Learn Angular

BY MAKING A QUIZ APP FROM SCRATCH

HUNGRYTURTLECODE.COM

Part 17

Check Out The Whole Course Index

The End Of The Quiz Controller Is Near

The only thing left for us to do with the quiz controller is just prompt the user when they have finished just to confirm they want to move onto the results page. The finalise flag will come in useful here to allow the use of ng-show to show the prompt when the end is reached.

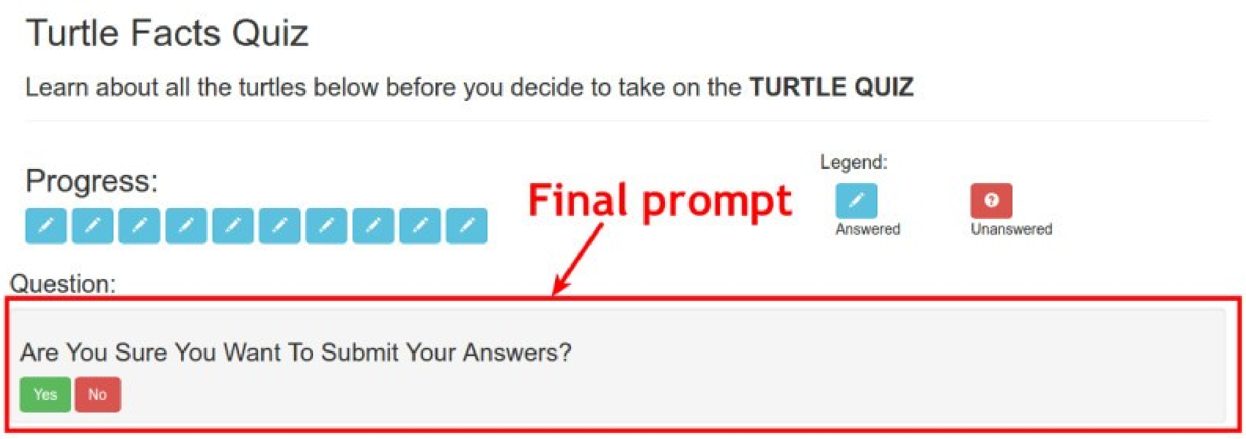
If you want to see the app for yourself, check it out here.

The git repo can be found here.

Click on the image to go to the video on youtube or go here https://www.youtube.com/watch?v=6uZfUfB4bN8&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq&index=17

The next part can be found here

The code for the little prompt is simple, it is just a bootstrap well with two buttons. We will place this markup at the end of the row that contains the well for the questions. This is the markup we will need.



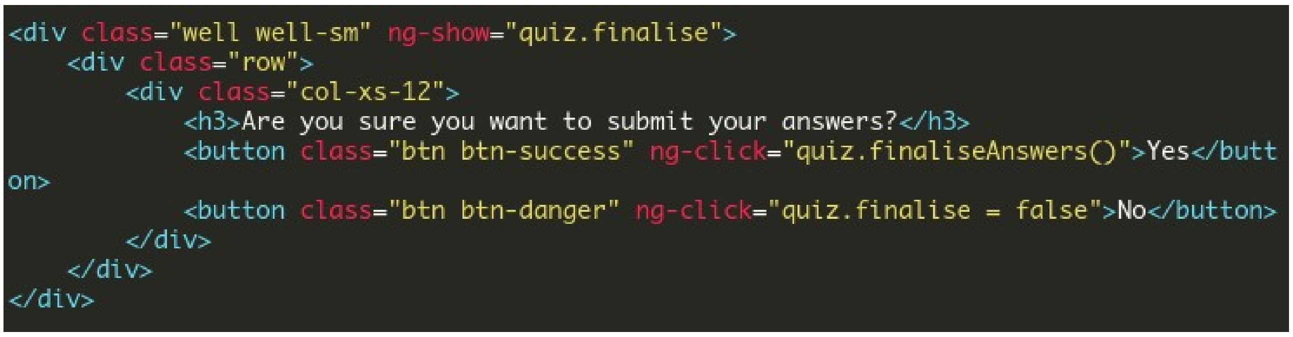
Turtle Facts Quiz

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

Progress:

Question:

Are You Sure You Want To Submit Your Answers?

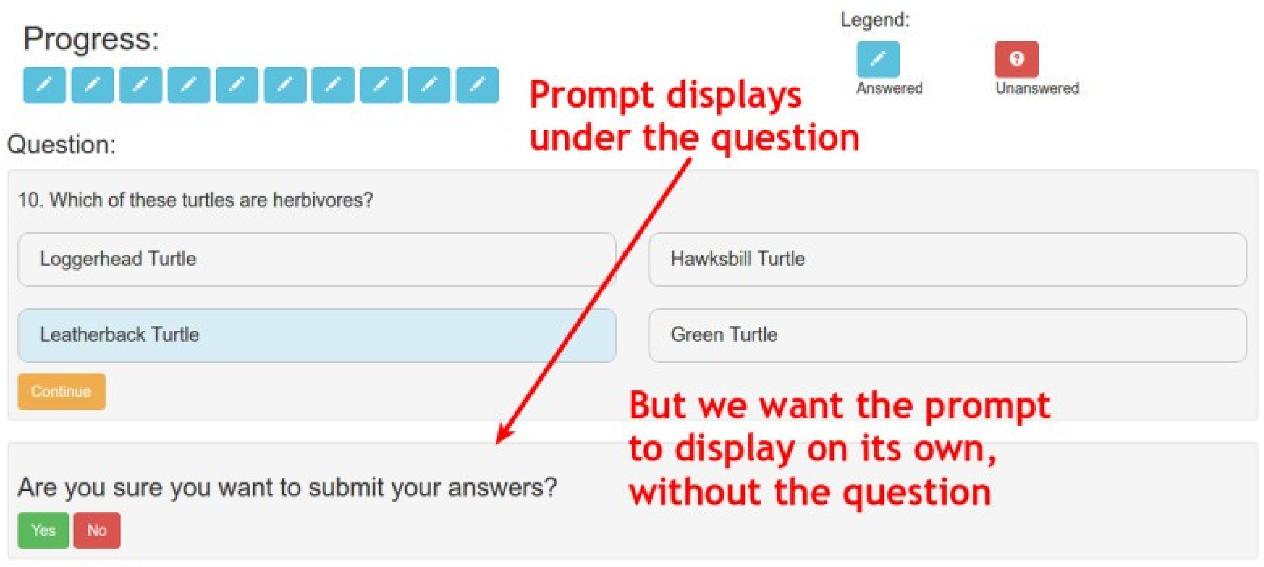


```
<div class="well well-sm" ng-show="quiz.finalise">
    <div class="row">
        <div class="col-xs-12">
            <h3>Are you sure you want to submit your answers?</h3>
            <button class="btn btn-success" ng-click="quiz.finaliseAnswers()">Yes</butt
on>
            <button class="btn btn-danger" ng-click="quiz.finalise = false">No</button>
        </div>
    </div>
</div>
```

Onto the well that of this markup we have added the trusty ng-show directive that will only show this prompt when the finalise property is true.

Each of the buttons both have an ng-click directive attached onto them. The no button simply sets finalise back to false which removes the prompt and keeps the user in the quiz to make some changes. Simple enough.

The yes button contains a function we will need to create. This function will reset all the properties and variables that we have been using throughout the course of the life of the quiz.


Progress:

Question:

10. Which of these turtles are herbivores?

Loggerhead Turtle

Hawksbill Turtle

Leatherback Turtle

Green Turtle

Are you sure you want to submit your answers?

