

# **UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ**

**FACULTAD DE INGENIERÍA DE SISTEMAS**



## **MONOGRAFIA:**

---

**“CSS Básico: Fundamentos para el diseño web  
moderno”**

---

## **INTEGRANTES:**

**Fernández Marcelo, Gerald**

**León García, David Daniel**

**Montes Vargas, Diego Ronaldo**

**Rojas Molina, Yaser**

**Samaniego Inga, Alex Piero**

**Vásquez Huamán, Jhair**

**HUANCAYO - PERÚ**

**2025**

# Índice

Introducción.....	4
Resumen .....	5
Abstract.....	5
Presentación del Tema.....	6
Objetivo General .....	7
Objetivos Específicos .....	7
Justificación.....	8
Metodología Utilizada.....	9
Desarrollo.....	10
1.1    Selectores y Especificidad.....	10
1.1.1    ¿Qué son los Selectores? .....	10
1.1.2    ¿Qué es la Especificidad? .....	13
1.1.3    Ejemplo.....	13
1.1.4    Selectores Compuestos.....	13
1.2    Propiedades de Textos y Fuentes.....	22
1.2.1    Principales Propiedades de Texto y Fuentes.....	22
1.2.2    Ejemplo.....	23
1.3    Modelo de Caja (Box Model) .....	23
1.3.1    Partes del Box Model .....	23
1.3.2    Ejemplo Gráfico (Descripción).....	24
1.3.3    Ejemplo Práctico.....	24
1.4    Colores y Fondos .....	24
1.4.1    Formas de Definir los Colores .....	24
1.4.2    Propiedades para Fondos.....	24
1.4.3    Ejemplo.....	25
1.5    Unidades de Medida.....	25
1.5.1    Unidades Absolutas.....	25
1.5.2    Unidades Relativas .....	25
1.5.3    Ejemplo.....	25
1.6    Posicionamiento .....	26
1.6.1    Tipos de Posicionamiento .....	26
1.6.2    Ejemplo.....	26
1.7    Flexbox (Diseño Flexibles) .....	26
1.7.1    Propiedades del Contenedor (Padre).....	26
1.7.2    Propiedades del Ítem (Hijo) .....	27

1.7.3	Ejemplo Básico .....	27
1.8	Grid Layout (Diseño de Cuadrícula) .....	27
1.8.1	Propiedades Principales del Contenedor (Grid) .....	27
1.8.2	Ejemplo Básico .....	28
1.8.3	CSS.....	28
1.8.4	HTML .....	28
1.9	Pseudo-Clases y Pseudo-Elementos .....	28
1.9.1	Pseudo-Clases.....	28
1.9.2	Ejemplo.....	28
1.9.3	Pseudo-Elementos .....	29
1.9.4	Ejemplo.....	29
1.10	Transiciones y Animaciones Básicas .....	29
1.10.1	Transiciones .....	29
1.10.2	Propiedades Comunes .....	29
1.10.3	Ejemplo .....	30
1.10.4	Animaciones.....	30
1.10.5	Ejemplo .....	30
	Conclusión.....	31
	Referencias Bibliográficas .....	33

## Introducción

CSS, siglas de Cascading Style Sheets (Hojas de Estilo en Cascada), es un lenguaje fundamental en el desarrollo web, utilizado para controlar el aspecto visual de los documentos HTML. Mientras que HTML define la estructura y el contenido de una página web (como encabezados, párrafos, listas o imágenes), CSS permite definir cómo se verá ese contenido: colores, tipografías, tamaños, márgenes, posiciones y muchos otros aspectos de diseño.

El uso de CSS ofrece una clara separación entre el contenido y la presentación, lo que facilita el mantenimiento del sitio y permite aplicar estilos de forma uniforme en múltiples páginas. Por ejemplo, si se desea cambiar el color de fondo de todas las páginas de un sitio web, solo se necesita modificar una sola regla CSS en lugar de editar cada archivo HTML individualmente.

CSS funciona mediante reglas que se aplican a elementos HTML. Cada regla tiene un selector (que indica a qué elemento se aplica) y un bloque de declaración (donde se especifican los estilos). Estas reglas pueden ser escritas directamente en el archivo HTML (estilo en línea o interno), pero lo más recomendable es usar un archivo externo .css, lo cual permite reutilizar los estilos y mantener el código más limpio y organizado.

## **Resumen**

La presente monografía aborda los aspectos fundamentales del lenguaje CSS (Cascading Style Sheets), utilizado para definir la presentación visual de los documentos HTML. Se expone de manera clara y concisa el uso de selectores, propiedades de texto, fuentes y el modelo de caja, así como el manejo de colores, fondos y unidades de medida. Además, se analizan las distintas formas de posicionamiento de elementos en la página, junto con los sistemas de diseño Flexbox y Grid, ampliamente utilizados en la maquetación moderna. También se exploran las pseudo-clases, pseudo-elementos y la incorporación de transiciones y animaciones básicas que enriquecen la experiencia visual. A través de este trabajo se ofrece una visión general y práctica sobre CSS, orientada al diseño de interfaces web más estructuradas, estéticas y funcionales.

