Programación Orientada a Objetos con TypeScript

Jhan Franco Avila Torres 1152490

Ender Jesus 1152476

William Camilo 1152457

Juan Jose Vera 1152489

TypeScript: La Versión Mejorada de JavaScript

Que es?

TypeScript es un superconjunto de JavaScript que añade tipos estáticos, clases, interfaces y otras características avanzadas.

Características Clave

Tipado estático opcional.

POO.

Compatibilidad con JavaScript.

lenguaje de alto nivel.

Beneficios

Mejora la escalabilidad, la legibilidad y la mantenibilidad del código, lo que lo hace ideal para proyectos a gran escala.

Historia de TypeScript

1

2010

Microsoft empieza el desarrollo de TypeScript como un superconjunto de JavaScript para abordar limitaciones de escalabilidad.

2

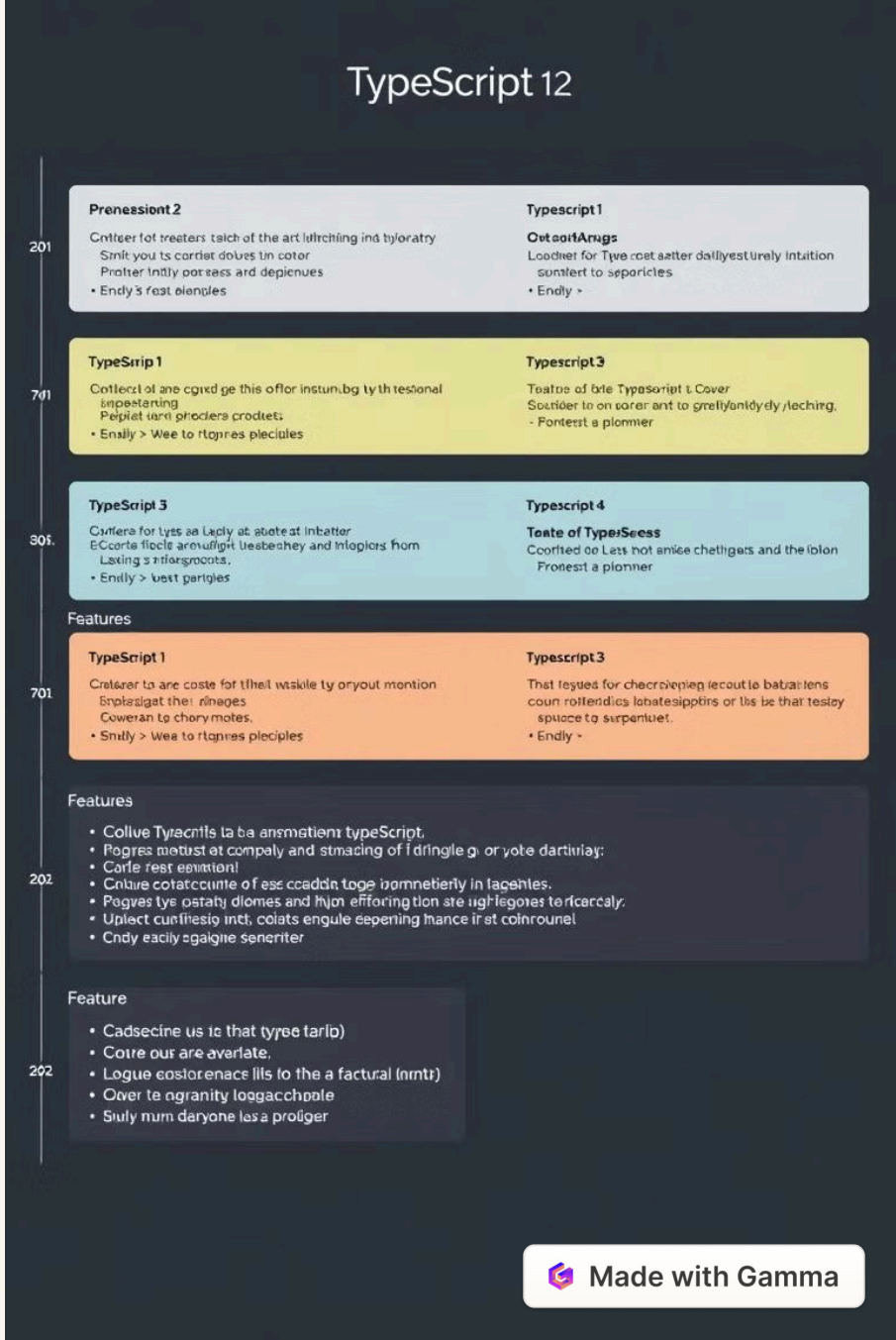
2012

Primera versión pública de TypeScript, que añade tipado estático y soporte para programación orientada a objetos.

