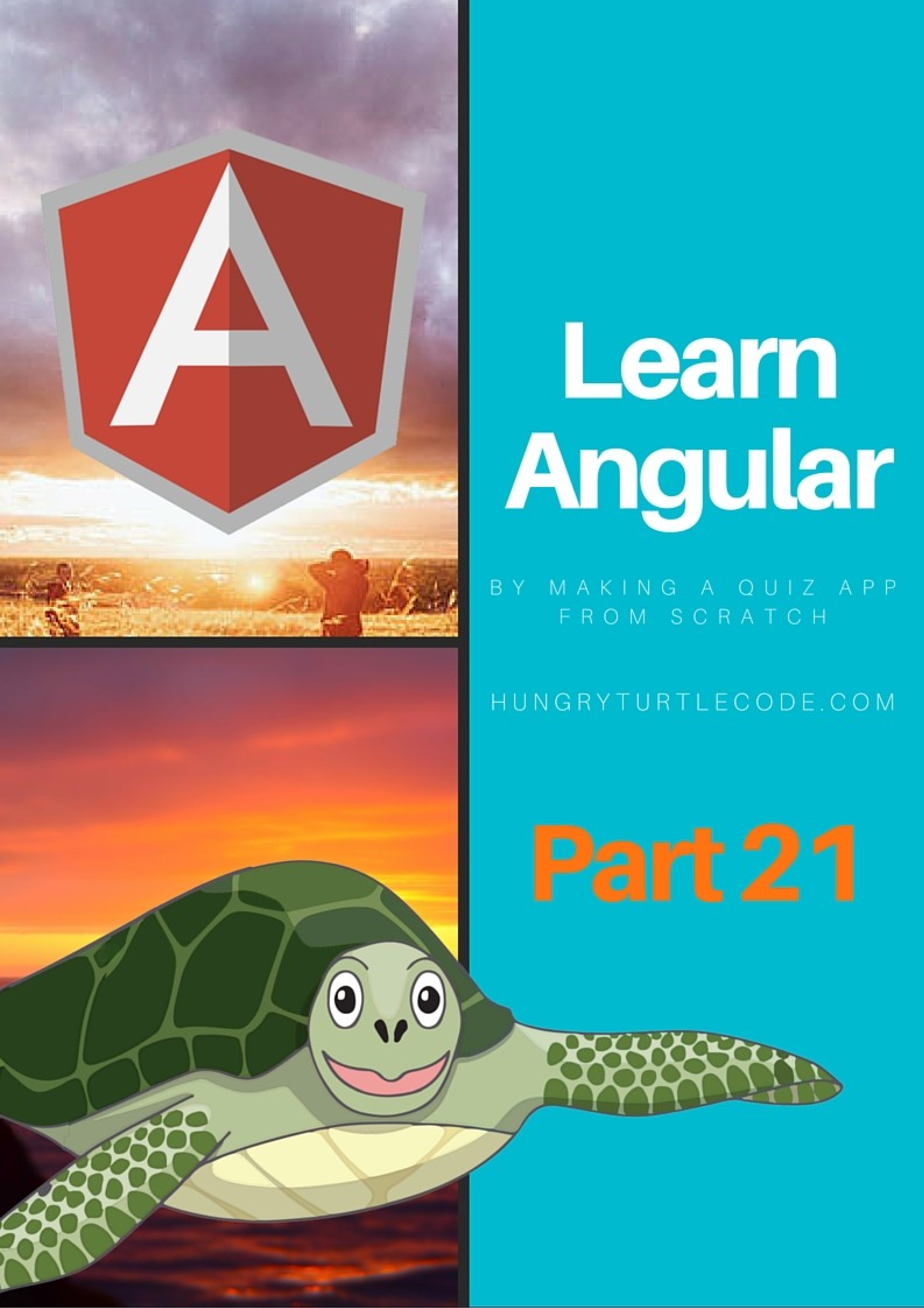


Learn Angular

BY MAKING A QUIZ APP FROM SCRATCH

HUNGRYTURTLECODE.COM

Part 21

Check Out The Whole Course Index

Deja Vu + A New Way To Use Ng-Class

Much like the previous part, the markup in this part will be very similar to that of the markup in the quiz controller. But we will spice things up a bit by adding new elements and showing a different way of using ng-class – using a function with ng-class instead of an object with name:value pairs.

