



# Curso de Front-End

## Clase 05



# Agenda de la clase



# Agenda

- Repaso.
- Flexbox. 🔥
- Frameworks. 🚀
- Bootstrap.
- Ejercicios.



# Repaso



# ¿Qué vimos la semana pasada?

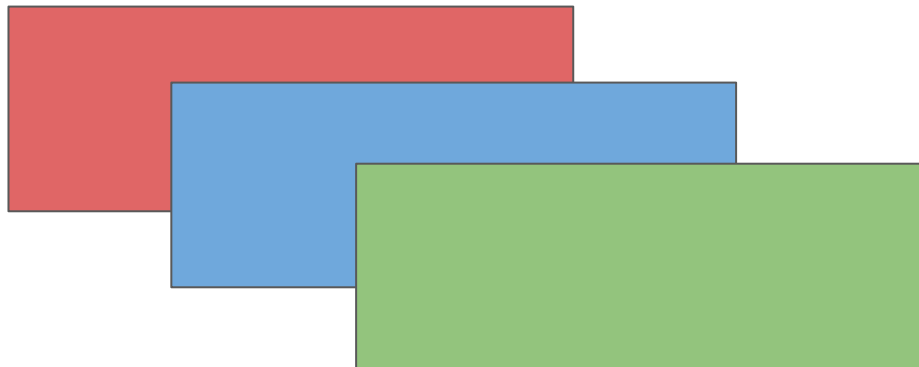
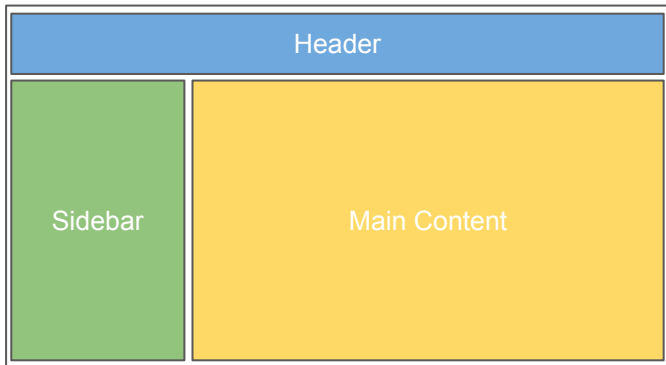
- **HTML** (para definir **QUÉ** elementos queremos presentes en la página)
  - Etiquetas: `<html>`, `<body>`, `<h1>`, `<p>`, `<a>`, `<img>`, `<ul>`, `<li>`, `<div>`, `<span>`, etc.
  - Atributos: `href`, `src`, `alt`, `style`, `id`, `class`, etc.
  - Semántica y accesibilidad.
- **CSS** (para definir **CÓMO** queremos que se vean dichos elementos)
  - Propiedades: `color`, `text-align`, `font-size`, `font-family`, `margin`, `padding`, etc.
  - Selectores: *element*, *id*, *class*, etc. – Jerarquía de selectores.
  - CSS dentro de HTML vs. CSS en un archivo externo.
  - Page Layout (ver próxima diapositiva).



# Page layout – Estructura de una página

En la última clase se vieron 3 propiedades de CSS que se pueden usar para estructurar una página: `position`, `display` y `float`.

⚠ Si bien dichas propiedades son útiles, debido a su dificultad de uso para armar *layouts*, han aparecido nuevas funcionalidades en CSS como [Flexbox](#) y [Grid](#).





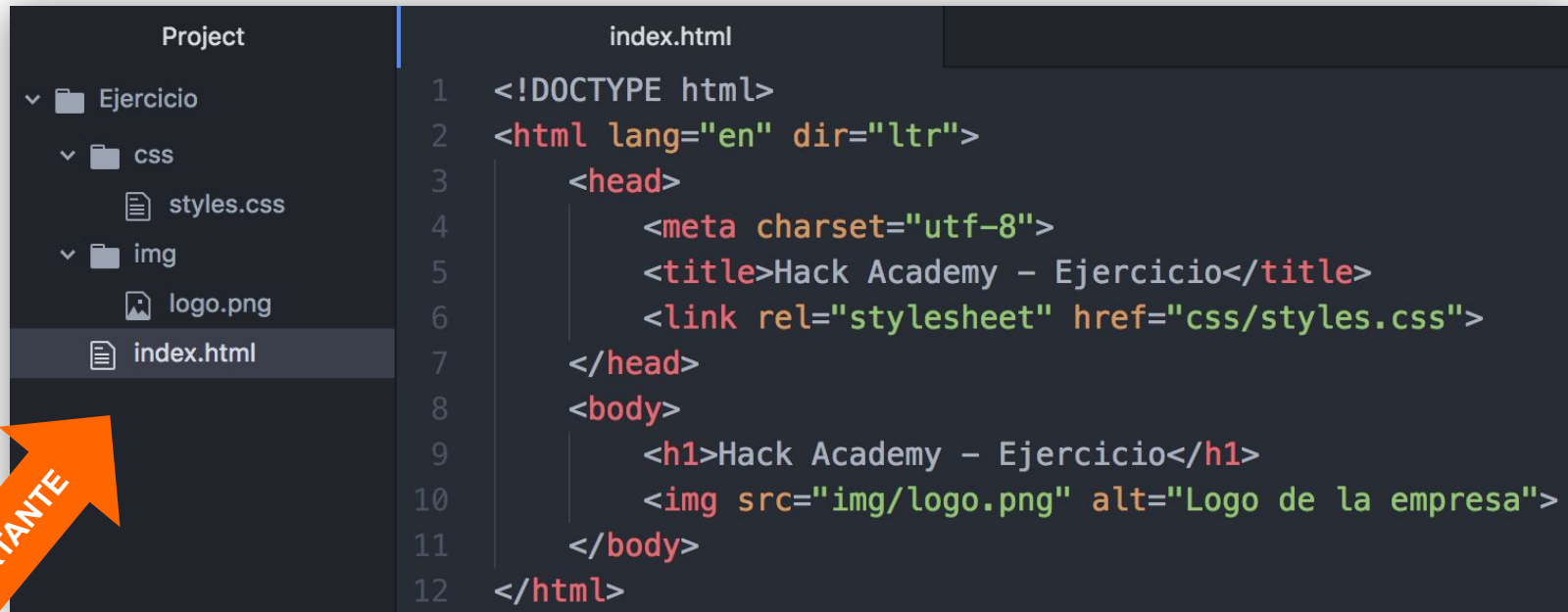
# Page layout – Estructura de una página

Resumen de `position`, `display` y `float`:

Propiedad	¿Qué hace?	Posibles valores
<b>position</b>	<p>Establece el tipo de posicionamiento de un elemento.</p> <p>Más información: <a href="https://css-tricks.com/almanac/properties/p/position/">https://css-tricks.com/almanac/properties/p/position/</a>. Se suele usar junto con las propiedades <code>top</code>, <code>bottom</code>, <code>left</code> y <code>right</code>.</p>	<code>static</code> <code>absolute</code> <code>relative</code> <code>fixed</code>
<b>display</b>	<p>Establece si un elemento debe ser mostrado o no, y en caso afirmativo, cómo se debe mostrar.</p>	<code>none</code> <code>block</code> <code>inline</code> <code>inline-block</code> <code>flex</code> <code>grid</code>
<b>float</b>	<p>Establece si un elemento debe “flotar”, y en caso afirmativo, hacia dónde debe flotar.</p> <p>Más información: <a href="https://css-tricks.com/all-about-floats/">https://css-tricks.com/all-about-floats/</a>.</p>	<code>left</code> <code>right</code> <code>none</code>



# Estructura básica de un proyecto



En general, todos los proyectos (ejercicios) con los que trabajaremos tendrán esta estructura básica. Los nombres de los archivos y carpetas son arbitrarios, pero es una **convención** llamarlos de esta forma y en **minúscula**. Para abrir un proyecto en VSC, ir al menú **File > Open Folder** en Windows o **File > Open** en Mac.





