CS7026: Authoring for Digital Media

HTML5 – What’s New?
The Philosophies Behind HTML5

- Behind HTML5 is a series of stated design principles.
  - Compatibility
  - Utility
  - Interoperability
  - Universal Access

- Let’s look at what some of these mean in practice:
  - Specifying current browser behaviours that are interoperable
  - Defining error handling for the first time.
  - Evolving the language for easier authoring of web applications.
Many of the current methods of developing sites and applications rely on undocumented (or at least unspecified) features incorporated into browsers over time.

One of the first tasks of HTML5 was to document the undocumented.

This was to increase interoperability by leaving less to guesswork for web authors and browser producers.
Error Handling

- It was also necessary to unambiguously define how browsers and other user agents should deal with invalid markup.

- This wasn’t a problem in the XML world; XML specifies ‘draconian error handling’ in with the browser is required to stop rendering if it finds an error.

- However, one of the main reasons for the success of the web had been that even bad code was rendered by most browsers.
Error Handling

- But each browser was free to decide how to render bad code. In the interests of interoperability it is important that error handling be identical across browsers.

- The HTML5 specification is over 900 pages long when printed.

- However only 300 or so are relevant to web authors, the rest of it is for implementers of browsers, telling them exactly how to parse markup, even bad markup.
Web Applications

- An increasing number of sites on the Web are what we’ll call web applications.

- I.e., they mimic desktop apps rather than traditional static text-images-links documents.

- E.g. Online word processors, photo editing tools, mapping sites, etc.

- These are heavily powered by JavaScript and pushed HTML 4 and XHTML1.0 to the edge of their capabilities.
Web Applications

- HTML5 specifies new DOM APIs for drag and drop, server sent events, drawing, video, audio etc..

- These new interfaces that HTML pages expose to JavaScript via objects in the DOM make it easier to write such applications using tightly specified standards rather than barely documented hacks.

- This means that there is an open standard (free to use and implement) that competed with Adobe Flash or Microsoft Silverlight.
Don’t Break the Web

- It is important that the millions of web pages already out there continue to render.

- So HTML5 is (mostly) a superset of HTML 4 that continues to define how browsers deal with legacy markup such as `<font>` or `<center>` and other such presentational tags.

- But authors should not use them, as they’re obsolete.
Don’t Break the Web

- HTML5’s unambiguous parsing rules should ensure that ancient pages will work interoperably, as the HTML5 parser will be used for all HTML documents.

- All of the major browser vendors have been working on new parser implementations that comply with the HTML5 standard.
HTML5 Parser

- **Apple** began developing an HTML5 parser in WebKit in 2010 and deployed it to end users in **Safari 5.1** in 2011.

- **Google** shipped it in **Chrome 7** a few months after it was implemented in WebKit.

- **Mozilla** made an experimental HTML5 parser available behind an about:config option in Firefox 3.6, and finally stabilized it for Gecko 2, which was incorporated in **Firefox 4**.

- **Microsoft** shipped a HTML5 Parser in **Internet Explorer 10** (released with Windows 8 in September 2012).

- **Opera** first unveiled its HTML5-compatible parser in February 2011, which it released an experimental build. The new parser, which is codenamed Ragnarok, was finally integrated in **Opera 11.60**.
What About XML?

- HTML5 is not an XML language. It must be served as text/html.

- If, however, you need to use XML, there is an XML serialisation called XHTML5.

- This allows all the same features but requires a more rigid syntax (exactly the same as XHTML 1.0).

- It must be served with an XML MIME type so won’t be processed by IE8 and its antecedents.
Upgrading to HTML5

- Upgrading a page to HTML5 is as simple as changing your doctype.

- The doctype should already be on the first line of every HTML page.

- Previous versions of HTML defined a lot of doctypes, and choosing the right one could be tricky.
Upgrading to HTML5

- In HTML5, there is only one doctype (which we have already been using):
  
  ```html
  <!DOCTYPE html>
  ```

- Upgrading to the HTML5 doctype won’t break your existing markup, because all the tags defined in HTML 4 and XHTML 1.0 are still supported in HTML5.

- It will, however, allow you to use - and validate - new elements.
Main Structure

- Although a lot of HTML5 is for making interactive applications using JavaScript, there are also 28 new elements with new semantics that can be used in traditional ‘static’ pages.

- There are also a swathe of new form controls that can abolish JavaScript form validations completely (more anon...).
Main Structure

- We will look at marking up a typical page with HTML5.

