«Talento Tech»

Front-End JS

Clase 04

Buenos Aires Ciudad

Clase N° 4: Introducción a CSS

Temario:

1. **Bases del CSS**
    - Concepto y utilidad de CSS
    - Separación de estilos y estructura HTML
    - Reutilización de estilos
2. **¿Cómo Incorporar CSS?**
3. **Selectores Básicos**
4. **Especificidad, Herencia, Cascada y Orden de las Reglas en CSS**
5. **Atributo básicos de css**
    - Color
    - Tamaño
6. **Metodología BEM**
    - Bloque
    - Elemento
    - Modificador

Objetivo de la clase:

En esta clase vamos a descubrir qué es el CSS y para qué sirve, entendiendo cómo nos ayuda a separar el contenido (HTML) del estilo visual de nuestras páginas web. Aprenderemos las diferentes formas de incorporar CSS en nuestros proyectos: en línea, dentro del mismo archivo HTML o mediante archivos externos, y por qué es importante reutilizar estilos para trabajar mejor. También conoceremos los selectores básicos que nos permiten elegir qué elementos queremos modificar, como el selector universal, de etiqueta, de clase y de ID. Además, vamos a entender conceptos clave como la especificidad, la herencia y la cascada, que determinan cómo se aplican y combinan las reglas de CSS, para que puedas controlar el diseño con precisión. Finalmente, veremos los atributos básicos para definir colores, tamaños y las unidades de medida más usadas, para que puedas darle vida y estilo a tus páginas desde cero. Por último implementaremos una de las metodologías más utilizadas a la hora de estilar.

Bases del CSS

CSS (Cascading Style Sheets u *“hojas de estilo en cascada”* en español) es la herramienta que usamos para dar vida a nuestras páginas web, dándoles color, formas y un diseño atractivo. Sin CSS, nuestras páginas serían simples bloques de texto e imágenes. Su principal función es controlar cómo se presentan los elementos HTML en el navegador. Pero, ¿qué es esto de la **"cascada"**? Este término hace referencia a cómo se resuelven los conflictos cuando varias reglas se aplican a un mismo elemento.

Importancia del CSS:

- **Separación de “concerns”:** CSS nos permite separar el contenido (HTML) de su presentación, lo que hace que todo sea más organizado. Imaginate tener todos los estilos mezclados con el HTML, ¡un lío total!
- **Reutilización de estilos:** Lo mejor de CSS es que los estilos que definimos en un archivo pueden ser utilizados en varias páginas, logrando así un diseño consistente sin repetir código y dando un mismo criterio estético a nuestras páginas web.

¿Cómo incorporar CSS?

Podemos agregar CSS a nuestras páginas de tres maneras diferentes. Algunas son mejores que otras según la situación, pero te cuento sobre cada una de ellas.

CSS en línea

Es el más básico y fácil de aplicar, pero no es realmente recomendable porque “ensucia” el HTML. Se usa con el atributo `style` dentro de una etiqueta.

Ejemplo:

```
<p style="color: red; font-size: 14px;">Este párrafo es rojo y tiene un tamaño de catorce píxeles</p>
```

CSS interno

Esta metodología la utilizaremos cuando queramos aplicar estilos sólo a una página específica. Colocamos el CSS dentro de la etiqueta `<style>` en el `<head>` del documento.

Ejemplo:

```
<head>
  <style>
      h1 {
          color: blue;
          background-color: yellow;
      }
  </style>
</head>
```

