Comp 324/424 - Client-side Web Design - Slides

Spring Semester 2018 - Week 2

Dr Nick Hayward

course details

Lecturer

- Name: Dr Nick Hayward
- Office: Doyle 307 (LSC)
- Office hours
  - *Monday afternoon by appointment (WTC)*
- Faculty Page

Important dates for this semester

- Martin Luther King, Jr. holiday - 15th January 2018
  - ***n.b.*** *no formal class: 15th January 2018*
- DEV week: 5th to 12th March 2018
  - ***n.b.*** *no formal class: 5th March 2018*
  - *presentation & demo: 12th March 2018 @ 4.15pm*
- Spring Break: 5th to 9th March 2018
- Easter holiday: 29th March to 2nd April 2018
  - ***n.b.*** *no formal class: 2nd April 2018*
- Final class: 23rd April 2018
  - *presentation & demo: 23rd April 2018 @ 4.15pm*
- Exam week: 30th April to 5th May 2018
  - *Final assessment due on 30th April 2018 by 4.15pm*

Presentation, reports &c.

- DEV week demo
  - *due Monday 12th March 2018 @ 4.15pm*
- final team demo
  - *due Monday 23rd April 2018 @ 4.15pm*
- final team report
  - *due Monday 30th April 2018 @ 4.15pm*

Initial Course Plan - Part I

(up to ~ DEV Week)

- Build and publish a web app from scratch
  - *general setup and getting started*
  - *maintenance and publication*
  - *basic development and manipulation (HTML, CSS, JS...)*
  - *add some fun with Ajax, JSON, server-side...*
  - *useful data storage techniques and options*
  - *testing...*

Initial Course Plan - Part 2

(Up to the end of the semester)

- Augment and develop initial app
- Explore other options
  - *further libraries and options*
  - *tools and workflows*
  - *visualisations, graphics...*
  - *publish (again...)*
- data options
  - *self hosted (MongoDB, Redis...)*
  - *APIs*
  - *cloud services, storage (Firebase, Heroku, mLab...)*
- React...

Assignments and Coursework

Course will include

- weekly bibliography and reading (where applicable)
- weekly notes, examples, extras...

Coursework will include

- quizzes or group exercises at the end of each section (Total = 30%)
  - *based on course notes, reading, and examples*
- development and project assessment (Total = 70%)
  - *mid-semester assessment (Total = 30%)*
    - end of DEV week
    - demo due Monday 12th March 2018 @ 4.15pm
  - *final assessment (Total = 40%)*
    - demo due Monday 23rd April 2018 @ 4.15pm
    - report due Monday 30th April 2018 @ 4.15pm

Quizzes, group exercises...

Course total = 30%

- at least one week notice before quiz
  - *average time ~40 minutes (can be extended...)*
  - *taken towards the end of class*
- group exercises
  - *help develop course project*
  - *test course knowledge at each stage*
  - *get feedback on project work*

Development and Project Assessment

Course total = 70% (Parts 1 and 2 combined total)

Initial overview

- combination project work
  - *part 1 = mid-semester **DEV Week** work (30%)*
  - *part 2 = final demo and report (40%)*
- group project (max. 5 persons per group)
- design and develop a web app
  - *purpose, scope &c. is group's choice*
    - **NO** blogs, to-do lists, note-taking...
    - chosen topic requires approval
    - **NO** content management systems (CMSs) such as Drupal, Joomla, WordPress...
    - **NO** PHP, Python, Ruby, C# & .Net, Go, XML...
    - **NO** CSS frameworks, such as Bootstrap, Foundation, Materialize...
  - *must implement data from either*
    - self hosted (MongoDB, Redis...)
    - APIs
    - cloud services, storage (Firebase, Heroku, mLab &c.)
    - **NO** SQL...

DEV Week Assessment

- web app developed from scratch
  - *examples, technology &c. outlined during first part of semester*
  - *e.g. HTML5, CSS, JS...*
  - ***NO** content management systems (CMSs) such as Drupal, Joomla, WordPress...*
  - ***NO** PHP, Python, Ruby, C# & .Net, Go, XML...*
  - ***NO** CSS frameworks, such as Bootstrap, Foundation, Materialize...*
- demo and project report
  - *due on Monday 12th March 2018 @ 4.15pm*
- anonymous peer review
  - *similar to user comments and feedback*
  - *chance to respond to feedback before final project*

Final Assessment

- working final app
  - ***NO*** *content management systems (CMSs) such as Drupal, Joomla, WordPress...*
  - ***NO*** *PHP, Python, Ruby, C# & .Net, Go, XML...*
  - ***NO*** *CSS frameworks, such as Bootstrap, Foundation, Materialize...*
- presentation and demo - live working app...
  - *due on Monday 23rd April 2018 @ 4.15pm*
  - *show and explain implemented differences from DEV week project*
  - *where and why did you update the app?*
  - *benefits of updates?*
- how did you respond to peer review?
- final report
  - *due on Monday 30th April 2018 @ 4.15pm*

Goals of the course

A guide to developing and publishing interactive client-side web applications and publications.

Course will provide

- guide to developing client-side web applications from scratch
- guide to publishing web apps for public interaction and usage
- best practices and guidelines for development
- fundamentals of web application development
- intro to advanced options for client-side development
- ...

Course Resources - part 1

Website

Course website is available at https://csteach424.github.io

- timetable
- course overview
- course blog
- weekly assignments & coursework
- bibliography
- links & resources
- notes & material

No Sakai

Course Resources - part 2

GitHub

- course repositories available at https://github.com/csteach424
  - *weekly notes*
  - *examples*
  - *source code (where applicable)*

Trello group

- group for weekly assignments, DEV week posts, &c.
- Trello group - COMP 424
  - *https://trello.com/csteach424*

Slack group

- group for class communication, weekly discussions, questions, &c.
- Slack group - COMP 424
  - *https://csteach424.slack.com*

Group projects

- add project details to course's Trello group, *COMP 424 - Spring 2018 @ LUC*
  - *Week 1 - Project Details*
  - *https://trello.com/b/3Dh6pCTu/week-1-project-details*
- create channels on Slack for group communication
- start working on an idea for your project
- plan weekly development up to and including DEV Week
  - *5th to 12th March 2018*
  - *DEV week demo on 12th March 2018*

Intro to Client-side web design

- allows us to design and develop online resources and publications for users
  - *both static and interactive*
- restrict publication to content
  - *text, images, video, audio...*
- develop and publish interactive resources and applications
- *client-side scripting* allows us to offer
  - *interactive content within our webpages and web apps*
- interaction is enabled via code that is downloaded and compiled, in effect, by the browser
- such interaction might include
  - *a simple mouse rollover or similar touch event*
  - *user moving mouse over a menu*
  - *simple but effective way of interacting*

Client-side and server-side - Part 1

Client-side

- scripts and processes are run on the user's machine, normally via a browser
  - *source code and app is transferred to the user's machine for processing*
- code is run directly in the browser
- predominant languages include HTML, CSS, and JavaScript (JS)
  - *HTML = HyperText Markup Language*
  - *CSS = Cascading Style Sheets*
  - *many compilers and transpilers now available to ease this development*
    - e.g. *Go* to *JavaScript*...
- reacts to user input
- code is often visible to the user (source can be read in developer mode etc...)
- in general, cannot store data beyond a page refresh
  - *HTML5 and local web APIs are changing this...*
- in general, cannot read files directly from a server
  - *HTTP requests required*
- single page apps create rendered page for the user

Client-side and server-side - Part 2

Server-side

- code is run on a server
  - *languages such as PHP, Ruby, Python, Java, C#...*
  - *in effect, any code that can run and respond to HTTP requests can also run a server*
- enables storage of persistent data
  - *data such as user accounts, preferences...*
- code is not directly visible to the user
- responds to HTTP requests for a given URL
- can render the view for the user on the server side

and so on...

Getting started

- basic building blocks include HTML, CSS, and JS
- many tools available to work with these technologies
- three primary tools help with this type of development
- web browser
  - *such as Chrome, Edge (IE?), Firefox, Opera, Safari...*
- editor
  - *such as Atom, Sublime, Microsoft's Visual Studio Code...*
- version control
  - *Git, (Mercurial, Subversion)*
  - *GitHub, Bitbucket...*

Getting started - Web Browsers

- choose your favourite
  - *Chrome, Firefox, Safari, Edge...*
  - *not IE*
- developer specific tools
  - *Chrome etc view source, developer tools, JS console*
  - *Firefox also includes excellent developer tools*
  - *Firebug*
- cross-browser extension for web developers
  - *Web Developer*

Getting started - Editors

Many different choices including

Linux, OS X, and Windows

- Atom
- Sublime
- Visual Studio Code

OS X specific

- BBEdit
  - *TextWrangler*

and so on.

Video - Atom 1.0

Source - YouTube - Introducing Atom 1.0

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 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 - structure of HTML

- basic HTML tag defines the entire HTML document

```
<html>
  ...
</html>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</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 may contain empty or *void* 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 may be 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,

```
<!-- opening element tag -->
<div>
<!-- void element -->
<br />
```

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 (i.e. name used in matching start tag)
4. optional space characters (one or more...)
5. a **>** character

For example,

```
<!-- element's matching end tag -->
</div>
```

NB: void elements, such as `<br />` or `<img />`, do *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

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 `/`*
  - *i.e. usage with a void element...*
- 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 - HTML5

- `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, e.g.

```
<!DOCTYPE html>
```

or

```
<!doctype html>
```

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

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
- for example, attribute values must not contain `U+0000` characters

```
U+0000 (NULL)
U+0022 (QUOTATION MARK, ")
U+0027 (APOSTROPHE, ')
U+003E (GREATER THAN, >)
U+002F (FORWARD SLASH, /)
U+003D (EQUALS, =)
```

- e.g W3C recommendations - 8.1.2.3
  - *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`

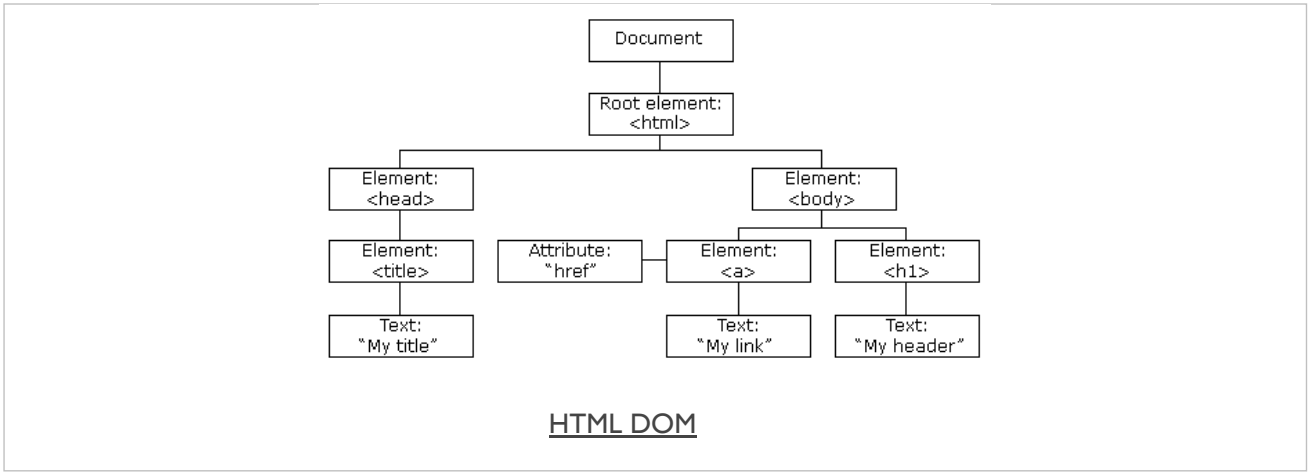
HTML - Character encoding - part 2

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
    - e.g. a dagger represented as `&dagger;` or `&#8224;` or `&#x2020;`
    - e.g. copyright symbol as `&#169`
    - lots of examples, W3 - Character Ref.

DOM Basics - intro

A brief introduction to the document object model (DOM)



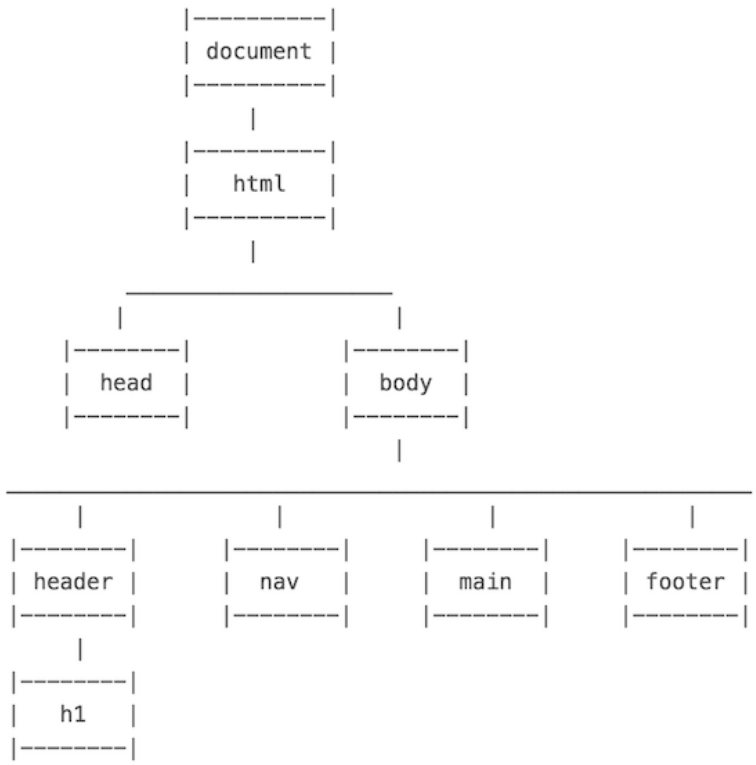
HTML DOM

- Source - W3Schools - JS HTML DOM

DOM Basics - what is DOM?

- **DOM** is a platform and language independent way
  - *to access and manipulate underlying structure of HTML document*
- structured as a representation of a tree data structure
  - *its manipulation follows this same, standard principle*
- DOM tree is constructed using a set of nodes
  - *tree is designed as a hierarchical representation of the underlying document*
- each node on our tree is an element within our HTML document
- inherent hierarchical order originates with the **root** element
  - ***root** sits at the top of our **tree***
  - *descends down following lineage from node to node*
- each node is a child to its parent
  - *we can find many siblings per node as well*
- root at the top of the tree...

Image - HTML DOM



HTML DOM

DOM Basics - useful elements

element tag	usage & description
`<html>`	container element for a HTML document
`<head>`	contains metadata and document information
`<body>`	contains main content rendered as the HTML document
`<header>`	page header...
`<nav>`	navigation, stores and defines a set of links for internal or external navigation
`<main>`	defined primary content area of document
`<footer>`	page footer...
`<section>`	a section of a page or document
`<article>`	suitable for organising and containing independent content
`<aside>`	defines content aside from the content which contains this element
`<figure>`	logical grouping of image and caption
`<img>`	image - can be local or remote using url in `src` attribute
`<figcaption>`	image caption
`<h1>, <h2>...`	headings from 1 to 6 (1 = largest)
`<a>`	anchor - link to another anchor, document, site...
`<p>`	paragraph
`<ul>, <ol>, <dl>`	unordered, ordered, definition lists
`<li>`	list item, used with `<ul>, <ol>...`
`<dt>`	definition term, used with `<dl>`
`<dd>`	definition description, used with `<dl>`
`<table>`	standard table with rows, columns...
`<tr> >`	table row, used with `<table>`
`<th>`	table heading, used with `<table>` and child to `<tr>`
`<td>`	table cell, used with `<table>` and child to `<tr>`
`<div>`	non-semantic container for content, similar concept to `<section>`
`<span>`	group inline elements in a HTML document
`<canvas>`	HTML5 element for drawing on the HTML page
`<video>`	HTML5 element for embedding video playback
`<audio>`	HTML5 element for embedding audio playback

NB: *`<div>` and `<span>` can be used as identifiers when there is no other suitable element to define parts of a HTML5 document. e.g. if there is no defined or significant semantic meaning...*

DOM Basics - sample

```
<!DOCTYPE html>
<html>
  <head>
    <base href="media/images/">
    <meta charset="UTF-8">
    <!-- week 3 - demo 1 -->
    <title>Week 3 - Demo 1</title>
  </head>
  <body>
    <header>
      <h1>Ancient Egypt</h1>
    </header>
    <nav>...</nav>
    <main>
      <section>
        <p>
          Welcome to the Ancient Egypt information site.
        </p>
        <figure>
          <img src="philae-demo2.jpg" alt="philae temple" width="333px"
          height="200px">
          <figcaption>Ptolemaic temple at Philae, Egypt</figcaption>
        </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 - DOM Basics - Sample

DOM Basics - `index.html` page

`index.html` usage and structure

- basic `index.html` page for loading web apps
- app will start with the `index.html` document
  - *html pages saved as `.html` or `.htm`*
  - *`.html` more common...*
- `index.html` acts as a kickstart
  - *for loading and rendering the app*
  - *loads other app resources - CSS, JS...*
- consistent elements in the HTML DOM
  - *`<html>`, `<head>`, and `<body>`*
- HTML5 apps will add
  - *`<header>`, `<main>`, and `<footer>` (when required)*
  - *many other elements for building the app...*

HTML Basics - metadata & <head> element - part I

- part of a HTML document's metadata
- allows us to set metadata for a HTML page
- customised just for that page or replicated as a site-wide implementation
- we can add numerous additional elements to <head>
- add similar links and code for JavaScript
  - *use the <script> element & attributes such as type and src*
  - *HTML4 requires type and src*
  - *HTML5 requires src*

```
<!-- HTML4 and XHTML -->
<script type="text/javascript" src="script.js"></script>
<!-- HTML5 -->
<script src="script.js"></script>
```

HTML Basics - metadata & <head> element - part 2

- add a <title> element with text added as the element content
  - *shown in the browser tab or window heading*

```
<title>Our Page Title</title>
```

- set a default base address for all relative URLs in links within our HTML

```
<base href="/media/images/" target="_blank">
```

- links now simply use the base URL or override with full URL

```
<img src="image.jpg">
<a href="http://www.flickr.com">Flickr</a>
```

- <meta /> adds metadata about the HTML document

```
<meta name="description" content="The Glass Bead Game" />
<meta name="keywords" content="novel, fiction, herman hesse, electronic edition" />
```

HTML - <head> element example

```
<head>
  <meta charset="utf-8">

  <title>Sample...</title>
  <meta name="description" content="sample metadata">
  <meta name="author" content="COMP424">

  <link href="style.css" rel="stylesheet">
  <script src="script.js"></script>

</head>
```

HTML Basics - <body> - part 1

intro

- 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
  - *
 for strict XHTML void*
- <hr> element adds a horizontal line
  - *<hr /> for strict XHTML void*
  - *implies rendering division*
  - *instead of defined structural divide...*
- comments can also be added through our HTML

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

HTML Basics - <body> - part 2

linking

- 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, e.g.

```
<!-- external link -->
<a href="http://www.google.com/">Google</a>
<!-- email link -->
<a href="mailto:name@email.com">Email</a>
<!-- internal page link -->
<a href="another_page.html">another page</a>
<!-- define internal anchor - using name attribute -->
<a name="anchor">Internal anchor</a>
<!-- define internal anchor - using ID attribute -->
<a id="anchor">Anchor</a>
<!-- internal anchor link -->
<a href="#anchor">Visit internal anchor</a>
<!-- internal anchor link on another page -->
<a href="/another_page.html#anchor">Visit internal anchor</a>
<!-- internal anchor link on a page on an external site -->
<a href="https://www.test.com/test.html#anchor">Visit internal anchor on external site</a>
```

- Demo - HTML - Internal Anchor

HTML Basics - <body> - part 3

linking - cont'd

- 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

```
<!-- open link in new window or tab -->
_blank
<!-- same frame -->
_self
<!-- open within parent frameset -->
_parent
<!-- open in the same window -->
_top
```

HTML Basics - <body> - part 4

images

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

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

tables

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

```
<table>
  <caption>424 - basic test table</caption>
  <tr>
    <th>heading 1</th>
    <th>heading 2</th>
  </tr>
  <tr>
    <td>row 1, cell 1</td>
    <td>row 2, cell 2</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

lists

- 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

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
  - *e.g. JavaScript...*

HTML - better markup

- web standards are crucial for understanding markup
  - *markup that goes beyond mere presentation*
- improved usage and structure, accessibility, integration...
- with standards, maintenance and extensibility becomes easier
- improved page structure and styling
  - *helps web designers and developers update and augment our code*
- poor markup usage
  - *to achieve a consideration and rendering of pure design*
  - *e.g. nesting tables many levels deep*
  - *adding images and padding blocks for positioning...*
- support for web standards continues to grow in popular browsers
- gives developers option to combine markup and styling
  - *HTML with CSS to achieve greater standards-compliant design*

HTML - markup and standards

- many benefits of understanding and using web standards, e.g.
- *reduced markup*
  - *less code, faster page loading*
  - *less code, greater server capacity, less bandwidth requirements...*
- *separation of concerns*
  - *content, structure, and presentation separated as needed*
  - *CSS used to manage site's design and rendering*
  - *quick and easy to update efficiently*
- *accessibility improvements*
  - *web standards increase no. of supported browsers & technologies...*
- *ongoing compatibility*
  - *web standards help improve chances of compatibility in the future...*

HTML - better structure

- consider *semantic* or *structured* markup
  - *within the context of app usage and domain requirements*
- trying to impart a sense of underlying meaning with markup
  - *correct elements for document markup*
- for a list
  - *use correct list group with list items - e.g.* `ul`, `li`...
- for a table
  - *consider table for data purposes*
  - *structure table & then consider presentation...*
- *semantic* markup helps create *separation of concerns*
  - *separate content and presentation*
  - *improves comprehension and usage*