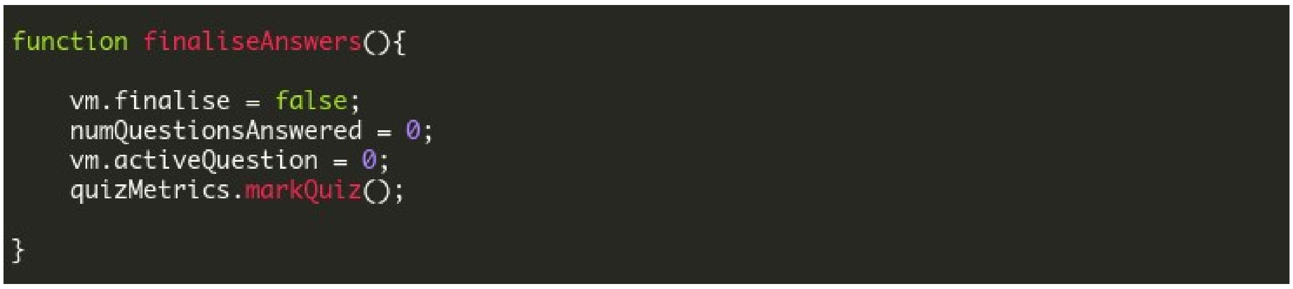
Legend:
Answered
Unanswered
Continue
Yes
No
Prompt displays under the question
But we want the prompt to display on its own, without the question

You may have noticed that the prompt simply displays under the quiz questions at the minute. This isn't exactly what we want. Instead, we want the questions to hide while the prompt is showing just to avoid possible confusion. This way the user is in no doubt what they have to do. We do this by adding an ng-hide to the question well div.

```
<h3>Question:</h3>
<div class="well well-sm" ng-hide="quiz.finalise">
```

Another Method Inside The Controller

The final thing for us to do is create the function that will run when the user clicks the yes button in the final prompt – the finaliseAnswers function. I won't bore you by repeating the process to add another function to the controller.



```
function finaliseAnswers(){

    vm.finalise = false;
    numQuestionsAnswered = 0;
    vm.activeQuestion = 0;
    quizMetrics.markQuiz();

}
```

Note that this is just the function itself. This will be inside the quiz controller and it will be added to the view model in the same way as before – using a named function – something like `vm.finaliseAnswers = finaliseAnswers.`

The first three lines inside the function are pretty self explanatory – just resetting the properties to their default values.

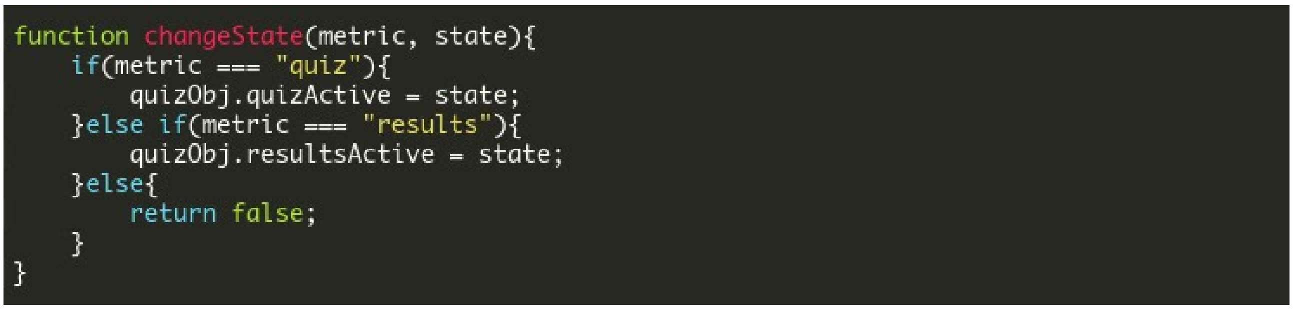
Next, we will call a function which we will create in the next part. This function will mark the answers the user gave against the correct answers to the quiz.

Refactoring the changeState Function

Earlier we created a function called changeState that takes a boolean argument and changes the state of the quiz to that boolean. However, shortly we will be creating another controller – the results controller. This means the results controller will also have a state that will need manipulating.

