ლაბორატორიული 9

1. **Ajax, XMLHttpRequest**

Ajax - ტექნოლოგია რომელიც საშუალებას გვაძლევს მივმართოთ სერვერს(გავაგზავნოთ request) და მივიღოთ მონაცემები გვერდის გადატვირთვის გარეშე, javascript-ის გამოყენებით.

XMLHttpRequest არის ძირითადი ობიექტი რომლის მეშვეობითაც ხდება სერვერზე ajax request-ების გაგზავნა და მონაცემების მიღება.

ობიექტის შექმნა:

var xhr = new XMLHttpRequest();

კონფიგურირებისთვის გამოიყენება მეთოდი open, რომელიც ღებულობს შემდეგ პარამეტრებს:

xhr.open(method, URL, async, user, password)

method – HTTP-მეთოდი, ყველაზე ხშირად GET ან POST, ასევე შესაძლებელია PUT ან DELETE.

URL - მისამართი სერვერზე, სადაც იგზავნება request-ი. აქ მოქმედებს შეზღუდვა Same Origin Policy, რაც გულისხმობს რომ request-ის გაგზავნა მხოლოდ იგივე დომეინზე იგივე პროტოკოლით და იგივე პორტის ნომერზე შეიძლება რომლიდანაც ჩაიტვირთა საიტი თავდაპირველად. შესაბამისად გვერდიდან მისამართით <https://mywebsite.com/home> request-ების გაგზaვნა შეიძლება ნებისმიერ მისამართზე რომელიც იწყება https://mywebsite.com-ით, მაგალითად <https://mywebsite.com/data>, <https://mywebsite.com/students/list> და ა. შ.

async - უნდა მოხდეს თუ არა მოთხოვნის შესრულება ასინქრონულად, თუ გადაეცემა false მოთხოვნა სრულდება სინქრონულად, ანუ მისი გაგზავნისას სკრიპტის შესრულება ჩერდება სერვერიდან პასუხის მოსვლამდე. Default-ად არის true, მეტიც სინქრონულად მოთხოვნის შესრულება არ არის რეკომენდირებული და არის deprecated.

user, password - გამოიყენება როდესაც სერვერის მხარეს არის HTTP ავტორიზაცია

open-ის გამოძახება მხოლოდ აკონფიგურირებს XMLHttpRequest-ის ობიექტს, ამ დროს არ ხდება უშუალოდ request-ის გაგზავნა. მოთხოვნის შესასრულებლად და მონაცემების უშუალოდ გაგზავნისთვის გამოიყენება send მეთოდი:

xhr.send([body])

body - არის მოთხოვნის ტანი, GET მოთხოვნის შემთხვევაში საერთოდ არ გადაეცემა, ხოლო POST-ის შემთხვევაში სწორედ body-ის გამოყენებით გადაეცემა მონაცემები სერვერს.

მოთხოვნის შესრულების დასრულებამდე გაუქმება შესაძლებელია შემდეგნაირად:

xhr.abort()

მოთხოვნის შესრულების შედეგად ხელმისაწვდომია შემდეგი ინფორმაცია:

* სტატუსი (status) რომელიც წარმოადგენს http status code-ს, მაგალითად 200, 404, 403 და ა. შ.
* statusText - სტატუსის ტექსტური აღწერა, მაგალითად OK, Not Found, Forbidden და ა. შ.
* responseText - უშუალოდ სერვერის მიერ დაბრუნებული მონაცემები (პასუხი)

როდესაც მოთხოვნა სრულდება ასინქრონულად, სერვერიდან პასუხის მოსვლის მომენტის გასაგებად არსებობს event-ი readystatechange, რომელიც რამდენჯერმე ხდება მოთხოვნის შესრულების სხვადასხვა ეტაპზე და მიმდინარე მდგომარეობის გაგება შესაძლებელია xhr.readyState თვისების მნიშვნელობის შემოწმებით. მისი შესაძლო მნიშვნელობებია:

UNSENT = 0; // საწყისი მდგომარეობა

OPENED = 1; // open გამოძახების შემდეგ

HEADERS\_RECEIVED = 2; // header-ების მიღების შემდეგ

LOADING = 3; // პასუხის ჩატვირთვისას მის დასრულებამდე (მონაცემთა მორიგი პაკეტის მიღებისას)

DONE = 4; // დასრულების შემდეგ

ქვემოთ ნაჩვენებია xhr-ის გამოყენებით სერვერიდან მონაცემების ასინქრონულად ჩატვირთვის მაგალითი (იგულისხმება რომ სერვერზე არის სტატიკური ტექსტური ფაილი students.json სტუდენტების შესახებ ინფორმაციით):

var xhr = new XMLHttpRequest();

xhr.open('GET', 'students.json', true);

xhr.send();

xhr.onreadystatechange = function() {

if (xhr.readyState != 4) return;

if (xhr.status != 200) {

// შეცდომის დამუშავება

alert(xhr.status + ': ' + xhr.statusText);

} else {

// შედეგის გამოტანა

alert(xhr.responseText);

}

}

მოთხოვნის შესრულებისას შესაძლებელია header-ების გაყოლება xhr ობიექტის შემდეგი მეთოდის გამოყენებით:

**setRequestHeader(name, value)**

მაგალითად:

xhr.setRequestHeader('Content-Type', 'application/json');

ასევე შესაძლებელია სერვერიდან პასუხთან ერთად დაბრუნებული header-ების წაკითხვა, მაგალითად:

xhr.getResponseHeader('Content-Type')

მოთხოვნის შესრულების მაქსიმალური დროის დაყენება შესაძლებელია შემდეგნაირად:

xhr.timeout = 30000; // 30 წამი

ამ დროის გასვლის შემდეგ თუ სერვერმა პასუხი არ დააბრუნა მოხდება timeout event-ი. მისი გამოყენება შემდეგნაირად შეიძლება:

xhr.ontimeout = function() {

alert( 'ლოდინის დრო ამოიწურა' );

}

1. **XMLHttpRequest-ის ალტერნატივა fetch**

XMLHttpRequest-ის უფრო თანამედროვე ალტერნატივა არის მეთოდი fetch:

let promise = fetch(url[, options]);

* url - მისამართი რომელზეც უნდა მოხდეს მოთხოვნის გაგზავნა
* options - არააუცილებელი კონფიგურაციის ობიექტი, რომლის გამოყენებითაც შესაძლებელია request-ის პარამეტრების განსაზღვრა

options ობიექტის თვისებები:

* method – HTTP მეთოდი
* headers – მოთხოვნის header-ები,
* body – მოთხოვნის ტანი FormData, Blob, სტრიქონი და ა. შ.
* mode – ერთ ერთი ჩამოთვლილთაგან «same-origin», «no-cors», «cors»
* credentials – ერთ ერთი ჩამოთვლილთაგან: «omit», «same-origin», «include». განსაზღვრავს უნდა გაყვეს თუ არა მოთხოვნას cookie და ავტორიზაციის header-ები
* cache - ერთ ერთი ჩამოთვლილთაგან: «default», «no-store», «reload», «no-cache», «force-cache», «only-if-cached».
* redirect – შესაძლებელია «follow» მითითება 30x (გადამისამართება) შემთხევვაშ რო გაყვეს ან «error» რათა გადამისამართება აღიქვას როგორც შეცდომა.

fetch მეთოდი აბრუნებს promise-ს რომელიც იძახებს callback-ებს სერვერიდან მოსული პასუხის შესაბამისი მონაცმების ობიექტით Response ან შედომის ობიექტით თუ მოთხოვნის შესრულებისას მოხდა შეცდომა.

fetch-ის გამოყენების მაგალითი:

fetch('/data/students.json')

