

JS frameworks: Angular

Single Page Applications

Single page apps typically have:

- “application-like” interaction
- dynamic data loading from the server-side API
- fluid transitions between page states
- more JavaScript than actual HTML

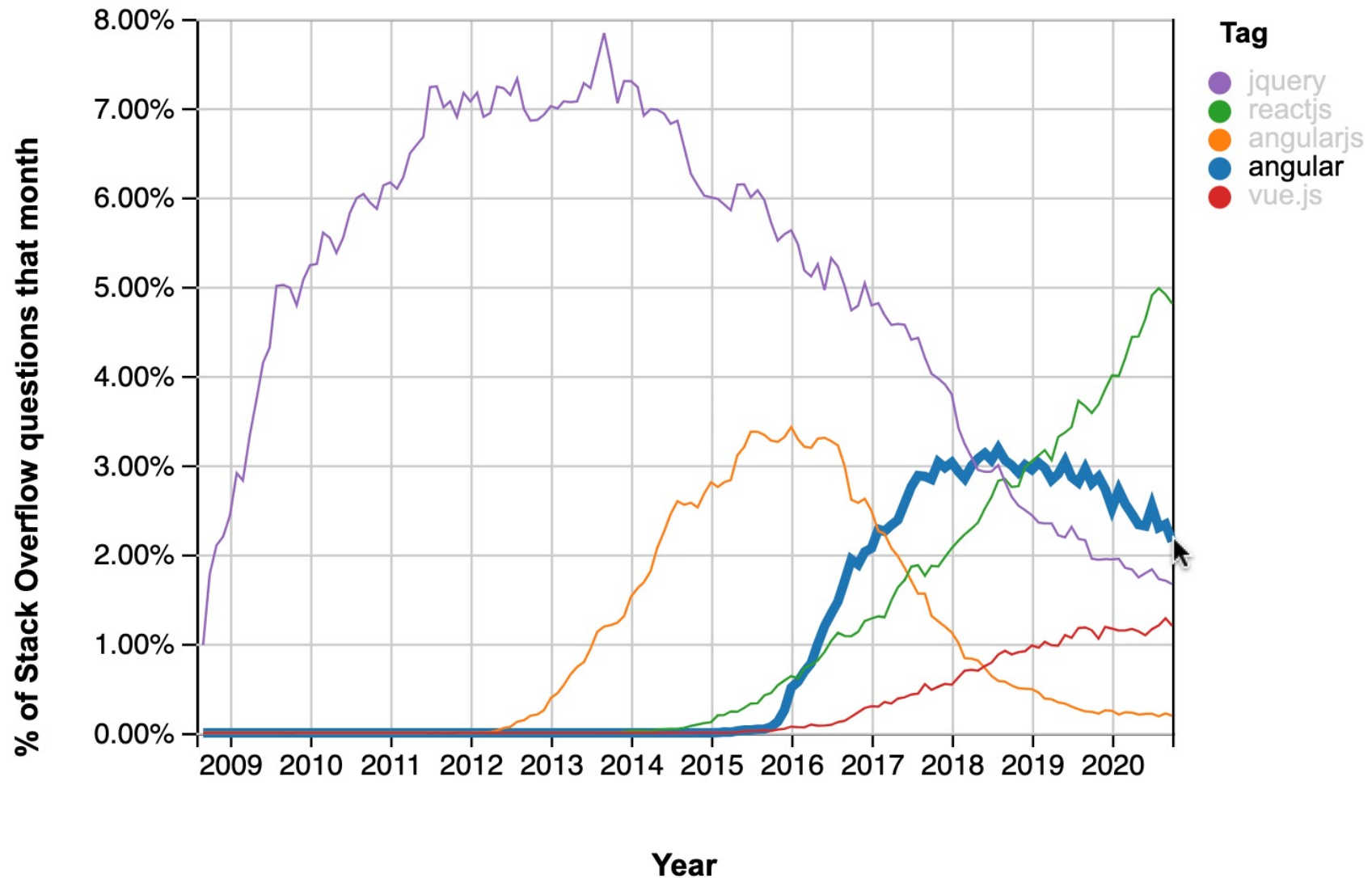
They typically do not have:

- support for crawlers (not for sites relying on search traffic) support for legacy browsers (IE7 or older, dumbphone browsers)

Problems with SPAs

- DOM Manipulation
How to manipulate the view efficiently?
 - Data Binding
How bind data from model to view?
 - View Loading
How to load the view?
 - History
What happens when pressing back button?
 - Routing
Readable URLs?
 - SEO
How to make content available to Search Engines?
- Lots of coding! You could use a framework instead ...

Jquery, Angular JS, Angular, React



<https://insights.stackoverflow.com/trends>

Angular

Angular provides a **framework for single-page apps**, where **most of the logic and data resides on the client**. Most apps still need to access a server using the HttpClient to access and save data.

Server-side rendering: allows you to run your Angular app **on the server**.

Angular Universal

Angular Universal generates static application pages on the server through server-side rendering (SSR).

Enabling server-side rendering:

- Facilitate web crawlers through search engine optimization (SEO)
- Improve performance on mobile and low-powered devices
- Show the first page quickly with a first-contentful paint (FCP)



see <https://angular.io/guide/universal>

<https://www.tektutorialshub.com/angular/server-side-rendering-using-angular-universal/>

Writing Angular apps

Angular apps are written by using:

- HTML
- TypeScript (or JS)
- ng primitives

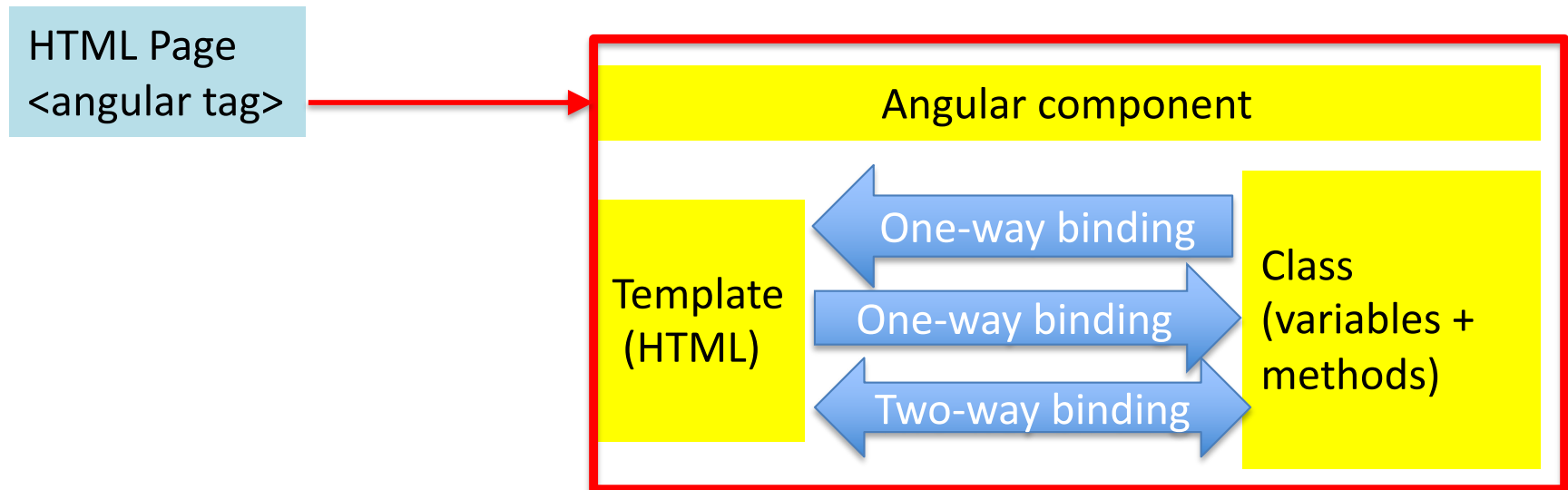
They can be deployed on any Web Server

TypeScript has to be transpiled

Q

**What is the overall
idea/architecture of Angular?**

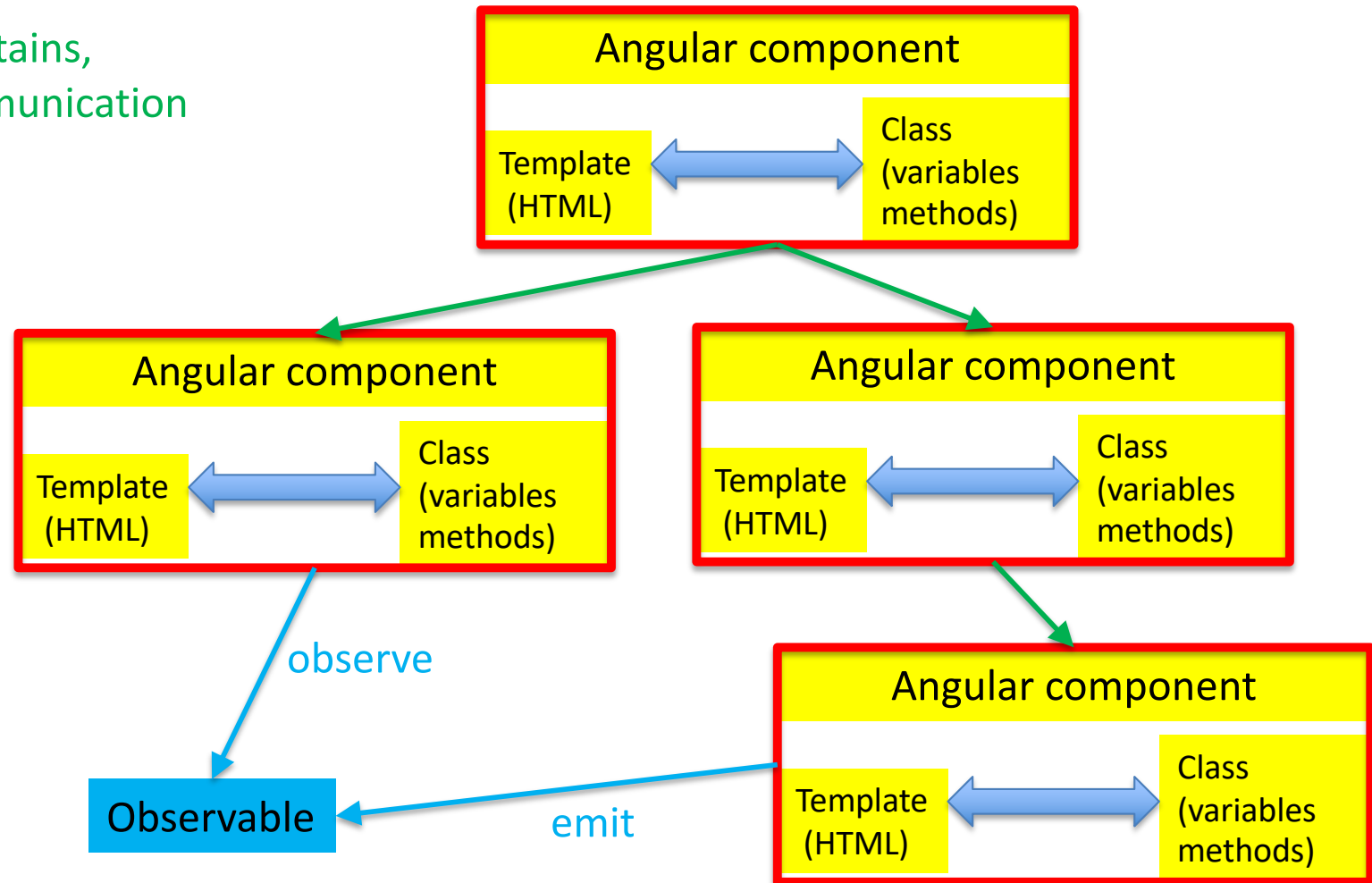
Angular architecture – Basic idea



Angular architecture

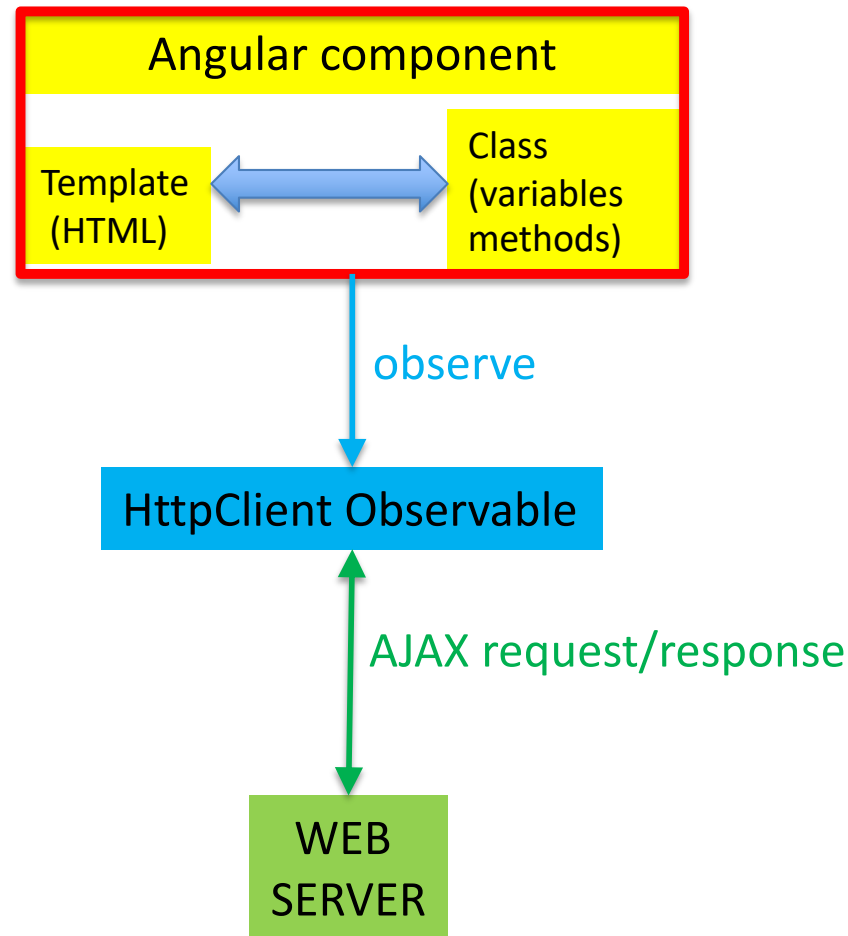
Multiple components

green arrow=contains,
father-child communication



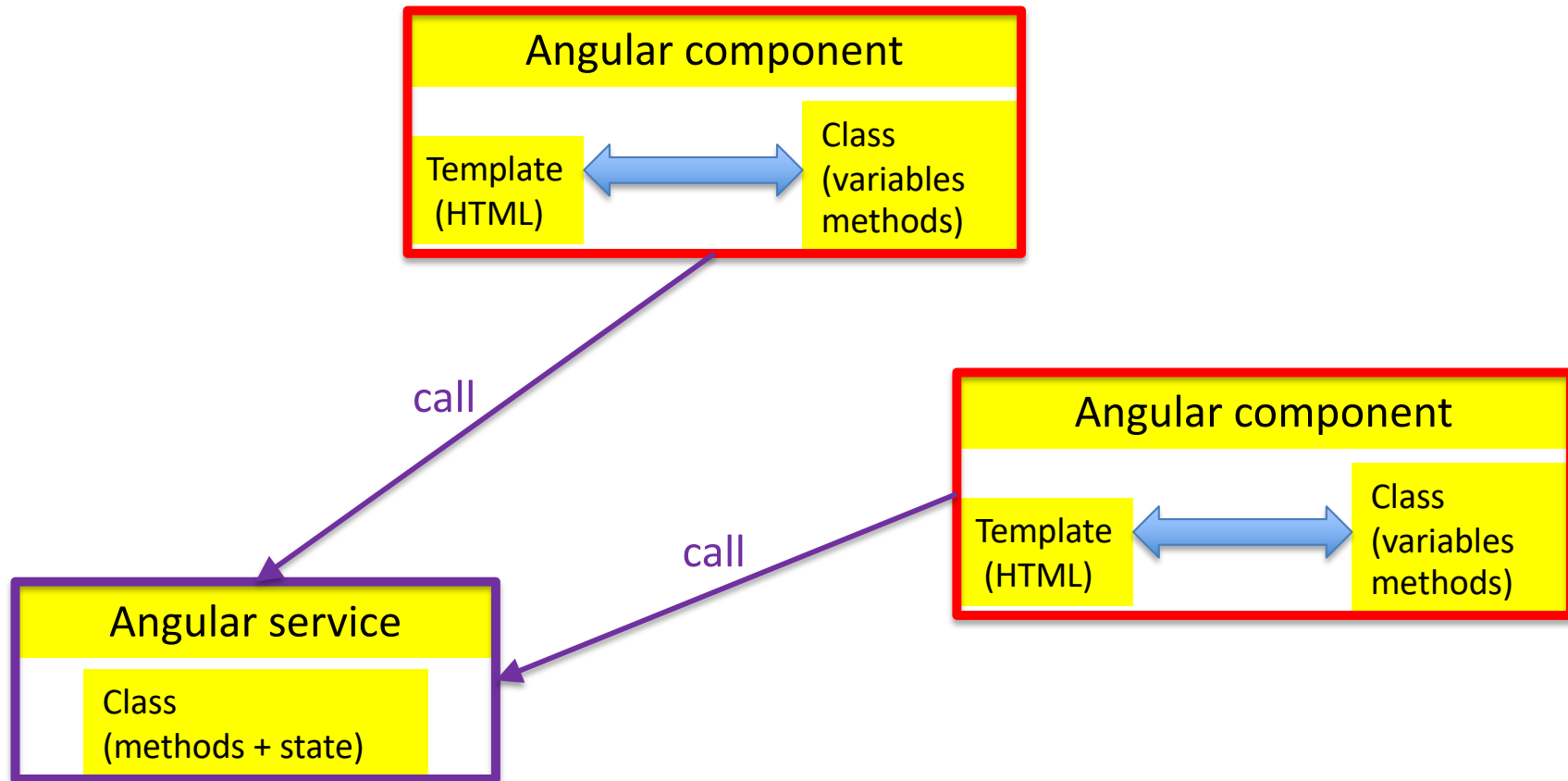
Angular architecture

Interacting with a web server



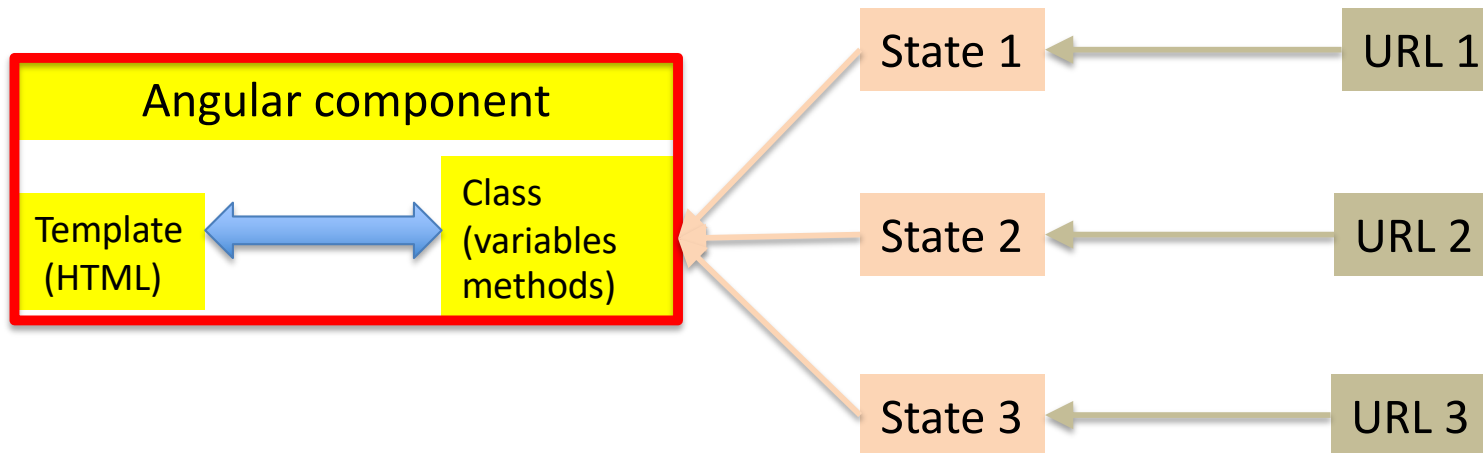
Angular architecture

Services



Angular architecture

Routing



Q

How do I set up the needed
environment?

Tools needed

Angular apps are written by using:

- HTML
- TypeScript (or JS)
- ng primitives

They can be deployed on any Web Server

TypeScript has to be transpiled

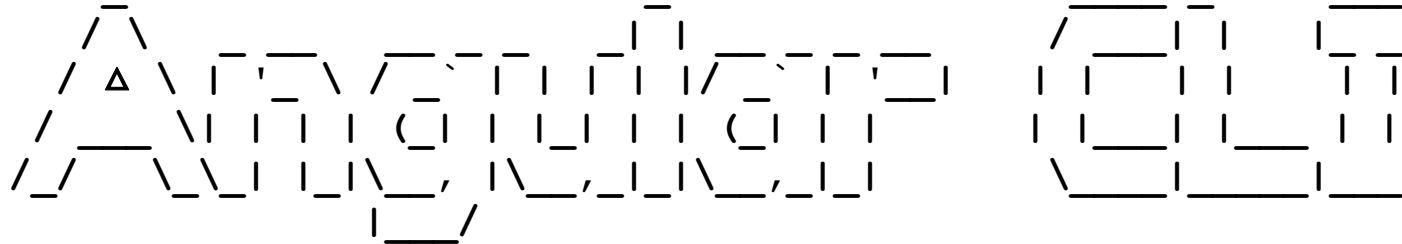
You need Angular Client Line Interface

<https://cli.angular.io/>

1. Install node
2. `npm install -g @angular/cli`

Checking installation

```
MR-MacBookPro:firstAngularProject ronchet$ ng --version
```



```
Angular CLI: 11.0.2
```

```
Node: 15.2.1
```

```
OS: darwin x64
```

```
Angular: 11.0.2
```

```
... animations, cli, common, compiler, compiler-cli, core, forms
```

```
... platform-browser, platform-browser-dynamic, router
```

```
Ivy Workspace: Yes
```

Package	Version

@angular-devkit/architect	0.1100.2
@angular-devkit/build-angular	0.1100.2
@angular-devkit/core	11.0.2
@angular-devkit/schematics	11.0.2
@schematics/angular	11.0.2
@schematics/update	0.1100.2
rxjs	6.6.3
typescript	4.0.5

Creating a new Angular project

MR-MacBookPro:include ronchet\$ **ng new firstAngularProject**

? Do you want to enforce stricter type checking and stricter bundle budgets in the workspace?

