**Cookies Vs Local Storage vs Session Storage.**

* All three of them will be saved user browser means chrome to firefox will not be same or shared
* Cookies are diff then other 2. And smaller info than local and session.
* Cookies and Local are available on any tab or window. But session stograge is only available in single tab
* So session storage is for 1 browser and as soon as user close the tab that session is set.
* But local storage is there forever unless user delete it.
* Cookies have to set when they are expired.
* Cookies are sent to the server everytime client request something but other 2 are in browser
* That is why cookies are smaller in term of size.

localStorage has few methods = clear, getItem, key, length, removeItem, setItem

# localStorage.setItem(‘name’,’jansir’) [so this takes 2 value as a parameter which are key and values and everyting has to be a string]

#localStorage.getItem(‘name’) [now we have to just set Item. We can console log it]

If there is no key by the name then it will return null

#localStorage.removeItem (‘name’)

Session storage has the same functionality like local storage.

If we want to update the value of the key just type

#sessionStorage.setItem(‘name’,’bob’) [simply set the item again it will overrite]

But cookies do not have that interface to interact

#document.cookie = ‘name=kyle’ [we can set that like this. And if we got to cookies we can see that]

What if we want to expire date ?

#document.cookie = ‘name=kyle; expires=’ + new Date(2020 , 0 , 1). toUTCString()

[this is how we set expire date. Zero is the index of January month]

Now if we add a new cookiye by the same name and value then it will not overrite like seesion or local just add a new cookie. Simple.

**Cache**

* Reduce Latency
* Cuts down the bandwidth
* Reduced load on the server

Browser cache is private to the user,depend on the cachin header

Proxy cache is cache of the proxy level.

Reverse proxy is closer to server, set up by the server

There are lot of diff types of Cache headers.

**OAuth Authentication and Authorization (Java Brain)**

Open standard authoraization.

From consumer to server to access token

From 3rd party to some ssite ilke fb google and then they return a access token to 3rd party. By authenticate and authorized you. Access token get expires too by OAUTH 2.0

oAuth was created to authorize not another person but another service. Oauth 2.0 is widely used.

Valet key to just start and shut down the car. So valet is just there to park the car. So the oauth flow is 3rd party system go to google for some of my picture and google comes to me that there is service wants these photos and doc of yours, what to do ? then user give permission or allow. Then google sends an access token. Just like the valet. Simple. So next time 3rd party just use the token. Simple.

This access token contains info about all the allower permission and also needs something that service can verify but cannot modify. It’s a token format JWT.

**JWT (JSON web TOKEN)**

Is there for authorization not authentication. To client post username and pass. Server create a JWT for a user and send JWT to browser. User send request with JWT and serer verify JWT signature and send back response.

**Verification and Validation**

* Verification = are you building it right ? so its every phase . Done by developer. Unit+integration testing. Static + Dyanamic
* Validation = have you built it right ? so its after everytiing is done. Done by tester. System testing. Just Dynamic.

**Minification**

Is a process of minimizing code and markup in your web pages and script file. Its heps to reduce load times and bandwith usage. Its speed up site UI to have better experice. So user can have the access of the website with limited data plan.

To make the code readable developer uses space/comment and extra code or variable in HTML CSS JS. Which takes time to convert or code in js. So removing all that would be good for server or browser to convert by the toll of less readable code and minimizing the file size.

**Obfuscation**

Is something to make something harder to understand. So its used to prevent people from understanding the meaning of something. And is often used with computer code to help prevent successful reverse engineering and/or theft of a products functionality. Obfuscation does not include no cryptography key just the algorithm it self. It protect intellectual property.

**CORS**

Cross origin resource sharing policy. Who ever sending request to whomever, if the domain of them two are different then by default request will get rejected. So sender and receiver is diff ? then browser will reject the response. Lets see how to fix it

When we request some meta value goes to server and same for response too. But when domain is diffrenent that means response does not have “access control aloow” meta value. So to fix it we have to change something from the server not from the browser. We have to make sure server allow our browser request and give that meta data so that we can accept. So with the help of npm and installing cors we can make sure that server allow that particular one url which we want to give. SIMPLE. And now if we go to dev toll network tab then we can see that “Access control allow” meta data is there in response. Server can edit the type of api request methods if its out of it then it will again send error