**Typescript**

**Getting Started**

1. [Introduction To TypeScript](https://www.tektutorialshub.com/typescript/introduction-to-typescript-what-is-typescript/)
2. [TypeScript Installation and Environment Setup](https://www.tektutorialshub.com/typescript/typescript-installation-environment-setup/)
3. [TypeScript Hello World Example](https://www.tektutorialshub.com/typescript/typescript-getting-started-with-hello-world-example/)
4. [Enable Compile on Save in Visual Studio Code](https://www.tektutorialshub.com/typescript/typescript-compile-on-save-in-visual-studio-code/)
5. [Syntax and Basic Rules](https://www.tektutorialshub.com/typescript/typescript-syntax/)

**Typescript Basics**

1. Typescript Data Types
2. Type Annotations
3. Variable Declaration
4. Identifiers & keywords
5. Variable Scope
6. Let, Var & Const
7. Constants
8. Type Inference

**Strings**

1. [String Data Type](https://www.tektutorialshub.com/typescript/typescript-string/)
2. [Template strings/Literal strings](https://www.tektutorialshub.com/typescript/template-literals-template-strings-in-typescript/)
3. [Tagged Templates](https://www.tektutorialshub.com/typescript/typescript-tagged-templates/)

**Numbers**

1. [Number Data Type](https://www.tektutorialshub.com/typescript/typescript-number/)
2. [NaN in Typescript](https://www.tektutorialshub.com/typescript/nan-in-typescript/)
3. [Min, Max & Safe Values](https://www.tektutorialshub.com/typescript/typescript-number-min-max-safe-values/)
4. [EPSILON & Floating Point Precision](https://www.tektutorialshub.com/typescript/typescript-epsilon-floating-point-precision/)
5. [Infinity](https://www.tektutorialshub.com/typescript/typescript-infinity/)

**BigInt**

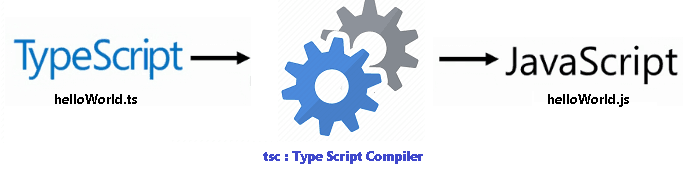
1. [BigInt data type](https://www.tektutorialshub.com/typescript/typescript-bigint/)
2. [BigInt Vs Number](https://www.tektutorialshub.com/typescript/typescript-bigint-vs-number/)

**Boolean**

1. [Boolean Data Type](https://www.tektutorialshub.com/typescript/typescript-boolean/)
2. **Introduction to typescript**

Typescript is superset language written over javascript .It added only syntactic suger over javascript to add static type instead dynamic type of javascript .

we can write the javascript and Still typescript still compile it just as it is javascriprt only .



**Note : TypeScript** is a primary language used in **Angular** application development.

## Benefits of TypeScript

Since in the end Typescript produces Javascript, you may wonder why not use javascript directly. The Benefit of TypeScript comes with its type system, which provides several benefits over javascript.

* 1. Optional Type System

TypeScript provides the static type system which provides great help in catching programming errors at compile time.

Javascript is a dynamically typed system. The variables can hold any values. The type of variable is determined on the fly. The javascript implicitly converts types for example string to a number. This is ok for a small app, but large apps this can be a lot of headaches. It is difficult to test to see if the proper types are passed and errors always happen at runtime.

* 1. Intellisense & syntax checking

The static Type system helps in provide better tooling support in IDE. The intellisense, syntax checking & code completion are few of the major benefits you get with the tooling support. This speeds up the development time and also ensures that the programmers make fewer mistakes with typos. All the major editors like VSCode, atom, sublime text includes the tooling support for Typescript.

* 1. Maintainable code

Typescript brings Types, Classes, interfaces & modules. it makes the code more maintainable and scalable. It much easier to organize the Typescript code, than a Javascript code.

