

# Angular

**Introdução, componentes e binding**

**Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)**

# Angular

- Framework opensource criado pelo Google;
- Baseado em componentes;
- Possui uma gama de ferramentas de infraestrutura integradas e gerenciáveis via node;
- Ideal para *Single-Page Applications* (SPA);
- <https://angular.io/>



# Ferramentas

- Node.js;
- Visual Studio Code;
- Angular CLI.

# Comandos a serem executados

- Instalando o Angular CLI:  
`>> npm install -g @angular/cli@7.0.6`
- Checando a versão:  
`>> ng --version`
- Criando o projeto do curso  
`>> ng new angular-tour-of-heroes`
- Testando a aplicação:
  - Dentro da pasta angular-tour-of-heroes:  
`>> ng serve --open`

# Caso apareça algo do tipo

```
found 14 vulnerabilities (9 low, 5 high)  
  run `npm audit fix` to fix them, or `npm audit` for details
```

<https://stackoverflow.com/questions/49582891/proper-way-to-fix-potential-security-vulnerability-in-a-dependency-defined-in-pa>

# Executando o projeto

- O comando anterior sobe um servidor de testes do node e abre a página no navegador:



**Welcome to app!**

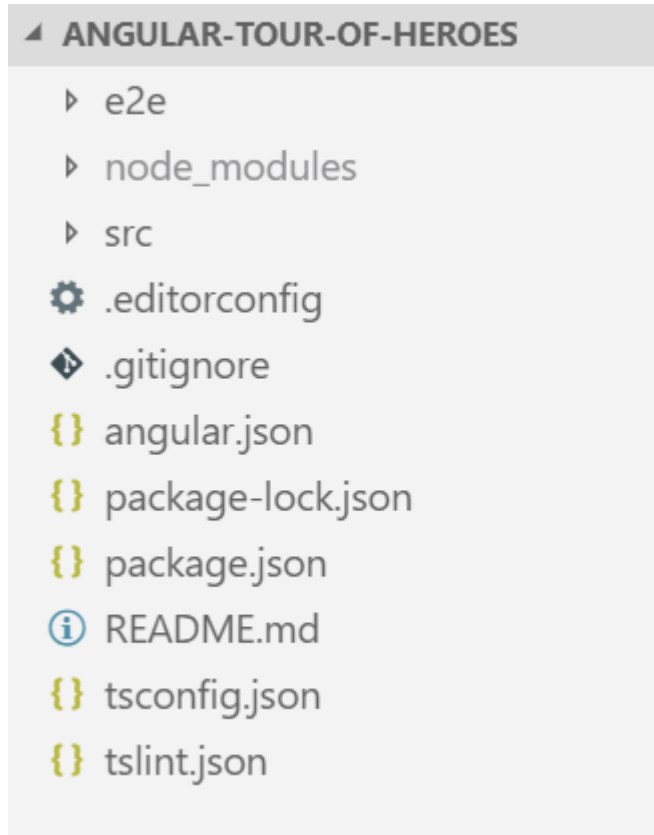


**Here are some links to help you start:**

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)

