

Github

Vercal

W14-P1: Do products_api_xx.js to get data from an api url

TKU | 2024React.js作品集範例 | Cak | Responsive Mega Menu Exam | 給底薪新手的教學指南 | course-api.com/javascript-st | P1_31 | 重新啟動即可更新

TKU GitHub Supabase Pictures | Unsplash W3Schools 線上成績平台 Project 參考資料 VScode c++ ChatGPT

Products_31 Demo

Liang Chen, 208410331



High-Back Bench
\$999



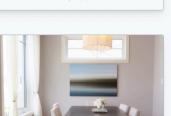
Albany Table
\$7999



Accent Chair
\$2599



Wooden Table
\$4599



Dining Table
\$699



Sofa Set
\$6999



Modern Bookshelf
\$899



Emperor Bed
\$2199



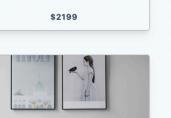
Utopia Sofa
\$3995



Entertainment Center
\$2998



Albany Sectional
\$1099



Leather Sofa
\$999

元素 錄製工具 主控台 原始碼

products_31.js:20 products_api_31.js:13

```
fetch data
  ↴
  ▾ 12: {<...>, <...>, <...>, <...>, <...>, <...>, <...>, <...>, <...>, <...>, <...>, <...>}
```

fields:
 colors: [<...> '#f15025', '#222']
 company: "ikea"
 featured: true
 image: Array(1)
 0:
 filename: "product-1.jpeg"
 height: 639
 id: "attvODMikF6G2jNI"
 size: 62864
 thumbnails: {small: <...>, large: <...>, full: <...>}
 type: "image/jpeg"
 url: "https://course-api.com/images/store/product-1.jpeg"
 width: 1000
 [[Prototype]] https://course-api.com/images/store/
 length: 1 product-1.jpeg
 [[Prototype]]: Array(0)
 name: "High-back bench"
 price: 999
 [[Prototype]]: Object
 id: "rec43wipXvP28vog"
 [[Prototype]]: Object
 1: {id: "rec4f2RIftCbd7Ah", fields: <...>}
 2: {id: "rec8kCm5mkbkiko", fields: <...>}
 3: {id: "recBohCqosotA040II", fields: <...>}
 4: {id: "recD613R2nbPhopy", fields: <...>}
 5: {id: "recNWGyP7kjFhsqW3", fields: <...>}
 6: {id: "recZEouq5bby4AE", fields: <...>}
 7: {id: "recjMKLjgTb2lD7sv", fields: <...>}
 8: {id: "recmg2aictaEJNzhu", fields: <...>}
 9: {id: "recvKMR3Yw0BEl3", fields: <...>}
 10: {id: "recaxFy5iWS39sgM", fields: <...>}
 11: {id: "recyqtRqlGnGt04Q5", fields: <...>}
 length: 12
 [[Prototype]]: Array(0)

