Comp 125 - Visual Information Processing

Spring Semester - 18th January 2018

Dr Nick Hayward

course details

Lecturer

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

TA

- Name: Catherine Litten

Course Schedule

Important dates for this semester

- Class schedule = Monday, Wednesday, & Friday @ 1.40pm
- Spring Break: 5th to 9th March 2018
  - ***n.b.*** *no formal class: 5th, 7th, & 9th March 2018*
- Easter holiday: 29th March to 2nd April 2018
  - ***n.b.*** *no formal class: 30th March & 2nd April 2018*
- Final class: 27th April 2018
  - *presentation & demo: 23rd, 25th, & 27th April 2018 @ 1.40pm*
- Exam week: 30th April to 5th May 2018
  - *Final assessment due on 3rd May 2018 by 1pm*

Assignments and Coursework

Course will include

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

Coursework will include

- quizzes or exercises at the end of each section (Total = 60%)
  - *course quizzes*
  - *exercises to test course knowledge, help develop understanding of course material...*
  - *code and application reviews*
  - *class discussions using Slack*
  - *various other assessments...*
- demo and report of final assessment (Total = 40%)
  - *demo during final week of semester*
  - *Monday 23rd April, Wednesday 25th April, and Friday 27th April 2018*
  - *report due 3rd May 2018 @ 1pm*

n.b. no final exam

Final Assessment

Initial overview

- develop an app concept and prototype
- working app (as close as possible...)
  - *must use technologies outlined during the course*
- show and explain code used to develop the app
- explain design decisions
  - *describe patterns used in design of app*
  - *layout choices...*
- show and explain implemented differences during app development
  - *where and why did you update the app?*
  - *perceived benefits of the updates?*
  - ...

Goals of the course

- introduction to programming and Computer Science
  - *general concepts*
  - *methodologies*
  - *patterns*
- introduction to web-based graphical design
- getting started with application design and development
  - *builds on web-based technologies*

Course Resources

Website

- course website is available at http://csteach125.github.io
  - *timetable*
  - *course overview*
  - *course blog*
  - *assignments & coursework*
  - *bibliography*
  - *links & resources*
  - *notes & material*

GitHub

- course repositories available at http://github.com/csteach125/
- weekly notes, examples, and source code (where applicable)

Slack

- Slack group available at https://csteach125.slack.com
- course updates, information on weekly assignments, general news, discussions...

n.b. no Sakai

Course technologies

- JavaScript (JS)
  - *ES6 (ECMAScript 2015)*
- HTML5
- CSS
- JS-based visualisation libraries
- various datastores
  - *local and online examples*

intro

- *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*
  - *using JavaScript*
- 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...