AJAX - Send PUT Requests

The PUT request is used to update data on the web server. In this request, the data is sent in the request body and the web server will replace the existing data with the new data. If the specified data does not exist, then it will add the replacement data as a new record in the server.

PUT request is quite different from the POST request in terms of the following points −

- PUT is used to update existing records whereas POST is used to add new records in the server.
- PUT request can be cached whereas POST request cannot be cached.
- PUT request is idempotent whereas POST request is non-idempotent.
- PUT request works as a specific whereas POST request works as an abstract.

Syntax

```
open('PUT', url, true)
```

Where, the open() method takes three parameters −

- **PUT** − It is used to update data on the web server.
- **url** − url represents the file url or location which will be opened on the web server.
- **true** − For asynchronous connection set the value to true. Or for synchronous connection set the value to false. The default value of this parameter is true.

How to send PUT Request

To send the PUT request we need to follow the following steps −

Step 1 − Create an object of XMLHttpRequest.

```
var variableName = new XMLHttpRequest()
```

Step 2 − After creating XMLHttpRequest an object, now we have to define a callback function which will trigger after getting a response from the web server.

```
XMLHttpRequestObjectName.onreadystatechange = function(){
   // Callback function body
}
```

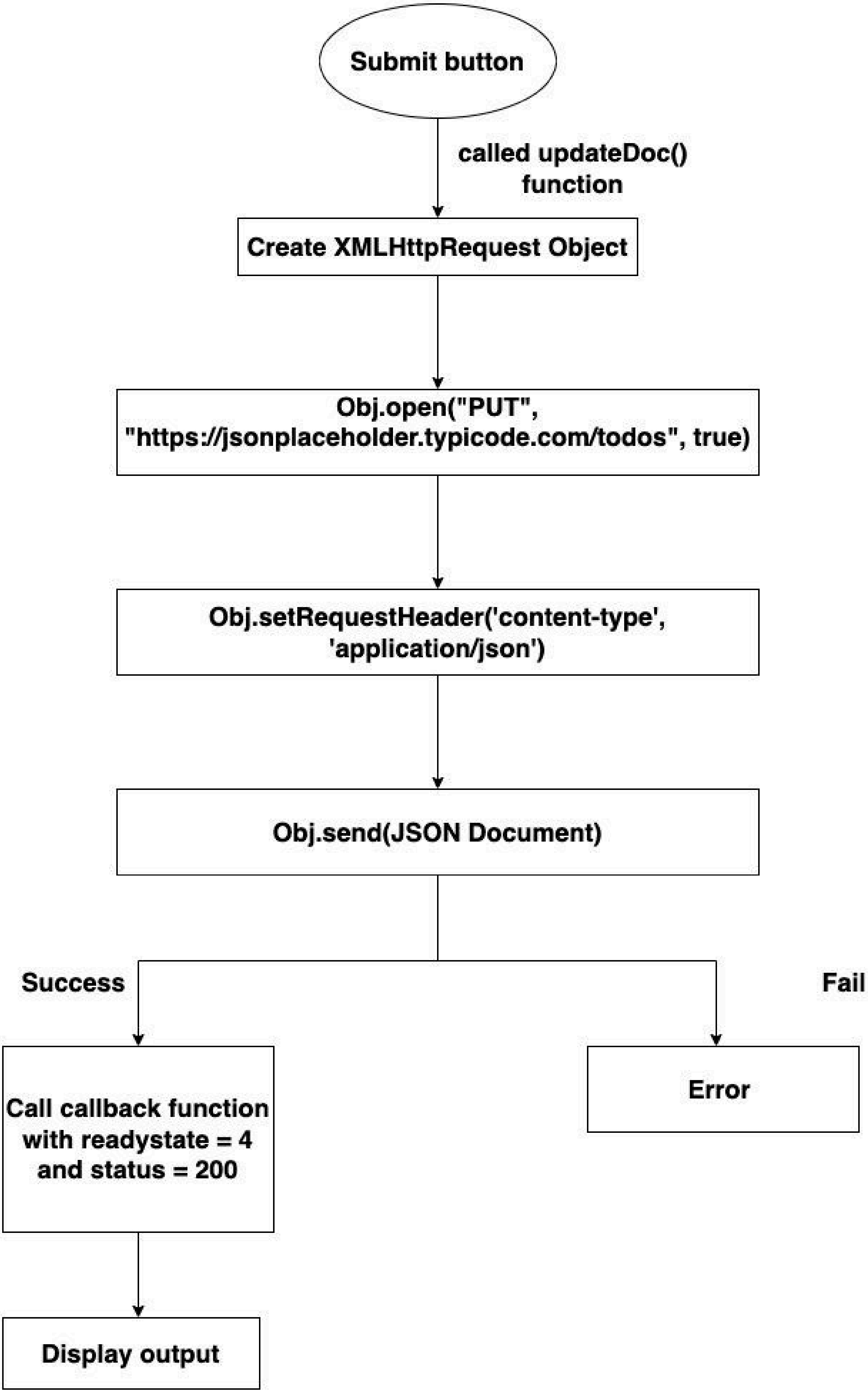
```
XMLHttpRequestObjectName.setRequestHeader('Content-type', 'application/json')
```

Step 4 − Set the HTTP header request using setRequestHeader(). It always calls after the open() method but before send() method. Here the content-type header is set to "application/json" which indicates that the data is going to be sent in JSON format.

