


CSS

CONCEPTOS

Sintaxis de etiquetas CSS

Todas las etiquetas CSS tendrán dos partes:

- Un ***selector***: Determina el elemento o elementos de la web a los que se les aplican los estilos.
- Las ***propiedades***: Indica una característica de estilo y su valor asociado (medida, color., etc).

```
body {  
  background-color:  burlywood;  
}
```



Aplicación de las CSS

Para aplicar reglas CSS a nuestra página HTML tenemos tres opciones:

- **Estilo in-line** → Dentro de una etiqueta con el atributo ***style*** modificando la apariencia del contenido de esa etiqueta.

```
<p style="color:green;">Párrafo de color verde.</p>
```

- **Opción desaconsejada para SEO.**
- Mezcla el contenido con el diseño.

Aplicación de las CSS

- **Incrustado en la cabecera** → Los estilos se declaran entre etiquetas `<style>` en la cabecera de la propia página `<head>`:

```
<html>
<head>
  <title>CSS incrustado en la cabecera</title>
  <style>p{color:green;}</style>
</head>
<body>
  <p>Párrafo de color verde.</p>
</body>
</html>
```

Aplicación de las CSS

- **Hojas de estilo externas** → Archivo externo con extensión .CSS referenciado desde la página mediante la etiqueta **<link>** con atributos **rel** y **href**:

index.html

```
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <p>Párrafo de color verde.</p>
</body>
</html>
```

styles.css

```
p {color:green;}
```

Atributos de **<link>**

- **rel** → debe tener el valor “stylesheet”
- **href** → debe indicar la ruta al fichero CSS.

Prioridad de las CSS



Selectores Básicos

En CSS, los selectores son patrones que se utilizan para seleccionar los elementos que desea modificar. En CSS disponemos de los siguientes selectores.

- Selectores de etiqueta.
- Selectores de clase.
- Selectores ID.
- Selectores de grupo.
- Selector universal.

Selectores básicos

➤ **Selector de etiqueta** → Se aplica a todos los elementos con la etiqueta indicada.

Ej: Se aplica a todos los elementos <p> de la página:

```
p {color: green;}
```

➤ **Selector de clase** → Se aplica a todos los elementos con el valor indicado en el atributo **class**.

Ej: Se aplica a todos los elementos con **class='blend'** (sin importar su etiqueta):

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

Se aplicará en:

```
<div class="blend otro-estilo"></div>
```



Un elemento puede tomar estilos de varias clases.
Si hay coincidencia prevalece la última indicada.

Identificadores de las clases.

➤ **Nombres descriptivos** → El nombre debe identificar los elemento/s al que se refiere la clase.

➤ **Evitar características visuales** → No incluir colores, tamaño, posición..., etc.

```
/* Selector con nombre que define la característica visual del color */  
.menu-red { background: red; }  
/* Utilizar mejor: */  
.nav-menu { background: red; }
```

➤ **Notación BEM** → La convención BEM divide los nombres de clase en: Bloque, Elemento y Modificador:

```
/* Bloque */  
.header {}  
  
/* Elemento */  
.header_logo {}  
  
/* Modificador */  
.header_logo-large {}
```

➤ **Separación entre palabras** → Pueden emplearse guiones bajos/medios, sin espacios, o combinando mayúsculas/minúsculas.

Selectores básicos

➤ **Selector identificador** → Se aplica al elemento con el valor del atributo *id*.

Ej: Se aplica a cualquier elemento con el atributo *id='cent'*:

```
#cent {color: blue;}
```

➤ **Selector de atributos** → Se aplica a todos los elementos con el atributo y valor indicados.

Ej: Se aplica a todos los elementos <input> con atributo *type='text'*:

```
input[type="text"] { border: 1px solid ■ black; }
```

Ej: Se aplica a todos los elementos con atributo *'disabled'*:

```
[disabled] { opacity: 0.5; }
```

Selectores básicos

➤ **Selector descendiente** → Se aplica sobre los elementos contenidos en otros elementos.

- `selector1 selector2 { }`

Ej: Se aplica sobre todos los elementos **<p>** contenidos dentro de un elemento **<div>**.

```
div p { color: black; }
```



El uso de selectores descendentes es recomendable siempre que sea posible antes de crear un selector clase o un selector identificador.

➤ **Combinación de selectores** → Se aplica sobre los elementos contenidos en otros elementos.

- `selector1, selector2 { }`

Ej: Se aplica sobre todos los elementos **<p>** y elementos **<div>**.

```
div, p { color: orange; }
```

Selectores básicos

➤ **Selector hijos** → Se aplica sobre los elementos hijos directos de otros.

- `selector1 > selector2 { }`

Ej: Se aplica sobre todos los elementos **<p>** contenidos directamente en un elemento **<div>**

```
div > p { color: white;}
```

➤ **Selector adyacente** → Se aplica sobre los elementos hermanos que van seguidos en el código HTML y contenidos en el mismo elemento padre.

- `selector1 + selector2 { }`

Ej: Se aplica sobre todos los elementos **<p>** que siguen a un **<div>** en el mismo contenedor

```
div + p { color: black;}
```

Pseudo-clases para selección.

Permiten la selección de elementos hermanos / hijos por su posición:

Pseudo-clase	Descripción
:first-child	Primer hijo
:last-child	Último hijo
:first-of-type	Primer hermano de su tipo
:last-of-type	Último hermano de su tipo
:only-child	Hijos únicos
:only-of-type	Únicos hermanos de su tipo
:empty	Elementos que no tienen hijos
:nth-child(n)	Enésimo elemento hijo (odd / even → impares / pares)
:nth-last-child(n)	Enésimo elemento hijo contando desde el último
:nth-of-type(n)	Enésimo hermano de su tipo
:nth-last-of-type(n)	Enésimo hermano de su tipo comenzando desde el último

Pseudo-clases de estados.

Permiten aplicar estilos en función del estado de un elemento:

Pseudo-clase	Descripción
:link	No visitado por el usuario
:visited	Visitado por el usuario
:hover	Modifica el estilo cuando un elemento apuntador pasa por encima
:active	Se activa cuando el usuario pulsa el elemento
:focus	Se activa cuando tiene el foco sobre el elemento

Ejemplos pseudo-clases CSS

Selectores Básicos :

```
p { color: blue; } /* Selecciona todos los elementos <p> */
.miClase { color: red; } /* Selecciona todos los elementos con clase "miClase" */
#miID { color: green; } /* Selecciona el elemento con ID "miID" */
* { margin: 0; padding: 0; } /* Selecciona todos los elementos */
a[href] { color: orange; } /* Selecciona todos los enlaces con el atributo href */
```

Selectores de atributos:

```
[disabled] { opacity: 0.5; } /* Selecciona elementos con el atributo "disabled" */
input[type="text"] { border: 1px solid black; } /* Selecciona inputs de tipo "text" */
a[href^="https"] { color: green; } /* Selecciona enlaces que comienzan con "https" */
a[href$=".pdf"] { color: red; } /* Selecciona enlaces que terminan en ".pdf" */
a[href*="example"] { color: blue; } /* Selecciona enlaces que contienen "example" */
```

Selectores combinados:

```
div p { color: purple; } /* Selecciona <p> dentro de cualquier <div> */
div > p { color: blue; } /* Selecciona <p> que son hijos directos de <div> */
h1 + p { color: grey; } /* Selecciona el <p> inmediatamente después de <h1> */
h1 ~ p { color: grey; } /* Selecciona todos los <p> que siguen a <h1> */
```

Ejemplos pseudo-clases CSS

Selectores de pseudo-clases:

```
a:hover { color: orange; } /* Cambia el color de un enlace al pasar el cursor */
input:focus { border-color: blue; } /* Cambia el borde del input cuando está enfocado */
p:first-child { font-weight: bold; } /* Selecciona el primer <p> hijo */
p:last-child { font-weight: bold; } /* Selecciona el último <p> hijo */
li:nth-child(2) { color: red; } /* Selecciona el segundo <li> */
li:nth-last-child(2) { color: green; } /* Selecciona el penúltimo <li> */
p:nth-of-type(3) { color: purple; } /* Selecciona el tercer <p> */
p:not(.miClase) { color: grey; } /* Selecciona todos los <p> que no tienen la clase "miClase" */
```

Selectores de pseudo-elementos

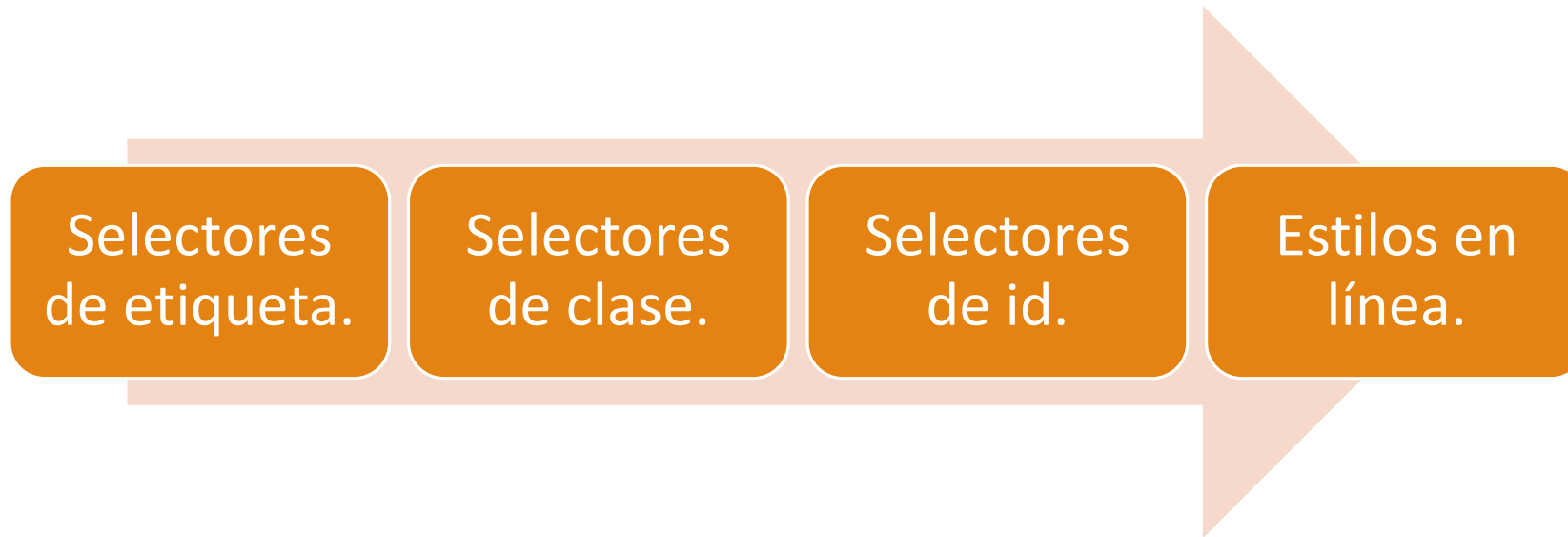
```
h2::before { content: "→ "; } /* Inserta una flecha antes de cada <h2> */
h2::after { content: "← "; } /* Inserta una flecha después de cada <h2> */
p::first-line { font-weight: bold; } /* Aplica negrita a la primera línea del párrafo */
p::first-letter { font-size: 2em; color: red; } /* Agrandar y cambiar de color la primera letra */
```

Selectores de estado para elementos de formularios

```
input[type="checkbox"]:checked { background-color: green; } /* Aplica color a checkboxes seleccionados */
input:disabled { opacity: 0.5; } /* Hace más claro un input deshabilitado */
input:enabled { border-color: blue; } /* Cambia el borde de inputs habilitados */
input:required { border-color: red; } /* Bordes rojos para inputs requeridos */
input:optional { border-color: green; } /* Bordes verdes para inputs opcionales */
```


Concepto de especificidad

Cuando se aplican múltiples estilos CSS a un elemento a través de diferentes selectores, prevalecen los estilos aplicados con mayor grado de especificidad:



Medidas absolutas / relativas

Las **unidades de longitud absoluta** son fijas y se muestran igual independientemente de las características del dispositivo.

px	Píxeles
in	Pulgadas (1 pulgada = 2.54 cm)
cm	Centímetros
mm	Milímetros
pt	Puntos (1 pt = 1/72 pulgadas)
pc	Picas (1 pica = 12 puntos)

Las **unidades de longitud relativa** se ajustan a cada dispositivo dependiendo de la resolución de la pantalla:

em	Relativo al tamaño de la fuente del elemento (2 em significa 2 veces el tamaño de la fuente actual)
%	Porcentaje (relativo al elemento padre)
vh y vw	Medidas relativas de acuerdo al viewport 1vh = 1% de la altura del viewport 100vh = altura del viewport
fr	Flexible Grid Units (fr) Se utiliza en Grid Layout y representa una fracción del espacio disponible en un contenedor

Se **recomienda usar unidades relativas** para mejorar el carácter responsivo de la página web y que se adapte a cualquier medio.

