Cookies and Other Client-Side Storage Techniques

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy

HTML5 Feature Areas

Semantics

CSS3

Multimedia

Graphics & 3D

Device Access

Performance

Offline & Storage

Connectivity

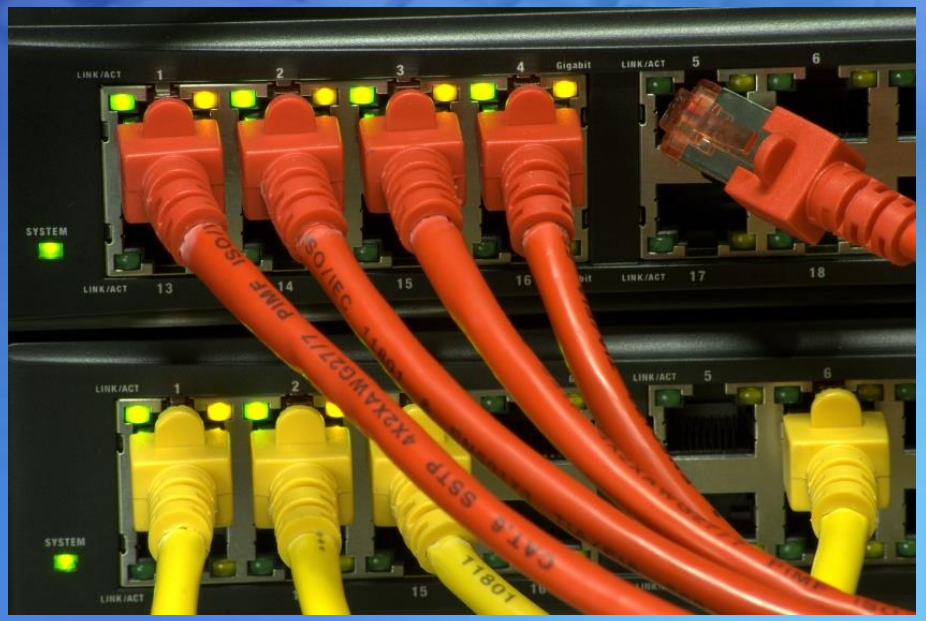
Offline and Storage

Offline and Storage (Cont.)

Source : Google,Introduction to HTML5, Oct. 7, 2009

Offline Web Applications

- Use complete web sites (documentation sets) in offline mode.
- Cache pages that have not been visited yet.
- Browsers cache data in an Application Cache.
- HTML5 allows online and offline detection.
- Allows prefetching of site resources.
  (can speed up pages)
- Simple manifest mechanism.



Example appcache File

appcache File

```
CACHE MANIFEST
# manifest version 1.0.1
# Files to cache
index.html
cache.html
html5.css
image1.jpg
img/foo.gif
http://www.example.com/styles.css

# Use from network if available
NETWORK:
network.html

# Fallback content
FALLBACK:
/ fallback.html
```

Example manifest Attribute

- Reference the manifest file:
  - Use `.appcache` extension (`*.manifest` also allowed)
  - Add as *attribute* to HTML element

HTML

```
<!DOCTYPE html>
<html manifest="offline.appcache">
  <head>
    <title>HTML5 Application Cache Rocks!</title>
```

