Axios

Prior Knowledge

- make http requests from node using an npm package, such as `node-fetch`
- use 3rd party apis as a service to manipulate and analyse data

Learning Objectives

- Understand how to use axios to retrieve data.
- Understand how to use the axios params option.
- Understand how to create a custom axios instance.

Using axios to make http requests

In its own words, axios is a:

"Promise based HTTP client for the browser and node.js"

To use axios, it must first be installed in an npm project: `npm install axios`

To use it to perform a `GET` request to a particular url, axios must be imported and the `.get` method invoked with the url.

```
import axios from 'axios';

console.log(axios.get('https://api.lyrics.ovh/v1/oasis/wonderwall'));
```

The above will log `Promise { <pending> }` to the console.

This is because the axios methods return a `promise` when invoked.

As the axios documentation shows, we can access the response or error received after the http response is received by ***chaining*** on the `.then()` and `.catch()` methods. Unlike fetch, axios will reject the returned promise if the status code of the response falls outside of the 2xx range. Useful no?

```
// Make a GET request to a URL
axios
  .get('https://api.lyrics.ovh/v1/oasis/wonderwall')
  .then((response) => {
    // handle success
    console.log(response);
  })
  .catch((error) => {
    // handle error
    console.log(error);
  })
  .then(() => {
    // always executed
    console.log('-----> me last!');
  });

// executed whilst the promise is being resolved/rejected
console.log('-----> me first!');
```

In order to make this **reusable** and **flexible** we could wrap the axios request in a function.

We must ensure that the axios request is returned from the function, otherwise the `getLyrics` function will not be returning a promise, and hence will not be able to invoke the `.then()` or `.catch()` methods.

```
const getLyrics = (artist, track) => {
  return axios
    .get(`https://api.lyrics.ovh/v1/${artist}/${track}`)
    .then((response) => response.data.lyrics);
};

getLyrics('oasis', 'wonderwall')
  .then((lyrics) => {
    console.log(lyrics);
  })
  .catch((err) => {
    console.log(err);
  });
```

Axios Params

Sometimes the request you want to send may have many different combinations of queries based on user inputs. Instead of trying to programmatically build the query ourselves, `axios` provides a way of passing in any possible number of queries and will build up the query string for you.

To pass any extra information to an axios get request, like a body or query, axios takes an optional second argument - a config options object. Inside this object under the key of params is another object. The key value pairs of this params object declares all the queries that will be appended onto the url.

The following will evaluate to a url string of `/data?id=12345&all_data=true`

```
axios.get('/data', {
  params: {
    id: 12345,
    all_data: true,
  },
});
```

This is especially useful when dealing with optional queries. If the value of a query is `undefined` then it will be omitted from the query altogether.

In the example below the filterTerm is undefined so will be omitted. The resulting query will then just contain the id, like so `/data?id=12345`

```
const filterTerm = undefined;

axios.get('/data', {
  params: {
    id: 12345,
    filterTerm,
  },
});
```

Note that for axios post requests, the second argument will be the ***request body***. A config options object in this case could still be passed as a third argument.

```
axios.post('/users', { firstName: 'John', lastName: 'Doe' }, { /* options */ });
```

Axios Instance

The docs show us how we can create a custom axios instance with the same available methods, just with additional configuration. This can be especially useful when we are making many requests to the same api within an application.

```
// axios instance
axios.get('https://nc-pets.com/api/dogs');
axios.get('https://nc-pets.com/api/cats');
axios.get('https://nc-pets.com/api/pigeons');

// axios custom instance
const petsApi = axios.create({
  baseURL: 'https://nc-pets.com/api'
});

petsApi.get('/dogs');
petsApi.get('/cats');
petsApi.get('/pigeons');
```