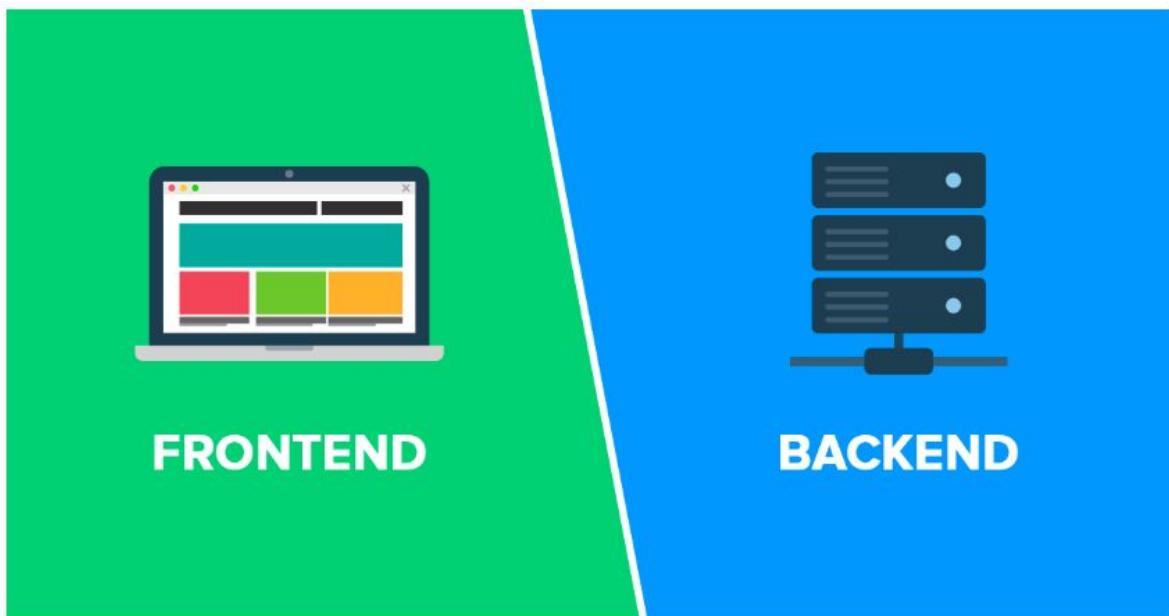
## Sprint 1

### Introducción Full stack

Un desarrollador Full Stack es el encargado de manejar cada uno de los aspectos relacionados con la creación y el mantenimiento de una aplicación web. Para ello es fundamental que el desarrollador Full Stack tenga conocimientos en desarrollo **Front-End** y **Back-End**. Además de manejar diferentes sistemas operativos y lenguajes de programación.

Dicho de otra forma... El desarrollador Full Stack es un profesional todoterreno. Una pieza fundamental en los departamentos de desarrollo de cualquier empresa. Conoce cómo se diseña la aplicación web basándose en principios del diseño UX/UI y además, sabe programarla.

Frontend es la parte de un sitio web que interactúa con los usuarios, por eso decimos que está del lado del cliente. Backend es la parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web, por eso decimos que el backend corre del lado del servidor. Estos dos conceptos explican, a grandes rasgos, cómo funciona una página web y son fundamentales para cualquier persona que trabaje en el mundo digital, ya sea en programación, marketing, diseño o emprendimiento.



## ¿Qué es una aplicación web?

Las aplicaciones web son herramientas a las que se accede a través de cualquier navegador (Google chrome, Firefox, Edge, etc) sin necesidad de contar con un programa instalado previamente. Sin duda cuentan con grandes ventajas, como la independencia del sistema operativo o la facilidad para actualizarlo. Sin olvidar la capacidad para almacenar la información de forma permanente en servidores web.

## ¿Qué es FrontEnd ?

Frontend es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

HTML, CSS y JavaScript son los lenguajes principales del Frontend, de los que se desprenden una cantidad de frameworks y librerías que expanden sus capacidades para crear cualquier tipo de interfaces de usuarios. React, Redux, Angular, Bootstrap, Foundation, LESS, Sass, Stylus y PostCSS son algunos de ellos.

## ¿Qué es BackEnd?

Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. El Backend también accede al servidor, que es una aplicación especializada que entiende la forma como el navegador solicita cosas.

Algunos de los lenguajes de programación de Backend son Python, PHP, Ruby, C# y Java, y así como en Frontend, cada uno de los anteriores tiene diferentes frameworks que te permiten trabajar mejor según el proyecto que estás desarrollando.

## HTML5

### ¿Qué es HTML?



El HTML es un lenguaje usado para desarrollar la estructura de una página web, las siglas HTML significa Hypertext Markup Language o en español lenguaje marcado de hipertexto.

Este lenguaje es codificado por los navegadores para que los usuarios puedan ver la información contenida en la página web.

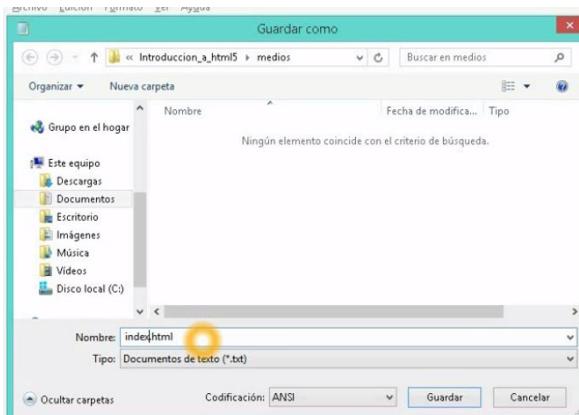
### ¿Qué se necesita para trabajar con HTML?

Para trabajar con HTML solo se necesita un editor de texto plano. Existen muchos programas de este tipo por ejemplo: bloc de notas, sublime text, aptana, brackets editor, visual code, atom, etc.

#### Creando nuestro primer archivo .HTML

1. Abrir el bloc de notas

- 
2. presionar el botón “guardar como” y nombrar el archivo index.html (siempre los navegadores van a buscar por defecto el nombre index.html como nuestra página principal).
  3. Abrimos el archivo creado, en el navegador que tengan instalado por ejemplo google chrome.



## Histórica de la evolución de HTML a HTML5

- 1991 - Nacimiento de HTML
- 1994 - Se funda W3C (World Wide Web Consortium) - Organismo oficial encargado de publicar estándares y recomendación para la web.
- 1998 - HTML 4.0 incluye soporte para scripts, hojas de estilo CSS, XML.
- 2000 - XHTML 1.0 Reformulación de HTML 4.0 basada en XML
- 2003 - XFORMS (formularios de HTML)
- 2004 - Se funda WHATWG (Web Hypertext Application Technology Working Group) (grupo de trabajo formado por Apple, Mozilla y Opera, trabajan para continuar el desarrollo HTML manteniendo la compatibilidad con versiones anteriores).
- 2008 - HTML5 surge del trabajo conjunto de W3C y WHATWG

## ¿Qué es HTML5?



**HTML5** es la última versión de HTML. El término representa dos conceptos diferentes: Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto más amplio de tecnologías que

---

permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance.

## Etiquetas HTML5

El lenguaje de marcado HTML se caracteriza por que se construye a partir de etiquetas (tags). Cuando decimos que HTML es un lenguaje de hipertexto significa que tiene texto oculto, y este se escribe dentro de etiquetas.

- Una etiqueta son palabras claves que encerramos con los signos < y >, dentro de estos signos escribimos el nombre de la etiqueta: <etiqueta>
- La mayoría de las etiquetas se deben escribir en par, es decir, una etiqueta de apertura y otra de cierre: <etiqueta> </etiqueta>
- Si necesitan buscar información sobre el uso de una etiqueta pueden hacerlo con el nombre de tag HTML.

## Estructura de una página web HTML5

La estructura de una página web HTML5 se construye de la siguiente manera:

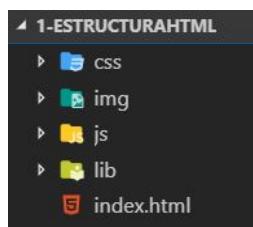


```
index.html •
1  <!DOCTYPE html>
2
3  <html>
4
5      <head>
6
7          </head>
8
9
10     <body>
11
12         </body>
13
14 </html>
15
16
```

- <!DOCTYPE html> Doctype es una etiqueta simple que no necesita cierre, su función es facilitar información del tipo de documento al servidor web que aloja la página, además es necesaria para la comunicación y entre el navegador y el servidor.

- 
- `<html> </html>` es una etiqueta par, es la que va a encerrar todo el documento html.
  - `<head> </head>` esta etiqueta par, define la cabecera del documento y contiene información del mismo, por ejemplo metadatos, scripts, estilos, ubicación de estilos, título de la página entre otros datos. Todo lo que esté dentro de estas etiquetas no es visible para el usuario.
  - `<body> </body>` Es una etiqueta par y es conocida como el cuerpo del sitio web, todo lo que se escriba dentro de esta etiqueta será visible por los navegadores.

Todos los sitios web, deben ser guardados como un proyecto por eso creamos las siguientes carpetas estándar que tiene un proyecto web.



1. Css: Esta carpeta contendrá las hojas de estilo css.
2. img: Esta carpeta contendrá las imágenes del proyecto
3. js: Esta carpeta contendrá los archivos de javascript
4. lib: Esta carpeta contendrá librerías si son necesarias.

Nota: Ver código en repositorio html/01-EstructuraHTML

## Estructura de la cabecera de un sitio web

A partir del proyecto anterior, veremos que componentes van dentro de la etiqueta `<head> </head>`.

`<meta>` Esta es una etiqueta simple, su función es añadir información de la página, los buscadores como google, consultan información de la etiqueta `<meta>` buscando coincidencias con lo que el usuario pretende encontrar. También se usa para agregar que tipo de idioma estamos trabajando en el lenguaje de marcado, para establecer esto las etiquetas poseen atributos.

```
index.html ✘
1  <!DOCTYPE html>
2
3  <html>
4
5
6
7  <head>
8
9    <meta charset="utf-8">
10   <meta name="author" content="Tutorial html Rollingcode">
11   <meta name="description" content="Agregar una descripción del sitio">
12   <meta name="keywords" content="tutorial, html, desarrollo web">
13
14   <title> Estructura de la cabecera</title>
15
16   <link rel="icon" href="img/icono.png">
17 </head>
18
19
20 <body>
21
22
23 </body>
24
25 </html>
```

- `<meta charset="utf-8">` Establece que nuestro sitio utiliza caracteres en español esto es “utf-8”
- `<meta name="author" content="Tutorial html Rollingcode">` Establece el autor del sitio.
- `<meta name="description" content="Agregar una descripción del sitio">` Establece la descripción que es utilizada posteriormente por los buscadores como google.
- `<meta name="keywords" content="tutorial, html, desarrollo web">` Establece una serie de palabras claves, que también son necesarias para mejorar los resultados en las búsquedas de google.
- `<title> Estructura de la cabecera</title>` Indica el título que aparece en la pestaña del navegador al ingresar a la web.
- `<link rel="icon" href="img/icono.png">` establece el icono o favicon que tendrá nuestra web, el atributo `href="..."` indica donde está ubicado ese icono.



---

Más información: Pueden encontrar mayor información acerca de las etiquetas html en las siguientes páginas:

1. [https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5\\_lista\\_elementos#Metadatos\\_del\\_documento](https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos#Metadatos_del_documento)
2. <https://www.w3schools.com/html/default.asp>

Nota: Ver código en repositorio html/02-EstructuraHead

### Estructura del cuerpo de un sitio web

## Estructura semántica HTML5



- `<header> </header>` Encabezado
- `<nav> </nav>` Barra de navegación
- `<section> </section>` Sección del sitio
- `<aside> </aside>` Sirve para identificar columnas
- `<article> </article>` Sirve para escribir artículos
- `<h1> </h1>` Etiqueta para escribir títulos, hay etiqueta de varios tamaños.
- `<p> </p>` Etiqueta de párrafo para escribir texto

- 
- **<b></b>** resaltar el texto que está contenido dentro
  - **<br>** salto de línea
  - **<hr>** Crea una línea
  - **<a href=""> </a>** Sirve para crear un vínculo, siempre lleva el atributo href para indicar a dónde se dirige ese vínculo.
  - **<img>** Esta etiqueta simple permite mostrar imágenes, hay que agregar el atributo src con la ubicación de la imagen.
  - **<ol> </ol>** Etiqueta para crear un conjunto de listas ordenada
  - **<li> </li>** Son los ítems de la lista
  - **<ul> </ul>** Etiqueta para crear una lista que no esté numerada.
  - **<table> </table>** Etiqueta para crear tablas.
  - **<tr> </tr>** Crea una fila
  - **<th> </th>** Para crear columnas
  - **<footer> </footer>** Etiqueta para crear el pie de página

Ver el resto de las etiquetas existentes de html5 en la siguiente web:

[https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5\\_list\\_elementos](https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_list_elementos)

Nota: Ver código en repositorio html/03-EstructuraBody

## Etiqueta audio y video

En HTML5 tenemos la ventaja de poder agregar archivos de audio y video.

### Etiqueta audio



**<audio> </audio>** dentro de esta etiqueta podemos agregar varios atributos como los mencionados a continuación:

```
<audio src="audio.mp3" preload="auto" controls></audio>
```

autoplay: esto reproducirá nuestro audio sin que apremos el botón play.

---

controls: mostrará los controles de reproducción.

loop: reproducirá nuestro audio constantemente.

Los **formatos de audio** más comunes que podemos usar, son los siguientes:

- **.ogg**: es un formato comprimido, de poco peso y es ideal para reducir el costo de ancho de banda, está habilitada para los navegadores de firefox, chrome y opera
- **.mp3**: es el formato más popular, es un archivo comprimido. Se puede usar en navegadores como safari, chrome, internet explorer, firefox.
- **.wav**: también es un formato muy conocido, el problema es que no tiene un sistema de comprensión, por lo que consume mayor cantidad de ancho de banda.
- **.acc**: es un formato que logra gran compresión de datos sin perder calidad, fue diseñado para reemplazar el archivo mp3.

Nota: Ver código en repositorio html/04-EstructuraAudio

ver mas información aqui: <https://developer.mozilla.org/es/docs/Web/HTML/Elemento/audio>

## Etiqueta video



`<video> </video>` dentro de esta etiqueta podemos agregar los mismos atributos vistos para audio y también podemos modificar el tamaño de alto y ancho del video.

```
<video src="..." controls autoplay loop> Tu navegador no implementa  
el elemento <code>video</code>. </video>
```

Extensiones admitidas:

- **.ogv**
- **.mp4**
- **.webm**

---

Nota: Ver código en repositorio html/05-EstructuraVideo

ver mas información aqui:

[https://developer.mozilla.org/es/docs/Web/HTML/Usando\\_audio\\_y\\_video\\_con\\_HTML5](https://developer.mozilla.org/es/docs/Web/HTML/Usando_audio_y_video_con_HTML5)

### Etiqueta de medios <iframe>

La etiqueta <iframe> nos permite Insertar o incrustar, documentos html, un video en línea, mapas, un reproductor de sonido en línea u otros elementos.

Veamos diferentes ejemplos de <iframe>

```
<!-- ETIQUETA PARA INSERTAR MULTIMEDIA O PÁGINAS EXTERNAS -->
<iframe width="500" height="280"
src="https://www.youtube.com/embed/MJIZ2gf3Wa8" frameborder="0"
allowfullscreen></iframe>

<iframe width="500" height="281"
src="https://player.vimeo.com/video/98417189" frameborder="0"
allowfullscreen></iframe>

<iframe width="500" height="281" frameborder="no"
src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/96144978&auto_play=false&hide_related=false&show_comments=true&show_user=true&show_reposts=false&visual=true"></iframe>

<iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3560.1061027457
31!2d-65.20935448518014!3d-26.836577383160417!2m3!1f0!2f0!3f0!3m2!1i1024!2
i768!4f13.1!3m3!1m2!1s0x94225c0e8d0271b7%3A0x7946062ac490db30!2sGral.+Paz+
576%2C+T4000+San+Miguel+de+Tucum%C3%A1n%2C+Tucum%C3%A1n!5e0!3m2!1ses-419!2
sar!4v1553006723151"
width="600" height="450" frameborder="0" style="border:0"
allowfullscreen></iframe>

<iframe src="https://www.lagaceta.com.ar/" width="100%"
height="600" scrolling=" auto " frameborder="0" AllowTransparency="true
"></iframe>
```

Como vemos en los ejemplos anteriores, podemos incrustar con <iframe> videos, mapas, páginas web dentro de nuestra web.

---

Nota: Ver código en repositorio html/06-Etiquetalframe

## Etiqueta div

La etiqueta `<div> </div>` se emplea para definir un bloque de contenido o sección de la página, para poder aplicarle diferentes estilos e incluso para realizar operaciones sobre ese bloque específico.

ejemplo:

```
<h1>Tim Berners-Lee</h1>
<div style="color: #040; font-style: italic">
  <p>Timothy "Tim" John Berners-Lee es un científico de la
  computación británico, conocido por ser el padre de la Web.
  Estableció la primera comunicación entre un cliente y un servidor
  usando el protocolo HTTP en noviembre de 1989.</p>
  <p>En octubre de 1994 fundó el Consorcio de la World Wide Web
  (W3C) con sede en el MIT, para supervisar y estandarizar el
  desarrollo de las tecnologías sobre las que se fundamenta la Web y
  que permiten el funcionamiento de Internet.</p>
</div>
```

## Etiqueta formulario

En HTML5 podemos crear formularios para que las personas puedan interactuar con la página.

la etiqueta a utilizar es `<form> </form>` y podemos agregarle muchos atributos, veamos algunos de ellos:

`<input>` Etiqueta de entrada de datos, uno de sus atributos es `type` que es el tipo de entrada que voy a utilizar, existen muchos tipos de entrada:

Algunos valores que puede tomar el atributo `type`:

- Text
- Button

- 
- checkbox
  - color
  - date
  - datetime
  - email
  - file

otro atributo muy utilizado es **placeholder**, sirve como una guía para mostrarle al usuario que puedo poner en ese campo.

Todo lo que escribo en el campo **<input>** será almacenado en un atributo llamado **value**, también puede establecer un valor por defecto que tendrá el atributo **value**.

```
<input id="texto" type="text" placeholder="Entrada de texto" value="" required>
```

Entrada de texto

Veamos el siguiente ejemplo de formulario para ver el uso de las distintas etiquetas y atributos de formulario:

```
1 <form>
2
3     <label for="texto">Entrada de texto</label>
4     <input id="texto" type="text" placeholder="Entrada de texto" value="" required>
5
6     <br>
7     <br>
8
9     <label for="correo">Entrada de correo</label>
10    <input id="correo" type="email" placeholder="Entrada de correo" value="" required>
11
12    <br>
13    <br>
14
15    <label for="pass">Entrada de contraseña</label>
16    <input id="pass" type="password" placeholder="Entrada de contraseña" value="" required>
17
18    <br>
19    <br>
20
21    <input type="number" value="1" min="0" max="3">
22
23    <br>
24    <br>
25
26    <select>
27        <option>Opcion 1</option>
28        <option>Opcion 2</option>
29        <option>Opcion 3</option>
30    </select>
31
32    <br>
33    <br>
34
35    <input type="checkbox" id="rojo"><label for="rojo">Rojo</label>
36    <input type="checkbox" id="amarillo"><label for="amarillo">Amarillo</label>
37    <input type="checkbox" id="verde"><label for="verde">Verde</label>
38
39    <br>
40    <br>
41
42    <input type="radio" id="blanco" name="radio" checked><label
for="blanco">Blanco</label>
43        <input type="radio" id="negro" name="radio"><label for="negro">Negro</label>
44
45    <br>
46    <br>
47
48    <input type="submit" value="Enviar Formulario">
49
50 </form>
```

---

Si visualizamos este ejemplo en el navegador, veremos el siguiente formulario:

Entrada de texto

Entrada de correo

Entrada de contraseña

Opcion 1 ▾

Rojo  Amarillo  Verde

Blanco  Negro

Nota: Ver código en repositorio html/07-EtiquetaForm

## CSS3

### ¿Qué es CSS?



El CSS es un lenguaje usado para definir y crear la presentación gráfica y visual de una página web. Su nombre proviene de la sigla (en inglés de Cascading Style Sheets), en español "Hojas de estilo en cascada". Entonces el css nos ayuda a dar estilos para mejorar la apariencia de la web que construimos en html.

### Historia y evolución de CSS



---

CSS se ha creado en varios niveles y perfiles. Cada nivel se construye sobre el anterior, generalmente añadiendo nuevas funciones.

Los perfiles son, parte de uno o varios niveles de CSS definidos para un dispositivo particular. Actualmente, pueden usarse perfiles para dispositivos móviles.

## CSS1

La primera especificación oficial de CSS, recomendada por la W3C fue CSS1, publicada en diciembre 1996, y abandonada en abril de 2008.

Algunas de las funcionalidades que ofrece son:

- Propiedades de las fuente, como tipo, tamaño, énfasis, etc.
- Color de texto, fondos, bordes u otros elementos.
- Atributos del texto, como espaciado entre palabras, letras, líneas, etcétera.
- Alineación de textos, imágenes, tablas u otros.
- Propiedades de caja, como margen, borde, relleno o espaciado.
- Propiedades de identificación y presentación de listas.

