UNIT IV : COOKIES & BROWSER DATA

Cookies– Basic Of Cookies

- A cookie is a **small text file that lets you store a small amount of data on the user's computer.**
- They are typically used for **keeping track of information such as user preferences that the site** can retrieve to personalize the page when **user visits the website next time**.
- Cookies are an old **client-side storage mechanism** that was originally designed for use by **server-side scripting languages**.
- Cookies can also be created, accessed, and modified directly using JavaScript, but the process is little bit complicated and messy.

Cookies– Basic Of Cookies

- Cookies are a plain text data record of 5 variable-length fields –

1. **Expires** – The date the **cookie will expire**.
2. **Domain** – The **domain name of your site**.
3. **Path** – The path to the **directory or web page that set the cookie**.
4. **Secure** – If this field contains the word "secure", then the cookie may only be **retrieved with a secure server**.
5. **Name=Value** – Cookies are **set and retrieved in the form** of key-value pairs

Cookies– Basic Of Cookies

- **Types of Cookies**

1. **Session Cookies** – These cookies are **temporary which are erased when the user closes the browser.** Even if the user logs in again, a new cookie for that session is created.
2. **Persistent cookies** – These cookies **remain on the hard disk drive unless user wipes them off or they expire**. The Cookie's expiry is dependent on how long they can last.

- JavaScript can create, read or delete a cookies using **document.cookie property**.

Creating Cookies

- Creation of Cookies is a simple techniques. For creating a cookie we need to assign value to **window.document.cookie.**
- **document.cookie = "cookiename=cookievalue"**
- Thus the name value pair separated by = sign & terminated by a delimiter like semicolon(;) the cookie can be assigned to **document.cookie.**

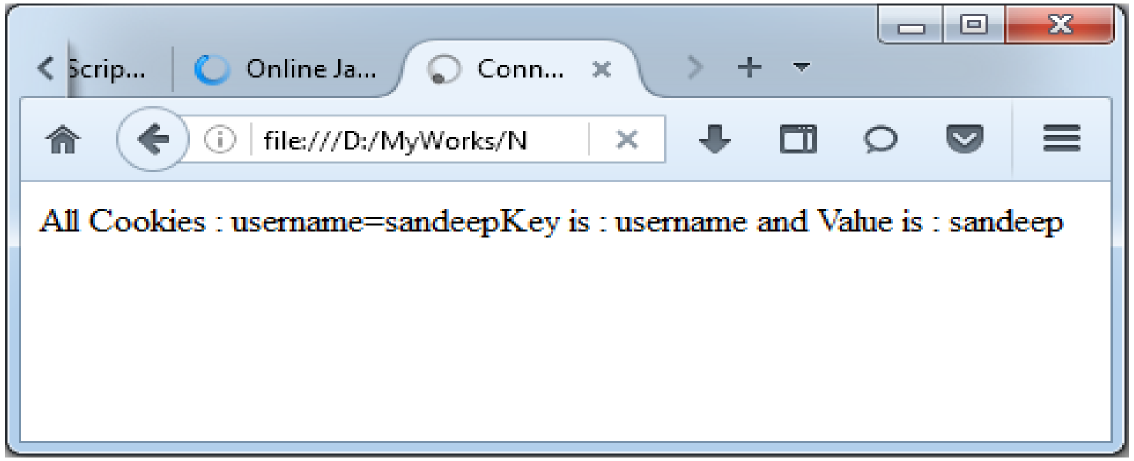
Creation of Cookies Example:

Create Cookie.docx

Reading A Cookie Value

- To create cookie & then read the value of the cookies created.
- Reading a cookie is just as simple as writing one, because the value of the document.
- The **document.cookie** string will keep a list of **name=value** pairs separated by semicolons, where **name** is the name of a cookie and value is its string value.
- Using **split()** function the string of cookies is break into key and values.

```
<html>
  <head>
      <script type = "text/javascript">
      <!--
        function ReadCookie()
        {
          var allcookies = document.cookie;
          document.write ("All Cookies : " + allcookies );
          cookiearray = allcookies.split(';');
          for(var i=0; i<cookiearray.length; i++) {
            name = cookiearray[i].split('=')[0];
            value = cookiearray[i].split('=')[1];
            document.write ("Key is : " + name + " and Value is : " + value);
          }
        }
     </script>      </head>
  <body>
       <form name = "myform" action = "">
       <p> click the following button and see the result:</p>
       <input type = "button" value = "Get Cookie" onclick = "ReadCookie()"/>
    </form>     </body>    </html>
```