## **Abstract**

This monograph addresses the fundamental aspects of the CSS (Cascading Style Sheets) language, used to define the visual presentation of HTML documents. It clearly and concisely explains the use of selectors, text properties, fonts, and the box model, as well as the management of colors, backgrounds, and units of measurement. It also analyzes the different ways of positioning elements on the page, along with the Flexbox and Grid design systems widely used in modern layout. It also explores pseudo-classes, pseudo-elements, and the incorporation of basic transitions and animations that enrich the visual experience. This work offers a general and practical overview of CSS, aimed at designing more structured, aesthetic, and functional web interfaces.

## **Presentación del Tema**

Las Hojas de Estilo en Cascada (CSS, por sus siglas en inglés Cascading Style Sheets) constituyen uno de los pilares fundamentales en el desarrollo de sitios web modernos. Mientras que el lenguaje HTML se encarga de definir la estructura y el contenido de una página web, CSS permite dotar a ese contenido de una presentación visual coherente, atractiva y funcional. A través de CSS es posible definir estilos como colores, tipos y tamaños de letra, márgenes, espacios internos, alineaciones, disposición de los elementos, e incluso efectos visuales y animaciones. Gracias a la separación entre contenido y presentación que ofrece CSS, se facilita el mantenimiento del código, la reutilización de estilos y la mejora de la experiencia de usuario.

Con el avance de la tecnología y el aumento del consumo digital, los usuarios no solo valoran que una página web funcione correctamente, sino que también esperan que su diseño sea intuitivo, visualmente atractivo y adaptable a distintos dispositivos. En este contexto, comprender los conceptos fundamentales de CSS básico permite a desarrolladores y diseñadores construir sitios web que cumplan con estas expectativas, respetando principios de estética, funcionalidad y accesibilidad. Por lo tanto, dominar CSS no es solo una cuestión técnica, sino una competencia clave para crear experiencias digitales de calidad.

## **Objetivo General**

Comprender los conceptos fundamentales de CSS y su correcta aplicación en la estilización y diseño visual de páginas web, con el fin de crear interfaces modernas, estructuradas y atractivas para el usuario.

## **Objetivos Específicos**

- Explicar la función de los selectores CSS y los distintos tipos existentes.
- Analizar el concepto de especificidad y su importancia en la resolución de conflictos de estilo.
- Aplicar propiedades relacionadas con el texto, fuentes, fondos y colores.
- Entender el Modelo de Caja (Box Model) como base del diseño de elementos en la web.
- Diferenciar y utilizar correctamente las unidades de medida absolutas y relativas en CSS.
- Implementar técnicas de posicionamiento de elementos usando valores como static, relative, absolute, fixed y sticky.
- Diseñar distribuciones responsivas y flexibles mediante los sistemas de diseño Flexbox y Grid.
- Emplear pseudo-clases y pseudo-elementos para estilizar elementos de forma dinámica.
- Incorporar transiciones y animaciones básicas que enriquezcan la experiencia de navegación.
- Analizar el concepto de especificidad y su importancia en la resolución de conflictos de estilo.

## **Justificación**

En el entorno digital actual, donde millones de usuarios interactúan diariamente con interfaces web, ofrecer una experiencia de usuario atractiva y eficiente se ha convertido en una necesidad. Un sitio web que presenta un diseño pobre, desorganizado o difícil de navegar genera desconfianza y puede impactar negativamente en la percepción del contenido o del servicio ofrecido. Por ello, el dominio de CSS básico representa una herramienta indispensable para los profesionales del desarrollo y del diseño web.

El conocimiento y correcto uso de CSS no solo mejora la apariencia de un sitio, sino que también contribuye al rendimiento, la accesibilidad y la capacidad de adaptación del contenido a diferentes dispositivos y resoluciones. En un mundo donde el acceso a la información se da a través de múltiples plataformas (computadoras, tablets, teléfonos móviles), el diseño responsivo y bien estructurado que CSS permite es un factor clave para garantizar la calidad de la experiencia digital.

Esta monografía se justifica en la necesidad de brindar una base sólida en el uso de CSS, enfocándose en aquellos conceptos y herramientas que permiten crear sitios web funcionales, modernos y visualmente agradables. Además, busca fomentar buenas prácticas de codificación y diseño que perduren más allá de las tendencias momentáneas.



