JavaScript Browser Objects

JavaScript Browser Objects:

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.
- Global functions are methods of the window object.

The open() method opens a new browser window, or a new tab, depending on your browser settings and the parameter values.

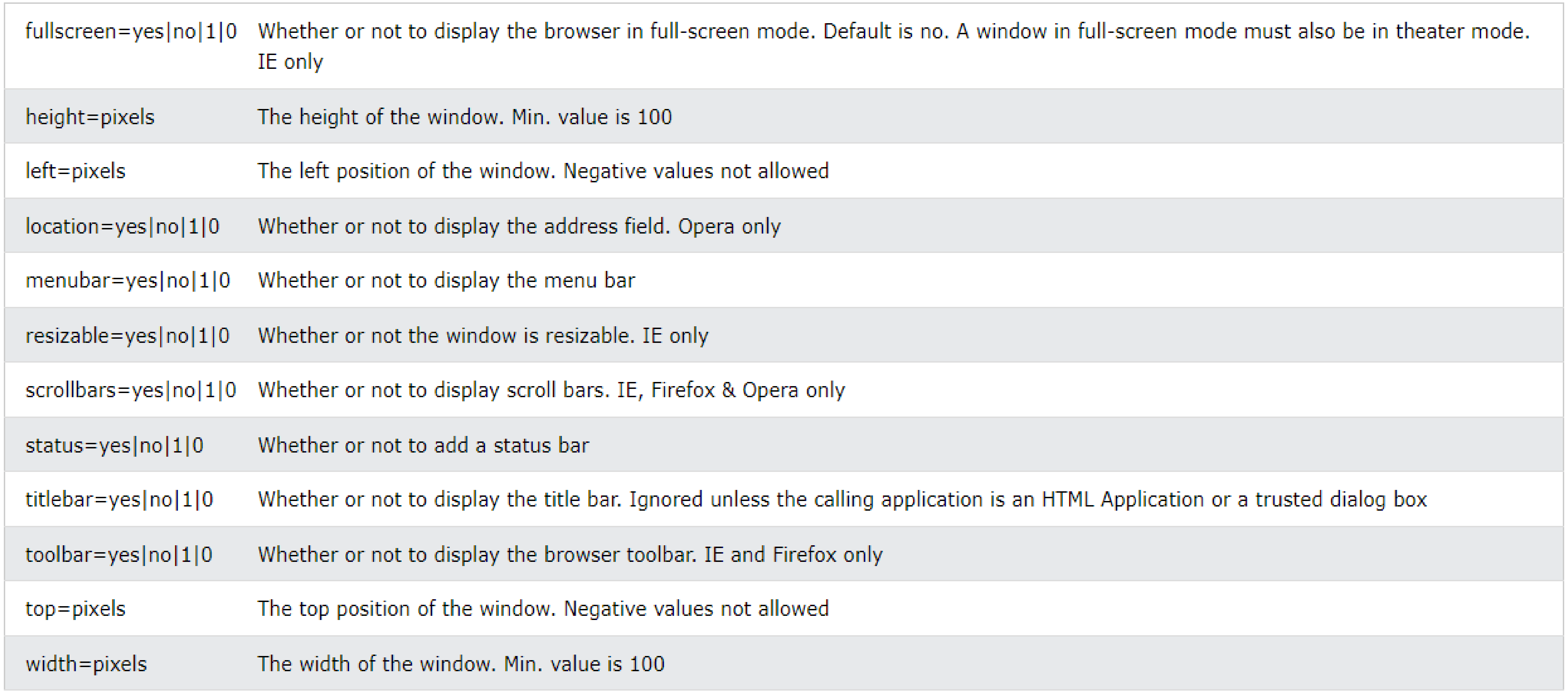
Syntax
window.open(*URL, name, specs, replace*)

Parameters

Parameter	Description
URL	Optional. The URL of the page to open. If no URL is specified, a new blank window/tab is opened
name	Optional. The target attribute or the name of the window. The following values are supported:

Value	Description
_blank	URL is loaded into a new window, or tab. This is the default
_parent	URL is loaded into the parent frame
_self	URL replaces the current page
_top	URL replaces any framesets that may be loaded
name	The name of the window (does not specify the title of the window)

specs	Optional. A comma-separated list of items, no whitespaces. The following values are supported:



fullscreen=yes|no|1|0	Whether or not to display the browser in full-screen mode. Default is no. A window in full-screen mode must also be in theater mode. IE only
height=pixels	The height of the window. Min. value is 100
left=pixels	The left position of the window. Negative values not allowed
location=yes|no|1|0	Whether or not to display the address field. Opera only
menubar=yes|no|1|0	Whether or not to display the menu bar
resizable=yes|no|1|0	Whether or not the window is resizable. IE only
scrollbars=yes|no|1|0	Whether or not to display scroll bars. IE, Firefox & Opera only
status=yes|no|1|0	Whether or not to add a status bar
titlebar=yes|no|1|0	Whether or not to display the title bar. Ignored unless the calling application is an HTML Application or a trusted dialog box
toolbar=yes|no|1|0	Whether or not to display the browser toolbar. IE and Firefox only
top=pixels	The top position of the window. Negative values not allowed
width=pixels	The width of the window. Min. value is 100

replace	Deprecated

1. Open a new window when clicking on button:

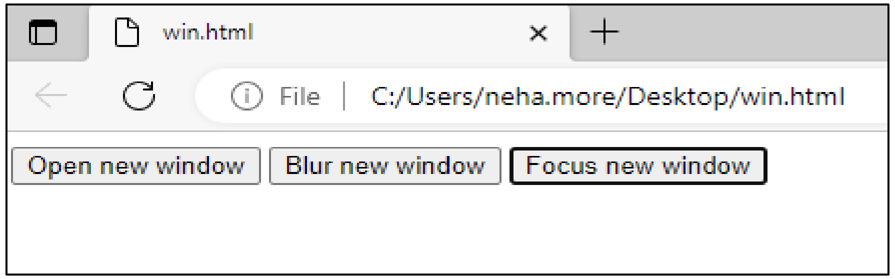
```
<script>
function openWin()
{
  window.open("https://www.w3schools.com");
}
</script>
</head>
<body>
<form>
  <input type="button" value="Open Window" onclick="openWin()">
</form>
```

