CS7026: Authoring for Digital Media

HTML5
We are going to:

Look at the structural elements a bit more closely by constructing a blog page
We have seen how a document’s outline can be structured, now we will look deeper to show how you can further structure your main content.

Let’s mark up a typical blog with HTML5. It’s a good archetype of modern websites with headers, footers, sidebars, multiple navigation areas, and a form, whether it’s a blog, a news site, or a brochure site (with products instead of news pieces).
Structuring Main Content Areas

- Take a look at the main content area of a blog. There may be multiple articles, each containing “metadata” and the actual textual content of that article.
Here’s some typical markup (simplified from a default WordPress theme)

```html
<div class="post">
<h2>Memoirs of a Dublin delinquent</h2>
<small>December 1st, 2016</small>
<div class="entry">
<p>Molly Malone’s poignant autobiography is this winter’s must-read.</p>
</div>

<p class="postmetadata">Posted in <a href="/?cat=3">Books category</a> | <a href="/?p=34#respond">No Comments</a></p>
</div>
```
Structuring Main Content Areas

- There is nothing major wrong with this markup. It will work fine in “HTML5” browsers.

- But apart from the heading for the blog post, there is no real structure - just meaningless `<div>`s and paragraphs.

- XHMTL1.0 gives us generic structures to mark up content. `<div>`, for example, is just a generic “box” that tells the browser “here’s some stuff , it all belongs together,” but it doesn’t mean anything; there’s no semantic value beyond “these belong together.”
Structuring Main Content Areas

- Where possible, we’ll replace generic boxes with new HTML5 elements, while still using `<div>` where there isn’t an appropriate element.
Structuring Main Content Areas

- As we have already seen you can replace the outer `<div class=”post”>` with `<article>`.

- But you can go further. The HTML5 `<header>` and `<footer>` elements can be used multiple times on a page, each time referring to the section it’s in.

- The heading and the time of posting is “introductory matter” and thus the job for `<header>`.

- Similarly, the metadata about the post that is currently in a paragraph with class “=postmetadata” is better marked up in HTML5 as a `<footer>`, which “typically contains information about its section, such as who wrote it, links to related documents, copyright data, and the like.”
Diagrammatically, the revised structure is:

```html
<article>
  <header>
    <h2>Memoirs of a Dublin delinquent</h2>
    <time datetime="2016-12-01">December 1st, 2016</time>
  </header>
  <p>Molly Malone’s poignant autobiography is this winter’s must-read.</p>
  <footer>
    Posted in <a href="/?cat=3">Books category</a>.<a href="/?p=34#respond">No Comments</a>
  </footer>
</article>
```
The `<time>` element

- `<time>` is a new element for unambiguously encoding dates and times for machines, while still displaying them in a human-readable way.

- The uses of this in web pages aren’t hard to imagine:
  - a browser could offer to add future events to a user’s calendar;
  - content aggregators could produce visual timelines of events;
  - a Thai-localised browser could offer to transform dates into Thai Buddhist era dates, which are numerically 543 years greater than their corresponding Western-style years.

- The spec says “The time element represents either a time on a 24-hour clock, or a precise date in the proleptic Gregorian calendar, optionally with a time and a time-zone offset.”
The `<time>` element

- The machine-readable part of the `<time>` element is usually encapsulated in the element’s `datetime` attribute.

- The content inside the element is what gets presented to end users.
  - `<time datetime=2016-11-14>14 November 2016</time>`
  - `<time datetime=2015-11-14>14<sup>rd</sup> November last year</time>`
  - `<time datetime=2016-11-14>Molly’s 21st birthday</time>`
  - `<time datetime=2016-11-14T02:00Z>8PM on Molly’s birthday</time>`
  - `<time datetime=20:00>8 PM</time>`
The `<time>` element

- If you’re happy to have the machine-readable format visible to the end user as well, you don’t need to use a separate `datetime` attribute.

- User agents should then simply pick the content of the element and interpret it: `<time>20:00</time>`
Machine-Readable Dates and Times

- To be machine-readable, dates must be in the format YYYY-MM-DD and may also include a time, prefixed with “T” to separate the date and time, in the format HH:MM.

- Optionally you can append seconds (separated from the minutes with a colon).

- Fractions of a second are allowed after a full stop mark.

- As you’ve seen above, you can give a time on the 24-hour clock with no date information.
The `<time>` element

- If you’re giving time and date together, you need to show the time zone: that’s either “Z” for Coordinated Universal Time (UTC), or an offset from UTC in hours and minutes, prefixed with a plus or minus.

- Putting that all together: “1979-10-14T12:00:00.001-04:00” represents one millisecond after noon on October 14th, 1979, in Eastern Standard Time during daylight saving time (UTC - 4 hours).
The `pubdate` attribute

- `pubdate` is a Boolean attribute to indicate that this particular `<time>` is the publication date of an `<article>` or the whole `<body>` content.

