

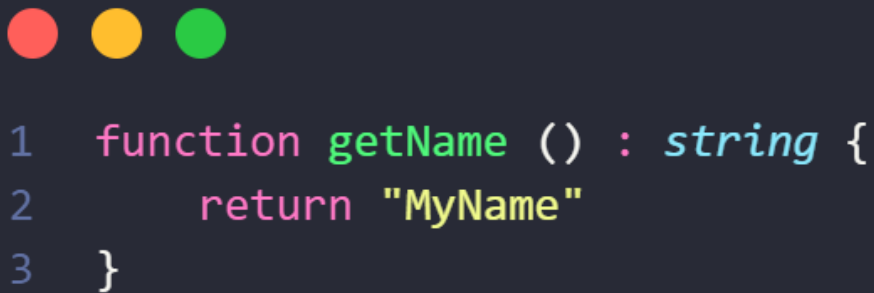


FUNCIONES EN TYPESCRIPT



DECLARACIÓN DE FUNCIONES

Para declarar funciones en TS es de la misma manera que en JS. Utilizando la palabra reservada `function` o con funciones flecha. Pero con la diferencia que puedo poner el tipo de dato que retornará mi función.

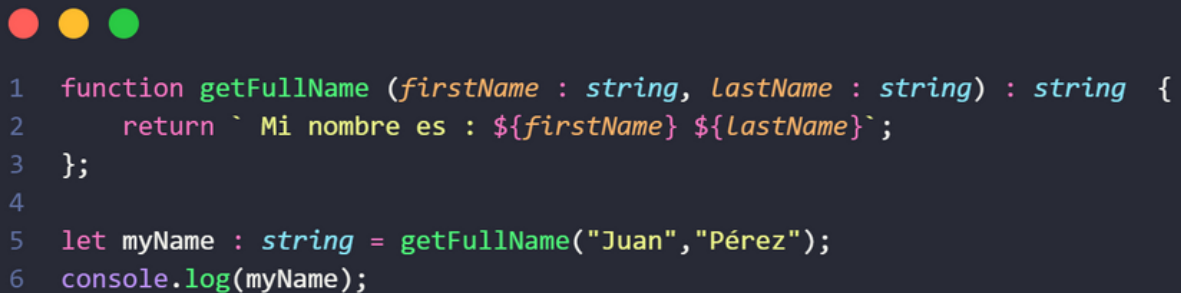


```
1  function getName () : string {  
2      return "MyName"  
3  }
```

Le indico a la función que retornará un string en este caso. Sin embargo puede ser otro tipo de dato, por ejemplo un booleano, un number, un arreglo, una tupla, void, etc...

PARÁMETROS OBLIGATORIOS DE LAS FUNCIONES

Como sabemos podemos pasarle argumentos a nuestras funciones, tanto funciones con la palabra reservada `function` como en funciones flecha. En TypeScript podemos crear funciones con parámetros obligatorios, es decir que tienen que ser del tipo de dato que especifiquemos, puede ser un string, un boolean, un array, etc...



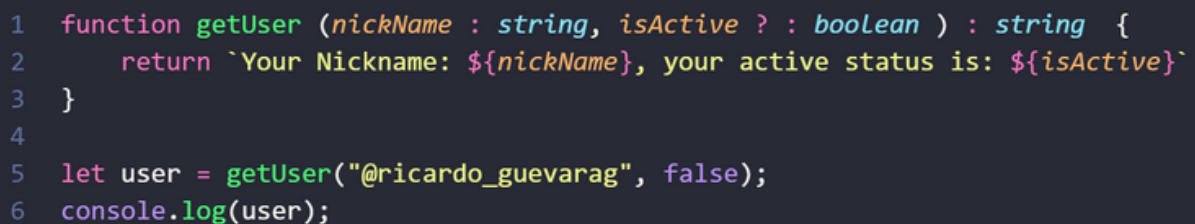
```
1  function getFullName (firstName : string, lastName : string) : string {  
2      return ` Mi nombre es : ${firstName} ${lastName}`;  
3  };  
4  
5  let myName : string = getFullName("Juan","Pérez");  
6  console.log(myName);
```

En este ejemplo declaramos una función que retorna un string y recibe como parámetro 2 strings.

PARÁMETROS OPCIONALES DE LAS FUNCIONES

En TypeScript de igual manera puedo indicar que algunos parámetros de las funciones son opcionales, es decir que no ocurrirá ningún error si no los pongo a la hora de ejecutar la función y tendrán como valor por defecto [undefined](#).

Para indicar que un parámetro es opcional basta con ponerle un signo de interrogación antes de declarar el tipo de dato.

