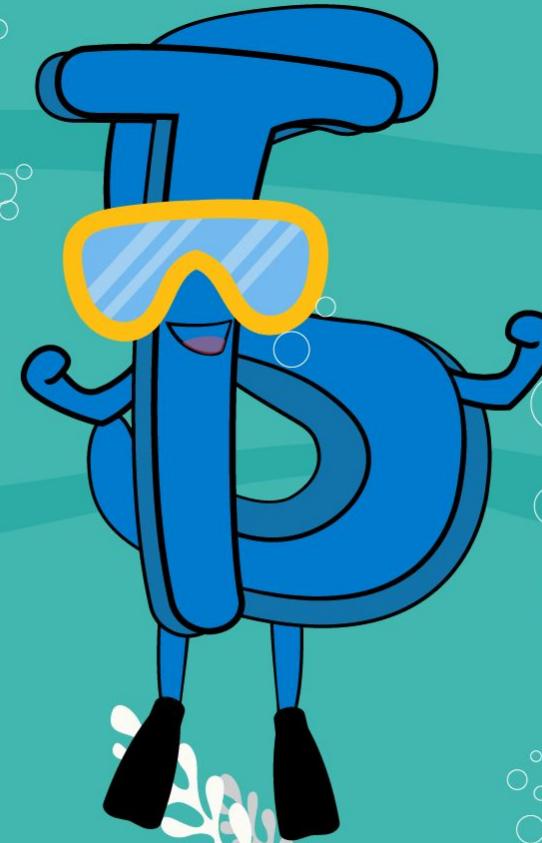


Hola!! TypeScript



**Vanessa
Aristizabal**

Frontend Developer

**Organizadora de
GDG & WTM
Medellín.**

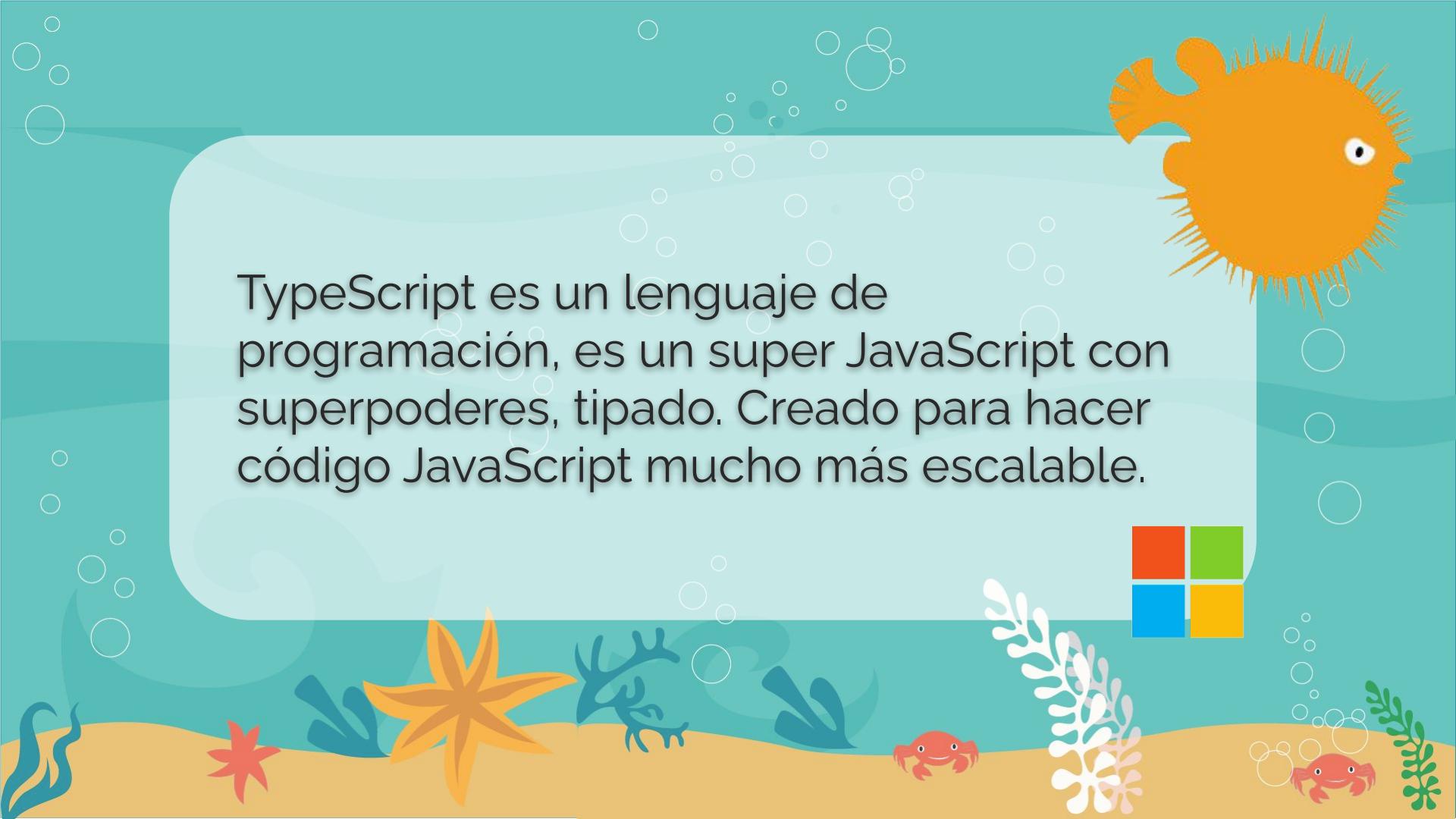
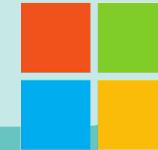
 @vanessamarely



