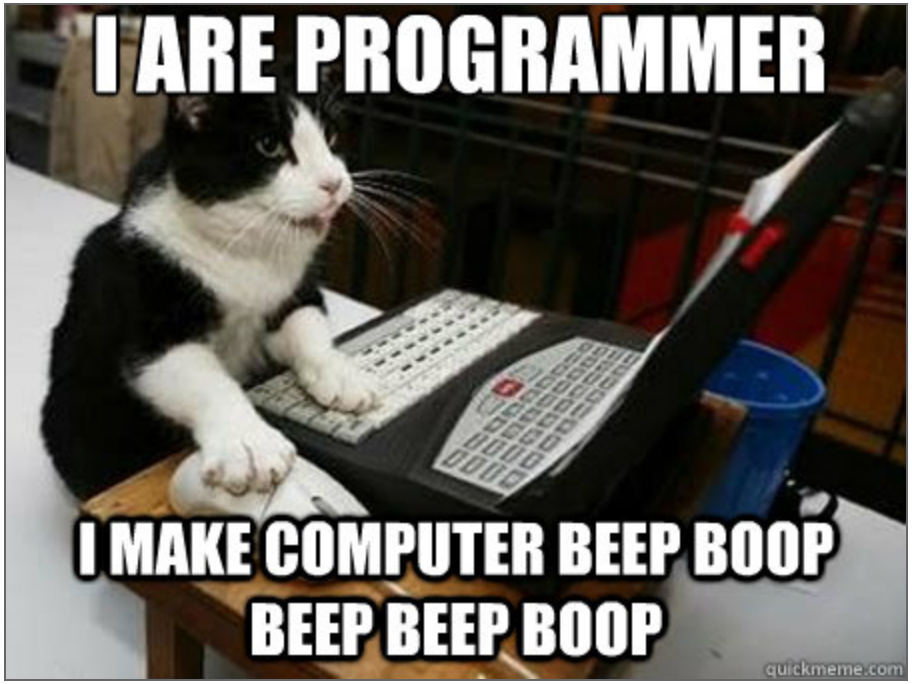
Welcome to INFO6250

Web Development Tools and Methods



Quick Questions

- Is this an online class?
    - No
- Is in-person attendance required
    - Yes (except for excused absences)
    - Not my decision
    - Let me know if sick or other excuse
        - I'm nice, we're all adults

Who is this guy?

Brett Ritter `<b.ritter@northeastern.edu>`

- He/Him
- Currently in Seattle
- WebDev since 1995
- Multiple languages, frameworks, platforms
- Both frontend and backend
- Part time instructor since 2017
    - Tell me where to improve my teaching

Not Perfect

I have a truly terrible memory.

Terrible

You have my permission to remind me, and keep reminding me, until something is done or I explicitly say "stop".

Funny

I tell jokes

Fortunately, they are all hilarious and you will laugh

Out loud

Try it now

Get Better

We will keep practicing on that

COVID Overview

- COVID sucks
- My rules are the same even without COVID
    - Being sick and spreading ANY illness is bad
- Thank you for keeping yourselves and others safe
- I will assist with this whenever possible

Vulnerabilities

- I have multiple comorbidities
- My wife takes medication that reduces her immune system
- Expect due caution from myself

Accommodations

- This course is hard, very busy
- Require/benefit from accommodations?
    - Let me know

Falling Behind

- This Course has a lot of work
- Intended to build mental pathways
- Needs time to do, time to "set"
- Falling behind is BAD
    - Hard to catch up!
    - Not just a matter of dedication
- Let me know ASAP if it happens

What does "Tools and Methods" mean?

Goal: Give a foundation in practical Web Development

- How to break down a problem
- How to code
    - with communication
- How to debug

We will USE languages/frameworks

- But we LEARN more general concepts

Full Stack Web Development

- Full Stack, not only backend
- NodeJS used, not Java
- Boston: Other 6250 courses
    - Java-focused
    - More backend-focused
- Default Banner/Canvas description may mislead

Web Development Tools and Methods

- The Hows and Whys of the Web
- Building a Multiple Page Web Application
    - NodeJS backend
    - HTML
    - CSS
    - JS frontend
- Building REST-based services (NodeJS backend)
- Building a Single Page Application
    - Plain JS and with React
- Basic Best Practices for the Web

But I'm not a JS Dev!

This course also for Java/C#/Python/Ruby/etc devs

- Languages/Frameworks have same result
- Languages/Frameworks regularly change
- Front end remains essentially JS-only
    - Defeated all comers so far
    - Plan to avoid JS is risky

What about the Lab class?

