

Web Application Technologies

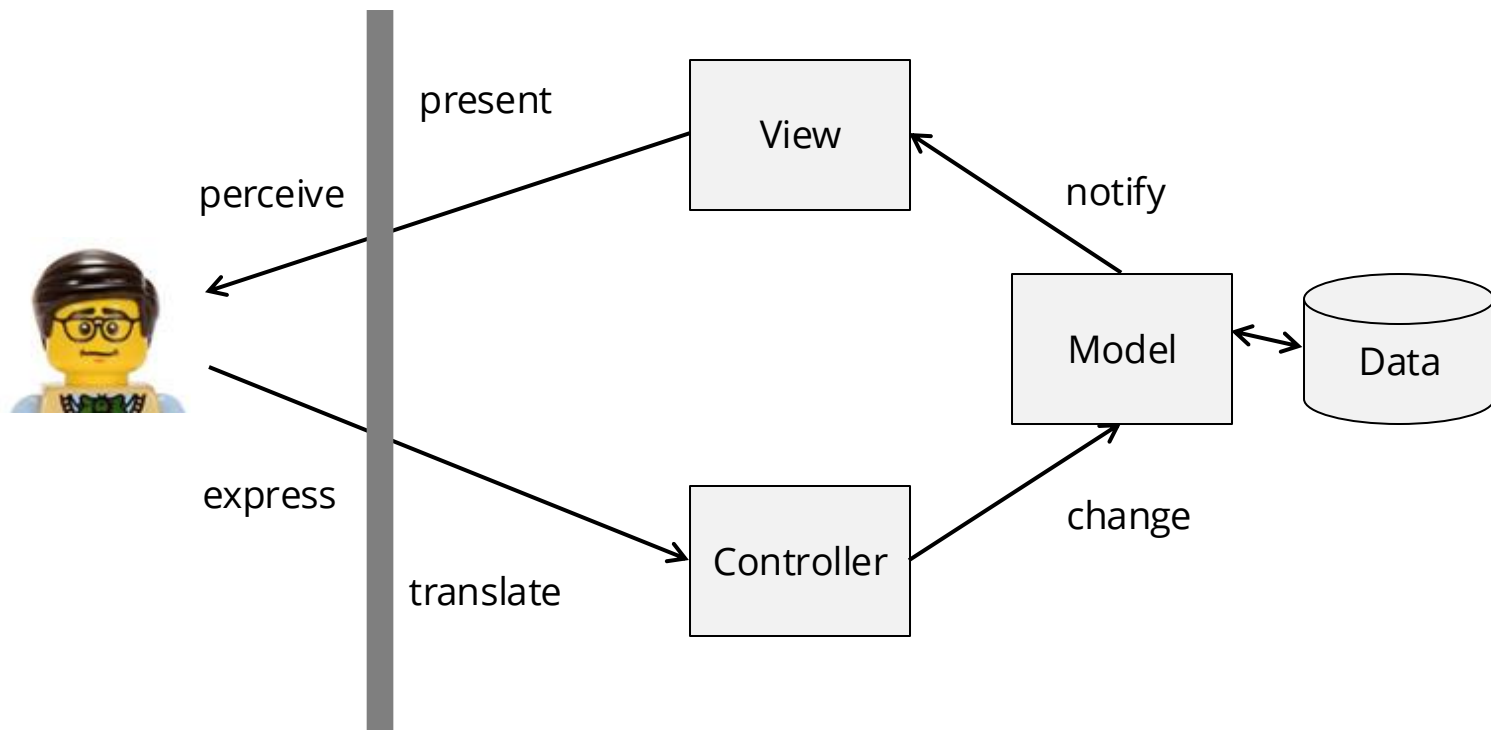
- Git and Gitlab
- VS Code
- Node.js
- npm
- Chrome/Chromium
- Vite
- SimpleKit and Preact

Web Applications

- A software interface accessed *through* a web browser
- The browser acts like an operating system
 - handles input, provides canvas for drawing, etc.
 - provides UI toolkit (HTML, CSS)
 - provides "machine code" layer (i.e. JavaScript compiler)
- Web apps typically delivered to users from a server
 - can be designed to run offline (Progressive Web Apps, Electron)
- Conceptual split between user interface and "business logic"
 - UI is client-side, business logic is server-side
- Historical connection to early client-server architectures

