



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.

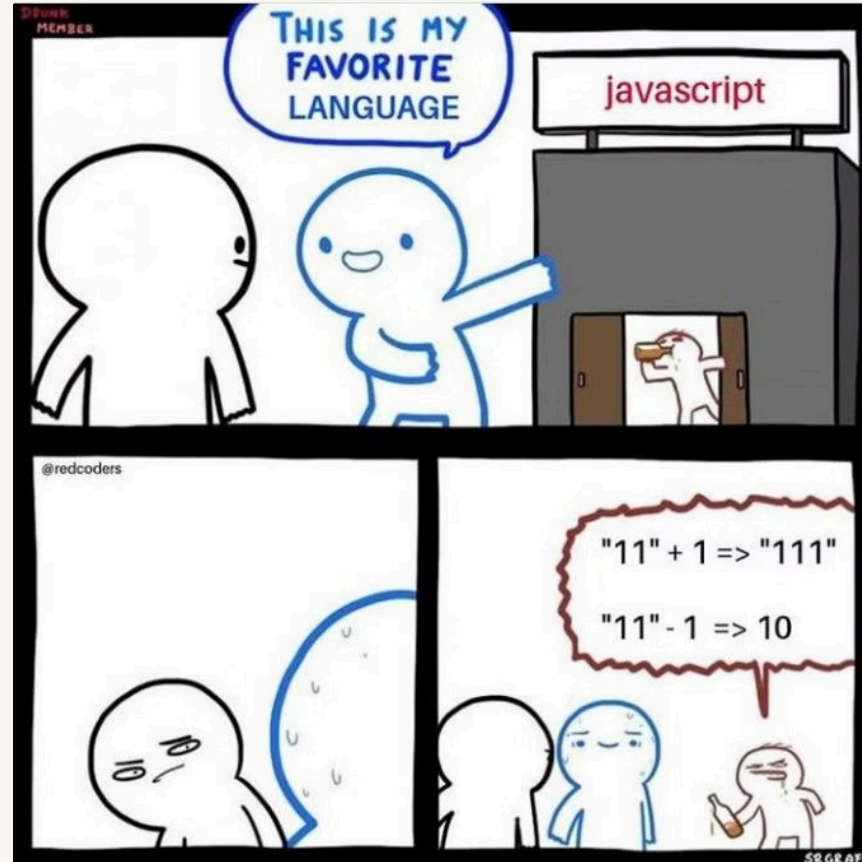
POO.

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

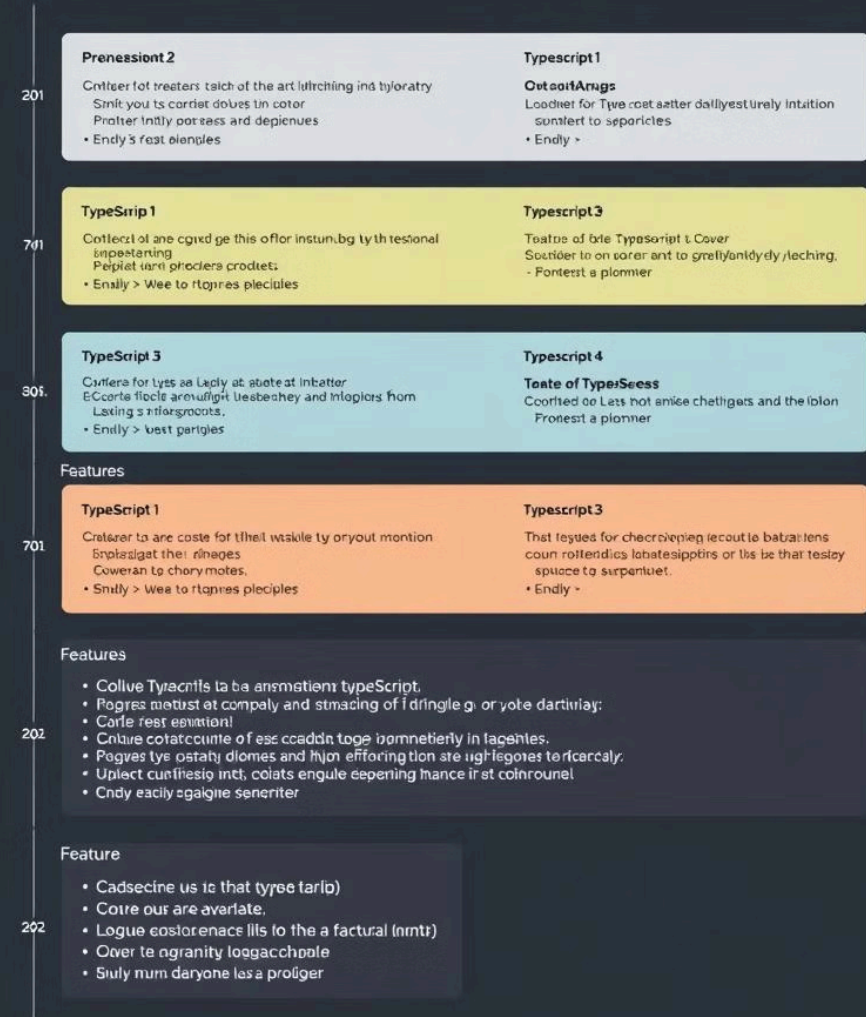
Primera versión pública de TypeScript, que añade tipado estático y soporte para programación orientada a objetos.

