6.170 – Software Studio
Introduction

Adam Chlipala, Mark Day, Daniel Jackson
September 3, 2014
Purpose of the Class, in a Nutshell?

More introspective than other CS classes: Think about the **process** for going from ideas to robust software systems.
Four Pillars

Hacking
ability to build software that works today

Design
which features? how to implement them?

Robust Coding
code-level techniques to support evolution

Professionalism
interacting with collaborators and customers

30% of grade
30% of grade
30% of grade
10% of grade
Concrete Example Domain: Web Applications

Can only practice the core skills in a concrete setting. We picked dynamic web applications. Guaranteed buzzword-complaint (e.g., AJAX, REST, ...). Side benefit of the course: learn about modern web development.

Big change from previous 6.170 offerings: All programming will be in JavaScript this term!
Who are we and why should you listen to us?
Adam Chlipala
<http://adam.chlipala.net/>

- Assistant professor in EECS/CSAIL since 2011
- Research in tools for programmers
  - (usually involving applied mathematical logic)
- About 2 (noncontiguous) years of commercial software engineering experience [starting with web apps in late 90's]
- But developing many open-source-y web apps since 1996
- Main developer of Ur/Web*, an avant-garde programming language for web applications
Mark Stuart Day

- Ph.D., MIT, 1995
- Lotus (bought by IBM for $3.5B, 1995)
- SightPath (bought by Cisco for $800m, 2000)
- Riverbed (IPO, 2006; Annual revenue $1B+)
- Notable systems: Riverbed Steelhead, Lotus Sametime, Cisco ACNS, IETF IMPP, Argus, Theta, Thor
- 30+ patents, including SSL optimization
- Gory details: www.markstuartday.com
Daniel Jackson

background
BA in physics (Oxford), PhD in CS (MIT)
software engineer (Logica UK)
research at Bell Labs and DEC
prof at CMU, then at MIT since 1997

interests
automatic analysis of software
safety & dependability
security analysis & design
new software paradigms
software design methods
teaching
Teaching Assistants

Bryan Collazo*
Vivek Dasari*
Cynthia Jing#
Rebecca Krosnick*
Charles Liu*
Michael Maddox*
Kathryn Siegel#
Kimberly Toy#
Evan Wang*
Emily Zhang#

* → MEng
# → undergrad
6.170 Stellar Site

http://stellar.mit.edu/S/course/6/fa14/6.170/

Please follow the instructions in an announcement, about filling out an enrollment form for this class.
Course Content

- Essential program structuring ideas, illustrated with JavaScript
  - Functional programming, objects, and related ideas
- Anatomy of web applications & associated protocols
- Data modeling & database engines
- Conceptual design for software
- Design of web service APIs
- Security
- How to give a presentation

And: learn by application in projects

Lecture stops halfway through the semester.
Good News #1: No exams (only short quizzes)

Lectures
big ideas &
example code

Recitations
in the trenches with
web technologies

5% of grade
(participation)

Quizzes
check your
understanding of big
ideas

10% of grade

Individual Projects
practice design & implementation in
relatively small chunks

30% of grade

Small Team Project
increase the scale & practice teamwork

20% of grade

Large Project
your team builds the app of your choice;
all class time in last month+!

35% of grade
Quizzes

• Test on the basic ideas from lecture.
• In lecture, 15 minutes each, multiple choice.
• One after each major unit of material.
• First one is two Mondays from today, on functional programming!
Projects

Individual projects
#0: Getting started with JavaScript
#1.1: Game of Life simulator in your browser
#1.2: Graphical user interface for Game of Life
#2.1: Build a simpler Twitter clone.
#2.2: Extend your Twitter clone with more complex data model.

Team projects
#3.1: Design an app that will use some new web-service API.
#3.2: Implement the API.
#3.3: Implement rest of app, using API.
Final: App of your choice
Collaboration & Grading Policies

- It's always OK to use publicly available software, with proper attribution, unless it comes from a prior version of 6.170, or if it is a complete implementation of one of our projects.
- It's not OK to get detailed project help from people outside the course staff & students.
- **Individual projects:** *no collaboration.*
- **Team projects:** within-team collaboration is OK.
- Slack days: four late days on *individual* projects to spend as you choose.
Administrivia

- Fill out the **6.170 enrollment form** on the web (by end of today).
- **Project 0** (basic JavaScript practice) due on Sunday.
- **Recitation** tomorrow:
  - You may go to whichever section you like each week.
  - Come with the following installed on your laptop: **Chrome** (unless you're already comfortable with a different browser's JavaScript development tools) and **Git**.
  - Will be an installation & usage bootcamp for some of the programming tools we'll use in 6.170.

Robust, Secure, Large, Evolvable
Four Crucial Goals in Software Engineering

**Robust?** Won't be broken by actions of well-meaning users, random hardware failures, etc.

**Secure?** Won't be broken by *malicious* users.

**Evolvable?** Easy to extend with new ideas by you & your team.

**Large?** Lots of code, to support lots of features!

**Large?** Efficient & spread across multiple servers. [i.e., *scalable*] (not a focus in 6.170)
Rest of today's class:
Let's build a simple web application, without using any of the astonishingly good ideas we'll learn in 6.170.

Don't worry about understanding the coding details. We'll come back and cover those bottom-up in future lectures & recitations.

Instead, let's think about what in this process leaves us skeptical that we can meet our goals for robust, secure, large, evolvable software.

Code found in a public GitHub repository:
http://github.com/6170-fa14/6.170-lectures
Complaints I Came Up With Before Class

- All in one file!
- Not otherwise broken into separately understandable pieces
- Generating HTML with JavaScript string munging is ugly.
- The pages we generate aren't very nice-looking.
- Potentially distracting to write page-handler code separated in text from place where it's used.
- This HTTP processing can get quite repetitive, even though it's super boring.
- Fun code injection attacks! (Enter HTML in textboxes.)
- Server only handles one request at a time. Might not be enough for a popular site.
- If we wanted to go concurrent, hard to see how to do it in this style. Tricky to maintain data structure invariants.
- Similarly, hard to see how to be robust in face of random hardware failures. Do data structures come back up in inconsistent states?
- How did we know this was a good app to implement, anyway?

Not addressed by 6.170:
- No static types. :-(
- Have to rely on testing to find mistakes like using a typoed URL or ID in JavaScript code.
Administrivia

• Fill out the **6.170 enrollment form** on the web (by end of today).
• **Project 0** (basic JavaScript practice) due on Sunday.
• **Recitation** tomorrow:
  – You may go to whichever section you like each week.
  – Come with the following installed on your laptop: **Chrome** (unless you're already comfortable with a different browser's JavaScript development tools) and **Git**.
  – Will be an installation & usage bootcamp for some of the programming tools we'll use in 6.170.