Bootcamp Full Stack

CSS avanzado

¿Qué es CSS?
Repasando conceptos

CSS

Sus siglas significan *Cascading Style Sheet.*

Este lenguaje **permite aplicar estilo o definir cómo se van a mostrar los elementos** que previamente fueron estructurados en **HTML**.

Con HTML, se puede indicar que un elemento es, por ejemplo, un enunciado. Luego, con CSS se determinará cómo será su apariencia, qué tipografía tendrá y cuál será su tamaño.

Este lenguaje se vale de **reglas de estilo** para poder trabajar. Estas reglas conforman su sintaxis y lo convierten en un lenguaje complementario de **HTML** simple y **fácil de utilizar**.

¿Cómo es el lenguaje CSS?

Reglas de estilo

Como mencionamos, el lenguaje CSS está conformado por reglas de estilo.

Las reglas de estilo se conforman a partir de la siguiente sintaxis.

```
selector {propiedades: valor;}
```

Es muy importante **no olvidar el “;”** luego de cada valor de propiedad.

Reglas de estilo

Lo que vemos al comienzo de la regla de estilo se denomina **“selector” e identifica aquello que afectamos con el css.**

El selector puede ser de diferentes tipos según su alcance y características (id, class, selector de etiqueta, etc.)

Lo que vemos entre las llaves se denomina “declaración de estilo” y está compuesta por la propiedad o propiedades que según su valor específico afectan al selector en sus características por ejemplo dependiendo la propiedad por ejemplo en su color tipográfico, color de fondo, width, height, entre otros.

Este es un ejemplo práctico para comenzar a comprender mejor este universo: mostramos un párrafo **(selector)**, modificado en su color de fondo **(propiedad,** en este caso background-color) y con un valor azul (valor de la propiedad):

```
p { background-color: blue; }
```

¿Cómo implementar CSS en un HTML?

Existen tres maneras:

En línea

Esta forma de trabajo se hace dentro del mismo elemento de **HTML** que se está modificando.

Como se ve en el siguiente ejemplo, se cambia el color de fondo de **un párrafo usando simplemente el atributo style:**

```
<p style="background-color: blue;"> </p>
```

Ejemplo de uso de CSS en línea

Un modo rápido de descubrir una implementación de css en línea es revisando nuestra casilla de correo.

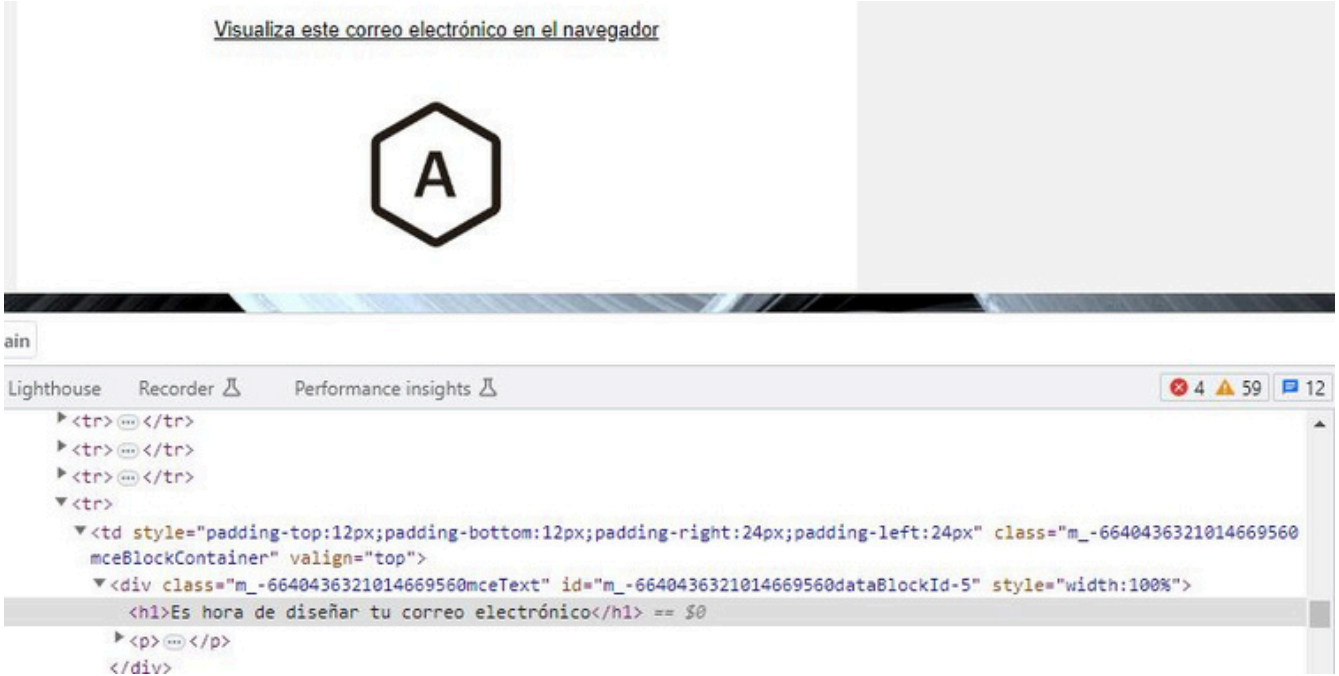
Por ejemplo, cuando recibimos publicidad o anuncios a perfiles a los que nos suscribimos, recibiremos los famosos *newsletters.* Estos elementos no son más que html que, al estar interpretados directamente por el servidor de correo electrónico (Gmail, Outlook, etc.), fueron trabajados con sugerencias y recomendaciones

completamente distintas a las que tomamos en cuenta al momento de trabajar por ejemplo con una interfaz Web.

Si inspeccionamos la imagen anterior en nuestro navegador (en este caso, desde una cuenta de Gmail en Google Chrome), veremos que el *newsletter* se conforma de filas y celdas provenientes del elemento table.

Es decir, a diferencia de una interfaz Web, donde trabajamos con contenedores, en este caso la maquetación se realiza a través de este elemento tabular, que contiene estilos en línea para poder lograr la visualidad anterior,

Ejemplo:


Visualiza este correo electrónico en el navegador
A
ain
Lighthouse
Recorder
Performance insights
4 59 12
<tr> … </tr>
<tr> … </tr>
<tr> … </tr>
<tr>
<td style="padding-top:12px;padding-bottom:12px;padding-right:24px;padding-left:24px" class="m_-6640436321014669560 mceBlockContainer" valign="top">

<h1>Es hora de diseñar tu correo electrónico</h1> == $0
 …

De forma interna

Esta forma de trabajo se hace dentro del mismo **HTML** que se está modificando.

Por ejemplo: es posible cambiar el color de fondo de un **párrafo con una regla de estilo, dentro del head del documento**.

Debemos anidar la regla de estilo en el elemento style, como se observa en la pantalla a continuación.

Ejemplo de trabajo estilos internos con CSS:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Curso de maquetación</title>
<style>

p { background-color: blue;}

</style>


</head>
```

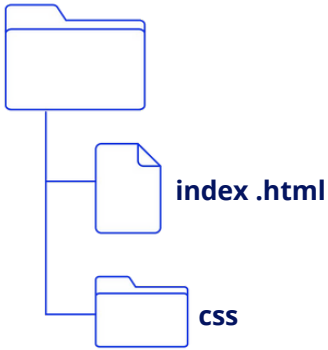
De forma externa

Esta forma de trabajo es la más recomendable, ya que con un sólo archivo **.css** se pueden modificar varios archivos **.html** al mismo tiempo.

Para poder iniciar este proceso, vamos a generar un archivo.css nuevo. Este se podrá generar desde el **propio editor de texto, como Visual Studio Code o Sublime Text**.

En este caso hemos elegido nombrar a nuestro archivo ***estilos.css.***

Para trabajar siempre es indicado crear carpetas, por eso hemos creado la carpeta ***css*** dentro de una carpeta principal donde ya se encuentra un archivo llamado ***index.html***.



Para continuar se debe **vincular el archivo** que se encuentra en la carpeta ***css,*** con el ***index.html***.

Para **vincular *estilos.css* con *index.html*** debemos generar un elemento nuevo en la estructura del html:

```
<link rel="stylesheet" href="css/estilos.css">
```

Este elemento llamado **link**, posee en su atributo **href** la ruta donde está guardado el archivo css vinculado.

Si se desea modificar otro html, sólo se debe colocar este mismo link.

Es importante no olvidarse que el link debe encontrarse anidado en el **head**, como veremos en la siguiente imagen:

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Curso de maquetación</title>
<link rel="stylesheet" href="css/estilos.css">
</head>
```

Primeras propiedades

Selectores iniciales

Selector de etiqueta

El selector de etiqueta está compuesto por el elemento de html que afectamos.

Si bien es fácil de implementar y se lo utiliza muchísimo, guarda una desventaja que es que **afecta a todos los elementos de su tipo:**

Selector compuesto descendente

El selector descendente permite **afectar a un elemento por encontrarse anidado en otro,** por ejemplo:

```
<header>
<h1>Gimnasio <span>vuelta de tuerca</span></h1>

                                    </header>
```

Para poder afectar sólo a ese **h1** y no a otro podemos:

```
header h1 { font-family: 'Poppins', sans-serif; font-weight: 300; font-size: 2em; }
```

Si queremos generar una lista más extensa, se pueden agregar cuanto elemento anidado tengamos:

```
header h1 span {font-family: miFuente; font-style: italic; font-weight:bold; }
```

A diferencia del selector anterior, el selector hijo no permite que haya elementos interpuestos entre el padre y el hijo.

En el siguiente html:

```
<header><span>Este es</span>
    <h1>Mi <span>enunciado </span></h1>
</header>
```

El selector descendente del ejemplo debajo convertiría en rojo ambos span:

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

Sin embargo, el siguiente selector hijo afectaría sólo al primero, dado que el otro se encuentra a su vez anidado en un h1, es decir,hay un elemento interpuesto entre ambos:

```
header > span { color: red;}
```

Selector compuesto adyacente

También podemos **afectar a un elemento por encontrarse luego de otro.**

```
<h2>Amor al deporte</h2>
<p> Lorem, ipsum dolor sit amet consectetur adipisicing elit.
```

En este caso si queremos **afectar sólo a aquellos párrafos que están luego de un enunciado h2** haremos lo siguiente:

```
h2 + p { color: black;}
```

Selector grupal

También podemos **afectar a varios elementos dentro de un grupo.**

```
h1,p, article { color: black;}
```

De esta manera en vez de escribir tres veces la misma regla, sabemos que el color negro afectará a tantos elementos se encuentren dentro del gurpo.

Propiedades de texto

Propiedades de texto

text-decoration

La propiedad **text-decoration** me permite trabajar con la **decoración de un texto.**

Como valores posibles podemos encontrar:

- `underline`
- `overline`
- `line-through`
- `none`

Existen otros valores que se pueden sumar a text-decoration, sugerimos revisar el material del curso de Desarrollo Web con HTML.

Esta propiedad es **muy útil cuando queremos quitarle a los vínculos su usual subrayado.** Por ejemplo:

text-transform

La propiedad **text-transform** me permite trabajar **convirtiendo un texto a mayúsculas o minúsculas.**

Como valores posibles podemos encontrar:

- `uppercase`
- `lowercase`
- `capitalize`
- `none`

Esta propiedad es muy útil cuando queremos trabajar en textos que desde el HTML han sido copiados o pegados sin ningún tipo de regla con referencia a mantener mayúsculas o minúsculas. Por ejemplo:

```
h1 { text-transform: uppercase;}
```

text-align

La propiedad **text-align** me permite **alinear el texto.**

Como valores posibles podemos encontrar:

- `center`
- `right`
- `left`
- justify

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

Existen otros valores de text-align , sugerimos revisar el material del curso de Desarrollo Web con HTML.

font-variant

La propiedad **font-variant** me permite darle **estilo al texto con versalitas.**

Como valores posibles podemos encontrar:

- `small-caps`
- none

```
p { font-variant: small-caps;}
```

font-weight

La propiedad **font-weight** me permite **darle peso a la tipografía.** Es importante entender que si utilizamos esta propiedad con sus posibles valores, también **el recurso real de tipografía diseñada para tal fin debe estar disponible,** sino el navegador puede engrosar la fuente pero no será fiel al diseño original de la misma.

Como valores posibles podemos encontrar:

- `bold`
- `normal`
- `bolder`
- `lighter`
- valores del 100 al 900

El ejemplo debajo muestra cómo darle peso ***bold*** a un párrafo:

```
p { font-weight:bold;}
```

font-style

La propiedad **font-style** me permite **darle estilo a la tipografía.** Es importante entender que si utilizamos esta propiedad con sus posibles valores, también **el recurso real de tipografía diseñada para tal fin debe estar disponible,** sino el navegador puede inclinar la fuente, pero no será fiel al diseño original de la misma.

Como valores posibles podemos encontrar:

- `normal`
- `italic`
- `oblique`

El ejemplo debajo muestra cómo usar ***itálicas*** en un párrafo:

```
p { font-style:italic;}
```

Selectores id y class

Selectores id

El selector de **id** permite modificar a un elemento, en el html, que contenga como valor de atributo id el mismo nombre:

```
<header id="encabezado">
```

De esta forma luego en nuestro *archivo.css* el elemento se genera de la siguiente manera:

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

La característica del **selector id es que solo puede utilizarse una vez por HTML.**

También es importante tener en cuenta sus **reglas de nomenclatura:**

- No puede comenzar con un número.
- Es case sensitive.
- No pueden haber elementos reservados del lenguaje (puntos, puntos y coma, etc).
- No se puede dejar espacios.

Selectores class

El selector de **class** permite modificar a un elemento que contenga como valor de atributo class, en el html, el mismo nombre:

```
<header class="encabezado">
```

De esta forma luego en nuestro *archivo.css* el elemento se genera de la siguiente manera:

```
.encabezado { color: blue; }
```

La característica del selector **class es que puede utilizarse varias veces en el mismo HTML.**

Las **reglas de nomenclatura** son similares a las del elemento id:

- No puede comenzar con un número.
- Es case sensitive.
- No pueden haber elementos reservados del lenguaje (puntos, puntos y coma, etc).
- No se puede dejar espacios.

Compuestos con class, id, y selectores de etiqueta

Luego de ver el trabajo con id y class, podemos implementar otros selectores ya vistos, por ejemplo, selectores grupales:

```
.encabezado, div, #caja { color: blue; }
```

O selectores descendentes:

```
header div { width: 50%;  background-color: rgba(255,255,255,0.5);}
```

Colores en la Web

Propiedades de color

`background-color`

Es una propiedad que **permite modificar el color de fondo de los elementos.**

Los valores de esta propiedad nos sirven para comprender de qué forma se trabaja con el color en la web.

Nuestro **modelo rgb** permite diferentes variantes:

- `rgb()`
- `#valoresHexadecimales`
- `hsl()`
- `rgba()`
- `hsla()`
- `palabras`

color

Es una **propiedad que permite modificar el color de la tipografía de nuestro documento**.

Sus valores nos sirven para comprender de qué forma se trabaja con el color en la web.
Vamos a profundizar:

- **rgb(): Con este modelo, se puede indicar la cantidad de intensidad de luz de cada canal.**

```
body { font-size: 30px; color: rgb(0,0,0)}
body { font-size: 30px; color: rgb(255,0,0)}
body { font-size: 30px; color: rgb(0,255,0)}
body { font-size: 30px; color: rgb(0,0,255)}
body { font-size: 30px; color: rgb(255,255,255)}
```

- **rgba():** Es el mismo modelo anterior pero sumamos **el canal *alpha,* que es la transparencia**. Este va del 0 a 1, siendo 0 transparencia plena y 1 color pleno:

```
body { font-size: 30px; color: rgba(0,0,0,0.5)}
```

- **hsl():** Este modelo de color hace referencia al **matiz, la saturación (grises) y la luminosidad (blanco o negro).**

- **hsla():** es igual al modelo anterior pero suma el **canal *alpha* como se hizo con rgba().**

```
body { font-size: 30px; color: hsl(200,50%,30%);}
body { font-size: 30px; color: hsla(100,50%,80%,0.5);}
body { font-size: 30px; color: hsla(50,50%,50%,0.5);}
```

- **Palabras:** podemos trabajar con **palabras que representan colores**.

