"The Web Design Workshop" Homework 6 (5 pts)

In this homework, you will practice making UI components using JQuery. With JQuery, you can make interactive UI components like modal boxes, popovers, and carousels. Once you know JQuery enough, there's no need to use external libraries like Twitter Bootstrap; just make your own!

This homework is longer and worth more (5 pts) than other homework, so make sure you start early!

Part 1. Modal Box

Modal Boxes are used to present additional information on top of a page. Examples of a modal box are a detailed item view on Pinterest, add event view on Facebook, etc.

Step 1. Skeleton
Modal boxes are consists of two main components:

1. Modal Overlay
2. Modal Box Wrapper
3. Modal Box

Modal Overlay and Modal Box Wrapper are enclosed within Modal Container.

Modal Overlay is the semi-transparent black layer (div) that goes on top of a page. You need a modal overlay because you need to make the page dimmer than normal, in order to have the user's attention focused on the modal box, not the surrounding page.

Modal Box Wrapper is a div enclosing the modal box, with a sole purpose of vertically centering the modal box that it contains. Note that you cannot both horizontally and vertically center an element if you don't have another div containing an element.

Modal Box is the actual box that you are trying to show on top of a page. Usually, a modal box will be both vertically and horizontally centered relative to the page.

Question 1. Write the CSS properties for Modal Overlay and Modal Box.

Modal Container should:

1. Be full width and height
2. Have position: fixed with z-index of 100

Modal Overlay should:

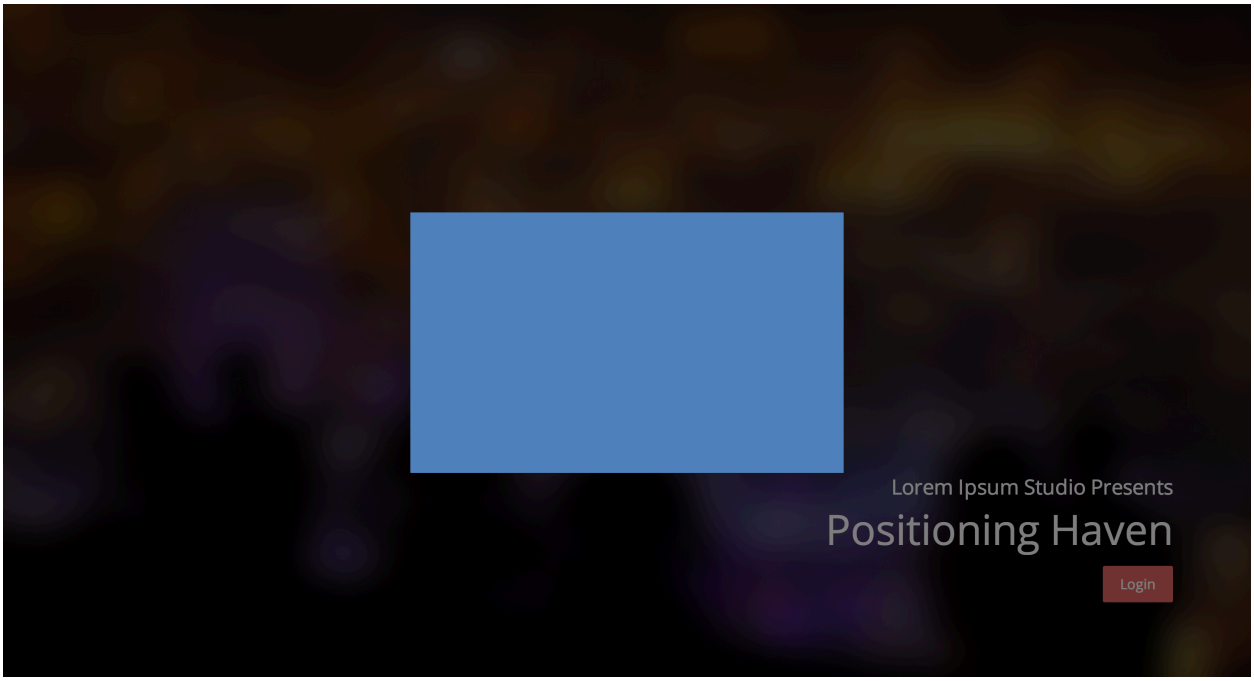
1. Be 50% transparent, black: background: rgba(0, 0, 0, 0.5);
2. Be full width and height
3. Should *be on top* of a page.

