Handling APIs in React

@developer.mohit

What is Fetch API?

- The fetch API provides an interface for fethcing resources.
- It uses request and response objects
- the fetch() method is used to fetch a resource (data)
- API--> Application Programming Interface based on request response cycle. client do some request from the frontend, API send the request to the server and it retrieve the data from server and send it to the frontend as a response.

Concept of RESTful API

- That adhereas to the principal and constrains of REST (Representational State Transfer).
- REST is an architectural style for designing network application and it relies on a stateless, client-server, cacheable communication, protocol-typically HTTP.
- These are used to interact with web services in a standardized way.

API Handing Methods

- GET --> This method is used to extract data from an API.
- POST --> This method is used to send data to the API.
- DELETE --> Deletes a specified resource
- PATCH --> Partially updates a specified resource.
- PUT --> updates a specified resource with new data.

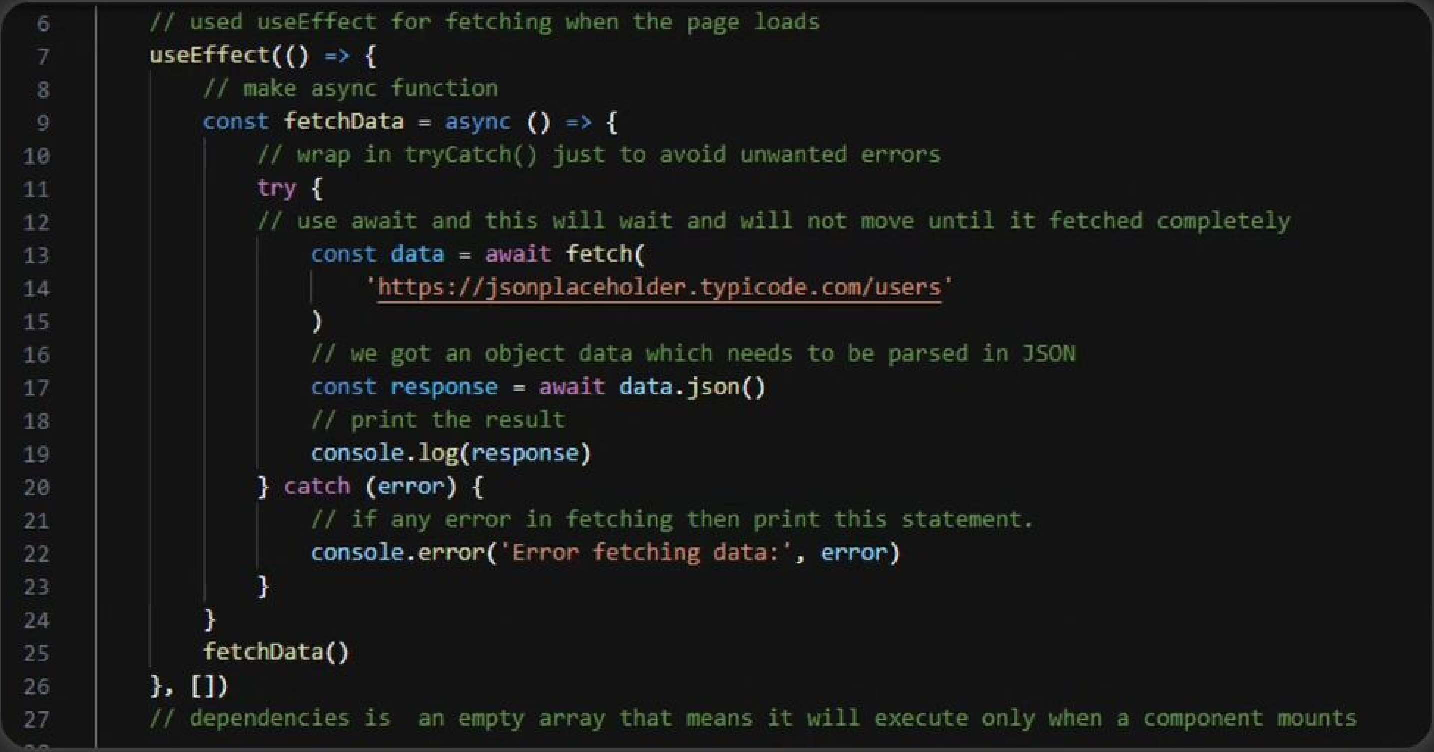
In this post we are going to learn how to do GET request using async/await and promises Rest methods will cover one by one later

- **PUT vs PATCH**

  PUT is altering resources when the client transmits data that revamps the whole resource.

  PATCH is transforming the resources when the client transmits partial data that will be updated without changing the whole data.

