



ACADEMY

LAPIDANDO TALENTOS



CONHECENDO O ANGULAR ~~10~~

AULA 3 – ANGULAR (PARTE 1)

Eder Franco @ FPF Tech

Agenda

1. Breve histórico
 - AngularJS;
 - Entendendo as versões.
2. Como era no AngularJS?
 - Arquitetura;
 - Migrando AngularJS para Angular;
3. Preparação do ambiente
4. Download e configuração
5. Angular CLI
 - Criando o primeiro projeto
 - Principais recursos
6. Arquitetura do Angular
7. Referências

Breve histórico

A long time ago in a galaxy far,
far away....

Breve histórico

Um framework Super-heróico de Javascript!



Breve histórico



Miško Hevery e Adam Abrons (2009)

Breve histórico

- Miško Hevery foi contratado pelo Google;
- Foi trabalhar em um projeto chamado Google Feedback;
- O projeto possuía 17 mil linhas de código JS;
- Miško fez uma afirmação forte: refazer o projeto em bem menos tempo com seu framework Angular JS;
- O projeto foi refeito em pouco mais de 3 semanas, com 1500 linhas;

Breve histórico

- O Angular foi abraçado pelo Google e passou a ser mantido por mais programadores da casa e utilizado em vários projetos;
- O nome Angular vem de uma referência sonora à palavra “angle”, referente ao ângulo formado pelos símbolos que formam tags HTML;
- Atualmente:
 - Angular 1.6.x (julho/2017);
 - Angular 6 (dezembro/2017);

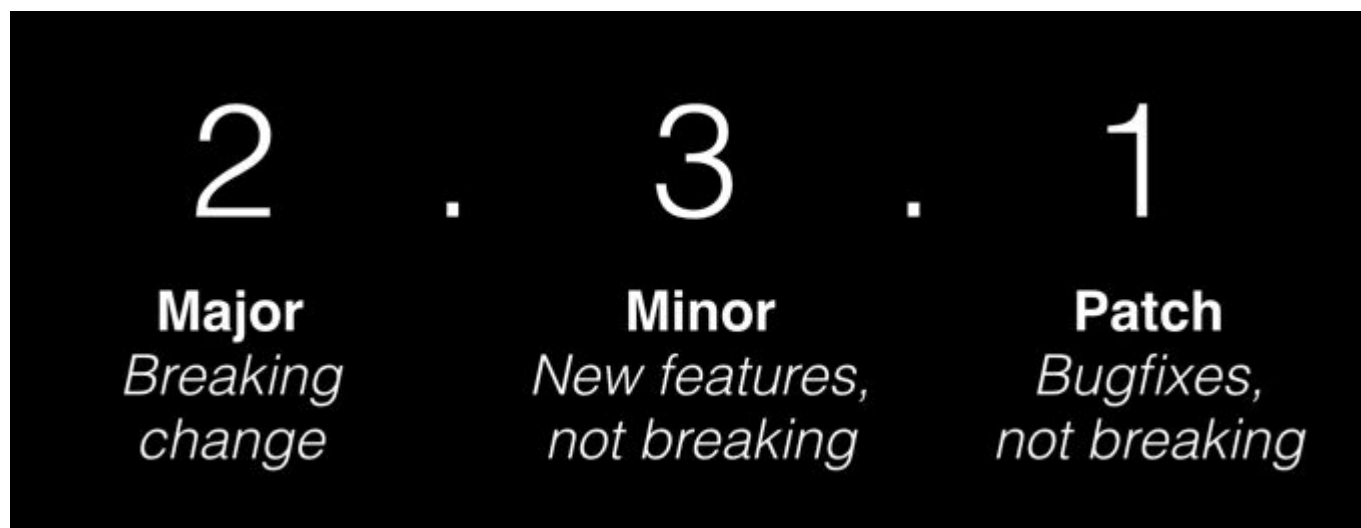
Entendendo as versões

Breve histórico

- Por que Angular?
- Vamos chamar somente de ANGULAR 😊
- Quando a nova versão do framework foi lançada, o time decidiu mudar o esquema de versão para Semantic Versioning (SEMVER);

Breve histórico

- Exemplo de *Semantic Versioning*:



Breve histórico

- Dessa forma:
 - Quando o novo Angular foi lançado, foi uma mudança ENORME: novas APIs, novos conceitos... Tudo novo. Então tornou-se a versão 2;
 - Em março de 2017 o time precisou atualizar dependências do Angular e estava migrando do TypeScript 1.8 para 2.2, o que foi uma grande mudança, que levaria ao incremento da *major version* para 3.

Breve histórico

- E por que não houve um Angular 3?
 - Por cauda do Angular Router!
 - As dependências do Angular são distribuídas todas na mesma versão, mas na época de lançamento do possível “Angular 3”, a versão do Angular Router estava uma *major release* acima das demais dependências;

Breve histórico

- E por que não houve um Angular 3?

@angular/core	v2.3.0
@angular/compiler	v2.3.0
@angular/compiler-cli	v2.3.0
@angular/http	v2.3.0
@angular/router	v3.3.0

Breve histórico

Breve histórico

- Para manter as versões alinhadas, o time do Angular decidiu “pular” a versão 3 e lançar o Angular 4 diretamente;



Breve histórico

Breve histórico

- Atualmente, o time do Angular segue este planejamento:
 - Releases de patches a cada semana;
 - 3 minor releases mensais após cada major release;
 - Uma major release com grandes mudanças e facilidade de migração a cada seis meses;



Major Release Cycle

Breve histórico

- Planejamento de próximas versões do Angular:

Predictable, Transparent & Incremental Evolution

Version 4	March 2017
Version 5	September/October 2017
Version 6	March 2018
Version 7	September/October 2018

(tentative schedule)

Como era no AngularJS?

