Exercise 18 – Advanced Bootstrap – JavaScript

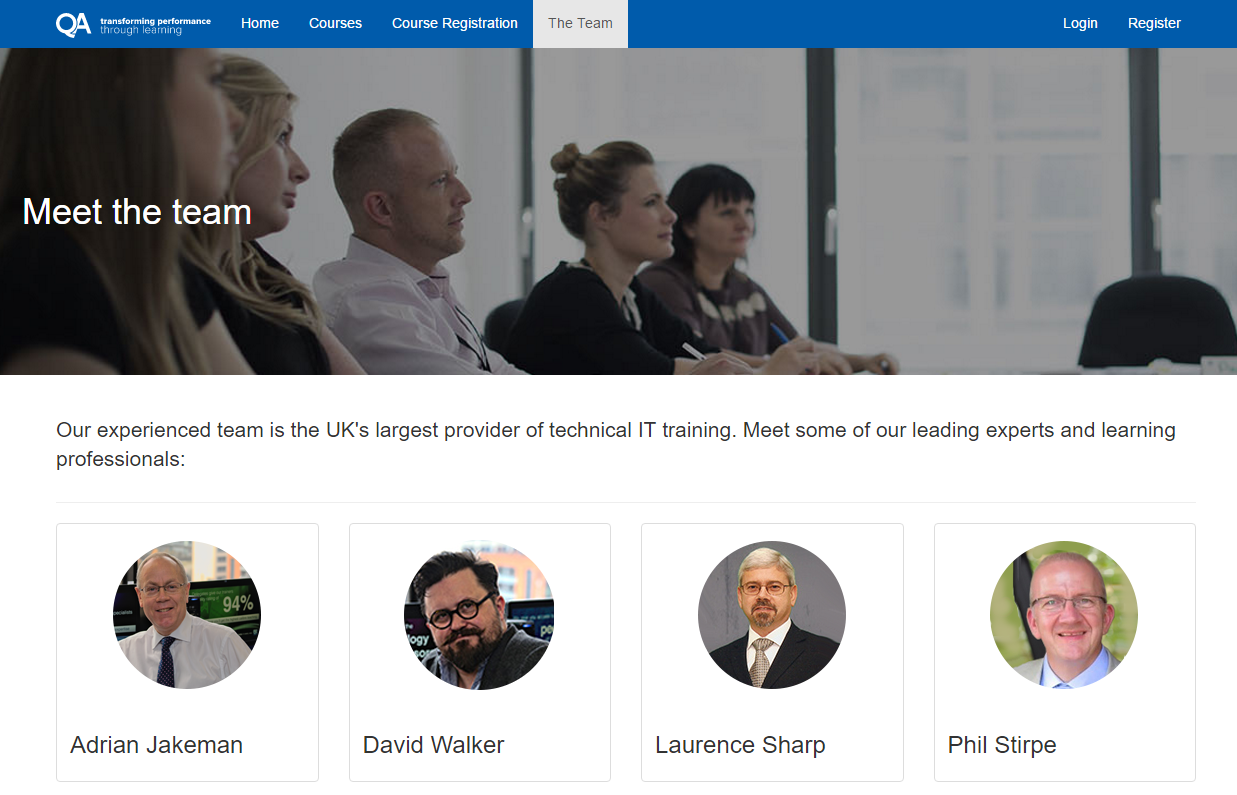
Objective

This exercise is designed to give you familiarity and practice working with the various JavaScript add-ons available with Bootstrap. In part one of the exercise you will create expandable content areas for our team member profiles, then in part two of the exercise you will expand our nav bar to contain a dropdown menu which gives further depth to our site. In the final part of the exercise you will improve the individual course pages to include a tabbed navigation element.

Part 1 – Enhancing The Team page

In this part of the exercise we’re going to enhance the “The Team” page to add some user interactivity and improve the aesthetics slightly. I’m not a big fan of uneven columns.

