**HTTP Requests with Fetch API**

JavaScript uses an [event loop](https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop) to handle asynchronous function calls. When a program is run, function calls are made and added to a stack. The functions that make requests that need to wait for servers to respond then get sent to a separate queue. Once the stack has cleared, then the functions in the queue are executed.

Web developers use the event loop to create a smoother browsing experience by deciding when to call functions and how to handle asynchronous events. To make asynchronous event handling easier, promises were introduced in ES6 JS.

1. **GET Requests using Fetch**

The fetch() function:

* Creates a request object that contains relevant information that an API needs (parameters) and sends that request object to the API endpoint provided.
* **Returns a promise** that ultimately resolves to a response object, which contains the status of the promise with information the API sent back.

Text

Description automatically generated

**Note:**

- The promise either resolves or rejects.

Resolves: The ok property of the response object returns a Boolean value 🡪 falsy value if failed.

Rejects: networkError

.json() also returns a promise 🡪 response.json() = jsonPromise

- The second .then() runs with the returned json response from the success.

1. **POST Request using Fetch**

Text

Description automatically generated

- Notice that the fetch() call takes two arguments: an endpoint and an object that contains information needed for the POST request.

- The object passed to the fetch() function as its second argument contains two properties: method, with a value of 'POST', and body, with a value of JSON.stringify({id: '200'});. This second argument determines that this request is a POST request and **what information will be sent to the API.**

- The rest is simlar to GET Request

1. **Async GET Requests**

Text

Description automatically generated

1. **Async POST Requests**

Text

Description automatically generated

Note that since resopnse.json() also returns a promise, we have to add another **await** keyword there.