LocalStorage

Client-Side Browser Storage

Cookies are shared by clients and servers, and are fairly small (4kb)
There are Storage technologies built-in to the browser to store data that doesn't need to be stored on the server
What information can you think of that is useful to store only on the client?

- Color theme preferences
- Saved shopping cart (until an "order" button is clicked to submit a request
- Previous searches
- ...

Browser storage technologies

`localStorage` and `sessionStorage` can be used to store data on the browser.
`localStorage` is a `window` property that allows you to save information across browser sessions (i.e after you close the browser)
`sessionStorage` is a `window` property that allows you to save information for this session only, and will be cleared when the page is closed.
Name/value pairs (seen in cookies and Storage) are supported by most every browser

Storage

There are four useful methods for `Storage`.

method	description
`setItem(keyName, keyValue)`	Sets the keyName location in storage to be keyValue
`getItem(keyName)`	Retrieves the keyValue in storage associated with keyName
`removeItem(keyName)`	Removes the keyName location in storage
`clear()`	Removes all key/values in the storage

storage-demo.html compares the two storage technologies using the code on the following slides. Inspect the Chrome Application Tab to see the difference between different browser windows.

localStorage example

```
window.localStorage.setItem("color-mode", "dark");
window.localStorage.setItem("language", "en");
window.localStorage.setItem("favorite-drink", "coffee");
let colorMode = window.localStorage.getItem("color-mode");
window.localStorage.removeItem("color-mode");
```

sessionStorage example

```
window.sessionStorage.setItem("color-mode", "dark");
window.sessionStorage.setItem("language", "en");
window.sessionStorage.setItem("favorite-drink", "coffee");
let colorMode = window.sessionStorage.getItem("color-mode");
window.sessionStorage.removeItem("color-mode");
```

Viewing Storage in the Applications Tab

localStorage

Set localStorage | Clear localStorage

sessionStorage

Set sessionStorage | Clear sessionStorage

Open the Applications tab!

localStorage populated!

Elements | Application | Console | Network | Sources | Performanc

Application
- Manifest
- Service Workers
- Clear storage

Storage
- Local Storage
  - https://courses.cs.washington.edu
- Session Storage
  - https://courses.cs.washington.edu
- IndexedDB
- Web SQL
- Cookies

Filter

Key	Value
color-mode	dark
language	en
vegetarian	true
favorite-drink	coffee

1 dark

Requirement: Values are Strings

But what if you want a key to hold a collection of items (e.g. a dictionary of preferences, or a current cart)?

```
let notes = [{ tag : "Personal", note : "Feed Orange." },
             { tag : "Personal", note : "Exercise Juice."}];
window.localStorage.setItem("notes", notes);
let data = window.localStorage.getItem("notes");
// { notes : [Object object] }
```

Solution: Save stringified object (possibly an array) and access as parsed JSON

```
window.localStorage.setItem("notes", JSON.stringify(notes));
let data = JSON.parse(window.localStorage.getItem("notes"));
// { notes : [{ tag : "Personal", note : "Feed Orange." },
//            { tag : "Personal", note : "Exercise Juice." }]}
```

HTTP and State

HTTP is a stateless protocol; it simply allows a browser to request a single resource from a web server.
Once the resource has been sent to the client, the server does not keep track of any information about what was sent (other than maybe in a log file of the transaction).
But then how can websites like Amazon.com, Google, etc. remember whether you're logged in and your shopping preferences?
How does a client uniquely identify itself to a server, and how does the server provide specific content to each client?

Cookies

A small (max 4kb) amount of information stored within the computer browser
Introduced in 1994 for the Netscape browser to improve shopping experience

Have many uses:

- Authentication, remembering login information
- User tracking
- Maintaining user preferences, shopping carts, etc.

Demo example of language preference cookie (try changing the language and finding the google translate cookie in your Chrome Applications tab - what happens when you refresh the page?).

Cookies: Shared Between Client and Server

Cookies are associated with certain websites. They consist of a key and a value. They also have an expiration date, and will go away when it is reached. If an cookie does not have an expiration date, it is usually a "session cookie", cleared when the browser is closed. (the King County's language cookie is a session cookie).

Cookies let us store information on a user's computer, for things like keeping them logged in even if they close the page. These are sent by the same client to the server in future requests (until they expire).

As a web developer, this is a great feature to have to improve user experience (but you should be clear about your Privacy Policy).

More Cookie Details

- Cookies are only data, not program code.
- Cookies can be used to track your viewing habits on a particular site.
  - Do you think this is a good thing? Or a bad thing?
- New privacy laws in Europe (GDPR) and California (CCPA) are making website owners rethink using cookies.

