

Diseño de Páginas Web

HTML



CSS



Índice de contenido

Requerimientos	4
HTML	5
- Disposición general de una página web	6
- Estructura en HTML	7
- Etiquetas básicas de HTML	9
- Atributos básicos de HTML	12
- Formularios de HTML	14
- Tablas de HTML	16
CSS	17
- Formas de escribir código CSS	18
- Estructura en CSS	19
- Selectores en CSS	20
- Rango de especificidad	21
- Metodología BEM	22
- Medidas	24
- Propiedades en CSS	25
✓ Márgenes de la página web	25
✓ Cajas (tamaños)	25
✓ Tipografía (tipo de letra)	26
✓ BoxModel (content, padding, border, margin)	27
✓ Position	30
✓ Display	33
- Imágenes	35
- Background	36
- Iconos	37
- Pseudo-elementos	38
- Pseudo-clases (:hover, etc)	40
- Responsive design – Mobile first	42
- Flex-box	47
- Grid	51
- Media queries	58
- Transition	60
- Keyframes (animaciones)	61
- Transform	62
- Variables	63
- Filter	64
- Otras propiedades en CSS	66
- ¿Cómo centrar un DIV?	67

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>index.html</title>
  <link rel="stylesheet" type="text/css" href="normalize.css">
  <link rel="stylesheet" type="text/css" href="estilo.css">
  <script src="https://kit.fontawesome.com/62ea397d3a.js" crossorigin="Anonymous"></script>
</head>

<body>

  <header>
    <nav>

      </nav>
    </header>

    <main>

    </main>

    <footer>

    </footer>

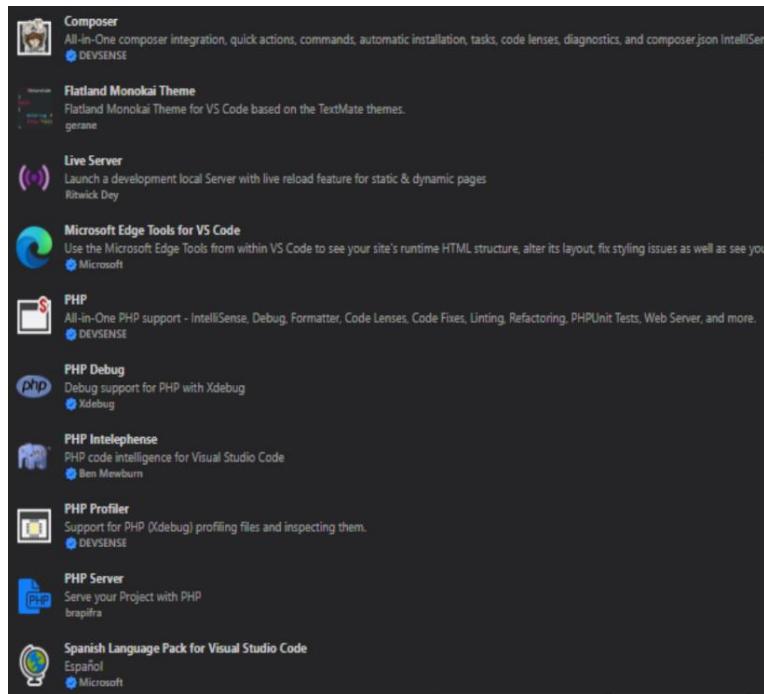
  </body>
</html>
```

```
* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
  font-weight: 100;
}
```

Requerimientos

Visual Studio Code

- Se requieren las siguientes extensiones



HTML



Establece el contenido de una página web, pero no su diseño ni funcionalidad.

Sólo sirve para estructurar una página web por medio de “**cajas**” llamadas “etiquetas”.

La página inicial de todo código HTML es titulada “index.html”.

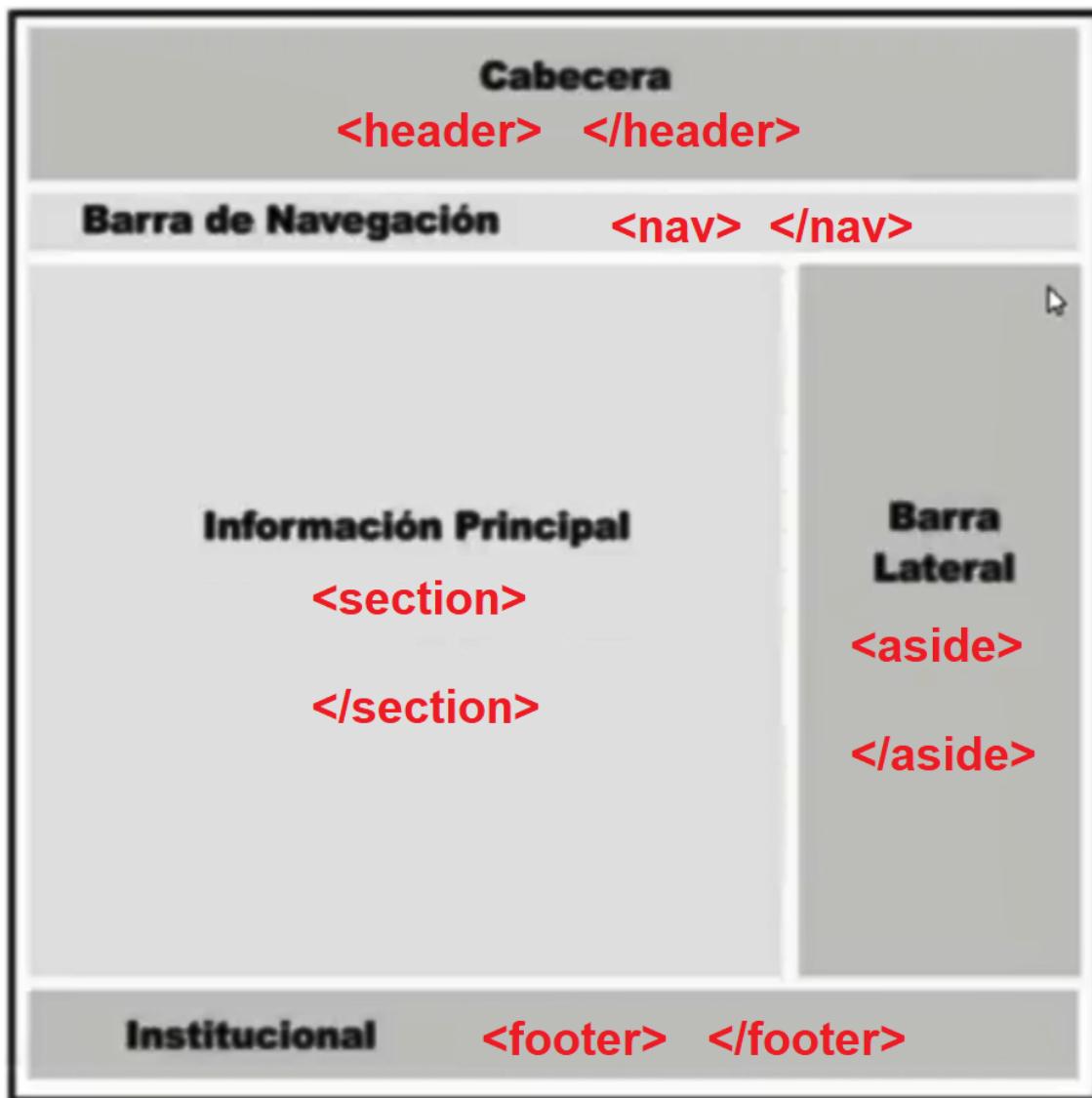
Document Object Model

Estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM.

Entonces, toda etiqueta ocupa un espacio en el DOM (invisible para el usuario) donde se posicionará dicha etiqueta y no otra.

Con CSS, encontraremos funciones que permitan “eliminar” el espacio utilizado de una etiqueta en el DOM, haciendo que una etiqueta pueda ocupar el mismo espacio en pantalla que otra, así como posicionar dicha etiqueta en cualquier parte de la página.

Disposición general de una página web



- **Cabecera**: logo y nombre de la página (título, subtítulo y descripción).
- **Barra de navegación**: menú o links.
- **Información principal**: artículos.
- **Barra lateral**: información sobre el artículo (fecha de publicación, etc.).
- **Institucional**: pie de página (información de creador del blog, etc.).



Estructura en HTML

HTML se utiliza para estructurar una página web por medio de etiquetas con el siguiente formato:

```
<etiqueta atributo="valor"> contenido </etiqueta>
```

- **Etiqueta** Nombre de la etiqueta con una función específica.
- **Atributo** Establece una propiedad específica a la etiqueta.
- **Valor** Característica que tendrá el atributo de la etiqueta.
- **Contenido** Contenido de la etiqueta.

Cajas de HTML

Las etiquetas de HTML representan “**cajas**” en la estructura de la página web.

Existen dos tipos de cajas:

- Display Block: ocupa toda la línea horizontal disponible.
- Display inline: ocupa sólo el contenido de la etiqueta.

```
<body>
<h1> Encabezado </h1>
<p> Esto es un <b> párrafo </b> y usé negrita
</p>
<div>
  <h2> Caja acotada por la etiqueta div </h2>
</div>
</body>
```

Nombre del primer archivo HTML

Toda página web tiene como su primer página el nombre “**index.html**”.

Versión de HTML del documento

En la primera línea del código, se debe aclarar la versión de HTML usada. Para ello se escribe: **<!DOCTYPE html>**



Jerarquías de HTML

text.html x Primer Pagina Web file:///C/Users/fazt/Desktop/primer-sitioweb/index.html

Titulo del Texto

mi primer parrafo de html

} cabeza <head>

} cuerpo <body>

Estructura jerárquica básica del código HTML

EXPLORER ...

✓ MIPRIMERHTML

index.html

```
1  <!DOCTYPE html>
2
3  <html>
4      <head>
5          ...
6      </head>
7
8      <body>
9          ...
10     </body>
11 </html>
```

... → contenido de la cabeza

... → contenido del cuerpo

Ejemplo de una primera página web y su estructura jerárquica

X / Primer Pagina Web file:///C/Users/fazt/Desktop/primer-sitioweb - Visual Studio Code

Archivo Editar Selección Ver Ir Depurar Tareas Ayuda

Titulo del Texto

mi primer parrafo de html

EXPLORADOR

EDITORES ABIERTOS

PRIMER-SITIOWEB

index.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Primer Pagina Web</title>
5      </head>
6      <body>
7          <h1>Titulo del Texto</h1>
8          <p>mi primer parrafo de html</p>
9      </body>
10 </html>
```



Etiquetas básicas de HTML

Cabeza (<head>)

Es la parte del código HTML donde se introduce el título de la pestaña que mostrará la página web y otros datos de la página que el usuario no puede ver.

- **<title>** Nombre de la página web. Es la única etiqueta visible por el usuario.
- **<link>** Permite agregar un ícono a la página web. Se debe especificar la ruta de la imagen con formato .ico `<link rel="icon" href="lucas.ico">`
- **<meta/>** Añade información de la página por medio de atributo de etiquetas. Esta etiqueta no tiene un par, por eso se puede agregar la barra slash (/) al final, aunque no es una buena práctica (se puede dejar sin dicha barra, pero en este resumen se lo pondrá para indicar cuando una etiqueta no tiene un par).
 - **name** nombre de la información que se añadirá
 - **description** descripción de la página web (70-140 letras).
 - **author** nombre del autor de la página web.
 - **keywords** palabras clave para encontrar la página web, separadas por comas.
 - **copyright** añade datos de la empresa registrada.
 - **robots** define si la página puede ser indexada o no. `"noindex"`
 - ❖ **content** información a añadir

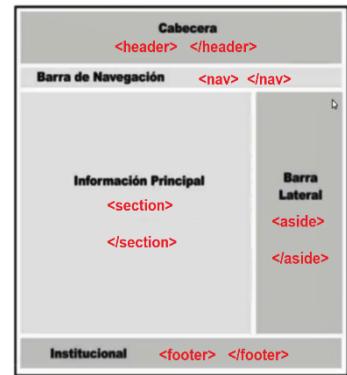
```
<meta name="description" content="Está es mi primer Pagina Web" />
<meta name="author" content="Fazt"/>
<meta name="keywords" content="Mi primer titulo, mi primer pagina, pagina web">
```
 - **charset** tipo de codificación del texto `<meta charset="utf-8" />`



Cuerpo (**<body>**)

Se divide en distintas secciones, para dar formato a la página web según la disposición general de una página web.

- **<header>** Cabecera de la página
 - ⊕ **<h1>** Encabezado (el número indica el tamaño de la letra). Se debe usar sólo una vez cada número.



- **<nav>** Barra de navegación (puede estar dentro del <header>)

- ⊕ **** Lista desordenada, “Unorder list”.
 - **** Item de la lista, “List Item”.
- ⊕ **** Lista ordenada, “Order list”. También utiliza ****.

```
<ul>
  <li>Principal</li>
  <li>ayuda</li>
  <li>contacto</li>
</ul>
```



- Principal
- ayuda
- contacto

List-style: none; → Quita los puntos de los list-item

- **<section>** Información principal

- ⊕ **<article>** Establece que se trata de un artículo. Se pueden añadir otras etiquetas dentro del par de esta etiqueta.

- ⊕ **<time>** Añade información sobre la fecha. Se puede añadir un atributo de etiqueta “**datetime**”.

```
<time datetime="12-10-2018">Publicado 12-10-2018</time>
```

- ⊕ **<p>** Párrafo.

- *lorem* Escribe 30 palabras de prueba en latín (presionando TAB).

➤ **<small>** Se puede usar dentro de un texto para achicar la letra de una sola porción de dicho texto.

➤ **** Registra una porción de texto que sirve para modificar luego con CSS.

➤ **<cite>** Establece una porción de texto como una cita, mostrándose en cursiva. Se puede, además, añadir un enlace a otra página.

➤ **** Establece una porción de texto en Negrita.



- **<i>** Establece una porción de texto en Itálica (letra inclinada).
- **<strike>** Establece una porción de texto como texto tachado.
- **
** Da un salto de línea (completa la línea de la caja).

- **<aside>** Barra lateral

✳️ **<blockquote>** Añade una cita.

```
<blockquote cite="http://faztweb.com">  
Mensaje 1  
</blockquote>
```

✳️ **<address>** Agrupa hiperenlaces.

- **<a>** Añade un link a otra página mediante el atributo de etiqueta *href*.

```
<a href="http://faztweb.com">Visita mi Sitio Web</a>
```

```
.nav__li a { /* links */  
text-decoration: none;
```

→ **Quita el subrayado**

- **<footer>** Pie de página.

