Website Creation (INFX1606)
Covers jQuery to the End

Lecture Dates: October 31st, 2011 / November 2nd, 4th, 7th, 9th, 14th, 16th – 28th, 30th, 2011 / December 2nd, 5th, 2011
http://moodle.cs.dal.ca | *Final*: December 12th 3:30 PM (2hr long)

---

jQuery

- Recall: `$` is short for writing out jQuery.
  - Input is the string representing a selector.
  - Looking for an id, class, tag you want to do something to.
- Chunk of code you want to run all at once (never have to reuse it).
  - **Anonymous Function**
  - Pass it in as a parameter.
- *Note*: JavaScript runs as fast as possible, not one line at a time. Example: the alert function as a callback rather than a new function after.
- Can call on what object in a selector is selected by using pseudoselector.
  - Example: `$('select option:selected').text();`
  - Because the jQuery library is downloaded, not relying on selector engine in browser like CSS does. More options than in CSS; more reliable. Huge advantage over browser.
  - Another layer of abstraction. Encapsulates a great deal of the difficult work.
- Can call on elements based on their content.
  - Example: `$("h3:contains('Click')").css('background','blue');`
  - Much quicker than adding a specific class. Can also allow for trickier targeting.
  - Example: Function `parent()`
    - Works off selector you initially write in.
    - Moves to parent tag.
    - h3 inside a div, for instance, that might not even possess an id or class.
    - Do not have to edit the HTML at all.
    - This kind of work is called *traversing* (traversing a tree). *Traversal calls* are the only function calls that will change the target you're working with. Other than that, can continue to add properties (chain).
    - Many *traversal functions* – see jQuery API.
- Similar to arrays, but a different structure: maps.
  - A chunk of settings, attributes you want to set.
  - Example, see `img_attribs` in sample.
  - A lot of different functions in jQuery, jQuery addons allow for these maps to passed in.
  - `$(".adding_css img").attr(mapping);`
- Cookies
  - Small text files passed back/forth between your computer and the website you are visiting every time you a visit a page on that website.
  - Can store information in this text file; pass information from page to page.
  - Allows you to maintain ***state*** in an otherwise stateless interaction.
  - The jQuery plugin for cookies makes setting simple cookies very straightforward – default

settings are sufficient for most applications. Expiry dates, secure connections, other options can be set.
    - Perfect place for user settings.
    - Login – not very secure for JavaScript.
- Can have cookies last longer than a session.
    - A session is as long as your browser is open.
    - When you quit browser, those are wiped.
    - All you have to do is pass a parameter; uses syntax of curly bracers – `{ expires: 7 }`
- Also flags to set it only when over an HTTPS connection.
- Cookie plugin not built into the library itself.
    - Remember order of inclusion.
    - Have jQuery library first, then plugins, then scripts.js (or any of your own scripts).
- jQuery UI series of plugins.
    - Another useful set of plugins. Series of controls and simple effects that you may use when building a typical webpage.
    - Great thing about JavaScript – pick through all source code to see what is under the hood.
    - If not, can simply use it through the use of documentation.
    - Again, *does not have to be written directly into your XHTML*.

- Midterm Review

Thinking Beyond "The Code"

- Building Websites
    - Simply building *a* website is straightforward. Very easy; just learn the languages and start coding.
    - XHTML: Structure, CSS: Style and Presentation, JavaScript: Interactivity
- Consideration: An *effective* website requires more than just writing code as we see fit.
    - Many rules.
    - Many guidelines that are open to interpretation.
    - An ever evolving and moving target – in terms of what we want to actually build to be successful; attention spans are small – trends change.
- Topics
    - Analytics – Who is visiting the site?
    - Search Engine Optimization – Getting found online.
        - Indexing of search engines depends on the parsing of your HTML.
        - You can structure your HTML in such a manner that it will show up higher depending on entered keywords.
        - Can have different versions of your site for search engine bots.
    - Accessibility – Maximizing your user base.
        - Not everyone browses web the same way.
        - If you have a phone, how do you break your site down, etc.
        - No plugins, deaf, etc.
    - Usability – Removing complexity from actions.
        - Build your page so that it is more appealing to *use*.

