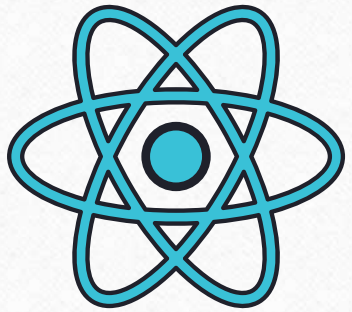
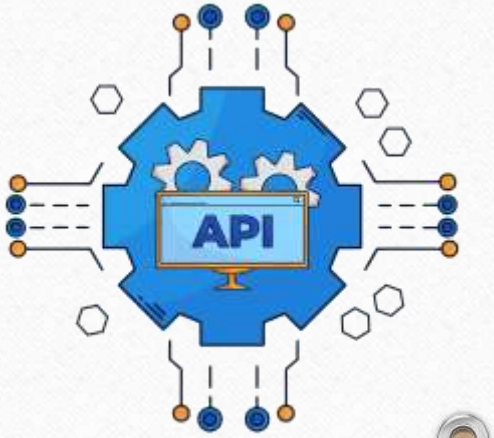


Introduction to Modern JavaScript (ES6+)



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 7: Why Learn ES6+ Features

- ✅ الكود بيبقى أقصر وأنظف
- ✅ أسهل في القراءة والفهم للفرق الكبيرة
- ✅ بيسهّل التعامل مع بيانات ومشاريع معقّدة
- ✅ ضروري جدًا في كل التقنيات الحديثة زي
- ✅ React, Node.js, Vueبيقلل الأخطاء البرمجية بشكل كبير

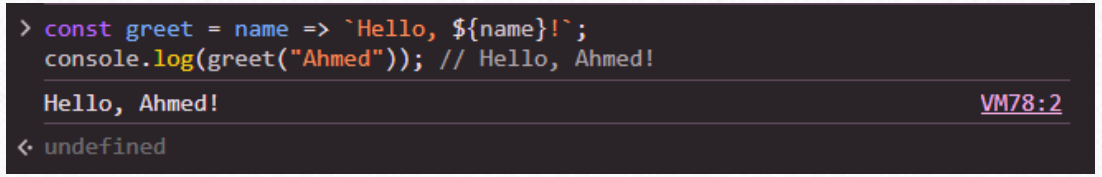
Day 7: Arrow Functions

Arrow Functions طريقة مختصرة لكتابة الدوال في جافاسكريبت، بتسهل الكود، وكمان بتحل مشاكل الـ this في بعض الحالات.

```
const functionName = (parameters) => expression;
```

```
const square = x => x * x;
console.log(square(5)); // 25
```

```
const greet = name => `Hello, ${name}!`;
console.log(greet("Ahmed")); // Hello, Ahmed!
```



لو فيه سطر واحد بيرجع قيمة، تقدر تشيل الأقواس و return.

Day 7: Arrow Functions - Task

🧠 Arrow Functions - Task

❓ السؤال:
عايزك تكتب دالة باسم doubleباستخدام Arrow Function، الدالة تاخد رقم
وترجع الرقم مضروب في 2، وكمان جرّب الدالة باستخدام .console.log

Day 7: Arrow Functions - Task

✅ Answer 1

```
const double = num => num * 2;
console.log(double(8)); // 16
```

```
> const double = num => num * 2;
  console.log(double(8)); // 16
  16
< undefined
```

```
> function double (num) { return num * 2}
  console.log(double(8)); // 16
  16
< undefined
```

Day 7: Destructuring - Arrays & Objects

Destructuringطريقة حديثة لاستخراج القيم من الكائنات أو المصفوفات بشكل مباشر، من غير ما تحتاج تكتب كود طويل.

```
const colors = ["red", "green", "blue"];
const [first, second] = colors;
console.log(first, second); // red green
```

```
const user = { name: "Sara", age: 30 };
const { name, age } = user;
console.log(name, age); // Sara 30
```

```
> const colors = ["red", "green", "blue"];
  const [first, second] = colors;
  console.log(first, second); // red green
  red green
< undefined
```

```
> const user = { name: "Sara", age: 30 };
  const { name, age } = user;
  console.log(name, age); // Sara 30
  Sara 30
< undefined
```

لو فيه سطر واحد بيرجع قيمة، تقدر تشيل الأقواس و.return

Day 7: Destructuring - Task

🧠 Destructuring - Task

❓ السؤال:
عندك الكود التالي، عايزك تستخدم Destructuring لاستخرج الـ username والـ email

```
const profile = {
  username: "ahmed99",
  email: "ahmed@test.com",
  country: "Egypt"
};
```

Day 7: Destructuring - Task

✅ Answer 2 – Destructuring - Task

```
const { username, email } = profile;
console.log(username, email); // ahmed99 ahmed@test.com
```

```
> const profile = {
    username: "ahmed99",
    email: "ahmed@test.com",
    country: "Egypt"
  };
< undefined
> const { username, email } = profile;
  console.log(username, email); // ahmed99 ahmed@test.com
  ahmed99 ahmed@test.com
< undefined
>
```

Day 7: Template Literals

Template Literals طريقة جديدة وأسهل لكتابة النصوص في جافاسكريبت، خصوصًا لو فيها متغيرات أو أكتر من سطر.

