Human-Computer Interaction Design

COGS120/CSE170 - "Intro. HCI"

Instructor: Philip Guo

Lab 1 - Version control and HTML (2016-09-29)

by Michael Bernstein, Scott Klemmer, and Philip Guo

[Announce on Piazza] Before coming to lab 1:

1. Create a GitHub account (if you don't have one)
2. Download Sublime Text 2 text editor for Windows/Mac https://www.sublimetext.com/2
3. Download Git for Windows/Mac https://git-scm.com/downloads

Wi-fi may be slow in the classroom, so please download and try to install everything beforehand. It's OK if you want to use your own text editor.

Why does this class have a lab?

For this class, coding is a means to an end (i.e., implementing your *design ideas*). Thus, we won't go deep into the computer science concepts behind any technologies. These labs are just a starting point. You'll probably need to learn more on your own.

Since this is not a coding class, you will *not* turn in labs for credit.

However, concepts from labs will be tested on Exam 1 and Exam 2.

These labs will help you make a personal portfolio web site

- **Never lose code**: source control
- **Get content on a page**: static HTML and CSS
- **Update that content dynamically**: node.js and Javascript
- **Analyze its use, make it mobile, make it look good**

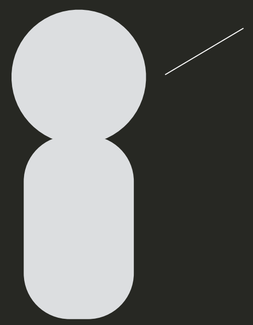
Let's try a new format for labs:

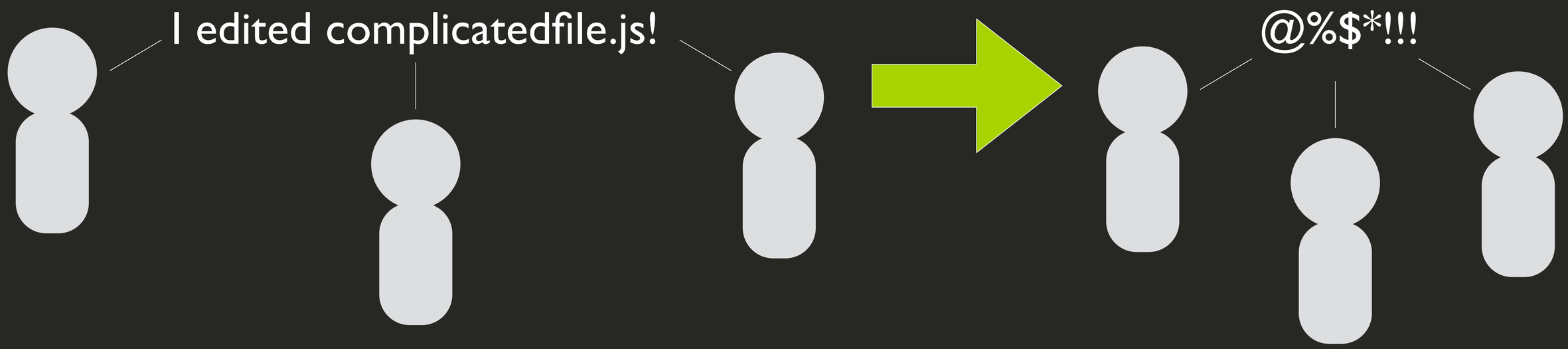
1. I'll give a mini-lecture on the slides for 20-30 minutes. You can follow along or jump ahead at your own pace. [room will be mostly silent so that everyone can hear clearly]

2. Then we will open it up for free-form lab work with TAs walking around to help. You can also come up to the podium to ask me for help too. [room will be louder]

