

A Hand Book Of AJAX

By -Sauraj

```
// AJAX : - asynchronous javascript and XML  
// IT IS SET OF WEB DEVELOPMENT TECHNIQUE"  
// USING MANY WEB APPLICATION ON THE CLIENT SIDE TO CREATE A  
ASYNCHRONOUS WEB APPLICATION  
// WITH AJAX WEB APPLICATION SEND AND RETRIEVE DATA FROM A SERVER  
ASYNCHRONOUS(in the background)  
  
// without interfacing the display and behaviour of the existing page  
// we don't use data in XML format anymore.  
// we use json now  
  
-----  
-----  
// we have three most common ways to create send request to the server  
-----  
  
// 1.) XMLHttpRequest (old way to do)  
// 2.) fetch API (new way to doing)  
// 3.) axios( this is the third party library)
```

40js XHR → XML HTTP Request

```

15 JS index.js ...\\ajax_U JS XHRjs U X < index.html ...\\ajax_U
js > Ajax > ajax_ > JS XHRjs > ...
1 console.log('leopoli');
2
3 const xhr = new XMLHttpRequest();
4 console.log(xhr);
5
6
7
8

```

(Object)

```

No Issues
leopoli XHR.js:1
XHR.js:4
XMLHttpRequest {onreadystatechange,
  null, readyState: 0, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, ...} 1
  onabort: null
  onerror: null
  onload: null
  onloadstart: null
  onprogress: null
  onreadystatechange: null
  ontimeout: null
  readyState: 0
  response: ""
  responseText: ""
  responseType: ""
  responseURL: ""
  responseXML: null
  status: 0
  statusText: ""
  timeout: 0
  upload: XMLHttpRequestUpload {oncancel: function}
  withCredentials: false
  [Symbol(Prototype)]: XMLHttpRequest

```

XMLHttpRequest
Object

{ our own Methods }

XHR :

→ we use XHR object to request HTTP to the Server

we use readymade APIs (servers)

<https://jsonplaceholder.typicode.com/>

Routes

All HTTP methods are supported. You can use http or https for your requests.

```

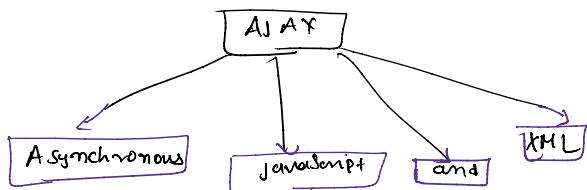
GET /posts
GET /posts/1
GET /posts/1/comments
GET /comments?postId=1
POST /posts
PUT /posts/1
PATCH /posts/1
DELETE /posts/1

```

Note: see guide for usage examples.

Resources
JSONPlaceholder comes with a set of 6 common resources:

/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users



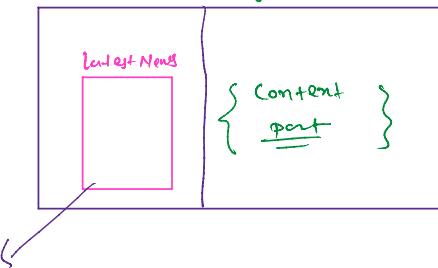
→ AJAX is a technique for creating fast and dynamic web pages.

How ?

Let's suppose we have a website having some header and one side bar



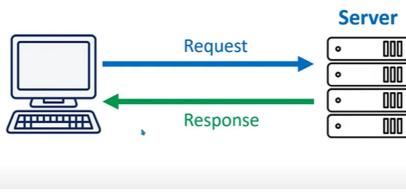
If it's a news website, whenever we click any news my head should not be changed as content part will be changed. (load)



I want to change only latest news among my content part (which is already someone reading)

In case of normal

website visitor will be request to the server for particular data (file/news/image--)



but in case of Aspx

→ [our request will not go directly to the server
we use JavaScript b/w user and server]

and this class which is belong to JavaScript
Name is

↳ XMLHttpRequest → class



[it will show page without the retraceing the page]

[we can update the Page Content without refreshing the page]

↑
we have 3 type of

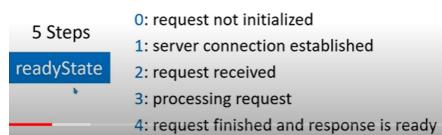
[we have 3 type of
Content

- ① Text File
- ② XML Data
- ③ JSON Data

the whole process divides into 5 steps

and we say that

readyState



When server send request

- ① status (kind of code which shows about result)
- ② expanded response XML



How Ajax work in JavaScript

- ① first we will create an object at XMLHttpRequest class

var xhttp = new XMLHttpRequest();

↳ using this object or reference

we can send or receive (request and response respectively) to the server

to do this task we have 2 Method

- ① open , ② send

- ① xhttp.open("GET", "filename.txt", true);
- ② xhttp.send();

