|  |
| --- |
|  |
| Angular 8 |
|  |
| TypeScript |

**TechBrain Express**

November 11, 2019

Authored by: Ashu

Angular -Typescript

TypeScript

**• List key features of ECMAScript6 (ES6)**

**• Explain the need and benefits of TypeScript**

**• Demonstrate compilation and execution of a typescript file**

**• Demonstrate defining objects using data types, interfaces and classes**

**• Interpret how to implement functions and for .. of in TypeScript**

Official website: [https://www.typescriptlang.org](https://www.typescriptlang.org/index.html)

Source code:<https://github.com/Microsoft/TypeScriptIssues>

# Explain the need and benefits of TypeScript

## Issues with Javascript?

Does not supports static type chekcing?

Time taking coding

Does not support generic

## Why Should We Use TypeScript?

JavaScript has seen a vast development in the last few years. It is the most versatile cross-platform language that is used to develop modern web applications. It can be used to develop both the client side of an application, with frameworks like Angular or React.js, as well as the server side, with frameworks such as Node.js. It is now being considered the most opted for language to build end-to-end applications. But, JavaScript was never meant for such large-scale application development. JavaScript is a dynamic programming language with no type system, unlike well-structured and refined languages such as C# or Java. A no type system means that a variable in JavaScript can have any type of value such as string, number, boolean etc. The type system increases the code quality, readability and makes it an easy to maintain and refactor code base. More importantly, errors can be caught at compile time rather than at run time.

Without the type system, it is difficult to scale JavaScript to build complex applications with large teams working on the same code.

Hence, the reason to use TypeScript is that it allows JavaScript to be used at scale.

TypeScript compiles into simple JavaScript. The TypeScript compiler is also implemented in TypeScript and can be used with any browser or JavaScript engines like Node.js. TypeScript needs an ECMAScript 3 or higher compatible environment to compile. This is a condition met by all browsers and JavaScript engines today.

Some of the most popular JavaScript frameworks like Angular.js and WinJS are written in TypeScript.

## Advantage of TypeScript over JavaScript

* TypeScript always highlights errors at compilation time during the time of development, whereas JavaScript points out errors at the runtime.
* TypeScript supports strongly typed or static typing, whereas this is not in JavaScript.
* TypeScript runs on any browser or JavaScript engine.
* Great tooling supports with IntelliSense, which provides active hints as the code is added.
* It has a namespace concept by defining a module.

## Disadvantage of TypeScript over JavaScript

* TypeScript takes a long time to compile the code.
* TypeScript does not support abstract classes.
* If we run the TypeScript application in the browser, a compilation step is required to transform TypeScript into JavaScript.

## Features

* **Cross-Platform:** TypeScript runs on any platform that JavaScript runs on. The TypeScript compiler can be installed on any Operating System such as Windows, MacOS and Linux.
* **Object Oriented Language:**TypeScript provides powerful features such as Classes, Interfaces, and Modules. You can write pure object-oriented code for client-side as well as server-side development.
* **Static type-checking:** TypeScript uses static typing. This is done using type annotations. It helps type checking at compile time. Thus, you can find errors while typing the code without running your script each time. Additionally, using the type inference mechanism, if a variable is declared without a type, it will be inferred based on its value.
* **Optional Static Typing:** TypeScript also allows optional static typing if you would rather use JavaScript's dynamic typing.
* **DOM Manipulation:** Just like JavaScript, TypeScript can be used to manipulate the DOM for adding or removing elements.
* **ES 6 Features:** TypeScript includes most features of planned [ECMAScript](https://www.tutorialsteacher.com/articles/what-is-ecmascript) 2015 (ES 6, 7) such as class, interface, Arrow functions etc.

## What is Type Script?

TypeScript is an open-source pure object-oriented programing language by Microsoft. It is a typed superset of JavaScript that compiles to plain JavaScript. TypeScript was developed under Anders Hejlsberg, who also led the creation of the C# language. TypeScript was first released in October 2012.

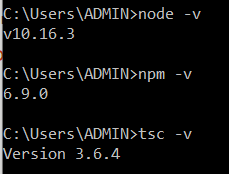
# Installation of TypeScript

There are two ways to install TypeScript:

1. Install TypeScript using Node.js Package Manager (npm).
2. Install the TypeScript plug-in in your IDE (Integrated Development Environment).

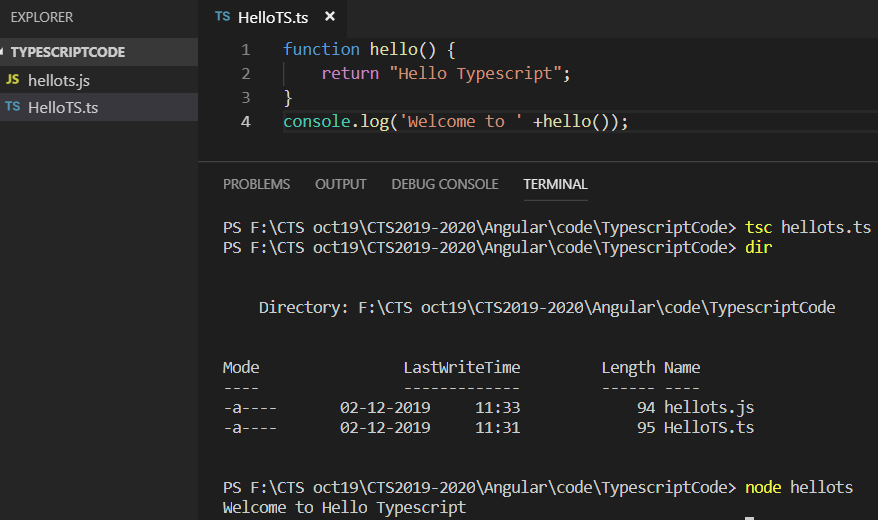
## Install TypeScript using Node.js Package Manager (npm)

1. Install Nodejs <https://nodejs.org/en/download/>
2. Check version > node –v
3. To install TypeScript
   1. $ npm install typescript --save-dev         //As dev dependency
   2. $ npm install typescript -g                      //Install as a global module
   3. $ npm install typescript@latest -g
4. Check tsc version > tsc-v



# Demonstrate compilation and execution of a typescript file

## First Program



## Variable

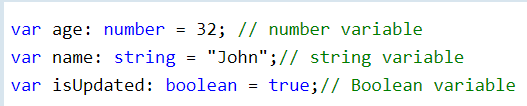
4 ways to declare a variable

* var [identifier] : [type-annotation] = value;
* var [identifier] : [type-annotation];
* var [identifier] = value;
* var [identifier];
* var name:string = "Daisy";

var empid:number = 1001;

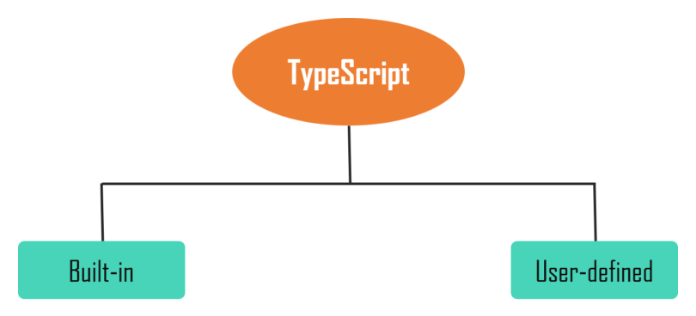
console.log("name"+name)

console.log("employee id "+empid)

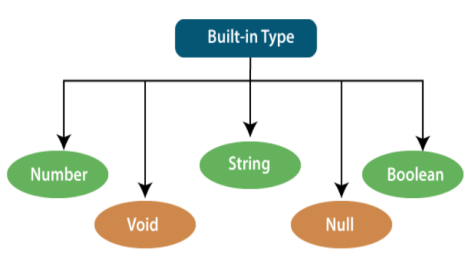
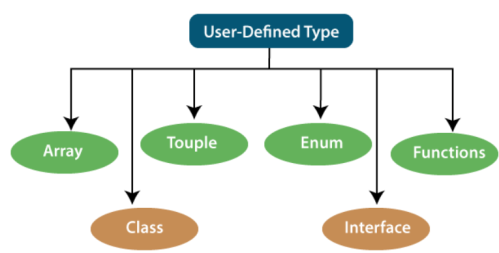


# Demonstrate defining objects using data types

## Data Type



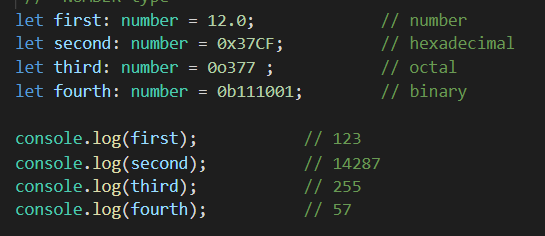
* **Built-in**: This includes number, string, boolean, void, null and undefined.
* **User-defined**: It includes Enumerations (enums), classes, interfaces, arrays, and tuple.

