software studio

introduction

Daniel Jackson
welcome
introduction

my career
› Physics undergrad (MA, Oxon)
› Computer science (PhD, MIT)
› Assistant professor (CMU)
› Professor (MIT)
› Associate Director (CSAIL)

my interests
› formal methods (http://alloy.mit.edu)
› design for usability, security, safety
› photography (http://dnj.photo)
› ethics, spirituality, aesthetics

some involvements
› NAS study chair: ‘sufficient evidence’
› MISTI-MEET faculty director
› MIT Hillel president
instructors

Eirik

Michael
teaching assistants

Kim, Harihar, Henry, Donald, Rachel, Katie, Tony, Neha, Dishaan
purposes of today’s class

after today’s class, you should
understand class logistics & structure
know what’s expected of you
understand two key software design ideas
... and be able to apply them
what you’ll do
two halves

first half
seven weekly solo assignments
2 classes + 1 recitation/week
no collaboration

second half
team project
mentoring + occasional classes
<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before 2pm</td>
<td>sleep</td>
<td></td>
<td>sleep</td>
<td>recitation (one hour, 10am-4pm)</td>
<td>sleep</td>
</tr>
<tr>
<td>2:30-4pm</td>
<td>class</td>
<td>sleep</td>
<td>class</td>
<td>pset due</td>
<td>pset</td>
</tr>
<tr>
<td>11pm</td>
<td>quiz due</td>
<td></td>
<td>quiz due</td>
<td>pset due</td>
<td></td>
</tr>
</tbody>
</table>

*Note:* Monday and Friday are dedicated to sleep. Tuesday and Wednesday focus on classes and recitations. Thursday includes a recitation and quiz due. Friday ends with a pset due.
scheduling details

quizzes
after each class, due 11pm

problem sets
out on Monday
material taught by Thursday
due next Thursday at 11pm

recitations
start between 10am and 3pm
choose the best hour

office hours
announced on Stellar
hand-in flexibility

slack days
three in total, use as you please

extensions
only with Medical or S3 notes
for all other circumstances, plan your time & slack days

lateness
10%/day penalty

what constitutes hand-in?
submitting hand-in form
with URL of commit
## Grading

<table>
<thead>
<tr>
<th></th>
<th>Team</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Final Project</strong></td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td><strong>Programming Assignments</strong></td>
<td>32%</td>
<td></td>
</tr>
<tr>
<td><strong>Design Assignments</strong></td>
<td>16%</td>
<td></td>
</tr>
<tr>
<td><strong>Quizzes</strong></td>
<td>6%</td>
<td></td>
</tr>
<tr>
<td><strong>Participation</strong></td>
<td>6%</td>
<td></td>
</tr>
</tbody>
</table>
tips

get help
piazza
office hours

start early
get a sense of what you’ll need
take advantage of staff
best way to debug: take a break

be incremental
build it up, easy first
check every little feature as you add it
be a scientist: try it and see
Gitless: a version control system

About Gitless

Gitless is an experimental version control system built on top of Git. Many people complain that Git is hard to use. We think the problem lies deeper than the user interface, in the concepts underlying Git. Gitless is an experiment to see what happens if you put a simple veneer on an app that changes the underlying concepts. Because Gitless is implemented on top of Git (could be considered what Git pros call a "porcelain" of Git), you can always fall back on Git. And of course your coworkers you share a repo with need never know that you're not a Git aficionado.

Check out the documentation to get started. If you are a novice user that never used any version control system the documentation should be enough to get you started. If you are a Git pro looking to see what's different from your beloved Git you'll be able to spot the differences by glancing through the Gitless vs. Git section.

For installation instructions see the readme file. After installation, you should be able to execute the gl command. The current Gitless version is 0.7 which was released on 4/2015 (release notes).

confused by git?

