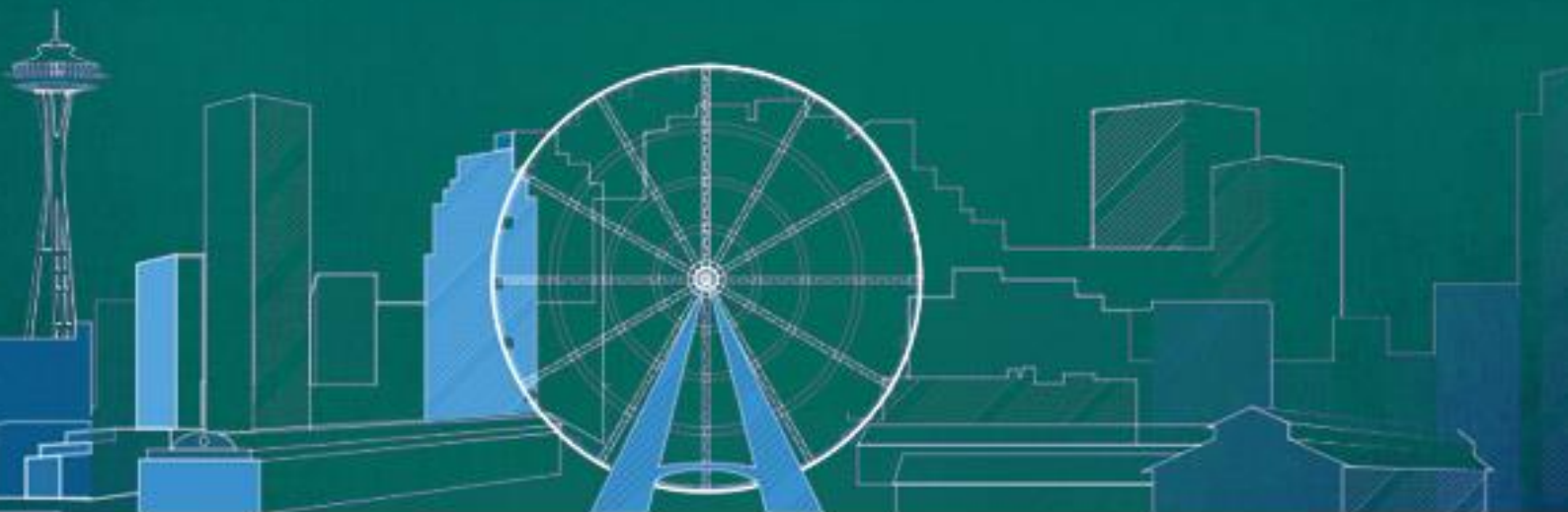




TypeScript



tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

TypeScript lets you write JavaScript the way you really want to. TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java. The popular JavaScript framework **Angular 2.0** is written in TypeScript. Mastering TypeScript can help programmers to write object-oriented programs and have them compiled to JavaScript, both on server side and client side.

Audience

Programmers coming from Object Oriented world will find it easy to use TypeScript. With the knowledge of TypeScript, they can build web applications much faster, as TypeScript has good tooling support.

Prerequisites

As a reader of this tutorial, you should have a good understanding of OOP concepts and basic JavaScript, to make the most of this tutorial.

Copyright & Disclaimer

© Copyright 2016 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial.....	i
Audience	i
Prerequisites	i
Copyright & Disclaimer.....	i
Table of Contents	ii
 1. TYPESCRIPT – OVERVIEW.....	 1
What is TypeScript?.....	1
Features of TypeScript.....	1
Why Use TypeScript?	2
Components of TypeScript	3
 2. TYPESCRIPT – ENVIRONMENT SETUP	 4
TypeScript – Try it Option Online	4
Local Environment Setup.....	5
Installing Node.js.....	5
IDE Support	8
Visual Studio Code	8
Brackets	11
 3. TYPESCRIPT – BASIC SYNTAX.....	 14
Your First TypeScript Code	14
Compile and Execute a TypeScript Program	14
Compiler Flags.....	15
Identifiers in TypeScript	16
TypeScript – Keywords	16
Comments in TypeScript	17

TypeScript and Object Orientation	18
4. TYPESCRIPT – TYPES	20
The Any type	20
Built-in types	20
5. TYPESCRIPT – VARIABLES.....	22
Variable Declaration in TypeScript	22
Type Assertion in TypeScript	24
Inferred Typing in TypeScript	25
TypeScript Variable Scope	25
6. TYPESCRIPT – OPERATORS.....	28
What is an Operator?	28
Arithmetic Operators	28
Relational Operators	30
Logical Operators	32
Short-circuit Operators (&& and).....	33
Bitwise Operators	34
Assignment Operators.....	36
Miscellaneous Operators	38
Type Operators	39
7. TYPESCRIPT – DECISION MAKING	41
The if Statement.....	42
The if...else Statement.....	43
The else...if Ladder	45
The switch...case Statement	46

8.	TYPESCRIPT – LOOPS	51
	The while Loop	52
	The for Loop	54
	The for...in loop	56
	The do...while loop	57
	The break Statement	60
	The continue Statement	61
	The Infinite Loop	63
9.	TYPESCRIPT – FUNCTIONS	65
	Defining a Function	65
	Calling a Function	65
	Returning Functions	66
	Parameterized Function	68
	Optional Parameters	69
	Rest Parameters	70
	Default Parameters	72
	Anonymous Function	73
	The Function Constructor	74
	Recursion and TypeScript Functions	75
	Lambda Functions	76
	Syntactic Variations	77
	Function Overloads	80
10.	TYPESCRIPT – NUMBERS	82
	Number Methods	85
	toExponential()	85
	toFixed()	86

toLocaleString()	87
toPrecision()	87
toString()	88
valueOf()	89
11. TYPESCRIPT – STRINGS	90
String Methods	92
charAt	93
charCodeAt()	94
concat()	95
indexOf()	95
lastIndexOf()	96
localeCompare()	97
replace()	98
search()	99
slice()	100
split()	101
substr()	101
substring()	102
toLocaleLowerCase()	103
toLocaleUpperCase()	104
toLowerCase()	104
toString()	105
toUpperCase()	105
valueOf()	106

