1DA1611:A
Advanced Internet Programming

Project nr. 9 Report: 2023-04-20

Illia Priadko (309062)

Plan:

- Analyze AngularJS and find a suitable use case for my project;
- Implement menu population from a json via AngularJS
- Testing and troubleshooting

Given task:

> As for the previous technologies, please try to get the maximum knowledge about AngularJS technology. **Present 4-5 examples of AngularJS applications in** different fields and explain them in order to show me that *you managed perfectly the technology*.
>
> When you write your reports, **do not just repeat what is in the description of the scripts**. Most of the elements are quite easily understandable. However, there are some tips that should be explained more clearly. There are things **you should explain how they have been adapted to your web page design, especially from the viewpoint of page responsiveness.**

1. AngularJS analysis

AngularJS is a JavaScript framework developed by Google for building dynamic, single-page web applications. AngularJS provides a robust set of features, which allow developers to create highly interactive and responsive web applications. It has been widely adopted by developers worldwide and continues to evolve as a powerful tool for front-end web development.

Some AngularJS examples (from w3schools):

Example 1:

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.
6.9/angular.min.js"></script>
<body>

<p>Try to change the names.</p>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName">
<br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "John";
    $scope.lastName= "Doe";
});
</script>

</body>
</html>
```

Try to change the names.

First Name: John
Last Name: Doe

Full Name: John Doe

Code begins by including the AngularJS library as a .js file. It creates an HTML page with a <p> and a tags. The div tag is given an ng-app directive with the value "myApp", which specifies the name of the AngularJS module that will be used to control this section of the HTML. The div tag also has an ng-controller directive with the value "myCtrl", which specifies the name of the controller function that will manage the data and behavior of this section. These two parts will also be of great importance later, once I get to making my own implementation of AngularJS code in the project.

Inside the div tag, there are two input fields with ng-model directives, which bind the input values to the corresponding properties inside the controller. The Full Name section concats two strings to display the full name (firstName and lastName together).

Finally, the code defines an AngularJS module with the name "myApp" and an AngularJS controller with the name "myCtrl". The controller initializes the firstName and lastName properties of the $scope object, which are used to set the initial values of the input fields.

Example 2:

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1
.6.9/angular.min.js"></script>
<style>
input.ng-invalid {
    background-color: lightblue;
}
</style>
<body>

<form ng-app="" name="myForm">
    Enter your name:
    <input name="myName" ng-model="myText" required>
</form>

<p>Edit the text field and it will get/lose classes
according to the status.</p>
<p><b>Note:</b> A text field with the "required"
attribute is not valid when it is empty.</p>

</body>
</html>
```

Enter your name:

Edit the text field and it will get/lose classes according to the status.

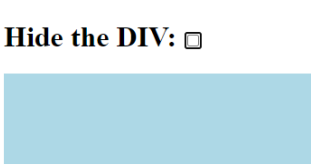
Note: A text field with the "required" attribute is not valid when it is empty.

This code snippet is used to validate the text input field, making it highlighted when it is empty. Useful in form applications, to let the user not leave required inputs empty.

Inside the form, there is an input field with the ng-model directive set to "myText". This directive binds the input field value to the $scope.myText variable in the AngularJS code. The input field also has the required attribute, which tells the browser to require a value to be entered before the form can be submitted. The code also includes CSS that styles the input field with a light blue background color when it is invalid (i.e., when it is empty) and removes the style when it is valid (i.e., when a value is entered).

Example 3:

```
<!DOCTYPE html>
<html>
<style>
div {
  transition: all linear 0.5s;
  background-color: lightblue;
  height: 100px;
  width: 100%;
  position: relative;
  top: 0;
  left: 0;
}

.ng-hide {
  height: 0;
  width: 0;
  background-color: transparent;
  top:-200px;
  left: 200px;
}

</style>
<script
src="https://ajax.googleapis.com/ajax/libs/angula
rjs/1.6.9/angular.min.js"></script>
<script
src="https://ajax.googleapis.com/ajax/libs/angula
rjs/1.6.9/angular-animate.js"></script>

<body ng-app="ngAnimate">

<h1>Hide the DIV: <input type="checkbox" ng-
model="myCheck"></h1>

<div ng-hide="myCheck"></div>

</body>
</html>
```



This example demonstrated animation features that are included with AngularJS’s animation modules. Both are included in the <script> sections.

The div element has a height of 100px, a width of 100%, and a background color of light blue. The ng-hide directive is used to hide the div element when the value of the myCheck variable is true. So the CSS code snipped included in the <style> section includes a transition property that specifies the duration of the animation. The .ng-hide includes properties as “destination”, i.e. what the position/colour/etc. should be, when the checkbox is checked. As a result, the div element is hidden or shown with a transition animation, by adding and removing a specific tag.

Example 4 is the project below.

