

CT053-3-1

Fundamentals of Web Design & Development



A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

JavaScript Part 3

JavaScript Browser Objects

- Screen Object
- Location Object
- Timing
- Cookies



JavaScript Window Screen

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

Properties	Description
screen.width	Returns the width of the visitor's screen in pixels.
screen.height	Returns the height of the visitor's screen in pixels.
screen.availWidth	Returns the width of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.
screen.availHeight	Returns the height of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.
screen.colorDepth	Returns the number of bits used to display one color.
screen.pixelDepth	Returns the pixel depth of the screen.

```
<script>
document.getElementById("demo").innerHTML =
"Screen width is " + screen.width;

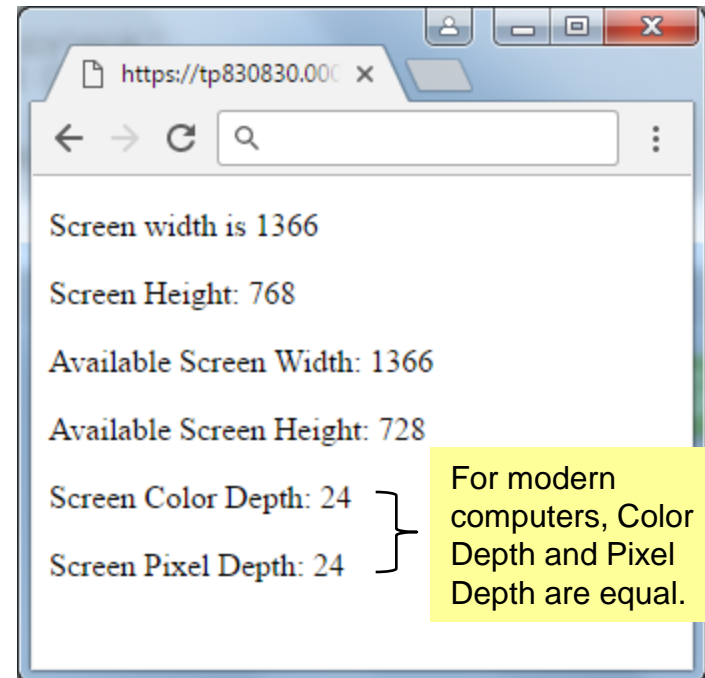
document.getElementById("demo2").innerHTML =
"Screen Height: " + screen.height;

document.getElementById("demo3").innerHTML =
"Available Screen Width: " + screen.availWidth;

document.getElementById("demo4").innerHTML =
"Available Screen Height: " + screen.availHeight;

document.getElementById("demo5").innerHTML =
"Screen Color Depth: " + screen.colorDepth;

document.getElementById("demo6").innerHTML =
"Screen Pixel Depth: " + screen.pixelDepth;
</script>
```

