JavaScript

JavaScript Window - The Browser Object Model: -

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

The 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.

Global variables are properties of the window object and 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");
```

```
document.getElementById("header"); // both are same
```

Window Size: -

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.

Example: -
```
let w = window.innerWidth; //1536
let h = window.innerHeight; // 185
```

Other Window 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

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

```
screen.width // 1536
```

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

```
screen.height // 864
```

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

screen.availWidth // 1536

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

screen.availHeight // 816

The **screen.colorDepth** property returns the number of bits used to display one color.

Screen.colorDepth // 24

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

screen.pixelDepth // 24

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.href** property returns the URL of the current page.

window.location.href // https://www.geeksforgeeks.org

The **window.location.hostname** property returns the name of the internet host (of the current page).

Window.location.hostname // www.geeksforgeeks.org

The **window.location.pathname** property returns the pathname of the current page.

window.location.pathname // "/"

The **window.location.protocol** property returns the web protocol of the page.

window.location.protocol // https:

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

window.location.port // “”

If the port number is default (80 for http and 443 for https), most browsers will display 0 or nothing.

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

```
function newDoc() {
  window.location.assign("https://www.w3schools.com")
}
```

You can click a button and call this function.

JavaScript Window History: -

The window.history object contains the browsers history.

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.

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.

JavaScript Window Navigator: -

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

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

navigator.cookieEnabled // true

The **appName** property returns the application name of the browser.

navigator.appName // Netscape

This property is removed (deprecated) in the latest web standard.

The **appCodeName** property returns the application code name of the browser.

navigator.appCodeName // Mozilla

This property is removed (deprecated) in the latest web standard.

The product property returns the product name of the browser engine.

This property is removed (deprecated) in the latest web standard. Most browsers returns Gecko as product.

The **appVersion** property returns version information about the browser.

navigator.appVersion // 5.0 (Windows)

The **userAgent** property returns the user-agent header sent by the browser to the server.

navigator.userAgent // Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101 Firefox/135.0

The information from the navigator object can often be misleading.

The navigator object should not be used to detect browser versions because

Different browsers can use the same name, and the navigator data can be changed by the browser owner.

Some browsers misidentify themselves to bypass site tests and also cannot report new operating systems, released later than the browser.

The **platform** property returns the browser platform (operating system) navigator.platform // Win32

The **language** property returns the browser's language.

navigator.language // en-US

The **onLine** property returns true if the browser is online.

navigator.onLine // true

The **javaEnabled()** method returns true if Java is enabled

navigator.javaEnabled() // false

JavaScript Popup Boxes: -

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

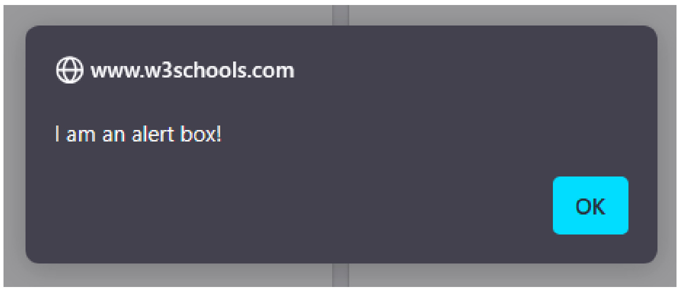
Alert Box: -

It is used to give information. When an alert box pops up, the user will have to click "OK" to proceed.

window.alert("sometext");

Also, you can omit window.

alert("I am an alert box!");


www.w3schools.com

I am an alert box!

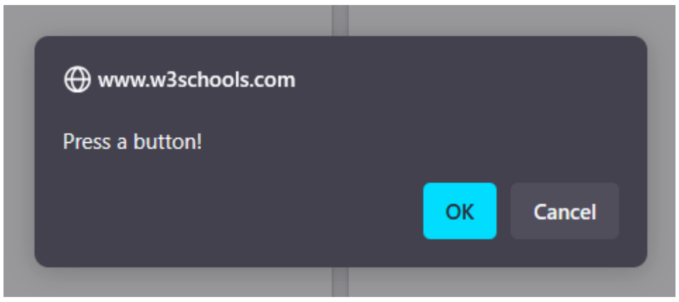
Confirm Box: -

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.

window.confirm("sometext");


www.w3schools.com

Press a button!

