

Video: W13-P1: Use fetch API to replace xhr approach

The image shows a development environment with VS Code on the left and a web browser on the right. The VS Code editor displays the source code for a web application. The browser shows the rendered output of the application, which is an asynchronous JavaScript demo.

VS Code Editor (Left):

- File Explorer: Shows a project structure with folders like 'demo', 'w01_dom_58', 'w02_dom_58', 'w02_tictactoe_58', 'w03_basics_58', 'w03_tictactoe_58', 'w04_basics_58', 'w05_basics_58', 'w10_mid1_m3_58', 'w10_product_su...', 'w11_async_js_58', 'w12_ajax_async...', '1-simple-text', '2-fetch-data', '3-fetch-json', 'w13_fetch...', '3-fetch-j...', 'api', 'JS app_58js U', 'index.ht...', '4-fetch-api', 'api', 'JS app_58js U', 'index.ht...', '.gitignore', '11.png', and 'notes.txt'.
- Code Editor: Shows the source code for 'app_58.js'. The code includes a button click event listener that calls 'getDataByFetchAPI2', which uses the 'fetch' API to retrieve data from 'api/person.json'. The data is then processed by 'displayItems' and rendered to the DOM. A 'getDataByXHR' function is also shown but not used.

Web Browser (Right):

- Address Bar: Shows the URL '127.0.0.1:5500/demo/w13_fetch_async_58/4-fetc...'. The console shows an error 'app_58.js:27'.
- Page Title: 'Asynchronous Js - Ajax Demo'.
- Content: A button labeled 'click me' is present. Below it, the text '陳立偉, 213410458' is displayed. A red box highlights the data returned by the fetch API, showing an array of objects:

```
data (4) [(-), (-), (-), (-)]
  0: {id: 1, name: '陳立偉'}
  1: {id: 2, name: '213410458'}
  2: {id: 3, name: 'john'}
  3: {id: 4, name: 'amy'}
  length: 4
  [[Prototype]]: Array(0)
```

b28f70b htchung Thu May 15 19:16:22 2025 +0800 W13-P1: Use fetch API to replace xhr approach

Video: W13-P2: Use Async Await to replace fetch API approach

The screenshot shows a web application running in a browser at 127.0.0.1:5500/demo/w13_fetch_async_58/5-asy... The application has a title "Asynchronous Js - Async Await Demo" and a button labeled "click me". Below the button, the fetched data is displayed: "陳立偉, 213410458", "john", and "amy".

The browser's developer console shows the following data structure:

```
data (4) [(-), (-), (-), (-)]
  0: {id: 1, name: '陳立偉'}
  1: {id: 2, name: '213410458'}
  2: {id: 3, name: 'john'}
  3: {id: 4, name: 'amy'}
  length: 4
  [[Prototype]]: Array(0)
```

The code in the background shows the implementation of the asynchronous data fetching using Async Await:

```
const btn = document.querySelector('.btn');
const url = './api/person.json';

btn.addEventListener('click', async () => {
  getDataByFetchAPI(url);
});

const getDataAsyncAwait = async (url) => {
  try {
    const response = await fetch(url);
    const data = await response.json();
    console.log('data', data);
    displayItems(data);
  } catch (err) {
    console.log(err);
  }
};

const getDataByFetchAPI = (url) => {
  fetch(url)
    .then((response) => response.json())
    .then((data) => {
      console.log('data', data);
      displayItems(data);
    })
    .catch((err) => console.log(err));
};

const displayItems = (persons) => {
  // ...
};

const getDataByXHR = () => {
  // ...
};
```

1d45c2f htchung Thu May 15 19:38:52 2025 +0800 W13-P2: Use Async Await to replace fetch API

Video: W13-P3: Get meals about cheese from TheMealDB

The screenshot displays a web development environment with two main windows. The left window shows a code editor with JavaScript code for fetching data from TheMealDB. The right window shows a live demo of the API.

Code Editor (Left Window):

```
const btn = document.querySelector('.btn');
const url = 'https://www.themealdb.com/api/json/v1/1/search.php?s=cheese';

btn.addEventListener('click', async () => {
  getDataAsyncAwait(url);
});

const getDataAsyncAwait = async (url) => {
  try {
    const response = await fetch(url);
    const data = await response.json();
    console.log('data.meals', data.meals);
    displayItems(data.meals);
  } catch (err) {
    console.log(err);
  }
};

const displayItems = (data) => {
  const displayData = data
  .map((item) => {
    return `
    <p>${item.strMeal}</p>
  `
  })
  .join('');
  const element = document.createElement('div');
  element.innerHTML = displayData;
  document.body.appendChild(element);
};
```

Live Demo (Right Window):

