CONCEPTOS DE WEB RESPONSIVO

ESPINOSA MARTINEZ CARLOS LEONEL

Aarón Hernández García

1.1.1 Conceptos de Web Responsivo

16/01/2024

CSS (Cascading Style Sheets)

¿Qué es CSS?

CSS, o Cascading Style Sheets, es un lenguaje utilizado para describir la presentación de un documento escrito en HTML o XML. CSS controla la apariencia visual y el diseño de los elementos en una página web, separando el contenido de la estructura visual.

Importancia en el diseño web

CSS es fundamental en el diseño web por las siguientes razones:

- Separación de contenido y diseño: Facilita el mantenimiento y actualización del código al mantener el HTML para el contenido y el CSS para la apariencia.
- Consistencia: Permite aplicar estilos uniformes a múltiples páginas de un sitio web.
- Adaptabilidad: Hace que las páginas sean responsivas, permitiendo diseños que se ajusten a diferentes dispositivos y tamaños de pantalla.
- Eficiencia: Reduce el tamaño del código HTML al centralizar los estilos en archivos CSS externos.

TypeScript

¿Qué es TypeScript?

TypeScript es un lenguaje de programación desarrollado por Microsoft que extiende JavaScript añadiendo tipado estático y otras características avanzadas. Aunque el código TypeScript se compila a JavaScript para ejecutarse en navegadores o entornos como Node.js, ofrece herramientas que facilitan el desarrollo de aplicaciones más robustas y mantenibles.

Beneficios frente a JavaScript

- Tipado estático: Reduce errores al tiempo de desarrollo mediante la detección anticipada de errores de tipo.
- Compatibilidad: Todo código JavaScript es código TypeScript válido.
- Autocompletado y herramientas: Mejora la experiencia del desarrollador con funcionalidades avanzadas en los editores de texto.
- Mantenibilidad: Facilita el trabajo en equipo y el escalamiento de proyectos.
- Características avanzadas: Incluye clases, interfaces, módulos y decoradores que no existen en JavaScript puro.

Tipos básicos en TypeScript

TypeScript incluye varios tipos básicos que ayudan a definir variables con mayor precisión. Algunos de los más comunes son:

string: Representa texto.

- *let nombre: string = "Juan";*

number: Representa números (enteros y decimales).

- *let edad: number = 25;*

boolean: Representa valores lógicos (true/false).

- *let activo: boolean = true;*

array: Representa una lista de valores.

- *let numeros: number[] = [1, 2, 3, 4, 5];*

tuple: Representa un array con un número fijo de elementos de tipos conocidos.

- *let persona: [string, number] = ["Ana", 30];*

any: Acepta cualquier tipo. Se recomienda usarlo con moderación.

- *let variable: any = "Hola";*
- *variable = 42;*

void: Representa la ausencia de un valor (usado en funciones que no retornan).

- *function saludar(): void {*

  console.log("Hola mundo");

  }

null y undefined: Representan ausencia de valor.

- *let variableNula: null = null;*
- *let variableIndefinida: undefined = undefined;*

Clases en TypeScript

Las clases son plantillas para crear objetos, permitiendo definir propiedades y métodos. TypeScript mejora las clases de JavaScript al añadir tipado y visibilidad.

```
class Persona {
  nombre: string;
  edad: number;

  constructor(nombre: string, edad: number) {
    this.nombre = nombre;
    this.edad = edad;
  }

  saludar(): void {
    console.log(`Hola, mi nombre es ${this.nombre} y tengo ${this.edad} años.`);
  }
}

const persona1 = new Persona("Carlos", 28);
persona1.saludar();
```

Herencia en TypeScript

La herencia permite que una clase hija adquiera propiedades y métodos de una clase padre. Esto promueve la reutilización de código y facilita la extensión de funcionalidades

```
class Animal {
  nombre: string;

  constructor(nombre: string) {
    this.nombre = nombre;
  }

  hacerSonido(): void {
    console.log("Sonido genérico de animal");
  }
}

class Perro extends Animal {
  raza: string;

  constructor(nombre: string, raza: string) {
    super(nombre); // Llama al constructor de la clase padre
    this.raza = raza;
  }

  hacerSonido(): void {
    console.log("Guau guau");
  }
}

const miPerro = new Perro("Fido", "Labrador");
miPerro.hacerSonido();
```