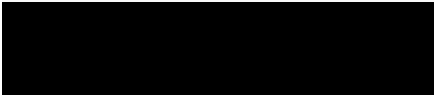


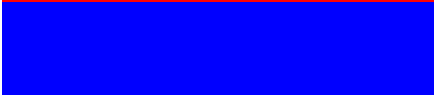


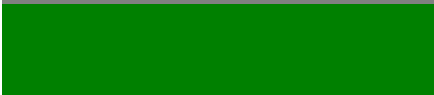

Medidas comunes absolutas / relativas

- **Unidad (em)** → Es especialmente útil para establecer tamaños proporcionales al tamaño de fuente. Aunque no hay un criterio definido, el organismo W3C, recomienda el uso de la unidad em para indicar el tamaño del texto. El tamaño de los ems se establece en base al tamaño que tenga definido el navegador.
- **Porcentaje (%)** → Representa una proporción del tamaño del elemento padre. Es útil para hacer diseños fluidos y responsivos teniendo en cuenta la relación de los elementos con su contenedor padre.
- **Viewport Width (vw) y Viewport Height (vh)** → Representan un porcentaje del ancho y alto de la ventana del navegador, respectivamente. Son útiles para crear diseños responsivos basados en el tamaño de la pantalla.
- **Flexible Grid Units (fr)** → Se utiliza en Grid Layout y representa una fracción del espacio disponible en un contenedor. Es útil para distribuir el espacio disponible entre elementos flexibles.

Definición de colores

- **Colores preestablecidos:** Accederemos directamente poniendo su nombre en inglés:
`p {color:blue}`
- **Rgb** (rojo, verde, azul): Donde rojo, verde y azul son números enteros desde 0 a 255 o porcentajes de 0% a 100%
`p {color:rgb(42,44,156)}`
- **Rgba**(rojo, verde, azul, opacidad): Idéntico al anterior pero con un valor decimal de transparencia comprendido entre 0 (transparente) y 1 (opaco).
`p {color:rgba(255,0,0,.5)}`
- **#RGB:** Donde R, G y B son números hexadecimales desde 0 hasta F:
`p {color: #353E9A}`

Colores básicos

Color	Nombre	HEX	RGB
	black	#000000	0,0,0
	white	#ffffff	255,255,255
	red	#ff0000	255,0,0
	blue	#0000ff	0,0,255
	yellow	#ffff00	255,255,0
	gray	#808080	128,128,128
	green	#008000	0,128,0
	lime	#00ff00	0,255,0

Propiedades *background / color*

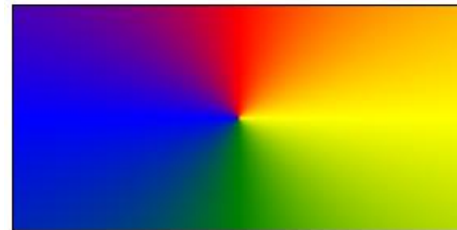
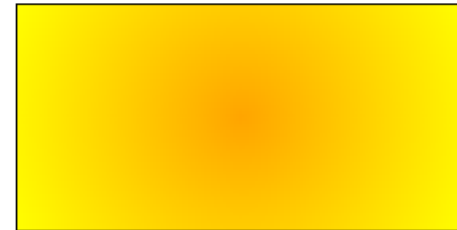
Propiedad	Descripción	Valores
color	Color del texto	RGB HSL HEX nombre del color RGBA HSLA
background-color	Color de fondo	RGB HSL HEX nombre del color RGBA HSLA transparent
background-image	Imagen de fondo	url(«...») none
background-repeat	Repetición de la imagen de fondo	repeat repeat-x repeat-y no-repeat
background-attachment	Desplazamiento de la imagen de fondo	scroll fixed
background-position	Posición de la imagen de fondo	percentage length left center right
background-size	Tamaño de la imagen de fondo	auto cover contain valor
opacity	Transparencia de un elemento	[0 – 1] (0 → totalmente transparente)

```
background: #f0f0f0 url('imagen.jpg') no-repeat center/cover;
```

Gradientes

Hay que especificar por orden, tipo de degradado, dirección del degradado, color inicial y color final:

```
/* Degradado lineal de arriba hacia abajo */  
.gradient1 {  
| background: linear-gradient(to bottom, red, yellow);  
}  
  
/* Degradado radial circular */  
.gradient4 {  
| background: radial-gradient(circle, orange, yellow);  
}  
  
/* Degradado cónico */  
.gradient6 {  
| background: conic-gradient(red, yellow, green, blue, red);  
}
```



Propiedades de textos

Estos estilos pueden aplicarse a todos los textos de un capa (herencia), o a una parte del texto contenida entre etiquetas **<spam>**:

Propiedad	Descripción	Valores
text-indent	Desplazamiento a la derecha de la primera línea del párrafo	longitud porcentaje
text-align	Alineamiento del texto	left right center justify
text-decoration	Efectos de subrayado y tachado	none underline overline line-through
letter-spacing	Espacio entre caracteres	normal longitud
word-spacing	Espacio entre palabras	normal longitud
text-transform	Transformación a mayúsculas / minúsculas	capitalize uppercase lowercase none
line-height	Tamaño del espacio entre líneas (interlineado)	longitud porcentaje
vertical-align	Alineación vertical del texto	top middle bottom baseline sub super valor

Propiedades de la fuente

Estos estilos pueden aplicarse a todos los textos de un capa (herencia), o a una parte del texto contenida entre etiquetas **<spam>**:

Propiedad	Descripción	Valores
font-family	Familias de fuentes	nombre-familia *
font-style	Estilo de la fuente	normal italic oblique
font-variant	Convierte a mayúsculas manteniendo todas las letras en un tamaño inferior a la primera	normal small-caps
font-weight	Anchura de los caracteres. Normal = 400, Negrita = 700	normal bold bolder lighter 100 200 300 400 500 600 700 800 900
font-size	Tamaño de la fuente	xx-small x-small small medium large x-large xx-large larger smaller longitud porcentaje

Fuentes personalizadas

Los ficheros de fuentes de texto son ficheros que contienen información para mostrar diferentes tipografías en dispositivos digitales.

Formatos más comunes:

- **TrueType Font** (*.ttf*)
- **OpenType Font** (*.otf*)



Deben observarse las condiciones de los derechos de autor de las fuentes obtenidas:

Summer Season - scratchones

in : Calligraphy, Script, Handwriting

Summer Season

Free for Personal Use

DOWNLOAD

Buy Commercial License



Fuentes personalizadas

Para emplear una fuente personalizada debemos descargar el fichero de fuente y referenciarla con la directiva **@font-face**:

```
@font-face {  
    font-family: 'miFuente';  
    src: url('Bauhaus.ttf');  
}
```

- **font-family**: Indica el identificador de la fuente.
- **src**: Indica la ruta al fichero de la fuente (.ttf / .otf).

Luego usamos la fuente:

```
body {  
    font-family: 'miFuente';  
}
```

LOREM IPSUM

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Suscipit tenetur perspiciatis vero asperiores libero enim facere minus doloremque molestias sapiente, quod nesciunt, saepe nam, ad ea eveniet rem ipsum officiis!

Propiedades de listas.

Propiedad	Descripción	Valores
list-style-type	Estilo aplicable a los marcadores visuales o viñetas de las listas	disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-latin upper-latin armenian georgian lower-alpha upper-alpha none
list-style-image	Imagen aplicable a las viñetas de las listas	url() none
list-style-position	Posición de las viñetas dentro de la lista	inside outside
list-style	Permite establecer varios estilos de la lista en una sola propiedad	list-style-type list-style-position list-style-image

Propiedad **list-style-type**

- Tipo círculo
- Tipo círculo
- Tipo círculo
- Tipo cuadrado
- Tipo cuadrado
- Tipo cuadrado

Listas ordenadas:

- I. Uno
- II. Dos
- III. Tres
- a. Uno
- b. Dos
- c. Tres

Propiedad **list-style-image**

- Lista con imagen externa
- Lista con imagen externa
- Lista con imagen externa

Propiedades de tablas.

Propiedades que afectan al aspecto de las tablas `<table>`.

Propiedad	Descripción	Valores
caption-side	Posición del título respecto la tabla	top bottom
table-layout	Establece el ancho de las columnas de la tabla	auto fixed
border-collapse	Selección del modelo de los bordes	collapse separate
border-spacing	Espaciado entre los bordes de celdas adyacentes	longitud
empty-cells	Visibilidad de los bordes de celdas sin contenido	show hide

Table-layout: auto, border-collapse: collapse

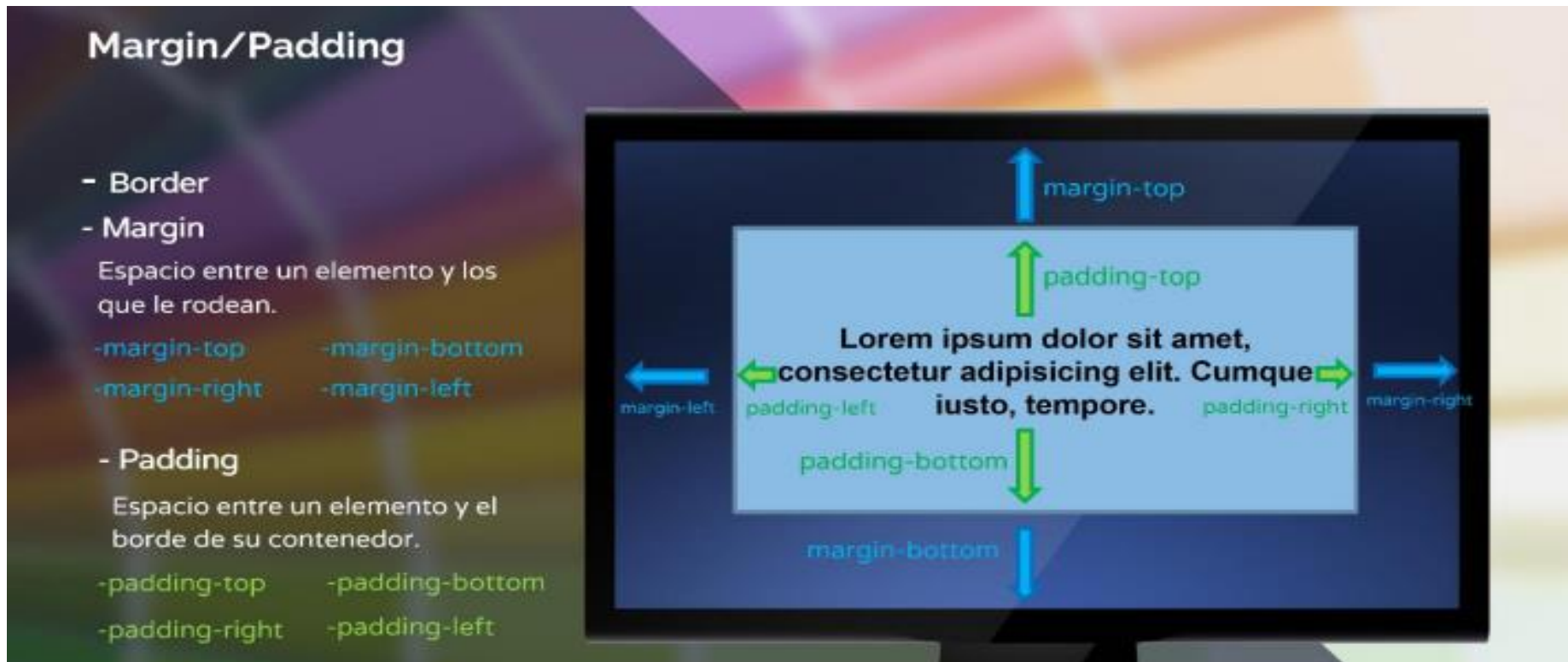
Columna 1	Columna 2	Columna 3
Fila 1.Lorem ipsum dolor atmet.	Fila 1	Fila 1
Fila 2	Fila 2	Fila 2
Fila 3	Fila 3	Fila 3

