«Talento Tech»

Desarrollo Web 1

Clase 10

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 10: Repaso y finalización del proyecto final

Temario:

- Diseño Responsive y Mobile First
- Flexbox y Flexbox Responsivo
- Grillas para Maquetación Web y Pseudo-Clases
- Transiciones, Transformaciones e Introducción a Bootstrap
- Dominios, Hosting y SEO

Media query

¿Alguna vez has abierto la misma página web en diferentes dispositivos como tablets o smartphones y has notado que se veía diferente?

Esto se debe a que el diseño se adapta al tamaño de pantalla del dispositivo. Esta adaptación es muy útil, ya que contribuye a la accesibilidad del sitio web y permite que los usuarios puedan visitarlo desde cualquier lugar.

Para lograr esta adaptación, se utilizan las media queries, que son consultas que el **CSS** realiza al navegador para determinar el tamaño de pantalla en el que se está visualizando la página. En función de este tamaño, el **CSS** aplica propiedades específicas para modificar los elementos según sea necesario. En la siguiente presentación, exploraremos más a fondo este concepto.

Responsive

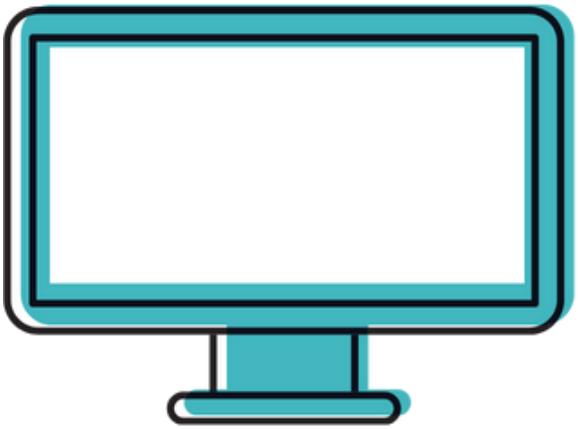
El término "responsive" o responsivo en español, se refiere a la capacidad de un sitio web para adaptarse y responder de manera óptima a diferentes tamaños de pantalla y dispositivos, brindando una experiencia de usuario adecuada en cada uno de ellos. Permite que el contenido y el diseño de un sitio web se ajusten automáticamente para adaptarse a dispositivos móviles, tabletas y pantallas de diferentes resoluciones.

.

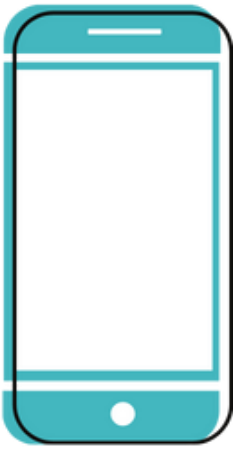
Cuando comenzamos a pensar en desarrollar una página web, **¿La imaginamos primero en computadora o en celular?**

Equipo Desktop

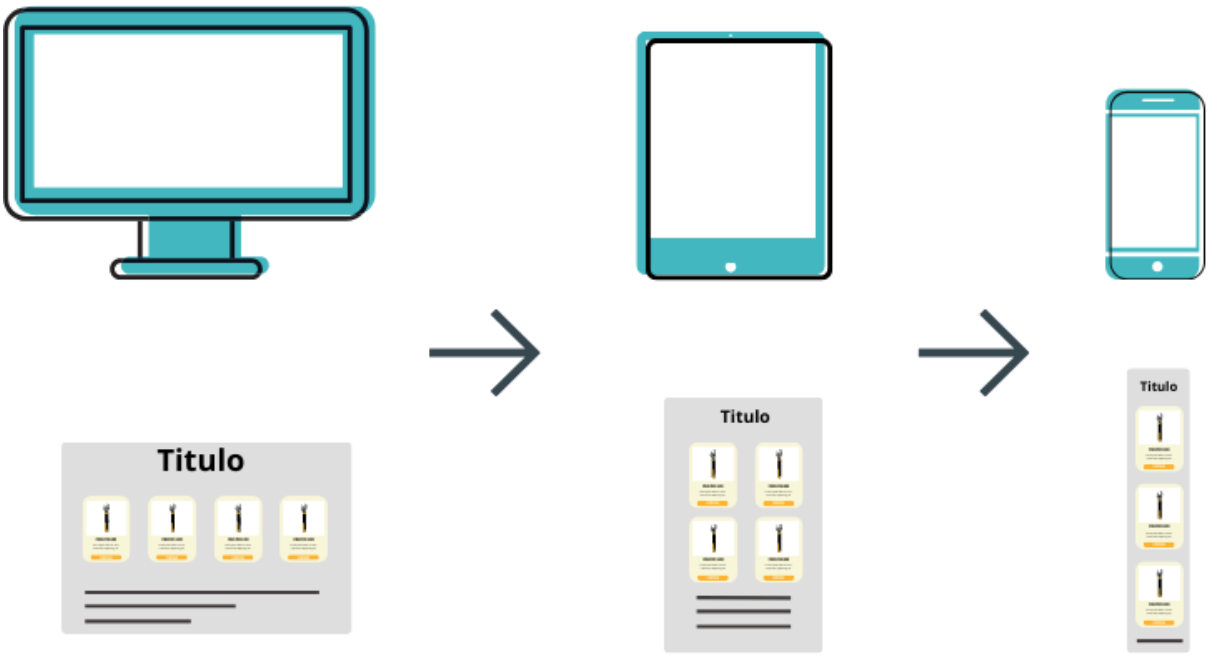
(Equipo Escritorio)



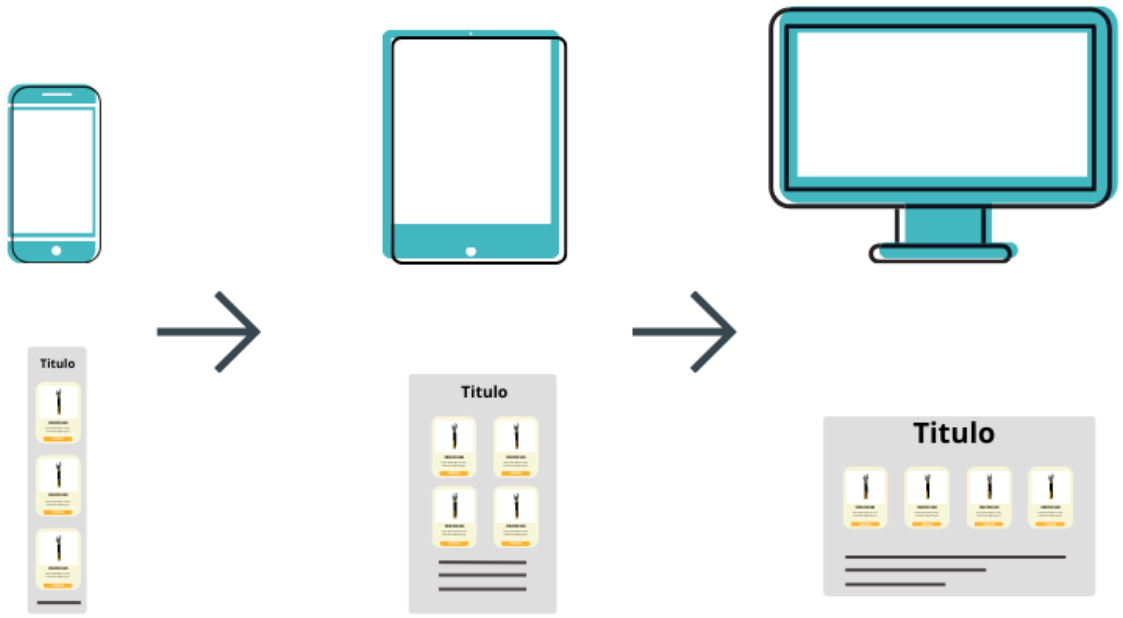
Equipo Mobile First
(Equipo Primero Mobil)



El equipo desktop primero piensa el desarrollo para computadoras y luego piensa cómo se va a adaptar a dispositivos más pequeños.



El equipo mobile primero piensa el desarrollo para dispositivos pequeños como celular y luego piensa cómo se va a adaptar a dispositivos más grandes.



