Bootstrap JavaScript Components

Objectives and Outcomes

In this module, we will be learning about Bootstrap's JavaScript components. These components have two parts, the CSS classes, and the JavaScript support. In this module we will learn about using the JS components without writing any JavaScript Code. This is possible using the data-* attributes that Bootstrap provides for us to control the components. At the end of this module, you will be able to:

Understand the various Bootstrap components that require JavaScript support in order to function
Use the data-* attributes that Bootstrap's JS API provides for you to control the JS components without writing a single line of JS code

1. Bootstrap JavaScript Components Overview

In the previous module we have seen a number of bootstrapped components that are supported through the CSS classes. In this module we'll look at the Java script supported bootstrap components. All these components that we examine this module will require a JavaScript plugin to be in place for them to work correctly.
0:28
As we have seen in the previous two modules, Bootstrap provides a rich set of CSS classes that can be applied to many HTML elements in order to create the Bootstrap components. These components, as we saw in the previous module, are created by just applying the Bootstrap's CSS classes. Now, you can use Bootstrap purely for its CSS classes and completely ignore the JavaScript plugins and the JavaScript based components in Bootstrap. But let's also examine how we can make use of the Java Script based components in Bootstrap in this module.
1:15
As we saw in the previous module, CSS classes for many components start with the base class, like for example the button class. Or from class or the table class and so on. But these classes can be further modified by applying additional CSS classes that belong to the same group of classes.
1:43
As an example, let's look at the button class. We can create a simple button by just applying the btn class, but we would want to further qualify it by giving it one of the standard colors. Perhaps even specify the size of the class. And also whether it should be a block button or not. So all these additional classes that belong to the button group of classes act as modifiers to the button base class.
2:19
Similarly, we have seen the nav class before, which was applied to URL and for creating a component within the nav bar. Be applied to the UL, the nav, and then the navbar-nav class too. In this module, we'll examine the nav class again, but here, we will apply the nav-tabs and the nav-pills class to create different kinds of navigational elements.
2:52
Now, let's talk about Bootstrap's JavaScript plugins themselves. Bootstrap's JavaScript support comes through a set of JavaScript plugins. These plugins are built upon Jquery, so Bootstrap's JavaScript competence depend upon Jquery to be present within your picture. That is the reason why, when we include in the Bootstraps, JS competence, we also include a Jquery under the pitch.
3:25
The plugins themselves have designs such that you can only include those plugins that you really need. If you want to include all the plugins, that is also feasible.
3:38

This graphic may help you to better understand the view of JavaScript, JQuery, and Bootstrap's components. If you look at JavaScript as the big heavy Sumo wrestler. Then JQuery is a condensed version of the sumo wrestler, more easier to handle, so to speak, than the full JavaScript. Because the JQuery library plugins that come in, are very very useful to achieve many interesting aspects within your web page. Now, what Bootstrap does is goes one more step further, and then builds upon JQuery plugins, and then boxes them in. Into tiny little components which can be used on their own.
4:26
You will pretty soon realize that you can use the Bootstrap's components without writing a single line of JavaScript code. That's the advantage that you get with Bootstrap's JS components. But again, it comes with less flexibility, when you use it without writing JavaScript code.
4:47
How do you include the Bootstrap's JavaScript plugins in your page?
4:53
We've already seen that at the bottom of the page we would include these two statements to import that JQuery library, as well as the Bootstraps complete set of JavaScript plugins.
5:13
Now again, as I said, Bootstraps JavaScript components can be used without writing a single line of code. Now this is feasible because Bootstraps applies what they call is a first class API. With data-stash attributes, for example the data-toggle, the data-spy and so on. Many attributes that Bootstrap JS components define. Now you can apply these attributes to the HTML elements that you use within your web page directly. And without even writing a single line of JavaScript code, the JavaScript components are ready to go for you.
5:53
Of course, they won't have the flexibility that you might get with directly writing JQuery code to control this components. But much of the functionality is very much available through the data star attributes. This is the approach that we're gonna explore in this module. We look at them all full featured JavaScript API in the next module.
6:27
In order for you to use the full fledged JavaScript API you need to be aware of the JQuery syntax and have enough knowledge of JavaScript.