Como era no AngularJS?

Características

- Framework para construção aplicações web dinâmicas;
- Single Page Application (SPA);
- Extensão do HTML por meio de diretivas;
- Permite a utilização de javascript puro e simples;
- Funciona em vários browsers (*cross browser compliance*);

Como era no AngularJS?

Características

- Facilidade para comunicação com APIs;
- Facilmente testável;
- Injeção de dependências (baixo acoplamento);
- MVW ([Model – View – Whatever](#)).
 - MVC, MVVM, MV...

Arquitetura

MODULE
<HTML NG-APP="MODULENAME">

CONFIG

ROUTES

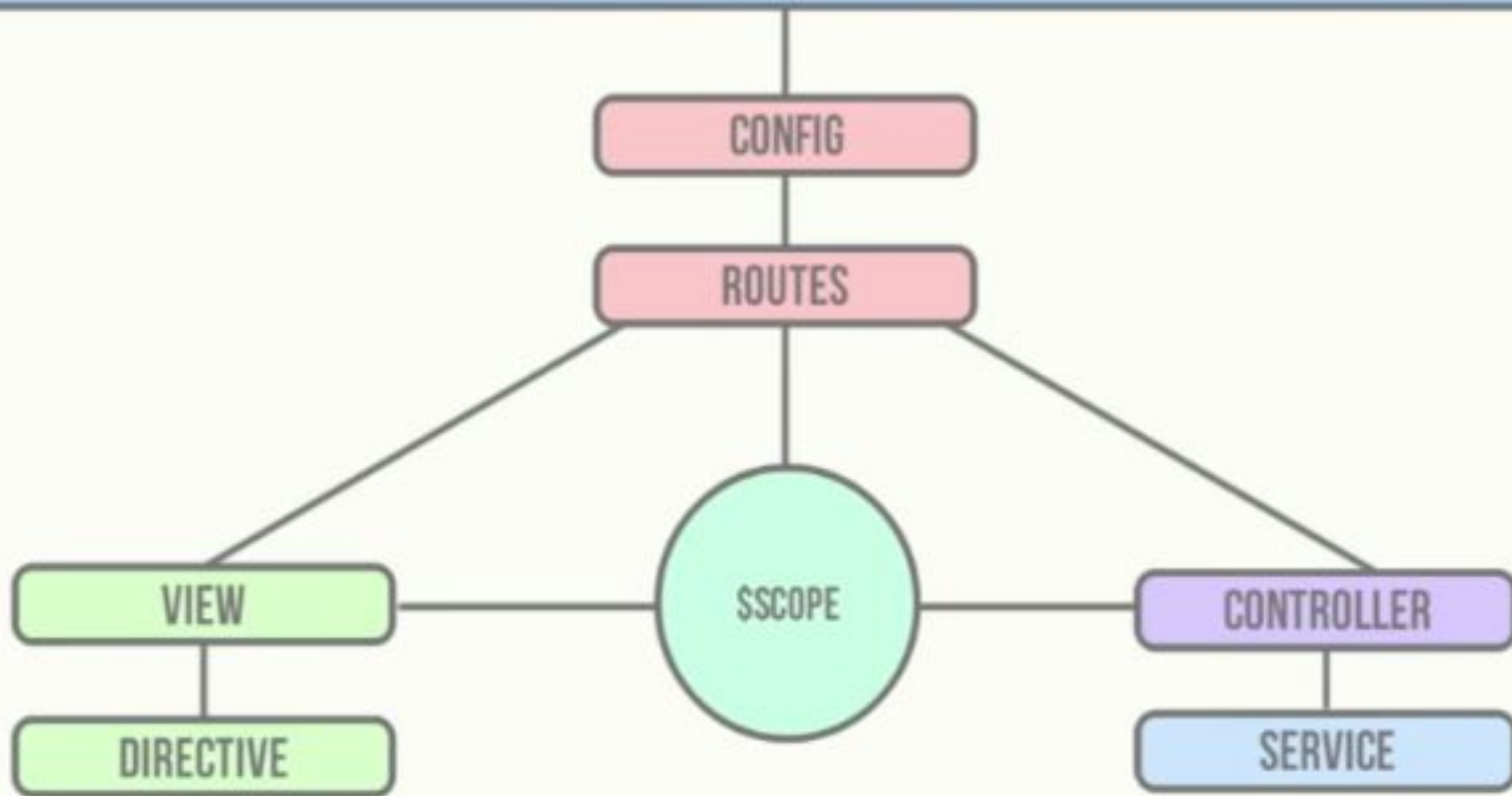
VIEW

DIRECTIVE

\$SCOPE

CONTROLLER

SERVICE



Como era no AngularJS?

Arquitetura

- A diretiva ng-app vincula um AngularJS Module ao HTML, criando um SPA;
- Blocos de configuração definem especificações de dependências internas e externas do AngularJS;
- Um bloco de execução (run) atua como uma espécie de método *main* para a aplicação em AngularJS;

Como era no AngularJS?

Arquitetura

- Routs (route) são uma forma de especificar views que serão carregadas dinamicamente e injetadas no conteúdo do app principal (inde.html), sem necessidade de recarregar todos os recursos da página;
- Um dos componentes mais importantes do AngularJS é o \$scope;

Como era no AngularJS?

