Maximilian 11/Jan/21

Angular is JS Framework, which allows to create reactive SPA’s

Angular is written in Typescript, uses MVC methodology.

AngulaJS (Angular 1). Angular is from Google

|  |  |  |
| --- | --- | --- |
| ^11.0.0 | Active | Nov 11, 2020 |

Npm install –g @angular/cli@latest

Ng new my\_dream\_app

Cd my\_dream\_app

Ng serve

--

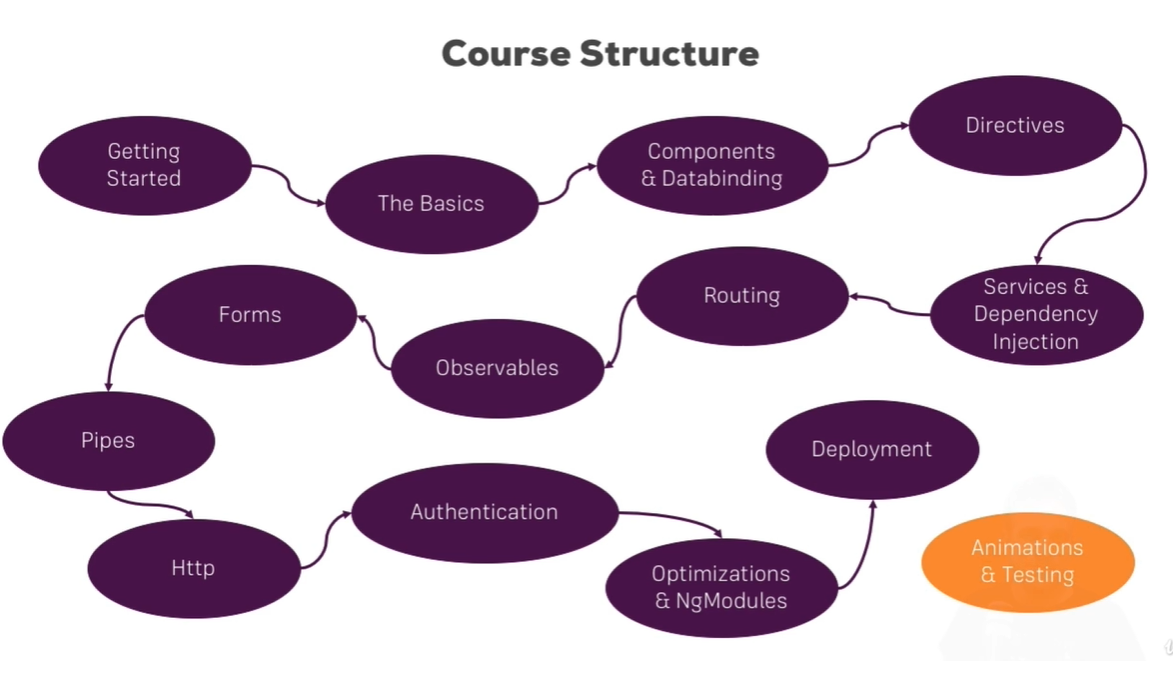
Inorder to create an app with ng, removed the npm, npm cache from appdata/roaming folder

Server runs on localhost:4200

WebStorm is an IDE for Angular development.

Angular has no DOM feature. Instead uses MVC

Pipes determine how the data is displayed in the html template.



Typescript is strongly typed. Typescript is compiled into js by cli.

Npm install –save bootstrap@3

Angular start: main.ts -> app.module.ts -> app.component.ts -> index.html (app.component.html)

Creating components using CLI: ng generate component <component-name>

Data binding in Angular: ->, <-, -> <-

Uses string interpolation {{}}, property binding [property] = value, two way binding [(ngModel)] = name directive

The ‘event’ should be given within parenthesis inside the element. ‘on’ is omitted. The method call should be within quotation marks.

$event inside the method parenthesis to get the event value.

Directives are instructions in the DOM

Inside a element property, \*ngIf = “property” either return true or false to determine the presence or absence of the element.

<ng-template #marker> -- use the marker with ‘else’

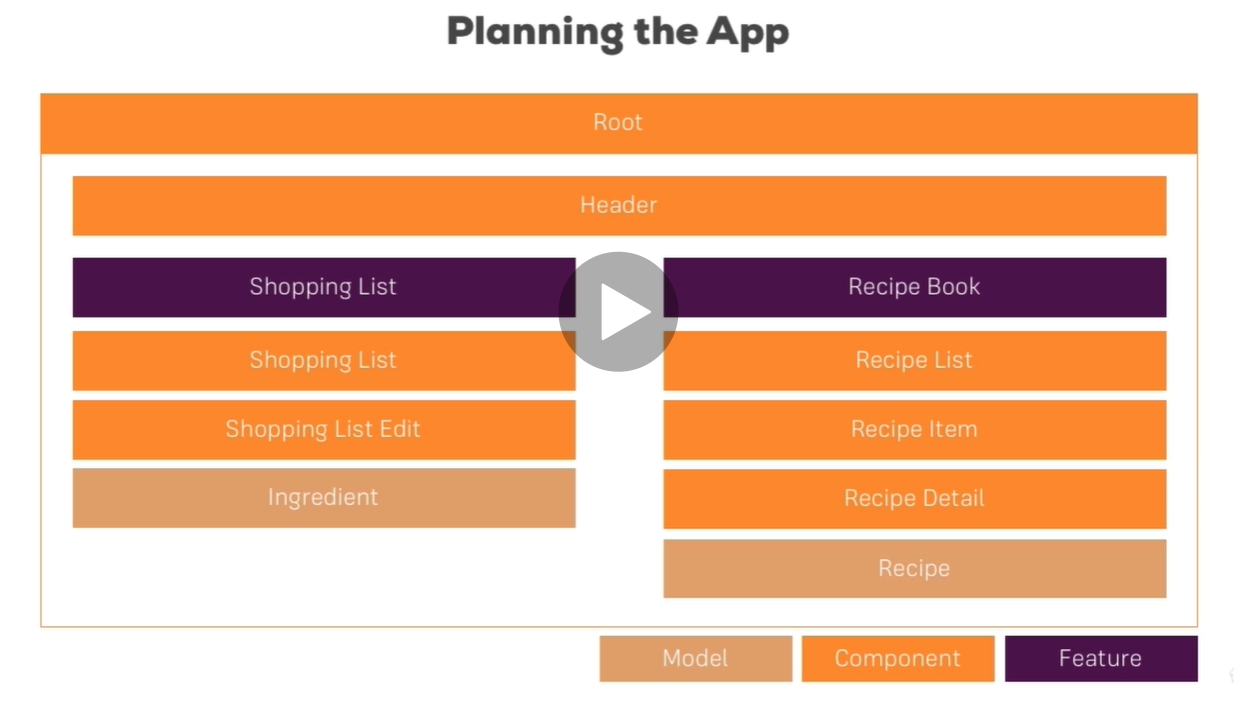
Styling elements dynamically with ‘ngStyle’

Unlike structural directives don’t add or remove elements. They only change the elements they were placed on.

Applying css classes dynamically with ngClass

Outputting lists with ngFor

Property directives are put inside square bracket. Structural directives are note.



Ng g c <component name> --skipTests true ( to skip generating testing files)

Bootstrap:

In head: <link href = "css/bootstrap.min.css" rel = "stylesheet">

In body: <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->

<script src = "https://code.jquery.com/jquery.js"></script>

<!-- Include all compiled plugins (below), or include individual files as needed -->

<script src = "js/bootstrap.min.js"></script>

/\* Small devices (tablets, 768px and up) \*/

@media (min-width: @screen-sm-min) { ... }

/\* Medium devices (desktops, 992px and up) \*/

@media (min-width: @screen-md-min) { ... }

/\* Large devices (large desktops, 1200px and up) \*/

@media (min-width: @screen-lg-min) { ... }

Occasionally these are expanded to include a **max-width** to limit CSS to a narrower set of devices.

@media (max-width: @screen-xs-max) { ... }

@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }

@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }

@media (min-width: @screen-lg-min) { ... }

<meta name = "viewport" content = "width = device-width, initial-scale = 1.0, maximum-scale = 1.0, user-scalable = no">

Responsive Images

Bootstrap 3 allows you to make the images responsive by adding a class **.img-responsive** to the **<img>** tag. This class applies **max-width: 100%;** and **height: auto;** to the image so that it scales nicely to the parent element.

<img src = "..." class = "img-responsive" alt = "Responsive image">

Use class **.container** to wrap a page's content and easily center the content's as shown below.

To add some emphasis to a paragraph, add class = "lead".

<abbr title = "World Wide Web">WWW</abbr><br>

<abbr title = "Real Simple Syndication" class = "initialism">RSS</abbr>

Form class = role

Div class = form-group // input class= form-control

|  |  |
| --- | --- |
| 1 | **btn**  Default/ Standard button. |
| 2 | **btn-primary**  Provides extra visual weight and identifies the primary action in a set of buttons. |
| 3 | **btn-success**  Indicates a successful or positive action. |
| 4 | **btn-info**  Contextual button for informational alert messages. |
| 5 | **btn-warning**  Indicates caution should be taken with this action. |
| 6 | **btn-danger**  Indicates a dangerous or potentially negative action. |
| 7 | **btn-link**  Deemphasize a button by making it look like a link while maintaining button behavior. |

[Live Demo](http://tpcg.io/PBs3Bn)

<img src = "/bootstrap/images/download.png" class = "img-rounded">

<img src = "/bootstrap/images/download.png" class = "img-circle">

<img src = "/bootstrap/images/download.png" class = "img-thumbnail">

## Showing and Hiding Content

You can force an element to be shown or hidden (including for screen readers) with the use of classes **.show** and **.hidden**.

You can hide an element to all devices except screen readers with the class **.sr-only**.

Glyphicons are icon fonts

<div class = "btn-group">

<button type = "button" class = "btn btn-default">Button 1</button>

<button type = "button" class = "btn btn-default">Button 2</button>

<div class = "btn-group">

<button type = "button" class = "btn btn-default dropdown-toggle" data-toggle = "dropdown">

Dropdown

<span class = "caret"></span>

</button>

<ul class = "dropdown-menu">

<li><a href = "#">Dropdown link 1</a></li>

<li><a href = "#">Dropdown link 2</a></li>

</ul>

</div>

</div>

<p>Tabs Example</p>

<ul class = "nav nav-tabs"> // nav-pills nav-stacked jav-justified

<li class = "active"><a href = "#">Home</a></li>

<li><a href = "#">SVN</a></li>

<li><a href = "#">iOS</a></li>

<li><a href = "#">VB.Net</a></li>

<li><a href = "#">Java</a></li>

<li><a href = "#">PHP</a></li>

</ul>

<p>Tabs With Dropdown Example</p>

<ul class = "nav nav-tabs">

<li class = "active"><a href = "#">Home</a></li>

<li><a href = "#">SVN</a></li>

<li><a href = "#">iOS</a></li>

<li><a href = "#">VB.Net</a></li>

<li class = "dropdown">

<a class = "dropdown-toggle" data-toggle = "dropdown" href = "#">

Java

<span class = "caret"></span>

</a>

<ul class = "dropdown-menu">

<li><a href = "#">Swing</a></li>

<li><a href = "#">jMeter</a></li>

<li><a href = "#">EJB</a></li>

<li class = "divider"></li>

<li><a href = "#">Separated link</a></li>

</ul>

</li>

<li><a href = "#">PHP</a></li>

</ul>

Breadcrumbs are a great way to show hierarchy-based information for a site.

[Live Demo](http://tpcg.io/FfiuMY)

<ul class = "pagination">

<li><a href = "#">&laquo;</a></li>

<li><a href = "#">1</a></li>

<li><a href = "#">2</a></li>

<li><a href = "#">3</a></li>

<li><a href = "#">4</a></li>

<li><a href = "#">5</a></li>

<li><a href = "#">&raquo;</a></li>

</ul>

[Live Demo](http://tpcg.io/Zj5RVx)

<ul class = "pager">

<li><a href = "#">Previous</a></li>

<li><a href = "#">Next</a></li>

</ul>

[Live Demo](http://tpcg.io/yzgQU0)

<ul class = "pager">

<li class = "previous"><a href = "#">&larr; Older</a></li>

<li class = "next"><a href = "#">Newer &rarr;</a></li>

</ul>

<ul class = "pager">

<li class = "previous disabled"><a href = "#">&larr; Older</a></li>

<li class = "next"><a href = "#">Newer &rarr;</a></li>

</ul>

[Live Demo](http://tpcg.io/G30fmH)

<span class = "label label-default">Default Label</span>

<span class = "label label-primary">Primary Label</span>

<span class = "label label-success">Success Label</span>

<span class = "label label-info">Info Label</span>

<span class = "label label-warning">Warning Label</span>

<span class = "label label-danger">Danger Label</span>

<a href = "#">Mailbox <span class = "badge">50</span></a>

Jumbotron: this component can optionally increase the size of headings and add a lot of margin for landing page content.

<div class = "jumbotron">

<div class = "container">

<h1>Welcome to landing page!</h1>

<p>This is an example for jumbotron.</p>

<p>

<a class = "btn btn-primary btn-lg" role = "button">Learn more</a>

</p>

</div>

</div>

<div class = "row">

<div class = "col-sm-6 col-md-3">

<a href = "#" class = "thumbnail">

<img src = "/bootstrap/images/kittens.jpg" alt = "Generic placeholder thumbnail">

</a>

</div>

</div>

<div class = "alert alert-success">Success! Well done its submitted.</div>

<div class = "alert alert-info">Info! take this info.</div>

<div class = "alert alert-warning">Warning ! Dont submit this.</div>

<div class = "alert alert-danger">Error ! Change few things.</div>

<div class = "progress">

<div class = "progress-bar" role = "progressbar" aria-valuenow = "60"

aria-valuemin = "0" aria-valuemax = "100" style = "width: 40%;">

<span class = "sr-only">40% Complete</span>

</div>

</div>

Panel components are used when you want to put your DOM component in a box.

