

MY Github URL

Vercel URL

Video: W12-P1: use xhr to display a simple text

The screenshot shows a developer environment with two main panes. On the left is a code editor with files like index.html, app_02.js, and sample.txt. The app_02.js file contains JavaScript code for an XMLHttpRequest. On the right is a browser window displaying the result of the code execution, titled 'Asynchronous JS - Ajax Demo'.

Code Editor (VS Code):

```
const xhr = new XMLHttpRequest()
console.log('xhr0', xhr)
xhr.open('GET', './api/sample.txt')
console.log('xhr', xhr)

xhr.onreadystatechange = () => {
  console.log('xhr', xhr)
  if (xhr.readyState === 4 && xhr.status === 200) {
    const text = document.createElement('p')
    text.textContent = xhr.responseText
    document.body.appendChild(text)
  } else {
    console.log({
      status: xhr.status,
      text: xhr.statusText
    })
  }
}
xhr.send()
```

Browser Output:

Asynchronous JS - Ajax Demo

李國蘋, 213410102

非同步 JavaScript 及 XML (Asynchronous JavaScript and XML · AJAX) 並不能稱做是種「技術」，而是 2005 年時由 Jesse James Garrett 所發明的術語，描述一種使用數個既有技術的「新」方法。這些技術包括 HTML 或 XHTML、層疊樣式表、JavaScript、文件物件模型、XML、XSLT 以及最重要的 XMLHttpRequest 物件。當這些技術被結合在 Ajax 模型中，Web 應用程式便能快速、即時更動介面及內容，不需要重新讀取整個網頁，讓程式更快回應使用者的操作。

DevTools is now available in Chinese
Don't show again Always match Chrome's language

Elements Console Sources Default levels

No Issues

```
xhr.onreadystatechange: null, readyState: 0, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, ...}
xhr.onreadystatechange: null, readyState: 1, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, ...
xhr.onreadystatechange: null, readyState: 2, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, ...
xhr.onreadystatechange: null, readyState: 3, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, ...
xhr.onreadystatechange: null, readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, ...
response: "非同步 JavaScript 及 XML (Asynchronous JavaScript and XML · AJ..."
```

ca55498 apple550678

Thu May 8 19:07:47 2025 +0800 Video: W12-P1: use xhr to display a sim

Video: W12-P2: click a button to fetch data

The screenshot shows a development environment with several windows open:

- Explorer:** Shows a file tree for "1132-1N-DEMO-02". Directories include "demo", "w02_dom_02", "w02_tictactoe_02", "w03_basics_02", "w03_tictactoe_02", "w04_basics_02", "w05_basics_02", "w06_prep_tests_02", "w10_mid1_p3_02", "w10_product_supas_02", "w11_async_js_02", and "w12_ajax_jsync_02". Files listed under "w12_ajax_jsync_02" are "1-simple-text", "api", "JS app_02.js", and "index.html".
- Code Editor:** The "JS app_02.js" file is open, showing the following code:

```
const btn = document.querySelector('.btn')
btn.addEventListener('click', () => {
  fetchData()
})

const fetchData = () => {
  const xhr = new XMLHttpRequest()
  console.log('xhr', xhr)

  xhr.open('GET', './api/sample.txt')
  xhr.onreadystatechange = () => {
    if (xhr.readyState === 4 && xhr.status === 200) {
      const text = document.createElement('p')
      text.textContent = xhr.responseText
      document.body.appendChild(text)
    }
  }
}
```
- Browser Output:** A browser window titled "Document" shows the result of the fetch operation. It displays the text "李國蘋, 213410102" and a blue button labeled "click me".
- Developer Tools:** The "Network" tab is selected in the developer tools. A red box highlights the request for "sample.txt".

