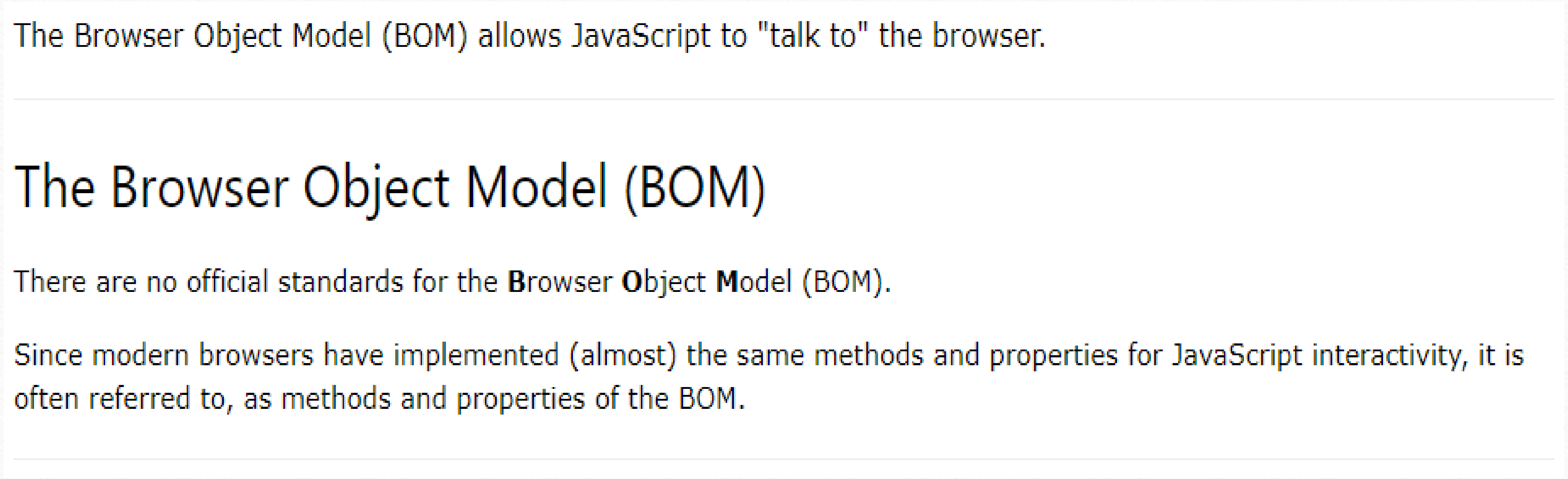
JavaScript

Presented By

Eng/Abanoub Nabil

JavaScript Window - The Browser Object Model(BOM)

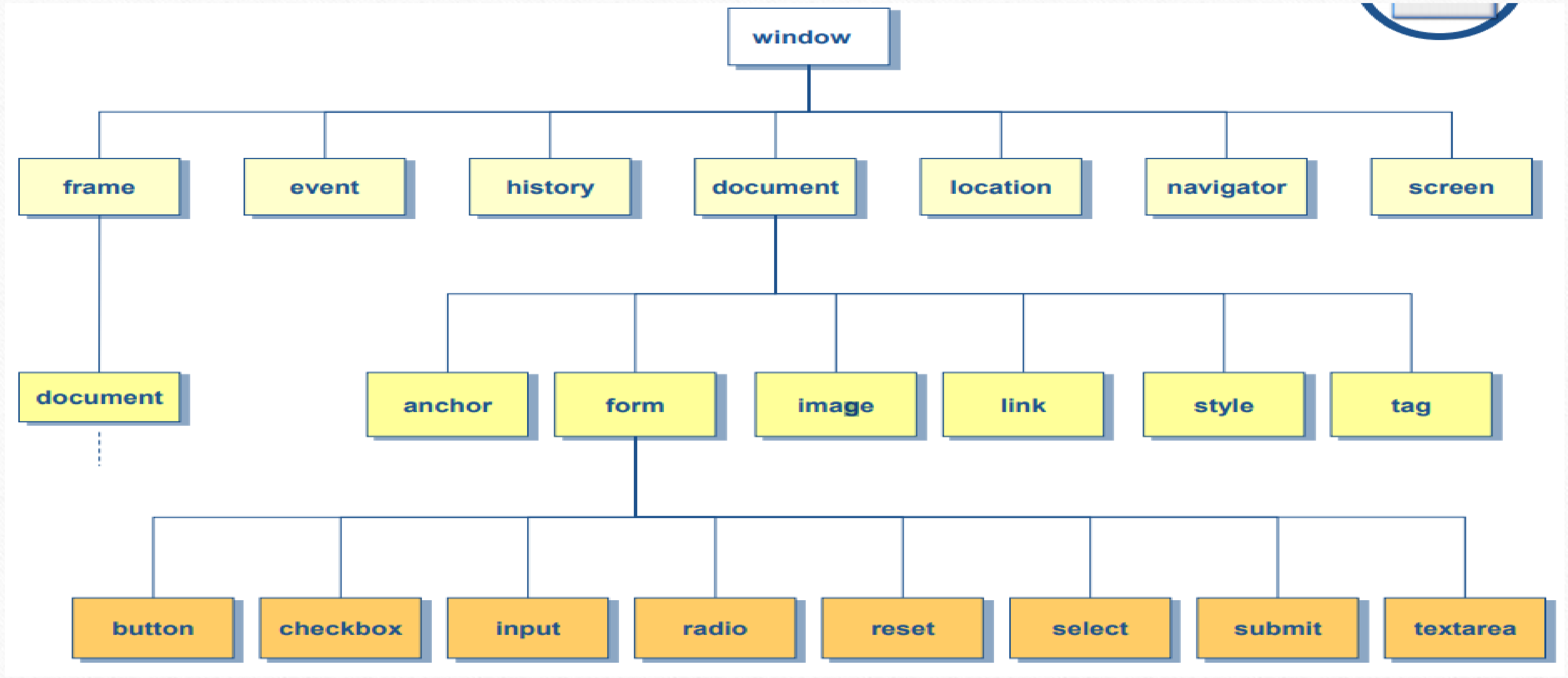

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.

The Browser Object Model (BOM)

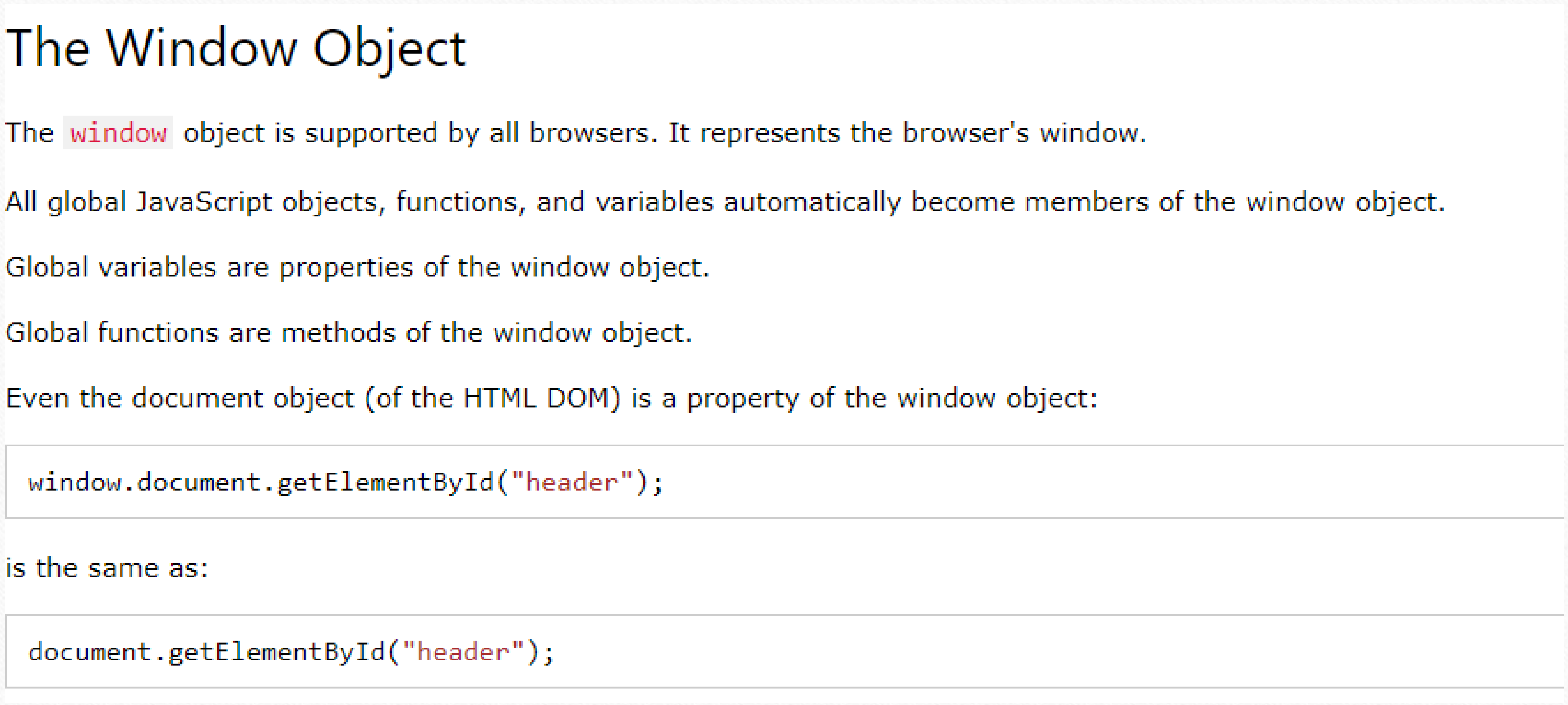
There are no official standards for the **B**rowser **O**bject **M**odel (BOM).

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.

JavaScript Window - The Browser Object Model(BOM)



- Every page has the following objects:
  - **window:** the top-level object; has properties that apply to the entire window.
  - **navigator:** has properties related to the name and version of the Navigator being used.
  - **document:** contains properties based on the content of the document, such as title, background color, links, and forms.
  - **location:** has properties based on the current URL.
  - **history:** contains properties representing URLs the client has previously requested.
  - **screen** : contains information about the visitor's screen.



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.

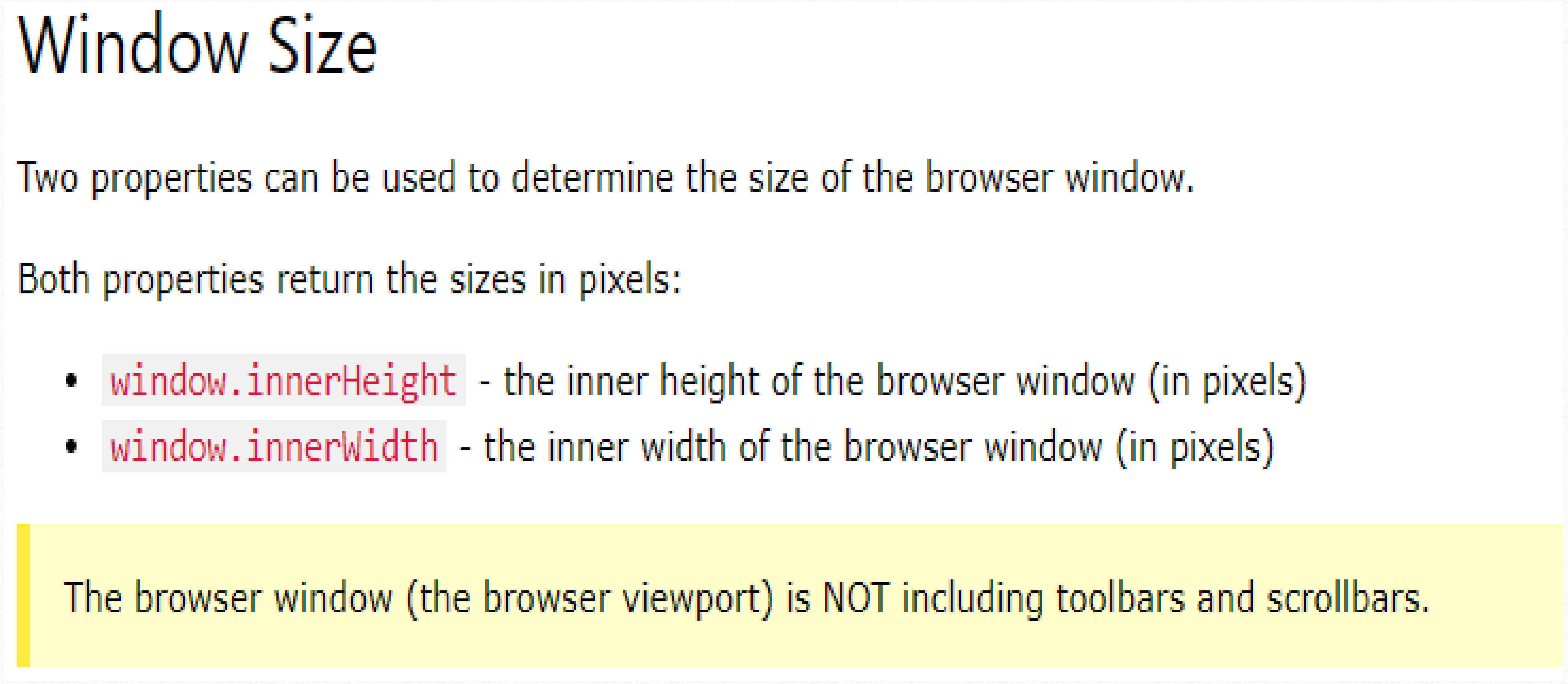
Global functions are methods of the window object.

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");
```



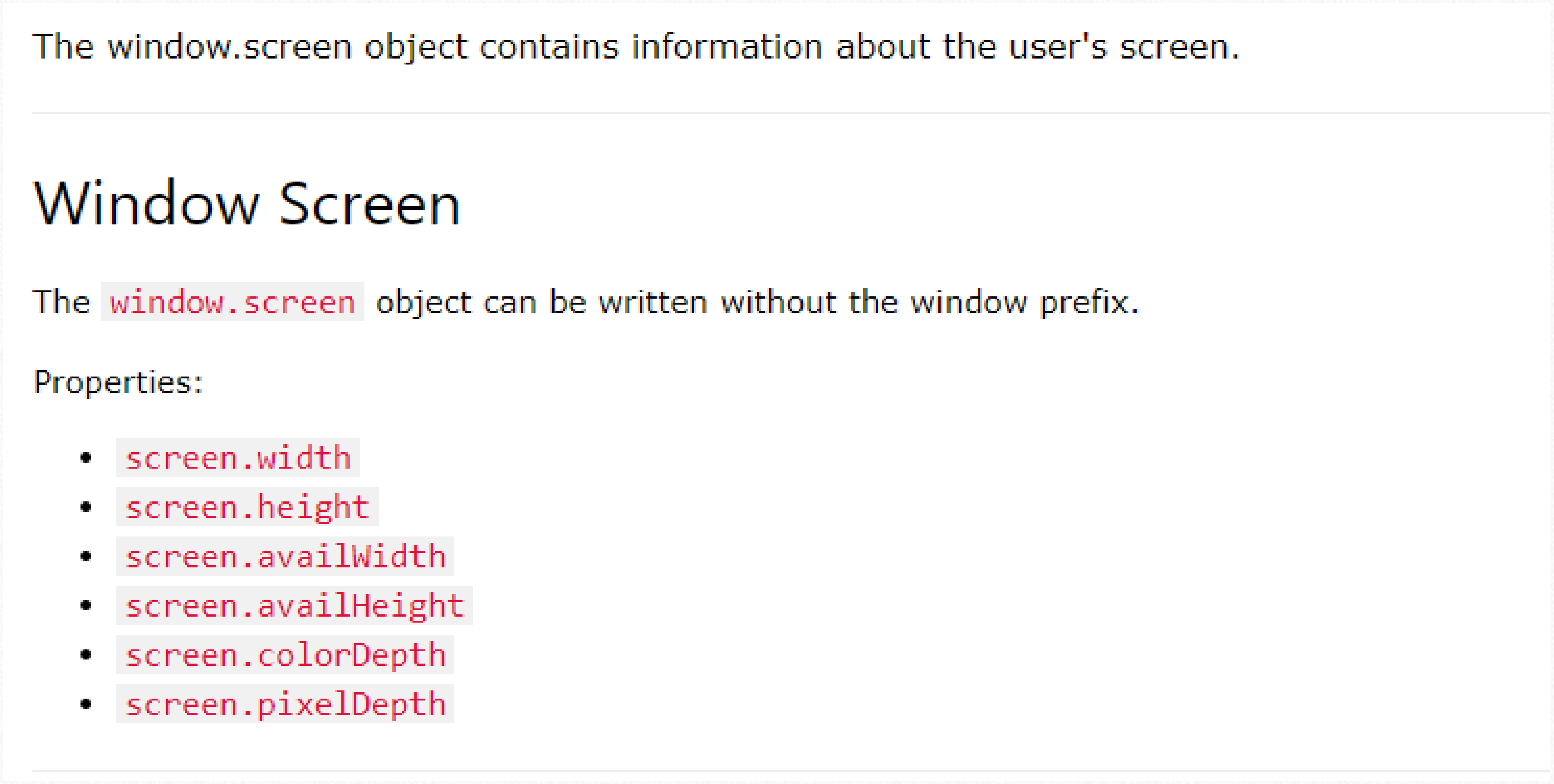
Window Size

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.



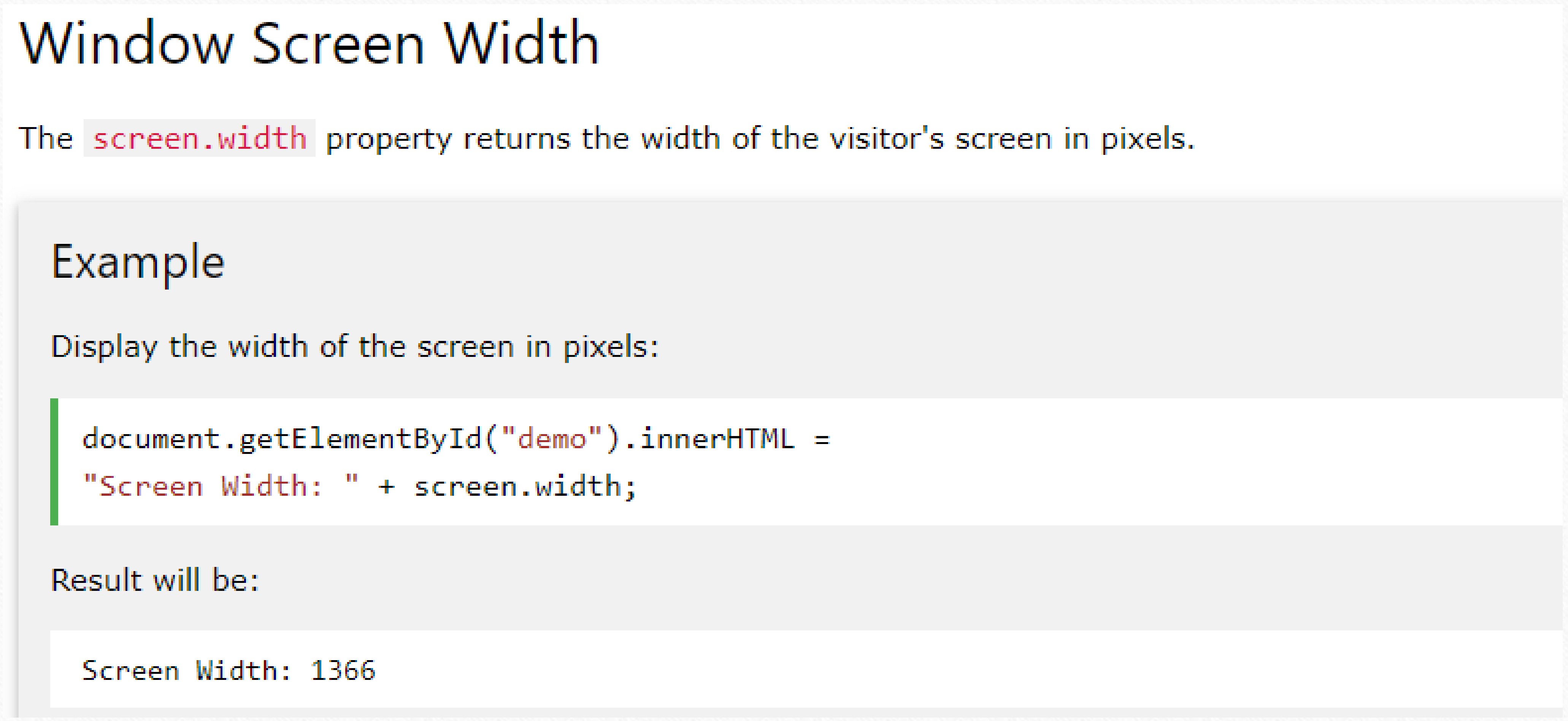
The window.screen object contains information about the user's screen.

Window Screen

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

Properties:

- `screen.width`
- `screen.height`
- `screen.availWidth`
- `screen.availHeight`
- `screen.colorDepth`
- `screen.pixelDepth`



Window Screen Width

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

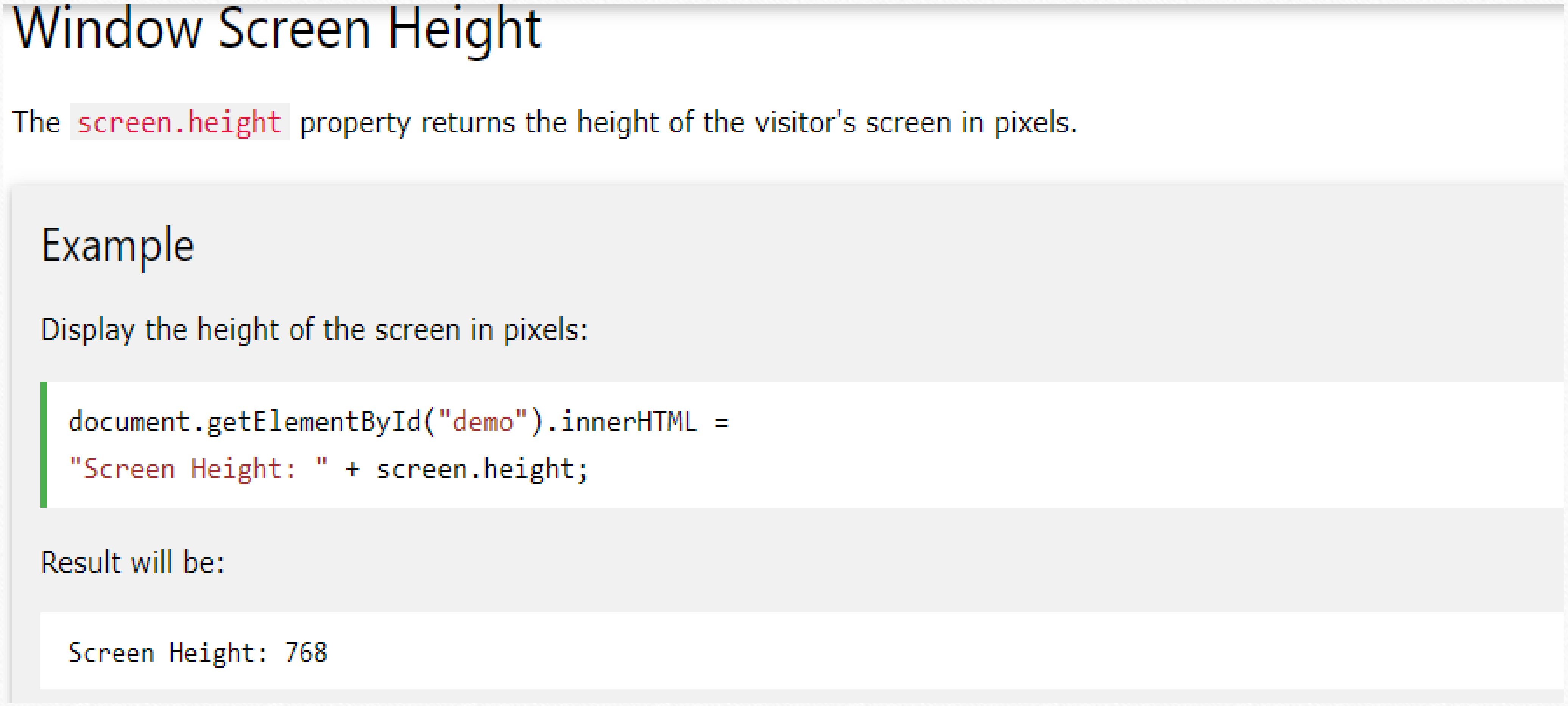
Example

Display the width of the screen in pixels:

```
document.getElementById("demo").innerHTML =
"Screen Width: " + screen.width;
```

Result will be:

```
Screen Width: 1366
```



Window Screen Height

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

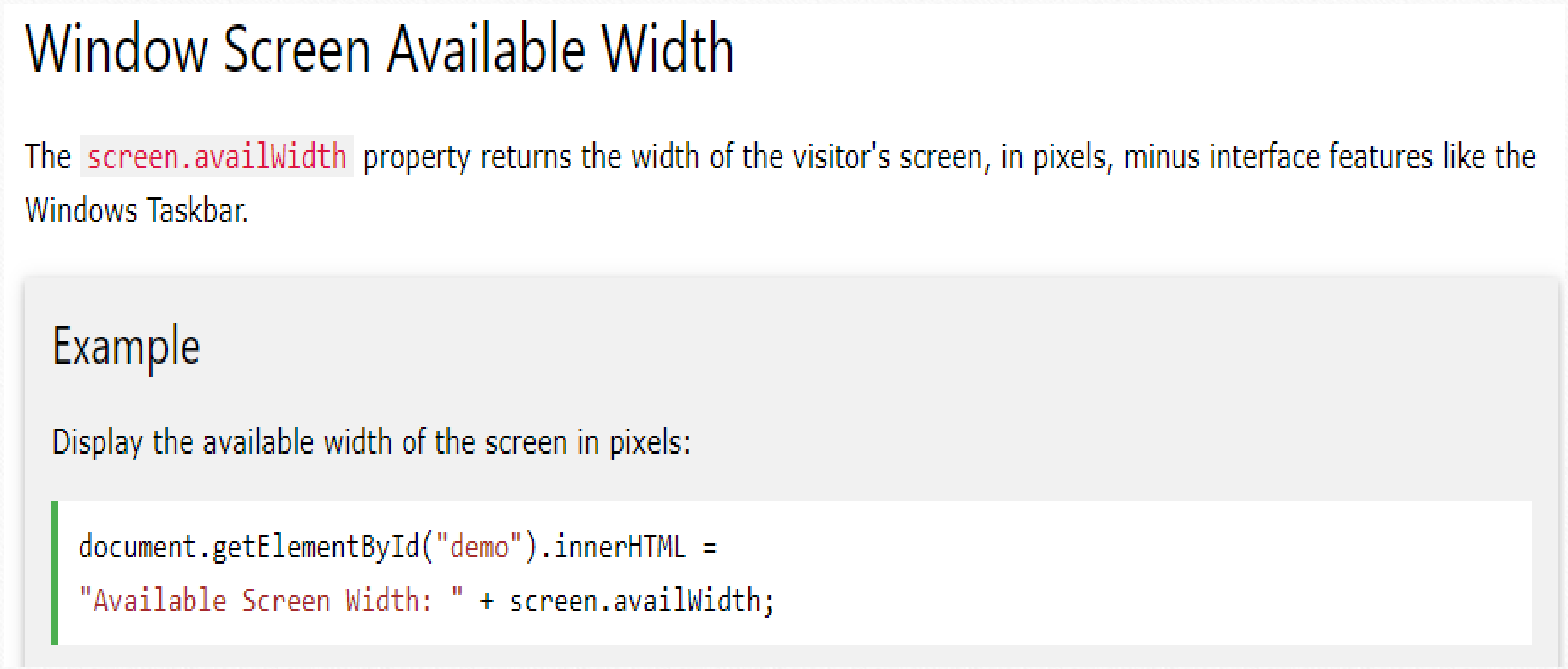
Example

Display the height of the screen in pixels:

```
document.getElementById("demo").innerHTML =
"Screen Height: " + screen.height;
```

Result will be:

```
Screen Height: 768
```



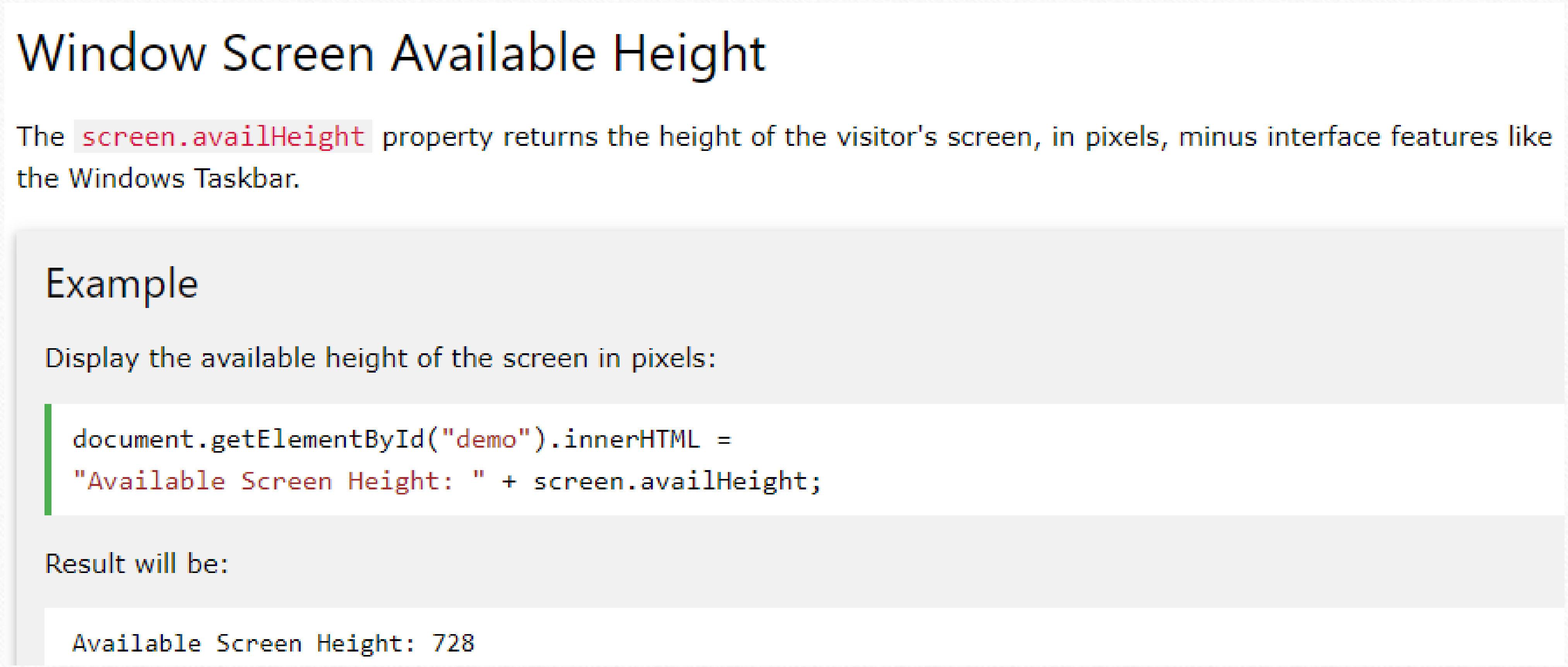
Window Screen Available Width

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

Example

Display the available width of the screen in pixels:

```
document.getElementById("demo").innerHTML =
"Available Screen Width: " + screen.availWidth;
```



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.

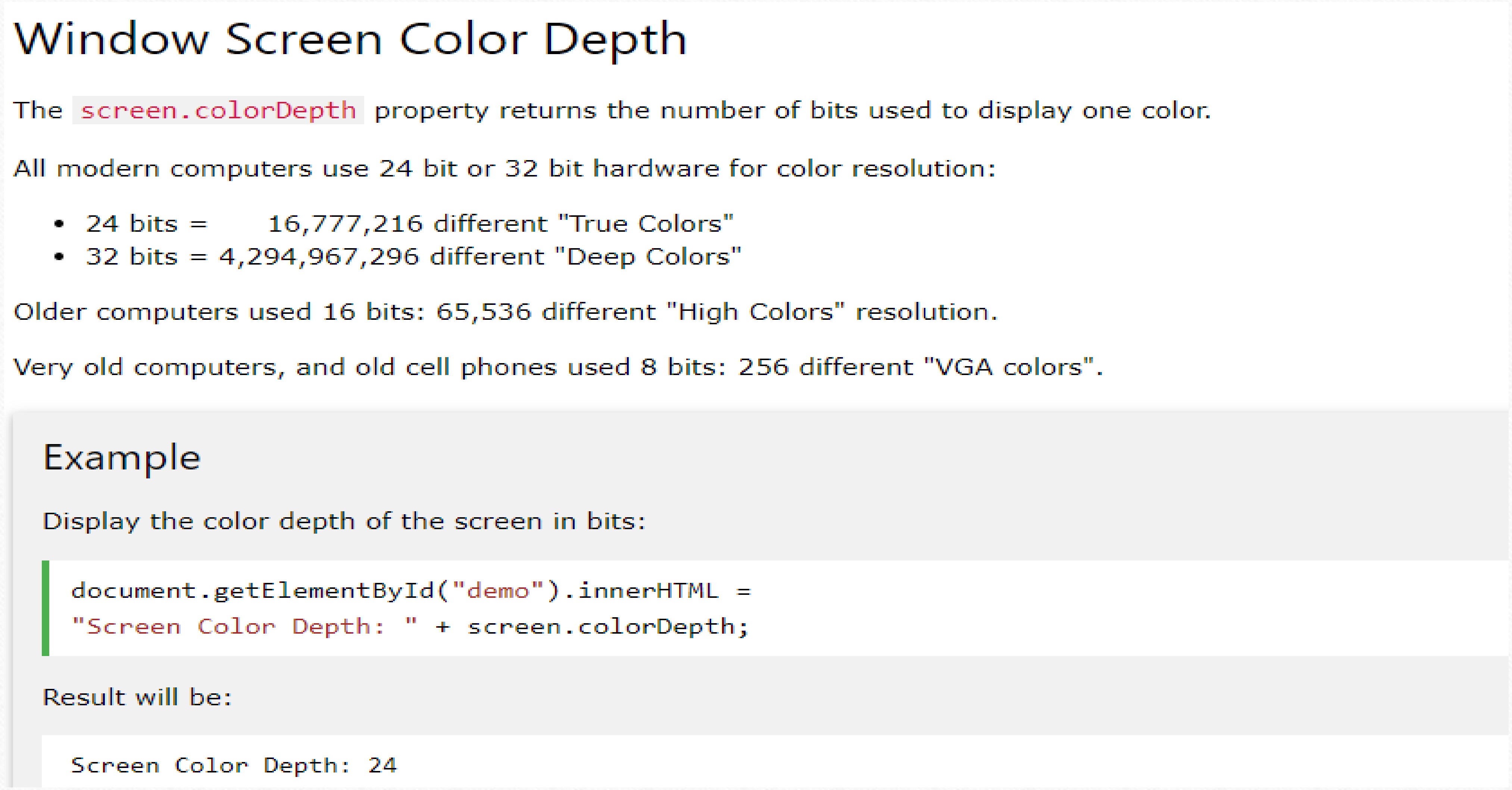
Example

Display the available height of the screen in pixels:

```
document.getElementById("demo").innerHTML =
"Available Screen Height: " + screen.availHeight;
```

Result will be:

```
Available Screen Height: 728
```



Window Screen Color Depth

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".

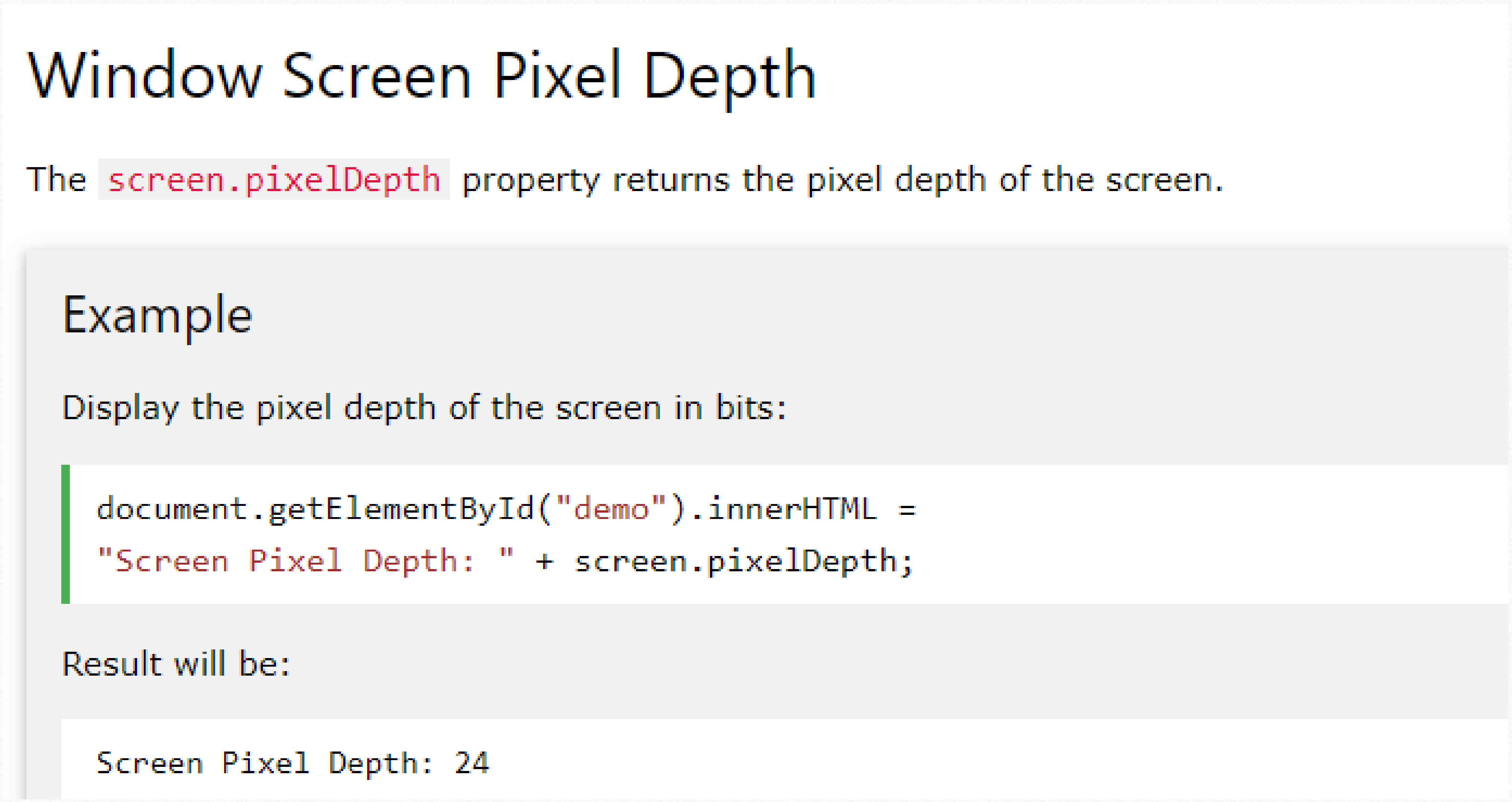
Example

Display the color depth of the screen in bits:

```
document.getElementById("demo").innerHTML =
"Screen Color Depth: " + screen.colorDepth;
```

Result will be:

```
Screen Color Depth: 24
```



Window Screen Pixel Depth

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

Example

Display the pixel depth of the screen in bits:

```
document.getElementById("demo").innerHTML =
"Screen Pixel Depth: " + screen.pixelDepth;
```

Result will be:

```
Screen Pixel Depth: 24
```

Window(Cont.)

❑ **Properties:**

Name	Description	Syntax
name	Return or set a window's name	window.name
status	Sets or returns String value containing the status bar text.	window.status="hi"
closed	Returns whether a window has been closed	window.closed
innerHeight	Returns the inner Height of a window's content area	window.innerHeight
innerWidth	Returns the inner width of a window's content area	window.innerWidth
outerHeight	Returns the outer height of a window, including toolbars/scrollbars	window.outerHeight
outerWidth	Returns the outer width of a window, including toolbars/scrollbars	window.outerWidth

Window(Cont.)

❑ **Properties:**

Name	Description	Syntax
screenLeft = screenX	Returns the horizontal coordinate of the window relative to the screen	window.screenLeft
screenTop = screenY	Returns the vertical coordinate of the window relative to the screen	window.screenY
pageXOffset = scrollX	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window	window.pageXOffset
pageYOffset = scrollY	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window	window.pageYOffset

Window(Cont.)

❑ Properties (Cont.):

Name	Description	Syntax
document	Reference to the current document object.	window.document
frames	An array referencing all of the frames in the current window.	window.frames[i]
frameElement	Returns the <iframe> element in which the current window is inserted	window.frameElement;
history	Reference to the History object of JavaScript	window.history
navigator	Reference to the browser application	window.navigator
location	Reference to the Location object of JavaScript	window.location

Window(Cont.)

❑ **Methods:**

Name	Description	Syntax
alert()	Displays an alert box with a message and an OK button	window.alert("Hello")
confirm()	Displays a dialog box with a message and an OK, returning true, and a Cancel, returning false	Window.confrim("Do you want to exit")
prompt()	Displays a dialog box that prompts the user for input	name=prompt("Please enter your name","")
open()	Opens a new browser window http://www.w3schools.com/jsref/met_win_open.asp	window.open(url, name, properties)
close()	close a specified window	window.close()

Window(Cont.)

❑ **Methods (Cont.):**

Name	Description	Syntax
focus()	Sets focus to the current window	window.focus();
blur()	Removes focus from the current window	window.blur()
getSelection()	Returns a Selection object representing the range of text selected by the user	window.getSelection();
stop()	Stops the window from loading	window.stop();
print()	Print the contents of the specified window.	window.print()

Window(Cont.)

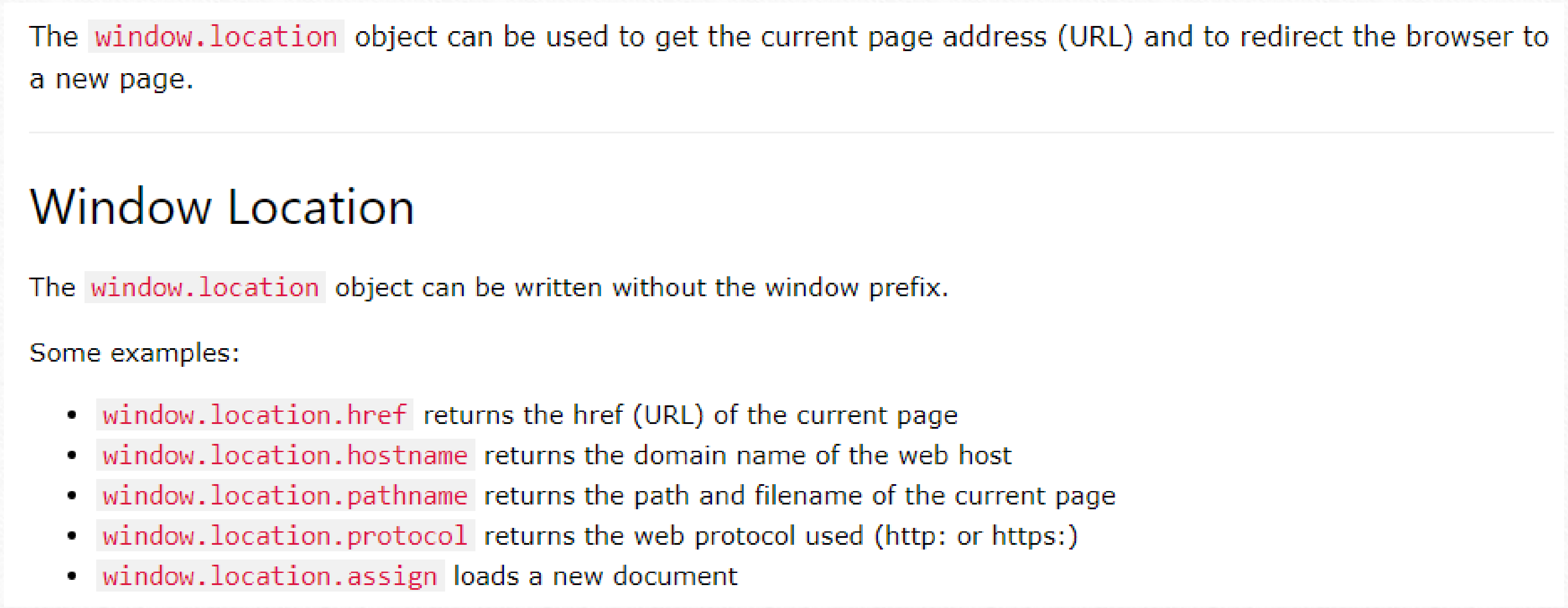
❑ Methods(Cont.):

Name	Description	Syntax
moveTo(h,v)	Moves the window to horizontal and vertical position relative top-left of screen:	window. moveTo(,)
moveBy(h,v)	Moves the window by + or - horizontal and vertical pixels:	window.moveBy(,)
resizeTo(h,v)	Changes the size of the window to horizontal and vertical number of pixels:	window.resizeTo(,)
resizeBy(h,v)	Changes the size of the window by + or - horizontal and vertical pixels:	window.resizeBy(,)
scrollTo(h,v)	Scrolls the document in the current window or frame to horizontal and vertical pixel postions from top of document	window.scrollTo(,)
scrollBy(h,v)	Scrolls the document in the current window or frame by + or - horizontal and vertical pixel from current position:	window.scrollBy(,)

❑ Methods(Cont.):

Name	Description	Syntax
setInterval(expression, interval)	Evaluates an expression at specified intervals	window.setInterval(*“alert()”,500*) Or window.setInterval(*funcName,500*) Or t= window.setInterval(*“alert()”,500*)
clearInterval(interval_Obj_Name)	Used to clear a time interval set using the above method	Window.clearInterval(t)
setTimeout()	Used to execute an expression or function after a time interval (in millisecond).	window.setTimeOut(*exp, time_interval*)
clearTimeout()	Used to clear a timeout set using the above method	

JavaScript Window Location



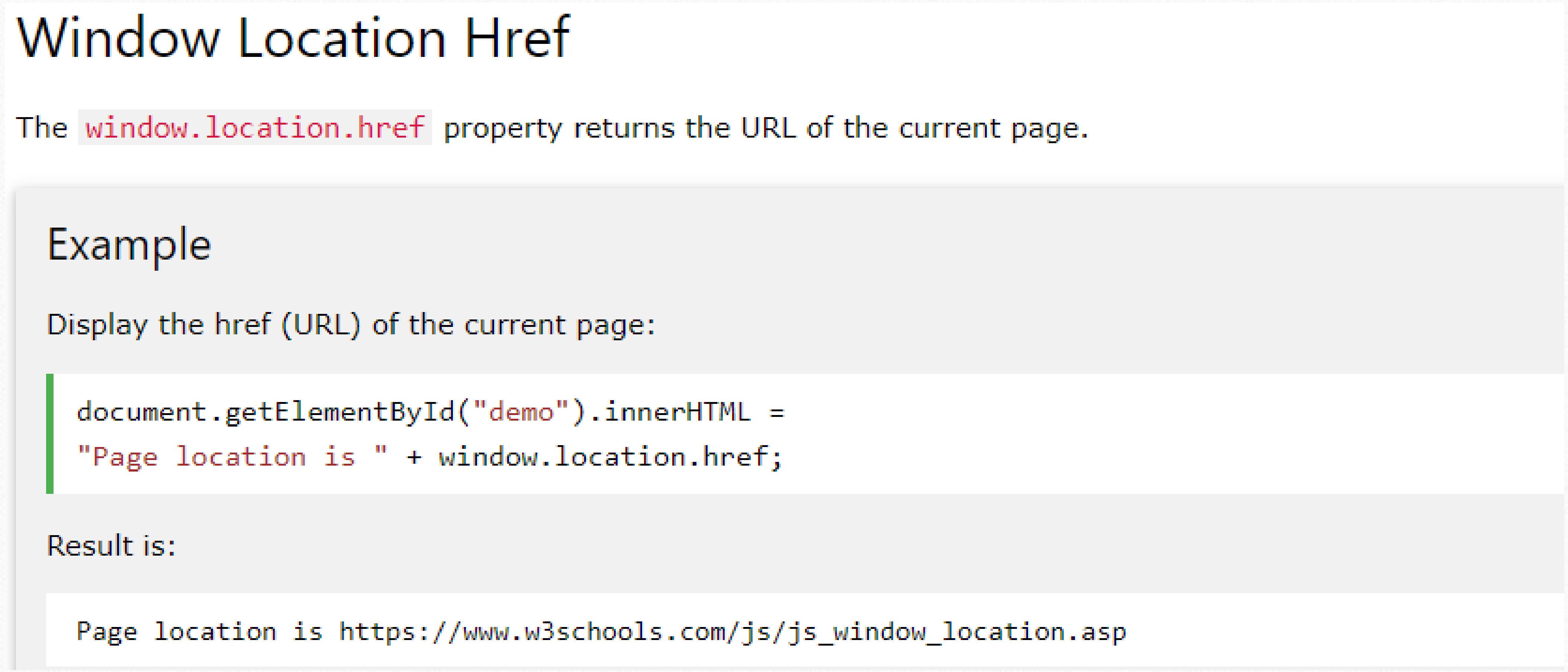
The `window.location` object can be used to get the current page address (URL) and to redirect the browser to a new page.

Window Location

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 used (http: or https:)
- `window.location.assign` loads a new document



Window Location Href

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

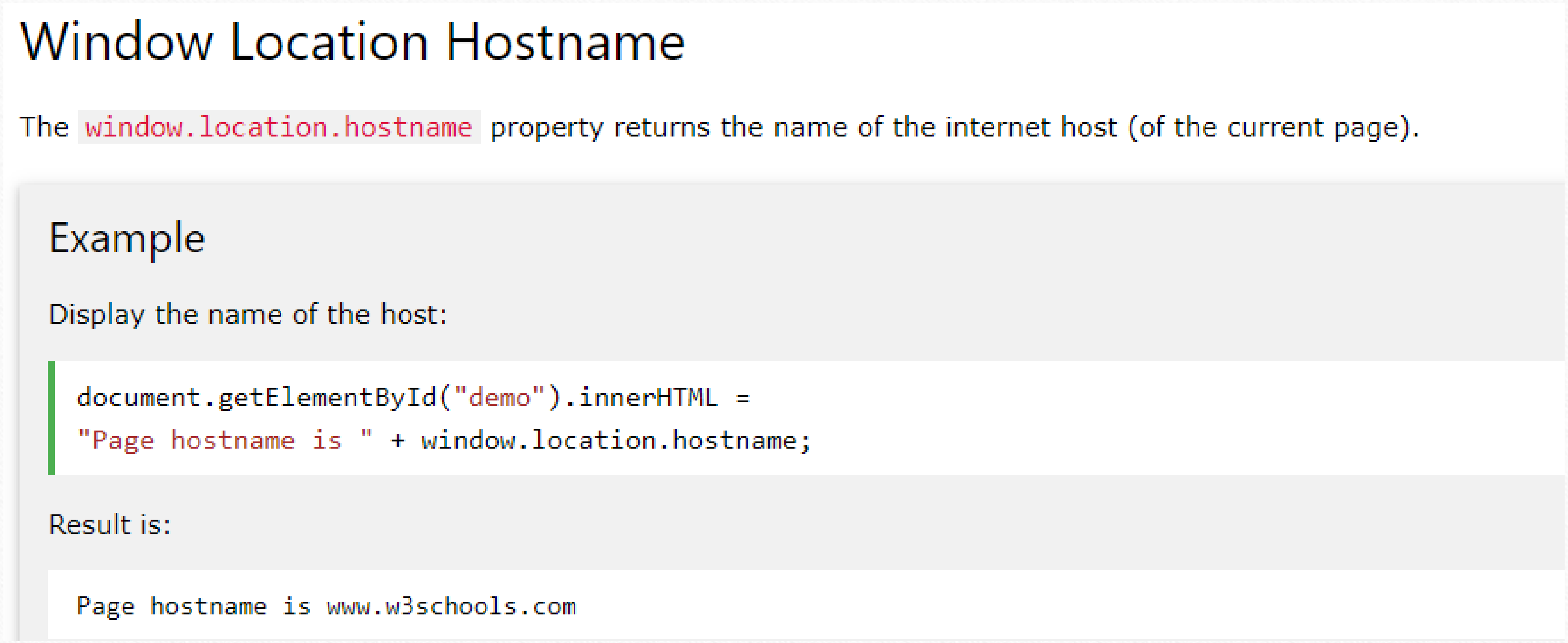
Example

Display the href (URL) of the current page:

```
document.getElementById("demo").innerHTML =
"Page location is " + window.location.href;
```

Result is:

```
Page location is https://www.w3schools.com/js/js_window_location.asp
```



Window Location Hostname

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

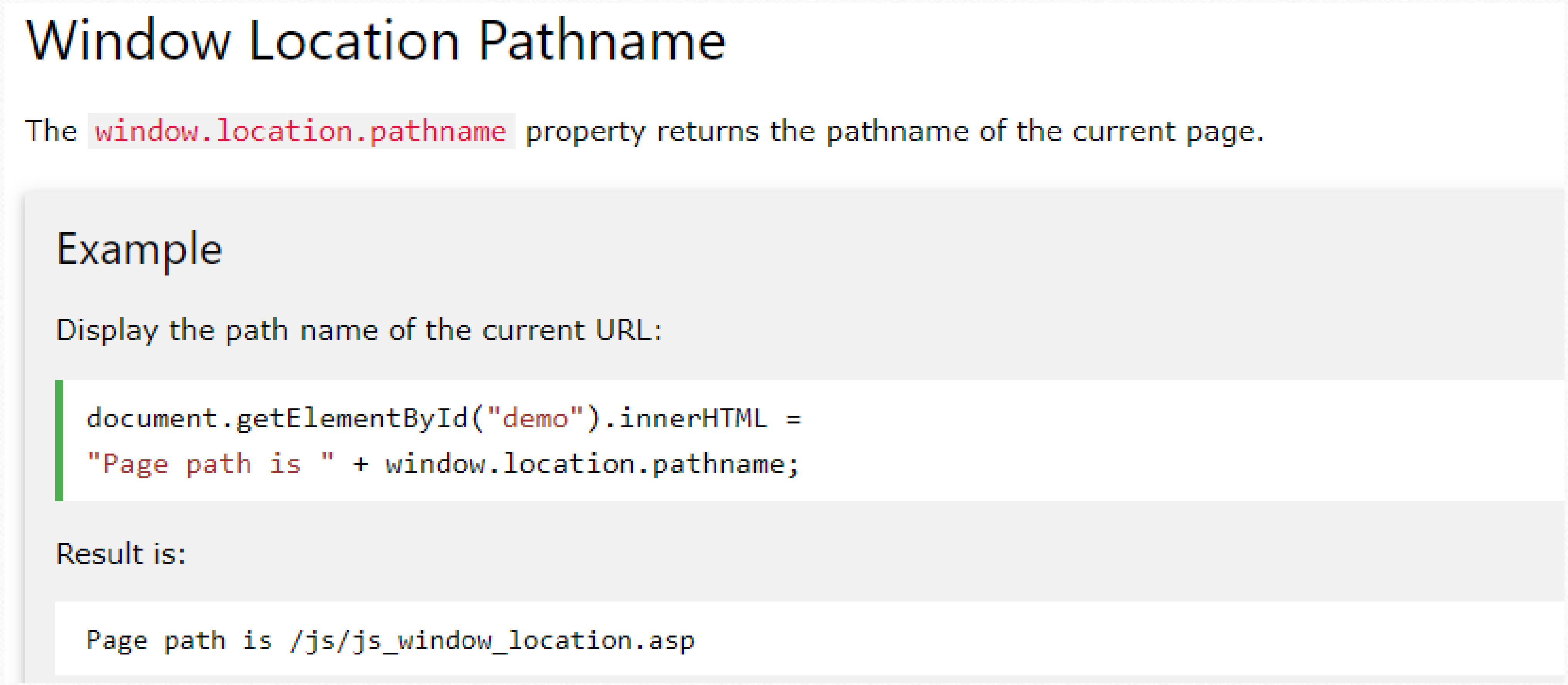
Example

Display the name of the host:

```
document.getElementById("demo").innerHTML =
"Page hostname is " + window.location.hostname;
```

Result is:

```
Page hostname is www.w3schools.com
```



Window Location Pathname

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

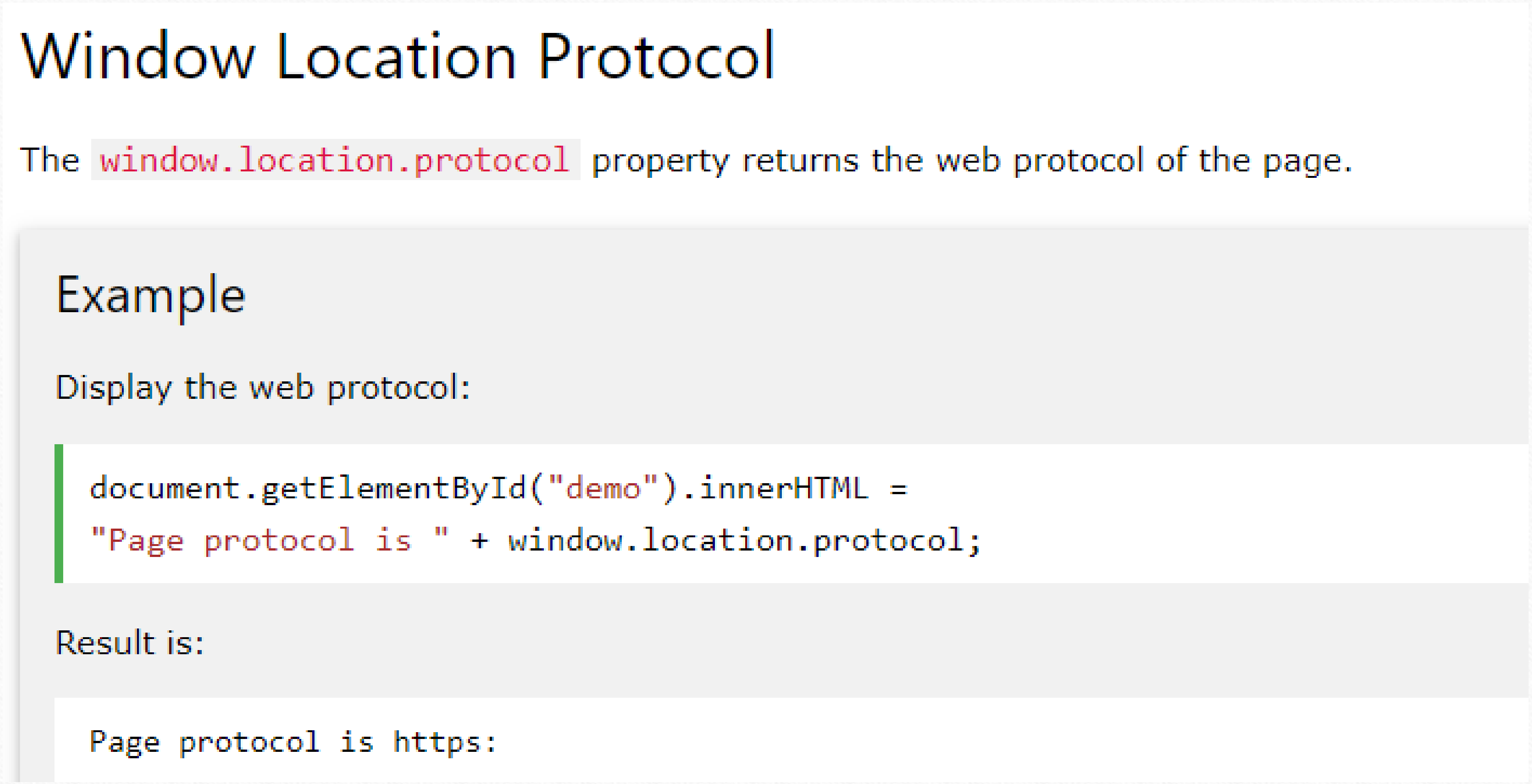
Example

Display the path name of the current URL:

```
document.getElementById("demo").innerHTML =
"Page path is " + window.location.pathname;
```

Result is:

```
Page path is /js/js_window_location.asp
```



Window Location Protocol

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

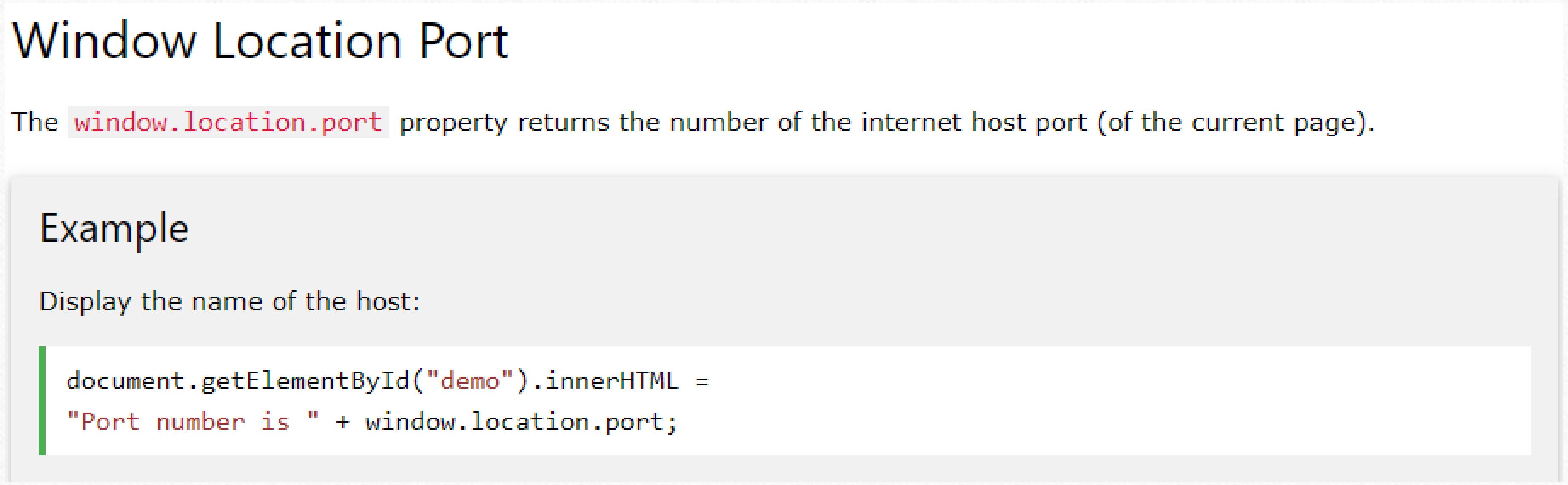
Example

Display the web protocol:

```
document.getElementById("demo").innerHTML =
"Page protocol is " + window.location.protocol;
```

Result is:

```
Page protocol is https:
```



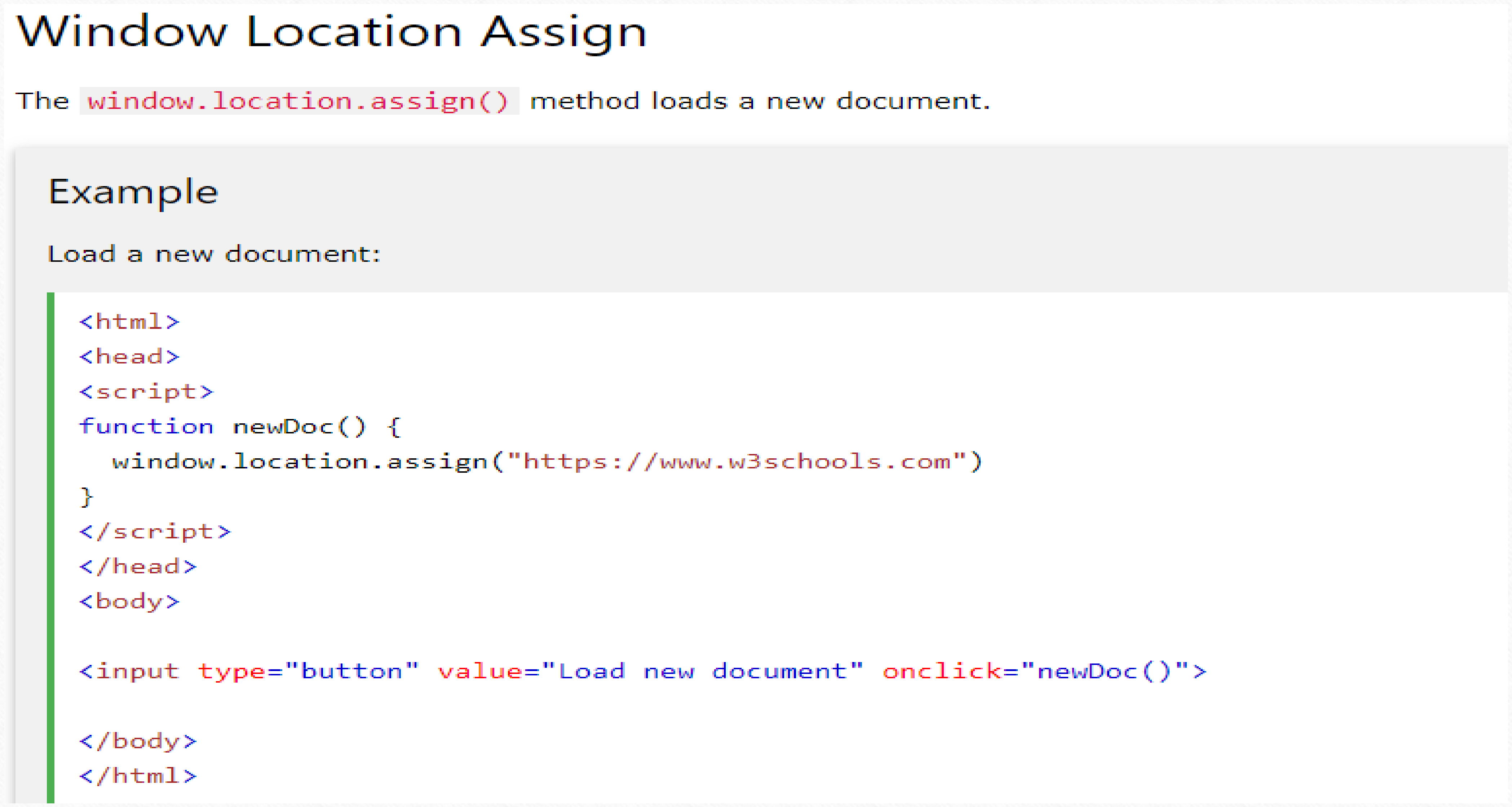
Window Location Port

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

Example

Display the name of the host:

```
document.getElementById("demo").innerHTML =
"Port number is " + window.location.port;
```



Window Location Assign

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

Example

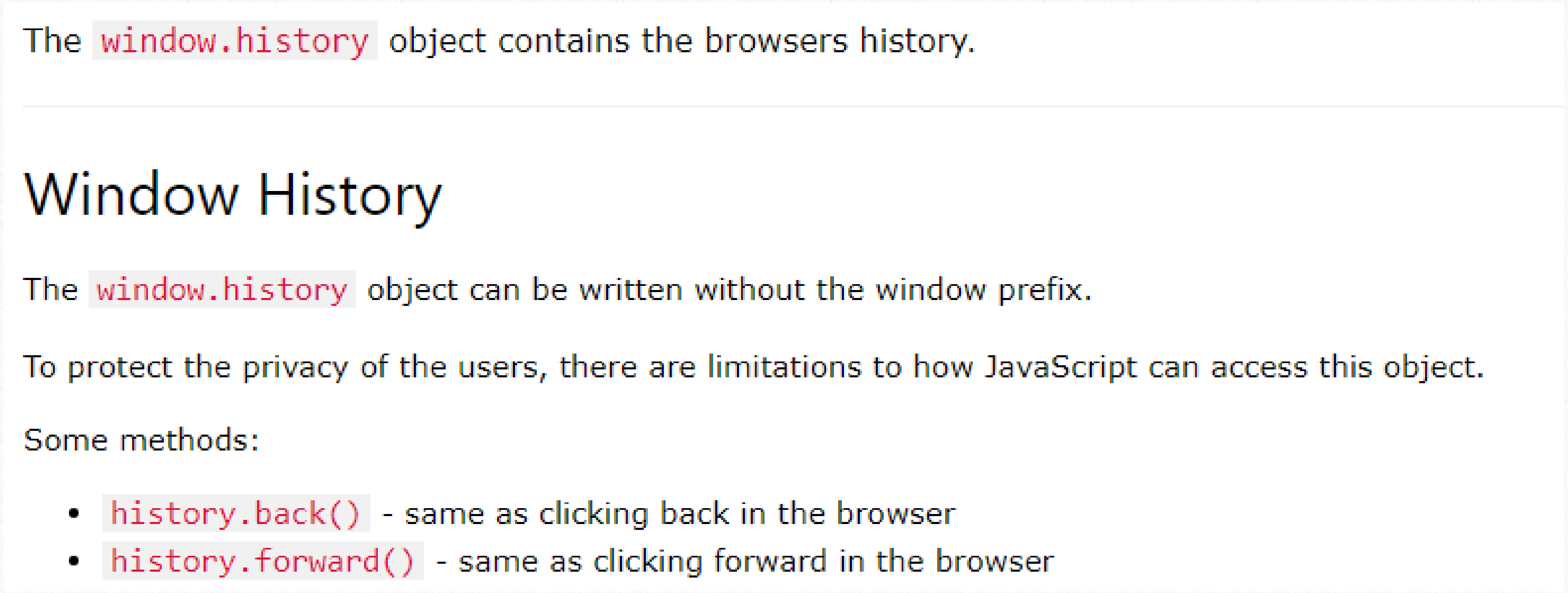
Load a new document:

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

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

</body>
</html>
```

JavaScript Window History



The `window.history` object contains the browsers history.

Window History

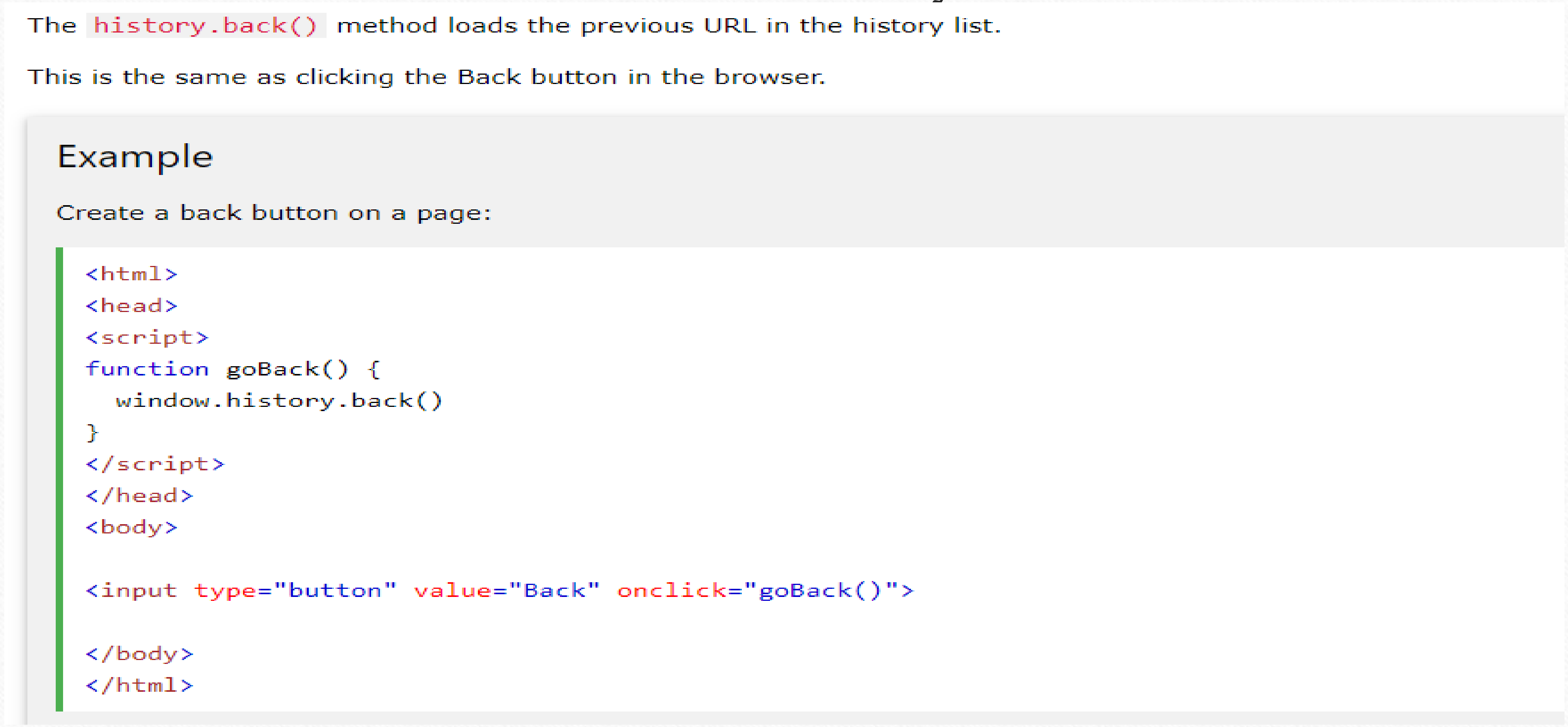
The `window.history` object can be written without the window prefix.

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

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

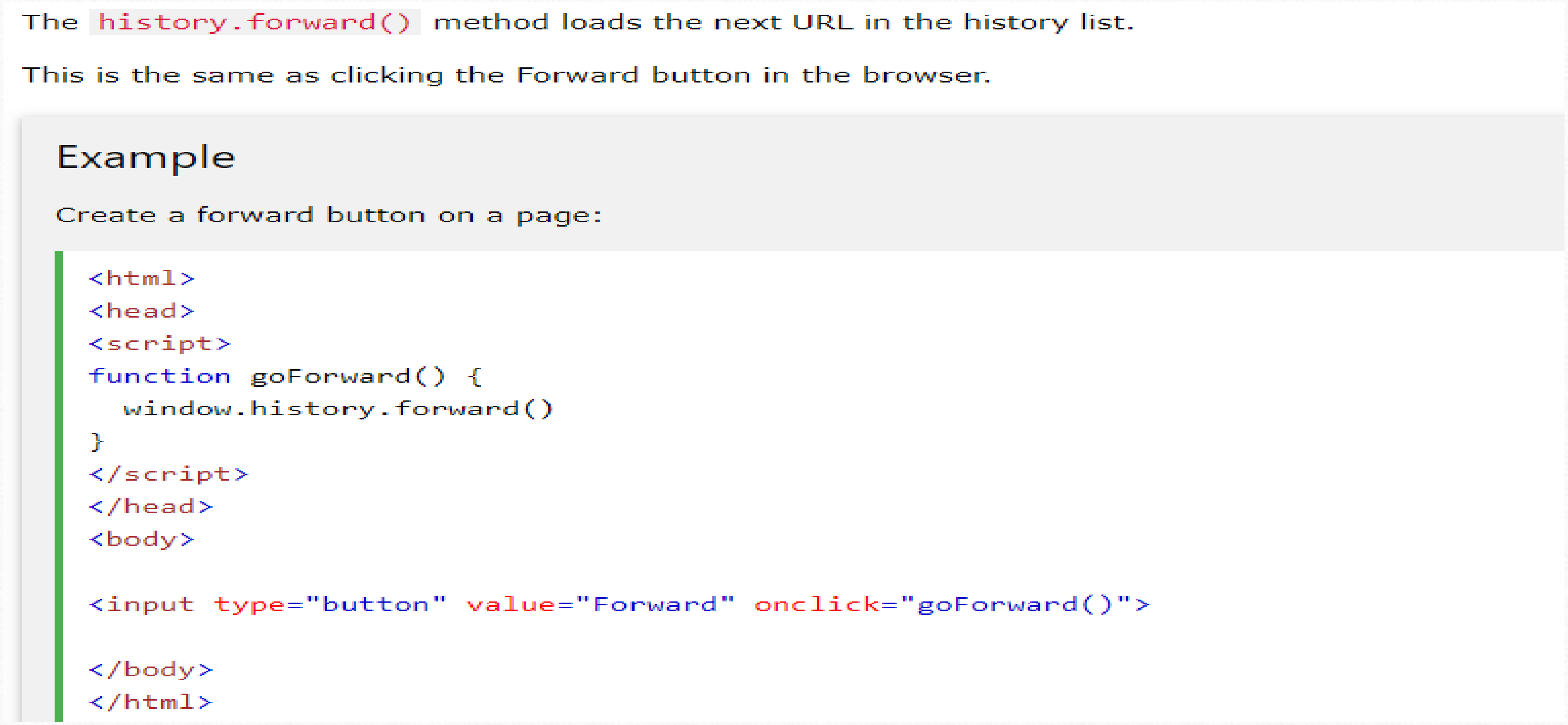
Create a back button on a page:

```
<html>
<head>
<script>
function goBack() {
  window.history.back()
}
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

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

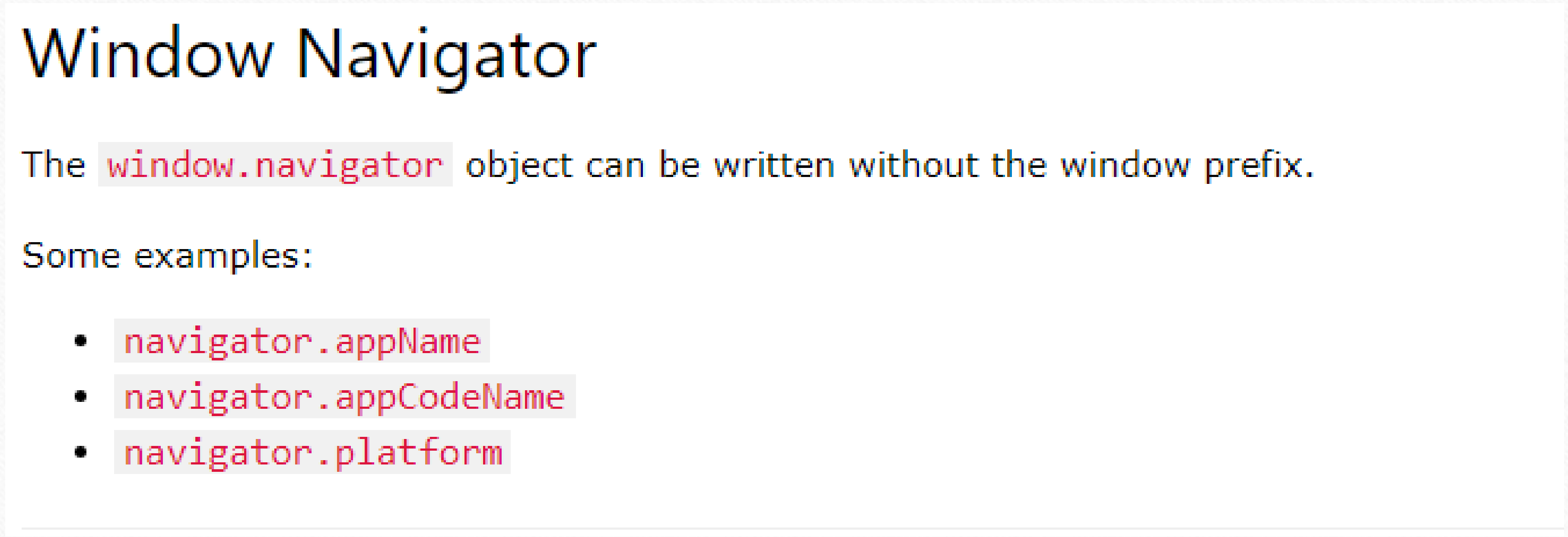
Create a forward button on a page:

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

<input type="button" value="Forward" onclick="goForward()">

</body>
</html>
```

JavaScript Window Navigator

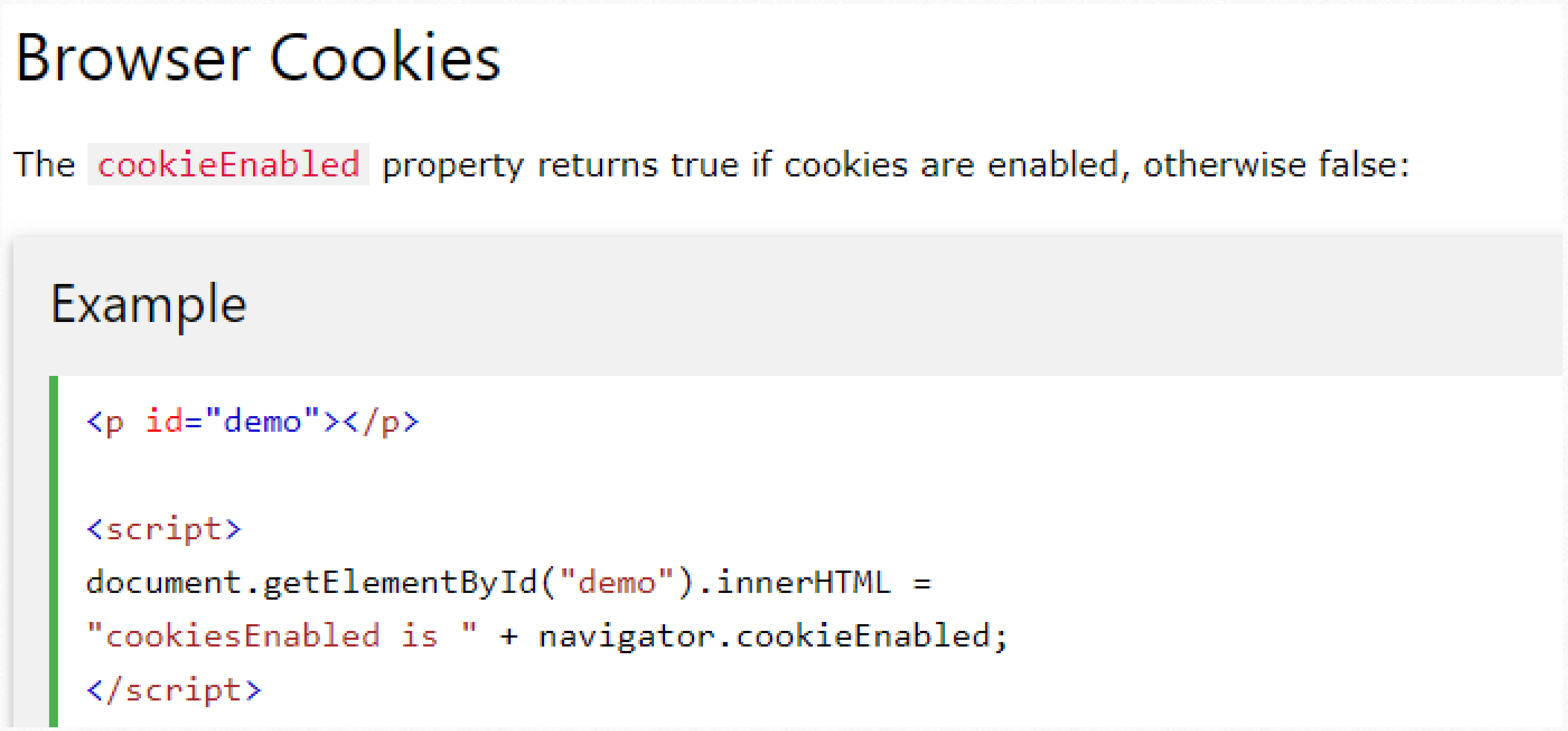


Window Navigator

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

Some examples:

- `navigator.appName`
- `navigator.appCodeName`
- `navigator.platform`



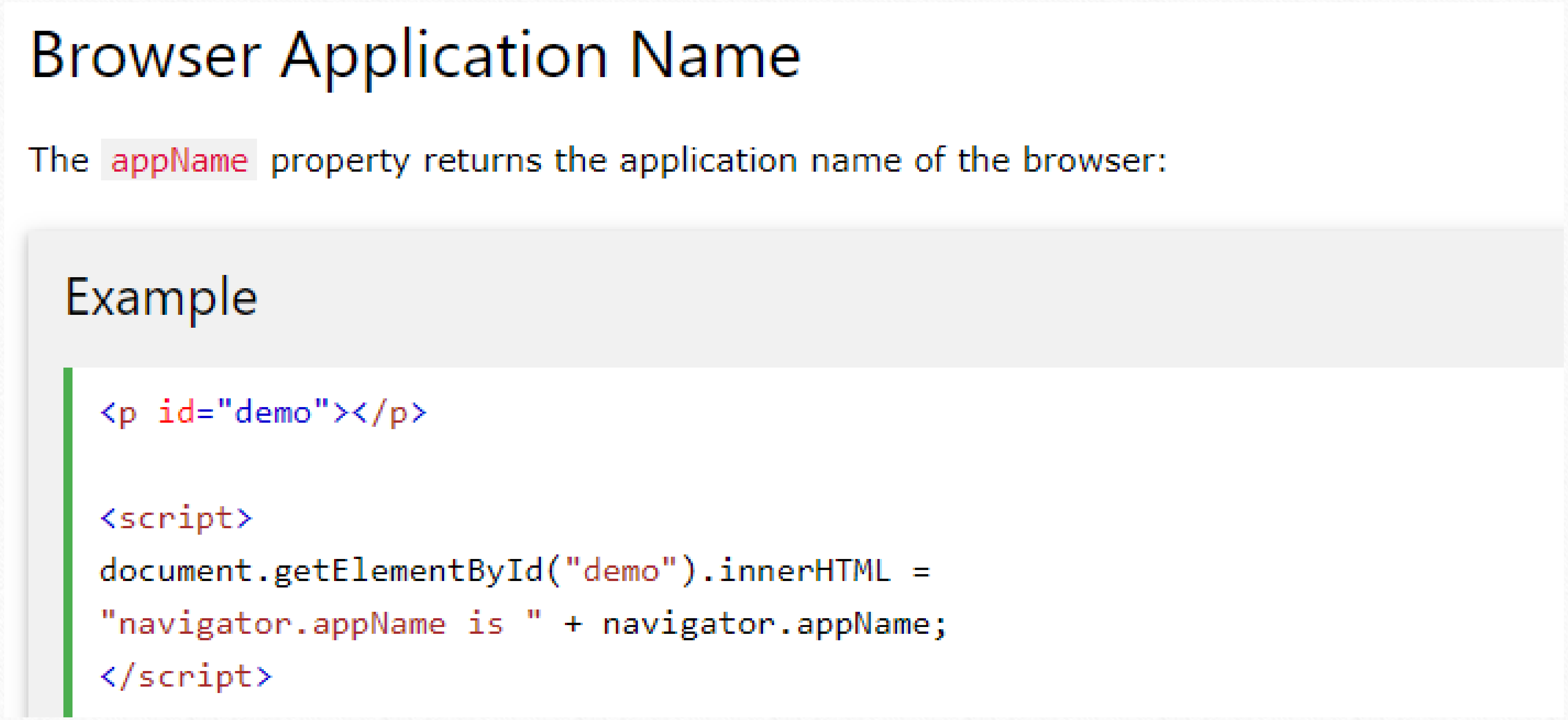
Browser Cookies

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

Example

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

<script>
document.getElementById("demo").innerHTML =
"cookiesEnabled is " + navigator.cookieEnabled;
</script>
```



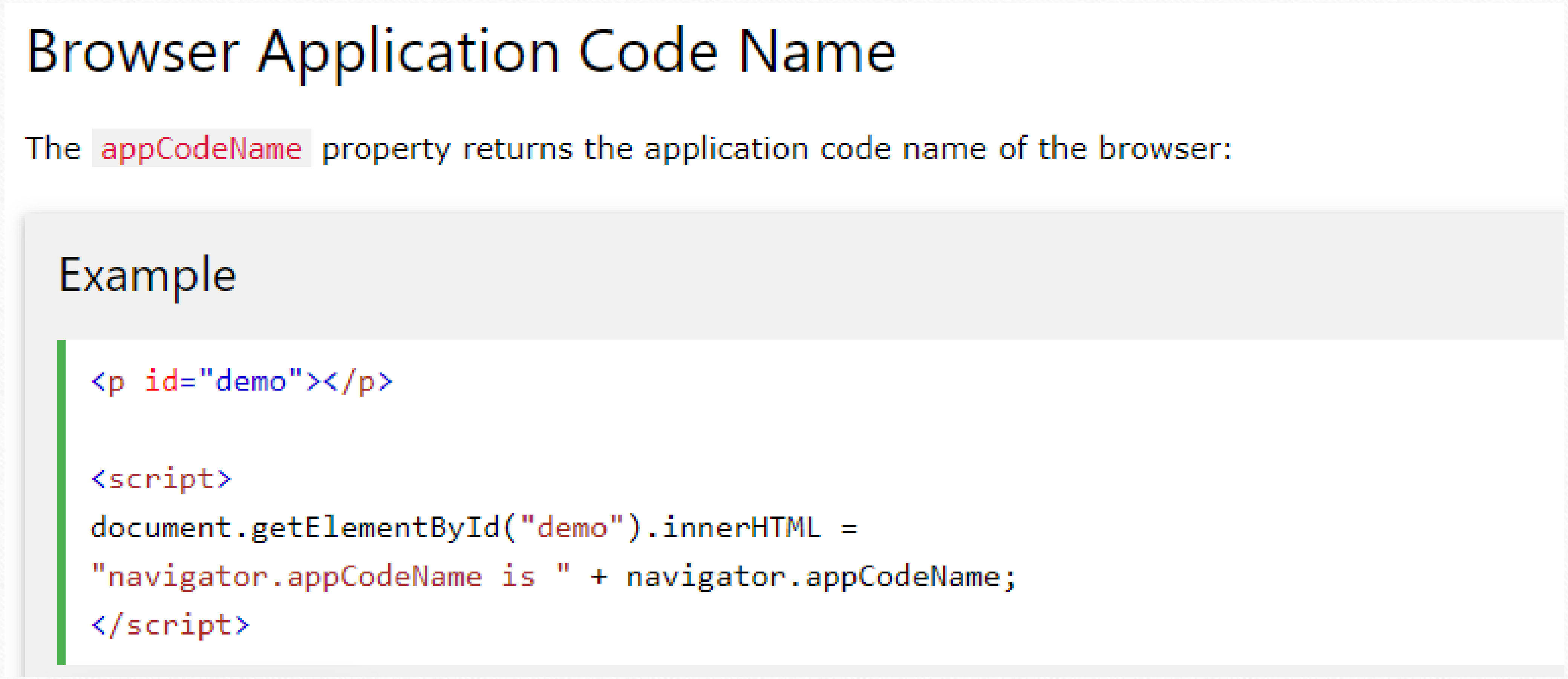
Browser Application Name

The `appName` property returns the application name of the browser:

Example

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

<script>
document.getElementById("demo").innerHTML =
"navigator.appName is " + navigator.appName;
</script>
```



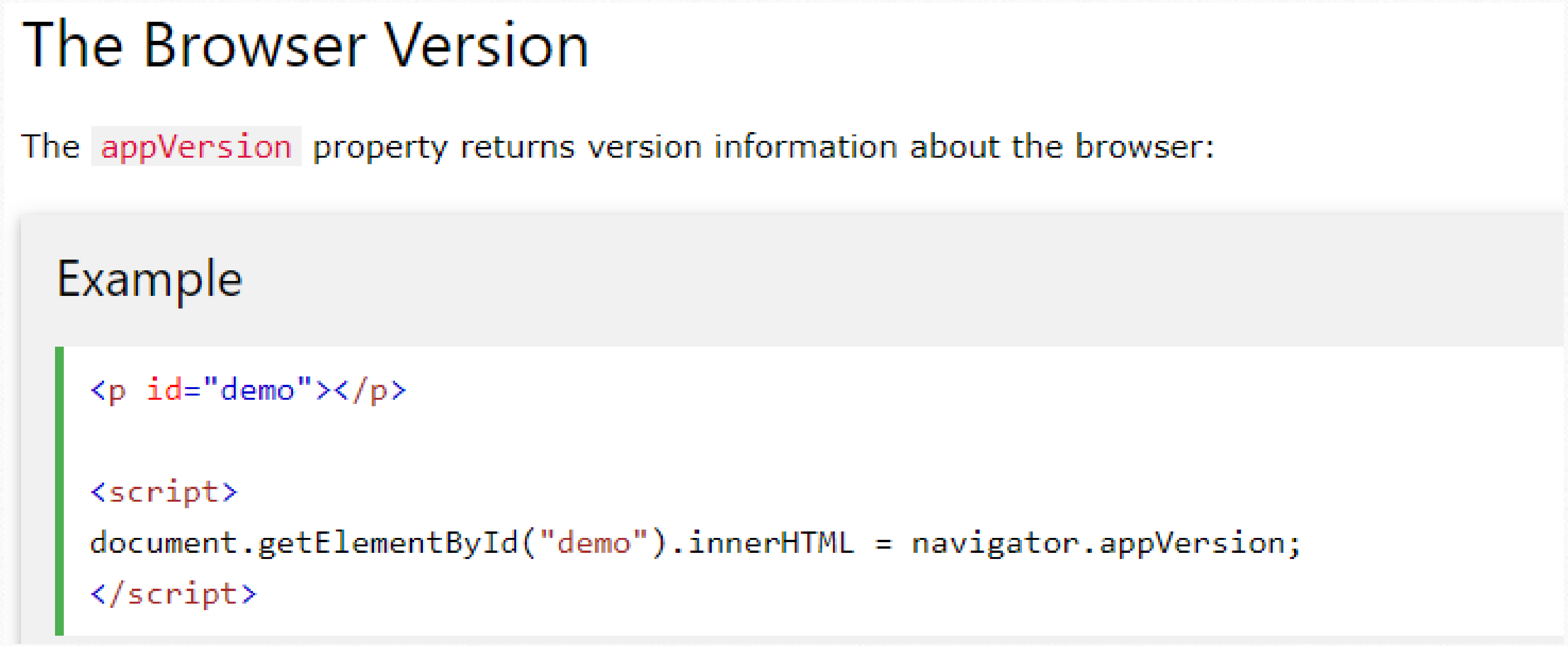
Browser Application Code Name

The `appCodeName` property returns the application code name of the browser:

Example

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

<script>
document.getElementById("demo").innerHTML =
"navigator.appCodeName is " + navigator.appCodeName;
</script>
```



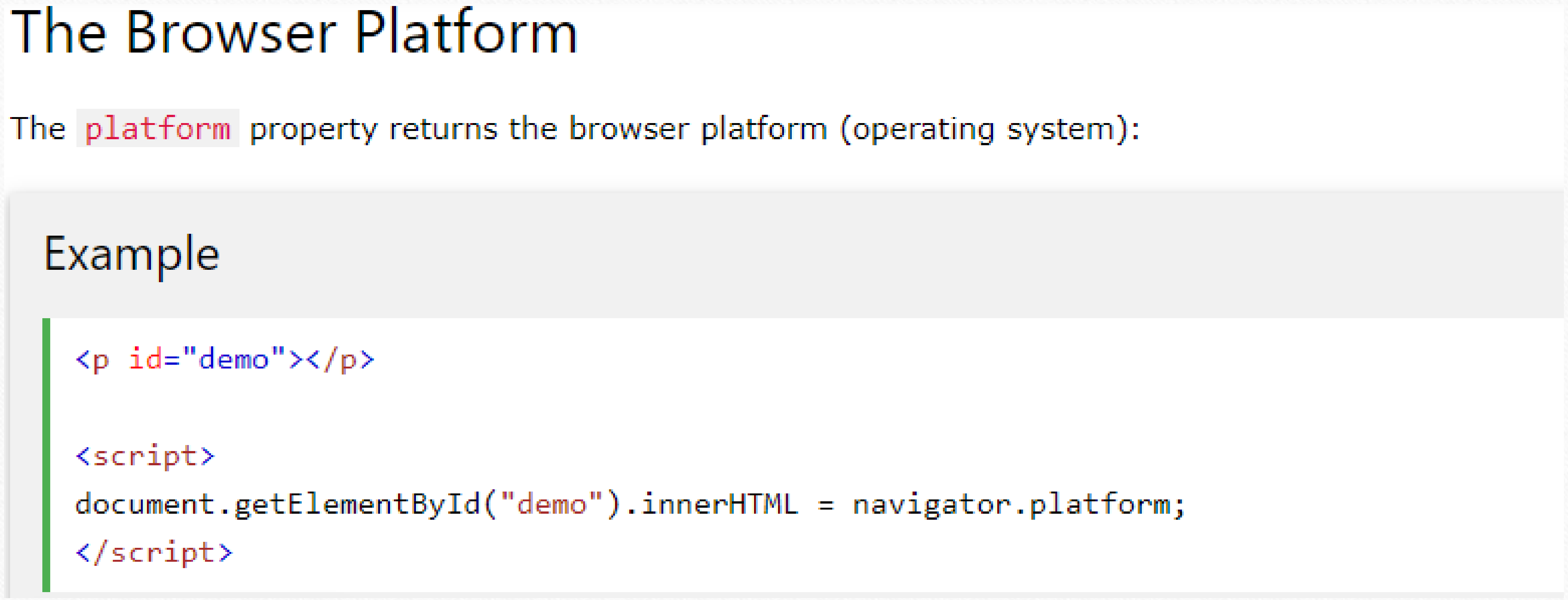
The Browser Version

The `appVersion` property returns version information about the browser:

Example

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

<script>
document.getElementById("demo").innerHTML = navigator.appVersion;
</script>
```



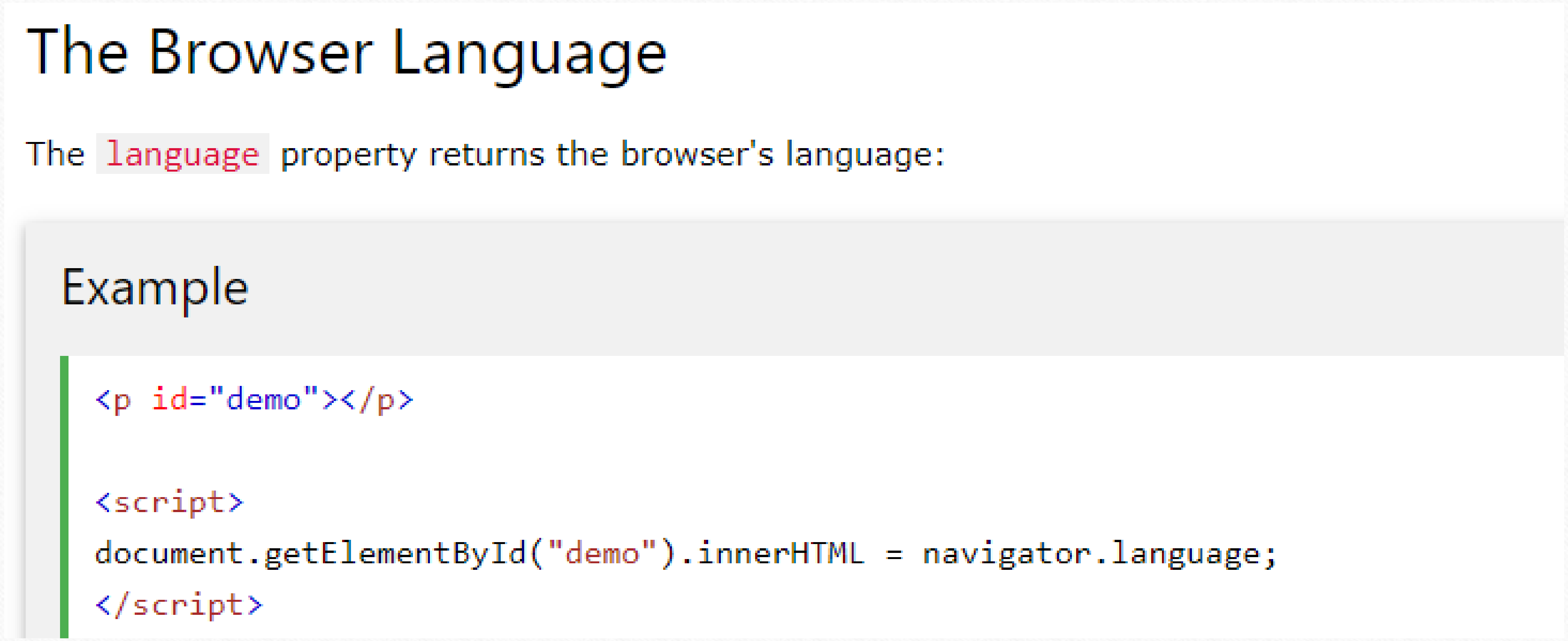
The Browser Platform

The `platform` property returns the browser platform (operating system):

Example

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

<script>
document.getElementById("demo").innerHTML = navigator.platform;
</script>
```



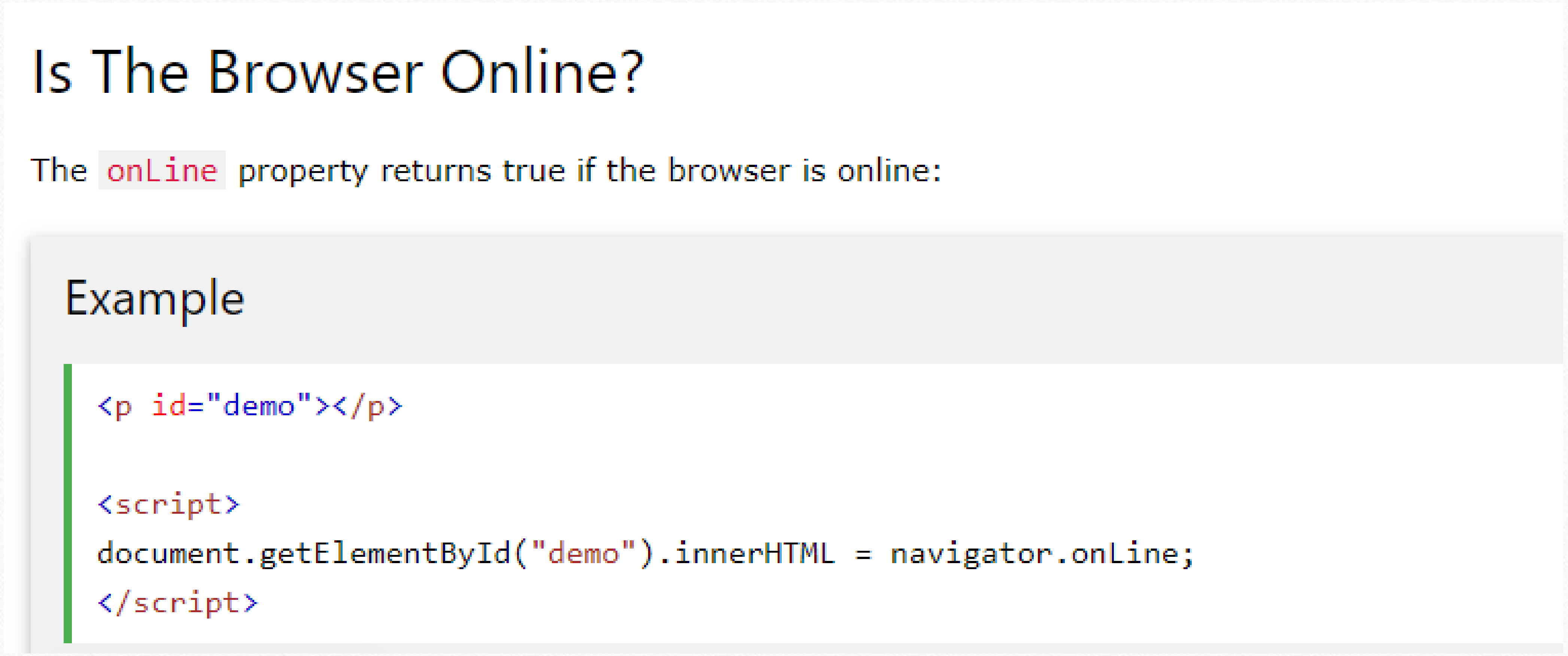
The Browser Language

The `language` property returns the browser's language:

Example

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

<script>
document.getElementById("demo").innerHTML = navigator.language;
</script>
```



Is The Browser Online?

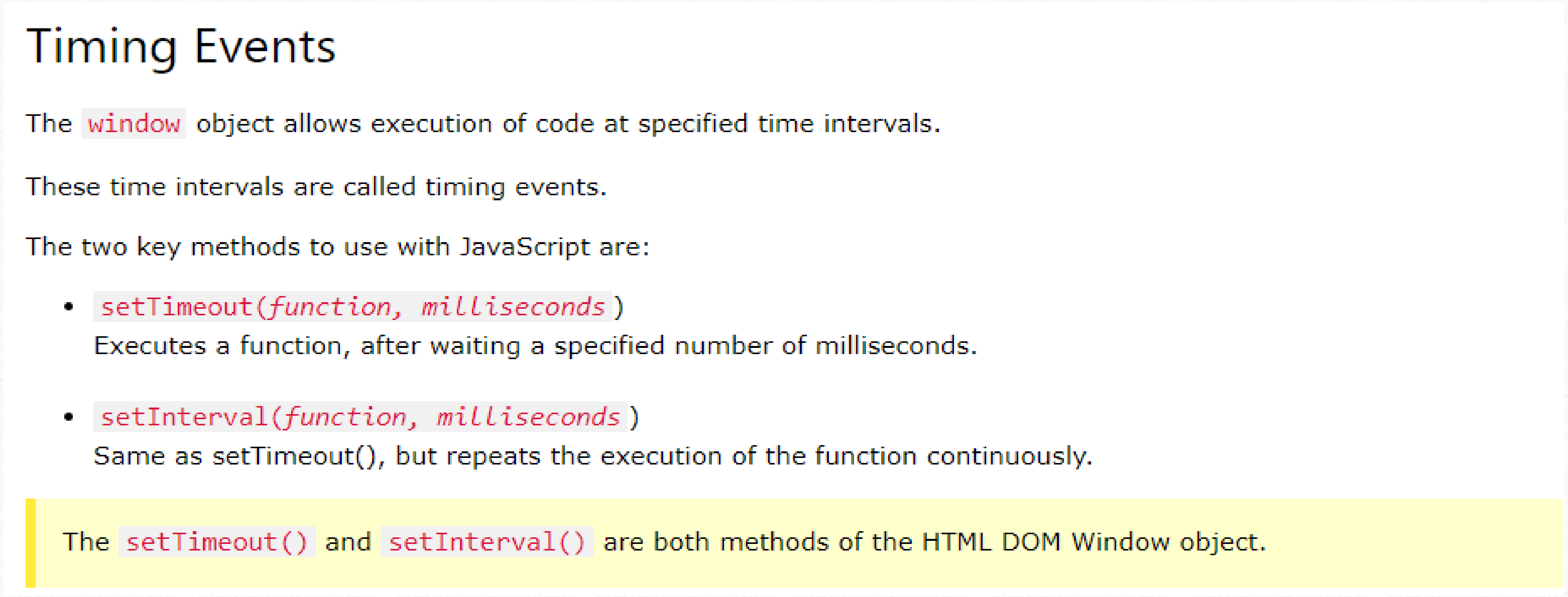
The `onLine` property returns true if the browser is online:

Example

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

<script>
document.getElementById("demo").innerHTML = navigator.onLine;
</script>
```

JavaScript Timing Events



Timing Events

The `window` object allows execution of code at specified time intervals.

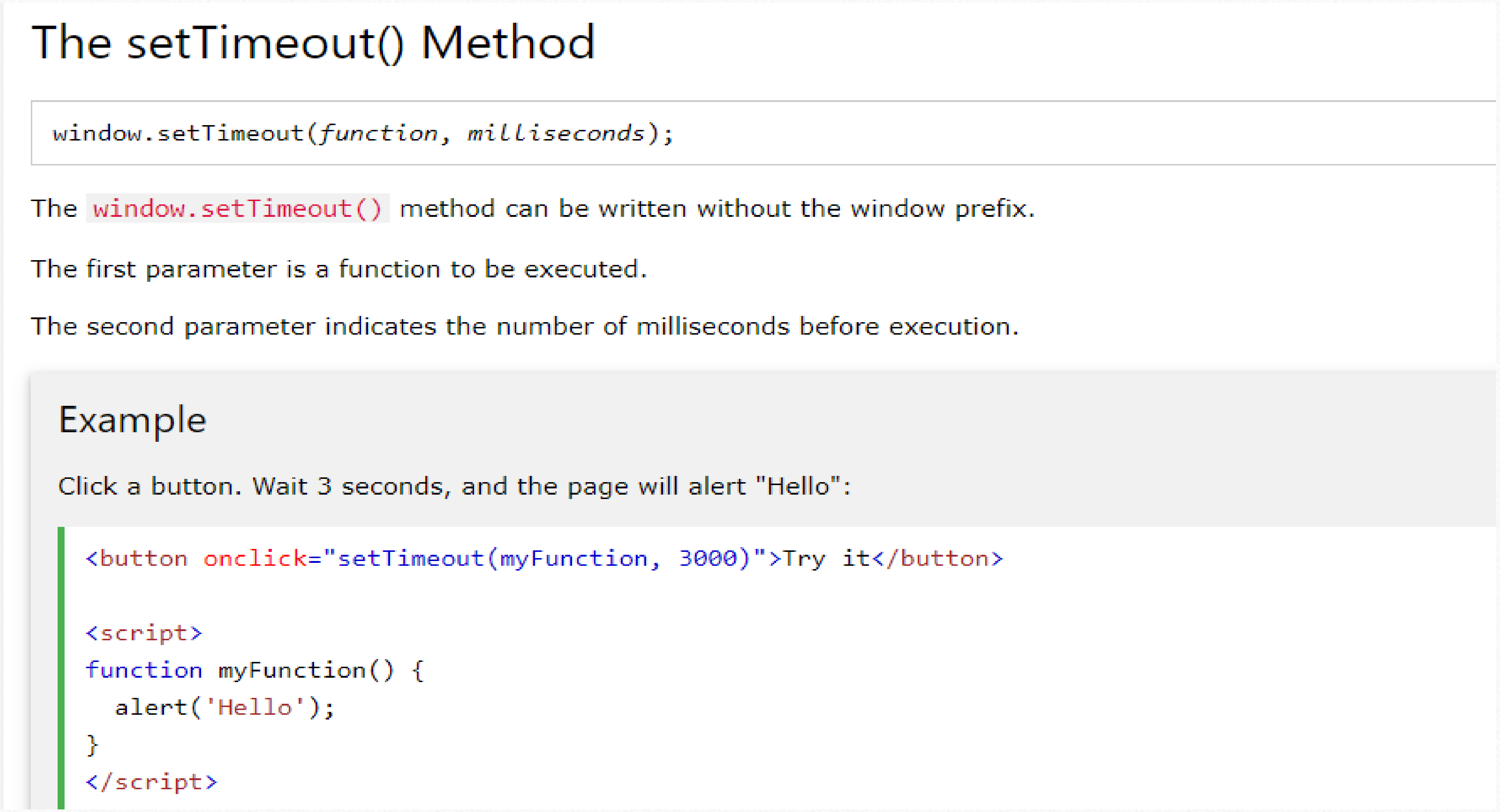
These time intervals are called timing events.

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 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.

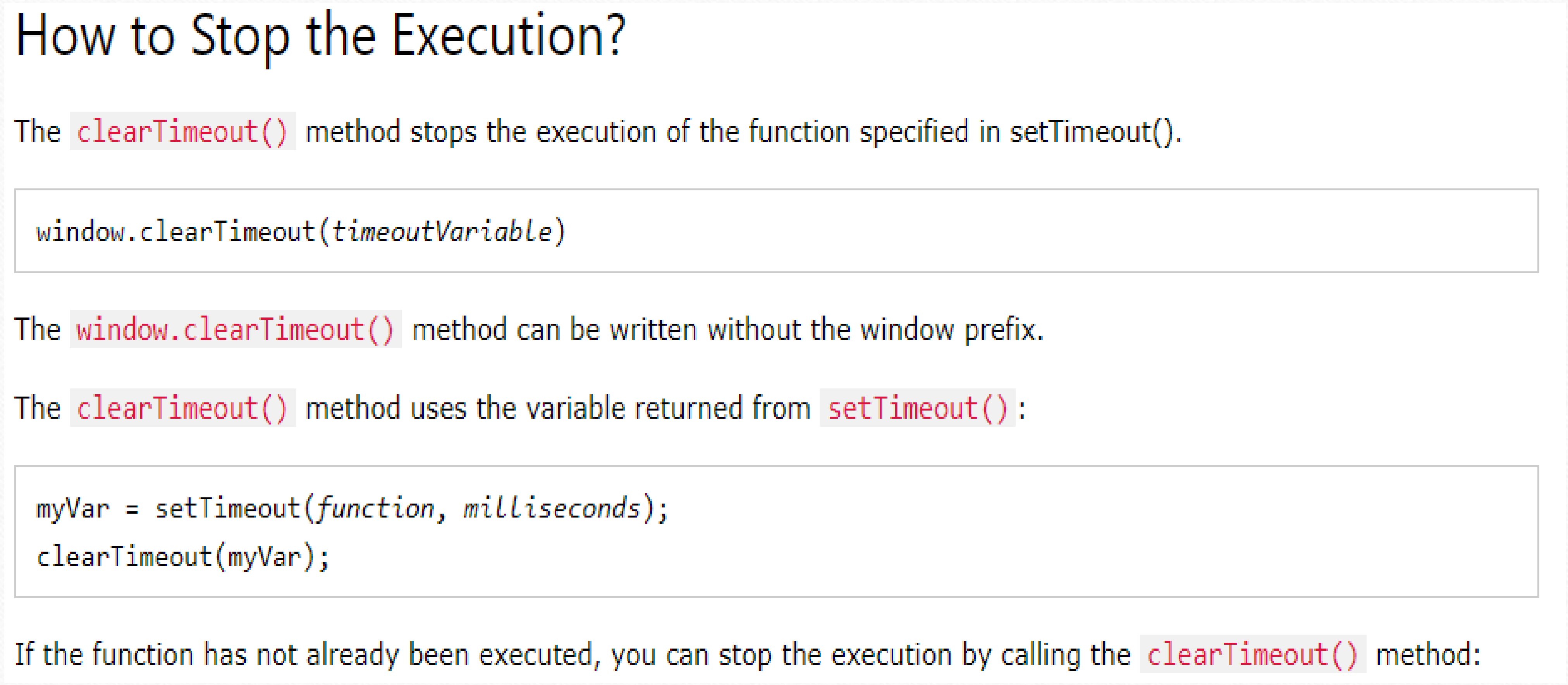
The second parameter indicates the number of milliseconds before execution.

Example

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

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

<script>
function myFunction() {
  alert('Hello');
}
</script>
```



How to Stop the Execution?

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

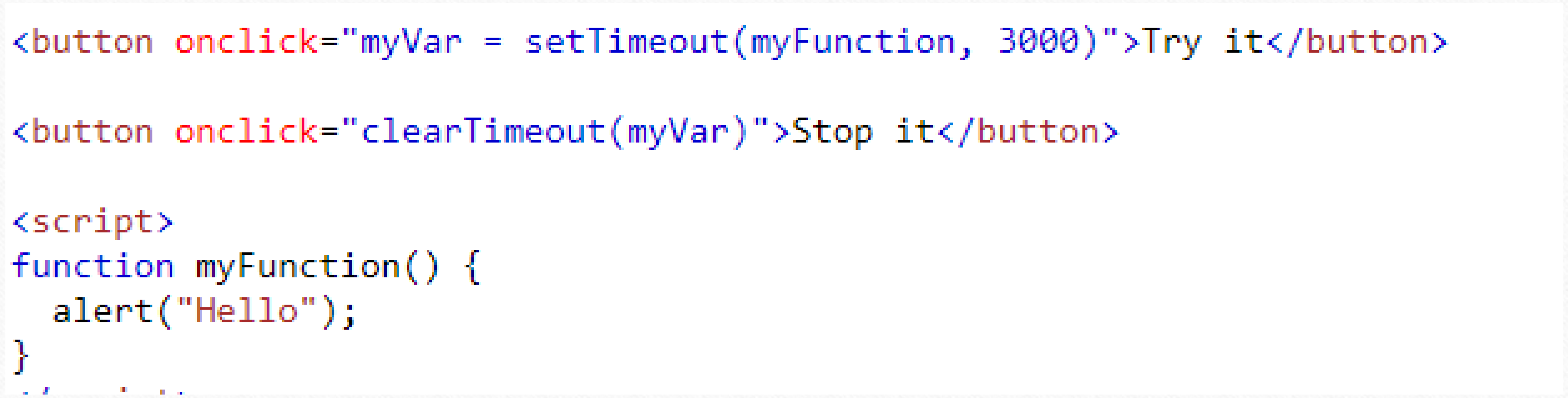
```
window.clearTimeout(timeoutVariable)
```

The `window.clearTimeout()` method can be written without the window prefix.

The `clearTimeout()` method uses the variable returned from `setTimeout()`:

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

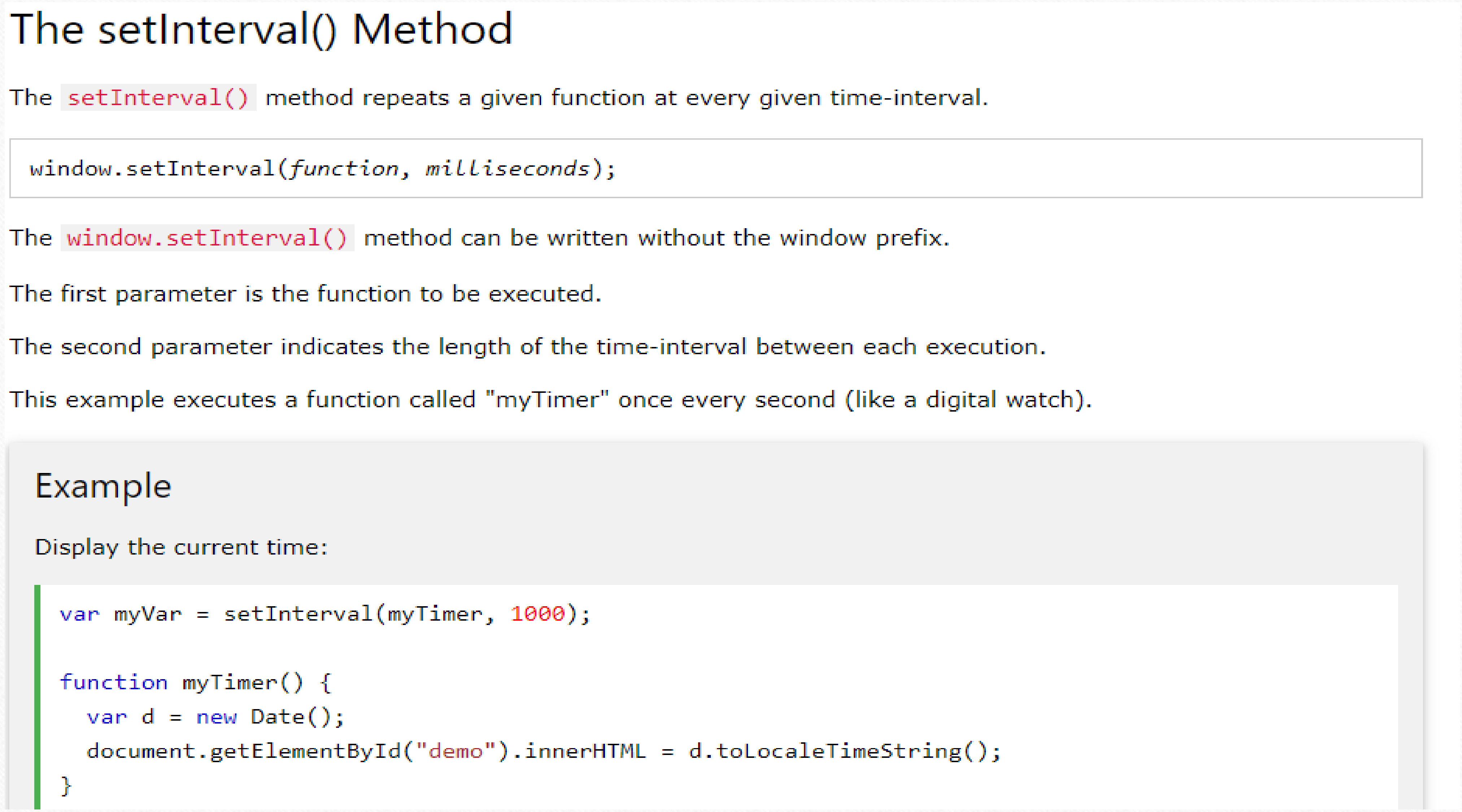
If the function has not already been executed, you can stop the execution by calling the `clearTimeout()` method:



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

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

<script>
function myFunction() {
  alert("Hello");
}
```



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.

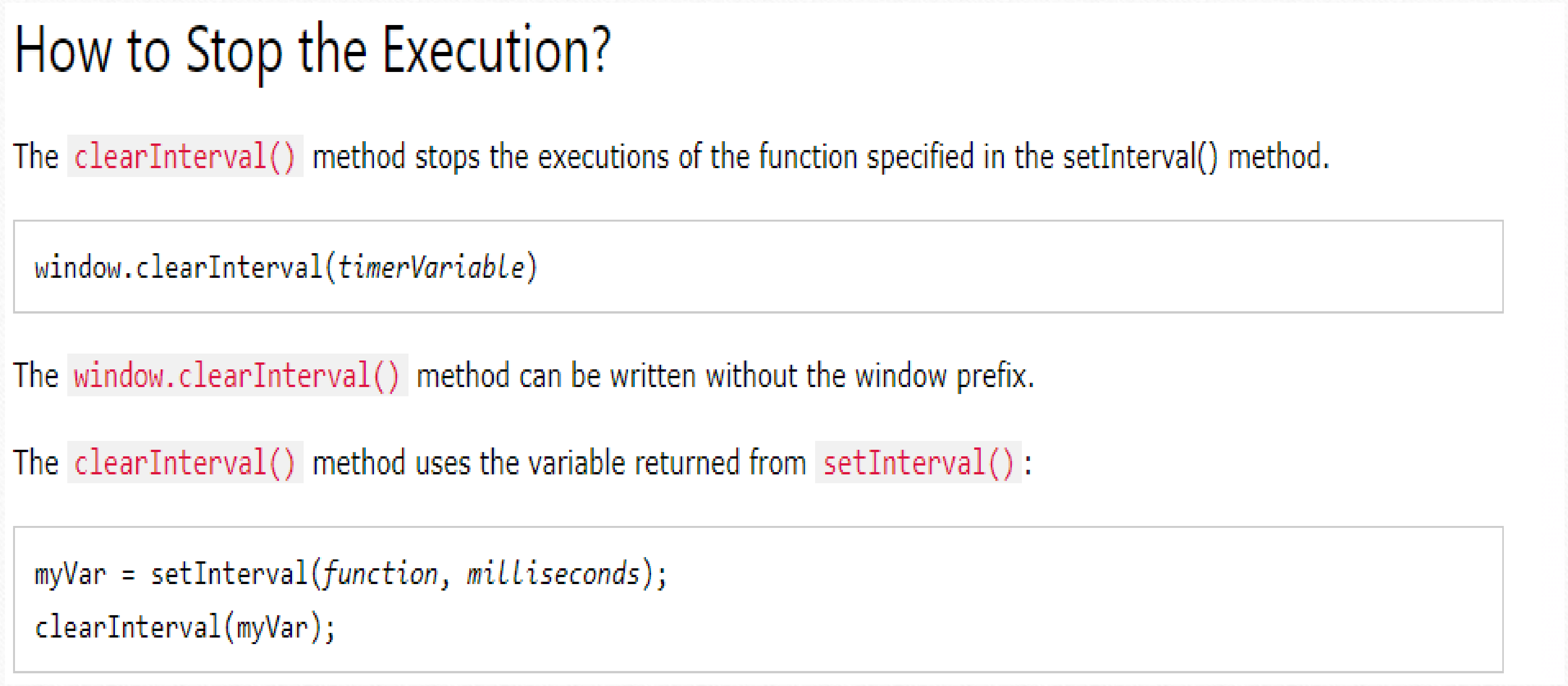
This example executes a function called "myTimer" once every second (like a digital watch).

Example

Display the current time:

```
var myVar = setInterval(myTimer, 1000);

function myTimer() {
  var d = new Date();
  document.getElementById("demo").innerHTML = d.toLocaleTimeString();
}
```



How to Stop the Execution?

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

```
window.clearInterval(timerVariable)
```

The `window.clearInterval()` method can be written without the window prefix.

The `clearInterval()` method uses the variable returned from `setInterval()`:

```
myVar = setInterval(function, milliseconds);
clearInterval(myVar);
```

addEventListener() method

- The addEventListener() method attaches an event handler to the specified element.
- The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
- The addEventListener() method makes it easier to control how the event reacts to bubbling.
- When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

addEventListener() method (cont.)

❑ Syntax:

```
element.addEventListener(event, function, [useCapture]);
```

```
document.getElementById(“b1”).addEventListener("click", myFunction);

function myFunction() {
        alert (“Button Clicked");
}
```

❑ The first parameter is the type of the event (like "click" or "mousedown").
❑ The second parameter is the function we want to call when the event occurs.
❑ The third parameter (optional parameter): is a boolean value specifying whether to use event bubbling or event capturing. Possible values:
  - true - The event handler is executed in the capturing phase
  - false- Default, the event handler is executed in the bubbling phase

Event Bubbling or Event Capturing?

There are two ways of event propagation in the HTML DOM, bubbling and capturing.

Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's "click" event should be handled first?

In *bubbling* the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the <div> element's click event.

In *capturing* the outer most element's event is handled first and then the inner: the <div> element's click event will be handled first, then the <p> element's click event.

With the addEventListener() method you can specify the propagation type by using the "useCapture" parameter:

```
addEventListener(event, function, useCapture);
```

addEventListener() method (cont.)

- You can easily remove an event listener by using the removeEventListener() method.

```
element.removeEventListener("mousemove", myFunction);
```

- Note: The addEventListener() and removeEventListener() methods are not supported in IE 8 and earlier versions and Opera 6.0 and earlier versions. However, for these specific browser versions, you can use the attachEvent() method to attach an event handlers to the element, and the detachEvent() method to remove it.

```
element.attachEvent(event, function);
element.detachEvent(event, function);
```

What is Events?

- ❑Events are actions that respond to user's specific actions.
- ❑Events are controlled in JavaScript using event handlers that indicate what actions the browser takes in response to an event.
- ❑Event handlers are created as attributes added to the HTML tags in which the event is triggered.
- ❑An Event handler adopts the event name and appends the word "on" in front of it.

