

Introducción

Cómo lucen los sitios web sin CSS (introduciendo Inspeccionar

Elemento en Chrome)

Hay 3 formas de adjuntar CSS a un documento: *inline*, *internal*, y *external*. **Inline e Internal son consideradas malas prácticas.** Solo utilizaremos *external* stylesheets, enlazadas desde el documento HTML utilizando la etiqueta <head>. Revisa el curso de HTML si no recuerdas cómo hacer esto.

CSS utiliza un esquema de prioridad para determinar qué reglas de estilo aplicar si más de una regla encaja con un elemento particular. En este esquema, las prioridades o pesos ("weights") bajan en cascada y son asignadas a reglas, por lo que los resultados son predecibles.

Una hoja de estilo es una lista de **reglas** (rules). Cada regla (o "rule-set") consiste en uno o más **selectores** (selectors) y un **bloque de declaración** (declaration block).

CSS Selector

Los *selectores* se utilizan para declarar los elementos de HTML para los cuales aplicará un estilo.

Los selectores más comunes son:

- Todos los elementos con una etiqueta específica de HTML (ej: p, h1)
- Los elementos especificados de acuerdo a los siguientes atributos:
 - **id** (son precedidos por # en CSS)
 - Debe ser único y solo puede ser utilizado una vez en tu página.
 - **class** (son precedidos por . en CSS)
 - Pueden ser utilizados muchas veces para compartir código CSS repetido.

Bloque de Declaración

Un bloque de declaración es una lista de declaraciones entre llaves "{ }". Cada declaración individual consiste en una **propiedad**, dos puntos (:), y un **valor**. Si un bloque tiene múltiples declaraciones, tienen que estar separadas por un punto y coma (;).

Algunos ejemplos:

Para todas las etiquetas “p” de HTML, haz el color de fuente azul (blue).

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

Para un elemento con el id “importante”, haz el tamaño de fuente (font size) de 36px.

```
#importante {  
  
    font-size: 36px;  
  
}
```

Para todos los elementos con la clase “info”, haz el fondo (background) verde y agrega un borde negro (black border) de 1px de ancho.

```
.info {  
  
    background-color: green;  
  
    border: 1px solid black;  
  
}
```

En estos ejemplos, **p**, **#importante** e **.info** son **selectores**. **Color**, **font-size**, **background-color** y **border** son **propiedades**.

Más sobre selectores:

- <http://www.webdesignerdepot.com/2013/08/10-css-selectors-you-shouldnt-code-without/>
- http://www.w3schools.com/cssref/css_selectors.asp

Ahora que sabemos un poco más sobre cómo usar selectores, hagamos una lista general de las propiedades internal para CSS:

1. element (1 punto)
2. .class (10 puntos)
3. #id (100 puntos)

Este es el orden de prioridad predeterminado. Adicionalmente, la especificidad también cuenta, por ejemplo, `ul li` anulará a `li`. W3C ha creado una buena tabla respecto a cómo deberías calcular el peso interno (internal weight):

```
li          {...} /* a=0 b=0 c=1 -> specificity = 1 */
ul li       {...} /* a=0 b=0 c=2 -> specificity = 2 */
ul ol li    {...} /* a=0 b=0 c=3 -> specificity = 3 */
li.red      {...} /* a=0 b=1 c=1 -> specificity = 11 */
ul ol li.red {...} /* a=0 b=1 c=3 -> specificity = 13 */
#list       {...} /* a=1 b=0 c=0 -> specificity = 100 */
```

- **a** representa el número de atributos #id en el selector
- **b** representa el número de atributos de clase (class)
- **c** representa el número de nombres de etiqueta

Combinar esto en un número te dará el peso real. Esto significa que `li #list` tendrá el mismo peso que `ul #list`. Esto es, probablemente, una de las cosas más confusas del esquema en cascada (cascade scheme), pero es de hecho más simple de lo que piensas: **se trata de contar**.

Elementos de Estilo

Más abajo se presentan algunas de las propiedades de estilo más comunes que afectarán a todos los elementos. No es una lista definitiva y deberías buscar otras propiedades que puedan ayudarte a agregar estilo a tus documentos.

width | height:

Las propiedades width (ancho) y height (altura) son utilizadas para determinar el tamaño de tus elementos. Los valores pueden ser expresados en píxeles (px) y porcentaje (%). Cuando se trabaja con contenido estático, es recomendable utilizar px puesto que definirás tu página para que no cambie. Si trabajas con un diseño responsivo, es mejor utilizar %,

Ten cuidado al establecer tus propiedades de altura ya que esto determinará cuánto contenido puede tener el elemento. Si no lo configuras, tu elemento se extenderá para encajar con el contenido. Por otro lado, si lo configuras y tienes más contenido que el que se puede mostrar, entonces tendrás que ajustar la altura manualmente cada vez que cambies el contenido o deberás utilizar la propiedad overflow.

Cuando quieras redimensionar tu elemento (ej: imágenes), puedes ajustar una propiedad (ancho o altura) y la otra se ajustará automáticamente para conservar las dimensiones relativas de la imagen.

```
a {  
    width: 25px;  
}  
  
div {  
    width: 100%;  
    height: 200px;  
}  
  
img {  
    width: 250px;  
}
```

overflow:

Esta propiedad determina qué debe pasar cuando el contenido dentro de un contenedor es demasiado para el tamaño del contenedor. Puedes configurar el contenedor para ocultar la información adicional que no cabe, mostrar la información como sea o agregar una barra de desplazamiento (scroll bar) con lo que el contenido podría ser visto pese a su tamaño.

```
div {  
    overflow: scroll;  
}
```

background:

La propiedad background (fondo) puede modificar el fondo de un elemento en una sola línea, lo que es más sencillo que separar cada propiedad en su propia línea de código. El color puede ser definido utilizando hex, rgb o código semántico (semantic code).

```
p {
```

```
background: #ffffff url("cherries.png") no-repeat fixed center;

}
```

background-color | background-image | background-position | background-size | background-repeat:

Estas propiedades ajustan el fondo de acuerdo a su tipo. Como background, background-color puede ser definida utilizando hex, rgb o código semántico.

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

div {
  background-image: url("cherries.png");
  background-position: center;
  background-size: auto;
  background-repeat: no-repeat;
}
```

border:

Esta propiedad ajusta todos los elementos de borde en una sola línea. Así, el primer valor es el grosor del borde. El segundo valor es tipo de borde, mientras que el tercero es el color del borde. El color puede ser hex, rgb o código semántico.

La propiedad border puede separarse en distintas líneas utilizando border-width, border-style y border color. Además, puedes seleccionar específicamente qué borde modificar utilizando border-top (arriba), border-bottom (abajo), border-right (a la derecha) y border-left (a la izquierda).

```
button {
  border: 2px dotted green;
}

div {
```

```
border: 1px solid #000000;

}

p {

border-right: 1px groove rgb(100,100,100);

border-left: 1px groove rgb(200,200,200);

}
```

border-radius

Esta propiedad permite dar una apariencia curva a las esquinas de tu borde. Los valores puede ser establecidos utilizando px o %.

```
button {

border-radius: 5px;

}
```

Texto de Estilo

A continuación algunas de las más comúnmente utilizadas propiedades de estilo de texto (text style properties). Esta no es la lista completa, por lo que te invitamos a buscar en internet otras propiedades de estilo de texto..

color:

El color de texto se especifica usando la propiedad de color. El valor de la propiedad de color puede ser expresada utilizando hex, RGB o código semántico.

```
p {

color: #ffffff;

}

a {

color: rgb(255, 255, 255);

}

span {
```

```
color: white;

}
```

text-align:

Esta propiedad se utiliza para establecer la alineación horizontal de cualquier texto. El texto puede estar centrado, alineado a la izquierda o a la derecha, o justificado. Esta propiedad solo funcionará si también presenta un bloque.

```
h1 {

    text-align: center;

}
```

text-decoration:

Es utilizada para agregar o remover underlines (subrayados), overlines (línea sobre) y line-throughs (tachado).

```
a {

    text-decoration: none;

}
```

Font (Fuente)

font-family:

La propiedad font-family especifica el estilo de fuente a ser usado por un elemento. Existen 2 tipos de nombres de font-family:

- named-family - Ejemplos: "times", "courier", "arial"
- generic-family - Ejemplos: "serif", "sans-serif", "cursive", "fantasy", "monospace"

Fuentes Web-safe:

Existen fuentes instaladas en casi todos los sistemas y se denominan web safe fonts (fuentes seguras para la web), porque en general se verán igual independiente del sistema que tenga el usuario. Algunas de ellas son: Verdana, Arial, Trebuchet MS, Times New Roman, Georgia, Andale Mono, Courier New, Comic Sans e Impact.

```
p {  
  
    font-family: "helvetica neue", arial, verdana, sans-serif;  
  
}
```

Font-family permite ingresar múltiples opciones cuando el navegador las recorre de izquierda a derecha hasta que encuentra una fuente en el sistema que puede ser usada. Esto significa que siempre debes asegurarte de tener una opción generic-family como tu último valor, ya que si alguno de los otros puede ser encontrado, el navegador utilizará la opción predeterminada para la familia especificada.

- El navegador busca la fuente Helvetica Neue en el sistema de usuario y si la encuentra la utiliza.
- Si no encuentra Helvetica Neue, el navegador busca Arial en el sistema del usuario y si la encuentra la utiliza.
- Si Arial no es encontrada, usará Verdana.
- Como último recurso, si no encuentra ninguna de las fuentes en el grupo de fuentes, volvemos a sans-serif instruyendo al navegador que use sea cual sea la fuente sans-serif predeterminada del sistema. No sabrás que se usará exactamente en este caso, pero al menos es mejor que terminar con la fuente predeterminada del navegador, Times new roman, la cual es una fuente serif.

Ten en cuenta que las fuentes con más de una palabra en su nombre deben ir entre comillas.

font-size:

La propiedad font-size (tamaño de fuente) puede ser expresada en 4 unidades diferentes: pt, px, em, %. Pt y px (punto y pixel) son consideradas dimensionamiento estático y no se ajustarán adecuadamente cuando se redimensione tu página. Em y % (medida responsiva y porcentaje) sí se ajustarán y, por ende, son las recomendadas por los desarrolladores. A continuación podrás ver un cuadro desplegable que muestra las equivalencias aproximadas entre las cuatro unidades.

```
h3 {
```



```
font-size: 10pt;

}

p {

    font-size: 14px;

}

a {

    font-size: 1.2em;

}

span {

    font-size: 80%;

}
```

font-style

Esta propiedad controla la inclinación de la letra. Algunos textos pueden tener la propiedad italic (cursiva) como parte de la fuente, por lo que seleccionarla sería simple. Para las fuentes que no tienen el estilo italic, el valor oblique puede ser utilizado para imitar el texto en italic.

```
span {

    font-style: italic;

}
```

font-weight

Esta propiedad define el grosor de una línea de caracteres. normal es normalmente el valor por defecto. Los valores pueden ser establecidos de forma numérica o semántica.

```
p {

    font-weight: bold;

}

span {
```

```
font-weight: 600;

}
```

¿Qué es una etiqueta span?

Si quieres resaltar un texto específico dentro de un párrafo o división, entonces puedes utilizar ``. A diferencia de `<div>` que sitúa al siguiente `<div>` en una línea separada, `` por defecto pone al elemento en la misma línea. Por ejemplo, considera

```
<h1>Hello <div>World</div> </h1>
```

Arriba por defecto pondríamos 'World' (Mundo) en una nueva línea. Sin embargo, si quisieras que World se mostrara en la misma línea, deberías escribir

```
<h1>Hello <span>World</span> </h1>
```

Luego, en el css, podrías hacer algo como esto para resaltar solo la palabra "World":

```
h1 span {

  font-weight: bold;

  color: blue;