<div class = "panel panel-default">

<div class = "panel-body">

This is a Basic panel

</div>

<div class = "panel-footer">Panel footer</div>

</div

------

Browser runs main.bundle.js. Source map provides for debugging.

Augury is a tool (chrome extension) to analyse the Angular application.

By default, all properties of a component are accessible only from inside of that component.

To make a property expose to the world, from a component, use @Input() <property name>

<property> = new EventEmitter <{emitted\_variable:type,}>()

@Output Eventproperty – to send data out of the component.

The property alias can only be used to assign the value. To access the value, property name has to be used.

---

Input: from top component to sub component

one.component.ts

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

@Component({

selector: 'app-one',

templateUrl: './one.component.html',

styleUrls: ['./one.component.css']

})

export class OneComponent implements OnInit {

@Input('lcl') lokal: {type: string, name: string }

constructor() { }

ngOnInit(): void {

}

}

---

one.component.html

<p>{{lokal.type}}</p>

---

app.component.html

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

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'sammys';

elements = [{type:"server", name:"apple"}]

}

----

app.component.html

<app-one \*ngFor="let element of elements"[lcl]="element" ></app-one>

Emitter (outputs value from the sub component to the top component)

Sub component: one.component.ts

import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';

@Component({

selector: 'app-one',

templateUrl: './one.component.html',

styleUrls: ['./one.component.css']

})

export class OneComponent implements OnInit {

@Input('lcl') lokal: {type: string, name: string };

@Output() jjServer= new EventEmitter<{age: number}>() // output

addServer(){ // emitter

this.jjServer.emit({

age: 47

})

}

constructor() { }

ngOnInit(): void {

}

}

--

One.component.html

<button (click)="addServer()"></button>

---

app.component.ts

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

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'sammys';

elements = [{type:"server", name:"apple"}]

myage = 'not set'

myServer(data:{age:number}){

this.myage = data.age

}

}

----

app.component.html

<app-one \*ngFor="let element of elements"[lcl]="element"

(jjServer)="myServer($event)" ></app-one>

<h2>My age is: {{myage}}</h2>

In Angular, the html element styles will be applied only to the element it defines. Not global

Declaring encapsulation:Viewencapsulation.None – will disable the Angular specific restrictions on global style property – allows it to apply globally – emulated is the default. ShadowDom is not supported by all the browsers.

Local reference can be given inside an element, using #name.

Access this by elementref.NativeElement.value

component.html

<button (click)="showVal(inputref)">showval</button>

<input type="text" placeholder="name" #inputref/>

<p>{{inval}}</p>

component.ts

inval = null

showVal(inref: HTMLInputElement){

this.inval = inref.value

}

-----

Element ref can be created by using a #name in the html element and @ViewChild(‘name’) elementref: ElementRef;

component.ts

@ViewChild('inputref',{static:true}) inputref: ElementRef

.showVal(inref){

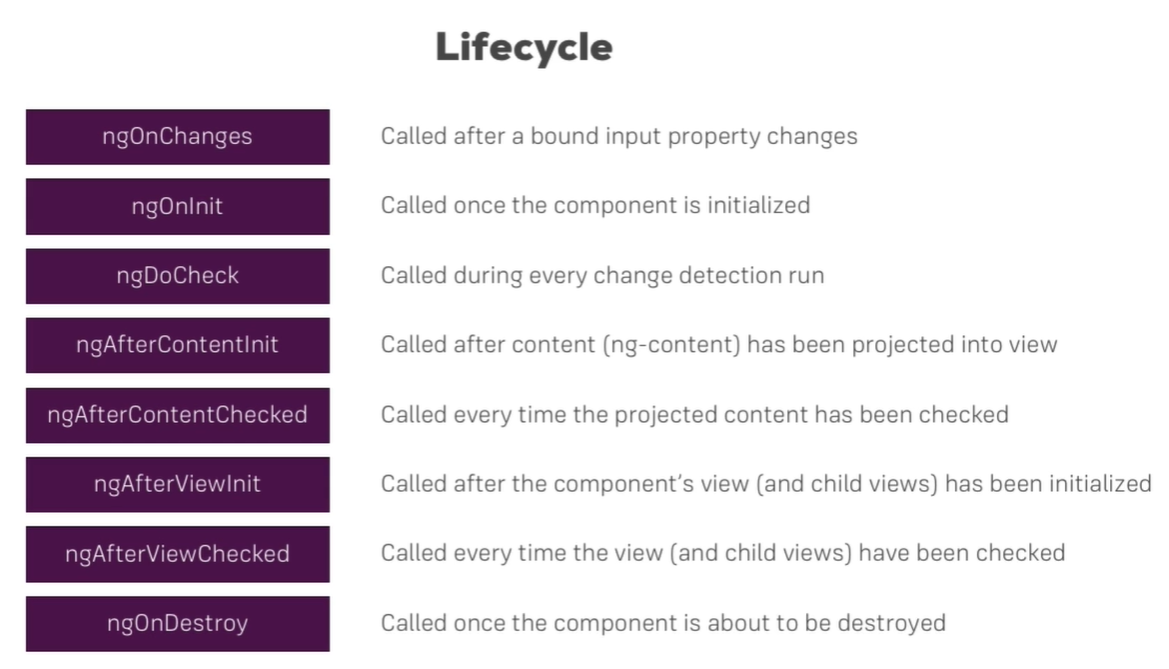
this.inval = inref.value

this.elref = this.inputref.nativeElement.value

}

-------------

<ng-content> - will display whatever placed inside the opening and closing of a component. – like props.child in React



ngOnChanges(changes:SimpleChanges) – receives an argument

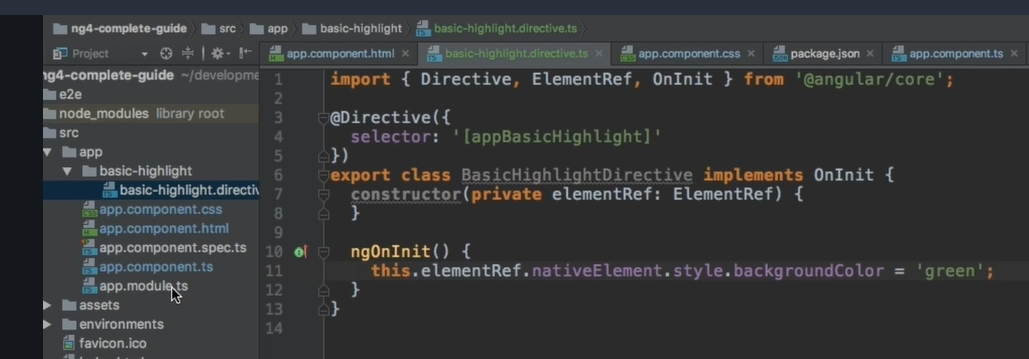
@ContentChild with reference to ngContent element, will give access to the content.

Attribute directives and structural directives.

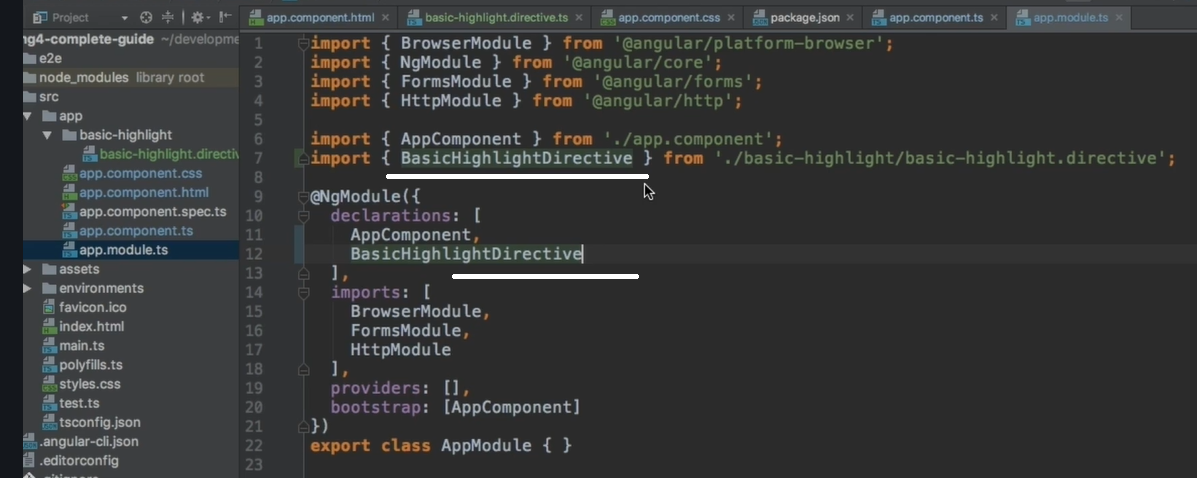
Structural directives \*ngIf and \*ngFor can’t be used on the same element at a time.

