React intermedio

Comunicación con el servidor

Uso del Fetch nativo del navegador

OpenWebinars

ÍNDICE

Realizando un GET básico con Fetch

- Una petición **GET** es equivalente a la que realiza el navegador cuando pedimos una URL a través de la barra de direcciones
- Cualquier petición **GET** consta de **Cabeceras**, **Dirección** y **Parámetros** (opcionales)
- Como todas las peticiones con **Fetch,** se produce de manera asíncrona y su respuesta viene en forma de objeto **Promise**
- La respuesta podrá venir en cualquier formato y dependerá del servidor, y en ocasiones, de la cabecera de contenido que el servidor reciba

Realizando un GET básico con Fetch

```
fetch('https://jsonplaceholder.typicode.com/posts').then(response => {
    // Como la web responde en formato JSON, devolvemos la respuesta parseada
    return response.json()
}).catch(error => {
    // Aquí podemos gestionar el error de la petición
});

fetch('http://www.geoplugin.net/xml.gp?base_currency=EUR').then(response => {
    // Como la web responde en formato XML, devolvemos la respuesta como texto
    return response.text()
}).catch(error => {
    // Aquí podemos gestionar el error de la petición
});
```

- El bloque **catch** de la promesa devuelta sólo se ejecutará en el caso de que haya un error en la petición, **PERO NO** en el caso de que la respuesta presente un código de estado distinto al de éxito (**2XX**)

Realizando un POST básico con Fetch

- Una petición **POST** es equivalente a la que realiza un formulario HTML básico por defecto
- Cualquier petición **POST** consta de **Cabeceras**, **Dirección, Parámetros y Cuerpo o Body** (Estos dos últimos opcionales)
- Las peticiones **POST** permiten un mayor número de estructuras para envío a través del **Body**, como archivos
- También las peticiones de tipo **POST** permiten un volumen de datos limitado únicamente por el servidor mientras las **GET** están limitadas a 2000 caracteres

Realizando un POST básico con Fetch

```
fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    body: JSON.stringify({
      title: 'Un título',
      body: 'Un cuerpo del artículo',
      userId: 1,
    }),
    headers: {
      'Content-type': 'application/json; charset=UTF-8',
    }
})
.then((response) => response.json());
```

- Importante poner la cabecera de contenido para que el servidor sepa que el Body de la petición se va a recibir en JSON.
- También indicar el charset, sobre todo si van a usarse caracteres especiales.

Objeto init: Opciones de la petición

```
{
  method: 'POST', // GET, POST, PUT, DELETE, y cualquier método HTTP.
  mode: 'cors', // no-cors, *cors, same-origin
  cache: 'no-cache', // *default, no-cache, reload, force-cache, only-if-cached
  credentials: 'same-origin', // include, *same-origin, omit
  headers: {
    // Las cabeceras de la petición como objeto de clave nombre de la cabecera y valor
     el valor de la cabecera
    'Content-Type': 'application/json'
  },
  redirect: 'follow', // Qué hacer cuando haya redirección (manual, *follow, error)
  referrerPolicy: 'no-referrer', // (no-referrer, *no-referrer-when-downgrade, origin,
origin-when-cross-origin, same-origin, strict-origin, strict-origin-when-cross-origin,
unsafe-url)
  // El body de la petición, debe de corresponderse con la  cabecera 'Content-Type'
    que enviemos
  body: JSON.stringify(data)
}
```

Cabeceras y respuesta de la petición

- Las cabeceras de la petición determinan factores importantes para que la petición tenga éxito y esté bien formada
- En el caso de Fetch se administran con la interfaz iteradora **Headers** que posee métodos comunes tales como **append()**, **entries()** o **has()**
- La cabecera **Content-Type** no sólo será informativa para el servidor, sino que la propia API la usará para saber cómo enviar la petición
- Otras cabeceras importantes como **Authorization** pueden utilizarse por el servidor para indicar el tipo de autenticación y el token a usar, Ej: **Authorization: Basic YffDGRpgFGvcGVuc2VzY21l**

Cabeceras y respuesta de la petición

```
fetch('https://jsonplaceholder.typicode.com/posts/1').then(function(response) {
    // Acceso a las headers de la respuesta, mediante get() podemos ver cabeceras
       independientes
    console.log(response.headers.get('Content-Type'));
    console.log(response.headers.get('cache-control'));
    // La variable status almacena el codigo de estado HTTP (200, 401, 500...etc)
    console.log(response.status);
    // La variable statusText almacena el literal asociado al estado HTTP (200 => OK,
      400 => Bad Request...etc)
    console.log(response.statusText);
    // La variable ok devuelve un booleano indicando si la respuesta fue exitosa (Está
     en el rango de respuesta 200-299) o no
    console.log(response.ok);
    // La variable type devuelve el tipo de respuesta de servidor (cors, basic...)
    console.log(response.type);
    // La variable url almacena la URL utilizada para la respuesta, en este caso es la
     misma de la petición, pero puede cambiar si existen redirecciones intermedias
    console.log(response.url);
    // Al utilizar el método return con json() devolvemos una nueva promesa cuya
    respuesta estará parseada en Javascript, si usamos .text() la devolveremos como un
     string y en el caso de .blob() devolveremos un objeto Blob de Javascrippt
    return response.json();
});
```

PARA RESUMIR

- Los dos tipos principales de peticiones o request más usadas son las tipo **POST** y las tipo **GET**, ambas tienen limitaciones distintas
- Las peticiones **GET** tienen **dirección**, **cabeceras**, **y parámetros** mientras que las **POST** aparte de esto también tienen **body (o cuerpo)**
- Las peticiones con Fetch están compuestas por un objeto **init** con unas cabeceras **Headers** y devuelven una respuesta en formato **Response**
- El éxito de una petición dependerá tanto de elegir correctamente las **cabeceras** como del envío del **body** así como del procesamiento de la **respuesta**