



Angular 2

Loiane Groner



github.com/loiane

loiane.com

loiane.training

10+ XP TI

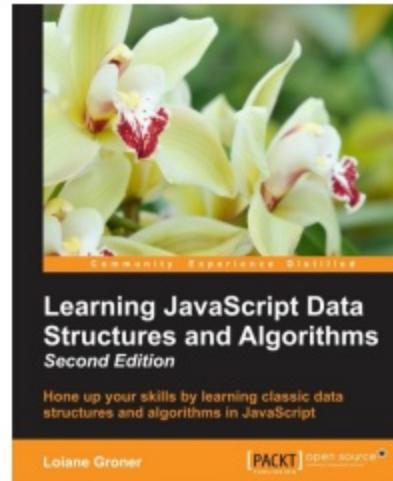
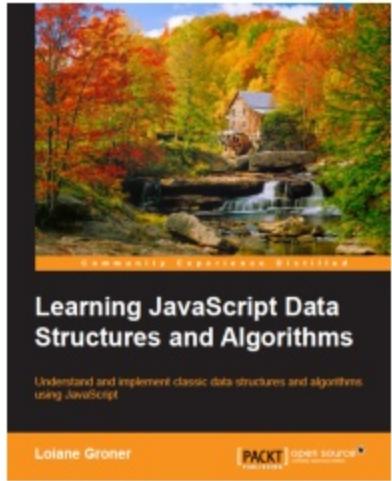
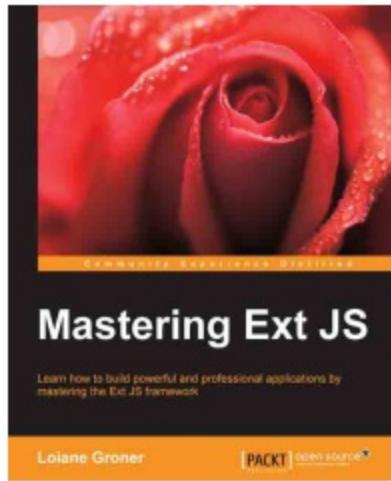
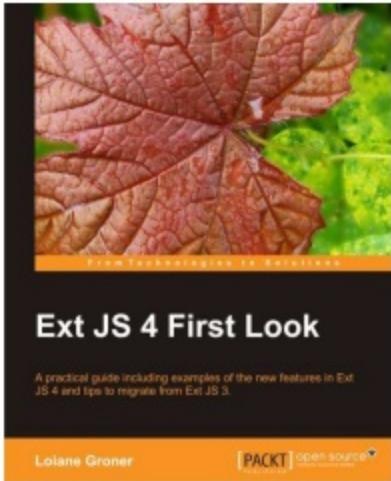
**Java, JavaScript, Sencha,
Phonegap/Ionic**

Blog: <http://loiane.com>

Cursos: <http://loiane.training>



Meus livros:



The logo consists of a red 3D hexagonal prism with the white text "01 Angular 2" on its faces.

01 Angular 2

Introdução

A faint silhouette of a city skyline with various buildings, including a water tower and a bridge, is visible at the bottom of the slide.

A AGENDA

- ▶ Introdução

A AGENDA

- ▶ Introdução
- ▶ Componentes e
Templates

A AGENDA

- ▶ Introdução
- ▶ Componentes e Templates
- ▶ Data binding

A AGENDA

- ▶ Introdução
- ▶ Componentes e Templates
- ▶ Data binding
- ▶ Diretivas



AGENDA

- ▶ Introdução
- ▶ Componentes e Templates
- ▶ Data binding
- ▶ Diretivas
- ▶ Serviços

A AGENDA

- ▶ Introdução
- ▶ Formulários
- ▶ Componentes e Templates
- ▶ Data binding
- ▶ Diretivas
- ▶ Serviços

A AGENDA

- ▶ Introdução
- ▶ Componentes e Templates
- ▶ Data binding
- ▶ Diretivas
- ▶ Serviços
- ▶ Formulários
- ▶ Roteamento

A AGENDA

- ▶ Introdução
- ▶ Componentes e Templates
- ▶ Data binding
- ▶ Diretivas
- ▶ Serviços
- ▶ Formulários
- ▶ Roteamento
- ▶ Integração com servidor



AGENDA

- ▶ Introdução
- ▶ Componentes e Templates
- ▶ Data binding
- ▶ Diretivas
- ▶ Serviços
- ▶ Formulários
- ▶ Roteamento
- ▶ Integração com servidor
- ▶ CRUD Mestre-Detalhe



REQUISITOS

- ▶ HTML / CSS/ JavaScript

- ▶ HTML / CSS/ JavaScript
- ▶ **NÃO** precisa saber AngularJS 1.x



ANGULAR 2

- ▶ <https://angular.io>



ANGULAR 2

- ▶ <https://angular.io>
- ▶ Parceria Google + Microsoft



ANGULAR 2

- ▶ <https://angular.io>
- ▶ Parceria Google + Microsoft
- ▶ Open Source (código no Github)



ANGULAR 2

- ▶ <https://angular.io>
- ▶ Parceria Google + Microsoft
- ▶ Open Source (código no Github)
- ▶ Atualmente na versão RC (Release Candidate)



ANGULAR 2

- ▶ <https://angular.io>
- ▶ Parceria Google + Microsoft
- ▶ Open Source (código no Github)
- ▶ Atualmente na versão RC (Release Candidate)
- ▶ Não é continuação da versão 1



ANGULAR 2

- ▶ <https://angular.io>
- ▶ Parceria Google + Microsoft
- ▶ Open Source (código no Github)
- ▶ Atualmente na versão RC (Release Candidate)
- ▶ Não é continuação da versão 1
- ▶ Foi reescrito



ANGULAR 2

- ▶ <https://angular.io>
- ▶ Parceria Google + Microsoft
- ▶ Open Source (código no Github)
- ▶ Atualmente na versão RC (Release Candidate)
- ▶ Não é continuação da versão 1
- ▶ Foi reescrito
- ▶ Uso de padrões web e Web Components



BLOCOS PRINCIPAIS

COMPONENTES

DIRETIVAS

ROTEAMENTO

SERVIÇOS

TEMPLATE

METADATA

DATA BINDING

INJEÇÃO
DEPENDENCIA



BLOCOS PRINCIPAIS

Componente



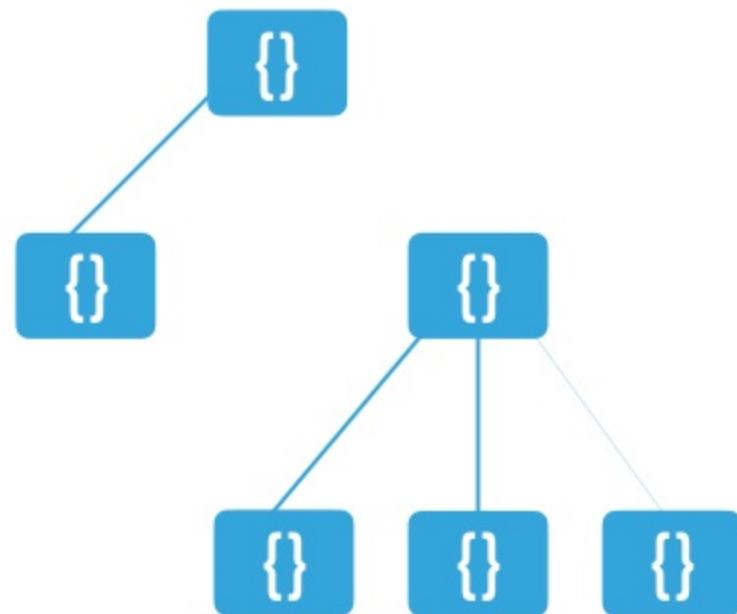
Encapsula:

- Template
- Metadata: processamento das classes
- Dado a ser mostrado na tela (Data Binding)
- Comportamento da VIEW



COMPONENTE

Componente Raiz (Root)



A COMPONENTE



Cabeçalho (Barra de navegação)



Barra
Lateral



Posts



COMPONENTE



Cabeçalho (Barra de navegação)



Barra
Lateral



Post



Post



Post



Post



COMPONENTE



Cabeçalho (Barra de navegação)



Barra
Lateral



Star Widget



Post



Post



Post

A COMPONENTE





COMPONENTE

Componente



Backend



Node.JS

Java

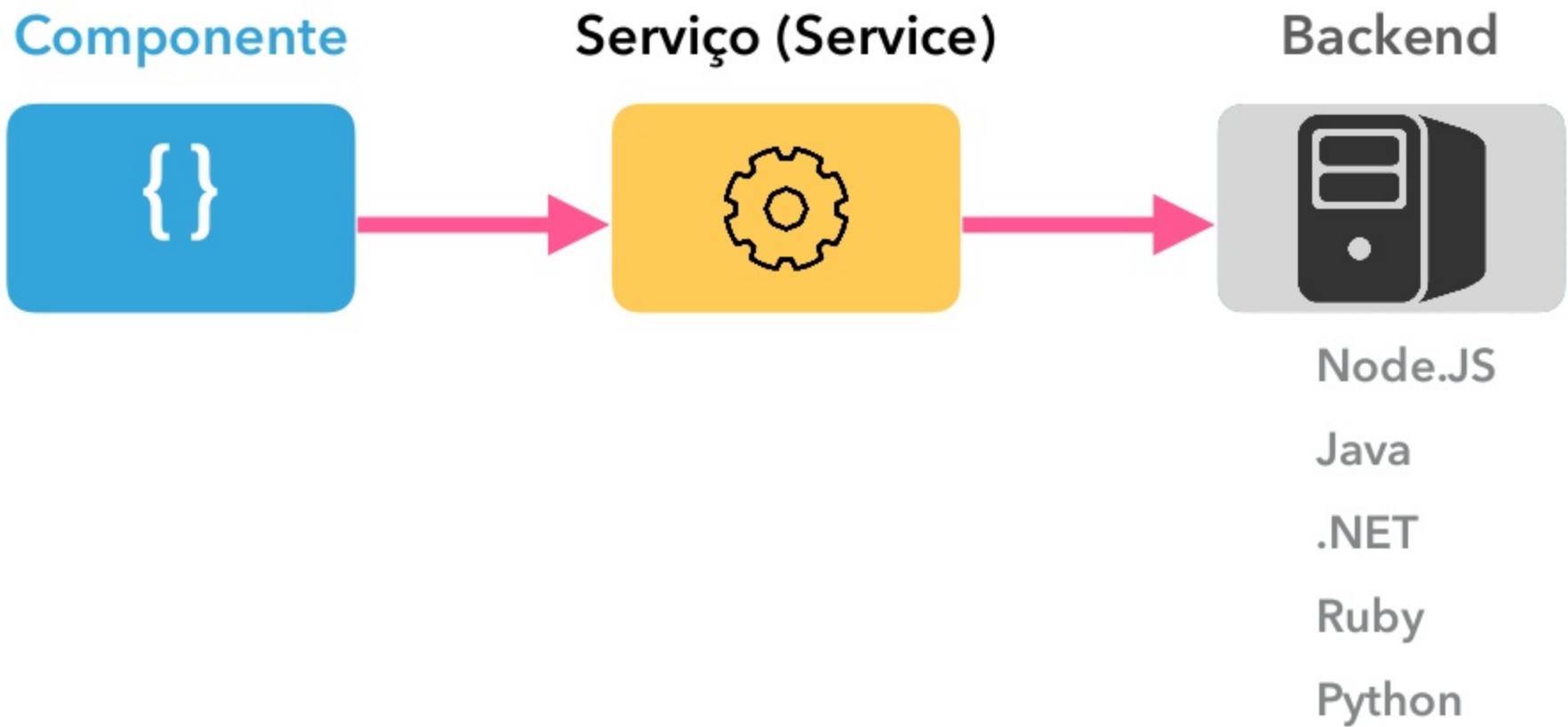
.NET

Ruby

Python



A SERVIÇO



A SERVIÇO





ROTEAMENTO

Router



Responsável pela navegação

A DIRETIVA

Diretiva

D

Responsável por modificar elementos DOM
e/ou seu comportamento

A DIRETIVA

pesquisar...

```
<input type="text" autoGrow>
```



Angular

```
<input type="text" autoGrow>
```



BLOCOS PRINCIPAIS

COMPONENTES

DIRETIVAS

ROTEAMENTO

SERVIÇOS

TEMPLATE

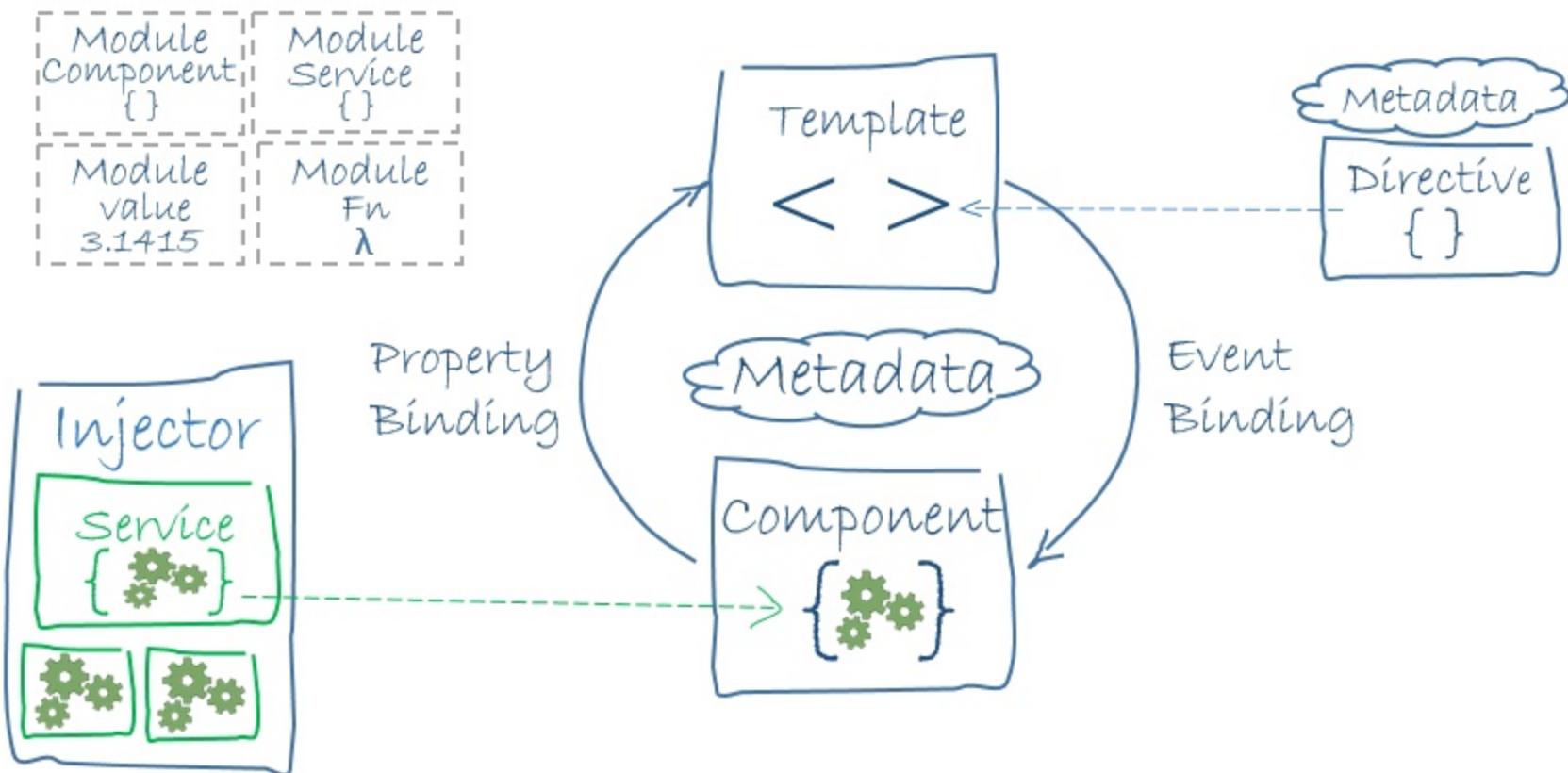
METADATA

DATA BINDING

INJEÇÃO
DEPENDENCIA



BLOCOS PRINCIPAIS





BLOCOS PRINCIPAIS

COMPONENTES

DIRETIVAS

ROTEAMENTO

SERVIÇOS

TEMPLATE

METADATA

DATA BINDING

INJEÇÃO
DEPENDENCIA



02 Angular 2

Ambiente de
Desenvolvimento



O QUE PRECISAMOS INSTALAR

- ▶ Node.JS



O QUE PRECISAMOS INSTALAR

- ▶ Node.JS
- ▶ NPM



O QUE PRECISAMOS INSTALAR

- ▶ Node.JS
- ▶ NPM
- ▶ TypeScript



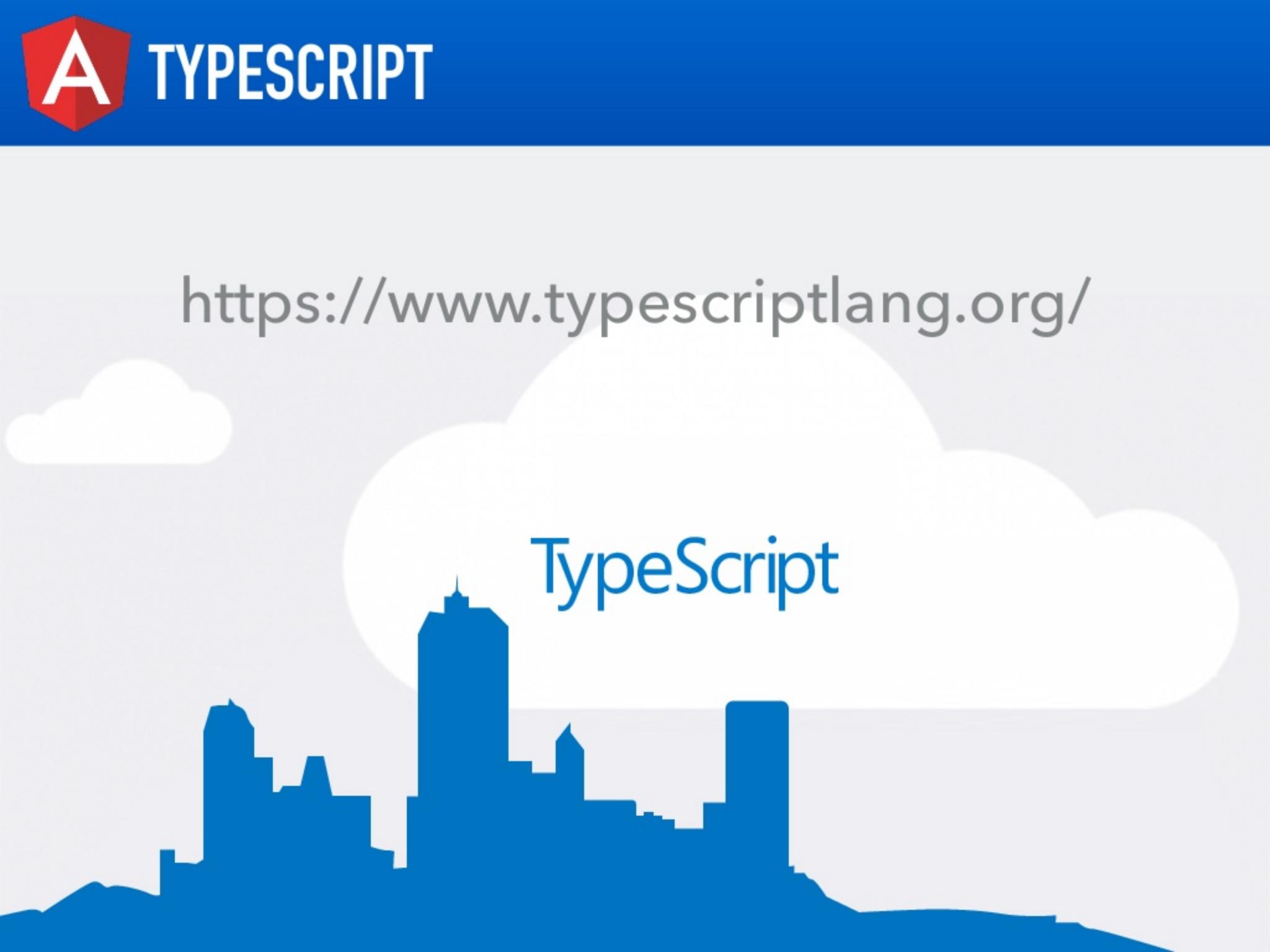
<https://nodejs.org>



TYPESCRIPT

<https://www.typescriptlang.org/>

TypeScript

A silhouette of a city skyline against a background of white clouds and a large, bright sun. The skyline features several buildings of varying heights, with one prominent skyscraper in the center.



```
$ npm install -g typescript
```

