



dev Senior

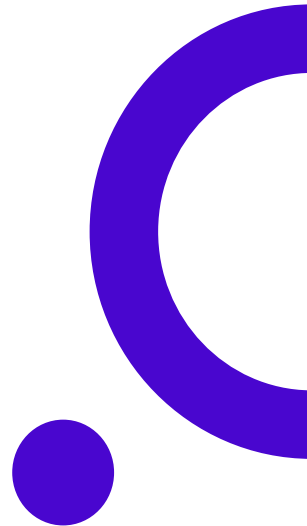
www.devseniorcode.com



Fundamentos de TypeScript

Tipado Estático y Clases

www.devseniorcode.com



Objetivos de Aprendizaje

- ✓ Comprender el tipado estático en TypeScript y su diferencia con JavaScript.
- ✓ Definir tipos primitivos y compuestos.
- ✓ Declarar y usar clases, constructores y métodos.
- ✓ Aplicar herencia y modificadores de acceso.




¿Qué es TypeScript?

- ▶ Superconjunto de JavaScript con tipado estático.
- ▶ Mejora el autocompletado y evita errores comunes.
- ▶ Código más robusto y mantenible.



Tipos de Datos en TypeScript

 Tipos Primitivos: string, number, boolean.

 Tipos Avanzados: arrays, tuplas, unión, tipos personalizados.

www.devseniorcode.com

Ejemplo de Tipado

```
let edad: number = 30;  
let nombre: string = "Ana";  
let activo: boolean = true;  
let tupla: [string, number] = ["id",  
100];  
  
type Usuario = { nombre: string; edad:  
number };
```

www.devseniorcode.com



dev
Senior



Instalación del compilador

Dentro del proyecto npm, ejecute el siguiente comando para instalar el compilador:

```
npm install typescript --save-dev
```

El compilador se instala en el directorio y se puede ejecutar con: `.node_modules\npx tsc`

```
npx tsc
```

www.devseniorcode.com



Tipos primitivos

```
let nombre: string = "Juan";  
let edad: number = 25;  
let esActivo: boolean = true;  
  
console.log("Nombre: "+nombre);  
console.log("Edad:"+edad);  
console.log("Estado:"+esActivo);
```





Tipos avanzados



dev
Senior

```
let ids: number[] = [1, 2, 3];  
let persona: { nombre: string; edad: number } = { nombre: "Ana", edad: 30 };  
  
ids.forEach(number => {  
  console.log(number);  
});  
  
console.log(persona.nombre, persona.edad);
```



www.devseniorcode.com



Alias

```
type Usuario = {  
  id: number;  
  nombre: string;  
  correo: string;  
};  
  
// Asignación de valores  
const usuario1: Usuario = {  
  id: 1,  
  nombre: "Laura Gómez",  
  correo: "laura.gomez@example.com"  
};  
  
console.log(usuario1);
```





Clases en TypeScript

- ▶ Molde para crear objetos con propiedades y métodos.
- ▶ Usa constructor para inicializar.
- ▶ Métodos definen comportamientos.

dev
Senior





Clases

```
class Persona {  
  nombre: string;  
  edad: number;  
  
  constructor(nombre: string, edad: number) {  
    this.nombre = nombre;  
    this.edad = edad;  
  }  
  
  saludar(): void {  
    console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`);  
  }  
}  
  
const persona1 = new Persona("Carlos", 28);  
persona1.saludar();
```

