



# **FINEST en Desarrollo Web**

## **Tema 4 - Page Layout**



# Agenda de la clase





# Agenda

- Repaso
- Page Layout:
  - CSS – position
  - CSS – display
  - Extra: CSS – float
- Ejercicios

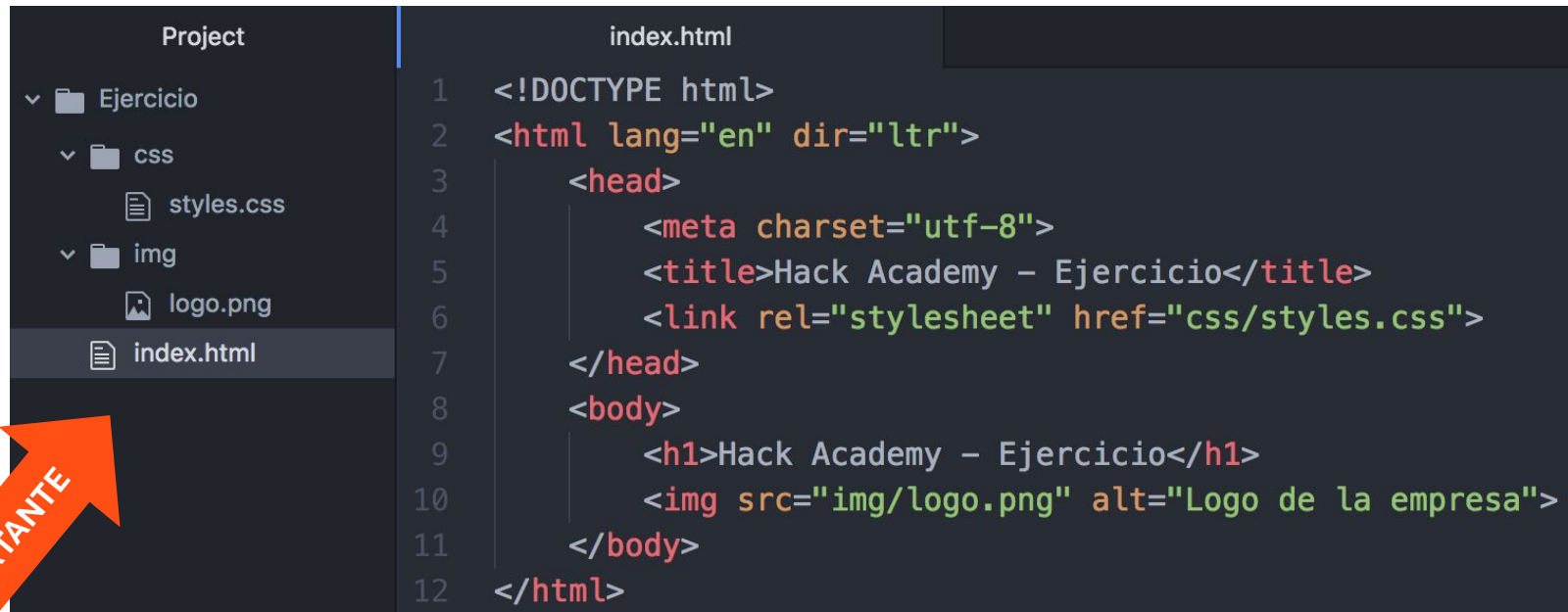


# Repaso





# Estructura básica de un proyecto 🙏



```
Project
├── Ejercicio
│   ├── css
│   │   └── styles.css
│   ├── img
│   │   └── logo.png
│   └── index.html
└── index.html

index.html
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5     <title>Hack Academy - Ejercicio</title>
6     <link rel="stylesheet" href="css/styles.css">
7   </head>
8   <body>
9     <h1>Hack Academy - Ejercicio</h1>
10    
11  </body>
12 </html>
```

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.



# Cosas importantes que hay que recordar

- Usar los **Developer Tools** del navegador.
- Leer **documentación**:
  - MDN: [https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting\\_Started](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_Started)
  - MDN (español): <https://developer.mozilla.org/es/docs/Web/CSS/Introducci%C3%B3n>
  - W3 Schools: <http://www.w3schools.com/css/>
- Usar mucho Google y **Stack Overflow**.
- Usar **indentación** correctamente. ----->
- ¡**Practicar** y **experimentar** mucho!
- ¡Hacer **preguntas**!

El código queda más legible y les simplifica la vida. Además, recuerden que programar suele ser un trabajo de equipo y hay que pensar en las personas que leerán nuestro código. Es poco profesional tener un código desprolijo.



# Links útiles

Aconsejamos mucho que puedan tomarse un tiempo para leer sobre estos temas:

- [id vs class](#)
- [Box Sizing](#) 👉 Particularmente la opción: `box-sizing: border-box;`
- [Collapsing Margins](#)



# CSS Box Model

Un h1, un p, un div, un span, etc., todos son tratados como boxes.