        - Very simple actions can remove complexity.
    - Design Considerations – Fashion trends happen online, too. (e.g. the Web 2.0 thing – lots of white space and fancy characters).
    - Security Consciousness – Be aware of what's happening on the web.
        - Comes from things that go beyond this course.
        - An awareness of it *is* inside the scope.
    - Content Management Systems – Building 'real-world' websites.
        - Most websites are built using software that generates the markup on the fly.
        - What we're building are more like *templates* – what we would replicate across many different pages.
        - High level discussion to this end.
    - Social Media APIs, RSS Feeds, Web Business Model Basics.
- Assignment #7 Posted – Due next Monday.
    - Potentially last assignment (last assignment only need to be done if marks have been perfect so far).
    - Creating a simple photo gallery – using jQuery.
    - First major requirement: collect **12 to 16 photos**.
    - Arrange them in a grid – use HTML and CSS to pull that together.
    - Use photos of a reasonable size. Resize 10MP photos down, etc.
        - Need 2 sizes – thumbnail, full-size.
        - Once images are all set up, implement a pop-up effect for these all with colorbox, some other image pop-up plugin.
    - Use a combination of cookie plugin and a form element to allow the user to set a "preference" of some CSS setting; could be a way to remember setting that rearranges grid of photos or as simple as affecting background.

Intro to Accessibility

- Until now, we've been simply seeing if code validated, etc.
- Accessibility relates to both…
    - The practice of making websites easier to operate by those who have a disability.
    - The practice of building sites accessible to users with technically limited devices, software (viewport). Example: Hover does not translate (well) to iPad.
- It leads to…
    - Usability, SEO (Search Engine Optimization), etc.
    - Much overlap between making pages technically and physically accessible; making sites perform better overall. Example: Validating a code – good structure. Easy to read.
- Keep mind open that the web can be accessed many different ways.
- Other consequences.
    - More accessible = more users = more popularity (= more profit).
    - Socially responsible.
    - The Law. Example: Target sued over inaccessible website.
- Article by Joe Clark
    - 56.7% of nondisabled people have Internet access.
    - 21.1% of people with vision problems have Internet access.

    - See more stats. http://joeclark.org/book/sashay/serialization/Chapter02.html
- Every user that reaches web-page should be able to take full advantage of the information available *wherever* possible.
    - Example: Having captions for images; subtitles/dubs for audio.
- Does not necessarily mean that accessing information is *easy*.
- Many guides available for this.
    - Industry body backed: W3C Web Accessibility Initiative
    - Government sponsored: Canadian 'Look and Feel' Policy
    - Swedish Public Sector Sites
- No online, overall enforcer of these guidelines. Left up to institutions to govern themselves.
- W3C is, really, once again the major player. WAI is on the second iteration (2.0).
    - 1.0 received criticism for focusing on just being technically accurate, not being user centric.
    - 2.0 incorporates a lot of this feedback; both in way written and goals advocates. A lot more qualitative.
- For example: `<img src="vacation.jpg" alt="picture" />`
    - Technically correct.
    - Rather useless.
- *Similar in spirit to* ***semantic markup*** – details are in how you write the code, not just that you write it.
- Another example: `<strong>` vs `<b>` and `<em>` vs `<i>`
    - Visually, look the same way.
    - However, semantic meaning is different; is interpreted differently by various software.
- Not always a simple tag switch.
- Making pages accessible *impact* many factors centring around the cost/time to produce site.
- Many questions have to be asked…
    - If this impact is worth it? (Socially/Financially)
    - What is the true audience? Potential audience? (Measured with Analytics)
- Typically involves *more verbose markup*.
    - Multiple versions of markup to accommodate different users.
    - More information to manage.
- Leverages a concept called ***graceful degradation***.
    - Markup not understood by a browser is simply ignored.
    - Gracefully degrades to a lesser experience. Nothing happens; rest of website renders.
    - Example: HTML5 Video `<video>` tag.
        - YouTube usually offers Flash video. With many Android phones, iPad/iPhone do not run Flash.
        - The video tag bundles the software required to play the video in the site. Bundles it right in. A great deal of different markup.
            - If device supports webm, will play it.
            - If not, according to graceful degradation, goes to next (after ignoring it).
            - This involves at least three copies of a video.
        - This accessibility is what has made YouTube get ahead.
- Could put a lot of effort into nothing.
    - Certain accessibility tags not even recognized by browsers.
    - Others arguably work against it. Example: *Accesskey*. Could be overwriting default

