Tabla de Contenidos

- TypeScript Funciones
- TypeScript CallBack
- Otros artículos relacionados

El concepto de TypeScript CallBack no tiene nada de innovador ya que see trata de manejar los callbacks clásicos de Javascript pero desde nuestro nuevo lenguaje. Para ello lo primero que tenemos es que repasar el concepto de función de JavaScript . Para ello usaremos la función de suma que es la más básica.

```
function suma (a,b) {

  return a+b;
}
```

Esto no tiene nada de especial simplemente suma 2 elementos imprimimos por la consola la invocación a la función.

```
console.log(suma(2,2));
```

TypeScript Funciones

Es una función válida a nivel de TypeScript ya que recordemos que un código javascript válido es también un código valido de TypeScript. Sin embargo podríamos haberlo hecho mejor definiendo una función fuertemente tipada.

```
function suma2(a:number,b:number):number {


  return a+b;
}
```

En este caso la función necesita que le pasemos los parámetros numéricos y sino el compilador de typescript nos avisará . Es claramente una ventaja. Ahora bien recordemos que javascript es un lenguaje funcional y en muchos casos soporta funciones que reciben otras funciones como parámetros. Un ejemplo sencillo es:

```
function operar(a,b,f) { return f(a,b); }
```

En este caso podemos operar sobre los parametros a,b con la función f.

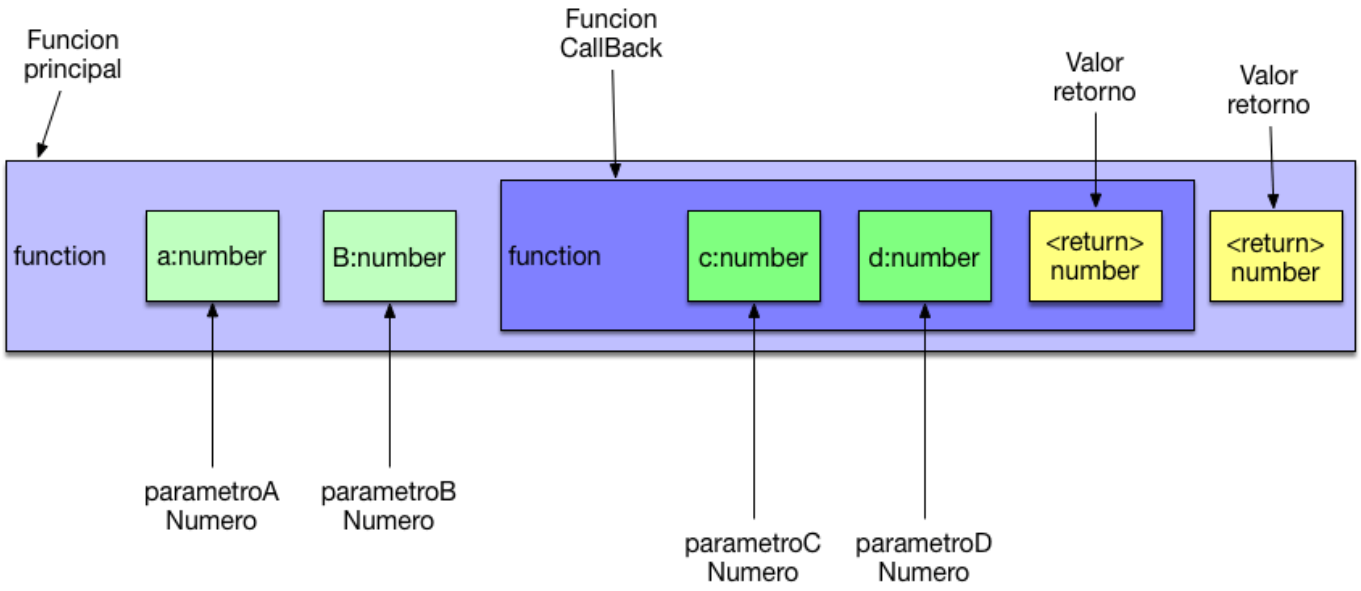
```
console.log(operar(2,2,suma));
```

TypeScript CallBack

El resultado por la consola será el mismo y el código lo hemos construido con javascript. Ahora bien no hemos usado sintaxis de tipado fuerte de TypeScript. ¿Como podríamos hacer algo de esto?.

```
var resultado=function
operar2(a:number,b:number,f:(c:number,d:number)=> number ): number {
  return f(a,b);
}
```

La sintaxis nos puede resultar en un principio extraña ya que uno de los parámetros que definimos es una función que tiene su firma en Typescript. El siguiente diagrama clarifica un poco:



Nos queda simplemente invocarla:

```
console.log(resultado(2,2,suma));
```

El resultado por la consola será 4. Acabamos de definir un callback en TypeScript utilizando los parámetros adecuados.

Otros artículos relacionados

1. TypeScript interface utilizando Angular DTOs
2. TypeScript Generics y su funcionamiento
3. 10 Características que me gustan de TypeScript

Referencias TypeScript

TypeScript