## **Metodología Utilizada**

Para la elaboración de esta monografía se empleó una metodología basada en la revisión bibliográfica y documental. Se consultaron fuentes confiables como la documentación oficial del World Wide Web Consortium (W3C), libros especializados en desarrollo web y diseño con CSS, así como artículos técnicos de referencia actualizados disponibles en plataformas de desarrollo reconocidas.

Además del análisis teórico, se presentan ejemplos prácticos y fragmentos de código comentado para ilustrar la aplicación de los conceptos explicados. Estos ejemplos tienen como objetivo facilitar la comprensión, especialmente para lectores que se inician en el uso de CSS. El enfoque didáctico de esta monografía permite al lector avanzar progresivamente desde los aspectos más básicos del lenguaje hasta técnicas más estructuradas, como el uso de Flexbox y Grid Layout.

# Desarrollo

## 1.1 Selectores y Especificidad

### 1.1.1 ¿Qué son los Selectores?

En CSS, un selector es la herramienta que indica a qué elementos HTML se aplicarán los estilos definidos en una regla. Es el primer componente de una regla CSS y funciona como una “dirección” que apunta a uno o varios elementos del documento HTML.

Existen varios tipos de selectores, cada uno con usos específicos:

- **Selectores de tipo:** se refieren directamente al nombre de una etiqueta HTML y aplican estilos a todas las instancias de esa etiqueta.

```
h1 {  
  color: blue;  
}
```

- **Selectores de clase:** se aplican a cualquier elemento que posea el atributo class con el nombre indicado. Se utilizan precedidos por un punto (.).

```
.titulo {  
  font-weight: bold;  
  font-size: 24px;  
}
```

- **Selectores de ID:** se aplican a un único elemento con un identificador único. Se utilizan precedidos por el símbolo #.

```
#principal {  
  margin: 0 auto;  
  width: 80%;  
}
```

- **Selectores de atributo:** seleccionan elementos con atributos específicos.

```
input[type="text"] {  
  border: 1px solid gray;  
}
```

- **Combinadores:** permiten seleccionar elementos en función de su relación con otros elementos (hijos, hermanos, descendientes, etc.).

```
div > p {  
  color: green;  
}
```

- **Pseudo-clases y pseudo-elementos:** se utilizan para aplicar estilos según estados o partes específicas del contenido.

### ¿Qué son?

Las pseudo-clases y pseudo-elementos permiten aplicar estilos a elementos HTML de forma más específica sin necesidad de modificar el marcado original. Son herramientas potentes para estilizar elementos según su estado, posición o partes internas.

### Pseudo-clases

Una pseudo-clase se utiliza para definir un estado especial de un elemento. Se escribe con dos puntos: seguidos del nombre de la pseudo-clase.

### Ejemplo:

```
/* Cambia el color del enlace cuando el cursor pasa sobre él */  
a:hover {  
  color: blue;  
}  
  
/* Aplica estilo al primer elemento hijo */  
li:first-child {  
  font-weight: bold;
```

```
}

/* Aplica estilo al input cuando está enfocado */
input:focus {
  border: 2px solid green;
}
```

## Pseudo-elementos

Un pseudo-elemento permite aplicar estilo a una parte específica del contenido de un elemento. Se escribe con doble dos puntos :: (aunque algunos navegadores permiten uno solo por compatibilidad).

### Ejemplo:

```
/* Estiliza la primera línea de un párrafo */
p::first-line {
  font-style: italic;
  color: darkgray;
}

/* Agrega contenido antes del texto de un párrafo */
p::before {
  content: "→ ";
  color: #888;
}

/* Agrega contenido después del texto */
p::after {
  content: " ✓";
  color: green;
}
```

### 1.1.2 ¿Qué es la Especificidad?

Cuando múltiples reglas CSS coinciden con un mismo elemento, la especificidad determina cuál tiene prioridad. Cada selector tiene un “peso” que el navegador interpreta para decidir qué estilo aplicar.

- Orden de prioridad (de mayor a menor):
- Estilos en línea (escritos directamente en el atributo style del HTML)
- Selectores de ID
- Selectores de clase, atributos y pseudo-clases
- Selectores de tipo o etiquetas
- Reglas universales o herencia (peso menor)

### 1.1.3 Ejemplo

#### HTML

```
<p id="importante" class="texto">Hola mundo</p>
```

#### CSS

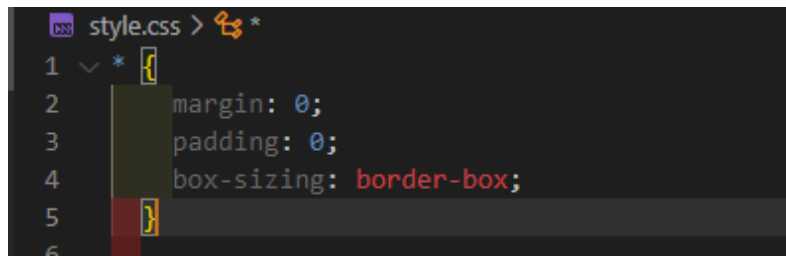
```
p { color: blue; }  
.texto { color: green; }  
#importante { color: red; }
```

El navegador aplicará el color **rojo**, ya que el selector de ID #importante tiene mayor especificidad que los demás.

