jQuery Part 1

IMY 220 ● Lecture 10

Using jQuery

Download the jQuery library.

Save it in the directory you are working from (a js or javascript directory in your project folder).

To use the library you just need to include it:

```
<head>
        <title> Hello jQuery world! </title>
        <script type="text/javascript" src="jquery-3.5.1.min.js"></script>
        <script type="text/javascript" src="script.js"></script>
</head>
```

The first script tag includes the jQuery library.

The second script tag is where you will be adding your jQuery code.

https://jquery.com/download/

Including jQuery

Using the Google CDN (Content Delivery Network)

Specifically designed to deliver content in a fast and scalable manner

Servers are spread out geographically so your SA client may be served from local bandwidth

The library is then not downloaded from your hosting but from Google servers.

```
<script
        src="https://code.jquery.com/jquery-3.5.1.min.js"
        integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
        crossorigin="anonymous">
</script>
```

Including jQuery

Local vs. hosted (CDN)

- See L4 - Bootstrap

Fallback method

Including jQuery

If you want the CDN to send you the latest version, leave out the number in the path:

Instead of 3.5.1 put 3.5 and the new release in the 3.5 series will be loaded.

Instead of 3.5 put 3 and the new release will be loaded.

Be careful using this method because the HTML and JS may need to be changed with a version update.

Compressed jQuery

Compressed and uncompressed copies of jQuery files are available.

The **uncompressed** file is best used during development or **debugging**;

The **compressed** file saves bandwidth and improves performance in **production**.

Anatomy of jQuery script

jQuery Alias

Including jQuery gives you access to a single jQuery function called jQuery.

The jQuery library is included in the jQuery namespace.

There is an even shorter way to access the jQuery library: use the $ sign.

You may want to stick to the jQuery call because when using a lot of different libraries, they may be fighting over the use of the $ sign.

selector	action	parameters
`jQuery('p')`	`.css`	`('color', 'blue');`
`$('p')`	`.css`	`('color', 'blue');`

jQuery Statements

Each jQuery command is made up out of 4 parts:

- jQuery function (the alias)

- selectors
  - The selector selects one or more elements on the web page.

- actions
  - The action that needs to be applied to the selected elements.

- parameters
  - How the actions are applied

jQuery Statements

Example:

```
jQuery('p').css('color','blue');
```

Selects all the paragraph tags in the HTML.

Applies a CSS **action** to it.

Changes the **colour to blue**.

The example required 2 arguments, but can be more or less.

They can be functions (event handler requires a pointer to a function).

CSS selectors and The DOM

The DOM – bits of rendered HTML.

```
<div id="heading">The heading</div>
```

div*#heading*

refers to a **div** element with an **id** of **heading**

- The **id** attribute must be unique.
- The **class** attribute allows assigning a name to multiple page elements.

Any element using the same class attribute will share the same CSS settings.

```
#footer { border: 2px solid black }
.warning { color: red }
```

is for the id attribute and . is for the class attribute

Making sure the page is ready

We can only interact with elements in the HTML if they have been loaded.

jQuery implements a built in event that executes the code as soon as possible.

```
$(document).ready(() => {
    alert('The document is ready!');
});
```

This code is inserted into the script.js file.
The first line is called the document-ready event.
Most jQuery is done after this event.
Doing this is a good idea to check if the jQuery library has been loaded correctly.
You only need to do the check once per page.

Making sure the page is ready

The code has the right anatomy:

- Selector: document
- Action: ready
- Parameter: a function that runs an alert

```
$(document).ready(() => {
    alert('The document is ready!');
});
```

There is a shortcut version of it:

```
$(() => {
    alert('The document is ready!');
});
```

Simple selecting

Selecting alternating entries in a table.

These alternate entries need to be gray.

First look at the markup of the table.

```
$('tr')
```

$('tr') will select all the rows.

