3 - Crea tu primer sitio web

Estructura de un archivo: <nombre de archivo>.<extension>

archivo\*\*.extension\*\*

carta\*\*.docx\*\*

hoja\*\*.xls\*\*

pagina\*\*.html\*\*

**HTML (Hyper Text Markup Language):** lenguaje sobre el que está construido internet, es un formato que engloba otras tecnologías que nos permiten usar sitios WEB. Ej: gifs, imagenes (png, jpg), video, diseño,

**CSS (Hojas de estilo en cascada):** lenguaje de declaración del estilo de diseño de una página web. Puede ir dentro del HTML o aparte.

**JavaScript:** Es un lenguaje de programación, de los más usados en el planeta. Con el se puede construir lo que sea. Pero antes se debe estructurar la información con HTML

💡 HTML define la estructura de una página, CSS el diseño de la interfaz y JS se encarga de las funcionalidades

Por dentro todos los archivos son textos, organizados de distintas forma. Por eso es posible abrir cualquier archivo con un editor de texto como block de notas

1ra página web con html:

<h1>Mi primer sitio WEB</h1>

Hola <strong>querida</strong> clase

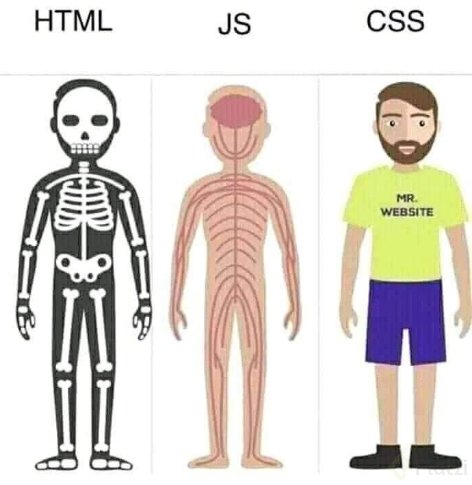
**HTML es muy sencillo de entender 💚**

.

Es fácil, imagina que tienes una hojita en blanco y quieres empezar a añadir cositas, en este caso, usamos HTML para empezar a añadir cositas en esa hojita en blanco 👇.  
.  
Siguiendo el ejemplo de la clase, si yo quiero agregar un título a mi página web ¿qué debería hacer? Pues por medio de HTML usar la etiqueta correspondiente para añadir ese título 😄, simplemente hay que decirle:  
.  
<h1>Este será mi título 😼</h1>  
.  
Ahora imagina que quieres agregar un párrafo… ¡también podemos! Solamente hay que decirle a HTML que me agregue un párrafo, así 👇:  
.  
<p>Aquí adentro puedo poner mi párrafo 😄</p>  
.  
¡Así de fácil! Simplemente tienes que decirle a HTML qué es lo que quieres agregar y él lo agregará. HTML es como un robotcito que siempre te hará caso y agregará todo lo que quieras 💚.  
.

**¿Y cuál es la diferencia entre <h1> y <title>? 😦**

.  
Bueno, <title> solamente lo usaremos para definir el título de la pestaña en nuestro navegador, solo sirve para eso, esta etiqueta nunca se pintará en nuestra página web, solo en la pestaña del navegador. Por otra parte, <h1> sí se pintará dentro de nuestra página web y esta etiqueta la podemos usar para poner títulos visibles dentro de la misma 😄.



Que es el Frontend?

**Frontend** es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

Un **programador Frontend** debe saber de código que entienda el navegador (HTML, CSS y JavaScript) para poder usar algunos *frameworks* o librerías que expanden sus capacidades para crear cualquier tipo de interfaces de usuarios. React, Redux, Angular, Bootstrap, Foundation, LESS, Sass, Stylus y PostCSS son algunos de ellos.