This setting helps improve maintainability and catch bugs ahead of time.

For more information, see <https://angular.io/strict> **Yes**

? Would you like to add Angular routing? **No**

? Which stylesheet format would you like to use?

> **CSS**

SCSS [<https://sass-lang.com/documentation/syntax#scss>]

Sass [<https://sass-lang.com/documentation/syntax#the-indented-syntax>]

Less [<http://lesscss.org>]

Stylus [<https://stylus-lang.com>]

CREATE firstAngularProject/README.md (1028 bytes)

CREATE firstAngularProject/.editorconfig (274 bytes)

...

CREATE firstAngularProject/e2e/src/app.e2e-spec.ts (670 bytes)

CREATE firstAngularProject/e2e/src/app.po.ts (274 bytes)

✓ Packages installed successfully.

Successfully initialized git.

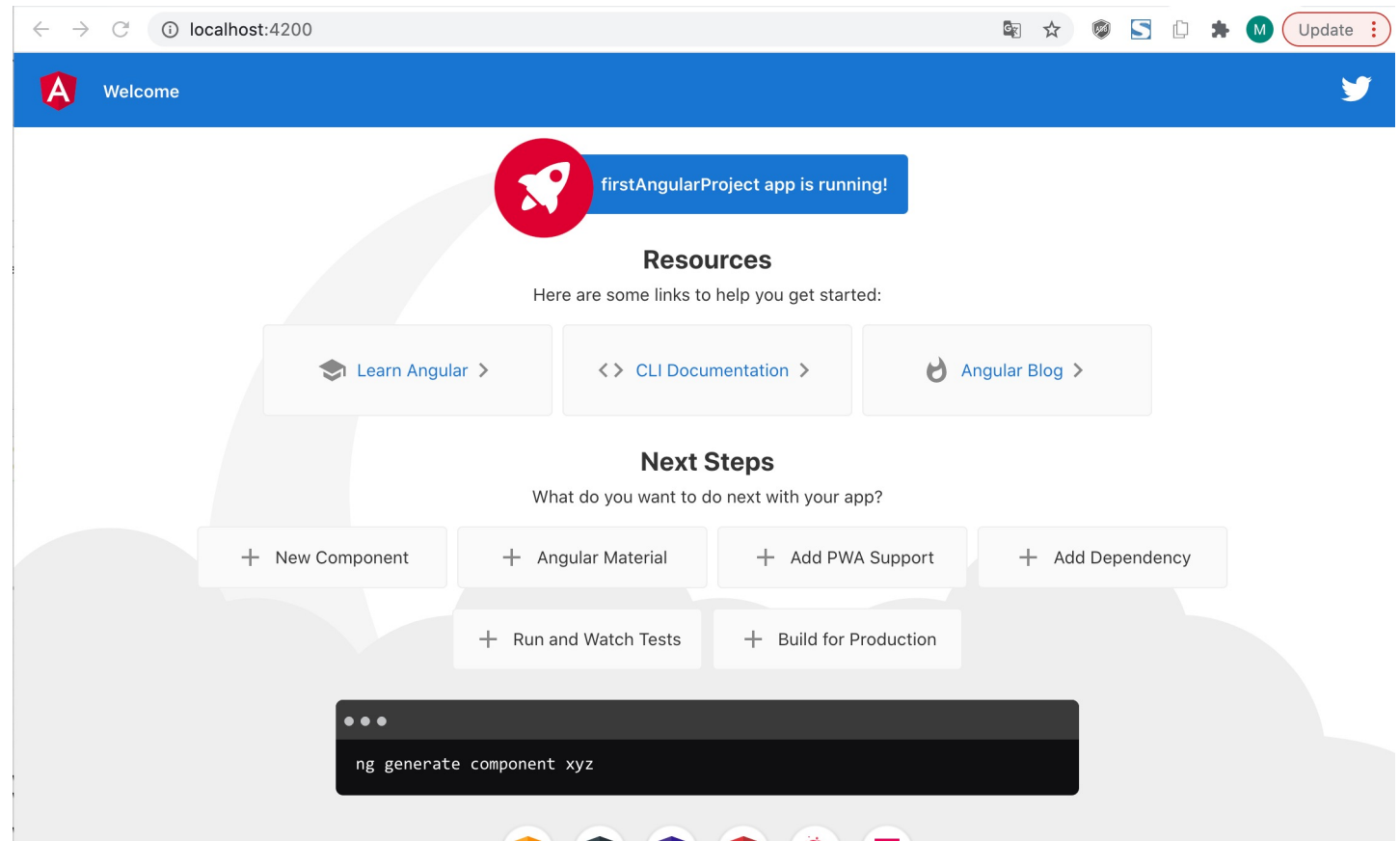
Running new Angular project with CLI

The Angular CLI includes a server, so that you can build and serve your app locally.

Navigate to the workspace folder, such as **firstAngularProject**.

Run the following command:

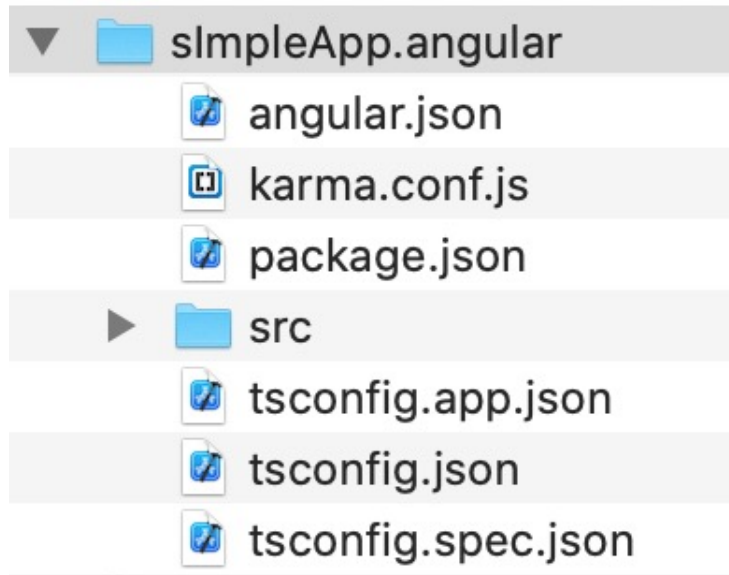
cd firstAngularProject
ng serve --open



Q

What is the structure of a basic app?

Package structure



angular.json: This is the configuration file for Angular CLI. The older versions of the Angular used the file angular-cli.json

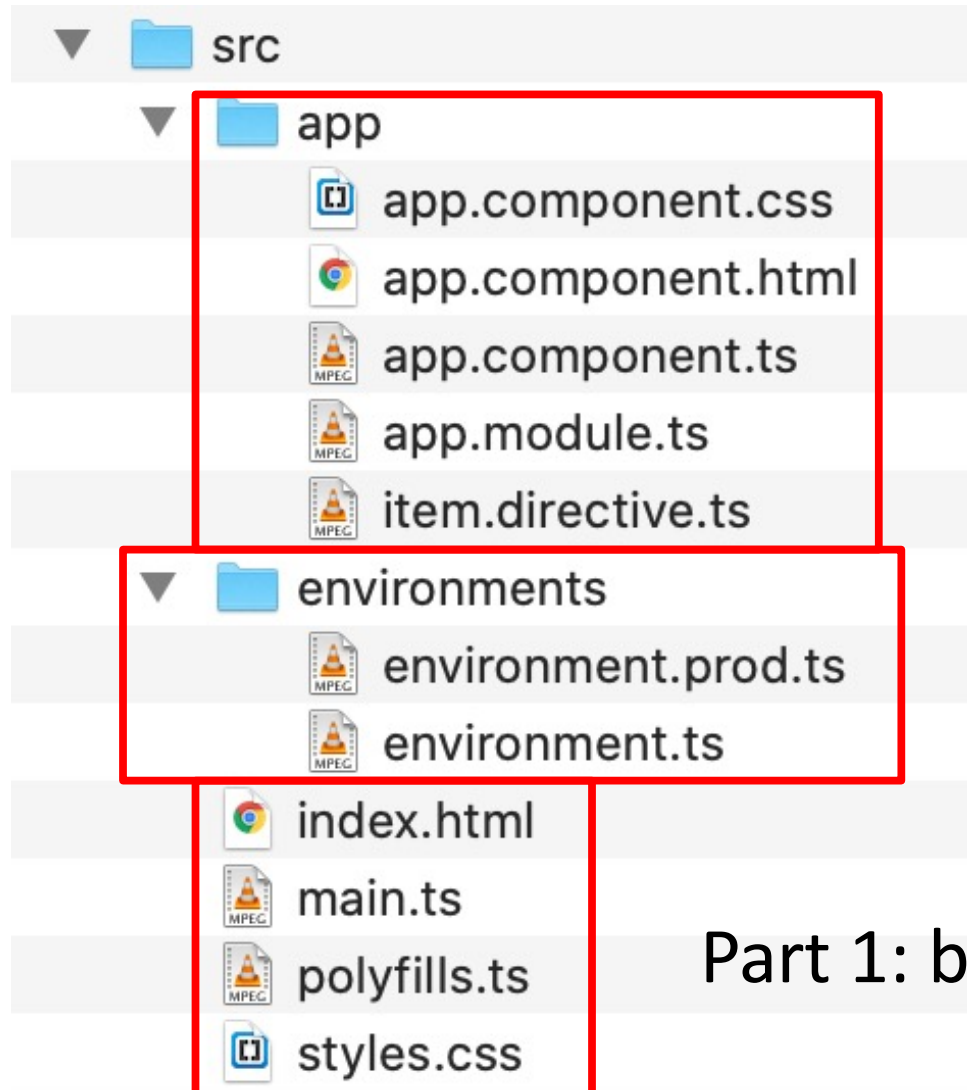
karma.conf.js: The Configuration file for the [karma test runner](#).

package.json: The [package.json is an npm configuration file](#), that lists the third-party packages that your project depends on.

tsconfig.json, tsconfig.app.json & tsconfig.spec.json are Typescript configuration files.

- [tsconfig.json](#) is the Typescript compiler configuration file. This file specifies the compiler options required for the Typescript to compile (transpile) the project.
- **tsconfig.app.json** is used for compiling the code
- **tsconfig.spec.json** for compiling the tests

Let's expand src



Part 3: the app

Part 2: deployment properties

Part 1: boot Section

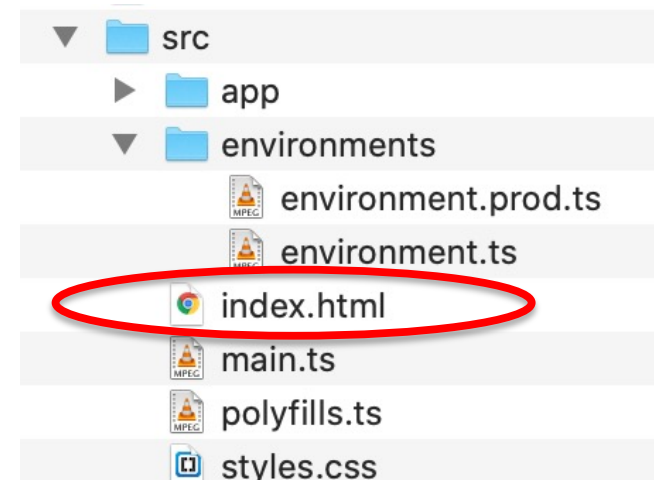
Part 1: boot Section

index.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>NgmoduleDemo</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root>Loading...</app-root>
</body>
</html>
```

There are neither javascript files nor css files in the index.html.

The body of the files has a special HTML tag.



Part 1: boot Section

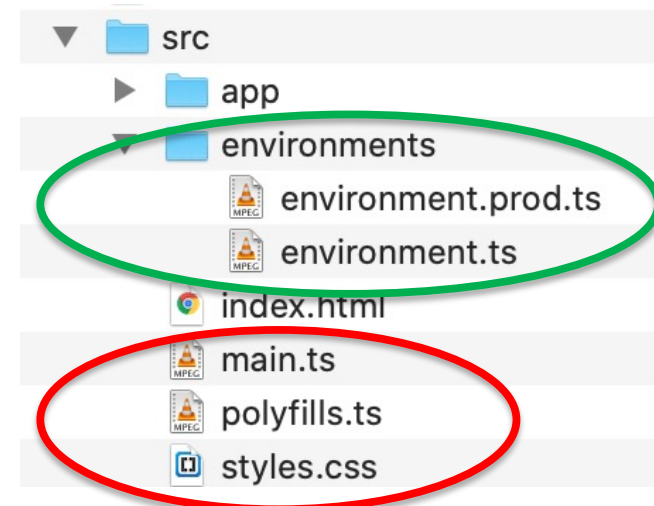
main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from '../app/app.module';
import { environment } from '../environments/environment';
if (environment.production) {
  enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule);
```

polyfill.ts: *obvious*
style.css: *obvious*

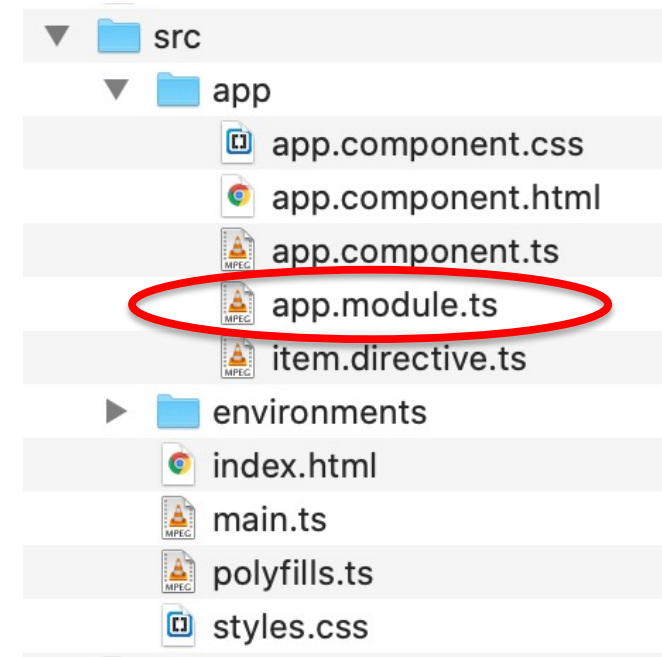
Part 2: deployment properties

environment.ts can be replaced during build by using the `fileReplacements` array.
`**ng build --prod**` replaces `environment.ts` with `environment.prod.ts`.
The production build optimizes, minimizes and uglifies the code.