Tabs and Tabbed Navigation

Objectives and Outcomes

In this lesson, we examine tabs and tabbed navigation. Tabs require Javascript support to be enabled for navigating the content. At the end of this lesson you will be able to:

Design a tabbed navigation for your content
Use tab panes to organize the content and navigate the content using tabbed navigation

2. Tabs, Pills and Tabbed Navigation

We will now talk about the use of tabs, pills and tabbed navigation to enable presentation of content to the user.
0:11
Obviously on any website there is a lot of content that can be presented to the user. But if we organize it in a way that is easy to navigate the content, then the visitors may find it more easy to find the information that they want. So, tabs and pills allow us to organize the content appropriately. Now, tabs

and pills provide the navigational elements for us. And then the content itself is enclosed in tabbed pills, as we will see when we go through this lesson.
0:46
Let's take a look at a simple example of how we can use tabs and tabbed navigation to organize content. So in this case we are looking at the information about the corporate leadership of our restaurant. Now the content which was listed all together has been organized into tabs indexed by the name of the corresponding leader here. So using the tabbed navigation we can easily move between the various descriptions of the different leaders using the tabbed navigation provided for us there.
1:25
This enables the visitors not to be overwhelmed with a lot of content but instead see the content presented in an organized manner.
1:37
So as we see, the tab and pills provide us with the navigation elements to be able to navigate the content that is already organized into multiple panes.
1:50
Let's now look at a simple example to understand how we go about implementing tabbed navigation on our page.
1:59
To implement tabbed navigation we're gonna use a UL with the classes nav nav-tabs. So this is a nav element, as we see. So that's why we use the nav class. And then the other qualifier is the nav-tabs class there. Now within that, each list item will be presented as a tab. So you can see in this code here the different list items presented are laid out on the screen as tabs seeing the example above.
2:35
Each of these tab elements carries a a tag with an href which we can use to defer to the tab pane that we're going to include next with the actual content of the tab.
2:50
Also, one of the list items will be marked as active. This means that that particular tab will be opened by default when the user navigates to this location on the page.
3:04
Instead of using the nav tabs class, we can use the nav pills class. The nav pills class turns the navigational elements into a list of items and the item marked as active will be clearly highlighted on the page there. So you can see that since I applied the active class to the first element, the corresponding a tag has been clearly marked out as a pill on the script.
3:40
But both nav tabs and nav pills provide exactly the same functionality. Depending on how you want to design your user interface, you may choose one or the other as the approach for designing your navigational elements.
3:57
To the navigational elements, we can also apply the nav-justified class. When you do that the nav elements will stretch the entire width of the parent element there and will occupy a proportional portion of the screen width for these nav elements. Now elements within the nav class can be disabled by using the disabled class applied to that particular element.
4:32
Pills, in addition, provide you with a way of stacking them vertically rather than horizontally. Nav tabs are always presented in a horizontal arrangement. Nav-pills, on the other hand, can be either presented horizontally, or if you use the nav-stacked class together with nav-pills then the items will be stacked vertically on the screen. We will see the use of nav pills in the next lesson.
5:04
The actual content that

5:07
forms the part of the tab will be enclosed inside a tab pane. So you will have a div with the class tab-pane and inside that the content will be presented. Now these tab panes are grouped together inside an outer div with the class tab-content.
5:27
And one of those tab panes will be designated as the active tab pane, which means that the content from that tab pane will be visible on the screen when you first reach that tab item there.
5:44
Going back to our example that we presented earlier, let's look at the actual code used for constructing those nav tabs. So here I present the UL class that enables me to construct the nav tabs. And also the tab content itself is enclosed inside a tab pane there, and
6:06
these tab panes are enclosed inside an outer div with the class tab-content.
6:14
One more point to note is that we need to connect the tabbed navigational elements to the content page. To do that for each of the tabbed panes we will give an ID, and the same ID will be used in the list item in the UL class to refer to this particular tab there. So in this example you can see that I have used the IDS Peter for this tab pane, and there comes warning list item in the UL class uses the href to Peter there. In addition, for the tabbed navigation elements to work correctly, they should be given the data attribute data toggle equal to tab there.
7:08
Now we will move on to our next exercise where we will actually construct the tabs example that you saw earlier. So we're going to go into the aboutus.html page, we're gonna use tabbed navigation to build the tab navigational elements there, and then we're gonna organize the content inside tabbed panes, and enclose that with a tab content div there.

