Fetch API - Body Data

Fetch API is modern technology to send or receive data asynchronously without refreshing a web page. It provides an interface to create HTTP requests in the web browser. It is supported by almost all modern web browsers. We can also say that, by using the Fetch API we can fetch resources like JSON data, HTML pages, etc from the web server and can send data to the server using different HTTP requests like PUT, POST, etc. So in this article, we will learn what is body data, and how we are we are going to use body data.

Body Data

In Fetch API, both request and response contain body data. Body data in the request is an instance which contains the data which we want to send to the server whereas body data in the response is an instance which contains the data requested by the user. It is generally used by PUT or POST requests to send data to the server. It can be an instance of ArrayBuffer, TypedArray, DataView, Blob, File, String, URLSearchParams, or FormData. While sending body data you also need to set a header in the request so that the server will know what type of the data is.

The Request and Response interface provides various methods to extract the body and they are –

- **Request. arrayBuffer()** – This method is used to resolve a promise with ArrayBuffer representation of the request body.
- **Request.blob()** – This method is used to resolve a promise with a blob representation of the request body.
- **Request.formData()** – This method is used to resolve a promise with formData representation of the request body.
- **Request.json()** – This method is used to parse the request body as JSON and resolve a promise with the result of parsing.
- **Request.text()** – This method is used to resolve a promise with a text representation of the request body.
- **Response.arrayBuffer()** – This method is used to return a promise which will resolve with an ArrayBuffer representation of the response body.
- **Response.blob()** – This method is used to return a promise which will resolve with a Blob representation of the response body.
- **Response.formData()** – This method is used to return a promise which will resolve with a FormData representation of the response body.

- **Response.json()** – This method is used to parse the response body as JSON and return a promise that will resolve with the result of parsing.
- **Response.text()** – This method is used to return a promise which will resolve with a text representation of the response body.

All these methods return a promise which will resolve with the actual content of the body.

Body data is generally used with the fetch() function. Here it is optional you can only use the body parameter when you want to send data to the server.

Syntax

```
fetch(resourceURL,{
   Method: 'POST',
   body:{
      Name: "Monika",
      Age: 34,
      City: "Pune"
   },
   headers: {'content-Type':'application/json'}
})
```

The parameters of the fetch() function –

- **resourceURL** – It represents the resource which we want to fetch. It can be a string, a request object, or a URL of the resource.
- **method** – It represents the request method such as GET, POST, PUT and DELETE.
- **headers** – It is used to add a header to your request.
- **body** – It is used to add data to your request. It is not used by GET or HEAD methods.

In the following program, we send body data using the POST method. So we create an HTML code in which we send data using JavaScript script to the server. In the script, we define a fetch() function which sends the data present in the body parameter to the given URL using the POST request method. Here the header is set to "application/json" which indicates that we are sending data. Before sending the request to the server we convert the data into JSON string with the help of JSON.stringify() function. After receiving the response from the server, we check if the response is ok or not. If yes, then we parse the response body into JSON using the response.json() function and then display the result on the output screen. If we get any error, then the error is handled by the catch() block.

Example

</> Open Compiler

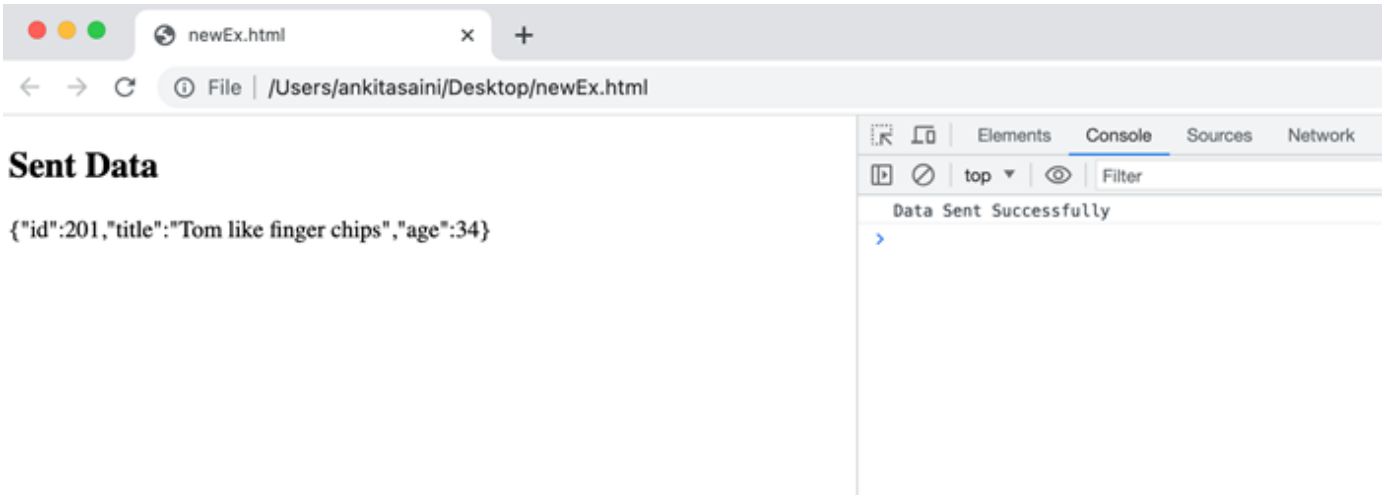
```
<!DOCTYPE html>
<html>
<body>
<script>
   // Retrieving data from the URL using the POST request
   fetch("https://jsonplaceholder.typicode.com/todos", {
      // Adding POST request
      method: "POST",

      // Adding body which we want to send
      body: JSON.stringify({
         id: 45,
         title: "Tom like finger chips",
         age: 34
      }),
      // Adding header
      headers:{"Content-type": "application/json; charset=UTF-8"}
   })
   // Converting received information into JSON
   .then(response =>{
      if (response.ok){
         return response.json()
      }
   })
   .then(myData => {
      // Display the retrieve Data
      console.log("Data Sent Successfully");

      // Display output
      document.getElementById("sendData").innerHTML = JSON.stringify(myData);
   }).catch(err=>{
      console.log("Found error:", err)
   });
```

```
</script>
   <h2>Sent Data</h2>
   <div>
      <!-- Displaying retrevie data-->
      <p id = "sendData"></p>
   </div>
</body>
</html>
```

Output



Conclusion

So, this is how we can use Body Data in Fetch API. Using the data body we can send data from the web browser to the web server or vice versa. In the request body, body data is only used with PUT and POST request methods because using this request we can send data to the server. It is not used with GET request because GET request is used to fetch data from the server. Now in the next article, we will learn Credentials in Fetch API.