TypeScript - Functions

Functions are the primary blocks of any program. In JavaScript, functions are the most important part since the JavaScript is a functional programming language. With functions, you can implement/mimic the concepts of object-oriented programming like classes, objects, polymorphism, and, abstraction.

Functions ensure that the program is maintainable and reusable, and organized into readable blocks. While TypeScript provides the concept of classes and modules, functions still are an integral part of the language.

In TypeScript, functions can be of two types: named and anonymous.

Named Functions

A named function is one where you declare and call a function by its given name.

Example: Named Function

 Copy

function display() {

console.log("Hello TypeScript!");

}

display(); //Output: Hello TypeScript

Functions can also include parameter types and return type.

Example: Function with Parameter and Return Types

 Copy

function Sum(x: number, y: number) : number {

return x + y;

}

Sum(2,3); // returns 5

Anonymous Function

An anonymous function is one which is defined as an expression. This expression is stored in a variable. So, the function itself does not have a name. These functions are invoked using the variable name that the function is stored in.

Example: Anonymous Function

 Copy

let greeting = function() {

console.log("Hello TypeScript!");

};

greeting(); //Output: Hello TypeScript!

An anonymous function can also include parameter types and return type.

Example: Function with Paramter and Return Types

 Copy

let Sum = function(x: number, y: number) : number

{

return x + y;

}

Sum(2,3); // returns 5

Function Parameters

Parameters are values or arguments passed to a function. In TypeScript, the compiler expects a function to receive the exact number and type of arguments as defined in the function signature. If the function expects three parameters, the compiler checks that the user has passed values for all three parameters i.e. it checks for exact matches.

Example: Function Parameters

 Copy

function Greet(greeting: string, name: string ) : string {

return greeting + ' ' + name + '!';

}

Greet('Hello','Steve');//OK, returns "Hello Steve!"

Greet('Hi'); // Compiler Error: Expected 2 arguments, but got 1.

Greet('Hi','Bill','Gates'); //Compiler Error: Expected 2 arguments, but got 3.

This is unlike JavaScript, where it is acceptable to pass less arguments than what the function expects. The parameters that don't receive a value from the user are considered as undefined.

Optional Parameters

TypeScript has an optional parameter functionality. The parameters that may or may not receive a value can be appended with a '?' to mark them as optional.

 Note:

All optional parameters must follow required parameters and should be at the end.

Example: Optional Parameter

 Copy

function Greet(greeting: string, name?: string ) : string {

return greeting + ' ' + name + '!';

}

Greet('Hello','Steve');//OK, returns "Hello Steve!"

Greet('Hi'); // OK, returns "Hi undefined!".

Greet('Hi','Bill','Gates'); //Compiler Error: Expected 2 arguments, but got 3.

In the above example, the second parameter name is marked as optional with a question mark appended at the end. Hence, the function Greet() accepts either 1 or 2 parameters and returns a greeting string. If we do not specify the second parameter then its value will be undefined.

Default Parameters

TypeScript provides the option to add default values to parameters. So, if the user does not provide a value to an argument, TypeScript will initialize the parameter with the default value. Default parameters have the same behaviour as optional parameters. If a value is not passed for the default parameter in a function call, the default parameter must follow the required parameters in the function signature. Hence, default parameters can be omitted while calling a function. However, if a function signature has a default parameter before a required parameter, the function can still be called, provided the default parameter is passed a value of undefined.

Example: Default Parameter

 Copy

function Greet(name: string, greeting: string = "Hello") : string {

return greeting + ' ' + name + '!';

}

Greet('Steve');//OK, returns "Hello Steve!"

Greet('Steve', 'Hi'); // OK, returns "Hi Steve!".

Greet('Bill'); //OK, returns "Hello Bill!"

When the default parameters precede required parameters in a function, they can be called by passing undefined.

Example: Function Call

 Copy

function Greet(greeting: string = "Hello", name: string) : string {

return greeting + ' ' + name + '!';

}

Greet(undefined, 'Steve');//returns "Hello Steve!"

Greet("Hi", 'Steve'); //returns "Hi Steve!".

Greet(undefined, 'Bill'); //returns "Hello Bill!"