# Flexbox

Otra forma de armar *layouts*.



# Flexbox (1/3)

Flexbox no es una propiedad CSS, sino un **conjunto de propiedades** CSS, para poder **armar el layout** de una página. Es decir, Flexbox permite posicionar elementos en la página, de forma horizontal y vertical.

El primer borrador de la [especificación de Flexbox](#) se publicó en julio 2009. Desde setiembre 2012 hasta el día de hoy (2022), la especificación se encuentra en “*candidate recommendation*”, lo cual significa que si bien no es definitiva, es estable y se encuentra [bastante muy soportada por los navegadores](#).



# Flexbox (2/3)

Flexbox es muy **poderoso**, pero también es **bastante complejo**.

Tiene tantas opciones de configuración, que se podría hacer un curso sólo de Flexbox. De hecho, hay uno muy bueno y gratuito hecho por Wes Bos:

<https://flexbox.io>. En el curso de Front-End nos limitaremos a hacer una breve introducción al tema.

Documentación: [MDN](#) / [CSS Tricks](#).

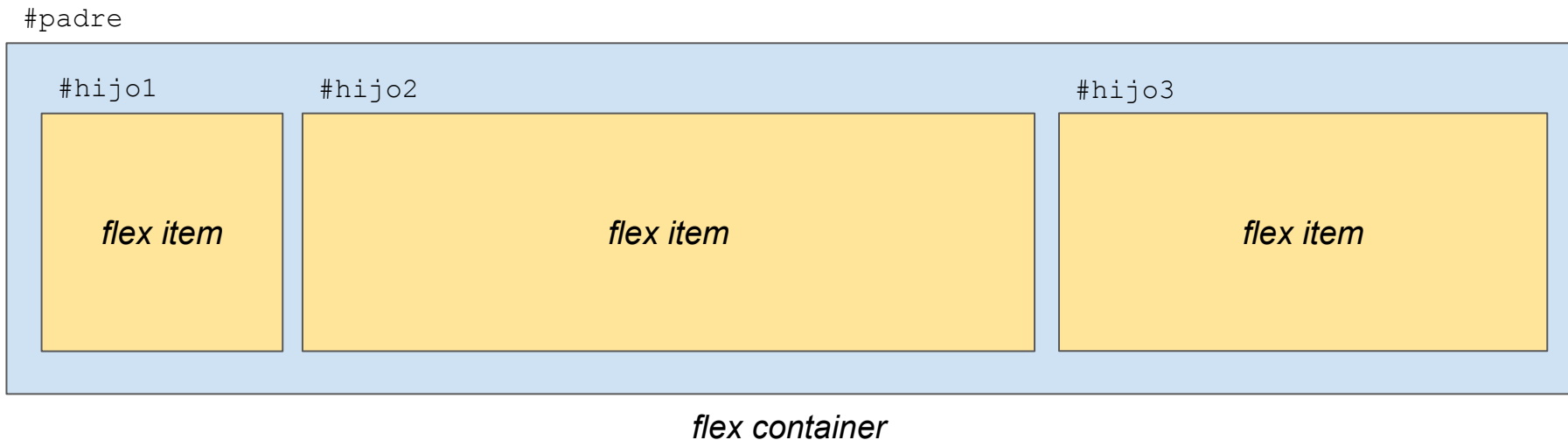
👉 Podemos hacer los primeros niveles entre todos, en clase.

👉 Juego muy bueno para aprender sobre el tema:  [FlexboxFroggy](#).



# Flexbox (3/3)

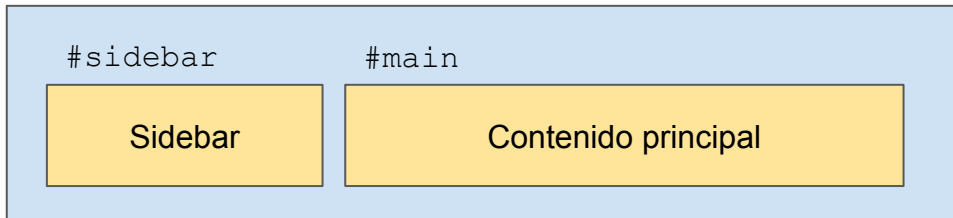
La idea principal de Flexbox es: dado un elemento *padre* (al cual se lo llamará *flex container*) posicionar sus elementos *hijos* (a los cuales se los llamará *flex items*).





# Flexbox – Ejemplo 1

#container



En este ejemplo, se establece que el *flex container* se comporte como una *row*, y por lo tanto, los *flex items* se comportan como columnas. La primera de ancho  $\frac{1}{3}$  y la segunda  $\frac{2}{3}$ .

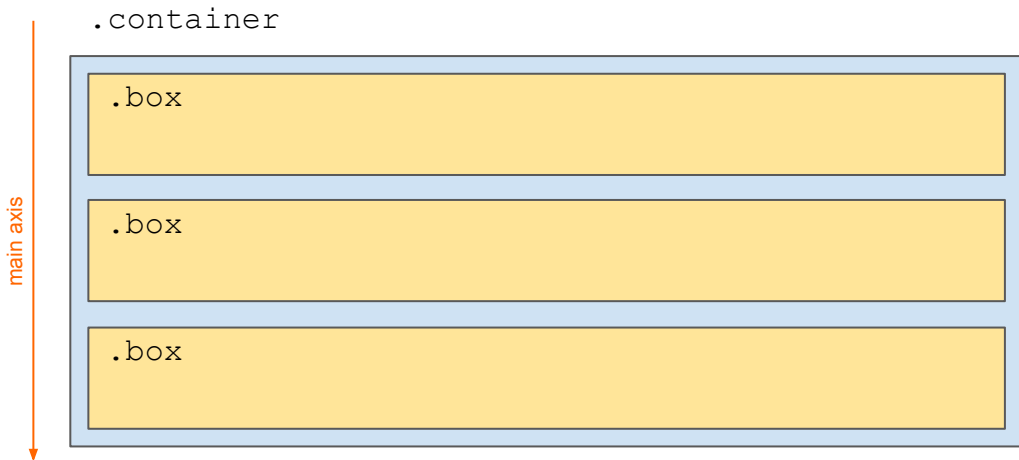
```
#container {  
  display: flex;  
  flex-direction: row; /* Default */  
}  
  
#sidebar {  
  flex: 1;  
}  
  
#main {  
  flex: 2;  
}
```



# Flexbox – Flex Direction

Por defecto, los *flex items* se posicionan en fila. En este caso, se dice que el **eje principal** (*main axis*) es el horizontal. Ver el ejemplo de la diapositiva anterior.

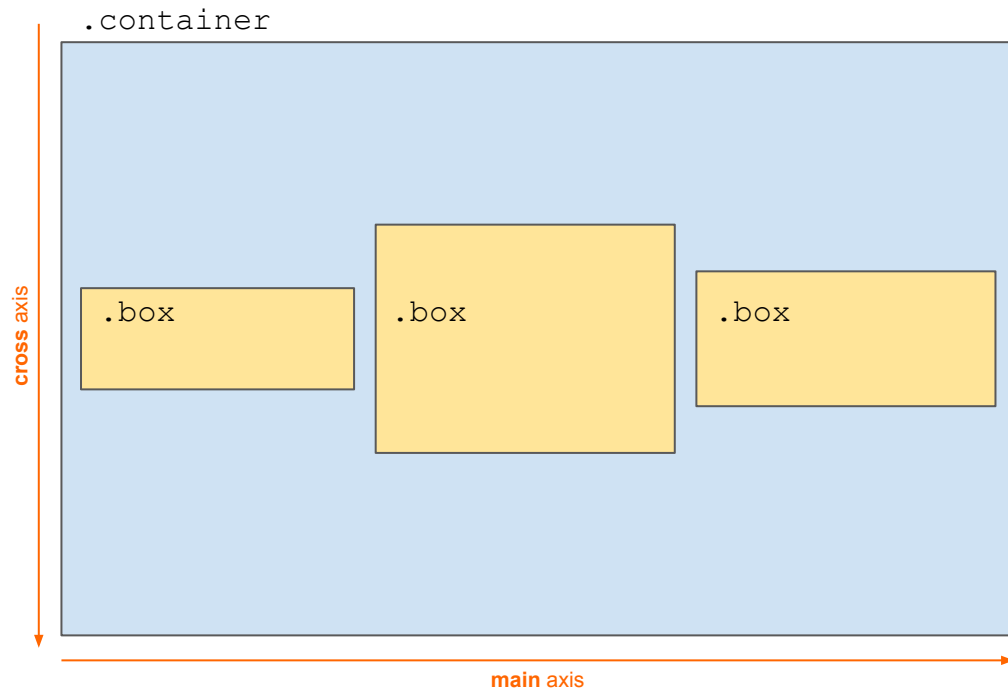
Sin embargo, Flexbox permite cambiar el eje principal (hacerlo vertical) y posicionar los *flex items* en columna, usando la propiedad `flex-direction`.



```
.container {  
  
  display: flex;  
  
  flex-direction: column;  
  
}
```

