Document Object Model Web Storage

- Introduction
- What is DOM Storage?
  - window.sessionStorage
  - window.localStorage
- The Storage Object
- Web Storage Events
  - onstorage
  - onstoragecommit
  - Security and Privacy
  - Top-Level Browsing Context and Hostname
  - Origin Determines Storage Limits
  - Clearing the Storage Areas
- References
  - Further Reading

Introduction

The Web Storage API includes two related mechanisms for persisting client-side data in a secure manner using the Document Object Model (DOM), sessionStorage and localStorage. These objects were introduced in Windows Internet Explorer 8.

⚠️ Note: For versions of Windows Internet Explorer prior to version 8, persistent data storage is implemented by the userData behavior.

Type		Last Visited [UTC]	Last Visited [Local]	Hits	URL	Host
domstore	☑	2011-11-21 13:11:09	2011-11-21 13:11:09	1	http://kb.digital-detective.co.uk/	kb.digital-detective.co.uk
domstore	☑	2011-11-07 10:41:24	2011-11-07 10:41:24	1	http://blog.digital-detective.co.uk/	blog.digital-detective.co.uk
domstore	☑	2011-10-12 09:07:13	2011-10-12 10:07:13	1	http://maps.google.co.uk/	maps.google.co.uk
domstore	☑	2011-10-07 10:50:32	2011-10-07 11:50:32	1	http://support.apple.com/	support.apple.com
domstore	☑	2011-10-06 13:42:12	2011-10-06 14:42:12	1	https://developer.mozilla.org/	developer.mozilla.org
domstore	☑	2011-09-15 10:43:29	2011-09-15 11:43:29	1	http://www.google.co.uk/	www.google.co.uk

DOMstore entries in NetAnalysis 1.5x

What is DOM Storage?

Web Storage is often compared to HTTP cookies. Like cookies, Web developers can store per-session or domain-specific data as name/value pairs on the client using Web Storage. However, unlike cookies, Web Storage makes it easier to control how information stored by one window is visible to another.

For example, a user might open two browser windows to buy airline tickets for two different flights. However, if the airline's Web application uses cookies to store its session state, information could "leak" from one transaction into the other, potentially causing the user to buy two tickets for the same flight without noticing. As applications become more capable of offline behaviours, such as storing values locally for later return to the server, the potential for this sort of information "leak" becomes more prevalent.

Web Storage also offers significantly more disk space than cookies. In Internet Explorer, cookies can store only 4 kilobytes (KB) of data. This byte total can be one name/value pair of 4 KB, or it can be up to 20 name/value pairs that have a total size of 4 KB. By comparison, Web Storage provides roughly 10 megabytes (MB) for each storage area.

Functionally, client storage areas are quite different from cookies. Web Storage doesn't transmit values to the server with every request as cookies do, nor does the data in a local storage area ever expire. And unlike cookies, it is easy to access individual pieces of data using a standard interface that has growing support among browser vendors.

window.sessionStorage

Session storage is designed for scenarios where the user is carrying out a single transaction. The sessionStorage attribute of the window object maintains key/value pairs for all pages loaded during the lifetime of a single tab (for the duration of the top-level browsing context). For example, a page might have a check box that the user selects to indicate that he wants insurance.

```
<label>
        <input type="checkbox" onchange="sessionStorage.insurance = checked">
        I want insurance on this trip.
</label>
```

A later page could then check, from script, whether the user had selected the check box.

```
if (sessionStorage.insurance) { ... }
```

The Storage object supports expando properties ('insurance' in the preceding example). If the property name does not exist, a key/value pair is automatically created to hold it. Note that key/value pairs are always stored as strings. Different data types such as numbers, Boolean values, and structured data must be converted to strings before persisting to a storage area.

After a value has been saved to sessionStorage, it can be retrieved by script running in another page in the same context. When another document is loaded, sessionStorage is initialised from memory for same-origin URLs. (See Security and Privacy section for more information.)

⚠ Note: Although it is allowed by the HTML5 (Working Draft), Internet Explorer 8 does not resume sessionStorage after browser crash recovery.

window.localStorage

The local storage mechanism spans multiple windows and persists beyond the current session. The **localStorage** attribute provides persistent storage areas for domains. It allows Web applications to store nearly 10 MB of user data, such as entire documents or a user's mailbox, on the client for performance reasons.

For example, a Web site can display a count of how many times the user has visited a page with the following script.

```
<p>
  You have viewed this page
  <span id="count">an untold number of</span>
  time(s).
</p>
<script>
  var storage = window.localStorage;
  if (!storage.pageLoadCount) storage.pageLoadCount = 0;
  storage.pageLoadCount = parseInt(storage.pageLoadCount, 10) + 1;
  document.getElementById('count').innerHTML = storage.pageLoadCount;
</script>
```

⚠ Note: Before incrementing pageLoadCount it must first be converted to a number with the parseInt Method (JScript 5.6).

Each domain and subdomain has its own separate local storage area. Domains can access the storage areas of subdomains, and subdomains can access the storage areas of parent domains. For example, `localStorage['example.com']` is accessible to `example.com` and any of its subdomains. The subdomain `localStorage['www.example.com']` is accessible to `example.com`, but not to other subdomains, such as `mail.example.com`.

The Storage Object

The following properties and methods are supported by both session and local storage objects.

Topic	Contents
clear	Removes all key/value pairs from the Web Storage area.
constructor	Returns a reference to the constructor of an object.
getItem	Retrieves the current value associated with the Web Storage key.
key	Retrieves the key at the specified index in the collection.
length	Retrieves the length of the key/value list.
remainingSpace	Retrieves the remaining memory space, in bytes, for the storage object.
removeItem	Deletes a key/value pair from the Web Storage collection.
setItem	Sets a key/value pair.

Web Storage Events

Internet Explorer fires events when data in a storage area is updated, so that information can synchronized between multiple instances of the browser or tabs.

The following events are supported:

- onstorage
- onstoragecommit

onstorage

The **onstorage** event is fired in a **document** when a storage area changes. All documents sharing the same session context, and those that are currently displaying a page from the same domain or subdomain where local storage is being committed, receive the event. If the target **document** object is not currently active, Internet Explorer does not fire any events.

onstoragecommit

Internet Explorer uses XML files to store local storage. The **onstoragecommit** event fires when a local storage is written to disk.

Security and Privacy

The data stored in local storage is much more public than that stored in cookies, which can be limited to a certain path within a domain. Even picking a hard-to-guess key won't provide any privacy because the **Storage** object provides a way to enumerate them.

Here are some things to consider:

- Top-Level Browsing Context and Hostname
- Origin Determines Storage Limits
- Clearing the Storage Areas

Top-Level Browsing Context and Hostname

Access to the session storage area is restricted by the top-level browsing context. In Internet Explorer, a new browsing context is created for every tab. Script running in one top-level browsing context has no access to storage created in another. Sites can add data to the session storage, and it will be accessible to *any page from that hostname* opened in the same window.

⚠️ Note**:** The port and protocol/scheme are not evaluated as a part of this check.

Origin Determines Storage Limits

Disk quota limits are imposed against the domain of the page that sets the value, rather than the domain where the value is being set. This prevents malicious scripts from using up the storage quota of a related domain. It also prevents such scripts from using random subdomains to store unrestricted amounts of data.

Storage size is calculated as the total length of all key names and values, and a single storage area can contain up to 10 million bytes. The **remainingSpac** property is used to determine the available storage space.

Clearing the Storage Areas

Session state is released as soon as the last window to reference that data is closed. However, users can clear storage areas at any time by selecting **Dele te Browsing History** from the **Tools** menu in Internet Explorer, selecting the **Cookies** check box, and clicking **OK**. This clears session and local storage areas for all domains that are not in the Favorites folder and resets the storage quotas in the registry. Clear the **Preserve Favorite Site Data** check box to delete all storage areas, regardless of source.

To delete key/value pairs from a storage list, iterate over the collection with **removeItem** or use **clear** to remove all items at once. Keep in mind that changes to a local storage area are saved to disk asynchronously.

References

- Introduction to Web Storage
- HTML5 (Working Draft)
- State and Storage

Further Reading

- Document Object Model (DOM)
- Wikipedia - Document Object Model
- Wikipedia - Web storage

- localStorage is not cookies
- Mozilla Developer Network - DOM Storage
- Web Storage: easier, more powerful client-side data storage
- The DOM and 3rd Party Javascript Vulnerabilities
- Running your web applications offline with HTML5 AppCache
- Taking your web apps offline: A tale of Web Storage, Application Cache and WebSQL