12. TYPESCRIPT – ARRAYS.....	107
Features of an Array.....	107
Declaring and Initializing Arrays	107
Accessing Array Elements.....	108
Array Object	109
Array Methods	111
concat().....	112
every().....	112
filter().....	113
forEach()	114
indexOf()	116
join()	116
lastIndexOf()	117
map()	118
pop()	119
push().....	119
reduce().....	120
reduceRight()	121
reverse().....	121
shift()	122
slice()	122
some().....	123
sort()	124
splice()	124
toString().....	126
unshift()	126

Array Destructuring.....	127
Array Traversal using for...in loop.....	127
Arrays in TypeScript	128
Multidimensional Arrays	128
13. TYPESCRIPT – TUPLES	133
Accessing values in Tuples.....	133
Tuple Operations.....	134
Updating Tuples	135
Destructuring a Tuple	135
14. TYPESCRIPT – UNION.....	137
Union Type and Arrays	139
15. TYPESCRIPT – INTERFACES.....	141
Declaring Interfaces	141
Union Type and Interface	143
Interfaces and Arrays	145
Interfaces and Inheritance	145
16. TYPESCRIPT – CLASSES.....	148
Creating classes	148
Creating Instance objects	150
Accessing Attributes and Functions	150
Class Inheritance	152
TypeScript – Class inheritance and Method Overriding	155
The static Keyword.....	157
The instanceof operator	158
Data Hiding	158

Classes and Interfaces	159
17. TYPESCRIPT – OBJECTS	161
TypeScript Type Template	162
Duck-typing	164
18. TYPESCRIPT – NAMESPACES	166
Defining a Namespace	166
Nested Namespaces	169
19. TYPESCRIPT – MODULES	171
Internal Module	171
External Module	172
20. TYPESCRIPT – AMBIENTS	177
Defining Ambients	177

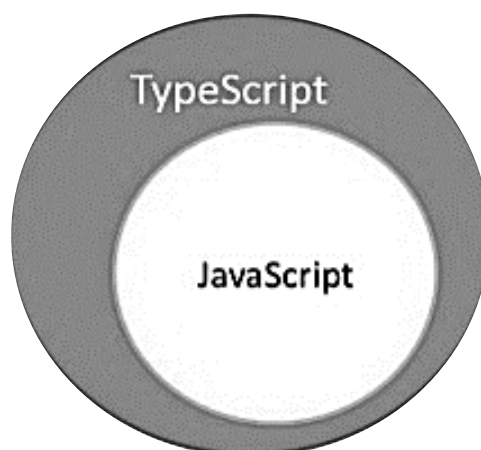
1. TypeScript – Overview

JavaScript was introduced as a language for the client side. The development of Node.js has marked JavaScript as an emerging server-side technology too. However, as JavaScript code grows, it tends to get messier, making it difficult to maintain and reuse the code. Moreover, its failure to embrace the features of Object Orientation, strong type checking and compile-time error checks prevents JavaScript from succeeding at the enterprise level as a full-fledged server-side technology. **TypeScript** was presented to bridge this gap.

What is TypeScript?

By definition, “TypeScript is JavaScript for application-scale development.”

TypeScript is a strongly typed, object oriented, compiled language. It was designed by **Anders Hejlsberg** (designer of C#) at Microsoft. TypeScript is both a language and a set of tools. TypeScript is a typed superset of JavaScript compiled to JavaScript. In other words, TypeScript is JavaScript plus some additional features.



Features of TypeScript

TypeScript is just JavaScript. TypeScript starts with JavaScript and ends with JavaScript. Typescript adopts the basic building blocks of your program from JavaScript. Hence, you only need to know JavaScript to use TypeScript. All TypeScript code is converted into its JavaScript equivalent for the purpose of execution.

TypeScript supports other JS libraries. Compiled TypeScript can be consumed from any JavaScript code. TypeScript-generated JavaScript can reuse all of the existing JavaScript frameworks, tools, and libraries.

JavaScript is TypeScript. This means that any valid **.js** file can be renamed to **.ts** and compiled with other TypeScript files.

TypeScript is portable. TypeScript is portable across browsers, devices, and operating systems. It can run on any environment that JavaScript runs on. Unlike its counterparts, TypeScript doesn't need a dedicated VM or a specific runtime environment to execute.

TypeScript and ECMAScript

The ECMAScript specification is a standardized specification of a scripting language. There are six editions of ECMA-262 published. Version 6 of the standard is codenamed "Harmony". TypeScript is aligned with the ECMAScript6 specification.



TypeScript adopts its basic language features from the ECMAScript5 specification, i.e., the official specification for JavaScript. TypeScript language features like Modules and class-based orientation are in line with the EcmaScript 6 specification. Additionally, TypeScript also embraces features like generics and type annotations that aren't a part of the EcmaScript6 specification.

Why Use TypeScript?

TypeScript is superior to its other counterparts like CoffeeScript and Dart programming languages in a way that TypeScript is extended JavaScript. In contrast, languages like Dart, CoffeeScript are new languages in themselves and require language-specific execution environment.

