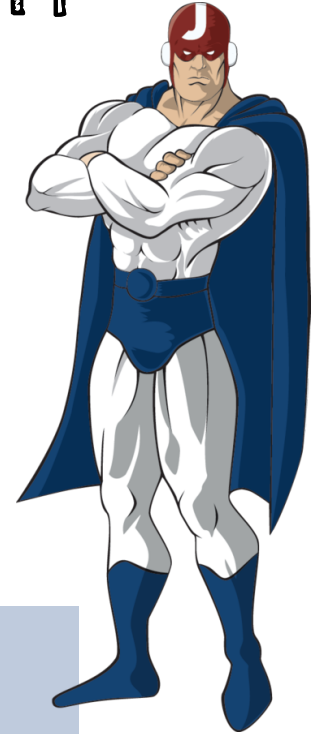


Getting started with Angular 2 hands-on workshop

by Gerard Sans (@gerardsans)

Laptops shiny and ready by **13:30**~ish
Instructions bit.ly/jfokus-ng2-lab



Agenda

13:30 Welcome

Instructions and setup

ES6/TypeScript

Practice, Q&A

Bootstrap

Practice, Q&A

Components and Services

Practice, Q&A

Component Router

Practice, Q&A

17:00 Wrapping up

A little about me



- Angular GDE (Google Developer Expert)
- JavaScript and Angular fanboy ^_^
- [Blog](#) | [@gerardsans](#) | github.com/gsans
- Community activist
 - Looking for Angular Meetup organisers
- Meetup Group. Follow us!
 - [AngularJS Labs \(@angularjs_labs\)](#)



ng-nl 2016 - February 18th

@ngnlconf

Talk: Introduction to RxJS 5



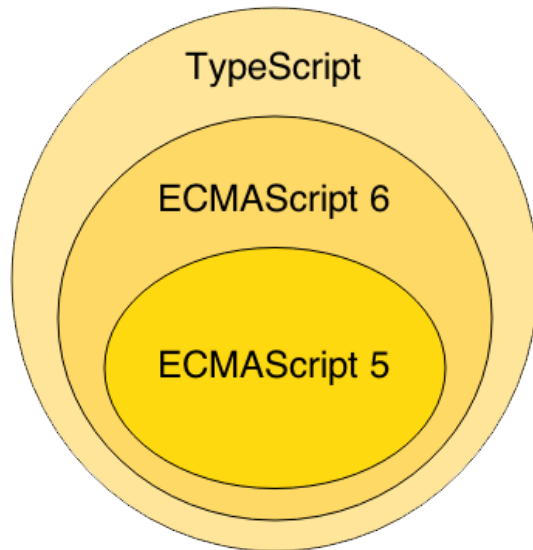
ng-conf - May 4-6th
@ngconf



[AngularConnect](#) - Sept 27-28th

[@AngularConnect](#)

