TYPESCRIPT BÁSICO

Typescript es un superconjunto de Javascript

Para trabajar con Typescript necesitamos descargar su librería con el siguiente comando:

```
npm install -g typescript
```

Comprobamos que lo tenemos correctamente instalado con:

```
tsc --version
```

Inicializamos la carpeta en la que estamos con un proyecto de Typescript con el comando:

```
tsc --init
```

Para que Typescript transpile todos los archivos cada vez que estos cambien, escribimos en la línea de comandos.

```
tsc --watch
```

Definición de tipos: **Boolean, Number, String, Array, Tuples, Enums, Any, Void, Null, Undefined**

Definición de **variables con let**. Let es más seguro utilizarlo.

Con Typescript tenemos clases, herencia, polimorfismo y posibilidad de declarar clases abstractas.

Código ejemplo

```
// Comprobar en página https://www.typescriptlang.org/play/index.html

// *** DEFINICIÓN DE TIPOS DE DATOS ***

// Boolean
let isBoolean: Boolean = false;

// Number
let decimal: number = 43.645;
let age: number = 30;
```

```ts
// String
let myName: string = 'Samuel';

// Array
let listAges: number[] = [10, 21, 30, 26];
let listNames: Array<string> = ['Juan', 'Pedro', 'Alfredo'];

// Tuples
let personTuple: [string, number];
personTuple = ['Samuel', 30];

// Enums
enum Color { Red, Green, Yellow, Purple, Brown, Black };

// Any
let notSure: any = '345';
notSure = 24;
notSure = true;

// Void
function alertUser(): void {
    alert('Alert with function void');
}

// Null
let u: null = null;

// Undefined
let ud: undefined = undefined;

// *** FUNCIONES: PARÁMETROS Y VALOR DEVUELTO ***

function maxValue(value1: number, value2: number): number {
    return Math.max(value1, value2);
}

maxValue(2, 9);


// *** CLASES ***

// Definición de clase
class Animal {
    readonly nombre: string;
    private edad: number;
    public colorPiel: Color;
    constructor(nombre: string, edad: number, colorPiel: Color) {
        this.nombre = nombre;
        this.edad = edad;
        this.colorPiel = colorPiel;
    }

    saludo(): string {
```

```
        return `Hola, ${this.nombre}`;
    }
}

let miGato = new Animal('Benito', 15, Color.Brown);
console.log(miGato.saludo());

// Herencia : extends

class Perro extends Animal {
    saludo(): string {
        return `Guau! Guau! ${this.nombre}`;
    }
}

let miPerro = new Perro('Dobi',8,Color.Black);
console.log(miPerro.saludo());

// Interfaz
// Una interfaz es un "contrato" o modelo de datos que no tiene métodos

interface Mensaje {
    titulo: string;
    mensaje: string;
    errorNumero?: number;
}

function EscribeMensaje(mensaje: Mensaje) {
    console.log(mensaje.titulo);
}

EscribeMensaje({ titulo : 'Prueba título', mensaje : 'Texto del mensaje'});

// *** GENÉRICOS ***
/* T se transforma al tipo de dato
any no se transforma al tipo */

class Coleccion<T> {
    private elementos: Array<T>;
    constructor() {
        this.elementos = [];
    }

    Add(elemento: T) {
        this.elementos.push(elemento);
    }
}

class PersonaInfo {
    readonly nombre: string;
    constructor(nombre: string) {
        this.nombre = nombre;
    }
}
```

```
let listaNombre = new Coleccion<string>();
listaNombre.Add('Juana');
let listaNumeros = new Coleccion<number>();
listaNumeros.Add(53);
let listaPersonasInfo = new Coleccion<PersonaInfo>();
listaPersonasInfo.Add(new PersonaInfo('Moisés'));


// *** MÓDULOS Y NAMESPACE

/* Un módulo nos permite crear nuestro propio ámbito(scope). Es decir,
las variables, functions, class, enums, etc. declaradas dentro del módulo
no son visibles al exterior, excepto si las marcamos como export.
Para el consumo de las variables, functions, class, enums, etc. tenemos
que importar el módulo con import */

namespace Validador {
    export interface NumeroValidador {
        isValidNumber(valor: number): boolean;
    }

    export class CodigoValidador implements NumeroValidador {
        isValidNumber(valor: number) {
            return valor > 0 && valor < 10; // P.ej.
        }
    }
}

let validador: Validador.NumeroValidador = new Validador.CodigoValidador;
console.log(validador.isValidNumber(12)); // false
```