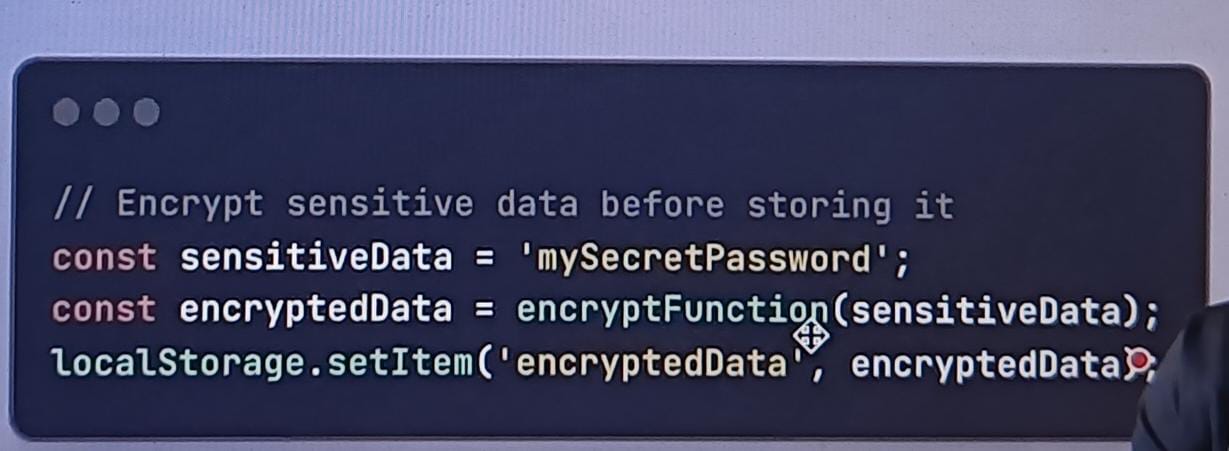
**Client-Side Storage Security**

1. Storing sensitive data on client storage –

This storage can be any, it can be cookie it can be local storage it can be index db, so we are talking about credentials, api key, or session token any of those things that we are storing in local, indexdb or cookies something has to be think twice.

