

Angular 2



Language Choices

Angular 1's Impact

Comparing Concepts from Angular 1 to 2

Resources



Language Choices



Coding Angular 2

ES5

ES6/ES2015+

TypeScript

Dart



Coding Angular 2

ES5

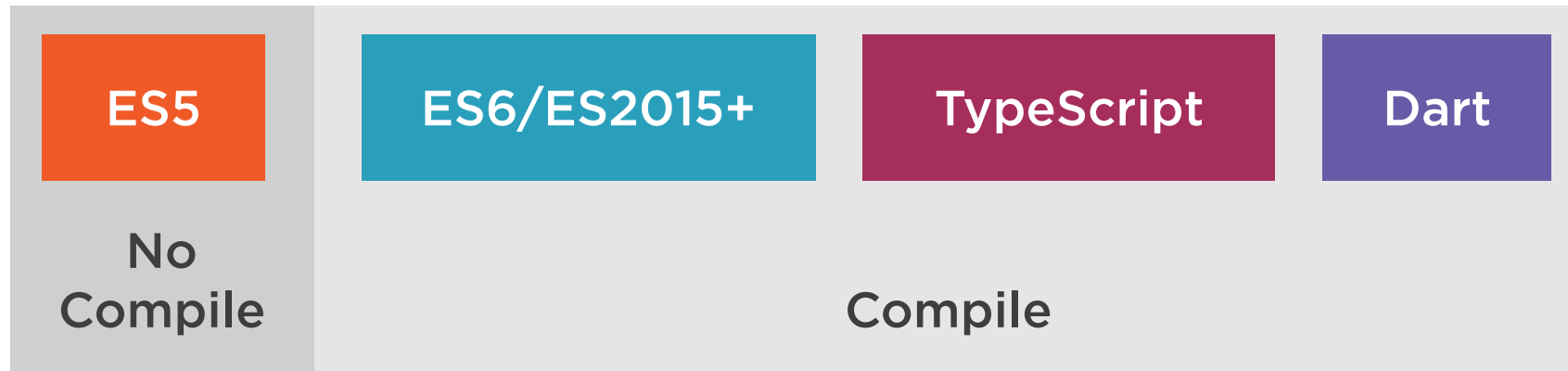
ES6/ES2015+

TypeScript

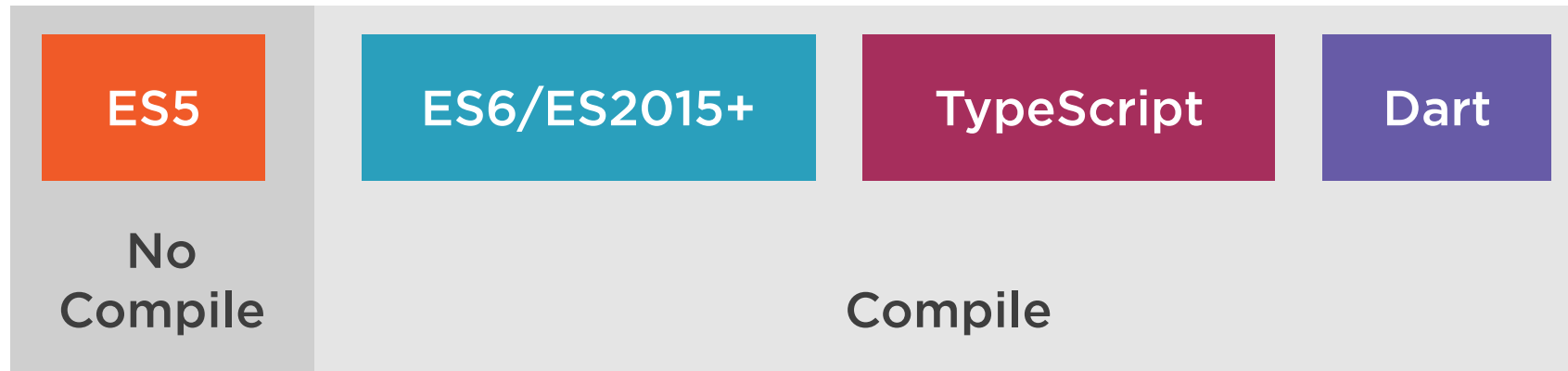
Dart



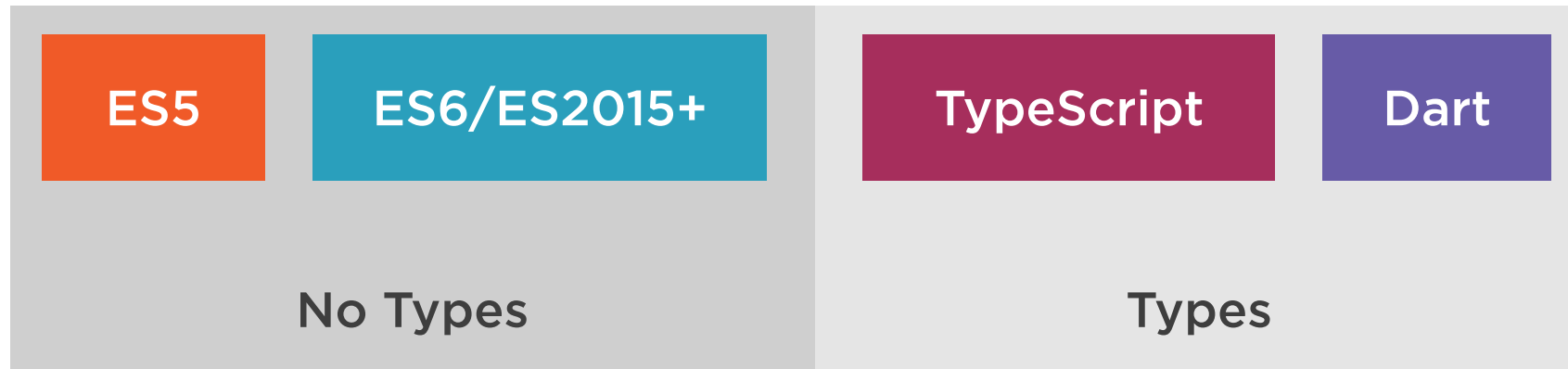
Coding Angular 2



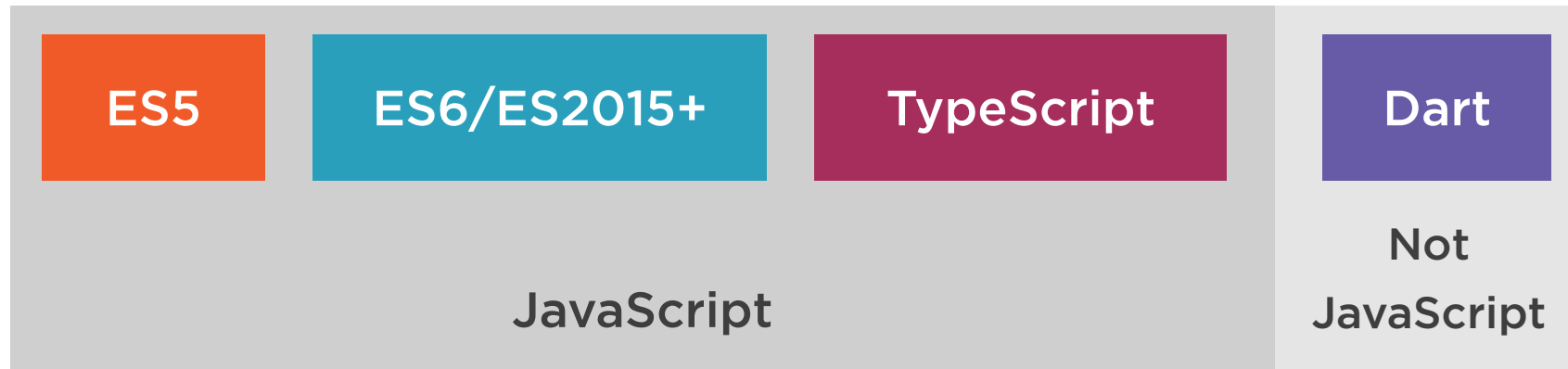
Coding Angular 2



Coding Angular 2



Coding Angular 2



Coding Angular 2

ES5

ES6/ES2015+

TypeScript

Dart



Coding Angular 2



Angular 1's Impact





Angular 1 is widely used
Huge ecosystem



1.1 Million Developers

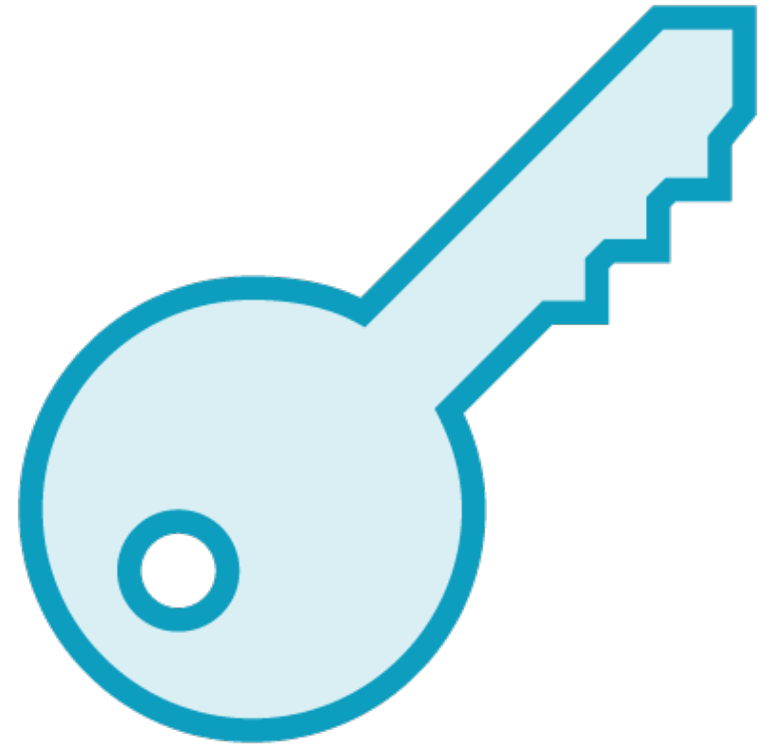


Leaning on Your Angular 1 Skills



Angular 1 to Angular 2

7 Key Comparisons