## CSS2

La especificación CSS2 fue desarrollada por la W3C y publicada como recomendación en mayo de 1998, y abandonada en abril de 2008.

Como ampliación de CSS1, se ofrecieron, entre otras las siguientes características:

- Funcionalidades propias de las capas como de posicionamiento relativo, niveles, etcétera.
- El concepto de “media types”.
- Soporte para las hojas de estilo auditivas
- Texto bidireccional, sombras, etc.
- La primera revisión de CSS2, usualmente conocida como “CSS 2.1”, corrige algunos errores encontrados en CSS2, elimina funcionalidades inoperables en los navegadores y añade alguna nueva especificación.

## CSS3

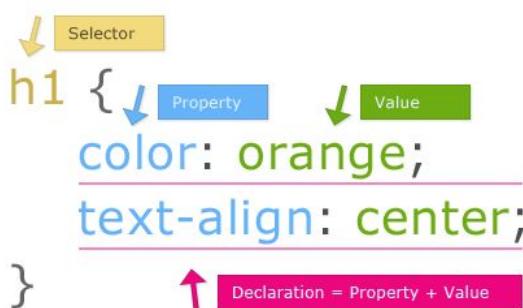
A diferencia de CSS2, que fue una gran especificación que definía varias funcionalidades, CSS3 está dividida en varios documentos separados, llamados “módulos”.

Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad.

Los trabajos en el CSS3, comenzaron a la vez que se publicó la recomendación oficial de CSS2, y los primeros borradores de CSS3 fueron liberados en junio de 1999.

Debido a la modularización del CSS3, diferentes módulos pueden encontrarse en diferentes estados de su desarrollo, con el tiempo CSS3 se convirtió en la recomendación oficial de la W3C.

## Sintaxis



```
<style>
/*
selector{
    propiedad: valor;
}
*/
body {
    background: blue;
}

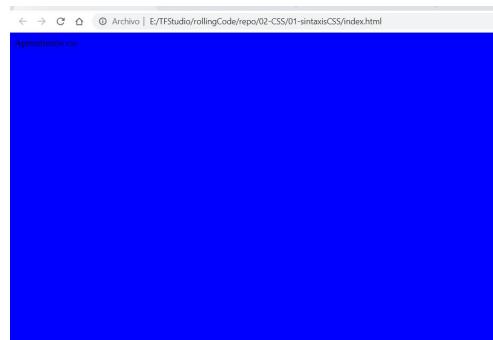
</style>
```

**Selector:** puede ser el nombre de la etiqueta, el id de la etiqueta, la clase de la etiqueta o el atributo de la etiqueta.

**Propiedad:** Se escribe dentro de llaves, puede ser una propiedad de objeto, forma, texto o de color. Siempre finalizó con punto y coma (`;`)

Para este ejemplo, creamos una página web y dentro de la etiqueta `<head> </head>` agregamos la etiqueta `<style></style>`, con esto indicamos que lo que está dentro de esa etiqueta serán los estilos que usará nuestra página, esta es una forma de agregar css a nuestro sitio, pero no es la que utilizaremos usualmente.

Si visualizamos en el navegador este ejemplo, veremos lo siguiente:



Si queremos **llamar un estilo a través del id**, crearemos el `<div>` con el id que queramos, luego en la sección de estilos llamaremos al id comenzando con el símbolo numeral (#) delante del id, de la siguiente manera `#id { ... }`

Ejemplo:

```
<style>
    /*
selector{
    propiedad: valor;
}
*/
body {
    background: blue;
}

#caja {
    width: 150px;
    height: 150px;
    background: yellow;
}
</style>
```

```
<body>
    <header>Aprendiendo css</header>
    <br>
    <div id="caja">Esta es una caja de prueba</div>
</body>
```

Si vemos este ejemplo en el navegador, visualizamos lo siguiente:



También podemos aplicar un **estilo** css a un **grupo de cajas** `<div>`, para ello podemos hacer uso del atributo clase del `<div class="...>`.

*Para tener en cuenta, los nombres de los atributos son palabras libres, pero se recomienda que no comience con signos o caracteres especiales y que tampoco comiencen con números.*

Entonces, creamos varias cajas con el mismo atributo class, luego en nuestra hoja de estilo podemos hacer referencia a esa clase comenzando con el carácter punto (.) seguido del nombre de la clase: `.nombreclase {... }`

Ejemplo:

```
<style>
/*
selector{

    propiedad: valor;
}
*/


body {
    background: ■blue;
}

#caja {
    width: 150px;
    height: 150px;
    background: ■yellow;
}

.grupo {
    width: 150px;
    height: 150px;
    background: ■orange;
    margin: 10px;
}

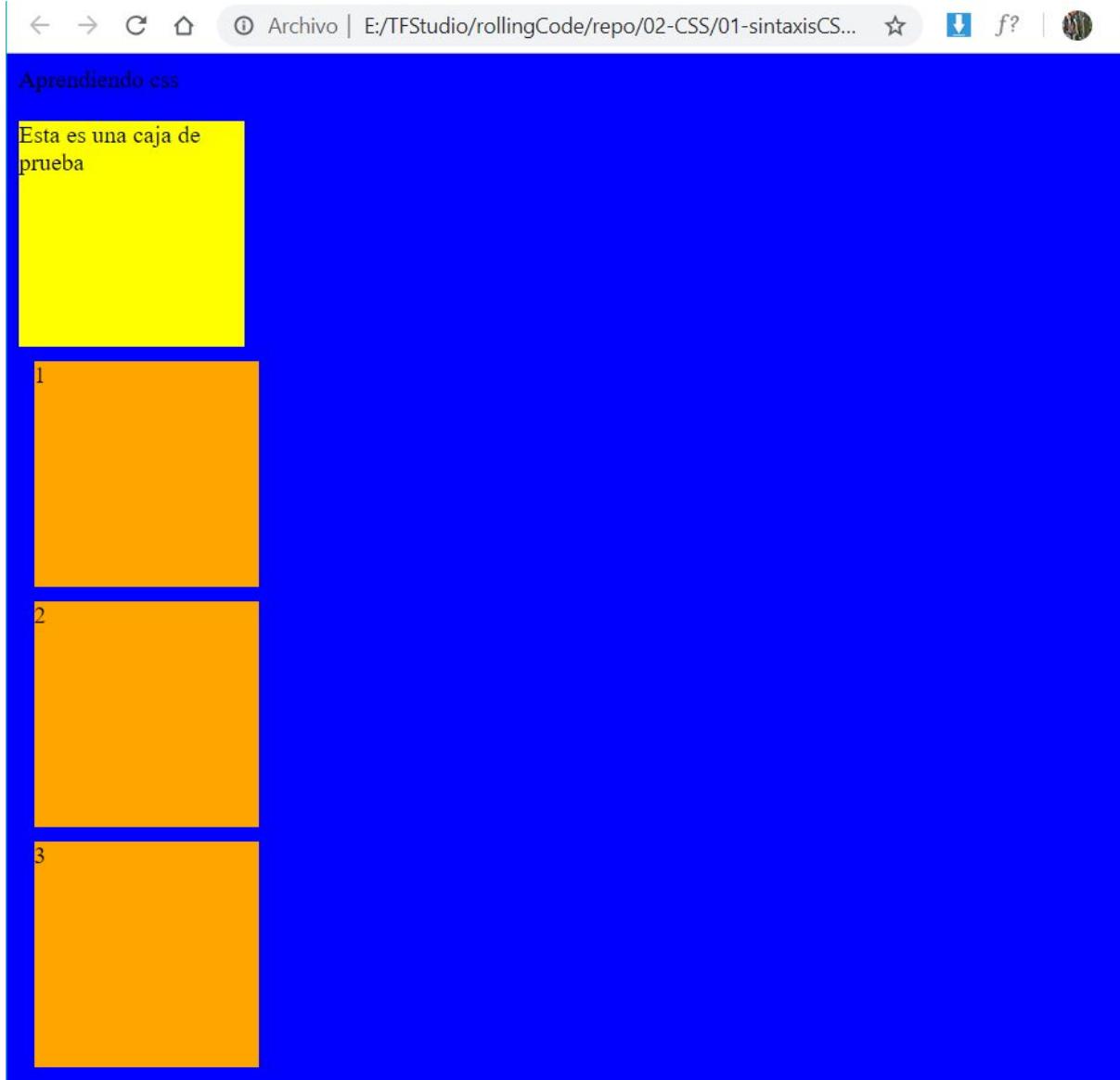
</style>
```

```
<body>

<header>Aprendiendo css</header>
<br>
<div id="caja">Esta es una caja de prueba</div>

<div class="grupo">1</div>
<div class="grupo">2</div>
<div class="grupo">3</div>
</body>
```

Si visualizamos el ejemplo en un navegador, veremos lo siguiente:



Nota: Ver código en repositorio CSS/01-sintaxisCSS

## Propiedades de forma

Las propiedades de forma son las que nos van a permitir dar el tamaño, espacio y color a las cajas <div> que estemos maquetando en el HTML.

Las más comunes son:

- width: Ancho
- height: Alto
- background: Color de fondo

- 
- margin: margen externo
  - border: color y tipo de borde, esta propiedad tiene tres parámetros, ancho de borde, tipo de borde, color. Por ejemplo: `border: 5px solid blue;`
  - padding: margen interno

Ejemplo de estas propiedades:

Creamos un archivo con el siguiente código:

```
18  <style>
19      header {
20          width: 1000px;
21          height: 200px;
22          background: #grey;
23          margin: 10px;
24          border: 5px solid #darkblue;
25          padding: 20px;
26      }
27
28      #logo {
29          width: 200px;
30          height: 200px;
31          background: #yellow;
32          border: #purple;
33          border-radius: 10px;
34      }
35  </style>
36 </head>
37
38
39 <body>
40
41     <header>
42         <div id="logo"></div>
43     </header>
44
45
46 </body>
47
48 </html>
```

si lo visualizamos en el navegador observaremos lo siguiente:



Nota: Ver código en repositorio CSS/02-propiedadesDeForma

Podemos consultar más información de las propiedades CSS en la siguiente web:

<https://www.w3schools.com/css/default.asp>

## Unidades de CSS

Un paso muy importante en el aprendizaje web es entender las unidades, tamaños y medidas que podemos utilizar en CSS. Una forma clara de entenderlo y llevarlo al mundo real es imaginar que vamos a dibujar algo, para empezar necesitaremos una hoja de papel, elegir el tamaño de la misma y el tipo de lápices que vamos a utilizar. Sin embargo, estos últimos no son tan importantes como lo son los trazos que podemos hacer con ellos. Dicho esto, ahora lo llevaremos al mundo virtual (al mundo de la web). El papel serán las diferentes resoluciones de pantalla que abarcan los dispositivos móviles, tabletas, equipos de mesa y portátiles; y los trazos son los distintos tipos de letra (fonts). Específicamente sus tamaños, distancia entre cada palabra e incluso entre cada letra. Las unidades y medidas en CSS están divididas en dos grandes grupos:

- Unidades Absolutas
- Unidades Relativas

### Unidades absolutas

Como su nombre lo indica, son unidades que están completamente definidas. Esto quiere decir que su valor no depende de otro valor de referencia. En este grupo encontramos los siguientes elementos:

- **cm:** Aumenta el tamaño de nuestro texto o elemento en centímetros
- **mm:** Aumenta el tamaño de nuestro texto o elemento en milímetros

- 
- **in:** Aumenta el tamaño de nuestro texto o elemento en pulgadas ( $1\text{in} = 96\text{px} = 2.54\text{cm}$ )
  - **px:** Aumenta el tamaño de nuestro texto en pixeles (Dependiendo del dispositivo, este elemento se clasifica también en Unidades Relativas). Esta unidad es la más utilizada.
  - **pt:** Aumenta el tamaño de nuestro texto en puntos ( $1\text{pt} = 1/72 \text{ de } 1\text{in}$ )
  - **pc:** Aumenta el tamaño de nuestro texto en picas ( $1\text{pc} = 12 \text{ puntos aprox}$ )

## Unidades relativas

Este tipo de unidad es más flexible y se expresa en forma de porcentaje (%). Me refiero a su flexibilidad porque se adapta de acuerdo al tamaño de otro valor de referencia. Por ejemplo, en el mundo real tu hermano y tú tienen dos balones de fútbol. Un balón es grande y el otro es pequeño, pero a cada uno le corresponde la mitad de cada uno de esos balones (50%). No importa el tamaño del balón, porque igual siempre vas a ser dueño del 50%. Entonces, si el balón grande redujera su tamaño, igual tendrías el 50% de este.

Ahora aplicamos este ejemplo a los distintos tipos de resolución de pantalla. Entonces, consideramos que las unidades relativas se subdividen en 2 grupos:

- Relativas a la tipografía
- Relativas al viewport

### Relativas a la tipografía

- **em:** Relativo al tamaño de fuente de un elemento. Si, por ejemplo, utilizamos  $3\text{em}$  (`font-size: 3em;`); mi texto aumentará 3 veces su tamaño en relación al tamaño que se esté utilizando por defecto.
- **ex:** Aumenta el tamaño de mi letra en cuanto a su altura.
- **Rem:** Es igual a (em) con la diferencia de que este actúa en la raíz. O sea, en la letra que esté utilizando por defecto.

### Relativas al viewport

- **vw:** Porcentaje relativo a la anchura del viewport.
- **vh:** Porcentaje relativo a la altura del viewport.
- **vmin:** Entre vw y vh toma el que tenga menor valor.
- **vmax:** Entre vw y vh toma el que tenga mayor valor.

Las unidades relativas al viewport son muy importantes y se deben tomar en cuenta cuando realizamos una web para varios tipos de resolución de pantalla. Combinando esta serie de

medidas con otros elementos que podemos agregar a nuestra sintaxis conseguiremos realizar un excelente trabajo.

Podemos visualizar ejemplos del uso de estos atributos en la siguiente web: [https://www.w3schools.com/css/css\\_units.asp](https://www.w3schools.com/css/css_units.asp)

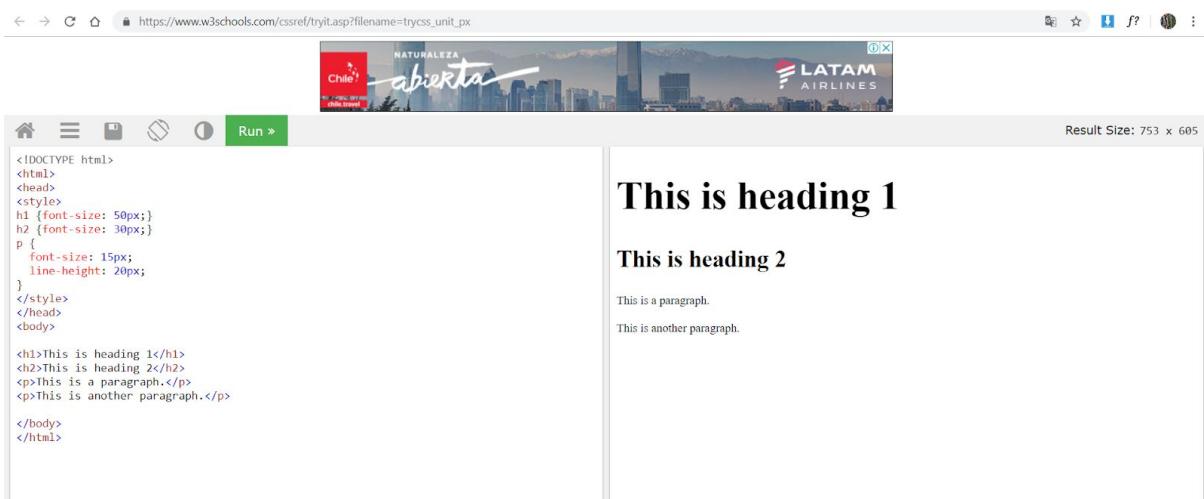
Para ver el ejemplo de pixeles, hacemos clic en el siguiente botón y nos llevará a una web que tiene a la derecha un editor de texto con el ejemplo y a la derecha podemos visualizar como queda el código utilizado.

## Absolute Lengths

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Unit	Description	
cm	centimeters	<a href="#">Try it</a>
mm	millimeters	<a href="#">Try it</a>
in	inches (1in = 96px = 2.54cm)	<a href="#">Try it</a>
px *	pixels (1px = 1/96th of 1in)	<a href="#">Try it</a>
pt	points (1pt = 1/72 of 1in)	<a href="#">Try it</a>
pc	picas (1pc = 12 pt)	<a href="#">Try it</a>



The screenshot shows a browser window with the URL [https://www.w3schools.com/cssref/tryit.asp?filename=trycss\\_unit\\_px](https://www.w3schools.com/cssref/tryit.asp?filename=trycss_unit_px). At the top, there's a navigation bar with back, forward, search, and other browser controls. Below the address bar is a header featuring the Chilean flag and the text "NATURALEZA abierta". To the right of the header is the LATAM Airlines logo. The main content area has a "Run" button at the top left. On the left, there's a code editor with the following HTML and CSS:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {font-size: 50px;}
h2 {font-size: 30px;}
p {
  font-size: 15px;
  line-height: 20px;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

The right side of the editor shows the resulting page output with two headings and two paragraphs. A red oval highlights the "Try it" button for the "px \*" row in the table above.

## Propiedades de color

En las hojas de estilo de CSS, tenemos tres formas de aplicar color:

1. Una forma es colocar la propiedad por ejemplo `background` y establecer el color, usando el nombre del color en inglés. Por ejemplo: `background: red;`
2. Otra forma de establecer color, es usar los colores **Hexadecimales**, estos son códigos de seis dígitos y siempre comienza con un símbolo numeral (#) delante de los números. Por ejemplo: `background: #ff0000;` Podemos encontrar en internet muchas páginas para ver la paleta de colores en hexadecimal, una de ellas es la siguiente: <https://htmlcolorcodes.com/es/>
3. La tercera forma es utilizar la sintaxis **RGBA**, en este caso se utilizan cuatro pares de número separados por coma, el primer par representa rojo, verde, azul, el último par es el grado de transparencia que tendrá el color. También pueden buscar en internet páginas que ofrecen paleta de colores en RGBA. Por ejemplo: `background: rgba(255, 0, 255, 1);`

Ejemplo:

```
<style>
    .grupo {
        width: 200px;
        height: 200px;
        margin: 30px;
    }

    #c1 {
        background: red;
    }

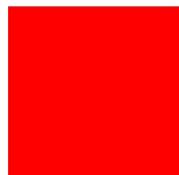
    #c2 {
        background: orange;
    }

    #c3 {
        background: rgba(255, 0, 255, 1);
    }
</style>
</head>

<body>
    <div id="c1" class="grupo"></div>
    <div id="c2" class="grupo"></div>
    <div id="c3" class="grupo"></div>
</body>
```

Si visualizamos el código anterior en el navegador veremos lo siguiente:

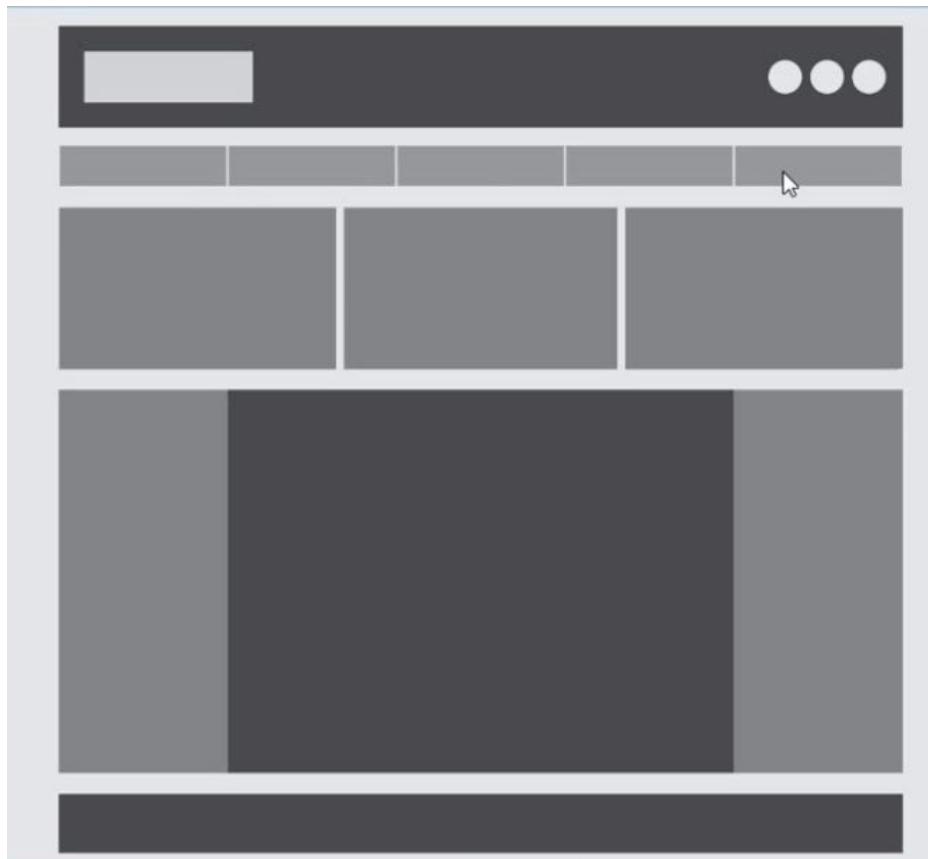
← → ⌂ ⌂ Archivo | E:/TFStudio/rollingCode/repo/02-CSS/03-propiedadDeColor/index.html



Nota: Ver código en repositorio CSS/03-propiedadesDeColor

### Propiedades de ubicación

Cuando maquetamos un sitio web, es importante armar lo que llamamos wireframe, maqueta o planos de un sitio web. Podemos tomar de ejemplo el siguiente esquema:



Podríamos decir de esta imagen, que tenemos el encabezado con unas divisiones internas que podrían ser el logo y los iconos de redes sociales. Luego tenemos otra división que sería la barra de navegación con cinco botones. Luego viene una sección “top o superior” donde podrían ir anuncios o titulares y contiene tres divisiones internas. Luego tenemos la sección principal, que también está dividido en tres partes, podríamos considerar la división central como el artículo principal. Por último tenemos la última división que sería nuestro footer o pie de página.

