জাভাস্ক্রিপ্ট এজাক্স এবং এসিনক্রোনাস প্রোগ্রামিং

Created @November 19, 2024 3:19 PM

01 - XMLHttpRequest

```
 // event - onload(), onerror()
// property - response, responseText, responseType, respons
eURL, status, statusText
// function - open(), send(), setRequestHeader()

// Clear the console for better visibility of outputs
console.clear();

// Function to make an HTTP request (GET, POST, PUT, PATCH,
DELETE)
const makeRequest = (method, url, data) => {
return new Promise((resolve, reject) => {
// Create a new instance of XMLHttpRequest
let xhr = new XMLHttpRequest();

// Initialize the request by specifying the HTTP method and
URL
    xhr.open(method, url);

 // Set the request header to inform the server that the da
ta will be in JSON format
    xhr.setRequestHeader('Content-Type', 'application/jso
n');

 // Define what happens when the request is successfully co
mpleted (onload event)
    xhr.onload = () => {
```

```
        // Parse the JSON response and log it to the consol
e
        let data = xhr.response;
        console.log(JSON.parse(data));
    };

  // Define what happens if there is a network or other err
or (onerror event)
    xhr.onerror = () => {
        console.log("Error Occurs.");
    };

  // Send the request with any data passed to the function
(for POST, PUT, PATCH)
    xhr.send(JSON.stringify(data));
});
};

// Function to make a GET request to retrieve data
const getData = () => {
makeRequest('GET', 'https://jsonplaceholder.typicode.com/po
sts')
.then((res) => console.log(res));  // Log the result when t
he promise resolves
}

// Function to make a POST request to send data
const sendData = () => {
makeRequest('POST', 'https://jsonplaceholder.typicode.com/p
osts', {
title: 'foo',
body: 'bar',
userId: 1,
});
}

// Function to make a PUT request to update an entire data
record
```

```
const updateData = () => {
makeRequest('PUT', 'https://jsonplaceholder.typicode.com/po
sts/1', {
id: 1,
title: 'fooMa',
body: 'barMa',
userId: 1,
});
}

// Function to make a PATCH request to update part of a dat
a record
const updateSingleData = () => {
makeRequest('PATCH', 'https://jsonplaceholder.typicode.com/
posts/1', {
title: 'Change Name: Folk Warner',
});
}

// Function to make a DELETE request to remove a data recor
d
const deleteData = () => {
makeRequest('DELETE', 'https://jsonplaceholder.typicode.co
m/posts/1');
}

// Uncomment the function calls below to test each request
type

// getData();         // Retrieve data with GET
// sendData();        // Send new data with POST
// updateData();      // Update existing data with PUT
// updateSingleData();// Partially update data with PATCH
// deleteData();      // Delete data with DELETE
```

কোড ব্যাখ্যা বাংলায়

মন্তব্য (Comments)

```
// event - onload(), onerror()
// property - response, responseText, responseType, respons
eURL, status, statusText
// function - open(), send(), setRequestHeader()
```

- এই অংশে `XMLHttpRequest` এর বিভিন্ন ইভেন্ট, প্রপার্টি এবং ফাংশনের উল্লেখ করা হয়েছে। `onload()` এবং `onerror()` ইভেন্ট; `response`, `status` ইত্যাদি প্রপার্টি; এবং `open()`, `send()` ইত্যাদি ফাংশনের কথা বলা হয়েছে।

Console পরিষ্কার করা

```
console.clear();
```

- কনসোল ক্লিয়ার করা হয়েছে, যাতে আউটপুট আরও স্পষ্টভাবে দেখা যায়।

HTTP অনুরোধ তৈরি করার ফাংশন

```
const makeRequest = (method, url, data) => {
    return new Promise((resolve, reject) => {
        // নতুন XMLHttpRequest অবজেক্ট তৈরি
        let xhr = new XMLHttpRequest();

        // অনুরোধের ধরন (GET, POST, ইত্যাদি) এবং URL নির্ধারণ করা
        xhr.open(method, url);

        // সার্ভারকে জানানো হচ্ছে যে ডেটা JSON ফরম্যাটে পাঠানো হবে
        xhr.setRequestHeader('Content-Type', 'application/j
son');

        // অনুরোধ সফলভাবে সম্পন্ন হলে কি হবে তা নির্ধারণ করা
        xhr.onload = () => {
            // JSON ডেটা পার্স করা এবং কনসোলে দেখানো
            let data = xhr.response;
            console.log(JSON.parse(data));
        };

        // অনুরোধে কোন ত্রুটি হলে কি হবে তা নির্ধারণ করা
        xhr.onerror = () => {
```

```
            console.log("Error Occurs.");
        };

        // অনুরোধ পাঠানো (যদি ডেটা থাকে তবে তা JSON-এ রূপান্তর করে পাঠানো)
        xhr.send(JSON.stringify(data));
    });
};
```