1

Angular Modules

Angular 1

```
(function () {  
  angular  
    .module('app', []);  
})();
```

Dependencies

`<html ng-app="app">`

Root module

Angular 2

```
import { NgModule } from '@angular/core';  
import { BrowserModule } from '@angular/platform-browser';  
  
@NgModule({  
  imports: [BrowserModule],  
  declarations: [ ... ],  
  bootstrap: [ ... ]  
})  
export class AppModule { }
```

Dependencies

Root module



2

Controllers to Components



Component

Angular 1

```
<body ng-controller="StoryController as vm">
  <h3>{{vm.story.name}}</h3>
  <h3 ng-bind="vm.story.name"></h3>
</body>

(function () {
  angular
    .module('app')
    .controller('StoryController', StoryController);

  function StoryController() {
    var vm = this;
    vm.story = { id: 100, name: 'The Force Awakens' };
  }
})();
```

Angular 2

```
<my-story></my-story>

import { Component } from '@angular/core';

@Component({
  selector: 'my-story',
  template: '<h3>{{story.name}}</h3>'
})
export class StoryComponent {
  story = { id: 100, name: 'The Force Awakens' };
}
```



3

Structural Built-In Directives

Angular 1

```
<ul>  
  <li ng-repeat="vehicle in vm.vehicles">  
    {{vehicle.name}}  
  </li>  
</ul>  
<div ng-if="vm.vehicles.length">  
  <h3>You have {{vm.vehicles.length}} vehicles</h3>  
</div>
```

Angular 2

```
<ul>  
  <li *ngFor="let vehicle of vehicles">  
    {{vehicle.name}}  
  </li>  
</ul>  
<div *ngIf="vehicles.length">  
  <h3>You have {{vehicles.length}} vehicles</h3>  
</div>
```