**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/>

Además de los estándares, frameworks, preprocesadores y librerías que maneja un Frontend Developer, me gustaría compartir estas herramientas que nos permiten un mejor desarrollo.  
**Herramientas**

1. [Mediaqueri](https://mediaqueri.es/): Algunas ideas sobre como deberíamos implementar el diseño responsive en nuestro sitio.
2. [Coolors](https://coolors.co/palettes/trending): Paletas de colores.
3. [Unsplash](https://unsplash.com/): Imagenes gratis (Da los créditos a los fotógrafos 😉).
4. [FontPair](https://fontpair.co/): Ver cuales fuentes puedes combinar. Y obviamente [Google Fonts](https://fonts.google.com/) para obtener esas fuentes.
5. [Icons8](https://icons8.com/): Iconos, vectores, música y algunos recursos más…

**Documentaciones**

1. [CSS](https://cssreference.io/)
2. [HTML](https://htmlreference.io/)
3. [Browserdiet](https://browserdiet.com/es/): Optimizar sitios web (Perder peso en la web 😛). Recuerda que Platzi tiene también un curso de [Web Performance](https://platzi.com/clases/web-performance/)
4. [WCAG](http://www.sidar.org/traducciones/wcag20/es/): Guía para mejorar la accesibilidad de nuestro sitio. Recuerda que Platzi tiene el [curso de accesibilidad web](https://platzi.com/clases/accesibilidad-web/).

**Practica jugando**

1. [Grid Layout](https://cssgridgarden.com/#es)
2. [Flexbox](https://flexboxfroggy.com/#es)

**Indispensables**

1. [Platzi](https://platzi.com/clases/2008-html-css/31054-que-es-el-frontend/platzi.com): Para nunca pares de aprender

Que es el 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:

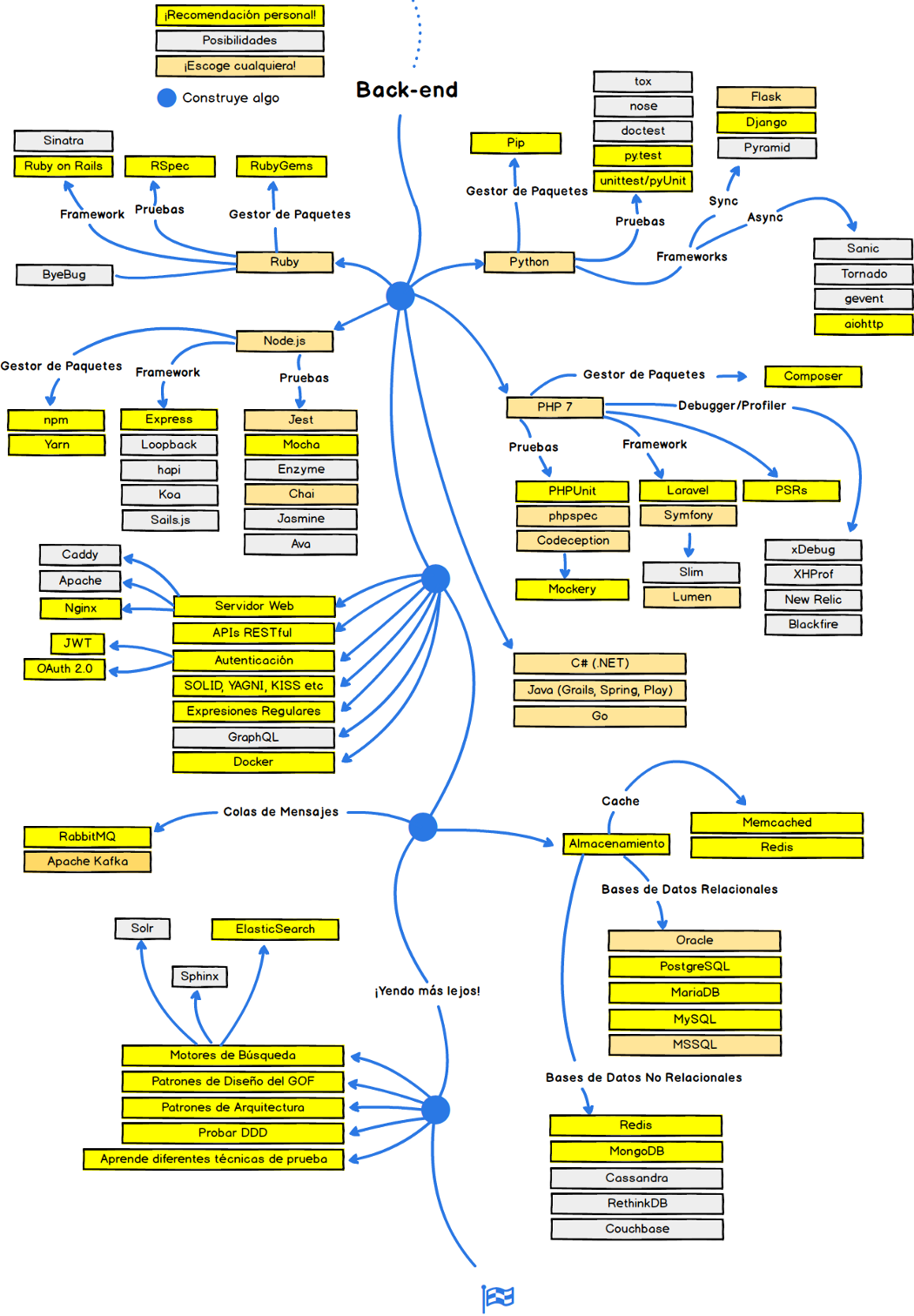
* Google Cloud
* DigitalOcean
* AWS
* Heroku, entre otras.

