Lecture 11: jQuery, Browser Based APIs, and Fundamental Web Accessibility

CS-546 – WEB PROGRAMMING

jQuery

What is jQuery?

"jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript."

- via https://jquery.com/

Simply put, jQuery is an amazing DOM manipulation library that also has handy tools for making easier AJAX calls (which we will see later on in the course).

While the DOM API is not hard to use, it is extremely large; DOM traversal is also non-trivial and has many cross-browser compatibility issues. jQuery handles these issues for you.

What does this do for us?

jQuery has been around for quite awhile

- Very feature-rich API full of things to make development easy.
- Relatively bug free
- Very performant

For the sake of our course, we will mainly use jQuery for:

- Easier DOM traversal in order to target elements easier, and target / operate on entire sets of elements
- Easier event capturing; jQuery has much more DAMP syntax than vanilla JavaScript
- Easier creation of elements; jQuery can take an HTML string with attributes, classes, etc and create the node tree for you rather than having to setup through properties manually.

Using jQuery

The jQuery library is, by default, exported in the global variable *$*.

When using jQuery, it is important to remember that *functions are objects*; that means that they have properties of their own.

The *$* variable is first and foremost a function that will take a CSS selector (or DOM node) as an argument, and match all elements with that selector. The *$* function will return an *array-like* object.

- Like an array, you will be able to iterate through those results. (ie: can use .length, indexes, etc)
- Unlike an array, there are hundreds of jQuery specific methods you can call on the set of matches.

Recap on basic CSS Selectors

Type	Selector	Matches
Id	#my_bio	<h2>My Name Is <span class="my-name">Phil</span></h2> **<main id=“my_bio”>This is a test <h2>I love blogging</h2></main>**
Class	.my-name	<h2>My Name Is **<span class="my-name">Phil</span>**</h2> <main id=“my_bio”>This is a test <h2>I love blogging</h2></main>
Elements	h2	**<h2>My Name Is <span class="my-name">Phil</span></h2>** <main id=“my_bio”>This is a test **<h2>I love blogging</h2>**</main>
Combinations	#my_bio h2	<h2>My Name Is <span class="my-name">Phil</span></h2> <main id=“my_bio”>This is a test **<h2>I love blogging</h2>**</main>

Using jQuery

The jQuery library is, by default, exported in the global variable *$*.

We will be using the most modern version of jQuery.

- http://api.jquery.com/

Take a look at http://localhost:3000/examples/jquery-dom after running this week's repository to witness jQuery targeting many elements, in many ways.

Several common jQuery functions are demonstrated on that page, however there are many, **many easy to use functions**.

The code is well-documented to demonstrate how to use and understand each of these functions; it is very worth looking up jQuery on your own to see how many methods there are at your disposal.

Events with jQuery

Syntactically, jQuery does away with the concept of “onEventListener” and instead gives as many methods for event manipulation as possible using actual methods to represent the event.

- ie: instead of `document.getElementById(myForm).addEventListener(‘submit’, callback)` it’s just `$(selector).submit(callback)`

See code for examples

Console API

The Console Object

As you may have noticed while developing in Node.js, there is a handy object called the *console* object that allows us to log output.

The console object exists in our browser, and we can see it from developer consoles

- Most browsers have built in consoles
- Some browsers have extensions such as Firebug with more robust developer tools, including consoles

The console object exposes many ways to log different types of information and provides many other useful tools, such as profiling methods.

More about console

- https://developer.mozilla.org/en-US/docs/Web/API/Console

Debugging with the Console

The console allows you to debug your application easily, since it can print out objects and allow you to explore them. It also allows you to group series of messages together in order to more easily read through related messages.

See http://localhost:3000/examples/jquery-dom and http://localhost:3000/examples/manual-dom and open a developer console (then maybe refresh the page!) to watch for console actions as you run through the page.

Window API

What is the window?

In browser based JavaScript, the window object is your access point to anything browser related. The window contains all the global variables, APIs, and methods that exist. It also has many ways to get or change things that are related to the state of your current window (ie: opening a new tab, or getting the current scroll location).

See http://localhost:3000/examples/window and its related files to see what you can do with the window

https://developer.mozilla.org/en-US/docs/Web/API/Window

Timeouts and Intervals

In web development, you very often want something to happen some amount of time later.

In JavaScript you can run functions on a delay (setting a timeout of n-milliseconds before a function runs) or to execute on an interval (running and re-running a function every n-milliseconds).

Other APIs

The Location API

The Location API (not the Geolocation API!) allows you to get information about the current location of the page.

You can access many attributes about the current location of the browser such as query parameters, protocol, ports, hashes, etc.

You can see information and methods in the Location API in our examples this week:

- http://localhost:3000/examples/location

https://developer.mozilla.org/en-US/docs/Web/API/Location

Why would I use the location API?

