

Curso html y css platzi

Que es el Frontend?

Cliente

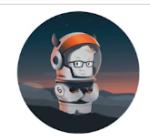


Es el que va a manejar las cosas del lado del cliente.

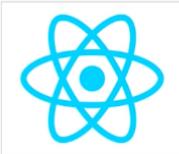
Estándares



Frameworks de CSS



Frameworks y librerías de JS



Preprocesadores de CSS



Compilador / Empaquetador de JS



Herramientas que maneja un frontend

Debido a que un frontend es el desarrollador que va a manejar las cosas del lado del cliente, las tecnologías con las que va a trabajar son:

- HTML: <https://devdocs.io/html/>
- CSS: <https://devdocs.io/css/>
- JavaScript: <https://devdocs.io/javascript/>
- Frameworks de CSS para frontend:
- Bootstrap: <https://getbootstrap.com/>
- Foundation CSS: <https://get.foundation/>
- Materialize CSS: <https://materializecss.com/>
- Los frameworks de JavaScript para frontend:
- React JS: <https://es.reactjs.org/>
- Angular JS: <https://angular.io/>
- Vue JS: <https://vuejs.org/>
- Preprocesadores de CSS:
- Stylus: <https://stylus-lang.com/>
- SASS: <https://sass-lang.com/>
- Compiladores / empaquetadores de JS:
- BABEL: <https://babeljs.io/>
- Webpack: <https://webpack.js.org/>
- Aporte creado por: Christian Tambo, Manuel Duarte.

Que es Backend?

Backend en programación corresponde al lado opuesto a un Front-end en un sitio web o aplicación, ya que el Backend trabaja en el lado del servidor, mientras el Frontend lo hace en el lado del cliente. Es el responsable de manejar toda la lógica que hay detrás de una petición dada por el navegador hacia el servidor.

Una característica que lo diferencia del Frontend es que no tiene estándares, puesto que tiene varios lenguajes de programación (Node.js, Python, PHP, Ruby, GO, Java, .NET entre otros) con los que debe trabajar. Cada uno de estos lenguajes tiene sus propios frameworks como:

- Django (Python)
- Lavarel (PHP)
- Rails (Ruby)
- Express (JavaScript)
- Spring (Java)

El Backend también tiene en cuenta la infraestructura donde va a realizarse el deploy de su aplicación (esto también puede ser tarea de un DevOps, un perfil dedicado a la infraestructura), con tecnologías como:

- Google Cloud
- DigitalOcean
- AWS
- Heroku, entre otras.

¿Qué es deploy?

Deploy es un término famoso entre los desarrolladores web. Puede significar muchas cosas, dependiendo del ambiente y de la tecnología usada. Sin embargo, los significados que más se refieren a la práctica y pueden resumir su función son: implantar, colocar en posición, habilitar para uso o, simplemente, publicar.

Por último, entramos en bases de datos, que son las encargadas de almacenar toda la información del proyecto. Los principales tipos son:

Bases de datos relacionales (como MySQL)

Bases de datos no relacionales (como mongoDB).

Aporte creado por: Matías Wasiak, Pedro Moreno.

¿Qué es FullStack?

Fullstack es un término utilizado para describir a los desarrolladores que conocen tanto los lenguajes de frontend como de backend. Principalmente, el desarrollo full stack se refiere al uso de JavaScript en el backend y de HTML/CSS/JavaScript en el frontend.

El nacimiento de tecnologías que funcionan entre el front-end y el back-end ha dado lugar a la proliferación de frameworks y herramientas de desarrollo “full stack”, que permiten a los desarrolladores construir sus propias aplicaciones web completas utilizando un solo lenguaje de programación, como Ruby on Rails o Django para Python.

Qué es un desarrollador fullstack

Los desarrolladores fullstack son profesionales que se encargan tanto de la parte visual y de interacción de un sitio (frontend), como de su lógica y funcionamiento del lado del servidor (backend). Un stack (en inglés: pila o montón) hace referencia al grupo de tecnologías que componen un sitio web en todos sus aspectos (desde la base de datos, hacia el manejo lógico y la interfaz visual). Un desarrollador fullstack en teoría es capaz de manejar la pila completa de un sitio y por ende entiende de tecnologías tanto de frontend como de backend, además del manejo de su base de datos.

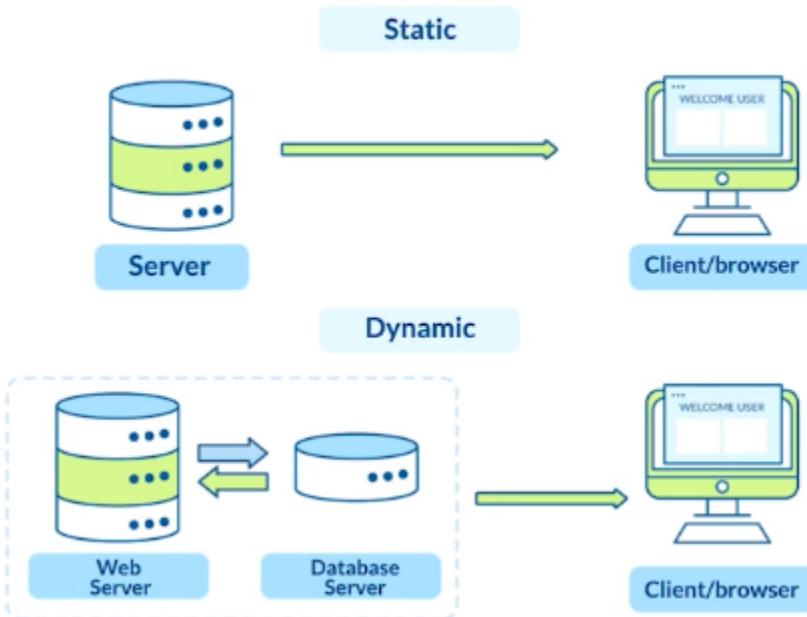
Ventajas de un desarrollador o desarrolladora Fullstack

El beneficio para el proyecto de alguien fullstack es que entiende muy bien cómo funciona un producto web de principio a fin, desde su diseño en mockup y deploy hasta producción.

Desventajas de ser desarrollador Fullstack

Un fullstack no maneja por completo todas las tecnologías de ambas partes, pues cada una requiere conocimiento profundo. De hecho, no es recomendado y no es sano. El desarrollo web evoluciona muy rápido y cada dos o tres meses tenemos algo nuevo, tanto en backend como en frontend.

Páginas Estáticas vs. Dinámicas



Los sitios web se comportan de forma diferente dependiendo de la forma en que fueron diseñados desde su concepción, tomando en cuenta la interacción con el usuario. Aquí veremos las diferencias entre sitios web estáticos y dinámicos:

Sitios Web Estáticos (Landing pages)

La información que contiene se mantiene constante y estática. No se actualiza con la interacción del usuario. Es conveniente para realizar landing pages (páginas informativas o de aterrizaje) o blogs. Se mostrarán siempre iguales para todos los usuarios.

Sitios Web Dinámicos (Web Apps)

También conocidos como aplicaciones web, actualizan su información con respecto a la interacción del usuario. Dependen de una base de datos, de donde extrae e ingresa información. Serán diferentes, dependiendo del usuario que la use y la información que se ingrese.

