Intro to APIs and Promises

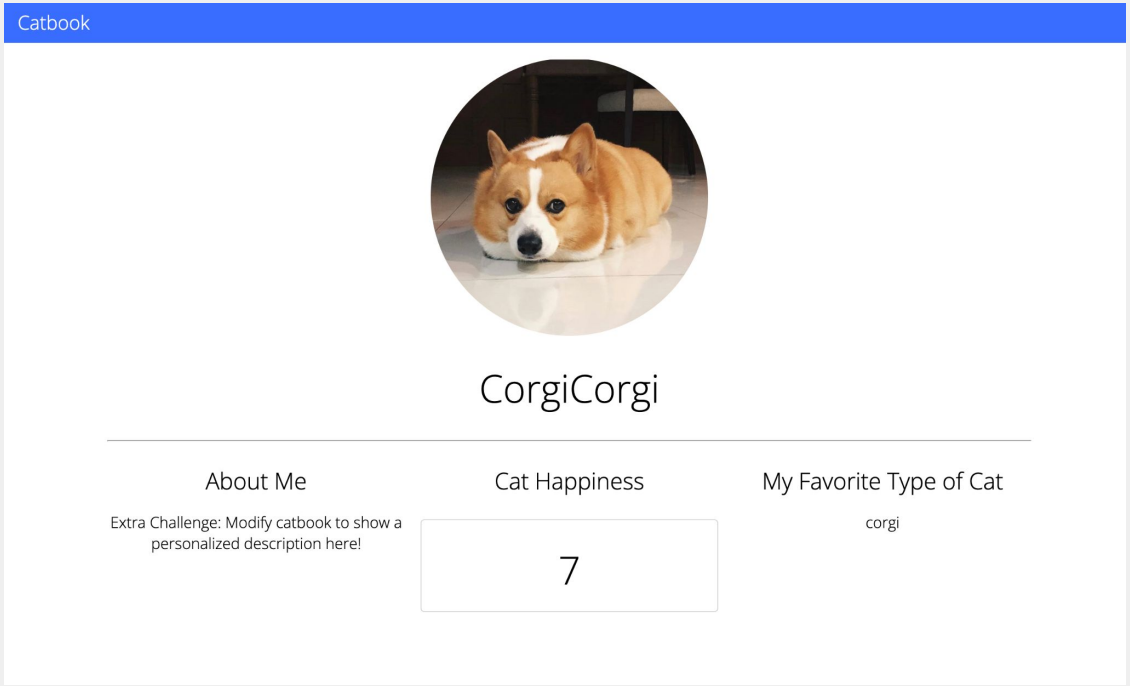
Helen Yang and Jay Hilton

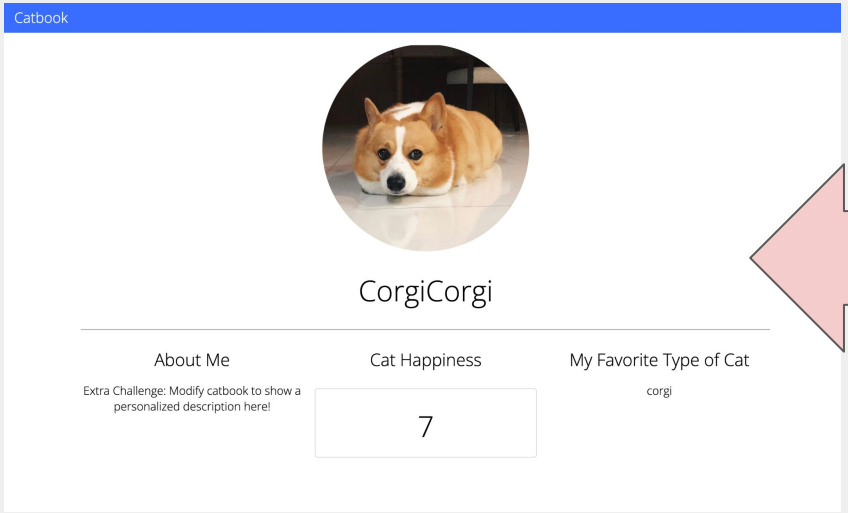

Sponsored by
Mobi

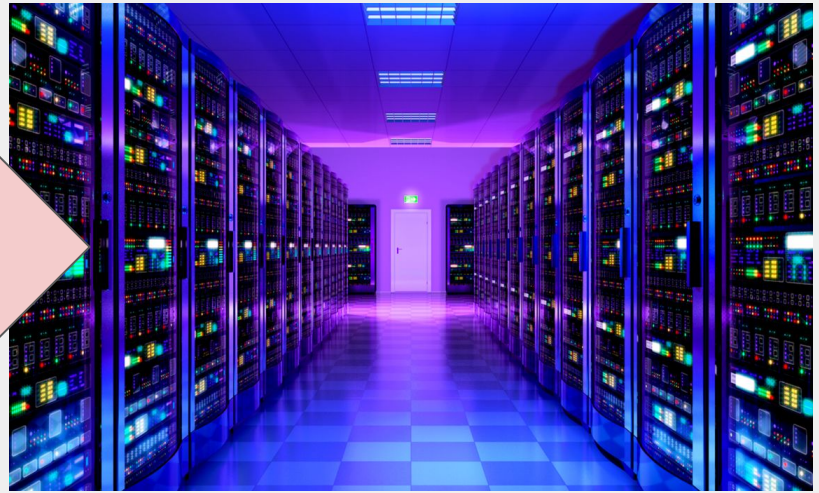
Time to explore something new!

At the moment our catbook is static, it doesn't really change with user data

Why is this? What's missing?