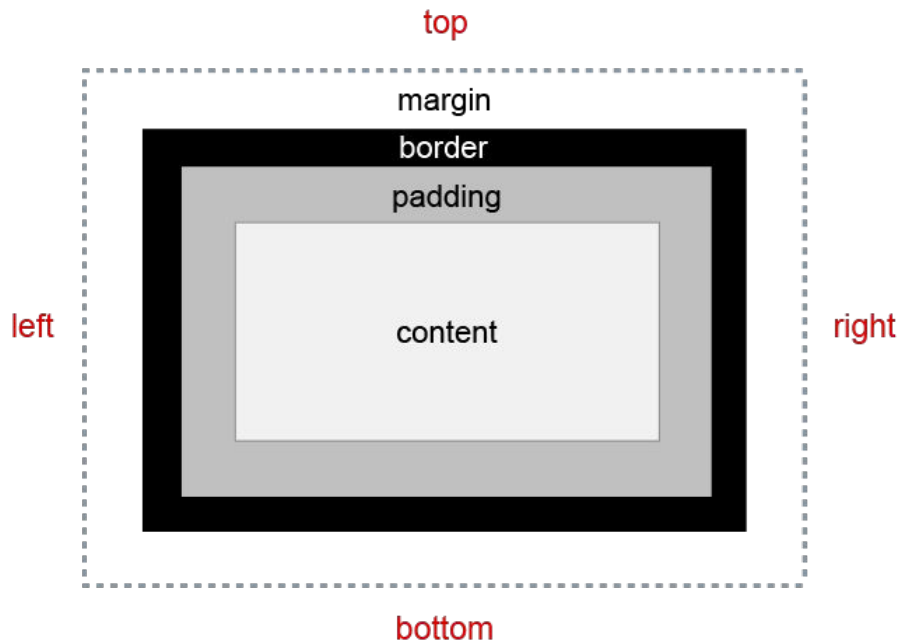
Todos los elementos HTML son tratados como **boxes** (cajas) por el navegador. Pueden ser `inline` o `block` boxes.

Un box consiste en:

- Content.
- Padding.
- Border.
- Margin.

Usar *Chrome Developer Tools* para inspeccionar *boxes*.

👉 Ver [video explicativo](#).

