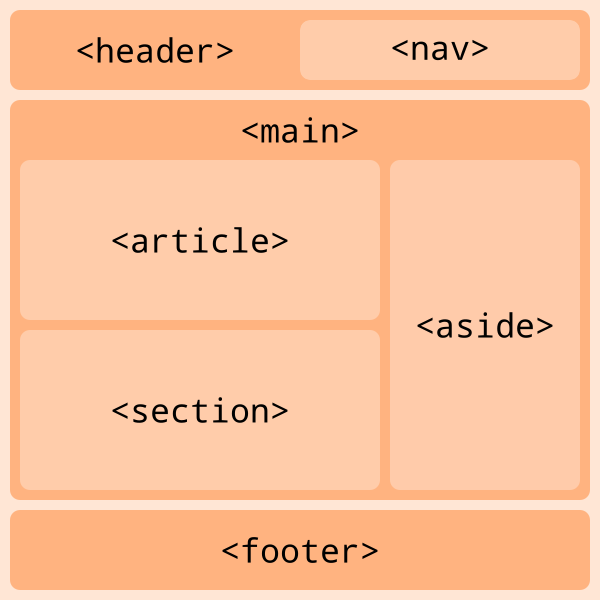
## Estructura de una pagina web



## Tipos de imágenes para la web

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.

Las imágenes png-8 tienen fondo transparente, y las png24 también solo que esta tienen mas colores por ende es mas pesado.

SVG esta imagen no se ve pixelada al ampliar ya que hay un algoritmo que hace que la imagen cresca de acuerdo a la pantalla.

Tamaño máximo recomendado para una imágen:

* **70kb**  
  .  
  Herramientas para optimizar imágenes:
* [**Tiny PNG:**](https://tinypng.com/) Comprime el tamaño de una imagen, para hacerla más ligera.
* [**Verefix:**](https://www.verexif.com/) Elimina los metadatos de una imagen, para reducir su tamaño.

**RECOMENDACIÓN PERSONAL**  
Yo he usado estas páginas y no se logra optimizar a esta capacidad promedio de 70kb, pero usando Photoshop cambiando el ancho de la imagen y guardandolo como una imagen para web, he podido optimizar de 12MB a 92KB, que en mí opinión ha sido mejor que usar algún sitio en internet.



Optimizar nuestro sitio web con las imagenes o iframes, es colocando en la etiqueta img el atributo loading con el valor lazy.

<**img** loading="lazy" src="manzana.png" alt="People">

Clase 13

Una forma de optimizar nuestro sitio web con las imagenes o iframes, es colocando en la etiqueta img el atributo loading con el valor lazy.

<**img** loading="lazy" src="manzana.png" alt="People">

Atributo loading  
El atributo loading permite al navegador retrasar la carga de imáges y de iframes que están fuera de pantalla, hasta que el usuario haga scroll cerca de ellas. Éste atributo soporta 3 valores:

* lazy: Retrasa la carga de la imagen hasta que el usuario alcanza con el scroll una distancia calculada desde el viewport.
* eager: Carga la imagen inmediatamente, sin importar donde está situada o colocada en la pantalla. En resumen, no hace lazy-loading.
* auto: Implementa el comportamiento por defecto del navegador para la carga de las imágenes. En resumen, poner auto es lo mismo que no poner el atributo loading.

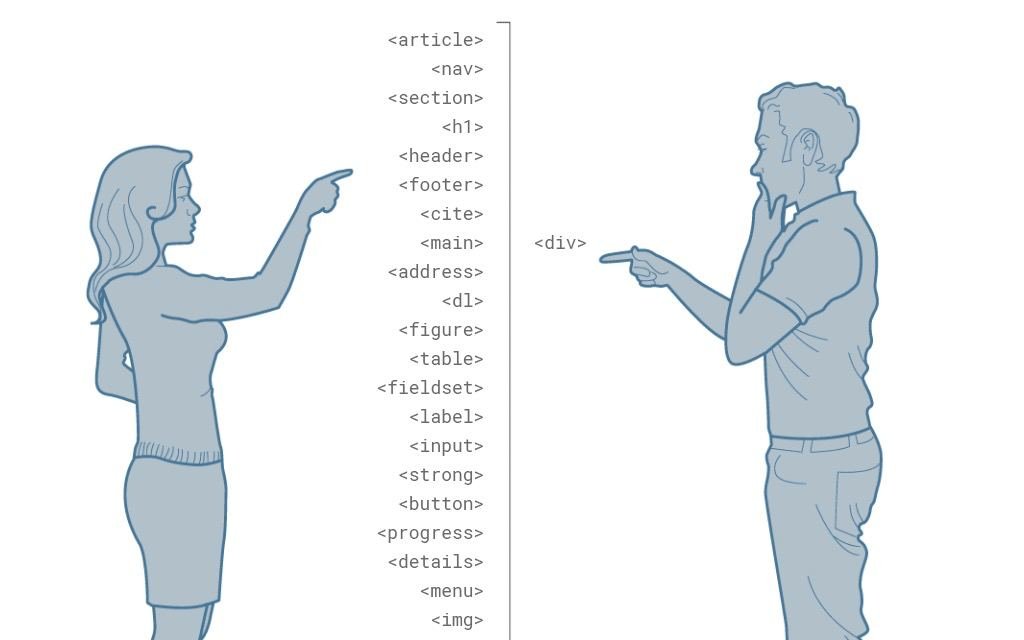
## Clase 14

**Figure** <figure><img /> </figure> es una etiqueta que permite almacenar una imagen en su interior. Es una mejor práctica comparada con usar solamente un contenedor div. Como complemento al contenedor figure, se utiliza la etiqueta figcaption <figcaption></figcaption>, que permite darle una pequeña descripción a la imagen, como el autor, fuente o algo por el estilo, que se mostrará usualmente abajo de la imagen.

Figcaption se diferencia del atributo Alt porque esta última muestra su descripción en texto en el navegador solamente al pasar el mouse por encima de la imagen (de ahí su utilidad para personas con discapacidad visual).

Es importante considerar que la etiqueta figure no es únicamente para imágenes:  
[El elemento HTML <figure>](https://platzi.com/clases/1802-accesibilidad-web/26072-que-es-el-html-semantico-y-por-que-es-importante/) representa contenido independiente, a menudo con un título. Por lo general, se trata de una imagen, una ilustración, un diagrama, un fragmento de código, o un esquema al que se hace referencia en el texto principal, pero que se puede mover a otra página o a un apéndice sin que afecte al flujo principal.

## Ejemplo

clase 15 -etiqueta video

La etiqueta **<video>**, tiene algunos atributos como:  
.

1. **controls:** agrega al video los controles necesarios para reproducir, pausar y adelantar.
2. **preload = auto:** hace que el navegador descargue el video, en el momento en el que se acceda a la página.

.  
La etiqueta **<source>,** se puede colocar dentro de una etiqueta **<video>** varias veces, para especificar diferentes rutas. Esto para asegurar que cualquier navegador pueda mostrar el video.

Estructura

    <main>

        <section>

            <video src="./claseVideo/video.mp4#t=200,260" controls preload="auto"></video>

        </section>

    </main>

src="./claseVideo/video.mp4#t=200,260" al finalizar el src del video se agrega “#t=” para indicar q e video se reproduzca desde cierto minuto y también hasta que minuto finalice en este caso inicia en el minuto -200- y finaliza en el minuto-260-.

Se usa la etiqueta source dentro del tag video, para que se cargue el video con otro formato para que sea compatible con cualquier video.