- ✅ بنستخدم العلامة `` بدل علامات التنصيص العادية
- ✅ بنقدر نحط متغيرات جوا النص باستخدام ${}
- ✅ بنقدر نكتب نص متعدد الأسطر بسهولة

```
const name = "Ali";
const age = 25;
console.log(`My name is ${name} and I am ${age} years old.`);
```

```
> const name = "Ali";
  const age = 25;
  console.log(`My name is ${name} and I am ${age} years old.`);
  My name is Ali and I am 25 years old.                    VM886:3
< undefined
```

```
const city = "Cairo";
console.log(`Welcome to ${city}!


Enjoy your stay.`);
```

```
> const city = "Cairo";
  console.log(`Welcome to ${city}!
  Enjoy your stay.`);
  Welcome to Cairo!
  Enjoy your stay.
< undefined
```

Day 7: Template Literals - Task

🧠 Template Literals - Task

❓ السؤال:
عندك المتغيرات التالية، عايزك تكتب رسالة ترحيب باستخدام Template Literals:

```
const user = "Salma";
const country = "Egypt";
```

الناتج المطلوب يكون:

“Hello Salma, welcome from Egypt!”

Day 7: Template Literals - Task

✅ Answer : Template Literals - Task

```
console.log(`Hello ${user}, welcome from ${country}!`);
```

```
> const user = "Salma";
  const country = "Egypt";
< undefined
> console.log(`Hello ${user}, welcome from ${country}!`);
  Hello Salma, welcome from Egypt!                    VM908:1
< undefined
```

Day 7: Modern JavaScript Modules

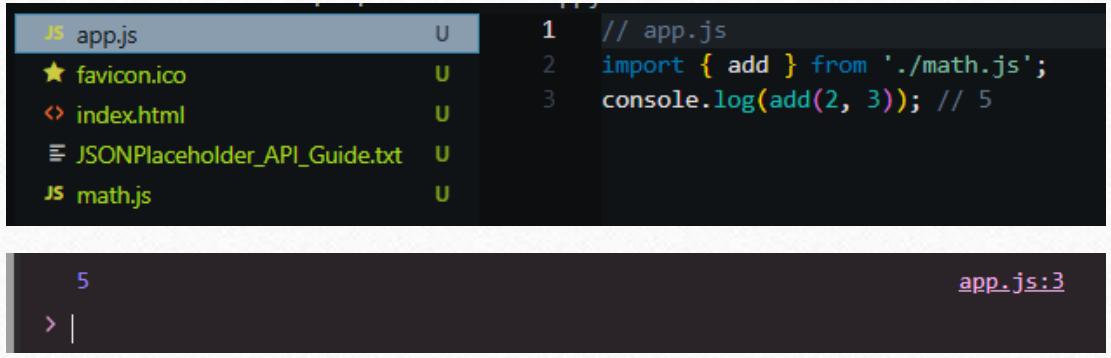
الـ Modules بتقسّم الكود لملفات منفصلة عشان الكود يكون منظم وسهل التوسيع، وبنستخدم export و import لتبادل الأكواد بين الملفات.

- ✅ export: لتصدير دوال أو متغيرات أو كلاسات
- ✅ import: لاستيراد الحاجات دي في ملف تاني

```
// math.js
export const add = (a, b) => a + b;
```

```
// app.js
import { add } from './math.js';
console.log(add(2, 3)); // 5
```

```
<script type="module" src="app.js"></script>
```



Day 7: Modules - Task

🧠 Modules - Task

❓ السؤال:

عايزك تكتب ملف باسم utils.jsفيه دالة باسم square ترجع مربع رقم، وفي ملف main.js استورد الدالة واختبرها مع الرقم 4.

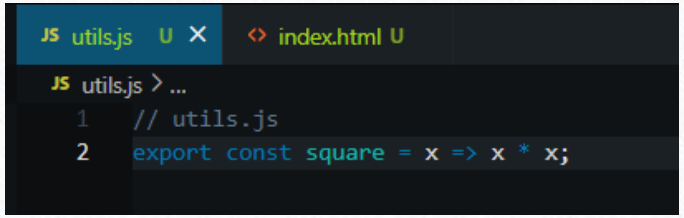
Day 7: Modules - Task

✅ Answer : Modules - Task

```
// utils.js
export const square = x => x * x;
```

```
// main.js
import { square } from './utils.js';
console.log(square(4)); // 16
```

```
<!-- index.html module script tag -->
<script type="module">
  import { square } from './utils.js';
  console.log(square(4)); // 16
</script>
```



Day 7: Classes & Inheritance

الكلاسات بتنظّم الكود بأسلوب الكائنات (OOP) ، بتقدر تعمل كائنات بخصائص ودوال، وكمان تقدر تعمل وراثة بين الكلاسات.

```
class Person {
  constructor(name) {
    this.name = name;
  }
  greet() {
    return `Hello, I'm ${this.name}`;
  }
}
```

main.js > ...

```
1  class Person {
2    constructor(name) {
3      this.name = name;
4    }
5    greet() {
6      return `Hello, I'm ${this.name}`;
7    }
8  }
9
10 var instructor_1 = new Person('ahmed');
11
12 console.log(instructor_1.greet());
13
```

```
Hello, I'm ahmed                                    main.js:12
```

Day 7: Classes & Inheritance

main.js > ...

```
1  class Person {
2    constructor(name) {
3      this.name = name;
4    }
5    greet() {
6      return `Hello, I'm ${this.name}`;
7    }
8  }
9
10 class Student extends Person {
11   constructor(name, level) {
12     super(name);
13     this.level = level;
14   }
15 }
16
17 var instructor_1 = new Person('ahmed');
18 var instructor_2 = new Student('ali', 2);
19
20 console.log(instructor_1.greet());
21 console.log(instructor_2.level , instructor_2.greet());
22
23
```

```
class Student extends Person {
  constructor(name, level) {
    super(name);
    this.level = level;
  }
}
```

```
Hello, I'm ahmed                                    main.js:20
2 "Hello, I'm ali"                                  main.js:21
```

