Comp 324/424 - Client-side Web Design

Fall Semester 2024 - Week 1

Dr Nick Hayward

Course details

Lecturer

- Name: Dr Nick Hayward
- Office hours
  - Tuesday afternoon by appointment @ LSC
- Faculty Page

Course schedule

Important dates for this semester

- Mid-semester break
  - NO class - Tuesday 8th October 2024
- Dev week demo & presentation
  - due Tuesday 15th October 2024 @ 7pm
- Final project demo & presentation
  - due Tuesday 3rd December 2024 @ 7pm
- Exam week: 9th to 14th December 2024
- Final project report
  - due Tuesday 10th December 2024 @ 9.30pm

Coursework schedule

Presentations, reports &c.

- Dev week demo & presentation
  - due Tuesday 15th October 2024 @ 7pm
- Final project demo & presentation
  - due Tuesday 3rd December 2024 @ 7pm
- Final project report
  - due Tuesday 10th December 2024 @ 9.30pm

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

- quizzes, exercises, and discussions
    - each quiz will include multiple choice questions
    - class and weekly discussion topics
- Dev Week demo & presentation
    - due Tuesday 15th October 2024 @ 7pm
- end of semester final assessment
    - final presentation and demo due Tuesday 3rd December 2024 @ 7pm
    - final report due Tuesday 10th December 2024 @ 9.30pm

Credits available during course

- course participation = 30
- quizzes = 1 per question
- discussions &c. = 5 per discussion
    - ~ 6 discussions during semester
- course project
    - Dev week = 25
    - final demo & report = 50

Participation

Course total = 30

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

Quizzes, exercises & discussions

- quizzes and exercises
  - test course knowledge at each stage
  - help develop course project
  - 1 credit per quiz question
- discussions
  - sample websites, games, services...
  - design topics, UI and UX concepts
  - topics posted to Sakai Forum
  - 5 credits per discussion topic

Project assessment

Initial overview

- combination project work
  - part 1 = Dev week demo - 25 credits
  - part 2 = Final demo and report - 50 credits
- 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...
    - **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.

Dev week demo & assessment

Course total = 25 credits

- design and 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 Tuesday 15th October 2024 @ 7pm

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
- final project report
  - written summary of project design, development, and research
  - no word/page limit…
  - suggested report outline will be provided
- final presentation and demo due Tuesday 3rd December 2024 @ 7pm
- final report due Tuesday 10th December 2024 @ 9.30pm

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)

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,

```html
<!-- 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,

```html
<!-- 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

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

```html
<!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">
```

```html
      <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

```javascript
/*
* 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 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

```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.

```
<!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

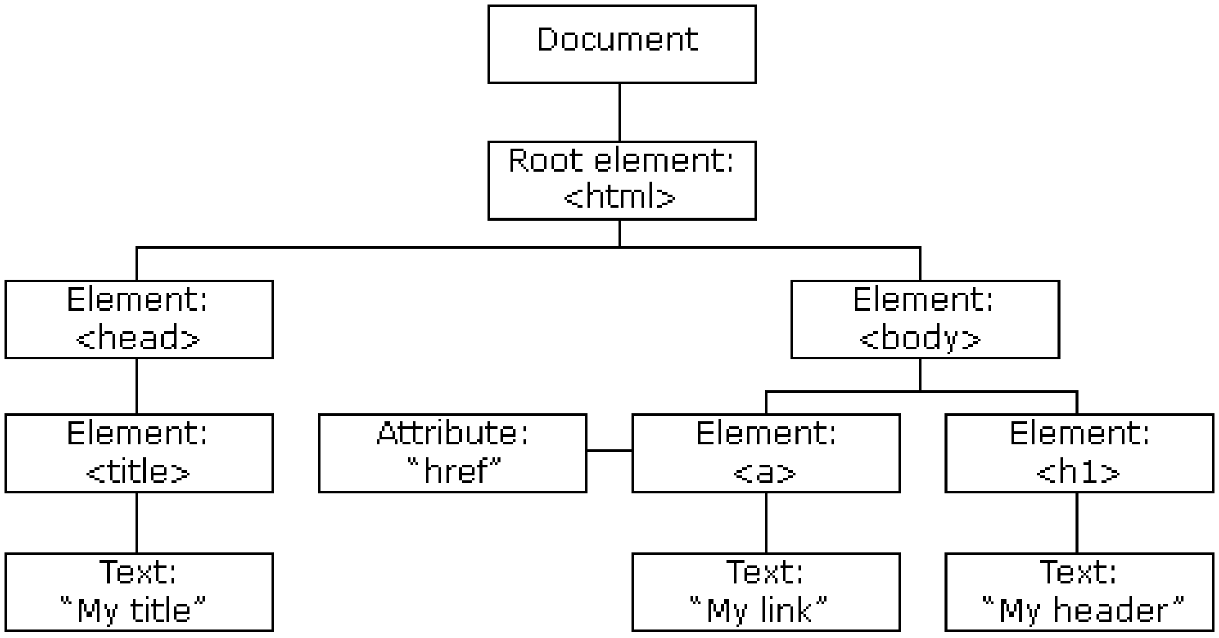

Figure 1: HTML DOM

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

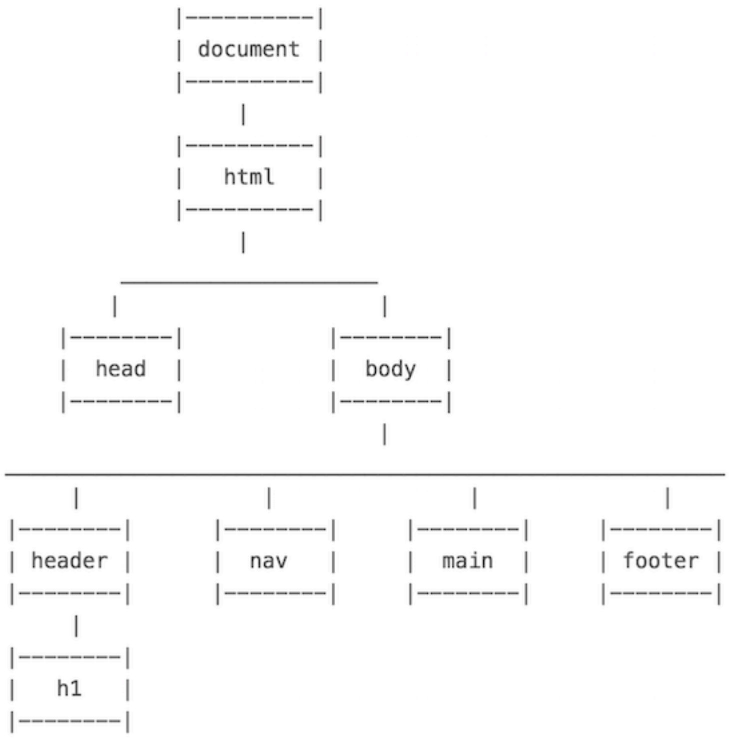

Figure 2: HTML DOM

DOM Basics - useful elements

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

element tag	usage & description
`<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">
    <!-- 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>
          <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...

Demos

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

References

- 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