**¿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).



Qué es un Full Stack?

**Full Stack** 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 para crear nuevas plataformas.



El nacimiento de tecnologías que funcionan entre el frontend y el backend 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 empleando un único lenguaje de programación, como Django para Python.

## Qué es un desarrollador Full Stack

Las y los **desarrolladores Full Stack** 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) se refiere al grupo de tecnologías que componen un sitio web en todos los aspectos (desde la base de datos, hacia el manejo lógico y la interfaz visual). Una o un **desarrollador Full Stack** en teoría es capaz de manejar la pila completa de un sitio, tanto de frontend como de backend, además sabe utilizar su base de datos.

**Qué hace un desarrollador Full Stack**

Una desarrolladora o **desarrollador Full Stack** entiende muy bien cómo funciona un producto web de principio a fin, desde su diseño en mockup y deploy hasta producción.

Este tipo de programador o programadora no maneja por completo todas las tecnologías de ambas partes, pues cada una requiere conocimiento profundo. De hecho, no es recomendado profundizar en tantas especialidades y no es sano. El desarrollo web evoluciona muy rápido y cada dos o tres meses encontrarás algo nuevo.

**Cómo ser un desarrollador Full Stack**

Para convertirse en Full Stack developer debes dominar las dos áreas básicas: [backend y frontend](https://platzi.com/blog/que-es-frontend-y-backend/" \t "_blank). Esto te ayudará a comprender cómo funciona cada uno y a generar una comunicación entre sí. La mayoría de la interacción entre estos dos mundos se basa en consumo de API y comprender cómo consumir una API desde el Frontend te ayudará a entender a cómo construir una API desde el Backend.

Además, aprender de las diferentes formas que tenemos de subir en internet una aplicación web te ayudará a entender las diferentes estrategias de *deploy* que existen y te permitirá ampliar tus conocimientos en el mundo del desarrollo web.

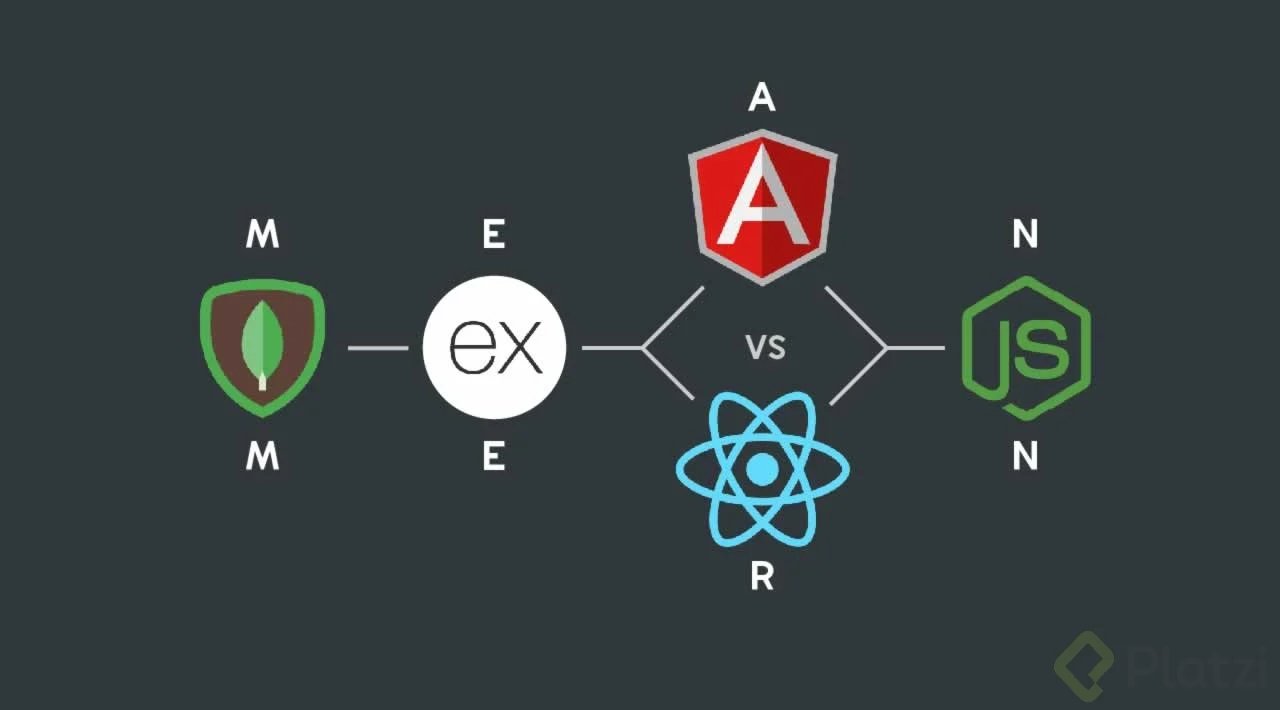
**Tecnologías Full Stack para desarrolladores**