Day 7: Classes - Task

🧠 Classes - Task

❓ السؤال:

اكتب كلاس باسم Product فيه الخصائص: name, price ، ودالة display ترجع نص بالشكل:

"Product: [name], Price: $[price]"

Day 7: Classes - Task

✅ Answer : Classes - Task

```
class Product {
  constructor(name, price) {
    this.name = name;
    this.price = price;
  }
  display() {
    return `Product: ${this.name}, Price: $${this.price}`;
  }
}

const item = new Product("T-shirt", 30);
console.log(item.display());
```

```
24  class Product {
25    constructor(name, price) {
26      this.name = name;
27      this.price = price;
28    }
29    display() {
30      return `Product: ${this.name}, Price: $${this.price}`;
31    }
32  }
33
34  const item = new Product("T-shirt", 30);
35  console.log(item.display());
```

```
Product: T-shirt, Price: $30                                    main.js:35
```

Break

Day 7: Array Methods - map, filter, reduce

المصفوفات من أهم الحاجات في جافاسكريبت، ومع ES6+ بقى عندنا دوال قوية بتسهّل التعامل معاها :

- ✅ map: بتعمل تعديل لكل عنصر وترجع مصفوفة جديدة
- ✅ filter: بترجع العناصر اللي بتحقق شرط معين
- ✅ reduce: بتدمج القيم للحصول على قيمة واحدة (زي المجموع)

```
const numbers = [1, 2, 3];
const doubled = numbers.map(n => n * 2);
console.log(doubled); // [2, 4, 6]
```

```
const prices = [10, 20, 30];
const total = prices.reduce((sum, price) => sum + price, 0);
console.log(total); // 60
```

```
const ages = [15, 22, 17, 30];
const adults = ages.filter(age => age >= 18);
console.log(adults); // [22, 30]
```

```
▸ (3) [2, 4, 6]                    arrays.js:3
▸ (2) [22, 30]                     arrays.js:7
60                                 arrays.js:11
```

Day 7: Array Methods - Task

🧠 Array Methods - Task

❓ السؤال:

عندك مصفوفة أرقام:

```
const numbers = [5, 10, 15, 20];
```

عايزك تعمل الآتي:

- ✅ تستخدم map لمضاعفة الأرقام
- ✅ تستخدم filter لاستخراج الأرقام الأكبر من 10
- ✅ تستخدم reduce لحساب مجموع القيم

Day 7: Array Methods - Task

✅ Answer : Array Methods - Task

```js
// map
const doubled = numbers.map(n => n * 2);
console.log(doubled); // [10, 20, 30, 40]

// filter
const greaterThan10 = numbers.filter(n => n > 10);
console.log(greaterThan10); // [15, 20]

// reduce
const total = numbers.reduce((sum, n) => sum + n, 0);
console.log(total); // 50
```

```js
var numbers = [5, 10, 15, 20];

// map
var doubled = numbers.map(n => n * 2);
console.log(doubled); // [10, 20, 30, 40]

// filter
var greaterThan10 = numbers.filter(n => n > 10);
console.log(greaterThan10); // [15, 20]

// reduce
var total = numbers.reduce((sum, n) => sum + n, 0);
console.log(total); // 50
```

```
▸ (4) [10, 20, 30, 40]                    arr_task.js:5
▸ (2) [15, 20]                            arr_task.js:9
50                                        arr_task.js:13
```

Day 7: Introduction to APIs

https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=https://jsonplaceholder.typicode.com

الـ API هو اختصار لـ Application Programming Interface، وهو عبارة عن وسيط يسمح للتطبيقات بالتواصل مع بعض. في الجزء ده هنتعامل مع API وهمي اسمه JSONPlaceholder عشان نتمرّن على استهلاك بيانات حقيقية.

الرابط الرئيسي:

https://jsonplaceholder.typicode.com

Day 7: JSONPlaceholder API - Available Endpoints

https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=https://jsonplaceholder.typicode.com

موقع JSONPlaceholder بيوفّر بيانات جاهزة للتجارب والتعلّم، منها:

- ✅ /posts – منشورات
- ✅ /comments – تعليقات
- ✅ /albums – ألبومات
- ✅ /photos – صور
- ✅ /to-do’s – مهام
- ✅ /users - بيانات مستخدمين

هنجرب الأمثلة كلها على الموقع ده.

Day 7: Fetch API - Get Data from JSONPlaceholder

باستخدام Fetch API في جافاسكريبت، بنقدر نطلب بيانات من السيرفر ونستخدمها في مشروعنا.

```javascript
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```


```
apis.js:3
(100) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
▸0: {userId: 1, id: 1, title: 'sunt aut facere repellat provident occaecati excepturi
▸1: {userId: 1, id: 2, title: 'qui est esse', body: 'est rerum tempore vitae\nsequi si
▸2: {userId: 1, id: 3, title: 'ea molestias quasi exercitationem repellat qui ipsa sit
▸3: {userId: 1, id: 4, title: 'eum et est occaecati', body: 'ullam et saepe reiciendis
▸4: {userId: 1, id: 5, title: 'nesciunt quas odio', body: 'repudiandae veniam quaerat
▸5: {userId: 1, id: 6, title: 'dolorem eum magni eos aperiam quia', body: 'ut aspernat
▸6: {userId: 1, id: 7, title: 'magnam facilis autem', body: 'dolore placeat quibusdam
▸7: {userId: 1, id: 8, title: 'dolorem dolore est ipsam', body: 'dignissimos aperiam d
▸8: {userId: 1, id: 9, title: 'nesciunt iure omnis dolorem tempora et accusantium', bo
▸9: {userId: 1, id: 10, title: 'optio molestias id quia eum', body: 'quo et expedita m
▸10: {userId: 2, id: 11, title: 'et ea vero quia laudantium autem', body: 'delectus re
▸11: {userId: 2, id: 12, title: 'in quibusdam tempore odit est dolorem', body: 'itaque
```


```html
<script src="apis.js"></script>
```

apis.js > ...


