CSE 331
Software Design & Implementation

Fall 2023

Section 1 – HW1 and Tools

Administrivia

- HW1 released tonight, due next Wednesday
- No more than **one** late day per assignment
- 4 late days in total
- If you haven't done the software setup yet, please look at the email sent last night!

Welcome

- Let’s all introduce ourselves:
  - Name and pronouns
  - Year
  - What other classes you are taking this quarter
  - Something fun you did over summer break

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/cse331-23au-materials/sec-fib.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 sec0

```
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

- We will want the **comfy-tslint** extension
  - verifies that our code satisfies the 331 coding convention
  - running `npm run lint` will also do this

NPM Start

- **NPM**: package manager (comes with Node)

- Use this command to start

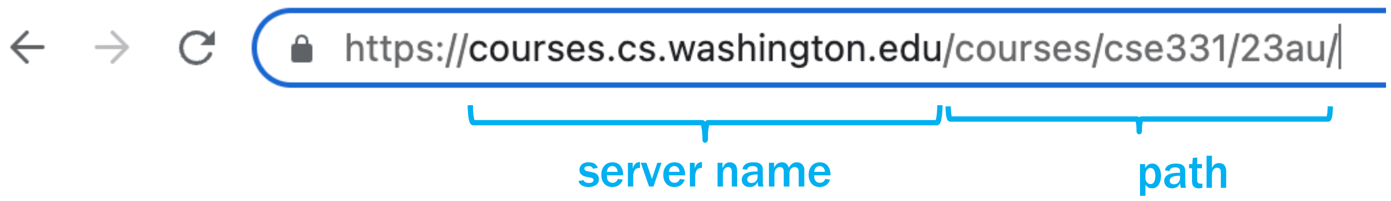
```
npm run start
```

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

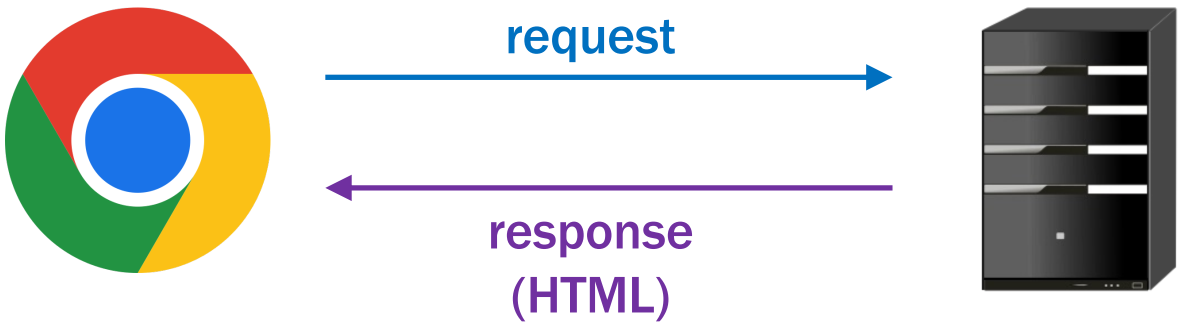
```
http://localhost:8080
```

Browser Operation

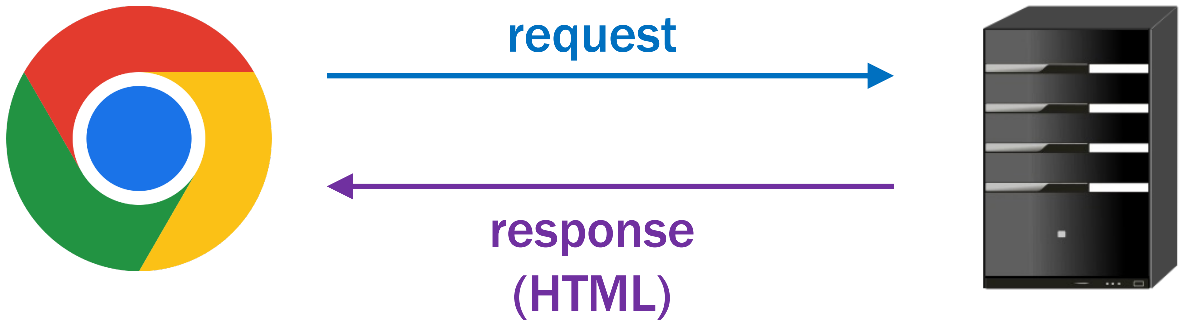
- Browser reads the URL to find the server to talk to



- Contact the given server and request the given path:



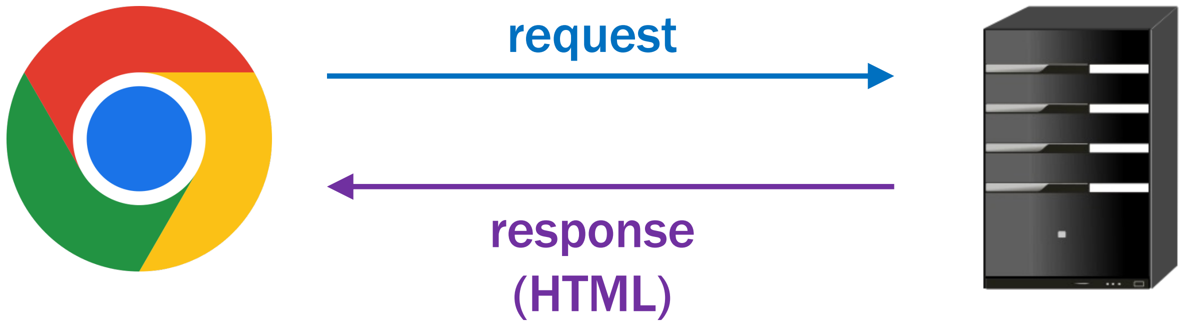
Browser Operation



- 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!

Starter Code Demo

- Starter code prints out the current date and time