主控台 新功能 X

1122-js-demo_31

```

JS product_localJson_31.js U JS products_api_31.js U JS products_31.js U ...
demo > w14_product_31 > js > JS products_api_31.js > [x] DisplayProducts > [x] productsContent > [x] product
1 //import { products_31, all_products_31 } from './p1_data_31.js';
2
3 let products_31 = [];
4
5 const url = 'https://www.course-api.com/javascript-store-products';
6 const productContainer = document.querySelector('.products-container');
7
8 //從網路上抓資料，非同步
9 const fetchData = async (url) => {
10   try {
11     const response = await fetch(url);
12     const data = await response.json();
13     console.log('fetch data', data);
14     return data;
15   } catch (error) {
16     console.log(error);
17   }
18 };
19
20 console.log('products_31', products_31);
21
22 const DisplayProducts = (products) => {
23   /*把products陣列用map轉到另外一個productsContent陣列*/
24   let productsContent = products
25   .map((product) => {
26     const { name, price, image, remote_url } = product.fields;
27     //const { id } = product;
28     return `
29       <div class="single-product">
30         <img
31           src=${image[0].url}
32           class="single-product-img img"
33           alt=${name}
34         />
35       </div>
36     `;
37   })
38   .join('');
39   productContainer.innerHTML = productsContent;
40 }
41
42 //join把陣列自串起來然後顯示 */
43 .join('');
44
45 //const DisplayProducts = (products) => {};
46
47 document.addEventListener('DOMContentLoaded', async () => {
48   //fetchData()先撈資料，從api url抓
49   products_31 = await fetchData(url);
50   DisplayProducts(products_31);
51
52 });
53

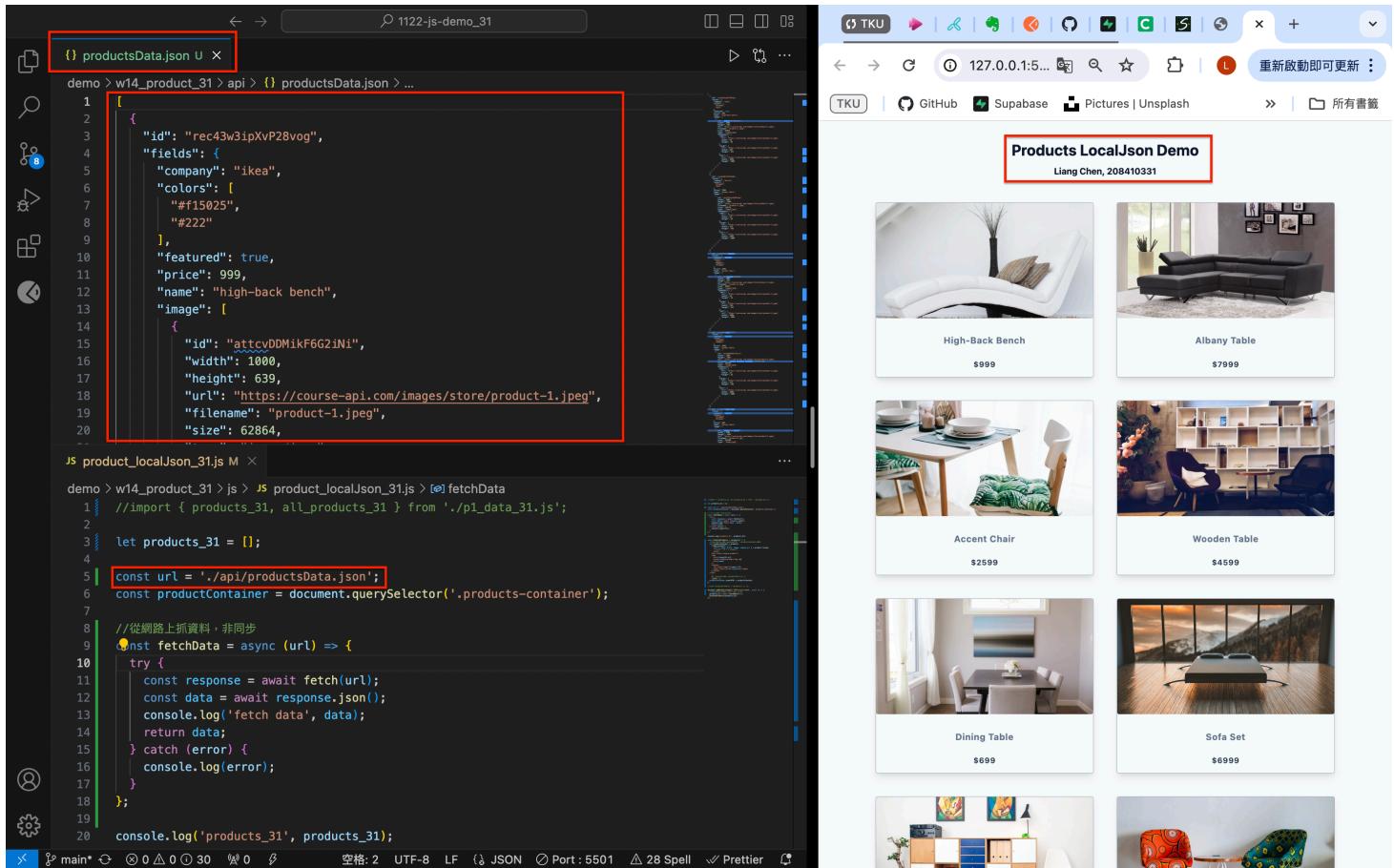
```