Arquitetura

- O \$scope é a “cola” entre uma view (HTML) e um controller;
- Cada view deve estar associada a um só controller;
- Services são uma forma de compartilhar lógica entre vários componentes, e são injetados nos controllers;

Como era no AngularJS?

Arquitetura

- Diretivas permitem criar elementos, atributos, classes e comentários que estendem o HTML e implementam recursos especiais nas views;
- Diretivas são utilizadas pelas views;
- Angular interpreta as diretivas e compila o HTML para gerar o resultado que visualizamos;

Como era no AngularJS?

Arquitetura

- Outros recursos interessantes do AngularJS:
 - Filtros (filters);
 - Sistema de eventos próprio (\$emit, \$on);
 - Watchers (\$watch);
 - Components (versão 1.5);



Como era no AngularJS?

Mãos à obra!

- Criando um CRUD:
 - Agora vamos criar uma aplicação simples com Angular JS e Bootstrap, de maneira manual;
 - Nossa aplicação fará um CRUD de um usuário somente no front-end (utilizando localStorage);
 - Este exercício será realizado em conjunto;



Migrando do AngularJS para o Angular

Migrando do AngularJS para o Angular

Alguns problemas do AngularJS

- Organização do código não era padronizada;
- Carregar arquivos JS do Angular e dependências era um processo manual;
- Debugar não era uma tarefa tão fácil;
- Dependência do jQuery;

Migrando do AngularJS para o Angular

Alguns problemas do AngularJS

- Muitas camadas e estruturas;
- Dificuldades para gerenciar escopo (hierarquia, compartilhamento);
- Injeção de dependências complicada;
- Sintaxe complicada e não padronizada;

Migrando do AngularJS para o Angular

Alguns problemas do AngularJS

- Muitas camadas e estruturas;
- Dificuldades para gerenciar escopo (hierarquia, compartilhamento);
- Problemas de performance por abuso de watchers e uso de *two-way data-binding*;
- Sérios problemas de performance em dispositivos móveis;

Migrando do AngularJS para o Angular

E o que aconteceu?

- Proliferação de um ferramental excessivo de bibliotecas, plug-ins, gerenciadores de tarefas, etc:
 - Grunt, gulp, bower, etc.
- Frameworks bem intencionados, mas que prejudicavam o trabalho:
 - Angular Material 1 excetuando em um aplicativo móvel;
- Dificuldades para simplificar o trabalho e ganhar produtividade;



RIP AngularJS (?)

Migrando do AngularJS para o Angular

O que é necessário para migrar?

- Reestruturar arquitetura;
- Utilizar um *module loader*;
- Adicionar TypeScript;
- Reescrever diretivas e *controllers* para *Components*;
- Criar uma aplicação híbrida;
- Seguir o Angular Code Style;

NO, THANKS



I'M GOOD

Migrando do AngularJS para o Angular

O que é necessário para migrar?

- O ideal é recomençar ou implementar novos recursos a partir de determinado ponto, e criar compatibilidade apenas visual entre as aplicações;
- Compreender a equivalência de conceitos entre os dois frameworks pode ajudar bastante neste processo:
 - <https://angular.io/guide/ajs-quick-reference>

Migrando do AngularJS para o Angular

AngularJS x Angular

- Interpolação
 - Permanece semelhante, mas agora é apenas *one-way*.

```
Your favorite hero is:  
{{vm.favoriteHero}}
```

```
Your favorite hero is:  
{{favoriteHero}}
```


Migrando do AngularJS para o Angular

AngularJS x Angular

- Filtros
 - Uso similar, mas agora se chama *pipes*, e muitos filtros *built-in* foram removidos.

```
<td>{{movie.title |  
uppercase}}</td>
```

```
<td>{{movie.title |  
uppercase}}</td>
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- Variáveis locais
- Nas *templates*, é possível criar variáveis locais usando *let*.

```
<tr ng-repeat="movie in  
vm.movies">  
  <td>{{movie.title}}</td>  
</tr>
```

```
<tr *ngFor="let movie of  
movies">  
  <td>{{movie.title}}</td>  
</tr>
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- *Bootstrapping*
 - Antes utilizava *ng-app*. Agora há uma classe e um módulo para este fim.
 - Ver <https://angular.io/guide/ajs-quick-reference#ng-app>

Migrando do AngularJS para o Angular

AngularJS x Angular

- Convenção de nomes:
 - As diretivas de atributos agora seguem a convenção de nomes camel case, ao invés da separação com traços.
 - Antes: ng-class, ng-if
 - Agora: ngClass, ngIf

Migrando do AngularJS para o Angular

AngularJS x Angular

- Vínculo de eventos
 - Antes era realizado com diretivas ng-<nome-do-evento>;
 - Agora é feito com o uso de parênteses:

```
<button ng-click="vm.toggleImage()">  
<button ng-click="vm.toggleImage($event)">
```

```
<button (click)="toggleImage()">  
<button (click)="toggleImage($event)">
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- ng-controller
 - Controllers não existem mais;
 - O vínculo de uma view a seu controlador lógico é feito com a criação de uma classe simples, com o *decorator* @Component;
 - Componentes são a principal estrutura do Angular;

Migrando do AngularJS para o Angular

AngularJS x Angular

