CS7026: Authoring for Digital Media

HTML5: The History of the Future
XHTML1.0 and beyond...

- The content of the XHTML 1.0 specification was identical to that of HTML 4.01.

- No new elements or attributes were added. The only difference was in the syntax of the language.

- Whereas HTML allowed authors plenty of freedom in how they wrote their elements and attributes, XHTML required authors to follow the rules of XML (a stricter markup language) upon which the W3C was basing most of their technologies.
XHTML 1.0 and beyond...

- Having stricter rules was generally seen to be a good thing. It encouraged a generation of developers to think about valid well-structured code.

- The publication of XHTML 1.0 coincided with the rise of browser support for CSS.

- As web designers embraced the emergence of web standards, led by The Web Standards Project, the stricter syntax of XHTML was viewed as a “best practice” way of writing markup.
XHTML 1.1

- Then the W3C published XHTML 1.1.

- While XHTML 1.0 was simply HTML reformulated as XML, XHTML 1.1 was real, honest-to-goodness XML.

- That meant it couldn’t be served with a mime-type of text/html.
An aside...

- MIME stands for Multi-purpose Internet Mail Extensions.

- MIME types form a standard way of classifying file types on the Internet.

- Internet programs such as Web servers and browsers all have a list of MIME types, so that they can transfer files of the same type in the same way, no matter what operating system they are working in.

- A MIME type has two parts: a type and a subtype. They are separated by a slash (/). E.g, the MIME type for Microsoft Word files is application and the subtype is msword. Together, the complete MIME type is application/msword.
But if you published a document with an XML mime-type, then the most popular web browser in the world at the time - Internet Explorer - couldn’t render the document.

It seemed as if the W3C were losing touch with the day-to-day reality of publishing on the web.
XHTML 1.1 and Beyond

- As far as the W3C was concerned, HTML was finished as of version 4.

- They began working on XHTML 2, designed to lead the web to a bright new XML-based future.

- Although the name XHTML 2 sounded very similar to XHTML 1, they couldn’t have been more different.
Unlike XHTML 1, XHTML 2 wasn’t going to be backwards compatible with existing web content or even previous versions of HTML.

Instead, it was going to be a pure language, more logical, better-designed, unburdened by the sloppy history of previous specifications.
It was a disaster.
The Schism: WHATWG

- A rebellion formed within the W3C. The consortium seemed to be formulating theoretically pure standards unrelated to the needs of web designers.

- Representatives from Opera, Apple, and Mozilla were unhappy with this direction. They wanted to see more emphasis placed on formats that allowed the creation of web applications.
The Schism: WHATWG

- Things came to a head in a workshop meeting in 2004. Ian Hickson, who was working for Opera Software at the time, proposed the idea of extending HTML to allow the creation of web applications. The proposal was rejected.

- The disaffected formed their own group: the Web Hypertext Application Technology Working Group, or WHATWG for short.

- Consisted of groups from Opera and Mozilla with Apple cheering by the sideline.
From the start, the WHATWG operated quite differently than the W3C.

The W3C uses a consensus-based approach: issues are raised, discussed, and voted on.

At the WHATWG, issues are also raised and discussed, but the final decision on what goes into a specification rests with the editor. The editor is Ian Hickson (who moved to Google).
From Web Apps 1.0 to HTML5

- On the face of it, the W3C process sounds more democratic and fair.

- In practice, politics and internal bickering can bog down progress.

- At the WHATWG, where anyone is free to contribute but the editor has the last word, things move at a faster pace.

- But the editor doesn’t quite have absolute power: an invitation-only steering committee can impeach him.
Initially, the bulk of the work at the WHATWG was split into two specifications: Web Forms 2.0 and Web Apps 1.0.

Both specifications were intended to extend HTML.

Over time, they were merged into a single specification called simply HTML5.
Reunification

- While HTML5 was being developed at the WHATWG, the W3C continued working on XHTML 2.

- It wasn’t going well.

- In October 2006, Sir Tim Berners-Lee wrote a blog post in which he admitted that the attempt to move the web from HTML to XML just wasn’t working.

