Website Creation (INFX1606)
Covers Stylesheets to JavaScript

Lecture Dates: October 3rd, 5th, 7th, 12th, 14th, 21st, 26th, 28th, 2011
http://moodle.cs.dal.ca | *Midterm*: October 19th, in class. | *Final*: December 12th 3:30 PM (2hr long)

Javascript Introduction

- In most webpages, HTML to develop, CSS to design, and Javascript to make the barebones work.
- References: CS Online Library, W3 Schools, Tutorials Online.
- Should not use any libraries not discussed in class (and not until they're covered).
- Javascript is:
  - Not related to Java.
  - Standardized version is ECMAscript (not by WC3).
  - Libraries used to solve complications, do complicated things in a broken down manner.
- An interpreted language (does not have to be compiled). Written in plain text.
  - Executed client side.
  - Browser support varies. Implemented differently in a lot of browsers.
  - Object oriented.
  - Test between both browsers.
- All of the code does not run immediately when a page loads unlike XHTML/CSS.
  - Proper language.
  - Support for control logic, variables, data types, functions, objects, maintaining state (between pages/visits – uses cookies).
- Can allow you to:
  - Run calculations based on changing information.
  - Manipulate the browser.
  - Build interactivity into a webpage.
  - Change the layout of the page after initial page load.
  - Maintain state, manipulate cookies.
- Can manipulate CSS, alter existing markup.
- It does not:
  - Directly write to databases.
  - Directly change information on the server (can only initiate page requests).
- The user is in control; can be easily turned off or not supported so be aware of this when building a webpage.
- **MIME Types: Multipurpose Internet Mail Extension Types. Standard way of identifying content being included. For example, when downloading files. Category/specifically.**
  - Will be on exam for sure.
  - Know what they are!
- Like CSS: can include in external file (`<head>` tag), include directly in page (`<script>` tags – in head if want immediately or if called later; if in the body, put it after, say, a `<p>` code so it executes that first since reading tag at a time – may miss the reading of a tag you want to reference).

- Event handler attributes. Executes a given set of JS code when event occurs.
  - Simple: `onclick`, `onload`.
  - `onblur` – when lose focus.
- Commenting is important!
- Good practice also dictates use of `<noscript>` tags. Will only display if JS not enabled. Useful when information must be communicated regardless of browser support.
- Assignment #4 Out Now! Little more structure. Build a site across 3 HTML files. Single external CSS file + a second for printing. JavaScript.
- DOM: **Document Object Model**
  - Has gone through many chances over the years – now standardized by W3C.
  - Provides a way of interpreting markup so that it can be affected by another language.
  - Must be supported by a browser for JS to work.
  - Built-in functions used to manipulate the browser, DOM.
  - Looks at WINDOW, History, Location, DOCUMENT, LINK, ANCHOR, FORM Elements.
- For form elements, have a `<label>` to identify them physically.
- Password-type text boxes *are not secure* – merely a physical mask.
- NOTE: Check out Firesheep.
- Check boxes, radio buttons, buttons that are of the same group should have the same `name`.

October 12th – JavaScript	*October 14th – Review*
October 17th – Class Cancelled (Extra Help!)	October 19th – Midterm
October 21st – Work Day (No new material).	

- Objects
  - Take their name from the same concept as real, tangible objects.
  - Can have things describe them (make each individual instance unique).
  - Actions an object can perform are just like functions, but when a function is part of an object we call it a *method*.
  - Properties an object has are called variables. In the context of an object, they are called instance variables.
  - Take the document object.
  - Can reference current date, Math functions, etc. (See W3Schools).
  - It is possible to create user-defined objects; we will deal with them as necessary.
- Manipulating Style
  - `document.getElementById('mydiv').style.border = "1px solid pink";`
  - Allowing multiple methods/values to be strung along one line.
  - Adding periods to keep referencing *backwards*.
  - In this case, setting the border of that element to be a 1pixel pink solid border.
  - CSS/HTML is how things are set at page load, usually, and JavaScript can manipulate it.
- To tweak little things and test it out: http://www.jsfiddle.net
- Camcelcase any CSS properties that are dashed (`border-top`).
- Getting values from forms.
  - `document.getElementById('number_one').value;`
  - From Radio Buttons: Use his code.