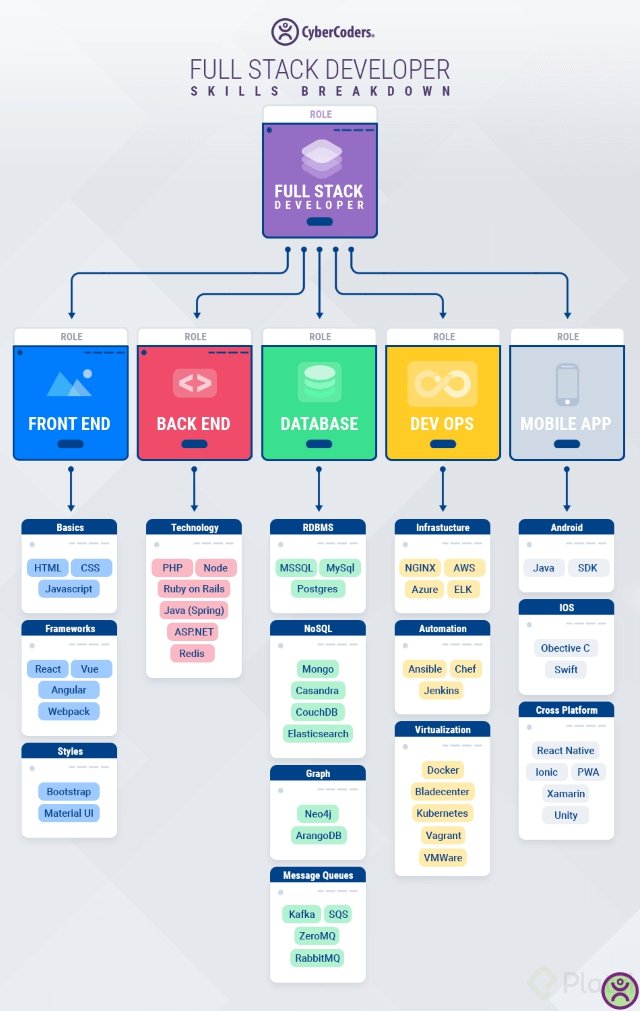
Aprender un *stack* conocido puede ayudarte a dar tus primeros pasos como desarrollador Full Stack y a escalar un proyecto a producción de una manera más rápida y con una mayor interacción. Para eso necesitas estas herramientas.

* LAMP (Linux, Apache, MySQL, PHP)



* MERN (MongoDB, Express.js, React, Node)
* PERN (PostgreSQL, Express.js, React, Node)





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:

## Sitos Web Estáticos

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

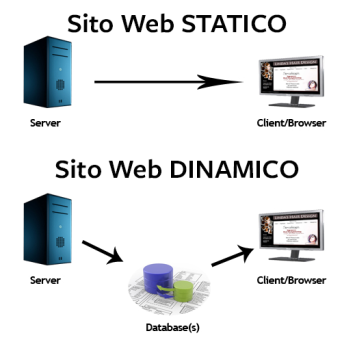
También conocidos como aplicaciones web, actualizan su información con respecto a la interacción del usuario. Dependen de una base de datos, de donde extrae e ingresa información. Serán diferentes, dependiendo del usuario que la use y la información que se ingrese.