✳️ **** Permite añadir una imagen mediante un atributo de etiqueta “*src*”. ``

- ✓ La imagen a utilizar, debe hallarse dentro de la carpeta de nuestro proyecto.
- ✓ Si se desea crear una carpeta dentro del directorio de nuestro proyecto para almacenar la imagen. Al momento de llamar a la imagen, se debe especificar su carpeta. ``
- ✓ Se puede añadir cualquier imagen de internet simplemente copiando el link de la imagen, iniciando con **HTTPS://**
- ✓ Si el archivo se encuentra en una carpeta anterior (local), se debe utilizar la expresión **..**/ para retroceder una carpeta.
- ✓ Se pueden utilizar los atributos **width** y **height** para modificar el tamaño de la imagen, sin embargo, luego no se podrá modificar con CSS. `height="600px" width="600px"`

- ✓ Se puede utilizar el atributo **alt** para que se muestre un nombre en caso que la imagen no se haya podido encontrar.
¡Importante!

```

```



- ✓ Se puede utilizar el atributo **title** para establecer un título a la imagen. Cuando el usuario posicione el puntero sobre la imagen, se mostrará el nombre elegido.

- ✓ Se puede colocar la imagen dentro de la caja **<center>** para centrar la imagen.

Nota: Sirve también para textos.

```
<center>
</video>
```

- ✳ **<audio>** Permite añadir un audio mediante un atributo de etiqueta “**src**”. Es necesario agregar el atributo “**controls**” para que el usuario pueda reproducir dicho audio. Nota: se puede añadir el link de un video, en ese caso, se reproducirá solo el audio de dicho video.



Atributos básicos de HTML

Permite dar propiedades a etiquetas específicas de HTML.

```
<etiqueta atributo="valor"></etiqueta>
```

- Lenguaje (*lang*): dentro de <html>
- Fecha (*datetime*): dentro de <time>

```
1  <!DOCTYPE html>
2  <html lang="es">
3  |   <head>
```

```
<time datetime="12-10-2018">Publicado 12-10-2018</time>
```

- Hiperenlace (*href*): dentro de <a>
 - o Abrir nueva página en una pestaña nueva: *target=_BLANK*

Haz [click aquí](https://google.com) para ir a Google

- o Direccionar a un punto específico de la página web:
Para poder redireccionar dentro de una página web, se debe seleccionar un objetivo (una etiqueta identificada). Para ello, primero debemos darle identidad (selector en CSS) a una etiqueta mediante el atributo **id** **<h2 id="codigo">El código es: ij3i4j3idcd1kgasok98</h2>**
Una vez establecido el id de la etiqueta objetivo, se puede utilizar el atributo **href** dentro de <a> para redirigir a la etiqueta con el id establecido, agregando un # al inicio:

```
<a href="#codigo">Click aquí para ir al código</a> <br>
```

Formularios

Los formularios en HTML son etiquetas que contienen **input**, procesamiento y **output** de datos.

Input

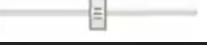
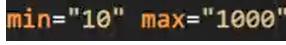
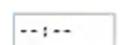
```
<form>
  <input type="" name="">
</form>
```



Permite la entrada de datos.

Donde el atributo **type** indica el tipo de input que se utilizará, y el atributo **name** establece el identificador de dicho input (para poder procesarlo luego).

Tipos de input:

- **text** Permite escribir texto (establecido por defecto). 
- **password** Permite escribir texto sin mostrarlo en pantalla. 
- **number** Permite escribir sólo números. 
- **email** Permite escribir sólo texto con formato email. 
- **color** Permite seleccionar un color:  → 
- **range** Permite elegir un número del A al B con una barra
El rango de valores se puede elegir con los atributos: 
- **date** Permite seleccionar una fecha. 
- **time** Permite seleccionar una hora. 
- **button** Muestra un botón, cuyo texto puede modificarse mediante el atributo "**value**". 
- **submit** Botón de inicio del procesamiento de los datos introducidos. 
 - **value** Texto dentro de la caja de input.
 - **required** Establece que un input debe ser llenado sin excepción.
 - **placeholder** Establece una pista en la caja para orientar al usuario.

```
<input type="password" required="">
```

TextArea

Una etiqueta **<textarea>** permite añadir un cuadro donde se pueda añadir texto, por default, la caja de texto puede ser redimensionada a cualquier tamaño por el usuario.

Algunos atributos útiles para esta etiqueta son:



- **readonly** Establece que la caja de texto no pueda ser editada.
- resize: none;** → **el usuario no podrá redimensionar un text area**

Method

Dentro de la etiqueta **<form>** se puede utilizar el atributo **method**, el cual sirve para establecer cómo/dónde se procesarán los datos (backend).

- **get** Se trabajan los datos localmente.
- **post** Se trabajan los datos en un servidor.
- **enc** Establece que los datos serán encriptados.

Agrupación de campos

Se utiliza **fieldset** para agrupar campos.

```
<form>
  <fieldset class="form-campos">
    <legend>Contactanos llenando todos los campos</legend>

    <div class="contenedor-campos">
      <div class="campo">
        <label>Nombre</label>
        <input type="input-text" placeholder="Tu nombre" name="nombre">
      </div>

      <div class="campo">
        <label>Teléfono</label>
        <input type="input-number" placeholder="Tu número" name="numero">
      </div>

      <div class="campo">
        <label>Mensaje</label>
        <textarea type="input-number" name="mensaje"> </textarea>
      </div>
    </div>

    <div class="boton-contenedor">
      <input type="submit" value="Publicar">
    </div>
  </fieldset>
</form>
```

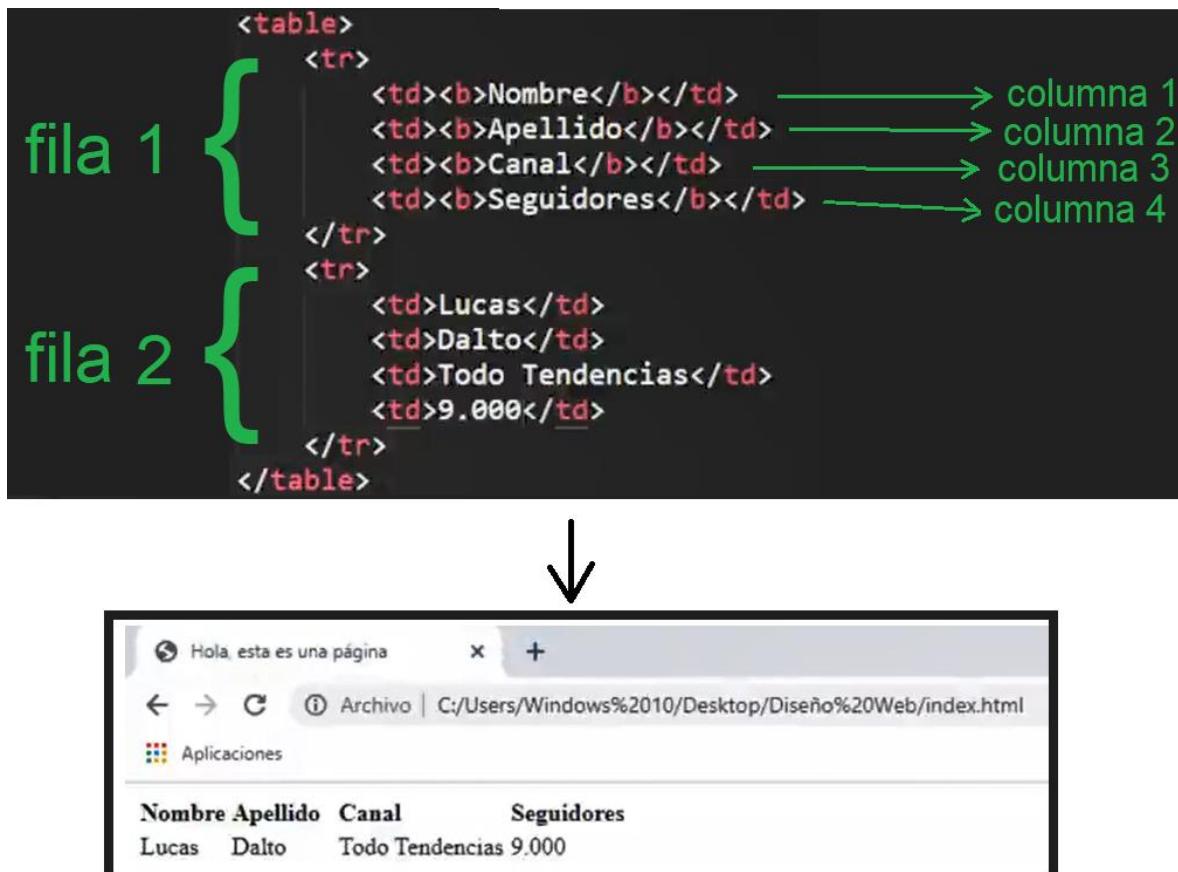
Contactanos llenando todos los campos

Nombre	<input type="text" value="Tu nombre"/>
Teléfono	<input type="text" value="Tu número"/>
Mensaje	<input type="text"/>
<input type="button" value="Publicar"/>	

Tablas

Estructuras que contienen **filas** y **columnas**. Se crean con la etiqueta **<table>**.

Una vez creada la tabla, se crea una fila mediante la etiqueta **<tr>** y luego se crea cada columna con la etiqueta **<td>**.



Se pueden centrar los textos de las celdas de la tabla con el atributo **<center>**.

Sin embargo, hay que agregarlo dentro de la caja de cada columna en cada fila. Existe un método más práctico de centrado en CSS.

Se pueden dar atributos a las tablas:

- **border** Establece un borde (en píxeles) a la tabla. Sin embargo, con CSS se pueden personalizar los bordes aún más.

CSS

(Cascading StyleSheets)



El lenguaje CSS, “Hojas de estilo en cascada”, sirve para tomar las etiquetas escritas con HTML y darles un estilo (colores, fondos, sombras, etc.).

Define los estilos, paso por paso, dando siempre prioridad a los últimos.

- 1- **COLOR DEL H1 = ROJO**
- 2- **TAMAÑO DEL H1 = 100PX**
- 3- **TAMAÑO DEL H1 = 70PX** →

Este será el tamaño que obtendrá el H1 finalmente



Formas de escribir código CSS

Estilo de un archivo externo (<link>)

Tipo de código más utilizada y aceptada.

```
<link rel="stylesheet" type="text/css" href="">
```

Con este código, se está pidiendo que traiga una “hoja de estilo”, que tipee texto CSS que está alojada en una determinada ruta (href).

Los archivos externos de los cuales obtendrá el estilo deseado, se almacenan en archivos de extensión .css

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Aprendiendo CSS con @SoyDalto</title>
5      <link rel="stylesheet" type="text/css" href="estilo.css">
6  </head>
7  <body>
8  ...
9  </body>
```

es necesario crear un archivo .css
para empezar a crear nuestro estilo

Estilo en línea

Se utiliza el atributo **style** directamente en una etiqueta.

```
<p style="color:red">
```

Estilo definido por una etiqueta (<style>)

Se trabaja con código CSS como una etiqueta más de HTML.

```
<style type="text/css">
...
</style>
```

→ código CSS

Estilo de import

```
@import
```

No recomendado.



Estructura en CSS

CSS se utiliza para dar estilo a la estructura generada con HTML mediante código con el siguiente formato:

```
selector {  
    propiedad: valor;  
}
```

- **Selector** Establece el elemento o conjunto de elementos a los cuales se desean cambiar sus propiedades.
- **Propiedad** Característica que se desea cambiar.
Por ejemplo: color, tamaño, etc.
- **Valor** El valor que tomará la propiedad seleccionada.
Por ejemplo: color verde (en sexagesimal).



Selectores

- **Universal** * Selecciona todos los elementos

```
* {  
    color: red;  
}
```

- **De tipo** Selecciona por elemento (etiqueta)

```
h1 {  
    color: red;  
}
```

- **Clases** • Selecciona todas las etiquetas que tengan un atributo **class** determinado.

```
<h1 class="myclass"> Hello World! </h1>
```

```
.myclass {  
    color: red;  
}
```

- **ID** # Selecciona una etiqueta específica que tenga el atributo **id** determinado. Tiene mayor rango de especificidad que las clases.

```
<h1 id="myelement"> Hello World! </h1>
```

```
#myelement {  
    color: red;  
}
```

- **Por atributo** [] Selecciona todas las etiquetas que tengan un atributo específico.

```
<h1 myatributo="myvalor"> Hello World! </h1>
```

```
[myatributo="myvalor"] {  
    color: red;  
}
```

- **Descendiente** Selecciona etiquetas dentro de otras etiquetas (contenedoras). La etiqueta a seleccionar es la que está más a la derecha, mientras que la etiqueta más contenedora está a la izquierda.

```
<p> <b>Hello</b> World! </p>
```

```
p b {  
    color: red;  
}
```

También se puede combinar con otros selectores.

```
.myclass b p {  
    color: red;  
}
```

- **Pseudo-clases** Permite generar selección en un evento. Es decir, que las etiquetas reciban interactividad con determinadas operaciones.

```
b:hover {  
    color: red;  
}
```

Las letras se volverán rojas cuando el puntero pase sobre las etiquetas



Rango de especificidad

En CSS existen distintos niveles de prioridad para establecer los estilos deseados.

Cuando se otorgan estilos en un mismo nivel de prioridad, siempre quedará establecido el que se escriba último, es decir, más abajo en el código (por eso es “cascada”).

Si se otorga un estilo en un nivel de alta prioridad, quedará establecido dicho estilo incluso si en otra línea del código se otorga un estilo diferente en un nivel de prioridad menor.



Ejemplo

Teniendo el siguiente código HTML:

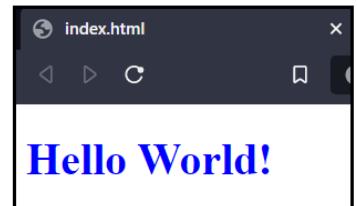
```
<body>
  <h1 id="titulo"> Hello World! </h1>
</body>
```

Y el siguiente código CSS:

```
#titulo {
  color: blue;
}

h1 {
  color: red;
}
```

se obtiene →



El selector de clase tuvo prioridad sobre el selector de etiqueta.

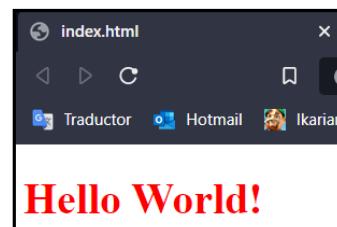
Important

Cuando se agrega “**!important**” luego del **valor**, se establece la prioridad máxima.