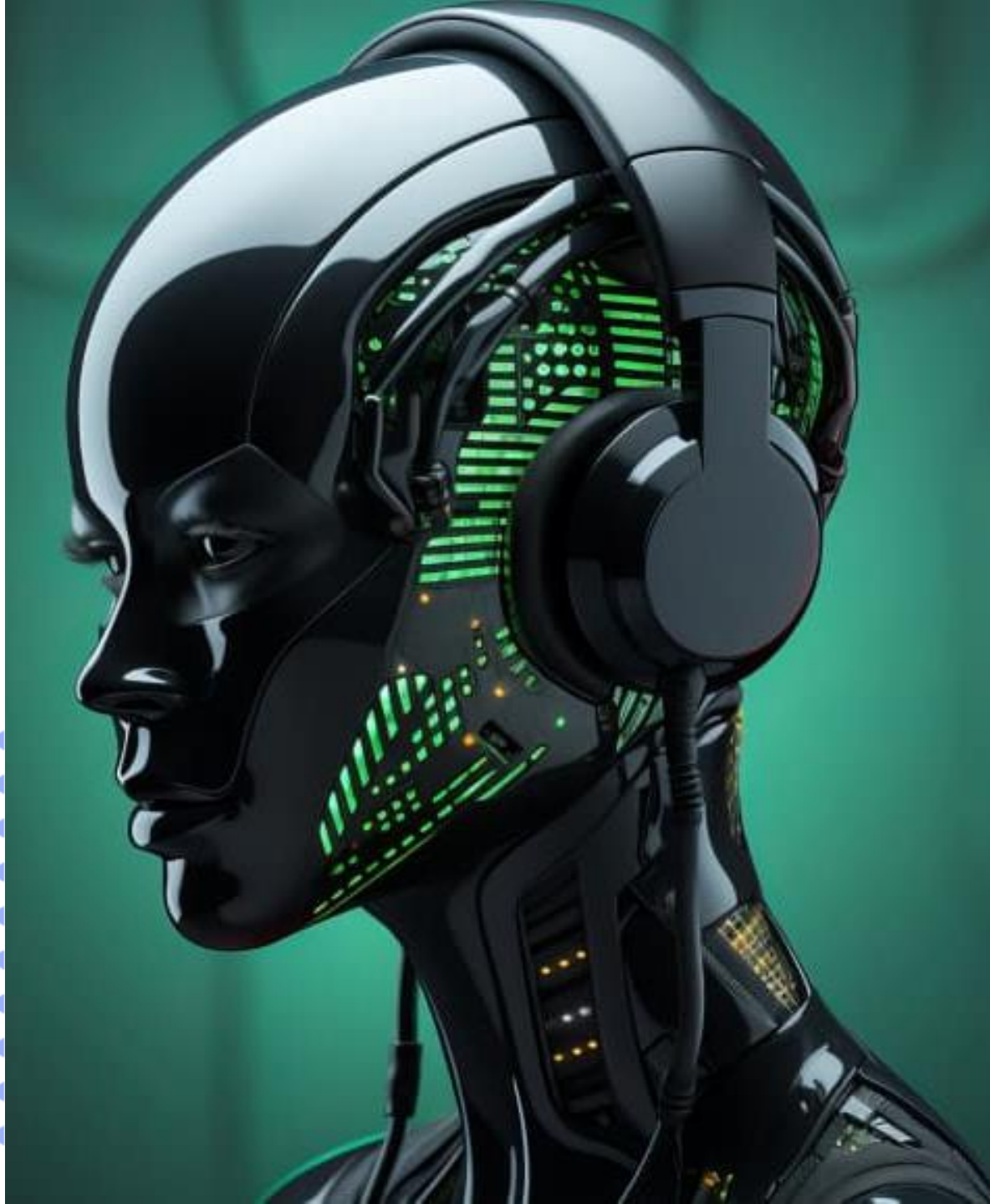




Modificadores de Acceso

- ✓ public: accesible desde cualquier parte.
- ✓ private: solo dentro de la clase.
- ✓ protected: clase y subclases.

www.devseniorcode.com





Modificador Acceso

dev
Senior

```
class Cuenta {  
  private saldo: number;  
  
  constructor(saldoInicial: number) {  
    this.saldo = saldoInicial;  
  }  
  
  consultarSaldo(): number {  
    return this.saldo;  
  }  
}  
  
const cuenta1 = new Cuenta(1500000);  
console.log(cuenta1.consultarSaldo());
```



