JavaScript is commonly used to interact with APIs to fetch data, submit data, or perform other operations using HTTP requests. Here are some examples of making API calls using JavaScript:

1. **Using the `fetch` API**:
   The `fetch` API is a modern and flexible way to make HTTP requests. It returns a Promise that resolves to the response object.

   ```javascript
   // Define a function to fetch data from an API
   async function fetchData() {
       const url = 'https://jsonplaceholder.typicode.com/todos/1';

       try {
           // Fetch data from the URL
           const response = await fetch(url);

           // Check if the response status is OK (200)
           if (!response.ok) {
               throw new Error(`HTTP error! Status: ${response.status}`);
           }

           // Parse the response as JSON
           const data = await response.json();

           // Log the data
           console.log('Data:', data);
       } catch (error) {
           // Handle errors
           console.error('Error fetching data:', error);
       }
   }

   // Call the function
   fetchData();
   ```

2. **Making a POST request using `fetch`**:
   Here's how you can make a POST request using the `fetch` API.

   ```javascript
   // Define a function to post data to an API
   async function postData() {
       const url = 'https://jsonplaceholder.typicode.com/posts';
   ```

```
    // Define the data to post
    const data = {
        title: 'foo',
        body: 'bar',
        userId: 1
    };

    try {
        // Make a POST request with the data
        const response = await fetch(url, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(data)
        });

        // Parse the response as JSON
        const responseData = await response.json();

        // Log the response data
        console.log('Response data:', responseData);
    } catch (error) {
        // Handle errors
        console.error('Error posting data:', error);
    }
}

// Call the function
postData();
```

3. **Making GET requests to multiple endpoints**:
   Here's an example that shows how to fetch data from multiple endpoints and handle the results concurrently using `Promise.all`.

```javascript
// Define a function to fetch data from multiple URLs
async function fetchMultipleData() {
    // URLs to fetch data from
    const urls = [
        'https://jsonplaceholder.typicode.com/todos/1',
        'https://jsonplaceholder.typicode.com/posts/1',
```

```
        'https://jsonplaceholder.typicode.com/comments/1'
    ];

    try {
        // Use Promise.all to fetch data from all URLs concurrently
        const responses = await Promise.all(urls.map(url => fetch(url)));

        // Parse the responses as JSON
        const data = await Promise.all(responses.map(response => response.json()));

        // Log the data from each endpoint
        console.log('Data from endpoints:', data);
    } catch (error) {
        // Handle errors
        console.error('Error fetching data:', error);
    }
}

// Call the function
fetchMultipleData();
```

These examples demonstrate how to use the `fetch` API to interact with APIs in JavaScript. You can use the `fetch` API for various HTTP methods such as `GET`, `POST`, `PUT`, `DELETE`, and more, by specifying the appropriate method and options in the request.