Modal Box Wrapper should:

1. Be vertically centered
2. Have height of 260px
3. Have full width
4. *Be on top of* a modal overlay

Modal Box should:

1. Be horizontally centered
2. Have width of 500px
3. Have height of 260px
4. Have a background color of #5485bb



Step 2. Content

Now that you have the skeleton of your modal box set up, it's time to populate your box with some content!

We will be creating a login modal. Our modal box will have three sections:

1. Modal Header
    1. Title (Should say Login)
2. Modal Content
    1. Enter username
    2. Enter Password
3. Modal Footer
    1. Submit Button
    2. Cancel Button

Question 2. Write the HTML code for the structure of the modal box described above.

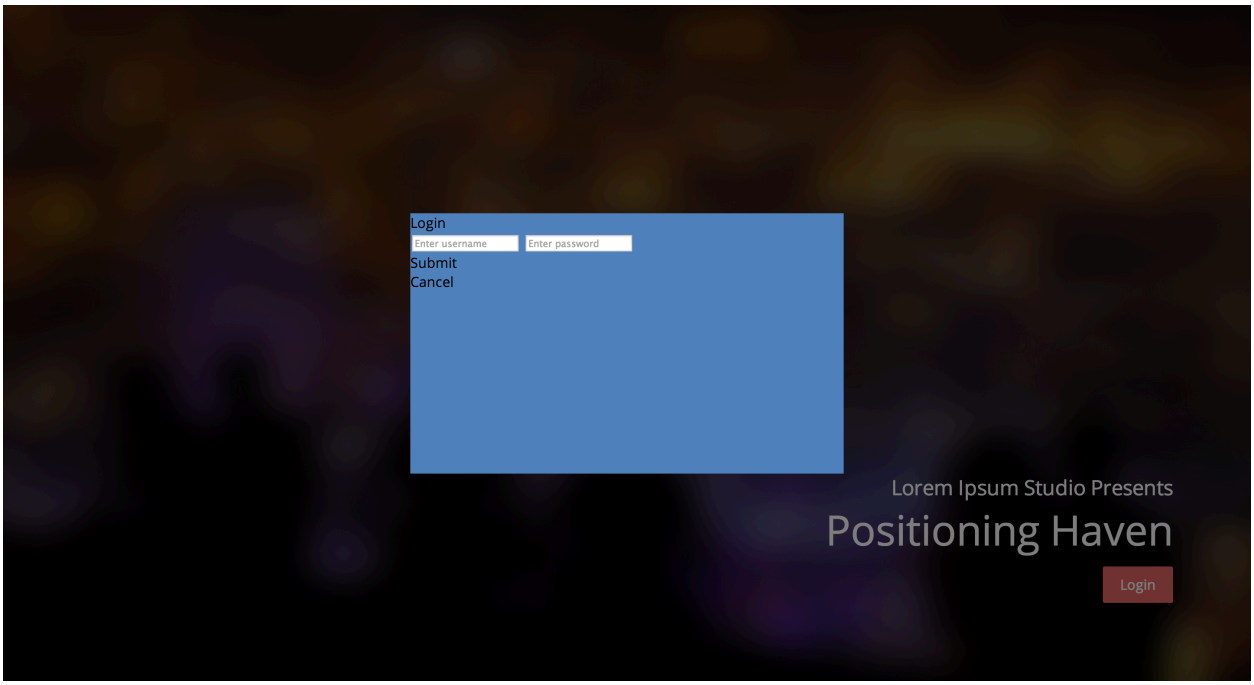
For username and password, you will have to take user's inputs. Therefore, you should use an HTML tag called <input>

