**Reporte técnico sobre las exploraciones realizadas**

**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**

**Alumno: Victor Duarte**

**C I.N: 4.725.315**

**1.** **Técnicas para combinar selectores de CSS y la utilización de herencias.**

Los selectores CSS son patrones que permiten seleccionar elementos HTML específicos para aplicarles estilos. Los diferentes tipos de selectores es esencial:

Selectores de tipo: Seleccionan elementos HTML por su nombre de etiqueta. Ejemplo: p (selecciona todos los párrafos). Se basan en la estructura semántica del HTML.

Selectores de clase: Seleccionan elementos con un atributo class específico. Ejemplo: .mi-clase (selecciona elementos con class="mi-clase"). Permiten la agrupación lógica de estilos para elementos similares.

Selectores de ID: Seleccionan un elemento único con un atributo id específico. Ejemplo: #mi-id (selecciona el elemento con id="mi-id"). Proporcionan una forma de aplicar estilos únicos a elementos individuales.

Selectores de atributo: Seleccionan elementos basados en la presencia o valor de sus atributos. Ejemplo: [type="text"] (selecciona elementos con type="text"). Permiten estilos basados en metadatos HTML.

Pseudoclases: Seleccionan elementos en función de su estado o posición. Ejemplo: hover (selecciona un elemento cuando el mouse está sobre él). Permite estilos dinámicos basados en la interacción del usuario.

Pseudoelementos: Seleccionan partes específicas de un elemento. Ejemplo: before (inserta contenido antes del contenido del elemento). Permiten estilos detallados de partes específicas de los elementos.

**Combinación de selectores: Precisión en el estilo**

Los combinadores permiten crear selectores más específicos y complejos:

Combinador descendiente (espacio): Selecciona elementos que son descendientes de otro elemento. Ejemplo: div p (selecciona todos los párrafos dentro de los divs). Se basa en la jerarquía del DOM (Document Object Model).

Combinador hijo (>): Selecciona elementos que son hijos directos de otro elemento. Ejemplo: div > p (selecciona solo los párrafos que son hijos directos de los divs). Proporciona un control más estricto sobre la jerarquía del DOM.

Combinador hermano adyacente (+): Selecciona el elemento que sigue inmediatamente a otro elemento hermano. Ejemplo: h1 + p (selecciona el primer párrafo que sigue a un h1). Permite estilos basados en la proximidad de los elementos.

Combinador hermano general (~): Selecciona todos los elementos hermanos que siguen a otro elemento. Ejemplo: h1 ~ p (selecciona todos los párrafos que siguen a un h1). Proporciona una selección más amplia de elementos hermanos.

**Herencia: Simplificación y eficiencia**

La herencia es un mecanismo poderoso que permite que ciertos estilos se propaguen de elementos padres a hijos:

Propiedades heredables: Algunas propiedades CSS se heredan de forma predeterminada (p. ej., font-family, color, text-align). Reduce la repetición de código y promueve la coherencia del estilo.

Propiedades no heredables: Otras propiedades no se heredan (p. ej., border, margin, padding). Evita la propagación no deseada de estilos que pueden afectar el diseño.

Control de la herencia: La palabra clave inherit permite forzar la herencia de una propiedad. La palabra clave initial permite establecer una propiedad a su valor inicial. La palabra clave unset permite quitar el valor asignado y si tiene valor heredado lo heredará, caso contrario lo establecerá a su valor inicial.

Proporcionan un control preciso sobre el comportamiento de la herencia.

**Especificidad: Resolución de conflictos**

La especificidad determina qué regla CSS se aplica cuando hay múltiples reglas en conflicto:

Cálculo de la especificidad: Los selectores de ID tienen la mayor especificidad, seguidos de las clases, los atributos y las pseudoclases, y luego los selectores de tipo y los pseudoelementos. Establece una jerarquía de importancia para las reglas CSS.

La cascada: La cascada CSS determina el orden en que se aplican las reglas CSS, teniendo en cuenta la especificidad y el orden de aparición. Permite la combinación y el ordenamiento de estilos de múltiples fuentes.

**Metodologías y mejores prácticas**

BEM (Block Element Modifier): Una metodología para crear nombres de clase CSS más organizados y mantenibles. Promueve el modularidad y la reutilización de estilos.