}
```

Para leer más sobre la diferente entre `<div>` y ``, puede

leer: https://www.codecademy.com/en/forum_questions/502ad0ea558dfe0002026d69

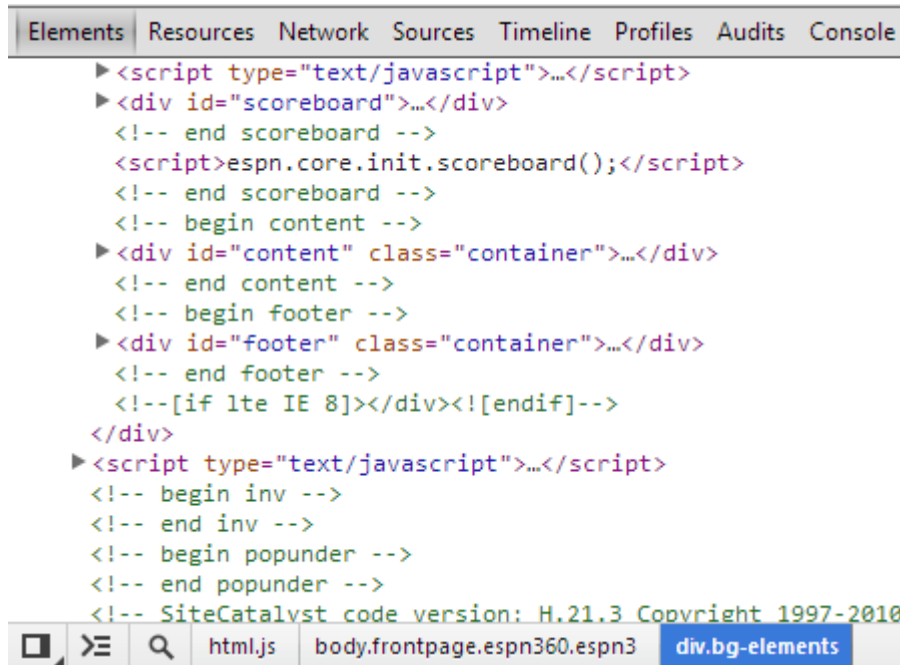
Cómo usar Inspeccionar Elemento (Inspect Element)

Video de Inspeccionar elemento de css

Editar HTML y CSS puede verse como desactivar una bomba, un pequeño cambio y tu proyecto enloquece. Tu mejor aliado al editar son las herramientas de desarrollador (Developer Tools).

Se incluyen en **Chrome**, **Firefox**, **la última versión de Internet Explorer (Edge)** y **Safari**.

Para acceder a Inspeccionar Elemento, simplemente usa el clic derecho en cualquier elemento de la ventana del navegador y cliquea en Inspeccionar (**Inspect**) ¡y listo!. Inspeccionar Elemento es una característica que muestra todas las propiedades del sitio web que estás viendo.



Esta es una captura de pantalla de la ventana inspeccionar de Chrome. En la barra gris de arriba, verás un conjunto de opciones incluyendo 'elements' y 'console', que muestran distinta información de la página.

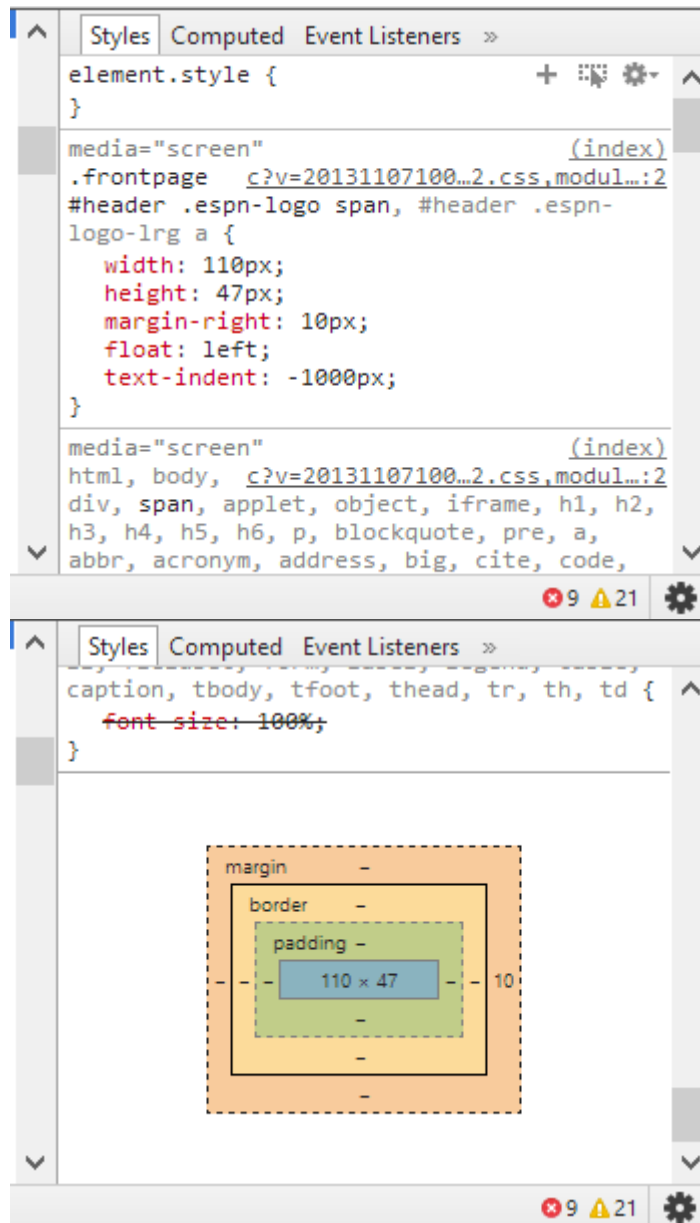
Pestaña de Elementos (Elements Tab)

Por las siguientes dos lecciones, nos centraremos en ESPN.com. Puesto que la mayoría de las empresas cambian sus sitios webs con frecuencia, ten en cuenta que los sitios de ESPN que verás, pueden ser un tanto diferentes a los ejemplos del vídeo de más abajo. Pese a ello, las mismas herramientas y conceptos pueden ser aplicados.

La **pestaña elements**, te permite ver el HTML que está siendo procesado por tu navegador. Puedes ver todas las *classes* e *ids* del contenido HTML, lo que es de gran ayuda para depurar CSS. La imagen de más arriba, muestra la pestaña de elementos de una porción del HTML de ESPN.com ¡Fíjate que incluso puedes ver comentarios!!

Si cliqueas en un elemento particular del HTML dentro del cuadro de **elementos**, verás que el lado derecho de la ventana de Inspeccionar Elemento cambia. Esta parte de la ventana muestra

una tonelada de información de **CSS y JavaScript**. Hay una pestaña a la derecha llamada 'estilos' (styles) que es por lejos la herramienta más útil al trabajar con CSS.



(Este es el elemento que detallan las 2 imágenes anteriores)

La pestaña **estilos** muestra **todo el el CSS procesado para un elemento particular**. La captura de pantalla anterior es la información de CSS para el logo de ESPN en su página web. A la izquierda de la imagen se muestra los estilos ejecutados del elemento y la hoja de estilo de donde fueron tomados. Baja hasta el final de la pestaña y verás la imagen a la derecha, que debería ser familiar. Esta imagen representa los valores del modelo de cajas del elemento en cuestión. Puedes ver el **margen (margin)**, **relleno (padding)**, **borde (border)** y las

dimensiones de cualquier elemento de la página. Aún mejor, puedes pasar por una parte específica de la caja en la ventana de inspeccionar elemento y la propiedad particular del elemento será resaltada en la página ¡Inténtalo! Esta herramienta es genial para ajustar la posición (los cambios solo se realizarán en el navegador y no se guardarán en el archivo CSS)

Mira la imagen nuevamente, verás un sector donde dice '**element.style**'. Ahí puedes escribir tu propio CSS y modificar lo que se muestra en la página web. Por supuesto, estos cambios no son permanentes, pero ya puedes imaginarte lo útil de esta herramienta. Otra cosa más que puedes hacer con todas las reglas de estilos de la pestaña estilos (styles), es desactivar las propiedades CSS asociadas des-marcándolas.

Cómo usar Inspeccionar Elemento

Genial, ¿y ahora qué?

Cualquier buen desarrollador web te dirá que **Inspeccionar Elemento** es una excelente herramienta en la lucha contra el código deficiente. Hay personas cuyo trabajo es específicamente hacer estas herramientas de desarrollo, por lo que estás en buenas manos. Inspeccionar es muy fácil de entender cuando la exploras ¡Empieza a usarla regularmente y te ahorrará mucho tiempo!

Selector de Color

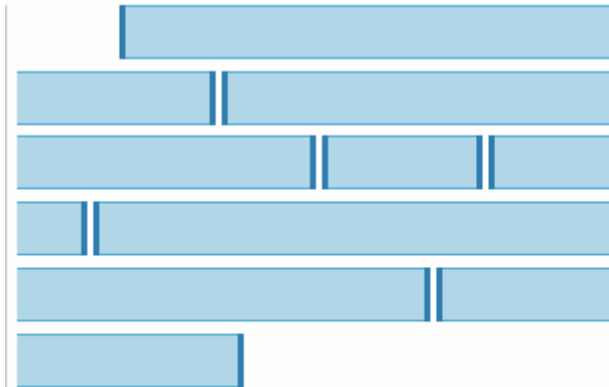
El diseño y selección de la paleta de colores representan una parte importante en la creación de una página web. Para ayudar con una selección de colores precisa, hay una variedad de herramientas disponibles para encontrar y definir los colores, además de armar paletas de colores. Es fácil encontrar en internet herramientas para generar esquemas de colores, pero encontrar el color exacto utilizado en el borde del botón de tu sitio web favorito puede ser más complicado. En esta sección esbozaremos cómo usar la herramienta de selector de color (colorpicker) de Chrome Dev, que nos ayudará a encontrar colores que nos interesan desde otros sitios web.

Para acceder al selector de color, primero necesitamos abrir el Inspector de elementos y seleccionar un elemento con un color de fondo. Luego, cliqueamos en el cuadrado de color de la ventana de Estilo (Styles). Entonces, se mostrará información respecto al color de fondo del elemento, permitiendo distintos ajustes como los valores del código de colores, distintas

Video de Inline y block css css3

Cada elemento HTML es por defecto inline o block.

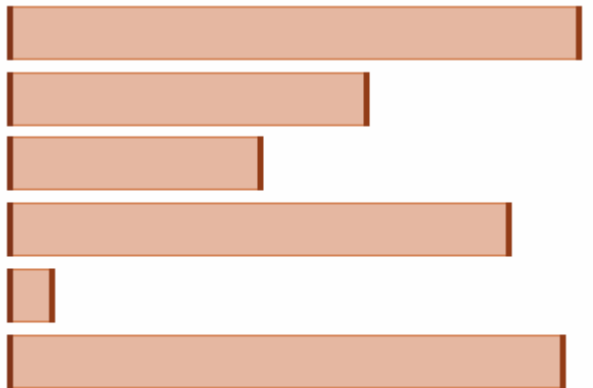
Los elementos *Inline* solo toman cuanto espacio sea necesario.



Inline elements flow from one line to the next.

Elementos Inline: `<a>`, ``, ``, `<input>`, `<label>`, `<select>`, `<textarea>`.

Los ítems **Block** toman el 100% del ancho del elemento padre (parent). Incluso si el ancho del elemento fuera menor que el padre, de todas formas abarcaría el 100% del ancho.



Block elements stack,
regardless of their widths.

Elementos Block: `<body>`, `<div>`, `<form>`, `<h1>`-`<h6>`, `<p>`, `<table>`, ``, ``, ``

No es válido poner elementos block dentro de elementos inline.

Ej:

No válido:

```
<a href="http://www.google.com">http://www.google.com</a>">www.google.com<
h1>Google</h1></a>
```

Válido:

```
<h1><a href="<a href="http://www.google.com">http://www.google.com</a>">Go
ogle</a></h1>
```

Sin embargo, podemos cambiar el estilo de visualización predeterminado de HTML utilizando CSS y la propiedad de visualización (display). Los posibles valores de esta propiedad son: **inline**, **block**, **none** e **inline-block**.

Ej: Enlaces Inline predeterminados

[One](#) [Two](#) [Three](#) [Four](#) [Five](#)

```
<a href="#">One</a>

<a href="#">Two</a>

<a href="#">Three</a>

<a href="#">Four</a>

<a href="#">Five</a>
```

```
a {

    margin: 10px;

    background: #eee;

    color: black;

    padding: 10px;

}
```

Convirtiéndolos en elemento de bloque (block):

One

Two

Three

Four

Five

```
<a href="#">One</a>

<a href="#">Two</a>

<a href="#">Three</a>

<a href="#">Four</a>

<a href="#">Five</a>
```

```
a {

  margin: 10px;

  background: #eee;

  color: black;

  display: block;

  padding: 10px;

}
```

En el reverso, supón que tenemos el siguiente menú de navegación:

```
<ul id="nav-menu">

  <li><a href="#">Home</a></li>

  <li><a href="#">Dashboard</a></li>

  <li><a href="#">About</a></li>
```

```
<li><a href="#">Contact Us</a></li>

</ul>
```

El HTML normalmente aparecería así:

- [Home](#)
- [Dashboard](#)
- [About](#)
- [Contact Us](#)

Cuando queremos hacer un menú horizontal en la parte superior de nuestra página, podemos hacer lo siguiente:

```
#nav-menu{

    list-style-type: none; /*This gets rid of the bullets.*/

}

#nav-menu li {

    display: inline;

}
```

Ahora se ve así:

[Home](#) [Dashboard](#) [About](#) [Contact Us](#)

Inline-block:

Considera el siguiente ejemplo:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

```
p {  
  width: 350px; margin: 20px; padding: 20px;  
  font-size: 14px;  
  background: #eee;  
}
```

Recuerda que las etiquetas `<p>` son elementos block por defecto. Cuando las visualizamos como inline, ambos actúan juntos en un solo párrafo.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

```
<p>Lorem ipsum dolor sit amet...</p>  
  
<p>Lorem ipsum dolor sit amet...</p><br>  
  
p {  
  font-size: 14px;
```

```
background: #eee;

display: inline;

}
```

Hemos perdido toda capacidad de establecer nuestro ancho y alto puesto que los dos párrafos están completamente integrados.

Los mismos párrafos con inline-block:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

```
p {

width: 180px;

margin: 10px;

padding: 20px;

font-size: 14px;

background: #eee;

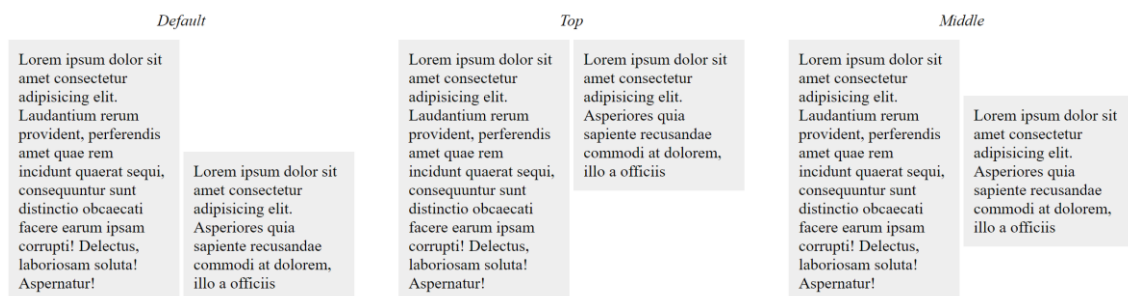
display: inline-block;

}
```

Lo que sucede aquí es que le estamos pidiendo al navegador que visualice los párrafos inline, pero permitiéndoles mantener sus características block-level. Esto significa que podemos fijar un ancho y alto manualmente y que ambos elementos permanezcan diferentes, además de aparecer uno junto al otro en el flujo del documento.

Alineación Vertical

Cuando comenzamos a utilizar inline-block, tal vez habremos notados que la posición vertical de los bloques puede empezar a comportarse de manera diferente a lo que esperábamos, especialmente cuando los bloques tienen diferentes alturas. Por defecto, los elementos se alinean con el borde de más abajo de los bloques dentro de la línea. Para ajustar este comportamiento, podemos usar la propiedad de CSS denominada `vertical-align` la cual tiene valores como top (arriba) y middle (al medio).



Lectura Complementaria:

<https://uniwebsidad.com/libros/css/capitulo-5?from=librosweb>

Modelo de Caja (Box Model) - Margen (Margin), Borde (Border), Relleno (Padding)

Video de modelo de caja

*Puedes descargar una copia del código anterior [aquí \(layouts.zip\)](#).

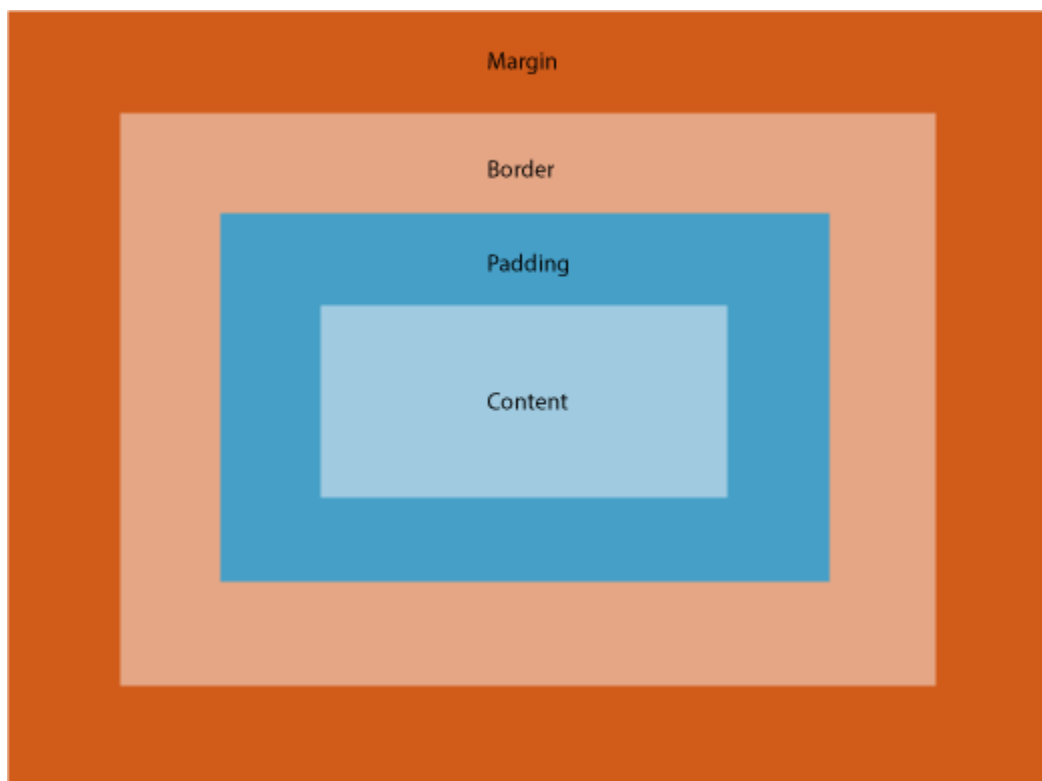
Todos los diseños web se realizan con elementos de bloque o block. Los diseñadores utilizan blocks (normalmente elementos `<div>`), para crear áreas rectangulares donde caben todos los contenidos. Solo existen 3 reglas:

- **Área total:** El espacio que un elemento ocupa y afecta.
- **Float, clear y overflow**
- **Elementos anidados (nested)**

Área Total

El ancho total se refiere a cuánto espacio horizontal utiliza un bloque. Esto incluye el margen, el borde y el relleno del bloque. Calcular el ancho, relleno y margen es a menudo el mayor dolor de cabeza para los diseñadores, pero es fácil ver cómo trabajan si utilizas el **modelo de caja**.

El modelo de caja se compone de las propiedades de **margen (margin)**, **borde (border)** y **relleno (padding)**.



El margen está afuera de los elementos de bloque, mientras que el relleno está dentro de ellos. Esto significa que usamos el margen para separar un bloque de las que lo rodean, y relleno para mover el contenido de un bloque lejos de sus bordes.

Podemos definir específicamente el margen, relleno o borde de cualquier lado de un elemento.

