Manual TypeScript.

#### **¿Qué es TypeScript y por qué usarlo?**

**TypeScript** es un superset de JavaScript, lo que significa que es JavaScript con "superpoderes". Fue creado por Microsoft para solucionar uno de los mayores problemas de JavaScript: la falta de tipado estático.

* **Tipado Estático:** Con TypeScript, puedes declarar explícitamente el tipo de una variable (string, number, boolean, etc.). Esto permite que el compilador de TypeScript detecte errores antes de que el código se ejecute.
* **Transpilación:** El código TypeScript (.ts) no se ejecuta directamente en los navegadores. En su lugar, es **transpilado** a JavaScript (.js) por un programa llamado tsc (TypeScript Compiler). Esto significa que tu código final será un JavaScript estándar que cualquier navegador o entorno (como Node.js) puede entender.

##### **Tipos de Datos Primitivos**

TypeScript añade tipos a las variables, lo que hace el código más claro y seguro.

* **string:** Para cadenas de texto.

Let nombre: string = "Juan";

* **number:** Para números enteros y flotantes.

Let edad: number = 30;

* **boolean:** Para valores de verdad (true o false).

Let esActivo: boolean = true;

* **any:** Un tipo que puede ser cualquier cosa. Úsalo con cuidado, ya que elimina las ventajas del tipado.

Let dato: any = "Hola";

dato = 10;

##### **Arreglos (Arrays)**

Puedes definir arreglos de un tipo específico.

Let frutas: string[] = ["manzana", "banana", "uva"];

Let numeros: Array<number> = [1, 2, 3, 4]; // Otra sintaxis

##### **Funciones**

Puedes tipar los parámetros y el valor de retorno de una función.

function saludar(nombre: string): string {

return `Hola, ${nombre}`;

}

// La siguiente línea daría un error en tiempo de compilación

// let saludo: string = saludar(123);

#### **Programación Orientada a Objetos (POO)**

TypeScript ofrece una experiencia de POO robusta, similar a la de otros lenguajes como Java o C#.

##### **Clases**

Las clases son planos para crear objetos.

class Persona {

private nombre: string; // La propiedad solo es accesible dentro de la clase

public edad: number; // Accesible desde cualquier lugar

constructor(nombre: string, edad: number) {

this.nombre = nombre;

this.edad = edad;

}

public mostrarDatos(): void {

console.log(`Nombre: ${this.nombre}, Edad: ${this.edad}`);

}

}

Let persona1 = new Persona("Ana", 25);

persona1.mostrarDatos(); // Salida: "Nombre: Ana, Edad: 25"

##### **Interfaces**

Las interfaces definen la "forma" de un objeto. Son útiles para asegurar que los objetos sigan una estructura específica.

interface Usuario {

id: number;

nombre: string;

email?: string; // El signo de interrogación indica que la propiedad es opcional

}

Function imprimirUsuario(usuario: Usuario) {

console.log(`ID: ${usuario.id}, Nombre: ${usuario.nombre}`);

}

let user = { id: 1, nombre: "Pedro", email: "pedro@mail.com" };

imprimirUsuario(user);  
  
  
**Cómo Descargar.**

1. **Instalar Node.js:** instalar Node.js si es que no lo tienes, que incluye npm (Node Package Manager).
2. **Instalar TypeScript:** Abrir la terminal o línea de comandos y ejecuta el siguiente comando globalmente.

npm install -g typescript