UNIVERSIDAD NACIONAL DE ASUNCIÓN

FACULTAD POLITÉCNICA

MAESTRÍA EN INGENIERÍA EN ELECTRÓNICA CON ÉNFASIS EN TECNOLOGÍA DE LA INFORMACIÓN

MÓDULO: SISTEMA DE INFORMACIÓN WEB

“Tarea 02: Explorando CSS”

Alumno:

- **Oscar Aureliano Caballero Mendoza**

Profesor:

- **Dr. Julio César Mello Román**

16/03/2025

ÍNDICE

Introducción.

El presente reporte técnico documenta el proceso de investigación llevado a cabo para completar la tarea asignada. Se describen los objetivos planteados, la metodología aplicada, las referencias bibliográficas utilizadas para fundamentar el trabajo y los ejercicios prácticos desarrollados se adjuntan en un archivo comprimido para su revisión.

Objetivos.

El objetivo es documentar el proceso de investigación, analizar los datos obtenidos, presentar los hallazgos más importantes y llegar a conclusiones basadas en esos datos. Los resultados proporcionan una visión clara de las mejores prácticas y tendencias actuales en CSS.

Metodología.

La metodología empleada se basa en diversas técnicas de exploración, que incluyen:

- **Investigación sobre técnicas de combinación de selectores de CSS y la utilización de herencia:** Profundizar en el conocimiento de cómo se pueden combinar selectores CSS para lograr estilos más eficientes y reutilizables, y cómo funciona la herencia en CSS.

- **Análisis de la prevalencia entre selectores CSS:** Estudiar qué selectores CSS son más utilizados y en qué contextos, para comprender las mejores prácticas y tendencias actuales.

- **Exploración de frameworks de CSS:** Investigar y analizar diferentes frameworks de CSS, como Bootstrap, Tailwind CSS o Materialize CSS, para evaluar sus ventajas y desventajas.

- **Desarrollo de ejemplos utilizando animaciones CSS sin uso de JavaScript:** Crear ejemplos prácticos de animaciones CSS para demostrar las capacidades de CSS en la creación de efectos visuales interactivos sin necesidad de JavaScript.

1. Combinación de selectores de CSS y la utilización de herencia.

1.1 Combinación de selectores de CSS.

La combinación de selectores en CSS permite estilizar elementos HTML de manera precisa y eficiente. Existen diversas técnicas para combinar selectores, las cuales pueden mejorar la especificidad y modularidad de los estilos.

A continuación, se presentan las principales formas de combinar selectores en CSS:

1.1.1 Combinadores de Selectores.

Los combinadores permiten definir relaciones entre selectores y especificar cómo deben aplicarse los estilos a los elementos dentro del DOM.

a) Combinador de Descendencia ( ): Aplica estilos a los elementos que están dentro de otro elemento (descendientes en cualquier nivel).

Ejemplo: section p { color: blue; }

Este selector aplica el color azul a todos los p dentro de un section.

b) Combinador de Hijo (>): Selecciona solo los elementos que son hijos directos de otro elemento.

Ejemplo: section > p { color: red; }

Este selector aplica el color rojo solo a los p que son hijos directos de section, ignorando los p anidados en otros contenedores.

c) Combinador de Hermano Adyacente (+): Selecciona un elemento que esté inmediatamente después de otro elemento.

Ejemplo: h1 + p { font-weight: bold; }

Este selector aplica negrita al p que sigue inmediatamente a un h1.

d) Combinador de Hermanos Generales (~): Selecciona todos los elementos hermanos que siguen a un elemento especificado.

Ejemplo: h1 ~ p { font-style: italic; }

Este selector aplica estilo cursiva a todos los p que siguen a un h1, sin importar si están directamente después o no.

1.1.2 Agrupación de Selectores

Permite aplicar un mismo conjunto de reglas a varios selectores simultáneamente.

Ejemplo: h1, h2, h3 { text-align: center; }

Este selector centra el texto de h1, h2 y h3.

