CSE 331
Software Design & Implementation

Spring 2023

Section 6 – Imperative Programming II

Administrivia

- HW6 released later today
  - Due Wednesday (5/10) @ 11:00pm

Arrays – Review

- Allows easy access both `A[0] and A[n-1]`. Where `n = A.length`
  - Bottom-up loops are now easy
- However, when we write "`A[j]`", we must also check $0 \le j < n$
  - New possibilities for bugs
  - TypeScript will not help us with this
- **Array Concatenation** – define operation "⧺" as array concatenation (makes clear arguments are arrays, not numbers)
- **Following properties hold for any arrays, A, B, C:**
  - A ⧺ [] = A = [] ⧺ A (“identity”)
  - A ⧺ (B ⧺ C) = (A ⧺ B) ⧺ C (“associativity”)

Mutating Arrays – Review

- **Assigning to array elements changes know state:**

  {{ A[j-1] < A[j] for any $1 \le j \le 5$ }}

  ```
  A[0] = 100;
  ```

  {{ A[0] = 100 and A[j-1] < A[j] for any $2 \le j \le 5$ }}

- **Can add to the end of an array:**

  ```
  A.push(100);
  ```

  {{ $A = A_0 \mathbin{+\!\!+} [100]$ }}

- **Can remove from the end of an array:**

  ```
  A.pop();
  ```

  {{ $A = A_0[0 \ldots n-1]$ }}

  A has one fewer element than before

Loop Invariants with Arrays – Review

- Heuristic for loop invariants: weaken the postcondition
  - Inv is just a **weakening** of the Post
  - Inv is simple an assertion that allows the post condition as a special case
  - Must also allow states that are easy to prepare
- Ex:

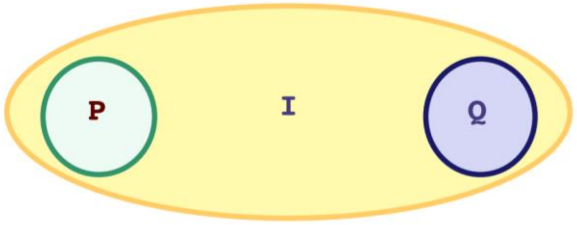
{{ **Inv**: $s = \text{sum}(A[0 .. j-1])$ and $j \leq A.\text{length}$ }} **sum of array**
{{ **Post**: $s = \text{sum}(A[0 .. n-1])$ }}

{{ **Inv**: $\text{contains}(A[0 .. j-1], x) = F$ }} **search an array**
{{ **Post**: $\text{contains}(A[0 .. n-1], x) = F$ }}

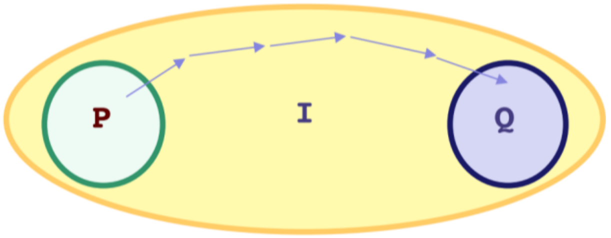
Loop Invariants with Arrays – Review

- Loop invariants allow for both start and stop states

```
{{ P }}
{{ Inv: I }}
while (cond) {
  S
}
{{ Q }}
```



- **Algorithm Idea:**
  - How you will get from start to stop state (e.g. j=j+1)
  - What partial progress looks like (e.g. loop invariant)



Specifying Functions – Review

- By default, no parameters are mutated
  - Must *explicitly* say that mutation is possible (default is not)

```
    /**
    * Reorder A so that the numbers are in increasing order
    * @param A array of numbers to be sorted
    * @modifies A
    * @effects A contains the same numbers but now in
increasing order
    */
    quickSort(A: number[]): void { ... }
```

