Facultad de Filosofía, Letras y Ciencias de la Educación

Carrera de Pedagogía Técnica de la Mecatrónica

Sistemas Microinformáticos y Programación

Tema:

Resumen CSS

Docente:

Alberto Andosilla

Estudiante:

Mateo Viscaino

Semestre:

Séptimo

Paralelo:

“A”

Periodo académico 2025-2025

Nociones Básicas

Una página web es realmente un documento de texto. En dicho documento se escribe código HTML, con el que se que crea el contenido de una web. Por otro lado, existe el código CSS, que unido al código HTML permite darle forma, color, posición (y otras características visuales) a un documento web.

La idea de CSS es la de utilizar el concepto de separación de presentación y contenido.

-Los documentos HTML (contenido) incluirán sólo información y datos, todo lo relativo a la información a transmitir.

-Los documentos CSS (presentación) incluirán sólo los aspectos relacionados con el estilo (diseño, colores, formas, etc..).

- **Forma de enlazar CSS**

Etiqueta <link rel="stylesheet"> Archivo CSS externo: El código se escribe en un archivo .css aparte.

- **Relación del HTML con CSS**

Debemos tener el documento .css enlazado desde nuestro documento .html. En su atributo href colocaremos el nombre del documento .css que contiene los estilos.

- **Sintaxis CSS**

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

```
Selector{
   Propiedad: Valor;
}
```

- **Combinador: is ()-Agrupación de selectores**

```
.container :is(.item, .parent, .element) {

}
```

- **Anidar código CSS**

- **Centrar el contenido de un elemento**

```
.parent {
  display: grid;
  place-items: center;
}
```

```
.parent {
  background: grey;

  & .element {
    background: darkred;

    &:hover {
      background: red;
    }
  }
}
```

- **Variables CSS**

```
.parent {
  --size: 300px;

  width: var(--size);
  height: var(--size);
  background: var(--color, grey);
}
```

- **Sintaxis flexible de rangos**

```
@media (800px ⩽ width ⩽ 1280px) {
  .menu {
    background: red;
  }
}
```

- **Tipos de unidades**

-Unidades absolutas: Tienen un valor fijo sin importar el contexto (como px, cm, pt).

-Unidades relativas: Se adaptan al entorno del elemento (como em, rem, %, vw, vh), lo que las hace ideales para diseño responsivo.

-Las unidades viewport en CSS se basan en el tamaño visible del navegador (la "ventana gráfica" o viewport). Son ideales para diseños responsivos. Las principales son:

-vw: 1% del ancho del viewport

-vh: 1% de la altura del viewport

-vmin: 1% del menor entre ancho y alto

-vmax: 1% del mayor entre ancho y alto

Se usan para ajustar tamaños de texto, secciones o elementos visuales que respondan al tamaño de la pantalla.

- **Rangos de dimensiones**

max-width: Ancho máximo que puede ocupar un elemento.

min-width: Ancho mínimo que puede ocupar un elemento.

max-height: Alto máximo que puede ocupar un elemento.

min-height: Alto mínimo que puede ocupar un elemento.

- **La propiedad overflow**

Overflow = [overflow-x] [overflow-y] = Propiedad de atajo que establece desbordamiento de ambos ejes.

- **La propiedad box-sizing**

border-box Las propiedades width y height incluyen el borde y relleno.

- **La propiedad margin**

El tamaño de dichos márgenes se puede alterar en conjunto (de forma general) o de forma específica a cada una de las zonas del elemento (izquierda, derecha, arriba o abajo). Basta con aplicar un ancho fijo al contenedor: width: 500px (por ejemplo) y luego aplicar un margin: auto.

- **La propiedad padding**

Existe una propiedad de atajo denominada margin y padding. Con estas propiedades evitamos tener que escribir los valores de cada parte.

- **Propiedades lógicas**

En lugar de usar direcciones físicas (izquierda, derecha, arriba, abajo), las propiedades lógicas usan términos como inicio (start) y fin (end), así como bloque y línea:

- Eje de bloque (block): dirección del flujo del contenido (por defecto, vertical).
- Eje de línea (inline): dirección en que se escribe el texto (por defecto, horizontal).

Física	Lógica	Uso
margin-top	margin-block-start	Margen al inicio del eje de bloque

padding-left	padding-inline-start	Relleno al inicio del eje en línea
border-right-width	border-inline-end-width	Grosor del borde al final del eje en línea
width	inline-size	Tamaño en el eje en línea
height	block-size	Tamaño en el eje de bloque

- **Bordes con imágenes**

CSS permite usar imágenes como bordes personalizados mediante la propiedad:

**border-image: url("imagen.png") slice repeat;

border-image-source: imagen del borde.

border-image-slice: cómo cortar la imagen (en 9 partes).

border-image-width: grosor del borde.

border-image-repeat: cómo se repite (stretch, repeat, round, space).

border-image-outset: separación desde el borde del elemento.

Color y Fondos

- **Códigos de Colores**

Detalla el uso de colores en distintos formatos (hexadecimal, rgb, hsl, palabras clave). Enseña cómo aplicar colores al texto, bordes, fondos, y cómo usar degradados (linear-gradient, radial-gradient). También muestra cómo trabajar con múltiples capas de fondo y optimización con image-set() para distintas resoluciones.