Name	Request URL	Request Method	Status Code	Remote Address	Referrer Policy
app_02.js	http://127.0.0.1:5500/demo/w12_ajax_jsync_02/2-fetch-data/api/sample.txt	GET	304 Not Modified	127.0.0.1:5500	strict-origin-when-cross-origin
index.html					
sample.txt					

ffac14f apple550678

Thu May 8 19:31:07 2025 +0800 Video: W12-P2: click a button to fetch (

Video: W12-P3: Run w10_product_supabase_xx, see how it works

=> `_supabase.from('product_xx').select('*');`

The screenshot shows a development environment with three main panes:

- Code Editor:** Displays a file named `product_supabase_02.js` containing JavaScript code. A red box highlights the line `_supabase.from('product_02').select('*')`. The code uses Supabase's client library to fetch products from a database.
- Browser Preview:** Shows a web page titled "Get Products from Supabase" with a message "李國蘋, 213410102". It displays two products: "1-Emperor Bed" (orange and red patterned chair) and another product (white table with green chairs). Below the first product is the price "\$21.99".
- Developer Tools:** An open DevTools window with the Network tab selected. A red box highlights the "Headers" section. The request details are as follows:
 - Request URL: `https://cokjgk-wbtwhcdfrjhba.supabase.co/rest/v1/product_02?select=*`
 - Request Method: GET
 - Status Code: 200 OK
 - Remote Address: 172.64.149.24:6443
 - Referrer Policy: strict-origin-when-cross-orig in

=> check response

The screenshot shows a development setup with three main components:

- Code Editor:** On the left, a code editor displays a file named `w10_product_supabase_02.js`. A specific line of code is highlighted with a red box:

```
let [ data, error ] = await _supabase.from('product_02').select('*')
```
- Browser Preview:** In the center, a browser window shows a product page titled "Get Products from Supabase". It displays two items: "1-Emperor Bed" and another item partially visible below it.
- Network DevTools:** On the right, the Chrome DevTools Network tab is open, showing a list of requests. One request, "product_02?select...", is highlighted with a red box and expanded to show its response payload. The payload contains an array of products, each with fields like id, title, price, category, img, and remote_img.

```
product_02?select=*
[
  {
    "id": 1,
    "title": "Emperor Bed",
    "price": 21.99,
    "category": "Lid",
    "img": "./images/product-1.jpg",
    "remote_img": "https://example.com/images/product-1.jpg"
  },
  {
    "id": 2,
    "title": "Accent Chair",
    "price": 5.99,
    "category": "Car",
    "img": "./images/product-2.jpg",
    "remote_img": "https://example.com/images/product-2.jpg"
  },
  ...
]
```

=> check how many http requests being done in fetchProducts

The screenshot shows a development setup with three main components:

- Code Editor:** An IDE window displays the file `w10_product_supabase_02.js`. A red box highlights the `fetchProducts` function, which contains a `try...catch` block for querying Supabase. The code also includes a `displayProducts` function.
- Browser Preview:** A browser window shows a product listing titled "Get Products from Supabase" with a user ID "李國蘋, 213410102". It features two items: "1-Emperor Bed" and a second item partially visible below it.
- Network DevTools:** The browser's Network tab is open, showing a list of requests. A red box highlights the "clientSupabase_0..." entry, which is a `fetch` request for the `product_02?select=*` endpoint. Below it, another red box highlights a series of 9 image requests for files named `product-1.jpg` through `product-9.jpg`.

The terminal at the bottom shows the command `use xhr to display a simple text` and the path `apple@apple_lee MINGW64 ~/code/1132-1N-demo-02`.

caa1e3c apple550678

Thu May 8 20:46:37 2025 +0800 Video: W12-P3: Run w10_product_supa_xx,

Video: W12-P4: Fetch person.json string and display name in the browser

=> use `JSON.parse()` to convert `responseText` to JSON array

The screenshot shows a developer environment with two main windows:

- VS Code (Left):** The Explorer sidebar shows a project structure for "1132-1N-demo-02". The "app_02.js" file is open, containing JavaScript code that fetches data from "person.json". A red box highlights the line where `JSON.parse(xhr.responseText)` is used to convert the response text into a JSON object.
- Chrome DevTools (Right):** The "Elements" tab shows the DOM structure of the "Asynchronous JS - Ajax Demo" page. The "Console" tab displays the JSON data received from the API, which is then converted into a string and displayed in the browser. A red box highlights the JSON data in the console log.