```
#titulo {
  color: blue;
}

h1 {
  color: red !important;
}
```

se obtiene →





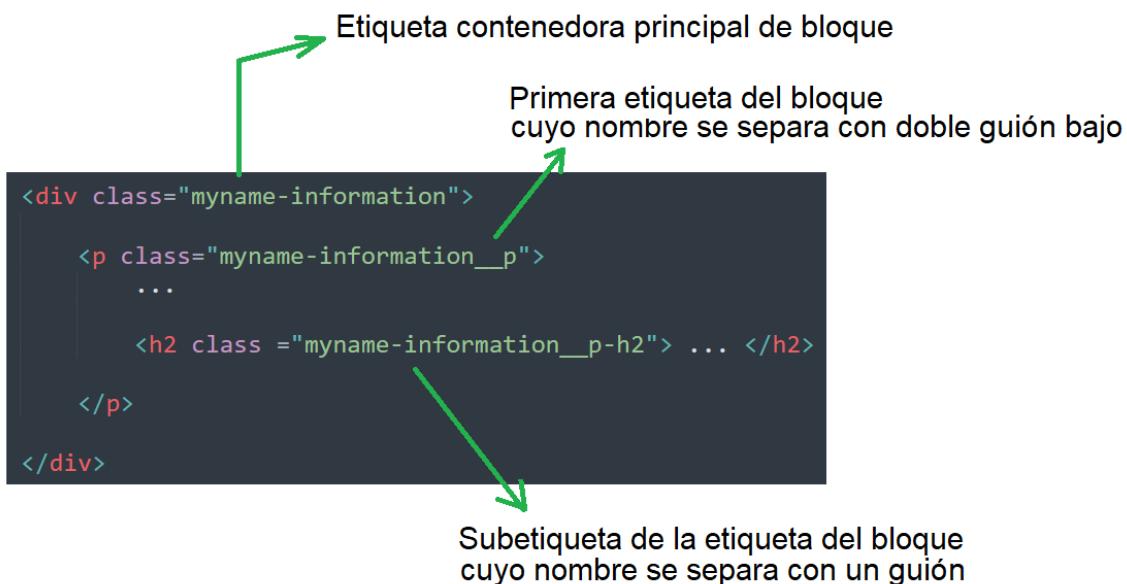
Metodología BEM

Consiste en evitar seleccionar muchos elementos y generar conflictos entre sí.

Es necesario conocer y saber qué se está seleccionando, y para ello se utilizan las clases (**class**). Dichas clases serán nombradas según su contenedor, agrupando así en bloques.

Las clases (**class**) se nombran de forma particular:

- La etiqueta contenedora principal tendrá el título que corresponda.
- La primera etiqueta dentro del contenedor principal deberá tener el título previamente establecido seguido de 2 guiones bajos y su nombre.
- Si existen más sub-etiquetas, deberán tener el nombre completo de su etiqueta contenedora seguido de 1 guión medio y su nombre.



Selectores múltiples

Se pueden escribir múltiples selectores para aplicar propiedades al mismo tiempo. Para ello, se los debe separar con comas:

```
.caja1, .caja2{
  width: 200px;
  height: 200px;
}
```



Metodología BEM con bloques de múltiples etiquetas

En un bloque se pueden encontrar varias etiquetas:

```
<div class="myname-formulario">
  <input type="text" name="myinputtxt1" class="myname-formulario__input">
  <input type="text" name="myinputtxt2" class="myname-formulario__input">
  <input type="text" name="myinputtxt3" class="myname-formulario__input">
  <input type="text" name="myinputtxt4" class="myname-formulario__input">
  <input type="text" name="myinputtxt5" class="myname-formulario__input">
  <input type="text" name="myinputtxt6" class="myname-formulario__input">
  <input type="text" name="myinputtxt7" class="myname-formulario__input">
  <input type="text" name="myinputtxt8" class="myname-formulario__input">
  <input type="text" name="myinputtxt9" class="myname-formulario__input">
</div>
```

A veces es necesario trabajar de forma específica con sólo una de ellas.

- Si la etiqueta requerida es la primera, se puede utilizar el formato *first-child*:

The screenshot shows a browser window titled "index.html". Inside the window, there is a form containing nine text input fields. The first input field, which corresponds to the element with the class ".myname-formulario__input" and the attribute "name='myinputtxt1'", has a red background color. A green arrow points from the explanatory text above to this red-highlighted input field.

- Si la etiqueta requerida puede variar a otra específica, se utiliza marcar la etiqueta con un doble guión medio seguido de la palabra "active". Con este método, se puede cambiar la etiqueta activa más adelante por medio de JavaScript.

The screenshot shows a browser window titled "index.html". Inside the window, there is a form containing four text input fields. The fourth input field, which corresponds to the element with the class ".myname-formulario__input--active" and the attribute "name='myinputtxt4'", has a red background color. A black arrow points from the explanatory text above to this red-highlighted input field.



Medidas

Existen dos unidades de medida:

- **Fijas** No son variables (píxeles, milímetros, puntos, etc.).
px mm pt
- **Relativas** Son variables y se adaptan a la caja contenedora.
Permiten un responsive design (adaptación a cualquier dispositivo).
 - REM Tiene de referencia el root.
 - EM Por defecto del navegador, **1 em son 16 píxeles**.
Una etiqueta contenedora puede definir el tamaño del em dentro de su bloque (se hereda).

```
.contact-form {  
    font-size: 25px;  
}  
  
.contact-form__h2 {  
    font-size: 1em  
}
```

Se define el valor de 1em según la medida fija establecida en la caja contenedora

1 em = 25 px

- **Viewport** Mide el tamaño de la pantalla en porcentaje.
 - Viewport Width vw Referencia a la medida horizontal.
 - Viewport Height vh Referencia a la medida vertical.

```
.contact-form {  
    width: 50vw; → 50% del tamaño horizontal  
    height: 100vh; → 100% del tamaño vertical  
}
```
- **Porcentajes** Mide el tamaño de la caja contenedora que lo contiene (forma parte de la teoría de box model).



```
selector {
    propiedad: valor;
}
```

Propiedades en CSS

Márgenes de la página web

Con el selector universal se puede alterar los márgenes de la página.

```
* {
    padding: 0px;
    margin: 0px;
}
```

Sin embargo, se puede utilizar un normalizador ya creado para corregir este y muchos otros problemas que el navegador puede darnos con su estilo por defecto. Para ello, hay que descargar **normalize.css** y luego referenciarlo en el <head> de nuestro HTML mediante la etiqueta <link>.

```
<link rel="stylesheet" type="text/css" href="normalize.css">
```

Es útil que en el archivo normalize.css se encuentren estas características:

```
img {
    border-style: none;
    max-width: 100%;
}
```

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

Caja

Se puede alterar el comportamiento de la caja de una etiqueta mediante la propiedad **display**.

- **block** La caja ocupa todo el espacio horizontal.
- **inline** La caja ocupa sólo su propio contenido, sin embargo, no se puede otorgar propiedades height ni width.
- **inline-block** La caja ocupa sólo su propio contenido pero mantiene las propiedades height ni width modificables.

```
h2 {
    display: inline;
}
```

Ancho y alto de la caja

Se puede utilizar la propiedad **height** y **width** para establecer el tamaño de la caja de una etiqueta.

Hay que tener en cuenta que, si se establece un tamaño menor al contenido de la caja, dicho contenido estará por fuera (siempre y cuando se haya agregado el **box-sizing** anteriormente explicado). **box-sizing: border-box;**

```
height: calc(100vh - 64px);
```

→ permite operaciones aritméticas



Tipografía

- **font-size** Tamaño de la letra (en píxeles, em, mm, etc.).

- **font-family** Tipografía (Arial, Calibri, Georgia, etc.).
font-family: georgia, sans-serif;
Si la primera selección no se encuentra disponible, se procede a utilizar la segunda.
 - ✓ Se puede importar fuentes de fonts.google.com y se copia la etiqueta “<link>” en el <head> de nuestro HTML. Luego se lo puede utilizar mediante el nombre dado en el mismo link, donde dice: family=**fontname**

- **font-weight** Grosor de la letra, medido del 0 al 1000+.

- **opacity** Transparencia del texto, va de 0 a 1.

Mediante la propiedad **text-align** se puede centrar el texto. **text-align: center;**

Color

Los colores en CSS se pueden establecer mediante su nombre (en inglés), código sexagesimal o RGB.

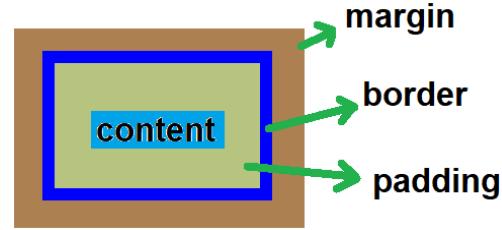
```
p {  
    color: red;  
    color: rgb(255, 0, 0);  
    color: rgba(255, 0, 0, 1.0);  
    color: #FF0000;  
    color: #FF0000FF;  
}
```

Opacidad / Transparencia



Box model

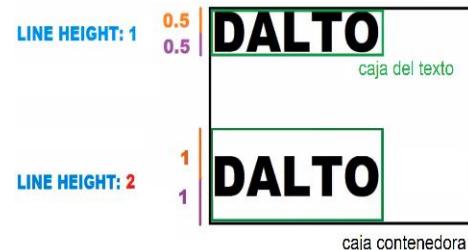
Modelo en que se trabaja con cajas. Dichas cajas tienen sus propiedades, las cuales tienen un rango de jerarquía.



1. Content

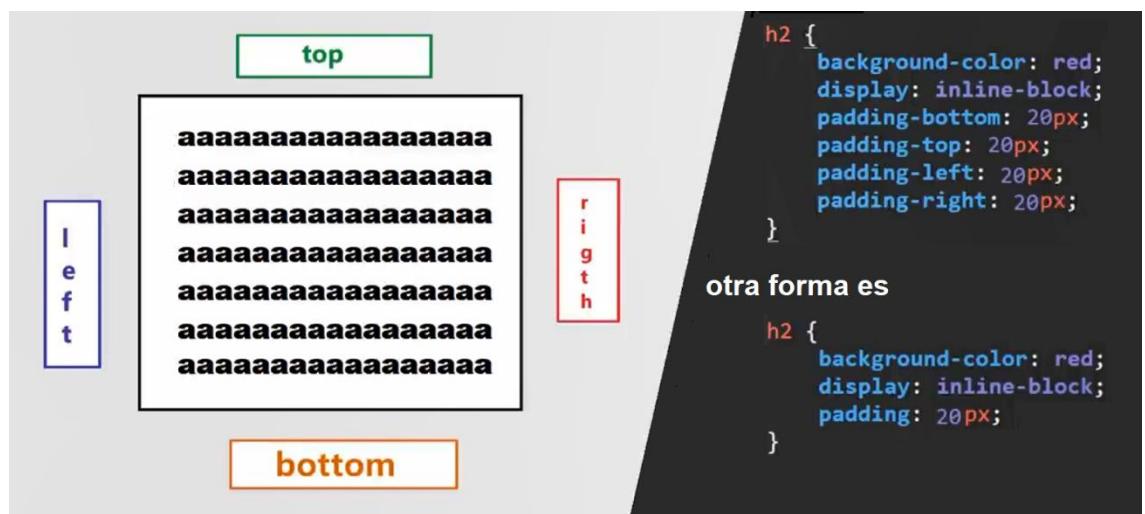
Es el contenido de la caja, por ejemplo, el texto que contiene un h1.

Se puede modificar con la propiedad **line-height**, el cual controla el tamaño que ocupa la letra en la caja que la contenga. Esta medida se hace desde el centro de la letra.



2. Padding

Distancia entre el texto (contenido/content) y la caja que lo contiene.



Otra forma acortada de escribir el padding es:





3. Border

➤ **border-radius**

Establece la suavidad de los bordes de las cajas.

```
h1 {  
background-color: blue;  
border-radius: 20px;  
}  
  
h2 {  
background-color: red;  
border-radius: 50%;  
}
```

H1

H2

➤ **border**

- *border-width*
- *border-style*
- *border-color*

Ancho del borde. sólido
Nombre del estilo (*solid*, *dashed*, *double*, *inset*, *outset*). guiones
Color (en sexagesimal). 2 líneas groove ridge

border: 1px solid #a22

Los bordes, siempre y cuando `box-sizing: border-box;` esté activo, consumirán un poco del espacio de la caja.

Cuando está activo `box-sizing: content-box;`, los bordes consumirán espacio del margin. Es decir, aumentará el tamaño de la caja.

4. Margin

Distancia entre una caja y otra. Es decir, sus márgenes.

Es necesario saber que, si se escriben dos etiquetas en renglones distintos, igualmente aparecerá un poco de margen.

Se puede aplicar a los cuatro lados de la misma forma que el padding.

```
<h2>Rancio 1</h2><h2>Rancio 2</h2>
```

Las cajas están juntas
si margin es 0

```
<h2>Rancio 1</h2>  
<h2>Rancio 2</h2>
```

Si margin es 0, igual
habrá una separación

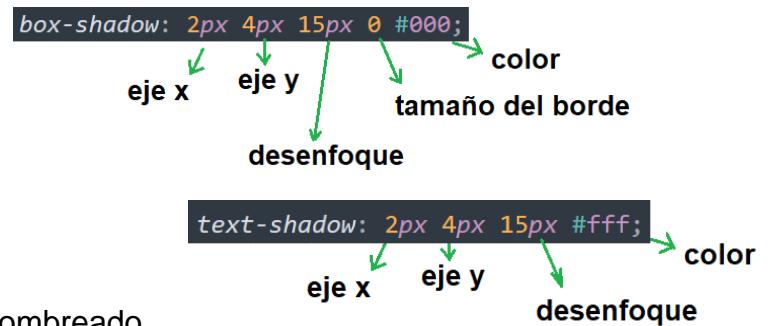
➤ **margin auto**

Centra la caja.



Sombreado

- **box-shadow**
- **text-shadow**



Si se desea aumentar la intensidad del sombreado, se puede duplicar el efecto agregando una coma y volviendo a escribir los parámetros a continuación.

Rotación

Se puede rotar una caja de etiqueta mediante la propiedad **transform: rotate**

transform: rotate(45deg); → de esta manera, se rotará 45°

Outline

Genera un borde que no afecta a otras cajas, es decir, no afecta al DOM (Document Object Model). Funciona como un resaltador y es afectado por el z-index.

Es lo que se utiliza, por defecto, en un input (edittext) de HTML cuando se selecciona.

outline: 10px solid blue; → sus características son igual a las de border

Cursor

La propiedad **cursor** cambia el cursor/puntero al estar por encima de un elemento/caja.

Ejemplo: **pointer**



Position

La propiedad **position** establece la posición de la caja de forma que cambiará el flujo de HTML, es decir, el orden de dibujado de las etiquetas.

Al alterar esta propiedad, se pueden alterar las características **top**, **left**, **bottom** y **right** de la caja, las cuales determinan su posición. Es necesario saber que top y left tienen prioridad sobre bottom y right, haciendo que estas últimas no sean tomadas en cuenta si se encuentran las primeras.

- **static** No tiene definida una posición. Así es por defecto.
- **relative** Guarda la posición de la caja en el DOM, permitiendo modificar la posición de la caja sin que otra caja pueda ocupar su espacio original.
- **absolute**
 - ✓ Elimina la posición de la caja en el DOM, es decir, ya no tendrá espacio reservado y otra caja puede ocuparlo.
 - ✓ Permite modificar la posición absoluta de la caja en todo el viewport (pantalla) o en su contenedor si es que está posicionado.
 - ✓ Si no se define posición de la caja (top, left, etc.), se posiciona por defecto donde debería estar según su etiqueta contenedora.
 - ✓ El ancho de la caja se ajusta al contenido (inline).
- **fixed** Es igual que *absolute* pero queda fijado a la pantalla. Suele aplicarse en menús y publicidades, donde el contenido de la página puede moverse pero la publicidad queda estática en la pantalla.

Para evitar que la caja se superponga con otras cajas, es recomendable agregar **padding-top** al <body> del HTML para que ninguna caja ocupe el espacio que ocupará esta caja. Esto provocará que la caja fixed también se mueva hacia abajo, para solucionarlo, se puede agregar un **margin** negativo del mismo tamaño que el establecido en el padding-top.