A TYPESCRIPT

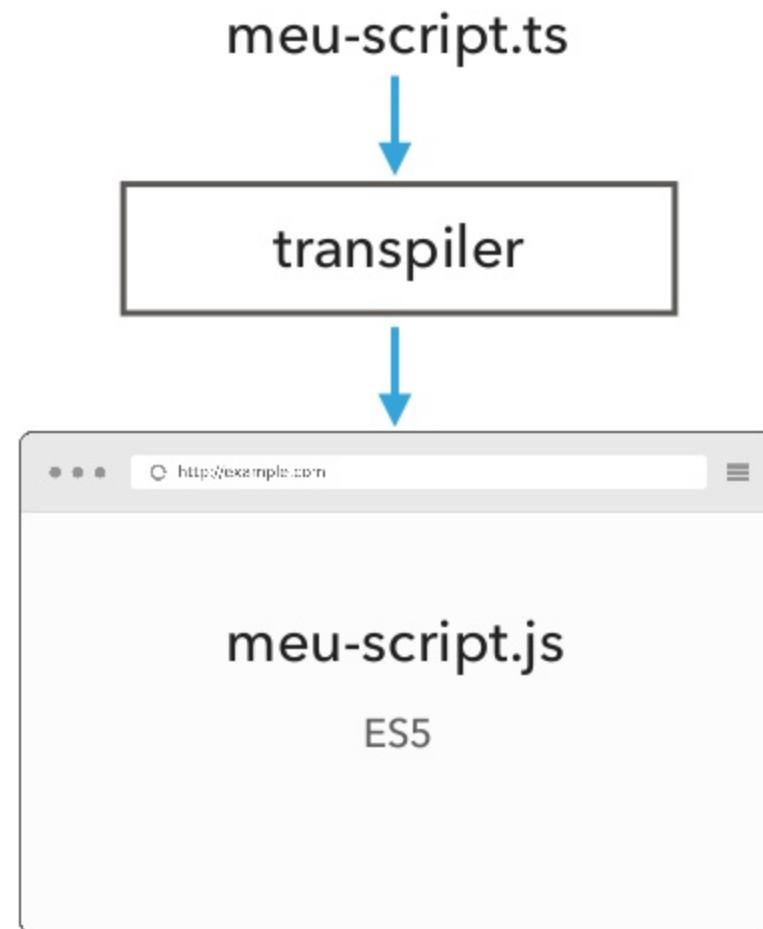
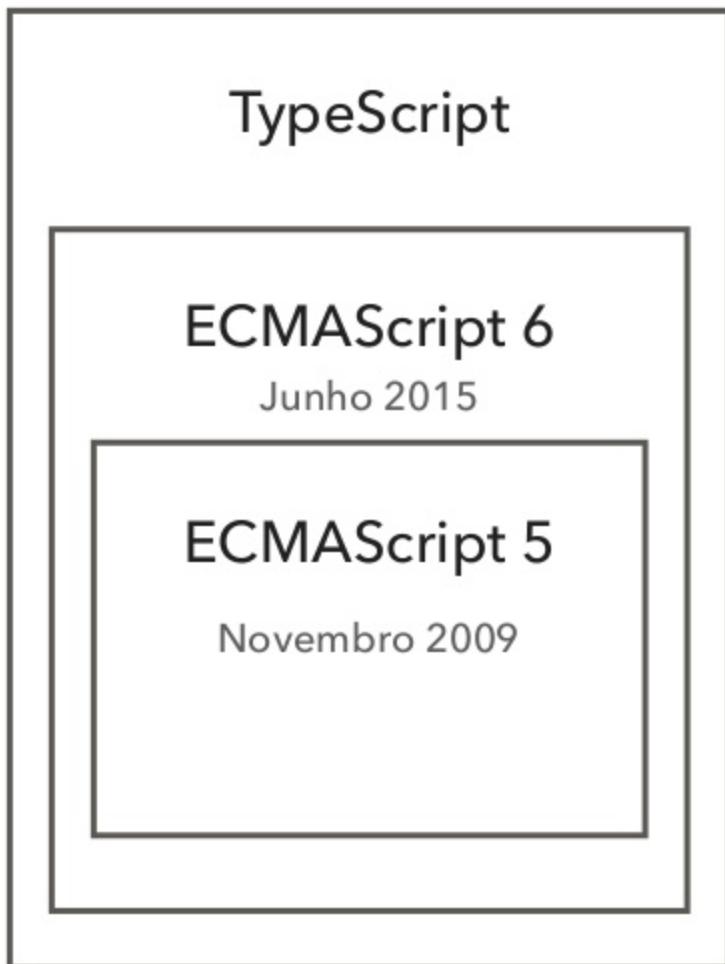


```
$ npm install -g typescript
```



```
$ sudo npm install -g typescript
```

SOMENTE MAC OU LINUX





LINGUAGEM

- ▶ TypeScript

- ▶ TypeScript

- ▶ Dart



LINGUAGEM

- ▶ TypeScript
- ▶ Dart
- ▶ ES 6 (ES 2015)



LINGUAGEM

- ▶ TypeScript
- ▶ Dart
- ▶ ES 6 (ES 2015)
- ▶ ES 5



Typings



The TypeScript Definition Manager

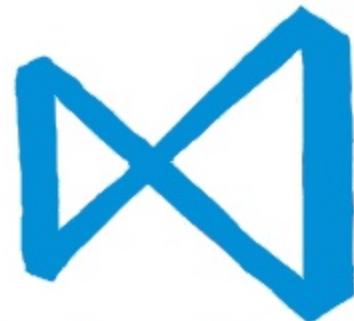
<https://github.com/typings/typings>



```
$ npm install typings --global
```



EDITOR DE TEXTO



Visual Studio Code



WebStorm



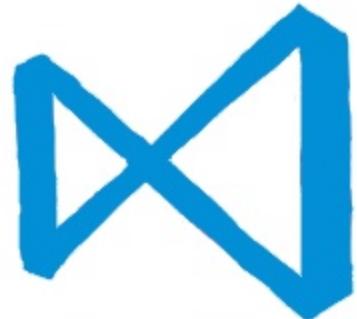
ATOM



Sublime Text



EDITOR DE TEXTO



Visual Studio Code

JÁ SUPORTA TS



WebStorm

JÁ SUPORTA TS



ATOM

[HTTPS://ATOM.IO/PACKAGES/ATOM-TYPESCRIPT](https://atom.io/packages/atom-typescript)

[HTTPS://GITHUB.COM/MICROSOFT/TYPESCRIPT-SUBLIME-PLUGIN](https://github.com/microsoft/typescript-sublime-plugin)



Sublime Text



03 Angular 2

Primeira app

Hello, World!



```
$ npm install -g angular-cli
```

Requer node >= 4

<https://github.com/angular/angular-cli>



```
ng new PROJECT_NAME
```

```
cd PROJECT_NAME
```

```
ng serve
```



ANGULAR SEED - SITE ANGULAR.IO

<https://angular.io/docs/ts/latest/quickstart.html>



SEED QUE VAMOS USAR PARA OS EXEMPLOS

>_

A large, white, stylized terminal prompt symbol (>_) is centered within a dark gray circle.

```
$ git clone https://github.com/loiane/angular-2-boilerplate.git
```

A PARA EXECUTAR



```
$ npm install
```

A PARA EXECUTAR

>_

```
$ npm install
```

>_

```
$ npm start
```

04 Angular 2

Entendendo o
projeto Hello, World!



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> .git
▼ app
  JS app.component.js
  LS app.component.js.map
  TS app.component.ts
  JS main.js
  LS main.js.map
  TS main.ts
▼ assets
  ▼ css
    SC app.css
> node_modules
> typings
  .gitignore
  LS index.html
  IN package.json
  MD README.md
  JS systemjs.config.js
  TC tsconfig.json
  TC typings.json
```



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> ⚡ .git
▼ └── app
    ├── app.component.js
    ├── app.component.js.map
    ├── app.component.ts
    ├── main.js
    ├── main.js.map
    ├── main.ts
    └── assets
        └── css
            └── app.css
> └── node_modules
> └── typings
    └── .gitignore
    └── index.html
    └── package.json
    └── README.md
    └── systemjs.config.js
    └── tsconfig.json
    └── typings.json
```

Código fonte TS + javascript gerado



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> ⚡ .git
▼ └── app
    ├── app.component.js
    ├── app.component.js.map
    ├── app.component.ts
    ├── main.js
    ├── main.js.map
    └── main.ts
└── assets
    └── css
        └── app.css
> └── node_modules
> └── typings
    └── .gitignore
    └── index.html
    └── package.json
    └── README.md
    └── systemjs.config.js
    └── tsconfig.json
    └── typings.json
```

Código fonte TS + javascript gerado

Arquivos CSS e assets do projeto



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> ⚙ .git
▼ └── app
    ├── app.component.js
    ├── app.component.js.map
    ├── app.component.ts
    ├── main.js
    ├── main.js.map
    └── main.ts
└── assets
    └── css
        └── app.css
> └── node_modules
> └── typings
    └── .gitignore
    └── index.html
    └── package.json
    └── README.md
    └── systemjs.config.js
    └── tsconfig.json
    └── typings.json
```

Código fonte TS + javascript gerado

Arquivos CSS e assets do projeto

Pacotes Node, instalados conforme
package.json

pacotes npm Angular 2 +
dependências + extras



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> ⚙ .git
▼ └── app
    ├── app.component.js
    ├── app.component.js.map
    ├── app.component.ts
    ├── main.js
    ├── main.js.map
    └── main.ts
└── assets
    └── css
        └── app.css
> └── node_modules
> └── typings
    └── .gitignore
    └── index.html
    └── package.json
    └── README.md
    └── systemjs.config.js
    └── tsconfig.json
    └── typings.json
```

Código fonte TS + javascript gerado

Arquivos CSS e assets do projeto

Pacotes Node, instalados conforme
package.json

index.html

pacotes npm Angular 2 +
dependências + extras



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> .git
< app
  app.component.js
  app.component.js.map
  app.component.ts
  main.js
  main.js.map
  main.ts
< assets
  < css
    app.css
  node_modules
  typings
  .gitignore
  index.html
  package.json
  README.md
  systemjs.config.js
  tsconfig.json
  typings.json
```

Código fonte TS + javascript gerado

Arquivos CSS e assets do projeto

Pacotes Node, instalados conforme
package.json

index.html

pacotes npm Angular 2 +
dependências + extras

configurações System.JS (módulos)



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> .git
< app
  app.component.js
  app.component.js.map
  app.component.ts
  main.js
  main.js.map
  main.ts
< assets
  < css
    app.css
  node_modules
  typings
  .gitignore
  index.html
  package.json
  README.md
  systemjs.config.js
  tsconfig.json
  typings.json
```

Código fonte TS + javascript gerado

Arquivos CSS e assets do projeto

Pacotes Node, instalados conforme
package.json

index.html

pacotes npm Angular 2 +
dependências + extras

configurações System.JS (módulos)

Configurações TypeScript



ESTRUTURA DO PROJETO - NOSSO EXEMPLO

```
angular-2-boilerplate
> .git
< app
  app.component.js
  app.component.js.map
  app.component.ts
  main.js
  main.js.map
  main.ts
< assets
  < css
    app.css
  node_modules
  typings
  .gitignore
  index.html
  package.json
  README.md
  systemjs.config.js
  tsconfig.json
  typings.json
```

Código fonte TS + javascript gerado

Arquivos CSS e assets do projeto

Pacotes Node, instalados conforme
package.json

index.html

pacotes npm Angular 2 +
dependências + extras

configurações System.JS (módulos)

Configurações TypeScript

Configurações Typings

A INDEX.HTML

```
<title>Angular 2 QuickStart</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap -->
<link rel="stylesheet" href="node_modules/bootstrap/dist/css/bootstrap.css">
  <!-- CSS -->
<link rel="stylesheet" href="assets/css/app.css">

<!-- 1. Load libraries -->
<!-- Polyfill(s) for older browsers -->
<script src="node_modules/es6-shim/es6-shim.min.js"></script>
<script src="node_modules/zone.js/dist/zone.js"></script>
<script src="node_modules/reflect-metadata/Reflect.js"></script>
<script src="node_modules/systemjs/dist/system.src.js"></script>
```

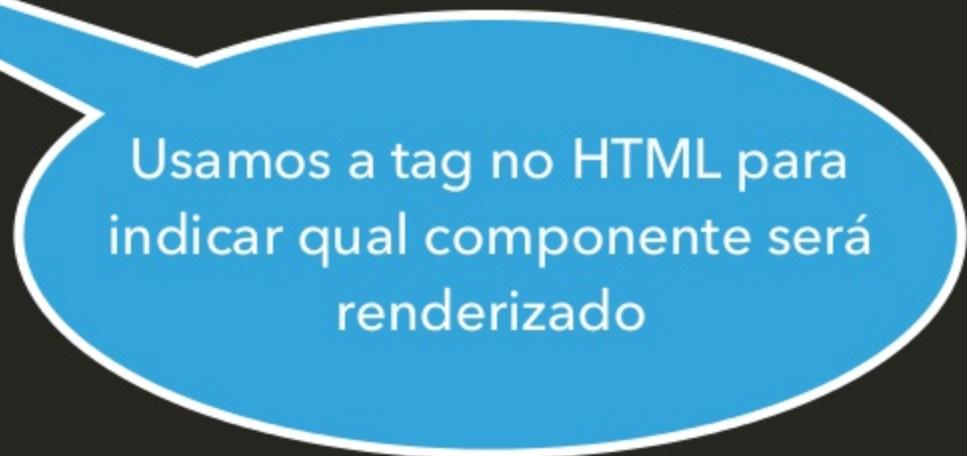
INDEX.HTML

```
<!-- 2. Configure SystemJS -->
<script src="systemjs.config.js"></script>
<script>
  System.import('app').catch(function(err){ console.error(err); });
</script>
```

```
<!-- 3. Renderizando a aplicação -->
<body>
  <my-app>Loading...</my-app>
</body>
```

A INDEX.HTML

```
<!-- 2. Renderizando a aplicação -->
<body>
  <my-app>Loading...</my-app>
</body>
```



Usamos a tag no HTML para indicar qual componente será renderizado



APP.COMPONENT.TS

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

@Component({
  selector: 'my-app',
  template: '<h1>My First Angular 2 App</h1>'
})
export class AppComponent {}
```



APP.COMPONENT.TS

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

@Component({
  selector: 'my-app',
  template: '<h1>My First Angular App</h1>'
})
export class AppComponent { }
```

Classe TypeScript



APP.COMPONENT.TS

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

@Component({
  selector: 'my-app',
  template: '<h1>My First Angular App</h1>'
})
export class AppComponent { }
```

Classe TypeScript

exporta classe para
que outras classes possam
importá-la



APP.COMPONENT.TS

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

@Component({
  selector: 'my-app',
  template: '<h1>My First Angular 2 App</h1>'
})
export class AppComponent {}
```

Decorator que
informa que essa classe é um
Componente



APP.COMPONENT.TS

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

@Component({
  selector: 'my-app',
  template: `
    <h1>Hello Angular</h1>
    <p>Welcome to my first Angular app!</p>
  `
})
export class AppComponent {}
```

E para usar esse decorator

precisamos importar a classe Component
do pacote do Angular



APP.COMPONENT.TS

```
import {Component} from '  
@Component({  
  selector: 'my-app',  
  template: '<h1>My First Angular 2 App</h1>'  
})  
export class AppComponent {}
```

Metadata: Nome da tag
que será criada



APP.COMPONENT.TS

```
import {Component} from '  
@Component({  
  selector: 'my-app',  
  template: '<h1>My First Angular 2 App</h1>'  
})  
export class AppComponent {}
```

Metadata:

Nome da tag (diretiva) que
será criada

Metadata: template HTML

 MAIN.TS

```
import {bootstrap}      from '@angular/platform-browser-dynamic'
import {AppComponent} from './app.component'

bootstrap(AppComponent);
```

A MAIN.TS

```
import {bootstrap}    from '@angular/platform-browser-dynamic'  
import {AppComponent} from './app.component'  
  
bootstrap(AppComponent);
```



inicializa a
app instanciando o
AppComponent

A MAIN.TS

```
import {bootstrap}      from '@angular/platform-browser-dynamic'  
import {AppComponent} from './app.component'  
  
bootstrap(AppComponent);
```

inicializa a
app instanciando o
AppComponent

Como
exportamos a classe, podemos
importar aqui

A MAIN.TS

```
import {bootstrap}      from '@angular/platform-browser-dynamic'  
import {AppComponent} from './app.component'  
  
bootstrap(AppComponent);
```

inicializa a app instanciando o AppComponent

não precisa do .ts

Como exportamos a classe, podemos importar aqui



DEPENDÊNCIAS - SYSTEMJS.CONFIG

```
// map tells the System loader where to look for things
var map = {
  'app':                      'app', // 'dist',
  'rxjs':                      'node_modules/rxjs',
  'angular2-in-memory-web-api': 'node_modules/angular2-in-memory-web-api',
  '@angular':                  'node_modules/@angular'
};

// packages tells the System loader how to load when no filename and/or no extension
var packages = {
  'app': { main: 'main.js', defaultExtension: 'js' },
  'rxjs': { defaultExtension: 'js' },
  'angular2-in-memory-web-api': { defaultExtension: 'js' }
};
```



DEPENDÊNCIAS - SYSTEMJS.CONFIG

Os arquivos .js compilados do
TypeScript que estão no diretório app
que serão executados no browser

```
// map tells the System loader where to look
var map = {
  'app': 'app', // 'dist',
  'rxjs': 'node_modules/rxjs',
  'angular2-in-memory-web-api': 'node_modules/angular2-in-memory-web-api',
  '@angular': 'node_modules/@angular'
};

// packages tells the System loader how to load when no filename and/or no extension
var packages = {
  'app': { main: 'main.js', defaultExtension: 'js' },
  'rxjs': { defaultExtension: 'js' },
  'angular2-in-memory-web-api': { defaultExtension: 'js' }
};
```



