

REST APIs

Michael Chang
Spring 2023

Plan for today

Structure of HTTP request/response

Method, path, query string, headers, body

Recap: fetch with async/await

Classes as data models

Handling errors

Using REST APIs

Sending data: method, headers, body

Keeping data model up to date

Classes and REST APIs

Classes can model resources

E.g. a Student or User class

Loading (reading) a resource

```
class Student {  
    /* Can't make constructor async */  
    static async load(id) {  
        let data = await ...;  
        return new Student(data);  
    }  
}
```

Classes and REST APIs

Classes can model resources

E.g. a Student or User class

Loading (reading) a resource

```
class Student {  
    constructor(data) {  
        /* Copy key/values from data to this */  
        Object.assign(this, data);  
        /* ... init private instance vars */  
    }  
}
```

REST APIs

Representational state transfer

Defines certain rules the API will follow

Resources

Each "thing" we want to send/receive is a "resource"

Identified by a URI (path)

E.g. /courses/CS193X or /users/mchang91

Servers return "representation" of the resource

Clients send (possibly partial) representations to update resources

Statelessness

Server doesn't "remember" clients

I.e. each request includes URI, other info

HTTP requests

Example

```
GET /students/mchang?include_photo=1
```

Method: what we want to do

GET: get some information

POST: send some information (and get response)

(For REST APIs)

PATCH: update a resource

DELETE: delete a resource

HTTP requests

Example

```
GET /students/mchang?include_photo=1
```

Path: the resource we're accessing

A URI (parts separated by /)

Some parts are fixed (e.g. "students")

Some are identifiers (e.g. "mchang91")

HTTP requests

Example

```
GET /students/mchang?include_photo=1
```

Query string: additional info about resource

Describe what you're looking for

Key/value pairs, separated by &

Keys and values are [URI encoded](#)

E.g. ?q=search+string&lang=en

Would encode two key/value pairs

q: "search string"

lang: "en"

HTTP requests

Example

```
POST /students/mchang/enroll
```

```
Content-Type: application/json
```

```
{"course": "cs193x"}
```

Headers: info about request

What browser we're using ([User-Agent](#))

What type of data we're sending ([Content-Type](#))

HTTP requests

Example

```
POST /students/mchang/enroll
```

```
Content-Type: application/json
```

```
{"course": "cs193x"}
```

Body: data sent to server

Only for non-GET requests

Used when sending full objects

May be the full object (e.g. to create it), partial object (to update), or specific parameters (for custom actions)

HTTP response

Example

HTTP/1.1 200 OK

Content-Type: application/json

```
{"id": "mchang", "firstName": "Michael", ...}
```

Status code: result of request

Gives a general indication of success/failure

Text after the number is generic, specified by HTTP

E.g. 200 will always be "OK", 404 will be "Not Found"

HTTP response

Example

HTTP/1.1 200 OK

Content-Type: application/json

```
{"id": "mchang", "firstName": "Michael", ...}
```

Headers: info about the response

The type of server ([Server](#))

The type of response data ([Content-Type](#))

HTTP response

Example

HTTP/1.1 200 OK

Content-Type: application/json

```
{"id": "mchang", "firstName": "Michael", ...}
```

Body: the resource, error message, etc.

When GETting a resource, probably the object

When an error occurs, often contains a message

When taking an action, info on success/failure

Common HTTP statuses

200 OK

Request was successful

400 Bad Request

Server couldn't understand the request, or couldn't do the thing

401 Unauthorized

Need to log in or send some credentials

403 Forbidden

Credentials provided, but you don't have access

404 Not Found

The thing you asked for isn't there

500 Server Error

Problem on the server side

Sending data to server

`fetch(url[, options])`

`options` is an object with following keys

`method`: HTTP method

`headers`: Object of HTTP headers to include in request

`body`: request body (for non-GET) (as a string)

When sending data to server

Query string goes in the URL

When including request body, need to set Content-Type header

E.g. `headers: { "Content-Type": "application/json" }`

Sending data to server

```
const postData = async () => {  
  let data = { num: 42 };  
  let res = await fetch("/api/path?param=binky", {  
    method: "POST",  
    headers: { "Content-Type": "application/json" },  
    body: JSON.stringify(data)  
  });  
  ...  
};
```


Data models

Useful to encapsulate data in classes

E.g. Student, Course

Methods for reading from and updating API

Note: constructor cannot be async

Instead, use a static method

Data models

Useful function: `Object.assign(dest, src)`

Copy all the keys from src into dest (overwriting)

E.g. `Object.assign(this, data)`

`myClass.toJSON()`

Define this method to control how `JSON.stringify` converts object into JSON

E.g. include only public instance variables

class Usage

As DOM component

Encapsulate logic for creating and managing elements

Allows reuse of components

Common features:

- Event handlers (don't forget to bind!)

- Private instance variables for DOM elements

- Public methods take DOM element and/or other components as argument

- Methods take callbacks to notify other classes of events

class Usage

As data model

Encapsulate logic and state for a resource (a "noun" in our system)

Provides a public interface for retrieving/updating data from/to API

Common features:

- async methods that make API requests

- static method(s) to retrieve instances from the API

- Easiest to keep data returned from API "public"

- Private instance variables for client-side-only state

- toJSON method to control how resource is sent back to server

Summary

Today

Rest APIs

Before next time

assign2.2 (CSS)

Next time

Wrap up client-side APIs

Start talking about servers, how to build these APIs