MVC

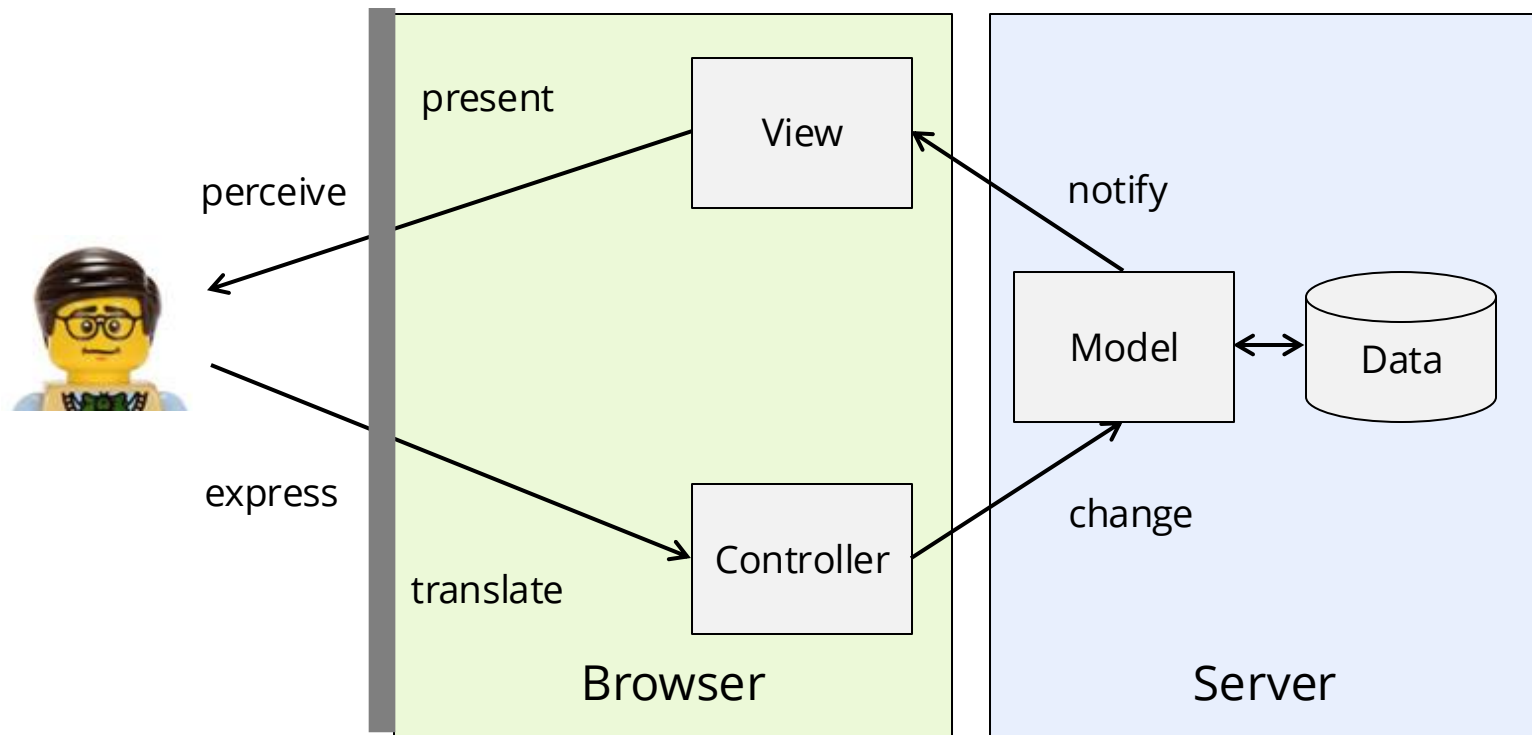
- Useful to consider data store as well



MVC View of Early Web Apps

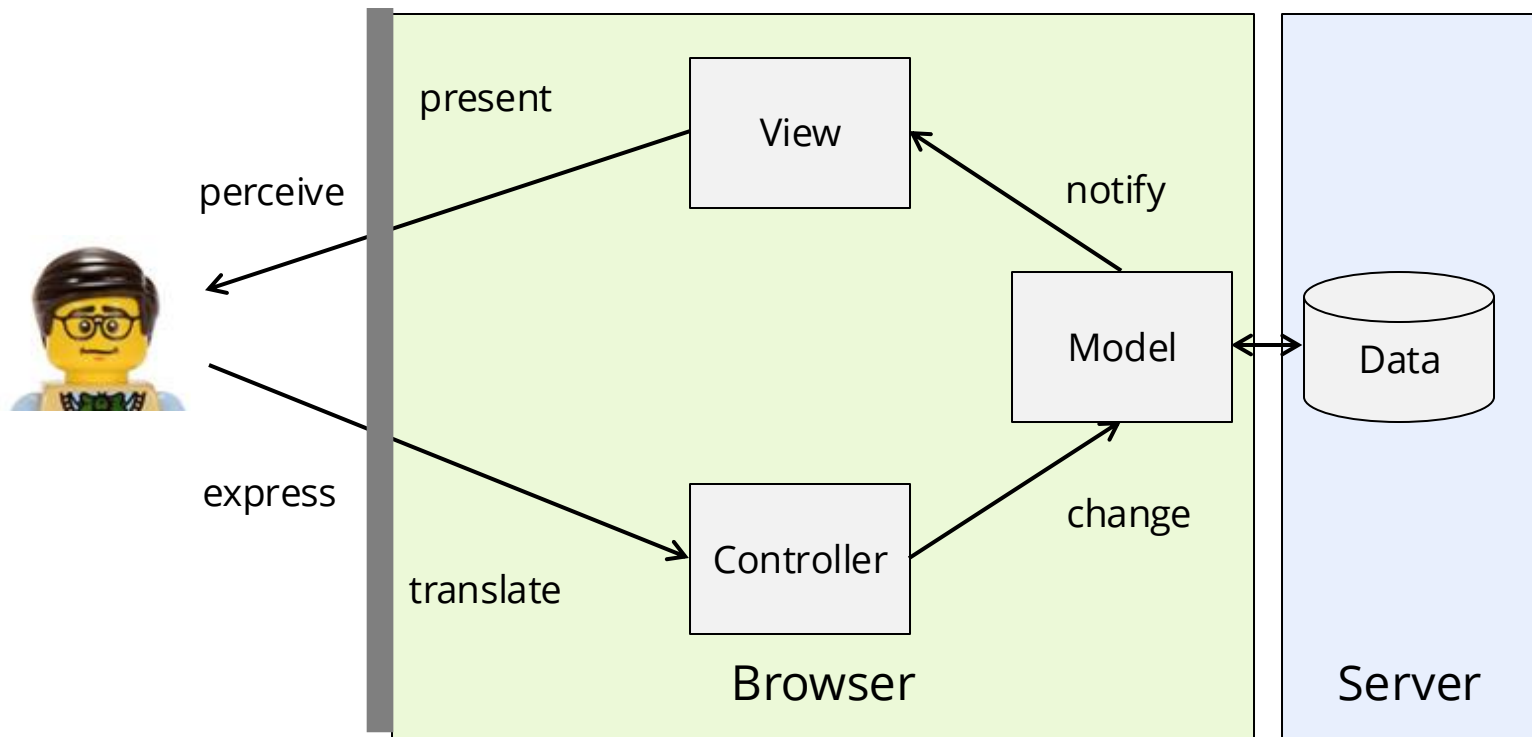
Note a recent trend moving back to this with, e.g. SSR with hydration, HTMX

- **Model** *on server* sends webpage to *browser* to render the **View**
- **Controller** *in browser* sends user events to **Model** *on server*
 - click on hyperlink, submit form, etc.
- **Model** processes changes, then sends new webpage to *browser* ...



MVC View of a Single Page Application (SPA)

- Browser handles full MVC cycle with data persisted on a server
- Model can request server data/processing too (e.g. Web APIs)
- In CS349, we focus on SPAs without server data/processing



CS 349 Development Environment

You're required to use a specific “stack” of "web dev" software

- to mitigate compatibility issues and enable us to provide support
- Development environment:
 - Git and Gitlab source code management
 - VS Code editor
 - Node development server
 - npm package manager
 - Chrome/Chromium browser
 - Vite front-end tooling
 - TypeScript language (details next lecture)
 - UI Frameworks (SimpleKit, Preact)

A0 is a small assignment to setup your development environment:

- (1) you must use the software versions specified in A0;
- (2) you must submit A0 before submitting any other assignments.

Git and Gitlab

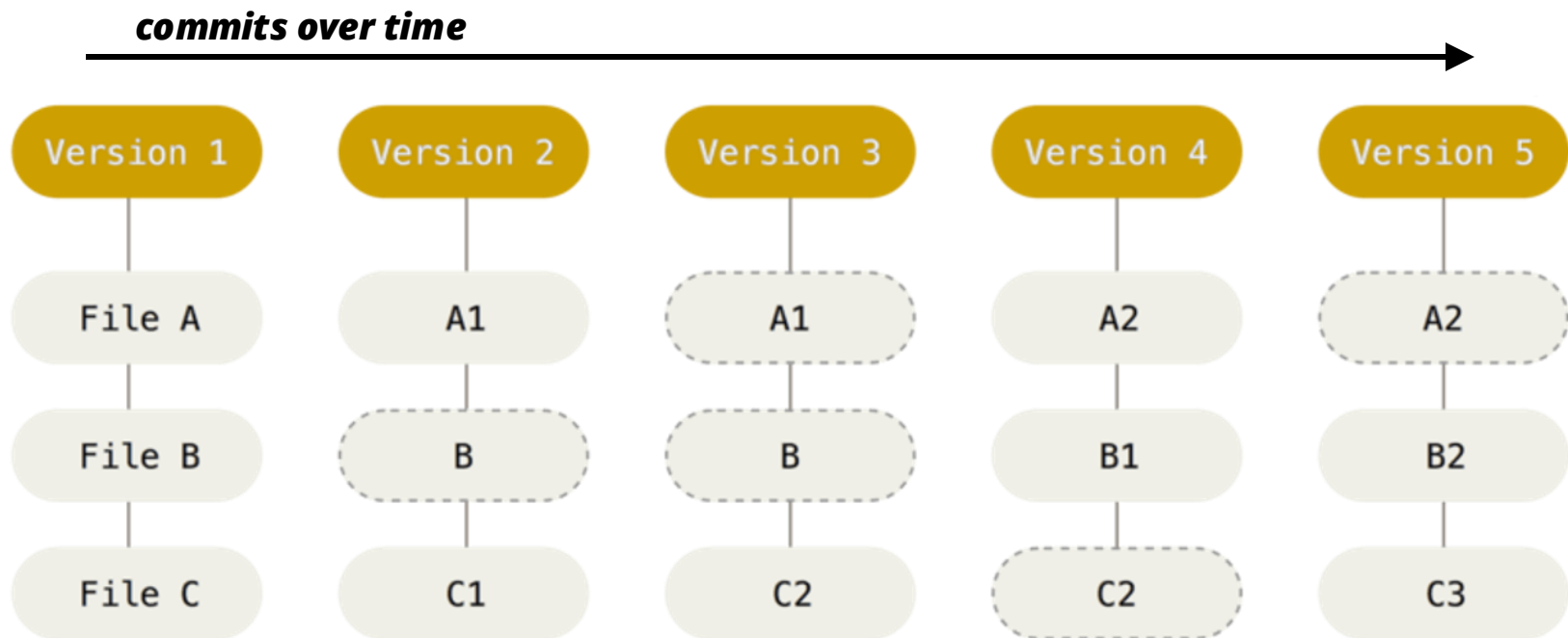


Git

- A version control system
 - tracks and manages changes to source code over time
- Benefits
 - enables multiple developers to collaborate on same source code
 - supports independent streams of changes, i.e. branching and merging
 - tracing changes to find bugs, audit code, etc.
 - secure and safe storage of source code
- Command line based, but integrated into editors (e.g. VS Code)
 - here are standalone GUIs, but not necessary for this course
- Install:
 - <https://git-scm.com/downloads>