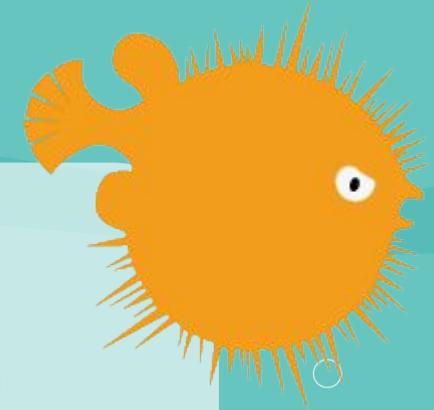
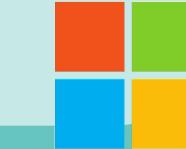
¿Qué es TypeScript?



TypeScript es un lenguaje de programación, es un super JavaScript con superpoderes, tipado. Creado para hacer código JavaScript mucho más escalable.

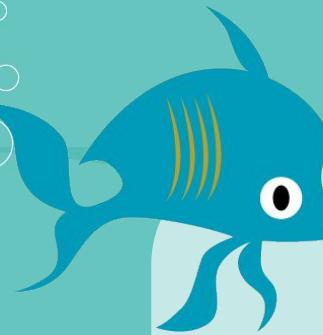


Utiliza el paradigma de programación Orientado a Objetos, donde podemos incluir clases como otros lenguajes como Java, C#, entre otros.



¿Cómo lo instalamos?



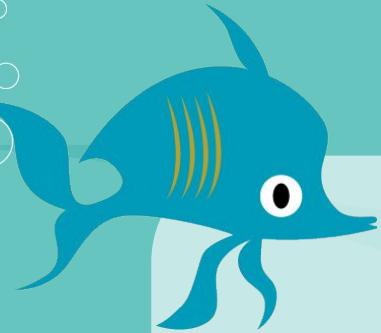


TS puede ser instalado dependiendo
de cómo lo usemos.





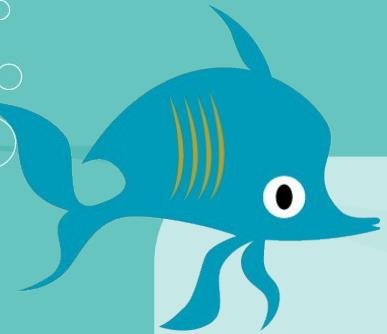
node.js



En la terminal lo instalamos:

```
npm install typescript -g
```

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



Ejecutamos el compilador:

tsc filename.ts



Visual Studio Code

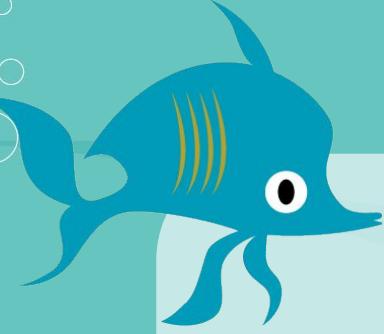
En VS Code necesitamos instalar
TypeScript



```
npm install typescript -g
```

Hello World!!!



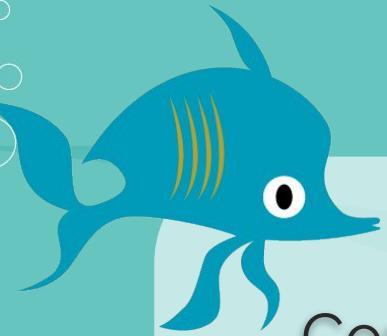


Hello World



Creamos nuestro archivo helloworld.ts

```
const message : string = 'Hello world';  
console.log(message);
```

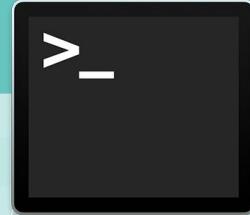
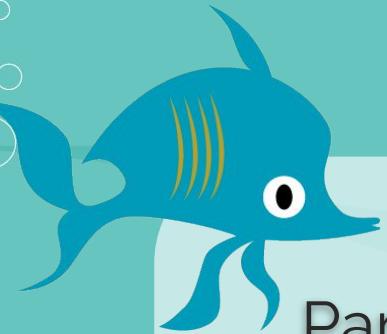


Compilamos nuestro código:

tsc helloworld.ts

Ejecutamos:

node helloworld.js

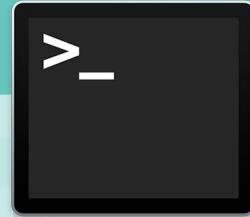
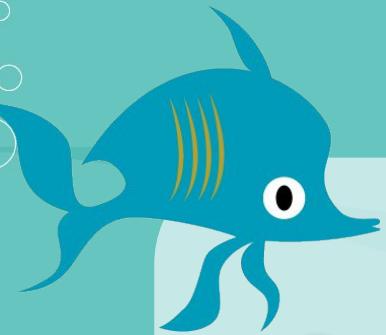


Para que el compilador se ejecute con
cada cambio

tsc helloworld.ts --watch

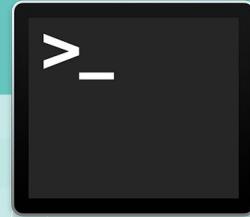
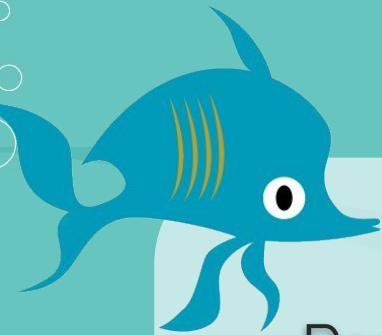
Ejecutamos:

node helloworld.js



Para crear la configuración de nuestro compilador:

```
tsc --init
```



Para colocar los archivos que deseamos compilar, modificamos nuestro archivo generado.

tsconfig.json

¿Qué es POO?





La programación orientada a objetos es un paradigma de programación que utiliza la abstracción para crear modelos basados en el mundo real.





Clase

Define las características del Objeto.

Objeto

Es la instancia de una Clase.

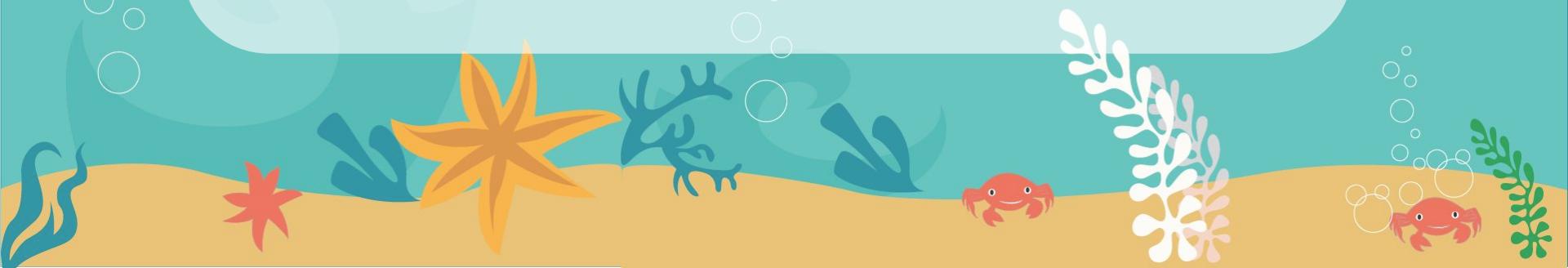


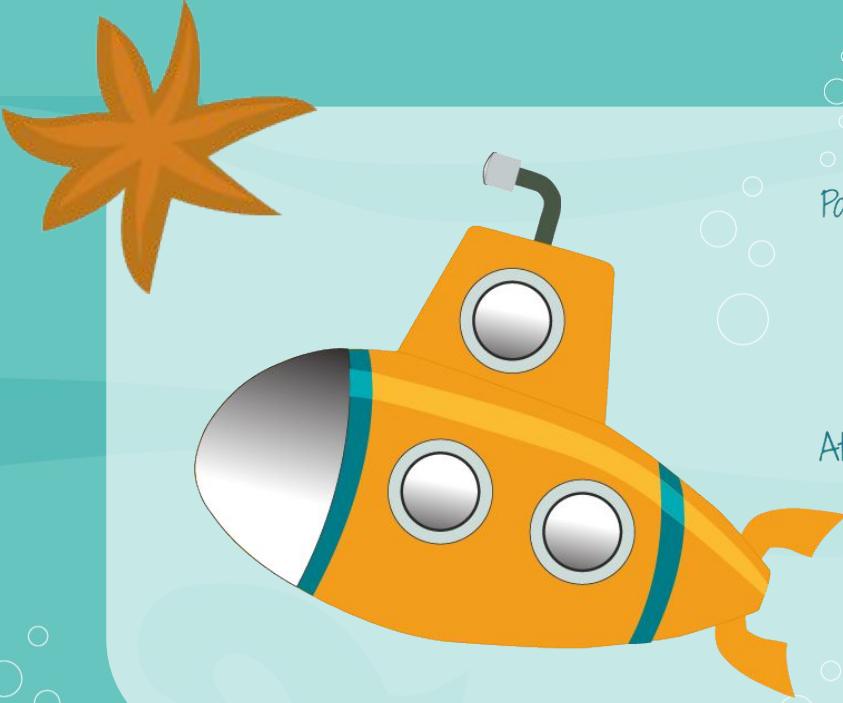
Atributos

Una característica del Objeto, como el color.

Métodos

El comportamiento de un Objeto, como caminar.





Palabra reservada Nombre de la clase

```
class Submarino {
```

nombre: string; Tipo de dato

Atributos

Métodos

```
mostrarNombre( ) : void {
```

Tipo de dato
de retorno

```
    console.log('Nombre : ' + this.nombre);
```

}

}



Constructor

Método que se llama en la creación de las instancias.



Palabra reservada Nombre de la clase

class Submarino {

 nombre: string; Tipo de dato

Atributos

 constructor

 constructor(nuevoNombre: string) {
 this.nombre = nuevoNombre;

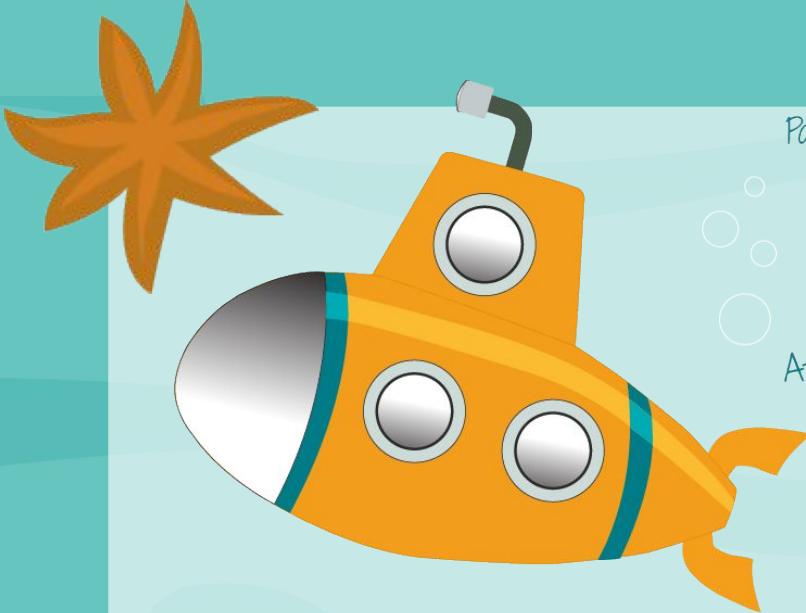
 }

const objeto1 = new Submarino('Nautilus');

Instancia clase

Abstracción

Abstraer algo hacia nuestro código para convertirlo en una clase.



```
Palabra reservada  
Nombre de la clase  
class Submarino {  
    nombre: string; // Tipo de dato  
    Atributos  
    Métodos  
    mostrarNombre( ) : void {  
        // Tipo de dato de retorno  
        console.log('Nombre : ' + this.nombre);  
    }  
}
```