<main>

        <section>

            <video controls preload="auto">

                <source src="./claseVideo/video.mp4#t=200,260" />

                <source src="./claseVideo/video.mp4#t=200,260" /> <!--video en otro formato-->

                <source src="./claseVideo/video.mp4#t=200,260" /> <!--video en otro formato-->

                <source src="./claseVideo/video.mp4#t=200,260" /><!--video en otro formato-->

            </video>

        </section>

    </main>

El navegador renderizará el primer video q pueda y no renderizará todos. Esto se hace por motivo de compatibilidad con otros navegadores y otros formatos.

Clase 16

Formularios---El mejor formulario es cuando no lo hay.

**Resumen**

**- <Form>; Es una etiqueta fundamentalmente para tener una mejor semántica.**

\*\*- <Label>; Etiqueta que representa la interfaz del usuario.

* for=" "; Permite asociar un control, en el se le dará un nombre personalizado
* **<Input>; Elemento donde el usuario da sus datos para el formulario.**
* type=“text”; Se aclara que los datos serán tipos textos (por ejemplo, nombre y apellido).
* type=“date”; Da una apariencia diferente al input más un calendario donde el usuario podrá poner una fecha mediante los números o facilitándose llenarlo con ayuda del calendario.
* type=“time”; También da una apariencia diferente pero para un horario.
* id=" "; Se le da un nombre (por así decirlo) al elemento para poder asociarlo junto al mismo nombre que el label en el que se le da al atributo de valor de for.
* ***<span>; Es para aplicar estilo al texto o agrupación de elementos. Lo que haría en si es que se colocará arriba o abajo (dependiendo donde lo pongas en el editor de código) donde obtendrá un texto.***

Estructura

<form action="">

    <label for="nombre">

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

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

    </label>

    <label for="inicio-platzi">

        <span>Q dia inició en PLATZI ?</span>

        <input type="date" id="inicio-platzi">

    </label>

    <label for="horario">

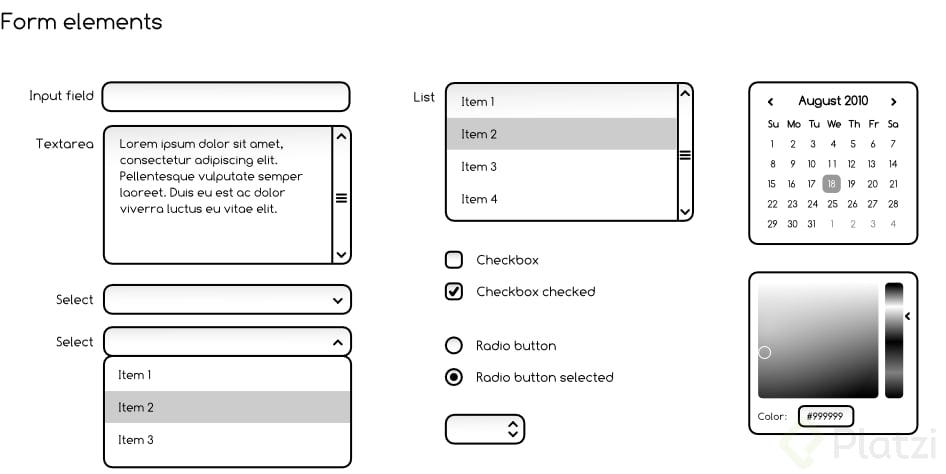
        <span>En que horario estudias ?</span>

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

    </label>

</form>

Tipos de inputs



Como hacer formularios bonitos :

<https://medium.com/@juancaferraris/dise%C3%B1ando-formularios-m%C3%A1s-efectivos-estructura-inputs-labels-y-acciones-81ac011ea05f>

Calendar

 <form action="">

        <label for="hora">

            <span>Hora</span>

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

            ">

        </label>

        <label for="dia">

            <span>Día</span>

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

            ">

        </label>

        <label for="semana">

            <span>Week</span>

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

            ">

        </label>

        <label for="mes">

            <span>Mes</span>

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

            ">

        </label>

        <input type="submit">

    </form>

Se recomienda usar chrome ya que mozilla no renderiza correctamente los inputs month, week.

Calendario completo

<form action="">

    <label for="calendario">

        <span>calendario</span>

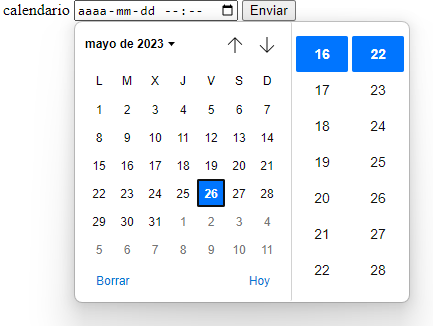
        <input type="datetime-local" id="calendario" name="calendario">

    </label>

    <input type="submit">

</form>

Así mismo se recomienda usarlo en Chrome en donde sale mas completo.



## Tipos de inputs

<https://developer.mozilla.org/es/docs/Web/HTML/Element/input>

## Classe 18-Autocomplete y required

<https://developer.mozilla.org/es/docs/Web/HTML/Atributos/autocomplete>  
Valores autocomplete:

**- “off”:**  
El navegador no puede ingresar o seleccionar automáticamente un valor para este campo. Es posible que el documento o la aplicación proporcione su propia función de autocompletar, o que los problemas de seguridad requieran que el valor del campo no se ingrese automáticamente.

**- “on”:**  
El navegador puede completar automáticamente la entrada. No se proporciona ninguna orientación sobre el tipo de datos que se esperan en el campo, por lo que el navegador puede usar su propio criterio.

**- “name”:**

El campo espera que el valor sea el nombre completo de una persona. Generalmente se prefiere usar “nombre” en lugar de dividir el nombre en sus componentes porque evita tratar con la amplia diversidad de nombres humanos y cómo están estructurados; sin embargo, puede usar los siguientes valores de autocompletar si necesita dividir el nombre en sus componentes:

1. **"honorific-prefix"**  
   Prefijo tipo “Mrs.”, “Mr.”, “Miss”, “Ms.”, “Dr.”, or “Mlle.”.
2. **"given-name"**  
   Primer nombre (first name)
3. **"additional-name"**  
   Segundo nombre (middle name)
4. **"family-name"**  
   Apellido (last name)
5. **"honorific-suffix"**  
   Sufijo tipo “Jr.”, “[B.Sc](http://b.sc/).”, “PhD.”, “MBASW”, or “IV”.
6. **"nickname"**  
   Un apodo.

**- "email"**  
Dirección de correo electrónico

**- “username”:**  
Nombre de usuario

**- “new-password”:**  
Una nueva contraseña. Al crear una nueva cuenta o cambiar contraseñas, este es el campo “Ingrese su nueva contraseña”, a diferencia de cualquier campo “Ingrese su contraseña actual” que pueda estar presente. Esto puede ser utilizado por el navegador tanto para evitar ingresar accidentalmente una contraseña existente como para ofrecer ayuda para crear una contraseña segura.

**- “current-password”:**  
La contraseña actual del usuario.

**-“organization-title”:**  
Un puesto de trabajo, o el título que tiene una persona dentro de una organización, como “Escritor técnico senior”, “Presidente” o “Líder asistente de tropa”.

**- “organization”:**  
El nombre de una empresa u organización, como “Acme Widget Company” o “Girl Scouts of America”.

**- “street-address”:**  
Una dirección postal. Puede ser varias líneas de texto y debe identificar completamente la ubicación de la dirección dentro de su segundo nivel administrativo (generalmente una ciudad o pueblo), pero no debe incluir el nombre de la ciudad, el código postal o el nombre del país.

**- “address-line1”, “address-line2”, “address-line3”:**  
Cada línea individual de la dirección postal. Estos solo deberían estar presentes si la “dirección postal” también está presente.