### 1.1.4 Selectores Compuestos

#### Selector universal

El selector universal en CSS es representado por el asterisco (\*). Su función principal es seleccionar todos los elementos del documento HTML, sin importar su tipo, clase, ID u otra característica. Este selector se utiliza comúnmente cuando se desea aplicar un estilo global o resetear márgenes y rellenos predeterminados del navegador.



```
1  * {  
2    margin: 0;  
3    padding: 0;  
4    box-sizing: border-box;  
5  }
```

En este ejemplo, todos los elementos del sitio web se verán afectados por estas reglas: se eliminan los márgenes y rellenos predeterminados, y se establece la propiedad box-sizing en border-box, lo cual facilita el control de dimensiones en el diseño web.

#### **Ventajas:**

- Uniformiza el estilo base en todos los navegadores.
- Útil para establecer estilos iniciales comunes en toda la página.
- Facilita la implementación de diseños consistentes.

#### **Precauciones:**

- Puede afectar el rendimiento si se usa en documentos muy grandes, ya que selecciona todos los elementos.
- No debe usarse para aplicar estilos específicos o complejos.

### **Selector de Etiqueta**

El selector de etiqueta, también conocido como selector de tipo, es uno de los selectores más básicos y comunes en las hojas de estilo en cascada (CSS). Este selector permite aplicar estilos directamente a todos los elementos HTML de un mismo tipo o etiqueta, como `<p>`, `<h1>`, `<div>`, `<a>`, entre otros, sin necesidad de utilizar clases o identificadores.

Este tipo de selector es útil cuando se desea estandarizar la apariencia de ciertos elementos en toda la página web, proporcionando consistencia visual y facilitando el mantenimiento del código.

```
style.css > ...  
1 p {  
2   font-size: 16px;  
3   line-height: 1.5;  
4   color: #333333;  
5 }
```

En este ejemplo, todas las etiquetas <p> (párrafos) del documento HTML tendrán el mismo tamaño de fuente, interlineado y color de texto. Este enfoque es ideal cuando se quiere mantener una estructura uniforme en bloques de contenido.

### **Ventajas:**

- Permite estilizar elementos de forma rápida y coherente.
- Reduce la necesidad de repetir clases cuando el estilo aplica a muchos elementos del mismo tipo.
- Mejora la legibilidad del código CSS.

### **Precauciones:**

- No debe utilizarse en exceso para personalizaciones específicas, ya que puede limitar la flexibilidad en el diseño.
- Puede ser sobrescrito por selectores más específicos como clases, identificadores o selectores compuestos.

### **Selectores de clase**

En CSS, los selectores de clase se utilizan para aplicar estilos a uno o varios elementos HTML que comparten la misma clase. Se definen anteponiendo un punto (.) al nombre de la clase.

#### **Sintaxis básica**

```
.nombre-clase {  
  
  propiedad: valor;  
  
}
```

#### **Ejemplo:**

```

<!DOCTYPE html>
<html>
<head>
  <style>
    .resaltado {
      color: red;
      font-weight: bold;
    }
  </style>
</head>
<body>

<p class="resaltado">Este texto está resaltado en rojo.</p>
<p>Este texto no tiene clase.</p>
<div class="resaltado">Este también está resaltado.</div>

</body>
</html>

```

### Características

- Puedes aplicar la misma clase a múltiples elementos.
- Un elemento puede tener varias clases separadas por espacios.
- Los nombres de clase no pueden comenzar con un número y deben evitar caracteres especiales (excepto guiones o guiones bajos).

### Selector ID en CSS

El selector ID en CSS se utiliza para aplicar estilos a un único elemento HTML que tiene un atributo id específico. A diferencia de las clases, que pueden ser reutilizadas en múltiples elementos, un ID debe ser único dentro de un documento HTML. Esto asegura que un único elemento sea seleccionado y estilizado con ese ID.

La sintaxis básica de un selector ID es la siguiente:

```

#nombre-id {
  propiedad: valor;
}

```

- El símbolo # indica que estamos usando un ID.



- nombre-id es el valor del atributo id del elemento HTML que se desea estilizar.
- Dentro de las llaves { } se especifican las propiedades CSS que se quieren aplicar al elemento.

### **Características Principales del Selector ID:**

#### **Unicidad**

- El valor de id debe ser único dentro del documento HTML. No puede haber más de un elemento con el mismo id.
- Esta característica garantiza que los estilos aplicados mediante el ID afecten exclusivamente a un solo elemento.

