Material descargable

Teoría **CCS**

egg

¡Hola de nuevo! 👋

Te damos la bienvenida a la teoría sobre CSS.

En la unidad anterior aprendimos acerca del papel fundamental de HTML (Hypertext Markup Language) en la estructuración lógica y semántica del contenido de una página web. Abordamos la creación de encabezados, párrafos, listas y enlaces, y cómo utilizar las etiquetas adecuadas para organizar la información de manera eficiente.

Ahora veremos qué es CSS, sus fundamentos y generalidades y cómo implementarlo en nuestros documentos HTML.

¡Que comience el viaje! 🚀

¿Qué es CSS?

CSS es el acrónimo de **Cascading Style Sheets**, o lo que sería en español Hojas de Estilo en Cascada. Es un lenguaje que sirve para especificar el estilo o aspecto de las páginas web. CSS se define basándose en un estándar publicado por una organización llamada W3C, que también se encarga de estandarizar el propio lenguaje HTML.

¿Qué motivó su creación?

El lenguaje HTML está limitado a la hora de aplicar forma a un documento. Sirve de manera excelente para especificar el contenido que debe tener una página web, pero no permite definir cómo ese documento se debe presentar al usuario.

Otro motivo que ha hecho necesaria la creación de CSS ha sido la separación del contenido de la presentación. Al inicio las páginas web tenían mezclado en su código HTML el contenido con las etiquetas necesarias para darle forma. Esto

tiene sus inconvenientes, ya que la lectura del código HTML se hace pesada y difícil a la hora de buscar errores o depurar las páginas. Además, desde el punto de vista de la riqueza de la información y la utilidad de las páginas a la hora de almacenar su contenido, es un gran problema que los textos estén mezclados con etiquetas incrustadas para dar forma a estos, pues se degrada su utilidad.

CSS como solución

Como hemos visto, para facilitar un correcto mantenimiento de las páginas web y para permitir que los diseñadores pudieran trabajar como sería deseable, había que introducir un nuevo elemento en los estándares y este fue el lenguaje CSS.

CSS se ideó para aplicar el formato en las páginas, de una manera mucho más detallada, con nuevas posibilidades que no estaban al alcance de HTML. Al mismo tiempo, gracias a la posibilidad de aplicar el estilo de manera externa al propio documento HTML, se consiguió que el mantenimiento de las páginas fuese mucho más sencillo.

Ventajas y características

El modo de funcionamiento de CSS consiste en definir, mediante una sintaxis especial, la forma de presentación que les aplicaremos a los elementos de la página.

Podemos aplicar CSS a muchos niveles, desde un sitio web entero hasta una pequeña etiqueta. Estos son los principales bloques de acción.

- **Una web entera:** de modo que se puede definir en un único lugar el estilo de toda una web, de una sola vez.
- **Un documento HTML o página en particular:** se puede definir la forma de cada uno de los bloques de contenido de una página, en una declaración que afectará a un solo documento de un sitio web.
- **Una porción del documento:** aplicando estilos visibles en un trozo de la página, como podría ser la cabecera.
- **Una etiqueta en concreto:** llegando incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante, ya que ofrece potencia en nuestra programación.

Esta sintaxis CSS permite aplicar al documento formato de modo mucho más exacto. Si antes el HTML era insuficiente para maquetar las páginas y teníamos que utilizar ciertas etiquetas para conseguir nuestros efectos, ahora tenemos muchas más herramientas que nos permiten definir esta forma:

- Podemos definir la distancia entre líneas del documento.
- Se puede aplicar indentado (sangrado) a las primeras líneas del párrafo.
- Podemos colocar elementos en la página con mayor precisión, y sin lugar a errores.
- Puede definirse la visibilidad de los elementos, márgenes, subrayados, tachados, etc.

Otra ventaja importante de CSS es la capacidad de especificar las medidas con diversas unidades. Si con HTML solamente podíamos definir atributos en las páginas con píxeles y porcentajes, ahora podemos hacerlo utilizando más unidades como:

- Píxeles (px) y porcentaje (%), como antes.
- Pulgadas (in).
- Puntos (pt).
- Centímetros (cm).

Sintaxis CSS

La meta básica del lenguaje Cascading Stylesheet (CSS) es permitir al motor del navegador pintar elementos de la página con características específicas, como colores, posición o decoración. La sintaxis CSS refleja estas metas y estos son los bloques básicos de construcción:

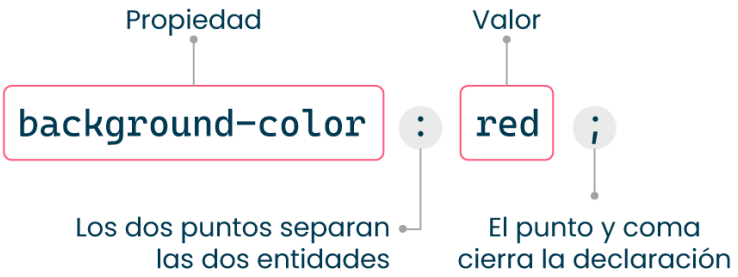
- **La propiedad:** Es un identificador, un nombre legible por humanos, que define qué característica es considerada.
- **El valor:** Describe como las características deben ser manejadas. Cada propiedad tiene un conjunto de valores determinados, definido por una gramática formal, así como un significado semántico, implementados por el motor del navegador.

Declaraciones de CSS

La función principal de CSS es configurar determinadas propiedades con valores específicos. **Este par (propiedad y valor) es llamado una declaración.**

Ambas propiedades y valores son sensibles a mayúsculas y minúsculas en CSS. El par se separa por dos puntos, **“ : ”**, y espacios en blanco antes, entre ellos y después.

Declaración CSS



Hay más de 100 propiedades diferentes en CSS y cerca de un número infinito de diferentes valores. No todos los pares de propiedades y valores son permitidos, cada propiedad define qué valores son válidos. Cuando un valor no es válido para una propiedad específica, la declaración es considerada inválida y es completamente ignorada por el motor del CSS.

Bloques de declaraciones

Las declaraciones pueden ser agrupadas en bloques, que es una estructura **delimitada por una llave de apertura, ‘{‘, y una de cierre, ‘}’.** Los bloques en ocasiones pueden anidarse, por lo que las llaves de apertura y cierre deben de coincidir.

Bloque CSS:

```
selector {
   Aquí escribiremos las declaraciones CSS
}
```

Las llaves delimitan el inicio y el final del bloque.

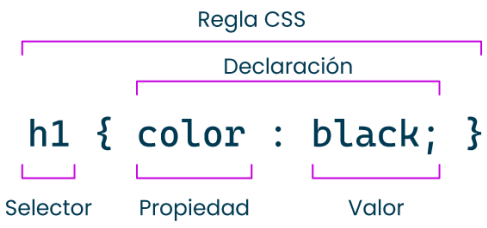
Esos bloques son naturalmente llamados bloques de declaraciones y las declaraciones dentro de ellos están separadas por un punto y coma, **“ ; ”**. Un bloque de declaración puede estar vacío (declaración nula). En este caso, los espacios en blanco alrededor de las declaraciones son ignorados. En cuanto a la última declaración de un bloque, esta no necesita terminar en un punto y coma, aunque es usualmente considerado una buena práctica porque previene el olvidar agregarlo cuando se extienda el bloque con otra declaración.