DEPENDÊNCIAS - SYSTEMJS.CONFIG

Os arquivos .js compilados do TypeScript que estão no diretório app que serão executados no browser

```
// map tells the System loader where to look
var map = {
  'app': 'app', // 'dist',
  'rxjs': 'node_modules/rxjs',
  'angular2-in-memory-web-api': 'node_modules/angular2-in-memory-web-api',
  '@angular': 'node_modules/@angular'
};

// packages tells the System loader how to load when no filename and/or no extension
var packages = {
  'app': { main: 'main.js', defaultExtension: 'js' },
  'rxjs': { defaultExtension: 'js' },
  'angular2-in-memory-web-api': { defaultExtension: 'js' }
};
```

app/main.js é responsável por inicializar a app



DEPENDÊNCIAS - SYSTEMJS.CONFIG

```
var packageNames = [
  '@angular/common',
  '@angular/compiler',
  '@angular/core',
  '@angular/http',
  '@angular/platform-browser',
  '@angular/platform-browser-dynamic',
  '@angular/router-deprecated',
  '@angular/testing',
  '@angular/upgrade',
];

// add package entries for angular packages in the form '@angular/common'
packageNames.forEach(function(pkgName) {
  packages[pkgName] = { main: 'index.js', defaultExtension: 'js' };
});

var config = {
  map: map,
  packages: packages
};

// filterSystemConfig - index.html's chance to modify config before we register it.
if (global.filterSystemConfig) { global.filterSystemConfig(config); }

System.config(config);
```



DEPENDÊNCIAS - SYSTEMJS.CONFIG

```
var packageNames = [
  '@angular/common',
  '@angular/compiler',
  '@angular/core',
  '@angular/http',
  '@angular/platform-browser',
  '@angular/platform-browser-dynamic',
  '@angular/router-deprecated',
  '@angular/testing',
  '@angular/upgrade',
];

// add package entries for angular packages in the form '@angular/common'
packageNames.forEach(function(pkgName) {
  packages[pkgName] = { main: 'index.js', defaultExtension: 'js' };
});

var config = {
  map: map,
  packages: packages
};

// filterSystemConfig - index.html's chance to modify config before we register it.
if (global.filterSystemConfig) { global.filterSystemConfig(config); }

System.config(config);
```

Carrega os arquivos do Angular 2



SISTEMA DE MÓDULOS

Ainda não existe um padrão, pode usar qualquer um:



SISTEMA DE MÓDULOS

Ainda não existe um padrão, pode usar qualquer um:

- ▶ System.JS



SISTEMA DE MÓDULOS

Ainda não existe um padrão, pode usar qualquer um:

- ▶ System.JS
- ▶ Webpack



SISTEMA DE MÓDULOS

Ainda não existe um padrão, pode usar qualquer um:

- ▶ System.JS
- ▶ Webpack
- ▶ Browserify



SISTEMA DE MÓDULOS

Ainda não existe um padrão, pode usar qualquer um:

- ▶ System.JS
- ▶ Webpack
- ▶ Browserify
- ▶ JSPM

A PARA EXECUTAR

>_

```
$ npm install
```

A PARA EXECUTAR

>_

```
$ npm install
```

>_

```
$ npm start
```

05 Angular 2

Criando um
Componente



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

meu-primeiro.component.ts

Palavras separadas por “-”

Ponto

“component” -> para indicar que é um componente

Ponto

ts -> extensão typescript



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

```
export class MeuPrimeiroComponent {}
```



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

```
export class MeuPrimeiroComponent {}
```



meu-primeiro.component.ts



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

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

export class MeuPrimeiroComponent {}
```



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

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

@Component({
})
export class MeuPrimeiroComponent {}
```



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

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

@Component({
  selector: 'meu-primeiro-component',
})
export class MeuPrimeiroComponent {}
```



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

```
import {Component} from '@angular/  
  @Component({  
    selector: 'meu-primeiro-component',  
  })  
export class MeuPrimeiroComponent {}
```



nome da DIV a ser criada



MEU-PRIMEIRO.COMPONENT.TS

Crie um arquivo dentro de app/hello chamado meu-primeiro.component.ts

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

@Component({
  selector: 'meu-primeiro-component',
  template: '<h2>Meu primeiro componente Angular 2!</h2>'
})
export class MeuPrimeiroComponent {}
```

A APP.COMPONENT.TS - USANDO O COMPONENTE

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

@Component({
  selector: 'my-app',
  template: `
    <h1>Angular 2 Boilerplate</h1>
    <p>Hello World!</p>
    <meu-primeiro-component></meu-primeiro-component>
  `
})
export class AppComponent {
```

A APP.COMPONENT.TS - USANDO O COMPONENTE



Angular 2 Boilerplate

Hello World!

```
<!DOCTYPE >
<html>
  <head>...</head>
  <body>
    <script id="__bs_script__">...</script>
    <script async src="/browser-sync/browser-sync-client.2.12.5.js"></script>
    <my-app>
      <h1>Angular 2 Boilerplate</h1>
      <p>Hello World!</p>
      ... <meu-primeiro-component></meu-primeiro-component>
    </my-app>
    <script>...</script>
  </body>
</html>
```

A APP.COMPONENT.TS - USANDO O COMPONENTE



Angular 2 Boilerplate

Hello World!

Cadê o componente????

```
<!DOCTYPE >
<html>
  <head>...</head>
  <body>
    <script id="__bs_script__">...</script>
    <script async src="/browser-sync/browser-sync-client.2.12.5.js"></script>
    <my-app>
      <h1>Angular 2 Boilerplate</h1>
      <p>Hello World!</p>
      ... <meu-primeiro-component></meu-primeiro-component>
    </my-app>
    <script>...</script>
  </body>
</html>
```



APP.COMPONENT.TS - USANDO O COMPONENTE

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

@Component({
  selector: 'my-app',
  template: `
    <h1>Angular 2 Boilerplate</h1>
    <p>Hello World!</p>
    <meu-primeiro-component></meu-primeiro-component>
  `,
  directives: [MeuPrimeiroComponent]
})
export class AppComponent {

}
```

A APP.COMPONENT.TS - USANDO O COMPONENTE



Angular 2 Boilerplate

Hello World!

Meu primeiro componente Angular 2!

```
<!DOCTYPE>
<html>
  <head>...</head>
  <body>
    <script id="__bs_script__">...</script>
    <script async src="/browser-sync/browser-sync-client.2.12.5.js"></script>
    <my-app>
      <h1>Angular 2 Boilerplate</h1>
      <p>Hello World!</p>
      ... <meu-primeiro-component>
        <h2>Meu primeiro componente Angular 2!</h2>
      </meu-primeiro-component>
    </my-app>
    <script>...</script>
  </body>
<html> <body> <my-app> <meu-primeiro-component>
```

Styles Event Listeners DOM Breakpoints Properties Sencha Properties Sencha Component



AO CRIAR UM COMPONENTE NO ANGULAR – CIDAR

- ▶ Class (Criar a classe)
- ▶ Import
- ▶ Decorate (Decorar a classe)
- ▶ Alterar
- ▶ Repetir

The Angular 2 logo, which consists of a red 3D hexagonal prism with the number "06" in white on its front face.

06 Angular 2

A faint silhouette of a city skyline with various buildings, including a water tower and a bridge, visible against a dark blue background.

Templates



TEMPLATES - USANDO INTERPOLATION

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

@Component({
  selector: 'cursos',
  template: `
    <p>Olá, meu nome é {{ nome }}</p>
  `
})
export class CursosComponent {
  nome = 'Loiane';
}
```



TEMPLATES - USANDO INTERPOLATION

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

@Component({
  selector: 'cursos', Interpolation
  template: `
    <p>Olá, meu nome é {{ nome }}</p>
  `
})
export class CursosComponent {
  nome = 'Loiane';
}
```



TEMPLATES - USANDO INTERPOLATION

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

@Component({
  selector: 'cursos', interpolation
  template: `
    <p>Olá, meu nome é {{ nome }}</p>
  `
})
export class CursosComponent {
  nome = 'Loiane';
}
```



TEMPLATES - USANDO INTERPOLATION



Angular 2 Boilerplate

Hello World!

Meu primeiro componente Angular 2!

Olá, meu nome é Loiane



TEMPLATES - DIRETIVAS

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

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos = ['Java', 'Angular 2', 'Estrutura de Dados'];
}
```



TEMPLATES - DIRETIVAS

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

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos = ['Java', 'Angular 2', 'Estrutura de Dados'];
}
```



TEMPLATES - DIRETIVAS

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

@Component({
  selector: 'cursos',
  template: `
    

### Cursos {{ tituloCursos }}


    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos = ['Java', 'Angular 2', 'Estrutura de Dados'];
}
```

Diretiva ngFor

Itera uma coleção/array



TEMPLATES - DIRETIVAS

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

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos = ['Java', 'Angular 2', 'Estrutura de Dados'];
}
```

Diretiva ngFor

Itera uma coleção/array

Variável local curso criada



TEMPLATES - DIRETIVAS

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

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos = ['Java', 'Angular 2', 'Estrutura de Dados'];
}
```

Diretiva **ngFor**

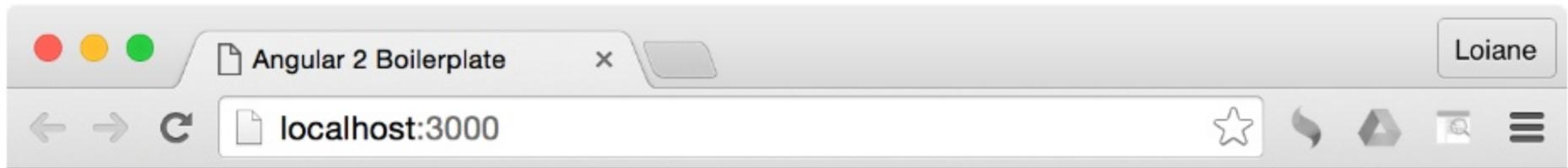
Itera uma coleção/array

Variável local **curso** criada

Interpolation da variável
local **curso**



TEMPLATES - DIRETIVAS



Angular 2 Boilerplate

Hello World!

Meu primeiro componente Angular 2!

Cursos loiane.training

- Java
- Angular 2
- Estrutura de Dados



TEMPLATES - DIRETIVAS

```
@Component({
  selector: 'cursos-list',
  /*template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `*/
  templateUrl: 'cursos.component.html',
  providers: [CursosService]
})
export class CursosComponent { }
```



TEMPLATES EM ARQUIVO SEPARADO

```
▼ └── app
    └── cursos
        ├── cursos.component.html
        ├── cursos.component.ts
        └── cursos.service.ts
    └── hello
        └── meu-primeiro.component.ts
    └── shared
        ├── auto-grow.directive.ts
        ├── app.component.ts
        └── main.ts
```

Guia Angular 2

<https://angular.io/styleguide>



07 Angular 2

Serviços (Services) e
Injeção de
dependência (DI)



SEPARANDO A LÓGICA DE NEGÓCIO DA VIEW

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

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos = ['Java', 'Angular 2', 'Estrutura de Dados'];
}
```

A SEPARANDO A LÓGICA DE NEGÓCIO DA VIEW

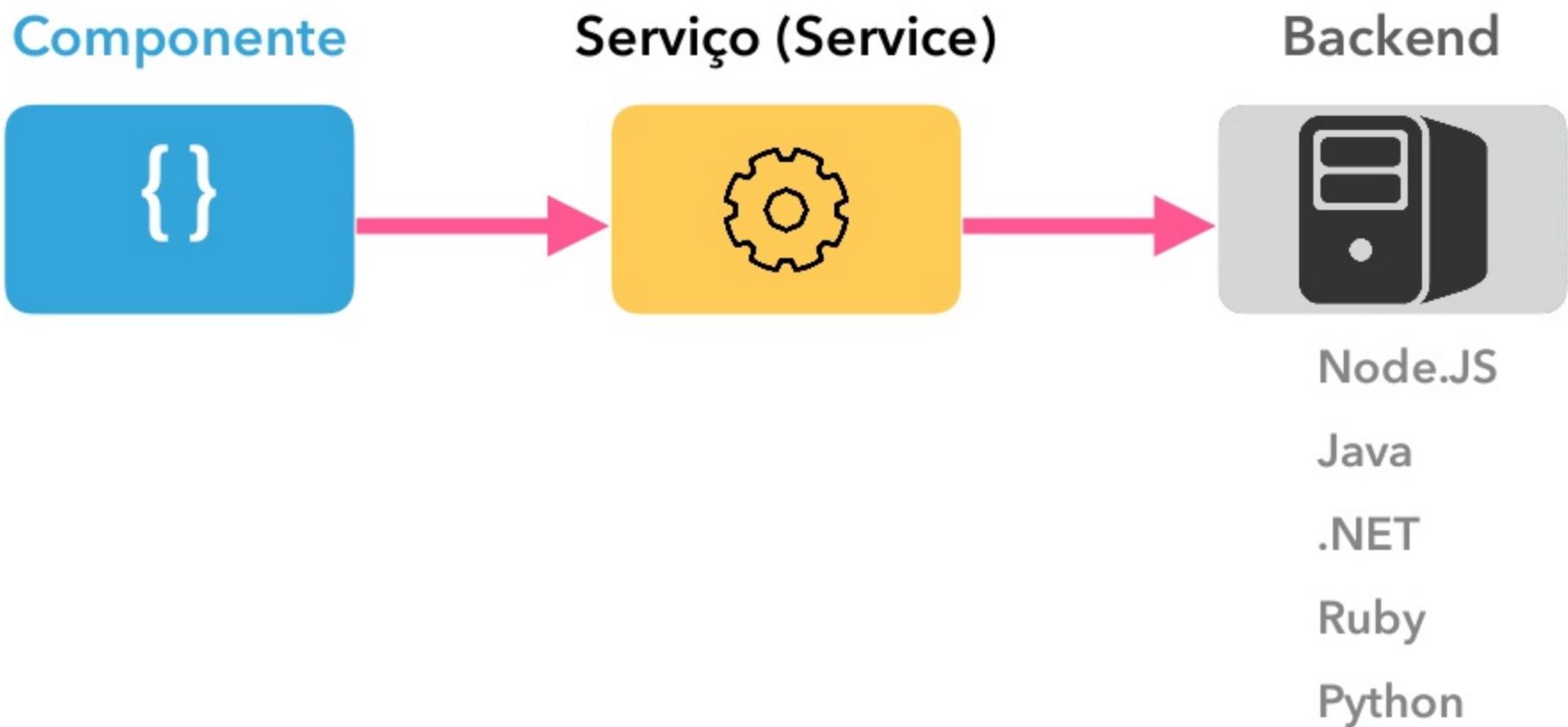
```
import {Component} from '@angular/core';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos = ['Java', 'Angular 2', 'Estrutura de Dados'];
}
```

Suponha que vamos obter
esses valores de um servidor.

Melhor prática seria separar a
lógica de negócio da VIEW

A ARQUITETURA





CRIANDO UM SERVIÇO (SERVICE)

Crie um arquivo dentro de app/cursos chamado cursos.service.ts

Repare na convenção de nomes

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

@Injectable()
export class CursosService {

  getCursos(): string[] {
    return ['Java', 'Angular 2', 'Estrutura de Dados'];
  }
}
```



CRIANDO UM SERVIÇO (SERVICE)

Crie um arquivo dentro de app/cursos chamado **cursos.service.ts**

Repare na convenção de nomes

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

@Injectable()
export class CursosService {
  getCursos(): string[] {
    return ['Java', 'Angular 2', 'Estrutura de Dados'];
  }
}
```



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {

  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {

  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```



import
do serviço

A blue callout bubble with a black arrow points from the bottom right towards the line of code 'import {CursosService} from './cursos.service';'. The text 'import do serviço' is contained within the bubble.



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```

import
do serviço

atributo local a ser
populado



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```

import
do serviço

atributo local a ser
populado

Injeção de
Dependência



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `
})
export class CursosComponent {
  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```

import
do serviço

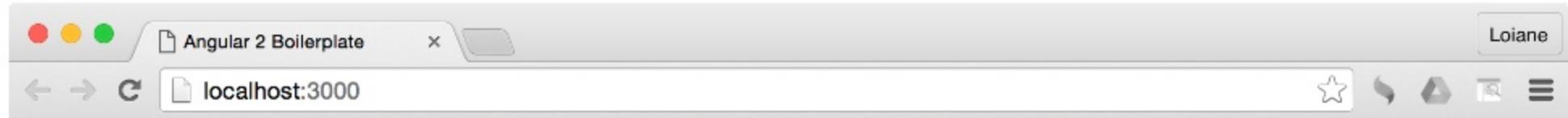
atributo local a ser
populado

Injeção de
Dependência

Chamada do
método do serviço



INJEÇÃO DE DEPENDÊNCIA



Angular 2 Boilerplate

Hello World!

Meu primeiro componente Angular 2!

Oops!

Elements Console Sources Network Timeline Profiles Resources Security Audits Sencha 17

top ▾ Preserve log

EXCEPTION: Error in ./AppComponent class AppComponent - inline template:4:8 angular2.js:25654

▶ EXCEPTION: Error in ./AppComponent class AppComponent - inline template:4:8 angular2.js:25644

▶ ORIGINAL EXCEPTION: No provider for CursosService! angular2.js:25644

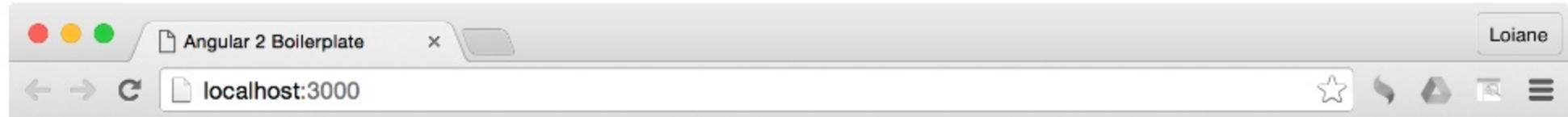
▶ ORIGINAL STACKTRACE: angular2.js:25644

▶ Error: DI Exception angular2.js:25644

```
at NoProviderError.BaseException [as constructor] (angular2.js:5496)
at NoProviderError.AbstractProviderError [as constructor] (angular2.js:1600)
at new NoProviderError (angular2.js:1624)
at ReflectiveInjector_.throwOrNull (angular2.js:6966)
at ReflectiveInjector_.getByKeyDefault (angular2.js:6990)
at ReflectiveInjector_.getByKey (angular2.js:6959)
at ReflectiveInjector_.get (angular2.js:6772)
at ElementInjector.get (angular2.js:15755)
at ApplyView.View.AppComponent0.createInternal (AppComponent.template.js:13)
```



INJEÇÃO DE DEPENDÊNCIA



Angular 2 Boilerplate

Hello World!

Meu primeiro componente Angular 2!

Oops!

Elements Console Console Sources Network Timeline Profiles Resources Security Audits Sencha 17

top ▾ Preserve log

EXCEPTION: Error in ./AppComponent class AppComponent - inline template:4:8

▶ EXCEPTION: Error in ./AppComponent class AppComponent - inline template:4:8

▶ ORIGINAL EXCEPTION No provider for CursosService!

▶ ORIGINAL STACKTRACE

Error: DI Exception