- We’ll transform the current markup structure of `<div>`s into a more semantic system using new HTML5 structural elements like `<nav>`, `<header>`, `<footer>`, `<aside>` and `<article>`.

- We’ll look at how these work, and how HTML5 documents have an unambiguous outline and are more ‘semantic’.
An Aside – browser modes...

- There are 2 browser modes (kind of):
  - **Quirks Mode**: Browsers violate current Web format specifications in order to avoid “breaking” pages authored according to practices that were prevalent in the late 1990s.
  
  - **Standards Mode**: Browsers try to give conforming documents the specification-wise correct treatment to the extent implemented in a particular browser.

- (And then there is **Almost Standards Mode**: For more See Henri Sivonen’s “Activating Browser Modes with Doctype”).
The `<head>`

- A DOCTYPE is required by browsers to trigger standards mode, and this string is the shortest string that does this reliably.

- Then we need to define the document’s character encoding.

  `<meta charset=utf-8>`
This differs in 3 ways from previous versions of html:

1. The `<meta>` tag is much shorter than the one previously used (`<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />`). This is still possible, but the shorter way is preferred as it’s easier to type and works everywhere already.

2. The attribute `charset="utf-8"` is not quoted.

3. The tag `<meta charset=utf-8 />` is not self-closed.
Syntax

- HTML is not an XML language so you don’t need to do those things. But you can if you prefer.

- All of these are equally valid HTML5:
  - `<META CHARSET=UTF-8>`
  - `<META CHARSET=UTF-8 />`
  - `<META CHARSET="UTF-8">`
  - `<META CHARSET="UTF-8" />`
  - `<meta charset=utf=8>`
  - `<meta charset=utf=8 />`
  - `<meta charset="utf=8">`
  - `<meta charset="utf=8" />`
  - `<Meta CHARset=utfF-8>`
Syntax

Why such lax syntax?

Browsers never cared about XHTML syntax if it was sent as text/html – only the XHTML validator did.

Therefore favouring one form over the other in HTML5 would be entirely arbitrary and cause pages that didn’t follow the format to be invalid, although they would work perfectly in any browser.

So HTML5 is agnostic about which one you use.
Pick a Style and Stick with it

- Just because you can mix formats, doesn’t mean you should.

- That would prove a maintenance nightmare, particularly in a large team.

- Pick the style that works for you and stick with it.
Valid HTML5

- Let’s cheat and, after adding the document title we’ll go straight to content:

```html
<!doctype html>
<meta charset=utf-8>
<title>Interesting Articles</title>
<p>HTML5 proves to be easier to write than XHTML 1.0 - Shock!</p>
```

- If we validate this, we find that it validates fine, yet it has no `<html>` tag, no `<head>`, and no `<body>`
Valid HTML5

- It turns out that these three elements are entirely optional, because browsers assume them anyway.

- A quick glance under the browser hood confirms this:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Interesting blog</title>
  </head>
  <body>
    Today I had porridge for breakfast. 14 hours later, I went to bed.
  </body>
</html>
```
Valid HTML5

- Because browsers do this, HTML5 doesn’t require these tags.

- However, omitting these elements from your markup is likely to confuse your colleagues.

- Also, skipping the `<html>` tag hurts your screen reader users, as that’s where you set the primary language of the document:

  `<html lang=en>`
Valid HTML5

- This is important as the word six, for example, is pronounced differently depending on whether the language is English or French.

- Also, IE requires the `<body>` element before it will apply CSS to style new HTML5 elements.

- So, in the interest of maintainability, we’ll add those optional elements to make what’s probably the minimum maintainable HTML5 page:
Valid HTML5

```html
<!doctype html>
<html lang=en>
<head>
<meta charset=utf-8>
<title>Interesting Articles</title>
</head>
<body>
<p>HTML5 proves to be easier to write than XHTML 1.0 – Shock!</p>
</body>
</html>
```
HTML5 – New Structural Elements

- We are going to:
  - Structure the main sections of a web page using `<header>`, `<footer>`, `<nav>`, `<aside>`, and `<article>`,
  - Style the new elements with CSS.
HTML5 Structural Elements

- HTML 4 reflects the early Web of scientists and engineers. Whereas HTML5 reflects the Web as it was during its development.

- Hickson did an analysis of class names over 1 billion web pages to find out what the real web was made of. [http://code.google.com/webstats/2005-12/classes.html](http://code.google.com/webstats/2005-12/classes.html)

- So there are some new semantic structural elements, many of them inspired by these class names, because that’s what developers actually build.
HTML5 Structural Elements

- `<section>`
- `<nav>`
- `<article>`
- `<aside>`
- `<hgroup>`
- `<header>`
- `<footer>`
- `<time>`
- `<mark>`
Here is a sample home page marked up as we do in XHTML 1.0 using the semantically neutral `<div>` element:

```html
<div id="header">  
<h1>Interesting Articles</h1> 
</div> 

<div id="sidebar">  
<h2>Menu</h2>  
<ul>  
<li><a href="last-week.html">Last week</a></li>  
<li><a href="archive.html">Archives</a></li>  
</ul> 
</div> 

<div class="post">  
<h2>HTML5 Rocks</h2>  
<p>HTML5 proves easier to write than XHTML1.0 – Shock!…</p> 
</div> 

<div class="post">  
<h2>XHTML 2.0 Dies</h2>  
<p>XHTML 2.0 is no longer in development…</p> 
</div> 

<div id="footer">  
<p><small>CS7026: Authoring for Digital Media.</small></p> 
</div>
```
By applying some simple CSS to it, we’ll style it:

```
#sidebar {float:left; width:20%;}
.post {float:right; width:79%;}
#footer {clear:both;}
```
Structural Elements

- Diagrammatically, the page looks like this:
There is nothing at all wrong with this markup (and it’ll continue validating and working perfectly well in the new HTML5 world).

However, most of the structure is entirely unknown to a browser, as the only real HTML element we can use for these important page landmarks is the semantically neutral `<div>` (defined in HTML 4 as “a generic mechanism for adding structure to documents”).
It’s possible to imagine a clever browser having a shortcut key that would jump straight to the page’s navigation.

The question is: how would it know what to jump to? Some users use `<div class=“menu”>` , others use `class=“nav”` or `class=“navigation”` or `class=“links”` and that’s only in English.

HTML5 gives us new elements that unambiguously denote landmarks in a page.
<header>  <h1>Interesting Articles</h1>  </header>
<nav>  <h2>Menu</h2>
<ul>
<li><a href="last-week.html">Last week</a></li>
<li><a href="archive.html">Archives</a></li>
</ul>
</nav>
<article>  <h2>HTML5 Rocks</h2>
<p>HTML5 proves easier to write than XHTML1.0 — Shock!....</p>
</article>
<article>  <h2>XHTML 2.0 Dies</h2>
<p>XHTML 2.0 is no longer in development...</p>
</article>
<footer>
<p><small>CS7026: Authoring for Digital Media.</small></p>
</footer>
Semantic Elements

- Diagrammatically, the HTML5 version is this:
Styling HTML5 with CSS

- Before we look in detail at when to use these new elements, and what they mean, let’s first style the basic structures of the page.

- Styling these new elements is pretty simple.

- Theoretically you can apply CSS to any arbitrary element, because, as the spec says, CSS “is a style sheet language that allows authors and users to attach style … to structured documents (e.g., HTML documents and XML applications)” and XML applications can have any elements they want.
Styling HTML5 with CSS

- Although you can use the new HTML5 elements now, some older browsers don’t necessarily understand them.

- They don’t do anything special with them and treat them like unknown elements you make up.

- Therefore, using CSS we can float `<nav>`, put borders on `<header>` and `<footer>`, and give margins and padding to `<article>` almost as easily we can with `<div>`s.
However, by default, CSS assumes that elements are `display:inline`.

So if you just set heights and widths to the structural elements as we do `<div>`s, it won’t work properly in older browsers until we explicitly tell the browser that they are `display:block`. 

---

39
Styling HTML5 with CSS

- So, to style our HTML5 to match our HTML 4 design, we simply need the styles:

  ```css
  header, nav, footer, article{display:block;}
  nav {float:left; width:20%;}
  article {float:right; width:79%;}
  footer {clear:both;}
  ```

- And a beautiful HTML5 page is born.
When to Use the New Structural Elements

```html
<header>

- In our example above, as on most sites, the header will be the first element on a page, and contains the title of the site, logos, links back to the home page, etc.

