

Todo sobre CSS

```
/* Como definir un id */
#titulo{

}
/* como definir una clase */
.centrar{

}
```

Como usar los atributos definidos

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Universidad CSS</title>
  <link rel="stylesheet" href="/css/estilos.css">
</head>
<body>
  <h1 id="titulo">Universidad CSS</h1>
  <p class="centrar">Iniciando el curso de css</p>
  <p class="centrar">Pagina web del curso</p>
</body>
</html>
```

Como implementar mas de una clase a una misma etiqueta

```
✓ .textoGrande{
  font-size: 20px;
}

<p class="centrar textoGrande">Iniciando el curso de css</p>
```

Selector universal

```
/* Selector universal */
*{
    background-color: ■ ghostwhite;
    color: ■ cornflowerblue;
}
```

Hay que tener siempre en cuenta si hay un atributo así:

```
#titulo{
    color: ■ red;
    text-align: center;
}
```

Este se volverá a sobrescribir y los que no tengan este atributo se comportaran como el selector

Agrupar selectores: Usar elementos de una sola vez

```
<h1>Universidad CSS</h1>
<h2>CSS - Cascading Style Sheets</h2>
<p>Iniciando el curso de css</p>
<p>Pagina web del curso</p>
<div>Nuevo texto</div>
```

```
h1, h2, p, div{
    color: ■ crimson;
    text-align: center;
}
```

SubClases

```
<p class="centrar">Iniciando el curso de css</p>
<p>Pagina web del curso</p>
<div class="centrar">Nuevo texto</div>
```

```
p.centrar{
    text-align: center;
}
```

Solo se va a aplicar la clase a la cual sea un párrafo (<p>) y contenga el atributo class="centrar"

Definir colores para bordes y sus tipos:

Como le definimos:

Border: (tamaño) (estilo de borde) (color);

Entonces podemos decir que

Border: 10px solid red;

Border: 10px dashed Pink

Definiendo tipos de bordes:

Solid: línea solida

Dashe: Línea punteada

Double: Doble línea

Dotted: Línea punteada con círculos. Hay mas tipo de bordes

Groove: Con surcos { Estos son parecidos}

Ridge: borde con cresta { “ ” }

Inset o outset: Cumplen la misma función pero inversa

None: sin borde

Hidden: borde oculto pero en esto varia cuando lo utilices al elemento

Si ponemos uno al lado del otro hace bordes mixto. Para cada lado

Definiendo ancho de bordes:

Utilizando width

Tamaños: thin, medium, thick

Usando border-width puedo definirlo por pixeles también

Border-width: 10px 5px 3px 2px; en sentido de las agujas del reloj

Si ponemos menos entr 1 y 3. Los valores se aplican para su inversa también por ejemplo arriba y abajo se aplica uno izquierda y derecha los otros en caso de que sea 10px 5px

Colores en css:

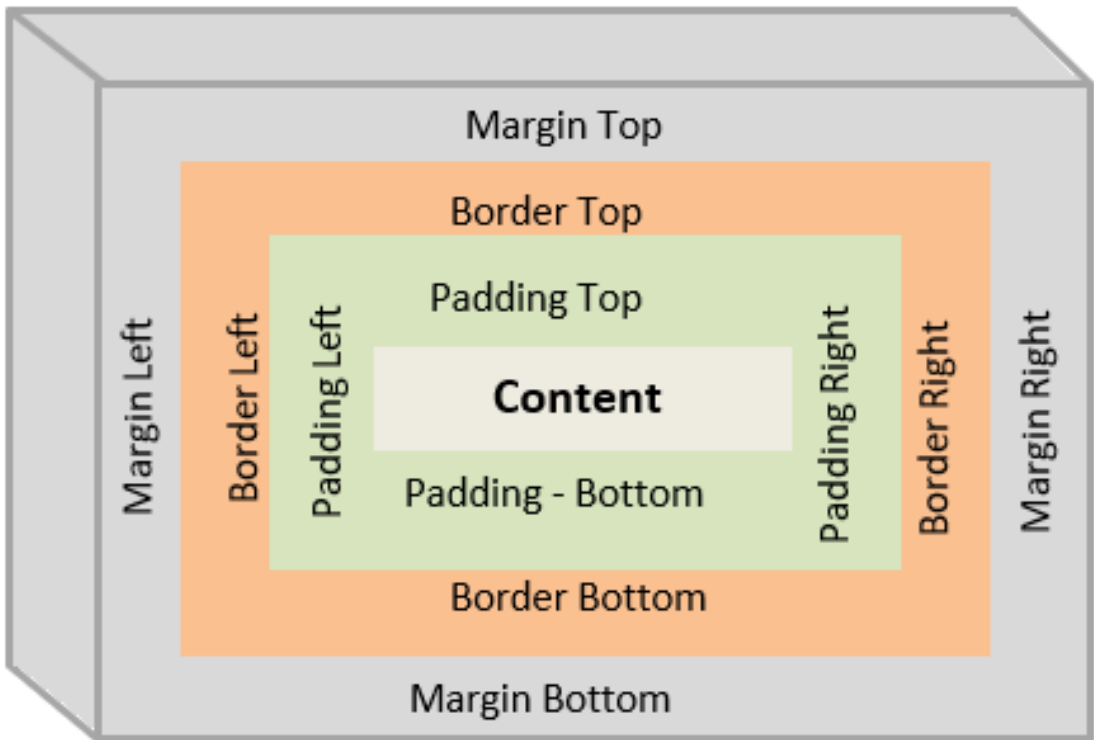
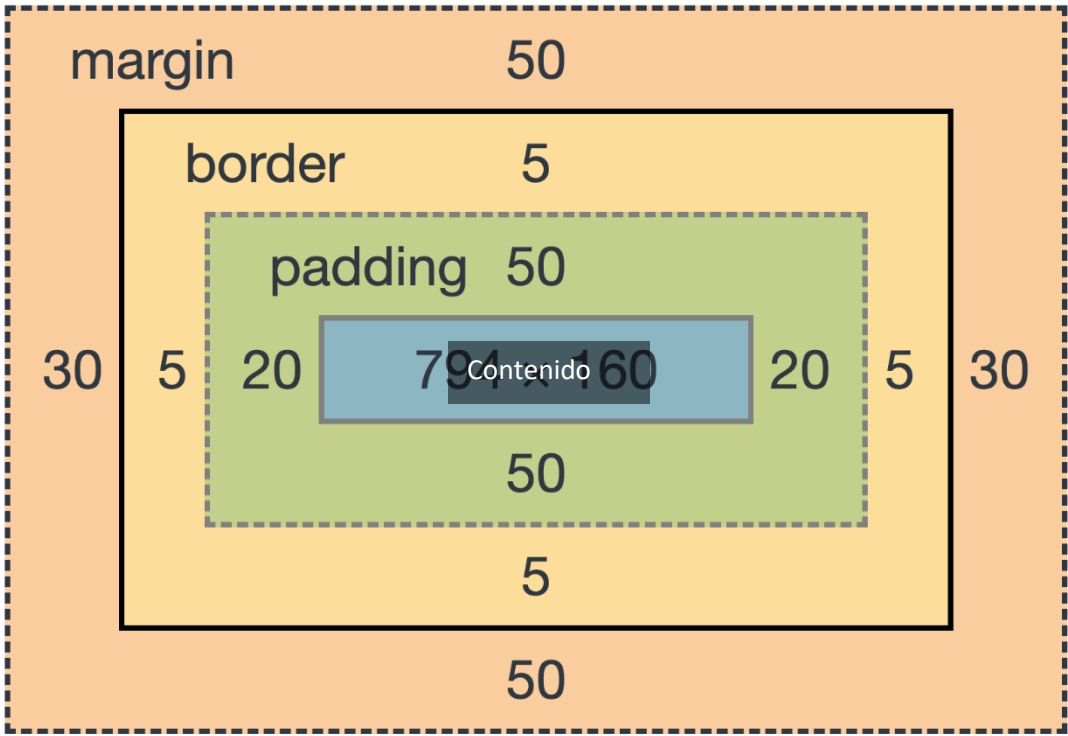
Se utiliza border-color: puedo aplicar para cada lado un color especifico

Bordes redondeados:

Se utiliza border-radius y se especifican los pixeles y los lados como por ejemplo:

Border-radius: 10px 0 10px 0;

Para modelar el contenido dentro de la pagina. Box-sizing: border-box; Esto ajusta toda la caja por generarla



Max-width: 100%;

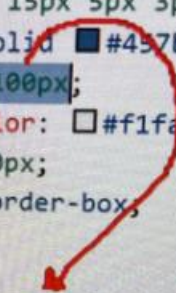
Esto me sirve para regular el contenido del texto. Dentro de un Contenido

Concepto de la propiedad

Inherit.

Esto sirve para especificar el elemento padre por ejemplo un div. Que es padre del elemento hijo que puede ser un párrafo o sea un <p>

