Contents

1 Are you Angular 2/Too/Tu Brute? 3

What, why, how - BarCampPhilly 2015 . . . 3
Preface . . . 3
Warning . . . 3
Angular2 is a complete rewrite . . . 3
Angular1 typical component . . . 3
Angular2 component . . . 4
Angular2 Component - ES5 . . . 4
Angular1 Services (one of many ways) . . . 5
Angular1 Using a service . . . 5
A model object in TypeScript . . . 5
TypeScript = Type Safety . . . 6
Static methods? Yep . . . 6
Angular2 services . . . 6
A sample method in TaskService . . . 7
For Observables (RxJavaScript) and Angular - READ THIS . . . 7
Using from a component . . . 8
Key features / benefits of Angular2 . . . 8
Why Angular2 . . . 9
Why TypeScript? . . . 9
Things you'll need to learn . . . 9
Tools to play . . . 9
Reading and Videos . . . 10
Demo - Tour of features . . . 10

Interesting sub-projects to watch 10
What isn't done yet? 10
Bottom line - PLAY 11

Chapter 1

Are you Angular 2/Too/Tu Brute?

What, why, how - BarCampPhilly 2015

Ken Rimple Chariot Solutions @krimple

Preface

Angular2 is NOT ready for production yet

Witness the alpha release chain (#45) and still breaking changes

But you need to know it

Warning

If you want to learn Angular2 you're going to have to fetch the github project and play around for now...

Angular2 is a complete rewrite

^^ me ^^

Angular1 typical component

Controller:

```
angular.module('foo')
.controller('MessageForYouSir', function($scope) {
  $scope.message = 'Look out for that rock!';
});
```

View mount of controller

```
<div ng-controller="MessageForYouSir">
  {{ message }}
</div>
```

Route mount of controller

```
$routeProvider.when('/msg', {
   controller: 'MessageForYouSir',
   templateUrl: 'messageTemplate.html'
});
```

Angular2 component

```
import { Component } from 'angular2/angular2';

@Component({
  selector: 'message-widget',
  template: `{{ message }}`,
  inputs: ['message']
})
export class MessageForYouSir {
    constructor(public message: string) {
      // do startup things here
    }
}
```

- That's ES6 and TypeScript, folks
- You can use ES5, too...

Angular2 Component - ES5

```
var MessageForYouSir = ng.
Component({
  selector: 'message-widget',
```

```
})
.View({
  template: '{{ message }}'
})
.Class({
  constructor: function(message) {
    this.message = message;
  }
});
```

Angular1 Services (one of many ways)

```
angular.module('foo')
.service('getTasks', function($http) {
  this.getTasks = function() {
    return $http.get(); // several ways to do this...
  }
});
```

- Dependency injection in the creating function
- Services are singletons

Angular1 Using a service

Using it

```
angular.module('foo')
.controller('TaskListCtrl', function($scope, taskService) {
      taskService.getTasks().then(success, error);
});
function success(payload) {
  $scope.tasks= payload.data;
}
function error(error) {
  $log.error(error.data, error.status);
}
```

A model object in TypeScript

```
export class TaskModel {
    description: string;
```

```
    priority: number;
    dueDate: Date;
    complete: boolean;
    completedDate: Date;
    constructor(description: string,
                priority: number,
                dueDate: Date, complete: boolean) {
        this.description = description;
        this.priority = priority;
        this.dueDate = dueDate;
        this.complete = complete;
    }
}
```

TypeScript = Type Safety

- And navigation of classes, methods, inheritence, etc...
- Most of this is part of ES6
- TypeScript adds types, annotations

Static methods? Yep

```
// parse from Strings in JSON, don't sully constructor
static fromJsonData(description: string,
                    priority: string, dueDate: string,
                    complete: string) {
  let npriority = Number.parseInt(priority);
  let ddueDate = new Date(dueDate);
  // blech Ken, really?
  let bcomplete = new Boolean(complete).valueOf();
  let taskModel =
  new TaskModel(description, npriority,
    ddueDate, bcomplete)
  return taskModel;
}
```

Angular2 services

```
@Injectable()
export class TaskService {
```

```
    public tasks:Array<TaskModel>;

    constructor(private http:Http) {
        this.tasks = [];
        this.fetchTasks();
    }
}
```

- `@Injectable()` - Allow other objects to inject this service into them
- `private http: Http` as a parameter - establishes the invisibility of the data element (cannot access as a property of taskService. . .)