behavior.

- Read over the actionable parts of the WAI (Both Versions).
  - 1.0/2.0.
  - Mostly focus on 2.0.
- *Note*: Cannot perfectly parse a webpage to say it is *accessible* (by a computer). The descriptions in WAI are meant to be human-readable. Also mash up against reality sometimes; not always perfect in given contexts.
- Web Content Accessibility Guidelines
  - http://www.w3.org/TR/WCAG20
  - Making it so no matter how you originally created media, can have it accessed by physical, technical disabilities.
  - Also helpful for SEOs. Cannot read audio, video, etc.
  - *Perceivable*, *Operable*, *Understandable*, *Robust* – Main guidelines.
  - (1) **Perceivable**
    - Text Alternatives – for non-text content. Example: Audio.
    - Time-Based Media – Alternatives for time-based media (example: time-based scrolling images). Like, a way to manipulate it (pause, skip).
    - Adaptable – content can be presented in different ways without losing information, structure.
    - Distinguishable – Easier to users to see/hear content including separating foreground from background. Idea of less is more. Contrast.
  - (2) **Operable**
    - Keyboard Accessible – all functionality available from a keyboard. This is really hard. A way this is built into browsers is with tab functionality. Can control where tab goes with HTML.
    - Enough Time – provide users enough time to read and use content.
    - Seizures – Caused by blinking lights.
    - Navigable – Provide ways to help users navigate, find content, determine where they are.
      - Bypass Blocks: Ways to bypass blocks of content repeated on multiple web pages.
      - Page Titled: Semantic.
      - *Bad*: No current page in sidebar nav bars. Idea is to create a mental model – easy way to see how the webpage works immediately.
      - Etc.
  - (3) **Understandable**
    - Readable – make text content readable, understandable.
      - Have to realize not everyone is a power user.
      - Language of Page: Default human language of each Web page can be programmatically determined.
      - Language of Parts: Of parts.
      - Red text on purple background, for instance. Color scheme.
      - Nice to note the `<abbr>` tag. Hovered over acronym and outputs the full word(s).
      - Usually has a dotted underline, or have a question mark show if you hover cursor over.
    - Predictable – Operate, appear in predictable ways.

            - Don't typically see only copy of the menu all the way on the bottom.
            - Consistent Navigation – occur in same relative order on every page unless change initiated by user.
            - Change on Request
            - Interface typically designed in such a way that you read from left-to-right. Navigation to main content to side content.
            - Rollovers tend to make people choose before they have to. Nice to just have sub-categories in category pages.
        - Input Assistance – Help users avoid, correct mistakes.
            - Labels, Instructions
            - Error identification.
            - Reversible submissions.
    - (4) **Robust**
        - Compatible – maximize compatibility with current and future user agents, including assistive technologies.
            - Parsing – elements have complete start, end tags. Nested accordingly.
            - Name, Role, Value – Semantic markup.
            - Orientation of page – i.e. on iPhone.

- **ASSIGNMENT #8** *Posted on Accessibility*.