# Flexbox – Alineación vertical (1/2)

Alinear elementos de forma vertical siempre fue algo complicado en CSS, sobre todo si las alturas varían. Gracias a Flexbox, este problema se resuelve muy fácilmente.



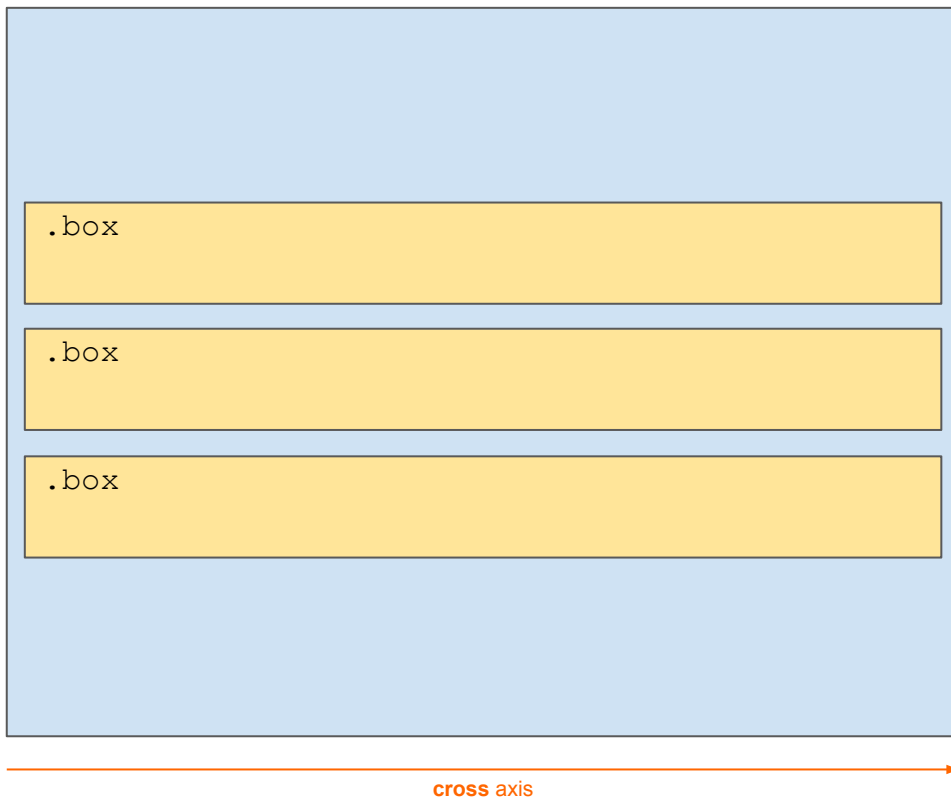
```
.container {  
  height: 800px;  
  display: flex;  
  flex-direction: row; /* Default */  
  align-items: center;  
}
```

**Nota 1:** La propiedad `flex-direction` en este caso no es necesaria, ya que `row` es el valor por defecto.

**Nota 2:** La propiedad `align-items` se usa para alinear los elementos en el `cross axis`, que podría ser el eje vertical u horizontal, dependiendo de cómo esté configurado el `flex-direction`.