問題 10 終端機 連接埠 輸出 個錯主控台 POSTMAN CONSOLE

(base) liangyu@chenliangyudeMacBook-Pro 1122-js-demo_31 %

e018002 liangyu9103 Thu May 23 19:25:22 2024 +0800 ### W14-P1: Do products_api_xx.

W14-P2: Do products_localJson_xx.js to get local json data (/api/productsData.json)



The screenshot illustrates the development process for a local JSON API. On the left, a code editor displays 'product_localJson_31.js' with a red box highlighting the line 'const url = './api/productsData.json';'. This line specifies the local JSON file to be fetched. On the right, a browser window shows a demo page titled 'Products LocalJson Demo' with a header 'Liang Chen, 208410331'. The page lists several products:

- High-Back Bench: \$999
- Albany Table: \$7999
- Accent Chair: \$2599
- Wooden Table: \$4599
- Dining Table: \$699
- Sofa Set: \$6999
- Two small decorative items at the bottom.

8ae90dd liangyu9103

Thu May 23 19:46:41 2024 +0800 ### W14-P2: Do products_localJs

W14-P3: Use SQL to create schemas and data for company_xx and products_xx

=> company_xx schema and data

The screenshot shows the Supabase SQL Editor interface. The left sidebar lists various database objects: company_31, menu_31, mid2_blog_31, SQL for card_31, SQL for company2_31, SQL for product_31, Table for Card Information, task_31, and Wild Oasis. The main area is titled "SQL Editor" and shows the following SQL code:

```
4
5 -- company2_31 table
6
7 CREATE TABLE company2_31 (
8   id serial NOT NULL PRIMARY KEY,
9   created_at timestamp NOT NULL DEFAULT NOW(),
10  name varchar(255),
11  email text,
12  phone varchar(255)
13 );
14
15 ALTER TABLE company2_31 ENABLE ROW LEVEL SECURITY;
16 create policy "Allow SELECT access for all users" on "public"."company2_31" as PERMISSIVE for SELECT to public using (true);
17 create policy "Allow INSERT access for all users" on "public"."company2_31" as PERMISSIVE for INSERT to public with check (true);
18 create policy "Allow UPDATE access for all users" on "public"."company2_31" as PERMISSIVE for UPDATE to public using (true) with
check (true);
19 create policy "Allow DELETE access for all users" on "public"."company2_31" as PERMISSIVE for DELETE to public using (true);
20
21 INSERT INTO company2_31 (name, email, phone)
22 values
23 ('Ikea', 'ikea@gmail.com', '0978513245'),
24 ('Liddy', 'liddy@gmail.com', '079244215'),
25 ('Marcos', 'marcos@gmail.com', '031989235'),
26 ('Caressa', 'caressa@gmail.com', '062528823')
27
```

The results section at the bottom shows the message: "Success. No rows returned". A red box highlights the "Run selected" button in the toolbar.

TKU W13-202 iClass 1122-2N Update liangyu91 tku_31 2024Real 給底片新 course-a Products 重新啟動即可更新

Table Editor liangyu9103's Org Free / tku_31 Enable branching

schema: public + New table Search tables... company2_31 company_31 guests_31 menu_31 mid2_blog_31 product_31 products_31 settings_31 store_31 task_31

Auth policies role postgres Realtime off API Docs

	id int4	created_at timestamp	name varchar	email text	phone varchar
1		2024-05-23 12:29:41.380883	Ikea	ikea@gmail.com	0978513245
2		2024-05-23 12:29:41.380883	Liddy	liddy@gmail.com	079244215
3		2024-05-23 12:29:41.380883	Marcos	marcos@gmail.com	031989235
4		2024-05-23 12:29:41.380883	Caressa	caressa@gmail.com	062528823