A sample method in TaskService

```
fetchTasks() {
  this.http.get('/todo/data/todos.data.json')
    .map((res) => { return res.json(); })
    .map((tasks) => {
      let result:Array<TaskModel> = [];

      tasks.forEach((task) => {
        result.push(
          TaskModel.fromJsonData(
            task.description, ...)
        );
    });
}
```

- an `Observable` is emitted from `http.get`
- You may provide functional processing methods to the Observable
- We use `map` here to transform the input to a single output

For Observables (RxJavaScript) and Angular - READ THIS

Rob Wormald's fantastic talk - Everything is a Stream - http://slides.com/robwormald/everything-is-a-stream#/

Using from a component

```
@Component({
    selector: 'task-list',
    bindings: [TaskService]
})
@View({
    templateUrl: '/todo/task-list.html',
    directives: [CORE_DIRECTIVES, TaskEntry]
})
export class TaskList {
  tasks: Array<TaskModel> = [];

  constructor(private taskService: TaskService) {
    this.tasks = taskService.tasks;
  }
}
```

- `bindings` - What services can be injected into us
- `private taskService: TaskService` - injecting an `Injectable()`

Key features / benefits of Angular2

- Typescript
- No more $scopes or controllers, custom ng-xxx directives for events, etc...
- No digest cycle - single top-down DAG updates
- Two-way binding not the default
- ES6 Modules
- Components are EASY
- Shadow DOM for component updates
- New Router (getting there)
- Decoupled rendering engine

Why Angular2

- Quicker
- More responsive
- Modular (use what you want only)

Why TypeScript?

- Targets ES6 language features
- Adds typings to JavaScript
- Development team wrote Angular2 in TypeScript and transpiles to ES5, Dart, ES6
- You can use lightweight editors with typings files (Vim, Sublime) to provide code fill-in and navigation

Things you'll need to learn

- Creating/using npm packages, npm install, etc (that's where the build tools are)
- Basic TypeScript language features
- TypeScript tooling (the npm TypeScript transpiler - tsc/tsconfig.json)
- The `tsun` package - a shell for TypeScript for experimenting
- ES6 language features - module system, constructor functions and classes, arrow functions, etc... - upgrade to Node.JS > 4.1 to get this natively in the node shell
- The Microsoft Reactive Extensions project for JavaScript - observables etc...

Tools to play

- Gitter (http://gitter.im/angular/angular) - great resource where the developers hang out and answer questions from all comers (A+!)
- Plunker (http://plnkr.co) - now provides ES6, ES5, TypeScript starters for Angular2 applications (HUGE)
- The GitHub repo for Angular2 (http://github.com/angular/angular)

Reading and Videos

- Lots of blog entries, look at Chariot's entries at (http://chariotsolutions.com/tags/angular2/)
- Egghead - (http://egghead.io) - lots of Angular tutorials, Angular2 content started up (subscription but cheap)
- Angular2 book from Ari Lerner (http://ng-book/2/) - he's working real-time to catch up to the github project and developers and it's a great intro

Demo - Tour of features

- Plunker starters and my TypeScript Angular seed
- Components and Bootstrap
- Services
- The http packages and Observables
- The router, basics

Interesting sub-projects to watch

- Benchmark suite - BenchPress - baked into Angular for all testing for metrics, you can use too
- Web Workers for mobile applications - puts the DOM updates on a Worker to speed up responsiveness
- Angular + NativeScript for potential fast binaries for mobile?
- Ionic2 is moving forward with thier Angular2 version

What isn't done yet?

- Router isn't finished with all cases
- Testing framework seems still not ready for TypeScript developers - but getting closer to reality - tooling needed
- Real documentation is far from complete - they are working on dev guides, documentation
- Some tools are being reworked - they recently dropped the `tsd` tool and DefinitelyTyped but are now talking about another one (egads)

Bottom line - PLAY

- Try using Angular2 and giving feedback to the team via gitter, GitHub issues, etc...
- Get ahold of Ari Lerner's book and exercises and work through them
- Play around in Plunker
- DO NOT use it in production yet!!! But we're closer! (Early 2016?)