Santiago Perez De Rosso

sign up for training session this **Thursday** at 7pm in 32-G882
enrollment form
due today by 11pm

first recitation
tomorrow, pick your time

JavaScript warmup
due Sunday 11pm

office hours
see Stellar

Gitless
Thursday 7pm in 32-G882
sign up link on Stellar
our first quiz
Great designers quiz

Who is Apple's chief designer?
- Tim Cook
- Jonathan Ive
- Steve Jobs

What company did Dieter Rams design for?
- Alessi
- Braun
- Sony

What does Frank Gehry design?
- Buildings
- Computers
- Typefaces

submit

JavaScript Alert
Congratulations! Your score is 1 out of 3
<!doctype html>
<html>
<head>
  <title>Quiz</title>
  <script src="jquery-2.1.4.min.js"></script>
</head>
<body>
  <p style="font-size:16pt"><b>Great designers quiz</b></p>
  <p><b>Who is Apple's chief designer?</b></p>
  <input type="radio" name="q1" value="Tim Cook">Tim Cook
  <br>
  <input type="radio" name="q1" value="correct">Jonathan Ive
  <br>
  <input type="radio" name="q1">Steve Jobs

  <p><b>What company did Dieter Rams design for?</b></p>
  <input type="radio" name="q2" value="Alessi">Alessi
  <br>
  <input type="radio" name="q2" value="correct">Braun
  <br>
  <input type="radio" name="q2">Sony

  <input type="button" value="submit" onclick="checkedOptions = $('input:radio:checked');
  score = 0;
  for (i = 0; i < checkedOptions.length; i++)
    if ($(checkedOptions[i]).val() == 'correct') score++;
  alert('Congratulations! Your score is ' + score + ' out of 3');">"</body></html>
team up
2-3 students/team

study the code
check you understand how it works

identify flaws & consider how easy to...
add a question
change the appearance
penalize for skipped questions
add another quiz on the same page
quiz redux
Great designers quiz

Who is Apple's chief designer?
- Tim Cook
- Jonathan Ive
- Steve Jobs

What company did Dieter Rams design for?
- Alessi
- Braun
- Sony

What does Frank Gehry design?
- Buildings
- Computers
- Typefaces

submit

JavaScript Alert
Congratulations! You scored 3 out of 3
<doctype html>
<html>
<head>
  <title>Quiz</title>
  <link rel="stylesheet" href="quiz.css" />
  <script src="jquery-2.1.4.min.js"></script>
  <script src="quiz.js"></script>
  <script src="quiz-designers.js"></script>
  <script src="quiz-display.js"></script>
  <script>
    // add quiz to DOM
    $(function () {displayQuiz(designerQuiz, $('body'))});
  </script>
</head>
<body>
</body>
</html>
html {
  font-family: "Georgia", serif;
  font-size: 12pt;
}

h1.title {
  font-size: 16pt;
  font-style: normal;
}

label {
  font-family: inherit;
  display: block;
  margin-bottom: 6pt;
}

input[type="radio"] {
  margin-right: 12pt;
}

p.stem {
  font-weight: bold;
  margin-top: 18pt;
  margin-bottom: 6pt;
}

button {
  margin-top: 12pt;
  padding: 6pt;
  font-size: 12pt;
}
create an object using an ADT and return it
```javascript
var Quiz = function (title) {
    var that = Object.create(Quiz.prototype);
    var questions = [];
    that.add = function (stem, options, answer) {
        questions.push({stem: stem, options: options, answer: answer});
    }
    that.each = function (f) {
        questions.forEach(f);
    }
    that.grade = function (responses) {
        return questions.reduce(function (score, question, index) {
            return score + (responses[index] == question.answer ? 1 : 0);
        }, 0)
    }
    that.total = function () {
        return questions.length;
    }
    that.title = function () {return title;}
    Object.freeze(that);
    return that;
};
```
var displayQuiz = function (quiz, container) {
    var titleElt = $("<h1>", {class: "title", text: quiz.title()});
    container.append(titleElt);
    var responseFuncs = [];
    quiz.each(function (question, qindex){
        var questionElt = $("<div>", {class: "question"});
        var stemElt = $("<p>", {class: "stem", text: question.stem});
        questionElt.append(stemElt);
        question.options.forEach(function (option, oindex) {
            var radioElt = $("<input>", {type: "radio", name: qindex, val: oindex});
            var optionElt = $("<label>", {text: option});
            optionElt.prepend(radioElt);
            questionElt.append(optionElt);
        });
        responseFuncs.push(function () {
            return questionElt.find(":radio:checked").val();
        });
        container.append(questionElt);
    })
    var button = $("<button>", {text: "submit"});
    container.append(button);
    button.click(function () {
        var responses = responseFuncs.map (function (f) {return f();});
        var score = quiz.grade(responses);
        alert ("Congratulations! You scored " + score + " out of " + quiz.total());
    })
}

quiz-display.js

make elements & add to DOM

avoid search over all radio buttons!

