Взаємодія з мережею. **XMLHttpRequest.** ***fetch***

https://uk.javascript.info/network

https://dev.to/ruppysuppy/7-free-public-apis-you-will-love-as-a-developer-166p

https://jsonplaceholder.typicode.com/

https://restcountries.com/#endpoints-full-name

https://github.com/public-apis/public-apis#animals

XMLHttpRequest

XMLHttpRequest

- вбудований об’єкт браузера, який дозволяє виконувати HTTP-запити за допомогою JavaScript
- може працювати з будь-якими даними (текст, `JSON, двійкові дані,` …)
- дозволяє завантажувати дані з сервера
- дозволяє завантажувати дані на сервер
- відслідковувати статус запиту
- відміняти запит і т.д.

Процес взаємодії з сервером складається з таких кроків

- Створення об’єкта XMLHttpRequest
- Ініціалізація
- Надсилання запиту
- Прослуховуємо відповідь

Кроки	Загальна форма	Приклад
Створення об'єкта ***XMLHttpRequest***	let xhr = new XMLHttpRequest()	`let xhr = new XMLHttpRequest()`
Ініціалізація	xhr.open(method, // GET, POST, PUT,DELETE URL, [async, //false – синхронний запит user, password] // для аутентифікації)	`xhr.open(` `'GET',` `'https://dog.ceo/api/breeds/image/random'` `)`
Надсилання запиту	xhr.send([body]) //тіло запиту (якісь дані, що надсилаємо)	`xhr.send()`
Прослуховуємо відповідь	(аналізуємо події)	`xhr.onload = function () {` `  if (xhr.status == 200) {` `  document.body.innerHTML = ` `` ` `` `  <img src="${JSON.parse(xhr.response).message}"> ` `` ` `` `  }` `}`
load – спрацьовує коли запит завершено і відповідь сервера повністю завантажено	xhr.onload = function() { . . . . . } xhr.status // 200, 204, 401, 404, 403, 500 xhr.statusText // OK-200, Not Found - 404, Forbidden – 403, … xhr.response //тіло відповіді	
error – спрацьовує коли запит не може бути виконано (проблеми з мережею, некоректний URL)	xhr.onerror = function() { . . . . }	xhr.onerror = function() { alert("Не вдалося виконати запит"); };
progress – періодично спрацьовує під час завантаження відповіді, та повідомляє, скільки байтів було завантажено.	xhr.onprogress = function(event) { . . . . . }	xhr.onprogress = function(event) { if (event.lengthComputable) { alert(`Отримано ${event.loaded} із ${event.total} байт`); } else { alert(`Отримано ${event.loaded} байт`); } }

Fetch API – спрощений інструмент для взаємодії з мережевими ресурсами

fetch – метод для організації запитів до серверів

Кроки		З використанням ***await***	Без використання ***await***
Виклик функції	fetch(url, [options]) **url** – адреса **options** – додаткові параметри	try { let ***response*** = await **fetch**(url)	**fetch**(url)
очікуємо і аналізуємо відповідь (якщо помилки не сталось) відповідь у об'єкті ***response***		. . . . аналіз ***response*** . . . . }	. then(***response*** => { . . . . аналіз ***response*** . . . . })
обробка виключних ситуацій (якщо сталася помилка)		catch(err){ . . . . . . }	.catch(err => { . . . . . . })
завершальні операції (виконуємо у будь-якому випадку)		finally { . . . . . . }	.finally(( ) =>{ . . . . . . })

З використанням ***XMLHttpRequest***	```let xhr = new XMLHttpRequest() xhr.open('GET', 'https://dog.ceo/api/breeds/image/random') xhr.send() xhr.onload = function () { if (xhr.status == 200) { document.body.innerHTML = ` <img src="${JSON.parse(xhr.response).message}"> ` } }```
З використанням fetch	```fetch('https://dog.ceo/api/breeds/image/random') .then((response) => response.json()) .then((data) => { document.body.innerHTML = `<img src="${data.message}">` })```
З використанням fetch	```async function getImage() { let response = await fetch('https://dog.ceo/api/breeds/image/random') if (response.ok) { const data = await response.json() document.body.innerHTML = `<img src="${data.message}">` } } getImage()```