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.

TypeScript 12



Versión Actual

TypeScript 5.6.3 Latest

For release notes, check out the [release announcement](#).




For the complete list of fixed issues, check out the


- [fixed issues query for Typescript 5.6.0 \(Beta\)](#).
- [fixed issues query for Typescript 5.6.1 \(RC\)](#).
- [fixed issues query for Typescript 5.6.2 \(Stable\)](#).
- [fixed issues query for Typescript 5.6.3 \(Stable\)](#).


Downloads are available on:


- [npm](#)
- [NuGet package](#)


▼ Assets 3


| | | |
|--|---------|-------|
|  typescript-5.6.3.tgz | 3.98 MB | Oct 8 |
|  Source code (zip) | | Oct 8 |
|  Source code (tar.gz) | | Oct 8 |


 60

 11

 19

 18

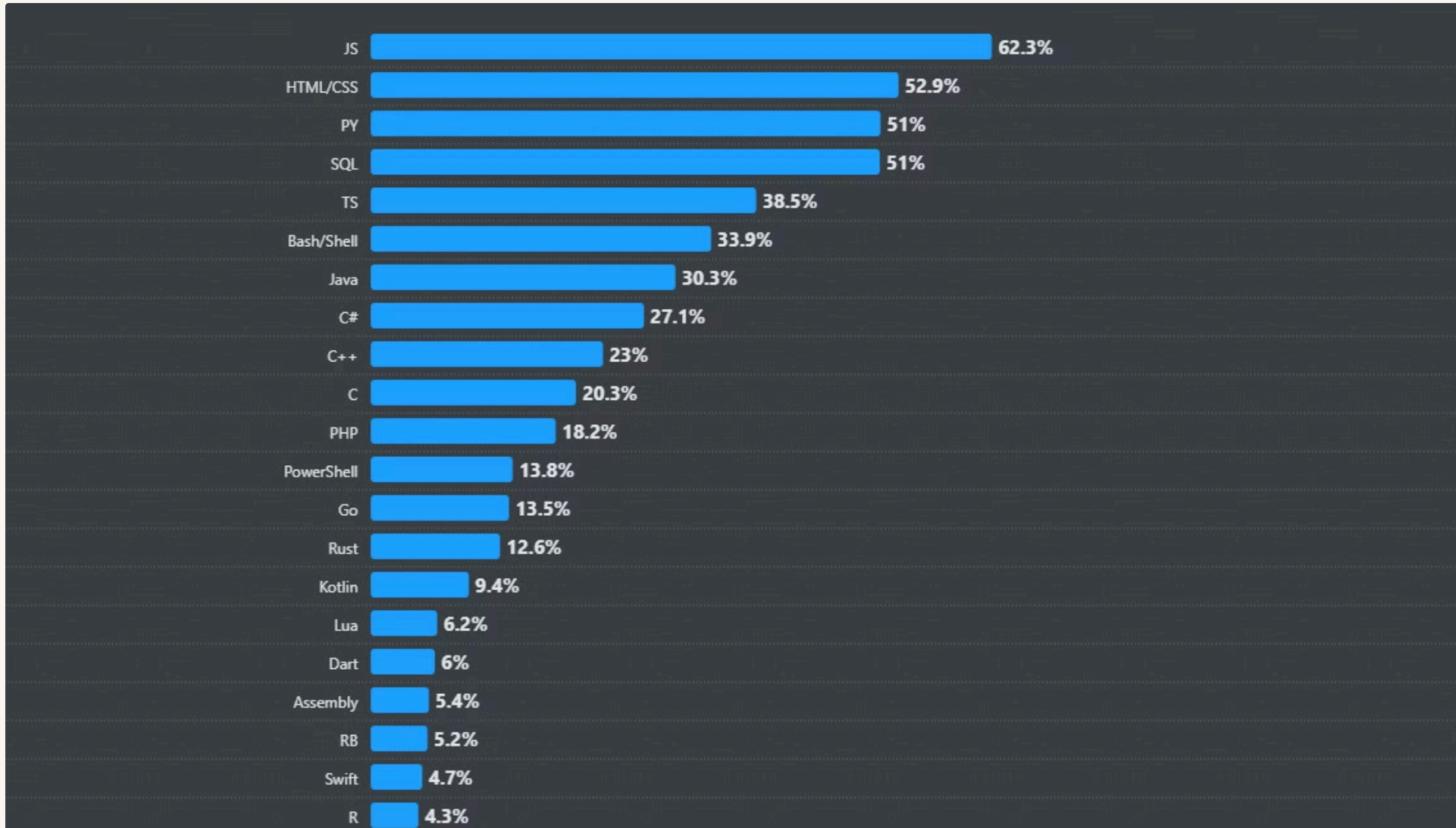
 17

 5

78 people reacted

La versión actual de TypeScript es la **5.6.3**, lanzada en octubre de 2024. Esta versión incluye varias mejoras, como la verificación de sintaxis en expresiones regulares y soporte para nuevos métodos propuestos en **Set**, como **union**, **intersection** y **difference**, entre otros.

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 |


```
1 //Tipos de variables
2 let extinsionDinosaurios: number = 76_000_000
3 let dinosaurioFavorito: string = "T-Rex"
4 let extintos: boolean = true
5
6 extintos = 2
7
8 ✓ function dinosaurio(config: string): string{