Cuando maquetamos un sitio web es muy importante conocer las propiedades de ubicación de css.

Los pasos a seguir son los siguientes:

1. Deberíamos construir primero la estructura con HTML5

```

22 <body>
23     
24     <header>
25         <div id="logo"></div>
26
27         <div id="icono1"></div>
28         <div id="icono2"></div>
29         <div id="icono3"></div>
30     </header>
31     
32
33     <!-- comienza barra de navegacion--&gt;
34     &lt;nav&gt;
35
36         &lt;ul&gt;
37             &lt;li class="botones"&gt;&lt;/li&gt;
38             &lt;li class="botones"&gt;&lt;/li&gt;
39             &lt;li class="botones"&gt;&lt;/li&gt;
40             &lt;li class="botones"&gt;&lt;/li&gt;
41             &lt;li class="botones"&gt;&lt;/li&gt;
42         &lt;/ul&gt;
43     &lt;/nav&gt;
44     <!-- fin barra de navegacion--&gt;
45
46     <!-- Comienza seccion top--&gt;
47     &lt;div id="top"&gt;
48
49         &lt;ul&gt;
50             &lt;li&gt;&lt;/li&gt;
51             &lt;li&gt;&lt;/li&gt;
52             &lt;li&gt;&lt;/li&gt;
53         &lt;/ul&gt;
54     &lt;/div&gt;
55
56     <!-- fin seccion top--&gt;
57
58     &lt;!-- Inicia seccion principal--&gt;
59     &lt;section&gt;
60         &lt;aside id="izq"&gt;&lt;/aside&gt;
61         &lt;article&gt;&lt;/article&gt;
62         &lt;aside id="der"&gt;&lt;/aside&gt;
63     &lt;/section&gt;
64     &lt;!-- fin seccion principal--&gt;
65
66     &lt;!-- Inicia footer--&gt;
67     &lt;footer&gt;
68
69     &lt;/footer&gt;
70     &lt;!-- finfooter--&gt;
71 &lt;/body&gt;
</pre>

```

2. Luego realizaremos el maquetado con css para ubicar las divisiones creadas anteriormente.

---

siguiendo el ejemplo del maquetado de la imagen anterior comenzamos a construir nuestro código css para imitar ese maquetado, veremos el siguiente código que representa las cajas principales del sitio y luego el maquetado final.

```
16  <style>
17      * {
18          margin: 0px;
19          padding: 0px;
20          list-style: none;
21      }
22
23      header {
24          position: relative;
25          margin: 20px auto;
26          width: 1000px;
27          height: 120px;
28          background: #444;
29      }
30
31      nav {
32          position: relative;
33          margin: auto;
34          width: 1000px;
35          height: 48px;
36          background: #aaa;
37      }
38
39      #top {
40          position: relative;
41          margin: 20px auto;
42          width: 1000px;
43          height: 192px;
44          background: #888;
45      }
46
47      section {
48          position: relative;
49          margin: auto;
50          width: 1000px;
51          height: 453px;
52          background: #aaa;
53      }
54
55      footer {
56          position: relative;
57          margin: 20px auto;
58          width: 1000px;
59          height: 70px;
60          background: #888;
61      }
62  </style>
63  </head>
```

Hasta este momento, visualizamos lo siguiente en el navegador:



3. Ahora continuamos agregando las cajas internas dentro de las cajas principales.

```
<style>
  * {
    margin: 0px;
    padding: 0px;
    list-style: none;
  }

  header {
    position: relative;
    margin: 20px auto;
    width: 1000px;
    height: 120px;
    background: #444;
  }

  #logo {
    position: absolute;
```

---

```
    top: 30px;
    left: 30px;
    width: 200px;
    height: 60px;
    background: #ccc;
}

.redes {
    position: absolute;
    width: 42px;
    height: 42px;
    background: #ffffff;
    border-radius: 100%;
}

#icono1 {
    top: 42px;
    right: 120px;
}

#icono2 {
    top: 42px;
    right: 69px;
}

#icono3 {
    top: 42px;
    right: 19px;
}

nav {
    position: relative;
    margin: auto;
    width: 1000px;
    height: 48px;
    background: #aaa;
}
```

---

```
.botones {
    float: left;
    width: 196px;
    height: 48px;
    background: #333;
    margin: 0px 2px;
}

#top {
    position: relative;
    margin: 20px auto;
    width: 1000px;
    height: 192px;
    /* background: #888; */
}

#top ul {
    width: 1010px;
    height: 192px;
    /*background: rgba(255, 0, 0, .5); */
}
/*En este caso seguimos la ruta hasta la caja que queremos modificar*/

#top ul li {
    float: left;
    width: 326px;
    height: 192px;
    background: black;
    margin-right: 10px;
}

section {
    position: relative;
    margin: auto;
    width: 1000px;
    height: 453px;
    background: #aaa;
}
```

---

```
/*En este caso llamamos a la parte izquierda de la caja aside, si
ponemos el nombre

aside #izq estamos haciendo referencia a la ruta de otra caja
si usamos aside#izq nos estamos refiriendo a la misma caja*/

aside#izq {
    position: absolute;
    left: 0;
    top: 0;
    width: 200px;
    height: 453px;
    background: #333;
}

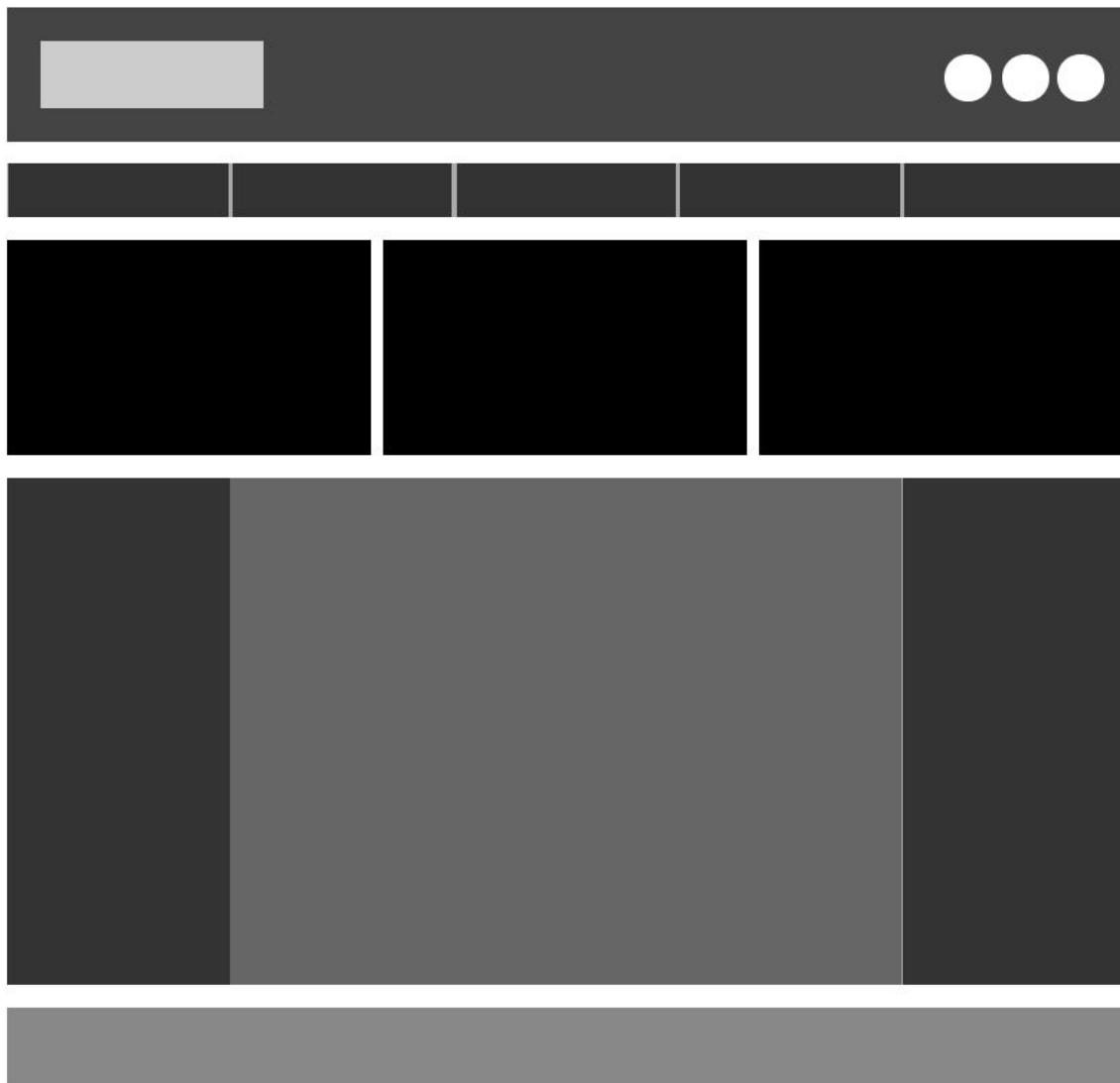
article {
    position: absolute;
    left: 200px;
    top: 0;
    width: 600px;
    height: 453px;
    background: #666;
}

aside#der {
    position: absolute;
    right: 0;
    top: 0;
    width: 200px;
    height: 453px;
    background: #333;
}

footer {
    position: relative;
    margin: 20px auto;
    width: 1000px;
    height: 70px;
    background: #888;
```

```
 }  
</style>
```

Si visualizamos estos cambios en el navegador, veremos lo siguiente:



Nota: Ver código en repositorio CSS/04-propiedadesDeUbicacion

Cuando estamos maquetando podemos maquetar los valores que vienen por defecto en las hojas de estilo, para eso utilizamos el asterisco (\*) este símbolo es denominado selector universal. Ejemplo de uso: \* { margin: .... }

## Propiedad position o posición

La propiedad de posición especifica el tipo de método de posicionamiento utilizado para un elemento, puede tomar los siguientes valores:

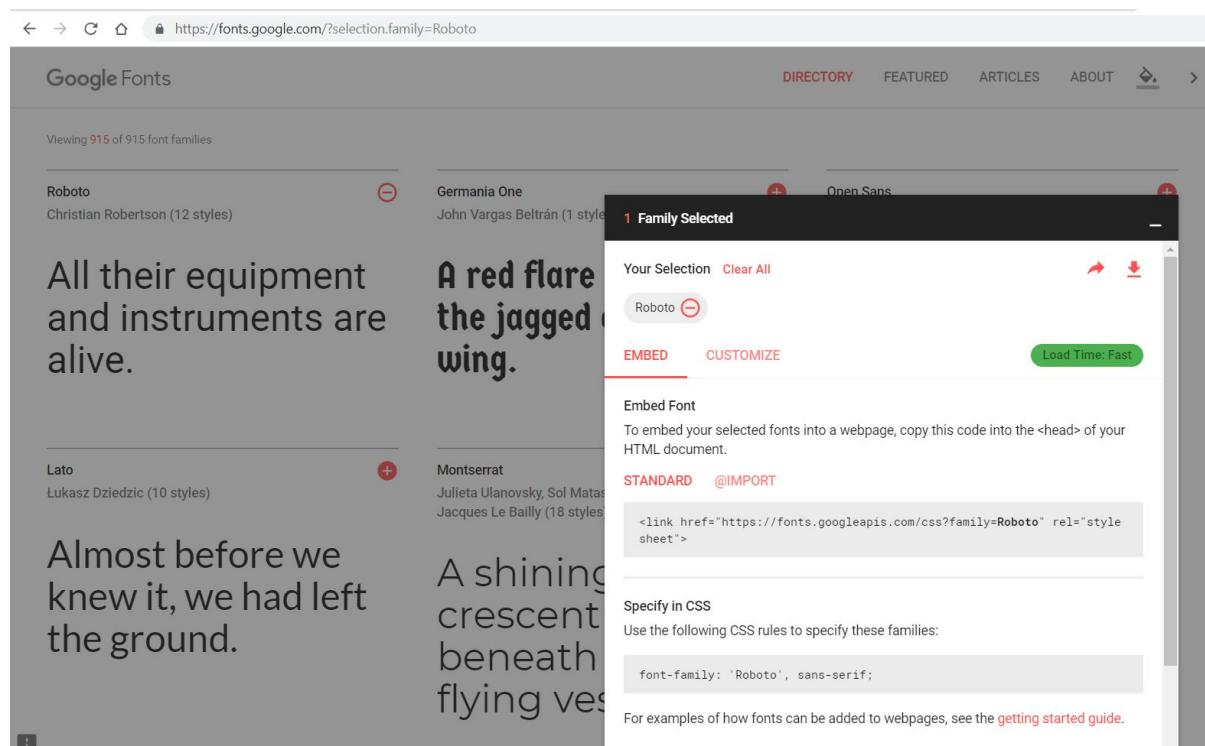
- static
- relative, esta posición es normalmente la más usada.
- fixed
- absolute
- sticky

Ver más información de la propiedad position en la siguiente web:  
[https://www.w3schools.com/css/css\\_positioning.asp](https://www.w3schools.com/css/css_positioning.asp)

## Propiedades de texto

Una forma para cambiar la fuente o font de nuestro sitio es agregando al css la siguiente propiedad: **font-family: nombreDeFuente;**

Si no queremos utilizar las tipografías tradicionales, también podemos hacer uso de otras descargadas del sitio de fuentes de google <https://fonts.google.com/>



The screenshot shows the Google Fonts website interface. On the left, there's a list of font families: Roboto, Lato, and Montserrat. The Roboto section is expanded, showing sample text: "All their equipment and instruments are alive." In the center, another section shows sample text: "A red flare the jagged wing." On the right, a modal window titled "1 Family Selected" shows "Roboto" listed under "Your Selection". Below the modal, there are sections for "Embed Font" (with code provided) and "Specify in CSS" (with a code snippet: "font-family: 'Roboto', sans-serif;").

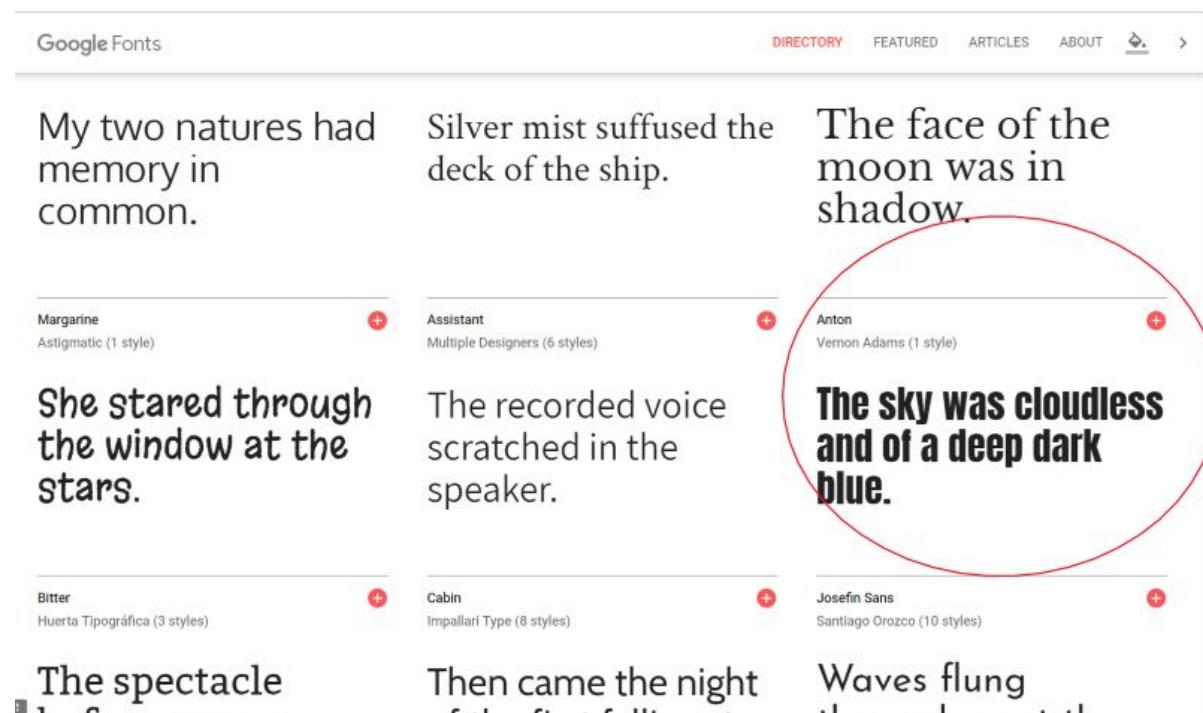
en este caso seleccionamos la fuente roboto.

## ¿Cómo implementamos en nuestro sitio las tipografías de google font?

En el siguiente ejemplo seleccionamos la fuente “Anton” de google fonts, para implementarla en nuestro sitio, debemos pegar el link que nos provee google dentro del `<head>` de nuestro sitio.

Luego dentro de nuestro css utilizaremos las reglas que también nos provee google.

```
font-family: 'Anton', sans-serif;
```



The screenshot shows the Google Fonts website interface. At the top, there's a navigation bar with links for DIRECTORY, FEATURED, ARTICLES, and ABOUT. Below the navigation, there are several font samples displayed in a grid. One sample, "Anton" by Vernon Adams (1 style), is highlighted with a red circle. The text in this sample is bold and black. Other samples shown include Margarine, Assistant, Cabin, and Josefina Sans. The "Anton" sample text is:

My two natures had  
memory in  
common.

Silver mist suffused the  
deck of the ship.

The face of the  
moon was in  
shadow.

**The sky was cloudless  
and of a deep dark  
blue.**

She stared through  
the window at the  
stars.

The recorded voice  
scratched in the  
speaker.

The spectacle

Then came the night

Waves flung

1 Family Selected

—

EMBED CUSTOMIZE Load Time: Fast

---

**Embed Font**

To embed your selected fonts into a webpage, copy this code into the <head> of your HTML document.

STANDARD @IMPORT

```
<link href="https://fonts.googleapis.com/css?family=Anton" rel="stylesheet">
```

---

**Specify in CSS**

Use the following CSS rules to specify these families:

```
font-family: 'Anton', sans-serif;
```

For examples of how fonts can be added to webpages, see the [getting started guide](#).

```
<head>

    <meta charset="utf-8">
    <meta name="author" content="Tutorial css Rollingcode">
    <meta name="description" content="Aprendiendo CSS">
    <meta name="keywords" content="tutorial, css, desarrollo web">

    <title> Propiedades de Texto css</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">

    <link rel="icon" href="img/icono.png">

    <style>
        * {
            margin: 0px;
            padding: 0px;
            list-style: none;
            font-family: 'Anton', sans-serif;
        }

        header {
            position: relative;
            margin: 20px auto;
            width: 1000px;
            height: 120px;
            background: #444;
        }
    </style>

```

**Recomendación:** No se recomienda tener muchas tipografías en un sitio web, las tipografías especiales están más recomendadas para logos, botones o títulos, pero para párrafos es necesario usar tipografías legibles como la sans-serif, arial, etc.

## Otras propiedades de texto

- La propiedad font-size sirve para modificar el tamaño de una fuente. `font-size: 50px;`
- La propiedad text-align cambiar la alineación del texto. `Text-align: center;`
- La propiedad color sirve para modificar el color del texto. `color: blue;`

## Otros recursos

### Texto de prueba “Lorem ipsum es”

Cuando necesitamos agregar texto para ir probando nuestro sitio web, podemos hacer uso del texto que genera la siguiente pagina web: <https://es.lipsum.com/> esta pagina nos sirve para poder crear párrafos, listas etc. con texto de prueba.

## Símbolos especiales

Existen símbolos especiales que en algunas ocasiones vamos a querer agregar en nuestro sitio web y para ello tendremos que utilizar el código HTML que representa a ese símbolo, por ejemplo para el símbolo de copyright © deberemos agregar en nuestra página el texto &copy; y cuando lo visualizamos en un navegador veremos el signo de copyright.

Podemos consultar una tabla completa de símbolos y el respectivo código html de cada uno en la siguiente pagina: <https://ascii.cl/es/codigos-html.htm>

Pueden ver mas información en la siguiente web:

[https://www.w3schools.com/css/css\\_text.asp](https://www.w3schools.com/css/css_text.asp)

## Propiedades de lista

### Posición de los marcadores de una lista de items

Podemos modificar la posición de los marcadores de una lista de items con la propiedad `list-style-position`, esta propiedad tiene dos valores posibles: `inside` y `outside`.

Ejemplo de `list-style-position: outside;`

- Coffee - A brewed drink prepared from roasted coffee beans...
- Tea
- Coca-cola

Ejemplo de `list-style-position: inside;`

- Coffee - A brewed drink prepared from roasted coffee beans...
- Tea
- Coca-cola

---

## Cambiar los marcadores de una lista de items

Podemos especificar que tipo de marcadores queremos usar en una lista, para esto usamos la propiedad **list-style-type**

Veamos el siguiente ejemplo de marcadores de lista:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
    list-style-type: circle;
}

ul.b {
    list-style-type: square;
}

ol.c {
    list-style-type: upper-roman;
}

ol.d {
    list-style-type: lower-alpha;
}
</style>
</head>
<body>

<p>Example of unordered lists:</p>
<ul class="a">
    <li>Coffee</li>
    <li>Tea</li>
    <li>Coca Cola</li>
</ul>

<ul class="b">
    <li>Coffee</li>
    <li>Tea</li>
    <li>Coca Cola</li>
</ul>
```

---

```
<p>Example of ordered lists:</p>
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

</body>
</html>
```

Example of unordered lists:

- Coffee
  - Tea
  - Coca Cola
- 
- Coffee
  - Tea
  - Coca Cola

Example of ordered lists:

- I. Coffee
  - II. Tea
  - III. Coca Cola
- 
- a. Coffee
  - b. Tea
  - c. Coca Cola

También podemos agregar una imagen personalizada, en lugar de los marcadores por defecto, para eso utilizamos `list-style-image`

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

```
<style>
ul {
    list-style-image: url('sqpurple.gif');
}
</style>
</head>
<body>

<ul>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Coca Cola</li>
</ul>

</body>
</html>
```

Si visualizamos el código anterior en un navegador obtendremos lo siguiente:

- Coffee
- Tea
- Coca Cola

Pueden ver mas información en la siguiente web:

[https://www.w3schools.com/css/css\\_list.asp](https://www.w3schools.com/css/css_list.asp)

Propiedades de enlace o link

Con css3 podemos darle diferentes estilos a un link. Por ejemplo:

Text Link Text Link

Link Button

Link Button

Podemos cambiar el estilo de los link, con propiedades que vimos anteriormente como color, background, font-family, pero también podemos modificar su estilo dependiendo del estado que tenga el link.

Los estados que puede tener un link son:

- 
- **a:link** - estado normal de un link que no fue utilizado.
  - **a:visited** - estado de un link que ya fue utilizado.
  - **a:hover** - estado cuando el usuario pasa el mouse sobre el link.
  - **a:active** - estado en el momento que el usuario hace clic en el link.

Ejemplo de uso:

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}
https://www.w3schools.com/css/default.asp
/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>

<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
</body>
</html>
```

Pueden visualizar este ejemplo en la siguiente web:

[https://www.w3schools.com/css/tryit.asp?filename=trycss\\_link](https://www.w3schools.com/css/tryit.asp?filename=trycss_link)

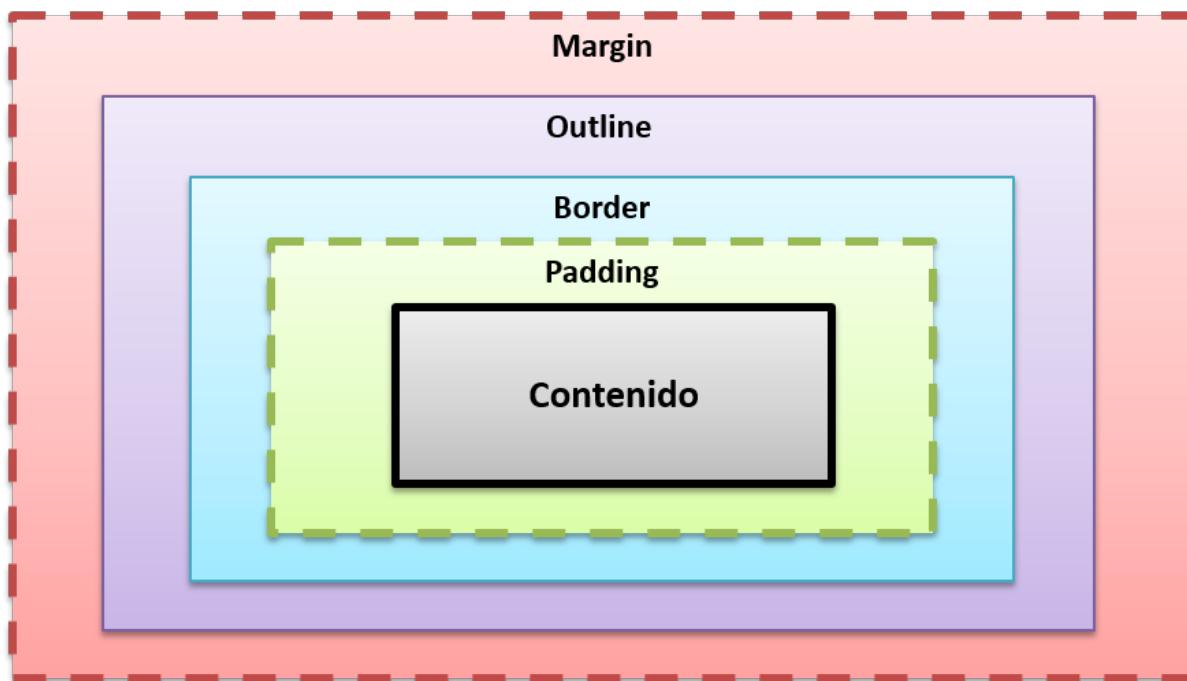
## Box model

### ¿Qué es Box Model?

El Box Model es aprender la forma en el que navegador organiza los elementos dentro de la pantalla. En muchas ocasiones el programador front-end confunde el rol que juega HTML y CSS para representar la información.

Lo primero que debemos de entender, es que cada elemento que definimos en un documento HTML se mostrará en el navegador como una caja, dicha caja tendrá 4 lados y se compondrá de 4 a 5 secciones según el tipo de elemento.

La única sección opcional es *outline*, la cual solo se utiliza para elementos de los formularios, por lo pronto, nos centraremos en las primeras 4 secciones compartidas por todos los elementos y más adelante retomaremos este último.

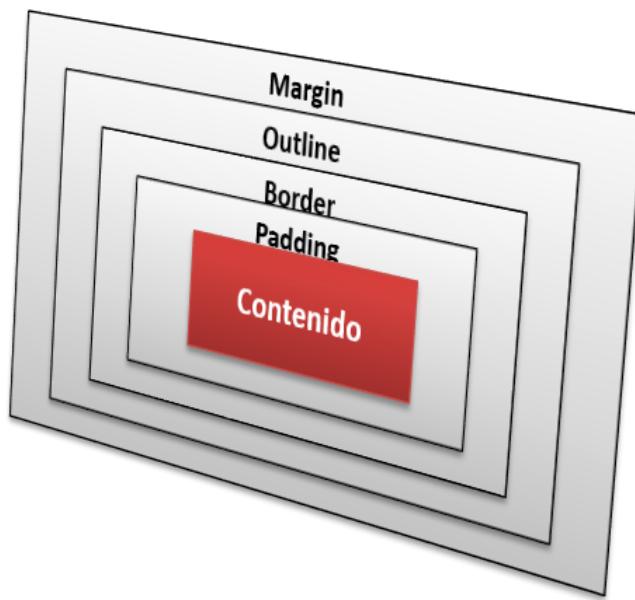


### Contenido

La sección del contenido la principal, contiene el “contenido” central a mostrar, es decir, un texto, una imagen, un video, etc. El contenido siempre es lo que queremos mostrarle al usuario y el resto de secciones solo sirven para decorarlo.

---

Esta sección se ajusta a la altura y ancho del “contenido” a mostrar, porque entre mayor sean los elementos mostrados, más grande será esta área. Esta sección en muchas ocasiones tiene un color o imagen de fondo para hacerla más atractiva.



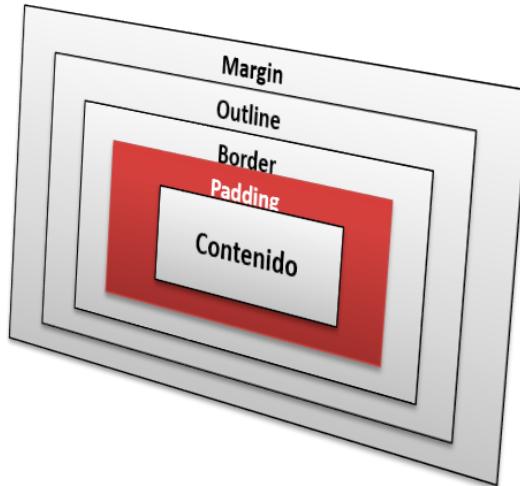
Como podemos observar en la imagen, el contenido es la sección central de todo el elemento, de tal forma, que el siguiente elemento que lo rodea es el padding. El tamaño de esta sección se puede modificar con las propiedades `height`, `width`, `max-height`, `max-width`, `min-height`, `min-width`.



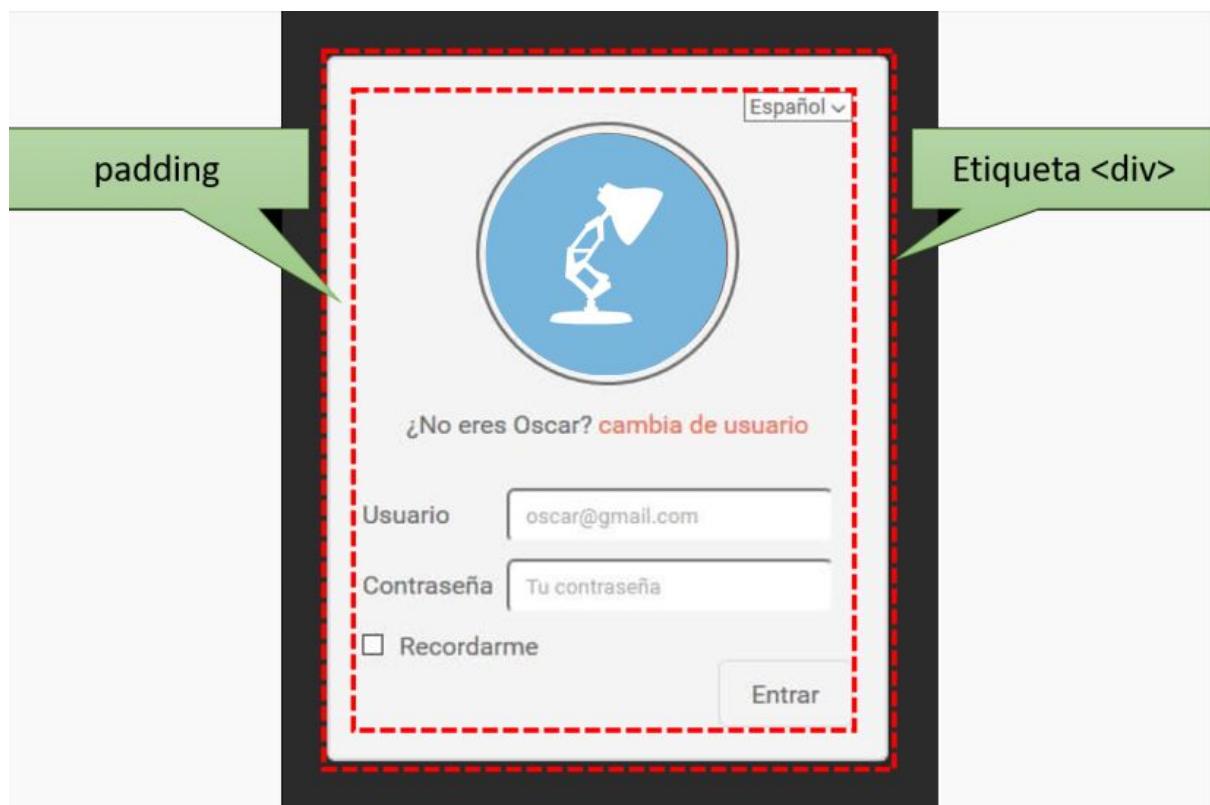
En la imagen podemos apreciar marcado en rojo, una etiqueta `<p>`, esta etiqueta tiene como contenido el texto “¿No eres Oscar? Cambia de usuario”, la etiqueta `<p>` rodea todo el texto, incluso, es más grande que el contenido, pues este lo rodea.

## Padding

El padding es una separación que existe entre el contenido y el borde, el cual se utiliza para dar una apariencia estética más atractiva y que el contenido no esté pegado al borde.



El padding sigue siendo parte de la caja visible, por lo que, si tenemos una imagen o color de fondo, este se extenderá a través del padding o relleno. El padding está delimitado por el borde.



En esta nueva imagen, podemos ver el mismo ejemplo, solo que esta vez nos hemos centrado en el `<div>` que contiene todo el contenido del formulario. Observemos que todos los componentes que tenemos en la pantalla están alineados a la caja punteada interna, esto es porque esto es “contenido” y la separación que existe entre las dos cajas punteadas

---

es conocido como padding o relleno. Observemos que el color de fondo (gris claro) se expande a través del contenido y el padding.

El **padding** puede ser establecido con las propiedades `padding` o con las propiedades más específicas ( `padding-top` , `padding-bottom` , `padding-left` , `padding-right` ).

## Border

El borde es una línea que podemos dibujar al final de la caja, esta se utiliza para darle una apariencia estética a la caja, pues nos permite dibujar una línea de algún color, la línea puede tener los siguientes estilos:

`solid`



`dashed`



`dotted`



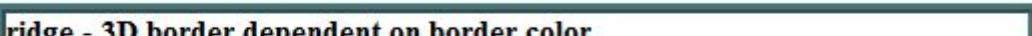
`double`



`groove - 3D border dependent on border color`



`ridge - 3D border dependent on border color`



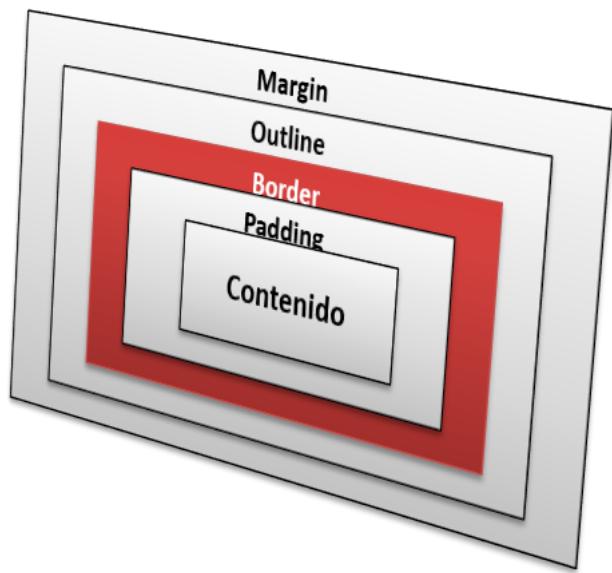
`inset - 3D inset border dependent on border color`



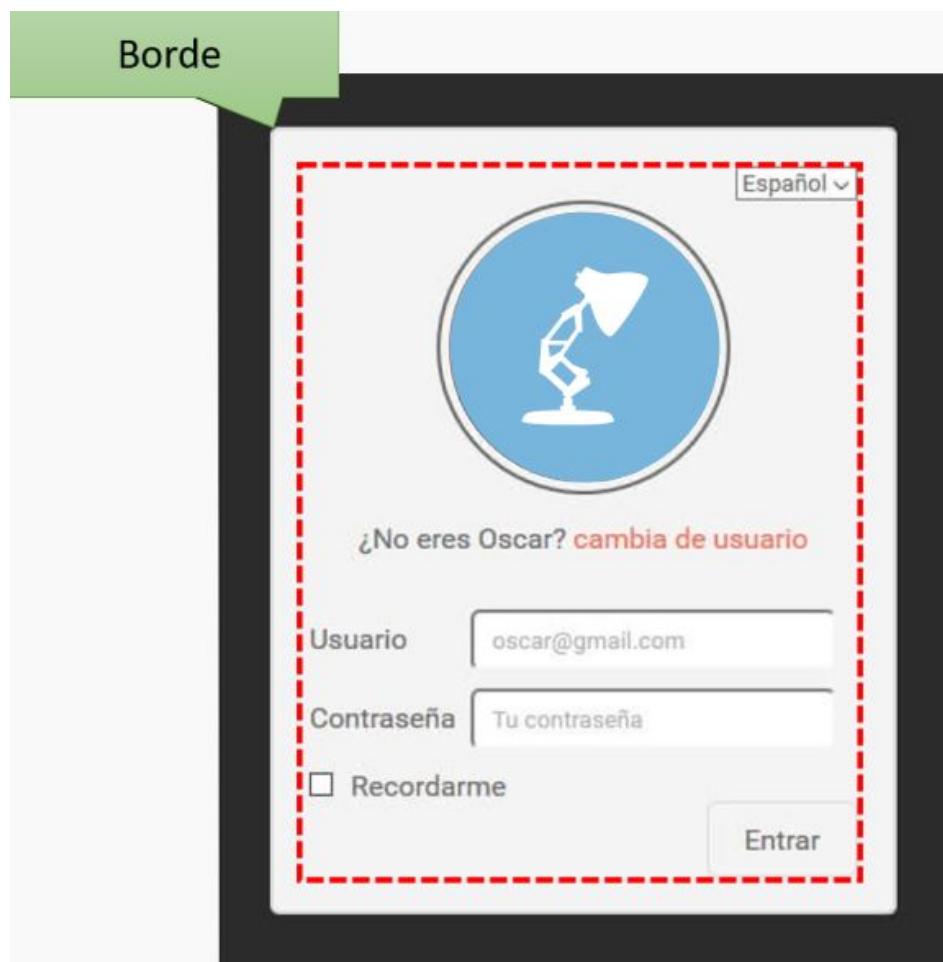
`outset - 3D outset border dependent on border color`



El **border** es la última parte visible de la caja y aunque la sección `outline` si es visible, este se pinta fuera de la caja. El **border** puede ser establecido con las propiedades, `border`, `border-top`, `border-bottom`, `border-left`, `border-right`.



La siguiente imagen muestra un ejemplo del border:

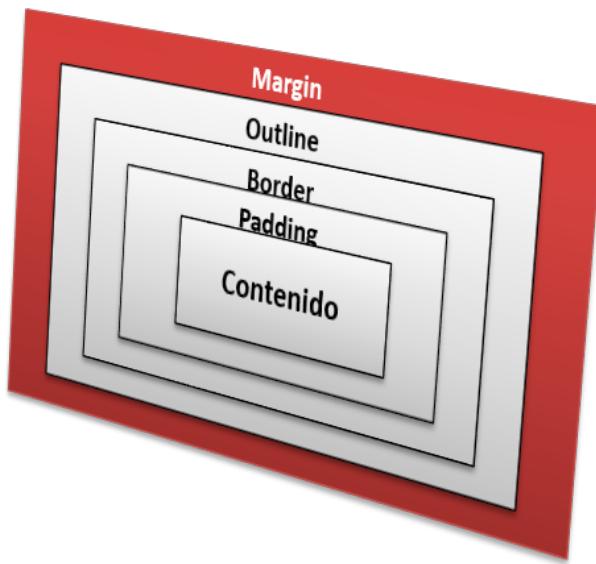


---

En la imagen podemos apreciar un borde muy suave, apenas perceptible, pero que sin duda le da una terminación más elegante, estamos utilizando la propiedad `border-radius` para darle un efecto redondeado en las esquinas.

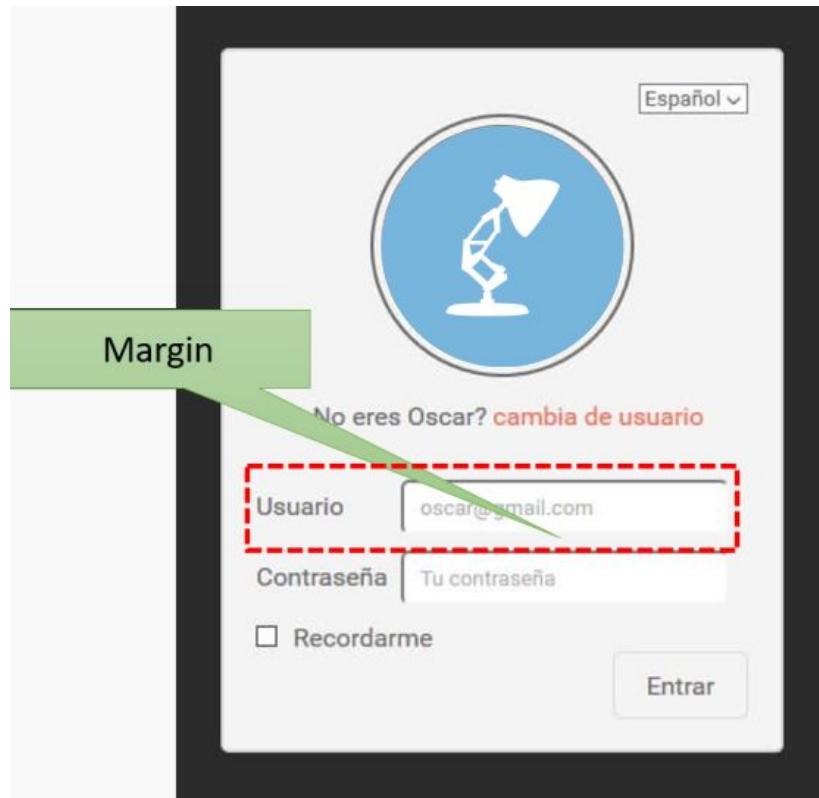
## Margin

El margen es la última sección de Box Model y se puede ver como una separación invisible que ayuda a separar un elemento de otro. Cuando definimos un color o imagen de fondo, este no se propaga a esta sección.



