Comp 388/424 - Client-Side Web Design

Fall Semester 2015 - Week 2

Dr Nick Hayward

HTML - Intro

- acronym for *HyperText Markup Language*
- simple way to structure visual components of a website or web application
- HTML also uses keywords, or element tags
    - *follow a rigidly defined syntax*
- helps us to create web pages and web applications
    - *web browsers, such as Chrome or Firefox, may render for viewing*
- an error can stop a web page from rendering
    - *more likely it will simply cause incorrect page rendering*
- interested in understanding the core of web page designing
    - *understand at least the basics of using HTML*

HTML - Element syntax - part 1

Constructed using elements and attributes, which are embedded within an HTML document.

Elements should adhere to the following,

- start with an opening element tag, and close with a matching closing tag
    - *names may use characters in the range **0-9**, **a-z**, **A-Z***
- content is, effectively, everything between opening and closing element tags
- elements can contain empty content
- empty elements should be closed in the opening tag
- most elements permit attributes within the opening tag

HTML - Element syntax - part 2

An element's *start* tag adheres to a structured pattern, which is exactly as follows,

1. a **<** character
2. tag name
3. optional **attributes**, which are separated by a space character
4. optional space characters (one or more...)
5. optional **/** character, indicating a **void** element
6. a **>** character

For example,

```
<div>
```

HTML - Element syntax - part 3

An element's *end* tag also adheres to a pattern, again exactly as defined as following,

1. a **<** character
2. a **/** character
3. element's tag name (ie: name used in matching start tag)
4. optional space characters (one or more...)
5. a **>** character

For example,

```
</div>
```

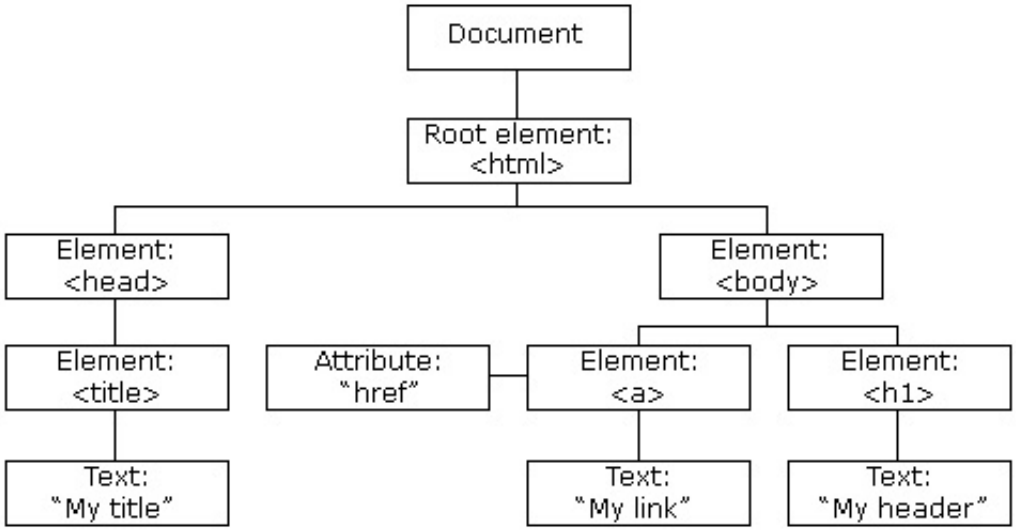
NB: void elements, such as `<br />` or `<img />`, must *not* specify end tags.

HTML - Element syntax - part 4

- HTML, XHTML, can be written to follow the patterns and layouts of XML
- HTML elements can also be nested with a parent, child, sibling...
    - *relationship within the overall tree data structure for the document*
- as the HTML page is loaded by a web browser
    - *the HTML DOM (document object model) is created*
- basically a tree of objects that constitutes the underlying structure
    - *the rendered HTML page*
- DOM gives us an API (application programming interface)
    - *a known way of accessing, manipulating the underlying elements, attributes, and content*
- DOM very useful for JavaScript manipulation

Image - HTML DOM Tree of Objects

W3C DOM Tree



Source - W3C

HTML - Attribute syntax - part 1

- HTML attributes follow the same design pattern as XML
- provide additional information to the parent element
- placed in the opening tag of the element
- follow the standard syntax of name and value pairs
- many different permitted legal attributes in HTML
- four common names that are permitted within most HTML elements
    - `class`, `id`, `style`, `title`

HTML - Attribute syntax - part 2

Four common names permitted within most HTML elements

- class
  - *specifies a classname for an element*
- id
  - *specifies a unique ID for an element*
- style
  - *specifies an inline style for an element*
- title
  - *specifies extra information about an element*
  - *can be displayed as a tooltip by default*

NB:

- cannot use same name for two or more attributes
- regardless of case
- on the same element start tag.

HTML - Attribute syntax - part 3

A few naming rules for attributes

- empty attribute syntax
  - `<input disable>`
- unquoted attribute-value syntax
  - `<input value=yes>`
  - *value followed by /, at least one space character after the value and before /*
- single quoted attribute-value syntax
  - `<input type='checkbox'>`
- double quoted attribute-value syntax
  - `input title="hello">`

NB:

- further specific restrictions may apply for the above
- consult W3 Docs for further details
- above examples taken from W3 Docs - Syntax Attributes Single Quoted

HTML - Doctype - part 1

- `doctype or DOCTYPE` is a special instruction to the web browser
    - *concerning the required processing mode for rendering the document's HTML*
- `doctype` is a required part of the HTML document
- first part of our HTML document
- should always be included at the top of a HTML document, eg:

```
<!DOCTYPE html>
```

- doctype we add for HTML5 rendering
- not a HTML element, simply tells the browser required HTML version for rendering

HTML - Doctype - part 2

- HTML4 needs to specify the required *DTD* (document type definition)
    - *legacy of that version's origins in SGML*
- HTML4 can specify different types of documents
    - *helps the browers render the page correctly, and as expected*
- different types include
    - *strict*
    - *contains all HTML elements and attributes (excluding presentation & deprecated elements such as `font`, & no framesets)*
    - *transitional*
    - *contains all HTML elements and attributes (including presentational & deprecated elements such as `font`, & no framesets)*
    - *frameset*
    - *same as transition DTD, but allows the use of framesets*
    - *XHTML 1.0 strict*
    - *XHTML 1.0 transitional*
    - *XHTML 1.0 frameset*
- more recent XHTML 1.1 DTD also available
    - *follows pattern of XHTML 1.0 strict*
    - *adds support for modules such as Ruby...*

HTML - Doctype - part 3

HTML4 Doctype examples include:

- strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

- transitional

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

- frameset

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

HTML - Doctype - part 4

XHTML Doctype examples include:

- XHTML 1.0 strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- XHTML 1.0 transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

- XHTML 1.0 frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

- XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

HTML - Character encoding - part 1

- element text, and attribute values, must consist of defined **Unicode** characters
  - *The Unicode Consortium*
  - *Unicode Information*
  - *Unicode examples - many, many examples...*
- As with most things, there are some exceptions
  - *must not contain `U+0000` characters*
  - *must not contain permanently undefined Unicode characters*
  - *must not contain control characters other than space characters*
  - *Space `U+0020`*
  - *Tab `U+0009`*
  - *Line feed `U+000A`*
  - *Form feed `U+000C`*
  - *Carriage return `U+000D`*

HTML - Character encoding - part 2

Basically, we use the following definable types of text for content etc.

- normal character data
- this includes standard text and character references
- cannot include non-escaped < characters
- replaceable character data
- includes elements for `title` and `textarea`
- allows text, including non-escaped < characters
- character references
    - *a form of markup for representing single characters*
    - *eg: a dagger* `&dagger;` *or* `&#8224;` *or* `&#x2020;`

XHTML vs HTML - part 1

- XHTML is often described as HTML redesigned as XML
- XHTML enforces correct markup of HTML
    - *follows same patterns of well-formed XML documents*
- primary differences between HTML and XHTML include
    - *XHTML DOCTYPE is* ***mandatory***
    - *`xmlns` attribute in `<html>` element is* ***mandatory***
    - ***mandatory*** *elements in XHTML include*
    - *`<html>`, `<head>`, `<title>`, and `<body>`*

XHTML vs HTML - part 2

Example XHTML 1.0 Strict template

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <head>
      <title>XHTML 1.0 Strict</title>
   </head>
   <body>
   </body>
</html>
```

XHTML vs HTML - part 3

XHTML elements adhere to the following rules,

- proper nesting
    - *elements must not overlap other elements*
    - *breaks the underlying tree DOM for the page. eg:*