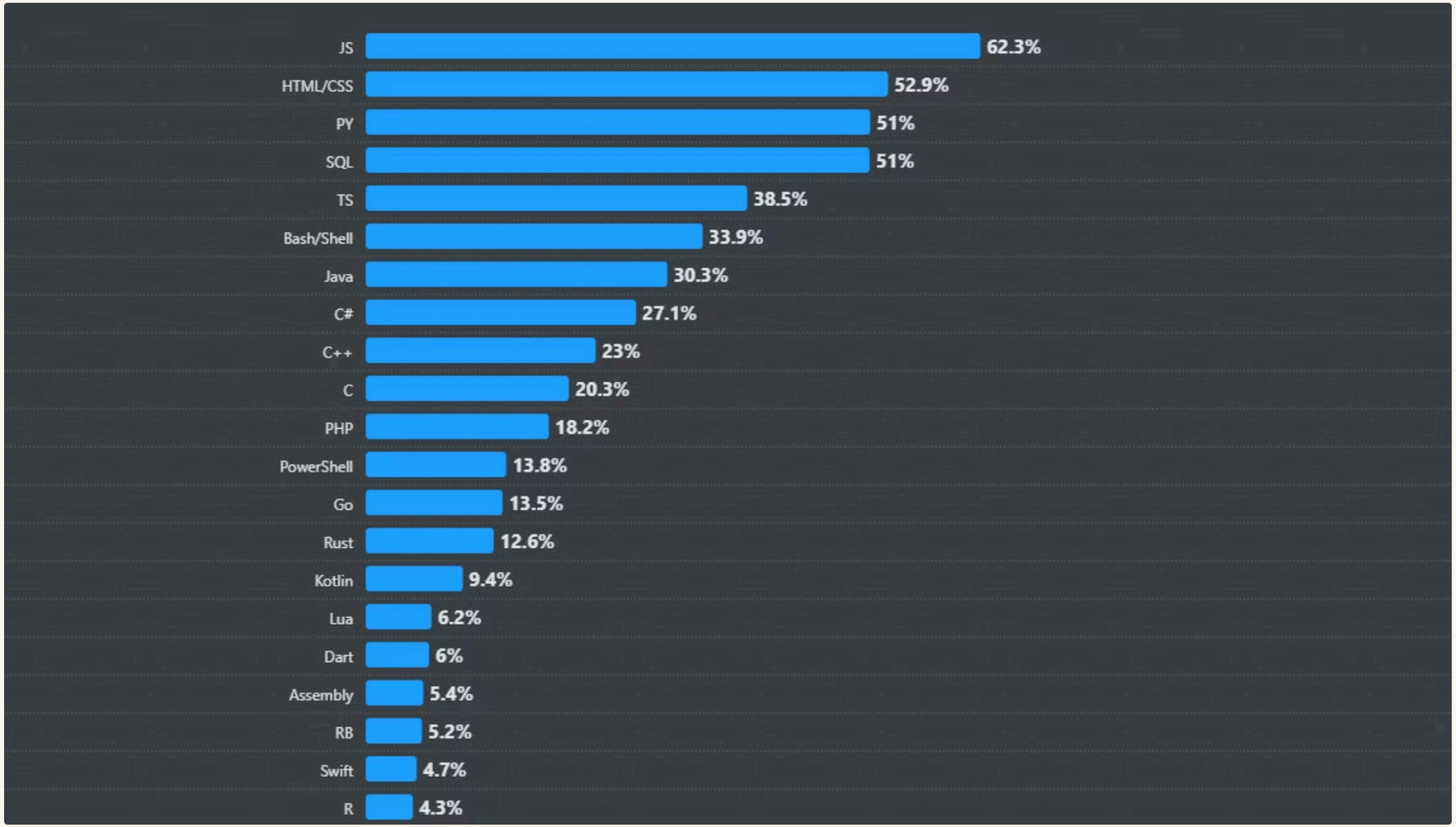
3

Actualidad

TypeScript se ha convertido en uno de los lenguajes más populares, con adopción masiva en frameworks como Angular y grandes empresas de tecnología.



Ranking



https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language

Aplicaciones de la POO con TypeScript

Desarrollo Web

TypeScript es ampliamente utilizado en el desarrollo de aplicaciones web modernas, como las basadas en frameworks como Angular y React.

Aplicaciones de Escritorio

La combinación de TypeScript y frameworks como Electron permite crear aplicaciones de escritorio multiplataforma.

Aplicaciones Móviles

Con herramientas como NativeScript, los desarrolladores pueden construir aplicaciones móviles nativas usando TypeScript.

Estructura, sintaxis y como escribir código en TypeScript

TIPOS DE DATOS EN TYPESCRIPT

JAVASCRIPT	TYPESCRIPT
number	Todos los tipos de JS
string	any
boolean	unknow
null	never
undefined	array
object	tuplas
function	enums

```typescript
//Tipos de variables
let extinsionDinosaurios: number = 76_000_000
let dinosaurioFavorito: string = "T-Rex"
let extintos: boolean = true

extintos = 2

function dinosaurio(config: string): string{
    return config