El margin se puede definir con las propiedades `margin`, `margin-top`, `margin-bottom`, `margin-right`, `margin-left`.

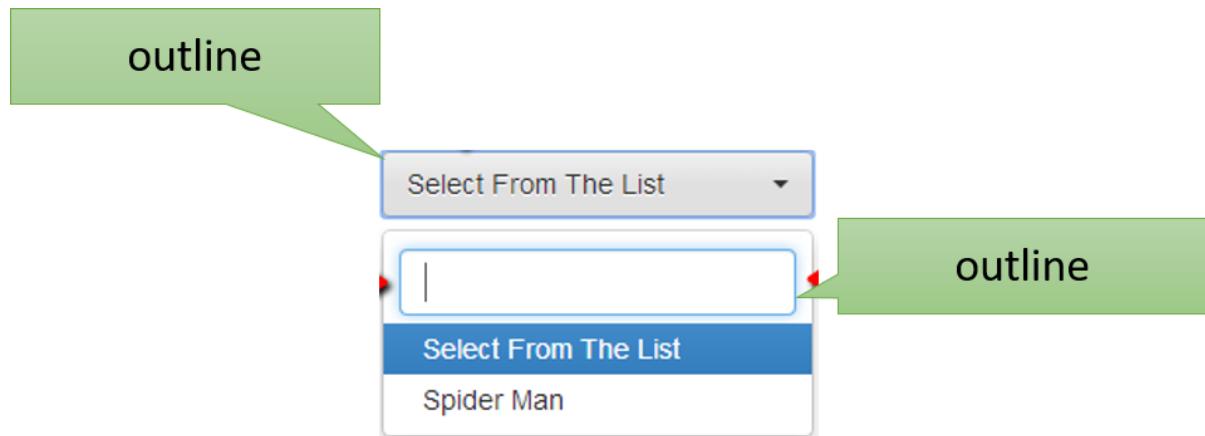
La siguiente imagen muestra mejor cómo es que utilizamos el margin:



En este caso, utilizamos un margin inferior (`margin-bottom`) para separar el primer campo del segundo, esta separación hace que las dos cajas no choquen y se vean más estéticas.

## Outline

El outline es una decoración gráfica que hace que los elementos que ganan el foco, que son por lo general los `<input>`, se colorean como con una ligera aura que los rodea:



---

El outline se establece mediante la propiedad outline y es posible establecer el grueso del outline, el color y el estilo.

Pueden obtener más información de Box Model, en la siguiente web:  
[https://developer.mozilla.org/es/docs/Learn/CSS/Introduction\\_to\\_CSS/Modelo\\_cajas](https://developer.mozilla.org/es/docs/Learn/CSS/Introduction_to_CSS/Modelo_cajas)

## Flexbox

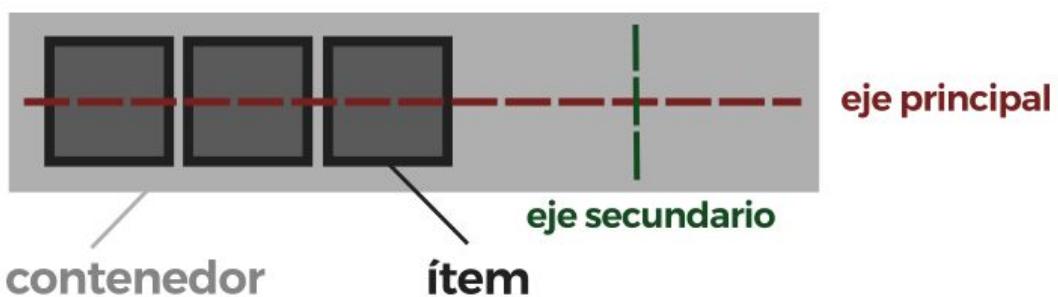
Tradicionalmente, en CSS se ha utilizado el posicionamiento (static, relative, absolute), los elementos en línea o en bloque (y derivados) o los float, lo que a grandes rasgos no dejaba de ser un sistema de creación de diseños bastante tradicional que no encaja con los retos que tenemos hoy en día: sistemas de escritorio, dispositivos móviles, múltiples resoluciones, etc.

Flexbox es un sistema de elementos flexibles que llega con la idea de olvidar estos mecanismos y acostumbrarnos a una mecánica más potente, limpia y personalizable, en la que los elementos HTML se adaptan y colocan automáticamente y es más fácil personalizar los diseños. Está especialmente diseñado para crear, mediante CSS, estructuras de una sola dimensión.

El uso de Flexbox está recomendado por la WC3 a partir de octubre de 2017.

### Conceptos

Para empezar a utilizar flexbox lo primero que debemos hacer es conocer algunos de los elementos básicos de este nuevo esquema, que son los siguientes:



- **Contenedor**: Existe un elemento padre que es el contenedor que tendrá en su interior cada uno de los ítems flexibles y adaptables.

- 
- **Eje principal:** Los contenedores flexibles tendrán una orientación principal específica. Por defecto es horizontal (fila).
  - **Eje secundario:** De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es horizontal, la secundaria será en vertical, y viceversa.
  - **Ítem:** Cada uno de los hijos flexibles que tendrá el contenedor en su interior.

## Los dos ejes de flexbox

Cuando trabajamos con flexbox necesitamos pensar en términos de dos ejes — el eje principal y el eje secundario. El eje principal está definido por la propiedad **flex-direction**, y el eje secundario es perpendicular a este. Todo lo que hacemos con flexbox está referido a estos dos ejes, por lo que vale la pena entender cómo trabajan desde el principio.

### El eje principal

El eje principal está definido por **flex-direction**, que posee cuatro posibles valores:

- row
- row-reverse
- column
- column-reverse

Si elegimos **row** o **row-reverse**, el eje principal correrá a lo largo de la fila según la dirección de la línea .



Al elegir **column** o **column-reverse** el eje principal correrá desde el borde superior de la página hasta el final — según la **dirección del bloque**.

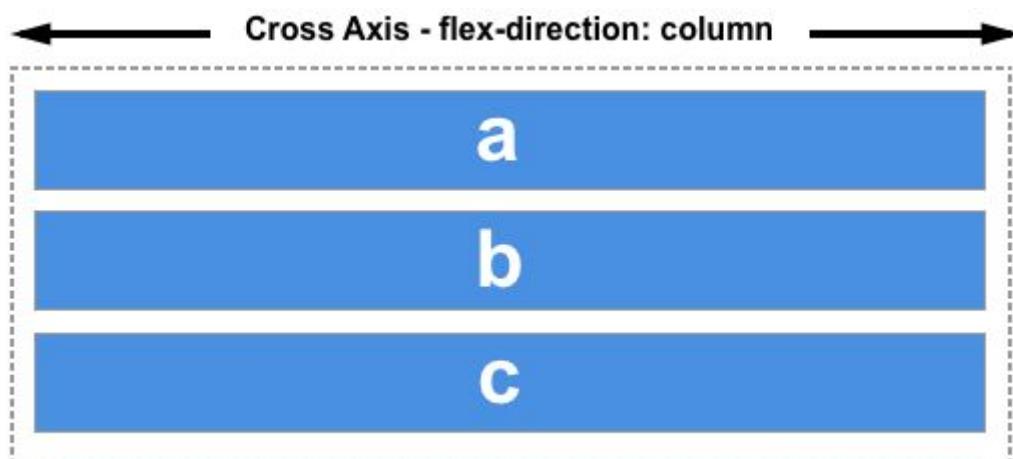


El eje secundario

El eje secundario va perpendicular al eje principal, y por lo tanto si `flex-direction` (del eje principal) es `row` o `row-reverse` el eje secundario irá por las columnas.



Si el eje principal es `column` o `column-reverse` entonces el eje secundario corre a lo largo de las filas.



Entender cuál eje es cuál es importante cuando empezamos a mirar la alineación y justificación flexible de los ítems; flexbox posee propiedades que permiten alinear y justificar el contenido sobre un eje o el otro.

### Líneas de inicio y de fin

En el pasado, CSS estaba muy inclinado hacia el modo de escritura horizontal y de izquierda a derecha. Los métodos modernos de layout acogen la totalidad de modos de escritura así que no es necesario asumir que una línea de texto empezará arriba del documento y correrá de izquierda a derecha, con nuevas líneas dispuestas una debajo de la otra.

Puede leer más acerca de la relación que hay entre flexbox y la especificación de los “Modos de Escritura”, sin embargo la siguiente descripción debería ayudar para explicar por qué no se habla de izquierda y derecha ni de arriba o abajo a la hora de describir la dirección en la que fluyen los ítems flex.

Si flex-direction es `row` y estoy trabajando en español, entonces el margen inicial del eje principal quedará a la izquierda, y el margen final a la derecha.



Si fuera a trabajar en árabe, entonces el margen inicial de mi eje principal quedaría a la derecha y el margen final a la izquierda.



---

En ambos casos el margen inicial del eje secundario estará en el extremo superior del contenedor flex y el margen final en el extremo inferior, ya que ambos idiomas tiene un modo de escritura horizontal.

Después de un tiempo, pensar en inicial y final en vez de izquierda y derecha se hará natural, y será útil cuando interactúe con otros métodos de layout tales como el CSS Grid Layout que sigue los mismos patrones.

### El contenedor flex

Un área del documento que contiene un flexbox es llamada contenedor flex. Para crear un contenedor flex, establecemos la propiedad del área del contenedor `display` como `flex` o `inline-flex`. Tan pronto como hacemos esto, los hijos directos de este contenedor se vuelven ítems flex. Como con todas las propiedades de CSS, se definen algunos valores iniciales, por lo que cuando creamos un contenedor flex todos los ítems flex contenidos se comportan de la siguiente manera.

- Los ítems se despliegan sobre una fila (la propiedad `flex-direction` por defecto es `row`).
- Los ítems empiezan desde el margen inicial sobre el eje principal.
- Los ítems no se ajustan en la dimensión principal, pero se pueden contraer.
- Los ítems se ajustarán para llenar el tamaño del eje cruzado.
- La propiedad `flex-basis` es definida como `auto`.
- La propiedad `flex-wrap` es definida como `nowrap`.

El resultado es que todos los ítems se alinearán en una sola fila, usando el tamaño del contenedor como su tamaño en el eje principal. Si hay más ítems de los que caben en el contenedor, estos no pasarán más abajo si no que sobrepasan el margen. Si hay ítems más altos que otros, todos los ítems serán ajustados en el eje secundario para alcanzar al mayor.

Por ejemplo:



```
.box {  
  display: flex;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three  
    <br>has  
    <br>extra  
    <br>text  
  </div>  
</div>
```

## Cambiar flex-direction

Al añadir la propiedad `flex-direction` en el contenedor flex nos permite cambiar la dirección de cómo los ítems son desplegados. Colocando `flex-direction: row-reverse` se mantendrá el despliegue a lo largo de la fila, sin embargo el inicio y final quedarán al revés del original.

Si cambiamos `flex-direction` a `column` el eje principal se cambiará y los ítems aparecerán en una columna. Colocando `column-reverse` las líneas de inicio y fin serán nuevamente puestas al revés.

Ejemplo:



The screenshot shows a code editor with two sections. The top section contains a visual representation of a flexbox container with three items: 'Three' (blue), 'Two' (orange), and 'One' (green). The bottom section contains the corresponding CSS and HTML code:

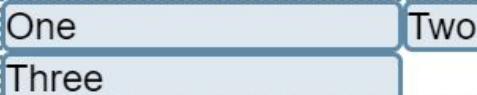
```
.box {  
  display: flex;  
  flex-direction: row-reverse;  
}  
  
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

## Contenedores flex Multi-línea con flex-wrap

Si bien flexbox es un modelo unidimensional, es posible lograr que nuestros ítems flex sean repartidos en varias líneas. Haciendo esto, se deberá considerar cada línea como un nuevo contenedor flex. Cualquier distribución del espacio solo sucederá dentro de esa línea, sin referenciar las líneas colaterales.

Para lograr repartirse en varias líneas añada la propiedad `flex-wrap` con el valor `wrap`. Cuando los ítems sean demasiados para desplegarlos en una línea, serán repartidos en la línea siguiente.

El siguiente ejemplo contiene ítems que se les ha asignado un ancho, donde el ancho total de los ítems excede al del contenedor flex. Cuando `flex-wrap` se coloca como `wrap`, los ítems se repartirán. Al colocarlo como `nowrap`, el cual es el valor inicial, estos se contraerán para calzar con el contenedor ya que usan los valores iniciales de flexbox que permiten que los ítems se contraigan. Al usar `nowrap` los ítems podrían salirse del margen si estos no pudieran contraerse, o no contraerse lo suficiente para ser calzados.



```
.box {  
    display: flex;  
    flex-wrap: wrap;  
}
```

```
<div class="box">  
    <div>One</div>  
    <div>Two</div>  
    <div>Three</div>  
</div>
```

## La abreviatura flex-flow

Se pueden combinar las propiedades `flex-direction` y `flex-wrap` en la abreviatura `flex-flow`. El primer valor especificado es `flex-direction` y el segundo valor es `flex-wrap`.

ejemplo:



```
.box {  
    display: flex;  
    flex-flow: row wrap;  
}
```

```
<div class="box">  
    <div>One</div>  
    <div>Two</div>  
    <div>Three</div>  
</div>
```

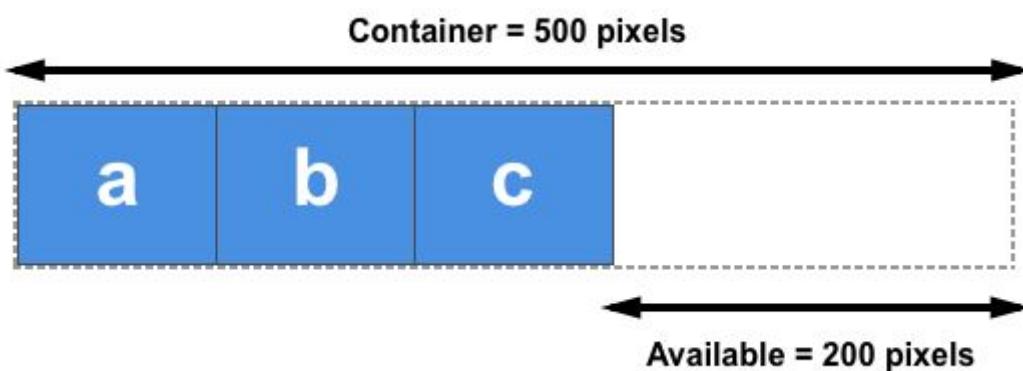
## Propiedades aplicadas a los ítems flex

Para obtener más control sobre los ítems flex podemos apuntarlos directamente. Hacemos esto a través de tres propiedades:

- `flex-grow`
- `flex-shrink`
- `flex-basis`

Antes de darle sentido a estas propiedades debemos considerar el concepto de espacio disponible. Lo que hacemos cuando cambiamos el valor de alguna de estas propiedades es cambiar la forma que se distribuye el espacio disponible entre nuestros ítems. Este concepto de espacio disponible es también importante cuando veamos la alineación de ítems.

Si tenemos tres ítems con un ancho de 100 píxeles en un contenedor de 500 píxeles de ancho, entonces el espacio que se necesita para colocar nuestros ítems es de 300 píxeles. Esto deja 200 píxeles de espacio disponible. Si no cambiamos los valores iniciales entonces flexbox colocará ese espacio después del último ítem.



Si en cambio quisiéramos que los ítems crecieran para llenar ese espacio, entonces necesitaremos un método para distribuir el espacio sobrante entre los ítems. Es justo lo que harán las propiedades flex que aplicaremos a dichos ítems.

### La propiedad `flex-basis`

Con `flex-basis` se define el tamaño de un ítem en términos del espacio que deja como espacio disponible. El valor inicial de esta propiedad es `auto` — en este caso el navegador revisa si los ítems definen un tamaño. En el ejemplo de arriba, todos los ítems tienen un ancho de 100 píxeles así que este es usado como `flex-basis`.

---

Si los ítems no tiene un tamaño entonces el tamaño de su contenido es usado como **flex-basis**. Y por eso, apenas declarado `display: flex` en el padre a fin de crear ítems flex, todos estos ítems se ubican en una sola fila y tomaron sólo el espacio necesario para desplegar su contenido.

### La propiedad flex-grow

Con la propiedad **flex-grow** definida como un entero positivo, los ítems flex pueden crecer en el eje principal a partir de **flex-basis**. Esto hará que el ítem se ajuste y tome todo el espacio disponible del eje, o una proporción del espacio disponible si otro ítem también puede crecer.

Si le damos a todos los ítems del ejemplo anterior un valor **flex-grow** de 1 entonces el espacio disponible en el contenedor flex será compartido igualitariamente entre estos ítems y se ajustarán para llenar el contenedor sobre el eje principal.

Podemos usar **flex-grow** apropiadamente para distribuir el espacio en proporciones. Si otorgamos al primer ítem un valor **flex-grow** de 2 y a los otros un valor de 1, entonces 2 partes serán dadas al primer ítem (100px de 200px en el caso del ejemplo de arriba) y 1 parte para cada uno de los restantes (cada uno con 50px de los 200px en total).

### La propiedad flex-shrink

Así como la propiedad **flex-grow** se encarga de añadir espacio sobre el eje principal, la propiedad **flex-shrink** controla cómo se contrae. Si no contamos con suficiente espacio en el contenedor para colocar los ítems y **flex-shrink** posee un valor entero positivo, el ítem puede contraerse a partir de **flex-basis**. Así como podemos asignar diferentes valores de **flex-grow** con el fin que un ítem se expanda más rápido que otros — un ítem con un valor más alto de **flex-shrink** se contraerá más rápido que sus hermanos que poseen valores menores.

El tamaño mínimo del ítem tendrá que ser considerado cuando se determine un valor de contracción que pueda funcionar, esto significa que **flex-shrink** tiene el potencial de comportarse menos consistentemente que **flex-grow**.

### Valores abreviados para las propiedades flex

Difícilmente veremos la propiedades `flex-grow`, `flex-shrink` y `flex-basis` usadas individualmente; si no que han sido combinadas en la abreviación `flex`. La abreviación `flex` permite establecer los tres valores en este orden: `flex-grow`, `flex-shrink`, `flex-basis`.

Veamos el ejemplo siguiente, recuerde que el primer valor es `flex-grow`. Dándole un valor positivo significa que el ítem puede crecer. El segundo es `flex-shrink` — con un valor positivo los ítems pueden contraerse. El valor final es `flex-basis`; este es el valor que los ítems usan como valor base para crecer y contraerse.



One      Two      Three

