Fetch API in JavaScript

The **Fetch API** provides a modern way to make HTTP requests in JavaScript. It returns a **Promise**, making it easier to handle asynchronous requests compared to older methods like `XMLHttpRequest`.

Basic Syntax

```
fetch(url, options)
  .then(response => {
    // handle response
  })
  .catch(error => {
    // handle error
  });
```

- `url`: The resource you want to fetch.
- `options`: (Optional) An object that contains settings such as method, headers, and body.

Example: GET Request

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then(response => {
    if (!response.ok) {
      throw new Error("Network response was not ok");
    }
    return response.json();
  })
```

```
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.error("Fetch error:", error);
  });
```

Example: POST Request

```
fetch("https://jsonplaceholder.typicode.com/posts", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    title: "foo",
    body: "bar",
    userId: 1,
  }),
})
  .then(response => response.json())
  .then(data => {
    console.log("Created:", data);
  })
  .catch(error => {
    console.error("Error:", error);
  });
```

Using `async/await`

```
async function fetchData() {
  try {
    const response = await fetch("https://jsonplaceholder.typicode.com/posts/1");
```

```
    if (!response.ok) {
      throw new Error("Network response was not ok");
    }
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error("Fetch error:", error);
  }
}

fetchData();
```

Key Points

- `fetch` returns a Promise that resolves to a `Response` object.
- You need to call methods like `.json()`, `.text()`, or `.blob()` to read the response body.
- Errors only occur on **network failure** or if `response.ok` is false.
- Supports request configuration with methods, headers, and body.