```
h1{
  text-align: center;
  color: #e63946;
}
div{
  padding: 10px 15px 5px 3px;
  border: 1px solid #437b9d;
  margin-left: 100px;
  background-color: #f1faee;
  max-width: 300px;
  box-sizing: border-box;
}
p.interno{
  margin-left: inherit;
}
```

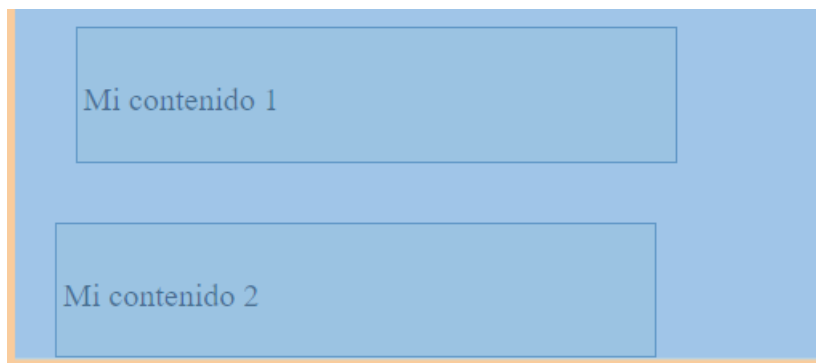
A red arrow originates from the 'margin-left: 100px;' property in the 'div' selector and points to the 'margin-left: inherit;' property in the 'p.interno' selector, illustrating the concept of inheritance.

Siempre y cuando div sea padre de p

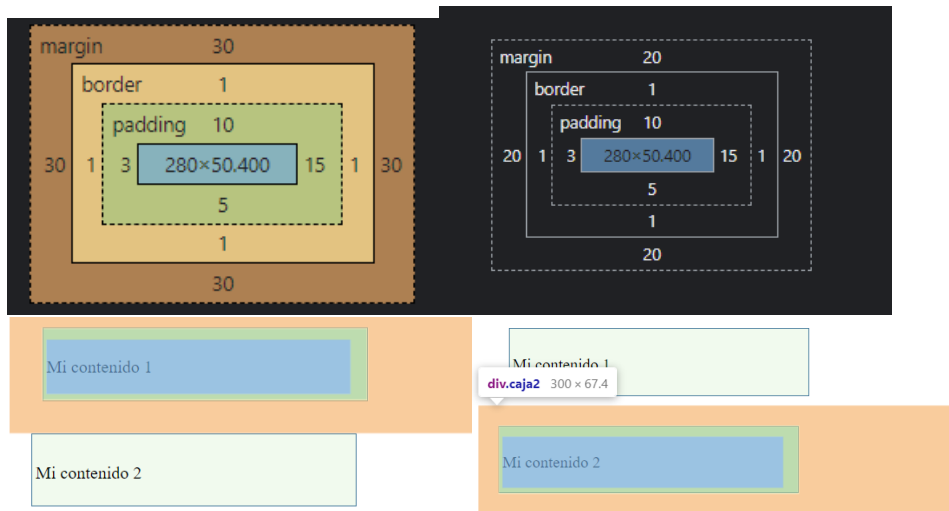
Margin Collapse

Esto solo ocurre con los márgenes superiores o márgenes inferiores Se chocan entre ellos y se fusionan

Entonces:



Como podemos ver uno desplaza al otro y encima lo fusiona



Como podemos ver uno es pisado por el otro y desplazado también

Poniendo el margin de los dos en 20 esto solucionaria el problema

Manejo de texto:

```

h1{
  color: #e452d3;
  background-color: #e8f97c;
  text-align: center;
  text-transform: uppercase;
  text-shadow: 2px 2px 2px #ced4da;
  letter-spacing: 7px;
  word-spacing: 10px;
  text-decoration: overline underline;
}

.caja{
  padding: 15px;
  margin: auto;
  width: 50%;
  background-color: violet;
  border-radius: 5px;
}

.texto{
  text-align: justify;
  /* direction: ltr; */
  text-indent: 30px;
  /* identa el texto */
  line-height: 1.5;
  white-space: normal;
  text-shadow: 1px 1px 2px #ced4da;
}

```

Manejo de Fuentes:

El nombre de la fuente se define como
Font-family: (Se especifican las fuentes)

Podemos utilizar Google Fonts

Esto puede tardar mas al ejecutar la pagina. Pero tambien se puede descargar para evitar esto

Se pega al final de head y se agregan los link

Tambien se puede importar en la hoja de estilos utilizando el import

Luego es la hoja de estilos nos indica como utilizarlo.

Usando el formato Font-family

Tambien se pueden agregar efectos a Google fonts

```
.fuente_roboto{  
  /*  
    font-family: 'Roboto', sans-serif;  
    font-style: normal;  
    font-weight: normal;  
    font-variant: small-caps;  
    font-size: 20px;  
    /*font: font-style font-variant font-weight *font-size/line-height *font-family;*/  
    font: italic small-caps bolder 20px Roboto, sans-serif;
```

Formas de resumir todos los Font. Resumir fuentes

Bootstrap:

```
h1{
  text-align: center;
  color: ■ crimson;
}

div{
  padding: 15px;
  margin: auto;
  max-width: max-content;
}

.icono_azul{
  font-size: 2.5em;
  color: ■ cornflowerblue;
}

.icono_flecha_roja{
  font-size: 3em;
  color: ■ red;
}
```

```
<title>Iconos en CSS</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.8.1/font/bootstrap-icons.css">
<link rel="stylesheet" href="/css/estilos.css">
</head>
<body>
  <h1>Iconos de Bootstrap</h1>
  <div>
    <p>
      <!-- i de icon -->
      <!-- defino la clase y luego busco por BI -->
      <!-- <i class="bi-alarm"></i> -->
      <i class="bi-alarm"></i>
      Alarma
    </p>
  </div>
  <div>
    <p>
      <i class="bi-alarm icono_azul"></i>
      Alarma Azul
    </p>
  </div>
  <div>
    <i class="bi bi-arrow-down-right-circle-fill icono_flecha_roja"></i>
    Flecha Roja
  </div>
</body>
</html>
```

Como utilizar los iconos Podemos ver que necesitamos de usar de mas de una clase una que pertenece al icono y otra que pertenece al atributo de estilo

Lugar de los iconos de Bootstrap: <https://icons.getbootstrap.com/#install>

Ahora vamos a ver iconos de Google: <https://fonts.google.com/icons>

Guia de uso: https://developers.google.com/fonts/docs/material_symbols

En este caso los iconos se ponen como parte del valor de este elemento

```
<body>
  <h1>
    <p>
      <i class="material-icons ">home</i>
      Iconos de Google
    </p>
  </h1>
  <div>
    <p>
      <!-- se escribe como parte del elemento -->
      <!--
      <span class="material-icons-outlined">search</span>
      -->
      <i class="material-icons">search</i>
      Busqueda
    </p>
  </div>
  <div>
    <p>
      <i class="material-icons listo">done</i>
      Listo
    </p>
  </div>
  <div>
    <p>
      <i class="material-icons flecha">arrow_right_alt</i>
      Flecha
    </p>
  </div>
```

