Data fetching with Fetch

Prior Knowledge

- Understanding of the client/server relationship when making http requests.
- Asynchronous code.

Learning Objectives

- Understand how to make a http request with the `fetch api`.

Fetch API

In order to make http requests in a React app we have the option of importing modules like node's `http` client or npm packages such as `axios`. As our code is run in the browser we can make use of the Fetch API without needing to install any additional packages.

Fetch provides a promise-based interface for making http requests with an excellent use guide on mdn

The first argument to fetch is the url to make the request to and the method will default to `GET`.

```
fetch('https://itunes.apple.com/search?term=beyonce')
  .then((response) => response.json())
  .then((body) => console.log(body));
```

Note that fetch resolves with the response object but with no body. As there are many formats the body could be in the response comes with several methods for parsing the body depending on it's format. Our responses will be in `json` so we must call the `response.json()` method in order to parse that body. There are similar methods for other body formats.

Configuring Requests

To configure the request and change properties such as the method and headers or attach bodies we can pass a second `options` argument to fetch. We add specific keys to this object in order to change the request properties. Some common properties are shown below:

```
fetch('https://www.example.com/api/people', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({ name: 'Paul' }),
}).then((response) => response.json());
```

A full list of options can be found here.

nb The body must be a string, other libraries may do this automatically but with `fetch` we are responsible for stringifying objects.