#### **Alta especificidad**

- Los selectores ID tienen una especificidad alta, lo que significa que sus reglas prevalecen sobre otras reglas con menor especificidad, como las que usan clases o selectores de tipo.

#### **Accesibilidad en JavaScript**

- Los elementos con un id específico pueden ser accedidos rápidamente mediante JavaScript utilizando document.getElementById('nombre-id'), lo cual facilita la manipulación e interacción con el DOM.

### **Ejemplo Básico de Uso del Selector ID:**

#### **HTML**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Ejemplo Selector ID</title>
  <link rel="stylesheet" href="estilos.css">
</head>
<body>
  <h1 id="titulo-principal">Bienvenido a Mi Página</h1>
  <p id="mensaje">Este es un mensaje importante.</p>
</body>
</html>
```

## CSS

```
#titulo-principal {
  color: navy;
  font-size: 32px;
  text-align: center;
}

#mensaje {
  color: darkred;
  background-color: #ffe4e1;
  padding: 10px;
  border: 2px solid red;
  font-weight: bold;
}
```

### Comparación con Otros Selectores

Selector	Símbolo	Aplicación	Especificidad
Tipo	p, div	A todos los elementos de ese tipo	Baja
Clase	.clase	A todos los elementos con esa clase	Media
ID	#id	A un único elemento con ese ID	Alta

El selector ID tiene una especificidad más alta que los selectores de tipo o clase. Esto significa que, si hay reglas conflictivas, el estilo aplicado con un selector ID tendrá prioridad sobre los demás.

### Agrupación de Selectores en CSS

La agrupación de selectores es una técnica utilizada en hojas de estilo en cascada (CSS) que permite aplicar un mismo conjunto de reglas de estilo a múltiples elementos HTML. Esta práctica busca optimizar el código, mejorar la legibilidad y reducir la redundancia, evitando la repetición de reglas idénticas para distintos selectores.

#### Sintaxis

La agrupación se logra mediante la separación de múltiples selectores mediante comas (.). La estructura general es la siguiente:

#### CSS

```
selector1, selector2, selector3 {  
  propiedad: valor;  
}
```

Cada uno de los selectores separados por coma recibirá el mismo conjunto de estilos definidos entre llaves.

### Ejemplo Básico

A continuación, se muestra un ejemplo de cómo aplicar un mismo color de texto a encabezados de distintos niveles:

CSS

```
h1, h2, h3 {  
  color: #333333;  
  font-family: 'Segoe UI', sans-serif;  
}
```

En este caso, los encabezados h1, h2 y h3 se verán afectados por las mismas propiedades: color de texto gris oscuro y fuente tipográfica Segoe UI.

### Aplicaciones Comunes

- Aplicar márgenes y padding uniformes a distintos tipos de contenedores (div, section, article).
- Unificar estilos de botones que comparten una lógica visual pero difieren en funcionalidad (.btn-guardar, .btn-cancelar, .btn-editar).
- Estilizar todos los elementos de formulario con bordes, tipografía y tamaños coherentes (input, select, textarea).

CSS

```
input, select, textarea {  
  border: 1px solid #cccccc;  
  padding: 10px;  
  font-size: 14px;  
}
```

### Ventajas de la Agrupación

- Mantenimiento sencillo: cualquier cambio en las propiedades afecta a todos los elementos agrupados, evitando la necesidad de modificar cada uno por separado.
- Código más limpio: reduce la cantidad de líneas de código y mejora la claridad visual.
- Rendimiento: permite al navegador interpretar los estilos más rápidamente debido a la simplicidad del archivo CSS.

## **Selector Descendente**

Un selector es una parte fundamental de una regla CSS que especifica a qué elementos HTML se le deben aplicar ciertos estilos. Existen diversos tipos de selectores: de tipo, de clase, de ID, de atributo, pseudo-classes, y combinadores.

Se representa por un espacio entre dos selectores:

```
A B {  
  propiedad: valor;  
}
```

Donde:

- A es el elemento ancestro (padre, abuelo, etc.).
- B es el elemento descendiente que será estilizado.

## **Sintaxis y Funcionamiento**

La sintaxis de un selector descendente es simple pero poderosa. Se basa en aplicar estilos a todos los elementos B que se encuentren dentro de un elemento A, sin importar el nivel de anidamiento.

### **Ejemplo básico:**

HTML:

```
<div class="contenedor">
  <p>Primer párrafo</p>
  <section>
    <p>Segundo párrafo dentro de sección</p>
  </section>
</div>
```

CSS:

```
.contenedor p {
  color: blue;
}
```

Este código aplicará el color azul a todos los párrafos que estén dentro de un elemento con clase contenedor, sin importar si están directamente dentro o dentro de otros elementos.

### Selector de Etiqueta de Clases

Este selector es un selector de etiqueta, que además restringe su aplicación a la condición de llevar el atributo de clase o de id que se le indica.