```
.listo{
  /* ya que de otro modo la imagen se veria feo */
  vertical-align: bottom;
  color: blue;
  font-size: 2.5em;
}
.flecha{
  vertical-align: bottom;
  color: red;
  font-size: 2.5em;
}
```

Ionicons

Pagina de referencia: <https://ionic.io/ionicons>

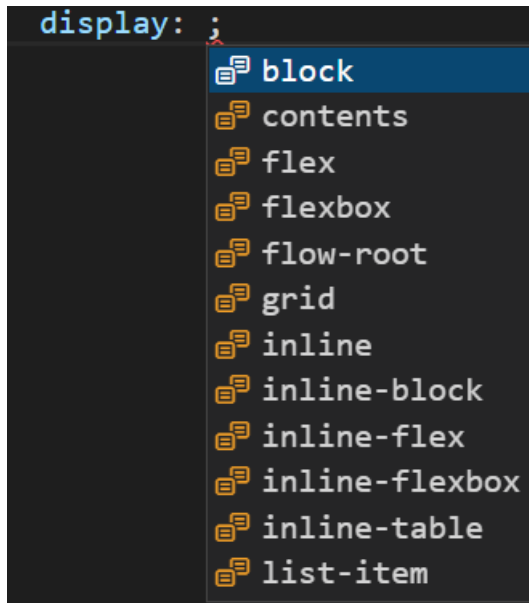
```
<body>
  <h1>
    <ion-icon name="home-outline"></ion-icon>
    Iconos de inioc
  </h1>
  <div>
    <p>
      <ion-icon name="search-outline"></ion-icon>
      busqueda
    </p>
  </div>
  <div>
    <p>
      <ion-icon name="checkmark-done-outline" class="listo"></ion-icon>
      Listo
    </p>
  </div>

  <script type="module" src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
  <script nomodule src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
</body>
```

Asi se utiliza y como podemos ver son un nombre y no una clase por lo que para editarlos debemos usar el atributo class

```
.listo{
  color: ■ aqua;
  vertical-align: auto;
  font-size: 2.5em;
}
```

Manejo de display: Con el vamos a poder trabajar con los tipos de bloques pudiendo así modificar lo que allí dentro se encuentra. Por ejemplo si es una lista pasarla a bloque y al revés igual. Para eso usamos la propiedad display



Algunos atributos tienen block por defecto y otros tienen inline. Nosotros vamos a poder variar esto según sean nuestros requerimientos

Centrar un div:

A menudo necesitamos poder administrar los bloques que ponemos para crear nuestra página, es por ello que como un div es capaz de ocupar todo el bloque de la página haciendo un width del 50% y luego seteando el margin en auto. Vamos a poder administrarlo como nosotros deseemos