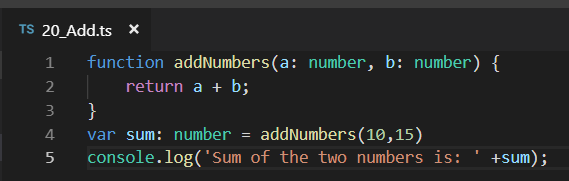
 

SNBNV ACTIEF

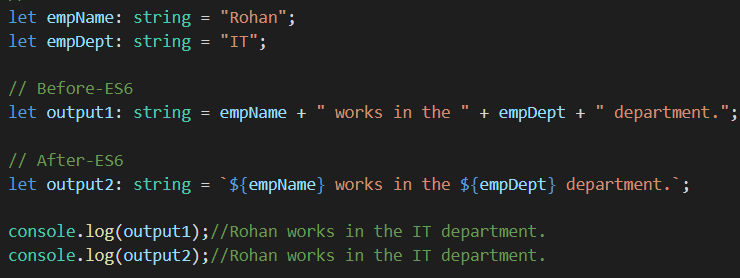
## Built-in

### Number





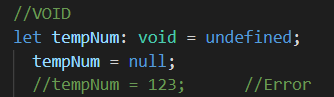
### String



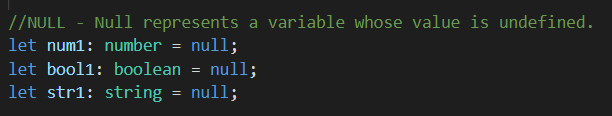
### Boolean

### 

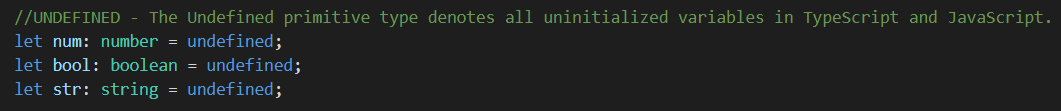
### Void



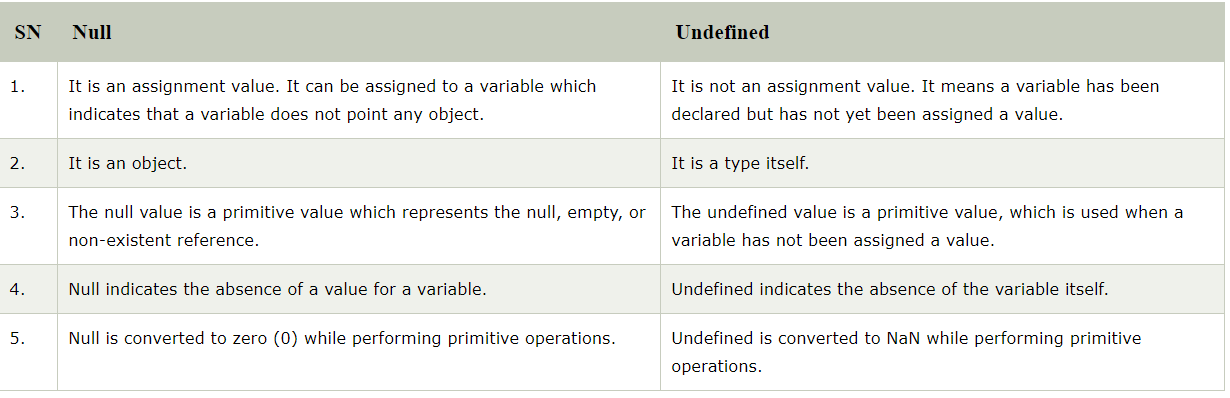
### Null



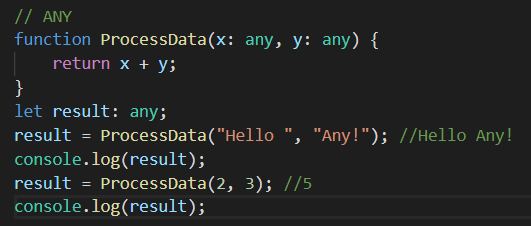
### Undefined



### Null vs Undefined



### Any



## Var Vs let

|  |  |
| --- | --- |
| Var (GLOBAL) | Let |
| Introduced with JavaScript. | Added in ES6 (ES 2015) version of JavaScript. |
| global scope. | limited to block scope. |
| declared globally and can be accessed globally. | declared globally but cannot be accessed globally. |
| Variable declared with var keyword can be re-declared and updated in the same scope. **Example:**  function varGreeter(){  var a = 10;  var a = 20; //a is replaced  console.log(a);  }  varGreeter(); | Variable declared with let keyword can be updated but not re-declared. **Example:**  function varGreeter(){  let a = 10;  let a = 20; //SyntaxError:  //Identifier 'a' has already been declared  console.log(a);  }  varGreeter(); |
| It is hoisted. **Example:**  {  console.log(c); // undefined.  //Due to hoisting  var c = 2;  } | It is not hoisted. **Example:**  {  console.log(b); // ReferenceError:  //b is not defined  let b = 3;  } |
|  |  |

|  |  |  |
| --- | --- | --- |
| Category | Let | Var |
| Global window object | Not added to global window | Added to global window |
| Block | limited use of those variables. | Not limited |
| Redeclaration | let variables cannot be re-declared | can be re-declared in the same scope. |

## Const

1. const declares a block-scoped variable with a constant value. It is basically variable declaration with ‘var‘ keyword where variable value is constant and cannot be changed.
2. const follows the same scoping principles as the let keyword.
3. if you know that the variable you are declaring cannot and should not be allowed to reassign, then declare it with const, else use the let keyword.