Lab 1: Version control

Lab 1: Why use version control



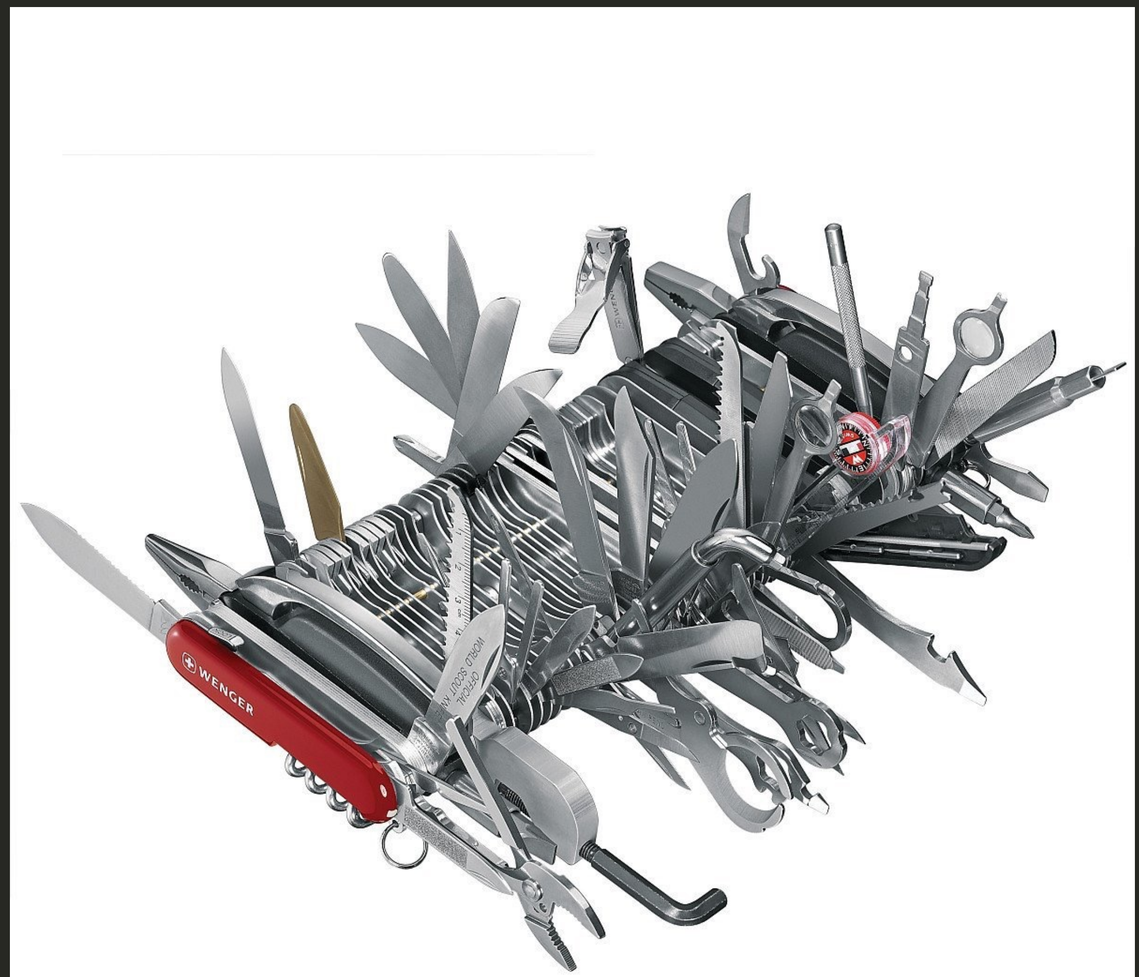


Lab 1: Why use version control

- Back up your code safely in the cloud: GitHub
- Make checkpoints and push to the cloud: git
- Deal with team edits: git conflicts
- Make a web page: static HTML

Lab 1: Version control

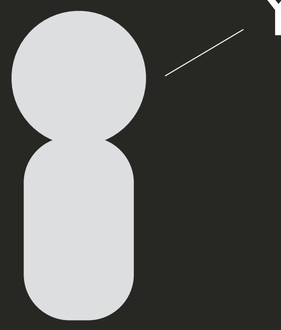
- We will be learning to use the Git version control system.



"Git is a knife whose handle is also a knife"

Why learn it? Because it's by far the most popular system in use today, especially in free and open source code. The GitHub website is a super convenient, free, and popular place to host your Git code repositories.

- Feel free to run ahead and aim for the lab stretch goal:
  - Turn the portfolio page scaffold into a fully functional web site with content from your previous projects. Must include images, descriptions, and multiple .html files.
- Even better: volunteer to help your classmates. Please be patient, though, since people come from all sorts of prior programming backgrounds.



- All speeds are welcome here. We really want everyone to get up to speed. Lab time (and office hours) are for you!