ES6 / TypeScript

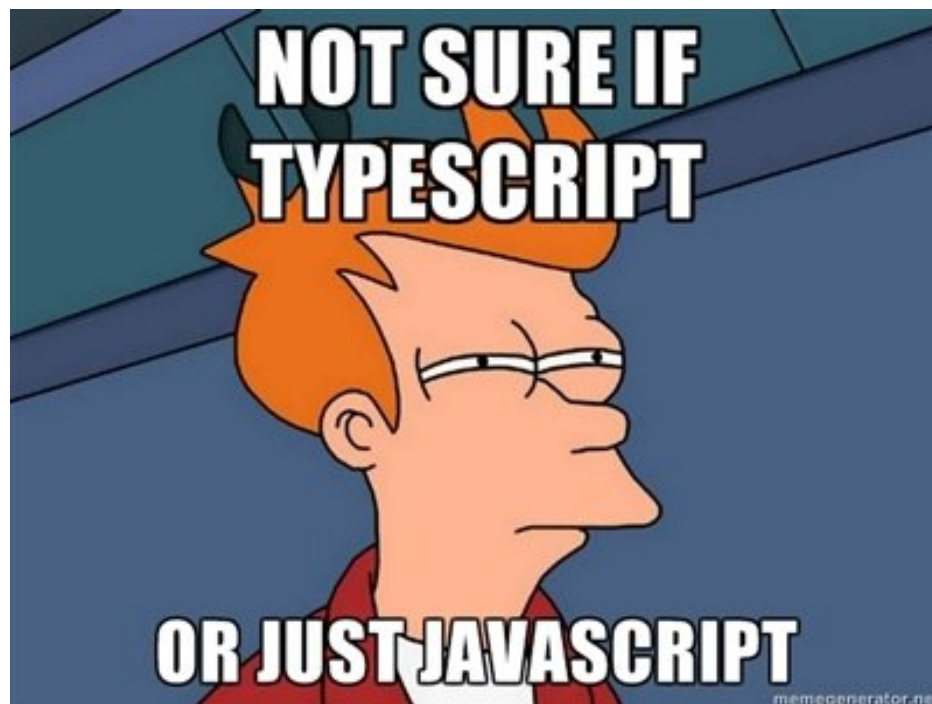


ES6 (ES2015)

- Classes, modules, arrow functions

TypeScript

- Types, annotations
- Better editor support
- Angular Team uses it ^_^



ES6: Overview

- Arrow functions and let keyword; Block scopes
- Classes and inheritance; Default parameters
- Destructured assignment
- Generators; Iterators; Maps
- Promises; Rest parameters; Sets
- Spread operator; Template Literals

Resources

- [jsbin](#) (ES6/Babel)
- [Just another introduction to ES6](#)

ES6 modules

ES6

```
// ES6 deconstructing
```

```
import { member } from "module-name";
import { reallyLongMemberName as alias } from "module-name";
import { member1, member2, [...] } from "module-name";
import defaultMember from "module-name";
```

```
//----- lib1.js -----
```

```
export function add(x, y) {
  return x + y;
}
export function square(x) {
  return x * x;
}
```

```
//----- lib2.js -----
```

```
export default function square(x) {
  return x * x;
}
```

```
//----- main.js -----
```

```
import { add as sum, square } from 'lib1';
import mySquare from 'lib2';
```

```
sum(1,2); // 3
mySquare(2,2); // 4
```

ES6: Classes

```
class Person {  
  //class constructor  
  constructor(name, age) {  
    this.name = name;  
    this.age = age;  
  }  
  //class method  
  sayName() {  
    console.log("I'm "+this.name);  
  }  
}  
  
let p = new Person("John", 25);  
p.sayName(); // I'm John
```

ES6

```
function Person(name, age) {  
  //class constructor  
  this.name = name;  
  this.age = age;  
}  
Person.prototype = {  
  //class method  
  sayName: function () {  
    console.log("I'm "+this.name);  
  }  
};  
  
var p = new Person("John", 25);  
p.sayName(); // I'm John
```

ES6: Destructuring

```
let [one, two] = [1, 2];  
let {three, four} = {three: 3, four: 4};  
  
console.log(one, two, three, four); // 1 2 3 4
```

ES6

```
var a = [1, 2];  
var one = a[0];  
var two = a[1];  
  
var b = {three: 3, four: 4};  
var three = b.three;  
var four = b.four;  
  
console.log(one, two, three, four); // 1 2 3 4
```

ES6: Template literals

ES6

```
//multi-line
let text = `string text line 1
string text line 2`;

//interpolation
let p = {name: "John", credits: 20};
let tpl = `Name: ${p.name}. Credits: ${p.credits + 10}`;
// Name: John. Credits: 30
```

```
//multi-line
var text = "string text line 1\n" +
  "string text line 2";

//interpolation
var p = {name: "John", credits: 20};
var tpl = "Name: " + p.name + ". Credits: " + (p.credits + 10);
// Name: John. Credits: 30
```

ES6: Arrow functions

```
let square = x => x * x;  
let add = (a, b) => a + b;  
let pi = () => 3.1415;
```

ES6

```
var square = function(x) { return x * x; };  
var add = function(a, b) { return a + b; };  
var pi = function() { return 3.1415; };
```

ES6: Default parameters

```
function sayMsg(msg='This is a default message.') {  
  console.log(msg);  
}  
sayMsg();  
sayMsg('This is a different message!');
```

ES6

```
function sayMsg(msg){  
  var msg = msg || 'This is a default message.';  
  console.log(msg);  
}  
sayMsg();  
sayMsg('This is a different message!');
```

Practice

www.es6fiddle.net

TypeScript +1.6

- Basic types: boolean, string, number, any
- Type annotations and compile-time type checking
- Type inference, Interfaces, Enumerated type, Mixins, Generics, Tuple

Resources

- [Handbook](#)