```
body {  
    padding-top: 100px  
}  
  
.caja-fixed {  
    position: fixed;  
    margin: -100px  
}
```



➤ **sticky**

Mezcla entre *relative* y *fixed*. La particularidad que da es que se queda fijo en pantalla pero no necesariamente de forma constante, sino que sólo cuando su DOM (el cual conserva) es alcanzado. Para ello es necesario utilizar la propiedad *top*.

Por ejemplo: Si colocamos una caja sticky a la mitad de una página web, cuando llegamos a la mitad de la página y seguimos bajando, a partir de ese momento la caja sticky se quedará fija en pantalla.

```
.caja-sticky {  
    position: sticky;  
    top: 0;  
    margin-top: 10px;  
}
```

Z-index

Es la posición en el eje Z. Esto determina qué caja se muestra por encima de otra. Sólo funciona cuando la caja está posicionada.

Si existiera una caja en la misma posición que otra caja, se mostrará la que tenga un Z-index mayor. En caso de tener un mismo Z-index, se mostrará la que fue dibujada última en HTML.

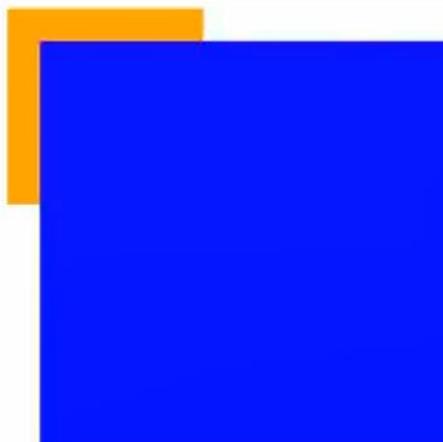
```
z-index: 0;
```

Por defecto, Z-index es cero.

- Existe un conflicto cuando se trata de cajas padre-hijo, es decir, de una etiqueta contenedora (padre) con una etiqueta dentro (hijo).

Para que la caja padre quede por encima de la caja hijo, se deben posicionar a ambas en *relative*, a la caja hijo darle z-index -1 y a la caja padre no definirle ningún z-index.

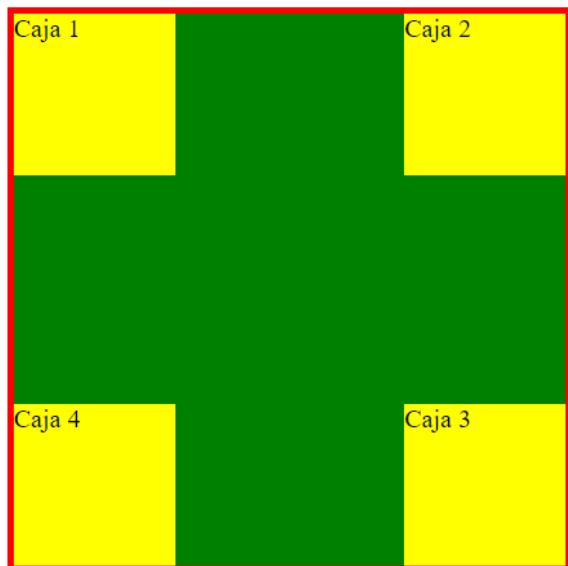
```
.contenedor {  
    width: 300px;  
    height: 300px;  
    background: blue;  
    margin: 40px;  
    position: relative;  
}  
  
.hijo {  
    background: orange;  
    height: 120px;  
    width: 120px;  
    position: relative;  
    top: -20px;  
    left: -20px;  
    z-index: -1  
}
```





HTML

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>index.html</title>
7      <link rel="stylesheet" type="text/css" href="normalize.css">
8      <link rel="stylesheet" type="text/css" href="estilo.css">
9  </head>
10 <body>
11     <div class="contenedor">
12         <div id="caja1">Caja 1</div>
13         <div id="caja2">Caja 2</div>
14         <div id="caja3">Caja 3</div>
15         <div id="caja4">Caja 4</div>
16     </div>
17
18 </body>
19 </html>
```



CSS

```
1  * { box-sizing: border-box;
2  padding: 0;
3  margin: 0;
4  }
5
6  div {
7      position: absolute;
8      display: block;
9  }
10
11 div div {
12     height: 100px;
13     width: 100px;
14     background: yellow;
15 }
16
17 .contenedor {
18     background: green;
19     position: relative;
20     border: 4px solid red;
21     margin: 50px auto;
22     height: 450px;
23     width: 450px;
24 }
25
26 #caja1 {
27 }
28
29 #caja2 {
30     right: 0;
31 }
32
33 #caja3 {
34     right: 0;
35     bottom: 0;
36 }
37
38 #caja4{
39     bottom: 0;
40 }
```

Nota: Si se establecen todas las posiciones (top, right, bottom y left) en 0 y el margin en auto, la caja se centrará.



Display

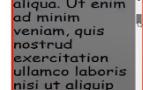
La propiedad **display** modifica el comportamiento de las cajas.

- **block** La caja ocupa todo el espacio horizontal disponible.
 - **inline** La caja ocupa sólo su contenido. Usado en textos.
 - **inline-block** Se comporta como un *inline* pero manteniendo la propiedad de poder modificar el tamaño de la caja con *height* y *width*.
 - **none** La caja se oculta completamente.
 - **list-item** Se comporta como una lista, tal como se vió en HTML.
en desuso
 - **table** Se comporta como una tabla de Excel, cuya caja ocupa todo el espacio horizontal.
 - **inline-table** Se comporta como una tabla de Excel inline pero la caja sólo ocupa el contenido.
 - **table-cell** Se comporta como una celda de una tabla de Excel.
 - **table-row** Se comporta como una fila de una tabla de Excel.
 - **table-column** Se comporta como una columna de una tabla de Excel.
-
- **flex** Se comporta como *block* pero no el contenido.
 - **grid** Se comporta como block pero no el contenido.
 - **inline-grid**
 - **inline-flex**



Overflow

El overflow es la barra de scroll que permite trasladar una pantalla o un contenedor de, por ejemplo, texto.

- **visible** Valor por defecto, muestra el texto sobrante por fuera de la caja que lo contiene.
- **auto** Agrega la clásica barra de scroll lateral, sólo si es necesario.
- **scroll** Agrega la barra de scroll incluso si no es necesario.
- **hidden** Fuerza a que no se muestre la barra de scroll y se oculta el contenido que salga de la caja.

lorem ipsum
dolor sit amet,
consectetur
adipiscing elit,
sed do eiusmod
tempor
incididunt ut
labore et dolore
magna aliqua. Ut
enim ad minim
veniam, quis
nostrud
exercitation
ullamco laboris
nisi ut aliquip
ex ea commodo
consequat. Duis
aute irure dolor
in reprehenderit
in voluptate velit
esse cillum dolore
eu fugiat nulla
pariatur. Excepteur
sint occaecat
cupidatat non
proident, sunt
in culpa qui
officia deserunt
mollit anim id est
laborum.

```
html {  
    scroll-snap-type: mandatory;  
}
```

Float

Su uso es obsoleto, sin embargo, se puede utilizar para lograr un buen efecto visual cuando se trabaja con una imagen y texto (al cual se le ajusta el *width* en porcentaje) en un mismo contenedor.

```
<div class='cont'>  
      
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod  
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,  
    quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo  
    consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse  
    cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non  
    proident, sunt in culpa qui officia deserunt mollit anim id est laborum.  
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,  
    quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo  
    consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse  
    cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non  
    proident, sunt in culpa qui officia deserunt mollit anim id est laborum.  
</div>
```

```
.cont {  
    margin: auto;  
    margin-top: 50px;  
    border: 4px solid red;  
    background: grey;  
    width: 50%;  
}  
.cont img {  
    float: right;  
    width: 300px;  
}
```





Imágenes

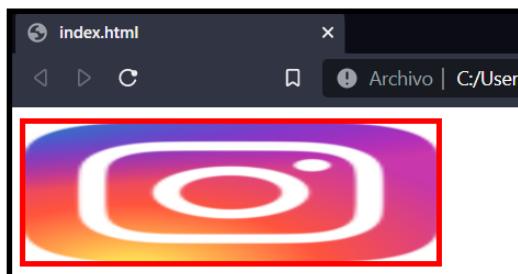
Object-fit: Fill

Se utiliza para trabajar con imágenes y modificar su resolución.

```
<div class="contenedor" style="outline: 4px solid red" style="z-index=2">
    
</div>
```

```
.contenedor {
    display: inline-block;
    margin: 2em;
    width: 300px;
    height: 100px;
}

.contenedor img {
    width: 100%;
    height: 100%;
```



Al modificar el tamaño de una imagen, esta se expande o reduce, muchas veces deformando la imagen.

➤ Contain

Las resoluciones de la imagen se ajustan a la caja, dejando espacios vacíos donde la imagen no llega ya que mantiene las proporciones originales.



➤ Cover

La imagen se ajusta al contenedor, recortando el sobrante.



➤ None

Mantiene la resolución original de la imagen. Si la imagen es más grande que el contenedor, sólo se mostrará lo que quepa de la imagen. Si la imagen es más pequeña que el contenedor, se mostrará toda la imagen en su tamaño real dejando espacio en blanco en el sobrante del contenedor.

```
.contenedor img {
    width: 100%;
    height: 100%;
    object-fit: none;
```

➤ Scale-Down

Elije el valor más bajo de resolución resultante entre *none* y *contain*.

Object-Position

Permite posicionar una imagen, ya sea fijándola en una dirección o moviéndola de forma completa con medidas.

- Direcciones (*top, right, bottom, left*)
- Medidas (*px, em, %, ...*)

```
.contenedor img {
    width: 100%;
    height: 100%;
    object-fit: cover;
    object-position: top;
```





Background

- **background-color** Establece un color de fondo a un elemento.
- **background-image: url()** Establece una imagen (la establecida en la url) sin respetar los tamaños de la imagen.
Si la imagen es más grande, se verá recortada.
Si la imagen es más pequeña, se mostrará repetida (a menos que se utilice la propiedad **background-repeat: no repeat;**, en dicho caso, el espacio sobrante será del color establecido en *background-color*.)
- **background-size** Escala una imagen establecida. 
`background-size: 100% 360px`
Se puede utilizar *cover*, etc.
- **background-clip** Establece desde dónde se mostrará la imagen.
- **background-origin** Establece desde dónde se origina la imagen.
 - border-box
 - padding-box
 - content-box
- **background-position** Establece la posición de la imagen en el background.
`background-position: right bottom;`
- **background-attachment**
 - scroll la imagen como fondo del elemento (por defecto).
 - fixed imagen queda fijada al fondo de la página, incluso si se desplaza por la página.
 - inherit



Iconos

Se pueden añadir íconos por medio de fontawesome.com

Para empezar a añadir íconos, es necesario copiar el **script** en el **<head>** de nuestro HTML. Por ejemplo:

```
<script src="https://kit.fontawesome.com/62ea397d3a.js" crossorigin="anonymous"></script>
```

Dichos íconos se añaden utilizando un nombre de **class** específico.

```
<body>
  <header>
    <nav class="nav">
      <ul class="nav__ul">
        <li class="nav__ul-li-home"> <i class="fas fa-home"></i> <a href="https://google.com"> Inicio </a> </li>
      </ul>
    </nav>
  </header>
```

Si el nombre de una class tiene un espacio, entonces son nombres separados, es decir, esta etiqueta tiene 2 .

Estos son los nombres específicos de class que se requiere para añadir un ícono de fontawesome.com

Este será el ícono



The screenshot shows a browser window with a title bar labeled "index.html". The main content area displays the word "Inicio" next to a small house icon. A white arrow points from the text "Este será el ícono" to the house icon in the browser.

Otra página con muchos íconos disponibles que no requiere un **script**, es tablericons.com



Pseudo-elementos

Los pseudo-elementos permiten sólo cambios visuales (como los outline) y se crean sobre un elemento.

- ⊕ **::first-line** *no funciona en inline* Selecciona la primera línea.
- ⊕ **::first-letter** *no funciona en inline* Selecciona la primera letra.

The diagram illustrates the application of the ::first-line pseudo-class. On the left, a code editor shows an HTML file with a

element containing "lorem ...". Below it is a CSS rule ".texto::first-line { color: blue; }". A green arrow points from this code to a browser window on the right. The browser window displays the text "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor" where the first line is colored blue. The browser status bar indicates the window is titled "index.html" and has dimensions "399.20px x 661.60px".

Afecta sólo la primera línea, sin importar el tamaño de la página (responsive)

- ⊕ **::placeholder** Permite añadir textos “de ayuda”.
Por ejemplo: en un input.

The diagram illustrates the ::placeholder pseudo-class. On the left, a code editor shows an HTML form with an input field and some CSS rules for styling it. The CSS includes rules for the form container and the input field itself. Three annotations with arrows point from the text to their respective effects: "Caja centrada con altura de 100 px" points to the margin and padding rules for the form; "Ancho es la mitad del contenedor (pantalla)" points to the width rule for the input; and "Ocupa todo el espacio que puede" points to the width and height rules for the input. A green arrow points from this code to a screenshot of a browser on the right. The screenshot shows a dark green page with a white input field containing the placeholder text "Type text here". A red annotation with an arrow points to the input field with the text "Cuando el usuario tipea, ese texto desaparece".

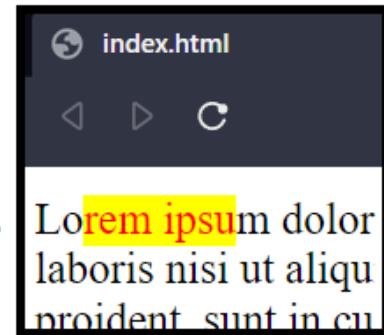


➊ ::selection

Permite cambiar las propiedades de la selección de texto.

```
<body>
  <p class="texto">
    lorem ...
  </p>
</body>
```

```
p::selection {
  color: red;
  background: yellow;
}
```



➋ ::after

HIJOS – CONTENT (necesario) – INLINE

➋ ::before

HIJOS – CONTENT (necesario) – INLINE

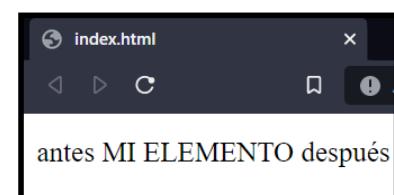
Son hijos del elemento al cual se aplican.

No afectan al DOM (su texto no se puede seleccionar).

Se pueden agregar imágenes.

```
<body>
  <p>
    MI ELEMENTO
  </p>
</body>
```

```
p::before {
  content: "antes";
}
p::after {
  content: "después";
}
```



Un buen estilo de diseño puede ser un *degradé* de color.

