Before we begin...

- Videos On!

Welcome

Agenda

- APIs
- AJAX

APIs

What are APIs?

- It stands for **Application Programming Interface** (**API**)
- Software that allows two programs to communicate with each other
  - It all starts with shared data and functionality
- The principle of API abstraction allows for decoupling applications
- They can be private or public, internal or external, paid or free

What are APIs?

- An API is a set of rules that allows for one piece of software to talk to another

What is an API?

Application - Any application

Programming - The engineering part that translates given inputs into outputs

Interface - The way we interact with the other server

Picture an ATM

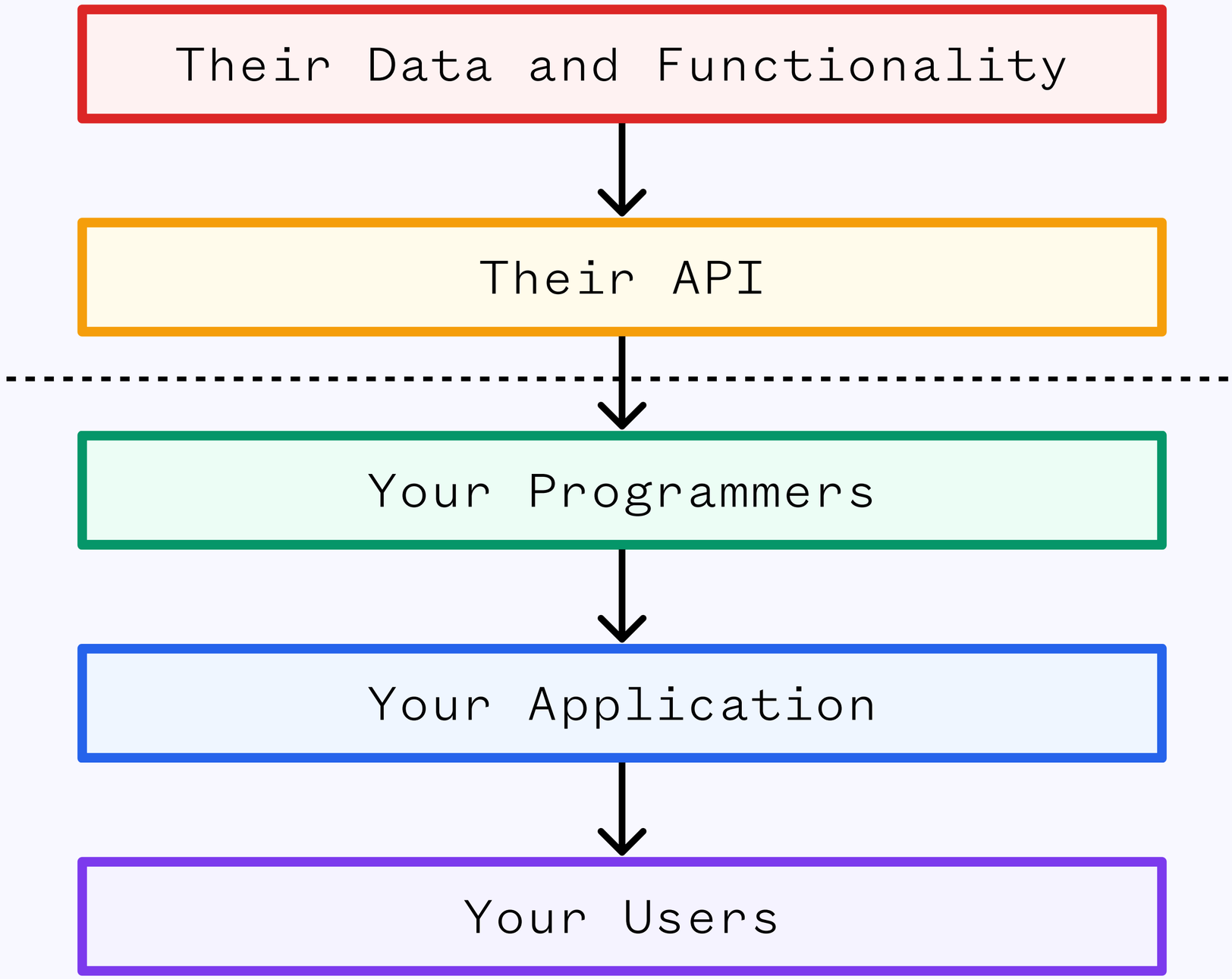
Types of APIs

- Operating Systems
- Remote APIs
- Libraries, frameworks and software development kits
- Web APIs

We will focus on the last two

For us though?

