TypeScript Functions – Examples

TypeScript Functions

Functions are the basic building blocks for any functional programming language. They make code readable, modular and maintainable.

Functions are logical blocks of code and they could be used to define smaller tasks in a bigger application.

There are two notable partitions in a function. They are :

1. **Function Declaration** : It provides information about the identifier to reference the function, parameters it can accept and the datatype that it can return.
2. **Function Definition** : It is the actual body of the function where the parameters are transformed and the result is returned.

Syntax

Following is the syntax to declare and define a function :

```
function function_name (parameter1[:type], parameter2[:type]) [:return_type] {
    // set of statements
    // that form the function body
}
```

- **function** is the keyword to declare a TypeScript function.
- **function_name** is used to call the function for performing the task (set of statements).
- **parameters** are optional and are comma separated.
- **return_type** of the function is optional. Based on the return statements inside the function body, TypeScript can infer the return type of the function.

Function Call

Using the function name provided in the declaration, the function could be called by providing the arguments (if any). This is called as function call.

Example – Simple Function

Following is a simple function where we compute sum of two numbers and return the result. We have mentioned all the datatypes of variables we used, datatypes of parameters, and the return type as well. Then we made a call to the function with arguments 10, 12.

example.ts

```
function sum(a:number, b:number) :number {
    var result:number = a+b
    return result
}

var c:number = sum(10, 12)

console.log(c)
```

Output

```
```

TypeScript can infer the types of variables and also the return type of function. So, when the above program is compiled to JavaScript code, following is the .js file.

example.js

```
function sum(a, b) {
    var result = a + b;
    return result;
}
var c = sum(10, 12);
console.log(c);
```

Default values to parameter of function

Default values can be assigned to parameters in function declaration. When an argument is not passed during function call, respective default value is considered.

Parameters with default values are usually declared at the last of parameters.

Using the same sum function, we shall demonstrate how to specify default parameters.

Example

example.ts

```
function sum(a, b=2) {
    return a+b
}

sum(10,12)  // a=10, b=22, result = 10 + 22 = 22
sum(10)     // a=10, b=2,  result = 10 + 2 = 12
```

Rest Parameters

Rest Parameters help to provide variable number of arguments during function call. But note has to be made that the arguments passed should be of same datatype.

Example

We shall use the same sum function, but with rest parameters.

example.ts

```
function sum(...values:number[]) :number {
    var result:number = 0
    for(var i in values)
        result += values[i]
    return result
}

sum(41,52,6,1,85,32)
```

Output

```
217
```

When the above function is compiled to JavaScript,

example.js

```
function sum() {
    var values = [];
    for (var _i = 0; _i < arguments.length; _i++) {
        values[_i] = arguments[_i];
    }
    var result = 0;
    for (var i in values)
        result += values[i];
    return result;
}
sum(41, 52, 6, 1, 85, 32);
```

Optional Parameters

Optional Parameters, by the name, are not mandatory to be passed in a function call. To declare a parameter as optional, question mark (?) has be followed after the name of the parameter in the function declaration.

Example

In the following example, a and b are mandatory parameters that has to be passed during function call. But c is an optional parameter.

example.ts

```
function sum(a,b,c?) :number {
    // a and b are mandatory
    // c is optional
    return a+b+c
}

sum(41,52)    // returns 93
sum(41,52,6)  // returns 99
```

Compiled JavaScript code is given below.

example.js

```
function sum(a, b, c) {
    // a and b are mandatory
    // c is optional
    return a + b + c;
}
sum(41, 52); // returns 93
sum(41, 52, 6); // returns 99
```

Conclusion

In this TypeScript Tutorial, we have learnt about basic TypeScript Functions, their syntax, default parameters, rest parameters with example TypeScript programs.