.then(function(response) {

alert(response.headers.get('Content-Type')); // application/json; charset=utf-8

alert(response.status); // 200

return response.json();

})

.then(function(students) {

alert(students.length); // 10

})

.catch( alert );

Response ობიექტს გააჩნია მეთოდები მონაცემების შესაბამის ფორმატში ამოღებისთვის:

* response.arrayBuffer()
* response.blob()
* response.formData()
* response.json()
* response.text()

თითოეული მათგანი აბრუნებს promise-ს, რომლის callback-შიც შესაძლებელია მონაცემების სასურველ ფორმატში მიღება და შემდგომი დამუშავება (მაგალითად გამოტანა).

1. **ბიბლიოთეკა axios**

http api-სთან მუშაობის ერთერთ მოსახერხებელ და ეფექტურ საშუალებას წარმოადგენს ბიბლიოთეკა axios. დაყენება:

npm install axios

გამოყენების მაგალითები:

GET:

const axios = require('axios');

// Make a request for a user with a given ID

axios.get('/user?ID=12345')

.then(function (response) {

// handle success

console.log(response);

})

.catch(function (error) {

// handle error

console.log(error);

})

.then(function () {

// always executed

});

// Optionally the request above could also be done as

axios.get('/user', {

params: {

ID: 12345

}

})

.then(function (response) {

console.log(response);

})

.catch(function (error) {

console.log(error);

})

.then(function () {

// always executed

});

// Want to use async/await? Add the `async` keyword to your outer function/method.

async function getUser() {

try {

const response = await axios.get('/user?ID=12345');

console.log(response);

} catch (error) {

console.error(error);

}

}

POST:

axios.post('/user', {

firstName: 'Fred',

lastName: 'Flintstone'

})

.then(function (response) {

console.log(response);

})

.catch(function (error) {

console.log(error);

});

რამდენიმე request-ის ერთდროული გაგზავნა:

function getUserAccount() {

return axios.get('/user/12345');

}

function getUserPermissions() {

return axios.get('/user/12345/permissions');

}

axios.all([getUserAccount(), getUserPermissions()])

.then(axios.spread(function (acct, perms) {

// Both requests are now complete

}));

დავალება:

* 1. L0901 დაწერეთ სკრიპტი რომელიც გამოიტანს ცხრილის სახით პოსტების სიას. მონაცემები უნდა ჩაიტვრითოს ველში(input) სასურველი მომხმარებლის id-ს ჩაწერის და ღილაკზე დაჭერის შემდეგ მისამართიდან [https://jsonplaceholder.typicode.com/posts?userId=](https://jsonplaceholder.typicode.com/posts?userId=1){სასურველი პოსტის id}  
     ცხრილში უნდა გამოტანილი იყოს შემდეგი სვეტები title, პოსტის ტექსტი, და სვეტი ღილაკით კომენტარების დათვალიერება
  2. L0902 დაწერეთ სკრიპტი რომელიც წინა დავალებაში გაკეთებულ ცხრილში კომენტარების ნახვა ღილაკზე დაკლიკვით (კონკრეტულ პოსტის გასწვრივ) გამოიტანს მეორე ცხრილში ამ პოსტზე არსებულ კომენტარებს. მიმდინარე მონიშნული პოსტის ფონი ამ დროს უნდა შიცვალოს (გახდეს მწვანე) რათა ჩანდეს რომელი პოსტი არის მონიშნული. მონაცემები უნდა ჩაიტვირთოს შემდეგი მისამართიდან: <https://jsonplaceholder.typicode.com/comments?postId>={სასურველი პოსტის id}

**შენიშვნა:**

API-ს დოკუმენტაცია იხილეთ მისამართზე <https://jsonplaceholder.typicode.com>

Request-ების გასაკეთებლად გამოიყენეთ fetch და axios (ერთი fetch-ით, მეორე axios)