Table-layout: fixed, border-collapse: separata, borde-spacing: 5px

Columna 1	Columna 2	Columna 3
Fila 1.Lorem ipsum dolor atmet.	Fila 1	Fila 1
Fila 2	Fila 2	Fila 2
Fila 3	Fila 3	Fila 3

Modelo caja-contenedor

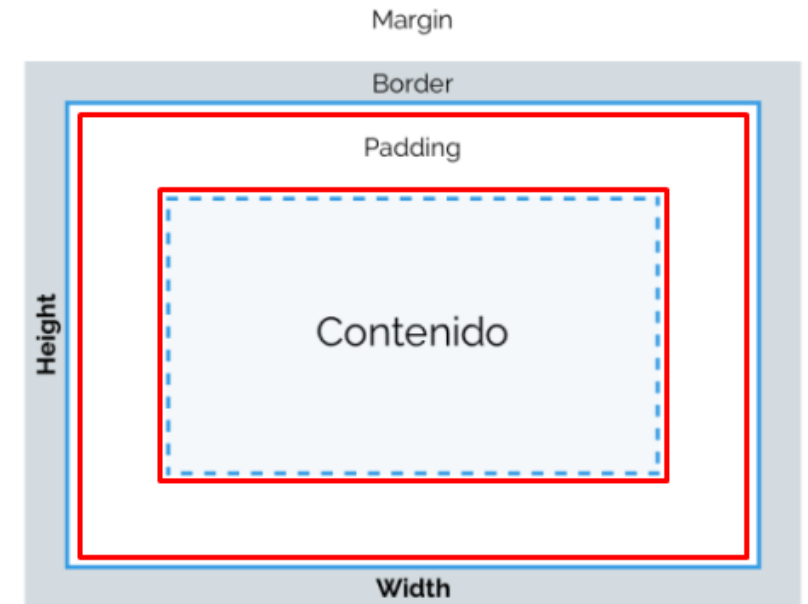
Todo elemento HTML dispone de una estructura tipo caja que se puede modificar usando las propiedades CSS:



Propiedad *padding*

El ***padding*** es el **margen interno** (relleno) y es la cantidad de espacio dejado entre el borde de un contenedor y su contenido.

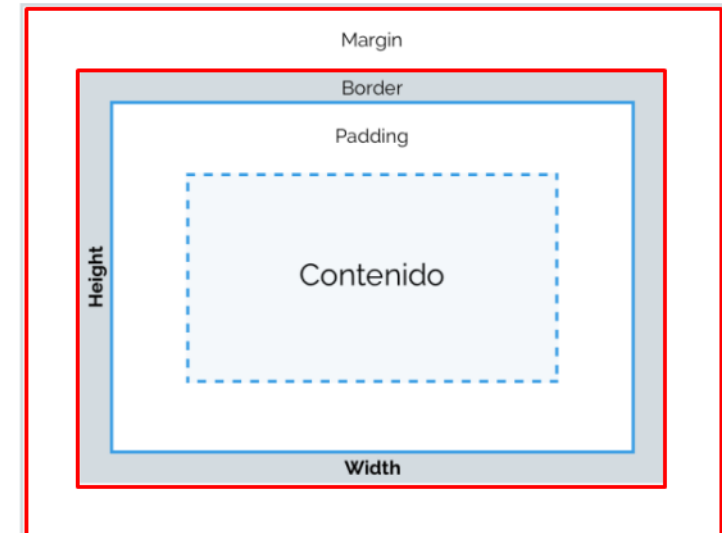
Propiedad	Descripción	Valores
padding-top padding-right padding-bottom padding-left	Tamaño del relleno superior, derecho, inferior e izquierdo	longitud porcentaje
padding	Tamaño del relleno	longitud porcentaje {1,4}



Propiedad *margin*

El **margin** es el **margen externo** de un elemento y es el espacio dejado entre un elemento y los dispuestos a su alrededor.

Propiedad	Descripción	Valores
margin-top margin-right margin-bottom margin-left	Tamaño del margen superior, derecho, inferior e izquierdo	longitud porcentaje auto
margin	Ancho de los márgenes	longitud porcentaje auto {1,4}



Propiedades abreviadas margen y relleno

La propiedad ***margin*** (*margen*) de CSS se utiliza para crear espacio alrededor de un elemento, separándolo de otros elementos.

```
/* Aplica a todos los cuatro lados */
```

```
margin: 1em;
```

```
/* Vertical | Horizontal */
```

```
margin: 5% auto;
```

```
/* Arriba | Horizontal | Abajo */
```

```
margin: 1em auto 2em;
```

```
/* Arriba | Derecha | Abajo | Izquierda */
```

```
margin: 2px 1em 0 auto;
```

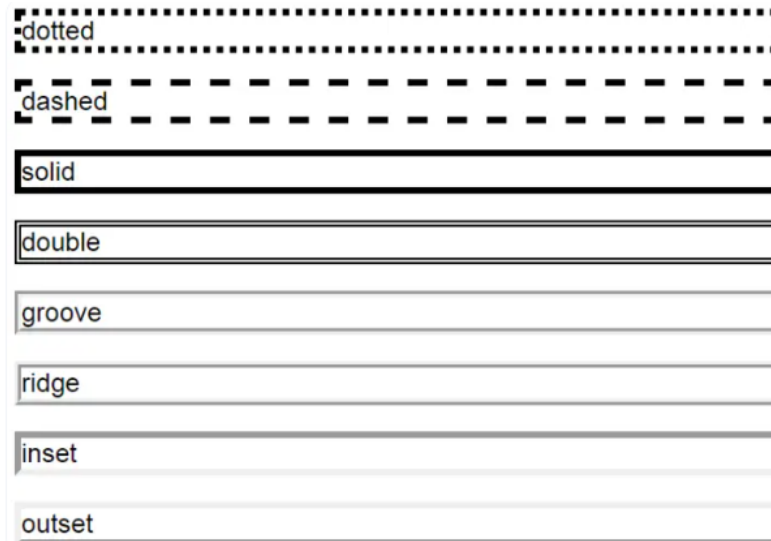
La propiedad ***padding*** (*relleno*) de CSS se utiliza para crear espacio dentro de un elemento, entre el contenido y el borde del mismo.

Propiedad *border*

La propiedad *border* define un borde alrededor de un elemento indicando el grosor, estilo y color del borde:

```
border: <ancho> <estilo> <color>;
```

Estilos:



Vista general de los tipos de borde en CSS

Existen propiedades para bordes solo en lados específicos:

```
border-top: 4px solid red;  
border-right: 2px dashed green;  
border-bottom: 4px solid blue;  
border-left: 2px dashed yellow;
```

Cada una sigue la misma sintaxis que *border* y permite personalizar el borde de cada lado de forma independiente.

Desbordamiento de contenidos

La propiedad **overflow** permite controlar el comportamiento del contenido que se encuentra en una caja o contenedor. Podemos especificar si queremos recortar un contenido, mostrar barras de desplazamiento o mostrar el contenido que excede de un elemento a nivel de bloque.

- **overflow: visible** (default). Hace que los contenidos se salgan del contenedor resultando visibles.
- **overflow: hidden**. Los contenidos que se salen del contenedor se ocultan y no se muestran barras de desplazamiento. De esta forma se puede controlar el tamaño del elemento y su contenido.
- **overflow: scroll**. Se muestra una barra de desplazamiento (horizontal y vertical) cuando los contenidos no entran en el contenedor o caja.
- **overflow: auto**. El navegador es el que decide si se muestran las barras de scroll o si se extiende el contenedor. En cualquier caso, gracias a este valor nunca se permite que el contenido desborde al contenedor. Este valor es muy interesante ya que si el contenido se sale por un lado (horizontal o vertical), sólo se muestra la barra de scroll de ese lado.

Propiedad *box-sizing*

Por defecto en el modelo de cajas de CSS, el ancho y alto asignado a un elemento es aplicado solo al contenido de la caja del elemento. Si el elemento tiene algún borde (border) o relleno (padding), este es entonces añadido al ancho y alto del tamaño de la caja o contenedor.

La propiedad ***box-sizing*** permite resolver esto mediante los valores *content-box* y *border-box*:

- ***content-box*** → es el comportamiento CSS por defecto para el tamaño de la caja (box-sizing). Si se define el ancho de un elemento en 100 píxeles, la caja del contenido del elemento tendrá 100 píxeles de ancho, y el ancho de cualquier borde o relleno será añadido al ancho final desplegado.
- ***border-box*** → toma en cuenta cualquier valor que se especifique de borde o de relleno para el ancho o alto de un elemento. Es decir, si se define un elemento con un ancho de 100 píxeles. Esos 100 píxeles incluirán cualquier borde o relleno que se añada, y la caja de contenido se encogerá para absorber ese ancho extra. Esta propiedad es especialmente útil para redimensionar cualquier elemento.

Elementos block / in-line

- Etiquetas BLOCK

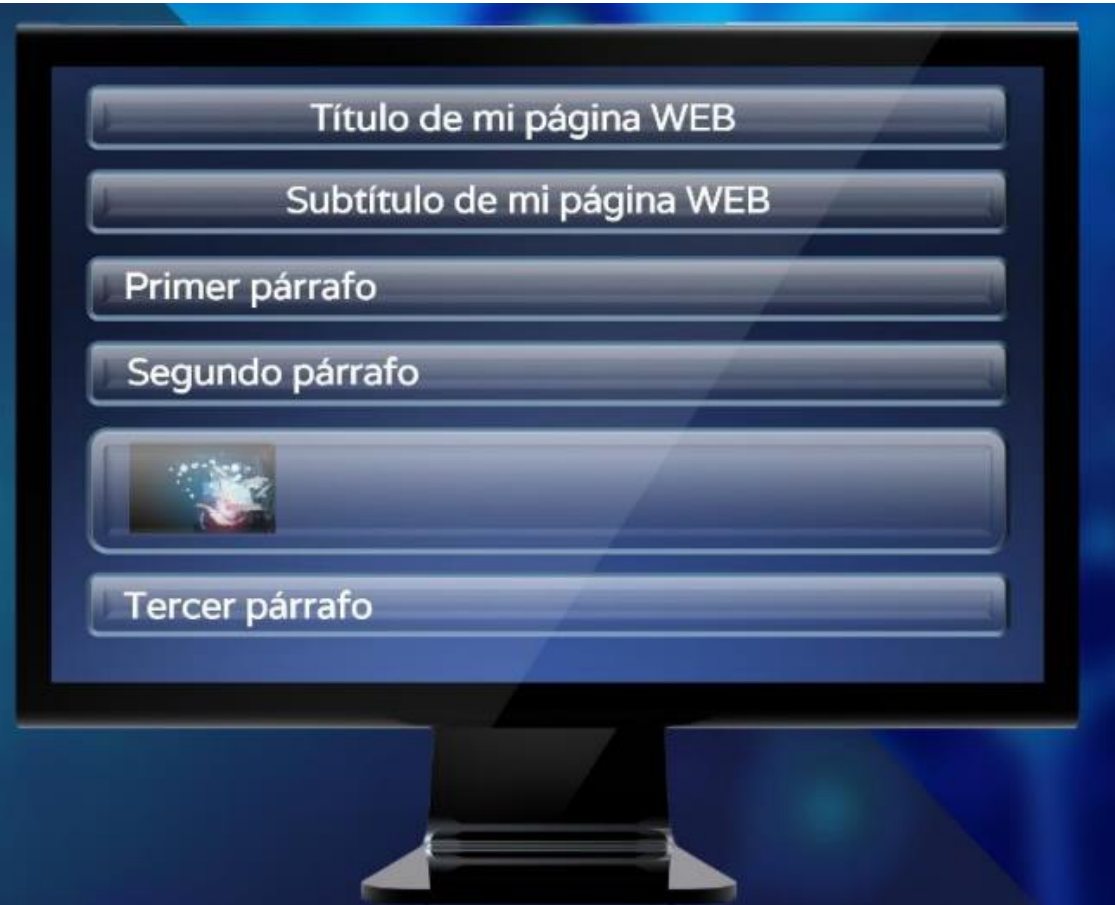
Provocan un salto de página por arriba y por debajo de la etiqueta y ocupan el 100% del ancho de la página. La mayoría de etiquetas de estructura son de este tipo.

```
<h1>Título de mi página WEB</h1>
<h2>Subtítulo de mi página WEB</h2>
<p>Primer párrafo</p>
<br>
<p>Segundo párrafo</p>
<div></div>
<br>
<p>Tercer párrafo</p>
```

- Etiquetas IN-LINE

No generan salto de línea. No se encuadran en un contenedor.