Establece un color de fondo con degradé

```
background: linear-gradient(to bottom, transparent, #222);
```

hacia dónde se realiza el degradé

color de inicio

color final



Pseudo-clases

Funcionan como los pseudoelementos y tiene la característica de “listener”.

✿ :hover

Detecta si el puntero está encima de un elemento (o bien si es pulsado cuando se trata de versión mobile).

```
<div>  
  Div de prueba  
</div>  
  
div {  
  height: 200px;  
  width: 200px;  
  background: slateblue;  
}  
  
div:hover {  
  background: red;  
}
```



* la propiedad **transition** permite cambiar la velocidad de transición.

```
.caja1 {  
  height: 200px;  
  width: 200px;  
  background: slateblue;  
  transition: background 1s, height 1s, width 1s  
}  
  
.caja1:hover {  
  background: darkslateblue;  
  height: 300px;  
  width: 300px;  
}
```

Se puede añadir que, al realizar un hover (posicionar el puntero por encima de un elemento), se realice una acción sobre un sub-elemento (elemento hijo).

```
.myclass:hover > subelement {  
  . . .  
}
```

Cuando se active el hover en el elemento **.myclass**, se aplicará la configuración deseada (. . .) sobre el **subelemento** dentro de **.myclass**



- ⊕ **:link** Cambia un link que aún no se visitó.
- ⊕ **:visited** Cambia un link que ya fue visitado.

The screenshot shows a code editor with the following CSS:

```
<a href="https://google.com" target="_BLANK">  
    Mi Link no visitado  
</a>
```

And the following CSS rule:

```
a:link {  
    color: red;  
}
```

A browser window titled "index.html" displays the text "Mi Link no visitado" in red. A green arrow points from the word "Link" in the text to the "color: red;" part of the CSS rule. Another green arrow points from the text "abre una nueva pestaña" to the "target='_BLANK'" attribute in the HTML code.

text-decoration: none; /* quita el subrayado del link */

- ⊕ **:active** Se activa cuando se presiona el botón del mouse sobre la etiqueta con esta pseudoclase.

- ⊕ **:valid** Selecciona todos los inputtext cuyo contenido sea válido.

- ⊕ **:focus** Se activa cuando algo está “seleccionado” esperando una escritura, por ejemplo, un input (edittext).

- ⊕ **:lang** Cambia el texto que esté determinado en un idioma específico.

The screenshot shows a code editor with the following HTML:

```
<div class="contenedor">  
    <b lang="en"> Text in english </b>  
    <b lang="es"> Texto en español </b>  
</div>
```

And the following CSS rule:

```
b:lang(en) {  
    color: red;  
}
```

A browser window titled "index.html" displays the text "Text in english" in red and "Texto en español" in black. A blue arrow points from the "Text in english" text to the "color: red;" part of the CSS rule.

- ⊕ **:first-child** Funciona como un selector, y selecciona el primer sub-elemento de un elemento.

The screenshot shows a code editor with the following CSS:

```
.grid-item:first-child {  
    background: red;  
}
```

And the following HTML:

```
<div class="grid-container">  
    <div class="grid-item">1</div>  
    <div class="grid-item">2</div>  
    <div class="grid-item">3</div>  
</div>
```

A red arrow points from the ".grid-item:first-child {" part of the CSS to the first "div" element in the HTML code.

- ⊕ **:nth-child(X)** Funciona como un selector, y selecciona el sub-elemento número X de un elemento.



Responsive Design – Mobile First

El Responsive Design permite trabajar con distintas resoluciones, es decir, adaptar un diseño a varias resoluciones para que pueda verse correctamente en diversos dispositivos.

El Mobile First es la adaptación de un contenido de mobile a resoluciones de páginas web de PC o de Tablet. Esta es la forma más actual (2022) de trabajar el Responsive Design.

Es importante que en el `<head>` de nuestro HTML tengamos la etiqueta:

```
<meta name="viewport" content="width=device-width, initial-scale=1">  
  @media only screen and(max-width: 800px) {  
    ...  
  } → selectores
```

Cuando la pantalla tenga un máximo de 800px de ancho...

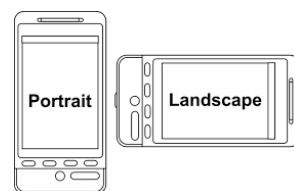
¡Tener en cuenta el rango de especificidad!

Layout

Distribución de los elementos en la página.

Orientación de pantalla

- **Landscape** Mas ancho que largo.
- **Portrait** Mas largo que ancho.





```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>index.html</title>
    <link rel="stylesheet" type="text/css" href="normalize.css">
    <link rel="stylesheet" type="text/css" href="estilo.css">
    <script src="https://kit.fontawesome.com/62ea397d3a.js" crossorigin="anonymous"></script>
</head>

<body>
    <header>
        <nav>
            <!-- Nuestra ul normal -->
            <ul class="nav">
                <li class="nav_li"> <i class="fas fa-home"></i> <a href="https://google.com"> Inicio </a> </li>
                <li class="nav_li"> <i class="fas fa-user-friends"></i> <a href="https://facebook.com"> Sobre nosotros </a> </li>
                <li class="nav_li"> <i class="fas fa-dollar-sign"></i> <a href="https://tiktok.com"> Gana dinero </a> </li>
                <li class="nav_li"> <i class="fas fa-question-circle"></i> <a href="https://instagram.com"> Ayuda </a> </li>
            </ul>

            <!-- Nuestra ul responsive -->
            <ul class="navr">
                <div class="navr__menu-container">
                    <div class="navr__menu-button fas fa-bars"> </div>
                </div>
                <div class="navr__li-container">
                    <li class="navr_li"> <i class="fas fa-home"></i> <a href="https://google.com"> Inicio </a> </li>
                    <li class="navr_li"> <i class="fas fa-user-friends"></i> <a href="https://fb.com"> Sobre nosotros </a> </li>
                    <li class="navr_li"> <i class="fas fa-dollar-sign"></i> <a href="https://tiktok.com"> Gana dinero </a> </li>
                    <li class="navr_li"> <i class="fas fa-question-circle"></i> <a href="https://instagram.com"> Ayuda </a> </li>
                </div>
            </ul>
        </nav>
    </header>

    <section>
    </section>

    <footer>
    </footer>

</body>

</html>
```



```
* {  
    box-sizing: border-box;  
    padding: 0;  
    margin: 0px;  
    font-weight: 100;  
}  
  
.navr {  
    display: none; /* por defecto se oculta el nav responsive */  
}  
  
/* Estilo inicial */  
/*************/  
.nav {  
    background: slateblue;  
}  
  
.nav_li { /* elementos del menú */  
    display: inline-block; → para que se muestren los links uno al lado del otro  
    padding: 12px; → separa el texto de los márgenes  
}  
  
.nav_li a { /* links */  
    text-decoration: none; → quita el subrayado por defecto de los links  
    color: #fff;  
}  
  
.nav_li i { /* íconos */  
    box-sizing: border-box;  
    text-align: right;  
    margin-right: 5px;  
    color: #fff;  
}  
  
/* Animación del nav */  
/*************/  
.nav_li:hover > a { → colorea el link cuando el puntero está encima  
    color: #ccc;  
}  
.nav_li:hover > i { → colorea el ícono cuando el puntero está encima  
    color: #ccc;  
}
```



```
/* Con la configuración dada hasta ahora, cuando el tamaño de pantalla llega a 500px de ancho,  
los botones comienzan a posicionarse debajo, dando un mal diseño */
```

```
.navr__menu-button {  
    /*float: right;*/  
    text-align: right;  
    width: 100%;  
    padding: 6px 20px;  
    font-size: 25px; /* cambia tamaño del icono */  
}
```

Tener en cuenta que la barra con el menú medirá 37px
(6px x2 de padding + 25px de font-size)

```
@media only screen and (max-width: 500px) {
```

```
.nav {  
    display: none; /* oculta el nav sin responsive design */  
}
```

```
nav {  
    height: 37px;  
    width: 100%;  
}
```

```
.navr {  
    display: block; /* muestra el nav con responsive design */  
    position: absolute; /* sin DOM */  
    width: 100%;  
    margin-top: 37px;  
    padding-top: 27px;  
    height: 37px;  
}
```

para separar un poco más el primer elemento del menú
(nota: esto hace que el tamaño total del nav sea de 64px en total)

```
.navr__li-container { /* contenedor de los elementos del menú */  
    position: relative; /* depositar */  
    top: -150vh;  
    transition: all 1s;  
    height: calc(100vh - 64px);  
    background-color: #9bf;  
    z-index: 1;  
    width: 75%;  
    float: right;  
}
```

para quitarlo de la pantalla (no se verá)

cuando haga la animación de mostrarse, lo hará suave

se resta el tamaño del menú (37px) + el espacio para separar el primer elemento del menú (27 px)

```
.navr__menu-container { /* contenedor del botón menú */  
    position: absolute;  
    width: 100%;  
    font-size: 25px;  
    padding: 6px 20px;  
    background-color: #69c;  
    margin-top: -64px;  
    z-index: 100;  
}
```

```
.navr:hover > .navr__li-container { /* ANIMACION */  
    top: -20px;  
}
```

regresa el contenedor de los elementos del menú a su posición original
para que se pueda visualizar como un menú desplegable

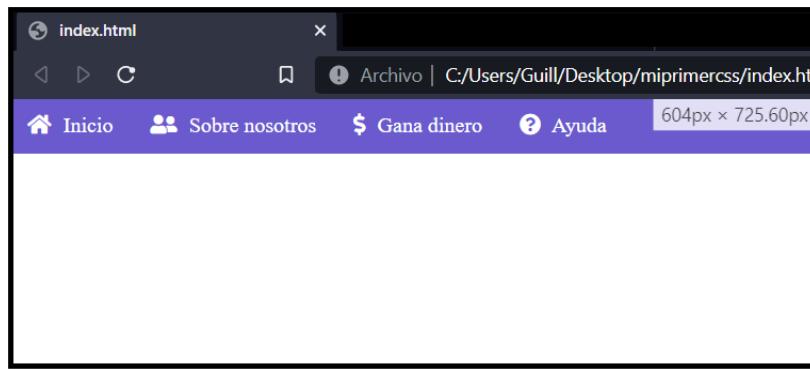


```
.navr_li /* elementos del menú responsive */  
width: 100%;  
padding: 12px 20px;  
list-style: none;  
}  
  
.navr_li i /* iconos del menú responsive */  
box-sizing: border-box;  
text-align: center;  
margin-right: 5px;  
width: 30px;  
color: #016;  
}  
  
.navr_li a /* links del menú responsive */  
color: #016;  
text-decoration: none;  
}
```

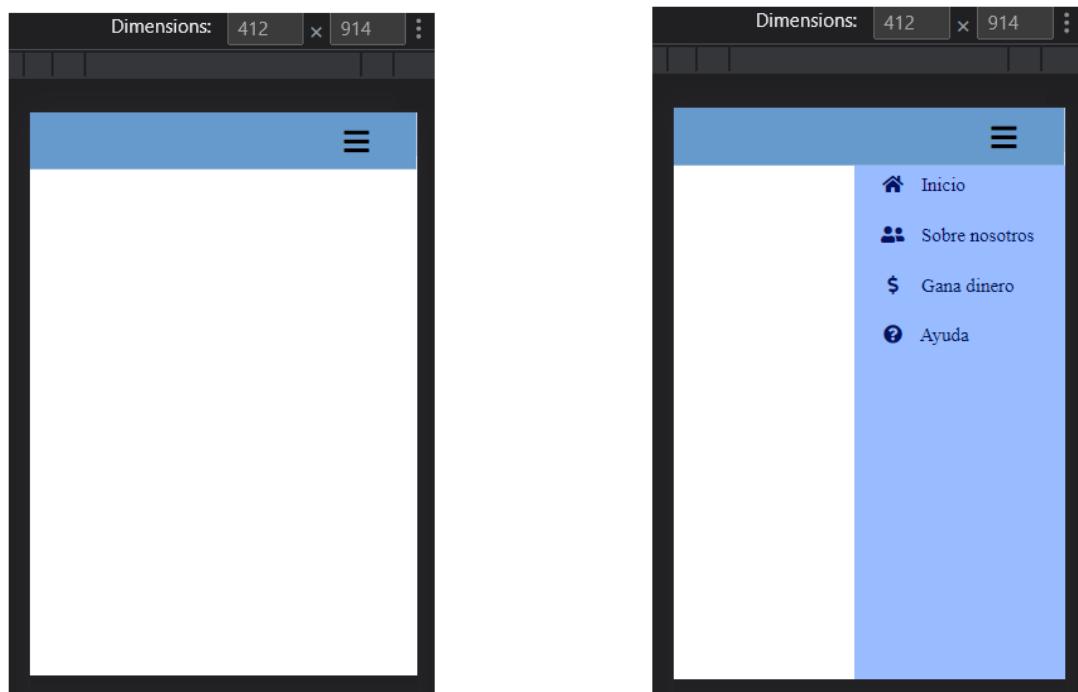
Diseño de los elementos del menú responsive (links, íconos y list-items)

elimina los ítems por defecto del list-item

De esta manera, nuestra página web quedará de esta forma:



Y, si el tamaño de ancho de la pantalla fuera menor a 500 píxeles, quedará de la siguiente manera.





Flex-Box

Requiere de dos componentes, el *flex-container* y el *flex-item*.

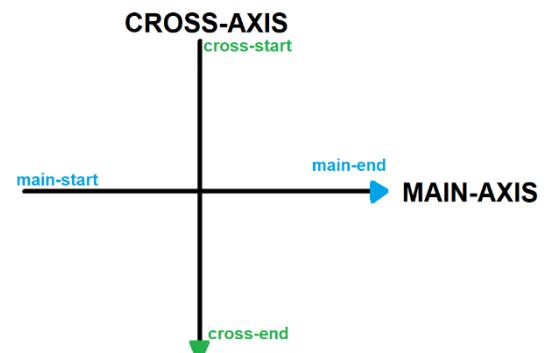
El *flex-container* se comporta como un bloque y es el que va a contener todos los elementos (hijos) denominados flex-item. Dicho contenido es el que tendrá las características de flexibilidad que ofrece Flex-box.

Los sub-elementos (hijos de los flex-item) no se consideran flex.

```
<div class="flex-container">
  <p class="flex-item">
    Yo soy un flex item. <b> Pero yo no lo soy </b>
  </p>
</div>
```

.flex-container {
 display: flex;
}

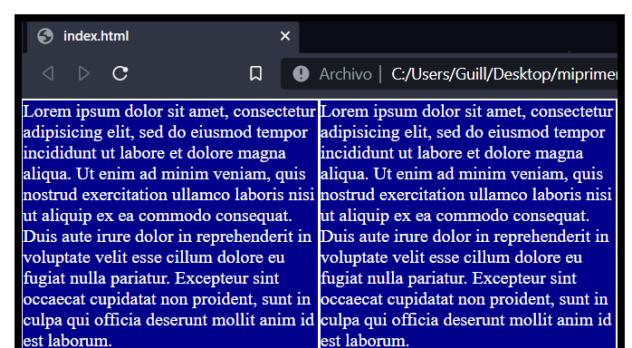
- Ejes del Flex-box
 - Main axis (eje X)
 - Cross axis (eje Y)
- Direcciones del Flex-box
 - Main-start → Main-end
 - Cross-start → Cross-end



Con esto sabemos que si agregamos varios flex-items, se posicionarán uno al lado del otro siguiendo el main-axis.