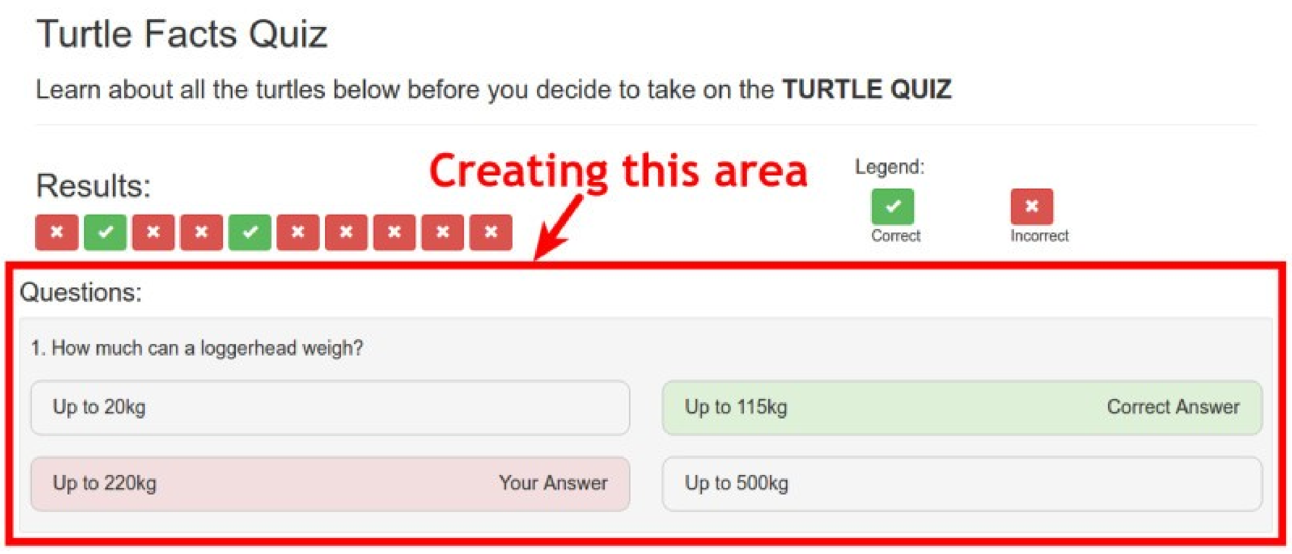
If you want to see the app for yourself, check it out here.

The git repo can be found here.

Click the image to go to the video on youtube or go here https://www.youtube.com/watch?v=bGyvJMprAcg&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq&index=21

The next part can be found here

Below the row we created in the last part we will continue to create the markup for the question area of the results section.


Turtle Facts Quiz

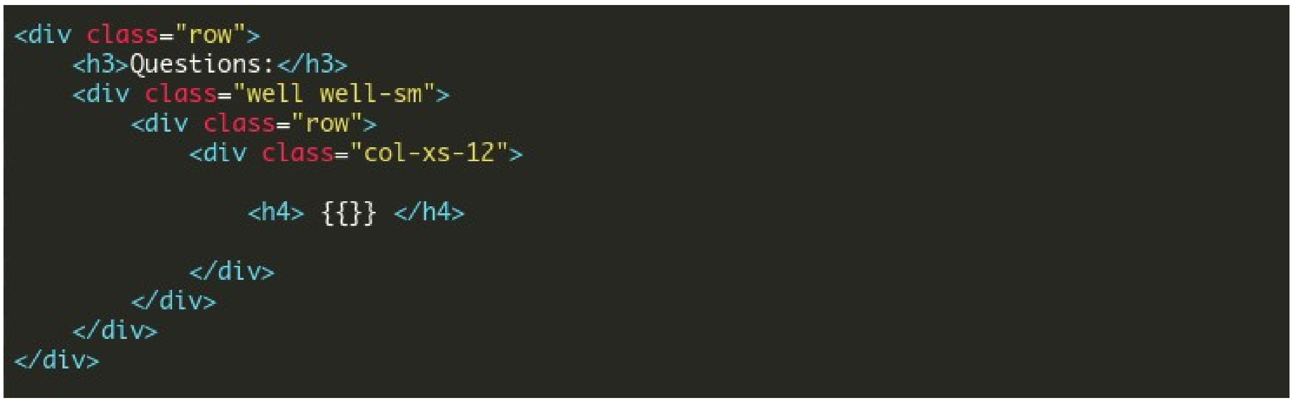
Learn about all the turtles below before you decide to take on the **TURTLE QUIZ**

Results:

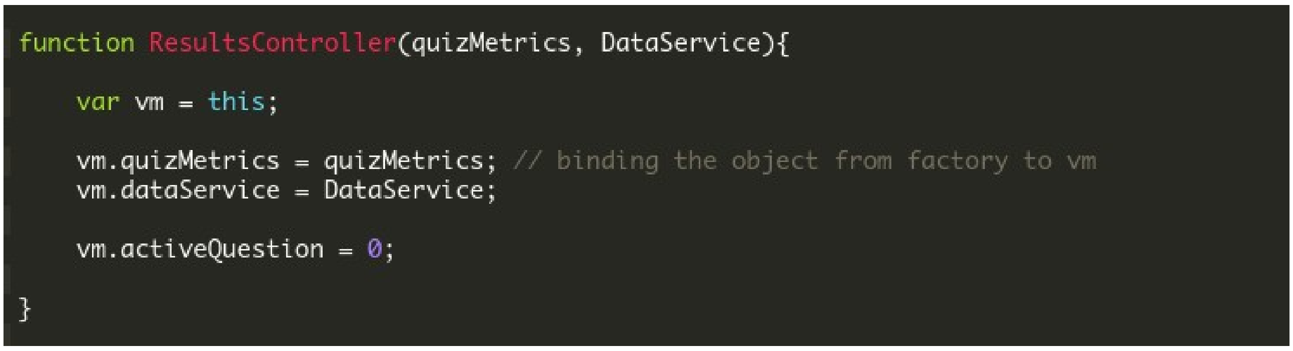
Questions:

1. How much can a loggerhead weigh?

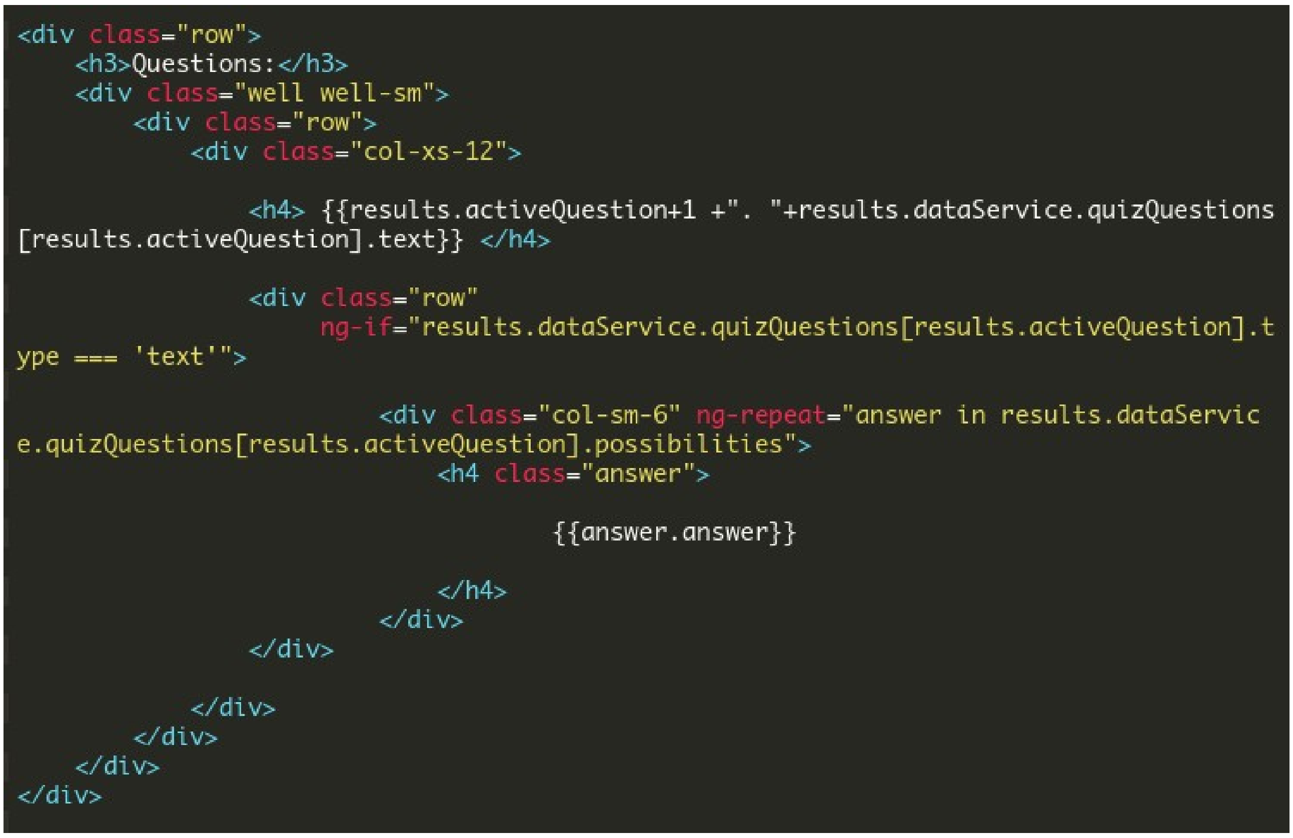
Up to 20kg	Up to 115kg — Correct Answer
Up to 220kg — Your Answer	Up to 500kg


```
<div class="row">
    <h3>Questions:</h3>
    <div class="well well-sm">
        <div class="row">
            <div class="col-xs-12">

                <h4> {{}} </h4>

            </div>
        </div>
    </div>
</div>
```

