Website Creation (INFX1606)
Covers Introductory Concepts to Stylesheets

Lecture Dates: September 9th, 12th, 14th, 16th, 19th, 21st, 23rd, 26th, 28th, 30th, 2011
http://moodle.cs.dal.ca | Lab: Attendance is optional. | No formal textbooks required.

Introductory Web Concepts

- This course is about *development* and NOT design. Learning the foundations of developing websites.
- Creating the art for a website is best learned in a more design oriented course.
- Weekly assignments – due the next week.
- What is a **web site**? Quote from Wikipedia.
  - Just a ***collection of files***. Comprised of files such as .html, .css, and .js
  - Doesn't matter where they are located – just changes how you access.
  - Local vs. Remote – whether or not they are on *your* computer or on someone else's.
  - Usually, websites are "on the web". Web is analogous to many interconnections between computers and other networks of computers.
  - Internet: massive amount of networks connected to each other. For instance, home network connects to ISP (i.e. Eastlink) which connect to each other.
  - Every **computer connected to a network = host. Host you fetch the website from is called a *server*, your host is called the *client*.**
  - **Server runs web server software** (Apache, Windows). The web browser on the client connects and retrieves the webpage.
  - **That means a host could be serving both purposes depending on software it is running. Location or components of computer are irrelevant.**
- The parts we "see" of how a computer connects to the internet include:
  - URLs, Domain Names, Protocol Names, etc.
  - **URL**: Uniform (Unique) Resource Locator, defines location of information and method by which it can be retrieved. URI = UR Identifier.
    - *Form*: `protocol://subdomain.domainname:port/path?querystring#fragment`
    - Certain parts can be omitted, default values used in stead.
    - *Protocol*: Network protocol defines a standard method of formatting data for the purpose of communication between computer systems. (Language) Examples: HTTP (HyperText Transfer Protocol) – foundation of the web, HTTPS, FTP (password is in plain text, insecure – old technology), SFTP (newer, more secure), file (local files).
      - Raw Protocol, Sample HTTP: GET /index.php HTTP/1.1 Host: www.cs.dal.ca
      - This is what browser sends, interprets response.
      - All server and browsers communicate with the same base terminology.
    - *Domain Name*: Critical, starting point, always required. Most basic form is an arbitrary name and a TLD (i.e. ca). *TLD* = Top Level Domain (standardized, limited number, exist for organization types, countries, etc.).
      - A lot of things were based on honesty.
      - A lot of things don't work that way anymore because of that (example: .com only made for corporations).

            - Domains can have any number of sub-domains, separated by periods.
            - The “www” used so frequently is *not* required. Only a convention.
            - Domains themselves handled by ICANN (Internet Corporation for Assigned Names and Numbers). DNS = Domain Name System; humans good with names, computers good with numbers.
            - Every online device has a number to identify it (IP Address), 4 integers long each with values between 0 – 255. Can put it in your web browser.
            - Ultimately, whole system is contingent on just a handful of vendors operating “Root Servers” to keep the name translation working. If they went down, Internet would effectively get to a halt. *Just everyone agreeing to do the same thing*!
            - You can buy a Domain Name but still need a web server host to put your files on!
        - *Port (Number)*: Integer value used to identify how a computer is listening for a conversation. HTTP is 80 by default. HTTPS is 443. Ports are useful/required for data moving between computers.
        - *Path*: Arbitrary name of the files/folders that you create for a website (arguably second-most important). The initial “/” counts for the base. Remember: No spaces in file names (not wrong, but not technically right).
            - index.* is named by convention.
        - *Query Strings* and *Fragments*: Talk about later.
        - **Standards**: Technical specs like DNS, URLs, standardized through memorandums called RFCs (Request for Comments).
        - IETF (Internet Engineering Task Force): Responsible for managing RFCs.
            - Funding comes from industry partners and government.
            - NSA, Universities, etc.
- Web Hosting
    - A bunch of web site files sitting on a computer not of much use.
    - Aside from knowing URL, you'll need a space to hold your website for all to see. (Web Hosting)
    - Can be purchased from a number of companies online for a monthly fee.
    - Demo of uploading.
        - File Types: File extensions serve two functions.
            - Indicate what kind of data is contained.
            - Indicate to OS what program should open.
            - Suggested Text Editors: TextWrangler (Mac) and Notepad++ (Windows) – both are free. TextMate is used by teacher.
        - SFTP Client – Recommended Clients: Firezilla (Windows) and Cyberduck (Mac) – both free.
        - We use *bluenose* file server (bluenose.cs.dal.ca). Use CSID.
        - Website on: http://web.cs.dal.ca/~safatli/
- Assignment due next Wednesday