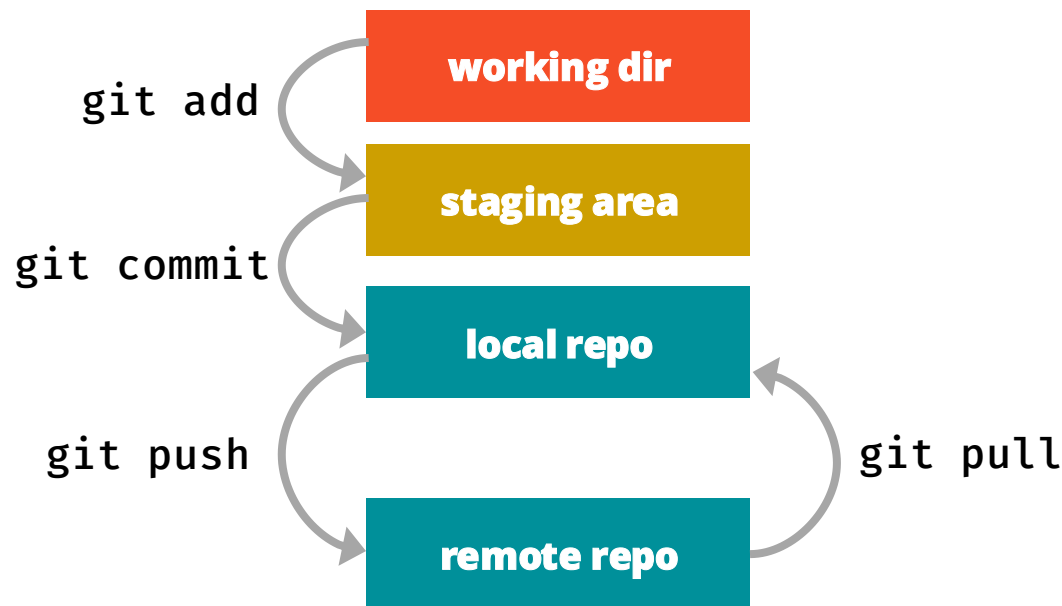
Git commit

- Core conceptual unit in Git is the **commit**
 - snapshots of "tracked" files as they change over time
 - every commit is an explicit action:
`git commit -m "fixed bugs"`



Git Concepts

- Git is designed around a distributed model:
 - **Working Directory:** a local copy of source code
 - **Staging Area:** intermediate area where commits are reviewed before completing the commit
 - **Repository (Repo):** a data structure of commits, usually there's a "local repo" on your machine *and* a "remote repo" on a server



Git Commands

- Common commands move files between working directory, staging area, and the local and remote repos

- Commands to manipulate the staging area and commits:

`git init` create new empty local repo

`git add <file>` add file from working directory to staging area

`git commit -m "description"` commit changes from staging area to local repo

`git status` Display list of changed files and staged files

- Commands to synchronize with a remote repository:

`git clone username@host:/path/to/repository` make a copy of a repo

`git pull` merge commits in remote repo with local repo

`git push` send commits in local repo to remote repo

Typical Git Workflow

1. Get copy of a remote repo on your computer

```
git clone username@host:/path/to/repository
```

2. Update your source code in the local repo:

New file

```
git add <file>
```

Delete file

```
git rm <file>
```

Rename file

```
git mv <old> <new>
```

Edit file

just edit it!

3. Add all source code updates to the staging area

```
git add -A
```

4. Check staging status to verify everything is ready to commit

```
git status
```

5. Make the commit

```
git commit -m "desc of commit"
```

6. Push changes to remote repo

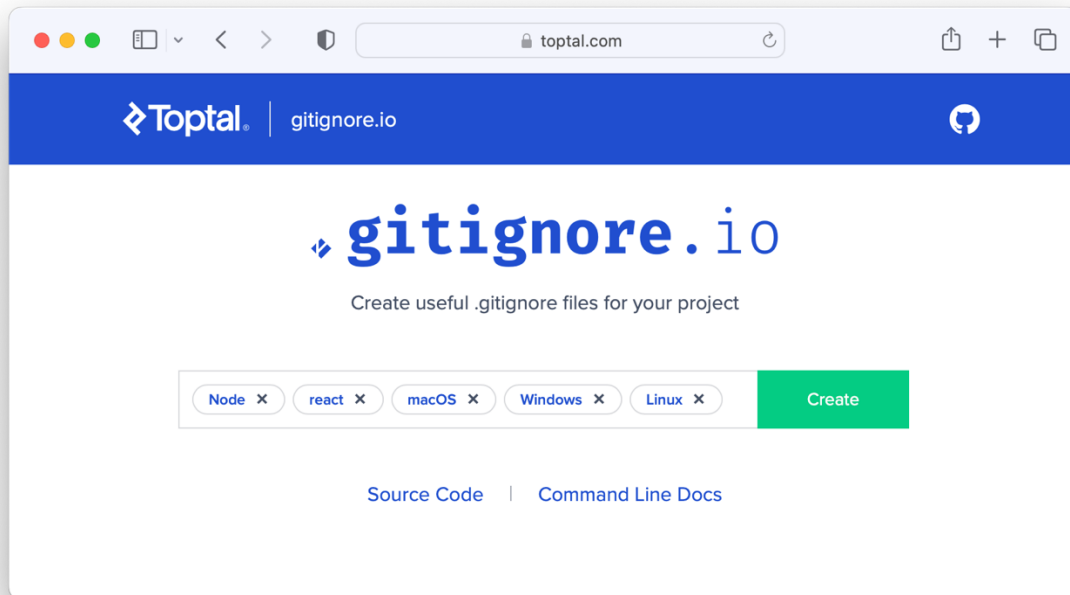
```
git push
```

7. Go to step 2

you don't have to push **every** commit to the server right away, can build up commits in local repo then push all at once

.gitignore

- A file to specify untracked files that Git should ignore
 - Some frameworks and development tools provide a .gitignore
 - Good practice to create a master .gitignore for a repo in the root
- Tool to generate a good base gitignore:
 - <https://www.toptal.com/developers/gitignore>
 - ignore tags for CS 349: **node, react, macos, linux, windows**



~~.DS_Store~~

Ignoring files that are already tracked

- What if you track a file, then want to ignore it later?
 - i.e. you forgot to ignore it when setting up your repo
- Files *already* tracked are not affected by changes to .gitignore

- If you want to ignore a file that was tracked by accident

```
git rm --cached [filename]
```

cached flag means remove from git not filesystem

```
git add --all
```

```
git commit -m "removed files tracked by mistake"
```

- If you want to ignore *all files* that were tracked by accident:

```
git rm --cached -r [directory]
```

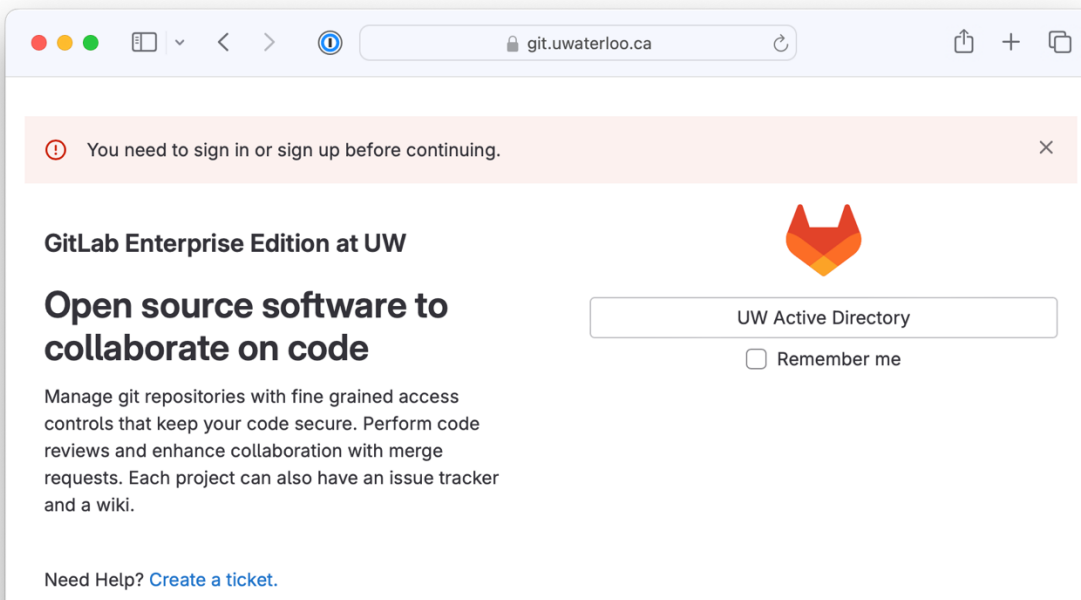
BE CAREFUL -r is recursive delete

```
git add --all
```

```
git commit -m "cleaning files that should have been ignored"
```

Gitlab

- Store and manage Git repos, a "remote repo"
 - similar to GitHub
- UWaterloo hosts its own GitLab installation: I think we host it?
<https://git.uwaterloo.ca/>
 - uses standard UWaterloo SSO authentication
- We use this for demo code and assignments in CS349
... but you can create personal repos too!




Clone CS349 Demo Code Repo

You should have lecture demo files in a local repo on your machine, and you should keep them up to date throughout the term.

1. Get a local copy of the cs349 demo code repo on your machine:

```
git clone https://git.uwaterloo.ca/cs349/public/1251 demos
```



local dir to
put repo

2. Follow the “Setup” instructions in the README

- especially to initialize SimpleKit git submodule

3. Follow “Keeping Up to Date” instructions in README

- some special setup and methods due to SimpleKit git submodule

Clone Your CS349 Assignment Repo

Get a copy of your assignment repo on your machine:

```
git clone https://git.uwaterloo.ca/cs349-winter2025/mbrehmer assignments
```

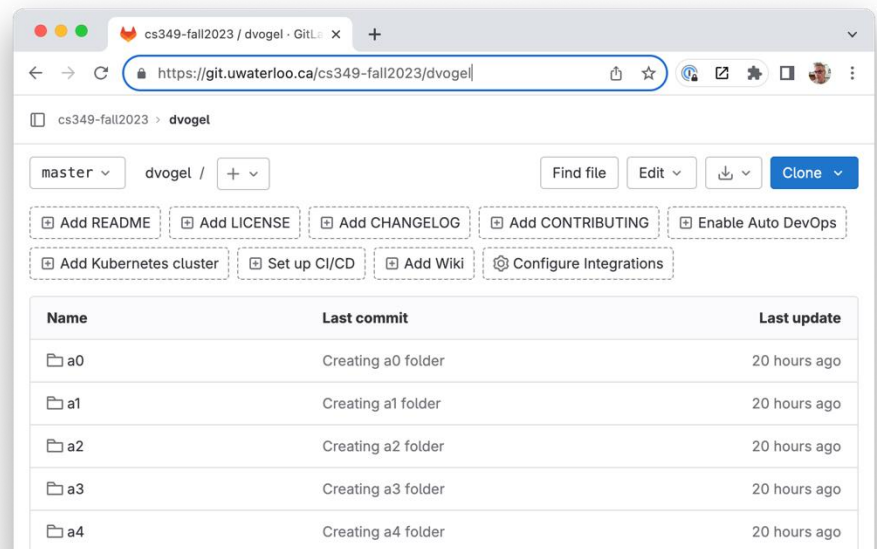
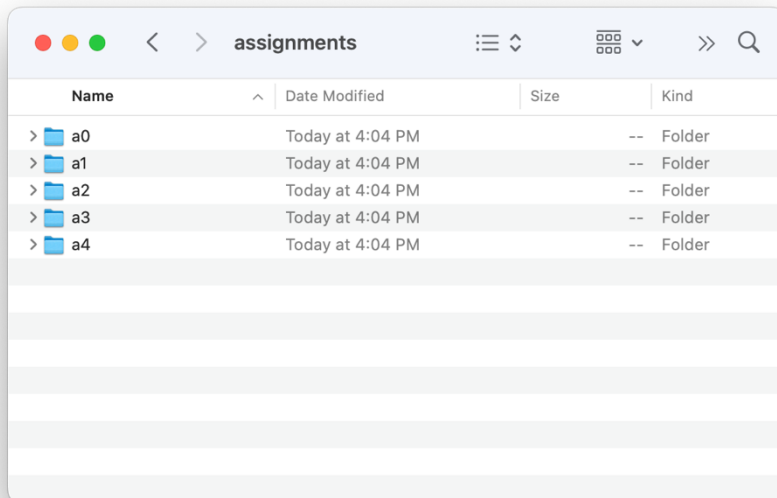
your uwid

local dir to
put repo

Check the remote repo links with `git remote -v`

You'll use this directory to work on your assignments

- Update your source code and commit locally often
- Push commits to your remote repo often



