Learn Angular

BY MAKING A QUIZ APP FROM SCRATCH

HUNGRYTURTLECODE.COM

Part 3

Check Out The Whole Course Index

1. Getting Started
2. Ng-controller directive and the (mis)use of $scope
3. You are here
4. Markup for the bootstrap modal
5. Using Angular Filters to create real time search
6. The powerful ng-click directive
7. Services in Angular Make everything easier
8. What is this infamous dependency injection in Angular?
9. Let’s Build A Factory
10. The ng-class directive
11. More Bootstrap Markup – The Well
12. Adding some logic to the controller
13. Making things disappear with ng-if
14. The $index property for ng-repeat
15. Reusing code is always a good idea
16. Using Bootstrap to help with styling error messages
17. The final prompt after the quiz
18. Marking the quiz
19. More dependency injection
20. Reusing and slightly modifying some previous Bootstrap
21. More than one way to use ng-class
22. Another Angular Filter
23. More usage of Ng-if
24. Finishing The App

Ng-Repeat Directive Will Do Your Work For You

Now armed with the ability to create controller properties with some data attached then inserting that data into our HTML we can move on to create something useful. We will be using the ng-repeat directive (which will save us a lot of typing. YAY!) along with bootstrap to start creating the list of turtles in our application.

Of course, in any application some data is needed. This data is usually fetched from anAPI on a backend server. However, for this tutorial we will just be focusing on the front end implementation of this application. As a result, we will need to simulate the data in some way.

If you want to check out the finished app, you can see it here.

The git repo can be found here.

Angular Video Tutorial Goodness

As always, for those of you more visually inclined this full tutorial is in video form below. For the rest of you continue reading below the video for the same content in written form.

Click the image to go to the video on youtube or go here https://www.youtube.com/watch?v=iAX67gisQ2M&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq&index=3

The next part of the series can be found here.

The way we will simulate the API request in this application is to simply paste the JSON data we would normally get from a server, straight into our code as a variable. We will then attach that variable as a property on our controller which will give us access to all of that data inside our HTML.

This variable will go inside our IIFE for the list controller but it will live outside of the function for the controller itself. We will then create a property on our controller that binds to this.

```
var turtlesData = [
        {
            type: "Green Turtle",
            image_url: "http://www.what-do-turtles-eat.com/wp-content/uploads/2014/10/Sea-Turtles-Habitat.jpg",
            locations: "Tropical and subtropical oceans worldwide",
            size: "Up to 1.5m and up to 300kg",
            lifespan: "Over 80 years",
            diet: "Herbivore",
            description: "The green turtle is a large, weighty sea turtle with a wide, smooth carapace, or shell. It inhabits tropical and subtropical coastal waters around the world and has been observed clambering onto land to sunbathe. It is named not for the color of its shell, which is normally brown or olive depending on its habitat, but for the greenish color of its skin. There are two types of green turtles–scientists are currently debating whether they are subspecies or separate species–including the Atlantic green turtle, normally found off the shores of Europe and North America, and the Eastern Pacific green turtle, which has been found in coastal waters from Alaska to Chile."
        },
        {
            type: "Loggerhead Turtle",
            image_url: "http://i.telegraph.co.uk/multimedia/archive/02651/loggerheadTurtle_2651448b.jpg",
            locations: "Tropical and subtropical oceans worldwide",
            size: "90cm, 115kg",
            lifespan: "More than 50 years",
            diet: "Carnivore",
            description: "Loggerhead turtles are the most abundant of all the marine turtle species in U.S. waters. But persistent population declines due to pollution, shrimp trawling, and development in their nesting areas, among other factors, have kept this wide-ranging seagoer on the threatened species list since 1978. Their enormous range encompasses all but the most frigid waters of the world's oceans. They seem to prefer coastal habitats, but often frequent inland water bodies and will travel hundreds of miles out to sea."
        },
        {
            type: "Leatherback Turtle",
```

So here we have simply created a turtlesData variable and set it equal to all of the JSON. The JSON itself is all the information for all of our turtles. There is a name, image, location, size, lifespan, diet and general description for each turtle.

Now we can create the property inside our controller function. That is as easy as creating a property and setting it equal to the variable we created called turtlesData.

```
function ListController(){
    var vm = this;

    vm.data = turtlesData;
}
```

This is because our controller function and the variable are within the same context in the code ie. they are both inside the same IIFE. (Don’t worry if you don’t understand that last sentence. It isn’t super important right now. Although, I will do a tutorial explaining all of that sort of stuff soon).

What is this Ng-Repeat Directive You Speak Of?

Now that we have access to the information about all of our turtles we can start thinking about how to create all of our markup.

The old fashioned way of doing that would be to hard code all of the HTML for each of our turtles individually. Of course this results in a lot of repeated code and waaaaaay too much typing for us to do. We are lazy remember!

In steps the ng-repeat directive. What this lovely directive allows us to do is declare an area of markup and tell Angular that it

should repeat that markup for each item in a dataset that we specify.

In our case we will create the markup for one of our turtles – also taking advantage of the {{}} binding to grab hold of each bit of information we want. Then using the ng-repeat directive, we will tell angular to simply repeat all of that markup for each turtle in the JSON. Phew! That's more like it.

We Need An Alias Again.

Much like we used an alias for our controller when we used the controller as syntax, we will also use an alias when using ng-repeat. The alias in this case will be the name we will use to reference each iteration of the repeating loop through our data.

The markup for a general ng-repeat will look something like this:

```
<div ng-repeat="data in controller.items">
    {{data}}
</div>
```

Here we have a property on our controller called items which is an object or array that we can loop through. We give each iteration through that loop an alias of data.

Using that alias we can put inside the {{}} syntax and that will print out the value of each respective value in the items array or object.

Of course, we can also harness this if our data is an object or array or objects or array ie nested objects or arrays (or multidimensional would be another name). This is in fact what

we are going to do.

The Markup For Our List of Turtles.

The top level property we are going to loop through is the data property on our list controller which of course is our JSON data. Each iteration through the ng-repeat will loop through each turtle. But each turtle has many properties such as type, image_url, location etc.

We can grab hold of these by using the dot notation something like this:

```
<div ng-repeat="turtle in list.data">
    {{turtle.type}}
</div>
```

This will print out the type property of each turtle in our data property. Using this we can now build the actual markup we need and lay out our information for each turtle nicely.

Bootstrapping Our App Markup With Bootstrap

This course is not focused on Bootstrap so I will not spend a lot of time explaining the different elements used. However, if you want to learn more let me know and I may do a full course on Bootstrap. Especially with Bootstrap 4 looming just around the corner.

Here is the start of the markup with the ng-repeat directive added in:

```
<div class="row">
    <div class="col-sm-6" ng-repeat="turtle in list.data">

    </div>
</div>
```

The class we have given the inside div here is what will make the element responsive to different screen sizes. The ng-repeat has been added to the inside div so that div and anything inside it will be repeated for every element in the data property on the list controller.

We now want to create the grey area that will contain the turtles themselves then add the image and all the info for each turtle respectively.

This is the Bootstrap to do that along with the bindings to the data that we need:

```
<div class="row">
    <div class="col-sm-6" ng-repeat="turtle in list.data">
        <div class="well well-sm">
            <div class="row">
                <div class="col-md-6">

                    <img ng-src="{{turtle.image_url}}"
                        class="img-rounded img-responsive well-image">
                </div>
                <div class="col-md-6">
                    <h4>{{turtle.type}}</h4>

                    <p><strong>Locations: </strong>{{turtle.locations}}</p>
                    <p><strong>Size: </strong>{{turtle.size}}</p>
                    <p><strong>Average Lifespan: </strong>{{turtle.lifespan}}</p>
                    <p><strong>Diet: </strong>{{turtle.diet}}</p>

                </div>
            </div><!-- row -->
        </div><!-- well -->
    </div><!-- col-xs-6 -->
</div>
```

ng-src? Wait! That’s new!

This should all be easy to understand for the most part. The one new thing that you haven’t seen yet is the ng-src on the image. Notice that the image does not have a normal src attribute at all.

The reason for this is because the URL for the image is coming from the JSON data and therefore we want to use the {{}} binding syntax to grab hold of it.

So you may think that we could just add that binding syntax to the normal src attribute and Angular will replace the binding with the url and the image will render just fine.

Unfortunately this is not the case. The reason for this is easy to understand though.

Basically, it comes down to the order things are rendered. With images when you use the normal src attribute the HTML tries to fetch the URL straight away as the page is loading. Importantly, this is before Angular has had time to load and hook into the page.

This of course means that the HTML will see {{turtle.image_url}} when it looks at the src attribute. This literal string of course isn’t a URL and the HTML will not know what to do with it.

Finally when Angular loads it will change that literal string to the URL that we want but by that stage the HTML thinks all of

the work with image urls is done and it will not try to fetch the url and therefore no image will be displayed.

To stop this problem, Angular added in a helpful directive called ng-src which is to be used in place of the normal src attribute when dealing with Angular model data.

Now the HTML doesn't see a normal src attribute so it will simply skip that image. But when Angular loads up and hooks into the page it will see the ng-src and fetch the image from the url that it obtains from our data and the image will now display as we want.

Finishing Off The List Markup

The final thing we need to do is add the markup for the "Learn More" button. We will be using simple Bootstrap classes to style the button. The markup itself will go directly after the paragraphs that contain all the turtle information.

We will need to used two Bootstrap attributes on this button as will intend this button to trigger the modal that we will create later.

```
<button class="btn btn-primary pull-right"
    data-toggle="modal"
    data-target="#turtle-info">Learn More</button>
```

The data-toggle bootstrap attribute let's bootstrap know that we intend to use this button to trigger a modal.

The data-target bootstrap attribute tells bootstrap exactly which modal we want to trigger with this button. In this case we have told it will be the modal with the id of "turtle-info". We

will need to remember that and give the modal that id when we create it.

Moving On To Part 4

That’s it for this part. In part 4 we will fix up the image sizing issues that we have with the images in our list and start creating the modal markup.

See you there…

Adrian