Angular Fundamentals

Chapter 2: Getting started with Angular

What is Angular and why use it:

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

How to open a new project;

ng new project_name

How to compile and run project*:*

npm start (it will run “ng start” on its own)

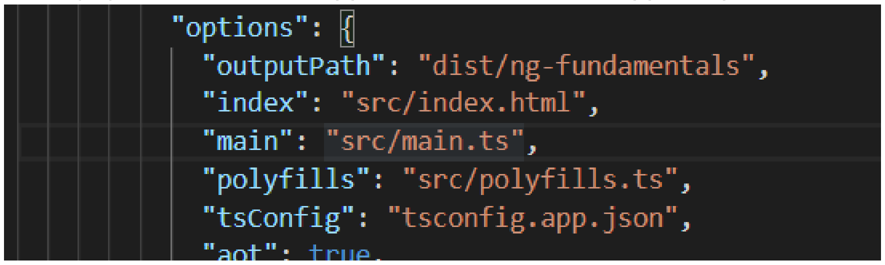
Note: I did not enable angular routing at time of creation as instructed in course

How bootstrap is connected to the app:a

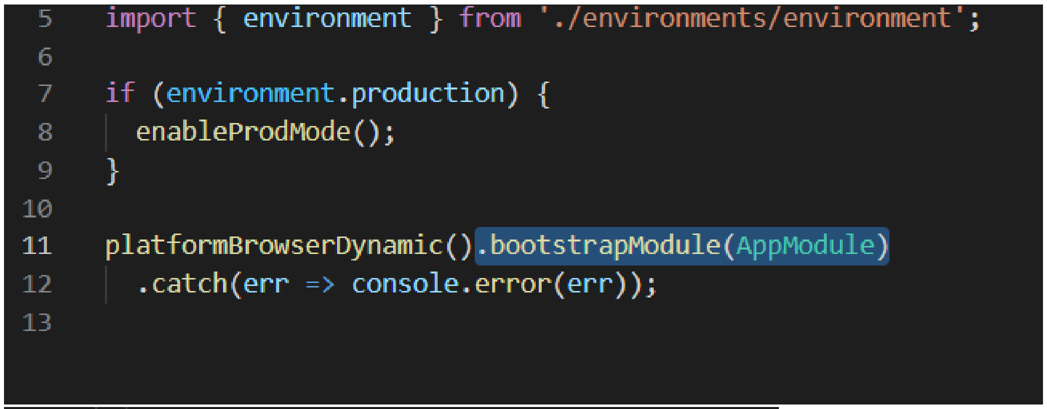
The Bootstrapping of our app begins with this main.ts file. And if we look over here in our angular.json file, you can see that this main property is pointing to our main.ts file. This is used by the webpack.config for our site, and it causes this main.ts file to be loaded when our app first loads. And you can see right here that we are Bootstrapping our app with our AppModule. if you remember from earlier, Angular applications are grouped into models, and every Angular application has an AppModule. That module is defined here, and you can see that the AppModule is Bootstrapped with the AppComponent. So that's how Angular knows about our AppComponent. But there's one more piece. This makes Angular aware of our component, but we haven't seen yet where we tell Angular to actually display this component. If we look at our AppComponent, you can see that it has this app-root selector. This selector defines the HTML tag to use in order to display this component. And if we look over here in this index.html file, you can see that we're using that app-root selector right here. This index.html file is what is first displayed when our app loads, and it is loading our AppComponent. So that's how this all comes together. Now to show that there's nothing magical about the naming of this AppComponent,

Images showing connection:

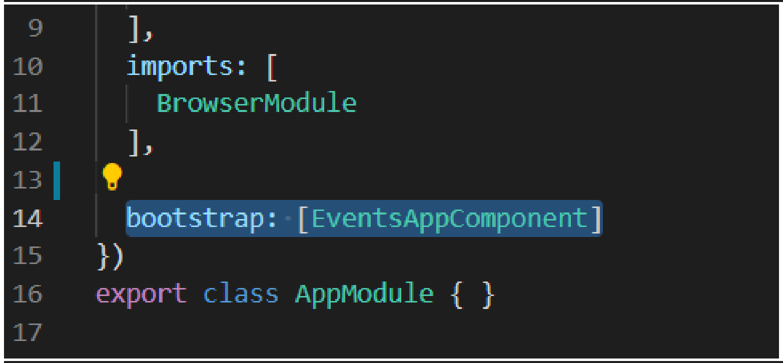
Package.json -> main.ts -> app.module.ts -> events-app.component.ts ->



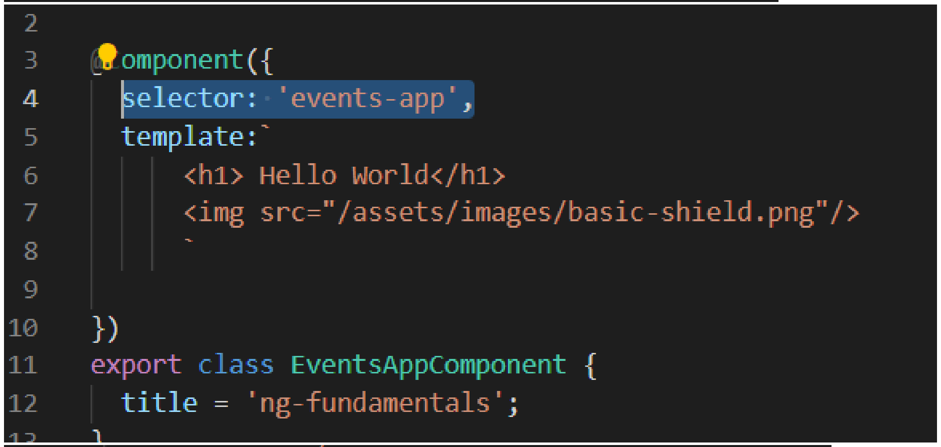
```
"options": {
  "outputPath": "dist/ng-fundamentals",
  "index": "src/index.html",
  "main": "src/main.ts",
  "polyfills": "src/polyfills.ts",
  "tsConfig": "tsconfig.app.json",
  "aot": true,
```



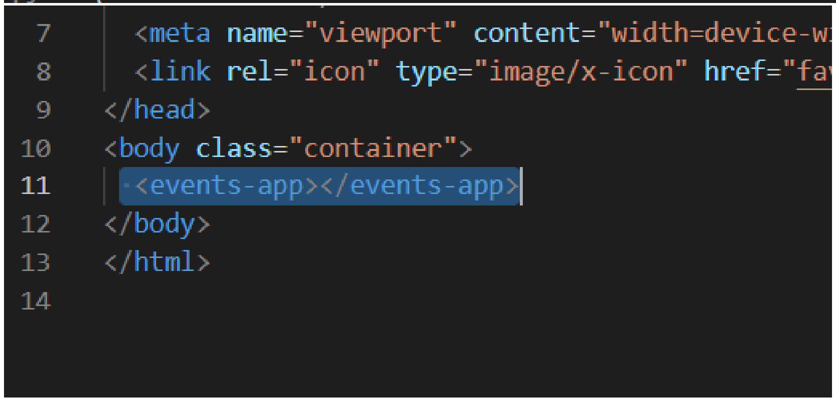
```
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));

```



```
  ],
  imports: [
    BrowserModule
  ],

  bootstrap: [EventsAppComponent]
})
export class AppModule { }

```



```

@Component({
  selector: 'events-app',
  template:`
    <h1> Hello World</h1>
    <img src="/assets/images/basic-shield.png"/>
    `

})
export class EventsAppComponent {
  title = 'ng-fundamentals';
}
```



```
    <meta name="viewport" content="width=device-wi
    <link rel="icon" type="image/x-icon" href="fa
  </head>
  <body class="container">
    <events-app></events-app>
  </body>
  </html>

```

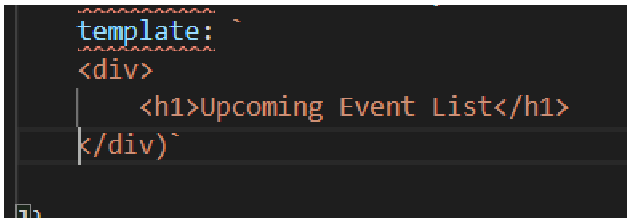
The images in order show how angular recognizes and uses Bootstrap for the application
What is Bootstrap and why is It used:

Bootstrap is a framework to help you design websites faster and easier. It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels, etc. It also gives you support for JavaScript plugins. ... Bootstrap's responsive CSS adjusts to phones, tablets, and desktops.