note binding of oindex & qindex

make array of response functions

add listener to button
separation of concerns
But nothing is gained —on the contrary! — by tackling these various aspects simultaneously. It is what I sometimes have called “the separation of concerns”, which, even if not perfectly possible, is yet the only available technique for effective ordering of one’s thoughts that I know of. This is what I mean by “focussing one’s attention upon some aspect”: it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect’s point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously.

On the role of scientific thought
EWD447, August 30, 1974
activity

change which module to ...?
add a question
put question stem in italics
penalize skipping questions
replace alert with nicer widget
add help text that follows stem
allow questions with >1 answer?
module dependences
the ‘uses’ relation

We say of two programs A and B that A uses B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A uses B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Note that to decide whether A uses B or not, one must examine both the implementation and the specification of A.

Designing Software for Ease of Extension and Contraction (1979)

David Parnas
b. 1941
module dependences

weak dependence
module A names module B

strong dependence
module A names features of module B

dependence notion developed for 6.170 (c. 1984)

roughly speaking:
A uses B iff A depends on B

Barbara Liskov
b. 1939
activity

construct dependence graph
weak: mentions module name
strong: mentions features

consider 6 modules
including jquery library

weak
strong

jquery.js
quiz-display.js
quiz-designers.js
quiz.js
quiz.html
quiz.css
a subset of modules that can be reused and tested independently

Notes: quiz.html has weak deps on quiz.js and quiz.css because it only mentions the module names, and not the features inside the modules. The CSS classes defined in quiz.css are regarded as features of that module, hence the dependence on quiz-display.js on quiz.css.
trickier case #1

instead of passing responses, pass response functions:
quiz.js now calls functions “exported” by quiz-display.js

```javascript
var button = $('<button>', {text: "submit"});
container.append(button);
button.click(function () {
    var score = quiz.grade(responseFuncs);
    alert("Congratulations! You scored " + score + " out of " + quiz.total());
})

that.grade = function (responseFuncs) {
    return questions.reduce(
        function (score, question, index) {
            return score + (responseFuncs[index]() == question.answer ? 1 : 0);
        }, 0)
}
```

upshot: no change in dependences!
this is a class ‘inversion of control’
trickier case #2

instead of passing responses, quiz.js examines radio buttons itself

```javascript
that.grade = function () {
    return questions.reduce(
        function (score, question, index) {
            var score = quiz.grade();
            alert("Congratulations! You scored " + score + " out of " + quiz.total());
        }, 0)
}
```

upshot: quiz.js now depends on jquery.js (very bad!)
quizzes does NOT depend on quiz-display.js
... but quiz.js does USE quiz-display.js
quality is not (just) correctness

low level
use strong idioms (eg, *each*)
don’t repeat yourself (eg, *3 questions, constant “3”*)

high level
separation of concerns: where to put things
dependences: avoiding bad references

consequences
code is easier to understand, test, modify & extend
# mapping ideas to assignments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JS warmup</td>
<td>learning basic JS</td>
</tr>
<tr>
<td>prefix match</td>
<td>JavaScript idioms</td>
</tr>
<tr>
<td>game of life</td>
<td>dependences &amp; SOC</td>
</tr>
<tr>
<td>fritter</td>
<td>learning node &amp; mongo</td>
</tr>
<tr>
<td>data models</td>
<td>high level view of data</td>
</tr>
<tr>
<td>concepts</td>
<td>high level view of behavior</td>
</tr>
<tr>
<td>final project</td>
<td>applying it all!</td>
</tr>
</tbody>
</table>