Nombres de colores (keywords)

CSS reconoce 147 nombres de colores estándar, como:

```
color: red;
```

Colores Hexadecimales (HEX)

Usan 3 o 6 caracteres para representar rojo, verde y azul.

```
color: #FF0000;    /* rojo */
```

Colores RGB

Define colores con valores de 0 a 255 para cada canal (Rojo, Verde, Azul).

```
color: rgb(255, 0, 0);      /* rojo */
```

Colores RGBA (RGB + Alpha)

Igual que RGB, pero incluye canal alfa (opacidad), de 0 (transparente) a 1 (opaco).

```
color: rgba(0, 0, 0, 0.5);   /* negro semi-transparente */
```

Colores HSL (Hue, Saturation, Lightness)

Una forma más intuitiva de ajustar tonos:

```
color: hsl(0, 100%, 50%);     /* rojo */
```

Colores HSLA (HSL + Alpha)

Como HSL pero con opacidad.

```
color: hsla(240, 100%, 50%, 0.3);  /* azul semi-transparente */
```

CurrentColor

Usa el valor actual de color para otra propiedad.

```
border-color: currentColor;
```

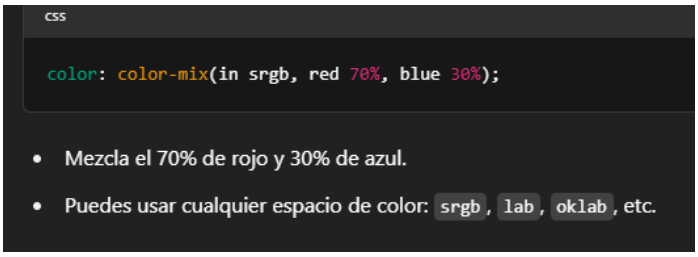
Transparent

Representa un color completamente transparente.

```
background-color: transparent;
```

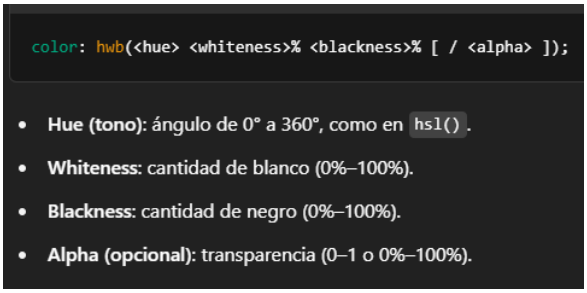
- **Colores Relativos**

CSS introdujo funciones como color-mix() y relative color syntax (from color with modifications), que permiten crear nuevos colores derivados de otros, modificando su tono, saturación, luminosidad, opacidad, etc.



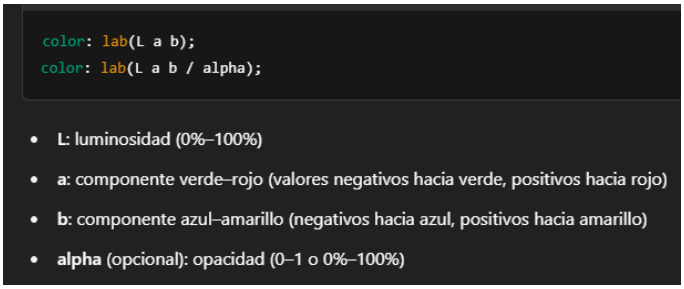
- **Función hwb() en CSS**

La función **hwb()** es una forma moderna de definir colores en CSS basada en el modelo Hue–Whiteness–Blackness. Fue introducida en CSS Color Level 4 y es más intuitiva que hsl() o rgb() en muchos casos.



- **Funciones lab() y oklab() en CSS**

Estas funciones permiten definir colores en espacios de color perceptualmente uniformes, es decir, pensados para cómo los humanos realmente percibimos los colores.



- **Variables en CSS (Custom Properties)**

Permiten almacenar valores reutilizables (como colores, tamaños o fuentes) en un solo lugar, haciendo que tu CSS sea más modular, legible y fácil de mantener.

```css
:root {
  --color-principal: #3498db;
  --tamanio-texto: 16px;
}

.elemento {
  color: var(--color-principal);
  font-size: var(--tamanio-texto);
}
```

- `--nombre-variable`: define una variable.
- `var(--nombre-variable)`: accede a su valor.

- **@property en CSS**

Permite registrar propiedades personalizadas (variables CSS) con:

-Un tipo de datos

-Un valor inicial

-Un valor inherits (si se hereda o no)

Es especialmente útil para que las variables CSS se puedan animar, lo cual antes no era posible directamente con var().

```css
@property --mi-variable {
  syntax: '<length>';
  inherits: false;
  initial-value: 0px;
}
```

Explicación:

- `--mi-variable`: nombre de la variable.
- `syntax`: tipo de dato permitido (`<length>`, `<color>`, `<percentage>`, etc.).
- `inherits`: indica si se hereda del elemento padre (`true` o `false`).
- `initial-value`: valor por defecto.

Funciones CSS

Las funciones CSS son expresiones especiales que realizan cálculos, transformaciones, combinaciones de valores o manipulación de colores, longitudes, imágenes, etc.