Ejemplo de páginas estáticas:

- Menú de un restaurante
- Blog de viajes

- Página informativa de un negocio

Ejemplo páginas dinámicas:

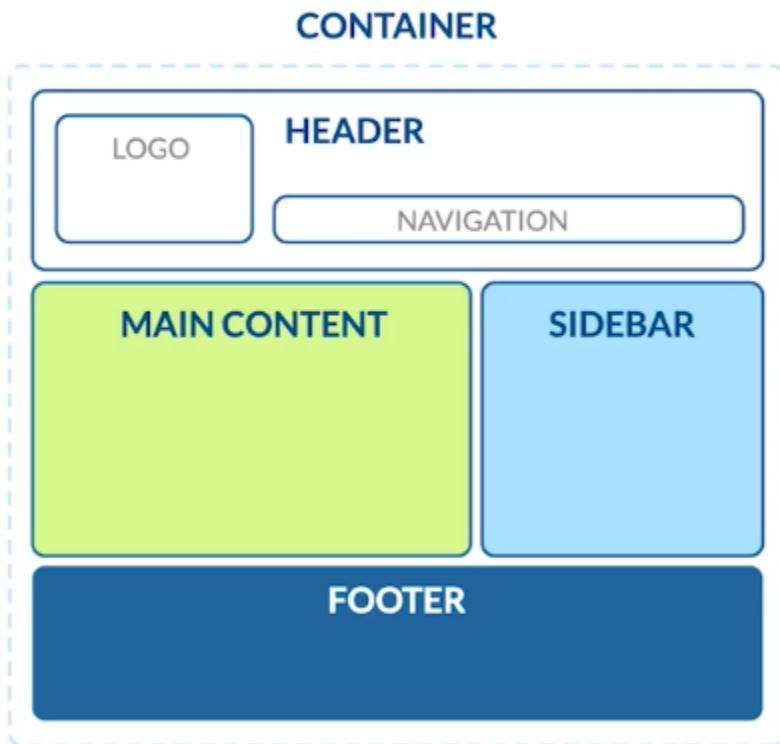
- Sistema de reporte de ventas
- Linkedin
- Banca en línea

Anatomía de una Página Web

HTML (HyperText Markup Language) es un lenguaje de marcado de texto. Se utiliza para darle una estructura al sitio web que estás visitando.

Estructura básica de HTML en una página Web

- Container: contenedor principal
- Header: cabecera de la página. Aquí usualmente encuentras el logo y el menú de navegación del sitio.
- Main content: estructura principal. Por ejemplo, el feed o lista de publicaciones de una red social.
- Sidebar: contenido secundario de una página, que usualmente se encuentra a los lados del contenido principal (o main).
- Footer: pie de página. Esto se encuentra al fondo del sitio web, salvo en casos de sitios web donde el scroll (o navegación hacia abajo) es infinito, por ende, no tendría sentido ponerlo al fondo.



Las etiquetas en HTML nos ayudan a diferenciar en qué parte del contenido nos encontramos.

La web se conforma de tres conceptos:

- URL: Uniform Resource Locator. El identificador único del sitio en el navegador (por ejemplo: <https://platzi.com>).
- HTTP: Protocolo de transferencia de hipertexto. Es el estándar que se utiliza para enviar datos a través de paquetes entre el cliente y el servidor.
- HTML: es el código que se emplea para estructurar el contenido de tu web, y darle sentido y propósito.
- HTML son siglas que corresponden a Hyper Text Markup Language (Lenguaje de Marcado de Hipertexto).

Hyper Text significa que el texto tiene interactividad, conexión con otros documentos.

- Markup significa que le pone etiquetas a los elementos. Por eso también se le conoce como un lenguaje de etiquetas.
- HTML es un lenguaje interpretado. Además, HTML es un estándar, así que no importa desde qué navegador o dispositivo se ejecute, el código sigue siendo el mismo en cualquier sitio.

Etiquetas del cuerpo del documento (body):

- article: diferencia partes del contenido que pueden vivir por sí mismas.
- nav: para hacer menús de navegación.
- aside: contenido menos relevante, como publicidad, etc.
- section: sirve para diferenciar las secciones principales del contenido.
- header: cabecera del documento.
- footer: pie de página del documento.
- h1 - h6: títulos de nuestro sitio web.
- table: tablas de contenidos, similar a la estructura de las hojas de cálculo.
- ul y ol: listas de ítems.
- div: cualquier división para organizar el contenido.
- h1 a h6: son etiquetas para indicar títulos con un estilo que destaca del resto.
- article: es la parte de nuestro contenido que puede vivir por sí mismo. Pueden haber tantos artículo como proyectos o eventos tenga nuestro portafolio.
- p: define el texto de un párrafo.
- small: aplica una apariencia de texto reducido en tamaño.
- strong: aplica al texto un formato de negritas.
- a: corresponde a un ancla o enlace a una url interna o externa del documento.
- img: con esta etiqueta podemos enlazar imágenes en el documento.
- figure: le da un contexto semántico a las imágenes.

Anatomía de una etiqueta de HTML

Una etiqueta HTML puede tener tantos atributos como deseas, y cada atributo tiene su propia función. En el siguiente ejemplo, veremos la forma en la que se compone una etiqueta HTML:



- No todas las etiquetas llevan una etiqueta de cierre. Las que llevan un cierre son aquellas que albergan un contenido que nos dice a dónde nos va a llevar (nombre de la página, nombre del link).
- Lo que va dentro de la etiqueta de apertura es un atributo (nombre del atributo = href y el valor del atributo es la url).
- El contenido + la etiqueta = Elemento

Tipos de Imágenes

Tipos de imágenes para web

Lossless (sin pérdida):

Capturan todos los datos del archivo original.

No se pierde nada del archivo original.

Puede comprimirse, pero podrá reconstruir su imagen al estado original

Lossy (con pérdida):

Se aproximan a su imagen original.

Podría reducir la cantidad de colores en su imagen o analizar la imagen en busca de datos innecesarios.

Por consiguiente puede reducir su tamaño, lo que mejora el tiempo de carga de la página, pero pierde su calidad.

Los archivos tipo lossy son mucho más livianos que los archivos tipo lossless, por lo que son ideales para usar en sitios en donde el tamaño del archivo y la velocidad de descarga son importantes.

Tabla de diferencias

	Categoría	Paleta	Uso
	Lossless	Máximo 256 colores	<ul style="list-style-type: none">• Animaciones simples• Gráficos con colores planos
	Lossless	Máximo 256 colores	<ul style="list-style-type: none">• Uso de transparencia• Sin animación• Ideal para íconos
	Lossless	Colores ilimitados	<ul style="list-style-type: none">• Similar a PNG-8• Maneja imágenes fijas de más colores y transparencia
	Lossy	Millones de colores	<ul style="list-style-type: none">• Imágenes fijas• Fotografía
	Vector / Lossless	Colores ilimitados	<ul style="list-style-type: none">• Gráficos / logotipos para web• Retina / pantallas de alta resolución



Formatos de imagen para web

GIF (Graphics Interchange Format): Formato de imagen sin pérdida, no se puede comprimir

PNG 8 (Portable Network Graphics): Formato de imagen sin pérdida, uso de colores de 256, se utiliza para logotipos e iconos para la página.

PNG 24 (Portable Network Graphics): Formato de imagen sin pérdida, utilización de colores ilimitados, alta calidad, si intentamos comprimir no ayudará demasiado por la gran cantidad de colores.

