Introduction to jQuery

Relevant Links

- Flanagan's book, Chapter 19
- Flanagan's book, Part IV (jQuery summary around page 945)
- devdocs on jQuery[1]
- jQuery front page[2]
- jQuery API[3]

Notes

jQuery is a very popular library for handling the DOM parts of an application. It offers a uniform interface, not marred by the various incompatibilities between browser implementations.

jQuery has 3 main components that we will utilize in due course:

- Page / element access and manipulation
- event-handling
- asynchronous data loading

Even if you do decide not to use it, you need to become familiar with it, as many other libraries offer a similar interface.

As an external library, jQuery needs to be first incorporated into your project via a script tag (or the AMD module methods we will learn about later). You can either download the version you want and link to it, or you can link to the jQuery CDN[4] (the wikipedia page[5] is a good place to start if you want to learn more about Content Delivery Networks and their advantages). So the simplest way to get jQuery into your code is to include the following script tag:

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
```

You can also load jQuery on a web page that doesn't have it loaded yet by doing something like the following on the console:

```
var script = document.createElement('script');
script.setAttribute('src', 'https://code.jquery.com/jquery-3.1.1.min.js');
document.body.appendChild(script);
```

You can follow this along by using the sample found here[6].

jQuery defines two global objects, that you can use to access it. One is jQuery, the other is $. For all our purposes, they are synonyms and we will be using $.

[1] http://devdocs.io/jquery/
[2] http://jquery.com/
[3] http://api.jquery.com/
[4] https://code.jquery.com/
[5] http://en.wikipedia.org/wiki/Content_delivery_network
[6] ../samplePage/test.html

The global (factory) function $

The global $ is a swiss-army knife, it is a function that sort of does everything. It has three main forms, depending on what the first argument is:

$(sel) If the first argument is a string that is a selector, it will return a "jQuery object" that represents the collection of all elements that match the selector. If a second "context" argument is passed, representing an element or elements, it will search for the elements *relative to that context*.

$(html) If the first argument is a string that is html text, then it will return a new object representing that html code. This object is currently detached (does not show up on the web page), but it can then be inserted into some point in the web page specified by future commands.

$(domEls) If the first argument is a dom element or array of dom elements, it will return a "jQuery object" that represents that same array of elements, but now as jQuery objects.

$(f) If the first argument is a function, then this function will be called when the document has finished loading. This is a good place to do page initialization.

Examples

Let's assume that jQuery was loaded on the sample web page we were looking at last time. We start by locating the `ul` element on the sidebar that contains our navigation links:

```
var linkList = $('#sidebar ul');
```

We will now create a new element and append it to that list:

```
$('<li><a href="http://www.google.com">Google!</a></li>')
  .appendTo(linkList);
```

Next we pick out the second `li` from within the list, and move it to the bottom of the list:

```
var el2 = $('li', linkList)[1];
linkList.append(el2);
```

We will learn about these methods in the rest of this section.

jQuery object methods

All jQuery objects respond to a similar set of methods. You can explore the whole API online[7], and in Part IV of Flanagan's book.

First we have collection methods.

each(f)[8] Given a function `f(i, el)`, will call the function for each element (if a collection).

[7] http://api.jquery.com/

get(i), el[i][9] gets the element at the i-th index. jQuery objects tend to be collections, this picks out a specific element from that collection.
map(f)[10] Translates all items of the array to a new array, via the function.
filter()[11] Filters the list based on the first argument. See the documentation for options.
add()[12] Adds more elements to the current selection, and returns the resulting object.
children()[13] Selects the children of the current selection.
closest()[14] Returns the closest ancestor of each element that matches certain criteria.
find()[15] Selects all descendants that match certain criteria.

Next we have some element methods that read/write element properties.

addClass[16]**, removeClass**[17]**, toggleClass**[18] Add / remove / toggle classes to each element in the collection.

This can actually toggle multiple classes at once.

attr[19] Get or set one more more attributes.

data[20] Get or set data attributes.

val[21] Get or set the "value" associated with the element.

Next we have methods for inserting/removing elements from the dom.

after()[22]**, before()**[23] Insert the provided content after / before each element.
append()[24]**, prepend()**[25] Appends / prepends the provided content at the end of each element.
appendTo()[26]**, prependTo()**[27] Inserts each element at the end / start of the argument.
html()[28] Sets/gets the html contents of the elements.
insertAfter()[29]**, insertBefore()**[30] Inserts each element after the argument. It will also clone them if the target argument is many elements.
wrap[31]**, wrapInner**[32] Various ways or wrapping specific HTML around/inside the elements.

There are a lot of other methods, mostly related to events, that we will revisit later.

Examples

We will run some examples on a barebones page. Save this page in an HTML file, open it in the browser, then fire up the console, to follow along.

```
<!DOCTYPE html>
<html>
<head>
  <title>A basic page</title>
</head>
<body>
```

```
  <div id="mainDiv">
    Basic content.
  </div>
  <script src="https://code.jquery.com/jquery-1.11.2.min.js"></script>
</body>
</html>
```

Okay, now first off, let's type `$` in the console. We should see that it exists and is a function.

Try each of these lines in order.

```
var mainDiv = $("#mainDiv");
mainDiv.addClass("foo").append("<p>Hi! I'm new!</p>");
mainDiv.text();                              // Text in mainDiv
mainDiv.html();                              // HTML in mainDiv
mainDiv.html("Oops it's all gone!");    // Replacing the html
$([1,2,3]).map(function(i) { return $('<p></p>').html("Hi! I'm " + i)[0]; })
          .appendTo(mainDiv).wrap('<li></li>');
$('<input type="input" />').appendTo('li').val(20);
$('<input type="input" />').appendTo('li').val(function(i) { return i; });
$('li').first().css('background-color', 'blue')
```

Let's look at a longer example of a page.

```
<!DOCTYPE html>
<html>
<head>
  <title>A basic page</title>
  <style>
  * { font-size: 16pt; }
  .big { color: blue; }
  .small { color: red; }
  </style>
</head>
<body>
  <div id="mainDiv"></div>
  <script src="https://code.jquery.com/jquery-latest.min.js"></script>
  <script>
  $(function() { // Code here runs once page has loaded.
    var ul = $("<ul></ul>").appendTo("#mainDiv"), v;
    for (var i = 0; i < 10; i += 1) {
      v = Math.random();
      $("<li>" + v + "</li>")
        .addClass(v > 0.5 ? "big" : "small")
        .appendTo(ul);
    }
    $('li').each(function(i, v) {
      $(v).fadeOut(Math.random() * 5000, function() {
        $(v).fadeIn(Math.random() * 5000);
      });
    });
  });
  </script>
</body>
</html>
```