Esta es una decisión que no debe tomarse a la ligera. Sea cual sea la mentalidad que elijamos para desarrollar el sitio, siempre vamos a tener que optimizarlo para todo tipo de dispositivos. Trabajar con mobile first, significa que el sitio web está originalmente optimizado para los dispositivos móviles y que la versión de escritorio es solo una adaptación de la versión móvil y lo mismo se aplica, aunque a la inversa, cuando pensamos en desktop first. Por eso, creo que lo mejor para decidir es considerar desde qué dispositivos tu sitio va a ser visto.

Media query

Las media queries (o consulta d medios en español) son una técnica de CSS que permite aplicar estilos diferentes según las características del dispositivo y la pantalla en la que se está visualizando un sitio web. Con las media queries, se pueden establecer condiciones y reglas CSS específicas para diferentes tamaños de pantalla, resoluciones, orientaciones o incluso características del dispositivo.
La sintaxis de la misma es:

@media (condición de tamaño de pantalla) {

/* Estilos para pantallas con ancho máximo o mínimo de 000 px*/
}

Esta regla CSS puede ir escrita en cualquier parte del documento, pero es recomendable que siempre quede al final de todas las reglas CSS, ya que es lo último que se suele ajustar en el momento del desarrollo.

Veamos su estructura con ejemplos.

Esta media query se lee:
“Para todas las resoluciones menores o iguales a 768px, aplica las siguientes reglas”

Esta media query se lee:
“Para todas las resoluciones mayores o iguales a 768px, aplica las siguientes reglas”

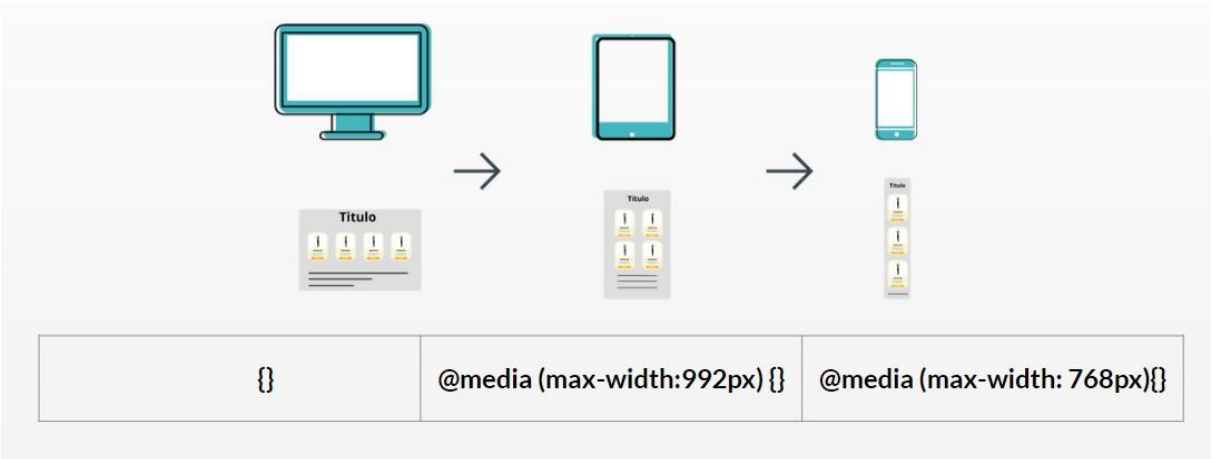
Medidas estándar

Si bien podemos usar muchas media queries, lo ideal es mantenernos dentro de un estándar, es por eso que te dejamos acá las medidas más populares que se usan al día de la fecha.

Dispositivos muy pequeños	Dispositivos pequeños	Dispositivos medianos	Dispositivos grandes	Dispositivos extra grandes
<576px	≥576px	≥768px	≥992px	≥1200px

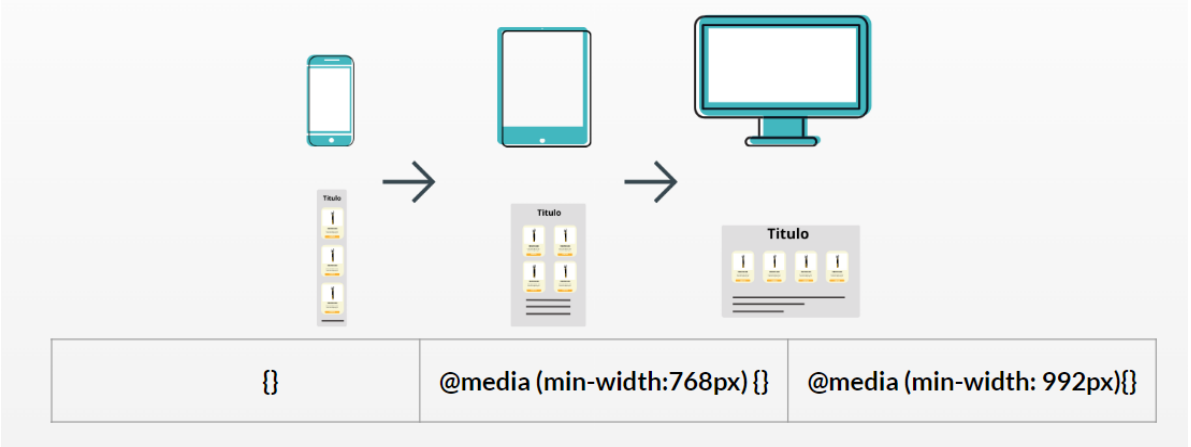
max-width

En este caso, somos equipo desktop, por lo tanto la media query a utilizar es con la condición max-width. La sentencia sería: asignar ciertas propiedades en dispositivos más pequeños que una cantidad determinada de px.



min-with

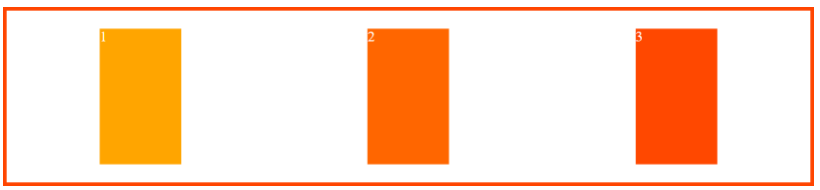
El equipo mobile first utilizará en su mayoría la media query con la condición min-with, y la sentencia sería: designar ciertas propiedades en dispositivos más grandes que tantos px.



Si bien ambas media query se pueden combinar, es recomendable trabajar con una sola de estas estrategias. **Sólo se trabaja combinando ambas maneras en casos muy puntuales.**

Layout responsive

Ya aprendimos cómo acomodar los elementos que componen el sitio web uno al lado de los otros en un tamaño de pantalla de computadora.
También aprendimos sobre las media query, que son métodos para modificar propiedades del sitio web en diferentes tamaños de pantalla.
Ahora nos falta generar la acción de responsive, es decir, hacer que los elementos cambien su disposición según el tamaño de pantalla. Veamos cómo lograrlo con este ejemplo.
Supongamos que tenemos este contenedor con tres divs en su interior.



Para poder cambiar el tamaño de la pantalla debemos hacer clic derecho sobre el sitio web y seleccionar inspeccionar. (también podemos acceder al modo inspeccionar presionando la tecla f12) Luego, en la opción de visualización responsiva para poder cambiar el tamaño de la pantalla libremente.

Hasta el momento, su código es así.

