**IP Exp 6**

**Shashwat Tripathi  
D15A 64**

**Batch C**

**Aim:** Experiment to study Advanced JavaScript concepts.

**Document Object Model (DOM):**

Definition: The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content dynamically.

Key Concepts:

1.DOM Tree: The DOM represents an HTML or XML document as a tree structure, where each node in the tree corresponds to a part of the document (e.g., elements, attributes, text).

2.Nodes: Nodes are the fundamental building blocks of the DOM. The main types of nodes include:

•Element Nodes: Represent HTML elements.

•Text Nodes: Contain the text within elements.

•Attribute Nodes: Represent attributes of elements.

3.Properties and Methods: You can access and manipulate the DOM using properties and methods exposed by JavaScript.

•Properties: Allow you to get or set the values of elements or attributes. For example, element.innerHTML gets or sets the HTML content of an element.

•Methods: Enable you to perform actions on elements. For example, element.appendChild(child) adds a child element to another element.

**Event Handling:**

Definition: Event handling in JavaScript allows you to respond to user interactions (e.g., clicks, mouse movements, keyboard inputs) and other events (e.g., page load, form submission) on a web page.

Key Concepts:

1.Event Listeners: You can attach event listeners to DOM elements to listen for specific events. Event listeners specify a callback function to execute when the event occurs.

element.addEventListener('click', function() { // Code to run when the element is clicked });

2.Event Object: When an event occurs, an event object is created and passed to the event handler. It contains information about the event, such as the target element and event type.

element.addEventListener('click', function(event) { console.log(event.target); // Access the element that triggered the event });

Types of Events:

There are various types of events in web development, including but not limited to:

1.Mouse Events:

•click: Occurs when a mouse click is detected.

•mouseover: Triggered when the mouse pointer enters an element.

•mouseout: Fired when the mouse pointer leaves an element.

2.Keyboard Events:

•keydown: Occurs when a key is pressed down.

•keyup: Triggered when a key is released.

3.Form Events:

•submit: Fired when a form is submitted.

•input: Occurs when the value of an input field changes.

4.Document Events:

•load: Triggered when the page finishes loading.

•DOMContentLoaded: Occurs when the HTML document has been fully loaded and parsed.

**Regular Expressions and Validation:**

Regular expressions (regex or regexp) are powerful tools for pattern matching and text manipulation. In JavaScript, you can create and use regular expressions with the RegExp object. Here are some important concepts related to regular expressions:

* Pattern Matching: Regular expressions define patterns to match against strings. For example, /pattern/ is a regular expression pattern.
* Metacharacters: Metacharacters like ^, $, ., \*, +, ?, |, [, ], {, }, and ( ) have special meanings in regular expressions.
* Character Classes: Square brackets [ ] can be used to define character classes. For example, [a-z] matches any lowercase letter.
* Quantifiers: Quantifiers like \*, +, and ? specify the number of times a character or group should be matched.
* Modifiers: You can use modifiers like i (case-insensitive) and g (global) to change how the regular expression behaves.
* Methods: JavaScript provides methods like test() and exec() for matching regular expressions against strings.
* Anchors: Anchors like ^ (start of string) and $ (end of string) are used to specify where a match should occur in the string.
* Capture Groups: Parentheses () are used to create capture groups that allow you to extract specific parts of a matched string.
* Regular expressions are essential for tasks like form validation, data extraction, and text manipulation in JavaScript.

1.Creating Regular Expressions: You can create regular expressions using literal notation (e.g., /pattern/) or the RegExp constructor.

const pattern = /abc/; const dynamicPattern = new RegExp('pattern');

2.Regex Methods:

•test(): Tests if a string matches a pattern and returns a boolean.

•exec(): Searches for a pattern in a string and returns an array with details.

const text = 'abc123'; const pattern = /[a-z]/; console.log(pattern.test(text)); // true

3.Common Patterns: Regex patterns can be used to validate email addresses, phone numbers, dates, and more.

const emailPattern = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/; console.log(emailPattern.test('user@example.com')); // true

4.String Methods: JavaScript provides methods like match(), replace(), and split() for working with regular expressions.

const text = 'abc123def456'; const numbers = text.match(/\d+/g); // ['123', '456']

Regular expressions are powerful tools for validating and manipulating strings in JavaScript, making them essential for tasks like form validation and text processing.

**Conclusion:**

Thus, we have understood the advanced features of JavaScript and implemented that in our programs.