All Cookies : username=sandeepKey is : username and Value is : sandeep

Deleting Cookies

- Cookies get deleted automatically when the **browser session ends or its expiration date is reached.**
- By **setting expiry date** we can delete the cookie.

```
<html>    <head>
      <script type = "text/javascript">
          function WriteCookie()
        {
          var now = new Date();
          now.setMonth( now.getMonth() -1 );
          cookievalue = escape(document.myform.customer.value) + ";"
          document.cookie="name=" + cookievalue;
          document.cookie = "expires=" + now.toUTCString() + ";"
          document.write ("Setting Cookies : " + "name=" + cookievalue );
        }
    </script>    </head>
  <body>
      <form name = "myform" action = "">
      Enter name: <input type = "text" name = "customer"/>
      <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
    </form>    </body>   </html>
```

Setting The Expiration Date Of Cookie

- We can extend the life of a **cookie beyond the current browser session** by **setting an expiration date and saving the expiry date within the cookie**.
- This can be done by setting the **‘expires’** attribute to a date and time.

```
<html>    <head>
      <script type = "text/javascript">
          function WriteCookie()
        {
          var now = new Date();
          now.setMonth( now.getMonth() +1 );
          cookievalue = escape(document.myform.customer.value) + ";"
          document.cookie="name=" + cookievalue;
          document.cookie = "expires=" + now.toUTCString() + ";"
          document.write ("Setting Cookies : " + "name=" + cookievalue );
        }
    </script>    </head>
  <body>
      <form name = "myform" action = "">
      Enter name: <input type = "text" name = "customer"/>
      <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
    </form>    </body>   </html>
```

Browser

- It is possible to **open a new browser window from a currently running JavaScript**. One can determine the size, location of this window, toolbar, scroll bar or any other style that normally the browser windows have.
- Once the new browser window is set up it is possible **to change the contents within that window dynamically.**

Opening A Window

- It is possible to open a new window from a JavaScript by simply clicking a button.
- For that purpose the window object is used. This window object has various useful properties & methods.
- To open a new windows we use open() method of window object.

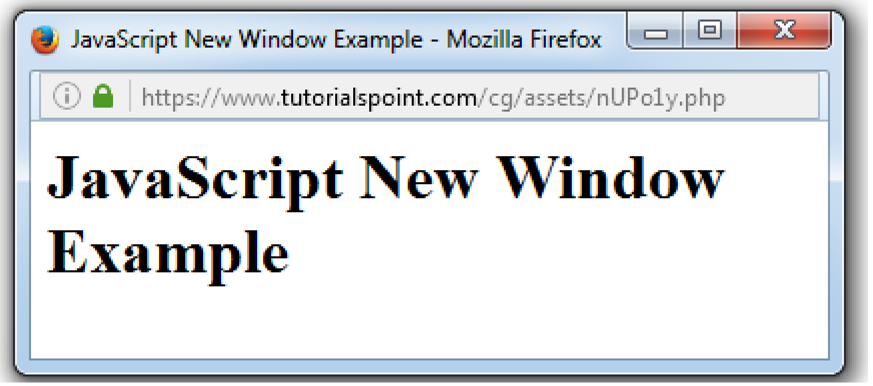
Opening A Window

- Syntax: **window.open(url, name, style);**

- **url:** An URL to load into the new window.
- **Name:** A name of the new window. Each window has a window.name, and here we can specify which window to use for the popup. If there's already a window with such name – the given URL opens in it, otherwise a new window is opened.
- **style:** The style of window includes various parameters such as menubar, toolbar, location, status, resizable, scrollbars, height & width of window .

```
<html>
<head>
 <title>JavaScript New Window Example</title>
</head>
<script type="text/javascript">
function poponload()
{
   testwindow = window.open("", "mywindow",
   "location=1,status=1,scrollbars=1,width=100,height=100");
   testwindow.moveTo(0, 0);
 }
 </script>
<body onload="javascript: poponload()">
<h1>JavaScript New Window Example</h1>
</body>
</html>
```

JavaScript New Window Example

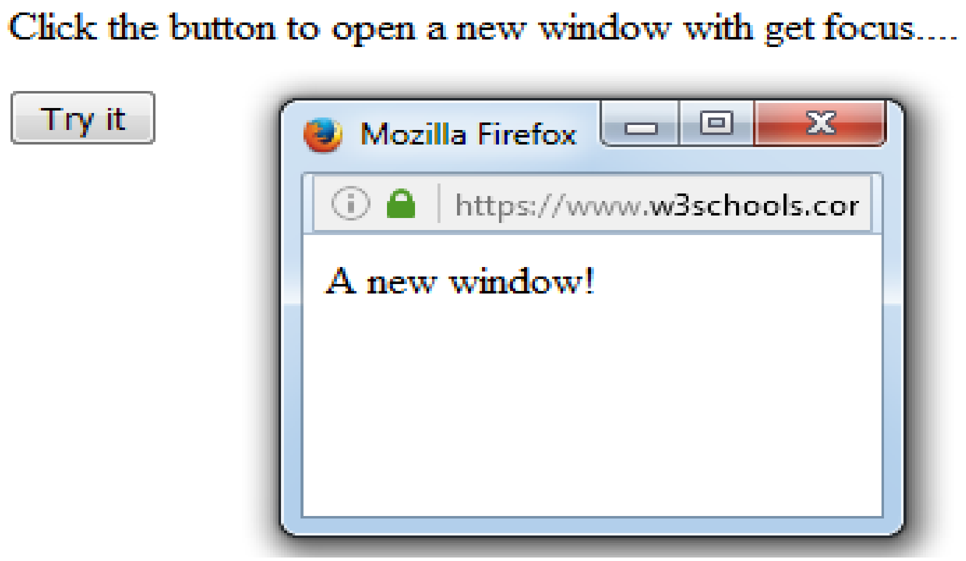

JavaScript New Window Example - Mozilla Firefox
https://www.tutorialspoint.com/cg/assets/nUPo1y.php

JavaScript New Window Example

Giving The New Window Focus

- The **focus**() method sets **focus** to the current **window**.
- This method makes a request to bring the current **window** to the foreground.