http://appcachefacts.info/

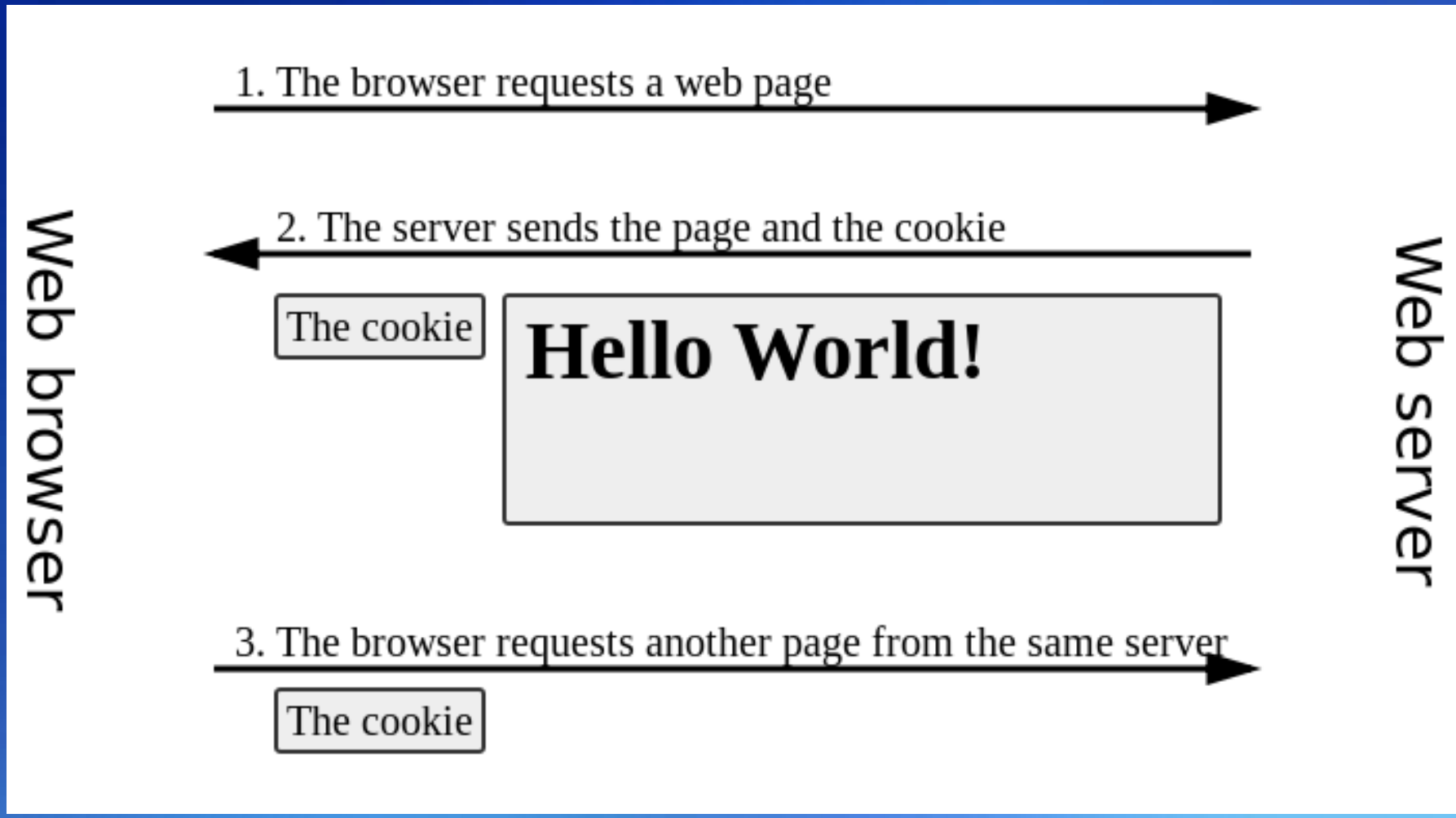
Web and DB Storage

- Many powerful new client-side storage options
- Web and Web Database storage a.k.a. "cookies on steroids”
- Web Storage (Session and Local Storage)
- Web Database Storage

What is a Cookie?

- The original name came from the term ***magic cookie*** - a token passed between two programs.
- Also known as an ***HTTP cookie***, ***web cookie***, or ***browser cookie***.
- Is a small piece of data(a variable) sent from a website and stored in a user's web browser while the user is browsing that website.
- Every time the user loads the website, the browser sends the cookie back to the server to notify the website of the user's previous activity.

What is a Cookie? (Cont.)



Setting a cookie

- Transfer of Web pages follows the HTTP.
- Regardless of cookies, browsers request a page from web servers by sending them a usually short text called ***HTTP request***.
- To access the page http://www.example.org/index.html, browsers connect to the server www.example.org sending it a request that looks like the following one:

```
GET /index.html HTTP/1.1
Host : www.example.org
```

Setting a cookie (Cont.)

- The server replies by sending the requested page preceded by a similar packet of text, called *HTTP response*.
- This packet may contain lines requesting the browser to store cookies:

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

(content of page)
```

Setting a cookie (Cont.)

- The server sends lines of Set-Cookie only if the server wishes the browser to store cookies.
- For example, the browser requests the page http://www.example.org/spec.html by sending the server www.example.org a request like the following:

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
Accept: */*
```

Storing and Reading Cookies

- Cookies are accessible, like most other browser elements, through the `document` object.
- To create a cookie, you'll need to provide a cookie *name*, or *key*, an associated *value*, a date when it *expires*, and a *path* associated with the cookie.
- To access a cookie, you access the `document.cookie` value and then parse the cookie out.

Storing and Reading Cookies (Cont.)

- To create a cookie, just assign the `document.cookie`value a string with the following format:

```
document.cookie="cookieName=cookieValue; expires=date; path=path";
```

Storing and Reading Cookies (Cont.)