- `makeRequest()` নামে একটি ফাংশন তৈরি করা হয়েছে, যা HTTP অনুরোধ তৈরি করে এবং একটি `Promise` রিটার্ন করে।
- `XMLHttpRequest` ব্যবহার করে একটি HTTP অনুরোধ করা হয়।
- `open()` মেথড দিয়ে HTTP মেথড ও URL সেট করা হয়।
- `setRequestHeader()` দিয়ে হেডারস ঠিক করা হয়, যাতে সার্ভার জানে যে JSON ডেটা পাঠানো হচ্ছে।
- `onload` ইভেন্টে রেসপন্স JSON ফরম্যাটে কনসোলে দেখানো হয়।
- `onerror` ইভেন্টে ত্রুটির ক্ষেত্রে একটি বার্তা কনসোলে দেখানো হয়।
- অবশেষে, `send()` মেথড দিয়ে অনুরোধ পাঠানো হয়।

GET অনুরোধের জন্য ফাংশন

```
const getData = () => {
    makeRequest('GET', '<https://jsonplaceholder.typicode.com/posts>')
        .then((res) => console.log(res));  // প্রমিস সমাধান হলে রেজাল্ট কনসোলে দেখানো হয়
}
```

- `getData()` ফাংশন `GET` মেথড ব্যবহার করে `makeRequest()` কল করে এবং ডেটা রিট্রিভ করে।

POST অনুরোধের জন্য ফাংশন

```
const sendData = () => {
    makeRequest('POST', '<https://jsonplaceholder.typicode.com/posts>', {
```

```
        title: 'foo',
        body: 'bar',
        userId: 1,
    });
}
```

- `sendData()` ফাংশন `POST` মেথড ব্যবহার করে নতুন ডেটা সার্ভারে পাঠায়। প্রোডাক্ট বা ডেটা সংক্রান্ত কিছু ফিল্ড যেমন `title`, `body`, `userId` পাঠানো হয়।

PUT অনুরোধের জন্য ফাংশন

```
const updateData = () => {
    makeRequest('PUT', '<https://jsonplaceholder.typicode.com/posts/1>', {
        id: 1,
        title: 'fooMa',
        body: 'barMa',
        userId: 1,
    });
}
```

- `updateData()` ফাংশন `PUT` মেথড ব্যবহার করে সম্পূর্ণ ডেটা আপডেট করে। `id`, `title`, `body` এবং `userId` আপডেট করা হয়।

PATCH অনুরোধের জন্য ফাংশন

```
const updateSingleData = () => {
    makeRequest('PATCH', '<https://jsonplaceholder.typicode.com/posts/1>', {
        title: 'Change Name: Folk Warner',
    });
}
```

- `updateSingleData()` ফাংশন `PATCH` মেথড ব্যবহার করে আংশিক ডেটা আপডেট করে। এখানে কেবল `title` আপডেট করা হয়।

DELETE অনুরোধের জন্য ফাংশন

```
const deleteData = () => {
    makeRequest('DELETE', '<https://jsonplaceholder.typicode.com/posts/1>');
}
```

- `deleteData()` ফাংশন `DELETE` মেথড ব্যবহার করে একটি নির্দিষ্ট ডেটা রেকর্ড মুছে ফেলে।

ফাংশন কল

```
// getData();          // GET অনুরোধ
// sendData();         // POST অনুরোধ
// updateData();       // PUT অনুরোধ
// updateSingleData();// PATCH অনুরোধ
// deleteData();       // DELETE অনুরোধ
```

- প্রত্যেকটি ফাংশন কল করার জন্য আলাদা লাইন। ব্যবহারকারীর সুবিধার্থে, চাইলে মন্তব্য থেকে আনকমেন্ট করে ব্যবহার করা যাবে।

