Comp 336/436 - Markup Languages

Fall Semester 2019 - Week 3

Dr Nick Hayward

Semantic HTML - intro

- importance of web standards
  - *and their application to HTML markup and documents*
- standards help drive a consideration of markup, e.g. HTML
  - *usage for what they mean*
  - *not simply how they will look...*
- semantic instead of purely presentational perspective
  - *introduction of meaning and value to the document*
- when pages are processed
  - *impart structure and meaning beyond mere presentation*
- a core consideration for usage of markup languages
- issues persist with HTML element usage
  - *e.g. inline elements such as `<b>` and `<i>`*

Semantic HTML - a reason to care

- Semantic HTML - opportunity to convey meaning with your markup
  - *meaning may be explicit due to the containing element*
  - *implicit due to a structured grouping of elements*
- markup makes it explicit to the browser
  - *underlying meaning of a page and its content*
- notion of meaning and clarity also conveyed to search engines
  - *fidelity with query and result...*
- semantic elements provide information beyond page rendering and design
- use semantic markup correctly
  - *create more specific references for styling*
  - *greater chance of rendering information correctly*

Semantic HTML - example usage

```
<!-- incorrect element chosen -->
<div id="code">
document.addEventListener('click', function () {
  console.log('Click received...');
});
</div>
```

```
<!-- correct element chosen -->
<code>
document.addEventListener('click', function () {
  console.log('Click received...');
});
</code>
```

- semantic example usage

Semantic HTML - correct usage

- need to ensure elements convey their correct meaning
  - *i.e. the meaning expected for the contained content*
- e.g. often see the following elements mis-used and applied incorrectly for markup,
  - *<p> - paragraphs*
  - *<ul - unordered list*
  - *<h1> to <h6> - headings*
  - *<blockquote> - blockquote*
- using <blockquote> to simply help indent text
  - *instead of CSS margins...*
- or the perennial mis-use of a <p>
  - *simply add extra space between elements*

```
<p> <p>
```

HTML - structure & validation - example

Using lists correctly...

```
<li>nice</li>
<li>cannes</li>
<li>menton</li>
```

- list markup looks OK
  - *still fails validation for an obvious reason*
  - *missing structural grouping for list items*
  - *not valid markup...*
- semantics of the overall list are missing
- example - basic list items

HTML - a semantic point of view

```
<ul>
  <li>nice</li>
  <li>cannes</li>
  <li>menton</li>
</ul>
```

- from the perspective of semantics
  - *meant to act as a group of items that belong together*
- denote such groupings with correct semantic markup
- structuring items to clearly denote their meaning and purpose
- consider global attributes
  - *https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes*
- example - basic group

HTML - semantics & XML

```
<places>
 <item>nice</item>
 <item>cannes</item>
 <item>menton</item>
</places>
```

- XML example of markup for places group and items
- entire list has a containing element, `<places>`
- grouping items has a number of benefits
  - *comprehension, legibility*
  - *ease of processing...*
- XSLT processing into HTML, plain text, PDF...
- XML principles incorporated into XHTML and HTML5
- example - basic xml list

HTML - benefits of structure & validation

- define and create a meaningful structure for required markup
  - *improves usage and flexibility as project develops*
  - *provides extensible structure for project*
- for example, benefits include
  - *helps increase ease of CSS styling*
  - *creates properly structured documents*
  - *improves general management of updates to markup*
  - ...
- easier to understand and easier to maintain and update
- structured, valid markup aids in repurposing data
  - *into various representations of information*

HTML - benefits of structure & validation - example 1

e.g. a standard list

```
<ul>
  <li>nice</li>
  <li>cannes</li>
  <li>menton</li>
  <li>antibes</li>
  <li>grasse</li>
</ul>
```

- example - basic group style

HTML - benefits of structure & validation - example 2

e.g. lists for navigation, menus, tabs...

```
<ul id="menutabs">
  <li><a href="nice">nice</a></li>
  <li><a href="cannes">cannes</a></li>
  <li><a href="menton">menton</a></li>
  <li><a href="antibes">antibes</a></li>
  <li><a href="grasse">grasse</a></li>
</ul>
```

- example - basic menu tabs

HTML - markup for headings - part 1

- HTML is flexible in markup usage
  - *due to presentational versus structural considerations*
- headings might be perceived as purely presentational, e.g.

```
<span class="heading">Chapter 1</span>
```

- issues with presentational markup, e.g.
  - *visual browsers with CSS will render as expected*
  - *no CSS, and browsers will render as normal text*
  - *non-visual browsers = normal text and no heading*
  - *accessibility issues...*
- search engines, ranking, spiders...
  - *will not process this markup as a heading*
  - *no semantic meaning...*
  - *recorded as normal text*
- CSS styles can be unique
  - *but restricted to `class` usage with `heading`*