- Use cookie names starting with a dollar sign (`$cookieName`), with an underscore (`_cookieName`), and with other characters.
- Use only primitive types (*string*, *boolean*, and *number*) that convert cleanly to strings.
- Also can use JavaScript Object Notation (*JSON*) conversion to set and retrieve complex objects in cookies, refer to

http://www.oreillynet.com/onlamp/blog/2008/05/dojo_goodness_part_8_jsonified.html .

Storing and Reading Cookies (Cont.)

- The expiration date must be in a specific GMT (UTC) format.
- Creating a *date* object and then using the `toGMTString` is sufficient to ensure that the date works.
- If no date is provided, the cookie is considered session-only - is eliminated as soon as the browser session ends.

Storing and Reading Cookies (Cont.)

- The cookie path is especially important.
- The *domain* and *path* are compared with the page request, and if they don't sync up, the cookie can't be accessed or set.
- This prevents other sites from accessing any and all cookies set on your browser, though this security has been circumvented in the past.

Storing and Reading Cookies (Cont.)

- A path setting of `path=/` sets the cookie's allowable path to the top-level directory at your domain.
- If you access the page at *http://somedomain.com*, this means the cookie is accessible by any subdirectory off of *http://somedomain.com*.
- If you specify a subdirectory, such as *path=/images*, the cookie is accessible only from web pages in this subdirectory.

Storing and Reading Cookies (Cont.)

- The following code snippet shows an example of a JavaScript function:

```
function setCookie(key,value) {
    var cookieDate = new Date(2010,11,10,19,30,30);
    document.cookie = key + "=" + encodeURI(value) + "; expires=" +
                      cookieDate.toGMTString() + "; path=/";
}
```

- Getting the cookie is not as easy because all cookies get concatenated into one string, separated by semicolons on the `cookie` object.

```
var1=somevalue; var2=3.55; var3=true
```

Storing and Reading Cookies (Cont.)

```
function readCookie(key) {
    var cookie = document.cookie;

    var first = cookie.indexOf(key+"=");

    // cookie exists
    if (first >= 0) {
        var str = cookie.substring(first,cookie.length);
        var last = str.indexOf(";");

        // if last cookie
        if (last < 0) last = str.length;

        // get cookie value
        str = str.substring(0,last).split("=");
        return decodeURI(str[1]);
    } else {
        return null;
    }
}
```

Storing and Reading Cookies (Cont.)

- To erase the cookie, eliminate its value or set the date to a past date, or do both, as the following JavaScript function demonstrates:

```
function eraseCookie (key) {
   var cookieDate = new Date(2000,11,10,19,30,30);
   document.cookie=key + "=; expires=" + cookieDate.toGMTString() + "; path=/";
}
```

- The next time the document cookie string is accessed, the cookie will no longer exist.

Storing and Reading Cookies (Cont.)

- Before you use any cookie functionality, it's best to first test to make sure cookies are implemented and enabled for the browser.
- To check whether cookies are enabled

```
if (navigator.cookieEnabled) ...
```

- Note that not all browsers return the correct value when testing the `cookieEnabled` property.

Lab1 : Setting, reading, and erasing cookies

- Web Browsers
  - IE10, Firefox, Google Chrome, Opera, Safari
- Text Editors
  - Notepad++ or Editplus
- Files
  - cookie.html

