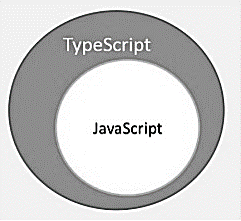
**TypeScript**

**Frequently Asked Questions**

1. What is TypeScript and Why Do We Need It?

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.



1. What are different Components of TypeScript?  
    There are mainly 3 components of TypeScript:
2. **Language** – The most important part for developers is the new language. The language consists of new syntax, keywords and allows you to write TypeScript.
3. **Compiler** – The TypeScript compiler is open source, cross-platform and open specification, and is written in TypeScript. Compiler will compile your TypeScript into JavaScript. And it will also emit error, if any. It can also help in concatenating different files to a single output file and in generating source maps.
4. **Language Service** – TypeScript language service which powers the interactive TypeScript experience in Visual Studio, [VS Code](http://www.talkingdotnet.com/what-is-visual-studio-code-and-difference-between-visual-studio-2015/), Sublime, the TypeScript playground and other editor.
5. How can you get TypeScript and Install it?

TypeScript can be installed and managed via npm, the Node.js package manager. To install TypeScript, first ensure the npm is installed properly. And then run the following command to install TypeScript globally on your system.

npm install -g typescript

TypeScript is included in Visual Studio 2013 Update 2 and Visual Studio 2015 by default. TypeScript also provides support for other editors like [Visual Studio Code](http://www.talkingdotnet.com/what-is-visual-studio-code-and-difference-between-visual-studio-2015/), sublime, Emacs and Vim.

1. How do You compile TypeScript files?

The extension for any TypeScript file is **“*.ts*”**. And any JavaScript file is TypeScript file as it is a superset of JavaScript. So change extension of “*.js*” to “*.ts*” file and your TypeScript file is ready. To compile any *.ts* file into *.js*, use the following command.

tsc <TypeScript File Name>

For example, to compile “*Helloworld.ts*”:

tsc helloworld.ts

and the result would be *helloworld.js*.

1. Is it possible to combine .ts files into a Single Multiple .js File?

Yes, it's possible. While compiling add --outFILE [OutputJSFileName] option.

tsc --outFile comman.js file1.ts file2.ts file3.ts

This will compile all 3 “*.ts*” file and output into single “*comman.js*” file. And what will happen if you don’t provide a output file name.

tsc --outFile file1.ts file2.ts file3.ts

In this case, *file2.ts* and *file3.ts* will be compiled and the output will be placed in *file1.ts*. So now your *file1.ts*contains JavaScript code.

1. Is It Possible to Compile .ts Automatically with Real Time Changes in .ts File?

Yes. Using --watch compiler option, this can be achieved.

tsc --watch file1.ts

This will first compile *file1.ts* in *file.js* and watch for the file changes. As soon as there is any change detected, it will compile it again. But you need to ensure that command prompt must not be closed, used with --watchoption.

1. Which Object Oriented Terms are Supported by TypeScript?

  TypeScript supports the following object oriented terms:

* Modules
* Classes
* Interfaces
* Data Types
* Member functions

1. Which are the Different Data Types Supported by TypeScript?  
     
    TypeScript supports the following data types:

* Boolean var bValue: boolean = false;
* Number var age: number = 16;
* String var name: string = "jon";
* Array var list:number[] = [1, 2, 3];
* Enum

1. How do you implement Inheritance in TypeScript?

Using extends keyword, we can implement inheritance.

class Animal {

public domestic:boolean;

constructor(public name: string) { }

}

class Cat extends Animal {

constructor(name: string, domestic: boolean)

{

super(name);

this.domestic = true;

}

}

class Tiger extends Animal {

constructor(name: string, domestic: boolean)

{

super(name);

this.domestic = false;