This will create a text input:
<input type="text" id="SOME ID THAT YOU WANT" placeholder="Enter username" />

This will create a password input:
<input type="password" id="SOME ID THAT YOU WANT" placeholder="Enter password"/>

For submit and cancel buttons, you should stack buttons *horizontally*. Recall from the lecture that there are three ways to do this:

1. Float: left or right with clearfix
2. Position: relative for container and Position: absolute for children
3. Display: inline-block



Question 3. Write the CSS properties for the Modal box described above.

For your reference, we have included a screenshot of what your modal box should look like after the CSS styling. You should be able to style the title, submit button, and cancel button from what you have learned during the lecture.

You might need to adjust the height of your modal box and its wrapper depending on the paddings you choose.

The input boxes, however, can be a little tricky. Input boxes already have default styles set by the browser. In order to create a custom-style input boxes, we must first override the default styles. To do this, add custom CSS rule for your input boxes:

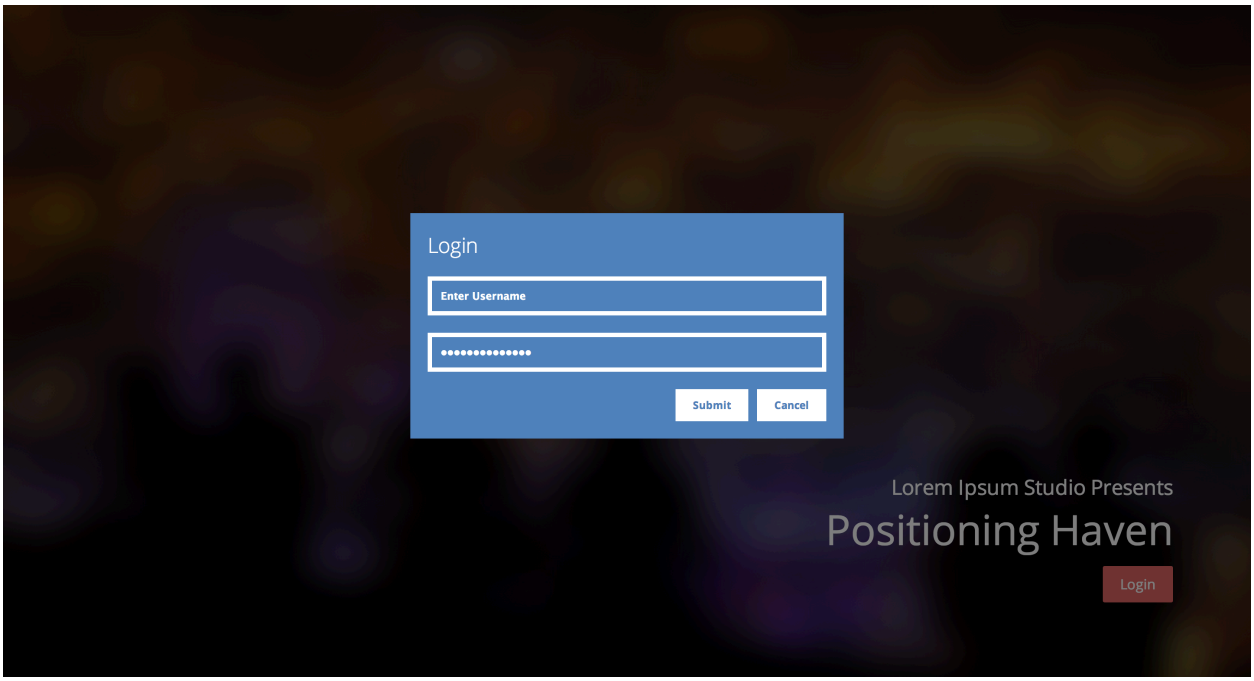
YOUR_INPUT_BOX {
border: 5px solid white; // Changes the style of the border
padding: 10px; // Adds some more padding
background: transparent // By default, the background is white. We want it to be transparent so that it has the background of the parent - #5485bb.
margin: 0px // By default, input boxes have some margins. We want to get rid of them.

