

My Github URL

W12-P1: use xhr to display a simple text

The screenshot shows the development environment for W12-P1. On the left, the VS Code Explorer shows the file structure of '1132_1N_demo_36'. The 'index.html' file is open, showing a JavaScript snippet that creates an XMLHttpRequest object, opens a GET request to '/api/sample.txt', and displays the response text in a paragraph element. The browser on the right shows the 'Asynchronous JS - Ajax Demo' page, which displays the fetched text: '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。'.

W12-P2: click a button to fetch data

The screenshot shows the development environment for W12-P2. On the left, the VS Code Explorer shows the file structure of '1132_1N_demo_36'. The 'index.html' file is open, showing a JavaScript snippet that adds a click event listener to a button with the id 'btn'. The event listener calls a 'fetchData' function, which creates an XMLHttpRequest object, opens a GET request to '/api/sample.txt', and displays the response text in a paragraph element. The browser on the right shows the 'Asynchronous JS - Ajax Demo' page, which has a 'click me' button. The Network tab in the browser shows the request to '/api/sample.txt' with a status code of 304 (Not Modified).

W12-P3: Run w10_product_supra_xx, see how it works

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

The screenshot shows a web browser displaying a product list titled "Get Products from Supabase" with the URL "127.0.0.1:5500/demo/w10_product_36/product_supa_36.html". The product list includes two items: "1-Emperor Bed" priced at \$21.99 and "2-Accent Chair" priced at \$25.99. The browser's developer tools are open, showing the Network tab with a list of requests. The request for "product_36?select=*" is highlighted, and its Headers tab is open, showing the Request URL, Request Method (GET), Status Code (200 OK), Remote Address, and Referrer Policy.

```
import { supabase } from './clientSupabase_36.js';

const productContainer = document.querySelector('.products-container');

let products_36 = []

const fetchProducts = async () => {
  try {
    let { data, error } = await supabase.from('product_36').select('*');
    console.log('data', data);
    return data;
  } catch (err) {
    console.log(err);
  }
}

const displayProducts = (products) => {
  let productsContent = products
  .map((product) => {
    const { id, title, price, remote_img } = product;
    return `
    <div class="single-product">
      <img
        src=${remote_img}
        class="single-product-img img"
      />
      <footer>
        <h3 class="name">${id}-${title}</h3>
        <span class="price">${price}</span>
      </footer>
    </div>
  `
  })
}
```

=> check response

The screenshot shows the same web browser and developer tools as before, but now the Response tab for the "product_36?select=*" request is open. The response is a JSON array of three product objects. The first product is "1-Emperor Bed" with price 21.99, the second is "2-Accent Chair" with price 25.99, and the third is "3-High-Back Bench" with price 9.99. The code in the background shows the fetchProducts function logging the data.

```
const fetchProducts = async () => {
  try {
    let { data, error } = await supabase.from('product_36').select('*');
    console.log('data', data);
    return data;
  } catch (err) {
    console.log(err);
  }
}
```

=> check how many http requests being done in fetchProducts

The screenshot shows a web application running on a local server. The browser displays a page titled "Get Products from Supabase" with a list of products. The first product is "1-Emperor Bed" priced at \$21.99, and the second is "2-Accent Chair" priced at \$25.99. The network console on the right shows a list of requests, including a WebSocket connection and several fetch requests for product data and images. A red box highlights the fetch requests for product data.

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 web application titled "Asynchronous JS - Ajax Demo". The browser displays a button labeled "click me". The network console on the right shows a fetch request to a JSON endpoint. The response is a JSON array of objects, each containing an id and a name. A red box highlights the responseText, which is a stringified JSON array. Another red box highlights the data property in the console, which is the parsed JSON array. The data array contains four objects: {id: 1, name: '何柏霞'}, {id: 2, name: '213410136'}, {id: 3, name: 'Andy'}, and {id: 4, name: 'Betty'}.

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

The screenshot displays a web application running in a browser at 127.0.0.1:5500. The application is titled "Asynchronous JS - Ajax Demo" and features a button labeled "click me". Below the button, the names of three people are listed: 何柏露, 213410136, Andy, and Betty. The browser's console shows the XMLHttpRequest (XHR) lifecycle, including the readyState changes and the final response data. The response data is an array of objects, each containing an id and a name.

```
app_36.js:10 XMLHttpRequest {onreadystatechange: null, readyState: 0, ...}
app_36.js:13 XMLHttpRequest {onreadystatechange: null, readyState: 1, ...}
app_36.js:16 XMLHttpRequest {readyState: 2, timeout: 0, withCredentials: false, ...}
app_36.js:31 {status: 200, text: 'OK'}
app_36.js:16 XMLHttpRequest {readyState: 3, timeout: 0, withCredentials: false, ...}
app_36.js:31 {status: 200, text: 'OK'}
app_36.js:16 XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, ...}
app_36.js:19 data: (4) [(-), (-), (-), (-)]
  0: {id: 1, name: '何柏露'}
  1: {id: 2, name: '213410136'}
  2: {id: 3, name: 'Andy'}
  3: {id: 4, name: 'Betty'}
  length: 4
  [[Prototype]]: Array(0)
```

=> check the Network, http response

Asynchronous JS - Ajax Demo

何柏霆, 213410136

click me

何柏霆

213410136

Andy

Betty

Network

Filter

AllFetch/XHRDocCSSJSFontImgMediaManifestWSWasmOther

10 ms20 ms30 ms40 ms50 ms60 ms70 ms80 ms90 ms100 ms110

Name

index.html

app_36.js

person.json

Headers

Preview

Response

Initiator

Timing

1

2

3

4

5

6

7

[

{ "id": 1, "name": "何柏霆"},

{ "id": 2, "name": "213410136"},

{ "id": 3, "name": "Andy"},

{ "id": 4, "name": "Betty"}

]

4 requests939 B transferred{ }

0316766vincent560

Thu May 8 20:43:32 2025 +0800

w12

=> git log for w12

Commits

main

All usersAll time

Commits on May 8, 2025

w12

vincent560 committed 1 minute ago · 1 / 1

0316766

5 / 6