```
14  /*Selecciona elementos especifico de la clase*/
15  v p.miClase{
16    |   font-size: 25px;
17    | }
18
```

Recordemos que distintos elementos de html pueden tener la misma clase. Al usar este selector solo selecciona a los elementos específicos seleccionados.

```
<div class="miClase"></div>
<p class="miClase">Mi parrafo</p>
<h3 class="miClase">Mi encabezado</h3>
```

No es lo mismo el selector p.tipo1 que el selector p .tipo1, el segundo es un selector descendente

```

/*Selector de etiqueta de clase*/
p.clase{
    color: ■ red;
}
/*Selector descendente*/
p .clase{
    font-family: 'Times New Roman', Times, serif;
}

```

### Ejemplo

El selector selecciona únicamente a las etiquetas p que lleven el nombre de clase parrafoModificar.

```

p.parrafoModificar{
    border: 1px solid ■ red;
}

```

Este selector se puede aplicar con el atributo id.

```

p#parrafoModificar{
    border: 1px solid ■ red;
    font-family: 'Times New Roman', Times, serif;
}

```

## 1.2 Propiedades de Textos y Fuentes

El texto es uno de los componentes más importantes de una página web, por lo tanto, su presentación debe ser clara, legible y coherente con el diseño general.

### 1.2.1 Principales Propiedades de Texto y Fuentes

- font-family: define la familia tipográfica.
- font-size: tamaño del texto.
- font-style: estilo de fuente (normal, cursiva, etc.).
- font-weight: grosor del texto.

- `line-height`: espacio entre líneas.
- `text-align`: alineación del texto.
- `text-transform`: transforma el texto (mayúsculas, minúsculas).
- `letter-spacing`: espacio entre letras.
- `word-spacing`: espacio entre palabras.

### 1.2.2 Ejemplo

```
p {  
  font-family: 'Helvetica', sans-serif;  
  font-size: 18px;  
  line-height: 1.6;  
  text-align: justify;  
  font-weight: 400;  
}
```

Un buen manejo de estas propiedades mejora la experiencia del usuario, haciendo que el contenido sea más agradable y fácil de leer.

## 1.3 Modelo de Caja (Box Model)

Todo elemento HTML se comporta como una **caja rectangular** compuesta por varias capas que definen cómo se posiciona y ocupa espacio en la página. Este modelo es clave para controlar el diseño y la distribución de los elementos.

### 1.3.1 Partes del Box Model

- **Content (Contenido)**: el área donde se muestra el texto o los elementos hijos.
- **Padding**: espacio entre el contenido y el borde del elemento.
- **Border**: borde que rodea el padding (opcional).
- **Margin**: espacio exterior entre el borde del elemento y otros elementos cercanos.

### 1.3.2 Ejemplo Gráfico (Descripción)

```
[ Margin ]  
[ [ Border ]  
[ [ [ Padding ]  
[ [ [ [ Content ] ] ] ]
```

### 1.3.3 Ejemplo Práctico

```
.box {  
  width: 300px;  
  padding: 20px;  
  border: 5px solid black;  
  margin: 30px;  
}
```

Este modelo permite entender cómo se calcula el tamaño final de un elemento en la página.

## 1.4 Colores y Fondos

CSS permite aplicar colores de diversas formas y añadir fondos a los elementos para mejorar el diseño visual.

### 1.4.1 Formas de Definir los Colores

- Por nombre: red, blue, green, etc.
- Hexadecimal: #ff0000, #00ff00
- RGB: rgb(255, 0, 0)
- RGBA: rgba(255, 0, 0, 0.5) (con opacidad)

### 1.4.2 Propiedades para Fondos

- background-color: color de fondo.
- background-image: imagen de fondo.
- background-repeat: repetición de la imagen.



- background-size: escala de la imagen.
- background-position: posición del fondo.

### 1.4.3 Ejemplo

```
body {  
  background-color: #f0f0f0;  
  background-image: url('fondo.jpg');  
  background-repeat: no-repeat;  
  background-size: cover;  
}
```

## 1.5 Unidades de Medida

CSS utiliza distintas unidades para definir dimensiones. Estas se dividen en:

### 1.5.1 Unidades Absolutas

- px: píxeles
- cm: centímetros
- in: pulgadas
- pt: puntos

### 1.5.2 Unidades Relativas

- %: porcentaje respecto al elemento contenedor
- em: relativo al tamaño de la fuente del elemento padre
- rem: relativo al tamaño de la fuente raíz (html)
- vw: ancho del viewport (pantalla)
- vh: alto del viewport

### 1.5.3 Ejemplo

El uso de unidades relativas mejora la adaptabilidad de las páginas a distintos dispositivos.