Page 1 of 1 100 rows 4 records Refresh Data Definition

=> products_xx schema and data

TKU W13-202 iClass 1122-2N Update liangyu91 tku_31 2024Real 給底片新 course-a Products 重新啟動即可更新

SQL Editor liangyu9103's Org Free / tku_31 Enable branching

+ New query Templates Quickstarts Search queries... YOUR QUERIES company_31 menu_31 mid2_blog_31 SQL for card_31 SQL for company2_31, pro... SQL for product_31 Table for Card Information task_31 Wild Oasis

```

29 -- products2_31 table
30
31
32 CREATE TABLE products2_31 (
33   id serial NOT NULL PRIMARY KEY,
34   created_at timestamp NOT NULL DEFAULT NOW(),
35   title varchar(255),
36   price real,
37   "companyId" int4,
38   "localImg" text,
39   "remoteImg" text,
40   CONSTRAINT fk_1 FOREIGN KEY ("companyId") REFERENCES company2_31(id) ON DELETE SET NULL ON UPDATE CASCADE
41 );
42
43 ALTER TABLE products2_31 ENABLE ROW LEVEL SECURITY;
44 create policy "Allow SELECT access for all users" on "public"."products2_31" AS PERMISSIVE FOR SELECT TO public USING (true);
45 create policy "Allow INSERT access for all users" on "public"."products2_31" AS PERMISSIVE FOR INSERT TO public WITH CHECK (true);
46 create policy "Allow UPDATE access for all users" on "public"."products2_31" AS PERMISSIVE FOR UPDATE TO public USING (true) WITH CHECK (true);
47 create policy "Allow DELETE access for all users" on "public"."products2_31" AS PERMISSIVE FOR DELETE TO public USING (true);
48
49 INSERT INTO products2_31 (title, price, "companyId", "localImg", "remoteImg")
50 VALUES
51 ('Emperor Chair', 21.99, 2, './images/product-1.jpg', 'https://ifafquhqbvtxtzaghzf.supabase.co/storage/v1/object/public/demo_xx_products_xx/product-1.jpg'),
52 ('Accent Chair', 25.99, 4, './images/product-2.jpg', 'https://ifafquhqbvtxtzaghzf.supabase.co/storage/v1/object/public/demo_xx_products_xx/product-2.jpg');

```

Results Chart Export role postgres Run selected %

Success. No rows returned

TKU W13-202 iClass 1122-2N Update liangyu9103 tku_31 2024Real 純底片 course-a Products_ 重新啟動即可更新

TKU GitHub Supabase Pictures | Unsplash W3Schools 線上成績平台 Project 參考資料 VScode c++ ChatGPT 所有書籤

Table Editor liangyu9103's Org Free / tku_31 Enable branching

schema: public + New table Search tables... bookings_31 cabins_31 card_31 company2_31 company_31 guests_31 menu_31 mid2_blog_31 product_31 products2_31 products_31 settings_31 store_31 task_31