# Flexbox – Alineación vertical (2/2)

.container



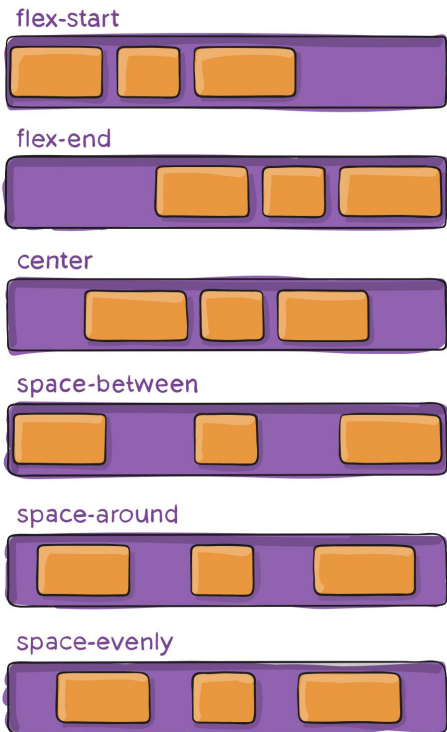
```
.container {
  height: 100vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

`vh` es una [unidad de medida](#) bastante nueva en CSS que permite definir alturas en relación al alto del viewport.

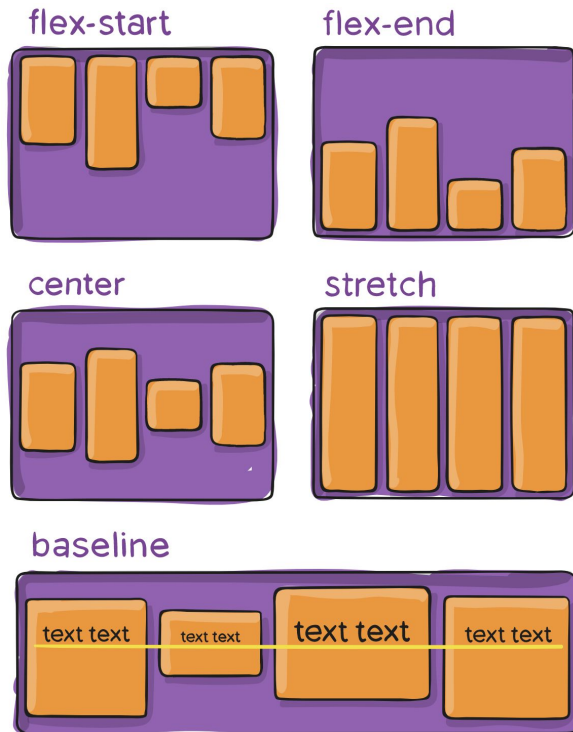


# Flexbox – Justify Content & Align Items

Valores para `justify-content`:



Valores para `align-items`:





# Ejercicio 1

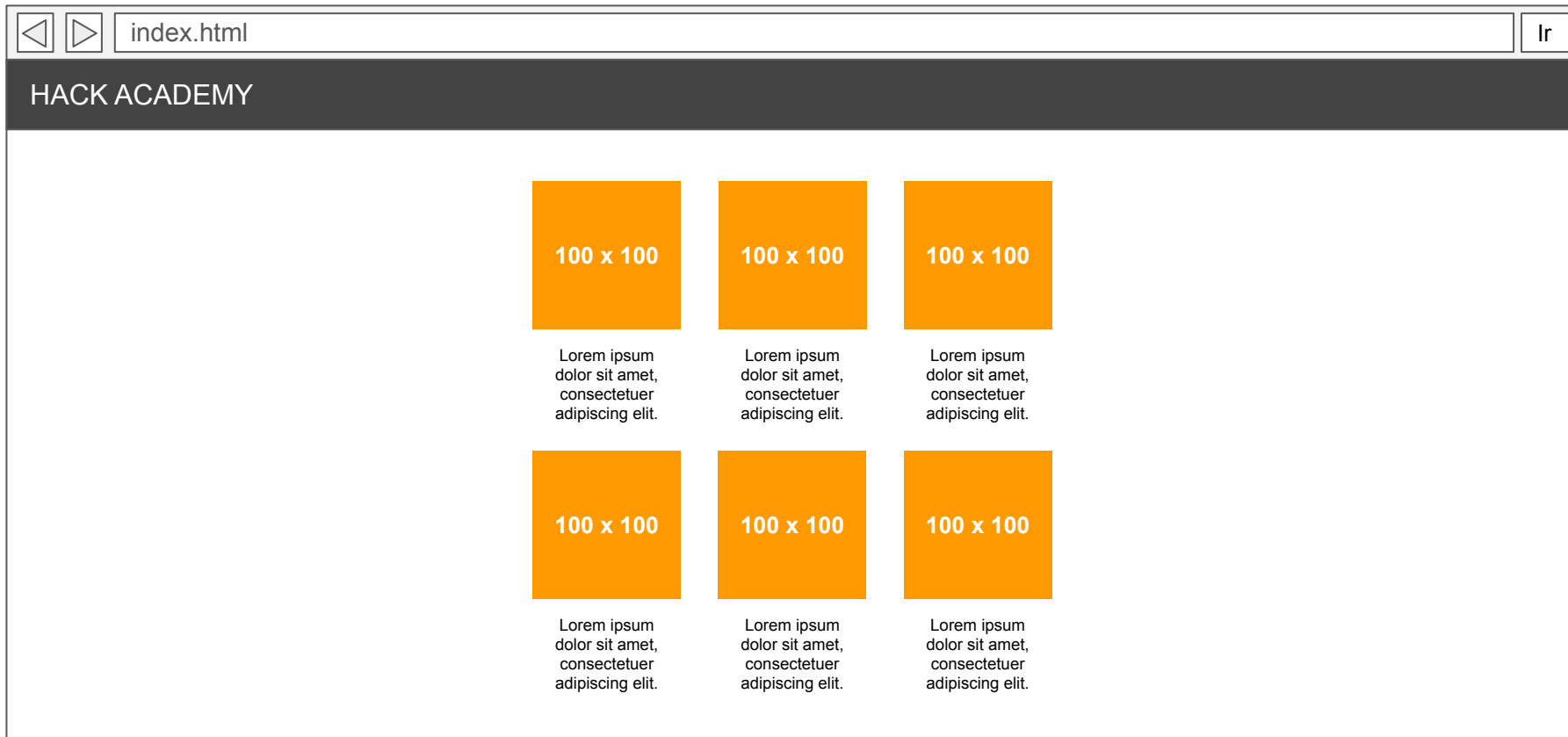


# Ejercicio 1

1. Crear una carpeta en el Escritorio (o donde prefieran) con el nombre `Clase05_Ejercicio_1`.
2. Abrir dicha carpeta en **Visual Studio Code**.  
Esto se puede hacer yendo al menú: `File > Open Folder` en Windows o `File > Open` en Mac.
3. Desde VSC, crear un archivo llamado `index.html` dentro de la carpeta.
4. Desde VSC, crear una carpeta `css` y dentro de la misma el archivo `styles.css`.
5. Escribir el HTML y CSS necesario para lograr un resultado similar al siguiente diagrama.

👉 Este ejercicio es igual al de la clase pasada, pero la diferencia es que en lugar de hacerlo con *floats*, hay que hacerlo con **Flexbox**.

# Ejercicio 1 (cont)





En resumen:

*“Armar el layout de una página usando las propiedades `position`, `display` y `float` no es fácil. Flexbox (y Grid) es poderoso, pero también puede ser complejo de utilizar.”*

¡Por suerte existen **frameworks CSS!**



# Frameworks



# ¿Qué es un Framework?

Un framework es un **conjunto de código pre-escrito** por alguien más, que se puede reutilizar para acelerar nuestro trabajo.

Generalmente, en el framework se **resuelven problemas comunes** con los que se enfrentan los desarrolladores día a día.

Es importante **saber cuándo utilizar** un framework y cuándo no.

## **Framework ≠ Library.**

Una librería también es un conjunto de código reutilizable.

La diferencia está en el "control" que tiene el programador sobre su código. Al utilizar un framework, el programador pierde control porque el framework "le exige" que haga las cosas de determinada manera. Al usar una librería, el programador tiene más control sobre qué usar y qué no, y sobre cómo quiere que funcione su aplicación.



# Bootstrap





# Frameworks CSS – Bootstrap

Existen varios frameworks de CSS.

El más popular y el que vamos a usar en el curso es **Bootstrap**:

- Creado por empleados de Twitter en **2011**.
- Es **open-source**.
- Es un framework CSS (aunque también trae algunas utilidades JavaScript).
- Sólo para la parte de **CSS** contiene aprox. [11.000 líneas de código](#)!
- Documentación: <http://getbootstrap.com>.



# Bootstrap – Versiones

El diciembre de 2020 salió la **versión 5** de Bootstrap y es la que utilizaremos en este curso.

La versión 5 **no soporta Internet Explorer**.



Si precisan compatibilidad con navegadores más antiguos, puede usar la versión 4 que funciona a partir de IE10 o la versión 3 que funciona a partir de IE8.



*“Cuando consulten una documentación,  
verifiquen qué versión están consultando”*

Documentación de Bootstrap: <https://getbootstrap.com/docs>.



# Bootstrap – Ventajas (1/2)

- **Grid System:** Funcionalidad que permite estructurar una página (armar el *layout*) de una forma muy sencilla (sin tener que usar propiedades CSS complicadas).
- **CSS Reset/Reboot:** Funcionalidad que elimina (*resetea*) todos los estilos que los navegadores *setean* por defecto. Por ejemplo: se *setean* los márgenes del `body` en 0 (cero).
- **Responsive Design:** Funcionalidad que ajusta de forma automática el diseño de nuestras páginas según el tamaño de pantalla del dispositivo utilizado.
- **Facilidad de uso:** Bootstrap es un framework muy sencillo de utilizar.





# Bootstrap – Ventajas (2/2)

- **Mobile first:** Bootstrap promueve que las páginas se diseñen pensando, en primera instancia, en dispositivos móviles. Es decir, al diseñar un sitio lo primero que deberíamos preguntarnos es cómo se ve en un celular y luego preguntarnos cómo se ve en un notebook/desktop.
- **Comunidad de usuarios:** Bootstrap cuenta con una gran comunidad de usuarios que están continuamente mejorando el framework. Al ser tan popular, facilita el trabajo con otros desarrolladores ya que probablemente todos hayan usado Bootstrap en algún momento.
- **Documentación:** Es clara, fácil de consultar y tiene buenos ejemplos.
- **Estilos y componentes:** Bootstrap trae un montón de estilos y componentes pre-hechos que permiten una rápida prototipación.



# Bootstrap – Desventajas

- Bootstrap puede ser muy **pesado** para un sitio simple y pequeño.  
La parte de CSS pesa 194 kB y la parte de JavaScript pesa  $\approx 80$  kB. Esto significa que al comenzar a usar Bootstrap, nuestro sitio ya pesa  $\approx 274$  kB aún sin haber escrito ni una sola línea de código.
- Sitios web creados usando Bootstrap *pueden* quedar demasiado **parecidos entre sí** y por lo tanto “no destacarse”.



# Bootstrap – Resumen de Ventajas y Desventajas

Ventajas	Desventajas
<ul style="list-style-type: none"><li>● <b>Grid System.</b></li><li>● CSS reset.</li><li>● Responsive design.</li><li>● Facilidad de uso.</li><li>● Mobile first.</li><li>● Comunidad de usuarios.</li><li>● Documentación.</li><li>● Estilos y componentes.</li></ul>	<ul style="list-style-type: none"><li>● Puede ser muy “pesado” para un proyecto chico.</li><li>● Sitios web creados usando Bootstrap pueden quedar demasiado parecidos entre sí y por lo tanto “no destacarse”.</li></ul>



## Ver ejemplos de Bootstrap

Notar que estos ejemplos se construyeron prácticamente sin necesidad de escribir código CSS.

[Ver otros ejemplos.](#)





# Bootstrap – Instalación



# Bootstrap – Instalación

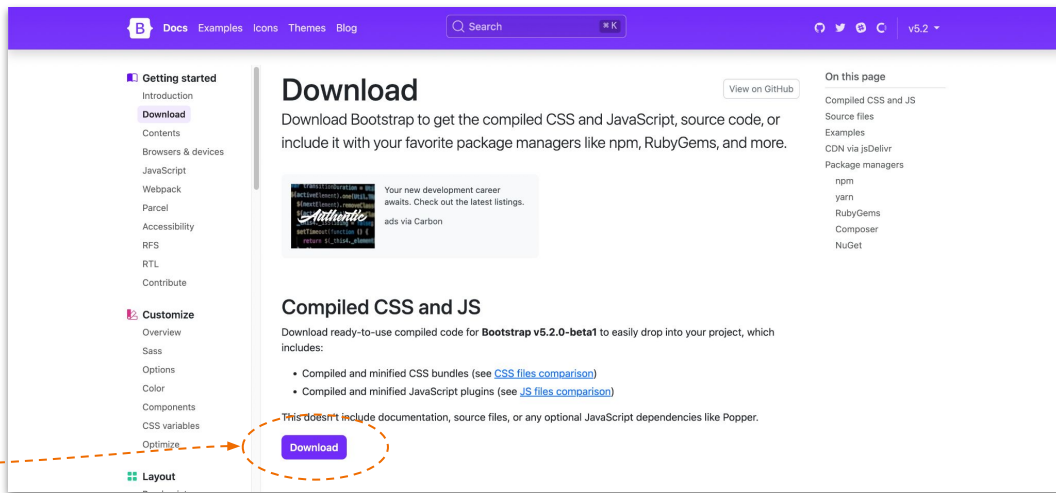
## Método 1

(No lo usaremos en el curso)



# Bootstrap – Instalación – Método 1

1. Descargar ZIP de: <https://getbootstrap.com/docs/5.2/getting-started/download/>



2. Descomprimir el ZIP y colocar el archivo `bootstrap.min.css` en la carpeta `css` de nuestro proyecto.



# Bootstrap – Instalación – Método 1 (cont)

3. Linkear el archivo `bootstrap.min.css` desde nuestra página HTML.

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  ...
  <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
  ...
</head>
```

