

“The Web Design Workshop”– Spring 2014

jQuery

Goals:

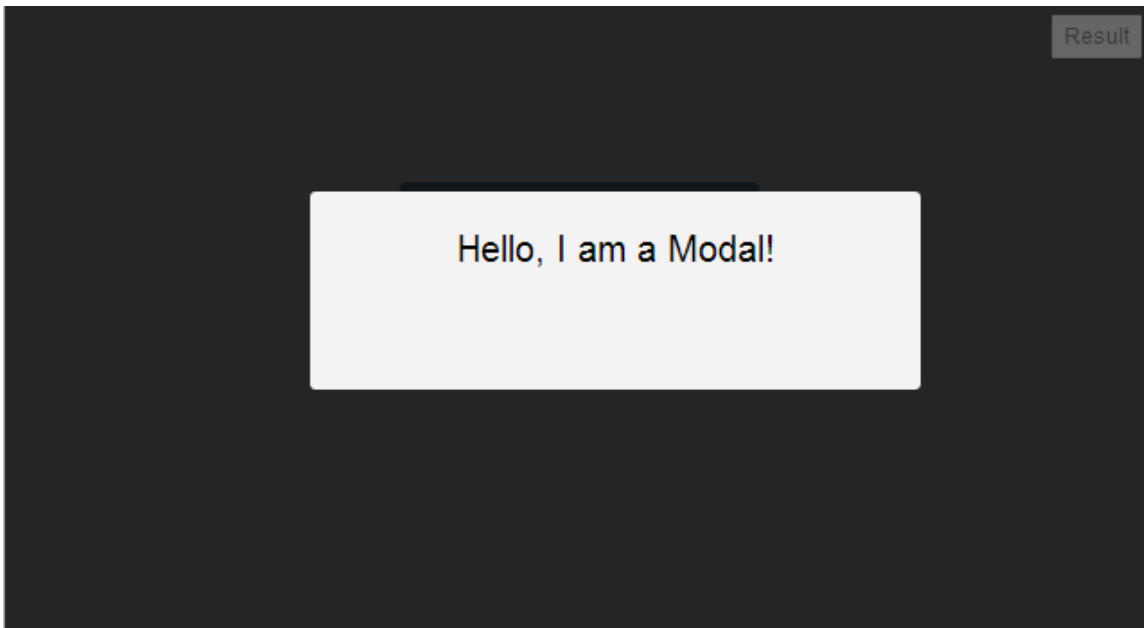
1. Creating a Modal
2. Toggling a Dropdown

1 – Creating a Modal

1. Go to <http://jsfiddle.net/H5EQf/2/> . We have set up the HTML and CSS for you. When opening the page, this is what you should see on the bottom right corner:



Our goal will be to apply jQuery to this button so when clicked, we get a popup modal, like so:



2. Before we start, take a look at the HTML element `#black`. Notice in our CSS that `#black` is currently set to **display: none**, and has a background color of black (that's semi-transparent). This black overlay is what you see in the image above, and is what we will use to cover our entire screen when a modal appears.

Also notice that `#black` is fixed, width and height set to 100%. This is to black out our entire window.

Inside `#black` we have an element `.modal`.

HTML	CSS
<pre>1 <div id="black"> 2 <div class="modal"> 3 Hello, I am a Modal! 4 </div> 5 </div> 6 <div id="button"> 7 Click to Toggle Modal 8 </div></pre>	<pre>25 #black { 26 background-color: rgba(0,0,0,0.85); 27 box-shadow: 0 0 5px #333; 28 -moz-box-shadow: 0 0 5px #333; 29 -webkit-box-shadow: 0 0 5px #333; 30 cursor: pointer; 31 display: none; 32 position: fixed; 33 width: 100%; 34 height: 100%; 35 top: 0; 36 left: 0;</pre>

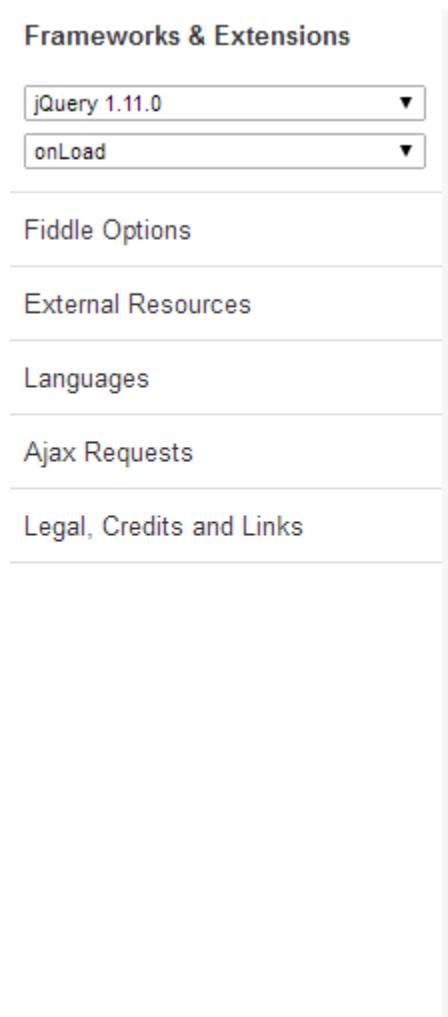
Our goal is to display this `#black` element (which will also display `.modal`) when we click `#button`.

3. First, go to your JavaScript box (bottom-left) and type the following:

```
1 $(document).ready(function() {  
2   |  
3 });
```

Remember that before we can apply jQuery events like **.click()**, we must tell our browser to use jQuery, with this **\$(document).ready(...)**

Also notice that on the left sidebar, we have to explicitly tell jsFiddle we want to use the jQuery library:



The image shows the left sidebar of the jsFiddle web application. The 'Frameworks & Extensions' section is expanded, showing two dropdown menus. The first dropdown is set to 'jQuery 1.11.0' and the second is set to 'onLoad'. Below this section are links for 'Fiddle Options', 'External Resources', 'Languages', 'Ajax Requests', and 'Legal, Credits and Links'.

Note:

When you are working in your text editor, remember to add the **<script ...></script>** tags for the jQuery library too before you even start a **\$(document).ready(...)**

4. Now, let's add a click event for our button!

First, select your button element with `$('#button')`. Then, apply the `.click()` event to the button. You will get the following:

```
1 $(document).ready(function() {  
2     $('#button').click(function() {  
3  
4     });  
5 });
```

JavaScript

Remember to add a **function() { }** inside your click event! Always attach a function to your events.

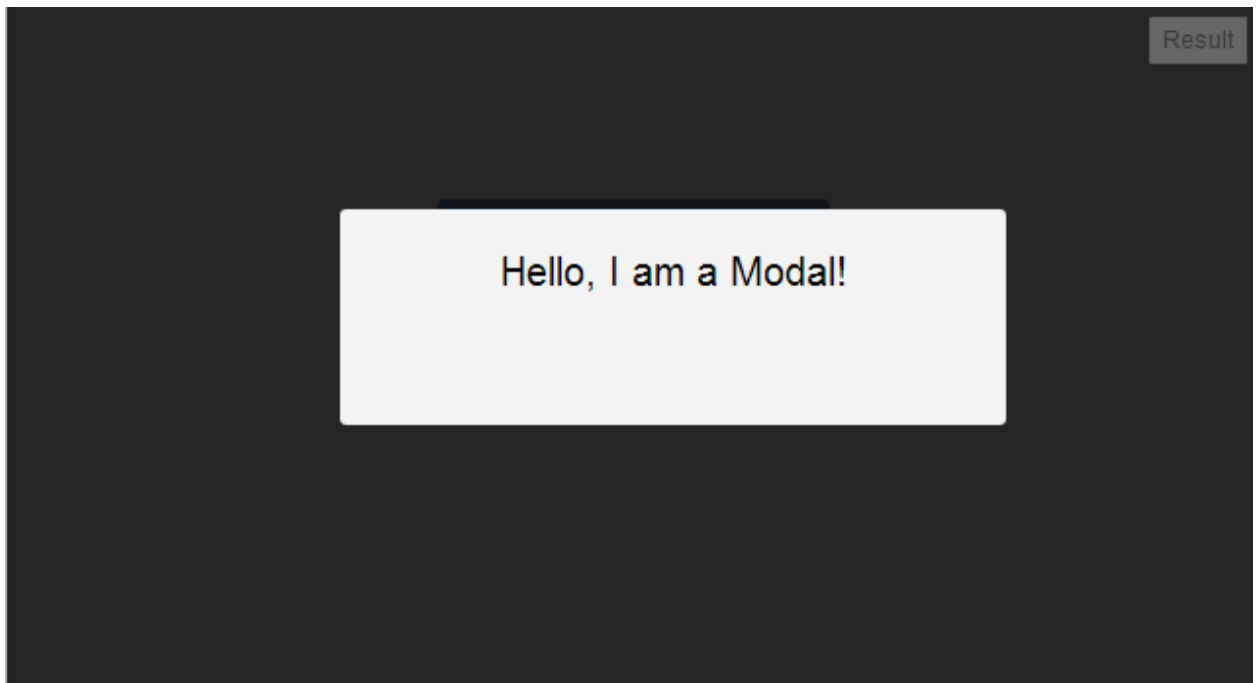
5. Alright! So you attached an *event* to an element! Now, all you need to do is to display *#black*. Do you remember what jQuery effects we can use?