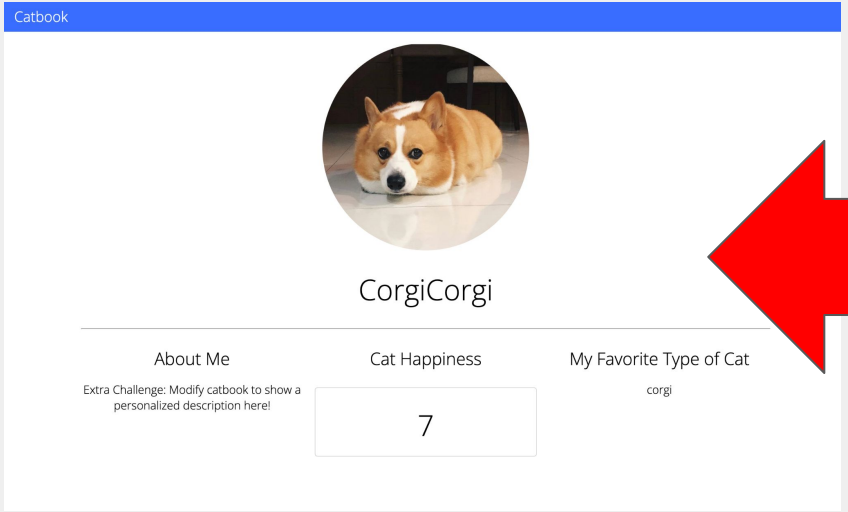


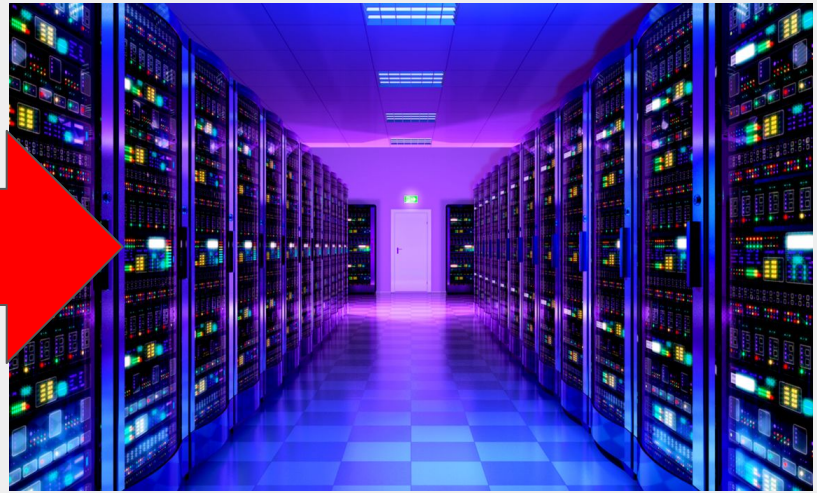
Frontend

Part that users directly interact with

Backend

- Data storage
- Data manipulation





Frontend

Part that users directly interact with

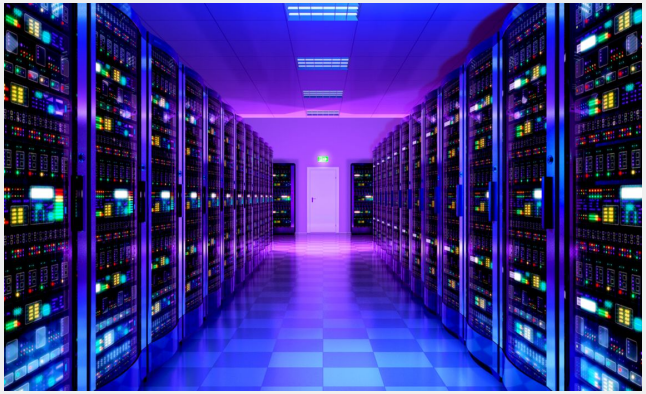
Backend

- Data storage
- Data manipulation

Client and Server

Requests

Responses



You and your teammates' browsers are **clients**

The device that holds your site's information is the **server**

How do we send information?

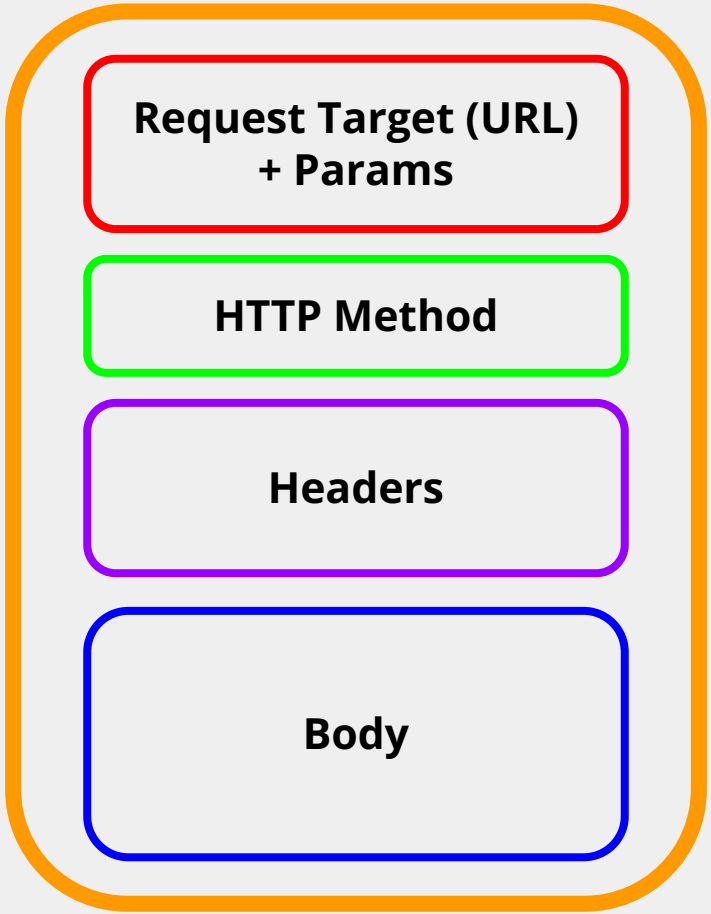
- How can you access information from servers miles away?
- Sadly email is not an acceptable answer.
- It has to do with a specific type of request/response


https://www


http://www

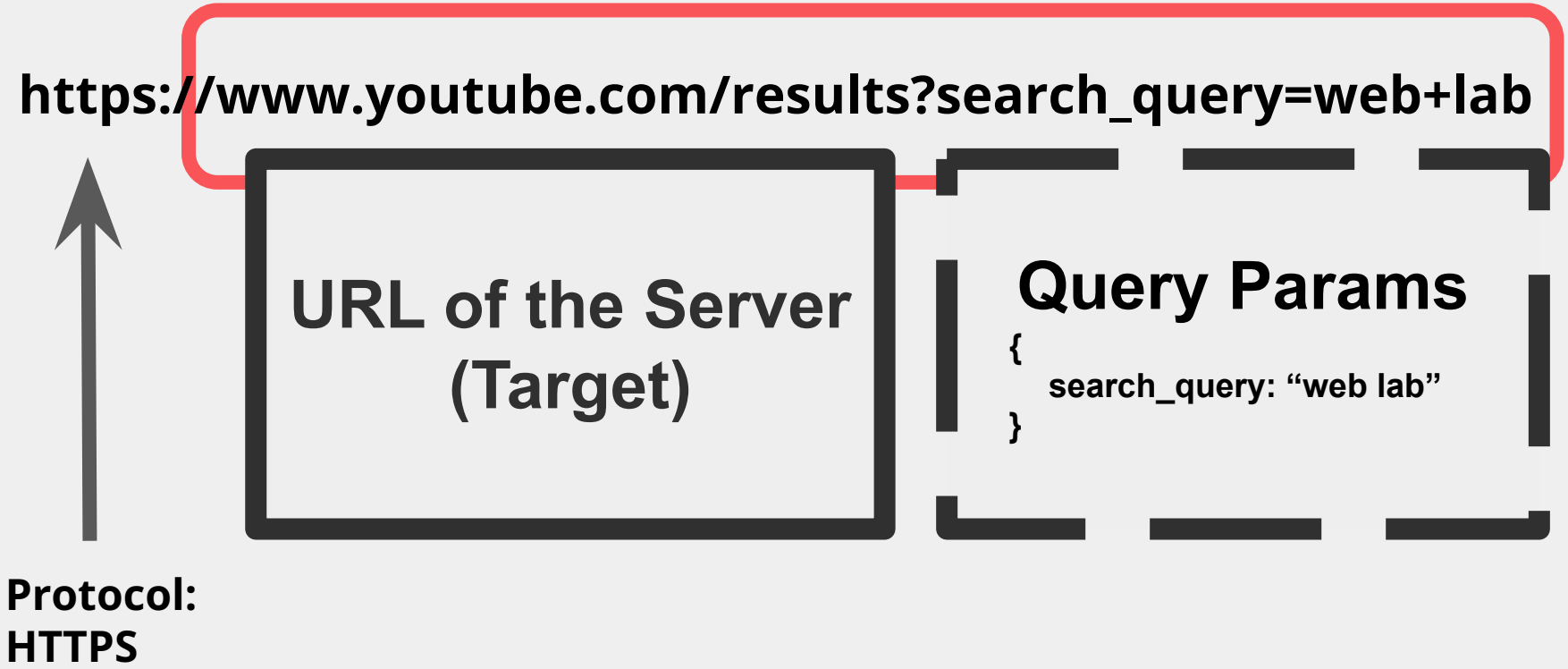
What is HTTP?