```
at NoProviderError.BaseException [as constructor] (angular2.js:5496)
at NoProviderError.AbstractProviderError [as constructor] (angular2.js:1600)
at new NoProviderError (angular2.js:1624)
at ReflectiveInjector_.throwOrNull (angular2.js:6966)
at ReflectiveInjector_.getByKeyDefault (angular2.js:6990)
at ReflectiveInjector_.getByKey (angular2.js:6959)
at ReflectiveInjector_.get (angular2.js:6772)
at ElementInjector.get (angular2.js:15755)
at ApplyView.View.AppComponent0.createInternal (AppComponent.template.js:143)
```

angular2.js:25654
angular2.js:25644
angular2.js:25644
angular2.js:25644
angular2.js:25644
angular2.js:25644



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `,
  providers: [CursosService]
})
export class CursosComponent {

  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `,
  providers: [CursosService]
})
export class CursosComponent {

  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```

Precisamos informar ao Angular que queremos que injete a classe CursosService nesse componente



USANDO INJEÇÃO DE DEPENDÊNCIA

```
import {Component} from '@angular/core';
import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  template: `
    <h3>Cursos {{ tituloCursos }}</h3>
    <ul>
      <li *ngFor="let curso of cursos">
        {{ curso }}
      </li>
    </ul>
  `,
  providers: [CursosService]
})
export class CursosComponent {

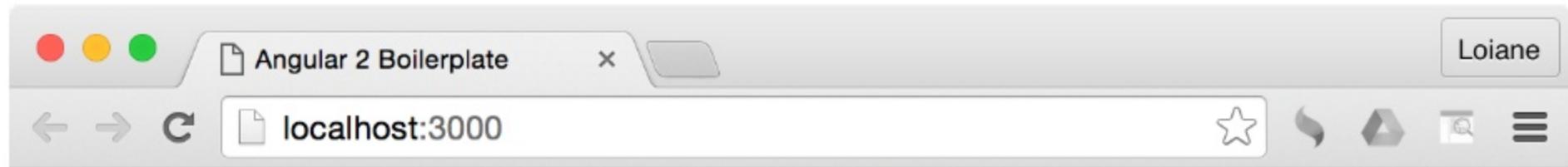
  tituloCursos = 'loiane.training';
  cursos;

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCursos();
  }
}
```

Precisamos informar ao Angular que queremos que injete a classe CursosService nesse componente

O provider cria/provém (fornece) uma instância da classe

A INJEÇÃO DE DEPENDÊNCIA



Angular 2 Boilerplate

Hello World!

Meu primeiro componente Angular 2!

Cursos loiane.training

- Java
- Angular 2
- Estrutura de Dados

The Angular logo is a red 3D hexagon with the number '08' in white on its front face.

08 Angular 2

Dica Produtividade:
Code Snippets



SNIPPETS PARA CRIAÇÃO DO ESQUELETO DA CLASSE ANGULAR2

- ▶ VS Code: <https://github.com/johnpapa/vscode-angular2-snippets>
- ▶ Atom: <https://atom.io/packages/angular-2-typeScript-snippets>



09 Angular 2

Property Binding

(Binding de propriedades)

+ Interpolation



DATA BINDING

<TEMPLATE>

{COMPONENT}



DATA BINDING

<TEMPLATE>

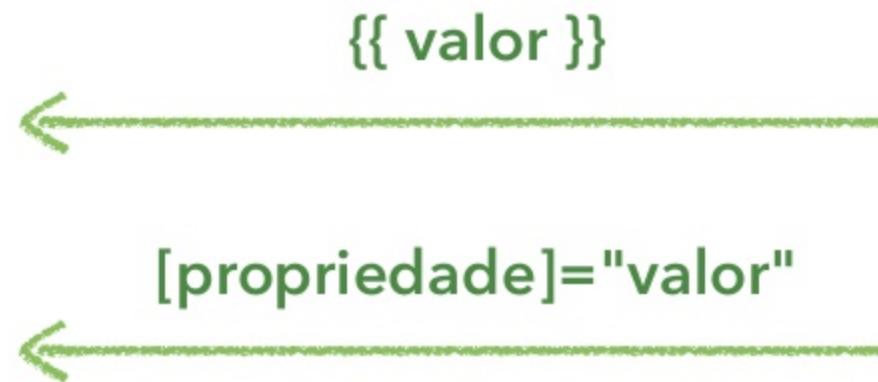
{{ valor }}

{COMPONENT}



DATA BINDING

<TEMPLATE>

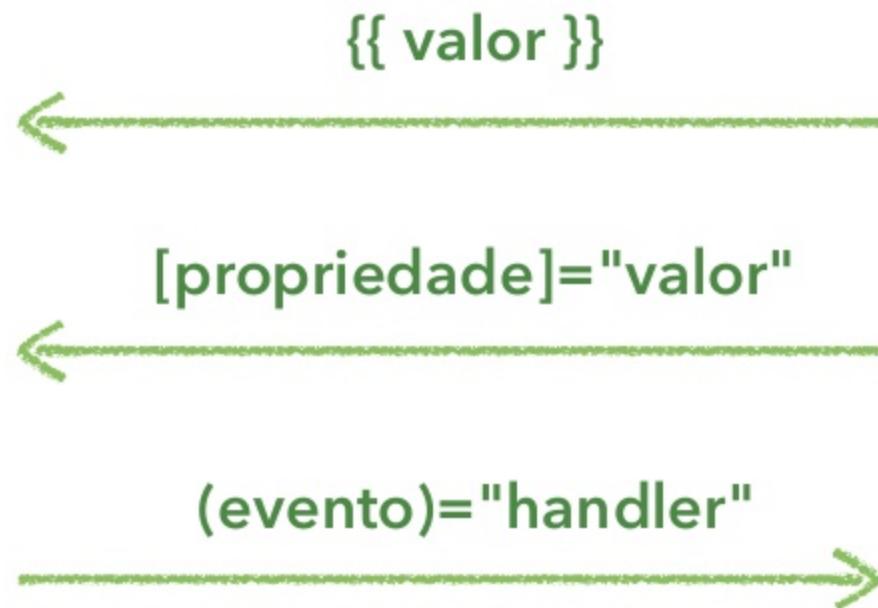


{COMPONENT}



DATA BINDING

<TEMPLATE>



{COMPONENT}



DATA BINDING

<TEMPLATE>

`{{ valor }}`

`[propriedade]="valor"`

`(evento)="handler"`

`[(ngModel)]="propriedade"`

{COMPONENT}

A INTERPOLATION

```
export class DataBindingComponent {  
  
  url = 'http://loiane.com';  
  urlImg = 'http://lorempixel.com/400/200/nature/';  
}
```

A INTERPOLATION

```
export class DataBindingComponent {  
  
    url = 'http://loiane.com';  
    urlImg = 'http://lorempixel.com/400/200/nature/';  
}  
  
<article>  
    <h3>Interpolation</h3>  
    <p>String renderizada com Interpolation: {{ url }}</p>  
</article>
```

A INTERPOLATION

```
export class DataBindingComponent {  
  url = 'http://loiane.com';  
  urlImg = 'http://lorempixel.com/400/200/nature/';  
}  
  
<article>  
  <h3>Interpolation</h3>  
  <p>String renderizada com Interpolation: {{ url }}</p>  
</article>
```



INTERPOLATION

- ▶ Valor do Component para o Template



INTERPOLATION

- ▶ Valor do Component para o Template
- ▶ É uma forma de property binding



INTERPOLATION

- ▶ Valor do Component para o Template
- ▶ É uma forma de property binding
- ▶ `<h3>{{ url }}</h3>`



INTERPOLATION

- ▶ Valor do Component para o Template
- ▶ É uma forma de property binding
- ▶ `<h3>{{ url }}</h3>`
- ▶ Permite uso de algumas expressões, como:



INTERPOLATION

- ▶ Valor do Component para o Template
- ▶ É uma forma de property binding
- ▶ `<h3>{{ url }}</h3>`
- ▶ Permite uso de algumas expressões, como:
 - ▶ `{{ 1 + 1 }}`



INTERPOLATION

- ▶ Valor do Component para o Template
- ▶ É uma forma de property binding
- ▶ `<h3>{{ url }}</h3>`
- ▶ Permite uso de algumas expressões, como:
 - ▶ `{{ 1 + 1 }}`
 - ▶ `{{ getValor() }}`



INTERPOLATION

- ▶ Valor do Component para o Template
- ▶ É uma forma de property binding
- ▶ `<h3>{{ url }}</h3>`
- ▶ Permite uso de algumas expressões, como:
 - ▶ `{{ 1 + 1 }}`
 - ▶ `{{ getValor() }}`
 - ▶ `{{ minhaVar && minhaVar2 }}`

A INTERPOLATION

```
<h3>{{ url }}</h3>
```

==

```
<h3 [textContent]="url"></h3>
```



PROPERTY BINDING

- ▶ Valor do Component para o Template



PROPERTY BINDING

- ▶ Valor do Component para o Template
- ▶ Usa-se colchetes



PROPERTY BINDING

- ▶ Valor do Component para o Template
- ▶ Usa-se colchetes
- ▶ Formato padrão é através de bind-
nomePropriedade



PROPERTY BINDING

- ▶ Valor do Component para o Template
- ▶ Usa-se colchetes
- ▶ Formato padrão é através de bind-
nomePropriedade
- ▶ Quando não existe uma propriedade no elemento,
usa-se [attr.colspan]



PROPERTY BINDING

```
<h3>Property binding</h3>

<img [src]="urlImg" />

```



PROPERTY BINDING

```
<h3>Property binding</h3>

<img [src]="urlImg" />

```

Todas as imagens tem o
mesmo resultado



10 Angular 2

Class e Style Binding (Binding de classes e estilos)



CLASS E STYLE BINDING - EXEMPLO 01

Selecione uma classe:

```
<select #class (change)="0">
  <option value="alert-success">Sucesso</option>
  <option value="alert-info">Info</option>
  <option value="alert-warning">Warning</option>
  <option value="alert-danger">Erro</option>
</select>
<br><br>
<div class="alert {{class.value}}" role="alert">
  Texto colorido conforme opção selecionada.
</div>
```



CLASS E STYLE BINDING - EXEMPLO 01

Variável local
chamada class

Selecione uma classe:

```
<select #class (change)="0">
  <option value="alert-success">Sucesso</option>
  <option value="alert-info">Info</option>
  <option value="alert-warning">Warning</option>
  <option value="alert-danger">Erro</option>
</select>
<br><br>
<div class="alert {{class.value}}" role="alert">
  Texto colorido conforme opção selecionada.
</div>
```



CLASS E STYLE BINDING - EXEMPLO 01

Selecione uma classe:

```
<select #class (change)="0">
  <option value="alert-success">Sucesso</option>
  <option value="alert-info">Info</option>
  <option value="alert-warning">Warning</option>
  <option value="alert-danger">Danger</option>
</select>
<br><br>
<div class="alert {{class.value}}" role="alert">
  Texto colorido conforme opção selecionada.
</div>
```

Variável local
chamada class

Property binding



CLASS E STYLE BINDING - EXEMPLO 02

```
<div class="alert" role="alert"
  [class.alert-success]="class.value == 'alert-success'" >
  Sucesso.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-info]="class.value == 'alert-info'" >
  Info.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-warning]="class.value == 'alert-warning'" >
  Warning.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-danger]="class.value == 'alert-danger'" >
  Erro.
</div>
```



CLASS E STYLE BINDING - EXEMPLO 02

```
<div class="alert" role="alert"
  [class.alert-success]="class.value == 'alert-success'" >
  Sucesso.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-info]="class.value == 'alert-info'" >
  Info.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-warning]="class.value == 'alert-warning'" >
  Warning.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-danger]="class.value == 'alert-danger'" >
  Erro.
</div>
```



CLASS E STYLE BINDING - EXEMPLO 02

```
<div class="alert" role="alert"
  [class.alert-success]="class.value == 'alert-success'" >
  Sucesso.
</div>
```

Só recebe a
class caso a
expressão seja
verdadeira

```
<div class="alert" role="alert"
  [class.alert-info]="class.value == 'alert-info'" >
  Info.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-warning]="class.value == 'alert-warning'" >
  Warning.
</div>
```

```
<div class="alert" role="alert"
  [class.alert-danger]="class.value == 'alert-danger'" >
  Erro.
</div>
```



CLASS E STYLE BINDING - EXEMPLO 03

```
<div [style.display]="class.value == 'alert-danger' ? 'block' : 'none'"  
      class="alert alert-danger" role="alert">  
    Esse texto só aparece em caso de erro!  
</div>
```



CLASS E STYLE BINDING - EXEMPLO 03

```
<div [style.display]="class.value == 'alert-danger' ? 'block' : 'none'"  
      class="alert alert-danger" role="alert">  
    Esse texto só aparece em caso de erro!  
</div>
```



Mostra o estilo
ou não?



CLASS E STYLE BINDING - EXEMPLO 03

Expressão com o valor da variável

```
<div [style.display]="class.value == 'alert-danger' ? 'block' : 'none'"  
      class="alert alert-danger" role="alert">  
    Esse texto só aparece em caso de erro!  
</div>
```

Mostra o estilo ou não?



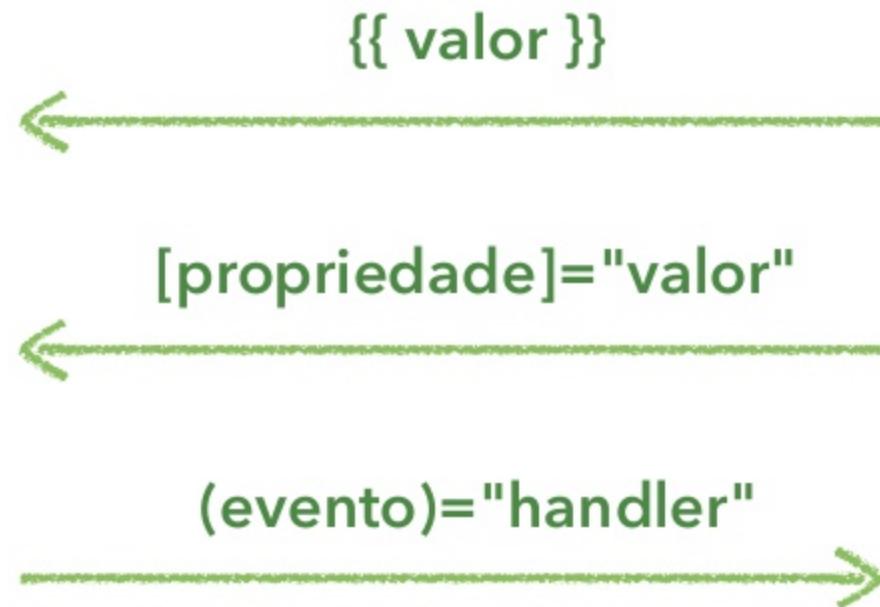
Angular 2

Event Binding
(Binding de eventos)



DATA BINDING

<TEMPLATE>



{COMPONENT}



EVENT BINDING

- ▶ Valor do Template para o Componente



EVENT BINDING

- ▶ Valor do Template para o Componente
- ▶ Usa-se parênteses <button (click)="onClick()" />



EVENT BINDING

- ▶ Valor do Template para o Componente
- ▶ Usa-se parênteses <button (click)="onClick()" />
- ▶ Método "handler" declarado no Componente



EVENT BINDING

- ▶ Valor do Template para o Componente
- ▶ Usa-se parênteses <button **(click)**=“onClick()”/>
- ▶ Método “handler” declarado no Componente
- ▶ Forma canônica <button **on-click**=“onClick()”/>



EVENT BINDING - EXEMPLO 01

```
<button class="btn" (click)="onClick()>Me clique!</button>
```



EVENT BINDING - EXEMPLO 01

```
<button class="btn" (click)="onClick()>Me clique!</button>
```

A blue callout bubble with a white border and a triangular pointer points from the bottom left towards the '(click)' part of the code. The text inside the bubble reads:

Evento a ser
"escutado"



EVENT BINDING - EXEMPLO 01

```
<button class="btn" (click)="onClick()>Me clique!</button>
```

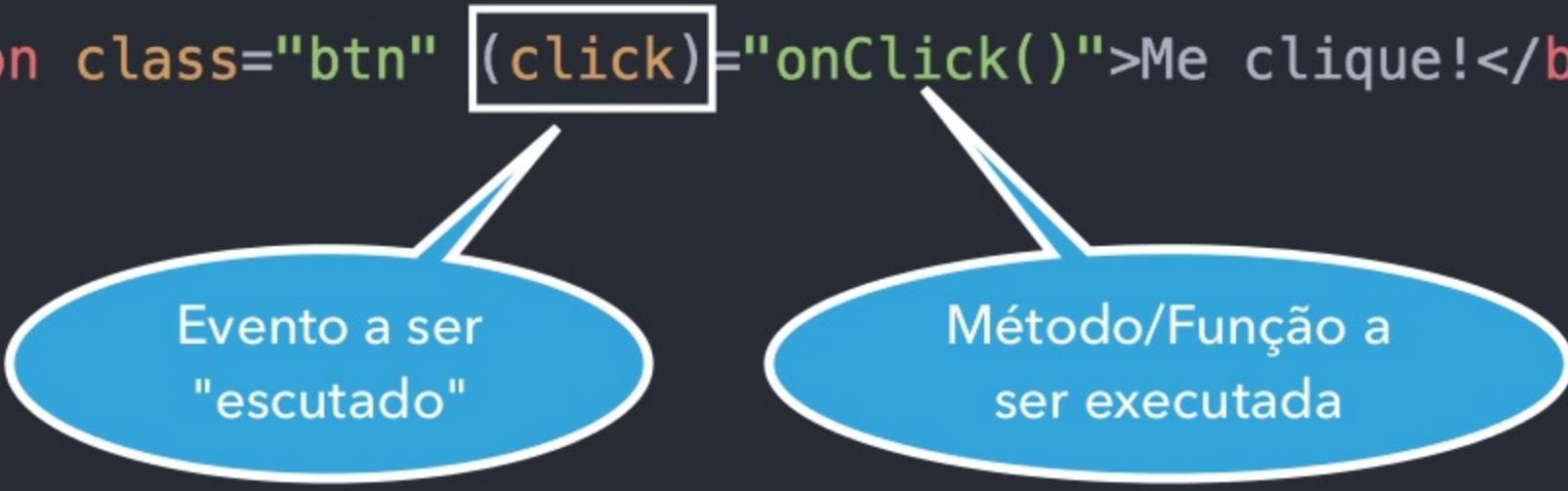
Evento a ser
"escutado"

Método/Função a
ser executada



EVENT BINDING - EXEMPLO 01

```
<button class="btn" (click)="onClick()>Me clique!</button>
```



Evento a ser
"escutado"

Método/Função a
ser executada

```
export class DataBindingComponent {  
  
  onClick() {  
    alert('Botão clicado!');  
  }  
}
```



EVENT BINDING - EXEMPLO 01

```
<button class="btn" (click)="onClick()>Me clique!</button>
```

Evento a ser
"escutado"

Método/Função a
ser executada

```
export class DataBindingComponent {
```

```
  onClick() {  
    alert('Botão clicado!');  
  }  
}
```



EVENT BINDING - EXEMPLO 02