Client-Side Programming & X(HTML)

- Webpage

    - A webpage can be considered to be a myriad of specially formatted text & included forms of media.
    - Up to the browser to interpret this text (website source) and render it accordingly.
- Rendering Engines
    - Part of the web browser that determines how the webpage is arranged in the viewport (space where the page is brought up).
    - Can be separate from the browser; can be built directly into an operating system.
    - Ideally, all should operate the same.
        - Support is better in recent years, but previous versions of browsers have been inconsistent in their rendering.
        - Testing in multiple browsers in required to ensure a consistent user experience.
    - Safari and Chrome = WebKit (Safari/Chrome almost always render the same way). Built into Macs, iOS, etc. Shows how interface can influence a lot of people (Chrome picked up a large audience right after release).
    - Firefox = Gecko / IE = Trident
    - W3C creates rules, defines the languages and how should be rendered. Standards/ conventions cover just about everything to do with the basics of the web.
- Web Languages
    - XHTML, CSS, JavaScript
    - XHTML
        - Defines structures, gives context, includes media.
        - W3C standard.
    - CSS
        - Defines style, positioning, colour, media.
        - W3C standard.
    - JavaScript
        - Adds interactivity, data processing, page manip.
        - Various industry bodies (helped standardized by ECMA International).
    - These three operate client-side (everything having to execute them happens on the computer with the browser).
    - Web Languages (three compliment each other).
    - Server-side languages include PHP, Python, Ruby, SHTML, etc. Do not work if you try to just open the file on your computer. Have to be processed by web server software. (Decisions being made).
- Flash
    - A "defacto standard" – not a standard designed by committee.
    - Must be installed in order for content to work.
    - Very processor intensive, not open – will not be developing in Flash in this course.
- Before programming, upgrade to the latest version.
- HTML
    - Foundation of a webpage. Used to bring structure to content (not style).
    - Is a markup language, involves annotating existing content – doesn't involve programming logic but can be structured to degrade gracefully.
        - For example, `<h1>John</h1>`
    - Syntax dictated by W3C – currently on its 4th iteration – the HTML5 standard is still being

    finalized.
  - XHTML defined after HTML4, now we're going back to just HTML (no X, in HTML5).
  - Example: remove styling from Dal.ca. Can *view source*. Plain HTML often written in just a slightly different way than often exposed.
  - Limited series of defined tags and accompanying attributes. Rules for how a tag displays by default.
  - Rules on how tags can be nested within each other.
    - Close things open most recently.
    - Failure to close tags confuses the browser and can cause unexpected output.
    - List of tags at: http://www.w3schools.com/tags/
  - Attributes determine how a tag behaves.
    - Comprised of a name and value pair, value surrounded by quotes.
  - Main difference between XHTML and HTML is how code is written (XHTML has a stricter set of rules).
  - XHTML: Whitespace is irrelevant, always use lowercase, always surround attribute values in quotes, fully structure, start and end with `<html> </html>`
- Developer Tools
  - Web Developer Toolbar (Firefox Plugin), Firebug (Firefox Plugin)
  - Developer Tools (Safari & Chrome)
- Document Structure
  - Some tags allowed to only be inside certain other tags.
- Head and Body
  - Beyond <html>, a XHTML document has 2 fundamental divisions: body and head.
  - Head Tag: Contains meta-information (information about other information). Describes the body.
    - Contains basic information like the title, keywords.
    - Contain technical information like CSS references, language.
    - We'll revisit the contents of it later and as necessary.
  - Body Tag: Contains actual content.
    - Only other second-tier tag (along with head tag, can only be one of each). Always follows head tag.
  - Basic text:
    - <p></p> paragraph tags (mainstay for text).
    - Line breaks done with <br />
    - <strong> and <em> add strength and emphasis (respectively) to your text. Change how void reader reads this text.
    - Can use <b></b> and <i></i> as well, but do not have the added accessibility the other two have.
  - Headers
    - <h1>, <h2>, …, <h6>
    - Arbitrary numbering an old standard (LaTEX), dictates the higher the number, more important the header is on the page.
  - Lists
    - Three types of lists: unordered, ordered, definition lists.
    - Unordered Lists