- It's short for “Hypertext Transfer Protocol”
- You’ve been using HTTP requests and responses this whole time!
- HTTPS is HTTP Secure
    - HTTP Secure uses encryption for secure communication over a computer network.



HTTP Request

Target (URL) and Query Params



HTTP(S) Methods

Indicates the Desired Action

- GET, well, gets data
- POST sends data, often creates change or new data
- PUT replaces data
- DELETE, well, deletes data
- You can find the rest here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

HTTP(S) Methods

Indicates the Desired Action

- **GET**, well, gets data
- **POST** sends data, often creates change or new data
- PUT replaces data
- DELETE, well, deletes data
- You can find the rest here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

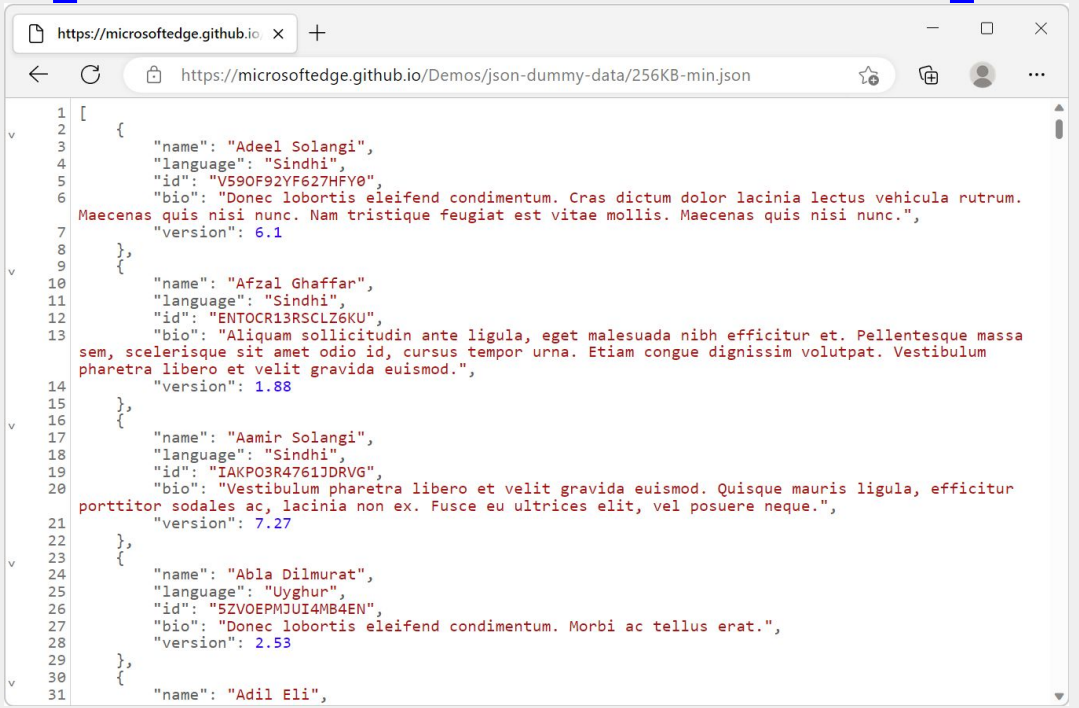
Request Headers

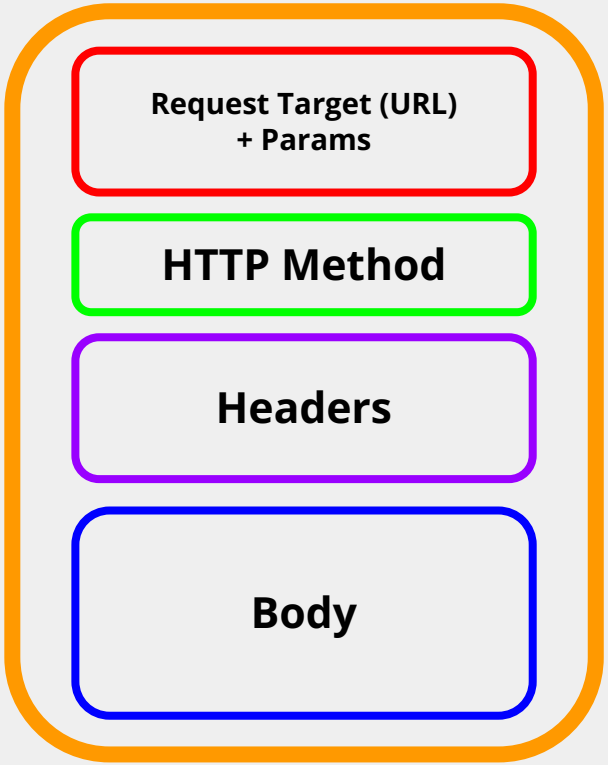
provides context for the HTTP request

aka “fancy stuff™”

```
GET /home.html HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9;
rv:50.0) Gecko/20100101 Firefox/50.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.
8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/testpage.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Mon, 18 Jul 2016 02:36:04 GMT
If-None-Match: "c561c68d0ba92bbeb8b0fff2a9199f722e3a621a"
Cache-Control: max-age=0
```

