Crear TypeScript Generics poco a poco se convertirá en algo muy común. El lenguaje cada día tiene mayor tracción y el uso de Genéricos en cualquier lenguage compilado es muy importante. Vamos a ver un ejemplo sencillo de como manejar clases genéricas utilizando Typescript. Para ello partiremos de dos clases muy sencillas Galleta y Golosina :

```
export class Galleta{

  sabor:string;
  constructor(sabor:string) {

    this.sabor=sabor;

  }

}
```

```
export class Golosina {

  nombre:string;
  sabor:string;

  constructor(nombre:string,sabor:string) {
    this.nombre=nombre;
    this.sabor=sabor;
  }
```

```
}
```

Como podemos ver ambas clases son muy parecidas. Vamos a crear ahora un par de clases que se denominen BolsaGalleta y BolsaGolosina que nos permitan almacenar Galletas y Golosinas.

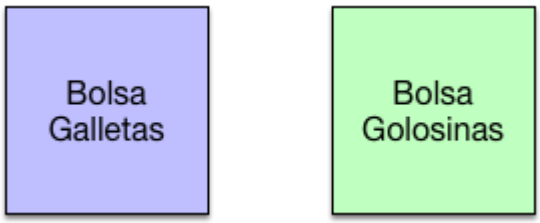
```
import {Galleta} from "./Galleta";
export class BolsaGalletas {

  lista:Array<Galleta>= new Array<Galleta>();


  add(galleta:Galleta) {

    this.lista.push(galleta);
  }


  remove(galleta:Galleta) {

    var index = this.lista.indexOf(galleta, 0);
    if (index > -1) {
     this.lista.splice(index, 1);
  }
}

  forEach(fn:(value: Galleta, index: number, array:
Galleta[])=>void):void {

    this.lista.forEach(fn);
```

```
  }

}
```

```
import {Golosina} from "./Golosina";

export class BolsaGolosinas {

  lista:Array<Golosina>= new Array<Golosina>();


  add(golosina:Golosina) {

    this.lista.push(golosina);
  }

  remove(golosina:Golosina) {

    var index = this.lista.indexOf(golosina, 0);
    if (index > -1) {
     this.lista.splice(index, 1);
  }
}

  forEach(fn:(value: Golosina, index: number, array:
Golosina[])=>void):void {

    this.lista.forEach(fn);
```

```
  }
}
```

Es más que evidente que las bolsas son muy parecidas y que únicamente cambia el tipo de la bolsa.

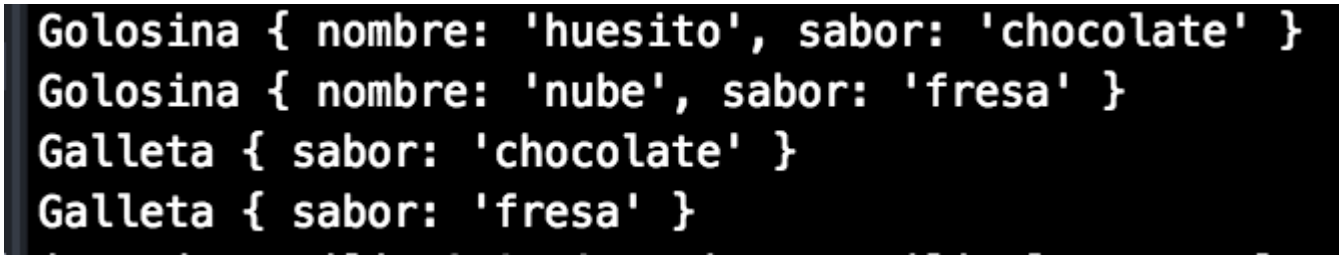


Vamos a ver el código del programa main que se encargaría de crear estas bolsas y añadir elementos a ellas.

```
import {BolsaGolosinas} from "./BolsaGolosinas";
import {BolsaGalletas} from "./BolsaGalletas";
import {Golosina} from "./Golosina";
import {Galleta} from "./Galleta";

let bolsaGolosinas= new BolsaGolosinas();
let g1= new Golosina("huesito","chocolate");
let g2= new Golosina("nube","fresa");

bolsaGolosinas.add(g1);
bolsaGolosinas.add(g2);
```

```
bolsaGolosinas.forEach(function(elemento) {

  console.log(elemento);
});

let bolsaGalletas= new BolsaGalletas();
let galleta1= new Galleta("chocolate");
let galleta2= new Galleta("fresa");

bolsaGalletas.add(galleta1);
bolsaGalletas.add(galleta2);


bolsaGalletas.forEach(function(elemento) {

  console.log(elemento);
});
```

Ejecutamos con node:



TypeScript Generics

Es evidente que el código esta fuertemente compartido y es prácticamente idéntico Por lo

tanto estamos ante un problema que puede ser solventado utilizando genéricos. Vamos a ver como definir una clase bolsa Genérica en TypeScript.

```
export class Bolsa<T> {

  lista:Array<T>= new Array<T>();


  add(elemento:T) {

    this.lista.push(elemento);
  }

  remove(elemento:T) {

    var index = this.lista.indexOf(elemento, 0);
    if (index > -1) {
     this.lista.splice(index, 1);
  }
  }


  forEach(fn:(value: T, index: number, array: T[])=>void):void {

    this.lista.forEach(fn);
  }

}
```

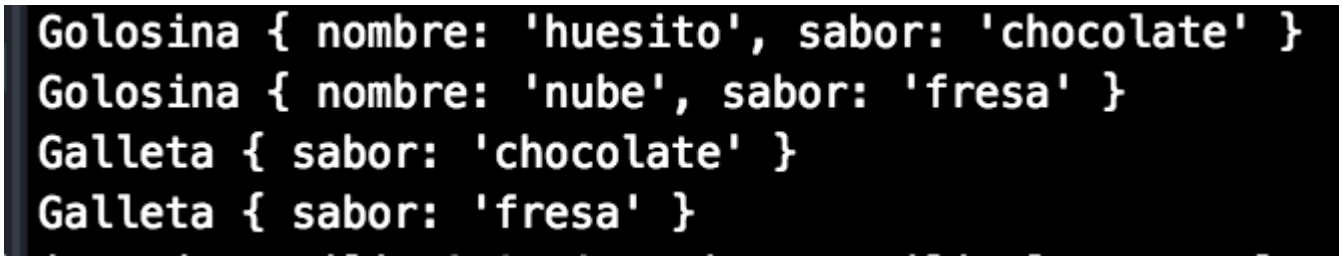
TypeScript Generics y Main

Es momento de utilizar esta clase genérica en nuestro programa Main para que veamos como utilizarla:

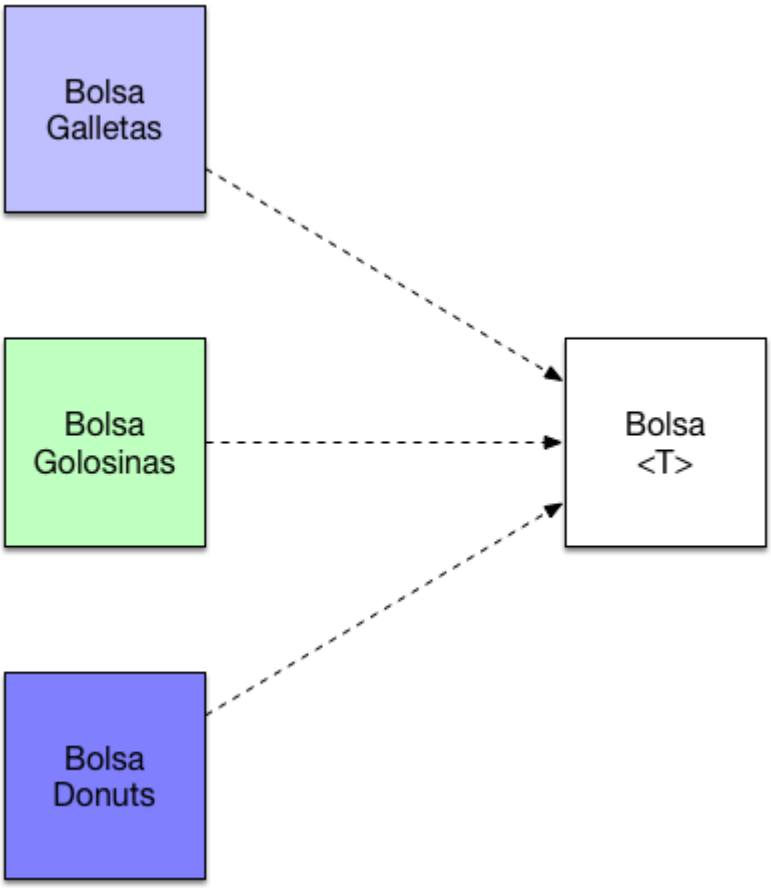
```
import {Bolsa} from "./Bolsa";
import {Golosina} from "./Golosina";
import {Galleta} from "./Galleta";
let bolsa= new Bolsa<Golosina>();

let g1= new Golosina("huesito","chocolate");
let g2= new Golosina("nube","fresa");
bolsa.add(g1);
bolsa.add(g2);

bolsa.forEach(function(elemento) {

  console.log(elemento.nombre);
  console.log(elemento.sabor);
});


let bolsa2= new Bolsa<Galleta>();

let galleta1= new Galleta("chocolate");
let galleta2= new Galleta("fresa");
bolsa2.add(galleta1);
bolsa2.add(galleta2);
```

```
bolsa.forEach(function(elemento) {


  console.log(elemento.sabor);
});
```

El resultado será idéntico :



Acabamos de simplificar nuestro código utilizando TypeScript Generics .



Otros artículos relacionados:

1. 10 Características que me gustan de TypeScript
2. 10 Atom plugins imprescindibles
3. Angular 5 Hello World y su funcionamiento
4. Angular async pipe y observables

Externos:

1. TypeScript