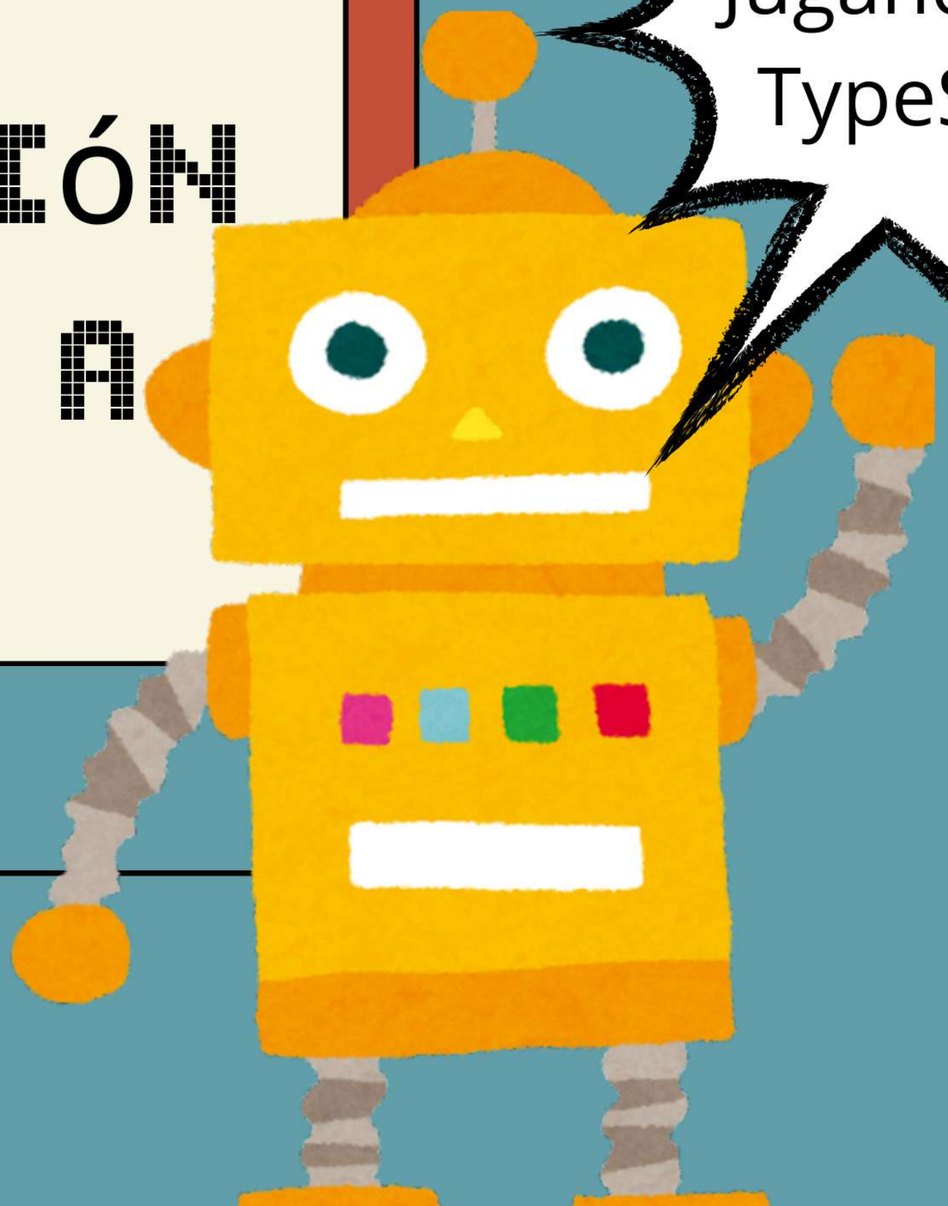
OOP (PROGRAMACIÓN ORIENTADA A OBJETOS)