- Reflects the additional time for the assignments
  - Seriously, set aside a lot of time!
  - I hate homework too
  - But you learn by doing
- I am not teaching during the lab time

Class is hard, worthwhile

- We cover a lot of material
- You must learn to apply new concepts
- A lot of time and work
- Goal is maximum preparation

Cannot teach it all

Too much to cover

- You end empowered to continue your education

Teaching Fundamentals

Pros:

- What you build from
- Fewer "surprise" gaps

Cons:

- Longer path to "wow!"
- Very rushed class

Ask Questions

Everyone learns differently

I have terrible memory

Ask Questions

Trying things

Best answers can be found by trying things

I set you up to *experiment*

- Don't cut-and-paste!
- Don't copy code!
- Don't use ChatGPT code (etc)

These are **losing strategies** in the long-term

- Learn HOW/WHY
- Then write yourself
    - From nothing!

It Depends

Most common answer: 'It Depends'

IS NOT: Does not matter

IS: Depends on what?

MIGHT BE: We are still figuring that out

Assignments

Assignments will NOT be "copy what I did"

- Instead, "apply the skills in a new way"

Leave time to do work!

Assignments are made to build skills

- Do not copy!
- Do not follow out of class tutorials!
- Do not ask a chat bot
- Goal is not "works", goal is SKILL

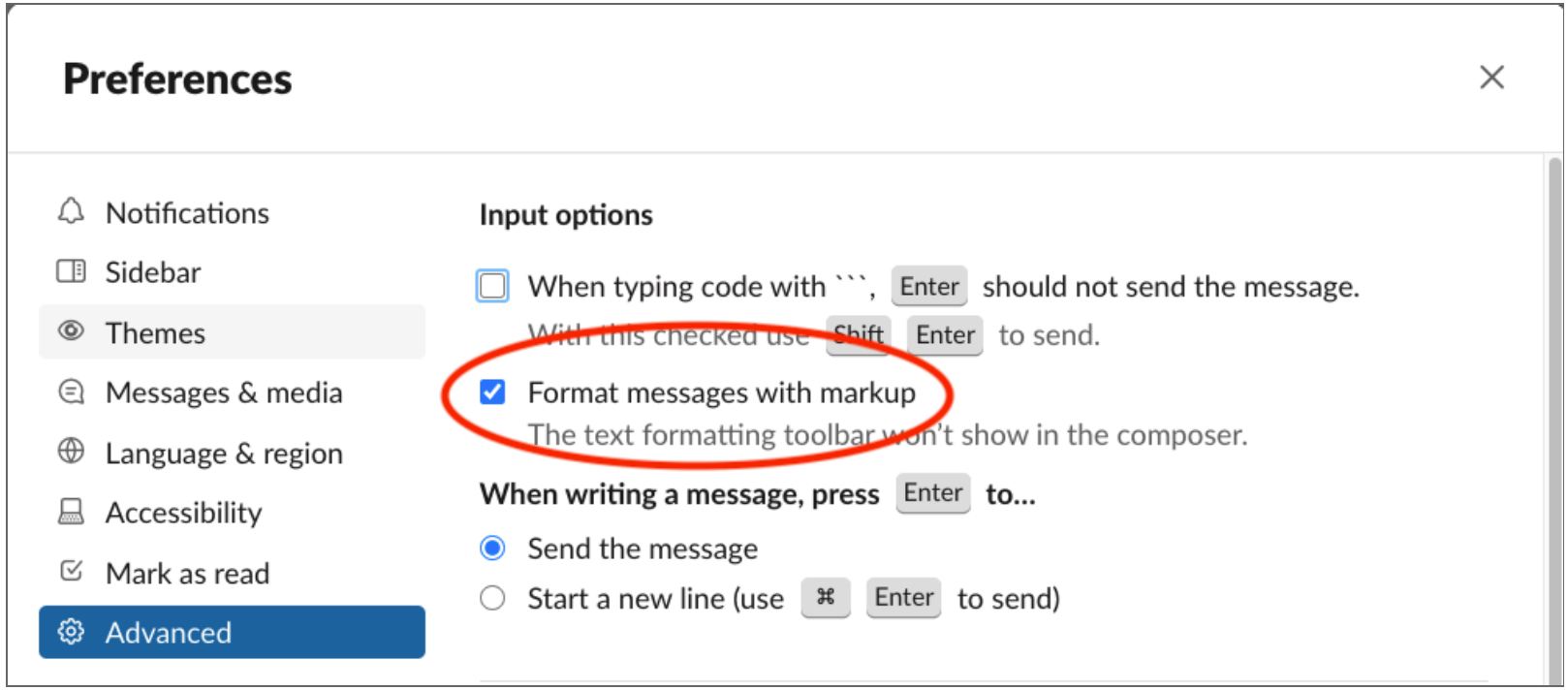
Slack

We communicate via Slack (not Teams)

- **https://rebrand.ly/svinfo6250-slack**
- You can email me, BUT
    - slower
    - terrible to talk about code
- Use it web, desktop, or phone
    - Notifications recommended!
- Slack is a useful job skill
    - Managing many messages/channels
    - Searching for past messages
    - Reminders

Configuring Slack for code

Update your Slack Preferences:



Mentioning code in Slack

When sending messages in Slack:

(Notice the *backtick* characters)

`this has *bold*` but `` `this does *not*` ``

Use backticks for a line with code in it.