```
div {  
  padding-top: 10px;
```

```
padding-right: 10px;

padding-bottom: 10px;

padding-left: 10px;

}
```

Además puedes usar la propiedad de taquigrafía (shorthand):

```
div {

padding-top: 25px;

padding-right: 50px;

padding-bottom: 75px;

padding-left: 100px;

}
```

que es equivalente a:

```
div {

padding: 25px 50px 75px 100px;

}
```

Y

```
div {

padding-top: 25px;

padding-right: 50px;

padding-bottom: 75px;

padding-left: 50px;

}
```

que es equivalente a:

```
div {
```

```
padding: 25px 50px 75px;  
}
```

Y

```
div {  
  
padding-top: 25px;  
  
padding-right: 50px;  
  
padding-bottom: 25px;  
  
padding-left: 50px;  
  
}
```

que es equivalente a:

```
div {  
  
padding: 25px 50px;  
  
}
```

Y

```
div {  
  
padding-top: 25px;  
  
padding-right: 25px;  
  
padding-bottom: 25px;  
  
padding-left: 25px;  
  
}
```

que es equivalente a:

```
div{  
  
padding: 25px;  
  
}
```


(El orden va en el sentido del reloj, arriba -> derecha -> abajo -> izquierda).

Entonces, y de acuerdo al modelo de caja, el ancho total de un elemento es:

(Margin x 2) + (Border x 2) + (Padding x 2) + Content Width (Ancho del contenido)

Calcular la altura (height) es más complicado ¿Por qué? Porque los márgenes **verticales se desploman**.

Ej:

HTML:

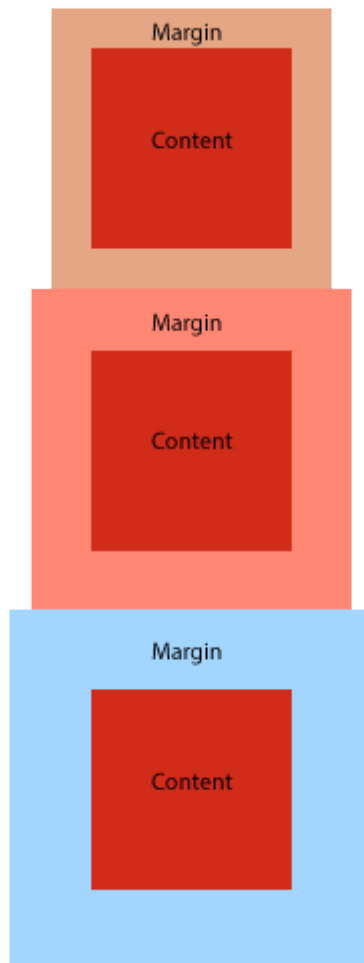
```
<div id="box-1">
</div>
<div id="box-2">
</div>
<div id="box-3">
</div>
```

CSS:

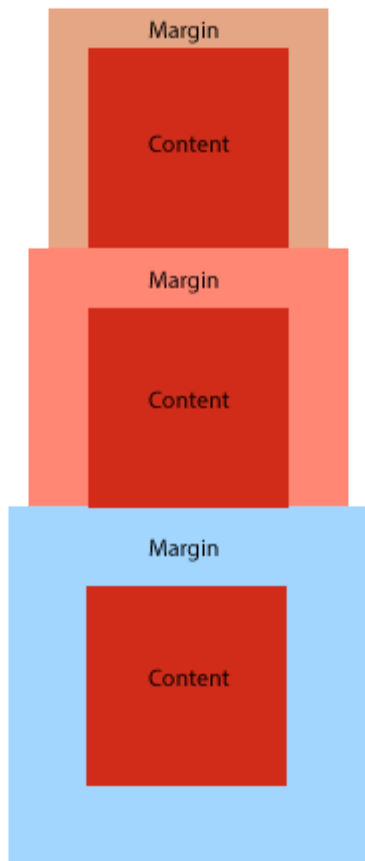
```
#box-1, #box-2, #box-3{
    height: 100px;
    width: 100px;
    background-color: red;
}
#box-1{
    margin: 20px;
}
#box-2{
    margin: 30px;
```

```
}  
  
#box-3{  
  
    margin: 40px;  
  
}
```

Tal vez piensas que esto se verá así:



Pero el código se verá más bien así:



Cuando los márgenes verticales de dos elementos se tocan, **solo el margen del elemento con el valor de margen más grande se mantendrá**, mientras que el margen del elemento con el menor valor de margen se reducirá a cero.

Existen otras situaciones donde los elementos no tienen un derrumbe o reducción en sus márgenes:

- Elementos flotantes
- Elementos absolutamente posicionados
- Elementos inline-block
- Elementos con la propiedad overflow establecida para cualquier otra cosa, menos para que sea visible (Estos no derrumban sus márgenes con sus hijos).
- Elementos eliminados o cleared (Estos no derrumban sus márgenes superiores con el margen inferior de su bloque padre).

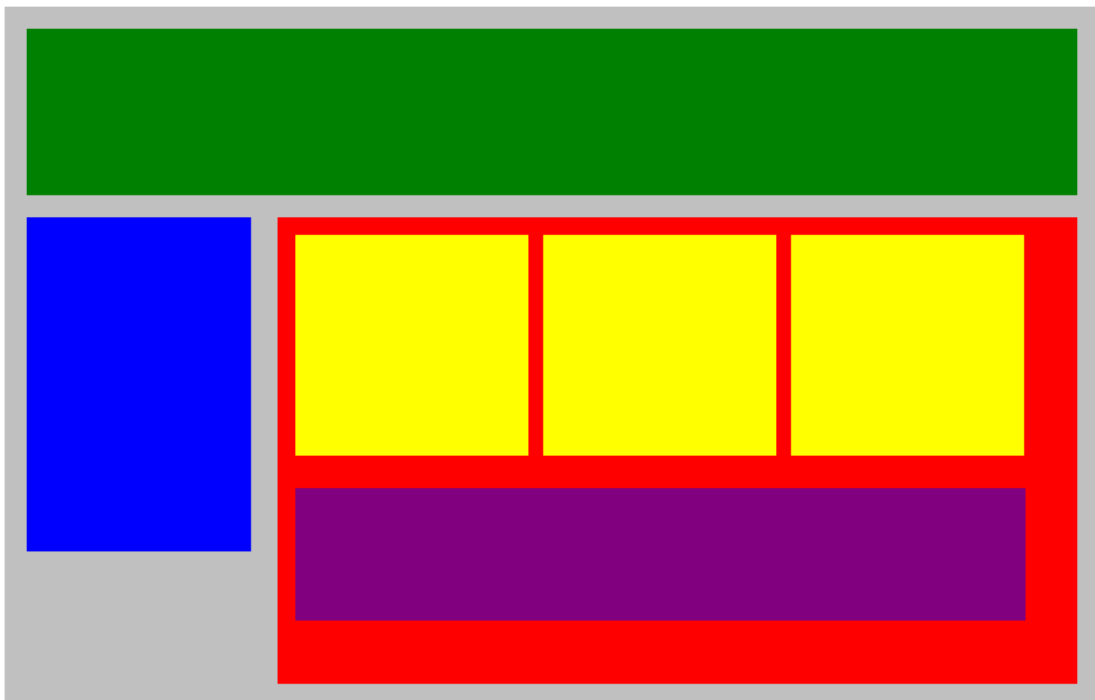
Concepto Lego

El concepto Lego es una forma de visualizar los diferentes elementos en tu sitio como piezas de lego. Así, primero colocarás los grandes bloques y luego bloques más pequeños dentro de ellos. Al focalizarse en los bloques grandes primero, puedes pensar en el diseño general de tu sitio. Luego de posicionar los bloques grandes, debes determinar qué bloques más pequeños existirán dentro de los bloques más grandes y situarlos de acuerdo a ello. Puede parecer más rápido mirar una captura de pantalla y empezar a escribir en HTML/CSS inmediatamente, pero dibujar las diferentes piezas de lego en la pizarra primero (las grandes, luego las más pequeñas) te ayudará a construir en HTML/CSS de manera más expedita.

Aplicando el Concepto Lego a un Sitio (WalMart) (VIDEO Concepto Lego)

Trazando tus Bloques

Intenta duplicar la siguiente imagen ajustando el código CSS. Usa **margins y paddings** para ajustar los espacios entre divisiones y utiliza la propiedad `display` para lograr situar cada bloque en el lugar correcto. Tal vez necesites propiedades de CSS adicionales.



Este es el código de HTML:

```

<!DOCTYPE html>

<html lang="en">

  <head>

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

    <title>Position Practice</title>

    <link rel="stylesheet" type="text/css" href="style.css">


  </head>

  <body>

    <div id="wrapper">

      <div id="header"></div>

      <div id="navigation"></div>

      <div id="main_content">

        <div class="subcontents"></div>

        <div class="subcontents"></div>

        <div class="subcontents"></div>

        <div id="advertisement"></div>

      </div>

    </div><!-- end of wrapper -->

  </body>

</html>

```

y CSS:

```

/*CSS reset settings here*/

*{

  margin: 0px;

  padding: 0px;

```

```
}

#wrapper{

    width: 950px;

    background-color: silver;

    margin: 0px auto;

}

#header{

    min-height: 150px;

    background-color: green;

}

#navigation{

    min-height: 300px;

    width: 200px;

    background-color: blue;

}

#main_content{

    min-height: 400px;

    width: 700px;

    background-color: red;

}

.subcontents{

    min-height: 200px;

    width: 210px;

    background-color: yellow;

}

#advertisement{

    min-height: 120px;
```

```
width: 660px;

background-color: purple;

}
```

No gastes más de 2-3 horas en esta actividad.

Mientras realizas esta actividad, utiliza min-height (altura mínima) y vertical align (alineación vertical) para darle una altura mínima a la separación y alinear verticalmente algunos de los inline-blocks. Por favor, NO uses float (utiliza display:inline-block).

- Work on the exercise above.
- Upload your updated HTML and CSS code here. Make sure that your HTML and CSS files are saved in a single folder. Have the folder compressed/zipped before uploading.

La Propiedad de Posición (Position Property Video: la propiedad de posicion)

Además de la propiedad de visualización (display property) y el modelo de caja, la propiedad position también puede ser utilizada para mover elementos en una página. Utiliza esta propiedad solo cuando lo necesites, de lo contrario, usa la propiedad display y el modelo de caja para posicionar tus elementos. La especificación de CSS nos ofrece 6 valores de posición, sin embargo, cubriremos las 4 más importantes: **static** (estático), **relative** (relativo), **absolute** (absoluto) y **fixed** (fijo)). Cada propiedad sirve a un propósito específico, por lo que entenderlo es la clave para dominar los diseños basados en CSS.

Considera lo siguiente:

HTML:

```
<div id="container">

  <div id="box-1">

    </div>
```

```
<div id="box-2">

</div>

<div id="box-3">

</div>

</div>
```

CSS:

```
#container{

  background-color: yellow;

  display: inline-block;

}

#box-1, #box-2, #box-3{

  height: 100px;

  width: 100px;

  margin: 10px;

  display: inline-block;

}

#box-1{

  background-color: green;

}

#box-2{

  background-color: pink;

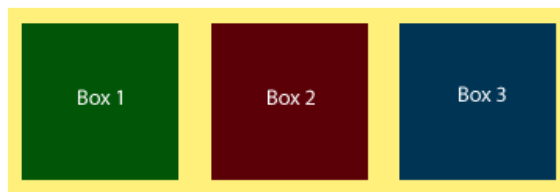
}

#box-3{

  background-color: blue;

}
```

Esto resulta en:



Static

El posicionamiento estático (static) viene predeterminado. Para cualquier elemento que tenga posición, static es parte del flujo de documentos normal. Las reglas sobre dónde se ubica y cómo afecta otras cajas o cuadros está definido por el modelo de caja (puedes revisar nuevamente la lección sobre el **modelo de caja**).

Un elemento situado de forma estática no tomará en cuenta ningún valor para las propiedades top (arriba), right (derecha), bottom (abajo) y left (izquierda), como tampoco ninguna declaración z-index. Para poder usar tales propiedades, tu elemento que necesariamente usar posicionamiento absolute, relative o fixed.

Absolute

Los elementos posicionados de forma absoluta (absolute) están completamente eliminados del flujo normal de documentos. En cuanto a los elementos que lo rodean, el elemento posicionado absolutamente no existe. Es como si la propiedad de visualización (display) del elemento estuviera fijada en “ninguna”. Entonces, si quieres mantener el espacio para que otros elementos no lo utilicen, debes seguir otros caminos.

Para fijar la ubicación de un elemento absolutamente posicionado se utilizan las propiedades top, right, bottom y left. Normalmente definirás solo 2, ya sea top o bottom, o left o right. Por defecto, cada una tiene un valor de ajuste automático.

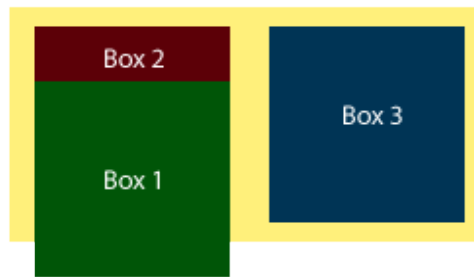
La clave para entender el posicionamiento absoluto, es comprender cuál es el origen. Si top es fijado en 20px, la pregunta sería: ¿20px de dónde?

Un elemento posicionado de manera absoluta (absolute) se ubica en relación al primer elemento padre (parent), el cual tiene una posición distinta de la estática aplicada a este. Si ningún elemento padre cumple esta condición, el elemento de posicionamiento absoluto se ubica en relación a la ventana del documento.

Utilizando el posicionamiento absolute en el box-1:

```
#container{  
    background-color: yellow;  
    display: inline-block;  
}  
  
#box-1, #box-2, #box-3{  
    height: 100px;  
    width: 100px;  
    margin: 10px;  
    display: inline-block;  
}  
  
#box-1{  
    background-color: green;  
    position: absolute;  
    top: 40px;  
}  
  
#box-2{  
    background-color: pink;  
}  
  
#box-3{  
    background-color: blue;  
}
```

Da como resultado:



Relative

Los elementos posicionados de forma relativa (relative) se ubican de acuerdo a las mismas propiedades top, right, bottom y left, las cuales pueden cambiarse fácilmente. En cierto sentido, agregar posicionamiento relativo es similar a agregar un margen, pero con una importante diferencia: los elementos alrededor de un elemento posicionado de manera relativa actúan como si ese cambio no existiera. Lo ignoran.

Usando posicionamiento relativo en el box-2:

```
#container{  
    background-color: yellow;  
    display: inline-block;  
}  
  
#box-1, #box-2, #box-3{  
    height: 100px;  
    width: 100px;  
    margin: 10px;  
    display: inline-block;  
}  
  
#box-1{  
    background-color: green;  
}
```

```
#box-2{

    background-color: pink;

    position: relative;

    top: 20px;

    left: 20px;

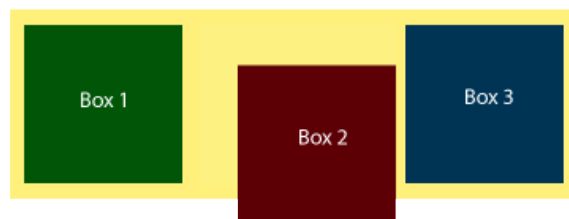
}

#box-3{

    background-color: blue;