JPG / JPEG (Photographic Experts Group): Formato de imagen con pérdida, perdemos calidad a la hora de comprimir las, pero llegan a ser óptimas para la carga en la página web.

SVG - Vector (Scalable Vector Graphics): Formato de imagen muy ligero sin pérdida, con svg no perdemos calidad, ya que está compuesta por vectores.

WebP: Es un formato gráfico en forma de contenedor que sustenta tanto compresión con pérdida como sin ella. Fue desarrollado por Google.

Optimizando imágenes

Recomendación de principiante usar imágenes de 70kb

Opciones para trabajar las imágenes

- Mejora el tamaño de tus imágenes
 - Tiny PNG
- Retira metadatos de tus imágenes
 - Verexif

Formularios

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formularios</title>
</head>
<body>
  <form action="">
    <label for="nombre">
      <span>Cual es tu nombre?</span>
      <input type="text" id="nombre" placeholder="Tu nombre" required>
    </label>
    <label for="inicio-platzi">
      <span>Que dia comenzaste en Platzi?</span>
      <input type="date" id="inicio-platzi" >
    </label>
    <label for="horario">
      <span>En que horario estudias?</span>
      <input type="time" id="horario" >
    </label>
```

```
    </form>
</body>
</html>
```

Notas: el id de label tiene que ser el mismo del input.

Placeholder: es para poner un texto de ayuda.

Autocomplete y require

Autocomplete, completa con información que el usuario ya nos proporcionó. Es el atributo autocomplete en el input.

```
<input type="text" name="nombre" id="nombre" autocomplete="name"/>
```

Require, le avisa al usuario que tiene que completar todos los campos. Usamos require en el input.

```
<input type="text" name="nombre" id="nombre" autocomplete="name" required/>
```

Select

Etiqueta <select>: Esta permite crear la lista, con las etiquetas <option>:

```
<select name="cursos" id="">
  <option value="JavaScript">JavaScript</option>
  <option value="HTML5">HTML5</option>
  <option value="CSS3">CSS3</option>
  <option value="Web Standards">Web Standards</option>
</select>
```

Etiqueta <input list = "cursos">: (con filtro se podría decir, queda mas practico para el usuario) De este modo, se puede utilizar una etiqueta <datalist> con etiquetas <option> dentro del input. De este modo, el usuario puede escribir dentro del input, y filtrar los resultados de la lista:

```
<input list="cursos" />
<datalist id="cursos">
  <option value="JavaScript"></option>
  <option value="HTML5"></option>
  <option value="CSS3"></option>
  <option value="Web Standards"></option>
</datalist>
```

Input type submit vs. Button tag

input type submit = lo utilizaremos solo en los formularios

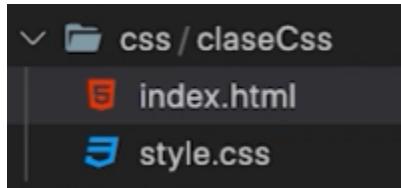
Button = lo utilizaremos en cualquier otro tipo de botón dentro de nuestro proyecto

```
<!-- // solo para formularios -->
<input type="submit" value="Nombre" /> //
con el atributo value podremos cambiar el
texto que se vera en el

<!-- // lo usaremos para cualquier caso
en el que necesitemos un boton -->
<button type="submit">Qué color te gusta?
</button>
<!-- // si quieres usar un button en un
formulario debes agregarle el type =
"submit" -->
```

¿Qué es CSS?

Dividimos css y html:



En el html lo link:

```
<link rel="stylesheet" href=".style.css">
```

Toma los estilos del archivo, que son :

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

```
css > claseCss > style.css > p
```

```
1  p {  
2      color: blue;  
3      font-size: 30px;  
4  }  
5
```

Podemos definir estilos por elementos o por clases.

Por clase sería:

CSS

HTML

```
/* class */  
.parrafo {  
    color: red;  
}
```

```
<p class="parrafo">Soy un texto</p>
```

Por ID:

CSS

HTML

```
/* ID */  
#texto {  
    color: yellow;  
    font-size: 24px;  
}
```

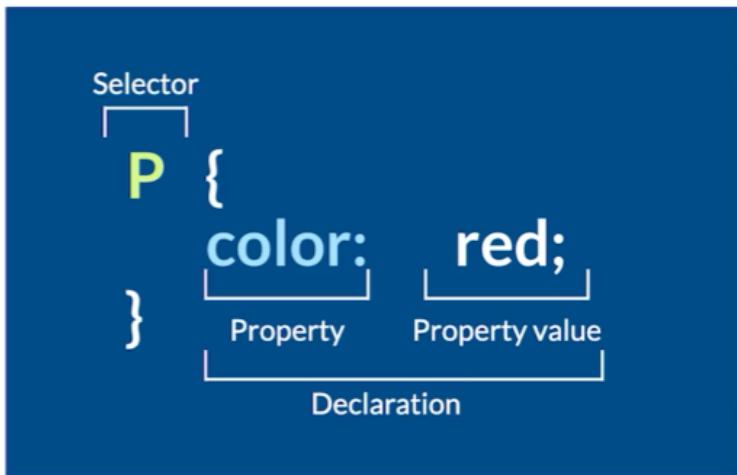
```
<p id="texto">Soy otro texto</p>
```

Pseudo clases y pseudo elementos

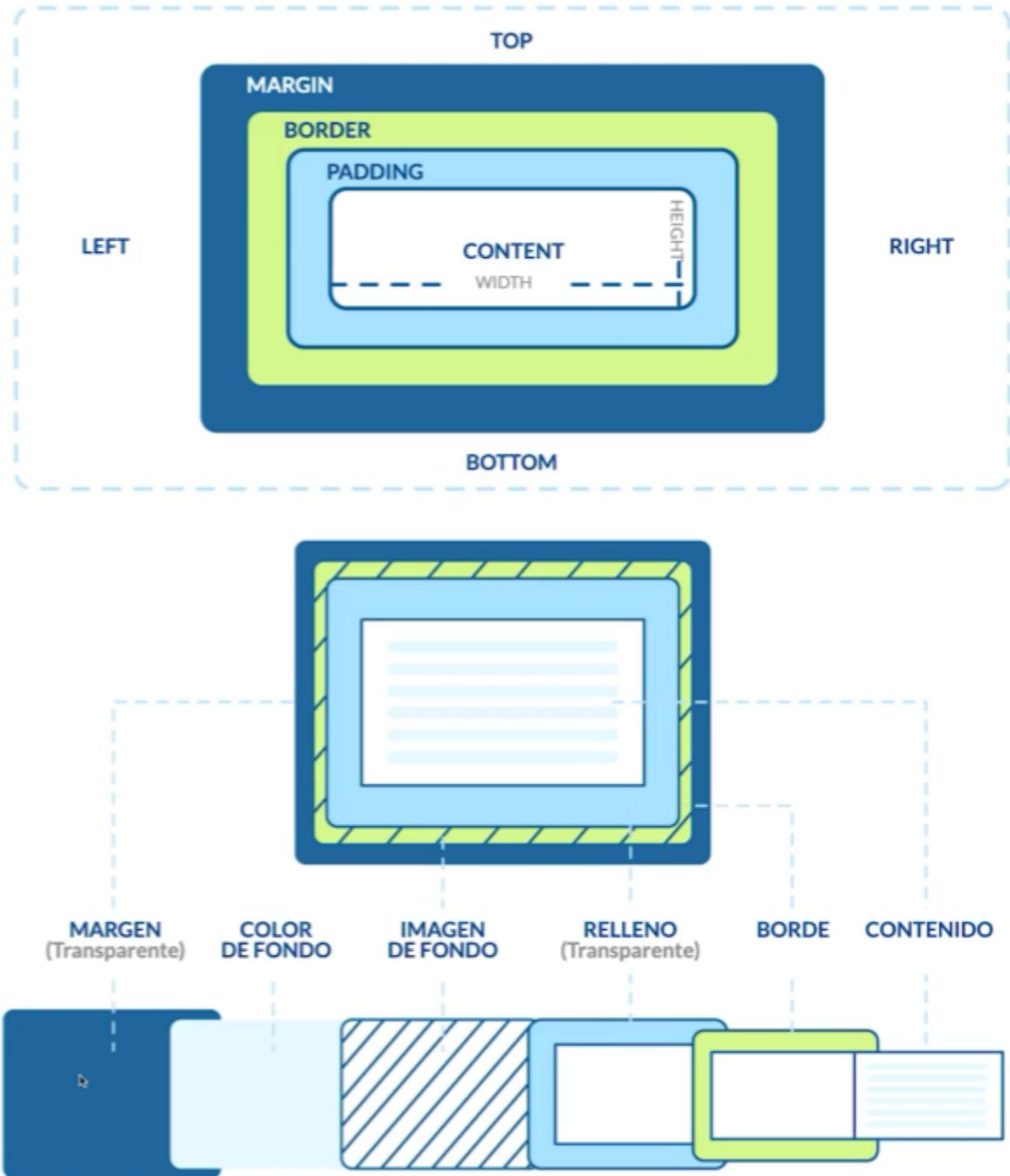
Metodología de cómo normar nuestras clases correctamente → BEM

