Web Development front-end native stack

Instructors

Eng. Ahmed Mohamed Abu-Bakr

IC Layout engineer
Full stack web developer Laravel-react

Eng. Nada Harby Motawe

Computer science engineer
Full stack web developer


FRONT END
BACK END
comic.browserling.com

Day 8: Callbacks vs. Promises vs. Async/Await

"Asynchronous code keeps your app responsive. Let’s compare the evolution of handling it!"

A. Callback Hell (The Old Way)

```
getUser(1, (user) => {
        getPosts(user.id, (posts) => {
                getComments(posts[0].id, (comments)
                => {
                console.log(comments); // Nested
                chaos!
                });
        });
});
```

Problem: *Hard to read and debug (Pyramid of Doom).*

B. Promises (ES6)

```
getUser(1)
        .then(user => getPosts(user.id))
        .then(posts =>
        getComments(posts[0].id))
        .then(comments =>
        console.log(comments))
        .catch(err => console.error(err));
```

Improvement: *Chaining instead of nesting.*

Day 8: Callbacks vs. Promises vs. Async/Await

"Asynchronous code keeps your app responsive. Let’s compare the evolution of handling it!"

C. Async/Await (Modern Solution)

```
// API افتراضي للتدريب: https://jsonplaceholder.typicode.com/
// دالة لجلب مستخدم
async function getUser(userId) {
        const response = await fetch(`https://jsonplaceholder.typicode.com/users/${userId}`);
        return response.json();
}
// دالة لجلب منشورات المستخدم
async function getPosts(userId) {
        const response = await fetch(`https://jsonplaceholder.typicode.com/posts?userId=${userId}`);
        return response.json();
}
// دالة لجلب تعليقات على منشور
async function getComments(postId) {
        const response = await fetch(`https://jsonplaceholder.typicode.com/comments?postId=${postId}`);
        return response.json();
}
```

Day 8: Callbacks vs. Promises vs. Async/Await

"Asynchronous code keeps your app responsive. Let’s compare the evolution of handling it!"

C. Async/Await (Modern Solution)

🔁 **Live Coding: Convert a callback-based function to async/await.**

```
async function fetchData() {
    try {
        const user = await getUser(1); // جلب المستخدم الأول
        console.log("User:", user);
        const posts = await getPosts(user.id); // جلب منشوراته
        console.log("Posts:", posts);
        const comments = await getComments(posts[0].id); // جلب تعليقات المنشور الأول
        console.log("Comments:", comments);
        return comments;
    } catch (err) {
        console.error("Error:", err);
    }
}
```

```
// تشغيل الدالة
fetchData().then(result =>
console.log("Final Result:", result));
```

Benefits:

✔️ Synchronous-like readability

✔️ Better error handling with try/catch

Problem: *Hard to read and debug (Pyramid of Doom).*

Day 8: Callbacks vs. Promises vs. Async/Await

"Asynchronous code keeps your app responsive. Let’s compare the evolution of handling it!"

أين تجد API وهمي للتدريب ؟

الوصف	الموقع
API وهمي للمستخدمين، المنشورات، التعليقات	JSONPlaceholder
بيانات الطقس	OpenWeatherMap
بيانات المستخدمين والمستودعات	GitHub API

Day 8: Callbacks vs. Promises vs. Async/Await

"Fetch is JavaScript's gateway to APIs. Let's master it!"

A. Basic GET Request

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
      .then(response => {
            if (!response.ok) throw new Error('Network error');
            return response.json();
      })
      .then(data => console.log(data))
      .catch(err => console.error(err));
```

Day 8: Callbacks vs. Promises vs. Async/Await

"Fetch is JavaScript’s gateway to APIs. Let’s master it!"

B. POST/PUT Requests

```
// POST Example
fetch('https://jsonplaceholder.typicode.com/posts', {
        method: 'POST’,
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ title: 'New Post', body: 'Hello World' })
});

// PUT Example (Update)
fetch('https://jsonplaceholder.typicode.com/posts/1', {
        method: 'PUT’,
        body: JSON.stringify({ title: 'Updated Post' })
});
```

Day 8: Callbacks vs. Promises vs. Async/Await

"Fetch is JavaScript's gateway to APIs. Let's master it!"

C. Handling Errors

```
async function fetchWithRetry(url, retries = 3) {
      try {
            const response = await fetch(url);
            if (!response.ok) throw new Error(`HTTP ${response.status}`);
            return await response.json();
      } catch (err) {
            if (retries > 0) return fetchWithRetry(url, retries - 1);
             throw err;
      }
}
```

🛠️ **Lab: Build a function to fetch user data with retry logic.**

Work while drinking coffe

"Day 8: Weather App Project"

"Let’s build an app that fetches live weather data!"

Starter Code:

```
<!DOCTYPE html>
<html>
<head>
	<title>Weather App</title>
	<style>
		.weather-card {
			background: #f0f8ff;
			padding: 20px;
			border-radius: 10px;
		}
	</style>
</head>
<body>
	<input id="cityInput" placeholder="Enter city">
	<button id="searchBtn">Search</button>
	<div id="weatherResult" class="weather-card"></div>
	<script src="app.js"></script>
</body>
</html>
```

Features:

✔️ Search by city name

✔️ Display temperature, humidity, and weather icons

✔️ Save recent searches to localStorage

"Day 8: Weather App Project"

"Let’s build an app that fetches live weather data!"

JavaScript (app.js):