সারসংক্ষেপ

এই কোডটি বিভিন্ন HTTP অনুরোধ (GET, POST, PUT, PATCH, DELETE) করার জন্য `XMLHttpRequest` ব্যবহার করেছে। প্রতিটি ফাংশন কনসোল আউটপুট এবং রেসপন্স পরিচালনার উপায় ব্যাখ্যা করেছে।

02 - Fetch Api

BackEndToFrontEnd

```
getProductListFromBackEnd();

async function getProductListFromBackEnd(){
```

```
    try{

      let myData = document.getElementById("myData")
      myData.innerText = "Loading..........."


      let URL = "<http://164.68.107.70:6060/api/v1/ReadProduct>"
      let res = await fetch(URL)    // await use korar karon hlo w8 korarnor jrno
      let data = await res.json()    //ager line e jdi await na dtm ta hle ai line e aisa porto tokhon pblm hoto karon amader data jdi na ase ta hle JSON ta pabo ki kore?

      myData.innerText = JSON.stringify(data)  // Amra amader Brower e JSON k dekhaite pari na tai JSON data tare amader String e convert kore nete hy tai amra (JSON.stringify) use korese JSON k String e convert korar jrno

      }catch(error){

        let myData = document.getElementById("myData")
        myData.innerText = "Something Went Wrong"
        console.log(error)

      }

    }
```

কোডের ব্যাখ্যা বাংলায়

```
getProductListFromBackEnd();
```

- `getProductListFromBackEnd()` ফাংশনটি কল করা হয়েছে যাতে সার্ভার থেকে ডেটা নিয়ে আসা যায়।

```
async function getProductListFromBackEnd() {
```

- `async` কীওয়ার্ড ব্যবহার করে একটি ফাংশন তৈরি করা হয়েছে যা অ্যাসিঙ্ক্রোনাস অপারেশন করতে পারে। এটি এমন ফাংশন, যা `await` ব্যবহার করতে পারবে।

```
try {
```

- `try` ব্লক শুরু করা হয়েছে যেখানে কোনো সম্ভাব্য ত্রুটি ছাড়া কোড নির্বাহ করার চেষ্টা করা হবে।

```
let myData = document.getElementById("myData");
```

- `myData` নামে একটি ভেরিয়েবল তৈরি করা হয়েছে যা `myData` আইডির HTML এলিমেন্টকে রেফারেন্স করে।

```
myData.innerText = "Loading.....";
```

- `myData` এলিমেন্টের `innerText` আপডেট করা হয়েছে, যাতে ব্যবহারকারীর জন্য "Loading....." মেসেজ প্রদর্শন করা হয়।

```
let URL = "<http://164.68.107.70:6060/api/v1/ReadProduct>";
```

- `URL` নামে একটি ভেরিয়েবল তৈরি করা হয়েছে যেখানে API-এর ঠিকানা রাখা হয়েছে।

```
let res = await fetch(URL);
```

- `fetch()` ফাংশন ব্যবহার করে `URL` থেকে ডেটা আনা হচ্ছে এবং `await` ব্যবহার করা হয়েছে যাতে সার্ভার থেকে ডেটা আসা পর্যন্ত এই লাইনটি অপেক্ষা করে।

```
let data = await res.json();
```

- `res` অবজেক্টে থাকা ডেটা `.json()` মেথডের মাধ্যমে JSON ফরম্যাটে কনভার্ট করা হয়েছে এবং `await` ব্যবহার করা হয়েছে যাতে JSON ডেটা আসা পর্যন্ত অপেক্ষা করা হয়।

```
myData.innerText = JSON.stringify(data)
```

- প্রাপ্ত JSON ডেটাকে `JSON.stringify(data)` মেথড দিয়ে স্ট্রিং-এ রূপান্তর করে `myData` এলিমেন্টে দেখানো হয়েছে, কারণ ব্রাউজার সরাসরি JSON অবজেক্ট প্রদর্শন করতে পারে না।

```
} catch (error) {
```

- `try` ব্লকে কোনো ত্রুটি ঘটলে `catch` ব্লক চালু হবে।

```
let myData = document.getElementById("myData");
myData.innerText = "Something Went Wrong";
```

