Browser Object Model

**1.** What is BOM

* The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.
* 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.
* BOM provides you with window objects, that can be used for multiple things, for example, to show the width and height of the window, as well as open or close windows.

**2.** What is Window Object

* The window object is supported by all browsers. It represents the browser's window.
* All global JavaScript objects, functions, and variables automatically become members of the window object. Even the document object (of the HTML DOM) is a property of the window object:
* window.document.getElementById("header") is same as document.getElementById("header");
* Two properties can be used to determine the size of the browser window. Both properties return the sizes in pixels:

window.innerHeight - the inner height of the browser window (in pixels)

window.innerWidth - the inner width of the browser window (in pixels)

* The browser window (the browser viewport) is NOT including toolbars and scrollbars.
* 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

**3.** What is Window Screen

* The window.screen object contains information about the user's screen.
* The window.screen object can be written without the window prefix.
* The Properties of window.screen:::

The screen.width property returns the width of the visitor's screen in pixels.

The screen.height property returns the height of the visitor's screen in pixels.

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

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

The screen.pixelDepth property returns the pixel depth of the screen.

The screen.colorDepth property returns the number of bits used to display one color. All modern computers use 24 bit or 32 bit hardware for color resolution:

* 24 bits =      16,777,216 different "True Colors"
* 32 bits = 4,294,967,296 different "Deep Colors"
* Older computers used 16 bits: 65,536 different "High Colors" resolution. Very old computers, and old cell phones used 8 bits: 256 different "VGA colors".
* The #rrggbb (rgb) values used in HTML represents "True Colors" (16,777,216 different colors)

**4.** What is 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. The window.location object can be written without the window prefix. Some examples:

window.location.href returns the href (URL) of the current page

window.location.hostname returns the domain name of the web host

window.location.pathname returns the path and filename of the current page

window.location.protocol returns the web protocol of the page (http: or https:)

The window.location.port property returns the number of the internet host port (of the current page).

window.location.assign() loads a new document

**5.** What is Window History

* The window.history object contains the browsers 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

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

**6.** What is Window Navigator

* The window.navigator object contains information about the visitor's browser. The window.navigator object can be written without the window prefix. Some examples:

navigator.appName

navigator.appCodeName

navigator.platform

* The cookieEnabled property returns true if cookies are enabled, otherwise false:

document.get..ById("demo").innerHTML =navigator.cookieEnabled;

* The appName property returns the application name of the browser. Strange enough, "Netscape" is the application name for both IE11, Chrome, Firefox, and Safari.
* The appCodeName property returns the application code name of the browser. "Mozilla" is the application code name for both Chrome, Firefox, IE, Safari, and Opera.
* The product property returns the product name of the browser engine. Do not rely on this. Most browsers returns "Gecko" as product name !!
* The appVersion property returns version information about the browser:
* The userAgent property returns the user-agent header sent by the browser to the server:
* The platform property returns the browser platform (operating system):
* The language property returns the browser's language:
* The onLine property returns true if the browser is online:
* The javaEnabled() method returns true if [Java](https://www.w3schools.com/java/default.asp) is enabled:

**7.** What is Popup Boxes.

* JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt 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: window.alert("*text*")/window.alert("*text*")
* 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**. window.confirm("*sometext*");
* 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. window.prompt("*sometext*","*defaultText*");
* To display line breaks inside a popup box, use a back-slash followed by the character n.

**8.** What is JS Timing Events.

* The window object allows execution of code at specified time intervals. These time intervals are called timing events.
* The two key methods to use with JavaScript are:

setTimeout(*function, milliseconds*)  
Executes a function, after waiting a specified number of milliseconds.

setInterval(*function, milliseconds*)  
Same as setTimeout(), but repeats the execution of the function continuously.

* The setTimeout() and setInterval() are both methods of the HTML DOM Window object.
* The window.setTimeout() executes a function, after waiting a specified number of milliseconds. It can be written without the window prefix. It has two parameters. The first parameter is a function to be executed. The second parameter indicates the number of milliseconds before execution.
* The clearTimeout() method stops the execution of the function specified in setTimeout(). The window.clearTimeout() method can be written without the window prefix. The clearTimeout() method uses the variable returned from setTimeout()

myVar = setTimeout(*function*,*milliseconds*);  
clearTimeout(myVar);

* If the function has not already been executed, you can stop the execution by calling the clearTimeout() method.
* The setInterval() method repeats a given function at every given time-interval. The window.setInterval() method can be written without the window prefix. The first parameter is the function to be executed. The second parameter indicates the length of the time-interval between each execution. window.setInterval(*function*,*milliseconds*);
* The clearInterval() method stops the executions of the function specified in the setInterval() method. The window.clearInterval() method can be written without the window prefix. The clearInterval() method uses the variable returned from setInterval():

window.clearInterval(timerVariable)

**9.** What is JS Cookies.

* Cookies let you store user information in web pages.
* Cookies are data, stored in small text files, on your computer.
* When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.
* Cookies were invented to solve the problem "how to remember information about the user":
  1. When a user visits a web page, his/her name can be stored in a cookie.
  2. Next time the user visits the page, the cookie "remembers" his/her name.
* Cookies are saved in name-value pairs like: username = John Doe.
* When a browser requests a web page from a server, cookies belonging to the page are added to the request. This way the server gets the necessary data to "remember" information about users.
* JavaScript can create, read, and delete cookies with the document.cookie property. document.cookie = "username=John Doe";
* You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed

document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";

* With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.
* document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
* With JavaScript, cookies can be read like this:let x =document.cookie;
* document.cookie will return all cookies in one string much like: cookie1=value; cookie2=value; cookie3=value;
* With JavaScript, you can change a cookie the same way as you create it: The old cookie is overwritten.

document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

* Deleting a cookie is very simple. You don't have to specify a cookie value when you delete a cookie. Just set the expires parameter to a past date: You should define the cookie path to ensure that you delete the right cookie. Some browsers will not let you delete a cookie if you don't specify the path.

document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";

* Not finished read in w3s

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

* Different browsers can use the same name
* 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