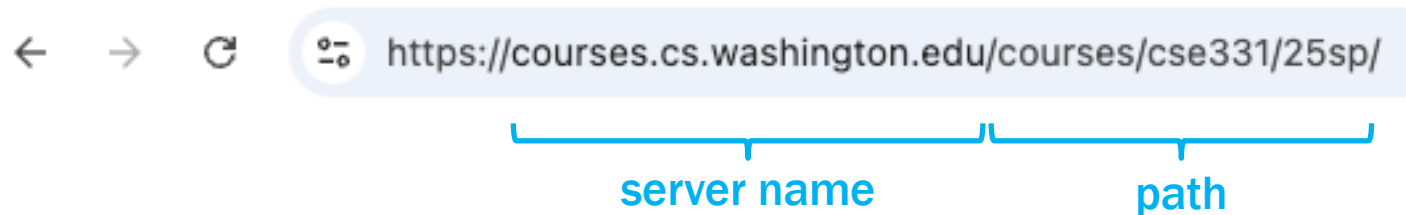


# HTTP Servers

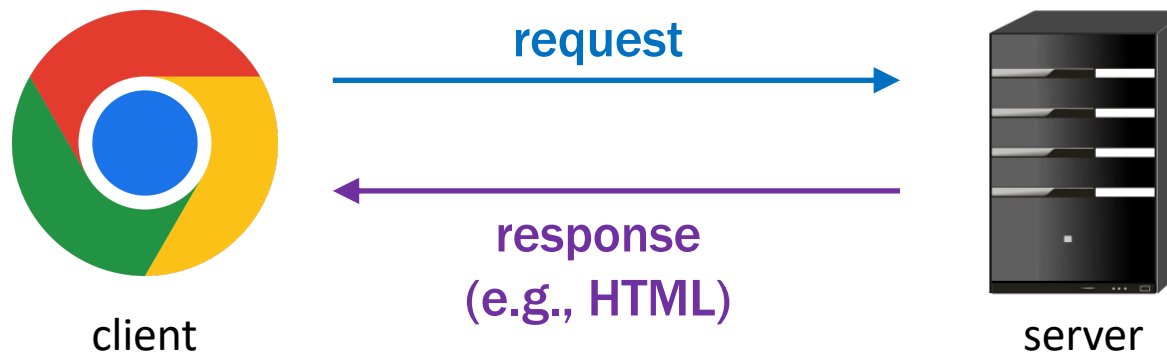
# Browser Operation

---

- Browser reads the URL to find what HTML to load



- Contacts the given server and asks for the given path



# URL Parts

---

- URLs have more parts than just server and path:



- **Server name** identifies the computer to talk to
  - uses the HTTP(S) protocol
- **Conceptually:**
  - **path** identifies code to execute on the server
  - **search** string is **input** passed to that file when run
  - (**fragment** will not be important for us)

# Query Parameters

---

- **Search string can pass multiple values at once**
  - we call these “query parameters”
- **Each parameter is of the form “name=value”**
  - no spaces around the “=”
- **Multiple values are placed together with “&”s in between**

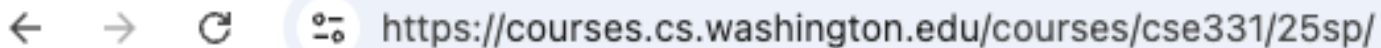
`?a=3&b=foo&c=Matt`

- encodes three query parameters: a is “3”, b is “foo”, c is “Matt”

# HTTP Terminology: Requests

---

- HTTP **request** includes
  - **URL:** path and query parameters
  - **method:** GET or POST
    - GET is used to *read* data stored on the server (cacheable)
    - POST is used to *change* data stored on the server
  - **body (for POST only)**
    - useful for sending large or **non-string** data with the request
- **Browser issues a GET request when you type URL**



← → ↻ ⓘ <https://courses.cs.washington.edu/courses/cse331/25sp/>

# HTTP Terminology: Responses

---

- **HTTP response** includes
  - **status code:** 200 (ok), 400-99 (client error),  
or 500-99 (server error)  
was the server able to respond
  - **content type:** text/HTML or application/JSON (for us)  
what sort of data did the server send back
  - **content**  
in format described by the Content Type
- **Browser expects HTML to display in the page**
  - we will always send JSON or text to the browser

# Example App: Interface

---

## Trivia

Question  
Answer

What is your favorite color?

Submit

User types “blue” and presses “Submit”...