- যদি কোনো ত্রুটি ঘটে, তাহলে `myData` এলিমেন্টের `innerText`এ "Something Went Wrong" দেখানো হবে।

```
console.log(error);
```

- ত্রুটিটি কনসোলে লগ করা হচ্ছে, যাতে ডেভেলপার সহজে ত্রুটির কারণ বুঝতে পারে।

```
}
```

- `catch` ব্লকের সমাপ্তি।

সারাংশ:

এই কোডটি সার্ভার থেকে একটি প্রোডাক্টের তালিকা নিয়ে আসে এবং তা ব্রাউজারে দেখায়। কোনো সমস্যা হলে, তা কনসোলে ত্রুটির বার্তা প্রদর্শন করে এবং ব্যবহারকারীকে জানায় যে কিছু ভুল হয়েছে।

FrontEndToBackEnd

```
<body>

<label>Product Name</label>
<input id="ProductName"/>
```

```
<br/>
<br/>
<label>Product Code</label>
<input id="ProductCode"/>
<br/>
<br/>
<label>Product Img</label>
<input id="Img"/>
<br/>
<br/>
<label>Unit Price</label>
<input id="UnitPrice"/>
<br/>
<br/>
<label>Qty</label>
<input id="Qty"/>
<br/>
<br/>
<label>Total Price</label>
<input id="TotalPrice"/>
<br/>
<br/>

<button onclick="SendDataToBackEnd()">Submit</button>

<h1 id="myMessage"></h1>


<script>

    async function SendDataToBackEnd(){

        let ProductName = document.getElementById("ProductN
ame").value;
        let ProductCode = document.getElementById("ProductC
ode").value;
        let Img = document.getElementById("Img").value;
        let UnitPrice = document.getElementById("UnitPric
```

```
e").value;
        let Qty = document.getElementById("Qty").value;
        let TotalPrice = document.getElementById("TotalPric
e").value;

        let DataForSend = {
            ProductName:ProductName,
            ProductCode:ProductCode,
            Img:Img,
            UnitPrice:UnitPrice,
            Qty:Qty,
            TotalPrice:TotalPrice
        }


        let myMessage = document.getElementById("myMessag
e");
        myMessage.innerText="Sending......"


        let URL = "<http://164.68.107.70:6060/api/v1/Create
Product>"


        let res = await fetch(
            URL,
            {
                method: "POST",
                headers:{"Content-Type":"application/jso
n"},
                body:JSON.stringify(DataForSend)
            }
        )


        let data = await res.json();

        myMessage.innerText = JSON.stringify(data)
```

```
    }

</script>
</body>
```

কোড ব্যাখ্যা বাংলায়

```
async function SendDataToBackEnd() {
```

- `async` কীওয়ার্ড সহ একটি ফাংশন তৈরি করা হয়েছে, যা অ্যাসিঙ্ক্রোনাস অপারেশন পরিচালনা করতে পারে।

```
    let ProductName = document.getElementById("ProductNam
e").value;
    let ProductCode = document.getElementById("ProductCod
e").value;
    let Img = document.getElementById("Img").value;
    let UnitPrice = document.getElementById("UnitPrice").va
lue;
    let Qty = document.getElementById("Qty").value;
    let TotalPrice = document.getElementById("TotalPrice").
value;
```

- প্রতিটি ইনপুট ফিল্ড থেকে ব্যবহারকারীর ইনপুট নেওয়া হয়েছে এবং ভেরিয়েবল হিসেবে সংরক্ষণ করা হয়েছে।

```
    let DataForSend = {
        ProductName: ProductName,
        ProductCode: ProductCode,
        Img: Img,
        UnitPrice: UnitPrice,
        Qty: Qty,
        TotalPrice: TotalPrice
    }
```

- `DataForSend` নামে একটি অবজেক্ট তৈরি করা হয়েছে যেখানে ইনপুট ডেটা গুলো JSON ফরম্যাটে রাখা হয়েছে।

```
    let myMessage = document.getElementById("myMessage");
    myMessage.innerText = "Sending......";
```

- `myMessage` এলিমেন্টের `innerText` আপডেট করে "Sending....." দেখানো হয়েছে, যা ব্যবহারকারীকে ডেটা পাঠানোর প্রসেস সম্পর্কে জানায়।

```
    let URL = "<http://164.68.107.70:6060/api/v1/CreateProduct>";
```

