5 Functions

Objectives

After completing this lesson, you should be able to do the following:

- Functions
- Default Function Parameters
- Anonymous Functions
- Arrow Functions in TypeScript
- Function Overloading

TS Functions

- A function is the set of input statements, which performs specific computations and produces output. It is a block of code that is designed for performing a particular task. It is executed when it gets invoked (or called). Functions allow us to reuse and write the code in an organized way.
- Functions in JavaScript are used to perform operations.

Example

```
function hello() //defining a function
{
  console.log ("hello function called");
  }
hello(); //calling of function
```

Default Function Parameters

- In Typescript, the function allows the initialization of parameters with default values if the parameter is undefined or no value is passed to it.

```
function show (num1, num2=200)
{
   console.log("num1 = " +num1);
   console.log("num2 = " +num2);

}
show(100);
```

Rest Parameters

- Rest parameters do not restrict you to pass the number of values in a function, but all the passed values must be of the same type. We can also say that rest parameters act as the placeholders for multiple arguments of the same type.
- For declaring the rest parameter, the name of the parameter should be prefixed with the **spread operator** that has three periods (not more than three or not less than three).

```
function show(a, ...args)
{
   console.log(a + " " + args);
}
show(50,60,70,80,90,100);
```

Output

```
50,60,70,80,90,100
```

Arrow Function In TypeScript

- Arrow functions are introduced in ES6, which provides you a more accurate way to write the functions in JavaScript. They allow us to write smaller function syntax. Arrow functions make your code more readable and structured.
- Arrow functions are **anonymous functions** (the functions without a name and not bound with an identifier). They don't return any value and can declare without the function keyword. Arrow functions cannot be used as the constructors. The context within the arrow functions is lexically or statically defined. They are also called as **Lambda Functions** in different languages.
- Arrow functions do not include any prototype property, and they cannot be used with the new keyword.

There are three parts to an Arrow Function or Lambda Function:

- **Parameters:** Any function may optionally have the parameters.
- **Fat arrow notation/lambda notation:** It is the notation for the **arrow (=>).**
- **Statements:** It represents the instruction set of the function.

```
const functionName = (arg1, arg2, ?..) => {
   //body of the function
}
```

```
// function expression

var myfun1 = function show() {
 console.log("It is a Function Expression");
}

// Anonymous function

var myfun2 = function () {
    console.log("It is an Anonymous Function");
  }

//Arrow function

var myfun3 = () => {
    console.log("It is an Arrow Function");
  };

myfun1();
myfun2();
myfun3();
```

Arrow Function do Contain

1. Optional parentheses for the single parameter
2. Optional braces for single statement and the empty braces if there is not any parameter required.
3. Arrow Function Can Have Parameters
4. Arrow function Can Have default parameters
5. Arrow function Can Have Rest parameters

Summary

In this lesson, you should have learned how to:

- Functions
- Default Function Parameters
- Anonymous Functions
- Arrow Functions in TypeScript
- Function Overloading