Filter Sort Insert Auth policies role postgres Realtime off API Docs

	id	created_at	title	price	co...	i...	localImg	remoteImg
	1	2024-05-23 12:43:13.428962	Emperor Bed	21.99	2	4	./images/product-1.jpg	https://ifafquhvbvtxtzaghzf.sup...
	2	2024-05-23 12:43:13.428962	Accent Chair	25.99	1	4	./images/product-2.jpg	https://ifafquhvbvtxtzaghzf.sup...
	3	2024-05-23 12:43:13.428962	High-Back Bench	9.99	1	1	./images/product-3.jpg	https://ifafquhvbvtxtzaghzf.sup...
	4	2024-05-23 12:43:13.428962	Wooden Table	19.99	1	4	./images/product-4.jpg	https://ifafquhvbvtxtzaghzf.sup...
	5	2024-05-23 12:43:13.428962	Dining Table	6.99	2	4	./images/product-5.jpg	https://ifafquhvbvtxtzaghzf.sup...
	6	2024-05-23 12:43:13.428962	Entertainment Center	21.99	2	2	./images/product-6.jpg	https://ifafquhvbvtxtzaghzf.sup...
	7	2024-05-23 12:43:13.428962	Albany Sectional	21.99	1	1	./images/product-7.jpg	https://ifafquhvbvtxtzaghzf.sup...
	8	2024-05-23 12:43:13.428962	Sofa Set	21.99	2	2	./images/product-8.jpg	https://ifafquhvbvtxtzaghzf.sup...
	9	2024-05-23 12:43:13.428962	Utopia Sofa	21.99	3	2	./images/product-9.jpg	https://ifafquhvbvtxtzaghzf.sup...
	10	2024-05-23 12:43:13.428962	Modern Bookshelf	21.99	3	3	./images/product-10.jpg	https://ifafquhvbvtxtzaghzf.sup...
	11	2024-05-23 12:43:13.428962	Albany Table	21.99	3	2	./images/product-11.jpg	https://ifafquhvbvtxtzaghzf.sup...
	12	2024-05-23 12:43:13.428962	Leather Sofa	21.99	2	2	./images/product-12.jpg	https://ifafquhvbvtxtzaghzf.sup...

Page 1 of 1 100 rows 12 records Refresh Data Definition

3c27898 liangyu9103 Thu May 23 20:52:04 2024 +0800 ### W14-P3: Use SQL to create s

W14-P4: Do products_supaxx.js to get data from Supabase

The screenshot displays a code editor and a browser window side-by-side.

Code Editor: Shows the file `products_supaxx_31.html`. The code uses `supabase` to fetch data from a database table named `products2_31`. It then maps the results to a product list. A specific product, `Emperor Bed`, is highlighted with a red box in both the code and the browser.

```
//import { products_31, all_products_31 } from './p1_data_31.js';
/*從supabase資料庫抓資料*/
import { supabase } from './clientSupabase_31.js';

//console.log('_supabase', _supabase);

let products2_31 = [];

const getProductsSupabase_31 = async () => {
  try {
    let { data, error } = await _supabase.from('products2_31');
    console.log('products data', data);
    return data;
  } catch (error) {
    console.log(error);
  }
};

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

console.log('product_31', products2_31);

const displayProducts = (products) => {
  /*把products陣列用map轉到另外一個productsContent陣列*/
  let productsContent = products.map((product) => {
    const [ title, price, localImg ] = product;
    //const { id } = product;
    return (
      <div class="single-product">
        <img
          src={`${localImg}`}
          class="single-product-img img"
          alt=${title}
        />
        <footer>
          <h3 class="name">${title}</h3>
          <span class="price">${price}</span>
        </footer>
      </div>
    );
  });
  /*join把陣列為自串起來然後顯示 */
  .join('');
}
```

Browser: Shows a grid of furniture items from a Supabase demo. The first item, `Emperor Bed`, is highlighted with a red box. To the right, the browser's developer tools show the JSON data for the `Emperor Bed` product, including its ID, company ID, creation date, localization, price, remote image URL, and title.

`Emperor Bed` details from developer tools:

- companyID: 2
- created_at: "2024-05-23T12:43:13.428962"
- localizing: "./images/product-1.jpg"
- price: 21.99
- remoteImg: "https://ifafquhvqbvtzaghf.supabase.co/v1/object/storage/v1/object/public/Emperor-Bed/Emperor-Bed-1.jpg"
- title: "Emperor Bed"

a79c728 liangyu9103 Thu May 23 21:06:43 2024 +0800 ### W14-P4: Do products_supaxx

W14 all code

```
git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2024-05-22"
```

a79c728 liangyu9103	Thu May 23 21:06:43 2024 +0800	### W14-P4: Do products_supaxx
3c27898 liangyu9103	Thu May 23 20:52:04 2024 +0800	### W14-P3: Use SQL to create s
8ae90dd liangyu9103	Thu May 23 19:46:41 2024 +0800	### W14-P2: Do products_localJs
e018002 liangyu9103	Thu May 23 19:25:22 2024 +0800	### W14-P1: Do products_api_xx.