Jugando con TypeScript

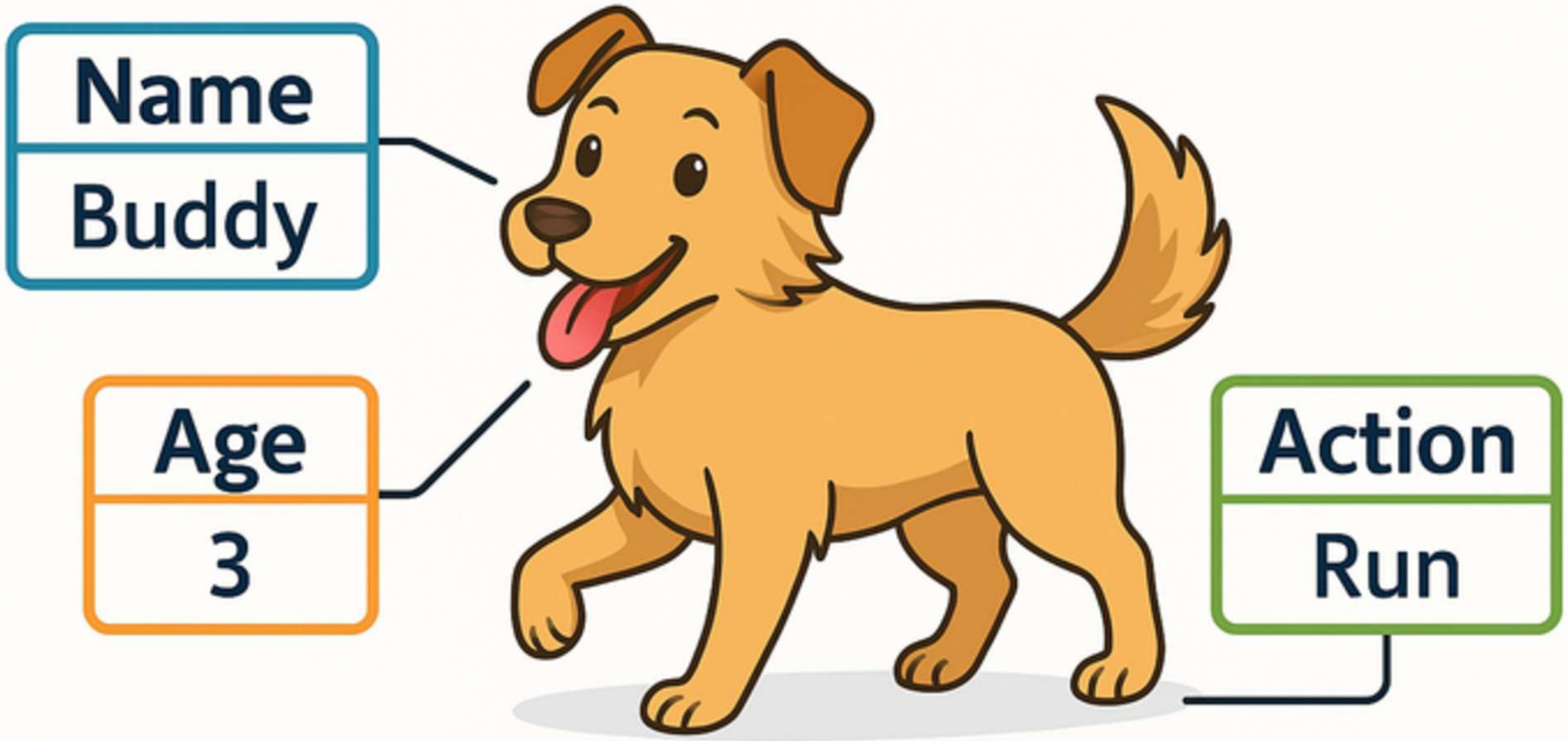


Start

¿QUÉ ES HERENCIA EN PROGRAMACIÓN?