```
<!-- incorrect overlapping -->
<div><p>some text...</div></p>
<!-- nesting -->
<div><p>some text...</p></div>
```

- must always be closed
    - *all elements must be closed with a matching closing tag. eg:*

```
<!-- incorrect -->
<p>some text...
<!-- correct -->
<p>some text...</p>
```

- empty elements must also be closed correctly

```
<!-- incorrect -->
<br >
<!-- correct -->
<br />
```

- must be in lowercase
- must have a root element

XHTML vs HTML - part 4

XHTML attributes adhere to the following rules,

- must be lower in case
- must be *quoted*
    - *double quotes is standard for attribute values. eg:*

```
<!-- incorrect -->
<p class=content>
<!-- correct -->
<p class="content">
```

- minimisation is forbidden
    - *must include quoted value. eg:*

```
<!-- incorrect -->
<input checked>
<!-- correct -->
<input checked="true">
```

XHTML vs HTML - part 5

We can also update and convert legacy HTML code using the following options,

- every page needs to include an XHTML doctype declaration, eg:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- we can also add an `xmlns` attribute to the `html` element of every page, eg:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

- update element names to ensure they are all **lowercase**
- ensure all elements are correctly closed
- update all attribute names to lowercase
- ensure all attribute values are correctly quoted

XHTML vs HTML - part 6

We can then double-check our XHTML using the W3C's validator,

- Markup Validation Service

HTML5 - intro

- finally became a standard in October 2014
- introduces many new features to HTML standard
- additional features include, for example
    - *new canvas element for drawing*
    - *video and audio support*
    - *support for local offline storage*
    - *content specific elements*
    - *including article, footer, header, nav, section*
    - *form controls such as*
    - *calendar, date, time, email, url, search*
- new input type attribute values
    - *assigned to provide better input control*

Check browser compatibility using HTML5 Test

HTML5 - Basic template

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>

</body>
</html>
```

HTML5 - Elements - part 1

- often known simply as **tags**
- elements allow us to add a form of metadata to our HTML page
- for example, we might add

```
<!-- a paragraph element -->
<p>add some paragraph content...</p>
<!-- a first heading element -->
<h1>our first heading</h1>
```

- this metadata used to apply structure to a page's content

HTML5 - Elements - part 2

- we can now add additional structure to our basic template

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!-- title for the webpage appears in the window, tab heading... -->
<title>Demo 1</title>
</head>
<body>
<h1>Our first web page</h1>
<p>
As we build our web apps, more elements and content will be added...
</p>
</body>
</html>
```

Demo 1

HTML5 - Comments

- comments are simple and easy to add to HTML
- add to HTML code as follows,

```
<!-- a comment in html -->
```

- comment not visible in the rendered page
- comment appears in the code for reference...

Image - HTML5 Sample Rendering 1

Rendering of demo 1

Source - Demo 1

HTML5 - Semantic elements - part 1

- new semantic elements added for HTML5
- known as **block-level** elements
- include the following elements,

```
<article>
<aside>
<details>
<figure>
<figcaption>
<footer>
<header>
<main>
<nav>
<section>
```

- better structure underlying documents
    - *add clear semantic divisions*

HTML5 - Semantic elements - part 2

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!-- our second demo with lots of new elements -->
<title>Demo 2</title>
</head>
<body>
<header>
<!-- navigation elements, links... -->
<h1>Our first web page</h1>
</header>
<nav>Option 1</nav>
<main>
<section>
<p>
As we build our web apps, more elements and content will be added...
</p>
<figure>
<img src="media/images/philae-demo2.jpg" alt="temple of philae"
width="333px" height="200px">
</figure>
</section>
<aside>
Temple at Philae in Egypt is Ptolemaic era of Egyptian history...
</aside>
</main>
<footer>
foot of the page...
</footer>
</body>
</html>
```

Demo 2

Image - HTML5 Sample Rendering 2

Rendering of demo 2

Our first web page

Option 1

As we build our web apps, more elements and content will be added to this template.

Temple at Philae in Egypt is Ptolemaic era of Egyptian history. Similar temples include Edfu... foot of the page...

Source - Demo 2

HTML5 - Semantic elements - part 3

- element tag `article` not used in previous demo
- `article` and `section` tag can both cause some confusion
- not as widely used as expected
- `div` element still widely seen in development
- HTML5 is supposed to have relegated `div`
    - *sectioning element of last resort...*
- `article` and `section`
    - *good analogy with a standard newspaper*
    - *different sections such as headlines, politics, health...*
    - *each section will also contain articles*
- HTML specification also states that an `article` element

> *represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication.*

HTML5 - Semantic elements and structure

- perceived issue or concern with HTML5 semantic elements
    - *how and when to add them to our document*
    - *where and when do we add them to our page?*
- non-semantic elements often considered simpler to apply
    - *generalised application and context for usage*

HTML5 - Semantic elements and structure - `<header>` and `<nav>`

- `<header>`
  - *used to collect and contain introductory content*
  - *semantically appropriate for the head or top of a page*
  - *technically feasible and acceptable to include multiple `<header>` elements*
  - *eg: `<header>` within main content, sidebar content, an article, a section...*
- `<nav>`
  - *short for navigation*
  - *stores and defines a set of lins for internal or external navigation*
  - *not meant to define all page navigation links*
  - *often considered suitable for primary site links*
  - *additional links can be placed in*
  - *sidebar, footer, main content...*

HTML5 - Semantic elements and structure - `<main>`

- this element tag defines our **main** content
- traditionally the central content area of our page or document
- HTML4 often used a `<div>` element
    - *plus a `class` or `id` to define central content*
    - *eg:*

```
<div id="main">
...
</div>
```

- HTML5 semantically defines and marks content as `<main>`
- `<main>` should not include any page features such as
    - *nav links, headers etc, that are repeated across multiple pages*
- cannot add multiple `<main>` elements to a single page
- must not be structured as a child element to
    - *`<article>`, `<aside>`, `<footer>`, `<header>`, or `<nav>`*

HTML5 - Semantic elements and structure - `<section>`, `<article>`, `<aside>`

- `<section>`
    - *defines a section of a page or document*
    - *W3C defines as follows,*

> *a section is a thematic grouping of content. The theme of each section should be identified, typically by including a heading as a child of the section element.*

Source - W3C Documentation

- a site can be sub-divided into multiple `<section>` groupings
    - *eg: as we might consider a chapter or section break in a book...*
- `<article>`
    - *suitable for organising and containing independent content*
    - *include multiple `<article>` elements within a page*
    - *use to establish logical, individual groups of content*
    - *again, newspaper analogy is useful to remember*
    - *eg: a blog post, story, news report...might be a useful article*
    - *key to using this element is often whether content can be used in isolation*
- `<aside>`
    - *used to define some content aside from containing parent content*
    - *normally used to help define or relate material to surrounding content*
    - *effectively acts as supporting, contextual material*

HTML5 - Semantic elements and structure - <figure>, <figcaption>

- <figure> & <figcaption>
  - *as with print media, we can logically group image and caption*
  - *<figure> acts as parent for image grouping*
  - *child elements include*
  - *<img> and <figcaption>*

```
<figure>
  <img src="media/images/philae-demo2.jpg" alt="temple of philae"
  width="333px" height="200px">
  <figcaption>Ptolemaic temple at Philae, Egypt</figcaption>
</figure>
```