Usability Starter

- Accessibility: Making information accessible.
- Usability: Enhancing that one step further; land on the page – reducing the barrier. Making it a quicker process.
- No all-encompassing guide like Accessibility. A lot of opinions.
- Refers to ease with which user can achieve their goal. Specially websites, software.
- Big business.
- Aim to:
    - Increase retention of information.
    - Increase *useful* time spent on site.
    - Increase user satisfaction; reduce frustration.
    - Decrease time to understand.
    - Remove roadblocks to reaching user goals – gather information, register for event, etc.
- Significant impact on website usage.
    - How quickly a user can login to your website, purchase a product, etc. plays a role in whether or not they will make a purchase.
    - Speed and efficiency – big bucks.
    - Good example: Amazon. 1-click technology. Patented.
- Misconceptions
    - Lead to unusable sites.
    - [1] "People like to work for it."
    - [2] "Gain a sense of accomplishment."

    - [3] “Too much expense / Nobody cares.”
    - Misconstrue valid notions such as:
        - Users like ability to show off accomplishments.
        - Scarcity of alternatives motivates users.
- If people are happy *AND* they have a sense of accomplishment, they’ll come back.
- Example: www.codeacademy.com
- Exceptions
    - Web pages do not always have to be usable.
    - Sometimes goal is to showcase your content in an awkward way. Example: *photographer websites*.
- A wide variety of factors including technical, cultural, learned, etc. factors.
    - *Technical*: Across devices.
    - *Cultural*: Language, etc.
    - Usability is both qualitative and quantitative.
    - UseIt.com is a treasure trove of tangible directions on usability.
    - High based on **human behaviour**.
    - http://www.useit.com/alertbox/
- **Key Usability Elements**
    - [1] Consistency
    - [2] Predictability
    - [3] Discoverability
    - [4] Complexity
    - Allows a user to build a ‘mental model’ of your page. Eventually, users make a mental model of how websites should work in general. Leveraging this is key.
    - Based on beliefs, not facts; model of what users know.
    - Certain level of inertia – one instance of being wrong will not entirely change the model but eventually flat out change the model.
        - Example: Links. Evolution from underlining to different colors.
        - Usually see underlines for hovers to assure you are correct.
    - [1] **Consistency**
        - Within your own site; across multiple sites.
            - Where are the navigation links?
            - Are all downloads of same file format? If link to files, should indicate that delivering something other than an HTML page.
            - Are the URLs consistent or do they change over time? Reliable URLs.
                - Think of spaces in addresses.
                - Try to always use lowercase for everything.
        - Maintaining this increases usage speed by allowing user to avoid re-factoring their ‘mental model’ of your site.
    - [2] **Predictability**
        - Closely tied with [1].
        - Comes about from behaviours specific to websites, from elements in daily life.
            - Underlined text is (rightfully) assumed to be a link.
            - Users usually scan from left → right, top → bottom; expect information to be along this scan line. Search usually at the top right.

        - Expectation to see an about us, contact us page.
    - User Interfaces – very important.
    - Making your site predictable allows user to rely on mental model they have created for websites in general.
- [3] **Discoverability**
    - Will not always have situations familiar to users.
    - In this case, goal to make website as easy to learn as possible.
    - Not entirely done by search.
    - Done by:
        - Using [1], [2] effectively.
        - Clear, concise documentation.
        - Clear, logical organization structure.
        - Not penalizing mistakes; be able to go back and resubmit.
    - Example: * = *Required* over large paragraphs.
    - *People Do Not Want To Think*. Maximize efficiency.
- [4] **Complexity**
    - Covers how well your website hides complicated situation from the user.
    - Want to hide complexity.
    - Example: When a paper form is turned into an electronic process.
        - Fields should automatically fill themselves in: shipping = billing address. Do not want to fill in multiple times.
        - Clearly indicate what fields = necessary.
        - Re-populate information if an error occurs.
        - Only show optional fields if they are required.
    - Doing less = more. Removing complexity of spending money, for instance.
- *Common Sense*
    - Most of this is common sense.
    - Cater to low-reading level (7 to 9).
    - Why do so many pages fail then?
        - Temptation to force user behaviour. *Needing to sign in, for instance, to view content.*
        - It's easier to write a simple instruction to the user than actually solve the problem.
        - Desire to use latest technology ahead of general acceptance. *If people do not upgrade their browser*, *do not get latest information. Agencies typically look for IE7 – 9, Firefox 3.5 – 7, Chrome, Safari (but still uses same engine as Chrome), lots of hand-waving at mobile (an extremely large number of them).*
        - Time constraints, *"that will do for now."*

Film: TED Talk – Rory Sutherland: Life lessons from an ad man.

