Tarea 02: Explorando CSS ---------- Aldo Reyes Mella

Instrucciones

- Preparar un reporte técnico sobre las exploraciones realizadas. Agregar referencias utilizadas al reporte y levantar con los ejercicios desarrollados en un archivo comprimido denominado nombre_apellido_tarea02.zip

Desarrollar las siguientes actividades

1. Investigar técnicas para combinar selectores de CSS y la utilización de herencias.
2. Explicar las prevalencias entre selectores CSS.
3. Investigar sobre framework de CSS
4. Realizar tres ejemplos utilizando animaciones con CSS sin javascript

Desarrollo

Introducción

Este reporte tiene como objetivo documentar las exploraciones realizadas en CSS, enfocadas en la combinación de selectores, herencia, prevalencia de reglas, frameworks y animaciones sin el uso de JavaScript. A lo largo del documento se presentarán ejemplos prácticos para ilustrar los conceptos.

1. Investigación sobre técnicas para combinar selectores de CSS y la utilización de herencias

Selectores combinados en CSS:

Los selectores en CSS pueden combinarse para aplicar estilos de manera más eficiente y precisa. Algunas de las técnicas incluyen:

Selector descendente (A B): Aplica estilos a los elementos B que están dentro de A.

```
div p {
   color: blue;
}
```

Aplica color azul a todos los <p> dentro de un .

Selector hijo directo (A > B): Aplica estilos solo a los hijos directos de A.

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

Solo afecta a los <p> que son hijos directos de .

Selector adyacente (A + B): Aplica estilos a B si está inmediatamente después de A.

```
h1 + p {
   font-size: 18px;
}
```

Cambia el tamaño del primer <p> que sigue inmediatamente a un <h1>.

Selector de hermanos generales (A ~ B): Aplica estilos a todos los elementos B que están después de A, aunque no sean inmediatos.

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

Todos los <p> después de un <h1> se mostrarán en cursiva.

Selector de atributo ([atributo]): Aplica estilos a elementos con un atributo específico.

```
input[type="text"] {
   border: 1px solid black;
}
```
Aplica un borde a los input de tipo texto.

Herencia en CSS:

La herencia permite que algunos estilos se transmitan de un elemento padre a sus hijos. Las propiedades que pueden heredarse incluyen:

- color
- font
- visibility

Sin embargo, otras propiedades como margin, padding, border, y background **no** se heredan por defecto. Para forzar la herencia, se puede usar inherit:

```
p {
   color: inherit;
}
```
Esto hace que el color del texto de <p> sea el mismo que el de su elemento padre.

2. Explicación de las prevalencias entre selectores CSS

Cuando hay reglas CSS en conflicto, se aplican según su especificidad. La prioridad sigue esta jerarquía:

1. **Estilos en línea** (style="") tienen la más alta prioridad.
2. **Selectores ID** (#miElemento) tienen más prioridad que las clases.
3. **Selectores de clase, atributo y pseudo-clases** (.miClase, [type="text"], :hover) tienen prioridad sobre los selectores de etiquetas.
4. **Selectores de etiqueta** (h1, p, div) tienen la menor prioridad.

Si hay reglas con la misma especificidad, la última declaración en la hoja de estilos se aplica.
El uso de !important anula cualquier regla, pero debe usarse con moderación:

```
p {
   color: blue !important;
}
```
Esto hará que todos los <p> sean azules, sin importar otras reglas aplicadas.

3. Investigación sobre frameworks de CSS

Un framework CSS es un conjunto de estilos predefinidos y herramientas que facilitan el desarrollo de interfaces. Algunos de los más populares incluyen:

- **Bootstrap:** Proporciona un sistema de cuadrícula, componentes reutilizables y clases para diseño responsive.
- **Tailwind CSS:** Permite el diseño con clases utilitarias sin necesidad de escribir CSS personalizado.
- **Foundation:** Similar a Bootstrap, pero con un enfoque en accesibilidad y diseño adaptable.
- **Bulma:** Basado en flexbox, facilita el diseño responsivo con una sintaxis sencilla.