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to open a new window with get focus....</p>
<button onclick="myFunction()">Try it</button>
<script>
        function myFunction()
        {
         var myWindow = window.open("", "", "width=200,height=100");
        myWindow.document.write("<p>A new window!</p>");
        myWindow.focus();
        }
</script>
</body>
</html>
```

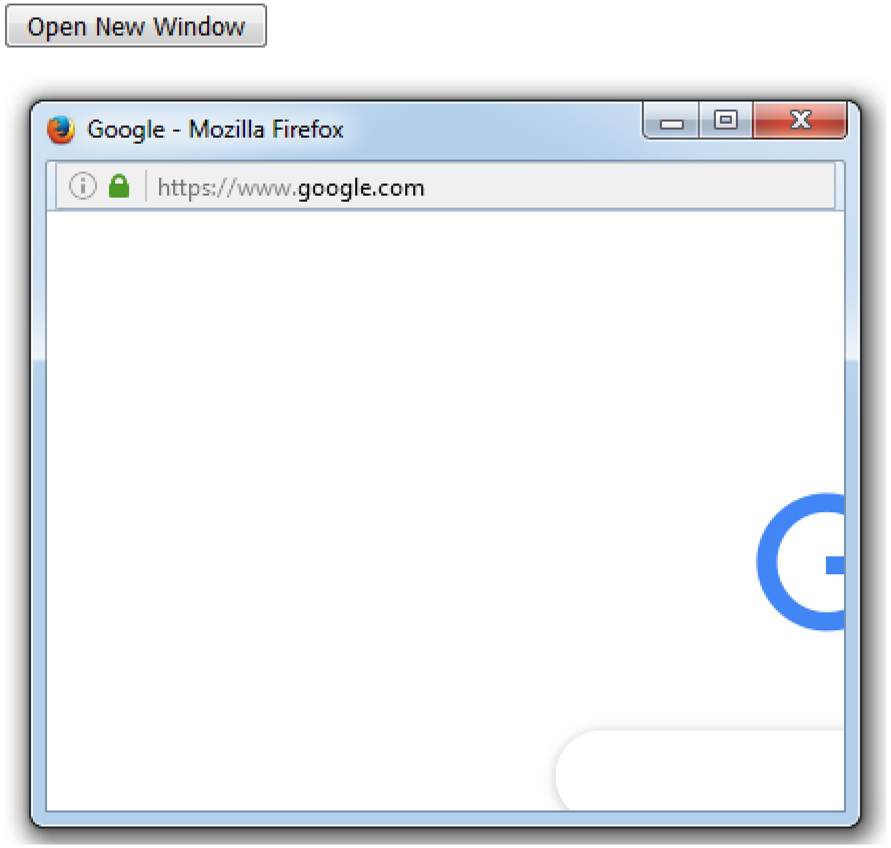

Click the button to open a new window with get focus....

A new window!

Window Position

- We can set the desired position for the window. Using the **left & top attributes values** the window position can be set.

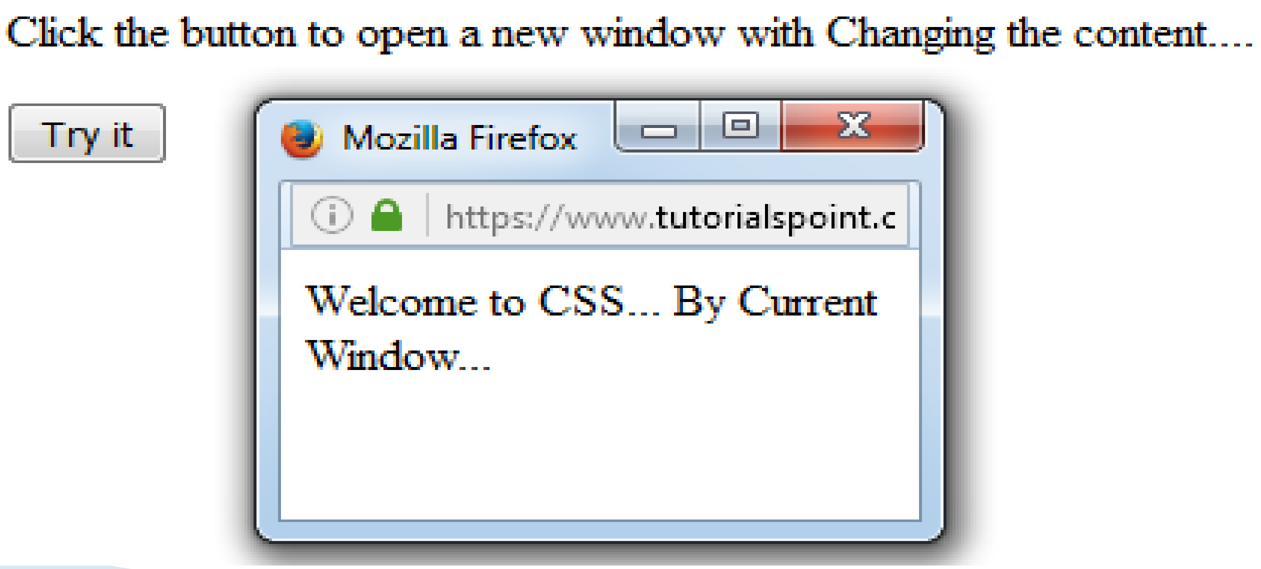
```
<HTML>
<HEAD>
<SCRIPT language="JavaScript">
function new_win()
{
window.open("http://www.google.com","mywin","width=400,height=300
, screenX=50,left=50,screenY=50,top=50");
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="myform">
<INPUT TYPE="button" value="Open New Win d
</FORM>
</BODY>
</HTML>
```



Changing The Content Of Window

- By writing some text to the newly created window we can change the contents of a window.

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to open a new window with Changing the content....</p>
<button onclick="myFunction()">Try it</button>
<script>
          function myFunction()
          {
           var myWindow = window.open("", "", "width=200,height=100");
           myWindow.document.write("<p>Welcome to CSS... By Current
Window...</p>");
          }
</script>
</body>
</html>
```



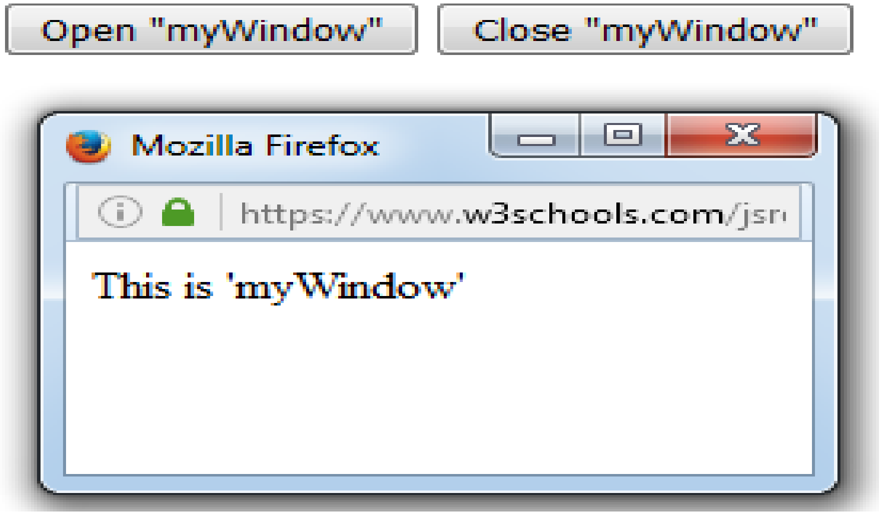
Click the button to open a new window with Changing the content....

