REVIEW MEETING 2

01/02/2020 TO 08/02/2020

CHAPTER 2: GETTING STARTED WITH ANGULAR

- *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: demo*

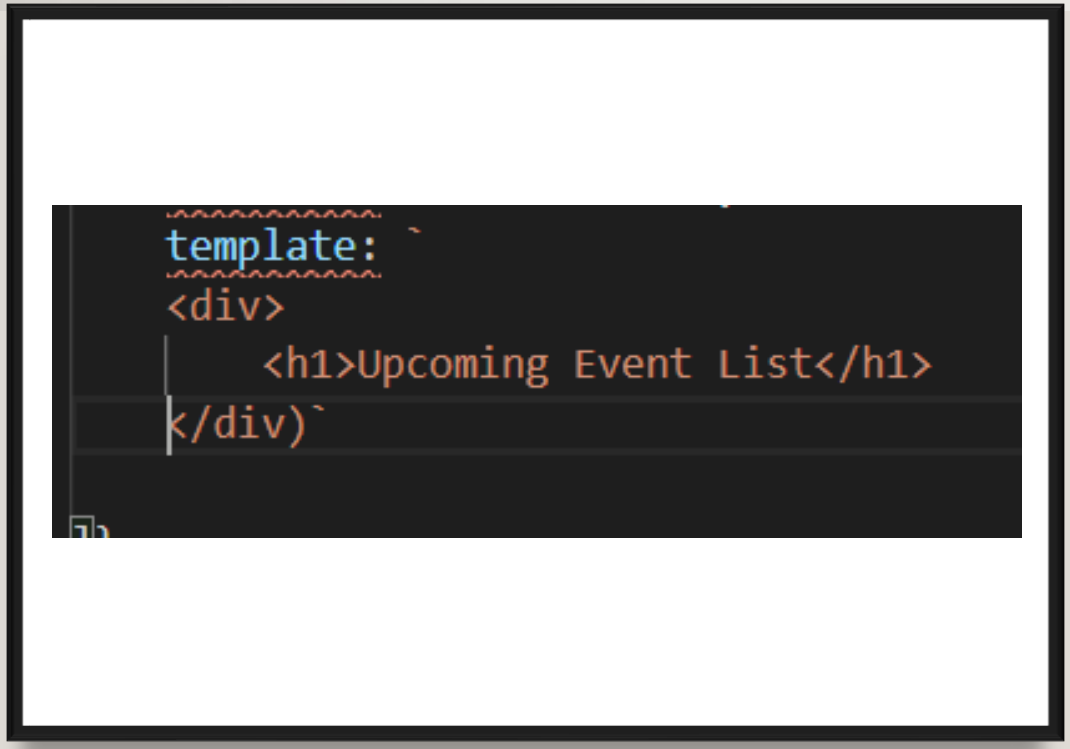
ANGULARJS VS ANGULAR

CHAPTER 2: CONT..

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.

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

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:

3 ways of communicating between parent and child components:

- Input properties
- Output properties
- Template variables.

INPUT VARIABLES

Passing content from the parent to the child component:

- 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.

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

OUTPUT VARIABLES

Passing content from Child to parent:

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.

```
        <button class ="btn btn-primary" (click)="h
    </div>
`


})

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

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

```
    <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);
    }
```

TEMPLATE VARIABLES

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

})
```

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

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

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):

- 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.

ANOTHER WAY OF REFERENCING VARIABLES

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

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

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

})

