DESARROLLO DE APLICACIONES WEB FRONT END

Módulo 4

OTEC PLATAFORMA 5 CHILE

MÓDULO 4

CLASE 1

Introducción

Fetch

Sintaxis

Response

Peticiones POST

Introducción

Comenzamos con el Módulo 4, y con esto nos acercamos a la cima del aprendizaje.

¡MOMENTO! ¿Qué quiere decir esto? Quiere decir que en la montaña de nuestro Bootcamp, ya escaló la parte más compleja, es decir caminó en subida todo este tiempo. De ahora en más, todo se vuelve más llevadero. Y con llevadero nos referimos a que los temas que estudiará no son tan complejos y son complementarios a todo lo que vinimos trabajando. Por esta razón, podrá asimilarlos con mayor facilidad que cuando comenzó en el Módulo 01.

Sin más preámbulos... Partimos...

Fetch

JavaScript puede enviar peticiones de red al servidor y cargar nueva información siempre que se necesite.

Por ejemplo, podemos utilizar una petición de red para:

- **Crear una orden**
- **Cargar información de usuario**
- **Recibir las últimas actualizaciones desde un servidor**

...Y todo esto sin la necesidad de refrescar la página.

Existen múltiples maneras de enviar peticiones de red y obtener información de un servidor. Comenzaremos con el el método fetch() que es moderno y versátil. Este método no es soportado por navegadores antiguos (sin embargo se puede incluir un polyfill), pero es perfectamente soportado por los navegadores actuales y modernos.

Sintaxis del Fetch

```
let promise = fetch(url, [options])
```

- **url: representa la dirección URL a la que deseamos acceder.**
- **options: representa los parámetros opcionales, como puede ser un método o los encabezados de nuestra petición.**

Si no especificamos ningún options, se ejecutará una simple petición GET, la cual descargara el contenido de lo especificado en el url.

El navegador lanzará la petición de inmediato y devolverá una promesa (promise) que luego será utilizada por el código invocado para obtener el resultado.

Por lo general, obtener una respuesta es un proceso de dos pasos.

Sintaxis del Fetch

Primero, la promesa promise, devuelta por fetch, resuelve la respuesta con un objeto de la clase incorporada Response tan pronto como el servidor responde con los encabezados de la petición.
En este paso, podemos chequear el status HTTP para poder ver si nuestra petición ha sido exitosa o no, y chequear los encabezados, pero aún no disponemos del cuerpo de la misma.

La promesa es rechazada si el fetch no ha podido establecer la petición HTTP, por ejemplo, por problemas de red o si el sitio especificado en la petición no existe. Estados HTTP anormales, como el 404 o 500 no generan errores.

Podemos visualizar los estados HTTP en las propiedades de la respuesta:

- status – código de estado HTTP, por ejemplo: 200.
- ok – booleana, true si el código de estado HTTP es 200 a 299.

Ejemplo:

```
let response = await fetch(url);

if (response.ok) {
 // si el HTTP-status es 200-299
 // obtener cuerpo de la respuesta
(método debajo)
 let json = await response.json();
} else {
 alert("Error-HTTP: " +
response.status);
}
```

Response

Segundo, para obtener el cuerpo de la respuesta, necesitamos utilizar un método adicional.

Response provee múltiples métodos basados en promesas para acceder al cuerpo de la respuesta en distintos formatos:

response.text(): lee y devuelve la respuesta en formato texto,

response.json(): convierte la respuesta como un JSON

response.body: es un objeto ReadableStream, el cual nos permite acceder al cuerpo como si fuera un stream y leerlo por partes.

Estos son solo algunos de los métodos para las respuestas.

Por ejemplo, si obtenemos un objeto de tipo JSON con los últimos commits de GitHub:

```
let url =
"https://api.github.com/repos/javascript-tutorial/en.javascript.info/commits";

let respuesta = await fetch(url);

let commits = await
respuesta.json(); // leer respuesta del cuerpo y devolver como JSON

alert(commits[0].author.login);
```

Importante:

Puede elegir un solo método de lectura para el cuerpo de la respuesta.

Si ya obtuvimos la respuesta con *response.text()*, entonces *response.json()* no funcionará, dado que el contenido del cuerpo ya ha sido procesado.

Peticiones *POST*

Para ejecutar una petición POST, o cualquier otro método, utilizaremos las opciones de fetch:

- **method: método HTTP, por ej: POST,**
- **body: cuerpo de la respuesta, cualquiera de las siguientes:**
  - **cadena de texto (ej. JSON-encoded)**
  - **Objeto FormData, para enviar información como multipart/form-data**
  - **URLSearchParams, para enviar información en cifrado x-www-form-urlencoded (no utilizado frecuentemente).**

El formato JSON es el más utilizado.

Tener en cuenta, si la respuesta del body es una cadena de texto, entonces el encabezado Content-Type será especificado como text/plain;charset=UTF-8 por defecto.

Pero, cómo vamos a enviar un objeto JSON, en su lugar utilizaremos la opción headers especificada a application/json, que es la opción correcta Content-Type para información en formato JSON.

Por ejemplo, el código debajo envía la información user como un objeto JSON:

```
let user = {
 nombre: "Flor",
 apellido: "Páez",
};

let respuesta = await
fetch("/article/fetch/post/user", {
 method: "POST",
 headers: {
   "Content-Type":
"application/json;charset=utf-8",
 },
 body: JSON.stringify(user),
});

let result = await respuesta.json();
alert(result.message);
```

MÓDULO 4

GRACIAS

Aquí finaliza la clase n°1 del módulo 4

OTEC PLATAFORMA 5 CHILE