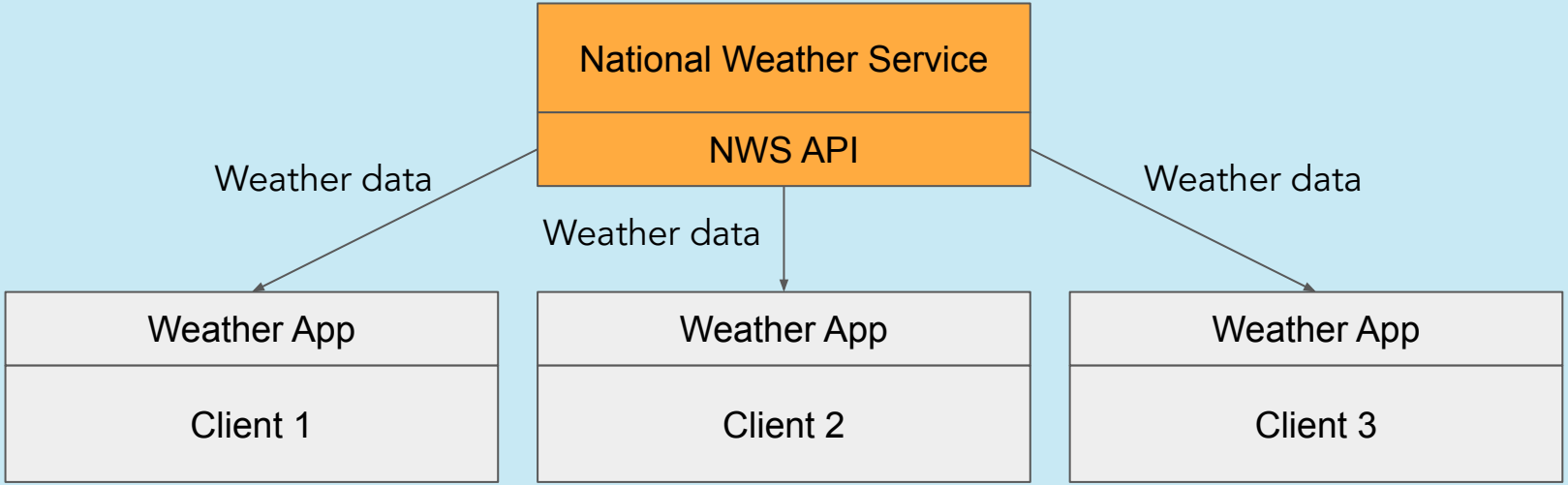
Cubstart Web Lecture 5

[start recording]

Administrivia

- HW4 (Gradescope assignment) is due on Friday at midnight!
- HW 5 will be released in the next few days
  - This HW is a bit difficult! Get started early and please ask questions on Ed
  - **Come to lab!** We will go over ~half of HW5 with you this Friday!

Recap: APIs


National Weather Service
NWS API
Weather data
Weather data
Weather data
Weather App
Client 1
Weather App
Client 2
Weather App
Client 3

HTTP: request-response model

- Client makes the request
- Server provides a response
- Application layer: HTTP

Type of HTTP Request

- GET
- POST
- PUT
- DELETE

/books

GET	/books	Lists all the books in the database
DELETE	/books/{bookId}	Deletes a book based on their id
POST	/books	Creates a Book
PUT	/books/{bookId}	Method to update a book
GET	/books/{bookId}	Retrieves a book based on their id

HTTP Status Codes

Level 200 (Success)

