Angular 5 Training Course

Exercise K-router

- **Routing** breaks the application into parts, based on rules derived from the browser url.
- In Angular we configure routes by **mapping paths to components**.
- **Routers** help create a single place application (SPA) where the URL in your browser defines the **state** of your application.
- **Client Side Routing**: when you change the URL in a SPA with a router, it does not require an HTTP request to a remote server.

Project setup

- Rebuild the project using **useful/router**.

```
npm install
ng serve --open
```

- **routes/app.routes.ts** defines an array of objects which will map urls to components.
- When the url changes we want a specific component to be rendered on screen.
- Define two new routes for radio and tv.

```
    { path: 'tv', component: TvComponent },
    { path: 'radio', component: RadioComponent}
```

- To enable this array, we pass it to the RouterModule in **app.module.ts**

```
    RouterModule.forRoot( AppRoutes )
```

- If we add a debugging parameter, we can see the router in the browser console if we navigate to the TV or RADIO urls.

```
RouterModule.forRoot( AppRoutes, { enableTracing: true } )
```

- We want changes to the url to cause different components to appear on the web page.

- In the main template app.component.html define where the router should inject content.

```
    <section class="page">
    <router-outlet></router-outlet>
    </section>
```

- Changing url in the browser address bar now injects different components into the main template.

Navigation bar

- We want the user to be able to change url by clicking on a link in the navigation bar.
- The routerLink directive does this. Add this to the main template.

```
  <a [routerLink]="['/tv']">TV</a>
  <a [routerLink]="['/radio']">RADIO</a>
```

- Add the routerActiveLink directive to style the currently selected link.

```
    [routerLinkActive]="['active']"
```

- This will make use of this rule defined in the CSS:

```
    .active{
        color:yellow;
    }
```

TV navigation

- Add navigation to the TV template

```
     [routerLink]="['/tv','4']" [routerLinkActive]="['active']"
      [routerLink]="['/tv','17']" [routerLinkActive]="
['active']"
```

- Clicking on the Channel-4 and Channel-17 buttons causing a runtime error.
- These routes need to be defined.

```
    { path: 'tv/4', component: TvComponent },
    { path: 'tv/17', component: TvComponent },
```

- This works but this approach does not scale well for dozens of channels.
- We want to define the channel number as a variable part of the route and be able to work out that variable in the TVComponent.

```
      { path: 'tv/:channel', component: TvComponent },
```

- ActivatedRoute is an Angular class that creates an Observable to monitor changes of route.

```
    constructor(private route: ActivatedRoute) {
      this.route.params.subscribe(r => console.log( r ));
```

- This will log an object to the browser console.

```
        {channel: "4"}
```

- We can use this to set a channel-number variable.

```
    channel: string;

    this.route.params.subscribe(r => this.setChannel(r))

    setChannel(r) {
  this.channel = r.channel ? r.channel : "";
  }
```

Radio routes

- Uncomment the Navigation Bar routes in the radio template.

```
    <a [routerLink]="['./4']" [routerLinkActive]="
['active']">Radio 4</a>
    <a [routerLink]="['./11']" [routerLinkActive]="
['active']">Radio 11</a>
```

- We need to define these routes.
- We can take a different approach defining **nested routes**.

```
    { path: 'radio', component: RadioComponent,
   children: [
       { path: '4', component: Radio4Component },
       { path: '11', component: Radio11Component }
   ]
 }
```

- We need to define where these nested routes appear in radio.component.html

```
    <router-outlet></router-outlet>
```

Programmatic navigation

- We can define programmatic navigation that is not connected to a link.

```
    constructor( private router: Router ) {}

    goRadio( n ) {
        this.router.navigate(['radio',n ]);
    }

    goChannel( channel ) {
        this.router.navigate( ['tv' , channel ] );
    }
```

Edge cases

- We need to handle the cases where the user provides no route, or where they provide a non-existent route.

```
{ path: '', redirectTo: 'tv', pathMatch: 'full' },
```

{ path: '**', redirectTo: 'tv' }

Router Guards

- **Router Guards** allow the developer to define custom logic which prevents a user from accessing certain routes under certain conditions.
- A Guard is defined as a class with a boolean function called **canActivate**.
- Edit guard/payradio.ts

- This boolean function only returns true if localstorage property payradio is set to 1.

```
canActivate() {
    return Number( localStorage.payradio ) === 1 ;
}
```

- Add it to a route. This route is now only available if this method returns true.

```
{ path: '11', component: Radio11Component, canActivate:[
PayRadio ] }
```