```
<div ng-controller="MovieListCtrl as vm">
```

```
@Component({  
  selector: 'app-movie-list',  
  templateUrl: './movie-list.component.html',  
  styleUrls: [ './movie-list.component.css' ],  
})
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- Diretivas substituídas por bind de propriedades:
- Com o uso de chaves é possível fazer o bind de propriedades HTML:
 - Antes: disabled, ng-hide, ng-show, ng-href

```
<h3 ng-show="vm.favoriteHero">  
  Your favorite hero is: {{vm.favoriteHero}}  
</h3>
```

```
<h3 [hidden]="!favoriteHero">  
  Your favorite hero is: {{favoriteHero}}  
</h3>
```


Migrando do AngularJS para o Angular

AngularJS x Angular

- *Two-way data-binding*
 - A diretiva ngModel agora requer o uso de uma notação diferente:

```
<input ng-model="vm.favoriteHero"/>
```

```
<input [(ngModel)]="favoriteHero" />
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- ng-repeat
 - A diretiva foi substituída pelo *ngFor

```
<tr ng-repeat="movie in vm.movies">
```

```
<tr *ngFor="let movie of movies">
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- Injeção de dependências
 - Foi simplificada com o uso dos construtores de classe.

```
MovieListCtrl.$inject =  
  ['MovieService'];  
function MovieListCtrl(movieService)  
{  
}
```

```
constructor(movieService:  
  MovieService) {  
}
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- Folhas de estilo
- Não existia *scope style*. Agora é possível definir `styleUrls` para cada componente.

```
<link href="styles.css"
rel="stylesheet" />
```

```
styleUrls: [ './movie-
list.component.css' ],
```

Migrando do AngularJS para o Angular

AngularJS x Angular

- *Scaffolding*
 - Não existiam ferramentas e padrões para definir a arquitetura do projeto e organização do código;
 - Agora existe o poderoso Angular CLI:
 - <https://cli.angular.io/>

Preparação do Ambiente

Preparação do Ambiente

Mãos à obra!

- Instalar o Node.js:
 - <https://nodejs.org/en/download/>
- Instalar o TypeScript
 - `npm install -g typescript`



Download e Configuração

Mãos à obra!

- Plugins para o Visual Studio Code:
 - Auto Import
 - TSLint
 - Angular 6 Snippets - TypeScript, Html, Angular Material, ngRx, RxJS & Flex Layout



Download e Configuração

Preparação do Ambiente

Mãos à obra!

- Instalar o Angular CLI
 - `npm install -g @angular/cli`
- Após finalizar a instalação:
 - `ng --version`



Angular CLI

Conhecendo o Angular CLI

- Baseado no Ember CLI:
 - <https://ember-cli.com/>
- Onde encontrar?
 - <https://github.com/angular/angular-cli>
- Wiki:
 - <https://github.com/angular/angular-cli/wiki/>

Criando o primeiro projeto

Criando o primeiro projeto

Criando e executando um novo projeto

- ng new <options>
 - ng new nome-do-projeto
- Executar o projeto
 - ng serve

```
PS C:\Users\ederf\ANGULAR6\exemplos-ng6> ng new projeto1
```

```
CREATE projeto1/angular.json (3566 bytes)
CREATE projeto1/package.json (1312 bytes)
CREATE projeto1/README.md (1025 bytes)
CREATE projeto1/tsconfig.json (384 bytes)
CREATE projeto1/tslint.json (2805 bytes)
CREATE projeto1/.editorconfig (245 bytes)
CREATE projeto1/.gitignore (503 bytes)
CREATE projeto1/src/environments/environment.prod.ts (51 bytes)
CREATE projeto1/src/environments/environment.ts (631 bytes)
CREATE projeto1/src/favicon.ico (5430 bytes)
CREATE projeto1/src/index.html (295 bytes)
CREATE projeto1/src/main.ts (370 bytes)
CREATE projeto1/src/polyfills.ts (3194 bytes)
CREATE projeto1/src/test.ts (642 bytes)
CREATE projeto1/src/assets/.gitkeep (0 bytes)
CREATE projeto1/src/styles.css (80 bytes)
CREATE projeto1/src/browserslist (375 bytes)
CREATE projeto1/src/karma.conf.js (964 bytes)
CREATE projeto1/src/tsconfig.app.json (194 bytes)
CREATE projeto1/src/tsconfig.spec.json (282 bytes)
CREATE projeto1/src/tslint.json (314 bytes)
CREATE projeto1/src/app/app.module.ts (314 bytes)
CREATE projeto1/src/app/app.component.html (1141 bytes)
CREATE projeto1/src/app/app.component.spec.ts (991 bytes)
CREATE projeto1/src/app/app.component.ts (207 bytes)
```

Criando o primeiro projeto

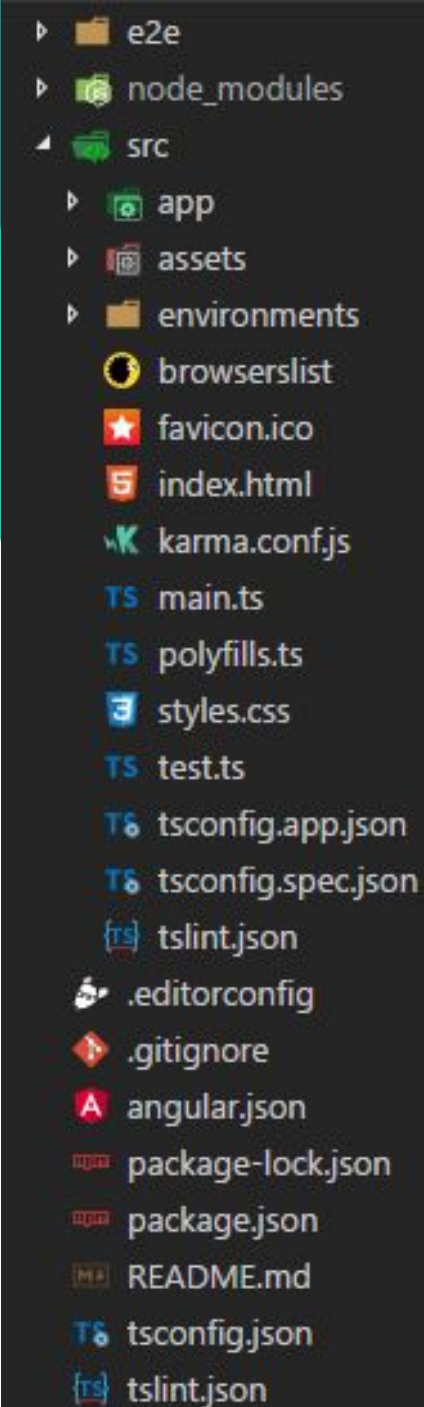
Criando e executando um novo projeto

- ng new <options>
 - ng new nome-do-projeto
- Executar o projeto
 - ng serve
 - O projeto é iniciado em http://localhost:4200
 - É possível escolher host e porta:
 - ng serve --host 0.0.0.0 --port 4201

Criando o primeiro projeto

Entendendo o que foi gerado

- Diretórios:
 - e2e – Testes *end to end*;
 - node_modules – Pacotes de dependências locais do npm (especificadas no arquivo package.json);
 - src – Código-fonte do aplicativo.