Open a new window and control its appearance

```
<html>
<head>
<script>
function openWin() {
  window.open("https://www.w3schools.com","_blank","toolbar=yes, status=no, menubar=yes, scrollbars=yes,
resizable=no, width=400, height=400");
}
</script>
</head>
<body>
<form>
  <input type="button" value="Open Window" onclick="openWin()">
</form>
</body>
</html>
```

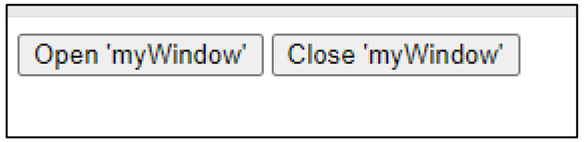
Blurr and focus a new window:

```
<html>
<head>
<script>
var myWindow;
function openWin() {
  myWindow = window.open("", "", "width=400, height=200");
  }
function blurWin() {
  myWindow.blur();
}
function focusWin() {
  myWindow.focus();
}
</script>
</head>
<body>

<input type="button" value="Open new window" onclick="openWin()">
<input type="button" value="Blur new window" onclick="blurWin()">
<input type="button" value="Focus new window" onclick="focusWin()">

</body>
</html>
```



Close the new window:

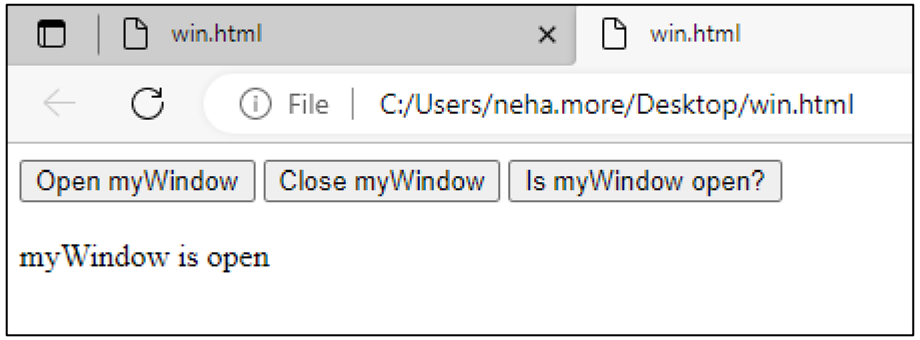
```
<html>
<head>
<script>
var myWindow;
function openWin() {
  myWindow = window.open("", "", "width=400, height=200");
}

function closeWin() {
  myWindow.close();
}
</script>
</head>
<body>

<input type="button" value="Open 'myWindow'" onclick="openWin()" />
<input type="button" value="Close 'myWindow'" onclick="closeWin()" />

</body>
</html>
```



Check whether new window has closed or not

```
<html>
<head>
<script>
var myWindow;
function openWin() {
  myWindow = window.open("", "", "width=400 ,height=200");
}
function closeWin() {
  if (myWindow) {
    myWindow.close();
  }
}
function checkWin() {
  msg = ""
  if (!myWindow) {
    msg = "was never opened";
  } else {
    if (myWindow.closed) {
      msg = "is closed";
    } else {
      msg = "is open";
    }
    }
  document.getElementById("msg").innerHTML =
  "myWindow " + msg;
}
```

```
</script>
</head>
<body>
<button onclick="openWin()">Open myWindow</button>
<button onclick="closeWin()">Close myWindow</button>
<button onclick="checkWin()">Is myWindow open?</button>
<br><br>
<div id="msg"></div>
</body>
</html>
```



```
myWindow is open
```

Move a new window to the specified position

```
<html>
<head>
<script>
var myWindow;
function openWin() {
  myWindow=window.open("", "", "width=400, height=200");
}

function moveWin() {
  myWindow.moveTo(300, 0);
  myWindow.focus();
}
</script>
</head>
<body>

<input type="button" value="Open myWindow" onclick="openWin()" />
<br><br>
<input type="button" value="Move myWindow" onclick="moveWin()" />

</body>
</html>
```

window.moveTo(x, y)

Parameter	Description
x	Required. A positive or negative number. The horizontal coordinate to move to.
y	Required. A positive or negative number. The vertical coordinate to move to.

Print the current page

```
<html>
<head>
<script>
function printPage() {
  window.print();
}
</script>
</head>
<body>

<input type="button" value="Print this page"
onclick="printPage()" />

</body>
</html>
```

Resize the window by the specified pixels

```
<html>
<head>
<script>
var w;
function openwindow() {
  w = window.open('','', 'width=100,height=100');
  w.focus();
}

function myFunction() {
  w.resizeBy(50, 50);
  w.focus();
}

</script>
</head>
<body>

<button onclick="openwindow()">Create window</button>
<button onclick="myFunction()">Resize window</button>

</body>
</html>
```

The resizeBy() method resizes a window by a specified amount relative to its current size. .

resizeBy(width, height)

Parameter	Description
width	Required. A positive or a negative number. The number of pixels to resize the width by.
height	Required. A positive or a negative number. The number of pixels to resize the height by.

Resize a window to a specified size

```
<html>
<head>
<script>
var w;
function openwindow() {
  w = window.open('','', 'width=100,height=100');
  w.focus();
}

function myFunction() {
  w.resizeTo(500, 500);
  w.focus();
}

</script>
</head>
<body>

<button onclick="openwindow()">Create window</button>
<button onclick="myFunction()">Resize window</button>

</body>
</html>
```

The resizeTo() method resizes a window to a new width and height.

```
window.resizeTo(width, height)
```

Parameter	Description
width	Required. The new window width, in pixels
height	Required. The new window height, in pixels

Scroll the content by the specified number of pixcel

Window scrollBy()

The scrollBy() method scrolls the document by the specified number of pixels.