# Estrutura do projeto

- A estrutura gerada



... e um zilhão de arquivos

# O HTML gerado

- Começando a entender o projeto pelo arquivo /src/index.html:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>AngularTourOfHeroes</title>
    <!-- ... -->
  </head>
  <body>
    <app-root></app-root>
  </body>
</html>
```

*E os links e imagens  
da página anterior?*

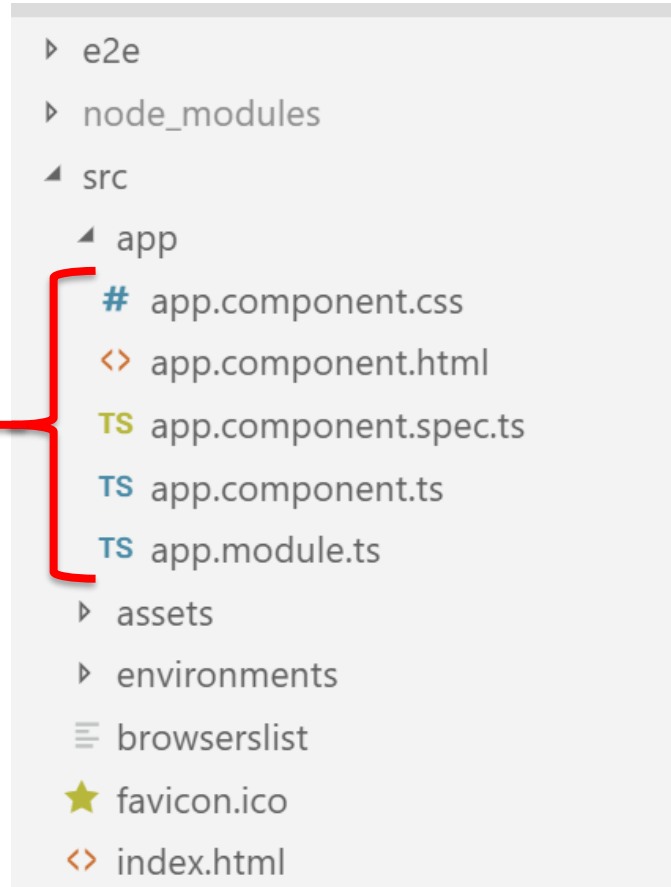




# A tag app-root

- É um componente e suas definições estão nos arquivos abaixo:

<app-root></app-root>

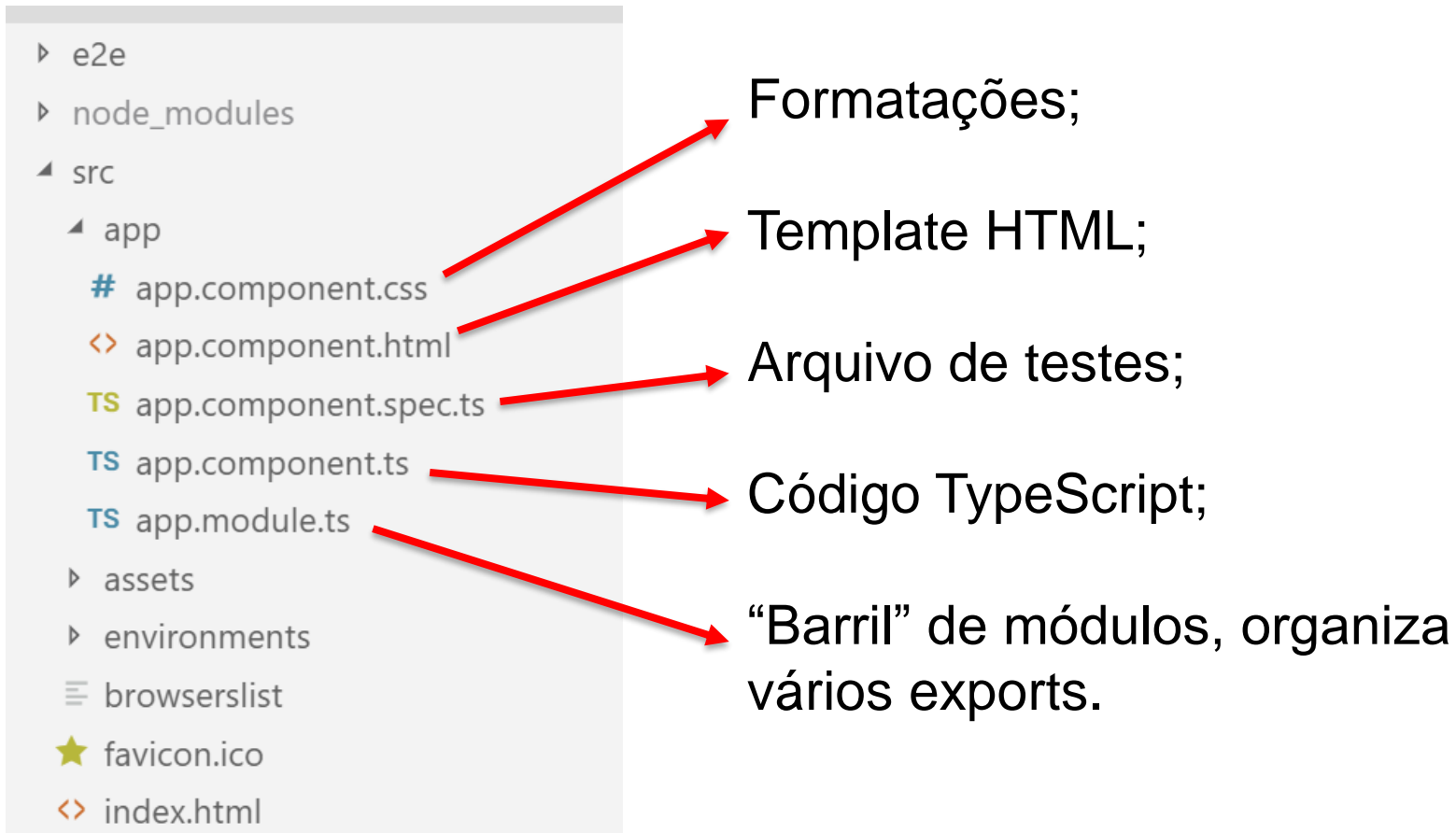


# Convenção

- O nome dos arquivos seguem o padrão:
  - <Nome do componente em minúsculo> + “.component.” + <extensão>
  - Exemplos:
    - app.component.ts
    - app.component.html
    - Etc...

# Componente gerado

- O nome do componente é **app** e os arquivos relacionados a ele são:



The diagram illustrates the file structure of an Angular component named 'app'. The structure is shown on the left, and the descriptions of the files are on the right, connected by red arrows.

**File Structure:**

- src
  - app
    - app.component.css
    - app.component.html
    - app.component.spec.ts
    - app.component.ts
    - app.module.ts
  - assets
  - environments
  - browserslist
  - favicon.ico
  - index.html

**Descriptions:**

- Formatações;
- Template HTML;
- Arquivo de testes;
- Código TypeScript;
- “Barril” de módulos, organiza vários exports.

# Componentes

- Os principais elementos do framework são os componentes:
  - Possuem e expõem funcionalidades;
  - Podem ter definições de apresentação;
  - Representam uma unidade;
  - Funcionam em composição com outros componentes formando um algo mais complexo.

# app.component.html

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>Welcome to {{ title }}! </h1>
  
</div>
<h2>Here are some links to help you start: </h2>
<ul>
  <li>
    <h2><a target="_blank" rel="noopener"
      href="https://angular.io/tutorial">Tour of Heroes</a></h2>
  </li>
  <li>
    <h2><a target="_blank" rel="noopener"
      href="https://github.com/angular/angular-cli/wiki">CLI
      Documentation</a></h2>
  </li>
  <li>
    <h2><a target="_blank" rel="noopener"
      href="https://blog.angular.io/">Angular blog</a></h2>
  </li>
</ul>
```