Same for the other elements:

```
$('p')
$('div')
$('h1')
$('input')
```

Simple selecting

We need to select only the table that is relevant:

```
$('#users') // select all elements with an id of users
```

```
$('.data')  // select all elements with a class of data
```

Using the class select will select all elements on the page with the same class

- You can be more specific by selecting all the **tables** on the page with the same **class**

```
$('table.data')
```

Narrowing down the selection

We selected the table with the “users” id.

We want to select the table rows in the table.

In the code, we place a space between ancestor and descendant:

```
$('#users tr')
```

Narrowing down the selection

```
<div>
        <p>
                <span></span>
        </p>
</div>
<div class="fancy">
        <p>
                <span>this</span>
        </p>
</div>
```

We want to select the span inside the paragraph tag.

Only the paragraph tag inside the div tag with a class called “fancy”.

```
$('div.fancy p span')
```

Testing our selection

Check if we selected the right elements.

Use the length function which returns the number of elements selected.

```
$(() => {
        alert($('#users tr').length + ' elements!');
});
```

Testing our selection

Narrow the search for only elements inside the tbody element.

```
$(() => {
        alert($('#users tbody tr').length + ' elements!');
});
```

The jQuery object now contains the 6 elements we wanted selected.

Filters

Narrowing the results down to every other row is simple.

Make use of a **filter** in jQuery.

Filters remove certain items, keeping only what you want.

```
$(() => {
        alert($('#users tbody tr:even').length + ' elements!');
});
```

The example above will return 3 elements (the elements with even indexes).

Filters

Filters are attached to the item you want it applied to.

It is defined by a colon (:).

There are other filters as well but will be discussed later on (find them in the online documentation).

Selecting multiple elements

If you want to apply the same action to different elements, you can use a single select statement.

```
$('p, div, h1, input')
```

To apply the same action to several elements in unrelated parts of the page.

Separating the selector strings with commas allows you to do this.

Reading CSS properties

```
$(() => {
        let fontSize = $('#users tbody tr:first').css('font-size');
        alert(fontSize);
});
```

This will alert the font size of the first element it finds in the selector.

You can ask for the property of multiple selected elements but it will only return the first one.

Reading CSS properties

You receive the element's calculated style:

The style the browser rendered.

Not the style from the CSS definition.

This is handy because for example:

You define a div tag to be 200px high.

The content of the div tag pushes it over 200px.

The height displayed in the browser will be returned, not 200.

Setting CSS properties

Now we want to change the font colour as well to make it stand out more.

Easiest way is to repeat the code:

```
$('#users tbody tr:even').css('background-color','#dddddd');
$('#users tbody tr:even').css('color', '#666666');
```

This may get messy especially when changing a lot of different properties.

Streamline by using an object literal:

```
$('#users tbody tr:even').css(
        {
                'background-color': '#dddddd',
                'color': '#F00'
        }
);
```

Setting CSS properties

What’s wrong with doing things this way?

Setting CSS properties

By inspecting the code you will find that the style changes have been inserted inline.

This is not good practice.

All your styles need to be in one place, in your CSS file.

Separation of concerns.

What would be a better way of dynamically managing styling?

Adding and Removing Classes

We can then use jQuery to manage the classes that correspond to different styling rules

```
$('div').addClass('class_name');
$('div').addClass('class_name1 class_name2 class_name3');
```

(The second line will add more than one class.)

This way we can write CSS in a separate stylesheet and manage how it gets dynamically applied using jQuery

Adding and Removing Classes

For our example we first need to add the new class to a CSS file:

```
.zebra {
        background-color: #dddddd;
        color: #666666;
}
```

Then the new jQuery code will be:

```
$('#users tr:even').addClass('zebra');
```

Removing classes are exactly the same but we use removeClass instead.

Modifying content

You can also change the content of the element.

```
$('p').html('good bye, cruel paragraphs!');
$('h2').text('All your titles are belong to us');
```

The above examples will both add text to the inside of the elements.

The difference between the two (html and text)?

Modifying content

.html() vs .text()

```
$('p').html('<b>Warning!</b> Text has been replaced... ');
$('h2').text('<b>Warning!</b> Title elements can be...');
```

The paragraph tags will now contain the above text but with a bold “Warning!” in front of the rest of the text.

