1/4 | Narasimha Rao T

# **☆** Important Features of ES6 (ECMAScript 2015):

#### 1. Let and Const

- let: Block-scoped variable declaration.
- const: Block-scoped constant declaration (can't be reassigned).

```
let name = 'John';
const age = 25;
```

#### 2. Arrow Functions

• Shorter syntax for function expressions.

```
const add = (a, b) => a + b;
```

#### 3. Template Literals

• Multi-line strings and string interpolation using backticks (`).

```
let name = 'Alice';
console.log(`Hello, ${name}!`);
```

#### 4. Default Parameters

• Allows default values for function parameters.

```
function greet(name = 'Guest') {
   console.log(`Hello, ${name}`);
}

greeting("Scott"); // Hello Scott
greeting(); // Hello Guest
```

#### 5. Destructuring Assignment

• Extract values from arrays or objects into variables.

```
const user = { name: 'Bob', age: 30 };
const { name, age } = user;
```

2 / 4 | Narasimha Rao T

#### 6. Rest and Spread Operators

• ... for gathering/rest arguments and spreading elements.

```
// Rest
function sum(...numbers) {
    let sumResult =0;

    for(let i in numbers)
    {
        sumResult += numbers[i];
    }
    return sumResult;
}

// Spread
const arr = [1, 2, 3];
const newArr = [...arr, 4, 5];
```

#### 7. Iterators and for...of Loop

• Easily iterate over iterable objects like arrays, strings, etc.

```
const arr = [1, 2, 3];
for (const num of arr) {
   console.log(num);
}
```

## **Ջ localStorage in JavaScript**

localStorage is a Web Storage API that allows you to store key-value pairs in the browser with no expiration.

### **✓** Features of localStorage:

- Data persists even after the browser is closed and reopened.
- Stores strings (you need to JSON.stringify() objects).
- Capacity: 5-10MB depending on the browser.

### **☑** Common localStorage Methods:

```
// Store data
localStorage.setItem('name', 'John');

// Retrieve data
let name = localStorage.getItem('name');
```

3 / 4 | Narasimha Rao T

```
// Remove specific item
localStorage.removeItem('name');

// Clear all data
localStorage.clear();
```

### **✓** Storing Objects/Arrays:

```
const user = { name: 'Alice', age: 25 };
localStorage.setItem('user', JSON.stringify(user));

const retrievedUser = JSON.parse(localStorage.getItem('user'));
console.log(retrievedUser.name); // Alice
```

## 

### **☑** What is Session Storage?

- sessionStorage is part of the Web Storage API that allows you to store key-value pairs per browser tab or window.
- Data is only available for the duration of the page session (until the tab/window is closed).
- Data is stored only as strings.

### **✓** Key Features of sessionStorage:

- Storage is **tab-specific**: Opening a new tab means a new session.
- Data is **cleared automatically** when the tab or window is closed.
- Storage capacity: Usually around **5MB** (browser-dependent).
- Faster access to small amounts of data.

### **☑** Common sessionStorage Methods:

```
// Store data
sessionStorage.setItem('username', 'John');

// Retrieve data
let user = sessionStorage.getItem('username'); // "John"

// Remove a specific item
sessionStorage.removeItem('username');

// Clear all data in sessionStorage
sessionStorage.clear();
```

3 / 4 tnrao.trainer@gmail.com

4 / 4 | Narasimha Rao T

### **✓** Storing Objects/Arrays in sessionStorage:

Just like localStorage, sessionStorage can only store strings.

```
const user = { name: 'Alice', age: 25 };
sessionStorage.setItem('user', JSON.stringify(user));

// Retrieve and parse
const retrievedUser = JSON.parse(sessionStorage.getItem('user'));
console.log(retrievedUser.name); // Alice
```

## **⋄** Difference between localStorage and sessionStorage:

Feature	localStorage	sessionStorage
Persistence	Data persists even after browser/tab is closed	Data cleared when tab/window is closed
Scope	Shared across all tabs/windows of the same origin	Specific to the tab/window where it's set
Capacity	~5MB to 10MB (browser-dependent)	~5MB (browser-dependent)
Storage Type	Key-value pairs (strings)	Key-value pairs (strings)
Use Case Example	Remembering user preferences, theme, login tokens	Temporary form data, session- specific info

## **Example Use Case of sessionStorage:**

```
// Store session data
sessionStorage.setItem('cart', JSON.stringify(['item1', 'item2']));

// Retrieve session data
const cartItems = JSON.parse(sessionStorage.getItem('cart'));
console.log(cartItems); // ['item1', 'item2']
```

# **Ջ** Summary:

- localStorage is best for long-term storage of data that persists between sessions.
- **sessionStorage** is perfect for **temporary data** that should be discarded when the user closes the tab or window (like shopping cart data in a checkout process or one-time form data).