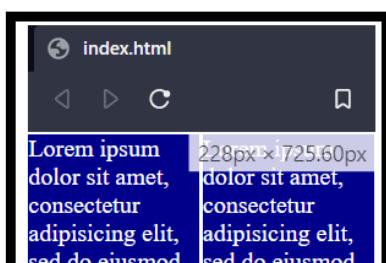
```
<div class="flex-container">
  <p class="flex-item">
    lorem
  </p>
  <p class="flex-item">
    lorem
  </p>
</div>
```

.flex-container {
 display: flex;
}
.flex-item {
 background: darkblue;
 color: white;
 margin: 1px;



No solo se posicionan uno al lado del otro, sino que mantienen la misma altura sin importar el contenido.

Además, el tamaño de las cajas se ajustan al tamaño de la pantalla / contenedor.





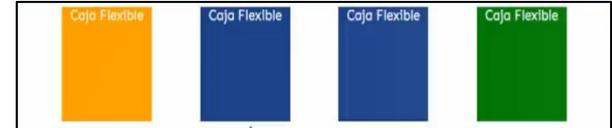
Propiedades de flex-container

Las propiedades del contenedor flex afectan a los ítems.

- **flex-direction** Permite cambiar la dirección del contenido.
 - **row** Los ítems se comportan como filas (se alinean en el main-axis). Propiedad por defecto.
 - **row-reverse** El main-axis cambia de sentido hacia la izquierda.
 - **column** Los ítems se comportan como columnas. El main-axis pasa a ser el Eje Y.
 - **column-reverse** Los ítem se comportan como columnas con sentido de abajo hacia arriba.
 - **order**
- **flex-wrap**
 - **wrap** Establece que, si hay definido un ancho/alto específico con *width* o *height*, ese tamaño no se altere. Cuando el tamaño de pantalla/contenedor no alcance para mostrar un flex-item, lo coloca en la siguiente fila / columna.
 - **wrap-reverse** Cuando un item no alcanza a mostrarse en pantalla, se acomoda en la fila / columna anterior.
 - **no-wrap** Propiedad por defecto. Los ítems intentan acomodarse al espacio de la pantalla / contenedor ajustando el tamaño de dichos ítems.
- **justify-content** Permite alinear ítems en el main-axis.
 - **center** Los ítems del flex-container se centran.
 - **space-around** Los ítems del flex-container se acomodian de forma que utilizan todo el espacio del main-axis disponible, con un mismo margen entre las paredes del contenedor / pantalla y entre los ítems (donde se apreciará como doble margen). Igual a `margin: auto;`



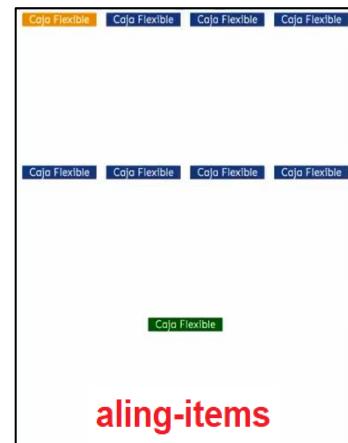
- **space-between** Los ítems del flex-container se acomodan con la misma distancia entre ítems, pero no respetan la distancia con las paredes del contenedor / pantalla.
- **space-evenly** Igual que *space-around*, pero el tamaño del margen entre las paredes del contenedor / pantalla y entre los ítems se calcula para que sean absolutamente iguales en el main-axis.



Para alinear en el cross-axis, se utiliza:

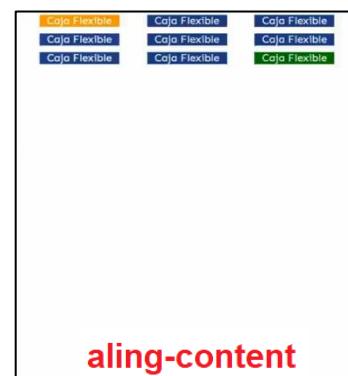
- **aling-items** (*cuando hay sólo una línea de ítems*) o **aling-content**
 - **stretch** Propiedad por defecto. Si hay un *height* establecido en el flex-container, las cajas intentarán ocupar todo el espacio disponible verticalmente (cross-axis).
 - **flex-start** Los ítems se posicionan al inicio del cross-axis, sin tender a ocupar todo el espacio vertical disponible. Es decir, las cajas se ajustan a su contenido.

aling-items
 - **center** Los ítems se centran en el cross-axis.
 - **flex-end** Los ítems se posicionan en el cross-end.
 - **baseline** Se utiliza cuando se quiere realizar un wrap reverse empezando desde el cross-end, para así lograr el mismo efecto que se muestra a la derecha pero desde debajo.



```
.flex-container {  
    display: flex;  
    flex-wrap: wrap-reverse;  
    justify-content: space-evenly;  
    height: 100vh;  
    align-content: baseline;  
}
```

ocupá toda la pantalla



aling-content

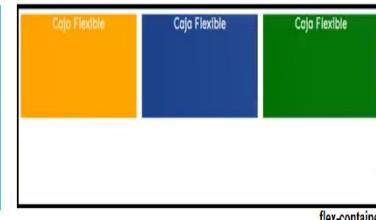


Propiedades de los flex-item

Es importante saber que, en un flex-item, la propiedad **margin** está invertida, es decir, se comporta al revés que en un contenedor normal. Esto quiere decir que la izquierda es la derecha, el top es el bottom, etc.

- **align-self** Alineación de un ítem específico en el cross-axis, cuyos valores posibles son los mismos que en *align-content* del flex-container.
- **flex-grow** Toma el espacio sobrante horizontal de las cajas de la línea y aplica dicho espacio a las cajas para llenar ese espacio.

```
.flex-item {  
background-color: #248;  
color: #fff;  
margin: 5px;  
min-width: 120px;  
text-align: center;  
height: 100px;  
flex-grow: 1  
}
```



De esta manera, se puede establecer un flex-container cuyos ítems se estiren ocupando todo el espacio horizontal y que, al achicar la pantalla hasta un *min-width* establecido, las cajas pasen a estar por debajo.

El flex-grow se puede establecer a una sola caja de toda la línea, entonces dicha caja será la única en redimensionarse.

```
.flex-container {  
display: flex;  
height: 100vh;  
flex-wrap: wrap;  
align-content: flex-start;  
}  
  
.flex-item {  
min-width: 100px;  
min-height: 100px;  
margin: 6px;  
flex-grow: 1;  
}
```

- **flex-shrink** Establece qué tanto espacio cederá cuando el contenedor donde se encuentra se achique (1 por defecto). Es decir, si una caja tiene flex-shrink 1 y otra caja tiene flex-shrink 2, la segunda caja cederá el doble de su tamaño cuando se reduzca para adaptarse al contenedor achicado.

Si hay un width establecido, todas las cajas llegarán a dicho width al mismo tiempo mientras se manipula el tamaño del contenedor.

Se pueden utilizar decimales.

Si se establece en 0, no cederá espacio.

- **flex-basis** Similar a *width* pero con mayor prioridad.
- **flex** Shorthand del flex.
- **order** Similar a z-index pero en el eje en que apunta el main-axis. Es decir, el ítem que tenga el valor más grande, estará en el main-end.

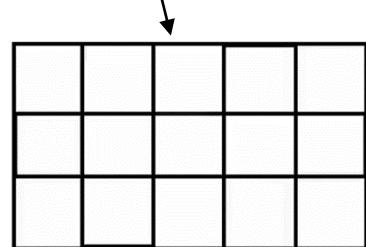
flex: 1 1 125px;
flex-grow flex-shrink flex-basis



Grid

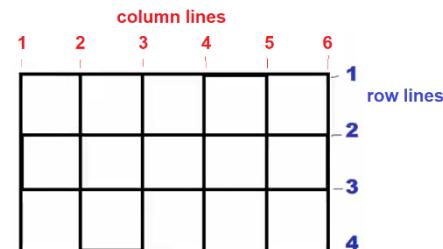
Valor de la propiedad *display*, es un estilo de layout que permite trabajar con grillas.

Funciona a partir de celdas, columnas, filas (row), tracks, áreas.



Conceptos básicos

- **Grid container** Bloque contenedor de la grilla (tabla).
- **Grid item** Elementos (sólo hijos directos) dentro del grid container.
Nota: No necesariamente son las celdas.
- **Grid cell** Celdas de la grilla.
- **Grid tracks**
 - **column** columnas verticales
 - **row** filas horizontales
- **Grid area** Conjunto de celdas no necesariamente consecutivas (definidas por el dev).
Ej: un conjunto de 2x2 celdas, o una fila completa.
- **Grid line**
 - **column line** líneas divisorias de las columnas.
 - **row line** líneas divisorias de las filas.

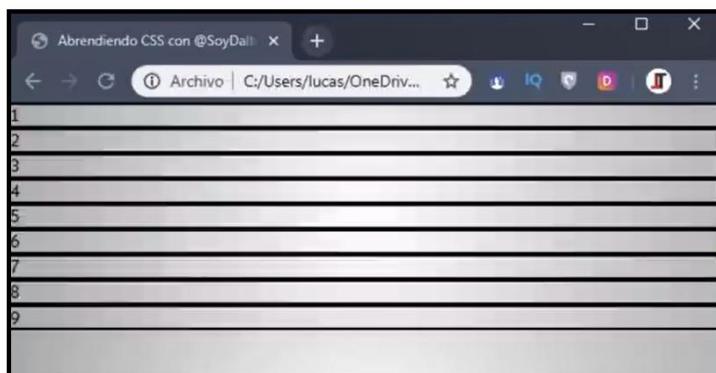


```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

```
body {
  background: radial-gradient(circle, #fff, #bbb);
}

.grid-container {
  display: grid;
}

.grid-item {
  border: 2px solid #000;
}
```



Al crear un grid-container, éste se comportará como un bloque. Por defecto, creará una sola columna y cada grid-item será una fila, denominados grid implícito (porque no lo tenemos programado).



Propiedades del grid-container

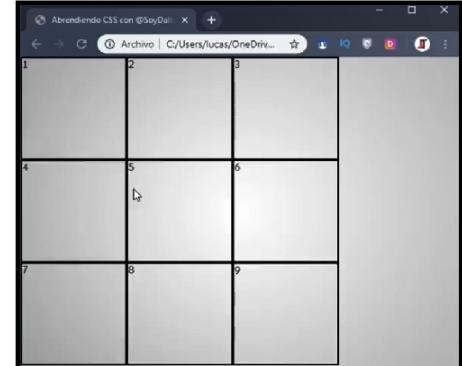
- **grid-template-rows** Establece la cantidad de filas del grid.
- **grid-template-columns** Establece la cantidad de columnas del grid.

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

```
.grid-container {
  display:grid;
  grid-template-rows: 150px 150px 150px;
  grid-template-columns: 150px 150px 150px;
}

.grid-item {
  border: 2px solid #000;
}
```

medidas de cada columna y cada fila



➤ Medidas

- Exactas: píxeles, etc.
- Flexibles:
 - ✓ **fr** el ancho de columna / alto de fila se ajustará al de la página.
 - ✓ **max-content** el ancho se ajusta a todo el contenido de la celda.
 - ✓ **min-content** el ancho se ajusta al tamaño mínimo del contenido (una palabra).
 - ✓ **minmax()** establece el tamaño entre dos parámetros (mínimo y máximo).

```
grid-template-columns: minmax(min-content, 200px);
```

➤ Función *repeat()*

Define varias veces un mismo valor, para no escribir varias veces lo mismo.

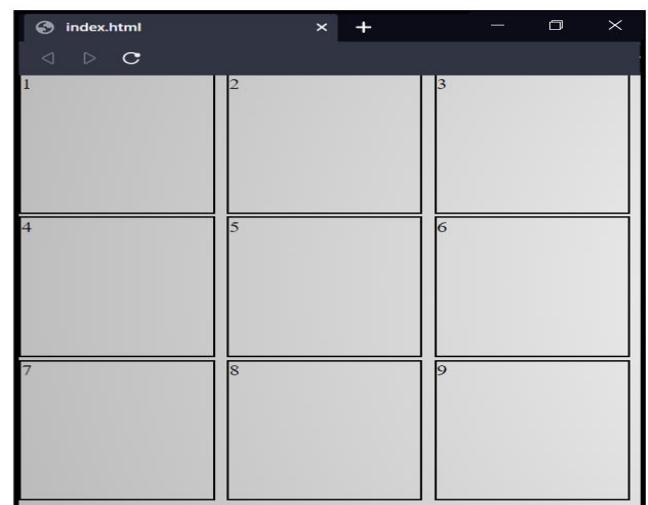
```
grid-template-rows: 150px 150px 150px; → grid-template-rows: repeat(3,150px);
```

- **grid-gap** Establece la distancia entre una celda y otra (no de los bordes).

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

```
.grid-container {
  display:grid;
  grid-template-rows: 1fr 1fr 1fr;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 3px 9px;
}
```

separación entre filas





Propiedades del grid-item

- **grid-row**
- **grid-column**

Modifica las *grid line* (líneas divisorias de columnas / filas) de una celda.

`grid-column: 1 / 3;`

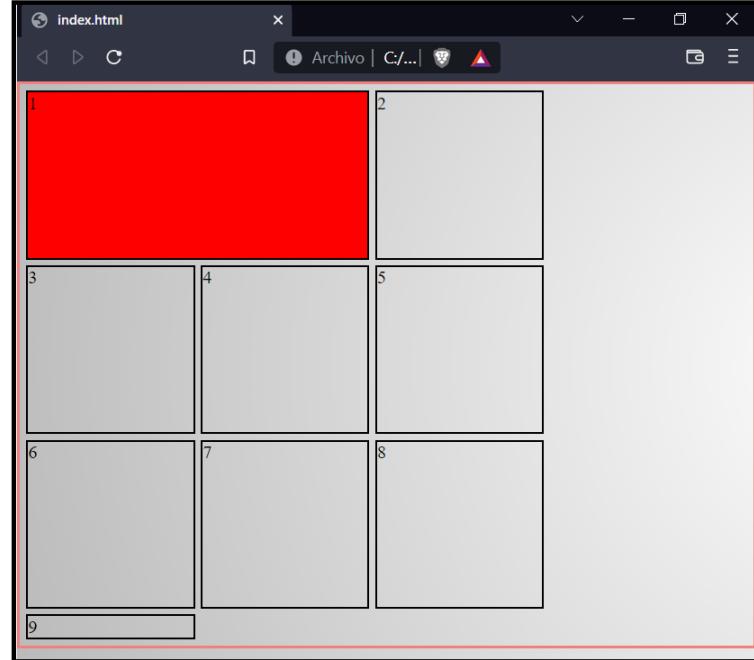
↳ grid line inicial / grid line final

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>

.grid-container {
  display:grid; /* bloque contenedor de la grilla */
  grid-template-rows: 150px 150px 150px;
  grid-template-columns: 150px 150px 150px;
  grid-gap: 5px; /* distancia entre celdas */
  padding: 5px; /* distancia de las celdas con el borde */
  border: 1px lightcoral solid 3px;
}

.grid-item {
  border: 2px solid #000;
}

.grid-item:first-child {
  background-color: red;
  grid-column: 1 / 3;
```



Al modificar las grid lines, se “empuja” todo el resto del contenido.

