AJAX - FormData Object

In AJAX, the FormData object allows you to create a set of key-value pairs which represent the form fields and their values, which can be sent using XMLHttpRequest. It is mainly used in sending form data but can also be used independently to send data. The data transmitted by the FormData object is in the same format as the data sent by the form's submit method.

To create a new FormData object AJAX provide FormData() constructor.

Syntax

```
const objectName = new FormData()
Or
const objectName = new FormData(form)
Or
const objectName = new FormData(form, mSubmit)
```

Where the FormData() can be used with or without parameters. The optional parameters used by the FormData() constructor are –

form – It represents an HTML <form> element. If the FormData object has this parameter, then the object will be populated with the current key-value pair of the form using the name property of each element for the key and their submitted value. It also encodes the input content of the file.

mSubmit – It represents the submit button the form. If mSubmit has a name attribute or an <input type = "image>, then its content will include in the FormData object. If the specified mSubmit is not a button, then it will throw a TypeError exception. If the mSubmit is not a member of the given form, then it will throw NotFoundError.

Methods

FormData object supports the following methods –

Sr.No.	Method Name & Description
1	**FormData.append()** This method is used to append a new value into an existing key. Or can add a new key if it is not present.

2	**FormData.delete()** This method is used to delete key-value pairs.
3	**FormData.entries()** This method returns an iterator that iterates through key-value pair.
4	**FormData.get()** This method returns the first value related to the given key from the FormData object.
5	**FormData.getAll()** This method is used to return an array of all the values related to the given key in the FormData object.
6	**FormData.has()** This method checks whether a FormData object contains the specified key or not.
7	**FormData.keys()** This method returns an iterator which iterates through all the keys of the key-value pairs present in the FormData object.
8	**FormData.set()** This method sets a new value for the existing key in the FormData object. Or can add new key/value if not exists.
9	**FormData.values()** This method returns an iterator which iterates through all the values present in the FormData object.

Creating FormData Object

To create and use FormData Object without using HTML form follow the following steps –

Step 1 – Create an XMLHttpRequest object using XMLHttpRequest () constructor –

```
var zhttp = new XMLHttpRequest();
```

Step 2 – Create a FormData object using FormData constructor –

```
const myForm = new FormData()
```

Step 3 – Use append() method to add key and value pairs –

```
myForm.append("KeyName", "keyValue")
```

Step 4 – Create a call back function to handle response

```
zhttp.onreadystatechange = function() {
   // Body
}
```

Step 5 – Now we use open() function. Inside open() function we pass a POST request along with the server URL we have to post our form data.

```
zhttp.open("POST", url, async)
```

Step 6 – So finally we use send() function to send requests to the server along with the FormData object.

```
zhttp.send(myForm);
```