HTML

```html
<div class="contenedor">
    <div class="caja uno">1</div>
    <div class="caja dos">2</div>
    <div class="caja tres">3</div>
</div>
```

CSS

```css
.contenedor {
  border: 4px solid orangered;
  margin: 0 auto;
  margin-top: 50px;
  display: flex;
  align-items: start;
  justify-content:space-around;
}
.caja {
  width: 90px;
  height: 150px;
  margin: 20px;
  display: flex;
  color: white;
}
```

CSS

```css
.uno {
  background-color: orange;
}
.dos {
  background-color: rgb(255, 102, 0);
}
.tres {
  background-color: rgb(255, 72, 0);
}
```

- Estructura HTML contenedor y cajas. Cada caja con sus clases
- CSS del contenedor y de las cajas
- CSS del color de cada caja

Aplicando media queries

Ahora, vamos a modificar las propiedades del contenedor y las disposiciones de las cajas cuando la pantalla mida menos de 768 px de ancho, y le asignaremos nuevas propiedades. Al final del código CSS entonces agregaremos el siguiente código. Veámoslo por partes.

```css
@media(max-width: 768px) {
  .contenedor{
    display: block;
    max-width: 300px;
  }
  .caja {
    margin: 0 auto;
    margin-top: 20px;
  }
}
```

- Declaramos la media query con el tamaño máximo de pantalla
- Le asignamos nuevas propiedades al contenedor:
-display:block; bloqueamos su flexibilidad
-Le damos un máximo de ancho de 300px
- Le asignamos nuevas propiedades a las cajas para centrarse y haya espacio entre ellas

Te invitamos a poner en práctica este ejercicio con un layout más completo:

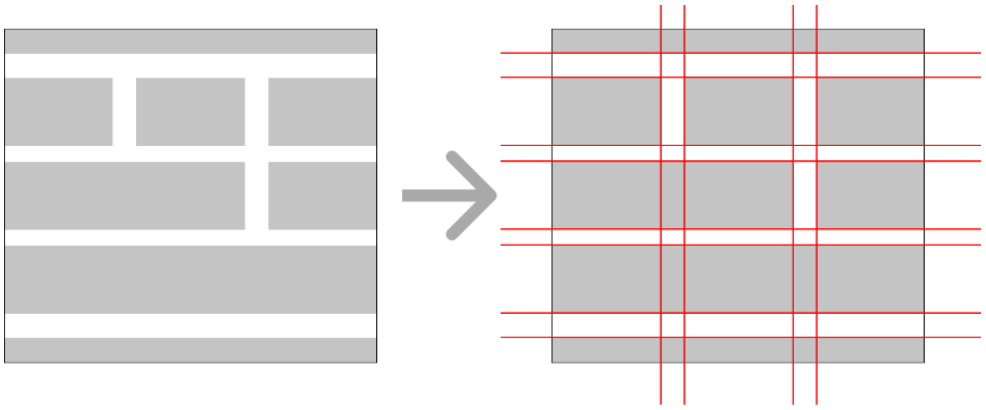
Poné a prueba una disposición más compleja para los div.
¿Pasa algo con las imágenes? ¿Hacen falta más divs?

Te recomendamos hacer uso de las media query en el tamaño de pantalla responsivo del inspector. Es decir, visualizar el sitio con un tamaño más pequeño y sobre esa vista realizar los cambios que creas necesarios.

Sistema de grillas

Las grillas son sistemas o estructuras de diseño que nos permiten organizar y distribuir elementos en una página de manera ordenada y flexible. Se basan en una estructura de filas y columnas que proporciona un marco para alinear y posicionar los diferentes elementos de contenido. Las grillas nos ayudan a crear diseños balanceados y proporcionales, facilitando la creación de diseños responsivos que se adaptan a diferentes tamaños de pantalla.

Supongamos que tenemos una web con este diseño.



Podemos pensar que podemos armar una grilla con 3x5 (tres columnas y cinco filas), y agregar espacio entre ellas. Luego distribuir en ella los diferentes elementos, como div, textos o imágenes que componen el sitio.

Cómo construimos una grilla.

Al igual que en flexbox, para armar una grilla precisamos un contenedor padre tenga la propiedad display:grid; para indicarle al contenedor que debe comportarse como grilla y que sus elementos hijos se acomoden dentro de ella. Para eso, cada elemento dentro de ese contenedor grilla tendrá sus propias coordenadas para saber dónde ubicarse.

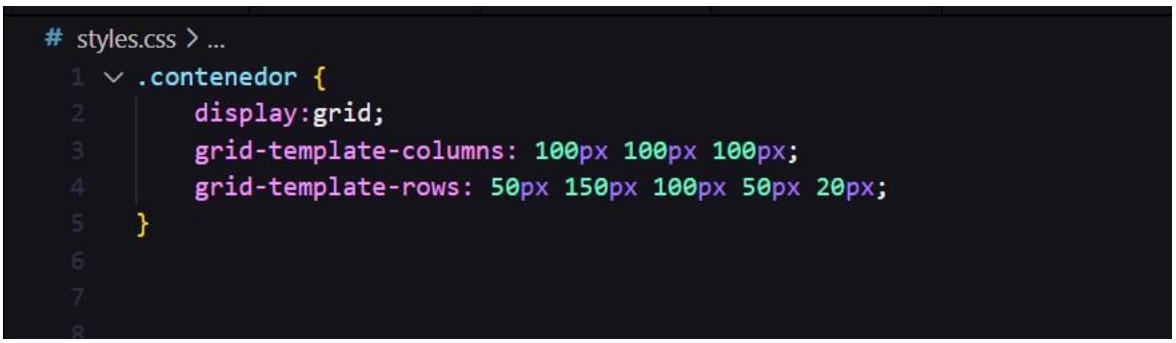
Proceso paso por paso:

1. Creamos un contenedor con la propiedad **display: grid**;

2. Asignamos la cantidad de columnas que va a tener la grilla escribiendo la propiedad grid-template-columns. Por ejemplo:
   grid-template-columns: 100px 100px 100px;
   La cantidad de valores que se escriban en esta propiedad, será la cantidad de columnas que tenga la grilla. En este ejemplo tendrá tres columnas de 100px de ancho cada una.

3. Lo mismo para designar la cantidad de columnas de la grillas. Por ejemplo:
   grid-template-rows: 50px 150px 100px 50px 20px;
   En este caso creamos filas con diferentes anchos.

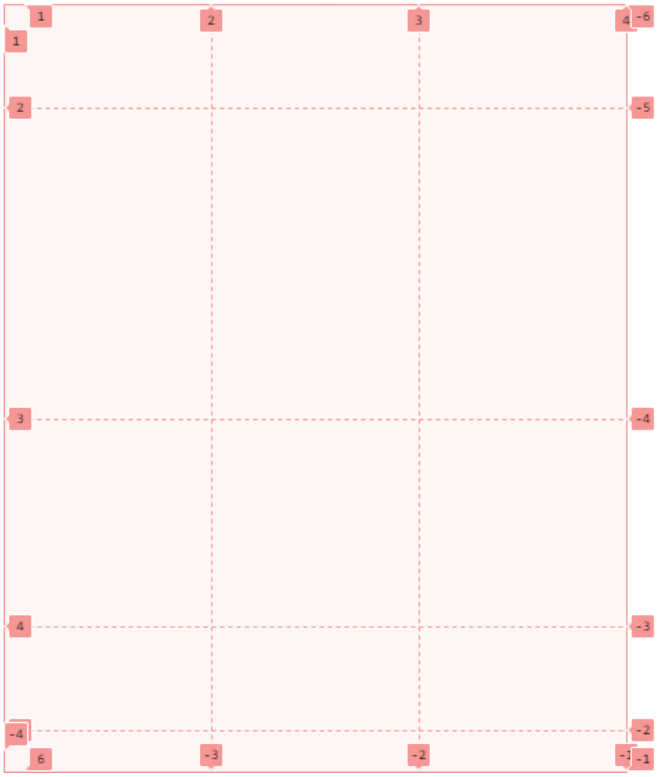
El código css sería así:



Para poder ver el resultado de la grilla, precisamos hacerla visible en el navegador. Al no tener elementos dentro del contenedor, la grilla está vacía, y por ende, invisible. Pero podemos ver su esqueleto.

Lo que se ve como resultado es la grilla vacía con sus respectiva cantidad de filas y columnas declaradas anteriormente.

Por cada línea divisoria veremos unos números. Esos números nos ayudarán a ubicar elementos dentro de la grilla y servirán como coordenadas.