HTML - markup for headings - part 2

- many different ways to markup content with HTML, e.g.

```
<p><b>Chapter 1</b></p>
```

- issues still exist with variant markup options, e.g.
  - *visual browsers will render text in bold & same size as default*
  - *unique styling is problematic...*
  - *search engines, ranking, spiders...*
    - will not process this markup as a heading
    - no semantic meaning...
    - recorded as normal text

HTML - markup for headings - part 3

- use markup correctly with structure and meaning, e.g.

```
<h3>Chapter 1</h3>
```

- benefits of this markup, e.g.
  - *conveys meaning to contained text*
  - *visual and non-visual browsers treat heading correctly*
    - regardless of any associated styles...
  - *easy to add unique styles with CSS*
  - *search engines &c. will interpret this markup correctly*
    - extract keywords, semantics, structure...

HTML - markup for tables

- great example of poor usage of HTML markup is `<table>` element
- main issue is use of nested tables and spacer elements, images...
- if used correctly in structured markup
  - *tables can be very useful structure*
  - *impart a sense of semantic organisation to data*
  - *creating various interpretive information*
- what is a table for?
  - *structuring data*
  - *data to impart curated information...*

HTML - markup for tables - example I

- simple table example - columns and rows for *presentation* purposes

```
<p>Travel Destinations</p>
<!-- basic table structure - minimal - rows and columns -->
<table>
  <tr>
    <td><b>Place</b></td>
    <td><b>Country</b></td>
    <td><b>Sights</b></td>
  </tr>
  <tr>
    <td>Nice</td>
    <td>France</td>
    <td>Cours Saleya</td>
  </tr>
  <tr>
    <td>Cannes</td>
    <td>France</td>
    <td>La Croisette</td>
  </tr>
  <tr>
    <td>Antibes</td>
    <td>France</td>
    <td>Picasso museum</td>
  </tr>
</table>
```

example

- example - basic table for presentation

HTML - markup for tables - example 2

- add semantic structure & elements to table caption - replace <p> with correct <caption> usage for a table...

```
<!-- basic table structure - minimal - add a caption -->
<table>
  <caption>Travel Destinations</caption>
  ...
```

- modern browsers style <caption> by default
  - *centred above the table*
- modify styling as required

example

- example - basic table caption

HTML - markup for tables - example 3

- add a summary attribute to the table

```
<!-- basic table structure - minimal - add summary attribute -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  ...
```

- add further meaning and structure to the table
  - *use of a summary attribute on the table element*
- processed by the browsers for semantics
- particularly useful for non-visual browsers

example

- example - basic table with summary

HTML - markup for tables - example 4

- add correct headers <th> to the table

```
<!-- basic table structure - minimal - add table headers -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  <tr>
    <th>Place</th>
    <th>Country</th>
    <th>Sights</th>
  </tr>
  ...
```

Benefits include:

- remove need for presentational markup, bold elements
- visual browsers process structural and presentation qualities of headings
- such heading elements can also be useful for non-visual browsers

example

- example - basic table with headers

HTML - markup for tables - example 5

- table markup and accessibility markup...

```
<!-- basic table structure - accessibility - add ids and headers -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  <tr>
    <th id="place">Place</th>
    <th id="country">Country</th>
    <th id="sights">Sights</th>
  </tr>
  <tr>
    <td headers="place">Nice</td>
    <td headers="country">France</td>
    <td headers="sights">Cours Saleya</td>
  </tr>
  ...
```

- creating a known relationship between the table's header, and its data
- a screen reader, for example, may read this table as follows,
  - *Place: Nice, Country: France, Sights: Cours Saleya*
- established a pattern to the output information for non-visual devices...

example

- example - basic table with accessibility

HTML - markup for tables - example 6

- add extra semantic markup for thead, tfoot, tbody...

```
<!-- basic table structure - add head, foot, body -->
<table summary="structured table data for travel destinations...">
  <caption>Travel Destinations</caption>
  <thead>
    <tr>
      ...
    </tr>
  </thead>
  <tfoot>
    <tr>
      ...
    </tr>
  </tfoot>
  <tbody>
    <tr>
      ...
    </tr>
  </tbody>
</table>
```

- head and foot elements customarily go above the table body
  - *allows modern browsers, readers, &c. to load that data first*
  - *then render the main table content*

Benefits include:

- better underlying structure to data
- greater ease for styling a table due to clear divisions in data and information
- structural and presentational markup now working together correctly...