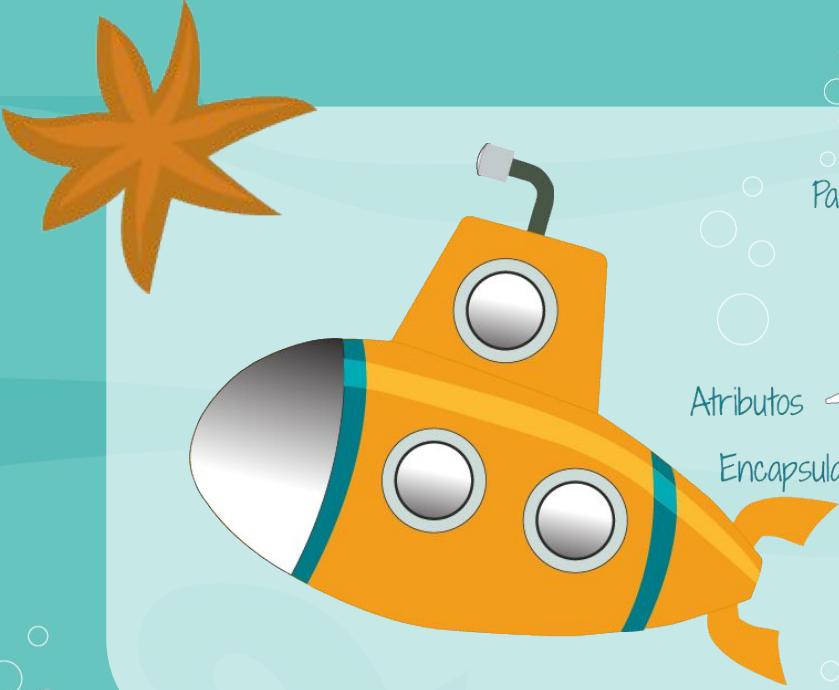
```
const objeto1 = new Submarino('Nautilus');  
objeto1.mostrarNombre();
```





Encapsulamiento

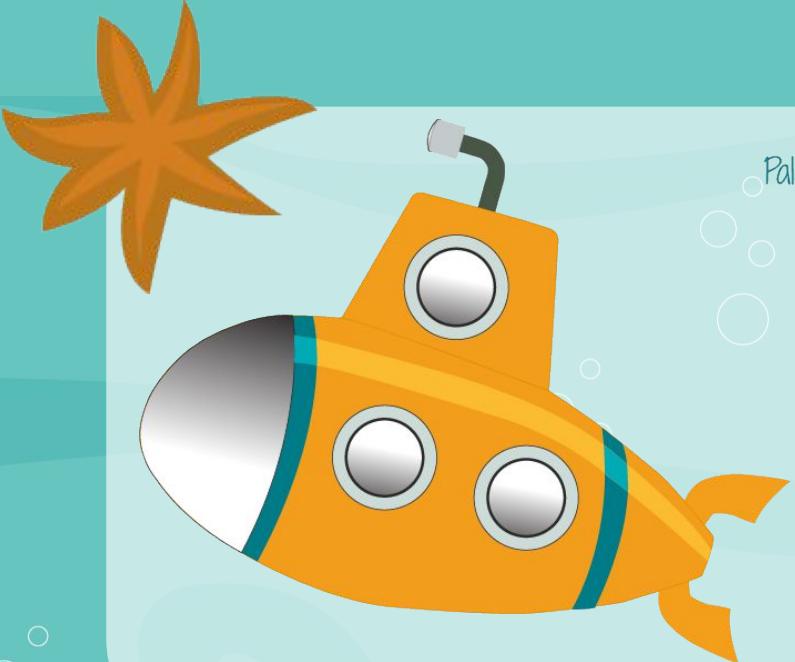
Una Clase sólo define las características del Objeto, un Método sólo define cómo se ejecuta el Método.



```
Palabra reservada      Nombre de la clase  
↓                         ↗  
class Submarino {  
    nombre: string;           ↗ Tipo de dato  
    private color: string;  
                                ↗ constructor  
    Encapsulamiento          ↗ constructor  
    constructor( nuevoNombre: string) {  
        this.nombre = nuevoNombre;  
    }  
}
```

Herencia

Una Clase puede heredar de otra Clase.



Palabra reservada

class Delta extends Submarino {

abrirEscotilla() : void {

console.log('Abrir Escotilla');

}

Hereda de la clase

```
const delta= new Delta('D5');
delta.mostrarNombre();
delta.abrirEscotilla();
```

Polimorfismo

Diferentes Clases podrían definir el mismo
método o propiedad.



```
class Submarino {  
    nombre: string;  
    color: string;  
  
    constructor(nombre: string, color: string){  
        this.nombre = nombre;  
        this.color= color;  
    }  
}
```

```
const delta= new Delta('D5', 'Azul');  
const vanguard= new Vanguard('D5', 'Azul');  
getColor( delta );  
getColor( vanguard );
```

```
class Delta extends Submarino {  
    abrirEscotilla( ): void {  
        console.log('Abrir Escotilla');  
    }  
}
```

```
class Vanguard extends Submarino {  
}
```

```
function getColor( submarino: Submarino){  
    console.log(submarino.color);  
}
```



Características





Tipado estático

- Las variables tienen un tipo de dato.
- Los valores que se asignan corresponden a su tipo de dato.