VS Code

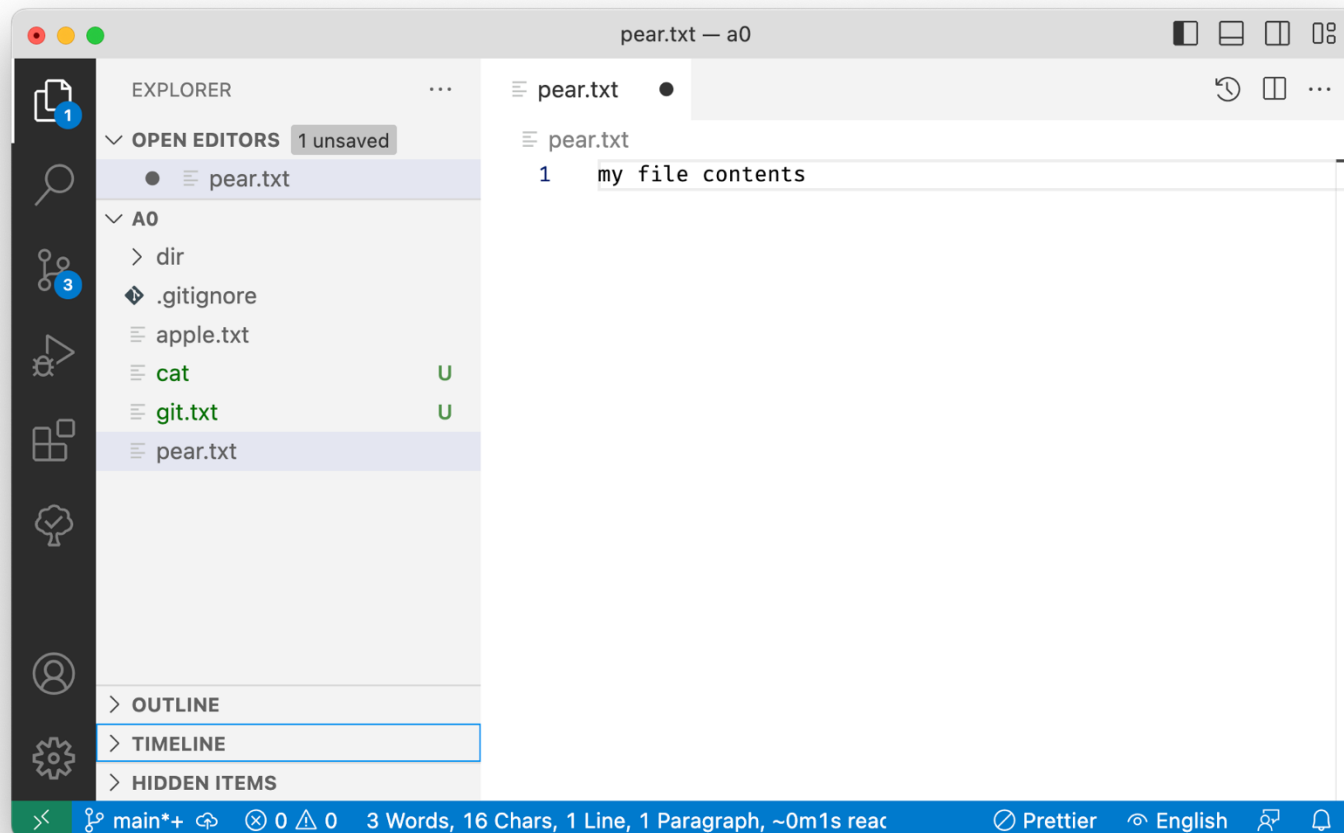


VS Code ("Visual Studio Code")

- Powerful source code editor for Windows, macOS, and Linux
- Built-in support for JavaScript, TypeScript, and Node.js.
 - huge ecosystem of extensions for other languages and runtimes (e.g. C++, C#, Java, Python, PHP, Go, .NET)
- From Microsoft, it's free, it's "built on open source"
 - it's not the same as Microsoft Visual Studio IDE
- VS Code is *built using web technologies*
 - JavaScript, Node.js, etc.
 - Packaged as a desktop app using the [Electron Framework](#)
- Download and install:
 - <https://code.visualstudio.com/>

Workspaces

- VS Code uses the concept of a "workspace"
 - In most cases, this is just a root directory to your source code
 - "create" a workspace by dragging folder onto VS Code window, then "Save as Workspace"

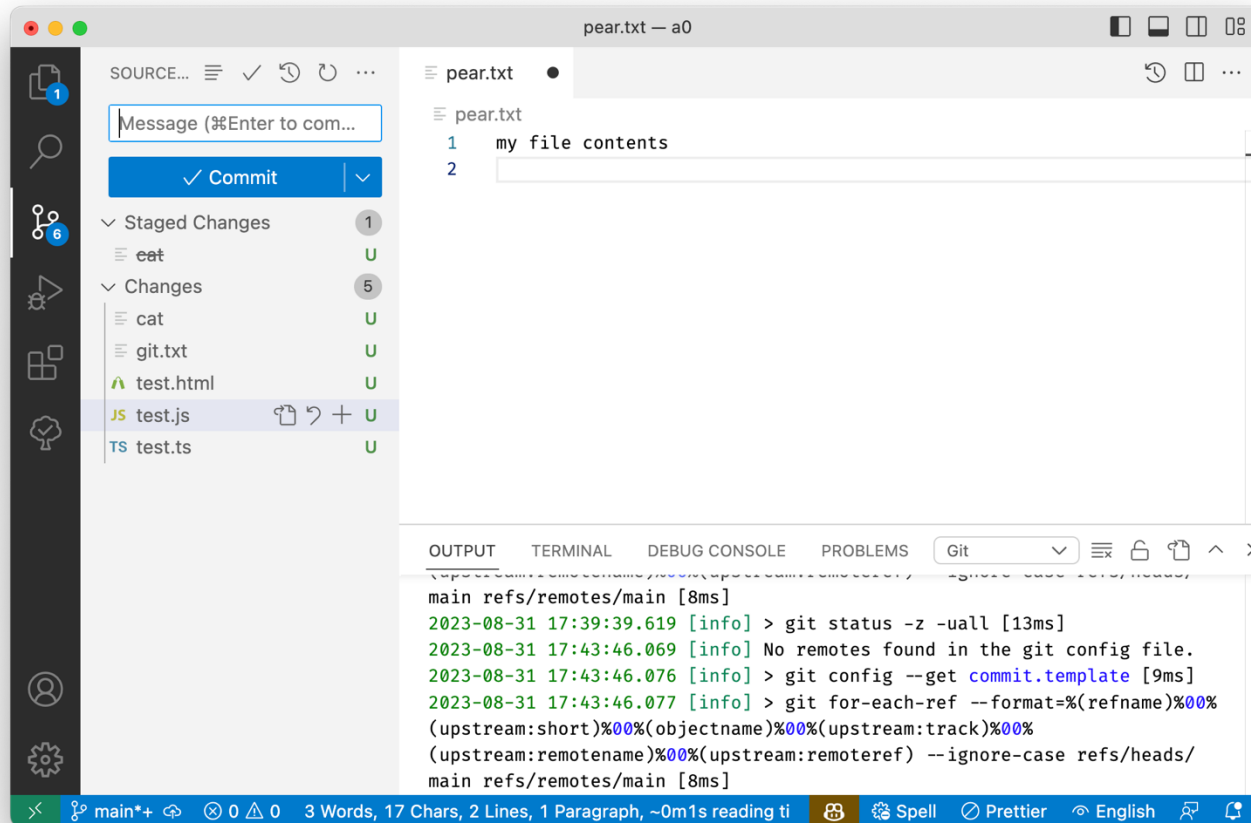


Some Notable Features

- Command Pallet
 - **CMD SHIFT P**
- Built-in terminal
 - **CMD J** to hide/show terminal
- Fix problem
 - **CMD .**
- Region Folding
 - `//#region This can be hidden`
 - `...`
 - `//#endregion`

Git Integration

- You can clone, add, commit, push, pull, and more in VS Code
- Setup and usage instructions:
 - <https://code.visualstudio.com/docs/sourcecontrol/intro-to-git>
 - (instructions apply to GitLab too, just use CS349 repo URLs)



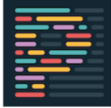
VS Code Git Demo

- Clone repo
- drag folder into VS Code
- edit README.md
- Stage (+), then enter msg and commit
- Push (i.e. "Sync Changes")
- Save workspace
- Check status
- Add .gitignore file with *.code-workspace
- Check status to see it's gone
- Create file.txt
- Enter msg and commit

VS Code Extensions

Many VS Code extensions available and they're easy to install

- always check if functionality already in VS Code, a lot is

Required for CS 349: **Prettier** 

- opinionated code formatting
- set it as formatter by calling format (e.g. SHIFT-OPTION-F)
- 🔥🔥 configure VS Code to “format on save”

