Agenda

- **Charsets**
- **HTML**, **XHTML**, **HTML5**
- **Semantics**
- **Forms**, **SVG**, **Multimedia**
- **To Do**

Character Sets And Languages

The W3C recommends the **UTF-8 charset** for web development. It includes an impressive array of languages and special characters, and is well supported in common devices.

Assign Character Set

If it is not specified, a web browser will choose the character set it thinks is most appropriate, but it is a better practice for the developer to ensure the correct character set by applying the **meta** tag to the **head** section of every HTML page:

<meta charset="utf-8" />

- Write all your text-based code (**.html**, **.css**, **.js**, **.php**, etc) in UTF-8. At development time, ensure your text editor is saving files in this format. This setting is usually found in the application's Preferences menu option.
- Unless your audience includes very old browsers, you can **code special characters** directly into the HTML (you dont need to use HTML entities for most special characters).
- You cannot copy-paste rich text (such as Microsoft Word, or WordPad) into your HTML. The rich text formatting may result in an unreadable mess.
- An HTTP server can over-ride your charset preferences when it generates the HTTP header. Details on how to control this will be discussed later in this course.

HTML lang Attribute

Use the **lang=""** attribute to **declare the language in HTML**. This is usually applied to the **html** element. Assigning the appropriate language code will apply to all contained content, unless over-ridden with another **lang=""**.

Use the language abbreviations from the **standard ISO 639-1** code set (don't use ISO 639-2).

HTML

What started as a language defined by **SGML** *(Standard Generalized Markup Language)*, HTML is used to mark up billions of pages, making up the bulk of the web.

- Late 1991, Tim Berners-Lee releases the first iteration of what would become HTML 2.0 (there was no real 1.0+)
- HTML was a language created at the dawn of the Web, using SGML (Standard Generalized Markup Language) as a template
- Over the next decade, various additions, improvements and changes are introduced
- **HTML 4.01** released in December 1999

- Initially used for documents and a place for the technically inclined, the web did not have the international pervasiveness it does today
- Most developers were new to the language and there were no real classes in it
- People had to learn as they went along
- A lot of early sites are poorly designed

HTML 4.01 template

The problem with the state of HTML then was its reputation as a loose language. Poorly structured code would be rendered differently across user agents, leading to forked code and less predictable results.

XHTML

The **XHTML specification** was the result of rewriting **HTML 4.01** using the ruleset derived from **XML**

- HTML 4.01 + XML = XHTML 1.0
- X = eXtensible
- X = XML related
- Better, more predictable coding patterns.
- Verifiable or "well-formed" code can be produced.
- Forces a higher standard of HTML.
- Predictable behavior across user agents.
- Porting to HTML5 is easy (if you also observe **semantic** guidelines).

XHTML Rules:

- All elements must be properly nested
    - This is not valid: ** Bolded Text **
    - This can be a problem with things like Lists <ol> and <ul>.
    - Especially when closing the List Item <li> tag that contains the <ul> or <ol> tag.
- All tags must close
    - Example: ** Some text here **
    - Empty Elements are closed as well: **
** or ****
    - An extra space is required before the slash to work with all browsers.
- All tag names must be in lower case
    - **<Img Src="some_image.gif">** is not valid
- Attribute names are always in lower case
    - ** ** should be ** **
- All attributes must be quoted
    - **<body bgcolor=black>** will not work, you need: **<body bgcolor="black">**
- Attributes cannot be minimized
    - **<input type="checkbox" checked>** cannot be used. The valid code is: **<input type="checkbox" checked="checked" />**
- Name Attribute is no longer used (except with forms). Replace "name" with "id".

        - **** will not validate. Use **id="myimage"** instead.
    - Mandatory Elements: Every XHTML document **must have** these elements:
        - **<html>**
        - **<head>**
        - **<title>**
        - **<body>**
        - The **<!DOCTYPE>** declaration *must* be there, but it is *part* of the document itself rather than an element of the document.
    - Documents must be well-formed
        - The document must conform to all of the above rules
    - Optional XML declaration
        - Not *required*, but good practice:
        - **<?xml version="1.0" encoding="ISO-8859-1"?>**
    - Optional Element: An XHTML document should also declare the type and character encoding in the head section to ensure the server sends the correct content-type header:
      <meta http-equiv="Content-Type" content="text/html" charset="UTF-8">

XHTML 1.0 template

HTML5

As of October 28, 2014, **HTML5 is the official W3C recommendation**

HTML5 is more evolutionary then revolutionary. The bulk of previous HTML elements and attributes are maintained, with a few deprecations, and several additions. Significant features include:

- Ultra simple Doctype: **<!DOCTYPE html>**
- Specifies how browsers should behave with imperfect code
- Ability to embed **XML**, **MathML** or **SVG** markup.
- Standardized Javascript API increases client side script compatibility across browsers
- Backwards compatible - older versions of HTML can effectively be updated by simply changing the doctype to HTML5
- Guiding principles: Enhance semantic coding, Support existing content, Pave the cowpaths

