Advanced internet programming

Project no. 9 – Angular JS – The basics

Hubert Bęczkowski, ID: 307290

What is angular?

Angular is a development platform, built on TypeScript. As a platform, Angular includes:

- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code

With Angular, you're taking advantage of a platform that can scale from single-developer projects to enterprise-level applications. Angular is designed to make updating as straightforward as possible, so take advantage of the latest developments with minimal effort.

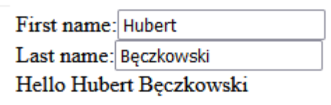
Main idea behind angular approach is to create an application out of building blocks, which we call components. A component includes a TypeScript class with a @Component() decorator, an HTML template, and styles. The component decorator specifies an Angular-specific information:

- A CSS selector that defines how the component is used in a template. HTML elements in your template that match this selector become instances of the component.
- An HTML template that instructs Angular how to render the component
- An optional set of CSS styles that define the appearance of the template's HTML elements

In this report, I am going to take a look at 4 example angular applications, which will allow me to show my newly obtained expertise on the topic.

Example 1. Dynamic updating of first and last name

```
<html ng-app>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.1/angular.min.js"></script>
  </head>
  <body>
    First name:<input ng-model="firstName" type="text"/>
    <br>
    Last name:<input ng-model="lastName" type="text"/>
    <br>
    Hello {{firstName}} {{lastName}}
  </body>
</html>
```



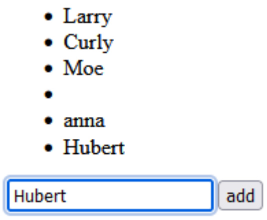
First name: Hubert
Last name: Bęczkowski
Hello Hubert Bęczkowski

The example shows how an ajax-angular script invoked from a library can create a very simple dynamically updating / responsive table. The head part of the code consists of the script that invokes a JS out of a library from cloud. In body, we create an input ng-model which asks a user to input their first and last name. Then, the input appears in output through it being classified in previous part and invoked at the end of the body part.

Example 2. Adding entries to a list

```
<html ng-app="nameApp">
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.1/angular.min.js"></script>
    <script>
      var nameApp = angular.module('nameApp', []);
      nameApp.controller('NameCtrl', function ($scope){
        $scope.names = ['Larry', 'Curly', 'Moe'];

        $scope.addName = function() {
          $scope.names.push($scope.enteredName);
        };
      });
    </script>
  </head>
  <body ng-controller="NameCtrl">
    <ul>
      <li ng-repeat="name in names">{{name}}</li>
    </ul>
    <form ng-submit="addName()">
      <input type="text" ng-model="enteredName">
      <input type="submit" value="add">
    </form>
  </body>
</html>
```



- Larry
- Curly
- Moe
-
- anna
- Hubert

Hubert

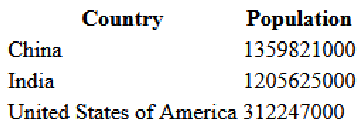
Presented code creates an interactive list that can have entries added to itself. In head, the bullet-point list is being created, with some predetermined names already in.

The script allows addition of names to the list, which can be useful in order to expand one in a quick manner. Although the idea is well done, a major issue is lack of possibility to remove names. It can be done with addition of a removeName option in script, and a clickable option in body.

```
 $scope.removeName = function(name) {
   var i = $scope.names.indexOf(name);
   $scope.names.splice(i, 1);
<ul>
  <li ng-repeat="name in names">{{name}}
    <a href="" ng-click="removeName(name)">x</a>
  </li>
</ul>
```

Example 3. Building a table

```
<html ng-app="countryApp">
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.1/angular.min.js"></script>
    <script>
      var countryApp = angular.module('countryApp', []);
      countryApp.controller('CountryCtrl', function ($scope){
        $scope.countries = [
          {"name": "China", "population": 1359821000},
          {"name": "India", "population": 1205625000},
          {"name": "United States of America","population": 312247000}
        ];
      });
    </script>
  </head>
  <body ng-controller="CountryCtrl">
    <table>
      <tr>
        <th>Country</th>
        <th>Population</th>
      </tr>
      <tr ng-repeat="country in countries">
        <td>{{country.name}}</td>
        <td>{{country.population}}</td>
      </tr>
    </table>
  </body>
</html>
```

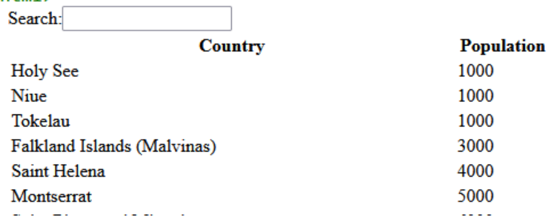


```
                    Country         Population
China                               1359821000
India                               1205625000
United States of America 312247000
```

Similarly to other examples, the table is having its values input in script while tuned through the invoked script. The body manages how the table is built, and invokes the values in other rows. By itself, it is a simple way to build tables, but without additional tools its quite lackluster, which we will expand upon in next example.

Example 4. Adding additional features to the table

```
<html ng-app="countryApp">
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.1/angular.min.js"></script>
    <script>
      var countryApp = angular.module('countryApp', []);
      countryApp.controller('CountryCtrl', ['$scope', '$http', function (scope, http){
        http.get('countries.json').success(function(data) {
          scope.countries = data;
        });
      }]);
    </script>
  </head>
  <body ng-controller="CountryCtrl">
    Search:<input ng-model="query" type="text"/>
    <table>
      <tr>
        <th>Country</th>
        <th>Population</th>
      </tr>
      <tr ng-repeat="country in countries | filter:query | orderBy:'population'">
        <td>{{country.name}}</td>
        <td>{{country.population}}</td>
      </tr>
    </table>
  </body>
</html>
```



```
Search:
                   Country                      Population
Holy See                                        1000
Niue                                            1000
Tokelau                                         1000
Falkland Islands (Malvinas)                     3000
Saint Helena                                    4000
Montserrat                                      5000
```

In this example, the table has been expanded upon by placing the table data in a Json file, which is invoked in script, from where it can be invoked easily.

The countries are being sorted in an ascending order of population value. The addition of searching option is done by adding a Search with input of ng-model, which seeks the data through query, which type is “text”.

By adding those features to our table, it creates an interactive and responsive tool, which allows for quick search of data. The only issue would be implementing a dynamic updating of data stored in json file, although I haven’t managed to add such feature to it.

Overall, angular is a great way to create interactive and responsive tools for the html pages, which can be easily and quickly updated, or have added desirable features.