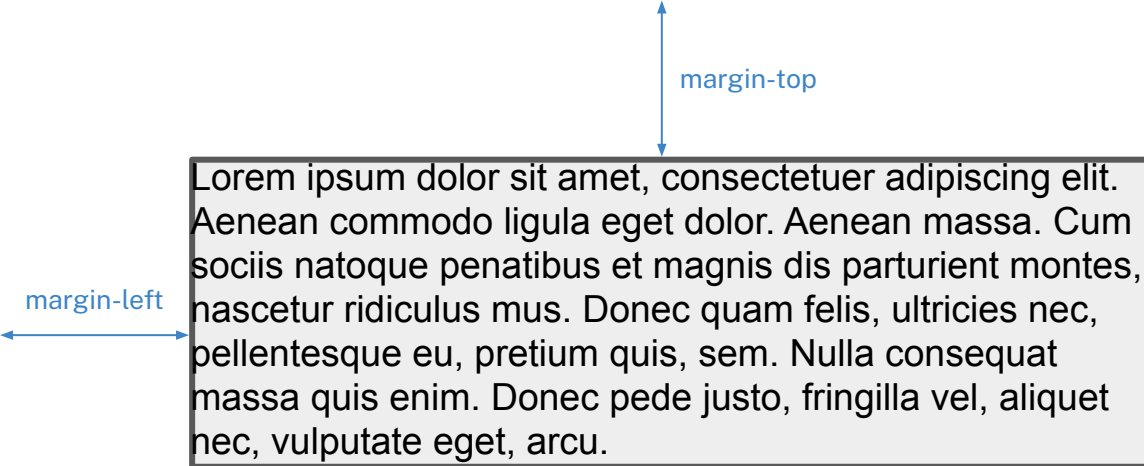




Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

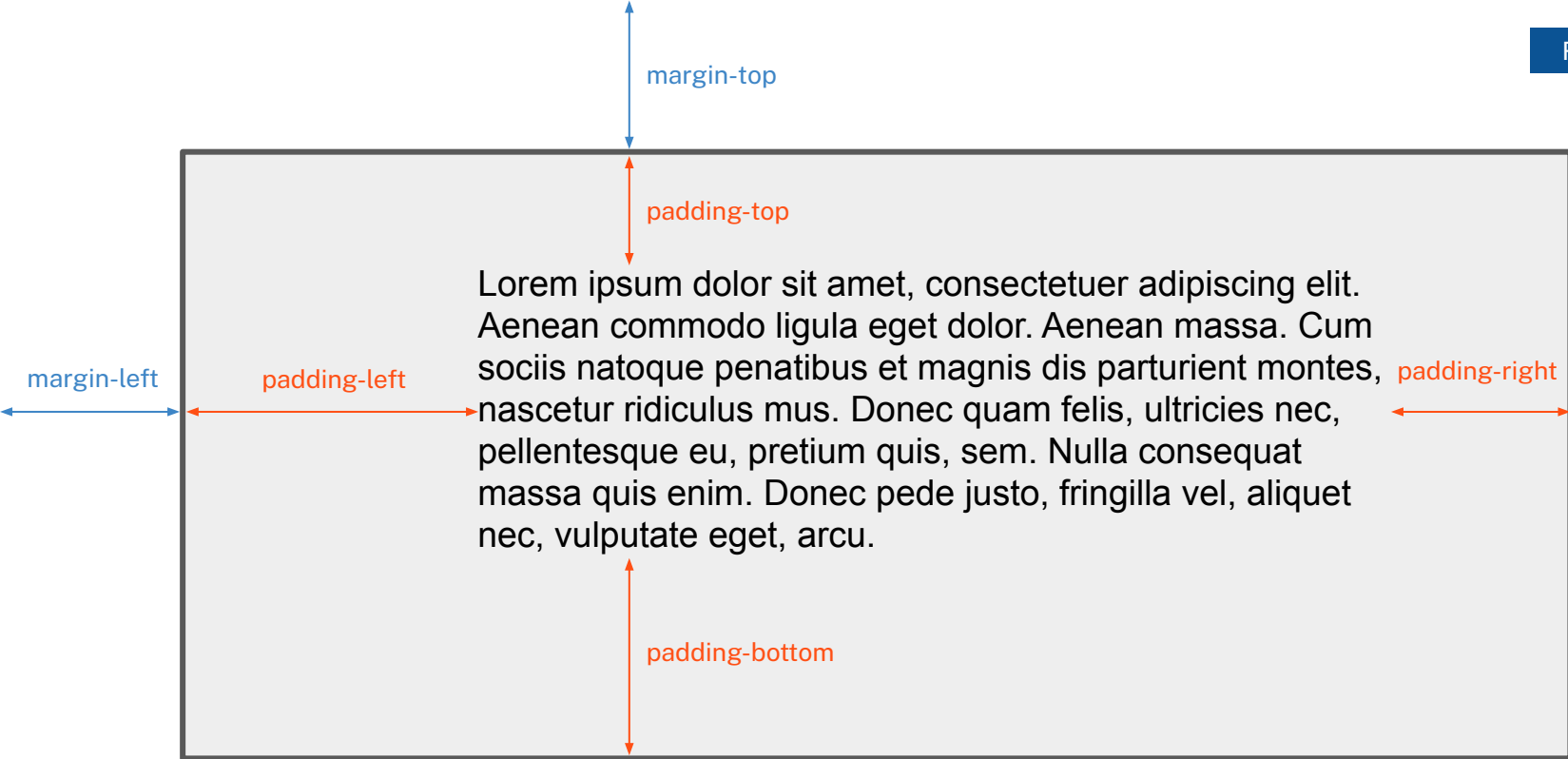


Una caja sin margin ni padding.

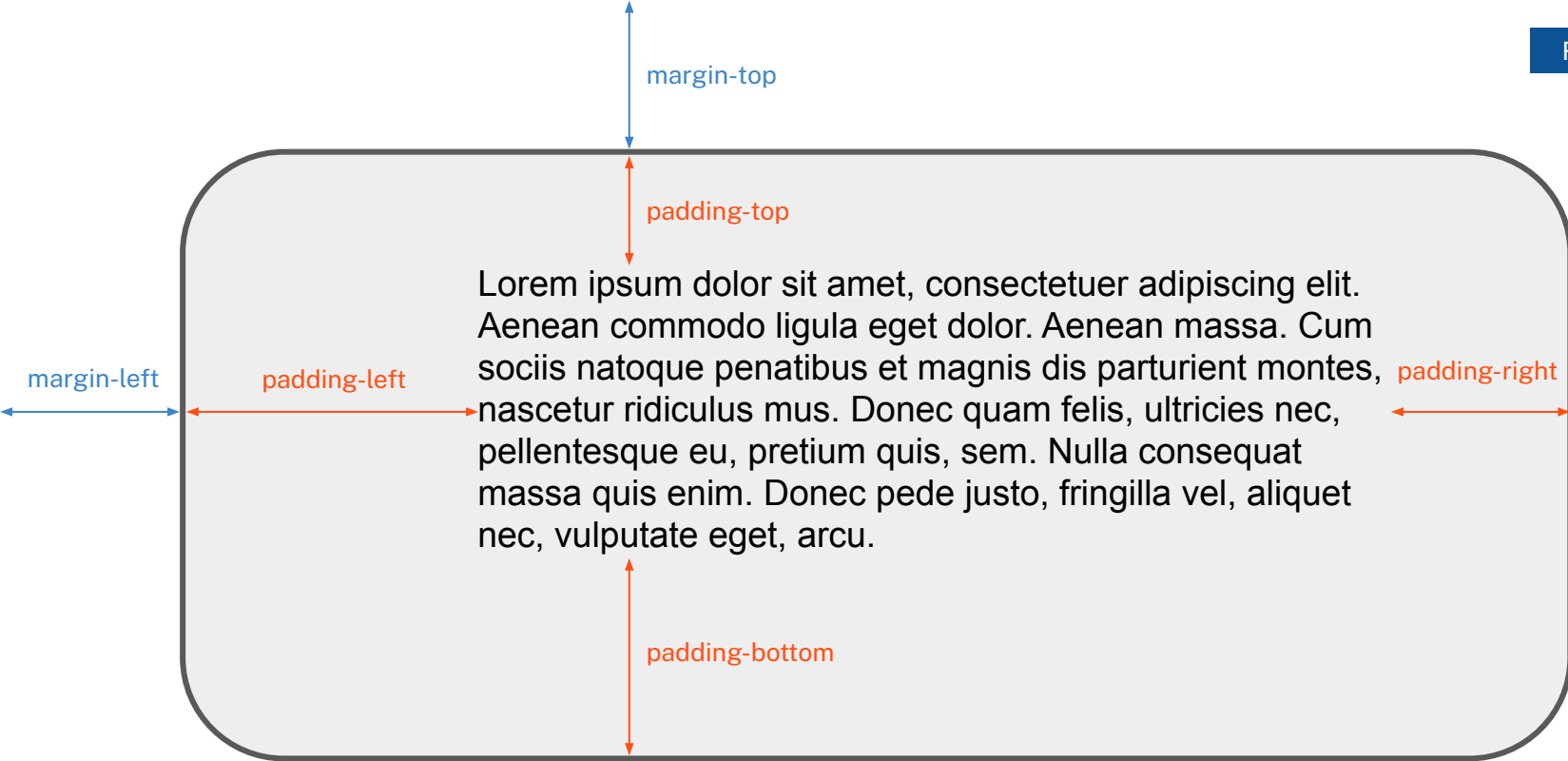


Se agregó margin.

Nota: si llegan a tener algún problema con los márgenes de dos elementos adyacentes, es decir, sobre márgenes que “chocan” entre sí, leer sobre [Collapsing Margins](#).



Se agregó padding.



Se agregó `border-radius`.



# Page Layout

(Estructura de una página)





# Page Layout – Estructura de una página (1/6)

Hasta el momento aprendimos a crear un montón de elementos HTML y aprendimos a darles estilos usando CSS.

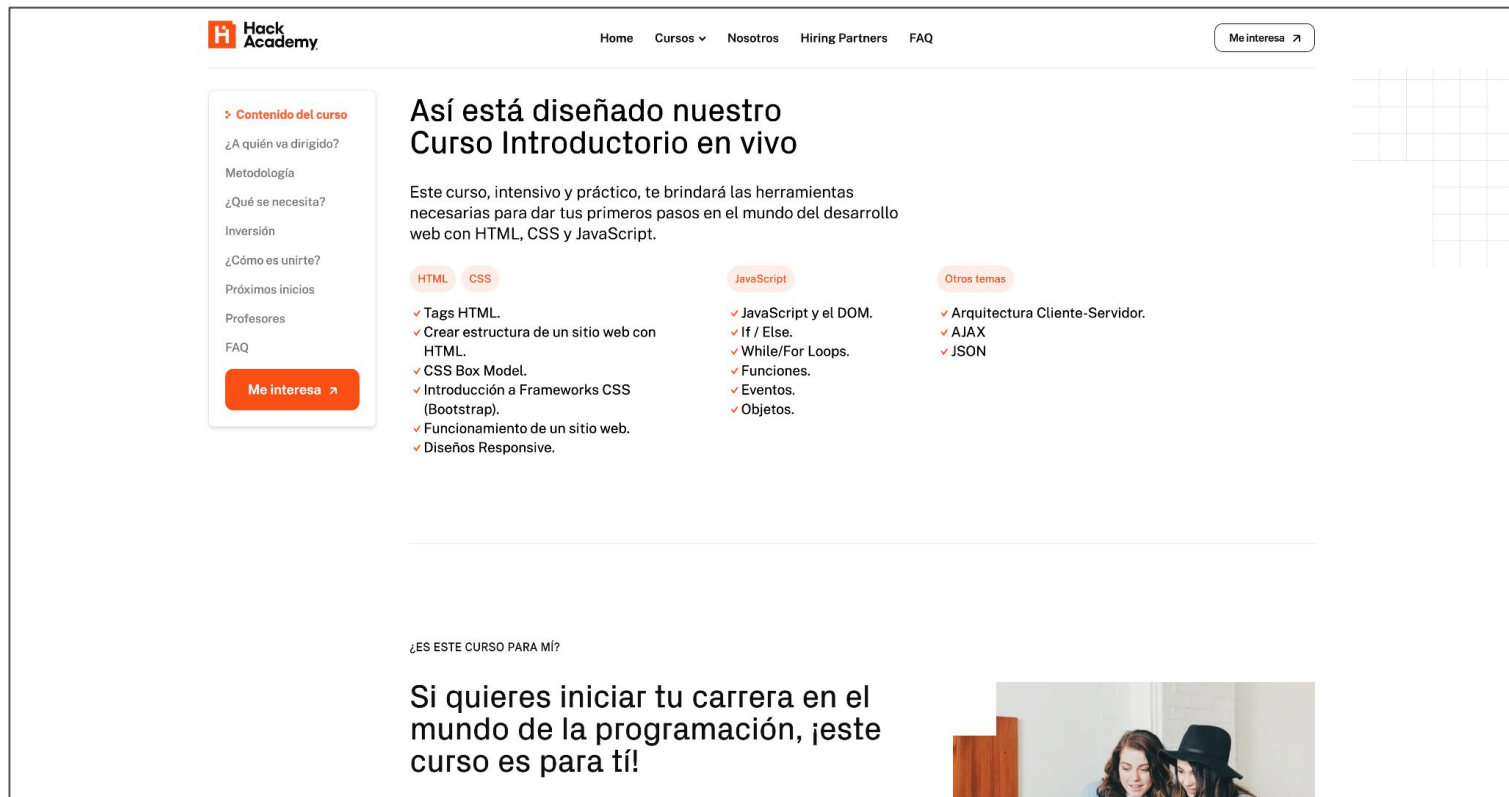
Si en el código creábamos tres párrafos `<p>`, uno debajo de otro, en ese orden se veían en la página.

¿Y si quisiésemos mostrarlos uno al lado del otro, es decir, en tres columnas? Para ello será necesario aprender una serie de **propiedades CSS** que nos permitirán estructurar una página tal como queramos.

Ver los siguientes ejemplos.



# Page Layout – Estructura de una página (2/6)





# Page Layout – Estructura de una página (3/6)

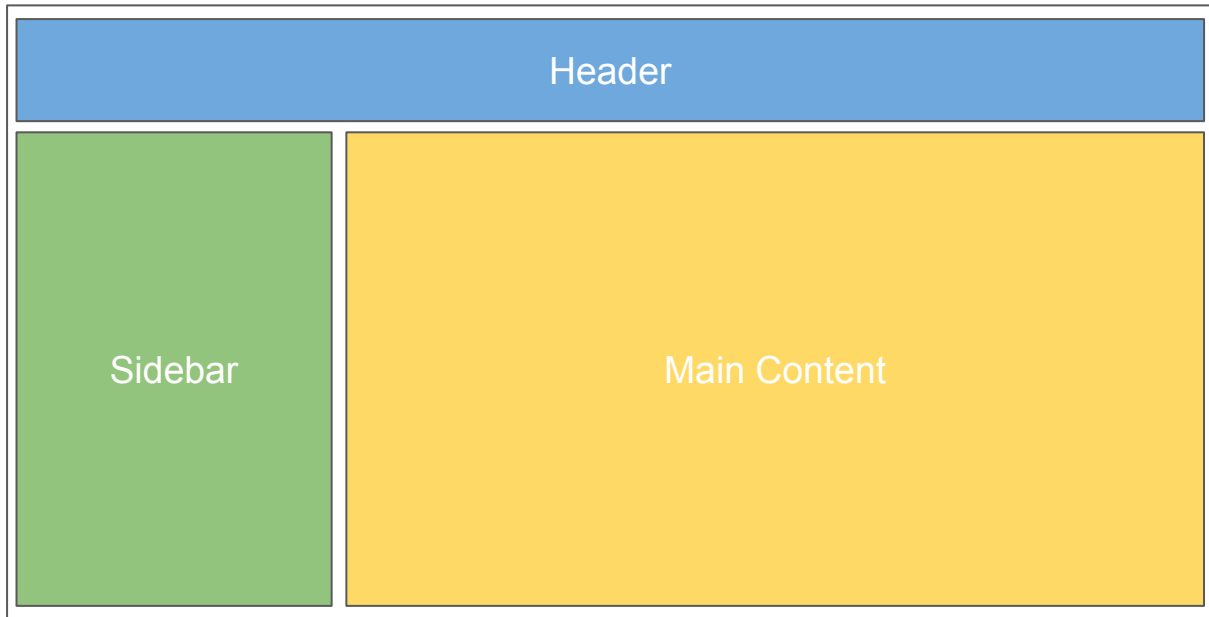






# Page Layout – Estructura de una página (4/6)

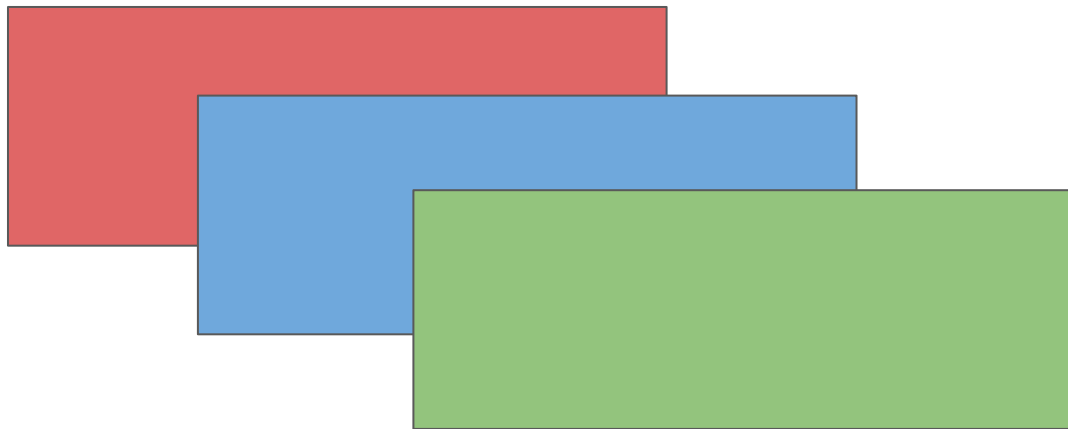
Otro ejemplo muy común de *layout* es definir un espacio para el cabezal, la barra lateral y el contenido principal de una página.





# Page Layout – Estructura de una página (5/6)

Controlando el *layout* de una página también se puede definir si un elemento debe aparecer por encima de otro, logrando un efecto tridimensional.





# Page layout – Estructura de una página (6/6)

Hoy veremos 3 propiedades **CSS** que se pueden usar para estructurar una página:

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



## AVISO:

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

Nota: Flexbox se puede usar [a partir de IE11](#) y Grid [a partir de Edge 16](#).



# CSS-position





# CSS – position

- `position` es una propiedad CSS utilizada para establecer el tipo de posicionamiento de un elemento.
- El valor por defecto es: `static`.
- También puede tomar los siguientes valores:
  - `absolute`
  - `relative`
  - `fixed`
  - `initial`
  - `inherit`

Estos son los que vamos a usar.



# CSS – position

Usando la propiedad `position`, **primero se define un eje de coordenadas** (usando los valores `absolute`, `relative` o `fixed`) y **luego se establece la posición** del elemento respecto a dicho eje.





