Report lab n.9

AngularJS is a JavaScript-based framework used for building dynamic web applications. It uses the MVC architecture and provides features like two-way data binding, dependency injection, directives, filters, and services.

In order to add AngularJS to any code it is necessary to import it:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular
.min.js"></script>
```

Example-1 ($http service)

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

    <p>Today's welcome message is:</p>

    <h1>{{myWelcome}}</h1>

    </div>

    <p>The $http service requests a page on the server, and the response is set
as the value of the "myWelcome" variable.</p>

    <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function($scope, $http) {
      $http.get("welcome.htm")
      .then(function(response) {
          $scope.myWelcome = response.data;
      });
    });
    </script>
```

This is a basic AngularJS application that uses the $http service to request data from a server and update the view. The controller function makes an HTTP GET request to the server, and when the response is received, it updates the $scope.myWelcome variable, which is displayed in an h1 element in the view using the {{myWelcome}} expression.

I can easly use this kind of approach in my web project like this:

From:

```
function ajaxConnect(setting) {
```

```
        $.ajax({
          url: setting.contentsource[1],
          cache: !this.bustajaxcache,
          success: function(response) {
            $('#' + setting.id).html(response);
            featuredcontentslider.buildpaginate(setting);
          }
        });
      }
```

To:

```
app.controller('myCtrl', function($scope, $http) {
    $scope.ajaxConnect = function(setting) {
      $http({
        method: 'GET',
        url: setting.contentsource[1],
        cache: !this.bustajaxcache,
      }).then(function(response) {
        $('#' + setting.id).html(response.data);
        featuredcontentslider.buildpaginate(setting);
      });
    };
```

Example-2(Animation)

In order to make animation we have also to import:

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-animate.js"></script>

```
<body ng-app="myApp">

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

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

  <script>
  var app = angular.module('myApp', ['ngAnimate']);
  </script>

  </body>
```

This is an AngularJS application that demonstrates how to use the ng-hide directive to hide an HTML element based on a Boolean value that is bound to an input element using the ng-model directive.

AngularJS provides several animation directives such as ng-show, ng-hide, ng-repeat, ng-class, ng-view and ng-include which can be used to animate the hiding/showing, adding/removing of elements, applying CSS classes, and changing of views in web applications.

Example-3(Filters)

```
<div ng-app="myApp" ng-controller="namesCtrl">
<p>Type a letter in the input field:</p>
<p><input type="text" ng-model="test"></p>
<ul>
  <li ng-repeat="x in names | filter:test">
    {{ x }}
  </li>
</ul>
</div>
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        'Jani',
        'Carl',
        'Margareth',
        'Hege',
        'Joe',
        'Gustav',
        'Birgit',
        'Mary',
        'Kai'
    ];
});
</script>
<p>The list will only consists of names matching the filter.</p>
```

This code defines an AngularJS application with a controller called namesCtrl. The controller sets up a list of names, and the view uses the ng-repeat directive to loop over the names and display them in a list. The ng-model directive is used to bind the input field to the $scope.test variable, which is used to filter the list of names based on the user's input. As the user types into the input field, the list of names displayed will only show those that match the current value of $scope.test.

In AngularJS, there are several built-in filters that can be used to format or transform data displayed in the view. Some of the most commonly used filters include:

uppercase: Converts a string to uppercase.

lowercase: Converts a string to lowercase.

currency: Formats a number as a currency string.

date: Formats a date object as a string.

json: Formats an object as a JSON string.

limitTo: Limits an array or string to a specified number of items or characters.

orderBy: Sorts an array by a specified property.

Filters can be chained together to create more complex transformations. For example, the currency filter can be combined with the uppercase filter to display a currency amount in all caps. Additionally, custom filters can be created using the filter method of a module, allowing for even more flexibility in data transformation.

Example-4(FORMS)

```
<div ng-app="myApp" ng-controller="formCtrl">
  <form novalidate>
    First Name:<br>
    <input type="text" ng-model="user.firstName"><br>
    Last Name:<br>
    <input type="text" ng-model="user.lastName">
    <br><br>
    <button ng-click="reset()">RESET</button>
  </form>
  <p>form = {{user}}</p>
  <p>master = {{master}}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.master = {firstName:"John", lastName:"Doe"};
    $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
    };
    $scope.reset();
});
</script>
```

This code shows an AngularJS application with a form containing two input fields for the user's first and last name. The form has a "RESET" button that calls the "reset" function defined in the controller. The controller sets the default values for the first and last name in the "master" object and copies them to the "user" object when the "reset" function is called. The "ng-model" directive binds the input fields to the "user" object. The "novalidate" attribute is used to disable browser validation.