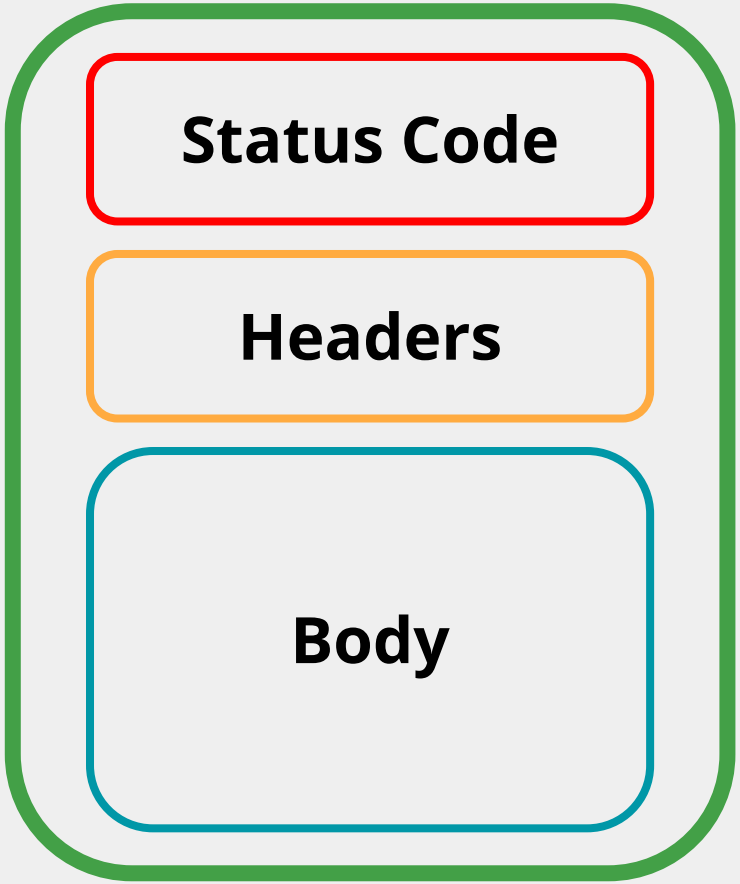
Request Body





Request

Request

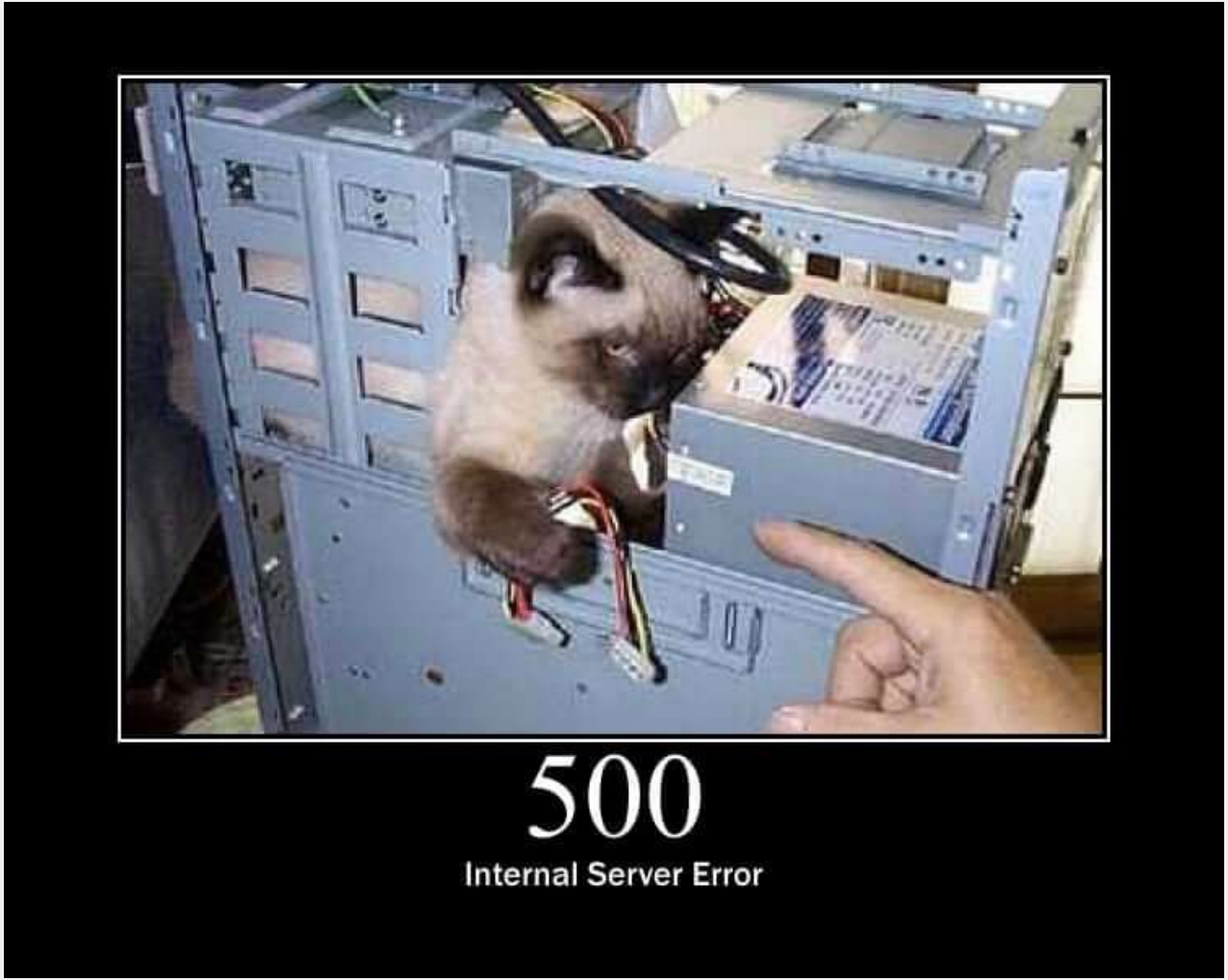


Response

Status Codes

The Dreaded 404


400
Bad Request


500
Internal Server Error

200

OK

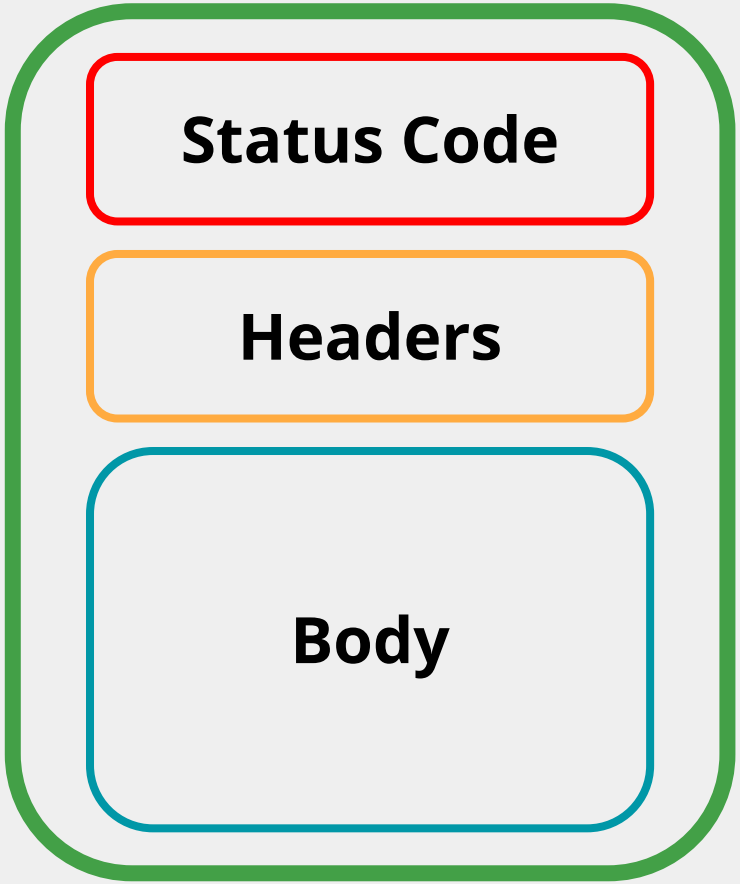
Status Codes

- 1xx- informational
- 2xx- you succeeded
- 3xx- redirect
- 4xx- you did something wrong
- 5xx- server did something wrong
- https://www.restapitutorial.com/httpstatuscodes.html

Status Codes

- 1xx- informational
- 2xx- you succeeded
- 3xx- redirect
- 4xx- you did something wrong
- 5xx- server did something wrong
- https://www.restapitutorial.com/httpstatuscodes.html