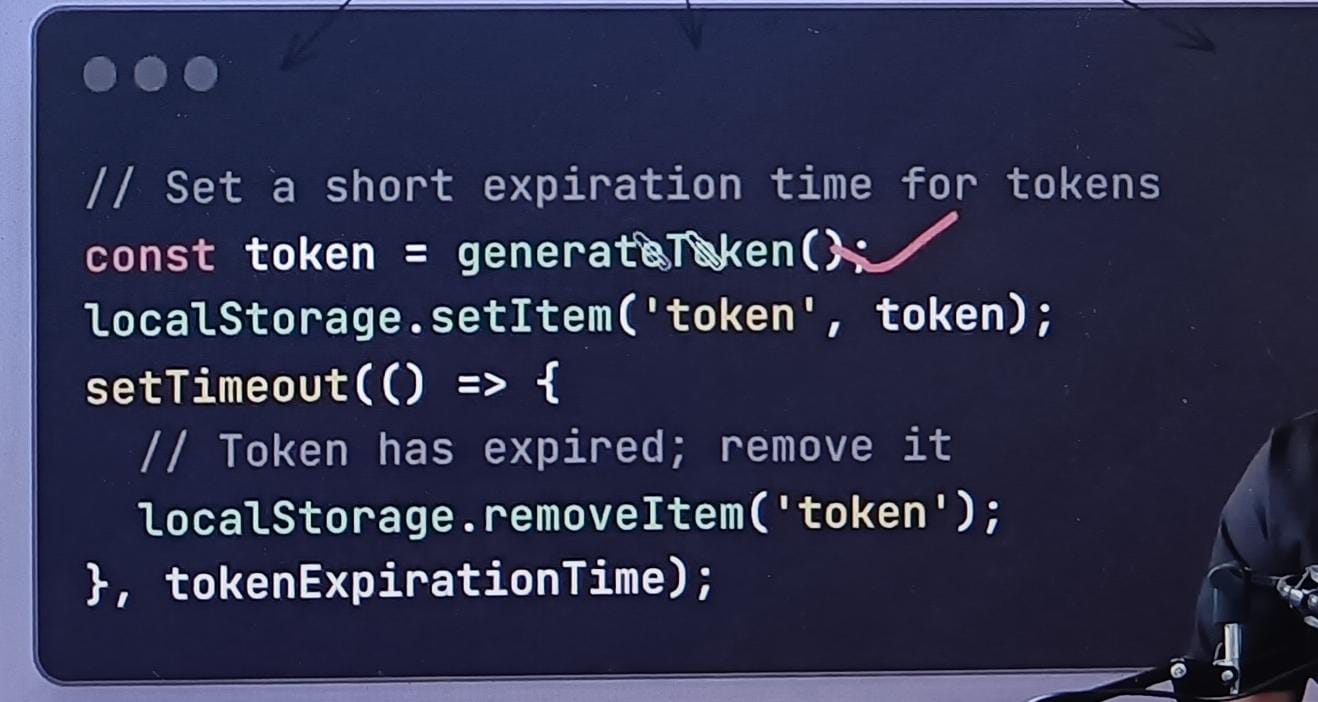
* Now what can be done, in case we have to store first of all it is recommended if we are dealing with such kind of data it is better if we store the data on the server instead of putting the data on the client.
* Encrypt your data, suppose if there is a sensitive data that need to be store like