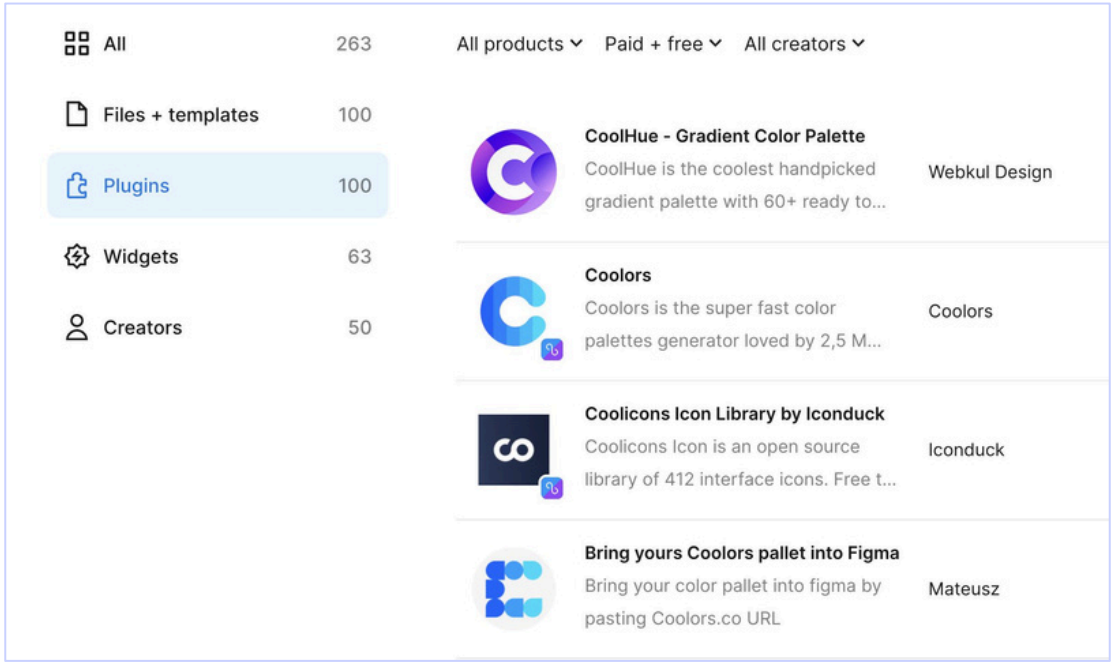
No existe, exactamente, la misma cantidad de palabras que de colores pero sí se puede trabajar con un gran número de valores posibles.

- **Valores hexadecimales:** se llaman de esa forma porque tienen de base 16.

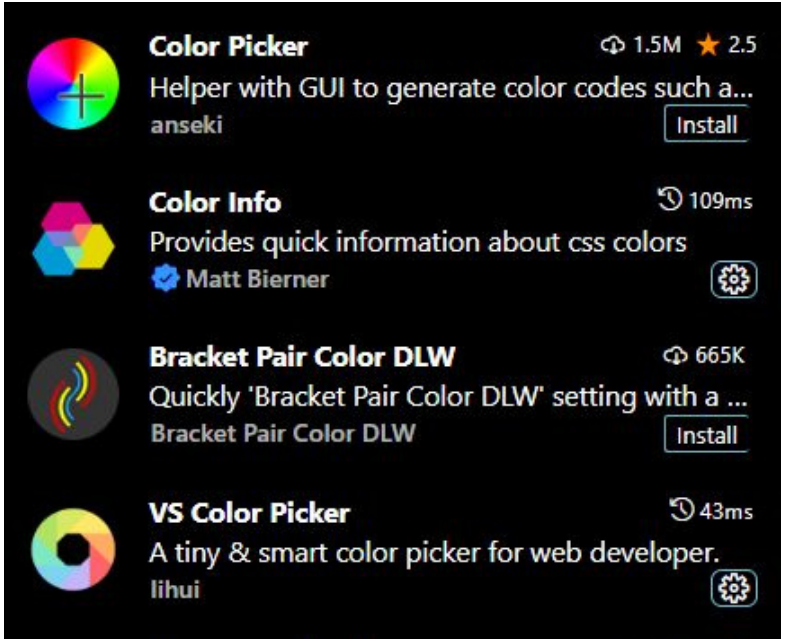
```
body { font-size: 30px; background-color: #333333;}
```

Este valor se puede conocer a través de una aplicación como Color Picker o trabajando con goteros como los que poseen programas de diseño tales como Adobe Photoshop o Adobe Illustrator.

En **Figma**, también existe una serie de ***plugins*** que nos permiten trabajar con colores, por ejemplo:



Extensiones sugeridas para trabajar con color desde **Visual Studio:**



Consideraciones

También recordemos que estos modos de color se pueden implementar en cualquier contexto donde haya color, esto es: un borde, un gradiente de fondo, un color tipográfico, o una sombra, entre otros.

Por lo tanto, lo aprendido hasta aquí será útil cuando se trabaje con propiedades nuevas en módulos siguientes.

Revisión

- Repasar los conceptos básicos del lenguaje.
- Descargar un editor de su preferencia **(Visual Studio Code o Sublime Text)**.
- Realizar un archivo .css y vincularlo con un archivo .html.
- Implementar las propiedades aprendidas.
- Repasar los conceptos vistos de `display:flex`.

- Para saber más sobre **flex**, copiar y pegar en la barra de dirección de tu navegador: css-tricks.com/snippets/css/a-guide-to-flexbox
- Ver todos los videos y materiales necesarios antes de continuar.