Syntax:

```
window.scrollBy(x, y)

Or

scrollBy(x, y)
```

Parameter	Description
x	Required. Number of pixels to scroll (horizontally). Positive values scroll to the right, negative values to the left.
y	Required. Number ofpixels to scroll (vertically). Positive values scroll down, negative values scroll up.

Scroll the document 100px horizontally:

```
<html>
<style>
body {width: 5000px}
button {position:fixed}
</style>
<body>
<h1>The Window Object</h1>
<h2>The scrollBy() Method</h2>
<p>Click to scroll the document.</p>
<p>Look at the horizontal scrollbar to see the effect.</p>
<button onclick="scrollWin()" style="position:fixed">Scroll 100px horizontally!</button>
<br><br>
<script>
function scrollWin() {
  window.scrollBy(100, 0);
}
</script>
</body>
</html>
```

Scroll the document 100px vertically:

```
<html>
<body>
<h1>The Window Object</h1>
<h2>The scrollBy() Method</h2>
<p>Click to scroll the document.</p>
<button onclick="scrollWin()" style="position:fixed">Scroll 100px
vertically!</button>
<br><br>

<h3>Some line breaks to enable scrolling:</h3>
<br>
<br>
<br>
<p>When I find myself in times of trouble</p>
<br>
<br>
<br>
<p>Mother Mary comes to me</p>
<br>
<br>
<br>
<p>Speaking words of wisdom, let it be</p>
<br>
<br>
<br>
<p>And in my hour of darkness she is standing right in front of me</p>
<br>
<br>
<br>
<p>Speaking words of wisdom, let it be</p>
<br>
<br>
```

```
<br>
<p>ROCK AND ROLL</p>
<br>
<br>
<br>
<p>SCROLL SCROLL SCROLL</p>
<br>
<br>
<br>
<p>ROCK AND ROLL</p>
<br>
<br>
<br>
<p>SCROLL SCROLL SCROLL</p>
<br>
<br>
<br>
<p>ROCK AND ROLL</p>
<br>
<br>
<br>
<p>ROCK AND ROLL</p>
<br>
<br>
<br>
<script>
function scrollWin() {
  window.scrollBy(0, 100);
}
</script>
</body>
</html>
```

The scrollTo()

This method scrolls the document to specified coordinates.

Syntax
```
window.scrollTo(x, y)
or
just:scrollTo(x, y)
```

Parameter	Description
x	Required. The coordinate to scroll to (horizontally), in pixels.
y	Required. The coordinate to scroll to (vertically), in pixels.

Scroll the document to the horizontal position 200:

```
<style>
body {
  width: 5000px;
}
</style>

<body>
<h1>The Window Object</h1>
<h2>The scrollTo() Method</h2>

<p>Click to scroll the document.</p>

<button onclick="scrollWin()" style="position:fixed">Scroll to 200
horizontally!</button><br><br>

<script>
function scrollWin() {
  window.scrollTo(200, 0);
}
</script>
```

Scroll the document to the vertical position 500:

```
<style>
body {
  height: 5000px;
}
</style>

<body>
<h1>The Window Object</h1>
<h2>The scrollTo() Method</h2>

<p>Click to scroll the document.</p>

<button onclick="scrollWin()" style="position:fixed">Scroll to 500
vertically!</button><br><br>

<script>
function scrollWin() {
  window.scrollTo(0, 500);
}
</script>
```

Window scrollX

- The scrollX property returns the pixels a document has scrolled from the upper left corner of the window.
- The scrollX property is read-only.

Syntax
```
window.scroll
```
or
just:
```
scrollX
```

Window scrollY

- The scrollY property returns the pixels a document has scrolled from the upper left corner of the window.
- The scrollY property is read-only.

Syntax
```
window.scrollY
```
or just:
```
scrollY
```

Window scrollX and scrollY

```
<html>
<head>
<style>
div {
  background-color: lightblue;
  height: 2000px;  width: 2000px;
}
</style>
</head>
<body>
<h1>The Window Object</h1>
<h2>The scrollX and scrollY Properties</h2>
<p>Click the button to scroll the document window 100px horizontally and vertically.</p>
<button onclick="myFunction()" style="position:fixed;">Click me to scroll</button><br><br>
<div></div>
<script>
function myFunction() {
  window.scrollBy(100, 100);
  alert("pageXOffset: " + window.scrollX + ", scrollY: " + window.scrollY);
}
</script>
</body>
</html>
```

Screen Object

The screen object contains information about the visitor's screen.

Screen Object Properties

Property	Description
availHeight	Returns the height of the screen (excluding the Windows Taskbar)
availWidth	Returns the width of the screen (excluding the Windows Taskbar)
colorDepth	Returns the bit depth of the color palette for displaying images
height	Returns the total height of the screen
pixelDepth	Returns the color resolution (in bits per pixel) of the screen
width	Returns the total width of the screen

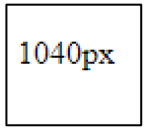
availHeight

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

```
screen.availHeight
```

Example:

```
<p id="demo"></p>
<script>
height = screen.availHeight;
document.getElementById("demo").innerHTML = height + "px";
</script>
```



```
1040px
```

availWidth

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

Example:

```
<p id="demo"></p>
<script>
let width = screen.availWidth;
document.getElementById("demo").innerHTML = width + "px";
</script>
```

```
1920px
```

Window screen.height

- The height property returns the total height of the user's screen.
- The height property returns the height in pixels.
- The height property is read only.

```
<p id="demo"></p>
<script>
let height = screen.height;
document.getElementById("demo").innerHTML =  height + "px";
</script>
```