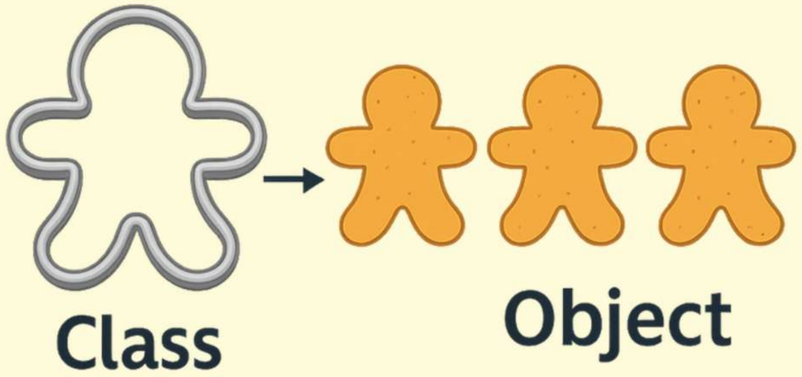
- Es una forma de escribir código pensando en “cosas” (objetos).
- Cada objeto tiene características (como nombre, color) y acciones (como correr, hablar).
- Ejemplo: un perro tiene nombre, edad, y puede ladrar.
- Otro ejemplo cotidiano: Una bicicleta tiene ruedas, frenos y puede pedalear → eso ya es pensar en objetos.
- ¿Por qué con TypeScript? es un lenguaje tipado y flexible.

OOP CONCEPTS



¿QUÉ ES UNA CLASE?

Una clase es como un molde




```
class Animal {
  name: string;

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

  speak() {
    console.log(`${this.name} hace un sonido.`);
  }
}
```

¿Qué es herencia?

Herencia significa que una clase nueva puede usar lo que ya tiene otra.

Ts

```ts
class Dog extends Animal {
  speak() {
    console.log(`${this.name} dice: ¡Guau!`);
  }
}
```

Ejemplo: una familia → el hijo hereda cosas del padre (como el apellido), pero tiene su propia voz.

Guerrero

Mago

Personaie

CONCEPTOS CLAVE

CONCEPTO	QUÉ SIGNIFICA	EJEMPLO VIDA REAL
Clase	Molde	Plano de una casa
Objeto	Instancia	Una casa construida
Herencia	Reutilización	Un coche eléctrico hereda de un coche normal
Encapsulación	Protección	Caja fuerte con llave
Polimorfismo	Flexibilidad	Un botón que hace cosas distintas según el contexto

COMANDOS TS

1. git init
2. npm init-y
3. npm i -D typescript
4. npx tsc --version
5. npx tsc --init
6. npm i ts-node -D
7. crear .gitignore (gitignore.io)
8. crear .editorconfig
9. npm install ts-node @types/node typescript --save-dev

Archivo tsconfig.json

```
{
  "compilerOptions": {
    "rootDir": "./src",
    "outDir": "./dist",
    "module": "commonjs",
    "target": "es2020",
    "moduleResolution": "node",
    "types": ["node"],
    "sourceMap": true,
    "declaration": true,
    "declarationMap": true,
    "strict": true,
    "isolatedModules": true,
    "skipLibCheck": true
  }
}
```

MODIFICADORES DE ACCESOS

Usa **private** para proteger datos sensibles, **protected** para permitir acceso desde clases hijas, y **public** para lo que debe ser visible externamente.

Modificador	Acceso desde la clase	Acceso desde la subclase	Acceso desde fuera
public	✓	✓	✓
privated	✓	✗	✗
protected	✓	✓	✗

REFERENCIAS CONSULTADAS

- @BitBoss: https://www.youtube.com/watch?v=SI7O81GMG2A&t=151s
- @BettaTech: https://www.youtube.com/watch?v=tTPeP5dVuA4
- Nicolás Molina (@nicobytes): https://www.youtube.com/watch?v=-Z5BNNQt4bE