Inside the {{}} syntax of the h4 we will need to reference the active question again to ensure the correct data is showing. This means we will need to create an active question variable on the results controller.


```
function ResultsController(quizMetrics, DataService){

    var vm = this;

    vm.quizMetrics = quizMetrics; // binding the object from factory to vm
    vm.dataService = DataService;

    vm.activeQuestion = 0;

}
```

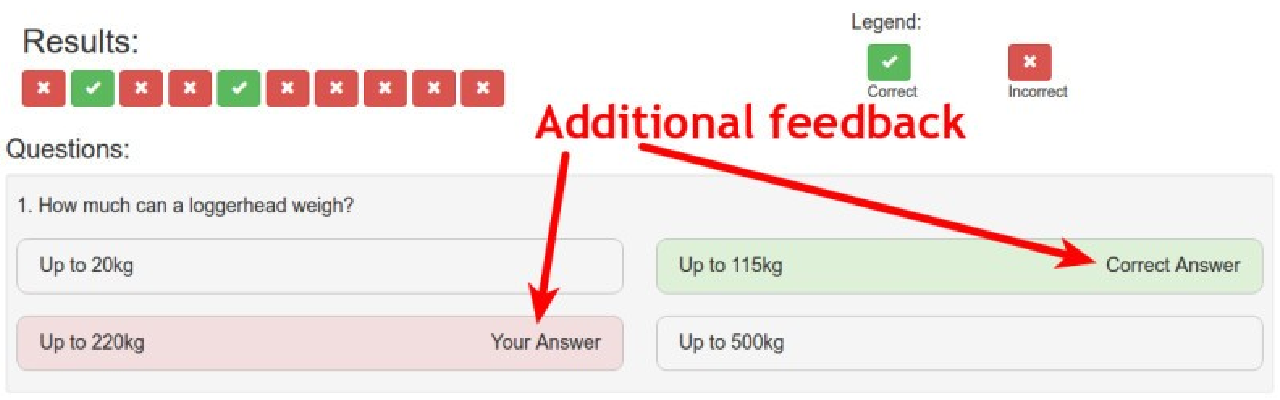
Now that that's done we can write the code into that h4. It will be much like the code in the h4 of the quiz controller so if you don't quite understand what is going on here go back to the part where we created this markup in the quiz controller. Continuing on, we will also ng-repeat all the possible answers of the active question.



```
<div class="row">
    <h3>Questions:</h3>
    <div class="well well-sm">
        <div class="row">
            <div class="col-xs-12">

                <h4> {{results.activeQuestion+1 +". "+results.dataService.quizQuestions
[results.activeQuestion].text}} </h4>

                <div class="row"
                     ng-if="results.dataService.quizQuestions[results.activeQuestion].t
ype === 'text'">

                         <div class="col-sm-6" ng-repeat="answer in results.dataServic
e.quizQuestions[results.activeQuestion].possibilities">
                             <h4 class="answer">

                                     {{answer.answer}}

                             </h4>
                         </div>
                </div>

            </div>
        </div>
    </div>
</div>
```

Adding More Feedback To Each Answer

If you have seen the final application, you will notice that the answers in the results section display “correct answer” and if you gave a different answer it will display “your answer” on that on to let you know what answer you gave and what the actual correct answer was.



To implement that we need to create the two sets of text and ng-show then depending on the conditions of each possible answer.

```
<h4 class="answer">

    {{answer.answer}}

    <p class="pull-right"
       ng-show="$index !== results.quizMetrics.correctAnswers[results.activeQuestion] &
& $index === results.dataService.quizQuestions[results.activeQuestion].selected">Your A
nswer</p>

    <p class="pull-right"
       ng-show="$index === results.quizMetrics.correctAnswers[results.activeQuestion]">
Correct Answer</p>

</h4>
```

Fancy Expressions and a Function With Ng-Class

The expressions in the above code snippet may look a bit daunting initially but let’s walk through them.