Request



Response

Response Headers

Information about the response

e.g. content-type, content-length

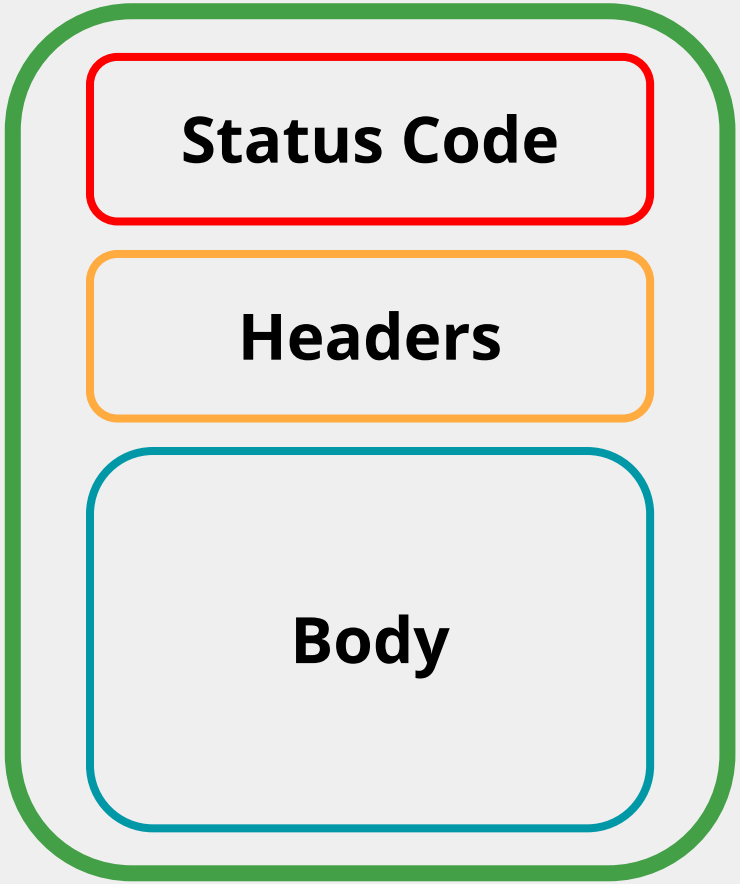
again aka “fancy stuff”

Response Body

The Response Data

E.g. Information about a user, if you are GET-ing their data

Usually JSON format



Response

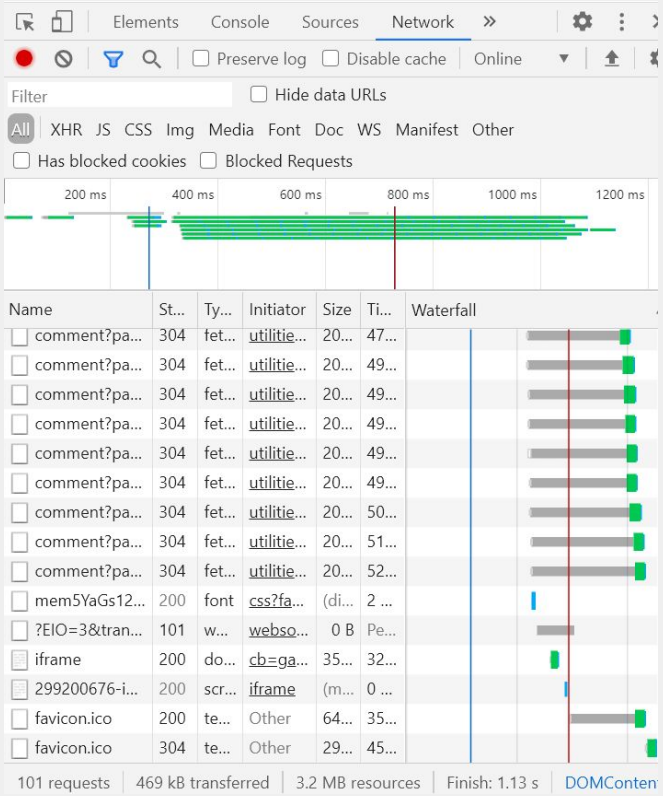
Questions?

Your requests are just out of sight...

- Open up some random page (maybe try catbook!)
- Open the developer section with Ctrl-Shift-i or Cmd-Shift-i or whatever applicable shortcut/method
- You should see a “Network” tab
- If you open it and refresh the page, you should see all the requests the page made just to load the site!

Here's what that looks like for Catbook

weblab.is/example



Typing a URL into any browser sets off a GET Request that often responds with HTML, CSS, and JS

Let’s make a couple requests…

- And I mean a couple.

GET -

http://weblab.mit.edu/yeet

GET -

http://www.ascii-art.de/ascii/my/giraffe_s.txt

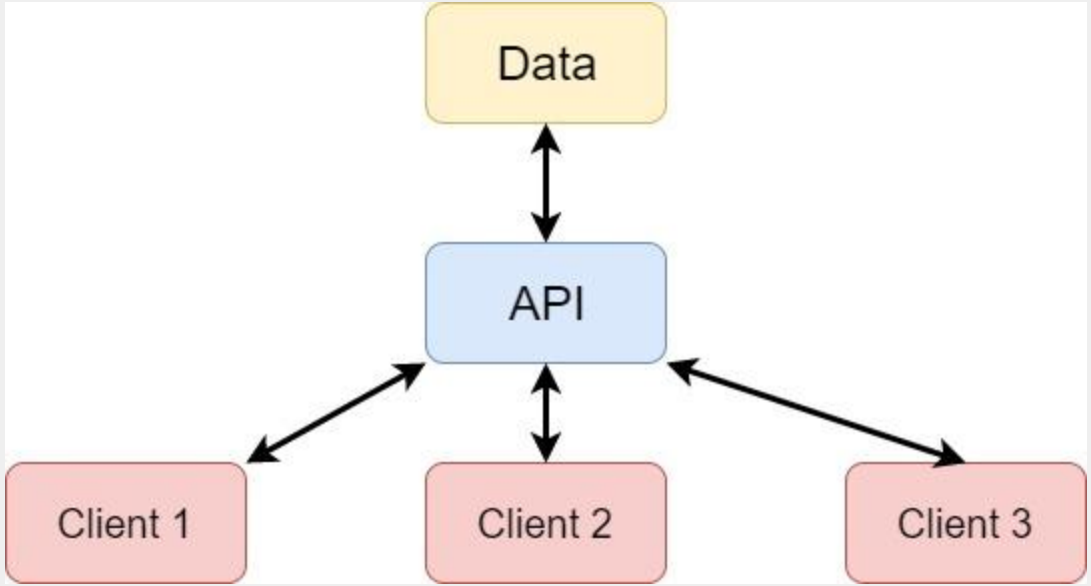
APIs

- “Application Program Interface”
- Simply a set of endpoints a service allows you to make requests to
- Companies such as Google, Amazon, and Facebook all provide APIs to allow others to use their services

The Purpose of an API

- You need to access data
- You're not going to access data on servers directly, that's both extremely inconvenient and a security nightmare
    - What if we want to access our own private data?
        - Shouldn't be able to access others' private data: API keys / authentication!
- APIs provide structured places (endpoints) for you to send requests to!
- Now you can do a bunch of things, simply by asking the right people very nicely.

The Purpose of an API



What does this API output?

- GET /api/add

What does this API output?

- GET /api/add

Where does the input go? URL or body?

What does this API output?

- GET /api/add

Where does the input go? URL or body?

URL (query parameters!)

What does this API output?

- GET /api/add
  - Input: { a: number, b: number }
  - Output: { result: number }
    - *Result will be mapped to a + b*

Where does the input go? URL or body?

URL (query parameters!)

What does this API output?

- GET /api/add
  - Input: { a: number, b: number }
  - Output: { result: number }
    - *Result will be mapped to a + b*
- GET /api/add?a=5&b=8
  - What does this output?

Where does the input go? URL or body?

URL (query parameters!)

What does this API output?

- GET /api/add
  - Input: { a: number, b: number }
  - Output: { result: number }
    - *Result will be mapped to a + b*
- GET /api/add?a=5&b=8
  - What does this output?
    - { result: 13 }
    - { 13 }
    - 13

Where does the input go? URL or body?

URL (query parameters!)

What does this API output?