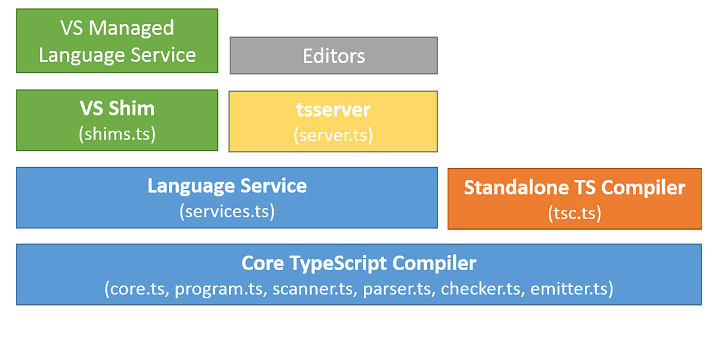
* 1. Language Enhancement

Typescript comes with several language features. It supports Encapsulation through classes and modules. Supports constructors, properties & functions. It has support for Interfaces. You can make use of Arrow functions or lambdas or anonymous functions.

The Typescript is still Javascript. It retains most of the javascript syntax. This means that all valid JavaScript code is valid TypeScript code. Hence the learning curve is lean. If you are not aware of OOP concepts like classes, interfaces, etc, you may need to learn them to get the best out of Typescript

## Typescript Components

The architecture of TypeScript is neatly organized in different layers as shown in the image below. The three major layers are



a.Core TypeScript Compiler.

b.Typescript Standalone Compiler

c. Typescript Language Services

Core TypeScript Compiler

The TypeScript compiler manages the task of type-checking our code and converting it into valid JavaScript code. The compiler is made up of several different layers like core, program, scanner, parser, checker & emitter, etc

Standalone compiler (tsc)

The batch compilation CLI. Mainly handle reading and writing files for different supported engines (e.g. Node.js)

Language Service

The Language Service supports the editors and other tools to provide better assistance in implementing features such as IntelliSense, code completion, formatting and outlining, colorization, code re-factoring like rename, Debugging interface helpers like validating breakpoints, etc.

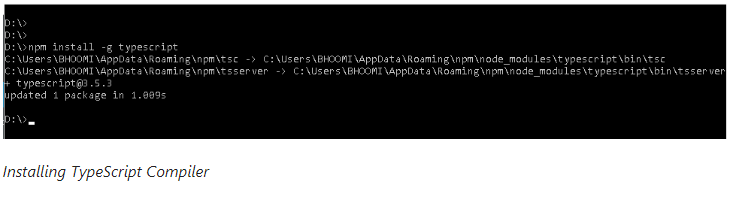
1. **Installation guide**

The Typescript is installed using the npm package manager or NPM. It is a default package manager for Node.js and is installed when the Node.js is installed.

You can download and install Node.js from downloading it from the official website https://nodejs.org/.

Once you install the Node, you can install the typescript compiler using the following command from the command prompt.

Npm install –g typescript



Installing Visual Studio Code

Next, we will need a Code editor. You have many choices here. For this tutorial, we will use Visual Studio Code.

You can download it from the official <https://code.visualstudio.com/download>

1. **Hello world first program using typescriprt**

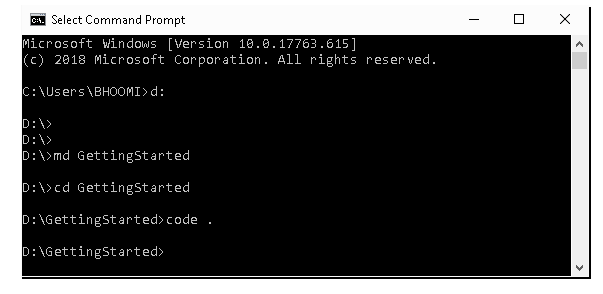
Typescript Hello World Example

Open the command prompt and create a folder GettingStarted and then cd into it.

md GettingStarted

cd GettingStarted

Now, Open Visual Studio Code or any text editor. Go to the GettingStarted folder**.**



You can open the Visual Studio Code just by typing the following command in the GettingStarted folder

code .

Copy the following code and save the file as helloWorld1.ts



The message is a variable of type string. A variable is a storage for the data. “message” is the name of the variable. In the code above, it stores the value “Hello World”, which is a string

We use let(or var) to create variables in typescript. You will learn more about the variable declaration in our next tutorial

console.log() writes the whatever the message passed onto it to the console.