            - Bullet list.
            - <ul></ul>
            - Can place any number of list items inside denoted by <li></li> tags.
        - Ordered List
            - Produce numbering.
            - <ol></ol>
        - Definition Lists
            - Terms and corresponding definitions.
- See assignment.
- Links
    - Hypertext links denoted using <a> tags. (a = "anchor")
    - *Hypertext reference* ("href") as a property.
- Images
    - Alt tags needed for it to be correct (*alternative text*).
    - Information on the image.
- HTML Comments
    - Annotations using comments.
    - `<!-- This text is commented -->`
    - Don't put anything security-sensitive in comments (gets sent along to browsers).
- Semantic Markup
    - Markup content according to what it is.
    - Lists for lists of links, for instance.
- Things to avoid.
    - <font>, <center> tags are from pre-HTML4
    - Frames, etc.
    - See slide.

X(HTML), continued

- Linking
    - Having covered images and hypertext links.
    - Can link externally or internally (outside or inside side).
        - External: use the full URL.
        - Hot-linking is generally frowned upon as it uses someone else's resources.
            - "Rude" – uses someone else's bandwidth.
            - Could change what image is, as well.
    - Internal image src links comes in 2 varities.
        - Relative and Absolute.
        - Relative: image being linked relative to the location of the page linking it.
        - Absolute: linked relative to their position in the file system.
            - Based off everything after the domain name, typically, on a webserver.
            - Example: `/~safatli/media/image.jpg`
        - "./" syntax to go up a folder.
    - Absolute links such as file:/// … will only usually work on your own computer.
- Anchor Links

    - "#anchorlink" – jumps to different points on a page.
    - Jumps to a target ID.
    - See slide.
- Tables
- Other Tags
- Unknown Tags
    - "*Graceful degradation*": web constantly evolving – modern markup can be used without worry of breaking the rendering in older browsers.
    - Tags thrown out by browser – not rendered.

HTML Entities, Document Types, etc.

- When building *valid* HTML documents, important to have HTML Entities.
    - Method of displaying characters that are either reserved (e.g. < sign) or non-standard (not directly mapped on keyboard, different languages).
    - All start with a & sign, end with ;.
    - For example: &#174; ®
- Cheat sheets on *Moodle*, W3Schools.
- Valid XHTML requires the use of these. Ampersands (&) cannot be written *directly* into the document (must be written as &amp;). Otherwise, encounter errors in HTML Validator.
- Validating & Doctypes
    - Doctypes: Document Type Declaration.
    - Indicates which DTD (Docu. Type Definition) will be used. Breaks down rules for what makes a valid webpage.
    - "html" indicates we're in HTML class of Doctypes, PUBLIC (all make use of it), "-//W3C…" they standardied it, DTD definition (modern Doctype), website where guaranteed file of definitions is.
    - Always come first – single line. Different ones out there.
    - Valid Tag list – using only tags with "S" (strict) in the DTD column at http://w3schools.com/tags.
    - Test at http://validator.w3.org.
    - Errors: (1) Typos, (2) Nesting tags incorrectly.
    - Strict Cheat Sheet: www.w3.org/2010/04/xhtml10-strict.html
- Valid Nesting
    - Lists *DO NOT* go inside <p> paragraph tags.
    - Content should be contained within a tag, always.
    - Etc.
- Validate **Frequently**
- Display Modes
    - Render certain tags different ways.
    - Came about because nobody "gave a crap" for a lot of years (would only work in certain browsers) – making sense of mess.
    - Proper, "Standards Mode" – if not validated, page will break.
    - Missing, improper Doctype cause the page to render in "Quirks Mode" – attempts to fix

stuff for you. (If didn't close tags, does it for you, etc.).

