# Learn Angular

BY MAKING A QUIZ APP FROM SCRATCH

HUNGRYTURTLECODE.COM

## Part 15

## Check Out The Whole Course Index

# Custom CSS For Image Feedback + Code Reuse

Having nice user feedback is great, but move forward to the image questions and you will find that it all breaks. There is no feedback whatsoever for the image questions. In this part we will fix that problem. Then we will see how we can start reusing code we have already written to make the progress buttons skip to their corresponding question.

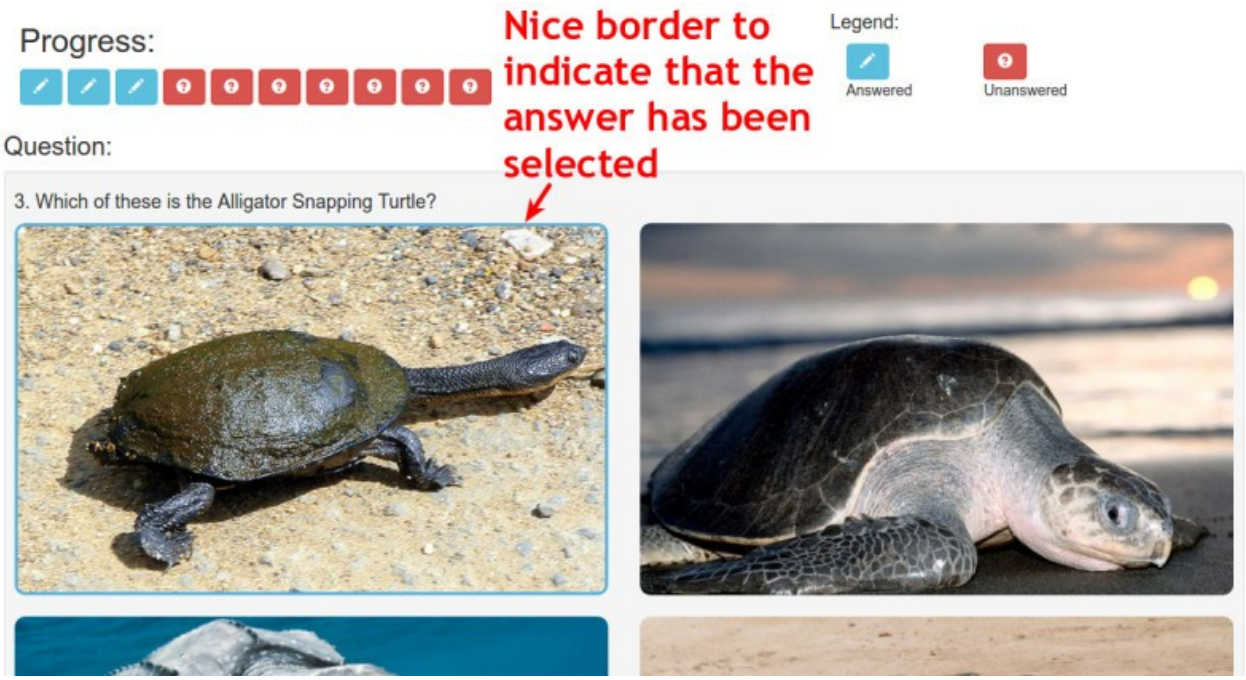
If you want to see the app for yourself, check it out here.

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=kDQco9gfYmo&index=15&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq

The next part can be found here

Much like we did for the text questions, we will add an ng-class and ng-click directive onto the row that handles the images. Instead of giving it a bootstrap class though, we will give it a custom class that we will style ourselves. This is because a background would be useless for our image. Instead, we want a nice border.




```
<div class="row"
     ng-if="quiz.dataService.quizQuestions[quiz.activeQuestion].type === 'image'">

          <div class="col-sm-6" ng-repeat="answer in quiz.dataService.quizQuestions[qui
z.activeQuestion].possibilities">
               <div class="image-answer"
                    ng-class="{'image-selected': $index === quiz.dataService.quizQuestion
s[quiz.activeQuestion].selected}"
                    ng-click="quiz.selectAnswer($index)">

                        <img ng-src="{{answer.answer}}">

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

The ng-click is identical this time as it was to the last. We are still using $index to update the selected flag on the data for the question.

### Let’s Get Stylish

The css required for this is extremely simple. All we need to do is add a border. We don’t need to round the edges as that has already been done by bootstrap. We add this line to the style.css.

```
.image-selected{
    border: 3px solid #56afdc;
}
```

## The Progress Buttons Need Attention

We have spoken about how the buttons in the progress area will be used to allow the user to navigate to specific questions in the quiz. Currently though, clicking on these buttons will not do anything. So let’s fix that by adding the functionality required.


### Reusing Code Is Good Right?

Remember we have already created a function called setActiveQuestion that will increment the active question to the next available unanswered question. But what if we allowed the

function to accept an optional argument of the index of question to go to.

This when the function is given an question index as an argument it will just automatically set activeQuestion to that index. But if no argument is given then it will just find the next unanswered question like it is doing now. This sounds like a good plan to me.

First things first, go to the markup for the buttons in the progress area. On this button html, add and ng-click directive that triggers the setActiveQuestion function. We can take advantage of the fact we are using ng-repeat to create all the buttons and use $index to find the index of the current button. This is what we will pass as the argument to setActiveQuestion.

```
<button class="btn"
        ng-repeat="question in quiz.dataService.quizQuestions"
        ng-class="{'btn-info': question.selected !== null, 'btn-danger': question.selec
ted === null}"
        ng-click="quiz.setActiveQuestion($index)">

    <span class="glyphicon"
          ng-class="{'glyphicon-pencil': question.selected !== null, 'glyphicon-questio
n-sign': question.selected === null}"></span>
</button>
```

Time to add the new functionality into the setActiveQuestion function.

We start by using a conditional to check if an argument was passed into the function. If it wasn’t we run the code we already have and if it was we set active question to the index passed in. Simples.

```
function setActiveQuestion(index){

    if(index === undefined){
        var breakOut = false;

        var quizLength = DataService.quizQuestions.length - 1;

        while(!breakOut){

            if(DataService.quizQuestions[vm.activeQuestion].selected === null){
                breakOut = true;
            }

        }
    }else{
        vm.activeQuestion = index;
    }

}
```

Using `index === undefined` we can check if the function has been given an argument or not. Giving us the exact functionality we want. The else block here is also pretty self explanatory.

## Onward and Upward…

Moving swiftly on to part 16 of this tutorial series. In the next part we will take a look at adding some basic error handling and the final parts of the quiz controller.

See you in part 16.

Adrian