2. **Implement menu population from a JSON via AngularJS**

As seen in Project 8, the menu essentially was a JSON list, based on which the header in the webpage would be populated with buttons that took the user to the respective Project (1-4), implemented via jQuery.

In this project (9), it was decided to replace the same function, however, to make it more compact and realized through use of AngularJS.

So the end goal, to reiterate, is to implement fetching buttons from a JSON file.

Here is the projects.json file that buttons will be built from:

```
[
    {
        "id": 1,
        "label": "Project 1",
        "date": "02/23/2023"
    },
    {
        "id": 2,
        "label": "Project 2",
        "date": "03/02/2023"
    },
    {
        "id": 3,
        "label": "Project 3",
        "date": "03/09/2023"
    },
    {
        "id": 4,
        "label": "Project 4",
        "date": "03/16/2023"
    }
]
```

As for the .HTML, the code changes are as follows:

```
                <div class="btn-group" role="group" id="btn-group-header" ng-app="myApp" ng-controller="myController">
                    <button type="button" ng-repeat="button in buttons" ng-click="buttonClicked(button)"
                        class="btn btn-primary">
                        <!-- Expression with button name and date -->
                        {{button.label+": "+button.date}}
                    </button>
```

Ng-app and ng-controller parts are now here, to correspond to the Angular module and controller defined below in the script. Ng-repeat is also here, to render as many buttons as there are sets in the .json file. Ng-click corresponds to the called function upon clicking on the button, also defined in the code below. Expression in {{}} refers to labels and date info of the button set.

.JS:

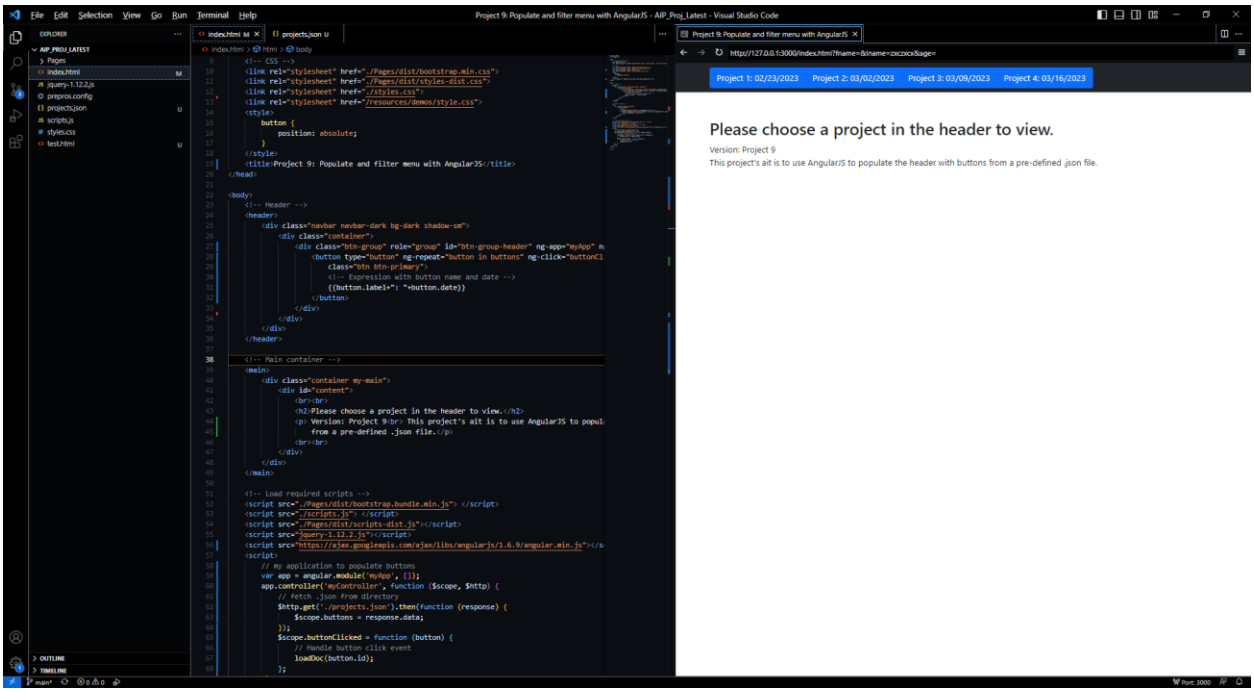
```
        // my application to populate buttons
        var app = angular.module('myApp', []);
        app.controller('myController', function ($scope, $http) {
            // fetch .json from directory
            $http.get('./projects.json').then(function (response) {
                $scope.buttons = response.data;
            });
            $scope.buttonClicked = function (button) {
                // Handle button click event
                loadDoc(button.id);
            };
        });
```

Comments left here are for explaining the different parts of the code. AngularJS here uses HTTP GET function to fetch data from the projects.json file, upon which it constructs buttons are defined in the .HTML part. As the button is clicked, it calls a function to load the document and replace the contents with the specified project. The projects are chosen based on the “id” parameter.

