Monday: Error Handling

Error Handling

What would happen if we make a bad request to the API and we get no response, or even worse, the servers are down and not working? We need to prepare our application for such an event so that it is not blank or broken. The subscribe function has an err function that gives us the capacity to handle errors. Let's use it to handle the occurrence of an error in which we get no response from the API.

src/app/goal/goal.component.ts

```
...
export class GoalComponent implements OnInit {
      this.http.get<ApiResponse>("http://quotes.stormconsultancy.co.uk/random.json").subscribe(data
=>{
      // Succesful API request
      this.quote = new Quote(data.author, data.quote)
    },err=>{
        this.quote = new Quote("Winston Churchill","Never never give up!")
        console.log("An error occurred")
    })
...
```

We have added the err function and specified the quote instance that should be created when we get no response and the error message to be logged in the console. Try messing with the URL by in this file by adding or omitting a character, and you'll see the err function handling the response for us.

Using Loaders

Since we are making requests to a remote server, it would be necessary to convince the user that there is something happening in the background, as the app awaits a response. This enhances the user experience of the app. We will do this using a **loader (https://github.com/MurhafSousli/ngx-progressbar/blob/master/README_V3.md#installation)** which is a type of a progress bar. It creates a visual animation in our app that convinces the users that there is something happening. To install this loader module, let's run this command in our terminal:

```
$ npm install --save @ngx-progressbar/core@3.0.2 @ngx-progressbar/http-client@3.0.2
```

This module needs a supplementary module that looks at observable data for the loader to work properly called rxjs-compat. Let's install it using the following command:

```
$ npm install --save rxjs-compat
```

Let's now add it to our root modules of our app to make it available for use.

src/app/app.module.ts

```
...
import { NgProgressModule } from '@ngx-progressbar/core';
import { NgProgressHttpClientModule } from '@ngx-progressbar/http-client';
...
imports: [
    ....
```

```
    NgProgressModule.forRoot(),
    NgProgressHttpClientModule
],
```

We have imported the normal loader and the loader that listens for our HTTP requests from the app and automatically displays progress according to our apps requests. We have then added both to the **imports** array. The forRoot() method makes the loader available and configurable at the root level of our app and supplies the dependencies it needs. We don't need any more configurations since the NgProgressHttpClientModule works with the requests made from our app. We only need to display the loader in our template.

src/app/goal/goal.component.html

```
<div class="container">
  <ng-progress></ng-progress>
  <h1>MY GOALS</h1>
.....
```

We have added tags in the template placed at the top where we want to see our progress bar. When we run our server and go to our app in the browser, we see the loader appear every time we refresh the page showing the progress of every new request.