HTML 5 template

Paving the Cowpaths

Common developer practices have been simplified

- Finally a DOCTYPE we can all remember:
  <!DOCTYPE html>
- Declaring the character encoding is easier:
  <meta charset="utf-8">

- When linking to external stylesheets the **type** attribute is not required:
  <link rel="stylesheet" href="style.css" />
- When embedding CSS the **type** attribute is not required:
  <style></style>
- Including javascripts is simplified as well (**type** attribute is not required)
  <script src="javascripts.js"></script>

HTML5: Enhanced Semantic Coding

It is crucial for all HTML content to be **semantically marked up**. This is how browsers, search engines, and screen readers make sense of online content.

HTML5 provides more than just adjustments to the markup, there are new, powerfully semantic elements:

- **<section>** is a tag representing a document or application section. in general, if you are planning to use a heading, start a new **section**. a **section** must contain a heading, and may contain **article**(s), possibly **aside**s or even sub-**section**s
- **<article>** is an independent piece of content in a document (it could stand alone if removed from the page). an **article** might be a blog post, forum thread, news story, collection of product information, etc
- **<aside>** is for content "only slightly" related to the main page content. **aside**s fill the role of a sidebar. if the content could be removed without reducing the meaning of the main content of the HTML document, then use an **aside**
- **<header>** and **<footer>** are for the header or footer of a page
- **<nav>** represents an area for navigation
- **<main>** used for containing content that is focused on the central topic of the page. this content is usually unique to the page, and not shared by other pages (main will usually NOT contain navigations, footers, sidebars, etc)
- **<figure>** will likely contain an **img** or graphic. self-contained content (could usually be removed from the page and stand on its own). allows for captioning of embedded content like an image graphic or video. if you want to associate a caption, add a **figcaption** child
- **<figcaption>** used as first or last child of **figure** to define the caption or legend for a figure

These represent some big changes in HTML, allowing for more flexibility in coding and specifying content. Many of these will replace and reduce the need for many **div** tags. eg: instead of the typical ** **, use **<header>**.

Having trouble deciding which tag to use for what content? Try this **HTML5 flowchart**.

HTML5 Semantic Alterations

HTML5 also brings a few notable semantic alterations to older tags:

- **<a>** though still an inline tag, it is now ok to nest multiple block level tags (headings, paragraphs etc) inside an anchor tag
- **<small>** no longer a 'physical' tag for smaller sized print, it now has semantic value: meaning 'small print', i.e. 'legalese'. the **big** element has been deprecated

- **<b>** no longer means 'render bold'. now it means the text is 'stylisticly offset from the normal text', without conveying any extra importance. to convey extra importance, use ****
- **<i>** now means the text is 'in an alternate voice or mood', without conveying any extra emphasis. to convey extra emphasis, use ****
- Deprecated tags: **<big>, <font>, <strike>**, and a few more. Developers should use CSS instead of these deprecated tags.
- Deprecated attributes: **align, bgcolor, border, height, size, type, width** and more. Developers should use CSS instead of these deprecated attributes.

HTML5 Content Models

Pre-HTML5, there were basically two categories of tags: inline and block. HTML5 intoduces a more nuanced set of categories that allow for greater semantic sectioning of content. The content model will help the browser to determine the semantics of your content.

Content Models

- text-level content: most inline tags
- grouping content: most block tags
- replaced content: all the form widgets and related tags
- embedded content: video, audio, canvas
- sectioning content: new structural, semantic options

Sectioning Content

The **<section>** tag can be used to semantically group content. It can remove ambiguity when a page is being processed by the browser, a screen reader, or search engine.
For example, imagine you have created the following code:
Since content that follows a heading is presumed to be associated with that heading, the code above carries plenty of semantic value already. But, if the **small** tag's content is intended to apply to all cities, a browser has no way of knowing that. It will instead assume the **small** tag is associated with the preceding heading (**<h3>Tokyo</h3>**)
Use the **section** tag to explicitly demarcate the start and end of the related content:
This new sectioning will inform the browser that the **small** is associated with the **h1**
In HTML5, each piece of sectioned content has its own self-contained outline. This means you wont need to worry as much about which level heading tag to use. You can use an **h1** inside a section and it will be treated as the heading of the section, and have lesser semantic impact than an **h1** that is at a higher level.

Forms

HTML 5 adds several new **form attributes** that prove quite useful, even if some are as yet poorly supported.