example

- example - basic table with head, foot, body

HTML - presentational vs structural

- consider *presentational* vs *structural*
  - *e.g.usage of quotations in markup*
  - *similar consideration to headings...*
- need to convey meaning and structure
- rather than a mere presentational instruction
- consider HTML's phrase elements
  - *e.g.* `<cite>`, `<code>`, `<abbr>`
- each phrase element imparts a sense of underlying meaning
  - *structure & then presentation...*

HTML - minimising markup

- noticeable benefit to creating sites with valid markup
  - *separation of structural from presentational*
  - *general reduction in required markup*
- simply conforming to the W3C's specifications
  - *does not inherently guarantee less code for your project*
  - *possible to include many unnecessary elements & retain valid markup*
  - *markup may still be valid*
- project issues may include:
  - *lack of efficiency*
  - *extraneous markup and code*
- to help minimise markup
  - *consider classes added to markup*
    - are there too many? are they all necessary? &c.
    - avoid class usage for unique reference
  - *avoid `<div>` usage for explicit block-level elements*

XML - intro

- XML = eXtensible Markup Language
- markup language similar to HTML
- designed to carry data but not display data
- XML tags are not pre-defined, you must define your own
- designed to be self-descriptive
- XML is a W3C recommendation

XML - simple example

```
<calendar>
  <date>4th November, 1922</date>
  <title>Tomb of Tutankhamun discovered</title>
  <location>Valley of the kings, Luxor, Egypt</location>
  <kv>62</kv>
</calendar>
```

XML - separation of data

- XML allows a designer to separate aspects of data, e.g.
  - *data that needs updating on a regular basis*
  - *from HTML code necessary to display content*
- XML can be stored in separate files
  - *updated as necessary without affecting the HTML*
- XML also helps simplifies data sharing
  - *stored in a plain text format*
  - *platform agnostic*
- easy to exchange XML files and data

XML - migration of data

- XML eases migration to new development platforms, languages systems...
- data can be available to all kinds of *reading machines*
  - *handheld computers, voice machines, news feeds, &c.*
  - *ease of provision for accessible devices and services*
- XML also used as base for creation of other internet technologies, e.g.
  - *XHTML*
  - *RSS*
  - *RDF*
  - *OWL*

XML - structure - intro

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<calendar>
  <date>4th November, 1922</date>
  <title>Tomb of Tutankhamun discovered</title>
  <location>Valley of the kings, Luxor, Egypt</location>
  <kv>62</kv>
</calendar>
```

- basic structure

XML - structure - syntax

- there must be closing tags, e.g.

```
<p>a new paragraph...</p>
```

and not

```
<p>a new paragraph...
```

- tags are case-sensitive
- elements must be properly nested, e.g.

```
<bold><italic>a new phrase...</italic></bold>
```

and not

```
<bold><italic>a new phrase...</bold></italic>
```

- must have a root element
- attribute values must be quoted
- entity references e.g. using a character such as < instead of `<`

XML - structure - entity references

character	reference	meaning
<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

XML - structure - comments

```
<!-- tomb discovered by Howard Carter -->
```

XML - recap

- first version of XML became a W3C Recommendation in 1998
- a useful format for data storage and exchange
  - *config files, data storage, data exchange...*
- XML is a simple, very flexible text format
  - *used for the description of marked-up electronic content*
- XML is classified as extensible because
  - *allows user to define markup elements*
  - *e.g. elements for a document, project, domain...*
- XML is a meta-language
  - *a means of formally describing a language*
  - *e.g. a markup language*
- XML useful for
  - *sharing data*
  - *abstraction of data and presentation*
  - *document reuse*
  - ...
- XML provides a basic syntax

XML - recap - origins

- XML emerged as a way to overcome the shortcomings of its two predecessors
  - *Standard Generalised Markup Language (SGML)*
  - *HyperText Markup Language (HTML)*
- HTML is too limited, while SGML is too complex
- XML is software and hardware independent
  - *light and simple tool for carrying information*
- XML helps science, industries, companies
  - *specify how to store specific data*
  - *store data in a machine compatible form*
  - *allows applications to run on any platform*
  - *easily import and process this data*
  - ...

XML - recap - usage

- XML has some important characteristics
  - *extensible so it does not contain a fixed set of tags*
  - *documents must be well-formed according to a strict set of rules*
  - *may be formally validated using DTDs or XML Schemas*
- HTML documents can contain errors (and often do)
  - *mdern browsers will still render the pages as well as possible*
- XML focuses on the meaning of data, not its presentation
- XML is not a replacement for HTML
- XML is used to transport data
- HTML is used to format and display the data for the Web &c.

XML - syntax

- XML is a formal specification for markup languages
  - *formal language specifications have an associated syntax*
- an XML document consists of the following:
  - *a prolog*
    - includes XML declaration
    - optional reference to external structuring documents
  - *the body*
    - consisting of a number of elements which may also contain attributes
- *prolog* is the declaration
  - *informs the machine that the document &c. is XML*
  - *includes any relevant additional information such as the encoding...*
- other components may be inserted in the *prolog*
  - *associated schemas (either DTDs or XML Schema)*
  - *or attached stylesheets (in CSS or XSL)*
- in the *body*
  - *elements are organized in a hierarchical structure (a tree)*
  - *one root element (Document element)*
  - *all other elements within the root*

XML - example structure

xml