Window screen.width

- The width property returns the total width of the user's screen.
- The width property returns width in pixels.
- The width property is read-only.

```
<p id="demo"></p>
<script>
let width = screen.width;
document.getElementById("demo").innerHTML =  width + "px";
</script>
```

Window screen.colorDepth

- The colorDepth property returns the screen's color depth.
- The colorDepth property returns the depth in bits per pixel.
- The colorDepth property is read-only.

```
<p id="demo"></p>
<script>
let depth = screen.colorDepth;
document.getElementById("demo").innerHTML =  depth + " bits per pixel";
</script>
```

```
24 bits per pixel
```

Window screen.pixelDepth

- The pixelDepth property returns the screen's color depth.
- The pixelDepth property returns the color depth in bits per pixel.
- The pixelDepth property is read-only.

```
<p id="demo"></p>
<script>
let depth = screen.pixelDepth;
document.getElementById("demo").innerHTML = depth + " bits per pixel";
</script>
```

```
24 bits per pixel
```

Location Object

- The location object contains information about the current URL.
- The location object is a property of the window object.
- The location object is accessed with:

window.location or just location

Location Object Properties

Property	Description
hash	Sets or returns the anchor part (#) of a URL
host	Sets or returns the hostname and port number of a URL
hostname	Sets or returns the hostname of a URL
href	Sets or returns the entire URL
origin	Returns the protocol, hostname and port number of a URL
pathname	Sets or returns the path name of a URL
port	Sets or returns the port number of a URL
protocol	Sets or returns the protocol of a URL
search	Sets or returns the querystring part of a URL

Window location.hash

- The location.hash property sets or returns the anchor part of a URL, including the hash sign (#).
- When location.hash is used to set the anchor part, do not include the hash sign (#).

```
<html>
<body>
<h1>The Window Location Object</h1>
<h2>The hash Property</h2>
<p><a id="w3s" href="/js/js_es6.asp#mark_array_from">JavaScript
2015 Array.from()</a><p>
<p id="demo"></p>
<script>
let url = document.getElementById("w3s");
document.getElementById("demo").innerHTML = "The anchor portion
of the URL is: " + url.hash;
</script>
</body>
</html>
```

The Window Location Object

The hash Property

JavaScript 2015 Array.from()

The anchor portion of the URL is: #mark_array_from

```
<p><a id="w3s" href="/js/js_es6.asp#mark_array_from">JavaScript
2015 Array.from()</a><p>
<p id="demo"></p>
<script>
location.hash = "mark_array_find";
document.getElementById("demo").innerHTML = "The anchor part is
now: " + location.hash;
</script>
```

JavaScript 2015 Array.from()

The anchor part is now: #mark_array_find

Window location.host

The location.host property returns the host (IP adress or domain) and port of a URL.

```
<h1>The Window Location Object</h1>
<h2>The host Property</h2>
<p id="demo"></p>
<script>
let host = location.host;
document.getElementById("demo").innerHTML = host;
</script>
```

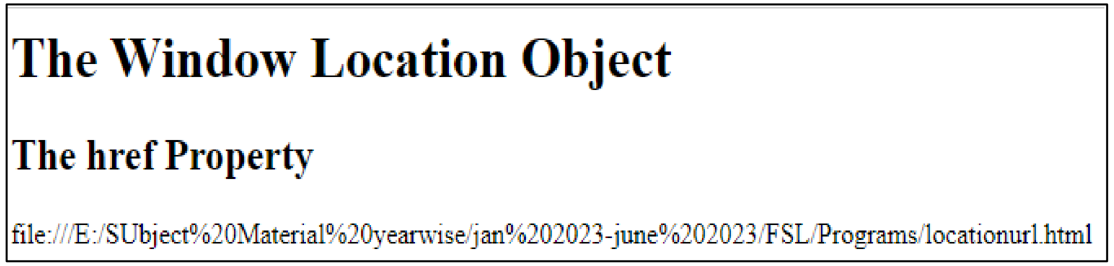
If the port number is not specified in the URL, or if it is a default port (80 for http) or (443 for https), most browsers will return an empty string.

Port 53: Domain Name System (DNS). ...
Port 80: Hypertext Transfer Protocol (HTTP). ...
Port 123: Network Time Protocol (NTP).

Window location.href

The location.href property sets or returns the entire URL of the current page.

```
<html>
<body>
<h1>The Window Location Object</h1>
<h2>The href Property</h2>
<p id="demo"></p>
<script>
let url = location.href;
document.getElementById("demo").innerHTML = url;
</script>
</body>
</html>
```



```
The Window Location Object

The href Property

file:///E:/SUbject%20Material%20yearwise/jan%202023-june%202023/FSL/Programs/locationurl.html
```

Set the URL of the current page:

```
<p>Click the button to set the href value to https://www.w3schools.com.</p>
<button onclick="myFunction()">Take me to w3schools.com</button>
<script>
function myFunction() {
  location.href = "https://www.w3schools.com";
}
</script>
```

