**Cookies**

Cookies are small chunks of data that are saved by your browser. A web page can cause your browser to save data in a cookie and the same or another page can later retrieve the saved data. The JavaScript cookie is a document object with the following parameters:

* *name*: the name of the cookie
* *expires*: the date/time when the cookie expires automatically
* *path*: the path of a cookie
* *domain*: the name of the server that created the cookie
* *secure*: whether to use encryption to read/set cookie

Since cookies represent data stored automatically by a web page on your computer they might pose certain security and privacy risks. The data could contain a virus, for example, or cookies can be (and are) are used to identify and follow individual access to web pages. For example, when you visit page X, you could receive a cookie with a unique identifier. When you later return to the page, it can check the value of the cookie and know whether you visited the page before, and when. It can even identify you as a unique user but it does not know your actual identity (name, address, etc).

To help protect your computer, cookies follow some rules:

* Only small amounts of data can be stored in a cookie
* Cookies are available via JavaScript only to the domain used when the cookie was created
* Cookies are available only to files in the same directory the cookie was created in (to make a cookie available to all files use as path “/”)
* Cookies can be manually manipulated in a web browser. In Firefox:
  + Tools -> Options -> Privacy -> Show Cookies will show all cookies stored
  + Tools -> Options -> Privacy lets you decide whether browser accepts cookies
  + Tools -> Clear Private Data lets you delete all stored cookies

**How to set and show a cookie:**

To set a cookie, you assign an appropriate value to “document.cookie”. It will automatically add your cookie to other cookies from your domain that might exist already to form a string of cookies. The cookie must follow a certain format. At a minimum it is:

name=value;expires=date;path=mypath

To show cookies, use for example:

alert("Cookies: " + document.cookies);

You should create a 'showCookies' function and create a button to call it. Note that showCookies might not show any cookies. That does not mean there are no cookies, only that there are no cookies *from our current domain*. To set a typical cookie you'd use:

document.cookies = "name='text';expires='Sun, 14 Jun 2009 16:16:42 GMT';path='/'"

Let's create a function setCookie to accept some input parameters and create a cookie accordingly as follows:

function setCookie(name, value, expireDays)

{

var expires = new Date();

expires.setDate(expires.getDate() + expireDays);

var myCookie= name + "=" + escape(value) +

";expires=" + expires.toGMTString() +

";path=/";

document.cookie = myCookie;

alert("Set cookie:" + myCookie);

}

A few things to note:

* Our function requires three inputs: the name of the cookie, the value of the cookie, and the number of days from today when the cookie should expire. Other parameters are possible as well
* We create a new date, add the number of days to it, and use the result to define an appropriate expiration date object, which we convert to GMT to ensure it’s in the format cookies like.
* To set the name=value we use the escape function for the value. That function is a JavaScript function to turn an arbitrary string into a safe and proper http string: no ‘funny’ escape characters for potential hanky-panky, replace special characters by proper HTML escape characters (e.g. a space by %20, a semicolon ; by %3B, a < by %3C, > by %3E, etc). That avoids the possibility of planting a virus in a cookie in case the value is typed by a user, and to use characters with special meanings such as < and >
* We use the path=/ to allow access to the cookie from all directories by default

Next, create a button and to input fields to set cookies with specific values. Set the cookie, then show all cookies. Note that document.cookie will only show the name=value combination of our cookies, not the details like expiration date, path, etc.

**How to delete a cookie:**

To delete a cookie is easy: we simply set the name of the cookie to an empty string and expire it yesterday. Even better, we can use the setCookie method to do the work for us:

function delCookie(name)

{

alert("Delete Cookie: " + name);

setCookie(name, "", -1);

}

Create a button and an input field to delete a named cookie.

**How to retrieve the value of a cookie:**

This is a little trickier: if document.cookie is not empty it consists of “name=value” pairs, *separated by semi-colons*. So if the list is not empty we split it at the semicolons into an array, each entry now being a name=value pair. Then we go through the array, split each entry at the equal sign and check the name until we find the right one.

**NOTE**: Here we reap the benefit of having used the escape function to set a cookie – why (hint: think semicolon)

Fortunately, splitting is easy thanks to the build-in split function that can be invoked on strings. It splits a string at the specified character and returns an array corresponding to the parts of the original string. For example if s=”one;two;three” then s.split returns the array [“one”, “two”, “three”]. A little complication arises because cookies are actually separated by a semi-colon *and a space*. To remove extra leading spaces, we define a new function removeLeadingSpace. Also, when we finally find the value of the cookie, we ‘unescape’ it before returning it, to undo the changes of the ‘escape’ function we used to set the cookie’s value. Here is the code:

function removeLeadingSpace(s)

{

while ((s != null) && (s.length > 0) && (s.charAt(0) == ' '))

s = s.substring(1,s.length);

return s;

}

function getCookie(name)

{

if ((document.cookie == null) || (document.cookie == ""))

{

return "";

}

else

{

var cookies = document.cookie.split(';');

for (var i = 0; i < cookies.length; i++)

{

var cookie = cookies[i].split('=');

if (removeLeadingSpace(cookie[0]) == name)

{

return unescape(cookie[1]);

}

}

return "";

}

}

Of course we should add a button and an input field to test our new function. We have, actually, created three (or four) universally useful functions to work with cookies. We can put these functions in a separate file, clean up the alert calls, add comments for better readability, and we can then use them in any of our pages dealing with cookies.

**Loading JavaScript from an external file**

In other words, create a new file of type JavaScript, put the following code into it (the above functions minus the alert calls plus comments), and save as “cookiecutters.js” in a directory of your choice on your web site:

**File cookiecutters.js**

/\* JavaScript library to handle Cookies. Provides the following functions:

\*

\* removeLeadingSpace(s) removes leading spaces from string s

\* setCookie(name, vale, expireDays) sets named cookie with given value

\* getCookie(name) returns value of named cookie

\* delCookie(name) deletes named cookie

\* showCookies() show all cookies in an alert box

\*

\* Author: Bert Wachsmuth

\* Version: March 2011

\*/

/\*

\* Returns the string s without leading spaces.

\*/

function removeLeadingSpace(s)

{

while ((s != null) && (s.length > 0) && (s.charAt(0) == ' '))

s = s.substring(1,s.length);

return s;

}

/\*

\* Creates a cookie with the given name, value, and days after today to expire

\*/

function setCookie(name, value, expireDays)

{

if (!((name == null) || (name == "")))

{

var expires = new Date();

expires.setDate(expires.getDate() + expireDays);

var myCookie= name + "=" + escape(value) +

";expires=" + expires.toGMTString() +

";path=/";

document.cookie = myCookie;

}

}

/\*

\* Shows all cookies from the current host in an alert box

\*/

function showCookies()

{

alert("My cookies: " + document.cookie);

}

/\*

\* Returns value of the named cookie or an empty string if cookie not found

\*/

function getCookie(name)

{

if ((document.cookie == null) || (document.cookie == ""))

{

return "";

}

else

{

var cookies = document.cookie.split(';');

for (var i = 0; i < cookies.length; i++)

{

var cookie = cookies[i].split('=');

if (removeLeadingSpace(cookie[0]) == name)

{

return unescape(cookie[1]);

}

}

return "";

}

}

/\*

\* Deletes the cookie with the given name if possible

\*/

function delCookie(name)

{

setCookie(name, "", -1);

}

**Homework:** Use the cookie-manipulating functions contained in cookiecutter.js to create a page that has links to switch styles between "normal" and "alternate". The page should use cookies to preserve the currently selected styles so that if the user loads the page at a later time, the last style selected will be used for the page.