/

AngularJS Tutorial

By Jakob Jenkov

Table of Contents

AngularJS is a JavaScript framework that makes it easier to implement RIA web applications. AngularJS is created by Google. You can find the AngularJS project here:

http://angularjs.org/

AngularJS is based on the MVC pattern (Model View Control). Therefore AngularJS separates your RIA application into models, views and controllers. The views are specified using HTML + AngularJS's own template language. The models and controllers are specified via JavaScript objects and JavaScript functions. Thus, the views are specified declaratively, as HTML normally is, and the models and controllers are specified imperatively, as JavaScript normally is.

If you don't know the difference between declarative and imperative programming, don't worry. It is not important to know before learning AngularJS. Besides, it is pretty simple to find the definition on the web.

AngularJS Hello World

Here is a simple "hello world" example made with AngularJS which shows the model, view and controller parts of an AngularJS app:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js
        </script>
    </head>

    <body ng-app="myapp">
        <div ng-controller="HelloController" >
            <h2>Hello {{helloTo.title}} !</h2>
        </div>

        <script>
            angular.module("myapp", [])
                .controller("HelloController", function($scope) {
                    $scope.helloTo = {};
                    $scope.helloTo.title = "World, AngularJS";
                } );
        </script>

    </body>
</html>
```

In this example the "view" is this part:

```
<div ng-controller="HelloController" >

    <h2>Hello {{helloTo.title}} !</h2>

</div>
```

Notice the ng-controller attribute. This attribute tells AngularJS what controller to use with this view. Notice also the {{helloTo.title}} text. This is part of AngularJS's template system. This tells AngularJS to write the "model" value named helloTo.title to the HTML at this location.

The "controller" is this part:

```
<script>
    angular.module("myapp", [])
        .controller("HelloController", function($scope) {
            $scope.helloTo = {};
            $scope.helloTo.title = "World, AngularJS";
        });
</script>
```

This code registers a controller function named `"HelloController"` in the angular module named `"myapp"`. Angular modules will be explained later in this tutorial.

The `$scope` parameter passed to the controller function is the `"model"`. The controller function adds a `helloTo` JavaScript object, and in that object it adds a `title` field. It is this `helloTo.title` value from the model which the view writes into the HTML.

Developing an AngularJS Application From Scratch

Now that you have seen the model, view and controller parts of an AngularJS application, let us build the above example application from scratch with explanations for each step.

AngularJS applications are a mix of HTML and JavaScript (as you have already seen) so the first thing we need is an HTML page:

```
<!DOCTYPE html>
<html>
    <head>
    </head>

    <body>
    </body>
</html>
```

Second, we need to include the AngularJS JavaScript file in the HTML page so we can use AngularJS:

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js
        </script>
    </head>

    <body>
    </body>
</html>
```

Remember to check AngularJS's website for the latest version of AngularJS so you do not keep using the version I use in this example.

Third, we need to tell AngularJS what part of our HTML page contains the AngularJS app. You do so by adding the `ng-app` attribute to the root HTML element of the AngularJS app. Typically, the root element is either the `html` element or the `body` element. In this example I insert the `ng-app` attribute into the `body` element:

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js
        </script>
    </head>

    <body ng-app="myapp">
    </body>
</html>
```

In this example I set the `ng-app` value to `myapp`. This means that all controller functions must be added to the `myapp` module. In other words, the AngularJS module named `myapp` contains all the controllers for this AngularJS application. The name `myapp` is something I have chosen. You can choose your module names freely.

Fourth, we need a view. A `view` is a section of HTML which is enclosed in an HTML element. Here is an example:

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js
        </script>
    </head>

    <body ng-app="myapp">
       <div ng-controller="HelloController">

            <h2>Hello {{helloTo.title}} !</h2>
```

```
        </div>
    </body>
</html>
```

In this example the view is the `div` element and everything inside it. The `div` element contains `ng-controller` attribute with the value `HelloController`. This means, that the controller function used by this view is named `HelloController`.

The view `div` element contains the HTML

```
<h2>Hello {{helloTo.title}} !</h2>
```

This is standard HTML except for the `{{helloTo.title}}` part. This part tells AngularJS to insert the model value named `helloTo.title` into the HTML at that location.

Fifth, we need a controller function. A controller function is a normal JavaScript function which takes a single parameter: The `$scope` parameter. The `$scope` parameter is the model object to be used by the controller function and the corresponding view. The controller function can insert data and functions into the model object. The view can then use the data and the functions.

Here is our little web app with the controller function inserted:

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js
        </script>
    </head>

    <body ng-app="myapp">
        <div ng-controller="HelloController" >
            <h2>Hello {{helloTo.title}} !</h2>
        </div>

        <script>
            angular.module("myapp", [])
            .controller("HelloController", function($scope) {
            $scope.helloTo = {};
            $scope.helloTo.title = "World, AngularJS";
            });
        </script>
    </body>
</html>
```

The controller function is registered in angular via the `angular.module(...).controller(...)` function call.

The `angular` object is a global object created by the AngularJS JavaScript which is included at the beginning of the page.

Notice the first of the parameters to the `angular.module()` function call. The value of that parameter is `"myapp"`. This name matches the name specified in the `ng-app` attribute in the `body` element. This way AngularJS knows that the controller function being registered belongs to the `myapp` module, which happens to be the module which this AngularJS application is using (specified in the `ng-app` attribute in the `body` element).

The `controller()` function call is what registers the controller function itself. The first parameter passed to the `controller()` function is the name of the controller function. This is the name you refer to in the `ng-controller` attribute of the view. The second parameter is the controller function itself.

AngularJS Application Execution

Now our example application is complete, so let me explain what happens when this page is loaded into the browser.

First the HTML document is loaded into the browser, and evaluated by the browser. At this time the AngularJS JavaScript file is loaded, the `angular` global object is created, and your JavaScript which registers controller functions is executed.

Second, AngularJS scans through the HTML to look for AngularJS apps and views. When AngularJS finds a view it connects that view to the corresponding controller function.

Third, AngularJS executes the controller functions and update (render) the views with data from the model populated by the controller. The page is now ready.

Fourth, AngularJS listens for browser events (e.g. input fields being changed, buttons clicked, the mouse moved etc.). If any of these browser events require a view to change, AngularJS will update the view correspondingly. If the event requires that the corresponding controller function is called, AngularJS will do that too. Not all events require the controller function to be called, even if the view is updated as a result of the event.

That is it. You are now ready to start playing around with your own AngularJS applications. There is a lot more to learn, but that is covered in subsequent texts in this tutorial (see top left corner of each page in this tutorial series for a list of topics in the series).