```
width: 50%;  
border: 3px solid firebrick;  
margin: auto;
```

Como podemos ver, seteamos la anchura con width y luego el margen que se auto acomode según el cuadro

Tipos de posicionamientos:

Tenemos:

- Static
- Relative
- Fixed (ajustado) Elemento que toma un valor fijo en la pantalla
- Absolute
- Sticky (pegajoso)

El posicionamiento sin posición es un atributo static

Position: "static";

Luego si esto es static esto no se puede mover

Pero si es relative entonces si.

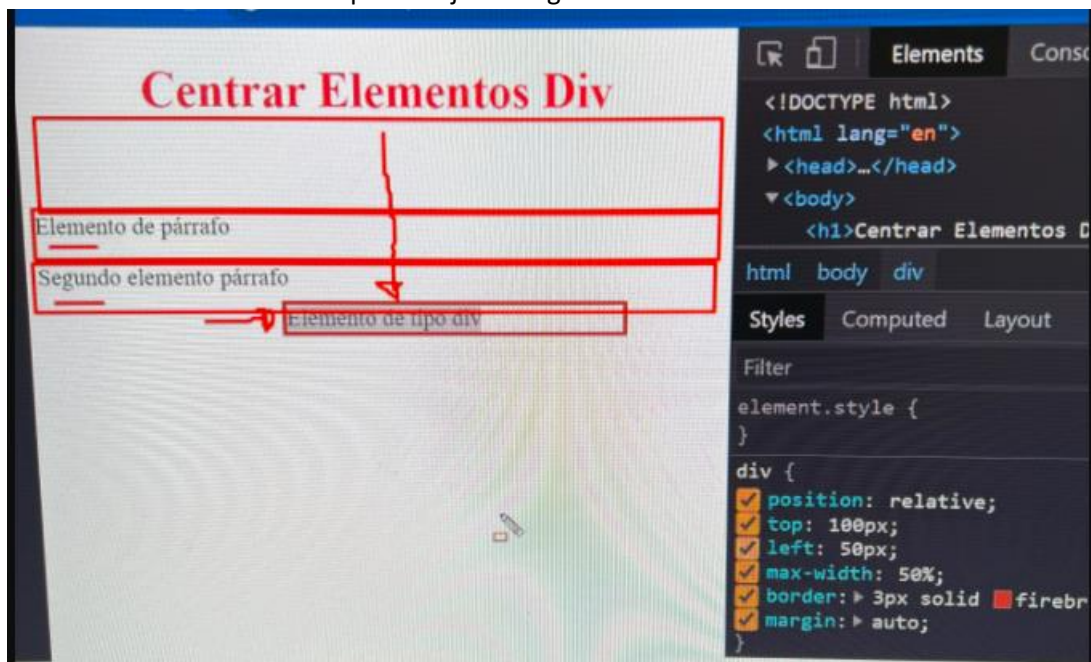
Entonces

Position:"relative"

Podemos darle valores a dicha posición, como:

Top, left, bottom, right, entonces alli podremos moverlo segun como deseemos

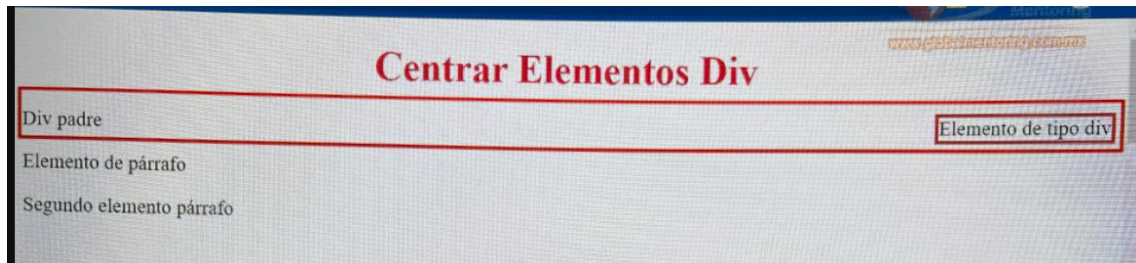
Si agregamos un párrafo dps de nuestro elemento div, esta no se vera afectada. Entonces El elemento div reservara su espacio dejando lugar a las afuera de donde este se encuentre.



Fixed: Es para definir que un elemento sea fijo

Se suele usar como un elemento div

Absolute. Nos da un valor absoluto y fijo

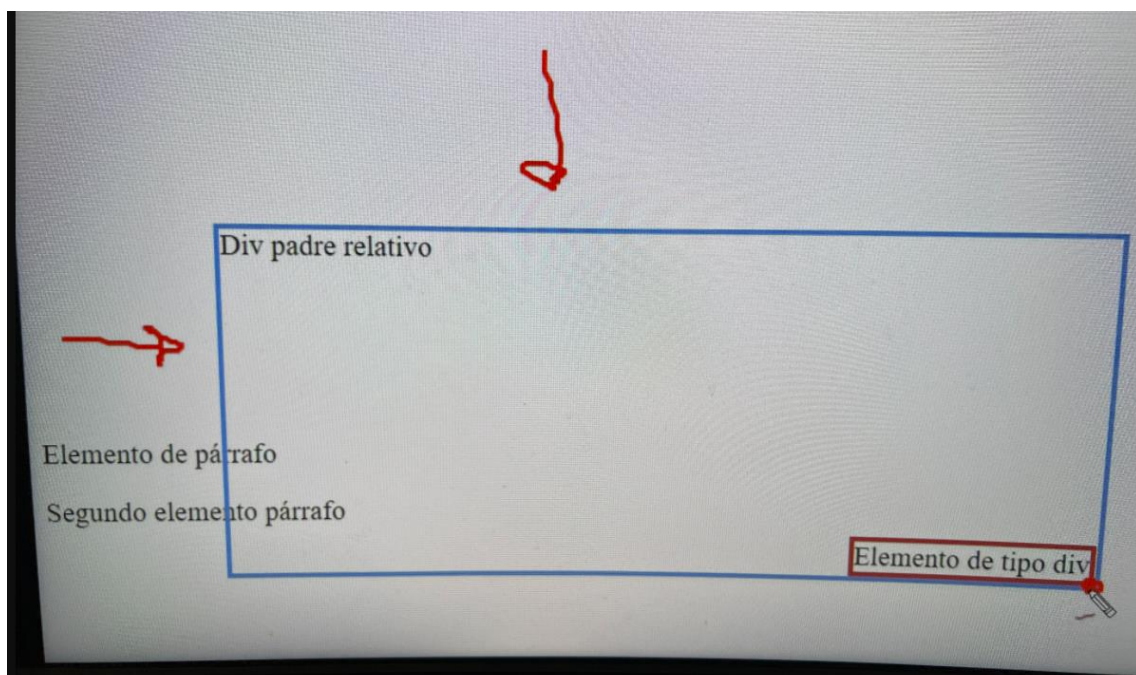


Como podemos ver absolute se utiliza como para dar un valor absoluto al cuadro padre entonces tenemos el div padre y el elemento del tipo div hijo como un valor absoluto dentro de ese elemento.

Y toma valores asi:

```
div.absoluto{  
  position: absolute;  
  bottom:0;  
  right:0;  
  border: 3px solid ■ firebrick;  
}
```

El movimiento que hagamos con el padre el hijo lo va a seguir siempre del mismo modo .