- Impulse Saving
- Intrinsic Value
- Trains – went to engineers/programmers: how do we make it more attractive to people?
    - E: Spend 6 billion dollars to make a straighter route. Get models to serve food.
    - Pr: HTML5 – new specs, etc.
- All comes down to what *makes people happier*. Assumption: you not going to a website to

watch a video vs. just for information.
    - Putting a video, or a link to a video.
    - Not using technology but human instincts, behavior.
- Speeding Signs – not post speed just a "frowny face".
    - Much more effective than speeding tickets.
    - Less cost than cameras.
    - Emotional trigger: all it took to make people slow down.
    - Web Analogy: Give users what they want; not flashiness.
- Netflix
    - Interface is not very good; bad response.
    - Scrolling movies – no indication can scroll.
    - *Remember*: Try to hide the complexity. Multiple accounts, etc.

Usability Testing

- Testing your site to see how usable it is.
- Include a variety of monitoring.
    - Record user sessions. *Have them describe what they are doing.*
    - Monitor eye movements. *Heat maps, for instance. Lots of heat may sometimes be a bad thing (e.g. lots of staring, searching).*
    - Record mouse movements. *Can do split testing – have half readers receive one version and another half another version, for instance.*
    - Record clicks/key presses (both successful, unsuccessful).
- A lot of website design can occur long before you code. Build as pictures, on paper, etc. Let them navigate those, for instance, instead of a usable page.
- Can hire firms to do this sort of thing. General strategy: reduce clicks, reduce error rates, get information out there.
- ***CRAP*** Design Principles; watch for against different resolutions.
- *College Students on the Web*.
    - Don't like learning new interfaces.
    - Do not want to be blasted with motion and audio at all times.
    - Prefer sites that look *clean and simple*. Search dominant.
    - When it takes off, social networking marketing works, but not always the case. Do not want all kinds of companies in their Facebook feed, etc.
- What Not To Do
    - Jakon Nielsen article about this. From 1996 but still applicable for the most part.
    - Frames
    - Gratuitous Use of Bleeding-Edge Technology
    - Scrolling Text, Marquees, Constantly Running Animations
    - Complex URLs
    - Outdated Information
    - Download Times
    - Legibility Problems – low contrast, bad fonts.
    - Non-Standard Links
    - Flash – on its way out, non-standard, usually completely custom and flashy. Like an ad.

- **Content Not Written for the Web**
  - Short, scannable, to the point (for short attention spans).
  - Speed is of the essence.
- **Bad Search** – talk about this later.
  - Most navigating around now done by search.
  - May not be just on your page; also through things like Google.
- **Opening New Browser Windows**
- PDF Files for Online Reading
  - Breaks Flow, usually.
  - Do not follow usual control.
- Browser Incompatibility – i.e. JavaScript
  - Use modern code, DOCTYPE.
  - Build up that way. Use JavaScript libraries.
- Cumbersome Forms
  - Used too often.
  - Often too big.
  - *Make shorter*! Support autofill. Flexible input. No Reset button.
- No Contact Information, Other Company Info
- Not necessarily bad, actually helpful: Fixed Page Widths – otherwise, putting too much onus on the user.
- Inadequate Photo Enlargement

Search Engine Optimization

- High volume of traffic a website receives is through search engines; even ones that are highly popular.
- Search Engine Optimization
  - A way to make your page as findable as possible.
  - Collection of techniques that web developers use to maximize a website's ranking in search engines.
- Why is it important?
  - High ranking: more traffic.
  - More traffic: more potential for income, popularity.
- SEO is more important on pages that are informational: advertising, places, people, topic reference, etc.
- All revolve around gaining a high ranking through means that are *free*. Technical rather than financial methods.
- Financially based means are search engine marketing.
- Early search engines relied heavily on developer-provided keywords. *Honesty*. Meta tags a prime example here. Naturally, was abused. 20-30% of search results were pornography.
- Evolved to use factors more complex; entire page content, formatting. Google was a real game changer in this field.
- Techniques guarded closely.
- Nowadays, do not know exactly how they rank is determined; all about testing what results get more traffic.