Archivo de Bootstrap instalado en carpeta `css` del proyecto. Es una ruta relativa.



# Bootstrap – Instalación

## Método 2

(El que usaremos en el curso)



# Bootstrap – Instalación – Método 2

1. Ingresar a: <https://getbootstrap.com/docs/5.2/getting-started/introduction/#quick-start>.
2. Copiar esta línea de código, la que dice `<link>`, y pegarla en el `<head>` de la página HTML.

The screenshot shows the Bootstrap 5.2 documentation page. The left sidebar has a 'Getting started' section with 'Introduction' selected. The main content area shows the '2. Include Bootstrap's CSS and JS' section. The code example for the HTML boilerplate is displayed, with the `<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap" />` line highlighted by an orange box. A dashed orange line originates from the instruction in step 2 of the list and points to this highlighted line. The right sidebar lists 'On this page' with links like 'Quick start', 'CDN links', etc.

```
<!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@5.2.0-beta1/dist/css/bootstrap" />
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap" />
  </body>
</html>
```

También se puede copiar todo el HTML y usarlo como "starter template".



# Bootstrap – Instalación – Método 2 (cont)

Al finalizar, les debería quedar algo así. También pueden ver un “starter template” [aquí](#).

```
<head>

  <meta charset="utf-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

  ...

  <link

    href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css"

    rel="stylesheet"

    integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNL/vI1Bx"

    crossorigin="anonymous"

  />

  ...

</head>
```



# Bootstrap – Instalación – Método 1 vs. Método 2

## Método 1 - Alojarse archivo CSS de Bootstrap en nuestra carpeta css.

- Lleva más tiempo de instalarse (hay que bajar archivos y colocarlos en la carpeta correcta). 😞
- Nuestro sitio no depende de un sitio externo para funcionar. 😊
- Nuestro sitio no precisa de Internet para funcionar. 😊

## Método 2 - Linkear a archivo CSS de Bootstrap en servidor externo (CDN).

- Es más rápido de instalar (no hay que descargar nada, sólo copiar y pegar una línea de código). 😊
- Si el servidor externo se cae, nuestro sitio no se verá bien. 😞
- El sitio precisa de Internet para funcionar. 😞
- Suele cargarse más rápido. 😊





# Bootstrap – Instalación (Responsive Design)

Para indicarle al browser cómo debe comportarse según el dispositivo es necesario agregar la siguiente línea de código en el `head` de la página. Agregarla siempre que se use Bootstrap y se quiera tener un sitio responsive.

```
<head>

...

<meta name="viewport" content="width=device-width, initial-scale=1">

...

</head>
```

Documentación: [https://developer.mozilla.org/en/docs/Mozilla/Mobile/Viewport\\_meta\\_tag](https://developer.mozilla.org/en/docs/Mozilla/Mobile/Viewport_meta_tag).



# Ejercicio 2

Instalación de Bootstrap



# Ejercicio 2 – Instalación de Bootstrap

1. Crear una carpeta en el Escritorio (o donde prefieran) con el nombre `Clase05_Ejercicio2`.
2. Abrir dicha carpeta en **Visual Studio Code**.  
Esto se puede hacer yendo al menú: `File > Open Folder` en Windows o `File > Open` en Mac.
3. Desde VSC, crear un archivo llamado `index.html` dentro de la carpeta.
4. Desde VSC, crear una carpeta `css` y dentro de la misma el archivo `styles.css`.
5. Instalar Bootstrap usando el “Método 2” visto anteriormente.
6. **“Seguir” al docente** durante las próximas diapositivas.



Unos pequeños comentarios sobre

# Responsive Design

# Responsive Design (1/2)

@media una funcionalidad de CSS.  
Es decir, es independiente de Bootstrap.



Si se inspeccionan los archivos CSS de Bootstrap, se verán códigos como este:

```
@media (min-width: 992px) {  
  .d-lg-none {  
    display: none !important;  
  }  
}
```

@media es una regla CSS que permite definir **estilos CSS específicos** para **ciertos tamaños de pantalla**. En este ejemplo se están definiendo estilos que sólo aplican para pantallas que tengan 992px de ancho o más, es decir, pantallas grandes (desktop).

Documentación: [https://developer.mozilla.org/en-US/docs/Web/CSS/Media\\_Queries/Using\\_media\\_queries](https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries).



# Responsive Design (2/2)

Otro ejemplo:

```
@media (min-width: 992px) and (max-width: 1199px) {  
    h1 {  
        font-size: 3rem;  
    }  
}
```

Aquí se le aplica al `h1` un tamaño de `3rem` sólo para cuando el ancho de la pantalla esté entre 992 y 1199px. Es lo que se Bootstrap considera pantallas medianas.



# Bootstrap – Grid System

# Bootstrap – Grid System (1/10)

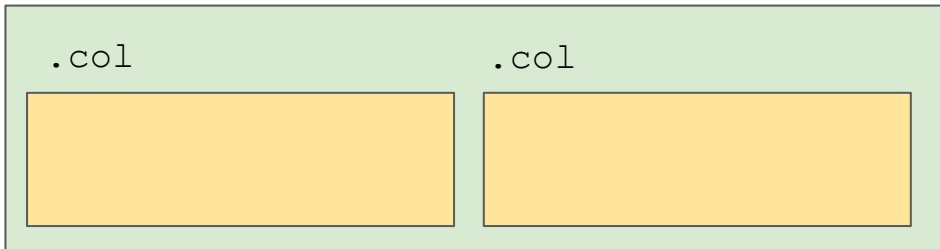
Para crear el grid, Bootstrap v4 usa flexbox. Bootstrap v3 usa float. Para nosotros es “transparente”.