- Try first: ask your friends, help your friends
- Try second: raise your hand to summon a TA in lab
- Try third: come to our office hours for more help

Installing Git

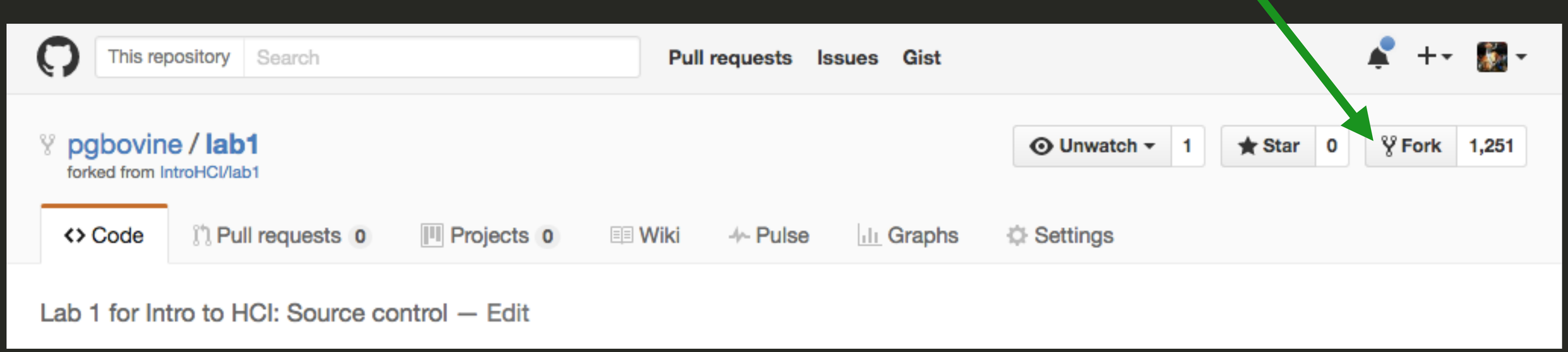
- On Windows, make sure you can run Git Bash after you install Git. Find it in the start menu or desktop.

- On Mac OS, you may be prompted to install the Xcode development environment first, which can be annoying. Bypass that if you don't want to download gigabytes of Xcode package to use Git. TAs can help. Google online for "git mac os x without xcode" to see resources such as:

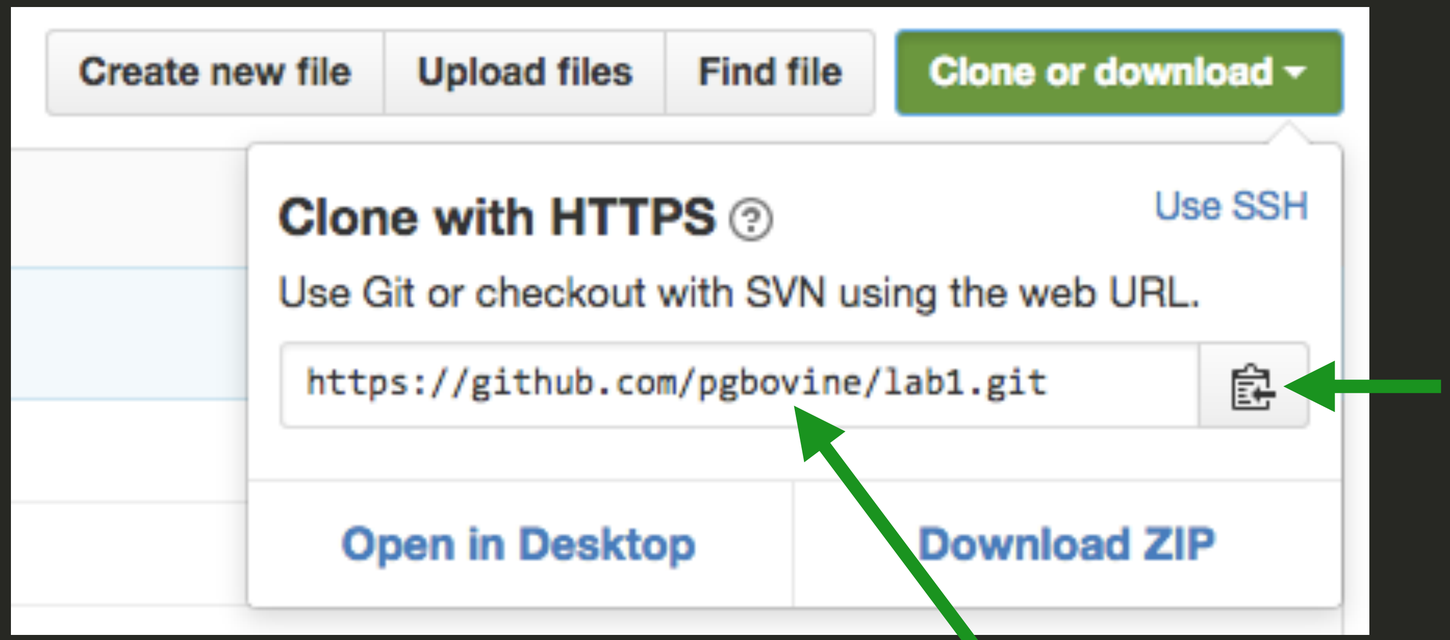
- http://blog.bobbyallen.me/2014/03/07/how-to-install-git-without-having-to-install-xcode-on-macosx/