```
<input type="text"
  (keyup)="onKeyup($event)"
  (keyup.enter)="onSave(input.value)"
  (blur)="onSave(input.value)"
  #input
>
<div>Conteúdo Atual: {{conteudoAtual}}</div>
<div>Conteúdo salvo: {{conteudoSalvo}}</div>
```



EVENT BINDING - EXEMPLO 02

\$event é do tipo
evento DOM

```
<input type="text"
  (keyup)="onKeyup($event)"
  (keyup.enter)="onSave(input.value)"
  (blur)="onSave(input.value)"
  #input
>
<div>Conteúdo Atual: {{conteudoAtual}}</div>
<div>Conteúdo salvo: {{conteudoSalvo}}</div>
```



EVENT BINDING - EXEMPLO 02

```
<input type="text"
  (keyup)="onKeyup($event)"
  (keyup.enter)="onSave(input.value)"
  (blur)="onSave(input.value)"
  #input>
<div>Conteúdo Atual: {{conteudo}}
<div>Conteúdo salvo: {{conteudoSalvo}}</div>
```

The diagram illustrates the event binding on the input element. A callout bubble points to the '\$event' parameter in the 'onKeyup' handler, stating: '\$event é do tipo evento DOM'. Another callout bubble points to the 'input.value' part of the 'onSave' handler, stating: 'acesso a variável local declarada'.



EVENT BINDING - EXEMPLO 02

```
export class DataBindingComponent {  
  
  conteudoAtual = '';  
  conteudoSalvo = '';  
  
  onKeyup(event:KeyboardEvent) {  
    console.log(event);  
    this.conteudoAtual = (<HTMLInputElement> event.target).value;  
  }  
  
  onSave(value: string) {  
    this.conteudoSalvo = value;  
  }  
}
```



EVENT BINDING - EXEMPLO 02

```
export class DataBindingComponent {  
  conteudoAtual = '';  
  conteudoSalvo = '';  
  
  onKeyup(event:KeyboardEvent) {  
    console.log(event);  
    this.conteudoAtual = (<HTMLInputElement> event.target).value;  
  }  
  
  onSave(value: string) {  
    this.conteudoSalvo = value;  
  }  
}
```

\$event tem
propriedades como
event.target e
event.target.value



EVENT BINDING - EXEMPLO 03

```
<span  
  (mouseover)="onToggleHover()"  
  (mouseout)="onToggleHover()"  
  [class.highlight]="isMouseover">  
  Passe o mouse sobre mim!  
</span>
```



EVENT BINDING - EXEMPLO 03

```
export class DataBindingComponent {  
  isMouseover = false;  
  
  onToggleHover() {  
    this.isMouseover = !this.isMouseover;  
  }  
}
```



EVENT BINDING - EXEMPLO 03

```
.highlight {  
  font-weight: bold;  
  background-color: yellow;  
  color: black;  
}
```



DICA: USANDO CSS NO COMPONENTE

```
styles: [
  `
    .highlight {
      font-weight: bold;
      background-color: yellow;
      color: black;
    }
  `,
],
styleUrls: ['data-binding.component.css']
```



DICA: USANDO CSS NO COMPONENTE

```
styles: [
  `
    .highlight {
      font-weight: bold;
      background-color: yellow;
      color: black;
    }
  `,
],
styleUrls: ['data-binding.component.css']
```

Note a convenção de
nomenclatura



QUAIS EVENTOS POSSO USAR?

- ▶ Lista com todos os eventos
- ▶ <https://developer.mozilla.org/en-US/docs/Web/Events>

The Angular 2 logo consists of a red 3D hexagonal shape with the number '12' in white on its front face, positioned to the left of the word 'Angular 2'.

12 Angular 2

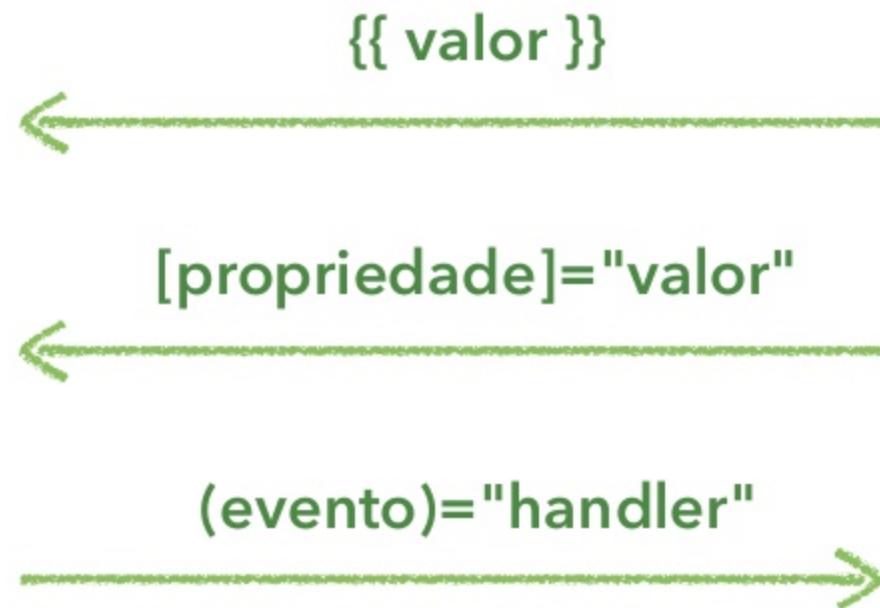
A faint, semi-transparent silhouette of a city skyline with various buildings and landmarks, serving as a background for the text.

Two-way data binding



DATA BINDING

<TEMPLATE>



{COMPONENT}



DATA BINDING

<TEMPLATE>

`{{ valor }}`

`[propriedade]="valor"`

`(evento)="handler"`

`[(ngModel)]="propriedade"`

{COMPONENT}



TWO-WAY DATA BINDING

- ▶ Valor do Template para o Componente e vice-versa



TWO-WAY DATA BINDING

- ▶ Valor do Template para o Componente e vice-versa
- ▶ Usa-se binding de eventos + propriedades



TWO-WAY DATA BINDING

Como manter o template e o componente atualizados?



TWO-WAY DATA BINDING

Como manter o template e o componente atualizados?

```
<input type="text"  
      [value]="nome"  
      (input)='nome = $event.target.value' />
```



TWO-WAY DATA BINDING

Como manter o template e o componente atualizados?

```
<input type="text"  
      [value]="nome"  
      (input)='nome = $event.target.value' />
```



TWO-WAY DATA BINDING

Como manter o template e o componente atualizados?

```
<input type="text"  
      [value]="nome"  
      (input)='nome = $event.target.value' />
```



TWO-WAY DATA BINDING

Como manter o template e o componente atualizados?

```
<input type="text"  
      [value]="nome"  
      (input)='nome = $event.target.value' />
```



Property binding +
event binding



TWO-WAY DATA BINDING COM NGMODEL

Aplicando o ngModel

```
<input type="text"  
       [ngModel]="nome"  
       (ngModelChange)="nome = $event" />
```



TWO-WAY DATA BINDING

- ▶ Valor do Template para o Componente e vice-versa
- ▶ Usa-se binding de eventos + propriedades
- ▶ Forma mais simples é fazer o binding no ngModel



TWO-WAY DATA BINDING

- ▶ Valor do Template para o Componente e vice-versa
- ▶ Usa-se binding de eventos + propriedades
- ▶ Forma mais simples é fazer o binding no ngModel
- ▶ Usa-se sintaxe de binding de eventos + propriedade:



TWO-WAY DATA BINDING

- ▶ Valor do Template para o Componente e vice-versa
- ▶ Usa-se binding de eventos + propriedades
- ▶ Forma mais simples é fazer o binding no ngModel
- ▶ Usa-se sintaxe de binding de eventos + propriedade:
 - ▶ `<input type="text" [(ngModel)]="nome" />`



TWO-WAY DATA BINDING

- ▶ Valor do Template para o Componente e vice-versa
- ▶ Usa-se binding de eventos + propriedades
- ▶ Forma mais simples é fazer o binding no ngModel
- ▶ Usa-se sintaxe de binding de eventos + propriedade:
 - ▶ `<input type="text" [(ngModel)]="nome" />`
- ▶ Forma canônica é



TWO-WAY DATA BINDING

- ▶ Valor do Template para o Componente e vice-versa
- ▶ Usa-se binding de eventos + propriedades
- ▶ Forma mais simples é fazer o binding no ngModel
- ▶ Usa-se sintaxe de binding de eventos + propriedade:
 - ▶ `<input type="text" [(ngModel)]="nome" />`
- ▶ Forma canônica é
 - ▶ `<input type="text" bindon-ngModel="nome" />`



TWO-WAY DATA BINDING

```
<input type="text" [(ngModel)]="nome" />
<input type="text" bindon-ngModel="nome" />
<br>
<p>Você digitou {{ nome }}</p>
```



TWO-WAY DATA BINDING

```
<input type="text" [(ngModel)]="nome" />
<input type="text" bindon-ngModel="nome" />
<br>
<p>Você digitou {{ nome }}</p>
```



TWO-WAY DATA BINDING

Two-way
data binding com
NgModel

```
<input type="text" [(ngModel)]="nome" />
<input type="text" bindon-ngModel="nome" />
<br>
<p>Você digitou {{ nome }}</p>
```



TWO-WAY DATA BINDING

Two-way
data binding com
NgModel

```
<input type="text" [(ngModel)]="nome" />
<input type="text" bindon-ngModel="nome" />
<br>
<p>Você digitou {{ nome }}</p>
```

Forma canônica



SINTAXE BANANA IN A BOX

SINTAXE “BANANA NA CAIXA”





TWO-WAY DATA BINDING

Meu nome é {{pessoa.nome}} e tenho {{pessoa.idade}} anos.
<input type="text" [(ngModel)]="pessoa.nome">
<input type="text" [(ngModel)]="pessoa.idade">


```
{nome: '', idade: 0}
```



TWO-WAY DATA BINDING

Meu nome é {{pessoa.nome}} e tenho {{pessoa.idade}} anos.
<input type="text" [(ngModel)]="pessoa.nome">
<input type="text" [(ngModel)]="pessoa.idade">



Two-way data
binding com objetos

{nome: '', idade: 0}

13 Angular 2

Reusando Componentes

Input properties



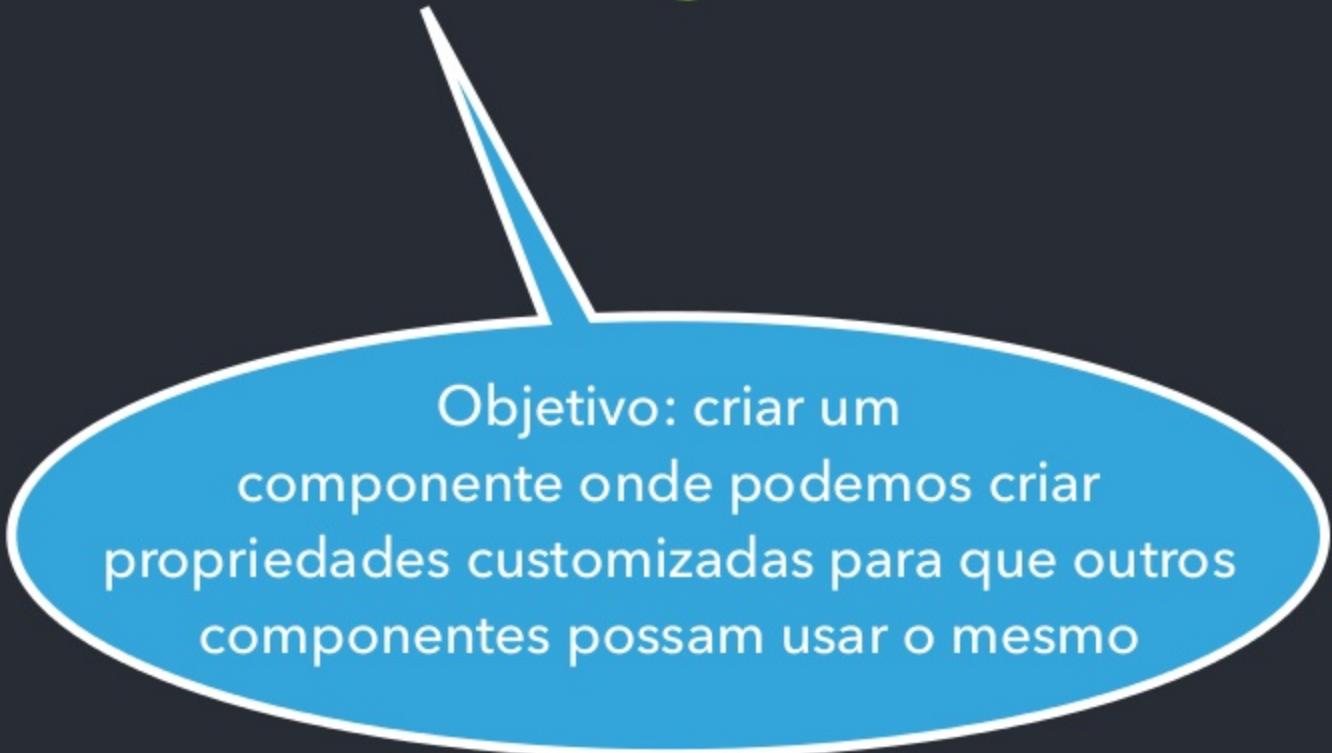
CRIANDO COMPONENTES REUTILIZÁVEIS

```
<curso nome="Angular2"></curso>
```



CRIANDO COMPONENTES REUTILIZÁVEIS

```
<curso nome="Angular2"></curso>
```



Objetivo: criar um componente onde podemos criar propriedades customizadas para que outros componentes possam usar o mesmo



CRIANDO COMPONENTES REUTILIZÁVEIS

```
@Component({
  selector: 'curso',
  template: `
    <p>{{curso}}</p>
  `
})
export class InputPropertyComponent {
  curso: string = null;
}
```

1 - Criamos o componente normalmente



INPUT PROPERTIES

```
@Component({
  selector: 'curso',
  template: `
    <p>{{curso}}</p>
  `
})
export class InputPropertyComponent {
  @Input() curso: string = null;
}
```



INPUT PROPERTIES

```
@Component({  
  selector: 'curso',  
  template: `  
    <p>{{curso}}</p>  
  `,  
})  
export class InputPropertyComponent {  
  @Input() curso: string = null;  
}
```

2 - Usamos a anotação `@Input` nos atributos que queremos expor



INPUT PROPERTIES

```
@Input('nome') curso: string = null;
```



INPUT PROPERTIES

```
@Input('nome') curso: string = null;
```

3 - Caso queira expor o atributo/propriedade com nome diferente, podemos passar como parâmetro para o @input



INPUT PROPERTIES

```
@Component({  
  selector: 'curso',  
  template: `  
    <p>{{curso}}</p>  
  `,  
  inputs: ['curso:nome']  
})
```

4 - Caso não queria usar o @Input(), pode declarar como meta-dado no @Component também



QUAL USAR? METADADO OU ANOTAÇÃO?

```
@Component({  
  selector: 'curso',  
  template: `'  
    <p>{{curso}}</p>  
  `,  
  inputs: ['curso:nome']  
})  
export class InputPropertyComponent {  
  @Input('nome') curso: string = null;  
}
```

Gosto pessoal.

Porém, o `@Input` é mais notável visualmente!

14 Angular 2

Reusando Componentes

Output properties



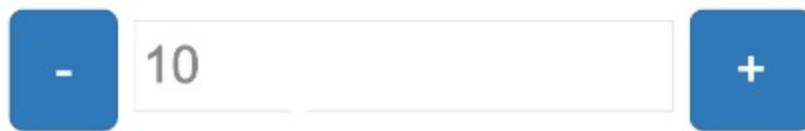
CRIANDO COMPONENTES REUTILIZÁVEIS

A number input component with a value of 10, flanked by minus and plus buttons.

The component consists of a horizontal input field containing the number '10'. To the left of the input is a blue button with a white minus sign ('-'). To the right is another blue button with a white plus sign ('+').



CRIANDO COMPONENTES REUTILIZÁVEIS



Componente customizado.

Objetivo: disparar um evento “mudou” toda vez que o usuário clicar nos botões de + ou -.

Evento recebe novo valor do input.



CRIANDO COMPONENTES REUTILIZÁVEIS

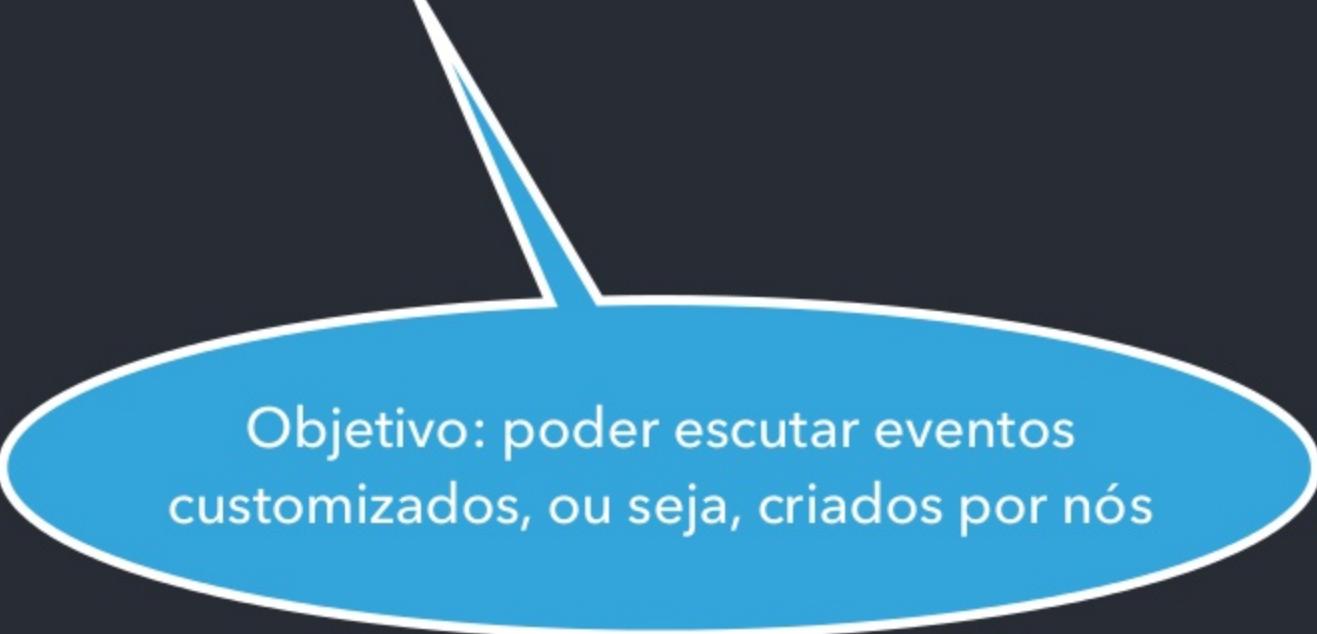
```
<contador [valor]="10" (mudou)="onMudou($event)"></contador>

onMudou(valor : any) {
  alert(valor.novoValor);
}
```



CRIANDO COMPONENTES REUTILIZÁVEIS