```
console.log(new Date());
```

- Find the Developer Console in Chrome
    - find the date that was printed
- Try reloading the page a few times
    - verify that a new date is printed out each time

Global Variables

- The `document` object stores the HTML tree
- The `window` object has information about the browser window
  - `window.location` stores information about the URL
  - if URL = `https://mail.google.com/mail/u/0/?zx=ABCD#inbox`

`window.location.hostname`	`“mail.google.com”`
`window.location.pathname`	`“/mail/u/0”`
`window.location.search`	`“?zx=ABCD”`
`window.location.hash`	`“#inbox”`

Search String

```
https://mail.google.com/mail/u/0/?zx=ABCD#inbox

window.location.hostname      “mail.google.com”
window.location.pathname      “/mail/u/0”
window.location.search        “?zx=ABCD”
window.location.hash          “#inbox”
```

- the hostname tells the browser what server to contact
- the pathname is the HTML file that is requested
- the search string is effectively an **argument** to that file
  - same code is executed in the browser
  - but code can behave differently due to different parameters
- the hash is not sent to the server (and we won’t use it this quarter)

Query Parameters

- Search string is a list of name=value pairs, separated by “&”s
  - these are often called “query parameters”
  - this example has 3 parameters (called a, c, and e)

```
…?a=b&c=d&e=f
```

- JavaScript includes built-in tools for parsing the search string

```
const params = new URLSearchParams(window.location.search);
console.log(params.get(“a”));  // prints “b”
```

  - params.get returns a string or null (why?)

Problem 1

- Change `index.tsx` to look for a parameter called “n”
  - if it is found, print the n-th Fibonacci number to the console
    - import fib function from fib.ts
  - if it is not found, then print an error message
  - if it is found but is not a non-negative integer, then print an error

Problem 2

- Let’s put something on the screen this time!

- Change the code to display an HTML paragraph
  - can be done something like this

```
const elem: HTMLElement | null = document.getElementById(‘main’);
if (elem !== null) {
  const root: Root = createRoot(elem);
  root.render(<p>Fibonacci number 5 is 8.</p>);
}
```

  - see the worksheet for the imports you will need

- Call to document.getElementById finds an HTML tag by id=“..” attribute
  - index.html includes a tag with id=“main”

HTML Literals

- JS / TS allow HTML literals in the code

- Like strings, you can substitute variable values into the HTML
  - uses {..} rather than ${..} (like `..` syntax)
  - can substitute into the text like this

```
const name = “Fred”;
root.render(<p>Hi, {name}!</p>);  // says Hi, Fred!
```

  - can also substitute attribute values

Problem 3

- Change the code to assume n = 0 if it was not provided
- Change the HTML to include links to pages for the prev/next Fib
- Use an “A” tag to make a link, e.g.:

```
<p>Show <a href="/index.html?n=3”>previous</a></p>
```

  - need to calculate the URL in a variable
  - then include it with `<a href={prevLink}>..</a>`
- Can only render one tree, so wrap multiple <p>s in a <div>
- **Challenge**: only show previous link if n > 0