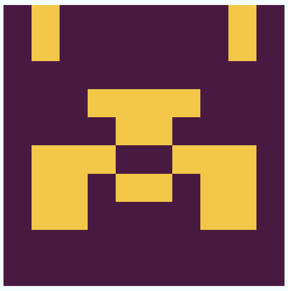
Requests



+3
Published Aug 30, 2021•**Updated Oct 13, 2023**

Contribute to Docs

`XMLHttpRequest` is a built-in browser object that allows HTTP requests to be made in JavaScript. It is used to fetch data from API’s.

Note: Use of synchronous `XMLHttpRequest` is not recommended. It is recommended to use the Fetch API instead, which includes the asynchronous `fetch()` method.

Getting Started

`XMLHttpRequest` can be used in two modes: synchronous and asynchronous.

1. Asynchronous Method

There are 4 steps to do this:

1. Create `XMLHttpRequest`:

   ```
   let xhr = new XMLHttpRequest();
   ```

   The constructor has no arguments.

2. Initialize it:

   ```
   xhr.open(method, URL, [async, user, password]);
   ```

   This method specifies the main parameters of the request:

   - `method`: HTTP-method. Usually `"GET"` or `"POST"`.
   - `URL`: The URL to request, a string, can be URL object.
   - `async`: If explicitly set to `false`, then the request is synchronous.
   - `user`, `password`: login and password for basic HTTP auth (if required).

   Please note that `open` call, contrary to its name, does not open the connection. It only configures the request, but the network activity only starts with the call of `send`.

3. Send it out.

   ```
   xhr.send([body]);
   ```

   This method opens up the connection and sends the request to server. The optional body parameter contains the request `body`.

   Some request methods like `GET` do not have a `body`. And some of them like `POST` use `body` to send the data to the server.

4. Listen to xhr events for response.

These three events are the most widely used:

- `load`: When the request is complete and the response is fully downloaded.
- `error`: When the request couldn’t be made successfully, For example, network down or invalid URL.
- `progress`: Triggers periodically while the response is being downloaded, reports how much has been downloaded.

```
xhr.onload = function () {
 alert(`Progress: ${xhr.status} ${xhr.response}`);
};

xhr.onerror = function () {
 alert(`Network Error Occurred`);
};

xhr.onprogress = function (event) {
 // Triggers periodically
 // event.loaded - how many bytes downloaded
 // event.lengthComputable = true if the server sent Content-Length header
```

```
  // event.total - total number of bytes (if lengthComputable)
  alert(`Received ${event.loaded} of ${event.total}`);
};
```

Once the server has successfully responded, we can receive the result in the following xhr properties:

status

HTTP status code (a number): 200, 404, 403 and so on, can be 0 in case of a non-HTTP failure.

statusText

HTTP status message (a string): usually OK for 200, Not Found for 404, Forbidden for 403 and so on.

response (old scripts may use responseText)

The server response body.

We can also specify a timeout using the corresponding property:

```
xhr.timeout = 10000; // Timeout in ms, 10 seconds
```

If the request does not succeed within the given time, it gets canceled and timeout event triggers.

2. Synchronous Method

If in the .open method the third parameter async is set to false, the request is made synchronously.

It might look good, but synchronous calls are used rarely, because they block in-page JavaScript till the loading is complete. In some browsers it becomes impossible to scroll. If a synchronous call takes too much time, the browser may suggest to close the “hanging” webpage.

Response Type

xhr.responseType property can be used to set the response format:

- "" (default): get as string,

- "text": get as string,
- "arraybuffer": get as ArrayBuffer
- "blob": get as Blob,
- "document": get as XML document (can use XPath and other XML methods) or HTML document (based on the MIME type of the received data),
- "json": get as JSON (parsed automatically).