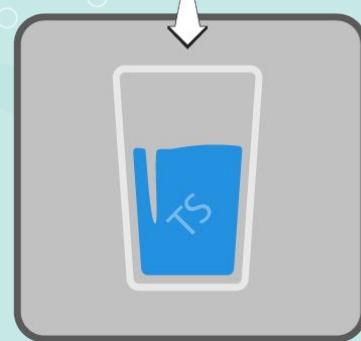


Creamos la variable

```
let nombre : string;  
nombre = 'TS';
```

Tipo de dato

Asignación de Valor





Tipos de Datos

- Booleans: tipo de dato lógico
- Number: tipo de dato numérico.
- String: tipo de dato de cadena de caracteres.



Tipos de Datos

```
let nombre : string = 'Nautilus';
const color: string = 'Rojo';
const tamano: number = 100;
const esRapido: boolean = true;
```



Tipos de Datos

- Array: colección de datos
- Tupla: es un arreglo que puede tener datos de varios tipos.
- Enum: enumeración de valores descriptivos.



Tipos de Datos

- Any: se usa cuando no queremos declarar un tipo de dato.
- Void: se usa para declarar funciones que no retornan nada.
- Null & undefined



Interfaces

Las interfaces nos ayudan a definir las variables o métodos que vamos a usar.



Palabra reservada



```
interface Pez {  
    nombre: string;  
}
```

Propiedad

Nombre de la interfaz

Tipo de dato



Type

Permite definir el tipo de dato; pero a diferencia de las interfaces no podemos extender un type, ni ampliar sus capacidades.



Palabra reservada

Nombre del type



type PezAmigable = string;

Tipo de dato



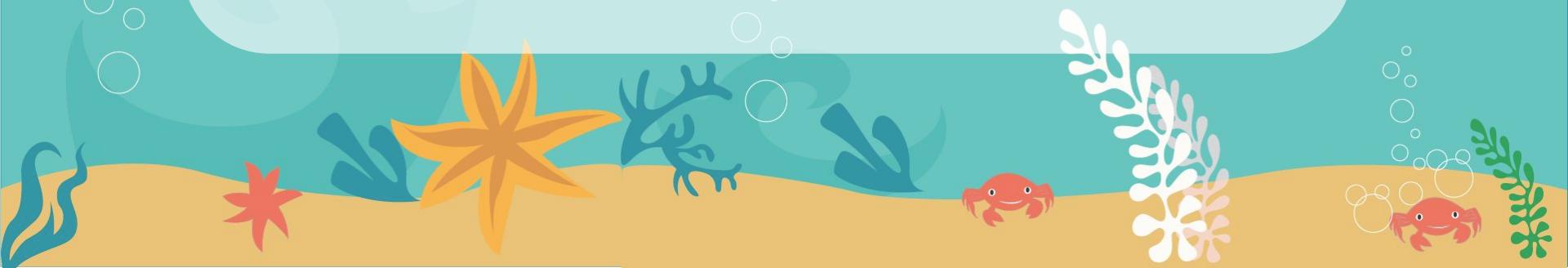
Clase

Una clase tiene atributos y métodos.
Los atributos son variables internas de la clase.
Los métodos, son las acciones a realizar dentro de la clase.





```
class MiClase {  
    /* atributos */  
    /* Métodos */  
}
```





Palabra reservada
Nombre de la clase

class Peces {

nombrePez: string; ◀ Tipo de dato

tipoPez: string;

constructor(nombrePez: string, tipoPez: string){

nombrePez = this.nombrePez;

Tipo de dato

tipoPez = this.tipoPez;

}

mostrarNombre() : void {

Tipo de dato
de retorno

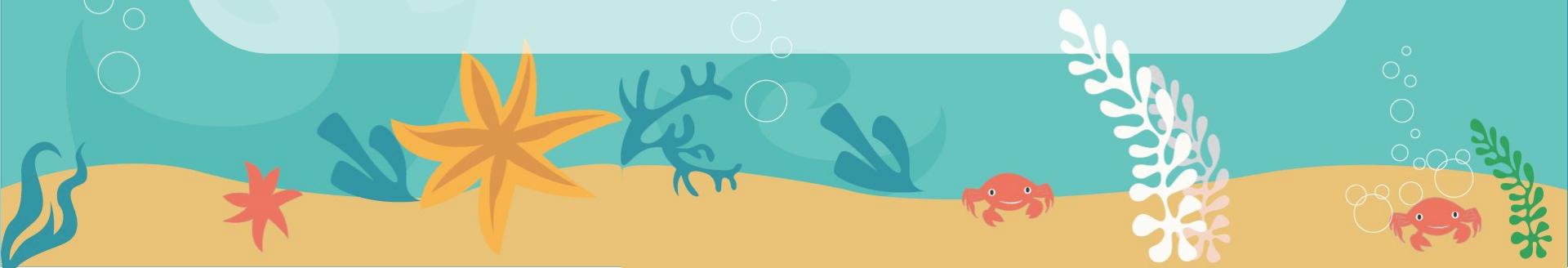
console.log('Nombre : ', this.nombrePez);