```javascript
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Day 7: Fetch API - Task

🧠 Fetch API - Task

❓ السؤال:

عايزك تكتب كود باستخدام Fetch عشان تجيب بيانات أول 5 منشورات من JSONPlaceholder وتطبعهم في الكونسول.

Day 7: Fetch API - Solution

✅ Answer : Fetch API - Solution

```
6   fetch('https://jsonplaceholder.typicode.com/photos?_limit=5')
7   .then(response => response.json())
8   .then(data => console.log(data))
9   .catch(error => console.error('Error:', error));
10
```

```
fetch('https://jsonplaceholder.typicode.com/posts?_limit=5')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

```
                                                                    apis.js:8
▼ (5) [{…}, {…}, {…}, {…}, {…}]
  ▶ 0: {albumId: 1, id: 1, title: 'accusamus beatae ad facilis cum similique qui sunt', url
  ▶ 1: {albumId: 1, id: 2, title: 'reprehenderit est deserunt velit ipsam', url: 'https://v
  ▶ 2: {albumId: 1, id: 3, title: 'officia porro iure quia iusto qui ipsa ut modi', url: 'h
  ▶ 3: {albumId: 1, id: 4, title: 'culpa odio esse rerum omnis laboriosam voluptate repudia
  ▶ 4: {albumId: 1, id: 5, title: 'natus nisi omnis corporis facere molestiae rerum in', ur
    length: 5
  ▶ [[Prototype]]: Array(0)
>
```

Day 7: Async/Await with JSONPlaceholder

بدل ما نستخدم then و catch بشكل متسلسل، ممكن نستخدم Async/Await عشان الكود يبقى أنضف وأسهل في القراءة، وده بيفيد جدًا مع الـ API.

```
async function fetchPosts() {
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/posts?_limit=5');
    const posts = await response.json();
    console.log(posts);
  } catch (error) {
    console.error('Error fetching posts:', error);
  }
}

fetchPosts();
```

Day 7: Async/Await with JSONPlaceholder

```javascript
async function fetchPosts(theAPISource) {
  try {
    const response = await fetch(theAPISource);
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error('Error fetching posts:', error);
  }
}

fetchPosts('https://jsonplaceholder.typicode.com/posts?_limit=5');
```

```
apis.js:3
(100) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
(5) [{…}, {…}, {…}, {…}, {…}]                apis.js:8
(5) [{…}, {…}, {…}, {…}, {…}]                apis.js:15
> fetchPosts('https://jsonplaceholder.typicode.com/posts?_limit=6')
< Promise {<pending>}
(6) [{…}, {…}, {…}, {…}, {…}, {…}]           apis.js:15
>
```

كدا بنتعامل بمرونة اكتر مع الAPI

بحيث أيا كان الAPI اللي عايزين

نجيب منه داتا هنستعمل معاه

function ثابته واحدة بس

Day 7: Async/Await with JSONPlaceholder

```
23  var arrowPosts = async (theAPISource) => {
24    try {
25      const response = await fetch(theAPISource);
26      const data = await response.json();
27      console.log(data);
28    } catch (error) {
29      console.error('Error fetching posts:', error);
30    }
31  }
32  arrowPosts('https://jsonplaceholder.typicode.com/posts?_limit=5');
```

```
▸ (5) [{…}, {…}, {…}, {…}, {…}]                                    apis.js:27
> arrowPosts("https://jsonplaceholder.typicode.com/posts?_limit=5")
< ▸ Promise {<pending>}
                                                                    apis.js:27
▾ (5) [{…}, {…}, {…}, {…}, {…}] i
  ▸ 0: {userId: 1, id: 1, title: 'sunt aut facere repellat provident occaecati excepturi op
  ▸ 1: {userId: 1, id: 2, title: 'qui est esse', body: 'est rerum tempore vitae\nsequi sint
  ▸ 2: {userId: 1, id: 3, title: 'ea molestias quasi exercitationem repellat qui ipsa sit a
  ▸ 3: {userId: 1, id: 4, title: 'eum et est occaecati', body: 'ullam et saepe reiciendis v
  ▸ 4: {userId: 1, id: 5, title: 'nesciunt quas odio', body: 'repudiandae veniam quaerat su
    length: 5
  ▸ [[Prototype]]: Array(0)
>
```

ودي طريقة نطبق بيها نفس المثال

بس بال arrow function.

Day 7: Async/Await - Task

🧠 Async/Await - Task

❓ السؤال:

عايزك تكتب دالة باستخدام Async/Await تجيب بيانات مستخدم معين (ID=1) من JSONPlaceholder وتطبع اسمه في الكونسول.

Day 7: Async/Await - Task

✅ Answer : Async/Await - Task

```
35  async function getUser(ID) {
36    try {
37      const response = await fetch(`https://jsonplaceholder.typicode.com/users/${ID}`);
38      const user = await response.json();
39      console.log(user.name);
40    } catch (error) {
41      console.error('Error:', error);
42    }
43  }
44
45  getUser(1);
```

```
Leanne Graham                                    apis.js:39
> getUser(2)
  ▸ Promise {<pending>}
Ervin Howell                                     apis.js:39
> getUser(4)
  ▸ Promise {<pending>}
Patricia Lebsack                                 apis.js:39
>
```

الحل دا اقدر اجيب بيه أي بوست من function واحدة ثابتة ودا هيقلل معايا الكود والاسطر ويزود المرونة والسرعة

Day 7: Display API Data in HTML

مش بس نطبع البيانات في الكونسول، كمان نقدر نعرضها داخل عناصر HTML باستخدام DOM.