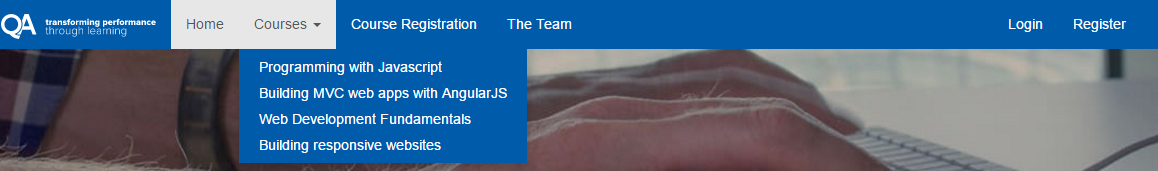
1. First of all, we need to add in some classes and id’s. Open up team.html and find the part of the code dealing with the team member profiles.
2. Each of the profiles has a paragraph element with a description of their profile. Add the class of “collapse” and an id of “<name>-collapse” to this paragraph element. So for example Adrian’s profile will have an id “adrian-collapse”
3. That deals with collapsing the paragraph of text. Easy cheesy right? Now we need to deal with expanding it. To the containing div with a class of thumbnail add a data-toggle with a value of “collapse”, and a data-target with a value equal to the id of the paragraph element you want to toggle the collapse of. So for example on Adrian’s thumbnail container you’d add data-target=”#adrian-collapse”.
4. You should have a page that looks like the below, and if you click on any part of the thumbnail it will expand to show you the text.



1. You’ll notice that when as the screen shrinks and the profiles start going into rows that when you expand/collapse the row above, profile start to jump left and right as you do so. Let’s fix that.
2. For the occasion when we have 2 profiles we need to add a clearfix every 2nd profile. We only have 4 and don’t need to worry about the last profile so we just need to add div between David’s and Laurence’s column containers with the class of “clearfix”. We don’t want that div there for large or medium screens so using Bootstrap’s utility classes we can hide them – add the classes “hidden-md” and “hidden-lg” to the clearfix.
3. We also need to consider the occasion when we have 3 profiles on the top row. Add a clearfix at the appropriate point. We want this to be hidden for all grid sizes except the medium grid – so we could either use similar hidden classes as before, or we could just add a “visible-md-block” class.
4. If you’ve added these classes in the correct places you should now find that the rows below the profile you’re expanding/collapsing will move down/up the page as you do so – rather than jump left and right to move out the way.
5. As a finishing touch we should really indicate to our users that these are clickable areas. Open up the app.css file and add an appropriate selector for the containing element of each profile and set the CSS cursor property to ‘pointer’ and the box-shadow property to “0 0 5px 0 #005BAB”

Part 2 – Enhancing the navigation bar

In this part of the exercise we’re going to create several “course” pages and we want these to be reachable from the navigation bar.