- GET /api/add
  - Input: { a: number, b: number }
  - Output: { result: number }
    - *Result will be mapped to a + b*
- GET /api/add?a=5&b=8
  - What does this output?
    - **{ result: 13 }**
    - { 13 }
    - 13

Where does the input go? URL or body?

URL (query parameters!)

Endpoints

- By accessing a URL, you are making a request to an endpoint
- GET catbook-5bwk.onrender.com/api/stories, a GET request for all stories
- POST catbook-5bwk.onrender.com/api/story, a POST request to make a story
- Multiple endpoints can exist under the same URL, but are differentiated based on the type of request
- catbook-5bwk.onrender.com/api/comment, gets or adds a comment based on GET or POST

Cool API Examples

MIT People:
https://tlepeopledir.mit.edu/q/kenchoi?_format=json&d=mit.edu

Massachusetts Institute of Technology

OpenWeather

[OpenWeather API]

[The Dog API]
Examples of Dogs

[Twillio API]

Automated Phone calls and Text Messages

twilio

Google Maps

[Google Maps API]

Directions, location information

[TwitterAPI]

OpenAI

[OpenAI API]

Language Models
GPT, code completion, DALL-E

Cool API Examples

MIT People:
https://tlepeopledir.mit.edu/q/kenchoi?_format=json&m=i&d=mit.edu

Massachusetts Institute of Technology

OpenWeather

[OpenWeather API]

[The Dog API]
Examples of Dogs

Goog

[Google M]

Directions, location information

[Twillio API]

Automated Phone

Many External APIs will require you to get an API key or token to either rate limit or charge you for using their API

It can go in query params, the body, or header of your HTTP request

I

[OpenAI API]

Language Models
GPT, code completion, DALL-E

POSTMAN

Tools for testing APIs

(great for debugging)

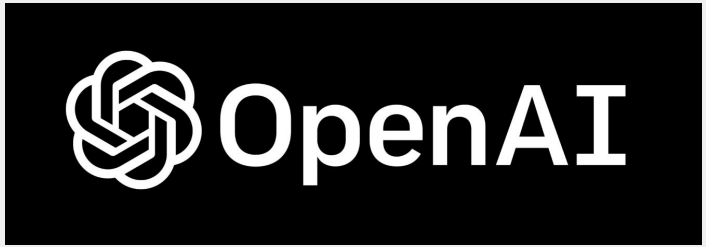
Postman.com

(requires sign-in, download version works on APIs that you run on your computer)

Alternative for quick demos:
https://reqbin.com/

Lots of options

Will you **send it** or **blend it**?



OpenAI API

Language Models
GPT, code completion, DALL-E

Preview of Making Requests to APIs in Javascript

get and post are functions written by weblab-staff and will be provided to you for your use, they actually are creating using a the [Javascript fetch method]

```
get("/api/getUserByNumber", {
  phoneNumber: "6172530418",
});
```

```
GET
/api/getUserByNumber?phoneNumber=6172530418
```

```
post("/api/adduser", {
  name: "Nick Tsao",
  school: "MIT",
  phoneNumber: "6172530148",
});
```

```
POST /api/adduser

Params: {
  name: "Nick Tsao",
  school: "MIT",
  phoneNumber: "6172530148",
}
```

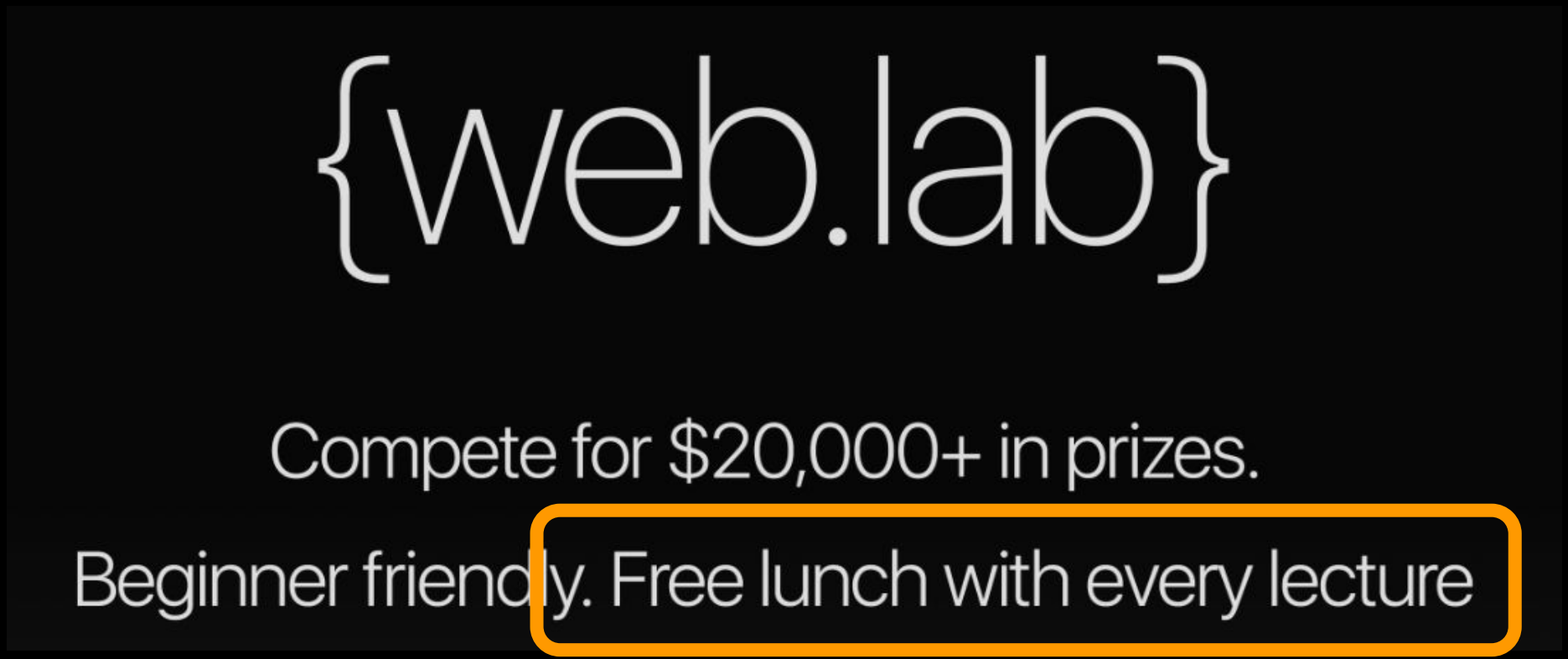
```
}
        const user: Promise<any>
const user = get("/api/getUserByNumber", {
  phoneNumber: "6172530418",
});
```

> **name**: Reif, L. Rafael
> **department**: Chair of the Corporation
> **email**: reif@mit.edu
> **phone**: 617-253-0148
> **address**: 3-207
> **title**: President Emeritus

Hold on a second, if I try to console.log(user), I don't get anything except:

```
[object Promise]
```

The return type of the **`get`** function is actually a **PROMISE**


{web.lab}
Compete for $20,000+ in prizes.
Beginner friendly. Free lunch with every lecture

Promises?

