# Learn Angular

BY MAKING A QUIZ APP FROM SCRATCH

HUNGRYTURTLECODE.COM

## Part 1

## Course Index

# I've Learnt Some JavaScript! What Now?

That may be something that pretty much all of us have said at some point or another. You have learnt some basic programming syntax, but you now want to build something. In comes this AngularJS Project.

A great way to continue your learning with JavaScript is to learn a framework. So in this course, we will be doing just that. The framework of choice this time is AngularJS.

AngularJS is a framework that has been built by Google and is very widely used for web projects for small companies and huge ones alike.

## Learn AngularJS by doing

By far the best way to learn anything is to just dive in and do it. So that is exactly what I am going to urge you to do.

This article and the below video are part one of a whole course that has been designed with beginners in mind. So it does not include a lot of heavy theory and ideas.

Instead it tries to give you the minimum effective dose for Angular. Taking your existing knowledge of Javascript and building on top of that with some new Angular code.

This results in some of the code not being perfect "Angular". So if you have used Angular before you may want to tell me off, but it is done like this intentionally to help those beginners.

# What Will We Be Building in This AngularJS Project?

Take a look at the video below (which is also a video version of this written tutorial) and you will see the application in action.

Or you can see the finished working application here.

The application itself is a quiz application with a little learning “Facts” area for users to swot up before they take the quiz. The user will then get their results when they are finished.

Check out the git repo for this project

## Let’s Get Going!


Click the image to go to video on youtube, or go to https://www.youtube.com/watch?v=yordL7Yczes&index=1&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq

So for those of you that prefer to read the tutorial. Let’s get started.

## The Code To Get Started With

This is the HTML markup that we will need to get going with the project.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Turtle Facts and Quiz</title>
    <!-- Bootstrap css and my own css -->
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
        integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
        crossorigin="anonymous">
    <link rel="stylesheet" href="css/style.css">
</head>
<body>

    <div class="container">
        <div class="page-header">
            <h1>Turtle Facts Quiz</h1>
            <h3>
                Learn about all the turtles below before you decide to take on the
                <strong>TURTLE QUIZ</strong>
            </h3>
        </div>

    </div>

    <!-- third party js -->
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0-rc.2/angular.min.js"></script>
    <script src="https://code.jquery.com/jquery-2.2.0.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-
    0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS" crossorigin="anonymous"></script>

    <!-- Our application scripts -->

</body>
</html>
```

It is simply a bootstrap layout with jQuery and Angular added in. I have added some basic markup at the top of the container which will be present throughout the whole application.

# Let Yourself Watch Your Code LIVE!

If you have been writing HTML for any length of time you will know how annoying it can be to write some code, then have to go into the browser to refresh.

So before we get really stuck into this project I am going to make a suggestion. Live-server.

## What is Live-Server?

Live-server is a tool that will allow you to automatically refresh your browser every time you save any of the files in your projects directory. Yes, it's awesome!

Of course, if you already have a way of live reloading your browser then skip to the code below. Any live reloading will work just fine.

Live-server is a node package that can be installed via NPM. If you do not have node installed head over to nodejs.org and install it on your machine.

Once you have node installed you can run the following command from cmd or terminal:

```
npm install -g live-server
```

That will install live server globally on your machine. Once that is done navigate to your project's directory in terminal and run the following line:

```
live-server
```

Yep, it’s that simple. Live-server will now scan your files for changes and when it sees one it will refresh your browser so you don’t have to.

## Your First Bit Of Angular

There is one bit of code that every single Angular application needs, and that is the ng-app directive.

In Angular, a directive is pretty much just a bit of code that tells Angular it needs to do something. Of course it is a bit more complex than that in reality. But that is enough for you to understand right now.

What the ng-app directive does, is tell Angular which parts of the HTML it is in control of. In our case we want it to control our entire page, so we will add it onto our HTML tag.

```
<html lang="en" ng-app="turtleFacts">
```

As you can see, it looks very much like a normal HTML attribute – that’s because it is. Inside the quotes for the ng-app directive, we give our application a name.

The name we give it here is important because that is the name we will be referencing when we start writing the Javascript to control our application.

# turtleFacts Angular Module

To create the Javascript to control the page, we first need to reference it in our HTML. The first bit of Javascript we are going to write is to define the module – which is effectively our application.