`position: fixed`

*Posicionar un elemento respecto a la ventana*

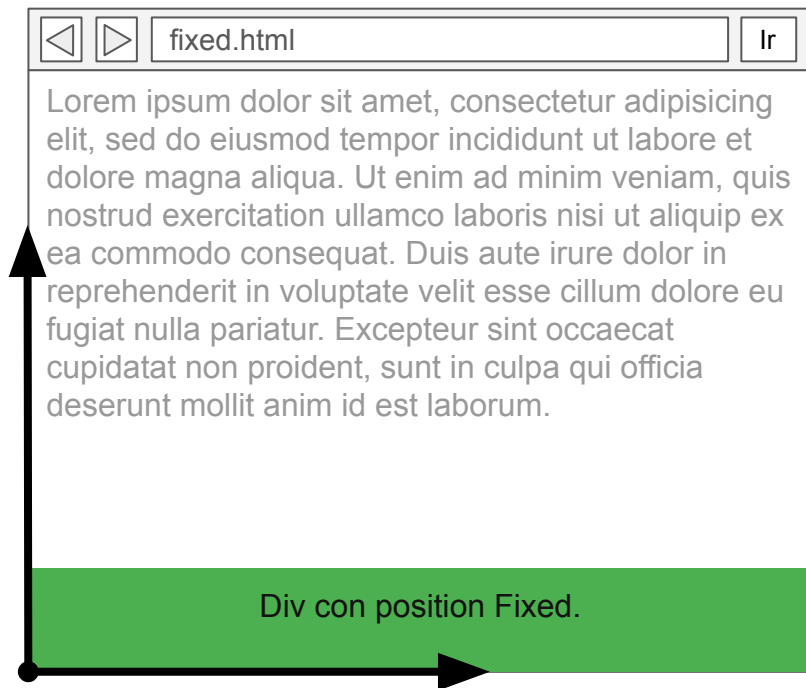






# CSS - position: fixed

El **elemento se posiciona con respecto a la ventana** del navegador. Provoca que el elemento quede **fijo** (no se mueve) al hacer *scroll*. Es necesario especificar las coordenadas del elemento con `top`, `bottom`, `left` y/o `right` (es decir, el desplazamiento desde los bordes).



```
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation...</p>
```

```
<div id="verde-fixed">
  Div con position Fixed.
</div>
```

```
#verde-fixed {
  position: fixed;
  bottom: 0px;
  left: 0px;
}
```

**Excepción:** si el elemento tiene uno de sus ancestros con una propiedad `transform`, `perspective`, o `filter` con un valor distinto de `none`, el elemento se "fija" a dicho ancestro (en lugar de "fijarse" a la ventana). Notar que esto no es algo común.



`position: static`

*Dejar posicionado un elemento de forma natural,  
según el orden en el código*

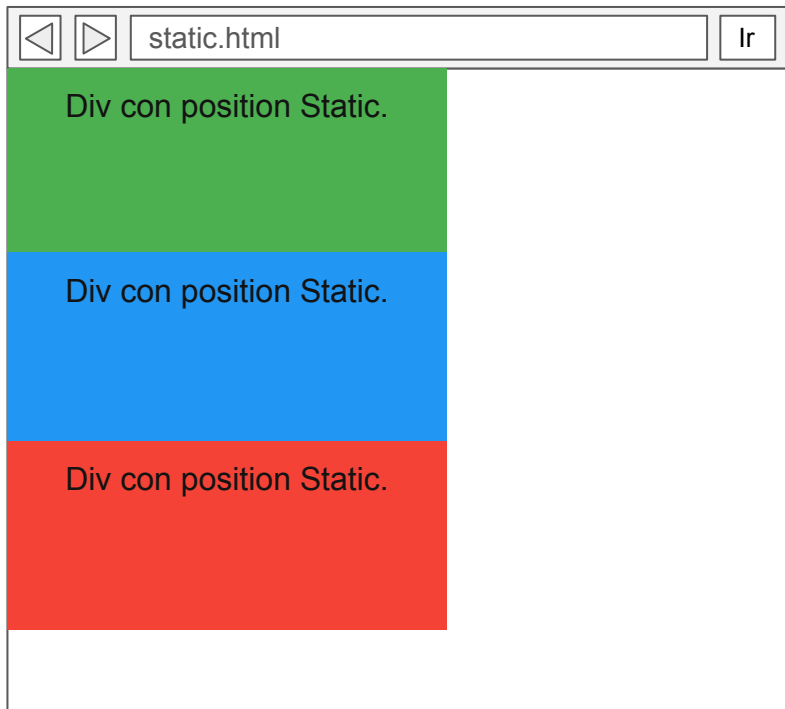




# CSS - position: static

El **elemento** se posiciona según el “**orden en el código**” (más exactamente, según el flujo normal del documento).

El valor `static` es el **valor por defecto** para la propiedad `position`.



```
<div id="verde" class="caja">
  Div con position Static.
</div>
<div id="azul" class="caja">
  Div con position Static.
</div>
<div id="rojo" class="caja">
  Div con position Static.
</div>
```

```
.caja {
  width: 400px;
  height: 100px;
  position: static; /* Es el valor por defecto */
}
```