Code blocks in Slack

Triple backticks ``` before and after your message - multiple lines

Text often better than screenshots

- Can copy and run!
- Can grab individual lines!

If using screenshots

- Use built in screenshot abilities!
- No need for screen selfies!

These are job skills!

Slack is a Job Skill

- You know how to use Chat apps, but…
    - Most coders will be in 10-20 **channels**
    - Search to find previous answers
- You need to tweak your notifications
    - I announce changes to assignments/classes!
    - Find sounds that inform without disruption
- You need to find information posted in the past
- Will need to **join** channels of interest
    - Such as **#articles** or **#funny**

Class Github

We use git and github.com

Each of you get a personal repository

- **https://rebrand.ly/svinfo6250-github**

You must have/get a github.com account

- Students new to coding may be unfamiliar
- git is a vital job skill for any programmer
- Devs will encounter github often
    - Even if not at their job

Git vs Github

`git` is the source control/version control system.

- tracks files
- changes to files
- many devs can have repos
- can pass files/changes between many repos

`github.com` provides a central place for repos. It has competitors (example: gitlab.com)

github uses git, git does not require github or a github competitor, though we will use github.

Github flow

See `readings/git/` in your repo

- Make edits in **feature branch based off of main**
- You **push** (send) that branch to github
- You create a **Pull Request (PR)** to **merge** your branch into `main` **on github**
- I/TA **review** and **approve** your request
    - We might request changes first
- I/TA merge your branch into `main` on github
    - On the job you will probably do this step
- You update your local `main` branch

Other git-based flows

Other flows of changes and branches exist

- This one (github flow) is the one we will use

Not all version control systems (VCS) are decentralized the way git is

Github acts as a central point for communications

- Your future job uses Github or similar

Local and Remotes

Local means your computer

When I give notes or assignments:

- I **pull** latest `main` from github to my local copy
- I update my local copy (**add**ing and **commit**ting)
- I **push** my `main` branch to github copy of your repo

You submit the same way:

- You **pull** changes from github to your local copy
- You make changes in a **feature branch**
- You **push** feature branch to github and create **PR**

Key Git Notes

- Always do work in the correct branch
- Always check `git status` before `git commit`
- Always check `git status` before `git push`
- When creating a PR, always check the file list
- Before creating a new feature branch
    - Always switch to `main` and pull latest

If you follow these instructions

- Each assignment is distinct and will not conflict

You will see me use the Command Line

I use the "command line" (terminal) a lot

- This may be all new to you
- This is an important programming skill
- Command Line Interfaces (CLI) are a powerful tool
    - Will see often in documentation/tutorials
- I don't *teach* the command line itself
    - Just the specific commands I use
    - I recommend you learn in more depth
- See `readings/the-command-line.md` for more info

How the Class works

- Assignment each class (15% total)
- Quiz each class (10% total)

3 Projects (25% each)

- Server-side Project
- Vanilla JS + REST Services Project
- Final Project

Common Grading questions

- No curve
- 90% is A-
- 93% and higher is A

How a Class works

- Lectures, sometimes labs (ungraded)
- Bio break 1/class (10 minute)
- Recorded online for review
    - University requires attendance!
- Slides as PDF added to repos
    - Usually before class
- Sometimes samples to repos
- Assignment due night before next class
    - Via github
- Canvas Quiz due night before next class
    - Open notes

How Quizzes work

In Canvas

- Multiple choice
- Covers materials from the class lecture
- NOT timed!
- You are welcome to consult notes, recordings, etc
    - Do NOT copy from other people
    - Do NOT google answers (Can mislead!)
- Due night before next class

Worst Quiz score is ignored for final grade

How Assignments work

- Build from skills shown in class
    - Leave time!
- Due night before next class
- Added to repos under `/work` (see README)
    - Each assignment is a different subdirectory
    - Remember: **branch** NOT the same as **folder**
- Submitted via github **Pull Request**
- TA/I will review and merge
    - May request changes
- No changes unless requested

Worst Assignment score is ignored for final grade

How Projects work

- Like Assignments
    - Pull request, Due date
    - Done at home, not in class
- In repos under `/project1`, `/project2`, `/final`
    - NOT `/work`
- Big chunk of grade each!
    - VERY important to do well
    - DO NOT copy work
        - "referencing" may be copying
    - "Demonstrate Skills from class"
- Minimal outside libraries