El [Grid System de Bootstrap](#) es un sistema de filas y columnas que permite armar el layout de nuestra página de una **forma muy sencilla**.

Las filas y columnas no son más que `divs` con ciertos estilos CSS.

`.row`



**Importante:** Las clases `.row` y `.col` las provee Bootstrap. No las tenemos que crear nosotros.

```
<div class="row">
  <div class="col">
    ...
  </div>
  <div class="col">
    ...
  </div>
</div>
```

Notar que para lograr este layout no fue necesario escribir ni una sola línea de CSS.



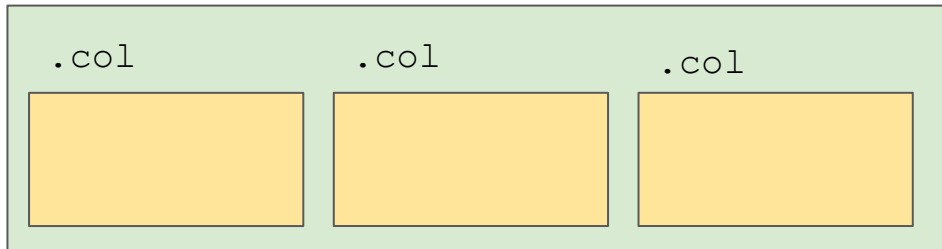


# Bootstrap – Grid System (2/10)

Para **agregar una columna**, sólo hace falta agregar otro elemento `<div class="col">`.

Se pueden agregar tantas columnas como se quiera.

`.row`



**Importante:** Las clases `.row` y `.col` las provee Bootstrap. No las tenemos que crear nosotros.

```
<div class="row">
  <div class="col">
    ...
  </div>
  <div class="col">
    ...
  </div>
  <div class="col">
    ...
  </div>
</div>
```

Notar que para lograr este layout no fue necesario escribir ni una sola línea de CSS.

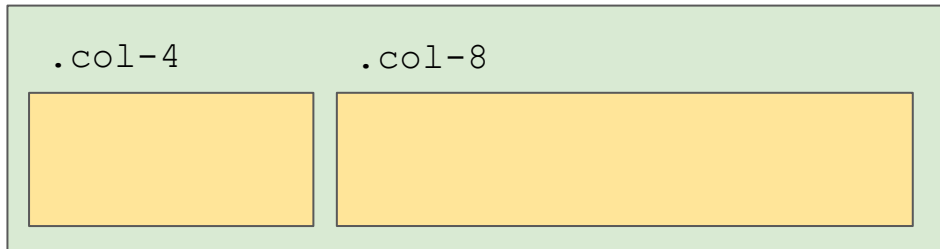


# Bootstrap – Grid System (3/10)

Bootstrap también permite **definir el ancho** de una **columna**.

Tener en cuenta que la suma de los anchos no puede superar **12**.

`.row`



**Importante:** Las clases `.row` y `.col-*` las provee Bootstrap. No las tenemos que crear nosotros.

```
<div class="row">
  <div class="col-4">
    ...
  </div>
  <div class="col-8">
    ...
  </div>
</div>
```

Notar que para lograr este layout no fue necesario escribir ni una sola línea de CSS.



# Bootstrap – Grid System (4/10) – Reglas

```
<div class="row">

  <div class="col">
    ...
  </div>

  <div class="col">
    ...
  </div>

</div>
```

- Toda columna debe ir inmediatamente adentro de una fila. Adentro de una fila, sólo se pueden colocar columnas.
- **Nuestro contenido** se agrega **dentro de las columnas** (jamás dentro de las filas).
- Dentro de las columnas incluso se puede agregar otros `<div>`.

IMPORTANTE



# Bootstrap – Grid System (5/10) – Reglas

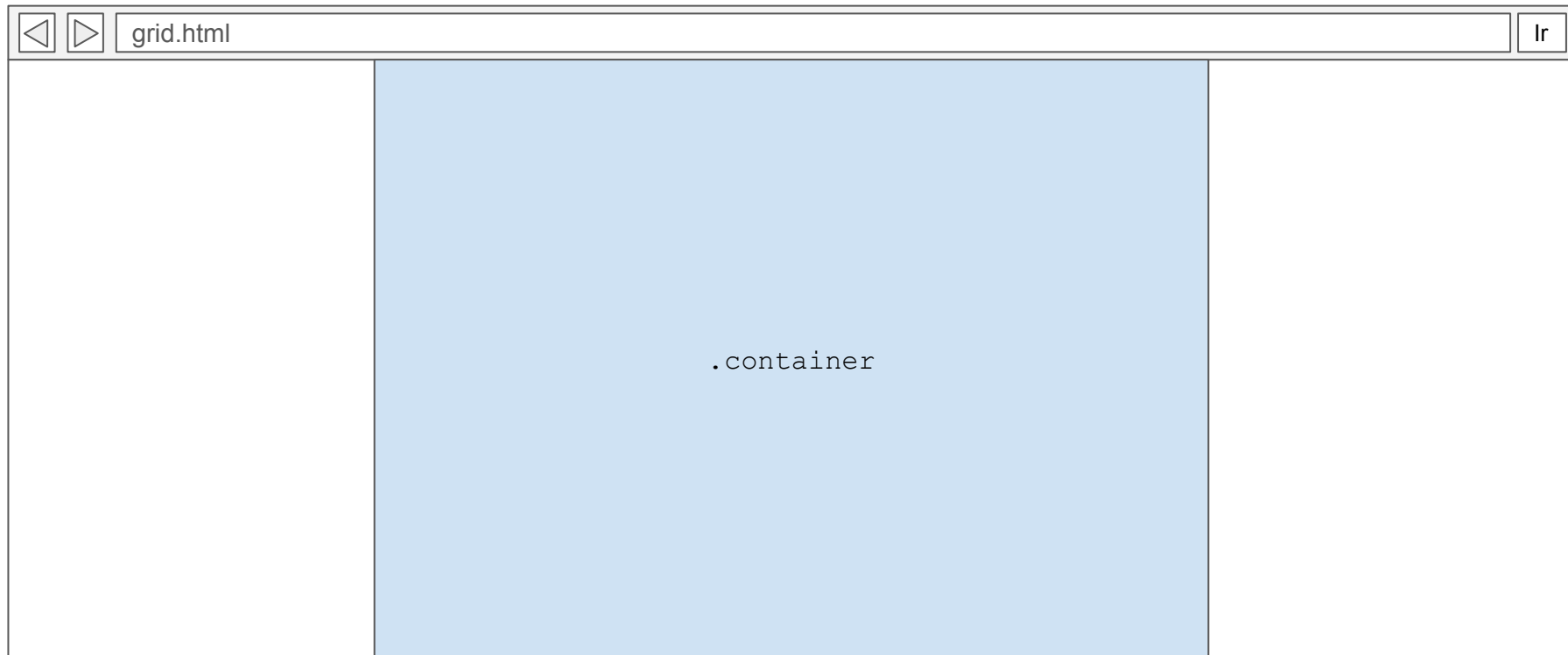
Otras reglas:

- Cuando utilicen columnas con anchos predefinidos, recuerden que la suma del anchos de las columnas de una fila debe ser menor a 12.
- En general, evitar cambiar los estilos de las filas y columnas (no les cambien el `float`, `position`, `display`, `margin` ni `padding`).
- En caso de necesitar layouts más complejos, es posible agregar filas adentro de columnas. Esto se conoce como anidación o [nesting](#).