4. Por último, podemos elegir el espacio que puede existir entre las columnas y filas generando un espacio entre sí (muy similar a margin, pero exclusivo para el sistema de grillas). Para lograr esto se utiliza:
   column-gap: 20px; Para dar espacios entre columnas, es decir, de forma vertical.
   row-gap: 10px; Para dar espacios entre filas, es decir, de forma horizontal. El código completo se vería así:

```css
.contenedor {
    display:grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 50px 150px 100px 50px 20px;
    column-gap: 20px;
    row-gap: 10px;
}
```

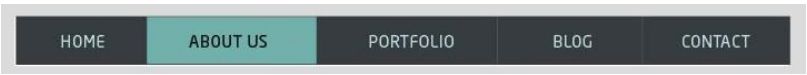
Flexbox

Hasta el momento, hemos estado agregando elementos de forma individual sin considerar su posición en el HTML. Sin embargo, con la introducción de flexbox, seremos capaces de organizar grupos de elementos de manera estratégica y crear diseños más complejos. Te invitamos a ver la siguiente presentación para aprender cómo lograrlo de manera efectiva.

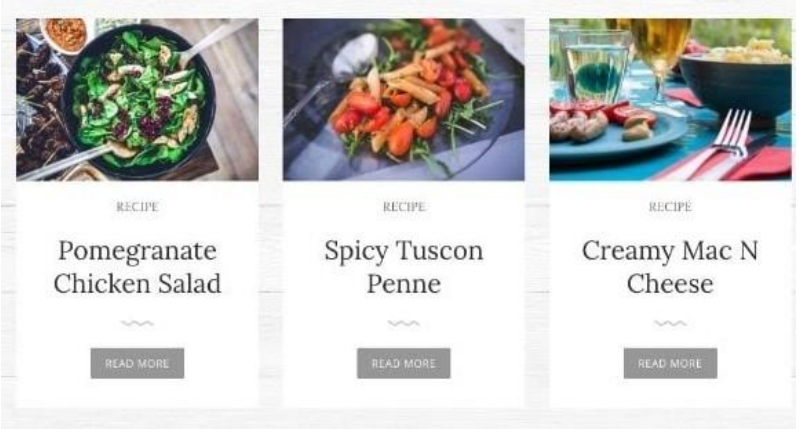
Flexbox es un método que permite crear diseños flexibles y responsivos en páginas web. Proporciona una forma eficiente y fácil de alinear, distribuir y organizar elementos en un contenedor, independientemente de su tamaño o estructura. En otras palabras, nos permite generar contenedores flexibles para poder ordenar los elementos uno al lado del otro, incluso los elementos de bloque.

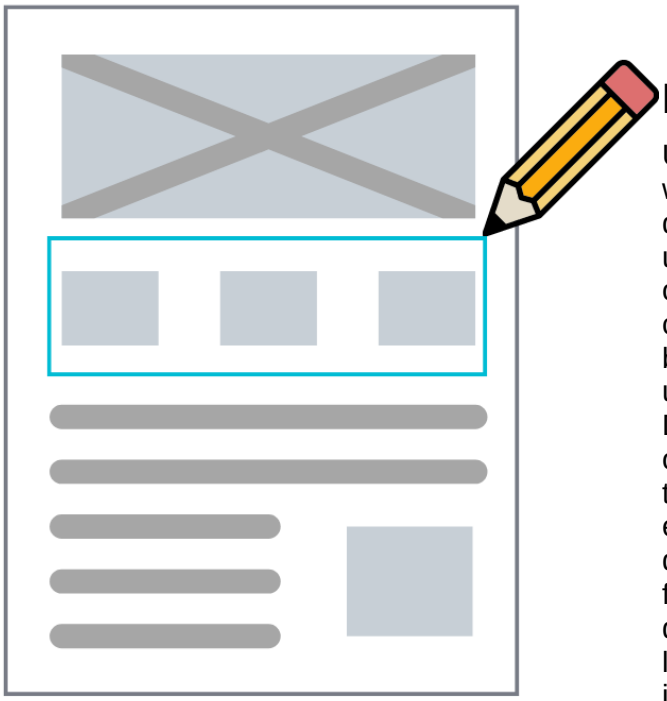
Ejemplos:

Veamos algunos ejemplos en sitios web donde se puede utilizar flexbox.



Algunos ejemplos pueden ser la barra de navegación donde hay enlaces unos al lado del otro, o cards (tarjetas). Pueden ser también galería de imágenes o layouts más complejos.





Layouts

Un layout, en el contexto del desarrollo web, se refiere a la estructura o disposición visual de los elementos en una página web. Es la forma en que se organizan y se distribuyen los componentes, como textos, imágenes, botones y otros elementos, para crear una interfaz coherente y atractiva. Determina la ubicación, el tamaño y el orden de los elementos en la página, y tiene un impacto significativo en la experiencia del usuario y en la facilidad de uso del sitio web. Un buen layout facilita la navegación, la comprensión de la información y la interacción con los elementos interactivos. Y lo más importante, marca el camino para los y las desarrolladoras que deben crear los códigos del sitio, cumpliendo una función de guía.

Display:Flex

Flexbox significa “caja flexible”. Quiere decir que lo que hace que los elementos se acomoden uno al lado del otro es su contenedor. Veamos un ejemplo.
Supongamos que tenemos el siguiente DIV y que contiene otros div más pequeños dentro.

El código es así:

HTML

```html
<div class="contenedor">
   <div class="caja uno"></div>
   <div class="caja dos"></div>
   <div class="caja tres"></div>
</div>
```

CSS

```css
.contenedor {
    border: 4px solid orangered ;
    margin: 0 auto;
    margin-top: 50px;
}
.caja {
    height: 50px;
    width: 50px;
    margin: 20px;
}
.uno{
    background-color: orange;
}
.dos{
    background-color: rgb(255, 102, 0);
}
.tres{
    background-color: rgb(255, 72, 0);
}
```

Para que las cajas se acomoden una al lado debemos aplicar la propiedad **display:flex**; en el contenedor. Esto afectará a los child o hijos directos del contenedor, no así a los nietos,

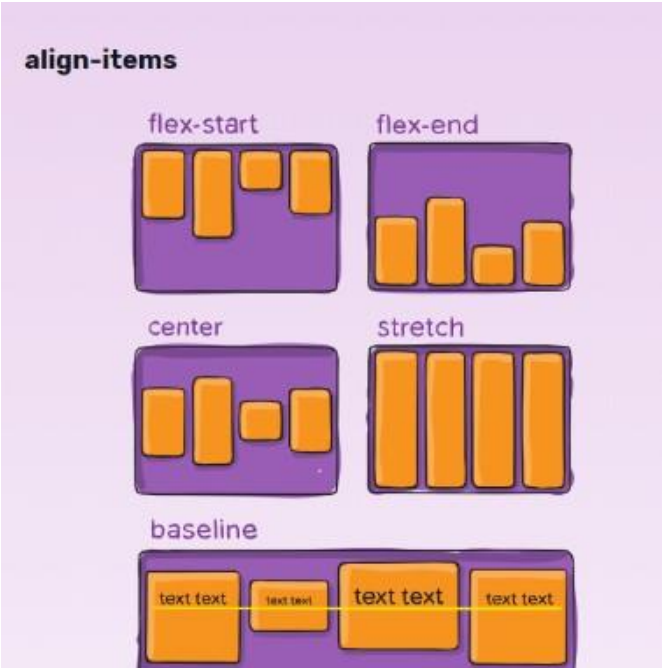
es decir, a los elementos que se encuentren dentro de las cajitas naranjas. Esto posicionará los elementos de la siguiente manera sin importar si son elementos de línea o de bloque.

CSS

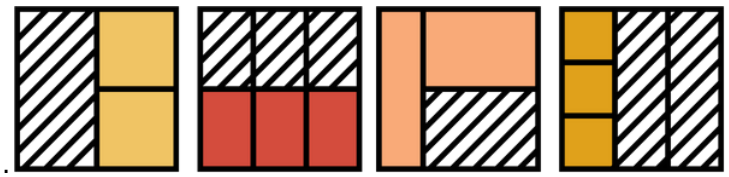
```css
.contenedor {
    border: 4px solid orangered ;
    margin: 0 auto;
    margin-top: 50px;
    display: flex;
}
```

Con una sola línea de código podemos lograr este reordenamiento de elementos. Pero, no es sólo esto. Ahora queda identificar en nuestros layout qué elementos están dentro de contenedores como hicimos en la clase 2 cuando estudiamos contenedores DIV y SPAN, y aplicar display:flex; en el caso que sea necesario. Luego, alinear los elementos de manera vertical u horizontal con más propiedades para el contenedor flexible.