The diagram illustrates structural directives in Angular 2. It features two orange callout boxes labeled "Structural Directive". One box points to the `*ngFor` attribute in the `` tag, and the other points to the `*ngIf` attribute in the `<div>` tag. A third orange callout box labeled "Local Variable" points to the `{{vehicle.name}}` interpolation within the `` tag. The code snippet is as follows:

```
<ul>
  <li *ngFor="let vehicle of vehicles">
    {{vehicle.name}}
  </li>
</ul>
<div *ngIf="vehicles.length">
  <h3>You have {{vehicles.length}} vehicles</h3>
</div>
```

Structural Directives

Indicated by the * prefix

Changes the structure

4

Data Binding

DOM

Component

Interpolation



One Way Binding



Event Binding



Two Way Binding



Interpolation

Angular 1

```
<h3>{{vm.story.name}}</h3>
```

Context



Angular 2

```
<h3>{{story.name}}</h3>
```



1 Way Binding

Angular 1

```
<h3 ng-bind="vm.story.name"></h3>
```

Angular 2

```
<h3 [innerText]="story.name"></h3>
```

Any HTML Element Property

```
<div [style.color]="color">{{story.name}}</div>
```



Event Binding

Angular 1

```
<button  
  ng-click="vm.log('click')"  
  ng-blur="vm.log('blur')">OK</button>
```

Angular 2

```
<button  
  (click)="log('click')"  
  (blur)="log('blur')">OK</button>
```



2 Way Binding on an <INPUT>

Angular 1

```
<input ng-model="vm.story.name">
```

Angular 2

```
<input [(ngModel)]="story.name">
```

**Football in a
Box**

[()]



5

Removes the Need for Many Directives

Angular 1

```
<div ng-style=
  "vm.story ?
    {visibility: 'visible'}
    : {visibility: 'hidden'}">

  
  <br/>
  <a ng-href="{{vm.link}}">
    {{vm.story}}
  </a>

</div>
```

Angular 2

```
<div [style.visibility]=
  "story ? 'visible' : 'hidden'">

  <img [src]="imagePath">
  <br/>
  <a [href]="link">{{story}}</a>

</div>
```



Angular 2 Template Concepts Remove 40+ Angular 1 Built-In Directives



No Longer Need These Directives Either

Angular 1

```
ng-click="saveVehicle(vehicle)"  
ng-focus="log('focus')"  
ng-blur="log('blur')"  
ng-keyup="checkValue()"
```

Angular 2

```
(click)="saveVehicle(vehicle)"  
(focus)="log('focus')"  
(blur)="log('blur')"  
(keyup)="checkValue()"
```



6

Se

Graphics Team:

Can we replace this with the same video you used in module 5 (services) ?

Angular 1

Factories

Services

Providers

Constants

Values

Angular 2

Class



Creating Services

Angular 1

```
angular
  .module('app')
  .service('VehicleService', VehicleService);

function VehicleService() {
  this.getVehicles = function () {
    return [
      { id: 1, name: 'X-Wing Fighter' },
      { id: 2, name: 'Tie Fighter' },
      { id: 3, name: 'Y-Wing Fighter' }
    ];
  }
}
```

Angular 2

```
import { Injectable } from '@angular/core';
```

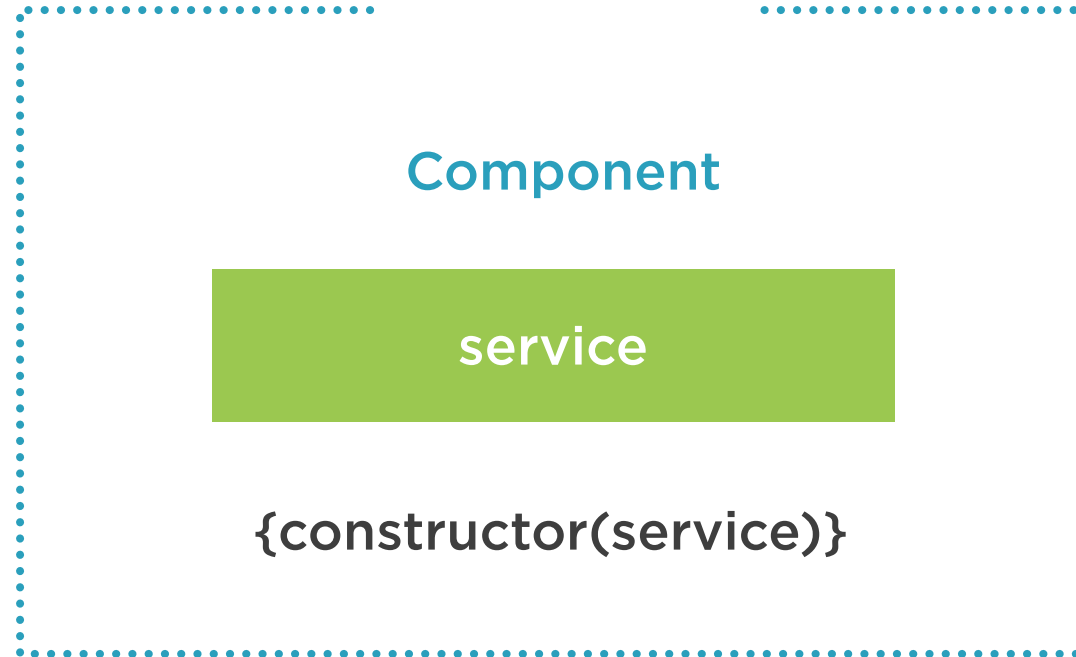
Just a Class

```
@Injectable()
export class VehicleService {
  getVehicles = () => [
    { id: 1, name: 'X-Wing Fighter' },
    { id: 2, name: 'Tie Fighter' },
    { id: 3, name: 'Y-Wing Fighter' }
  ];
}
```



7

Dependency Injection



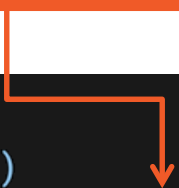
Registering Services with the Injector

Registration

Angular 1

```
angular
  .module('app')
  .service('VehicleService', VehicleService);

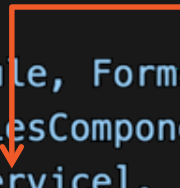
function VehicleService() {
  this.getVehicles = function () {
    return [
      { id: 1, name: 'X-Wing Fighter' },
      { id: 2, name: 'Tie Fighter' },
      { id: 3, name: 'Y-Wing Fighter' }
    ];
  };
}
```



Angular 2

Registration

```
@NgModule({
  imports: [BrowserModule, FormsModule],
  declarations: [VehiclesComponent],
  providers: [VehicleService],
  bootstrap: [VehiclesComponent],
})
export class AppModule { }
```

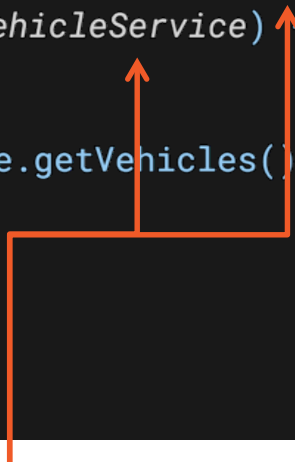


Dependency Injection

Angular 1

```
angular
  .module('app')
  .controller('VehiclesController', VehiclesController);

VehiclesController.$inject = ['VehicleService'];
function VehiclesController(VehicleService){
  var vm = this;
  vm.title = 'Services';
  vm.vehicles = VehicleService.getVehicles();
}
```



Injection

Angular 2

```
@Component({
  moduleId: module.id,
  selector: 'my-vehicles',
  templateUrl: 'vehicles.component.html',
})
export class VehiclesComponent {
  vehicles = this.vehicleService.getVehicles();

  constructor(private vehicleService: VehicleService) { }
}
```