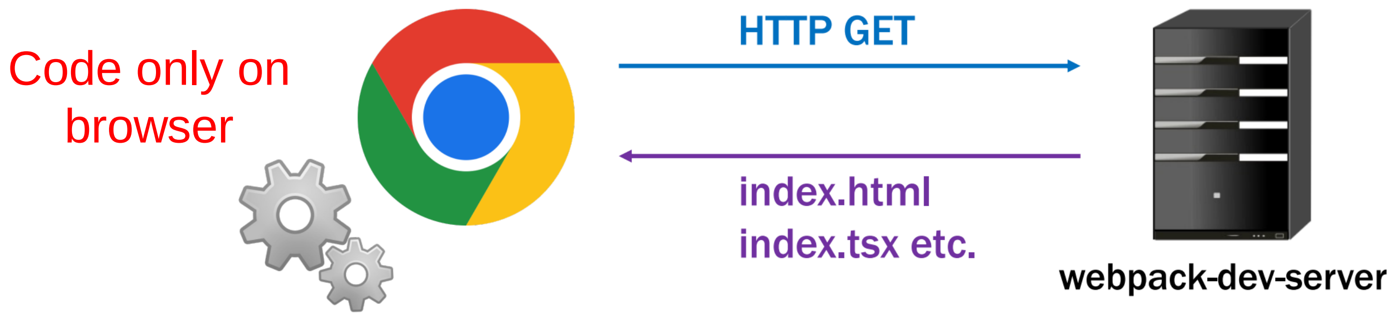
  - Anything that might be changed is listed in `@modifies`
    - Not a promise to modify it – A could already by sorted!
  - `@effects` gives the promise about result after mutation
    - Like `@returns` but for mutated values, not return values.

Client-Side vs Server-Side – Review

- **Client-Side JavaScript**
  - Code so far has run inside the browser
    - webpack-dev-server handles HTTP requests
    - Sends back our code to the browser
  - In the browser, executes code of index.tsx
- **Server-Side JavaScript**
  - Can run code in the server as well
    - Returns different data for each request (HTML, JSON, etc.)
  - Can have code in *both* browser and server

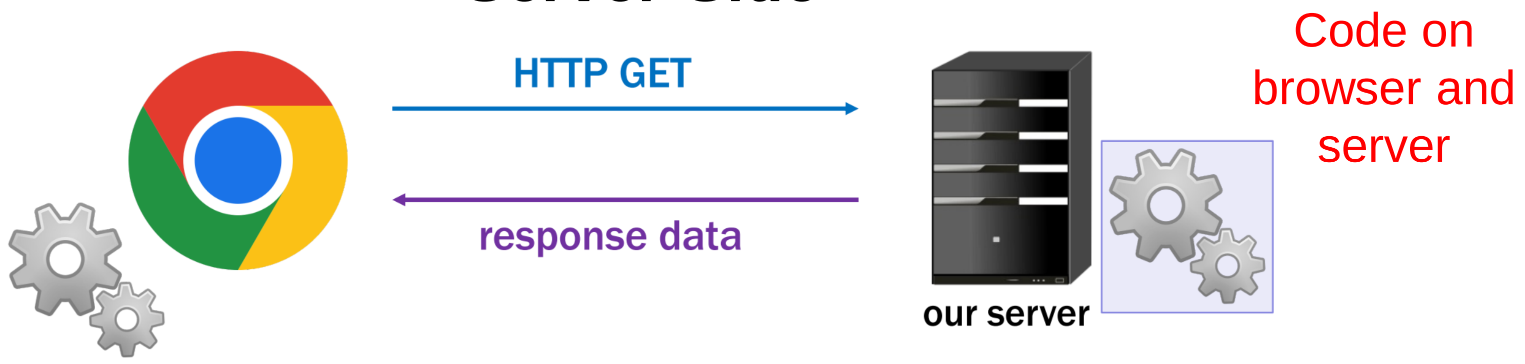
Client-Side vs Server-Side – Review

Client-Side



vs.

Server-Side



Custom Server

```
function F(req: Request, res: Response): void {
    const name: string | undefined = req.query.name;
    if (name === undefined) {
        res.status(400).send(“Missing ‘name’”);
        return;
    }
    res.send({message: `Hi, ${name}`});
}

const app = express();
app.get(“/foo”, F);
app.listen(8080);
```

- Set status to 400 to indicate a client error (Bad Request)
- Set status to 500 to indicate server error
- Default status is 200 (OK)