Now let's discuss this with the help of the example –

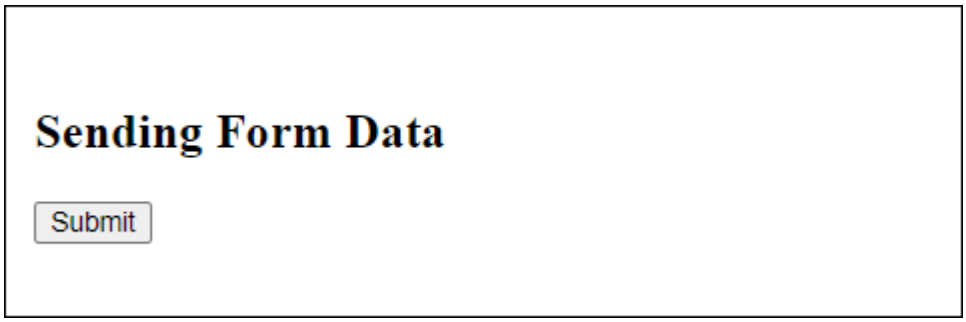
Example 1

</> Open Compiler

```
<!DOCTYPE html>
<html>
<body>
<script>
   function dataDoc() {
      // Creating XMLHttpRequest object
      var zhttp = new XMLHttpRequest();

      // Creating FormData object
      const myForm = new FormData();

      // Assigning the form data object with key/value pair
      myForm.append("title", "AJAX Tutorial")
      myForm.append("UserId", "232")
      myForm.append("Body", "It is for web development")
      myForm.append("Age", "33")
```

```
      // Creating call back function to handle the response
      zhttp.onreadystatechange = function() {
         if (this.readyState == 4 && this.status == 201) {
            document.getElementById("example").innerHTML = this.responseText;
            console.log("Form Data Posted Successfully")
         }
      };
      // Specify the method as POST, URL, and set async to true
      zhttp.open("POST", "https://jsonplaceholder.typicode.com/todos", true);

      // Sending the request to the server
      zhttp.send(myForm);
   }
</script>
<h2>Sending Form Data</h2>
<button type="button" onclick="dataDoc()">Submit</button>
<div id="example"></div>
</body>
</html>
```

Output



So when the user clicks on the "Submit" button dataDoc() function is called. Then dataDoc() function first creates a new XHR object and a new FormData object. Then add new key-value pairs in the FormData object using the append() method. Next, it set up a call-back function which handles the response from the server. This function is triggered when the value of the readyState property = 4 and the value of the Status property = 201. Finally, it calls the open() method and initialises it with the HTTP POST method with the URL of the server and at last it calls send() method to send the FormData request to the server.

So when the response comes from the server the call-back function displays the result and prints the "Form Data Posted Successfully" message on the console log.

