**TYPE SCRIPT**

**Introduction**:

Typescript is a superset of JavaScript best transpiler that compiles the Object Oriented code to plain JavaScript. This is a pure Object Oriented Language with Classes and Interfaces. This is an Extension of JavaScript where we can write OOPS code and that compiles back to pure / Core JavaScript which runs on any browser , any OS and any Device.

TypeScript is developed and Maintained by Microsoft and is an Open Source Project.

The Popular Front End Framework Angular 5 is also built on TypeScript.

The Current version is 2.8.1 released 27th, March, 2018.

Typescript is one of the best transpiler as of today which supports almost all the modern features of ECMA Script and in future also it will supports all the new upcoming features of ECMA Script.

We will write code in TypeScript and compiles it back to core JavaScript of any version.

As JavaScript is a weakly typed language where in TypeScript is a Strongly Typed Language, in order to avoid unnecessary errors caused by JavaScript at Runtime.

# **TypeScript Summary:**

|  |  |
| --- | --- |
| Type | Details |
| Developed by | **Microsoft** |
| License | **Open Source** |
| Current Version | **2.8.1 released 27th,March,2018** |
| Official Website | [**https://www.typescriptlang.org/**](https://www.typescriptlang.org/) |
| Practice Online | [**https://www.typescriptlang.org/play/index.html**](https://www.typescriptlang.org/play/index.html) |
| Known as | **Best Transpiler for JavaScript** |
| Compatible with | **Any Browser , Any OS , Any Device** |

# **ECMA Script**

JavaScript is also called as ECMA Script. ECMA Stands for European Computers Manufacturers Association.

ECMA is the standard body which releases the different versions of ECMA Script / JavaScript. The current version of ECMA Script is **ECMA Script-2017** (8th Edition).

This ECMA Script language features can be used by all the browsers, to implement the support for running various versions of JavaScript/ECMA Script.

As ECMA releases the new features, which are not automatically available by all the browsers and most browsers supported version of ECMA Script is **ES-5** version.

This ECMA Script 5 is the version, which is used by almost all the browsers supports as today.

Here we needs to understand if ECMA releases a new feature / standard, however it does not mean that every feature of the standard body ECMA is the standard and immediately available in all the browsers.

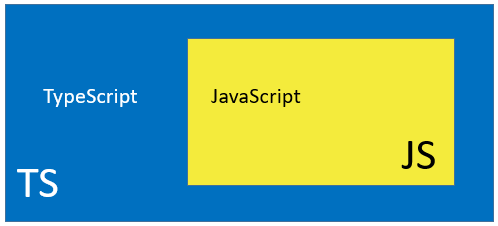
In reality some implementations / features are already available in all the browsers and which are not standardized by ECMA and also the features and some implementations are standardized by ECMA which are not available in all browser.

Now we understood that there is a gap between the ECMA Script versions used by all the modern browsers and the version which are released by ECMA.

As a JavaScript developer may get frustrate because there are lot of new features coming in but there is no actual browser support for those features in order to leverage those features.

This is the place where Transpiler comes into picture to fill the gap, where the transpiler compiles and transforms from one standard to another standard.

So TypeScript is one of the best transpiler as of today and we can leverage all the features of ECMA script and even future versions of ECMA Script and transpiles to ES-5 and which is fully supported by all the browsers as of today.



There are many Transpilers available in the market along with TypeScript is Babel Script. TypeScript is more popular and even the Angular Framework uses TypeScript.

TypeScript supports all the features of ES6 and also the future versions of ECMAScript and compiles them to any chosen version of JavaScript like ES5.

# **Differences between TYPESCRIPT and JAVASCRIPT**

|  |  |
| --- | --- |
| JAVASCRIPT | TypeScript |
| JAVASCRIPT is a Dynamic Typed Language | TS is a Static Typed Language |
| This is most forgiving language and weakly typed language | This is particular about typing and it is a strongly typed language like Java |
| As this is not a Strict typing, it leads to many unexpected runtime errors during development time | As this is a Strong typing, it avoids almost all unexpected runtime errors during development time, especially unexpected errors seen in JAVASCRIPT |
| It applies typing in Dynamic runtime and which leads to many runtime errors | It applies type in the development time itself and avoids runtime errors |
| JAVASCRIPT is great for Web Browser Object Model | It promotes stability and maintainability |

NOTE:

JavaScript is a dynamic typing language means we can provide the typing information for any variables at runtime. But for TypeScript we can provide the typing information only at the time of declaration of variables. This is called static typing.