1.1.3 Selectores Avanzados con Atributos

Permiten seleccionar elementos en función de sus atributos y valores.

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

Este selector cambia el color de los enlaces que tienen el atributo target="_blank".

1.1.4 Selectores Pseudo-clases y Pseudo-elementos

Las pseudo-clases permiten seleccionar elementos en un estado específico, mientras que los pseudo-elementos permiten estilizar partes de un elemento.

Ejemplo: a:hover { text-decoration: underline; }

Este selector subraya los enlaces cuando el usuario pasa el cursor sobre ellos.

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

Este selector aplica negrita a la primera línea de un párrafo.

1.2 Herencia.

La herencia en CSS es un mecanismo que permite que ciertos estilos aplicados a un elemento padre se transmitan automáticamente a sus elementos hijos. Este comportamiento facilita el mantenimiento del código, ya que evita la repetición innecesaria.

Sin embargo, no todas las propiedades CSS son heredadas por defecto, solo aquellas relacionadas con el texto.

1.2.1 Propiedades heredables y no heredables

a) **Propiedades heredables:** Estas propiedades se transmiten automáticamente de un elemento padre a sus hijos.
   Ejemplos comunes:
   - **color**: Define el color del texto.
   - **font-family, font-size, font-weight**: Propiedades relacionadas con la tipografía.
   - **text-align, text-indent**: Propiedades relacionadas con la alineación y sangría del texto.

- **line-height**: Define el espacio entre líneas.

b) **Propiedades No Heredables:** Estas propiedades no se transmiten automáticamente a los elementos hijos. Si deseas que un elemento hijo tenga el mismo estilo, debes aplicarlo directamente.

Ejemplos comunes:

- **border:** Define el borde de un elemento.
- **margin:** Define el margen de un elemento.
- **padding:** Define el relleno de un elemento.
- **background:** Define el fondo de un elemento.
- **display:** Define como se muestra un elemento.
- **width y height:** Definen el ancho y el alto de un elemento.

1.2.2 Control de la herencia en CSS.

CSS permite modificar el comportamiento de la herencia mediante los valores especiales **inherit**, **initial** y **unset**.

a) **inherit (Forzar la herencia):** Este valor obliga a un elemento a heredar explícitamente el valor de una propiedad de su elemento padre.

Cuando usarlo: Cuando quieres asegurarte de que un elemento hijo tenga exactamente el mismo estilo que su padre, incluso si otros estilos intentan cambiarlo.

Ejemplo:

```
<p style="color: blue;">Este es un párrafo. <button style="color: inherit;">Botón</button></p>
```

En este caso, el botón tomará el color azul del párrafo, ya que se está forzando a heredar ese valor.

b) **initial (Restablecer al valor por defecto de CSS):** Este valor restablece una propiedad a su valor predeterminado según la especificación CSS.

Cuando usarlo: Cuando quieres eliminar cualquier estilo heredado y volver al comportamiento predeterminado del navegador.

Ejemplo:

```
<p style="color: red;">Este es un párrafo. <button style="color: initial;">Botón</button></p>
```

El botón volverá a su color de texto predeterminado (normalmente negro), ignorando el color rojo del párrafo.

c) **unset (Combinación de inherity initial):** Este valor actúa de manera diferente dependiendo de si la propiedad es heredable o no:

- Si la propiedad es heredable, se comporta como **inherit**.

- Si la propiedad no es heredable, se comporta como **initial**.

Cuando usarlo: Cuando no estás seguro de si una propiedad es heredable y quieres un comportamiento consistente.
Ejemplo:
Este es un párrafo. <button style="color: unset;">Botón</button>

En este caso, el color es una propiedad heredable, por lo tanto, el botón tomara el color verde del párrafo.

Pero si la propiedad fuese border, el botón no heredaría ningún valor, ya que border no es heredable.

En resumen:
- **inherit:** Fuerza la herencia del valor del padre.
- **initial:** Restablece al valor predeterminado de la propiedad.
- **unset:** Actúa como inherit para propiedades heredables y como initial para propiedades no heredables.