The live demo shows the results of the API call. The URL bar indicates the request: `127.0.0.1:5500/demo/w13_fetch_async_58/6-meals`. The response is a JSON object with the following structure:

```
{
  "meals": [
    {
      "idMeal": "52779",
      "strArea": "American",
      "strCategory": "Starter",
      "strCreativeCommonsConfirmed": null,
      "strImageSource": null,
      "strIngredient1": "Flour",
      "strIngredient2": "Butter",
      "strIngredient3": "Egg",
      "strIngredient4": "Salt",
      "strIngredient5": "Cheese",
      "strIngredient6": "Milk",
      "strIngredient7": "Eggs",
      "strIngredient8": "Parmesan Cheese",
      "strIngredient9": "Plum tomatoes",
      "strIngredient10": "White Vinegar",
      "strIngredient11": "Basil",
      "strIngredient12": "",
      "strIngredient13": "",
      "strIngredient14": "",
      "strIngredient15": "",
      "strIngredient16": null,
      "strIngredient17": null,
      "strIngredient18": null,
      "strIngredient19": null,
      "strIngredient20": null,
      "strInstructions": "Crust: make a dough from 250g flour (I like mixing",
      "strMeal": "Cream Cheese Tart",
      "strMealThumb": "https://www.themealdb.com/images/media/meals/wurru14",
      "strMeasure1": "250g",
      "strMeasure2": "125g",
      "strMeasure3": "1",
      "strMeasure4": "Pinch",
      "strMeasure5": "300g",
      "strMeasure6": "100ml milk",
      "strMeasure7": "3"
    }
  ]
}
```

601a109 htchung Thu May 15 20:02:17 2025 +0800 W13-P3: Get meals about cheese from TheMealDB

Video: W13-P4: Get Products from local json and from API

=> Get products from local json

The screenshot shows a VS Code editor with two files open: `product_localjson_58.js` and `product_api_58.js`. The `product_localjson_58.js` file contains the following code:

```
const url = './api/javascript-store-products.json';

let products_58 = [];

const fetchProducts = async (url) => {
  try {
    const response = await fetch(url);
    const data = await response.json();
    console.log('data', data);
    return data;
  } catch (err) {
    console.log(err);
  }
};

const displayProducts = (products) => {
  let productsContent = products
  .map((product) => {
    const { name, price, image } = product.fields;
    return `
    <div class="single-product">
      <img
        src=${image[0].url}
        class="single-product-img"
        alt=${name}
      />
      <footer>
        <h3 class="name">${name}</h3>
        <span class="price">${price}</span>
      </footer>
    </div>
  `;
  });
  productContainer.innerHTML = productsContent;
  document.addEventListener('DOMContentLoaded', async () => {
    products_58 = await fetchProducts(url);
    //console.log('products_58', products_58);
    displayProducts(products_58);
  });
};
```

The web browser shows the "Get Products from Supabase" page with the following content:

Get Products from Supabase
陳立偉,213410458

High-Back Bench
\$999

The browser's developer console shows the following data:

```
data product_localjson_58.js:11
(12) [(-), (-), (-), (-), (-), (-), (-), (-), (-), (-), (-), (-)]
  fields: (2) ["#f15025", "#2c3e50"]
    colors: (2) ["#f15025", "#2c3e50"]
    company: "ikea"
    featured: true
    image: Array(1)
      0: {id: "attcv00MkF6G21H", length: 1, [[Prototype]]: Array(0)}
    name: "high-back bench"
    price: 999
    [[Prototype]]: Object
```

=> Get products from API

The screenshot shows a VS Code editor with two files open: `product_api_58.js` and `product_localjson_58.js`. The `product_api_58.js` file contains the following code:

```
const url = 'https://www.course-api.com/javascript-store-products';

let products_58 = [];

const fetchProducts = async (url) => {
  try {
    const response = await fetch(url);
    const data = await response.json();
    console.log('data', data);
    return data;
  } catch (err) {
    console.log(err);
  }
};

const displayProducts = (products) => {
  let productsContent = products
  .map((product) => {
    const { name, price, image } = product.fields;
    return `
    <div class="single-product">
      <img
        src=${image[0].url}
        class="single-product-img"
        alt=${name}
      />
      <footer>
        <h3 class="name">${name}</h3>
        <span class="price">${price}</span>
      </footer>
    </div>
  `;
  });
  productContainer.innerHTML = productsContent;
  document.addEventListener('DOMContentLoaded', async () => {
    products_58 = await fetchProducts(url);
    //console.log('products_58', products_58);
    displayProducts(products_58);
  });
};
```

The web browser shows the "Get Products from Supabase" page with the following content:

Get Products from Supabase
陳立偉,213410458

High-Back Bench
\$999

Albany Table
\$7999

The browser's developer console shows the following data:

```
data product_api_58.js:11
[Five Server] connectIng...
[Five Server] connected.
  fields: (2) ["#f15025", "#2c3e50"]
    colors: (2) ["#f15025", "#2c3e50"]
    company: "ikea"
    featured: true
    image: Array(1)
      0: {id: "rec3w3f0u4p0wag", length: 1, [[Prototype]]: Array(0)}
    name: "high-back bench"
    price: 999
    [[Prototype]]: Object
```

91ef277 htchung Thu May 15 20:59:42 2025 +0800 W13-P4: Get Products from local json and ·