2.3 CSS externo

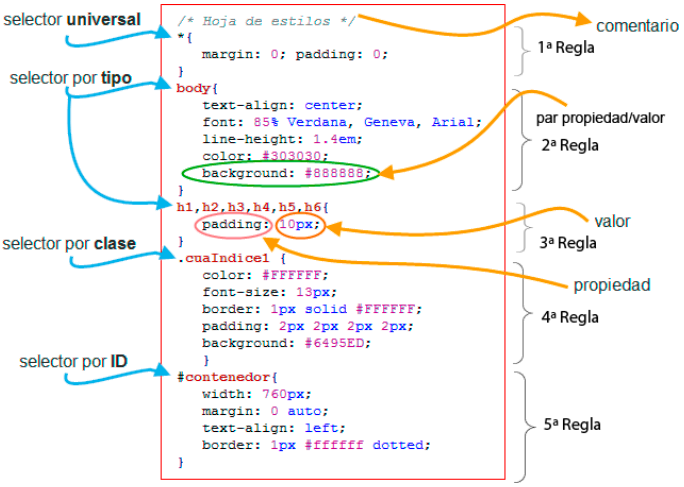
El mejor y más utilizado. Guardamos el CSS en un archivo aparte y lo vinculamos con el HTML usando la etiqueta `<link>`. ¡Ideal para proyectos más grandes!

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

🛑Usá CSS en línea solo para pruebas rápidas o correcciones puntuales.
CSS interno podés usarlo en ejercicios simples.Pero para todo el desarrollo del proyecto: **¡CSS externo siempre!**

Selectores básicos

En CSS, los selectores nos permiten aplicar estilos a los elementos HTML que queramos, eligiendo específicamente a cuáles de ellos afectaremos con los estilos especificados. Hay varias formas de asignar estilos, a continuación explicaremos los más básicos.



Selector

- Es lo que usamos para decir a qué elemento HTML le vamos a aplicar el estilo.
- **Ejemplo:** `h1`, `.titulo`, `#principal`
- Va al inicio de la regla, antes de las llaves.

Propiedad

- Es la **característica** que queremos modificar del elemento, como el color, tamaño, fuente, margen, etc.
- **Ejemplo:** `color`, `font-size`, `background`
- Va **dentro de las llaves**, seguida de dos puntos (`:`).

Valor

- Es el **valor específico** que le damos a la propiedad, o sea, **cómo queremos que se vea**.
- **Ejemplo:** `blue`, `16px`, `center`
- Va **después de los dos puntos**, seguido de un punto y coma (`;`).

```
selector {
      propiedad: valor;
}
```

Selector universal

Este selector **(*)** afecta a todos los elementos de la página. Es útil para aplicar reglas generales.

Ejemplo:

```
* {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}
```

Selector de etiqueta

Aplica estilos a todos los elementos de un tipo, por ejemplo, a todos los párrafos `<p>`.

Ejemplo:

```
p {
   color: black;
```

```
    font-size: 16px;
}
```

Selector de clase

Si querés aplicar un estilo a varios elementos, pero no a todos, podés usar clases. Para eso, definimos una clase con un punto (`.`) y se la asignamos a los elementos que queramos

Ejemplo:

```
.highlight {
    background-color: yellow;
}
```

Selector de ID

Para un estilo único, usamos un **ID**. Solo puede aplicarse a un elemento específico. Lo definimos con un numeral **(#).**

Ejemplo:

```
#main-title {
    font-size: 24px;
    color: green;
}
```

!important

A veces hay conflictos de estilos y queremos asegurarnos de que **ese estilo sí o sí se aplique**. Para eso usamos `!important`.

"Si hay estilos que se pisan entre sí y necesitás que uno gane sí o sí, le ponés `!important` al final de su valor."