9     return config
10 }
```

```
10 //Arrays
11 let animales: string[] = ["perro","gato","vaca"]
12 let nums: number[] = [1,2,3]
13 let nums2: Array<number> = []
14
15 nums.push(4)
16 nums.forEach(function(value){
17     console.log(value)
18 })
```

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

```

24 //Enums
25 const chica = "s"
26 const mediana = "m"
27 const grande = "l"
28 const extragrande = "xl"
29
30 enum Talla {Chica = "s", Mediana = "m", Grande = "l", ExtraGrande = "xl"}
31
32 const variable1 = Talla.Grande
33 console.log(variable1)
34
35 const enum LoadingState {Idle, Loading, Success, Error}
36 const estado = LoadingState.Success

```

```

39 //Objetos (Literales)
40 ✓ const objeto:{
41     readonly id:number,
42     nombre:string,
43     talla: Talla
44 } = {id:1,
45     nombre:"",
46     talla:Talla.Mediana
47 }
48
49 objeto.nombre = "Jhan"
50
51 ✓ type Direccion ={
52     direccion:string,
53     casa?:number
54 }
55
56 ✓ type Persona = {
57     readonly cedula:number,
58     nombre:string,
59     casado:boolean,
60     direccion:Direccion
61 }
62
63 ✓ const objeto2:Persona = {
64     cedula:1,
65     nombre:"Camilo",
66     casado:true,
67     direccion:{
68         direccion:"UFPS",
69     }
70 }
71
72 const arreglo: Persona[] = [] //Arreglo de Objeto Persona