We could do `.show()`, but that will display the modal instantly. Let's make the modal fade in gradually at 400 milliseconds.

```
1 $(document).ready(function() {  
2     $('#button').click(function() {  
3         $('#black').fadeIn(400);  
4     });  
5 });
```

JavaScript

6. Fantastic! If you try clicking the button now, you should be presented with the modal!



7. Whenever you code, always consider two-way actions. For example, in this case you brought up the modal by clicking the button. To get rid of it, what should you do?

To make things simple, let's click anywhere in the *#black* element (which is the entire screen) and this will make the modal disappear. Essentially, this will reverse our previous jQuery code.

You should get this:

```
1 $(document).ready(function() {  
2     $('#button').click(function() {  
3         $('#black').fadeIn(400);  
4     });  
5     $('#black').click(function() {  
6         $('#black').fadeOut(400);  
7     });  
8 });|
```

JavaScript

Notice this time, we are fading out *#black* instead of fading it in. Also notice we bound this click event to *#black* instead of *#button*.

(Note: For **`$('#black').fadeOut(400)`** we could also write it as **`$(this).fadeOut(400)`** since the element being clicked and faded out are the same)

Go on and try testing it!

8. Great job! We just learned how to do a simple modal! 😊

Bonus for the Curious Students

1. You may notice that because *.modal* is inside *#black*, even if you click *.modal* when it appears the *.modal* will fade away. Why?

Simple: Because *.modal* is inside *#black*, and we bound the **.click()** event to *#black*. How would you make it so that if you click *.modal* nothing will fade away, only if you click the black areas surrounding it?

We will go over this next week with DOM manipulation. But, there is a jQuery function called **.stopPropagation()** that we can apply when clicking *.modal*.

```
1 $(document).ready(function() {  
2     $('#button').click(function() {  
3         $('#black').fadeIn(400);  
4     });  
5     $('#black').click(function() {  
6         $('#black').fadeOut(400);  
7     });  
8     $('.modal').click(function(e) {  
9         e.stopPropagation();  
10    });  
11 });
```



Originally when you click *.modal*, the browser will see if *.modal* has any event bound to it. If not, it goes to its parent element *#black* and checks to see if *#black* has any event bound to it, which it does (**.click()**). Thus, the browser will respond with *#black*'s click event!

However, if you apply a **.stopPropagation()** to the *action* of clicking *.modal* (we call this **e** in the picture above, which I shorten for **event**), this will stop the browser from applying *.modal*'s parent's events (such as **\$('#black').click(...)**). This **e** is very special, and has several functions that you can apply to it, such as **stopPropagation()**.

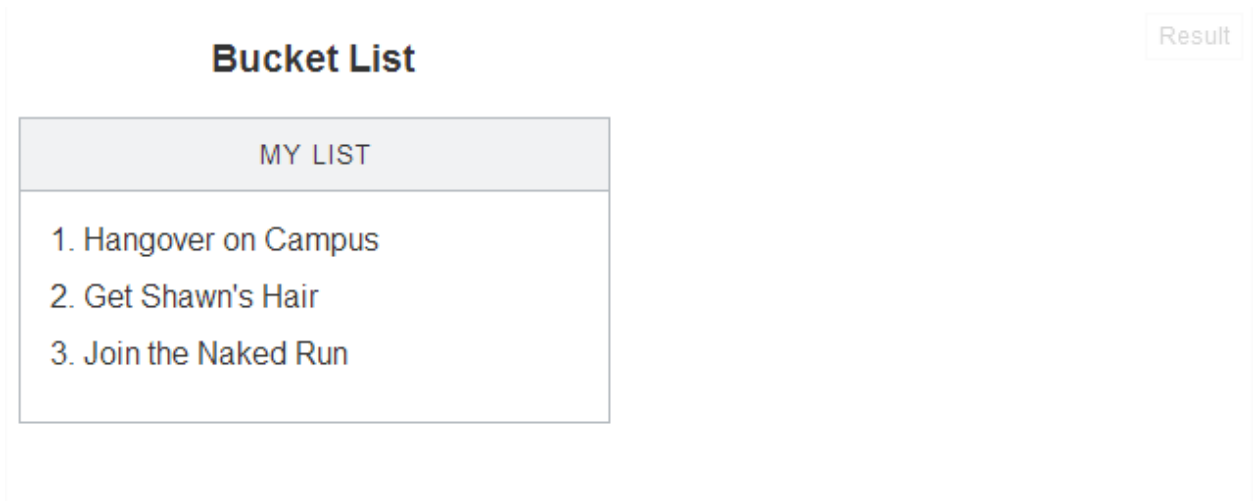
It's a powerful concept, and we will dive deeper into this next week, don't worry.

2 – Toggling a Dropdown

1. So you know how to do click events! Let's start **toggles**!

What is a toggle? It is doing some action when you click on an element, and doing a different action when you click on it again.

Our test subject will be located here: <http://jsfiddle.net/BY8b3/2/>



2. Our goal is to make it so that when we click the bar that says “MY LIST”, the bucket list will drop. If we click the bar again, the bucket list will hide.

The first thing you want to do is find **.panel .content** in the CSS and add a style of **display: none** so that initially, the bucket list does not show.

3. Next, like we did in the last example, we must tell the browser we are going to use some jQuery:

```
1 $(document).ready(function() {  
2     |  
3 });
```

4. Alright, let's apply a **.click()** event to our bucket list bar, called **.header**.

```
1 $(document).ready(function() {  
2     $('#header').click(function() {  
3  
4     });  
5 });
```

5. Next, we want our bucket list to appear. Like with modal example, we don't want it to appear instantly. Let's have it slide down at a speed of 200 milliseconds.

```
1 $(document).ready(function() {  
2     $('#header').click(function() {  
3         $('#content').slideDown(250);  
4     });  
5 });
```

JavaScript

6. Try clicking the "MY LIST" bar! Your bucket list should slide down.

Bucket List

Result

MY LIST
1. Hangover on Campus 2. Get Shawn's Hair 3. Join the Naked Run

7. Now, the difference between this example and the modal example is that we want to apply **two different actions** for the same div element (**.header**) instead of two div elements as in the modal example (**#black** and **#button**).

Thus, to make the bucket list slide back up requires new knowledge of jQuery functions that we have not talked about.

Let's preview them:

.addClass() – This will add a new class to your HTML element. So if our *.header* element

currently looks like: `<div class="header">My List </div>` , and we add some class say **clicked** (to indicate we clicked the *.header* and the bucket list dropped down), our HTML element will look like: `<div class="header clicked`

Notice the space between header and clicked. This just means our element has two classes attached to it.

.hasClass() – We will check whether or not our *.header* element has class **clicked**. If so, we want to slide up our bucket list and remove the class **clicked**. If not, we will slide it down and add the class **clicked**.

.removeClass() – When we click *.header* once more, this will slide up our bucket list and we will remove the class **clicked**. Thus, if we clicked *.header* a third time, it will do our first action again, which is to slide down the bucket list.

Put this altogether, and we get:

```
1 $(document).ready(function() {
2     $('.header').click(function() {
3         if($(this).hasClass('clicked')){
4             $('.content').slideUp(250);
5             $(this).removeClass('clicked');
6         } else {
7             $('.content').slideDown(250);
8             $(this).addClass('clicked');
9         }
10    });
11 });|
```

JavaScript

Woah! Longer right? It is definitely longer, but does what we need it to do. Try testing it out!

Note:

jQuery 1.8 and under used to have a way to do toggles, with the **.toggle()** event that took in not 1 but **2** functions, which is what we have just implemented with all this addClass, hasClass stuff.

Awesome job so far! ☺

Congratulations! Let's review what we learned:

- How to start jQuery with `$(document).ready(...)`
- How to apply select elements with `$(...)`
- How to bind an event to an element (`.click()`)
- How to do `slideUp/Down` and `fadeIn/Out`
- How to work a modal
- How to work a toggling menu/dropdown
- How to utilize additional classes to simulate toggling