**- “address-level4”:**  
El nivel administrativo más detallado, en direcciones que tienen cuatro niveles.

**- “address-level3”:**  
El tercer nivel administrativo, en domicilios con al menos tres niveles administrativos.

**- “address-level2”:**  
El segundo nivel administrativo, en domicilios con al menos dos de ellos. En países con dos niveles administrativos, normalmente sería la ciudad, pueblo, aldea u otra localidad en la que se encuentra la dirección.

**- “address-level1”:**  
El primer nivel administrativo en la dirección. Esta suele ser la provincia en la que se encuentra la dirección. En los Estados Unidos, este sería el estado. En Suiza, el cantón. En el Reino Unido, la ciudad postal.

**- “country”:**  
Código de país

**- “country-name”:**  
Nombre del país

**- “postal-code”:**  
Código postal o ZIP

**- “cc-name”:**  
El nombre completo tal como está impreso o asociado con un instrumento de pago, como una tarjeta de crédito. Por lo general, se prefiere utilizar un campo de nombre completo en lugar de dividir el nombre en pedazos.

**- “cc-given-name”:**  
Nombre (first name) en un instrumento de pago como una tarjeta de crédito.

**- “cc-additional-name”:**  
Un segundo nombre (middle name) como se indica en un instrumento de pago o tarjeta de crédito.

**- “cc-family-name”:**  
Un apellido, tal como figura en una tarjeta de crédito.

**- “cc-number”:**  
Un número de tarjeta de crédito u otro número que identifique un método de pago, como un número de cuenta.

**- “cc-exp”:**  
Una fecha de vencimiento del método de pago, generalmente en el formato “MM / YY” o “MM / YYYY”.

**- “cc-exp-month”:**  
El mes en el que vence el método de pago

**- “cc-exp-year”:**  
El año en el que vence el método de pago

**- “cc-csc”:**  
El código de seguridad del instrumento de pago; en las tarjetas de crédito, este es el número de verificación de 3 dígitos que se encuentra en el reverso de la tarjeta.

**- “cc-type”:**  
El tipo de instrumento de pago (como “Visa” o “Master Card”).

**- “transaction-currency”:**  
La moneda en la que se realizará la transacción.

**- “transaction-amount”:**  
La cantidad, dada en la moneda especificada por “transacción-moneda”, de la transacción, para un formulario de pago.

**- “language”:**  
Un idioma preferido, dado como una etiqueta de idioma BCP 47 válida.

**- “bday”:**  
Una fecha de nacimiento, como una fecha completa.

**- “bday-day”:**  
El día del mes de una fecha de nacimiento.

**- “bday-month”:**  
El mes de una fecha de nacimiento.

**- “bday-year”:**  
El año de una fecha de nacimiento.  
\*\*  
**- “sex”:**  
Una identidad de género (como “Mujer”, “Fa’afafine”, “Hombre”), como texto de forma libre sin nuevas líneas

**- “tel”:**  
Un número de teléfono completo, incluido el código del país. Si necesita dividir el número de teléfono en sus componentes, puede usar estos valores para esos campos:

1. **“tel-country-code”:**  
   El código de país, como “1” para Estados Unidos, Canadá y otras áreas de América del Norte y partes del Caribe.
2. **"tel-national"**  
   El número de teléfono completo sin el componente de código de país, incluido un prefijo interno del país. Para el número de teléfono “1-855-555-6502”, el valor de este campo sería “855-555-6502”
3. **“tel-area-code”:**  
   El código de área, con cualquier prefijo interno del país aplicado si corresponde.
4. **“tel-local”:**  
   El número de teléfono sin el código de país o área. Esto se puede dividir en dos partes, para los números de teléfono que tienen un número de intercambio y luego un número dentro del intercambio. Para el número de teléfono “555-6502”, utilice “tel-local-prefix” para “555” y “tel-local-suffix” para “6502”.
5. **“tel-extension”:**  
   Un código de extensión de teléfono dentro del número de teléfono, como un número de habitación o suite en un hotel o una extensión de oficina en una empresa.

**- “impp”:**  
Una URL para un punto final de protocolo de mensajería instantánea, como “xmpp: [username@example.net](mailto:username@example.net)”.

**- “url”:**  
Una URL, como una página de inicio o la dirección del sitio web de la empresa, según corresponda, dado el contexto de los otros campos del formulario.

**- “photo”:**  
La URL de una imagen que representa a la persona, la empresa o la información de contacto proporcionada en los otros campos del formulario.

Ejemplo

<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>En que 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">

</form>

* El atributo **autocomplete = “tipo-del-input”** hace que el navegador auto-complete los formularios, según el tipo de input. El atributo se coloca en la etiqueta **<input>**.
* El atributo **require,** evita que se envíe información del formulario, si el input está vacío. El atributo se coloca en la etiqueta **<input>**.

Tips:

**Con el editor de texto VS Code, contamos con emmet, el cual nos permite insertar el siguiente comando para crear un input con sólo un Enter… copie y pegue 😉**

**form**>**label**>span+**input**

Clase 19-select

Existen dos forma haas la actuaidad para realizar un input del tipo select.

Con el input select e datalist, el **select** permite hacer scroll hasta llegar a la opción necesitada,

*<select name="cursos" id="">*

*<option value="Javascript">Javascript</option>*

*<option value="HTML">HTML</option>*

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

*<option value="WEB STANDARDS">WEB STANDARDS</option>*

*</select>*

mientra que el input del tipo list permite encontrar la opción mientras se teclea un carácter. Ej

      <input *list*="cursos">

        <datalist *id*="cursos">

            <option *value*="Javascript"></option>

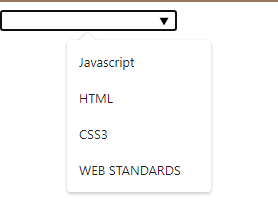
            <option *value*="HTML"></option>

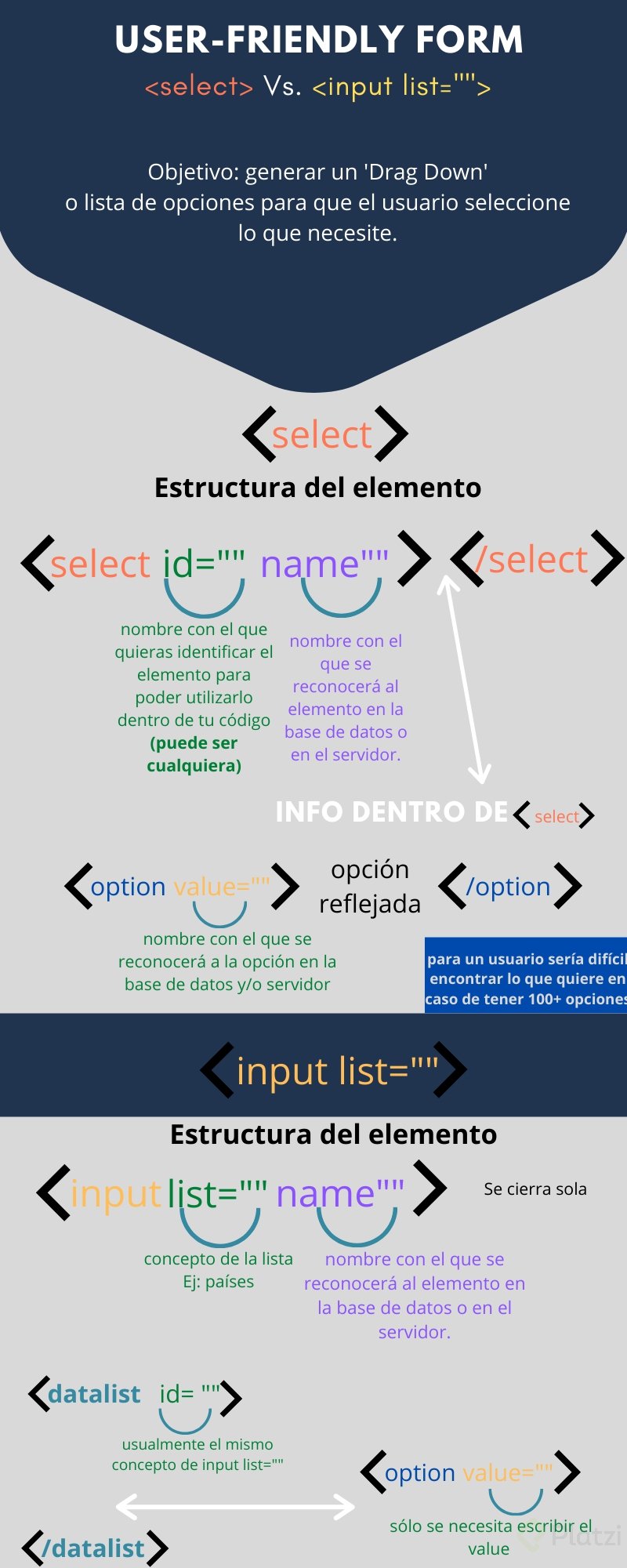
            <option *value*="CSS3"></option>

            <option *value*="WEB STANDARDS"></option>

        </datalist>

En primera instancia se muestra la lista completa, luego al teclear un carácter solo aparecen las coincidencias.





Clase 20 -Input type submit vs. Button tag

Existen dos tipos de botones y los dos se pueden usar para formularios pero se recomienda.

input type submit = lo utilizaremos solo en los formularios

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

    <input *type*="submit" *value*="input-submit"/>

    <button *type*="submit">Que color te gusta</button>

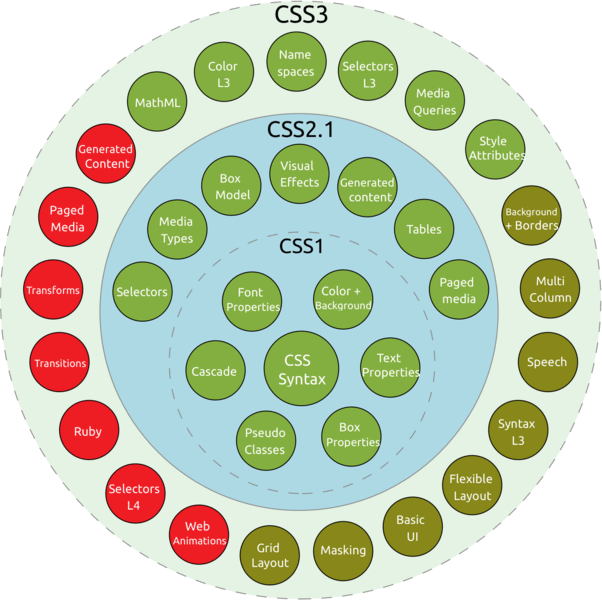
Cabe recalcar que el elemento button se puede personalizar más fácilmente.

Hay 3 tipos de botón:  
Submit - Que envía automaticamente y por defecto los datos de un form  
Reset - Reseta todos los datos dentro de un form  
Button - es solo un botón, no tiene funcionalidad por defecto definida y es totalmente moldeable.

La diferencia es que <button> que puede tener contenido, y la etiqueta <input> no.

Aunque el texto de un botón puede ser especificado en un <input> por el atributo value, tu no puedes agregar texto o contenido personalizado ni semantico en este espacio (como por ejemplo un emoji, un grafico, una imagen), asi que la etqueta <button> tiene un rango de posibilidades mas amplio que <input type=submit>

Clase 21-CSS –evolucino de css



Clase 23-Pseudo clases y pseudo elementos

Metodología BEM-para nombrar clases en css, sirve para generar nombres de clases y hacer que css sea escalable sin morir en el intento ejemplo:

Es nombrar clases siguiendo el siguiente patrón: BLOQUE\_\_ELEMENTO–MODIFICADOR

Por ejemplo:

<header>

        <nav>

            <ul *class*="main\_nav">

                <li *class*="main\_nav\_item"><a *href*="">Home</a></liv>

                <li *class*="main\_nav\_item"><a *href*="">Curses</a></li>

                <li *class*="main\_nav\_item"><a *href*="">Instrutivos</a></li>

                <li *class*="main\_nav\_item"><a *href*="">Blog</a></li>

            </ul>

        </nav>

    </header>

Ejemplos de pseudoclase

a: { nombre de pseudoclase }

.main\_nav\_item a:hover{

    color:blue;

}

.main\_nav\_item a:active{

    color:red;

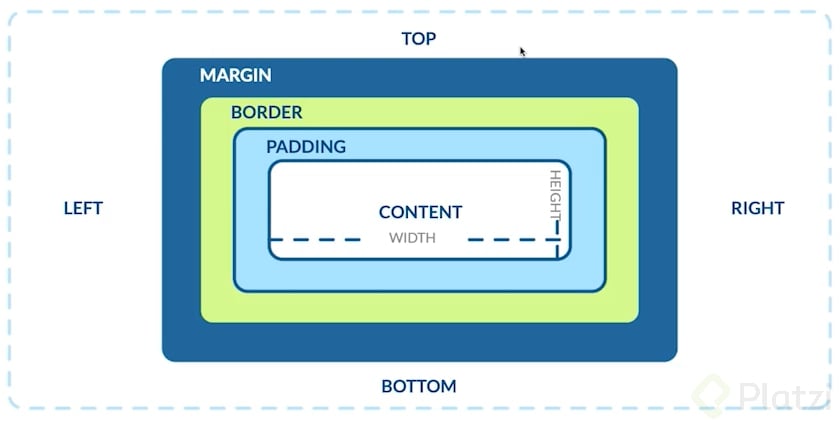
}

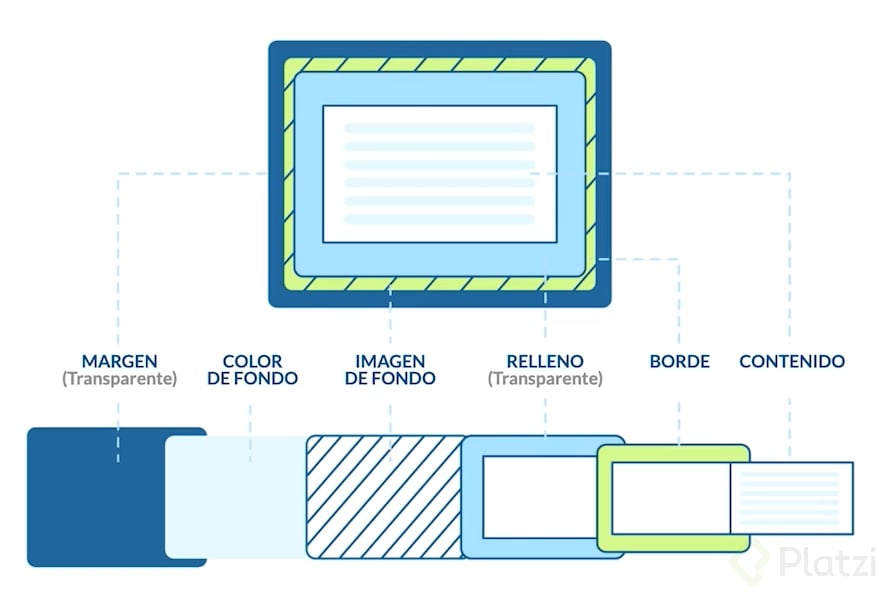
**Anatomía de una regla CSS**

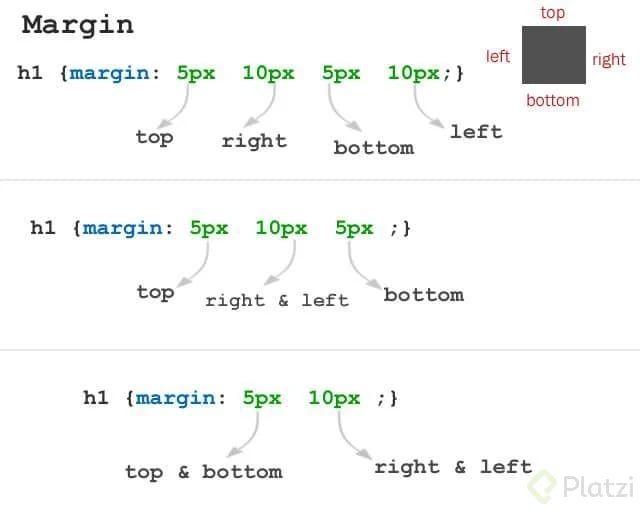


**Modelo de caja**

**Los elementos que renderiza el navegador son cajas, cada elemento del html es una caja.**

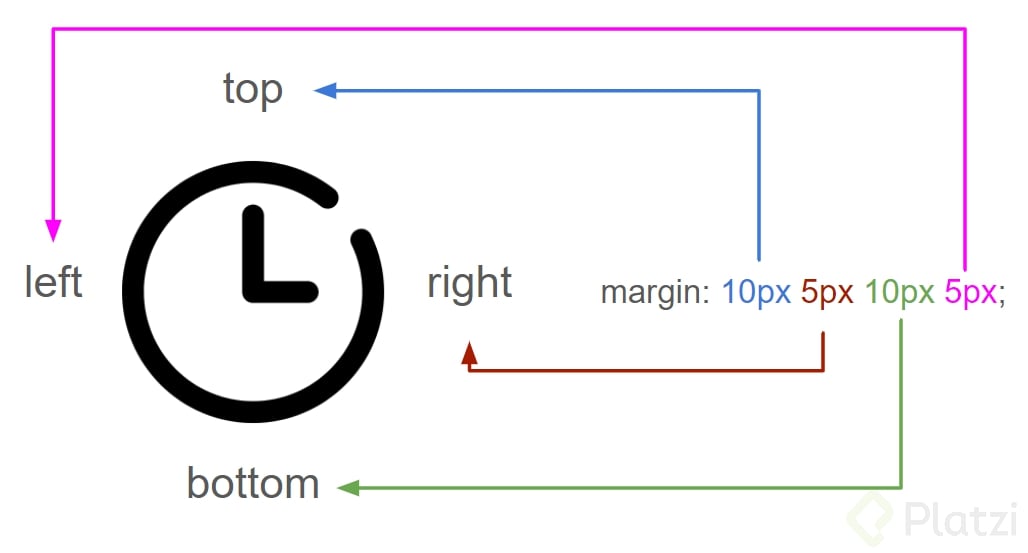






Para usar padding/margin les recomiendo el siguiente truco:  
Piensen en el sentido de las manecillas del reloj empezando desde las 12 y en ese orden va a ir tomando los valores para asignarlos al elemento.

Tener encuenta la propiedad box sizing con el valor border-box, ya que permite al navegador hacer un calculo automatico del width de un elemento para evitar tener un scroll extra ya que cuando colocamos el width en al 100% este solo toma el width del contenido y hace que dentro del modelo de caja ocupe el 100%, pero no considera el border, padding haciendo que se genere un scroll horizontal , el mismo que se quita con la propiedad mencionada anteriormente ya que gracias a la propiedad si se considera automáticamente y se suma al width el border, padding(el margin no se recalcula).



**TIP:** Hay una forma de hacer que CSS calcule el tamaño de un elemento (**width o height, por ejemplo**), restándole cierta cantidad.

Por ejemplo:  
Imagina que quieres colocar **2 cajas dentro de una caja padre** y quieres que cada una tome el **50% de ancho**, pero que cada una tenga un **margen a la izquierda de 10px**. Si colocas width de 50% a cada caja y además le colocas margen, esto hará que las cajas queden una arriba de la otra, porque al agregarle 20px de espacio en márgenes, vas a hacer que ya no ajuste el 50% a cada caja.

Para hacer que ambas cajas sigan tomando el 50% contando los márgenes, puedes hacer lo siguiente:

.caja-hijo

{

width: calc(50% - 20px);

}

Esto hará que el ancho se calcule, tomando en cuenta el 50% y los 20px que mantegan de margen.

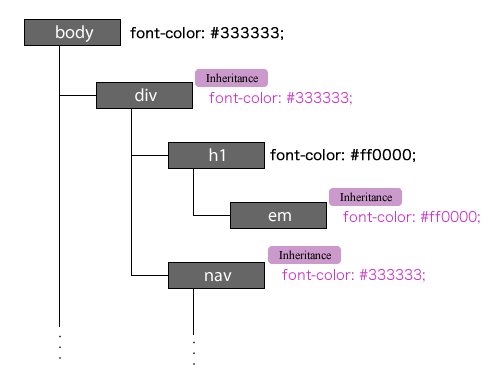
**Herencia**

Es el código css que va a pasar de un padre a un hijo, existen herencias q se pueden romper.

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 tratara de heredar el valor de su elemento padre si este esta 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**



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

**Si dos declaraciones tienen la misma importancia, la especificidad de las reglas decidirá cual se debe aplicar.**

**Si las reglas tienen la misma especificidad, el orden de las fuentes controla el resultado final.**

1. Importancia

1. El navegador carga los estilos por defecto del navegador.

2. Se implementan los estilos .css

3. Se aplican los estilos q tengan el valor !important (no se recomienda usarlo)

1. Especificidad

Estos numeritos son “1.0.0.0.0” 5 digitos separados por un punto de derecha a izquierda menos a mas importante.



Para evitar rescribir estilos y que se rompa nuestro código tenemos que evitar los !important, Iniline styles #id y tag.

1. Orden de las fuentes.

Es la forma en la que nosotros mandamos a llamar los estilos, como css se aplica en cascada las declaraciones que esten al final anularan las declaraciones previas en caso de un conflicto. Es decir se reescriben los estilos.

**Demo de especufIdad y orden en selectores**

Cuando colocamos un Font-family se cargan más nombres de fuentes distintas a la que solicitamos, eso pasa para que en caso de que no haya la primera fuente disponible en este caso “Arial”, se aplica la fuente segunda “Helvetica”, en caso de que haya ninguna de las dos se aplica la tercera “sans-serif” Ejemplo:

#page-title{

    font-family: Arial, Helvetica, sans-serif;

}

El navegador muestra la especificidad tener muy en cuenta este dato:

En este ejemplo se esta modificando el Font family a la misma etiqueta html y se aplica el estilo del id, y luego aparece tachado el estilo de la clase y del tag. Esto lo hace el navegador por especificidad.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Los estilos en línea también se los llama estilos embebidos.

TIP:

\*\*Recuerda! \*\* Si no se cargan los **estilos que tu deseas** es por que hay otro(s) elector(es) con **mayor importancia que el tuyo**

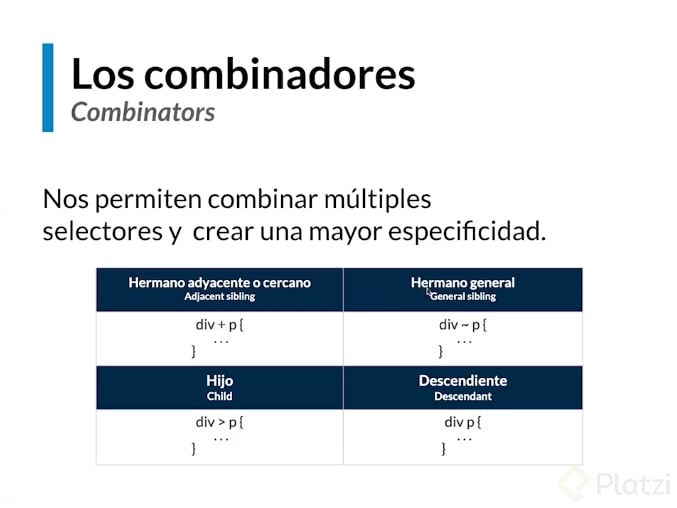
Utilizar los id para estilar no son muy buena practica.