```

package.json x
1 {
2   "name": "projeto1",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve",
7     "build": "ng build",
8     "test": "ng test",
9     "lint": "ng lint",
10    "e2e": "ng e2e"
11  },
12  "private": true,
13  "dependencies": {
14    "@angular/animations": "^6.0.3",
15    "@angular/common": "^6.0.3",
16    "@angular/compiler": "^6.0.3",
17    "@angular/core": "^6.0.3",
18    "@angular/forms": "^6.0.3",
19    "@angular/http": "^6.0.3",
20    "@angular/platform-browser": "^6.0.3",
21    "@angular/platform-browser-dynamic": "^6.0.3",
22    "@angular/router": "^6.0.3",
23    "core-js": "^2.5.4",
24    "rxjs": "^6.0.0",
25    "zone.js": "^0.8.26"
26  },
27  "devDependencies": {
28    "@angular/compiler-cli": "^6.0.3",
29    "@angular-devkit/build-angular": "~0.6.8",
30    "typescript": "~2.7.2",
31    "@angular/cli": "~6.0.8",
32    "@angular/language-service": "^6.0.3",
33    "@types/jasmine": "~2.8.6",
34    "@types/jasminewd2": "~2.0.3",
35    "@types/node": "~8.9.4",
36    "codifyer": "~4.2.1",
37    "jasmine-core": "~2.99.1",
38    "jasmine-spec-reporter": "~4.2.1",
39    "karma": "~1.7.1",
40    "karma-chrome-launcher": "~2.2.0",
41    "karma-coverage-istanbul-reporter": "~2.0.0",
42    "karma-jasmine": "~1.1.1",
43    "karma-jasmine-html-reporter": "^0.2.2",
44    "protractor": "~5.3.0",
45    "ts-node": "~5.0.1",
46    "tslint": "~5.9.1"
47  }
48 }

```

Criando o primeiro projeto









Entendendo o que foi gerado

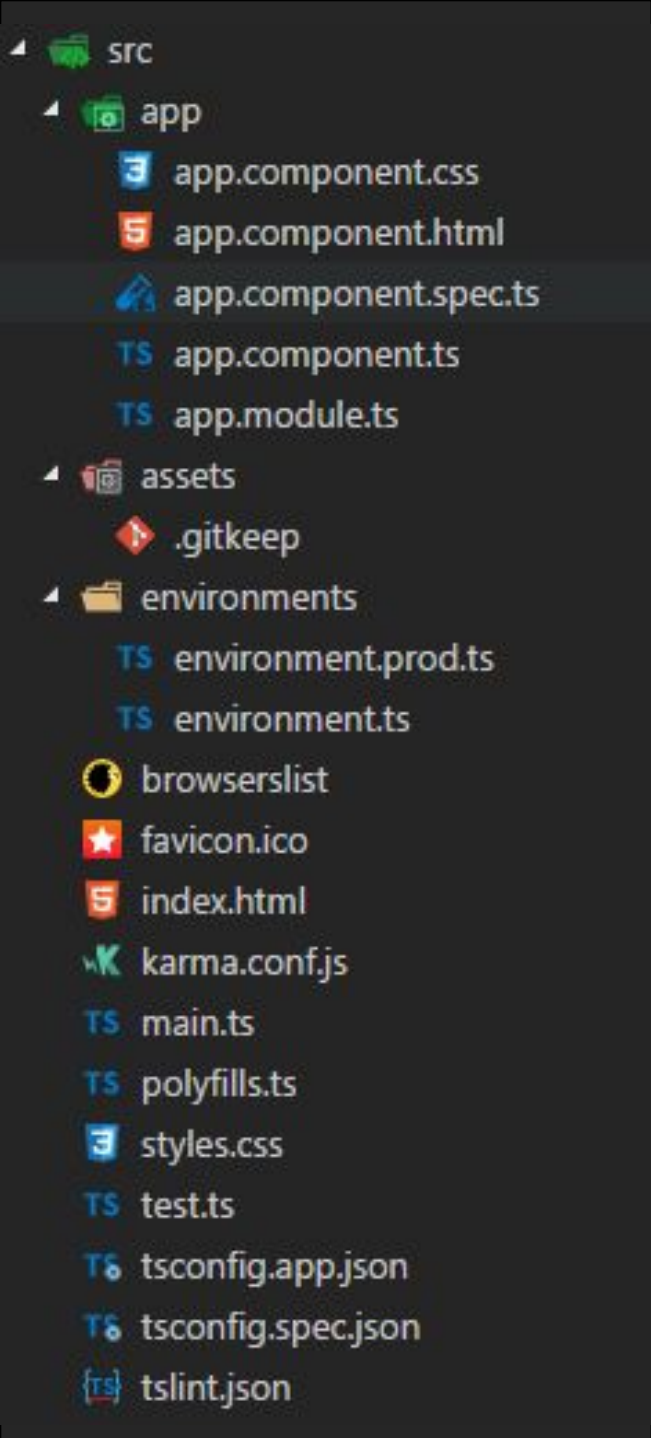
- package.json
 - Definições do projeto;
 - Scripts (ng serve, build, test, etc);
 - Lista de dependências;

Criando o primeiro projeto

Entendendo o que foi gerado

- package-lock.json – Arquivo de “lock” das versões de dependências;
