Introducción a `fetch` en JavaScript

El objeto `fetch` en JavaScript es una función incorporada que proporciona una forma fácil y moderna de realizar peticiones de red y recuperar recursos a través de la web. Es ampliamente utilizado para obtener y enviar datos utilizando protocolos como HTTP y HTTPS.

Sintaxis básica de `fetch`

La sintaxis básica para utilizar `fetch` es la siguiente:

```
fetch(url)
  .then(response => {
    // Código para manejar la respuesta
  })
  .catch(error => {
    // Código para manejar errores
  });
```

Aquí está el significado de cada parte:

- `fetch(url)` : Invoca la función `fetch` pasando la URL del recurso que deseas recuperar. Esto devuelve una promesa que representa la respuesta a la petición de red.
- `.then(response => { ... })` : Utiliza el método `.then()` para manejar la respuesta exitosa de la petición. La respuesta se pasa como argumento a la función de devolución de llamada, donde puedes realizar acciones como leer los datos de la respuesta, procesarlos o mostrarlos en la página.
- `.catch(error => { ... })` : Utiliza el método `.catch()` para manejar cualquier error que ocurra durante la petición. El error se pasa como argumento a la función de devolución de llamada, donde puedes manejarlo de manera apropiada.

Ejemplos de uso de `fetch`

A continuación, se presentan algunos ejemplos de cómo utilizar `fetch` para realizar diferentes tipos de peticiones:

Obtener datos de una API

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => {
    // Código para manejar los datos obtenidos
    console.log(data);
  })
  .catch(error => {
    // Código para manejar errores
    console.error('Error:', error);
  });
```

En este ejemplo, se realiza una petición a una API utilizando `fetch` y se espera que la respuesta sea un objeto JSON. Utilizamos el método `.json()` en la respuesta para convertir los datos en formato JSON en un objeto JavaScript utilizable.

Enviar datos a un servidor

```
const data = { username: 'John', password: 'secret' };

fetch('https://api.example.com/login', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(data)
})
  .then(response => response.json())
  .then(result => {
    // Código para manejar la respuesta del servidor
```

```
    console.log(result);
  })
  .catch(error => {
    // Código para manejar errores
    console.error('Error:', error);
  });
```

En este ejemplo, se envían datos al servidor utilizando el método HTTP POST. Los datos se pasan en el cuerpo de la petición como una cadena JSON utilizando `JSON.stringify()`.

Manejar errores de red

```
fetch('https://api.example.com/data')
  .then(response => {
    if (!response.ok) {
      throw new Error('Error de red');
    }
    return response.json();
  })
  .then(data => {
    // Código para manejar los datos obtenidos
    console.log(data);
  })
  .catch(error => {
    // Código para manejar errores
    console.error('Error:', error);
  });
```

En este ejemplo, se verifica si la respuesta de la petición tiene un estado no exitoso (por ejemplo, un código de respuesta HTTP 404). Si la respuesta no es exitosa, se lanza un error personalizado para manejarlo en el bloque `.catch()`.

Conclusión

`fetch` es una poderosa función de JavaScript que simplifica la realización de peticiones de red y el manejo de respuestas. Permite obtener y enviar datos de manera eficiente, así

como manejar errores de manera adecuada. ¡Aprovecha `fetch` para trabajar con servicios web y API de forma sencilla y efectiva!