Client Side Data Persistence

Before HTML5, if your web page wanted to store any data, it could do so only by using JavaScript code to write data to a "cookie" (discussed below). The cookie data would be stored on the user's PC/MAC AND it would also be sent along with every server request. HTML5 introduced "Web storage" (also called "HTML5 Storage" and "DOM Storage"), allowing for larger amounts of data to be stored more securely and without affecting website performance.

Web storage provides two objects for storing data on the client:

- window.localStorage - stores data with no expiration date
- window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

LocalStorage:

1. Web storage can be viewed simplistically as an improvement on cookies, providing much greater storage capacity. Available size is 5MB which considerably more space to work with than a typical 4KB cookie.
2. The data is not sent back to the server for every HTTP request (HTML, images, JavaScript, CSS, etc) - reducing the amount of traffic between client and server.
3. The data stored in localStorage persists until explicitly deleted. Changes made are saved and available for all current and future visits to the site.
4. It works on same-origin policy, meaning that data stored by one domain name will only be available to pages from that same domain. More specifically, the requests must have the same URI scheme, hostname, and port number. For example, if you saved data from http://www.mysite.com/, requesting that data from the following URLs would result in failure.
    - https://www.mysite.com/ – Different protocol (or URI scheme).
    - http://www.mysite.com:8080/myUrl – Different port (since HTTP requests run on port 80 by default).
    - http://www.myothersite.com/ – Different domain.
    - http://mysite.com/ – Treated as a different domain as it requires the exact match (missing "www.").

sessionStorage:

1. sessionStorage is similar to localStorage (data saved is only available to "same-origin" pages) but additionally, it is only available to pages within the same window (or tab in browsers like Chrome and Firefox).
2. Once the window (or tab) is closed, the storage is deleted

Cookies:

1. We can set the expiration time for each cookie
2. The 4K limit is for the entire cookie, including name, value, expiry date etc. To support most browsers, keep the name under 4000 bytes, and the overall cookie size under 4093 bytes.
3. The data is sent back to the server for every HTTP request (HTML, images, JavaScript, CSS, etc) - increasing the amount of traffic between client and server.

Code Examples - localStorage Object:

The localStorage object stores the data with **no expiration date**. The data will not be deleted when the browser is closed, and should persist until the user explicitly deletes the storage (e.g., by clearing browser data).

```
// Store
localStorage.setItem("lastname", "Smith");

// Retrieve
var theLastName = localStorage.getItem("lastname");

// Remove
localStorage.removeItem("lastname");

// Show all data in localStorage
console.log("local storage");
for (var i = 0; i < localStorage.length; i++)   {
   console.log(localStorage.key(i) + ":" + localStorage.getItem(localStorage.key(i)) + ", ");
}
```

Code for the sessionStorage Object is the same as the code for the localStorage object, **except** that the data stored is only available to the same browser window (or tab) and the data is deleted when the user closes the window (or tab).