```
<body>
    <div id="posts-container"></div>
    <script>
        async function displayPosts() {
        const response = await fetch('https://jsonplaceholder.typicode.com/posts?_limit=5');
        const posts = await response.json();

        const container = document.getElementById('posts-container');
        posts.forEach(post => {
            container.innerHTML += `
            <div class="post">
                <h3>${post.title}</h3>
                <p>${post.body}</p>
            </div>
            `;
        });
        }

        displayPosts();

    </script>
```

Day 7: Display API Data - Task

🧠 Display API Data - Task

❓ السؤال:

عايزك تستخدم نفس الأسلوب لعرض بيانات مستخدم معين داخل عنصر HTMLبالشكل التالي:

```
Name: [name]
Email: [email]
```

Day 7: Async/Await - Task

✅ Answer : Async/Await - Task

```
30  <div id="user-container"></div>
31   <script>
32      async function displayUser() {
33          const response = await fetch('https://jsonplaceholder.typicode.com/users/1');
34          const user = await response.json();
35
36          const container = document.getElementById('user-container');
37          container.innerHTML = `
38              <p>Name: ${user.name}</p>
39              <p>Email: ${user.email}</p>
40          `;
41      }
42
43      displayUser();
44   </script>
```

Name: Leanne Graham

Email: Sincere@april.biz

Break

Day 7: Error Handling with Fetch API

لما تتعامل مع API، لازم تتوقع إن في أي وقت ممكن يحصل أخطاء، سواء في الاتصال أو في البيانات، عشان كده لازم نستخدم try/catch ونتأكد من حالة الاستجابة.

```
<script>
    async function fetchData() {
        try {
            const response = await fetch('https://jsonplaceholder.typicode.com/posts/1000');

            if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
            }

            const data = await response.json();
            console.log(data);
        } catch (error) {
            console.error('Error fetching data:', error);
        }
    }

    fetchData();
</script>
```

```
▶ GET https://jsonplaceholder.typicode.com/posts/1000 404 (Not Found)   dom_api.html:50
▶ Error fetching data: Error: HTTP error! status: 404                   dom_api.html:59
    at fetchData (dom_api.html:53:23)
```

Day 7: Error Handling - Task

🧠 Error Handling - Task

❓ السؤال:

اكتب دالة باستخدام Async/Await تجيب بيانات من /users/1000 وتتعامل مع الحالة لو المستخدم مش موجود، وتطبع رسالة واضحة في الكونسول.

Day 7: Error Handling - Solution

✅ Answer : Error Handling - Solution

```
async function getUser() {
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/users/1000');

    if (!response.ok) {
      throw new Error('User not found');
    }

    const user = await response.json();
    console.log(user);
  } catch (error) {
    console.error(error.message);
  }
}

