Street Fighter: jQuery Events Walkthrough

Estimated Time: **1-2 hours**

In this assignment, you'll write the code to handle when the user hovers over Ryu. When this happens, you want Ryu to move to his "ready position". You'll use `.show()` and `.hide()` to accomplish this.

Open Your Files

Most of the this assignment will be in js/app.js, so open that file in Sublime Text. You'll also want to keep main.html open, so you can reference it by selecting elements for jQuery objects, so open that file, too. This is a good occasion to use View --> Layout --> Columns: 2 , like you did in the last assignment. Also open up main.html in Chrome so you can preview your page as you work.

Document Ready

Even before you start writing the code for the events, you know you want to wrap it in a `$(document).ready(function(){});` block, which will prevent the jQuery from executing before the page has loaded, so go ahead and put that code in app.js. Inside of that file (which should be blank at this point), put the following code:

```
$(document).ready(function() {

});
```

You'll be putting your event-related code within this function.

Event Strategy

When main.html loads, your .ryu-still div appears, and although .ryu-ready and .ryu-throwing will load as part of the DOM since they're in the HTML, they won't be displayed because you have set their display property to none in main.css. You need a way to register when a user hovers over and clicks on Ryu, and for this, you're going to take advantage of the fact that all of your Ryu image divs are contained within a single .ryu container. You want to know when a user is interacting with this .ryu container div irrespective of which Ryu image happens to be displaying. That means you'll want to `listen` for events on that div in particular, and `handle` the response.

Ryu Ready 1

You will now write the code that will move Ryu into his ready position. For this, you'll listen for a `mouseenter` event on the .ryu div, and handle it with a function that shows an alert. So modify app.js so it looks like this:

```
$(document).ready(function() {
  $('.ryu').mouseenter(function() {
    alert('mouse entered .ryu div');
  });
});
```

You have added the alert function call, so you can confirm that your event is firing as expected. Before you bother writing the code that determines what happens when this event fires, you should ensure that the event happens. Alerts or console.logs are a good technique for this. Save this change to app.js, then go to Chrome and reload the page. After the page loads, hover your mouse over Ryu, and you should get an alert with the message indicated.

What is that code actually doing? First, you have the `$(document).ready(function(){})`, which is an `event handler` that listens for the page to be ready, then handle's that by calling the function you passed in. Inside that function, you select the DOM element with the class ryu, and attach a new event handler to it `$('.ryu').mouseenter(function(){})`. This event handler is attached to every element with the ryu class, and the function you passed in, called a `callback function`, is stored in the browser memory. Every time the mouseenter event happens, the browser invokes that callback function from its memory in response to the event. You can think of event handlers like a cause and effect relationship. The function is the effect of the event (cause).

Ryu Ready 2

Now that you know your event is firing, you need to write the code that will hide the .ryu-still div and show the .ryu-ready div. You'll use the .show() and .hide() methods to achieve this. Inside of the mouseenter callback function, replace the alert with the following code:

```
  $('.ryu-still').hide();
  $('.ryu-ready').show();
```

This code hides div.ryu-still and shows div.ryu-ready, whose initial state was `display: none`. Back in Chrome, refresh the page, then move your mouse over Ryu. He should move into his ready position. Note, however, that when you move the mouse away from Ryu, he remains in the ready position, which makes sense, because you haven't written that code yet.

The browser is currently listening for mouseenter events on the div.ryu, but you want it to listen for mouseleave events as well. In situations such as these, you can take advantage of `method chaining`. Like before, confirm that you can get the mouseleave event to fire, then write the code that will move Ryu back to his still position. Modify app.js so it looks like this:

```
$(document).ready(function() {
  $('.ryu').mouseenter(function() {
    $('.ryu-still').hide();
    $('.ryu-ready').show();
  })
  .mouseleave(function() {
    alert('mouse left');
  });
});
```

Note: In the chained code, the `.mouseleave` method call is on a new line — this makes the code more readable. Also, note the because you want this chaining effect, the line before `.mouseleave` does not have a semi-colon, allowing the mouseleave to chain. In this case, it's good to think of the semi-colon as a way to end a thought. Here, the thought ends after the mouseleave, so that's where it should go. Save the file with the added method call, head back to Chrome, and refresh the page. Now, when you hover your mouse over Ryu, you should see the image change, and when you move your mouse away, you should get the alert if the event is firing correctly.

Now replace that alert with code that will make Ryu reassume his still position. Replace the alert() with the following code:

```
$('.ryu-ready').hide();
$('.ryu-still').show();
```

Save those changes, then go back to Chrome and refresh your page (are you starting to notice a pattern here?). Try moving the mouse over and then away from Ryu. When the mouse leaves, Ryu should now return to his still position.

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

✓ Mark as completed

Previous

Next