TS: Basic types

```
var isDone: boolean = false;           // boolean      TS
var height: number = 6;                 // number
var name: string = "bob";               // string

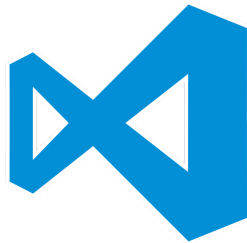
var list: number[] = [1, 2, 3];         // arrays
var list: Array<number> = [1, 2, 3];    // generics

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

var notSure: any = 4;                   // if not defined
var list: any[] = [1, true, "free"];

function warnUser(): void {             // return type
    alert('WAT?');
}
```

TS: IDEs



Play time

TypeScript playground

Angular 2 Basic App

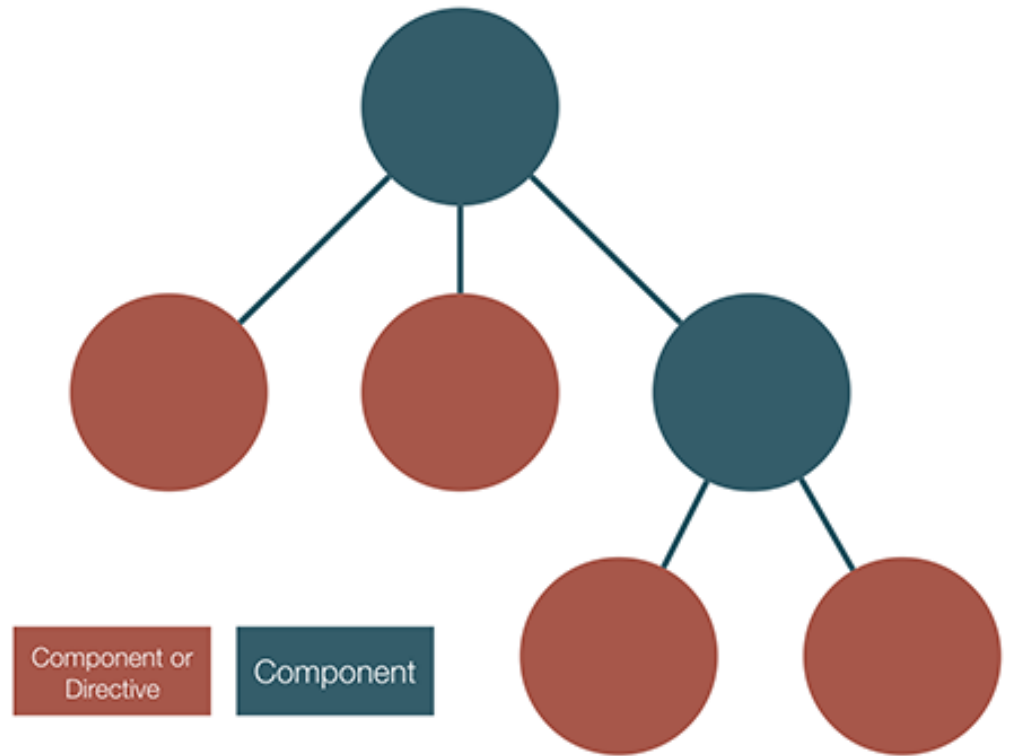
We are going to build a basic Application covering:

- Bootstrapping
- Components, Services
- Template syntax
- Component router

Bootstrapping

- Angular Application instantiation
- Requires root component and global dependencies
- Root Component (App)
- Global Dependencies
 - Classes. Eg: Router, Forms, Http
 - Global Values
 - Vendor dependencies

Component Tree



source: [blog](#)

Component

```
import {Component} from 'angular2/core';
import {FORM_DIRECTIVES} from 'angular2/common';
import {UserService} from '../services/usersService';

@Component({
  selector:      'home', // <home></home>
  styles:        [`h1 { color: red }`],
  template:      `<h1>Home</h1>`,
  directives:    [FORM_DIRECTIVES],
  providers:     [userService]
})
export class Home { ... }
```


Template Syntax

Syntax	Binding type
<code><h1>{{title}}</h1></code>	Interpolation
<code><input [value]="firstName"></code>	Property
<code><li [class.active]="isActive"></code>	Class
<code><div [style.width.px]="mySize"></code>	Style
<code><button (click)="onClick(\$event)"></code>	Event
<code>[(ngModel)]="data.value"</code>	Two-way

Component Router

- Features:
 - Multiple views, nested views, emphasis on components and re-usability.
- Usage
 - Include router dependencies
 - Define routes using @RouteConfig
 - Templates: router-outlet, routerLink