getUser();
```

Day 7: API Recap & Best Practices

أهم النقاط اللي لازم تفتكرها لما تشتغل مع أي API:

- ✅ تأكد من الرابط والـ endpoint
- ✅ استخدم Async/Await لتسهيل الكود
- ✅ دايمًا حط try/catch لعلاج الأخطاء
- ✅ تحقق من حالة الاستجابة (response.ok)
- ✅ لما البيانات تيجي، نظّمها كويس في الواجهة

Day 7: Practical Lab - Complete API Integration

في الجزء العملي ده، هنطبّق كل اللي اتعلمناه:

- ✅ جلب بيانات من JSONPlaceholder
- ✅ عرضها في الصفحة
- ✅ التعامل مع الأخطاء
- ✅ كتابة الكود بأسلوب نظيف ومنظّم

Day 7: Practical Lab - Complete API Integration

🧠 Practical Lab - Complete API Integration

❓ السؤال:

عايزك تعمل صفحة تعرض قائمة مهام (To-Do List) جايه من:

https://jsonplaceholder.typicode.com/todos?_limit=5

لكل مهمة، يظهر:

العنوان

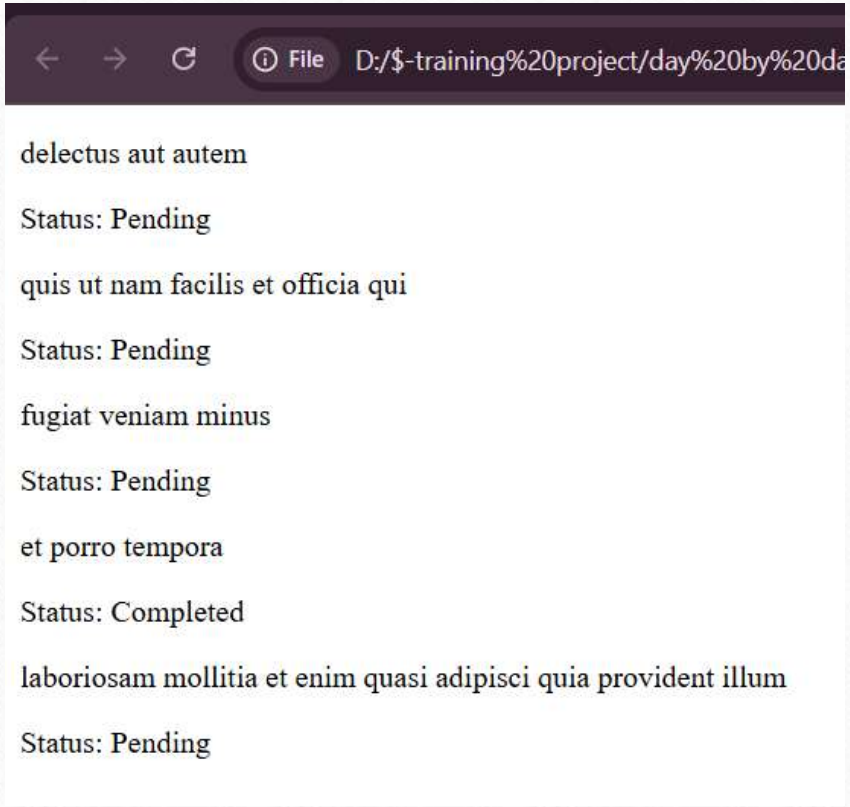
الحالة (مكتملة أو لا)

Day 7: Practical Lab - Suggested Solution

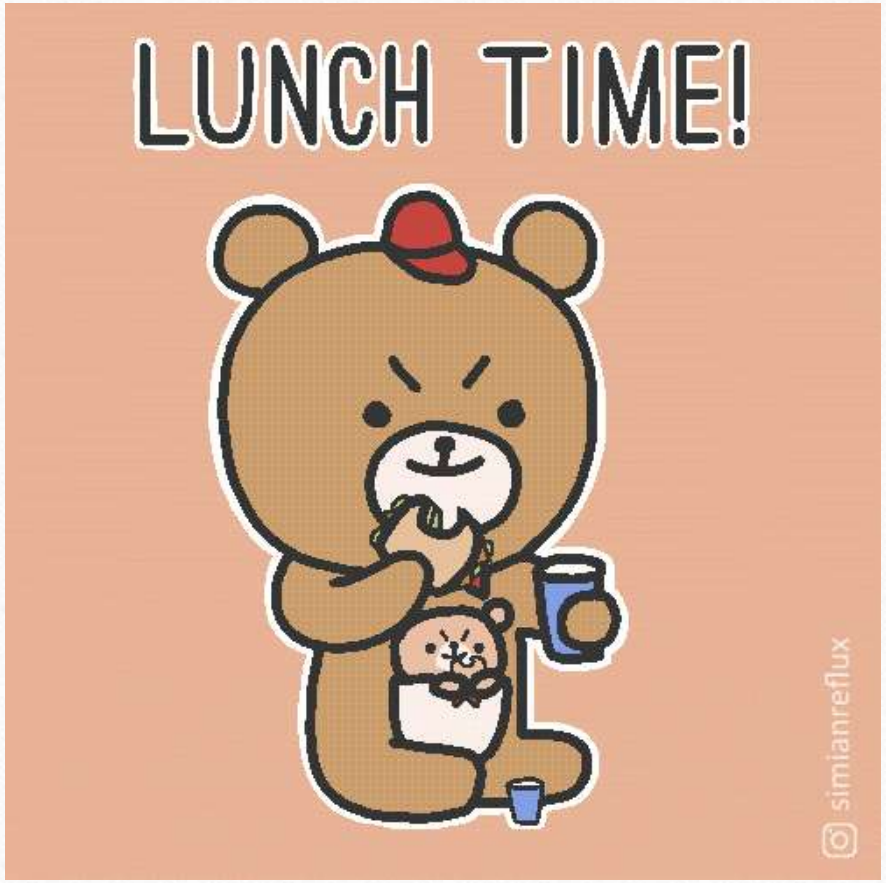
✅ Answer : Practical Lab - Suggested Solution

```html
<div id="todo-list"></div>
<script>
    async function displayTodos(toDoParams) {
        const response = await fetch(toDoParams);
        const todos = await response.json();

        const container = document.getElementById('todo-list');
        todos.forEach(todo => {
            container.innerHTML += `
            <div>
                <p>${todo.title}</p>
                <p>Status: ${todo.completed == true ? 'Completed' : 'Pending'}</p>
            </div>
            `;
        });
    }
    displayTodos('https://jsonplaceholder.typicode.com/todos?_limit=5');
</script>
```



Lunch



Day 7: Advanced Array Methods Recap

راجع معايا بسرعة أهم دوال التعامل مع المصفوفات اللي هتستخدمها يوميًا:

- ✅ map: لإنشاء مصفوفة جديدة بناءً على تعديل القيم
- ✅ filter: لاستخراج عناصر بتحقق شرط
- ✅ reduce: لدمج عناصر المصفوفة لقيمة واحدة
- ✅ find: بترجع أول عنصر يحقق الشرط
- ✅ some: بتتحقق إذا كان فيه على الأقل عنصر يحقق الشرط
- ✅ every: بتتحقق إذا كل العناصر بتحقق الشرط

Day 7: Advanced Array Methods - Examples

```js
const users = [
  { id: 1, name: "Ali" },
  { id: 2, name: "Sara" }
];

const user = users.find(u => u.id === 2);
console.log(user); // { id: 2, name: "Sara" }
```

```js
const numbers = [1, 3, 5, 8];
const hasEven = numbers.some(n => n % 2 === 0);
console.log(hasEven); // true
```

```js
const ages = [20, 25, 30];
const allAdults = ages.every(age => age >= 18);
console.log(allAdults); // true
```

Day 7: Advanced Array Methods - Task

🧠 Advanced Array Methods - Task

❓ السؤال:

عندك المصفوفة دي:

```
const tasks = [
  { id: 1, title: "Task 1", completed: true },
  { id: 2, title: "Task 2", completed: false },
  { id: 3, title: "Task 3", completed: true }
];
```

عايزك تعمل الآتي:

✅ تستخدم find لإيجاد أول مهمة مكتملة

✅ تستخدم some للتحقق إذا فيه أي مهمة غير مكتملة

✅ تستخدم every للتحقق إذا كل المهام مكتملة

Day 7: Advanced Array Methods - Solution

✅ Answer : Advanced Array Methods - Solution

```
const tasks = [
  { id: 1, title: "Task 1", completed: true },
  { id: 2, title: "Task 2", completed: false },
  { id: 3, title: "Task 3", completed: true }
];
```

```
▼ {id: 1, title: 'Task 1', completed: true} i                dom_api.html:132
    completed: true
    id: 1
    title: "Task 1"
  ▶ [[Prototype]]: Object
true                                                         dom_api.html:136
false                                                        dom_api.html:140
>
```

```
// find
const firstCompleted = tasks.find(t => t.completed);
console.log(firstCompleted);

// some
const hasIncomplete = tasks.some(t => !t.completed);
console.log(hasIncomplete); // true

// every
const allCompleted = tasks.every(t => t.completed);
console.log(allCompleted); // false
```

Day 7: Project Structure - Shopping Cart

في المشاريع الواقعية، لازم يكون عندك تنظيم للكود، عشان تعرف تتحكم في الخصائص والعمليات.

هنا هنبني سلة تسوّق بسيطة باستخدام:

- ✅ Class لتنظيم العناصر
- ✅ LocalStorage لحفظ البيانات
- ✅ دوال حساب الإجمالي والخصم

Day 7: Shopping Cart - Class Example

```
<script>
    class ShoppingCart {
        constructor() {
            this.items = JSON.parse(localStorage.getItem('cart')) || [];
        }

        addItem(item) {
            this.items.push(item);
            this._saveToStorage();
        }

        getTotal() {
            return this.items.reduce((sum, item) => sum + item.price, 0);
        }

        clearCart() {
            localStorage.setItem('cart', '[]');
            this.items = [];
        }

        _saveToStorage() {
            localStorage.setItem('cart', JSON.stringify(this.items));
        }
    }

    var cart = new ShoppingCart();
    cart.addItem({ price: 10 });
    cart.addItem({ price: 20 });
    cart.addItem({ price: 70 });

    console.log(cart);
</script>
```

```
▼ ShoppingCart {items: Array(3)} i
  ▼ items: Array(3)
    ▶ 0: {price: 10}
    ▶ 1: {price: 20}
    ▶ 2: {price: 70}
      length: 3
    ▶ [[Prototype]]: Array(0)
  ▶ [[Prototype]]: Object
> localStorage
< ▶ Storage {cart: '[{"price":10},{"price":20},{"price":70}]', length: 1}
> cart.getTotal()
< 100
> cart.clearCart()
< undefined
> cart
< ▶ ShoppingCart {items: Array(0)}
> localStorage
< ▶ Storage {cart: '[]', length: 1}
>
```

Day 7: Advanced Array Methods - Task

🧠 Advanced Array Methods - Task

❓ السؤال:

عايزك تكمل الكلاس بإضافة دالة:

- ✅ removeItem: تقدر تمسح عنصر حسب الـ id
- ✅ clearCart: تفرّغ السلة بالكامل

Day 7: Shopping Cart - Solution

✅ Answer : Shopping Cart - Solution

```
<script>
    class ShoppingCart {
        constructor() {
            this.items = JSON.parse(localStorage.getItem('cart')) || [];
        }
        addItem(item) {
            this.items.push(item);
            this._saveToStorage();
        }
        getTotal() {
            return this.items.reduce((sum, item) => sum + item.price, 0);
        }
        _saveToStorage() {
            localStorage.setItem('cart', JSON.stringify(this.items));
        }
        removeItem(index) {
            if (index >= 0 && index < this.items.length) {
                this.items.splice(index, 1); // remove 1 element at the given index
                this._saveToStorage();
            } else if (index > this.items.length) {
                console.log('out side the range');
            } else if (index < 0 ){
                console.log("so small");
            } else{
                console.log("index is not a number");
            }
        }
        clearCart() {
            this.items = [];
            this._saveToStorage();
        }
    }

    var cart = new ShoppingCart();
</script>
```

```
> cart
< ▸ ShoppingCart {items: Array(0)}
>         cart.addItem({ price: 10 });
          cart.addItem({ price: 20 });
          cart.addItem({ price: 70 });
< undefined
> cart
< ▸ ShoppingCart {items: Array(3)}
> cart.removeItem(-1)
  so small                                    dom_api.html:174
< undefined
> cart.removeItem(100)
  out side the range                          dom_api.html:172
< undefined
> cart.removeItem(10+"aa")
  index is not a number                       dom_api.html:176
< undefined
> cart.removeItem("aa")
  index is not a number                       dom_api.html:176
< undefined
> cart
< ▸ ShoppingCart {items: Array(3)}
```

```
> cart.removeItem(1)
< undefined
> cart
< ▸ ShoppingCart {items: Array(2)}
>
```

Day 7: Shopping Cart - Total with Discount

عايزين نحسّن الكلاس بإضافة خصم تلقائي لو فيه 3 عناصر أو أكتر.