Fork your first repository

- Find Lab 1 at: `https://github.com/pgbovine/lab1.git`
- Forking will *copy* the repository to your GitHub account



Copy your own git URL to the clipboard



Use https. Your Clone URL should look like this:
https://github.com/<YOUR GitHub USERNAME>/lab1.git

not this … https://github.com/pgbovine/lab1.git
(since this is my username)
Leave this browser tab open!

Open your terminal

- Mac OS X
  - Applications → Utilities → Terminal
  - Or search “Terminal” in Spotlight
- Windows: Git Bash
  - Start Menu → Git → Git Bash
- Create a directory named “introHCI”
  - Using the command-line: `mkdir introHCI`
- Change your directory to introHCI in the terminal:
  - `cd introHCI`
- Typing the `pwd` command should show `introHCI`

The parts of the command-line interface in your terminal application:

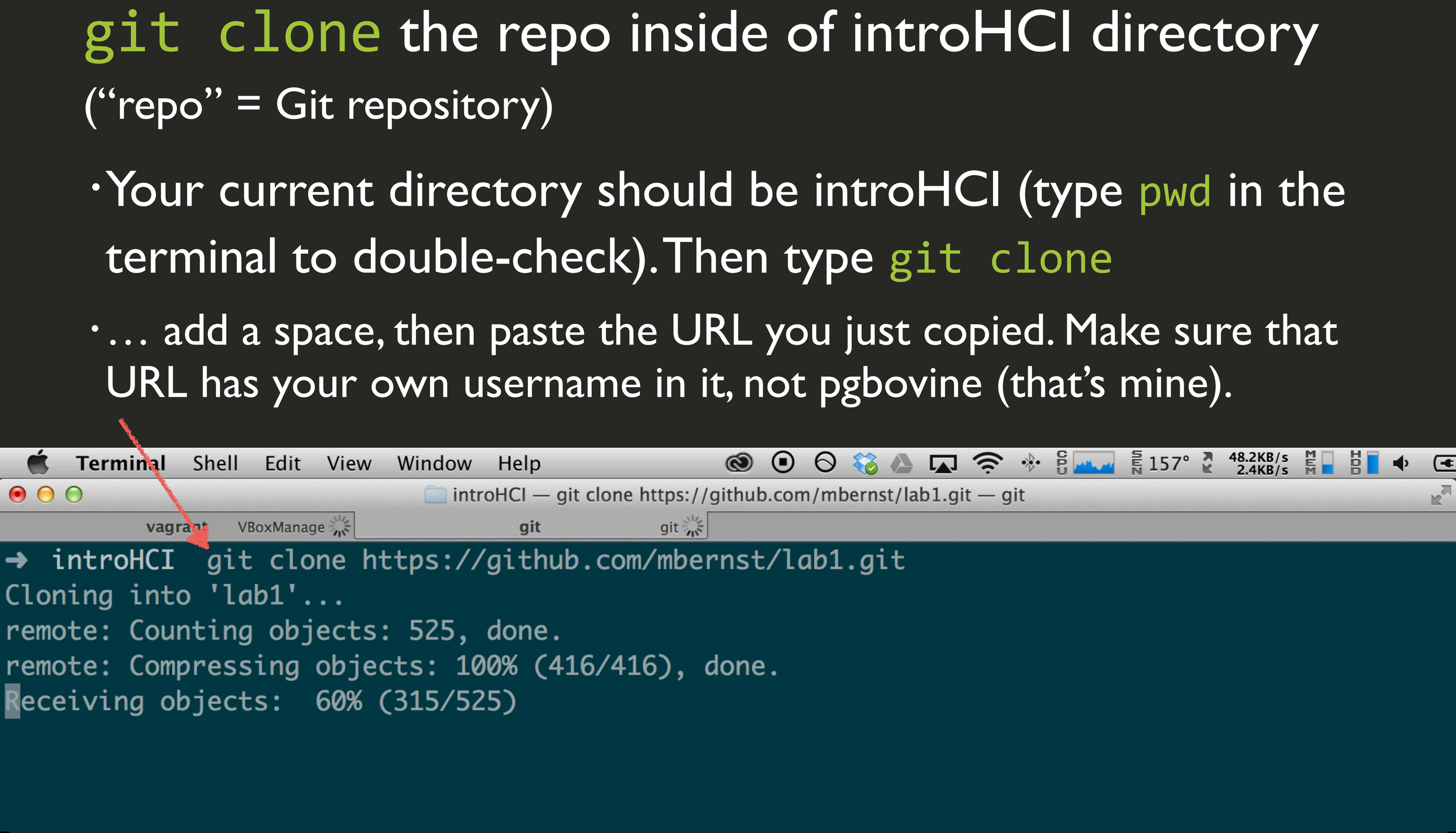
username:~/Documents/introHCI$ cd lab1

- Important terminal commands
  - `cd directoryname`: change directory to `directoryname`
  - `ls`: list all files and directories in the current directory
  - `pwd`: which directory am I in?

`git clone` the repo inside of introHCI directory

("repo" = Git repository)

- Your current directory should be introHCI (type `pwd` in the terminal to double-check). Then type `git clone`
- … add a space, then paste the URL you just copied. Make sure that URL has your own username in it, not pgbovine (that's mine).



Windows: how do I paste into Git Bash?

- Click on the terminal icon in the upper left of the window: Edit, Paste. For a shortcut: Alt-spacebar, then e, then p. If that seems needlessly complicated, you can also enable QuickEdit mode by clicking on the terminal icon in the upper left of the window: Defaults(or Properties). Go to the Options tab and enable "QuickEdit Mode". Now you may paste by right-clicking in the terminal.

Open your text editor

- Mac OS X
  - Applications → Sublime Text 2
  - Or search “Sublime Text 2” in Spotlight
- Windows
  - Start Menu → Sublime Text 2