- Note: Prevent trying to show the user that programming is going on.
- Inspector can do some error/warning-catching for JavaScript.
- Half-Semester Review – if in the slides or said in-class, fair game.
  - One major question: interpreting a large amount of HTML along with some CSS and draw how a page may look.
  - *Basics*: Web Browsers, Rendering Engines (what it means to render), URLs, Client/Servers, Hosts, DNS, HTTP, FTP, web languages written in plain text and executed as a browser.
  - *XHTML*: What is it, involvement of W3C (non-profit, help standardize the language – not any particular company), terminology, writing properly formatted, major divisions (<head>,<body), tags for marking up text, hyper links and syntax, images and syntax, comments, recognize entities (like `&copy;`), doctypes, semantic markup, deprecated markup, *Remember! Never use XHTML to affect the presentation of a page*.
    - Recall – Tag: `<p>` and Element: `<p> Content </p>`
    - Proper nesting is important. Most recent closed first.
    - XHTML vs. HTML. ID/Classes.
    - Know things like `<p><h1> </h1></p>` not valid because headers are not paragraphs.
    - Know hyperlinks need `http://` otherwise, will go to a file on your server.
    - Biggest thing about images is that they are not complete image tags without a source and an `alt` text. Needs these two attributes.
    - Table: know that you have header, then footer, then body.
    - Recognize that when you have a doctype, forces browser to run in standards mode. If not validated or no doctype, triggers quirks mode. Tries to compensate for any errors you may have made. Generally undesirable.
  - *CSS*: Cascading Style Sheets, benefits/drawbacks (like not working correctly across all browsers), including style rules in your page, syntax, selectors, IDs/Classes, W3C Box Model, style sheets just for a particular medium like print and mobile, backgrounds/colors/ fonts/borders/images in CSS/floats/margins/alignments/positions, psedo classes (Lord Vader Handle Formerly Anakin).
    - Should know how to break down selectors and make very specific ones.
      - `a` vs. `div a`
      - `#magic a`
      - `div, a` (both independently)
      - Know about `clear` property.
      - Automatic margins: effectively CSS solution to centering (do not work top and bottom).
      - Alignment effectively applied using floats.
      - Position: if have a webpage, and a series of `div`s, and write another div into one of them, position attribute set to `static` (default value).
        - If change the value to be absolute, and then add `top`/`left` (only ever have to use two of the tools; do not need left and right). Looks for first parent that does not have static for its position value – therefore would go to webpage if no other.
        - However, if we were to give its parent box a relative value for its position, get best of both worlds – still in document flow but is now non-static. The smaller child box would go to the upper edge of the parent with top: 0, left 0.

        - Should be used ideally sparingly. Fragile, resizing can break.
        - Fixed is similar. Fixed to screen as well – stays still in viewport.
    - *Know box model*!
  - *JavaScript* (not a real focus): Client side, scripting language, not sanctioned by W3C, useful for interactivity, can manipulate elements of page after it is loaded, not exceedingly supported across all browsers, can interact with XHTML/CSS, can be enabled/disabled, `<noscript>` tags, commenting, including, order of execution, event handlers, conditions (if/else), recognizing functions, storing values, objects (document object), reading values from web forms.
    - Know form elements. Purposes of them.
  - Types of questions – will be based around general knowledge and specific recall of precise code.
    - Interpreting code to lay out a webpage.
    - True/false.
    - Fill in the blank.
    - MCs
    - Short answer.
- October 21st – Work Day (Move JavaScript!)
  - Assignment 5 Posted! **Due**: Next Thursday 27 October, 2011
    - Create a layout.
    - Have a number of form inputs. See table in PDF.
    - Does not have to have a *submit* button. Merely take note of the *triggers* for each form input.
  - JavaScript – Make the code yours.
    - Using the utility function is fine (for radio buttons).
    - Will eventually learn jQuery, exposure to PHP (but not expected to write it).
- October 26th – Even More!
  - Text Inputs: `document.getElementById().value`

jQuery

- First things first – Assignment 6 posted.
  - Very functional – not creating a page full of content, etc.
  - Take your page and create a field set tag (simple box you put a lot of form elements in), center it, add a legend (gives it a little label), and add either radios or dropdowns that whenever you make a selection, the box moves.
  - Moves to corners of viewport.
  - Can use either plain JavaScript code or jQuery to accomplish this assignment.
  - Due: November 4th, 2011 (Next *Friday*).
  - Merely an hour's work if familiar with CSS, event handlers from last assignment.
- Before we get into jQuery, let's talk about **arrays** a little bit more.
  - See Lecture 10 sample (shopping list).
  - Essence of an array: fitting multiple bits of information in one variable name.
- Difference references can be seen at W3Schools (JavaScript and HTML DOM).
- *jQuery* – much faster, more convenient. Quarter of a code to accomplish same job; but under

hood, uses the same JavaScript we have been using.
    - Cross-browser JS library.
    - Facilitates faster JS programming.
    - Increases the separation of structure from form and function.
    - Is *not* a separate language.
    - Allows you to associate event handlers to your HTML without having it actually inside your HTML.
- Used on *many* large websites.
- Created by John Resig in 2006; currently working for Mozilla. Not standardized. Documentation at website.
- Installation
    - Included like any other file (see assignments).
    - Important is included first before any other JS files.
    - Avoid including other libraries if possible.
- Practically no scripting actually has to be inside your HTML anymore. Back to merely being about structure.
- See sample from today.
    - $ - object. Same as saying doMath( );
    - Allows us to start using jQuery code.
    - Parameter you pass in is a *selector* – something like you write in CSS (a string).
    - Following the selector, can start calling functions – come from jQuery library.
- Full list of functions from jQuery library is on the website (API).
    - `attr` – gets value of attribute for first element in set of matched elements, sets it, etc.
    - Allows you to dynamically write in a new part of the HTML.
    - `click` – binds an event handler *onClick* dynamically to an element.
- Also provides events that are not provided in base JavaScript.
    - Example: `toggle`
    - Usually have to do onMouseClick, etc.
    - Can break down work to less and less.
- Chains
    - Do multiple things to an element.
    - Keep doing things by adding .
    - In regular JavaScript, cannot do multiple things in one line to the document call.
- Code that runs immediately should be wrapped in a .ready() function.
    - Waits until document is ready to execute that code.
    - If HTML has not actually been processed by computer, if you try to run any function, will try adding it to HTML that does not exist yet (JavaScript execution can be quicker than HTML loading).
    - See sample.