🔷 1. Funciones de color

Función	Descripción
`rgb()` / `rgba()`	Colores en formato rojo-verde-azul (+ alfa)
`hsl()` / `hsla()`	Tono, saturación y luminosidad (+ alfa)
`hex()`	Color hexadecimal como `#FF0000`
`hwb()`	Tono-blancura-negrura (CSS Color 4)
`lab()` / `oklab()`	Colores perceptualmente uniformes
`color-mix()`	Mezcla dos colores (CSS Color 5)
`color-contrast()`	Escoge el color con mejor contraste

🔷 2. Funciones de imagen

Función	Descripción
`url()`	Carga una imagen desde una URL
`linear-gradient()`	Degradado lineal entre colores
`radial-gradient()`	Degradado circular o elíptico
`conic-gradient()`	Degradado en forma de cono (CSS 4)
`image-set()`	Carga imágenes en diferentes resoluciones
`element()`	Usa otro elemento como imagen (limitado)

🔷 3. Funciones de transformación (`transform`)

Función	Descripción
`translate(x, y)`	Mueve un elemento en el eje X e Y
`rotate()`	Rota un elemento (grados, radianes)
`scale()`	Escala un elemento
`skew()`	Inclina un elemento
`matrix()`	Transformación completa 2D

🔷 4. Funciones matemáticas

Función	Descripción
`calc()`	Realiza cálculos aritméticos (`+`, `-`, etc.)
`min()`	Usa el valor mínimo entre opciones
`max()`	Usa el valor máximo entre opciones
`clamp()`	Restringe un valor entre un mínimo y máximo

🔷 5. Funciones para variables y propiedades

Función	Descripción
`var()`	Accede a una variable CSS
`env()`	Usa variables del entorno del dispositivo
`attr()`	Usa atributos HTML como valor CSS (limitado)

- **CSS no es un lenguaje de programación porque no tiene variables. Incorrecto**

Existen las variables CSS (llamadas realmente CSS custom properties) desde 2015. Incluso es posible tiparlas, estableciendo un tipo de dato CSS concreto.

- **CSS no es un lenguaje de programación porque no tiene condicionales». Incorrecto**

Existe una propuesta que ya está siendo implementada en navegadores para añadir condicionales en CSS mediante ternarios if() y consultas de estilos style(), así como reglas @when / @else, más orientadas a media queries y soporte del navegador.

- **CSS no es un lenguaje de programación porque no permite cálculos». Incorrecto**

CSS tiene funciones matemáticas como calc() que permiten realizar cálculos incluso entre diferentes unidades, así como funciones matemáticas

- **CSS no es un lenguaje de programación, sólo sirve para páginas webs». Incorrecto.**

Su uso principal es crear páginas webs o, junto a frameworks de Javascript, aplicaciones web, pero también se pueden crear apps o juegos móviles o incluso definir la presentación de documentos impresos.

- **Formas de simular condicionales en CSS**

Aunque no hay if() directo como en JavaScript, CSS moderno permite comportamientos condicionales mediante:

Necesidad	Solución moderna en CSS
Compatibilidad de propiedades	`@supports`
Tamaños, temas, orientación	`@media`
Estructura HTML	`:has()`
Lógica matemática	`min()` / `max()` / `clamp()`

- **@function en CSS**

CSS puro (nativo del navegador) no soporta @function como lo hace JavaScript o preprocesadores como Sass. No puedes definir funciones personalizadas directamente en CSS.

sí se usa @function en preprocesadores como Sass/SCSS.

- **Random**

Esta limitación desaparece con la incorporación de funciones CSS como random() o random-item():

random(min, max)	Genera un número aleatorio entre min y max.
random-item(val1, val2, ...)	Devuelve uno de los valores propuestos al azar.

- **attr()**

Se utiliza para obtener el valor de un atributo HTML que hemos seleccionado con CSS. Sin embargo, la sintaxis de esta función se ha visto ligeramente ampliada, y ahora tenemos más posibilidades:

`attr(name)`	Obtiene el valor del atributo `name` del elemento HTML seleccionado.
`attr(name datatype)`	Idem al anterior, pero establece el valor con el tipo de dato `datatype`.
`attr(name datatype unit)`	Idem al anterior, pero utilizando la unidad `unit` en el tipo de dato `datatype`.
`attr(name datatype, fallback)`	Idem al anterior, pero además establece un valor por defecto `fallback`.

- **Tipos de datos en CSS**

Se suelen denominar syntax (sintaxis) y deben colocarse entre signos angulares < y >

Sintaxis	Descripción	Ejemplos
<angle>	Permite indicar ángulos: unidades como deg o turn, entre otras.	45deg, 0.5turn
<color>	Indica colores CSS en formato hexadecimal, rgb() u otros.	#485432, #888
<custom-ident>	Identificadores personalizados (similar a un string, pero más limitado)	"nombre"
<image>	Incluye el tipo <url> o valores de gradientes CSS.	linear-gradient(...)...
<integer>	Indica valores numéricos enteros, ya sean positivos o negativos.	42, 10
<length>	Distancia/tamaño en una unidad CSS absoluta, relativa o de viewport.	10px, 2rem, 90vw...
<length-percentage>	Permite tanto valores <length> como <percentage>.	10px, 3rem, 60%
<number>	Permite tanto valores <integer>, como valores decimales.	4, 4.6
<percentage>	Única y exclusivamente valores de porcentajes.	70%
<resolution>	Indica resoluciones, útiles en media queries (dpi, dppx u otros).	96dpi, 300dpi
<string>	Secuencia de carácteres delimitado por comillas simples o dobles.	"«nombre»"
<time>	Valores de tiempo, como por ejemplo, s o ms.	5s, 500ms
<url>	Indica una URL mediante la función url() de CSS.	url(image.png)
<transform-function>	Funciones de transformación.	scale(1.2) o rotate(5deg)...
<transform-list>	Lista de varias funciones de transformación de las anteriores.	scale(2) translate(50px)...