margin-bottom: 20px; // However, we do want to have some spacing between each input box.
width: 460px; // Assuming we have a padding of 20px all around, the right dimension for your input box is 460px. If you have different padding, then adjust the width of your input box so that it fits your modal box.
font-size: 12px; // Overrides the default value of font-size
font-weight: 700; // Overrides the default value of font-weight to make it thicker.
color: white; // Overrides the default value of color to make the font color white.
}

To style your placeholder, you should use –webkit-input-placeholder property CSS selector:

#THE ID OF YOUR INPUT BOX::-webkit-input-placeholder {
YOUR CSS HERE
}

Don't forget to add "cursor: pointer" for your buttons so that your mouse cursor changes to a pointer like an actual link!



Step 3. JQuery

Now that you have created the looks of your modal box, let's apply some JQuery so that it actually does its job.

Here's how the modal box will work with JQuery:
1. When a user clicks on "Login" Button in the main page, the modal box will show up.
2. When a user clicks on "Login" or the "Cancel" Button, the modal box will go away.
3. Whan a user clicks on anywhere other than the modal box when the modal box is visible, the modal box will go away.

Before you proceed to question 4 and 5, add "display: none;" to your modal-container.

Question 4. Implement #1 and #2 from the scheme described above.

To get the modal box to show up, you should have the entire modal container to show up. This is because modal container contains both modal overlay and modal box, and modal box needs the modal overlay to establish the contrast that we need.

Similarly, to hide the modal box, you should hide the entire modal container.

Question 5. Implement #3 from the scheme described above.

#3 is a bit more complicated. There are three preferred ways to implement #3 using JQuery:

1. Event.stopPropagation();
2. JQuery not selector
3. Click event on modal overlay

For this exercise, we will be using the easiest method, click event on modal overlay.

The basic idea is this:
We want to execute the hide() function when a user clicks outside of the modal box. Since the modal box is on top of a modal overlay, click on modal box will not count as clicking on modal overlay. But since modal overlay covers the rest of the content, clicking on modal overlay is the same as "clicking outside of the modal box."

Therefore, to implement #3, simply execute the hide() function, similar to what you did for #2, but execute the hide() function when a user clicks on modal overlay.

Congratulations! Now you have a working modal box! Try experimenting with different designs of modal box if you have time. :)

Part 2. Carousels

A Carousel is a box that contains multiple images that switch upon a user interaction or a timeout. Here's an example of a carousel:
http://webdesigntutsplus.s3.amazonaws.com/tuts/342_bootstrap_carousel/Carousel-Files-COMPLETE/index.html

There are two ways to make a carousel:
1. Using position absolute/relative: A little trickier, but allows you to switch to next image even after reaching the last item.
2. Using overflow: Easier, but you cannot switch to next image after reaching the last item.

In this assignment, we will be using Method 2: overflow.

Step 1. Skeleton
Carousels consist of three main components:

1. Carousel Container
2. Carousel
3. Carousel items

Before explaining Carousels, let's first go over the CSS overflow property.
There are three values that are frequently used for CSS overflow:

1. overflow: visible – This is the default. If content goes outside of a div, the content will still be visible. For example, if you have a width 40px element inside a width 20px element, the inner content will be visible.
2. overflow: scroll – If content goes outside of a div, the content outside the boundary of the parent div will not be visible. However, you will be able to scroll to the content. For example, if you have a height 60px element inside a height 30px element, the content will initially be invisible, but you can scroll through the parent element and see the remaining element.
3. overflow: hidden – If content goes outside of a div, the content outside the boundary of the parent div will not be visible. You cannot even scroll.

Here's a detailed explanation of CSS overflow property: http://css-tricks.com/the-css-overflow-property/

For carousels, we have a carousel container with width 960px. Inside it, there will be 5 images. Initially, we can only see one image. However, when a user clicks on next, the currently visible image will scroll to the left and the user will see the second image. To do this, we structure the carousel like this:

1. Carousel Container with width 960px and **overflow: hidden**
2. Carousel with width 960px * 5 = 4800px
3. Carousel items with width 960px and float: left

Carousel itself will be width 4800px because there will be 5 carousel items (images) with width 960px each, floated left. If the width is any less than 4800px, then the 5 images will not be able to stack horizontally properly. If the width is any more than 4800px, then the div will have some empty spaces.

By setting overflow: hidden on carousel container, any element that goes outside of the carousel container will not be visible. Since we have a carousel with width 4800px, we can only see the first carousel item, and any remaining item will be invisible.

The height of a carousel container, carousel, and carousel items will be 518px.
The carousel container will have position: relative property because we will use it to position our prev and next buttons. Prev and Next buttons are already CSS-styled for you.

Question 6: Write the CSS properties for Carousel Container, Carousel, and Carousel items.

You should notice that your carousel gets cut off by the carousel container due to overflow: hidden property, like the screenshot above.

Step 2. JQuery

As of now, the user can only see the first image, and there's no way to see the remaining images. We will add some JQuery to make this work.

Here's our JQuery will work:

1. When a user clicks on the "Prev" button, carousel will *slide to the right,* revealing the previous element.
2. When a user clicks on the "Next" button, carousel will *slide to the left*, revealing the next element.

Now, the question is: "How can we achieve *slide to the right/left*" effect?

To do this, we simply adjust the margin-left property of the carousel (which has width of 4800px). It is important to note **that you can have a margin-left of negative value**. For example, if you have margin-left: 100px, the div will make 100px of spacing from its current left position, pushing it to the right. If you have **margin-left: -100px**, the div will make 100px of spacing toward the opposite direction, pushing it to the left.

To achieve slide to the left, we add -960 to the current value of margin-left. This will push the carousel by 960px (which is the width of each carousel item) toward the left.

Likewise, to achieve slide to the right, we add 960 to the current value of margin-left. This will push the carousel by 960px toward the right.

To get the current value of margin-left in an integer format, use:

parseInt($('#carousel').css('margin-left').replace("px", ""));

This will get the current CSS value of margin-left, take out the trailing "px" keyword, and then parses the string as a Javascript integer.

To achieve the "slide" effect, we will use JQuery animate() function. animate() is just like css() function, but it animates the transition, instead of just adding the css value. For example, if you set margin-left to -960px for an element with margin-left of 0px, then animate() will increment the margin-left from 0px to -960px for 500 milliseconds.

You should also make sure that you have not reached the start or the end of a carousel. If you are on your first image and a user presses the "Prev" button, it should not slide to the right. To do this, add an if-else statement:

if (CURRENT MARGIN == 0) {
return false;
} else {

YOUR JQUERY HERE

}

If you are on your last image and a user presses the "Next" button, it should not slide to the left.

if (CURRENT MARGIN == 3840) {
return false;
} else {

YOUR JQUERY HERE

}

Note that it will be helpful if you store the current margin as a Javascript variable somewhere in your function.

Question 7: Implement the rest of JQuery for the Carousel slide effect described above.

You now have a homebrewed JQuery Carousel! Experiment with different sizes, especially for a full-screen carousel. What should be the width value of Carousel container, Carousel, and Carousel items if you are trying to make the carousel full-screen? Of course, this question is entirely optional. ;)

Submission

To submit, send the following files to submission@thewebdesignworkshop.co

1. A zipped copy of hw6 folder

Please don't change the folder name! ☺
This assignment is due on Thursday, April 17th at 6PM.