- সার্ভারের API এন্ডপয়েন্ট `URL` হিসেবে সংরক্ষণ করা হয়েছে যেখানে POST রিকোয়েস্ট পাঠানো হবে।

```
    let res = await fetch(
        URL,
        {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify(DataForSend)
        }
    );
```

- `fetch()` মেথড ব্যবহার করে `URL` এ POST রিকোয়েস্ট পাঠানো হয়েছে। `await` ব্যবহারের কারণে সার্ভারের রেসপন্স না আসা পর্যন্ত এই লাইনটি অপেক্ষা করবে।
- `headers` অংশে বলা হয়েছে যে ডেটা `application/json` ফরম্যাটে পাঠানো হবে।
- `body` অংশে `DataForSend` অবজেক্টকে `JSON.stringify()` দিয়ে JSON ফরম্যাটে কনভার্ট করা হয়েছে।

```
    let data = await res.json();
```

- সার্ভারের রেসপন্স JSON ফরম্যাটে কনভার্ট করে `data` ভেরিয়েবলে রাখা হয়েছে।

```
    myMessage.innerText = JSON.stringify(data);
```

- `myMessage` এলিমেন্টে সার্ভারের রেসপন্স স্ট্রিং আকারে দেখানো হয়েছে, যাতে ব্যবহারকারী রেসপন্স ডেটা দেখতে পারে।

সারসংক্ষেপ:

এই কোডটি একটি HTML ফর্ম তৈরি করে যা ব্যবহারকারীকে প্রোডাক্টের তথ্য ইনপুট করতে দেয়। `SendDataToBackEnd()` ফাংশনটি ক্লিক ইভেন্টে ডাকা হয় এবং ফর্ম ডেটা POST রিকোয়েস্টের মাধ্যমে সার্ভারে পাঠানো হয়।

03 - Axios Api

GetRequest

```
function ExcGetRequest(){
let URL = "<http://164.68.107.70:6060/api/v1/ReadProduct>";
let Configuration = {method:"GET"};

fetch(URL,Configuration)
.then(response => response.json())
.then(result => console.log(result))
.catch(error => console.log('error',error));
}
```

GET অনুরোধ ব্যাখ্যা

```
function ExcGetRequest() {
```

- `ExcGetRequest` নামে একটি ফাংশন সংজ্ঞায়িত করা হয়েছে যা GET অনুরোধ পাঠানোর জন্য।

```
let URL = "<http://164.68.107.70:6060/api/v1/ReadProduct>";
```

- `URL` নামক একটি ভেরিয়েবল তৈরি করা হয়েছে যেখানে সার্ভারের API-এর ঠিকানা রাখা হয়েছে।

```
let Configuration = { method: "GET" };
```

- `Configuration` নামে একটি অবজেক্ট তৈরি করা হয়েছে যা HTTP অনুরোধের ধরন `GET` হিসেবে নির্ধারণ করে।

```
fetch(URL, Configuration)
```

- `fetch()` ফাংশন ব্যবহার করে `URL`এ GET অনুরোধ পাঠানো হয়েছে, যেখানে `Configuration` অবজেক্টটি নির্দেশ দেয় কী ধরনের অনুরোধ পাঠানো হবে।

```
.then(response => response.json())
```

- সার্ভার থেকে পাওয়া রেসপন্স `response.json()` মেথড দিয়ে JSON ফরম্যাটে রূপান্তরিত করা হয়েছে।

```
.then(result => console.log(result))
```

- JSON ডেটা `result` হিসেবে কনসোলে লগ করা হয়েছে, যা সার্ভারের থেকে প্রাপ্ত ডেটা প্রদর্শন করে।

```
.catch(error => console.log('error', error));
```

- যদি অনুরোধ ব্যর্থ হয়, `.catch()` ব্লকে ত্রুটি কনসোলে লগ করে।

PostRequest

```
function ExecPostRequest(){
let URL = "<http://164.68.107.70:6060/api/v1/CreateProduct
>";

let BodyData = {ProductName:"Demo",ProductCode:"2113131",Im
g:"abc", UnitPrice:"12", Qty:"1", TotalPrice:"12"}
```

```
let Configuration = {
  method:"POST",
  headers: {'Accept': 'application/json','Content-Type': 'a
pplication/json'},
  body: JSON.stringify(BodyData)
}

fetch(URL,Configuration)
.then(response => response.json())
.then(result => console.log(result))
.catch(error => console.log(error));

}
```