As JavaScript is dynamic typing language, it is best suitable for BOM (browser object model) and with TypeScript adds static typing to the JavaScript to add more stability and more maintainability of the program and avoids unexpected runtime errors for JavaScript.

Instead of writing direct JavaScript, if we use TypeScript to generate JavaScript which compiles and gives the most stable and most maintainable JavaScript and most powerful JavaScript.

So this is the reason why Angular JS is completely redesigned to use TypeScript and renamed to Angular.

# **TYPESCRIPT Editors**

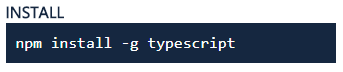
We can use any text editor to work with TypeScript as follows,

1. Any text editor + TS compiler (command prompt)
2. Atom Text Editor by Github - open source
3. Visual Studio Code by Microsoft - open source
4. Brackets Editor by Adobe Systems - open source
5. Sublime Text by Sublime – Semi licensed
6. WebStorm by JetBrains – Fully Licensed

# **Installing TypeScript**

We can install the Type using the Node JS’ Node Package Manager (npm).

1. Install Node JS
2. Check the versions of **node** and **npm**
3. Install “typescript” using npm



1. Check the version of TypeScript

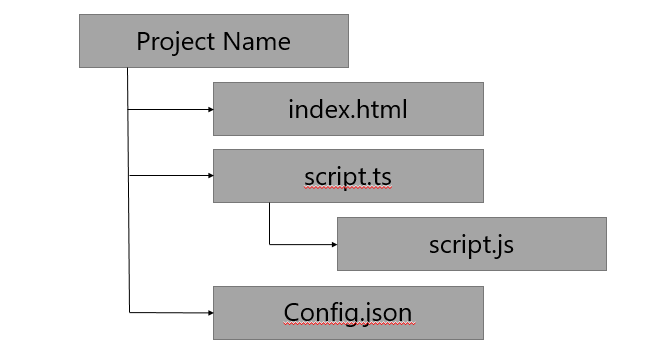


1. Compile the TypeScript using **tsc**



# **Project Structure of TypeScript**

We must follow the below project structure for the entire course.



**ECMA Script 6 Features:**

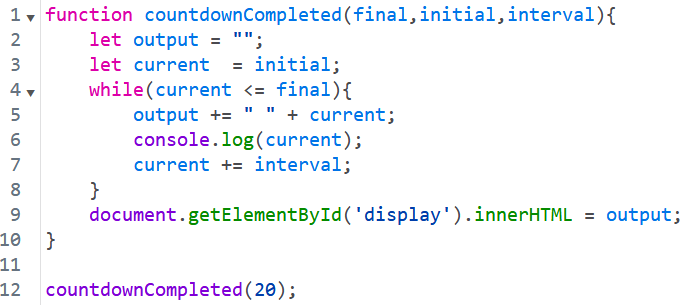
TypeScript leverages to use all the ECMA script features and compile them to ‘ESS’ plain JavaScript to work on any modern browser.

Here we use all the ‘ES6, features in TypeScript and compile them to ‘ES5’ JavaScript code and execute them in any browser.

Now let’s understand the ‘ES6’ features and their usage.

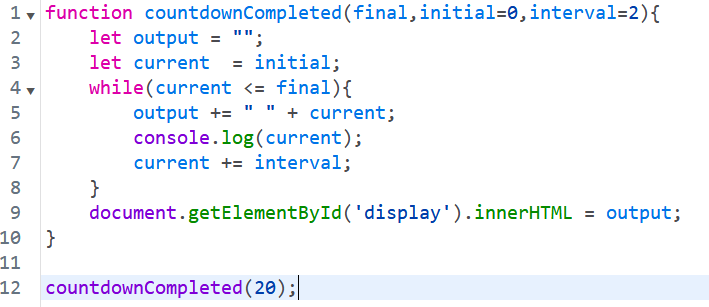
## **Default Parameters:**

This feature is to provide some default values to optional parameters, if any of the parameter is not supplied, then the program will be using these default parameters.