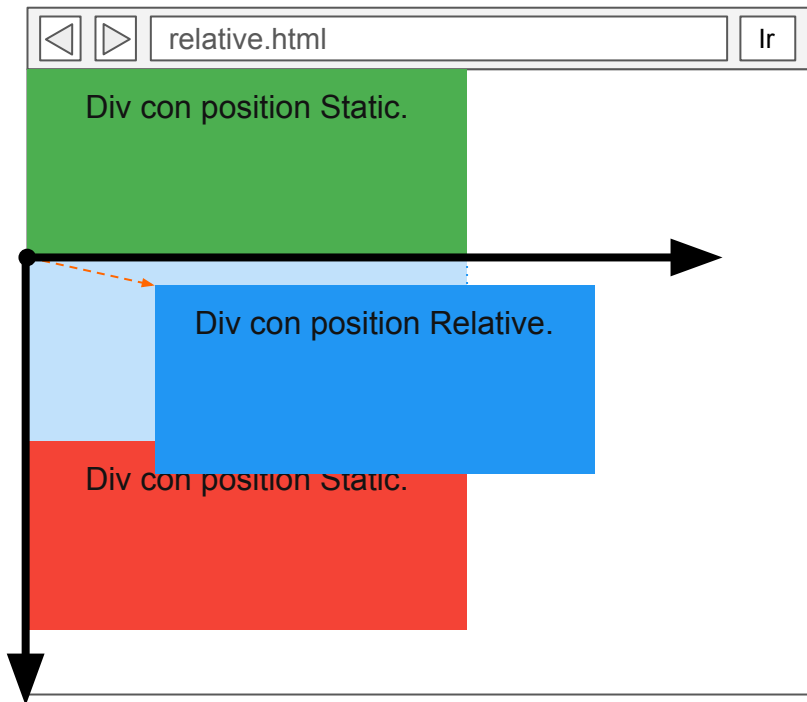
`position: relative`

*Posicionar un elemento con respecto  
a su posición natural 🤔*



# CSS - position: relative

El **elemento** se posiciona con respecto a su **posición natural**. Es necesario especificar las **coordenadas** del elemento con `top`, `bottom`, `left` y/o `right` (es decir, el desplazamiento desde los bordes). Si no se especifican las coordenadas, el elemento se comporta como `static`.



```
<div id="verde" class="caja">
  Div con position Static.
</div>
<div id="azul-relative" class="caja">
  Div con position Relative.
</div>
<div id="rojo" class="caja">
  Div con position Static.
</div>
```

```
#azul-relative {
  position: relative;
  top: 10px;
  left: 40px;
}
```



position: absolute

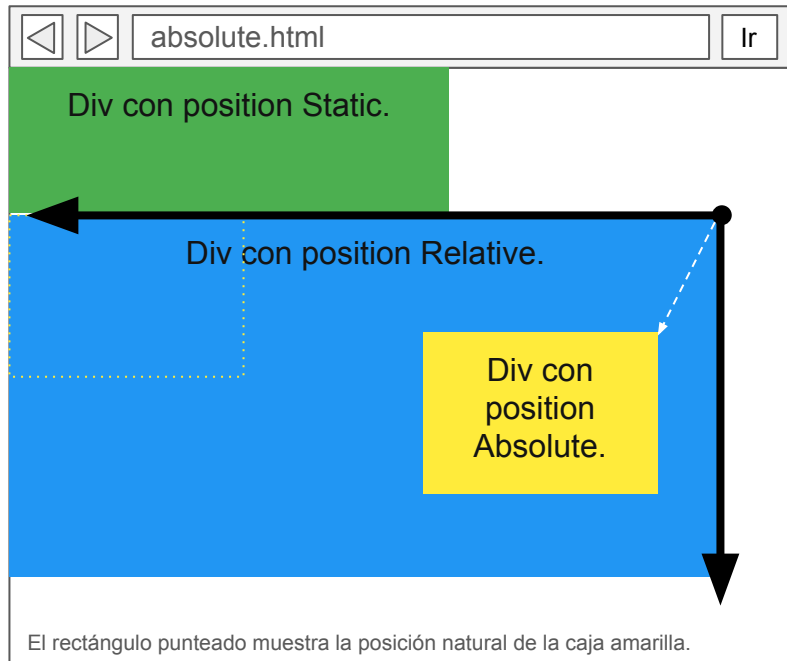
*Posicionar un elemento con respecto a su padre*  
(Definición simplificada)





# CSS - position: absolute

El **elemento** se posiciona con respecto a su **primer ancestro posicionado** (es decir, cuyo valor de position no es static). Es necesario especificar las **coordenadas** del elemento con `top`, `bottom`, `left` y/o `right` (es decir, el desplazamiento desde los bordes). Si no se especifican las coordenadas, el elemento se comporta como `static`.

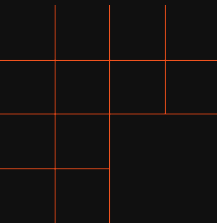


```
<div id="verde" class="caja">
  Div con position Static.
</div>
<div id="azul-relative" class="caja">
  Div con position Relative.
  <div id="amarillo-absolute" class="caja">
    Div con position Absolute.
  </div>
</div>
```

```
#amarillo-absolute {
  position: absolute;
  top: 50px;
  right: 10px;
}
```



# CSS - display







# CSS – display

Junto con `position`, `display` es otra de las propiedades importantes de CSS para controlar la estructura (o *layout*) de una página web.

¿Se acuerdan que los elementos HTML son de tipo `block` o `inline`?

Gracias a `display`, se pueden cambiar estos valores y *setear* otros como:

```
#un-elemento {  
    display: block; /* Setea al elemento como de tipo block. */  
    display: inline; /* Setea al elemento como de tipo inline. */  
    display: none; /* Oculta el elemento. */  
    display: inline-block; /* Similar a inline, pero puede tener width y height */  
}
```

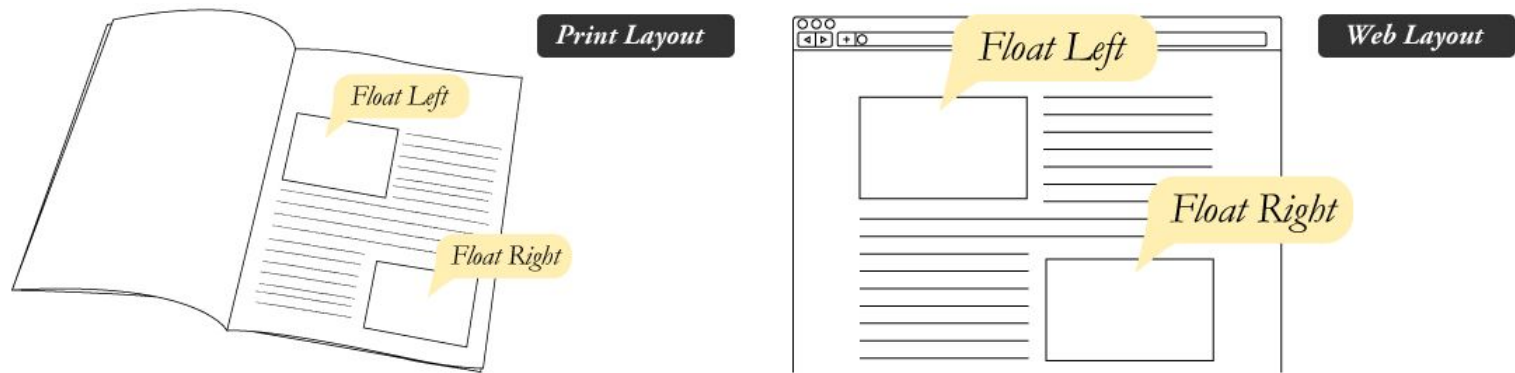
# CSS - float



# CSS – float (1/3)

La propiedad CSS `float` especifica que un elemento debe salir de su flujo normal y posicionarse a la izquierda o a la derecha de su contenedor. Se dice que se hace “flotar” al elemento hacia un lado o el otro.