POST অনুরোধ ব্যাখ্যা

```
function ExecPostRequest() {
```

- `ExecPostRequest` নামে একটি ফাংশন তৈরি করা হয়েছে যা POST অনুরোধ পাঠানোর জন্য।

```
let URL = "<http://164.68.107.70:6060/api/v1/CreateProduct
>";
```

- `URL` ভেরিয়েবলে POST অনুরোধের API ঠিকানা রাখা হয়েছে।

```
let BodyData = { ProductName: "Demo", ProductCode: "211313
1", Img: "abc", UnitPrice: "12", Qty: "1", TotalPrice: "12"
};
```

- `BodyData` নামে একটি অবজেক্ট তৈরি করা হয়েছে যেখানে POST অনুরোধের জন্য প্রয়োজনীয় ডেটা রয়েছে।

```
let Configuration = {
  method: "POST",
  headers: { 'Accept': 'application/json', 'Content-Type':
'application/json' },
```

```
  body: JSON.stringify(BodyData)
};
```

- `Configuration` অবজেক্টে POST অনুরোধের মেথড, হেডার এবং `body` সংজ্ঞায়িত করা হয়েছে।
  - `method: "POST"` নির্দেশ দেয় অনুরোধের ধরন।
  - `headers` অংশে কনটেন্ট টাইপ এবং গ্রহণযোগ্য ডেটা ফরম্যাট নির্ধারণ করা হয়েছে।
  - `body: JSON.stringify(BodyData)` দিয়ে `BodyData` অবজেক্টটি JSON স্ট্রিং হিসেবে রূপান্তরিত করা হয়েছে।

```
fetch(URL, Configuration)
```

- `fetch()` ফাংশন ব্যবহার করে `URL`এ POST অনুরোধ পাঠানো হয়েছে এবং `Configuration` অবজেক্ট ব্যবহার করে অনুরোধের বিবরণ নির্ধারণ করা হয়েছে।

```
.then(response => response.json())
```

- রেসপন্স JSON ফরম্যাটে রূপান্তরিত করা হয়েছে।

```
.then(result => console.log(result))
```

- সার্ভার থেকে প্রাপ্ত রেসপন্স `result` কনসোলে প্রদর্শিত হয়েছে।

```
.catch(error => console.log(error));
```

- যদি অনুরোধে কোনো সমস্যা হয়, `.catch()` ব্লকে ত্রুটি কনসোলে লগ করে।

04 - Ajax Jquery

```
// 4 common ways to make API calls in JavaScript:
// 1. XMLHttpRequest (older and lower-level API)
// 2. fetch (modern and built-in API)
// 3. axios (third-party library with promises)
// 4. jQuery (using $.ajax)

// Ensure to include the jQuery library CDN in your HTML fi
le for $.ajax to work:
// <script src="https://ajax.googleapis.com/ajax/libs/jquer
y/3.5.1/jquery.min.js"></script>

// AJAX: Asynchronous JavaScript and XML (used to send/rece
ive data from a server asynchronously)

console.clear();

// Function to make an asynchronous AJAX request using jQue
ry
const makeRequest = async (url, method, data) => {
    try {
        const result = await $.ajax({
            url: url,       // The API endpoint
            method: method, // HTTP method (GET, POST, PUT,
PATCH, DELETE)
            data: data      // Data to send with POST, PUT,
PATCH requests (optional)
        });
        return result;      // Return the response data if
the request is successful
    } catch (error) {
        console.log(error); // Log any errors if the reques
t fails
    }
}

// Function to GET data (retrieve a specific post) from the
server
const getData = () => {
```

```
    makeRequest('https://jsonplaceholder.typicode.com/posts/1', 'GET')  // GET request to fetch data
        .then((res) => {
            console.log(res);  // Log the received data
        }).catch((err) => {
            console.log(err);  // Log any errors
        });
};

// Function to POST (create new data) to the server
const createData = () => {
    makeRequest('https://jsonplaceholder.typicode.com/posts', 'POST', {  // POST request to create new data
        title: 'foo',  // Example data to create a new post
        body: 'bar',
        userId: 1,
    })
        .then((res) => {
            console.log(res);  // Log the created data from the server
        }).catch((err) => {
            console.log(err);  // Log any errors
        });
};

// Function to PUT (update existing data) on the server
const updateData = () => {
    makeRequest('https://jsonplaceholder.typicode.com/posts/1', 'PUT', {  // PUT request to update the entire post
        title: 'fooMa',  // Updated data
        body: 'barMa',
        userId: 1,
    })
        .then((res) => {
            console.log(res);  // Log the updated data from the server
        }).catch((err) => {
            console.log(err);  // Log any errors
```

```
        });
};

// Function to PATCH (partially update existing data) on th
e server
const updateSingleData = () => {
    makeRequest('https://jsonplaceholder.typicode.com/post
s/1', 'PATCH', {  // PATCH request to partially update the
post
        title: 'Jason Holder'  // Only the 'title' field is
updated
    })
        .then((res) => {
            console.log(res);  // Log the updated data
        }).catch((err) => {
            console.log(err);  // Log any errors
        });
};

// Function to DELETE data (remove a post) from the server
const deleteData = () => {
    makeRequest('https://jsonplaceholder.typicode.com/post
s/1', 'DELETE')  // DELETE request to remove the post
        .then((res) => {
            console.log(res);  // Log the response (likely
empty or confirmation message)
        }).catch((err) => {
            console.log(err);  // Log any errors
        });
};

// Uncomment the function calls below to test the API reque
sts

// getData();         // Test GET request
// createData();      // Test POST request
// updateData();      // Test PUT request
```

```
// updateSingleData();// Test PATCH request
// deleteData();      // Test DELETE request
```