```

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.

```
letteres: {}
direre/Saich {
  defleca/Tecriect
  <esnguraatle:
    <Chpacr deffting tog/ tale>
    <our Cirartes Sgrt2S>
    canarder cents/TantrfDalle-68J,263.951>
  corton tile:
    charrefille(OnjerrArt):
    callersillcan(Ingestts)
    calle Danggle:
    Cratericcksion: PBoleust-Ingraristtrintparle: (0)_irtts>

  innrer <23l:PaperClogefale>
  inere Objects:
    thperacciogl/Taascerrh/Tectts: '359_Arcft>
    toprrccting/VassclclSe: -(Mecter6L);
    taptracing/Inssclates/Bertach Coll>
    loor. couftst>

  Classer 9IT/Nesen/Tojects:
    ivare Pesteciocl. cuft:
    End a PactiveCile:
    table eec//tanper (OFFnagerstt)>
    there defferPestecio:
    toater'Tanper: InlessPavecty: (filer(1.efate)>
    toprrerefiularTlayereFcoote:
    tantartatte: Gmd/Tongent: Thanuserries: (90.104_Sarth)>

  Claser: /SACP Cyl-25L>

  tantartion: (nalenglects:
  teprrroton: !alechr/ledayssty: /22.7er1>
  teprrering/Cplasses cuts:
  teprrroton: !Onpearledeyssty: /29,5013_archl);

  janser <PDCFS.jess>
  teprrerion Inplessar(onger$16>
  toprrroton: !opl/leashrTestartsr: ((4,1.20.466)>
  ghere jle:
  innte soir: (Onyiert):
  fonsertedkingl(facerBalserty: (Est9:2.563_jectal))
  toprrercting/Ongeartalt>
  inote jecte(Objecs cols>
  taprrroton: !Reperrlodgyer: (06,1iyS66>
  inere leth Cangeratl>
  taprrroctiching/fectemakeuste(fires:7515_arch)>
  taprrertting/Onject:
  taprrrotack: (R0npert612>
  jee>
  inar /2icigrate 104>
  taprrertiong/alsertnet: mlf229.49.76.70<49-9>
```

Conceptos de Clases y Objetos

```
1  //Clases Objetos
2  class Pelicula{
3      //Propiedades
4      nombre: string
5      protagonistas?: string[]
6
7      //Constructor
8      constructor(nombre:string, protagonistas?:string[]){
9          this.nombre = nombre,
10         this.protagonistas = protagonistas
11     }
12
13     //Metodos
14     proyectarPelicula():void{
15         console.log(`${this.nombre} se esta proyectando`)
16     }
17 }
18
19 let titanic = new Pelicula("titanic")
20 let vengadores = new Pelicula("End Game", ["Spiderman", "Ant Man"])
21
22 titanic.proyectarPelicula()
23 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())
```

```
// 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;
}
```

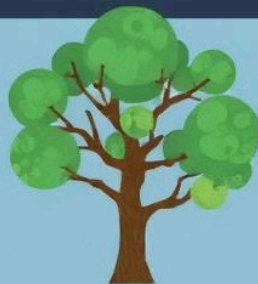

airteralvy

```
/ arrays(elleititl);  
/sargetatdet((let, spaciety riim);  
"abircdoccentes( arbackerlatestly lerf);  
/cuser(hestilate:(on(.larlayer (nathchiotan(AttAlt=all)));  
/eaplr(ust ortar/icles.ar)  
/tarerdegradeer((et((evtllits=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(Matittlstrees-  
chulcracter(insiatiIntertesber  
◊arglotofrarp-(Scret)>  
Thaleriettor(natiatTircessrr)>  
enterricicor((reth vnatilz-up lity);>  
Thalcrtotactllatlat(fircessrr)
```



linked lists

```
> cratlclackenoie jets -ralect);  
ttres nelaTclet;  
cbach tebcebetlat(:opa(Inateult)  
Frecchtores(lat(OUdlienedl);  
/reatlancolor(.drondlest; seltactlat.(rfrl);  
"railstetcoricltectDalecsing(,.fiC.).g "iree:  
/ccoplectobitlss(com/pet(.alse.pretlOI(, larl)  
real(onn(apgr(iree:;  
> reitectogcahatle(ng >ciseh rir");
```

Contenedores y Asociaciones



Arrays

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



Listas Enlazadas

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



Diccionarios

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



Árboles

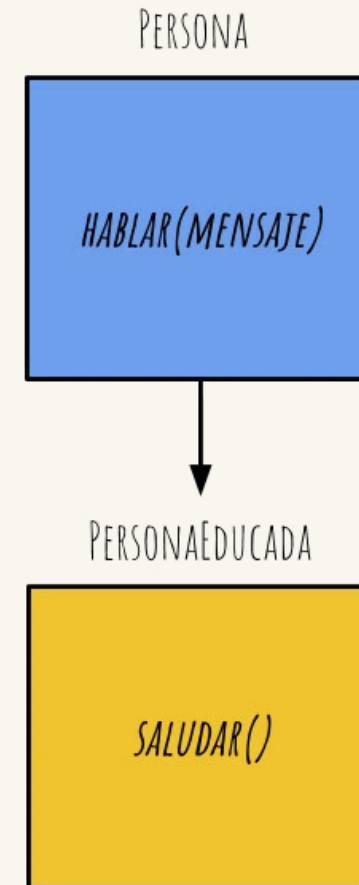
Los árboles son estructuras jerárquicas de nodos que permiten organizar y navegar datos de manera eficiente.