Example 2

</> Open Compiler

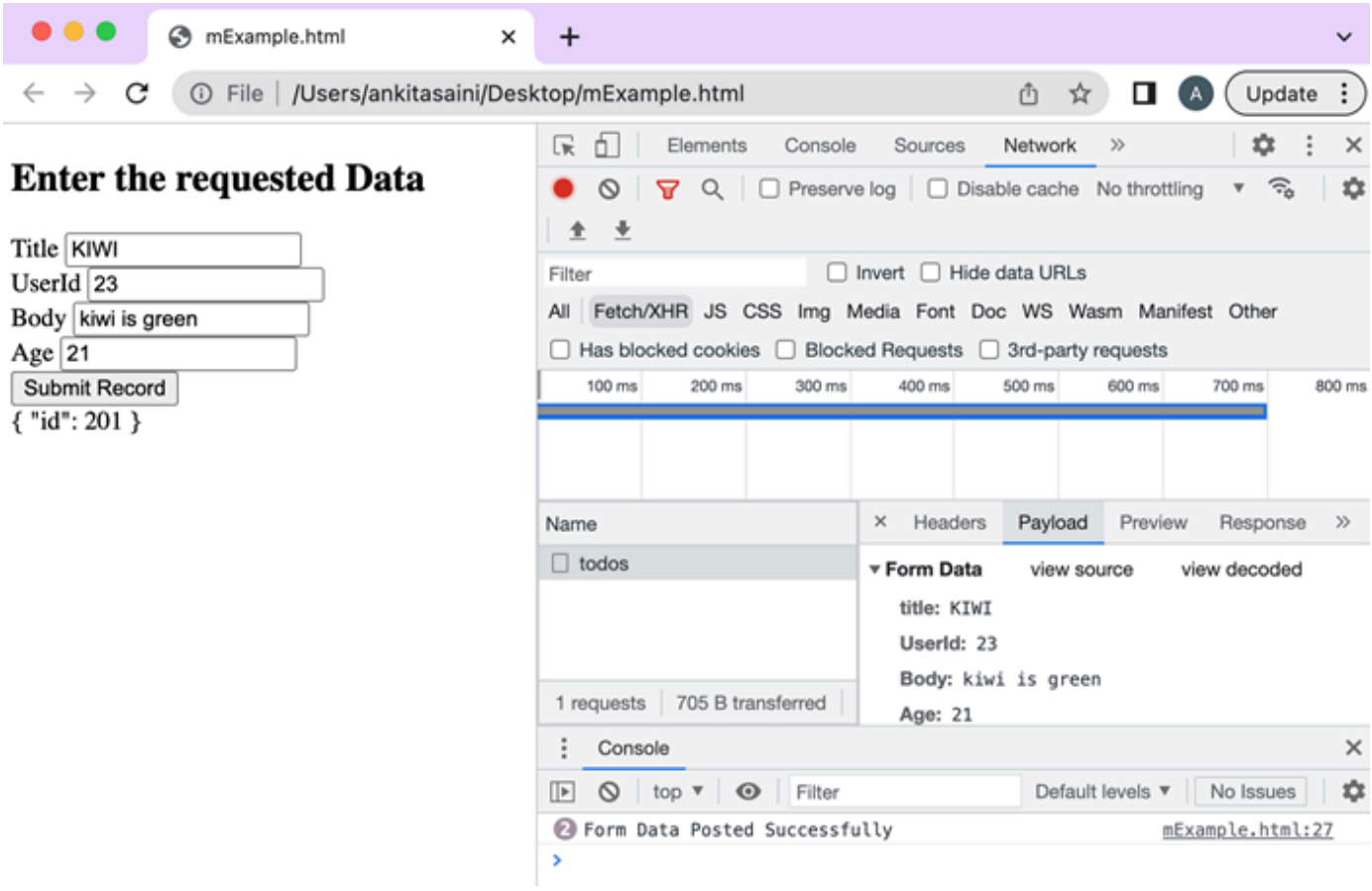
```
<!DOCTYPE html>
<html>
<body>
<script>
   function sendFormData() {
      // Creating XMLHttpRequest object
      var zhttp = new XMLHttpRequest();

      // Creating FormData object
      const myForm = new FormData();

      // Assigning the form data with key/value pair
      myForm.append("title", document.querySelector('#Utitle').value)
      myForm.append("UserId", document.querySelector('#UId').value)
      myForm.append("Body", document.querySelector('#Ubody').value)
      myForm.append("Age", document.querySelector('#Uage').value)

      // Creating call back function to handle the response
      zhttp.onreadystatechange = function() {
         if (this.readyState == 4 && this.status == 201) {
            document.getElementById("example").innerHTML = this.responseText;
            console.log("Form Data Posted Successfully")
         }
      };
      // Post/Add form data on the server
      zhttp.open("POST", "https://jsonplaceholder.typicode.com/todos", true);

      // Sending the request to the server
      zhttp.send(myForm);
   }
</script>
   <!--Creating simple form-->
   <h2>Enter the requested Data</h2>
   <label for="Utitle">Title</label>
   <input id="Utitle" type="text" name="title"><br>

   <label for="UId">UserId</label>
```

```
    <input id="UId" type="number" name="UserID"><br>

    <label for="Ubody">Body</label>
    <input id="Ubody" type="text" name="body"><br>

    <label for="Uage">Age</label>
    <input id="Uage" type="number" name="age"><br>

    <button type="button" onclick="sendFormData()">Submit Record</button>
    <div id="example"></div>
</body>
</html>
```

Output

Here in the below image after entering details when we click on the submit button the data will send to the server and the server returns the id and display message in the console.

Here in the above code, we collect data from the user and submit the data using JavaScript with XMLHttpRequest.



So when the user clicks on the "Submit Record" button sendFormData() function is called. The sendFormData() function first creates a new XHR object and a new FormData object. It appends the form data that was keys and its values are input by the user using the append() method. Next, it set up a call-back function which handles the response from the server. This function is triggered when the value of the readyState property = 4 and the value of the Status property = 201. Finally, it calls the open() method and initialises it with the HTTP POST method with the URL of the server and at last it calls send() method to send the FormData request to the server.

The response from the server the callback function shows the result and prints the message on the console log.

Conclusion

So this is how we can use the FormData object. It is also an important object for storing various types of data like files, plain text, JSON documents, etc. Now in the next article, we will learn how to send POST requests using XMLHttpRequest.