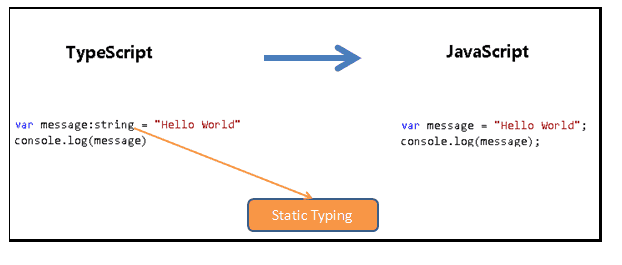
Compile/Execute TypeScript Programs

Now, go back to the command prompt or click on View -> Terminal in Visual Studio Code menu, which will open the PowerShell terminal window

Typescript codes are compiled using the Typescript compiler ie. tsc. Run the following command to compile

tsc helloWorld1

The tsc will look for the helloWorld1.ts file and compiles and generates the helloWorld1.js file. Open the file and inspect its content.



Running the Hello World using nodeJs

We will use the node.js to execute it. Run the following in the command window

node helloWorld1



Running TypeScript in the web app

Create the new typescript file hellowWorld2.ts

|  |  |
| --- | --- |
| Letmessage:string = "Hello World"  alert(message) |  |

The only change we have from the above code is we removed console.log() to alert(),

The alert() method displays the message passed onto it in a window, with an **ok** button. This method works only in the browser, hence we did not use in the previous code.

Compile the helloWorld2 as shown below



The above will generate the helloWorld2.js file. Now you can refer this in your HTML file. Create the index.html with the script tag pointing to the hellowWorld2.js



|  |  |
| --- | --- |
|  |  |

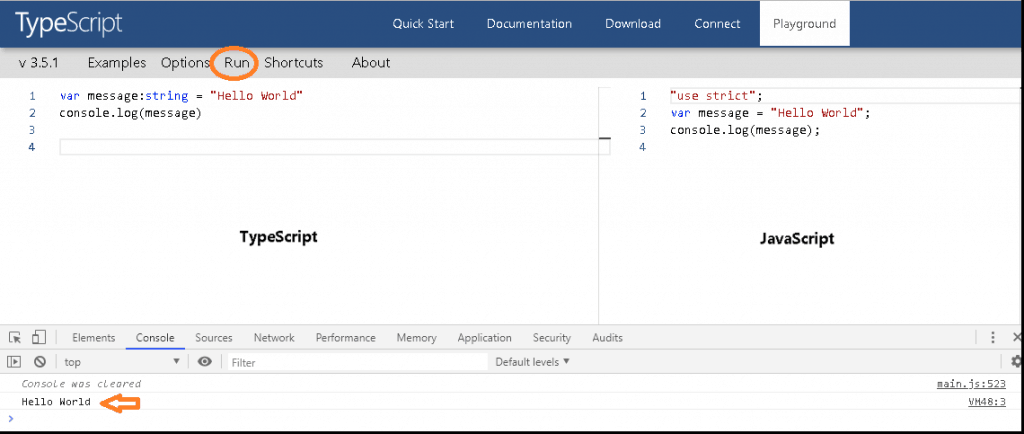
Open the file using the chrome or any browser, you should be able to see the alert window.

Running in Typescript Playground

You can also run typescript in using the TypeScript Playground. Head over to Typescript Playground <https://www.typescriptlang.org/play/index.html#code/G4QwTgBAtgpgznEBzGAuOAXMBLAdkiAXggCIAJGAG0oHsIB1GsSgExIgCgBjG3OGyjAB0tJAApYCZDACUHDkA>

As you type the code, you will see the javascript code is automatically generated in the right-hand side pane.

Open the Developer Console (ctrl-shit-I in chrome) and click on run and see the Hello World appearing in the console window.



## Compile Error

As we mentioned in the previous article on, the typescript compiler warns us if there are any **type** mismatches.

For Example, create helloWorld3.ts and add the following code. Here we are assigning a string to a number variable.



Instantly the Visual Studio Code underlines the message in red, indicating the error in the code. Also, when you compile, the compiler will throw the following error.





|  |  |
| --- | --- |
|  |  |

Typescript will generate the javascript code even if there is a compile error. The helloWorld3.ts shown above contains an error but the helloWorld3.js is still produced. This is intentional. This allows us to export our javascript code to typescript and progressively update it to Typescript.

1. [**Enable Compile on Save in Visual Studio Code**](https://www.tektutorialshub.com/typescript/typescript-compile-on-save-in-visual-studio-code/)