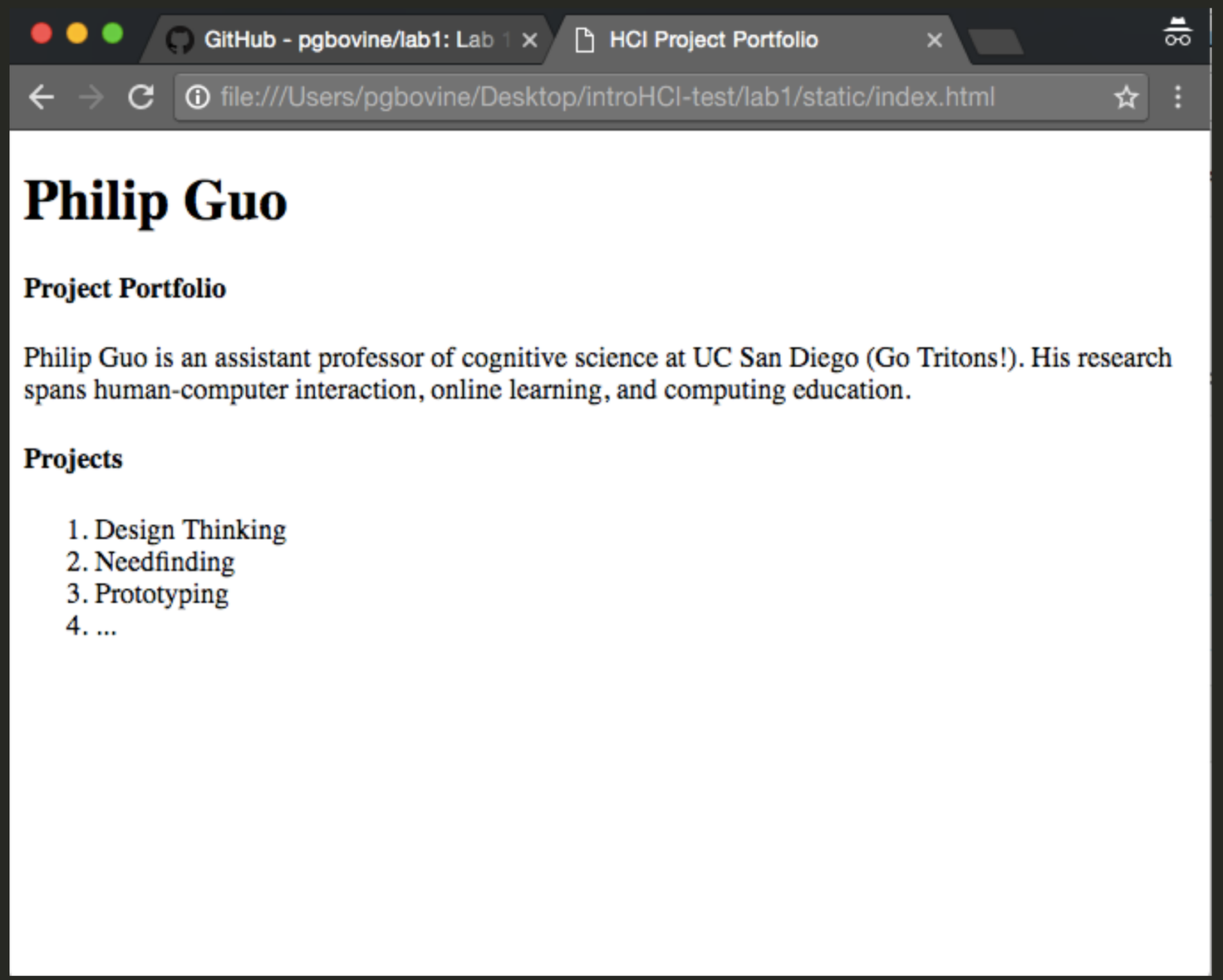
Add your info into lab1/static/index.html

- This HTML will be rendered by the server

index.html

```
<!doctype html>

<html>
<head>
    <meta charset="utf-8">
    <title>HCI Project Portfolio</title>
</head>

<body>
    <!-- this is a comment in HTML -->
    <h1>Scott Klemmer</h1> <!-- h1 through h5 are headers. The higher the number,
    the smaller the header -->
    <h4>Project Portfolio</h4>

    <!-- p means paragraph -->
    <p>Scott is an associate professor of Cognitive Science and Computer Science &
    Engineering at UC San Diego.</p>

    <h4>Projects</h4>
    <ol>
        <li>Waiting in line</li>
        <li>Needfinding</li>
        <li>Prototyping</li>
        <li>...</li>
    </ol>
```

Browse to index.html using the file on your hard drive, open it from browser



Lab 1 stretch goal reminder

- Turn the portfolio page scaffold into a full web site and push it to GitHub
- It must contain:
  - Content from your previous projects
  - Images and descriptions
  - Multiple .html files linked together via hyperlinks
- Even better: add styling

Set up Git so we can commit with sensible messages. Type in these commands in your terminal …

```
git config --global user.name <YOUR NAME IN QUOTES>
git config --global user.email <YOUR EMAIL IN QUOTES>
git config --global push.default simple
```

For example, for Michael …

```
➜  lab1 git:(master) git config --global user.name "Michael Bernstein"
➜  lab1 git:(master) git config --global user.email "msb@cs.stanford.edu"
➜  lab1 git:(master) git config --global push.default simple
➜  lab1 git:(master)
```

Version control with Git

Git server
(e.g., GitHub)

Your local client
(your own computer)

`git status` and `git add`

```
➜  lab1 git:(master) ✗ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   static/index.html
#
no changes added to commit (use "git add" and/or "git commit -a")
➜  lab1 git:(master) ✗ git add static/index.html
➜  lab1 git:(master) ✗
```