1.2.3 Ventajas de la herencia en CSS

La herencia en CSS ofrece diversas ventajas que facilitan y optimizan el desarrollo web que son:
- **Menos código:** Evita repetir estilos, haciendo el código más limpio y fácil de mantener.
- **Consistencia:** Asegura que los elementos tengan un aspecto uniforme en toda la página.
- **Flexibilidad:** Permite controlar qué estilos se heredan y cuáles no.
- **Rendimiento:** Hojas de estilo más pequeñas se cargan más rápido, mejorando la velocidad del sitio.

2. Las prevalencias entre selectores CSS.

La prevalencia o especificidad de los selectores CSS determina qué regla de estilo se aplica a un elemento cuando hay múltiples reglas en conflicto. Cuando dos o más reglas apuntan al mismo elemento y definen estilos para la misma propiedad, la regla con mayor especificidad será la que se aplique.

2.1 ¿Qué es la Especificidad?

La especificidad es un sistema que calcula el "peso" de un selector CSS. Cuanto más específico sea un selector, mayor será su peso y, por lo tanto, prevalecerá sobre selectores menos específicos. Esto permite a los desarrolladores controlar con precisión qué estilos se aplican a los elementos de una página.

2.2 Jerarquía de Especificidad.

La jerarquía de especificidad en CSS es un conjunto de reglas que el navegador sigue para determinar qué estilos se aplican a un elemento cuando hay múltiples reglas en conflicto. Es esencial comprender esta jerarquía para controlar con precisión el aspecto de tus páginas web.

Aquí te presento un desglose de la jerarquía de especificidad, desde la menor a la mayor:

a) **Selectores universales (*) y combinadores (+, >, ~, 'espacio'):** Tienen la menor especificidad.

b) **Selectores de tipo (elementos), pseudo-elementos (::) y pseudo-clases (:)**: Aumentan la especificidad.

c) **Selectores de clase (.) y selectores de atributo ([])**: Tienen mayor especificidad.

d) **Selectores de ID (#)**: Tienen la mayor especificidad.

e) **Estilos en línea (style="...")**: Tienen la mayor especificidad de todas, superando incluso a los selectores de ID.

f) **!important**: Esta regla, puesta despues de un valor css, sobreescribe cualquier otra regla de especificidad, por lo tanto debe de ser usada con mucha precaución.

2.3 Cálculo de Especificidad.

La especificidad en CSS se calcula para determinar qué regla de estilo se aplica a un elemento cuando hay múltiples reglas en conflicto. Es un sistema de "peso" que asigna valores a los selectores CSS, y el selector con el valor más alto tiene prioridad.

La especificidad se representa mediante cuatro valores: (a, b, c, d).

- **a:** Número de selectores de ID.
- **b:** Número de selectores de clase, selectores de atributo y pseudo-clases.
- **c:** Número de selectores de tipo y pseudo-elementos.
- **d:** Selectores universales.

El selector con el valor más alto en cualquiera de estas partes tiene mayor especificidad.

Ejemplos

- #mi-id (0, 1, 0, 0) > .mi-clase (0, 0, 1, 0) > div (0, 0, 0, 1) > * (0, 0, 0, 0)

  Este ejemplo ilustra la jerarquía de especificidad.

  #mi-id tiene la mayor especificidad porque es un selector de ID (0, 1, 0, 0).

.mi-clase tiene una especificidad menor porque es un selector de clase (0, 0, 1, 0).

div tiene aún menos especificidad porque es un selector de tipo (0, 0, 0, 1).

***** tiene la menor especificidad de todas porque es el selector universal (0, 0, 0, 0).

Por lo tanto, si estas cuatro reglas CSS se aplicaran al mismo elemento, la regla #mi-id prevalecería.