এখানে প্রতিটি লাইনের ব্যাখ্যা বাংলা ভাষায় দেওয়া হল:

কনসোল পরিষ্কার করা

```
console.clear();
```

- ব্রাউজারের কনসোল পরিষ্কার করে, যাতে আগের কোনো লগ না থাকে এবং নতুন কোড রান করার সময় কনসোল পরিষ্কার থাকে।

অ্যাসিঙ্ক্রোনাস AJAX অনুরোধ করার ফাংশন

```
const makeRequest = async (url, method, data) => {
    try {
        const result = await $.ajax({
            url: url,        // API এর ঠিকানা (endpoint)
            method: method, // HTTP মেথড (যেমন: GET, POST,
PUT, PATCH, DELETE)
            data: data       // POST, PUT, PATCH অনুরোধের জন্য
ডেটা (ঐচ্ছিক)
        });
        return result;       // অনুরোধ সফল হলে রেসপন্স ডেটা
ফেরত দেয়
    } catch (error) {
        console.log(error); // অনুরোধ ব্যর্থ হলে ত্রুটি লগ করে
    }
};
```

- `makeRequest` নামের অ্যাসিঙ্ক্রোনাস ফাংশন সংজ্ঞায়িত করা হয়েছে যা jQuery এর `$.ajax()` ব্যবহার করে HTTP অনুরোধ করে।
- `url`: অনুরোধের ঠিকানা।
- `method`: HTTP অনুরোধের ধরন (যেমন: GET, POST)।
- `data`: প্রয়োজনীয় হলে ডেটা পাঠায় (শুধুমাত্র POST, PUT, PATCH এর জন্য)।
- `try...catch` ব্যবহার করে ত্রুটি হ্যান্ডলিং করা হয়েছে:

- `await $.ajax(...)`: অনুরোধ সম্পূর্ণ হওয়া পর্যন্ত অপেক্ষা করে এবং রেসপন্স ফেরত দেয়।
- যদি অনুরোধ ব্যর্থ হয়, `catch` ব্লকে ত্রুটি কনসোলে লগ করে।

GET ডেটা নেওয়ার ফাংশন (সার্ভার থেকে নির্দিষ্ট পোস্ট)

```
const getData = () => {
    makeRequest('<https://jsonplaceholder.typicode.com/posts/1>', 'GET')  // GET অনুরোধ পাঠায়
        .then((res) => {
            console.log(res);  // প্রাপ্ত ডেটা কনসোলে লগ করে
        }).catch((err) => {
            console.log(err);  // ত্রুটি থাকলে তা লগ করে
        });
};
```