- `git status`

  git tells you which files would get committed

- `git add filename`

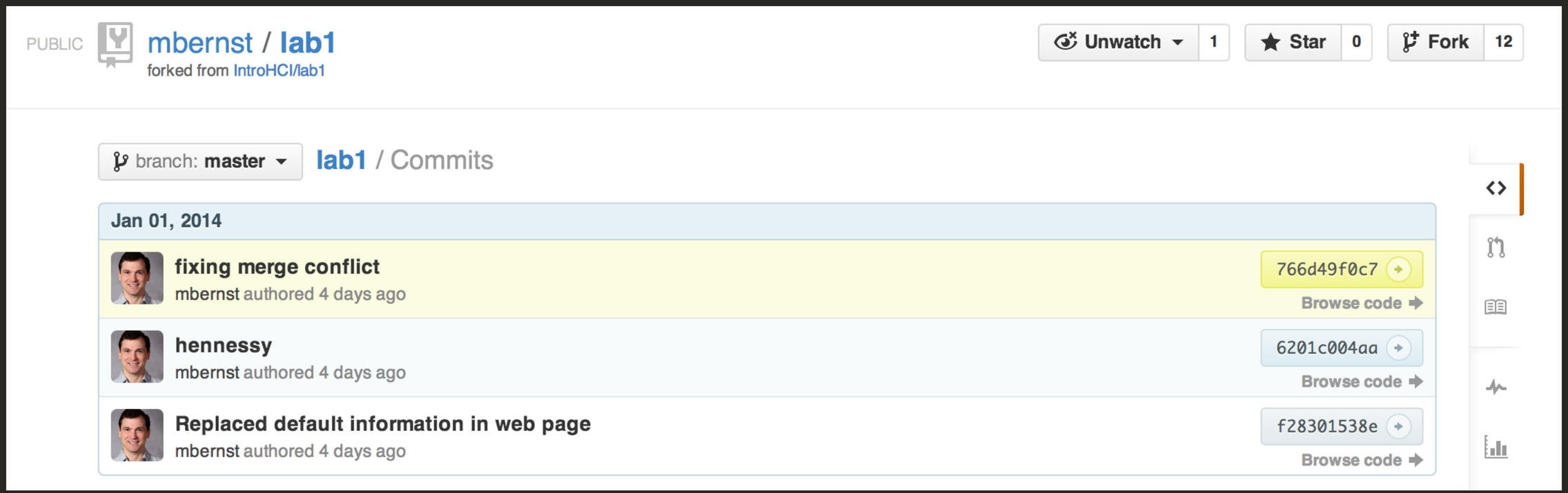
  Tells git that a file should be included in the commit

Commit, pull and then push

```
➜  lab1 git:(master) ✗ git commit -m "Replaced default information in web page"
[master 82c891f] Replaced default information in web page
 1 file changed, 2 insertions(+), 2 deletions(-)
➜  lab1 git:(master) git pull
Already up-to-date.
➜  lab1 git:(master) git push
```

- `git commit -m “Commit message”` performs a local commit.
  It is *not* pushed to the server. Do this often: it’s a save point.
- `git pull` pulls (brings) in any new changes from the server
- `git push` pushes (backs up) all local commits to the git server