Herencia y Polimorfismo en TypeScript



Ejemplo de herencia en TypeScript

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

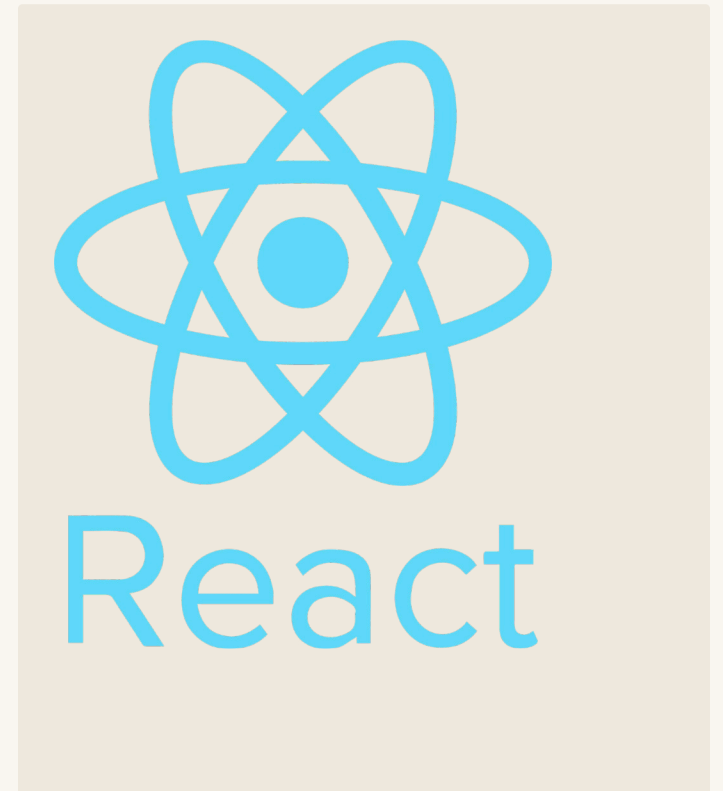


Ejemplo de Polimorfismo en TypeScript

```
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**

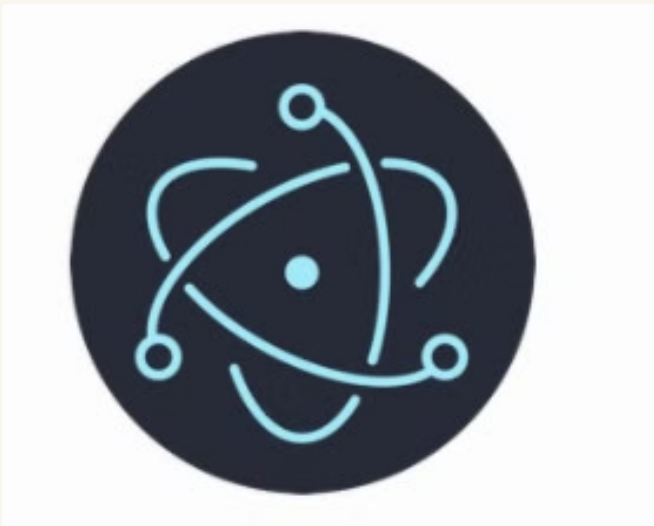
```
○ PS E:\DESCARGAS 2\Programacion Franco\CURSO TYPESCRIPT> tsc -w
```

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

Desktop, Web y Móvil

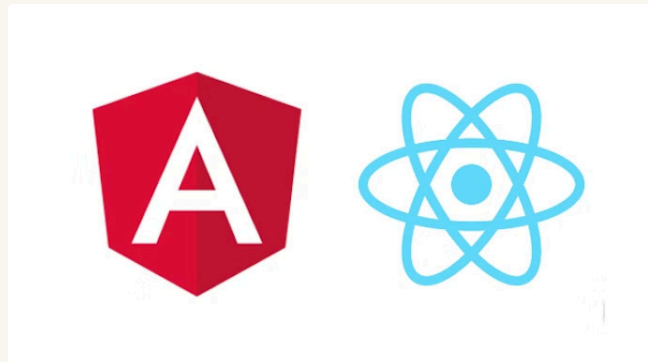
Escritorio

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.

