Course Details

Lecturer

- **Name:** Dr Nick Hayward
- **Office hours**
  - *Tuesday by appointment*
- **Faculty Page**
Course Schedule

Important dates for this semester

- Project outline and mockup - presentation & demo
  - 24th September 2019 @ 7pm

- Mid-semester break
  - n.b. no formal class: 8th October 2019

- DEV week: 22nd to 29th October 2019

- DEV week - presentation & demo
  - 29th October 2019 @ 7pm

- Final class: 3rd December 2019

- Final presentation & demo
  - 3rd December 2019 @ 7pm

- Exam week: 9th December to 14th December 2019
  - Final assessment due on 10th December 2019
Coursework schedule

Presentations, reports &c.

- project outline & mockup
  - due Tuesday 24th September 2019 @ 7pm

- DEV week demo
  - due Tuesday 29th October 2019 @ 7pm

- final team demo
  - due Tuesday 3rd December 2019 @ 7pm

- final team report
  - due Tuesday 10th December 2019 @ 7pm
Initial Course Plan - Part 1

(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

- exercises and discussions (Total = 20%)
  - various individual or group exercises and discussions
- project outline & mockup (Total = 15%)
  - brief group presentation of initial concept and mockup
  - due Tuesday 24th September 2019 @ 7pm
- DEV week assessment (Total = 25%)
  - DEV week: 22nd to 29th October 2019
  - presentation & demo: 29th October 2019 @ 7pm
- end of semester final assessment (Total = 40%)
  - demo due Tuesday 3rd December 2019 @ 7pm
  - final report due Tuesday 10th December 2019 @ 7pm
Exercises & discussions

Course total = 20%

- exercises
  - help develop course project
  - test course knowledge at each stage
  - get feedback on project work

- discussions
  - sample websites and applications
  - design topics, UI and UX concepts

- extras
  - code and application reviews
  - various other assessments
  - peer review of demos
Development and Project Assessment

Course total = 80% (Parts 1, 2 and 3 combined)

Initial overview

- combination project work
  - part 1 = project outline & mockup (15%)
  - part 2 = DEV Week development & demo (25%)
  - part 3 = 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...
Course total = 15%

- begin outline 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...
    - chosen topic requires approval
  - presentation should include mockup designs and concepts
Assessment will include the following:

- brief presentation or demonstration of current project work
  - ~ 5 to 10 minutes per group
  - analysis of work conducted so far
  - presentation and demonstration
    - outline current state of web app concept and design
    - show prototypes and designs
  - due Tuesday 24th September 2019 @ 7pm
Course total = 25%

- 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

DEV week assessment will include the following:

- brief presentation or demonstration of current project work
  - ~ 5 to 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 Tuesday 29th October 2019 @ 7pm
Final Assessment

Course total = 40%

- 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...)
    - 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...
  - 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 demo
  - due on Tuesday 3rd December 2019 @ 7pm

- final report
  - due on Tuesday 10th December 2019 @ 7pm
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 324/424 - Fall 2019 @ LUC'
  - https://trello.com/csteach424

Slack group

- group for class communication, weekly discussions, questions, &c.
- Slack group - 'COMP 324/424 - Fall 2019 @ LUC'
  - https://csteach424.slack.com
Group projects

- add project details to course's Trello group, COMP 324/424 - Fall 2019 @ LUC
  - Week 1 - Project Details
  - https://trello.com/b/2gepsyfx/week-1-project-details

- create channels on Slack for group communication
  - please add me to the private channel

- start working on an idea for your project

- plan weekly development up to and including DEV Week
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
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*
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

Introducing Atom 1.0!

Source - YouTube - Introducing Atom 1.0
- 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>
  ...
</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>
```
Constructing HTML documents involves 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**
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 />
```
An element's \textit{end} tag also adheres to a pattern, again exactly as defined as following,

\begin{enumerate}
\item a \texttt{<} character
\item a \texttt{/} character
\item element's tag name (i.e. name used in matching start tag)
\item optional space characters (one or more...)
\item a \texttt{>} character
\end{enumerate}

For example,

\begin{verbatim}
<!-- element's matching end tag -->
</div>
\end{verbatim}

\textbf{NB: void} elements, such as \texttt{<br \>} or \texttt{<img \>}, do \textit{not} specify end tags.
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 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
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
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
<!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.body.getElementsByTagName('blockquote');

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

- example - Basic Attribute
```javascript
/*
 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.

```html
<!DOCTYPE html>
```

or

```html
<!doctype html>
```

- doctype we add for HTML5 rendering

- not a HTML element, simply tells the browser required HTML version for rendering
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...
HTML DOM
# DOM Basics - useful elements

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

**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...
Welcome to the Ancient Egypt information site.

Temple at Philae in Egypt is Ptolemaic era of Egyptian history.

foot of the 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...
Demos

- Basic Attribute
- Basic Attribute 2
- DOM Basics - Sample
Resources

  http://www.w3.org/blog/2014/10/application-foundations-for-the-open-web-platform/

- MDN - Using Dynamic Styling Information
- The Unicode Consortium
- Unicode Information
- Unicode examples
- W3 Docs for further details