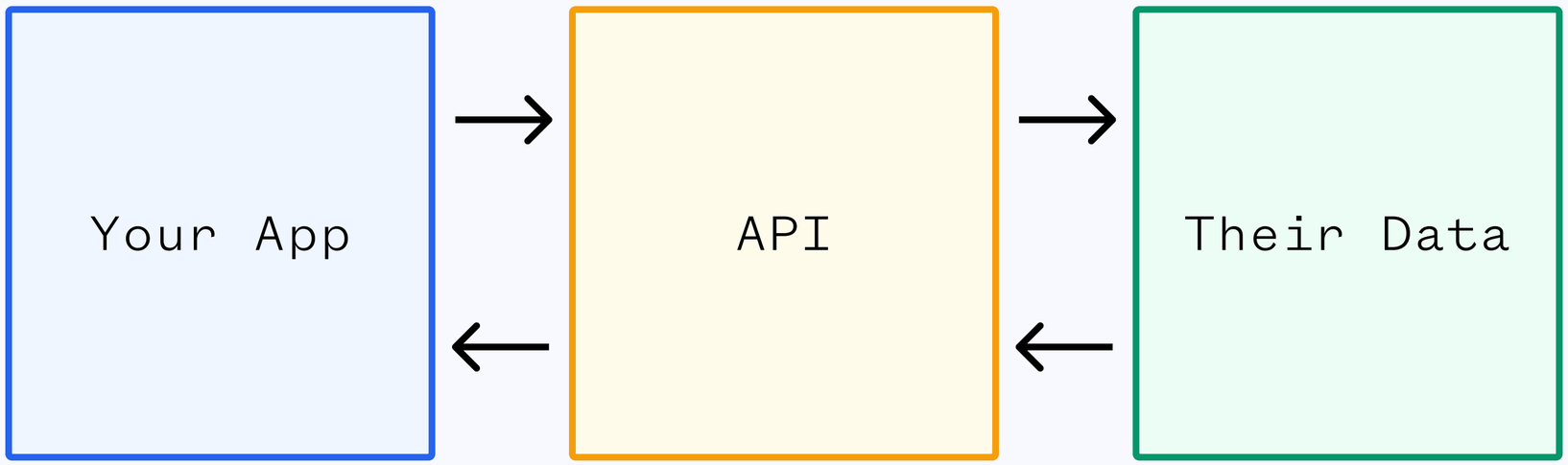
- There are browser APIs (think things like the DOM)
- There are third-party APIs, where we access data from a remote server