- **background-image**

En el caso de querer utilizar imágenes de fondo, como ya hemos dicho, utilizaremos la propiedad background-image y en el valor, el nombre de la imagen (o la dirección URL donde está alojada), siempre rodeada del texto url().

Propiedad	Valor	Significado
background-image	none	No utiliza ninguna imagen de fondo.
background-image	url("imagen.jpg")	Usa la imagen indicada como fondo.
background-image	image-set(...)	Indica una imagen con fallbacks.
background-image	GRADIENT	Utiliza un gradiente de tipo lineal, radial o cónico.

- **Fondos y gradientes múltiples**

CSS permite aplicar varias capas de fondo a un mismo elemento combinando imágenes, colores y gradientes, separados por comas.

```
background-image: capa1, capa2, capa3;
background-position: pos1, pos2, pos3;
background-size: size1, size2, size3;
```

📝 Propiedades compatibles con múltiples capas

Propiedad	Qué hace
`background-image`	Define imágenes o gradientes
`background-position`	Posición de cada capa
`background-size`	Tamaño de cada fondo
`background-repeat`	Repetición (o no) por capa
`background-attachment`	Fijo o desplazable (scroll) por capa

- **image-set() en CSS**

Permite proporcionar múltiples versiones de una imagen (generalmente con diferentes resoluciones o formatos), para que el navegador elija la más adecuada

```css
background-image: image-set(
  "imagen-1x.jpg" 1x,
  "imagen-2x.jpg" 2x
);
```

O también:

```css
background-image: image-set(
  url("foto.webp") type("image/webp"),
  url("foto.jpg") type("image/jpeg")
);
```

- **Modelo de Caja (Box Model) → Objeto visual básico**

Todo elemento HTML en CSS se trata como una caja u "objeto visual" con estas partes:

content: contenido del elemento

padding: espacio interior

border: borde alrededor

margin: espacio exterior

object-fit → Objetos multimedia

Usada para controlar cómo se ajustan imágenes o videos dentro de su contenedor.

```css
img {
  width: 100%;
  height: 300px;
  object-fit: cover;
}
```

Valores comunes:

- `fill`: estira la imagen para llenar el contenedor
- `contain`: se ajusta sin recortarse
- `cover`: se ajusta recortando para llenar todo
- `none`: no se ajusta
- `scale-down`: la más pequeña entre `none` y `contain`

object-position → Posición del objeto dentro de su caja

Complementa a object-fit para mover la imagen dentro del área disponible:

```css
img {
  object-fit: cover;
  object-position: top right;
}
```

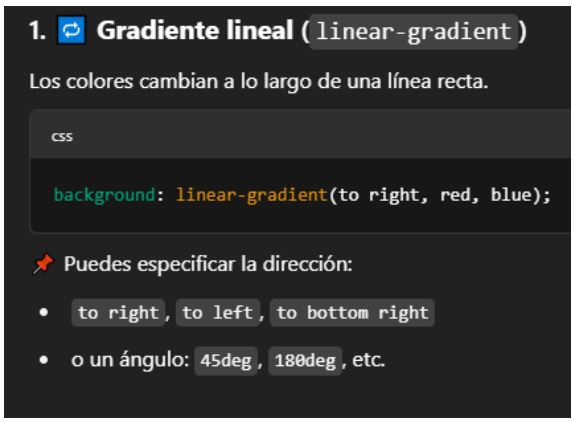
- **Procesamiento de imágenes en CSS**

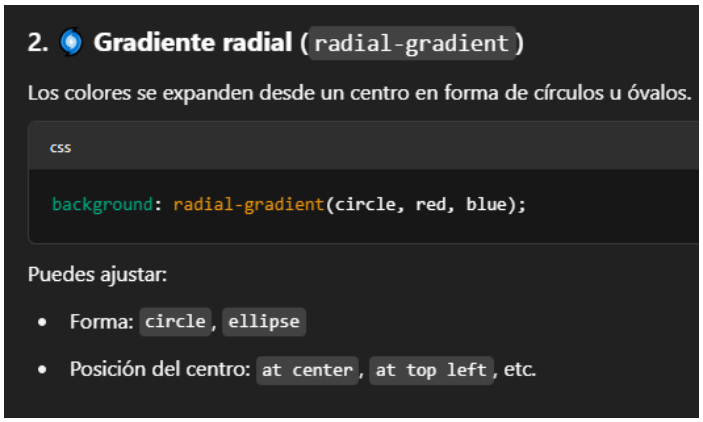
Se refiere a aplicar filtros, efectos, transformaciones, optimizaciones o ajustes visuales a imágenes usando solo CSS, sin editar los archivos originales.