# Bootstrap – Grid System (6/10) – Containers

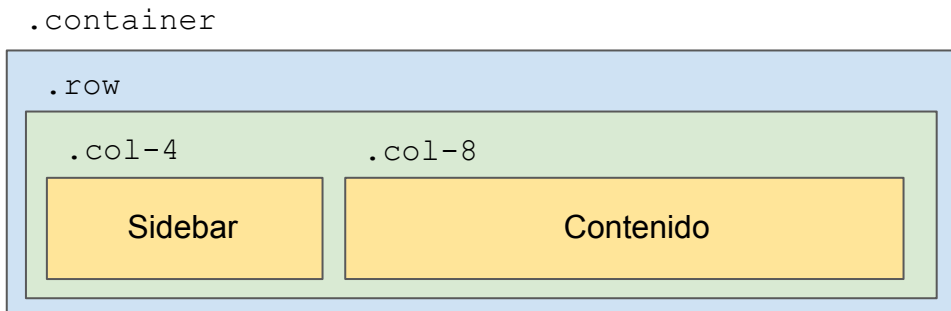
Documentación: <https://getbootstrap.com/docs/5.2/layout/containers/>.





# Bootstrap – Grid System (7/10) – Containers

Para que las **filas** no toquen los bordes de la página y para que queden **centradas** en la misma, es común colocarlas inmediatamente adentro de divs con clase `.container` (ancho fijo) ó `.container-fluid` (ancho 100%).

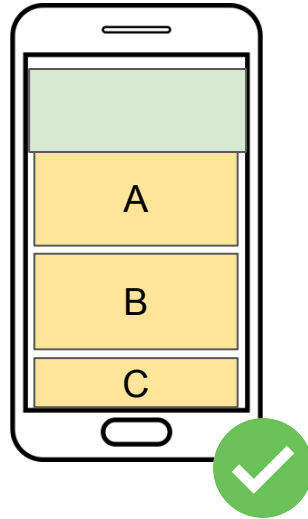
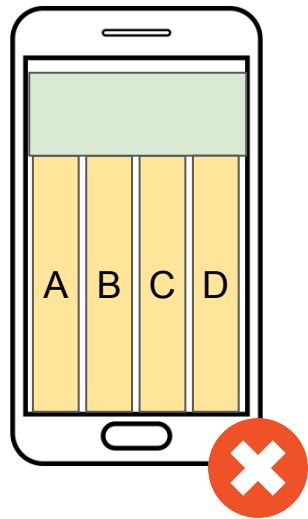
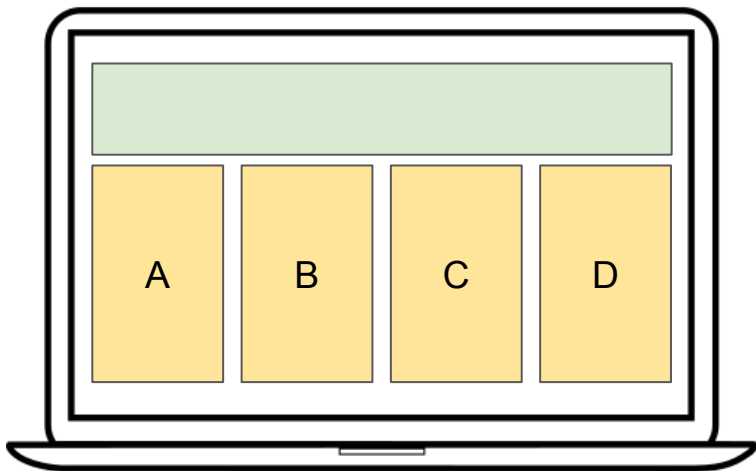


```
<div class="container">
  <div class="row">
    <div class="col-4">
      Sidebar
    </div>
    <div class="col-8">
      Contenido
    </div>
  </div>
</div>
```



# Bootstrap – Grid System (8/10) – Columnas Responsive

No siempre se querrá que las columnas se vean tanto en **mobile** como en **desktop**. A veces se querrá modificar el comportamiento de las mismas según el tamaño de la pantalla. Por ejemplo, 4 columnas en mobile suelen quedar demasiado apretadas y suele ser mejor mostrarlas apiladas (una sobre la otra).





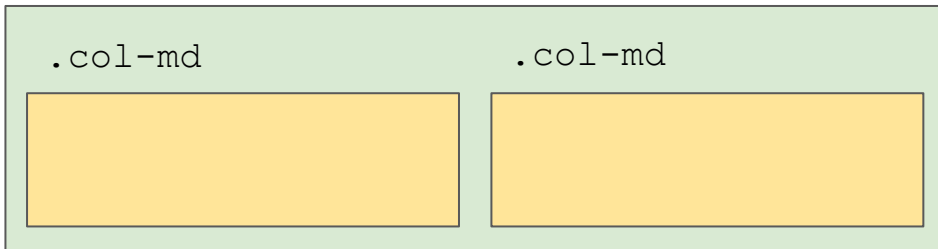
# Bootstrap – Grid System (9/10) – Columnas Responsive

Usando los tamaños `sm`, `md`, `lg`, `xl` y `xxl`, se le puede indicar a las columnas a partir de qué tamaño de pantalla deberán “funcionar”.

En el siguiente ejemplo, las dos columnas sólo se verán a partir de pantallas medium en adelante, es decir, mayores o iguales a 768px.

Para pantallas más chicas, los `<div>` se verán apilados.

`.row`



```
<div class="row">
  <div class="col-md">...</div>
  <div class="col-md">...</div>
</div>
```





# Bootstrap – Grid System (10/10) – Responsive breakpoints

```
/* X-Small devices (portrait phones, less than 576px)  
No media query for `xs` 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) { ... }
```

```
/* X-Large devices (large desktops, 1200px and up) */  
@media (min-width: 1200px) { ... }
```

```
/* XX-Large devices (larger desktops, 1400px and up) */  
@media (min-width: 1400px) { ... }
```



# Bootstrap – Botones



# Bootstrap – Botones

Botón Azul

Botón Verde

Botón Rojo

Notar que no fue necesario escribir ni una sola línea de CSS.


```
<button type="button" class="btn btn-primary">Botón Azul</button>  
<button type="button" class="btn btn-success">Botón Verde</button>  
<button type="button" class="btn btn-danger">Botón Rojo</button>
```

Las clases `.btn`, `.btn-primary`, `.btn-success` y `.btn-danger` las provee Bootstrap. No las tenemos que crear nosotros.

Documentación: <https://getbootstrap.com/docs/5.2/components/buttons/>.

# Ejercicio 3

 Este es un ejercicio importante que todos deberían poder terminar.

 Probablemente el tiempo de hoy no sea suficiente, por lo que le seguiremos dedicando tiempo durante la próxima clase.



# Ejercicio 3

1. Crear una carpeta en el Escritorio (o donde prefieran) con el nombre `Clase05_Ejercicio3`.
2. Abrir dicha carpeta en **Visual Studio Code**.  
Esto se puede hacer yendo al menú: `File > Open Folder` en Windows o `File > Open` en Mac.
3. Desde VSC, crear un archivo llamado `index.html` dentro de la carpeta.
4. Desde VSC, crear una carpeta `css` y dentro de la misma el archivo `styles.css`.
5. Instalar Bootstrap usando el “Método 2” visto anteriormente.
6. Crear el HTML y CSS para lograr un resultado similar al siguiente diagrama.

# Ejercicio 3 (cont)

La idea es crear una página como la de la derecha.