Injection





Removal of
Many
Directives

Angular
Modules

Structural
Directives

Components

Services

Data Binding

Dependency
Injection

Angular 1 to Angular 2

7 Key Comparisons



Many Angular 1 Concepts Translate

Services

Filters / Pipes

Routing

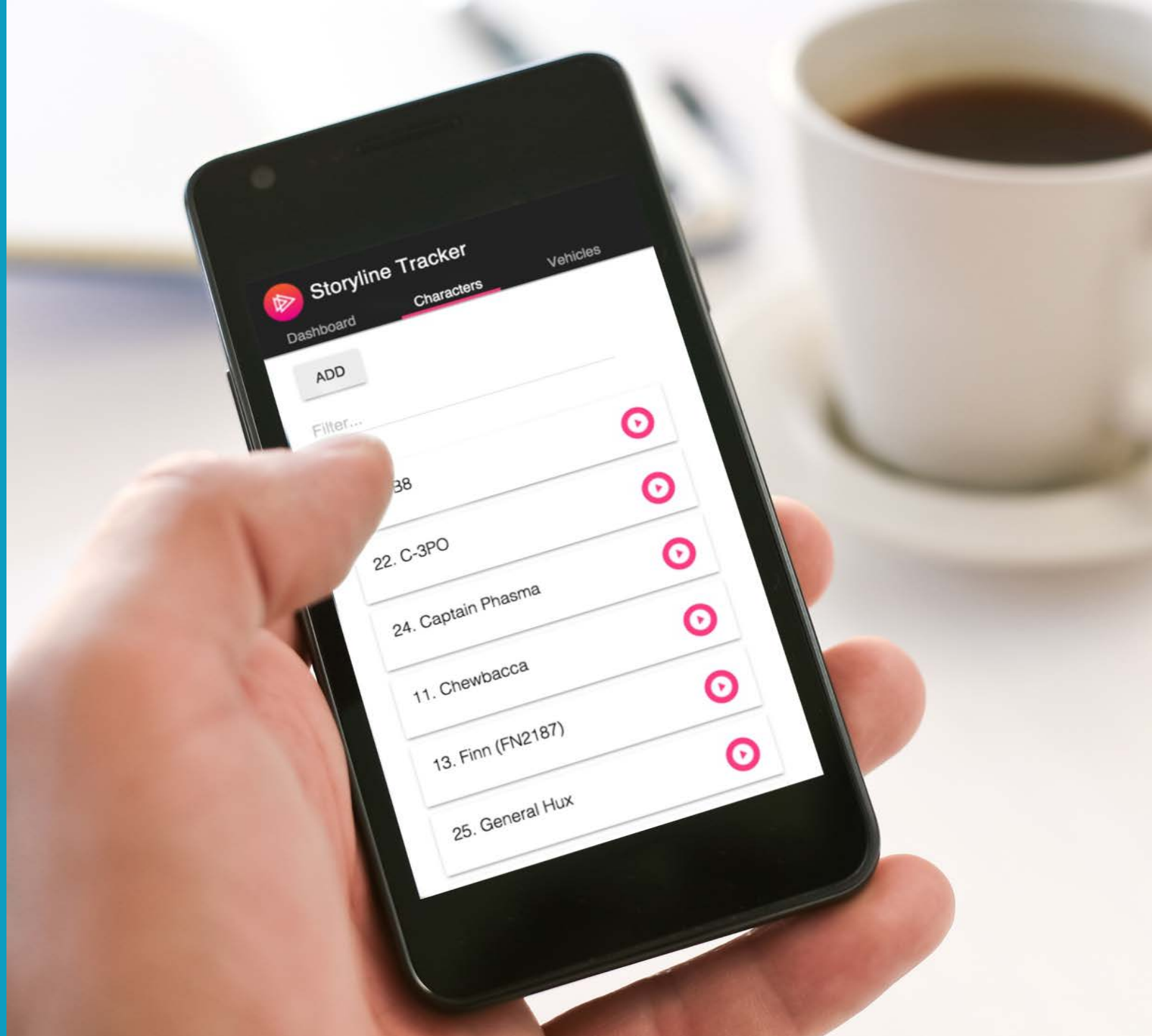
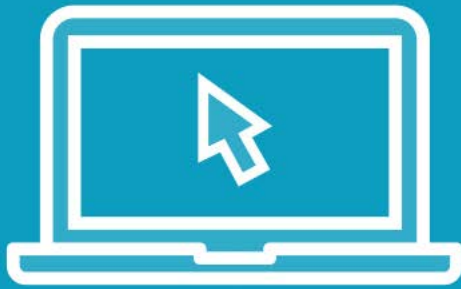
HTTP

