Application Programming Interface

BACKEND DEVELOPMENT

CSIS 3380 *FULL STACK DEVELOPMENT WITH JAVASCRIPT*

ANU GUPTA

guptaa10@douglascollege.ca

Outline

Introduction to API

Endpoints, Path and Parameters

API Authentication

Making GET requests

Using Express to render a website with live API data

Making asynchronous requests using Axios and Fetch API

Application Programming Interface

- It is a set of rules and protocols that define how different software can interact with each other.
- Let's say we have a program called Program A, and we have another one called Program B
- If we wanted program A and program B to talk to each other because they provide different pieces of functionality and we want them to work together, then we would need them to be able to communicate to each other.
- We would need some kind of interface in between these two programs so that they can interact with each other.

API is the an interface between two pieces of software. By defining a protocol or a set of rules that you can follow, then the different pieces of software know how to interact with each other and what they should expect when they do interact with the other service.

Application Programming Interface

APIs define the methods and data formats that developers can use to request and exchange information or perform specific functions between different software systems.

APIs allow you to access another app's data or functionality. We can send an API a request detailing the information we want.

For example, if you want to make a get request to OpenWeatherMap to get the latest weather data so you can display it in your own application,

APIs tells you how you can make these requests and how the data that's coming back is structured.

Different Types of APIs

GraphQL

REST API

SOAP

gRPC

There represent different architectural styles for creating an API.

REST APIs

REST stands for **representational state transfer** and was created by computer scientist Roy Fielding.

REST APIs: These use HTTP requests to perform CRUD (Create, Read, Update, Delete) operations on resources, typically using standard HTTP methods (GET, POST, PUT, PATCH, DELETE).

Example

Check the API documentation at:

http://open-notify.org/Open-Notify-API/ISS-Location-Now/

- Make a get request and then use the latitude and longitude in Google maps to know the live location of International Space Station

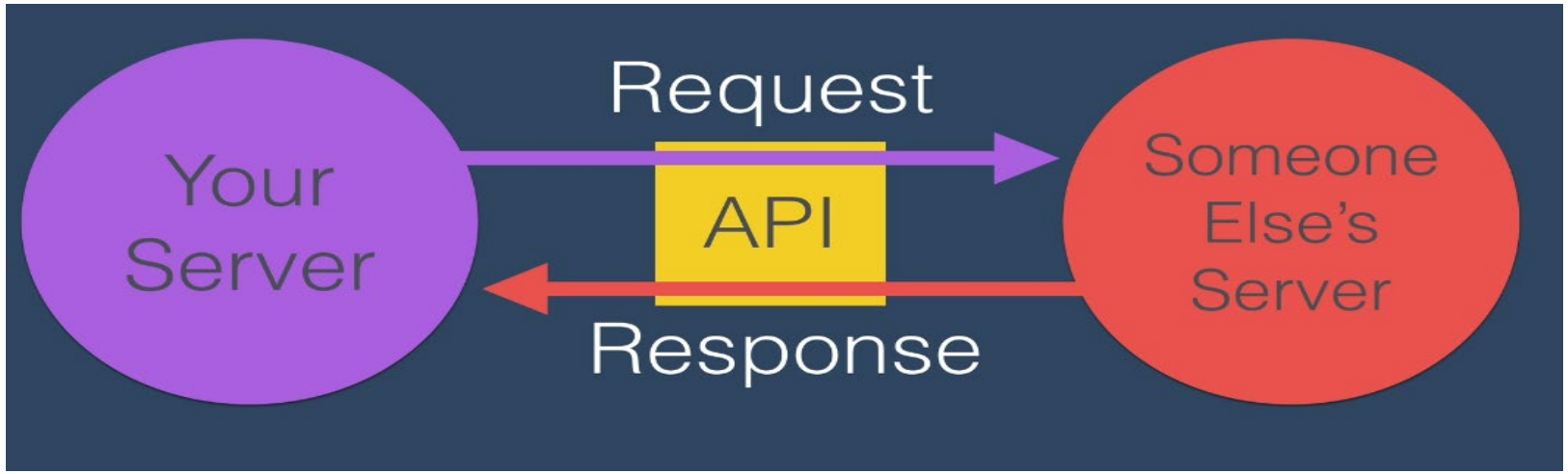
https://bored-api.appbrewery.com/

https://sv443.net/jokeapi/v2/

Why we need API

It’s a contract between the data provider and the developer.

It states that these are all the things that developers can access, and these are the methods, the objects, the protocols that you would use to access them.



API Endpoints

API Endpoints:

An API endpoint is a digital location where an API receives requests about a specific resource on its server.

In APIs, an endpoint is typically a uniform resource locator (URL) that provides the**location of a resource on the server.**

- It is the access point to send the HTTP requests to and get the response.

Example:

https://api.kanye.rest/ sample response {"quote":"Winning is the only option"}

API Endpoints

API Endpoints:

Normally you will have a base URL and then you will have a forward slash and an endpoint.

This endpoint can be different for different purposes.

BaseURL/Endpoint/

Examples

https://bored-api.appbrewery.com/random

https://bored-api.appbrewery.com/filter

An endpoint is a different route on the API provider server and in the documentation they will usually give you an example of which endpoints you can use and what is the purpose of that endpoint.

Query Parameters

In order to allow the API to be flexible enough to deal with custom queries, APIs allow you to provide parameters. And parameters go at the end of the URL, after a question mark, like this.

BaseURL/Endpoint?query=value

Examples:

https://bored-api.appbrewery.com/filter?type=education

Multiple Query Parameters

You can give multiple query parameters separated by &

BaseURL/Endpoint?query1=value1&query2=value2

https://bored-api.appbrewery.com/filter?type=social&participants=2

Path parameters

Path parameters are more for identifying a resource by some specific parameter.

Query parameters as more for filtering and searching and

A path parameter does not take the form of a key/value pair.

https://v2.jokeapi.dev/joke/Programming

https://bored-api.appbrewery.com/activity/5914292

The last one is a path parameter. It comes after the endpoint.
It helps to narrow down on a specific piece of data you want from an external server.

https://v2.jokeapi.dev/joke/Programming?contains=debugging

- We want a programming joke that contained the word '**debugging**'

API Authentication

Some websites provide very simple pieces of data at the endpoints.

Now when developers build more complex applications that might be used by hundreds or thousands of users, then these web sites think about how to either monetize the use of their data or how to limit the use to a threshold.

And the way that they would do that is through **authentication.**

So every time you make a request through the API, they have to be able to identify you as the developer, and they have to keep track of how often you're using their server to get data, and then charge you, or limit you, accordingly.

OpenWeatherMap asks for subscription so you can use their services.

JavaScript Object Notation - JSON

It's a way to format data that can be sent over the internet in a readable but also efficient way.

And as the name suggests, it's structured after a JavaScript object.

Example:

```
{
  "activity": "Host a movie marathon with some friends",
  "type": "social",
  "participants": 3,
  "accessibility": "Few to no challenges",
  "duration": "hours",
  "kidFriendly": true,
}
```

Converting between JSON and JS object

Converting from a JSON object to a Java Script object

JSON → JS Object

const data = JSON.parse(jsonData)

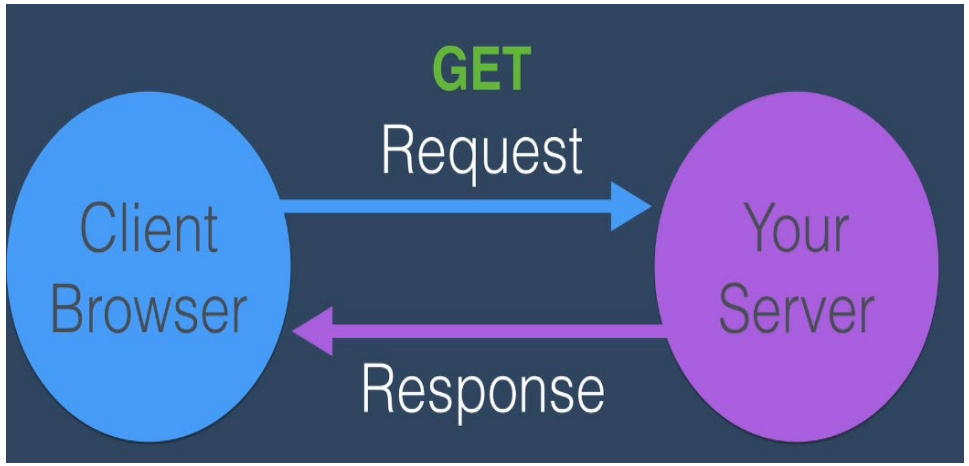
Converting from a Java Script object to a JSON object

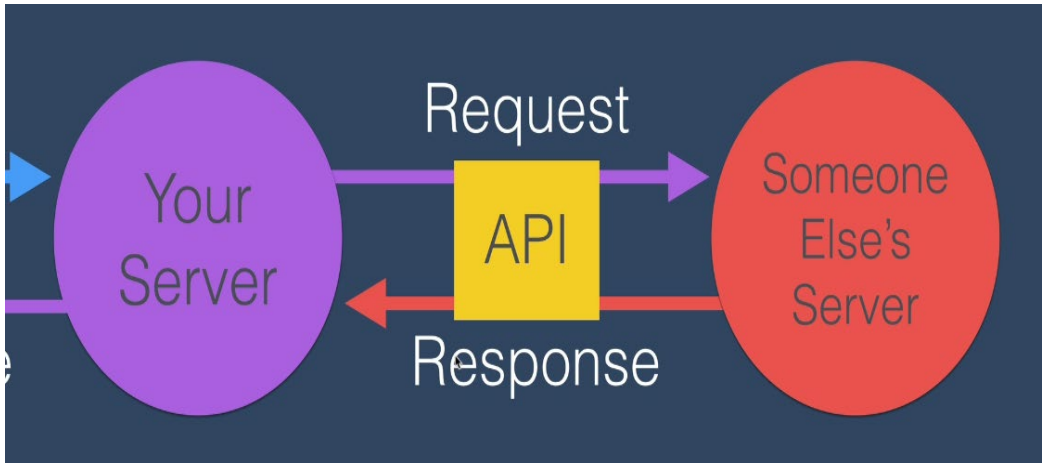
JS Object → JSON

const jsonData = JSON.stringify(data)

Server-side API requests

1. Making requests from your server using Node and HTTPS





Axios

Axios is a promise-based HTTP Client for node.**js** and the browser.

Axios is a JavaScript library used to make HTTP requests from node.js or XMLHttpRequests from the browser that also supports the ES6 Promise API [medium.com]

A **Promise** is an object representing the eventual completion or failure of an asynchronous operation.

Use async/await or .then() to resolve a promise.

A Promise is in one of these states [mdn]

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

Fetch API

Node. js Fetch API is a built-in module in Node. js that enables developers to make API requests and handle responses on the server.

It also provides a simple way to set and retrieve HTTP headers and allows us to customize requests with options such as the request method, headers, and body.

High level overview

1. The client makes a **GET request to our server because** the client wants to get some HTML, CSS and JavaScript files from our server.
2. Now at this point, our server should return all of those pieces of data in the form of an http response.
3. But in order to be able to prepare the response, our server needs some data from somebody else's server, so we're going to make a request to that other server. And it's again going to be a GET request, so that they will give us a response in the form of the data that we need.
4. And we're going to do this via their API, so via the menu that they provided to us where they've specified what are the things that we need to pass over, like parameters, paths, key value pairs in order to get the response and the data that we want.
5. Once we get the data, we can go back and incorporate that data into the files that we can send back to our client.

Implementation

Client makes a **GET request to our server**

```
app.get("/", async (req, res) => {
…
}
```

Implementation

Server sends back an http response to the client.

In this example, server wants to send an hml file that it has fetched from a third party server.

The request is made using axios

```
app.get("/", async (req, res) => {
    try {
        const response = await axios.get("https://bored-api.appbrewery.com/random");
        const result = response.data;
        // console.log(result);
        res.render("solution1.ejs", { data: result })
    } catch (err) {
        console.error("failed to make a request", err.message);
        res.render("index.ejs", { error: err.message });
        //The res. render() function is used to render a view and sends the rendered HTML string to the client
    }
});
```

Implementation

Now the client can make further request like a POST request and ask for some more information.

The server in turn will process the POST request and do more actions:

- Server can send another GET request to a third party server and get more information
- Server then processes the response and send it back to the client.