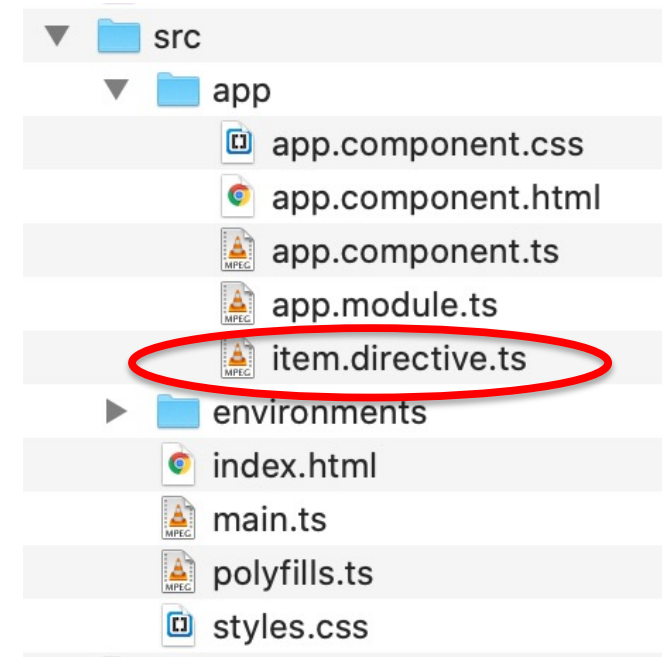
app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { AppComponent } from './app.component';
import { ItemDirective } from './item.directive';
// @NgModule decorator with its metadata
@NgModule({
  declarations: [
    AppComponent,
    ItemDirective
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



item.directive.ts

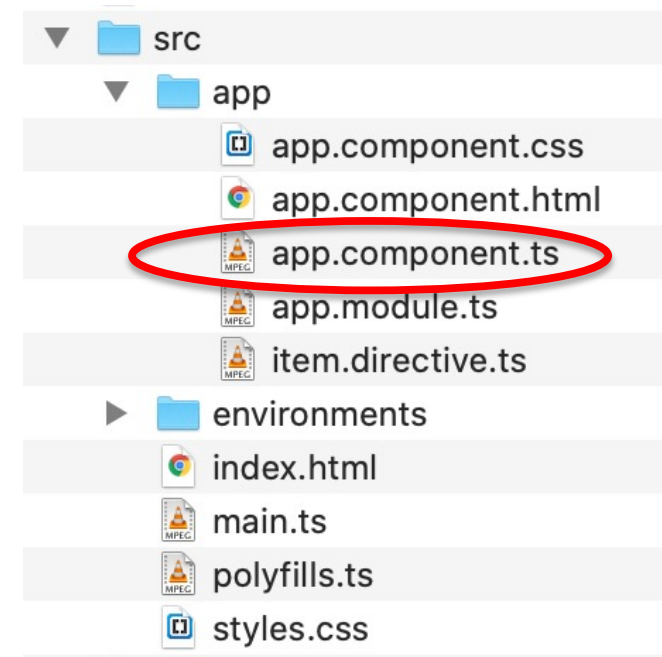
```
import { Directive } from '@angular/core';
@Directive({
  selector: '[appItem]'
})
export class ItemDirective {
  // code goes here
  constructor() { }
}
```



Part 3: the app

app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app works!';
}
```



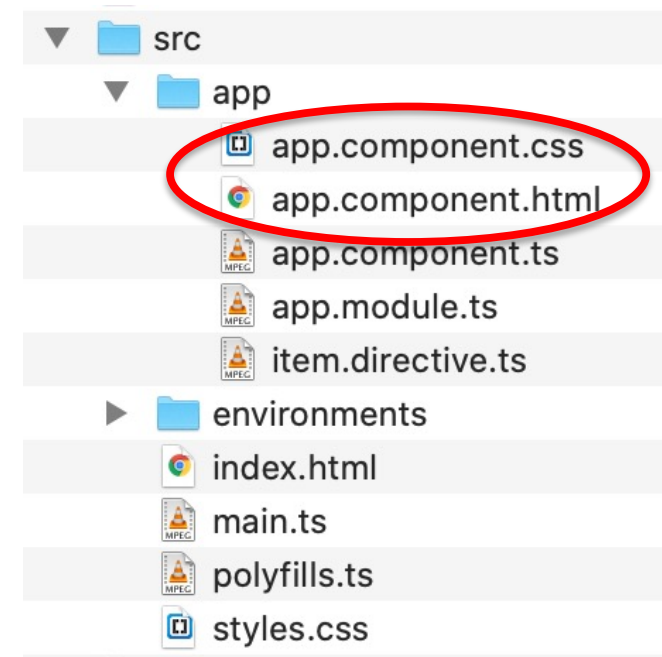
Part 3: the app

app.component.html

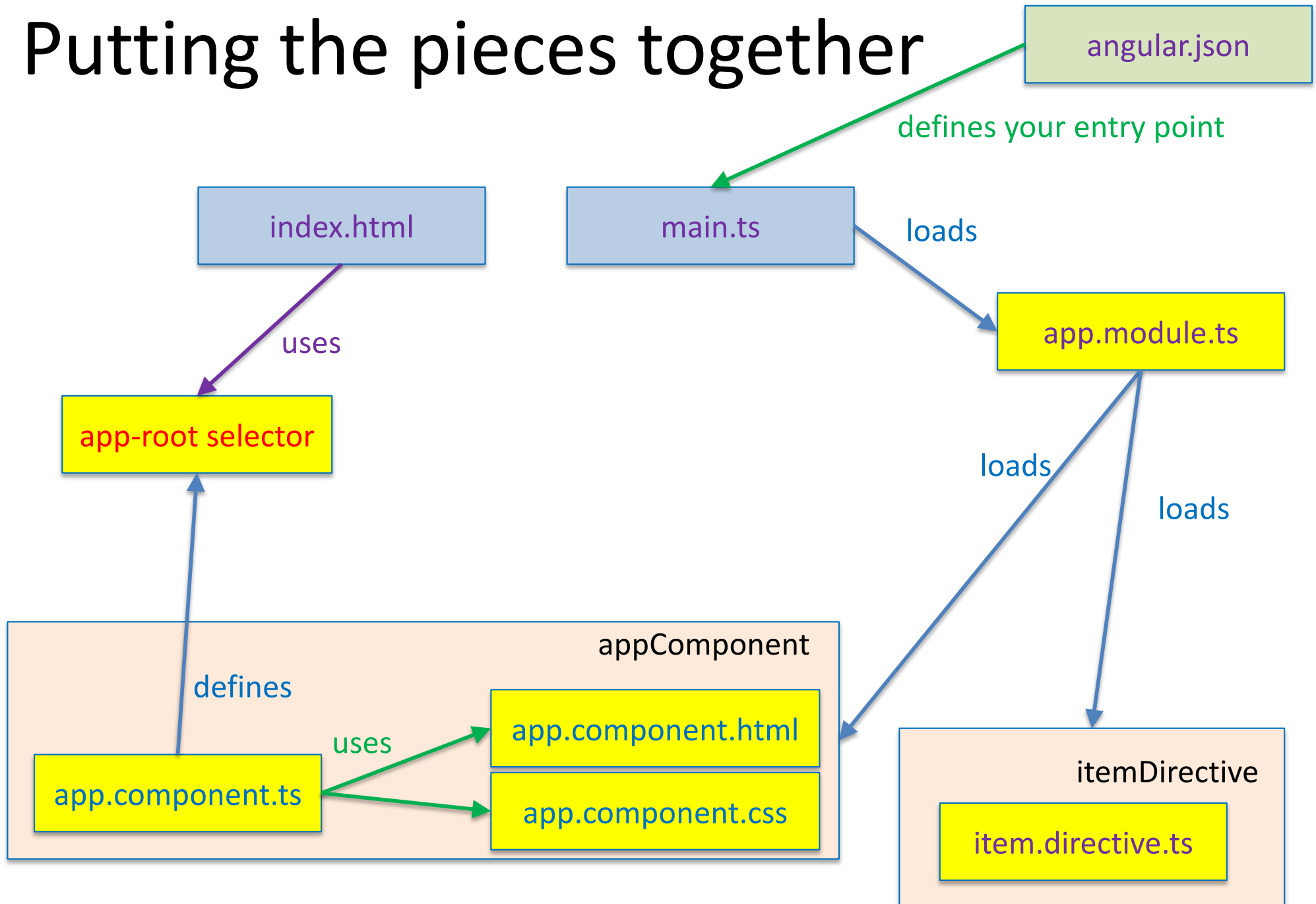
```
<h1>
  {{title}}
</h1>
```

app.component.css

empty



Putting the pieces together



Q

How do I customize my first app?

Building and running the project

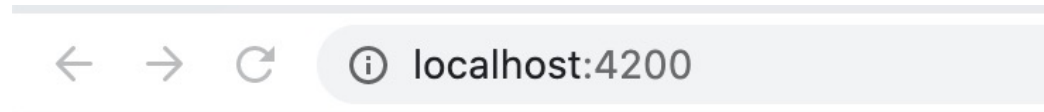
```
mkdir simpleProject.
```

```
cd simpleProject
```

```
// replace the src directory with the src directory available from the web site of the course
```

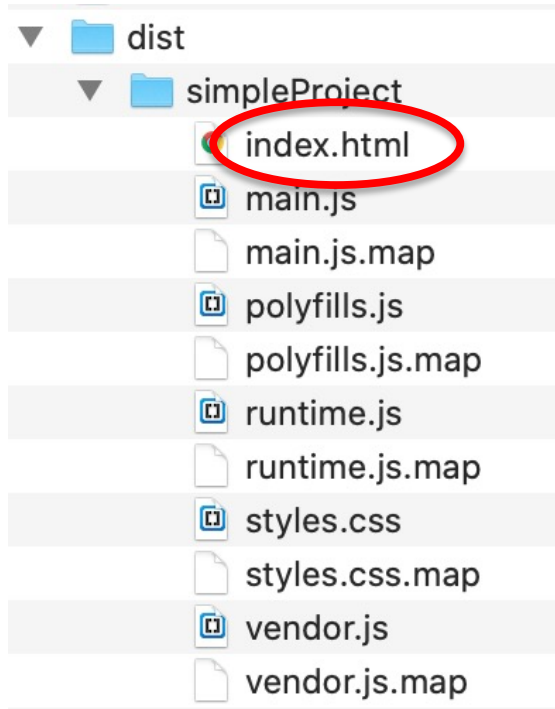
```
ng build
```

```
ng serve --open
```



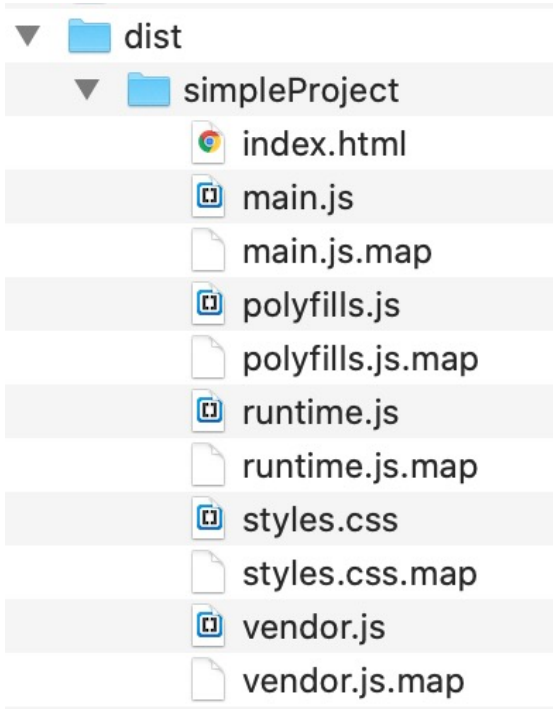
app works!

The distribution



```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>NgmoduleDemo</title>
  <base href="/">
  <meta name="viewport" content="width=device-width,
    initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <app-root>Loading...</app-root>
  <script src="runtime.js" defer></script>
  <script src="polyfills.js" defer></script>
  <script src="vendor.js" defer></script>
  <script src="main.js" defer></script></body>
</html>
```

The distribution



`runtime.js`: Webpack runtime file

`polyfills.js` – Polyfill scripts for supporting the variety of the latest modern browsers

`styles.js` – This file contains the global style rules bundled as javascript file.

`vendor.js` – contains the scripts from the Angular core library and any other 3rd party library.

`main.js` – code of the application.

webpack is a module bundler. it scans our application looking for javascript files and merges them into one (or more) big file. Webpack has the ability to bundle any kind of file like JavaScript, CSS, SASS, LESS, images, HTML, & fonts, etc.

<https://webpack.js.org/>

if you want to know more...

The inner working of Angular is well described in

<https://www.tektutorialshub.com/angular/angular-bootstrapping-application/>

Q

How can I do quick practice?

Quick development and testing

You can use **stackblitz**:

<https://stackblitz.com/angular/glkadjjjbdq?file=src%2Fapp%2Fapp.component.html>

The screenshot displays the StackBlitz web IDE interface. The top navigation bar includes a lightning bolt icon, the text "Angular Project", and buttons for "Fork" and "Share". The right side of the top bar shows the username "ronchet", a "Live" indicator, and a "Close" button. The left sidebar contains a "PROJECT" section with "INFO" (Angular Example - Bootstrapping) and "FILES" (a file tree showing src, app, environments, and various files like app.component.html, app.module.ts, etc.). The main editor area shows the code for "app.component.html" with the following content:

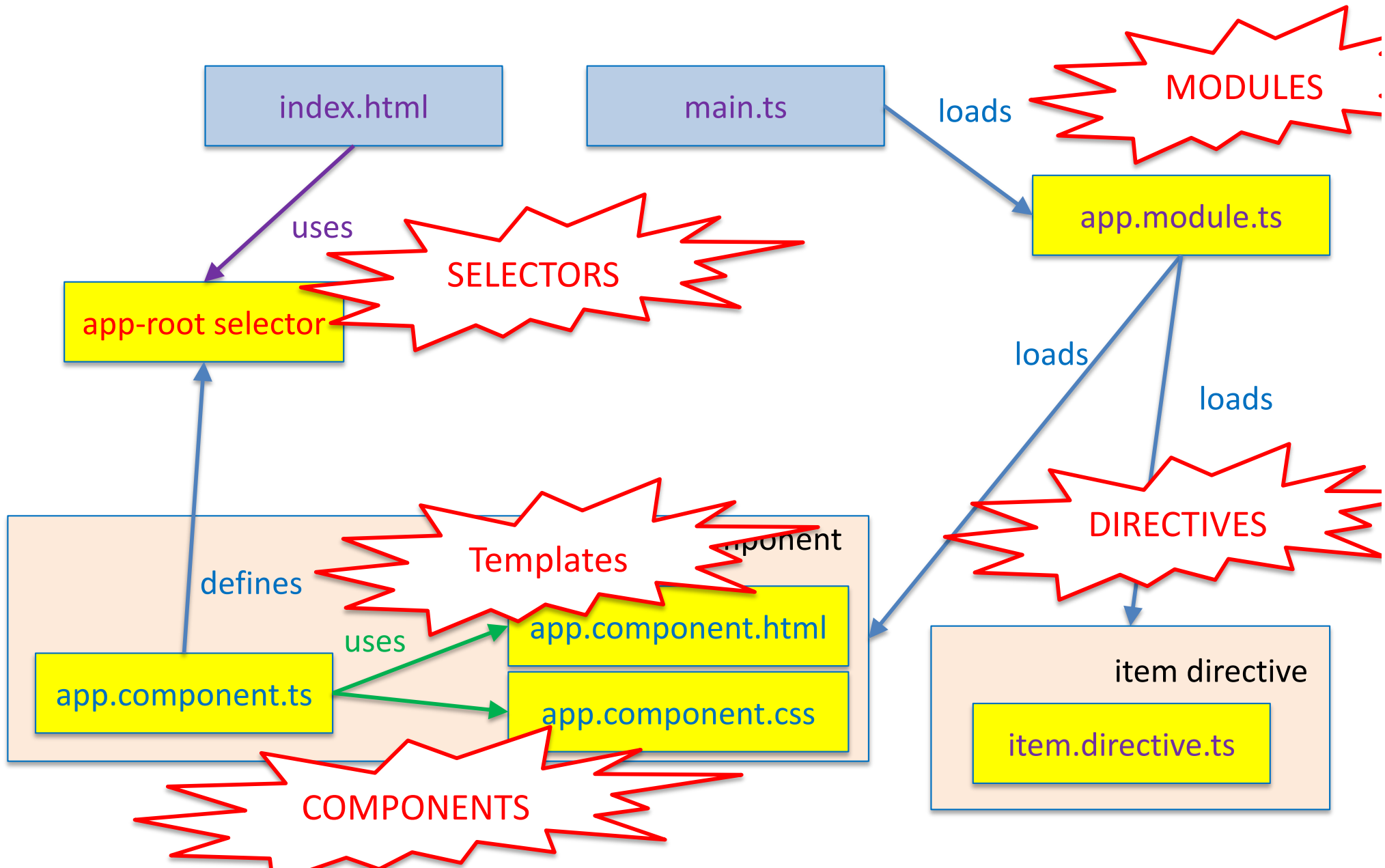
```
1 <h1>
2   {{title}}
3 </h1>
4 <input (input)="handleInput($event)">
5 <p>You have entered {{value}}</p>
6
7
8 <!--
9 Copyright Google LLC. All Rights Reserved.
10 Use of this source code is governed by an MIT-style
11 license that
12 can be found in the LICENSE file at
13 https://angular.io/license
14 -->
```

The right pane shows a live preview of the application. It features the text "app works!" in a large blue font. Below it is an input field containing the text "hel". Underneath the input field, the text "You have entered hel" is displayed. At the bottom of the interface is a "Console" panel with a single log entry.

Q

Which are the architectural elements?

Putting the pieces together



Components and templates

A component controls a patch of screen. It contains

- the data
- user interaction logic that defines how the View looks and behaves.

A Component consists of three main building blocks

- Template
- Class
- MetaData

The component has a representation (view), which is defined by a **template**. A template is portion of HTML that tells Angular how to render the component.

a component:

needs an import

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

needs a decorator, with binding to a template

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

defines a class, with properties and behaviours

```
export class AppComponent {  
  aProperty = 'app works!';  
  doSomething();  
}
```

ng secondProject

```
CREATE secondProject/README.md
CREATE secondProject/.editorconfig
CREATE secondProject/.gitignore
CREATE secondProject/angular.json
CREATE secondProject/package.json
CREATE secondProject/tsconfig.json
CREATE secondProject/tslint.json
CREATE secondProject/.browserslistrc
CREATE secondProject/karma.conf.js
CREATE secondProject/tsconfig.app.json
CREATE secondProject/tsconfig.spec.json
```

By Convention, the file name starts with the **feature name** (app) and then followed by the **type of class** (component), separated by a dot. The extension used is **ts** indicating that this is a typescript module file.

```
CREATE secondProject/src/favicon.ico (948 bytes)
CREATE secondProject/src/index.html (299 bytes)
CREATE secondProject/src/main.ts (372 bytes)
CREATE secondProject/src/polyfills.ts (2826 bytes)
CREATE secondProject/src/styles.css (80 bytes)
CREATE secondProject/src/test.ts (753 bytes)
CREATE secondProject/src/assets/.gitkeep (0 bytes)
CREATE secondProject/src/environments/environment.prod.ts
CREATE secondProject/src/environments/environment.ts
CREATE secondProject/src/app/app-routing.module.ts
CREATE secondProject/src/app/app.module.ts
CREATE secondProject/src/app/app.component.css
CREATE secondProject/src/app/app.component.html
CREATE secondProject/src/app/app.component.spec.ts
CREATE secondProject/src/app/app.component.ts
CREATE secondProject/e2e/protractor.conf.js
CREATE secondProject/e2e/tsconfig.json
CREATE secondProject/e2e/src/app.e2e-spec.ts (664 bytes)
CREATE secondProject/e2e/src/app.po.ts (274 bytes)
```


Q

Where is a component loaded?

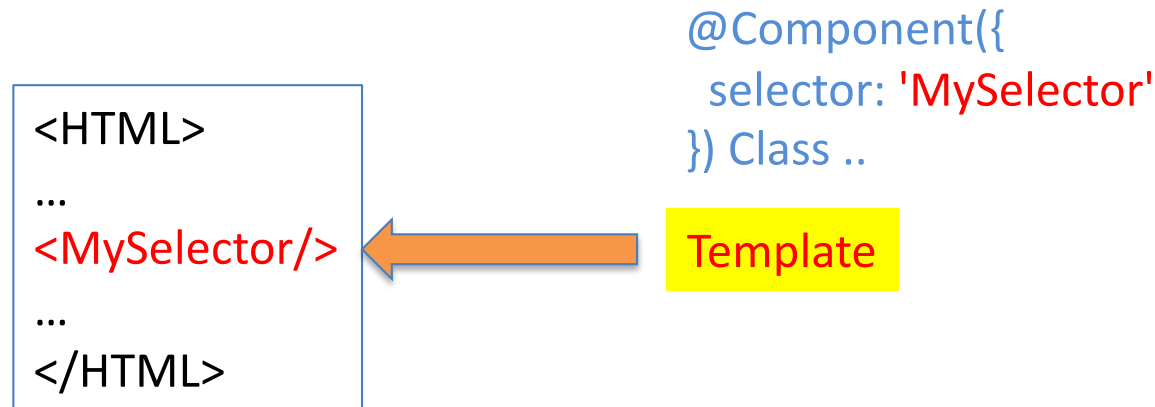
a component defines a selector:

When we build Angular Components, we are actually building new HTML elements.

We specify the name of the HTML element in the selector property of the component metadata.

Then we use it in our HTML: Angular,

- searches for the selector in the HTML file
- instantiates the component
- renders the Template associated with the component inside the selector.



Other types of associations for a selector:

Selectors can be used for:

- CSS Classes

`<div class="app-root"></div>`

```
@Component({  
  selector: '.app-root'  
}) Class ..
```

Template

note the dot

- Attribute names

`<div app-root></div>`

```
@Component({  
  selector: '[app-root]'  
}) Class ..
```

Template

- Attribute name and value

`<div app="components"></div>`

```
@Component({  
  selector: 'div[app=components]'  
}) Class ..
```

Template