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

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
    }

}
```

```
<!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>
```

NAVIGATION BAR (CREATING AND CONNECTING COMPONENTS)

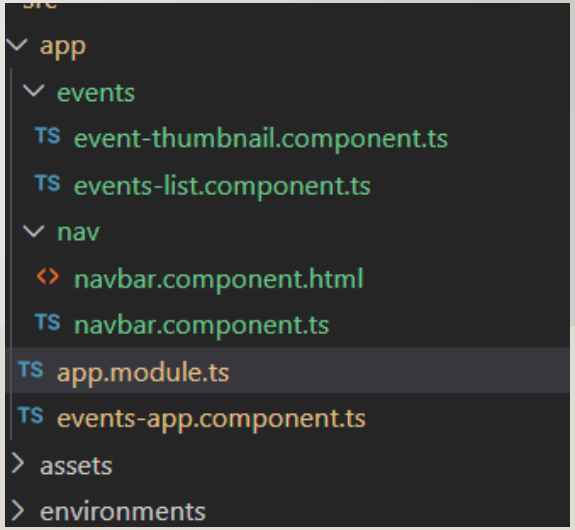
- Create a folder, name it nav, crate a navbar component, set imports, selector and class.
- Set the navbar component in app components inside the template.
- To set styles for the nav bar, add styles in nav bar component under template-> styles.

Finally add nav bar component in app module.

NAV BAR ADDED

```
rc > app > nav > TS navbar.component.ts > NavBarComponent
1   import {Component} from '@angular/core';
2
3   @Component ({
4       selector: 'nav-bar',
5       templateUrl: './navbar.component.html',
6       styles: [`
7       .nav.navbar-nav {font-size: 15px;}
8       #searchForm {margin-right:100px;}
9       @media (max-width: 1200px) {#searchForm {display:none}}`
10      ]
11  })
12
13  export class NavBarComponent{
14
15  }
```

Nav Bar component

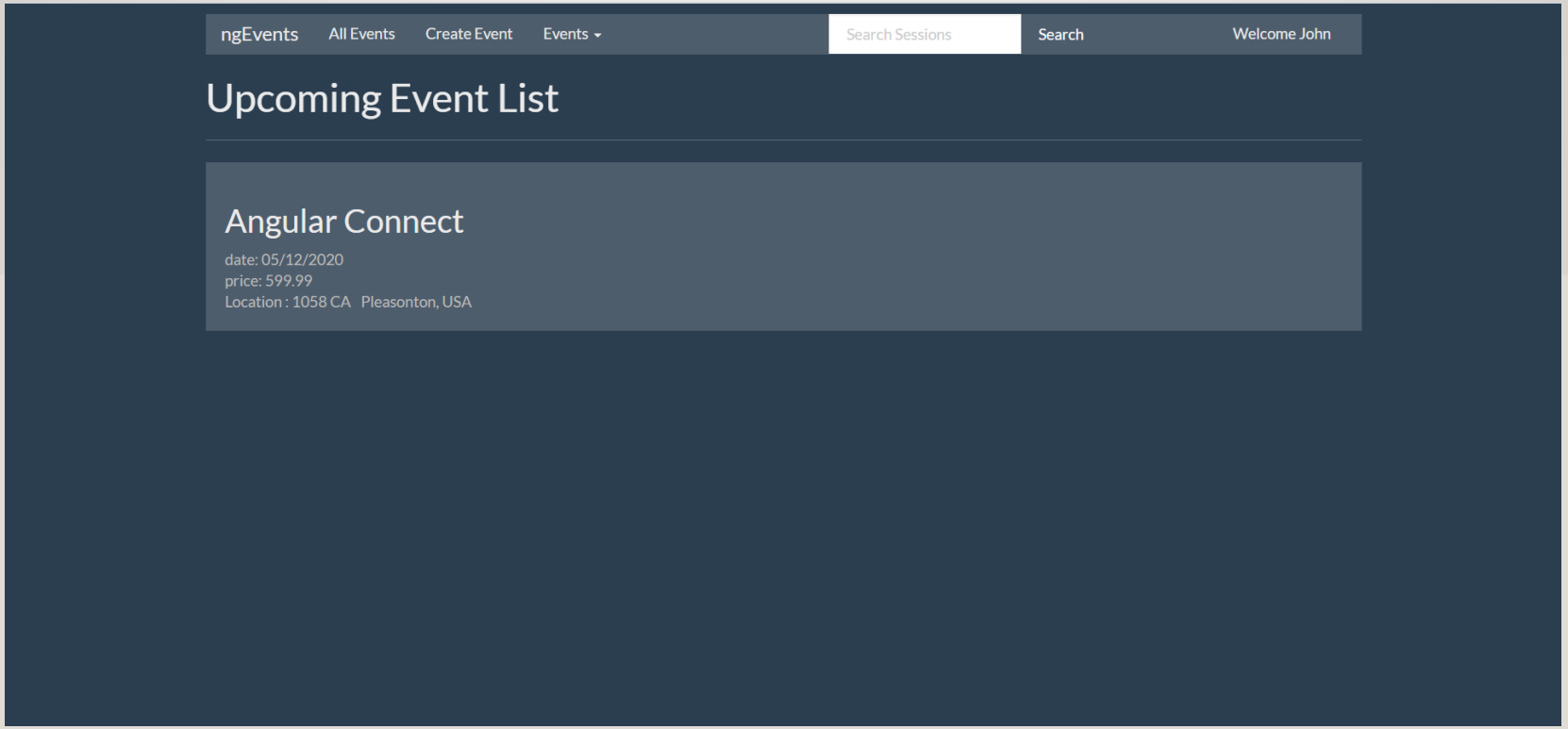



```
import { EventsAppComponent } from './events-app.component';
import { EventThumbnailComponent } from './events/event-thumbnail.component';
import { EventsListComponent } from './events/events-list.component';
import { NavBarComponent } from './nav/navbar.component';

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

App module

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

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

App component



THE APPLICATION AT THE END OF CHAPTER 3

CHAPTER 4: EXPLORING ANGULAR TEMPLATE SYNTAX

- 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

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

- *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.

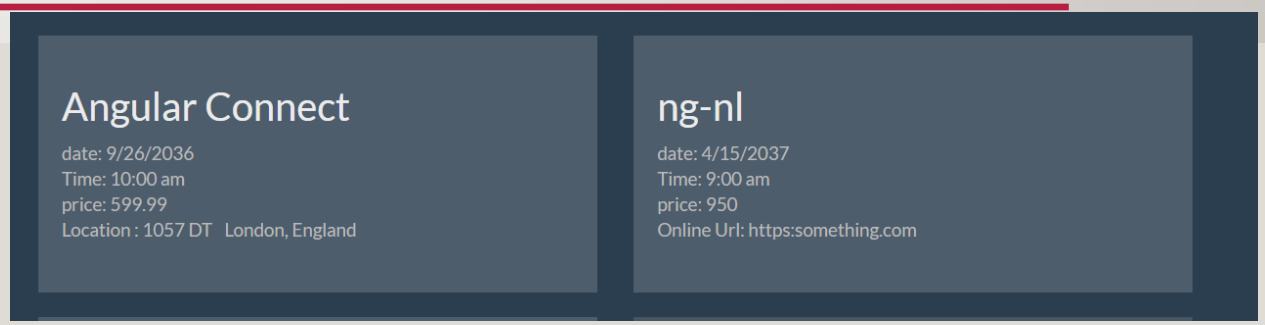
```
<event-thumbnail *ngFor = "let event of events" [event] = "event"> </event-thumbnail>
v>`
```

*NGIF: USED TO GUARD AGAINST NULL VALUES IN THE APP

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

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

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)




```
    <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>
```

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

*NGSWITCH

```
<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>
```

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>
```

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

        return 'green bold';

    return '';}
```

- The three classes shown are the forms of the function which can be used to create the class.
- All styles are referred from the styles tag placed inside the components

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

        return ['green', 'bold'];

    return [];}
```

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

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>
```

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

- Similar to ngClass but the css has to mentioned specifically
- 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: Show code
3. In app Module.ts , import Event Service as a provider
4. Import OnInit and Eventservice into events-list.components

And add in the constructor and functions:

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

App module

```
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 { }
```

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,
            level: "Intermediate"
```

Adding onInit and eventService to EventList component

```
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();
   }

}
```

Using “toastr” module to display toast messages on click on the blocks