- **Keep code TIDY**!
  - New context, more semantic markup – more hooks for styling, etc.
  - Easier to read, parse.

Cascading Style Sheets (CSS)

- Webpage Style
- Cascading Style Sheets
  - Client side web technology controlling the presentation of content.
  - Standard maintained by W3C.
  - Interpreted language – not markup *nor* functional – still just plain text (just a series of reserved words & syntax).
  - Several versions. Primarily working with version 2.
- Benefits
  - Written in a separate file.
  - Much more rapid than markup buried into code.
  - See http://www.csszengarden.com.
- Difficulties
  - More of a learning curve.
  - Sometimes interpreted differently on various browsers (IE is the main culprit).
- Cascading Style Sheets – breakdown of the name.
  - Cascading – styles "cascade" through nested HTML.
  - Style – controls the style/presentation of tags.
  - Sheet – CSS is essentially just a long sheet of rules.
- *Strict Rules* – validator, too.
- Some style included by default (basic CSS built into the browser).

Assignment 2 posted this evening.

- Three main ways to include your CSS/style.
  - Included from an external document.
  - Nested in the <head> tag.
  - Included in a style attribute on a specific tag.
- Almost always from an external document.
  - Value for media include "all", "screen", "print", "handheld"
  - Allows you to handle accessibility for different devices.
  - All: all browsers, Screen: desktops, some handhelds, Print: print-version, Handheld: mobile version.
- No syntax to properly start/end an external style sheet. Ideally named `.css`.
- Style tags nested in the <head> tag can only be used on the page they are defined.
- Reusability of CSS code.
- Arbitrary Divisions
  - `<div>` and `<span>` tags

    - Div: tags default to block display. (Used 99% of the time)
        - Cause line breaks.
        - Stretch across viewport, force elements down.
    - Span: tags default to inline display. (A few words, lines)
        - Do not push anything down.
        - Don't get in the way of each other.
    - Very useful for styling content.
- CSS Example
    - `/* This is a comment! */` comment syntax
    - `p` CSS affects all paragraphs on your page.
- CSS Breakdown
    - Selector (i.e. p from the earlier example)
    - Curly bracers contain Property:Value pairs (end with ;).
- Last and most specific selector always wins (when you have conflicting ones).
- Certain CSS rules "cascade" into the elements nested inside them automatically.
- ID vs. Class Selectors – only difference is that a specific ID can only be used once per page. Classes can be reused endlessly. For example: `#header` ID for a div element vs. paragraphs with certain classes.
    - Think semantic naming.
    - Only start names with a-z.
- Can combine ID/Class and tags (see slide) in CSS.
- IDs/Classes always considered more specific than general tag selectors. IDs trump Classes.
    - Example:

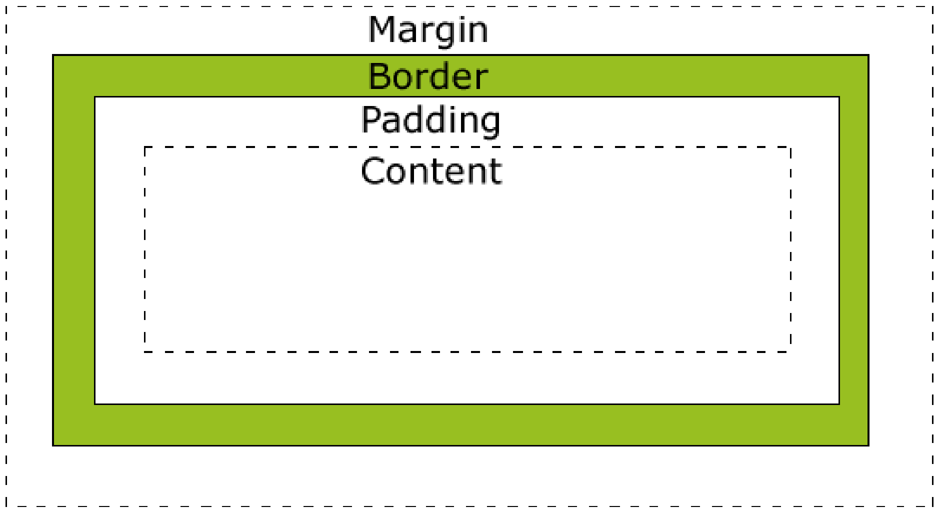
```
<p id="test" class="case">

#test {
    color: red;
}


p.case {
    color: blue;
}
```

    - The #test above trumps the p.case.
- See color example slide (* = wildcard).
- There are a lot of modern ways to style in advance manner – but not all browsers can use it. Should work in as many places as possible.
- Tags not explicitly covered in class include: `<address>`, `<quote>`, `<block>`.

CSS Properties – Units, Boxes, Colours

- Positioning, sizing, spacing, colours, fonts, list styles, borders, accessibilities – wide range of properties available.
- Developer Toolkits are really helpful here.

- Positioning of Elements require 3 compontents
    - CSS Rule
    - Value
    - Unit of measure.
        - Relative – based off parent, font size in use.
        - Absolute
- Any rule using this syntax requires every component.
- Relative Units
    - em – comes from world of print.
    - 1 em = font size of parent tag (16 pixels).
        - Tend to scale well with increasing “zoom” settings in browsers.
    - ex – stands for letter ‘x’
        - Not used that much.
    - % – percentage
        - Calculates sizing based on properties of the tags surrounding target tag.
- Absolute Units
    - px – pixels: main measure of a computer screen.
    - Pixels available are absolute. Calculations are pretty easy.
    - cm, mm, in, pt (1/72 in), pc (12 pt)
    - More useful when dealing with print rather than screen.
    - Varying screen sizes would make designing in these measurements very difficult.
- Older browsers don’t work with pixels well (cannot zoom in on pixel-based sites).
- First property set we’ll cover: *Boxes*
    - How things align against eachother, etc.
    - CSS Box Model (Probably most important part in CSS)
    - Can apply box styling effects to any tag; wraps around everything in that tag.
    - Around this “invisible” element is a margin (outside spacing, outside border), border (border around content), padding (spacing between border and content), width (actual width of boxes *content* – not anything else).



    - See later image showing more details.

        - More padding, margin, etc. take up more room across than just the width property.
        - Width only specifies width of the inside content.
        - Total Width = `LM+LB+LP+Width+RP+RB+RM`
        - Never specify `width` as 100%; box will use more.
        - Early versions of IE treated the `width` property as including everything except the margin (wrong!). To maintain compatibility, modern versions of IE continue to render boxes improperly when running in quirks mode – ultimately, boxes will come out correct because of Doctype.
    - Property keywords come in shorthand/verbose versions (see slides).
    - Shorthand: Clockwise.
    - Two borders with no margin between them will display overlapped. Otherwise, no interference between tags.
- Colours, cont'd.
    - Just writing the word works.
    - But there are 3 ways in general to specify a colour value: Raw RGB, RGB Hex, Colour Names.
    - **Raw RGB**: `rgb(255,255,255)` = white
    - **RGB Hex**: # followed by 6 values. Can be abbreviated as 3 characters if pattern repeats.
    - **Colour Names**: As before.
    - Once-upon-a-time, there used to be websafe colours (screens, computers did not present as many colors as they do today).
    - Think `<div>` tags.
- `Max-` and `min-` width/height is supported in modern browsers to ensure an object does not get smaller or larger even if the window is resized. Larger monitors.
- Width and height does not work on inline elements!
- Margins do not always require numeric values – use `auto` to achieve automatic spacing on either side (equal amount of space no both sides). Effectively centres tag within container tag.
    - Does not work vertically.
    - Can specify it for the whole margin, but pragmatically this works better: `margin: 20px auto;`
    - IE, again, gets this wrong by default (automatic margins). Quirks mode uses text-align (not margins) for centering everything.
    - Providing a doctype gives correct behaviour. Text-align, to let older IE browsers work, should be applied to main body, not div tags.
- Visibility
    - Visible vs. hidden.
    - Tag still rendered in document flow.
- Display
    - More useful than visibility.
    - Controls a target display mode: `block`, `inline`, `none`.
    - Everything will move if something is removed, contrary to visibility.
- Document Flow
- Float
    - Useful for lining block content up in a row, wrapping content, pushing content up the page, out of the normal flow.