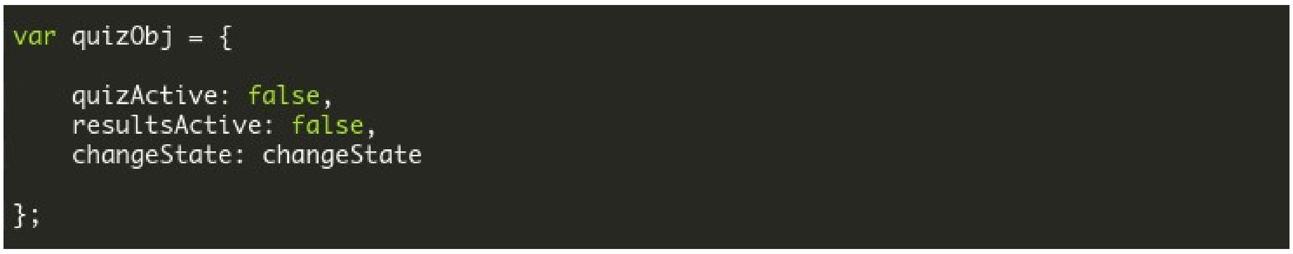
So let’s modify the changeState function to take a second argument which will be the metric to change – quiz or results controller. This way the same function can be used to manipulate the state of both areas of the quiz.

Here is the new version of the changeState function.



```
function changeState(metric, state){
    if(metric === "quiz"){
        quizObj.quizActive = state;
    }else if(metric === "results"){
        quizObj.resultsActive = state;
    }else{
        return false;
    }
}
```

I also added a new property to the quizObj object to make this possible – resultsActive. This is needed to ensure we actually have a state to change.

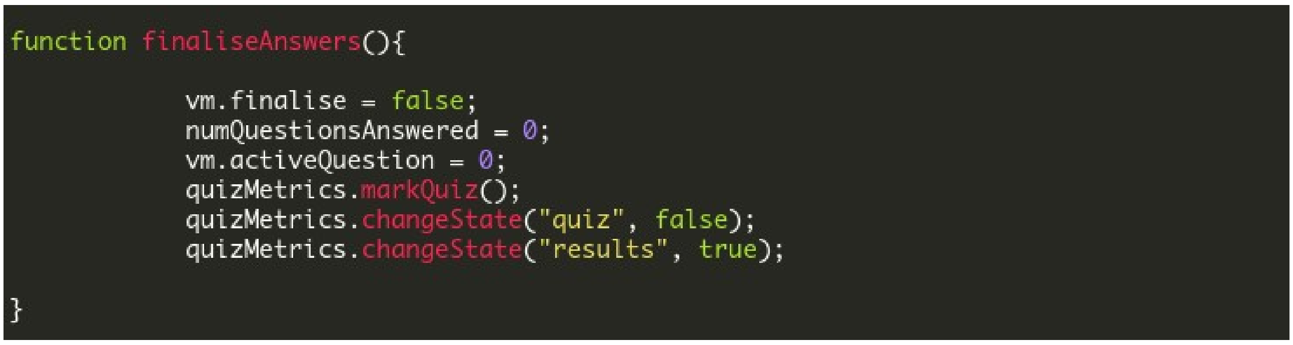


```
var quizObj = {

    quizActive: false,
    resultsActive: false,
    changeState: changeState

};
```

With this new function, we can complete the finaliseAnswers function by setting the quiz state to false (which removes the quiz from the view) and set the results state to true, which will take us to the results page.



```
function finaliseAnswers(){

            vm.finalise = false;
            numQuestionsAnswered = 0;
            vm.activeQuestion = 0;
            quizMetrics.markQuiz();
            quizMetrics.changeState("quiz", false);
            quizMetrics.changeState("results", true);

}
```

Remember To Go Back And Change What We Have Broken

Right now, we have broken the call to the changeState function we had in the list controller that triggered the quiz in the first place. We only passed in a state and not a metric to that. The code we broke was in the activateQuiz function inside the list controller.

```
function activateQuiz(){

    quizMetrics.changeState("quiz", true);

}
```

Now the code is working again.

The Quiz Needs To Be Marked!

We called the markQuiz function but it is yet to be created. So in the next part we will create that function inside the quizMetrics factory.

See you in part 18.

Adrian