Functions in Typescript

Chapter 6

Aitrich

Functions

- Functions are the fundamental building block of any applications.
- A function is a set of statements to perform a specific task. Functions organize the program into logical blocks of code.
- A well-designed function enhances code readability, maintainability, and reusability.

Function Aspects

- There are three aspects of a function.

Function declaration	• A function declaration tells the compiler about the function name, function parameters, and return type. • function functionName([arg1, arg2, ...argN]);
Function definition	• It contains the actual statements which are going to executes.
Function call	• We can call a function from anywhere in the program. • FunctionName();

- Types of functions

1. Named Function
2. Anonymous function

Function Types

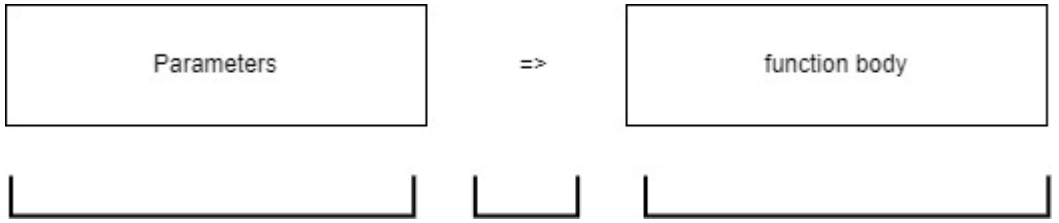
Named Functions

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

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.




```
function findJob(jobTitle1: string, jobTitle2: string): string
{
        return job1+job2;
}
```

```
const login= function(password: string, confirm-password: string): string
        {
                If(password==confirm-password)
                return true;
        };
```

Aitrich

High Order Function in TypeScript

```
function x(fn){
fn();
}

function y(){
console.log("hello from y");
}

x(y);
```

```
//x is higher order function
//y is call back function
```

Functions that take other functions as parameters or which return functions are called higher-order functions(HOF).

Higher-order functions are a very powerful feature and central to the functional programming paradigm.

Function Parameter

Parameters are the values or arguments that passed to a function.

In TypeScript, the compiler accepts the same number and type of arguments as defined in the function signature.

If the compiler does not match the same parameter as in the function signature, then it will give the compilation error.

- Function parameter can be categories into the following

Optional Parameter

Default Parameter

Rest Parameter

Optional Parameter

TypeScript compiler will throw an error if we try to invoke a function without providing the exact number and types of parameters as declared in its function signature.

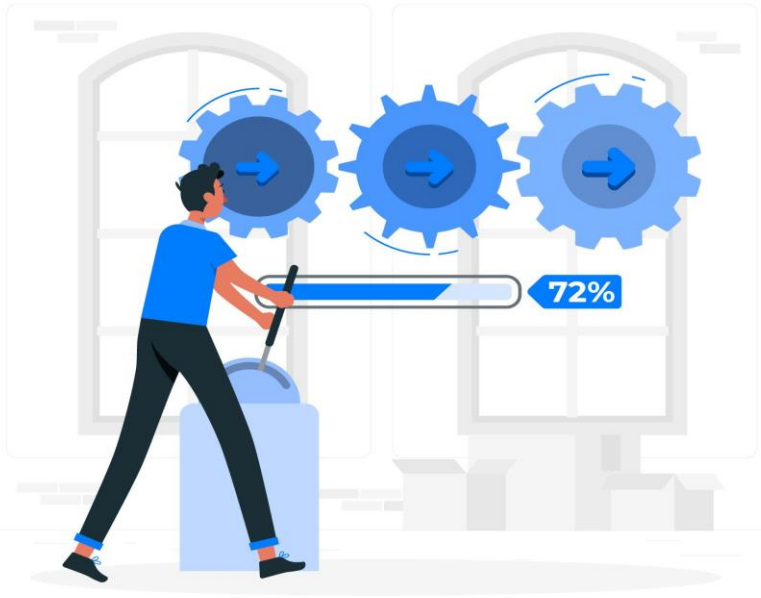
To overcome this problem, we can use optional parameters by using the question mark sign ('?').

It means that the parameters which may or may not receive a value can be appended with a "?" sign to mark them as optional.

```
function function_name(parameter1[:type],
parameter2[:type], parameter3 ? [:type]) { }
```

Aitrich

Optional Parameter




```
function showDetails(id:number,name:string,e_mail_id?:string)
{
    console.log("ID:",    , " Name:",        );
    if(e_mail_id!=undefined)
        console.log("Email-Id:",e_mail_id);
}
showDetails(101,"Virat Kohli");
showDetails(105,"Sachin","sachin@javatpoint.com");
```

Default Parameter

TypeScript provides an option to set default values to the function parameters.

If the user does not pass a value to an argument, TypeScript initializes the default value for the parameter.

```
function function_name(parameter1[:type],
parameter2[:type] = default_value) { }
```

```
postJob(jobTitle: string, jobType: string = "Full-time")
{
      console.log(`[${this.portalName}] Job Posted:
${jobTitle} (${jobType})`);
   }
}

const myJobPortal = new JobPortal("MyJobPortal");

myJobPortal.postJob("Software Engineer");
//Returns  “Software Engineer”   “Full-time”
myJobPortal.postJob("Data Analyst", "Part-time");
//Returns  “Software Engineer”   “Full-time”
```

Rest Parameter

The rest parameter is used to pass zero or more values to a function.

We can declare it by prefixing the three "dot" characters ('...') before the parameter.

It allows the functions to have a different number of arguments without using the arguments object.

The TypeScript compiler will create an array of arguments with the rest parameter so that all array methods can work with the rest parameter.

function
function_name(parameter1[:type],
parameter2[:type],
...parameter[:type]) { }

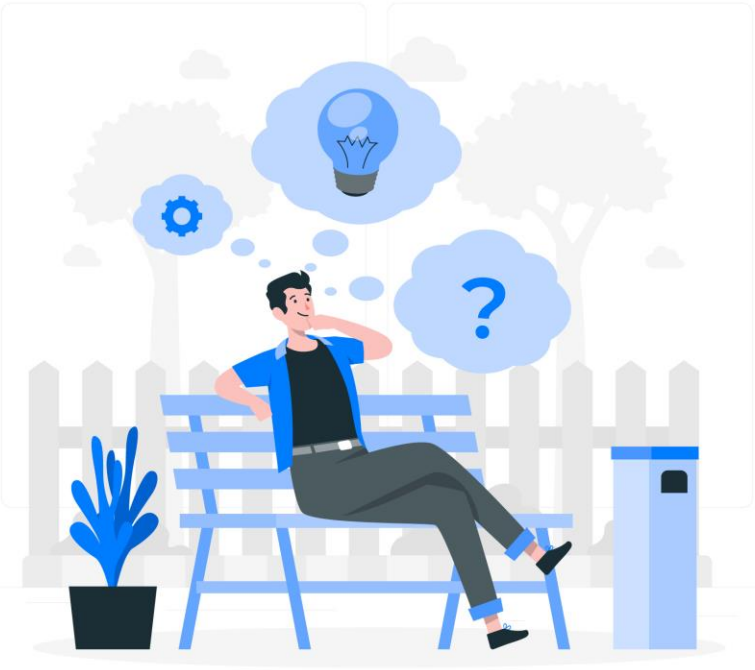
Rest Parameter

```
function jobList ( jobs : string, ...list : string[] )
{

    return jobs + " " + list . join(", ") + "!";
}

jobList ( "Java Devloper", "Angular Developer",
".NetDeveloper"

jobList("Meanstack Developer");
```



Aitrich


THANK
YOU