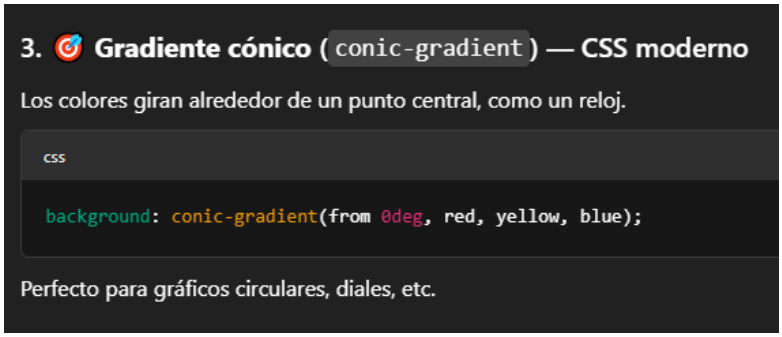
Técnica	Qué hace
`filter`	Aplica efectos visuales (blur, grayscale, etc.)
`transform`	Rota, escala o mueve imágenes
`object-fit/position`	Ajuste en su contenedor
`clip-path`, `mask`	Recorte creativo y máscaras
`image-set()`	Imágenes adaptativas por resolución o formato
`opacity`, `mix-blend-mode`	Control de transparencia y mezclas

- **Gradientes en CSS**

Los gradientes en CSS son transiciones suaves entre dos o más colores, sin necesidad de usar imágenes. Se utilizan como fondos (background) y permiten crear efectos visuales modernos, limpios y adaptables.

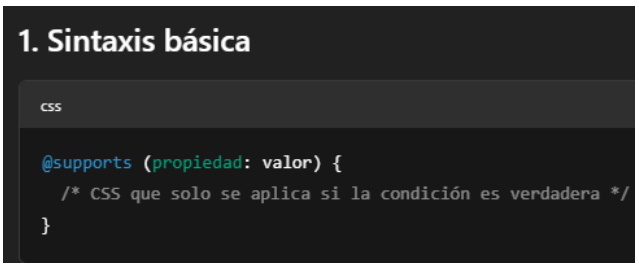






@supports en CSS (Feature Queries)

Te permite preguntar al navegador si admite una propiedad, valor o conjunto de declaraciones y, según la respuesta, aplicar estilos condicionales. Es el equivalente nativo a “feature-detection” que antes se resolvía con JavaScript o polyfills.



- **Selectores básicos en CSS**

Los selectores en CSS permiten seleccionar elementos HTML para aplicarles estilos. Los selectores básicos son los más fundamentales y utilizados.

Selector	Ejemplo	Selecciona...
Universal	`*`	Todos los elementos
Tipo	`p`	Todos los `<p>`
Clase	`.clase`	Elementos con `class="clase"`
ID	`#id`	Elemento con `id="id"`
Agrupado	`h1, p`	Todos los `<h1>` y `<p>`

Selectores de atributos en CSS

Los selectores de atributos te permiten seleccionar elementos basados en la presencia o el valor de un atributo HTML específico. Son muy útiles cuando quieres aplicar estilos dinámicamente sin añadir clases.

1. ✅ Presencia de atributo

Selecciona elementos que **tienen** el atributo (sin importar el valor).

```css
input[required] {
  border: 2px solid red;
}
```

2. □ Atributo con valor exacto

```css
a[target="_blank"] {
  color: green;
}
```

3. 🚀 Valor que comienza con (^=)

```css
a[href^="https"] {
  color: blue;
}
```

👉 Enlaces cuyo `href` comienza con `"https"` (protocolo seguro).

4. 🎯 Valor que termina con ($=)

```css
img[src$=".jpg"] {
  border-radius: 8px;
}
```

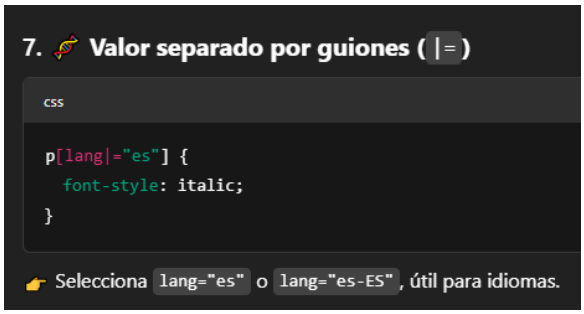
👉 Imágenes que terminan en `.jpg`.

5. 🔎 Valor que contiene (*=)

```css
div[class*="advertencia"] {
  background: yellow;
}
```

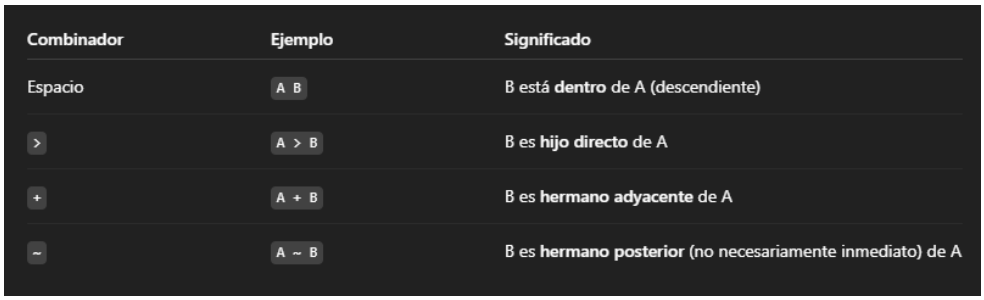
6. 🧩 Valor separado por espacios (~=)

```css
div[class~="destacado"] {
  font-weight: bold;
}
```



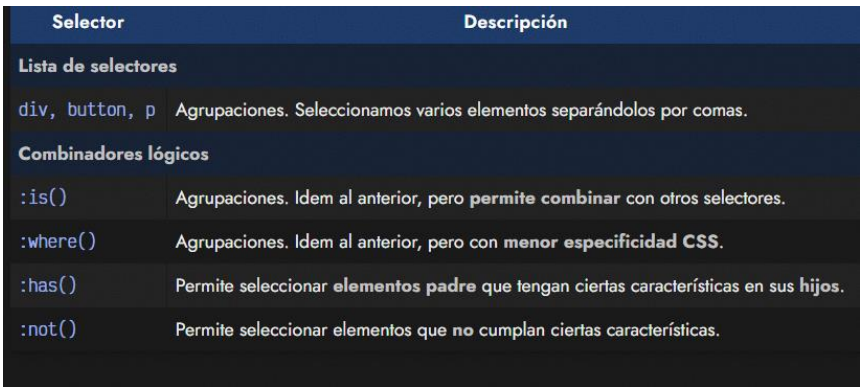
- **Combinadores en CSS**

Permiten seleccionar elementos en relación con otros elementos dentro del árbol del DOM. Son fundamentales para aplicar estilos según jerarquía, cercanía o relación entre elementos HTML.



- **Combinadores lógicos**

Nos permiten seleccionar elementos con ciertas restricciones y funcionan como una pseudoclase, sólo que se le pueden pasar parámetros, ya que son de tipo pseudoclase funcional.



- **Pseudoclases en CSS**

Son palabras clave que se agregan a los selectores para definir un estado especial de un elemento, sin necesidad de añadir clases o IDs en el HTML. Permiten aplicar estilos cuando un elemento está en cierta condición o interacción.

Pseudoclase	Uso / Descripción
`:hover`	Cuando el cursor está sobre el elemento
`:focus`	Cuando un elemento (input, botón) está enfocado
`:active`	Mientras se hace clic en el elemento
`:visited`	Enlaces ya visitados
`:first-child`	El primer hijo dentro de su padre
`:last-child`	El último hijo dentro de su padre
`:nth-child(n)`	El enésimo hijo (puede ser fórmula, como `2n` para pares)
`:not(selector)`	Elementos que **no** coinciden con el selector dado

- **Pseudoelementos en CSS**

Los pseudoelementos son palabras clave que te permiten estilizar partes específicas de un elemento HTML o insertar contenido sin modificar el HTML real. Funcionan como "elementos ficticios" dentro del documento.

Pseudoelementos	Significado
Contenido generado	Información generada desde CSS, sin existir en el HTML.
`::before`, `::after`	
Contenido tipográfico	Pseudoelementos relacionados con temas de tipografías.
`::first-line`, `::first-letter`	
Contenido destacado	Pseudoelementos para remarcar o destacar información.
`::selection`, `::target-text`, `::spelling-error`, `::grammar-error`	
WebComponents	Pseudoelementos relacionados con WebComponents
`::part`, `::slotted`	
View Transition API	Pseudoelementos de transición de cambio de página.
`::view-transition`, `::view-transition-group`, `::view-transition-image-pair` `::view-transition-new`, `::view-transition-old`	
Otros pseudoelementos	Pseudoelementos de otras categorías variadas
`::marker`, `::placeholder`, `::file-selector-button`	

- **La minificación**

Acción de eliminar carácteres o comentarios en un archivo de código para el navegador (.html, .css o .js, por ejemplo), ya que su omisión no hace que el código deje de funcionar. El objetivo es reducir su tamaño total, y por lo tanto, descargarlos más rápido.

Herramienta	Modalidad	Características
CSS Nano	CLI, PostCSS	Para automatizar desde terminal o desde PostCSS.
Clean CSS	CLI, Node	Para automatizar desde terminal.
CSSO	Node	Optimizador de CSS (clean, compress and restructuring)
HTML Minifier Terser	CLI, Node	Minificador de HTML, CSS y Javascript
Sqwish	Node	Compresor de CSS basado en Node
CSS Compressor	Online	Opciones variadas: grado de compresión, optimizaciones...
YUI Compressor	Java	Compresor CSS histórico de Yahoo

- **StyleLint**

Existen unas herramientas llamadas linters, que se encargan de revisar nuestro código a medida que lo vamos escribiendo y nos avisan de posibles problemas y malas prácticas.

Instalación de StyleLint

$ npm install --save-dev stylelint stylelint-config-standard

Configuración de StyleLint

$ npx stylelint src/index.css

-npx es una herramienta que ayuda al sistema a buscar el comando de Node

-stylelint es el nombre del linter que vamos a iniciar

-src/index.css es la ruta de nuestro archivo .css que vamos a analizar

- **Nesting**

Tradicionalmente, el código CSS se ha trabajado siempre mediante selectores CSS. Sin embargo, la idea detrás del concepto de CSS Nesting (código CSS anidado) es tener fragmentos o bloques de código uno dentro de otros, haciendo que estos selectores sean mucho más intuitivos para el programador

```
.container {
  width: 800px;
  height: 300px;
  background: grey;

  .item {
    height: 150px;
    background: orangered;
  }
}
```

- **Metodología CSS**

Se considera una metodología a una serie de métodos, consejos o forma de trabajar, que en el caso de seguir sus recomendaciones será mucho más fácil mantener y organizar tu código CSS

Año	Nombre de la metodología	Tipo	Popularidad
2009	OOCSS (Object-oriented CSS)	CSS Semántico	★ ★ ★
2009	BEM (Block Element Modifier)	CSS Semántico	★ ★ ★ ★ ★
	Soporte en Sass		
2010	Atomic CSS	CSS Atómico	★ ★ ★
	Atomizer, Tachyons, BassCSS		
2011	SMACSS (Scalable Modular Architecture CSS)	Arquitectura CSS	★ ★ ★
2014	SUITCSS	Arquitectura CSS	★ ★
2014	CSS-in-JS	CSS desde Javascript	★ ★ ★ ★
	Styled Components, Emotion, CSS Modules, WebComponents		
2015	RSCSS (Reasonable Standard for CSS)	CSS Semántico	★
2015	ITCSS (Inverted Triangle CSS)	Arquitectura CSS	★ ★
2016	FLOCSS (File, Location, Organism, Component, State, Scope)	Arquitectura CSS	★
2017	BEMIT (BEM + Inverted Triangle)	Arquitectura CSS	★ ★
2017	ABEM (Atomic Block Element Modifier)	CSS Semántico	★
2017	Utility-First CSS	CSS Atómico	★ ★ ★ ★ ★
	TailwindCSS, UnoCSS		
2018	CUBE CSS	CSS Semántico	★ ★

- **Tipografías**

Las tipografías son estilos o diseños visuales de las letras y caracteres que se usan para mostrar texto. Afectan la legibilidad, el tono y la estética del contenido escrito.

- **Bases de fuentes y tipografías**

Las fuentes se clasifican en familias como:

Serif: Letras con pequeños remates (ej. Times New Roman).

Sans-serif: Sin remates, más limpias (ej. Arial).

Monospace: Cada letra ocupa el mismo espacio (ej. Courier).

Cursivas o script: Imitan escritura a mano.

Estas bases ayudan a elegir la fuente adecuada según el contexto.

- **Decoraciones de texto**

Son efectos que puedes aplicar al texto, como:

Subrayado (underline)

Tachado (line-through)

Sombras para dar profundidad.

Estas decoraciones mejoran la presentación o enfatizan partes del texto.

- **Ajuste y balance de textos**

Consiste en controlar espacios para mejorar la legibilidad:

Espaciado entre letras (letter-spacing)

Espaciado entre palabras (word-spacing)

Interlineado o altura de línea (line-height)

Ajustes para evitar cortes o saltos de línea incómodos y mejorar el flujo visual.

- **Contorno en tipografías**

Es la técnica de agregar un borde o línea alrededor de las letras para destacar el texto o mejorar su visibilidad sobre fondos complejos. En CSS se puede simular con sombras o filtros, ya que no hay propiedad directa para contornos.

- **Textos con degradados**

Aplicar un degradado de color como relleno del texto (ejemplo: de azul a rosa). Se hace usando propiedades como background-clip y text-fill-color para crear efectos visuales modernos y llamativos.

- **Textos y alineaciones**

Controlar la alineación del texto dentro de su contenedor:

Izquierda (por defecto en la mayoría de idiomas)

Centro

Derecha

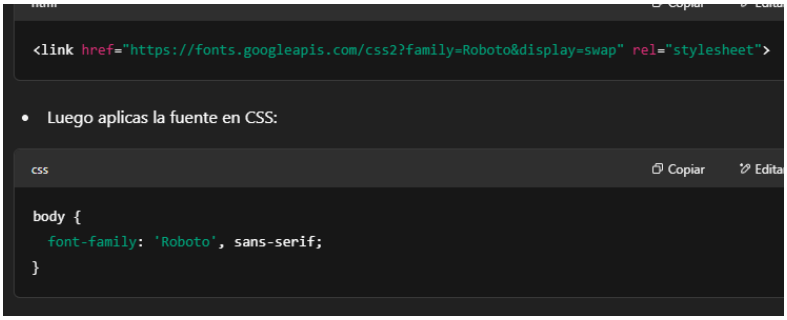
Justificado (alineado a ambos lados, formando bloques limpios)

Esto ayuda en la estructura y estética del contenido.

- **Tipografías con Google Fonts**

Google Fonts es una plataforma gratuita y muy popular que ofrece una gran variedad de fuentes web listas para usar.

Puedes acceder a cientos de familias tipográficas que funcionan en todos los navegadores modernos.



- **Tipografías con Fontsource**

Fontsource es una solución para proyectos que prefieren gestionar las fuentes localmente en lugar de depender de servicios externos.