}
```

Da como resultado:



Fixed

El posicionamiento fixed o fijo, actúa de manera similar al posicionamiento absoluto con un par de diferencias. Primero, el elemento de posicionamiento fixed siempre se posiciona en función de la ventana del navegador y toma los ya ahora conocidas propiedades top, right, bottom y left. La segunda diferencia es inherente al nombre, los elementos posicionadas de manera fija (fixed) están fijos. No se mueven a medida que se desplaza la página.

Extras

Un recurso genial para saber cómo centrar

elementos: <http://designshack.net/articles/css/how-to-center-anything-with-css/>

Una forma fácil de centrar un elemento dentro de un contenedor (container):

HTML:

```
<div id="container">
```

```
<div id="center-me">

</div>

</div>
```

CSS:

```
#container{

    width:200px;

    height:200px;

    background: blue;

    position:relative;

}

#center-me{

    background: red;

    height: 100px;

    width: 100px;

    margin: auto;

    position: absolute;

    left: 0;

    bottom: 0;

    top: 0;

    right: 0;

}
```

Otra forma de centrar horizontalmente un elemento:

HTML:

```
<div id="container">

    <div id="center-me">

</div>
```

```
</div>
```

CSS:

```
#container{  
  
  width:200px;  
  
  height:200px;  
  
  background: blue;  
  
  position:relative;  
  
}  
  
#center-me{  
  
  background: red;  
  
  height: 100px;  
  
  width: 100px;  
  
  margin: 0 auto;  
  
}
```

Compatibilidad del Navegador

Cada navegador puede entregar atributos predeterminados a los elementos HTML, provocando que los sitios se vean diferentes dependiendo de la versión del navegador o si estás viendo el sitio en IE, Firefox, Chrome, etc. Por ejemplo, en Internet Explorer, un elemento H1 puede tener cierto margen (margin) y relleno (padding) haciendo que tu sitio se vea de una forma, mientras que Chrome y Firefox pueden darle a H1 un margen y relleno ligeramente diferente. Estas diferencias podrían ser un dolor de cabeza, especialmente si estás tratando de hacer que tu sitio se vea consistente en los distintos navegadores o versiones de navegador.

Para resolver los problemas entre navegadores, es recomendable utilizar 'RESET' y 'NORMALIZE' y definir cuáles debieran ser los atributos predeterminados (en vez de confiar en los valores predeterminados que vienen con cada navegador). De hecho, a los reclutadores les encanta preguntar sobre esto y ver si sabes como hacer que tu sitio se vea consistente en los diferentes navegadores. Cuando te pregunten esto, deberías poder explicar (con seguridad)

qué significa reset/normalizing y por qué se utilizan. Además, deberías contarles que es importante validar tu HTML/CSS a menudo, puesto que hacer esto te da una idea de cómo arreglar inconvenientes que podrían ocasionar problemas de compatibilidad entre navegadores (y frecuentemente, un HTML/CSS inválido puede conducir a un comportamiento extraño puesto que el navegador no arreglaría el HTML/CSS inválido de forma correcta).

CSS Reset

Esta es una muy buena explicación de los CSS resets, por qué los usamos y algunos ejemplos:

<http://perishablepress.com/a-killer-collection-of-global-css-reset-styles/>

Normalize

Una alternativa a los CSS resets, ampliamente utilizada, es normalize.css, que también cubre los elementos de HTML5. Para más información respecto a lo que hace y por qué puede ser una mejor opción que los CSS resets, ingresa aquí:

<https://github.com/necolas/normalize.css/>

Preguntas Frecuentes

P: ¿Puedo agregar múltiples clases a un elemento?

R: Sí. Simplemente separa los nombres de clase con un espacio. Ej:

```
<div class="nav side-bar"></div>
```

P: ¿Puedo agregar una clase y un id al mismo elemento?

R: Sí. Un elemento puede tener un id y múltiples clases. Ej:

```
<div id="primary-nav" class="nav side-bar"></div>
```

P: ¿Tengo que agregar una clase o un id para apuntar a un elemento anidado?

R: No como una regla. Considera el siguiente ejemplo:

```
<ul id="nav-menu">
  <li><a href="#">Home</a></li>
  <li><a href="#">Dashboard</a></li>
```

```
<li><a href="#">About</a></li>

<li><a href="#">Contact Us</a></li>

</ul>
```

Para focalizarse en los enlaces (links) y eliminar el subrayado de estos, podemos hacer lo siguiente:

```
#nav-menu li a{

  text-decoration: none;

}
```

Esto le dice al navegador que primero ubique los elementos con id nav-menu, luego que encuentre toda la lista de ítems dentro de las listas desordenadas y que, finalmente, encuentre los enlaces dentro de dichas listas.

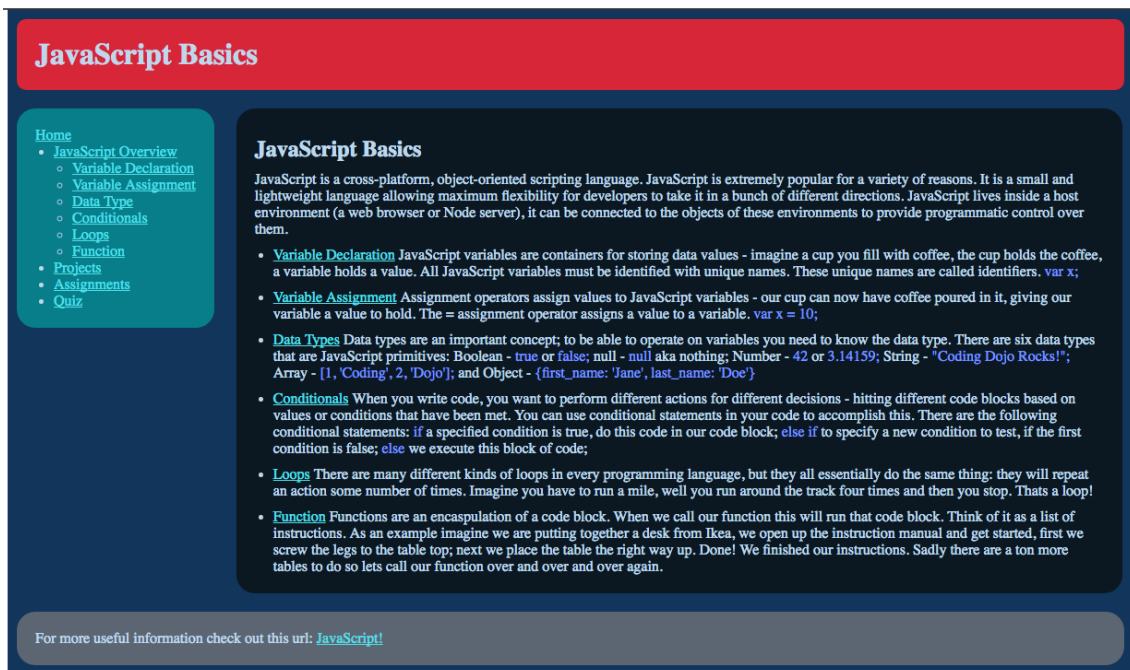
Por lo general, debes hacer tu código más fácil de leer evitando agregar demasiadas clases e ids.

Conceptos Básicos de JavaScript

Recrea la imagen de más abajo utilizando solo lo que has aprendido de HTML y CSS. Usa display: inline-block/inline/block para hacer todas las actividades. Puedes bajar la imagen aquí: [Principios Básico de JavaScript](#).

Antes de enviar tu trabajo:

- Recuerda que estás acá para aprender: No intentes encontrar el código fuente original de estas imágenes.
- Asegúrate de haber validado tu HTML.
- **Además, NO lo hagas responsivo, simplemente hazlo para un ancho fijo (establece el ancho en 970px).** Si tienes tiempo durante el bootcamp, visita nuestro curso de diseño responsivo más tarde, pero por ahora primero intenta dominar los principios de CSS para un ancho fijo. El curso de diseño responsivo está ubicado al final del track de Principios Básicos de la Web, por lo que puedes hacerlo más tarde en tu bootcamp.



Consejos Útiles - basic 6 y construcción en capas (Video)

Cuando posiciones elementos, utiliza principalmente margin, padding, display, vertical-align, width y height (los “basic 6” o “6 básicos”). NO uses float, position:relative o position:absolute. Focalízate en esos 6 básicos cuando desarrolles las actividades. Además, evita usar Stack Overflow, sobre todo al principio, considerando la complejidad de las típicas respuestas sobre posicionamiento en CSS que puedes encontrar ahí.

Demo

Construyendo en Capas

HTML Primero y CSS segundo

Ten cuidado con tomar esto fuera de contexto. Cuando recomendamos que sea HTML primero y CSS segundo, nos referimos a que es necesario situar el primer elemento de HTML y agregarle estilo. A continuación se añade el segundo elemento en HTML y también se le agrega estilo. Si por el contrario, escribes todo el HTML primero y después aplicas CSS, complicarás en extremo el desarrollo de tus proyectos. Este enfoque de verdad funciona y te permite mantener un ritmo constante. Por lo tanto, mantén un buen aspecto de tu proyecto a medida que construyes en capas, como haciendo un pastel.

Construyendo en Capas

Construir sitios web complejos es más simple de lo que piensas. Los sitios webs se dividen en secciones, que serían como las capas de la torta. Primero creamos la base con la capa de texto, y así hasta que terminamos de agregar capas a nuestro pastel. En HTML y CSS hacemos esta primera sección (la base) que llamamos el “contenedor” o “envoltorio”. Agregamos el HTML al contenedor, luego el CSS y chequeamos el resultado. Con este enfoque, construir incluso los sitios web más complejos, se transforma en algo tan sencillo como “trazar tus bloques” será luego de alguna práctica. Cada sección se construye de manera independiente, desde la capa inferior div, hacia arriba e izquierda a derecha, como se muestra en el video.

Outline es una propiedad muy útil que se puede aplicar a todos los elementos. Sin un color de fondo o borde en cada elemento, es difícil adivinar el inicio y el comienzo de un elemento, por lo que es justo aquí donde outline puede ayudar. Outline aplica un borde en todos los elementos sin el ancho adicional que la propiedad de borde (border) agregaría, lo que en su mayoría se usa para desarrollo. A diferencia de la propiedad border, cuando quitamos la propiedad outline en nuestro CSS, nuestros elementos no alterarán su tamaño. Outline es una gran propiedad de estilo para darle a nuestro proyecto un pseudo cuadriculado en todos los elementos para una mejor visualización.

/style.css

```
* { outline: 1px dotted red; }
```

Portafolio

Descompón la imagen de más abajo en HTML y CSS apropiado.


Antes que envíes tu trabajo:

- Descarga las [imágenes recortadas](#) para este ejercicio.
- Asegúrate de haber validado tu HTML en <https://validator.w3.org/>
- Asegúrense de que han refactorizado su código. No usen tags <div>, clases o ids innecesarios.

BONUS: Implementen las páginas "La Mode", "Family Contacts" y "fiveEleven".

Jayne Doe

Master of shadows and the Internet!




About...


Hello! I'm an extremely driven and creative Full Stack Developer who is currently open for career opportunities as a front-end or back-end web developer in the Greater Seattle Area.

I'm a recent graduate of Coding Dojo, a coding school located in Bellevue, WA that teaches 3 full stacks in 3 months. I'm capable of learning new technologies very quickly, and am always looking for opportunities to further expand my skills and grow as a developer.


[Continue reading...](#)

La Mode


La Mode is an Ecommerce website for designed to market various clothing products. Users are able to view the available garments, select their desired quantity, and compile a shopping cart for making a final purchase.

Technologies:


La Mode

Family Contacts


Family Contacts is a free application for managing, sharing, and visualizing your family relationships for both extended and immediate family. As an essential feature of the project, your log-in information determines who you can see and reveals how people are related to you.

Technologies:

Family Contacts





fiveEleven

Five Eleven is a data visualization project built on the Python software stack. The application transforms the developer job-hunt into a more visual, user friendly experience. Based on location, technology popularity, and other vast data sets, users may easily visualize the varying sizes and concentrations of the junior developer job market across the nation.

Technologies:

fiveEleven

jdoe@gmail.com | 206-555-1212

 [jaynedev](#)  [jayne-doe](#)  [jdtweets](#)  [jaynedev](#)

Consejo Útil

En la sección anterior, mostramos podías ajustar el ancho y la altura de una imagen así:

```

```

Como ya has aprendido CSS, NO agregues ancho y alto directamente en la etiqueta de imagen, más bien haz que el archivo CSS especifique esos atributos. En el mismo sentido, deja de usar cualquier etiqueta html que hayas utilizado en el pasado (come or <u>), puesto que si algo está en negrita o subrayado, esto debiera definirse en el CSS y nunca en el HTML.

Mostrando Bloques

¡Esta actividad es opcional! Solo hazla si completaste las actividades del día.

Utiliza **margins** y **padding**s para ajustar los espacios entre divisiones, y usa la propiedad **display** para poder poner cada bloque en su lugar correcto.

Puedes descargar las imágenes utilizadas en el siguiente enlace: [Imágenes](#)



-

Trabaja en el ejercicio y sube tu código a continuación. Pon tu HTML, CSS y archivos de imagen en una carpeta que debes comprimir antes de subir.

“Acerca de Python” Actividad de CSS

Descompón la imagen de más abajo en HTML y CSS. Puedes descargar la actividad aquí: [Python](#)

Antes de enviar tu trabajo:

- Recuerda que estás acá para aprender: No busques el código fuente original de estas imágenes.
- Usa [éstas imágenes](#) para el ejercicio.
- Asegúrate de haber validado tu HTML y que no estás usando ningún float: izquierda o derecha.
- Para “>>” en la lista, no te preocupes de que se vea exacto a “>>”. Puedes simplemente usar un círculo normal para esta actividad.
- **NO lo hagas responsivo, simplemente haz que esto funcione para un ancho fijo (establece el ancho en 970px).** Te adentrarás en nuestro curso de diseño responsivo más adelante. Por ahora, intenta primero dominar los principios de CSS para ancho fijo..



•

Desarrolla este ejercicio y luego sube tu trabajo aquí. Debes poner todos tus archivos html, css e imágenes en una carpeta y comprimirla antes de subirla.

Internet

Para esta actividad, queremos que veas cuantas veces usas las etiquetas <div>. Siempre que tengas un elemento dentro de una división, significa que no necesitas esa división. Por ejemplo, imagina tienes un código como este:

```
<div id="banner">

  <h1>Word Stream</h1>

</div>

<div id="left_nav">

  <div class="title">

    <h3>WordStream for PPC</h3>

  </div>
```

```
<div class="heading">

  <div class="emphasis">Drive more Profits Through PPC!</div>

</div>

</div>
```

Mirando este código, ¿puedes identificar las divisiones innecesarias? ¡Esperamos que sí! Nuevamente, cuando tengas un elemento dentro de una división, esto será redundante. Piensa cómo puedes refactorizar el código anterior para hacerlo más simple. Una opción es hacer algo como esto:

```
<h1 id="banner">Word Stream</h1>

<div id="left_nav">

  <h3>WordStream for PPC</h3>

  <h4>Drive more Profits Through PPC!</h4>

</div>
```

Esperamos que hayas entendido el punto. ***Poner un solo elemento en otro elemento es como poner un regalo de navidad en un paquete y luego poner ese paquete dentro de otro paquete más grande (que finalmente no es necesaria).***

Los mejores programadores siempre desarrollan con menos código, y por ende, queremos que practiques revisar tu código y tratar siempre de hacer las cosas de la manera más simple posible. Pocas líneas de código, menos texto, siempre será mejor. Recuérdalo siempre.

Ahora, duplica la página de más abajo en función de lo que acabas de aprender.



The Internet

The World Wide Web

So, what's an Internet?

An **interconnected network** of computers all around the world!

[WATCH A VIDEO](#)

[MORE RESOURCES](#)

Reinvent your career in just 14 weeks



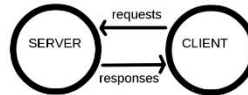
[Learn More >>>](#)



Clients and Servers

Computers connected to the **Web** are called **clients** and **servers**.

- Clients** are the typical Web user's Internet-connected devices (for example, your computer connected to your Wi-Fi, or your phone connected to your mobile network) and **Web-accessing software** available on those devices (usually a web browser like Firefox or Chrome).
- Servers** are computers that store **webpages**, **sites**, or **apps**. When a client device wants to access a webpage, a copy of the webpage is downloaded from the server onto the client machine to be displayed in the user's web browser.



So What Actually Happens?

- The browser goes to the **DNS server** and finds the real address of the **server** that the website lives on.
- The browser sends an **HTTP request** message to the server asking it to send a copy of the website to the client.
- If the server approves the client's request, the server sends the client a **"200 OK" message**, which means "Permission granted! Here you go!", and then starts sending the website's files to the browser as a series of small chunks.
- The browser assembles the small chunks into a complete website and displays it to you!



The Big 3 - HTML, CSS, JavaScript

While the server can process information in many different languages, the files that they serve to the client are *always* going to be some combination of HTML, CSS, and JavaScript!

[Learn more about the Big 3 here!](#)

- Es recomendable que reinicies y normalices los atributos CSS antes de que agregues tus propios atributos CSS.
- Descarga las [imágenes recortadas](#) para este ejercicio.
- Asegúrate de haber validado tu HTML y que no estás utilizando ningún float: izquierda o derecha.
- Asegúrate de haber rechequeado tu código para evitar usos redundantes de ids o classes. Cerciórate de que no tener ningún elemento individual dentro de una división.
- De ser necesario, refactoriza tu código HTML.
- Desarrolla este ejercicio y luego sube tu trabajo aquí. Debes poner todos tus archivos html, css e imágenes en una carpeta y comprimirla antes de subirla.

Media Queries

@media

CSS utiliza la regla @media para detectar información sobre el dispositivo utilizada para acceder al sitio web. Por ejemplo, puede ser utilizado para detectar el tamaño del dispositivo,

sus capacidades y su orientación. Para este curso, lo usaremos para determinar el tamaño del dispositivo, pero ten en cuenta de que es capaz de mucho más.

Diferentes dispositivos

Tenemos 3 categorías básicas de dispositivos: teléfonos, tablets y desktops (computador de escritorio). Por supuesto que hay más, pero estos son los 3 principales. Nos enfocaremos en utilizar la consulta medios (media queries) para determinar qué categoría de dispositivo está siendo utilizado y responder apropiadamente. Para lograrlo, necesitaremos saber cómo definimos cada una de estas categorías.

Dispositivo	Tamaño
Teléfono	Ancho menor o igual a 480px
Tablet	Ancho entre 481px y 1023px
Desktop	Ancho mayor o igual a 1024px

Unidades Flexibles

El diseño web responsivo se basa en hacer nuestro sitio web fluido y proporcional. De esta forma, a veces no hace sentido implementar elementos en la página que tengan un tamaño fijo en píxeles. Todavía usaremos píxeles, por ejemplo, normalmente usamos píxeles para el relleno (padding), puesto que normalmente queremos la misma cantidad de relleno sin importar el tamaño de la pantalla, pero ahora usaremos también unidades flexibles para los elementos de la página que necesitan ser fluidos. Por ejemplo, podemos usar porcentaje (%) para configurar un elemento con el fin de que tome una cierta cantidad del ancho disponible, el cual es determinado por el tamaño de la pantalla. Otras unidades flexibles incluyen em y rem. La unidad rem es un multiplicador del tamaño de fuente predeterminada del navegador. La unidad em es un multiplicador del padre del elemento. Experimenta con estas unidades para entender cómo se comportan.

Ejemplo

Digamos que tenemos 4 botones en nuestro sitio web a los cuales quisiéramos cambiar el color dependiendo del tamaño de la pantalla. A los usuarios de desktops les daremos botones rojos, los usuarios de tablets tendrán botones azules y los usuarios de teléfonos tendrán botones amarillos. Le daremos a los botones la clase `.navbtn`.

Para comenzar, así es como configuraríamos una consulta de medios (media query) en nuestro CSS para ver si la pantalla del usuario tiene 480px de ancho o menos. Si es así, entonces apuntaremos a todos los elementos con la clase `.navbtn` y configuraremos su fondo en amarillo.

```
@media only screen and (max-width: 480px) {  
  
    /* we will set these stylings only if the device is a screen with a width of 480px or less */  
  
    .navbtn {  
  
        background-color: yellow;  
  
    }  
  
}
```

Recuerda que CSS se lee de arriba a abajo, por lo que si las condiciones de la consulta de medios se cumplen, los estilos dentro podrían anular cualquier estilo que hayas puesto en `.navbtn` en el código arriba de la consulta de medios. Cualquier código que hayas escrito abajo de la consulta de medios podría invalidar los estilos que pusimos aquí.

Intenta cambiar por tu cuenta los colores de los botones de rojo a azul o amarillo, dependiendo del tamaño de la pantalla. Vuelve a ver el video de arriba si es que necesitas una pista. Luego cambia la orientación de los botones en función del tamaño de la pantalla.

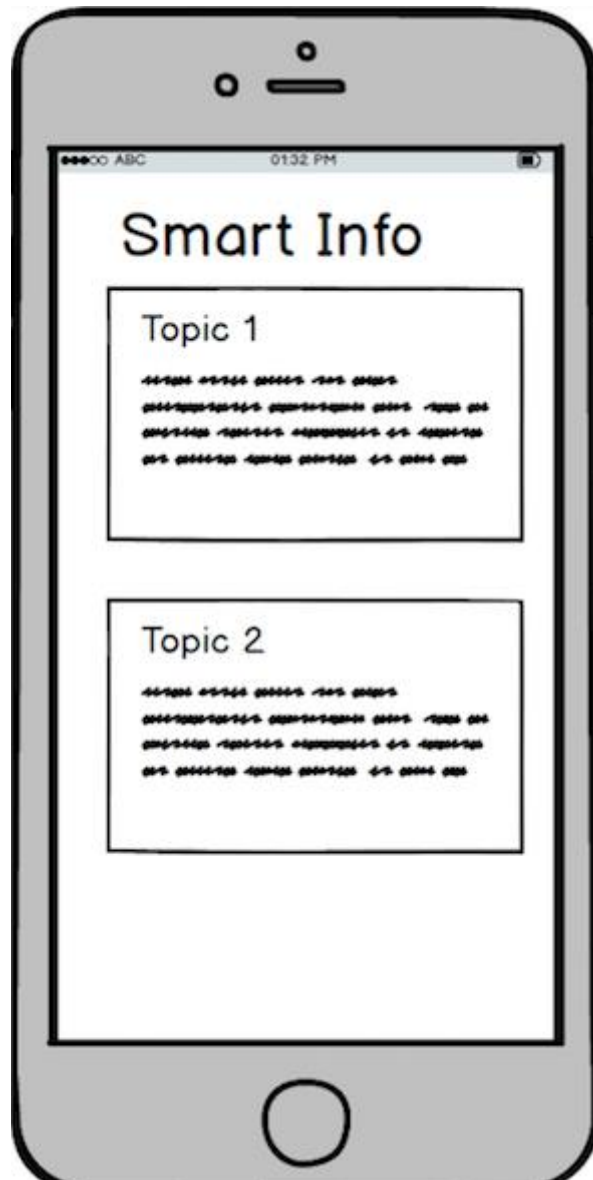
- Para usuarios de desktop, ordena los 4 botones en una línea.
- Para usuarios de tablet, pon 2 botones por línea.
- Para usuarios de teléfonos, dale a cada botón su propia línea.

El Sistema Grid (cuadrícula)

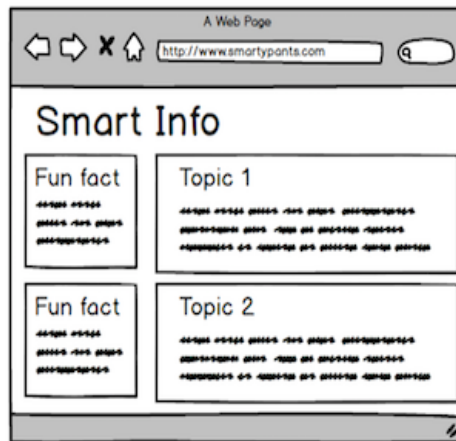
¿Por qué usar el sistema grid?

Para agilizar el proceso de diseño de sitios web responsivos, muchos desarrolladores se imaginan la pantalla dividida verticalmente en columnas distribuidas uniformemente. Lo anterior combinado con dividir la pantalla en filas horizontales, permite que posicionar nuestros elementos en la página sea mucho más fácil.

No necesitas usar la librería de alguien más para tener acceso al sistema grid. Ya sabes cómo usar las consultas de medios, ahora combínalos con unidades flexibles (por ejemplo, para configurar el ancho usa porcentajes en lugar de píxeles) y puedes crear tu propio grid.



Aquí, hemos pensado primero en el diseño de nuestro sitio web para dispositivos móviles. La idea es que nuestra información se muestre de manera simple, esto es, una lista vertical que un usuario puede ver fácilmente desplazándose en el teléfono. Usando un grid, cada tema obtiene todas las columnas disponibles. Para este ejemplo, asumamos que estamos dividiendo nuestra pantalla en 4 columnas, por lo tanto, cada columna obtiene el 25% de ancho disponible.



Ahora pensemos en un dispositivo más grande. Si tenemos más espacio, podemos agregar un dato curioso (fun fact) a cada fila. Daremos una columna de las 4 disponibles para este contenido extra y dejaremos las 3 siguientes para el contenido principal.

Lo anterior significa que en nuestro html le daremos diferentes clases a nuestros elementos. Por ejemplo, nuestros elementos opcionales deberían tener una clase que nos permita esconderlos cuando no son requeridos. Para que nuestro código sea fácil de leer, llamemos a nuestras clases con un sistema para poder identificar para qué tamaño de pantalla están dirigidas y cuántas columnas ocuparán. Nuestras clases seguirán el patrón **tamaño de la pantalla - número de columnas**. Por ejemplo, la clase `.med-1` significará que para pantallas medianas, el elemento ocupará una columna.

```
<div class="col sm-0 med-1">

  <h5>Did you know?</h5>

  <p>If you could fold a piece of paper 42 times, it would be tall enough
to reach the moon.</p>

</div>
```

En nuestro css, realizaremos consultas de medios para determinar qué clases estamos usando y cuáles ignoramos, dependiendo del tamaño de la pantalla.

```
@media only screen and (max-width: 480px) {

  .sm-0 {

    /* we will not display anything with the class .sm-0 if the screen
's width is less than 480px */
```

```
        display: none;

    }

}

@media only screen and (min-width: 481px) {

    .med-1 {

        /* anything with the class .med-1 will get 25% of the available width if the screen's width is greater than 480px */

        width: 25%;

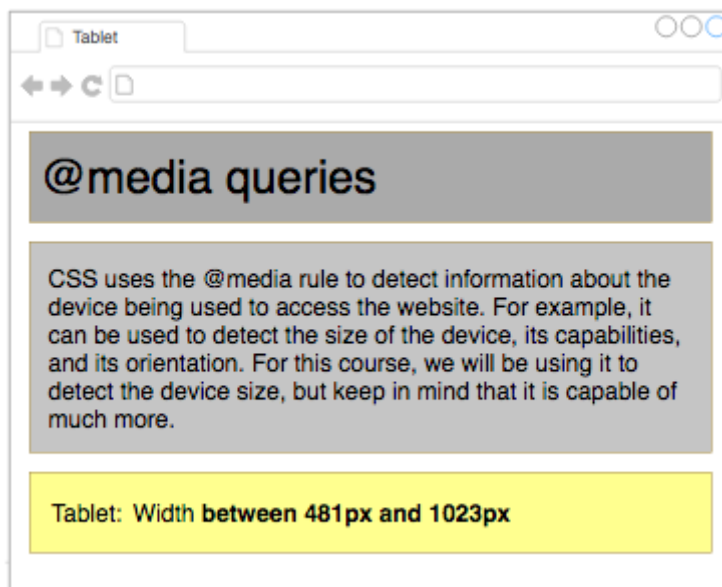
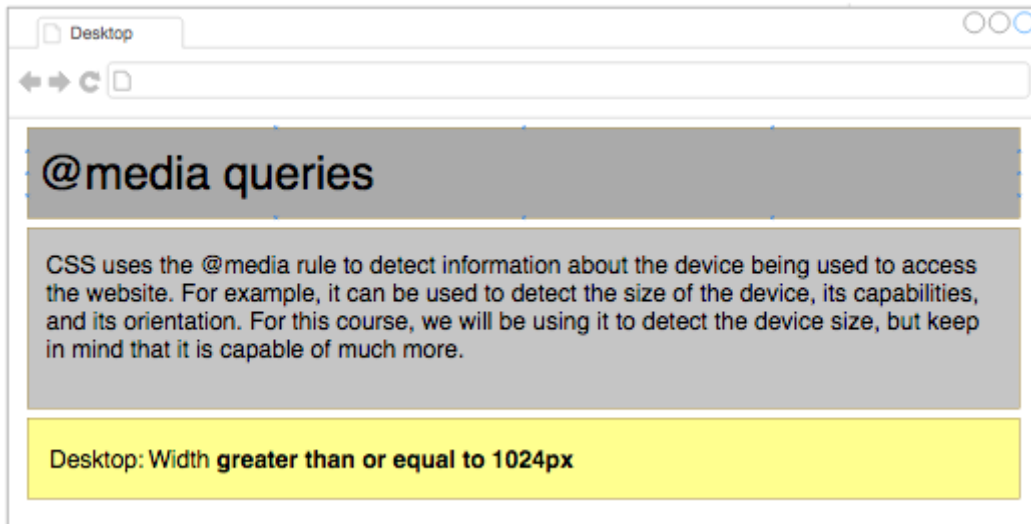
    }