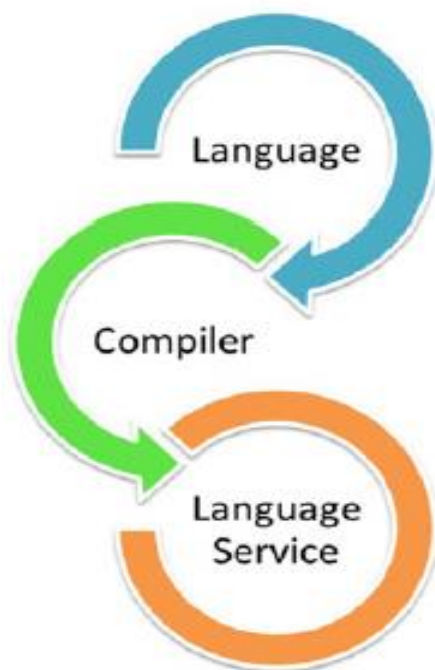
The benefits of TypeScript include:

- **Compilation:** JavaScript is an interpreted language. Hence, it needs to be run to test that it is valid. It means you write all the codes just to find no output, in case there is an error. Hence, you have to spend hours trying to find bugs in the code. The TypeScript transpiler provides the error-checking feature. TypeScript will compile the code and generate compilation errors, if it finds some sort of syntax errors. This helps to highlight errors before the script is run.
- **Strong Static Typing:** JavaScript is not strongly typed. TypeScript comes with an optional static typing and type inference system through the TLS (TypeScript Language Service). The type of a variable, declared with no type, may be inferred by the TLS based on its value.
- TypeScript **supports type definitions** for existing JavaScript libraries. TypeScript Definition file (with **.d.ts** extension) provides definition for external JavaScript libraries. Hence, TypeScript code can contain these libraries.
- TypeScript **supports Object Oriented Programming** concepts like classes, interfaces, inheritance, etc.

Components of TypeScript

At its heart, TypeScript has the following three components:

- **Language:** It comprises of the syntax, keywords, and type annotations.
- **The TypeScript Compiler:** The TypeScript compiler (tsc) converts the instructions written in TypeScript to its JavaScript equivalent.
- **The TypeScript Language Service:** The "Language Service" exposes an additional layer around the core compiler pipeline that are editor-like applications. The language service supports the common set of a typical editor operations like statement completions, signature help, code formatting and outlining, colorization, etc.



Declaration Files

When a TypeScript script gets compiled, there is an option to generate a **declaration file** (with the extension **.d.ts**) that functions as an interface to the components in the compiled JavaScript. The concept of declaration files is analogous to the concept of header files found in C/C++. The declaration files (files with **.d.ts** extension) provide intellisense for types, function calls, and variable support for JavaScript libraries like jQuery, MooTools, etc.

2. TypeScript – Environment Setup

In this chapter, we will discuss how to install TypeScript on Windows platform. We will also explain how to install the Brackets IDE.

TypeScript – Try it Option Online

You may test your scripts online by using The TypeScript editor at <http://www.typescriptlang.org/Playground>. The online editor shows the corresponding JavaScript emitted by the compiler.



You may try the following example using **Playground**.

```
var num:number=12
console.log(num)
```

On compiling, it will generate following JavaScript code

```
//Generated by typescript 1.8.10
var num = 12;
console.log(num);
```

The output of the above program is given below:

```
12
```

Local Environment Setup

Typescript is an Open Source technology. It can run on any browser, any host, and any OS. You will need the following tools to write and test a Typescript program:

A Text Editor

The text editor helps you to write your source code. Examples of a few editors include Windows Notepad, Notepad++, Emacs, vim or vi, etc. Editors used may vary with Operating Systems.

The source files are typically named with the extension **.ts**

The TypeScript Compiler

The TypeScript compiler is itself a **.ts** file compiled down to JavaScript (.js) file. The TSC (TypeScript Compiler) is a source-to-source compiler (transcompiler / transpiler).



The TSC generates a JavaScript version of the **.ts** file passed to it. In other words, the TSC produces an equivalent JavaScript source code from the Typescript file given as an input to it. This process is termed as transpilation.

However, the compiler rejects any raw JavaScript file passed to it. The compiler deals with only **.ts** or **.d.ts** files.