- Why not just assume that any `<time>` element in an `<article>`’s `<header>` is its publication date?
The **pubdate** attribute

- Consider this example:

```html
<article>
<header>
  <h1>Come to my party on <time datetime=2016-12-01>1 December</time></h1>
  <p>Published on <time datetime=2016-06-20 pubdate>20 June 2016</time></p>
</header>
<p>I’m throwing a party for my birthday</p>
</article>
```

- There are two dates within the `<header>`: the date of the actual party and the publication date of the article. The **pubdate** attribute is required to remove any ambiguity.
Styling Multiple Headers and Footers

- This main change with our article makeover is that each article can have its own `<header>` and `<footer>`.

- This means that, in addition to the “main” header and footer on a page, each article can have its own headers and footers.

- They can be separately styled with CSS: `body header` and `body footer` target the “main” headers and footers (assuming that they’re direct descendants of `<body>`), whereas `article header` and `article footer` target the inner structures.
Styling Multiple Headers and Footers

- Define generic header and footer styles, and then redefine/override them for article header and article footer:

```css
header {display:block; color:red; text-align:right;} /*page header */
article header {color:blue; text-align:center;} /*article header */
```

- Note that so far, you’ve introduced no ids or classes as hooks for CSS.
Using multiple `<footer>`s on the same element

- The spec says “Footers don’t necessarily have to appear at the end of a section, though they usually do,” and it allows an element to have two or more footers.

- A simplified version of the example in the spec is

  ```html
  <body>
  <footer><a href="/">Back to index...</a></footer>
  <h1>Lorem ipsum</h1>
  <p>Lorem ipsum</p>
  <footer><a href="/">Back to index...</a></footer>
  </body>
  ```
Using multiple `<footer>`s on the same element

- The reason for this is that the elements are supposed to be non presentational.

- If “back to index” is the footer below the article, and you choose to have “back to index” above the article, too, you should use the same element for the same content, regardless of where it appears.
Adding Blogposts and Comments

- If we add this to our page, we have a page with a header, footer, navigation, content area containing several articles (blog posts), each with its own header and footer.

- But what about comments?

- The specification recommends the use of nested `<article>`s: “When article elements are nested, the inner article elements represent articles that are in principle related to the contents of the outer article. For instance, a blog entry on a site that accepts user-submitted comments could represent the comments as article elements nested within the article element for the blog entry.”
Adding Blogposts and Comments

- So let’s do that. Blog comments are typically shown in chronological order and have information such as author’s name and URL—in short, header information:
Come to my party on 1 December

Published on 20 June 2016

I’m throwing a party for my birthday

Published in the Party category by Nina

Comment from Mary B at 8.45 on 1 May 2016

I’ll be there. I very much enjoy parties

Comment from John B at 10.45 on 2 May 2016

Sorry. Am washing my hair.
Even More New Structures!

<aside>

- We’ve used `<aside>` to mark up sidebars. It represents “a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography. The 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.”

- Using an `<aside> inside` an `<article>`, for example, is the right place for tangentially related information or pull quotes about that article, but not, let’s say, page-wide navigation.
Even More New Structures

<details> (not fully supported)

- Introduces native support for a common behaviour - providing an expanding/collapsing area - removing the need for custom JavaScript

<details>
<summary>Photograph details</summary>
<p>Photograph taken on <time datetime=2010-12-25>Christmas Day 12</time> with a Canon IXUSi.</p>
<p><small>Copyright TCD, <address>nina@nina.com</address>.</small></p>
</details>
Even More New Structures!

- The contents of the descendent `<summary>` element are focusable and act as a control that, when activated by mouse or keyboard, expand or collapse the remainder of the element.

- If no `<summary>` element is found, the browser supplies its own default control text, such as “details” or a localised version.

- Browsers add some kind of icon to show that the text is “expandable” such as the arrow in Chrome.
Even More New Structures!

- `<details>` can optionally take the `open` attribute to ensure that the element is open when the page is loaded:
  
  `<details open>`

- Note that the element isn’t restricted to purely textual markup— it could be a login form, an explanatory video, a table of source data for a graph...
Even More New Structures!

<figure>

- Previously, the only way to add a caption to a picture to explain the image or to give attribution to the photographer, has been with text that runs into surrounding content, with no way to explicitly associate it with the image.

- The <figure> element wraps an image (or a video, or block of code, or a supporting quotation) and its caption, which goes in the <figcaption> element:
Even More New Structures!

```html
<figure>
   <img src=turtle.jpg alt=""/> <!-- no alt, as it’s covered by the figcaption -->
   <figcaption>
   A picture of a green sea turtle
   <small>Photo &copy; Mary B</small>
   </figcaption>
</figure>
```
Even More New Structures!

<mark>

The <mark> element allows you to do the markup equivalent of using a highlighter pen to bring out some words on a printed page.

It’s not the same as emphasis - for that you use <em>. But if you had some existing text and wanted to bring something to the fore that isn’t emphasised in the text, you could use <mark> and style it to be italics, or with a yellow highlighter-pen background colour.

In print, you’ll often see the phrases “my italics” or “emphasis added”.
Even More New Structures!