Optional: **GitHub CoPilot** 

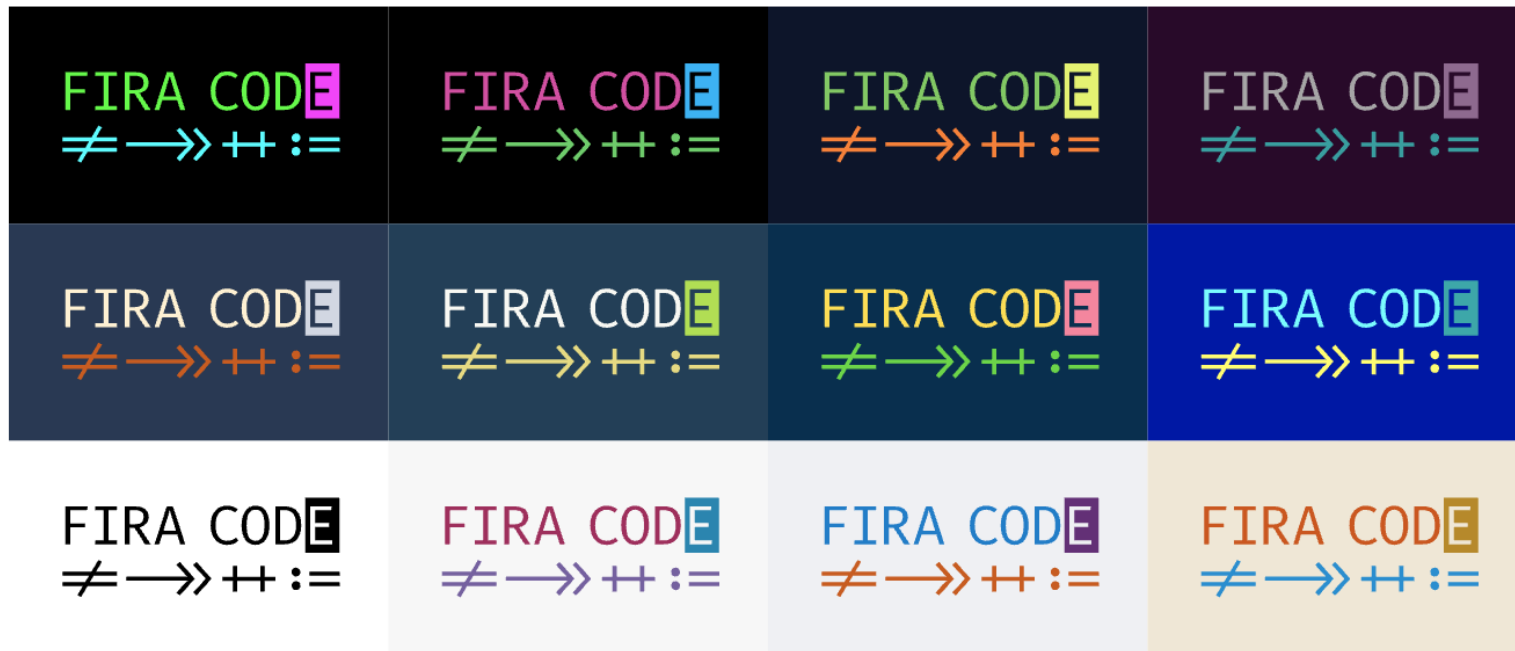
- AI code completion and code generation
- good for learning about how to use an API
- fine for generating small functions that aren't focus of course
- often makes mistakes, you must understand what it generates
- **you must document where you used it in assignments**

Font Ligatures

- *Ligature*: a unique character created by joining multiple characters

<= === => **becomes** ≤ ≡ ⇒

- Easy to add to VS Code
 - install a font with ligatures, like **Fira Code**
 - configure a VS Code setting



Node.js



Node.js

- An open-source, cross-platform JavaScript runtime environment
 - event driven and asynchronous
 - can be used to develop server-side applications
 - also useful for webdev toolchains, transpiling, dev server, etc.
- We won't use Node.js directly in this course
- Install options
 - download installer from <https://nodejs.org/en>
 - use package manager (e.g. HomeBrew on MacOS)
 - use **nvm** (Node Version Manager)

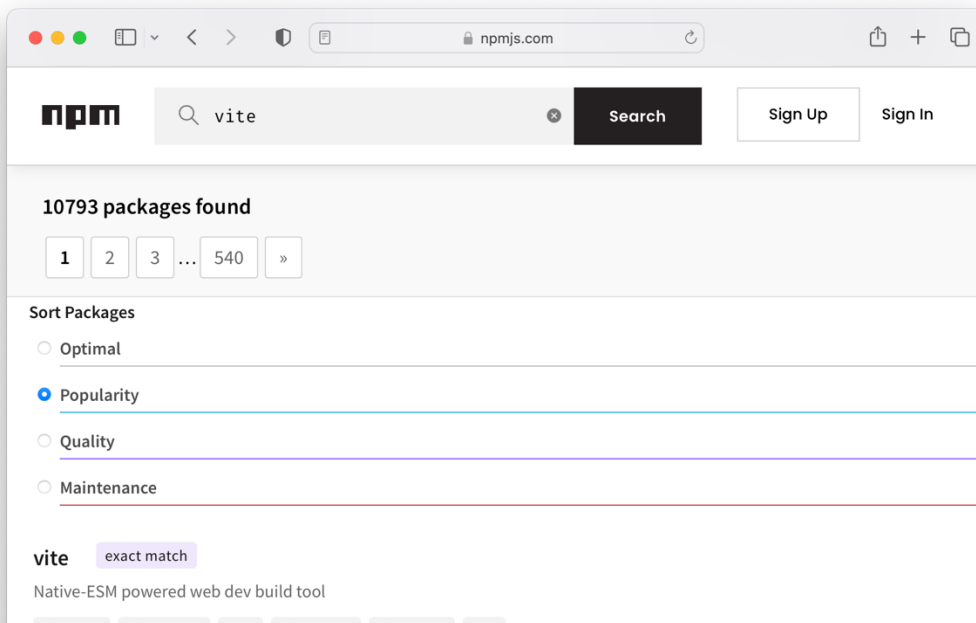
npm



npm

- Installed with Node.js
- A library/registry of JavaScript software packages
 - name means "Node Package Manager", but does more now
- Command-line tools to:
 - install packages
 - manage dependencies
 - manage development environment by running scripts

npm denies this



Common npm usage

- Initialize Node project

```
npm init or npm create # alias for init
```

we'll see this form with Vite

- Install a package from the npm library/registry

```
npm install <package> or npm i <package>
```

- install options

```
--save-dev # package is for development only
```

```
-g # install package globally (i.e. in your system)
```

- Run script

```
npm run <script-name>
```

- List installed packages

```
npm list or npm list -g # to list global packages
```

- List outdated packages

```
npm outdated
```

Returns nothing if not outdated

- Update a package

```
npm update <package> or npm up <package>
```

Node project files

package.json

- list of all packages installed in project
- every `npm install` adds to this file, often with many dependent packages as well
- has information to re-create installed packages
- ```
node init # if package.json exists, installs all packages
```
- add to your repo

## package-lock.json

- information to **more precisely** reproduce `/node_modules`
- add to your repo

## node\_modules/

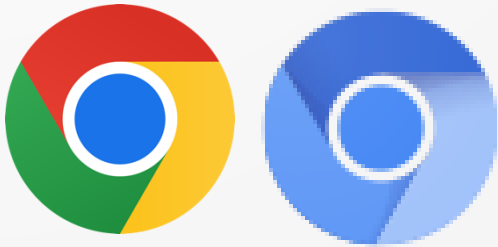
- quickly becomes very large, 1000s of small files
- 🔥 important to ignore `node_modules/` (must be in `.gitignore`) 🔥
- avoid syncing with Dropbox, GDrive, etc.
- can just delete it, then run `npm install` to re-install all packages

