

¿Qué es CSS y qué aprenderé en este módulo?

CSS (hojas de estilo en cascada) es un lenguaje para diseñar páginas web. Con CSS, se puede controlar el diseño y la apariencia visual de las páginas web. En este módulo, aprenderá los conceptos básicos de CSS, desde la sintaxis básica hasta la creación de hojas de estilo CSS para diseñar el contenido de la página web.

En este módulo te enseñaremos cómo usar CSS para controlar el estilo básico de las páginas web, como el color de fondo, el tamaño de fuente y sus posiciones. Además, aprenderás a usar los selectores de CSS para diseñar elementos de página específicos, como encabezados, párrafos y listas.

¡Por último, aprenderás a usar propiedades CSS más avanzadas para crear diseños complejos y sofisticados, como diseños en columnas o fila, diseños flexibles con la combinación de ambas y más!

Para empezar 🦉

CSS (Cascading Style Sheets) es un lenguaje web para aplicar formato visual (color, tamaño, separación y ubicación) al HTML. CSS puede hacer un texto más grande, negrita o itálica, pero no reemplaza los `strong`, `em` y `h1`. Su objetivo es separar la semántica y estructura (el HTML) del formato con que se pretende mostrar. Con CSS podés cambiar por completo el aspecto de cualquier etiqueta HTML, en esta lección aprenderemos los conceptos básicos para poder realizarlo.

¿Qué es CSS?

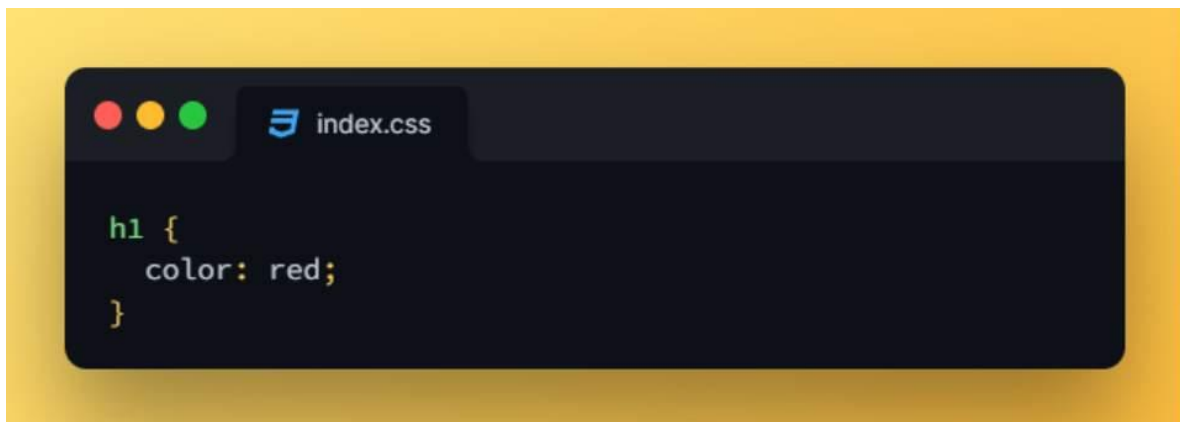
CSS (Cascading Style Sheets) es un lenguaje web para aplicar formato visual (color, tamaño, separación y ubicación) al HTML. CSS puede hacer un texto más grande, negrita o itálica, pero no reemplaza los `strong`, `em` y `h1`. Su objetivo es separar la semántica y estructura (el HTML) del formato con que se pretende mostrar. Con CSS puedes cambiar por completo el aspecto de cualquier etiqueta HTML.

En esta lección aprenderemos los **conceptos básicos** para poder realizarlo.

Sintaxis CSS

El CSS al igual que el HTML tiene una sintaxis a la hora de crear las clases y sus atributos.

```
selector {  
    propiedad1: valor;  
    propiedad2: valor;  
}
```

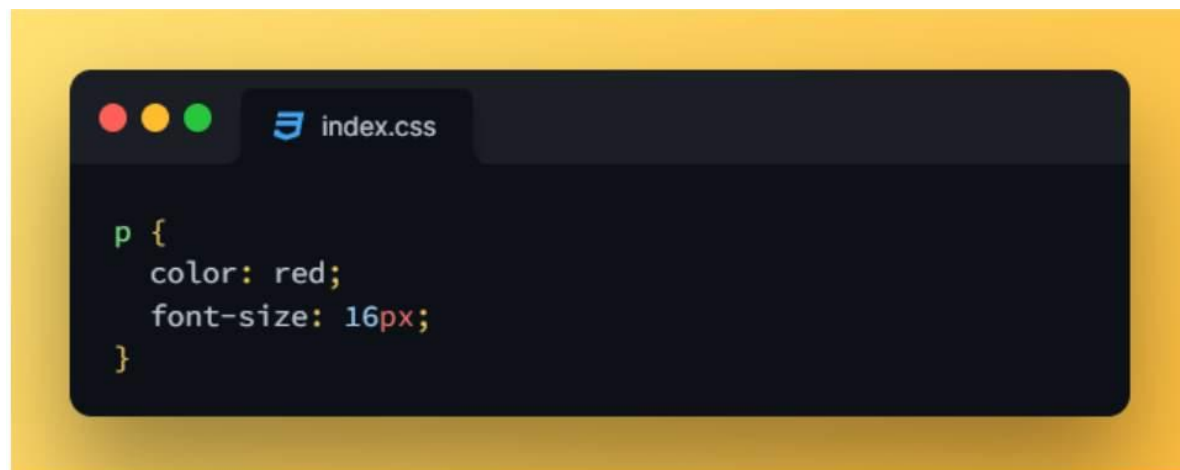
A screenshot of a code editor window titled 'index.css'. The editor has a dark background and shows a CSS rule for the 'h1' selector. The rule is: 'h1 { color: red; }'. The text is color-coded: 'h1' is green, '{', 'color:', and '}' are yellow, and 'red' is red. The editor is set against a yellow gradient background.

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

Reglas Sintácticas

Regla 1

Cada declaración CSS está formada por un juego de pares propiedad: valor; no es con un igual '=' (como pasa con los atributos HTML), sino con dos puntos ':'. Si un elemento se le aplica más de una propiedad se deben separar con punto y coma ';'.

A screenshot of a code editor window titled 'index.css'. The editor has a dark background and shows a CSS rule for the 'p' selector. The rule is: 'p { color: red; font-size: 16px; }'. The text is color-coded: 'p' is green, '{', 'color:', 'font-size:', and '}' are yellow, 'red' is red, and '16px' is blue. The editor is set against a yellow gradient background.

```
p {  
  color: red;  
  font-size: 16px;  
}
```

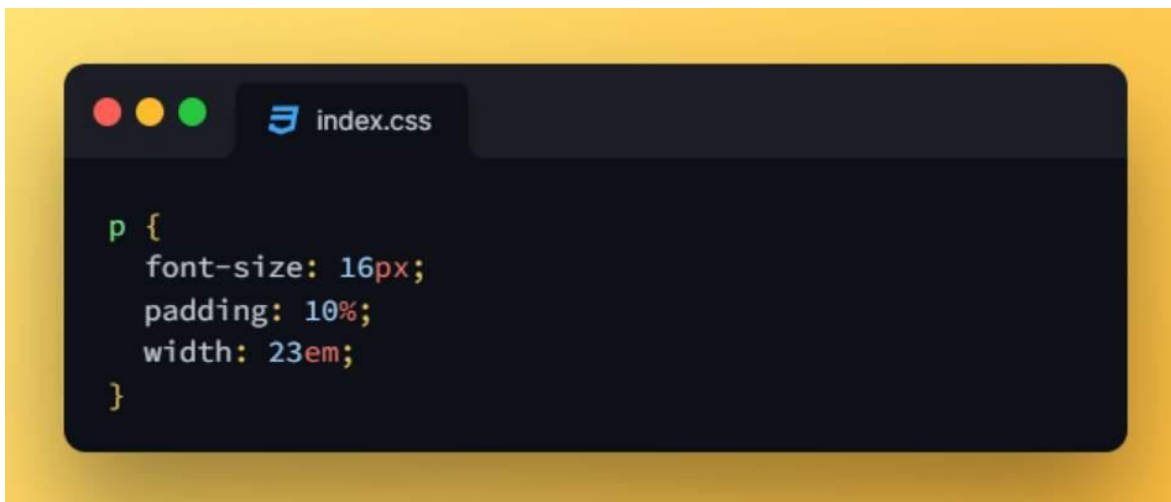
Regla 2

Tampoco se ve afectado por el espacio en blanco. Las propiedades se pueden escribir de corrido o una debajo de la otra. Los comentarios se hacen como en Javascript (utilizando `/* ... */`). Lo que esté comentado, será ignorado por CSS.



Regla 3

Siempre que la propiedad representa un número, el valor debe indicar en qué unidad se expresa. Entre el número y la unidad no pueden existir espacios.



Regla 4

Siempre que la propiedad representa un color, el valor se puede expresar de tres maneras distintas: Por nombre del color (en inglés). Por hexadecimal (numeral + 6 caracteres). Por rgb (rojo, verde, azul), tres números de 0 a 255, separados por coma.

Los componentes RGB de un color también se pueden indicar mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia es que en este caso el valor de los componentes RGB puede tomar valores entre 0% y 100%. Por tanto, para transformar un valor RGB decimal en un valor RGB porcentual, es preciso realizar una regla de tres considerando que 0 es igual a 0% y 255 es igual a 100%.

A screenshot of a code editor window titled 'index.css'. The editor has a dark background and shows the following CSS code for a paragraph element:

```
p {  
  color: red;  
  background-color: #FF0033;  
  border-color: rgb(255, 0, 0);  
}
```

Regla 5

Si se necesita aplicar el mismo formato CSS a más de un elemento diferente, no hace falta escribir dos veces todas las propiedades. Si se escribe más de un elemento, separado por comas, aplica el mismo formato a todos.

A screenshot of a code editor window titled 'index.css'. The editor has a dark background and shows the following CSS code that applies the same styles to multiple elements:

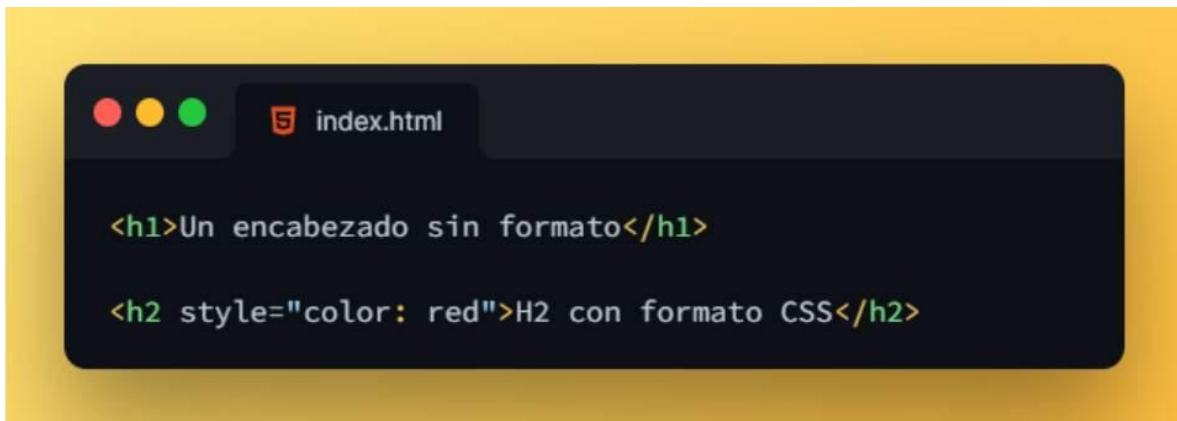
```
#elemento1, .grupo1, p {  
  color: red;  
  padding: 10%;  
  border: solid 2em #33FF44;  
}
```

Insertar CSS en HTML

Realmente existen tres formas distintas de insertar CSS en una página construida en HTML, y estas son:

Forma interna 1

Todas las etiquetas HTML tienen el atributo `style=""` y entre comillas se escribirán las reglas CSS para formatear únicamente ese elemento. Es muy poco recomendable, se suele usar para “parches” específicos, o pruebas. Se hace difícil mantenerlo a largo plazo, y nos puede traer conflictos de estilos más adelante.

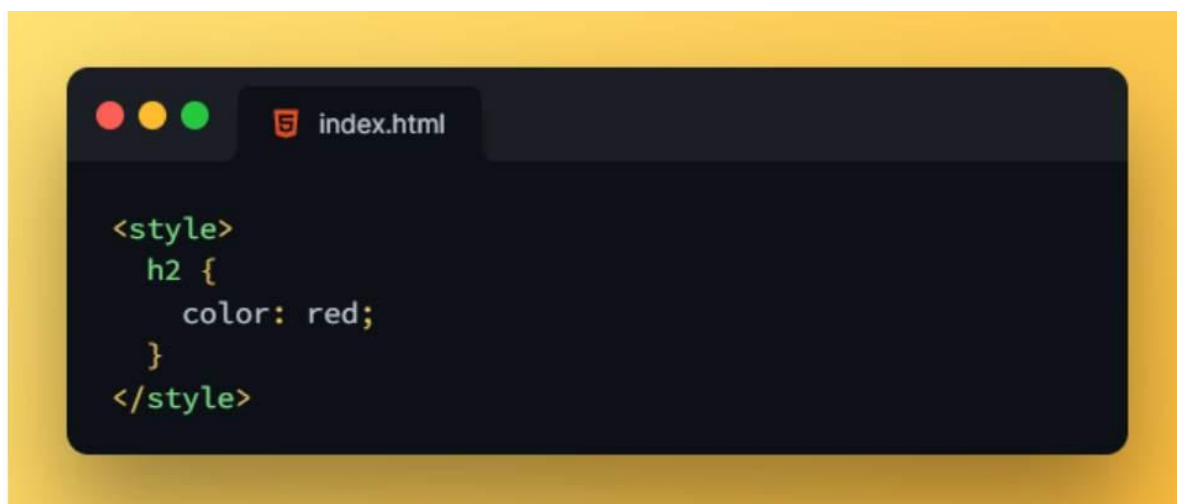
A screenshot of a code editor window titled 'index.html' with a dark theme. The editor contains two lines of HTML code. The first line is `<h1>Un encabezado sin formato</h1>`. The second line is `<h2 style="color: red">H2 con formato CSS</h2>`.

```
<h1>Un encabezado sin formato</h1>

<h2 style="color: red">H2 con formato CSS</h2>
```

Forma interna 2

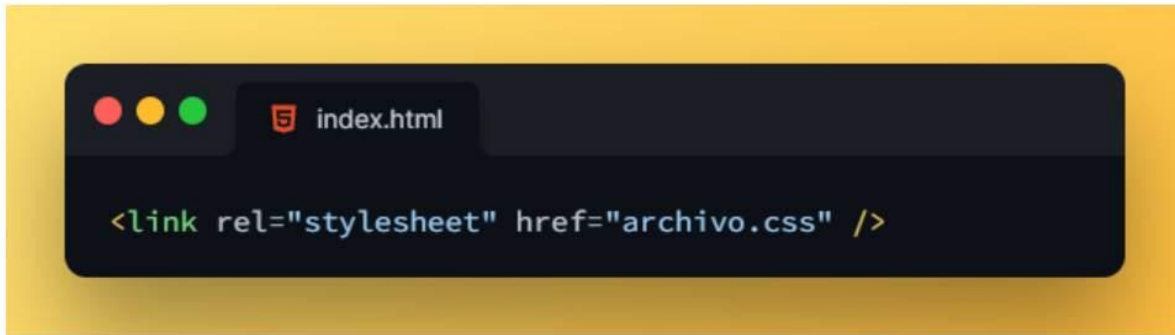
Existe una etiqueta `<style></style>` que va en el `<head>` y contendrá las reglas CSS para formatear únicamente el archivo HTML donde se haya insertado. Si bien es recomendable que esté dentro de la etiqueta `<head>`, puede estar en `<body>`, pero sería más desprolijo. Tampoco se recomienda utilizar esta forma, por las mismas razones que la anterior.

A screenshot of a code editor window titled 'index.html' with a dark theme. The editor contains a single block of CSS code enclosed in `<style>` and `</style>` tags. The CSS rule targets the `h2` selector and sets the `color` to `red`.

```
<style>
  h2 {
    color: red;
  }
</style>
```

Forna externa

Existe una etiqueta `<link/>` que va en el `<head>` y se usa para cargar un archivo externo –con extensión `.css`– que permite formatear múltiples archivos HTML. El `<link/>` no funciona si no tiene el atributo `“rel”`. Debe tener el valor `stylesheet` (hoja de estilos).



Tipos de selectores CSS

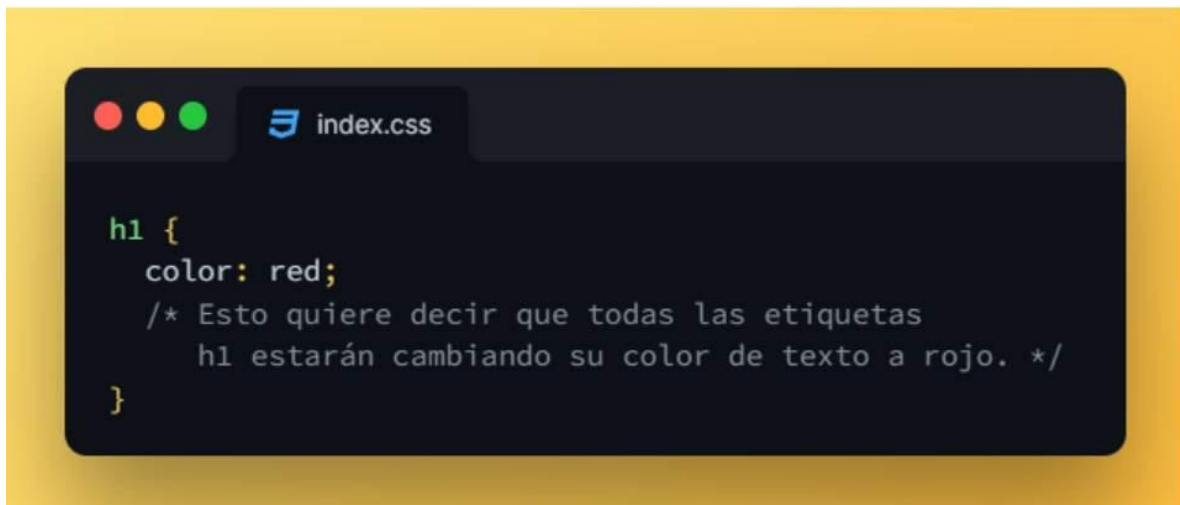
En CSS, existen tres tipos de selectores:

- 'etiqueta',
- 'clase'
- 'identificación'.

Entonces, teniendo este código HTML, podemos decir que:

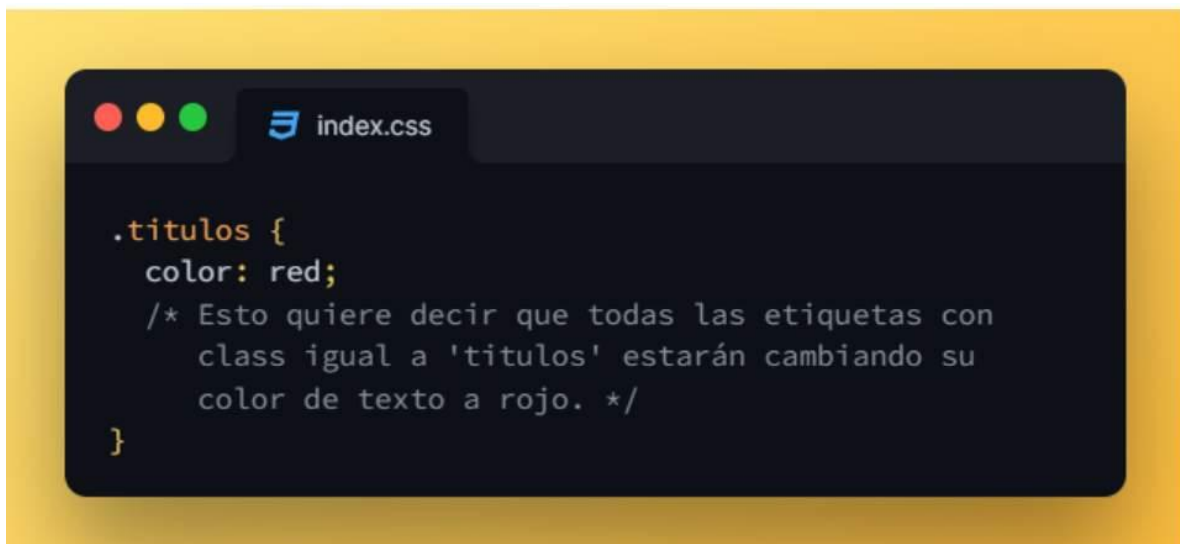


Los selectores de tipo 'tag', hacen referencia al uso del nombre de una etiqueta HTML como selector. En este caso, todo el código CSS asignado a un selector de tipo tag, se aplicará a todas las etiquetas HTML con el mismo nombre.

A screenshot of a code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) and a tab labeled 'index.css'. The code is as follows:

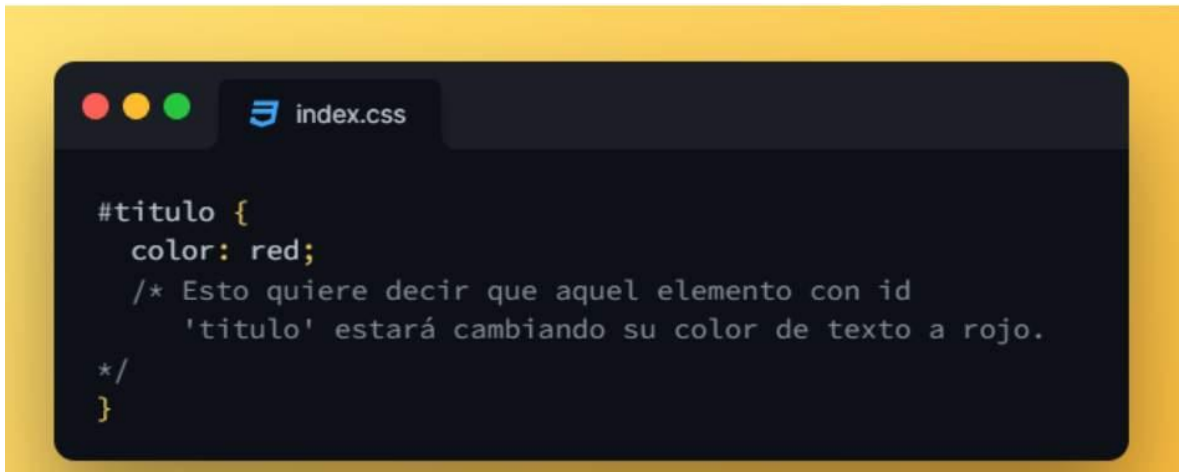
```
h1 {  
  color: red;  
  /* Esto quiere decir que todas las etiquetas  
    h1 estarán cambiando su color de texto a rojo. */  
}
```

Si utilizamos el atributo 'class' de los elementos HTML, podemos generar grupos de elementos en los que se apliquen las mismas reglas CSS. Puedes usar los nombres que quieras, siempre y cuando empiecen con letras, y pongas un “.” adelante para llamarlos en el CSS. Lo recomendable es poner un nombre que haga referencia a los estilos que tendrá.

A screenshot of a code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) and a tab labeled 'index.css'. The code is as follows:

```
.titulos {  
  color: red;  
  /* Esto quiere decir que todas las etiquetas con  
    class igual a 'titulos' estarán cambiando su  
    color de texto a rojo. */  
}
```

Si utilizamos el atributo 'id' de los elementos HTML, podemos darle un nombre identificador único a cada uno de ellos. Puedes usar los nombres que quieras, siempre y cuando empieces con letras, y pongas un “#” adelante para llamarlos en el CSS. Al ser único, no podemos repetir su nombre en ningún otro elemento, sirve para darle estilos a un elemento en específico. Si repetimos un id, solo se utilizarán los estilos al primer elemento que tenga su nombre.



Prioridades y precedencia de selectores

En cuanto a la precedencia de declaraciones, tenemos que tener en cuenta que, cuando varias reglas CSS apuntan al mismo elemento HTML:

- Si son propiedades distintas, se suman (se combinan).
- Si tienen alguna propiedad repetida, sólo queda una (la última).

Ejemplo inicial	Resultado final
<pre>h1 { color: red; } ... h1 { font-size: 2em; }</pre>	<pre>h1 { color: red; font-size: 2em; }</pre>
<pre>h1 { color: red; } ... h1 { color: green; }</pre>	<pre>h1 { color: green; }</pre>

En cuanto a la prioridad de los selectores, tenemos que tener en cuenta que:

- Los estilos aplicados con un ID, pisan a cualquier otro estilo.
- Los estilos aplicados con Class, sobrescriben los estilos aplicados con selectores de etiquetas, pero no a los selectores de ID.
- Por último, los selectores de etiquetas tienen la menor precedencia.

Si utiliza estilos inline (aplicados con el atributo style en los elementos HTML), sobrescribirán cualquier estilo de las páginas externas de CSS.

Se podría decir que los estilos en línea son los que tienen una mayor especificidad, por lo tanto no es recomendable utilizarlos en tu página.

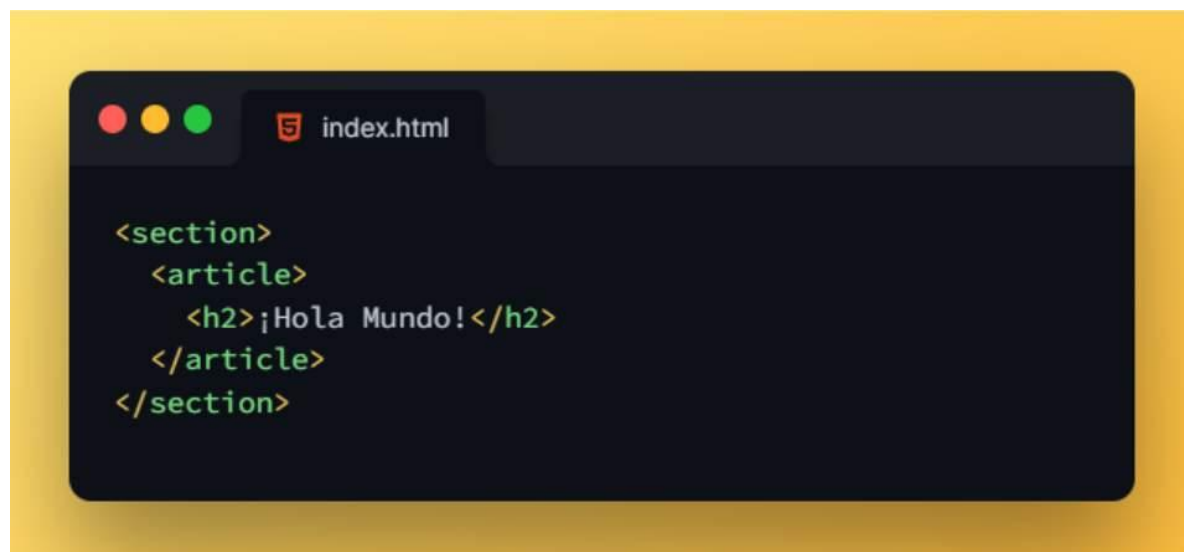
Por lo tanto, podemos decir que nuestro gráfico de precedencias se ve así:

Estilos inline > ID > Class > Etiquetas

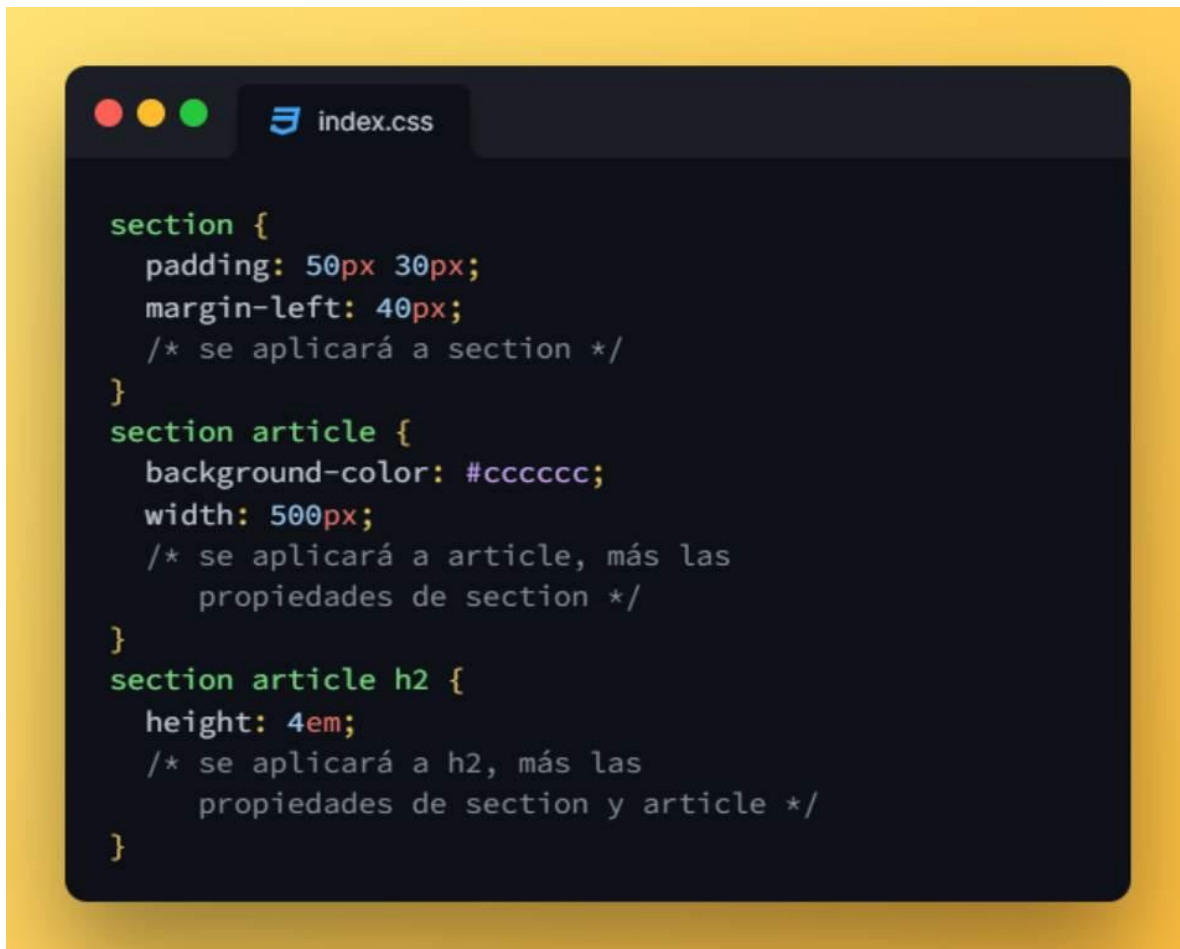
Herencia y Cascada

Cuando tienes una etiqueta “dentro” de otra, es decir, se encuentran anidadas, lo que haces es aplicar el concepto de padres e hijos, o herencia de estilos. Podrás heredar de un elemento padre el tamaño de letra y color de la misma, a menos que el elemento hijo tenga otros estilos aplicados.

Un elemento padre puede tener muchos hijos, y todos ellos heredan sus características, pudiendo tener también características particulares.



```
<section>
  <article>
    <h2>¡Hola Mundo!</h2>
  </article>
</section>
```

A screenshot of a code editor window with a dark background and yellow border. The window has a title bar with three colored circles (red, yellow, green) and a tab labeled 'index.css'. The code is written in a light green monospace font. It defines three CSS rules: 'section' with padding and margin, 'section article' with background color and width, and 'section article h2' with height. Each rule includes a comment in Spanish explaining its application.

```

section {
  padding: 50px 30px;
  margin-left: 40px;
  /* se aplicará a section */
}
section article {
  background-color: #cccccc;
  width: 500px;
  /* se aplicará a article, más las
    propiedades de section */
}
section article h2 {
  height: 4em;
  /* se aplicará a h2, más las
    propiedades de section y article */
}

```

El concepto de 'cascada' define una ruta u orden de anidamiento que deben cumplir los elementos dentro del HTML. Se trata de una lista de elementos (ya sea por etiqueta, ID o clase) separados por espacios. Si algún elemento cumple esta "ruta" le aplica el formato CSS al último elemento de la lista (el que está justo antes de la apertura de llaves). Se usa para no plagar el HTML de atributos CLASS e ID.

Por ejemplo:

Tenemos dos listas distintas (un y un) y necesitamos que los de cada lista sean de distintos colores. Podemos aplicarle un class a cada list-item (pero esto nos obliga a copiar y pegar el class tantas veces como items tenga cada lista) o hacer una cascada discriminando los dentro de cada lista.