```
<contador [valor]="10" (mudou)="onMudou($event)"></contador>
```

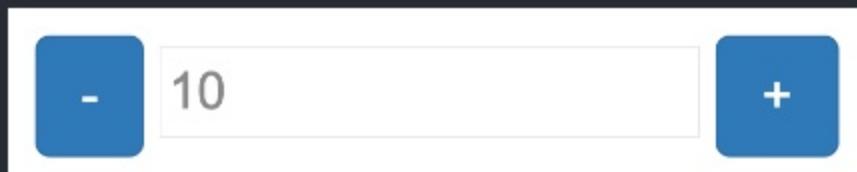


Objetivo: poder escutar eventos
customizados, ou seja, criados por nós

```
onMudou(valor : any) {  
  alert(valor.novoValor);  
}
```



CRIANDO COMPONENTES REUTILIZÁVEIS



```
<div>
  <button type="button" class="btn btn-primary"
    (click)="decrementa()"></button>
  <input type="text" [value]="valor" readonly>
  <button type="button" class="btn btn-primary"
    (click)="incrementa()"></button>
</div>
```

1 - Criamos o componente normalmente



CRIANDO COMPONENTES REUTILIZÁVEIS

```
@Input() valor : number = 0;
```

```
decrementa(){
  this.valor--;
}

incrementa(){
  this.valor++;
}
```

A user interface element consisting of a horizontal input field containing the number '10'. To the left of the input field is a blue button with a white minus sign (-). To the right is a blue button with a white plus sign (+).

1 - Criamos o componente normalmente



CRIANDO COMPONENTES REUTILIZÁVEIS

```
@Output() mudou = new EventEmitter();
```

2 - Criamos um atributo do tipo EventEmitter (que irá emitir o evento “mudou”.

Anotamos com o @Output para mostrar que o atributo será exposto como valor de saída do Component



CRIANDO COMPONENTES REUTILIZÁVEIS

3 - Emitimos o evento com o valor que desejamos expor

```
decrementa(){
  this.valor--;
  this.mudou.emit({novoValor: this.valor});
}

incrementa(){
  this.valor++;
  this.mudou.emit({novoValor: this.valor});
}
```



CRIANDO COMPONENTES REUTILIZÁVEIS

```
@Output('outroNomeEvento') mudou = new EventEmitter();
```

4 - Também podemos expor o evento com um nome diferente do que utilizado internamente.



CRIANDO COMPONENTES REUTILIZÁVEIS

```
outputs: ['mudou']
```

5 - O Output também pode ser usado em forma de metadado



15 Angular 2

Ciclo de vida do
Componente



LIFE CYCLE HOOKS

#

EVENTO (HOOKS)

QUANDO?

1

ngOnChanges

antes #2 e quando valor property-binding é atualizado



LIFE CYCLE HOOKS

#	EVENTO (HOOKS)	QUANDO?
1	ngOnChanges	antes #2 e quando valor property-binding é atualizado
2	ngOnInit	quando Component é inicializado



LIFE CYCLE HOOKS

#	EVENTO (HOOKS)	QUANDO?
1	<code>ngOnChanges</code>	antes #2 e quando valor property-binding é atualizado
2	<code>ngOnInit</code>	quando Component é inicializado
3	<code>ngDoCheck</code>	a cada ciclo de verificação de mudanças



LIFE CYCLE HOOKS

#	EVENTO (HOOKS)	QUANDO?
1	ngOnChanges	antes #2 e quando valor property-binding é atualizado
2	ngOnInit	quando Component é inicializado
3	ngDoCheck	a cada ciclo de verificação de mudanças
4	ngAfterContentInit	depois de inserir conteúdo externo na view



LIFE CYCLE HOOKS

#	EVENTO (HOOKS)	QUANDO?
1	ngOnChanges	antes #2 e quando valor property-binding é atualizado
2	ngOnInit	quando Component é inicializado
3	ngDoCheck	a cada ciclo de verificação de mudanças
4	ngAfterContentInit	depois de inserir conteúdo externo na view
5	ngAfterContentChecked	a cada verificação de conteúdo inserido



LIFE CYCLE HOOKS

#	EVENTO (HOOKS)	QUANDO?
1	ngOnChanges	antes #2 e quando valor property-binding é atualizado
2	ngOnInit	quando Component é inicializado
3	ngDoCheck	a cada ciclo de verificação de mudanças
4	ngAfterContentInit	depois de inserir conteúdo externo na view
5	ngAfterContentChecked	a cada verificação de conteúdo inserido
6	ngAfterViewChecked	a cada verificação de conteúdo / conteúdo filho



LIFE CYCLE HOOKS

#	EVENTO (HOOKS)	QUANDO?
1	ngOnChanges	antes #2 e quando valor property-binding é atualizado
2	ngOnInit	quando Component é inicializado
3	ngDoCheck	a cada ciclo de verificação de mudanças
4	ngAfterContentInit	depois de inserir conteúdo externo na view
5	ngAfterContentChecked	a cada verificação de conteúdo inserido
6	ngAfterViewChecked	a cada verificação de conteúdo / conteúdo filho
7	ngOnDestroy	antes da diretiva/component ser destruído

 LIFE CYCLE HOOKS

```
export class LifeCycleComponent implements OnChanges,  
  OnInit, DoCheck, AfterContentInit,  
  AfterContentChecked, AfterViewInit, AfterViewChecked,  
  OnDestroy {  
  
  ngOnChanges() {  
    this.log('ngOnChanges');  
  }  
  
  ngOnInit() {  
    this.log('ngOnInit');  
  }  
  
  ngDoCheck() {  
    this.log('ngDoCheck');  
  }  
}
```

A LIFE CYCLE HOOKS

```
export class LifeCycleComponent implements OnChanges,  
OnInit, DoCheck, AfterContentInit,  
AfterContentChecked, AfterViewInit, AfterViewChecked,  
OnDestroy {  
  
  ngOnChanges() {  
    this.log('ngOnChanges');  
  }  
  
  ngOnInit() {  
    this.log('ngOnInit');  
  }  
  
  ngDoCheck() {  
    this.log('ngDoCheck');  
  }  
}
```

Note o nome da interface e
o nome do método a ser
implementado:

ng + NomeInterface

16 Angular 2

Acesso de variáveis locais do
Template com ViewChild

```
template: `
  <p #variavelLocalP>{{valorComBind}}</p>
`,

@ViewChild('variavelLocalP') variavelLocalP : HTMLElement;

ngAfterViewInit() {
  this.log('ngAfterViewInit');
  console.log(this.variavelLocalP);
}
```



Angular 2

Mudanças Angular RC 5

NgModule



O QUE MUDOU NO RC5?

- ▶ Introdução do conceito de ngModule
 - ▶ Breaking change a partir do RC 6 / Release
 - ▶ Forma de organizar a aplicação em módulos
 - ▶ Toda a app precisa de um módulo raiz (root)
 - ▶ É possível também criar módulos de features



COMO ATUALIZAR O PROJETO PARA O RC 5?

- ▶ 1 - Atualizar a dependencies conforme <https://github.com/angular/quickstart/blob/master/package.json>
- ▶ 2 - Substituir o app.component.spec.ts (<https://github.com/angular/quickstart/blob/master/app/app.component.spec.ts>)
- ▶ 3 - Substituir o main.ts (<https://github.com/angular/quickstart/blob/master/app/main.ts>)
- ▶ 4 - Criar arquivo app.module.ts (<https://github.com/angular/quickstart/blob/master/app/app.module.ts>)
- ▶ 5 - Refatorar!



COMO ATUALIZAR O PROJETO PARA O RC 5?



```
> npm outdated  
> npm update
```



COMO ATUALIZAR O PROJETO PARA O RC 5?

```
> npm install -g npm-check-updates  
> npm-check-updates -u  
> npm update
```



COMO ATUALIZAR O PROJETO PARA O RC 5?

```
> npm install -g npm-check-updates  
> npm-check-updates -u  
> npm update
```

atualiza o package.json
também



APP.MODULE.TS (NOVO ARQUIVO)

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

import { AppComponent }      from './app.component';

import { MeuPrimeiroComponent } from './hello/meu-primeiro.component';
import { CursosComponent }    from './cursos/cursos.component';
import { CursosService}      from './cursos/cursos.service';

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



APP.MODULE.TS (NOVO ARQUIVO)

```
import { NgModule }           from '@angular/core';
import { BrowserModule }    from '@angular/platform-browser';
import { AppComponent }      from './app.component';
import { MeuPrimeiroComponent } from './hello/meu-primeiro.component';
import { CursosComponent }   from './cursos/cursos.component';
import { CursosService}      from './cursos/cursos.service';

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

Import do decorator e módulo obrigatório para a app



APP.MODULE.TS (NOVO ARQUIVO)

```
import { NgModule }           from '@angular/core';
import { BrowserModule }    from '@angular/platform-browser';
import { AppComponent }      from './app.component';
import { MeuPrimeiroComponent } from './hello/meu-primeiro.component';
import { CursosComponent }   from './cursos/cursos.component';
import { CursosService}      from './cursos/cursos.service';

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

Import do decorator e módulo obrigatório para a app

Import do componente principal



APP.MODULE.TS (NOVO ARQUIVO)

```
import { NgModule }           from '@angular/core';
import { BrowserModule }    from '@angular/platform-browser';
import { AppComponent }      from './app.component';
import { MeuPrimeiroComponent } from './hello/meu-primeiro.component';
import { CursosComponent }   from './cursos/cursos.component';
import { CursosService }     from './cursos/cursos.service';

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

Import do decorator e módulo obrigatório para a app

Import do componente principal

Import das classes



APP.MODULE.TS (NOVO ARQUIVO)

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

import { AppComponent }     from './app.component';

import { MeuPrimeiroComponent } from './hello/meu-primeiro.component';
import { CursosComponent }   from './cursos/cursos.component';
import { CursosService}      from './cursos/cursos.service';

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



APP.MODULE.TS (NOVO ARQUIVO)

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

import { AppComponent }      from './app.component';
import { MeuPrimeiroComponent } from './hello';
import { CursosComponent }   from './cursos/cursos';
import { CursosService}      from './cursos/cursos.service';

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

imports: módulos a serem importados



APP.MODULE.TS (NOVO ARQUIVO)

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

import { AppComponent }      from './app.component';
import { MeuPrimeiroComponent } from './hello';
import { CursosComponent }   from './cursos/cursos';
import { CursosService}      from './cursos/cursos.service';

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

imports: módulos a serem importados

declarations: directives



APP.MODULE.TS (NOVO ARQUIVO)

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

import { AppComponent }      from './app.component';
import { MeuPrimeiroComponent } from './hello';
import { CursosComponent }   from './cursos/cursos';
import { CursosService}      from './cursos/cursos.service';

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

imports: módulos a serem importados

declarations: directives

providers (serviços)



APP.MODULE.TS (NOVO ARQUIVO)

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

import { AppComponent }     from './app.component';
import { MeuPrimeiroComponent } from './hello';
import { CursosComponent }   from './cursos/cursos';
import { CursosService}      from './cursos/cursos.service';

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

imports: módulos a serem importados

declarations: directives

providers (serviços)

Componente principal da app



MUDOU O CONTEÚDO DO MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app.module';

platformBrowserDynamic().bootstrapModule(AppModule);
```

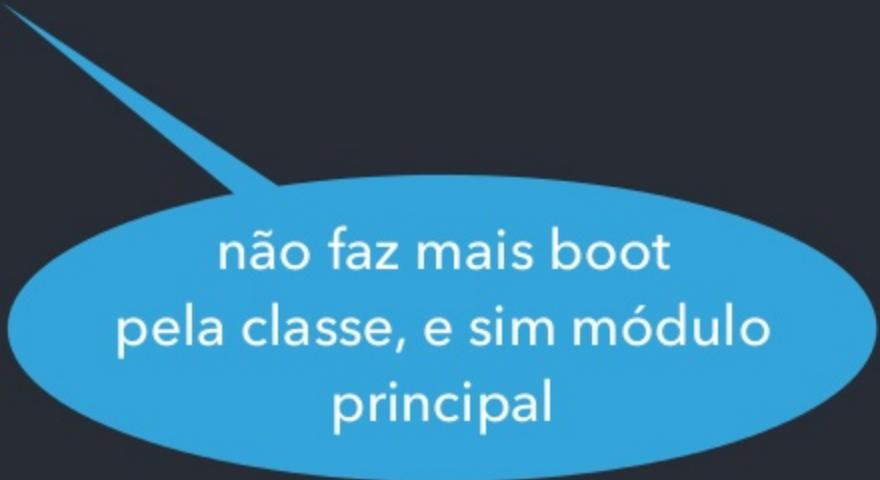


MUDOU O CONTEÚDO DO MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app.module';

platformBrowserDynamic().bootstrapModule(AppModule);
```



não faz mais boot
pela classe, e sim módulo
principal



NOS COMPONENTES

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

@Component({
  selector: 'my-app',
  template: `
    <h1>Angular 2 Boilerplate</h1>
    <input type="text" autoGrow />
    <meu-primeiro-component></meu-primeiro-component>
    <cursos></cursos>
    `
})
export class AppComponent { }
```



NOS COMPONENTES

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

@Component({
  selector: 'my-app',
  template: `
    <h1>Angular</h1>
    <input type="text" placeholder="Type something...">
    <meu-primeiro-componente></meu-primeiro-componente>
    <cursos></cursos>
  `,
})
export class AppComponent { }
```

não precisa mais do import e nem do meta-dado directives pra usar a tag cursos



É POSSÍVEL CRIAR MÓDULO DE FEATURE

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

import { CursosComponent } from './cursos.component';
import {CursosService} from './cursos.service';

@NgModule({
  imports: [ CommonModule ],
  declarations: [CursosComponent],
  providers: [CursosService],
  exports: [CursosComponent]
})
export class CursosModule { }
```



É POSSÍVEL CRIAR MÓDULO DE FEATURE

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
```

```
import { CursosComponent } from './';
import {CursosService} from './'
```

usa-se o
CommonModule e não
BrowserModule

```
@NgModule({
  imports: [ CommonModule ],
  declarations: [CursosComponent],
  providers: [CursosService],
  exports: [CursosComponent]
})
export class CursosModule { }
```



É POSSÍVEL CRIAR MÓDULO DE FEATURE

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
```

```
import { CursosComponent } from './';
import {CursosService} from './'
```

usa-se o
CommonModule e não
BrowserModule

```
@NgModule({
  imports: [ CommonModule ],
  declarations: [CursosComponent],
  providers: [CursosService],
  exports: [CursosComponent]
})
```

```
export class CursosModule { }
```

classes a serem expostas
em outros módulos



É POSSÍVEL CRIAR MÓDULO DE FEATURE

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

import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  moduleId: module.id,
  templateUrl: 'cursos.component.html'
})
export class CursosComponent {

  tituloCursos = 'loiane.training';
  cursos: String[];

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCurso();
  }
}
```



É POSSÍVEL CRIAR MÓDULO DE FEATURE

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

import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  moduleId: module.id,
  templateUrl: 'cursos.component.html'
})
export class CursosComponent {

  tituloCursos = 'loiane.training';
  cursos: String[];

  constructor(cursosService: CursosService){
    this.cursos = cursosService.getCurso();
  }
}
```

precisa importar pois
está sendo usada/declarada no
construtor



É POSSÍVEL CRIAR MÓDULO DE FEATURE

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

import {CursosService} from './cursos.service';

@Component({
  selector: 'cursos',
  moduleId: module.id,
  templateUrl: 'cursos.component.html'
})
export class CursosComponent {

  tituloCursos = 'Loiane.training';
  cursos: any[] = [];

  constructor(private cursosService: CursosService) {
    this.cursos = this.cursosService.getCurso();
  }
}
```

precisa importar pois
está sendo usada/declarada no
construtor

não precisa mais do meta-
dado provider



REFATORANDO

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

import { AppComponent }  from './app.component';

import {MeuPrimeiroComponent} from './hello/meu-primeiro.component';
import {AutoGrowDirective} from './shared/auto-grow.directive';

import {CursosModule} from './cursos/cursos.module';

@NgModule({
  imports: [ BrowserModule, CursosModule ],
  declarations: [
    AppComponent,
    MeuPrimeiroComponent,
    AutoGrowDirective],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

A REFATORANDO

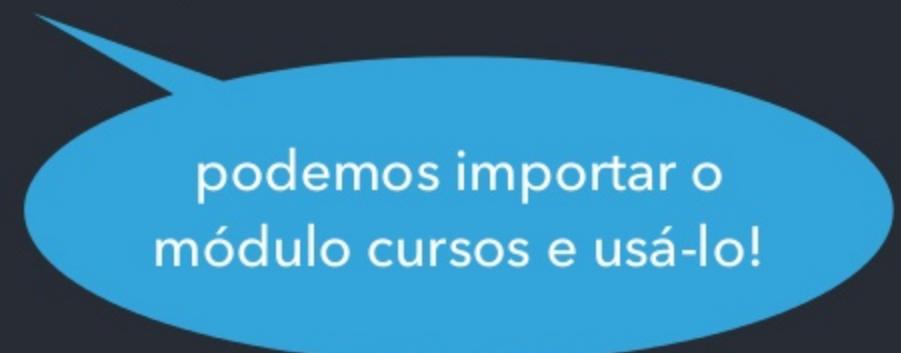
```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent }  from './app.component';

import {MeuPrimeiroComponent} from './hello/meu-primeiro.component';
import {AutoGrowDirective} from './shared/auto-grow.directive';

import {CursosModule} from './cursos/cursos.module';

@NgModule({
  imports: [ BrowserModule, CursosModule ],
  declarations: [
    AppComponent,
    MeuPrimeiroComponent,
    AutoGrowDirective],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```



podemos importar o
módulo cursos e usá-lo!



PROJETO DATA-BINDING

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

import { AppComponent }  from './app.component';

import { DataBindingComponent } from './data-binding/data-binding.component';
import { LifeCycleComponent } from './life-cycle/life-cycle.component';

@NgModule({
  imports: [ BrowserModule, FormsModule ],
  declarations: [
    AppComponent,
    DataBindingComponent,
    LifeCycleComponent
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```



PROJETO DATA-BINDING

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

import { AppComponent }      from './app.component';
import { DataBindingComponent } from './data-binding.component';
import { LifeCycleComponent } from './life-cycle.component';

@NgModule({
  imports: [ BrowserModule, FormsModule ],
  declarations: [
    AppComponent,
    DataBindingComponent,
    LifeCycleComponent
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

two-way data-binding ([ngModel])
precisa do import

17 Angular 2

Angular CLI: Instalação e
criação de projetos: ng new,
ng init e ng serve



```
$ npm install -g angular-cli
```

Requer node >= 4

<https://github.com/angular/angular-cli>



ANGULAR CLI: CRIANDO PROJETO

```
> ng new NomeProjeto  
> cd NomeProjeto  
> ng serve
```



ANGULAR CLI: CRIANDO PROJETO



```
> ng new NomeProjeto  
> cd NomeProjeto  
> ng serve
```

A terminal window icon is shown at the top left, featuring three colored dots (red, yellow, green) on a grey background. Below it is a black rectangular area containing three command-line entries: 'ng new NomeProjeto', 'cd NomeProjeto', and 'ng serve'. The entire terminal window is set against a light blue background.

Cria um projeto angular 2 e executa npm install



ANGULAR CLI: CRIANDO PROJETO

```
● ● ●  
> ng new NomeProjeto  
> cd NomeProjeto  
> ng serve
```

Cria um projeto angular 2 e executa npm install

"serve" o projeto ao browser.

Similar ao npm start que estávamos executando



ANGULAR CLI: CRIANDO PROJETO EM DIRETORIO EXISTENTE