Events

Promises



Demo



Angular 2 Team Resources



Angular Documentation

<http://angular.io/docs>

The screenshot displays the Angular documentation website. The top navigation bar is blue with the Angular logo and links for FEATURES, DOCS, EVENTS, NEWS, and GET STARTED. A search bar labeled 'SEARCH DOCS...' is on the left. A sidebar menu on the left lists various documentation sections: QUICKSTART, TUTORIAL, BASICS (highlighted), and a list of topics under BASICS including Overview, Architecture, Displaying Data, User Input, Forms, Dependency Injection, Template Syntax, and Angular Cheat Sheet. Below these are DEVELOPER GUIDE, COOKBOOK, and API REFERENCE. The main content area has a blue header with 'STYLE GUIDE' and a dropdown menu set to 'Angular 2 for TypeScript'. The text 'Write Angular 2 with style.' is followed by an introduction to the style guide's purpose. A section titled 'Style Vocabulary' explains that guidelines describe good or bad practices. It then details the wording of guidelines, showing three types: 'Do' (green header), 'Consider' (blue header), and 'Avoid' (red header), each with a brief explanation.

ANGULAR

FEATURES DOCS EVENTS NEWS GET STARTED

SEARCH DOCS...

QUICKSTART

TUTORIAL

BASICS

1. Overview

2. Architecture

3. Displaying Data

4. User Input

5. Forms

6. Dependency Injection

7. Template Syntax

8. Angular Cheat Sheet

9. Style Guide

10. Glossary

DEVELOPER GUIDE

COOKBOOK

API REFERENCE

STYLE GUIDE

Angular 2 for TypeScript

Write Angular 2 with style.

The purpose of this style guide is to provide guidance on building Angular applications by showing the conventions we use and, more importantly, why we choose them.