```
.container {  
  width: 80%;
```

```
font-size: 1.2rem;
height: 100vh;
}
```

## 1.6 Posicionamiento

La propiedad position permite cambiar el flujo normal de los elementos en la página.

### 1.6.1 Tipos de Posicionamiento

- static: valor por defecto; sigue el flujo del documento.
- relative: relativo a su posición original.
- absolute: se posiciona respecto al ancestro con position diferente de static.
- fixed: se mantiene fijo respecto a la ventana del navegador.
- sticky: se comporta como relative hasta que se alcanza un punto, luego actúa como fixed.

### 1.6.2 Ejemplo

Este posicionamiento es útil para mantener encabezados visibles durante el scroll.

```
header {
  position: sticky;
  top: 0;
  background-color: white;
  z-index: 10;
}
```

## 1.7 Flexbox (Diseño Flexibles)

**Flexbox** es un modelo de diseño unidimensional que permite distribuir espacio y alinear elementos de forma eficiente en una fila o columna, adaptándose fácilmente a distintos tamaños de pantalla.

Para utilizar Flexbox, se debe establecer el valor display: flex en el contenedor padre. A partir de allí, sus elementos hijos se convierten en ítems flexibles que pueden alinearse y distribuirse según diferentes propiedades.

### 1.7.1 Propiedades del Contenedor (Padre)

- display: flex;

- flex-direction: dirección de los elementos (row, column, row-reverse, column-reverse)
- justify-content: alinea horizontalmente (flex-start, center, space-between, etc.)
- align-items: alinea verticalmente (stretch, center, flex-start, etc.)
- flex-wrap: permite que los ítems se ajusten a múltiples líneas

### 1.7.2 *Propiedades del Ítem (Hijo)*

- flex-grow: define cuánto puede crecer
- flex-shrink: define cuánto puede reducirse
- flex-basis: tamaño inicial
- align-self: alineación individual

### 1.7.3 *Ejemplo Básico*

Flexbox es ideal para diseños lineales y adaptativos, como barras de navegación o secciones horizontales.

```
.contenedor {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}
```

## 1.8 Grid Layout (Diseño de Cuadrícula)

**Grid Layout** es un sistema de diseño bidimensional, es decir, permite organizar elementos en filas y columnas al mismo tiempo, ofreciendo un control total sobre la estructura de la página.

### 1.8.1 *Propiedades Principales del Contenedor (Grid)*

- display: grid;
- grid-template-columns: define el número y tamaño de columnas
- grid-template-rows: define el número y tamaño de filas

- **gap:** establece espacio entre filas y/o columnas
- **grid-column** y **grid-row:** controlan la posición del elemento dentro del grid

### 1.8.2 Ejemplo Básico

### 1.8.3 CSS

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  gap: 20px;  
}
```

### 1.8.4 HTML

```
<div class="grid">  
  <div>Elemento 1</div>  
  <div>Elemento 2</div>  
  <div>Elemento 3</div>  
</div>
```

Grid es perfecto para construir diseños complejos, como galerías, dashboards o páginas con estructuras variadas.

## 1.9 Pseudo-Clases y Pseudo-Elementos

CSS ofrece herramientas para aplicar estilos condicionales o específicos usando **pseudo-clases** y **pseudo-elementos**.

### 1.9.1 Pseudo-Clases

Aplican estilos según el estado o posición del elemento.

- **:hover:** cuando el usuario pasa el mouse
- **:focus:** cuando el elemento está enfocado (por ejemplo, un input activo)
- **:nth-child(n):** selecciona el hijo número  $n$

### 1.9.2 Ejemplo

```
a:hover {  
  color: blue;
```

```
text-decoration: underline;
}
```

### 1.9.3 Pseudo-Elementos

Permiten aplicar estilos a partes específicas del contenido del elemento.

- **::before**: inserta contenido antes del elemento
- **::after**: inserta contenido después
- **::first-letter**: estiliza la primera letra
- **::first-line**: estiliza la primera línea

### 1.9.4 Ejemplo

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

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

Estas herramientas mejoran la interactividad y el detalle visual sin necesidad de modificar el HTML.

## 1.10 Transiciones y Animaciones Básicas

CSS no solo se usa para dar estilo estático: también permite añadir **efectos dinámicos** como transiciones suaves o animaciones personalizadas, enriqueciendo la experiencia del usuario sin necesidad de JavaScript.

### 1.10.1 Transiciones

Permiten que los cambios de una propiedad ocurran de forma progresiva en lugar de instantánea.