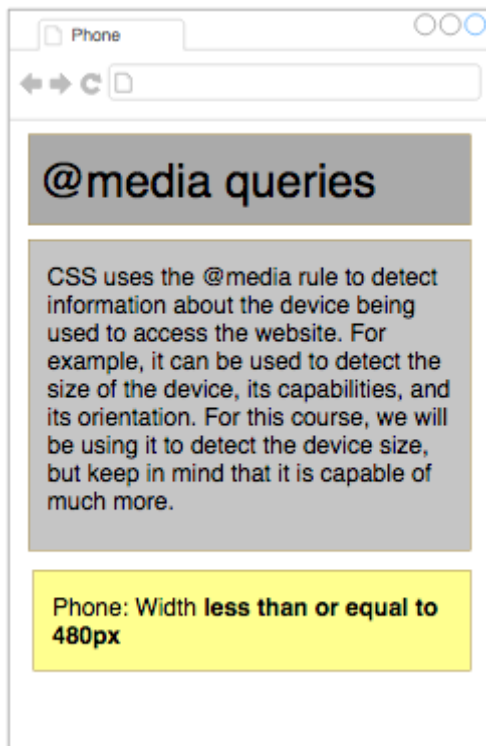
}
```

En combinación con la habilidades de css, podrás posicionar elementos en una página y construir tu grid o cuadrícula responsiva en muy poco tiempo. Te recomendamos armar tu propia cuadrícula responsiva en vez de utilizar las que vienen diseñadas en librerías como Bootstrap o Materialize. Si creas tu propia cuadrícula, incluso si usaras una librería a futuro, entenderás como funciona este sistema y cómo usarlo de manera efectiva.

Simplemente Responsivo

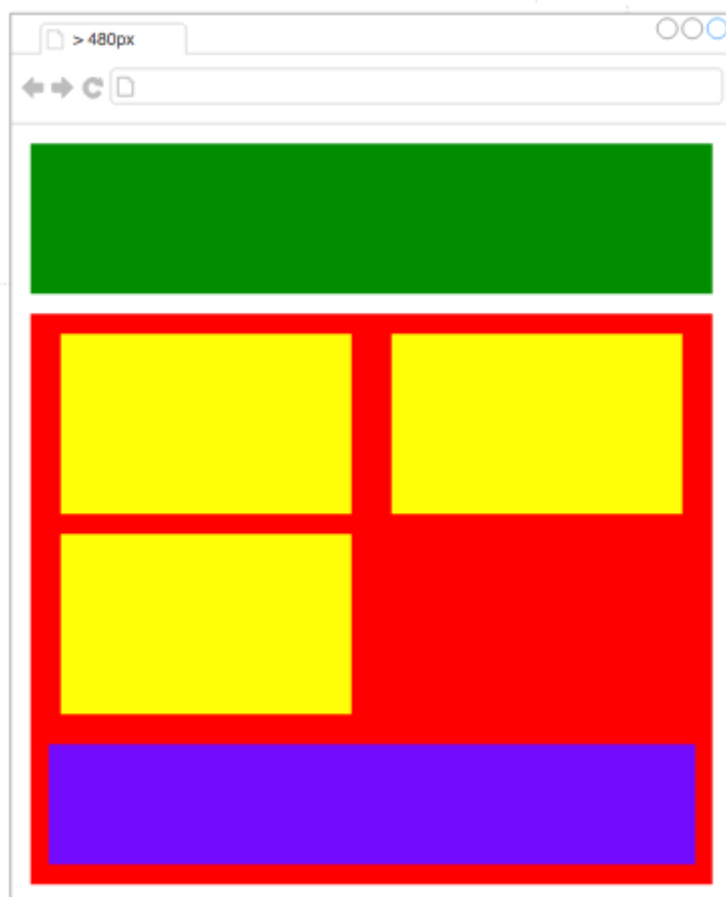
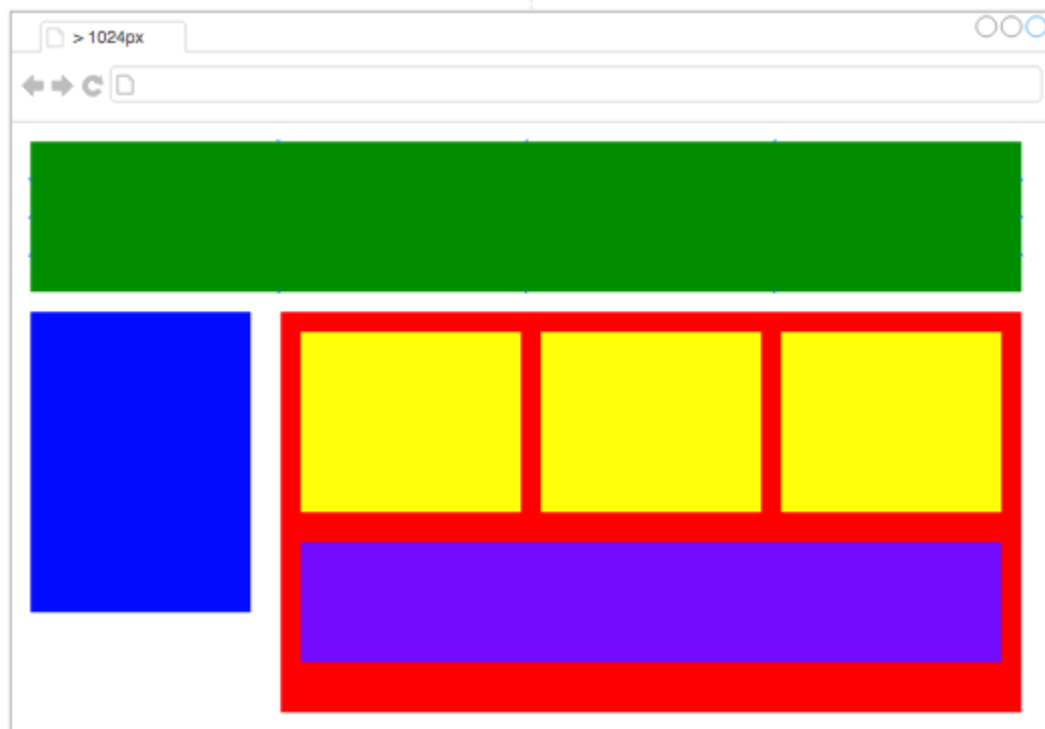
Crea el sitio web a continuación utilizando consultas de medios (media queries). Para este ejercicio, no necesitas cambiar las etiquetas de título a medida que cambia el tamaño de la pantalla.

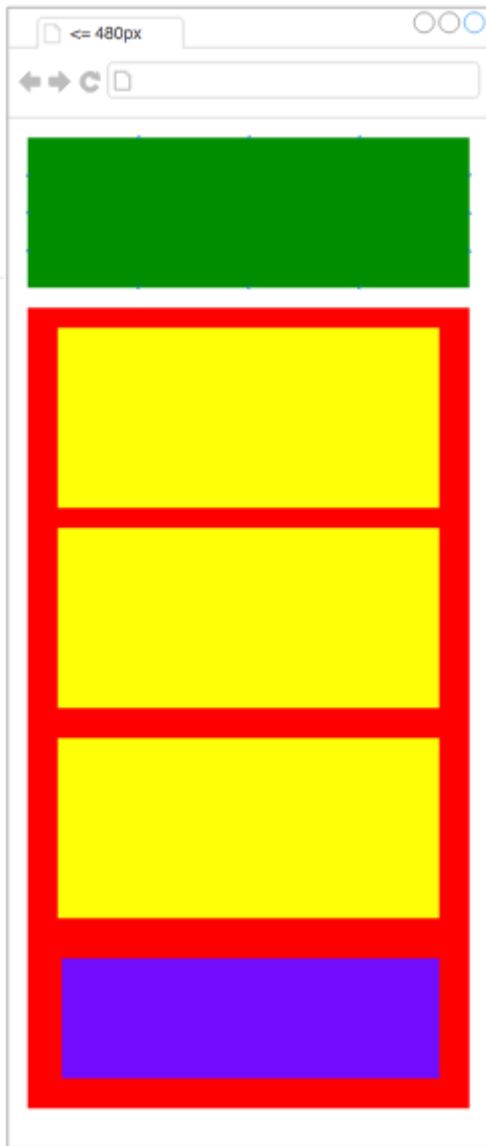




Bloques Responsivos

Convierte a responsivo lo realizado en la actividad 'Trazando tus bloques', como se muestra más abajo. Utiliza 3 puntos de quiebre, 480px o menos para pantallas de teléfono, 481px o más para pantallas de tablet y 1024px o más para pantallas de pc o laptops. Te en cuenta que la caja azul desaparece en las pantallas medianas y pequeñas.





Algoritmo IV

Esperamos que ahora te sientas cómodo con los desafíos de predicción. Si no es así, vuelve a hacer los desafíos de predicción de algoritmos al menos 2 veces antes de que finalice este semana.

1. Dados un array y un valor Y, cuenta e imprime (print) el número de valores del array que sean mayores que Y.
2. Dado un array, imprime los valores máximos (max), mínimos (min) y promedio (average) para el array.
3. Dado un array de números, crea una función que dé como resultado un nuevo array donde los valores negativos se reemplacen por el texto (string) 'Dojo'. Por ejemplo, `reemplazarNegativos([1,2,-3,-5,5])` debiera devolver `[1,2, "Dojo", "Dojo", 5]`.

4. Dado un array y su respectivo índice, remueve los valores en el rango del índice dado(acortando el array). Por ejemplo, `removeRango([20,30,40,50,60,70],2,4)` debiera devolver `[20,30,70]`.

Ahora, vuelve a algorithm app (algorithm.codingdojo.com) y haz los 13 desafíos en menos de 2 minutos cada uno. Si no puedes, por favor dedica hasta dos hora rehaciendo los desafíos de algoritmos (tanto los desafíos de predicción como los 13 desafíos).

¡Libro de Algoritmos de Coding Dojo!

Entender algoritmos es muy importante y es algo que harás casi todos los días en el Dojo.

Escribimos un libro en 2016 para ayudar a nuestros estudiantes a mejorar en algoritmos. Este libro es muy completo y debieras estudiarlo durante el bootcamp y, especialmente, luego de graduarte.

Muchos de nuestros estudiantes han dedicado tiempo preparando y estudiando los conceptos en el libro, y así se sienten más confiados y preparados durante las entrevistas laborales. Usa el libro como un recurso complementario para tus sesiones de algoritmos en el Dojo.

[¡Descarga el libro!](#)

Bootstrap, Foundation y LESS

Bootstrap

Objetivos

1. Familiarizarse con Bootstrap, una librería/framework muy popular que utilizan muchos desarrolladores.
2. Como desarrollador, lo más probable es que uses un archivo CSS creado por otros. Esta es una buena oportunidad para que sepas cómo aprovechar un framework de css creado por otras personas.
3. Que puedas apreciar el diseño simple que ofrece Bootstrap.

Introducción

¿Qué es Bootstrap?

En palabras de sus creadores, bootstrap es un “elegante, intuitivo y poderoso framework mobile-first y front-end, para el desarrollo de la web más rápido y fácil”.

Para usar Bootstrap, primero pone las siguientes líneas en la sección <head> de tu archivo:

```
<!-- Required meta tags -->

<meta charset="utf-8">

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

<!-- load Bootstrap -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" 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>
```

Ahora puedes continuar **y agregar clases Bootstrap a tus elementos en HTML** con el fin de lograr fácilmente un diseño consistente, limpio y responsivo.

Actividad - Qué aprender primero

Luego de dedicar algunos minutos a revisar todos los ejemplos

en <https://getbootstrap.com/docs/4.1/examples/>, por favor, crea un sitio web simple utilizando los elementos de Bootstrap a continuación. Ignora todos elementos de bootstrap que no sean los siguientes:

Containers (contenedores) - <https://getbootstrap.com/docs/4.0/layout/overview/> (solo revisa la clase “container” y sáltate todo lo demás)

Buttons (botones) - <https://getbootstrap.com/docs/4.0/components/buttons/>

btn

btn-default

btn-primary

btn-success

Jumbotron - <https://getbootstrap.com/docs/4.0/components/jumbotron/>

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

Forms (formularios)

form-group

form-control

Sistema grid o de cuadrícula (para posicionar y dimensionar elementos):

row

col-md-4

col-md-offset-2

Como aún no has aprendido jQuery, no te preocupes de ninguna parte de javascript en Bootstrap, es decir, cualquier cosa que hace algo cuando cliques en un elemento HTML específico (ej: botón desplegable, etc). Sólo enfócate en cómo hacer que los elementos de HTML se vean bien.

No utilices más de 3 horas aprendiendo Bootstrap. Hay muchas otras características dentro de Bootstrap, pero puedes aprenderlas más tarde, quizá cuando construyas tu primer proyecto.

Al final de esta actividad, deberías sentirte cómodo creando un sitio como <https://getbootstrap.com/docs/4.1/examples/jumbotron/>

(excepto por el menú desplegable).

LESS

Preprocesador CSS

NOTA: Esta actividad requiere el uso de un servidor local (utilizando MAMP o WAMP). Puedes leer sobre las herramientas ahora y volver a esta actividad luego de haber instalado y usado un servidor en tu primer stack.

Hay un montón de herramientas geniales disponibles para ayudarte a organizar mejor tu código CSS, incluyendo preprocesadores como LESS y SASS. Los preprocesadores te permiten escribir tu código en una versión abreviada de CSS, que luego se transforma en un CSS regular para el navegador. Dos de las herramientas más populares son LESS y SASS. Ambos compilan su versión de CSS en CSS normal, pero lo hacen de forma diferente. LESS es una librería de JavaScript y puede compilar en el lado del cliente, mientras que SASS solo trabaja en el lado del servidor. SASS requiere estar un poco familiarizado con la línea de comando y Ruby. Por esta razón usaremos LESS.

Puedes leer las características del lenguaje LESS y cómo usar mix-ins, en la documentación de LESS acá: <http://lesscss.org/features/>

Uso del lado del cliente (client-side)

El lado del cliente es la forma más simple para empezar a desarrollar con LESS adecuadamente, pero en producción, cuando el rendimiento y la fiabilidad es importante, recomendamos pre compilar utilizando node.js o cualquier de las muchas herramientas de terceros disponibles (aprenderás más adelante en el bootcamp cómo precompilar LESS/SASS usando Node.js o Rails).

Practica LESS utilizando las páginas web que creaste en las actividades previas o puedes crear una nueva página web.

Para comenzar, enlaza tu `.less` stylesheets (hojas de estilo) con el atributo `rel` como `"stylesheet/less"`:

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />
```

A continuación, [descarga less.js](#) e inclúyelo en una etiqueta `<script></script>` en el elemento `<head>` de tu página:

```
<script src="less.js" type="text/javascript"></script>
```

O el CDN, aquí:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/less.js/2.5.1/less.min.js"></script>
```

Consejos:

- Asegúrate de incluir tus stylesheets **antes** del script.
- Cuando enlazas más de una stylesheet .less cada una es compilada independientemente. Entonces, cualquier variable, mix-ins o espacio de nombre que defines en una stylesheet, no es accesible en ninguna otra.
- El archivo HTML que crees deberá ejecutarse en un servidor, tu localhost. Sigue las instrucciones a continuación respecto a donde ubicar tus archivos, luego dirígete a "localhost" en tu navegador (escribe localhost en la barra de direcciones). Si llamaste a tu archivo index.html, verás que carga automáticamente. De lo contrario, cliquea en el nombre de tu archivo .html. Si estás teniendo problemas, revisa Inspeccionar Elemento para cerciorarte de que tus stylesheets y tu archivo javascript para LESS se están cargando correctamente.
 - Si tienes WAMP para Windows, pon tus archivos en c:\wamp\www
 - Si tienes MAMP para Mac, pon tus archivos en Applications/MAMP/htdocs/ o dirige tu servidor a la carpeta del proyecto.

Referencias: <http://lesscss.org>

Lectura Complementaria: <https://www.dariobf.com/aprender-less-i/>

Git en 20 minutos

Objetivos

1. Aprender cómo funciona Git y por qué es útil.
 2. Aprender cómo configurar un repositorio git local en tu computador.
 3. Aprender cómo llevar esto a GitHub.
-

Descripción general de Git

Esta es una breve descripción sobre cómo usarás Git. No te preocupes si no te hace sentido la primera vez, simplemente mira el vídeo una o dos veces más hasta que lo entiendas. De lo contrario, sigue adelante mira una demostración del uso de git en la terminal.

Git Commands

- **git init** - inicia el repositorio.
- **git add .** - agrega al área de preparación (staging area), todos los archivos que fueron cambiados desde el último respaldo.
- **git status** - Te muestra todos los archivos que fueron cambiados desde el último respaldo y cuáles ya fueron agregados al área de preparación.
- **git commit -m "..."** - realiza los cambios en el repositorio.
- **git checkout ____** - cambia al nombre de rama (branch) proporcionado en tu repositorio. Esto creará una nueva rama si el nombre entregado no existe.
- **git branch** - muestra todas las ramas de tu git y marca en cuál estás actualmente.
- **git log** - muestra todos los respaldos creados en el repositorio.
- **git blame ____** - muestra quién escribió qué línea de código o, en otras palabras, a quién culpar por esa línea de código particular.
- **git remote add origin ____** - solicita a git agregar un lugar remoto llamado 'origin' a un URL____ remoto.
- **git push** - Lleva los cambios en tu repositorio local al repositorio remoto.
- **git pull** - Extrae los cambios en un repositorio remoto a tu repositorio local.
- **git clone ____** - clona un repositorio remoto en ____ a tu carpeta local.

Forking (bifurcación), Pull Request (solicitud de extracción), Branch (rama) y Merging (fusionar) (Video)

Sigue adelante y mira el vídeo a continuación, pero no te preocupes demasiado sobre cómo exactamente puedes crear una rama, fusionar código, etc. El vídeo es para ayudarte a entender profundamente qué significan estos términos. Una vez que hayas dominado los comandos básicos de Git enseñados acá, puedes profundizar en estas características avanzadas a través de nuestro curso de Git.

Practicando Git

Objetivos

1. Aprender comandos básicos del terminal (ej: pwd, ls, cd, mkdir, touch)
2. Utilizar comandos básico de git

Actividad

1. Familiarízate con los comandos básicos de la terminal, incluyendo: pwd, ls, ls -l, cd, mkdir y touch.
2. Crea un repositorio git local en la carpeta de la actividad anterior y lleva esto a tu GitHub.
3. Encuentra algún repositorio interesante en GitHub que te interese, para clonarlo en tu computador.
 1. Si no puedes encontrar algo interesante, puedes clonar https://github.com/choi5983/street_fighter.git o <https://github.com/choi5983/bomberman.git>. Estos fueron juegos de demostración construidos durante la clase.
4. Más tarde este fin de semana, cuando tengas tiempo, visita el curso completo de Terminal y Git [aquí](#)

Vídeo de Información General

Notas adicionales respecto al vídeo

1. Si nunca has utilizado git, cuando llevas códigos a GitHub, se te preguntará por tus credenciales. Una vez que las agregas, se enviarán tus códigos a GitHub. Basta con hacer esto una vez ya tus credenciales serán recordadas y utilizadas automáticamente cuando quieras llevar código a GitHub.
2. Una vez que hayas actualizado los archivos en tu repositorio git actual, necesitarás ejecutar cada vez “git add” y “git commit -m___”. Cada vez que hayas actualizado tu repositorio localmente, puedes enviar tus códigos a GitHub (ej: git push origin master).

3. Para el curso detallado de Terminal y Git,

visita: <http://learn.codingdojo.com/m/2/5594/36677>

Comandos Básicos de Terminal para Mac/Linux

Para comenzar, abre tu terminal y prueba los comandos a continuación. Ten en cuenta que la mayoría de estos funcionan bien en Windows command prompt (cmd), y las diferencias existentes se notan.

Present working directory

```
pwd
```

Usa este comando cada vez que no estés seguro donde estás actualmente en tu estructura de archivos. *pwd* significa Present Working Directory o Directorio de Trabajo Actual. En Windows puedes usar *pwd* o *cd*.

List files (lista de archivos)

```
ls
```

Utiliza el comando *ls* para ver todos los directorios y archivos que están en tu directorio actual. Para ser más claros, un directorio es simplemente un sinónimo de carpeta. En 'cmd' utiliza el comando *dir*.

List files (long form) / Lista de archivos (forma extensa)

```
ls -l
```

Utiliza este comando para ver una forma extensa de los directorios y archivos dentro de tu directorio actual.

Verás algunos mensajes incomprensibles a un lado. El 'dwx' es simplemente un permiso de archivo para ese directorio o archivo en particular. Por ejemplo, el primer set de '-'s que sigue a 'd' son los permisos autorizados para el usuario. Si dice 'rwx' significa que el usuario está autorizado para leer, escribir y ejecutar el archivo. Si solo dice 'r--' significa que el usuario solo puede leer el archivo. El segundo set de tres '-'s representa el permiso para un grupo. Los últimos tres '-'s representan el permiso para personas que no son usuarios y no pertenecen al

grupo autorizado. Tendremos que cambiar algunos de estos permisos cuando implementamos nuestras aplicaciones en la nube. Por ahora, es importante saber que hay diferentes permisos de archivo respecto a leer, escribir o ejecutar, y estos permisos pueden variar dependiendo si eres el usuario o del grupo especificado en la lista.

Cambiando al directorio principal

```
cd ..
```

Tal como el comando `'cd .' , '..'` representa al directorio principal de la carpeta en la que estamos. Si ejecutas este comando y luego ejecutas el comando `'pwd'`. verás que navegamos a una estructura de directorio (carpeta). Si sientes que la Estructura de Archivo empieza a asemejarse al DOM (Document Object Model), es porque, de hecho, son muy similares.

Cambiando a cualquier carpeta

```
cd nombre_carpeta
```

El comando `'cd'` seguido de un espacio y el nombre de la carpeta que quieres acceder te llevará a ese directorio, si es que existe. El ejemplo anterior te llevaría al directorio `'nombre_carpeta'`.

Creando una nueva carpeta

```
mkdir nombre_nuevo_directorio
```

Este comando creará un nuevo directorio llamado `'nombre_nuevo_directorio'`. Cualquiera texto que escribas en este comando será el nombre de tu nuevo archivo. Luego de crear el directorio, puedes ejecutar `'cd nombre_nuevo_directorio'` para entrar a ese directorio.

Creando un nuevo archivo

```
touch index.html  
touch style.css
```

El comando `'touch'` creará un archivo vacío con el nombre y extensión que especifiques. Puedes crear todo tipo de archivos si agregas el tipo de extensión. Ejecuta los comandos del ejemplo y luego ejecuta `'ls'` y así verás tus nuevos archivos en la lista.

Comandos Básicos de Git

- `git init` - Inicia un repositorio git local en la carpeta
- `git add .` - Agrega todos los archivos nuevos en preparación (staging)
- `git status`
- `git commit -m "__message__"` - realiza los cambios
- `git remote add origin __github_url__` - agrega un lugar remoto llamado 'origin' que apunta a tu github_url
- `git push` - lleva tus últimos cambios a github
- `git clone __github_url__` - clona un repositorio de github a tu directorio de trabajo local

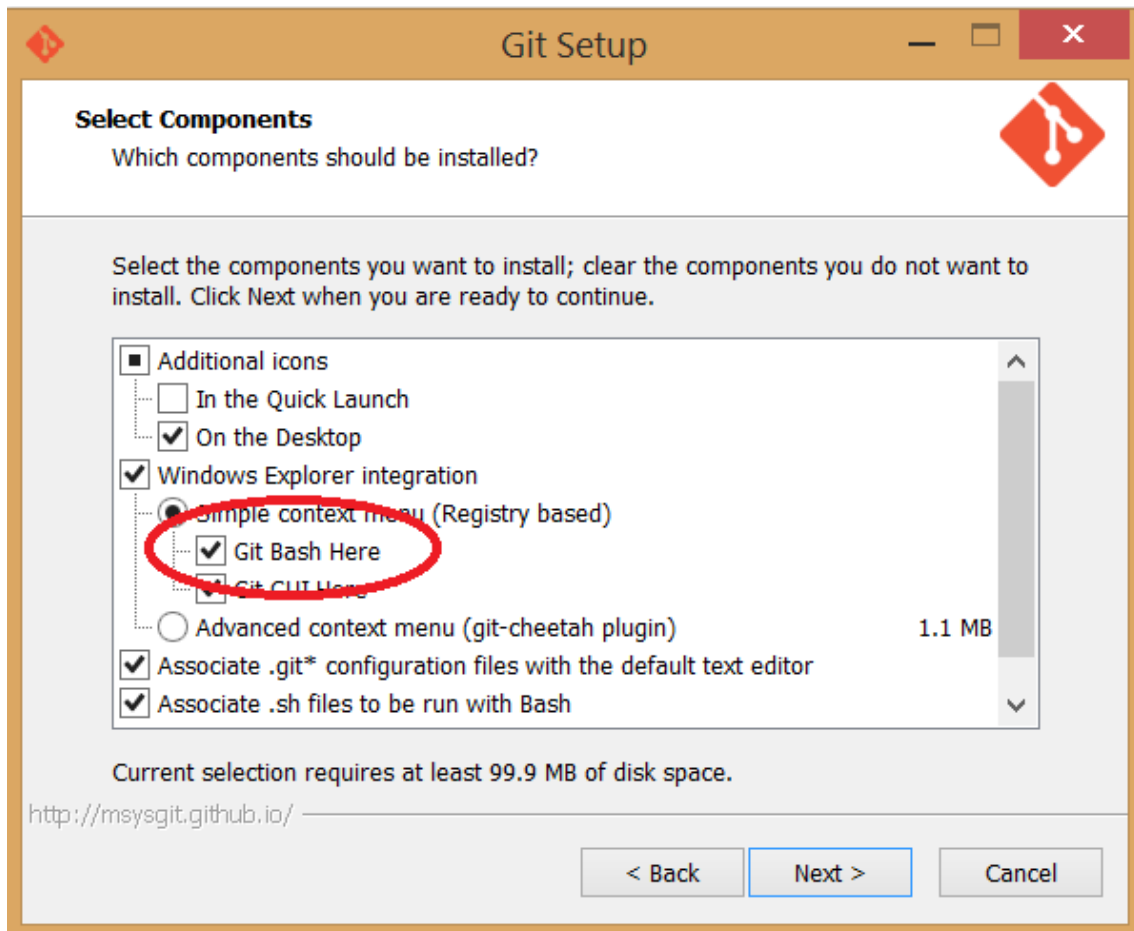
Instalando Git Bash para Windows

(Si eres usuario de Mac, puedes saltarte estos pasos)

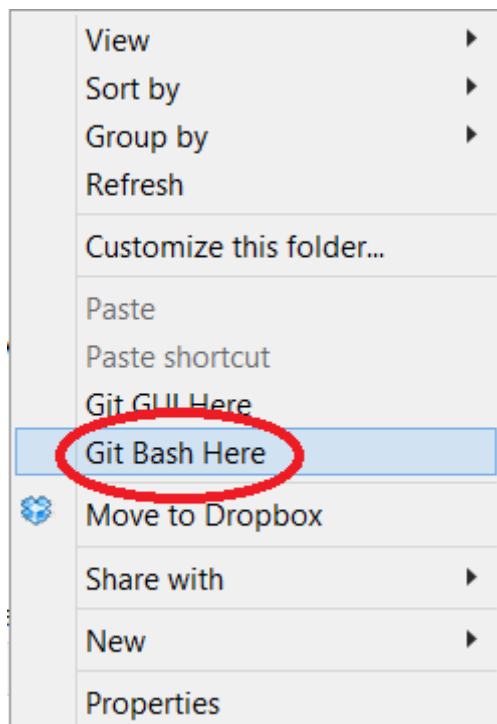
Los usuarios de Windows no solo tienen una, sino que 2 terminales nativos: Command Prompt (AKA CMD) y PowerShell, y ninguno tiene el mismo set de comandos que el terminal bash Mac/Linux. Para evitar confusiones y porque será importante saber cómo usar un terminal bash más adelante en el bootcamp, te recomendamos descargar un emulador bash, así puedes usar los mismos comandos que tus compañeros con Mac o Linux. De todas formas es importante que sepas CMD PowerShell o ambos.

Git Bash

Sigue [este link](#) para instalar Git. Cuando lo instales, asegúrate de marcar la casilla Git Bash Here. Esto no solo instalará tu terminal Git Bash, sino que también lo integrará con Windows Explorer.



Si ves el Git Bash here en tu opciones del botón derecho, quiere decir que lo instalaste bien.



Abre el terminal Bash y confirma que ves lo siguiente:

A screenshot of a MINGW32 terminal window. The title bar shows the path 'MINGW32:/c/Users/B'. The terminal output displays the Git welcome message for version 1.9.5-preview20141217, instructions on how to use 'git help', and the current shell prompt 'B@SHODAN ~ (master) \$'.

¡Genial! ¡Funciona! Ahora puedes ejecutar comandos Unix-style.

Files

Proyecto de Prototipo Clickable

Si terminaste todos los desafíos de de algoritmos y has experimentado con Bootstrap y Foundation, entonces puedes comenzar tu proyecto construyendo tu primer prototipo clickable.

Información General

¿Qué es un prototipo clickable?

Antes de crear cualquier producto, es muy recomendable que el equipo construya primero un esquema de página (wireframe) y luego desarrolle el sitio web en html, css y javascript. Esta primera versión del sitio web sin backend, se llama prototipo clickable. Más adelante, cuando te adentres en Python y Django, aprenderás cómo agregar las funcionalidades backend, pero siempre es importante crear este prototipo clickable primero y luego agregar las funcionalidades backend, en lugar de construir el html y css mientras se crea el backend.

En qué Prototipo Clickable Trabajar

1. Elige trabajar en un sistema de e-commerce, un Panel de Control del Usuario (User Dashboard), un sistema de entrada y salida de los empleados por reloj o un sitio para tu portafolio.
 2. Si tienes alguna otra idea para trabajar, esquematízala en la pizarra o un papel, luego utiliza Balsamiq o Powerpoint para diseñar cómo tu sitio web se vería y pídele al instructor que mire tu esquema de página y te de feedback.
- Sistema de e-commerce: [eCommerce.pdf](#)
 - https://s3.amazonaws.com/General_V88/boomyeah2015/codingdojo/curriculum/content/chapter/eCommerce.pdf
 - EPS (employee clock in/out management system o sistema de gestión de entrada y salida de los empleados por reloj): [epswireframe.pdf](#)
 - https://s3.amazonaws.com/General_V88/boomyeah2015/codingdojo/curriculum/content/chapter/epswireframe.pdf
 - Panel de Control del Usuario: [user-dashboard.pdf](#)

Construyendo un sitio web para tu portafolio

Otro proyecto en el que puedes trabajar es crear tu propio sitio web para tu portafolio, donde te presentes y destagues los futuros proyectos en los que trabajarás. Además, puedes poner tu CV, información de contacto, información adicional de ti, etc. Si haces este sitio, asegúrate de que sea responsivo y que se vea bien. Es muy recomendable que todos los estudiantes tengan su propio sitio de portafolio.

Algunos ejemplos de portafolios de nuestros alumnos

1. <http://mattfarley.ca/>
2. <https://jonny.me/>
3. <http://kyletsuyemura.com/>
4. <http://neholasgomez.com/>

Desafío Adicional

Desarrolla las primeras páginas de tu sitio de manera responsiva. Será una muy buena experiencia para ti e impresionará a quienes vean tu proyecto.

CSS3



CSS3 es una serie de módulos diseñados para implementarse de forma separada e independiente. Pese a que hay partes de CSS3 que no se han implementado en ningún navegador, eso no significa que no podamos usar CSS3, de hecho, podemos utilizar algunas partes desde ya, siempre y cuando sepamos cuándo y cómo agregarlo.

Podemos agrupar los elementos visuales de nuestro sitio web en 2 categorías, crítico y no crítico. Los elementos visuales críticos incluyen la marca (branding), la usabilidad, la accesibilidad y el diseño. Los elementos visuales no críticos incluyen la interacción, las recompensas visuales, el feedback y el movimiento. Tenemos que asegurarnos de aplicar CSS3 a las áreas no críticas porque sería riesgoso utilizar CSS3 en las áreas críticas. CSS3 es la guinda del pastel, por eso revisaremos un pequeño grupo de sus propiedades que han alcanzado suficiente soporte para ser utilizadas.

First Child, Last Child, nth Child

Dado:

```
<body>

  <p>hello</p>

  <div>

    <p>dojo</p>

  </div>

</body>
```

¿Cómo seleccionarías solo el primer párrafo y no el párrafo secundario que está dentro de la división? Por favor revisa: <http://krasimirtsonev.com/blog/article/CSS-Understanding-first-child-second-child-and-nth-child-nth-of-type>.

A algunos alumnos les preguntan esto durante sus entrevistas de trabajo.

Box Shadow

Con la propiedad box-shadow puedes agregar sombra a un elemento con opciones de dirección, difuminación y color de la sombra. Puedes aprender más sobre box-shadow [aquí](#).

```
.awesome {  
  
  -webkit-box-shadow: 10px 10px 5px #333;  
  
  -moz-box-shadow: 10px 10px 5px #333;  
  
  box-shadow: 10px 10px 5px #333;  
  
}
```

Opacity

Con la propiedad opacity puedes definir cuan opaco sea un elemento. El valor 1 significa que el elemento es completamente opaco mientras que 0 significa transparente (invisible). Puedes aprender más sobre opacity [aquí](#).