- Getting Started
    - Manual submission to index.
    - Wait to be indexed automatically.
- Automated indexing is done through software; crawler. Move through websites by following links embedded on page.
- If do not want site indexed, can write a textfile (robots.txt) to put in your directory. Standard has no official oversight.
- Two main techniques now:
    - White Hat – Playing by the rules.
        - Encouraged.
        - Organization of content on page – put content first, *as high as you can*.
        - Good keyword choice.
        - Submitting sites to web directories.
        - Being linked to by authoritative sites.
        - Good domain name: harder than you think. Most taken.
        - Clean, validating code.
    - Black Hat – Anything goes. Get as high as you can before you get banned, put extremely low.
        - Can be very effective. Right up until you are banned (permanently, temporarily).
        - Ultimately detrimental to creating a useful internet.
        - Results achieved tend to benefit only the website owner.
        - Temporary gain for long-term pain.
        - Some examples:
            - Cloaking: Relies on serving a different webpage from the same domain, dependent on the user agent or IP.
                - User agent: Web browser, etc. Long string of text. See what OS they're using, what version.
                - This *could* be used to good benefit, using jQuery, PHP.
                - Cloaking uses code (server-side, usually).
                - Start looking for the web crawlers (Google Bot, etc.).
                - Cloaked version would be overpopulated with keywords, link exchanges, etc. Clearly *not* a usable page.
                - Extremely effective, but #1 reason to get banned (not creating anything to serve someone).
            - Link Farming
                - Search engines give higher credence to pages with a multitude of incoming links.
                - Can ask other people to exchange links with you.
                - And when everyone is being honest, generally a solid assumption that if everyone linking to a site, must be important.
            - Keyword Stuffing
                - Stuff into markup, meta tags, content area.
                - To their credit, search engines do not place nearly as much value on meta data as they once did.
- *Search Engines*: do not neglect region-specific search engines.

    - Google, etc.
    - WoframAlpha
    - Yandex (Asia)
- Effectiveness
    - Continually adding, tweaking content.
    - Keeps search engines interested; crawling more often.
    - Can be measured using analytics software.
    - Important to notice “organic traffic”. Came naturally; did not pay for it.
    - Direct traffic relevant too.
    - Converting organic traffic into direct is often a top priority.
- Split Testing
    - Involves running 2 designs simultaneously.
    - Different designs, different users.
    - Quick way to generate statistics, can be analyzed.
    - Helps show what should be changed, should not.

Analytics

- ***Example***: Google Analytics; Third-Party. Increasingly more popular option.
    - Makes it easier to share the stats.
    - Full pages, but can get it down between elements on the page.
    - Depends on data points being sent back.
    - Key to them is to notice a little bit of JavaScript added to the page. Has a unique identifier, usually.
    - Approximately half of the websites online.
    - Bounce Rate: Page was loaded, then left. Bounced off site.
        - May have to do with the fact all computers in building load it at default.
        - May go to web cameras, then leave.
- Can do log files. If you have control on your webserver and know the computer running it.
    - Get logs.
    - When start parsing files, bunch of plain text. Copy of user agent.
- ***Note***: Large bodied organizations have a limited number of IPs they can distribute. Limited area of influence, so can map to certain areas.
    - IP maps can be expensive. Google offering mapping for free.
    - Knowing where visits are from: influence what languages you serve up.
    - Can decide how to redesign to make it more appealing. Some languages go in other directions, so may have to redesign site a bit.
- *Languages*: Not necessarily location; just what computer is running.
    - Not talking about HTML.
    - Talking about design.
    - Using language as a tool.
- *Traffic Sources*: Direct Traffic vs. Referring Sites vs. Search Engines.
- *Search Engines*: Keywords, Engines, etc.

Leveraging Website Management Software

- There are many more languages used to build things online.
- Many categories of software.
- Biggest thing we haven't talking about, but very much a reality, are CMSs. Content Management Software.
    - XHTML, CSS, JavaScript are all languages key to developing website, but suffer from a major deficiency. (Client-Side Technology).
        - For example, very tedious to work with over large sites.
        - With small sites, not a major task to copy, paste code in order to maintain a consistent layout.
        - Very few websites are only constructed out of 4, 5 pages.
        - Also cannot do large-scale saving of customization, information. E.g. Person staying logged in.
    - In practice, very few sites are made entirely with client-side software.
    - Instead, server-side software used to repeatedly generate client-side code that is sent to users.
- Server Side
    - Code written in a server-side language; this is *never* exposed to the user (unless an error has occurred).
    - All of the code is run on the web server.
    - Only client side content is sent back to the user.
    - Languages include: PHP Hypertext Processor (PHP), Active Server Pages (ASP), Perl, Python, Java.
    - Can use these to dynamically build much larger-scale websites.
- CMS Templates
    - CMS generates client-side code from a small collection of "templates".
    - Process has many advantages:
        - Changes propagated immediately to all pages.
        - Code only needs to be written once.
        - Pages generated dynamically, modified depending on specific information they are displaying.
        - Information can be customized to visitor.
        - Enables writers that do not know web languages to author content.
    - For example: `<div><?php echo $content; ?></div>`
    - More code comes in before this, but you have one variable here that can change.
- Other advantages of CMS Software
    - Separates webpage structure from content.
    - Automated menu creation & updating.
    - Automated handling of URLs and filenames.
    - Handles uploading of media files (images, video, audio).
    - Empower multiple authors to maintain a single site.
    - Format content into multiple display streams.
- Requirements for setting up a CMS:
    - Hosting Space, Database Access
    - Knowledge of client-side code in order to customize templates.

    - Working understanding of server-side languages in order to install. Very straight-forward, usually.
    - Functional knowledge of server-side languages in order to customize operation.
- Examples
    - Wordpress
    - Drupal
    - Joomla
    - MediaWiki
- All of these packages are *free to download*, written by communities.
- Some may be available for both download for you to host it, or have it hosted by someone else. Others may be hosted only.
- Demo – installing Drupal.
    - Cuts down 100 to 150 hours of work into about 25 hours when we talk about the kind of scale we're talking about.
    - Make a lot of things very convenient to set up.
- Collectively referred to as "***Web 2.0 Apps***"
    - Idea that web is in its second evolution.
    - Before, web was mainly informational – get information you want it and maybe share it manually.
    - Now, just with Google Apps alone, can chat, do homework, set up advertisements, etc.
- Generally has two sides:
    - Public/registered user side – this half is viewable to all visitors to the side, either with/ without an account.
    - Admin side – half is viewable only to site operator(s).
- Both sides are rendered using same language, technology.
    - Just control interfaces.
    - Change what user sees.
- Risks
    - Start doing dynamic work now. Changes; delivers or behaves in many ways.
    - Execution of codes on server exposes it to attack.
    - Form elements to upload content can also be used to upload more code.
        - Requires safeguards to ensure that only safe markup is processed.
        - For example, `<strong>` tags are allowed, however `<script>` tags would not be.
        - XSS – Cross Site Scripting
    - Misconfiguration giving access to secure files.
    - Session Stealing: Do not use a password that you use anywhere else. Do not usually use https://
    - Using out-of-date software.
- *Risks do not outweigh the benefits*.

Being Security Conscious

- *Final assignment due next week,* still.
- *Exam*: About 2 hours; will have 3 hours allotted.

- When building, using websites.
- Will not necessarily have been issues up until now. Not dealing with particularly complicated languages, at the moment (client-side). Generally benign at what they can accomplish.
    - Static (Unchanging).
    - Visitor-independent.
    - Simply delivered to the user, no server-side execution.
- With the introduction of CMS technology, host of new situations are brought into fold.
    - Dynamic pages.
    - Visitors must be able to "log in". Remembered; server-side + cookies.
    - Data retrieved from a database before being manipulated into HTML data.
        - Not often clean. Fed in by other users, typically.
        - May be feeding information completely benign (ASCII characters).
        - May be feeding slightly dangerous characters (apostophes, $, %, etc.). Mean something when in code. Could break output, or trying to put it in on purpose – execute entirely new code.
- Good list is with the OWASP people.
    - OWASP Top 10 – 2010.
    - These issues to do with lots of data being moved around.
    - ***Example***: Injection Flaws – tricking into executing code (example SQL).
        - Myriad of ways to sanitize, filter. Many people do not do this, though.
        - Maybe forgot, not aware, etc.
    - ***Example***: Cross-Site Scripting (XSS) Flaws
        - Similar in concept. Still have boxes feeding code in.
        - But feed JavaScript – send to web browser.
        - For example, put `<script></script>` tags.
        - Can hijack user session (session identifier in cookie), deface web sites, redirect user to malicious sites.
        - Application: *Firesheep*. How easy it is to take hold of someone else's information.
    - ***Example***: Insecure Direct Object References – access files, directories, database keys – access unauthorized data.
    - ***Example***: Cross-Site Request Forgery
        - When you set up a form, you can set where that form submits to.
        - Usually have it send to same website.
        - Can do it to another website. If you look at any form in web, can look at HTML.
        - Can have forms working in opposite way (not really the same thing): really good spam.
        - If have a URL in a <img src=""/ >, will visit that page. Browser loads images immediately.
    - ***Example***: Security Misconfiguration
        - Default passwords, etc.
        - Facebook themselves have 100s of webservers. They had a single one go wonky one day – started showing code. Very quickly scooped up.
    - ***Example***: Insecure Cryptographic Storage
        - Not properly protecting sensitive data (credit cards, SSNs, authentication

credentials, etc.).
        - Not actually encrypted all data.
        - Proof-positive that whenever you get an e-mail that asks for your password to check your account, should not send password.
        - Have a master account.
        - Many of them have MD5 encryption – can sometimes decrypt easily.
    - ***Example***: Failure to Restrict URL Access
        - More specific of Direct Object References.
        - Not actually being checked, verified.
    - ***Example***: Not using HTTPS. (May be computationally more expensive, but something to consider).
    - ***Example***: Invalidated Redirects, Forwards
        - Attackers can redirect victims to phishing, malware sites, or use forwards to access unauthorized pages.
        - Should take means to prevent this.
- Be aware that you're giving away information whether you like to or not.
- Specific Examples
    - *Login Status*: Seeing if logged into websites. Social Engineering Attacks – say you're hacked, etc.
    - *Users/Admins*: Gmail Engineer who used his access to start reading peoples' e-mails.
- Precautions
    - Keep up-to-date.
    - Avoid sites that *look* suspicious.
    - Encrypt files you store on remote servers.
    - Do not store information online you would not be comfortable having exposed.
    - Never give passwords to anyone. Use different passwords on different systems.

Exam Review

- jQuery Library
    - Creator
    - Not a different language.
    - Simplifies development – shorter, easier to understand. Cross-browser development really well.
    - More advanced selector engine than one in browser. Very different. Reason why it can be more advanced because you have to download library when you visit webpage.
    - Cannot force anyone to get a new browser.
    - Anonymous Functions
- *(...)*
- Web Analytics
    - Useful for evaluating what work to perform.
    - E.g. Google Analytics – piece of JavaScript.
- Content Management Systems (CMSs)
    - Software needed for larger systems.
    - Used in real-world settings.

    - Example: *Wordpress*
- Web Security