### 1.10.2 Propiedades Comunes

- **transition-property**: la propiedad a animar
- **transition-duration**: duración del efecto

- **transition-timing-function:** curva de aceleración (ease, linear, etc.)
- **transition-delay:** retardo en comenzar

### 1.10.3 Ejemplo

```
button {  
  background-color: green;  
  transition: background-color 0.5s ease;  
}  
  
button:hover {  
  background-color: darkgreen;  
}
```

### 1.10.4 Animaciones

Permiten definir una serie de etapas o fotogramas (keyframes) que describen cómo debe evolucionar una propiedad.

### 1.10.5 Ejemplo

```
@keyframes aparecer {  
  from { opacity: 0; }  
  to { opacity: 1; }  
}  
  
.caja {  
  animation: aparecer 1s ease-in-out;  
}
```

Con estos efectos, los sitios web pueden sentirse más modernos, suaves y profesionales.

## Conclusión

La comprensión de los selectores CSS y de sus diversos tipos ha demostrado ser fundamental para aplicar estilos de forma precisa y eficiente sobre los elementos HTML. Cada tipo de selector básico, combinador, pseudo-clase o pseudo-elemento ofrece diferentes niveles de especificidad, permitiendo construir estructuras de estilo organizadas, escalables y adaptadas a las necesidades de cualquier proyecto web.

El análisis de la especificidad permitió entender su papel crucial en la resolución de conflictos de estilo en CSS. La correcta interpretación y aplicación de las reglas de especificidad asegura que los estilos se apliquen de manera coherente, evitando ambigüedades y reduciendo la dependencia de prácticas poco recomendables.

La adecuada utilización de las propiedades de texto, fuentes, fondos y colores refuerza la estética, la legibilidad y la comunicación visual en el diseño web. El manejo correcto de estos aspectos no solo embellece las interfaces, sino que también promueve la accesibilidad y mejora la experiencia del usuario en diversas plataformas.

El entendimiento del Modelo de Caja (Box Model) ha sido esencial para abordar el diseño y distribución de elementos en la web. El conocimiento de cómo interactúan el contenido, el padding, el borde y el margen permite controlar el espacio y las relaciones entre elementos, asegurando composiciones visuales ordenadas y armónicas.

La diferenciación entre unidades absolutas y relativas en CSS ha evidenciado su importancia en la creación de diseños tanto precisos como flexibles. Mientras las unidades absolutas ofrecen medidas fijas para casos específicos, las relativas permiten una mejor adaptación a diferentes dispositivos y resoluciones, favoreciendo la creación de sitios web responsivos.

El estudio de las técnicas de posicionamiento en CSS ha permitido comprender las distintas maneras en que los elementos pueden ubicarse dentro del flujo del documento. Cada valor aporta soluciones específicas para escenarios particulares, contribuyendo a un diseño dinámico, organizado y coherente con las necesidades funcionales del proyecto web.

La implementación de los sistemas de diseño Flexbox y Grid ha evidenciado su eficacia para construir distribuciones responsivas y flexibles. Estos métodos facilitan el diseño de

interfaces adaptativas que responden adecuadamente a diferentes tamaños de pantalla, mejorando la experiencia del usuario y optimizando el rendimiento del desarrollo.

La utilización de pseudo-clases y pseudo-elementos en CSS ha permitido enriquecer las posibilidades de estilización dinámica sin necesidad de alterar la estructura HTML. Estos recursos ofrecen alternativas eficientes para mejorar la interacción y la apariencia visual, aportando un valor añadido a la usabilidad y el atractivo de los sitios web.

La incorporación de transiciones y animaciones básicas ha demostrado que es posible agregar dinamismo y fluidez a la navegación web, elevando la calidad estética y mejorando la interacción del usuario. Su uso adecuado contribuye a construir experiencias digitales más atractivas, modernas y profesionales.

La profundización en el concepto de especificidad reafirma su relevancia como principio central en el manejo de estilos en CSS. Una correcta gestión de la especificidad es indispensable para mantener el orden en las hojas de estilo, asegurar el funcionamiento esperado del diseño y facilitar el mantenimiento a largo plazo del proyecto.



## Referencias Bibliográficas

*Primeros pasos en CSS / MDN.* (2025, March 27). MDN Web Docs.

[https://developer.mozilla.org/es/docs/conflicting/Learn\\_web\\_development/Core/Styling\\_basics](https://developer.mozilla.org/es/docs/conflicting/Learn_web_development/Core/Styling_basics)

*W3Schools.com.* (n.d.). <https://www.w3schools.com/css/>

Coyier, C. (2024, August 12). *CSS Flexbox Layout Guide / CSS-Tricks.* CSS-Tricks.

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

*CSS grid layout - CSS: Cascading Style Sheets / MDN.* (2025, February 7). MDN Web

Docs. [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_grid\\_layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout)