Web storage

From Wikipedia, the free encyclopedia
(Redirected from Web Storage)

Web storage and **DOM storage** (document object model) are web application software methods and protocols used for storing data in a web browser. Web storage supports persistent data storage, similar to cookies but with a greatly enhanced capacity[1] and no information stored in the HTTP request header.[2] There are two main web storage types: local storage and session storage, behaving similarly to persistent cookies and session cookies respectively.

Web storage is being standardized by the World Wide Web Consortium (W3C). It was originally part of the HTML 5 specification, but is now in a separate specification.[3] It is supported by Internet Explorer 8, Mozilla-based browsers (e.g., Firefox 2+, officially from 3.5),[4] Safari 4, Google Chrome 4 (sessionStorage is from 5), and Opera 10.50. As of 14 March 2011 Opera and IE9 supports the storage events.[5]

Contents

- 1 Features
  - 1.1 Storage size
  - 1.2 Client-side interface
  - 1.3 Local and session storage
  - 1.4 Interface and data model
- 2 Usage
  - 2.1 sessionStorage
  - 2.2 localStorage
  - 2.3 Accessing data for the currently browsed domain
  - 2.4 Data types
- 3 Nomenclature
- 4 Web Storage Management
- 5 See also
- 6 References
- 7 External links

Features

Web storage can be viewed simplistically as an improvement on cookies. However, it differs from cookies in some key ways.

Storage size

Web storage provides far greater storage capacity (5 MB per origin in Google Chrome,[6] Mozilla Firefox,[7] and Opera; 10 MB per storage area in Internet Explorer;[8] 25MB per origin on BlackBerry 10 devices) compared to 4 kB (around 1000 times less space) available to cookies.

Client-side interface

Unlike cookies, which can be accessed by both the server and client side, web storage falls exclusively under the purview of client-side scripting.

Web storage data is not automatically transmitted to the server in every HTTP request, and a web server can't directly write to Web storage. However, either of these effects can be achieved with explicit client-side scripts, allowing for fine-grained tuning of the desired interaction with the server.

Local and session storage

Web storage offers two different storage areas—local storage and session storage—which differ in scope and lifetime. Data placed in local storage is per origin (the combination of protocol, hostname, and port number as defined in the same origin policy) (the data is available to all scripts loaded from pages from the same origin that previously stored the data) and persists after the browser is closed. Session storage is per-page-per-window and is limited to the lifetime of the window. Session storage is intended to allow separate instances of the same web application to run in different windows without interfering with each other, a use case that's not well supported by cookies.[9]

Interface and data model

Web storage currently provides a better programmatic interface than cookies because it exposes an associative array data model where the keys and values are both strings. An additional API for accessing structured data is being considered by the W3C Web Applications Working Group. [10]

Usage

Browsers that support web storage have the global variables 'sessionStorage' and 'localStorage' declared at the window level. The following JavaScript code can be used on these browsers to trigger web storage behaviour:

sessionStorage

```
// Store value on browser for duration of the session
sessionStorage.setItem('key', 'value');

// Retrieve value (gets deleted when browser is closed and re-opened)
alert(sessionStorage.getItem('key'));
```

localStorage

```
// Store value on the browser beyond the duration of the session
localStorage.setItem('key', 'value');

// Retrieve value (persists even after closing and re-opening the browser)
alert(localStorage.getItem('key'));
```

Accessing data for the currently browsed domain

The following code can be used to retrieve all values stored in local storage for the currently browsed domain (the domain for the web page that is being browsed).

This JavaScript code can be executed using development tools available in most modern browsers such as the IE Developer Toolbar, Chrome Developer Tools, the Firebug extension in Firefox, or Opera Dragonfly:

```
var output = "LOCALSTORAGE DATA:\n-----------------------------------\n";
if (window.localStorage) {
    if (localStorage.length) {
        for (var i = 0; i < localStorage.length; i++) {
            output += localStorage.key(i) + ': ' + localStorage.getItem(localStorage.key(i)) + '\n';
        }
    } else {
        output += 'There is no data stored for this domain.';
    }
} else {
    output += 'Your browser does not support local storage.'
}
console.log(output);
```

Data types

Only strings can be stored via the Storage API.[11] Attempting to store a different data type will result in an automatic conversion into a string in most browsers. Conversion into JSON (JavaScript Object Notation), however, allows for effective storage of JavaScript objects.

```
// Store an object instead of a string
localStorage.setItem('key', {name: 'value'});
alert(typeof localStorage.getItem('key')); // string

// Store an integer instead of a string
localStorage.setItem('key', 1);
alert(typeof localStorage.getItem('key')); // string

// Store an object using JSON
localStorage.setItem('key', JSON.stringify({name: 'value'}));
alert(JSON.parse(localStorage.getItem('key')).name); // value
```

Nomenclature

The W3C draft is titled "Web Storage", but "DOM storage" is also a commonly used name.[12][13]

The "DOM" in DOM storage doesn't literally refer to the Document Object Model. According to the W3C, "The term DOM is used to refer to the API set made available to scripts in Web applications, and does not necessarily imply the existence of an actual Document object..."[14]

Web Storage Management

Storage of web storage objects is enabled by default in Mozilla Firefox and SeaMonkey, but can be disabled by setting the "about:config" parameter "dom.storage.enabled" to false.[15]

Mozilla Firefox stores all web storage objects in a single file named webappsstore.sqlite. The sqlite3 command can be used to show the elements stored therein.[16]

There are browser extensions/add-ons for Google Chrome and Mozilla Firefox available that let the user deal with web storage, such as "Click&Clean"[17][18] and "BetterPrivacy" which can be configured to remove the whole web storage automatically on a regular basis.[19][20][21]

See also

- HTTP cookies
- Indexed Database API[22] (formerly WebSimpleDB)
- Web SQL Database
- Local Shared Objects in Adobe Flash

- userData Behavior in Internet Explorer
- Google Gears for IE, Firefox, Safari and Windows Mobile

References

1. ^ Opera Web Storage, 2011 http://dev.opera.com/articles/view/web-storage/
2. ^ AndyHume.net, 2011 http://blog.andyhume.net/localstorage-is-not-cookies
3. ^ Web Storage (http://www.w3.org/TR/webstorage/). W3.org. Retrieved on 2011-06-12.
4. ^ Mozilla Developer Center: DOM Storage (https://developer.mozilla.org/En/DOM:Storage#Description). Developer.mozilla.org. Retrieved on 2011-06-12.
5. ^ [1] (http://www.codeproject.com/Articles/162783/HTML5-Web-Storage-in-Essence). HTML5 Web Storage in Essence (2011-02-28). Retrieved on 2012-03-30.
6. ^ chrome.storage.local.QUOTA_BYTES (http://developer.chrome.com/extensions/storage.html#property-local-QUOTA_BYTES) Chrome extension developer documentation.
7. ^ John Resig: DOM Storage (http://ejohn.org/blog/dom-storage/). John Resig, *ejohn.org*. Retrieved on 2011-06-12.
8. ^ MSDN: Introduction to DOM Storage (http://msdn.microsoft.com/en-us/library/cc197062(VS.85).aspx#_dom). *Microsoft Developer Network*, *msdn.microsoft.com*. Retrieved on 2011-06-12.
9. ^ W3C: Web Storage draft standard (http://dev.w3.org/html5/webstorage/#introduction). Dev.w3.org (2004-02-05). Retrieved on 2011-06-12.
10. ^ W3C: Indexed Database API (http://www.w3.org/TR/IndexedDB/). W3C. Retrieved on 2012-02-12.
11. ^ W3C, 2011 http://dev.w3.org/html5/webstorage/
12. ^ Mozilla Developer Center: DOM Storage (https://developer.mozilla.org/En/DOM:Storage). Developer.mozilla.org. Retrieved on 2011-06-12.
13. ^ MSDN: Introduction to DOM Storage (http://msdn.microsoft.com/en-us/library/cc197062(VS.85).aspx). Msdn.microsoft.com. Retrieved on 2011-06-12.
14. ^ W3C: Web Storage draft standard (http://dev.w3.org/html5/webstorage/#terminology). Dev.w3.org (2004-02-05). Retrieved on 2011-06-12.
15. ^ Mozillazine article on disabling Web Storage Objects in about:config. Kb.mozillazine.org. Retrieved on 2011-06-12.
16. ^ Firefox’s Super Cookies (http://www.cerias.purdue.edu/site/blog/post/firefoxs-super-cookies), Cerias, January 16, 2008
17. ^ "Click&Clean" extension for Google Chrome (http://www.hotcleaner.com/clickclean_chrome.html). Hotcleaner.com (2011-06-01). Retrieved on 2011-06-12.
18. ^ "Click&Clean add-on for Mozilla Firefox (https://addons.mozilla.org/en-US/firefox/addon/clickclean/). Addons.mozilla.org. Retrieved on 2011-06-12.
19. ^ Mozilla add-ons page for "Better Privacy" (https://addons.mozilla.org/en-US/firefox/addon/betterprivacy/). Addons.mozilla.org. Retrieved on 2011-06-12.
20. ^ Homepage of "Better Privacy", with some further references to blogs and articles (http://netticat.ath.cx/extensions.html). Netticat.ath.cx. Retrieved on 2011-06-12.
21. ^ Google Chrome Browser Client-Side Storage (http://www.hotcleaner.com/web_storage.html). Hotcleaner.com. Retrieved on 2011-06-12.
22. ^ Indexed Database API (http://www.w3.org/TR/IndexedDB/). W3.org. Retrieved on 2011-06-12.

External links

- W3C: Web Storage (http://www.w3.org/TR/webstorage/)
- MSDN: Introduction to DOM Storage (http://msdn.microsoft.com/en-us/library/cc197062(VS.85).aspx)
- Mozilla Developer Center: DOM Storage (https://developer.mozilla.org/En/DOM:Storage)
- Opera: Web Storage: easier, more powerful client-side data storage (http://dev.opera.com/articles/view/web-storage/)
- BB10: HTML5 WebWorks Api Reference (https://developer.blackberry.com/html5/api/localStorage.html)