- angular.json – Definições do angular CLI e configurações gerais;
- .gitignore e .editorconfig – Configurações do git e do editor de código;
- tsconfig.json e tslint.json – Configurações do globais do TypeScript e do TSlint;

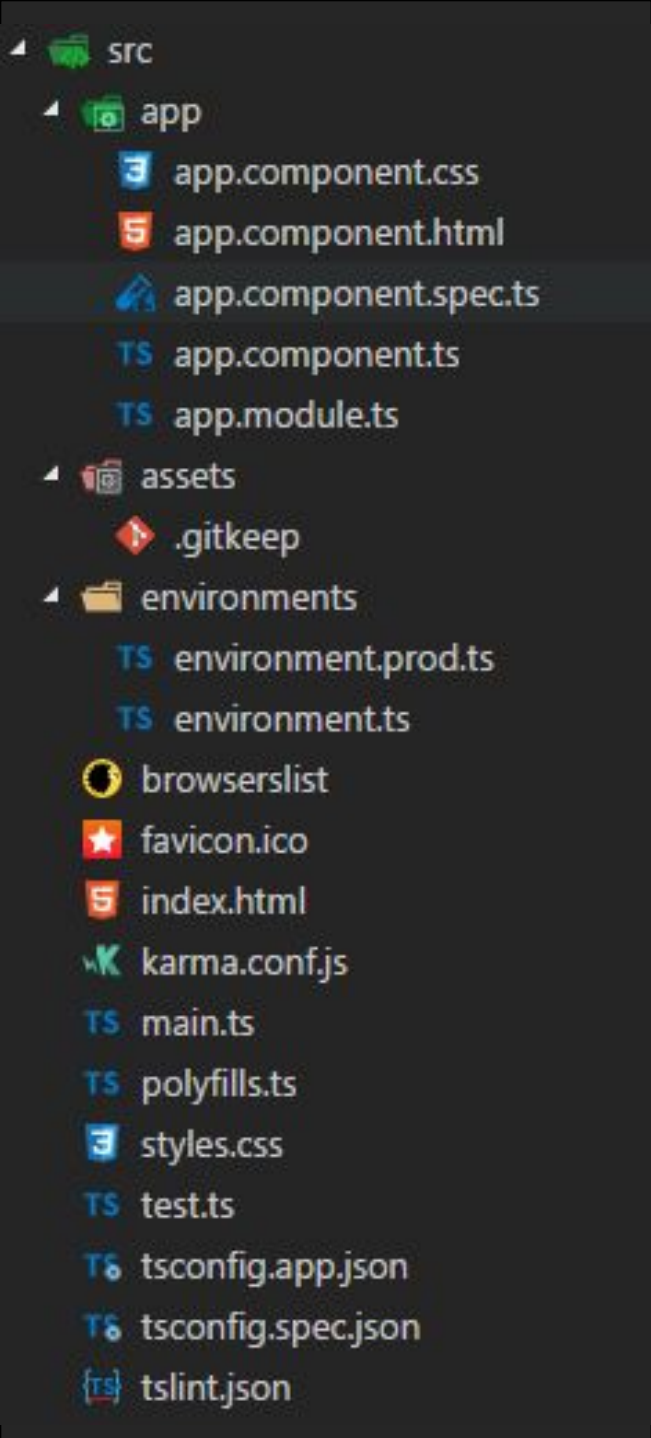
 .editorconfig
 .gitignore
 angular.json
 package-lock.json
 package.json
 README.md
 tsconfig.json
 tslint.json



Criando o primeiro projeto

Entendendo o que foi gerado

- app – Código fonte do app (módulo, componente global e estilo global);
- assets – Assets do projeto, como imagens e bibliotecas que não sejam do npm;
- environments – Definições de variáveis de ambiente (dev, prod, etc);
- style.css – Estilos globais da aplicação;
- karma.conf.js – Definições do framework de testes;



Criando o primeiro projeto

Entendendo o que foi gerado

- browserlist – Definições de browsers suportados para o plugin de *autoprefixer* de css;
- main.ts – O arquivo que inicializa o Angular;
- tsconfig.app.json, tslint.json – Definições específicas do TS para o projeto;
- polyfills.ts – Configurações auxiliares de suporte ao ES6 em diferentes browsers;
- test.ts – Configurações globais para inicialização dos testes do projeto.

Principais recursos

Instalar bibliotecas de terceiros

- Instalação de bibliotecas de terceiros é realizada com npm
 - `npm install nome-da-biblioteca --save`
- Devem ser instaladas também as definições de tipo @types
 - `npm install d3 --save`
 - `npm install @types/d3 --save-dev`

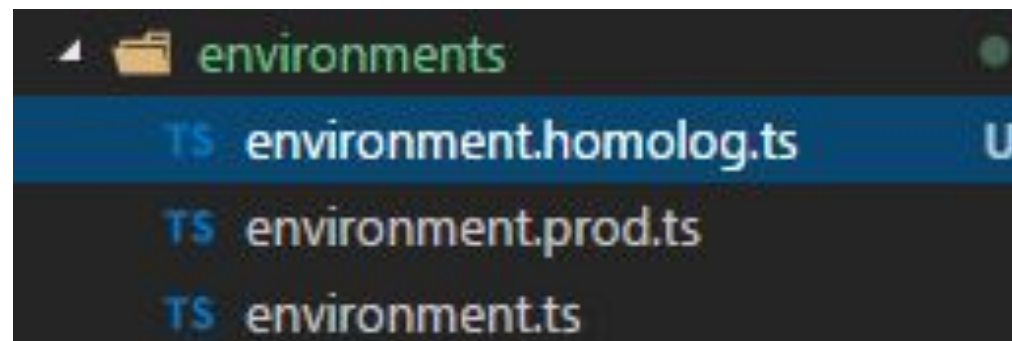
Variáveis de ambiente

- Podemos mapear ambientes no arquivo cli.json
- Estas variáveis serão carregadas dinamicamente, de acordo com o ambiente definido no build (utilizando a opção `–prod`, por exemplo);