Window location.origin

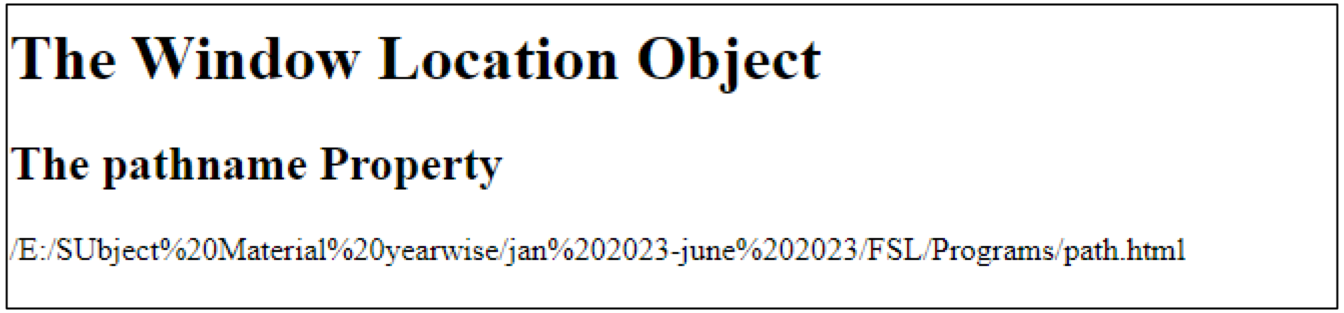
- The origin property returns the protocol, hostname and port number of a URL.
- The origin property is read-only.
- If the port number is not in the URL, or if it is a default port like 80 (Http), or 443 (https), some browsers will not display the port number.

```
<p id="demo"></p>
<script>
let origin = location.origin;
document.getElementById("demo").innerHTML = origin;
</script>
```

Window location.pathname

The pathname property sets or returns the pathname of a URL (page).

```
<html>
<body>
<h1>The Window Location Object</h1>
<h2>The pathname Property</h2>
<p id="demo"></p>
<script>
let path = location.pathname;
document.getElementById("demo").innerHTML = path;
</script>
</body>
</html>
```



The Window Location Object

The pathname Property

```
/E:/SUbject%20Material%20yearwise/jan%202023-june%202023/FSL/Programs/path.html
```

Window location.port

- The port property sets or returns the port number of a URL.
- If the port number is not specified in the URL, or if it is a default port (80 for http) or (443 for https), most browsers will return an empty string.

```
<h1>The Window Location Object</h1>
<h2>The port Property</h2>

<p id="demo"></p>

<p><b>Note: </b>If the port number is default (80 for http and 443 for https), most browsers return
nothing.</p>

<script>
let port = location.port;
document.getElementById("demo").innerHTML = "The port number of the current page is: " + port;
</script>

</body>
</html>
```

Window location.protocol

- The protocol property sets or returns the protocol of the current URL, including the colon (:).
- The protocol is a standard that specifies how data are transmitted between computers.

Parameter	Description
protocol	The protocol of the URL. •Examples:file: •ftp: •http: •https: •mailto:

```
<p id="demo"></p>
<script>
let protocol = location.protocol;
document.getElementById("demo").innerHTML = protocol;
</script>
```

Window location.reload()

- The reload() method reloads the current document.
- The reload() method does the same as the reload button in your browser.

```
<script>
location.reload();
</script>
```

Window location.search

- The search property returns the querystring part of a URL, including the question mark (?).
- The search property can also be used to set the querystring.
- The querystring part is the part of the URL after the question mark (?).
- The querystring is used for parameter passing.

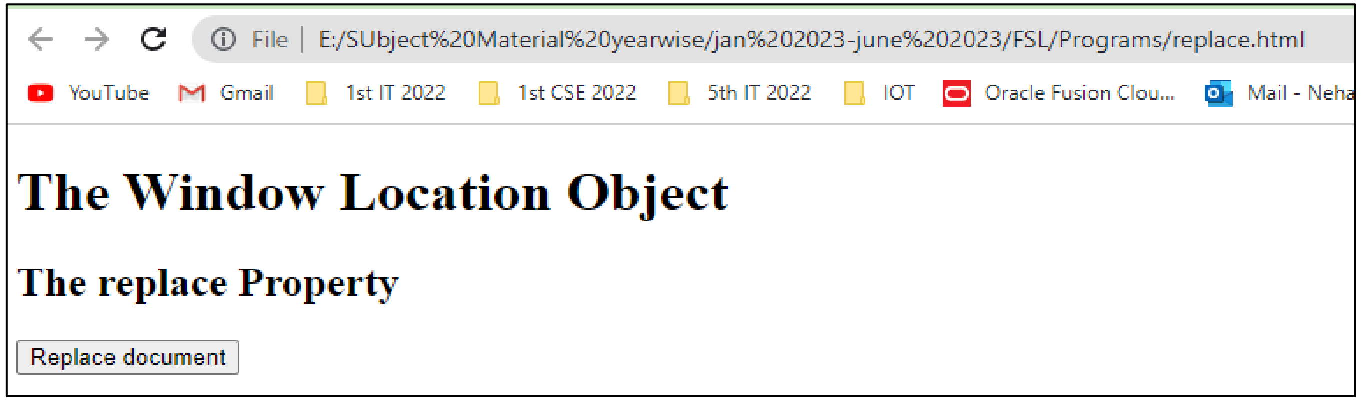
```
<p><a id="w3s" href="https://www.w3schools.com/?answer=yes">
https://www.w3schools.com/?answer=yes</a></p>
<p id="demo"></p>
<script>
let anchor = document.getElementById("w3s");
let query = anchor.search;
document.getElementById("demo").innerHTML = "The query portion of the URL is: " + query;
</script>
```

The Window Location Object
The search Property
https://www.w3schools.com/?answer=yes
The query portion of the URL is: ?answer=yes

Window location.replace()

- The replace() method replaces the current document with a new one.

```
<button onclick="myFunction()">Replace document</button>
<script>
function myFunction() {
  location.replace("https://www.w3schools.com")
}
</script>
```

