**TypeScript**

**TypeScript Definition:**

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 Type Script:**

**Type is just Java Script:**

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.

**Java Script 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 ECMA Script 6 specification. Additionally, TypeScript also embraces features like generics and type annotations that aren’t a part of the EcmaScript6 specification.

**Why we use Type Script?**

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 Type Script:**

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 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 intelligence for types, function calls, and variable support for JavaScript libraries like jQuery, MooTools, etc.



Every language specification defines its own syntax. A TypeScript program is composed of

* Modules
* Functions
* Variables
* Statements and Expressions
* Comments

Your First Type Script Code

Let us start with the traditional “Hello World” example

var message:string=”Hello World”

console.log (message)

**Compile and Execute a Type Script** **Programme:**

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

**Step1**. 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

**Compiler Flag:**

Compiler flags enable you to change the behavior of the compiler during compilation. Each compiler flag exposes a setting that allows you to change how the compiler behaves.

|  |  |
| --- | --- |
| **S.No.** | **Compiler flag & Description** |
| 1. | **--help**  Displays the help manual |
| 2. | **--module**  Load external modules |
| 3. | **--target**  Set the target ECMA version |
| 4. | **--declaration**  Generates an additional .d.ts file |
| 5. | **--remove Comments**  Removes all comments from the output file |
| 6. | **--out**  Compile multiple files into a single output file |
| 7. | **--sourcemap**  Generate a source map (.map) files |
| 8. | **--module noImplicitAny**  Disallows the compiler from inferring the any type |
| 9. | **--watch**  Watch for file changes and recompile them on the fly |

**Identifiers in Type Script:**

Identifiers are names given to elements in a program like variables, functions.

Identifiers Are:

* Identifiers can include both, characters and digits. However, the identifier cannot begin with a digit.
* Identifiers cannot include special symbols except for underscore (\_) or a dollar sign ($).
* Identifiers cannot be keywords.
* They must be unique.
* Identifiers are case-sensitive.
* Identifiers cannot contain spaces.

The Following Tables Lists a few of valid and Invalid Identifiers:-

|  |  |
| --- | --- |
| **Valid identifiers** | **Invalid identifiers** |
| first Name | Var |
| first\_name | first name |
| num1 | first-name |
| $result | 1number |

**TypeScript Key:**

Keywords have a special meaning in the context of a language.

|  |  |  |  |
| --- | --- | --- | --- |
| break | as | any | switch |
| case | if | throw | else |
| var | number | string | get |
| module | type | instanceof | typeof |
| public | private | enum | export |
| finally | for | while | void |
| null | super | this | new |
| in | return | true | false |
| any | extends | static | let |
| package | implements | interface | function |
| new | try | yield | const |
| continue | do | catch |  |

**Whitespace and Line Breaks:**

TypeScript ignores spaces, tabs, and newlines that appear in programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

**TypeScript is Case-sensitive:**

TypeScript is case-sensitive. This means that TypeScript differentiates between uppercase and lowercase characters.

**Semicolons are Optional**

Each line of instruction is called a **statement**. Semicolons are optional in TypeScript.

**Example** console.log (“hello World”)

console.log(“we are learning TypeScript”)

**Comments in TypeScript:**

Comments are a way to improve the readability of a program. Comments can be used to include additional information about a program like author of the code, hints about a function/ construct etc. Comments are ignored by the compiler.

TypeScript supports the following types of comments

* **Single-line comments ( // )** − Any text between a // and the end of a line is treated as a comment
* **Multi-line comments (/\* \*/)** − These comments may span multiple lines.

**Example** //this is single line comment

/\* This is a Multi-line comment\*/

**TypeScript and Object Orientation:**

class Greeting {

greet():void {

console.log("Hello World!!!")

}

}

var obj = new Greeting();

obj.greet();

The above example defines a class *Greeting*. The class has a method *greet ()*. The method prints the string “Hello World” on the terminal. The **new** keyword creates an object of the class (obj). The object invokes the method *greet ()*.

On compiling, it will generate following JavaScript code.

//Generated by typescript 1.8.10

var Greeting = (function () {

function Greeting () {

}

Greeting.prototype.greet = function () {

console.log ("Hello World!!!");

};

return Greeting;

}());

var obj = new Greeting ();

obj.greet()

The output of the above program is given below

Hello World!!!

Q1.What are the Challenges in Modern Web Development?

Ans: 1.Unified UX

2.Fluid UX

3.User and SEO Friendly URL's(Routing)

4.Loosely Coupled and Extensible Architecture.

5.Simplified Deployment and Install

Q2.What is Solution?

Ans: 1:Better I will Build a SPA(AngularJs) For you world popular SPA Twitter.Youtube is a Podcasting.

2.Better I will now build a progressive Web Appication on Angular7.

Q3.How to Build SPA?

Ans:You can use JavaScript and HTML .

jQuery HTML

Then why Angular AngularJs.

Q4.What is Problem in JavaScript and jQuery?

Ans:JavaScript is a Language,jQuery is a Library and AngularJs is Framework.Angular7 is a Platform.

Q5.What is problem jQuery?

Ans:jQuery is a Library.



Q.



**Basic Requirement to build End to End Web Application:**



**Language/Tool Description**

HTML This is a markup Language used for Presentation.

CSS It make the presentation more responsive.

**Client Side Script:**

It reduces the burden on server by handling Interactions and Validations Client Side.

Example:JavaScript,ECMAScript

**Server Side Script:**

It is a script employed on Server to generate response customized for ever Client request.

Example:PHP,JSP,ASP

**Database:**

It defines the data,queries,reports and analysis required for application.

Ex.RDBMS,Oracle,Non-RDBMS

**Middleware:**

It is a software framework used in Multitier Application to handle communication

Example:JBOSS,Express

**IDE'S:**

It provides an Integrated Development Environment to Build,Debug,Test and Deploy Applications.

**DTP:**

Publishing Tools Required for Publishing Images and Animation for web

**Note:**

A Software Stack can be used for buiding End to End Application,Which includes MEAN,MERN etc.

M-MongoDB[Database]

E-Express[Middleware]

A-Angular[ClientSide]

N-Nodejs[ServerSide]

R-Reactjs[ClientSide]

**Challenges in Modern Web Development:**

1.**Unified UX**

1.a web application must reach broad range of devices ie.from Browser to mobile

* It must Work on low Bandwidth devices
* It should not optimize its content
* Mobile users must get access to everything
* Application must have the same feel and look across devices.