The headers will contain the full string with tags and all.

Modifying content

You can also fetch the content of the elements:

```
alert($('h2:first').text());
```

Event Handlers

jQuery has a number of built-in event handlers, for example

```
$('button').click(function(){
    // Do something when button is clicked
});
```

You can find a list of event handlers here:
https://api.jquery.com/category/events/

.click() vs. .on('click')

```
$("button.alert").click(function(){
    alert(1);
});
```

is a "shortcut method" for:

```
$("div#container").on('click', 'button.alert', function(){
    alert(1);
});
```

Reasons to use .on

- It uses less memory
- It works for dynamically added elements.

.on is the recommended way.

http://stackoverflow.com/questions/9122078/difference-between-onclick-vs-click

Hiding and revealing elements

The client wants to hide the disclaimer on the site after the client has read it.

Add a button in the HTML to do it:

```
<input type="button" id="hideButton" value="hide" />
```

We want the disclaimer element with a ID of “disclaimer” to hide:

```
$('#hideButton').on('click', function(){
      $('#disclaimer').hide();
});
```

Hiding and revealing elements

If we want to refer to the element that fired the event we use the this keyword.

We also need to convert the JS object to a jQuery object (we just wrap it in a jQuery selector).

```
$('#hideButton').on('click', function(){
        $(this).hide(); // a curious disappearing button.
});
```

Above code hides the button that triggered the event.

$(this) is a cleaner way to do than having to re-select it.

Showing a hidden object is exactly the same but with the show() action.

Hiding and revealing elements

Note that since arrow functions don't create their own this, you can't refer to the element the same way using arrow functions

```
$('#hideButton').on('click', () => {
        $(this).hide(); // this won't work
});
```

You *could* save the element as a variable and then refer to it when adding the event handler and inside the event handler function…

…or you could just not use arrow functions for this purpose

Toggling Elements

```
<input type="button" id="toggleButton" value="toggle" />
```

```
$('#toggleButton').on('click', function(){
        if($('#disclaimer').is(':visible')) {
                $('#disclaimer').hide();
        } else {
                $('#disclaimer').show();
        }
});
```

In the above code we used the is action:

- The action takes the same selectors we normally pass to the jQuery function.
- It checks to see if it matches the element it was called on.
- In the above code it checks if the selected #disclaimer is also selected by the pseudo selector :visible.
- It returns true or false depending if the elements matches the selector.

Toggling Elements

Toggling: going from one state to the other and back.

jQuery has built in toggle functionality

For hide/show, use toggle:

```
$('#toggleButton').on('click', function(){
        $('#disclaimer').toggle();
});
```

This will toggle the element to show or hide.

Toggling Elements

Now we want to change the value of the button based on the state of the element (hide or show).

```
$('#toggleButton').on('click', function(){
        $('#disclaimer').toggle();

        if($('#disclaimer').is(':visible')) {
                $(this).val('Hide');
        } else {
                $(this).val('Show');
        }
});
```

Remember that the selector $(this) refers to the element that caused the event to fire—in this case, the button.

Default Event Actions

Common method for controlling event flow in jQuery: preventDefault

Stop the browser from doing the default action an event would normally perform.

Example: Use it to stop a link from loading its target when clicked:

```
$("a").on("click", function(event) {
      event.preventDefault();
});
```

When you use return false, it will be as if you used both preventDefault and stopPropagation.

(You can use isDefaultPrevented and isPropagationStopped to check if the event flow has been modified.)

Default Event Actions

You can use event.currentTarget to get the element that is calling the event, for example

```
$("a").on("click", function(event) {
    console.log(event.currentTarget.innerHTML);
});
```

Or, using an arrow function

```
$("a").on("click", event => {
    console.log(event.currentTarget.innerHTML);
});
```

References

jQuery: Novice to Ninja by Earle Castledine and Craig Sharkie

http://api.jquery.com/