W3C Web Storage Standard

Web Presentation

by Perry Abramowitz
INFSCI 2560, Fall 2013

Local Storage

One thing that native applications have historically had over web applications is the ability to use the client's machine for storing and retrieving data for use in the application. Native applications can store data in a computer registry, INI files or even imbed a database for use in the application. Web apps have clumsily made use of cookies as substitutes for local storage, but cookies have a number of limitations: they are burdensomely sent with every http request, they send the data unencrypted, and they are limited to about 4 KB of data. Come HTML5 to natively implement local storage in web browsers so it is available even when cookies or third-party browser plugins are not.

The Web storage API stores information locally for later use in the web application. (Note: in Firefox, if cookies disabled, so is Web storage.) Cookies are better for sharing client side data with the server side because cookies append themselves to every request automatically, but when communication with the server is unnecessary for your application, use of local data storage will optimize performance. Again, if you don't need to transmit the local data to the server, then avoid the performance lag of transmitting the local data back and forth from client to server by using Web storage API.

The Web Storage Recommendation has been standardized by W3C as of 30 July 2013. The Web storage specification defines an API for persistent data storage of key-value pair data in Web clients. There are two storage mechanisms in the API. The first is called sessionStorage and the second, localStorage. With sessionStorage, data added will be accessible to any page from the same site opened in that window. When the browser is closed, the data stored is lost. On the other hand, localStorage is designed for storage that spans multiple windows, and lasts beyond the current session.

How it Works

Web storage simply associates a key with a value. An example from "Introducing HTML5" by Bruce Lawson and Remy Sharp (names have been changed to protect the innocent):

```
localStorage.superHero = "Yang Zhang Wham Bang";

localStorage.superVillain = "Perry the Pirate";


//some super hero fight occurs delete localStorage.superVillain;

//the page is reload, brower restarted—we don't care—we're
superheroes!

alert("The world's baddest badass is: " +
localStorage.superHero);
```

In the code above we have two name-value pairs. For the first, "superHero" is the name and "Yang Zhang Wham Bang" is the value. The second pair is "superVillain" and "Perry the Pirate".

Limitations

- No direct access to data for the db administrator. The information is stored locally on client side. Web storage is a client side storage engine. Web storage is divorced from the server.
- The fanciest thing about PHP, Apache, SQL on the server is sharing information. Web storage is not your one stop shop for social media applications.
- Storage limited to 5 MB (megabytes)
- The Web storage getItem method only supports strings. In other words, all data collected is changed to a string for storage and retrieval. If you are storing something other than a string, you'll need to coerce it yourself when you retrieve it. The good news is that most browsers have JSON (JavaScript Object Notation) which includes a method to store data as an object able to preserve the data type.
- Data is more vulnerable that server-side storage.
- The user can't access the data from multiple clients.

One of the more serious limitations that I discovered—and have not found a workaround for—can be intimated from a statement on w3cschools.com: "With HTML5, web pages can store data locally within the user's browser." This sounds all positive and brings a smile to your face, but let me tell you the agony in the words "within…browser." While trying to develop a demo to-do list using web storage, I noticed my to-do list would render just fine…until I switched browsers. The fabulous persistent local storage data would be gone. Apparently, according to one unverifiable source, storage is stored as per the browser specific paths. So your to-do list you created in Firefox cannot be retrieved in Chrome, etc. What isn't the place where the data is stored standardized?

Benefits

- Storing a lot more data.
- Simplicity.

- Wide browser support.
- Cookies expire, localStorage does not.
- Enables to data to application available when offline.
- Web storage is compatible with mobile OS as well: http://mobilehtml5.org/.

How to Use HTML5's Data Storage API

Introducing HTML's Data Storage API into your website can add fun and functionality. For instance, you can keep a count of how many times a visitor to your website has clicked a link (or anything for that matter) by writing a simple function and adding `onclick="clickCounter()"` to your HTML5 markup. Whatever entity you add the onclick to will fire the function and keep a count in local storage.

Think of what fun you could have with your CSS when certain counts are reached. A simple example would be to reorder lists based on clicks, or change the color of some text. The possibilities are endless, and enticing.

Put this code in the <script> section:

```
function clickCounter(){
localStorage.clickcount=Number(localStorage.clickcount)+1;
}
```

In the example above "clickcount" is the name (or key) and the value is the count.

Technical Matters

A listing of the possibilities available to us as web developers or coders probably can't hurt at this point. The Object is known as Storage (with its two object attributes: sessionStorage and localStorage) and its methods can be called like so: `objectName.methodName()`. Here is a list of the available methods: length, key, getItem, setItem, removeItem, and clear.

Properties are the values associated with an object. The syntax for accessing the property of an object is: `objectName.propertyName`. Here is a list of the available properties: key, oldValue, newValue, url, and storageArea.

In order to disambiguate, there is both a key method and a key property. The key property is the value associated with the value in storage, as explained above. The key(n) method returns the name of the nth key in the list.

The methods are all self-explanatory. No funny business here. The properties as well are pretty clear. Two possible exceptions are the url property which represents the address of the document whose key changed; and the storageArea property which represents the Storage object that was affected. Indeed, there is a reason why these two are obscure. Perhaps they are useful in recovery of the object, but I am unsure.

Practical Matters and Possibilities

It isn't hard to find uses for Web Storage in computer applications. The API could be used for storage of user preferences, or a place to save usernames, or even as a way to maintain progress after a browser is closed, such as a resuming a game in another browser by recalling the status of a player before the browser was closed. It could also be useful in modifying the layout style of a page based on user behavior. For instance, if a user is consistently navigating to a portion of the website, why not use web storage to place a direct link to the user's preferred portal on the home page? Some developers use web storage during the development phase when they are prototyping larger applications because of its simplicity. The possibilities are intriguing.

Bibliography

Lawson, Bruce, and Remy Sharp. 2012. *Introducing HTML5*. Berkeley: New Riders.

Hickson, Ian, ed. "W3C Web Storage Recommendation," last modified July 30, 2013,

http://www.w3.org/TR/webstorage/.

Pilgrim, Mark. 2011. *Dive Into HTML5*. http://diveintohtml5.info/.