Style Vocabulary

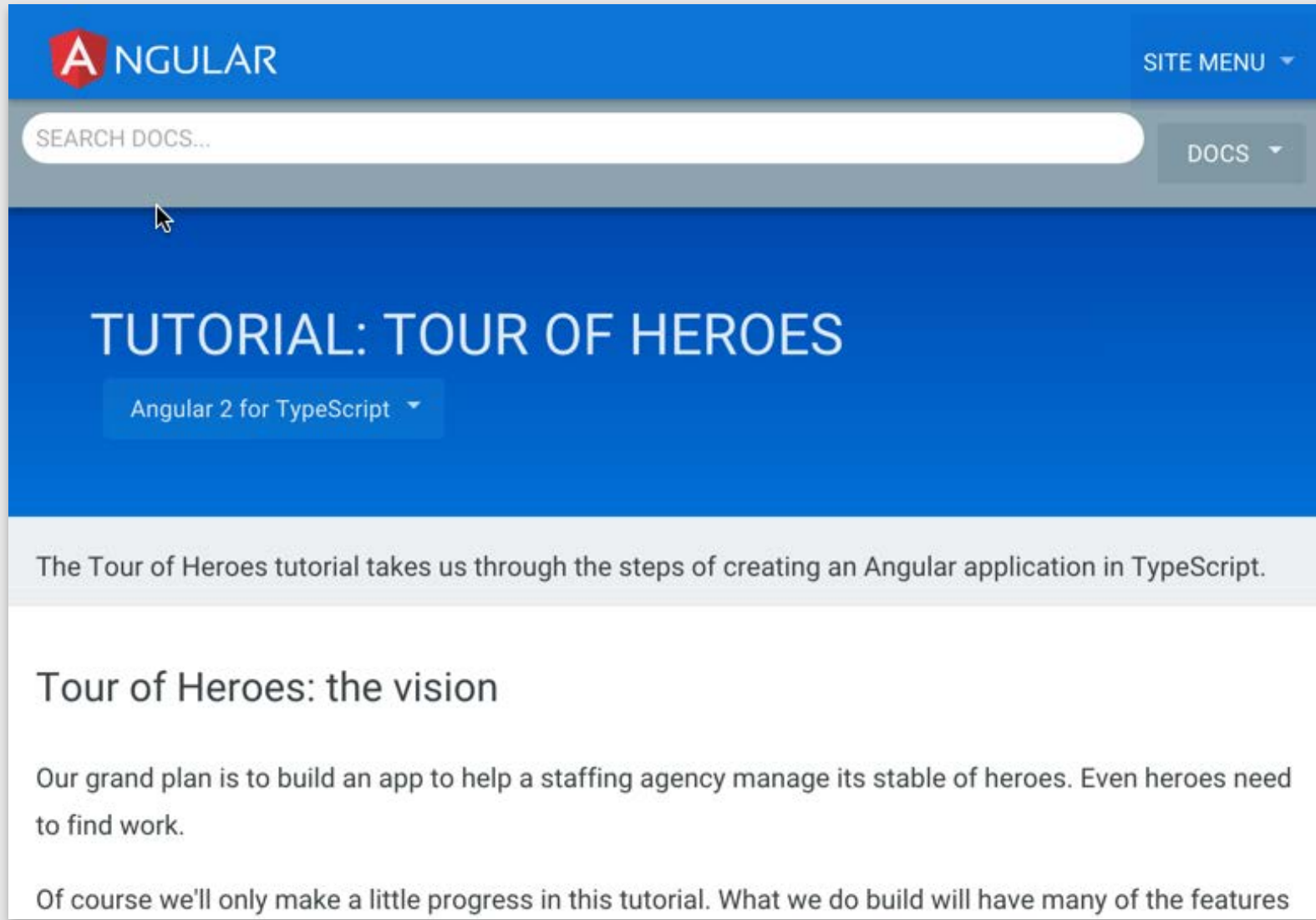
Each guideline describes either a good or bad practice, and all have a consistent presentation.

The wording of each guideline indicates how strong the recommendation is.

- Do** is one that should always be followed. *Always* might be a bit too strong of a word. Guidelines that literally should always be followed are extremely rare. On the other hand, we need a really unusual case for breaking a *Do* guideline.
- Consider** guidelines should generally be followed. If you fully understand the meaning behind the guideline and have a good reason to deviate, then do so. Please strive to be consistent.
- Avoid** indicates something we should almost never do. Code examples to *avoid* have an unmistakable red header.



Tour of Heroes Tutorial



The screenshot shows the Angular website's header with the Angular logo and 'ANGULAR' text. A 'SITE MENU' dropdown is visible. Below the header is a search bar labeled 'SEARCH DOCS...' and a 'DOCS' dropdown. The main content area has a blue background with the title 'TUTORIAL: TOUR OF HEROES' and a sub-header 'Angular 2 for TypeScript'. Below this, a light gray box contains the text: 'The Tour of Heroes tutorial takes us through the steps of creating an Angular application in TypeScript.' The main content area is white and contains the heading 'Tour of Heroes: the vision' followed by two paragraphs of text. A mouse cursor is visible over the search bar. A play button icon is in the bottom right corner.

ANGULAR

SITE MENU

SEARCH DOCS...

DOCS

TUTORIAL: TOUR OF HEROES

Angular 2 for TypeScript

The Tour of Heroes tutorial takes us through the steps of creating an Angular application in TypeScript.

Tour of Heroes: the vision

Our grand plan is to build an app to help a staffing agency manage its stable of heroes. Even heroes need to find work.

Of course we'll only make a little progress in this tutorial. What we do build will have many of the features

Comparing Angular 1 to Angular 2



TypeScript is a Superset of ES6/ES2015+

40+ Built-In Directives Removed

Bind to any HTML Element Property or Event

Angular 1 Concepts Map to Angular 2

