AngularTS

Building apps with AngularJS & TypeScript

Agenda

TypeScript: Whats ‘n’ Whys

Integrating with JavaScript libraries

AngularJS: Whats ‘n’ Whys

Reference Project

In Practice

Dos ‘n’ Don’ts

Conclusion

Agenda

TypeScript: Whats ‘n’ Whys

Integrating with JavaScript libraries

AngularJS: Whats ‘n’ Whys

Reference Project

In Practice

Dos ‘n’ Don’ts

Conclusion

TypeScript: Whats ‘n’ Whys

Whats

- High level language that compiles to JavaScript
- Object Oriented (or not)
- Strongly typed (or not)
- MS backed
- Open-source
- Focuses on tightly integrate existing JS libs
- Closures, Generics, Annotations and Interfaces
- String interpolation
- ECMA6 now, ECMA7 soon

TypeScript: Whats ‘n’ Whys

Whys

- We can go full-typed!
- All JS is valid TS
- We get to use newer ES features before browsers actually support it
- Compile-time exists!
- Way to bypass what stinks in JS
- We get to keep using our favorite JS libs
- Used and endorsed by Angular 2
- Arrow functions
- Porting from JavaScript is easy

TypeScript: Whats ‘n’ Whys -> Types

```
// There are 3 basic types in TypeScript
var isDone: boolean = false;
var lines: number = 42;
var name: string = "Anders";

// When it's impossible to know, there is the "Any" type
var notSure: any = 4;
notSure = "maybe a string instead";
notSure = false; // okay, definitely a boolean

// For collections, there are typed arrays and generic arrays
var list: number[] = [1, 2, 3];
// Alternatively, using the generic array type
var list: Array<number> = [1, 2, 3];

// For enumerations:
enum Color {Red, Green, Blue}
var c: Color = Color.Green;

// Lastly, "void" is used in the special case of a function returning nothing
function bigHorribleAlert(): void {
   alert("I'm a little annoying box!");
}
```

TypeScript: Whats 'n' Whys -> Functions

```
// Functions are first class citizens, support the lambda "fat arrow" syntax and
// use type inference

// The following are equivalent, the same signature will be infered by the
// compiler, and same JavaScript will be emitted
var f1 = function(i: number): number { return i * i; }
// Return type inferred
var f2 = function(i: number) { return i * i; }
var f3 = (i: number): number => { return i * i; }
// Return type inferred
var f4 = (i: number) => { return i * i; }
// Return type inferred, one-liner means no return keyword needed
var f5 = (i: number) =>  i * i;
```

TypeScript: Whats ‘n’ Whys -> Interfaces

```
// Interfaces are structural, anything that has the properties is compliant with
// the interface
interface Person {
   name: string;
   // Optional properties, marked with a "?"
   age?: number;
   // And of course functions
   move(): void;
}

// Object that implements the "Person" interface
// Can be treated as a Person since it has the name and move properties
var p: Person = { name: "Bobby", move: () => {} };
// Objects that have the optional property:
var validPerson: Person = { name: "Bobby", age: 42, move: () => {} };
// Is not a person because age is not a number
var invalidPerson: Person = { name: "Bobby", age: true };
```

TypeScript: Whats 'n' Whys -> Classes

```
// Classes - members are public by default
class Point {
   // Properties
   x: number;

   // Constructor - the public/private keywords in this context will generate
   // the boiler plate code for the property and the initialization in the
   // constructor.
   // In this example, "y" will be defined just like "x" is, but with less code
   // Default values are also supported

   constructor(x: number, public y: number = 0) {
       this.x = x;
   }

   // Functions
   dist() { return Math.sqrt(this.x * this.x + this.y * this.y); }

   // Static members
   static origin = new Point(0, 0);
}
```

TypeScript: Whats ‘n’ Whys

- Inheritance
- Generics
- String interpolation
- Modules

Agenda

TypeScript: Whats ‘n’ Whys

Integrating with JavaScript libraries

AngularJS: Whats ‘n’ Whys

Reference Project

In Practice

Dos ‘n’ Don’ts

Conclusion

Integrating with JavaScript libraries

- Declaration files (.d.ts) are used to describe libraries
- Typings: TypeScript Definition Manager
    - Has a command line utility that works much like NPM or Bower
        - npm install typings --global
        - typings install angular --global --save
    - Use /// <reference path=”/typings/index.d.ts”> to use the definitions

Agenda

TypeScript: Whats ‘n’ Whys

Integrating with JavaScript libraries

AngularJS: Whats ‘n’ Whys

Reference Project

In Practice

Dos ‘n’ Don’ts

Conclusion

AngularJS: Whats ‘n’ Whys

Whats

- MV* Framework for the web
- Data binding
- Dependency Injection
- Promises
- Directives
- Reusable components
- Localization
- Form validation

AngularJS: Whats ‘n’ Whys

Whys

- Single Page Web Applications are hard to implement; Angular makes it easy
- JS is hell; Angular helps keeping it in check
- Abstracts data-to-presentation conversion
- Enables componentization
- Facilitates separation between presentation and business logic

But HOW!?

Services & Factories

Controllers

Routing

Directives

Dependency Injection

Data Binding

Agenda

TypeScript: Whats ‘n’ Whys

Integrating with JavaScript libraries

AngularJS: Whats ‘n’ Whys

Reference Project

In Practice

Dos ‘n’ Don’ts

Conclusion

Reference Project

Agenda

In Practice

Agenda

TypeScript: Whats ‘n’ Whys

Integrating with JavaScript libraries

AngularJS: Whats ‘n’ Whys

Reference Project

In Practice

Dos ‘n’ Don’ts

Conclusion

Dos ‘n’ Don’ts

Do

- Use ‘controllerAs’ syntax
- Use $inject
- Use Angular services more broadly
- Componentize your application
- Use Angular’s own services ($timeout, $interval, $location)
- Use promises instead of callbacks

Dos ‘n’ Don’ts

Don’t

- Overuse events
- Overuse $watch
- Use $rootScope as a global repository
- Manipulate DOM in controllers
- Use ng-init

Agenda

TypeScript: Whats ‘n’ Whys

Integrating with JavaScript libraries

AngularJS: Whats ‘n’ Whys

Reference Project

In Practice

Dos ‘n’ Don’ts

Conclusion

Conclusion

AngularTS

Building apps with
AngularJS & TypeScript

Bonus: Helpful Links

https://learnxinyminutes.com/docs/typescript/

https://www.typescriptlang.org/docs/handbook/writing-declaration-files.html