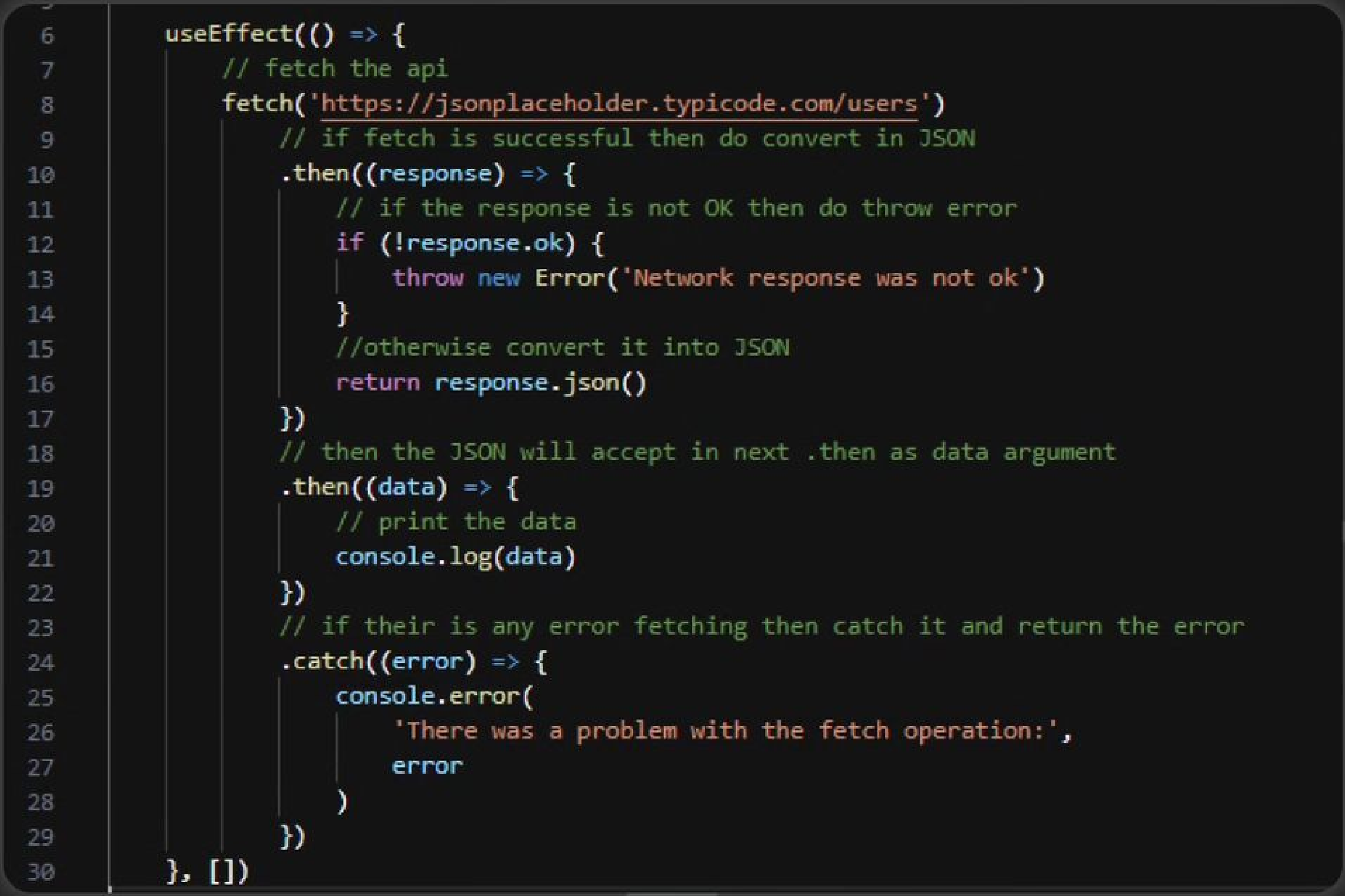
Fetch using async/ await



```
// used useEffect for fetching when the page loads
useEffect(() => {
    // make async function
    const fetchData = async () => {
        // wrap in tryCatch() just to avoid unwanted errors
        try {
        // use await and this will wait and will not move until it fetched completely
            const data = await fetch(
                'https://jsonplaceholder.typicode.com/users'
            )
            // we got an object data which needs to be parsed in JSON
            const response = await data.json()
            // print the result
            console.log(response)
        } catch (error) {
            // if any error in fetching then print this statement.
            console.error('Error fetching data:', error)
        }
    }
    fetchData()
}, [])
// dependencies is  an empty array that means it will execute only when a component mounts
```

@developer.mohit

Fetch using Promises



```javascript
useEffect(() => {
    // fetch the api
    fetch('https://jsonplaceholder.typicode.com/users')
        // if fetch is successful then do convert in JSON
        .then((response) => {
            // if the response is not OK then do throw error
            if (!response.ok) {
                throw new Error('Network response was not ok')
            }
            //otherwise convert it into JSON
            return response.json()
        })
        // then the JSON will accept in next .then as data argument
        .then((data) => {
            // print the data
            console.log(data)
        })
        // if their is any error fetching then catch it and return the error
        .catch((error) => {
            console.error(
                'There was a problem with the fetch operation:',
                error
            )
        })
}, [])
```

@developer.mohit

Fetch using callback

- *Using the Fetch API with callbacks instead of Promises isn't a common pattern in modern JavaScript due to the cleaner syntax and better error handling provided by Promises and async/await.*

Fetch using axios

- *axios is a broader term and will be explained in detail after this concept. From now we will understand how to do other HTTP methods using async/await and promises only. Don’t worry We will learn everything about axios and will avoid using callbacks*