Especificidad, herencia, cascada y orden de las reglas en CSS

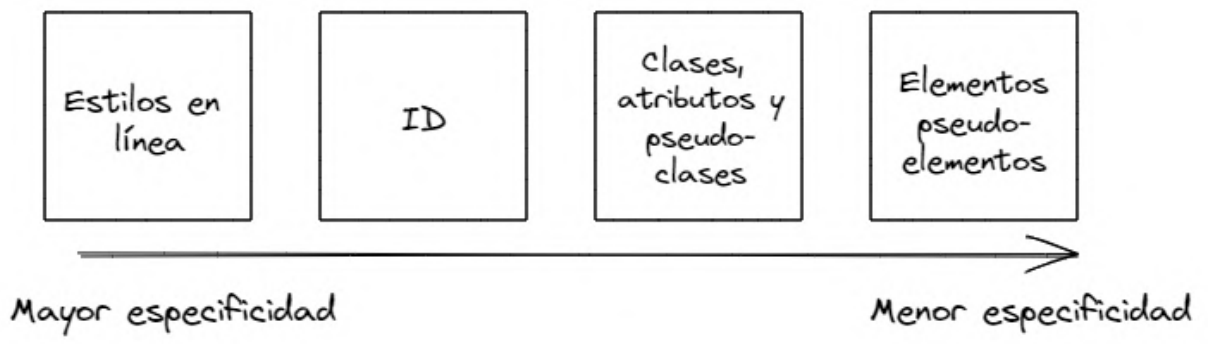
Cuando varios estilos intentan aplicarse al mismo elemento, CSS utiliza un sistema de prioridades para determinar cuál regla tendrá efecto. Este sistema se basa en los conceptos de **especificidad**, **herencia**, **cascada** y el **orden de las reglas**.

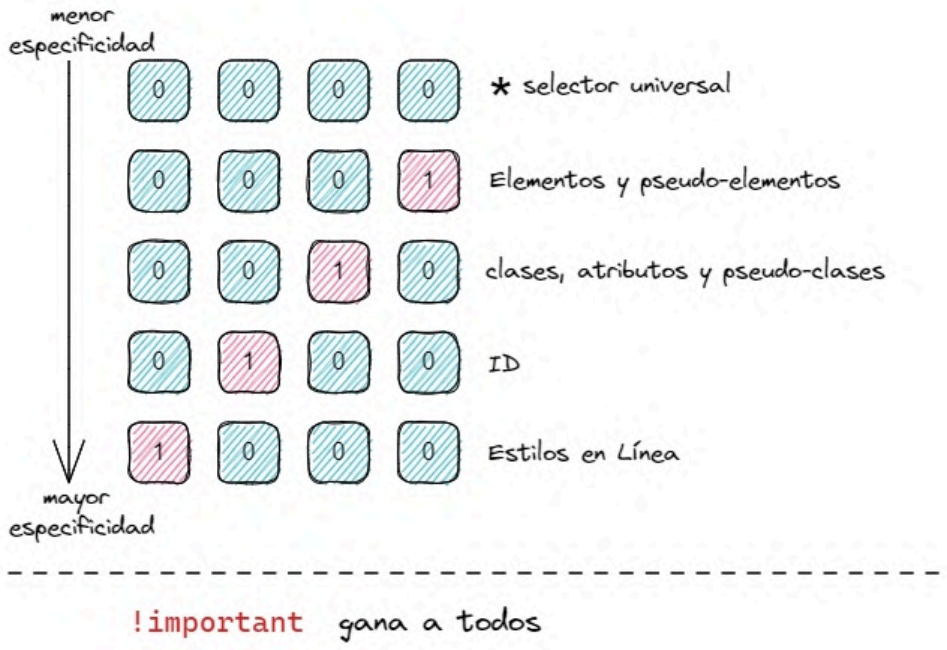
Especificidad

La especificidad indica la prioridad de una regla CSS según el tipo de selector que se utiliza. Cuanto más específica sea la regla, mayor prioridad tendrá. El orden de especificidad, de mayor a menor, es:

- Estilos en línea (atributo `style` dentro de la etiqueta HTML)
- Selectores de ID (`#id`)
- Selectores de clase (`.clase`), pseudo-clases y atributos
- Selectores por etiqueta (elementos HTML)

Por ejemplo, un estilo definido en línea siempre tendrá prioridad sobre un estilo aplicado mediante una clase o un ID.





Herencia

Algunas propiedades CSS, como `color` o `font-family`, se **heredan automáticamente** de los elementos padres a sus hijos. Esto significa que si defines el color en un contenedor `<div>`, todos los elementos dentro de ese contenedor adoptarán ese color, a menos que tengan una regla específica que lo sobreescriba.

Cascada

Cuando dos o más reglas aplican a un mismo elemento y tienen la **misma especificidad**, CSS aplica la regla que aparece **más abajo** en el archivo CSS. Esto significa que las reglas que se escriben al final del archivo tienen prioridad sobre las escritas antes, si la especificidad es igual.