}
```

```typescript
//Arrays
let animales: string[] = ["perro","gato","vaca"]
let nums: number[] = [1,2,3]
let nums2: Array<number> = []

nums.push(4)
nums.forEach(function(value){
    console.log(value)
})
```

```typescript
//Tuplas
let tuplas: [number, string] = [1, "uno"]
let tupla: [number, string[]] = [1, ["uno", "dos"]]
```

```ts
//Enums
const chica = "s"
const mediana = "m"
const grande = "l"
const extragrande = "xl"

enum Talla {Chica = "s", Mediana = "m", Grande = "l", ExtraGrande = "xl"}

const variable1 = Talla.Grande
console.log(variable1)

const enum LoadingState {Idle, Loading, Success, Error}
const estado = LoadingState.Success
```

```ts
//Objetos (Literales)
const objeto:{
    readonly id:number,
    nombre:string,
    talla: Talla
} = {id:1,
    nombre:"",
    talla:Talla.Mediana
}

objeto.nombre = "Jhan"

type Direccion ={
    direccion:string,
    casa?:number
}

type Persona = {
    readonly cedula:number,
    nombre:string,
    casado:boolean,
    direccion:Direccion
}

const objeto2:Persona = {
    cedula:1,
    nombre:"Camilo",
    casado:true,
    direccion:{
        direccion:"UFPS",
    }
}

const arreglo: Persona[] = [] //Arreglo de Objeto Personma
```