So there is a mySecretPassword, some secret has to be stored into your local storage what you do before putting actual data over here you can put your encrypted data write any kind of encryption decryption logic which is more secure we can use, it can be cryptographic or it can be any such thing basically where you can encrypt and decrypt if required, so you put it in the encrypted format so that anyone who have direct access to the data may not be able to derive it until and unless known the algorithm, knowns the shalt basically in order to decrypt it.

Now there are always chances that it can be leak because all of these things are done on the client but still it’s a bit Sefer aa compare to other.

* Token Expiry –



So, what we are doing in this case I am generating some token, that token has to be kept into the local storage, along with that the moment that the token is actually set there is SetTimeout which is basically initiated, now after some time this token is automatically going to deleted, now it has loopholes in terms of what if the browser gets closed and alls, I am not going to get into those details.

1. Authentication –

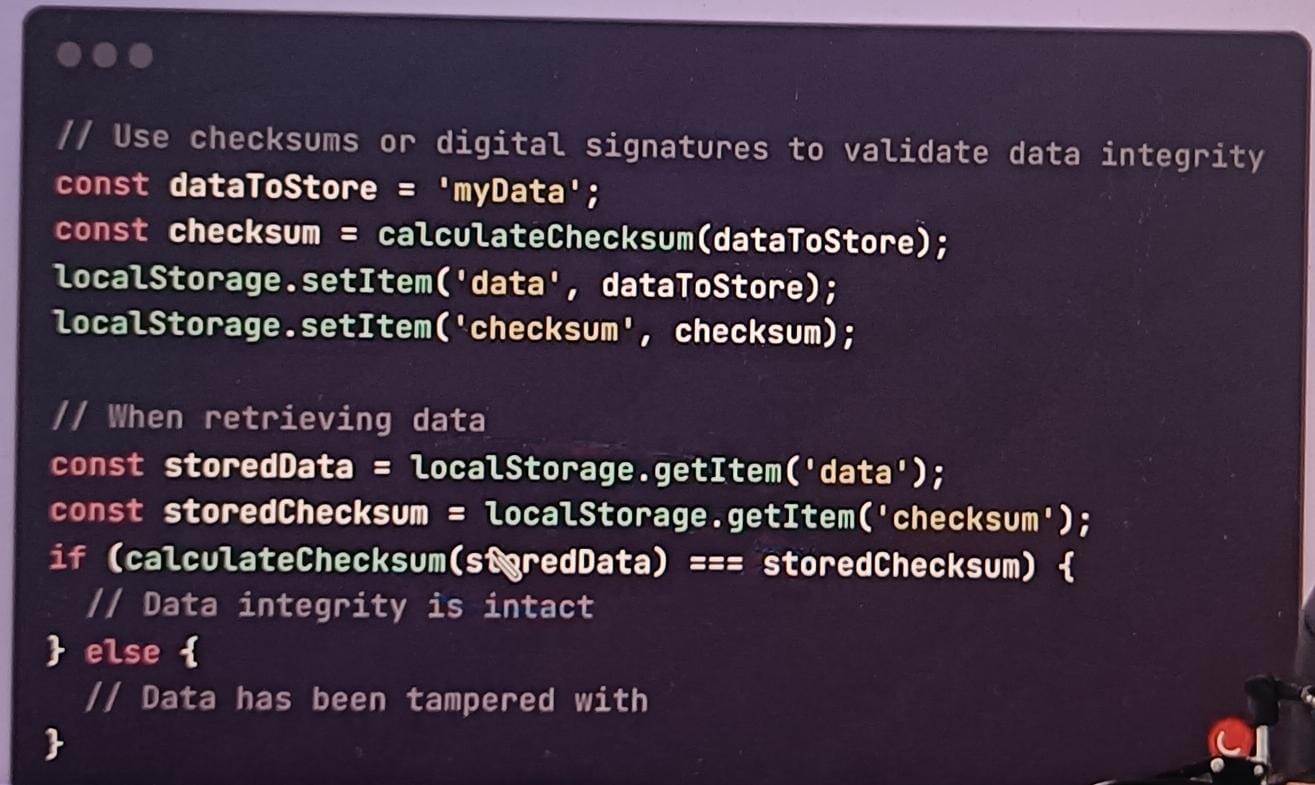
* Please use a right set of authentications whenever you talk about storing any authentication token on the client, it is recommended to use something called JWT OR OAuth.

Instead of directly putting the credentials or token if you have JWT or OAuth token, that is still better but again not 100% guaranteed safe but you are actually getting into something which is better that the normal solution that you have.

* You should have your token expiry, your session token or any of these token that you have related to the authentication you should define the right set of expiry which is very very important in this case.
* Anything you are dealing with the authentication based on some token which is stored on the client it is also recommended that you have multi actor authentication where either you take extra otp, take some extra information that the user is basically logged in properly or not logged in whenever you see something critical going on or something unexpected basically going on.

