



پوهنتون کاردان
KARDAN UNIVERSITY

Modern JavaScript and Browser Storage

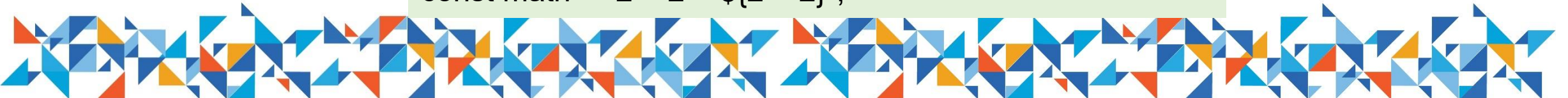
Khalilullah Akbari

k.akbari@kardan.edu.af

Template Literals

- **Definition:** String literals that allow embedded expressions, multi-line strings, and string interpolation using backticks (```).
- **Key Feature:** Uses `${expression}` syntax for interpolation.

```
const name = "Alice";  
const role = "Developer";  
  
// Multi-line and Interpolation  
const greeting = `Hello, ${name}!  
You are a ${role}.  
Next year, you will be a Senior ${role}.`;   
  
// Math inside interpolation  
const math = `2 + 2 = ${2 + 2}`;
```



Optional Chaining (?.)

- **Definition:** An operator that short-circuits and returns undefined if the reference is nullish (null or undefined), preventing runtime errors.
- **Key Feature:** Safely access deeply nested object properties without verbose if checks.

```
const user = {  
  name: "Bob",  
  address: {  
    city: "New York"  
  }  
};
```

```
// Without Optional Chaining (Throws Error if address is missing)  
// const zip = user.address.zipCode;
```

```
// With Optional Chaining (Safely returns undefined)  
const zip = user.address?.zipCode;  
const phone = user.contact?.phone?.number; // Returns undefined
```



Nullish Coalescing (??)

- **Definition:** A logical operator that returns the right-hand operand only when the left-hand operand is null or undefined.
- **Key Feature:** Unlike `||`, it does *not* treat 0, "", or false as fallback triggers.

```
let score = 0;
let username = "";

// Logical OR (||) - Flawed for 0 or empty strings
let finalScore = score || 100; // Returns 100 (Incorrect!)
let displayName = username || "Guest"; // Returns "Guest"
(Incorrect!)

// Nullish Coalescing (??) - Correct behavior
let safeScore = score ?? 100; // Returns 0 (Correct!)
let safeName = username ?? "Guest"; // Returns "" (Correct!)
```



Modules (import / export)

- **Definition:** A native system to organize, encapsulate, and share code across multiple JavaScript files.
- **Key Feature:** Uses export to expose code and import to consume it.

```
// --- math.js ---  
export const PI = 3.14159;  
export function add(a, b) { return a + b; }  
export default function subtract(a, b) { return a - b; }  
  
// --- app.js ---  
import subtract, { PI, add } from './math.js';  
  
console.log(add(2, 3)); // 5  
console.log(subtract(5, 2)); // 3
```

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

JSON (JavaScript Object Notation)

- **Definition:** A lightweight, text-based data interchange format. It is language-independent but uses JavaScript-like syntax.
- **Key Feature:** Easily convert JS objects to strings (**stringify**) and strings back to objects (**parse**).

```
const user = { name: "Charlie", age: 30, active: true };

// 1. Object to JSON String (For sending to APIs / Storage)
const jsonString = JSON.stringify(user);
// Result: '{"name":"Charlie","age":30,"active":true}'

// 2. JSON String to Object (For reading from APIs / Storage)
const parsedUser = JSON.parse(jsonString);
console.log(parsedUser.name); // "Charlie"
```



Local Storage & Session Storage

- **Definition:** Web APIs that allow you to store key-value pairs directly in the user's browser.
- **LocalStorage:** Persists until explicitly cleared (no expiration).
- **SessionStorage:** Clears automatically when the page/tab is closed.

```
setItem("key","value")
```

```
getItem("key")
```

```
removeItem("key")
```

```
const theme = { mode: "dark", fontSize: 16 };
```

```
// Storing data (Must be converted to string via JSON)  
localStorage.setItem("userTheme", JSON.stringify(theme));  
sessionStorage.setItem("tempData", "Hello");
```

```
// Retrieving data  
const savedTheme = JSON.parse(localStorage.getItem("userTheme"));
```

```
// Removing data  
localStorage.removeItem("userTheme");  
localStorage.clear(); // Wipes everything for the domain
```