Chapter 3: Creating and Communicating between Angular Components

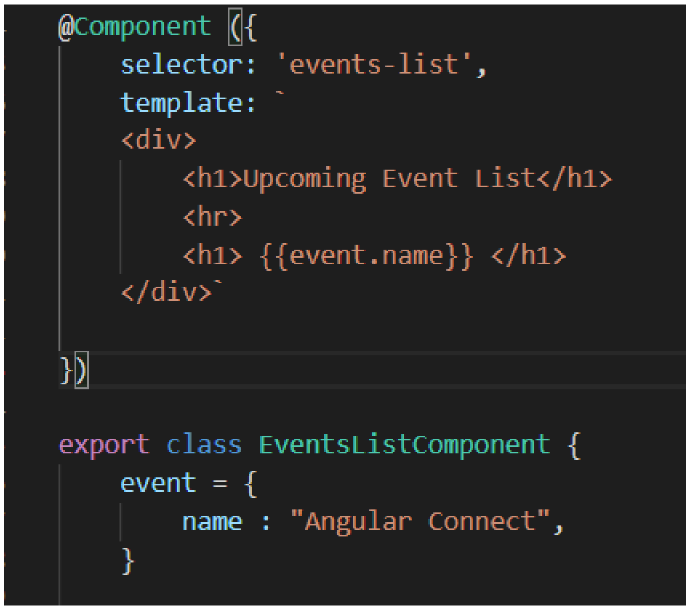
To add multiline Html in the template bracket, replace ‘ ‘ with ` ` (back ticks – tilde option)

Data-Bound components:



```
    template: `
    <div>
        <h1>Upcoming Event List</h1>
    </div>`
```

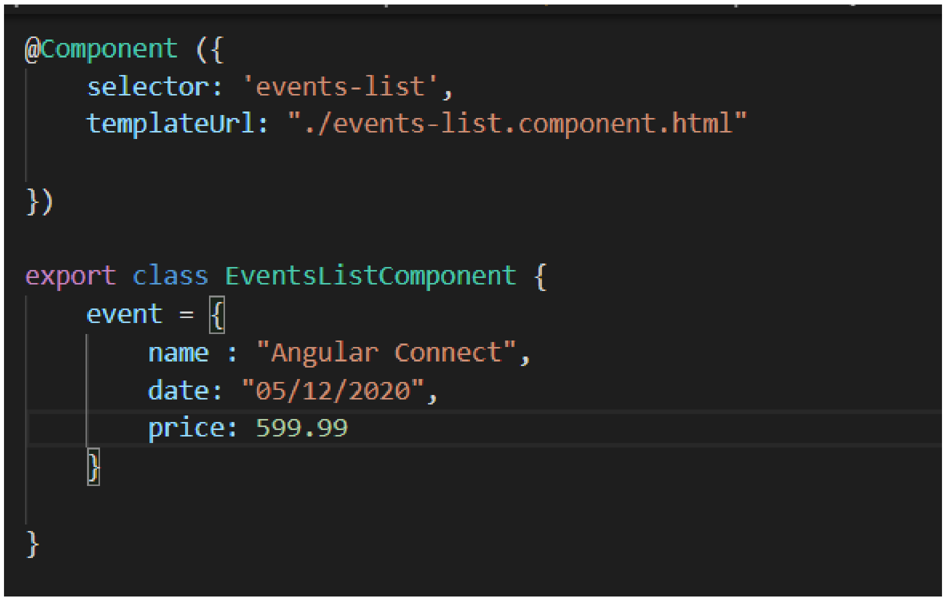
We can create a class inside a .component.ts file reference the values in the template option in the same file.



```
@Component ({
    selector: 'events-list',
    template: `
    <div>
        <h1>Upcoming Event List</h1>
        <hr>
        <h1> {{event.name}} </h1>
    </div>`

})

export class EventsListComponent {
    event = {
        name : "Angular Connect",
    }
```

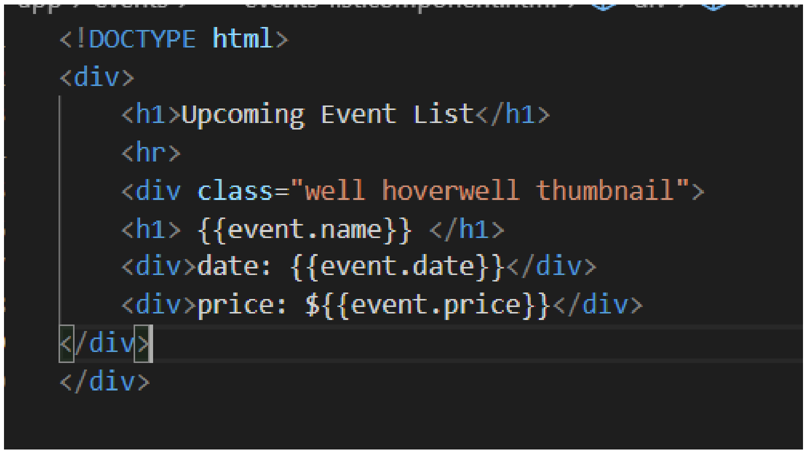
The {{event.name}} is the syntax to represent and reference the data inside an object in the class

We can reference the html code from another file using templateUrl tag.



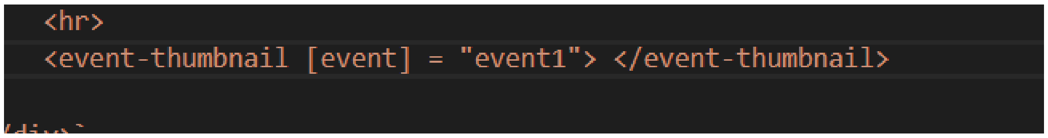
```
@Component ({
    selector: 'events-list',
    templateUrl: "./events-list.component.html"

})

export class EventsListComponent {
    event = {
        name : "Angular Connect",
        date: "05/12/2020",
        price: 599.99
    }

}
```

And inside the html file:



```
<!DOCTYPE html>
<div>
    <h1>Upcoming Event List</h1>
    <hr>
    <div class="well hoverwell thumbnail">
    <h1> {{event.name}} </h1>
    <div>date: {{event.date}}</div>
    <div>price: ${{event.price}}</div>
</div>
</div>
```

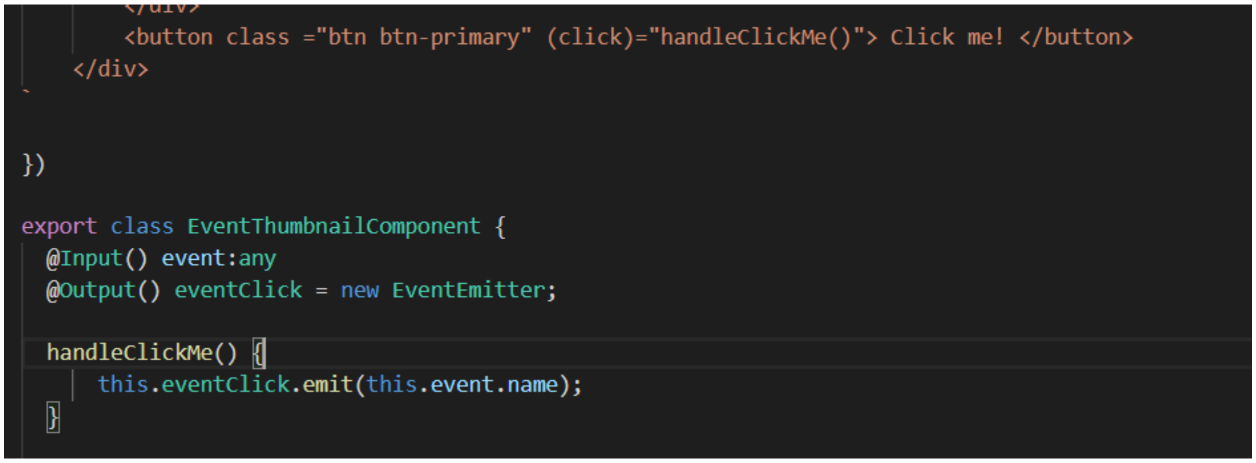
Passing content from the parent to the child component:



```
  <hr>
  <event-thumbnail [event] = "event1"> </event-thumbnail>
```

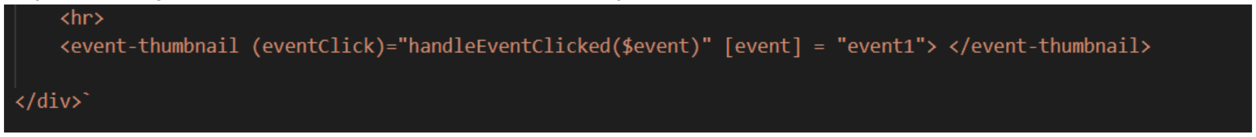
The selector initialized in the child component is <event-thumbnail>
The "event" is initialized in the child component and event1 is initialized in the parent component (events-list.component.ts) to hold the data. Inside the child component (event-thumbnail.component.ts) we import "Input from @angular/core" to use this particular syntax shown in the image above.

Passing content from Child to parent:



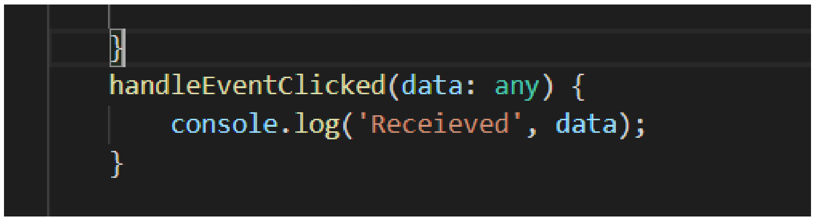
```
        </div>
        <button class ="btn btn-primary" (click)="handleClickMe()"> Click me! </button>
    </div>
`

})

export class EventThumbnailComponent {
  @Input() event:any
  @Output() eventClick = new EventEmitter;

  handleClickMe() {
    this.eventClick.emit(this.event.name);
  }
```

We created a button and a function called handleClickMe() , which call an Output and Event Emitter import. Using the function we send the data to the parent.



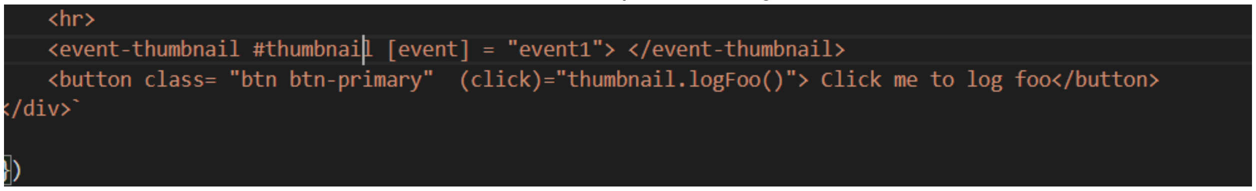
```
    <hr>
    <event-thumbnail (eventClick)="handleEventClicked($event)" [event] = "event1"> </event-thumbnail>

</div>`
```

Where the html has been edited to handle the click. The html shows that an event ($event) will be the input to the function handleEventClicked once the function eventClick has been called in the child.
And the function to show output is:



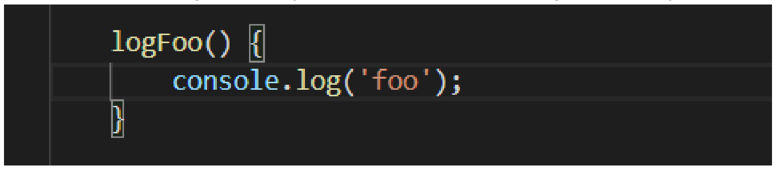
```
    }
    handleEventClicked(data: any) {
        console.log('Receieved', data);
    }
```

We can also reference and call child function from parent using reference variables:



```
    <hr>
    <event-thumbnail #thumbnail [event] = "event1"> </event-thumbnail>
    <button class= "btn btn-primary"  (click)="thumbnail.logFoo()"> Click me to log foo</button>
</div>`

})
```

Here #thumbnail is our ref variable and on click of the button, the log.Foo function is called which exists in the child component (event-thumbnail.component.ts):

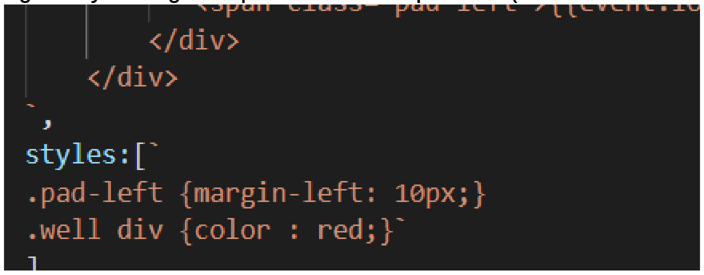


```
    logFoo() {
        console.log('foo');
    }
```

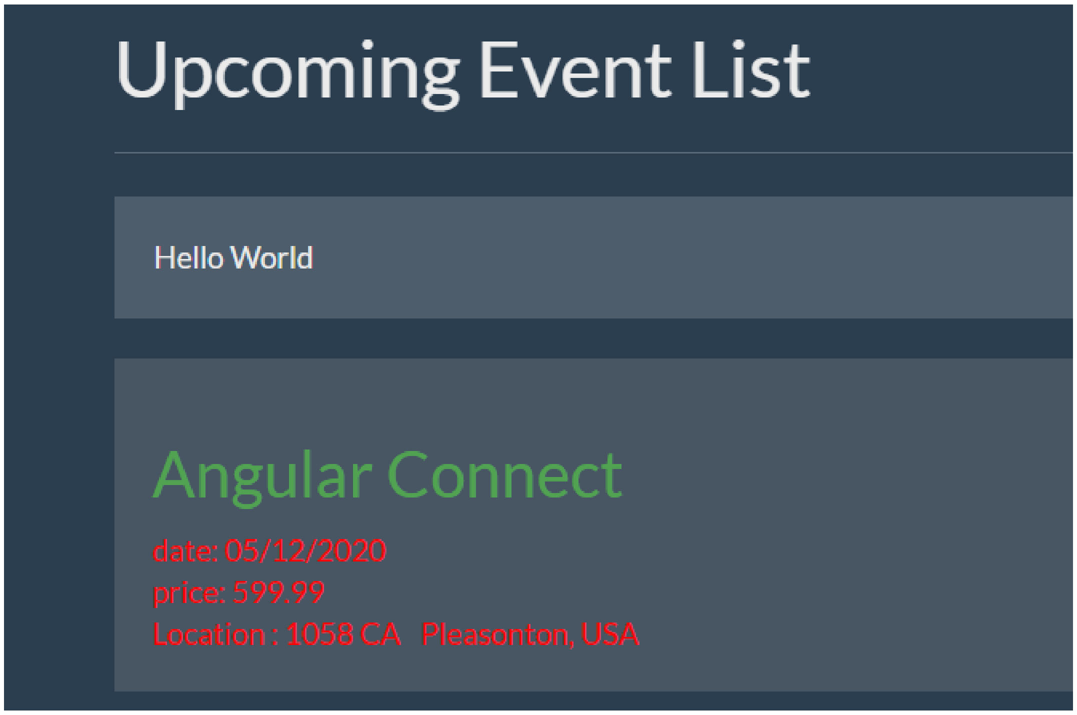
This is more simple that input and output method used previously. But here logFoo is a public function. Also works with referencing values like {{thumbnail.name}} in parent where name is a variable in the child.

3 ways of communicating between parent and child components:
Input properties
Output properties
Template variables.
Styling with CSS:

Based on the location of the style tag , the scope of the styling varies.
Eg: if styles tag is kept in child component (event-thumbnail)



```
            </div>
        </div>
    `,
    styles:[`
    .pad-left {margin-left: 10px;}
    .well div {color : red;}`
    ]
```



Upcoming Event List

Hello World

Angular Connect
date: 05/12/2020
price: 599.99
Location : 1058 CA   Pleasonton, USA

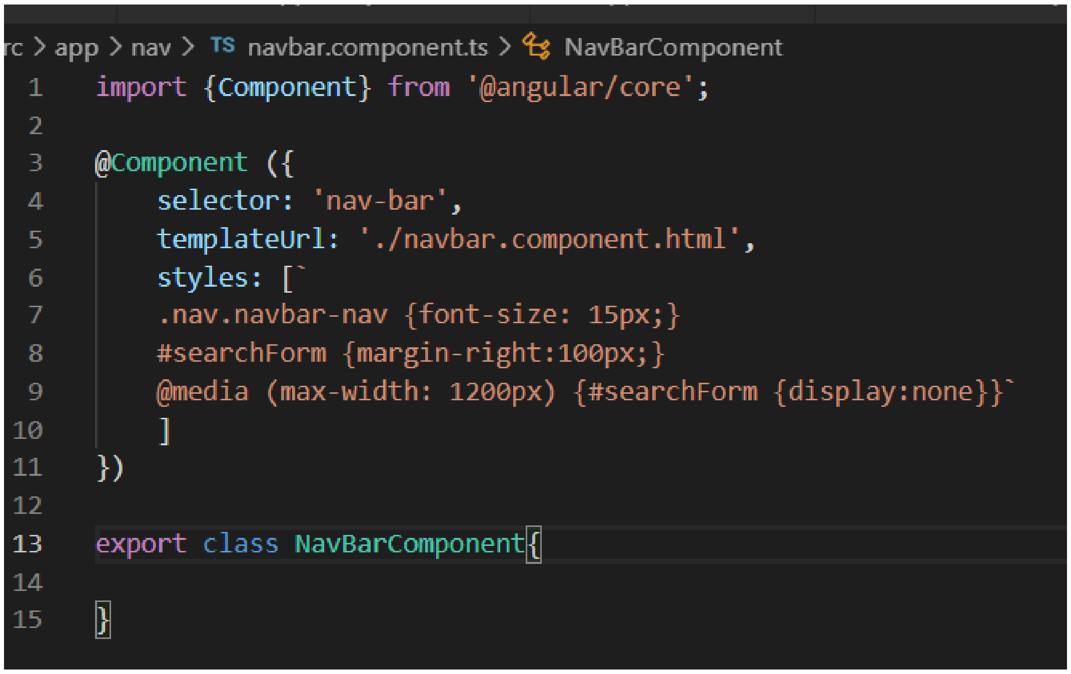
Here only the date, price and location are colored and not the Hello World
. If the .well div style is moved to the parent component, then only Hello World will be colored.

Making changes to style.css will make changes on the global scope.

Adding Component to app: Adding Site header.

Create a folder, name it nav, crate a navbar component, set imports, selector and class.
Set the navbar component in app components inside the template.
To set styles for the nav bar, add styles in nav bar component under template-> styles.

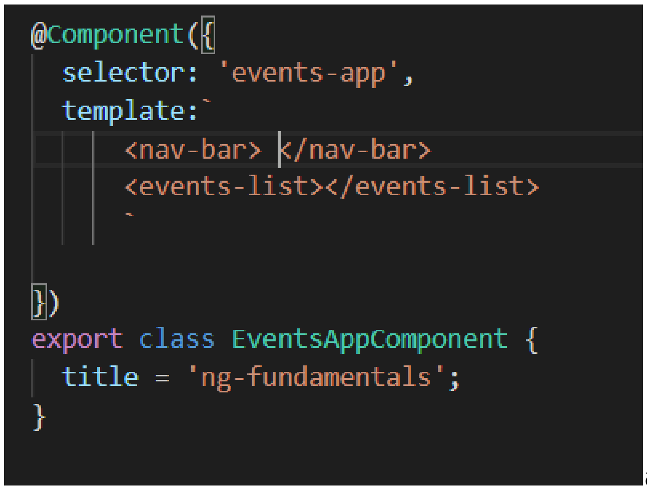
Finally add nav bar component in app module.



rc > app > nav > TS navbar.component.ts > NavBarComponent

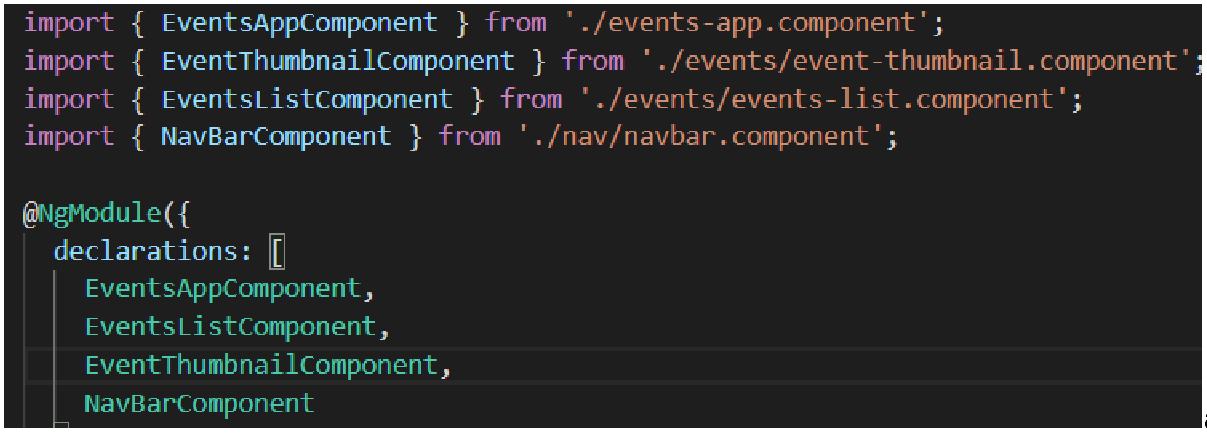
```
import {Component} from '@angular/core';

@Component ({
    selector: 'nav-bar',
    templateUrl: './navbar.component.html',
    styles: [`
    .nav.navbar-nav {font-size: 15px;}
    #searchForm {margin-right:100px;}
    @media (max-width: 1200px) {#searchForm {display:none}}`
    ]
})

export class NavBarComponent{

}
```

navbar comp



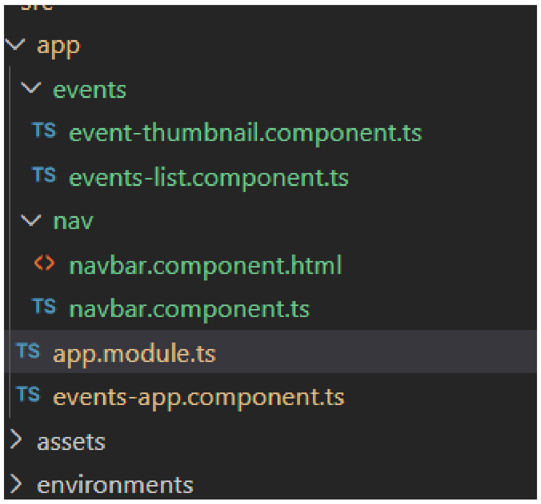
```
@Component({
  selector: 'events-app',
  template:`
      <nav-bar> </nav-bar>
      <events-list></events-list>
      `

})
export class EventsAppComponent {
  title = 'ng-fundamentals';
}
```

app component



```
import { EventsAppComponent } from './events-app.component';
import { EventThumbnailComponent } from './events/event-thumbnail.component';
import { EventsListComponent } from './events/events-list.component';
import { NavBarComponent } from './nav/navbar.component';

@NgModule({
  declarations: [
    EventsAppComponent,
    EventsListComponent,
    EventThumbnailComponent,
    NavBarComponent
```

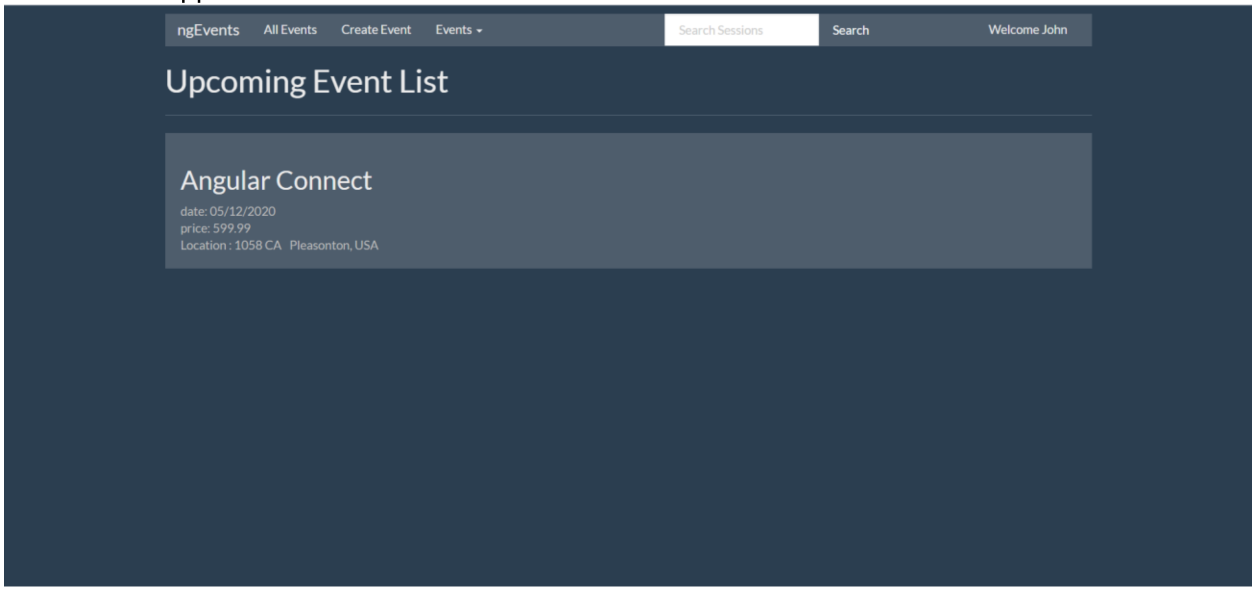
app module



```
app
  events
    event-thumbnail.component.ts
    events-list.component.ts
  nav
    navbar.component.html
    navbar.component.ts
  app.module.ts
  events-app.component.ts
assets
environments
```

The files

End of Ch3 – App:

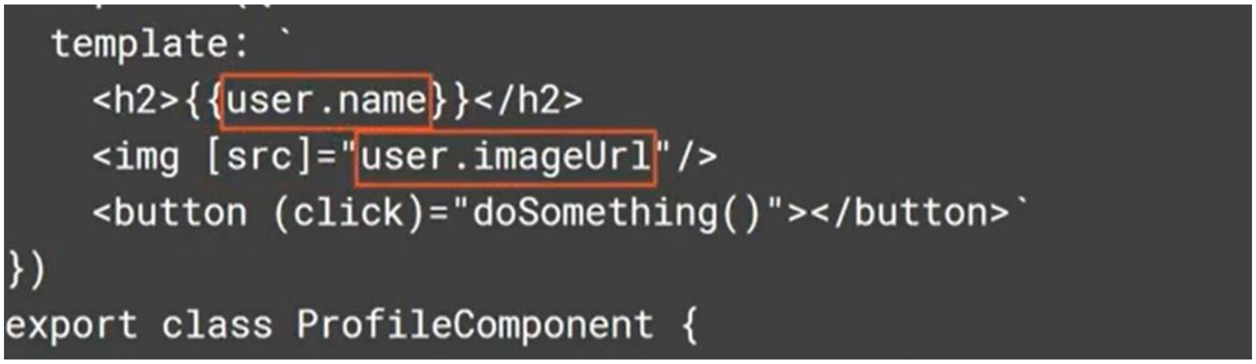


Upcoming Event List

Angular Connect

date: 05/12/2020
price: 599.99
Location : 1058 CA Pleasonton, USA

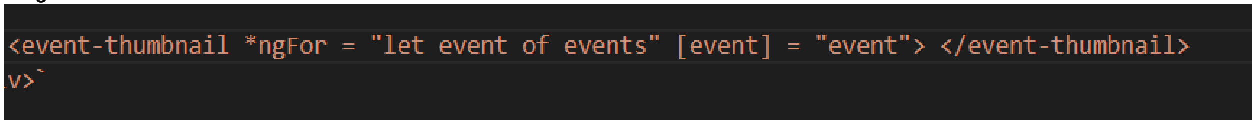
Chapter 4: Exploring Angular template syntax



```
  template: `
    <h2>{{user.name}}</h2>
    <img [src]="user.imageUrl"/>
    <button (click)="doSomething()"></button>`
})
export class ProfileComponent {
```

Interpolation: user.name
Property binding: [src] -> use square brackets
Expressions: user.name, user.imageUrl
Event bindings: (click) -> use parenthesis
Statements: “doSomething()” -> when button is clicked this function is called

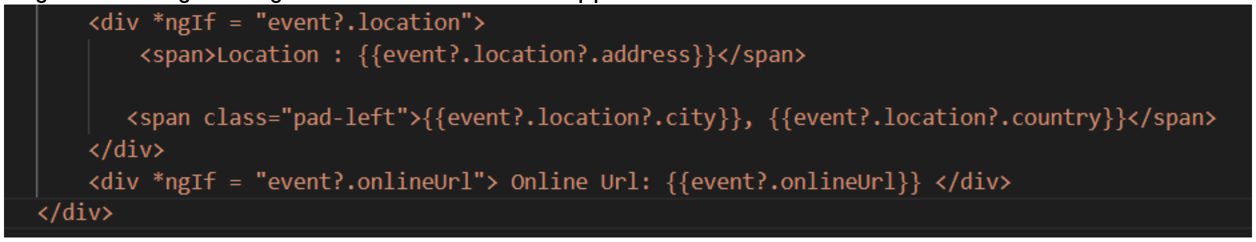
*ngFor:



```
<event-thumbnail *ngFor = "let event of events" [event] = "event"> </event-thumbnail>
v>`
```

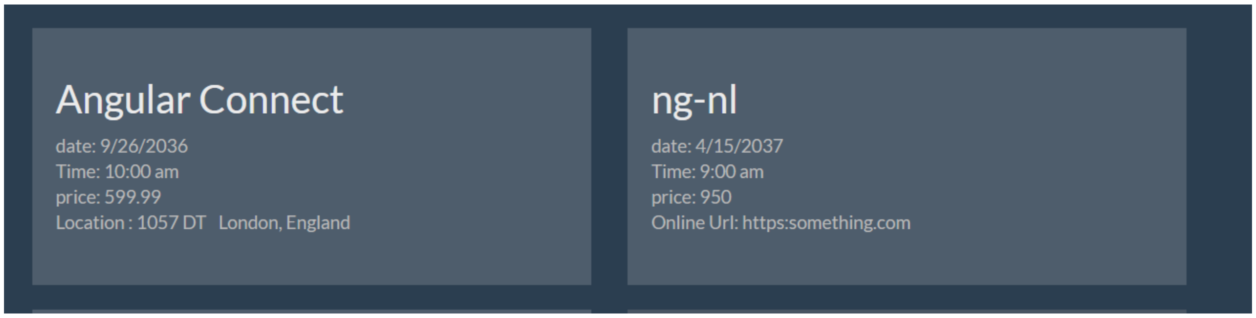
*ngFor = “let event of events” [event] = “event”
Here event is being assigned every element of the array events in the same file. and its being displayed by the following [event] from the thumbnail folder.

*ngIf: used to guard against null values in the app



```
    <div *ngIf = "event?.location">
        <span>Location : {{event?.location?.address}}</span>

       <span class="pad-left">{{event?.location?.city}}, {{event?.location?.country}}</span>
    </div>
    <div *ngIf = "event?.onlineUrl"> Online Url: {{event?.onlineUrl}} </div>
</div>
```

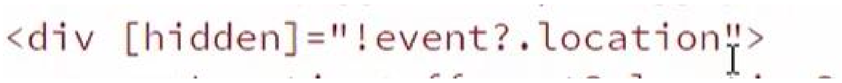
*ngIf = ... this only sets that row in the app if there is a value provided to the expression , else removes it



Angular Connect
date: 9/26/2036
Time: 10:00 am
price: 599.99
Location : 1057 DT   London, England

ng-nl
date: 4/15/2037
Time: 9:00 am
price: 950
Online Url: https:something.com

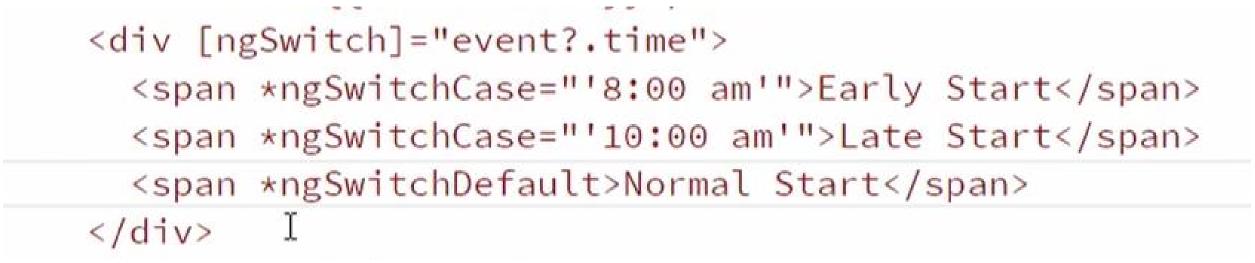
Here, the onlineUrl for the first one and location for the second one have been removed because they haven't been given any value.



```
<div [hidden]="!event?.location">
```

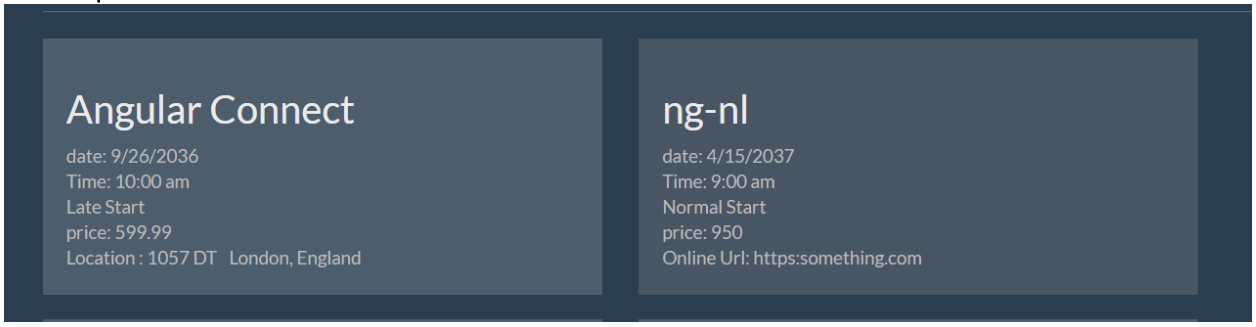
This works just like *ngIf but its an html element and not an angular element. It shows up as hidden in the console (Fn + F12)

ngSwitch:



```
    <div [ngSwitch]="event?.time">
      <span *ngSwitchCase="'8:00 am'">Early Start</span>
      <span *ngSwitchCase="'10:00 am'">Late Start</span>
      <span *ngSwitchDefault>Normal Start</span>
    </div>
```

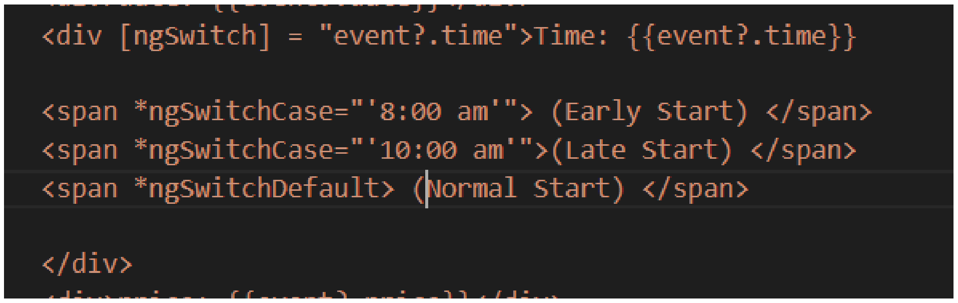
Works like any other switch and case statements
The output:



Angular Connect
date: 9/26/2036
Time: 10:00 am
Late Start
price: 599.99
Location : 1057 DT London, England

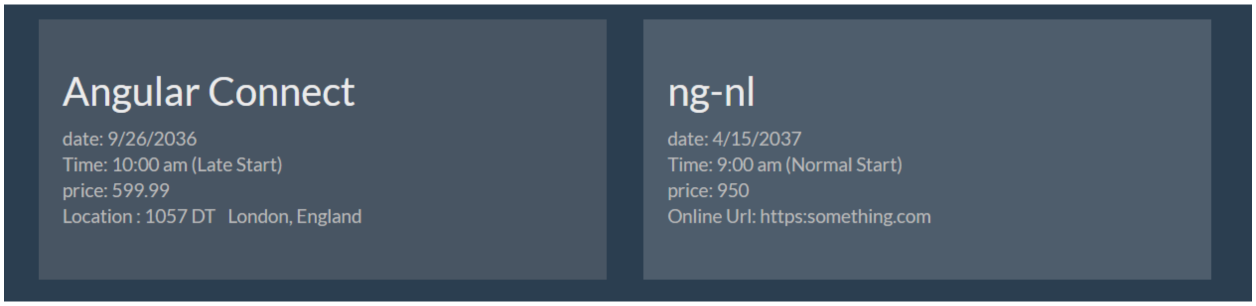
ng-nl
date: 4/15/2037
Time: 9:00 am
Normal Start
price: 950
Online Url: https:something.com

It can be used inside the div statements of time as well so the text appears right next to the time:



```
  <div [ngSwitch] = "event?.time">Time: {{event?.time}}

  <span *ngSwitchCase="'8:00 am'"> (Early Start) </span>
  <span *ngSwitchCase="'10:00 am'">(Late Start) </span>
  <span *ngSwitchDefault> (Normal Start) </span>

  </div>
```

Output:



Angular Connect

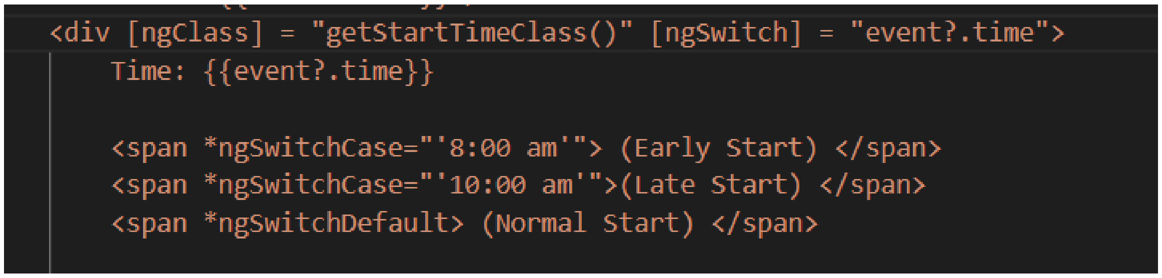
date: 9/26/2036
Time: 10:00 am (Late Start)
price: 599.99
Location : 1057 DT   London, England

ng-nl

date: 4/15/2037
Time: 9:00 am (Normal Start)
price: 950
Online Url: https:something.com

ngSwitch and case statements should have the same datatype.

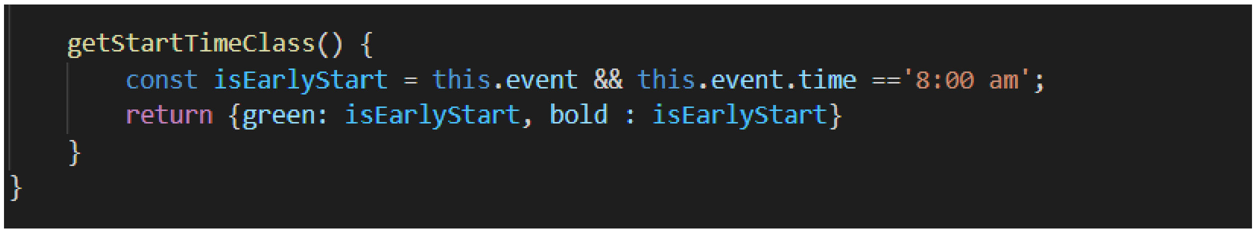
Styling with ngClass:



```
<div [ngClass] = "getStartTimeClass()" [ngSwitch] = "event?.time">
    Time: {{event?.time}}

    <span *ngSwitchCase="'8:00 am'"> (Early Start) </span>
    <span *ngSwitchCase="'10:00 am'">(Late Start) </span>
    <span *ngSwitchDefault> (Normal Start) </span>
```

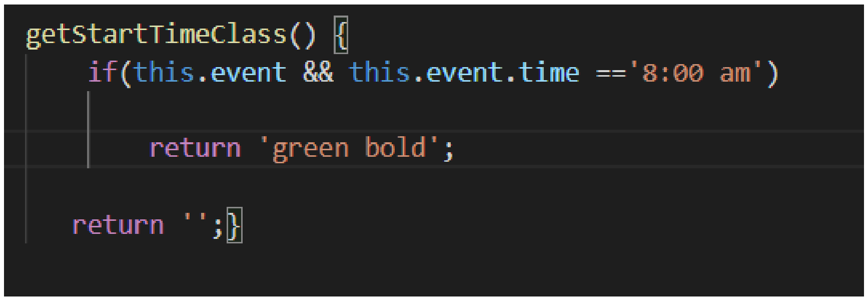
ngClass will reference the classes returned by the function with the styles set in the file.

the function can be:



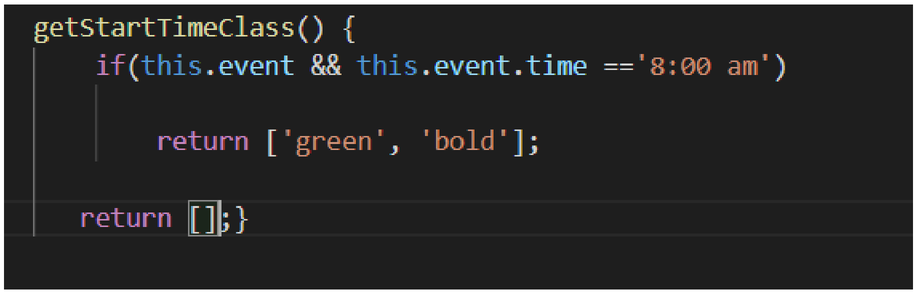
```
    getStartTimeClass() {
        const isEarlyStart = this.event && this.event.time =='8:00 am';
        return {green: isEarlyStart, bold : isEarlyStart}
    }
}
```

Or



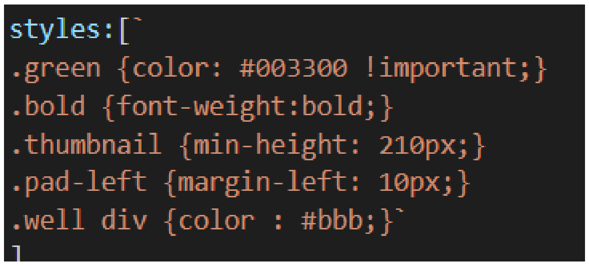
```
 getStartTimeClass() {
     if(this.event && this.event.time =='8:00 am')

         return 'green bold';

    return '';}
```

Or



```
getStartTimeClass() {
    if(this.event && this.event.time =='8:00 am')

        return ['green', 'bold'];

    return [];}
```

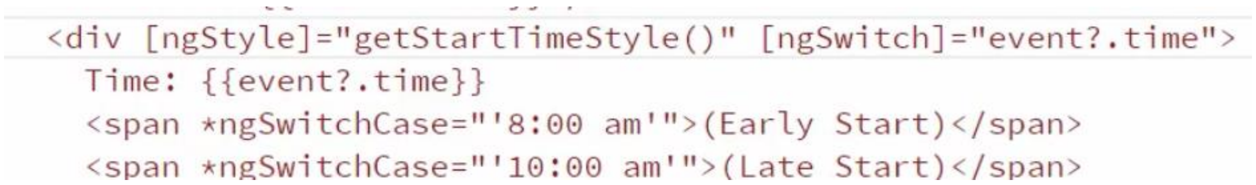
Where the styles are:



```
styles:[`
.green {color: #003300 !important;}
.bold {font-weight:bold;}
.thumbnail {min-height: 210px;}
.pad-left {margin-left: 10px;}
.well div {color : #bbb;}`
]
```

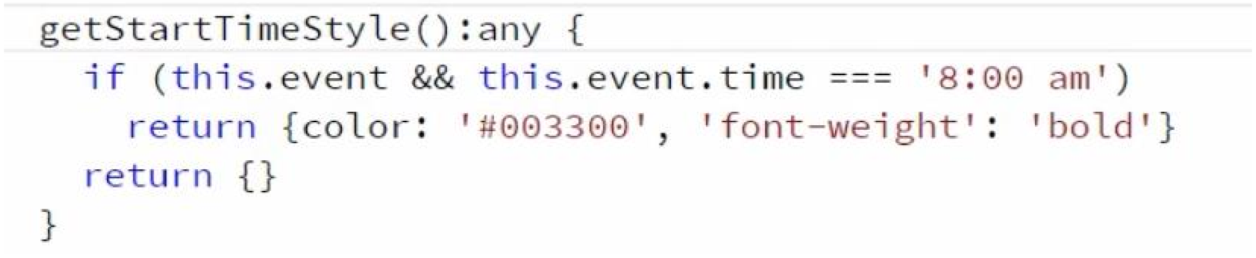
Even if the div tag containing ngClass has another css applied using class , then both the references will be applied.

Using ngStyle:



```
  <div [ngStyle]="getStartTimeStyle()" [ngSwitch]="event?.time">
    Time: {{event?.time}}
    <span *ngSwitchCase="'8:00 am'">(Early Start)</span>
    <span *ngSwitchCase="'10:00 am'">(Late Start)</span>
```

Similar to ngClass but the css has to mentioned specifically



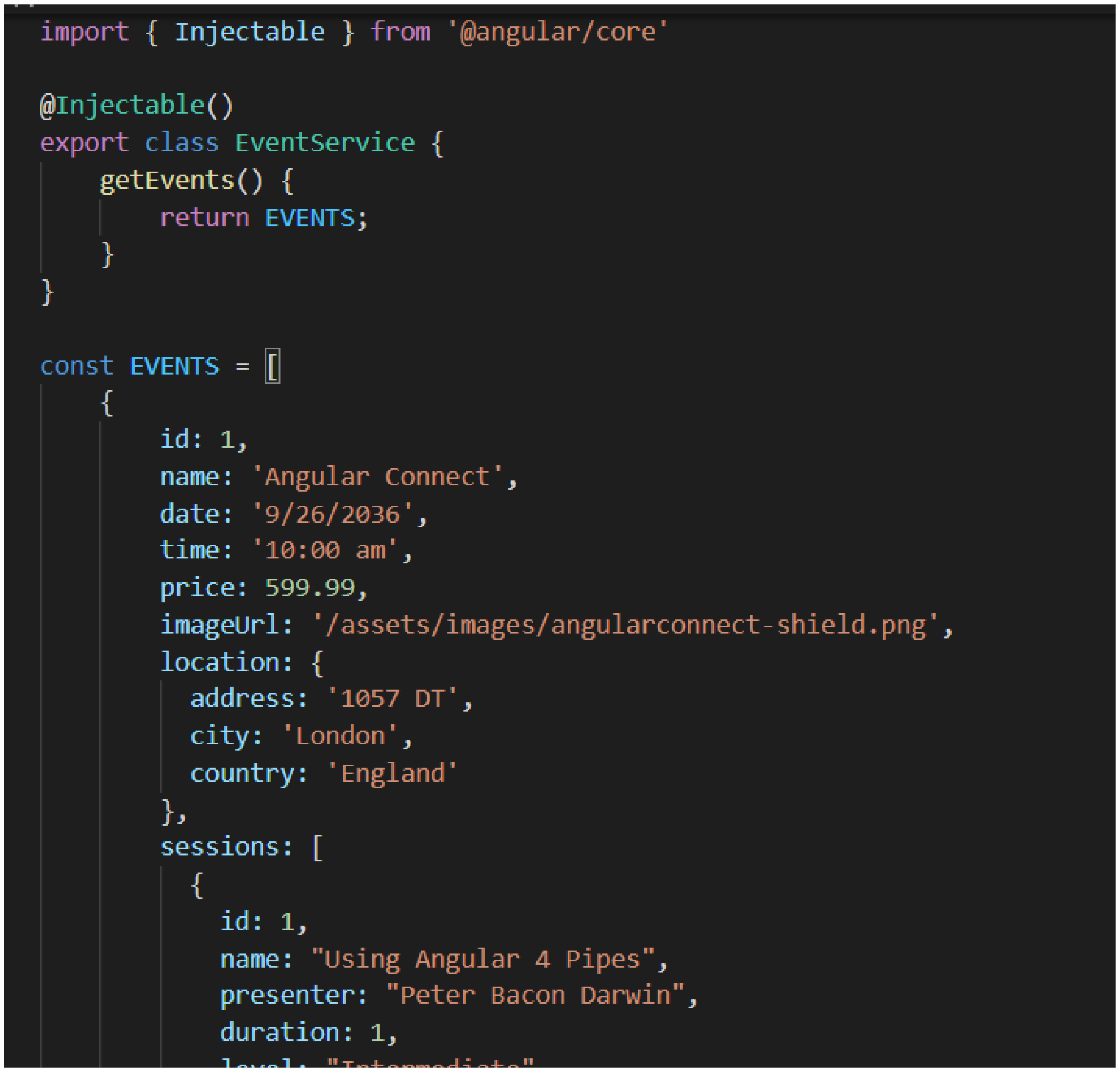
```
getStartTimeStyle():any {
  if (this.event && this.event.time === '8:00 am')
    return {color: '#003300', 'font-weight': 'bold'}
  return {}
}
```