```
"configurations": {
  "production": {
    "fileReplacements": [
      {
        "replace": "src/environments/environment.ts",
        "with": "src/environments/environment.prod.ts"
      }
    ],
    "optimization": true,
    "outputHashing": "all",
    "sourceMap": false,
    "extractCss": true,
    "namedChunks": false,
    "aot": true,
    "extractLicenses": true,
    "vendorChunk": false,
    "buildOptimizer": true
  },
}
```


Variáveis de ambiente

- Por padrão, o angular.json traz mapeadas configurações para um ambiente adicional de produção;
- Os arquivos de variáveis de ambiente estão no diretório *environments* (e novos arquivos podem ser criados nestes diretório):



Variáveis de ambiente

- Sempre que um novo arquivo de variáveis de ambiente for adicionar, é necessário criar um novo bloco de configurações no arquivo angular.json.

```
"homolog": {  
  "fileReplacements": [  
    {  
      "replace": "src/environments/environment.ts",  
      "with": "src/environments/environment.homolog.ts"  
    }  
  ],  
  "optimization": false,  
  "outputHashing": "all",  
  "sourceMap": true,  
  "extractCss": false,  
  "namedChunks": false,  
  "aot": false,  
  "extractLicenses": false,  
  "vendorChunk": false,  
  "buildOptimizer": false  
}
```

Variáveis de ambiente

- Constantes definidas como variáveis de ambiente podem ser importadas nos componentes:

```
export const environment = {  
  production: false,  
  envName: 'dev'  
};
```

```
import { Component } from '@angular/core';  
import { environment } from '../environment';  
  
