Intro to TypeScript

Steve Hartzog

TypeScript: What is it?

Introduced in October 2012 by Ander’s Hejlsberg

Free & Open source language[1]

It is a “superset of JavaScript”[2]

Build Plugins for maven, gradle[3], grunt and gulp

Rapidly becoming an Industry Standard (Angular)

1. https://en.wikipedia.org/wiki/TypeScript
2. https://en.wikipedia.org/wiki/TypeScript#Compatibility_with_JavaScript
3. https://github.com/ppedregal/typescript-maven-plugin, https://github.com/sothmann/typescript-gradle-plugin

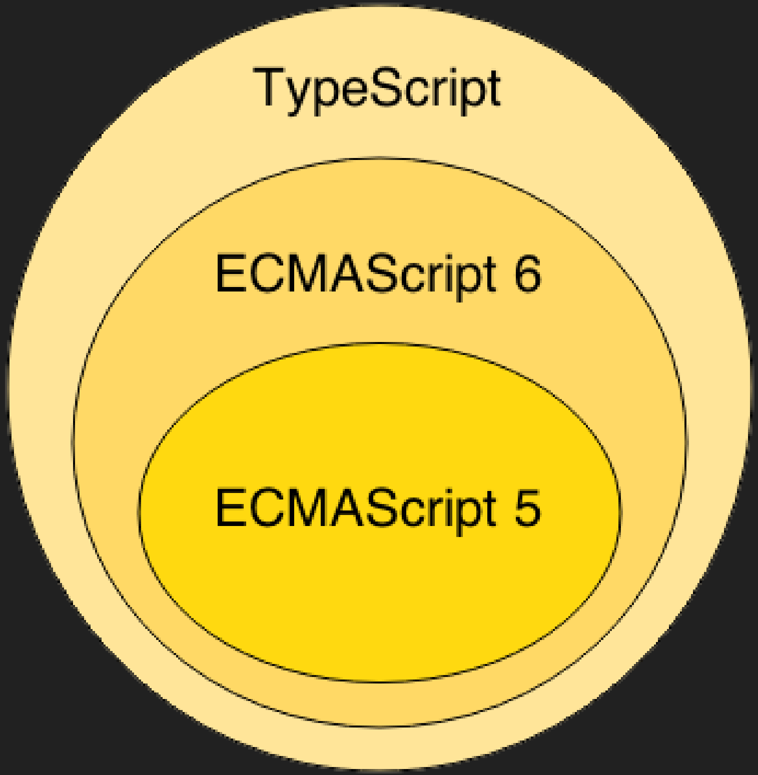
What is a superset?

Yep. TypeScript includes:

TypeScript

ES2015

ES5



It’s ES2015!

So, it has everything ES2015 has from Justin’s presentation:

- Arrow Functions
- Spread
- Block scoping (let & const)
- Classes with constructors, get/set
- Enhanced Object Literals
- Template Strings
- Destructuring

And a few ES2015 things not in Justin’s Talk:

- ECMAScript Modules
- Method properties (func shorthand in JSON)
- Base Class Access via super()
- Typed Arrays

ALL of ES2015, PLUS:

- Annotations
- **Decorators** (ES7)
- Type inference
- Generics
- **Compile-time type checking**
- **Enums!**

- **Namespaces**
- Extended Class Syntax
  - **Public** / **Private**
  - **Interfaces**
  - Optional Properties

Why are types good?

```
function doSomething (smidge) {
  if (smidge instanceOf repo
      && smidge.myProp) {
       var a = smidge.myProp;
  }
  //...
}
```

```
doSomething (smidge:repo) {
  var a = smidge.myProp;
  ...
}

// much simpler
```

TypeScript: JavaScript at Scale

Checks code ***on the fly*** at design time

- tsc watcher for design time analysis
- Compatible with most editors
- Allows you to catch bugs prior to JS build

Transpiles to ES5

- Plugs right in where babel is used
- ES5 output is ~<u>**HALF**</u> the size of Babel’s transpilation of ES2015

Design Time Analysis and Compiler Errors

1. Finding bugs **earlier** in your development process
2. F**ix a bug** in your logic **at design time** rather than fixing the bug at run time.
    a. A compile-time bug will fail every time you run the compiler, but a runtime bug can hide.
3. Runtime bugs can slip under a crack in your logic and lurk there (sometimes for months) until discovered.

TypeScript: Types

```
var num: number = 123;
function identity(num: number): number {
    return num;
}
```

TypeScript: Decorators

```
@myClassDecorator
class identity(num: number): number {
  @myPropertyDecorator
  var num: number = 123;

  @log
  public f(@myParamDecorator x:
number) {
  }
}
```

```
function log(target: Function,
  key: string, value: any): T {
  return {
    value: function(...args: any[]) {
      var a = args.map(a.map(
        a => JSON.stringify(a)).join();
      var result = ;
      var r = ;
      console.log(`Call: ${key}(${a}) => ${r}`);
      return result;
    }
  };
}
```

TypeScript: Namespaces

```
namespace utility {
  export function log(msg) { console.log(msg); }
  export function error(msg) { console.error(msg); }
}
```

TypeScript: Interfaces

```
interface Point {
    x: number; y: number;
}
declare var myPoint: Point;
```

TypeScript: Enums

TypeScript: Generics

```
function identity<T>(arg: T): T {
    return arg;
}
```