Angular fundamentals

1. **Create project**

* ng new ng-projectName

1. **Install bootstrap**

* Npm install ngf-bootstrap –save
* Angular json: add inside styles bootstrap.min.css and inside script add bootstrap.js

1. **Input**

* Component1TS: @Input() event:any;
* HTMLComponent2: <[event]=”event1”>   
  variable inside component2 event1={}

1. **Output**  
   - Component 1 TS- eventClick = new EventEmitter()  
   function() {  
    this.eventClick.emit(“text”)  
   }

- Component 2 – <div (eventClick)=” function1($event)”>  
function1 (data) {  
 console.log(data);  
}

1. **Call Parent function via templete inside child component**- On child component add following   
   <event-thumbnail #thumbnail></event-thumbnail>  
   <button (click)=”thumbnail.functionParent()>
2. **Example**\*\* Interpolation  
   <h2> {{ user.name }}</h2  
     
   \*\* Property binding  
   <img [src]=”user.imgUrl”/>  
     
   \*\* If variable inside html is null  
   event?.name  
     
   \*\* Delete element if value is false  
    \*ngIf=” ”  
     
   \*\* Hide element if value is false  
   [hidden]=” ”  
     
   \*\*NgSwitch (example)  
   <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>  
  
\*\*NgClass  
[class.green] = “event.time == ‘10’”  
  
[ngClass]=”{green: event.time ==’10’,  
 bold: event.time ==’8’}”  
  
[ngClass]=function()  
[style.color]=expression  
[ngStyle]=”{ ‘color’:””, ‘font’:’’}”  
  
\*\*Cast to number  
- Add + before element - + this.element  
  
\*\*Cast to boolean  
- Add !! before element - !!this.element

1. **Implement toaster (optional)**- npm install toastr—save  
   - Angular.json in style and script add toastr css and js file
2. **Routing and Navigating**8.1Example  
   - <router-outlet></ router-outlet>  
   - appRoutes = [ { path: 'events', component: EventsListComponent, redirectTo: ', pathMatch: 'full' } ]  
   -Inside module imports add following:  
   RouterModule.forRoot(appRoutes)  
   - Check inside index.html -> <base href=”/”>  
     
   8.2 Route with params  
   - { path: 'events/:id', component: EventDetailsComponent }  
   - Read ‘id’ inside component – Add constructor route:ActivatedRoute  
   (id) - +this.route.snapshot.params[‘id]  
     
   8.3 Go to route via html  
   [routerLink]=[‘/events’, event.id]  
     
   8.4 Navigate vi code  
   - Constructor router:Router  
   - this.router.navigate([‘/events’]);  
     
   8.5 Guard  
   \*\*Create service and add to module  
   \*\*import {} from “@angular/router”  
     
   1) Can Activate  
    CanActivate(route:ActivatedRouteSnapshot) {  
    /\*Read id\*/  
    +route.params[‘id];  
    this.router.navigate([‘/404’]);  
    - Add canActivate to route  
    { path: 'events/:id', component: EventDetailsComponent, canActivate: [EventRouteActivator]}  
     
   2)DeActivate  
     
   - In route add canDeactivate  
   { path: 'events/new', component: CreateEventComponent, canDeactivate: ['canDeactivateCreateEvent']}  
   -In module add providers   
   {

provide: 'canDeactivateCreateEvent',

useValue: checkDirtyState

}  
- checkDirtyState function should return boolean value

8.6 Resolver

- Create resolver service - export class EventListResolver implements Resolve<any>

- Add constructor for service that we should use

- map from rxjs/operators

- Add resolve function (example):  
resolve() {

return this.eventService.getEvents().pipe(map(events => events));

}

- In module import providers EventListResolver

- In route add following:  
{ path: 'events', component: EventsListComponent, resolve: {events: EventListResolver} }

- Inside component add constructor ActivatedRoute and read events data:  
this.events = this.route.snapshot.data['events'];