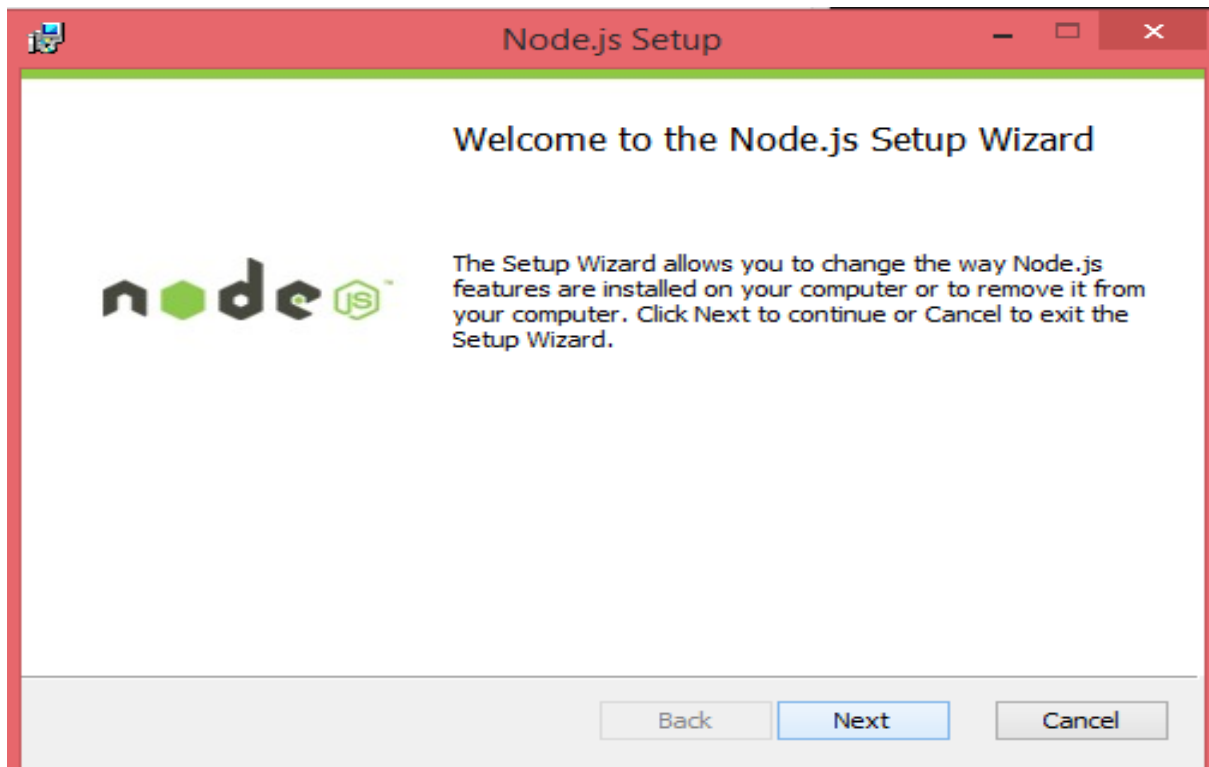
Installing Node.js

Node.js is an open source, cross-platform runtime environment for server-side JavaScript. Node.js is required to run JavaScript without a browser support. It uses Google V8 JavaScript engine to execute code. You may download Node.js source code or a pre-built installer for your platform. Node is available here: <https://nodejs.org/en/download>

Installation on Windows

Follow the steps given below to install Node.js in Windows environment.

Step 1: Download and run the .msi installer for Node.

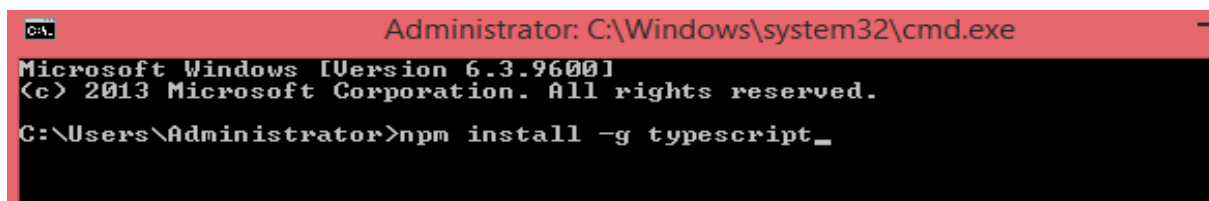


Step 2: To verify if the installation was successful, enter the command ***node -v*** in the terminal window.

```
C:\Users>node -v
v4.2.3
C:\Users>_
```

Step 3: Type the following command in the terminal window to install TypeScript.

```
npm install -g typescript
```

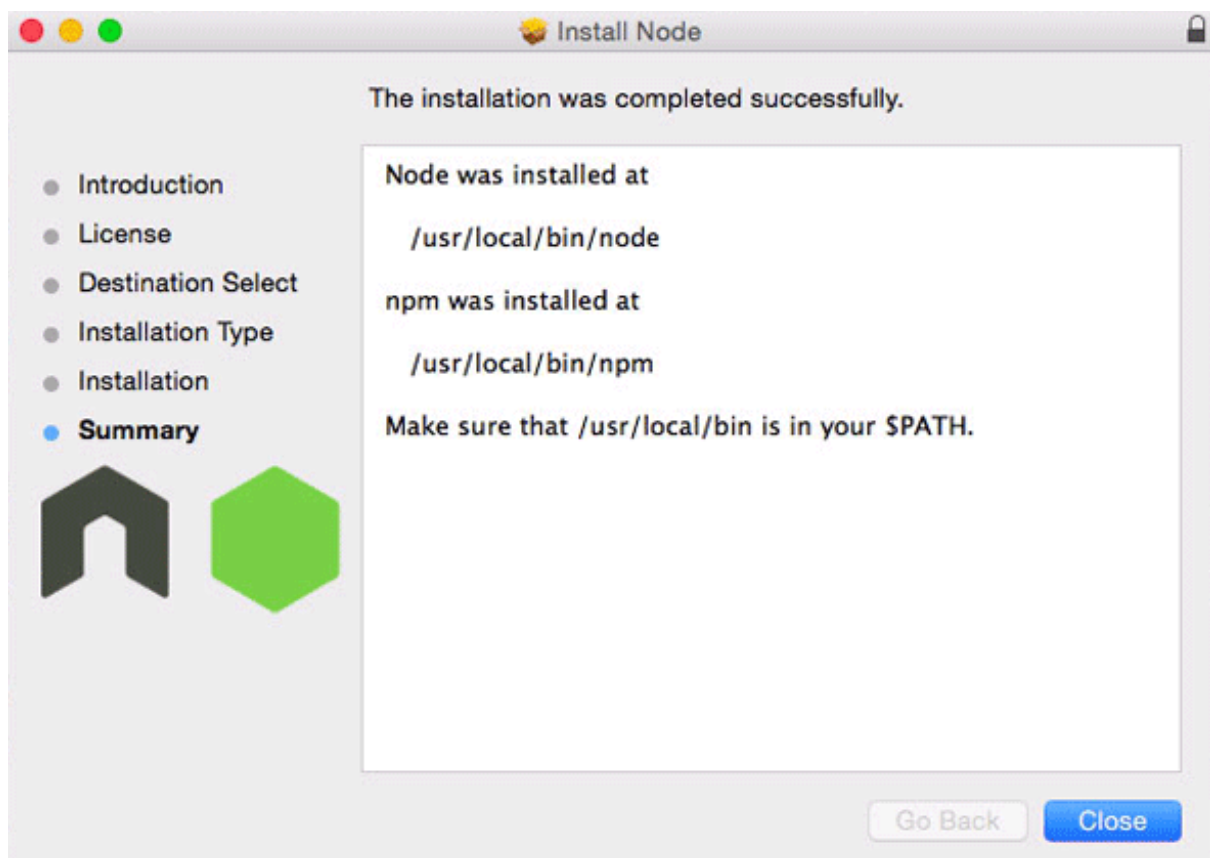


Installation on Mac OS X

To install node.js on Mac OS X, you can download a pre-compiled binary package which makes a nice and easy installation. Head over to <http://nodejs.org/> and click the install button to download the latest package.