Conceptos Clave de POO

1 Clases y Objetos

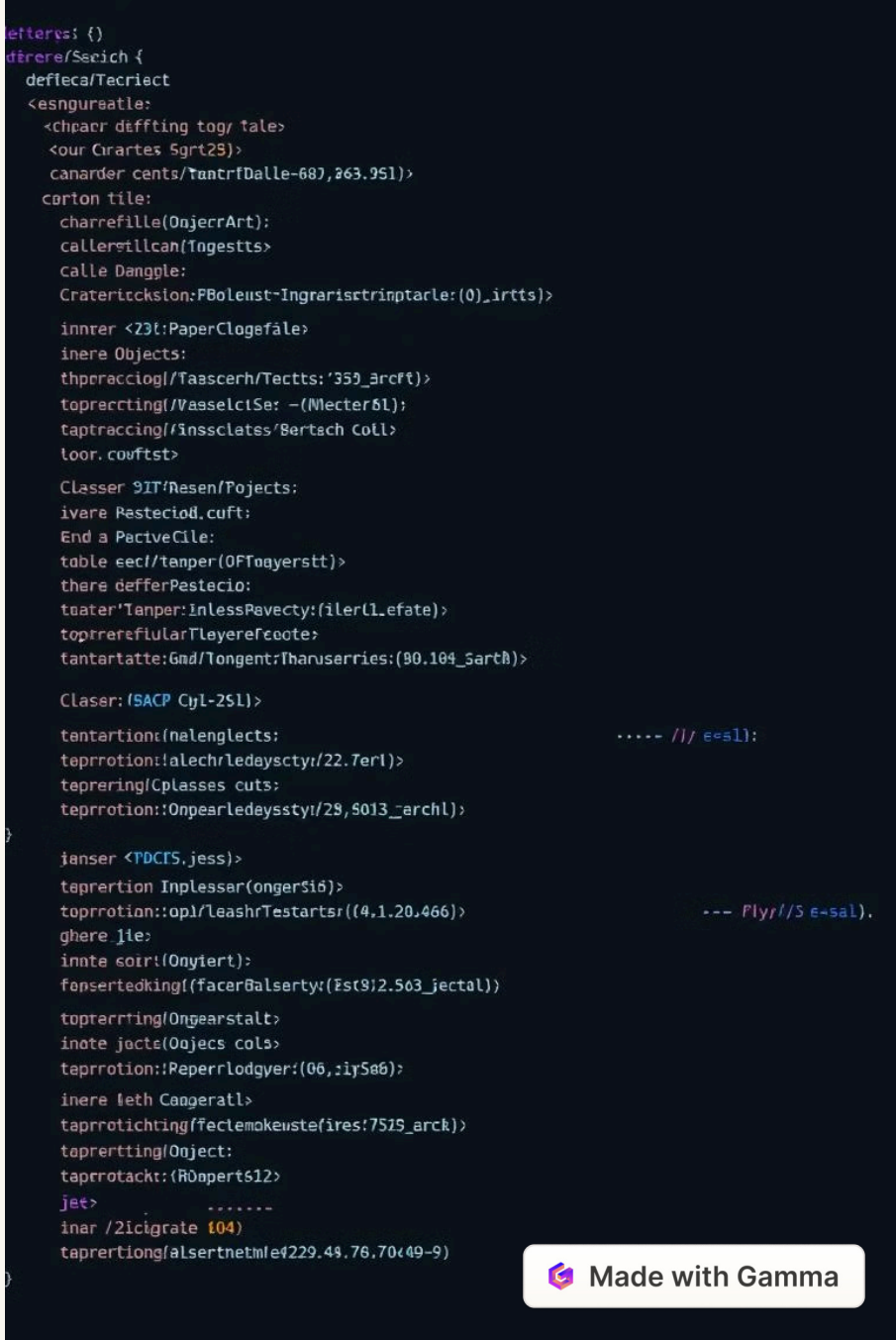
Las clases definen las características y comportamientos de los objetos, que son instancias concretas de esas clases.

2 Encapsulamiento

El encapsulamiento oculta los detalles internos de un objeto, exponiendo solo los métodos y propiedades necesarios.

3 Herencia y Polimorfismo

La herencia permite a las clases hijas heredar propiedades y métodos de sus clases padres, mientras que el polimorfismo les permite tener comportamientos específicos.



Conceptos de Clases y Objetos

```
//Clases Objetos
class Pelicula{
    //Propiedades
    nombre: string
    protagonistas?: string[]

    //Constructor
    constructor(nombre:string, protagonistas?:string[]){
        this.nombre = nombre,
        this.protagonistas = protagonistas
    }

    //Metodos
    proyectarPelicula():void{
        console.log(`${this.nombre} se esta proyectando`)
    }
}

let titanic = new Pelicula("titanic")
let vengadores = new Pelicula("End Game", ["Spiderman", "Ant Man"])

titanic.proyectarPelicula()
console.log(vengadores)
```

