Welcome to INFO6150

INFO6150: Web UX Engineering (UI/UX)

Quick Questions

- What is the attendance policy?
- I'm new to programming, is that a problem?
- Can I get an A in this class, for real?
- What is the extension policy?
- What is the late work policy?

Attendance

Attendance is required, except for excused absences

- Not my decision
- University can be SERIOUS about this!
- Let me know if sick or other excuse
    - In advance of class!
    - I'm nice, we're all adults

Previous Experience

This course typically has both:

- People new to coding entirely
    - Often switching fields
- People with professional web dev experience

I try to help both groups

- Please be attentive and patient
    - Particularly in first few weeks

Can I really get an A?

- No curve, anyone CAN get an A
    - Many do
    - "A" must be earned
- I focus on actually teaching
    - Not a good class to "just have" while applying/interviewing
    - ChatGPT answers heavily penalized!
        - Interviews won't accept either
- Start simple, steep learning curve

Extensions

I am reasonably generous with extensions

- Real Life happens
- No excuses needed
- No penalty for granted extensions
- Builds Real Job Skills

BUT!

- **Ask BEFORE day assignment is due**
    - Unpredictable problems are fine
- Don't overuse
- On-time grading gets priority before extensions

Late Work Policy

Work turned in late defaults to a 0

- *IF* time permits, gets graded
- Minimum 10% penalty
- Ask for extensions!
    - Before the day they are due

What was that about ChatGPT?

- LLM generated work (ChatGPT, Claude, etc)
    - Horrible for learning
- "Great if you verify it"
    - **When learning, you *cannot* verify**
        - "it looks okay" is not verifying
- If your work isn't something you understand
    - Heavy to complete grade penalty
    - Interviews will expect you can explain
- This class builds on itself
    - You must understand each lesson

Who is this guy?

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

- He/Him
- Currently in Seattle
- WebDev since 1995 (not designer)
- 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

You will laugh out loud

Try it now

Get Better at Laughing

We will keep practicing that

What is UX Engineering?

- Newer term
    - Still being defined in practice
- Front End Web Developer is vague
    - More Coding?
    - More Design?
- "Designer" doesn't always mean ANY web skills
- Front End Web Dev + Designer = UX Engineering
    - NOT the skill sets of both
    - The blurry area in-between the two

Will you have a Designer?

General situation:

- Smaller shops (coding workplaces) = no designer
- Larger shops will have designers

Always better off with a good designer

- But often have to "make do" without

Limitation: I am not a designer

- I do not have design skills or experience
- I DO have experience working with designers
    - On real teams in different industries
- I will teach you how to work with designers
    - What to consider when you don't have one
    - What designers often overlook
- How to create different web interactions (UI)
    - How to decide if **helpful** interaction (UX)
- NEU offers separate design courses

The Web is Awesome

- Literally changed the world
- Socializing, Dating, Reference, Commerce
- Based in programming
    - Zero copy cost!
        - What other industry has that?
    - Build tools to save repeat effort!
        - Benefit from tools others write!

Coding Web UI/UX is Awesome

- Every website requires discovering how it works
  - Humans are naturally curious
  - Humans are also impatient
- Learning good UX is learning how humans work
  - Creating good UX is immensely rewarding
- Web UI/UX makes good functionality FEEL good
  - Power + satisfaction
- EVERYONE can run your code!
  - Does not require installation
  - Does not care about Operating System
  - Does not care about mobile/desktop

Web UI/UX evolves quickly!

- https://www.webdesignmuseum.org/gallery/apple-1996
- https://www.webdesignmuseum.org/gallery/twitter-2006
- https://www.webdesignmuseum.org/gallery/netflix-2002

Not all UI is fancy, not all is new

- **https://sans.style/**
- **https://en.wikipedia.org/wiki/Cat**

What does INFO6150 teach?

- How to implement UI/UX decisions
  - Using HTML, CSS, JS
  - Native "vanilla" JS
  - Intro to React
- Important needs in UI/UX
  - Accessibility (a11y)
  - Semantics
  - Maintainability
  - Avoiding Deceptive Patterns
  - Basic Principles

What does INFO 6150 NOT teach

- Any Deep Design Concepts
    - I cover more HOW than WHY
    - Taught from Web Dev perspective
- Server-side/Backend logic
    - See INFO 6250 for this
- Advanced Front End logic
    - See *my* INFO 6250 (full stack) for this
    - Note: other INFO 6250 backend only!
- Using Libraries (mostly)
    - Libraries are great!
    - But we are training *your* skills

What to expect

- Git
- Lots of HTML/CSS
- Focus on best practices
    - NOT "it works"
    - NOT "looks right"
- Many syntaxes
    - HTML
    - CSS
    - JS
    - React

How this course flows

- Weekly lectures
- Weekly assignments
    - Submitted via Github
    - Due night before next class
- Weekly Quizzes
    - On Canvas
    - Open Notes
    - Not time-limited
    - Due night before next class
- Projects (solo)
    - Like big assignments

How the Course is Graded

- Quizzes (10% of grade, lowest dropped)
- Assignments (15% of grade, lowest dropped)
- Projects (each 25% of grade)

Final Project can raise low grades!

Virtual Office Hours

Instructor:

- Mon 2pm-3pm (ET) / 11am-noon (PT)
- Tue 2pm-3pm (ET) / 11am-noon (PT)
- Wed (no office hours)
- Thu (no office hours)
- Fri 2pm-3pm (ET) / 11am-noon (PT)
- Other times by appointment
- Available on Slack for quick questions

TA Office Hours: TBD

Teaching Assistant(s)

- Have taken this course!
    - Use their knowledge and experience
- Virtual Office Hours
    - To be announced

Core Class Concepts

The Concepts you'll hear from me repeatedly

1. Programming is Communication
2. More Changes than is New
3. Accessibility is Functionality
4. Complexity is the Enemy
5. Working is Not Enough
6. Avoiding "tutorial hell"

Programming is Communication

All Programmers tell the computer what to do

Most Programmers get the right result

Good Programmers have solutions that are:

- Understood
- Reliable
- Easy to change (successfully)

Most of Programming isn't computer code

- **Programming is Communication**

More Changes than is New

- School assignments usually all new work
- Real tasks almost always changes to existing
    - Rarely "new" work
- Devs won't read every existing line

Work must make it easy to:

- Find relevant spot
- Understand the concepts
- Make changes using those concepts

Accessibility is Functionality

- Accessibility = Content available to all
    - "ability to access"
- Web successful because content is accessible
    - Not just to humans
    - Humans are important, though!
- Web predates smartphones, smart watches, IoT
    - But still works on them
- Improving Accessibility
    - Improves Functionality

"It looks fine to me" isn't how Web changed the world and is still here 30 years later

Complexity is the Enemy

Complexity is a paradox

- To do things requires complexity
- Complexity makes it harder to do things

> *"Everything should be as simple as possible, but not simpler"*

Coding is choosing **where to put the complexity**

We must be aware of our constant enemy, Complexity

Working is Not Enough

Code that runs and gives a correct answer is "working"

- That is not enough
- Is it understood?
- Can it be changed?
- Efficiency?
- What is it dependent on?
- Works with different browsers?
- Works on different devices?
- Accessibility?

Avoiding Tutorial Hell

Tutorial Hell: When new students study tutorials, but find they can't generalize any of their knowledge to solve real problems

I emphasize the opposite - **why** we make choices

I encourage you to wrestle with problems rather than look up a solution

I provide examples and have you practice using those in new ways

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

Tools for the Course

I have built this course to use tools like a job would

- I will note exceptions

Canvas

- Quizzes
    - Open book
    - Not timed
- Grades
    - Can take up to a week after due date
    - Details will be in Github
- Class Recordings
    - Check the date/time!
    - School requires physical attendance!

Zoom

- I will try to record class sessions
    - Technical issues always possible!
- Virtual Office Hours
    - Link posted in Canvas

Operating System

Mac, Windows, *nix

- All allowed for course (tools work on all)
- All exist in jobs
- Mac common (for devs) in bigger teams/companies
- Windows common in small teams/companies
- Windows common in explicit MS stack (.NET)
- *nix never common, but is underlying tech

I use Mac

- Windows users may need to adjust (rare)

Browser

A web browser is a significant tool for…web

- Course will use Chrome browser
- Current most common with users
- Many Devs prefer Firefox for development
- Our techniques should work in all major browsers
    - Developer Tools (DevTools) minor differences
- Real work should always be tested in all major
- But course will use only Chrome

Editor / IDE

- Used to edit HTML/CSS/JS/JSX/Markdown files
- Course does not have any requirement
- Instructor will use **vim** and **VSCode**
    - Instructor is old like dirt
    - Also vim presents fewer distractions
- **VSCode** is recommended for students
    - But not required
    - Most common with devs

Slack

We use **Slack** as our in-class chat

- Matches more jobs
- Actual job SKILL
    - Searching
    - Bouncing between channels
    - Not missing replies
    - Not starting with "Hi" (**https://nohello.com**)
- Sign up here: **https://rebrand.ly/vtlinfo6150-slack**
    - I need to unlock existing accounts
        - If you started the class previously

Slack Notes

- Good for code snippets
  - Use ` (backtick) around a command
  - Use ``` (triple backtick) around code block
- Screen shots often problematic
  - Unreadable on mobile
  - Can't copy code to test
- Feel free to ask questions on Slack!
  - I'll respond when available
  - If you don't hear back, remind me!
- I announce changes to assignments, class cancellations, etc

Git

git is a version control system

- Tracks changes to files
- HEAVILY used in jobs
    - When not git, will be some other VCS
- Git is a lot to take in!
    - Notes will have suggestions to learn it better
- Being able to use git great for jobs
- Knowing more than the minimum even better!
- You will need to download/install

Github

- `git` manages a local (your computer) "repository"
- Github is a central repository ("repo")
- Each student will have their own Github repo
    - Get yours here: **https://rebrand.ly/vtlinfo6150-github**
- I push notes and assignments to github
- You pull to your local repo from github
- You push work from your local repo to github
- TAs and I grade and merge (more detail later)
    - Real job: You merge

NodeJS and related tools (npm, npx)

- Javascript (JS) for server-side
- We use it but not write for it
    - That's my 6250 class
- You will need to download and install it
    - Either "Latest Stable" or "LTS" version
    - Do not install node libs/packages as "sudo"/administrator

Class Recordings

- I try to record classes
  - Links available on Canvas
- Always a risk of technical problems!
  - Missing class is a risk
- University has physical attendance policy
  - Remember to get excused absences!

Accommodations

If you benefit from accommodations

- Lasting or temporary
- For reasons physical, mental, or emotional
- Let me know and I will work with you
- Your reasons can stay private

Changes this Semester - Website

Based on feedback from previous semesters:

- "You talk so much, it is hard to follow for 3 hours"

Trying this semester:

- Made a website of material last semester, will be adjusting
    - You will be asked to read it between classes
- Two 10/min breaks

ChatGPT (etc)

- Many students last semester ignored my teaching
    - Work quality dropped mid-semester
    - Ended semester with less knowledge than previous years
    - Caused conflict with grades and expectations
- Setting clear standards up front

Grading Scales

- Department Grading Scale says 95%+ to get an A
- I switched to more clear rubric
    - Assignments had few points, making even one mistake harsh to grade

Changes this semester:

- New Grading system (again)
    - Missing "Critical" requirements mean -10% from assignment
    - "Additional Requirements" mean -2% each

Your Repository

Key things to find in your repo

- Syllabus (`/syllabus.md`)
- Do Not Copy policy (`/do-not-copy-work.md`)
- Samples (`/samples`)
- Class Notes (`/classes`)
- Class Assignments (`/work`)
- Projects (`/project1`, `/project2`, `/final`)

Tips for Excelling in this course

- I teach more than is required to pass with an A
- What I teach should help you on the job
- Ask questions!
- Use DevTools to debug/diagnose!
- Do the assignments from scratch
    - NOT using others' code "as a reference"
        - Do NOT copy code
    - Practice teaches MORE than main lesson
    - Not relying on ChatGPT/LLMs
- Start Projects ASAP
    - Coding always takes longer than expected

If you fall behind...

- Tell me
    - As soon as you notice
- We can figure out how to fix things
    - "Working harder" is a nice thought
        - Often that's not the problem
- This is a JOB SKILL
    - Manager and team want to know ASAP
        - So they can prevent problems
    - Not about blame

Use Caution with Online Resources

- A lot in Webdev has changed over time!
    - Lots of outdated/misleading resources
- Never trust any source older than _3_ years old
- Prefer **MDN** as a source
    - W3Schools is "okay"
    - I recommend MDN first
- **Never copy/paste code** (HTML/CSS/JS)
    - Understand and recreate it first
    - This includes ChatGPT
    - Copying limits your skill