```
    <!-- Our application scripts -->
    <script src="js/app.js"></script>

</body>
</html>
```

We will add it into a file called app.js inside our js/ folder.

Then we will hop into our app.js folder and start creating the module.

We will start the javascript by creating an Immediately Invoking Function Expression (IIFE)

```
(function(){

    // Angular Code

})();
```

This serves to encapsulate our code. In other words, it keeps our code separate from any other code running on the page and keeps everything clean.

It is ultimately just a function that will run automatically when the page is run. Inside it we will put all of the code we write.

## Defining the Module

To define the module we need to grab hold of the Angular object then call .module and pass into it the name of the module and any dependencies.

```
(function(){

    /*
     * Declaration of main angular module for this application.
     *
     * It is named turtleFacts and has no dependencies (hence the
     * empty array as the second argument)
     */
    angular
        .module("turtleFacts", [ ]);

})();
```

I have put the .module call on a separate line from the angular call but you do not have to. I have just done it this way as it is the convention for Angular.

Notice that despite having no dependencies, we still have to pass an empty array as the second argument.

This is something that tripped me up when I first started. I assumed as we have no dependencies, we could just leave it blank. This is not the case, though.

In Angular, if we call .module and only pass in a name without the second argument, we are calling that module and asking for it to be returned to us. However, in this instance, we want to create the module.

This is why we need the second argument. It let’s Angular know we are creating a module and not calling one.

# AngularJS Controller – Another Directive

Now that Angular knows we are creating a module and we have told it what part of the HTML that module controls we can now start to create the code that does the controlling on behalf of the module – a controller.

In most Angular applications, you will have many controllers. Each in control of a small part of the HTML. This serves a few purposes. It keeps our code clean and easier to understand and it also makes it easier to test.

Our first controller will be in charge of the list of turtles that we saw in the app demo.

So let's create a div and give it the ng-controller directive then tell it which controller we should use to control that particular div.

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

</div>
```

We have called the controller listCtrl in this case. So now let's create another JS file to hold this controller.

We will put the file in directory called controllers/ that will live in our js folder and we will call the file list.js

So our custom scripts area looks like this so far:

```
<!-- Our application scripts -->
   <script src="js/app.js"></script>
   <script src="js/controllers/list.js"></script>

</body>
</html>
```

## Why Put The Controller Into A New File?

When we write Angular, we like to separate our concerns. By that I mean we have very defined bits of code that do one thing and one thing only.

A good practice to get into that will help with this separation of concerns is to put all code that does something different into a new file.

A nice by-product of this is that the files will be short and thus easier to read and ultimately easier to maintain.

## Our Controller Code

Start off the list.js file with the same IIFE that we have seen before. Then inside that we need to call our turtleFacts module.

We do that in the same way as we created the module, we just leave off the second argument to .module.

Once we have the module returned to us we can chain on other methods. This time we will chain on the .controller method.

The controller method takes two arguments, the name of the controller and the function that holds the code for the controller.

```
(function(){

    angular
        .module("turtleFacts")
        .controller("listCtrl", ListController);

    function ListController(){
        // List Controller Logic
    }

})();
```

In this case I have decided to use a named function as the second argument to .controller then define the function explicitly below.

### Another Way – Although Not My Favourite

Although it is pretty common to use an inline anonymous function as the second argument like this:

```
(function(){

    angular
        .module("turtleFacts")
        .controller("listCtrl", function(){
            // List Controller Logic
        });
})();
```

This method is perfectly valid. Feel free to use it. However, I do not like doing it this way because it can make the code slightly harder to read.

I like to be as explicit and clean as possible. So I find directly naming the function and explicitly declaring it is more natural for me. But you do which ever way you prefer.

## The Famous {{}} Angular Syntax

The first thing I ever saw of Angular was a line of code in an HTML file that used the {{}} syntax. This is used to tell Angular we are using some sort of expression and not a literal string as it would usually be in an HTML document.

So something like `{{2+4}}` will be evaluated as an expression by angular and the result will be the number 6 printed out to the screen.

When inside our controller HTML we can take advantage of this {{}} syntax to bind on to properties that we define in our controller.

The great thing about this is that if we then change that property on our controller at some point during the lifetime of the application the value on the screen will automatically update without us having to do anything. This is part of the beauty of Angular data binding.

In part 2 of this series we will take a look at how we can create a property on our controller and utilise that inside our HTML using this new syntax.

See you there

Adrian