Si se desea seleccionar otro grid-item, es recomendable utilizar la pseudo-clase :nth-child(X)

Se puede utilizar la palabra reservada **span** para indicar un número específico de grid lines. `grid-column: 1 / span 3;`

↳ ocupará desde la columna 1 y 3 columnas a la derecha

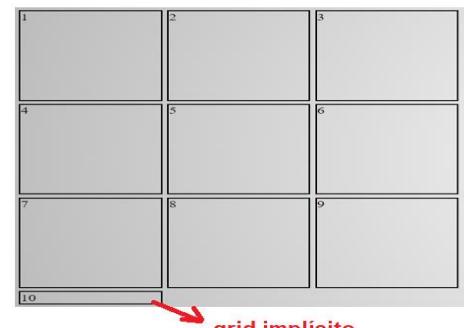


Grid implícito

Un grid implícito es un grid-item que no tiene espacio para ubicarse en la grilla.

Por ejemplo, si tenemos 10 grid items pero sólo 9 celdas (3 columnas y 3 filas), el décimo grid item será un grid implícito.

Los grid implícitos tienen sus propiedades específicas, las cuales se establecen en el grid-container.

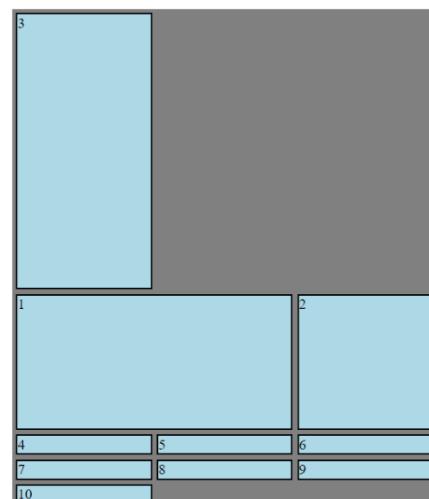


- **grid-auto-rows** Define la medida de las filas implícitas que se crearán.
- **grid-auto-columns** Define la medida de las columnas implícitas que se crearán.
- **grid-auto-flow** Define dónde se agregarán los grid implícitos (en nuevas filas o en nuevas columnas). **grid-auto-flow: column;**

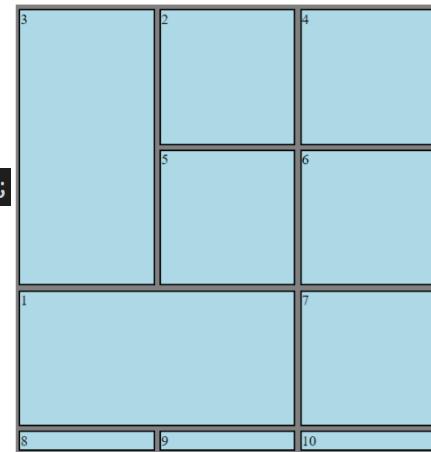
Es posible que, en la grilla, queden celdas vacías. Esto puede suceder cuando se modifican las grid-line, estableciendo un inicio en una grid-line posterior a la inicial.

Dichas celdas vacías se rellenan utilizando el valor **dense**. El algoritmo busca cuál es la celda que quepe más próxima y la coloca en las celdas vacías.

```
.grid-container {  
    display:grid;  
    grid-template-rows: repeat(3,150px);  
    grid-template-columns: repeat(3,150px);  
    grid-gap: 5px;  
    padding: 5px;  
    border: 2px solid #000;  
    background: gray;  
}  
  
.grid-item {  
    border: 2px solid #000;  
    background: lightblue;  
}  
  
.grid-item:first-child {  
    grid-column: 1 / span 2;  
}  
  
.grid-item:nth-child(3) {  
    grid-row: 1 / 3;  
}
```



grid-auto-flow: dense;





Grid dinámico

Para que la grilla se considere dinámica, debe ser capaz de adaptarse a distintas resoluciones.

Cuando se utiliza `grid-template` y se establece un tamaño en `fr`, dichas celdas serán encogibles sólo hasta donde su contenido lo permite. Por ejemplo: la palabra más larga.

```
grid-template-columns: repeat(3, minmax(min-content, 1fr) );
```

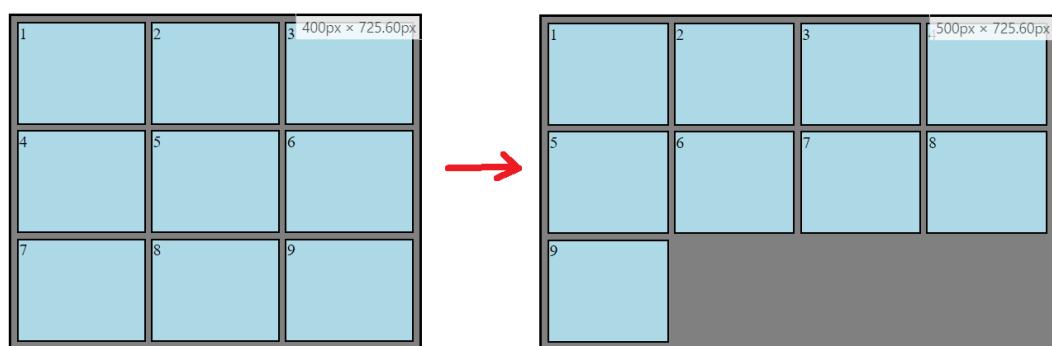
Sin embargo, esto sólo ajusta el tamaño de las columnas existentes.

Se pueden agregar columnas / filas dinámicamente con:

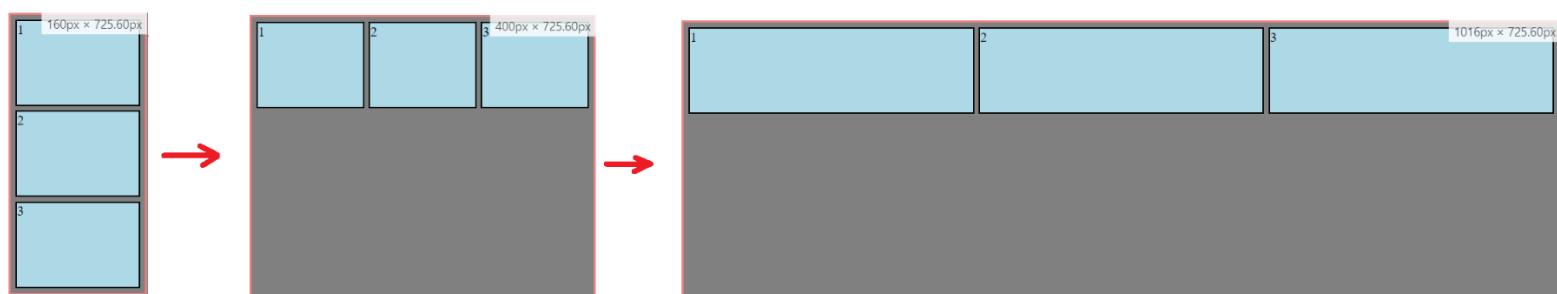
- **auto-fill** Permite añadir una nueva columna / fila cuando el contenido lo permite.

```
grid-template-columns: repeat(auto-fill, minmax(100px, 1fr) );
```

En este caso, cuando el ancho de página permita que se pueda añadir una nueva columna de 100px, la añadirá.



- **auto-fit** Permite que el contenido se adapte a las columnas / filas existentes.





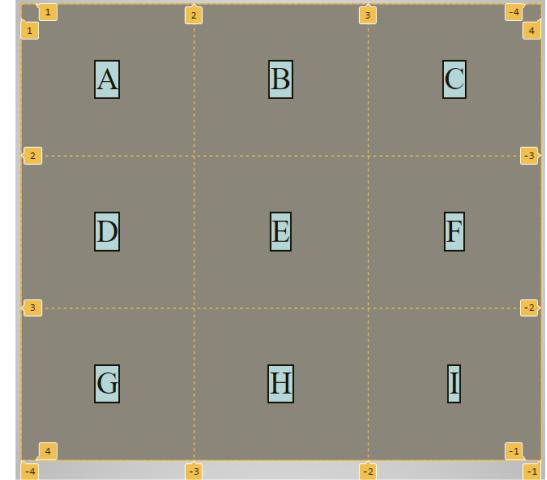
Alineación y control de flujo en bloque

Se aplica al grid-container para alinear todos los ítems o bien las grid-lines por separado.
Es importante saber que la propiedad “transition” no funciona en estas alineaciones.

- Alineación de ítems

- **justify-items** Alinea horizontalmente todos los grid-items.
- **align-items** Alinea verticalmente todos los grid-items.
 - stretch
 - center
 - left
 - right

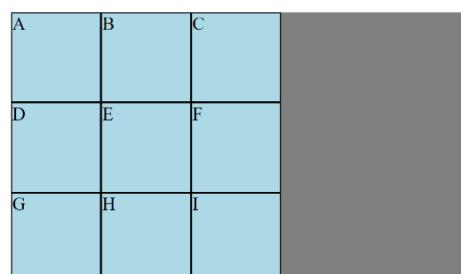
```
.grid-container {  
    display:grid;  
    margin: 5px;  
    background: #gray;  
    grid-template-rows: repeat(3, 150px);  
    grid-template-columns: repeat(3, minmax(100px, 1fr));  
    justify-items: center;  
    align-items: center;  
}  
  
.grid-item {  
    border: 2px solid #000;  
    background: #lightblue;  
    font-size: 30px;  
}
```



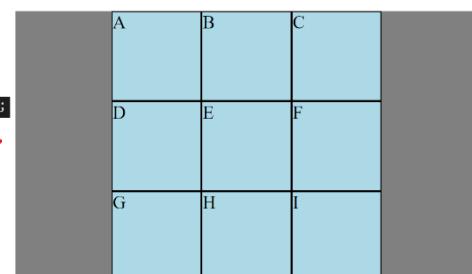
- Alineación de filas / columnas

- **justify-content** Alinea las columnas.
- **align-content** Alinea las filas.
 - space-around
 - space-between
 - space-evenly

igual que flex



justify-content: center;



Alineación individual

Se aplica sobre el grid-item.

- **justify-self** Alinea horizontalmente un grid-item específico.
- **align-self** Alinea verticalmente un grid-item específico.
 - stretch
 - start
 - center
 - end

```
.grid-item:nth-child(3) {  
    place-self: start end;  
} align justify
```

- **order** Altera la posición del grid-item respecto de otros.

Grid area

Forma de asignar un área específica a un conjunto de celdas (no necesariamente consecutivas).

Se utiliza la propiedad **grid-template-areas** sobre el grid-container de manera que queden “dibujadas” las áreas. Es necesario tener establecidas las filas y columnas que se van a utilizar.



Para que las celdas se acomoden correctamente al área asignada, se las debe colocar asignar manualmente al conjunto de grid-items.

```
.grid-header {
  background: #f96;
  grid-area: header;
}

.grid-main {
  background: #96f;
  grid-area: main;
}

.grid-aside {
  background: #888;
  grid-area: aside;
}

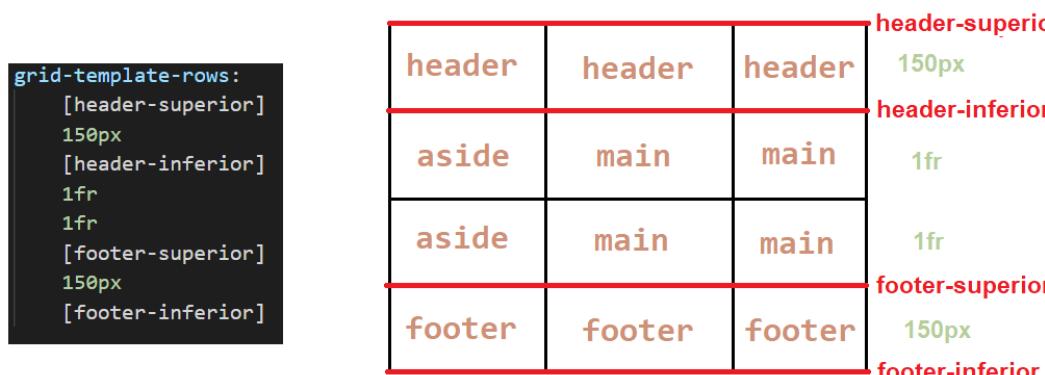
.grid-footer {
  background: #6f9;
  grid-area: footer;
}
```

```
<div class="grid-container">
  <div class="grid-item grid-header">Header</div>
  <div class="grid-item grid-main">Main</div>
  <div class="grid-item grid-aside">Aside</div>
  <div class="grid-item grid-footer">Footer</div>
</div>
```

2 clases

Nombramiento de grid-lines

Un grid-line puede ser nombrado en el momento en que es creado con grid-template.



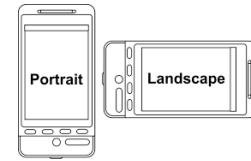


Media queries

Se utiliza en consultas de medios para aplicar diferentes estilos para diferentes tipos y dispositivos de medios. Son condicionales.

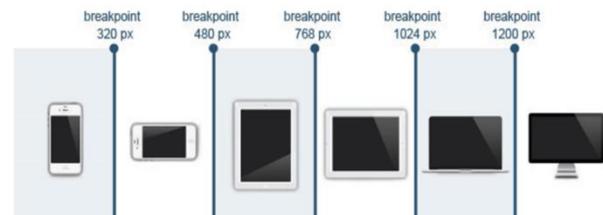
Permite consultar y validar:

- ✓ Ancho y alto de la ventana gráfica.
- ✓ Ancho y alto del dispositivo.
- ✓ Orientación del dispositivo (*landscape* o *portrait*).
- ✓ Resolución



Hay diferentes tipos de consultas:

- **all** Apto para todos los dispositivos.
- **print** Visualización de documentos en la vista previa de impresión.
- **screen** Destinado principalmente a pantallas.
- **speech** Destinado a sintetizadores de voz.



Existen diferentes modalidades de trabajo

- **Mobile first** Se desarrolla para resoluciones de mobile y que se pueda adaptar a otros dispositivos. Es la modalidad más recomendada.
- **Desktop first** Se desarrolla para resoluciones de computadora primero.
- **Content first** El contenido se adapta a las resoluciones. Pensado para marketing.