# **npx (Node Package Execute)**

- execute an arbitrary command from an npm package (either one installed locally, or fetched remotely), in a similar context as running it via `npm exec`
- Example:
  - npx some-package**
  - If `some-package` is *in your path* (i.e. it was installed using npm), then it runs the local version of the package
  - If `some-package` is *not in your path* (i.e. not installed), then it downloads the latest version of the package and runs it

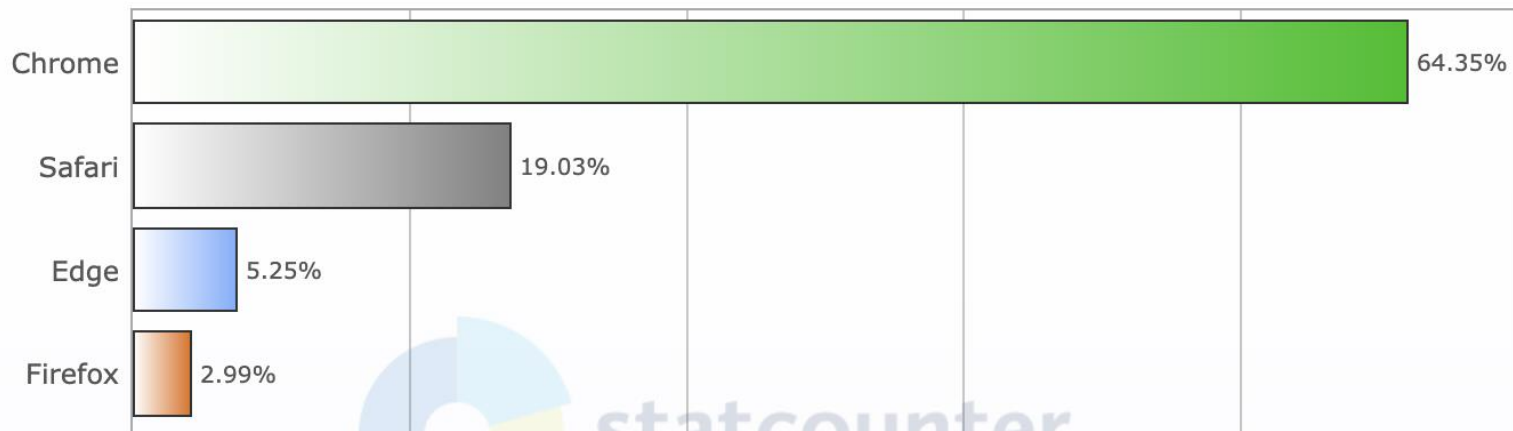


# Chrome and Chromium



# Google Chrome or Chromium Browser

- You're required to use a standard browser in this course
  - Latest stable release of Chrome:  
[https://www.google.com/intl/en\\_ca/chrome/](https://www.google.com/intl/en_ca/chrome/)
  - Chromium if you'd prefer to stay out of the Google ecosystem  
<https://www.chromium.org/Home/>
  - Other Chromium-based browsers should work
- TAs will only mark using a Chrome or Chromium browser



# Web Browsers Typically Have Two Main Parts

- **JavaScript Engine**

- execute JavaScript

- **Rendering Engine**

- transform HTML documents and other resources of a web page into an interactive user interface

# Chrome uses the V8 JavaScript Engine

Open-source, written in C++

Two main parts:

- **Interpreter**

- reads JavaScript code and executes it directly

- **Just-in-time (JIT) compiler**

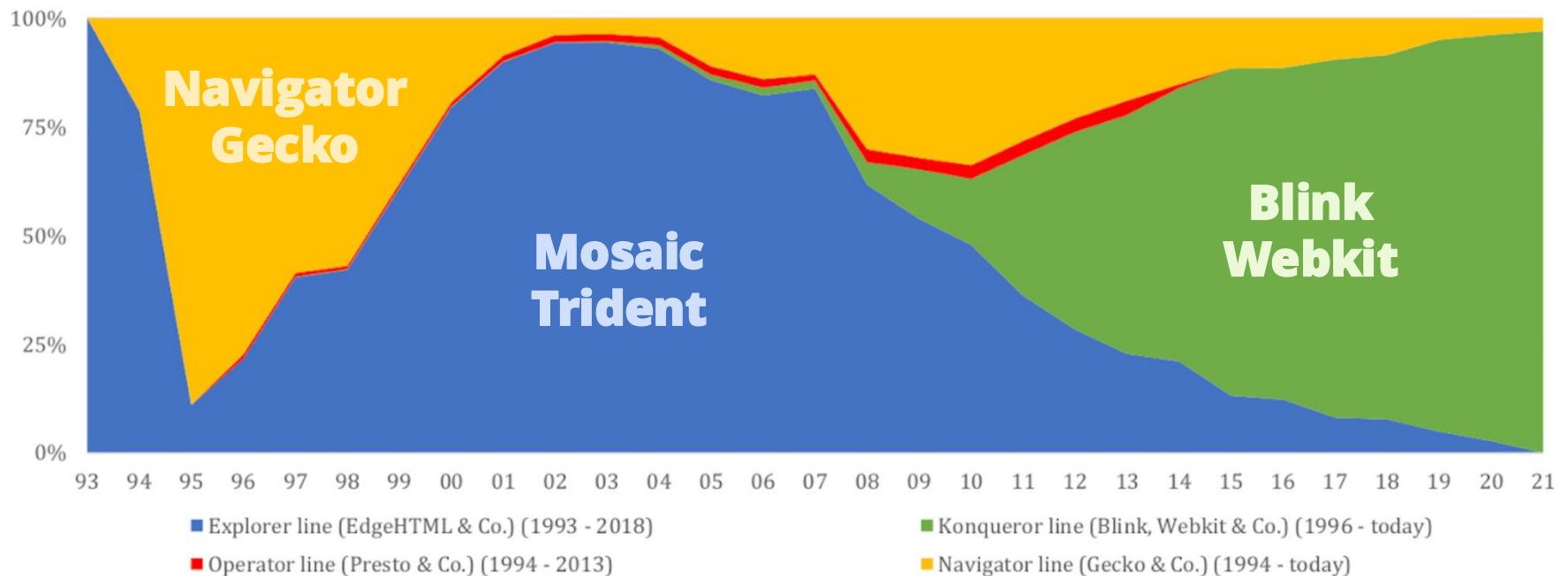
- compile frequently executed code to machine code (for faster execution)

Can execute JavaScript with surprising efficiency



# Chrome uses the Blink Rendering Engine

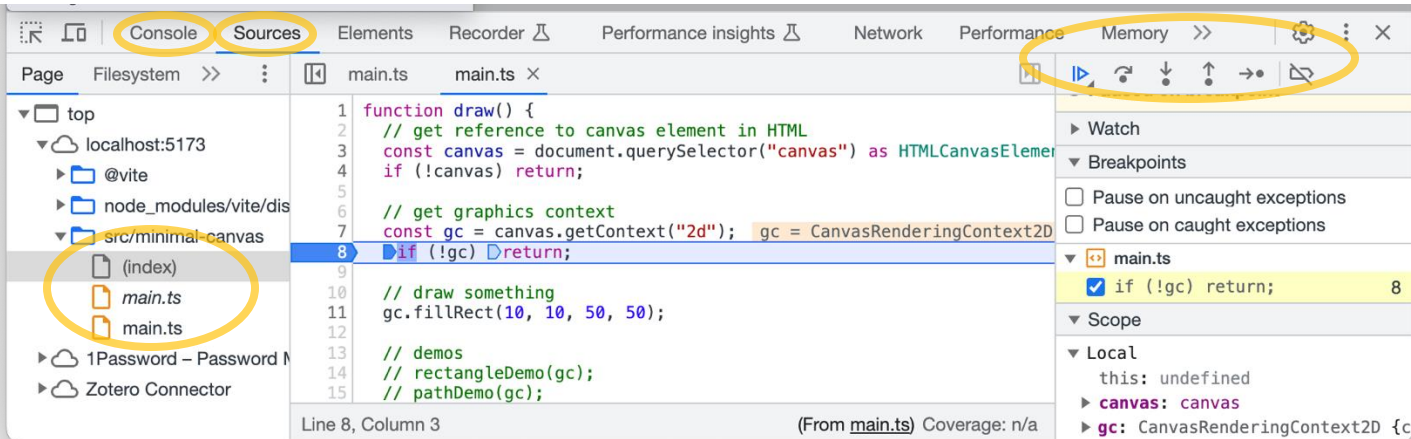
- implements Document Object Model (DOM)
  - layout of elements
  - rendering elements (styles, etc.)
- security between documents
- navigation using hyperlinks and forms
- Blink is a fork of the Webkit browser engine