Властивості/методи об’єкта відповіді ***response***

Властивість	Опис
response.ok	`true` якщо код відповіді у діапазоні 200-299, тобто запит виконано успішно
response.status	код відповіді https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D1%96%D0%B2_%D1%81%D1%82%D0%B0%D0%BD%D1%83_HTTP
response.type	тип відповід
response.statusText	текстове повідомлення, що відповідає поточному статусу
response.headers	заголовки відповіді (об’єкт Headers) містить додаткові параметри
response.body	є об’єктом ReadableStream (містить дані, що передані з сервера)
response.url	адреса

https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D1%96%D0%B2_%D1%81%D1%82%D0%B0%D0%BD%D1%83_HTTP

Властивості/методи об’єкта відповіді ***response***

Метод	Опис	Приклад
response.text()	дозволяє отримати з response пересланий текст	let response = await fetch(url); let textData = await ***response.text***() ----- або ------ fetch(url) .then((response) => ***response. text*** ()) .then((textData) => { ... опрацьовуємо надіслані дані data ... })
response.json()	декодує відповідь (JSON.parse(…)), що надіслати у форматі JSON	let response = await fetch(url); let data = await ***response.json***() ----- або ------ fetch(url) .then((response) => ***response.json***()) .then((data) => { ... опрацьовуємо надіслані дані data ... })
response.blob()	дозволяє отримати відповідь як Blob (наприклда, для передачі зображень)	let response = await fetch(url); let blob = await ***response.blob***(); let img = document.createElement('img'); document.body.append(img); img.src = ***URL.createObjectURL(blob)***
response.formData()	дозволяє отримати відповідь як formData	
response.arrayBuffer()	дозволяє отримати відповідь як ArrayBuffer	

Параметри запиту fetch(url,***options***)

options дозволяє передати додаткові параметри

Параметр	Опис	Приклад
method	GET, POST, PUT, DELETE . . . (описує ціль запиту)	fetch(url, { ***method: "GET",*** . . . . . . }) . . . . . .
headers	дозволяє передати додаткові параметри, що стосуються запиту	fetch(url, { method: "GET", ***headers: {*** "Accept": "application/json", "Content-Type": "application/json" ***}*** }) . . . . . .
body	тіло запиту (коли хочемо передати дані користувача, які необхідно передати на сервер (можливо для збереження)	fetch(url, { method: ***"POST"***, ***body:*** деякі_дані }) . . . . . . *---- приклад, надсилаємо JSON ----* *let user = { name: 'John', age:21 }* fetch(url, { ***method: "POST",*** headers: { "Accept": "application/json", "Content-Type": "application/json" }, ***body: JSON.stringify(user)*** })
credentials, cache, redirect, referrer, referrerPolicy, keepalive, …….		

Заголовок запиту

- є об’єктом Headers
- дозволяє передати додаткові параметри (параметри аутентифікації, тип даних, …)
- колекція даних ключ/значення
- список заголовків
  https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D1%96%D0%B2_HTTP

Методи об’єкта ***Headers***

Метод	Опис	Приклад
Конструктор ***new Headers***()	Створення нового об’єкта запиту	const headers = new Headers()
append(key, value)	Додає новий параметр	headers.append("Accept", "application/json")
set (key, value)	Додає/змінює параметр	headers.set("Accept", "text/xml")
delete(key)	Видаляє параметр	headers.delete("Accept")

https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D1%96%D0%B2_HTTP

Надсилання даних

Надсилання простих значень

Загальна форма	Приклад
fetch(url, { method: "POST", body: дані })	fetch(url, { method: "POST", body: "Hello" })

Надсилання JSON

Загальна форма	Приклад
fetch(url, { method: "POST", headers: { "Accept": "application/json", "Content-Type": "application/json" }, body: дані у форматі JSON)	*let user = { name: 'John', age:21 }* fetch(url, { ***method: "POST",*** headers: { "Accept": "application/json", "Content-Type": "application/json" }, ***body: JSON.stringify(user)*** })