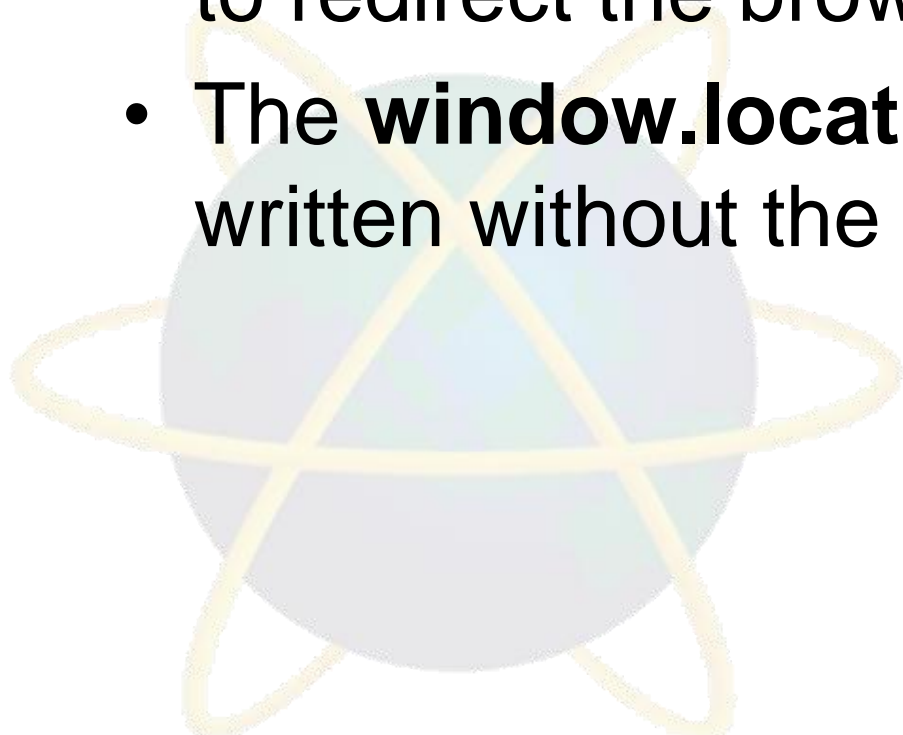


Check your mobile phone
screen info by visiting this URL

<http://bit.ly/checkyourscreeninfo>

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



window.location usage

<http://www.apu.edu.my/explore-apu/university/index.html>

Method	Description	Return
window.location.href	returns the href (URL) of the current page	http://www.apu.edu.my/index.html
window.location.hostname	returns the domain name of the web host	www.apu.edu.my
window.location.pathname	returns the path and filename of the current page	/explore-apu/university/index.html
window.location.protocol	returns the web protocol used (http: or https:)	http:
window.location.assign()	loads a new document	loads a new document on current document

JavaScript 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.
- Methods:
 - `history.back()` - same as clicking back in the browser
 - `history.forward()` - same as clicking forward in the browser

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.

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.

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).

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

```
<script>
```

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

```
function myTimer() {  
    var d = new Date();
```

```
document.getElementById("demo").innerHTML  
= d.toLocaleTimeString();  
}</script>
```

(a)

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

```
<script>
```

```
var d = new Date();
```

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

```
</script>
```

(b)

What is the output of (a) and (b) ?

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);
```

JavaScript 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":
 - When a user visits a web page, his name can be stored in a cookie.
 - Next time the user visits the page, the cookie "remembers" his name.

Create a Cookie with JavaScript

- JavaScript can create, read, and delete cookies with the **document.cookie** property.
- With JavaScript, a cookie can be created like this:

```
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  
2017 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  
2017 12:00:00 UTC; path=/";
```

Read a Cookie with JavaScript

- With JavaScript, cookies can be read like this:

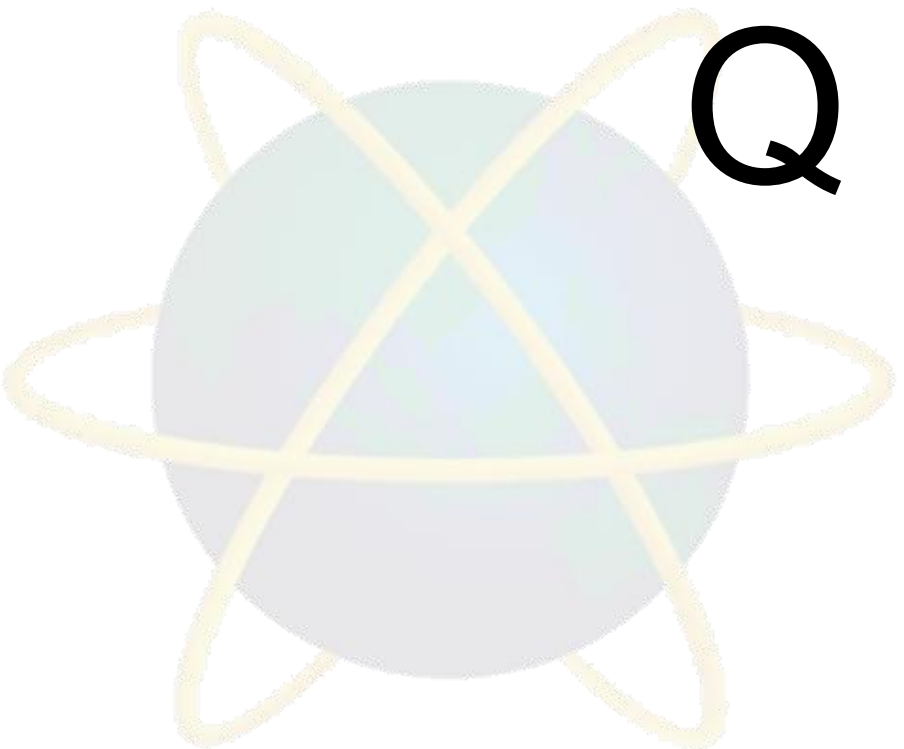
```
var x = document.cookie;
```

- document.cookie will return all cookies in one string much like:

```
cookie1=value; cookie2=value; cookie3=value;
```


Quick Exercise

1. By using JavaScript Window History method, write a HTML and JavaScript to demonstrate “Previous” and “Next” hyperlink for a web page.
2. Write a JavaScript code snippet to check the availability of cookie. If cookie is available, redirect the page to “home.html” or else “error.html”.



Q & A