**Escala para responsive a cualquier dispositivo.**

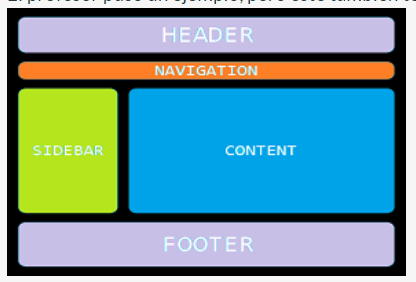
meta name="viewport" content="width=device-width, initial-scale=1.0"

**Body Structure**

header

section

footer



**Lists**

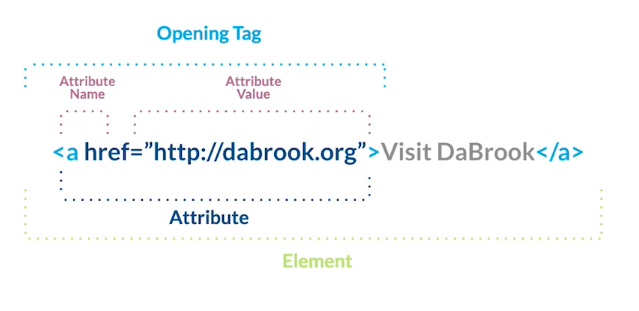
ul => unordened list

ol => ordened list

li => list item

a => "link"

**Opening TAG**



**Etiquetas Multimedia**

**Images**

Lossy vs Lossless (with loss/without loss)

There are two types of images, lossy or with loss, it depends that how the format handles images

**Lossless**

The image formats without loss get all data from our original file. These kinds of images used to be very heavy.

**Lossy**

The image formats with loss are very similar to their original file, which can reduce the amount of the color or analyze the image to find unnecessary data. It will reduce the file size, although could reduce the image quality.

**Formats:**

* **GIF-Lossless:** (Grafphics Interchange format)
* **PNG 8 – Lossless:** (Portable Network Graphics) 256 colors. Can use transparencies
* **PNG 24 – Lossless:** (Portable Network Graphics) More than 256 colors. Can use transparencies
* **JPG/JPEG – Lossy:** (Photographic Experts Group)
* **SVG – Vector/Lossless** (Scalable Vector Graphics)



**Images Optimization**

The average size of about 70 kB

Customization image options:

* Improved the size of your images **(Tiny PNG)** <https://tinypng.com/>
* Remove metadata of your images **(Verexif)** <https://www.verexif.com/>

**TAG IMG**

<img src="" alt="">

**src :** path or directory where images are located

**alt:**  Image description in case the image could not be upload

**Tag <figure>** Create a container for images, do not use <div>, good practice

**Tag <figcaption>** Description or text for image

**TAG VIDEO**

<video src="./videos/Maroon 5 - Girls Like You ft. Cardi B.mp4" controls></video>

**src:** path or directory where video is located

**controls:** show control buttons to play the video

**preload:** upload video before they click the play button

**#t=10,60** Define start and end time

**FORMS**

<form action="">

        <label for="nombre">

            <span>¿Cual es tu nombre?</span>

            <input type="text" name="" id="nombre" placeholder="nombre">

        </label>

        <label for="fecha-inicio">

            <span>¿Qué día comenzaste a aprender?</span>

            <input type="date" name="" id="fecha-inicio">

        </label>

        <label for="horario">

            <span>¿En que horario estudias?</span>

            <input type="time" name="" id="horario">

        </label>

    </form>

**Tag <form>:** Create a form

**Tag <label>:** Box to insert text

**Tag <span>:** Insert a coment

**Tag <input>:** Define the Kind of data that would put into the label

**Calendar**

<!-- <form action="">

        <label for="hora">

            <span>Hora</span>

            <input type="time" name="hora" id="hora">

        </label>

        <label for="dia">

            <span>Día</span>

            <input type="date" name="dia" id="dia">

        </label>

        <label for="semana">

            <span>Semana</span>

            <input type="week" name="semana" id="semana">

        </label>

        <label for="mes">

            <span>Mes</span>

            <input type="month" name="mes" id="mes">

        </label>

        <input type="submit"/>

    </form> -->

    <form action="">

        <label for="calendario">

            <span>Calendario</span>

            <input type="datetime-local" name="calendario" id="Calendario">

        </label>

        <input type="submit" name="" id="">

    </form>

There are two ways to créate a calendar:

1.- Create one input for each label hour, day, week, mount.

2.- Create an input type=”**datetime-local”.** More efficient

**Autocomplete Forms**

<form action="">

            <label for="nombre">

                <span>¿Cual es tu nombre</span>

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

            </label>

            <label for="correo">

                <span>¿Cual es tu correo?</span>

                <input type="email" name="correo" id="correo" autocomplete="email" required>

            </label>

            <label for="pais">

                <span>¿Cual es tu pais?</span>

                <input type="text" name="pais" id="pais" autocomplete="country" required>

            </label>

            <label for="cp">

                <span>¿Cual es tu codigo postal?</span>

                <input type="text" name="cp" id="cp" autocomplete="postal-code" required>

            </label>

            <input type="submit" name="" id="">

        </form>

**Attribute autocomplete:** Complete form with browser information that the user fills out before

**Require:** The user has to fill the form and the website would not submit data before filling all blanks

**Select**

<main>

        <!-- <select name="cursos" id="">

            <option value="javascript">JavaScript</option>

            <option value="HTML5">HTML5</option>

            <option value="CSS3">CSS3</option>

            <option value="Web Standards">Web Stand</option>

        </select> -->

        <input list="cursos">

        <datalist id="cursos">

            <option value="JavaScript"></option>

            <option value="HTML5"></option>

            <option value="CSS3"></option>

            <option value="Web Standards"></option>

        </datalist>

    </main>

**Tag <select>:** Define a list to select and option

**<Datalist>:** Define a list to select an option and allow fast search “more efficent”

**Input type submit vs. Button**

**Input:** use to form

**Button:** use to all things except in forms

**CSS**

Cascading style sheets

**How to use CSS by selector, by class and by ID**

<p>

p{

atribute

}

**<p class=”parrafo”>**

.parrafo{

atribute

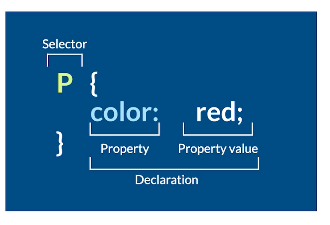
}

**<p id=”texto”>**

.parrafo{

texto

}



[FAQ / Methodology / BEM](https://en.bem.info/methodology/faq/#why-bem) : Describe how to name class elements pseudo-class and pseudo-elements

list-style: none;  /\*Remove dots of the list\*/

Pseudo class 🡺 Define the style of a especial state of an element

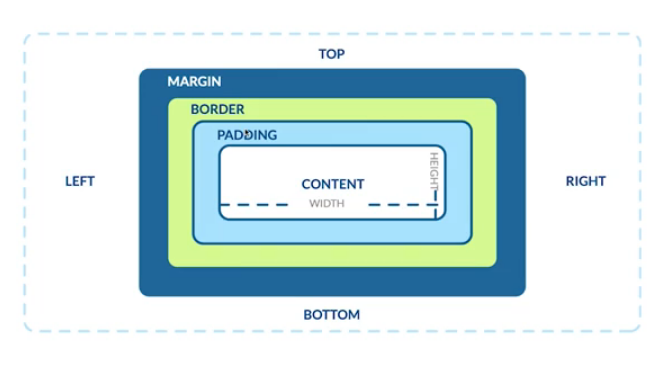
.main\_nav\_\_item a:hover{/\*Change item color when put the mouse in there\*/

Pseudo element 🡺 Define the style of a specific part of an element

.main\_nav\_\_item a:active{ /\*Change item color when press click with the mouse\*/

.main\_nav\_\_item a::after{/\* Insert an object after the item\*/

**BOX MODEL**



\*{ 🡺 Global Selector, Use always

    box-sizing: border-box; /\*Calculate the max-width of the window and  recalculate it in relation to padding and border \*/

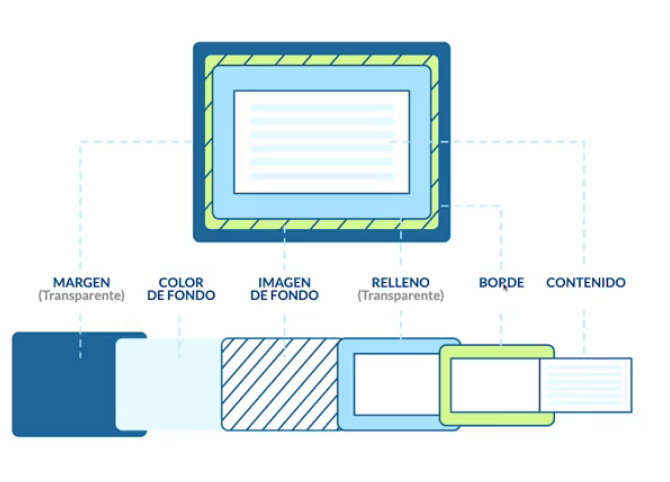
    padding:0;

    margin: 0;

}

**Herencia**

font-size: inherit; 🡺 Take de atribute values of the father class



**Como se controla el orden al declarar CSS**

**Importancia**

**Especificidad**

**Orden de las Fuentes**

**Importancia -> especificidad -> Orden de las Fuentes**

**Importancias:**

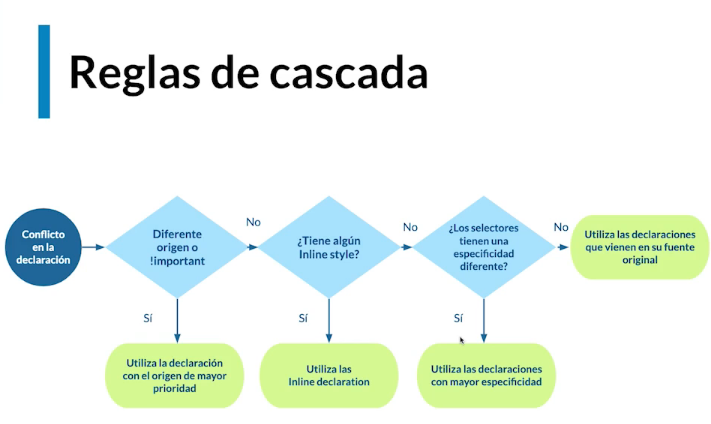
Hoja de estilo de agente de usuario (Estilos del navegador)

Declaraciones normales en hojas de estilo de autor (Nuestro .css)

Declaraciones importantes en hojas de estilos de autor (utilizar el **!important)**

**Especificidad**





**Orden de las Fuentes**

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

NO es buena práctica usar **!important**

NO es buena práctica usar estilos embebidos

NO es buena práctica usar identificadores de elemento excesivamente

**COMBINADORES**

Nos permiten combinar múltiples selectores y crear una mayor especificidad.



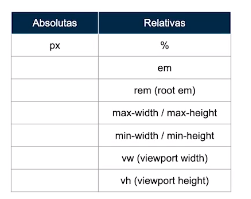
**Hermano adyacente o cercano**

h2+p{/\* Significa  que aplica el estilo a un p que este cerca de un h2\*/

**Hermano general**

<https://flukeout.github.io/> Práctica de combinadores

**Medidas Absolutas vs. Relativas**



**EM:** It depends on parent element

**REM:** Easily scalable in responsiveness, always have as a reference html tag

html{

    font-size: 62.5%;

}

p{

    font-size: 1.6rem;

}

Esto lo que hará es darle al html un valor de 10px ya que 16px - 62.5% = 10px, ahora si por ejemplo a una etiqueta le asignamos 2rem este hará referencia a 20px, o si por ejemplo le damos un valor de 1.5rem su valor será de 15px

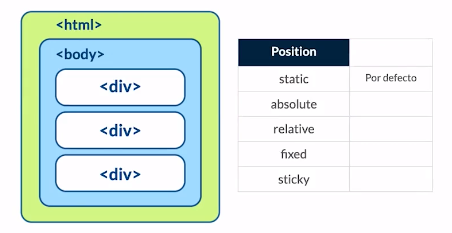
**Max-width / max-height:** maximo valor hasta el cual puede crecer un elemento

**Min-width / min-height:** minimo valor hasta le cual puede decrecer un elemento

**vw :** Ocupa todo el ancho de la ventana

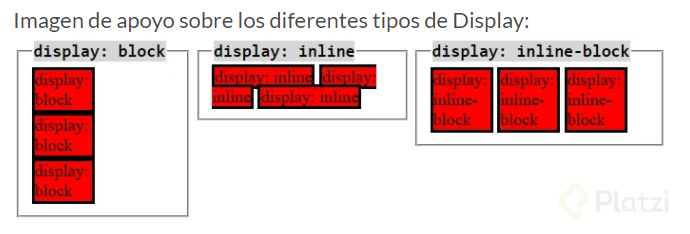
**vh :** Ocupa todo el alto de la ventana

**Position**



**Relative:** no se queda fijo el element, permite utilizar top botton left right

**Display**



**Elemtos en un div:** por default display block

**Span:** pone display inline

**Display: flex;** Permite que todos los elementos del padre puedan estar una despues de otro siempre y cuando haya espacio. Por default siempre pone en fila o tipo row.