Exercise (Video): Tabs (沒看)

Exercise (Instructions): Tabs (沒看)

Hide and Seek

Objectives and Outcomes

In this lesson we learn about the collapse javascript plugin that allows us to hide and reveal content. In particular we explore its use in creating an accordion. We subsequently explore the Scrollspy plugin in order to highlight the nav elements based on the scroll position. We also use the Affix plugin to first let the nav element scroll with the page, but get fixed at a position on the page after part of the scrolling of the page. At the end of this lesson, you will be able to:

Use the collapse plugin to hide/reveal content
Construct the accordion using panels and panel-group class
Use the scrollspy to reflect the current position of the scroll
Employ the affix plugin to fix the position of the nav element after an initial scroll

3. Collapse

We will now talk about the collapse plugin, which is one of the Bootstrap JavaScript components. We

will see how we can make use of the collapse plugin to show and hide content on our web page. We will also examine how the collapse plugin has been leveraged with our nav bar, in order to collapse the nav bar when you see that
0:28
page on extra small screens. And then reveal their nav bar items upon clicking of a button. We'll then see how we can make use of the collapse plugin together with the panel
0:44
class and the panel group classes to construct an accordion.
0:52
The collapse plugin provides a very quick and easy way of showing and hiding content on our web page. To toggle the show and hide behavior of our content we need to make use of either a button or a link, which, when clicked, will toggle the content. Now we'll look at a simple example of the collapse plugin in use and then review the code thereafter. Taking a look at this simple example, you can see that we have some content displayed on the left side here, and then a button on the right side. So when you click on the button, the information will be either revealed or hidden. Right now, we start out with the information being shown on the page. When you click on the button, the information is hidden and then when you click on the button again, the information is shown. So this toggling of behavior is enabled by using the collapse plugin. Let's now review the source code for the collapse example that we just saw. So in this case, this collapse example is implemented by using a div with a class collapse and in applied to it. So we have two classes there. Collapse class and in class, together being applied to this div. So this is the div that contains the content that is going to be either shown or hidden by clicking on the button. So the next part is the button itself, which is going to trigger this toggling of the showing and hiding of the content. So this toggle button is implemented by using a standard a tag with the button class as btn-primary, as you see in the example there. In addition, for the toggle to take place, we have a data attribute called data-toggle. And then the data-toggle with the collapse attribute. And the href of this a tag will refer to the ID of the div which contains the content that should be shown and hidden upon clicking of this button.
3:17
So, together with these, this code implements the collapsing and hiding of the content for this simple example. Returning to the navigation bar that we have seen in the very first week, you'll recall that on extra small screens, we had a button on the right side which when clicked would either reveal or hide the content. Now, this action was actually implemented using the collapse plugin there. At that point, I mentioned that I would come back to this in one of the later modules. So now we will go back and reexamine the code that implemented this collapse behavior and then understand how we are leveraging the collapse plugin to enable this behavior. Looking back at the source code that we had seen earlier, you can now see that the button has been implemented there, with the data-toggle as collapse and the data-target pointing to the navbar. If you recall from the week one module where we explained navigation, I had pointed out these two items there. Now, having known how the JavaScript components operate in Bootstrap, it all makes a lot more sense. So with this data-toggle that we implemented there, data-toggle collapse and the data-target is navbar. Now the whole navigation element, the ul was enclosed inside a div, which was given the class navbar-collapse and collapse. So what happens is that when you view this web page on an extra small screen, by default the navigation bar containing the navigation elements is collapsed. Because this div has only been given the collapse class there. It has not been given the in together with the collapse class. If you give the in class together with the collapse class the div will start in the shown mode there, which means that the content will actually be shown on the screen. So since we avoided the use of the in class, the content will start in the collapsed border. When the button is clicked, the content will revealed. And when the button is again clicked, the content will be hidden. So that is how the navigation bar worked for the collapse plugin that we used earlier.