- updated demo with figure grouping - Demo 3

Image - HTML5 Page Structure - Part 1

HTML5 Semantic elements - Part 1

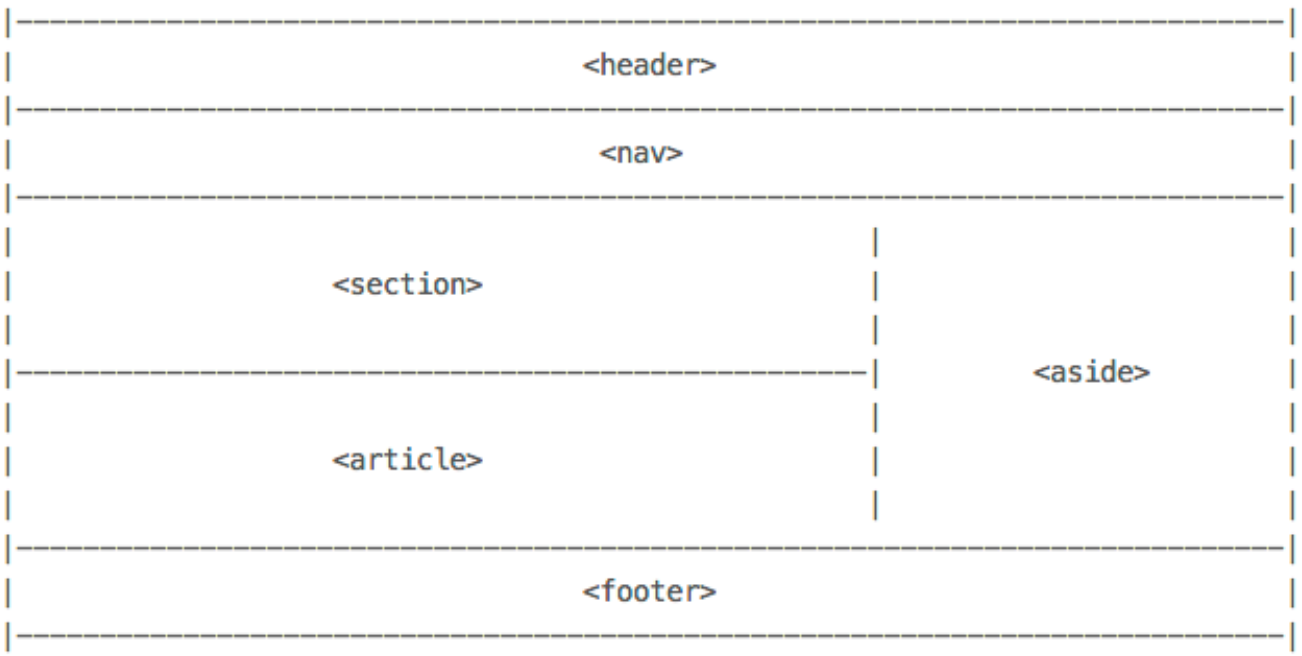
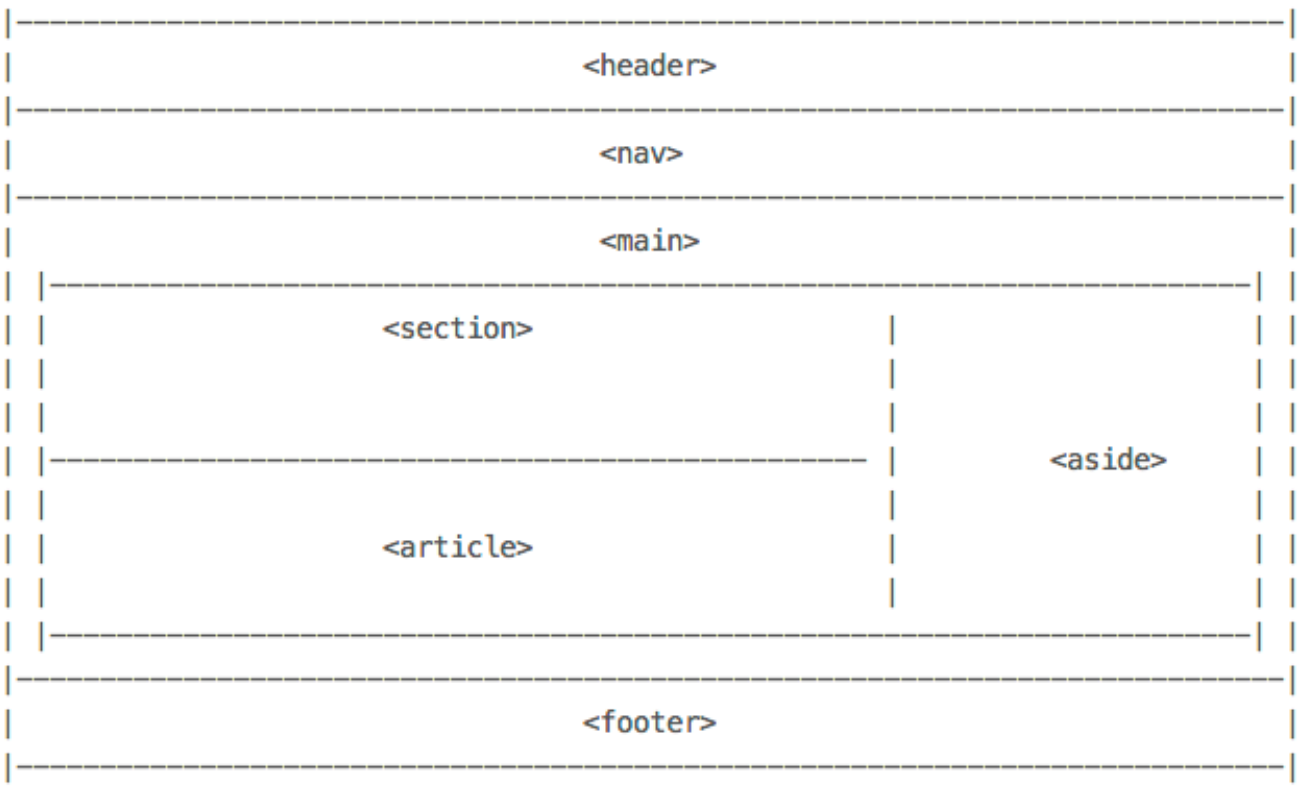


Image - HTML5 Page Structure - Part 2

HTML5 Semantic elements - Part 2



HTML5 - Page structure - Part 3

- not included `<html>` and `<body>` tags in diagrams
  - *required for all HTML documents*
- divided the page into four logical, semantic divisions
  - *header*
  - *nav*
  - *main*
  - *footer*
- we could move `<nav>` into the `<header>` division at the top
  - *not always necessary and not compulsory*
- we could also add a sidebar etc for further division of content

HTML5 - Extra elements - intro

- many other interesting and useful new HTML5 elements
    - *in addition to semantic elements*
- some struggle for browser compatibility
- useful new elements such as
    - *graphics and media*
- HTML5 APIs introduced as well, including
    - *App Cache*
    - *Drag/Drop*
    - *Geolocation*
    - *Local Storage*
    - ...
- again, check browser support and compatibility

Browser check

- Can I Use_____?
    - *eg: Can I Use Drag and Drop?*

HTML5 - Extra elements - media - part 1

`<video>` element

- until HTML5, video playback reliant on plugins
    - *eg: Adobe Flash*
- embed video using element tag `<video>`
- add attributes for
    - *height, width, controls...*
- not all web browsers support all video codecs
- option to specify multiple video sources
- best supported codecs include
    - *MP4 (or H.264), WebM, OGG...*
- good general support for `<video>` element
- check browser support for `<video>` element
    - *Can I use_____video?*

HTML5 - Extra elements - media - part 2

<video> - a quick example might be as follows,

```
<video width="300" height="240" controls>
  <source src="media/video/movie.mp4" type="video/mp4">
  <source src="media/video/movie.webm" type="video/webm">
  Your browser does not support the video tag.
</video>
```

Demo 4 - HTML5 Video playback

HTML5 - Extra elements - media - part 3

`<audio>` element

- HTML5 also supports standardised element for embedded audio
- supported codecs for `<audio>` playback include
    - *MP3 and mp4*
    - *WAV*
    - *OGG Vorbis*
    - *3GP*
    - *m4a*
- again, check browser support and compatibility
    - *Can I use_____audio?*
- fun test of codecs
    - *HTML5 Audio*

HTML5 - Extra elements - media - part 4

<audio> - a quick example might be as follows,

```
<audio controls>
  <source src="media/audio/audio.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
```

Demo 5 - HTML5 Audio playback

HTML5 - Extra elements - graphics - part 1

- graphics elements are particularly fun to use
- use them to create interesting, useful graphics renderings
- in effect, we can draw on the page
- `<canvas>` element acts as a placeholder for graphics
    - *allows us to draw with JavaScript*
- draw lines, circles, text, add gradients...
    - *eg: draw a rectangle on the canvas*

HTML5 - Extra elements - graphics - part 2

<canvas> will be created as follows,

```
<canvas id="canvas1" width="200" height="100">
  Your browser does not support the canvas element.
</canvas>
```

then use JavaScript to add a drawing to the canvas

```
<script type="text/javascript">
var can1 = document.getElementById("canvas1");
var context1 = can1.getContext("2d");
context1.fillStyle="#000000";
context1.fillRect(0,0,150,75);
</script>
```

Result is a rendered black rectangle on our web page.

Demo 6 - HTML5 Canvas - Rectangle

HTML5 - Extra elements - graphics - part 3

A cube can be created as follows,

```
<script type="text/javascript">
function draw() {
/*black cube*/
var can1 = document.getElementById("canvas1");
var context1 = can1.getContext("2d");
context1.fillStyle="#000000";
context1.fillRect(0,0,50,50);
}
</script>
```

Again, we end up with the following rendered shape on our canvas.