Getting/setting the URL Hash

- URL Hashes are everything after the # in a URL; this is not sent to the server!
- You can use this information to include the concept of ‘state’ to your page.

Getting/setting querystring values

- Every if you do not use QS values on your server, you may want to use them in your JavaScript code; you can do that by accessing the *location.search* property.

Refreshing page and changing location

- Sometimes, you want to change where the user is based on interaction with the website. The Location API allows you to do that just by setting a string value of the new location
- You can also refresh a page

The LocalStorage API

LocalStorage allows you to store information across page views. This allows you to set values and retrieve them at a later time. LocalStorage stores all data as strings!

You can see uses of the LocalStorage API in the examples:

- http://localhost:3000/examples/localstorage

https://developer.mozilla.org/en-US/docs/Web/API/Storage/LocalStorage

Getting and setting localstorage values

Everything in localStorage is stored in the format of string-key -> string-value (even booleans, numbers, etc).

You will have to use the JSON.stringify() and JSON.parse() methods to encode your complex data types into a string and decode them, respectively.

The SessionStorage API

SessionStorage also allows you to store information across page views. This allows you to set values and retrieve them at a later time. SessionStorage stores all data as strings!

The major difference between SessionStorage and LocalStorage is that SessionStorage is specific to the page session. Opening a new tab or window initiates a new session.

https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage

Misc APIs

Audio / Video API

Modern browsers allow you to use JavaScript to manipulate interaction with videos, allowing you to create custom video players and audio players with great ease!

- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML5_audio_and_video

Canvas API

We can make a great degree of art in JavaScript by using a combination of the canvas HTML tag and using the Canvas API!

- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

Many modern charting tools use the Canvas API extensively.

Geolocation

We can track status about the user's location, as well. This is useful for searching for results in a user's general area, or showing things on maps, collecting analytics, and so on.

- https://developer.mozilla.org/en-US/docs/Web/API/Geolocation

Basic Web Accessibility

What is accessibility?

As web developers, we have the opportunity to make sure that our websites and web applications are consumable by people with a number of disabilities.

Even simple pages can have a number of issues that cause a person with some form of disability to be unable to fully use it; a form without labels, for example, is much harder for a screen reader to parse. A visually impaired user would struggle.

Even your design affects web accessibility: a lack of color contrast can make text nearly invisible to some of your users!

Testing accessibility

There are many ways we can test for accessibility, but to start, we will be using the tota11y tool

- http://khan.github.io/tota11y/

The tota11y tool is an accessibility visualizer that can be installed via a bookmarklet.

This tool will allow you to identify how assistive technologies would interpret your website.

Lack of color contrast

When text is overlaid on top of a background color that is not contrasting enough, people with visual impairments can sometimes not see the text.

Fixing a lack of color contrast is relatively easy to do. All you need to do is update the background and/or text color to have a greater degree of contrast.

When designing a website, it is useful to choose a limited number of colors to use. It strengthens your branding, and makes it trivially easy to fix this issue across an entire website when only a few colors are used. You would simply have to tweak the branding colors in order to make it consistently accessible across your entire website.

Improperly ordered/layered headings

Some people set headings based on how big the text should be, rather than how important the content is semantically; an out of order heading can confuse assistive technologies.

- The technology starts jumping around to what it assumes the most important information is based on the heading

Fixing improperly ordered headings is simple: consider your content before you write your code, and make it follow a normal hierarchy throughout your entire website.

Unclear link text

Sometimes, people use icons or images for their links instead of descriptive text; this is perfectly okay! But very often, they do not provide screen-reader visible text.

- Text can also be useless; ie: “click here”

Fixing this is easy:

- Make sure your links have a proper *title* attribute
- Make sure any images you wrap anchors around have *title* and *alt* attributes
- Using CSS tricks to make text that is *only* visible on a screen-reader

Unlabeled inputs

Assistive technologies rely on labels in order to properly describe forms.

Every input you write from now on should have an *id*, and a label that references that *id* with the *for* attribute.

Unlabeled alt text on images

Images without alternative text are utterly useless in regards to visual impairments. Assistive technologies will read off URLs, or skip the images entirely.

You should therefore *never* use images to represent text, and all images you use should have *alt* text.

How can we address these issues?

By tweaking our document and adding proper attributes/labels and updating our designs to factor in visual issues, we can fix a majority of accessibility issues.

Some issues are more complex; sometimes, users can see the document just fine but have limited mobility and can only use a keyboard. As you make more complex web applications, you have to make more and more considerations to make your website both functional and accessible.

Going forward

Going forward, **all HTML submitted must pass tota11y tests.**

Points will be deducted for labs and final project components that fail accessibility checks.