
CSE 331

Software Design & Implementation

Autumn 2024
Section 1 – HW1 and Tools

Welcome

- Let's all introduce ourselves:
 - Name and pronouns
 - Year
 - What other classes you are taking this quarter
 - Food that starts with the first letter of your name

Administrivia

- HW 1 released later today, due Saturday (October 5th) at 11pm

Coding Setup

Software we will use

- **Bash:** command-line shell (built-in on Mac, see course website to download Windows version)
 - Run `echo "${BASH_VERSION}"` to check for download
- **Git:** version control system (built-in on Mac, Windows version comes with Bash, above)
- **Node:** executes JavaScript code on the command-line (see link on course website to install)
 - Run `node -v` to check for download
- **NPM:** package manager (comes with Node, above)
- **VS Code** or the editor of your choice

Node Demo

- **Node**: executes JavaScript code on the command-line (see link on course website to install)
 - Run `node -v` to check for download
- Useful for playing with the JavaScript language
- Try this to see what it does (does it crash?)
 - first start `node` and then type this in:

```
const x = {a: 1, b: "two"};  
console.log(x.c);
```

Git Demo

- **Git:** version control system (built-in on Mac, Windows version comes with Bash, above)
- Almost all professionals use some kind of version control system
 - git is probably the most popular today
 - git can be tricky to learn / understand
- We will only need it for getting the starter code
 - here is the command for sec1 (similar command for HW1)

```
git clone  
https://gitlab.cs.washington.edu/cse331-24au-materials/sec01.git
```

NPM Demo

- **NPM**: package manager (comes with Node)
- Used to
 - install all the libraries needed for our code
 - compile, test, and run our code
- Use this command to install the libraries needed for sec1

```
npm install --no-audit
```

(leaving off `--no-audit` will generate some **bogus** error messages)

VSCode Demo

- **VS Code** or the editor of your choice
- VS Code is relatively lightweight IDE
 - primary support for JavaScript and TypeScript (good for us)
- Extensions provide support for other languages and tools

NPM Start

- **NPM:** package manager (comes with Node)
- Use this command to start

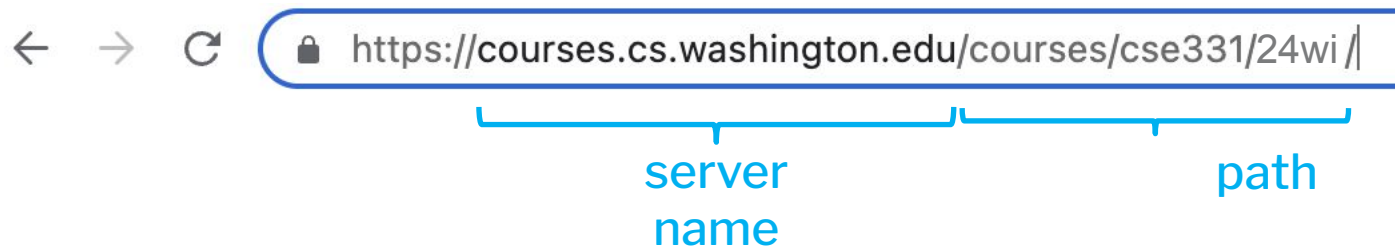
```
npm run start
```

- Then navigate to this URL in Chrome to see it work

```
http://localhost:8080
```

Browser Operation

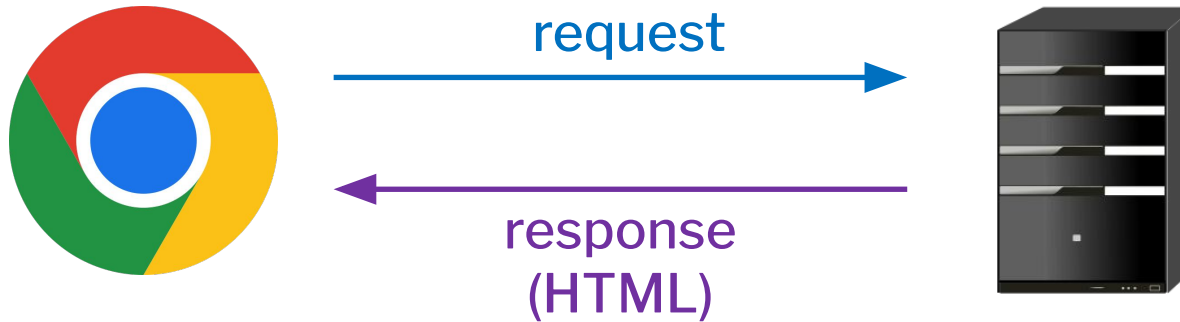
- Browser reads the URL to find the server to talk to