Install the package from the **.dmg** by following the install wizard which will install both node and **npm**. npm is Node Package Manager which facilitates installation of additional packages for node.js.



Installation on Linux

You need to install a number of dependencies before you can install Node.js and NPM.

- **Ruby** and **GCC**. You'll need Ruby 1.8.6 or newer and GCC 4.2 or newer.
- **Homebrew**. Homebrew is a package manager originally designed for Mac, but it's been ported to Linux as Linuxbrew. You can learn more about Homebrew at <http://brew.sh> and Linuxbrew at <http://brew.sh/linuxbrew>.

Once these dependencies are installed, you may install Node.js by using the following command on the terminal:

```
brew install node.
```

IDE Support

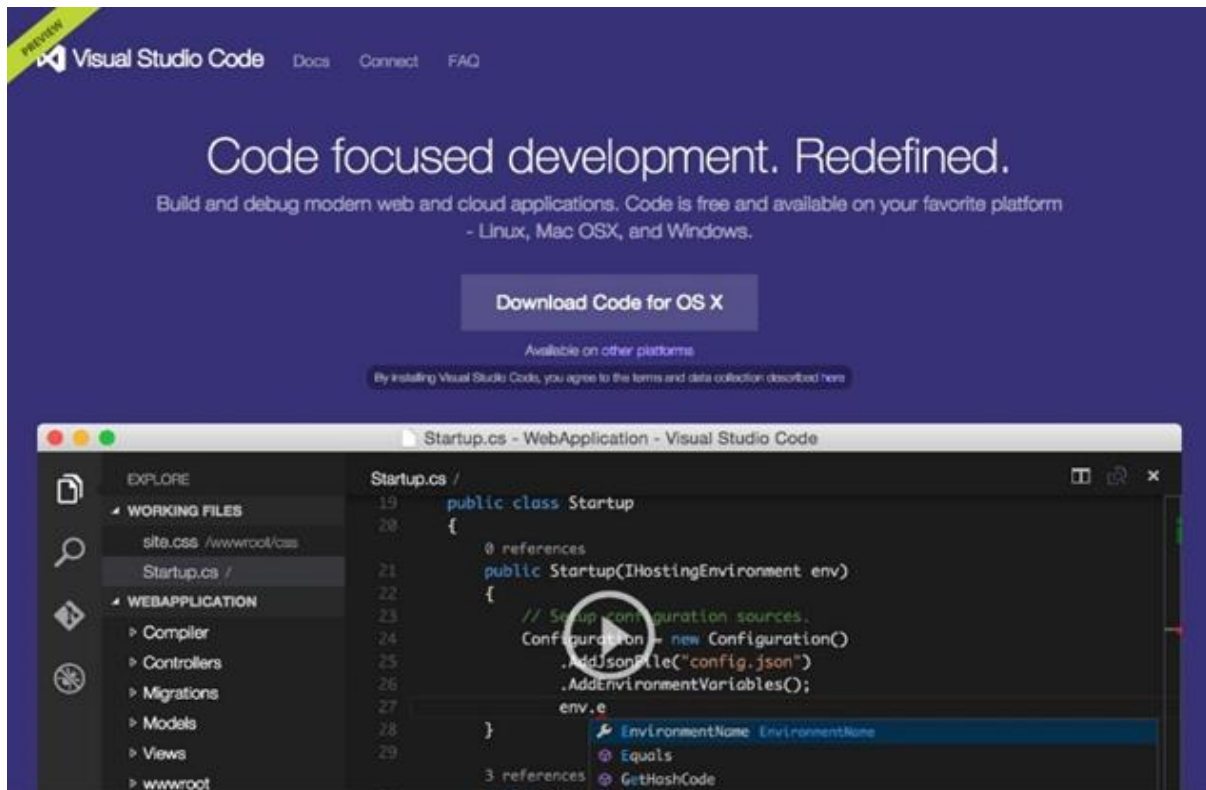
Typescript can be built on a plethora of development environments like Visual Studio, Sublime Text 2, WebStorm/PHPStorm, Eclipse, Brackets, etc. Visual Studio Code and Brackets IDEs are discussed here. The development environment used here is Visual Studio Code (Windows platform).


Visual Studio Code

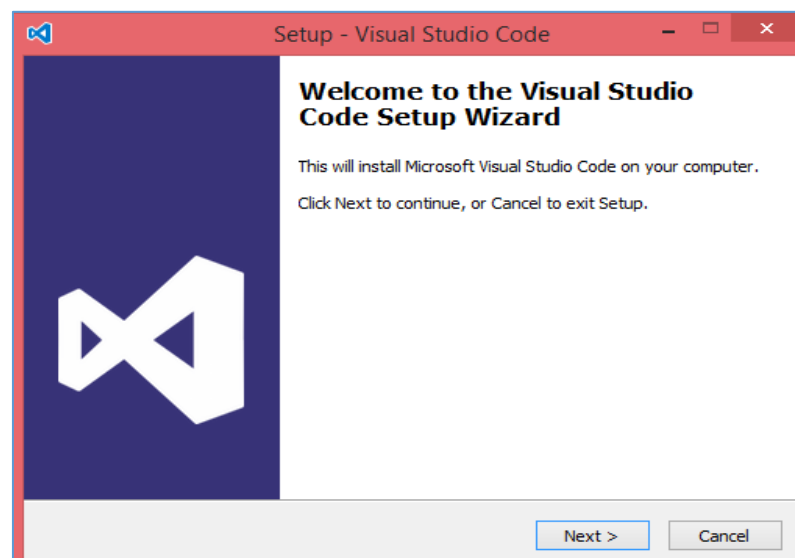
This is an open source IDE from Visual Studio. It is available for Mac OS X, Linux and Windows platforms. VScode is available at: <https://code.visualstudio.com>

