27. JavaScript Window - Browser Object Model

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.

27.1. The Browser Object Model (BOM)

There are no official standards for the **B**rowser **O**bject **M**odel (BOM).

Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

27.2. The Window Object

The **window** object is supported by all browsers. It represent the browsers window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Global variables are properties of the window object.

Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

```
window.document.getElementById("header");
```

is the same as:

```
document.getElementById("header");
```

a) Window Size

Three different properties can be used to determine the size of the browser window (the browser viewport, NOT including toolbars and scrollbars).

For Internet Explorer, Chrome, Firefox, Opera, and Safari:

- window.innerHeight - the inner height of the browser window
- window.innerWidth - the inner width of the browser window

For Internet Explorer 8, 7, 6, 5:

- document.documentElement.clientHeight
- document.documentElement.clientWidth
- or
- document.body.clientHeight
- document.body.clientWidth

A practical JavaScript solution (covering all browsers):

Example

```
var w=window.innerWidth
|| document.documentElement.clientWidth
|| document.body.clientWidth;

var h=window.innerHeight
|| document.documentElement.clientHeight
|| document.body.clientHeight;
```

The example displays the browser window's height and width: (NOT including toolbars/scrollbars)

27.3. Other Window Methods

Some other methods:

- window.open() - open a new window
- window.close() - close the current window
- window.moveTo() -move the current window
- window.resizeTo() -resize the current window

I) JavaScript Window Screen

The window.screen object contains information about the user's screen.

a) Window Screen

The **window.screen** object can be written without the window prefix.

Some properties:

- screen.availWidth - available screen width
- screen.availHeight - available screen height

b) Window Screen Available Width

The screen.availWidth property returns the width of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.

Example

Return the available width of your screen:

```
<script>
document.write("Available Width: " +
screen.availWidth);
</script>
```

The output of the code above will be:

```
Available Width: 1440
```

c) Window Screen Available Height

The screen.availHeight property returns the height of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.

Example

Return the available height of your screen:

```
<script>
document.write("Available Height: " +
```

```
    screen.availHeight);
    </script>
```

The output of the code above will be:

```
    Available Height: 876
```

c) All Screen Properties in One Example

Example

All screen properties in one example:

```
    <!DOCTYPE html>
    <html>
    <body>

    <h3>Your Screen:</h3>

    <script>
    document.write("Total width/height: ");
    document.write(screen.width + "*" + screen.height);
    document.write("<br>");
    document.write("Available width/height: ");
    document.write(screen.availWidth + "*" +
    screen.availHeight);
    document.write("<br>");
    document.write("Color depth: ");
    document.write(screen.colorDepth);
    document.write("<br>");
    document.write("Color resolution: ");
    document.write(screen.pixelDepth);
    </script>

    </body>
    </html>
```

II) JavaScript Window Location

The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.

a) Window Location

The **window.location** object can be written without the window prefix.

Some examples:

- location.hostname returns the domain name of the web host
- location.pathname returns the path and filename of the current page
- location.port returns the port of the web host (80 or 443)
- location.protocol returns the web protocol used (http:// or https://)

b) Window Location Href

The location.href property returns the URL of the current page.

Example

Return the entire URL (of the current page):

```
<script>
document.write(location.href);
</script>
```

The output of the code above is:

```
http://www.iesmanacor.com/js/js_window_location.asp
```

c) Window Location Pathname

The location.pathname property returns the path name of a URL.

Example

Return the path name of the current URL:

```
<script>
document.write(location.pathname);
```

```
    </script>
```

The output of the code above is:

```
    /js/js_window_location.asp
```

d) Window Location Assign

The location.assign() method loads a new document.

Example

Load a new document:

```
    <html>
    <head>
    <script>
    function newDoc()
      {
      window.location.assign("http://www.w3schools.com"
    )
      }
    </script>
    </head>
    <body>

    <input type="button" value="Load new document"
    onclick="newDoc()">

    </body>
    </html>
```

III) JavaScript Window History

The window.history object contains the browsers history.

a) Window History

The **window.history** object can be written without the window prefix.

To protect the privacy of the users, there are limitations to how JavaScript can access this object.

Some methods:

- history.back() - same as clicking back in the browser
- history.forward() - same as clicking forward in the browser

b) Window History Back

The history.back() method loads the previous URL in the history list.

This is the same as clicking the Back button in the browser.

Example

Create a back button on a page:

```
<html>
<head>
<script>
function goBack()
  {
  window.history.back()
  }
</script>
</head>
<body>
<input type="button" value="Back" onclick="goBack()">
</body>
</html>
```

c) Window History Forward

The history forward() method loads the next URL in the history list.

This is the same as clicking the Forward button in the browser.

Example

Create a forward button on a page:

```
<html>
<head>
<script>
function goForward()
  {
  window.history.forward()
  }
</script>
</head>
<body>
<input type="button" value="Forward" onclick="goForward()">
</body>
</html>
```

IV) JavaScript Window Navigator

The window.navigator object contains information about the visitor's browser.

a) Window Navigator

The **window.navigator** object can be written without the window prefix.

Example

```
<div id="example"></div>

<script>
txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
txt+= "<p>User-agent language: " + navigator.systemLanguage +
"</p>";
document.getElementById("example").innerHTML=txt;
</script>
```

b) Warning !!!

The information from the navigator object can often be misleading, and should not be used to detect browser versions because:

- The navigator data can be changed by the browser owner
- Some browsers misidentify themselves to bypass site tests
- Browsers cannot report new operating systems, released later than the browser

c) Browser Detection

Since the navigator object can be misleading about browser detection, using object detection can be used to sniff out different browsers.

Since different browsers support different objects, you can use objects to detect browsers. For example, since only Opera supports the property "window.opera", you can use that to identify Opera.

Example:

```
if (window.opera) {...some action...}
```

V) JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

a) Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
window.alert("sometext");
```

The **window.alert** method can be written without the window prefix.

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("I am an alert box!");
}
</script>
</head>
<body>

<input type="button" onclick="myFunction()"
value="Show alert box">

</body>
</html>
```

b) Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
window.confirm("sometext");
```

The **window.confirm()** method can be written without the window prefix.

Example

```
var r=confirm("Press a button");
if (r==true)
  {
  x="You pressed OK!";
  }
else
  {
  x="You pressed Cancel!";
  }
```

c) Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
window.prompt("sometext","defaultvalue");
```

The **window.prompt()** method can be written without the window prefix.

Example

```
var person=prompt("Please enter your name","Harry
Potter");
if (person!=null && person!="")
  {
  x="Hello " + person + "! How are you today?";
  }
```

d) Line Breaks

To display line breaks inside a popup box, use a back-slash followed by the character n.

Example

```
alert("Hello\nHow are you?");
```