Computer Science S-75: Building Dynamic Websites
Harvard Summer School
Summer 2012

Project 1: C$75 Finance 2.0

due Wed 7/25, noon ET

This version of Project 1 is meant <u>only</u> for students who have already taken CS50 (or CSCI E-52).

Goals.

- Introduce you to Ajax.
- Introduce you to third-party APIs.

Academic Honesty

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed (*e.g.*, by some problem set or the final project). Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or accidentally shared in the course's virtual classroom) or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this course that you have submitted or will submit to another. Nor may you provide or make available your or other students' solutions to Project 0, Project 1, or Project 2 to individuals who take or may take this course (or CSCI E-75) in the future.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss problem sets with classmates, but you may not share code. You may also turn to the Web for instruction beyond the course's lectures and sections, for references, and for solutions to technical difficulties, but not for outright solutions to problems on projects. However, failure to cite (as with comments) the origin of any code or technique that you do discover outside of the course's lectures and sections (even while respecting these constraints) and then integrate into your own work may be considered academic dishonesty.

If in doubt as to the appropriateness of some discussion or action, contact the staff.

All forms of academic dishonesty are dealt with harshly. If the course refers some matter to the Administrative Board and the outcome for some student is disciplinary action, the course reserves the right to impose local sanctions on top of that outcome for that student that may include, but not be limited to, a failing grade for work submitted or for the course itself.

Grades.

Your code (CSS, HTML, JavaScript, PHP, *etc*.) will be evaluated along the following axes.

Scope. To what extent does your code implement the features required by our specification?
Correctness. To what extent is your code consistent with our specifications and free of bugs?
Design. To what extent is your code written well (*i.e.*, clearly, efficiently, elegantly, and/or logically)?
Style. To what extent is your code readable (*i.e.*, commented and indented with variables aptly named)?

BitBucket.

☐ So that you have a place to store revisions of this project privately, head to `https://bitbucket.org/` and create a **project1** repo!

Implementation.

☐ Your mission for this project is to implement C$75 Finance 2.0, a mashup that allows users to retrieve a lot more information about individual stocks than CS50 Finance did. The overall design and aesthetics of this site are ultimately up to you, but we do require that your site meet some requirements. All other details are left to your own creativity and interpretation. You are welcome to start this project from scratch or, if you'd prefer, build upon your code from CS50. Just realize that only the features below are required for this project. (In other words, this project does not require that users be able to register, log in, buy, or sell.)

Feature Requirements.

☐ Your site's homepage must allow a user to input a company's stock symbol via a form. Upon submission of that form via (a non-Ajax) GET, the user should see a page with:

- ☐ That same form, so that they can input another stock symbol.
- ☐ That stock's Last Trade price. That price should be updated every 10 seconds via Ajax.[1]
- ☐ A chart depicting that stock's daily price history. By default, the chart must display the past 5 (trading) days of (Prev Close) prices, but the user must be able to change that range (via links or form inputs) to any of 5d, 1m, 3m, 6m, YTD, 1y, 2y, 5y, and 10y.[2] When a new range is selected, the chart must be updated via Ajax. The chart must be an interactive, JavaScript-based chart from Google's Visualization API (*i.e.*, not an image or Flash-based). Odds are you'll want to use a Dygraph or Line Chart.[3]
- ☐ Links to recent articles about the company.

Technical Requirements.

☐ You're welcome to develop your site on any computer using any IDE or text editor, even without using the CS50 Appliance, but you must ultimately ensure that it works within the CS50 Appliance at a URL of `http://project1/` when installed in `/home/jharvard/vhosts/project1/`.

☐ Only files that should be web-accessible should live in `project1/html/`; everything else should live in `project1/` or some (other) subdirectory therein.

☐ Your site must pull its data from Yahoo! Finance.[4]

[1] To avoid same-origin policies, you might want to leverage JSONP from `http://developer.yahoo.com/yql/`.
[2] For the purposes of these ranges, we leave it to you to decide how many days are in a month and in a year.
[3] `https://developers.google.com/chart/interactive/docs/gallery`
[4] Look for CSVs of interest at URLs like `http://finance.yahoo.com/q?s=AAPL` and `http://finance.yahoo.com/q/hp?s=AAPL`. And look for RSS at URLs like `http://finance.yahoo.com/q/h?s=AAPL`.

- ☐ You are welcome, but not required, to use any of the JavaScript libraries recommended in Lecture 6's slides.
- ☐ You should avoid redundant HTML; factor out markup common to multiple pages using PHP functions or "templates" (*i.e.*, PHP files that you `require` in others).
- ☐ Your markup language should be valid (or "tentatively" valid) HTML5, as per `http://validator.w3.org/`, unless some feature of your site requires otherwise (for the sake of some browser); explain in HTML comments any intentional invalidities. Your HTML should also be as pretty-printed as possible. Your CSS need not be valid.
- ☐ Your PHP must be extensively commented and be as pretty-printed as possible.
- ☐ You may use a WYSIWYG editor to generate HTML and/or CSS that you would like to use in your site.
- ☐ If you integrate third-party CSS or JavaScript libraries into your project, cite their origin with comments.
- ☐ If you incorporate or adapt snippets of PHP code from the Web into your project (*e.g.*, examples from `php.net`), cite the code's origins with comments.
- ☐ If you incorporate images from the Web into your project, cite the images' with comments.
- ☐ Your website must appear and behave the same on the latest versions of at least two of these browsers:
  - ☐ Chrome
  - ☐ Firefox
  - ☐ Internet Explorer
  - ☐ Opera
  - ☐ Safari

Exit Interview.

☐ Once done with your site, put together a readme in a file called `README` that lives in the same folder as the rest of your project.

Treat this readme as your opportunity not only to explain but to justify your design decisions. Tell us why you modeled your database tables as you did. Tell us why you chose, say, select menus over radio buttons for some feature. Tell us with which two (or more) browsers we should evaluate your site. And give us an overall sense of how your site works (*e.g.*, tell us which files do what). But still be succinct; keep this readme to just a few paragraphs in length.

How to Submit.

☐ Just prior to this project's deadline, instructions for submitting your work will be announced at `https://www.cs75.net/`. Be sure to look for those directions and then submit your work prior to this project's deadline.