# Browser Developer Tools

- Modern web browsers include developer tools:
  - inspect currently-loaded HTML, CSS, and JavaScript
  - report page asset requests, how long they took to load
  - simulate different devices, pages sizes, network speeds
- In CS349, most important ones are:
  - JavaScript **Console**
  - **Sources** (for debugging)
- **OPTION + CMD + j** to open developer tools and show console

memorize this shortcut!



**Vite**



# Vite

(pronounced "veet")

"Next Generation Frontend Tooling"

Vite has two main parts:

1. A **dev server** to run code in a non-production environment
  - runs local webserver
  - watches TypeScript source files and re-transpiles as needed
  - uses optimizations like Hot Module Replacement (HMR).
2. A **build command** to bundle code for deployment to production
  - uses rollup to optimize code and assets

In CS349, we only use the *dev server* part



# Create a Vite Project

- Vite has templates to setup a project for different frameworks
- Setup Vite project by choosing name and templates interactively:  
`npm create vite@latest`
- (can also provide project name and template in command args)

| JavaScript              | TypeScript                 |
|-------------------------|----------------------------|
| <a href="#">vanilla</a> | <a href="#">vanilla-ts</a> |
| <a href="#">vue</a>     | <a href="#">vue-ts</a>     |
| <a href="#">react</a>   | <a href="#">react-ts</a>   |
| <a href="#">preact</a>  | <a href="#">preact-ts</a>  |
| <a href="#">lit</a>     | <a href="#">lit-ts</a>     |
| <a href="#">svelte</a>  | <a href="#">svelte-ts</a>  |
| <a href="#">solid</a>   | <a href="#">solid-ts</a>   |

we'll use this for first part of course

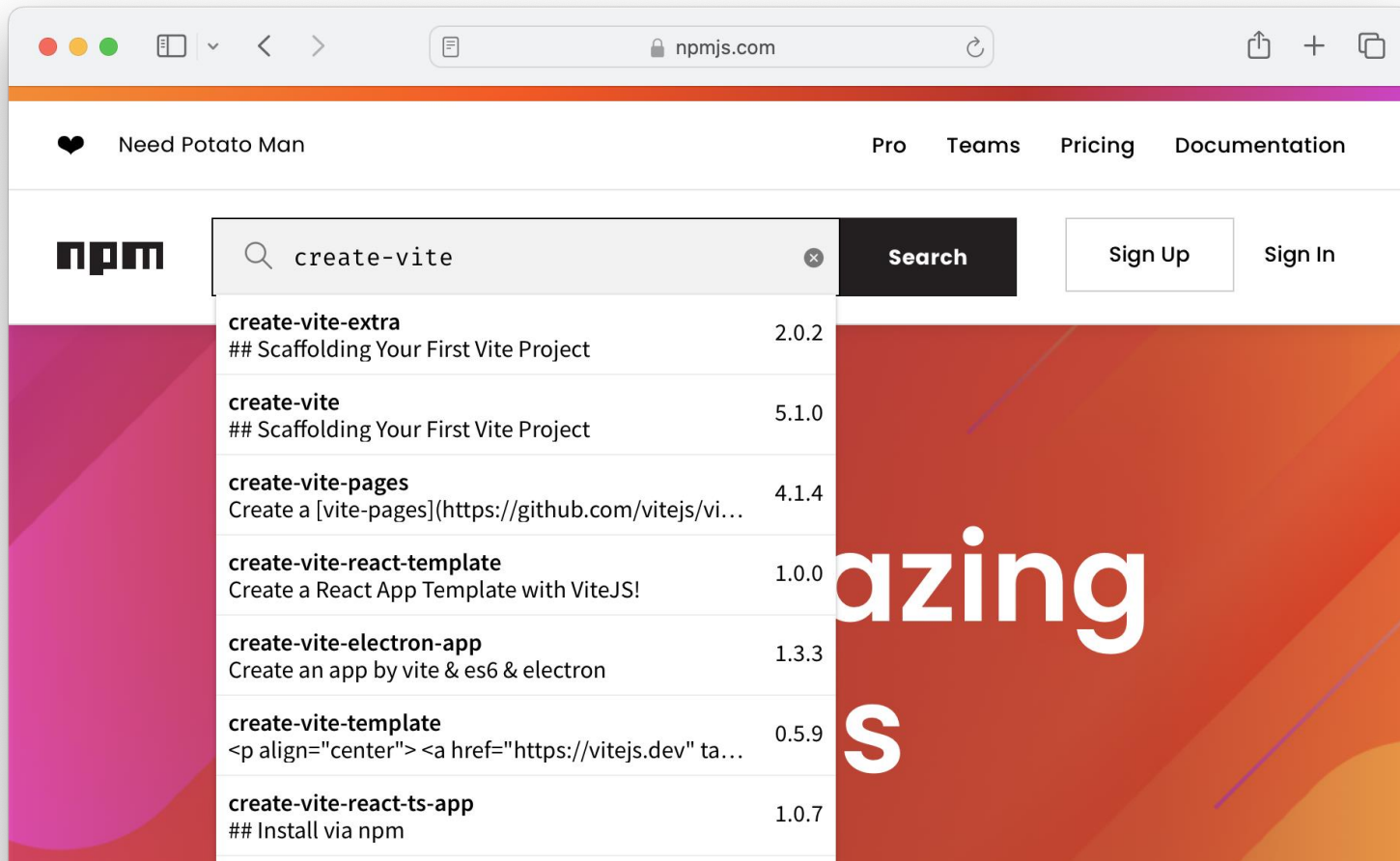
later, we'll use this

# npm create vite@latest

npm create initializes a project directory using a "create" package

vite in this context means the create-vite package

@latest just means "use latest version" of create-vite package



# Vite Project Setup Demo

Create project with Vanilla TypeScript template

```
npm create vite@latest
```

Run Vite dev server

```
npm run dev
```

Examine Vite project structure

- index.html, especially `<script type = "module" ...`
- src/ directory

Show how to create “clean” project:

- simplify index.html
- Remove demo files from src/ folder:  
counter.ts, style.css, typescript.svg, vite-env.d.ts
- Clear contents of main.ts

# UI Frameworks

SimpleKit

Preact

# SimpleKit

- A very simple user interface toolkit for teaching UI architecture
  - Built for CS 349
  - You'll learn how it works and how to use it in lectures
  - We'll use it for A1 and A2
- <https://www.npmjs.com/package/simplekit>



# Preact

- A fast 3kB alternative to React with the same modern API
  - We'll use it for A4
- <https://preactjs.com/>



# Exercise



## 1. Sign on to cs349 Piazza

- All course announcements will be in there

## 2. Clone the cs349 Demo Code Repo

- Then keep it up to date throughout the term

## 3. Clone your cs349 Assignment Repo

- Student assignment repos are auto generated
- You'll get an email with the URL

## 4. Do A0

- You have everything you need to complete it now
- It should take less than 30 minutes  
(unless you run into issues)