- style="color:red;" > #mi-id

  Este ejemplo muestra que los estilos en línea tienen mayor especificidad que los selectores de ID. Un estilo definido directamente en el atributo style de un elemento siempre anulará cualquier regla definida en una hoja de estilo externa, incluso si esa regla utiliza un selector de ID.

- p {color:blue !important;} > p {color:red;}

  La regla p {color:blue !important;} anulará la regla p {color:red;}.
  !important se utiliza para dar la máxima prioridad a una regla CSS, pero se debe usar con precaución porque puede dificultar el mantenimiento del código.

3. Framework de CSS.

Los frameworks de CSS son bibliotecas de código preescrito que proporcionan una estructura básica para el diseño de sitios web.

En términos más técnicos, son bibliotecas de código CSS preescrito que ofrecen:

- **Estructura:** Definen cómo se organizará el contenido en tu página web, generalmente a través de sistemas de cuadrículas (grids).
- **Estilos:** Proporcionan estilos visuales predefinidos para elementos comunes como botones, formularios, tablas, etc.
- **Componentes:** Ofrecen elementos de interfaz de usuario listos para usar, como menús de navegación, modales y carruseles.
- **Utilidades:** Incluyen clases CSS que simplifican tareas de estilo comunes, como agregar márgenes, cambiar colores o ajustar la tipografía.

3.1 Ventajas de usar frameworks de CSS:

- **Desarrollo más rápido:** Los frameworks de CSS permiten crear diseños complejos de forma rápida y sencilla.
- **Consistencia:** Garantizan que el diseño sea coherente en todo el sitio web.

- **Adaptabilidad:** Muchos frameworks incluyen sistemas de rejilla adaptables que funcionan bien en diferentes dispositivos.

- **Accesibilidad:** Algunos frameworks incluyen características que mejoran la accesibilidad de los sitios web.

3.2 Frameworks de CSS populares.

- **Bootstrap:** Uno de los frameworks de CSS más populares, conocido por su sistema de rejilla adaptable y su amplia gama de componentes.

  Ejemplo: <button class="btn btn-primary">Botón Primario</button>

  Este código crea un botón con el estilo primario definido por el framework CSS que se esté utilizando. Es muy probable que sea un botón azul con texto blanco.

- **Tailwind CSS:** Un framework de utilidades que permite crear diseños personalizados de forma rápida y sencilla.

  Ejemplo:
  <button class="bg-blue-500 text-white py-2 px-4 rounded">Botón Personalizado</button>

  Este código crea un botón azul con texto blanco, relleno y esquinas redondeadas.

- **Materialize:** Un framework basado en Material Design, el lenguaje de diseño de Google.

  Ejemplo:

```
<div class="card">
        <div class="card-content">
                <span class="card-title">Título de la Tarjeta</span>
                <p>Contenido de la tarjeta.</p>
        </div>
 </div>
```

  Este código genera una tarjeta con sombra y una estructura organizada, siguiendo los principios de Material Design.

- **Foundation:** Un framework adaptable y flexible. Al igual que Bootstrap que ofrece un sistema de cuadricula adaptable y una amplia gama de características.

- **Ejemplo:**

```
<div class="grid-x grid-margin-x">
        <div class="cell small-12 medium-6 large-4">
                <p>Contenido de la celda.</p>
        </div>
        <div class="cell small-12 medium-6 large-8">
```

```
                <p>Otra celda con contenido.</p>
            </div>
</div>
```

Este código muestra la creación de una rejilla con tres tamaños de pantalla diferentes, para que el contenido se vea de forma óptima en cualquier dispositivo.

Bibliografìa.

- **J. D Gauchat (2017).** El gran libro de HTML5, CSS3 y JavaScrit, 3ª Edicion. Editorial Marcombo, S.A.
- **Arnaldo Pérez Castaño (2015).** HTML Y CSS FÁCIL. Editorial Marcombo, S.A.
- **Diego C Martín.** Uso de Selectores y Herencias en CSS. https://www.diegocmartin.com/selectores-herencia-css/
- **Edgar D'Andrea.** CSS paso a paso.