TypeScript (parte8 Genéricos)

LABORATORIO DE COMPUTACIÓN III

UTNFRA – TÉCNICO SUPERIOR EN PROGRAMACIÓN --

Genéricos

En JavaScript no existen los genéricos porque en JavaScript todo es dinámico.

```
2
3   function imprimeConsola ( parametro ){
4       console.log(parametro);
5   }
6
7   imprimeConsola( 123 );
8   imprimeConsola( new Date() );
9   imprimeConsola( {} );
10  imprimeConsola( {nombre: "Bruno"} );
11
```

Creando funciones genéricas

```
function regresar( parametro:any ){
    return parametro;
}

console.log( regresar( 15.4356 ).          );
console.log( regresar( "Ricardo Tapia")    );
console.log( regresar( new Date())         );
```

Si coloco un punto. TypeScript no me Puede ayudar porque no sabe que devuelve la función regresar

Si como programador se lo que va a retornar , en este caso un número puedo escribir alguna función que posean los number.

```
function regresar( parametro:any ){
    return parametro;
}

console.log( regresar( 15.4356 ).toFixed(2) );
console.log( regresar( "Ricardo Tapia")     );
console.log( regresar( new Date())          );
```

```
function regresar( parametro:any ){
    return parametro;
}

console.log( regresar( 15.4356 ).toFixed(2) );
console.log( regresar( "Ricardo Tapia").toFixed(2));
console.log( regresar( new Date())          );
```

Si me equivoco TypeScript no me va a marcar el error Y esto no va a funcionar porque los string no tienen El método toFixed()

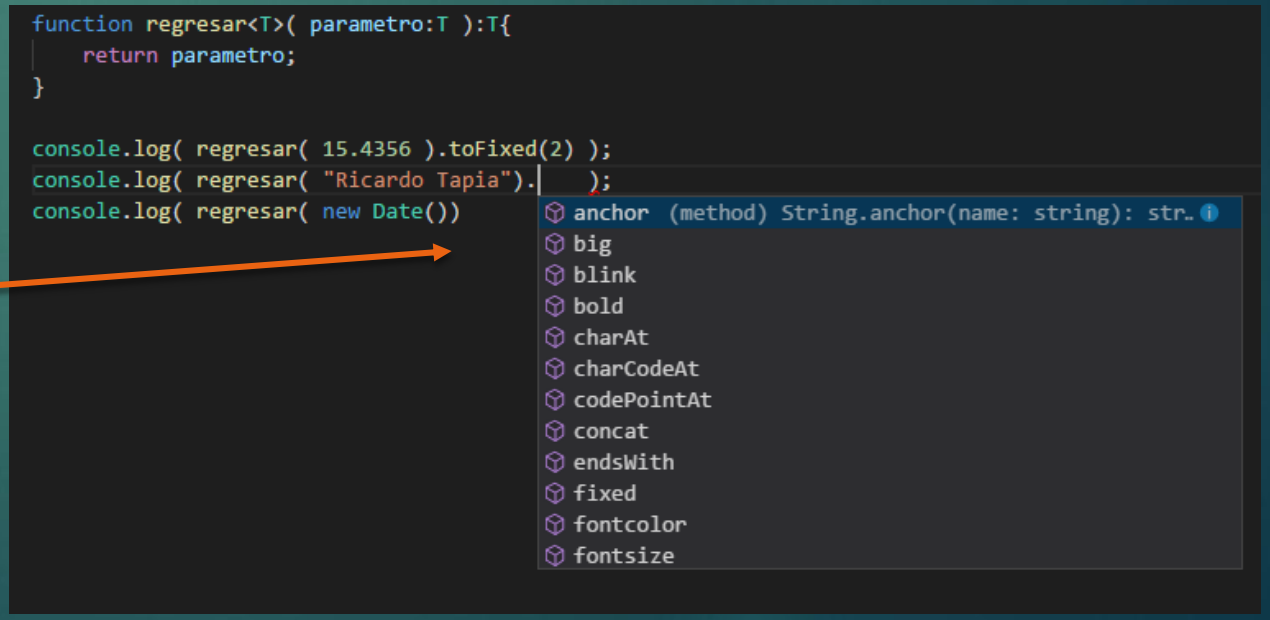
Función genérica

```
function regresar<T>( parametro:T ){
    return parametro;
}

console.log( regresar( 15.4356 ).toFixed(2) );
console.log( regresar( "Ricardo Tapia").toFixed(2));
console.log( regresar( new Date())          );
```

Ahora que la función es genérica si me indica que hay un error

Cuando coloco el punto me indica los posibles métodos de acuerdo con el tipo de dato



Utilizando una función genérica

```
function funcionGenerica<T>( parametro:T){
    return parametro;
}

type Heroe = {
    nombre: string,
    nombreReal: string
}

type Villano = {
    nombre: string,
    poder: string
}

let deadpool ={
    nombre: "Deadpool",
    nombreReal: "Wade Winston Wilson",
    poder: "Regeneración"
};

console.log( funcionGenerica(deadpool). );
```

nombre
nombreReal
poder

```
let deadpool ={
    nombre: "Deadpool",
    nombreReal: "Wade Winston Wilson",
    poder: "Regeneración"
};

console.log( funcionGenerica<Heroe>(deadpool).);
```

nombre
nombreReal

```
let deadpool ={
    nombre: "Deadpool",
    nombreReal: "Wade Winston Wilson",
    poder: "Regeneración"
};

console.log( funcionGenerica<Villano>(deadpool).);
```

nombre
poder

Arrays genéricos

```
// Array Generico
let Heroes: Array<string> = ["Flash", "Superman", "Batman"];

// Array Explicito
let villanos:string[] = ["Lex Luthor", "The Joker"];
```

Clases genéricas

```
4   class Rectangulo {
5       base;
6       altura;
7       area():number{
8           return this.base * this.altura;
9       }
10  }
11
12  let rectangulo = new Rectangulo();
13
14  rectangulo.base = 10;
15  rectangulo.altura = 10;
16
17  console.log(rectangulo.area());
18
```

Si queremos hacer genérica
Nuestra clase, nos dice que
La operación aritmética puede

```
23  class Rectangulo<T>{
24      base<T>;
25      altura<T>;
26      area():number{
27          return this.base * this.altura;
28      }
29  }
30
31  let rectangulo = new Rectangulo();
32
33  rectangulo.base = 10;
34  rectangulo.altura = 10;
35
36  console.log(rectangulo.area());
```

Clase normal

```
23  class Rectangul
24      base<T>;
25      altura<T>;
26      area():numb
27          return this.base * this.altura;
28      }
29  }
30
```

[ts] The left-hand side of an arithmetic operation must be of type 'any', 'number' or an enum type.

(method) Rectangulo<T>.base<T>(): any