}

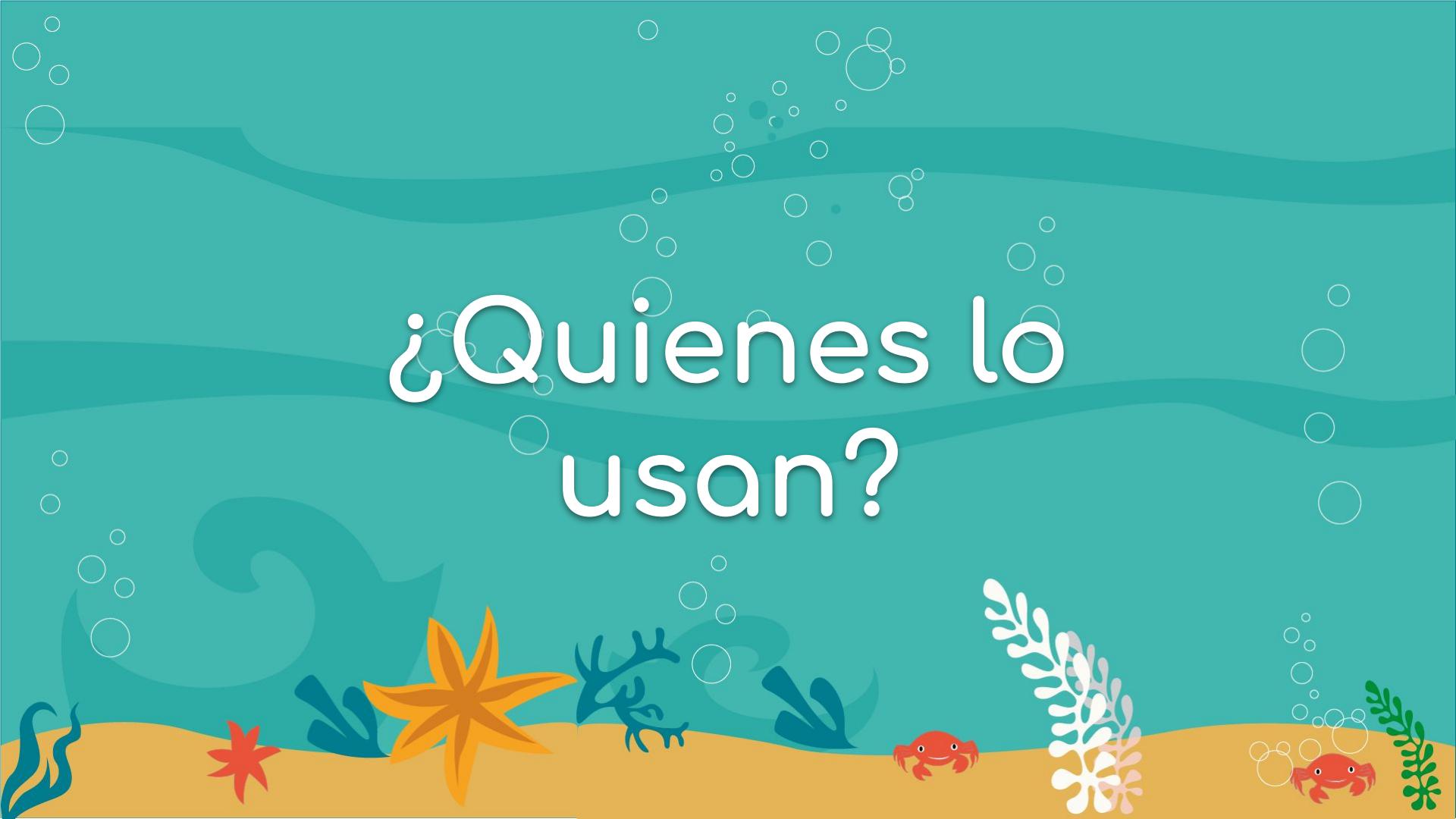


Otras características

- Argumentos con tipos de datos
- Tipos de datos en los retornos de una función

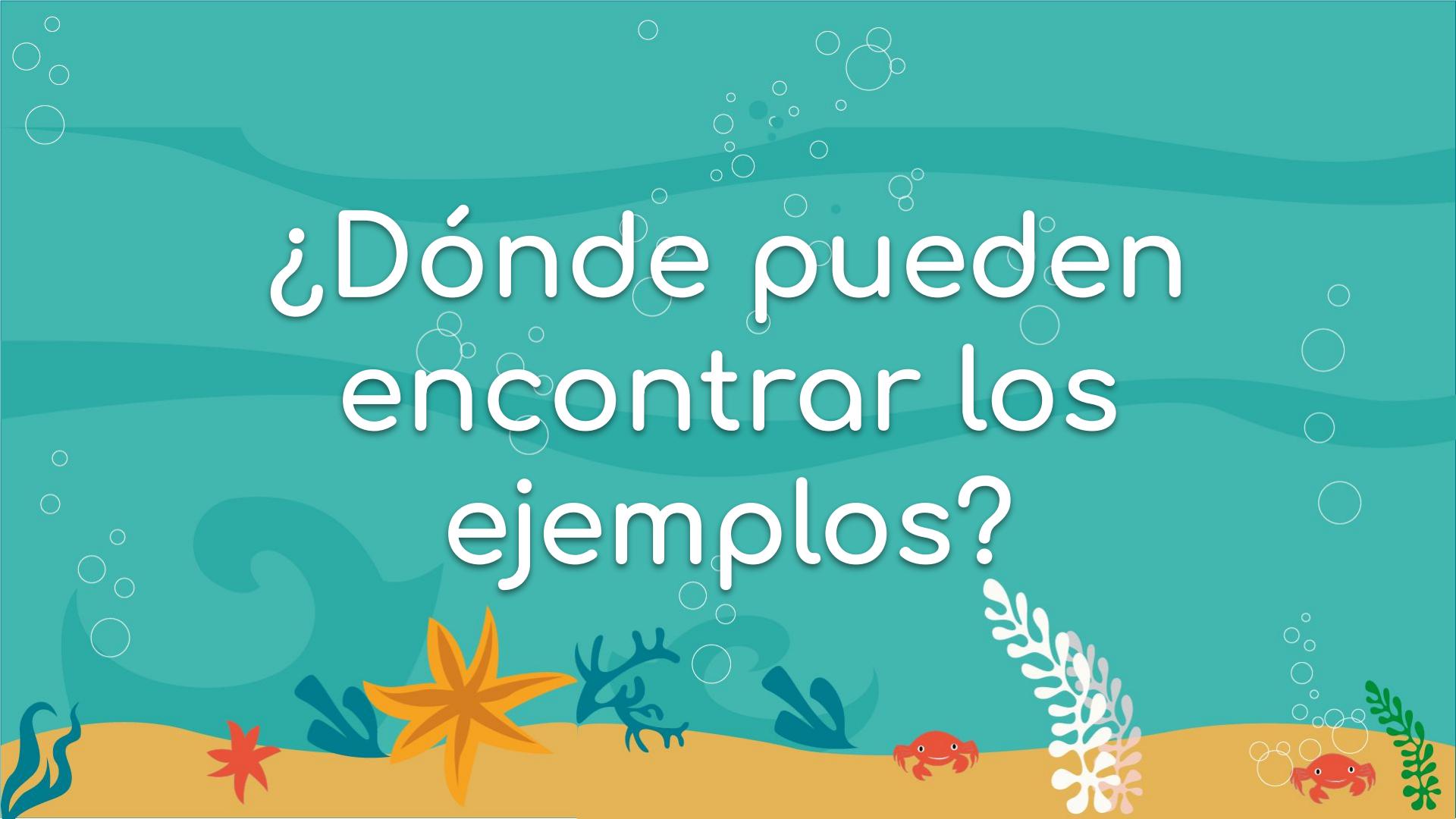


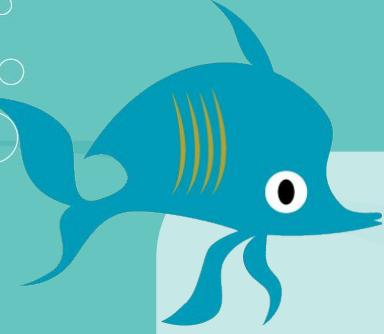
¿Quienes lo usan?





¿Dónde pueden
encontrar los
ejemplos?





Github

<https://github.com/vanessamarely/typescript-talk>

<https://github.com/AtlanticBootcamps/webinar-typescript-sep-202>

< / > ATLANTIC

Boarding to develop



Gracias!!!