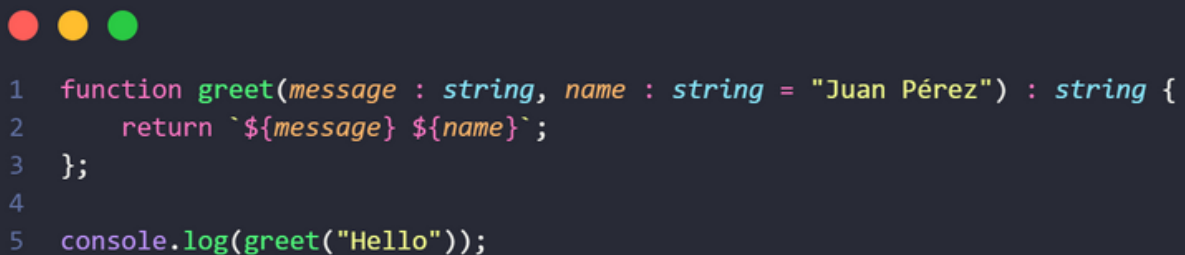


```
1 function getUser (nickName : string, isActive ? : boolean ) : string {  
2     return `Your Nickname: ${nickName}, your active status is: ${isActive}`  
3 }  
4  
5 let user = getUser("@ricardo_guevarag", false);  
6 console.log(user);
```

En este ejemplo le indico a la función que el parámetro `isActive` es opcional. Como buena práctica los valores opcionales deben ir al final de todos los parámetros, de lo contrario TypeScript arrojará un error.

PARÁMETROS POR DEFECTO EN LAS FUNCIONES

Dentro de los parámetros de mi función puedo indicar que habrá un valor por defecto, simplemente agregando el tipo de dato e inicializando el valor del parámetro le indico a TypeScript que es un valor por defecto.

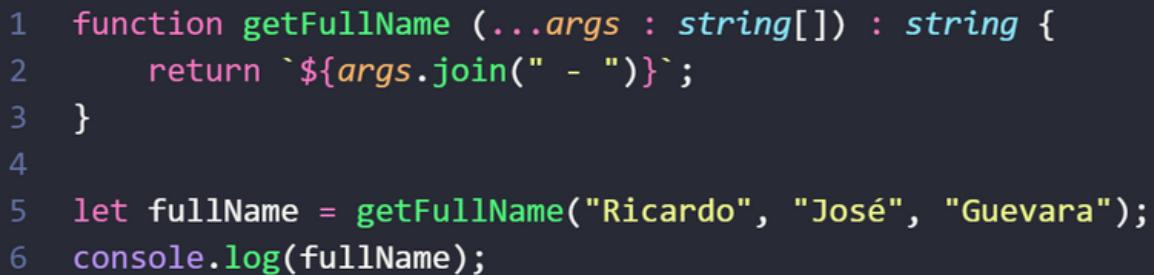


```
1 function greet(message : string, name : string = "Juan Pérez") : string {  
2     return `${message} ${name}`;  
3 };  
4  
5 console.log(greet("Hello"));
```

En este ejemplo le indico a la función que el valor por defecto del parámetro name es Juan Pérez, por lo cual no necesito pasarle un valor a la hora de ejecutar la función, pasando solamente el valor del primer parámetro.

PARÁMETROS REST EN LAS FUNCIONES

Las funciones me permiten tener como argumento el operador spread [...], para devolver todos los parámetros que yo le pase cuando ejecute mi función. Por defecto esto tendrá como tipo de dato un arreglo de lo que sea: `any[]`, a menos que yo especifique un tipo de dato en específico.




```
1 function getFullName (...args : string[]) : string {  
2     return `${args.join(" - ")}`;  
3 }  
4  
5 let fullName = getFullName("Ricardo", "José", "Guevara");  
6 console.log(fullName);
```

En este ejemplo le indico a mi función que los argumentos que recibirá será un array de strings y que retornará un string. De esta manera le puedo pasar los argumentos que yo quiera y se almacenarán en el array.

VARIABLES DE TIPO FUNCTION

Las variables de tipo `Function` son aquellas que únicamente se les asignará y reasignará una función como valor.



```
1  let myFunction : Function;
2  const increment = (a : number, b : number) : number => a + b;
3  const decrement = (a : number, b : number) : number => a - b;
4
5  myFunction = increment;
6  console.log(myFunction(4,4));
7
8  myFunction = decrement;
9  console.log(myFunction(5,2));
```

En este ejemplo le indico a la variable `myFunction`, que solo puede recibir como valor una función. Entonces le asigno la primer función que es `increment` y después le reasigno el valor con la función `decrement`, lo cual es completamente permitido porque ambas son funciones.