`` `<a>` `<image>`



Elementos block / in-line

Elemento en bloque

- Capas
- Párrafos
- Encabezados
- Listas
- Tablas
- etiquetas semánticas
- formularios...,

Elementos en línea

- Enlaces
- Imágenes
- modificadores de texto
- elementos de formulario.

Propiedad *display*

Para que un elemento en línea pase a ser de bloque utilizamos '*block*' en la propiedad ***display***:

```
img {  
  display: block;  
}  
  
<body>  
    
    
    
</body>
```



Propiedad *display*

Para que un elemento de bloque pase a ser en línea utilizamos: 'inline'

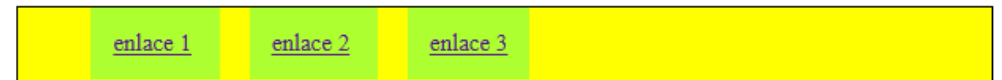
```
<div class="capa2">
  <ul>
    <li>
      <a href="index.html">enlace 1</a>
    </li>
    <li>
      <a href="index.html">enlace 2</a>
    </li>
    <li>
      <a href="index.html">enlace 3</a>
    </li>
  </ul>
</div>
```



```
.capa2 {
  background: yellow;
  border: 1px solid black;
}

.capa2 li {
  display: inline;
  background-color: greenyellow;
  margin: 10px;
  padding: 15px;
}
```

- [enlace 1](#)
- [enlace 2](#)
- [enlace 3](#)



Propiedad *display*

El modo 'inline-block'

- No comienza en una nueva línea
- Permite establecer *width* y *height* y respeta el *margin* y el *padding* superiores e inferiores.

```
.capa3 {  
  background: yellow;  
  border: 1px solid black;  
}  
.capa3 li {  
  display: inline-block;  
  background-color: greenyellow;  
  margin: 10px;  
  padding: 15px;  
}
```

```
<div class="capa2">  
  <ul>  
    <li>  
      <a href="index.html">enlace 1</a>  
    </li>  
    <li>  
      <a href="index.html">enlace 2</a>  
    </li>  
    <li>  
      <a href="index.html">enlace 3</a>  
    </li>  
  </ul>  
</div>
```

[enlace 1](#)

[enlace 2](#)

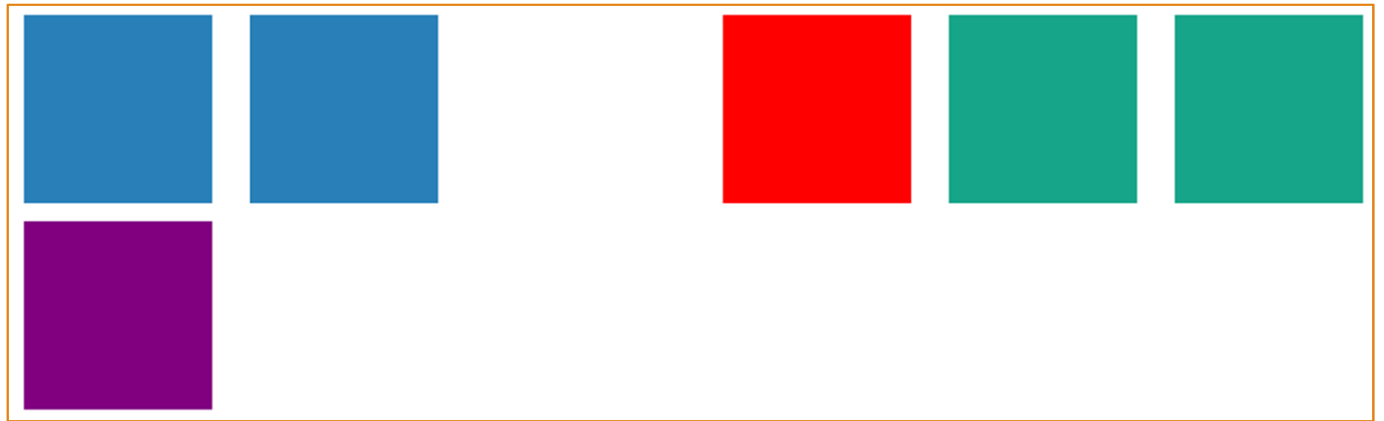
[enlace 3](#)

Propiedades de flujo de elementos.

La propiedad ***float*** de CSS permite alterar el flujo normal posicionando cada elemento a izquierda o derecha de su contenedor.

```
.a {  
  float: left;  
  margin: 10px;  
  height: 100px;  
  width: 100px;  
  background-color: #2980B9;  
}  
.b {  
  float: right;  
  margin: 10px;  
  height: 100px;  
  width: 100px;  
  background-color: #17A589;  
}  
.c {  
  float: right;  
  margin: 10px;  
  height: 100px;  
  width: 100px;  
  background-color: red;  
}  
.d {  
  clear: both;  
  margin: 10px;  
  height: 100px;  
  width: 100px;  
  background-color: purple;  
}
```

```
<body>  
  <div class="a">  
  </div>  
  <div class="a">  
  </div>  
  <div class="b">  
  </div>  
  <div class="b">  
  </div>  
  <div class="c">  
  </div>  
  <div class="d">  
  </div>  
</body>
```

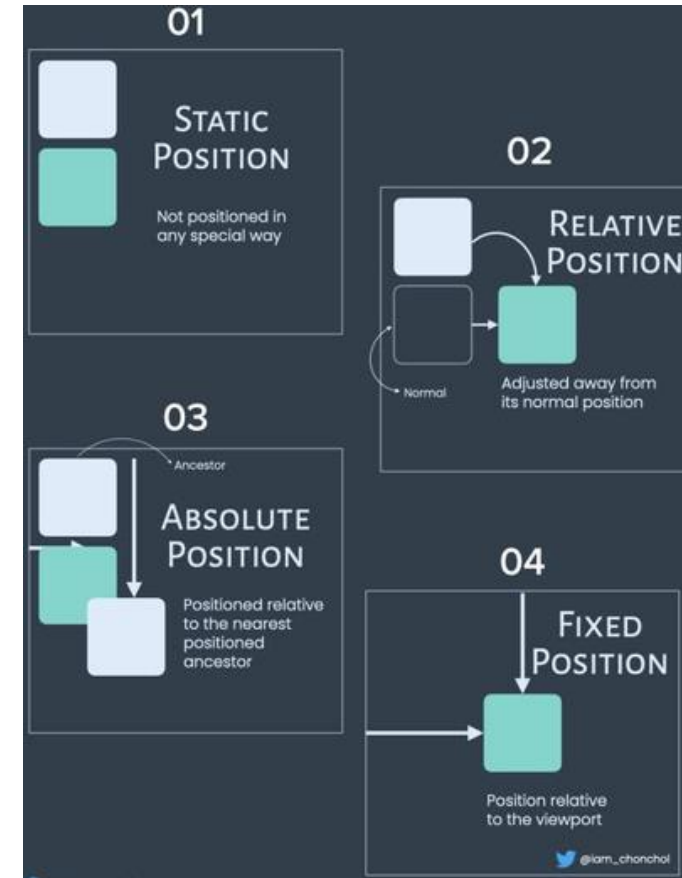


La propiedad **`clear: both;`** se utiliza para restaurar el flujo normal del documento y así los elementos dejan de flotar hacia la izquierda, la derecha o ambos lados.

Posicionamiento

Se establece mediante la propiedad ***position***.

- **static**: los elementos se posicionan de acuerdo al flujo normal de la página. Es la posición natural de los elementos. No son afectados por las propiedades ***top***, ***bottom***, ***left*** y ***right***.
- **relative**: los elementos se posicionan de forma relativa a su posición normal.
- **fixed**: los elementos se posicionan de forma relativa a la ventana del navegador. Su posición permanece fija aunque se desplace la ventana.
- **absolute**: los elementos se posicionan de forma relativa al primer elemento padre que tenga una posición distinta a static. En caso contrario se posiciona en relación a la ventana del navegador.



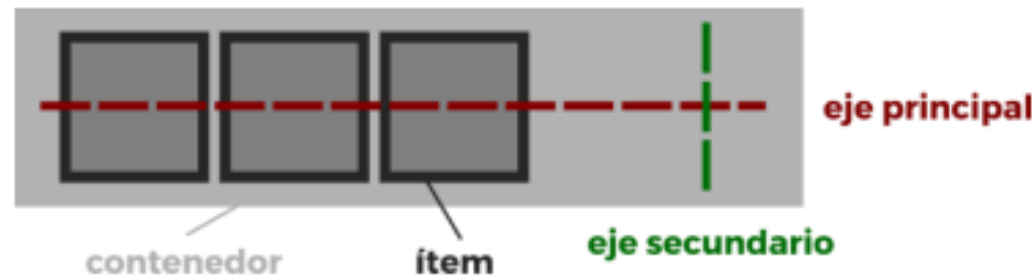
Estrategias de diseño (posicionamiento)

- **FlexBox** → modelo de diseño unidimensional que permite organizar los elementos en una sola dirección (fila o columna). Permite alinear y distribuir elementos de manera flexible. Facilita crear diseños responsivos y complejos.
- **CSS Grid** → modelo de diseño bidimensional que organiza los elementos en filas y columnas. Permite un mayor control sobre la posición y el tamaño de los elementos en el diseño.

Flexbox

Estructura

- **Contenedor:** Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles.
- **Eje principal:** Los contenedores flexibles tendrán una orientación principal específica. Por defecto, el eje principal del contenedor flexbox es en horizontal (en fila).
- **Eje secundario:** De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical (y viceversa).
- **Ítem:** Cada uno de los hijos que tendrá el contenedor en su interior.



Flexbox

Contenedor

Para activar el modo flexbox, indicamos sobre el elemento contenedor la propiedad `display` y especificamos el valor `flex` o `inline-flex` (dependiendo de cómo queramos que se comporte el contenedor):

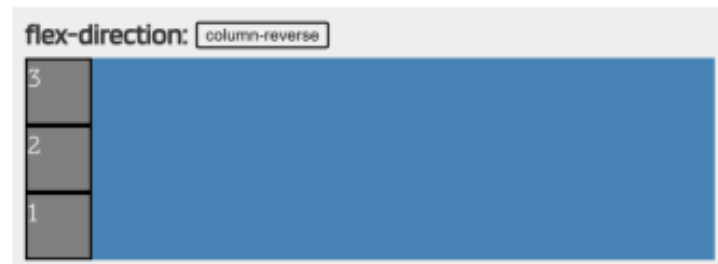
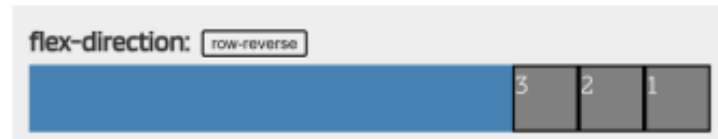
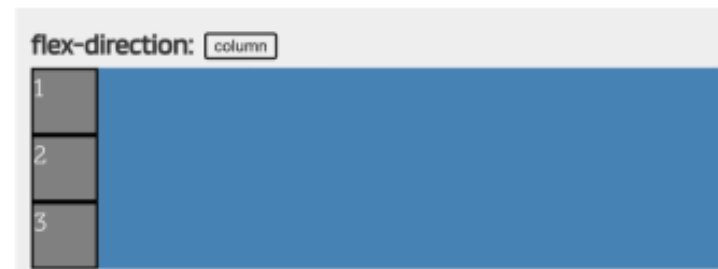
Tipo de elemento	Descripción
inline-flex	Establece un contenedor en línea, similar a inline-block (ocupa solo el contenido).
flex	Establece un contenedor en bloque, similar a block (ocupa todo el ancho del padre).

Flexbox

Dirección de ejes: *flex-direction*

Propiedad	Valor	Significado
flex-direction	row row-reverse column column-reverse	Cambia la orientación del eje principal.

Valor	Descripción
row	Establece la dirección del eje principal en horizontal.
row-reverse	Establece la dirección del eje principal en horizontal (invertido).
column	Establece la dirección del eje principal en vertical.
column-reverse	Establece la dirección del eje principal en vertical (invertido).

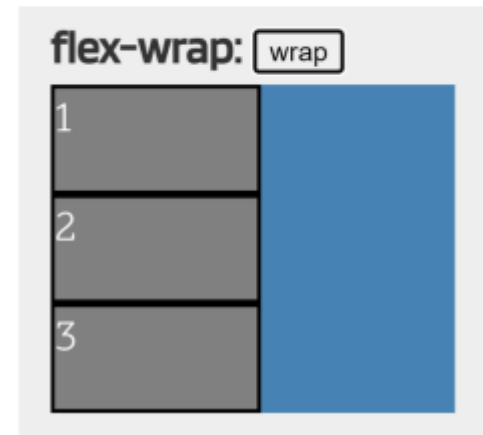
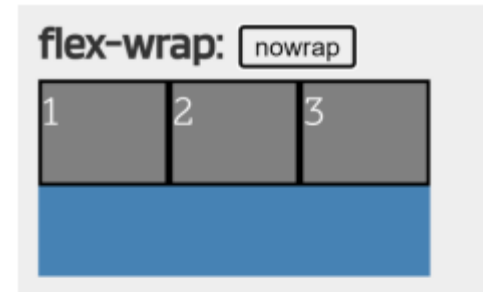


Flexbox

Control multilinea: *flex-wrap*

Propiedad	Valor	Significado
flex-wrap	nowrap wrap wrap-reverse	Evita o permite el desbordamiento (multilinea).

Valor	Descripción
nowrap	Establece los ítems en una sola línea (no permite que se desborde el contenedor).
wrap	Establece los ítems en modo multilinea (permite que se desborde el contenedor).
wrap-reverse	Establece los ítems en modo multilinea, pero en dirección inversa.



Flexbox

Propiedad atajo: ***flex-flow***

Permite resumir los valores de las propiedades *flex-direction* y *flex-wrap*, especificándolas en una sola propiedad:

```
flex-direction: row;  
flex-wrap: wrap;
```

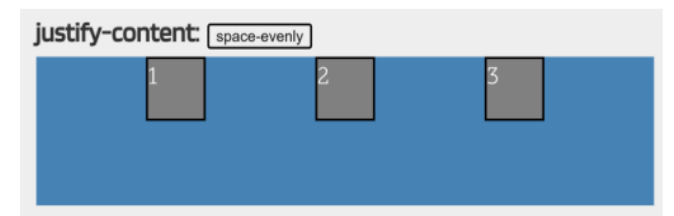
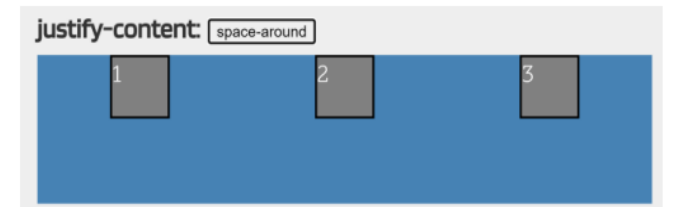
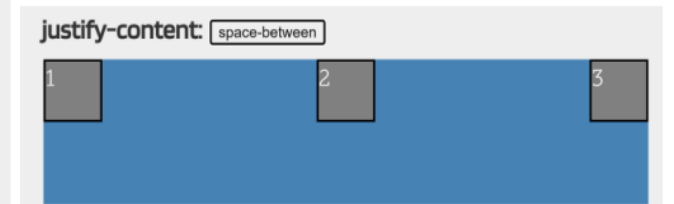
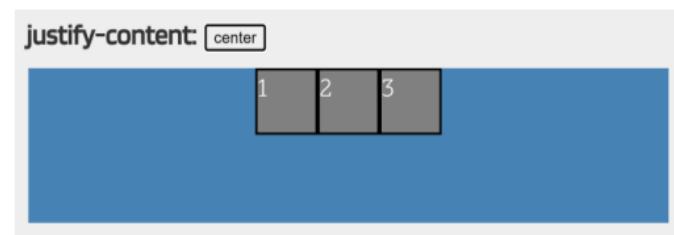
Puede resumirse en:

```
flex-flow: row wrap;
```

Flexbox

Alineamiento del eje principal: *justify-content*

Valor	Descripción
flex-start	Agrupar los ítems al principio del eje principal.
flex-end	Agrupar los ítems al final del eje principal.
center	Agrupar los ítems al centro del eje principal.
space-between	Distribuye los ítems dejando el máximo espacio para separarlos.
space-around	Distribuye los ítems dejando el mismo espacio alrededor de ellos (izq/dcha).
space-evenly	Distribuye los ítems dejando el mismo espacio (solapado) a izquierda y derecha.



Flexbox

Alineamiento del eje secundario: *align-items*

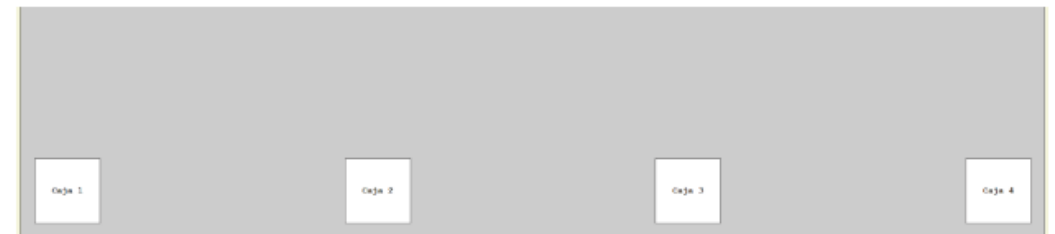
Se encarga de alinear los ítems en el eje secundario del contenedor (**para contenedores de una sola línea**)

Valor	Descripción
flex-start	Alinea los ítems al principio del eje secundario.
flex-end	Alinea los ítems al final del eje secundario.
center	Alinea los ítems al centro del eje secundario.
stretch	Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
baseline	Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.

```
align-items:center;
```



```
align-items:flex-end;
```

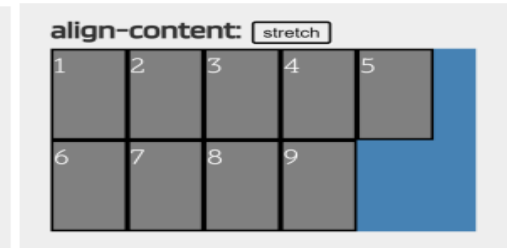
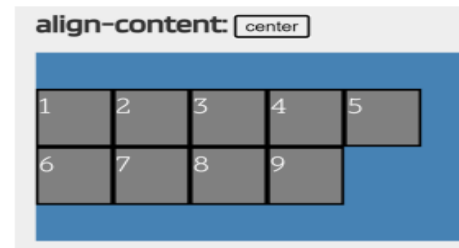
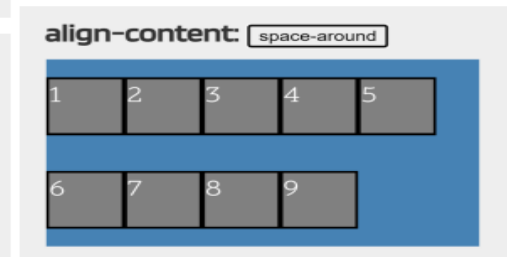
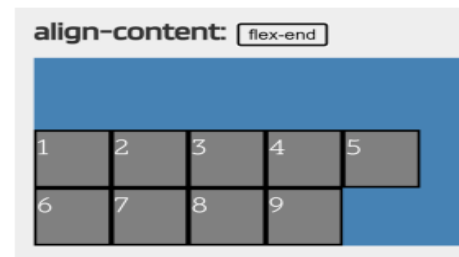
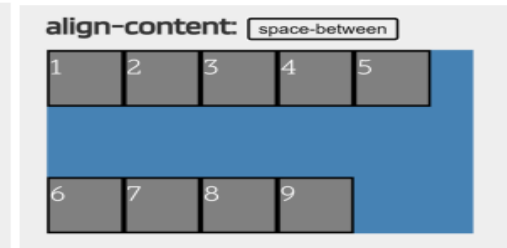
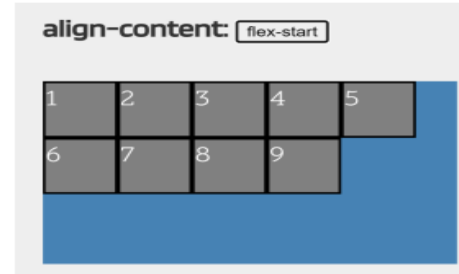


Flexbox

Alineamiento del eje secundario: *align-content*

Actúa sobre cada una de las líneas de un contenedor multi-línea (no tiene efecto sobre contenedores de una sola línea).

Valor	Descripción
flex-start	Agrupar los ítems al principio del eje principal.
flex-end	Agrupar los ítems al final del eje principal.
center	Agrupar los ítems al centro del eje principal.
space-between	Distribuye los ítems desde el inicio hasta el final.
space-around	Distribuye los ítems dejando el mismo espacio a los lados de cada uno.
stretch	Estira los ítems para ocupar de forma equitativa todo el espacio.



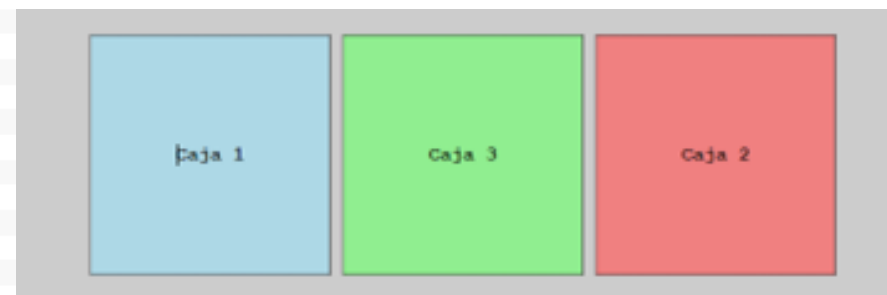
Flexbox

Propiedades de elementos hijos

- **order** → Indica el orden de aparición de los elementos.

```
.flex-container{  
  border: 1px solid black;  
  background: #ccc;  
  padding: 20px;  
  display: flex;  
  flex-flow: row wrap;  
  justify-content: center;  
  align-content: center;  
  height: 600px;  
}  
  
.c1{  
  background-color: lightblue;  
  order: 1;  
}  
  
.c2{  
  background-color: lightcoral;  
  order: 3;  
}  
  
.c3{  
  background-color: lightgreen;  
  order: 2;  
}
```

```
<section class="flex-container">  
  <div class="caja c1">  
    Caja 1  
  </div>  
  
  <div class="caja c2">  
    Caja 2  
  </div>  
  
  <div class="caja c3">  
    Caja 3  
  </div>  
</section>
```



Flexbox

Propiedades de elementos hijos

- ***flex-basis*** → Establece el tamaño base de cada elemento.

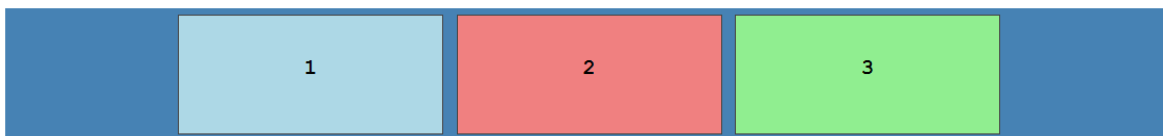
```
.container {  
  display: flex;  
  flex-flow: row wrap;  
  justify-content: center;  
  background-color: steelblue;  
  height: 100px;  
}
```

```
.item1 {  
  background-color: lightblue;  
  flex-basis: 200px;  
}
```

```
.item2 {  
  background-color: lightcoral;  
  flex-basis: 200px;  
}
```

```
.item3 {  
  background-color: lightgreen;  
  flex-basis: 200px;  
}
```

Si no hay espacio los elementos desbordan a la fila siguiente



```
<div class="container">  
  <div class="item item1">1</div>  
  <div class="item item2">2</div>  
  <div class="item item3">3</div>  
</div>
```

Los elementos no crecen para adquirir el espacio sobrante.

Flexbox

Propiedades de elementos hijos

- ***flex-grow*** → Establece la proporción en que cada elemento toma el espacio sobrante.

```
.item1 {  
  background-color: lightblue;  
  flex-basis: 200px;  
  flex-grow: 1;  
}
```

```
.item2 {  
  background-color: lightcoral;  
  flex-basis: 200px;  
  flex-grow: 2;  
}
```

```
.item3 {  
  background-color: lightgreen;  
  flex-basis: 200px;  
  flex-grow: 3;  
}
```

Si no hay espacio los elementos desbordan a la fila siguiente



```
<div class="container">  
  <div class="item item1">1</div>  
  <div class="item item2">2</div>  
  <div class="item item3">3</div>  
</div>
```

Los elementos crecen para adquirir el espacio a partir de su tamaño en proporción al valor de la propiedad ***flex-grow***.

Flexbox

Propiedades de elementos hijos

- ***flex-shrink*** → Establece la proporción en que cada elemento se encoge si no hay espacio suficiente

```
.container {  
  display: flex;  
  flex-flow: row nowrap;  
  justify-content: center;  
  background-color: steelblue;  
  height: 100px;  
}  
  
.item1 {  
  background-color: lightblue;  
  flex-basis: 300px;  
  flex-shrink: 1;  
}  
  
.item2 {  
  background-color: lightcoral;  
  flex-basis: 300px;  
  flex-shrink: 2;  
}  
  
.item3 {  
  background-color: lightgreen;  
  flex-basis: 300px;  
  flex-shrink: 1;  
}
```

Para que los elementos se encojan, no deben poder desbordar.



```
<div class="container">  
  <div class="item item1">1</div>  
  <div class="item item2">2</div>  
  <div class="item item3">3</div>  
</div>
```

Los elementos se encogen al reducirse el tamaño en base al valor de la propiedad ***flex-shrink***

Flexbox

La propiedad abreviada ***flex*** permite indicar los valores para las propiedades ***flex-basis***, ***flex-grow*** y ***flex-shrink***.

```
.flex-item {  
  flex: 1 0 100px; /* flex-grow: 1, flex-shrink: 0, flex-basis: 100px */  
}
```

```
.item1 {  
  background-color: lightblue;  
  flex: 1 0 200px;  
}  
  
.item2 {  
  background-color: lightcoral;  
  flex: 2 0 200px;  
}  
  
.item3 {  
  background-color: lightgreen;  
  flex: 3 0 200px;  
}
```

Media Querys

Una media query es una característica de CSS que permite aplicar estilos condicionalmente según:

- Características del dispositivo
- Tamaño de la pantalla
- Resolución
- Orientación

Esto es fundamental para el diseño responsive, ya que permite que una página web se vea bien en una variedad de dispositivos y tamaños de pantalla.

Formato:

```
@media not|only mediatype and (media feature) { CSS Code }
```

Media Querys

Ejemplos:

- **max-width:** Aplica los estilos si el ancho de la pantalla es menor o igual a un valor específico.
- **min-width:** Aplica los estilos si el ancho de la pantalla es mayor o igual a un valor específico.

```
@media (max-width: 768px) { ... }  
@media (min-width: 1024px) { ... }
```

- **orientation:** Permite aplicar estilos basados en la orientación del dispositivo, es decir, si está en horizontal o vertical:

```
@media (orientation: landscape) { ... }  
@media (orientation: portrait) { ... }
```

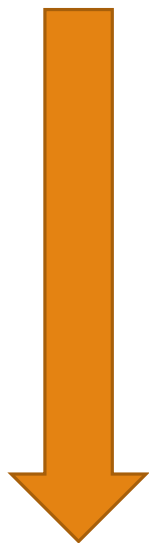
- **resolution:** Se usa para apuntar a pantallas con resoluciones específicas:

```
@media (min-resolution: 300dpi) { ... }
```

Mobile-First / Desktop-First

Estrategias

```
/* Estilos base para dispositivos móviles */
body {
  background-color: red;
}
p:after {
  color: white;
  content: "Móvil";
}
/* Media query para tabletas */
@media (min-width: 600px) {
  body {
    background-color: green;
  }
  p:after {
    color: white;
    content: "Tablet";
  }
}
/* Media query para desktop */
@media (min-width: 1024px) {
  body {
    background-color: blue;
  }
  p:after {
    color: white;
    content: "Escritorio";
  }
}
```



```
/* Estilos base para escritorios */
body {
  background-color: blue;
}
p:after {
  color: white;
  content: "Escritorio";
}
/* Media query para tabletas (por ejemplo, anchura entre 600px y 1024px) */
@media (min-width: 600px) and (max-width: 1023px) {
  body {
    background-color: green;
  }
  p:after {
    content: "Tablet";
  }
}
/* Media query para dispositivos móviles (por ejemplo, anchura menor a 600px) */
@media (max-width: 599px) {
  body {
    background-color: red;
  }
  p:after {
    content: "Móvil";
  }
}
```

CSS Grid

CSS Grid es un sistema de diseño bidimensional en CSS que permite dividir la página en una rejilla a partir de la cual se pueden posicionar los diferentes elementos de manera muy sencilla.

En contraposición:

Flexbox se crea diseños siguiendo una dimensión (fila / columna), mientras que CSS Grid crea diseños dividiendo el espacio en varias filas y columnas.

CSS Grid

- **Contenedor:** El contenedor recibe la propiedad *display: grid/inline-grid*.
- **Elementos:** Son los elementos dentro del contenedor que se distribuyen en la cuadrícula repartidos en filas y columnas. (*grid items*).

```
<!DOCTYPE html>
<html>
<head>
<style>
.contenedor{
  display: grid;
}
</style>
</head>
<body>
<div class="contenedor"> <!-- contenedor padre-->
  <div class="rejilla">Item 1</div> <!-- Ítems del grid -->
  <div class="rejilla">Item 2</div>
  <div class="rejilla">Item 3</div>
  <div class="rejilla">Item 4</div>
</div>
</body>
</html>
```

CSS Grid

Propiedades del contenedor.

- **grid-template-columns** → Define el número y el tamaño de las columnas en la cuadrícula
- **grid-template-rows** → Define el número y el tamaño de las filas de la cuadrícula.

Propiedad	Valor	Descripción
grid-template-columns	<i>[col1] [col2] ...</i>	Tamaño de cada columna
grid-template-rows	<i>[fila1] [fila2] ...</i>	Tamaño de cada fila

- Ejemplo:

```
grid-template-columns: 70% 30%;           /* 2 columnas a 70% - 30% del ancho contenedor */
grid-template-rows: 30px 20px auto 20px;  /* 4 filas */
```

CSS Grid

Ejemplo: Rejilla de una única fila con 4 columnas a diferentes tamaños.

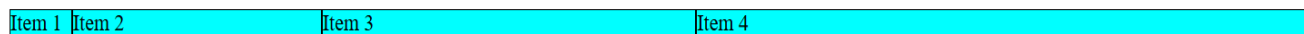
```
.contenedor{
  display: grid;
  grid-template-columns: auto auto auto auto;
}
.contenedor{
  display: grid;
  grid-template-columns: 50px 200px 300px 500px;
}
.contenedor{
  display: grid;
  grid-template-columns: 1fr 2fr 1fr 3fr;
}
.rejilla {
  border: 1px solid black;
  background-color: cyan;
}
```

```
<h2> grid-template-columns: auto auto auto auto;</h2>
<div class="contenedor"> <!-- contenedor padre-->
<div class="rejilla">Item 1</div> <!-- Ítems del grid -->
<div class="rejilla">Item 2</div>
<div class="rejilla">Item 3</div>
<div class="rejilla">Item 4</div>
</div>
```

grid-template-columns: auto auto auto auto;



grid-template-columns: 50px 200px 50px 500px;



grid-template-columns: 1fr 2fr 1fr 3fr;



CSS Grid

Propiedades del contenedor.

- **column-gap**: Define el espaciado entre columnas
- **row-gap**: Define el espaciado entre filas.

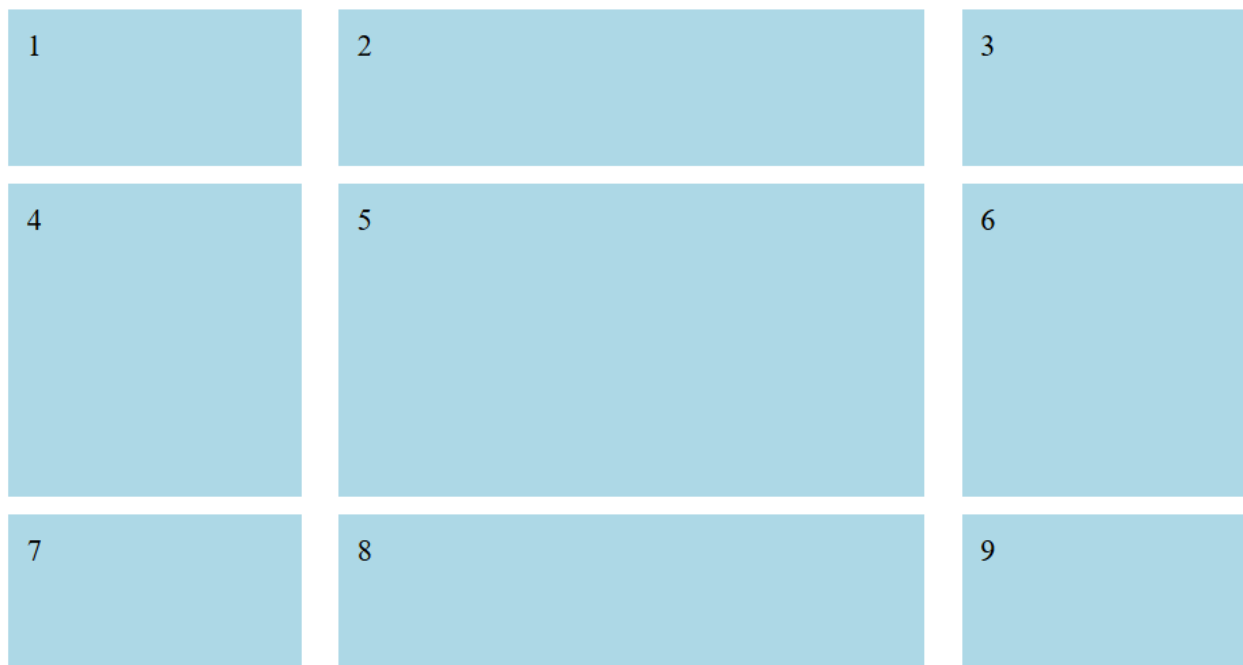
Propiedad	Descripción
column-gap	Espaciado entre columnas
row-gap	Espaciado entre filas

- **grid-gap**: Propiedad abreviada que permite definir a la vez el espaciado fila/columna

CSS Grid

Ejemplo: 3 filas x 3 columnas con diferentes proporciones y espaciados.

```
.grid-container {  
  height: 50vh;  
  display: grid;  
  grid-template-columns: 1fr 2fr 1fr;  
  grid-template-rows: 1fr 2fr 1fr;  
  column-gap: 20px;  
  row-gap: 10px;  
  /* grid-gap: 10px 20px; */  
}  
.grid-item {  
  background-color: lightblue;  
  padding: 10px;  
}  
  
<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>
```



CSS Grid

Propiedades de los elementos:

- **grid-column**: Controla la posición y el tamaño de un elemento en el eje horizontal (las columnas).

`grid-column: <start-line> / <end-line>;`

- `<start-line>`: Línea donde empieza el elemento (número de línea, nombre de línea, o palabra clave como `span`).
- `<end-line>`: Línea donde termina el elemento (similar al anterior).
- **grid-row**: Funciona de manera similar a *grid-column*, pero en el eje vertical (las filas).

CSS Grid

Ejemplo 2: 3 filas x 3 columnas con diferentes proporciones y espaciados.

```
.grid-layout{
  display: grid;
  grid-template-columns: 70% 30%;          /* 2 columnas a 70% - 30% del ancho contenedor */
  grid-template-rows: 30px 20px auto 20px; /* 4 filas a 30px, 20px auto y 20px de alto */
  width: 80%;
  height: 80vh;
  margin: 0 auto;
  row-gap: 10px; /* Espacio solo entre filas */
  column-gap: 20px; /* Espacio solo entre columnas */
}
```

```
.caja{
  border: 2px solid black;
  background-color: lightblue;
  text-align: center;
}
```

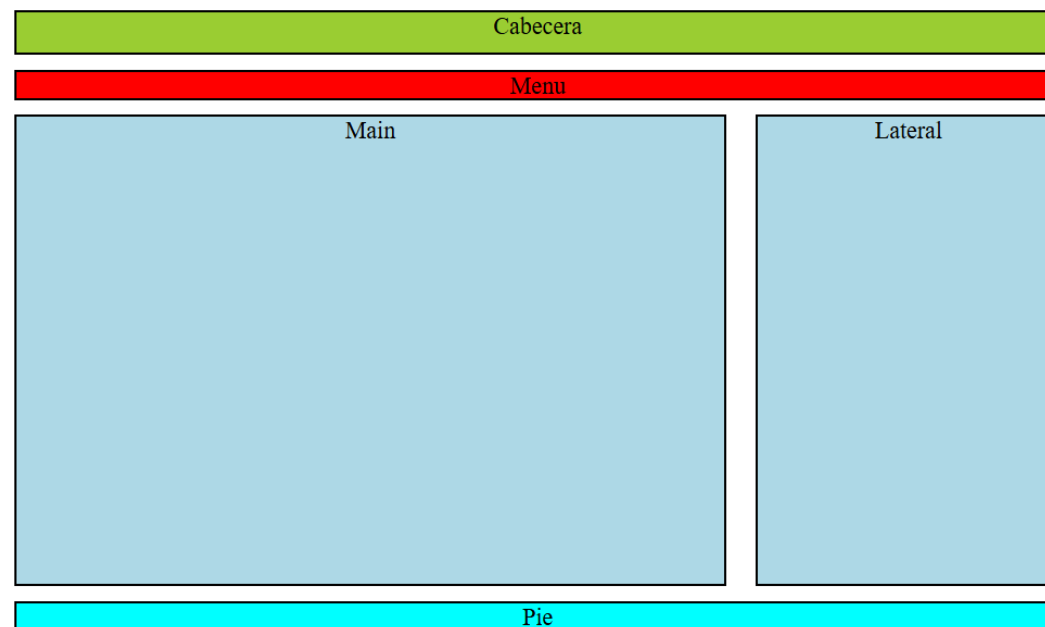
```
.c1 {
  background-color: yellowgreen;
  grid-column: 1/3;
}
```

```
.c2 {
  background-color: red;
  grid-column: 1/3;
}
```

```
.c5 {
  background-color: aqua;
  grid-column: 1/3;
}
```

```
<div class="grid-layout">
  <div class="caja c1">Cabecera</div>
  <div class="caja c2">Menu</div>
  <div class="caja c3">Main</div>
  <div class="caja c4">Lateral</div>
  <div class="caja c5">Pie</div>
</div>
```

Los elementos se extienden de la columna 1 a la 3 ocupando dos columnas.



CSS Grid

- **grid-template-areas** (**Contenedor**) → Esta propiedad permite nombrar áreas de la cuadrícula, haciendo que el diseño sea más legible:

```
.grid-container {  
  display: grid;  
  grid-template-columns: 200px 1fr; /* 2 columnas: una fija de 200px y otra flexible */  
  grid-template-rows: 100px 1fr 100px; /* 3 filas: header de 100px, contenido flexible y footer de 100px */  
  grid-template-areas:  
    "header header"  
    "sidebar main"  
    "footer footer";  
  grid-gap: 10px; /* Espaciado entre las áreas */  
  height: 100vh; /* Altura del viewport completa */  
}
```

El número de veces y ubicación de su denominación determina su posición y distribución entre las filas y columnas

CSS Grid

- **grid-area (Elementos)** → Esta propiedad determina a qué área pertenece un elemento.

```
.grid-container {  
  display: grid;  
  grid-template-columns: 200px 1fr; /* 2 columnas:  
  grid-template-rows: 100px 1fr 100px; /* 3 filas:  
  grid-template-areas:  
    "header header"  
    "sidebar main"  
    "footer footer";  
  grid-gap: 10px; /* Espaciado entre las áreas */  
  height: 100vh; /* Altura del viewport completa */  
}
```

/* Estilos para las áreas de la cuadrícula */

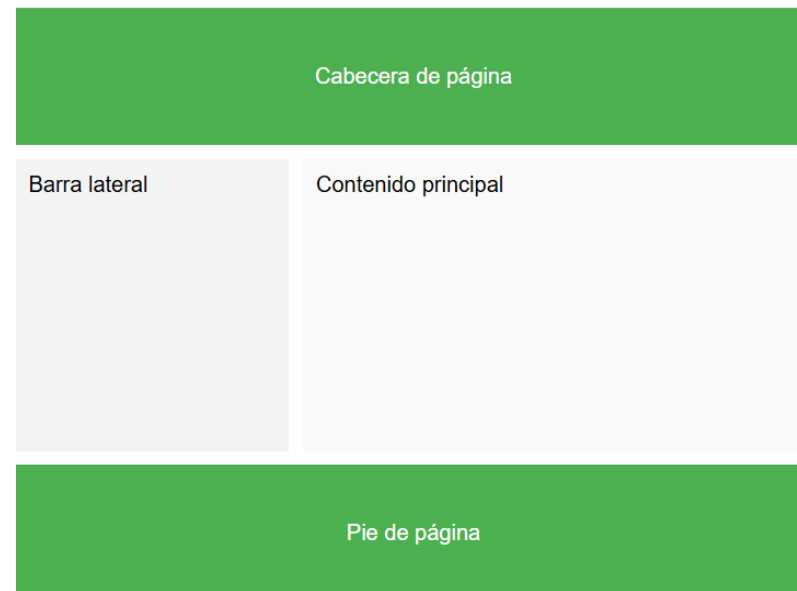
```
.cabecera {  
  grid-area: header;  
  background-color: #4CAF50;  
  color: white;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

```
.lateral {  
  grid-area: sidebar;  
  background-color: #f4f4f4;  
  padding: 10px;  
}
```

```
.principal {  
  grid-area: main;  
  background-color: #f9f9f9;  
  padding: 10px;  
}
```

```
.pie {  
  grid-area: footer;  
  background-color: #4CAF50;  
  color: white;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

```
<div class="grid-container">  
  <header class="cabecera">Cabecera de página</header>  
  <nav class="lateral">Barra lateral</nav>  
  <main class="principal">Contenido principal</main>  
  <footer class="pie">Pie de página</footer>  
</div>
```



Bordes redondeados

Tenemos la propiedad *border-radius*, que permite definir bordes redondeados en las esquinas, especificando las medidas del radio que deben darse a la curva de las esquinas.

```
border-radius: 10px 20px 30px 40px;
```

- Esquina superior izquierda: 10px
- Esquina superior derecha: 20px
- Esquina inferior derecha: 30px
- Esquina inferior izquierda: 40px



Los bordes redondeados pueden emplearse tanto con capas, botones o hipervínculos para conseguir un aspecto más vistoso.

```
apa {  
  width: 300px;  
  height: 150px;  
  background-color: lightgreen;  
  text-align: justify;  
  padding: 20px;  
  border-radius: 20px;  
}
```

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Dolore, deserunt repellendus amet, consequatur quasi sint, nisi reprehenderit enim provident corporis quaerat! Ea officia dolorum alias, similique harum rerum! Quo, veniam.

Sombras de caja

En CSS3, las sombras se crean usando la propiedad *box-shadow* para elementos en bloque (como div, section, etc).

```
box-shadow: desplazamientoX desplazamientoY difuminado extensión color;
```

```
.capa {  
  width: 300px;  
  height: 150px;  
  background-color: lightgreen;  
  text-align: justify;  
  padding: 20px;  
  border-radius: 20px;  
  box-shadow: 5px 5px 10px 5px rgba(0, 0, 0, 0.5);  
}
```

Lorem ipsum dolor sit amet consectetur,
adipisicing elit. Dolore, deserunt repellendus
amet, consequatur quasi sint, nisi
reprehenderit enim provident corporis quaerat!
Ea officia dolorum alias, similique harum
rerum! Quo, veniam.

Sombras de texto

Para aplicar sombras a textos en CSS3, se utiliza la propiedad *text-shadow*. Esta propiedad permite crear efectos de sombra alrededor del texto, lo que añade profundidad y estilo visual.

```
text-shadow: desplazamientoX desplazamientoY difuminado color;
```

Ejemplo:

```
h1 {  
  font-size: 50px;  
  color: ■ navy;  
  text-shadow: 2px 2px 5px ■ gray;  
}
```

Sombra básica

Transiciones

Las transiciones en CSS nos permiten aplicar un cambio de estilo gradual a los elementos del documento HTML pudiendo indicar la duración del cambio entre estilos.

```
transition:[propiedad a modificar] [Duración] [Tipo de animación] [Retardo];
```

- **[propiedad]:** Propiedad sobre la que se aplica la transición.
- **[duración]:** Duración de la transición desde el estado inicial al final *expresada en segundos*.
- **[tipo]:** Modo en que se aplica la transición a lo largo del tiempo.
- **[retardo]:** Tiempo que el navegador espera antes iniciar la transición *expresada en segundos*.

Ejemplo:

```
transition: background-color 0.2s ease-in-out;
```

Transiciones

Una vez definida una transición sobre la propiedad de un elemento, cualquier cambio en su valor se aplica mediante la transición.

```
.caja {  
  border: 1px solid ■black;  
  text-align: center;  
  line-height: 50px;  
  width: 50%;  
  margin: 0 auto;  
  height: 50px;  
  
  /* Color de fondo original de la capa */  
  background-color: ■rgb(146, 202, 221);  
  /* Configuración de la transición sobre propiedad 'background-color' */  
  transition: background-color 0.2s ease-in-out;  
}
```

Se aplica transición a la propiedad
'background-color'

```
.caja:hover {  
  /* Cambio color de fondo al pasar el ratón por encima */  
  /* Se aplica transición sobre el cambio de esta propiedad */  
  background-color: ■lightcyan;  
}
```

El cambio de color de fondo al pasar el
ratón por encima se aplica gradualmente.

Propiedades y tipos de transición

Propiedades soportadas para transición:

```
background-color
border
border-radius
color
top
bottom
left
right
box-shadow
width
height
line-height
margin
opacity
word-spacing
letter-spacing
fill
padding
stroke
text-shadow
vertical-align
visibility
z-index
```

Tipos de animación → Determina la 'velocidad' de la transición durante su duración:

- **linear**: la velocidad de la animación es uniforme en todo el recorrido.
- **ease**: la velocidad se acelera al inicio, luego se retarda un poco y se acelera al final de nuevo.
- **ease-in**: la animación empieza lentamente y va aumentando progresivamente.
- **ease-out**: la animación empieza muy rápida y va descendiendo progresivamente.
- **ease-in-out**: la animación empieza y acaba lentamente, y es en la parte central del recorrido donde la velocidad es más rápida.

Transformaciones

la propiedad ***transform*** permite aplicar efectos como cambios en su posición, rotación, escalado, traslación e inclinación:

```
transform: tipo_de_transformación(valor) otro_tipo_de_transformación(valor);
```

- **scale**: modifica el tamaño de los elementos con uno o dos valores, que representan la cantidad de escala que se aplica en cada dirección: ***scale (x)*** o ***scale(x,y)***.

```
transform: scale(0.5); /* Escala el elemento a la mitad */
```

- **Translate**: cambia la posición del elemento hacia la izquierda, derecha, arriba o abajo. La función se establece con uno o dos valores: ***translate(x)*** o ***translate(x,y)***

```
transform: translate(10px); /* Traslada el elemento 10px hacia la derecha */
```

- **Rotate**: gira o rota los elementos en grados: ***rotate(v)***

```
transform: rotate(45deg); /* Rota el elemento 45 grados */
```

Transformaciones

la propiedad *transform* permite aplicar efectos como cambios en su posición, rotación, escalado, traslación e inclinación.

Función	Descripción	Ejemplo
Scale	Se establece con uno o dos valores, que representan la cantidad de escala que se aplica en cada dirección: <code>scale(x)</code> o <code>scale(x,y)</code> . Cuando el valor está fuera del rango [-1, 1], el elemento crece a lo largo de esa dimensión. Cuando está dentro del rango el elemento se encoge.	<code>transform: scale(0.5)</code>
Translate	Cambia la posición del elemento hacia la izquierda, derecha, arriba o abajo. La función <code>translate()</code> se establece con uno o dos valores: <code>translate(x)</code> o <code>translate(x,y)</code> . Sus valores pueden estar definidos en píxeles, porcentajes,...	<code>transform: translate(10px)</code>
Rotate	Gira o rota los elementos en grados: <code>rotate(v)</code> .	<code>transform: rotate(45deg)</code>
Skew	Distorsiona los elementos según el ángulo en grados. La función <code>skew()</code> se establece con uno o dos valores: <code>skew(x)</code> o <code>skew(x,y)</code> .	<code>transform: skew(45deg)</code>
Matrix	Mueve o transforma los elementos con precisión de píxel. La función <code>matrix()</code> se establece con seis valores numéricos: <code>matrix(a,b,c,d,x,y)</code> . Los dos últimos valores representan la traslación y los primeros la transformación lineal.	<code>transform: matrix(0.5, 0.1, 0.5, 1, 10, -2)</code>

Animaciones

- La propiedad **transition** en CSS permite realizar cambios suaves entre dos estados de un elemento cuando ocurre un evento, como pasar el cursor sobre él (*:hover*). Es ideal para efectos básicos como cambios en el color, tamaño o posición, pero solo anima entre un estado inicial y final, sin pasos intermedios.
- La propiedad **transform** modifica instantáneamente la apariencia de un elemento, permitiendo rotarlo, escalarlo o moverlo en el espacio. Aunque no genera animaciones por sí mismo, puede combinarse con **transition** o **animation** para lograr efectos más elaborados.
- La propiedad **animation** permite definir múltiples estados intermedios en una animación y controlar aspectos como la duración, la repetición y la dirección. A diferencia de **transition**, las animaciones pueden ejecutarse automáticamente al cargar la página o repetirse indefinidamente sin requerir la interacción del usuario.