Step 5 − At last, we convert the JSON data into the string using stringify() method and then send it to the web server using send() method to update the data present on the server.

```
XMLHttpRequestObjectName.send(JSON.stringify(JSONdata))
```

The following flow chart will show the working of the example −


Submit button
called updateDoc() function
Create XMLHttpRequest Object
Obj.open("PUT", "https://jsonplaceholder.typicode.com/todos", true)
Obj.setRequestHeader('content-type', 'application/json')
Obj.send(JSON Document)
Success
Fail
Call callback function with readystate = 4 and status = 200
Error
Display output

Example

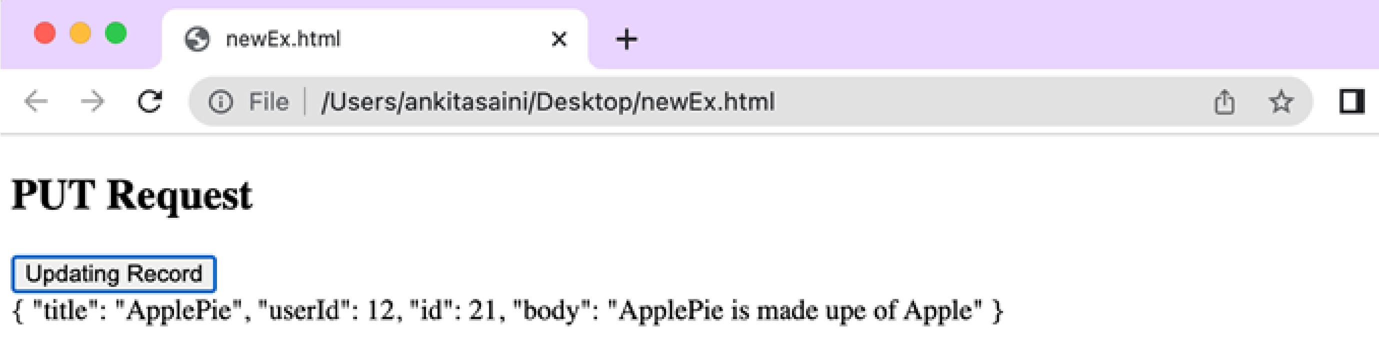
</> Open Compiler

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

      // Creating call back function
      uhttp.onreadystatechange = function() {
         if (this.readyState == 4 && this.status == 200) {
            document.getElementById("sample").innerHTML = this.responseText;
            console.log("Data update Successfully")
         }
      };
      // Updating the given file
      uhttp.open("PUT", "https://jsonplaceholder.typicode.com/todos/21", true);

      // Setting HTTP request header
      uhttp.setRequestHeader('Content-type', 'application/json')

      // Sending the JSON document to the server
      uhttp.send(JSON.stringify({
         "title": "ApplePie",
         "userId": 12,
         "id": 32,
         "body": "ApplePie is made up of Apple"
      }));
   }
</script>
<h2>PUT Request</h2>
<button type="button" onclick="updateDoc()">Updating Record</button>
<div id="sample"></div>
```

```
</body>
</html>
```

Output

The output returned by the server after clicking on the updating button.



PUT Request

Updating Record

{ "title": "ApplePie", "userId": 12, "id": 21, "body": "ApplePie is made upe of Apple" }

Here in the above code, we are updating an existing record, so for updating we create a JSON document. To update the data we click on the "Updating Record" button. So when we click on the "Submit" button, the updateDoc() function is called. This function creates an XMLHttpRequest object. Then call the open() method of the XHR object to initialise the request with the HTTP PUT method and the URL of the server which is "https://jsonplaceholder.typicode.com/todos/21". Then call the setRequestHeader() method to set the content type of the request as JSON. After that calls send() function to send the request along with the JSON document. When the server receives the request, it updates the specified record with the new data.

If the update is successful, then the callback function is called with "ready state = 4 (that means the request is completed)" and "status = 200(that means the successful response)" Then updated data will display on the screen. It also prints a message to the console representing that the data was updated successfully.

Here is the JSON.stringify() method is used to convert JSON documents into a string. It is necessary because XHR requests can only send text data.

NOTE − While you are working with the PUT method it is necessary to mention the record id in the URL like that "https://jsonplaceholder.typicode.com/todos/21". Here we are updating a record whose id is 21.

Conclusion

So this is how we can send PUT requests using XMLHttpRequest. It is generally used to update or modify the data present on the server. Now in the next article, we will learn how to send JSON data.