Day4JS: Browser Object Model (BOM) in JavaScript – Detailed Explanation

The **Browser Object Model (BOM)** allows JavaScript to interact with the browser outside of the webpage's content. Unlike the **Document Object Model (DOM)**, which focuses on manipulating HTML and XML documents, BOM provides access to browser functionalities, including:

- The **window** object
- The **navigator** object
- The **location** object
- The **history** object
- The **screen** object

These objects allow developers to control browser behavior, retrieve information about the browser, manipulate windows, and manage browser history.

---

1. Window Object

The window object is the top-level object in BOM, representing the browser window. All global JavaScript functions and objects belong to window.

Key Properties and Methods of window:

1.1 Window Size & Position

```
console.log(window.innerWidth);  // Width of the window
console.log(window.innerHeight); // Height of the window
```

- innerWidth and innerHeight: Get the current viewport dimensions (excluding browser UI like toolbars).
- outerWidth and outerHeight: Include the entire browser window size.

1.2 Window Popups and Alerts

```
window.alert("Hello, welcome to JavaScript!");  // Displays an alert box
let confirmResponse = window.confirm("Are you sure?");
console.log(confirmResponse);  // true (OK) or false (Cancel)
let userInput = window.prompt("Enter your name:");
console.log(userInput);
```

- alert(): Displays a simple alert box.
- confirm(): Asks the user for confirmation (OK or Cancel).
- prompt(): Asks for user input.

1.3 Opening and Closing Windows

```
let newWindow = window.open("https://www.google.com", "_blank",
"width=500,height=500");

newWindow.close(); // Closes the new window
```

- open(URL, target, features): Opens a new browser window.
- close(): Closes the current window.

1.4 Browser Events in the Window Object

```
window.onload = function() {
    console.log("Page fully loaded!");
};

window.onresize = function() {
    console.log("Window resized! New size:", window.innerWidth,
window.innerHeight);
};
```

- onload: Fires when the page is completely loaded.
- onresize: Fires when the browser window is resized.

---

2. Navigator Object

The navigator object provides information about the browser and device being used.

Key Properties of Navigator:

```
console.log(navigator.userAgent);  // Browser information

console.log(navigator.platform);   // OS details (e.g., "Win32" or "Linux x86_64")

console.log(navigator.language);   // Browser language (e.g., "en-US")

console.log(navigator.onLine);     // Checks if the browser is online (true/false)
```

- userAgent: Returns details about the browser.

- platform: Returns the operating system.
- language: Returns the preferred language of the browser.
- onLine: Checks if the browser is connected to the internet.

---

3. Location Object

The location object allows you to get and manipulate the current URL.

Key Properties and Methods of Location:

```
console.log(location.href);     // Returns the current URL
console.log(location.hostname); // Returns the domain name
console.log(location.pathname); // Returns the relative path
console.log(location.protocol); // Returns "http:" or "https:"

// Redirect to another URL
location.href = "https://www.example.com";

// Reload the page
location.reload();
```

- href: Gets or sets the full URL.
- hostname: Returns the domain name.
- pathname: Returns the relative path after the domain.
- protocol: Returns the protocol (http, https).
- reload(): Reloads the current page.

---

4. History Object

The history object allows navigation through the browser's history.

Key Methods of History:

```
history.back();  // Go back to the previous page
```

```
history.forward(); // Go forward to the next page

history.go(-2); // Go back two pages

history.go(2); // Go forward two pages
```

- back(): Goes one step back in the browsing history.
- forward(): Moves one step forward.
- go(n): Moves forward or backward by n pages.

---

5. Screen Object

The screen object provides information about the user's screen.

Key Properties of Screen:

```
console.log(screen.width);  // Full screen width

console.log(screen.height); // Full screen height

console.log(screen.availWidth);  // Available width (excluding taskbars)

console.log(screen.availHeight); // Available height

console.log(screen.colorDepth);  // Color depth of the screen
```

- width and height: Total screen resolution.
- availWidth and availHeight: Excludes taskbars and toolbars.
- colorDepth: Shows color depth in bits per pixel.

Summary: BOM vs. DOM

Feature	BOM	DOM
Purpose	Interacts with the browser	Manipulates HTML content
Root Object	window	document
Key Objects	window, navigator, location, history, screen	document, elements, nodes
Example	window.alert()	document.getElementById("id")

Conclusion

The **Browser Object Model (BOM)** enables JavaScript to interact with the browser outside of the webpage content. It includes:

1. **Window Object** - Controls alerts, pop-ups, browser size, and events.
2. **Navigator Object** - Provides browser and system information.
3. **Location Object** - Allows URL manipulation.
4. **History Object** - Enables navigation through browsing history.
5. **Screen Object** - Retrieves screen resolution and display details.

By mastering BOM, developers can enhance user experience by controlling browser behaviors, handling redirections, and managing windows dynamically. 🚀

Day 5: JavaScript and the Browser

1. Browser Object Model (BOM)

The Browser Object Model (BOM) allows JavaScript to interact with the browser. It provides objects like window, navigator, location, history, and screen.

1.1 Working with the Browser Window

The window object represents the browser window and provides methods to interact with it.

Examples:

// Display an alert box

window.alert("Hello, welcome to JavaScript!");

// Ask for user confirmation

let isConfirmed = window.confirm("Do you want to proceed?");

console.log("User choice:", isConfirmed);

// Prompt user input

let userName = window.prompt("Enter your name:");

console.log("User name:", userName);

==========

1.2 Handling Browser Events

BOM allows handling browser events like page load, resizing, etc.

Examples:

// Execute when the window loads

window.onload = function() {

    console.log("Page fully loaded!");

};

// Execute when the window is resized

window.onresize = function() {

```
    console.log("Window resized! New width: ", window.innerWidth);
};
```

---

2. Timers in JavaScript

JavaScript provides two key timer functions:

- setTimeout: Executes a function after a specified time (one-time execution).
- setInterval: Repeats execution at regular intervals.

2.1 setTimeout Example

```
// Display a message after 3 seconds
setTimeout(function() {
    console.log("This message appears after 3 seconds");
}, 3000);

// Changing background color after 5 seconds
setTimeout(function() {
    document.body.style.backgroundColor = "lightblue";
}, 5000);

// Delaying function execution with parameters
function greet(name) {
    console.log("Hello, " + name + "!");
}
setTimeout(greet, 4000, "Alice");
```

2.2 setInterval Example

```
// Display a message every 2 seconds
let intervalId = setInterval(function() {
```

```
    console.log("This message appears every 2 seconds");
}, 2000);

// Stop the interval after 10 seconds
setTimeout(function() {
    clearInterval(intervalId);
    console.log("Interval stopped");
}, 10000);

// Updating the clock every second
function updateClock() {
    let now = new Date();
    console.log("Current Time: " + now.toLocaleTimeString());
}
setInterval(updateClock, 1000);

// Animate text color change every second
let colors = ["red", "green", "blue", "purple"];
let index = 0;
setInterval(function() {
```

```
    document.body.style.color = colors[index % colors.length];
    index++;
}, 1000);
```

---

3. Working with the Browser Console

The console is useful for debugging and testing JavaScript code.

3.1 Console Methods

```
console.log("This is a log message"); // General log
console.warn("This is a warning message"); // Warning message
console.error("This is an error message"); // Error message
```

3.2 Debugging Example

Use console.log to debug variable values.

```
let num = 10;
console.log("Initial num value:", num);
num += 5;
console.log("Updated num value:", num);
```

3.3 Using console.table for Better Debugging

```
let users = [
    { id: 1, name: "Alice" },
    { id: 2, name: "Bob" }
];
console.table(users);
```

Summary

1. BOM provides window object functions like alert, confirm, prompt.
2. Handling Browser Events like onload and onresize allows interaction with the window.

3. Timers (setTimeout, setInterval) help schedule code execution.
4. Console (console.log, console.warn, console.error, console.table) is useful for debugging.

With these concepts, you can efficiently interact with the browser and debug JavaScript code!
