# ¿Qué es CSS?

Diferencia entre HTML (estructura) y CSS (presentación)

**CSS o (Cascading Style Sheets)** es un lenguaje que sirve para darle estilo a las páginas web, es decir, para definir colores, tamaños, posiciones, fuentes, entre otras cosas.

En pocas palabras

**HTML** es la estructura o el esqueleto de la página. Por ejemplo, crea los títulos, párrafos, imágenes, etc.

**Mientras que el CSS** se encarga de cómo se ve esa estructura (colores, tamaños, márgenes,etc).

## ¿Cómo se conectan?

Se puede conectar CSS a HTML de tres formas:

- En línea: dentro de una etiqueta con `style="..."`
- Interno: dentro de `<style>...</style>` en el `<head>` del HTML
- Externo: enlazando un archivo `.css` con `<link rel="stylesheet" href="estilos.css">`

Sintaxis básica:

**La estructura del CSS es:**

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

Ejemplo:

```
p {
    color: blue;
    font-size: 16px;
  }
```

En este ejempló, "`p`" es el selector (selecciona todos los párrafos), dentro de las llaves `{}` están las reglas con propiedades y valores.

Selectores esenciales:

**Tipo:** selección de etiquetas Ejemplo: `p`

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

**Clase:** selecciona los elementos con una clase. "Empieza con punto `.`"

```
.boton { background: green; }
```

**ID:** selecciona un elemento único. "Empieza con `#`"

```
#principal { font-weight: bold; }
```

**Descendiente:** selecciona elementos dentro de otros. Ejemplo: todos los `span` dentro de `div`

```
div span { color: orange; }
```

```
```

**Agrupados:** aplican reglas a varios selectores a la vez

```
h1, h2, h3 { margin-bottom: 10px; }
```

**Pseudoclases:** estilos en estados especiales

```
a:hover { color: purple; } /* cambia color al pasar el mouse */
```

**Pseudoelementos:** agregar contenido o estilo antes o después

```
p::before { content: "Nota: "; color: gray; }
```

## Cascada, herencia y especificidad:

**Cascada:** si varios estilos afectan un elemento, gana el que está más abajo o con más prioridad.

**Herencia:** algunos estilos como color o fuente se heredan de padres a hijos, pero otros como margen no.

**Especificidad:** se calcula según el tipo de selector (id > clase > tipo).

Ejemplo:

```
p { color: blue; }       /* especificidad baja */
.boton { color: red; }   /* más específica */
#principal { color: green; } /* la más específica */
```

Si un" `p`" tiene clase `.boton` y id `#principal`, el color que se verá será **verde** porque el id tiene mayor especificidad.

## Modelo de caja (box model):

### Modelo de caja (box model)

Cada elemento se compone de:

**Content (contenido):** lo que contiene (texto, imagen)

**Padding:** espacio interno entre contenido y borde

**Border:** borde visible alrededor

**Margin:** espacio externo entre elementos

**Ejemplo CSS con bordes visibles:**

```
div {
```

```
  width: 200px;
  padding: 10px;
  border: 2px solid black;
  margin: 20px;
}
```

### Unidades y colores

- **Unidades:**
    - **`px`:** son los píxeles fijos
    - **`%`:** El porcentaje del contenedor padre
    - **`em`:** relativo al tamaño de fuente del elemento padre
    - **`rem`:** relativo al tamaño raíz (html)
    - **`vh`, `vw`:** porcentaje del alto/ancho de la ventana
- **Colores:**
    - Hexadecimal: `#ff0000` (rojo) (hay muchos más)
    - RGB: `rgb(255, 0, 0)`
    - HSL: `hsl(0, 100%, 50%)`

***Cuando elegir***

Hex y RGB son comunes, HSL es útil para ajustar brillo o saturación fácilmente.

## Tipografía:

- `font-family`: el tipo de letra
- `font-size`: es el tamaño
- `line-height`: el espacio entre líneas para legibilidad
- `font-weight`: el grosor (normal, bold, números)

Ejemplo:

```css
body {
    font-family: Arial, sans-serif;
    font-size: 16px;
    line-height: 1.5;
    font-weight: 400;
    }
```

## Layout moderno:

**Flexbox:** organiza en una dirección (fila o columna), útil para alinear elementos en línea.

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

**Grid:** organiza en filas y columnas, ideal para layouts complejos.

```css
.grid {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-gap: 10px;
    }
```

## Responsive y media queries:

Sirven para que la página se adapte a diferentes tamaños de pantalla.

Ejemplo:

```css
@media (max-width: 600px) {
    body {
    background-color: lightgray;
    }
    }
```

Este código cambia el fondo si la pantalla es menor a 600px

## Buenas prácticas:

Organiza el código y el uso de comentarios

Usar nombres claros y descriptivos para las clases (`.btn-enviar` en vez de `.b1`)

Evitael uso de `!important` para no complicar la cascada

Mantén consistencia en las unidades (`px` o `em`, no mezclar mucho)

Asegurar un buen contraste de colores para accesibilidad y que el texto sea legible