A great 10-minute video here with a simple overview of how different types of cookies work

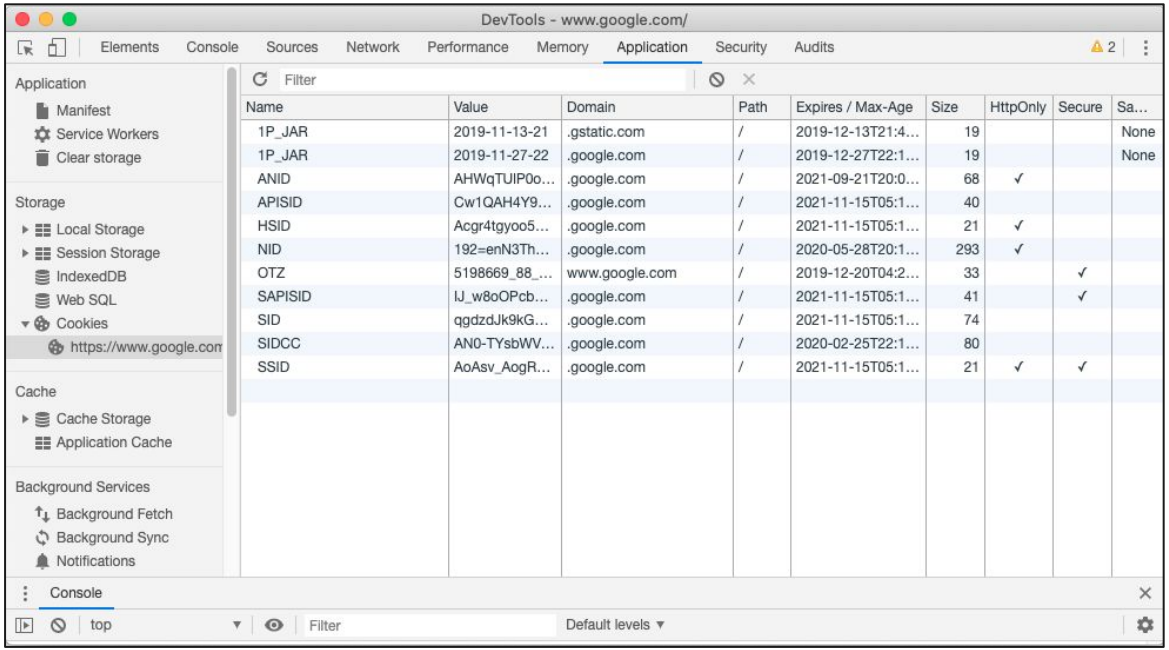
Format of Cookies

Cookies are essentially just key -> value pairs with some extra information.

- path: which page does this apply to?
- domain: the domain (usually the current one you're on)
- max-age: how long (in seconds) is this cookie good for
- expires: what's the expiration date for this cookie?
- secure: HTTPS-only?
- samesite: other domains are not allowed to eat this cookie.

Viewing Cookies

You can view your cookies for different sites on the Chrome Tools Application Tab. Try it out on different websites!



How cookies can be set and retrieved

Client-Side

- JavaScript can set and retrieved using document.cookie
- More details here

Server-Side (Node/Express, but also other server-side languages):

- When the browser requests a page, the server may send back a cookie(s) with it to store on the client - this can be done in different server-side languages.
- Future requests to the server will bring back the cookie until it expires.
- If your server has previously sent any cookies to the browser, the browser will send them back on subsequent requests to remind the server who is connecting to it.

How to read and use a cookie?

```
document.cookie
```

Returns a long string, looking something like:

```
"_ga=GA1.2.1860036673.1569205383; dwf_sg_task_completion=False;
_gid=GA1.2.1782728754.1588304902; lux_uid=158861962030447448"
```

This string follows a pattern (RegEx, anyone?), like `key=value; key2=value2`
Can be tricky to parse correctly. So... there's another way.

Cookie Footnote: Cookies and Expiration Time

Note: For *never-expire-cookies* we used the arbitrarily distant date `Fri, 31 Dec 9999 23:59:59 GMT`. If, for any reason, you are afraid of such a date, use the *conventional date of the end of the world* `Tue, 19 Jan 2038 03:14:07 GMT` – which is the maximum number of *seconds* elapsed since 1 January 1970 00:00:00 UTC expressible by a signed 32-bit integer (i.e., `01111111111111111111111111111111` which is `new Date(0x7fffffff * 1e3)`).

Summary: Cookie vs. Browser Storage

Note that storage limits for local/session storage depend on browser and device (Desktop vs. Mobile).

	Cookies	Local Storage	Session Storage
Size	4kb	~10mb	~5-10m
Expires	Manually Set	Never	When browser is closed
Storage Location	Browser & Server	Browser Only	Browser only
Sent w/ HTTP Requests	Yes	No	No