## HOISTING

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.

<https://scotch.io/tutorials/understanding-hoisting-in-javascript>

# Demonstrate defining objects using Interface

# Use of Interface

# We can use the interface for the following things:

* Validating the specific structure of properties
* Objects passed as parameters
* Objects returned from functions.

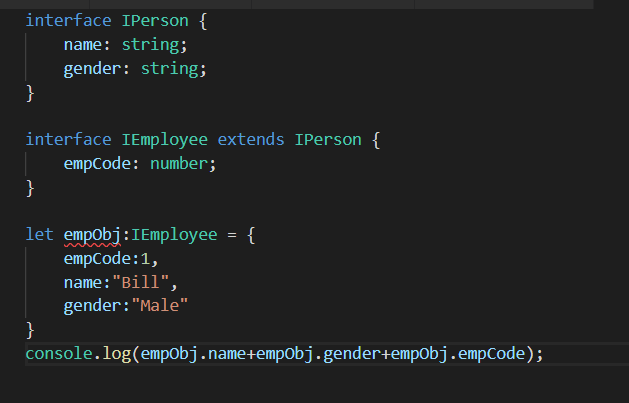
An Interface is a structure which acts as a contract in our application. It defines the syntax for classes to follow, means a class which implements an interface is bound to implement all its members.

We cannot instantiate the interface, but it can be referenced by the class object that implements it. The TypeScript compiler uses interface for type-checking (also known as "duck typing" or "structural subtyping") whether the object has a specific structure or not.

The interface contains only the declaration of the methods and fields, but not the implementation.

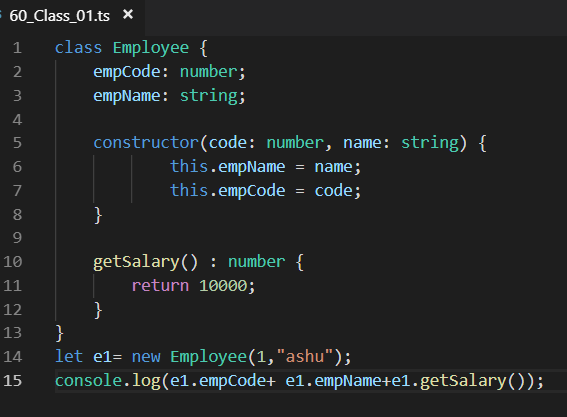


## Inheritance Interface

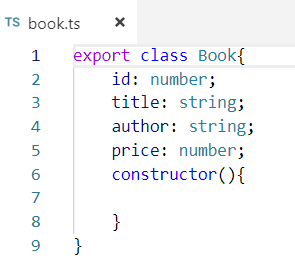


Note: Objects of IEmployee must include all the properties and methods of the IPerson interface otherwise, the compiler will show an error.