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

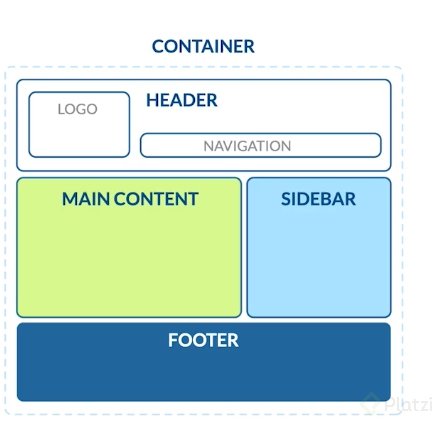


HTML: Hypertext Markup Language

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](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.

Index y su estructura básica: head

Casi siempre, nuestro primer archivo HTML, debe llamarse “index”

<!DOCTYPE html>

Es el tipo de documento: HTML

<html lang="en">

</html>

Es el cuerpo total, todo lo que va dentro es nuestra pagina

<head>

Va todo lo importante para el navegador, para cargar el proyecto, Pero no es visual para el usuario, Dependencia, librerías, fuentes, CSS va ligado en el Head, aquí es donde los mandamos a llamar. (Va el cerebro, nuestras conexiones nerviosas, lo q sabemos)

Etiquetas meta

        <meta charset="UTF-8"> #(ES PARA QUE EL NAVEGADOR ENTIENDA CARACTERES ESPECIALES por ejemplo emojis) UTF 16 y 32 es para otros caracteres

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

#(ESTA ES AUTOMATICA; SUPONGO Q ES PA COMPATIBILIDAD)

        <meta name="viewport" content="width=device-width, initial-scale=1.0"> (ESTO ES PARA DECIRLE AL NAVEGADOR QUE EL TAMAÑO DE LA PAGINA SEA DE………)

<meta name="description" content="Esta pagina te mostrará foto de gatitos."

#(Tipo de descripcion de nuestra pagina, por ejemplo, ESTA PAGINA TE MOSTRARÁ FOTOS DE GATOS “ESTO AYDUA EN CEO; EN EL BUSCADOR APARECE ESTO”)

<meta name="robots" content="index,follow"> (ESTO ES PARA LOS ROBOTS DE BUSQUEDA, LE DAMOS AUTORIZACIÓN A PODER SEGUIR NUESTRA PAGINA)

Etiquetas meta, dan cierta información al navegador, para que sepa como marcar nuestro proyecto.

<link rel="stylesheet" href="./basic.css">

Aqui linkeamos nuestro estiloo en el document de CSS.

Index y su estructura básica: body

**Body** es la etiqueta que identifica la parte visible de nuestro sitio web. Dentro del body, se añadirán las etiquetas para marcar los elementos visuales del sitio web, como logotipo, menús de navegación, contenido principal, entre otrs. [Es muy importante usar HTML semántico](https://platzi.com/clases/1802-accesibilidad-web/26072-que-es-el-html-semantico-y-por-que-es-importante/) y no llenar todo de <div> para que nuestro sitio sea mejor interpretado por el navegador y, por lo tanto, más accesible.

## 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 calculo.
* ul y ol: listas de items.
* 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 artícle 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.

## Ejemplo de un body con etiquetas semánticas HTML.

<body>

    <header> <!--Sección superior de nuestro website-->

      <nav></nav> <!--Sección de navegación de nuestro website, siempre dentro del header-->

    </header>

    <main> <!--Main es el contenido central de nuestro website, "la parte del medio"-->

      <section>

        <!--Nuestro website puede estar divido por secciones, por ejemplo platzi tiene 3: El navegador de cursos y rutas, el feed y nuestras rutas de aprendizaje-->