- Contact the given server and request the given path:

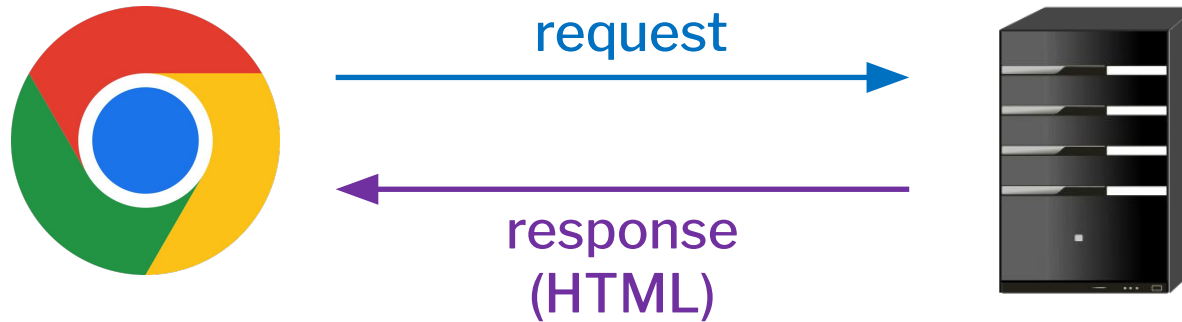


Browser Operation



- HTML page can load JavaScript
 - starter code's `index.html` includes `index.tsx`
- Each time the page loads, browser executes `index.tsx`

Development Environment



- “`npm run start`” starts a server that the browser can contact
 - server is running on this machine (localhost)
 - (more on servers later this quarter...)
- This server returns `index.html` but adds compiled JS into the page
 - also adds code to reload if the source code is changed!

Custom Server (Review)

- **Query Parameters** (e.g. ?name=Fred) in requests (req)

```
const F = (req, res) => {
  if (req.query.name === undefined) {
    res.status(400).send({response: "Missing
`name`"});
    return;
  }
  res.send(`Hi, ${req.query.name}`);
}
```

HTTP Terminology (Review)

- **HTTP** Request include:
 - URL: Path and query parameters
 - Method: Get/Post
 - Get is used to read data on the server (can paste raw url in browser and get result back)
 - Post is used to change data on the server (cannot paste raw url in browser)
 - Body (for Post only)
 - used for sending large or non-string data to server
- **HTTP** Response Status Codes include:
 - 200 (ok)
 - 400-499 (Client error)
 - 500-599 (Server error)


Bug Journalling


Practice Log: <https://comfy.cs.washington.edu/service/hw1-practice>

Homework Log: <https://comfy.cs.washington.edu/service/hw1>

Debugging Log

New Entry

Start Time: 

End Time: 

Breaks: (in minutes) — subtracted from end time - start time

Failure

Briefly describe the program behavior that looks wrong:

Experiments

[New Experiment](#)

Error

Did you find the error?

[Add](#)

[Cancel](#)

Bug Journaling

- **Before you start debugging:** Add a bug log entry and document the error and start time of when you started debugging
- **After you have finished debugging:** Fill out the log entry with your findings from the debugging session. This will include:
 - Any experiments you ran to find the bug
 - If you found the code error (if so, where)
 - If you found the defect (if so, where)
 - If type checking would have prevented the bug and why
- Be sure to click `Save Log` after adding an entry as to not lose progress
- 1 bug should be 1 log entry. Do not split a single bug into multiple log entries.
- For initial HWs, you must debug for at least 4 hours and at most 6 hours regardless of whether you fix the bug or not

Debugging Tips

- Check the easy stuff (save files, restart server)
- Create the smallest input that re-creates the error
- Look for common small errors
 - using ==
 - misnaming variables. *Note:* Type checkers will help you catch misnamed functions/variables, not just incorrect types
 - incorrect arguments
- Use Scientific Method to gain understanding of problem
- Talk through the problem with someone