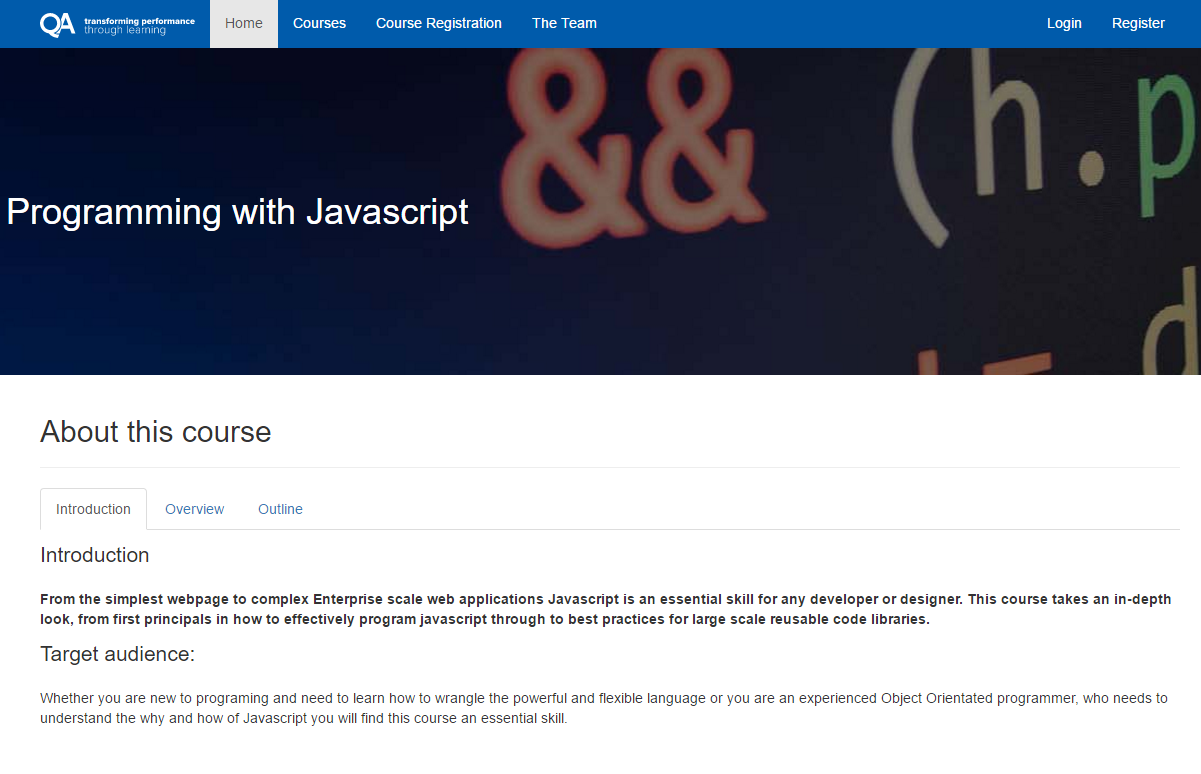
1. In the resources folder you’ll find four HTML files all with a similar markup structure. Copy qajavsc, qaangularjs, qawebcss and qaeffweb.html into the parent folder where your existing html files are.
2. Open index.html; we’ll work on our nav element in here and then copy the changes across the remaining html files on our site.
3. First of all find the list item that contains the link to courses.html; this is going to be where we’re working. Give it a class of “dropdown”
4. Give the anchor tag a data-toggle attribute with a value of “dropdown” and a class of “dropdown-toggle”.
5. We’ll want to tell our users this is a dropdown item and not another link, so insert a span with the class of “caret” immediately following the innertext of the anchor element.
6. Now to create the actual markup for the dropdown. Immediately following the anchor tag, still within the list item, create a new unordered list with a class of dropdown-menu. Create a list item and anchor tag for each course page.
7. Save and load your page – the navbar should look like this:
8. Remembering that all we’ve done is add extra mark-up into the original list item that held the link to courses.html, replace that list item in all the html pages that make up this website with this new list item – your navigation should then be seamless across the whole site.

Part 3 – Adding tabs to the course pages

If you open up any of the course pages we just finished creating our navigation for you will note that there are three discreet sections of content. One is for the Introduction, another for the Overview and another for the Outline. Currently these are all presented straight up on the page – we’re going to split each section into tabs the user can navigate into if they’d like.

1. The first step to creating our tabbed layout is to create the wrapper. Below the row that houses our page-header create a new row with a col-xs-12 column inside. This is where our tabs will go.
2. To create the tabs themselves you’ll need an unordered list with the classes “nav” and “nav-tabs” and a role of “tablist”
3. Inside the list create three list items with a “role” attribute with the value of “presentation”
4. Inside each list item create an anchor with the following attributes and values:

|  |  |
| --- | --- |
| role | tab |
| data-toggle | tab |

1. Each of the anchor tags represents a link to the sections we’re creating tabs for, Introduction, Overview and Outline. As such we want the inner text of the three anchor tags to match up with those three sections. Make it so.
2. Now for the tab panes. Create a div with the class “tab-content” and within this create three divs with the class “tab-pane”. Each of these three divs will be the panels of the tabbed section. Give each DIV an ID that reflects the content they’ll be holding: Introduction, Overview or Outline.
3. Cut and paste the content into the panels you just created.
4. We now need to link the tabs to the panels, so go back to your anchor tags and give them all a href attribute that points to the ID of the panels you just created. For accessibility you should also add an “aria-controls” attribute which also takes the id as its value (without the # prefix).
5. Finally, give the first list item and the first tab-pane an additional class of “active” so they are the ones on display when the page first loads.
6. If you’ve done everything correctly, you should have a page that looks like the below and you are able to navigate through each tab.
7. Now that you know what you’re doing, why not go and give the same tab treatment to the remaining three course pages.