Welcome to CSS... By Current Window...

Closing A Window

- he **close** method closes only windows opened by **JavaScript** using the open method.

```
<!DOCTYPE html>  <html> <body>
<button onclick="openWin()">Open "myWindow"</button>
<button onclick="closeWin()">Close "myWindow"</button>
<script>
var myWindow;
function openWin()
{
  myWindow = window.open("", "myWindow",
"width=200,height=100");
myWindow.document.write("<p>This is 'myWindow'</p>");
}
function closeWin()
 {
  myWindow.close();
}
</script> </body> </html>
```



This is 'myWindow'

Scrolling A Web Page

- We can scroll **horizontally or vertically using ScrollTo() function**.

```
<html> <head>
<script type="text/javascript">
function scrollWindow()
{
  window.scrollTo(100,0)
}
</script> </head>
<body> <form>
<input type="button" onclick="scrollWindow()" value="Scroll">
</form>
<br><br><br><br><br><br>
1
<br><br><br><br><br><br><br><br><br><br>
2
<br><br><br><br><br><br><br><br><br><br>
3
<br><br><br><br><br><br><br><br><br><br>
4
</body> </html>
```

Multiple Window At Once

- It is possible to open multiple windows at a time. By using **open() method.**

```
<html>
<head>
<script type="text/javascript">
function open_win()
{
   window.open("http://www.java2s.com/")
   window.open("http://www.google.com/")
}
</script>
</head>
<body>
<form>
<input type=button value="Open Windows" onclick="open_win()">
</form>
</body>
</html>
```

Creating A Web Page In New Window

- We can create a web page using the window object with the help of write method.
- Inside the write() we have to write   the content of the web page with help of the html elements such as <head>,<body>,<h1>.

```
<HTML>
<HEAD>
<SCRIPT language="JavaScript">
<!--
function new_win()
{
   var mywin=window.open("","mywin","width=400,height=300")
   mywin.document.write("<html>");
   mywin.document.write("<head>");
   mywin.document.write("<title>WEB SITE DEMO</title>");
   mywin.document.write("</head>");
   mywin.document.write("<body>");
   mywin.document.write("<h2>This is a new Web Page</h2>");
   mywin.document.write("<h3>Welcome User...!!!!</h2>");
   mywin.document.write("</body>");
   mywin.document.write("</html>");

}
</SCRIPT> </HEAD>
<BODY>
<FORM name="myform">
<INPUT TYPE="button" value="Create Web
Page" onClick="new_win()">
</FORM> </BODY> </HTML>
```

JavaScript In URLs

- JavaScript code can be included on the client side.
- JavaScript can be specified in **URL using the pseudo-protocol specifier.**
- This special protocol type specifies that the **body of the URL is arbitrary JavaScript code to be interpreted** by the JavaScript interpreter.
- For Example: We can type the following code in URL bar:

  javascript:alert("Hello World!")
- If the JavaScript code in a JavaScript: URL contains multiple statements, the statements must be separated from one another by semicolons. Such a URL might look like the following:
- For Example: **javascript:var now = new Date(); "The time is:" + now ;**

JavaScript In URLs

- JavaScript has several security issues that need attention.
- One of the most common JavaScript security vulnerabilities is **Cross-Site Scripting (XSS)**. Cross-Site Scripting vulnerabilities enable attackers to manipulate websites to return malicious scripts to visitors. These malicious scripts then execute on the client side in a manner determined by the attacker. This vulnerability may cause the user data theft, account tampering and so on.
- **Cross-Site Request Forgery(CSRF)** is another issue in JavaScript Cross-Site Request Forgery involves taking over a impersonating a user's browser session by hijacking the session cookies. CSRF attacks can trick the users into executing malicious actions the attacker wants unauthorized actions on the website.