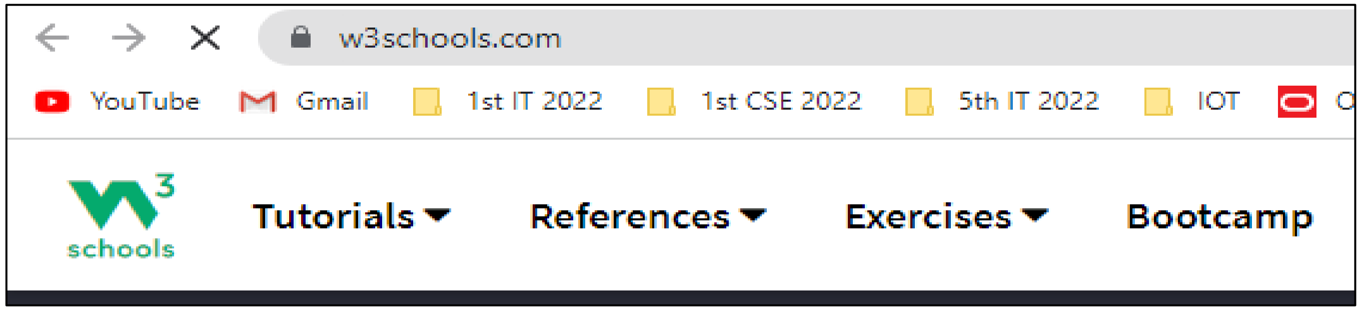


File | E:/SUbject%20Material%20yearwise/jan%202023-june%202023/FSL/Programs/replace.html

YouTube Gmail 1st IT 2022 1st CSE 2022 5th IT 2022 IOT Oracle Fusion Clou... Mail - Neha

The Window Location Object

The replace Property



w3schools.com

YouTube Gmail 1st IT 2022 1st CSE 2022 5th IT 2022 IOT

JavaScript 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() Method

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

- The window.setTimeout() method can be written without the window prefix.
- The first parameter is a function to be executed.
- The second parameter indicates the number of milliseconds before execution.

Click a button. Wait 3 seconds, and the page will alert "Hello":

```
<html>
<body>
<h2>JavaScript Timing</h2>
<p>Click "Try it". Wait 3 seconds, and the page will alert "Hello".</p>
<button onclick="setTimeout(myFunction, 3000);">Try it</button>
<script>
function myFunction() {
  alert('Hello');
}
</script>
</body>
</html>
```

Stop the Execution

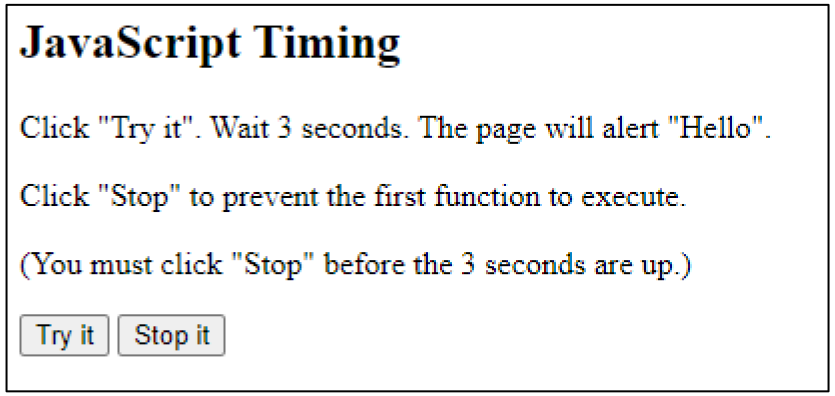
The clearTimeout() method stops the execution of the function specified in setTimeout().

```
window.clearTimeout(timeoutVariable)
```

- The window.clearTimeout() method can be written without the window prefix.
- The clearTimeout() method uses the variable returned from setTimeout():

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

```
<html>
<body>
<h2>JavaScript Timing</h2>
<p>Click "Try it". Wait 3 seconds. The page will alert "Hello".</p>
<p>Click "Stop" to prevent the first function to execute.</p>
<p>(You must click "Stop" before the 3 seconds are up.)</p>
<button onclick="myVar = setTimeout(myFunction, 3000)">Try it</button>
<button onclick="clearTimeout(myVar)">Stop it</button>
<script>
function myFunction() {
  alert("Hello");
}
</script>
</body>
</html>
```



JavaScript Timing

Click "Try it". Wait 3 seconds. The page will alert "Hello".

Click "Stop" to prevent the first function to execute.

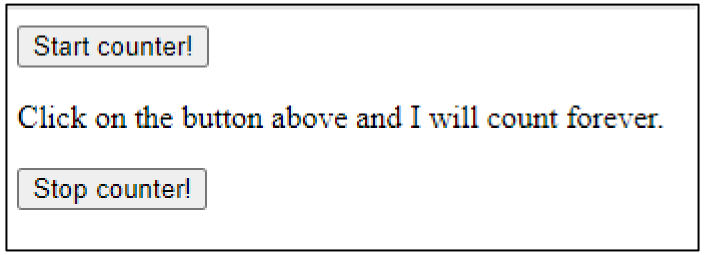
(You must click "Stop" before the 3 seconds are up.)

Timing event in an infinite loop

```
<html>
<body>
<button onClick="setInterval(myCounter, 1000)">Start counter!</button>
<p id="demo">Click on the button above and I will count forever.</p>
<script>
var c = 0;
function myCounter() {
  document.getElementById("demo").innerHTML = ++c;
}
</script>
</body>
</html>
```

Timing event in an infinite loop-with stop button

```
<html>
<body>
<button onClick="myTimer = setInterval(myCounter, 1000)">Start
counter!</button>
<p id="demo">Click on the button above and I will count forever.</p>
<button onClick="clearInterval(myTimer)">Stop counter!</button>
<script>
var c = 0;
function myCounter() {
  document.getElementById("demo").innerHTML = ++c;
}
</script>
</body>
</html>
```



Click on the button above and I will count forever.