```
< tag onEvent = "JavaScript commands;">
```

- ❑Thus the "click" event becomes the onClick event handler

Mouse Events

Event handler	Description
onMouseDown	when pressing any of the mouse buttons.
onMouseMove	when the user moves the mouse pointer within an element.
onMouseOut	when moving the mouse pointer out of an element.
onMouseUp	when the user releases any mouse button pressed
onMouseOver	when the user moves the mouse pointer over an element.
onClick	when clicking the left mouse button on an element.
onDblClick	when Double-clicking the left mouse button on an element.
onDragStart	When the user has begun to select an element

Keyboard Events

Event handler	Description
onKeyDown	When User presses a key
onKeyPress	When User holds down a key
onKeyUp	When User a key

Other Events

Event handler	Description
onAbort	The User interrupted the transfer of an image
onBlur	when loosing focus
onFocus	when setting focus
onChange	when the element has lost the focus and the content of the element has changed
onLoad	a document or other external element has completed downloading all the data into the browser
onUnload	a document is about to be unloaded from the window
onError	When an error has occurred in a script.
onMove	when moving the browser window

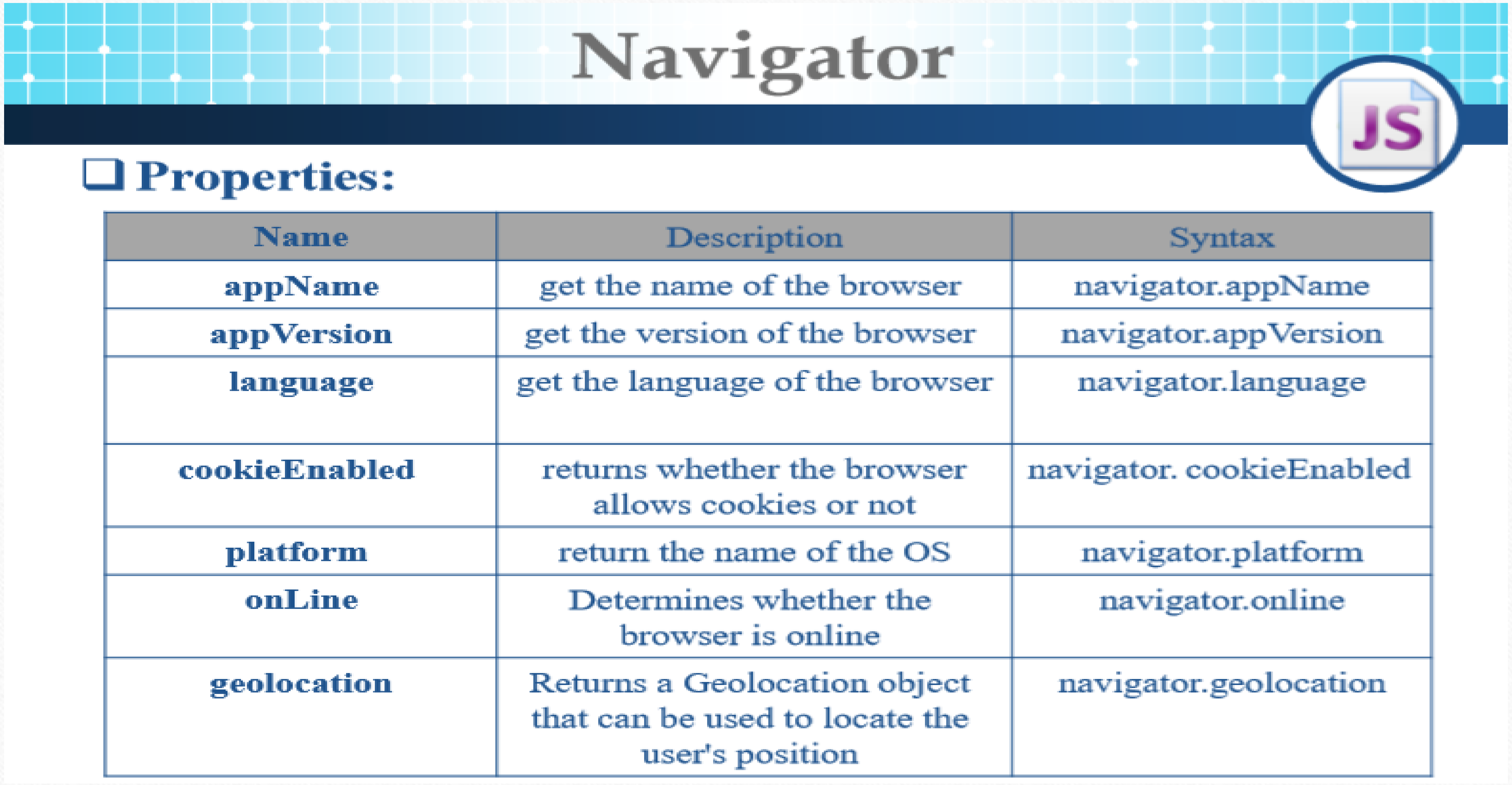
Other Events(Cont.)

Event handler	Description
OnReset	When the user clicks the form reset button
onSubmit	When the user clicks the form submit button
onScroll	When the user adjusts an element's scrollbar
onResize	When the user resizes a browser window
onHelp	When the user presses the F1 key
onselect	When selecting text in an input or a textarea element
onStart	When A marquee element loop begins
onFinish	When a marquee object finishes looping
onSelectStart	When the user is beginning to select an element

Screen(Cont.)

❑ **Properties:**

Name	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



Navigator

- **Properties:**

Name	Description	Syntax
appName	get the name of the browser	navigator.appName
appVersion	get the version of the browser	navigator.appVersion
language	get the language of the browser	navigator.language
cookieEnabled	returns whether the browser allows cookies or not	navigator. cookieEnabled
platform	return the name of the OS	navigator.platform
onLine	Determines whether the browser is online	navigator.online
geolocation	Returns a Geolocation object that can be used to locate the user's position	navigator.geolocation

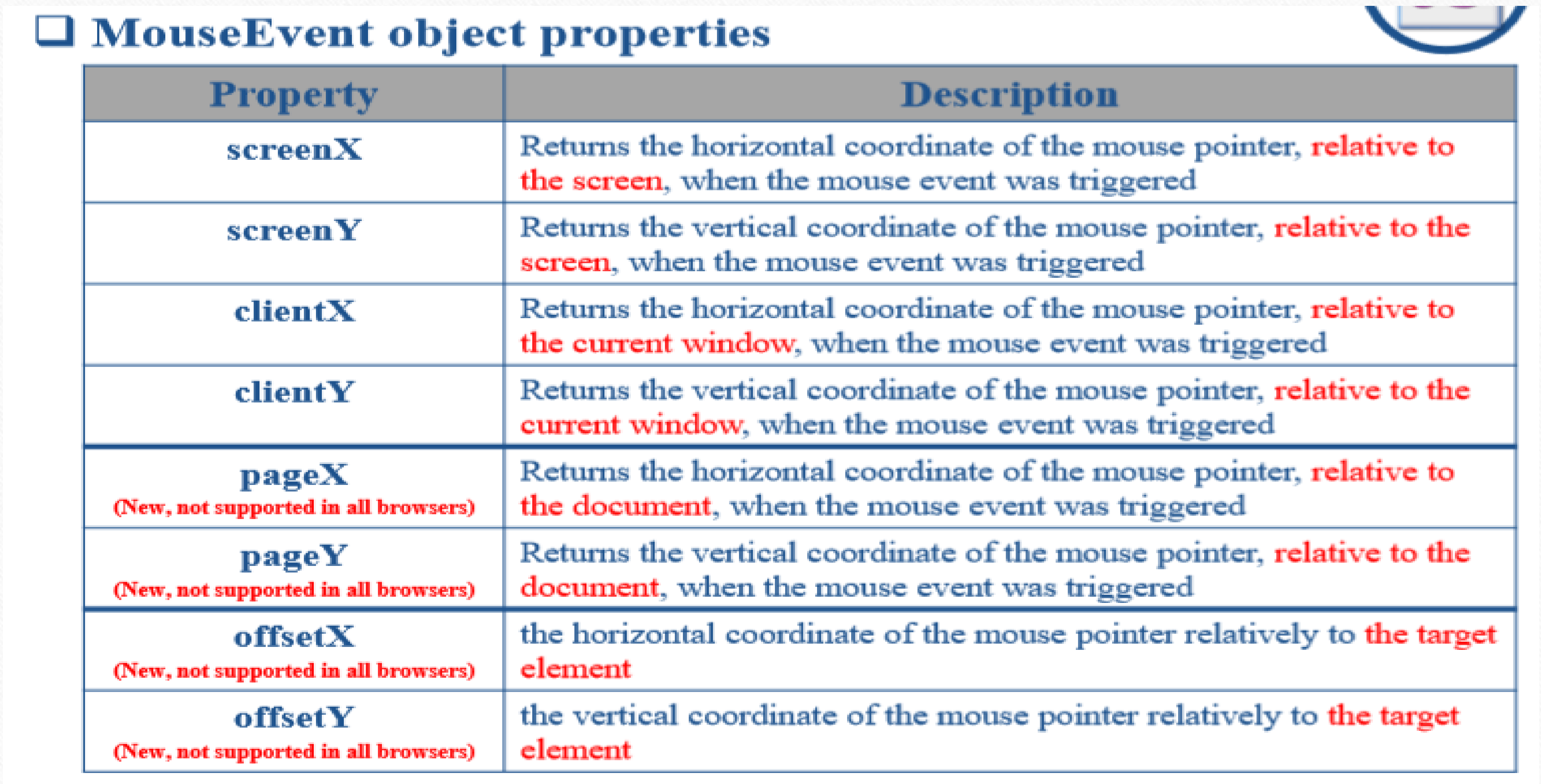
Location (Cont.)

❑ **Properties:**

Name	Description
href	Sets or returns the entire URL
hash	Sets or returns the anchor part (#) of a URL
search	Sets or returns the querystring part of a URL

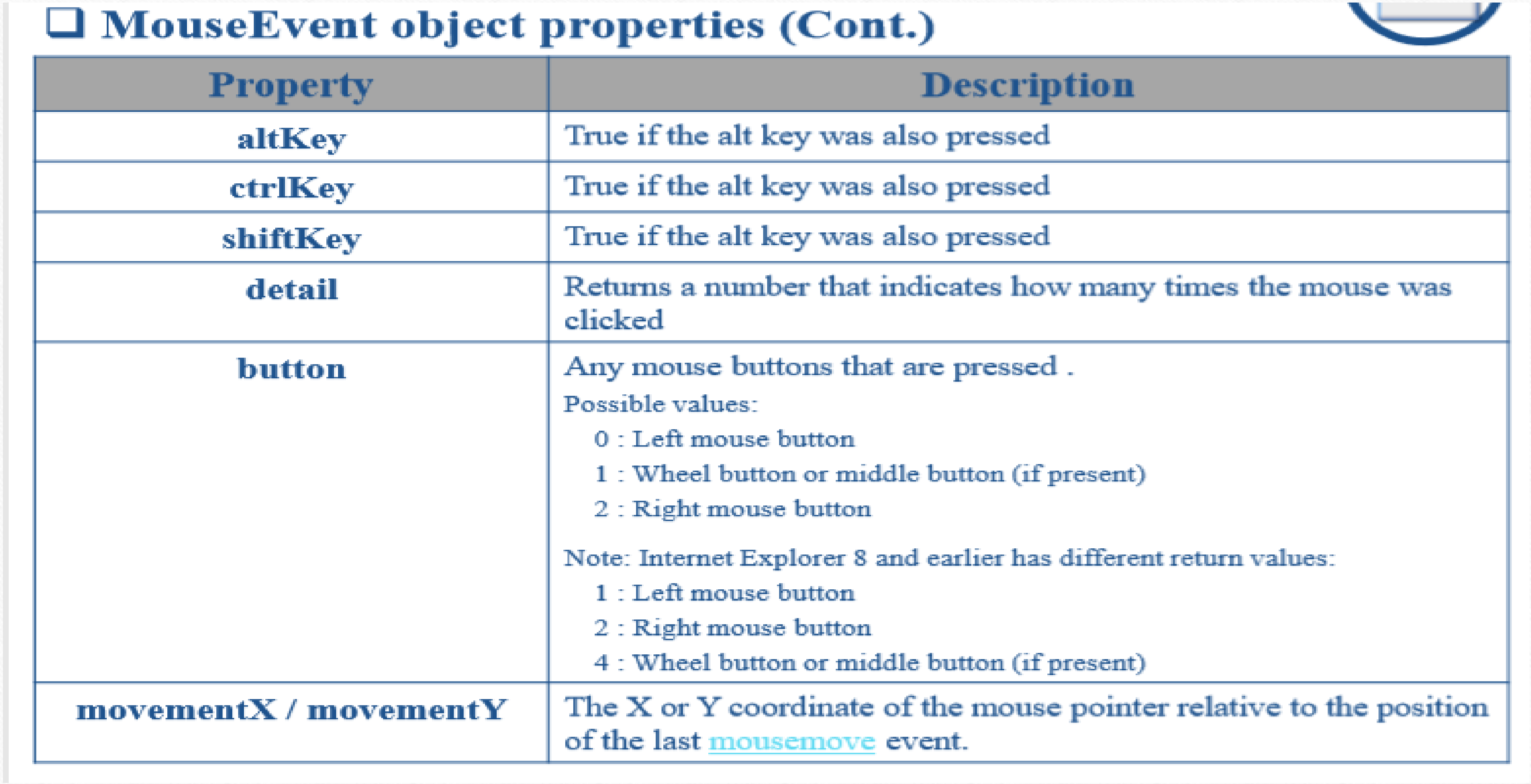
❑ **Methods:**

Name	Description
replace(URL)	Replaces the current document with a new one
assign(URL)	almost the same as replace method. The difference is that it creates an entry in the browser's history list, while replace() doesn’t
reload()	Reloads the current document



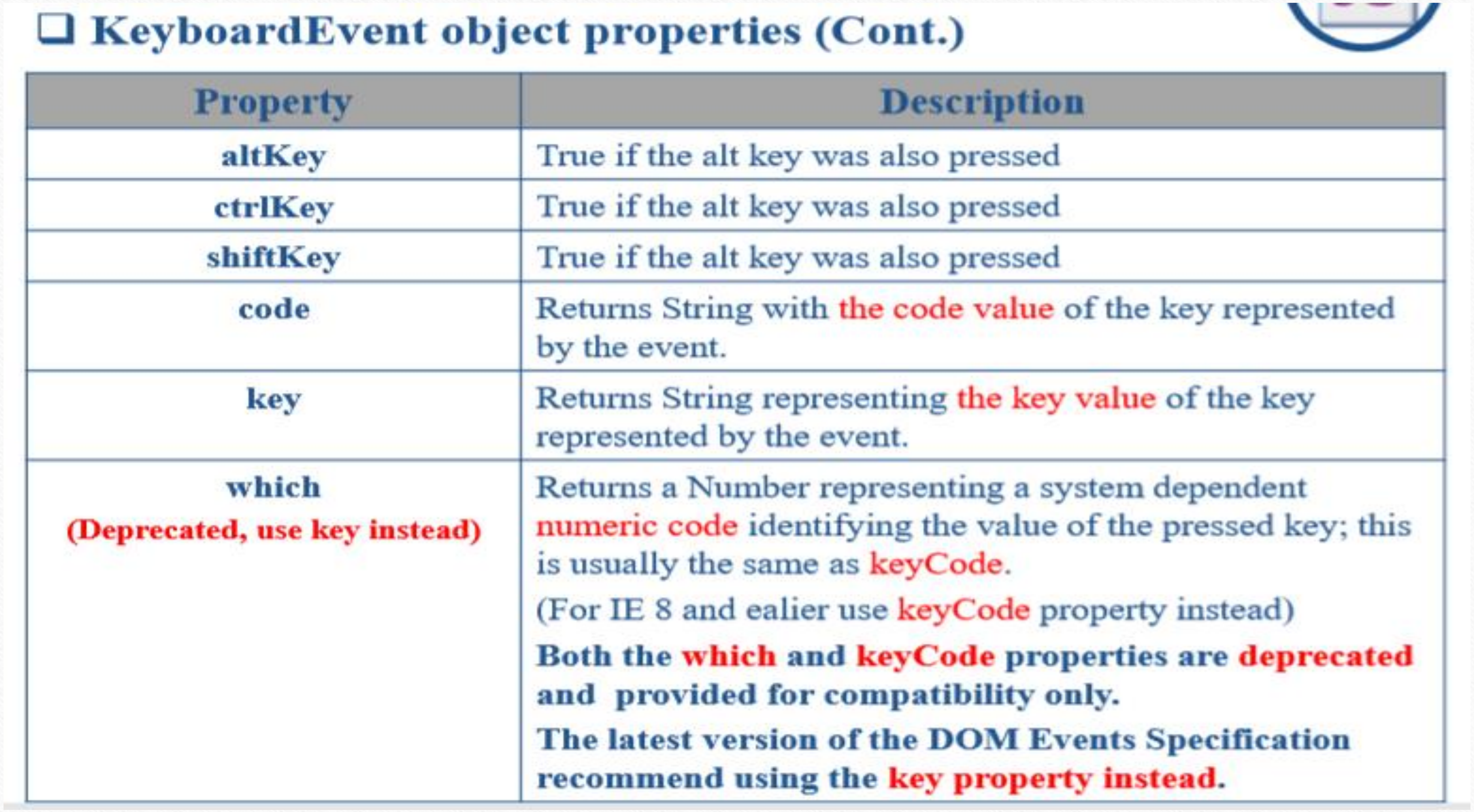
❑ MouseEvent object properties

Property	Description
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when the mouse event was triggered
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when the mouse event was triggered
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
pageX (New, not supported in all browsers)	Returns the horizontal coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
pageY (New, not supported in all browsers)	Returns the vertical coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
offsetX (New, not supported in all browsers)	the horizontal coordinate of the mouse pointer relatively to the target element
offsetY (New, not supported in all browsers)	the vertical coordinate of the mouse pointer relatively to the target element



❑ MouseEvent object properties (Cont.)

Property	Description
altKey	True if the alt key was also pressed
ctrlKey	True if the alt key was also pressed
shiftKey	True if the alt key was also pressed
detail	Returns a number that indicates how many times the mouse was clicked
button	Any mouse buttons that are pressed . Possible values: 0 : Left mouse button 1 : Wheel button or middle button (if present) 2 : Right mouse button Note: Internet Explorer 8 and earlier has different return values: 1 : Left mouse button 2 : Right mouse button 4 : Wheel button or middle button (if present)
movementX / movementY	The X or Y coordinate of the mouse pointer relative to the position of the last mousemove event.



❑ KeyboardEvent object properties (Cont.)

Property	Description
altKey	True if the alt key was also pressed
ctrlKey	True if the alt key was also pressed
shiftKey	True if the alt key was also pressed
code	Returns String with the code value of the key represented by the event.
key	Returns String representing the key value of the key represented by the event.
which **(Deprecated, use key instead)**	Returns a Number representing a system dependent numeric code identifying the value of the pressed key; this is usually the same as keyCode. (For IE 8 and ealier use keyCode property instead) **Both the which and keyCode properties are deprecated and provided for compatibility only.** **The latest version of the DOM Events Specification recommend using the key property instead.**