# app.component.ts

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

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

```
export class AppComponent {  
  title = 'app';  
}
```



# app.component.ts

- O decorator @Component possui definições de:
  - o nome da tag do componente (selector)
  - o arquivo de renderização; (templateUrl);
  - Arquivos de formatação usados (styleUrls).

## app.component.ts

- A classe AppComponent encapsula os atributos do componente;
- Uma instância dessa classe é criada quando a tag `<app-root></app-root>` é utilizada.



# Missão

- Encontrar os arquivos transpilados



# Data Binding

- Processo de vincular atributos de um componente a uma expressão `{{ }}` no template HTML;
- As chaves duplas são a sintaxe de ligação de interpolação (binding) do Angular:

– No HTML:

```
<h1>
```

```
  Welcome to {{ title }}!
```

```
</h1>
```

– No componente:

```
export class AppComponent {
```

```
  title = 'app';
```

```
}
```

# Alterando o componente app

- Em vez de criar um componente do zero, vamos alterar o já existente;
- Em app.component.ts altere a linha:

```
//...
```

```
export class AppComponent {  
  title = 'Tour of Heroes';  
}
```

- Veja que a página index.html é automaticamente atualizada.

# Alterando o componente app

- Em app.componente.html, apague todo o conteúdo/template original;
- Adicione a linha abaixo:

```
<h1>{{title}}</h1>
```

# Adicionando um estilo

- Cada componente tem seu próprio arquivo de estilos;
- Para estilos que se aplicam a todos os componentes, há o arquivo src/styles.css;
- Adicione os estilos abaixo nesse arquivo:

```
h1 {
  color: #369;
  font-family: Arial, Helvetica,
  sans-serif;
  font-size: 250%;
}
h2, h3 {
  color: #444;
  font-family: Arial, Helvetica,
  sans-serif;
  font-weight: lighter;
}

body {
  margin: 2em;
}
body, input[type="text"], button {
  color: #888;
  font-family: Cambria, Georgia;
}
/* everywhere else */
* {
  font-family: Arial, Helvetica,
  sans-serif;
}
```

# Criando um novo componente

- Para criar um novo componente, utilizamos o Angular CLI;
- Criaremos um componente chamado heroes:  
    >> ng generate componente heroes
- O CLI cria uma nova pasta src/app/heroes/, e gera 4 arquivos do componente:

```
D:\curso_angular\angular-tour-of-heroes>ng generate component heroes
CREATE src/app/heroes/heroes.component.html (25 bytes)
CREATE src/app/heroes/heroes.component.spec.ts (628 bytes)
CREATE src/app/heroes/heroes.component.ts (269 bytes)
CREATE src/app/heroes/heroes.component.css (0 bytes)
UPDATE src/app/app.module.ts (475 bytes)
```

# Novo componente

```
import { Component, OnInit } from '@angular/core';
```

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

```
export class HeroesComponent implements OnInit {
```

```
  constructor() { }
```

```
  ngOnInit() {  
  }
```

```
}
```

# Novo componente

- O método `ngOnInit`: faz parte do ciclo de vida de um componente;
- O Angular chama o `ngOnInit` logo após criar o componente;
- É utilizado para realizar alguma lógica de inicialização.



# Módulos

- O AppModule.ts é um modulo TypeScript:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

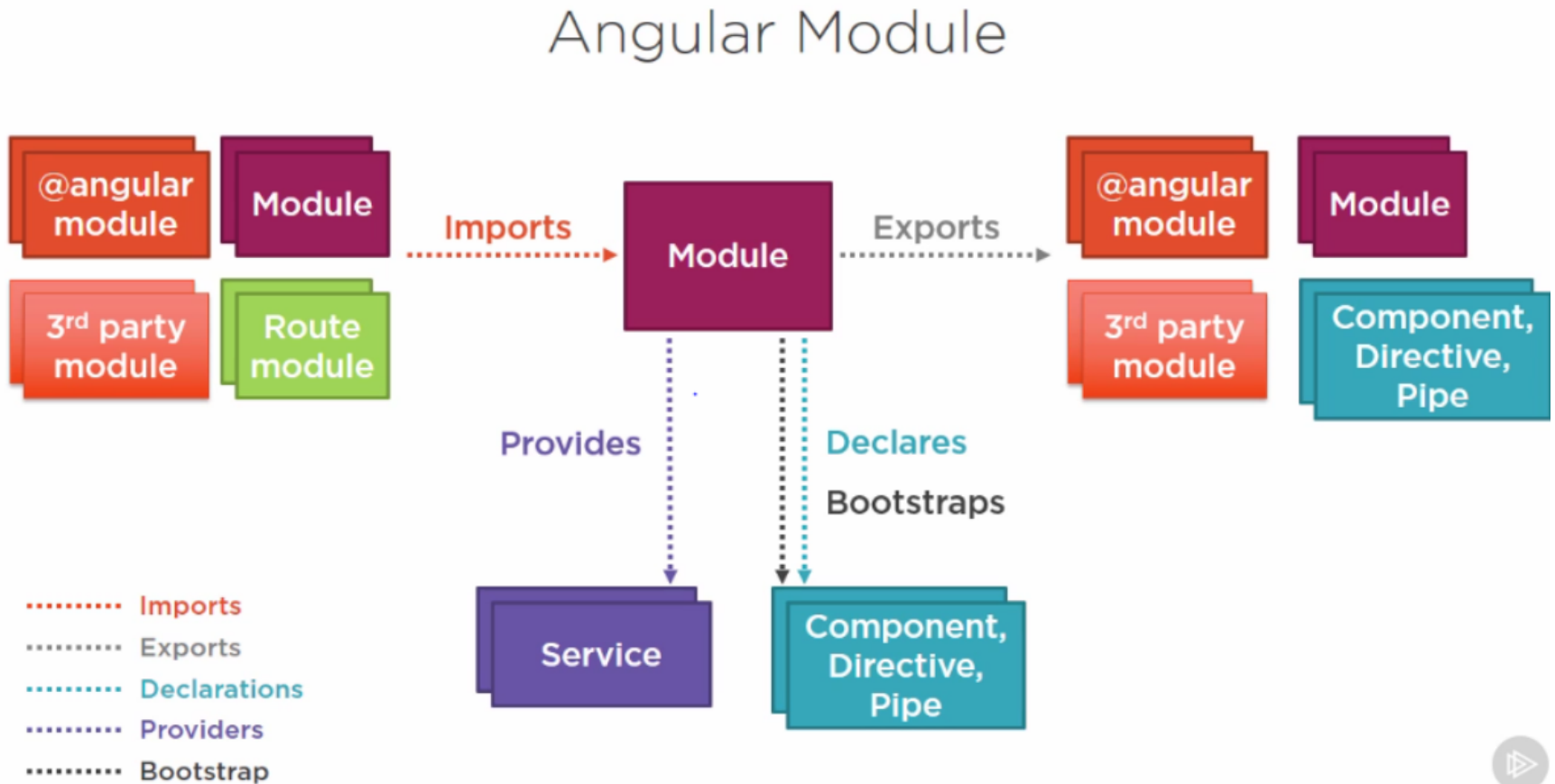
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HeroesComponent } from './heroes/heroes.component';