Ejemplo con Bootstrap:

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
<button class="btn btn-primary">Botón Bootstrap</button>
```

4. Ejemplos de animaciones con CSS

Ejemplo 1: Animación de cambio de color

```
@keyframes cambioColor {
   0% { background-color: red; }
   50% { background-color: blue; }
   100% { background-color: red; }
}
```

```
.caja {
   width: 100px;
   height: 100px;
   background-color: red;
   animation: cambioColor 3s infinite;
}
```

Ejemplo 2: Animación de desplazamiento

```
@keyframes mover {
   0% { transform: translateX(0); }
   100% { transform: translateX(300px); }
}

.caja {
   width: 100px;
   height: 100px;
   background-color: green;
   animation: mover 2s infinite alternate;
}
```

Ejemplo 3: Efecto de opacidad

```
@keyframes fadeInOut {
   0% { opacity: 0; }
   50% { opacity: 1; }
   100% { opacity: 0; }
}

.texto {
   font-size: 24px;
   color: black;
   animation: fadeInOut 3s infinite;
}
```

Para implementar estos ejemplos en HTML:

```
<div class="caja"></div>
<p class="texto">Texto animado</p>
```

Estos ejemplos ilustran diferentes efectos usando solo CSS sin JavaScript.

```
<!DOCTYPE html>
<html lang="es">
<head>
      <title>Animaciones con CSS</title>
   <link rel="stylesheet" href="animaciones.css">
</head>
<body>
   <header>
      <h1>Ejemplos de Animaciones con CSS</h1>
   </header>

   <section>
```

```
    <h2>Animación de Cambio de Color</h2>
    <div class="caja color"></div>
  </section>

  <section>
    <h2>Animación de Desplazamiento</h2>
    <div class="caja mover"></div>
  </section>

  <section>
    <h2>Animación de Opacidad</h2>
    <p class="texto">Este texto aparece y desaparece</p>
  </section>

  <footer>
    <p>&copy; 2025 Animaciones con CSS. Todos los derechos reservados.</p>
  </footer>
</body>
</html>
```

Archivo css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background-color: #f0f0f0;
  margin: 0;
  padding: 0;
}

header {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  font-size: 24px;
}

section {
```

```
    margin: 20px;
    padding: 20px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
    position: relative;
    bottom: 0;
    width: 100%;
}

.caja {
    width: 100px;
    height: 100px;
    margin: 20px auto;
}

/* Animación de Cambio de Color */
@keyframes cambioColor {
    0% { background-color: red; }
    50% { background-color: blue; }
    100% { background-color: red; }
}

.color {
    background-color: red;
    animation: cambioColor 3s infinite;
}

/* Animación de Desplazamiento */
@keyframes mover {
    0% { transform: translateX(0); }
```

```
    100% { transform: translateX(300px); }
}

.mover {
    background-color: green;
    animation: mover 2s infinite alternate;
}

/* Animación de Opacidad */
@keyframes fadeInOut {
    0% { opacity: 0; }
    50% { opacity: 1; }
    100% { opacity: 0; }
}

.texto {
    font-size: 24px;
    color: black;
    animation: fadeInOut 3s infinite;
}
```

Conclusión

El uso de selectores combinados en CSS permite estructurar y organizar mejor el código. La herencia optimiza la aplicación de estilos, mientras que la prevalencia de selectores ayuda a resolver conflictos en la hoja de estilos. Los frameworks CSS facilitan el desarrollo rápido y eficiente de interfaces. Finalmente, las animaciones CSS permiten agregar dinamismo sin depender de JavaScript.

Referencias

- Mozilla Developer Network (MDN). "CSS Selectors." Disponible en: https://developer.mozilla.org/es/docs/Web/CSS/CSS_Selectors
- W3Schools. "CSS Specificity." Disponible en: https://www.w3schools.com/css/css_specificity.asp
- Bootstrap Official Documentation. Disponible en: https://getbootstrap.com/
- Tailwind CSS Documentation. Disponible en: https://tailwindcss.com/docs