Orden de las reglas

En resumen, cuando dos reglas afectan al mismo elemento y tienen la misma especificidad, **se aplicará la que esté definida al final del código CSS**. Esto es porque CSS sigue un orden de lectura de arriba hacia abajo.

Ejemplo:

```
p {
   color: blue;
}
p {
   color: red; /* Esta regla se aplicará porque está después */
}
```

Propiedades Básicas en CSS

- **`color`**
  Cambia el color del texto.
  Ej: `color: red;` pinta el texto de rojo.

- **`background-color`**
  Cambia el color de fondo del elemento.
  Ej: `background-color: yellow;` pone fondo amarillo.

- **`text-align`**
  Alinea el texto dentro del elemento.

Valores comunes: `left` (izquierda), `center` (centrado), `right` (derecha).
Ej: `text-align: center;` centra el texto.

- **`font-weight`**
  Controla el grosor del texto.
  Valores comunes: `normal` (normal), `bold` (negrita).
  Ej: `font-weight: bold;` hace el texto negrita.

- **`text-decoration`**
  Agrega decoración al texto, como subrayado o tachado.
  Valores comunes: `none`, `underline`, `line-through`.
  Ej: `text-decoration: underline;` subraya el texto.

- **`font-style`**
  Cambia el estilo del texto a cursiva o normal.
  Valores: `normal`, `italic`.
  Ej: `font-style: italic;` pone el texto en cursiva.

- **`font-size`**
  Controla el tamaño del texto.
  Se puede usar con unidades como `px`, `em`, `rem`, `%`.
  Ejemplo: `font-size: 16px;` — define el tamaño del texto a 16 píxeles.

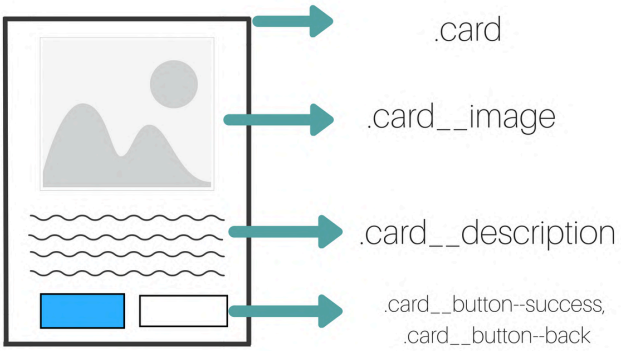
- **`visibility`**
  Controla si el elemento es visible o no, pero mantiene su espacio.
  Valores: `visible` (visible), `hidden` (invisible).
  Ej: `visibility: hidden;` oculta el elemento pero mantiene su espacio.

- **`display`**
  Controla cómo se muestra el elemento.
  Valores comunes: `block` (bloque), `inline` (en línea).
  Ej: `display: none;` oculta completamente el elemento.

- **`cursor`**
  Cambia el tipo de cursor cuando se pasa por encima del elemento.
  Ej: `cursor: pointer;` muestra la mano que indica clickeable.

BEM

La metodología BEM, o Block Element Modifier, es un enfoque para estructurar y organizar código CSS que se centra en la creación de componentes reutilizables y mantenibles. Divide la interfaz de usuario en bloques independientes, cada uno con elementos y modificadores, utilizando convenciones de nomenclatura para facilitar la comprensión y el mantenimiento del código.



Block (Bloque)

Es un contenedor principal que representa una parte independiente del sitio.
Ejemplo:

```
<article class="noticia"></article>

.noticia {
  background: lightgray;
}
```

Element (Elemento)

Es una parte interna del bloque, como un título o un botón.
Se escribe como `bloque__elemento`.

```
<h1 class="noticia__titulo">Título</h1>

.noticia__titulo {
  text-transform: capitalize;
}
```

Modifier (Modificador)

Sirve para cambiar la apariencia o comportamiento del bloque o elemento.
Se escribe con `--modificador`.

```
<article class="noticia noticia--destacada">
  <h1 class="noticia__titulo noticia__titulo--uppercase">Título</h1>
</article>

.noticia--destacada {
  background-color: cornsilk;
}
```

```css
.noticia__titulo--uppercase {
  text-transform: uppercase;
}
```

Ventajas

- Código más legible y reutilizable
- Evita conflictos de estilos
- Ideal para trabajar en equipo

Desventajas

- Las clases pueden ser largas
- No siempre es necesario en proyectos pequeños

¿Cuándo usar BEM?