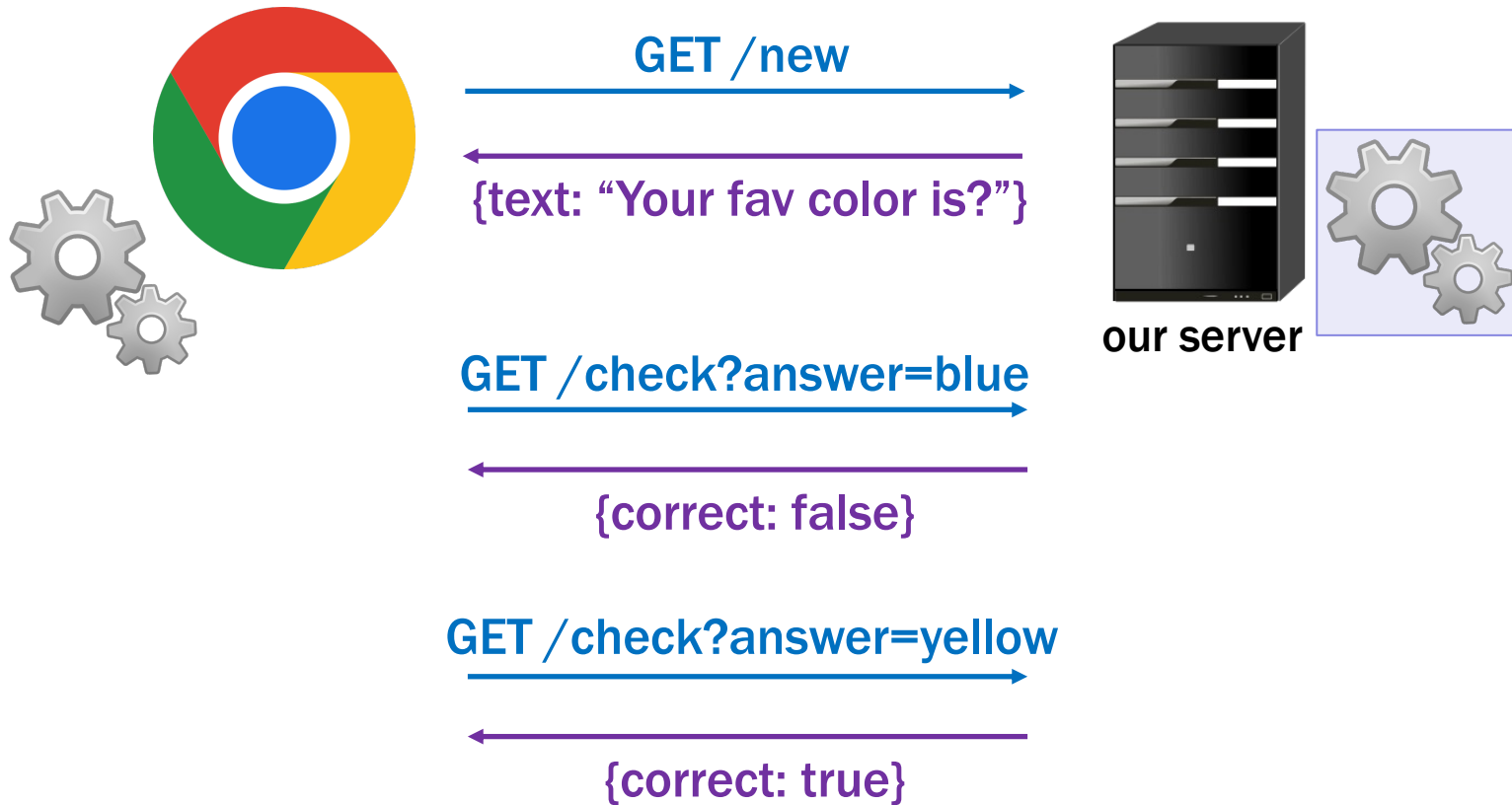
Sorry, your answer was incorrect.

New Question

# Example App: Requests and Responses






---

Apps will make sequence of requests to server



# “Network” Tab Shows Requests

---

Name	Status
 localhost	200
 qna.js	200
 new	200
 favicon.ico	200
 check?index=0&answer=blue	304

- **Shows every request to the server**
  - first request loads the app (as usual)
  - “new” is a request to get a question
  - “check?index=0&answer=blue” is a request to check answer
- **Click on a request to see details...**

# “Network” Tab Shows Request & Response

Name	× Headers	Preview	Response	Initiator	Timing
localhost	▼ General				
qna.js	Request URL: http://localhost:8080/new				
<b>new</b>	Request Method: GET				
favicon.ico	Status Code: 🟢 200 OK				
check?index=0&answer=blue	Remote Address: [::1]:8080				
5 requests   8.9 kB transferred		Referrer Policy: strict-origin-when-cross-origin			

Name	× Headers	Preview	Response	Initiator	Timing
localhost	1	{"index":0,"text":"What is your favorite color?"}			
qna.js					
new					
favicon.ico					
check?index=0&answer=blue					
5 requests   8.9 kB transferred		{}			

# JSON

---

- **JavaScript Object Notation**

- text description of JavaScript object
- allows strings, numbers, null, arrays, and records
  - no undefined and no instances of classes
  - no `'..'` (single quotes), only `".."`
  - requires quotes around keys in records

- **Translation into string done *automatically* by send**

```
res.send({index: 0, text: 'What is your ...?'});
```

Name	×	Headers	Preview	Response	Initiator	Timing
localhost	1			<code>{"index":0,"text":"What is your favorite color?"}</code>		
qna.js						
new						