Code in app_02.js (highlighted area):

```
const fetchData = () => {
  const xhr = new XMLHttpRequest()
  console.log('xhr0', xhr)

  xhr.open('GET', url)
  console.log('xhr', xhr)

  xhr.onreadystatechange = () => {
    console.log('xhr', xhr)
    if (xhr.readyState === 4 && xhr.status === 200) {
      const data = JSON.parse(xhr.responseText)
      console.log('data in string', JSON.stringify(data))
      const text = document.createElement('p')
      text.textContent = xhr.responseText
      document.body.appendChild(text)
    } else {
      console.log({
        status: xhr.status,
        text: xhr.statusText,
      })
    }
  }
}
```

Console Output in DevTools (highlighted area):

```
[{"id": 1, "name": "李國蘋"}, {"id": 2, "name": "213410102"}, {"id": 3, "name": "apple"}, {"id": 4, "name": "karina"}]
```

Console Output in DevTools (highlighted area):

```
[{"id": 1, "name": "李國蘋"}, {"id": 2, "name": "213410102"}, {"id": 3, "name": "apple"}, {"id": 4, "name": "karina"}]
```

=> extract name from data and show it in the browser

The screenshot shows a developer's environment with two browser windows and a code editor.

Code Editor (VS Code):

- Left Sidebar:** Shows open editors and a tree view of files and folders, including "demo", "w05_basics_02", "w06_prep_tests_02", "w10_mid1_p3_02", "w10_product_supra_02", "w11.async_js_02", "w12_ajax.json_02", "sample.txt", "app_02.js", and "index.html".
- Right Sidebar:** Shows the file structure and content of "app_02.js". The code uses XMLHttpRequest to fetch JSON data from "person.json" and displays it in an HTML page.
- Content Area:** The code defines a function to handle the response from "person.json". It parses the JSON, logs the data, and creates a string representation of the data to insert into a div element.

Browser 1 (Top Left): A Microsoft Edge browser window showing the content of "index.html". It contains a button labeled "click me" which, when clicked, triggers an event listener that logs the user's name and ID to the console.

Browser 2 (Top Right): A Microsoft Edge browser window showing the content of "index.html". It contains a list of names: 李國蘋, 213410102, apple, and karina.

Console (Bottom): Shows the output of the browser's JavaScript console. It includes the log message from the button click and the data object returned from the XMLHttpRequest.

```
const fetchData = () => {
    const xhr = new XMLHttpRequest()
    console.log('xhr0', xhr)

    xhr.open('GET', url)
    console.log('xhr', xhr)

    xhr.onreadystatechange = () => {
        console.log('xhr', xhr)
        if (xhr.readyState === 4 && xhr.status === 200) {
            const data = JSON.parse(xhr.responseText)
            console.log('data', data)
            // console.log('data in string', JSON.stringify(data))
            const displayData = data
                .map((item) => {
                    return `
                        <p> ${item.name} </p>
                    `
                })
                .join('')
            const element = document.createElement('div')
            element.innerHTML = displayData
            document.body.appendChild(element)
        } else {
            console.log({
                status: xhr.status,
                text: xhr.statusText,
            })
        }
    }
    xhr.send()
}

const clickMe = () => {
    console.log('click me')
    const data = [
        {id: 1, name: '李國蘋'},
        {id: 2, name: '213410102'},
        {id: 3, name: 'apple'},
        {id: 4, name: 'karina'}
    ]
    document.getElementById('list').innerHTML = data.map(item => `<p> ${item.name} </p>`).join('')
}
```

```
[{"status": 200, "text": "OK"}]
```

```
[{"id": 1, "name": "\u6587\u5316\u548c\u53d6\u5316"}, {"id": 2, "name": "213410102"}, {"id": 3, "name": "apple"}, {"id": 4, "name": "karina"}]
```

=> check the Network, http response

The screenshot shows a browser window with three tabs: "Document", "Supabase Products Demo", and "Document". The URL is 127.0.0.1:5500/demo/w12_ajax_jsync_02/3-fetch-json/index.html. The main content area displays "Asynchronous JS - Ajax Demo" with a button labeled "click me" and some text below it. A red box highlights the "click me" button and the text area. To the right, the Chrome DevTools Network tab is open, showing a list of requests. One request, "person.json", is highlighted with a red box. The "Response" tab is selected, showing the JSON data:

```
1 [ { "id": 1, "name": "李國蘋" },  
2 { "id": 2, "name": "213410102" },  
3 { "id": 3, "name": "apple" },  
4 { "id": 4, "name": "karina" }  
5 ]  
6  
7
```

beb9422 apple550678 Thu May 8 20:43:40 2025 +0800 ideo: W12-P4: Fetch person.json string :