- The spec also says “When used in the main prose of a document, it indicates a part of the document that has been highlighted due to its likely relevance to the user’s current activity.”
Redefined Elements

- HTML5 redefines some existing elements as well as adding new ones.

- Some have radically changed, others have just been tweaked.

```html
<ol>
  
  - In HTML 4, the start attribute on `<ol>` was deprecated, as it was deemed presentational.

  - HTML5 reverts this decision. If you want an ordered list to start at five rather than line one, use: `<ol start=5>`
```
Redefined Elements

- Something nice that isn’t yet fully implemented is the `reversed` attribute. Consider the following example:
  ```html
  <h3>Top five ice-cream flavours</h3>
  <ol reversed>
    <li>Vanilla</li>
    <li>Strawberry</li>
    <li>Lemon</li>
    <li>Mint</li>
    <li>Chocolate</li>
  </ol>
  ```

- This creates a list that counts down from five (Vanilla) to one (Chocolate).
Redefined Elements

<cite>

In HTML 4, the <cite> element could be used to mark up the name of a speaker: As <cite>Harry S. Truman</cite> said,<q lang="en-us">The buck stops here.</q>

HTML5 disallows this: “A person’s name is not the title of a work - even if people call that person a piece of work - and the element must therefore not be used to mark up people’s names.”

This makes existing content that conforms to the rules of HTML 4 nonconforming to the rules of HTML5, although it will never be flagged as invalid by a validator, as a machine has no way of knowing that “Harry S. Truman” is a name rather than the title of a biography called “Harry S. Truman.”

In his 24ways.org article, “Incite a riot”, Jeremy Keith wrote “Join me in a campaign of civil disobedience against the unnecessarily restrictive, backwards-incompatible change to the cite element.”
Redefined Elements

<address>

- <address> is for contact details of the author of the current <article> or document, not as a generic element for postal addresses.

- The contact details can be email address, postal address, or any others.

- What’s new is that you can have multiple addresses in a document, one inside each <article>.

- Author information associated with an article element does not apply to nested article elements, so a blog post in an <article> can have an <address> for its author, and each blog comment (which you remember is a nested <article>) can have the <address> of its commenter.
Redefined Elements

<em>, <i>

- <em> marks up emphasis of the kind that subtly changes the meaning of a sentence; if the question is “Did you say you live in Paris?” the answer might be marked up as:

  <p>No, my <em>name</em> is Paris. I live in <em>Troy</em>.</p>

- If you have relative levels of importance, you can nest <em> elements to make the contents extra emphatic.
Redefined Elements

- The `<i>` element “represents a span of text in an alternate voice or mood, or otherwise offset from the normal prose, such as a taxonomic designation, a technical term, an idiomatic phrase from another language, a thought, a ship name, or some other prose whose typical typographic presentation is italicized.”

- Here are some examples of `<i>` where `<em>` would not be appropriate:
  - `<p>`The `<i>Titanic</i> sails at dawn.</p>`
  - `<p>`The design needs a bit more `<i lang=fr>ooh la la</i>`. </p>`
Redefined Elements

<strong>, <b>

- The strong element represents strong importance for its contents but, unlike <em>, it does not change the meaning of the sentence.

- E.g., <p><strong>Warning! This banana is dangerous.</strong></p>

- You can nest strong elements to make them extra-important.
The `<b>` element “represents a span of text to be stylistically offset from the normal prose without conveying any extra importance, such as key words in a document abstract, product names in a review, or other spans of text whose typical typographic presentation is boldened.”

For example:

```html
<p>John never forgot his fifth birthday - feasting on <b>Heinz beans</b> and the joy of opening his gift: a <b>Beano!</b> annual.</p>
```
Redefined Elements

The `<hr>` element is now media-independent and indicates “a paragraph-level thematic break.”

A comment on HTML5doctor put it nicely: It’s the markup equivalent of the “***” that is often used in stories and essays.
The `<small>` element has been completely redefined, from simply being a generic presentational element to make text appear smaller to actually represent “small print,” which “typically features disclaimers, caveats, legal restrictions, or copyrights. Small print is also sometimes used for attribution, or for satisfying licensing requirements.”

If the whole page is a “legalese” page, don’t use `<small>`.

`<small>` has no bearing on `<strong>` or `<em>` elements.
Some elements you may know from HTML4 have been made completely obsolete in HTML5, such as `<applet>` (use `<embed>` instead), `<big>`, `<blink>`, `<center>`, `<font>`, and `<marquee>`.

They will not validate and must not be used by authors.

Frames are gone (but `<iframe>` remains).

HTML5 browsers must still render these elements, of course, as there are plenty of them still out there. But you must avoid them.
Lab Exercise

- Take your current website and starting with the homepage and, in pairs, analyse the content to see how you could structure it using HTML5.

- Save it under a new name and rewrite the code using HTML5 structural elements.

- Save your CSS file under a new name and rewrite it to reflect the changes in your CSS.

- Link your new HTML5 page to your new CSS file.