```
<!-- BLOQUE -->
<main class="Padre">
    <!-- BLOQUE__ELEMENTO -->
    <section class="Padre__Hijo">
        <!-- BLOQUE__ELEMENTO--MODIFICADOR --
    >
        <article class="Padre__Hijo--Mayor">
    </article>
    </section>
    <section class="Padre__Hijo"></section>
    <section class="Padre__Hijo"></section>
    <section class="Padre__Hijo"></section>
</main>
```

Anatomía de una regla de CSS



Modelo de caja

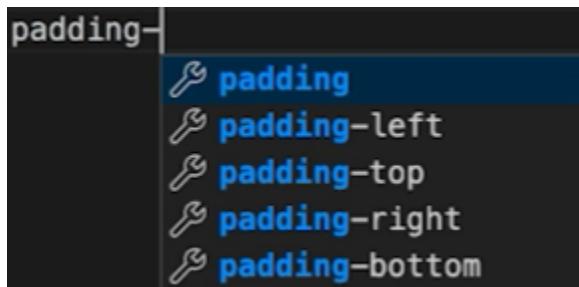


Ejemplo:

```
main {
    width: 100%;
    height: 500px;
    border: 10px solid grey;
    padding: 20px 35px;
}
```

Notas padding:

- si solo le podemos el 20px el navegador van a entender que tiene que ponerle 20px arriba, abajo a la izquierda y a la derecha.
- Así como lo muestra en la imagen quiere decir, el primer dígito se lo asigna a arriba y abajo y el 35 a la izquierda y la derecha.
- Si queremos ser más específicos podemos hacer:



Border sin scroll:

```
* {
    box-sizing: border-box;
    padding: 0;
    margin: 0;
}
```

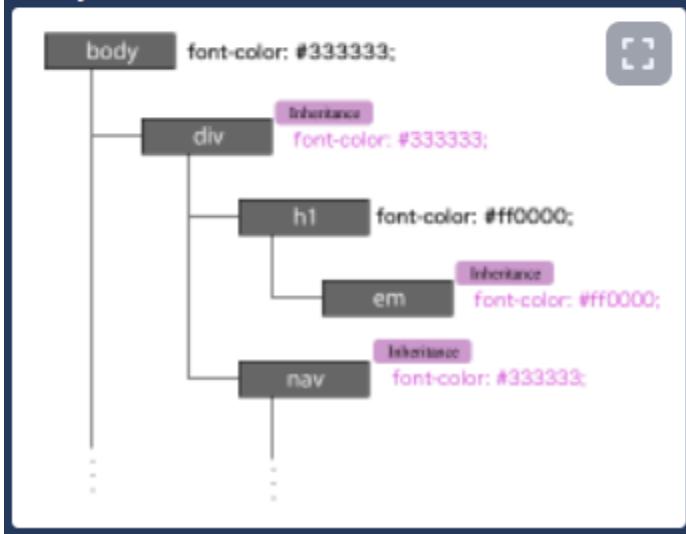
Herencia

Herencia y sus valores:

Inherit. Este es un valor por medio de una *keyword* que especifica que, a la propiedad que se la apliquemos debe de heredar los valores de su elemento padre. Podemos decir que la palabra **Inherit** significa "*Usa el valor de mi padre*", si el elemento padre no tiene definido dicho valor el navegador seguirá el DOM hasta que encuentre un elemento superior que lo contenga, y en ultima instancia de no tenerlo ningún elemento superior se aplicara el valor por defecto.

Initial. Este valor pertenece a la especificación CSS3 y cuando aplicamos a una propiedad el valor *initial* estamos dando el valor inicial y predefinido por el navegador en cuestión.

Upset. Este valor *unset* es una combinación entre *inherit* y *initial*, cuando utilizamos este valor en una propiedad esta tratará de heredar el valor de su elemento padre si este está disponible, de no ser así este valor colocará el valor de la propiedad en su valor inicial, como si usáramos *inherit* e *initial* juntos.



Especificidad en selectores

Temas importantes para usar correctamente las herencias

1. Importancia
2. Especificidad
3. Orden en las fuentes

La importancia es uno de los conceptos más... pues... importantes (haha, ya sé!). Si dos declaraciones tienen la misma importancia, la especificidad de las reglas decidirá cuál se debe aplicar. Si las reglas tienen la misma especificidad, el orden de las fuentes controla el resultado final.

Importancia

1. Hoja de estilo de agente de usuario (Estilos del navegador)
2. Declaraciones normales en hojas de estilo de autor (Nuestro .css)
3. Declaraciones importantes en hojas de estilos de autor (utilizar el !important)

Especificidad

Selectores	Especificidad
!important	1,0,0,0
Inline styles	0,1,0,0
#id	0,0,1,0
.class	0,0,0,1
tag	0,0,0,0

!Importan no es recomendable usarlo porque es mala práctica.

A: h1
B: #content h1
C: <div id="content"><h1 style="color : #fffff"> Heading </h1></div>

- ▲ La especificidad de A es 1 (es una etiqueta)
- ▲ La especificidad de B es 101 (un #(id) y una etiqueta)
- ▲ La especificidad de C es 1000 (es un estilo en línea)

Mayor nivel de especificidad
por lo tanto se aplica Platzi

24

Reglas de cascada



Orden de las fuentes

En tus estilos, las declaraciones al final del documento anularán a las que sucedan antes en caso de conflicto.

NOTA: Si tenemos .class #id se aplicarán los estilos del ID porque el navegador los toma como más importantes. #id es más importante que: class y tag.

Tratar de no usar id en lo posible, puede romper la cascada.

Para css es más importante tener dos clases que una clase y una etiqueta.

EJ:

```
.nav a {  
    color: white;  
    background-color: #13a4a4;  
    padding: 5px;  
    border-radius: 2px;  
    text-decoration: none;  
}  
  
.nav .blog {  
    background-color: red;  
}
```

Combinadores: Adjacent Siblings (combinators)

Nos permiten combinar múltiples selectores y crear una mayor especificidad.

Hermano adyacente o cercano Adjacent sibling	Hermano general General sibling
div + p { } ...	div ~ p { } ...
Hijo Child	Descendiente Descendant
div > p { } ...	div p { } ...

EJ

Hermano adyacente o cercano

Si queremos que todas las h2 que estén cerca de un p se pinten de color rojo hacemos así:



```
h2 + p {  
    color: red;  
}
```

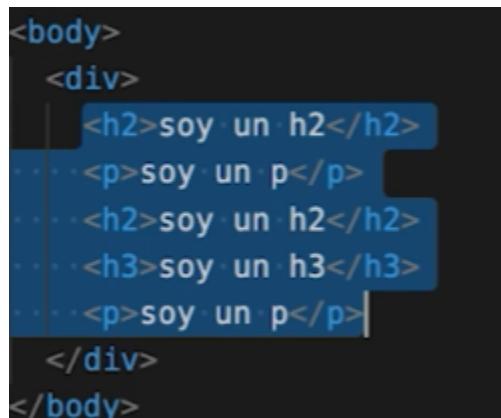


soy un h2
Soy un p

Hermano general

Usamos simbolo equivalente ~

Vamos a modificar todas las p que tengan como hermano general un h2:



```
<body>  
  <div>  
    <h2>soy un h2</h2>  
    <p>soy un p</p>  
    <h2>soy un h2</h2>  
    <h3>soy un h3</h3>  
    <p>soy un p</p>  
  </div>  
</body>
```

soy un h2

soy un p

soy un h2

soy un h3

soy un p



```
h2 ~ p {  
    color: red;  
}
```

Hijo y Descendientes

```
<body>
  <div>
    <article>
      <p>soy un texto</p>
    </article>
    <article>
      <p>soy un texto</p>
    </article>
    <section>
      <div>
        <p>soy un texto</p>
      </div>
    </section>
    <p>soy un texto</p>
  </div>
</body>
```

```
#div > p {
  color: red;
```

soy un texto

soy un texto

soy un texto

soy un texto

La regla dice que nuestra p hija directa de div sea de color rojo. Las dos de arriba no se pusieron de color rojo porque su padre directo es article.

Ahora usamos descendientes, a todas las etiquetas p que estén dentro de un div aplicarle color rojo .

```
div p {
  color: red;
```

Medidas

Absolutas	Relativas
px	%
	em
	rem (root em)
	max-width / max-height
	min-width / min-height
	vw (viewport width)
	vh (viewport height)

EM = es un acrónimo de elemento y lo que hace es tomar el tamaño de fuente que tenga el padre directo ejemplo:

```
.container {  
    font-size: 20px  
}  
  
.container div {  
    font-size: 2em  
}
```

Unidades absolutas

Generalmente se usa el pixel “px”.

```
h1{  
    width: 500px;  
}
```

El problema de usar medidas absolutas es que pueden ser desproporcionadas para distintos tipos de dispositivos.

Unidades Relativas

*

Porcentaje

```
/*Ocupará todo el ancho de la pantalla  
sin importar el tamaño*/  
h1{  
    width: 100%;  
}
```

em

Depende exclusivamente del cuerpo tipográfico. Es decir del font-size de la etiqueta de su elemento padre.

```
body{  
    font-size: 20px;  
}  
h1{  
    font-size: 2em; /* 2 x 20 = 40px */  
}
```

Lo bueno de esta unidad es que si en un futuro modificamos el valor del font-size, entonces sus otros valores se ajustarán como podría pasar en el siguiente ejemplo de código.

```
p {  
    font-size: 24px;  
    margin-top: 1em; /* 1 x 24 = 24px */  
    margin-bottom: 1em; /* 1 x 24 = 24 px*/  
}
```

rem (root em)

En este caso, rem siempre va a depender del elemento raíz, de la etiqueta html.

```
html{  
    font-size: 20px;  
}  
h1{  
    font-size: 2em; /* 2 x 20 = 40px */  
}
```

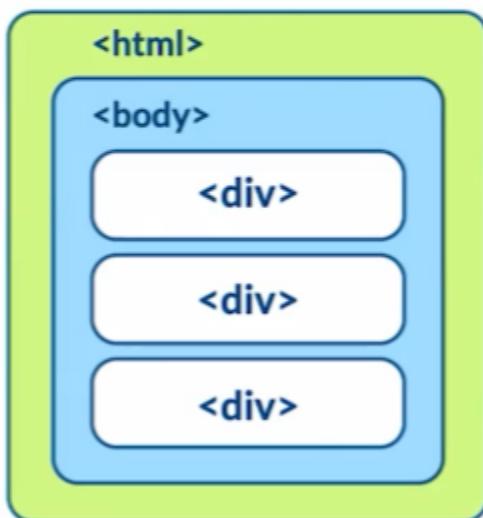
Truco que nos recomendó el profe

```
/*Como por defecto el font-size es 16px,  
el 62,5% nos daría 10px*/  
html {  
    font-size: 62.5%;  
}  
p {  
    font-size: 1.6rem; /* 1.6 x 10 = 16px  
*/  
}  
/*entonces podemos calcular rem  
fácilmente multiplicando por 10
```

Position

Como posicionar, por defecto están en estáticos.

Posicionamiento de elementos

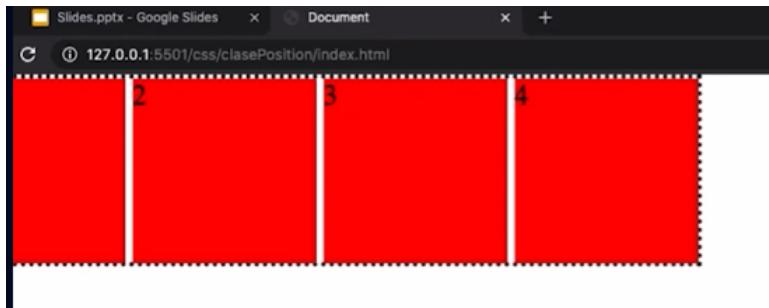


Position	
static	Por defecto
absolute	
relative	
fixed	
sticky	

Reiniciar estilos:

```
* {  
    box-sizing: border-box;  
    margin: 0;  
    padding: 0;  
}
```

Ejemplo:



```
</head>  
<body>  
    <div class="parent">  
        <div class="box" id="one">1</div>  
        <div class="box" id="two">2</div>  
        <div class="box" id="three">3</div>  
        <div class="box" id="four">4</div>  
    </div>  
</body>  
</html>
```

