Course Syllabus: INFO 6250

This course covers Web Development, focusing on both the fundamentals and from them modern development practices for the web. The class will use Javascript on both the front- and back-ends of web application development, but the lessons learned will be applicable to many languages.

- Basic git use as a version control system for shared applications and development
- How information flows in a web application between various machines
- Modern JS development for the frontend (both "vanilla" JS and React-based)
- NodeJS development of a backend end, both server generated content using the express framework (minimal) and service development.
- Web security fundamental practices, both front- and back-end.
- The basics of how many development teams breakdown application needs and complete work

What is NOT covered

- How to program in general
- HTML and CSS details
- Languages other than Javascript
- Mobile development
- Accessibility (a11y), Internationalization (i18n), or Localization (l10n)
- SQL/NoSQL usage or database architectures/maintenance

Grading: I reserve the right to adjust based on your final demonstration of applied knowledge.

```
20% Assignments, Participation, Quizzes
60% Exams (3 Exams, 20% each)
20% Final Project
```

Basic Requirements and Expectations:

- Basic familiarity with CSS and HTML is assumed (see https://developer.mozilla.org/en-US/docs/Learn)
- Basic exposure to programming concepts (variables, functions, looping) is assumed
- You will have to use git and github.com following the instructions given
- There is no textbook for the class, but a number of online articles will be shared.
- Many topics will be introduced in class but require you to perform additional research/experimentation
- Additional software (without cost) is required. Installation and configuration is your responsibility (Mac, Windows, or *nix)
- Students should ask questions where anything is unclear
- A great deal of work will be done online, in and out of class

A more detailed listing of requirements and expectations will be shared in the class github repository

Expected Class Schedule (subject to change):

Section 1: Web Fundamentals

- Protocols, Web, HTTP, servers/webservers, browsers, clients, URL/URI, HTTP as stateless, request/response, headers/body
- The role of HTML, CSS, and JS
- the DOM, semantic HTML, MDN, the Browser Wars, evergreen browsers, the unreliability of not-that-old information
- HTML best practices, CSS best practices
- absolute vs relative paths/URLs
- multiple-page web applications
- static vs dynamic assets, client-side/server-side
- cookies, localStorage
- programming languages as communication, idioms, static/dynamic languages, weak/strong typing
- Javascript syntax, NodeJS, npm/yarn, package.json, global vs local installs, JSON
- debugging JS, unit tests, testing pyramid, TDD
- functions as objects, prototypes, 'this'(context), callbacks, threads, try/catch, closures, scopes
- Object Oriented Programming, Procedural programming, Functional Programming
- templates, Model-View-Controller(MVC)
- application state, state in model vs state in DOM

At the end of Section 1 you should be able to write a simple multiple page web application using NodeJS that serves semantic HTML and styles with CSS. You will receive from github repository updates and submit your work via Pull Requests (PRs) in the same fashion that many employers conduct their work.

Section 2: The Recent Web

- HTTPS/SSL, public-key encryption, certificates, Authentication, Authorization
- asynchronous events (async), Promises, XHR/fetch/AJAX, HTTP verbs (methods), REST, GraphQL, services/endpoints
- polyfills, minifiers, linters, bundlers, transpilers, CSS preprocessors, builds
- Frontend frameworks/libraries, React, virtual DOM, JSX, Single Page Applications (SPA)
- props vs state, Pure components vs stateful components, render props, Higher Order Components
- Same Origin Policy(SOP), CORS, XSS, XSRF/CSRF
- Application state management

At the end of Section 2 you should be able to write a simple single page web application (SPA) calling RESTful, external services, and use NodeJS to provide those service endpoints

Section 3: The Modern Web Industry

- Databases, CRUD, SQL and NoSQL, SQL Injection, OWASP
- Progressive Web Apps (PWA)
- as-a-service (PaaS, FaaS)
- Agile, change management, Software Patterns
- Mockups, wireframes, prototypes
- Obfuscation, copyrights, and module licensing
- Password hashing/salting, JWT, Oauth, OIDC
- Isomorphic/Universal Javascript
- JS on other platforms, websockets

At the end of Section 3 you should be able to analyze provided designs to matching write a complex single page web application (SPA) and prior to actual coding identify potential problems with development due to insufficient/poor requirements.

Final Projects Due