

Advanced Javascript

Storage, Cache, Cookie & Objects

Yash Lathiya

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No. | Topic | Date | Page No. |
|  |  |  |  |
| 1 | Session & Local Storage | 01/08/2023 | 2 |
| 2 | Cookies | 02/08/2023 | 3 |
| 3 | Browser Debugging | 02/08/2023 | 4 |
| 4 | Caching | 03/08/2023 | 5 |
| 5 | OOJS | 04/08/2023 | 7 |
| 6 | ECMAScript6 | 04/08/2023 | 7 |

Table of Contents

HTMl Web Storage

Local Storage & Session Storage

* With web storage, web applications can store data locally within the user's browser.
* Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
* Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
* HTML 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)
* Opening a page in a new tab or window creates a new session with the value of the top-level browsing context, which differs from how session cookies work.
* Opening multiple tabs/windows with the same URL creates sessionStorage for each tab/window.
* Duplicating a tab copies the tab's sessionStorage into the new tab.
* Closing a tab/window ends the session and clears objects in sessionStorage.

Cookies

* Cookies are data, stored in small text files, on your computer.
* When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.
* Cookies were invented to solve the problem "how to remember information about the user":
* When a user visits a web page, his/her name can be stored in a cookie.
* Next time the user visits the page, the cookie "remembers" his/her name.
* Cookies are saved in name-value pairs like:

username = yashLathiya

* How to access & extract information from cookie, that implementation is showed in demo.

Browser Debugging

Inspect Element Window

* Inspect element window is a window which is used by developers to understand effect of code, and actually which type of properties contains component of webpage.
* It contains style, code, console, application, etc. sections.
* Through which we can understand web page creation & working properly.

Breakpoint

* A breakpoint is a point of code where the debugger will automatically pause the JavaScript execution.
* When we add breakpoints & rerun the webpage, the script pauses on our breakpoints and we can debug any function or execution stack.
* Style & console component can be also used for CSS & javascript respectively.

Caching

* Cache provides persistent storage mechanism for request/responce which is browser depended.
* Cache items do not expire until they're deleted.
* It reduces time of opening webpage.
* [Memorization](https://en.wikipedia.org/wiki/Memoization) is an optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again.

//More about programming (similar to dynamic programming)

* How to check cache is supported or not ?

let isCacheSupported = 'caches' in window;

Caches is instance of cacheStorage.

Creating a cache

* Cache can be initialized by open method by giving cacheName, which will returns a promise.
* If cache is already present, it will not create it.
* We can’t create cache for other domains, cache can be created for origin only.

Adding

* Adding of cache can be done by 3 methods : add, addAll & put.
* After providing url, response is cached.
* Put is used for key-value pair (url, responce).

Retrieving from cache

* Using cache.match() method, we can get response e=which was stored in url.
* We can retrieve all items by using caches.keys() method which give all key pairs , & we can make array of response.

Remove from cache

* .delete(“url”) method is used for removing cache. By iterate cache in caches..
* In order to delete cache completely caches.delete(“cacheName”)

CacheStorage

* If we want to use multiple cache instances, then it is handled by cacheStorage - it will contain all cache instances.
* It contains following methods..
* CacheStorage.has()
* CacheStorage.match()
* CacheStorage.open()
* CacheStorage.delete()
* CacheStorage.keys()
* Works of method is similar to Caches

OOJS & ECMAScript6

* OOJS is object oriented javascript and ECMAScript6 is a version of javascript with standardization.
* Implementation of both topics is demonstrated in demo with proper comments.

<https://github.com/RKITSoftware/Yash-Lathiya/tree/main/Demo/GUI%20Basic%20Demo/5_Advance_Javascript>