```
getTotal() {
  const subtotal = this.items.reduce((sum, item) => sum + item.price, 0);
  return this.items.length >= 3 ? subtotal * 0.9 : subtotal;
}
```

```
    getTotal() {
        // return this.items.reduce((sum, item) => sum + item.price, 0);
        const subtotal = this.items.reduce((sum, item) => sum + item.price, 0);
        return this.items.length >= 3 ? subtotal * 0.9 : subtotal;
    }
    saveToStorage() {
```

```
> cart.clearCart()
< undefined
>         cart.addItem({ price: 10 });
          cart.addItem({ price: 20 });
          cart.addItem({ price: 70 });
< undefined
> cart.getTotal()
< 90
> cart
< ▾ ShoppingCart {items: Array(3)} i
    ▾ items: Array(3)
      ▸ 0: {price: 10}
      ▸ 1: {price: 20}
      ▸ 2: {price: 70}
        length: 3
      ▸ [[Prototype]]: Array(0)
    ▸ [[Prototype]]: Object
> (70 + 20 + 10)
< 100
> (70 + 20 + 10) * .9
< 90
```

Day 7: Shopping Cart - Task

🧠 Shopping Cart - Task

❓ السؤال:

جرّب الكلاس بإضافة 3 عناصر:

✅ اسم المنتج

✅ السعر

✅ ID

بعدها اطبع الإجمالي وشوف الخصم اتطبق ولا لا.

Day 7: Shopping Cart - Solution

✅ Answer : Shopping Cart - Solution

```
const cart = new ShoppingCart();

cart.addItem({ id: 1, name: "Shirt", price: 30 });
cart.addItem({ id: 2, name: "Jeans", price: 50 });
cart.addItem({ id: 3, name: "Shoes", price: 100 });

console.log(cart.getTotal());
// الخصم %10 اتطبق على الإجمالي
```

```
cart.addItem({ id: 1, name: "Shirt", price: 30 });
cart.addItem({ id: 2, name: "Jeans", price: 50 });
cart.addItem({ id: 3, name: "Shoes", price: 100 });

console.log(cart.getTotal());
// الخصم %10 اتطبق على الإجمالي
162                                                    VM2345:6
undefined
(30+50+100) * .9
162
```

Day 7: Debugging Common Mistakes

كلنا بنقع في أخطاء برمجية، الأهم هو نعرف إزاي نكتشفها ونصلّحها بسرعة.

أمثلة شائعة:

- ❌ لوب شغال بعد نهاية المصفوفة
- ❌ نسيان استخدام
- ❌ return استدعاء دالة مش موجودة
- ❌ مشاكل في التعامل مع LocalStorage

Day 7: Debugging - Buggy Cart Example

```
class BuggyCart {
  constructor() {
    this.items = [];
  }

  addItem(item) {
    this.items.push(item);
  }

  getTotal() {
    let total = 0;
    for (let i = 0; i <= this.items.length; i++) {
      total += this.items[i].price;
    }
    return total;
  }
}
```

المشكلة:

✅ استخدمنا <= بدل < في اللوب،

ده بيدخلنا على عنصر undefined

وبالتالي بيطلع خطأ.

Day 7: Clean Code & Best Practices

```
class BuggyCart {
  constructor() {
    this.items = [];
  }

  addItem(item) {
    this.items.push(item);
  }

  getTotal() {
    let total = 0;
    for (let i = 0; i <= this.items.length; i++) {
      total += this.items[i].price;
    }
    return total;
  }
}
```

أفضل الممارسات اللي لازم تتعود عليها:

- ✅ كود بسيط وواضح
- ✅ الأسماء معبرة وواضحة
- ✅ دوال صغيرة بتركز على مهمة واحدة
- ✅ علاج الأخطاء دايمًا
- ✅ استخدم LocalStorage أو حلول تخزين منظمة
- ✅ اختبر الكود جزء جزء

"Day 7: 📜 Recap - What We Covered Today"

النهاردة مرينا على أهم مفاهيم وأدوات الجافاسكريبت الحديثة اللي بتستخدم يوميًا في أي مشروع واقعي:

- ✅ Arrow Functions - كتابة دوال مختصرة وواضحة
- ✅ Destructuring - استخراج القيم بسهولة من الكائنات والمصفوفات
- ✅ Template Literals - تنسيق النصوص بأسلوب أنظف
- ✅ Modules - تنظيم الكود وتقسيمه لملفات
- ✅ Classes - بناء كائنات منطقية منظمة بأسلوبOOP
- ✅ Array Methods - التعامل الذكي مع المصفوفات (map, filter, reduce, etc)
- ✅ API Integration - التعامل مع بيانات واقعية منJSONPlaceholder
- ✅ Debugging - اكتشاف وتصحيح الأخطاء
- ✅ Best Practices - أسلوب كتابة كود نظيف ومحترف

Day 7: Final Practice - Quick Tasks

🧠 Final Practice - Quick Tasks

❓ السؤال:

- ✅ اكتب Arrow Function ترجع مربع رقم
- ✅ استخدم Destructuring لاستخراج خصائص من كائن
- ✅ استخدم Template Literals لطباعة رسالة ترحيب
- ✅ استخدم fetch لجلب أول 3 منشورات من JSONPlaceholder وطباعتها

Day 7: Final Practice - Quick Tasks

✅ Answer : Final Practice - Quick Tasks

```
// Arrow Function
const square = n => n * n;
console.log(square(4)); // 16

// Destructuring
const user = { name: "Ahmed", age: 28 };
const { name, age } = user;
console.log(name, age);

// Template Literals
console.log(`Hello ${name}, your age is ${age}.`);

// Fetch API
fetch('https://jsonplaceholder.typicode.com/posts?_limit=3')
  .then(response => response.json())
  .then(data => console.log(data));
```

Break

The End

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

https://www.facebook.com/abobakr143

https://wa.me/201113284597