



# Learn Angular

BY MAKING A QUIZ APP  
FROM SCRATCH

[HUNGRYTURTLECODE.COM](http://HUNGRYTURTLECODE.COM)

## Part 13



## Check Out The Whole Course Index

1. [Getting Started](#)
2. [Ng-controller directive and the \(mis\)use of \\$scope](#)
3. [Looping around with the ng-repeat directive](#)
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. [You are here](#)
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](#)

# What (ng-)if You Don't Want To Display It All

The quiz is starting to look distinctly like a quiz! Go us. But unfortunately, the image questions do not display correctly. They are showing the url instead of the image. In this part we will introduce the [ng-if directive](#) to solve that problem.


If you want to see the app for yourself, [check it out here](#).



The git repo [can be found here](#).



Click the image or go to [https://www.youtube.com/watch?v=BRD5DUQgLF4&list=PLqr0oBkln1FBmOjK24\\_B4y\\_VAA8736wPq&index=13](https://www.youtube.com/watch?v=BRD5DUQgLF4&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq&index=13)

[The next part can be found here](#)

Progress: 

Legend:  Answered  Unanswered

Question:

3. Which of these is the Alligator Snapping Turtle?

[https://c1.staticflickr.com/3/2182/2399413165\\_bcc8031cac\\_z.jpg?zz=1](https://c1.staticflickr.com/3/2182/2399413165_bcc8031cac_z.jpg?zz=1)

[http://images.nationalgeographic.com/wpf/media-live/photos/000/006/cache/ridley-sea-turtle\\_688\\_600x450.jpg](http://images.nationalgeographic.com/wpf/media-live/photos/000/006/cache/ridley-sea-turtle_688_600x450.jpg)

<https://static-secure.guim.co.uk/sys-images/Guardian/Pix/pictures/2011/8/13/1313246505515/Leatherback-turtle-007.jpg>

[https://upload.wikimedia.org/wikipedia/commons/e/e3/Alligator\\_snapping\\_turtle\\_-\\_Geierschildkr%C3%B6te\\_-\\_Alligatorschildkr%C3%B6te\\_-\\_Macrochelys\\_temminckii\\_01.jpg](https://upload.wikimedia.org/wikipedia/commons/e/e3/Alligator_snapping_turtle_-_Geierschildkr%C3%B6te_-_Alligatorschildkr%C3%B6te_-_Macrochelys_temminckii_01.jpg)

**Currently, the URLs are displaying instead of the images**

[Continue](#)

The way we will solve this is by duplicating the html that displays the possible answers using ng-repeat. Then instead of displaying the text using answer.answer inside an h4, we will add an image using ng-src="answer.answer". This way the url is in the place it should be - the src attribute of an image.

```
<div class="row">
  <div class="col-sm-6" ng-repeat="answer in quiz.dataService.quizQuestions[quiz.activeQuestion].possibilities">
    <div class="image-answer">
      
    </div>
  </div>
</div>
```

Note that we have surrounded the image with a div of class image-answer. This will be used later to make sure that all images are uniform size on the page and not displaying in their native sizes.

However, we now have two blocks of code that will both display. When it's a text based question, the text will display nicely, but we will also have an image that is trying to fetch a url that doesn't exist (the text of the possible answers is obviously not a url). We get the opposite problem when it's an image question.

So what we want to do is somehow only show one of these code blocks depending on what style of question the currently active question is. This is where the [ng-if directive](#) comes in.

## What Exactly Does This Ng-If Do?

Ng-If achieves the exact functionality we are looking for. It accepts an expression that evaluates to a boolean; if this value is true it will add the html that it is attached to, to the page. If the value is false the html will not be rendered at all.

This may sound similar to what the ng-show and ng-hide directives do, but there is one key distinction. Ng-show and ng-hide are just like adding **display: none** to the css for that element. The element is still there in the html document, it just cannot be seen on the screen – it is hidden. But ng-if does not even render the html to the page at all.

The reason we are using ng-if here instead of ng-show or hide is because if we use ng-show or hide we will still end up with images that have src attributes that are not urls. The browser will still try to find them as urls and when they can't they throw errors. Ng-if never allows this scenario and is thus a bit cleaner. We like clean code.



Here is the two blocks of code with the ng-if directives added to both

```
<div class="row"
  ng-if="quiz.dataService.quizQuestions[quiz.activeQuestion].type === 'text'">
  <div class="col-sm-6" ng-repeat="answer in quiz.dataService.quizQuestions[quiz.activeQuestion].possibilities">
    <h4 class="answer">
      {{answer.answer}}
    </h4>
  </div>
</div>

<div class="row"
  ng-if="quiz.dataService.quizQuestions[quiz.activeQuestion].type === 'image'">
  <div class="col-sm-6" ng-repeat="answer in quiz.dataService.quizQuestions[quiz.activeQuestion].possibilities">
    <div class="image-answer">
      
    </div>
  </div>
</div>
```

As you can see, this is why we added the question type to the JSON for each question. We can now simply query what the type of the currently active question is and we are done.

## MOAR CSS Rules!

There isn't much css that needs to be added to fix the sizing issues that we are left with on the images. But they need to be done.

```
.image-answer{
  cursor: pointer;
  height: 350px;
  width: 100%;
  overflow: hidden;
  border-radius: 10px;
  margin-bottom: 20px;
}
.image-answer img{
  width: 100%;
  height: auto;
}
```

All of this should be pretty self explanatory. It will scale up/down all images to the same size and style them nicely with rounded corners etc. We also add the pointer cursor to indicate it is clickable to the user.

## See You In Part 14

That's all for this part. In [part 14](#) we will add some feedback in the interface to let the user know they have successfully selected an answer.

See you over there in [part 14](#).

Adrian