```
.awesome {  
  
  opacity: 0.5;  
  
}
```

RGBA

RGBA no es una propiedad de CSS, sino que es un nuevo modelo de color introducido en CSS3, que permite especificar el nivel de opacidad con un valor de color RGB. Puedes aprender más de RGBA [aquí](#).

```
.awesome {  
  
  background-color: rgba(0, 0, 0, 0.8);  
  
}
```


Múltiples Imágenes de Fondo

CSS3 te permite implementar múltiples imágenes de fondo en un elemento (separado por comas). Así es como las personas crean el efecto efecto de desplazamiento de paralaje (parallax scrolling effect). Puedes aprender más sobre múltiples imágenes de fondo [aquí](#).

```
body {  
  
    background: url(first_image.png) no-repeat top left,  
  
    url(second_image.png) repeat-x bottom left;  
  
}
```

Prefijos del Navegador

Los Prefijos CSS del Navegador (CSS Browser Prefixes) o prefijos de Proveedor de CSS (CSS Vendor prefixes) son una forma en que los desarrolladores de un navegador agregan soporte para las nuevas características de CSS antes que estas sean totalmente compatibles en todos los navegadores. Algunos ejemplos de estos prefijos son '-webkit-' y '-moz'. Deberías revisar [esto](#) para más información sobre prefijos.

Otros Recursos

También puedes hacer transiciones y transformar tus elementos con CSS3, lo que alguna vez solo fue posible con JavaScript. Puedes leer más sobre transiciones [aquí](#) y sobre transformaciones [acá](#).

Recursos Adicionales

Lorem Ipsums

Tal vez has escuchado sobre Lorem Ipsum, el texto de relleno preferido por muchos diseñadores. Aquí tienes algunos Ipsums no tradicionales para alegrarte el día ¡Siéntete libre de elegir y usar el que te guste!

- Real gibberish - <http://randomtextgenerator.com/>
- Online dating Ipsum - <http://laurenhallden.com/datingipsum/>
- A delicious Ipsum - <http://cupcakeipsum.com/>

- Arrested Development Ipsum - <http://bluthipsum.com/>
- Futurama Ipsum - <http://chrisvalleskey.com/fillerama/>
- Hipster Ipsum - <http://hipsteripsum.co/>
- Bacon Ipsum - <http://baconipsum.com/>
- Samuel L. Ipsum - <http://slipsum.com/> (NSFW)
- Gangsta Ipsum - <http://lorizzle.nl/>

Imágenes y Fondos

- Imágenes de tipo “placeholder” para todas tus necesidades: <http://lorempixel.com/>
- Porque internet necesita más gatos: <http://placekitten.com/>
- Toolkit de íconos para CSS: <https://fortawesome.github.io/Font-Awesome/>
- Imágenes hi-res gratuitas: <http://unsplash.com/>
- Fondos: <http://subtlepatterns.com/>
- Varios recursos gratuitos incluyendo
íconos: <http://www.webdesignerdepot.com/category/freebies/>
- Hablando de íconos, puedes usar este sitio y generar CSS para hojas de
sprite: <http://www.spritecow.com/>

Color Schemes

- Muy flexible, muy útil: <http://colorschemedesigner.com/>
- ¿Has pensado alguna vez “¡Oh Dios, amo los colores de esta foto!?” Entonces
revisa: <http://www.degraeve.com/color-palette/index.php>
- Similar a lo anterior: <http://www.cssdrive.com/imagepalette/>

Otras Herramientas

- Recortar - Permite a los diseñadores medir los píxeles y recortar imágenes fácilmente.
- Selector de color - Selector y editor de color
gratuito: <http://annystudio.com/software/colorpicker/#jcp-download>

Nuevas Tendencias

Toma esto con cautela ¡Recuerda que las tendencias van y vienen, pero un buen diseño es para siempre!

<https://webflow.com/blog/20-web-design-trends-for-2019>

Esta ingeniosa herramienta tomará tus archivos de texto y agregará un markdown de HTML
¡Úsalo con precaución y siempre revisa dos veces el output!

<http://daringfireball.net/projects/markdown/>

Tabla Periódica

Esta es una actividad opcional. Trabaja en ella solo si ya terminaste todas las actividades previas de CSS. Crea una tabla periódica que se vea similar a la de la imagen a continuación (descarga la imagen [aquí](#)).

¡No uses tablas!

Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18																
Period	1	2																2																
1	1 H 1.008																	2 He 4.0026																
2	3 Li 6.94	4 Be 9.0122											5 B 10.81	6 C 12.011	7 N 14.007	8 O 15.999	9 F 18.998	10 Ne 20.180																
3	11 Na 22.990	12 Mg 24.305											13 Al 26.982	14 Si 28.085	15 P 30.974	16 S 32.06	17 Cl 35.45	18 Ar 39.948																
4	19 K 39.098	20 Ca 40.078	21 Sc 44.956	22 Ti 47.887	23 V 50.942	24 Cr 51.996	25 Mn 54.938	26 Fe 55.845	27 Co 58.933	28 Ni 58.693	29 Cu 63.546	30 Zn 65.38	31 Ga 69.723	32 Ge 72.63	33 As 74.922	34 Se 78.96	35 Br 79.904	36 Kr 83.798																
5	37 Rb 85.468	38 Sr 87.62	39 Y 88.906	40 Zr 91.224	41 Nb 92.906	42 Mo 95.96	43 Tc [97.91]	44 Ru 101.07	45 Rh 102.91	46 Pd 106.42	47 Ag 107.87	48 Cd 112.41	49 In 114.82	50 Sn 118.71	51 Sb 121.76	52 Te 127.60	53 I 126.90	54 Xe 131.29																
6	55 Cs 132.91	56 Ba 137.33	57 La 138.91	58 Ce 140.12	59 Pr 140.91	60 Nd 144.24	61 Pm [144.91]	62 Sm 150.36	63 Eu 151.96	64 Gd 157.25	65 Tb 158.93	66 Dy 162.50	67 Ho 164.93	68 Er 167.26	69 Tm 168.93	70 Yb 173.05	71 Lu 174.97	72 Hf 178.49	73 Ta 180.95	74 W 183.84	75 Re 186.21	76 Os 190.23	77 Ir 192.22	78 Pt 195.08	79 Au 196.97	80 Hg 200.59	81 Tl 204.38	82 Pb 207.2	83 Bi 208.98	84 Po [209]	85 At [209.98]	86 Rn [222.02]		
7	87 Fr [223.02]	88 Ra [226.025]	89 Ac [227.03]	90 Th 232.04	91 Pa 231.04	92 U 238.03	93 Np [237.05]	94 Pu [244.06]	95 Am [243.06]	96 Cm [247.07]	97 Bk [247.07]	98 Cf [251.08]	99 Es [252.08]	100 Fm [257.10]	101 Md [258.10]	102 No [259.10]	103 Lr [260.10]	104 Rf [261.10]	105 Db [262.11]	106 Sg [266.12]	107 Bh [268.13]	108 Hs [271.13]	109 Mt [273.15]	110 Ds [281.16]	111 Rg [284.16]	112 Cn [285.17]	113 Nh [284.18]	114 Fl [289.19]	115 Mc [289.19]	116 Lv [293]	117 Ts [293]	118 Og [294]		
*Lanthanoids			*	57 La 138.91	58 Ce 140.12	59 Pr 140.91	60 Nd 144.24	61 Pm [144.91]	62 Sm 150.36	63 Eu 151.96	64 Gd 157.25	65 Tb 158.93	66 Dy 162.50	67 Ho 164.93	68 Er 167.26	69 Tm 168.93	70 Yb 173.05																	
**Actinoids			**	89 Ac [227.03]	90 Th 232.04	91 Pa 231.04	92 U 238.03	93 Np [237.05]	94 Pu [244.06]	95 Am [243.06]	96 Cm [247.07]	97 Bk [247.07]	98 Cf [251.08]	99 Es [252.08]	100 Fm [257.10]	101 Md [258.10]	102 No [259.10]																	

Asegúrate de no utilizar divisiones innecesarias. Además, valida tu código antes de que tu mentor/instructor lo revise ¡Algunos de nuestros estudiantes han completado esta actividad con menos de 150 líneas de código! Intenta hacer algo similar sin, por supuesto, poner demasiadas cosas en la misma línea de código.

Por favor, no utilices más de 4 horas en esta actividad.

•

Por favor, no utilices más de 4 horas en esta actividad.

Bootstrap

Objetivos

1. Familiarizarte con Bootstrap, un marco/librería CSS muy popular utilizado por muchos desarrolladores.
2. Como desarrolladores, lo más probable es que utilicemos un archivo CSS creado por otros. Esta es una buena oportunidad para que sepas cómo aprovechar un marco CSS creado por otros.
3. Apreciar el diseño simple que ofrece Bootstrap.

Introducción

¿Qué es Bootstrap?

En palabras de sus creadores, Bootstrap es "un elegante, intuitivo y potente marco de interfaz de usuario para un desarrollo web más fácil y rápido".

Para utilizar Bootstrap, primero colocamos las siguientes líneas en la sección <head> de nuestro archivo:

```
<!-- Etiquetas meta requeridas -->

<meta charset="utf-8">

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

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

<!-- Etiquetas meta de JavaScript Opcionales-->

<!-- Primero jQuery, luego Popper.js, y por último Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfxDz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKA1Zap3H66lZ81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvf08shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script>
```

Ahora puedes continuar y **agregar clases Bootstrap a tus elementos HTML** para obtener fácilmente un diseño consistente, limpio y responsivo.

Diseño de Páginas

Usando Bootstrap para los Diseños

Para empezar, veremos el diseño general utilizando contenedores, columnas y un sistema de cuadrícula. Bootstrap utiliza *flexbox* para desarrollar un sistema de cuadrícula donde un contenedor es dividido hasta en 12 columnas. Se requiere de una clase padre `.container` para poder utilizar las columnas.

Veamos el siguiente código:

```
<div class="container">

  <div class="row">

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

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

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

  </div>

</div>
```

Esto producirá 3 columnas iguales en la página ($4+4+4 = 12$). Si agregamos más clases Bootstrap al código, podríamos ver un resultado como el siguiente:

```
<div class="container text-center">

  <div class="row">

    <div class="col bg-secondary text-light m-2">One</div>

    <div class="col bg-secondary text-light m-2">Two</div>

    <div class="col bg-secondary text-light m-2">Three</div>

  </div>

</div>
```

One

Two

Three

Las columnas también se pueden anidar dentro de otras columnas. En este ejemplo vamos a anidar 3 columnas iguales dentro de su columna principal.

```
<div class="container text-center">

  <div class="row">

    <div class="col bg-secondary text-light m-2">One</div>

    <div class="col bg-secondary text-light m-2">Two

      <div class="row">

        <div class="col bg-info text-light m-2">A</div>

        <div class="col bg-info text-light m-2">B</div>

        <div class="col bg-info text-light m-2">C</div>

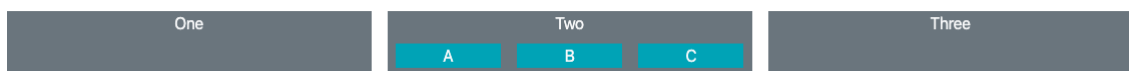
      </div>

    </div>

    <div class="col bg-secondary text-light m-2">Three</div>

  </div>

</div>
```



Incluso podemos establecer el ancho de la columna y compensarlo en el diseño. El siguiente ejemplo centrará la columna en la página y tendrá por un ancho predeterminado de 6 de las columnas disponibles.

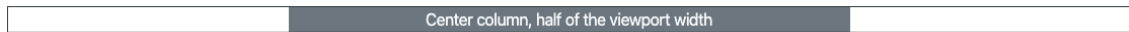
```
<div class="container border border-dark">

  <div class="row">

    <div class="col-6 offset-3 bg-secondary text-light">Center column,
    half of the viewport width</div>

  </div>

</div>
```



Clases muy útiles para los diseños

Contenedores - <https://getbootstrap.com/docs/4.4/layout/overview/> - (solo aprenda sobre la clase *container* y omita el resto)

Sistema de Cuadrícula - <https://getbootstrap.com/docs/4.4/layout/grid/> - (para posicionar y dimensionar las columnas):

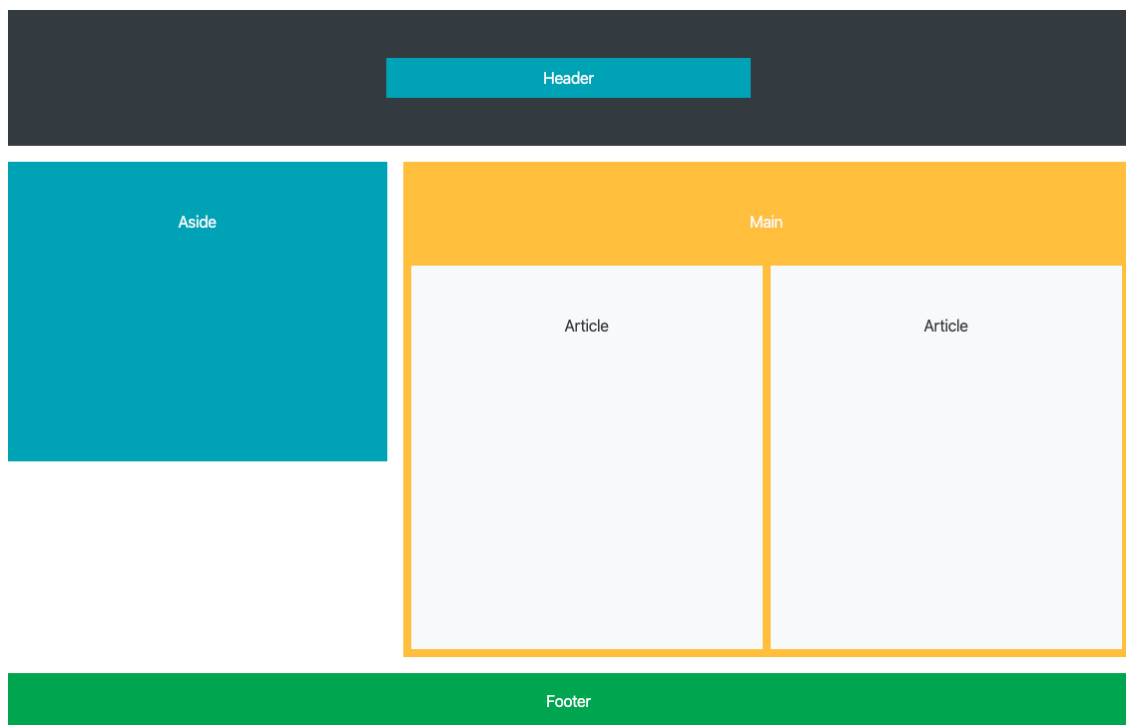
Esquema de Color - <https://getbootstrap.com/docs/4.4/utilities/colors/> - (útil para ver como están quedando las columnas)

Espaciados - <https://getbootstrap.com/docs/4.4/utilities/spacing/> - (margenes y espaciados)

Dimensiones - <https://getbootstrap.com/docs/4.4/utilities/sizing/> - (alto y ancho de los elementos)

Trazando Bloques con Bootstrap

Utilice solo clases Bootstrap para replicar el siguiente diseño.



```
<div>

  <div>

    <div>
```

```
        <div>Header</div>

    </div>

</div>

<div>

    <div style="height:300px;">Aside</div>

    <div>Main

        <div style="height:400px;">

            <div>Article</div>

            <div>Article</div>

        </div>

    </div>

</div>

<div>

    <div>Footer</div>

</div>

</div>
```

Utilice las siguientes clases para esta actividad

- .col-{numero, blank, auto}
- .row
- .bg-{color}
- .p-{numero} (el espaciado (*padding*s) puede ser aplicado en diferentes formas. Más detalles [aquí](#)).
- .m-{number} (lo mismo que el atributo anterior)
- .text-{color}
- .container
- .offset-{numero}