```
.box {  
  display: flex;  
}  
  
.one {  
  flex: 1 1 auto;  
}  
  
.two {  
  flex: 1 1 auto;  
}  
  
.three {  
  flex: 1 1 auto;  
}  
  
<div class="box">  
  <div class="one">One</div>  
  <div class="two">Two</div>  
  <div class="three">Three</div>  
</div>
```

Hay además algunas abreviaturas de valores que cubren la mayoría de los casos de uso. Se ven con frecuencia utilizados en tutoriales, y en muchos casos es todo lo que necesitamos usar. Los valores predefinidos son los siguientes:

- `flex: initial`
- `flex: auto`
- `flex: none`
- `flex: <positive-number>`

---

Fijando `flex: initial` el ítem se restablece con los valores iniciales de Flexbox. Es lo mismo que `flex: 0 1 auto`. En este caso el valor de `flex-grow` es 0, así que los ítems no crecerán más de su tamaño `flex-basis`. El valor `flex-shrink` es 1, así que los ítems pueden contraerse si es necesario en vez de salirse de los márgenes. El valor de `flex-basis` es auto. Los ítems pueden definir un tamaño en la dimensión del eje principal, o bien obtener su tamaño por el contenido del los mismos.

Usar `flex: auto` es lo mismo que usar `flex: 1 1 auto`, es como con `flex:initial` pero en este caso los ítems pueden crecer y llenar el contendor así como encoger si se requiere.

Al usar `flex: none` se crearán ítems flex totalmente inflexibles. Es como escribir `flex: 0 0 auto`. Los ítems no pueden ni crecer ni encoger pero serán colocados usando flexbox con `flex-basis` en auto.

Una abreviación que es común en tutoriales es `flex: 1` o `flex: 2` y más. Es como usar `flex: 1 1 0`. Los ítems pueden crecer o encoger con un `flex-basis` de 0.

## Alineación, justificación y distribución del espacio libre entre ítems

Una característica clave de flexbox es la capacidad de alinear y justificar ítems sobre los ejes principal y secundario, y distribuir el espacio entre los ítems flex.

### Align-items

La propiedad `align-items` alineará los ítems sobre el eje secundario.

El valor inicial para esta propiedad es `stretch` razón por la cual los ítems se ajustan por defecto a la altura de aquel más alto. En efecto se ajustan para llenar el contenedor flex — el ítem más alto define la altura de este.

En cambio definimos `align-items` como `flex-start` para que los ítems se alineen al comienzo del contenedor flex, `flex-end` para alinearlos al final, o `center` para alinearlos al centro.

### justify-content

La propiedad `justify-content` es usada para alinear los ítems en el eje principal, cuyo `flex-direction` define la dirección del flujo. El valor inicial es `flex-start` que alineará los

---

ítems al inicio del margen del contenedor, pero también se podría definir como `flex-end` para alinearlos al final, o `center` para alinearlos al centro.

También podemos usar `space-between` para tomar todo el espacio sobrante después de que los ítems hayan sido colocados, y distribuir de forma pareja los ítems para que haya un espacio equitativo entre cada ítem. O bien, usamos el valor `space-around` para crear un espacio equitativo a la derecha e izquierda de cada ítem.

Pueden ver más información y ejemplos de flexbox en la siguiente web:  
[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)

## Responsive

### ¿Qué es el diseño web responsive?

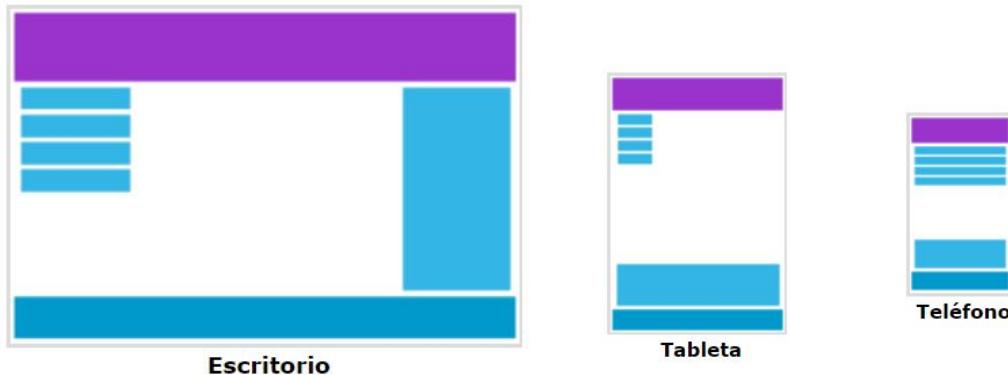
El diseño web responsive o adaptativo es una técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos. Desde ordenadores de escritorio a tablets y móviles.

Hoy en día accedemos a sitios web desde todo tipo de dispositivos; ordenador, tablet, smartphone... por lo que, cada vez más, nos surge la necesidad de que nuestra web se adapte a los diferentes tamaños de los mismos.

### ¿En qué consiste el diseño responsive?

Se trata de redimensionar y colocar los elementos de la web de forma que se adapten al ancho de cada dispositivo permitiendo una correcta visualización y una mejor experiencia de usuario. Se caracteriza porque los layouts (contenidos) e imágenes son fluidos y se usa código media-queries de CSS3.

El diseño responsive permite reducir el tiempo de desarrollo, evita los contenidos duplicados, y aumenta la viralidad de los contenidos ya que permite compartirlos de una forma mucho más rápida y natural.



Se basa en proporcionar a todos los usuarios de una web los mismos contenidos y una experiencia de usuario lo más similar posible, frente a otras aproximaciones al desarrollo web móvil como la creación de apps, el cambio de dominio o webs servidas dinámicamente en función del dispositivo.

### Etiqueta Meta Viewport

El viewport es la zona visible por el usuario de una página web. Varía con el dispositivo, y será más pequeño en un teléfono móvil que en una pantalla de ordenador.

Antes de las tabletas y los teléfonos móviles, las páginas web fueron diseñadas sólo para las pantallas del ordenador, y era común que las web tengan un diseño estático y un tamaño fijo.

Entonces, cuando comenzamos a navegar por internet usando las tabletas y los teléfonos móviles, las páginas web de tamaño fijo eran demasiado grandes. Para solucionar este problema, los navegadores de los dispositivos reducen la escala de toda la página web para ajustarse a la pantalla.

### Estableciendo el Viewport

HTML5 introdujo un método para que los diseñadores web tengan control sobre el área de visualización, a través de la etiqueta `<meta>`.

Debe incluir los siguientes `<meta>` en todas sus páginas web:

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0">
```

Un `<meta>` viewport da las instrucciones a su navegador sobre cómo controlar las dimensiones y la escala de la página.

`width=device-width` establece el ancho de la página (que variará en función del dispositivo).

La `initial-scale=1.0` fija el nivel de zoom inicial cuando la página se carga por primera vez por el navegador.

Este es un ejemplo de una página web sin la etiqueta meta de visualización, y la misma página web con la etiqueta meta de visualización:



Sin la etiqueta meta de visualización



Con la etiqueta meta de visualización

Tamaño del contenido para el Viewport

A continuación veremos algunos consejos para tener en cuenta cuando diseñamos un sitio web:

- Los usuarios utilizan el scroll para desplazarse verticalmente por los sitios web tanto en dispositivos de escritorio y móviles - pero no horizontalmente.

---

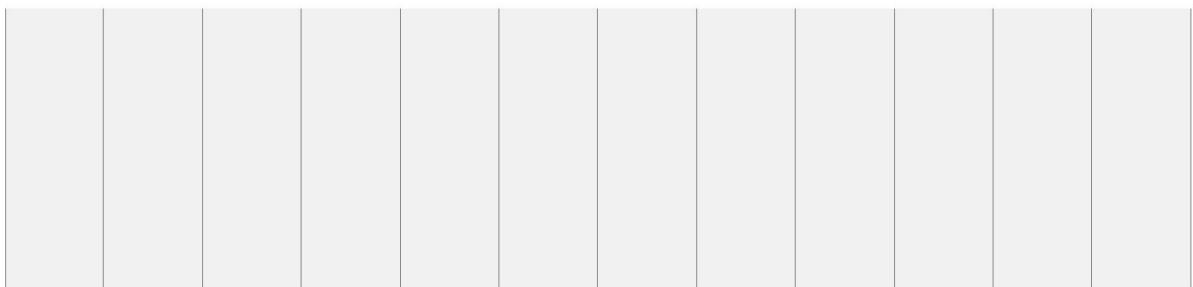
Por lo tanto, si el usuario se ve obligado a desplazarse horizontalmente, o alejar la imagen, para ver toda la página web, esto sería en una experiencia de usuario pobre.

Algunas reglas adicionales que siguen:

1. No utilizar elementos grandes de ancho fijo - Por ejemplo, si una imagen se visualiza en una anchura mayor que el área de visualización puede causar el desplazamiento horizontal. Recuerde que debe ajustar este contenido para que quepa en el ancho del viewport.
2. No deje que el contenido se base en un viewport particular, para rendir bien - Las dimensiones de la pantalla y el ancho en píxeles CSS varían ampliamente entre los dispositivos, el contenido no debe depender de un ancho particular.
3. Usar media queries para aplicar un estilo diferente en pantallas pequeñas y grandes - Considerar el uso de anchos relativos, como valor width: 100%. Además, tengan cuidado con el uso de valores de posición absolutos grandes. Puede hacer que el elemento caiga fuera del viewport en dispositivos pequeños.

## Grid-view

Muchas páginas web están basados en un sistema de cuadrículas, lo que significa que la página se divide en columnas:



El uso de grid-view es muy útil en el diseño de páginas web. Esto hace que sea más fácil colocar los elementos en la página.



Una grid-view responsive tiene 12 columnas y un ancho total de 100%, esta se encoge y expande a medida que cambia el tamaño de la ventana del navegador.

## Media Query

Media Query es una técnica introducida en CSS3, utiliza la regla @media para incluir un bloque de propiedades CSS sólo si una determinada condición es verdadera.

Por ejemplo:

Si la ventana del navegador es 600px o más pequeño, el color de fondo será azul claro.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
body {
    background-color: lightgreen;
}

@media only screen and (max-width: 600px) {
    body {
        background-color: lightblue;
    }
}
</style>
</head>
```

```
<body>
```

```
<p>Resize the browser window. When the width of this document is 600 pixels or less, the background-color is "lightblue", otherwise it is "lightgreen".</p>
```

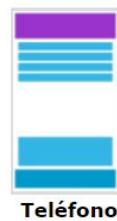
```
</body>
```

```
</html>
```

Resize the browser window. When the width of this document is 600 pixels or less, the background-color is "lightblue", otherwise it is "lightgreen".

Agregar un punto de interrupción o breakpoint

Podemos añadir un punto de interrupción en ciertas partes del diseño para establecer cómo se comportará este diseño en cada lado del punto de ruptura.



Podemos visualizar mejor este concepto con el siguiente ejemplo:

[https://www.w3schools.com/css/tryit.asp?filename=tryresponsive\\_breakpoints](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_breakpoints)

---

Siempre diseñar siguiendo el concepto de Mobile First

Mobile First significa diseñar primero para móviles antes de diseñar para el escritorio o cualquier otro dispositivo (Esto hará que la visualización de la página sea más rápida en los dispositivos más pequeños).

### Breakpoint típicos

Existen muchas pantallas y dispositivos con diferentes altos y anchos, por lo que es difícil de crear un punto de interrupción exacto para cada dispositivo. Para simplificar esta tarea, podemos orientarnos con los siguientes breakpoints sugeridos:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

podemos ver un ejemplo de uso de estos breakpoints en la siguiente web  
[https://www.w3schools.com/css/tryit.asp?filename=tryresponsive\\_mediaquery\\_breakpoints](https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery_breakpoints)

### Frameworks (bootstrap 4, foundation, material)

¿Qué es un framework web y qué ventajas aportan?

---

A la hora de empezar un nuevo proyecto de diseño web, siempre hay que realizar varias tareas de organización y estructuración. Las tecnologías web que se van a usar, el estilo de los diferentes elementos de la web, metodologías, tiempos de entrega, tipografías y colores... éstas son sólo algunas de las cosas que organizar en el equipo de trabajo, para realizarlas contamos con varias herramientas como son los frameworks web, los cuales nos facilitarán muchas de estas tareas y nos ayudarán a desarrollar un proyecto web de calidad.

Vamos a ver qué es un framework web y algunas de las ventajas que nos aportan según nuestras necesidades.

### ¿Qué es un framework web?

Los frameworks web son un conjunto de herramientas, estilos y librerías dispuestas a través de una estructura o esqueleto base, para el desarrollo de aplicaciones web más escalables y sencillas de mantener.

Gracias a estos frameworks web, podemos ahorrar grandes cantidades de tiempo y costes, pero vamos a profundizar más en las ventajas que tienen, causantes de su gran éxito y expansión.

### **Ventajas de los frameworks web**

#### 1 – Documentación y comunidad.

La cantidad de documentación que podremos encontrar sobre un framework, suele ser enorme y además con una gran comunidad detrás, respondiendo preguntas y desarrollando nuevas funcionalidades.

#### 2 – Reutilización del código.

Uno de los puntos fuertes de los frameworks es la modularidad de su código y la capacidad para poder hacer múltiples proyectos con el mismo código, cambiando simplemente los textos.

#### 3 – Arquitectura y metodología.

La mayoría de frameworks del mercado usan arquitecturas y metodologías actuales, como el Modelo-Vista-Controlador (MVC).

#### 4 – Plantillas web.

Las plantillas facilitan mucho el trabajo de los desarrolladores web y los frameworks no se quedan atrás en esto. Algunos frameworks Frontend como Bootstrap cuentan con grandes cantidades de plantillas y componentes desarrollados por su extensa comunidad.

## 5 – Seguridad web.

Los frameworks suelen contar con medidas de seguridad para proteger nuestros datos y los de nuestros clientes, ayudando en gran medida en uno de los temas que lleva de cabeza a grandes empresas de servicios web.

## 6 – Posicionamiento en motores de búsqueda.

El posicionamiento web SEO on page es muy importante si queremos lograr aparecer en las primeras posiciones de buscadores como Google. Por eso muchos frameworks web ya implementan en su estructura código para poder lograrlo más fácilmente.

## Framework front-end

En el mercado existen numerosos frameworks utilizados para facilitar la tarea de frontend, entre los más utilizados actualmente tenemos los siguientes:

- Bootstrap - web: <https://getbootstrap.com/>
- Material Design - web: <https://material.io/design/>
- Foundation- web: <https://foundation.zurb.com/>

Cada uno de estos frameworks funciona de una forma bastante similar, en este curso utilizaremos el framework bootstrap ya que es el más utilizado en el mercado actual.

## Bootstrap 4

### Introducción a Bootstrap

### Historia



Bootstrap fue creado originalmente en el año 2011 como parte del proyecto Twitter, con la idea de mantener una consistencia de los elementos del frontend. Al no tener una convención sobre las formas en la que se desarrollaba la plataforma de Twitter y ser tantas personas involucradas en ese proyecto, cada una con una forma distinta de abordar los problemas, esto generaba inconsistencias inevitables, además que el costo de mantener el desarrollo era muy grande. Entonces los ingenieros de Twitter Mark Otto y Jacob Thornton proponen utilizar un

---

framework que habían desarrollado anteriormente (bootstrap) para que todo el equipo siga una guia e intenten minimizar las inconsistencias. Esta iniciativa triunfo en Twitter, llevando a que el equipo completo trabajara más rápido, de forma más eficaz y con menos inconsistencias.

Aunque comenzó como una solución interna en Twitter, pronto se dieron cuenta del potencial de la herramienta y en Agosto de 2011, el framework “Bootstrap” fue lanzado al público como proyecto Open Source en Github. En los meses posteriores, miles de desarrolladores de todo el mundo contribuyeron al proyecto y Bootstrap se convirtió en el proyecto Open Source más activo del mundo, convirtiéndose en “el framework de presentación más popular para desarrollar proyectos responsive y para móviles”.

Desde su lanzamiento fueron creando distintas versiones, actualmente la última versión estable es la 4.3.

### **¿Qué es bootstrap?**

Es un conjunto de herramientas que nos sirve para realizar el desarrollo frontend. No es solo una hoja de estilo css, sino que consta de un conjunto de componentes como html, css y javascript, lo cual lo convierte en un kit completo de desarrollo frontend.

Se puede incluir en distintos proyectos sin importar el lenguaje con el que se decida realizar el backend. por ejemplo se puede utilizar en proyectos con .net de microsoft, angular o react.

Nota: Hasta la versión 3 de bootstrap era considerado un framework, desde la versión 4 se lo considera un toolkit.

### **¿Para qué sirve?**

Sirve para desarrollar aplicaciones responsivas y mobile-first mucho más fácil, rápida y estandarizada.

### **¿De qué hablamos cuando nos referimos a mobile-first?**

Es una estrategia en la cual el código se diseña primero para dispositivos pequeños como los móviles y luego para dispositivos más grandes.

En versiones anteriores de bootstrap se iba adaptando desde un tamaño grande a uno pequeño, es decir, desde la pantalla de una computadora, a una tablet y luego a la pantalla de un celular, esto cambió con el concepto de mobile-first.

## Integrando Bootstrap a un proyecto

Toda la documentación bootstrap, pasos de instalación, como funcionan sus componentes, ejemplos etc, pueden encontrarla en la página oficial de bootstrap <https://getbootstrap.com/>

Existen varias formas de integrar bootstrap a un proyecto, estas son descritas a continuación:

1. Una forma es utilizando los archivos Sass y JavaScript de Bootstrap a través de npm, Composer o Meteor. Para proceder con la instalación se ejecuta dentro del proyecto el siguiente comando:

```
npm install bootstrap
```

2. Otra opción es usar el BootstrapCDN, donde solo deben incluir al proyecto las referencias los siguientes archivos css y js. Esta instalación la veremos en un ejemplo.

### Solo CSS

```
<link rel="stylesheet"  
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  
      integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
```

### JS, Popper.js y jQuery

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"  
      integrity="sha384-q8i/X+965Dz0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo" crossorigin="anonymous"></script>  
<script  
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/  
      popper.min.js"  
      integrity="sha384-UO2eT0CpHqdsJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>  
<script  
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"  
      integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFF/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
```

Nota: Estas archivos de referencia son tomados de la página oficial de bootstrap, siempre es conveniente copiarlos de la página oficial ya que van cambiando las versiones. <https://getbootstrap.com/>

- 
3. Otra alternativa para integrar bootstrap es descargando los archivos fuentes, compilarlos y agregar las librerías a nuestro proyecto.

### Práctica Instalación de Bootstrap 4

1- En Visual Studio Code, creamos una página index.html y establecemos las etiquetas correspondientes:

```
<!DOCTYPE html>
<html lang="es">

<head>
  <h1>Título</h1>
</head>

<body>
  <p>texto de prueba</p>
</body>

</html>
```

2- si abrimos este archivo en un navegador, por ejemplo: chrome. veremos la siguiente pagina

---

# Titulo

texto de prueba

3- para instalar bootstrap, vamos a utilizar el segundo método mencionado anteriormente, es decir, dentro de la etiqueta <header> agregaremos la referencia de estilo, y dentro de la etiqueta <body> el resto de las referencias. El archivo quedaría de la siguiente manera.

```
<!DOCTYPE html>
<html lang="es">

<head>
    <h1>Título</h1>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
        integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2M
        ZW1T" crossorigin="anonymous">
</head>

<body>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6
        jizo" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
        integrity="sha384-U02eT0CpHqdSJQ6hJty5KVphtPhzWj9W01c1HTMGa3JDZwrnQq4sF86dIHND
        z0W1" crossorigin="anonymous"></script>
```

```
<script  
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"  
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFF/nJGzIxFDsf4x0xIM+B0  
7jRM" crossorigin="anonymous"></script>  
  
<p>texto de prueba</p>  
  
</body>  
  
</html>  
  
</html>
```

4- Al guardar los cambios, podemos verificar que bootstrap está funcionando correctamente, si cambio la tipografía por la que bootstrap usa por defecto. Entonces en chrome veremos la siguiente imagen:

# Titulo

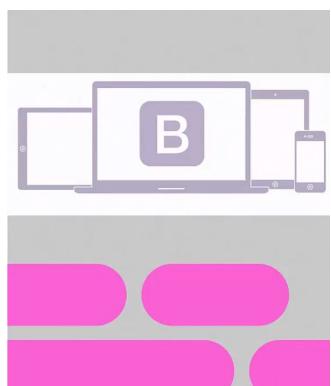
texto de prueba

- Ver código de practica: 1-InstalarBootstrap

## Conceptos importantes

---

### Viewport



Lo podemos considerar como el área que tenemos disponible para nuestro sitio web, es decir, el contenedor donde se muestra nuestro sitio web. Si lo vemos en una computadora por ejemplo el viewport sería el navegador.

Debido a la naturaleza mobile-first de bootstrap 4 nuestro viewport por defecto es el de dispositivos móviles, es decir, con tamaños menores a 586 píxeles, lo que haremos entonces será diseñar nuestro sitio web en base a esta medida y partir de ahí, para adaptarlo a otros tamaños como tablets o monitores.

---

En versiones anteriores de bootstrap lo que hacíamos era diseñar en base a un navegador o aún tamaño promedio de una computadora e ir adaptando ese sitio a dispositivos móviles.

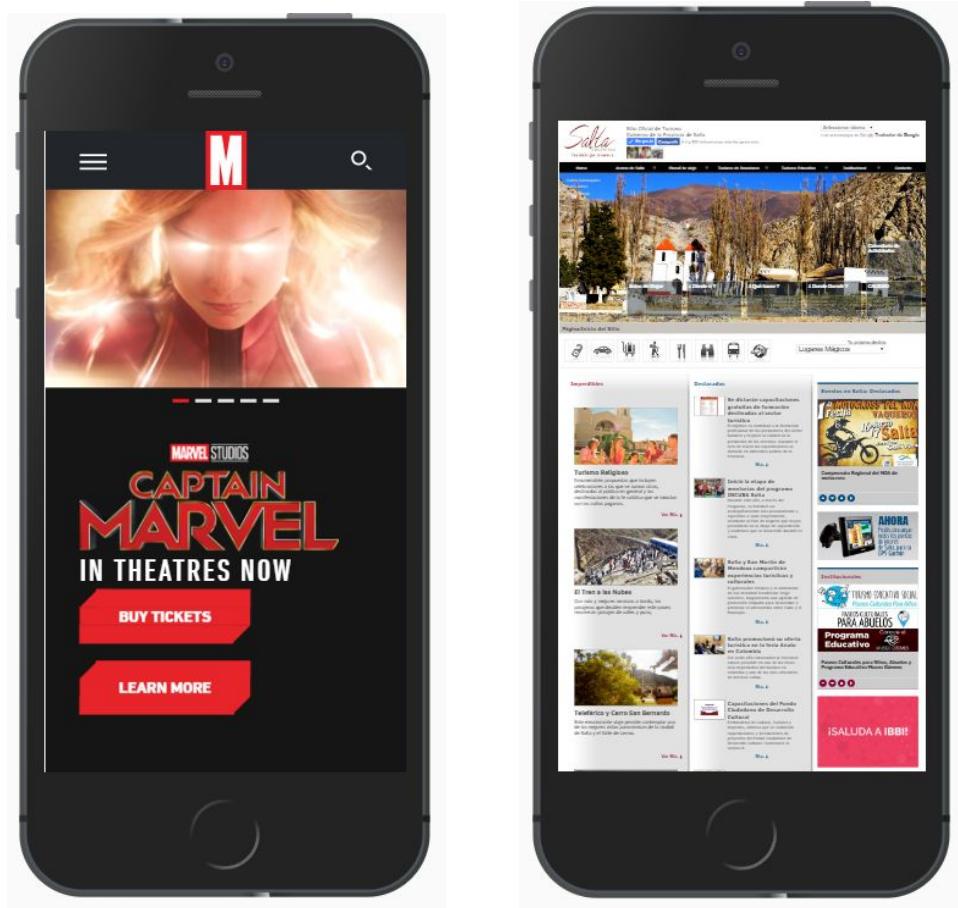
### Importante para bootstrap 4

Como mencionamos anteriormente bootstrap 4 es mobile-first, debido a esto se recomienda agregar a nuestra página index la siguiente etiqueta, para lograr el correcto funcionamiento cuando se renderiza nuestra web.

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

el nombre de la etiqueta es viewport, en el atributo content dice que el width (ancho) es igual al width del dispositivo, la escala será igual a 1, y el shrink-to-fit = no indica que si el sitio se va hacer pequeño, en nuestro caso ponemos que no, porque nosotros vamos a crear nuestra aplicación partiendo del tamaño correcto para móviles y los elementos se adaptarán al dispositivo, pero en los casos donde no se crea la web responsive pueden ver que los sitios se hacen mucho más pequeños.

por ejemplo:



El primer caso, muestra como se ve la página de Marvel en dispositivos móviles y podemos apreciar que es correctamente responsive.

El segundo caso es la página de turismo de salta, podemos observar que esta página se hace mucho más chica para que podamos ver su contenido.

por lo tanto el código de nuestra página index queda de la siguiente manera:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
```

---

```
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJ1SAwiGgFAW/dAiS
6JXm" crossorigin="anonymous">>

<title>Hello, world!</title>
</head>
<body>
  <h1>Hello, world!</h1>

  <!-- Optional JavaScript -->
  <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG
5KkN" crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa
0b4Q" crossorigin="anonymous"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8j0t6vLEHfe/JQGiRRSQQxSffWpi1MquVdAyjUar5+76PV
CmYl" crossorigin="anonymous"></script>
  </body>
</html>
```

## Contenido de Boostrap 4

Bootstrap 4 en general está dividido en cuatro elementos principales: Layout, Content, Components, Utilities.

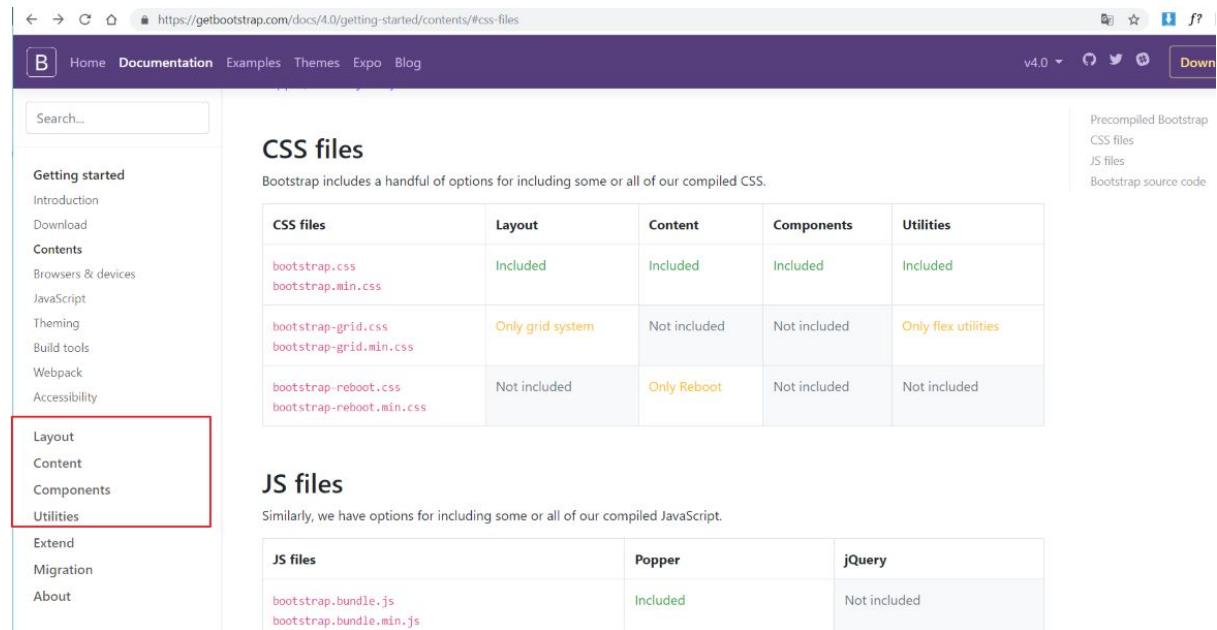
**Layout:** se refiere a componentes y opciones para dar estructura a tu proyecto. Incluye por ejemplo un sistema de cuadrículas o grid system entre otros.

**Content:** incluye elementos de tipografía como headers o títulos, elementos para incluir código y también tiene contenido para imágenes, figuras y tablas.

**Components:** son diversos elementos que incluyen CSS y JS. Entre los componentes podemos encontrar por ejemplo alertas, botones, carrusel, dropdown, modals, etc.

**Utilities:** estos son elementos que nos permitirán manejar de manera más rápida y fácil aspectos sencillos pero que se utilizan frecuentemente. Por ejemplo visibility que sirve para mostrar u ocultar elementos. otras utilidades son los bordes, los colors, flex para usar flexbox, position y text.

Podemos encontrar más información sobre estos elementos en la página oficial de bootstrap.



The screenshot shows the official Bootstrap documentation for version 4.0. The left sidebar has a red box around the 'Layout', 'Content', 'Components', and 'Utilities' sections. The main content area has two tables: one for 'CSS files' and one for 'JS files'. The 'CSS files' table includes columns for file name, layout, content, components, and utilities. The 'JS files' table includes columns for file name, Popper, and jQuery.

CSS files	Layout	Content	Components	Utilities
bootstrap.css bootstrap.min.css	Included	Included	Included	Included
bootstrap-grid.css bootstrap-grid.min.css	Only grid system	Not included	Not included	Only flex utilities
bootstrap-reboot.css bootstrap-reboot.min.css	Not included	Only Reboot	Not included	Not included

JS files	Popper	jQuery
bootstrap.bundle.js bootstrap.bundle.min.js	Included	Not included

Dependiendo del proyecto que se vaya a realizar, podemos considerar que no es necesario importar todo el contenido de bootstrap, si solo necesitamos incluir algunas cosas, podemos guiarnos de la tabla que se muestra en la imagen anterior donde se ve que archivo css y js debemos importar en nuestro proyecto para algunos o todos los elementos mencionados anteriormente, layout, content, components y utilities.

## Soporte en dispositivos y navegadores

Se puede verificar en la página web oficial de bootstrap cuales son los dispositivos y navegadores compatibles. Cuandos nos dan las especificaciones de un proyecto tenemos que pensar cuales son los dispositivos que usarán nuestra aplicacion o pagina web, para ellos podemos verificar en la siguiente tabla que dispositivos aceptan el uso de bootstrap 4.

Home Documentation Examples Themes Expo Blog v4.0 ▾

Search...

**Getting started**

- Introduction
- Download
- Contents
- Browsers & devices**
- JavaScript
- Theming
- Build tools
- Webpack
- Accessibility
- Layout
- Content
- Components
- Utilities
- Extend
- Migration
- About

**Mobile devices**

Generally speaking, Bootstrap supports the latest versions of each major platform's default browsers. Note that proxy browsers (such as Opera Mini, Opera Mobile's Turbo mode, UC Browser Mini, Amazon Silk) are not supported.

	Chrome	Firefox	Safari	Android Browser & WebView	Microsoft Edge
<b>Android</b>	Supported	Supported	N/A	Android v5.0+ supported	Supported
<b>iOS</b>	Supported	Supported	Supported	N/A	Supported
<b>Windows 10 Mobile</b>	N/A	N/A	N/A	N/A	Supported

**Desktop browsers**

Similarly, the latest versions of most desktop browsers are supported.

	Chrome	Firefox	Internet Explorer	Microsoft Edge	Opera	Safari
<b>Mac</b>	Supported	Supported	N/A	N/A	Supported	Supported
<b>Windows</b>	Supported	Supported	Supported, IE10+	Supported	Supported	Not supported

For Firefox, in addition to the latest normal stable release, we also support the latest [Extended Support Release \(ESR\)](#) version of Firefox.

## Sistema de grilla y estructura

### Layout: Container y responsive breaks

Podemos decir que layout es la estructura o distribución de elementos que tendrá nuestro proyecto.

### Containers o contenedores

Los containers nos permiten agregar elementos de acuerdo a nuestras necesidades. Por ejemplo para ver cómo se usa, podemos copiar el ejemplo de la web de bootstrap.

```
<div class="container">
  <!-- Content here -->
</div>
```

Esto nos crea un container similar al siguiente.



Podemos variar de muchas formas la clase container, por ejemplo si queremos un container que ocupe el 100% del viewport, podemos agregar la clase “container-fluid” y obtendremos el siguiente aspecto.

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



## Responsive breakpoints

Dependiendo del tamaño del viewport es que cambiará el tamaño del container, lo que nos indica a partir de qué tamaño de viewport cambiará el container son las @media queries.

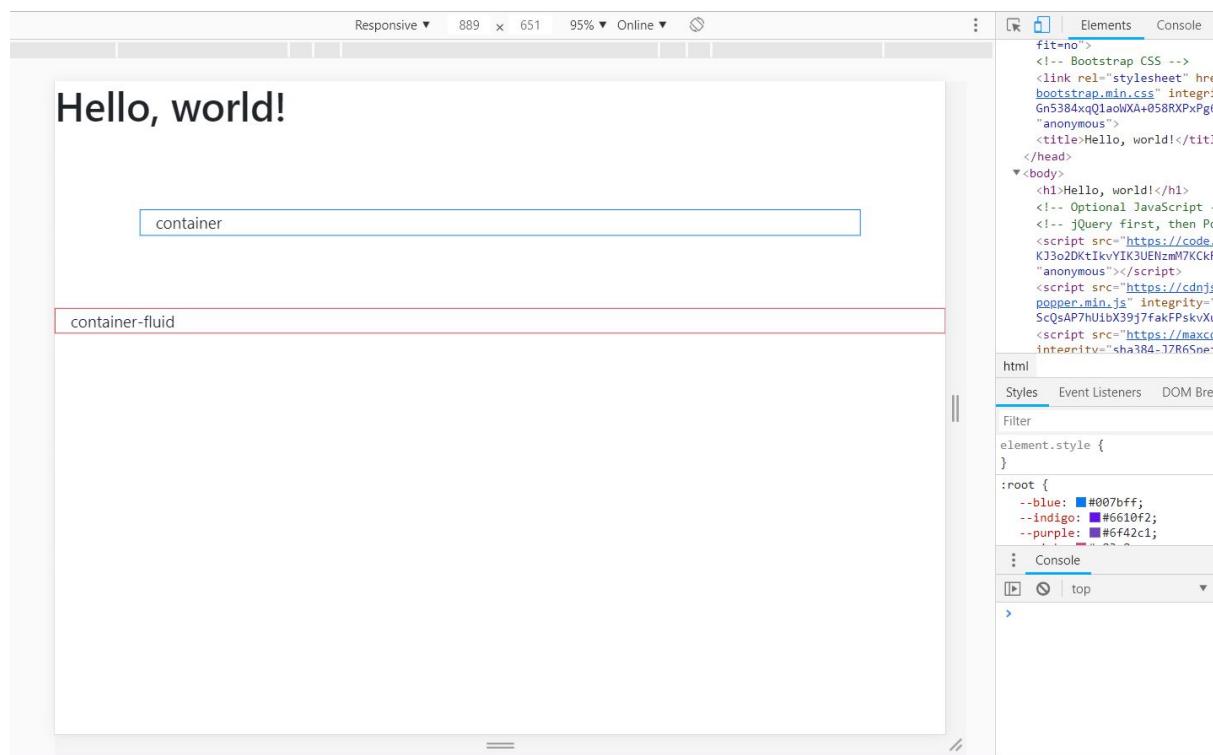
```
// Extra small devices (portrait phones, less than 576px)  
// No media query since this is the default in Bootstrap  
  
// Small devices (landscape phones, 576px and up)  
@media (min-width: 576px) { ... }  
  
// Medium devices (tablets, 768px and up)  
@media (min-width: 768px) { ... }  
  
// Large devices (desktops, 992px and up)  
@media (min-width: 992px) { ... }  
  
// Extra large devices (large desktops, 1200px and up)  
@media (min-width: 1200px) { ... }
```

Estos breakpoints ya están definidos en bootstrap y solo los utilizaremos en nuestra hoja de estilo css cuando necesitemos que nuestro proyecto cambie algo dependiendo de un tamaño en particular de viewport.

## Práctica de Containers

Crear un proyecto con dos contenedores, uno con la clase “container” y otro con la clase “container-fluid” y probar con las herramientas disponibles para desarrolladores del navegador chrome cómo se comporta en los distintos dispositivos.

Ejemplo de como quedarían los dos componentes, resaltando los bordes.



- Ver código de práctica: 2-ContainerEnBootstrap

## Otras clases de bootstrap

En el ejemplo anterior utilizamos una clase de bootstrap que es `class="border border-primary"` y `class="border border-danger"`, existen muchas clases más que podemos ver en la página oficial de bootstrap, algunas de ellas son por ejemplo:

- Border Color



```
<span class="border border-primary"></span>
<span class="border border-secondary"></span>
<span class="border border-success"></span>
<span class="border border-danger"></span>
<span class="border border-warning"></span>
<span class="border border-info"></span>
<span class="border border-light"></span>
<span class="border border-dark"></span>
<span class="border border-white"></span>
```

[Copy](#)

- Border-radius



```







```

[Copy](#)

- Color

```
.text-primary
.text-secondary
.text-success
.text-danger
.text-warning
.text-info
.text-light
.text-dark
.text-muted
.text-white
```

```
<p class="text-primary">.text-primary</p>
<p class="text-secondary">.text-secondary</p>
<p class="text-success">.text-success</p>
<p class="text-danger">.text-danger</p>
<p class="text-warning">.text-warning</p>
<p class="text-info">.text-info</p>
<p class="text-light bg-dark">.text-light</p>
<p class="text-dark">.text-dark</p>
<p class="text-muted">.text-muted</p>
<p class="text-white bg-dark">.text-white</p>
```

[Copy](#)

- Background-color



.bg-primary

.bg-secondary

.bg-success

.bg-danger

.bg-warning

.bg-info

.bg-light

.bg-dark

```
<div class="p-3 mb-2 bg-primary text-white">.bg-primary</div>
<div class="p-3 mb-2 bg-secondary text-white">.bg-secondary</div>
<div class="p-3 mb-2 bg-success text-white">.bg-success</div>
<div class="p-3 mb-2 bg-danger text-white">.bg-danger</div>
<div class="p-3 mb-2 bg-warning text-dark">.bg-warning</div>
<div class="p-3 mb-2 bg-info text-white">.bg-info</div>
<div class="p-3 mb-2 bg-light text-dark">.bg-light</div>
<div class="p-3 mb-2 bg-dark text-white">.bg-dark</div>
<div class="p-3 mb-2 bg-white text-dark">.bg-white</div>
```

[Copy](#)

Existen muchas clases mas que pueden observar en la web de bootstrap.

## Layout: grid system

---

Bootstrap 4 está basado en Flexbox, como dijimos anteriormente, Flexbox es un tipo de layout establecido en css3 como lo fue el block layout, inline layout o table layout en versiones anteriores.

Básicamente flexbox se refiere a estilos que proveen una forma más eficiente de estructurar, alinear y distribuir el espacio entre elementos aun cuando el tamaño de los elementos es dinámico.

Bootstrap 4 tiene integrado un sistema de grilla que nos ayudará acomodar los elementos en la página, estableciendo un orden y que sea fácil de estructurar, pero además este sistema nos ayudará a que el sitio web sea responsive y podamos cambiar el tamaño y la ubicación de los elementos dependiendo del viewport que estemos usando para visualizar la web.

### Ejemplo.

One of three columns	One of three columns	One of three columns
----------------------	----------------------	----------------------

```
<div class="container">
  <div class="row">
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
  </div>
</div>
```

Copy

1. Debemos tener en cuenta que siempre para usar esta estructura debemos comenzar con el `<div class="container">` dentro de este puede hacer muchas filas (row) y dentro de este varias columnas.

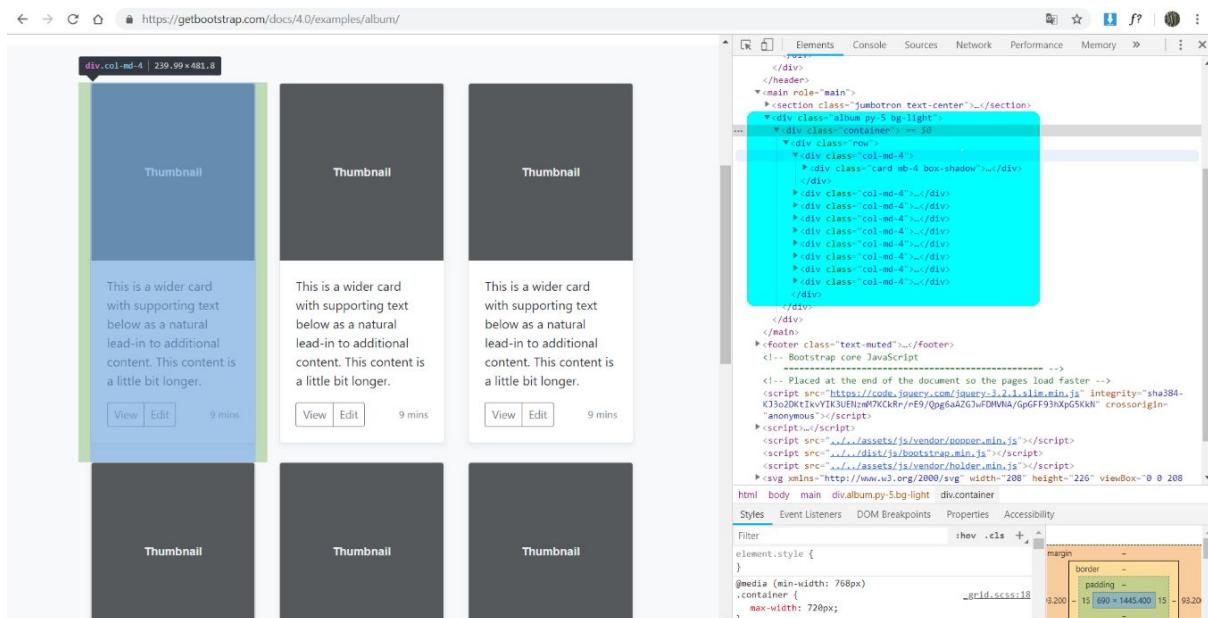
## 2. La cantidad máxima de columnas que se pueden agregar por fila es 12.

Las clases de bootstrap 4 están creadas para los breakpoints vistos anteriormente, para saber qué clase usar podemos guiarnos con la siguiente tabla.

	Extra small	Small	Medium	Large	Extra large	Tamaño del viewport
	<576px	≥576px	≥768px	≥992px	≥1200px	
<b>Max container width</b>	None (auto)	540px	720px	960px	1140px	
<b>Class prefix</b>	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-	
<b># of columns</b>	12					
<b>Gutter width</b>	30px (15px on each side of a column)					
<b>Nestable</b>	Yes					
<b>Column ordering</b>	Yes					

## Inspeccionar el grid system

Para comprender mejor el funcionamiento del grid system de bootstrap 4, podemos inspeccionar con las herramientas para desarrolladores de google chrome la siguiente pagina, que es un ejemplo de la web oficial de bootstrap: <https://getbootstrap.com/docs/4.0/examples/album/>



---

## Header y navegación

### Agregando un header

Todos los sitios web llevan un elemento html `<header>` podemos agregar distintas clases de bootstrap a la etiqueta header, dentro del `<header>` tendremos un `<div>` con la clase `main-menu` como se muestra a continuación: dentro de la etiqueta body:

```
<!-- HEADER -->
<header class="fixed-top">
  <div class="main-menu">
    <nav class="navbar">
      ...
    </nav>
  </div>
</header>
```

### Agregando un Navbar

El menú de navegación o navbar de bootstrap 4, tiene muchas opciones, ahora veremos algunas características:

1. Para asegurara una mejor experiencia se recomienda que el navbar este contenido dentro de un elemento html `<nav>`  
`<nav class="navbar navbar-light bg-light">`  
 `<a class="navbar-brand" href="#">Navbar</a>`  
`</nav>`
2. el elemento que contiene el navbar debe contener la clase `navbar`
3. Si se requiere un comportamiento diferente, como colapsar el menú en un dispositivo móvil, se debe agregar la clase `navbar-expand`.
4. También se puede usar containers dentro del navbar.

### Ejemplo

Navbar Home Link Disabled

Search

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarTogglerDemo03" aria-control
    <span class="navbar-toggler-icon"></span>
  </button>
  <a class="navbar-brand" href="#">Navbar</a>

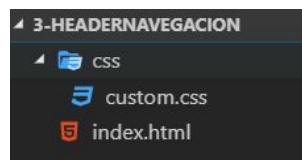
  <div class="collapse navbar-collapse" id="navbarTogglerDemo03">
    <ul class="navbar-nav mr-auto mt-2 mt-lg-0">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>
```

<https://getbootstrap.com/docs/4.0/components/navbar/>

## Estilos adicionales del header

Se puede crear páginas personalizadas, podemos utilizar bootstrap como base y agregar nuestros propios estilos.

Ejemplo:



Agregamos al proyecto anterior una carpeta con un archivo custom.css

Agregamos la referencia dentro de la página index.html al estilo custom.css

```
<link rel="stylesheet" href="css/custom.css">
```

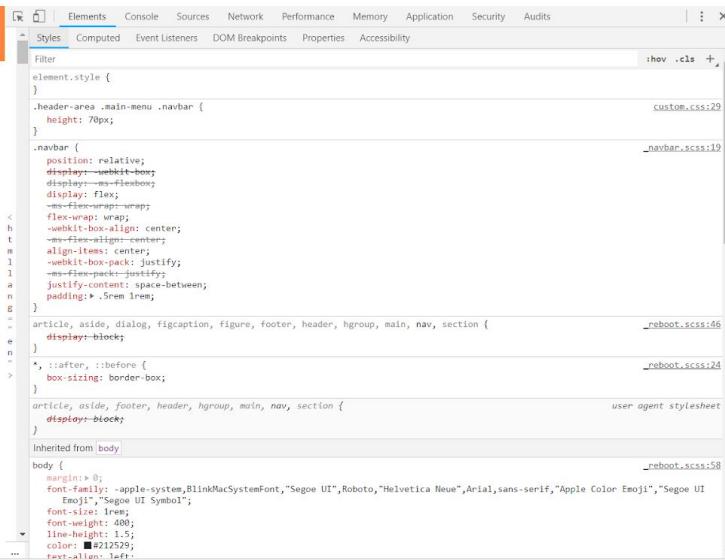
Código de archivo custom.css

```
/*
*****
```

## | HEADER

```
*****  
*/  
  
.header-area .main-menu,  
.header-area .main-menu,  
.dropdown-menu {  
    background: var(--orange-corral);  
    box-shadow: 0 3px 16 0 rgba(0, 0, 0, .1);  
}  
  
.header-area .main-menu .navbar {  
    height: 70px;  
}  
  
.header-area .main-menu .navbar .nav-link {  
    font-size: 14px;  
    color: var(--white-egg);  
    padding: 8px;  
}  
  
.header-area .main-menu .navbar .nav-link:hover {  
    font-weight: 600;  
}
```

Si vemos la web creada en el navegador:



Podemos observar en este ejemplo los estilos usados y como se sobreescreiben, regularmente lo que tendremos serán estilos por defecto que maneja el navegador, los estilos del framework en este caso bootstrap 4 y finalmente nuestros estilos personalizados en este caso nuestro custom.css.

## Diseño responsive

Si queremos que el diseño sea responsive, podemos seguir dos formas:

1. Agregar clases de bootstrap específicas de los tamaños de viewport, es decir, de los breakpoints vistos anteriormente
2. En caso que no existan las clases o no se adapten a lo que necesito, entonces podemos agregar estilos a las @media queries.

Continuando con el ejemplo anterior, si queremos que los items del menú de navegación están centrados sólo en los tamaños xs y sm, pero queremos que en los tamaños posteriores esté alineado a la derecha, entonces puedo agregar una clase ya definida por bootstrap 4.

```
<ul class="nav ml-auto mr-auto mr-md-0">
```

`mr-md-0` esta clase indica que tiene un margin right 0 únicamente para las medidas md (mediano en adelante), es decir, mediano, grande y extragrande.

También puedes utilizar las @media querie para sobreescribir comportamientos, por ejemplo agregaremos un archivo responsive.css y agregar la referencia al index, para ver cómo se comporta.

---

## contenido de archivo responsive.css

```
media (max-width: 575.98px) {  
    .header-area .main-menu .navbar {  
        height: auto;  
    }  
}  
  
@media (min-width: 576px) {  
    .header-area .main-menu .navbar .nav-link {  
        padding: .5rem 1rem;  
    }  
}  
  
@media (min-width: 768px) {  
    .header-area .main-menu .navbar .nav-link {  
        font-size: 16px;  
    }  
}  
  
@media (min-width: 992px) {}  
@media (min-width: 768px) {}
```

En el archivo index, agregamos la referencia

```
<link rel="stylesheet" href="css/responsive.css">
```

Si vemos la web creada en el navegador:

The screenshot shows the Chrome DevTools interface with the "Elements" tab selected. The left pane displays the DOM tree for a page with a header section. The right pane shows the computed styles for the selected element, which is a header area with a fixed top position. The styles include various CSS properties like height, width, font-family, and margin. The "Styles" panel at the top right lists "Computed" and "Event Listeners". The bottom status bar indicates the window is 829px wide by 705px high.

```
<!DOCTYPE html>
<html lang="en">
  <head></head>
  <body>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3oJzjLqCHPlfYiPmQZKtXuOw+TgkxUdGm+QWf4q5c+MqZlGZnZqE5Xe/a" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/Sc0O1HqT8pkwXUKxv2uXfU9IqA7M5FZDX6" crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.min.js" integrity="sha384-vZ2WRJMwsfZc8Lrqf52FFBLWTVwqJN9S8ErtmpdgXrufnudfK1CvU" crossorigin="anonymous"></script>
    <!-- HEADER -->
    <header class="header-area fixed-top">
      <div class="main-menu">
        <!-- nav class="nav-bar" --> &gt; 50
          <div class="container">
            <nav>
              </nav>
            </div>
          </div>
        </header>
        <br>
        <br>
        <br>
      </body>
    </html>
```

En este ejemplo en un dispositivo de 829 x 705 px, de acuerdo a las clases agregadas podemos observar que los elementos del menú quedan alineados a la derecha y con un tamaño de fuente más grande. En un dispositivo más pequeño se ve como la siguiente imagen.

El texto está centrado y el tamaño es mas chico

- Ver código de practica: 3 - headerNavegacion

## Flexbox en bootstrap

Para utilizar flexbox en bootstrap 4, tenemos que agregar un <div> clase `.d-flex`

I'm a flexbox container!

```
<div class="d-flex p-2">I'm a flexbox container!</div>
```

Copy

Por supuesto que existen muchas variaciones de flex, por ejemplo puedo usar flex para los distintos viewport antes vistos.

## Dirección

Variaciones para `.d-flex` y `.d-inline-flex`.

- `.d-flex`
- `.d-inline-flex`
- `.d-sm-flex`
- `.d-sm-inline-flex`
- `.d-md-flex`
- `.d-md-inline-flex`
- `.d-lg-flex`
- `.d-lg-inline-flex`
- `.d-xl-flex`
- `.d-xl-inline-flex`

También podemos dar dirección a los elementos usando por ejemplo, `.flex-row` como el siguiente ejemplo:

Flex item 1 | Flex item 2 | Flex item 3

Flex item 3 | Flex item 2 | Flex item 1

```
<div class="d-flex flex-row">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
<div class="d-flex flex-row-reverse">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
```

Copy

---

De la misma manera también podemos ordenar los elementos en columna en su versión normal y en reversa, como se ve a continuación:

Flex item 1

Flex item 2

Flex item 3

Flex item 3

Flex item 2

Flex item 1

```
<div class="d-flex flex-column">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
<div class="d-flex flex-column-reverse">
  <div class="p-2">Flex item 1</div>
  <div class="p-2">Flex item 2</div>
  <div class="p-2">Flex item 3</div>
</div>
```

Copy

## Justificar contenido

También podemos justificar el contenido, esto se puede hacer con una propiedad de css **justify-content**, pero en bootstrap 4 lo podemos hacer agregando a la etiqueta <div> la clase **justify-content** y además podemos indicar con las clases de bootstrap a donde queremos que estén alineados los elementos.

```
<div class="d-flex justify-content-start">...</div>
```



The image shows a grid of six rows, each containing three light purple rectangular boxes labeled "Flex item". The rows demonstrate different horizontal alignment properties:

- Row 1: justify-content: start; items are aligned to the left.
- Row 2: justify-content: end; items are aligned to the right.
- Row 3: justify-content: center; items are centered horizontally.
- Row 4: justify-content: between; items have space between them.
- Row 5: justify-content: around; items have equal space around them.

Copy

```
<div class="d-flex justify-content-start">...</div>
<div class="d-flex justify-content-end">...</div>
<div class="d-flex justify-content-center">...</div>
<div class="d-flex justify-content-between">...</div>
<div class="d-flex justify-content-around">...</div>
```

## Alinear items

Podemos alinear los ítems o childs dentro de un display, esta clase se refiere a la alineación vertical que tendrán los hijos. por ejemplo



The image shows a grid of two rows, each containing three light purple rectangular boxes labeled "Flex item". The rows demonstrate different vertical alignment properties:

- Row 1: align-items: start; items are aligned to the top.
- Row 2: align-items: end; items are aligned to the bottom.

```
<div class="d-flex align-items-start">...</div>
```

Esta clase **align-items-start**, alinea los ítems partiendo desde la principio del parent verticalmente hablando.



The image shows a grid of two rows, each containing three light purple rectangular boxes labeled "Flex item". The rows demonstrate different vertical alignment properties:

- Row 1: align-items: start; items are aligned to the top.
- Row 2: align-items: end; items are aligned to the bottom.

```
<div class="d-flex align-items-end">...</div>
```

En este caso la clase **align-items-end**, alinea los ítems partiendo desde el final del parent verticalmente hablando.

---

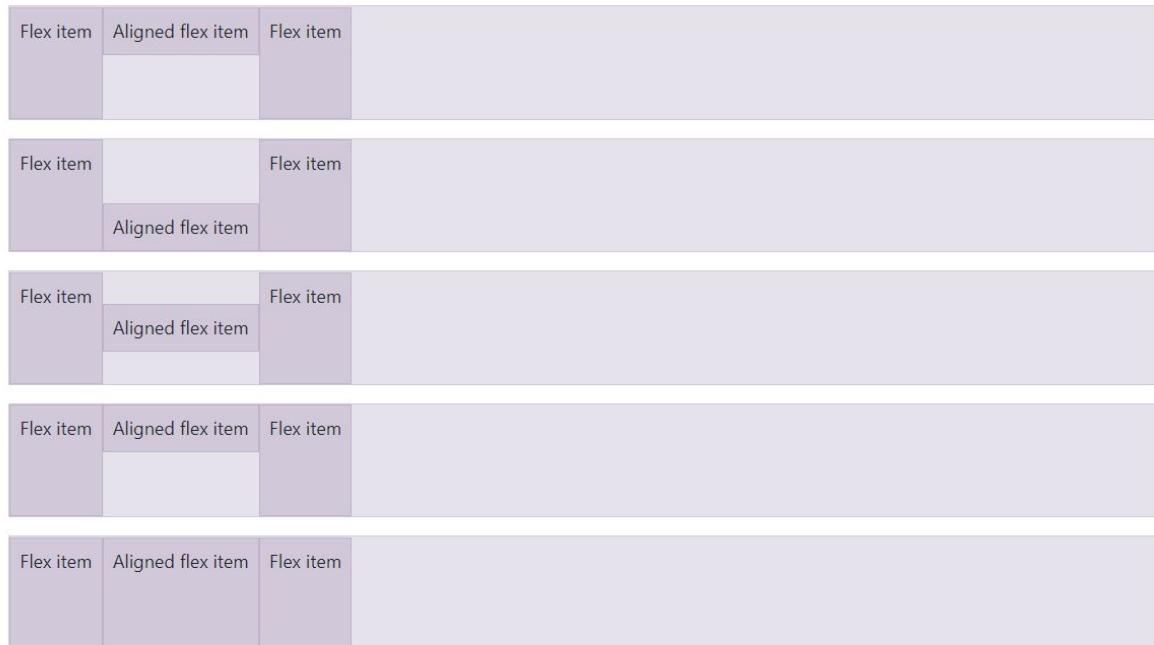
Hay otros tipos de alineación que podemos ver en la documentación de página oficial de bootstrap.

## Align self

Hasta ahora las clases vistas, se aplican a los parents, pero modifican el comportamiento que tendrán los ítems child's.

También podemos modificar el comportamiento de los ítems child's, por ejemplo con la clase

`align-self` donde establecemos la alineación vertical que tendrá cada ítem o elemento hijo.



```
<div class="align-self-start">Aligned flex item</div>
<div class="align-self-end">Aligned flex item</div>
<div class="align-self-center">Aligned flex item</div>
<div class="align-self-baseline">Aligned flex item</div>
<div class="align-self-stretch">Aligned flex item</div>
```

Copy

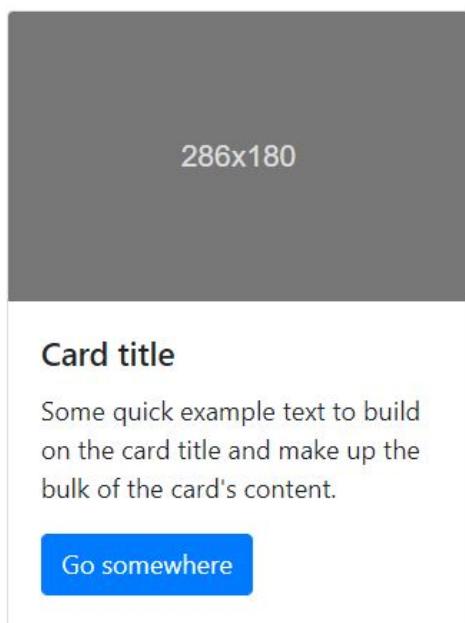
Investigar el resto de las clases de flex de bootstrap, siguiendo la documentación oficial.  
<https://getbootstrap.com/docs/4.0/utilities/flex/>

## Cards (Tarjetas)

Las tarjetas o componentes cards de bootstrap 4, son elementos que permiten agrupar diversos tipos de información o elementos en su interior, es básicamente un contenedor que permite agregar varios elementos como headers, footer, imágenes, botones entre otros. Este componente está basado en flexbox por lo que sus elementos son fáciles de acomodar como vimos.

Este componente no tiene asociado un margen o width por defecto, por lo que estos atributos serán los mismos que el parent o el que definamos nosotros.

Por ejemplo:

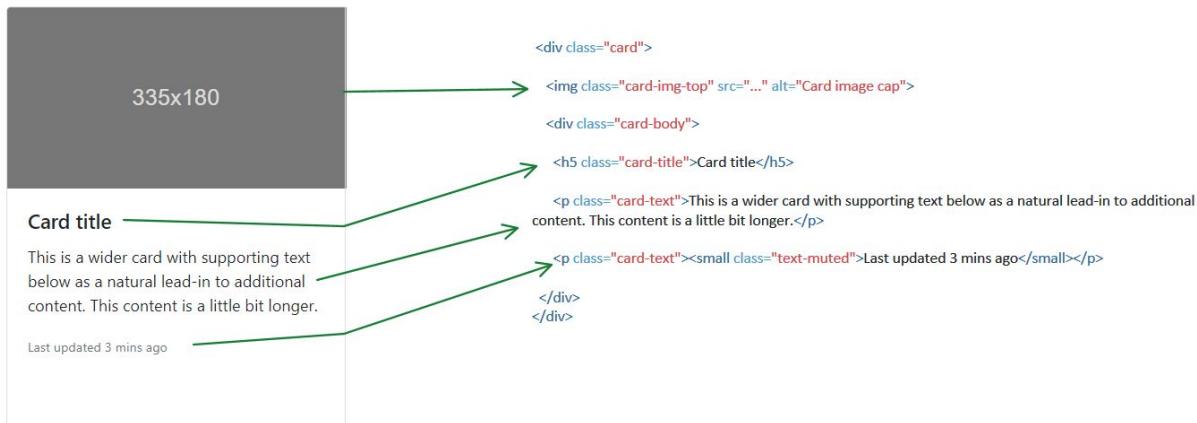


```
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Vemos que en el código anterior se agregó un tamaño, si no se lo hubiera agregado la card tomaría todo el tamaño del parent.

No es necesario incluir imágenes en una card, también puede contener sólo información textual.

En bootstrap 4 existen muchas clases para trabajar con las cards nombraremos las mas usadas:



Se puede consultar más información de las cards, en la documentación oficial de bootstrap.

## Glosario

Término	Descripción
<b>A</b>	
Algoritmo	Es un proceso para resolver un problema. Por lo general es ordenado y secuencial
<b>B</b>	
Bootstrap	Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web
<b>C</b>	
Comentario	Un comentario es un bloque de código que no se ejecuta por el intérprete o el compilador.
Compilador	Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser un código

---

	intermedio , o simplemente texto. Este proceso de traducción se conoce como compilación.
CSS	CSS (siglas en inglés de Cascading Style Sheets), en español "Hojas de estilo en cascada", es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.
D	
E	
ECMAScript	Es una especificación de lenguaje de programación publicada por ECMA International. ECMAScript define un lenguaje de tipos dinámicos ligeramente inspirado en Java y otros lenguajes del estilo de C. Soporta algunas características de la programación orientada a objetos mediante objetos basados en prototipos y pseudoclases.
Editor de texto	Un editor de texto es un programa informático que permite crear y modificar archivos digitales compuestos únicamente por textos sin formato, conocidos comúnmente como archivos de texto o “texto plano”.
F	
Flexbox	Se refiere a estilos que proveen una forma más eficiente de estructurar, alinear y distribuir el espacio entre elementos aun cuando el tamaño de los elementos es dinámico.
G	
H	
Hexadecimal	El sistema hexadecimal es el sistema de numeración posicional que tiene como base el 16. Sus números están representados por los 10 primeros dígitos de la numeración decimal, y el intervalo que va del número 10 al 15 están representados por las letras del alfabeto de la A a la F.
HTML	HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de las siglas que corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto.
I	
IDE	Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment, es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.
Index	Por lo general es el nombre que recibe la primera página de un sitio web. index.html
Inicializar variable	Inicializar una variable es asignarle un valor por defecto en el momento que estoy creando la variable. Si no inicializo la variable el valor por defecto es undefined

Intérprete	intérprete o interpretador es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel. Los intérpretes se diferencian de los compiladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.
<b>J</b>	
Java	Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.
Javascript	JavaScript es un lenguaje de programación, se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web.
Javascript	(abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,[3] basado en prototipos, imperativo, débilmente tipado y dinámico.
<b>K</b>	
<b>L</b>	
Layout	Layout es un término de la lengua inglesa que no forma parte del diccionario de la Real Academia Española (RAE). El concepto puede traducirse como “disposición” o “plan” y tiene un uso extendido en el ámbito de la tecnología.
<b>M</b>	
Maquetación	La diagramación, también llamada a veces maquetación, es un oficio del diseño que se encarga de organizar en un espacio distintos contenidos.
<b>N</b>	
Navegador	Un navegador web (en inglés, web browser) es un software, aplicación o programa que permite el acceso a la Web. Por ejemplo google chrome, mozilla firefox, microsoft edge, internet explorer, opera, etc.
Node	Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.
<b>Ñ</b>	
<b>O</b>	
<b>P</b>	
Píxel	Unidad básica de una imagen digitalizada en pantalla a base de puntos de color o en escala de grises.
<b>Q</b>	
<b>R</b>	
Renderización	Renderización (del inglés rendering) es un anglicismo usado en jerga

	informática para referirse al proceso de generar una imagen visible e inteligible para el ser humano, a partir de información digital.
Responsive	El Responsive Design o diseño adaptativo, es la técnica que se usa en la actualidad para tener una misma web adaptada a las diferentes plataformas que nos brinda la tecnología: ordenador, tablet y Smartphone.
RGB	RGB es una sigla formada por los términos de la lengua inglesa red (“rojo”), green (“verde”) y blue (“azul”). El concepto suele emplearse para referirse a un modelo cromático que consiste en representar distintos colores a partir de la mezcla de estos tres colores primarios.
RGBA	RGBA son la siglas para red green blue alpha. Suele describirse como un espacio de color, aunque en realidad es la combinación de un modelo de color RGB con un cuarto adicional denominado alpha channel.

## Bibliografía

Información de html5 -

[https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5\\_lista\\_elementos](https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos)

Informacióny ejemplos de CSS - <https://www.w3schools.com/css/default.asp>

Box Model -

[https://developer.mozilla.org/es/docs/Learn/CSS/Introduction\\_to\\_CSS/Modelo\\_cajas](https://developer.mozilla.org/es/docs/Learn/CSS/Introduction_to_CSS/Modelo_cajas)

Web oficial de Bootstrap - <https://getbootstrap.com/>

Guia completa de flexbox - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)