```
const API_KEY = 'YOUR_OPENWEATHER_API_KEY';
const BASE_URL =
'https://api.openweathermap.org/data/2.5/weather';
async function getWeather(city) {
        try {
                const response = await fetch(`${BASE_URL}?q=$
                {city}&appid=${API_KEY}&units=metric`);
                if (!response.ok) throw new Error('City not found');
                const data = await response.json();
                displayWeather(data);
        } catch (err) {
                alert(err.message);
        }
}
```

Features:

✔️ Search by city name
✔️ Display temperature, humidity, and weather icons
✔️ Save recent searches to localStorage

"Day 8: Weather App Project"

"Let’s build an app that fetches live weather data!"

Continue JavaScript (app.js):

```
function displayWeather(data) {
        const { name, main, weather } = data;
        document.getElementById('weatherResult').innerHTML = ` <h2>${name}</h2> <p> 🌡️ Temp: $
        {main.temp}°C</p> <p>💧 Humidity: ${main.humidity}%</p> <p>🌤️ Condition: $
        {weather[0].description}</p> <img src="https://openweathermap.org/img/wn/$
        {weather[0].icon}.png"> `;
}
        document.getElementById('searchBtn').addEventListener('click', () => {
                const city = document.getElementById('cityInput').value; if (city) getWeather(city);
        }
);
```

🌐 **API Reference:**
OpenWeatherMap API Docs

"Day 8: Debugging & Optimization"

"Let’s build an app that fetches live weather data!"

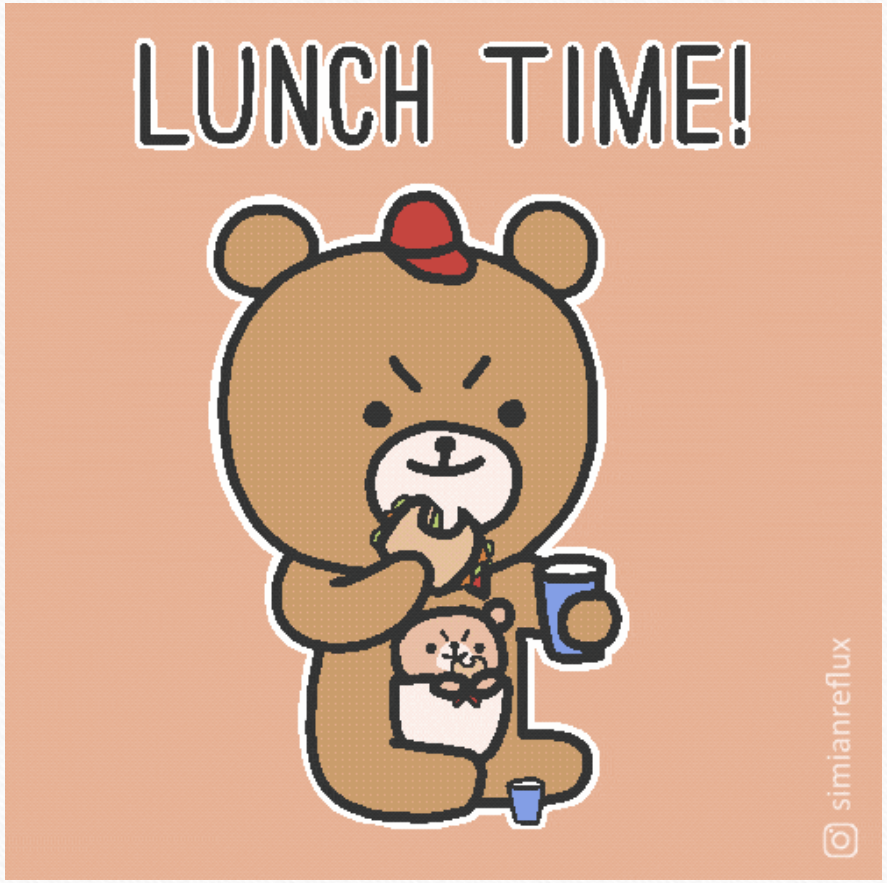
Common Issues & Fixes:

Issue	Solution
CORS errors	Use a proxy or enable CORS server-side
API key not working	Check for typos or regenerate the key
Network failures	Add try/catch and retry logic

Debugging Tools:
Chrome DevTools:
- **Network Tab:** Inspect API requests/responses.
- **Console:** Log intermediate data.

Postman: Test APIs before coding.
🔍 **Exercise:** Debug a pre-built broken weather app.

🍜 Lunch



"Day 8: Project: Shopping Cart with API"

"Let’s build a cart that pulls products from an API!"

Features:

- ✔️ Fetch products from https://fakestoreapi.com/products
- ✔️ Add/remove items (immutable updates)
- ✔️ Calculate totals with reduce

🍵 Break

"Day 8: 🎯 Key Takeaways"

- ✅ **Wrote concise code with ES6+**
- ✅ **Built modular API services**
- ✅ **Processed API data with array methods**
- ✅ **Created an API-driven shopping cart**

Next Day: Asynchronous JS with error handling!

"Day 8:📚 Resources"

MDN Fetch API Guide

Async/Await Explained

OpenWeatherMap API

Next Day: Asynchronous JS with error handling!

"Day 8: 🎯 Key Takeaways"

📂 Project Files

```
day8/
├── index.html
├── app.js
├── styles.css
└── README.md
```

The End

in the end I hope you understood all I said
contact on :

https://www.facebook.com/abobakr143

https://wa.me/201113284597