↳ send the request to server

- ① open ("GET", any file, true/false)

Method

Synchronous
or
asynchronous

gt is about asynchronous

get id about environment
mode on/off

[If I make get false then get
could not we send my
response immediately

{ later one week done then
another will be done
{ then another

- * we check ready state always \Rightarrow if it help us to
know my response is coming from server or not

To check get we check one condition.

```
xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
        } } }  
};  
then we can perform our
```

this prototype will check
of entire server if
response status is of

\rightarrow work according to our requirement

Ex \rightarrow

```
var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
        document.getElementById("demo").innerHTML = this.responseText;  
    }  
}  
xhttp.open("GET", "filename.txt", true);  
xhttp.send();
```

checking my
ready state
every time

[get my readyState value == 4
then we can get response from server

```

< ajax > < ajax1.html > html > body > script > loadDta > onreadystatechange
</head>
<body>
<p id="demo"> Here we have to load data</p>
<button onclick="loadDta()">Click me </button>

<script>
    function loadDta() {
        // object creation
        var xhr = new XMLHttpRequest();
        // status checking
        xhr.onreadystatechange = function(){
            if(xhr.readyState== 4 && xhr.status== 200){
                var val = xhr.responseText;
                console.log(this.responseText);
                // document.getElementById("demo").innerHTML = v
            }
        }
        xhr.open("GET","./textfield.txt",true);
        xhr.send()

    }
</script>

```

Here we have to load data
Click me

DevTools - 127.0.0.1:5500/js/Ajax/ajax1.html

hii my name is sauraj
aare ladke log ghus gye hai
samam le kr pahuch

baby tu nikal
-> ham log ka momy papa jante hai
jo krna hai kr lo

Now using a fake post website URL to show their data on my website

```

< ajax > < ajax1.html > html > body > script > loadDta > onreadystatechange
    document.getElementById("demo").innerHTML = val;
}

// to handle the error
else if(this.responseText==4 &&this.status==404){
    console.log(" fail to load ");
}

// here i am just past the dummy 100 post data Link to
// show this data on my website
xhr.open("GET","https://jsonplaceholder.typicode.com/posts",true);
xhr.send()

}
<script>

```

ajax1.html:41

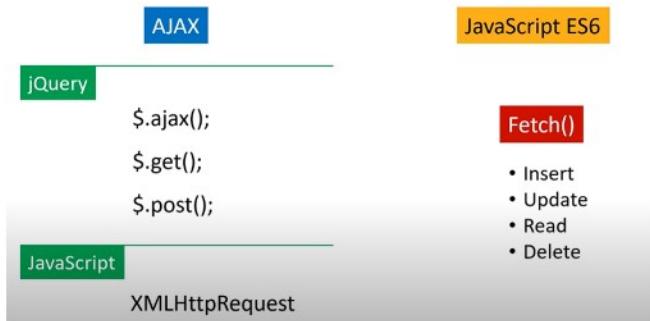
```

[
    {
        "userId": 1,
        "id": 1,
        "title": "sunt aut facere repellat provisione occaecati excepturi optio reprehenderit",
        "body": "quia et suscipit\\nuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"
    },
    {
        "userId": 1,
        "id": 2,
        "title": "qui est esse",
        "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis possimus qui neque nisi nulla"
    },
    {
        "userId": 1,
        "id": 3
    }
]

```

* Fetch Method () ES6v

what is Fetch () Method ?



fetch Method return a promise

```

graph TD
    fetch["fetch(file / URL)"]
    fetch -- "resolved" --> resolved["resolved"]
    fetch -- "reject" --> reject["reject"]
  
```

So we can use then → function

If fetch return successful then we can use then

then(callback fn) → It will take any callback

Now see How fetch works in my case

fetch(file / URL) → Promise

fetch(file / URL).then() → return promise

fetch(file / URL).then(function(response){})

return response.data;
↳ from server
any variable name
↳ JSON

go inside
2nd then function

```

fetch(file / URL).then(function(response){
    return response.data;
}).then(function(result){
    console.log(result);
});
  
```

↳ then Method take a function as a argument
(for more details see promise)

```

        console.log(result);
    });
}

a function as a argument
( for more details see promise
note)

```

If my promise fail then we use catch to stop error message

```
.catch(function(error){
    console.log(error);
```

```
fetch(file / URL).then(function(response){
    return response.data;
}).then(function(result){
    console.log(result);
}).catch(function(error){
    console.log(error);
});
```

fetch method work on live Server

example (working with simple text file)

The screenshot shows a browser's developer tools console with the URL `127.0.0.1:5500/js/Ajax/fetch_method/`. The output in the console is:

```
hi my name is sauraj
aare ladke log ghus gye hai
saman le kr pahuch
-----
baby tu nikal
-> ham log ka momy papa jante hai
jo krna hai kr lo
```

To the right of the code, handwritten notes explain the process:

- A bracket points from the `response` variable in the code to the output in the console, with the text "when promise full fill it resolved and it return".
- A bracket points from the `response` variable to the word "data" in the output, with the text "response variable hold it data is returning".
- A bracket points from the `text()` method in the code to the output, with the text "text format".
- A bracket points from the `val` variable in the code to the output, with the text "we are using then val very easy".