- **placeholder="value"**: prepopulates field with data
- **autofocus="autofocus"**: sets the input to have cursor focus
- **required="required"**: ensures field is filled in before submission
- There are also several new **<input>** type attribute variants:

- **type="email"**: checkes for the pattern of emails
- **type="url"**: web addesses
- **type="date"**: calendar popup
- **type="tel"**: telephone numbers
- **type="search"**: formats text input as search input
- **type="color"**: color picker popup
- **type="range"**: sliding scales
- **type="pattern"**: regular expression pattern matching

Scaleable Vector Graphics

Scaleable Vector Graphics (SVG) can be embedded into HTML5 documents. SVG are 100% scalable without the pixelated effect that scaled raster images can suffer from. SVG file data is stored as text, so they are much smaller in file size than a raster equivalent.

You can use SVG graphics with the **img** tag, just as with raster graphics. You can also use .svg as a CSS **background-image**.

****

The **png** on the left is 26KB. The **.svg** on the right is only 4KB.

Alternatively, you can use SVG code 'inline' with the **svg** tag. The advantage of doing this is it will result in one less request/response between the client and the server. The disadvantage to inline SVG is that it adds considerable clutter to your your **.html** code. Just copy the source code from your **.svg** file and paste it directly into your **.html**!

```
<svg viewBox="0 0 55 28">
    <!-- svg code goes here... sometimes there is a LOT of code here! -->
</svg>
```

SVG Tools

Use an SVG application to help you create and manipulate your SVG code.

- Adobe Illustrator
- Inkscape (Open source, Windows/Mac/Linux)

Multimedia

HTML5's more responsive multimedia features have made it easier to provide appropriately formatted video and audio to the client.

Audio

The audio element provides a way to offer many audio formats to the client, allowing the browser to choose the one most suitable.

The most common audio formats for the web
are: **audio/mp3**, **audio/mpeg** and **audio/ogg**.
Each audio tag may contain:

- **source** tags as children of the **audio** tag, one for each audio format available. Each **source** must have a **src** (path to audio file) and **type** (audio file MIME type)
- **controls="controls"** give the user control over the audio *important for usability*
- **loop="loop"** loop audio playback
- **preload="none"** dont load the audio until the user clicks 'play', **preload="metadata"** preloads the clip duration statistic only, **preload="auto"** loads the audio clip to the client before they click 'play'
- **autoplay="autoplay"** begin playing audio as soon as page is loaded *NOT very user friendly*
- you may also wish to include a direct link for your users to download the file

```
<audio controls="controls">
        <source src="media/house-a-square.mpg" type="audio/mpeg" />
        <source src="media/house-a-square.mp3" type="audio/mp3" />
        <source src="media/house-a-square.ogg" type="audio/ogg" />
</audio>
```

Download the audio

Video

The video element provides a way to offer many video formats to the client, allowing the browser to choose the one most suitable.
The most common video formats for the web
are: **video/mp4**, **video/ogg** and **video/webm**.
Each video tag may contain:

- **source** tags as children of the **video** tag, one for each video format available. Each **source** must have a **src** (path to video file) and **type** (video file MIME type)
- **controls="controls"** give the user control over the video playback *important for usability*
- **width="250" height="100"** set the size of the video (use CSS instead. absolute values only, no percentages)
- **poster="image.jpg"** display a static image when the video is not playing
- **loop="loop"** loop playback
- **muted="muted"** begin with audio silenced
- **preload="none"** dont load the video until the user clicks 'play', **preload="metadata"** preloads the clip duration statistic only, **preload="auto"** loads the clip to the client before they click 'play'
- **autoplay="autoplay"** begin playing video as soon as page is loaded *NOT very user friendly*
- you may also wish to include a direct link for your users to download the file

```
<video          preload="none"
        controls="controls"
        poster="media/poster-bear-in-water.jpg">

        <source src="media/bear-in-water.webm" type="video/webm" />
        <source src="media/bear-in-water.mp4" type="video/mp4" />
        <source src="media/bear-in-water.ogg" type="video/ogg" />
</video>
```

Download the video

HTML5 Today

User agent browsers will have varying support for the HTML5 specification.

- caniuse.com is an excellent source for compatibility testing
- Try the HTML5Test to test a specific browser.
- To ensure early versions of Internet Explorer will render HTML5 tags correctly, you can add the HTML5 shiv:
    - **<script>document.createElement("article");</script>**
    - This creates an element called **<article>** which the browser (IE, in this case) can understand and style appropriately
- Creating an entry for every element can get a bit heavy, so you can use the HTML5shiv script written by Remy Sharp. Insert a coded-comment that will apply the shiv ONLY with IE browsers earlier than version 9:
- **<!--[if lt IE 9]>**
- **<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js">**
  </script>

<![endif]-->

- Since HTML5 understands both HTML4 and XHTML tags, converting an older HTML file into HTML5 is simply a matter of updating the DOCTYPE and meta tag for charset.

To Do

- Download, review, and complete the homework assignment from D2L
- Review the HTML5 semantic tags, know how and when to use them to maxmize the semantic strucure of your HTML