```
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
```

Prompt Box: -

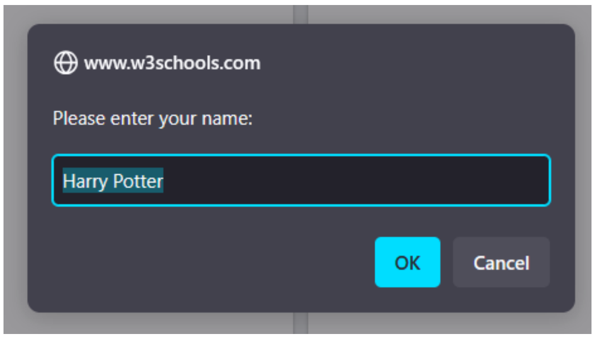
A prompt box is often used if you want the user to input a value before entering a page.

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.

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

```
let person = prompt("Please enter your name", "Harry Potter");
```



www.w3schools.com

Please enter your name:

Harry Potter

Line Breaks: -

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

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

JavaScript Timing Events: -

JavaScript can be executed in time-intervals. This is called timing events.

The setTimeout() Method: -

```
window.setTimeout(function, milliseconds);
```

The first parameter is a function to be executed.

The second parameter indicates the number of milliseconds before execution.

```
<button onclick="myVar = setTimeout(myFunction, 3000)">Try it</button>

<script>

function myFunction() {

  alert('Hello');
```

}

</script> // Click button. Wait 3 seconds, and the page will alert "Hello"

The **clearTimeout()** method stops the execution of the function specified in setTimeout().

window.clearTimeout(timeoutVariable)

If the function has not already been executed, you can stop the execution by calling the clearTimeout() method.

<button onclick="myVar = setTimeout(myFunction, 3000)">Try it</button>

<button onclick="clearTimeout(myVar)">Stop it</button>

You must click "Stop" before the 3 seconds are up.

The setInterval() Method: -

The setInterval() method repeats a given function at every given time-interval.

window.setInterval(function, milliseconds);

The second parameter indicates the length of the time-interval between each execution.

This example executes a function called "myTimer" once every second (like a digital watch).

```
setInterval(myTimer, 1000);

function myTimer() {

  const d = new Date();

  document.getElementById("demo").innerHTML = d.toLocaleTimeString();

}
```

The **clearInterval()** method stops the executions of the function specified in the setInterval() method.

let myVar = setInterval(function, milliseconds);

clearInterval(myVar);

JavaScript Cookies: -

Cookies are data, stored in small text files, on your computer.

When a user visits a web page, his/her name can be stored in a cookie. 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.

Create a Cookie with JavaScript: -

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, 27 Feb 2025 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=/";
```

Read a Cookie with JavaScript: -

```
let x = document.cookie;
```

document.cookie will return all cookies in one string much like: cookie1=value; cookie2=value; cookie3=value;

Change a Cookie with JavaScript: -

you can change a cookie the same way as you create it. The old cookie is overwritten.

Delete a Cookie with JavaScript: -

Just set the expires parameter to a past date.

The Cookie String: -

The document.cookie property looks like a normal text string. But it is not.

If you want to find the value of one specified cookie, you must write a JavaScript function that searches for the cookie value in the cookie string.

JavaScript Cookie Example: -

In the example to follow, we will create a cookie that stores the name of a visitor.

The first time a visitor arrives to the web page, he/she will be asked to fill in his/her name. The name is then stored in a cookie.

The next time the visitor arrives at the same page, he/she will get a welcome message.

For the example we will create 3 JavaScript functions:

A function to set a cookie value.

A function to get a cookie value.

A function to check a cookie value.

First, we create a function that stores the name of the visitor in a cookie variable.

```
function setCookie(cname, cvalue, exdays) {
  const d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  let expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```

The parameters of the function above are the name of the cookie (cname), the value of the cookie (cvalue), and the number of days until the cookie should expire (exdays).

Then, we create a function that returns the value of a specified cookie.

```
function getCookie(cname) {
  let name = cname + "=";
  let decodedCookie = decodeURIComponent(document.cookie);
  let ca = decodedCookie.split(';');
  for(let i = 0; i <ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

Take the cookiename as parameter (cname).

Create a variable (name) with the text to search for (cname + "=").

Decode the cookie string, to handle cookies with special characters, e.g. '$'

Split document.cookie on semicolons into an array called ca (ca = decodedCookie.split(';')).

Loop through the ca array (i = 0; i < ca.length; i++), and read out each value c = ca[i]).

If the cookie is found (c.indexOf(name) == 0), return the value of the cookie (c.substring(name.length, c.length).

If the cookie is not found, return "".

we create the function that checks if a cookie is set.

If the cookie is set it will display a greeting.

If the cookie is not set, it will display a prompt box, asking for the name of the user, and stores the username cookie for 365 days, by calling the setCookie function.

```
function checkCookie() {
  let username = getCookie("username");
  if (username != "") {
   alert("Welcome again " + username);
  } else {
    username = prompt("Please enter your name:", "");
    if (username != "" && username != null) {
      setCookie("username", username, 365);
    }
  }
}
```