TYPESCRIPT

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases.

Tipos de datos

- string
- number
- boolean
- undefined
- null
- void
- never

Características

- Tipado Fuerte
- Debugging
- Interfaces
- Clases

Ejemplos

```
let carName: string = 'renault'
let fuel: number | string = 75;
let isNew: boolean = false;

carName = 'chevrolet';
fuel = 'FULL'; // o fuel = 80
isNew = true;

console.log( carName, fuel, isNew )
```

```
let carName: string = 'renault'
let fuel: number | string = 75;
let isNew: boolean = false;

carName = true; // Marca error porque
                // espera un string

fuel = 'FULL'; // o fuel = 80

isNew = 'yes'; //Marca error porque
               //espera un booleano

console.log( carName, fuel, isNew )
```

Como se observa en el ejemplo, podemos especificarle a las variables que clases de datos queremos aceptar en ellas; de esta manera minimizamos comportamientos extraños que puedan ocurrir en la aplicación, además que permite encontrar algunos errores en el mismo instante que estas escribiendo tu código.

Otra de las razones por la cual es útil typescript, es que permite auto documentar el código, de esta manera cuando otro desarrollador trabaje en nuestra aplicación ya sabrá que valores espera cada variable.

Fuentes: Wikipedia, dev.to

Arreglos

```
let caracteristicas: any[]
let caracteristicas = []
```

```
let caracteristicas: string[] = ['rojo', '4 puertas', 'deportivo']
```

Los array por defecto vienen con el tipo de dato any[], este tipo de dato hace referencia a que acepta cualquier tipo de dato que se le agregue en su índices

Si queremos que solo acepte por ejemplo solo valores tipo string debemos decírselo como se observa en la imagen al costado.

Objetos

```
interface Car {
    name: string;
    fuel: number;
    features: string[];
}

const car: Car = {
    name: 'Renault',
    fuel: 100,
    features: ['rojo', '4 puertas', 'deportivo']
}
```

```
interface Car {
    name: string;
    fuel: number;
    features: string[];
    model?: number;
}

const car: Car = {
    name: 'Renault',
    fuel: 100,
    features: ['rojo', '4 puertas', 'deportivo']
}
```

Para tipar un objeto en typescript es necesario utilizar interfaces, estas interfaces permiten decirle a nuestro código como van a lucir nuestros objetos; es decir, permite decirle que propiedades va a tener y además de que tipo van a ser estas interfaces

En la imagen que se encuentra a un costado, se puede observar que la propiedad "model" tiene un signo de interrogación, esto es para decirle a typescript que esa propiedad puede ser opcional

Funciones

```
const sum = (a: number, b: number): number => {
    return a + b
}

console.log(sum(5, 3))
```

Typescript permite que podamos tipar los argumentos de las funciones y también, el tipo de dato que va a retornar la función. esto es útil porque podemos controlar de mejor manera como se quiere que trabajen dichas funciones, y es mas fácil conocer cual es el funcionamiento de dicha función.