With this it can specific and not added into the style tags shown above while using classes.

Chapter 5: Creating reusable Angular Services

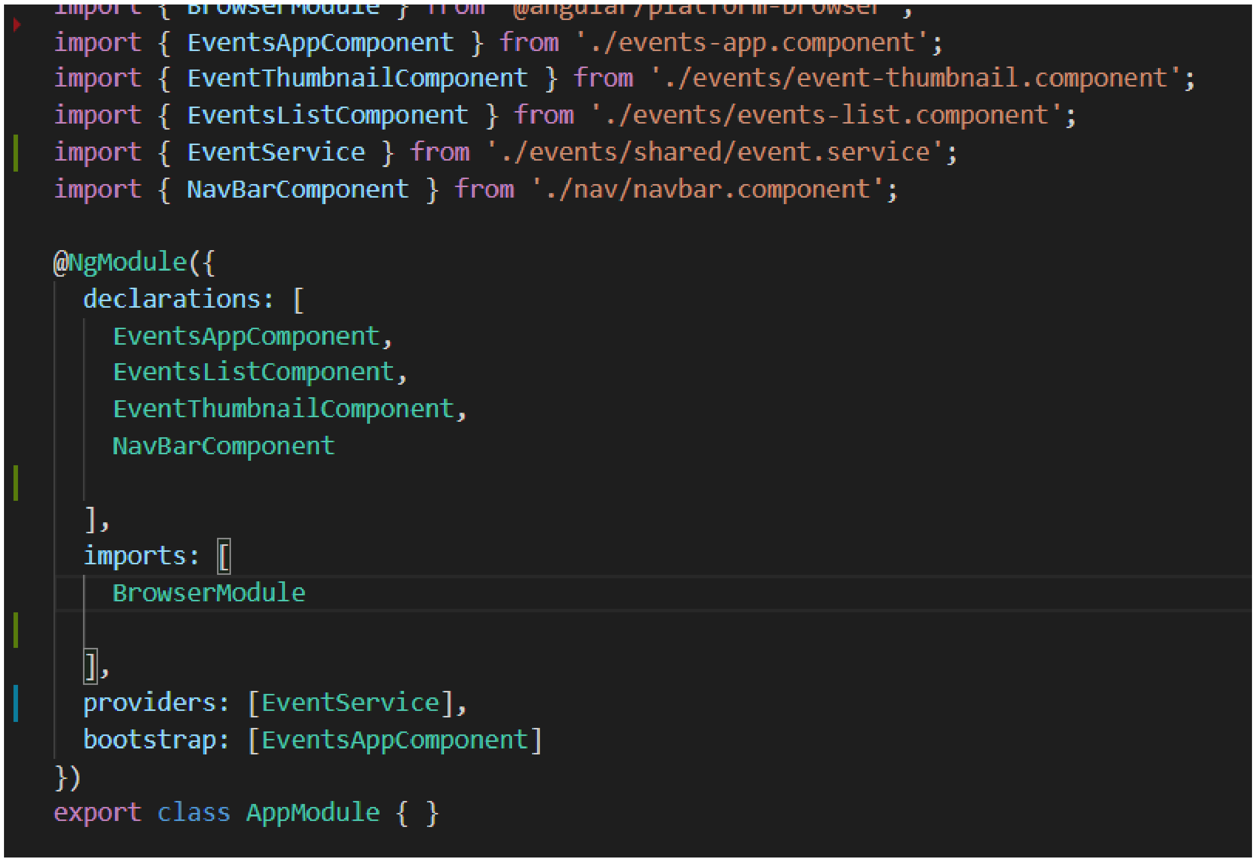
To send all events to another file as a service:

1. Create a folder names shared inside events and a file named event.service.ts
2. inside event.service .ts:



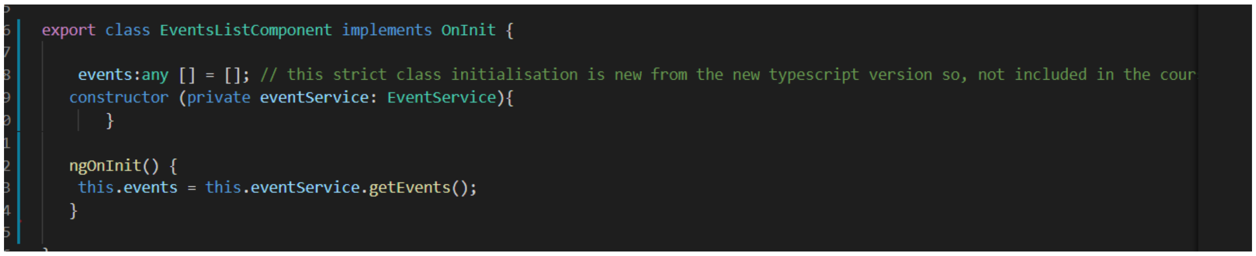
```
import { Injectable } from '@angular/core'

@Injectable()
export class EventService {
    getEvents() {
        return EVENTS;
    }
}

const EVENTS = [
    {
        id: 1,
        name: 'Angular Connect',
        date: '9/26/2036',
        time: '10:00 am',
        price: 599.99,
        imageUrl: '/assets/images/angularconnect-shield.png',
        location: {
          address: '1057 DT',
          city: 'London',
          country: 'England'
        },
        sessions: [
          {
            id: 1,
            name: "Using Angular 4 Pipes",
            presenter: "Peter Bacon Darwin",
            duration: 1,
```

3. In app Module.ts , import Event Service as a provider



```
import { BrowserModule } from '@angular/platform-browser';
import { EventsAppComponent } from './events-app.component';
import { EventThumbnailComponent } from './events/event-thumbnail.component';
import { EventsListComponent } from './events/events-list.component';
import { EventService } from './events/shared/event.service';
import { NavBarComponent } from './nav/navbar.component';

@NgModule({
  declarations: [
    EventsAppComponent,
    EventsListComponent,
    EventThumbnailComponent,
    NavBarComponent

  ],
  imports: [
    BrowserModule

  ],
  providers: [EventService],
  bootstrap: [EventsAppComponent]
})
export class AppModule { }
```

4. Import OnInit and Eventservice into events-list.components
 And add in the constructor and functions:



```
export class EventsListComponent implements OnInit {

   events:any [] = []; // this strict class initialisation is new from the new typescript version so, not included in the cour
  constructor (private eventService: EventService){
    }

  ngOnInit() {
   this.events = this.eventService.getEvents();
  }

```

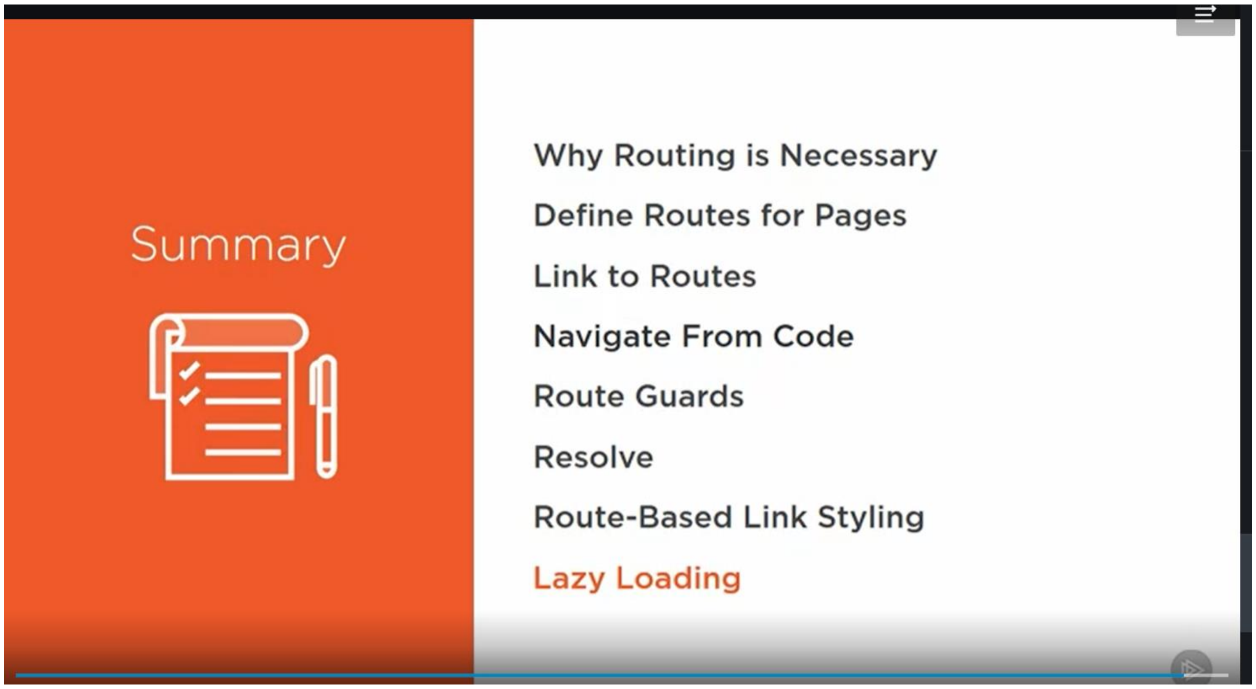
That’s how we store all the events as a service provider.