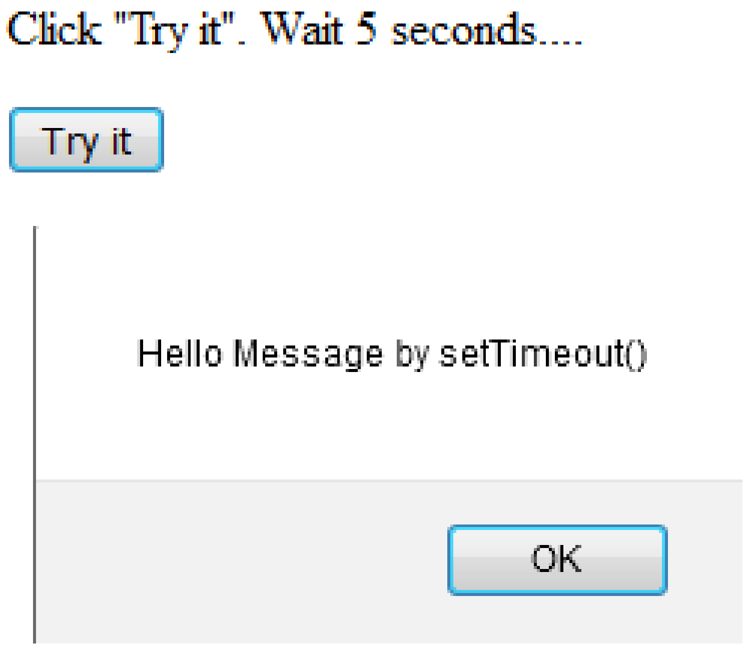
Timer s

- 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:

1. **setTimeout(function, milliseconds)**

- Executes a function, after **waiting a specified number of milliseconds.**
- The first parameter is a function to be executed.
- The second parameter indicates the number of milliseconds before execution.

```
<!DOCTYPE html>
<html>
<body>
<p>Click "Try it". Wait 5 seconds....</p>
<button onclick="setTimeout(myFunction, 5000);">Try
it</button>
<script>
function myFunction()
{
  alert('Hello Message by setTimeout()');
}
</script>
</body>
</html>
```


Click "Try it". Wait 5 seconds....

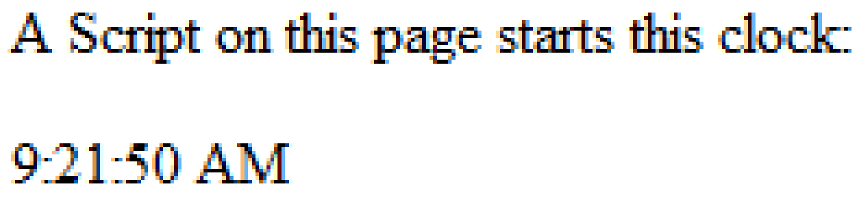
Hello Message by setTimeout()

Timers

2. **setInterval(function, milliseconds)**

- Same as setTimeout(), but repeats the execution of the function continuously.
- The first parameter is the function to be executed.
- The second parameter indicates the length of the time-interval between each execution.

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



```
A Script on this page starts this clock:

9:21:50 AM
```

Browser Location

- The **window.location** object is **useful for finding out the current location or path of the web page.**
- Properties of window.location as follow:

1. **window.location.hostname**
2. **window.location.pathname**
3. **window.location.protocol**
4. **window.location.assign**

```
<!DOCTYPE html>
<html>
<body>
   <p id="ID"></p>
<script>
document.getElementById("ID").innerHTML= "This web page
is at path: "+window.location.pathname;
</script>
</body>
</html>
```

Output: **This web page is at path: /cg/assets/PM7hUK.php**

```
<!DOCTYPE html>
<html>
<body>
   <p id="ID"></p>
<script>
document.getElementById("ID").innerHTML= "This web page
is using the protocol: "+window.location.protocol;
</script>
</body>
</html>
```

Output: **This web page is using the protocol: https:**

Browser History

- The **window.history** object is used for **displaying browser history**.
- There are two methods window.history as follow:

1. **window.history.back() :** This method loads the previous URL in the history list.
2. **window.history.forward() :** This method loads the next URL in the history list.

```
 <html>
<head>
<script>
function MoveBack()
{
   window.history.back();
 }
function MoveForward()
{
   window.history.forward();
 }
</script>
</head>
<body>
   <form name= "form1">
      <input type = "button" value ="Back" onclick="MoveBack()">
      <input type = "button" value ="Forward" onclick="MoveForward()">
</form>   </body>   </html>
```

Thank you