Bloque CSS con declaraciones:

```
{
    background-color : red;
    background-style : none;
}
```

¿Qué partes conforman a una declaración en CSS?

El siguiente esquema muestra las partes que forman un estilo CSS muy básico:



Los diferentes términos se definen a continuación:

- **Regla:** cada uno de los estilos que componen una hoja de estilos CSS.
- **Selector:** indica el o los elementos HTML a los que se aplicará la regla CSS.

- **Declaración:** especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad:** permite modificar el aspecto de una característica del elemento.
- **Valor:** indica el nuevo valor de la característica modificada en el elemento.

Incluir CSS a HTML

Hablemos de cómo incluir CSS en un documento HTML, una tarea fundamental para dar estilo y personalidad a nuestras páginas web. Sin embargo, es importante tener en cuenta que hay distintas maneras de hacerlo y todas tienen sus momentos y lugares apropiados.

Comenzaremos desde el nivel más específico hasta el más general, es decir, aumentaremos la dificultad y la importancia progresivamente.

Pequeñas partes de la página

Estilo Inline

Para definir estilos en partes específicas de una página, utilizamos el atributo **style** en la etiqueta correspondiente. La sintaxis CSS para este atributo determina las características de los estilos. Veamos un ejemplo en el que ciertas palabras de un párrafo se muestran en color amarillo:

 Yo estudié en Egg Cooperation programación

💡 Recuerda: Aunque este método es útil para casos puntuales, no es recomendable para estilos a gran escala porque dificulta la mantenibilidad del código.

Estilo a nivel de elemento

Este método permite dar un estilo específico a una etiqueta completa. Por ejemplo, podemos hacer que un párrafo completo sea de color rojo. Para esto, utilizamos el atributo **style** en la etiqueta en cuestión.

Esto es un párrafo de color rojo.

Estilo a nivel de bloque

Utilizando la etiqueta ** **, podemos definir secciones de una página y aplicarle estilos con el atributo style. Así, podemos definir estilos para todo un bloque de la página de una sola vez.

">

<h3>Estas etiquetas van en azul y negrita</h3>

Seguimos dentro del DIV, por lo tanto permanecen los estilos

Estilo a nivel de página

Una opción más potente y cómoda es definir estilos a nivel de página completa en la cabecera del documento. Esto nos permite evitar "ensuciar" las etiquetas HTML colocando el atributo **style**. Además, podemos reutilizar estos estilos en diferentes etiquetas a lo largo del documento, simplificando el código y facilitando los cambios de estilo a gran escala.

A grandes rasgos, entre **<style>** y **</style>**, se coloca el nombre de la etiqueta (o selector) para la que queremos definir los estilos y entre llaves -{ }- colocamos en sintaxis CSS las características de estilos. El concepto de selectores lo veremos más adelante.

```
<html>
<head>
  <title>Ejemplo de estilos para toda una página</title>
  <style>
    h1 { text-decoration: underline; text-align: center }
    p { font-family: Arial, Verdana; color: white; background-color: black }
    body { color: black; background-color: #cccccc; text-indent: 1cm }
  </style>
</head>
<body>
  <h1>Pagina con estilos</h1>
  <p>Pagina con estilos de ejemplo</p>
</body>
</html>
```

Como se puede apreciar en el código, hemos definido que la etiqueta <h1> se presentará

- Subrayado
- Centrada

También, por ejemplo, hemos definido que el cuerpo entero de la página (etiqueta <body>) se le apliquen los estilos siguientes:

- Color del texto negro
- Color del fondo grisáceo
- Margen lateral de 1 centímetro

💡 **Nota**: Los estilos definidos para un elemento padre son heredados por los elementos hijos, a menos que estos últimos tengan estilos propios.

Estilo a nivel de sitio web

El método más poderoso y recomendado para el desarrollo de hojas de estilo es **el uso de una hoja de estilo externa (archivo .css).** Este archivo solo contiene las declaraciones de estilo de la página y se enlaza a todas las páginas del sitio. Esto permite a todas las páginas compartir una declaración de estilos, reutilizando el código CSS de una manera más eficiente y manteniendo una coherencia en el diseño.

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

💡 **Beneficios**: reduce las líneas de código HTML, mejora la velocidad de descarga, separa adecuadamente el contenido (HTML) de la presentación (CSS) y facilita el mantenimiento del sitio web.

Incorporar estilos desde una hoja de estilos

Veamos ahora cómo es el proceso para incluir estilos con un archivo externo.

Crear una Hoja de Estilos Externa

Primero, necesitaremos un archivo de texto con la extensión .css. Este es el lugar donde escribiremos nuestro código CSS. Este archivo no debe contener ningún código HTML, solo CSS. Aquí hay un ejemplo que denominaremos **"estilos.css".**

```css
p {
  font-size: 12px;
  font-family: Arial, Helvetica;
  font-weight: normal;
}

h1 {
  font-size: 36px;
  font-family: Verdana, Arial;
  text-decoration: underline;
  text-align: center;
  background-color: Teal;
}

body {
  background-color: #006600;
  font-family: Arial;
  color: White;
}
```

Enlazar la Hoja de Estilos a tu Página Web

Ahora necesitamos vincular la hoja de estilos que acabamos de crear a nuestra página web. Para ello, usaremos la etiqueta **<link>** dentro de la etiqueta **<head>** de nuestro HTML. Esta etiqueta <link> necesita dos atributos:

- **rel:** Este atributo describe la relación entre la hoja de estilos y la página HTML. Para los archivos CSS, siempre usamos el valor "stylesheet".
- **href:** Este atributo especifica la ruta al archivo CSS. Esta ruta puede ser absoluta o relativa, y puede apuntar a un recurso interno o externo.

Aquí está un ejemplo de cómo se ve una página web completa que enlaza a la hoja de estilos que hemos creado:

```html
<html>
  <head>
    <link rel="stylesheet" href="estilos.css">
    <title>Ejemplo de página con estilos externos </title>
  </head>
  <body>
    <h1>Página con estilos</h1>
    <p>Página con estilos de ejemplo</p>
  </body>
</html>
```

Selectores CSS

Teniendo en cuenta que ya podemos asignarle estilos a todo un sitio web, mediante un archivo css que usa selectores para elegir las etiquetas a las que asignarles los estilos, también tenemos que entender que existen varios tipos de selectores.