- `getData` ফাংশন সংজ্ঞায়িত করা হয়েছে যা GET অনুরোধ করে নির্দিষ্ট URL-এ।
- সফল হলে `.then()` এর মাধ্যমে রেসপন্স কনসোলে প্রিন্ট করে।
- ত্রুটি হলে `.catch()` ব্লকে তা লগ করে।

POST ফাংশন (সার্ভারে নতুন ডেটা তৈরি করা)

```
const createData = () => {
    makeRequest('<https://jsonplaceholder.typicode.com/posts>', 'POST', {  // POST অনুরোধ পাঠায়
        title: 'foo',  // নতুন পোস্টের জন্য উদাহরণ ডেটা
        body: 'bar',
        userId: 1,
    })
        .then((res) => {
            console.log(res);  // সফল হলে তৈরি হওয়া ডেটা লগ করে
        }).catch((err) => {
            console.log(err);  // ত্রুটি লগ করে
        });
};
```

- `createData` ফাংশন POST অনুরোধ পাঠায় যা নতুন ডেটা তৈরি করে।
- ডেটা (যেমন `title`, `body`, `userId`) `makeRequest` ফাংশনে প্রেরণ করা হয়।
- সফল হলে রেসপন্স প্রিন্ট করে এবং ত্রুটি থাকলে তা লগ করে।

PUT ফাংশন (বিদ্যমান ডেটা সম্পূর্ণ আপডেট করা)

```
const updateData = () => {
    makeRequest('<https://jsonplaceholder.typicode.com/post
s/1>', 'PUT', {  // PUT অনুরোধ পাঠায়
        title: 'fooMa',  // আপডেট হওয়া ডেটা
        body: 'barMa',
        userId: 1,
    })
        .then((res) => {
            console.log(res);  // সফল হলে আপডেট হওয়া ডেটা ল
গ করে
        }).catch((err) => {
            console.log(err);  // ত্রুটি লগ করে
        });
};
```

- `updateData` ফাংশন PUT অনুরোধ পাঠায়, যা পুরো পোস্ট আপডেট করে।
- রেসপন্স লগ করে এবং ত্রুটি হলে তা দেখায়।

PATCH ফাংশন (আংশিক ডেটা আপডেট করা)

```
const updateSingleData = () => {
    makeRequest('<https://jsonplaceholder.typicode.com/post
s/1>', 'PATCH', {  // PATCH অনুরোধ পাঠায়
        title: 'Jason Holder'  // কেবল 'title' ফিল্ড আপডেট হ
য়
    })
        .then((res) => {
            console.log(res);  // আপডেট হওয়া ডেটা লগ করে
        }).catch((err) => {
            console.log(err);  // ত্রুটি লগ করে
```

```
        });
};
```

- `updateSingleData` ফাংশন PATCH অনুরোধ পাঠায় যা পোস্টের কেবল নির্দিষ্ট অংশ (যেমন `title`) আপডেট করে।
- সফল হলে আপডেট ডেটা লগ করে, ত্রুটি থাকলে তা দেখায়।

DELETE ফাংশন (সার্ভার থেকে পোস্ট মুছে ফেলা)

```
const deleteData = () => {
    makeRequest('<https://jsonplaceholder.typicode.com/posts/1>', 'DELETE')  // DELETE অনুরোধ পাঠায়
        .then((res) => {
            console.log(res);  // রেসপন্স (সাধারণত খালি বা নিশ্চিতকরণ বার্তা) লগ করে
        }).catch((err) => {
            console.log(err);  // ত্রুটি লগ করে
        });
};
```

- `deleteData` ফাংশন DELETE অনুরোধ পাঠায় যা নির্দিষ্ট পোস্ট মুছে ফেলে।
- সফল হলে রেসপন্স লগ করে, ত্রুটি হলে তা দেখায়।

ফাংশন কল করার জন্য মন্তব্যকৃত অংশ

```
// getData();          // GET অনুরোধ পরীক্ষা
// createData();       // POST অনুরোধ পরীক্ষা
// updateData();       // PUT অনুরোধ পরীক্ষা
// updateSingleData();// PATCH অনুরোধ পরীক্ষা
// deleteData();       // DELETE অনুরোধ পরীক্ষা
```

- পরীক্ষার জন্য ফাংশনগুলো কল করা হয়েছে। নির্দিষ্ট ফাংশন চালাতে চাইলে মন্তব্য অংশটি আনকমেন্ট করুন।