5:53
So that is how the navigation bar with the collapse plugin together worked for the extra small screens.
6:03
Going to back to our about our aboutus.html page. Now I have replaced the entire tabbed navigation that I had for the corporate leadership information earlier, using an accordion. The accordion works by using a panel group together with a set of panels that enclose the content. We have already seen the panel component in Bootstrap in the previous module. So let's see the accordion in action. So, when you look at the web page, you will see that the web page starts out with that first panel containing the details of the CEO being shown, and the remaining panels being hidden. Now, if you click on any one of the other panels, then the result is that that particular panel gets revealed, and the remaining panels get hidden. So you can see the behavior with the remaining panels too. So you can click on any one of those panels heading and then the contents of the panel will be revealed, while the contents of the remaining panels will be hidden. So this is the accordion behavior that I was talking about. Let's now quickly look at how we implement this accordion behavior using the panel group and the panels.
7:27
Going to the source code, we now see the code for the accordion shown on the left-side and the accordion itself as seen on the web page shown on the right side. So when you look at the code on the left side you see that we have enclosed the entire content in an outer div which has been given the class as panel-group, and has been given the ID as accordion there. And inside there, individual panels are enclosing the actual content. So, the individual panels, as you can see here, has a panel-heading and the panel-body down below here. In the panel-heading, you can see that I am using the h3 class with the panel title, and then inside the h3 class, I am including the name of the corporate leader. And this name is enclosed inside an a tag with the role button and the data-toggle given as collapse. So, now you can see why clicking on that link or corresponding to the corporate leader opens and closes that particular panel content there. So, the data-toggle and then the href, obviously, pointing to the panel itself down below. In addition, I am also giving another data attribute called data-parent with the accordion. So, this is declaring that this panel forms
9:11
one of the panels in the panel group there with the id accordion. So this is why the data-parent is given the id as accordion, which is the id of the panel group there. Now, the content itself is enclosed inside a panel-collapse class with a collapse and in here. Now this means that this particular content will start out by being shown on the screen. So that is why on the right-hand side, you can see that the CEO's panel starts out as open, but the remaining panels we will not use the in class for the remaining panel collapses. We will only use the collapse class there. And note also that the ID is given as Peter, which corresponds to the href that you gave for the a tag earlier there.
10:05
Inside this panel-collapse, we are enclosing the actual content, the p element that we had, with the description inside a inner div, with the class panel-body. So the outer div enables the collapse behavior to work. And the inner div actually contains the panel-body, which is displayed on the screen on the right side. So, when you click on the name of the corporate leader, the corresponding panel is either hidden or revealed. Because of the enclosing of all these panels inside a panel group with the id accordion, and the data-parent attribute specified as accordion for each of the panels. When you click on any one of those panel headings, that particular panel will be revealed and will also result in the remaining panel bodies being collapsed correspondingly. So, this is what implements the accordion behavior that you see on the web page.

4. Scrollspy and Affix

Let us now understand the Scrollspy and Affix plug-ins that are available within Bootstrap. The

Scrollspy plug-in is applied to a nav element, such that as you scroll your web page, whatever is currently visible on the screen will be highlighted in the nav page. So that the corresponding nav element In the nav bar, or in the nav item that you display, would be highlighted on the screen. It is best for us to go and look at an example to understand this behavior before we come back and examine how we would implement the Scrollspy and the Affix plugin. These two go together in some sense, so that's the reason why I'm covering the two together.
0:57
To understand these two plugins, let's go to our bootstrap web page itself. Getbootstrap.com and I am right now under the Javascript part of the bootstrap web page. You can see that on the right side, we have a nav bar here. This must have been implemented using nav-pills. So as you scroll, you can see that the nav bar on the right side, scrolls up.
1:33
As you scroll the page, but when you hit the top, notice the subtle change in the behavior. When the scroll goes up to a certain point, the nav bar suddenly
1:49
jumps up and then gets itself fixed to a location. As you scroll now you can see that the item that you're viewing on the webpage gets highlighted in the nav items there.
2:04
And as you keep scrolling, the individual items get highlighted in the nav bar there. So scrolling more quickly, you can see that at every single point the corresponding item in the nav is being highlighted by changing its color on the right side.
2:29
Scrolling further you can see that behavior again one more time. When you scroll back, you will see that the plugin again jumps back and then starts scrolling back with the page at some point. Now, these two behaviors are implemented using the Scrollspy and Affix plugins. The Scrollspy plugin tracks the current scroll position of the web page, and then reflects that by highlighting the appropriate nav item in the nav element there. The Affix plug-in enables your nav element to move with the scroll up to a certain point and when it reaches a certain point, then the Affix plugin will fix the position. This is done by changing the position CSS property of that element from relative to fixed position.
3:32
The Scrollspy plugin is enabled as follows. We need to apply certain data attributes to an element of the webpage which we want to track. Now Scrollspy expects that this element of the web page will have a position that's relative. Now this is easiest done by assigning this to the body element of our web page, our body tag of our web page, because the body tag by default gets the position . So which means that the Scrollspy will be able to track the scroll position of the inner elements inside the body tag.
4:18
To do that, we are going to assign to the body a data spy attribute with scroll. And then I gave the data target pointing to the ID of the scroll nav element that we're going to introduce later. So in this example, I am giving the nav element a an ID as myScrollspy. And so that’s why I’m giving you the data target. And the data offset is 200, which means that the actual tracking of the scroll position begins 200 pixels from the top. So this is the offset that we give, essentially to avoid tracking the upper 200 pixels. Recall that when you look at your web page, the top part of the web page contains the jumbotron. We are not going to track the part of the jumbotron, we want to track the rest of our content of the webpage, the actual content in our webpage. So that's why I have given it a data offset of 200.
5:23
Now, to apply the Scrollspy itself, I need to introduce a nav element into my webpage. To this nav element, I introduce the property classes, but, in particular, I give it an id, which is the same as the ID that I use in the data target in my body tag. Now inside that, we will use the UL class with the nav and in this example I am using the nav-pills and nav-stacked classes there. So this will create the nav

element as a nav-pills element with the pills stacked vertically, one on top of the other. Now, inside here, of course, I have the list Items here each corresponding to the various content rows that I have in my webpage.
6:21
For each of those rows I need to give the corresponding id that I am using in the href attribute for these a tags here.
6:31
Let us now see this Scrollspy in action on our webpage. So I have used this inside my aboutus.html page. So you can see that I have the nav element on the right side, which is a nav-pills stacked. So as I scroll the page, you can see that the nav element scrolls with it. I'm also using the affix here, so I'm going to explain the affix part a little bit later. So as you scroll, as our row comes into our view there, the corresponding item Is being highlighted in my nav element. So I can see that currently, I'm viewing the History. So as I scroll to see the Corporate, that will be highlighted. And then as I scroll past, and then the Facts & Figures come up, that item will be highlighted in the nav page. Scrolling backwards, you see the same behavior one more time.
7:35
Now let's look at how the affix part of it works. So for the affix plugin to work, you see that for the ul class I have given it the data spy attribute as affix and then I set the data-offset-top as 400. This means that as the scrolling happens, when my webpage scrolls past the 400 pixel mark from the top,
8:00
up to that point my nav element will scroll together with the webpage. At that point, what happens is that the affix plugin will change the class given to this ul from affix top to the affix class there. Now, when the change happens, the position CSS property of this nav element will be changed from position relative to position fixed. So at that point this nav element gets fixed on the webpage to a specific location. Now, when does that fixing get triggered? That is specified by the data-offset-top attribute that I am giving here. And then I am specifying the position is 400, so when you scroll past the 400 pixel mark, at that point that nav element gets fixed. How do you decide this? That you will have to look at the structure of your webpage and then decide at what point it is appropriate to fix that nav element into a given position.
9:16
So, again, reviewing, we see that the Affix plugin is triggered by using the data-spy in the affix, and the data-offset-top with 400 applied to the ul class that we saw. In addition, inside my style start CSS file I am going to add an affix class with the property as top fixed to 100 pixels. What this means is that my nav element is going to be fixed from the top position of the webpage, 200 pixels. So when I change the position property to position fixed, the location is 100 pixels from the top of the webpage.
10:03
Going back to our webpage, you see that as I scroll the webpage my nav element scrolls with it. But at a certain point, the nav element gets fixed there, so you can see the triggering of the fixing of the nav element there. That nav element now is fixed at 100 pixels from the top of the webpage there, and subsequently will remain fixed at that position when I scroll. When I scroll back again, when I reach the 400 pixel mark, then the affix class will be removed and affix top class will be fixed, will be given to this nav element. And then the position is changed to relative at that point, so that's why the nav element, again, scrolls with the web page.
11:00
This completes our discussion on the Affix plugin.
11:04
Now, in the next exercise, we're going to make use of the accordion, the Scrollspy and Affix, and then apply each to the aboutus.html page that we have been designing so far. We're going to enclose the corporate leadership content into an accordion and we're gonna add another nav element to the right

hand side to track the position of the scroll of our webpage. So we're going to make use of the Scrollspy for this purpose. And then we're going to that Scrollspy nav element, we're gonna apply the Affix plugin to fix the position when we scroll the webpage.

Exercise (Video, Instructions) (沒看)

此後省略每部分開頭的 Objectives and Outcomes

5. Tooltips, Popovers and Modals

Let's now talk about Tooltips, Popovers, and Models. These are three different ways of showing content to the users.
0:13
Earlier we had seen the use of the collapse
0:17
Plugin where we were able to show and hide content on the webpage. But in those applications
0:27
we had to have the content already embedded in the webpage. Tooltips, popovers, and modals are the way of overlaying content onto your webpage.
0:42
In terms of flexibility, tooltips are the simplest, but provide very little flexibility. Popovers provide more flexibility than tooltips. Modals are, of course, a comprehensive way of presenting information to the users. So we'll examine each of these In more detail, next. To understand how tool tips work, let's visit our web page. And, then when we hover onto the reserve table button, you can see that a message pops up on our web page.
1:19
This is enabled using a tooltip.
1:26
Tooltips are a simple way of presenting some information to the users. A tooltip pops up whenever the users hovers onto a button or a link to which the tooltip has been added. To add a tooltip to an a tag or a button tag you will add three attributes.
1:47
The first one, data toggle is equal to tooltip
1:52
specifies that the tooltip has been added to the bottom tag. The second attribute, the title gives the message string that'll be shown inside the tooltip. The third attribute is data placement, which is either top, bottom, left, or right. And this specifies which direction the tooltip is going to pop up.
2:18
For the tooltip to work correctly, you will also need to include a small amount of JavaScript code at the end of your page. Right after where you import the bootstrap JavaScript classes, you will include the script shown here. This script specifies that for all those
2:44
tags to which the data toggle tooltip attribute has been added, the tooltip will be enabled.
2:52
Looking at the example of how you allow tooltip to a on a button tag. Here, I have ironed the tooltip to the a tag, by adding the data-toggle. And the data placement and the title to the a tag here.
3:10
This is what causes the tooltip to be popped up when you have it on to the button in our webpage.
3:19

Popovers, are a generalization of tooltips. Popovers allow you to present both a title as well as the content in the popover. The title will be presented in bold at the top, and the content will be included at the bottom of the popover. Popovers have to be also enabled explicitly by including the JavaScript code shown in there into our web page. So as an example this is what a popover will look like when you click on the button. So in this case we have associated the popover with this button here, by including the data toggle as popover. The title with the information over displayed in the popovers title, and the data content carries the information to be displayed in the content part of the popover. Now the popover also can take placement
4:18
attribute which specifies which side a popover should be shown.
4:25
In this example the popover will be triggered when you click on the button, unlike tooltips, which will automatically appear when you hover onto the button or the link. The popover has to be triggered by clicking on the button or the link. And next time when you click again on the button of the link the popover will be hidden. So the popover will remain open until you click again onto the button of the link.
4:55
Modals are a more gentle way of presenting information to the users. Modals allow you to define the information to represented in a lot more detail. You can even format the information and specify the information in the form of many HTML code. The models contain both a header, body, and a footer. Typically, you will include most of the information in the body of the message. The footer is usually used to include buttons, maybe cancel or an action button, okay? The header usually carries some information about the specific modal.
5:39
Let's look at an example and then come back and examine how we create a modal.
5:44
Going back to our web page, all right? You can see that there is a log in link on the right side of the Nav bar. When you click on it, a modal will pop up. So you see that I have designed this modal to include their log in form that previously existed in the nav bar. So I have moved the form into this modal here. So here, in this modal, we can see that there is a header and the body of the model contains the font. This particular example doesn't have a footer.
6:20
Now looking at the modal example, you see that the modal is created by defining a div, and giving it an ID, and the class to be given to the div is modal. If you want the modal to fade in when the button or the a tag is clicked, then we give the fade class also to the modal. Inside the modal, we'll have an inner div with the modal dialogue class. In addition you can specify the size of the modal as modal hyphen lg or modal hyphen sm. The default size would be invoked if you do not specify the size.
7:01
The modal also includes an inner div. The modal content. Now inside the modal content div you would include the header, body and the footer. The header will be another div with the class model header. The body will be a div with the class model body and then, of course, the footer will be a div with the class modal footer. Now, in this example, we are going to include the form HTML code inside the modal body of this particular modal.
7:37
How do you trigger the modal? To do that, you need to include in the information in either the a tag. A button tag. So to an A tag or a button tag you will give the data toggle as a modal, and the data target will be the ID of the modal to be specified. So in this example, the login modal's ID is specified as the data target. So that's why when the link, the login link on the and navbar is clicked, the modal pops up onto the webpage.

8:13
We'll now move on to the next exercise. While we're there we will consider tooltips and modals. So we introduce a tooltip so that when the user hovers onto to the reserve table button, the tooltip also pops up.
8:27
In addition, we'll have the login form enclosed inside a modal, and so when the user clicks on the login button in the nav bar, the modal will pop up on the screen.

6. Carousel

Let us now look at another interesting JavaScript component in Bootstrap called the carousel. The carousel allows you to present a slideshow on your webpage.
0:14
Let's take a look at an example and then we'll come back and consider the carousel in more detail.
0:21
Going to our webpage, you can see that I have already included the carousel component on the webpage. So you can see that the carousel presents a set of slides that will keep sliding from left to right on the webpage. So this is an interesting way of presenting content to the users.
0:42
So a carousel is an interesting component that allows us to present a slideshow on the webpage, as you saw from the example. So the slides will keep sliding from left to right with a time duration, typically around three seconds between slides. The carousel also provides manual control so that the users can navigate to any of the slides within the carousel.
1:08
We will look at all these in more detail next.
1:13
To add a carousel to your webpage, you will add a div with the classes carousel and slide and also the data attribute, data-ride, carousel. This is what turns this into a carousel control by the JavaScript plugin.
1:33
To add slides to the carousel, we will, inside the div that we have already introduced for the carousel, we will introduce another div with the class as carousel-inner.
1:46
And inside this div, individual slides will be identified by an inner div class item. One of those inner divs will be identified as an active, meaning that when you first go to the webpage, that particular slide will be presented to that user. So that is the first slide in the slide deck being presented to the user. Now, each item can include an image together with a caption. So inside the item div, you would include another div with the class carousel caption. So inside this inner div with carousel caption, you can include any HTML code. Typically, we would have a heading and some text included there.
2:36
Now, to introduce manual controls to the carousel, we can have two different kinds of controls. One control allows you to select the specific slide. Now, this is introduced to the carousel in the form of darts at the bottom, as you can see in the image there. You can select any one of those slides by clicking one of the unfilled darts. If you click on those unfilled darts, you will move to that particular slides there. The indicated controls are introduced into the carousel by including an ol with the class carousel controls, and the list items in there will contain two data attributes, the data target, which is the mycarousel, and then the data-slide-to, which will give the number of the slide to which you should move when you click on that particular item.
3:33

Carousel's also introduced left and right indicators to the carousel. So the left and right indicators can be shown as left and right arrow buttons on the two ends. In this case, we are introducing them using the a tags with the classes left carousel-control and right carousel-control.

3:57

These classes can include inside a glyphicon, so in this case I am using the left chevron and the right chevron as the glyphicon inside, so that these controls can be shown on the screen as the left and right buttons there. Clicking on these will manually take you to the previous and the next slide in the carousel slide deck.

4:23

Now we will move on to the next exercise. There we will introduce a carousel into our webpage in the index storage TML page. Now, we will also introduce manual controls to this carousel so we learn how to introduce manual controls and make use of them on our carousel.