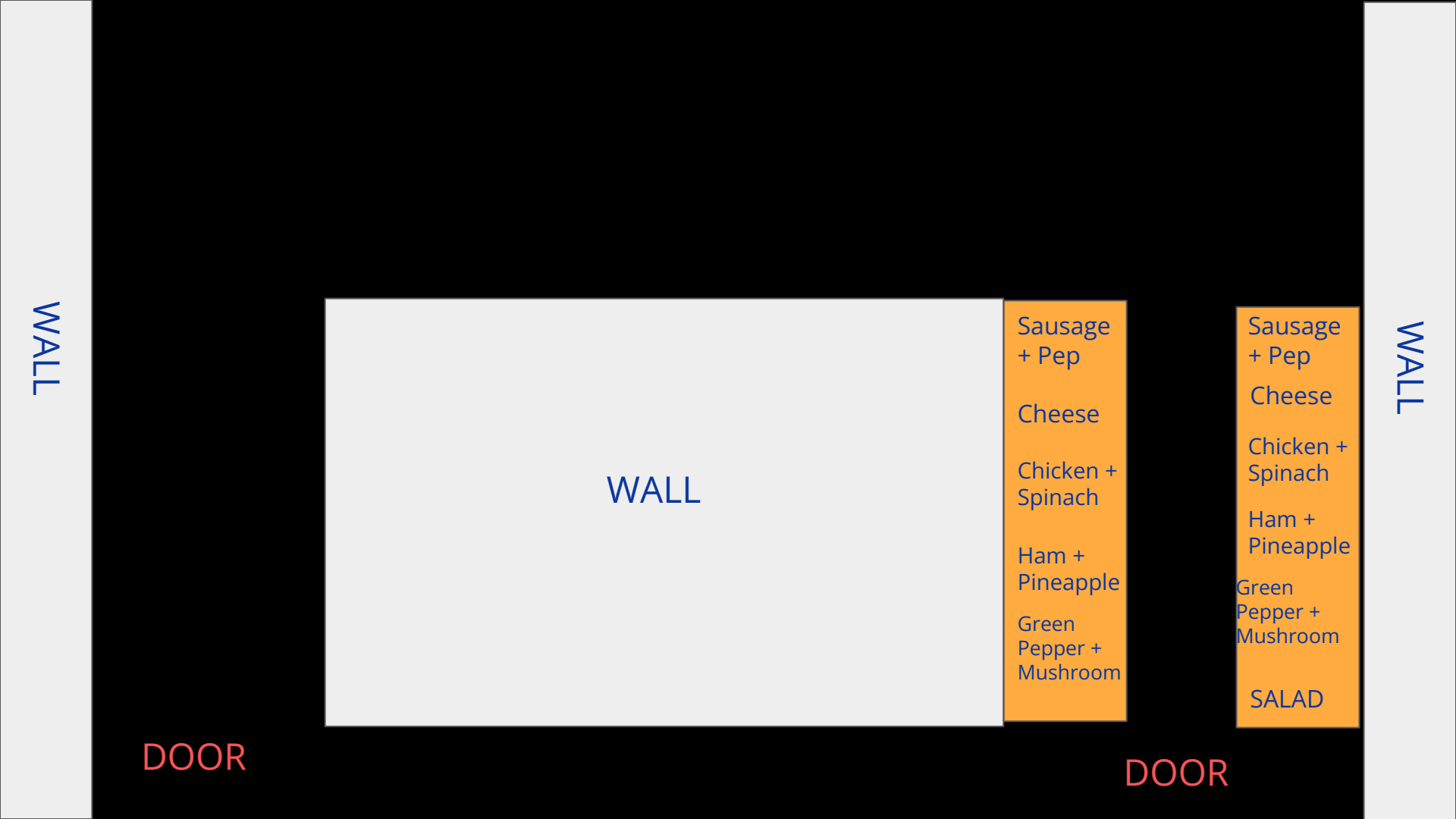
Come back at 1:10pm :)

Remember not to eat in the lecture hall!

```
const lunchTime = new Date('Jan 10, 2024 12:30:00')

const lunchPromise = new Promise((resolve, reject) => {
  setInterval(() => {
    if (Date.now() > lunchTime) {
      resolve('lunch time!')
    }
  }, 1000 * 60)
})

lunchPromise
  .then((val) => console.log(val))
  .catch((err) => console.log(`Oops, we forgot to order lunch today: ${err}`))
```

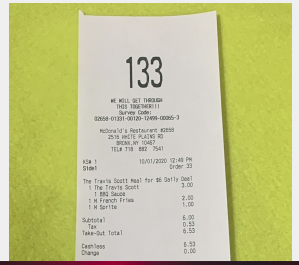

WALL
WALL
Sausage + Pep
Cheese
Chicken + Spinach
Ham + Pineapple
Green Pepper + Mushroom
Sausage + Pep
Cheese
Chicken + Spinach
Ham + Pineapple
Green Pepper + Mushroom
SALAD
WALL
DOOR
DOOR

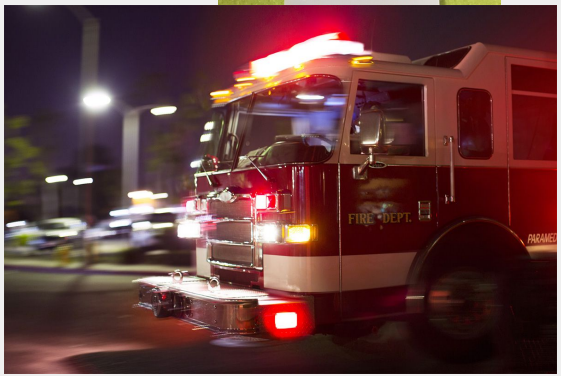
Promises

“I promise I will try to make a burger for you” - weblab-staff

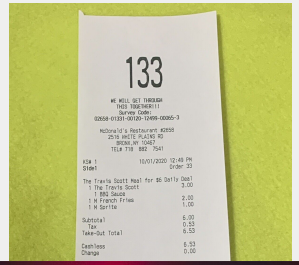
"I promise I will try to make a burger for you" - weblab-staff