```
> mkdir NomeProjeto  
> cd NomeProjeto  
> ng init  
> ng serve
```



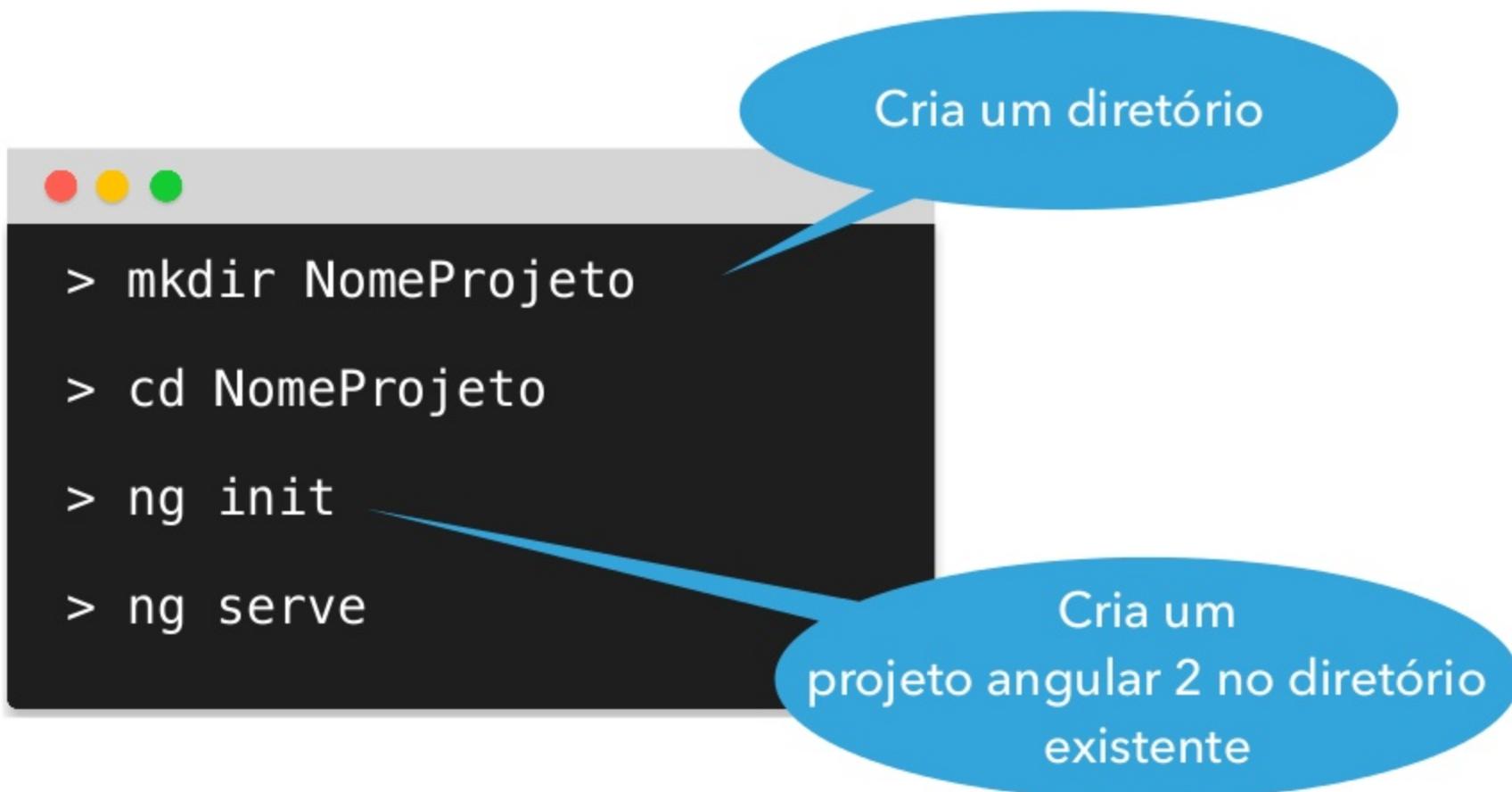
ANGULAR CLI: CRIANDO PROJETO EM DIRETORIO EXISTENTE

```
> mkdir NomeProjeto  
> cd NomeProjeto  
> ng init  
> ng serve
```

Cria um diretório



ANGULAR CLI: CRIANDO PROJETO EM DIRETORIO EXISTENTE



18 Angular 2

Angular CLI:

Criando components, services:
ng generate



ANGULAR CLI: CRIANDO COMPONENTS

```
> cd NomeProjeto  
> ng generate component meu-primeiro  
> ng g component meu-primeiro
```



ANGULAR CLI: CRIANDO COMPONENTS

```
> cd NomeProjeto  
> ng generate component meu-primeiro  
> ng g component meu-primeiro
```



ANGULAR CLI: CRIANDO COMPONENTS

```
projeto-teste — bash — 80x11
[loiane:angular-cli loiane$ cd projeto-teste
[loiane:projeto-teste loiane$ ng g component meu-primeiro
Could not start watchman; falling back to NodeWatcher for file system events.
Visit http://ember-cli.com/user-guide/#watchman for more info.
installing component
  create src/app/meu-primeiro/meu-primeiro.component.css
  create src/app/meu-primeiro/meu-primeiro.component.html
  create src/app/meu-primeiro/meu-primeiro.component.spec.ts
  create src/app/meu-primeiro/meu-primeiro.component.ts
  create src/app/meu-primeiro/index.ts
loiane:projeto-teste loiane$
```



ANGULAR CLI: CRIANDO SERVIÇOS

```
> cd NomeProjeto  
> ng generate service meu-primeiro/  
meu-primeiro
```



ANGULAR CLI: CRIANDO SERVIÇOS

```
> cd NomeProjeto  
> ng generate service meu-primeiro/  
meu-primeiro
```



Cria um serviço no
módulo meu-primeiro



ANGULAR CLI: CRIANDO SERVIÇOS

```
▼ └── src
    └── app
        └── meu-primeiro
            ├── index.ts
            ├── meu-primeiro.component.css
            ├── meu-primeiro.component.html
            ├── meu-primeiro.component.spec.ts
            ├── meu-primeiro.component.ts
            ├── meu-primeiro.service.spec.ts
            └── meu-primeiro.service.ts
```



ANGULAR CLI: CRIANDO SERVIÇOS

```
▼  src
  ▼  app
    ▼  meu-primeiro
      index.ts
      meu-primeiro.component.css
      meu-primeiro.component.html
      meu-primeiro.component.spec.ts
      meu-primeiro.component.ts
      meu-primeiro.service.spec.ts
      meu-primeiro.service.ts
```

Cria um serviço no
módulo meu-primeiro

Arquivo spec.ts é para testes
unitários



ANGULAR 2 STYLE GUIDE

▼ meu-primeiro

- index.ts
- meu-primeiro.component.css
- meu-primeiro.component.html
- meu-primeiro.component.spec.ts
- meu-primeiro.component.ts
- meu-primeiro.service.spec.ts
- meu-primeiro.service.ts



ANGULAR 2 STYLE GUIDE

▼ meu-primeiro

- index.ts
- meu-primeiro.component.css
- meu-primeiro.component.html
- meu-primeiro.component.spec.ts
- meu-primeiro.component.ts
- meu-primeiro.service.spec.ts
- meu-primeiro.service.ts



ANGULAR 2 STYLE GUIDE

▼ meu-primeiro

- index.ts
- meu-primeiro.component.css
- meu-primeiro.component.html
- meu-primeiro.component.spec.ts
- meu-primeiro.component.ts
- meu-primeiro.service.spec.ts
- meu-primeiro.service.ts



ANGULAR 2 STYLE GUIDE

- ▼ meu-primeiro
 - TS index.ts
 - SCSS meu-primeiro.component.css
 - HTML meu-primeiro.component.html
 - TS meu-primeiro.component.spec.ts
 - TS meu-primeiro.component.ts
 - TS meu-primeiro.service.spec.ts
 - TS meu-primeiro.service.ts

Arquivos são gerados
usando o padrão de
nomenclatura e boas práticas
segundo o style guide



ANGULAR CLI: CRIANDO ARQUIVOS

Tipo Arquivo	Comando
Component	<code>ng g component meu-component</code>
Service	<code>ng g service meu-servico</code>
Directive	<code>ng g directive minha-diretiva</code>
Pipe	<code>ng g pipe meu-pipe</code>
Class	<code>ng g class minha-classe</code>
Interface	<code>ng g interface minha-interface</code>
Enum	<code>ng g enum meu-enum</code>



Angular 2

Atualizando o angular-cli para RC5 e webpack



ANGULAR CLI: UPDATE RC5

- ▶ Situação atual (agosto/2016):
 - ▶ upgrade do build para o webpack (estava usando o broccoli)
 - ▶ Versão ainda em beta (1.0.0-beta.11-webpack.2)
 - ▶ é a versão que usa RC5
- ▶ Atualize o Node e o NPM
 - ▶ Node 6
 - ▶ NPM 3



ANGULAR CLI: UPDATE RC5

```
> npm uninstall -g angular-cli  
> npm cache clean  
> npm install -g angular-cli@webpack
```



ANGULAR CLI: UPDATE RC5



```
> ng new meu-projeto  
> cd meu-projeto  
> rm -rf node_modules dist tmp typings  
> npm install --save-dev angular-cli@webpack  
> ng init
```



ANGULAR CLI: UPDATE RC5

- ▶ Depois da atualização, os seguintes arquivos podem ser removidos:
 - ▶ `./config/karma-test-shim.js`
 - ▶ `./config/environment.js`
 - ▶ `./src/system-config.ts`
 - ▶ `./angular-cli-build.js`
 - ▶ `./typings.json`
 - ▶ `./.clang-format` (se existir)



19 Angular 2

Angular CLI:
Usando Sass, Less ou Stylus



ESCOLHENDO O PRÉ-PROCESSADOR

AO GERAR UM NOVO PROJETO

```
> ng new meu-projeto --style=sass  
> ng new meu-projeto --style=less  
> ng new meu-projeto --style=stylus
```



ESCOLHENDO O PRÉ-PROCESSADOR

EM UM PROJETO EXISTENTE

```
> ng set defaults.styleExt scss  
> ng set defaults.styleExt less  
> ng set defaults.styleExt styl
```



AO CRIAR UM NOVO COMPONENTE

```
▼ └── teste
    ├── index.ts
    ├── teste.component.html
    └── teste.component.scss
    ├── teste.component.spec.ts
    └── teste.component.ts
```



20 Angular 2

Angular CLI:

`ng lint, ng test, ng e2e`



```
loiane:_04-diretivas loiane$ ng lint
Could not start watchman; falling back to NodeWatcher for file system events.
Visit http://ember-cli.com/user-guide/#watchman for more info.

> -04-diretivas@0.0.0 lint /Users/loiane/Documents/development/angular2-precurso/_04-diretivas
> tslint "src/**/*.ts"

src/app/diretiva-ngif/diretiva-ngif.component.ts[15, 25]: missing whitespace
src/app/diretiva-ngif/diretiva-ngif.component.ts[11, 9]: expected nospace before colon in property-declaration
src/app/diretiva-ngif/diretiva-ngif.component.ts[13, 16]: expected nospace before colon in property-declaration
src/app/diretiva-ngswitch/diretiva-ngswitch.component.ts[11, 11]: expected nospace before colon in property-declaration
src/system-config.ts[1, 1]: " should be '
src/system-config.ts[21, 12]: missing whitespace

Lint errors found in the listed files.
```

ANG LINT

```
loiane:_04-diretivas loiane$ ng lint
Could not start watchman; falling back to NodeWatcher for file system events.
Visit http://ember-cli.com/user-guide/#watchman for more info.

> -04-diretivas@0.0.0 lint /Users/loiane/Documents/development/angular2-precurso/_04-diretivas
> tslint "src/**/*.ts"

All files pass linting.
loiane:_04-diretivas loiane$
```



loiane:_04-diretivas — Google Chrome Helper < angular-cli TERM_PROGRAM=Apple_Terminal M2=/Users/loiane/...

```
[loiane:_04-diretivas loiane$ ng test
Could not start watchman; falling back to NodeWatcher for file system events.
Visit http://ember-cli.com/user-guide/#watchman for more info.
Built project successfully. Stored in "dist/".
```

Build successful - 1237ms.

Slowest Trees	Total
---------------	-------

BroccoliTypeScriptCompiler	560ms
vendor	450ms
HandlebarReplace	172ms

Slowest Trees (cumulative)	Total (avg)
----------------------------	-------------

BroccoliTypeScriptCompiler (1)	560ms
vendor (1)	450ms
HandlebarReplace (1)	172ms

```
06 08 2016 11:33:31.323:WARN [karma]: No captured browser, open http://localhost:9876/
```

```
06 08 2016 11:33:31.343:INFO [karma]: Karma v0.13.22 server started at http://localhost:9876/
```

```
06 08 2016 11:33:31.348:INFO [launcher]: Starting browser Chrome
```



Angular 2

Angular CLI:

Estrutura do projeto



PRA QUE SERVE O ANGULAR CLI?

- ▶ Cria toda a estrutura do projeto
- ▶ Gera página HTML inicial, arquivos Typescript iniciais, arquivos CSS e arquivos de testes unitários
- ▶ Cria arquivo package.json com todas as dependências do Angular 2
- ▶ Instala todas as dependências do node (npm install)
- ▶ Configura o Karma para executar os testes unitários com Jasmine
- ▶ Configura Protractor para executar os testes end-to-end (E2E)
- ▶ Inicializa um repositório git no projeto e faz o commit inicial
- ▶ Cria todos os arquivos necessários para fazer o build da aplicação para produção



ESTRUTURA DIRETÓRIOS DO PROJETO

- ▼ projeto-teste
 - ▶ .git
 - ▶ config
 - ▶ config - diretório que contém configuração para deploy/build e teste.
 - ▶ e2e
 - ▶ dist - diretório onde é gerado o build da aplicação. Ignorado pelo git
 - ▶ node_modules
 - ▶ e2e - diretório que contém os scripts para testes end-to-end
 - ▶ public
 - ▶ node_modules - diretório que contém os pacotes npm da app (package.json). Também ignorado pelo git
 - ▶ src
 - ▶ public - diretório genérico que contém um arquivo .npmignore
 - ▶ README.md
 - ▶ angular-cli-build.js
 - ▶ angular-cli.json
 - ▶ package.json
 - ▶ tslint.json
 - ▶ typings.json



ESTRUTURA CÓDIGO FONTE (SCR)

```
▼ └── src
    └── app
        ├── meu-primeiro
        ├── shared
        │   └── .DS_Store
        ├── app.component.css
        ├── app.component.html
        ├── app.component.spec.ts
        ├── app.component.ts
        ├── environment.ts
        ├── index.ts
        ├── meu-primeiro.service.spec.ts
        └── meu-primeiro.service.ts
    └── .DS_Store
    └── favicon.ico
    └── index.html
    ├── main.ts
    ├── system-config.ts
    └── tsconfig.json
    └── typings.d.ts
```

- ▶ `index.html` - página HTML principal da aplicação, que faz o startup.
- ▶ `main.ts` é o código que carrega a aplicação. Somente deve ser editado caso seja necessário adicionar mais módulos na app.
- ▶ `system-config.ts` configura as dependências da app. Bibliotecas e extensões angular 2 devem ser adicionadas aqui para que sejam carregadas na app.
- ▶ `tsconfig.json` contém as configurações do compilador do typescript
- ▶ `typings.d.ts` é usado para declarações de tipos que a app usa
- ▶ `index.ts` contém o export de todos os arquivos do módulo



LEMBRE-SE SEMPRE DE CONSULTAR A DOCUMENTAÇÃO!

A DOCUMENTAÇÃO É A SUA MELHOR AMIGA!
DESENVOLVA O HÁBITO DE LER OS DOCS!

► <https://angular.io/docs>



ESTRUTURA PACKAGE.JSON

DEPENDENCIES X DEVDEPENDENCIES

- ▶ `dependencies`: dependências necessárias para executar a aplicação.
- ▶ `devDependencies`: dependências necessárias para desenvolver a aplicação (não necessárias após o build de produção).



ESTRUTURA PACKAGE.JSON: DEPENDENCIES

```
"dependencies": {  
    "@angular/common": "2.0.0-rc.4",  
    "@angular/compiler": "2.0.0-rc.4",  
    "@angular/core": "2.0.0-rc.4",  
    "@angular/forms": "0.2.0",  
    "@angular/http": "2.0.0-rc.4",  
    "@angular/platform-browser": "2.0.0-rc.4",  
    "@angular/platform-browser-dynamic": "2.0.0-rc.4",  
    "@angular/router": "3.0.0-beta.2",  
    "es6-shim": "0.35.1",  
    "reflect-metadata": "0.1.3",  
    "rxjs": "5.0.0-beta.6",  
    "systemjs": "0.19.26",  
    "zone.js": "0.6.12"  
},
```



ESTRUTURA PACKAGE.JSON: DEPENDENCIES

- ▶ @angular/core - pacote principal do framework Angular 2. Contém decorators e metadados, Component, Directive, injeção de dependência e os hooks de ciclo de vida do Component.
- ▶ @angular/common - Serviços, pipes e diretivas comuns fornecidas pelo time de dev do Angular.
- ▶ @angular/compiler - Template de compilação do angular. Entende o código dos templates e converte em código que faz a app ser executada e renderizada. Desenvolvedores não interagem com esse pacote diretamente (apenas usamos seu código).
- ▶ @angular/forms - contém todo o código para construção de formulários no angular 2.
- ▶ @angular/platform-browser - contém todo o código relacionado ao DOM e ao browser, especialmente as parte que ajudam a renderizar o DOM. Esse pacote também contém o método para fazer o bootstrap da aplicação para builds de produção que pré-compila os templates.
- ▶ @angular/platform-browser-dynamic - Contém os Providers e o método para iniciar as aplicações que compilam templates no lado cliente. Não usa compilação offline. Usada para fazer bootstrap durante desenvolvimento e exemplos plunker.
- ▶ @angular/http - fornece o cliente HTTP.
- ▶ @angular/router - classes de roteamento.



ESTRUTURA PACKAGE.JSON: DEPENDENCIES: POLYFILLS

- ▶ es6-shim - biblioteca que permite compatibilidade de engines JS antigas com a especificação do ES 2015, ou seja, emula as funcionalidades do ES 2015 em browsers que suportam somente ES5.
- ▶ reflect-metadata - dependência compartilhada entre o Angular e o compilador TypeScript. Permite o uso de decorators no código (annotations). Isso permite ao desenvolvedores fazer upgrade no TypeScript sem precisar de fazer upgrade no Angular. Esse é o motivo desta ser uma dependência da aplicação e não do angular.
- ▶ rxjs - extensão para a especificação dos Observables (programação assíncrona). Reactive extensions for JavaScript.
- ▶ system.js - módulo para carregamento dinâmico compatível com a especificação de módulos do ES 2015. Como alternativa também podemos usar o Webpack. O SystemJS é o módulo utilizando oficialmente na documentação do Angular 2.
- ▶ zone.js - extensão (plugins) útil para tarefas assíncronas (chamadas de Zones).

A ESTRUTURA PACKAGE.JSON: DEVDEPENDENCIES