Installation on Windows

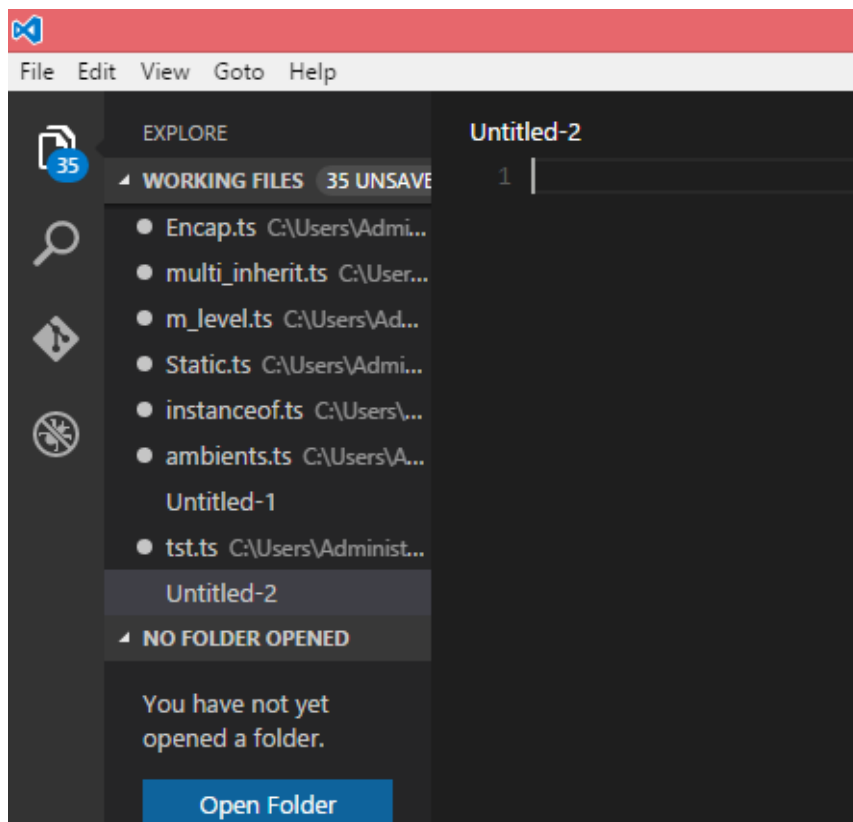
Step 1: Download Visual Studio Code for Windows.



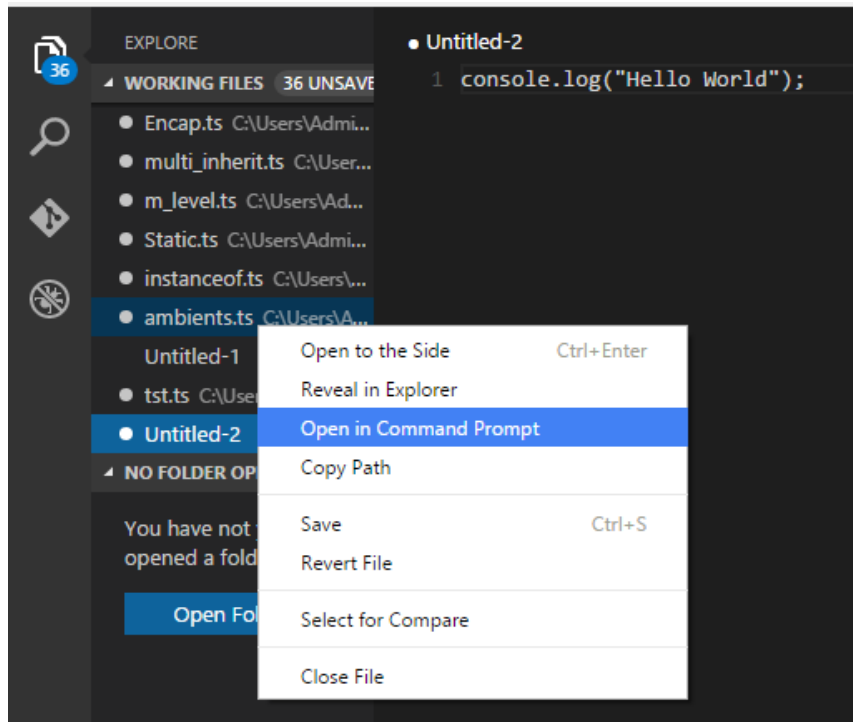
Step 2: Double-click on VSCodeSetup.exe  to launch the setup process. This will only take a minute.



Step 3: A screenshot of the IDE is given below:



Step 4: You may directly traverse to the file's path by right clicking on the file → open in command prompt. Similarly, the Reveal in Explorer option shows the file in the File Explorer.