```
margin: 0;
padding: 0;
}
.parent {
  border: 2px solid black dotted;
  display: inline-block;
}
.box {
  display: inline-block;
  background-color: red;
  width: 100px;
  height: 100px;
}
```

Display

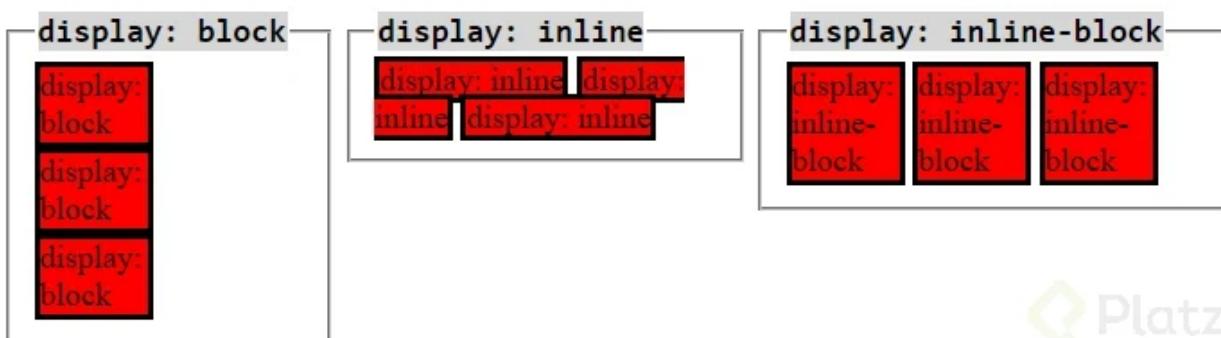
En un elemento con display:inline no puedo usar margin ni padding arriba ni abajo, solo derecha e izquierda. Tampoco se puede aplicar width o height.

En un elemento con display:block el contenido del elemento toma el 100% del width, se puede usar margin y padding por todos los lados.

En un elemento con display:inline-block, se puede usar margin y padding por todos lados, así como darle width y height, y el contenido es del mismo tamaño que el elemento.

Etiquetas como p y div vienen por Default con un display:block

Etiquetas como span viene por Default con un display:inline



Display Flex

Flexbox Cheat Sheet

simonpaix

Parent properties

display: enables flex context for all direct children.

```
.container{  
  display: flex; // or inline-flex  
}
```

flex-direction: sets the main-axis.

row
row-reverse

column

column-reverse

flex-wrap: allows the items to wrap as needed.

no-wrap
wrap
wrap-reverse

justify-content: defines alignment along the main axis.

flex-start
flex-end
center
space-between
space-around
space-evenly

align-items: defines alignment along the cross axis.

flex-start
flex-end
center
stretch
baseline
space-between
space-around
baseline
stretch

LearnPine

Children properties

order: changes the order of flex items.

```
.item {  
  order: 3 // the default is 0  
}  
  
-1 0 1 2 3
```

flex-grow: allows item to grow using remaining space.

```
.item-1 { flex-grow: 0 } .item-1 { flex-grow: 1 }  
//default  
1 2 3 1 2 3
```

Tip: If all items have flex-grow: 1, the remaining space is distributed equally.

flex-shrink: defines the ability for a flex item to shrink.

```
.one { flex-shrink: 1; }  
.two { flex-shrink: 2; }  
.three { flex-shrink: 3; }  
.four { flex-shrink: 4; }
```

Tip: Defaults to 1. The highest value the more it shrinks compared to siblings.

flex-basis: sets the default size of a flex item. It accepts:

- specific values : pixels, rm, %
- auto : defaults to width or height property
- content : automatic sizing, based on its content
- global values : inherit, initial, unset

align-self: overrides default alignment (or the one specified by align-items) for a specific item.

flex-start
center
flex-end
baseline
stretch

FLEXBOX FROGGY

<https://flexboxfroggy.com/#es>

Alignment in CSS

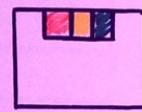
justify-content



flex-start



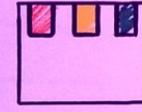
flex-end



center



space-between



Space-Around



space-evenly.

align-items



flex-start



flex-end



stretch

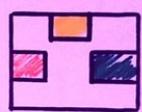


baseline

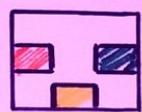


center

align-self



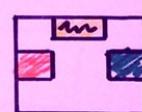
flex-start



flex-end



stretch

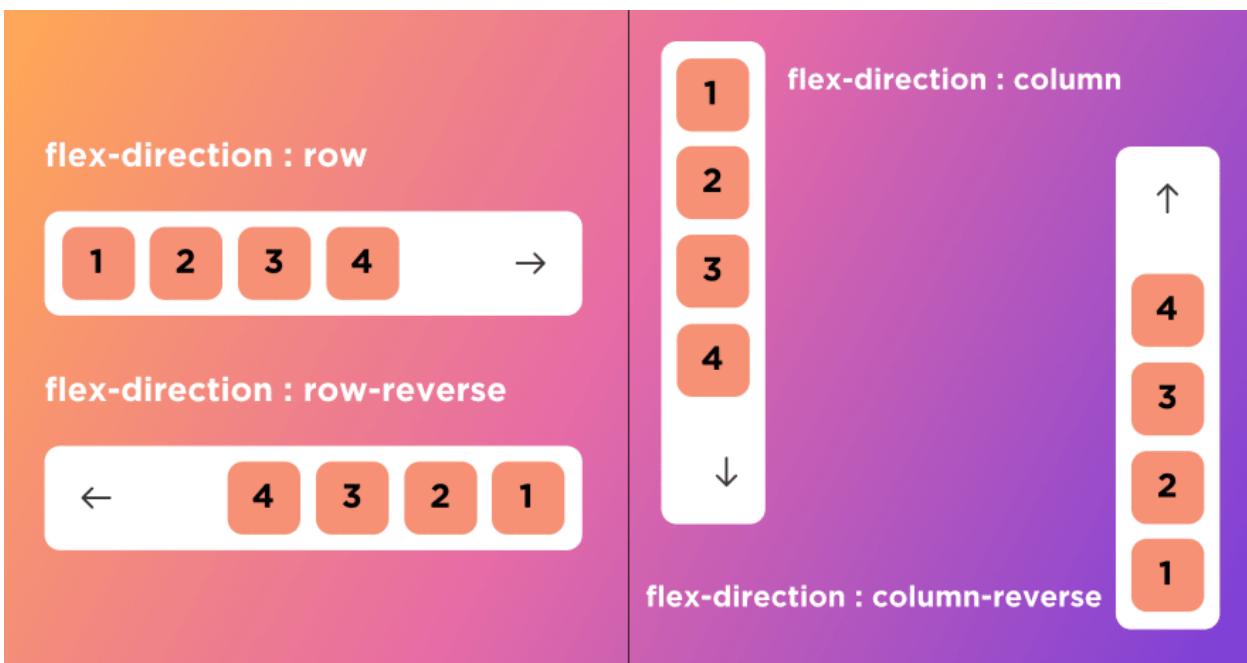
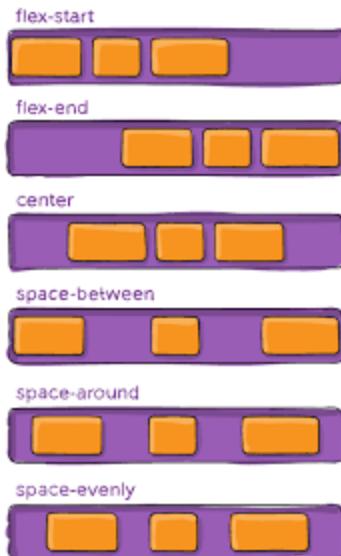


baseline



center

* align-self is applied on yellow item.



Las ranas necesitan ponerse en el mismo orden que sus hojas de lirio usando **flex-direction**. Esta propiedad CSS define la dirección de los elementos en el contenedor, y acepta los siguientes valores:

- **row**: Elementos son colocados en la misma dirección del texto.
- **row-reverse**: Elementos son colocados en la dirección opuesta al texto.
- **column**: Elementos se colocan de arriba hacia abajo.
- **column-reverse**: Elementos se colocan de abajo hacia arriba.

Flexbox layouts

```
.container {  
    border: 0.3rem solid black;  
    display: flex;  
    align-items: flex-start;  
    height: 50vh;  
}
```

Variables

VARIABLES

40

Las variables en CSS deben declararse dentro de un selector.

Generalmente se utiliza :root para que la variable sea global.

El nombre de una variable debe comenzar con dos guiones(--) y

distingue entre mayúsculas y minúsculas.



```
:root {  
    --my-color: yellow;  
}  
body {  
    color: var(--my-color); → todos los textos, párrafos,  
    titulos, encabezados, etc  
    serán de color amarillo por  
    HERENCIA.  
}  
h1 {  
    --my-color: red; → Pero todos los h1 serán rojos porque  
    utilizamos la CASCADA para redefinir  
    el valor de la propiedad --color  
}
```

Platzi

Variable setiadas

```
:root {  
  --primary-color: #003476;  
  --secondary-color: #b4d2f7;  
  --header-size: 4rem;  
  --font: 1.8rem;  
}
```

Y ahora pasa usarlas

```
header {  
  width: 100vw;  
  height: 15vh;  
  background-color: var(--primary-color);  
}
```

Web fonts

Genericas	Fuentes	
serif	Times New Roman	Georgia
sans-serif	Helvetica	Verdana
cursive	Dancing Script	Great Vibes
monospace	Courier New	Roboto Mono

Descargar fuentes en

fonts.google.com

The screenshot shows the Roboto font family page on Google Fonts. The main area lists five font styles: Medium 500 italic, Bold 700, Bold 700 italic, Black 900, and Black 900 italic. Each style has a '+ Select this style' button and a '- Remove this style' button. To the right, there's an 'Embed' section with code snippets for HTML and CSS, and a 'Selected family' sidebar.

Glyphs

A	B	C	Č	Ć	D	Đ	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	Š	T	U	V	W	X	Y	Z	Ž
a	b	c	č	ć	d	đ	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	š	t	u	v	w	x	y	z	ž

Responsive Design

Media Queries importantes

<https://www.mydevice.io/>

Mobile First / Only

```

@media (min-width: 480px) {
    ...
}

@media (min-width: 768px) {
    ...
}

@media (min-width: 1024px) {
    ...
}

```

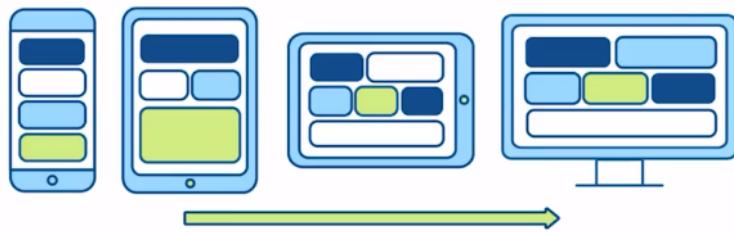
Iniciar desde las pantallas más chicas a las pantallas más grande

```
  <link href="style.css" rel="stylesheet"<!-- Tus estilos para enfocados a mobile --&gt;
  &lt;link href="tablet.css" rel="stylesheet" media="screen and (min-width: 768px)"&gt;</pre>
```

Estrategias de responsive: mostly fluid

<https://web.dev/learn/design/>

Mostly Fluid



```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
html {
  font-size: 62.5%;
}
.container {
  display: flex;
  flex-wrap: wrap;
}
.c1,
.c2,
.c3,
.c4,
.c5 {
  width: 100%;      [
  min-width: 150px;
  height: 150px;
}
.c1 {
  background-color: #003476;
}
.c2 {
  background-color: #0062d2;
}
.c3 {
  background-color: #b4d2f7;
}
.c4 {
  background-color: #d5dfef;
}
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <link rel="stylesheet" href="./style.css" />
  </head>
  <body>
    <main class="container">
      <div class="c1"></div>
      <div class="c2"></div>
      <div class="c3"></div>
      <div class="c4"></div>
      <div class="c5"></div>
    </main>
  </body>
</html>
```



Modo tablet

Se definen al final del archivo css, definimos el media y le marcamos que cuando pase de 600 px cambien. Cuando pasa a modo tablet cambiamos algunos componente de jugar.

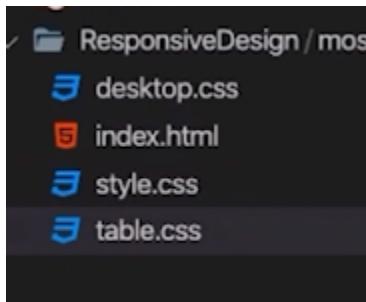
```
@media (min-width: 600px) {
  .c2,
  .c3,
  .c4 {
    width: 50%;
  }
}
```

Ahora para cuando pasa los 800 px

```
@media (min-width: 800px) {  
    .container {  
        width: 800px;  
        margin-left: auto;  
        margin-right: auto;  
    }  
    .c1 {  
        width: 60%;  
    }  
    .c2 {  
        width: 40%;  
    }  
    .c3,  
    .c4 {  
        width: 33%;  
    }  
    .c5 {  
        width: 34%;  
    }  
}
```

Si queremos sacarle los espacios blancos de los costados le sacamos las cosas del .container.

Buena práctica separar los archivos.



En el html

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8" />  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <title>Document</title>  
    <link rel="stylesheet" href="./style.css" />  
    <link  
      rel="stylesheet"  
      href="./table.css"  
      media="screen and (min-width: 600px)"  
    />  
    <link  
      rel="stylesheet"  
      href="./desktop.css"  
      media="screen and (min-width: 800px)"  
    />  
  </head>
```

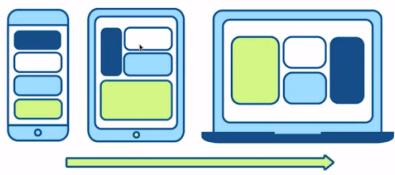
Por ejemplo en table.css quedaría

A screenshot of a code editor window titled "ResponsiveDesign > mostlyFluid > table.css". The code shown is:

```
1 .c2,
2 .c3,
3 .c4 {
4   width: 50%;
5 }
6
```

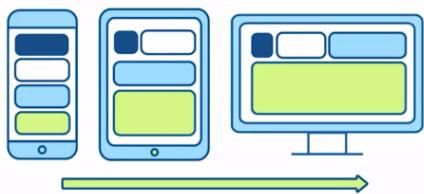
Layout shifter css

Layout Shifter



Column drop

Column Drop



Recomendaciones

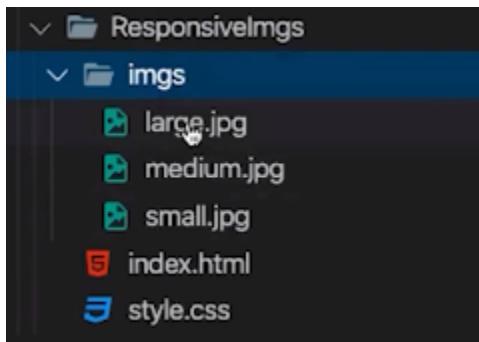
Dividiendo nuestro codigo lograremos que sea mas rapido

Separa siempre tus archivos de CSS
por break point

- mobile.css / style.css
- tablet.css
- desktop.css

Responsive imagenes

Descargamos 3 imágenes con tamaños distintos:



```
ResponsiveDesign > Responsivelmgs > index.html > html > body > main > picture > 6
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Document</title>
7      <link rel="stylesheet" href="./style.css" />
8  </head>
9  <body>
10     <main>
11         <picture>
12             <source media="(min-width:1300px)" srcset="./imgs/large.jpg" />
13             <source media="(min-width:1000px)" srcset="./imgs/medium.jpg" />
14             
15         </picture>
16     </main>
17 </body>
18 </html>
19
```

```
ResponsiveDesign > Responsivelmgs > style.css
1  .|
2  img {
3      width: 100%;
4  }
```

Semántica / Accesibilidad

Recomendación de curso de accesibilidad = <https://platzi.com/cursos/accesibilidad-web/>

Textos: no usar píxeles para que los usuarios puedan agrandar las letras. El texto siempre se usa rem

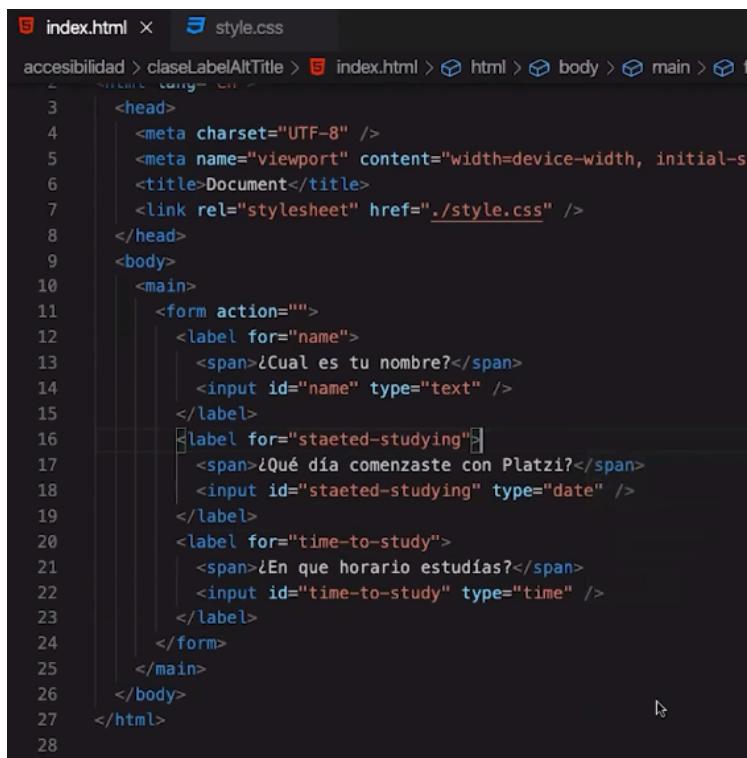
```
h1 {  
    font-size: 3rem;  
}
```

Labels en los inputs que utilicemos (se usa mucho en formularios) - Si quieras saber más sobre labels te dejo este par de recursos: label - MDN y HTML <label> Tag

El atributo alt (en las imágenes) - El **alt** facilita que una persona con limitación visual que usa un software para leer la página web pueda escuchar de qué se trata la imagen.

El atributo title que se usa en la etiqueta de anchor (<a>) y en la etiqueta imagen ().

Podemos hacer que con un click en el label ya te ponga automáticamente en el input para llenar:



```
index.html x style.css  
accesibilidad > claseLabelAltTitle > index.html > html > body > main > fo  
3   <head>  
4     <meta charset="UTF-8" />  
5     <meta name="viewport" content="width=device-width, initial-sco  
6     <title>Document</title>  
7     <link rel="stylesheet" href="./style.css" />  
8   </head>  
9   <body>  
10  <main>  
11    <form action="">  
12      <label for="name">  
13        <span>¿Cuál es tu nombre?</span>  
14        <input id="name" type="text" />  
15      </label>  
16      <label for="staeted-studying">  
17        <span>¿Qué día comenzaste con Platzi?</span>  
18        <input id="staeted-studying" type="date" />  
19      </label>  
20      <label for="time-to-study">  
21        <span>¿En qué horario estudias?</span>  
22        <input id="time-to-study" type="time" />  
23      </label>  
24    </form>  
25  </main>  
26 </body>  
27 </html>
```

Para software de lecturas, screen reader.

ALT

Para darle una descripción de la imagen

```
<section>
  
</section>
```

Recomendación para seguir con web:

Lecturas recomendadas

-  Escuela de Desarrollo Web
<https://platzi.com/web/> 
-  Curso Básico de JavaScript
<https://platzi.com/clases/basico-javascript/> 
-  Curso de CSS Grid Layout
<https://platzi.com/clases/css-grid-layout/> 
-  Curso de JavaScript Engine (V8) y el Navegador
<https://platzi.com/clases/javascript-navegador/> 
-  Curso de Debugging con Chrome DevTools
<https://platzi.com/clases/devtools/> 
-  <https://platzi.com/clases/ecmascript-6/> 
<https://platzi.com/clases/ecmascript-6/> 
-  ¿Qué sigue después de aprender HTML y CSS?
<https://platzi.com/blog/despues-aprender-html-css/> 

5 ETIQUETAS HTML ESENCIALES | CHEAT SHEET

Estructura de una etiqueta HTML

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8   ...
9 </body>
10 </html>
11 
```

Para la base	
Etiqueta	Descripción
<!doctype>	Indica la versión de HTML
<html>	Es la raíz del archivo
<head>	Engloba los metadatos
<title>	Título en la pestaña del navegador
<link>	Enlaza archivos CSS
<style>	Escribe CSS dentro del doc HTML
<script>	Enlaza o escribe JavaScript en el doc HTML
<meta>	Características sobre la página
<body>	Elementos visibles en la página

Para estructurar el contenido	
Etiqueta	Descripción
	Crea una lista sin orden
	Crea una lista ordenada
	Define los artículos de las listas

Para el texto	
Etiqueta	Descripción
<p>	Crea párrafos
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	Estructura tus títulos y subtítulos. h1 es el más importante de la jerarquía
Este es un h1 Este es un h2 Este es un h3 Este es un h4 Este es un h5 Este es un h6	
	Texto en negritas
<i>	Texto en cursiva
<u>	Text subrayado
<a>	Agrega un link
	Texto importante de la página
<blockquote>	Texto en formato de cita
 	Crea un salto de linea

Para multimedia	
Etiqueta	Descripción
	Inserta una imagen
<video>	Inserta un video
<iframe>	Inserta contenido de otras páginas
<audio>	Inserta un audio

Para crear tablas	
Etiqueta	Descripción
<table>	Crea una tabla
<tr>	Crea las filas
<td>	Crea las columnas
<th>	Crea el título de cada columna
<col>	Especifica las propiedades de las columnas

Para crear formularios	
Etiqueta	Descripción
<form>	Crea un formulario
<input>	Campo para introducir datos
<label>	El título del input
<textarea>	Campo para introducir mucho texto
<button>	Un botón

CONTACTO

Nombre:

E-mail:

Mensaje:

Enviar Mensaje

Practica HTML y haz desarrollo web profesional en platzi.com/htmlpro