Надсилання з використанням FormData

Даний об'єкт автоматично формується, якщо дані пересилаються з HTML-форми form

Методи об'єкта FormData

Метод	Опис	Приклад
Конструктор	Створення нового об'єкта let formData = new FormData()	let formData = new FormData()
formData.append(name, value) formData.set(name, value)	додавання/зміна простого параметра	formData.append('userName', 'Ivan') formData.set ('userName', 'Petro')
formData.append(name, blob, fileName) formData.set(name, blob, fileName)	додавання/зміна параметра – файл (blob) та його ім'я (fileName)	
formData.get(name)	зчитування параметра з ключем name	
formData.getAll(name)	зчитування усіх значень параметрів з ключем name	
formData.has(name)	перевірка існування параметра з ключем name	
formData.delete(name)	видалення параметра з ключем name	
formData.keys()	ітератор ключів	
formData.values()	ітератор значень	
formData.entries()	ітератор пар (**[name, value]**)	
Перебір параметрів	for(let **[name, value]** of **formData**) { . . . . . . }	for(let **[name, value]** of **formData**) { alert(`${name} = ${value}`) }

Приклад надсилання простих даних

Створюємо об’єкт FormData	let ***formData*** = new FormData()
Додаємо дані	***formData***.append(‘userName’, ‘Ivan') ***formData***.append (‘userAge’,21)
Надсилаємо дані	let response = await fetch(url, { method: 'POST', body: ***formData*** })

Приклад надсилання файлу

Елемент розмітки	Picture: <**input type="file"** name=" fileInput" accept="image/*">
Додаємо метод обробки події ***change*** для `input`	***fileInput***.addEventListener("***change***", event => { const files = event.target.files //файли, що було вибрано *uploadFile*(files[0]) //викликаємо функцію, для надсилання файлу })
	async ***function*** *uploadFile****(file)****{*
Створюємо об'єкт FormData	let ***formData*** = new FormData()
Додаємо дані	***formData***.append('file', ***file***)
Надсилаємо дані	let response = await fetch(url, { method: 'POST', headers: { 'Content-Type': 'multipart/form-data; ', }, body: ***formData*** })
	. . . . .
	}

```
<img src="" id="imgPreview" />
<label>
  Picture:
  <input type="file" name="picture" id="fileInput" accept="image/*" />
</label>
<script>
  let url = 'https://jsonplaceholder.typicode.com/posts'
  fileInput.addEventListener('change', (event) => {
    const files = event.target.files //файли, що було вибрано
    readImageAsDataURL(files[0])
    uploadFile(files[0]) //викликаємо функцію, для надсилання файлу
  })
  //функція для зчитування даних і отримання як base64
  function readImageAsDataURL(file) {
    let reader = new FileReader()
    reader.onload = (e) => {
      imgPreview.src = e.target.result
    }
    reader.readAsDataURL(file)
  }
  //функція для надсилання файлу
  async function uploadFile(file) {
    let formData = new FormData()
    formData.append('file', file)
    let response = await fetch(url, {
      method: 'POST',
      body: formData,
    })
  }
</script>
```

Приклад доступу до сайтів з потребою попередньої реєстрації
(з використанням ***apiKey)***

1) Переходимо на потрібний сайт, реєструємось і отримуємо apiKey

2) Додаємо отримайний apiKey у заголовок під час звертання до сторінок сайту

```
function addImage(imgSrc) {
  document.body.insertAdjacentHTML(
    'afterbegin',
    `<img src =${imgSrc} width="100px">`
  )
}
const apiKey ='live_CKS8vMZ8FWMZuXcwYsV4i

//--- масив з шляхами до API ----
const apiEndpoints = {
  allBreadsList: 'https://api.thecatapi.com/v1/images/search?limit=30',

}
async function loadCats() {
  const url = apiEndpoints.allBreadsList
  return new Promise((resolve, reject) => {
    fetch(url, {
      headers: {
        'x-api-key': apiKey,
      },
    })
      .then((response) => response.json())
      .then((data) => {
        data.forEach((cat) => {
          addImage(cat.url)
        })
      })
      .catch((err) => {
        console.log(err)
      })
  })
```

https://developers.thecatapi.com/view-account/ylX4blBYT9FaoVd6OhvR?report=FJkYOq9tW