TypeScript with ES6

HOW - Hands On Workshop

By Vijay Shivakumar

About me

Vijay ShivaKumar

Designer | Developer | Trainer

Training on web and Adobe products from past 10+ years

About the Program

Prerequisites

Introduction

History

Installation and configuration

The types in TypeScript

Disclaimer

Developers are fiercely opinionated about languages, especially JavaScript.

Topics I cover may change in future with new releases

A day or two for this topic is not justified.

(I encourage keep learning after this training too…)

I am not going to cover every thing in the language.

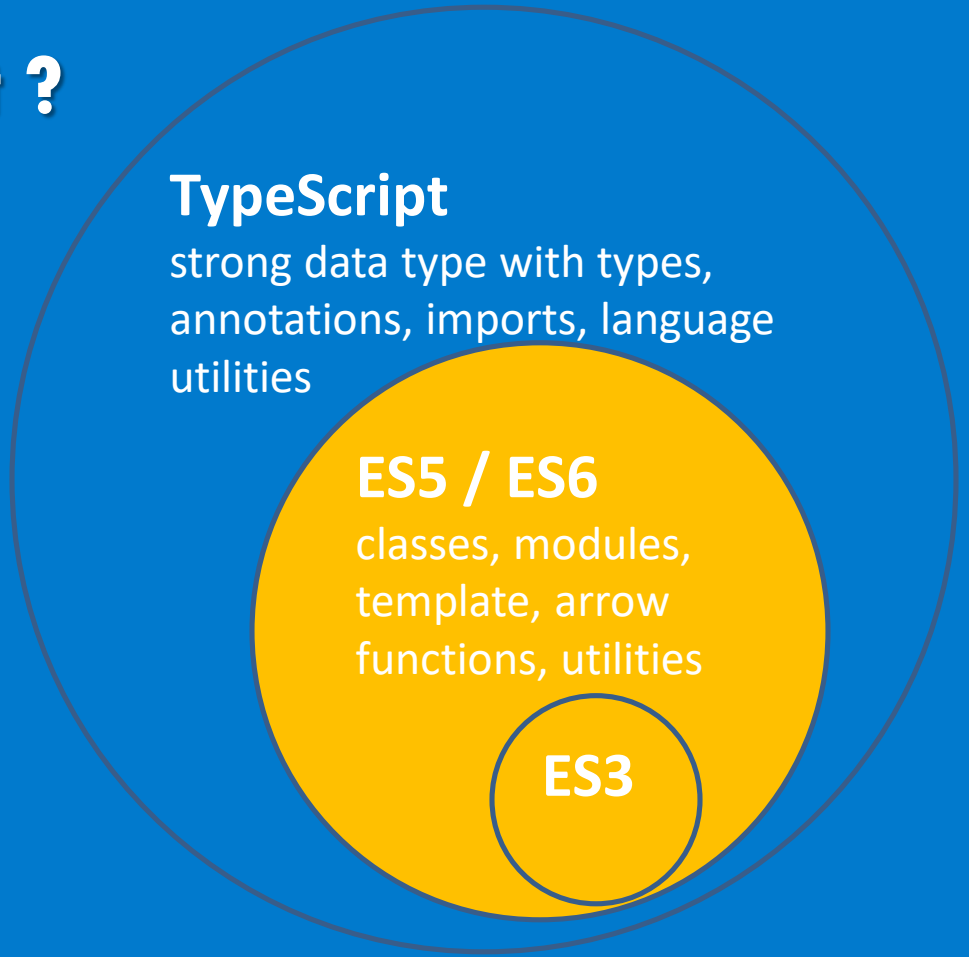
I encourage you to dig the documentation rather than rely online videos or tutors

(there are chances that they are out dated)

Before we begin…

- IDE
  - Aptana Studio
  - WebStorm (Recomended)
  - Sublime
  - Visual Studio Code
- Platform
  - NodeJS
  - Typescript Module

What is TypeScript ?



History and Milestones of JavaScript

JavaScript developed by Brendan Eich in 1995

Netscape released in 1996 supporting JavaScript

Microsoft created JScript dialect in 1996

Ecma International standardized ECMA-262
to settle disputes between browser vendors

ECMA-262 standardized ECMAScript

ECMA-262 had 5 editions

ECMAScript 6 (ES6) was finalized in June 2015

About TypeScript

The TypeScript programming language was developed by Developers at Microsoft.

It is an open source programming language.

The TypeScript code is compiled into JavaScript, so we can basically

use TypeScript wherever we use JavaScript.

Anders Hejlsberg

Why TypeScript ?

Its just a superset of JavaScript so any standard JavaScript is a valid TypeScript file

Give Types and Safety to JavaScript

Write error free code, faster…

Compiles to ES3 JavaScript by default

TypeScript lets you write JavaScript the way you really want to

"Microsoft's TypeScript may be the best of the many JavaScript front ends. It seems to generate the most attractive code...” - Douglas Crockford

Advantages and Disadvantages

- Advantage
  - More Flexible
  - Independent from Developer Environment
  - Easier to compile correctly
- Disadvantage
  - Complexity
  - Setup Time

Setup : Installation / Configuration

```
npm install -g typescript
```

`tsc -v`	`--version`
`tsc -h`	`--help`
`tsc -t`	`--target`
`tsc -w`	`--watch`
`tsc --outFile`	concat and output to a single file
`tsc --outDir`	compiles containing .ts files to .js
`tsc --sourceMap`	generates a .map file for code assist

https://www.typescriptlang.org/docs/handbook/compiler-options.html

What does TypeScript bring.. ?

Types / Generics

Function

Interfaces

Classes

Modules

Namespaces

Decorators

Types

- Boolean
- Number
- String
- Array
- Enum
- Null
- Undefined
- Any
- Function
- Void
- Object
- Classes
- Interfaces

Inference of Type is default behavior

```
let username = “guest”; // will be inferred as a string

function somefun(){
    return 123
};

let someval = “string”;
someval = somefun(); // error
```

Adding Type Annotations

```
let username:string = “guest”; // will remain a string
username = 007; // error

function somefun():number{
    return 123
};

let someval:string = “string”;
someval = somefun(); // error
```

Functions in Typescript

You can add types to arguments

Make arguments optional by using “?”

Set the return type of a function.. To make function not to return set it as void

```
let myfun = function(arg1:string, arg2:number, arg3?:string):string{
        return “welcome to your life”
    }
```

Enums | Enumerations

```
enum Power {weak, recovering, strong};       // weak = 0, recovering = 1, strong = 2;
enum Power {weak = 1, recovering, strong}; // weak = 1, recovering = 2, strong = 3;
enum Power {weak = 5, recovering, strong}; // weak = 5, recovering = 6, strong = 7;

let heroPower:Power = Power.recovering ; console.log(heroPower) // 6 as per the previous line
let powerString = Power[heroPower] ; console.log(powerString) // recovering
```

General Types

Implicit Type and Explicit Type

When to be explicit

Features of TypeScript

TypeScript has a structural type system.
Type-checking focuses on the 'shape' that values have.
Interfaces have no run-time representation.
Support optional parameters.
TypeScript supports ES6 for class-based OOP.
Public, protected and private members.
All properties are public in runtime.
Each member is public by default.
Static members are always public.
TypeScript does not support multiple inheritance.
TypeScript supports local types.

Thank you

please forward your comments and feedback

vijay.shivu@gmail.com