Starting with the “Correct Answer” text. The expression in the ng-show is comparing the $index (which is the current index of the ng-repeat) to the correct answer for this particular question. We are doing this by referencing the correctAnswers property that we added to the quizMetrics factory earlier. Using activeQuestion to get the answer for the current active question.

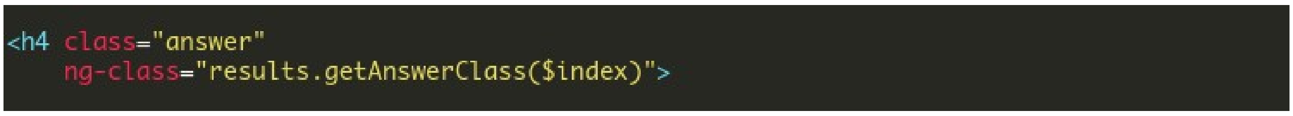
The ng-show on the “Your Answer” text is a bit more complicated. We are using two conditions here. The first one is checking the this possible answer is not the correct answer (using the same method as we did for the “Correct Answer” area). We do this as if the user gave the correct answer we can simply display “Correct Answer”, no need to tell them that was also the answer they gave. The absence of the “Your Answer” prompt implies their answer is correct.

The next condition to the expression is checking if this answer is in fact that answer that the user gave. We do this by referencing the selected property on the question, which of course stores the user’s answer for that particular question.

Both of these conditions have to be true, for the “Your Answer” text to display on the question.

The Questions Need Backgrounds

We want to style the correct answer with a green background and if the user answered incorrectly, style their answer with a red background. We will do this using ng-class. But instead of using the object notation for ng-class like we have done in the past, we will give it a function that returns the class to display.

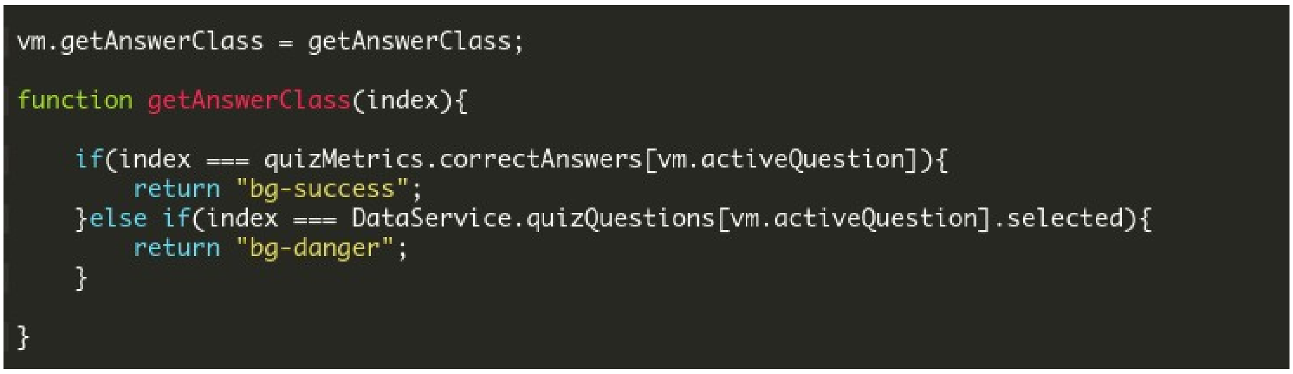


```
<h4 class="answer"
    ng-class="results.getAnswerClass($index)">
```

It calls the getAnswerClass function (which we will create shortly) and passes in the $index from the ng-repeat. This function can then do some logic and figure out which class should be added and return that from the function. Whatever is returned by the function will be added as a class to the element by ng-class.

Heading into the results controller, let’s create the function. All we will need to do inside the function is figure out if the argument passed in ($index) is the index of the correct answer. We will do this by referencing the array of correct answers we have used in the past.

If the argument is the correct answer, we return the green background class which is the bootstrap class of bg-success. If that conditional fails, there is an else if block that test if the argument passed in is the answer the user gave. If it is, then the class of bg-danger is returned, which is the red background.



```
vm.getAnswerClass = getAnswerClass;

function getAnswerClass(index){

    if(index === quizMetrics.correctAnswers[vm.activeQuestion]){
        return "bg-success";
    }else if(index === DataService.quizQuestions[vm.activeQuestion].selected){
        return "bg-danger";
    }

}
```

We can have the else if simply check if the answer is the one the user gave and not check if it also isn’t the correct answer (like we did in the ng-show) because we already know the answer wasn’t correct as the first if statement failed.

End Of The Road For This Part

There isn’t much left for us to do in the application. The next thing to do is to create the function that will allow the user to hop between questions using the buttons at the top.

See you in part 22 for that

Adrian