## Class

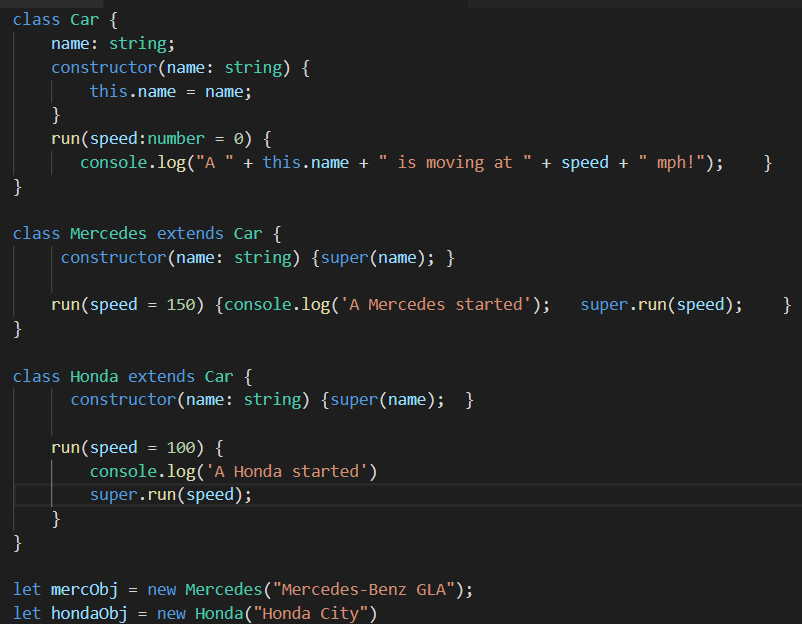


ASSIGNMENT: Class Book.ts class

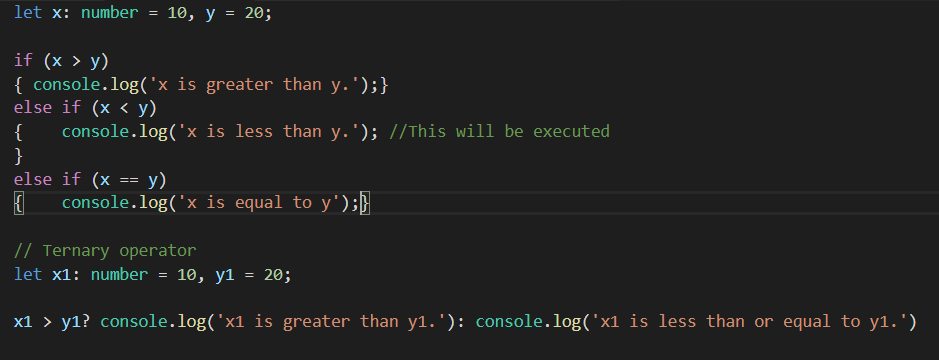


Create Book.json

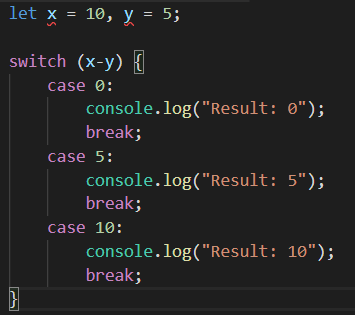
## Class +Inheritance+ Method Override



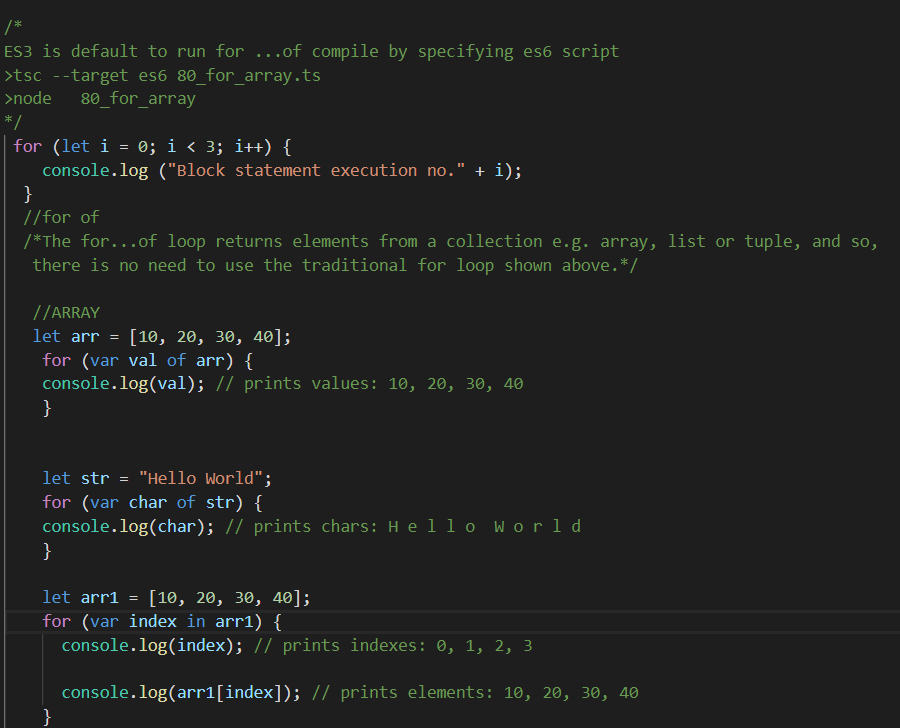
**Interpret how to implement If**



**Interpret how to implement Switch**

****

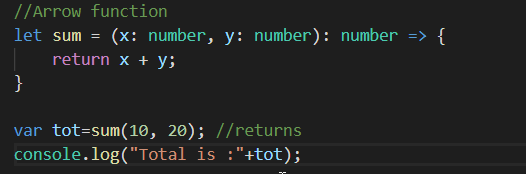
**For + Array**

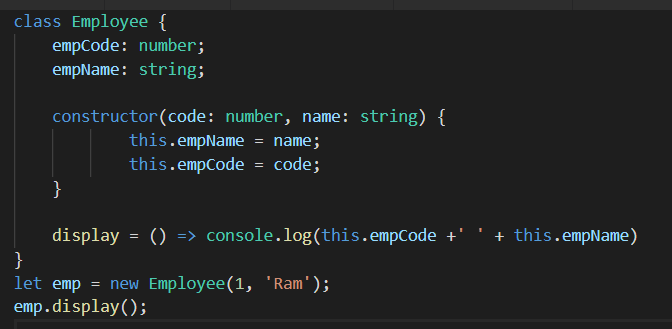
****

## Functions- Arrow

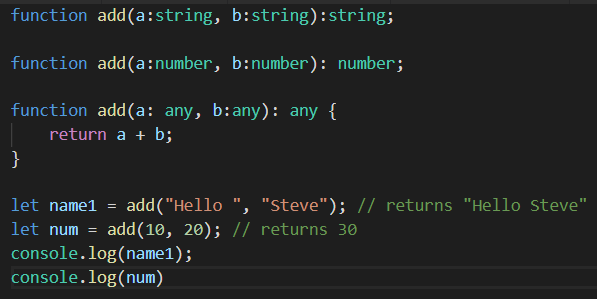
Fat arrow notations are used for anonymous functions i.e for function expressions. They are also called lambda functions in other languages.

Using fat arrow (=>) we drop the need to use the 'function' keyword. Parameters are passed in the angular brackets <>, and the function expression is enclosed within the curly brackets {}.