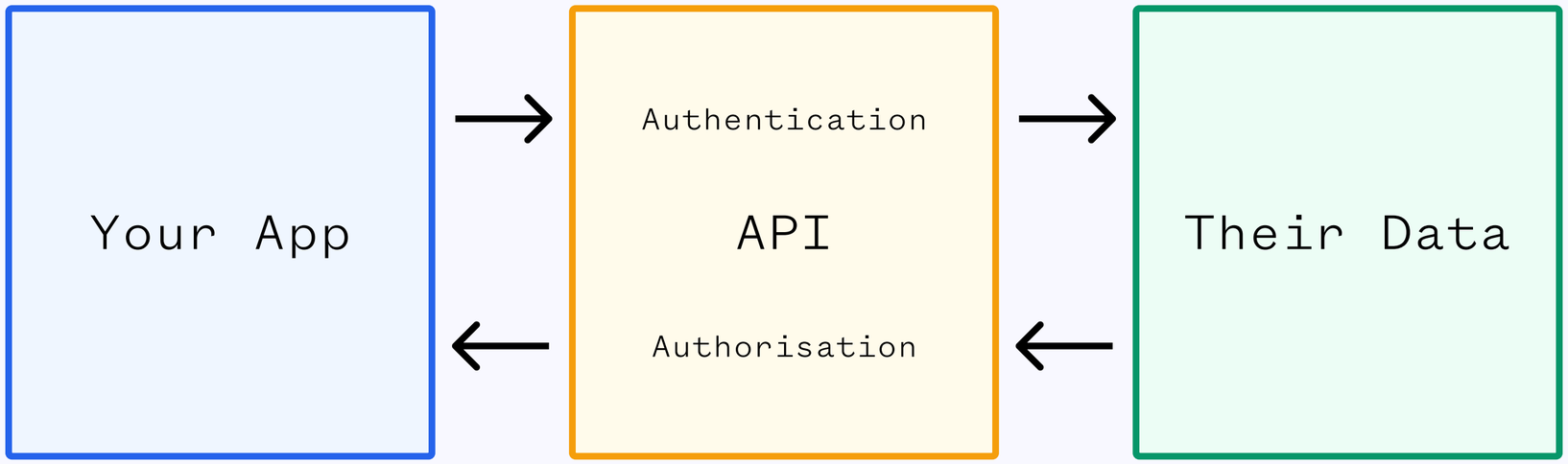

Their Data and Functionality
Their API
Your Programmers
Your Application
Your Users

How do they work?

- **Their Data and Functionality**: Anything is chosen to be shared
- **Their API**: The gateway to those assets
- **Your Application**: You developers code your application (which is powered by the API)
- **Your Developers**: The API is exposed to your developers
- **Your Users**: Your users use the app and reap the benefits


Cafe
Waitstaff
Kitchen


Your App
API
Their Data


Your App
Authentication
API
Authorisation
Their Data

Benefits of APIs for Providers

- APIs let you build one app off another
- APIs are like a universal plug
- APIs can be profitable
- APIs can allow you to decouple code - can make your applications more performant and finding developers easier
- APIs can essentially outsource complexity
- API providers can promote your application

Benefits of APIs for Consumers

- APIs can allow you to get data, and add features, that would otherwise be difficult and time-consuming
- APIs can make your app realtime
- APIs can introduce similar flows to an application (e.g. logins etc.)
- Apps using APIs can provide useful starting data for users

Downsides

- Building them
- Maintenance
- Hosting
- Documentation
- For developers - future support
- Reliance on other services

Authentication and Authorization

Authentication

Are you allowed to access the system?

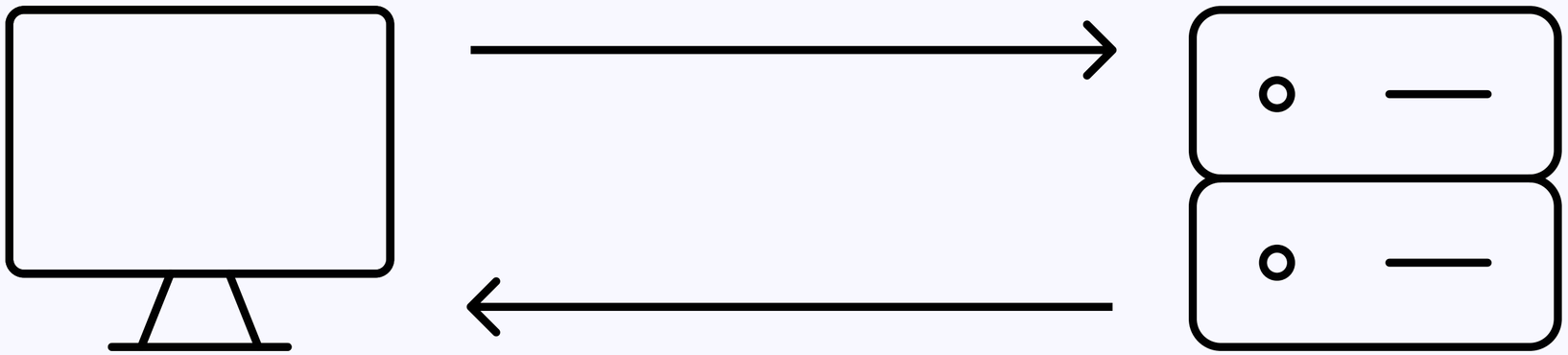
Authorization

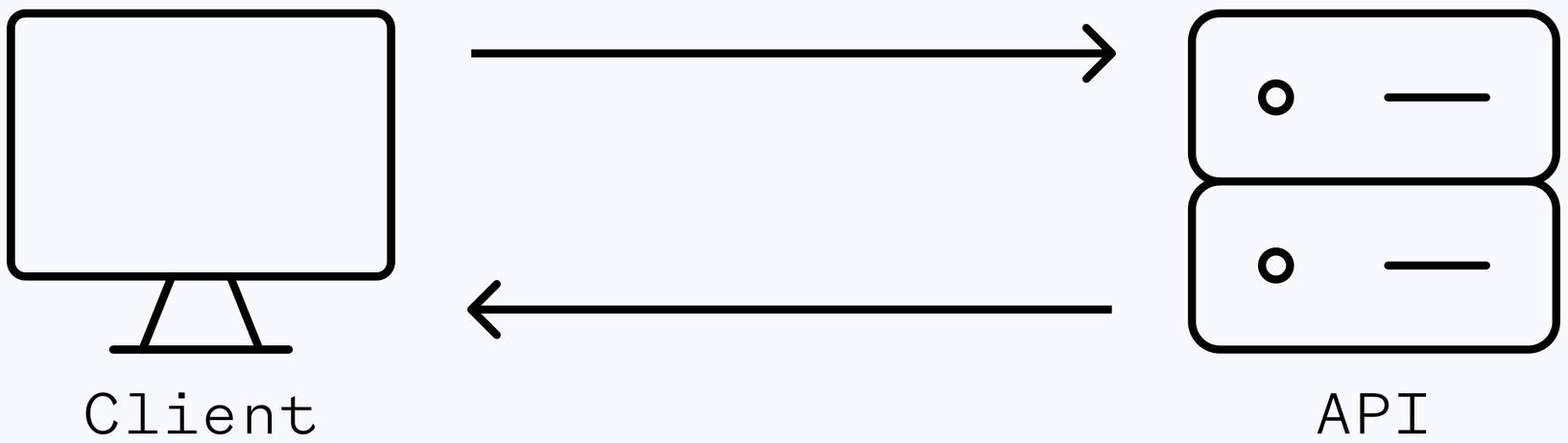
What can you access in the system?

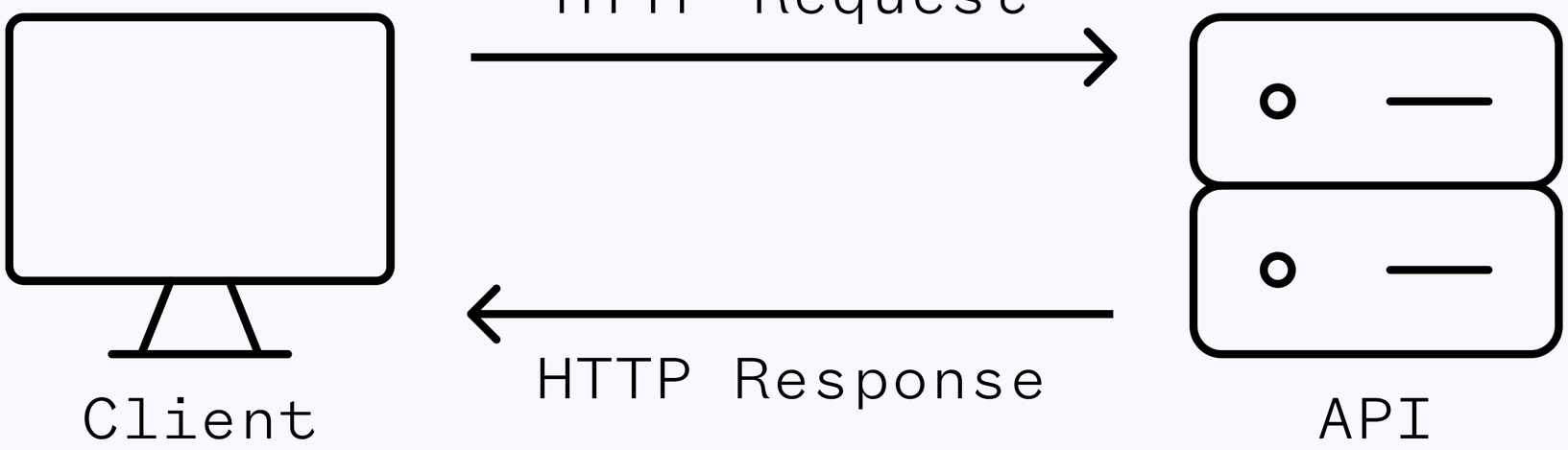
Often we need to sign up to APIs

Internal Workings of APIs

Let's rewind a little




Client
API


HTTP Request
HTTP Response
Client
API

HTTP Requests

Making HTTP Requests

We need to provide some information when we speak to APIs

1. **The API Endpoint**: The URL we are getting the data from
2. **The HTTP Method**: Describes our intentions with the data
3. **The HTTP Headers**: Key-value pairs that provide internal configuration data (e.g. for configuration or authentication)
4. **The Body Data**: The data we want to provide to the server

HTTP Requests

Let's review URLs

- It stands for Uniform Resource Locator
- It's an address for a unique resource on the web
- It's made up of lots of parts! Let's see them now

URLs

```
https://twitter.com/GA
```

- `https://` - Scheme or Protocol
- `twitter.com` - Domain
- `/GA` - Path

URLs

```
https://www.youtube.com/watch?v=tc8DU14qX6I
```

- `https://` - Scheme or Protocol
- `www.youtube.com` - Domain
- `/watch` - Path
- `?v=tc8DU14qX6I` - Query Parameters

URLs

```
https://ga.co/education/javascript-development/sydney#curriculum
```

- `https://` - Scheme or Protocol
- `ga.co` - Domain
- `/education/javascript-development/sydney` - Path
- `#curriculum` - Anchor

URLs

```
https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL
```

- `https://` - Scheme or Protocol
- `developer` - Subdomain
- `mozilla.org` - Domain
- `/en-US/docs/Learn/Common_questions/What_is_a_URL` - Path

URLs

```
https://subdomain.domain.com:80/path/to/resource?key1=value1#anchor
```

- `https://` - Scheme or Protocol
- `subdomain` - Subdomain
- `domain.com` - Domain
- `:80` - Port
- `/path/to/resource` - Path
- `?key1=value1` - Query Parameters
- `#anchor` - Anchor

HTTP Methods

There are a lot of different types of HTTP Methods, but in each case they define our intentions with the resource (the data). The main ones are:

- **POST** - Creating Data
- **GET** - Reading Data
- **PUT** / **PATCH** - Updating Data
- **DELETE** - Deleting Data

These correspond to the main Database Operations we can perform (CRUD - create, read, update, delete)

HTTP Headers

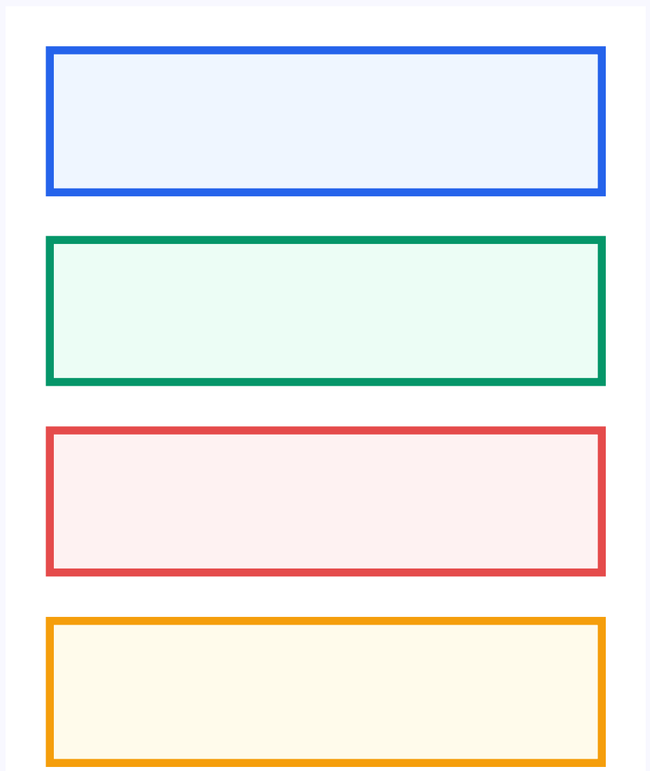
- Key-value pairs
- Used to share information between the client and the server
- You can store things like cookies, metadata and authentication tokens in here
  - The most common though is to describe what data you'd like back using `"Content-Type"`
- Here is a list of all valid headers

Body Data

- The body contains any data that you'd like to provide to the server
  - Usually as a JSON-encoded string (more on that soon)

A Quick Recap

HTTP Request



1 API Endpoint (URL)

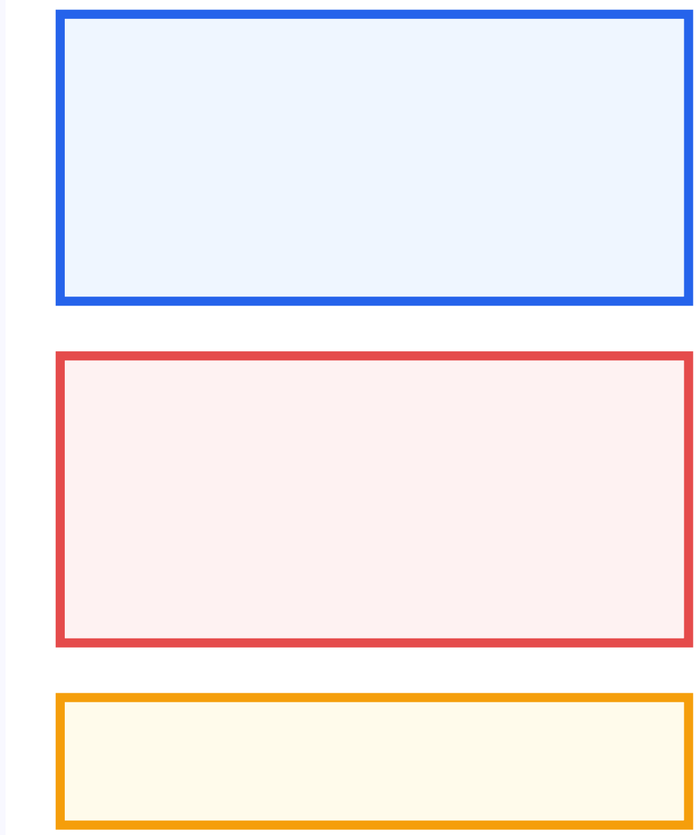
2 HTTP Method

3 HTTP Headers

4 Body Data

HTTP Responses

HTTP Response



1 Response Data

2 Status Code

3 Extra things

Responses

Once the request has been made the server will send back an HTTP response, which is made up of data and a status code. There are lots of different status codes, but they are somewhat organized:

- **100+**: Informational Response
- **200+**: Request has succeeded
- **300+**: Request has been redirected
- **400+**: An error on the client's end has occurred
- **500+**: An error on the server's end has occurred

Here is a list of all of the status codes

AJAX

What is AJAX?

- **A**synchronous **J**avaScript **A**nd e**X**tensible Markup Language
- It is a way to make your pages live
  - You can talk to other servers while you are still on the page
- It is a technique to send and retrieve information behind the scenes without needing to refresh the page
- It's JavaScript's way of speaking to APIs

Where is AJAX used?

- In feeds (such as Twitter)
- Chat rooms and messaging apps
- For voting and rating
- Autocompletion
- Form submission and validation
- To access data that you don't have
- To show extra information
- In games (e.g. to save scores)

Why is AJAX so useful?

- It makes your pages live
- It is much faster
- It tends to give a greater user experience
- It is fancy
- It is popular in the workplace
- It is the foundation of things such as APIs

Some Scenarios

1. Someone liking a post on social media
2. Someone signing up to an account
3. Someone checking the weather forecast
4. Someone in a chat room, or a messaging app

How do we work with it?

- XMLHttpRequest
- Fetch

One Thing!

To make API Requests, we **need a server** (it doesn't work on a file URL)

For the moment, we will be using HTTP Server for this

To start the server, run the following command in the directory you want to serve (and open the URL it provides):

```
http-server .
```

fetch

What is fetch?

- It is a way to make AJAX requests
- It is a function that is automatically defined for us

What is fetch?

You make an HTTP Request with it

Specifying the URL, the method etc.

It'll return a promise that will eventually represent the completed HTTP Response

- Most of the time, the data is returned as JSON (we need to convert this into a regular JavaScript object)
  - We can work with that returned data in a `.then`
  - We can handle errors in a `.catch`

What is JSON?

- It stands for **J**ava**S**cript **O**bject **N**otation
- A format for storing and sharing information
- It looks very similar JavaScript Objects, but keys are quoted

What is JSON?

It's a format that is really easy to work with, particularly in JavaScript

```
// We can convert an object into JSON
let jsonString = JSON.stringify(obj);

// We can convert JSON into an object
let convertedObj = JSON.parse(json);
```

Using Fetch

```
fetch(URL, httpOptions)
  .then(successHandler)
  .catch(errorHandler);
```

Using Fetch

1. We make an HTTP Request
2. We wait for the HTTP Response to come back
3. We convert the Response Data into JS data types (using methods like `.json`)
4. We use the data

Using Fetch

```
function convertToJSObject(res) {
  return res.json();
}

function handleData(data) {
  console.log(data);
}

fetch("http://api.open-notify.org/astros.json")
  .then(convertToJSObject)
  .then(handleData);
```

Now you know how many people are in space at the moment!

Parameters

- We can attach a Query String or Parameters to a URL
- This provides extra information to the API
- Works in a similar way to an object - query strings have keys and values
- Looks something like this:

```
?keyOne=valueOne&keyTwo=valueTwo&keyThree=valueThree
```

Using Fetch

```
function convertToJSObject(res) {
  return res.json();
}

function handleData(data) {
  console.log(data);
}

fetch("https://randomuser.me/api/?results=10")
  .then(convertToJSObject)
  .then(handleData);
```

Using Fetch

```
function convertToJSObject(res) {
  return res.json();
}

function handleData(data) {
  console.log(data);
}

fetch("https://randomuser.me/api/?results=10&gender=male")
  .then(convertToJSObject)
  .then(handleData);
```

OMDB API

Steps

- Go to OMDB API's Website
- Get an API Key from here and select Free
- Input your details
- Check your email
- Click the verify API key link
- Copy your API key from that email

Using Fetch

```
let apiKey = "YOUR API KEY GOES HERE";

function convertToJSObject(res) {
  return res.json();
}

function handleData(data) {
  console.log(data);
}

fetch(`http://www.omdbapi.com/?t=Jaws&apikey=${apiKey}&plot=full`)
  .then(convertToJSObject)
  .then(handleData);
```

Open Weather Map

Steps

- Go to the Open Weather Map API website
- Sign up for an API key here
- Fill in your details
- Log in
- Go to the API Key Tab on your settings page
- Copy the API Key
- It'll take a little while for the API Key to work

Using Fetch

```
let apiKey = "YOUR API GOES HERE";
let baseURL = "http://api.openweathermap.org/data/2.5/weather";
let parameters = `?q=Sydney&units=metric&appid=${apiKey}`;

function convertToJSObject(res) {
  return res.json();
}

function handleData(data) {
  console.log(data);
}

fetch(baseURL + parameters)
  .then(convertToJSObject)
  .then(handleData);
```

Some other APIs

- Rest Countries
- Programming Quotes
- Air Quality Index
- Dad Jokes
- Fixer
- Sunrise/Sunset
- Many more here
  - Make sure you don't use any with OAuth (yet)

Resources

- Rapid API
- MDN: Using Fetch
- CSS Tricks: Using Fetch
- Scotch.io: Fetch
- David Walsh: Fetch
- Google Developers: Fetch
- Google Developers: Working with the Fetch API
- MDN: Fetch API

That's all for tonight!

Homework

- Do more with APIs!
- Track the International Space Station's Position
  - Using this API
  - Show the current location of it in your HTML
  - Bonus: Link it up to the Google Maps API!
- Or any other API you want
  - Note: Don't use ones requiring OAuth. We will be going through this later

What's Next?

- More APIs
- More AJAX

Thank You!