Selector universal

Se utiliza para seleccionar todos los elementos de la página. El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML (por ahora no es importante fijarse en la parte de la declaración de la regla CSS):

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

Selector de etiqueta

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {
  text-align: justify;
  font-family: Verdana;
}

/* El siguente ejemplo selecciona todas las tablas y div de la página:*/

table, div {
  border: 1px solid red;
}
```

Selector descendente

Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

El selector del siguiente ejemplo selecciona todos los elementos de la página que se encuentren dentro de un elemento <p>.

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

Selector de clase

¿Qué hacer para aplicar estilos solo al primer párrafo?

Una de las soluciones más sencillas para aplicar estilos a un solo elemento de la página consiste en utilizar el atributo class de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar.

Ejemplo:

HTML:

```
<body>
  <p class="destacado"> Párrafo 1</p>
  <p class="error"> Párrafo 2</p>
  <p> Párrafo 3</p>
</body>
```

CSS:

```
.destacado {
  font-size: 15px;
}
.error {
  color: red;
}
```

En nuestro archivo CSS para especificar una clase, vamos a poner punto (‘.’) y el nombre de la clase que queremos que coincida con valor que pongamos en nuestro atributo class en el html.

Entonces, en el ejemplo podemos ver como el primer párrafo tiene el valor destacado y el segundo párrafo el valor error para el atributo class y en nuestro archivo CSS, hemos definido un estilo para esas clases.

El beneficio del atributo class, además de dejarnos asignar estilos a un solo elemento, es que después podemos reutilizar esa class para asignarle ese estilo a otros párrafos concretos o a otras etiquetas, solo deberemos ponerle el valor de un estilo que ya existe en el atributo class.

Selector de ID

En un documento HTML, los selectores de ID de CSS buscan un elemento basado en el contenido del atributo id. El atributo ID del elemento seleccionado debe coincidir exactamente con el valor dado en el selector. Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo id no se puede repetir en dos elementos diferentes de una misma página.

HTML:

```
<div id="identificador"> ¡Este div tiene un ID especial! </div>
<div> Este solo es un div regular. </div>
```

CSS:

```
#identificador{
background-color: blue;
}
```

En nuestro archivo CSS para especificar un ID, vamos a poner el numeral ('#') y el nombre del ID que queremos que coincida con el valor que pongamos en nuestro atributo ID en el html.

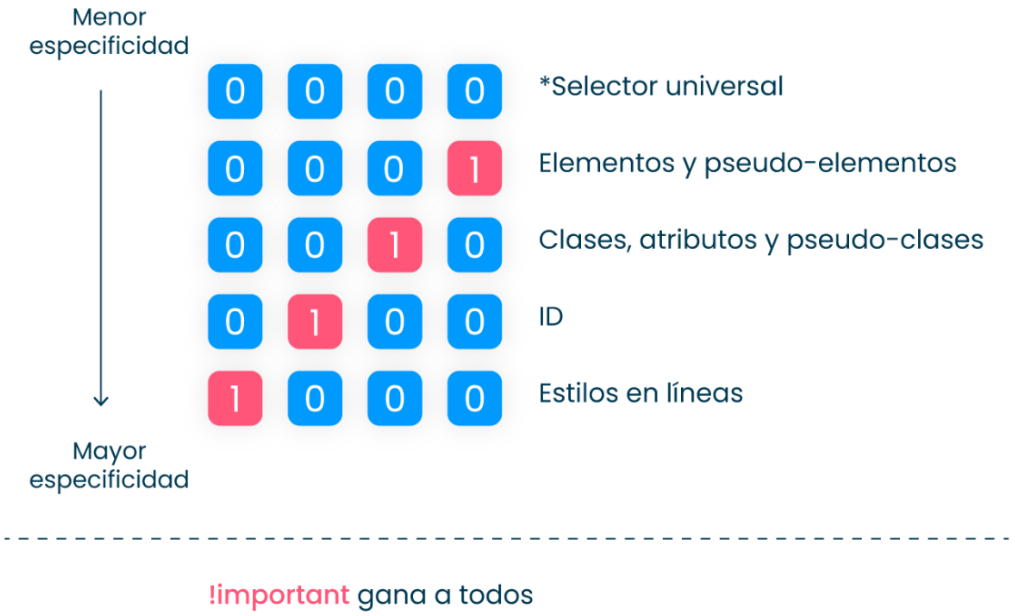
Como podemos ver el ID, es muy parecido al atributo class, pero la diferencia es que el ID se puede usar para identificar un solo elemento, mientras que una clase se puede usar para agrupar más de uno.

Reglas de especificidad

Las ***reglas de especificidad de los selectores de CSS*** determinan qué estilo se aplicará a un elemento HTML cuando existen múltiples reglas que compiten por dar estilo al mismo elemento.

Cada selector tiene un valor de especificidad que se basa en su tipo (ID, clase, elementos o universales) y en su posición en la hoja de estilos. El selector con la mayor especificidad ganará y se aplicará a ese elemento en particular:

- Selectores universales: Tienen una especificidad de 0, es decir, no se les asigna ningún valor de especificidad.
- Selectores de elementos y pseudo-elementos: Tienen una especificidad de 1.
- Selectores de clase, atributo y pseudo clase: Tienen una especificidad de 10.
- Selectores de ID: Tienen una especificidad de 100.
- Estilos en líneas: Tienen una especificidad de 1000.



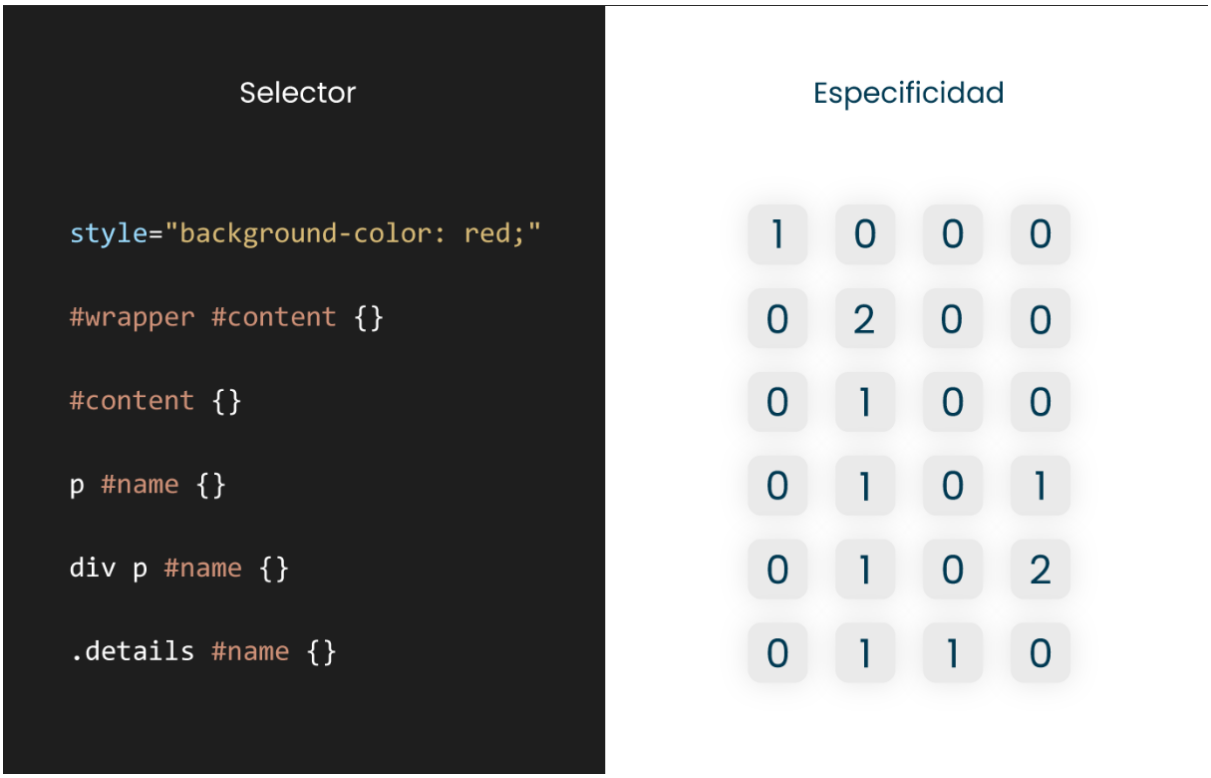
El valor de especificidad se calcula sumando los valores de los diferentes tipos de selectores utilizados en una regla de estilo. Un estilo en línea (que luego veremos con mayor profundidad) tiene la máxima prioridad en CSS con una especificidad mayor que la de cualquier selector de ID, clase o elemento. Un selector de ID tiene un valor de especificidad mayor que un selector de clase, y un selector de clase tiene un valor mayor que un selector de elementos.

Por ejemplo, si hay un selector de elemento y un selector de clase que están aplicando diferentes estilos a un mismo elemento HTML, el selector de clase tendrá una especificidad mayor y, por lo tanto, su estilo será el que se aplicará al elemento.

En otras palabras, **cuando se aplican múltiples reglas a un elemento aquella que sea más específica será la que se aplique a este último. Si dos reglas tienen la misma especificidad, se aplica la que aparece más tarde en el código CSS.**

📖 También **existe el "!important" que no provee especificidad pero sobreescribe cualquier otra regla sin importar cuanta especificidad tenga.** No se recomienda su uso porque es difícil de depurar y no respeta la cascada de estilos.

Para que quede un poco más claro, veamos un ejemplo de diferentes selectores con su correspondiente valor de especificidad:



Es importante tener en cuenta las reglas de especificidad al diseñar hojas de estilo para evitar conflictos y asegurarse de que los estilos se apliquen correctamente.

💡*En el caso de los anidamientos, los puntos de especificidad se suman. Por ejemplo*:

- *Con el siguiente HTML*:

```
<div class="container">
<h1 class="title">Título</h1>
</div>
```

- *Y el siguiente código CSS*:

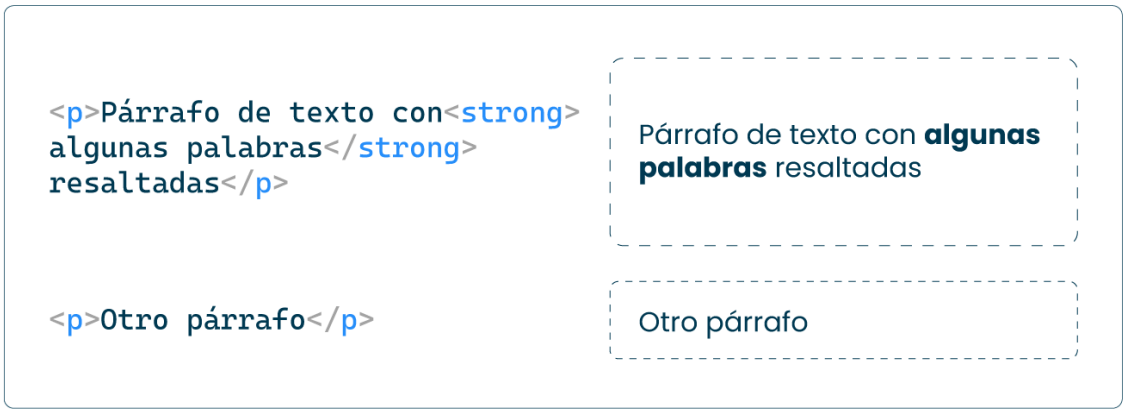
```
.container .title {
    color: red;
}
```

Este selector tiene una especificidad de 20 (10 por la clase .container y 10 por la clase .title). Si se aplicara un estilo diferente al mismo elemento utilizando un selector con una especificidad de 10, el selector más específico tendría prioridad y se aplicaría el estilo correspondiente.

Modelo de caja

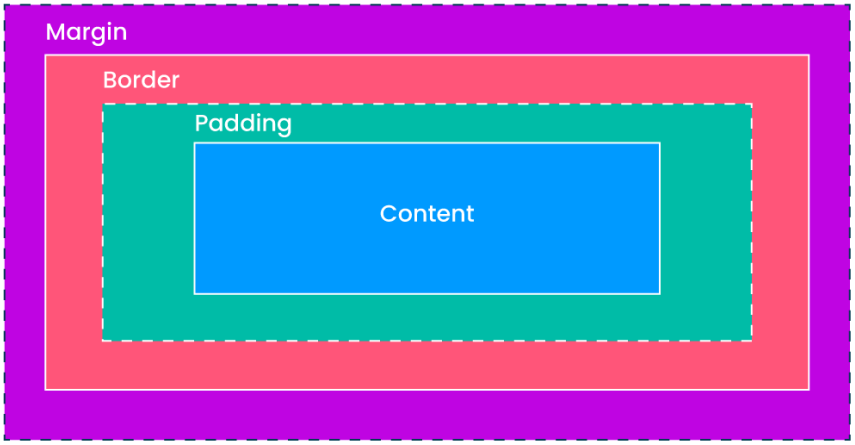
El modelo de cajas o "box model" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares.

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. La siguiente imagen muestra las tres cajas rectangulares que crean las tres etiquetas HTML que incluye la página:



Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde.

Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características. Cada una de las cajas está formada por cuatro partes, tal y como muestra la siguiente imagen:



- **Contenido (content):** se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Relleno (padding):** espacio libre opcional existente entre el contenido y el borde.
- **Borde (border):** línea que encierra completamente el contenido y su relleno.
- **Margen (margin):** separación opcional existente entre la caja y el resto de las cajas adyacentes.

Existen otras dos partes de una caja que son:

- **Imagen de fondo (background image):** imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo (background color):** color que se muestra por detrás del contenido y el espacio de relleno.

El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).

Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, si la imagen de fondo no

cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo. Combinando imágenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.

Propiedad de Display

Cada elemento HTML tiene un valor de visualización predeterminado según el tipo de elemento que sea. El valor de visualización predeterminado para la mayoría de los elementos es **block** o **inline**.

- Los elementos de **bloque (block)** ocupan todo el ancho de la pantalla
- Los elementos **en línea (inline)** sólo ocupan el espacio necesario.

Otras propiedades de display que existen son:

- **none**: Oculta completamente el elemento, es decir, no se muestra en la página web. Es muy útil para hacer que un elemento desaparezca temporalmente o para ocultarlo en diferentes condiciones de visualización (esto se logra manipulando el CSS con javascript).
- **inline-block**: Combina las características de los elementos inline y los elementos block. Los elementos inline-block se muestran como elementos en línea, pero permiten establecer ancho y altura como si fueran elementos de bloque.
- **inherit**: Hace que el elemento herede el valor de la propiedad display de su elemento padre.
- **flex**: Convierte un elemento en un contenedor flexible, lo que permite controlar el tamaño y la posición de sus elementos hijos.
- **grid**: Convierte un elemento en un contenedor de cuadrícula, lo que permite colocar elementos hijos en filas y columnas.

👉 **Ver video sobre display**

Propiedad de Position (posición)

Algunos valores comunes para la propiedad de position (“posicionamiento”) incluyen:

- **static:** Este es el valor predeterminado de la propiedad de "position". Cuando se establece el valor de "static", no se aplica ningún posicionamiento especial al elemento y se colocará en el lugar que le corresponda en el flujo normal del documento.
- **relative:** El elemento se posiciona en relación con su posición normal, es decir, puede moverse hacia arriba, abajo, izquierda o derecha desde su posición inicial.
- **absolute:** El elemento se posiciona en relación a su “ancestro posicionado” más cercano, es decir, al “elemento padre” que tenga un valor de posición distinto a "static". Si no hay ningún “ancestro posicionado”, el elemento se posiciona en relación al bloque contenedor inicial del documento (generalmente al “body”). Cuando se establece el valor de "absolute" en un elemento, se elimina del flujo normal del documento, lo que significa que no afecta a la posición de otros elementos en la página.
- **fixed:** El elemento se posiciona en relación con la ventana del navegador, lo que significa que permanece en la misma posición incluso cuando se desplaza la página.

  Cuando se establece el valor de "fixed" en un elemento, también se elimina del flujo normal del documento. El elemento se coloca en la posición especificada con las propiedades "top", "right", "bottom" y "left". Si no se especifica ninguna de estas propiedades, el elemento se coloca en la esquina superior izquierda de su “ancestro posicionado” más cercano.

- **sticky**: El elemento se comporta como un elemento posicionado relativamente hasta que alcanza una posición específica en la página, donde se vuelve fijo.
- **inherit**: Hereda el comportamiento del elemento padre para dicha propiedad.

👉 **Ver video sobre position**

💡 **Tip:** Puedes utilizar el centro de práctica de W3School para afianzar los conceptos y ejercitar tanto la propiedad **display** como **position**

Pseudo-clases

CSS también permite aplicar diferentes estilos a un mismo enlace en función de su estado. De esta forma, es posible cambiar el aspecto de un enlace cuando por ejemplo el usuario pasa el ratón por encima o cuando el usuario pincha sobre ese enlace.

Como con los atributos id o class no es posible aplicar diferentes estilos a un mismo elemento en función de su estado, CSS introduce un nuevo concepto llamado **pseudo-clases**. En concreto, CSS define las siguientes cuatro pseudo-clases:

- **:link,** aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- **:visited,** aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- **:hover**, aplica estilos al enlace sobre el que el usuario ha posicionado el puntero del ratón. • :active, aplica estilos al enlace que está clickeado el usuario.

Esto se vería de la siguiente forma:

```
/* link sin visitar */
a:link {
color: red;
}
/* link visitado*/
a:visited {
color: green;
}
/* mouse sobre el link*/
a:hover {
  color: pink;
```

```
}
/* link clickeado */
a:active {
color: blue;
}
```

Tablas en CSS

Bordes de la tabla

Para especificar los bordes de la tabla en CSS, use la **border** propiedad.

El siguiente ejemplo se especifica un borde sólido para los elementos **<table>, <th> y <td>:**

```
table, th, td {
  border: 1px solid;
}
```

Nombre de pila	Apellido
Pedro	Grifo
Lois	Grifo

Mesa de ancho completo

La tabla anterior puede parecer pequeña en algunos casos. Si necesitas una tabla que abarque toda la pantalla (ancho completo), puedes hacerla con **width: 100%** al elemento <table>:

Nombre de pila	Apellido
Pedro	Grifo
Lois	Grifo

```css
table {
  width: 100%;
}
```

Alineación horizontal

La propiedad **text-align** establece la alineación horizontal (como izquierda, derecha o centro) del contenido en <th> o <td>.

De forma predeterminada, el contenido de los elementos **<th>** está alineado al centro y el contenido de los elementos **<td>** está alineado a la izquierda.

Para alinear al centro el contenido de los elementos <td> también, puedes usar **text-align: center**

Nombre de pila	Apellido	Ahorros
Pedro	Grifo	$100
Lois	Grifo	$150
José	Swanson	$300

```css
td {
  text-align: center;
}
```

Para alinear a la izquierda, deberías reemplazar *center* por *left*.

Formularios en CSS

Los formularios que creamos en HTML pueden cambiar mucho su estilo si les aplicamos CSS!

Intenta aplicar este CSS a un formulario que tengas creado, **¿Qué sucede?**

```css
input[type=text] {
  width: 100%;
  padding: 12px 20px;
```

```
  margin: 8px 0;
  box-sizing: border-box;
}
```

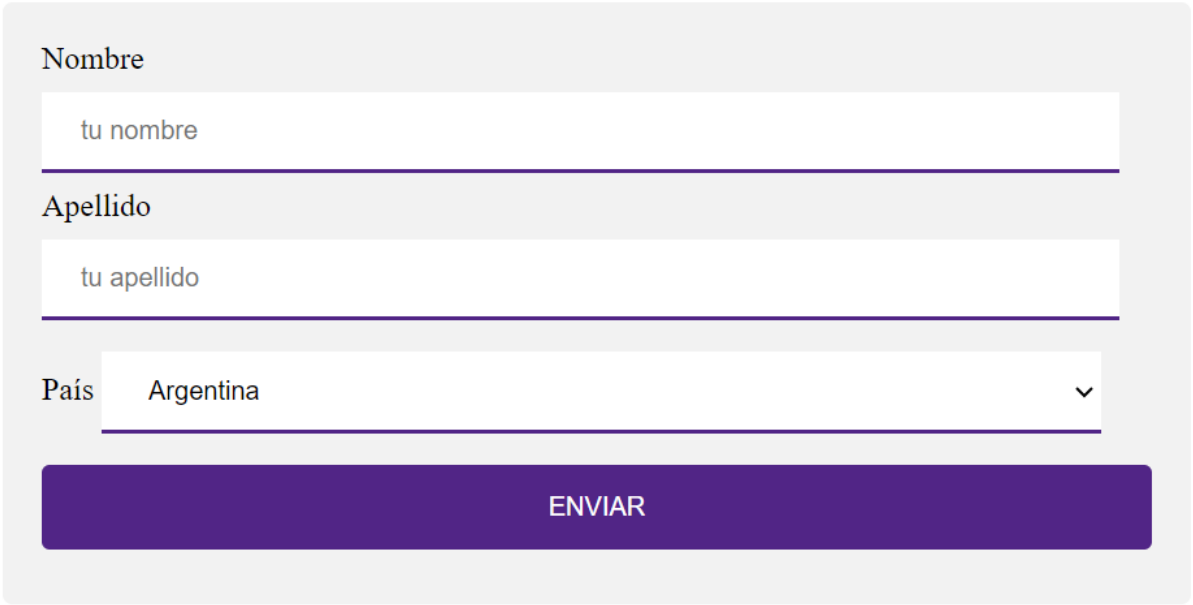
Las posibilidades son muchas y diversas, dentro de un formulario puedes modificar por ejemplo:

- el borde (border)
- color de fondo (background-color)
- el foco (con el pseudo selector :focus)
- transition (usar la propiedad de transición para agrandar el campo del input)
- estilizar el botón (con bordes redondeados, color de fondo etc.)

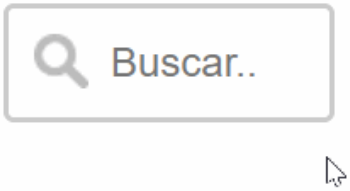
Veamos unos ejemplos:

- Si solo queremos resaltar el borde inferior del campo podemos utilizar por ejemplo, el siguiente código:

```
border: none;
border-bottom: 2px solid #512586;
```



- Si queremos realizar una animación en un campo, podemos utilizar por ejemplo, el siguiente código:



```
input[type=text] {
  width: 130px;
  box-sizing: border-box;
  border: 2px solid #ccc;
  border-radius: 4px;
  background-image: url('searchicon.png');
  background-position: 10px 10px;
  background-repeat: no-repeat;
  padding: 12px 20px 12px 40px;
  transition: width 0.4s ease-in-out;
}
```

```
input[type=text]:focus {
  width: 100%;
}
```

Así como estos puedes aplicar muchos otros estilos, te recomendamos que lo puedas probar y jugar con los diferentes estilos!

Diseño responsivo

El ***diseño responsivo*** (“responsive”) es la práctica de crear páginas web que se adaptan automáticamente a diferentes tamaños y orientaciones de pantalla, brindando una experiencia de visualización óptima en varios dispositivos (de escritorio, tablets o smartphones).

Para realizar un diseño responsivo se utilizan las "media queries", que son una técnica de CSS que permite aplicar estilos específicos a un documento HTML en función de las características del dispositivo que lo está visualizando, como el tamaño de la pantalla, la resolución, la orientación, entre otros. De esta manera, es posible crear diseños web adaptables y optimizados para diferentes dispositivos.

Las ***media queries*** se definen usando la sintaxis **"@media"**, seguido de un conjunto de reglas CSS.

```
@media (regla-de-media-queries: valor) {
  reglas css
}
```

A continuación, te compartimos todas las opciones de uso posible de las ***media queries***:

- **min-width:** Se aplica cuando la anchura de la pantalla es mayor o igual a un valor específico.
- **max-width:** Se aplica cuando la anchura de la pantalla es menor o igual a un valor específico.
- **min-height:** Se aplica cuando la altura de la pantalla es mayor o igual a un valor específico.
- **max-height:** Se aplica cuando la altura de la pantalla es menor o igual a un valor específico.
- **orientation:** Se aplica según la orientación del dispositivo, ya sea portrait (vertical) o landscape (horizontal).
- **aspect-ratio:** Se aplica según la relación entre el ancho y la altura de la pantalla.
- **device-aspect-ratio:** Se aplica según la relación entre el ancho y la altura del dispositivo.
- **resolution:** Se aplica según la resolución de la pantalla en píxeles por pulgada (PPI) o en puntos por pulgada (DPI).
- **min-resolution:** Se aplica cuando la resolución de la pantalla es mayor o igual a un valor específico.
- **max-resolution:** Se aplica cuando la resolución de la pantalla es menor o igual a un valor específico.

- **pointer:** Se aplica según el tipo de dispositivo de entrada que se esté utilizando, como none (para pantallas táctiles) o coarse (para dispositivos con puntero menos preciso).
- **hover:** Se aplica según la presencia o no de un dispositivo de puntero que permita la interacción a través de hover.
- **any-pointer:** Se aplica si el dispositivo tiene algún tipo de dispositivo de entrada de puntero.
- **any-hover:** Se aplica si el dispositivo tiene algún tipo de dispositivo de puntero que permita la interacción a través de hover.

En el siguiente video te ayudamos a comprender mejor **cómo funcionan las medias queries más utilizadas:**

👉 **Ver video**

Flexbox y Grid

Flexbox y ***Grid*** son *módulos modernos de diseño CSS* que ofrecen opciones poderosas y flexibles para crear diseños responsivos.

💡 *Pssst, Recuerda que uedes usar el siguiente "prompt" en el **chat GPT**: "Explicar de manera simple y concisa, y que sea fácil de entender: ¿Qué son los módulos modernos de diseño CSS?".*

Flexbox está diseñado para diseños unidimensionales, lo que significa que *se utiliza para crear diseños en una sola dirección, ya sea en filas o columnas*. Con Flexbox, podemos controlar la alineación, el tamaño y el orden de los elementos en su diseño. Es especialmente útil para diseñar elementos como menús de navegación, listas de elementos, elementos de una página de perfil de usuario y formularios.

Por otro lado, **Grid está diseñado para diseños bidimensionales**, lo que significa que *se utiliza para crear diseños en dos direcciones, tanto filas como columnas.* Con Grid, podemos crear diseños complejos y precisos que permiten la colocación de elementos en cualquier lugar dentro del contenedor de diseño. Es especialmente útil para diseñar elementos como diseños de página, diseños de tarjetas y diseños de revistas.

Ambos módulos *tienen algunas características en común* como:

- Capacidad de controlar el espaciado entre los elementos.
- Capacidad de hacer que los elementos cambien de tamaño y posición automáticamente en función del tamaño de la pantalla del usuario.
- Capacidad de controlar el flujo del contenido.

También son **compatibles con la técnica de diseño responsivo**, lo que significa que se pueden ajustar automáticamente para adaptarse a diferentes tamaños de pantalla y dispositivos.

💡 ***Son dos herramientas útiles para el diseño web y ofrecen una amplia variedad de opciones para crear diseños flexibles y responsivos. La elección de cuál usar dependerá de la naturaleza del proyecto y del tipo de diseño que se esté buscando.***

Ahora, te invitamos a abrir el enlace del material complementario y a leer acerca de las **propiedades de Flexbox** y sus posibles valores:

👉 **Propiedades de Flexbox**

¿Demasiada información? ¡No te preocupes! Vas a poder ver el siguiente video el cual te ayudará a visualizar y comprender mejor cómo funciona Flexbox:

👉 **Ver video**

¡Muy bien! Ahora continuemos con las propiedades de Grid y sus posibles valores:

👉 **Propiedades de Grid**

💡 ***Para seguir profundizando sobre el tema pueden recurrir a la*** 👉**teoría de Flexbox** ***y*** 👉**teoría de Grid**

CSS Anexo

Comentarios

Los comentarios se utilizan para explicar ciertas partes del código y pueden ayudar cuando alguien más mira el código o para futuras referencias.

💡 **tip: Los navegadores ignoran los comentarios.**

Sintaxis del comentario

Comienza con **/*** y termina con ***/**

Ejemplo**:**

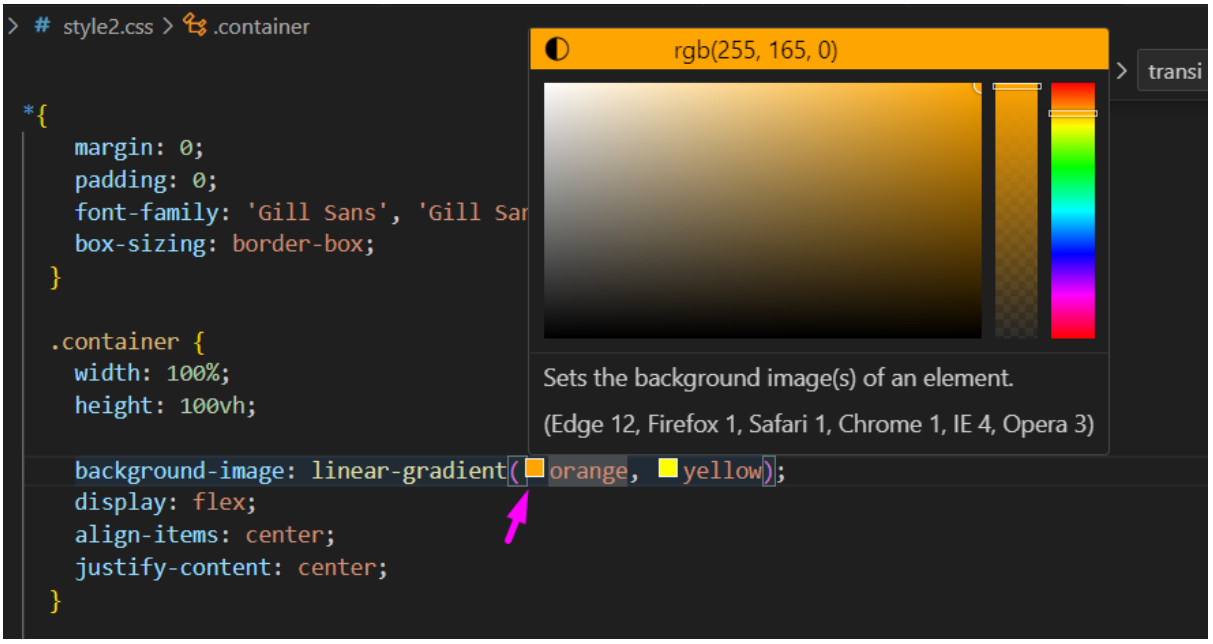
```
/* This is a single-line comment */
p {
  color: red;
}
```

Colores

En CSS los colores se pueden escribir de diferentes formas

- por el nombre en inglés (red, green, blue)
- rgb(255, 99, 71)
- hex(#ff6347)

Al utilizar Visual Studio Code, cuando escribirnos un color en CSS coloca un recuadro al lado que al hacer click se despliega una paleta para seleccionar



Si queremos cambiar el color de un texto (por poner un ejemplo) lo podremos hacer de la siguiente manera:

```
p {
color: blue;
color: rgb(51, 0, 255);
color: #0000ff
}
```

💡**Recuerda seguir investigando las propiedades de color y todo lo que puedes hacer con ellas!**

Transiciones

Las transiciones en CSS se utilizan para mejorar la experiencia del usuario al proporcionar retroalimentación visual a la interacción del usuario.

Al interactuar con un sitio web, es posible que notes que muchos elementos tienen estados. Por ejemplo:

- Los menús desplegables pueden estar en estados “abierto” o “cerrado”.
- Los botones pueden cambiar de color al recibir foco o al pasar el cursor sobre ellos.
- Los modales aparecen y desaparecen.

Por defecto, CSS cambia el estilo de estos estados de manera instantánea.

Utilizando transiciones CSS, podemos interpolar entre el estado inicial y el estado objetivo del elemento.

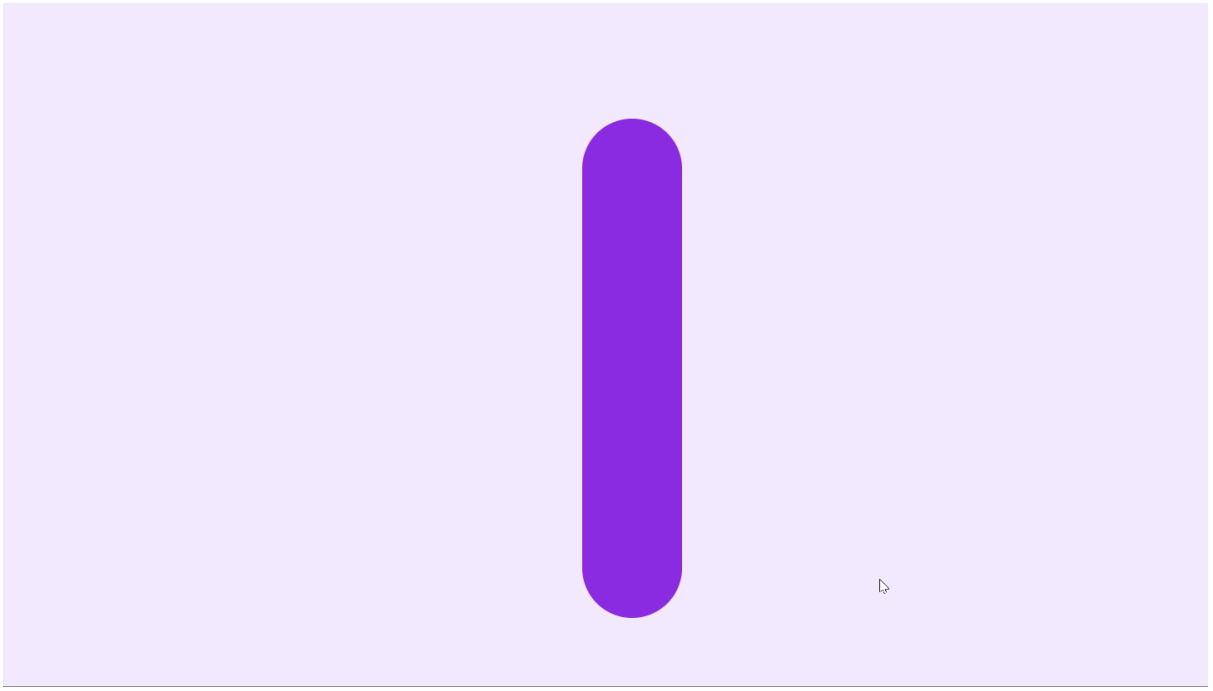
💡 ***Interpolación:*** **Es el proceso de crear pasos intermedios con una transición para pasar de un estado a otro**

Para aplicar una transición necesitamos explicitar dos cosas:

- La propiedad CSS a la que se desea aplicar el efecto.
- La duración del efecto.

Es importante tener en cuenta que si no se especifica la duración, la transición no se aplicará, ya que el valor predeterminado es 0.

Vamos a ver un ejemplo:



El código para este ejemplo es el siguiente:

Formato de elemento

```
.img-box{
  width: 100px;
  height: 500px;
  margin: 10px;
  border-radius: 50px;
  background-color: blueviolet;
  position: relative;
  transition: width 0.5s;
}
```

*La propiedad CSS a la que vamos a aplicar el efecto es **"width",** la duración **"0.5s"**

Cambio de ancho

```
.img-box:hover{
    width: 300px;
    cursor: pointer;
  }
```

El efecto de transición comenzará cuando la propiedad CSS especificada (ancho) cambie de valor.

💡***Para seguir profundizando sobre el tema de transiciones pueden recurrir a la*** 👉teoría de W3 Schools

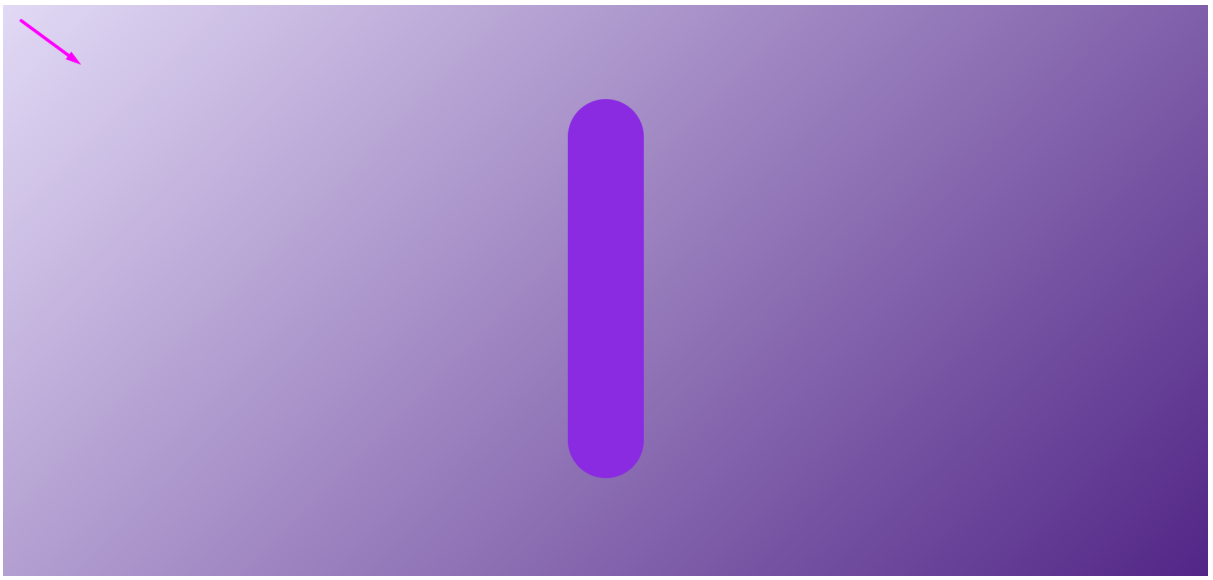
Degradados o Gradientes

Los degradados CSS le permiten mostrar transiciones suaves entre dos o más colores específicos.

En CSS tenemos 3 tipos de degradados:

- lineales (baja/arriba/izquierda/derecha/diagonalmente)
- radiales (definidos por su centro)
- cónicos (girados alrededor de un punto central)

Vamos a ver un ejemplo:



El código para este ejemplo es el siguiente:

```
.container{
   background-image: linear-gradient(135deg, #e3dbf6, #512586);
}
```

Degradado Lineal

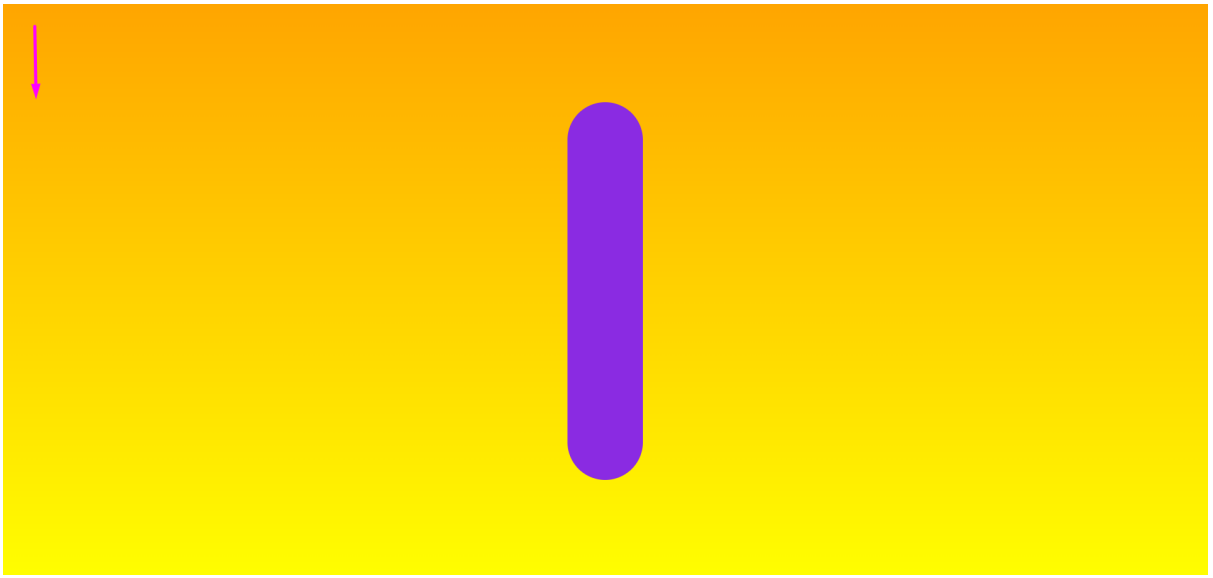
Sintaxis

La sintaxis para escribir un degradado lineal es la siguiente:

```
background-image: linear-gradient(dirección, color1, color2, ...);
```

Si solo colocamos los colores, el degradado se hará de arriba hacia abajo por default . Por ejemplo:

```
#degradado {
  background-image: linear-gradient(orange, yellow);
}
```



Si queremos indicar una dirección específica, podemos hacerlo con:

- los grados (180deg, ...)
- con las palabras reservadas (to right - to bottom right, ...)

¡Prueba diferentes formas y genera tus propios degradados!

💡***Para seguir profundizando sobre el tema de transiciones pueden recurrir a la*** 👉teoría de W3 Schools