Installation on Mac OS X

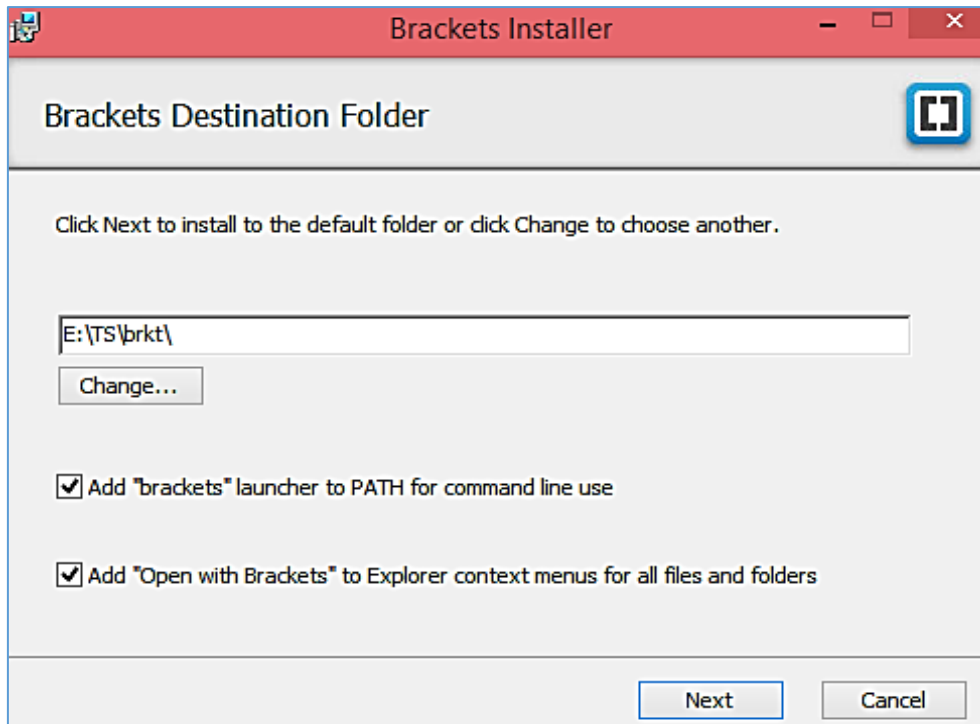
Visual Studio Code's Mac OS X specific installation guide can be found at <https://code.visualstudio.com/Docs/editor/setup>

Installation on Linux

Linux specific installation guide for Visual Studio Code can be found at <https://code.visualstudio.com/Docs/editor/setup>


Brackets

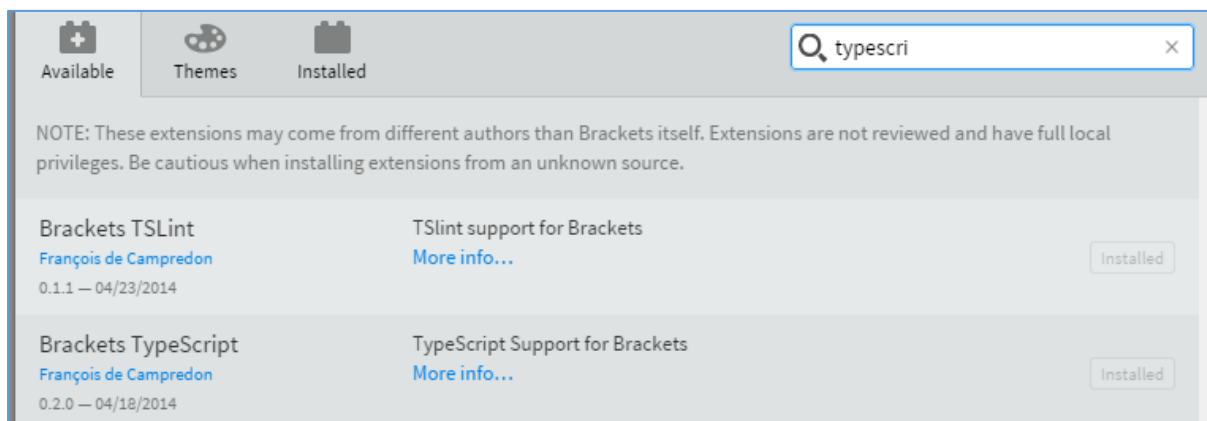
Brackets is a free open-source editor for web development, created by Adobe Systems. It is available for Linux, Windows and Mac OS X. Brackets is available at <http://brackets.io>



