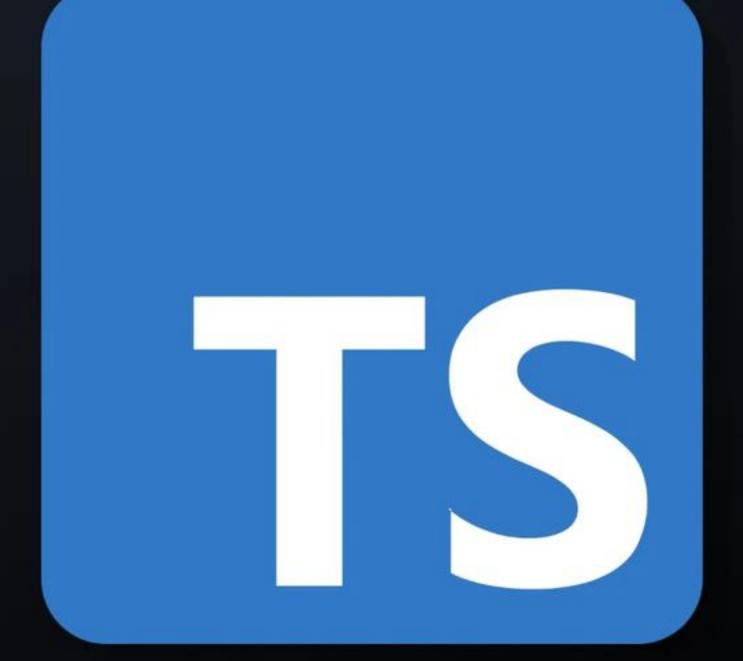
TS SERIES #05

# TypeScript Guide: Functions And Its Parameters

EP: 05







## TypeScript Functions

Functions in TypeScript are blocks of code that can be called to perform specific tasks. They allow the organization of reusable and maintainable code.



## Function Types

Function types are a way to specify the types of the parameters and return value of a function.

A function type has two parts: parameters and return type.

```
1   //Syntax:
2  let myFunction: (param1: type, param2: type) ⇒
3  returnType;
4
5
6   // Example:
7  let add: (a: number, b: number) ⇒ number;
8
9  add = (x, y) ⇒ {
10   return x + y;
11 };
12  console.log(add(5, 3)); // Output: 8
```



## Optional Parameters

Optional parameters allow functions to be called with varying numbers of arguments. Optional parameters are denoted by a question mark? after the parameter name.

```
rs demo.ts
//Syntax:
function functionName(param1: type, param2?: type):
returnType {
     // function body
                         demo.ts
                // Example:
                 function greet(name: string, greeting?: string): string {
                     if (greeting) {
                         return `${greeting}, ${name}!`;
                     } else {
                         return `Hello, ${name}!`;
                 console.log(greet("Alice"));
                 // Output: Hello, Alice!
             10
             11
                 console.log(greet("Alice", "Good morning"));
             12
                 // Output: Good morning, Alice!
```



#### Default Parameters

Default parameters allow parameters to have a default value if no value or undefined is passed. Similar to JavaScript, you can use default parameters in TypeScript with the same syntax:

```
1 //Syntax:
2 function functionName(param1: type, param2: type = defaultValue):
3 returnType {
4    // function body
5 }
```

```
## demo.ts

// Example:
function greet(name: string, greeting: string = "Hello"): string {
    return `${greeting}, ${name}!`;
}

console.log(greet("Alice"));
// Output: Hello, Alice!

console.log(greet("Alice", "Good morning"));
// Output: Good morning, Alice!
```



#### Rest Parameters

A rest parameter allows a function to accept zero or more arguments of the specified type.

To declare a rest parameter, you prefix the parameter name with three dots and use the array type as the type annotation

```
rs demo.ts
   //Syntax:
   function functionName(...params: type[]): returnType {
        // function body
 5
   // Example:
6
   function getTotal(...numbers: number[]): number {
       let total = 0;
8
        numbers.forEach((num) \Rightarrow total += num);
9
       return total;
10
11 }
12
   console.log(getTotal()); // 0
13
14
   console.log(getTotal(10, 20)); // 30
   console.log(getTotal(10, 20, 30)); // 60
15
```



## Function Overloading

Function overloading allows the creation of multiple function signatures for a single function, enabling different type combinations of parameters. It allows you to establish the relationship between the parameter types and result types of a function.

```
rs demo.ts
1 //Syntax:
   function functionName(param1: type): returnType;
   function functionName(param1: type, param2: type): returnType;
   function functionName(param1: any, param2?: any): any {
       // function body
6 }
   // Example:
8
   function add(a: number, b: number): number;
   function add(a: string, b: string): string;
10
   function add(a: any, b: any): any {
11
      return a + b;
13
14
   console.log(add(5, 3)); // Output: 8
15
   console.log(add("Hello, ", "world!")); // Output: Hello, world!
16
```



### Arrow Functions

Arrow functions in TypeScript provide a concise syntax for writing functions and come with features like lexical 'this' binding, which can simplify working with functions, especially in object-oriented programming and callback scenarios.

```
rs demo.ts
   //Syntax:
    (param1: Type1, param2: Type2): ReturnType ⇒ {
        // function body
    // single-line functions
 6
    (param1: Type1, param2: Type2): ReturnType ⇒ expression;
 8
   // Example:
 9
   const add = (a: number, b: number): number \Rightarrow {
10
        return a + b;
11
12 };
13
   console.log(add(5, 3)); // Output: 8
14
```