CSS Modules: Una técnica para encapsular estilos CSS dentro de componentes específicos. Evita conflictos de nombres de clase y mejora la organización del código.

CSS-in-JS: Una técnica para escribir estilos CSS directamente dentro de componentes JavaScript. Permite un control más dinámico y flexible de los estilos.

**2. Las prevalencias entre selectores CSS.**

En CSS, la prevalencia (o especificidad) determina qué reglas de estilo se aplican cuando hay conflictos entre múltiples reglas que afectan al mismo elemento. Es esencial comprender la prevalencia para controlar con precisión el estilo de tus páginas web. La prevalencia es un algoritmo que el navegador utiliza para calcular la importancia relativa de las reglas CSS. Cuando múltiples reglas de estilo compiten para aplicarse a un elemento, la regla con la mayor prevalencia "gana".

**Cálculo de la prevalencia**

La prevalencia se calcula basándose en cuatro categorías, a las que a veces se hace referencia como un sistema de valores:

Estilos en línea: Estilos definidos directamente en el atributo style de un elemento HTML IDs: Selectores que utilizan el atributo id (por ejemplo, #mi-id). Clases, atributos y pseudoclases: Selectores que utilizan el atributo class (por ejemplo, .mi-clase), selectores de atributos (por ejemplo, [type="text"]) y pseudoclases (por ejemplo: hover).

Tipos y pseudoelementos**:** Selectores de tipo (por ejemplo, p, div) y pseudoelementos (por ejemplo: before,:: after).

Selector universal: El selector universal (\*) tiene una prevalencia de 0. Herencia: Los estilos heredados tienen la menor prevalencia de todas.

**Reglas de prevalencia**

Estilos en línea: Tienen la máxima prevalencia. Cualquier estilo definido en línea anula todas las demás reglas.

IDs: Los selectores de ID tienen una alta prevalencia. Una regla con un selector de ID siempre anula las reglas con selectores de clases, atributos o tipos.

Clases, atributos y pseudoclases: Estos selectores tienen una prevalencia media. Una regla con un selector de clase anula las reglas con selectores de tipo.

Tipos y pseudoelementos: Estos selectores tienen la prevalencia más baja.

La cascada: Si dos reglas tienen la misma prevalencia, la última regla definida en la hoja de estilos tiene prioridad.

! important: La declaración ! important anula todas las demás reglas de prevalencia, excepto los estilos en línea. Sin embargo, se recomienda usar !important con moderación, ya que puede dificultar el mantenimiento del código.

**Ejemplos prácticos**

Un estilo definido con un selector de ID (#mi-id) siempre anulará un estilo definido con un selector de clase (.mi-clase).

Un estilo definido con un selector de clase (.mi-clase) siempre anulará un estilo definido con un selector de tipo (p).

Si dos reglas tienen la misma prevalencia, la que aparece más abajo en la hoja de estilos se aplicará.

**Consejos y buenas prácticas**

Evita el uso excesivo de selectores de ID.

Utiliza selectores de clase y atributos para un estilo más flexible y mantenible.

Comprende la prevalencia para evitar conflictos de estilo inesperados.

Utiliza herramientas de desarrollo del navegador para inspeccionar la prevalencia de las reglas de estilo.

**3. Framework de CSS**

Los frameworks de CSS son herramientas esenciales en el desarrollo web moderno. Proporcionan una base sólida y eficiente para estilizar páginas web, ahorrando tiempo y esfuerzo a los desarrolladores. Aspectos clave: Son bibliotecas de código CSS prescrito que ofrecen una colección de estilos, componentes y funcionalidades reutilizables. Simplifican el proceso de diseño, permitiendo crear interfaces de usuario consistentes y atractivas de manera más rápida. Ayudan a garantizar la compatibilidad entre diferentes navegadores y dispositivos.

**Ventajas de utilizar frameworks de CSS:**

Desarrollo más rápido: Los componentes y estilos predefinidos aceleran el proceso de diseño.

Consistencia: Aseguran un diseño uniforme en toda la página web.

Responsividad: Muchos frameworks incluyen sistemas de rejilla y componentes adaptables a diferentes tamaños de pantalla.

Mantenibilidad: La estructura organizada y modular facilita la actualización y el mantenimiento del código.