The setInterval() Method

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

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

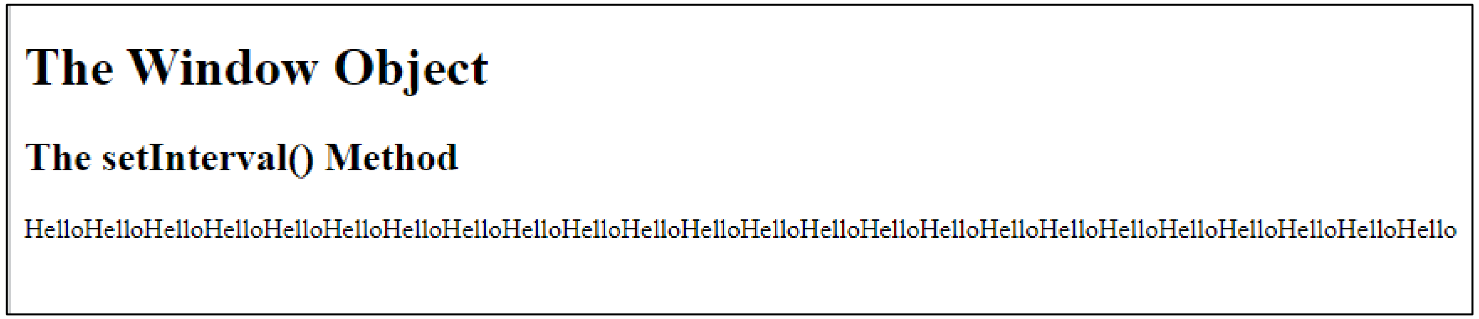
- The window.setInterval() method can be written without the window prefix.
- The first parameter is the function to be executed.
- The second parameter indicates the length of the time-interval between each execution.
- This example executes a function called "myTimer" once every second (like a digital watch).

```
<html>
<body>
<h2>JavaScript Timing</h2>
<p>A script on this page starts this clock:</p>
<p id="demo"></p>
<script>
setInterval(myTimer, 1000);
function myTimer() {
  const d = new Date();
  document.getElementById("demo").innerHTML = d.toLocaleTimeString();
}
</script>
</body>
</html>
```

toLocaleTimeString():The toLocaleTimeString() method returns the time portion of a date object as a string, using locale conventions.

Display "Hello" every second (1000 milliseconds):

```
<html>
<body>
<h1>The Window Object</h1>
<h2>The setInterval() Method</h2>
<p id="demo"></p>
<script>
setInterval(displayHello, 1000);
function displayHello() {
  document.getElementById("demo").innerHTML += "Hello";
}
</script>
</body>
</html>
```



The Window Object

The setInterval() Method

```
HelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHello
```

The Window History Object

- The history object contains the URLs visited by the user (in the browser window).
- The history object is a property of the window object.
- The JS history object contains an array of URLs visited by the user. By using the history object, you can load previous, forward, or any particular page using various methods.
- The history object is accessed with:

window.history or just history

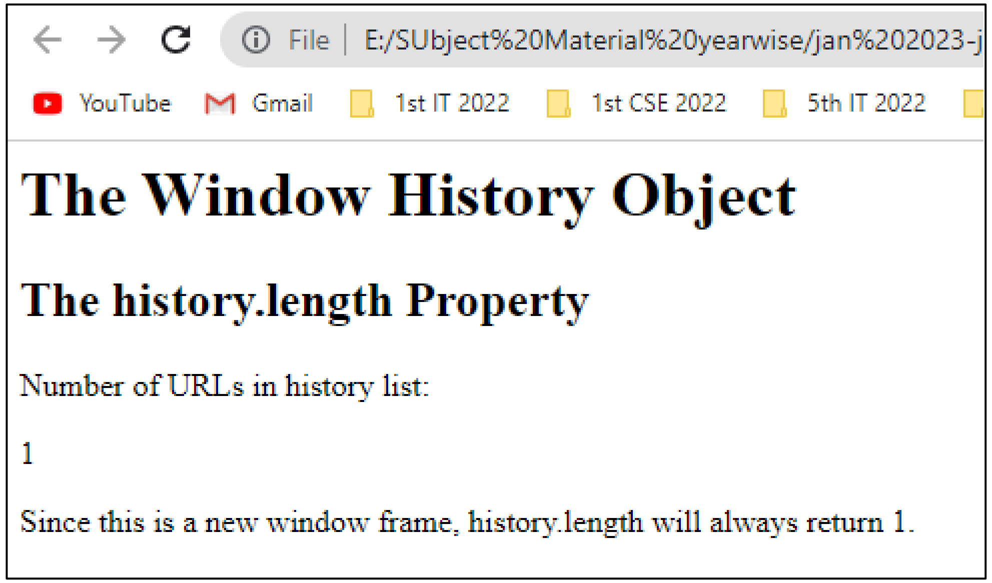
History Object Properties and Methods

Property/Method	**Description**
back()	Loads the previous URL (page) in the history list
forward()	Loads the next URL (page) in the history list
go()	Loads a specific URL (page) from the history list
length	Returns the number of URLs (pages) in the history list

Windows history.length:

Get the number of URLs in the history list:

```
<html>
<body>

<h1>The Window History Object</h1>
<h2>The history.length Property</h2>

<p>Number of URLs in history list:</p>
<p id="demo"></p>

<p>Since this is a new window frame, history.length will always
return 1.</p>

<script>
let length = history.length;
document.getElementById("demo").innerHTML = length;
</script>

</body>
</html>
```



The Window History Object

The history.length Property

Number of URLs in history list:

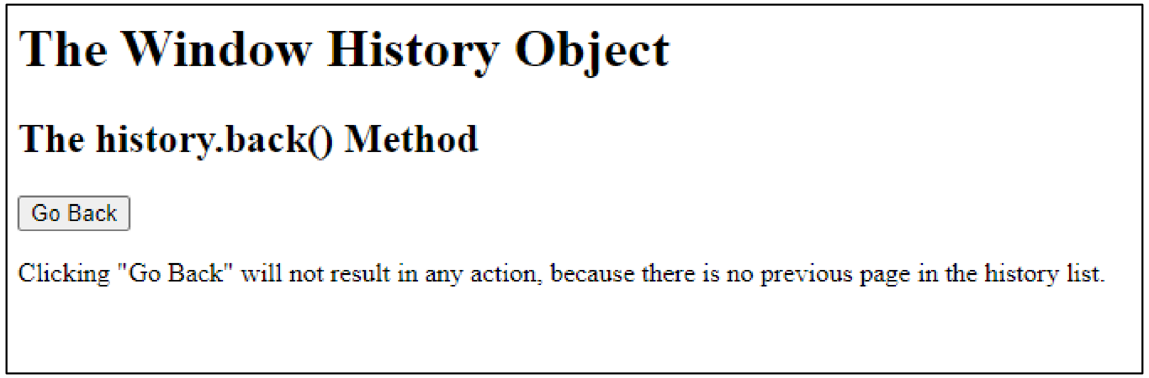
1

Since this is a new window frame, history.length will always return 1.