In the above example the function takes in the parameters final, initial, interval.

Let’s make “initial” and “interval” are optional parameters, we can use in the following way:



In the above function, we have provide a default parameters for **initial** and **interval** values,

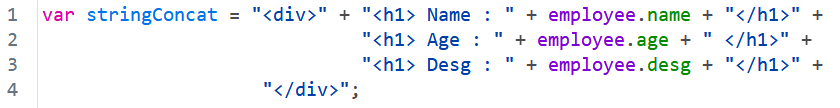
If that values are not provided, then these default values will take into consideration.

## **Template Strings**

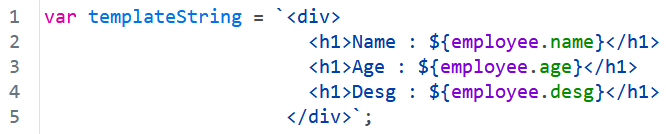
Normally we use String Concatenation operator to combine two or more strings. This approach is more difficult if mix the HTML tags with normal Strings. We resolve this by using Template Strings.

In this template string, we use back tick (`) symbol to combine any kind of strings literals or HTML code or any JavaScript code instead of using string concatenation operator.

By using String Concatenation operator we can mix the HTML code with normal Strings as follows,



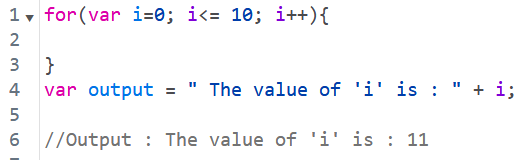
We can replace this by using Template String with back tick ( ` ) operator as follows,



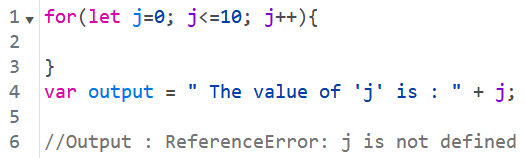
## **Let and Const**

**Let** is one of the **ECMA Script6** feature which is used to declare variables in TypeScript same as the ‘**var’** keyword in JavaScript.

The problem with the **var** keyword is, once we create a variable inside if block or for loop, we can still access that variable outside if block or for loop. We can avoid access of the variables which are declared inside if block or for loop using **let** keyword.



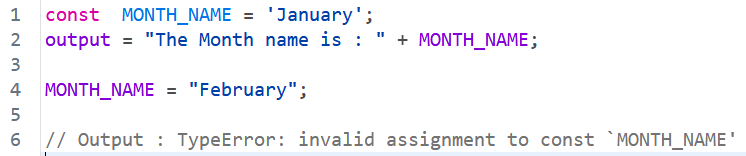
We can avoid the access any variable declared inside any block is by using **let** keyword as follows,



**Const** keyword is used to declare some final variables, once a variable is declared with **const**, we can’t able to change the value of it and also it avoid access of that variable outside the declared block.



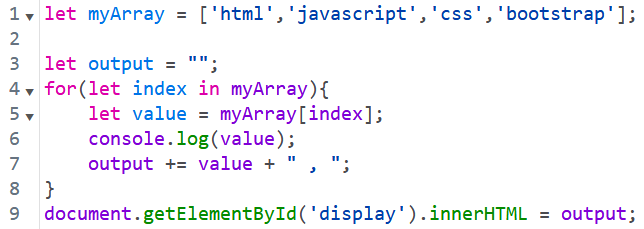
Once a variable is declare using **const**, we can’t change the value of it. If we are trying to change the value of it, we may get the Compile Time error as follows,



## **For …of loops**

Normally we use **for…in loop** to loop through an array and we will get the index and with the index we can get the actual value of an array as follows,

This is available in ES5 version only.



From ES6 version onwards, ECMA has provided a new concept called **for…of loop**.

By using this for…of loop, we can directly get the actual value of an array instead of getting an index of an array as follows,