Usá BEM para mantener tu CSS ordenado, especialmente si trabajás en equipo o usás frameworks donde querés personalizar estilos sin romper la estructura.

💭**Storytelling**

¡Hora de darle estilo a Talento Lab!

Recordemos…

Hasta ahora, ya construiste la estructura base, agregaste navegación, imágenes, videos y formularios. Pero…
¿Cómo hacemos para que el sitio luzca profesional y atractivo?
¡Llegó el momento de darle estilo!

Lucía – Product Owner

“Un sitio sin estilo puede ser funcional, pero poco atractivo para el usuario. Queremos que Talento Lab sea limpio, claro y moderno. Para eso, es esencial que aprendas a separar la estructura HTML del diseño con CSS.”

Tomás – Desarrollador Senior

“Vamos a trabajar con un archivo CSS externo, para que el diseño esté organizado y sea fácil de mantener. Además, vamos a darle vida a la barra de navegación para que sea más amigable y profesional.”

Ejercicio práctico #1

Incorporar CSS externo

¿Qué tenés que hacer?

- Crear un archivo llamado `styles.css` en la misma carpeta donde está tu archivo HTML.
- Vincular ese archivo CSS dentro del `<head>` de todos tus HTML con la línea:

```
<link rel="stylesheet" href="styles.css">
```

- Asegurarte de tener un `<header>` con un título y un `<footer>` con un texto de derechos reservados.

```
<header>
    <h1>Mi Sitio Web</h1>
</header>
<footer>
    <p>© 2024 Mi Sitio Web. Todos los derechos reservados.</p>
</footer>
```

- En el archivo `styles.css`, aplicar estilos para:
  - Cambiar el color de fondo del header y footer.
  - Cambiar el estilo de la fuente (tipo, tamaño, color).
  - Aplicar márgenes o padding para separar elementos.
  - Usar al menos **un selector de clase** y **un selector de elemento**.

¡Acordate de ir usando todo lo que ya vimos!

- ➢ **HTML semántico**: Usá etiquetas como `<section>`, `<article>`, `<nav>`, `<main>`, etc.
- ➢ **Listas y enlaces**: Incluí una lista o menú de navegación con enlaces (usando rutas relativas).
- ➢ **Multimedia**: Podés insertar una imagen con su atributo `alt` bien escrito.
- ➢ **Tablas**: Si querés, agregá una tabla con información de ejemplo.
- ➢ **Formularios**: Probá incluir un pequeño formulario de contacto.
- ➢ **Elementos en bloque y en línea**: Observá cómo se comportan cuando aplicás estilos.
- ➢ **Metodología BEM**: Si te animás, nombrá clases usando la lógica de bloque-elemento-modificador.
- ➢ **Rutas relativas**: Asegurate de que todos tus recursos (CSS, imágenes, etc.) estén bien vinculados.

Ejercicio práctico #2:

Modificar la barra de navegación (Navbar)

¿Qué tenés que hacer?

- Asegurarte que tu HTML tiene una barra de navegación similar a esta:

```
<nav>
      <ul>
           <li><a href="#">Inicio</a></li>
           <li><a href="#">Acerca de</a></li>
           <li><a href="#">Servicios/Productos</a></li>
           <li><a href="#">Contacto</a></li>
       </ul>
</nav>
```

- En `styles.css`, aplicar estos estilos:
    - Usar `list-style: none;` para que la lista no tenga viñetas.
    - Quitar el subrayado de los enlaces con `text-decoration: none;`.
    - Cambiar el color del texto de los enlaces con la propiedad `color`.


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
Buenos
Aires
Ciudad