Custom directive:

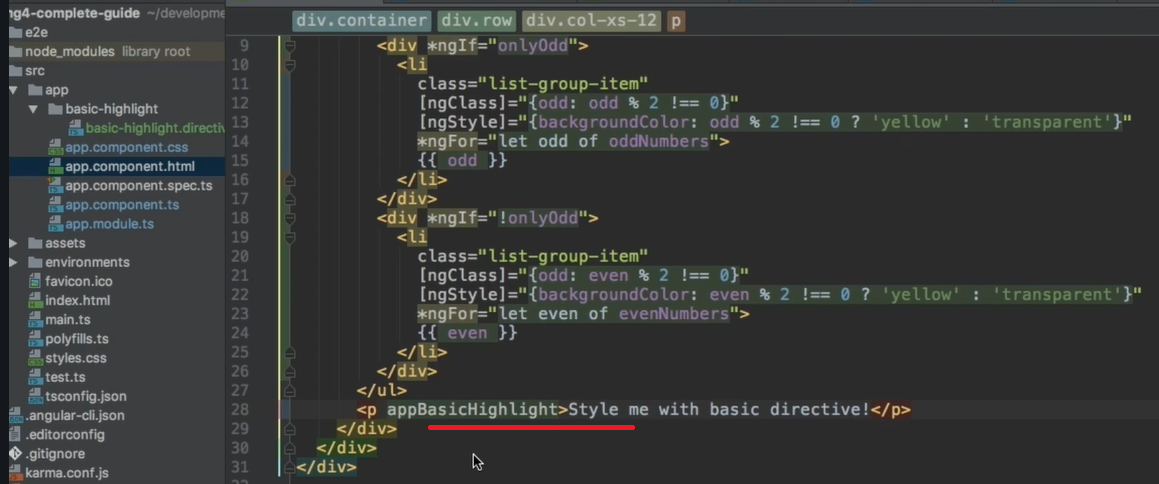
Component.html



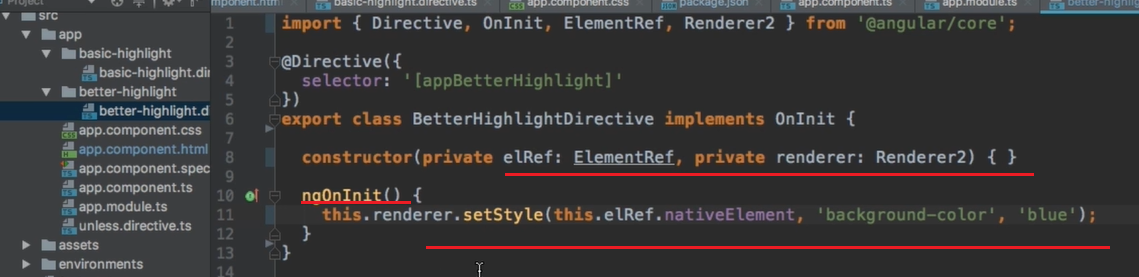
App.module



Component.html

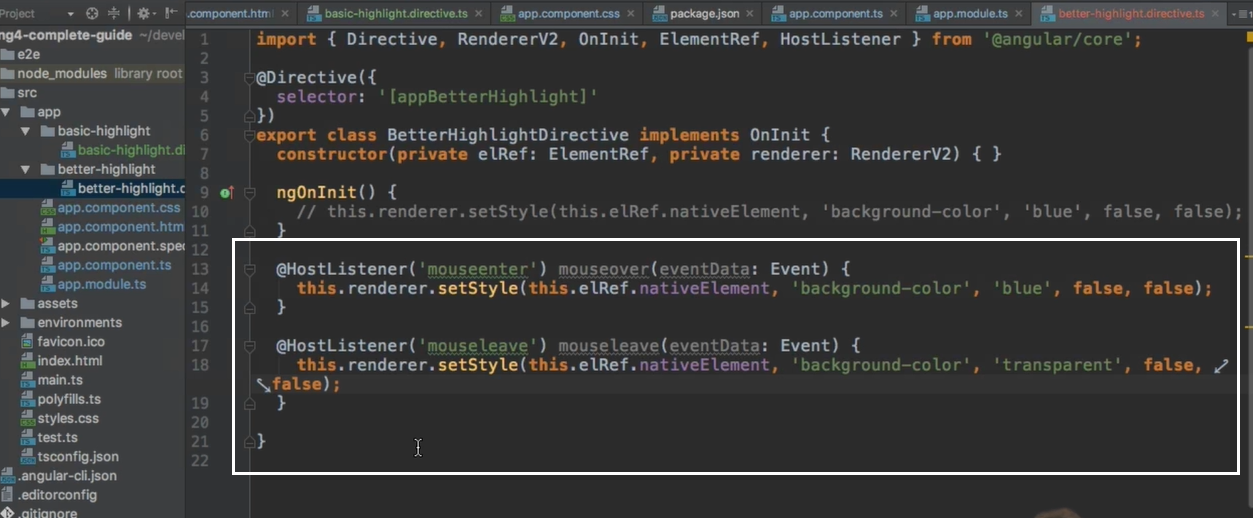


Renderer2:

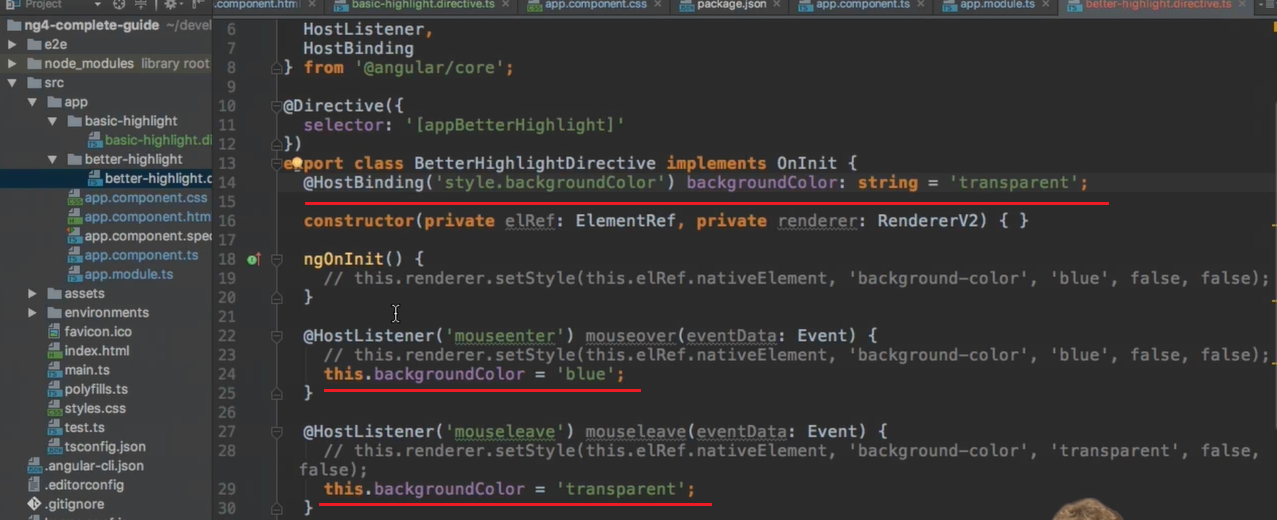


To create a directive in CLI, use ng c d ----- d for directive

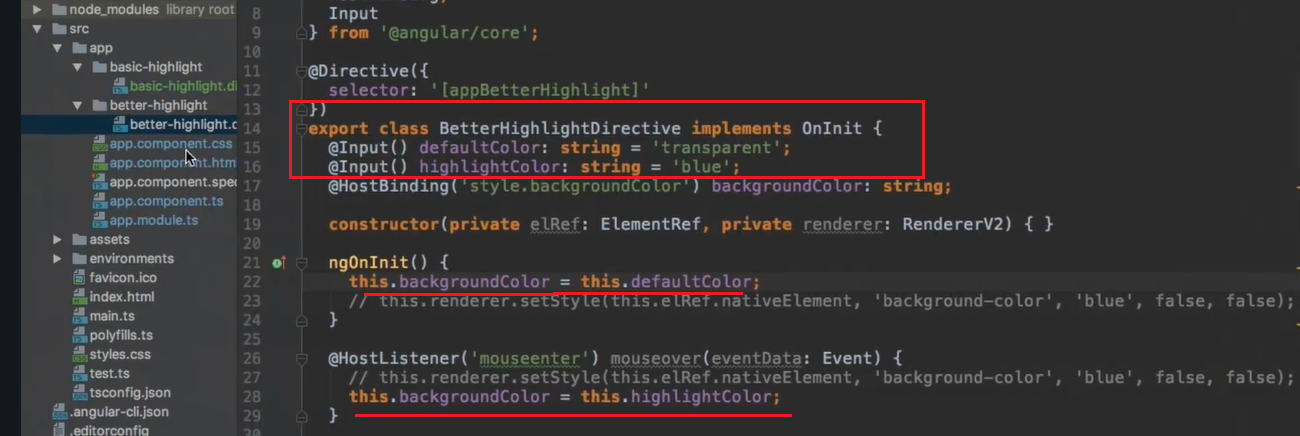
Host Listener for events:

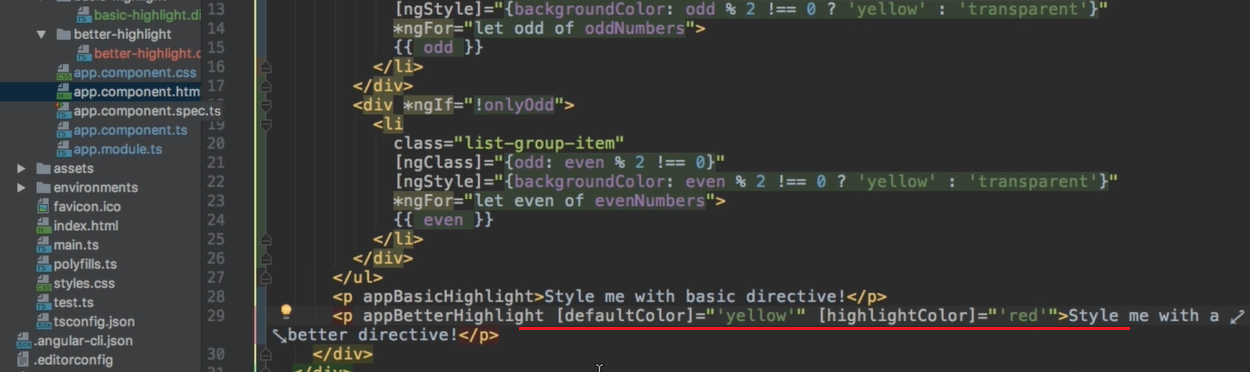


Host binding property:



Binding to directive properties:



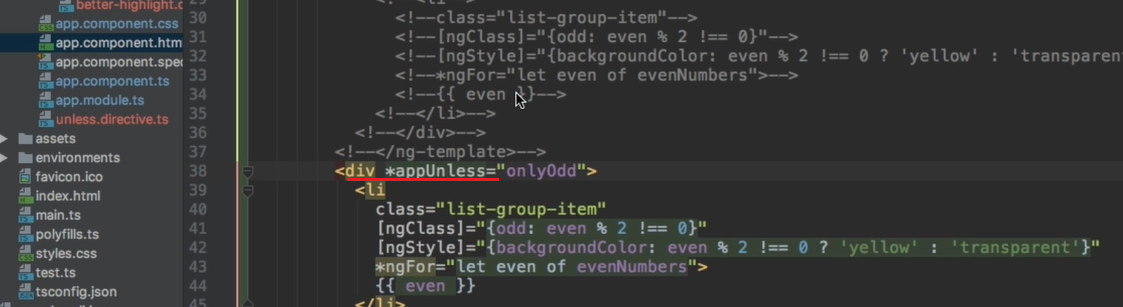


There are only property binding, event binding, 2way binding and string interpolation

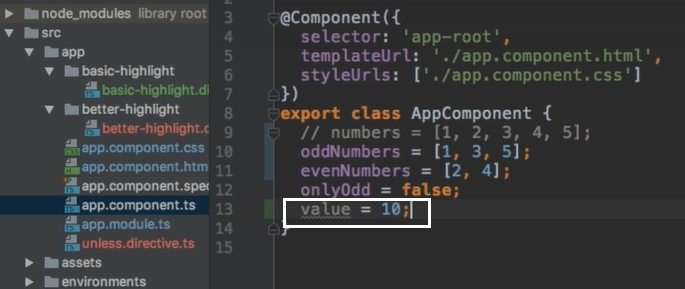
<ngTemplate> can be used to provide [ngIf] ------ without \*

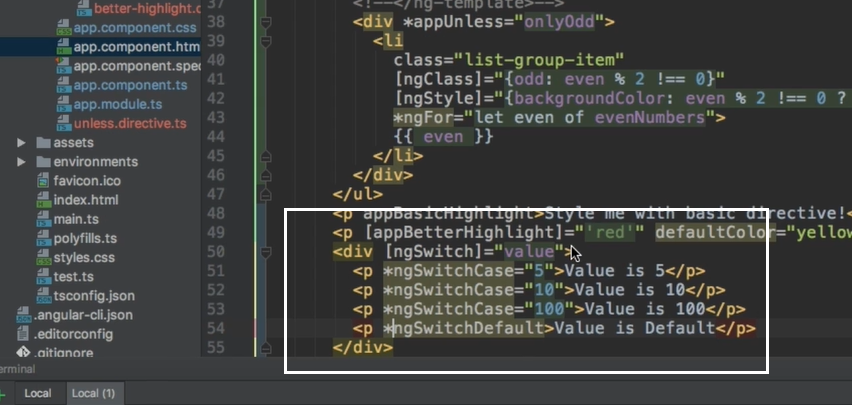
Custom directive:





ngSwitch:





Services and dependency injection

\*star