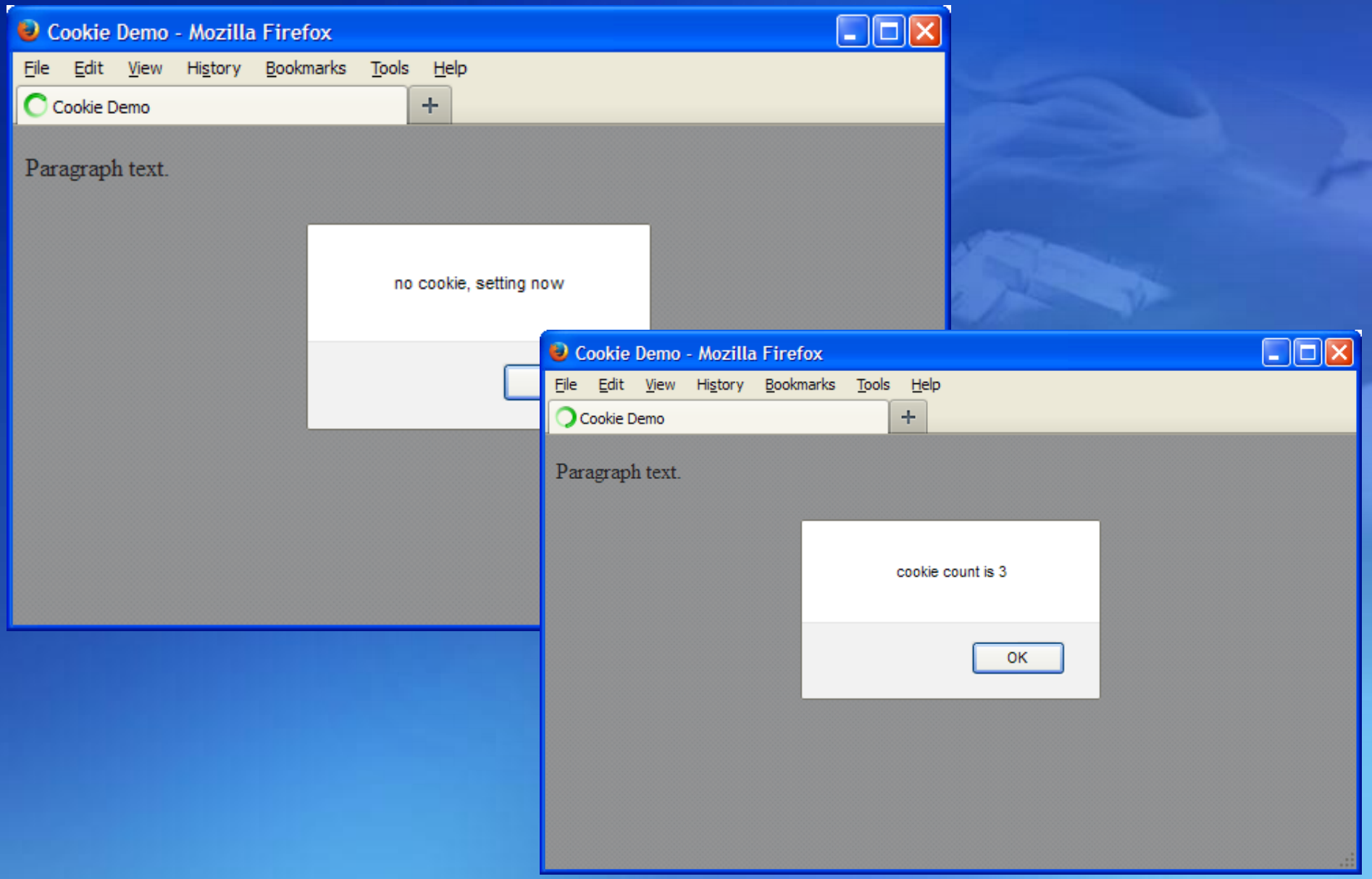
Lab1 : cookie.html (1/2)

```
5       <meta charset="utf-8">
6       <script>
7           window.onload = function(){
8               if (navigator.cookieEnabled) {
9                   var sum = readCookie("sum");
10                  if (sum) {
11                      var iSum = parseInt(sum) + 1;
12                      alert("cookie count is " + iSum);
13                      if (iSum > 5) {
14                          eraseCookie("sum");
15                      } else {
16                          setCookie("sum",iSum);
17                      }
18                  } else {
19                      alert("no cookie, setting now");
20                      setCookie("sum", 0);
21                  }
22              }
23          }
24          // set cookie expiration date in year 2015
25          function setCookie(key,value) {
26              var cookieDate = new Date(2015,11,10,19,30,30);
27              document.cookie=key + "=" + encodeURI(value) + "; expires=" +
28              cookieDate.toGMTString() + "; path=/";
29          }
```

Lab1 : cookie.html)2/2)

```
30          // each cookie separated by semicolon;
31          function readCookie(key) {
32              var cookie = document.cookie;
33              var first = cookie.indexOf(key+"=");
34              // cookie exists
35              if (first >= 0) {
36                  var str = cookie.substring(first,cookie.length);
37                  var last = str.indexOf(";");
38                  // if last cookie
39                  if (last < 0) last = str.length;
40                  // get cookie value
41                  str = str.substring(0,last).split("=");
42                  return decodeURI(str[1]);
43              } else {
44                  return null;
45              }
46          }
47          // set cookie date to the past to erase
48          function eraseCookie (key) {
49              var cookieDate = new Date(2000,11,10,19,30,30);
50              document.cookie=key + "= ; expires="+cookieDate.toGMTString()+"; path=/";
51          }
52      </script>
53  </head>
54  <body>
55      <p>Paragraph text.</p>
```

Lab1 : Result


Cookie Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Cookie Demo
Paragraph text.
no cookie, setting now
Cookie Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Cookie Demo
Paragraph text.
cookie count is 3
OK