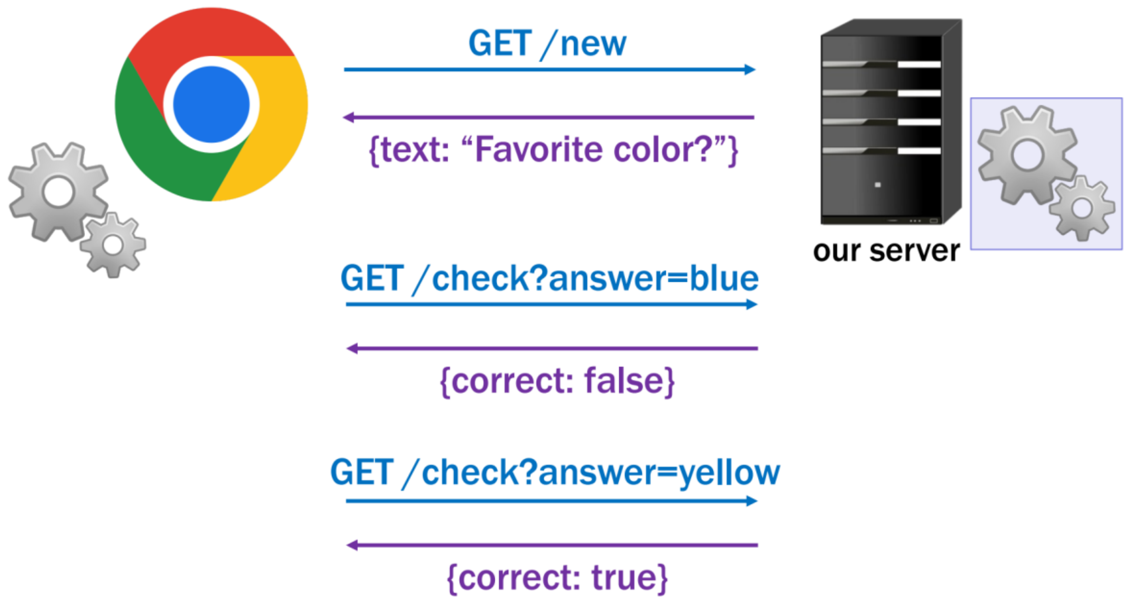
- send of string returned as Text/HTML
- send of record returned as JSON
  - client will then be able to parse it into a JS record
  - record can contain string, number, null, arrays, records (no undefined)

- Request for https://localhost:8080/foo will call F
- Mapping from “/foo” to F is called a “route”
- Can have as many routes as we want (with different URLs)

Custom Server

Server-Side JavaScript

- Apps will make sequence of requests to server
  - e.g., in HW6:



GET /new

{text: “Favorite color?”}

GET /check?answer=blue

{correct: false}

GET /check?answer=yellow

{correct: true}

Question 2a

```
// Returns an index i such that words[i] === w or -1 if no such index exists
function findWord(words: string[], w: string): number
```

(a) Fill in the missing parts so that the code is correct with the invariant provided.

```
let i: number = ____________; // fill this in
{{ Inv: there is no index $0 \le j \le i$ such that words$[j] = w$ }}
while (____________) {          // fill this in

                                // fill this in


   i = i + 1;
}
{{ there is no index $0 \le j \le$ words.length $- 1$ such that words$[j] = w$ }}
return -1;
```

Question 2b

(c) Fill in the missing parts so that the code is correct with the invariant provided.

This version leaves the line where `i` is incremented as before, but now the invariant is changed.

```
let i: number = ____________; // fill this in
{{ Inv: there is no index $0 \le j \le i - 1$ such that words$[j] = w$ }}
while (____________) {        // fill this in
   i = i + 1;

                              // fill this in

}
{{ there is no index $0 \le j \le$ words.length $- 1$ such that words$[j] = w$ }}
return -1;
```

Question 3

The following function takes an array of characters and returns a new array that is the same except that all space characters are removed (remember that there are not `chars` in TypeScript, so we represent characters with `strings` of length 1.):

$$\begin{aligned}
\textbf{func}\ \text{del-spaces}([\,]) &:= [\,] \\
\text{del-spaces}(L + [\text{`` ''}]) &:= \text{del-spaces}(L) \\
\text{del-spaces}(L + [c]) &:= \text{del-spaces}(L) + [c] \qquad \text{if } c \neq \text{`` ''}
\end{aligned}$$

(a) Write a specification for a function with the following signature that mutates the array passed in.

```
function delSpaces(str: string[]): void
```

(b) How can we weaken that post-condition into something that is easy to make true at the top of the loop? What exit condition will tell us that the loop holds at the end.

Question 3 continued

(c) Implement the loop with the invariant above.

Be sure to document your loop invariant. Your code must be correct with that invariant.

Question 4

In this problem, we will write some code that runs in the server rather than in the browser (the "client"). Inputs will be passed in the query parameters as usual, but the response will be a JavaScript object serialized into text format ("JSON") and sent back to the browser. This is the most convenient format for a JavaScript client to parse back into an object.

To get started, check out the starter code using the command

```
git clone https://gitlab.cs.washington.edu/cse331-23sp-materials/sec-chatbot.git
```

Installing the modules with `npm install -no-audit`, then start the server with the command `npm run start` and then point your browser at `http://localhost:8080`. You should see a question on the screen. Unfortunately, it always asks the same question "What is your favorite color?" We would like it to return some more interesting questions.

Some more interesting questions are provided in `trivia.ts`. The `TRIVIA` list contains a number of records, each of which has a question and the answer. (Feel free to add more.)