

Desarrollo y programación de Aplicaciones Avanzadas Angular

2 TypeScript

CORE
networks



Fundamentos de TypeScript

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft.

Es un superconjunto de JavaScript que le añade características para desarrollos de gran escala, siendo sus principales:

- Tipado estático.
- Clases e interfaces para objetos.
- Soporte para las características de ECMAScript 6.

<https://www.typescriptlang.org/>



Instalación

Instalación de NodeJs y NPM

Paquete instalador para Windows en <https://nodejs.org/en/download/>

Comprobar en la consola con:

```
node -v && npm -v
```

Instalación de TypeScript

```
npm install -g typescript  
tsc -v
```

Transpilación a JavaScript

En la consola con el comando:

```
tsc archivo.ts -w
```

Sintaxis tipado estático de datos

Sintaxis básica

```
let variable: tipo;
```

El tipado es denominado gradual, es decir si TypeScript no conoce el tipo infiere su tipo de la misma manera que JavaScript.

```
let variable = 12; // inferencia a tipo number
```

Tipos de datos

string

number

boolean

tipo[] ó Array<tipo>

object ó { }

any

void

Tipos de datos de unión

Permiten varios tipos:

```
let resultado: string | number;
```

Tipos de datos en funciones

Permite el tipado de parámetros y salida de la función

```
function suma (a: number, b: number): number {  
    return a + b;  
}
```

También permite que los parámetros sean opcionales con el operador ?

```
... mensaje?: string...
```

Tipos de datos genéricos

Se identifica un tipo genérico:

```
function devResultado<T> (a:T): T {  
    return a;  
}
```

Que posteriormente es definido en la ejecución de la función:

```
let b = devResultado<string>('Aprobado');
```

Clases (ECMAScript 6)

```
class DatosJugador {  
    public nombre: string;  
    public apellidos: string;  
  
    constructor(nombre: string, apellidos: string){  
        this.nombre = nombre;  
        this.apellidos = apellidos;  
    }  
    marcarGol(): void {  
        this.goles++;  
    }  
}
```

Constructor breve

```
class DatosJugador {  
    public nombre: string;  
    public apellidos: string;  
  
    constructor(public nombre: string, public apellidos: string){  
        this.nombre = nombre;  
        this.apellidos = apellidos;  
    }  
    marcarGol(): void {  
        this.goles++;  
    }  
}
```

Diferencias con Javascript

- Tipado estático frente a tipado dinámico
- Transpilación de código para poder ser interpretado por el navegador
- Errores en tiempo de desarrollo frente a errores en tiempo de ejecución
- Mejor soporte para navegadores gracias a la transpilación.