Formato de media query

```
@media screen and (min-width: 400px) {
```

tipos condicional condición

Puede haber más de una condición:

```
@media screen and (min-width: 400px) and (max-width: 650px) {
```



```
.content {
  display: flex;
  flex-direction: column; /* dirección del main-axis vertical */
  height: 100vh; /* ocupa toda la pantalla */
}

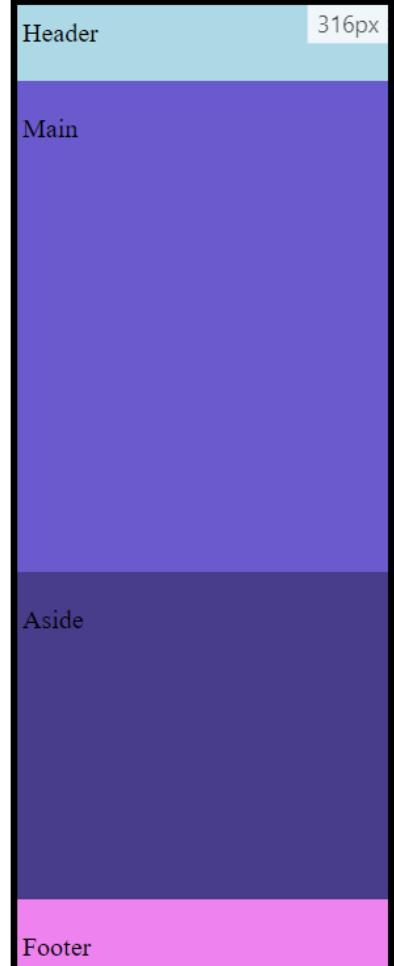
.header, .main, .aside, .footer {
  padding: 20px;
}

.header {
  background: #lightblue;
  flex-basis: 60px; /* width */
}

.main {
  background: #slateblue;
  flex-basis: 300px; /* width */
  flex-grow: 2; /* rellena cajas con espacio sobrante en la línea */
  flex-shrink: 0; /* establece el espacio que cederá al achicarse la pantalla */
}

.aside {
  background: #darkslateblue;
  flex-basis: 200px; /* width */
  flex-grow: 2; /* rellena cajas con espacio sobrante en la línea */
  flex-shrink: 0; /* establece el espacio que cederá al achicarse la pantalla */
}

.footer {
  background: #violet;
  flex-grow: 1; /* rellena cajas con espacio sobrante en la línea */
}
```



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

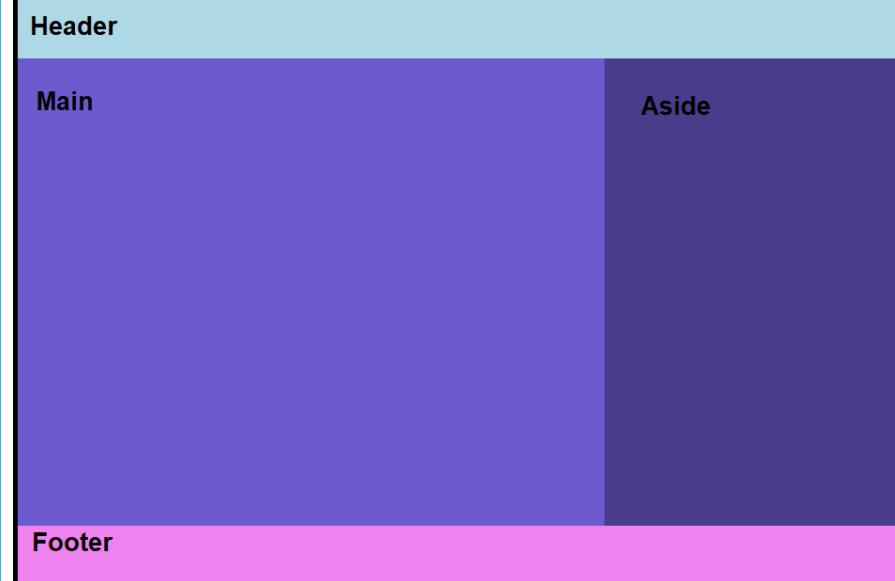
  .content {
    display: grid;
    grid-template-rows: 80px 1fr 1fr 1fr 1fr 80px; /* 6 filas */
    grid-template-columns: repeat(3, 1fr); /* 3 columnas */
  }

  .header {
    grid-column: 1 / span 3;
  }

  .main {
    grid-column: 1 / span 2;
    grid-row: 2 / 6;
  }

  .aside {
    grid-row: 2 / 6;
  }

  .footer {
    grid-column: 1 / span 3;
  }
}
```





Transition

La transición se aplica a cualquier cambio de cualquier elemento / propiedad. Para que dicha transición se ejecute, es necesario disparar un evento como [:hover](#), [onClick\(\)](#).

- **transition-property** Propiedad de un elemento que tendrá una transición.
 - inherit hereda el valor de propiedad de la caja que lo contiene.
- **transition-duration** Tiempo que dura la transición.
- **transition-delay** Tiempo que demora en iniciar la transición.
- **transition-timing-function** Curva de tiempo que va a tardar en realizarse la animación. Trabaja con curvas de bezier.
 - linear siempre a misma velocidad.
 - ease inicia rápido y enlentece luego bruscamente.
 - ease-in inicia lento y acelera luego.
 - ease-out inicia rápido y enlentece suave luego.
 - ease-in-out inicia lento, acelera y luego enlentece de nuevo.
 - step-start
 - step-end
 - steps (int, start|end)
 - initial
 - inherit



Keyframes (animaciones)

Las animaciones se definen mediante la regla @KeyFrames

```
@keyframes nombre-de-mi-animacion {  
    from {  
        /* propiedades del elemento al inicio */  
    }  
    to {  
        /* propiedades del elemento al final */  
    }  
}
```

```
@keyframes nombre-de-mi-animation {  
    0% {  
        /* propiedades del elemento al inicio */  
    }  
    50% {  
        /* propiedades del elemento a la mitad de la animación */  
    }  
    100% {  
        /* propiedades del elemento al final */  
    }  
}
```

- **animation-name** Nombre de la animación.
- **animation-duration** Duración de la animación.
- **animation-delay** Tiempo que tardará en iniciar la animación.
- **animation-timing-function** Idem transition. Trabaja con curvas de bezier.
 - **cubic-bezier()** Con esta función se establece la velocidad variable de la animación mediante una cuva de bezier. Existen generadores de curvas de bezier en internet que facilitan el trabajo.

cubic-bezier(0.7, 0.1, 0.1, 1.0)
- **animation-iteration-count** Cantidad de veces que se ejecuta la animación. Puede ser “infinite”.
`animation-iteration-count: infinite;`
- **animation-direction** Dirección en que se realiza la animación.
 - **normal** se ejecuta desde el *from* (por defecto).
 - **reverse** se ejecuta desde el *to*.
 - **alternate** se dirige de *from* a *to* y luego en reversa (sólo cuando iteration-count es mayor que 1).
 - **alternate-reverse**
- **animation-fill-mode** Determina la propiedad final.
 - **none** al finalizar la animación, vuelve a la normalidad.
 - **forwards** se queda con la propiedad del final de la animación.
 - **backwards** se queda con la propiedad del primer fotograma.
 - **both** inicia con el valor establecido en el *from*, incluso si hay establecido un delay, y termina como *forwards*.
 - **initial**
 - **inherit**

```
.container {  
    padding: 20px 5px;  
    background: #86f;  
}  
  
.caja {  
    background: #025;  
    height: 80px;  
    width: 80px;  
    margin: 20px;  
    animation-name: desplazarse;  
    animation-duration: 3s;  
    position: relative;  
}  
  
@keyframes desplazarse {  
    0% {  
        left: 0;  
        background: #025;  
    }  
    50% {  
        background: red;  
    }  
    100% {  
        left: 80%;  
        background: green;  
    }  
}
```



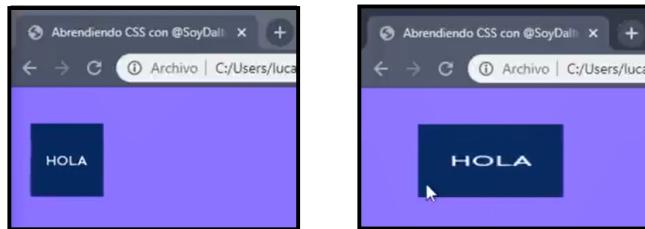
Transform

Permite transformar un objeto mediante funciones.

- **transform:** translateX() – translateY()
si se utilizan porcentajes, serán según el tamaño de la caja.
- **transform:** scale() – scaleX() – scaleY()
si se utilizan números sin unidad, serán cantidad de veces el tamaño de la caja.

```
.caja {  
    transition: transform 1s;  
}  
  
.caja:hover {  
    transform: scaleX(2) translateY(40px);  
}
```

si se escribieran en líneas diferentes,
la función transform sólo tomaría la
última en escribirse y no las dos.



- **transform:** rotate() – rotateX() – rotateZ() – rotate3D()
Se utilizan unidades de ángulo (deg, rad, etc.)



- **transform:** skew()

El texto que está dentro de la caja también se modifica por su característica de vector, es decir, al agrandar/alterar la caja, el texto adapta su resolución a su contenedor, haciendo que no pierda su calidad.

Para que el texto dentro de la caja no se altere al aplicar un *transform* a la caja, se puede aplicar el proceso inverso al elemento del texto.

```
<div class="container">  
    <div class="caja"><b>HOLA</b></div>  
</div>
```

Una forma recomendada de aplicar estas formas a cajas a cajas es utilizando **clip-path**, de los cuales existen generadores en internet.

```
.caja {  
    transform: skew(20deg);  
}  
  
b {  
    margin: auto;  
    transform: skew(-30deg);  
}
```

CSS clip-path maker

```
clip-path: polygon(50% 0%, 0% 100%, 100% 100%);
```



Variables

Una variable está formada por un **espacio en el sistema** de almacenaje (memoria principal de un ordenador) y un **nombre simbólico** (un identificador) que está asociado a dicho espacio.

Dicho espacio almacena un **valor**, es decir, contiene una cantidad de información conocida o desconocida. El valor de la variable puede cambiar durante el curso de la ejecución de un programa.

- **Variable global** Pueden utilizarse desde cualquier elemento, es decir, se puede usar con cualquier selector. Se asigna dentro del selector `:root`
- **Variable local** Puede utilizarse sólo dentro del selector en el cual fue definido.

```
:root { }
```

Para definir una variable, se utiliza el siguiente formato:

```
:root {  
    --color-rojo: #ff0102;  
}
```

The code is highlighted with a blue box. Two green arrows point from text labels to specific parts of the code:

- An arrow points to the color value `#ff0102` with the label **valor asignado a la variable**.
- An arrow points to the symbol `--color-rojo` with the label **nombre simbólico de la variable (iniciando con doble guión medio)**.

Para utilizar la variable en algún elemento, se utiliza el siguiente formato:

```
div {  
    background-color: var(--color-rojo);  
}
```



Filter

Es una forma de dar filtro a las imágenes.

Se deben aplicar los valores dentro de la misma propiedad.

- **none** *por defecto*
- **blur** *(px)*
- **brightness** *(0 - 1)*
- **contrast** *(number or %)*
- **saturate** *(%)*
- **opacity** *(%)*
- **drop-shadow** *(medidas)*

- **grayscale** *(%)*
- **hue-rotate** *(deg)*
- **invert** *(%)*
- **sepia** *(%)*
- **url**

Desenfoca

Brillo

Contraste

Saturación

Opacidad / Transparencia

Añade sombreado.

`filter: brightness(4) contrast(3);`

`filter: drop-shadow(0px 0px 10px #444);`

Nota: se puede aplicar varias veces
para un efecto más notable.

Rota la gamma de colores.

Invierte los colores. (50% da gris)





```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>index.html</title>
    <link rel="stylesheet" type="text/css" href="normalize.css">
    <link rel="stylesheet" type="text/css" href="estilo.css">
</head>

<body>
    <div class="container">
        <div class="caja"></div>
        <div class="caja"></div>
    </div>
</body>
</html>
```

```
* {
    box-sizing: border-box;
    padding: 0;
    margin: 0px;
    font-weight: 100;
}

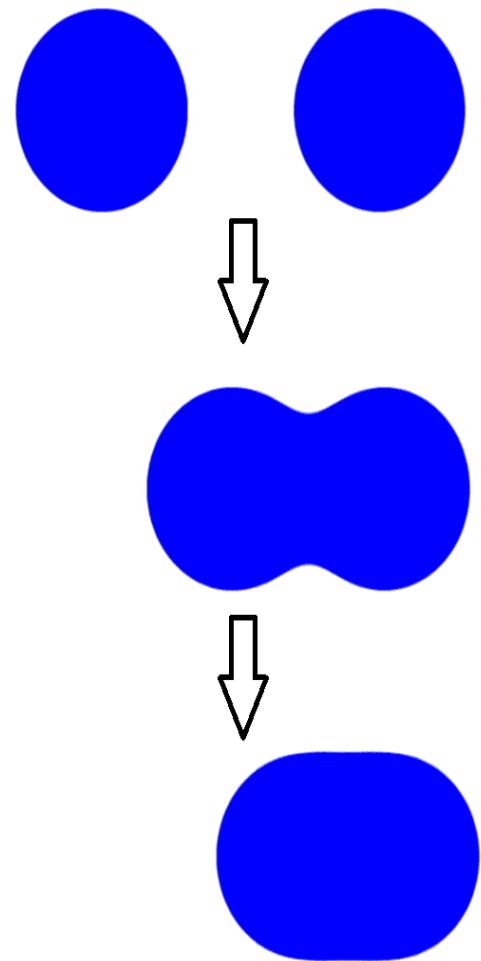
.container {
    filter: contrast(30);
    background-color: white;
}

.caja {
    display: inline-block;
    width: 150px;
    height: 150px;
    background: blue;
    margin: 40px;
    border-radius: 50%;
    filter: blur(20px);
}

.caja:first-child {
    animation: trasladar 3s alternate infinite;
}

@keyframes trasladar {
    0% {
        transform: translate(0);
    }

    100% {
        transform: translate(160px);
    }
}
```





Otras propiedades de CSS

- **direction** Cambia la dirección del texto
 - ltr Left to Right
 - rtl Right to Left
- **letterspacing** Separa las letras una cierta cantidad de px.

```
<a href="#content">Click aca</a>
<div>
    Lorem ipsum dolor sit amet, c
    sed do eiusmod
    tempor incididunt ut labore e
    enim ad minim veniam,
    quis nostrud exercitation ull
    ex ea commodo
</div>

<div id="content">
    Lorem ipsum dolor sit amet, c
    sed do eiusmod
    tempor incididunt ut labore e
</div>
```

Al hacer click en este link, la página se desplazará hasta el elemento con este ID

- **scroll-behavior** Alivianda el movimiento del scroll (por ejemplo, al hacer click en un link que hace desplazar la página). Se aplica al **body**.
 - smooth
- **user-select** Establece si un texto puede ser seleccionado o no por el usuario.
`user-select: none;`
- **text-shadow**



¿Cómo centrar un DIV?

```
<div class="container">
|   <p> Estoy Centrado :D </p>
</div>
```

```
.container {
  display: grid;
  place-items: center;
}
```

o bien

```
.container {
  display: flex;
  align-items: center;
  justify-content: center;
}
```