NANA TAHIR HAMMOND

INDEX NO : 050919001

COURSE : WEB API

LECTURER : AUGUTUS BUCKMAN

Content of ajax request

1. XHR

XMLHttpRequest is an object such as (a native component in most other browsers, an ActiveX object in Microsoft Internet Explorer) that permits a web page to make a request to a server and get a response back without reloading the entire page. The user continues on the same page that is page did not reload, and more importantly, they will not actually see or notice the processing occur — that is, they will not see a new page loading, not by default at least.

Using the XMLHttpRequest object makes it possible for a developer to change a page earlier loaded in the browser with data from the server without having to request the full page from the server again.

Performing GET request using XHR

```
const Http = new XMLHttpRequest();

const url='http://yourdomain.com/';

Http.open("GET", url);

Http.send();

Http.onreadystatechange=(e)=>{

console.log(Http.responseText)

}
```

Performing Post request using XHR

```
var xhr = new XMLHttpRequest();

xhr.open("POST", '/submit', true);

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xhr.onreadystatechange = function() {

  if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
        // Request finished. Do processing here.

    }

}

xhr.send("name=Ketan&id=1");
```

2. Fetch API

Fetch API is the new choice to XMLHttpRequest for getting resources from the server. Unlike XMLHttpRequest, it has a more powerful feature set and a more meaningful name. Fetch is also flexible and easy to use because of its syntax and structure. However, the thing that sets it apart from other AJAX HTTP libraries is that it is supported by all modern web browsers. Fetch follows a request-response approach where Fetch makes a request and returns a promise that resolves to the Response object.

Pros of using Fetch API

It’s flexible and easy to use

Callback hell is avoided by Promises

Supported by all modern browsers

Follows request-response approach

Cons of using Fetch API

Doesn’t send cookie by default

CORS is disabled by default

Performing GET Request in Fetch API

```
fetch('https://www.yourdomain.com', {

        method: 'get'

    })

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

    .then(jsonData => console.log(jsonData))

    .catch(err => {

            //error block
```

```
      }
```

Performing POST Request in Fetch API

```
var url = 'https://www.yourdomain.com/updateProfile';

var data = {username: 'courseya'};

fetch(url, {

  method: 'POST', // or 'PUT'

  body: JSON.stringify(data), // data can be `string` or {object}!

  headers:{

    'Content-Type': 'application/json'

  }

}).then(res => res.json())

.then(response => console.log('Success:', JSON.stringify(response)))

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

3. jQuery

jQuery is a client-side programming language you can be used to create cool and amazing web applications. It's free, yet powerful, comparatively easy to set up and learn, and it has multiple extensions and plugins available to do anything you could imagine or think off. You can get started quickly, and you won't outgrow it later when you get really good at it.

Pros of using Jquery

The greatest advantage of jQuery is its simplicity.

It is also incredibly flexible because jQuery allows users to add plug-ins.

It is also a very fast solution to your problems. While there may be "better" solutions, jQuery and its development teamwork to make sure you can implement jQuery quickly and effectively, which saves money.

Open-source software means quick growth and the freedom of developers to provide the best service possible without corporate red tape.

Cons of using Jquery

It is also, frequent updates mean community members are also unlikely to provide solutions.

There are also multiple varieties of jQuery available currently and some are less compatible than others.

Sometimes jQuery is slower compared to CSS in some cases. At that time its simplicity becomes a curse, as it is not meant for client-side interactions.

Jquery Interview Questions

Performing GET Request in Jquery

```
$.ajax({
  url: '/users',
  type: "GET",
  dataType: "json",
  success: function (data) {
    console.log(data);
  },
  error: function (error) {
    console.log(`Error ${error}`);
  }
});
```

Performing POST Request in Jquery

```
$.ajax({
  url: '/users',
  type: "POST",
  data: {
    name: "Ipseeta",
    id: 1
  },
  dataType: "json",
  success: function (data) {
    console.log(data);
  },
  error: function (error) {
    console.log(`Error ${error}`);
  }
});
```

4. Axios

Axios is one among many available promise-based HTTP client that works both in the browser and in a node.js environment. It basically provides a single API for dealing with XMLHttpRequest s and node's HTTP interface. Apart from that, it binds the requests using a polyfill for ES6 new's promise syntax.

Advantages of using Axios

Out-of-the-box Promise support

Client-side support for protecting against XSRF

Can capture requests or responses before they are carried out.

Automatic transforms for JSON data

Supports Promise API

Can set or cancel a request

Can set a response timeout

It works on both Nodejs and Browser

Performing GET Request in Axios

```
axios.get('/get-user', {
  params: {
   ID: 1
```

```
    }
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  })
  .then(function () {
    // always executed
  });
```

Performing POST Request in Axios

```
axios.post('/user', {
    name: 'Sanjeev',
    id: 1
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

5. Request

The Request library is one of the simplest ways to make HTTP calls. The structure and syntax are very similar to that of how requests are handled in Node.js. Currently, the project has 18K stars on GitHub and deserves a mention for being one of the popular HTTP libraries available.

Syntax

```
var request = require('request');

request('http://www.yourdomain.com', function (error, response, body) {

  console.log('error:', error);

  console.log('statusCode:', response && response.statusCode);

  console.log('body:', body);

});
```

6. SuperAgent

SuperAgent is a lightweight and progressive AJAX library that's focused more on readability and flexibility. SuperAgent also boasts of a gentle learning curve unlike other libraries out there. SuperAgent has a request object that accepts methods such as GET, POST, PUT, DELETE, and HEAD.

Pros of SuperAgent

It has a plugin-based environment and ecosystem where plugins could be built and developed for extra or additional functionality.

Easily Configurable.

Nice interface for making HTTP requests.

Multiple functions chaining to send requests.

Has to support for upload and download progress.

Has support for chunked transfer encoding.

Old-style callbacks are supported

Numerous plugins available for many common features

Performing Get Request

```
request

   .get('/user')
```

```
  .query({ id: 1 })

  .then(res => {

  });
```

Performing Post Request

```
request.post('/user')

   .set('Content-Type', 'application/json')

   .send('{"name":"Ipseeta","id":1}')

   .then(callback)

   .catch(errorCallback)
```