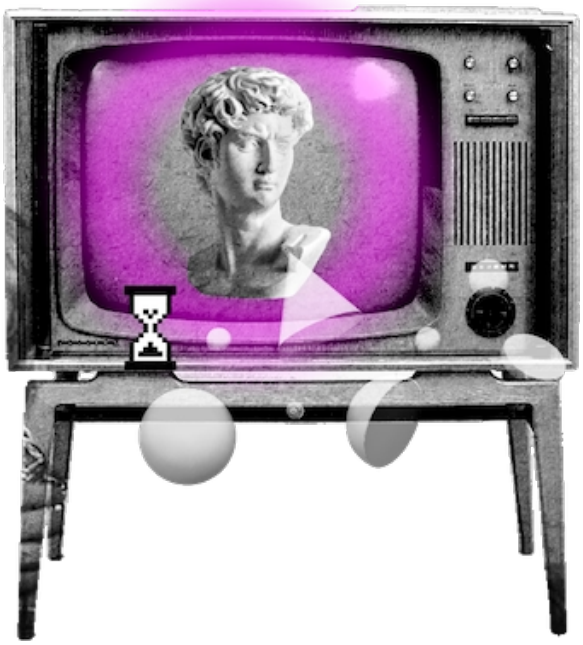
NUCBA

DISEÑO WEB

CSS 101



APUNTES

CSS - Sintaxis, Selectores y Propiedades Básicas

¿Qué es CSS?

CSS es un acrónimo que significa **Hojas de Estilo en Cascada (Cascading Style Sheets)**. Es un lenguaje que usamos para **describir cómo queremos que se vean los documentos HTML en una página web**. Con CSS, podemos **especificar aspectos visuales** como los colores, los tipos de letra, los tamaños y las posiciones de los elementos en una página.

Además, si bien se puede trabajar con CSS en los archivos html , permite separar la **estructura y el contenido de la página de su apariencia visual**. Esto significa que podemos tener un archivo HTML que describe la estructura de la página y un archivo CSS que se encarga de su presentación. Esto nos da un mayor control y flexibilidad sobre cómo queremos que se vea nuestro sitio web.

En resumen, CSS es un lenguaje que nos permite **mejorar la apariencia y el estilo de nuestro sitio web**, y es un componente clave en el desarrollo de páginas web modernas y atractivas.

Historia de CSS

CSS fue creado en el año **1996** por el **World Wide Web Consortium (W3C)**, con el objetivo de **mejorar la presentación de páginas web y permitir un mayor control y flexibilidad sobre el diseño de sitios web.**

Esta tecnología ha evolucionado a lo largo de los años, y actualmente se encuentra en su **tercera versión**, conocida como **CSS3**. Sin embargo, a lo largo de su historia, se han publicado varias versiones importantes de CSS:

1. **CSS1**: La primera versión de CSS, publicada en 1996, introdujo las funciones básicas para describir la presentación de páginas web. Incluía reglas para estilos de texto, colores, bordes y otros aspectos visuales.
2. **CSS2**: Publicada en 1998, la segunda versión de CSS agregó nuevas características y mejoró las existentes, incluyendo soporte para dispositivos de pantalla, mejoras en la gestión de la información de estilo y más.
3. **CSS3**: La tercera versión de CSS, publicada en su primera versión base en 1999, ha sido desarrollada y ampliada a lo largo de los años, y es la versión actual de CSS. CSS3 incluye una gran cantidad de nuevas funciones y mejoras, como animaciones, efectos de sombra, transformaciones y más, y es la versión más ampliamente utilizada en la actualidad. Es importante tener en cuenta que se trata de una versión que aún sigue en desarrollo y se continúan agregando nuevas características.

¿Por qué utilizar CSS?

Existen varios motivos por los cuales CSS es vital a la hora de incursionar en el desarrollo web. Entre ellos podemos encontrar:

1. **Mejora la apariencia**: Con CSS, podemos controlar el estilo y la presentación de un sitio web de manera eficiente, lo que significa que podemos mejorar su aspecto y hacerlo más atractivo para los usuarios.

2. **Separación de contenido y estilo**: Al separar el contenido HTML del estilo CSS, podemos mantener el código más organizado y fácil de mantener. Esto también significa que podemos aplicar estilos a varias páginas web a la vez sin tener que repetir el mismo código.

3. **Adaptabilidad a diferentes dispositivos**: Con CSS, podemos crear estilos que se ajusten a diferentes tipos de dispositivos, como ordenadores, tabletas y teléfonos móviles. Esto significa que los usuarios pueden tener una experiencia óptima en cualquier dispositivo.

4. **Accesibilidad:** CSS permite crear estilos accesibles para personas con discapacidades, lo que significa que los usuarios con discapacidades pueden acceder y navegar por el sitio web de manera más fácil.

Formas de aplicar CSS

Existen 3 maneras de aplicar CSS a nuestros archivos HTML: **Agregando estilos en línea, Usando la etiqueta style dentro de la etiqueta <head> y utilizando un archivo externo.**

Estilos en Línea

Podemos agregar estilos **directamente a una etiqueta HTML específica** utilizando el atributo **“style”**. Por ejemplo:

```
<p style="color: blue;">Este es un párrafo con estilo en línea.</p>
```

Párrafo con un único estilo agregado.

```
<p style="color: blue; font-size: 18px; text-align:
center;">Este es un párrafo con varios estilos en línea.</p>
```

Párrafo con varios estilos agregados.

Estilos en el encabezado

Podemos agregar estilos a todas las páginas de un sitio web agregando **una sección de estilos en el encabezado de cada página**. Esto se hace utilizando la etiqueta **<style>** en el encabezado de la página. Por ejemplo:

```
<head>
  <style>
    p {
      color: blue;
    }
  </style>
</head>
```

Archivo externo

Podemos mantener los estilos en un **archivo CSS externo** y vincularlo a nuestras páginas HTML utilizando la etiqueta **<link>** en el encabezado de cada página. Por ejemplo:

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

En la mayoría de los casos, es recomendable utilizar un archivo CSS externo para mantener nuestros estilos organizados y fácilmente mantenibles.

Sintaxis CSS

La sintaxis de CSS se compone de tres partes principales: **selectores, propiedades y valores.**

1. **Selectores**: Determinan a qué elementos HTML se aplicarán los estilos. Por ejemplo, un selector puede ser una etiqueta HTML específica como <**p**> o una **clase** o **ID** personalizados que hayas definido en tu HTML.

2. **Propiedades**: Definen los aspectos específicos que se quieren estilizar, como el color del texto o el espacio alrededor de un elemento.

3. **Valores** Especifican cómo se quiere que se vea esa propiedad. Por ejemplo, el color del texto puede ser "rojo" o "#ff0000".

Ejemplo básico de Sintaxis

```
p{
    color: red;
    text-align: center;
    font-size: 20px;
}

.titulo {
    color: blue;
    text-align: left;
    font-size: 30px;
}

#subtitulo{
    color: green;
    text-align: right;
    font-size: 25px;
}
```

En este ejemplo podemos ver e identificar los distintos selectores , valores y propiedades.

Entre los **selectores** podemos ver el selector de etiqueta , con la etiqueta **<p>** , el selector de clase , con la clase "**.titulo"** y el selector de id , con el id "#**subtitulo"**

Por otro lado, en cuanto a propiedades podemos identificar 3: **color, text-align** y **font-size**.

Finalmente, tenemos los valores. Por un lado, valores de color como **green, blue y red** , valores para la alineación del texto , como **center , left** y **right** , y finalmente valores de tamaño de letra, como **20px , 30px** y **25px**.

Además podemos ver que los selectores se colocan antes de la llave de apertura que envuelve los estilos que se van a declarar , siendo cada combinación de propiedad y valor una **declaración.** Finalmente, la sintaxis concluye con la llave de cierre que envuelve las declaraciones de estilos

Todas estas propiedades, selectores y varios de los valores los vamos a ir viendo a lo largo de la cursada, pero es importante que ustedes como alumnos comiencen a experimentar con las distintas propiedades y valores a fines de conocerlas y aprender a dominarlas.

Selectores de CSS

Un selector de CSS es un patrón que se utiliza para seleccionar elementos HTML específicos en un documento y aplicar estilos a ellos.

En esta sección, vamos a estar viendo los principales selectores de CSS.

Selector universal

Sirve para agregar estilos a **todos** los elementos de una página web. Se suele utilizar para hacer un "**reset"** de estilos.
Para su sintaxis utilizamos el asterisco **" * "**

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

En este caso de ejemplo, todos los elementos del html tendrán como color de letra el color rojo.

Selector de etiqueta

Esta etiqueta sirve para aplicarle estilo a todos los elementos del tipo especificado, cualquiera sea la etiqueta elegida.

```
p {
  color: red;
}

h1 {
  color: blue;
}

div {
  color: green;
}
```

En estos casos de ejemplo, le estamos cambiando el color de letra a los elementos **<p> , <h1> y ** de nuestro documento HTML.

Selector de clase

Una **clase** en HTML es una etiqueta que se utiliza para **identificar un elemento o grupo de elementos con un nombre específico**. Esto permite a los desarrolladores de aplicaciones web **aplicar estilos específicos** a esos elementos a través de CSS.

Cada clase tiene un **nombre único** para identificar a los elementos que se desea aplicar un estilo en particular. Además las clases se pueden **reutilizar en diferentes elementos** de nuestra página.

```
<header class="header">
    <nav class="navbar">
        <ul class="navlist">
            <li class="navbar-item"><a class="navbar-link" href="#">Home</a></li>
            <li class="navbar-item"><a class="navbar-link" href="#">About</a></li>
            <li class="navbar-item"><a class="navbar-link" href="#">Contact</a></li>
        </ul>
    </nav>
</header>
```

En el ejemplo de arriba, hemos aplicado clases a todos los elementos de una **barra de navegación**.

Podemos, en base a eso, dejar en claro algunas pautas de uso de las clases:

1. **Utilizar nombres descriptivos:** El nombre de una clase debe ser claro y descriptivo, evitando nombres genéricos como **“clase1”** o **“clase2”**. El objetivo de esto es poder indicar claramente el propósito de una clase y a qué elemento hace referencia de nuestra página, facilitando así la comprensión del código.

2. **Mantenibilidad:** Las clases deben ser lo **suficientemente genéricas como para ser reutilizadas, pero también deben ser lo suficientemente específicas como para no causar problemas en la aplicación de estilos**. Por ejemplo, la clase navbar-link es lo suficientemente específica para saber qué se trata de un link del navbar y poder utilizarla para **todos los links del navbar**, pero no entra en detalle de cómo son esos links.

3. **Utilizar minúsculas y evitar usar tildes:** Se recomienda usar minúsculas para definir clases ya que es una **convención estándar en la industria** hacerlo, ya que ayuda a mantener la **consistencia y la legibilidad del código**. Además, ayuda a la mejor comprensión del código. Es recomendable utilizar mayúsculas únicamente si decidimos nombrar nuestras clases que tengan más de una palabra con el método **camelcase** (navbarLink , navbarItem , etc)

4. **No utilizar espacios para definir una clase individual:** Los espacios a la hora de nombrar clases son **caracteres inválidos**. Al colocar uno, el código interpretará que estamos poniendo dos clases diferentes. Por lo tanto, si tenemos que usar dos palabras para definir una única clase, utilizaremos **camelcase** (navbarLink) o **guiones medios o bajos** (navbar-link / navbar_link).

Utilizando múltiples clases en un solo elemento

Como vimos en el apartado anterior, es posible colocar más de una clase a un elemento HTML, y para eso, vamos a estar utilizando los espacios, siendo cada espacio la separación entre una clase y la otra.

```
<p class="parrafo parrafo-grande">Hola soy un párrafo con la letra muy grande</p>
```

Aplicando estilos a las clases en CSS

Para aplicar estilos a una clase, vamos a utilizar la siguiente sintaxis:

```
.parrafo{
  color: red;
  text-align: center;
}

.parrafo-grande{
  font-size: 30px;
}
```

Como podemos ver, colocamos un **“.”** delante de cada nombre de clase que queremos estilar y luego entre llaves colocamos nuestras declaraciones de estilos.

Selector de Id

Un **ID** en HTML es un **atributo** que se utiliza para **identificar de manera única** un elemento en una página HTML.

No puede haber más de un elemento con el mismo ID en una página HTML, por lo tanto, **no podemos repetirlos** , solo debemos usarlo en un único elemento.

```
    <section id="hero">
        <!-- Contenido del hero -->
    </section>
    <section id="contact">
        <!-- Contenido de contact -->
    </section>
    <section id="footer">
        <!-- Contenido del footer -->
    </section>
```

En el ejemplo podemos ver 3 secciones distintas de una página web, a las cuales les colocamos un identificador único y específico para cada una.

Aplicando estilos a los ID'S en CSS

Para aplicar estilos a un id especifico, vamos a utilizar la siguiente sintaxis:

```css
#hero{
  background-color: red;
}

#contact{
  background-color: blue;
}

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

Como podemos ver, colocamos un **"#"** delante de cada nombre de clase que queremos estilar y luego entre llaves colocamos nuestras declaraciones de estilos.

Selector de elementos descendientes

Un selector de elementos descendientes en CSS se utiliza para **seleccionar y aplicar estilos a elementos que están contenidos dentro de otro elemento en la estructura de la página HTML**. Se pueden seleccionar elementos descendientes utilizando la sintaxis de selector de espacio.

```html
    <section id="hero">

        <p class="parrafo">Hola soy un parrafo</p>
        <p class="parrafo">Hola soy un parrafo</p>
        <p class="parrafo">Hola soy un parrafo</p>

        <div>Soy un div con contenido</div>
        <div>Soy un div con contenido</div>
        <div>
            <p class="parrafo">Hola soy un parrafo</p>
        </div>

    </section>
```

```css
#hero .parrafo {
  /* Estilos de los párrafos */
}

#hero div {
  /* Estilos de los divs */
}
```

En este ejemplo , aplicamos estilos a **todos los divs y elementos con clase párrafo que están dentro de la sección hero, sin excepción.**

Selector de elementos hijo

Un **selector de elementos hijo** en CSS se utiliza para **seleccionar y aplicar estilos** a elementos que son **hijos directos** de otro elemento en la estructura de la página HTML. Se pueden seleccionar elementos hijo utilizando la sintaxis de selector de **mayor que (">")**. Por ejemplo:

```
    <section id="hero">

        <p class="parrafo">Hola soy un parrafo</p>
        <p class="parrafo">Hola soy un parrafo</p>
        <div>
            <p class="parrafo">Hola soy un parrafo</p>
        </div>

    </section>
```

```
#hero > .parrafo{
  /*....*/
}
```

En este caso, solo los párrafos que son **hijos directos** del elemento con id **hero** se verán afectados por los estilos que pongamos entre las llaves.

El párrafo que se encuentra dentro del **div** , si bien se encuentra en la seccion **hero**, no es hijo directo de dicho **hero**, es hijo directo del elemento **div**, por lo cual no se verá alcanzado por los estilos.

Selector de hermanos adyacentes

Un **selector de hermanos adyacentes** en CSS se utiliza para **seleccionar y aplicar estilos** a elementos que son **hermanos directos** en la estructura de la página HTML. Se pueden seleccionar elementos hermanos adyacentes utilizando la sintaxis de **selector de adyacencia (+)**.

Este selector se puede utilizar, por ejemplo, para darle funcionalidad al menú de barras que suelen tener las páginas web en su versión mobile y/o tablet

Cuando hacemos referencia a **hermanos directos** o **adyacentes** nos referimos a aquellos elementos que son **elementos hijo** del mismo **elemento padre** y que se encuentran colocados en la estructura del código de manera sucesiva.

Por otro lado, es importante destacar que solo se podrá seleccionar a un **hermano adyacente** que se encuentre con posterioridad dentro del código, y no uno que se encuentre antes del elemento mediante el cual seleccionaremos un **hermano directo.**

Veamos un ejemplo de cómo utilizar el selector de hermanos adyacentes:

```
    <section id="hero">
        <p class="parrafo">Hola soy el parrafo 1</p>
        <h1>Hola soy un título</h1>
        <p class="parrafo">Hola soy el parrafo 2</p>
        <p class="parrafo">Hola soy el parrafo 3</p>
        <p class="parrafo">Hola soy el parrafo 4</p>
    </section>
```

```
h1 + .parrafo {
  background-color: red;
}
```

En este caso, vemos que todos los párrafos y nuestro título son **elementos hermanos**, ya que son todos **hijos directos.**

Ahora bien, como mencionamos anteriormente, el selector de hermanos adyacentes **selecciona** elementos posteriores a, en este caso, nuestro **h1**, por lo cual los estilos no van a afectar al párrafo 1.

Por otro lado, mencionamos también que **selecciona** al hermano que se encuentra de **manera sucesiva** en el código, lo cual hace que los estilos no se apliquen al **párrafo 3** ni el párrafo 4.

Por ende, los estilos aplicarán al **párrafo 2** , ya que cumple con ambas características necesarias: Estar colocado luego de nuestro **h1** en el código y además está colocado de manera sucesiva.

Selector General de hermanos

Un **selector de hermanos generales** en CSS se utiliza para **seleccionar y aplicar estilos** a elementos que son **hermanos consecutivos** en la estructura de la página HTML. Se pueden seleccionar elementos hermanos consecutivos utilizando la sintaxis de virgulilla **(~)**.

```
h1 ~ .parrafo {
  background-color: red;
}
```

Tomando como base el ejemplo de los hermanos adyacentes, se mantendrá que no se aplicarán los estilos al primer párrafo, ya que si bien son hermanos no es consecutivo en el código, pero si aplicará los estilos a los otros 3 párrafos que están luego de nuestro **h1**, ya que son elementos hermanos que se encuentran posteriormente en el código que nuestro título.

Aplicando estilos a múltiples elementos

Css nos permite aplicar estilos a más de un elemento a la vez, separando cada uno de los elementos a los que queremos aplicar estilos mediante el uso de una **coma (“,”).**

```
h1 ~ .parrafo,
#hero h2,
div {
  background-color: red;
}
```

En este caso de ejemplo, aplicará el mismo estilo a los párrafos que sean hermanos consecutivos del **h1** con la clase “**parrafo”**, a todos los **h2** que se encuentren dentro de nuestra sección **hero** y a todos los **divs** de nuestra estructura HTML.

Especificidad en CSS

En las hojas de estilo en cascada, los estilos se aplican en el orden en que aparecen en el código. Por ejemplo, si definimos un estilo para la clase párrafo al principio de nuestro CSS pero sobre el final de nuestro documento volvemos a estilarla cambiando las mismas propiedades que utilizamos arriba, estos nuevos estilos van a sobreescribir a los anteriores.

Por otra parte, los estilos más específicos anulan a los estilos menos específicos. Aquí es donde entra en juego el concepto de **especificidad.**

La **especificidad** en CSS se refiere a la **prioridad** que se le da a un estilo determinado en el caso de que existan varios estilos que se aplican al mismo elemento. En otras palabras, la especificidad determina **qué estilo se aplicará en caso de conflicto**.

Los niveles de especificidad en orden descedente en CSS son:

1. Estilos por defecto del navegador
2. Estilos inline (estilos aplicados directamente en un elemento a través del atributo "style")
3. Selectores de ID
4. Selectores de clase, atributos y pseudoclases
5. Selectores de etiqueta

Teniendo en cuenta esto, un estilo aplicado en una hoja de estilos externa tendrá menos especificidad que un estilo inline, por lo tanto, los estilos que se aplicarán al elemento en cuestión son los que estarán especificados en los estilos inline del propio elemento dentro de nuestro archivo **HTML**.

Niveles de especificidad

La especificidad se mide en una escala de **4 niveles de números separados por puntos**, donde cada nivel representa uno de los tipos de selectores mencionados anteriormente. Por ejemplo, un selector de ID tiene una especificidad de 0,1,0,0, mientras que un selector de etiqueta tiene una especificidad de 0,0,0,1.

Nivel	Selector	Especificidad
1	Estilo por defecto del navegador	0,0,0,0
2	Estilo inline	1,0,0,0
3	Selector de ID	0,1,0,0
4	Selector de clase, atributos y pseudoclases	0,0,1,0
5	Selector de etiqueta	0,0,0,1

Si hay varios estilos que se aplican a un mismo elemento, se aplicará el estilo con la especificidad más alta.

Unidades de medida

Las **unidades de medida** en CSS son sistemas de medida utilizados para especificar el tamaño y la posición de los elementos en una página web.

Tenemos dos tipos de unidades de medida: **unidades absolutas** y **unidades relativas**.

Unidades absolutas

Las **unidades de medida absolutas** en CSS son un sistema de medida que se refiere a una **cantidad fija y constante de medida**. Estas unidades **no cambian en relación al tamaño de la pantalla o al tamaño de la fuente**. Algunos ejemplos de unidades de medida absolutas en CSS son:

Pixeles (px): Unidad de medida que se refiere a la cantidad de píxeles en la pantalla. Normalmente va a ser esta la medida absoluta que vamos a utilizar.

Puntos (pt): Unidad de medida utilizada en la impresión y en el diseño gráfico.

Centímetros (cm), milímetros (mm), pulgadas (in) y picas (pc): unidades de medida utilizadas para impresión y diseño gráfico.

Las unidades de medida absolutas se utilizan cuando es necesario tener un **control preciso y constante** sobre el tamaño de los elementos en una página web.

Unidades relativas

Las unidades de medida relativas en CSS son un sistema de medida que se refiere a una cantidad proporcional a otro elemento o valor. Estas unidades cambian en relación al tamaño de la pantalla o al tamaño de la fuente.

Las unidades de medida relativas son útiles cuando se desea que los elementos cambien de tamaño en función del tamaño de la pantalla o de la fuente. Esto puede ser útil para lograr una experiencia de usuario más uniforme en diferentes dispositivos y tamaños de pantalla. No obstante, es importante que cuando las utilicemos sepamos **en relación a que está tomando un tamaño el elemento al cual le estamos aplicando unidades de medidas relativas.**

Tipos de unidades relativas

PORCENTAJE

El porcentaje es una unidad de medida que se refiere a un porcentaje de la anchura o altura del contenedor del elemento. Por ejemplo:

```
section {
  width: 120px;
  height: 120px;
}

div {
  width: 100%;
  height: 50%;
}
```

Siendo el **div** un elemento contenido por la etiqueta section, podemos decir que el **elemento contenedor** de nuestro div es la **section**, lo cual también quiere decir que la **section es el elemento padre del div**.

Teniendo en cuenta eso, la **altura (height)** y **ancho (width)** de nuestro elemento **div** será será un **porcentaje del alto y ancho de nuestro elemento section**.

Por lo tanto nuestro div, que ocupa un **50% del alto de su contenedor** , medirá **60px** de **alto**, mientras que al tener un **ancho del 100% de su contenedor**, medirá **120px** de **ancho**.

EM

Los **em** son una unidad de medida relativa en CSS que se utiliza para especificar la longitud de un elemento en términos del tamaño de fuente. Una unidad de "em" equivale al tamaño de fuente actual del elemento en el que se utiliza.

Veamos el siguiente ejemplo:

```
    <section id="hero">
        <h1 class="titulo">Hola soy el titulo</h1>
        <p class="parrafo">Hola soy el parrafo</p>
    </section>
```

```
section {
  font-size: 24px;
}

h1 {
  font-size: 1.5em; /* 36px */
}

p {
  font-size: 0.5em; /* 12px */
}
```

Podemos notar que tanto el **h1** como el **p** son hijos directos de nuestra **section**.

Entonces, al tener en nuestra sección un tamaño de fuente de 24px , los elementos internos tendrán de base un **em** equivalente a 24px. Entonces, si, por ejemplo, al elemento **h1** le ponemos un font-size de **1.5em** , este font-size equivaldrá a **24px + 12px** , ya que es la mitad de **24px**. Por otro lado, nuestro párrafo con un font-size de **0.5em** equivaldría a la mitad de los **24px** del font-size de nuestra sección , siendo esto equivalente a **12px**.

Importante: El uso de **em** no se encuentra limitado únicamente a definir el tamaño de fuente de los elementos. Siempre se basará en el tamaño base de font-size del elemento padre, pero podemos aplicar para **ancho, alto y cualquier otra propiedad en la que se pueda definir un tamaño.**

REM

La unidad de medida **rem** en CSS es una unidad de medida relativa que se basa en el **tamaño de la fuente del elemento html**. El valor **1rem** equivale al tamaño de fuente actual del elemento **html** , que por defecto es de **16px**.

Por ejemplo, si el tamaño de la fuente del elemento **html** es de **16px**, entonces **1rem** equivale a **16px**. Si definimos un elemento con un tamaño de fuente de **2rem**, entonces su tamaño de fuente será de **32px**.

```
html{
  font-size: 20px
}

h1 {
  font-size: 1.5rem; /* 30px */
}

p {
  font-size: 0.5rem; /* 10px */
}
```

El uso de **rem** en lugar de **px** es útil para hacer que la apariencia de un sitio web sea adaptable a diferentes tamaños de pantalla. Por ejemplo, si aumentamos el tamaño de la fuente del elemento **html**, todos los elementos que usen "rem" para definir su tamaño de fuente también aumentarán su tamaño de fuente.

Importante: El uso de **rem** no se encuentra limitado únicamente a definir el tamaño de fuente de los elementos. Siempre se basará en el tamaño base de font-size del elemento **html**, pero podemos aplicar para **ancho, alto y cualquier otra propiedad en la que se pueda definir un tamaño.**

```
#hero {
  height: 10rem; /* 16px * 10 = 160px*/
  width: 20em; /*16px * 20 =  240px*/
}
```

Viewport Width y Viewport Height (vw - vh)

Existen dos unidades de medida que están basadas en el tamaño del "viewport", que es la área visible de una página web en un dispositivo.

1. **Viewport Width (vw):** Es una unidad de medida relativa al **ancho** del **viewport** de una página. **1vw** equivale al **1% del ancho del viewport**, mientras que **100vw equivale al ancho total del viewport**.

2. **Viewport Height (vh):** Es una unidad de medida relativa al **alto** del **viewport** de una página. **1vh** equivale al **1% del alto del viewport**, mientras que **100vh equivale al alto total del viewport**.

```
#hero {
  height: 50vh; /* 50% del alto del viewport (La mitad)*/
  width: 100vw; /*100% del ancho del viewport*/

}
```

Definiendo colores

En CSS, hay varias formas de especificar un color, incluyendo:

1. **Nombres de colores**: CSS tiene una lista de nombres de colores predefinidos que se pueden usar para especificar un color. Por ejemplo: **"red", "blue", "green**", etc.

2. **Hexadecimal**: Es un sistema de codificación de colores que usa una combinación de números y letras para representar un color. Por ejemplo: **"#FF0000"** representa el **rojo**.

3. **RGB**: Este sistema especifica un color en términos de valores de intensidad de los componentes rojo, verde y azul. Por ejemplo: **"rgb(255, 0, 0)"** representa el **rojo**.

4. **RGBA**: Es similar a RGB, pero también incluye un valor de opacidad, que controla la transparencia del color. Por ejemplo: **"rgba(255, 0, 0, 0.5)"** representa un **rojo semitransparente**.

5. **HSL**: Este sistema especifica un color en términos de tono, saturación y luminosidad. Por ejemplo: **"hsl(0, 100%, 50%)"** representa el **rojo**.

6. **HSLA**: Es similar a HSL, pero también incluye un **valor de opacidad**.

Cualquiera de estos formatos se puede usar para especificar un color en CSS, y la forma que escojas dependerá de tus preferencias personales y del tipo de tareas que estés realizando.

Background

Es la propiedad que se utiliza para definir el **fondo de un elemento**. Con ella, podemos establecer el **color de fondo**, agregar una **imagen de fondo**, y controlar cómo se **repetirá o se ajustará esa imagen**.

Background-color

Es la propiedad que se utiliza para definir un color de fondo

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

Background-image

Es la propiedad que se utiliza para definir una imagen de fondo para un elemento. Se utiliza la palabra reservada **url** para luego poner entre paréntesis la ruta de la imagen (ya sea que la traigamos de internet o este en nuestro ámbito de desarrollo local)

```
#hero {
  background-image: url(hero-bg.jpg);
}
```

Background-repeat

Es la propiedad que se utiliza para controlar cómo se **repite la imagen de fondo** en un elemento en caso que la misma no ocupe todo el espacio de nuestro elemento. Puede tomar los siguientes valores:

1. **repeat:** la imagen se repite tanto horizontal como verticalmente hasta llenar todo el elemento.
2. **repeat-x:** La imagen se repite horizontalmente hasta llenar el ancho del elemento.
3. **repeat-y:** La imagen se repite verticalmente hasta llenar la altura del elemento.
4. **no-repeat:** La imagen se muestra una sola vez en el centro del elemento.

```
#hero {
  background-image: url(hero-bg.jpg);
  background-repeat: repeat;
}
```

Background-position

Es la propiedad que se utiliza para controlar la **posición de la imagen de fondo en un elemento**. Puede tomar una combinación de valores en píxeles, porcentajes o palabras clave como "center" o "top left" para especificar la posición de la imagen de fondo.

Algunos de los valores más comunes son:

1. **center:** La imagen se posiciona en el centro del elemento
2. **top/bottom:** La imagen se posiciona en la parte superior o inferior del elemento respectivamente
3. **left/right:** La imagen se posiciona en el borde izquierdo o derecho del elemento, respectivamente.
4. **top left , top right , bottom left , bottom right:** La imagen se posiciona en uno de los cuatro puntos cardinales del elemento
5. **Valores separados:** También es posible especificar dos valores separados por un espacio para controlar la posición horizontal y vertical de la imagen de fondo, respectivamente.

```
#hero {
  background-image: url(hero-bg.jpg);
  /* background-position: left; */

  /*Separados horizontal y vertical*/
  background-position: center top;
}
```

Background-size

Es la propiedad que se utiliza para controlar el tamaño de imagen de fondo de un elemento.

Algunos de los valores más comunes son:

1. **auto:** La imagen se muestra en su tamaño original
2. **cover:** La imagen se amplía o reduce para cubrir completamente el elemento, sin distorsionarse.
3. **contain:** La imagen se amplía o se reduce para cubrir completamente el elemento, sin distorsionarse.
4. **Píxeles y porcentajes:** La imagen se amplía o se reduce para que se ajuste completamente dentro del elemento, sin cortarse.

```
#hero {
  background-image: url(hero-bg.jpg);
  background-size: cover;
}
```

Background como shorthand

Podemos utilizar la propiedad **background** como shorthand de las propiedades anteriormente mencionadas.

```
#hero {
  background: url(imagen.jpg) rgba(255, 255, 255, 0.8) no-repeat center center fixed/cover;
}
```

En este ejemplo, se establece una **imagen de fondo** (imagen.jpg), un **color de fondo** opaco con una transparencia del 80% (rgba(255, 255, 255, 0.8)), que **no se repetirá** (no-repeat) y se **posicionará en el centro del contenedor** tanto horizontal como verticalmente (center center). Además, la imagen de fondo se **fijará en su posición** (fixed) y se **ajustará automáticamente al tamaño** del contenedor (cover).

Es importante aclarar que no es necesario que se coloquen todas. Por ejemplo, podríamos solamente colocar el color y CSS lo interpretará correctamente.

Color

Es la propiedad que se utiliza para **cambiar el color de la fuente** de un elemento. Podemos usar todos los formatos de colores que vimos anteriormente.

```
#hero {
  color: white;
  color: #ffffff;
  color: rgb(255,255,255)
}
```

IMPORTANTE: es común, salvo en casos muy puntuales que al poner la propiedad color en un elemento que contiene otros, los elementos internos tomen el color de letra que se le colocó al elemento padre, es decir, los elementos hijos van a **heredar** la propiedad **color** del elemento que los contiene.

Box-shadow

Es la propiedad que se utiliza para **agregar una sombra** a un elemento HTML. La sombra se aplica alrededor del contenedor del elemento y puede ser utilizada para proporcionar una sensación de profundidad o para destacar un elemento en particular.

La sintaxis básica es la siguiente:

```
#hero {
  /* offset-x | offset-y | blur-radius | spread-radius | color */
  box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);
}
```

1. **offset-x:** Especifica la posición horizontal de la sombra en relación al elemento. Un valor positivo mueve la sombra a la derecha, mientras que un valor negativo la mueve a la izquierda.

2. **offset-y:** Especifica la posición vertical de la sombra en relación al elemento. Un valor positivo mueve la sombra hacia abajo, mientras que un valor negativo la mueve hacia arriba.

3. **blur-radius:** Especifica la cantidad de desenfoque de la sombra. Un valor mayor crea una sombra más difusa y extendida, mientras que un valor menor crea una sombra más nítida y pequeña.

4. **spread-radius:** Especifica la cantidad de extensión de la sombra en todas las direcciones. Un valor positivo agranda la sombra, mientras que un valor negativo la reduce.

5. **color:** Especifica el color de la sombra.

6. **inset:** Sumado a los valores de arriba , tenemos el valor inset, que se utiliza para especificar que la sombra sea interna.

De todos los anteriormente mencionados, los únicos valores obligatorios son los **offset** y el **color**.

Tipografía en CSS

La tipografía en CSS se refiere a la apariencia de los textos en un documento HTML. Se puede controlar la tipografía de un texto especificando atributos como el tipo de letra, el tamaño, el estilo (negrita, cursiva, etc.), la alineación, el color, entre otros.

Estos atributos se pueden especificar en las reglas de estilo CSS para un elemento HTML específico, o para una clase de elementos. Veamos algunas propiedades que se aplican a los textos.

Font-family

Esta propiedad es utilizada para **especificar la familia de fuentes** a utilizar en un elemento HTML. Es posible especificar una lista de familias de fuentes, en orden de preferencia, y el navegador elegirá la primera que tenga disponible en el sistema.

```
body {
  font-family: 'Montserrat', sans-serif;
}
```

Font-size

Esta propiedad es utilizada para establecer el **tamaño de la fuente** para un elemento HTML. Puedes especificar el tamaño de la fuente en diferentes unidades.

Se pueden utilizar las distintas unidades de medida que vimos anteriormente para definir el tamaño.

```
h1 {
  font-size: 20px;
  font-size: 1.5rem;
  font-size: 10%;
}
```

Es importante tener en cuenta que el tamaño de la fuente también puede ser influenciado por otros factores, como la configuración del usuario o las preferencias de accesibilidad en el navegador.

Recomendamos que al elegir la fuente se elija un tamaño de fuente que sea legible y accesible para la mayoría de los usuarios , sin exagerar en el tamaño de la misma.

Font-size

Esta propiedad es utilizada para establecer el **grosor de la fuente** para un elemento HTML.

El grosor de la fuente puede ser especificado con **números o palabras clave**.

```
h1 {
  font-weight: 700;
}
p {
  font-weight: bold;
}
```

Los números que se utilizan para especificar el grosor de la fuente varían de **100 a 900**, donde **100 es muy delgado y 900 es muy grueso**. Las palabras clave **"normal" y "bold"** son sinónimos de **400 y 700**, respectivamente. Algunos otros valores comunes incluyen **"lighter" y "bolder"**, que **ajustan el grosor de la fuente en relación al grosor predeterminado del elemento**.

Es importante tener en cuenta que **no todas las fuentes tienen todos los valores de grosor de fuente disponibles**. Por lo tanto, es importante elegir una fuente que tenga el grosor de fuente deseado.

Font-style

Esta propiedad es utilizada para establecer el **estilo de la fuente** para un elemento HTML. Los estilos de fuente incluyen **normal, italic y oblique**.

```
h1 {
  font-style: normal;
  font-style: italic;
  font-style: oblique;
}
```

El estilo **"normal"** es el estilo de fuente predeterminado y no incluye **inclinación**. El estilo **"italic"** incluye una **inclinación hacia la derecha** y es comúnmente utilizado para destacar texto o dar una sensación más relajada a la tipografía. El estilo **"oblique"** es similar al "italic", pero es **generado automáticamente por el software** en lugar de estar disponible como una fuente independiente.

Es importante tener en cuenta que no todas las fuentes tienen estilos de fuente adicionales disponibles. Por lo tanto, es importante elegir una fuente que tenga el estilo de fuente deseado.

Text-align

Esta propiedad es utilizada para establecer la **alineación horizontal** del texto dentro de un elemento HTML.

```
h1 {
  text-align: center;
  text-align: right;
  text-align: justify;
}
```

Los valores posibles para **text-align** incluyen **"left", "right", "center", "justify" y "inherit"**. La alineación de texto **"left"** es la **alineación predeterminada**, que alinea el texto a la **izquierda**. La alineación de texto **"center" centra** el texto **horizontalmente** dentro del elemento. La alineación de texto **"right"** alinea el texto a la **derecha**. La alineación de texto **"justify"** justifica el texto, es decir, **distribuye el texto uniformemente** a lo largo del **ancho** completo del elemento. La alineación de texto **"inherit"** hereda la alineación de texto del **elemento padre**.

Vertical-align

Esta propiedad se utiliza para establecer la **alineación vertical de un elemento en relación a su elemento padre o contenedor**.

Los valores más comunes de esta propiedad son **top, middle** y **bottom.**

```
.top {
  vertical-align: top;
}

.middle {
  vertical-align: middle;
}

.bottom {
  vertical-align: bottom;
}
```

En este ejemplo, los elementos con la clase **.top** tendrán una alineación vertical en la **parte superior**, los elementos con la clase **.middle** tendrán una alineación vertical en el **medio** y los elementos con la clase **.bottom** tendrán una alineación vertical en la **parte inferior**.

Text-shadow

Esta propiedad se utiliza para establecer una **sombra al texto** de un elemento HTML. La sombra se agrega detrás del texto y se puede controlar su posición, tamaño y color.

```
h1 {
  /* offset-x offset-y blur-radius color */
  text-shadow: 2px 2px 2px #333;
}
```

Estas propiedades funcionan de la misma manera que en la propiedad **box-shadow**.

Text-transform

Esta propiedad se utiliza para **transformar el texto** de un elemento HTML a **mayúsculas**, **minúsculas** o **capitalizado**.

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

El valor **“uppercase”** convertirá el texto a **mayúsculas**, el valor **“lowercase”** a minúsculas y **“capitalize”** transformará en **mayúscula la primera letra** de cada palabra.

Text-decoration

Esta propiedad se utiliza para **aplicar o quitar decoraciones** al texto de un elemento html, como **subrayados, tachados y líneas por encima**.

```
a {
  text-decoration: underline;
  text-decoration: none;
  text-decoration: overline;
  text-decoration: line-through;
}
```

Podemos aplicar cualquiera de estos tipos de decoración a nuestros textos.

Los valores posibles para text-decoration son:

1. **underline**: aplica un subrayado al texto.
2. **overline**: aplica una línea encima del texto.
3. **line-through**: aplica un tachado al texto.
4. **none**: quita cualquier decoración previa. Lo vamos a usar mucho para quitar el subrayado por defecto de los **enlaces**.

También es posible controlar el **color y el estilo** de la línea de decoración utilizando las propiedades **text-decoration-color y text-decoration-style**, respectivamente. Por ejemplo:

```
a {
  text-decoration: underline;
  text-decoration-color: blue;
  text-decoration-style: dotted;
}
```

Letter-spacing

Esta propiedad permite **controlar el espacio** entre las letras en un elemento HTML.

Se utiliza para **ajustar el espacio entre las letras en una línea** de texto, de manera que pueda verse más **legible o estética**.

```
h1 {
  letter-spacing: 2px;
}
```

Podemos utilizar como valor las diferentes **unidades de medida** que vimos anteriormente.

Es importante tener en cuenta que el espacio entre letras puede **afectar la legibilidad** del texto, por lo que es necesario utilizarlo con cuidado. Un **espacio excesivo** entre las letras puede hacer que el texto **se vea desordenado**, mientras que un **espacio insuficiente puede hacer que sea difícil de leer**.

Word-spacing

Esta propiedad permite controlar el **espacio entre las palabras** en un elemento HTML. Se utiliza para **ajustar el espacio entre las palabras en una línea de texto**, de manera que pueda verse más legible o estética.

```
p {
  word-spacing: 20px;
}
```

Podemos utilizar como valor las diferentes **unidades de medida** que vimos anteriormente.

Line-height

La propiedad **line-height** en CSS permite **controlar la altura de línea** de un elemento HTML. Esta propiedad se utiliza para ajustar la altura de la línea de texto, de manera que pueda verse más legible o estética.

```
p {
  line-height: 1.5;
}
```

En este ejemplo, se establece la altura de línea de los **párrafos** en 1.5 veces el **alto de la fuente utilizada.**

Podemos utilizar como valor las diferentes **unidades de medida** que vimos anteriormente.

Es importante tener en cuenta que la altura de línea puede afectar la legibilidad del texto, por lo que es necesario utilizarla con cuidado. Una altura de línea **demasiado baja** puede hacer que el texto sea **difícil de leer**, mientras que una altura de línea **demasiado alta** puede hacer que el texto se vea **demasiado espaciado.**

Google fonts

Google Fonts es una **colección gratuita de fuentes tipográficas** que se pueden usar en proyectos web y de diseño gráfico. Fue lanzado por Google en 2010 y desde entonces se ha convertido en una de las fuentes más populares de fuentes en línea.

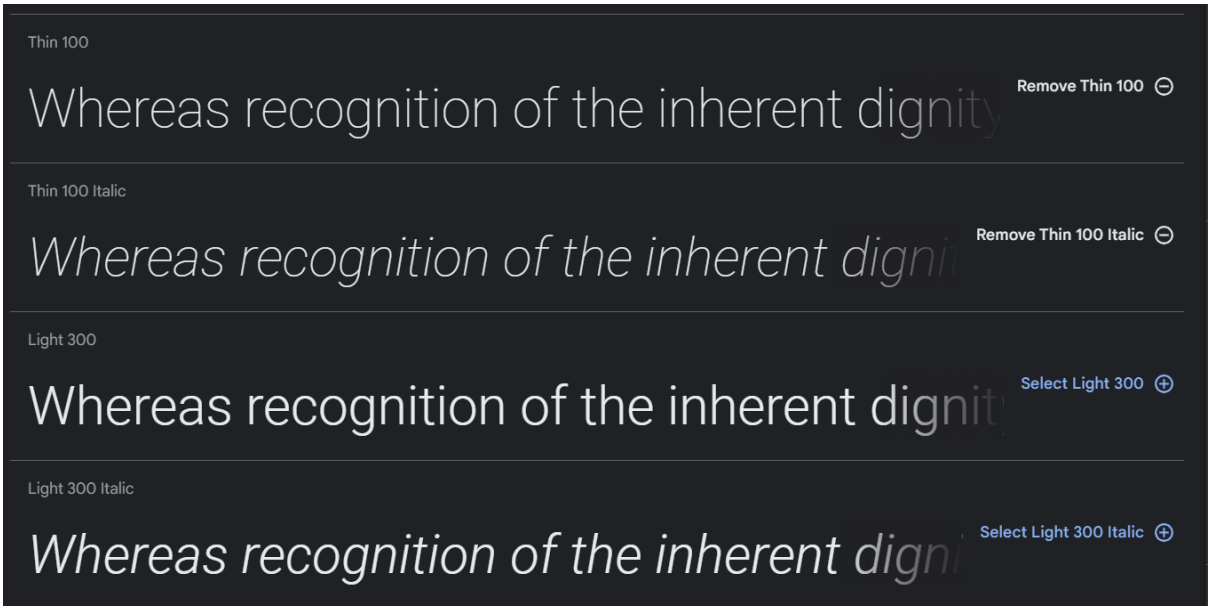
Google Fonts incluye más de **900 fuentes** de diferentes estilos y categorías, desde fuentes serif y sans-serif hasta fuentes con símbolos y emojis. Las fuentes están disponibles para su uso gratuito en proyectos personales y comerciales, lo que las hace ideales para sitios web, folletos, tarjetas de visita y otros proyectos de diseño.

Importando fuentes desde google-fonts

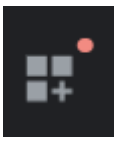
A continuación vamos a delinear los pasos para poder importar una fuente desde google fonts en nuestros proyectos. Para este ejemplo, utilizaremos la fuente **“Roboto”.**

En primer lugar, Ingresamos a **https://fonts.google.com/** y hacemos click en la fuente **“Roboto”.**

Luego, vamos a estar seleccionando cuáles son los pesos de fuente de la fuente “**Roboto”** que nos interesa tener en nuestro proyecto. Esto lo hacemos clickeando en el botón de más. que aparece en cada opción.



Luego, para abrir la barra lateral, clickeamos en el botón que aparece arriba a la izquierda en la barra de navegación que es para ver las fuentes seleccionadas.

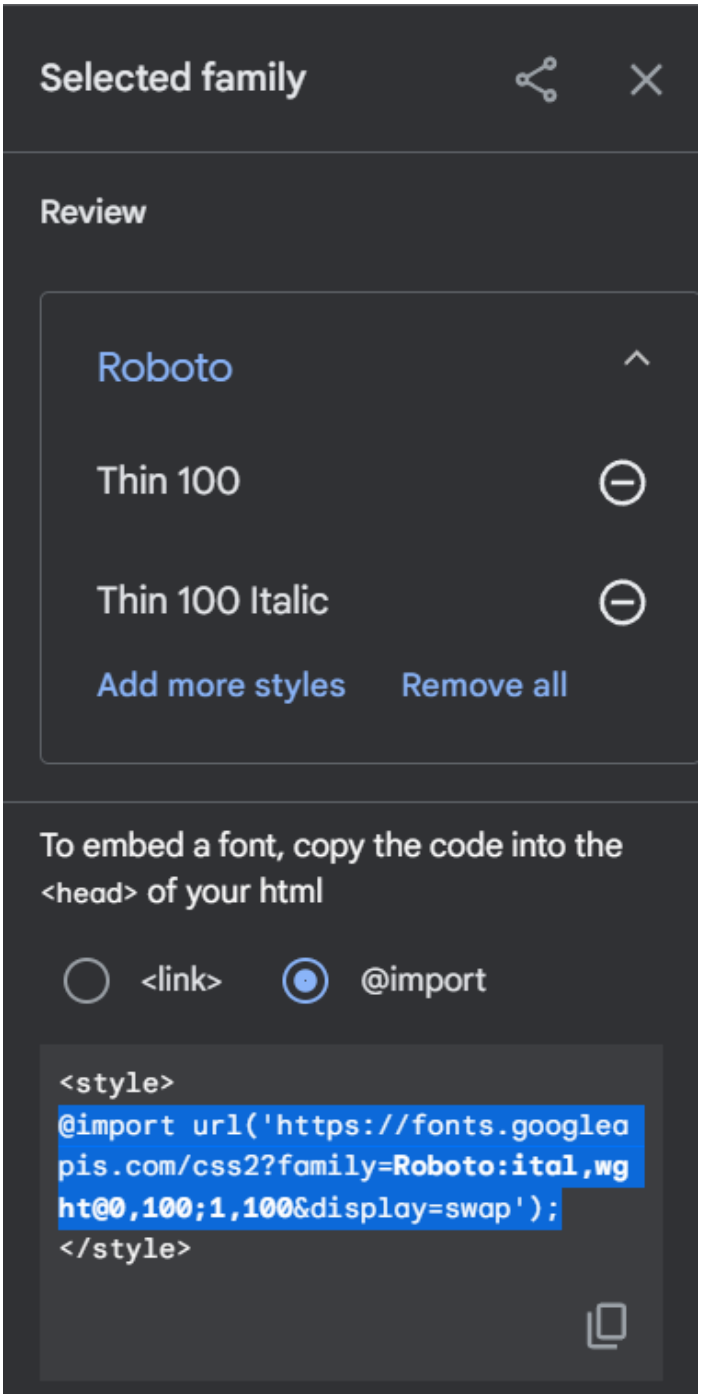


Esto desplegará la barra lateral que nos permitirá realizar el siguiente paso.

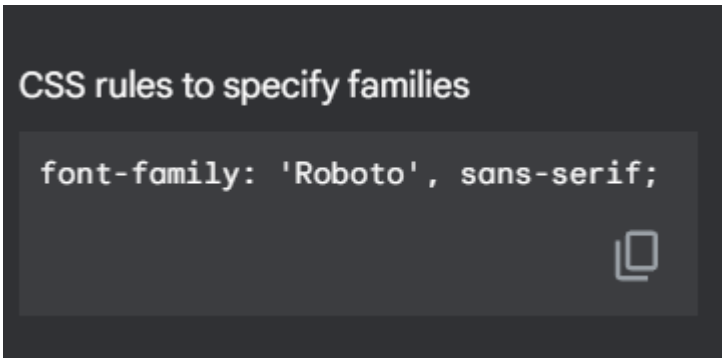
Dentro de la barra lateral tendremos dos opciones de importación.

La primera es **<link>** , la cual podremos copiar y pegar en el **head** de nuestro html para importarla en nuestro proyecto.

La segunda es **import** , la cual podremos copiar y pegar en nuestro archivo **Css** para importar la fuente. Recomendamos colocarla arriba de todo en nuestra hoja de estilos. **Importante:** Debemos quitar las etiquetas style que nos da esta opción, ya que están puestas por si queremos pegarlo dentro de nuestro **html** , lo cual no recomendamos.



Finalmente, copiamos las reglas que nos indica dentro de la misma barra lateral en el elemento al cual le queramos colocar la fuente seleccionada.



Estilos de listas

Podemos dar estilos a las listas de nuestra estructura html. A continuación, podremos ver las distintas propiedades que podemos utilizar para hacer esto:

1. **list-style-type:** Especifica el tipo de marcador que usará en la lista. Por defecto es un punto, pero podría ser un número, una letra minúscula, etc.
2. **list-style-position:** Especifica si los marcadores de la lista se deben posicionar dentro o fuera del contenido de la lista
3. **list-style-image:** Especifica una imagen personalizada para usar como marcador de la lista en lugar de un tipo de marcador predeterminado.
4. **list-style:** También podemos usar el shorthand **list-style** para especificar todo en una misma línea. Normalmente vamos a usar esta propiedad para deshabilitar los estilos de las listas colocando **list-style: none;** .

```
ul {
        list-style-type: square;
        list-style-position: inside;
        list-style-image: url(mi-imagen.png);
      }
```

Alto y Ancho

Si bien ya hemos visto implícitamente estos dos conceptos, veamos especificamente para que sirven las propiedades **height** y **width.**

Height

En CSS, **height** es una propiedad que se utiliza para establecer la altura de un elemento HTML, como un div, una imagen o un contenedor de texto. Se puede definir la altura de un elemento en cualquiera de las unidades de medida que hemos visto anteriormente.

```
div{
  height: 200px;
  height: 40%;
  height: 4rem;
  height: 5em;
  height: 30vh;
}
```

Width

En CSS, **width** es una propiedad que se utiliza para establecer el ancho de un elemento HTML, como un div, una imagen o un contenedor de texto. Se puede definir el ancho de un elemento en cualquiera de las unidades de medida que hemos visto anteriormente.

```
div{
  width: 200px;
  width: 40%;
  width: 4rem;
  width: 5em;
  width: 30vh;
}
```

Nucba tip: *Sigan investigando sobre los temas aquí expuestos. Practiquen, apoyense en la documentación oficial y sobre todo mantengan la constancia y la curiosidad a medida que vayan aprendiendo cosas nuevas.*

#HappyCoding 🚀