```
<?xml version="1.0" standalone='yes'?>
<library>
  <book category="non-fiction">
    <title>The Discovery of the Tomb of Tutankhamen</title>
    <author>Howard Carter</author>
    <year>1923</year>
  </book>
</library>
```

tree

```
 /library
    |-book
       |---title
       |---author
       |---year
```

XML - structure - elements

```
<library>
  <book category="non-fiction">
    <title>The Discovery of the Tomb of Tutankhamen</title>
    <author>Howard Carter</author>
    <year>1923</year>
  </book>
</library>
```

- XML element
  - *everything from start to end tag*
  - *inclusive of the tags*
- element can contain
  - *other elements*
  - *text*
  - *attributes*
  - *or a mix of all of the above...*
- example listing - a bit of fun

XML - structure - elements - naming rules

There are some naming rules for use with XML elements, e.g.

- elements can contain letters, numbers, and other characters
- elements cannot start with a number or punctuation character
- elements cannot start with the letters xml (or XML, or Xml, etc)
- elements cannot contain spaces

XML - structure - elements - best practices

There are also some best practices for use with XML elements, e.g.

- make element names descriptive and easy to read
- try to avoid hyphenated words
  - *some software may perceive this as a subtraction*
- try to avoid using a . period
  - *some software may perceive this as a defined property...*
  - *e.g. property of an object...*
- avoid colons in the element name
  - *reserved for the namespace in an XML document*

XML - structure - elements - extensible

- XML elements are also extensible
- extend an element and it should not break an application
- XML is extensible
  - *not a fixed format like HTML*
- XML is a metalanguage
  - *a language for describing other languages*
- describe your own languages, e.g.
  - *RDF, SVG...*
  - *list of XML markup languages*

XML - structure - attributes

```
<book type="print">Hannibal's Footsteps</book>
```

- additional information about the element
- attribute values must be quoted
- double or single quotes

```
&quot;
```

XML - structure - attribute usage

- some of the problems with using attributes include,
  - *attributes cannot contain multiple values (elements can)*
  - *attributes cannot contain tree structures (elements can)*
  - *attributes are not easily expandable (for future changes)*
- metadata is a common use of attributes for the element content
  - *e.g. an id or reference number*

XML - structure - attribute vs element

- different discussions amongst various scientific and technical communities
  - *when and why to encode information into attributes or as content in elements*
- there is not a prescribed or specific rule for this choice
- the choice will often depend on the developer or designer
  - *domain influence as well*
  - *company practice and preferences*
  - *parser or tools for transforming XML*
  - ...
- using attributes at all is often a matter of choice and expediency
- e.g. what's the difference between the following?

```
<title><primary>The Discovery of the Tomb of Tutankhamen</primary></title>
```

or

```
<title type="primary">The Discovery of the Tomb of Tutankhamen</title>
```

XML - structure - attribute usage

- XML attributes are normally used to describe elements
  - *provide additional information about elements*
- metadata (data about data) commonly stored as attributes
- data stored as elements
- styles and patterns of usage will also develop with experience
  - *often informed by best practices in a given domain or community*

e.g.

```
<library>
  <book category="non-fiction">
    <title type="primary">The Discovery of the Tomb of Tutankhamen</title>
    <author>Howard Carter</author>
    <year>1923</year>
  </book>
</library>
```

XML - test 1

Think about how you might encode the following information:

- a car

- a sports team

- 2 Musical CDs including each song per album per CD (e.g. each album has 5 songs)

Consider how to represent and encode these examples in XML.

Save the file as a `.xml` file, e.g. `cars.xml`.

XML markup may be created using a standard editor, e.g.

• Visual Studio Code • Atom Editor

or a specific tool such as

• OxygenXML

Demos

HTML

- basic list items
- basic group
- basic group style
- basic menu tabs
- basic table presentation
- basic table caption
- basic table with summary
- basic table with headers
- basic table with accessibility
- basic table - head, foot, body
- semantic usage

XML

- basic XML
- basic structure

References

- MDN - HTML Block-level vs Inline
- MDN - HTML `<b>` element
- MDN - HTML Global Attributes
- MDN - HTML Heading elements
- MDN - HTML `<table>` element
- MDN - HTML `<ul>` element
- Wikipedia - list of XML markup languages