200 : OK

201 : Created

203 : Non-Authoritative Information

204 : No Content

Level 400

400 : Bad Request

401 : Unauthorized

403 : Forbidden

404 : Not Found

409 : Conflict

Level 500

500 : Internal Server Error

503 : Service Unavailable

501 : Not Implemented

504 : Gateway Timeout

599 : Network timeout

502 : Bad Gateway

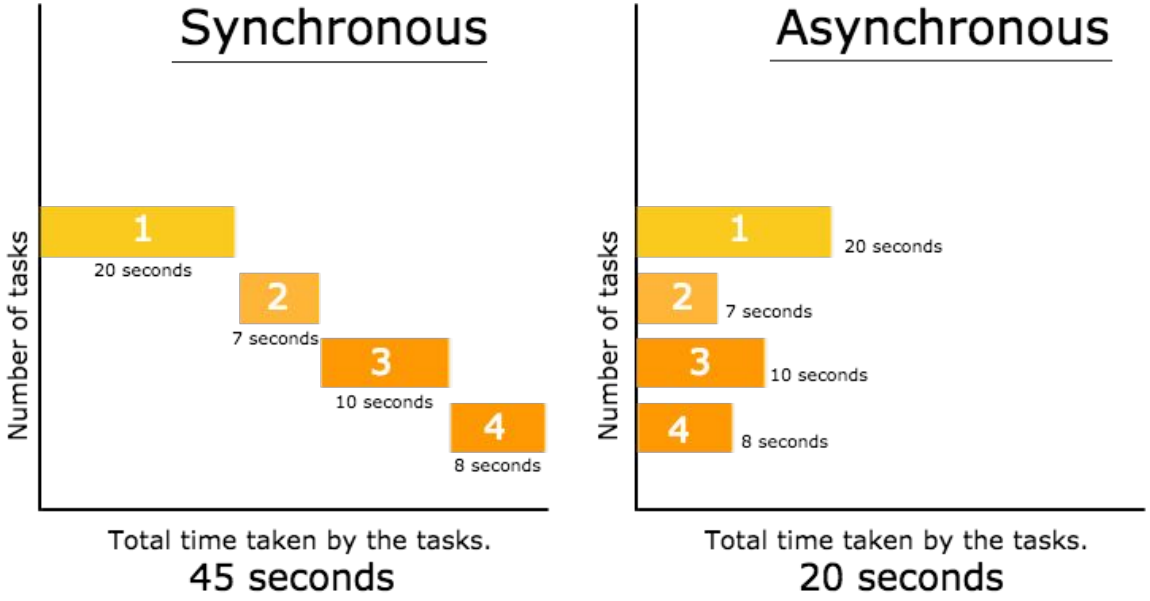
How do we integrate APIs into a web application?

Asynchronous Functions

Synchronous JavaScript

JavaScript is synchronous and single-threaded.

Data retrieval is generally asynchronous (think Instagram).



Why JavaScript can be Tricky

```
function getData() {
    const pokemon = fetchRandomPokemon();
    console.log(pokemon)
}
```

This function retrieves data, meaning it is **asynchronous**!

<u>Console:</u>
>>>undefined

This function is still fetching data as JavaScript runs the following line!

So what is `pokemon` if it is not yet the retrieved pokemon data?

Promises

What is a Promise?

Two definitions:

1. “A Promise represents the eventual completion (or failure) of an asynchronous operation and its resulting value.”

2. “A Promise is a proxy for a value not necessarily known when the promise is created.”

Promise States

- **pending**: initial state, neither fulfilled nor rejected.
- **resolved**: meaning that the operation was completed successfully.
- **rejected**: meaning that the operation failed.

Async Functions return a Promise

```
function getData() {
    const pokemon = fetchRandomPokemon();
    console.log(pokemon)
}
```

Remember that this function retrieves data, meaning it is asynchronous!

This promise is in the **pending** state when JavaScript runs *console.log(pokemon)*, which is why it prints 'undefined'

How do we force JavaScript to follow the asynchronous timeline of data retrieval?

Async, Await

The Solution: Async, Await

```
async function getData() {
    const pokemon = await fetchRandomPokemon();
    console.log(pokemon)
}
```

Console:

```
{
    "name": "squirtle"
    "Id": 24
}
```

Try Catch Blocks

```
async function getData() {
    try {
        const pokemon = await fetchRandomPokemon();
        console.log(pokemon)
    } catch (error) {
        console.log("Oops")
    }
}
```

How do we fetch data/resources from an API in JavaScript?

fetch()

- Allows developers to fetch resources from a server!
- Syntax: fetch(apiURL, options) *second parameter optional

```
async function getData() {
    const response = await fetch("https://www.boredapi.com/api/activity/")
    const data = await response.json()
    console.log(data)
}
```

Parses the response body as JSON

Console:

```
{
    activity: "Touch grass",
    minprice: 0,
    maxprice: 0
}
```

fetch()

You can send requests with **options** as the **second parameter**

```
const response = await fetch("http://www.apiExample.com", {
    method: 'GET',
    headers: {
        "Content-Type": "application/json"
    }
});
```

It’s demo time 🤩

HTML file from demo:

index.html × script.js

index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dogs</title>
</head>
<body>
    <h1>Dogs</h1>
    <div>
        <select id="select-breed"></select>
        <button id="load">Load</button>
    </div>

    <div>
        <img id="dog-pic" />
    </div>
    <script src="script.js"></script>
</body>
</html>
```

JS file from demo:

index.html | script.js

script.js > ...

```javascript
const selectEl = document.getElementById('select-breed')
const imageEl = document.getElementById('dog-pic')
const buttonEl = document.getElementById('load')

async function setup() {
    const response = await fetch("https://dog.ceo/api/breeds/list/all")
    const data = await response.json()

    const breeds = Object.keys(data.message)

    for (const breed of breeds) {
        // something to do with breed
        const optionEl = document.createElement('option')

        optionEl.textContent = breed
        optionEl.value = breed

        selectEl.appendChild(optionEl)
    }
}

async function loadImage() {
    const selectedBreed = selectEl.value
    const response = await fetch("https://dog.ceo/api/breed/" + selectedBreed+ "/images/random")
    const data = await response.json()

    imageEl.src = data.message;
}

buttonEl.addEventListener("click", loadImage)

setup()
```

[stop recording]

Attendance: Lecture 5

https://forms.gle/LAAZ28LAEzEcpfP59

Secret word: