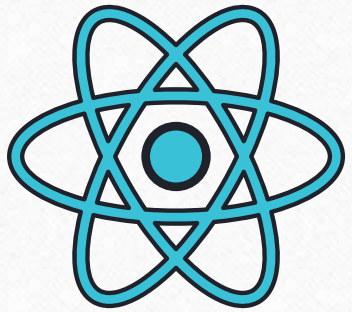
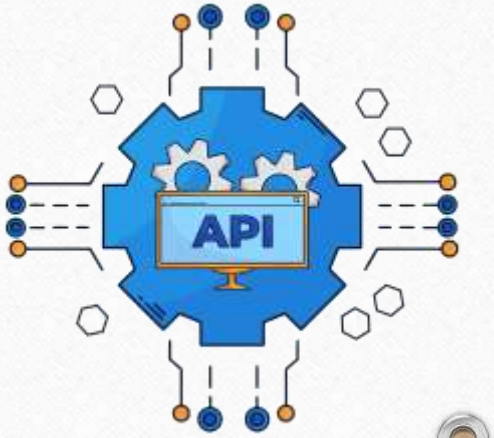


Js intro



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 4: DOM Recap – What, Why & How

“The DOM (Document Object Model) lets JavaScript read and manipulate HTML like a tree. It’s how you make webpages interactive!”

```
const heading = document.getElementById('main-title');
heading.textContent = "Updated Heading";


document.querySelector('.highlight').style.color = 'blue';


const newItem = document.createElement('li');
newItem.textContent = 'New item';
document.getElementById('list').appendChild(newItem);
```

Day 4: Arrays – Data Storage Superpower

“Arrays hold multiple values in one variable. Think of them as lists or collections—great for storing user input, tasks, or results.”

```
const numbers = [10, 20, 30];
console.log(numbers[1]); // 20
```

```
const colors = ['red', 'green'];
colors.push('blue'); // ['red', 'green', 'blue']
colors.pop(); // removes 'blue'
```

```
const mixed = [1, "hello", true];
console.log(typeof mixed[2]); // boolean
```

Day 4: Modifying Arrays – Tools You Need

“These methods—like splice, shift, unshift—give you full control of what’s inside the array.”

```
const books = ['A', 'B'];
books.unshift('Start'); // ['Start', 'A', 'B']
books.shift(); // ['A', 'B']
```

```
const todos = ['Eat', 'Code', 'Sleep'];
todos.splice(1, 1, 'Walk'); // ['Eat', 'Walk', 'Sleep']
```

```
const colors = ['red', 'blue'];
console.log(colors.includes('blue')); // true
```

Break

Day 4: Loops – Repeat Like a Pro

"Loops automate tasks. If you're repeating anything, there's probably a better way using a loop."

```
for (let i = 0; i < 3; i++) {
  console.log(i); // 0, 1, 2
}
```

```
const names = ['Ali', 'Sara'];
for (const name of names) {
  console.log(name);
}
```

```
const user = { name: 'Lina', age: 30 };
for (const key in user) {
  console.log(`${key}: ${user[key]}`);
}
```

Day 4: Lab – DOM-Based Task List

"Let's build a to-do list using DOM methods. It looks real—but doesn't persist."

```html
<input id="taskInput">
<button id="addBtn">Add Task</button>
<ul id="taskList"></ul>
```

```javascript
document.getElementById('addBtn').addEventListener('click', () => {
  const input = document.getElementById('taskInput');
  const li = document.createElement('li');
  li.textContent = input.value;
  document.getElementById('taskList').appendChild(li);
  input.value = '';
});
```

Day 4: Why Data Disappears – Array + DOM

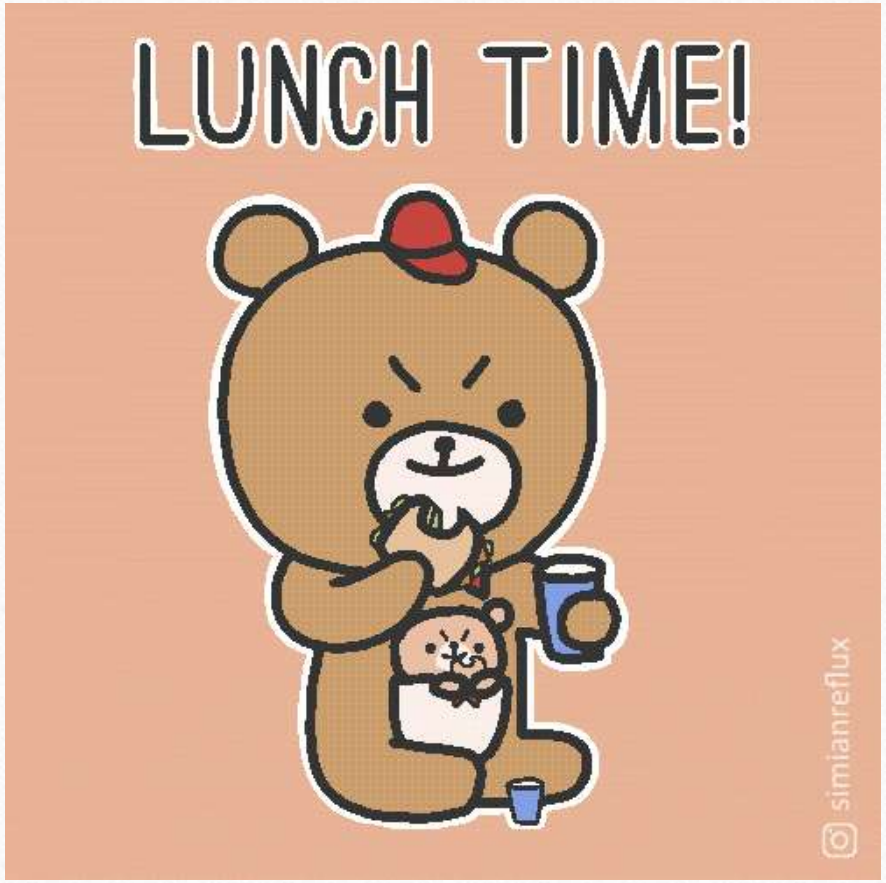
"If we only use the DOM, data disappears on reload. We need a storage system—enter: localStorage."

```
let tasks = [];

function addTask(text) {
  tasks.push(text);
  renderTasks();
}

function renderTasks() {
  document.getElementById('taskList').innerHTML = tasks.map(t => `<li>${t}</li>`).join('');
}
```

Lunch



"Day 4: What is localStorage?"

“localStorage saves your data between page loads. It's like your app's memory.”

```
localStorage.setItem('theme', 'dark');
console.log(localStorage.getItem('theme')); // 'dark'
```

```
const user = { name: 'Ali', age: 22 };
localStorage.setItem('user', JSON.stringify(user));
```

"Day 4: Making the Task List Persistent?"

“Now we’ll make our task manager smart—it will remember tasks using localStorage.”

```
let tasks = JSON.parse(localStorage.getItem('tasks')) || [];

function saveTasks() {
  localStorage.setItem('tasks', JSON.stringify(tasks));
}

function addTask(text) {
  tasks.push({ text, done: false });
  saveTasks();
}
```

"Day 4: Add + Toggle + Delete Tasks"

“Now that we can store tasks, let’s give full control—check/uncheck, delete, and sort.”

```
function toggleTask(id) {
  const task = tasks.find(t => t.id === id);
  if (task) task.done = !task.done;
  saveTasks();
}
```

```
 function deleteTask(id) {
   tasks = tasks.filter(t => t.id !== id);
   saveTasks();
 }
```

Break

"Day 4: Lab – Full UI with Form, Priority, Date"

“Let’s now build a real-world UI—tasks with priority, due date, and full form submission.”

```html
<form id="taskForm">
  <input id="taskInput" required>
  <select id="prioritySelect">
    <option value="low">Low</option>
    <option value="high">High</option>
  </select>
  <input type="date" id="dueDate">
  <button>Add Task</button>
</form>
```

```javascript
document.getElementById('taskForm').addEventListener('submit', (e) => {
  e.preventDefault();
  const input = document.getElementById('taskInput');
  const priority = document.getElementById('prioritySelect').value;
  const dueDate = document.getElementById('dueDate').value;
  tasks.push({ id: Date.now(), text: input.value, priority, dueDate, done: false });
  saveTasks();
  renderTasks();
});
```

"Day 4: Common Mistakes"

“Even pros make mistakes. Let’s look at common bugs and how to avoid them.”

```
// ❌ Reloads the page if not prevented
form.addEventListener('submit', e => e.preventDefault());

// ❌ Not stringifying
localStorage.setItem('user', { name: 'Ali' }); // wrong

// ✅ Correct way
localStorage.setItem('user', JSON.stringify({ name: 'Ali' }));
```

"Day 4: Mini Project + Stats + Search"

“Wrap-up challenge: add search, task stats, sorting. This will prepare you for real-world apps.”

```
function searchTasks(query) {
  return tasks.filter(t => t.text.toLowerCase().includes(query.toLowerCase()));
}
```

```
function getStats() {
  return {
    total: tasks.length,
    done: tasks.filter(t => t.done).length,
    pending: tasks.filter(t => !t.done).length
  };
}
```

The End

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

https://www.facebook.com/abobakr143

https://wa.me/201113284597