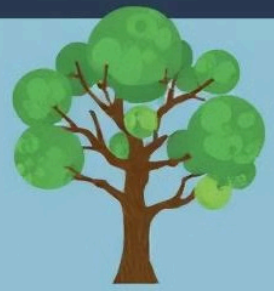
```
26  //Encapsulamiento y Genericos
27  class Sorteo<T>{
28      private numero?: T;
29
30      constructor(private nombre:string){}
31
32      getNumero(){
33          return this.numero
34      }
35
36      setNumero(numero:T):void{
37          this.numero = numero
38      }
39
40      public sortear():string{
41          return "para "+this.nombre+" el ticket es " + this.numero
42      }
43  }
44
45  let sorteo = new Sorteo<number>("Franco")
46  sorteo.setNumero(4)
47  console.log(sorteo.sortear())
48
49  let sorteo2 = new Sorteo<string>("Maria")
50  sorteo2.setNumero("A5")
51  console.log(sorteo2.sortear())
```

```ts
// INTERFACES

// Interface básica:
interface Persona {
  nombre: string;
  edad: number;
}

// Interface con propiedades opcionales:
interface Producto {
  nombre: string;
  precio: number;
  descripcion?: string;
}

// Interface para funciones:
interface Comparador {
  (a: number, b: number): boolean;
}

// Interface para clases (class interfaces):
interface Persona {
  nombre: string;
  edad: number;
  saludar(): void;
}
```

```
airteralvy
/ arrays(elletitl);
  /sargetatdet((let. spaciettly riim):
  "atircdoccentes( arbackerlatestly lerf);
  /cuser(hestilate:(on(.larlayer(nathchiotan(AttAit=all));
  /eaplr(ust ortar/icles.ar)
  /tarerdecradeer((et((evtlltts=d.bt alat);
```

```
dtherapp
Thatr vic(odrer(rus)
Thathrler(o.acticetton(Cylal aplf Fullotr),
Inscn(bbt cpptlatetterletity) I S(F hap");
```

```
instlenals wnal isturtfrees
> contartof(UhtiatiatTipersesher
  chaletortlor(Matittistrees-
  thulcrtacter(insiatintertesber
  ⋄arglotofrarp-(Scret)›
  Thaleriettor(natiatTircessrr)›
  enterricicor((reth vnatilz-up lity);›
  Thalcrtotactllatlat(ircessrr)
```



```
linked lists
> cratlclackenole jests -ralect);
  ttres nelaſclet;
  cbach tebcebetlat(ıopa(Jnateult)
  Frecchtores(lat(0Udienedl);
  /reatlancolor(,drondlest; seltactlat.(rfrl);
  "railstetcoricltectdalecsing(,.fiC.).g "iree:
  /ccoplectcbltlss(com/pct(.alse.pretlOI(, lart)
  renl(onn(apgr(iree::
> reitectogcahatle(ng >ciseh rir");
```

Contenedores y Asociaciones

Arrays

Los arrays son contenedores que almacenan colecciones de objetos del mismo tipo.

Diccionarios

Los diccionarios (a.k.a. mapas o tablas de hash) asocian claves con valores de diferentes tipos.

Listas Enlazadas

Las listas enlazadas son contenedores lineales de objetos que se enlazan dinámicamente.