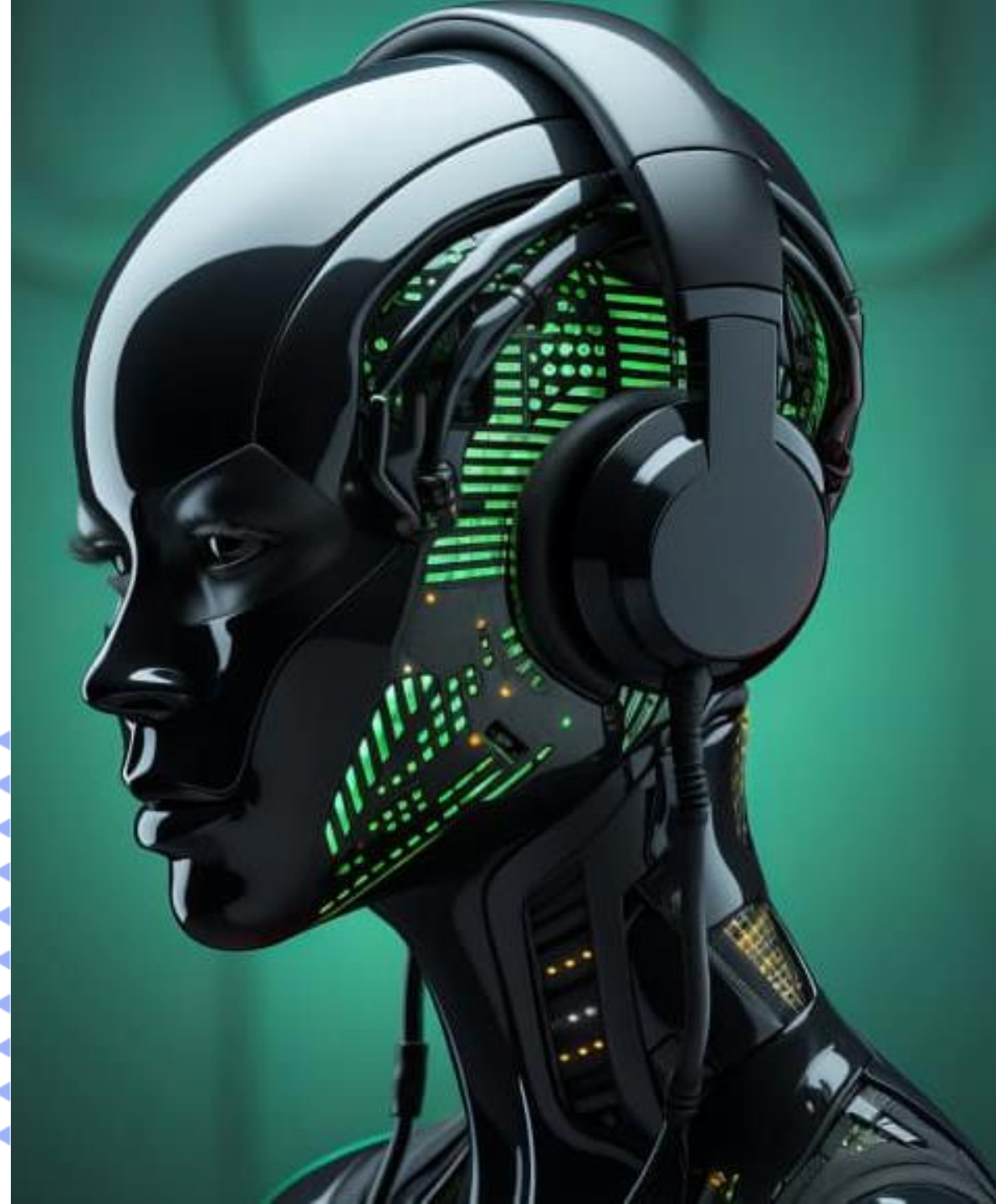
www.devseniorcode.com



Herencia en Clases

```
class Empleado extends Persona  
{  
    constructor(...) { super(...); }  
    mostrarCargo(): void { ... }  
}
```

www.devseniorcode.com





Ejercicio Práctico

- 👉 Crea una clase Producto con nombre, precio y stock.
- 👉 Método mostrarInfo().
- 🔄 Extra: heredar en ProductoDigital con método descargar().

Cierre y Recursos



dev
Senior



Usa tipos siempre que puedas.



Las clases organizan el código.



<https://www.typescriptlang.org/docs/>



<https://www.typescriptlang.org/play>