Alineación de elementos:



Ya aprendimos a hacer un contenedor flexible. Ahora, debemos alinear y centrar los elementos que contiene. Para ello existen propiedades para alinear los elementos de manera horizontal y vertical y que deben aplicarse al contenedor, no a los elementos que están adentro. Es al contenedor al que debemos pedirle cómo debe ordenar sus children.



justify-content: … ;

Justify content es una propiedad cuyos valores permite ordenar elementos de forma horizontal.

justify-content:start;

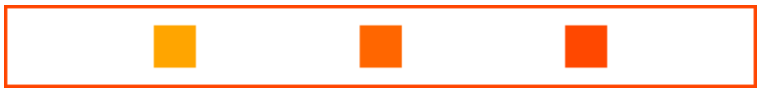
justify-content:end;

justify-content:center;

justify-content: space-evenly;

justify-content: space-around;

justify-content: space-between;



align-items: … ;

Align-items es una propiedad cuyos valores permite ordenar elementos de forma vertical.

align-items: start;

align-items: end;

align-items: center;

align-items: stretch;

align-items: baseline;

Self: Alineado individual

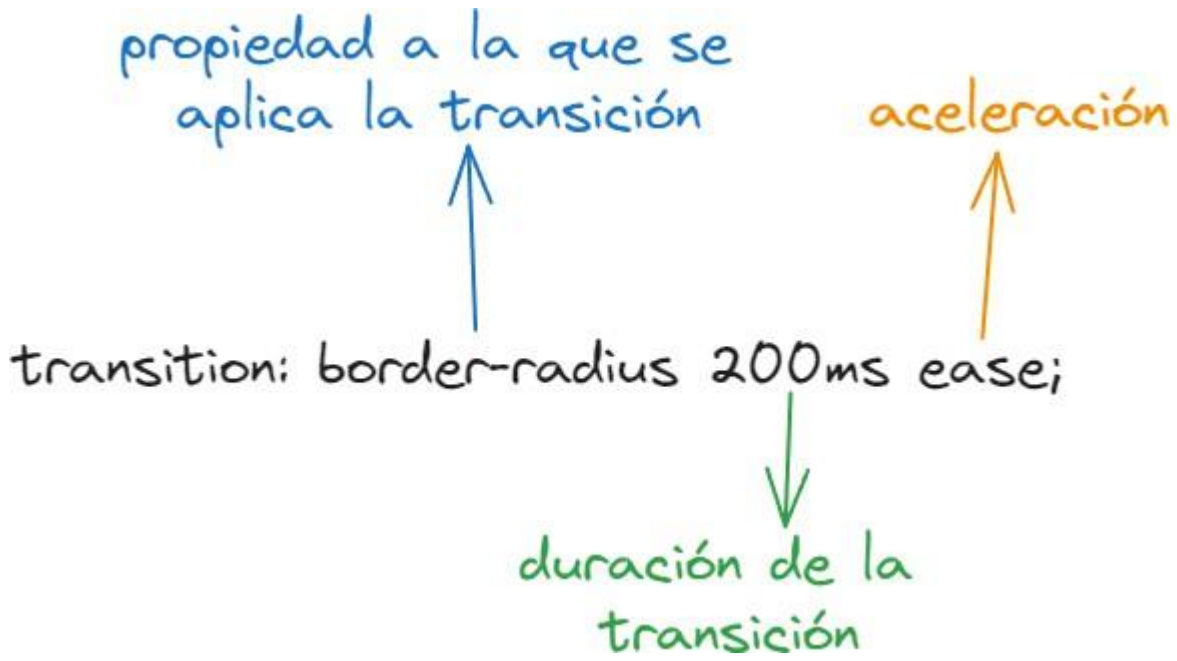
Además de acomodar los elementos de manera grupal, también podemos hacerlo de manera individual, en el caso de que se quiera que un elemento se ubique en un lugar diferente pero que siga formando parte del mismo contenedor. Esto se logra con **justify-self**

para el eje horizontal, y **align-self para el eje vertical**, aplicándolo directamente sobre el elemento a acomodar y no en el contenedor.
En este caso el contenedor tiene **align-items:start**; pero el segundo contenedor tiene **align-self:end;** haciendo que se coloque de manera casi independiente en la base del contenedor y no en el inicio o start.

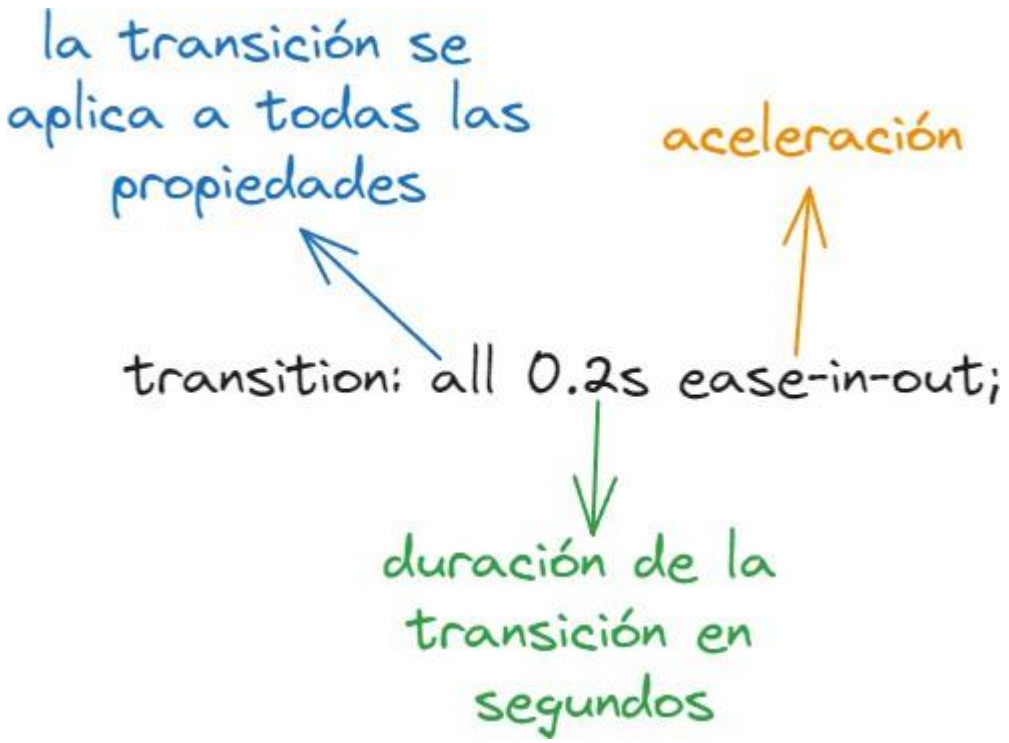
Transition:

Como su nombre lo indica en inglés, transition es transición. Se refiere a una propiedad que podemos aplicar a un elemento para suavizar o transicionar los cambios efectuados con el hover cuando le pasamos el mouse por encima. En el apartado anterior generamos cambios en los elementos con transform, pero con la transition lo que logramos es manipular a través de propiedades la velocidad y el modo en el que un elemento pasa de un estado a otro.
Para ver cómo lograr este efecto te invitamos a ver la siguiente presentación.

La propiedad transition acepta varios valores. Por ejemplo:



Si más de una propiedad va a sufrir cambios en nuestro elemento podemos utilizar la palabra **all** para que la transición se aplique a todas ellas.



Uso de transition

La propiedad transition es utilizada en el grupo de propiedades “original” del elemento. Y aparte, como venimos trabajamos, las propiedades que se harán visible con hover. Veamos la siguiente estructura repasando el tema anterior de transform.

```css
.caja {
  width: 50px;
  height: 50px;
  background-color: orange;
  margin: 50px;
  transition: transform 200ms ease;
}

.caja:hover{
  transform: rotate(20deg);
}
```

En este ejemplo, la propiedad que va a transicionar es la de “transform”, que sucederá durante el hover. Es decir, rotate.

Bootstrap

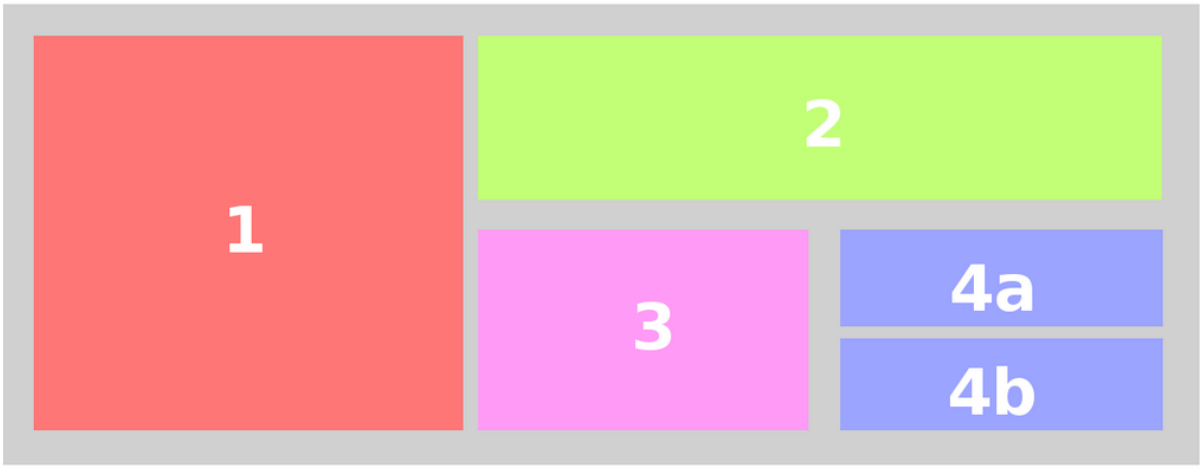
Bootstrap es una librería muy popular de desarrollo web de código abierto que facilita la creación de sitios web y aplicaciones con un diseño responsivo y moderno. Proporciona un conjunto de estilos CSS predefinidos, componentes reutilizables y scripts de JavaScript, que permiten a los desarrolladores crear interfaces de usuario atractivas y funcionales de manera rápida y sencilla.

Características de Bootstrap

Algunas características importantes de Bootstrap son:

Grilla flexible

Bootstrap utiliza un sistema de 12 columnas que permite organizar y distribuir el contenido de manera flexible en diferentes dispositivos y tamaños de pantalla.



Componentes predefinidos

Bootstrap ofrece una amplia variedad de **componentes listos para usar**, como botones, barras de navegación, tarjetas, formularios, entre otros. Estos componentes **se pueden personalizar y combinar** fácilmente para adaptarse a las necesidades del proyecto.

Estilos y temas: Bootstrap proporciona una amplia gama de estilos CSS predefinidos que permiten mejorar la apariencia de los elementos y lograr una coherencia visual en todo el sitio web. También ofrece la posibilidad de personalizar los estilos y crear temas personalizados.

Compatibilidad con dispositivos móviles: Bootstrap está diseñado para ser responsivo, lo que significa que los sitios web y aplicaciones desarrollados con Bootstrap se adaptan automáticamente a diferentes tamaños de pantalla y dispositivos, como teléfonos móviles y tablets.

Integración de JavaScript: Bootstrap incluye una serie de scripts de JavaScript que agregan funcionalidades interactivas a los componentes, como carruseles, modales, pestañas, entre otros. Estos scripts facilitan la implementación de elementos interactivos sin necesidad de escribir código JavaScript personalizado.

En la siguiente presentación verás el paso a paso de cómo linkear la librería desde su sitio oficial.

Link del sitio oficial de Bootstrap haciendo clic acá

¡Manos a la obra!

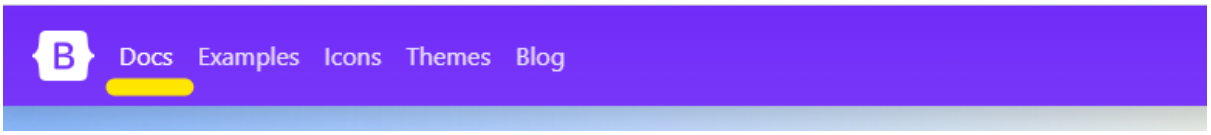
Integrando Bootstrap

Bootstrap contiene un gran conjunto de bloques de código HTML, CSS y JS pre armados que nos permite como desarrolladores ahorrar mucho tiempo de trabajo, pero para utilizarlo hay que conocer su estructura. Podríamos pensar que se trata de “materia prima” que podemos adaptar a nuestras necesidades. Es código ya empezado que podemos “amoldar” a un proyecto personal.

Vamos a integrar bootstrap al proyecto siguiendo estos pasos.

Tené en cuenta que estos pasos se están llevando a cabo con esta versión en el 2024, que a futuro puede cambiar.

Paso 1) Ingresamos al sitio oficial de bootstrap https://blog.getbootstrap.com/ y nos dirigimos a la parte superior izquierda para ingresar a la parte de “Docs” o documentación.



Al igual que como vimos con las librerías, los marcos de trabajo también vienen con su respectiva documentación para ser estudiada con el fin de utilizar correctamente sus herramientas y evitar errores en el código.

Paso 2) Para ayudarte podés poner el sitio web en español. En la parte de “inicio rápido” vemos que nos pide colocar <meta name="viewport" content="width=device-width, initial-scale=1"> en el head. Esta línea de código ya la tenemos gracias al atajo Emmet en VS Code. 😎

1. **Cree un nuevo** `index.html`**archivo en la raíz de su proyecto.** Incluya la `<meta name="viewport">`etiqueta también para un comportamiento receptivo adecuado en los dispositivos móviles.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

Paso 3) En este paso incluimos los link de CSS y JS de este framework.

2. **Incluye CSS y JS de Bootstrap.** Coloque la `<link>` etiqueta en el `<head>` para nuestro CSS y la `<script>` etiqueta para nuestro paquete de JavaScript (incluido Popper para colocar menús desplegables, ventanas emergentes e información sobre herramientas) antes del cierre `</body>`. Obtenga más información sobre nuestros enlaces de CDN .

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel='
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js'
  </body>
</html>
```

Haciendo clic en este ícono que acompaña al código podrás copiar y pegar los ejemplos de forma más rápida. 

¡Atención!

El link CDN de CSS que vá dentro del <head>, va por encima del link CSS. ¿Por qué? Porque así podremos sobreescribir las propiedades de bootstrap con las nuestras personalizadas.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/
bootstrap.min.css" rel="stylesheet"
integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02i
uK6FUUVM" crossorigin="anonymous">
<link rel="stylesheet" href="./css/index.css">
```

El link de <script> va justo antes del cierre de <body>

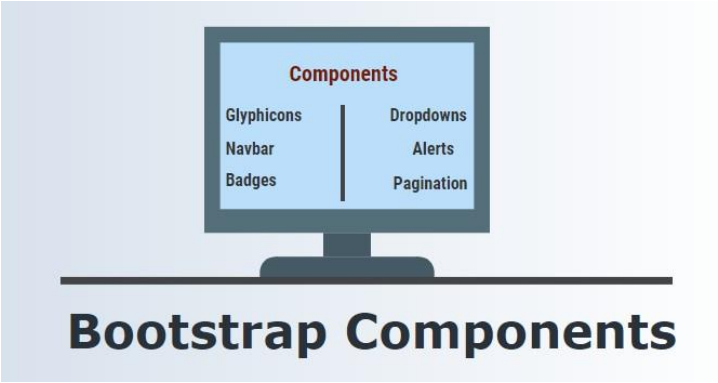
```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/
dist/js/bootstrap.bundle.min.js"
integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdi
JJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>

</body>
```

Siempre que utilices esta librería es recomendable traer los links correspondientes desde el sitio web oficial para asegurarte de que están actualizados.

¡Y listo!
Ya podemos utilizar los componentes en nuestros proyectos.

Integrando componentes

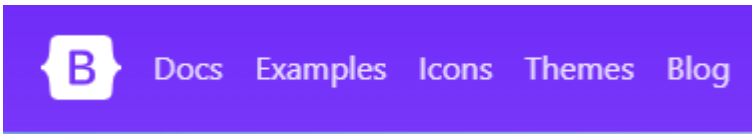


Una vez que tenemos linkeado el archivo CSS y JavaScript proporcionado por Bootstrap, es momento de ir a buscar los componentes que nos serán útiles en la construcción de nuestro sitio. Verás que, como se mencionó anteriormente, todo está predeterminado. Los colores, la tipografía, los márgenes y espacios, las medidas. Pero a no decepcionarse porque estos componentes se pueden modificar a gusto y necesidad de cada proyecto, Es decir, los atributos que ya vienen hechos se pueden sobrescribir con un archivo CSS creado por nosotros mismos. Para ver más detalles del paso a paso te invitamos a ver la siguiente presentación.

Ya aprendimos que Bootstrap es una librería que contiene código de terceros listo y disponible para que utilicemos. En el caso de esta librería, el código listo y disponible se encuentra tanto en el .css como en el .js.
Como es código que vamos a querer usar lo tenemos que integrar a nuestro proyecto. Para eso utilizamos la etiqueta <link> para el CSS y la etiqueta <script> para el JS.

Todas las instrucciones se encuentran en la página
https://getbootstrap.com/
igual que en la guía anterior.

Una vez que tenemos integrada la librería a nuestro proyecto, no nos queda más que comenzar a buscar en la documentación cómo utilizarla. Si observamos la página de Bootstrap, vamos a ver en la barra de navegación un link que dice “Docs”



Si hacemos click ahí, nos llevará al a página de la documentación de la librería.

Como verás, la página está en inglés. Si manejás inglés no hay problema pero si no, siempre existe la opción de traducir el sitio gracias al traductor de Google:

A fines prácticos el lenguaje de la guia va a seguir con la traducida al español

En la documentación vamos a encontrar toda la información para utilizar la librería. Mientras más leamos, mejor vamos a poder utilizar y conocer la librería. Es super importante leer la documentación de todas las librerías que vayamos integrando a nuestro proyecto.
¿Hay que leer todo antes de utilizarla?
No. Si bien está bueno tener un amplio conocimiento de la librería, muchas veces se puede aprender a demanda. Cuando nos encontramos con algo que queremos hacer, leemos y buscamos a ver si encontramos una solución.

En esta guía te contamos lo que sí o sí tenés que saber de Bootstrap, pero hay mucho más por explorar.

Componentes

Una vez que hicimos click en “Docs”, estamos en la sección de la documentación. A la izquierda hay una barra de navegación que nos muestra todos los temas, algo así como un índice.
Cada sección vale la pena ser explorada y vamos a encontrar con que hay secciones más complicadas y otras más fáciles. Por ahora nos vamos a enfocar en la sección de “Componentes”.

Los componentes de Bootstrap son código HTML que ya vienen con las clases adecuadas para que se muestren de una forma particular.
El paso siguiente es copiar estos códigos y pegarlos en nuestro proyecto, y ya vamos a tener los componentes funcionando.

Barra de navegación/ Navbar

Cuando hacemos click en Componentes > Barra de navegación, se nos despliega toda la documentación pertinente a la barra de navegación. En los primeros párrafos nos explica cómo funciona, qué clases admite y otros detalles.
Si comenzamos a ir hacia abajo en la página vamos a ver que hay ejemplos de barra de navegaciones junto con el código a copiar.

La lógica con los componentes, es siempre la misma:

1. Hacemos click en el componente que nos interesa.
2. Leemos la información / documentación.
3. Buscamos el modelo del componente que nos sirva.
4. Copiamos el código.
5. Lo pegamos en nuestro proyecto..

No olvidemos tener linkeado el CSS y el JS de Bootstrap para que puedan funcionar los componentes que queremos incorporar.

Una vez que encontremos el componente que nos guste y resuelva la necesidad que tenemos, simplemente lo pegamos en nuestro proyecto.

Si ahora abrimos nuestro archivo, vamos a ver algo como lo siguiente:
Es decir que tenemos la navbar lista para usar en nuestro proyecto.

¿Eso es todo?
Bueno… depende. Eso es todo a la hora de integrar un componente, pero ahora podemos cambiar al código y modificarlo para que se adecúe a las necesidades que tenemos.

Modificando la barra de navegación / navbar
Una vez que detectamos cada elemento en su línea de código traída desde Bootstrap, es leer sus clases. Veamos el siguiente ejemplo de un elemento <ul> del navbar anterior:

Su línea de código es así:

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

Como podemos ver en este ejemplo el elemento <ul> trajo clases ya asignadas desde la librería. Para saber más sobre el significado de estas clases podemos ver el instructivo de cada una de ellas que nos ofrece Bootstrap.

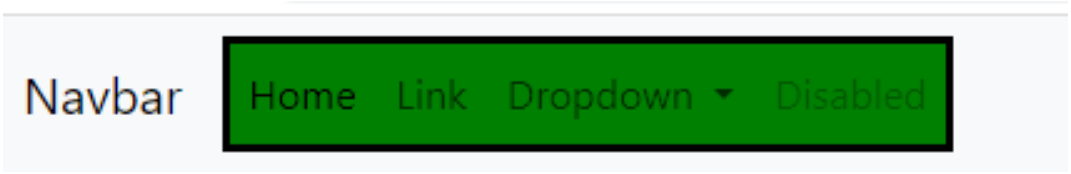
¡No te preocupes!. Puede ser un poco confuso al principio pero te contamos que no es necesario que utilices las clases que ofrece la librería si o si (aunque es mejor no eliminarlas hasta no estar seguros).

La buena noticia es que las podemos sobreescribir. Es decir, podemos inventar nuestras propias clases y aplicarlas. Veamos un ejemplo:

Para sobrescribir una característica predeterminada por Bootstrap lo que debo hacer es buscar en mi propio archivo HTML el elemento que copié desde la librería (continuamos con el ejemplo de elemento <ul>) y cuando lo encuentro, creo una nueva clase.

El siguiente paso, es agregar propiedades a esa clase desde CSS. Como hacemos normalmente, ingresamos al archivo CSS previamente linkeado al archivo HTML, nombramos a la clase y le asignamos las propiedades y valores que deseamos cambiar del elemento.

Como resultado, el nuevo navbar modificado se verá de la siguiente manera:



Con este ejemplo pudimos ver cómo sobreescribir clases y asignarles otras que se ajusten a las necesidades de cada proyecto, y las funcionalidades quedan intactas.

Conceptualizando:

Si bien no vamos a ir componente por componente, lo importante acá es registrar el modo de operar que tuvimos:

1. Linkeamos CSS y JS de bootstrap en nuestro documento.
2. Buscar los componentes que nos sirven y los integramos al proyecto copiando y pegando el código que nos ofrece la librería.
3. Investigamos sus líneas de código y efectuamos alguna modificación agregando clases y propiedades nuevas desde CSS.

Siempre tengan presente que pueden incorporar sus propios estilos y sobrescribir, en caso de ser necesario. El uso de Bootstrap es el piso, el techo es el que ustedes quieran.

Grillas con Bootstrap (grid system)

Bootstrap nos provee un sistema de grillas muy potente para crear layouts responsive. Con un enfoque mobile-first vamos a poder crear diseños con formas muy variadas para cualquier tipo de dispositivo o pantalla. Lo lograremos gracias a su sistema de 12 columnas y 5 puntos de quiebre para trabajar el responsive con muchas clases predefinidas.

¿Cómo funciona?

El sistema de grillas de bootstrap usa una serie de contenedores, filas y columnas para organizar y alinear el contenido de nuestra página. Está basado en flexbox y es completamente responsive. Así que deberás estar familiarizado con flexbox antes de comenzar a explorar el sistema de grillas.

Veamos un primer ejemplo para apreciar lo sencillo que resulta trabajar con este sistema una vez que nos hayamos familiarizado con él.

```
<div class="container">
    <div class="row">
      <div class="col-sm">
        La primera de tres columnas
      </div>
      <div class="col-sm">
        La segunda de tres columnas
      </div>
      <div class="col-sm">
        La tercera de tres columnas
      </div>
    </div>
  </div>
```

Analicemos

Analicemos lo que está sucediendo

- Las 3 columnas están contenidas dentro de un contenedor. Los contenedores nos ayudan a centrar y acomodar horizontalmente el contenido. Usamos la clase .container para tener un ancho dinámico y responsive manteniendo el contenido

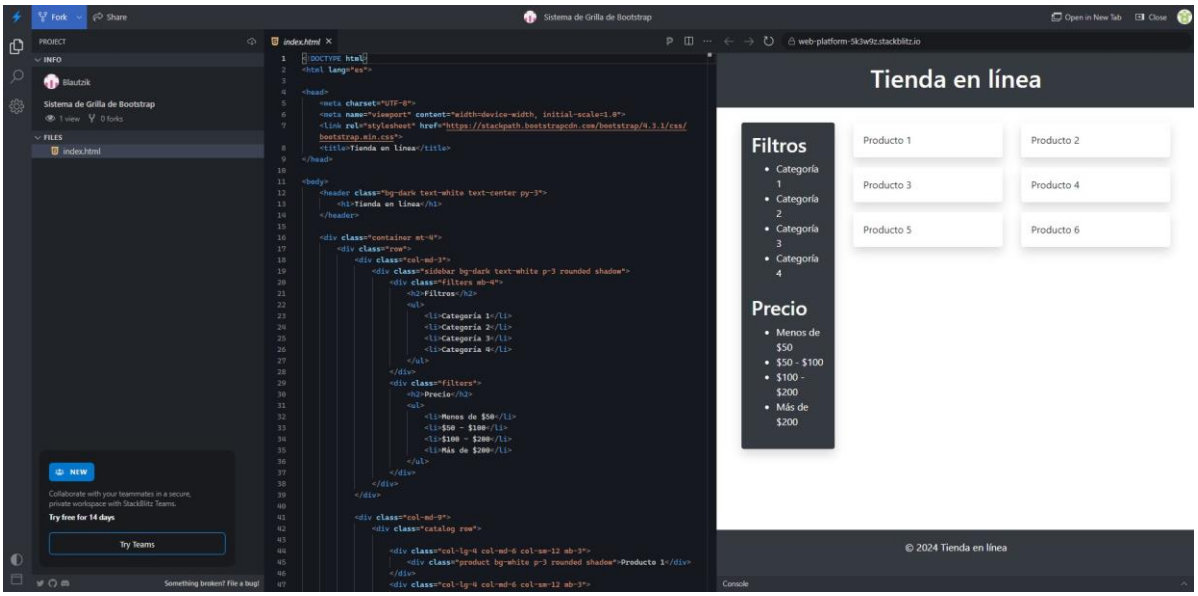
centrado. (para ocupar el 100% del ancho de la pantalla deberíamos usar la clase .container-fluid).

- La row envuelve las columnas. es comparable a la fila de una tabla. Cada row puede dividirse en 12 columnas
- El contenido debe ir siempre en columnas y solamente elementos que sean columnas pueden ser hijos directos de una row.
- Gracias a flexbox, las columnas que no tengan un ancho definido, automáticamente se van a transformar en columnas con el mismo ancho.
- Para hacer la grilla responsive contamos con 5 breakpoints (xs, para pantallas de smartphone (menos de 576px). sm, para dispositivos pequeños (más de 576px. md, para tablets (más de 768 px). lg, para notebooks o desktop (mas de 992px). xl, para pantallas extra grandes (más de 1200 px)
- Al tener 3 elementos con la clase .col-sm estamos indicando que cada elemento ocupará un ancho del 33% del espacio disponible a partir del breakpoint sm. es decir que en pantallas de más de 576px de ancho se verán separadas en 3 columnas pero en pantallas más pequeñas se verán una debajo de la otra.

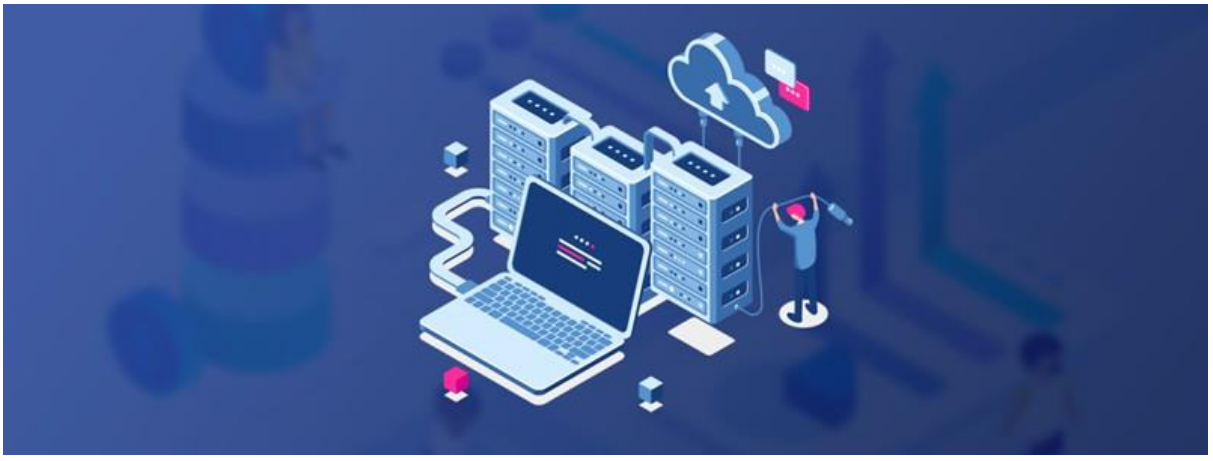
Ejemplo

Les compartimos este ejemplo para que puedan ver cómo crear un layout similar al que hicimos en la clase de grillas utilizando solamente bootstrap

https://stackblitz.com/edit/web-platform-5k3w9z?file=index.html



Dominio y hosting



Para que un sitio web funcione, necesita ambas cosas: un hosting y un dominio web.Poniéndolo en palabras sencillas, hosting es el espacio de almacenamiento donde se

alojan los archivos que componen tu sitio web, y el dominio es el nombre con el que tus visitantes accederán a él. Ahondemos más en el tema para quitarte todas las dudas.

Dominios:

Un dominio en la web es la dirección única que identifica un sitio en Internet. Por ejemplo, "www.ejemplo.com.ar" es un dominio. Los dominios se utilizan para facilitar la navegación y para identificar de manera única a cada página. Están compuestos por dos partes: el nombre de dominio y la extensión.

El nombre es la parte principal y única del dominio. En el ejemplo "**www.ejemplo.com.ar"**, **"ejemplo"** es el nombre de dominio.

La extensión de dominio, es la parte que sigue al nombre. En "www.ejemplo.com.ar", ".com.ar" es la extensión de dominio.

Registrar un dominio implica pagar una tarifa para asegurar el derecho de uso exclusivo de ese nombre en Internet por un período de tiempo específico. El registro de dominios se realiza a través de proveedores de dominios autorizados. Al registrar un dominio a nuestro nombre somos “dueños” de esa dirección en internet.

Algunos proveedores populares en argentina son:
Nic.ar
Don web

Hosting

El **hosting**, es un servicio que permite a los usuarios guardar y publicar sus páginas web en Internet. Todos los archivos, imágenes y contenido relacionado con tu sitio deben residir en un servidor conectado a Internet para que los usuarios puedan acceder a ellos.

Al contratar un servicio de hosting, estamos comprando (o alquilando) espacio en sus servidores para almacenar estos archivos. Algunos proveedores de hosting también ofrecen otros servicios como cuentas de email, bases de datos, etc. En síntesis, un proveedor de hosting nos brinda todo lo necesario para que nuestra página esté online.

En resumen, mientras que un dominio es la dirección única que identifica tu sitio web en Internet, el hosting es el servicio que te permite almacenar y compartir los archivos de tu página para que otros puedan visitarlo. Los dos son fundamentales para tener un sitio web en funcionamiento


Buenos Aires
aprende
Buenos
Aires
Ciudad