Animaciones

Propiedades:

- **animation-name** → Nombre de la animación necesaria para la regla `@keyframes` que describe los fotogramas de la animación.

Los **@keyframes** son un conjunto de fotogramas clave que describen cómo se muestra cada elemento animado durante la secuencia de la animación. La sintaxis es la siguiente:

```
.mi-elemento {  
  animation-name: mi-animacion;  
}
```

```
@keyframes mi-animacion {  
  from { opacity: 0; }  
  to { opacity: 1; }  
}
```

```
@keyframes identifier {  
  from {  
    ...  
  }  
  percentage {  
    ...  
  }  
  to {  
    ...  
  }  
}
```

- **identifier** Nombre que identifica la animación en la propiedad: **animation-name**.
- **from** → Desde (por ejemplo: desde 0%).
- **to** → Hasta (por ejemplo hasta 100%).
- **percentage** → Porcentaje intermedio de las veces que va a ocurrir una animación.

Animaciones

Propiedades:

- **animation-delay** → Determina un tiempo de espera antes de que la animación comience.

```
.mi-elemento {  
  animation-delay: 500ms; /* Espera medio segundo antes de comenzar */  
}
```

- **animation-direction** → Indica si la animación debe alternar la dirección en cada ciclo : *normal*, *reverse*, *alternate*, *alternate-reverse*.

```
.mi-elemento {  
  animation-direction: alternate; /* Va y viene */  
}
```

- **animation-duration** → Establece cuánto tiempo tarda en completar un ciclo de la animación en segundos (s) o milisegundos (ms).

```
.mi-elemento {  
  animation-duration: 2s; /* Completa la animación en 2 segundos */  
}
```

- **animation-iteration-count** → Define cuántas veces se repetirá la animación. *infinite* para una animación sin fin.

```
.mi-elemento {  
  animation-iteration-count: infinite; /* Repite la animación indefinidamente */  
}
```

Animaciones

Propiedades:

- **animation-play-state** → Permite pausar y reanudar la reproducción de la animación: *running, paused*.

```
.mi-elemento:hover {  
  animation-play-state: paused; /* Pausa la animación al pasar el mouse */  
}
```

- **animation-timing-function** → Controla el ritmo de la animación, definiendo cómo se acelera y desacelera durante su ejecución: *linear, ease, ease-in, ease-out, ease-in-out*:

```
.mi-elemento {  
  animation-timing-function: ease-in-out; /* Comienza lentamente, se acelera en el  
medio, y termina lentamente */  
}
```

- **animation-fill-mode** → Decide los estilos que se aplican a un elemento antes y después de su animación: *none, forwards, backwards, both*.

```
.mi-elemento {  
  animation-fill-mode: forwards; /* Mantiene los estilos del último keyframe después de  
finalizar */  
}
```

Modos de fusión

Los modos de fusión son una propiedad que permite mezclar el color de fondo de un elemento con el color del elemento superpuesto, que puede ser un texto o un contenedor.

Se emplea la propiedad: ***mix-blend-mode***.

```
.contenedor-superpuesto {  
  mix-blend-mode: <modo-de-mezcla>;  
}
```

Los diferentes modos de mezcla (fusión) *manipulan la transparencia y la opacidad de los elementos*, logrando efectos de fusión, aclarado, superposición, saturación o contraste, entre otros.

Modos de fusión

Blend Mode	Descripción	Características
<i>multiply</i>	Multiplica los colores de la capa superior con los de la capa inferior.	– Oscurece la imagen resultante.
<i>screen</i>	Invierte los colores de la capa superior y los multiplica con los de la capa inferior.	– Aclara las áreas de la imagen.
<i>overlay</i>	Combina Multiply y Screen, oscureciendo o aclarando la capa inferior dependiendo de sus colores.	– Produce un efecto de contraste y saturación en la imagen resultante.
<i>darken</i>	Conserva los colores más oscuros de las dos capas.	– Selecciona el color más oscuro de cada píxel de ambas capas.
<i>lighten</i>	Conserva los colores más claros de las dos capas.	– Selecciona el color más claro de cada píxel de ambas capas.
<i>color-dodge</i>	Aclara los colores de la capa inferior dependiendo de los colores de la capa superior.	– Aclara las áreas de la imagen de la capa inferior.
<i>color-burn</i>	Oscurece los colores de la capa inferior dependiendo de los colores de la capa superior.	– Oscurece las áreas de la imagen de la capa inferior.
<i>difference</i>	Resta los colores de la capa superior de los de la capa inferior.	– Produce un efecto de inversión de colores.
<i>exclusion</i>	Similar a Difference, pero con una apariencia menos extrema.	– Combina los colores de ambas capas con un efecto de exclusión.
<i>hard-light</i>	Combina Multiply y Screen, pero de una manera más intensa.	– Produce un efecto más fuerte de contraste y saturación.
<i>soft-light</i>	Similar a Overlay, pero con una apariencia más suave.	– Produce un efecto más suave de contraste y saturación.
<i>hue</i>	Preserva el brillo y la saturación de la capa inferior, pero adopta el tono de la capa superior.	– Aplica el tono de la capa superior a la capa inferior.
<i>saturation</i>	Preserva el brillo y el tono de la capa inferior, pero adopta la saturación de la capa superior.	– Aplica la saturación de la capa superior a la capa inferior.
<i>color</i>	Preserva el brillo de la capa inferior, pero adopta el tono y la saturación de la capa superior.	– Aplica el tono y la saturación de la capa superior a la capa inferior.
<i>luminosity</i>	Preserva el tono y la saturación de la capa inferior, pero adopta el brillo de la capa superior.	– Aplica el brillo de la capa superior a la capa inferior.

Modos de fusión

Ejemplo de modo de fusión:



lighten

overlay

darker

difference



Filtros

Los filtros son efectos visuales que se pueden aplicar a elementos de la página mediante la propiedad ***filter***:

```
selector {  
  filter: <función-de-filtro>(<valor>) ;  
}
```

Cada función de filtro representa un determinado efecto especial cuya aplicación se ajuste en base al valor indicado como argumento.

Filtros

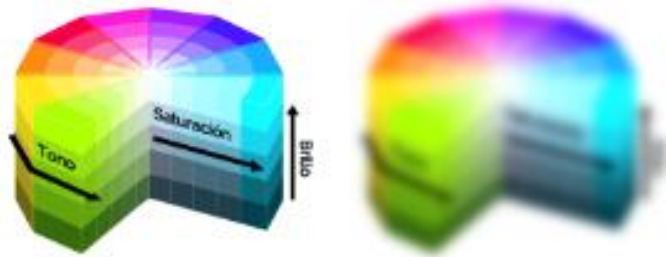
Funciones de filtros:

Función	Descripción	Valores
blur(<valor>)	Aplica un desenfoque al elemento	<valor> cantidad de desenfoque a aplicar
brightness(<valor>)	Ajusta el brillo del elemento	<valor> 0: completamente oscuro, 1: original
contrast(<valor>)	Ajusta el contraste del elemento	<valor> 0: completamente gris, 1: original
grayscale(<valor>)	Convierte el elemento a escala de grises	<valor> 0: original, 1: escala de grises
hue-rotate(<valor>)	Gira el espectro de colores del elemento	<valor> es la cantidad de grados en sentido horario 0 a 360deg
invert(<valor>)	Invierte los colores del elemento	<valor> 0: original, 1: invertido
opacity(<valor>)	Ajusta la opacidad del elemento	<valor> 0: transparente, 1: original
saturate(<valor>)	Ajusta la saturación del elemento	<valor> 0: grisáceo , 100%: original
sepia(<valor>)	Aplica un tono sepia al elemento	<valor> 0: original, 1: sepia completo

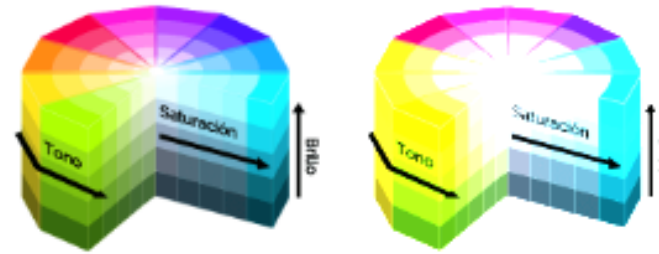
Filtros

Ejemplos de filtros:

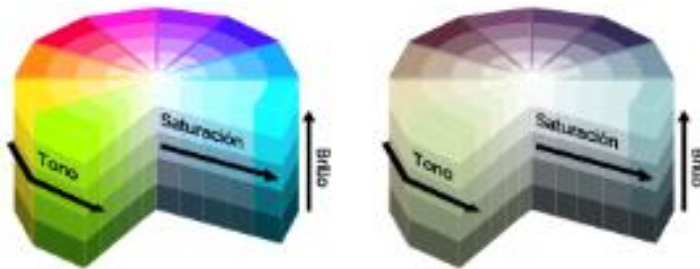
- `Blur(5px)`



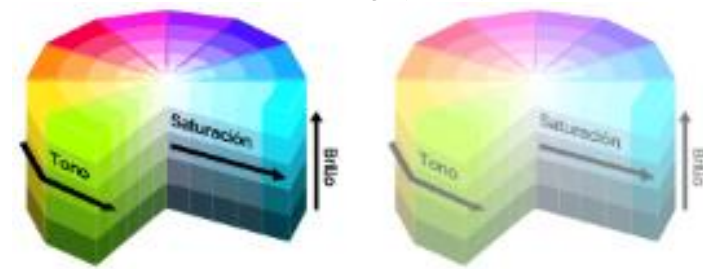
- `brightness(0.5)`



- `saturate(0.5)`



- `opacity(0.5)`



Filtros VS Modos de fusión

Diferenciar los filtros de los modos de fusión:

- **Modos de fusión:** mezcla entre dos elementos superpuestos.
- **Filtros:** efectos en una imagen o elemento.

Clave:

Los modos de fusión son más útiles cuando se necesita mezclar dos o más elementos superpuestos, mientras que los filtros son más adecuados para aplicar efectos globales a elementos individuales, como imágenes.

Prefijos propiedades no estandarizadas

El uso de prefijos es/era una medida empleada anteriormente para habilitar propiedades CSS nuevas que no eran compatibles con todos los navegadores.

Prefijo	Navegador
-moz-	Firefox
-webkit-	Safari y Chrome
-o-	Opera
-khtml-	Konqueror
-ms-	Internet Explorer y Microsoft Edge

```
-moz-transform: rotate(20deg); /* Firefox */  
-webkit-transform: rotate(20deg); /* Safari y Chrome */  
-o-transform: rotate(20deg); /* Opera */  
-ms-transform: rotate(20deg); /* Internet Explorer y Microsoft Edge */
```

- Web de compatibilidad CSS navegadores: <https://caniuse.com/>
- VisualCode puede instalar la extensión “*autoprefixer*” para incluir prefijos en propiedades CSS que no son estables/compatibles con todos los navegadores.

Web referencia online-estilos CSS

<https://htmlcheatsheet.com/css/>



Herramientas útiles.

Probar el diseño en diferentes navegadores o emplear herramientas como ***browserling***.

<https://www.browserling.com/>



Validar errores de código y CSS mediante herramienta de validación:

<https://jigsaw.w3.org/css-validator/>



Comprobar soporte de elementos HTML y propiedades CSS en los navegadores:

<https://caniuse.com/>

DevTools

La pestaña (*Elements*) de las herramientas del desarrollador (DevTools) del navegador permite examinar y modificar los estilos efectivos aplicados a cada elemento HTML:

