

Comp 324/424 - Client-side Web Design

Fall Semester 2017 - Week 1

Dr Nick Hayward

course details

Lecturer

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

course schedule

Important dates for this semester

- Labor Day - 4th September 2017
 - **n.b.** *no formal class: 4th September 2017*
- DEV week: 9th to 16th October 2017
 - **n.b.** *no formal class: 9th October 2017*
 - *presentation & demo: 16th October 2017 @ 4.15pm*
- Thanksgiving break: 22nd to 25th November 2017
- Final class: 4th December 2017
 - *presentation & demo: 4th December 2017 @ 4.15pm*
- Exam week: 11th December to 16th December 2017
 - *Final assessment due on 11th December 2017 by 4.15pm*

coursework schedule

Presentation, reports &c.

- DEV week demo
 - *due Monday 16th October 2017 @ 4.15pm*
- final team demo
 - *due Monday 4th December 2017 @ 4.15pm*
- final team report
 - *due Monday 11th December 2017 @ 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 16th October 2017 @ 4.15pm
 - *final assessment (Total = 40%)*
 - demo due 4th December 2017 @ 4.15pm
 - report due Monday 11th December 2017 @ 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. 4 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** PHP, Python, Ruby, Go, XML, Bootstrap...
 - *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** PHP, Python, Ruby, Go, XML, SQL, Bootstrap...
- demo and project report
 - *due on Monday 16th October 2017 @ 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** PHP, Python, Ruby, Go, XML, SQL, Bootstrap...
- presentation and demo - live working app...
 - *due on Monday 4th December 2017 @ 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 11th December 2017 @ 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 I

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 2017 @ LUC*
 - *Week 1 - Project Details*
 - *<https://trello.com/b/mkDZFkad/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
 - *9th to 16th October 2017*
 - *DEV week demo on 16th October 2017*

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 I

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*

Video - Microsoft Edge

Introducing Microsoft Edge: The New Windows 10 Browser



Source - YouTube - Introducing Microsoft Edge

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 I.0

Introducing Atom 1.0!



Source - YouTube - Introducing Atom I.0

Browser technologies

- browser rendering engines
- web standards
 - *HTML*
 - *CSS*
 - *XML*
 - *XHTML*
- application foundations
- open web platform

Browser rendering engines

- Until 2013, *WebKit* was the default rendering engine for both Safari and Chrome
- Google switched to the open source alternative, *Blink*, whilst Safari continues to use *WebKit*
- Firefox continues to use the *Gecko* rendering engine
- Microsoft's new Edge browser uses a new proprietary engine called *EdgeHTML*
 - *fork of the Trident rendering engine*
 - *Microsoft notes that EdgeHTML will largely behave like Chrome and Safari*

Web standards

- many disparate web standards
 - *include the broader internet beyond www...*
 - *subset of particular interest to web developers*
- primary web standards
 - **Recommendations** published by the W3C (World Wide Web Consortium)
 - **Unicode** standards published by the Unicode Consortium
 - **ECMA** standards now published by ECMA International
 - *more to come later in the semester...*

W3C Recommendations - part I

Recommendations of the W3C of particular interest includes

- HTML (HyperText Markup Language)
 - *key building block of the web*
 - *stored as plain text*
 - *includes selection of tags*
 - *e.g. headings, images, links, lists, paragraphs, tables...*
- CSS (Cascading Style Sheet)
 - *commonly used with HTML*
 - *controls rendering and stylistic characteristics of a web page*
 - *CSS concerned with presentation of the structure and data*

W3C Recommendations - part 2

Recommendations of the W3C of particular interest includes

- XML (Extensible Markup Language)
 - *often considered a meta-language*
 - *follow-on from SGML (Standard Generalised Markup Language)*
 - *used to describe data & not presentation, rendering of data*
 - *element tags not inherently pre-defined*
 - *foundation for many XML languages such as RSS, MathML, MusicML...*
- XHTML (Extensible HyperText Markup Language)
 - *attempt to update and rewrite HTML based on experience from XML*
 - *very similar to HTML with stricter rules*
 - *e.g. HTML lapse in enforcing case sensitivity, closing tags...*
 - *strict rules structure inherited from XML style languages*

Video - W3C Web standards for the future



01:50



Source - Vimeo - W3C

Application foundations - Part I

W3C, on the occasion of HTML5 achieving the status of W3C Recommendation, proposed

a set of technologies for developing distributed applications with the greatest interoperability in history.

Application Foundations for the Open Web Platform

- known as the OWP (Open Web Platform)
- driven by a blog post by Jeff Jaffe in October 2014
 - *suggested W3C's next priority should be Open Web Platform*
 - *OWP should be easier to use for developers*

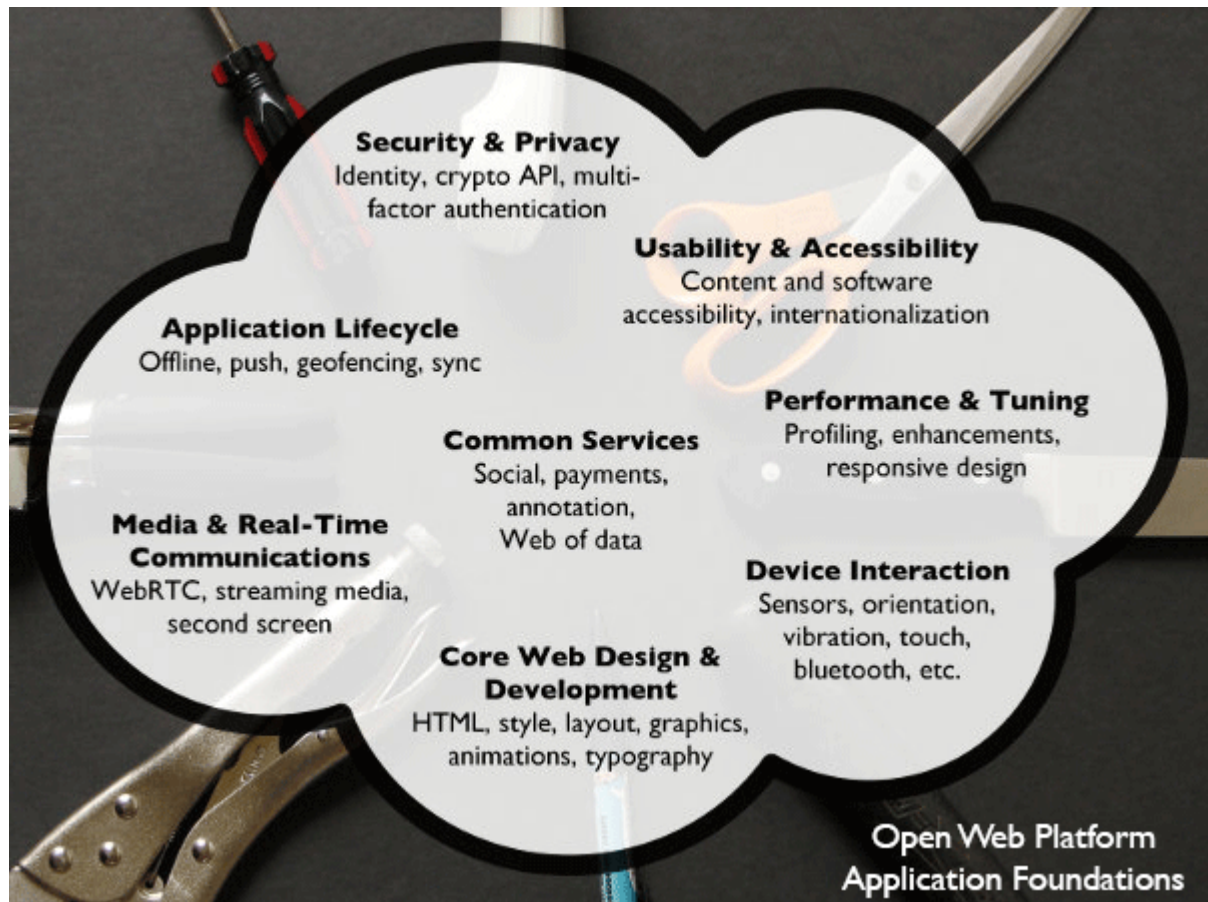
Application foundations - Part 2

Jaffe defined eight **Foundations** in that particular post, which include the following

- Security and Privacy
- Core Web Design and Development
- Device Interaction
- Application Lifecycle
- Media and Real-Time Communications
- Performance and Tuning
- Usability and Accessibility
- Services

Further information and updates can be found at the W3C's [App Foundations](#) website.

Image - Open Web Platform



Source - W3C

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 - Element syntax - part I

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

HTML - attribute syntax - part I

- 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](#)

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/>
- W3 Docs for further details