Además, provoca que los elementos `inline` aparezcan a su alrededor (`wrap`).



Más información: <https://css-tricks.com/all-about-floats/>.

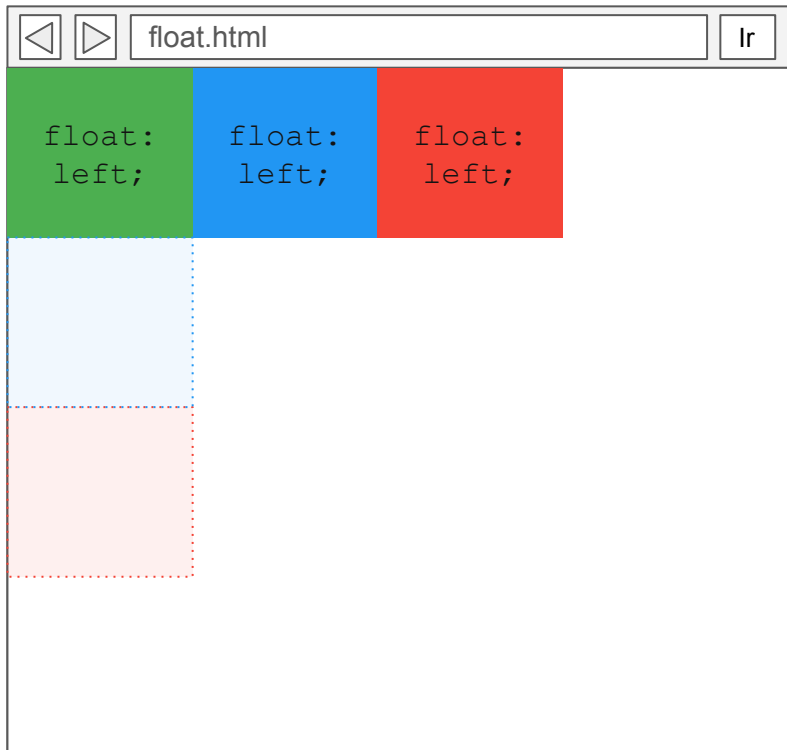
# CSS – float (2/3)

Ejemplo usando `float: left`.

EXTRA



Actualmente es muy poco usado y desaconsejamos usar `float` para lograr este tipo de comportamientos. Para estos casos se sugiere usar algo más moderno como Flexbox.



```
<div id="verde" class="caja">
  Float Left
</div>
<div id="azul" class="caja">
  Float Left
</div>
<div id="rojo" class="caja">
  Float Left
</div>
```

```
.caja {
  width: 100px;
  height: 100px;
  float: left;
}
```

# CSS – float (3/3)

Ejemplo usando `float: right`.

EXTRA



Este es, quizás, el único caso que sigue resolviéndose frecuentemente usando `float`.



```
<div id="anaranjado">  
  Float Right  
</div>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipisicing  
elit, sed do eiusmod tempor incididunt ut labore et dolore  
magna aliqua. Ut enim ad minim veniam, quis nostrud  
exercitation...</p>
```

```
#anaranjado {  
  background-color: #FF6600;  
  text-align: center;  
  width: 100px;  
  height: 100px;  
  float: right;  
}
```



# Links útiles





# Links útiles sobre Layout

Aconsejamos mucho que puedan leer sobre estos temas:

- [Learn CSS Layout.](#)
- [Learn CSS Positioning in Ten Steps.](#)
- [BrainJar – CSS Positioning.](#)
- [Khan Academy – CSS position.](#)
- [CSS Tricks – Position.](#)
- [CSS Tricks – All about floats.](#)



# Ejercicio 1










# Ejercicio 1

1. Crear una carpeta en el Escritorio (o donde prefieran) con el nombre `Clase04_Ejercicio_2`
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.

# Ejercicio 1 (parte 1/2)



 index.html Ir



Canadá

La idea de este ejercicio es practicar el uso de la propiedad `position: absolute` con el fin de hacer aparecer el texto “CANADÁ” arriba de una imagen, sobre la esquina superior izquierda. El texto debe quedar a 10px del borde superior y a 10px del borde izquierdo.

# Ejercicio 1 (parte 2/2)



lr



La idea de este ejercicio es practicar el uso de la propiedad (`display`), con el fin de hacer aparecer cada `div` uno al lado del otro, (en la misma línea)



# Ejercicio extra






# Ejercicio 2

1. Crear una carpeta en el Escritorio (o donde prefieran) con el nombre `Clase04_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.

# Ejercicio (extra)

EXTRA



index.html

## HACK ACADEMY

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

La idea de este ejercicio es practicar el uso de la propiedad `position: fixed` para el Header/Navbar. Se deberán generar varios párrafos de “relleno” para generar un *scroll* en la página.  
**PD:** También se podría haber usando `position: sticky`.