  - For example, having an image squeeze into text.
  - Sends a tag to either side of the page (left or right) or `none`. Causes other content to wrap around (removed from document flow).
  - Behave somewhat like inline tags.
- Activity – Positioning
- `#wrap` – containing space for the entire webpage.
- `clear` – can have a value of left, right, or both – very nice property in CSS – e.g. footer underneath floated elements, clearfix. Corrects wrap. "No floating elements allowed on the left or the right side of a specified paragraph".
- `border-radius` – most elegant way to curve boxes. Will not work with any browsers older than a year.
- Precise Positioning: Sort of "anti-" web development.
  - Usually evident in programs built by programs like *Dreamweaver*.
  - Code is much longer.
- Reference Points
  - In CSS, called `position`. Default: `static`.
  - Can also have *fixed* (referenced off viewport and no other elements), *absolute*, *relative*.
  - Can be done from top, bottom, left, right.
  - Only works if something other than static position has been set.
  - Absolute: typically used for popups.
  - `z-index` can be applied to anything that is non-static (highest index placed on top).
- Setting a non-static position on, say, the wrap allows absolute positioning to work off of that containing element rather than the viewport.
- Backgrounds
  - `background-color`
  - `background-image`
  - `background-repeat`
  - `background-attachment` (fixed or moving)
  - `background-position`
- Background CSS is effective for:
  - Placing images across large portions of your page.
  - Placing single images part of the design and not the focus of the page.
- Image Files
  - Important for web.
  - JPEG, GIF, PNG – compressed formats. PNG = alpha layer.
- Fonts
  - Font families, styles sizes, etc.
  - Font properties cascade into children tags.
  - Font-weight: done in 100s, up to 700 (no 459, just 500). Can do in keywords: `bold`, `bolder`.
  - ***Will be tested on.***
- Serif vs. Sans-serif
  - Serif: Fancier, 'tails', better generally for paper, big font.
  - Sans-serif: No 'tails', all "business", better for reading on screens in smaller fonts (main content).

- Tables, cont’d.
    - Border-spacing
    - Border-collapse: uniformity amongst all table border styles.
- Pseudo Classes
    - Similar to normal classes.
    - ***Not explicitly stated in markup; only in CSS.***
    - ***Link, visited, hover, focus, active*** – in this order. If do one, have all 5 there even if they’re the same. Lord Vader Handle Formerly Anakin
    - Example: `a:visited { … }`
    - Focus: Tabbing between links. “Highlighted”.
- Your should be able to:
    - Create print-only style sheets. Can use Print Preview to test it.
    - `use:media="print"`

- Liquid vs. Fixed Layout
    - Fixed: Does not move when window is resized.
    - Liquid: Resizes completely.
        - Effective if mixed with min-, max-width.
    - A concern for huge resolutions. Fixed is not by accident – easier to take in information when you don’t have to keep reading left to right.
    - For photo galleries, purpose-built sites: liquid widths are nice!
    - Mobile resolutions: liquid is also nice.
    - Responsive design: different CSS rules when you start resizing (i.e. mobile).
        - `newstartns.ca`
        - Elaboration of this concept.
- Alignment: Think if in-lined or not (lines are “boxes”).
- Can horizontally align lists (CSS). `display: inline;`
    - Or! Use float and list-style-type (none)