  “The attempt to get the world to switch to XML, including quotes around attribute values and slashes in empty tags and namespaces, all at once didn’t work.”
Reunification

- A few months later, the W3C issued a new charter for a HTML Working Group.

- Rather than start from scratch, they wisely decided that the work of the WHATWG should be used as the basis for any future version of HTML.
Reunification

- All of this stopping and starting led to a somewhat confusing situation.

- The W3C was simultaneously working on two different, incompatible types of markup: XHTML 2 and HTML 5 (note the space before the letter five).

- Meanwhile a separate organization, the WHATWG, was working on a specification called HTML5 (with no space) that would be used as a basis for one of the W3C specifications!
WHATWG Process

- The WHATWG process has been extraordinarily open—anyone could join the WHATWG mailing list and contribute to the spec.

- Every email was read by Hickson or the core WHATWG team (included Brendan Eich (inventor of JavaScript and Mozilla CTO), David Hyatt (Safari and WebKit Architect) and Hakon Wium Lie (Opera CTO)).

- Good ideas were implemented and bad ideas rejected regardless of the source.

- Good ideas were adopted from Twitter, Blogs, IRC...
The fog of confusion began to clear in 2009. The W3C announced that the charter for XHTML 2 would not be renewed.

The format had been as good as dead for several years; this announcement was little more than a death certificate.
Long Live XHTML Syntax

- Meanwhile, authors who had been writing XHTML 1 in order to enforce a stricter writing style became worried that HTML5 would herald a return to sloppy markup.

- As you’ll see, that’s not necessarily the case. HTML5 is as sloppy or as strict as you want to make it.
Not so happy ever after…

- For a number of years, both groups then worked together.

- In 2011, however, the groups came to the conclusion that they had “different goals”:
  - “the W3C wanted to publish a "finished" version of "HTML5", while the WHATWG wanted to continue working on a Living Standard for HTML, continuously maintaining the specification rather than freezing it in a state with known problems, and adding new features as needed to evolve the platform” (WHATWG).
Not so happy ever after…

- “Since then, the WHATWG has been working on this specification (amongst others), and the W3C has been copying fixes made by the WHATWG into their fork of the document, as well as making other changes, some intentional and some not, with no documentation listing or explaining the differences.”

- [http://developers.whatwg.org/introduction.html](http://developers.whatwg.org/introduction.html)
The Timeline of HTML5

- This left two groups working on HTML5:
  - The WHATWG creating an HTML5 specification using its process of “commit then review.”
  - The W3C HTML Working Group taking that specification and putting it through its process of “review then commit.” As you can imagine, it was an uneasy alliance.

- Still, there seemed to finally be some consensus about that “space or no space?” question (it’s no space).
The Timeline of HTML5

- In September 2014, W3C moved HTML5 to Proposed Recommendation.

- On 28 October 2014, HTML5 was released as a stable W3C Recommendation, meaning the specification process is complete.

- However there wasn’t a single point in time that it was declared that the language was ready to use. Instead, parts of the specification have been used as web browsers started to support those features (around 2010).
The Timeline of HTML5

- Remember, HTML5 isn’t a completely new language created from scratch. It’s an evolutionary rather than revolutionary change in the ongoing story of markup.

- If you are currently creating websites with any version of HTML, you’re already using HTML5.
The Philosophies Behind HTML5

- Behind HTML5 is a series of stated design principles.

- There are 3 main aims:
  - Specifying current browser behaviours that are interoperable
  - Defining error handling for the first time.
  - Evolving the language for easier authoring of web applications.
Existing Web Pages

- 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 implementors 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, 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 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:
  ```html
  <!DOCTYPE html>
  ```

- Upgrading to the HTML5 doctype won’t break your existing markup, because all the tags defined in HTML 4 and XMLTL 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’.
The `<head>`

- First the DOCTYPE:
  
  ```html
  <!doctype html>
  ```

- That’s it.

- No URLs.

- No version number.

- That is all.
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.

```html
<!doctype html>
<meta charset=utf-8>
```
You may notice 3 oddities:

1. The `<meta>` tag is much shorter than the one we’ve used before (`<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=utF-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:
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>
```