        <article>

          <!--Contenido independiente de la página. Es reutilizable-->

        </article>

      </section>

      <ul> <!--Lista desordenada: Sin numerar-->

        <li><!--Item List. Elementos de la lista--></li>

      </ul>

      <ol></ol> <!--Lista ordenada: Numerada-->

    </main>

    <footer> <!--Sección final de nuestro website-->

    </footer>

    <p>Soy un texto</p> <!--Párrafo, texto-->

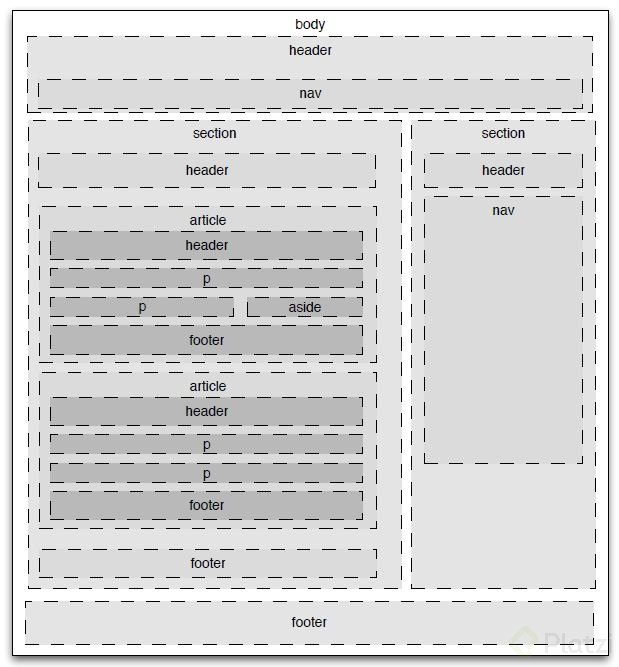
    <h1>Soy un titulo</h1>

    <!--Títulos, muestran el texto más grande y con negrilla. Existen desde el h1 al h6-->

    <a href="#">Soy un link</a>

    <!--Enlaces/links que nos permitirán movernos entre páginas.-->

  </body>

 <article>  
<aside>  
<details>  
<figcaption>  
<figure>  
<footer>  
<header>  
<main>  
<mark>  
<nav>  
<section>  
<summary>  
<time>  
estos son los elementos semanticos mas utiles

LINK DE MÁS ETIQUETAS: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

Anatomía de una etiqueta de HTML.

Una etiqueta HTML puede tener tantos atributos como desees, 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

Las imágenes representan una pieza fundamental al momento de mostrar contenido para web. Aquí conoceremos los principales tipos de imágenes web y sus formatos.

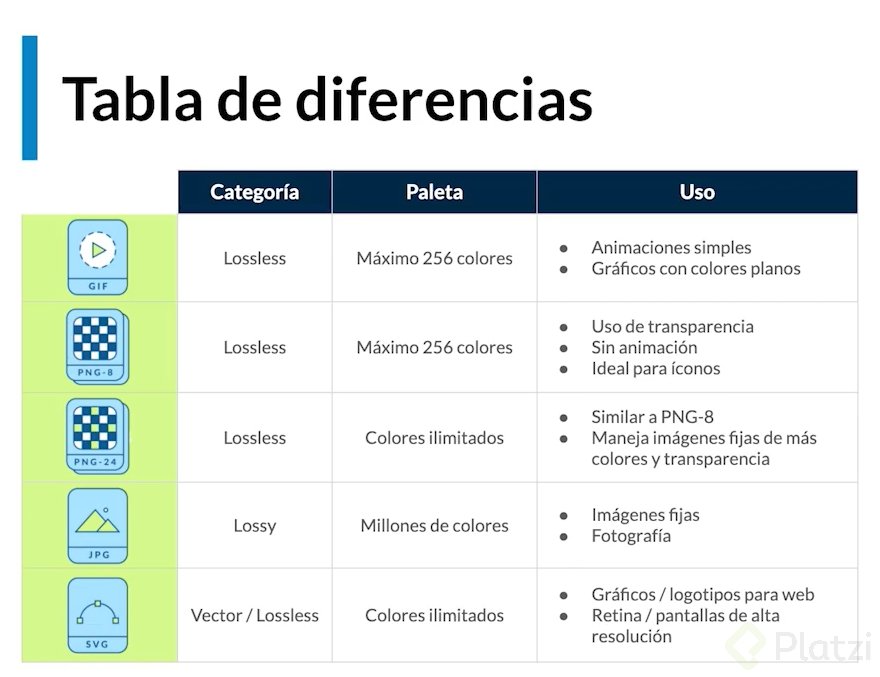
## Tipos de imágenes para web

### 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.



## 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 comprimirlas, 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.