

introducción a TypeScript y ECMAScript 6 (ES6)

- uso intensivo en aplicaciones Angular
- decoradores = característica experimental en ES6, pero consolidados en Angular
- JavaScript tiene tipado débil de tipos, lo que no nos avisa de situaciones que pueden producir errores.

TypeScript

- <https://www.typescriptlang.org/>
- superconjunto de ECMAScript 6 (lo dota de mayores funcionalidades, destacando el tipado de datos)
- ficheros **.ts**, no ejecutables directamente, necesita ser compilado
 - por defecto al compilar se genera código ECMAScript 5
- definición de los tipos en los parámetros de las funciones

instalación

- `npm install -g typescript`

compilación

- compilar (tsc = TypeScript Console): `tsc <APP>`
- compilar automáticamente el fichero ante cambios: `tsc <APP> -w`
- compilar automáticamente todos los ficheros ante cambios:

```
tsc --init # genera fichero tsconfig.json
tsc -w # generará el fichero .js correspondiente para cada .ts
```

- este fichero **tsconfig.json** además:
 - versión ES que debe generarse
 - nivel exigencia de los warnings
 - compilación incremental o no
 - directorios de origen y destino
 - si se permiten características experimentales (aka decoradores)

ejemplo

app.js

```
var alumno = {
  nombre: "Joan"
}

function saludar(nombre) {
  console.log("Hola " + nombre);
}
```

```
saludar(alumno);
```

- `node app.js` → Hola [Object object]

app.ts

```
var alumno = {  
  nombre: "Joan"  
}  
  
function saludar(nombre:string) {  
  console.log("Hola " + nombre);  
}  
  
saludar(alumno);
```

tipos

- se declaran los tipos, aunque pueden ser inferidos por el valor
- number, string, boolean
- vector (array)
 - del mismo tipo

```
let myArray: number[] = [15,16];
```

```
let myArray: Array<number> = [15,16];
```

- tupla:
 - diferentes tipos

```
let tupla: [string, number, boolean] = ["Juan", 8, true];
```

- enum

```
enum Vehiculo { Coche, Camion, Bus }
```

```
let miVehiculo: Vehiculo = Vehiculo.camion;  
console.log(miVehiculo) // 1  
let otroVehiculo: string = Vehiculo[1];  
console.log(otroVehiculo) // Camion
```

- any:

```
let cantidad: any = 1; # pierde el sentido usar typescript!
```

- void

- valor vacío

```
function miFuncion(): void {  
  console.log(Math.PI * 3);  
}
```

}

- null
- undefined
- never
- object
- type assertions:
 - conversiones a convenio y responsabilidad del programador
 - aka **casting**
 - ```
let miVariable: any = "Soy una cadena";
let unValor: string <string>MiVariable;
let unValor: string = miVariable as string;
```

## clases

- definición de clases
- por defecto visibilidad pública, permite **public**, **private**, **protected**
- ```
class Greeter {
  constructor(private greeting: string) { }
  greet() {
    return "Hello, " + this.greeting;
  }
}
```

 - al usar **private**, sabe que **greeting** es un atributo de la clase
- extends: creación de subclases a partir de otra
- implements:
 - implementa un **interface**:
 - declaración sin código ni valores
 - creación de nuevos tipos
- abstract: clase abstracta no instanciable
- super: hacer referencia a la superclase desde una subclase

interfaces

- se declarar indicando las propiedades y métodos que contendran
- sin valores ni métodos:

```
interface Sumergible {
  tiempoMaxBajoElAgua: number;
  profundidadMaxima: number;

  ascender(): void;
}
```

- ```
interface Producto {
 readonly nombre: string; // solo lectura
 precio?: number; // optional
 stock: number;
}

let miProd: Producto;
```

```
miProd = { nombre: "Reloj XYZ", stock: 50 }
console.log(miProd.nombre + ": " + miProd.stock + " unidades");

miProd.nombre = "Gafas sol ABC";
console.log(miProd.nombre);
```

- como nuevo tipo:

```
interface CitaCalendario {
 fechaHora: Date;
 titulo: string;
 lugar: string;
}
let cital: CitaCalendario;
```

- readonly: solo permite una sola asignación
- ? : valor opcional

From:  
<https://matebcn.eu/> - **miguel angel torres egea**

Permanent link:  
<https://matebcn.eu/development:angular:introduccion-typescript>

Last update: **25/01/2020 19:05**