1. Data Integrity –

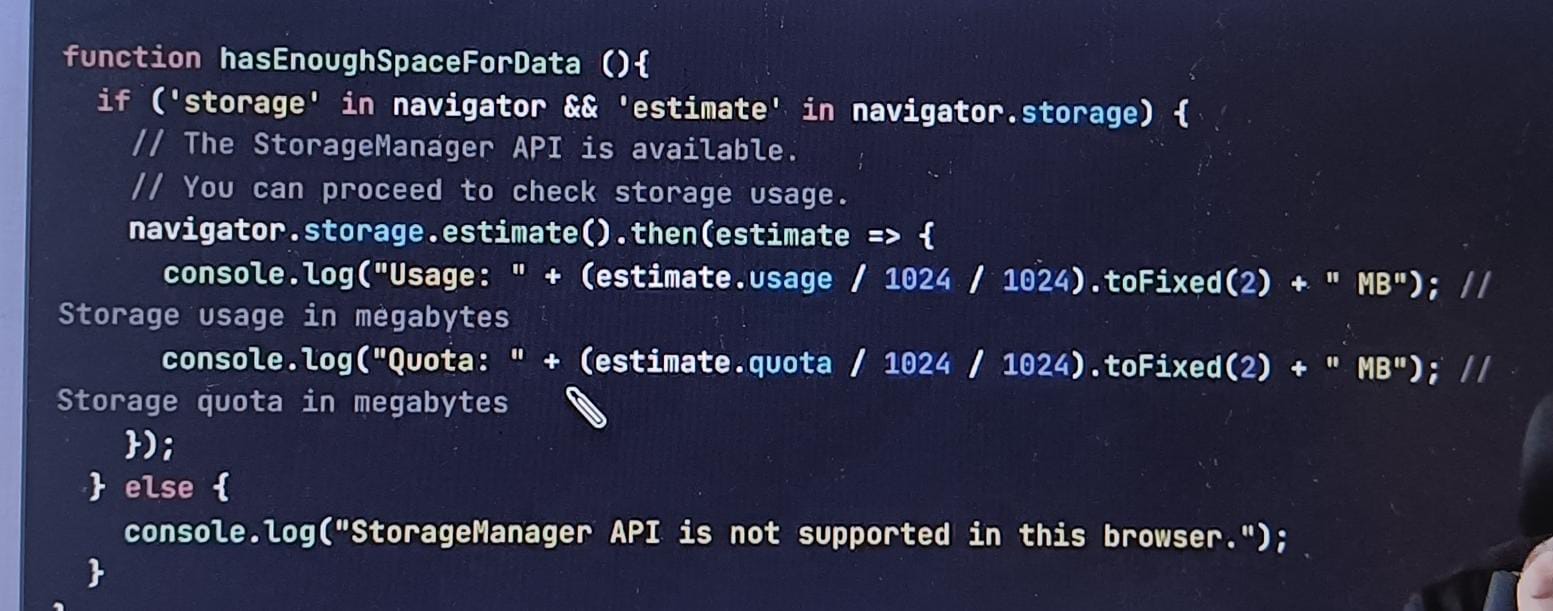
* Consider example you have to store some information, we talked about you can encrypt some data and you can put in the local storage and when you are reading it back you decrypt, what if someone got that strategy ok this is the algo that you are using and the person basically able to manipulate the data which is stored in your local storage, now in such cases adding some data integrity using some checksum kind of mechanism it can help you.
* So, create a checksum and put that checksum along with your data, so what is that checksum? You have original data, you create a checksum related to that now whenever with same data you are creating another checksum that will always be the same, in case if someone temper your data which is your encrypted data then checksum for that will be different. Now you will be able to identify, now person has to go to another level which is not just to decrypt but also to find out where that checksum is, which checksum corresponds to what kind of data, there so many things that person have to understand in order to get and reveal like how to temper the things at a multiple layer.



In this case we have some data that need to be stored, what we do with that data we are creating some checksum, what you do you store the data along with that you store the checksum of that data also (line no 3&4), now when you are retrieving that data you check if the checksum of that stored data is actually same (line no 7) if same no one have temper the data in between, if not then something has to be fixed and you say no no no this is the data is tempered and I am not going to use the data as it is, that is again all of these technique on the client layer to make it more future proof basically that is we are trying.

1. Storage Limit–

* We have typically when we store the cookie and token in local storage we have 5 to 10 mb of data that can be stored on the client-side local storage.
* Now in this case what can be done, one of the ways is your browser basically helps you how much storage you have already consuming.
* So there are certain apis which are provided by browser which are storage related api for example.



