BOM

The Browser Object Model (BOM) in JavaScript refers to a set of objects provided by web browsers to interact with the browser itself. While the Document Object Model (DOM) represents the structure and content of a web page, the BOM provides objects that represent the browser's interface.

1. Window Object:

The window object represents the browser window or tab and provides methods and properties to manipulate it.

Example: Manipulating Window Properties

```
<body>
<button onclick="resizeWindow()">Resize Window</button>


<script>
function resizeWindow() {
   // Resize the window to a specific width and height
   window.alert (“I am browser object model”);
}
</script>
</body>
```

Method of window object model

1. alert(message):

javascript

```
window.alert("This is an alert message!"); // Displays an alert dialog with the message
```

2. confirm(message):

javascript

```
var result = window.confirm("Are you sure you want to proceed?"); // Displays a confirmation dialog
if (result) {
   console.log("User clicked OK");
} else {
   console.log("User clicked Cancel");
}
```

3. prompt(message, default):

javascript

```
var userInput = window.prompt("Please enter your name:", "John Doe"); // Displays a prompt dialog
console.log("User entered: " + userInput);
```

4. setTimeout(function, milliseconds, args):

javascript

```
function greet(name) {
   console.log("Hello, " + name + "!");
}

window.setTimeout(greet, 2000, "John"); // Executes greet("John") after 2
seconds
```

5. clearTimeout(timeoutID):

javascript

```
var timeoutID = window.setTimeout(function() {
   console.log("This will not be executed.");
}, 2000);

window.clearTimeout(timeoutID); // Cancels the setTimeout before it executes
```

6. setInterval(function, milliseconds, args):

javascript

```
function displayTime() {
   var now = new Date();
   console.log("Current time: " + now.toLocaleTimeString());
}
```

var timer = window.setInterval(displayTime, 1000); // Displays current time every second

7. clearInterval(intervalID):

javascript

```
var intervalID = window.setInterval(function() {
   console.log("This will not be executed.");
}, 1000);

window.clearInterval(intervalID); // Cancels the setInterval before it executes
```

8. open(url, target, specs, replace):

javascript

```
var newWindow = window.open("https://www.example.com", "_blank"); // Opens a new tab with example.com
```

9. close():

javascript

```
window.setTimeout(function() {
   window.close(); // Closes the current browser window after 3 seconds
}, 3000);
```

10. print():

javascript

```
window.print(); // Opens the print dialog for the current document
```

13. scrollTo(x, y):

javascript

```
window.scrollTo(0, 500); // Scrolls the window to the top-left corner with a vertical offset of 500 pixels
```

These examples demonstrate various methods provided by the window object and how they can be used in JavaScript.

2. Navigator Object:

The navigator object provides information about the browser itself, such as its name, version, and platform.

Example: Accessing Browser Information

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Navigator Object Example</title>
</head>
<body>
<script>
// Accessing user agent information
console.log("User Agent:", window.navigator.userAgent);

// Accessing browser version
console.log("Browser Version:", window.navigator.appVersion);

// Accessing preferred language
console.log("Preferred Language:", window.navigator.language);

// Checking if cookies are enabled
```

```
console.log("Cookies Enabled:", window.navigator.cookieEnabled ? "Yes" :
"No");


// Checking online status
console.log("Online Status:", window.navigator.onLine ? "Online" : "Offline");


// Accessing geolocation service
if (window.navigator.geolocation) {
   console.log("Geolocation is supported.");
} else {
   console.log("Geolocation is not supported.");
}
</script>
</body>
</html>
```

3. Screen Object:

The screen object provides information about the user's screen, such as its width, height, and color depth.

Here are some of the common properties of the window.screen object:

1. width: Width of the screen in pixels.

2. height: Height of the screen in pixels.

3. availWidth: Available width of the screen for content in pixels, excluding system UI elements like taskbars.

4. availHeight: Available height of the screen for content in pixels, excluding system UI elements.

5. colorDepth: Number of bits used to represent the color of a single pixel on the screen.

6. pixelDepth: Same as colorDepth, representing the color depth in bits.

7. orientation: An object containing information about the screen's orientation. It usually has properties like angle and type.

For example:

javascript

```
console.log("Screen width:", window.screen.width);

console.log("Screen height:", window.screen.height);

console.log("Available width:", window.screen.availWidth);

console.log("Available height:", window.screen.availHeight);

console.log("Color depth:", window.screen.colorDepth);

console.log("Pixel depth:", window.screen.pixelDepth);

console.log("Orientation:", window.screen.orientation.type);
```

These properties provide information about the screen's dimensions, available space, color capabilities, and orientation, which can be useful for building responsive web applications or adjusting layouts based on screen characteristics.

Example: Getting Screen Information

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Screen Object Example</title>
</head>
<body>
<script>
// Accessing screen width
console.log("Screen Width:", window.screen.width);

// Accessing screen height
console.log("Screen Height:", window.screen.height);

// Accessing screen color depth
console.log("Screen Color Depth:", window.screen.colorDepth);

// Accessing screen pixel depth
```

```
console.log("Screen Pixel Depth:", window.screen.pixelDepth);
console.log("Orientation:", window.screen.orientation.type);
</script>
</body>
</html>
```

4. History Object:

The history object represents the browsing history of the current window.

Example: Navigating through History

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>History Object Example</title>
</head>
<body>
<button onclick="goBack()">Go Back</button>
<button onclick="goForward()">Go Forward</button>

<script>
function goBack() {
   // Navigate back in history
   window.history.back();
}

function goForward() {
   // Navigate forward in history
   window.history.forward();
}
</script>
</body>
</html>
```

5. Location Object:

The location object represents the URL of the current window and provides methods for navigating to different URLs.

Example: Manipulating Location

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Location Object Example</title>
</head>
<body>
<button onclick="redirectToGoogle()">Redirect to Google</button>

<script>
function redirectToGoogle() {
  // Redirect to Google's homepage
```

```
    window.location.href = 'https://www.google.com';
}
</script>
</body>
</html>
```

These examples demonstrate how to utilize various features of the Browser Object Model in JavaScript to interact with the browser environment effectively.