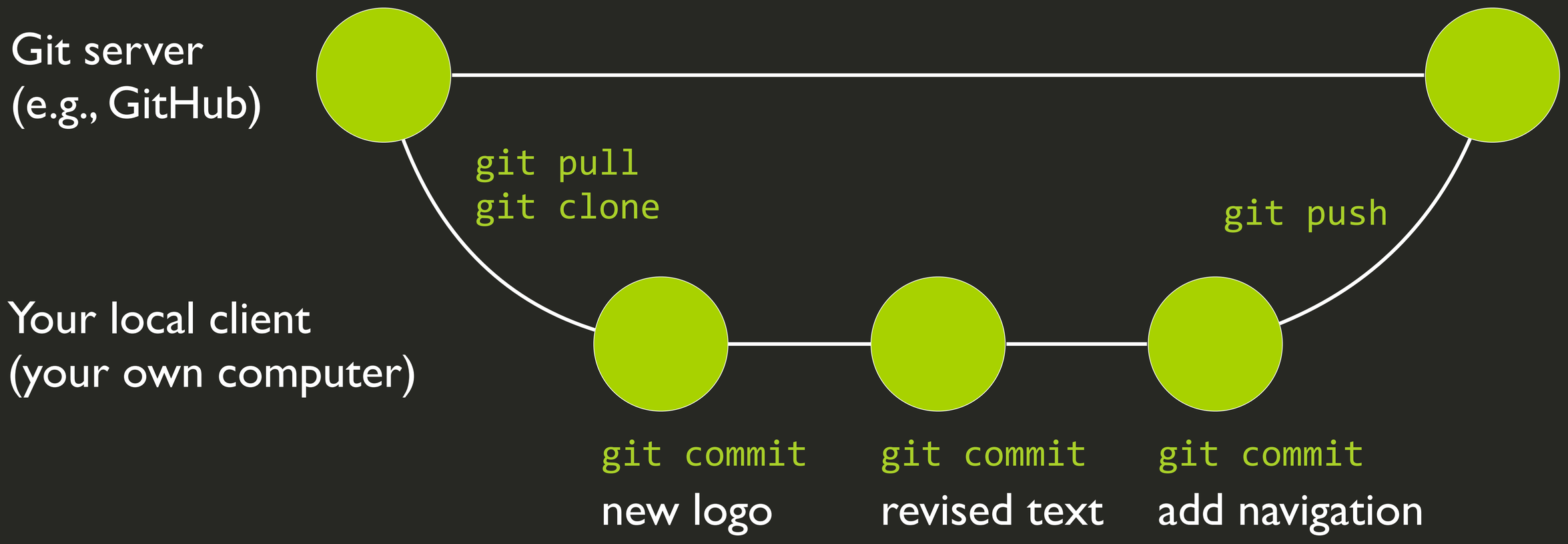
GitHub website reflects your commit



Other basic git commands

- `git commit -a -m "Commit message"` performs a local commit of all files that have changed (instead of `git add` each of them)
- `git log` shows your history of commits
- `git diff` shows what changed but has not been committed

So far: one-person version control



Question: What's the difference between Git and GitHub?

EVERYTHING AFTER THIS IS OPTIONAL (not on exam)

Multi-person version control

Git server

Your partner

Your local client


```
git pull
```

Merging and conflicts

- Please wait while we add a conflicting change to the upstream repository

- [we will manually change the contents of index.html so that it hopefully conflicts with yours]

Merging and conflicts

- Add the upstream repository as a remote server and merge it in to discover the conflict [this isn't the most realistic scenario, but it's a quick way to show conflicts]
- git remote add upstream https://github.com/pgbovine/lab1.git
- git pull upstream master

```
➜  lab1 git:(master) git remote add upstream https://github.com/IntroHCI/lab1.git
➜  lab1 git:(master) git pull upstream master
```

Resolving merge conflicts

```
<!doctype html>

<html>
<head>
    <meta charset="utf-8">
    <title>HCI Project Portfolio</title>
</head>

<body>
    <!-- this is a comment in HTML -->
<<<<<<< HEAD
    <h1>Scott Klemmer</h1> <!-- h1 through h5 are headers. The higher the number,
    the smaller the header -->
    <h4>Project Portfolio</h4>

    <!-- p means paragraph -->
    <p>Scott is an associate professor of Cognitive Science and Computer Science &
    Engineering at UC San Diego.</p>
=======
    <h1>John Hennessy</h1> <!-- h1 through h5 are headers. The higher the number,
    the smaller the header -->
    <h4>Project Portfolio</h4>

    <!-- p means paragraph -->
    <p>John Hennessy is president of Stanford University. He takes HCI classes in
    his spare time.</p>
>>>>>>> c17fc57096f3daa4ed0566aab987670de6a874b6

    <h4>Projects</h4>
    <ol>
        <li>Waiting in line</li>
```

Line 11, Column 14; Undo: Insert Characters Tab Size: 4 HTML

Resolving merge conflicts

- Push the merged version back to the repository with `git add`, `git commit`, `git pull` and `git push`

```
➜  lab1 git:(master) ✗ git add static/index.html
➜  lab1 git:(master) ✗ git commit -m "fixing merge conflict"
[master 1180cca] fixing merge conflict
➜  lab1 git:(master) git pull
Already up-to-date.
➜  lab1 git:(master) git push
```

Parting advice: avoid merge conflicts by coordinating with your teammates.

- It's OK to both work on the same file at once, but make sure you're not working "near" each other in the same file. Coordinate manually.

- The advantage of something like Google Docs is that you get real-time sync and multiple cursors so that you won't have merge conflicts.