Demo 7 - HTML5 Canvas - Cube

- modify drawing for many different shapes and patterns
    - *simple lines, circles, gradients, images...*

1. shows different rendered shapes on a canvas.
    - Demo 8 - HTML5 Canvas - Assorted Shapes
2. little retro games
    - Demo 9 - HTML5 Canvas - Retro Breakout Game

HTML Basics - <body> - part 1

- to define the main body of the web page we use the <body> element
- headings can be created using variants of
    - *<h1>, <h2>.....<h6>*
- we can now add some simple text in a <p> element

```
<p>...</p>
```

- add a line break using the <br /> element
- <hr /> element adds a horizontal line

- comments can also be added through our HTML

```
<!-- comment... -->
```

HTML Basics - <body> - part 2

Linking in HTML

- linking is an inevitable part of web design and HTML usage
- can be considered within three different contexts
    - *linking to an external site*
    - *linking to another page within the same site*
    - *linking different parts of the same page*
- add links to text and images within the HTML
- <a> element for links plus required attributes
    - `<a href="http://www.google.com/">Google</a>` *or*
    - `<a href="mailto:name@email.com">Email</a>`
    - `<a href="/another_page.html">another page</a>`
    - `<a name="anchor">Internal Anchor</a>` *or*
    - `<a id="anchor">Anchor</a>`
    - `<a href="#anchor">Visit Internal Anchor</a>` *or*
    - `<a href="/another_page.html#anchor">Visit External Anchor</a>`

Demo 10 - HTML - Internal Anchor

HTML Basics - <body> - part 3

Linking in HTML - continued

- standard attributes supported by <a> element include
    - *class, id, lang, style, title...*
- optional attributes are available for <a> element including
    - *target, href, name...*
- target attribute specifies where the link will be opened relative to the current browser window
- possible attribute values include

```
_blank
_self
_parent
_top
```

HTML Basics - <body> - part 4

Working with images

- <img> allows us to embed an image within a web page
- <img> element requires a minimum *src* attribute

```
<img src="image.jpg" />
```

- other optional attributes include
    - *class, id, alt, title, width, height...*
- use images as links
- image maps

```
<map name="textmap">
  <area shape="rect" coords="..." alt="Quote 1" href="notes1.html" />
</map>
```

HTML Basics - <body> - part 5

Adding a table

- organise data within a table starting with the `<table>` element
- three primary child elements include
    - *<tr>, <th>, <td>*

```
<table>
<tr>
<th>header 1</th>
</tr>
<tr>
<td>row 1, cell 1</td>
</tr>
</table>
```

- also add a `<caption>`
- span multiple columns using the `colspan` attribute
- span multiple rows using the `rowspan` attribute

HTML Basics - <body> - part 6

Organising a list

- unordered list <ul>, ordered list <ol>, definition list <dl>
- and <ol> contains list items

```
<ul>
<li>...</li>
</ul>
```

```
<ol>
<li></li>
</ol>
```

- definition list uses <dt> for the item, and <dd> for the definition

```
<dl>
<dt>Game 1</dt>
<dd>our definition</dd>
</dl>
```

HTML Basics - <body> - part 7

Using forms

- used to capture data input by a user, which can then be processed by the server
- <form> element acts as the parent wrapper for a form
- <input> element for user input includes options using the *type* attribute
    - *text, password, radio, checkbox, submit*

```
<form>
Text field: <input type="text" name="textfield" />
</form>
```

- process forms using
    - *eg: PHP, JavaScript...*

Demos

- Demo 1 - Our first web page
- Demo 2 - New elements added
- Demo 3 - Semantic structuring
- Demo 4 - HTML5 Video playback
- Demo 5 - HTML5 Audio playback
- Demo 6 - HTML5 Canvas - Rectangle
- Demo 7 - HTML5 Canvas - Cube
- Demo 8 - HTML5 Canvas - Assorted Shapes
- Demo 9 - HTML5 Canvas - Retro Breakout Game
- Demo 10 - HTML - Internal Anchor

References

- HTML5 Audio formats
- HTML5 Test
- Perishable Press - Barebones Web Templates
- The Unicode Consortium
- Unicode Information
- Unicode examples
- W3 Docs for further details
- W3C HTML5 Documentation
- W3Schools - HTML5 Semantic Elements
- W3Schools - DOM Image