Aplicación Clase

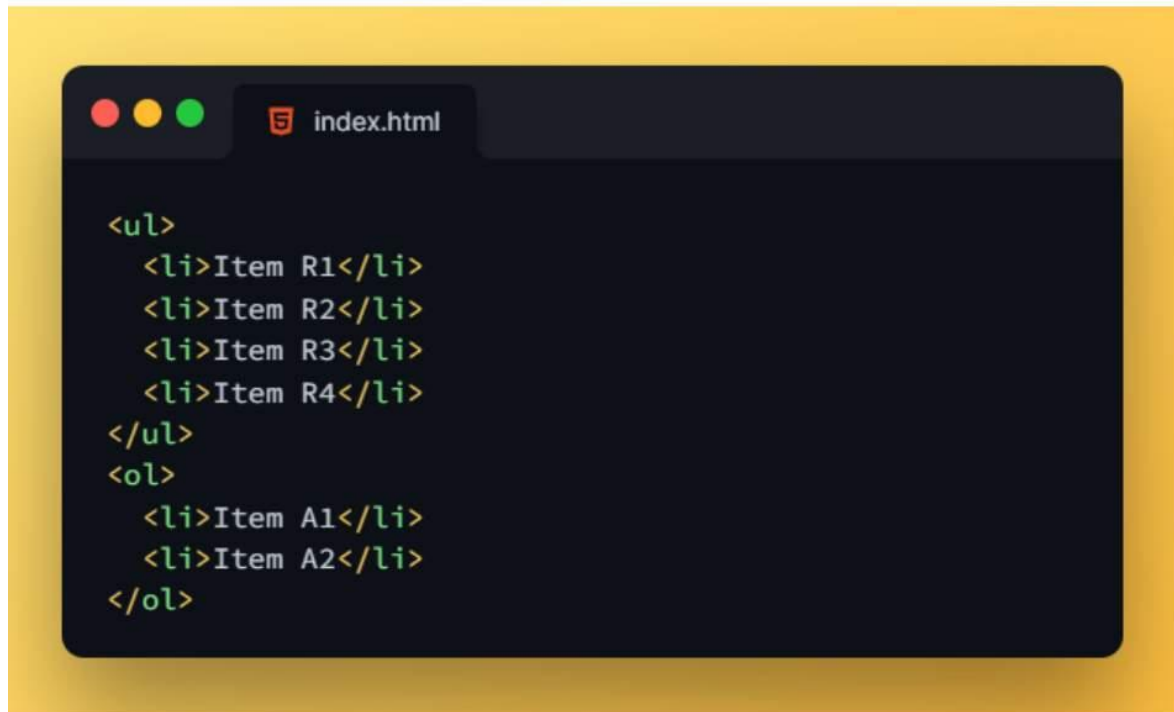
index.html

```
<ul>
  <li class="rojo">Item R1</li>
  <li class="rojo">Item R2</li>
  <li class="rojo">Item R3</li>
  <li class="rojo">Item R4</li>
</ul>
<ol>
  <li class="azul">Item A1</li>
  <li class="azul">Item A2</li>
</ol>
```

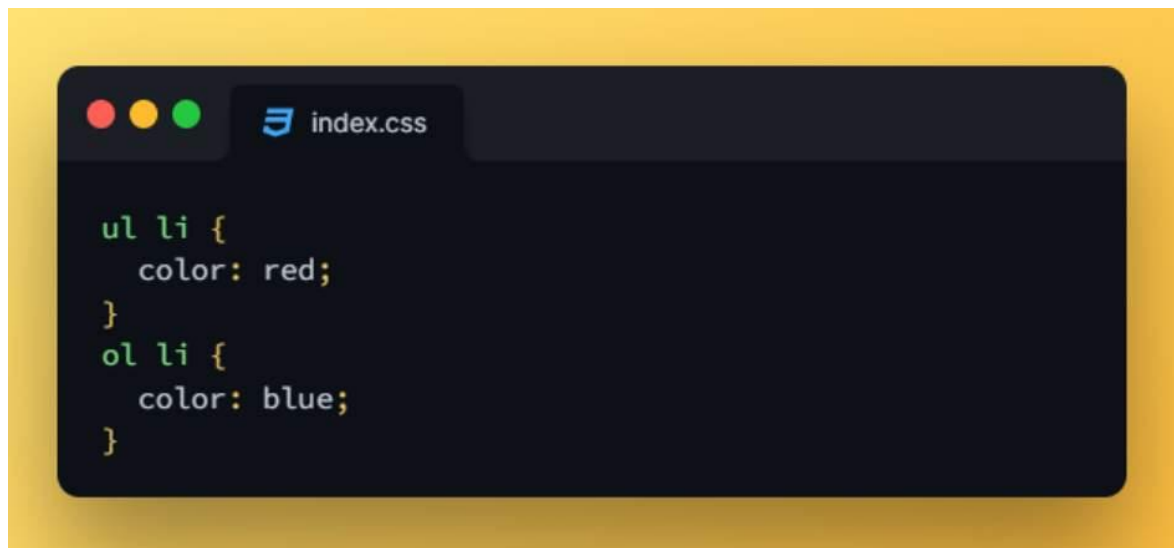
index.css

```
.rojo {
  color: red;
}
.azul {
  color: blue;
}
```

Aplicación Cascada:

A screenshot of a code editor window with a dark theme. The title bar shows three colored window control buttons (red, yellow, green) and a tab labeled 'index.html' with a small icon. The code is written in a light green monospace font on a dark background. It defines an unordered list with four items (R1, R2, R3, R4) and an ordered list with two items (A1, A2).

```
<ul>
  <li>Item R1</li>
  <li>Item R2</li>
  <li>Item R3</li>
  <li>Item R4</li>
</ul>
<ol>
  <li>Item A1</li>
  <li>Item A2</li>
</ol>
```

A screenshot of a code editor window with a dark theme. The title bar shows three colored window control buttons (red, yellow, green) and a tab labeled 'index.css' with a small icon. The code is written in a light green monospace font on a dark background. It defines two CSS rules: one for 'ul li' with 'color: red;' and one for 'ol li' with 'color: blue;'.

```
ul li {
  color: red;
}
ol li {
  color: blue;
}
```

¿Que son las propiedades del color?

Dentro de las propiedades de textos y listas, nos encontramos con una lista bastante extensa, en esta oportunidad veremos las más utilizadas:

La **propiedad color** selecciona el valor de color del texto de un elemento y sus decoraciones.

Debes tener en cuenta que existen distintos valores que pueden tomar esta propiedad, pero nos centraremos en tres:

- Nombre del color (por ejemplo: rojo).
- Hexadecimal (por ejemplo: #FFFFFF).
- RGB (por ejemplo: 50, 212, 227). Si agregas un valor más, puedes manejar su opacidad. Cada color permite hasta 256 valores.



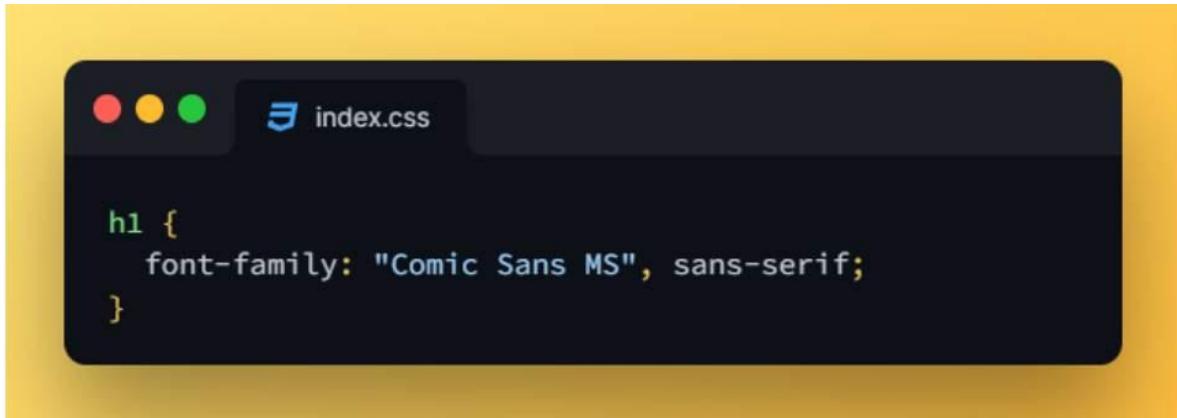
Familia tipográfica Propiedad

La propiedad font-family define una lista de fuentes o familias de fuentes, con un orden de prioridad, para utilizar en un elemento seleccionado.

Su posible valor puede ser cualquier nombre de una tipografía disponible en su computador.

Cada sistema operativo y navegador interpretan de forma distinta las fuentes predeterminadas.

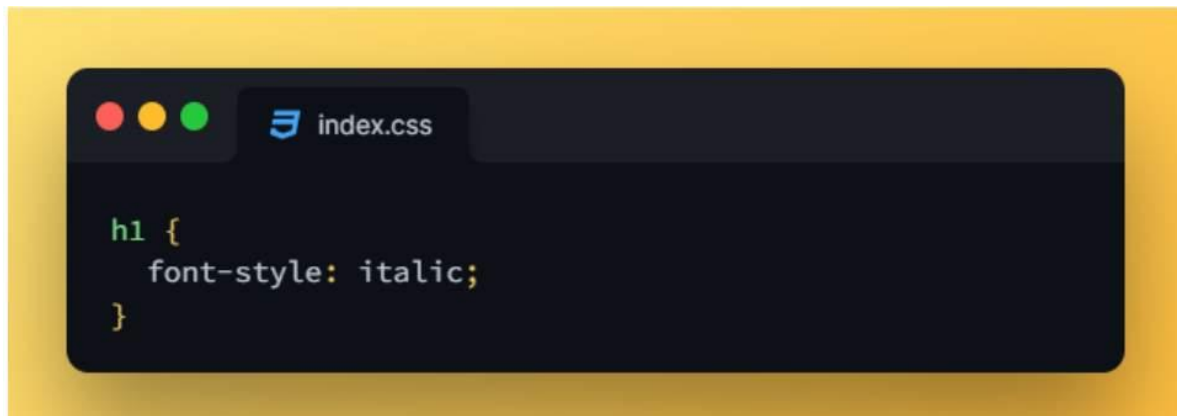
- **Serif** : «Times New Roman» en Windows, y «Times» en Macintosh (diferente a la de Windows).
- **Sans serif**: «Arial» en Windows, y «Helvetica» en Macintosh.
- **Monospace** : «Courier New» en Windows, «Courier» en Macintosh, y por lo general «VeraSans» o «DejaVuSans» en Linux.



Propiedad estilo de fuente

La propiedad **font-style** permite definir el aspecto de una familia tipográfica, en referencia a la inclinación de la letra.

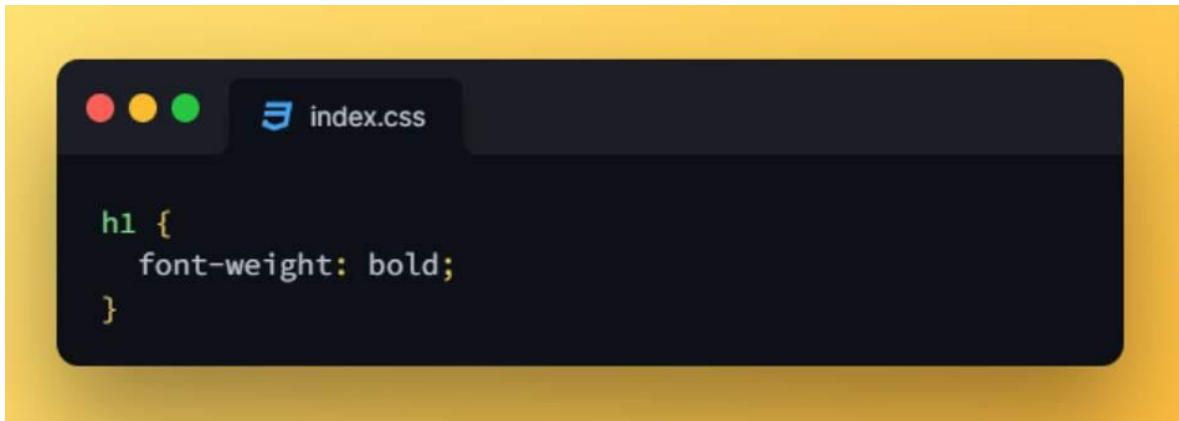
Sus posibles valores más utilizados: normal, *cursiva* .



Propiedad font-weight

La propiedad font-weight especifica el peso o grosor de la letra. Algunos tipos de letra sólo están disponibles en normal y bold.

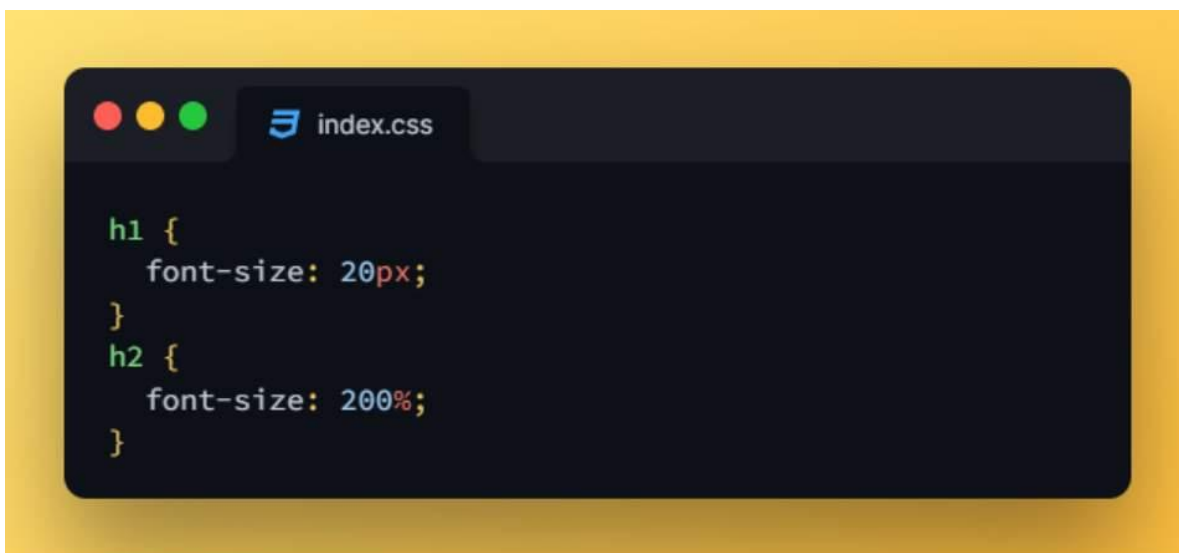
Sus posibles valores más utilizados: normal, negrita, (números).



Propiedad font-size

La propiedad font-size especifica el tamaño de la letra. Este tamaño puede, a su vez, alterar el aspecto de alguna otra cosa, ya que se usa para calcular la longitud de las unidades em y ex.

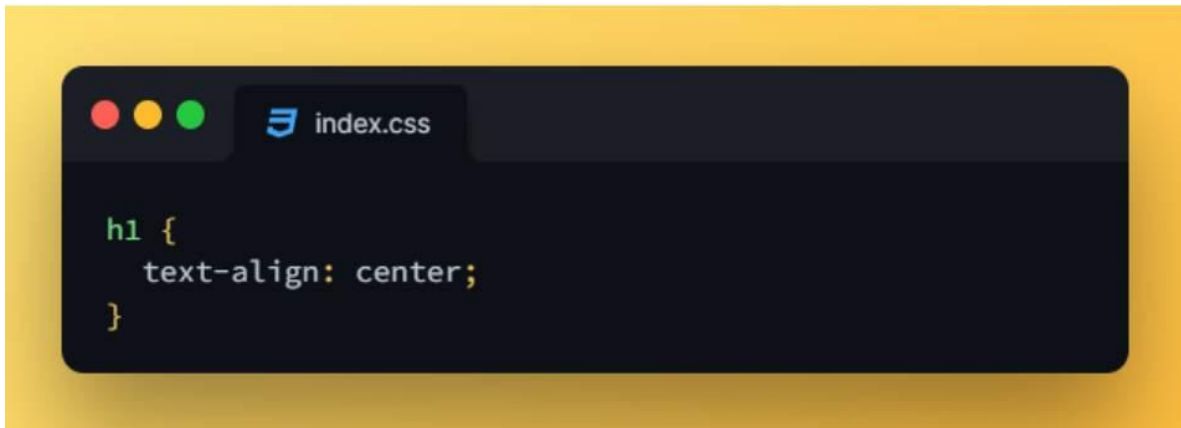
Sus posibles valores más utilizados: (números que incluyen unidad de medida).



Propiedad text-align

La propiedad text-align establece la alineación horizontal del contenido dentro de un elemento de bloque o cuadro de celda de una tabla. Esto significa que funciona como alineación vertical pero en dirección horizontal.

Sus posibles valores más utilizados: izquierda, derecha, centro, justificar.

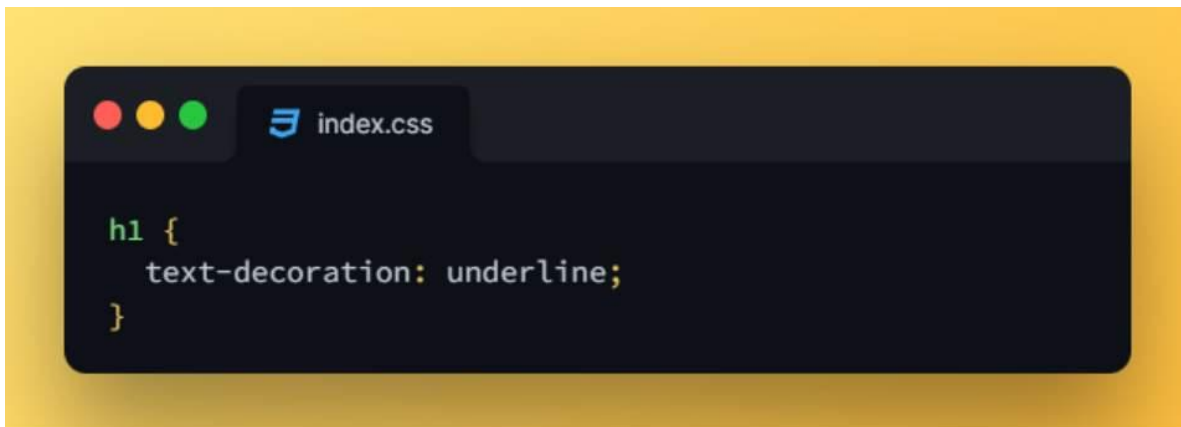


Propiedad text-decoration

La propiedad text-decoration se usa para establecer el formato de texto normal subrayado y suprarayado, tachado, entre otros.

Sus posibles valores más utilizados son: underline, overline, line-through.

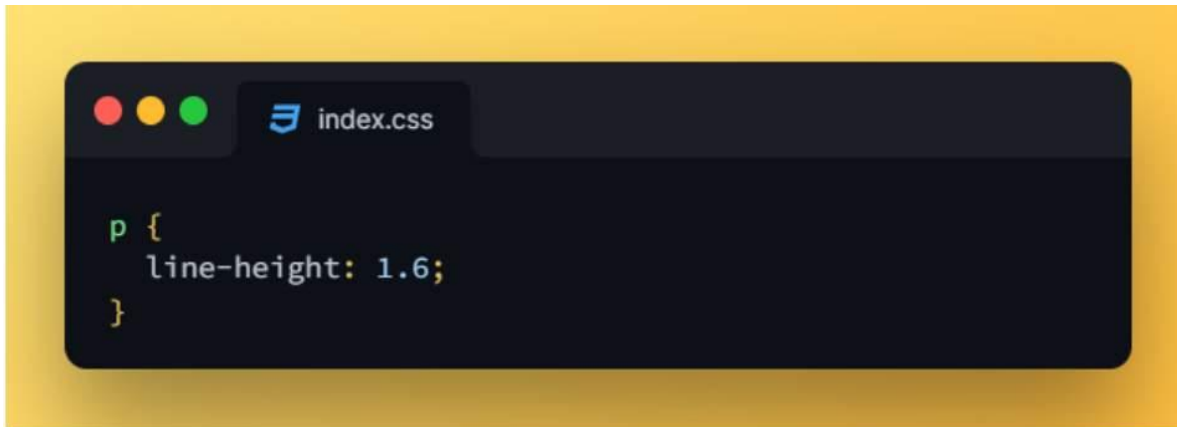
En caso de querer quitarle la decoración a algún texto, se debe asignar el valor none.



Propiedad line-height

La propiedad line-height establece el tamaño del interlineado entre líneas de texto en bloque.

Sus posibles valores más utilizados: ninguno, (números que incluyen unidad de medida).

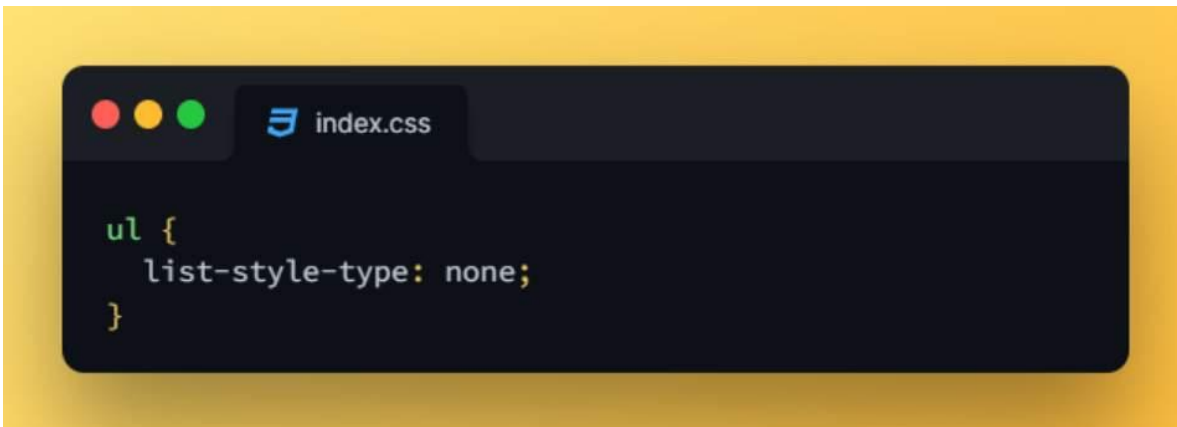


Propiedad list-style-type

La propiedad list-style-type establece el tipo de ícono enumerador de un elemento de lista.

Sus posibles valores más utilizados son: disco, círculo, cuadrado, decimal, romano superior.

En caso de querer quitarle las viñetas a la lista, se debe asignar el valor none.



Unidades de medida

Las medidas en CSS se utilizan, entre otras, para definir la altura, la anchura y los márgenes de los elementos y para establecer el tamaño de la letra del texto. Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).

CSS divide las unidades de medida en dos grupos: absolutas y relativas:

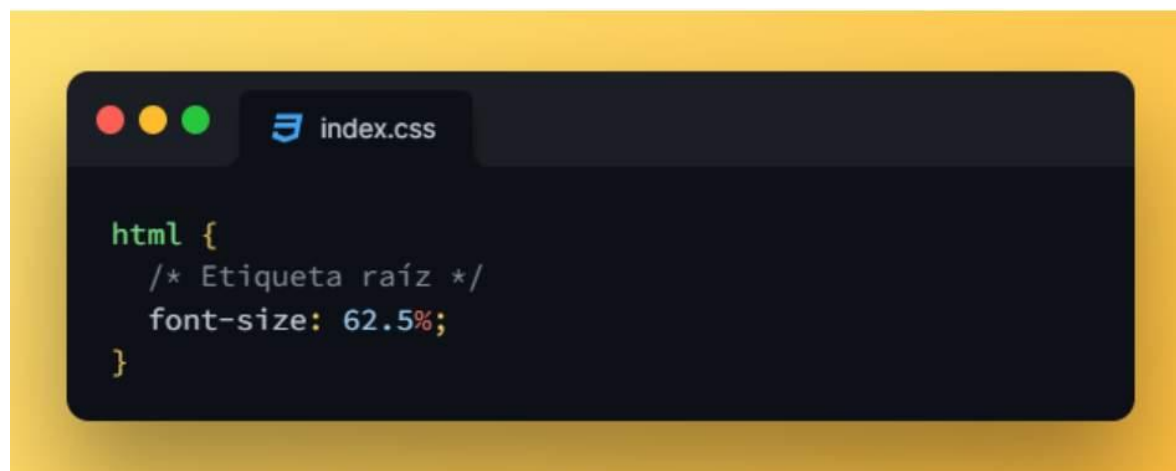
- Las medidas relativas definen su valor en relación con otra medida, por lo que para obtener su valor real, se debe realizar alguna operación con el valor indicado.
- Las unidades absolutas fortalecen de forma completa el valor de una medida, por lo que su valor real es directamente el valor indicado.

Si el valor es 0, la unidad de medida es opcional. Si el valor es distinto a 0 y no se indica ninguna unidad, la medida se ignora completamente. Algunas propiedades permiten indicar medidas negativas, aunque habitualmente sus valores son positivos. Si el valor decimal de una medida es inferior a 1, se puede omitir el 0 de la izquierda (0.5em es equivalente a .5em).

Dentro de las unidades de medida más utilizadas, están:

Absolutas	Relativas
Px (píxeles): es la unidad de medida que usan las pantallas.	Rem: relativa a la configuración de tamaño del elemento raíz (etiqueta html).
	Porcentaje: tomando en cuenta que 16px es 100%.
	Viewport: se utilizan para diseños responsive.

Ahora veamos qué medida es más conveniente para los textos. Si nosotros configuramos el tamaño de fuente del elemento html en 62.5%, hacemos que en vez de que 16px sea el 100% del valor a tomar en cuenta para calcular las unidades relativas, se use 10px.



Tipografías locales y web

Vimos que usando “font-family”, es posible agregar algunas fuentes limitadas que poseemos en nuestras computadoras, pero podemos usar muchísimas opciones de fuentes si las descargamos y las agregamos a nuestro directorio raíz.



El valor de la propiedad `src` debe indicar en qué parte de nuestro directorio raíz guardamos nuestra tipografía post descarga.

Ahora, si hacemos esto, es muy probable que nuestro proyecto crezca en peso en cuanto a la cantidad de archivos que lo componen, entre otros conflictos que se pueden presentar, es por eso que existe la posibilidad de utilizar bancos tipográficos online, el más conocido es Fuentes de Google.

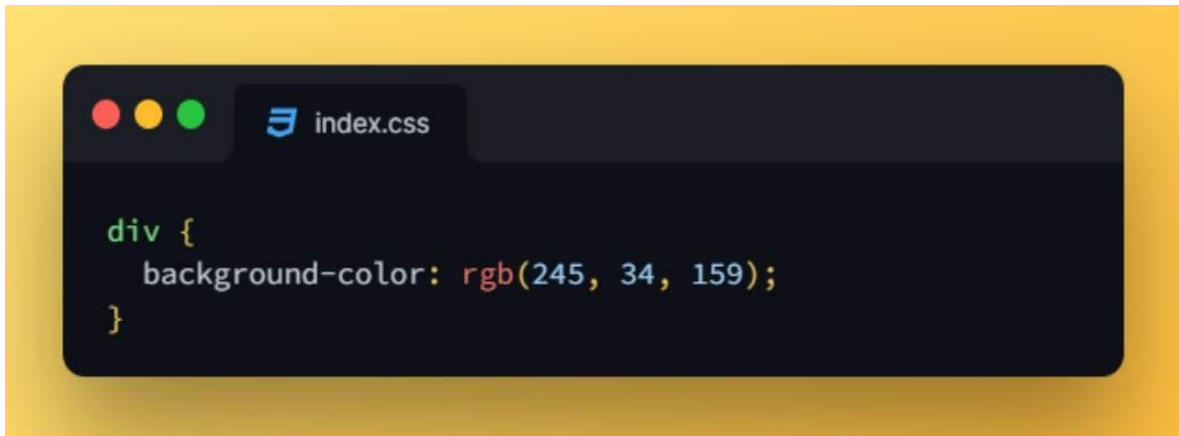
Para poder utilizar una tipografía disponible en Google Fonts, debemos seleccionarla, elegir sus variables y copiar el código que nos provee la misma plataforma. Aquí dejamos un ejemplo:



LOS ESTILOS DE IMÁGENES

Propiedad `background-color`

La forma de indicar el color de fondo de una etiqueta desde CSS es con la propiedad `background-color`. Como valor, podemos utilizar uno de los colores predeterminados, el sistema hexadecimal, el sistema RGB o la palabra clave transparente.



Propiedad background-image

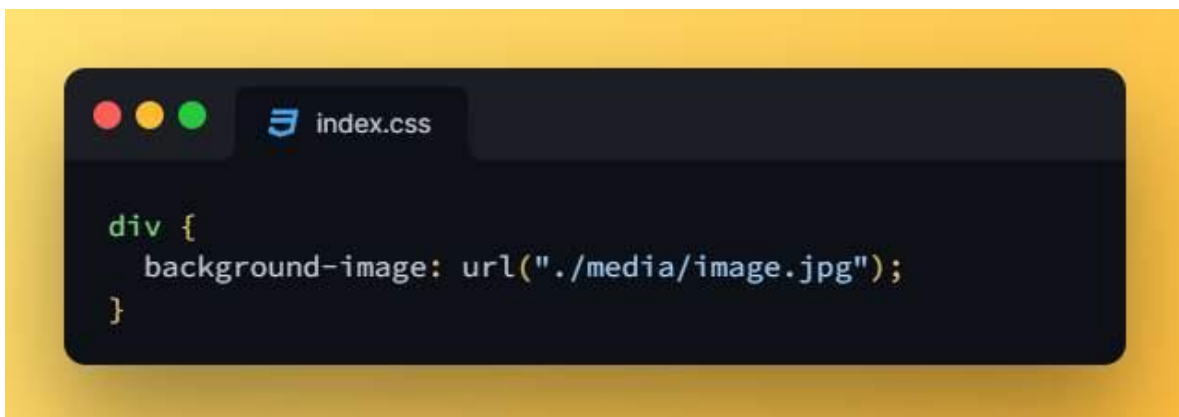
No sólo podemos aplicar color, sino que además podemos aplicar una imagen como fondo de una etiqueta. La propiedad a utilizar, en este caso, es **background-image**.

Las imágenes de fondo se indican a través de su URL, que pueden ser absolutas o relativas, y deben ser enlazadas mediante la función `url("...")`, pasada como valor de la propiedad.

Suele ser recomendable crear una carpeta de imágenes que se encuentre en el mismo directorio que los archivos CSS y que almacene todas las imágenes utilizadas en el diseño de las páginas.

CSS no permite establecer de forma simultánea un color y una imagen de fondo. De hecho, es recomendable aplicar ambas, ya que si por alguna razón, no se muestra la imagen de fondo, vamos a poder ver el color.

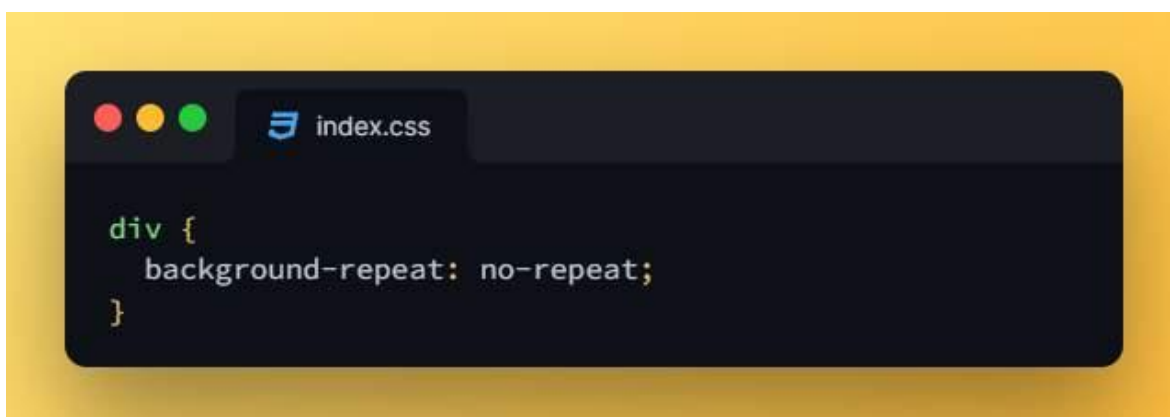
Algo importante a tener en cuenta, son los formatos de imágenes que podemos utilizar para el fondo. Los mismos son: JPG, PNG o GIF. En todos los casos, tienen que estar en modo RGB.



Propiedad background-repeat

La propiedad `background-repeat` define cómo se repiten los fondos del documento. Un fondo de imagen puede ser repetido sobre el eje horizontal, el eje vertical, ambos ejes, o no estar repetido.

Sus posibles valores son: repetir-x, repetir-y, repetir, espacio, redondo, no-repetir.



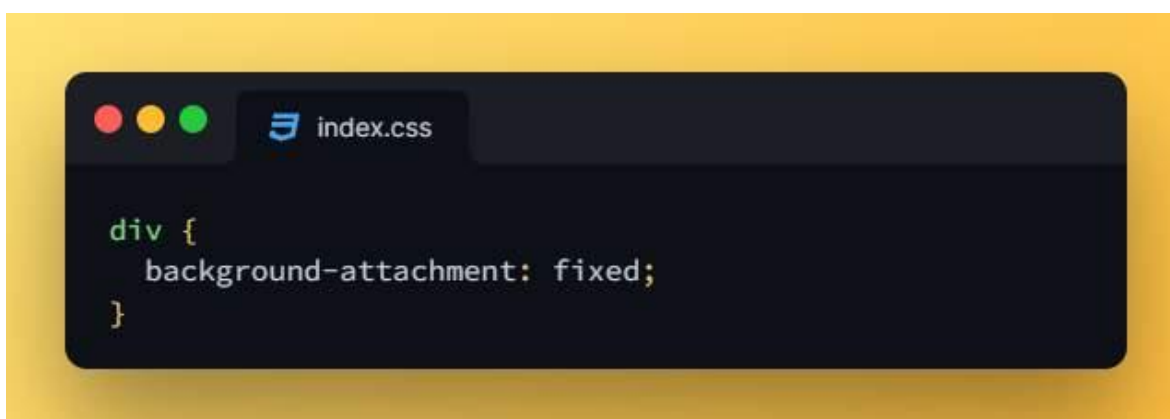
Propiedad background-position

Además de seleccionar el tipo de repetición de las imágenes de fondo, CSS permite controlar la posición de la imagen dentro del fondo del elemento mediante la propiedad background-position.

En los casos que elegimos que la imagen no se repite, lo que sucede es que esta imagen se alinea con la esquina superior izquierda de la caja. La propiedad background-position nos permite elegir la posición de dicha imagen dentro de la caja.

Sus posibles valores son: arriba, abajo, centro, izquierda y derecha, aunque estas también se pueden combinar. Por ejemplo, podemos decir que nuestra imagen de fondo está alineada en el centro superior.

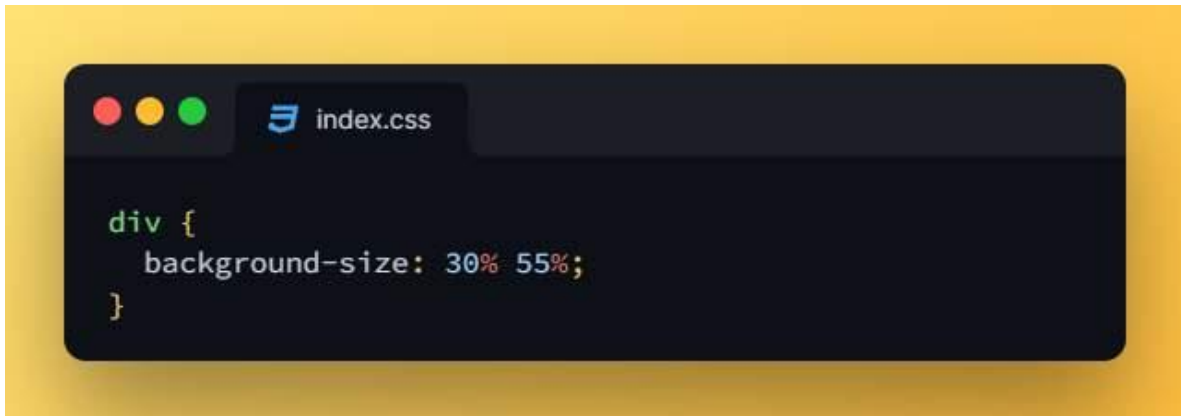
Otra opción es utilizar números, de esta manera vamos a poder indicar la coordenada exacta de la posición de la imagen con respecto a la caja en donde está ubicada como fondo.



Propiedad background-size

La propiedad background-size especifica el tamaño de las imágenes de fondo.

Sus posibles valores son: container, cover, auto, o también, 1 o 2 números. Dichos números deben incluir sus unidades de medida, y hacen referencia al ancho y alto. Si solo se incluye 1, se aplica la misma medida a ambos ejes.



desarrollo conceptual

Una pseudo-clase CSS es una palabra clave que se agrega a los selectores y que especifica un estado especial del elemento seleccionado.

Por ejemplo, :hover utilice un estilo cuando el usuario haga pase su mouse sobre el elemento especificado por el selector.

Su sintaxis es la siguiente:

```
selector:pseudo-clase {  
  propiedad1: valor;  
  propiedad2: valor;  
}
```

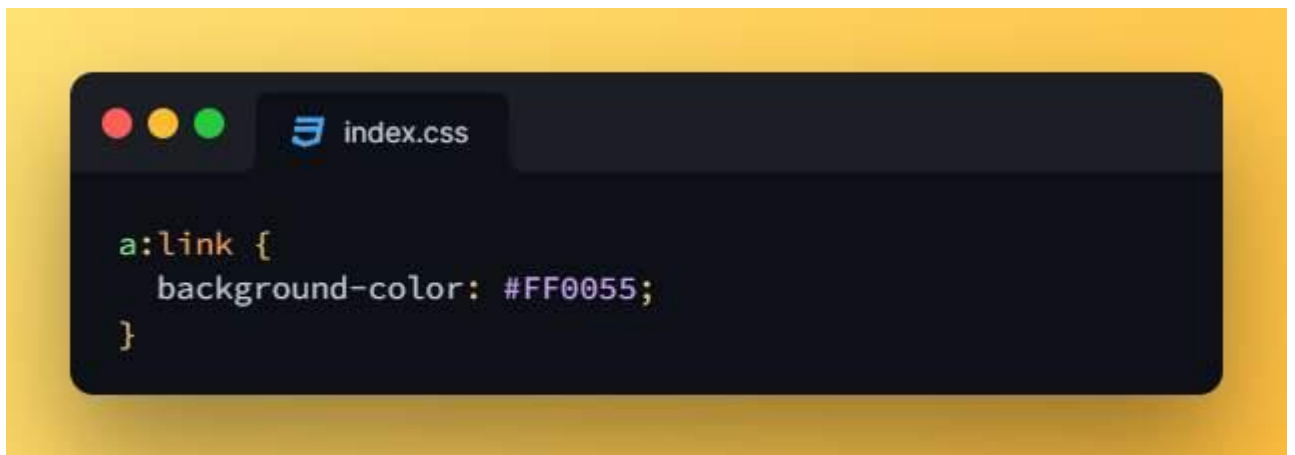
Las pseudo-clases, junto con los pseudo-elementos, permiten aplicar un estilo a un elemento no sólo en relación con el contenido del árbol de documento, sino también en relación a factores externos como:

- El historial del navegador (:visited),
- El estado de su contenido (como :checked en checkbox de forms),
- La posición del ratón (:hover).

Los estilos de las pseudoclasas link, visited, hover, y active pueden ser anulados entre ellos dependiendo del orden en el que fueron definidos. Para darle un estilo apropiado a los enlaces, coloque las reglas como lo define el orden LVHA: :link — :visited — :hover — :active.

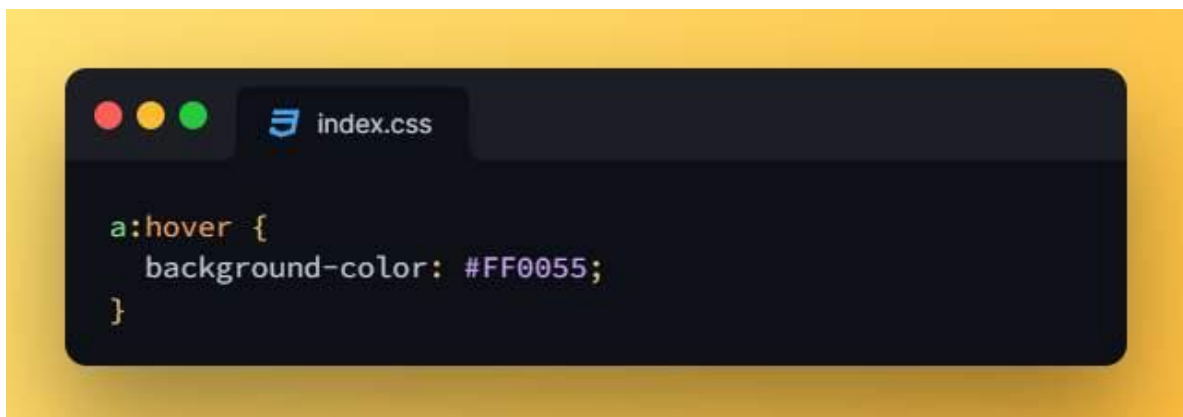
Pseudo-clase :enlace

La pseudo-clase :link representa un elemento que aún no se ha visitado. Coincide con cada elemento no visitado <a>, <area>, o <link> que tiene un atributo href.



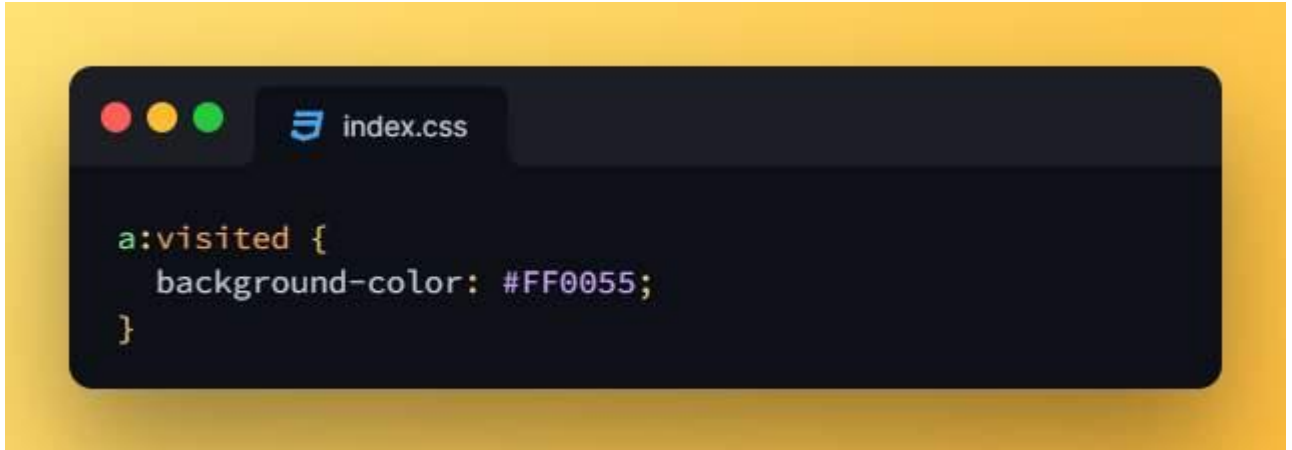
Pseudo-clase: hover

La pseudo-clase :hover coincide cuando el usuario interactúa con un elemento con un dispositivo señalador, pero no obstante no lo activa. Generalmente se activa cuando el usuario se desplaza sobre un elemento con el cursor (puntero del mouse).



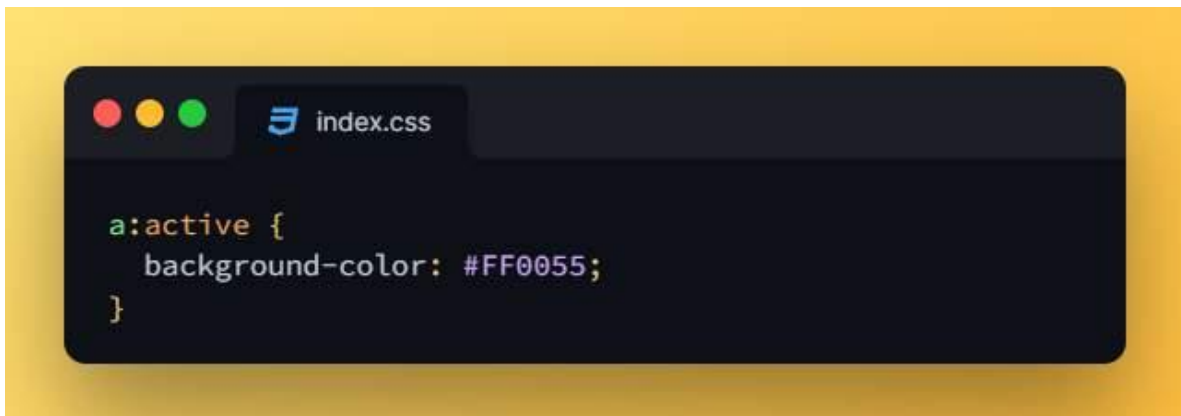
Pseudoclase: visitado

La pseudo-clase :visited representa enlaces que el usuario ya ha visitado. Por motivos de privacidad, los estilos que se pueden modificar con este selector son muy limitados.



Pseudo-clase: activo

La pseudo-clase :active representa un elemento que el usuario está activando. Cuando se usa un mouse, la "activación" generalmente comienza cuando el usuario presiona el botón primario del mouse y termina cuando se suelta.



Pseudo-clase: foco

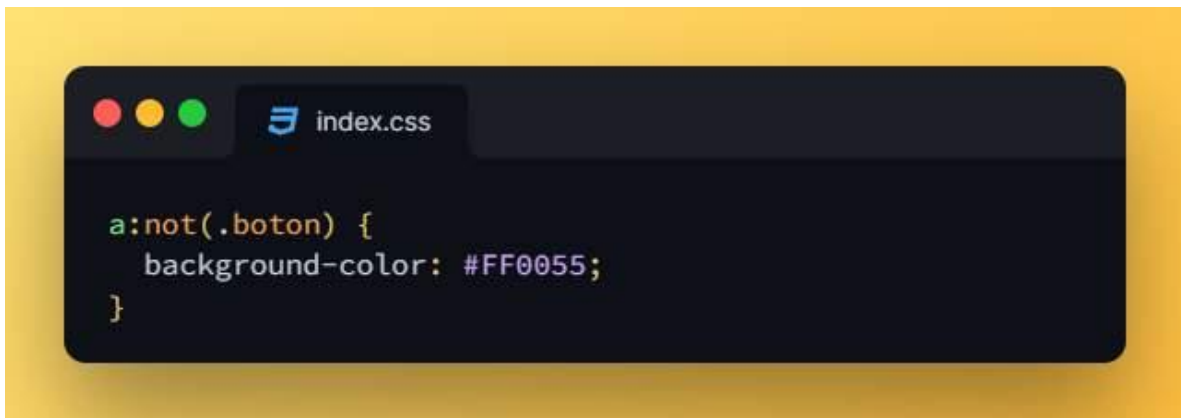
La pseudo-clase :focus representa un elemento (como una entrada de formulario) que ha recibido el foco. Generalmente se activa cuando el usuario hace clic, toca un elemento o lo selecciona con la tecla "Tab" del teclado.

Pseudo-clase :not()

La pseudo-clase :not() representa elementos que no coinciden con una lista de selectores. Es decir, aplica a todos los elementos A, excepto a los incluidos como elementos B.

Como evita que se seleccionen elementos específicos, se lo conoce como la pseudo-clase de negación.

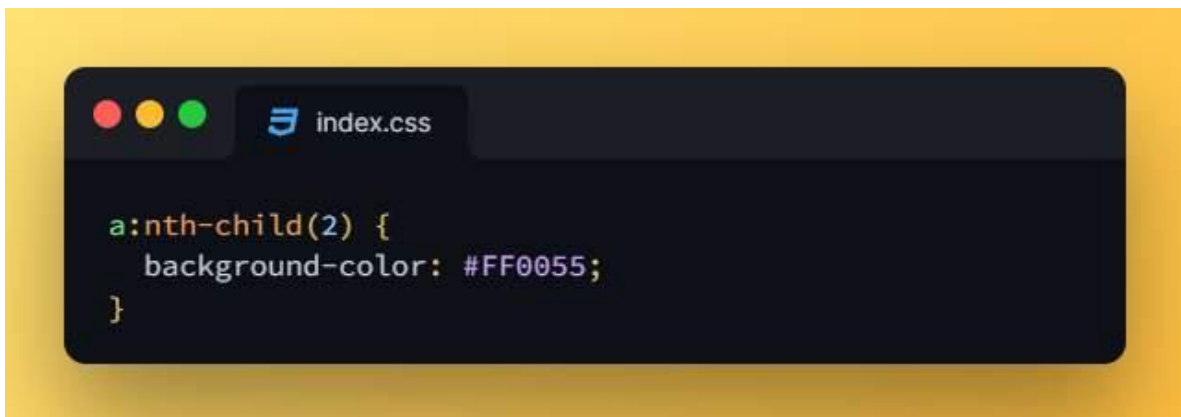
Por ejemplo, podemos aplicar unos estilos a todos los elementos “a” y evitar que éstos se apliquen a otras capas con un id o class determinado. En el siguiente ejemplo hacemos justo esto, impidiendo que se apliquen los estilos a los elementos del tipo <a> que tienen como clase “botón”.



Pseudo-clase :nth-child()

Esta fue la solución a todos los problemas de maquetación de antaño con las filas y los bloques.

Con :nth-child(N) podremos aplicar sus estilos a todos los elementos hijos cuya posición sea un número “N” respecto a un padre. Este número N no tiene que ser, no obstante, un número entero para especificar una posición fija (la posición 2, por ejemplo), ya que también permite insertar fórmulas y palabras específicas.



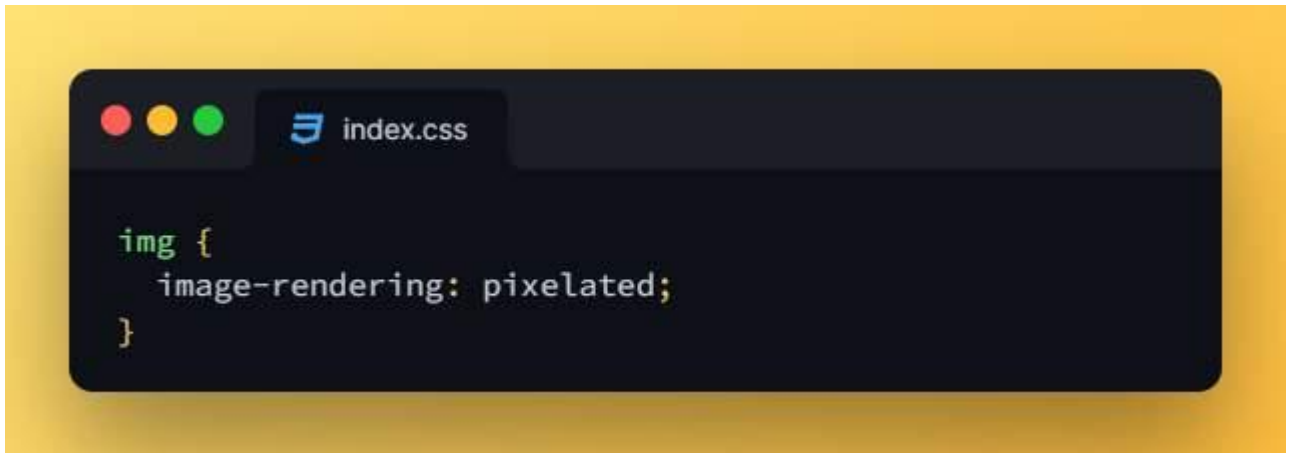
Existen infinidad de propiedades CSS para realizar multitud de acciones con imágenes. Por ejemplo, podemos incluso establecerlas como fondo, o bien podemos modificar su comportamiento, como agregándoles un borde y recortándolas. Sin embargo, también hay propiedades CSS relacionadas con imágenes no tan conocidas, las cuales, al igual que las anteriores, también pueden ser utilizadas con la etiqueta IMG de HTML.

En esta lección hablaremos de propiedades que te van a permitir, desde poder controlar la sombra de una imagen hasta establecer su nitidez, lo que nos ayuda a controlar mejor la apariencia y la posición de las imágenes agregadas con la etiqueta .

Controlar el renderizado de una imagen

Cuando se escala una imagen, el navegador suaviza la imagen para que no se vea pixelada. Pero, dependiendo de la resolución de la imagen y la pantalla, hay veces que esto no nos sirve. Puedes controlar este comportamiento del navegador con la propiedad `image-rendering`.

Sus posibles valores son: bordes nítidos y pixelados.



Estirar imágenes

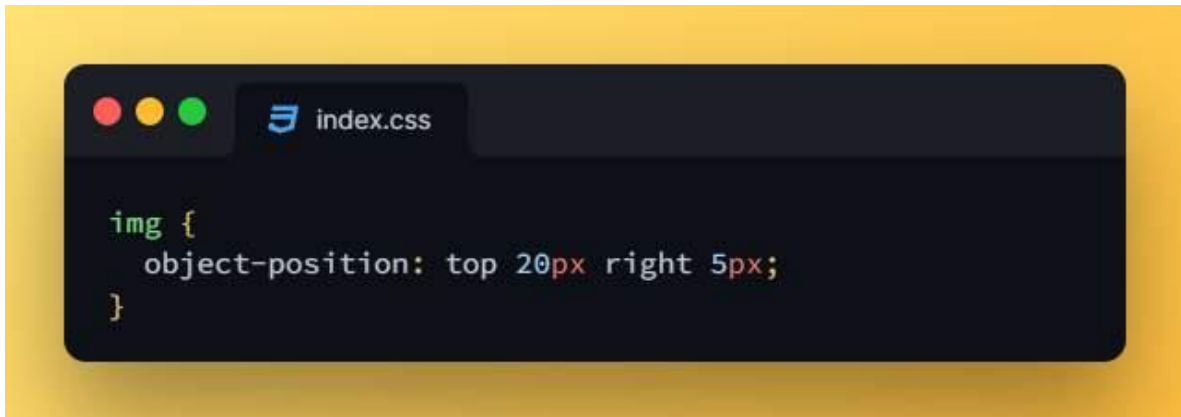
Los valores contienen, cubren y llenar nos suenan. Los utilizamos en la propiedad `background-size` para controlar la imagen de fondo se acopla al elemento al que pertenece. La propiedad `object-fit` es bastante similar, ya que también determina cómo una imagen se ajusta dentro de su contenedor.

Sus posibles valores son: contener, cubrir, llenar, reducir.

Mover imágenes

La propiedad `background-position` es una propiedad complementaria de la propiedad `background-size`. En el caso de `object-fit` también cuenta con su propiedad complementaria, `object-position`.

La propiedad `object-fit` mueve una imagen dentro de un contenedor de imágenes a las coordenadas dadas. Las coordenadas se pueden definir como unidades de longitud absoluta, unidades de longitud relativa, palabras clave (`top`, `left`, `center`, `bottom` y `right`), o una combinación válida de ellas (`top 20px right 5px`, `center right 80px`).

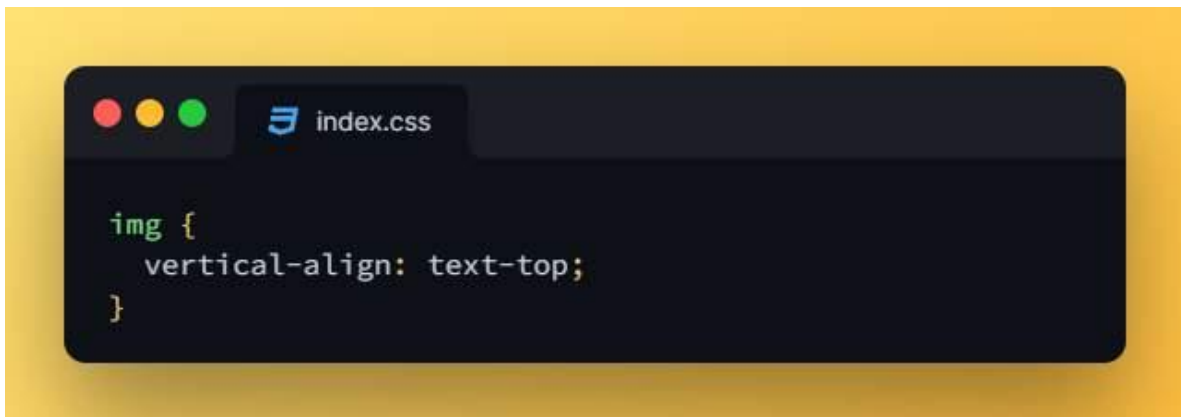


Situar imágenes

Muchas veces agregamos un `` (que por defecto es inline) junto a una cadena de texto para agregar información o simplemente, por estética. En este tipo de casos, alinee el texto y la imagen en la posición deseada puede ser algo un poco complicado, sobre todo si no sabe qué propiedad utilizar.

La propiedad `vertical-align` se puede utilizar para alinear un elemento inline dentro de un contenedor inline y, por lo tanto, puede utilizarse para alinear una imagen con una línea de texto.

Sus posibles valores son: línea base, superior, medio, inferior, sub, texto superior.



Agregar filtros a imágenes

Cuando nuestro sitio web comienza a crecer, utilizamos cada vez más y más imágenes en él, lo cual puede provocar algo de confusión, dificultando que el usuario pueda identificar cuál debe asociar con la información que se encuentra leyendo. O también, puede ser que necesite aplicar una misma imagen en color, escala de grises y sepia en un mismo proyecto y editar una imagen por cada una terminando siendo muy costoso y pesado para el sitio. Para solucionar esto, existe el filtro de propiedad.

Esta propiedad dota de efectos gráficos como el desenfoque o cambio de color en la representación antes de que se muestre el elemento. Los filtros se utilizan para ajustar la representación de imágenes, fondos o bordes.

Suele recibir como valor diferentes funciones CSS, puedes ver todas estas funciones y ejemplos de ellas en este sitio.



Para empezar 🦿

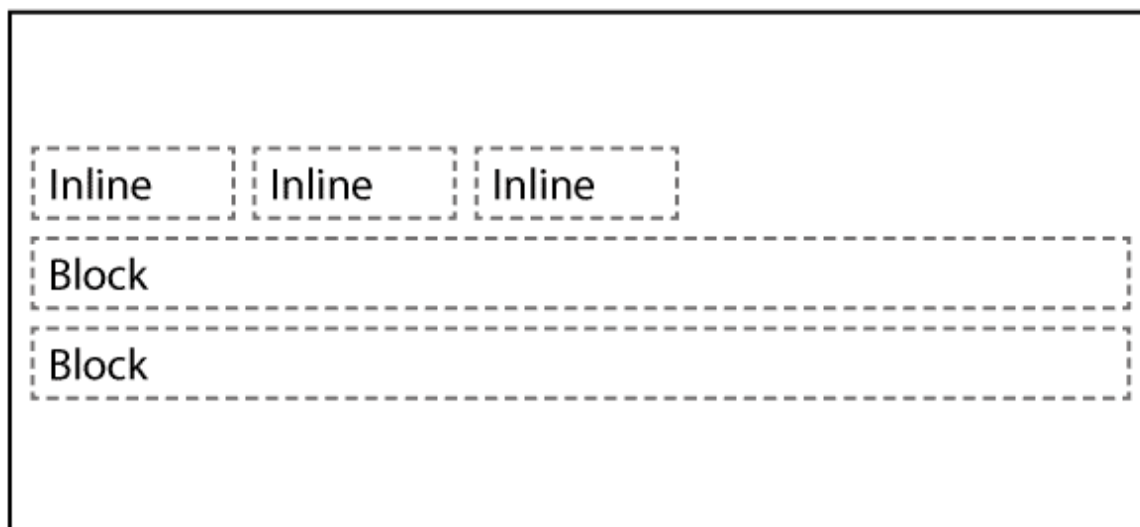
Como hemos podido observar, a la hora de trabajar con diferentes contenidos incrustados en nuestros sitios web, en ocasiones no se comportan como quisiéramos, siendo las líneas de texto de un párrafo muy extenso, o las imágenes demasiado grandes para el espacio en donde están incluidas .

En esta lección hablaremos sobre aquellas propiedades CSS asociadas al tamaño de los elementos, y además de aquellas que nos van a ayudar a crear espacios de descanso visual entre ellos y su contenido.

Tipos de elementos

Cuando hablamos de la forma que tienen los elementos HTML, debemos tener en cuenta que todos son “cajas”. Estas cajas, rectangulares por defecto, ocupan diferentes tamaños según el tipo de elemento. El estándar HTML clasifica a todos sus elementos en dos grandes grupos: elementos en línea (inline) y de bloque (block).

- Los elementos de bloque siempre empiezan en una nueva línea, y ocupan todo el espacio disponible hasta el final de la misma (100%).
- Por otra parte, los elementos en línea no empiezan aunque en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.



Les dejamos un cuadro con todos los elementos HTML que pertenecen a cada uno de los grupos:

Elementos en Línea	Elementos en Bloque
a, abbr, acronym, b, bdo, big, br, button, cite, code, dfn, em, i, img, input, kbd, label, map, object, q, samp, script, select, small, span, strong, sub, sup, textarea, time, tt, var	address, article, aside, audio, blockquote, canvas, dd, div, dl, fieldset, figcaption, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, hr, li, main, nav, noscript, ol, output, p, pre, section, table, tfoot, ul, video

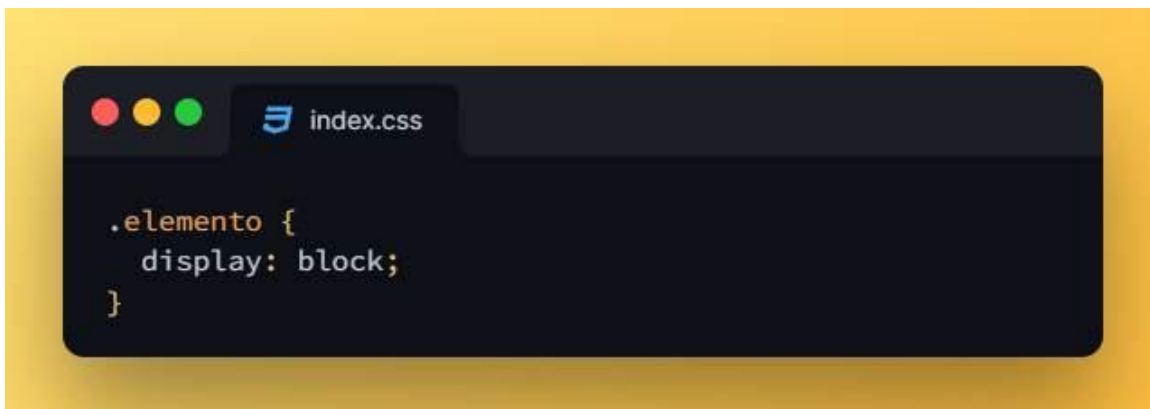
Propiedad Display

La propiedad display, se encarga de definir cómo se ve y comporta un elemento HTML. Las dos funciones más importantes que cumple son:

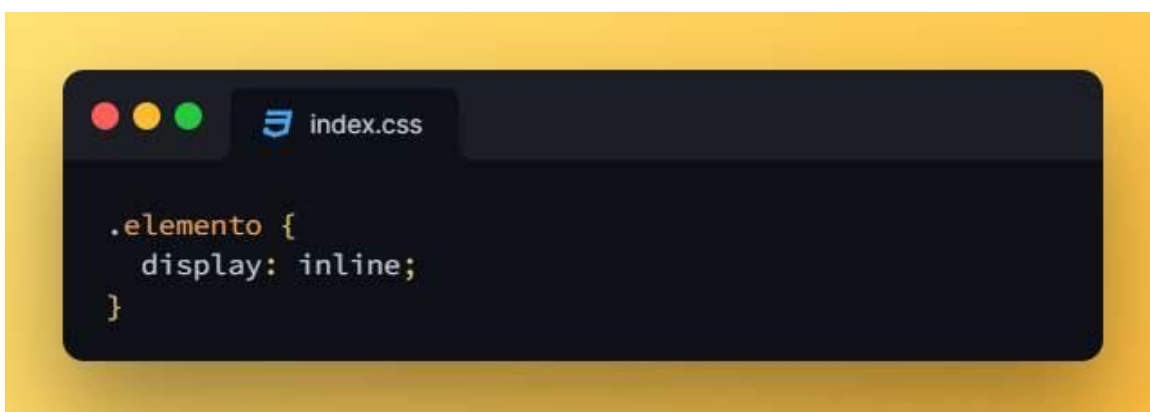
- Convertir un elemento de bloque a uno de línea.
- Convertir un elemento de línea a uno de bloque.

Eso se hace con los valores block e inline respectivamente:

Block : Transforma el elemento en línea en uno de bloque.



Inline: Transforma el elemento en bloque en uno de línea.



Bloque en línea

Hay un valor de la propiedad display que permite tomar lo mejor de ambos grupos, llamado "inline-block". Brinda la posibilidad de tener "padding" y "margin" hacia arriba y abajo, cuando el valor "inline" no nos lo permite. Les dejamos un cuadro comparativo de los tres valores:

A code editor window with a dark background and a yellow title bar. The title bar has three colored circles (red, yellow, green) and a tab labeled 'index.css'. The editor contains the following CSS code:

```
.elemento {  
  display: inline-block;  
}
```

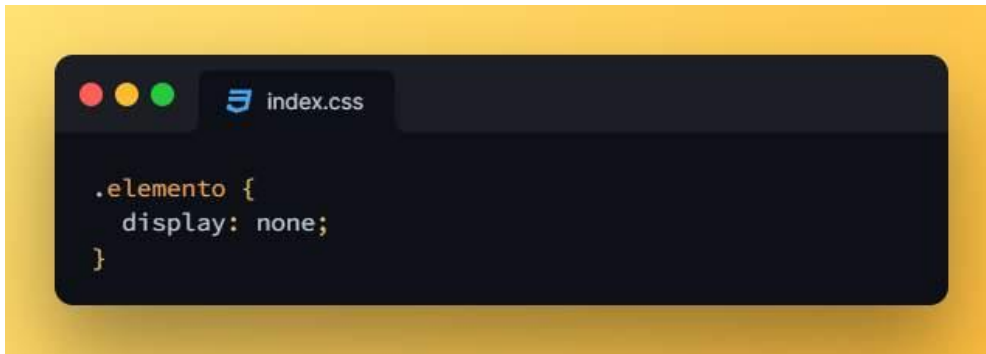
Diferencias entre block y inline

	Width	Height	Padding	Margin
block	SI	SI	SI	SI
inline	NO	NO	Solo izquierdo y derecho	Solo izquierdo y derecho
inline-block	SI	SI	SI	SI

A code editor window with a dark background and a yellow title bar. The title bar has three colored circles (red, yellow, green) and a tab labeled 'index.css'. The editor contains the following CSS code:

```
.elemento {  
  display: inline-block;  
}
```

El display tiene también un valor para quitar un elemento del diseño “display: none”; lo oculta, y además lo quita del flujo del documento (no ocupa espacio en su lugar).

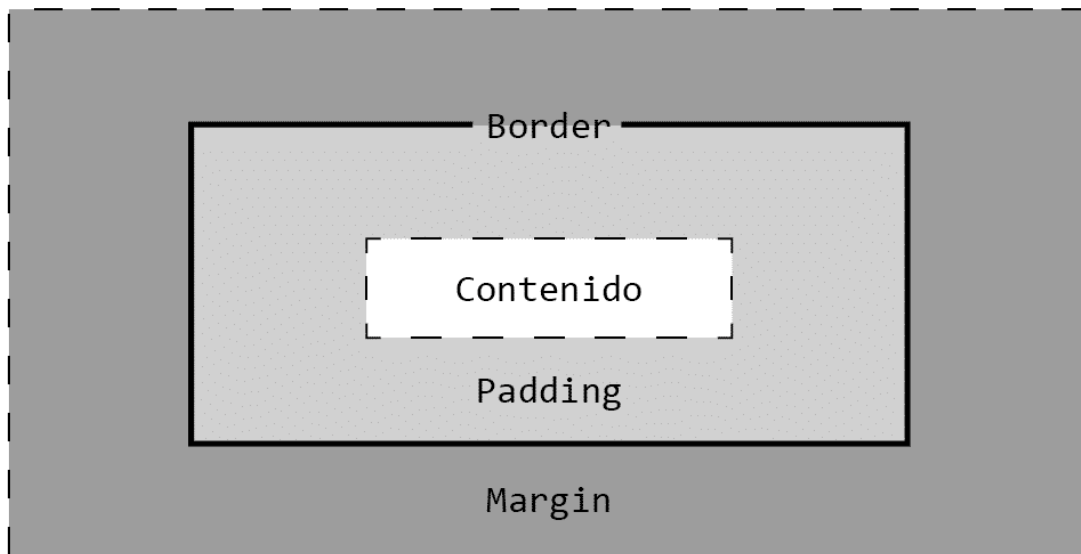


Modelo de caja

El concepto de que “todo es una caja”, da lugar a algo llamado en la web como “box model” o “modelo de caja”. Sin importar si son de línea o de bloque (pero tienen su incidencia en lo que son), todas las etiquetas tienen propiedades en común.

Propiedades en común de los elementos

- **CONTENIDO** Es el espacio para el texto, imagen o contenido incrustado.
- **RELLENO** Es la separación entre el borde y el contenido de la caja. Es un espacio interior.
- **BORDE** Es el límite entre el elemento y el espacio externo.
- **MARGEN** Es la separación entre el borde y el afuera de la caja. Es un espacio exterior.

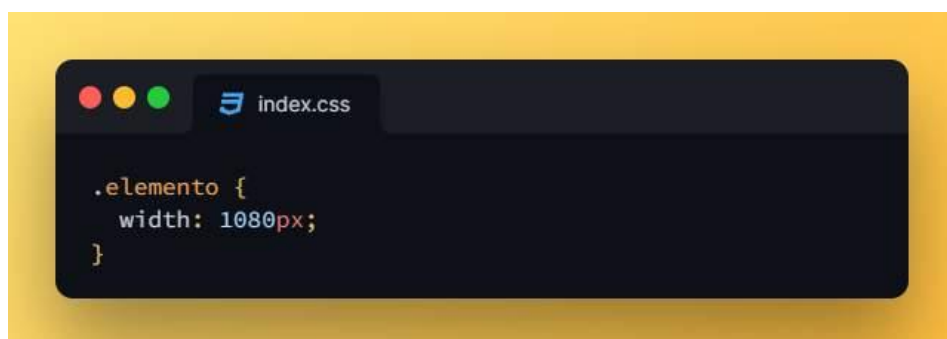


Ancho y alto de la caja

Existen dos propiedades que nos permiten definir el tamaño de los elementos, correspondientes al ancho y alto, veremos ejemplos de cómo usarlas:

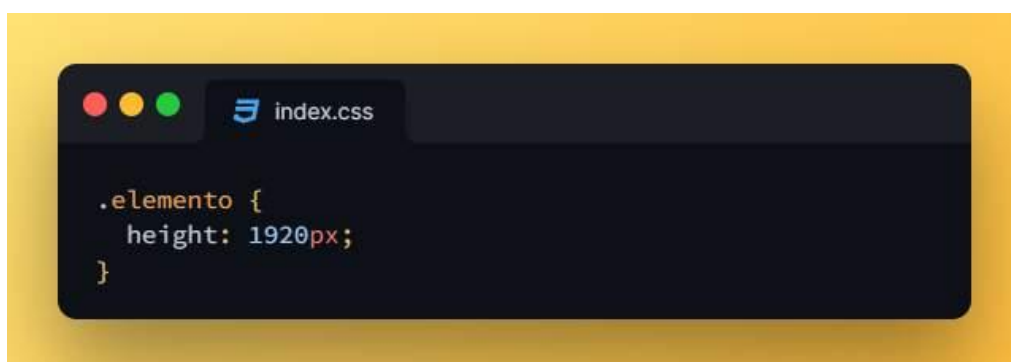
Ancho

Se denomina width a la propiedad CSS que controla la anchura de la caja de los elementos. Dicha propiedad no admite valores negativos, y aquellos en porcentaje se calculan a partir de la anchura de su elemento padre. Puede ser utilizado en cualquier tipo de elemento, incluidas las imágenes.



Alto

La propiedad CSS que controla la altura de la caja de los elementos se denomina altura. No admite valores negativos, y aquellos en porcentaje se calculan a partir de la altura de su elemento padre. Puede ser utilizado en cualquier tipo de elemento, incluidas las imágenes.



Propiedad Margen (márgenes)

Las propiedades margin-top, margin-right, margin-bottom y margin-left se utilizan para definir los márgenes de cada uno de los lados del elemento por separado, pero también puedes definir los cuatro lados al mismo tiempo con su forma abreviada "margin" (los valores se definen en sentido horario - arriba, derecha, abajo, izquierda -) o sólo aquellos que necesiten.

Propiedad Padding (relleno)

Las propiedades padding-top, padding-right, padding-bottom y padding-left se utilizan para definir los espacios internos de cada uno de los lados del elemento por separado, pero también puedes

definir los cuatro lados al mismo tiempo con su forma abreviada. padding” (los valores se definen en sentido horario - top, right, bottom, left -) o sólo aquellos que necesiten



Propiedad Border (bordes)

Las propiedades border-top, border-right, border-bottom, y border-left se utilizan para definir los bordes de cada lado del elemento por separado, pero también puedes definir los cuatro lados al mismo tiempo con su forma abreviada “border” o sólo aquellos que lo necesiten.

A diferencia de los márgenes y padding, los bordes se forman con tres valores:

- tipo de borde(se abre en una nueva pestaña)).
- Grosor (ancho del borde(se abre en una nueva pestaña)).
- Color (borde-color(se abre en una nueva pestaña)).

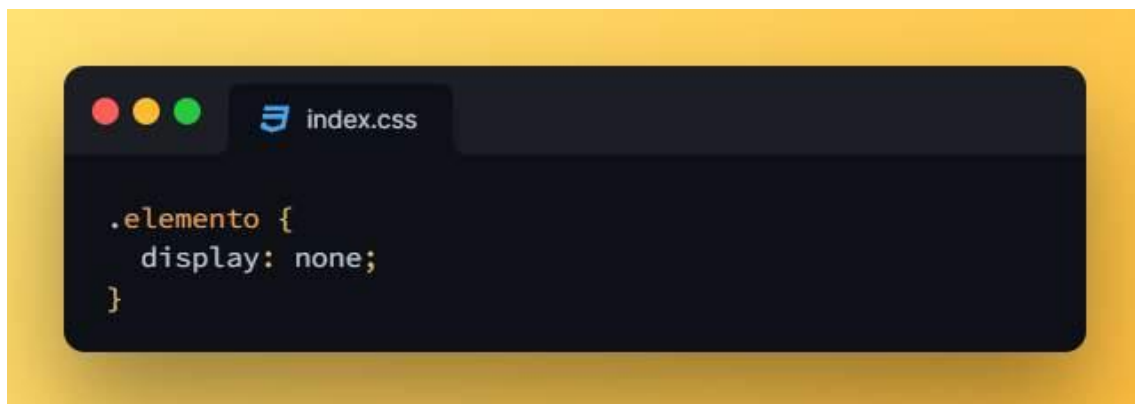


Propiedad Overflow

La propiedad overflow de CSS establece el comportamiento deseado para el desbordamiento de un elemento, es decir, cuando el contenido de un elemento es demasiado grande para caber en su contexto de formato de bloque o línea, en ambas direcciones.

Tiene 4 posibles valores:

- Visible : Es el valor por defecto. El excedente es visible.
- Hidden : El excedente no se muestra (lo recorta y no lo muestra).
- Scroll : Genera una barra de scroll en los dos ejes (X e Y) del elemento, aunque no se necesite.
- Auto : Genera el scroll solo en el eje necesario.



Anteriormente hemos visto muchas propiedades que nos ayudan a cambiar el aspecto visual de nuestros elementos HTML, e incluso, hemos visto otras propiedades que nos permiten cambiar su comportamiento y disposición para hacer más fácil su organización en el campo. Pero, ¿Qué pasaría si te digo que hay una forma más sencilla de organizar los elementos?

En esta lección hablaremos sobre el famoso “Flexbox”, un método de diseño de página unidimensional para organizar elementos de un contenedor padre en filas o columnas.

Definición y Usos

Flexbox es un modo de diseño que nos permite crear estructuras para sitios web de una forma más fácil. Podrás posicionar un elemento en la posición que desees horizontalmente y por si fuera poco también en forma vertical.

Además, no solo puedes posicionar elementos vertical y horizontalmente, sino que puedes establecer cómo se distribuirán, el orden que tendrán e incluso el tamaño que tendrán en proporción a otros elementos. Esto es perfecto para crear diseños adaptables a dispositivos móviles (Responsive Design).

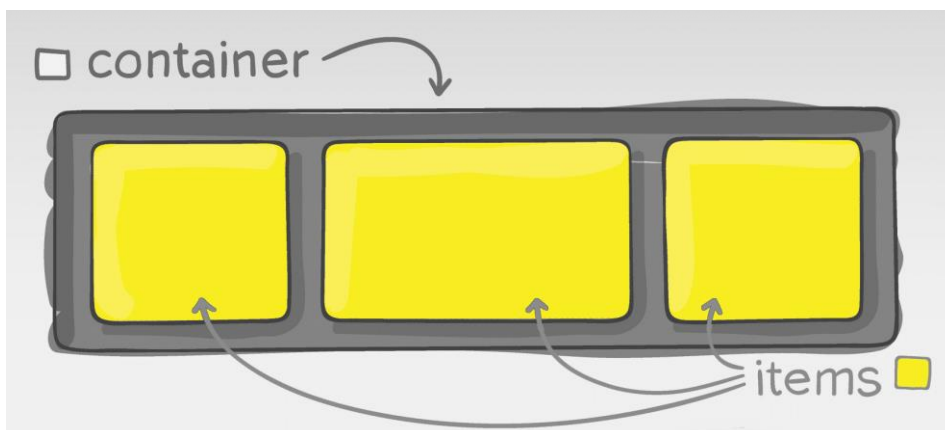
Flexbox no es una propiedad ni un conjunto de propiedades. Flexbox es un nuevo modelo de diseño que viene a incorporarse a los ya existentes en CSS y reemplaza los modelos anteriores como:

- Bloquear Los elementos aparecen uno debajo de otro ocupando todo el ancho disponible.
- En línea Los elementos aparecen uno al lado del otro en una línea y saltan a la línea siguiente al ocupar el espacio disponible
- Mesa Los elementos imitan la distribución de una tabla HTML, con filas, encabezados y columnas.
- Posicionado Los elementos pueden romper el flujo y posicionarse en cualquier lugar del documento.

En fin, debemos saber que un modelo de diseño es un conjunto de algoritmos que determinan el tamaño y la posición de los elementos con respecto a sus hermanos y ancestros. Entonces, ¿Qué se puede hacer con flexbox? Entendiendo lo que significa un modelo de diseño, con flexbox podemos hacer lo siguiente:

- Distribuir los elementos en sentido vertical u horizontal.
- Reordenar la aparición de los elementos sobreescibiendo su aparición en el navegador.
- Ajustar dinámicamente las dimensiones de los elementos para evitar desbordamientos (overflow) respecto a su padre.
- Redefinir el sentido del flujo de los elementos (hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha).
- Alinear los elementos respecto al padre o respecto a sus hermanos.

Conceptos básicos de Flexbox: Para entender bien este modelo de Layout debemos entender algunos conceptos básicos. Primero la disposición flex debe de estar constituida por elementos padres e hijos, el padre será el contenedor Flexible “flex container” y los hijos inmediatos serán los elementos Flexibles “flex item”.



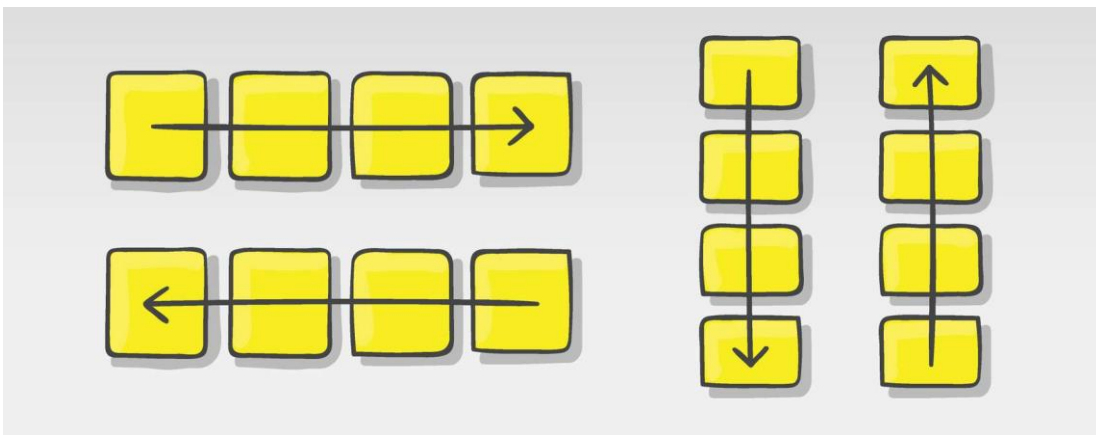
¿Cómo empezamos con Flexbox?

Para comenzar a utilizar Flexbox lo primero que debemos hacer es establecer la propiedad `display` con el valor `flex` en el elemento padre. “`display: flex`” es la única propiedad que necesitamos para configurar el contenedor principal y de esta manera todos sus hijos inmediatos se cambiarán en elementos flexibles de forma automática.

Propiedad `flex-direction`

Esta propiedad me va a permitir manejar el direccionamiento de los elementos flexibles, nos va a permitir especificar si queremos que los elementos flexibles se dispongan en filas o columnas.

Sus posibles valores son: `fila`, `fila-reversa`, `columna`, `columna-reversa`.

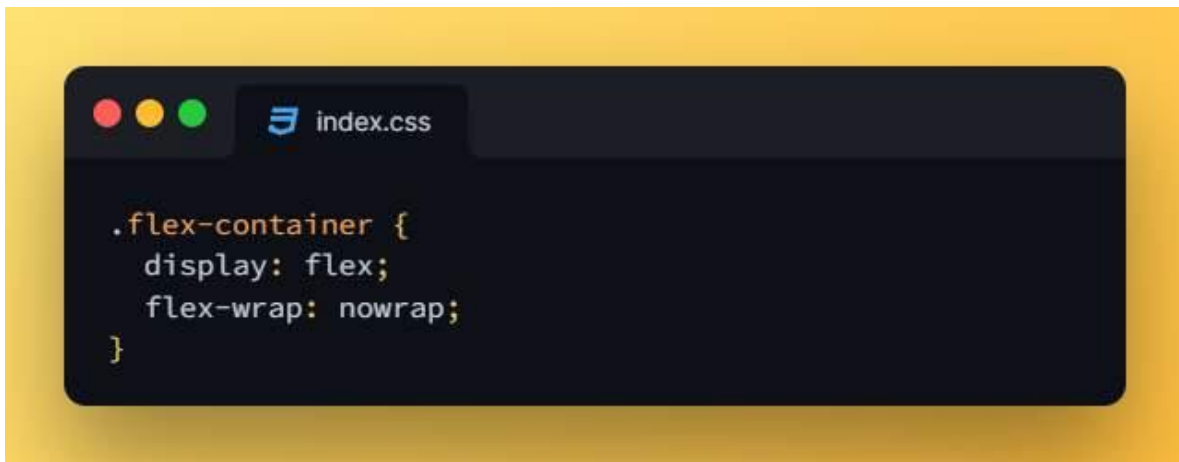
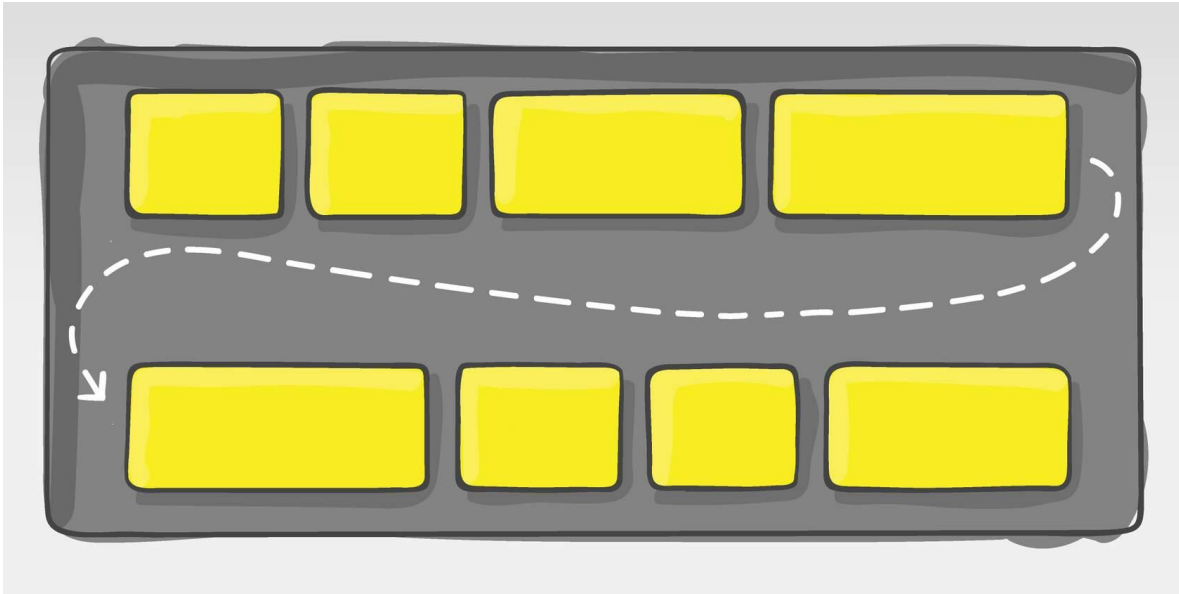


Propiedad `flex-wrap`

El comportamiento inicial del contenedor flexible es poder mantener los elementos flexibles en su eje horizontal sin importar que las dimensiones de estos elementos cambien, pero hay ocasiones donde vamos a querer controlar este alineamiento y hacer que los elementos puedan saltar de línea para poder mantener una apariencia deseado en estos artículos flexibles. Con `flex-wrap`

vamos a poder especificar si queremos que los artículos puedan saltar a una nueva línea si el contenedor flexible se queda sin espacio.

Sus posibles valores son: wrap, wrap-reverse, nowrap.

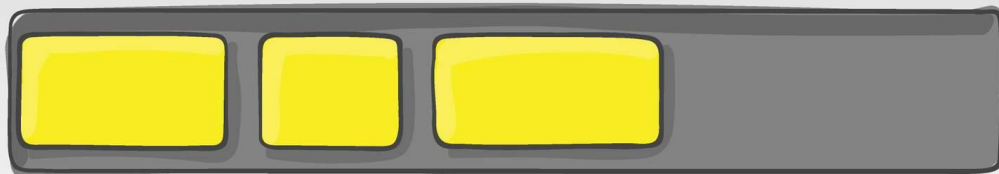


Propiedad justificar-contenido

Justify-content nos va a permitir alinear los elementos en el eje horizontal de la línea actual del contenedor flexible, esto puede ser de forma vertical u horizontal según las particularidades con flex-direction, también nos va a ayudar a distribuir los flex items en el contenedor flexible cuando los artículos no utilizan todo el espacio disponible en su eje principal actual. Esto es declarar la forma en que el navegador debe distribuir el espacio disponible entre los elementos flexibles.

Sus valores más utilizados son: flex-start, flex-end, center, space-between, space-around, space-evenly.

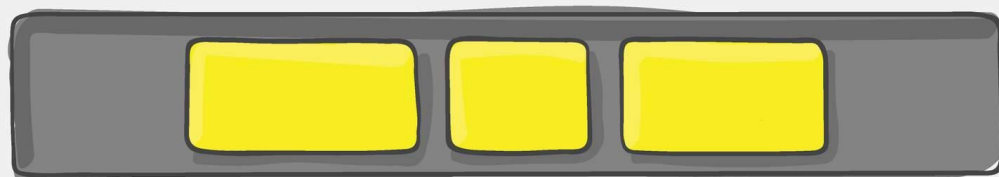
flex-start



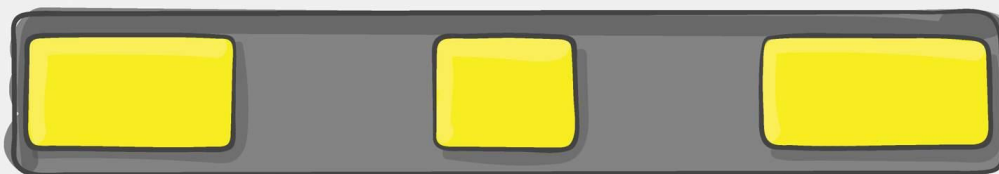
flex-end



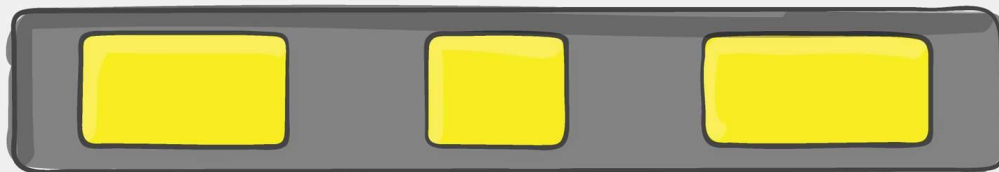
center



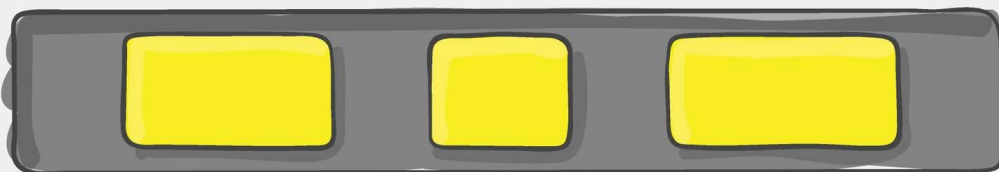
space-between



space-around



space-evenly





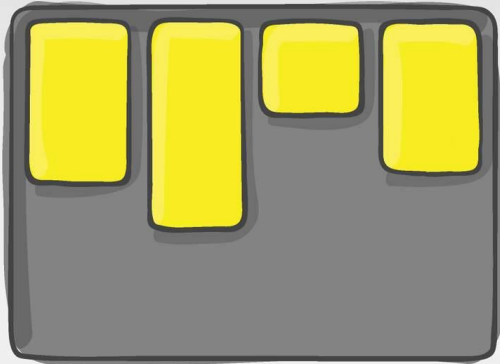
Puedes probar su funcionamiento [aquí\(se abre en una nueva pestaña\)](#). Esta misma propiedad se utiliza para CSS Grids, por lo que verá en el ejemplo de la demostración que se está utilizando para varias filas y columnas, pero la distribución y comportamiento de los elementos sigue siendo la misma.

Propiedad align-items

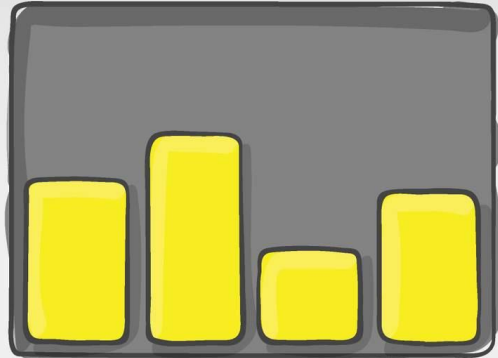
Align-items nos permite establecer la alineación que tendrán por defecto los elementos flexibles, es similar a la propiedad justificar-contenido pero esta vez la dirección es perpendicular. Es decir, align-items nos va a permitir organizar los elementos en el eje secundario del contenedor flex.

Sus valores más utilizados son: stretch, flex-start, flex-end, centro, línea de base.

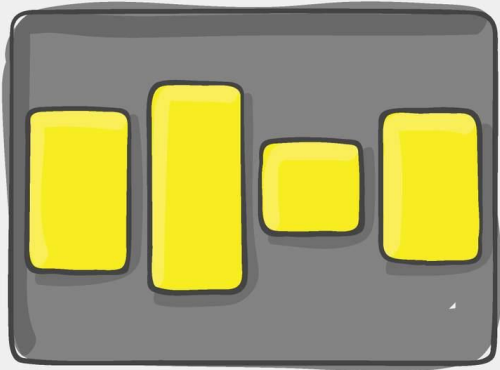
flex-start



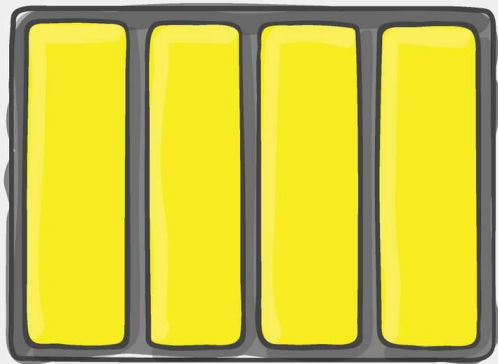
flex-end



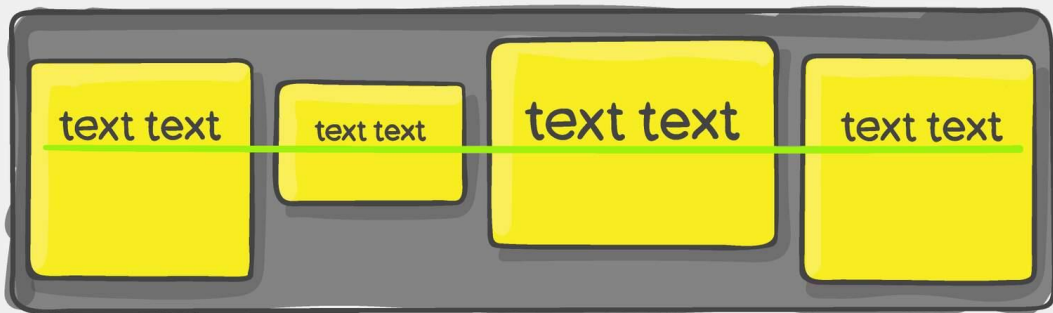
center



stretch



baseline





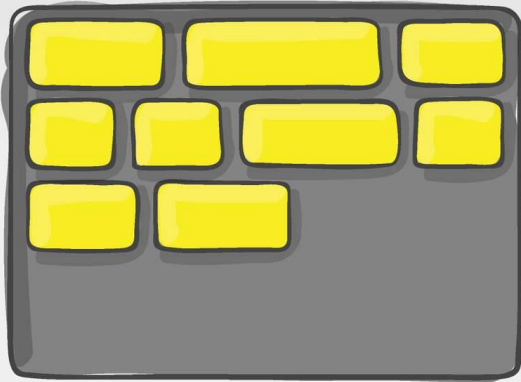
Esta misma propiedad se utiliza para CSS Grids, por lo que verá en el ejemplo de la demostración que se está utilizando para varias filas y columnas, pero la distribución y comportamiento de los elementos sigue siendo la misma.

Propiedad align-content

La propiedad align-content alinea los elementos flexibles cuando estos no usan todo el espacio disponible en el eje vertical del contenedor flexible.

Sus valores más utilizados son: stretch, flex-start, flex-end, center, space-between, space-around, space-evenly.

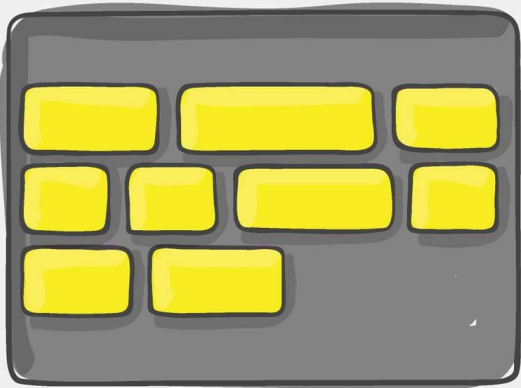
flex-start



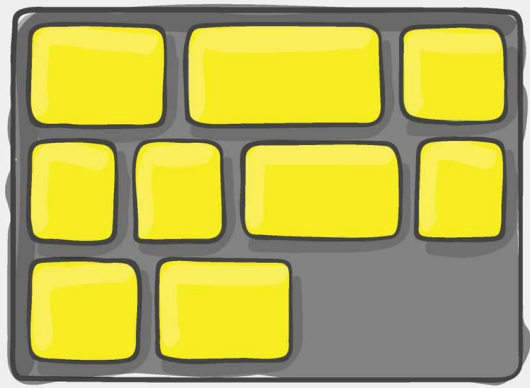
flex-end



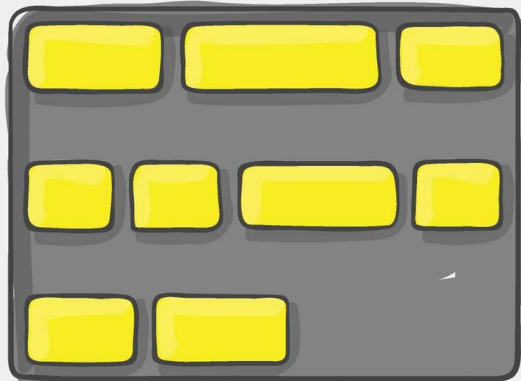
center



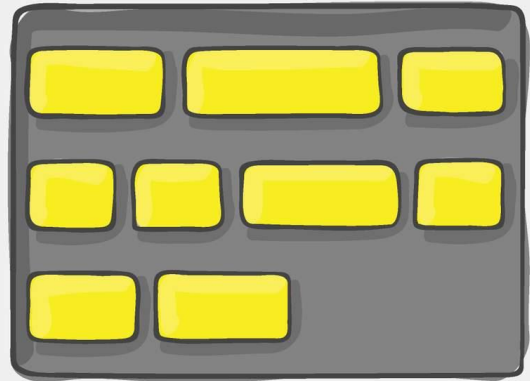
stretch



space-between



space-around

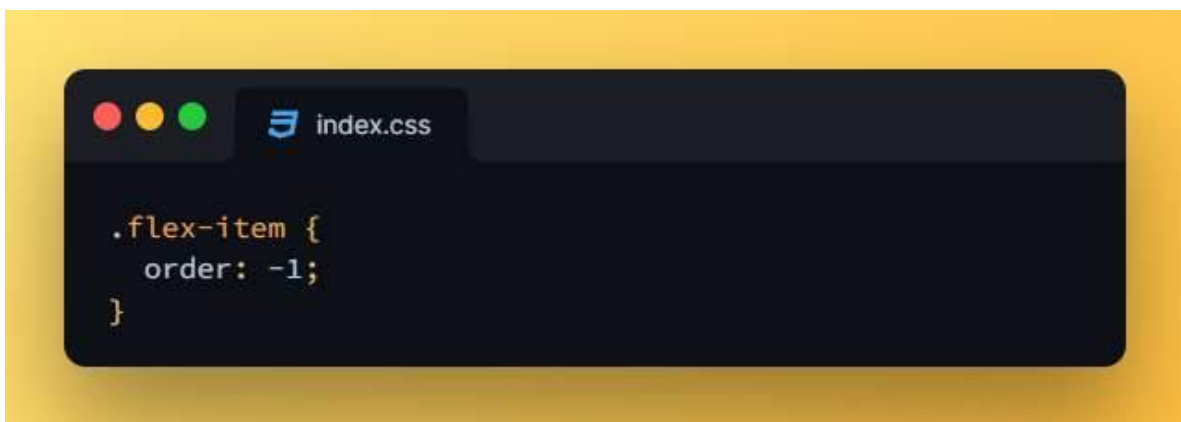
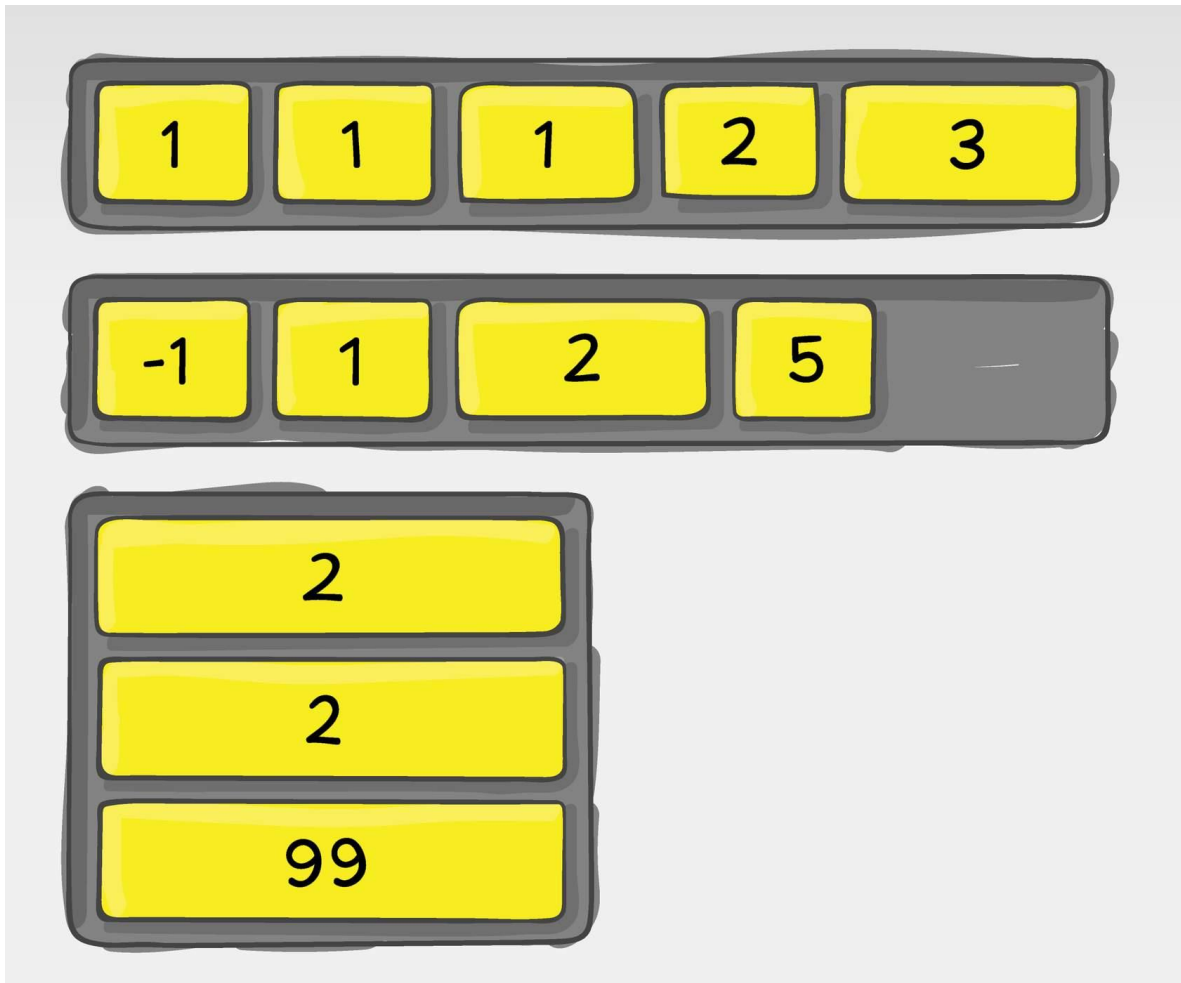




Esta misma propiedad se utiliza para CSS Grids, por lo que verá en el ejemplo de la demostración que se está utilizando para varias filas y columnas, pero la distribución y comportamiento de los elementos sigue siendo la misma.

orden de propiedad

Esta propiedad permite modificar el orden de aparición de un elemento. Recibe como valor números enteros positivos o negativos. Su valor por defecto es 1.



¿Qué hemos aprendido?

En esta lección hemos conocido **Flexbox**, y su gran capacidad para organizar y cambiar la disposición de elementos de forma unidireccional en filas o columnas a partir de su contenedor padre 🚀

En la siguiente lección veremos una herramienta aún más poderosa, que nos facilitará el trabajo de igual forma que Flexbox para **organizar elementos** en conjunto a lo largo de la página, pero en este caso de forma bidimensional, en columnas y filas al mismo tiempo ²