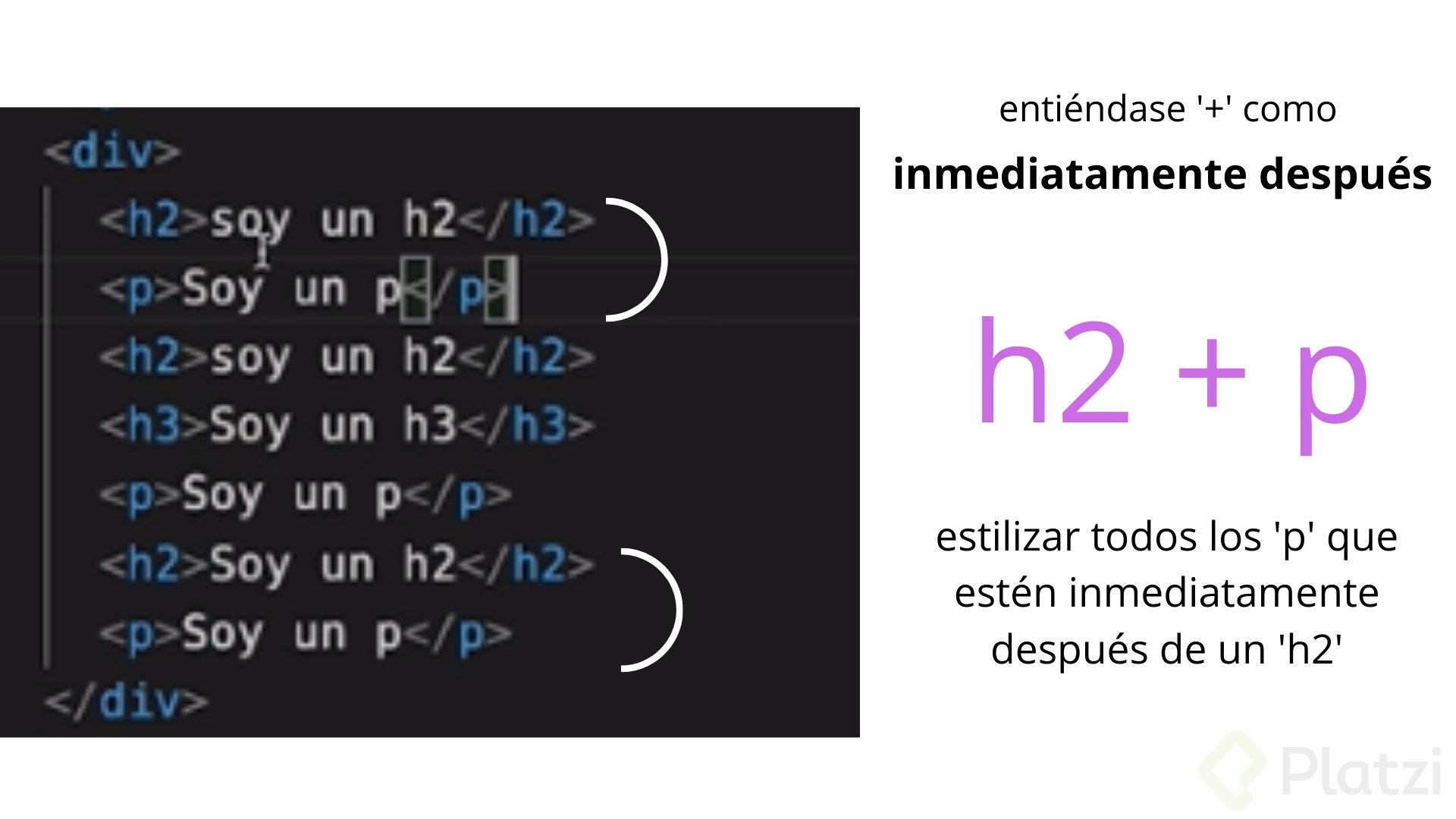
**Malas Prácticas dichas en Clase Hasta Ahora**

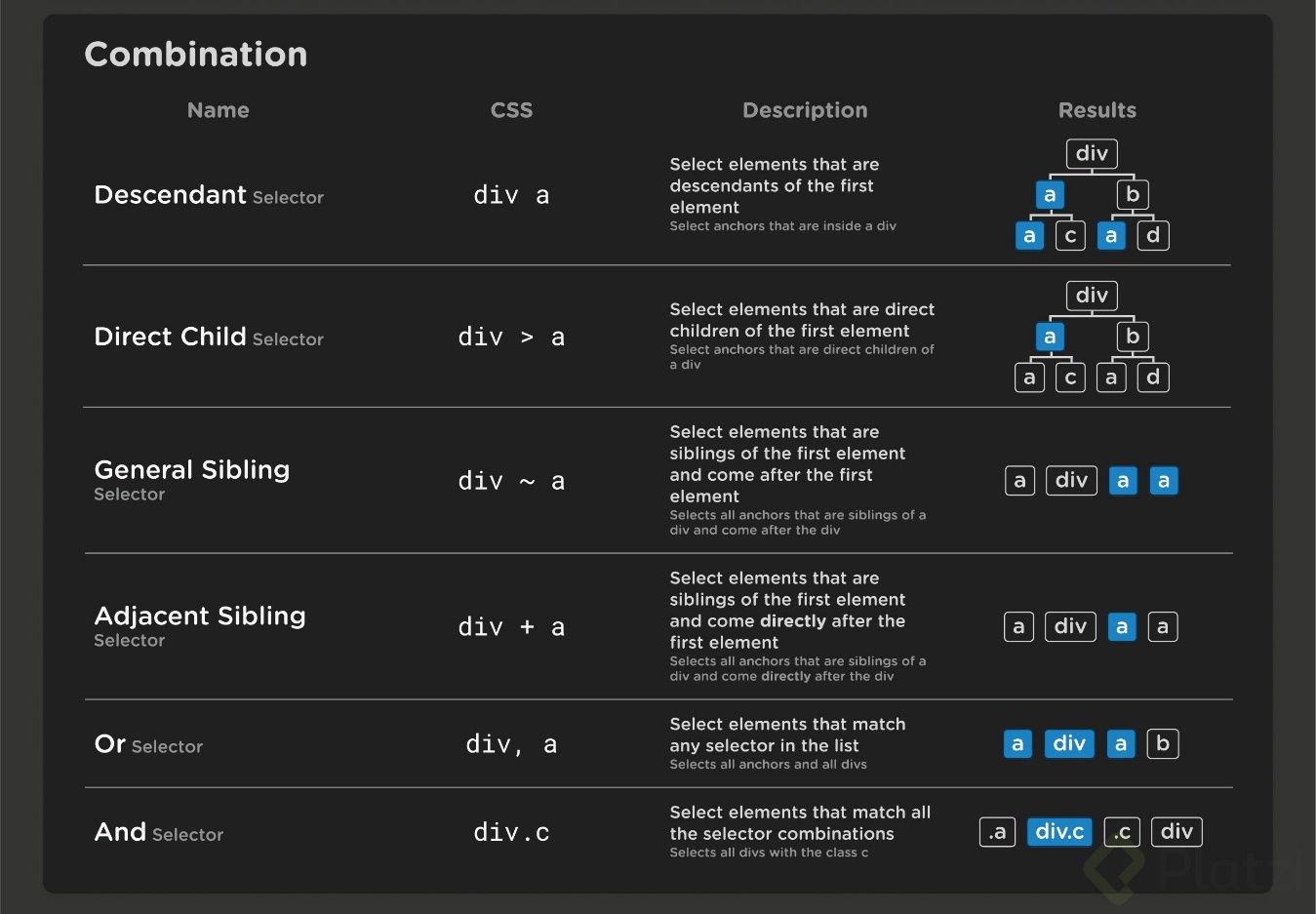
* Utilizar tanto **id** en CSS
* Utilizar el **!important**
* Utilizar la etiqueta **<style>** dentro del archivo html
* Utilizar el atributo style dentro de las etiquetas html
* Utilizar div para contener todo ignorando los header, nav, section, article, etc.
* No utilizar la etiqueta **<form>** para hacer formularios
* Utilizar las etiquetas <select> y <option> para hacer selectores o menús desplegables.
* No nombrar el primer archivo html del proyecto como index.html
* No tener archivos .css para cada pantalla de un proyecto.
* Tener todo el css junto en un solo archivo.
* No ponerle el atributo alt a una imagen
* Poner imágenes dentro de <div> en vez de **<figure>**
* Utilizar textos solo en mayúscula en HTML, en vez de utilizar el atributo de CSS, text-transform, con el valor uppercase. Ya que al hacer esto pareciera que estuvieras gritando.
* Poner videos que se reproduzcan solos.
* No optimizar las imágenes.
* No tener cuidado de cual es el formato ideal para las imágenes y su respectivo peso.
* No tener cuidado con la respectiva semántica de HTML, y con la sintaxis adecuada para CSS.
* No cerrar las etiquetas que se cierran en sí mismas como <br/>
* No comentar partes esenciales de tu código.
* No poner la etiqueta **<meta name=”robots” content=”index,follow”>** en tu proyecto para que los navegadores los puedan ubicar mejor.
* No usar la etiqueta **<meta name=”viewpor” content=”width=device-width, initial-scale=1.0”>** para hacer tu proyecto responsive.
* No poner el atributo **autocomplete=”valor”** en los campos de tu formulario para hacerle la vida más fácil al usuario
* No usar el atributo **required** en los campos obligatorios de tu formulario como una primera capa de seguridad

**Combinadores en css (combinators)**

Esto evita el sobre uso de clases y id a la hora de estilizar







**Adjacent sibling (hermano adyacente)**

**Hermano cercano, nos permite estilizar un elemento que este junto a otro q definamos nosotros. Ej**

**Texto

Descripción generada automáticamente con confianza media**

**En este caso me permite cambiar de color a la etiqueta p que cumple la condición de hermano adyacente.**

h2 + p{

    color: red;

}

**General sibling**

Le aplica el estilo siempre y cuando existan como hermanos sean o no cercanos.

h2~p{

    color: red;

}

**Texto

Descripción generada automáticamente con confianza baja**

**Direct child**

div>p, es decir q se va aestilizar al hijo directo p que tiene como padre un elemento div.

Ejemplo:

  <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>

div>p{

    color: red;

}

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

**Descendant**

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>

div p{

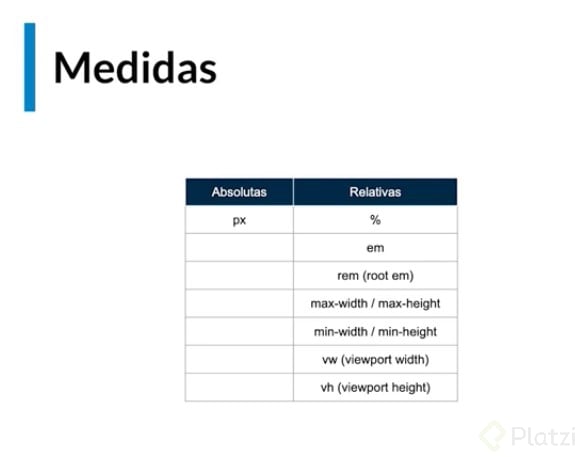
    color: red;

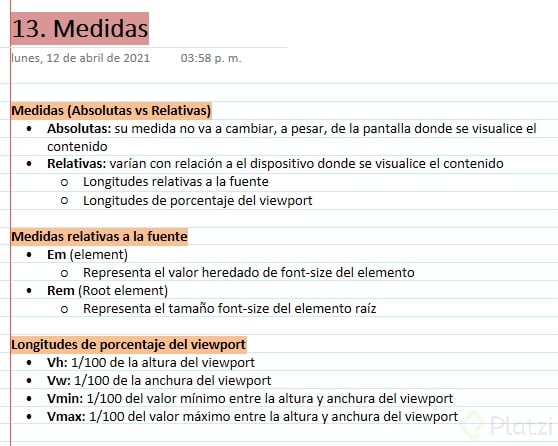
}

Diagrama

Descripción generada automáticamente con confianza media

**Medidas**





**Medidas em**

## em: va a tomar el tamaño de fuente que tenga el padre directo, o en su defecto, a su ancestro más próximo que tenga un tamaño declarado.

**1em =16px**

**Em toma como referencia el tamaño del padre directo.**

Una utilidad que se le suele dar a em (Cuando la aplicas bien) es la de escalar una página web, es decir, simplemente cambiando el tamaño de fuente de un padre general, puedes escalar todo un sitio web con base en dicha medida y cambiando solo un valor ^^

Hay que tener cuidado cuando se anidan los elementos ya que em toma la medida del padre directo, y al anidar este ira escalándose o incrementando.

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**

**Medidas rem**

Rem siempre va a tener referencia al estilo q tenga la etiqueta root en este caso siempre va a ser html, q el navegador le agrega q es 16px. De donde 1rem=16px.

Existe un truco para facilitar el calculo del rem, q es establecer un Font-size en el html en 62.5% q quiere decir que de ahora en adelante 1rem será igual a 10px, para poder hacer calculos mas rapidos a la hora de establecer dimensiones. De donde al decir que el Font-size de la etiqueta p es 1.6 rem por 10px q equivale un rem viene a dar q la etiqueta p va a tener un Font-size de 16px.

html{

    font-size: 62.5%;

}

p{

    font-size: 1.6rem;

}

**Estructura definitiva de un archivo css ,usar siempre las medidas rem seteadas previamente.**

\*{

    box-sizing: border-box;

    margin: 0;

    padding: 0;

}

html{

    font-size: 62.5%;

}

**Max/Min width**

Cuando vayamos a utilizar el min y el max en el width debemos tener un width base casi siempre en %.

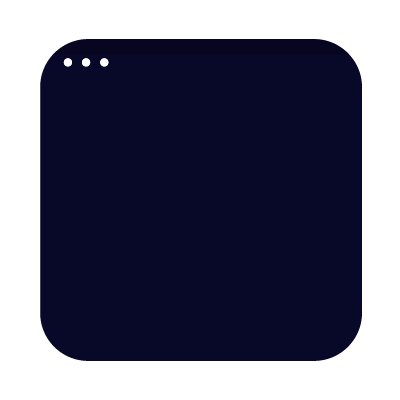
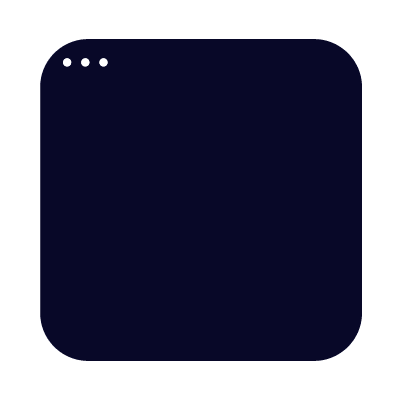
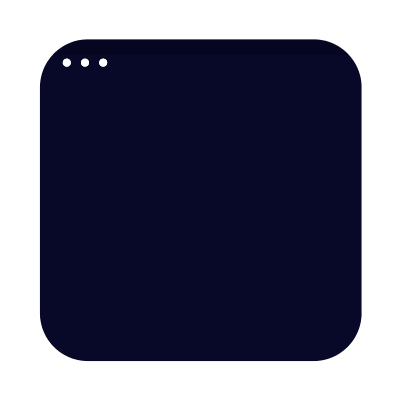
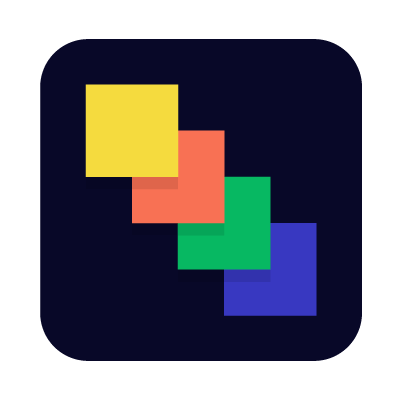
**Position**

Todas las etiquetas tienen un posicionamiento static por defecto y no se puede usar el posicionamiento botón, up, left y right.

Interfaz de usuario gráfica

Descripción generada automáticamente

**Valores de position**  
**Static:** Posición por defecto de los elementos, conservan la posición y espacio de donde son colocados (estáticos). No se puede usar top, right, bottom y left en esta posición.  
**Absolute:** Permanecen en la posición de donde fueron colocados pero pierden su espacio físico (se sobreponen a los elementos que ocupan dicho espacio), se los puede posicionar mediante las propiedades top, right, bottom y left.  
Importante: Al aplicar las propiedades top, right, bottom y left se tomará de referencia al contenedor más cercano con posición relativa.  
**Relative:** Conservan su posición original y espacio físico pero se los puede posicionar mediante las propiedades top, right, bottom y left sin perder dicho espacio físico.  
**Fixed:** Pierden su espacio físico y permanecen de forma fija (siguen el scroll, se colocan al lado izquierdo del viewport), se los puede posicionar mediante las propiedades de top, right, bottom y left.  
**Sticky:** Conservan su espacio físico pero cuando el scroll los alcanza lo siguen (sin perder dicho espacio físico), es muy usado para barras de navegación y se lo puede posicionar con las propiedades top, right, bottom y left.

**Display**

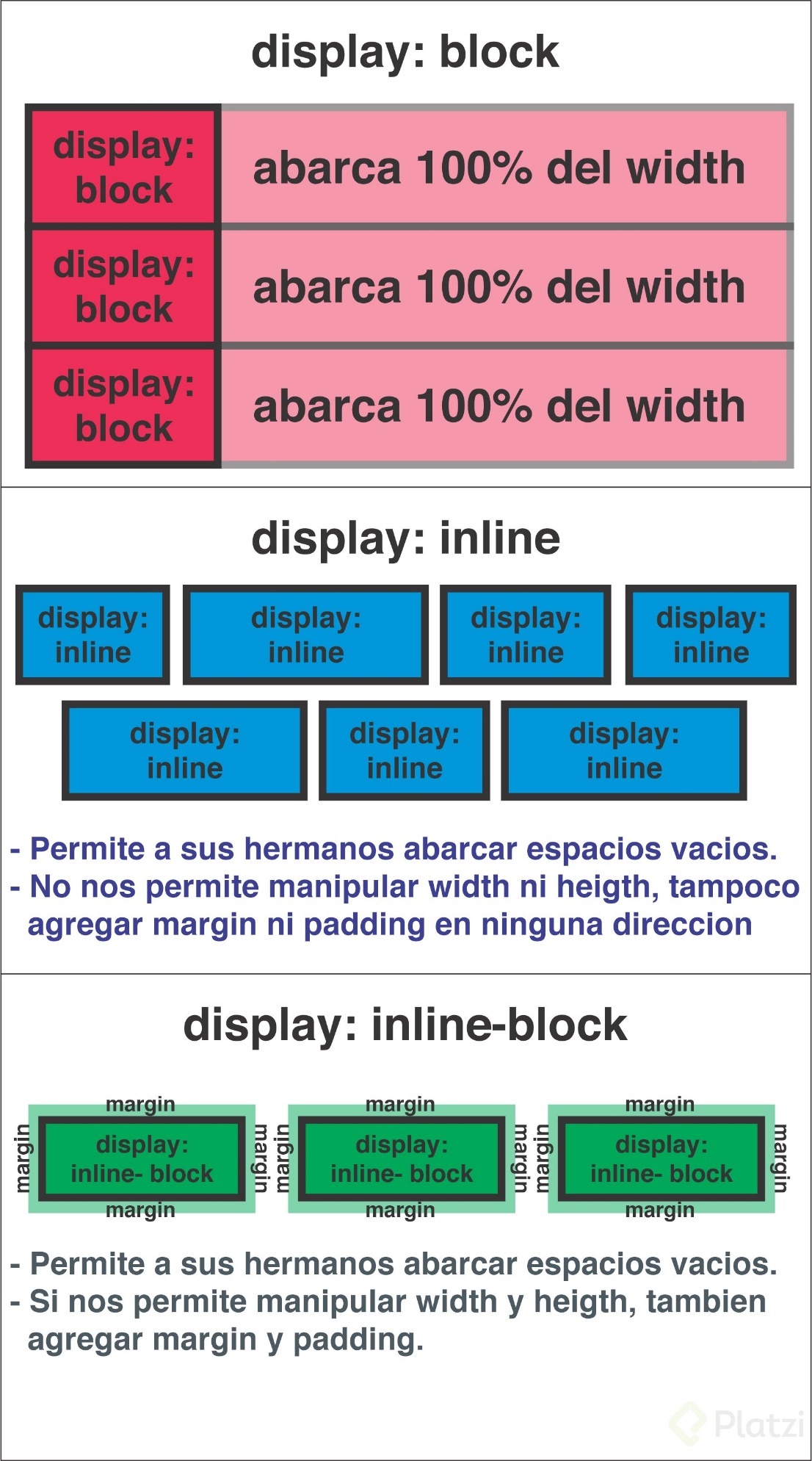
Texto

Descripción generada automáticamente

Ej de display:block en donde no importa el contenido el contenedor ocupara el 100% del width

Interfaz de usuario gráfica

Descripción generada automáticamente



En esta clase vimos los 3 tipos de display Base.  
Los cuales fueron:

**Block**: Estos toman el 100% del width, por lo que un elemento no puede posicionarse a un lado de el.  
Se le puede poner el width deseado, height deseado, añadir margin, padding sin problema. Pero recordando que ocupara este elemento todo el largo de una Fila por asi decirlo.

**Inline**: Estos elementos solo ocuparan el ancho dependiento de su contenido. Por lo tanto estos elementos si permiten que si un elemento cabe a lado suyo, se posicione este ahi.  
Las \*desventajas es que no se les puede modificar el width, height, ni colocar margin u padding tanto top, como bottom.

**inline-block**: Este tiene la combinación de los 2 anteriores. Haciéndolo un mejor candidato para usarlo.  
Permite modificar su width, height, añadirle margin, padding sin problemas y lo mejor es que mientras que haya espacio a un lado suyo, este permitirá posicionar mas elementos ahi.

Y además de estos existen muchos otros tipos de display, de los cuales hay dos muy interesantes, que son el display: flex; y el display: grid; Son práctiacamente lo mejor que le ha podido pasar a CSS porque hace que el posicionamiento de cajas sea extremadamente fácil