Exercise 6 - Building Custom Directives

Objective

To create a custom attribute directive and apply it to our app

Overview

In this exercise you are going to build a custom Attribute Directive which we can use to apply a greyscale->full-colour effect triggered by a user hovering over any element we place it on.

Instructions

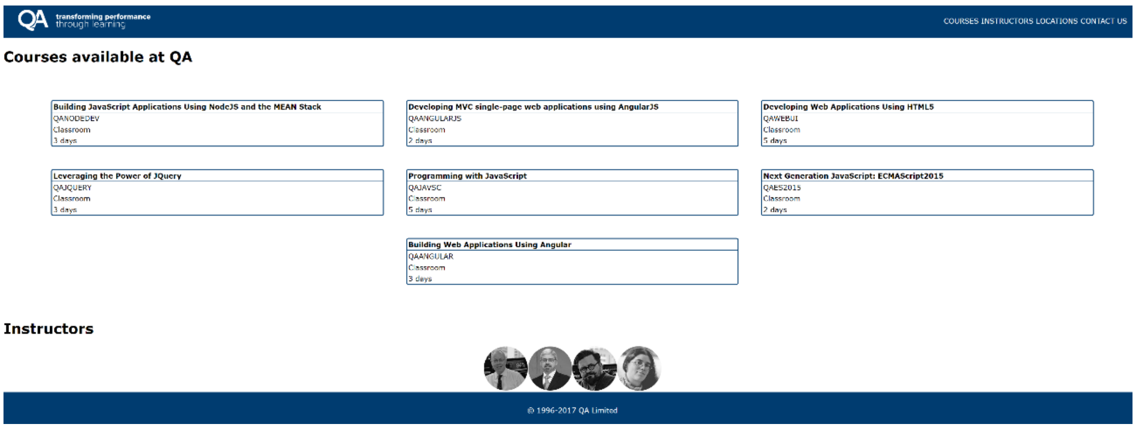
Creating the Directive

1. The directive we're going to create here is fairly generic and could be used across the application and so we're going to want to make it easily accessible to various components. Create a shared module which will house this directive.
2. Now within this new shared module, create a Greyscale directive. This will be an attribute directive so ensure that the appropriate item is injected in the constructor. Don't forget that Directives need to be exported.
3. Within the directive use the element reference object to set the below styles on the target (native) element

filter	grayscale(100%)
transition	filter 1s

4. Import the shared module into the instructors module and apply your new directive to the gallery images within instructors-gallery.

5. If you test the page now you should find it looks like the below image, with grey instructor images



transforming performance through learning
COURSES INSTRUCTORS LOCATIONS CONTACT US

Courses available at QA

Building JavaScript Applications Using NodeJS and the MEAN Stack
QANODEDEV
Classroom
3 days

Developing MVC single-page web applications using AngularJS
QAANGULARJS
Classroom
2 days

Developing Web Applications Using HTML5
QAWEBUI
Classroom
5 days

Leveraging the Power of JQuery
QAJQUERY
Classroom
3 days

Programming with JavaScript
QAJAVSC
Classroom
5 days

Next Generation JavaScript: ECMAScript2015
QAES2015
Classroom
2 days

Building Web Applications Using Angular
QAANGULAR
Classroom
3 days

Instructors

© 1996-2017 QA Limited

Reacting to Users

We now need to colourise the images when a user hovers over each

6. Create two methods in the directive onMouseOver and onMouseLeave which have the appropriate HostListeners attached to them. In the onMouseOver method the grayscale filter should be set to 0% and in onMouseLeave it should be set back to 100%.
7. Test the page and you should find the greyscale images transition to full colour when you mouse over them.

Taking Input from Consumers

An important aspect of Directives is their ability to take input from the consumers of them. In this instance it would be good to be able to specify how much grayscale filter should be used. See if you can do this without any further direction - but the steps below can be followed if you wish.

8. First create an Input property on the directive called appGreyscale - this means the consumers can simply use the directive attribute itself to set the value rather than adding another attribute to the element in addition to the directive selector. Set its value to "100%" so we have a default.
9. Update the code in the directive to use this new property in place of the hard-coded 100% we put in before.
10. Now update the template to pass 50% in to the directive and test the page. You should find that the images are somewhat more colourful than they started at before and they still go full-colour when hovered.
11. Do your images start at 100% greyscale when you loaded the page? If so, give some thought as to when Input properties are bound.