

---

# CSE 331

## Software Design & Implementation

Spring 2025

Section 2 – HW2 and Browser Operations

# Administrivia

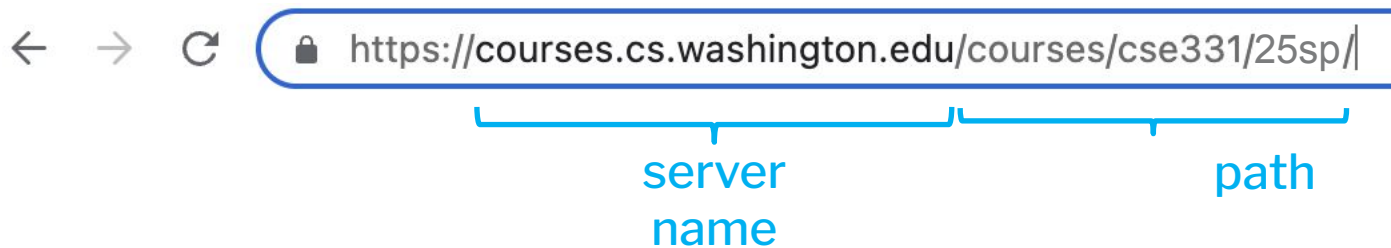
---

- **Homework 2:**
  - Due Wednesday, April 16th @ 11pm
  - Released this evening

# Browser Operation (Review)

---

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

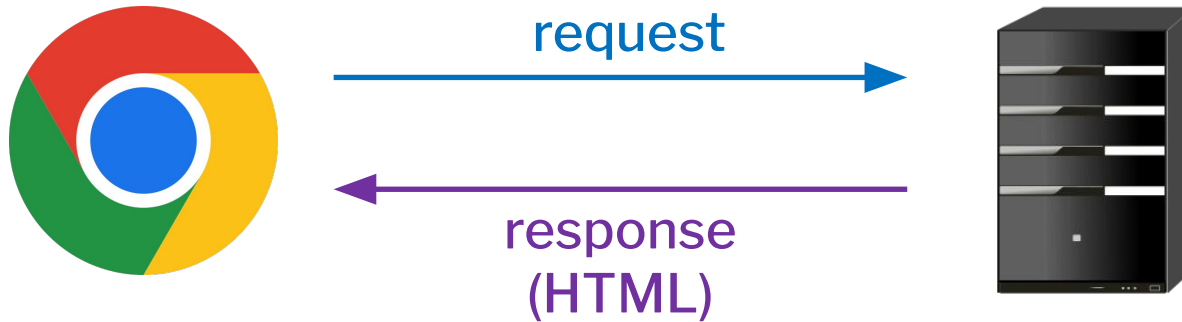


- Contact the given server and request the given path:



# Browser Operation (Review)

---



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

# React

---

- UI library with syntax called JSX:

```
const x = <p>Hi there!</p>;
```

- Breaks interface into components

```
class HiElem extends Component {  
  constructor(props) {  
    super(props);  
  }  
  render = () => {  
    return <p>Hola, Kevin!</p>;  
  };  
}
```

- Must have a single root tag (must be a tree)

e.g., cannot do this: `return <p>one</p><p>two</p>;`

# React - Event Handler (Review)

---

- Passing method to be called as argument:

```
<button onClick={this.doEspClick}>Esp</button>
```

- Creating event handler:

```
doEspClick = (evt) => {  
  this.setState({lang: "es"};  
};
```

- Must call `setState` to change the state (*do not* directly modify `this.state`)

# TypeScript Review

---

- TypeScript includes declared types for variables
- Compiler checks that the types are valid
  - extremely useful!
  - produces JS just by *removing* the types
- If you leave off the type, TS will try to guess it

# Basic Data Types (Review)

---

number

bigint

string

boolean

null

undefined

Object (record types)

Array (e.g., `string[]` as in Java)

unknown (could be anything)

**any** (turns off type checking — do not use!)

literal values (ex “foo” or “foo” | “bar”)

# Creating New Types (Review)

---

- **Union Types** `string | bigint`
  - can be either one of these
- **Record Types** (creator picks the names) :
  - anything with *at least* fields “x” and “s” (could have more fields)  

```
const p: {x: bigint, s: string} = {x: 1n, s: 'hi'};  
console.log(p.x); // prints 1n
```
- **Tuple Types** (user picks the names): `[bigint, string]`
  - give names to the parts (“destructuring”) to use them  

```
const [x, y] = p;  
console.log(x); // prints 1n
```

# Small Note On `<select>`

---

```
<select value={color} onChange={handleColorChange}>
  <option value="red">Red</option>
  <option value="blue">Blue</option>
  <option value="green">Green</option>
</select>
```

- **Controlled Component:**

- the “`value`” attribute, which is the color in this example, determines the current selected `option`.

- **Event Handler:**

- The `onChange` function, in this case `handleColorChange` updates the color value when a new option is chosen.

- **Options:**

- Each `<option>` tag provides a selectable item with a specific `value`.
- When you select one of the `<option>` tags, the `value` of that tag is then set to the `value` in the `<select>`, and is used by the `event handler`

# Bug Journaling

---

- <https://comfy.cs.washington.edu/service/hw2-practice>
- Make sure to save and wait for website to say “Saved” before closing
- Copy entire line of code into bug journal (not just line number)

## Mutation

Was this failure caused by mutating something that should not have been mutated?  

Briefly explain why or why not:

Array declared const was not intended to be mutated.

# Bug Journal Clarifications

---

- **Experiments:** *Any* steps taken to find the bug
  - be sure to document entire debugging process (can and most likely will include dead end experiments)
  - experiments should typically help inform you about the next experiment until you find the actual bug
- **Mutation:** This means mutating something that *should not have* been mutated (this does not mean mutating something incorrectly)