```
"devDependencies": {  
  "angular-cli": "1.0.0-beta.10",  
  "codelyzer": "0.0.20",  
  "ember-cli-inject-live-reload": "1.4.0",  
  "jasmine-core": "2.4.1",  
  "jasmine-spec-reporter": "2.5.0",  
  "karma": "0.13.22",  
  "karma-chrome-launcher": "0.2.3",  
  "karma-jasmine": "0.3.8",  
  "protractor": "3.3.0",  
  "ts-node": "0.5.5",  
  "tslint": "3.11.0",  
  "typescript": "1.8.10",  
  "typings": "1.3.1"  
}
```



ESTRUTURA PACKAGE.JSON: DEPENDENCIES: POLYFILLS

- ▶ angular-cli: ferramenta de linha de comando para gerenciar projetos angular 2.
- ▶ codelyzer: lint (análise de código) para angular 2
- ▶ ember-cli-inject-live-reload: ferramenta de live reload
- ▶ jasmine-core: arquivos principais jasmine para node.js
- ▶ jasmine-spec-reporter: relatório em tempo real para BDD com Jasmine
- ▶ karma: ferramenta de testes que cria um web server e executa código de teste em cada um dos browsers conectados
- ▶ karma-chrome-launcher: launcher do karma para o chrome
- ▶ karma-jasmine: adaptador para o jasmine
- ▶ protractor: framework de teste end to end (integração) para angular
- ▶ ts-node: módulo typescript para node.js
- ▶ tslint: lint (análise de código) para typescript
- ▶ typescript: compilador typescript
- ▶ typings: gerenciador de arquivos de definição para typescript



22 Angular 2

Angular CLI:

Gerando o build de produção



GERANDO O BUILD DE DESENVOLVIMENTO



```
> cd NomeProjeto  
  
> ng build --target=development --environment=dev  
  
> ng build --dev --e=dev  
  
> ng build --dev  
  
> ng build
```



BUILD DE DEV

```
✓ └── dist
    └── app
        > └── meu-primeiro
        > └── shared
        ┌── .DS_Store
        └── app.component.css
        └── app.component.html
        └── app.component.js
        ┌── app.component.js.map
        └── app.component.spec.js
        ┌── app.component.spec.js.map
        └── environment.js
        ┌── environment.js.map
        └── index.js
        ┌── index.js.map
        └── meu-primeiro.service.js
        ┌── meu-primeiro.service.js.map
        └── meu-primeiro.service.spec.js
        ┌── meu-primeiro.service.spec.js.map
```

- ▶ Útil para integrar o código do Angular 2 com o projeto de backend (PHP, Java, .NET, Python, Ruby etc)
- ▶ Código que dá pra debugar
- ▶ Apenas copia o código de desenvolvimento para o diretório *dist* (apenas código .js)



GERANDO O BUILD DE PRODUÇÃO



```
> cd NomeProjeto  
> ng build --target=production --environment=prod  
> ng build --prod --env=prod  
> ng build --prod
```



BUILD DE PROD

```
✓ └─ dist
    └─ app
        └─ meu-primeiro
            └─ meu-primeiro.component.css
            └─ meu-primeiro.component.html
            └─ .DS_Store
        └─ app.component.css
        └─ app.component.html
    └─ vendor
        └─ .DS_Store
    └─ .npmignore
    └─ favicon.ico
    └─ index.html
    └─ main.js
    └─ system-config.js
```

- ▶ Obfusca e minifica o código JS da aplicação
- ▶ Arquivos CSS e templates HTML não são minificados
- ▶ diretório *vendor* contém todo o código Angular 2 + dependências



MUDAR O DIRETÓRIO DE OUTPUT

CRIAR ARQUIVO .EMBER-CLI NO PROJETO

```
□ .ember-cli
1  {
2      "output-path": ".../contacts-spring-mvc/WebContent/"
3  }
```



MUDAR O DIRETÓRIO DE OUTPUT

CRIAR ARQUIVO .EMBER-CLI NO PROJETO

```
□ .ember-cli
1  {
2      "output-path": ".../contacts-spring-mvc/WebContent/"
3  }
```

TOMAR CUIDADO POIS SE O DIRETÓRIO JÁ EXISTIR, O MESMO SERÁ DELETADO
E RECREADO



EXEMPLO DE ARQUITETURA DOS PROJETOS

```
▼ └ angular-cli
    ▼ └ contacts-spring-mvc
        > └ .settings
        > └ build
        > └ sql
        > └ src
        ▼ └ WebContent
            > └ app
            > └ vendor
            ┌ .DS_Store
            ┌ .npmignore
            ┌ favicon.ico
            ┌ index.html
            ┌ main.js
            ┌ system-config.js
            ┌ .classpath
            ┌ .DS_Store
            ┌ .project
            ┌ readme
        > └ projeto-teste
```



EXEMPLO DE ARQUITETURA DOS PROJETOS

```
angular-cli
  contacts-spring-mvc
    .settings
    build
    sql
    src
  WebContent
    app
    vendor
    .DS_Store
    .npmignore
    favicon.ico
    index.html
    main.js
    system-config.js
    .classpath
    .DS_Store
    .project
    readme
  projeto-teste
```

WORKSPACE DO PROJETO ANGULAR2 + JAVA



EXEMPLO DE ARQUITETURA DOS PROJETOS

```
└── angular-cli
    ├── contacts-spring-mvc
    │   ├── .settings
    │   ├── build
    │   ├── sql
    │   └── src
```

```
    └── WebContent
        ├── app
        ├── vendor
        ├── .DS_Store
        ├── .npmignore
        ├── favicon.ico
        ├── index.html
        ├── main.js
        └── system-config.js
```

```
    .classpath
```

```
    .DS_Store
```

```
    .project
```

```
    readme
```

```
    projeto-teste
```

WORKSPACE DO PROJETO ANGULAR2 + JAVA

OUTPUT BUILD PROJETO ANGULAR 2



EXEMPLO DE ARQUITETURA DOS PROJETOS

```
angular-cli
  contacts-spring-mvc
    .settings
    build
    sql
    src
  WebContent
    app
    vendor
    .DS_Store
    .npmignore
    favicon.ico
    index.html
    main.js
    system-config.js
  .classpath
  .DS_Store
  .project
  readme
  projeto-teste
```

WORKSPACE DO PROJETO ANGULAR2 + JAVA

OUTPUT BUILD PROJETO ANGULAR 2

PROJETO ANGULAR 2



23 Angular 2

Angular CLI:

instalando bibliotecas

(bootstrap, jquery, semantic ui, etc)



INSTALANDO A DEPENDÊNCIA

<https://github.com/valor-software/ng2-bootstrap>



```
> cd NomeProjeto  
> npm install ng2-bootstrap --save  
> npm install moment --save
```



PROJETO/SRC/SYSTEM-CONFIG.TS

```
/******  
 * User Configuration.  
*****  
/** Map relative paths to URLs. */  
const map: any = {  
  'moment': 'vendor/moment/moment.js',  
  'ng2-bootstrap': 'vendor/ng2-bootstrap'  
};  
  
/** User packages configuration. */  
const packages: any = {  
  'ng2-bootstrap': {  
    defaultExtension: 'js'  
  },  
  'moment': {  
    format: 'cjs'  
  }  
};
```

```
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">
```

The logo consists of a red 3D hexagonal prism with the number '24' in white on its front face and the word 'Angular' in white on its right face.

24 Angular 2

A faint blue silhouette of a person standing on a bridge, looking out over a city skyline with various buildings and landmarks.

Diretivas: Introdução



O QUE SÃO DIRETIVAS?

DIRETIVAS SÃO INSTRUÇÕES

```
<ul>
  <li *ngFor="let curso of cursos">
    {{ curso }}
  </li>
</ul>
```



O QUE SÃO DIRETIVAS?

DIRETIVAS SÃO INSTRUÇÕES

```
<ul>
  <li *ngFor="let curso of cursos">
    {{ curso }}
  </li>
</ul>
```

"Itere todos os **cursos**, e a cada iteração, atribua o valor do elemento atual à uma variável **curso**. Replique também o elemento **** com valor da variável **curso** a cada iteração."



O QUE SÃO DIRETIVAS?

DIRETIVAS SÃO INSTRUÇÕES

```
template:  
  <cursos-lista></cursos-lista>
```



O QUE SÃO DIRETIVAS?

DIRETIVAS SÃO INSTRUÇÕES

```
template:  
  <cursos-lista></cursos-lista>
```

“Crie um componente do Tipo (classe) especificado e renderize a view (template) desse componente nesse lugar”.



TIPOS DE DIRETIVAS

DIRETIVAS ESTRUTURAIS

Interagem com a view e
modificam a estrutura do DOM
e/ou código HTML

`*ngFor`

`*ngIf`



TIPOS DE DIRETIVAS

DIRETIVAS ESTRUTURAIS

Interagem com a view e modificam a estrutura do DOM e/ou código HTML

`*ngFor`

`*ngIf`

DIRETIVAS DE ATRIBUTOS

Interagem com o elemento em que foram aplicadas

`ng-class`

`ng-style`



25 Angular 2



*nglf

A CONDICIONAL IF

```
var cursos = [];  
  
if (cursos.length > 0){  
    // alguma lógica de programação  
} else {  
    // alguma outra lógica de programação  
}
```



```
<div *ngIf="cursos.length > 0">
  Lista de cursos aqui
</div>
<div *ngIf="cursos.length == 0">
  Não existem cursos para serem listados.
</div>
```



[DICA] ALTERNATIVA: HIDDEN

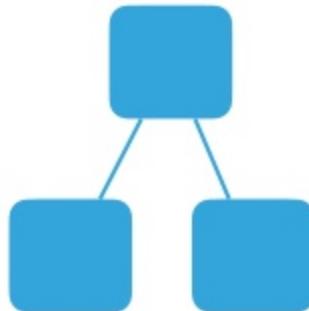
```
<div [hidden]="!mostrarCursos">  
  Lista de cursos aqui  
</div>  
<div [hidden]="mostrarCursos">  
  Não existem cursos para serem listados.  
</div>
```



*NGIF X HIDDEN

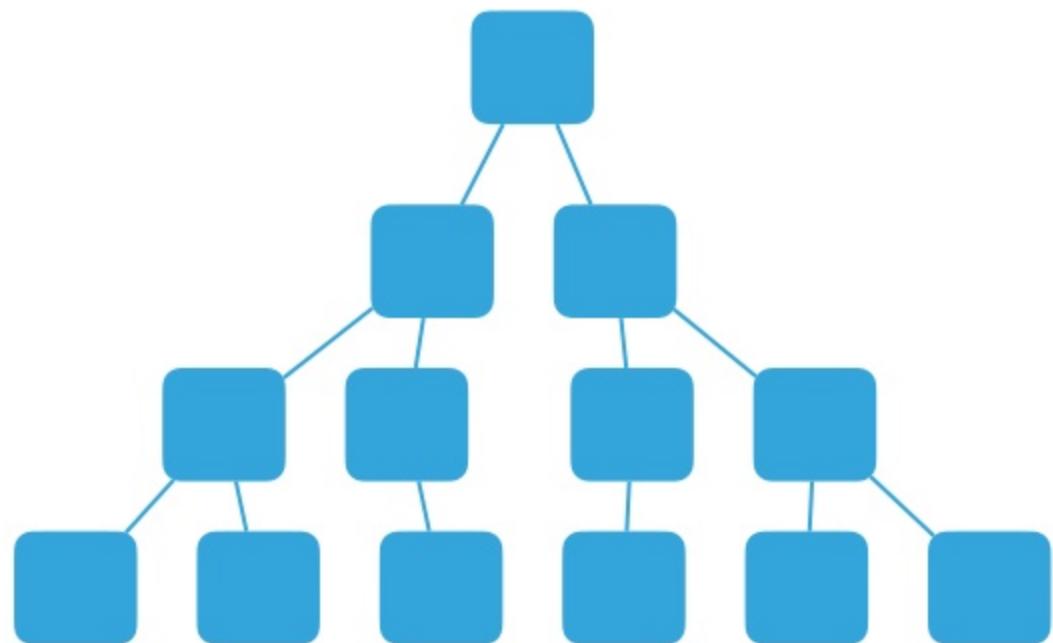
[hidden]

recomendado para árvore de elementos pequenas



*ngIf

recomendado para árvore de elementos grandes

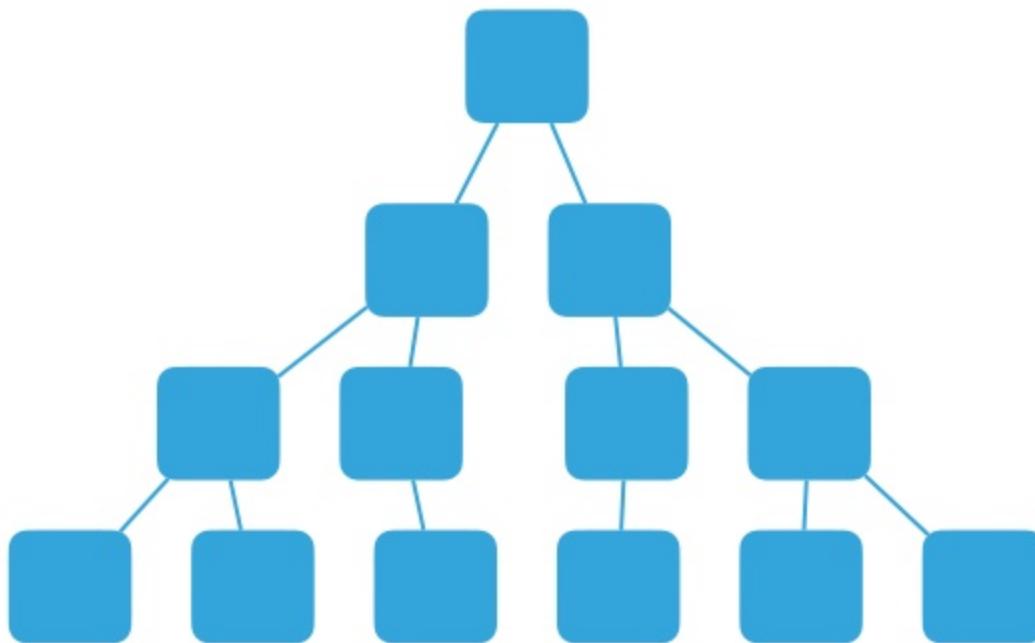




*NGIF X HIDDEN - EXCEÇÃO

[hidden]

é menos custoso usar [hidden] caso o custo de
criar a árvore de elementos seja grande





26 Angular 2

ngSwitch e *ngSwitchCase



CONDICIONAL SWITCH-CASE

```
var viewMode = 'mapa';

switch (viewMode){
    case 'mapa': //lógica mapa;
        break;
    case 'lista': //lógica lista;
        break;
    default: // lógica padrão
}
```



ANGSWITCH E *ANGSWITCHCASE

```
<ul class="nav nav-pills">
  <li [class.active]="viewMode == 'mapa'">
    <a (click)="viewMode= 'mapa'">Mapa</a>
  </li>
  <li [class.active]="viewMode == 'lista'">
    <a (click)="viewMode= 'lista'">Lista</a>
  </li>
</ul>

<div [ngSwitch]="viewMode">
  <p *ngSwitchCase="'mapa'">Modo mapa ativado</p>
  <p *ngSwitchCase="'lista'">Modo lista ativado</p>
  <p *ngSwitchDefault>Modo padrão ativado</p>
</div>
```



27 Angular 2



*ngFor

A LOOP FOR

```
for (let i = 0; i < cursos.length; i++) {  
  let curso = cursos[i];  
}
```



*NGFOR

```
<ul>
  <li *ngFor="let curso of cursos, let i = index">
    {{ i + 1 }} - {{ curso }}
  </li>
</ul>
```



28 Angular 2

Diretivas:

o porquê de usar o *
e <template>



```
<div template="ngIf mostrarCursos">
  Lista de cursos aqui
</div>
```

```
<template [ngIf]="mostrarCursos">
  <div>Lista de cursos aqui</div>
</template>
```



NGSWITCH E *NGSWITCHCASE

```
<div [ngSwitch]="viewMode">
  <template [ngSwitchCase]="'mapa'" ngSwitchDefault>
    Modo mapa ativado
  </template>
  <template [ngSwitchCase]="'lista'">
    Modo lista ativado
  </template>
</div>
```



NGFOR

```
<ul>
  <li template="ngFor let curso of cursos, let i = index">
    {{ i + 1 }} - {{ curso }}
  </li>
</ul>

<ul>
  <template ngFor let-curso [ngForOf]="cursos" let-i="index">
    <li> {{ i + 1 }} - {{ curso }} </li>
  </template>
</ul>
```



29 Angular 2



ngClass



```
<h1>
  <i class="glyphicon"
    [class.glyphicon-star-empty]="!meuFavorito"
    [class.glyphicon-star]="meuFavorito"
    (click)="onClick()"
  ></i>
</h1>
```



```
<h1>
  <i class="glyphicon"
    (click)="onClick()"
    [ngClass]="{
      'glyphicon-star-empty' : !meuFavorito,
      'glyphicon-star' : meuFavorito
    }"
  ></i>
</h1>
```



30 Angular 2



ngStyle



NGSTYLE



NGSTYLE

The logo consists of a red 3D hexagonal prism with the number '31' in white on its front face.

31 Angular 2

A faint silhouette of a person standing in front of a city skyline is visible at the bottom of the slide.

Operador elvis



ELVIS OPERATOR ?

32 Angular 2

ng-content



<https://github.com/loiane/curso-angular2>

loiane.training/curso/angular-2/

The screenshot shows the Angular 2 course page. At the top, there's a navigation bar with links for CURSOS, LOGIN / SIGNUP, CURSOS, SOBRE, AJUDA, BLOG, and a search icon. Below the navigation is a banner with the text "Angular 2" and "GRÁTIS". On the left, there's a blue sidebar with the Angular logo and the text "Angular 2". The main content area has a pink header bar with "HOME". The main text describes the course as a framework for developing web and mobile applications, mentioning its reimplementation from scratch. It highlights features like components, templates, directives, services, data-binding, reactive forms, and routing. It also states that no prior knowledge of Angular 1 is required. The sidebar on the left lists course details: "EM BREVE", "ACESSO ILIMITADO", "ACESSO ILIMITADO", "1 AULAS", and "15 MINUTOS", each with a corresponding icon.

Angular 2 é um framework que permite desenvolver aplicações web e mobile, mantido pela Google. Apesar de ser a segunda versão do framework, Angular 2 não é a continuação do Angular 1 com melhores e novas funcionalidades, foi reescrito.

Nesse curso vamos aprender como desenvolver com Angular 2 e TypeScript. Você vai aprender os conceitos básicos e intermediários do Angularjs, desde os primeiros passos até uma aplicação completa com requisições ao servidor e rotas.

Vamos aprender os conceitos de componentes, templates, diretivas, serviços, data-binding, validação de formulários, extensões reactive, roteamento (single page application - SPA) e como conectar ao servidor.

Não é necessário saber Angular 1 para esse curso, já que o Angular 2 foi reescrito. Vamos aprender todos os conceitos do zero, então não se preocupe!

Nível: Iniciante ao Intermediário

Curso em andamento – as aulas ainda estão sendo publicadas

Link com o calendário/agenda das aulas: <https://github.com/loianetraining/calendario>



**Material apoio + certificado do curso
Cadastro em:**

<http://loiane.training>

Obrigada



<http://loiane.training>



<http://loiane.com>



<facebook.com/loianegroner>



<twitter.com/loiane>



<https://github.com/loiane>



<youtube.com/loianegroner>