

Bootstrapping

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

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

Bootstraps the app, using the root component from the specified NgModule.

NgModules

Template syntax

```
<input [value]="-
 firstName">
```

Binds property *value* to the result of expression *firstName*

```
<div [attr.rol-
 e]="myAriaRol-
 e">
```

Binds attribute role to the result of expression *myAriaRole*.

```
<div [class.ex-
 tra-sparkle]="is-
 Delightful">
```

Binds the presence of the CSS class *extra-sparkle* on the element to the truthiness of the expression *isDelightful*.

```
<div [style.wi-
 dth.px]="mySi-
 ze">
```

Binds style property width to the result of expression *mySize* in pixels. Units are optional.

```
<button (click)="read-
 Rainbow($eve-
 nt)">
```

Calls method *readRainbow* when a click event is triggered on this button element (or its children) and passes in the event object.

```
<div title="Hello
 {{ponyName}}">
```

Binds a property to an interpolated string, for example, "Hello Seabiscuit". Equivalent to: `<div [title]="Hello ' + ponyName">`

```
<p>Hello
 {{ponyName}}
 </p>
```

Binds text content to an interpolated string, for example, "Hello Seabiscuit".

```
<my-cmp [(titl-
 e)]= "name">
```

Sets up two-way data binding. Equivalent to: `<my-cmp [title]="name" (titleChange)="name=$event">`

Template syntax (cont)

```
<video #movieplayer ...>
 <button (click)="movieplaye-
 r.play()"> </video>
```

Creates a local variable *movieplayer* that provides access to the video element instance in data-binding and event-binding expressions in the current template.

```
<p *myUnless="myExpres-
 sion">...</p>
```

The *** symbol turns the current element into an embedded template. Equivalent to: `<ng-template [myUnless]="myExpression"><p>... </p></ng-template>`

```
<p>Card No.: {{cardNumber |
 myCardNumberFormatter}}
 </p>
```

Transforms the current value of expression *cardNumber* via the pipe called *myCardNumberFormatter*.

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

@NgModule({ declarations: ..., imports: ..., exports: ..., providers: ..., bootstrap: ... })
class MyModule {}

declarations: [MyRedComponent, MyBlueComponent, MyDatePipe]

imports: [BrowserModule, SomeOtherModule]

exports: [MyRedComponent, MyDatePipe]

providers: [MyService, { provide: ... }]

entryComponents: [SomeComponent, OtherComponent]

bootstrap: [MyAppComponent]

Defines a module that contains components, directives, pipes, and providers

Components, directives, and pipes that belong to this module

Modules to import into this module - Everything from the imported modules is available to declarations of this module

Components, directives, and pipes visible to modules that import this module

Dependency injection providers visible both to the contents of this module and to importers of this module

Components not referenced in any reachable template, for example dynamically created from code

Components to bootstrap when this module is bootstrapped

<p>Employer: {{employer?.companyName}}</p>

The safe navigation operator (?) means that the employer field is optional and if undefined, the rest of the expression should be ignored.

<svg:rect x="0" y="0" width="100" height="100"/>

An SVG snippet template needs an svg: prefix on its root element to disambiguate the SVG element from an HTML component.

<svg> <rect x="0" y="0" width="100" height="100"/></svg>

An <svg> root element is detected as an SVG element automatically, without the prefix.



By dmntswntns

Not published yet.
Last updated 27th August, 2019.
Page 1 of 2.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>