* Now working with fake json data

<https://jsonplaceholder.typicode.com/users>

→ to use fake data we are

using now

```
<script>
// 10 fake data we are using here
// it will return json file
fetch("https://jsonplaceholder.typicode.com/users").then((respon
se)=>{
    return response.text();
}).then((val)=>{
    // console.log(val);
});
```

```

    // // document.write(val);
    for(var key in val){
        document.write(`${val[key].name}-${val[key].email}<br>`);
    }
}).catch((err)=>{
    console.log(err +" ->can't fetch data");
})
</script>

```

```

<!-- ----- -->
<!-- how we will save data in server -->
<!-- i want to insert data , update data and delete the data -->
<!-- ----- -->

```

So ~~to~~ with the help of `fetch()`
Fetch() – Insert, Update, Delete

`fetch(file / URL);`

↗ Server side language file
(PHP, Python ... --)

Syntax

`fetch (file / URL, { })`

↗ parameters
[Ex → some Properties]

`fetch (file / URL, { }`

① method : "post", → for insert the data
 Put → for update the data
 Delete → to delete the data

② body : data,
 ↗ this data is use to store in server
 ↗ it will be either
 form data or JSON Data or Text

③ header : {
 ↗ in side header we pass data format which
 ↗ will pass through body (Content-Type)
 'Content-Type': 'application / json',
 ↗ To handle all requests.

'Content-Type': 'application/json',
 },
 ↓
 data is in json form which i passed
 through body
 ↗ basically gt is showing
 If gt is not a json data then we have
 one more option (form data)

Syntax for form data
 ↗ we have a method
 in JavaScript name →
 [form data]

Content-Type: 'application/x-www-form-urlencoded'

```

fetch(file / URL, {
  method: "POST", → "PUT" "DELETE" "GET"
  body: data, → Form Data / JSON Data / Text
  header: {
    'Content-Type': 'application/json',
  },
});
  ↓
'Content-Type': 'application/x-www-form-urlencoded'
  
```

Let's code

I want to store my data into database
 we will use fake json data

go → json placeholder

Resources

JSONPlaceholder comes with a set of 6 common resources

/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users

Note: resources have relations. For example: posts have many photos, ... see guide for the full list.
<https://jsonplaceholder.typicode.com/guide/>

Note: resources have relations. For example: posts have many photos, ... see [guide](#) for the full list. <https://jsonplaceholder.typicode.com/guide/>

```

<script>
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  body: JSON.stringify({
    title: 'foo',
    body: 'bar',
    userId: 1
  }),
  headers: {
    'Content-type': 'application/json; charset=UTF-8'
  }
})
.then((response) => response.json())
.then((json) => console.log(json));
</script>
</body>
</html>

```

we can't send object directly
using `JSON.stringify` we can't object to JSON format
basically it's a object of JavaScript

→ we can write it in diff ways

```

<script>
var obj = {
  title: 'foo',
  body: 'bar',
  userId: 1,
}
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  body: JSON.stringify(obj),
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  },
})

```

→ first we create object
then pass here to convert it into JSON format

To update

```

<script>
//-----
// Updating a resource
//-----
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'PUT',
  body: JSON.stringify({
    id: 1,
    title: 'foo',
    body: 'bar',
    userId: 1,
  }),
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  },
})
.then((response) => response.json())
.then((json) => console.log(json));

```

when we have to update we have to path

→ Through form I am storing the data in database

```
.then((json) => console.log(json));
```

→ Through form I am storing the data in database

The screenshot shows a browser developer tools console with the following details:

- Network Tab:** Shows a POST request to `https://jsonplaceholder.typicode.com/posts` with the following headers:
 - Content-Type: application/json; charset=UTF-8
 - Content-Length: 143
 - Method: POST
- Request Body:** A JSON object:

```
{title: 'mrinal', body: 'nsbhf', userId: 1234}
```
- Response Body:** A JSON object:

```
{id: 101, title: "mrinal", body: "nsbhf", userId: 1234}
```
- Console Tab:** Shows the following log statement:

```
.then((json) => console.log(json));
```
- Script Tab:** Shows the JavaScript code for the `index.html` file, which includes a `fetch` call to the same endpoint.