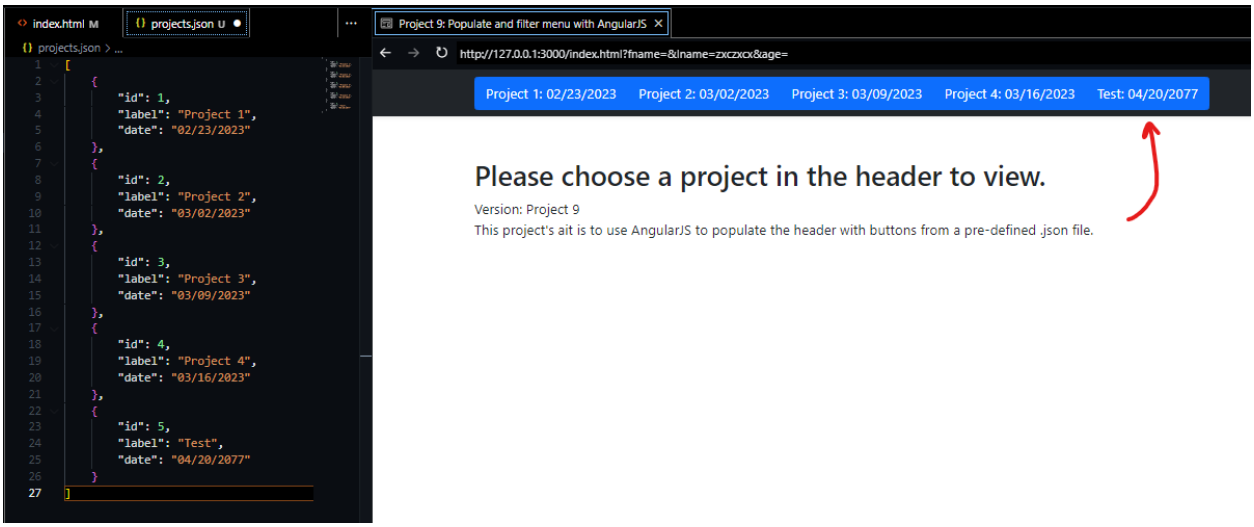
Note: comparing to the jQuery implementation, this is much more compact.

3. Testing and troubleshooting

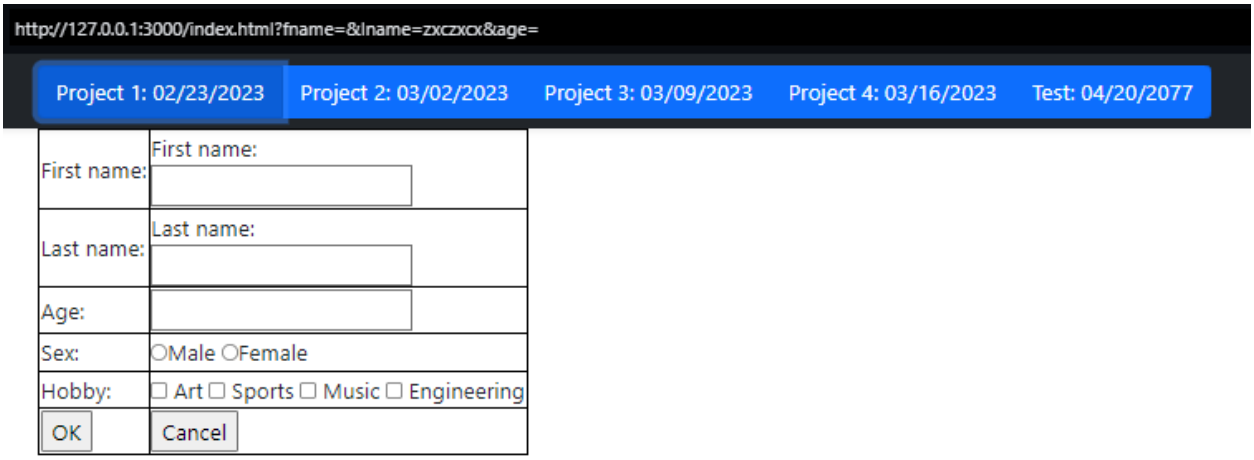
The page was tested, this time, by using VSCode’s Preview extension, which essentially is a local webserver, that allows to preview the page in real time, as it is being changed.



To prove that the button list is dynamic, I have added a new position in the .json file, named “Test”, which is to be located alongside actual buttons.



If there was a webpage defined for “id”: 5, it would also load that, similar to the rest of buttons:



The final webpage is available at https://illiapriadko.github.io/AIP_Proj_Latest/

As well as the source code repository: https://github.com/illiapriadko/AIP_Proj_Latest

Conclusion:

After having made this project, I am now familiarized with AngularJS, its use cases, as well as have implemented its functionalities to improve and minimize my webpage’s code. It is a very versatile framework that it widely used in web development, so being familiar with it is of great help.