How the Final Project works

- Guidelines given
- Full React SPA + REST services
    - Minimal outside libraries
- Potential Showcase for NEU
- Submitted as Pull Request in repo
    - in `/final`
- Limited time! No extensions!
- Chance to raise grade
    - "Demonstration of skills from class"

Instructor Virtual Office Hours

- Mon: 2pm-3pm (ET) / 11am-noon (PT)
- Tue: 2pm-3pm (ET) / 11am-noon (PT)
- Wed: No Office Hours
- Thu: 2pm-3pm (ET) / 11am-noon (PT)
- Fri: 2pm-3pm (ET) / 11am-noon (PT)
- Other times by appointment
- Available on Slack for quick questions

Other Details

- No video requirement
- No breakout rooms
- Canvas for grades and quizzes only
- Slack preferred over Email
- TA Office Hours TBD

Important Details

- These are not normal times, I understand
- Request Assignment extensions
    - No excuse required!
    - But request at least 1 day IN ADVANCE!
        - Job skill!
- DO NOT COPY WORK YOU SUBMIT
- Demonstrate skills from this course
    - Not just working code

Final Projects: NO extensions

- Except unquestionable emergencies

DO NOT COPY WORK

- I prefer learning to grades
    - But grades should be fair
- Most learning is practice
    - Finding little lessons
- Copying reduces practice
    - Whatever you call it ("referencing")
- NOT WORTH THE RISK
    - Use my generous extension policies
- See "do-not-copy-work" in your repository

Large Language Models (ChatGPT etc)

- Don't want to be old/out-of-touch
    - But want to teach
- LLMs are NOT "AI"
    - No "understanding"
    - Just predictive text
- Might be helpful on job (maybe)
    - NOT helpful to learn!
    - Like copying, cuts practice
    - Too often it is WRONG
    - You lack context to know

Do and Do Not

- DO ask questions
- DO not worry about bothering me
    - I will tell you BEFORE that happens
- DO NOT worry about looking uninformed
    - You are literally students
- DO NOT expect to catch up by working harder
    - Good attitude, but…
    - Time, not you, is the problem
    - Easier to fix earlier rather than later
    - Your employer will follow the same rules

A Unique Warning about the Web

You WILL be expected to learn a lot of detail online

BUT a **lot** of info about web tech is outdated

- Don't use any sources older than 3 years ago

Really. **3 years max**. Or extra work and wrong work.

Hint:

- One reason ChatGPT often gives poor advice
- "Works" is not the same as "Good"

Common Questions

- Do I need to know HTML/CSS?
- Do I need to know programming?
- Do I need to know JS?
- Can I use another language?
- Can I use this outside library?
- Can I use this other framework?
- Can I use this other IDE?
- Can the instructor send class materials out?

Class Prerequisites

- Do I need to know HTML/CSS?
  - You are expected to know the basics
  - The 6150 class is great and recommended
  - You CAN learn alongside class (but **harder**)
  - **https://developer.mozilla.org/en-US/docs/Learn**

Class Prerequisites (cont)

- Do I need to know programming?
    - Basics (variables, conditionals, looping, functions) are expected
    - Be ready to learn how things are different
- Do I need to know JS (Javascript)?
    - No expectation

Using other tools

- Can I use another language?
    - No, we use only JS for my sanity
    - Lessons are general!
- Can I use Typescript?
    - No, we rely on JS fundamentals
    - TS is fine, just not part of the class

Using other tools (cont)

- Can I use this outside library?
    - (example: Bootstrap)
    - No, we need to exercise the fundamentals
    - Exceptions are explicit per assignment
- Can I use this other framework?
    - (example: Angular/Vue)
    - No, we use React for core principles
        - and my sanity
- Can I use this other IDE?
    - Yes, but I don't know that IDE

Class Materials

Can the instructor send class materials out?

Slides and code samples will be added to your repos

- `git checkout main`
- `git pull origin main`

If you feel something is missing, ask for it via Slack

Anything in the moment MIGHT NOT be saved/sent

- Ask questions during class!

What to expect

Early in Semester:

- Lots of fundamental concepts
- Rushing HTML/CSS/JS
- Webserver concepts
- Client side JS

Later in Semester:

- Network and services
- Asynchronous code
- React Fundamentals

Core Lessons From This Course

- Programming is Communication
- "Working" isn't the same as "Good"
- Complexity is the Enemy

Why?

- Majority of dev time spent changing existing code
- Dev time is expensive!
- "Good" code can be understood quickly
    - Enough to make changes that work
- Complexity makes it hard to make changes

Summary - You

- Will **remind me** if you need something
- Can and will **use and check** our Slack
- Have your own **github repo**
- Have a **local copy** of the repo
- Will **install** and **configure** software per repo
- Will **submit a PR** that provides your info
- Will **not use old** online information
- Will **not copy** work
- Will **ask questions**