So inside our navigator we have something called storage, and we have something called estimate inside our storage(if statement line no2) you actually check if that is present and they you can calculate the estimate of how much basically this usage are available(console logs line no 4,5), depending on the quota so either you calculate the usage or the total quota both you can calculate, you can calculate how much it is used and what is the total quota which is available because the total quota is different browser to browser which is very important so in this particular case you decide this is my used and this is my total now I will decide how much basically it can store so your data is not lost, those kind of things makes lot of sense.\

1. localStorage, SessionStorage, indexedDB, Cookie, Cache –

localStorage have kind of 5-10 mb per domain

sessionStorage also 5-10 mb

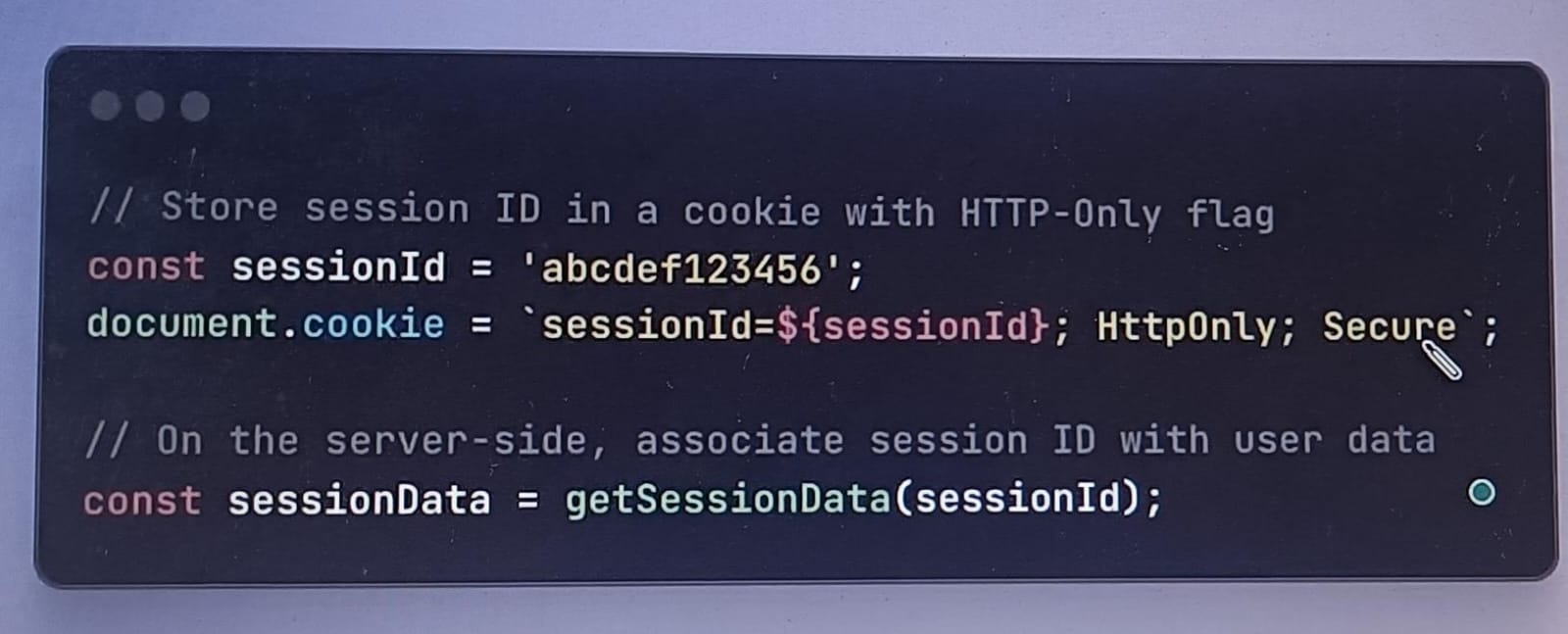
indexedDB also have 50mb-100mb approximate

Cookie – 4kb – 20kb per cookie

Cache - 100mb data

1. Session Management –

We are talking about any sessions or any token that we store those tokens, especially the cookies that I am talking about those tokens should have http only and secure kind of thing so in such case there are two things we have to understand.



When we set a header or a cookie with this particular token along with that we say this is only for HttpOnly and a secure which means I can access only on the https 2nd this is HttpOnly what does that mean? I can access this on the server I cannot use JavaScript code in order to access this any cookie which is set using HttpOnly.