@Component({  
  moduleId: module.id,  
  selector: 'myapp-app',  
  templateUrl: 'myapp.component.html',  
  styleUrls: ['myapp.component.css']  
})  
export class MyAppAppComponent {  
  title = 'myapp works!';  
  environmentName = environment.envName;  
}
```

Lista dos principais comandos

- *Build* e outros recursos para *deploy* da aplicação serão discutidos na Aula 4;
- A documentação completa do Angular CLI está disponível em:
 - <https://github.com/angular/angular-cli>

Scaffold	Usage
Component	<code>ng g component my-new-component</code>
Directive	<code>ng g directive my-new-directive</code>
Pipe	<code>ng g pipe my-new-pipe</code>
Service	<code>ng g service my-new-service</code>
Class	<code>ng g class my-new-class</code>
Guard	<code>ng g guard my-new-guard</code>
Interface	<code>ng g interface my-new-interface</code>
Enum	<code>ng g enum my-new-enum</code>
Module	<code>ng g module my-module</code>

Outros recursos importantes (próximas aulas):

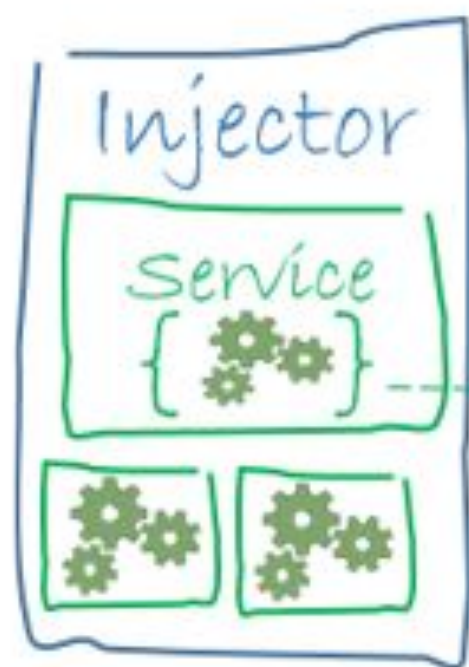
- Instalando bibliotecas globais;
 - Bootstrap e Angular Material, por exemplo;
- Criando projeto com pré-processadores:
 - `ng new projeto --style=scss`
- Criando projeto com roteamento:
 - `ng new projeto --routing`

Arquitetura

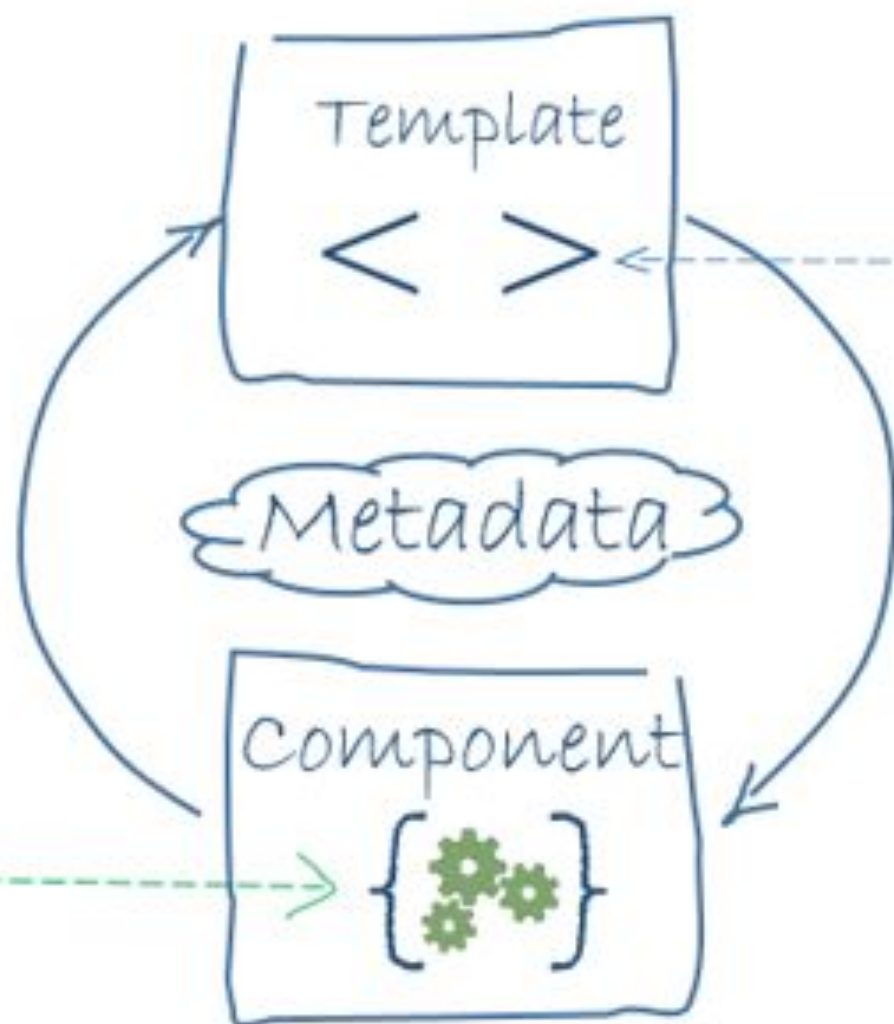
Angular

- Definições de Arquitetura do Angular têm como elemento central o componente;
- Todos os principais recursos do Angular são classes ES6 / TS com *decorators* para atribuição de comportamentos especiais;
- Informações detalhadas sobre a arquitetura do Angular estão disponíveis em:
 - <https://angular.io/guide/architecture>

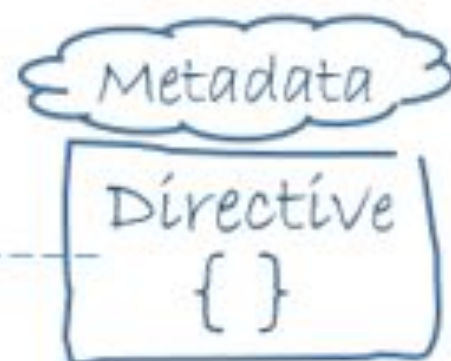
Module Component { }	Module Service { }
Module value 3.1415	Module Fn λ



Property Binding



Event Binding



Referências

Referências

- Angular Blog. **...Just Angular**. Disponível em <http://angularjs.blogspot.com.br/2016/12/ok-let-me-explain-its-going-to-be.html> Acesso em 03/06/2017.
- Angular Blog. **Versioning and Releasing Angular**. Disponível em http://angularjs.blogspot.com.br/2016/10/versioning-and-releasing-angular.html#Timebased_release_cycles_18 Acesso em 03/06/2017.
- GRANT, Garry. **Top 5 Issues Large Sites Have with AngularJS**. Disponível em <https://www.searchenginepeople.com/blog/top-5-technical-issues-large-sites-angularjs.html> Acesso em: 07/10/2017.
- Angular. **Angular Official Website**. Disponível em <https://angular.io> Acesso em: 07/10/2017.

Eder Martins Franco

eder.franco@fpf.br
efranco23@gmail.com



facebook.com/efranco23
linkedin.com/in/efranco23
moodle.franco.eti.br