@NgModule({
  declarations: [ AppComponent, HeroesComponent ],
  imports: [ BrowserModule, AppRoutingModule ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Componentes e módulos

- O AppModule é também o módulo raiz da aplicação;
- Os componentes são organizados em módulos;
- Para que um componente seja acessível, ele deve estar declarado em um módulo;
- Os módulos fazem importações, exportações e declarações de componentes;
- Um modulo também é chamado de Barril;

# Componentes e módulos



<https://medium.com/@yonem9/angular-qu%C3%A9-son-los-m%C3%B3dulos-y-c%C3%B3mo-se-refactoriza-una-aplicaci%C3%B3n-9457550e8e9>

elydasilvamiranda@gmail.com

# Adicionando uma propriedade

```
//...
```

```
export class HeroesComponent implements OnInit {
```

```
    hero = 'Windstorm';
```

```
    constructor() { }
```

```
//...
```

```
}
```

# Exibindo o Hero

- Em hero.componente.html adicione a tag:  
{{hero}}
- Exibindo o componente em app.componente.html adicione a linha:

```
<h1>{{title}}</h1>
```

```
<app-heroes></app-heroes>
```



**Tour of Heroes**

Windstorm

# Criando uma classe

- No nosso exemplo, temos uma propriedade muito limitada para o componente;
- O ideal é termos uma classe de “modelo” que contenha não só o nome do herói;
- Podemos criar um arquivo hero.ts dentro da pasta /src/app a classe Hero:

```
export class Hero {  
  id: number;  
  name: string;  
}
```

# Utilizando a classe Hero

- Em TypeScript podemos “instanciar” uma classe como sendo um dicionário de propriedades.
- Dessa, forma, podemos instanciar uma classe Hero dentro do componente da seguinte forma:

```
//...
```

```
export class HeroesComponent implements OnInit {  
    hero: Hero = {  
        id: 1,  
        name: 'Windstorm'  
    };  
};
```

```
//...
```

```
}
```

# Atualizando a visualização

- Em heroes.componente.html:

```
<h2>{{hero.name}} Details</h2>
```

```
<div><span>id:</span>{{hero.id}}</div>
```

```
<div><span>name:</span>{{hero.name}}</div>
```

**Tour of Heroes**

Windstorm Details

id:1

name:Windstorm



# Angular

**Introdução, componentes e binding**

**Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)**