- The specification says: “The header element represents a group of introductory or navigational aids ... Note: A header element is intended to usually contain the section’s heading (an h1–h6 element or an hgroup element), but this is not required. The header element can also be used to wrap a section’s table of contents, a search form, or any relevant logos.”
```
When to Use the New Structural Elements

- The first thing to note is that a header element is not required; in our example, it’s superfluous as it surrounds just the `<h1>`.

- Its value is that it groups “introductory or navigational” elements, so here’s a more realistic example:

```
<header>
  <a href="/"><img src="logo.png" alt="home"></a>
  <h1>Interesting Articles</h1>
</header>
```
When to Use the New Structural Elements

- Many websites have a title and a tagline or subtitle.

- To make the main heading and subtitle into one logical unit the
they can be grouped in the new `<hgroup>` element:

```html
<header>
  <a href="/"
       ><img src="logo.png" alt="home"></a>
  <hgroup>
    <h1>My Poetry</h1>
    <h2>A Collection of my Favourite Poems</h2>
  </hgroup>
</header>
```
When to Use the new Structural Elements

- The header can also contain navigation:

```html
<header>
  <a href="/"><img src="logo.png" alt="home"></a>
  <hgroup>
    <h1>My Poetry</h1>
    <h2>A Collection of my Favourite Poems</h2>
  </hgroup>
  <nav>
    <ul>
      <li>Home</li>
      <li><a href="authors.html">Authors</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>
  </nav>
</header>
```
When to Use the new Structural Elements

<nav>

- The `<nav>` element is designed to mark up navigation.

- Navigation is defined as being links around a page (e.g., a table of contents at the top of an article that links to anchor points on the same page) or within a site.

- As with all of the new structural elements, you’re not restricted to one `<nav>` per page.

- You might very well have site-wide `<nav>` in a header, a `<nav>` which is a table of contents for the current content, and a `<nav>` below that which links to other related content on your site.
When to Use the New Structural Elements

- It often makes sense to use a list for your navigation as it both semantically correct and it gives you more hooks for CSS (see http://www.alistapart.com/articles/taminglists/).

- However it’s not mandatory. This is perfectly valid:

```html
<nav>
  <p><a href="/">Home</a></p>
  <p><a href="/about">About</a></p>
</nav>
```
When to Use the New Structural Elements

- You can include headings for navigation, too:

  ```html
  <nav>
    <h2>Main navigation</h2>
    <ul>
      <li><a href="/about">About me</a></li>
      <li><a href="/news">News</a></li>
    </ul>
  </nav>
  ```
When to Use the New Structural Elements

Grouping `<nav>` and other elements

- Many sites have a sidebar that includes multiple blocks of navigation and other non-navigation content.

- The `<nav>` elements contained in the new `<aside>` element “can be used for typographical effects like pull quotes or sidebars, for advertising, for groups of nav elements, and for other content that is considered separate from the main content of the page.”

http://dev.w3.org/html5/spec/semantics.html#the-aside-element.
When to Use the New Structural Elements

<aside>
<nav>
<h2>Pages</h2>
<ul>
  ...
</ul>
</nav>

<nav>
<h2>Recent comments</h2>
<ul>
  ...
</ul>
</nav>

<section>
<h2>blah blah</h2>
<a>Web hosting by LovelyHost</a>
<img src="...">
<p>Powered by <a>WordPress</a></p>
<p><a>Comments (RSS)</a></p>
</section>
</aside>
When to Use the New Structural Elements

```html
<footer>
```

- Defined as representing “a footer for its nearest ancestor sectioning content or sectioning root element.” (“Sectioning content” includes article, aside, nav, section, and “sectioning root elements” are blockquote, body, details, fieldset, figure, td).

- Again, there can be more than one footer on a page.

- The spec continues “A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like.”
When to Use the New Structural Elements

- Our footer holds information which you can wrap in a `<small>` element, too.

- `<small>` has been redefined in HTML5; previously it was a presentational element, but in HTML5 it represents small print.

- “Small print typically features disclaimers, caveats, legal restrictions, or copyrights. Small print is also sometimes used for attribution, or for satisfying licensing requirements.”
When to Use the New Structural Elements

<article>

> The main content of our page contains a few articles. We wrap each one up in an <article> element.

> <article> is specified thus: “The article element represents a component of a page that consists of a self-contained composition in a document, page, application, or site and that is intended to be independently distributable or reusable, e.g., in syndication.”

> Basically some stand-alone content.

> A blog post, a tutorial, a news story, comic strip, or a video with its transcript all fit perfectly into this definition.
When to Use the New Structural Elements

- Note that, as with `<nav>`, the heading goes inside the element, so...

  `<h1>My article</h1>`
  `<article>`
  `<p>Blah blah</p>`
  `</article>`

  is incorrect; it should be

  `<article>`
  `<h1>My article</h1>`
  `<p>Blah blah</p>`
  `</article>`