**Flex-direction:** Permite establecer la direccion de flex o la distribucion.

**Flex-wrap:** Con base al viewport width permite ir ajustando los tamanos de acuerdo al contenido

**Flex-grow: #;** Permite crecer de acuerdo al numero para rellenar espacio, a mediad que se hace pequeno, se elimina ese relleno hasta llegar a su tamano original.

**Justifiy-content:** Permite alinear el contenido horizontalmente

**Align-items:** Permite alinear el contenido verticalmente

**Order: #;** Permite cambiar el orden de los elementos, primero ordena los que no tiene el atributo order y luego los que si tienen en el orden que especifica su valor.

**Flexbox Layouts**

.container {

border: 0.3rem solid black;

display: flex;

flex-wrap: wrap;

}

.box {

height: 10rem;

flex-basis: 10rem;

flex-grow: 1;

}

**flex-grow** → Toma el espacio que sobra para que no haya espacios en blanco. Al darle el valor 1 toma el espacio sobrante que deja otro elemento.

**flex-basis** → Define el tamaño que tendrá un elemento antes de que se distribuya al tamaño restante.

**align-items: strech**; → Estira los elementos al 100% del valor de su contenedor padre. Ocupa todo el espacio vertical.

**baseline** → Toma solo el alto de su contenido. Lo limita a su propia altura.

**order** → Cambia el orden de los hijos del contenedor padre.

\*Todos los contenedores que no tengan orden se pasan para la izquierda. Y los que sí lo tengan empiezan desde donde terminan los que no tienen.

**VARIABLES**

**:root{}** Para declarar variables, apunta a html

El nombre de la variable debe comenzar con --

--primary-color: blue;

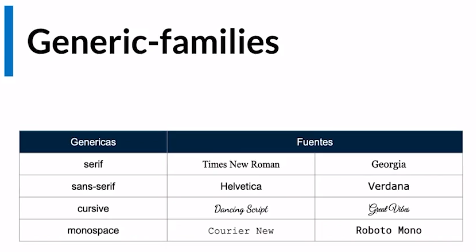
--secundary-color: red;

Para usarlas se usa la function var(nombre\_var) siendo nombre\_var, el nombre que el pusimos a la variable

Sirve para que no se pierdan formatos y fuentes en la pagina y se mantenga estandar

**FONTS**

Fuente genericas ya vienen por default en la PC



- Cambiar la fuenta del navegador en configuracion diseno

- Si no carga la Fuente de la pagina web carga por defecto la que esta configurada en el navegador

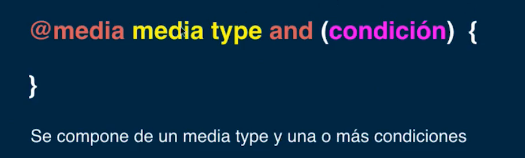
Google fonts -> Para buscar diferentes tipos de letra, afecta la performance

Buena practica; Importar las fuentas y ponerlas dentro de la etiqueta head

Buena practica; Solo cargar un tipo de fuente

display=swap ; signinfica que primero carga todo el contenido de la pagina y luego cambia el tipo de letra

**Media Queries**



**@media screen and (max-width:768px){}:**

Todas las pantallas con un maximo tamano de 768px

**@media screen and (max-width:768px) and (min-width:480px){}:**

Todas las pantallas con un ancho de 480px hasta 768px cumplen con esta condición

**Mobile First**

**min-width = desde**

@media screen and (min-width:320px)

@media screen and (min-width:480px)

@media screen and (min-width:768px)

@media screen and (min-width:1024px)

**Desktop First**

**max-width = hasta**

@media screen and (min-width:1024px)

@media screen and (min-width:768px)

@media screen and (min-width:480px)

@media screen and (min-width:320px)

**CSS Positions**

* **static**: es la propiedad por defecto.

Con las otras opciones, se activan las propiedades de top, bottom, left, right y z-index.

* **relative**: el objeto se mueve en base al lugar donde se encuentra originalmente.
* **absolute**: el objeto se ubica de manera absoluta con el elemento más cercano que tenga posición relativa o con el body.
* **fixed**: El elemento se muestra de manera fija en el viewport.
* **sticky**: El elemento se queda de manera fija una vez que aparece en pantalla.

<https://codepen.io/pen/>

**Página de Iconos:** <https://icomoon.io/>