¡Esto es increíble considerando que recién vamos 5 clases! 🥳🥳🥳

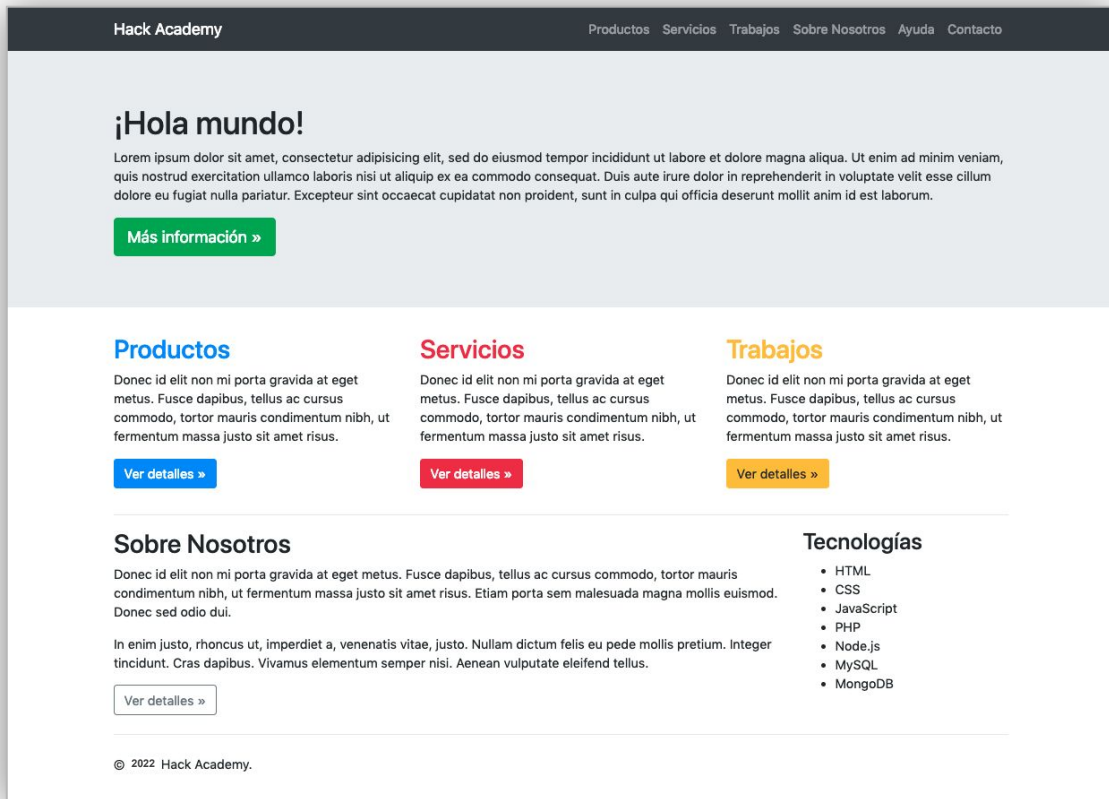
**Tip 1:** Aprovechar Bootstrap al máximo.

⇒ Probablemente no tendrán que escribir CSS.

**Tip 2:** Ir de a poco, por ejemplo, de arriba hacia abajo. Empezar por el *header*.

**Tip 3:** Ver las siguientes *slides*.

Verifiquen que están utilizando la versión de Bootstrap correcta.





# Ejercicio 3 (cont)

## Navbar

Hack Academy

Productos Servicios Trabajos Sobre Nosotros Ayuda Contacto

Bootstrap provee un componente para esto llamado “*navbar*” cuya documentación se puede consultar aquí: <https://getbootstrap.com/docs/5.2/components/navbar/>.

Lamentablemente, el primer ejemplo provisto en la documentación es algo complejo y por eso les dejamos un ejemplo más sencillo en la siguiente *slide*.

Nota: Hasta no dar JavaScript, el *navbar* no funcionará en *mobile*.

## Ejemplo: Navbar

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
    <a class="navbar-brand" href="https://ha.dev">Hack Academy</a>
    <button
      class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarLinks"
      aria-controls="navbarLinks" aria-expanded="false" aria-label="Toggle navigation"
    >
      <span class="navbar-toggler-icon"></span>
    </button>
    <div
      id="navbarLinks"
      class="collapse navbar-collapse justify-content-end"
    >
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" href="#">Productos</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Servicios</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

⚠ Cuidado al copiar este código del **PDF**.  
Podría contener caracteres extraños.  
También se puede obtener de [aquí](#).





# Ejercicio 4

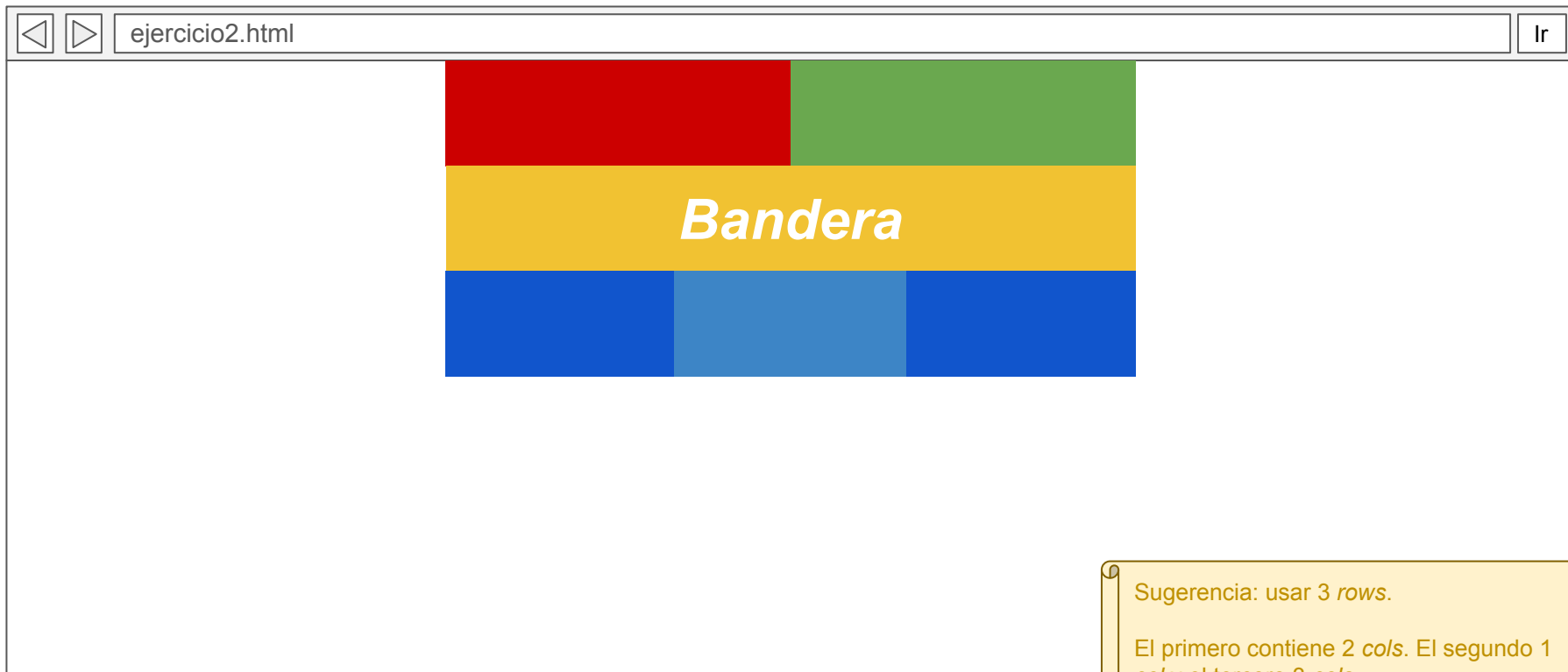


# Ejercicio 4

1. Crear una carpeta en el Escritorio (o donde prefieran) con el nombre `Clase05_Ejercicio4`.
2. Abrir dicha carpeta en **Visual Studio Code**.  
Esto se puede hacer yendo al menú: `File > Open Folder` en Windows o `File > Open` en Mac.
3. Desde VSC, crear un archivo llamado `index.html` dentro de la carpeta.
4. Desde VSC, crear una carpeta `css` y dentro de la misma el archivo `styles.css`.
5. Instalar Bootstrap usando el “Método 2” visto anteriormente.
6. Crear el HTML y CSS para lograr un resultado similar al siguiente diagrama.

# Ejercicio 4

EXTRA



Sugerencia: usar 3 rows.

El primero contiene 2 cols. El segundo 1 col y el tercero 3 cols.