"I promise I will try to make a burger for you" - weblab-staff

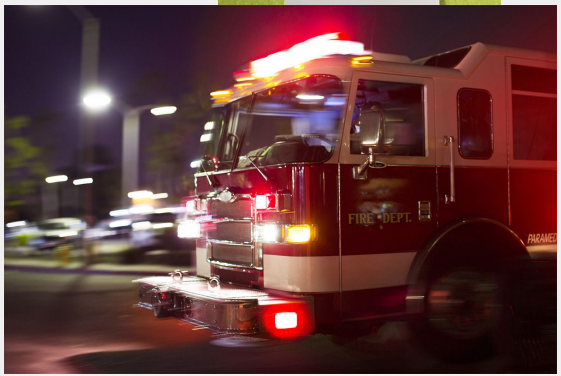




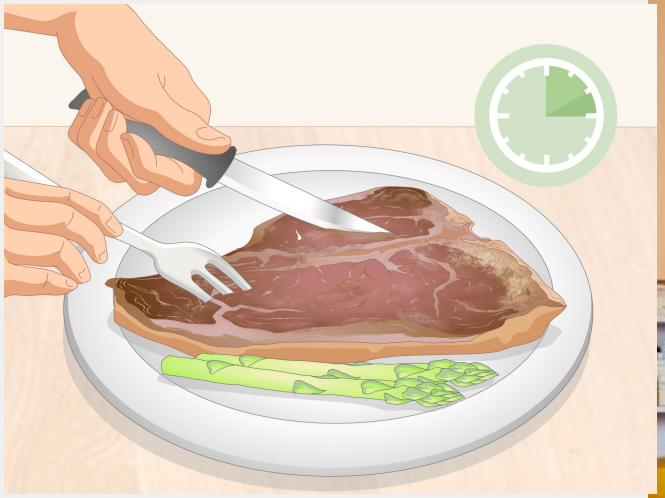
Promise
<u>fulfilled</u>



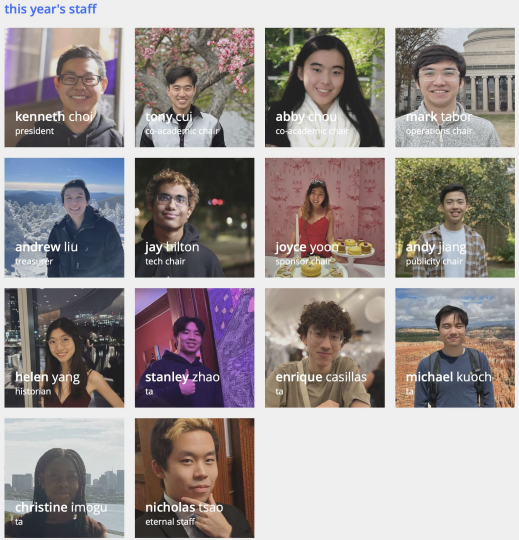
Promise
<u>rejected</u>



Synchronous Processes



Cool web.lab kitchen

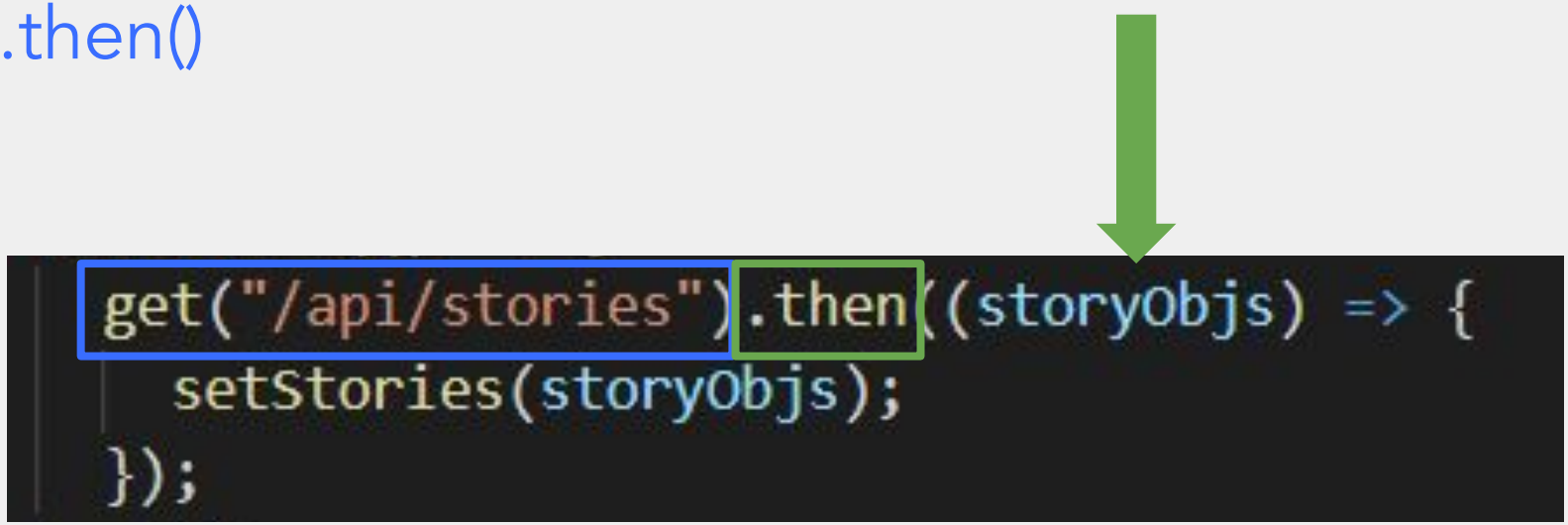


Asynchronous Processes

Promises allow for asynchronous processing

More on this tomorrow, but see how to use promises in Javascript!

.then()



Once the promise is **fulfilled**, do stuff (call a callback function).
Returns a promise.

.catch()

```
get("/api/stories").then((storyObjs) => {
  setStories(storyObjs);
}).catch((err) => {
  console.log("this is so sad: ", err.message);
});
```

Once the promise is **rejected**, do stuff (call a callback function).
Returns a promise.

Things like calling an API takes time
Promises don't hold things up

```
myOneSecondPromise().then((value) => {
  console.log('Promise fulfilled with value:', value)
}).catch((err) => {
  console.log('Error', err)
})

console.log('the circumference of earth is', Math.PI * Math.pow(6378, 2), 'km')
console.log('Herro')      You, 1 second ago • Uncommitted changes
```

```
myOneSecondPromise().then((value) => {
  console.log('Promise fulfilled with value:', value)
}).catch((err) => {
  console.log('Error', err)
})

console.log('the circumference of earth is', Math.PI * Math.pow(6378, 2), 'km')
console.log('Herro')      You, 1 second ago • Uncommitted changes
```

```
hello, the circumference of earth is 127796483.13063137 km    VM108:16
Herro                                                         VM108:17
Promise fulfilled with value: hello                           VM108:11
```

What gets printed here?

- **The promise below is going to reject**

```
const f = (promise) => {
  promise.then((val) => console.log("a")).then((val) => console.log("b")).catch((err) =>
console.log("error!"));
};

const promise = /* Some way of getting a promise that will reject. Maybe there's a network issue. */
f(promise)
```

What gets printed here?

- **The promise below is going to reject**

```
const f = (promise) => {
  promise.then((val) => console.log("a")).then((val) => console.log("b")).catch((err) =>
console.log("error!"));
};

const promise = /* Some way of getting a promise that will reject. Maybe there's a network issue. */
f(promise)
```

- “a”, “b”, “error!”
- “a”, “error!”
- “b”, “error!”
- “error!”
- “a”, “b”

What gets printed here?

- **The promise below is going to reject**

```
const f = (promise) => {
  promise.then((val) => console.log("a")).then((val) => console.log("b")).catch((err) =>
console.log("error!"));
};

const promise = /* Some way of getting a promise that will reject. Maybe there's a network issue. */
f(promise)
```

- “a”, “b”, “error!”
- “a”, “error!”
- “b”, “error!”
- **“error!”**
- “a”, “b”

How about here?

- **The promise below is going to resolve after a little while**

```
const f = (promise) => {
  promise.then((val) => console.log("a")).then((val) => console.log("b")).catch((err) =>
console.log("error!"));
};

const promise = /* Some way of getting a promise that will accept. */
f(promise)

console.log("Hi there!");
```

How about here?

- **The promise below is going to resolve after a little while**

```
const f = (promise) => {
  promise.then((val) => console.log("a")).then((val) => console.log("b")).catch((err) =>
console.log("error!"));
};

const promise = /* Some way of getting a promise that will accept. */
f(promise)

console.log("Hi there!");
```

- “a”, “Hi there!”
- “Hi there!”, “a”, “b”
- “b”, “a”, “Hi there!”
- “Hi there!”, “a”
- “a”, “b”, “Hi there!”

How about here?

- **The promise below is going to resolve after a little while**

```
const f = (promise) => {
  promise.then((val) => console.log("a")).then((val) => console.log("b")).catch((err) =>
console.log("error!"));
};

const promise = /* Some way of getting a promise that will accept. */
f(promise)

console.log("Hi there!");
```

- “a”, “Hi there!”
- **“Hi there!”, “a”, “b”**
- “b”, “a”, “Hi there!”
- “Hi there!”, “a”
- “a”, “b”, “Hi there!”