Toastr module : Doubt – have not included in the code

Verify Ch 5 video 4 to add in.

Main vulnerability: toastr module requires jquery to be version 3+, but ngf bootstrap

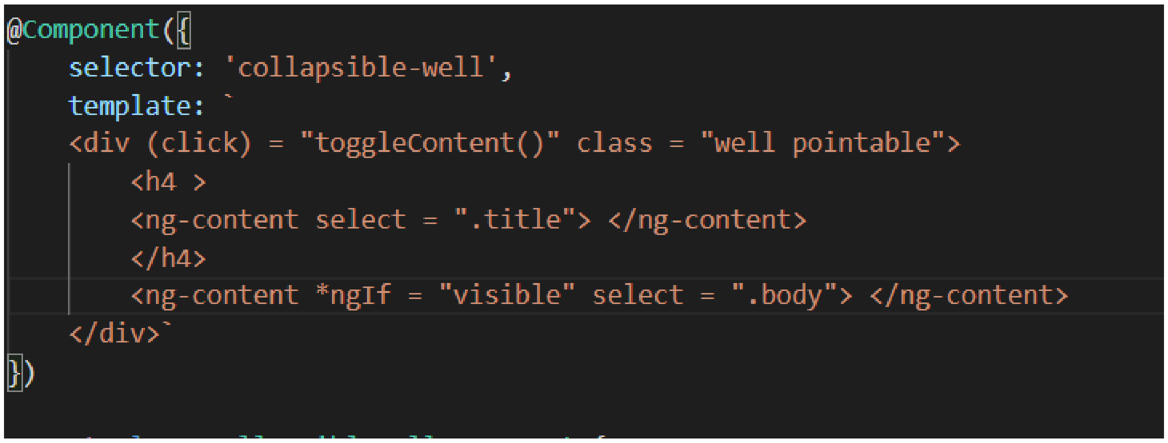
Chapter 6:
Routing and navigating pages


Summary

- Why Routing is Necessary
- Define Routes for Pages
- Link to Routes
- Navigate From Code
- Route Guards
- Resolve
- Route-Based Link Styling
- Lazy Loading

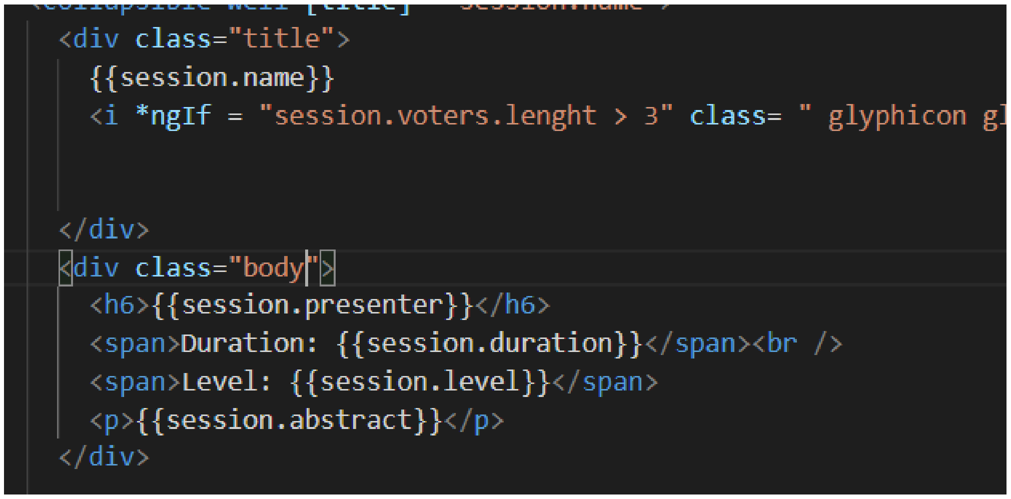
Chapter 9:

Ng content selector types:



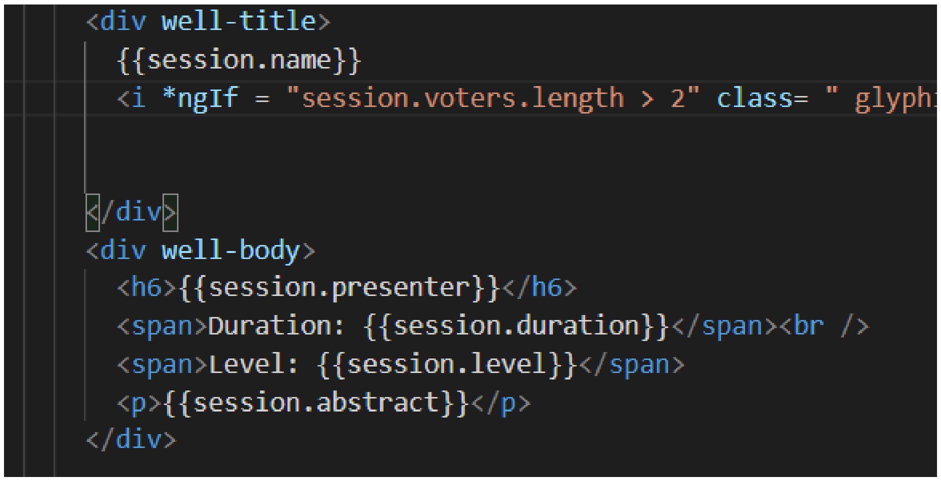
```
@Component({
    selector: 'collapsible-well',
    template: `
    <div (click) = "toggleContent()" class = "well pointable">
        <h4 >
        <ng-content select = ".title"> </ng-content>
        </h4>
        <ng-content *ngIf = "visible" select = ".body"> </ng-content>
    </div>`
})
```

Using class selector as class = title and class = body to call the content where and when required

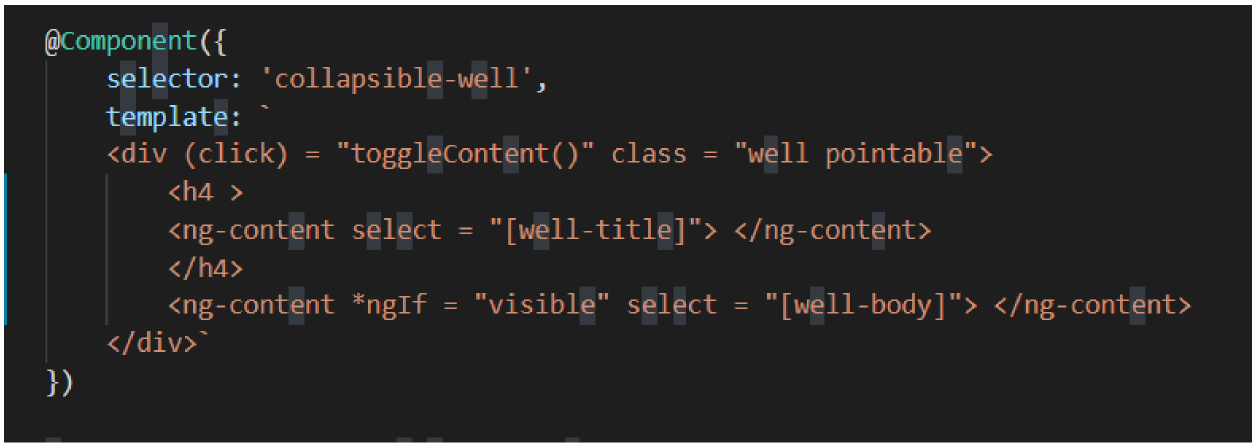


```
  <div class="title">
    {{session.name}}
    <i *ngIf = "session.voters.lenght > 3" class= " glyphicon gl


  </div>
  <div class="body">
    <h6>{{session.presenter}}</h6>
    <span>Duration: {{session.duration}}</span><br />
    <span>Level: {{session.level}}</span>
    <p>{{session.abstract}}</p>
  </div>
```

Or



```
    <div well-title>
      {{session.name}}
      <i *ngIf = "session.voters.length > 2" class= " glyphi


    </div>
    <div well-body>
      <h6>{{session.presenter}}</h6>
      <span>Duration: {{session.duration}}</span><br />
      <span>Level: {{session.level}}</span>
      <p>{{session.abstract}}</p>
    </div>
```



```
@Component({
    selector: 'collapsible-well',
    template: `
    <div (click) = "toggleContent()" class = "well pointable">
        <h4 >
        <ng-content select = "[well-title]"> </ng-content>
        </h4>
        <ng-content *ngIf = "visible" select = "[well-body]"> </ng-content>
    </div>`
})
```

Using attribute selector well-body and well-title to not have mix ups with names during styling