8.7 Active links  
- To HTML link add following :  
routerLinkActive="active"  
- If link has same prefix add following:  
[routerLinkActiveOptions]="{exact:true}"  
- Make css changes for active link, for example li > a.active

8.9 Lazy loading

- Create new module  
- In new module imports: [ RouterModule.forChild]  
- Add to parent route for new module:  
{ path: 'user', loadChildren:'./user/user.module#UserModule'}   
- Child route add following:  
{ path: 'profile', component: ProfileComponent}  
- Add router link to html which combinate two path:  
[routerLink]="['user/profile']"

1. **Ng-content - example**  
     
   - collapsible-well  
     
   @Component({

selector: 'collapsible-well',

template: `

<div (click)="toggleContent()" class="well pointable">

<h4>

<ng-content select=".title"></ng-content>

</h4>

<ng-content \*ngIf="visible" select="[well-body]"></ng-content>

</div>

`

})  
  
- session-list.component.html

<collapsible-well [title]="session.name">

<div class="title">

{{session.name}}

<i \*ngIf="session.voters.length > 3" class="glyphicon glyphicon-fire" style="color:red"></i>

</div>

<div well-body>

<h6>{{session.presenter}}</h6>

<span>Duration: {{session.duration}}</span><br />

<span>Level: {{session.level}}</span>

<p>{{session.abstract}}</p>

</div>

</collapsible-well>

1. **Create Pipe (Example)**  
     
   import { Pipe, PipeTransform } from "@angular/core";

@Pipe({name: 'duration'})

export class DurationPipe implements PipeTransform {

transform(value: number): string {

switch(value) {

case 1: return 'Half Hour'

case 2: return 'One Hour'

case 3: return 'Half Day'

case 4: return 'Full Day'

default : return value.toString();

}

}

}  
  
- html  
{{session.duration | duration}}  
  
\*\*Add Pipe in declaration of module  
  
 **11. Filter and Sort**  
  
export class SessionListComponent implements OnChanges{

@Input() sessions:ISession[];

@Input() filterBy: string;

@Input() sortBy: string;

visibleSessions: ISession[] = [];

ngOnChanges() {

if (this.sessions) {

this.filterSessions(this.filterBy);

this.sortBy === 'name' ? this.visibleSessions.sort

(sortByNameAsc) : this.visibleSessions.sort(sortByVotesDesc);

}

}

filterSessions(filter) {

if(filter === 'all') {

this.visibleSessions = this.sessions.slice(0);

} else {

this.visibleSessions = this.sessions.filter(session => {

return session.level.toLocaleLowerCase() === filter;

})

}

}

}

function sortByNameAsc(s1: ISession, s2:ISession) {

if (s1.name > s2.name) {

return 1;

} else if (s1.name === s2.name) {

return 0;

} else {

return -1;

}

}

function sortByVotesDesc(s1: ISession, s2: ISession) {

return s2.voters.length - s1.voters.length;

}

* HTML -   
    
  <div class="btn-group btn-group-sm" style="margin-right: 20px;margin-left:20px">

<button class="btn btn-default" [class.active]="sortBy==='name'" (click)="sortBy='name'">By Name</button>

<button class="btn btn-default" [class.active]="sortBy==='votes'" (click)="sortBy='votes'">By Votes</button>

</div>

<div class="btn-group btn-group-sm">

<button class="btn btn-default" [class.active]="filterBy==='all'" (click)="filterBy='all'">All</button>

<button class="btn btn-default" [class.active]="filterBy==='beginner'" (click)="filterBy='beginner'">Beginner</button>

<button class="btn btn-default" [class.active]="filterBy==='intermediate'" (click)="filterBy='intermediate'">Intermediate</button>

<button class="btn btn-default" [class.active]="filterBy==='advanced'" (click)="filterBy='advanced'">Advanced</button>

</div>

<session-list [sortBy]="sortBy" [filterBy]="filterBy" \*ngIf="!addMode" [sessions]="event?.sessions"></session-list>

1. **UseClass provider**  
   { provide: EventRouteActivator, useClass: EventRouteActivator }
2. **Search Sessions(example)** searchSessions(searchTerm: string) {

var term = searchTerm.toLocaleLowerCase();

var results: ISession[] = [];

EVENTS.forEach(event => {

var matchingSessions = event.sessions.filter(session => session.name.toLocaleLowerCase().indexOf(term) > -1);

matchingSessions = matchingSessions.map((session:any) => {

session.eventId = event.id;

return session;

})

results = results.concat(matchingSessions);

})

var emitter = new EventEmitter(true);

setTimeout(() => {

emitter.emit(results);

}, 100);

return emitter;

}

1. **Adding jQuery**

* Create file jQuery.service.ts:  
    
  import { InjectionToken } from '@angular/core';

export let JQ\_TOKEN = new InjectionToken<Object>('jQuery');

* Inside module add following:  
    
  import { JQ\_TOKEN } from './common/index';

let jQuery = window['$'];

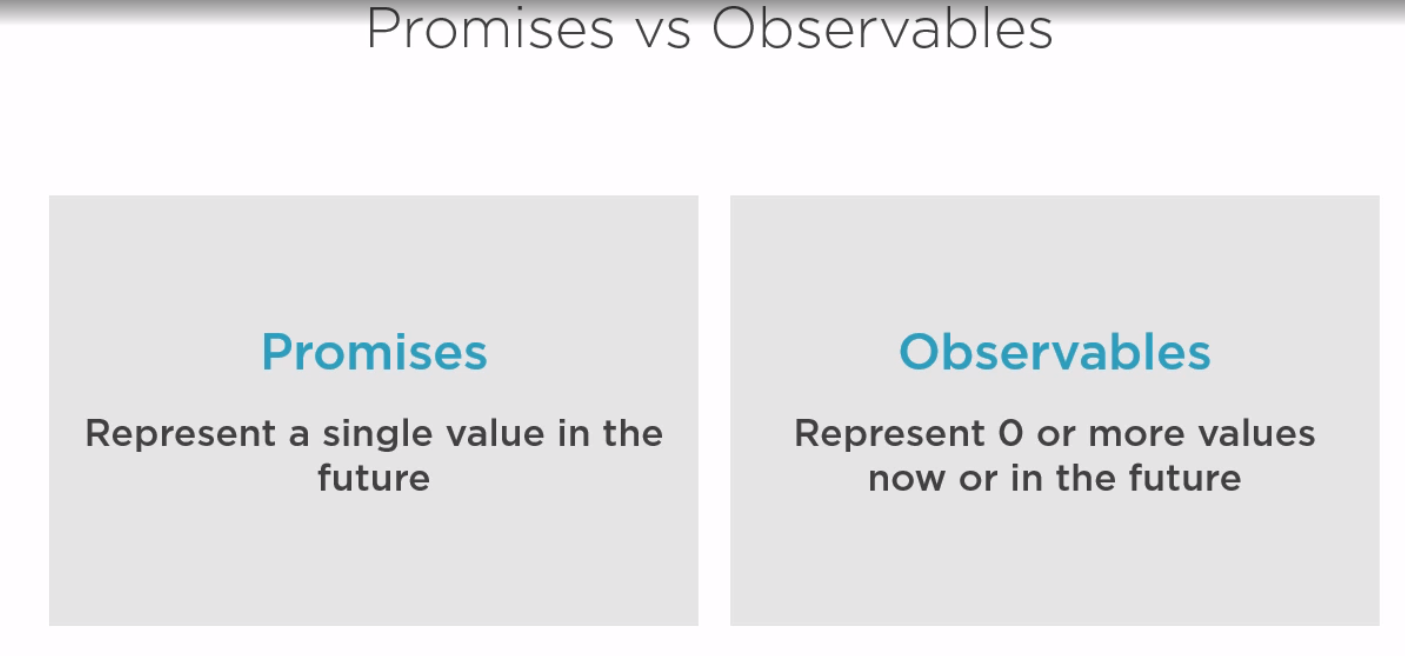
providers: [

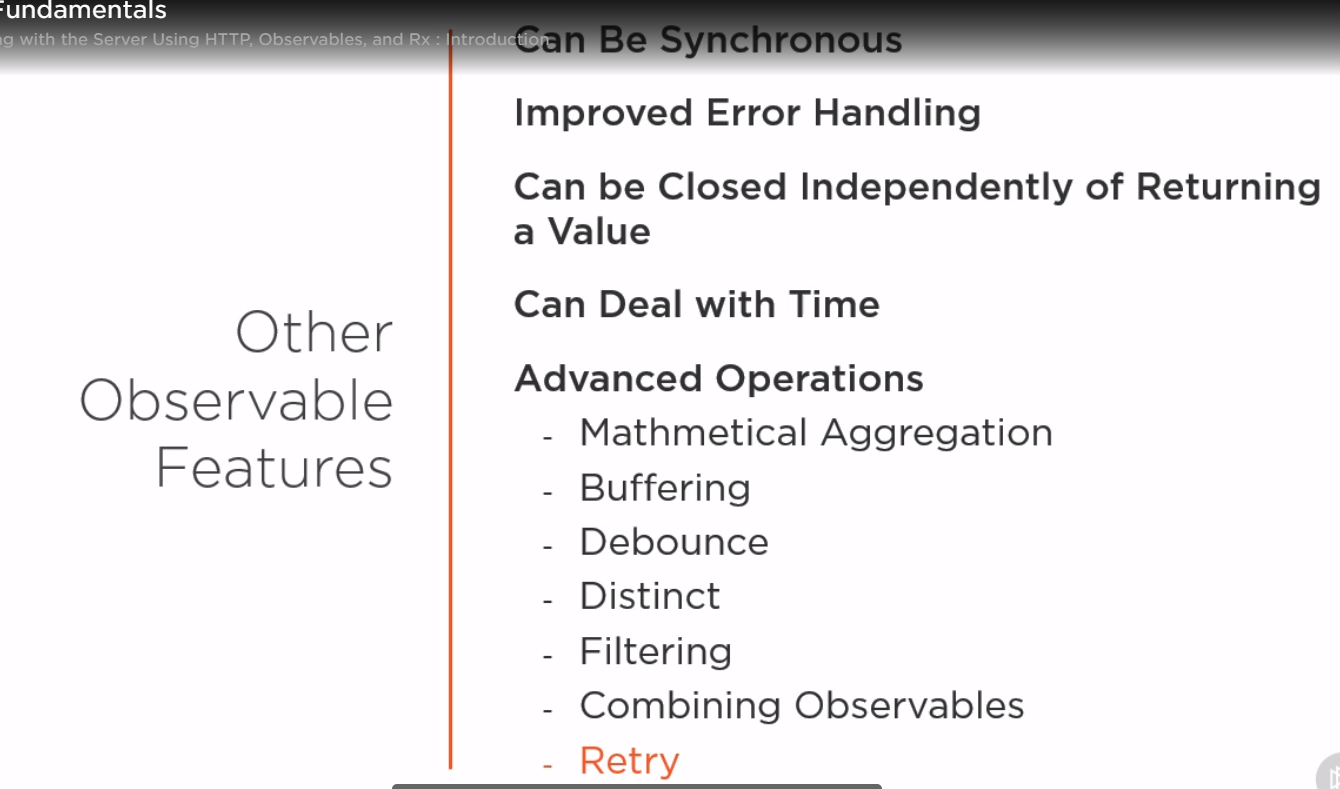
{ provide: JQ\_TOKEN, useValue: jQuery }

...

]

1. **Communicating with the Server using HTTP, Observables and Rx**

****

****

Import { HttpClienModule } from ‘@angular/common/http’;

1. Testing

