ASYNC AWAIT

Basics of async await:

When making an API call, the response typically includes information such as the HTTP status code, headers, and the actual data returned by the server. The response can be handled using `async/await` in the following way:

>**Making the API Call:** You use a function that returns a promise, such as `fetch`, `axios`, or `axios.get`, to make the API call. The function sends a request to the server and returns a promise that resolves with the response. For example:

```
async function fetchData() {
  try {
    const response = await fetch('https://api.example.com/data');
    // Handle the response
  } catch (error) {
    // Handle the error
  }
}
```

>**Extracting Data**: Once you have the response, you can extract the data from it. The data might be in JSON format, XML, or any other format depending on the API. In the case of JSON data, we can use the `json()` method on the response object to parse it. For example:

```
async function fetchData() {
  try {
    const response = await fetch('https://api.example.com/data');
    const data = await response.json();
    // Handle the data
  } catch (error) {
    // Handle the error
  }
}
```

>**Error Handling**: If the API call encounters an error, such as a network issue or a server error, it will result in a rejected promise. In such cases, the code execution will jump to the `catch` block where we can handle the error. For example:

```
async function fetchData() {
  try {
    const response = await fetch('https://api.example.com/data');
    const data = await response.json();
    // Handle the data
  } catch (error) {
    console.error('Error:', error);
    // Handle the error
  }
}
```

> Within the `catch` block, we can log the error or perform any necessary error handling, such as displaying an error message to the user or retrying the request.

By using `async/await` in combination with API calls, we can write cleaner and more readable code that waits for the asynchronous operations to complete before moving forward, and handle errors in a straightforward manner.

Here's a breakdown of how `async/await` works in React:

```jsx
import React, { useState } from 'react';
import axios from 'axios';

const MyComponent = () => {
  const [data, setData] = useState(null);

  const fetchData = async () => {
    try {
      const response = await axios.get('https://api.example.com/data');
      setData(response.data);
    } catch (error) {
      console.error('Error:', error);
    }
  };

  return (
    <div>
      <button onClick={fetchData}>Fetch Data</button>
      {data && (
        <div>
          {/* Render the data */}
        </div>
      )}
    </div>
  );
};

export default MyComponent;
```