Funciona mediante paquetes npm que se pueden instalar y usar en frameworks modernos (React, Vue, Next.js, etc.).

- **La regla @font-face**

Esta regla CSS permite declarar fuentes personalizadas que no están instaladas en el sistema del usuario.

Puedes cargar fuentes locales o externas especificando la ruta al archivo de la fuente (.woff, .woff2, .ttf, .eot).

- **Modo de carga de tipografías**

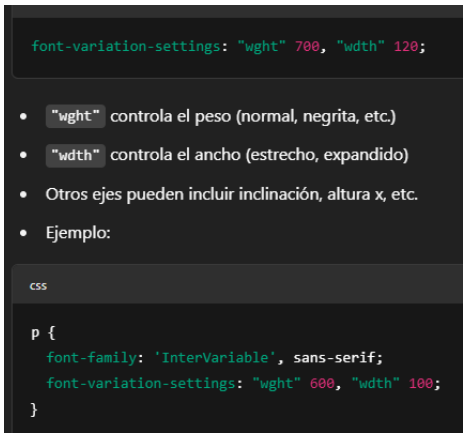
El modo de carga controla cómo se muestran las fuentes mientras se descargan, para evitar el "FOIT" (Flash Of Invisible Text) o "FOUT" (Flash Of Unstyled Text).

Usando la propiedad font-display dentro de @font-face, defines el comportamiento:

- **Fuentes variables CSS**

Las fuentes variables son un formato avanzado que agrupa múltiples estilos (peso, ancho, inclinación) en un solo archivo.

Esto reduce la cantidad de archivos a descargar y permite cambiar estilos de forma dinámica con CSS.



- **Características personalizadas**

Muchas fuentes OpenType tienen características avanzadas que se activan para mejorar la estética y funcionalidad:

Ligaduras (liga) para combinar letras (ej. “fi” → fi)

Números tabulares para alinear cifras

Versalitas para mayúsculas pequeñas

Alternancias estilísticas para variar el diseño de letras

Estas características se activan con:

- **Layouts**

Un layout en CSS es la forma en que organizamos y posicionamos los elementos dentro de una página web. Sirve para estructurar el contenido visualmente: definir dónde va el menú, el contenido principal, las columnas, el pie de página, etc.

El layout controla la distribución y alineación de los elementos en la pantalla.

Enseña a crear estructuras de página responsivas y organizadas:

- Flexbox: distribución unidimensional (horizontal o vertical), alineación con justify-content y align-items, y control de crecimiento (flex-grow).
- Grid: estructuras bidimensionales complejas, definición de áreas (grid-template-areas), fracciones (fr), líneas y nombres.

Explica cuándo usar cada sistema y cómo combinarlos.

- **Tipos principales de layouts en CSS**

Layout estático o normal

-Es el flujo natural de los elementos en HTML.

-Los elementos se colocan uno debajo de otro (block) o uno al lado del otro (inline).

-No se usa ninguna propiedad especial.

Float (flotado)

Usado para colocar elementos a la izquierda o derecha.

Los elementos flotados se sacan del flujo y permiten que el texto o contenido fluya alrededor.

```css
img {
  float: left;
  margin-right: 10px;
}
```

Flexbox (Flexible Box Layout)

-Para layouts unidimensionales: filas o columnas.

-Muy útil para centrar, distribuir espacio, ordenar elementos.

-Propiedad principal: display: flex;

```css
.container {
  display: flex;
  justify-content: center; /* horizontal */
  align-items: center; /* vertical */
  height: 100vh;
}
```

CSS Grid

-Para layouts bidimensionales: filas y columnas.

-Permite dividir el espacio en una cuadrícula flexible.

-Propiedad principal: display: grid;

```css
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
}
```

Position (Posicionamiento)

-Para posicionar elementos de forma exacta y fuera del flujo normal.

-Valores: relative, absolute, fixed, sticky.

```css
.box {
  position: absolute;
  top: 10px;
  right: 20px;
}
```

- **Interfaz de Usuario (UI)**

Se centra en estilizar formularios, botones, inputs y otros componentes interactivos. Enseña buenas prácticas para accesibilidad (colores, foco, texto alternativo) y cómo usar appearance, accent-color, y pseudoestados para mejorar la experiencia del usuario sin JavaScript.

- **Efectos**

Explora cómo aplicar efectos visuales:

- Transiciones: animación suave entre cambios de estado (transition)
- Transformaciones: rotación, escala, traslación (transform)
- Animaciones: creación de movimientos personalizados con @keyframes

También se cubren efectos 3D y filtros (filter: blur, brightness).

- **Responsive**

Creación de sitios adaptables usando:

-Media queries para cambiar estilos según resolución (@media)

-Diseño fluido con unidades relativas (em, %)

-Técnicas modernas como clamp(), min(), max() y container queries

-Todo orientado a que un mismo diseño funcione bien en móviles, tablets y escritorios.

- **Transformaciones y Dibujos**

Muestra cómo usar solo CSS para crear figuras y gráficos complejos (círculos, triángulos, iconos), aprovechando clip-path, shape-outside, gradientes y bordes. Ideal para reducir imágenes externas y mejorar rendimiento.