CS7026: Authoring for Digital Media

HTML5: Main Structure
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.

- look at HTML5 Shiv.
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 XHTML 1.0 – 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>
```
Structural Elements

- By applying some simple CSS to it, we’ll style it:
  ```css
  #sidebar {float:left; width:20%;}
  .post {float:right; width:79%;}
  #footer {clear:both;}
  ```
Structural Elements

Diagrammatically, the page looks like this:

- **div id="header"**
- **div id="sidebar"**
  - **div class="post"**
  - **div class="post"**
- **div id="footer"**
Structural Elements

- 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.
<h1>Interesting Articles</h1>

<h2>Menu</h2>

<ul>
  <li><a href="last-week.html">Last week</a></li>
  <li><a href="archive.html">Archives</a></li>
</ul>

<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.
Styling HTML5 with CSS

- 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`.
Styling HTML5 with CSS

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

<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:

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

<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...

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

is incorrect; it should be

```html
<article>
  <h1>My article</h1>
  <p>Blah blah</p>
</article>
```
HTML Shiv (an important aside...)

Shiv (or Shim): an application compatibility workaround
HTML5 Shiv

- Every browser has a list of HTML elements that it supports.

- Elements not on this list are treated as “unknown elements”.

- There are 2 fundamental questions regarding these “unknown elements”
  - How should the element be styled?
  - What should the element’s DOM look like?
The Document Object Model (briefly)

- The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents.

- It defines the logical structure of documents and connects web pages to scripts or programming languages.

- In the DOM, documents have a logical tree-like structure.
Mozilla Firefox’s list of elements is stored in `nsElementTable.cpp`.

This includes information about what kinds of other elements each element can contain.

E.g. If you include markup like `<p><p>`, the second paragraph element implicitly closes the first one, so the elements end up as siblings, not parent and child.

But if you write `<p><img>`, the image does not close the paragraph, because Firefox knows that `<p>` is a block element that can contain the inline element `<img>`. So the `<img>` ends up as a child of the `<p>` in the DOM.
However, when it comes to “unknown elements” different browsers answer these questions in different ways.

Of the major browsers, Microsoft Internet Explorer’s answer to both questions was the most problematic.

The first question should be relatively simple to answer: don’t give any special styling to unknown elements. Just let them inherit whatever CSS properties are in effect wherever they appear on the page, and let the page author specify all styling with CSS.
Unfortunately, Internet Explorer (prior to Version 9) does not allow styling on unknown elements. For example, if you had this markup:

```html
<style type="text/css">
article {display: block; border: 1px solid red }
</style>
...  
<article>
<h1>Welcome to TCD</h1>
<p>This is your <span>first day</span>.</p>
</article>
```

Internet Explorer (up to and including IE 8) will not put a red border around the article.
HTML5 Shiv

- The second problem is the DOM that browsers create when they encounter unknown elements.

- If IE doesn’t explicitly recognize the element name, it will insert the element into the DOM as an empty node with no children.

- All the elements that you would expect to be direct children of the unknown element will actually be inserted as siblings instead.
So this is the DOM that HTML5 dictates:

```
article
 | 
 +-h1 (child of article) 
 | 
 | +-text node "Welcome to TCD"
 | 
 +-p (child of article, sibling of h1) 
 | 
 +-text node "This is your "
 | 
 +-span 
 | 
 | |-text node "first day"
 | 
 |+-text node "."
```
But this is the DOM that Internet Explorer actually creates:

```html
article (no children)
| h1 (sibling of article)
  | text node "Welcome to TCD"
  | p (sibling of h1)
  | text node "This is your "
  | span
  | text node "first day"
  | text node "."
```
However, if you create a dummy `<article>` element with JavaScript before you use it in your page, Internet Explorer will magically recognize the `<article>` element and let you style it with CSS.
HTML5 Shiv

```html
<html>
<head>
<script>
document.createElement("article");
</script>
</head>
<body>
<article>
<h1>Welcome to TCD</h1>
<p>This is your <span>first day</span>.</p>
</article>
</body>
</html>
```
HTML5 Shiv

- This works in all versions of Internet Explorer, all the way back to IE 6.

- We can extend this technique to create dummy copies of all the new HTML5 elements and then just start using them without having to worry too much about non-HTML5-capable browsers.
HTML5 Shiv

- Remy Sharp has done just that, with his aptly named “HTML5 enabling script” or HTML5 shiv

- The script has gone through several revisions, but this is the basic idea:

```html
<!--[if lt IE 9]>
<script>
var e = ("abbr,article,aside,audio,canvas,datalist,details," + "figure,footer,header,hgroup,mark,menu,meter,nav,output," + "progress,section,time,video").split(',');
for (var i = 0; i < e.length; i++) {
document.createElement(e[i]);
}
</script>
<!-[endif]-->
```
The `<!--[if lt IE 9]>` and `<![endif]-->` bits are conditional comments.

Internet Explorer interprets them like an if statement: “if the current browser is a version of Internet Explorer less than Version 9, then execute this block.”

Every other browser will treat the entire block as an HTML comment. The net result is that Internet Explorer (up to and including Version 8) will execute this script, but other browsers will ignore it altogether.

This makes your page load faster in browsers that don’t need this hack.
The JavaScript code itself is relatively straightforward:

1. The variable `e` ends up as an array of strings like "abbr", "article", "aside", and so on.
2. Then we loop through this array and create each of the named elements by calling `document.createElement()`.

This is enough to get Internet Explorer to treat these elements the way we want them to be treated when we actually use them later in the page.
That “later” bit is important. This script needs to be at the top of your page - preferably in your `<head>` element - not at the bottom.

That way, Internet Explorer will execute the script `before it parses your tags and attributes`.

If you put this script at the bottom of your page, it will be too late. Internet Explorer will have already misinterpreted your markup and constructed the wrong DOM, and it won’t go back and adjust it just because of this script.
HTML5 Shiv

- Remy Sharp has “minified” this script and hosted it on Google Project Hosting.

- The script itself is open source and MIT-licensed, so you can use it in any project.

- If you like, you can even “hotlink” the script by pointing directly to the hosted version, like this:

```
<head>
<meta charset="utf-8" />
<title>My HTML5 Page</title>
<!--[if lt IE 9]>
<![endif]-->
</head>
```