Compatibilidad: Ayudan a resolver problemas de compatibilidad entre navegadores.

**Tipos de frameworks de CSS:**

Existen varios tipos de frameworks, cada uno con sus propias características y enfoques:

Frameworks completos: Ofrecen una amplia gama de componentes y funcionalidades, como Bootstrap y Foundation.

Frameworks de utilidad: Se centran en proporcionar clases de utilidad para construir diseños personalizados, como Tailwind CSS.

Frameworks de diseño de materiales: Implementan los principios de diseño de materiales de Google, como Materialize.

**Ejemplos de frameworks populares:**

Bootstrap: Uno de los frameworks más populares, conocido por su sistema de rejilla y su amplia colección de componentes.

Tailwind CSS: Un framework de utilidad que permite crear diseños altamente personalizados mediante la combinación de clases predefinidas.

Materialize: Un framework basado en el diseño de materiales de Google, ideal para crear interfaces de usuario modernas y atractivas.

**Consideraciones al elegir un framework:**

Las necesidades del proyecto: Considera el tipo de proyecto y las funcionalidades requeridas.

La curva de aprendizaje: Algunos frameworks pueden ser más fáciles de aprender que otros.

La personalización: Evalúa la capacidad de personalización del framework para adaptarlo a tus necesidades.

El tamaño del framework: Algunos frameworks pueden ser más pesados que otros, lo que puede afectar el rendimiento de la página.

La comunidad: Es muy importante que el framework tenga una comunidad grande que pueda ayudar a resolver dudas.

En resumen, los frameworks de CSS son herramientas valiosas para el desarrollo web. Al elegir el framework adecuado, los desarrolladores pueden mejorar la eficiencia, la consistencia y la calidad de sus proyectos.

**4. Tres ejemplos utilizando animaciones con CSS sin javascript**

**Ejemplo 1: Desvanecimiento de entrada**

Este ejemplo muestra cómo hacer que un elemento aparezca gradualmente mediante una animación de desvanecimiento.

. desvanecer-entrada {

opacity: 0;

animation: desvanecer 1s ease-in-out forwards;

}

@keyframes desvanecer {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

**Descripción:**

* El elemento .desvanecer-entrada comienza con una opacidad de 0, haciéndolo invisible.
* La propiedad animation aplica la animación llamada desvanecer, que dura 1 segundo, utiliza una curva de tiempo ease-in-out (acelera y desacelera), y mantiene el estado final (forwards).
* El @keyframes desvanecer define la animación, cambiando la opacidad de 0 a 1.

**Ejemplo 2: Rotación continua**

Este ejemplo muestra cómo hacer que un elemento gire continuamente

.rotar-continuo {

animation: rotar 4s linear infinite;

}

@keyframes rotar {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

**Explicación****:**

* El elemento .rotar-continuo aplica la animación rotar, que dura 4 segundos, utiliza una curva de tiempo linear (velocidad constante), y se repite infinitamente (infinite).
* @keyframes rotar define la animación, rotando el elemento 360 grados.

**Ejemplo 3: Desplazamiento horizontal**

Este ejemplo muestra cómo hacer que un elemento se desplace horizontalmente de un lado a otro.

.desplazar-horizontal {

animation: desplazar 3s ease-in-out infinite alternate;

}

@keyframes desplazar {

from {

transform: translateX(-100px);

}

to {

transform: translateX(100px);

}

}

**Descripción:**

* + El elemento .desplazar-horizontal aplica la animación desplazar, que dura 3 segundos, utiliza una curva de tiempo ease-in-out, se repite infinitamente, y alterna la dirección en cada iteración (alternate).
  + @keyframes desplazar define la animación, desplazando el elemento 100 píxeles a la izquierda y luego 100 píxeles a la derecha.

Puntos clave:

@keyframes: Define la secuencia de la animación.

animation: Aplica la animación a un elemento.

Transition vs. animation:

* + transition anima cambios de propiedades cuando un estado cambia (por ejemplo, al pasar el ratón).
  + animation permite animaciones más complejas y controladas.

Curvas de tiempo: Controlan la velocidad de la animación (por ejemplo, ease, linear, ease-in-out).

Iteraciones: Controlan cuántas veces se repite la animación (infinite, números).

Dirección: Controla la dirección de la animación (alternate, reverse).