## Function – Function Overloading



## Function – Rest

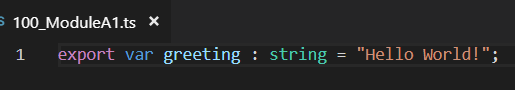
When the number of parameters that a function will receive is not known or can vary, we can use rest parameters. In JavaScript, this is achieved with the "arguments" variable. However, with TypeScript, we can use the rest parameter denoted by ellipsis ....

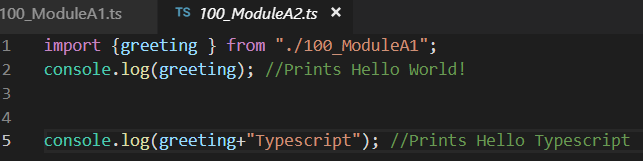


NOTE:  rest parameters must come last in the function defination, otherwise the TypeScript compiler will show an error. The following is not valid.

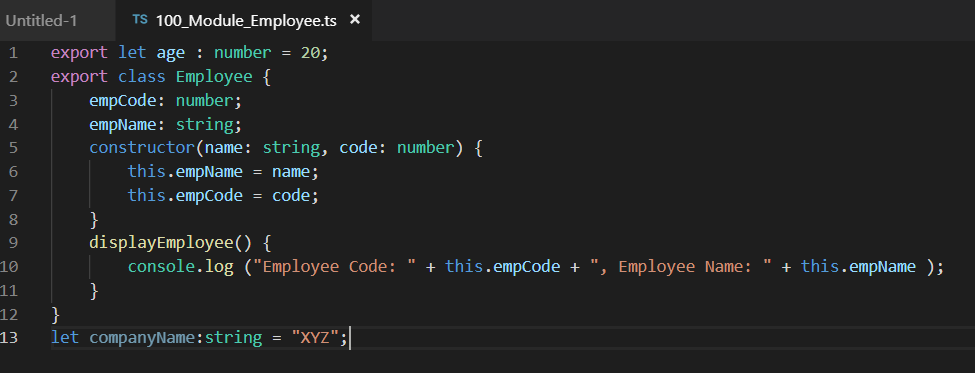
# Module

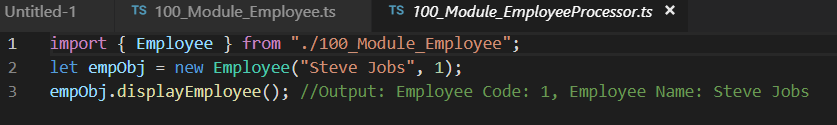
The TypeScript code we write is in the global scope by default. If we have multiple files in a project, the variables, functions, etc. written in one file are accessible in all the other files.

Lab1: 

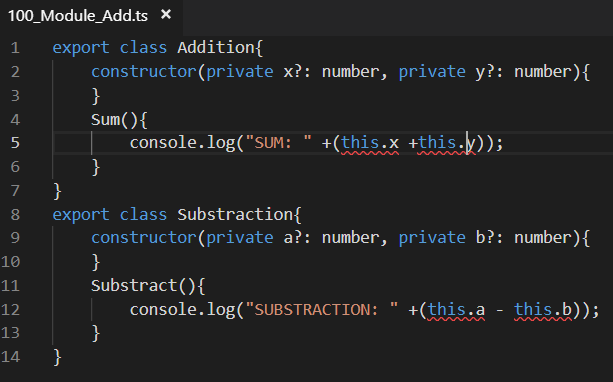


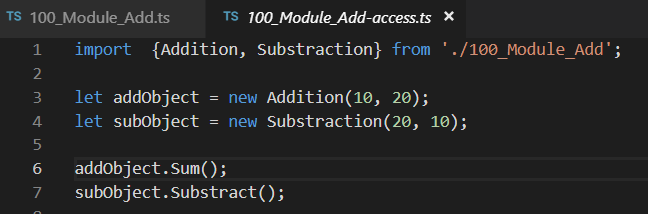
Lab2:





### Lab:

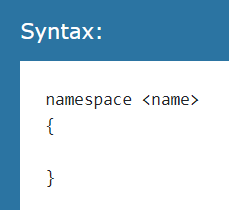




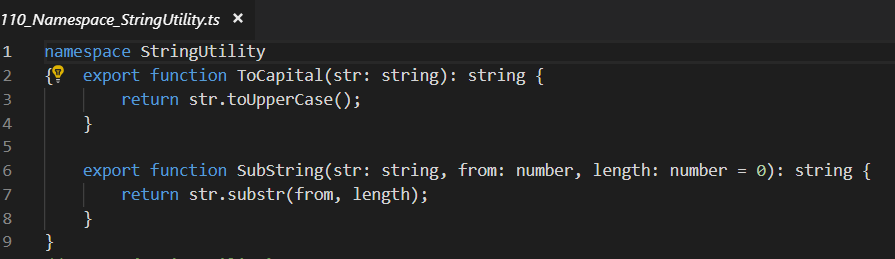


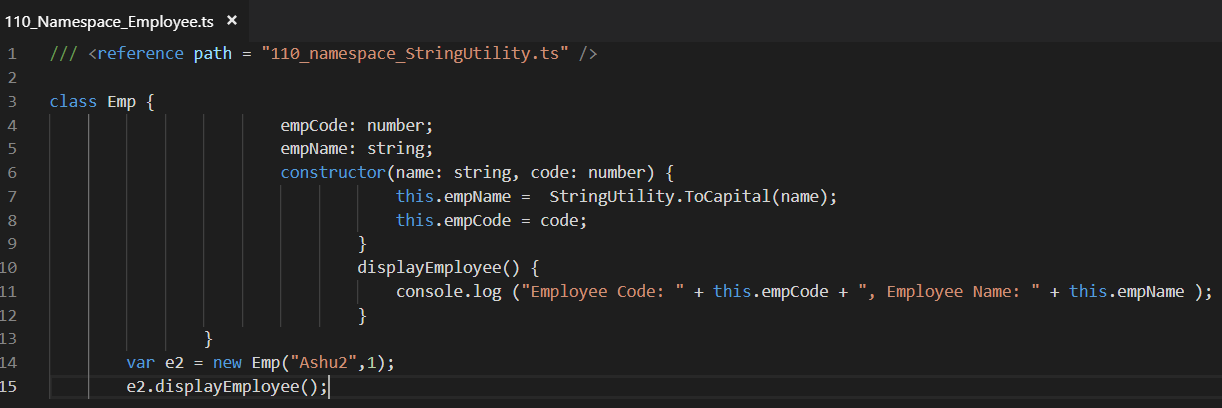
# Namespace

The namespace is used for logical grouping of functionalities. A namespace can include interfaces, classes, functions and variables to support a single or a group of related functionalities.



Lab: Declare toString and tocapital() in namespace and access in another file

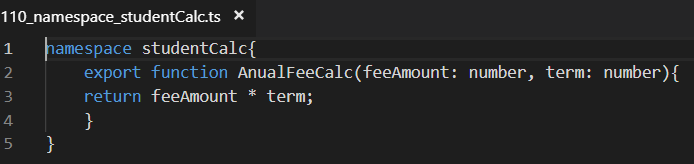


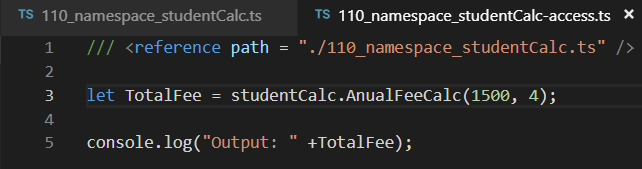


Compile and run



Lab:

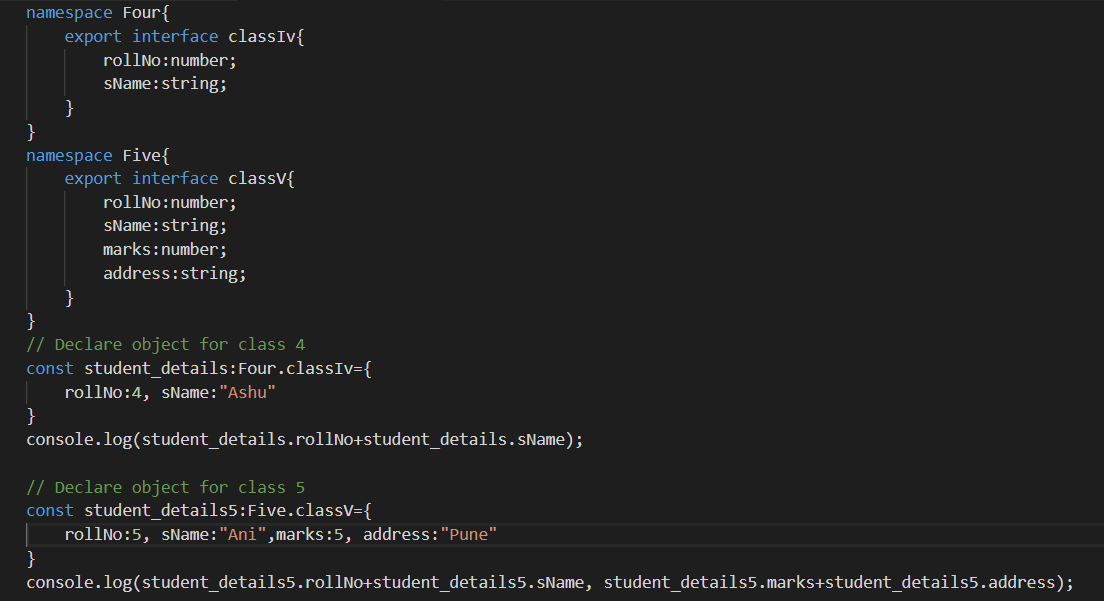
****

****

Compile and Run

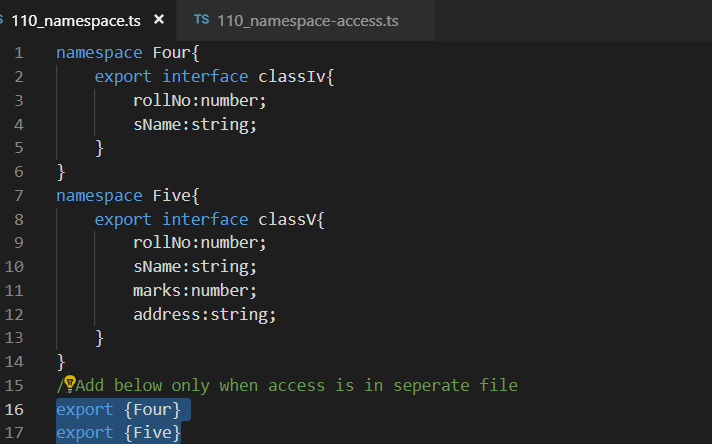
****

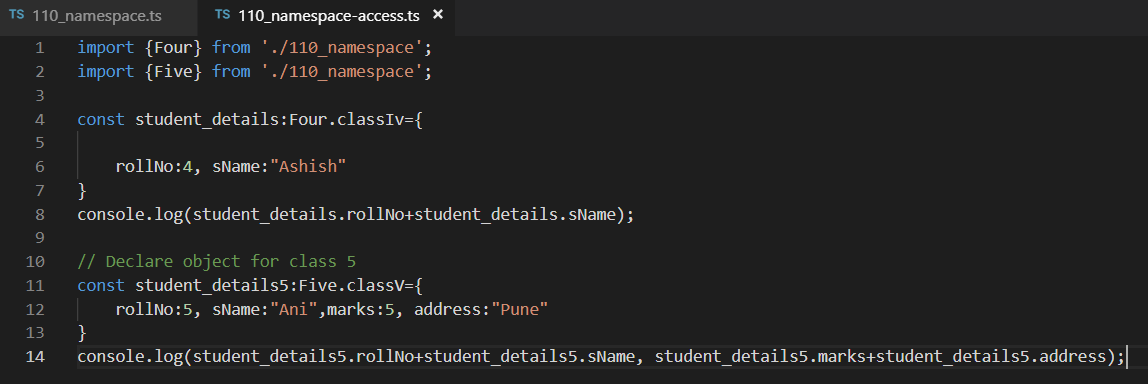
Within same file



# Module Vs Namespace

## With different file:Need to export interfaces





Compile and run 110\_namespace-access.ts

Module Vs Namespace