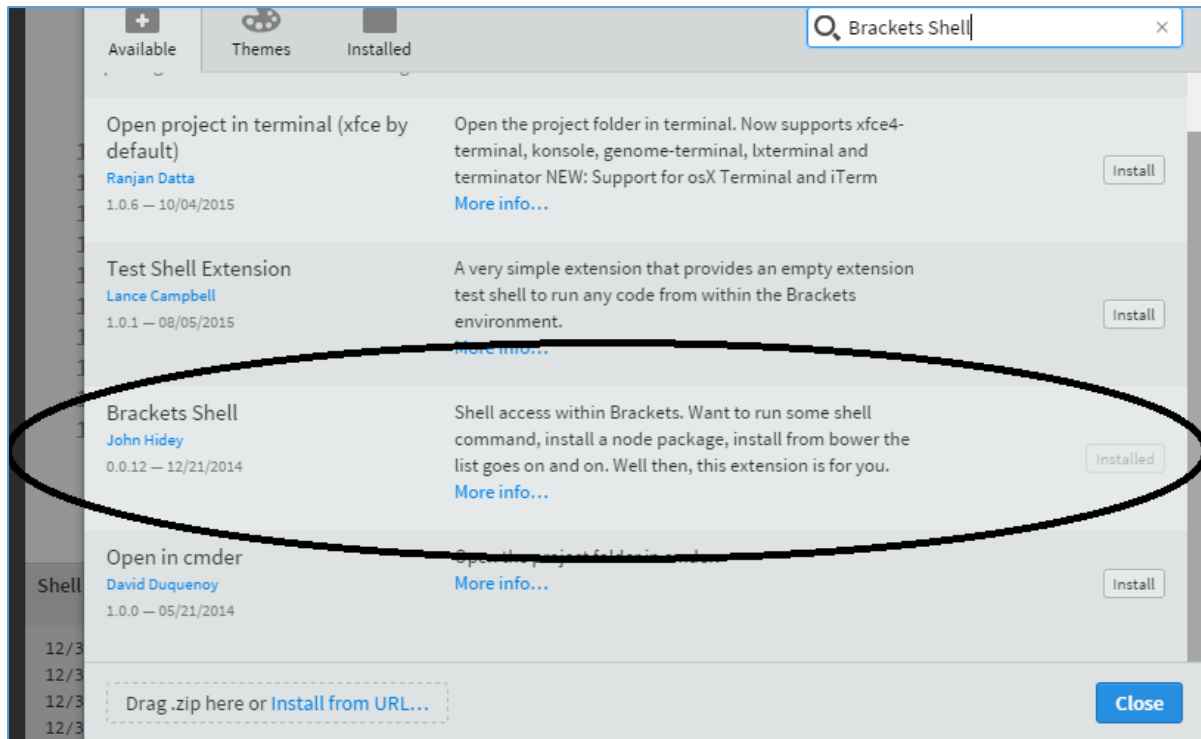
TypeScript Extensions for Brackets


Brackets supports extensions for adding extra functionality via the Extension Manager. The following steps explain installing TypeScript extensions using the same.

- Post installation, click on the extension manager icon  on the right-hand side of the editor. Enter typescript in the search box.
- Install the Brackets TSLint and Brackets TypeScript plugins.



You can run DOS prompt / shell within Brackets itself by adding one more extension Brackets Shell.



Upon installation, you will find an icon of shell on the right-hand side of the editor . Once you click on the icon, you will see the shell window as shown below:

```

Shell

D:\ts-projects>dir

Volume in drive D is New Volume
Volume Serial Number is B86C-C26C

Directory of D:\ts-projects

    10:23 PM    <DIR>        .
    10:23 PM    <DIR>        ..
           0 File(s)                0 bytes
           2 Dir(s)  93,937,332,224 bytes free

D:\ts-projects>

```

Note: Typescript is also available as a plugin for Visual Studio 2012 and 2013 environments (<http://www.typescriptlang.org/#Download>). VS 2015 and above includes TypeScript plugin by default.

Now, you are all set!!!

3. TypeScript – Basic Syntax

Syntax defines a set of rules for writing programs. Every language specification defines its own syntax. A TypeScript program is composed of:

- Modules
- Functions
- Variables
- Statements and Expressions
- Comments

Your First TypeScript Code

Let us start with the traditional "Hello World" example:

```
var message:string="Hello World"
console.log(message)
```

On compiling, it will generate following JavaScript code

```
//Generated by typescript 1.8.10
var message = "Hello World";
console.log(message);
```

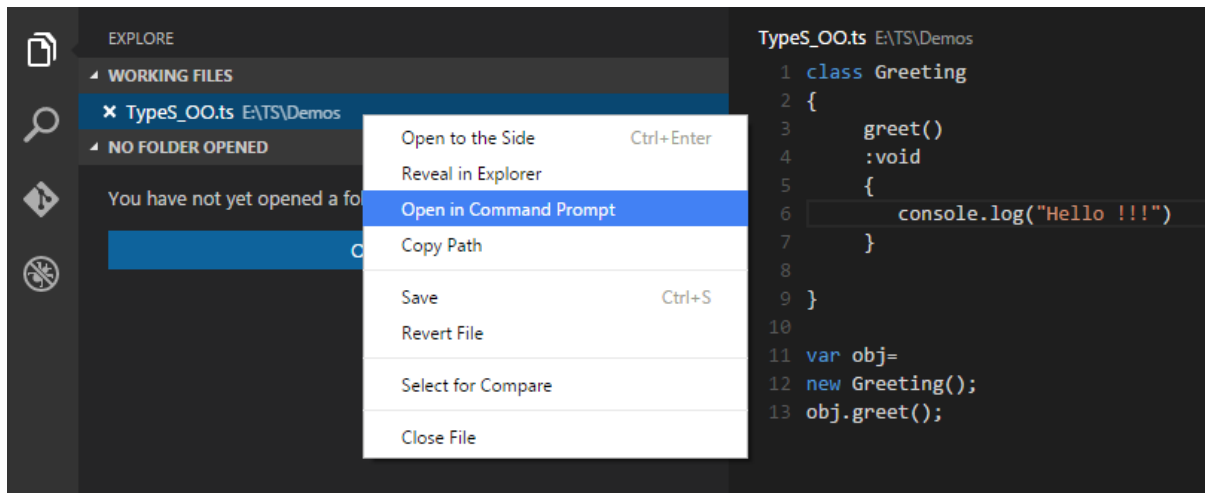
- Line 1 declares a variable by the name message. Variables are a mechanism to store values in a program.
- Line 2 prints the variable's value to the prompt. Here, console refers to the terminal window. The function *log ()* is used to display text on the screen.

Compile and Execute a TypeScript Program

Let us see how to compile and execute a TypeScript program using Visual Studio Code. Follow the steps given below:

Step 1: Save the file with .ts extension. We shall save the file as Test.ts. The code editor marks errors in the code, if any, while you save it.

Step 2: Right-click the TypeScript file under the Working Files option in VS Code's Explore Pane. Select Open in Command Prompt option.



Step 3: To compile the file use the following command on the terminal window:

```
tsc Test.ts
```

Step 4: The file is compiled to Test.js. To run the program written, type the following in the terminal:

```
node Test.js
```


End of ebook preview

If you liked what you saw...

Buy it from our store @ **<https://store.tutorialspoint.com>**