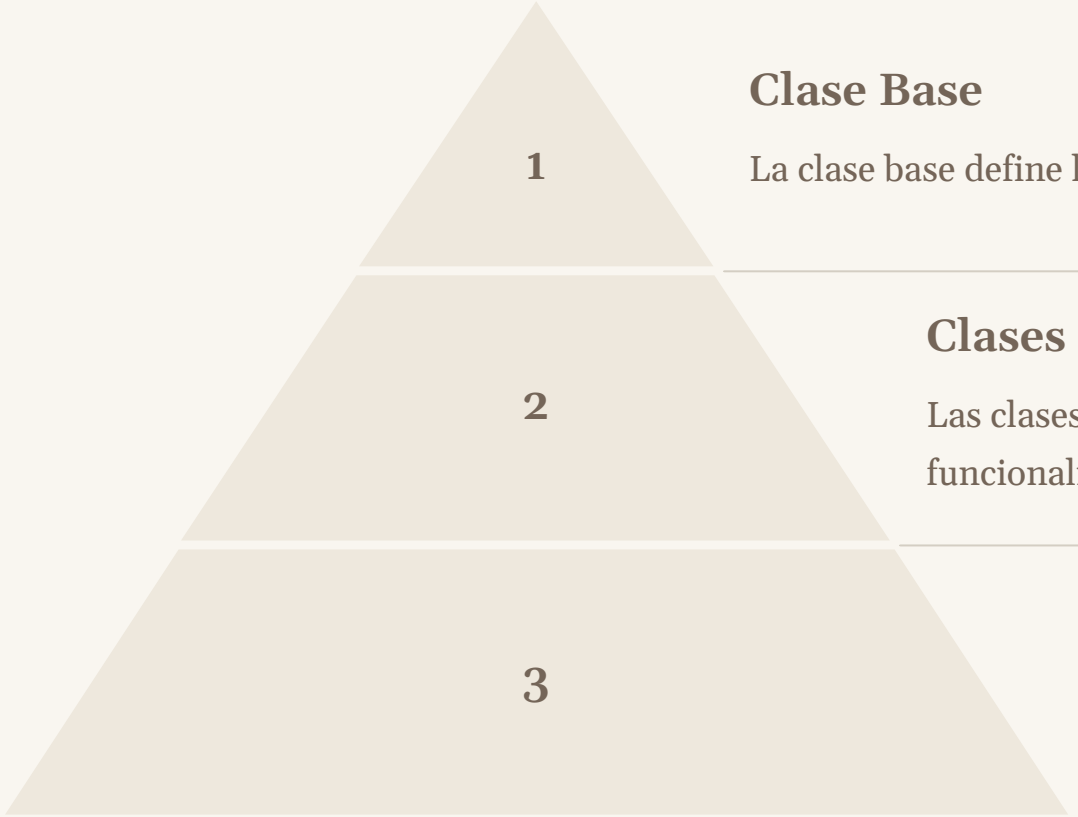
Árboles

Los árboles son estructuras jerárquicas de nodos que permiten organizar y navegar datos de manera eficiente.

Como declarar un Array en TypeScript

```
let heroes: string[] = [
    "Iron Man",
    "Spiderman",
    "Thor",
    "Hulk",
    "Black Widow",
    "Hawk Eye"
];
```

Herencia y Polimorfismo en TypeScript



Clase Base

La clase base define las características comunes a todas las clases hijas.

Clases Derivadas

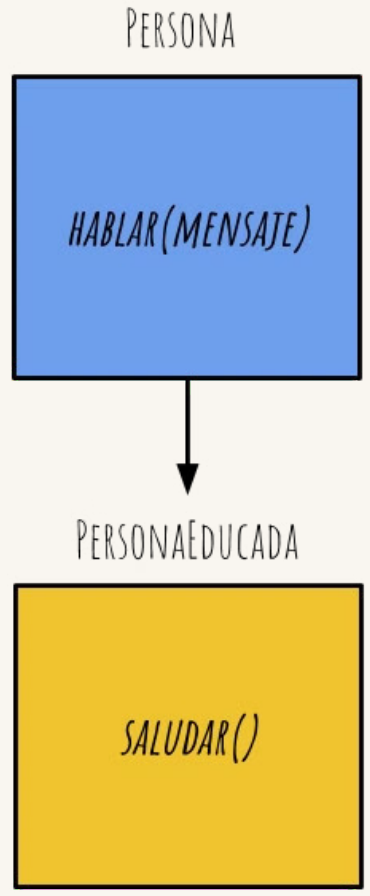
Las clases derivadas heredan de la clase base y pueden añadir o modificar funcionalidades.

Polimorfismo

Las clases derivadas pueden implementar sus propios métodos con el mismo nombre que los de la clase base.

Ejemplo de herencia en TypeScript

```
class PersonaEducada extends Persona {
    saludar() {
      this.hablar('Buen Dia señor')
    }
}
let persona1 = new PersonaEducada();
persona1.saludar();
```

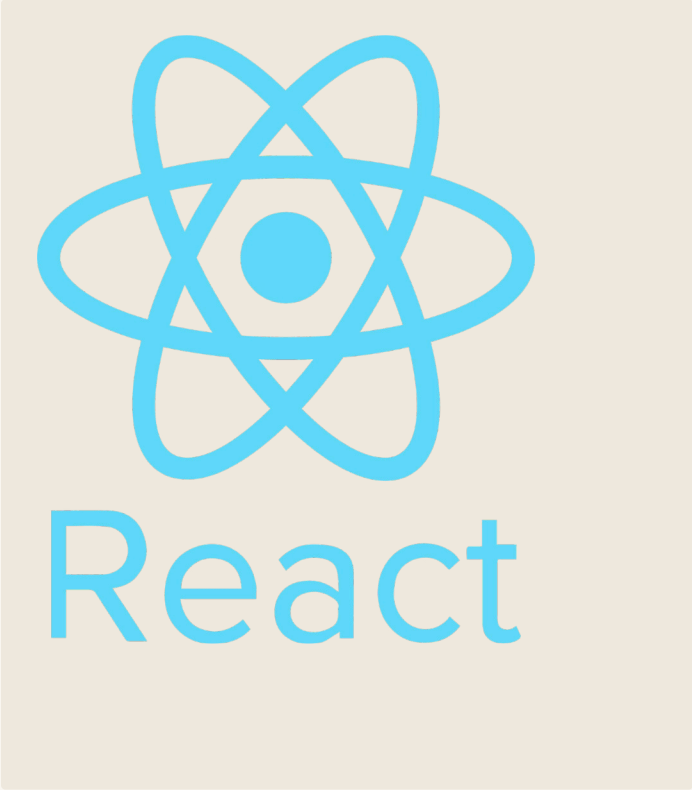


Ejemplo de Polimorfismo en TypeScript

```typescript
class Animal {
    comer() {
        console.log('El animal come');
    }
}

class Perro extends Animal {
    comer() {
        console.log('El perro come croquetas');
    }
}

class Gato extends Animal {
    comer() {
        console.log('El gato come pescado');
    }
}
```

GUI, Consola, Desktop, Web y Móvil

Interfaz Grafica



Consola

```
PS E:\DESCARGAS 2\Programacion Franco\CURSO TYPESCRIPT> tsc -v
Version 5.6.3
```

Nos da la versión de **TypeScript** de nuestro equipo

```
PS E:\DESCARGAS 2\Programacion Franco\CURSO TYPESCRIPT> tsc -init


   target: es2016
   module: commonjs
   strict: true
   esModuleInterop: true
   skipLibCheck: true
   forceConsistentCasingInFileNames: true
```

Crea el archivo de configuración del compilador **tsconfig.json**

```
PS E:\DESCARGAS 2\Programacion Franco\CURSO TYPESCRIPT> tsc index.ts
```

Compila y traduce un archivo de TypeScript a **JavaScript**

```
PS E:\DESCARGAS 2\Programacion Franco\CURSO TYPESCRIPT> tsc -w
```

Entra al modo **Observador** del archivo seleccionado (**Tiempo Real**)

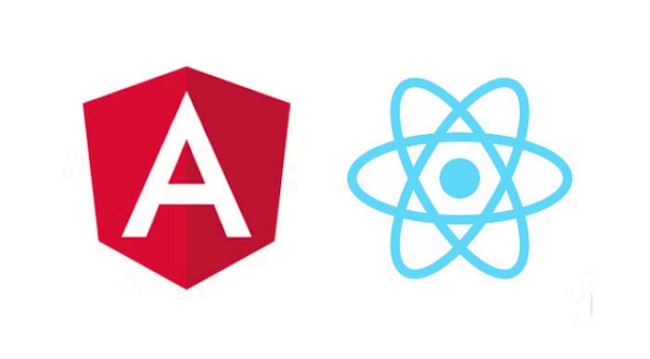
Desktop, Web y Móvil

Escritorio

Es posible desarrollar para Desktop con herramientas como Electron.

Web

Es su uso mas popular, implementándose en proyectos web de gran producción y escala.



Movil

Gracias a React Native se pueden desarrollar aplicaciones Móviles con TS.