Window history.back()

- The history.back() method loads the previous URL (page) in the history list.
- The history.back() method only works if a previous page exists.

```
<html>
<body>

<h1>The Window History Object</h1>
<h2>The history.back() Method</h2>

<button onclick="history.back()">Go Back</button>

<p>Clicking "Go Back" will not result in any action, because
there is no previous page in the history list.</p>

</body>
</html>
```



The Window History Object

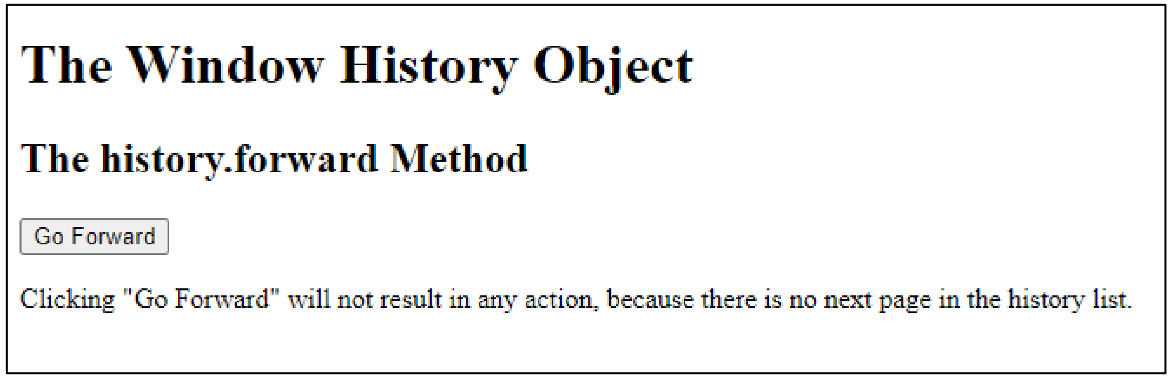
The history.back() Method

Clicking "Go Back" will not result in any action, because there is no previous page in the history list.

Window history.forward()

- The history.forward() method loads the next URL (page) in the history list.
- The history.forward() method only works if a next page exists.

```
<html>
<body>

<h1>The Window History Object</h1>
<h2>The history.forward Method</h2>

<button onclick="history.forward()">Go Forward</button>

<p>Clicking "Go Forward" will not result in any action, because
there is no next page in the history list.</p>

</body>
</html>
```



The Window History Object

The history.forward Method

Clicking "Go Forward" will not result in any action, because there is no next page in the history list.

Window history.go()

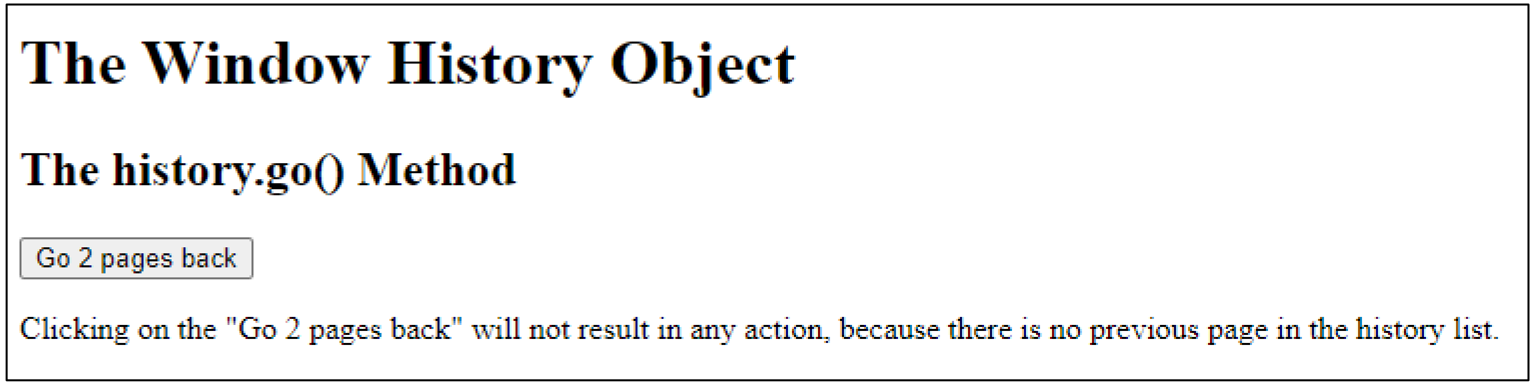
- The history.go() method loads a URL (page) from the history list.
- The history.go() method only works if the page exist in the history list.

```
<html>
<body>
<h1>The Window History Object</h1>
<h2>The history.go() Method</h2>
<button onclick="history.go(-2)">Go 2 pages back</button>
<p>Clicking on the "Go 2 pages back" will not result in any
action, because there is no previous page in the history
list.</p>
</body>
</html>
```

history.go(0) reloads the page.

history.go(-1) is the same as history.back().

history.go(1) is the same as history.forward().



The Window History Object

The history.go() Method

Clicking on the "Go 2 pages back" will not result in any action, because there is no previous page in the history list.