ITC002A

Typescript Fundamentals

Agenda

1

Getting started with Typescript

Javascript and ECMAScript, Usage and benefits, TSC Compiler

Introduction

- Javascript, a scripting language most often used for client-side web development.
- Also used for server-side with NodeJS
- Javascript, an implementation of the ECMAScript Standard

ECMAScript

- ECMAScript: **E**uropean **C**omputer **M**anufacturers **A**ssociation **S**cript
- Also known as ECMA-262
- Standard for scripting languages, first edition published in 1997
- Versions: ES1, ES2, ES3, ES4, ES5(2009), ES6 (2015)
- Since 2016: ES2016, ES2017, …, ES2022
- **ES6** is the version supported by new browsers and NodeJs

Problems

- Not designed for application scale development
- Does not support static styping
- Lack of structuring mechanisms
- No compile-time intellisence or assistance

"You can write large programs in JavaScript. You just can't maintain them"

Anders Hejlsberg (Father of C#)

Alternatives

CoffeeScript

ClojureScript

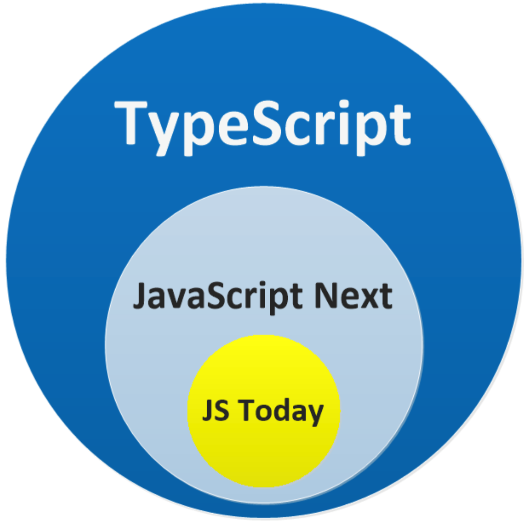
Dart

TypeScript

What is Typescript ?

- Typed **superset** of Javascript
- **Strongly typed object-oriented** programming language
- Compile to plain JS
- Support the latest ES features (ECMAScript)
- Designed by Anders Hejlsberg
- Released in October 2012
- Latest version: 4.9

TypeScript vs JavaScript



ES 2015+	TS Only
• Let/const • Classes • Static • Getter/Setter • Modules • Arrow Functions • Promises • Template Strings • Async/Await • Iterator • …	• Interfaces • Types • Enums • Generics

Benefits

- Typescript provides an optional type system for Javascript
- **Why you should use types ?**
  - Catches error at compile-time
  - Support Object Oriented Programming
  - Does not need language specific runtime
  - Better live documentation

Static-type checking

- Typescript perform type checking at compile time
- Early detection of type errors
- Consequential gain in the adaptability of storage use

```
var value = 5;
value = "hello";
// error: Type '"hello"' is not assignable to type 'number'.
```

Non-Exception Failures

- Typescript helps to detect errors that are **not exceptions**
- Runtime errors becomes compile-time errors

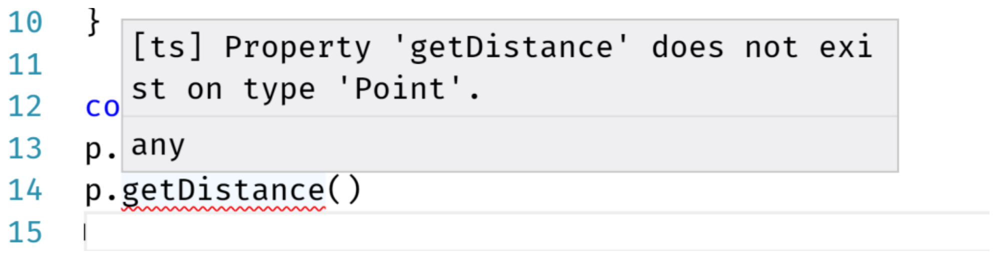
```
const user = {
    name: "Daniel",
    age: 26,
  };

user.location;

Property 'location' does not exist on type '{ name: string; age: number; }'
```

Tools support

- Using Typescript enable IDE assistance
- Intellisense, quick fixes, errors highlighting…etc


```
}
                                                                                
co
p.
p.getDistance()

```
[ts] Property 'getDistance' does not exist on type 'Point'.

any

How to compile ?

- Typescript comes with a Compiler

```
npm install -g typescript
```

- Run tsc to compile ts into js

```
tsc file.ts
```

$$\begin{tikzcd} \text{Sample.ts} \arrow[r, "\textit{tsc Sample.ts}"] & \text{Sample.js} \end{tikzcd}$$

Usage

- Create a .ts file

```
function add(...numbers: Array<number>): number{
  return numbers.reduce((prev, current) => prev + current, 0);
}
const result = add(1, 2);
console.log(result)
```

- Compile into a .js file

```
function add() {
  var numbers = // ...
  return numbers.reduce(function (prev, b) { return prev + b; }, 0);
}
var result = add(1, 2);
console.log(result);
```

- Add .js file into an HTML file to run on client-side

```
<head>
  <script src="scripts/basics.js"></script>
</head>
```

- **OR** run on server-side using NodeJS

Compiler configuration

- TSC is customizable though
  - CLI flags
  - Configuration file : tsconfig.json

- To create a default configuration file

```
tsc --init
```

- Specified options like
  - Supporting ES5 or older browsers
  - Include/Exclude file
  - Path aliases
  - Enable/Disable strict type-checking
  - …

Example (tsconfig.json)

```
{
  "compilerOptions": {
    /* Basic Options */
    "target": "es5", /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017', or 'ESNEXT'. */
    "module": "commonjs", /* Module code generation: 'none', commonjs', 'amd', 'system', 'umd', 'es2015', or 'ESNext'. */
    "lib": ["dom", "dom.iterable", "es2015"], /* Specify library files to be included in the compilation: */
    "allowJs": false, /* Allow javascript files to be compiled. */
    // ...

    /* Strict Type-Checking Options */
    "strict": true /* Enable all strict type-checking options. */
    // ...

    /* Additional Checks */
    // ...

    /* Module Resolution Options */
    // ...

    /* Source Map Options */
    // ...

    /* Experimental Options */
    // "experimentalDecorators": true, /* Enables experimental support for ES7 decorators. */
    // "emitDecoratorMetadata": true, /* Enables experimental support for emitting type metadata for decorators. */
  }
}
```