```
div.relATIVO{  
    max-width: 300px;  
    height: 200px;  
    position: relative;  
    top: 10px;  
    left: 100px;  
    border: 3px solid ■ cornflowerblue;  
}  
  
div. absoluto{  
    position: absolute;  
    bottom: 0;  
    right: 0;  
    border: 3px solid ■ firebrick;  
}
```

Aca podemos ver el valor relativo, y absoluto uno que se mueve libremente o sea que se agrega donde debe hacerlo y sin pisar nada y el hijo anclado como absoluto

Ahora por ultimo vamos a ver el tipo sticky:

Siempre tiene que tener el top con 0

Tiene la función de funcionar como un objeto relativo y cuando sale de pantalla se pone de manera fija. Y cuando volver a ajustar la pantalla se ajusta a su lugar específico

Función de z-index, con esta vamos a poder tener un manejo como en el plano xyz para nuestros módulos de html


```

h1{
    text-align: center;
    color: crimson;
}
/* al usar el div y contenedor es para que aplique al lugar específico
que se le asigna y no a todos los contenedores en este caso */
div.contenedor{
    position: relative;
    border: 3px dashed greenyellow;
    width: 30%;
    margin: auto;
}
/* para que no ocupe toda la pantalla y el margin para que se alinee solo
*/
div.centro{
    position: absolute;
    /* para que este a la mitad */
    top: 50%;
    /* antes del width este objeto ocupa el espacio necesario y no el del
todo el contenedor que lo contiene */
    width: 100%;
    /* border: 1px blue solid; */
    text-align: center;
    font-size: 25px;
}
img.imagen{
    /* para que use todo el contenedor */
    width: 100%;
    opacity: .2;
}

```

Es importante remarcar que con la posición absoluta el texto ocupa siempre el tamaño de su mismo valor. Pero si agrandamos este se extenderá por el ancho de la pantalla y no por el alto. Así con este vamos a poder centrarlo o arreglarlo

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Centrar elementos DIV</title>
    <link rel="stylesheet" href="/css/estilos.css">
</head>
<body>
    <h1>Imagen con Texto Centrado</h1>
    <div class="contenedor">
        
        <div class="centro">Logo html5</div>
    </div>
</body>
</html>

```

Overflow en HTML

Cuando tenemos texto dentro de un elemento

Siempre por defecto tenemos que esta el overflow visible

Pero con hidden lo podemos ocultar

Y con scroll lo podemos recorrer

Y con auto el css y el html lo definira

Todo esto sucede por lo general cuando especificas un atributo de alto o de ancho siempre se configura para el eje x o el eje y

Propiedad Float:

Float nos permite colocar mas de un elemento en la misma línea. Utilizando dos div distintos pueden ser colocados al costado para que estos se vayan agrando ocupando una misma línea

Float tiene atributos como left right entre otros

Podemos separar elementos a travez de clases y personalizarlo

Podemos usar clear para limpiar una línea y no considerar la propiedad de float

```
div.contenedor{
  width: 30%;
  height: 100px;
  overflow: auto;
}
div.contenedor.flotado{
  float: left;
}
div.contenedor.flotado.contenedor1{
  border: 3px dashed greenyellow;
}
div.contenedor.flotado.contenedor2{
  border: 3px dashed cornflowerblue;
  float: right;
}
```

Como podemos ver en la imagen aca tenemos un div a la izquierda y otro en su extremo opuesto sin conservar nada en el centro. Para agregar un nuevo div extra en una línea nueva se utiliza la propiedad **Clear** esta nos anexa una nueva línea y rellena los espacios vacíos para que podamos trabajar en una nueva línea

Both porque no tenemos elementos flotados a la izquierda o derecha

No se usa

Flotado

```
div.contenedor.contenedor3{
  clear: both;
  border: 3px dashed darkorange;
  margin: auto;
}
```

Los elementos span son de tipo inline no podemos controlar el margen, el alto o el padding. Pero al hacerlo un bloque si vamos a poder tener mas control sobre este.

Con margin auto siempre centraremos nuestro elementos.

Con la propiedad display nos permite tratar a un elemento como bloque en este caso, al bloque de span lo vamos a transformar

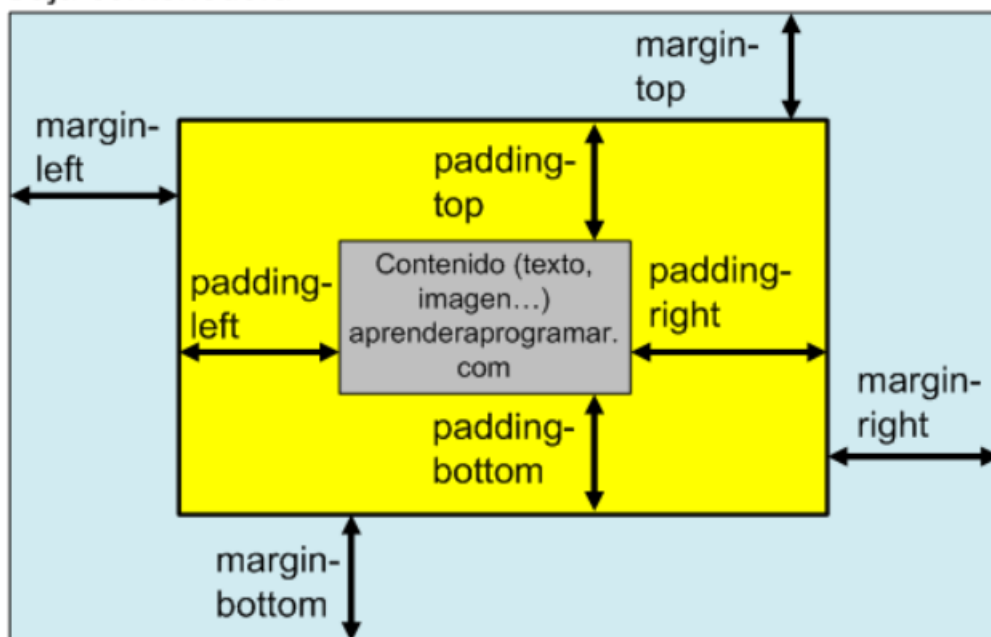
Para que se comporte en una línea y un bloque se usa inline-block

Ahora agregamos un menú de navegación

Gracias al inline-block nos permite trabajar con los elementos en bloque y a su vez nos permite trabajar con los elementos en línea. Pero el fuerte es que con el bloque vamos a poder asignarle valores extra. Como un fondo un alto un ancho (padding y margin) y lo que deseemos en cambio en línea va a ser un valor fijo el alto y al ancho ya que usaran solo el tamaño dependiente de ellos.

Tener en cuenta padding y margin

Caja contenedora



Selectores en css

- 1) **Descendiente**: En cualquier nivel interno. O sea que se va a aplicar a todos sus sub niveles sin importar si existe un elemento directo dentro del div por ejemplo o uno mas interno aun como section.
- 2) **Child** o hijo: En este caso se tiene en cuenta sus hijos directos y no uno que se encuentre dentro del mismo contenedor pero a su vez este dentro de otro contenedor como un section

Se indica de esta manera:

```
<body>
  <h1>Selectores Combinados en CSS</h1>
  <div class="contenedor descendiente">
    Selectores descendientes (en cualquier nivel interno):
    <p>Primer renglón</p>
    <p>Segundo renglón</p>
    <section>
      <p>Tercer renglón</p>
    </section>
  </div>
  <div class="contenedor hijo">
    Selectores Hijo (child):
    <p>Primer renglón</p>
    <p>Segundo renglón</p>
    <section>
      <p>Tercer renglón</p>
    </section>
  </div>
```

Para este caso, vamos a tener en cuenta: cada ocasión que aplica. Si aplicamos descendiente o si aplicamos child:

Como podemos ver. Aquí sucede, cuando se aplica descendiente p que se aplica todo dentro de tus Métodos. Y en cambio con **Child** se aplica solo para Los hijos directos en este caso <p> y lo simboliza con el símbolo ">"

Si nos posicionamos encima de dichos div.contenedores: vamos a poder ver que el primero aplica para todos los p sin importar donde o sea hijos no directos y el otro solo para los p directos o hijos directos

```
div.contenedor{
  width: 80%;
  height: 20%;
  margin: auto;
  border: 3px solid #a8dad6;
  padding: 15px;
  color: #457b9d;
}
div.contenedor.descendiente p{
  background-color: #fdffb6;
}
div.contenedor.hijo > p{
  background-color: #ffc6ff;
}
```

- 3) **Selector sibling o adyacente:** Con este tipo de selector vamos a tomar el primer método que se indique por eso adyacente es el que continua directamente luego del primer método indicado y cerrado. No tendrá en cuenta lo que dentro del div exista sino dps del div lo que venga con la etiqueta deseado solo se cumple una vez.

```
div.contenedor.adyacente + p {  
  background-color: #cafffbf;  
}
```

Así es como definimos el elemento adyacente:

Definimos un elemento adyacente con "+"

```
<div class="contenedor adyacente">  
  Selector adyacente:  
  <p>Primer renglón</p>  
</div>  
<p>Segundo renglón</p>  
<p>Tercer renglón</p>
```

- 4) **Selector hermano:** Se encuentran al mismo nivel que el div y a diferencia del adyacente que se aplica al final del div aca se aplica a todos los elementos tipo párrafo dps del tipo div. O bueno puede ser otro tipo como un ancla o link

```
<div class="contenedor hermano">  
  Selector hermano (sibling):  
  <p>Primer renglón</p>  
</div>  
<p>Segundo renglón</p>  
<p>Tercer renglón</p>
```

```
div.contenedor.hermano ~ p {  
  background-color: #9bf6ff;  
}
```

```
div.contenedor {  
  width: 80%;  
  height: 20%;  
  margin: auto;  
  border: 3px solid #a8dadc;  
  padding: 15px;  
  color: #457b9d;  
}  
div.contenedor.descendiente p {  
  background-color: #fdffb6;  
}  
div.contenedor.hijo > p {  
  background-color: #ffc6ff;  
}  
div.contenedor.adyacente + p {  
  background-color: #cafffbf;  
}  
div.contenedor.hermano ~ p {  
  background-color: #9bf6ff;  
}
```

Aca podemos ver todos los tipos de :
SELECTORES

Pseudo clases: O clases que no existen

Aplicando la clase contenedor oculto:

Utilizamos

El método hover () que especifica dos funciones para ejecutar cuando el puntero del mouse pasa sobre los elementos seleccionados dentro de esta clase.

```
<body>
  <h1>Pseudo Clases en CSS</h1>
  <div class="contenedor oculto">
    Pasa por encima
    <p>Se muestra el párrafo</p>
  </div>
```

Definimos una clase de este tipo

y:

```
div.contenedor{
  width: 30%;
  height: 20%;
  margin: auto;
  border: 3px solid #a8dadc;
  padding: 15px;
  color: #457b9d;
  text-align: center;
}
div.contenedor:hover{
  background-color: #457b9d;
  color: #f1faee;
}
```

Con hover le damos una tarea al pasar por

encima y ahora lo vamos a ocultar

```
div.contenedor.oculto:hover p{
  display: block;
  background-color: #a8dadc;
  color: #023e8a;
  padding: 15px;
}
```

Como podemos ver. Ahora al contenedor

oculto al pasar la flecha por encima (hover) y teniendo en cuenta la clase p va a tratar a la clase p como un bloque y le va a agregar funciones de padding o margin y un nuevo color



```
div.contenedor.primer p:first-child{
  font-variant: small-caps;
}
```

Con esa pseudo clase vamos a poder modificar el tamaño de nuestro primer tipo de elemento en este caso párrafo.

Podemos ver que al escribir una clase y asignarle con ":" un valor a ese parámetro de la clase. Obtenemos nuevas funciones que modificaran nuestro archivo. Siempre es indicado tener en cuenta que por ejemplo donde pongamos la función hover o la función first-child ya que estas nos traerán reacciones distintas

Es importante aclarar que: Cuando utilizamos pseudo clases se utiliza los dos puntos y cuando utilizamos pseudo elementos se utilizan dos dos puntos.

Entonces a un párrafo que le aplicamos una pseudo clase se vería como en lo escrito anteriormente pero si a este le agregamos 2 dos puntos obtendremos nuevas mejoras como por ejemplo

```
div.contenedor | p::selection{  
  color:  red;  
  background-color:  yellow;  
}
```

y así lo vemos

```
<div class="contenedor">  
  ...  
  <p ::selection>  
Selector Specificity: (0, 1, 3)
```

Así trabaja

Entonces aplicara al momento de seleccionar el párrafo un sombreado de otro color

Gracias al método hover el cual es una pseudo clase. Podemos hacer que una imagen se vuelva colorida al pasar sobre ella. Con un opacity de 50% a 100%

Ahora vamos a ver como aplicar **Gradientes**:

En primer instancia vamos a aplicar un gradiente lineal

```
<h1>Fondo con Gradiente en CSS</h1>
<div class="contenedor">
  <p>Logo de HTML</p>
  
</div>
/body>
/html>
```

```
html{
  height: auto;
  min-height: 100%;
}
body{
  background: linear-gradient(■ #f72585, ■ #3a0ca3);
}
```

Como podemos ver. Se define html como auto para que dependiendo del tamaño de la pagina el html se ajuste y luego un min-height para que el mínimo del alto de la pagina sea el 100%

Y luego definimos el gradiente lineal que vamos a utilizar. Como podemos ver. Es muy importante aclarar que esto sirve para que el gradiente no se quede solo en una parte de la pantalla y que se aplique a toda la pagina web

```
/* background: linear-gradient(#f72585, #3a0ca3); */
/* background: linear-gradient(to right, #f72585, #3a0ca3); */
/* background: linear-gradient(#24404d, #6dbff1, black); */
/* background: linear-gradient(to bottom right, #f72585, #3a0ca3); */
/* background: linear-gradient(145deg, #f72585, #3a0ca3); */
background: radial-gradient(circle, ■ #0d47a1, ■ #64b4f5dc);
color: ■ #f1faee;
}
```

Acá podemos ver todas las variaciones de gradiente

Deg es para indicar un angulo

Ahora conoceremos el text-shadow y box-shadow uno para texto y otro para cajas como div. O sombras. Donde se indican los (el eje x, eje y, color) o (x,y, blureado en pixeles también, y color?) (2px, 2px, 5px, grey) y también tenemos (x,y,b, tamaño, color)

Flexbox, esto viene a remplazar el float o algunas mejores mecánicas para así poder acomodar nuestros tipo párrafo o algún otro tipo de escritura

Seguiremos utilizando la propiedad display con el atributo flex

Y luego solo se deben adaptar a través de un selector combinado lo que querramos modificar

```
div.contenedor{
  display: flex;
  background-color: #560bad;
}
div.contenedor > p{
  background-color: #4895ef;
  color: #f1faee;
  margin: 20px;
  padding: 15px;
  font-size: 1.5em;
}
```

Entonces primero tratamos al div como un display flex esta propiedad nos ayudara a que los elementos de allí dentro se pongan al costado. Luego le damos un color

Por ultimo a los elementos tipo párrafo lo vamos a tratar para ello creamos una clase nueva

div.contenedor y como vimos anteriormente utilizando > podemos editar todos los tipo p que se encuentren dentro del div.contenedor

Flex-flow: row wrap

Propiedad de atributos

Para manejar herramientas tipo párrafo dentro de un div

El cual con esta función podemos contenerlo y ejecutar diferentes tareas como dar un tamaño

```
div.contenedor{
  display: flex;
  background-color: #560bad;
  /* flex-direction: column-reverse; */
  /* flex-direction: row; */
  flex-wrap: wrap;
  flex-flow: row wrap;
  justify-content: center;
  height: 300px;
  align-items: stretch;
  /* align-content: center; */
}
```

Métodos de alineación y métodos de trabajo con ítems interiores

