

Comp 324/424 - Client-side Web Design

Spring Semester 2024 Week 2

Dr Nick Hayward

Course details

Lecturer

- Name: Dr Nick Hayward
 - Office hours
 - Monday afternoon by appointment @ WTC
 - [Faculty Page](#)
-

Course schedule

Important dates for this semester

- NO class Monday 15th January 2024
 - Week 5 - Project outline & mockup presentation
 - due Monday 12th February 2024 @ 4.15pm
 - Spring break
 - NO class Monday 4th March 2024
 - Week 10 - DEV Week demo & presentation
 - due Monday 18th March 2024 @ 4.15pm
 - Week 15 - Final project demo & presentation
 - due Monday 22nd April 2024 @ 4.15pm
 - Exam week: 29th April to 4th May 2024
 - Final project report
 - due Monday 29th April 2024 @ 4.15pm
-

Coursework schedule

Presentations, reports &c.

- Week 5 - Project outline & mockup
 - due Monday 12th February 2024 @ 4.15pm
 - Week 10 - Dev Week demo & presentation
 - due Monday 18th March 2024 @ 4.15pm
 - Week 15 - Final project demo & presentation
 - due Monday 22nd April 2024 @ 4.15pm
 - Final project report
 - due Monday 29th April 2024 @ 4.15pm
-

Initial course plan - part 1

- 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...
 - initial testing...
-

Initial course plan - part 2

- 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, Mongo...)
 - Project management, build tools &c.
-

Assignments and coursework

Course will include

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

Coursework will include

- discussions
 - class and weekly discussion topics
 - various exercises, code reviews &c.
 - Project outline & mockup
 - due Monday 12th February 2024 @ 4.15pm
 - Dev Week demo & presentation
 - due Monday 18th March 2024 @ 4.15pm
 - end of semester final assessment
 - final presentation and demo due Monday 22nd April 2024 @ 4.15pm
 - final report due Monday 29th April 2024 @ 4.15pm
-

Credits available during course

- course participation = 30
- discussions &c. = 5 per discussion
 - ~ 6 discussions during semester
- course project
 - project outline & mockups = 15
 - Dev week = 25

- final demo & report = 50
 - ~ 150 credits total
-

Participation

Course total = 30

- in-class participation & attendance
 - participation in class discussions
 - participation in group projects
 - peer review of demos
 - ...
-

Discussions & exercises

Course total = 5 credits per topic

- discussions
 - sample websites, games, services...
 - design topics, UI and UX concepts
 - topics posted to Sakai Forum
 - ~ 5/6 discussion topics during semester
 - 5 credits per discussion topic
 - exercises
 - test course knowledge at each stage
 - help develop course project
 - extras
 - peer review of project demos
 - code, application reviews
 - ...
-

Project assessment

Initial overview

- combination project work
 - part 1 = Project outline & mockup - 15 credits
 - part 2 = Dev week demo - 25 credits
 - part 3 = Final demo and report - 50 credits
- group project (max. 4 persons per group)
- design and develop a web app
 - purpose, scope &c. is group's choice
 - * **NO** blogs, to-do lists, note-taking...
 - * **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, Mongo &c.)
 - * **NO** SQL...e.g. MySQL, PostgreSQL &c.

Project outline & mockup assessment

Course total = 15 credits

- begin development and design of a web application
 - built from scratch
 - * HTML5, CSS...
 - builds upon examples, technology outlined during first part of semester
 - purpose, scope &c. is group's choice
 - **NO** blogs, to-do lists, note-taking...
 - presentation should include initial designs, mockups, and any current HTML5 and CSS

Project outline & mockup assessment

Assessment will include the following:

- brief presentation or demonstration of current project work
 - ~ 10 minutes per group
 - analysis of work conducted so far
 - presentation and demonstration...
 - * outline initial project idea and concept
 - * outline current state of web app concept and design
 - * show mockups and designs
 - * ...
 - due Monday 12th February 2024 @ 4.15pm

Dev week demo & assessment

Course total = 25 credits

- continue development of a web application
 - built from scratch
 - HTML5, CSS, plain JavaScript...
- continue design and development of initial project outline and design
- working app (as close as possible...)
 - **NO** content management systems (CMSs) such as Drupal, Joomla, WordPress...
 - **NO** PHP, Python, Ruby, C# & .Net, Java, Go, XML...
 - **NO** CSS frameworks, such as Bootstrap, Foundation, Materialize...
 - **NO** CSS preprocessors such as Sass...
 - **NO** template tools such as Handlebars.js &c.
- data may be implemented from either
 - self hosted (MongoDB, Redis...)
 - APIs
 - cloud services (Firebase...)
 - **NO** SQL...e.g. (you may **NOT** use MySQL, PostgreSQL &c.)
- outline research conducted
- describe data chosen for application
- show any prototypes, patterns, and designs

Dev week demo & assessment

DEV week assessment will include the following:

- brief presentation or demonstration of current project work
 - ~ 10 minutes per group
 - analysis of work conducted so far
 - * e.g. during semester & DEV week
 - presentation and demonstration
 - * outline current state of web app
 - * explain what works & does not work
 - * show implemented designs since project outline & mockup
 - * show latest designs and updates
 - due Monday 18th March 2024 @ 4.15pm
-

Final project assessment

Course total = 50 credits

- continue to develop your app concept and prototypes
 - working app
 - **NO** content management systems (CMSs) such as Drupal, Joomla, WordPress...
 - **NO** PHP, Python, Ruby, C# & .Net, Java, Go, XML...
 - **NO** CSS frameworks, such as Bootstrap, Foundation, Materialize...
 - **NO** CSS preprocessors such as Sass...
 - **NO** template tools such as Handlebars.js &c.
 - must implement data from either
 - * self hosted (MongoDB, Redis...)
 - * APIs
 - * cloud services (Firebase &c...)
 - * **NO** SQL...e.g. (you may **NOT** use MySQL, PostgreSQL &c.)
 - explain design decisions
 - describe patterns used in design of UI and interaction
 - layout choices...
 - what else did you consider, and then omit? (again, why?)
 - show and explain implemented differences from DEV week
 - where and why did you update the app?
 - perceived benefits of the updates?
 - how did you respond to peer review?
-

Final project assessment

Assessment will include the following:

- final presentation & demonstration of project work
 - ~ 10 minutes per group
 - analysis of work conducted during semester
 - presentation and demonstration
 - * outline state of web app concept and design
 - * show final working version of web app
 - explain designs, patterns &c.
 - explain what does and does not work in the final app
 - any other pertinent information on project design & development

- due Monday 22nd April 2024 @ 4.15pm
 - final project report
 - written summary of project design, development, and research
 - no word/page limit...
 - suggested report outline will be provided
 - due Monday 29th April 2024 @ 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://csteach324-424.gitlab.io>

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

Course resources - part 2

GitLab

- course repositories available at <https://gitlab.com/csteach324-424>
 - weekly notes
 - examples
 - source code (where applicable)
-

Citation and attribution of work

- AI and associated generative tools may be used for project work
 - **n.b.** citation must be provided for any submitted material from such sources
 - e.g. code, documentation &c. generated by ChatGPT or other AI based services
 - code submitted from other derived sources must also include an appropriate citation
 - e.g. from articles, websites, other projects, open source projects and repositories &c.
 - ...
-

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, Firefox, Opera, Safari...
 - editor
 - such as [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
 - cross-browser extension for web developers
 - [Web Developer](#)
-

Getting started - Editors

Many different choices including

Linux, OS X, and Windows

- [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 `
` or ``, 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
-

Example - DOM structure & JavaScript

- [traverse DOM tree with JavaScript generator](#)
-

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](#)
-

Example - HTML - custom attributes - part 1

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>JS tests - DOM creation - Attributes</title>
  </head>
  <body>
    <header>
      <h3>JS tests - DOM dynamic creation - Attribute Access</h3>
    </header>
    <section id="content">
```

```
<p>
  <blockquote id="berryhead" data-visible="true">
    Shine through the gloom, and point me to the skies
  </blockquote>
</p>
</section>
<script type="module" src="./attributes.js"></script>
</body>
</html>
```

Example - HTML - custom attributes - part 2

```
/*
 * attributes.js
 * - basic access for custom attributes
 */

// get example blockquote nodes
let quotes = document.getElementsByTagName('blockquote');

// loop through quotes - freeze quotes object using Array.from to create array
for (let quote of Array.from(quotes)) {
  if (quote.getAttribute('data-visible')) {
    quote.setAttribute('data-visible', 'false');
  }
}
```

- [example - Basic Attribute](#)

Example - HTML - custom attributes - part 3

```
/*
 * attributes.js
 * - basic access for custom attributes
 * - add event listener for mouse click
 */

// get example blockquote nodes
let quote = document.getElementById('berryhead');

// add event listener to quotes object
quote.addEventListener('click', () => {
  if (quote.getAttribute('data-visible') === 'true') {
    quote.setAttribute('data-visible', 'false');
    quote.style.color = '#779eab';
  } else {
    quote.setAttribute('data-visible', 'true');
    quote.style.color = '#000';
  }
});
```

- example - [Basic Attribute 2](#)
- [MDN - Using Dynamic Styling Information](#)

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

DOM Basics - intro

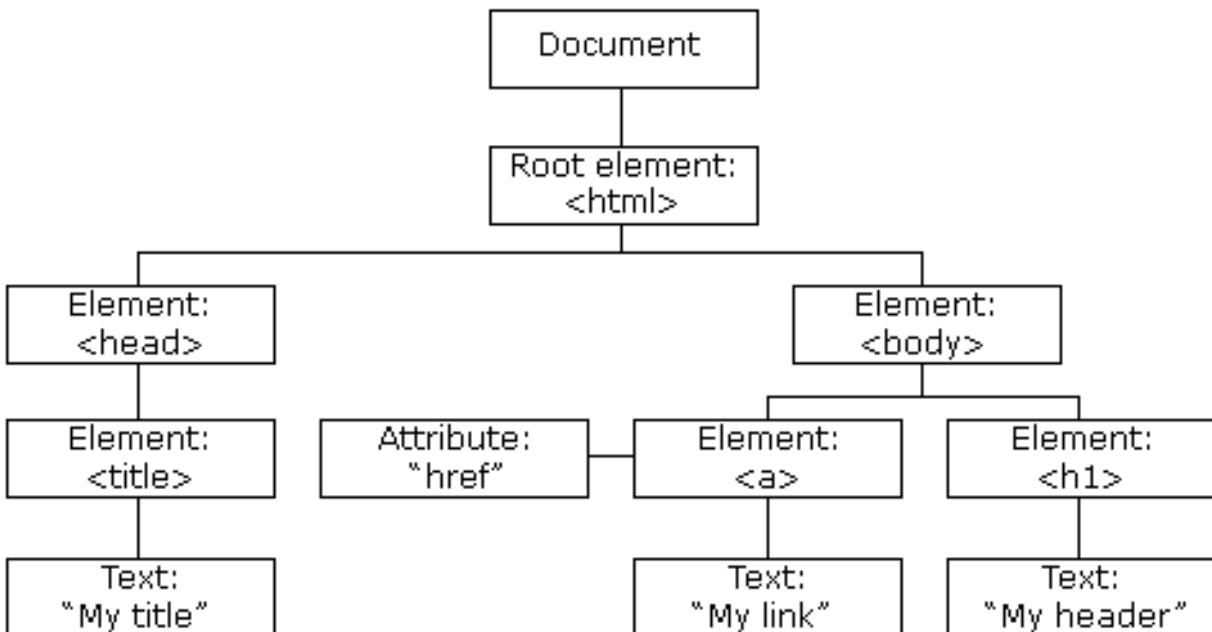


Figure 1: HTML DOM

A brief introduction to the document object model (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

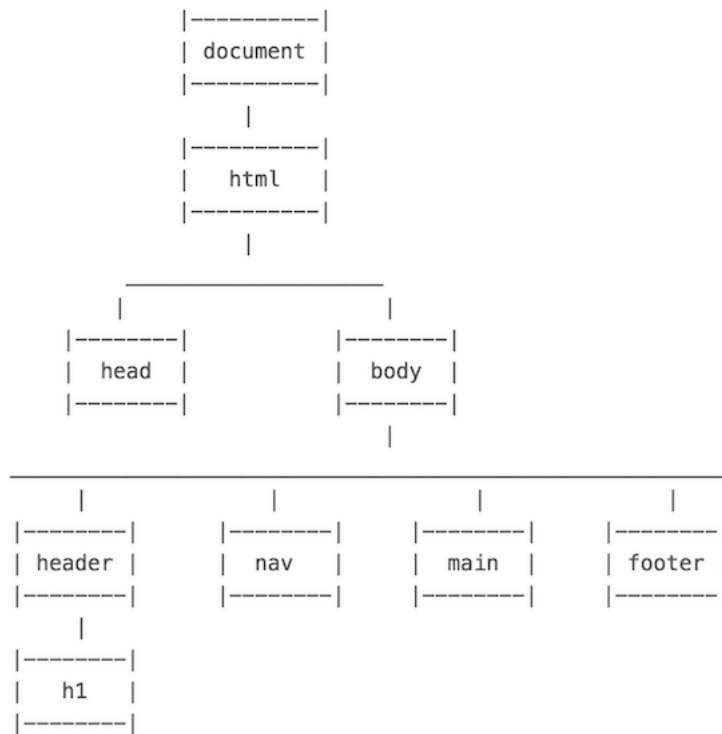


Figure 2: HTML DOM

DOM Basics - useful elements

element tag	usage & description
<code><html></code>	container element for a HTML document

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

NB: `<div>` and `` 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">
    <!-- demo-->
    <title>Demo</title>
  </head>
  <body>
    <header>
      <h1>Ancient Egypt</h1>
    </header>
    <nav>...</nav>

```

```

<main>
  <section>
    <p>
      Welcome to the Ancient Egypt information site.
    </p>
    <figure>
      
      <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 1

- 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

```

<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 Basics - `<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 `
` 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
```

- [Demo - HTML - Internal Anchors with Scroll](#)
-

Video - Hyperlink

TED: How the hyperlink changed everything

Source - [YouTube](#)

HTML Basics - `<body>` - part 4

images

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

```


```

- 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>
```

- [Demo - Woolf Online](#)
-

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
 - Demo - Basic Structural Example
-

Demos

- Basic Attribute
 - Basic Attribute 2
 - Basic Structural Example
 - DOM Basics - Sample
 - Traverse DOM tree with JavaScript generator
-

Resources

- Jaffe, Jim., *Application Foundations For The Open Web Platform*. W3C. 10.14.2014. <http://www.w3.org/blog/2014/10/application-foundations-for-the-open-web-platform/>
- JS Info - DOM Nodes
- W3 Docs for further details