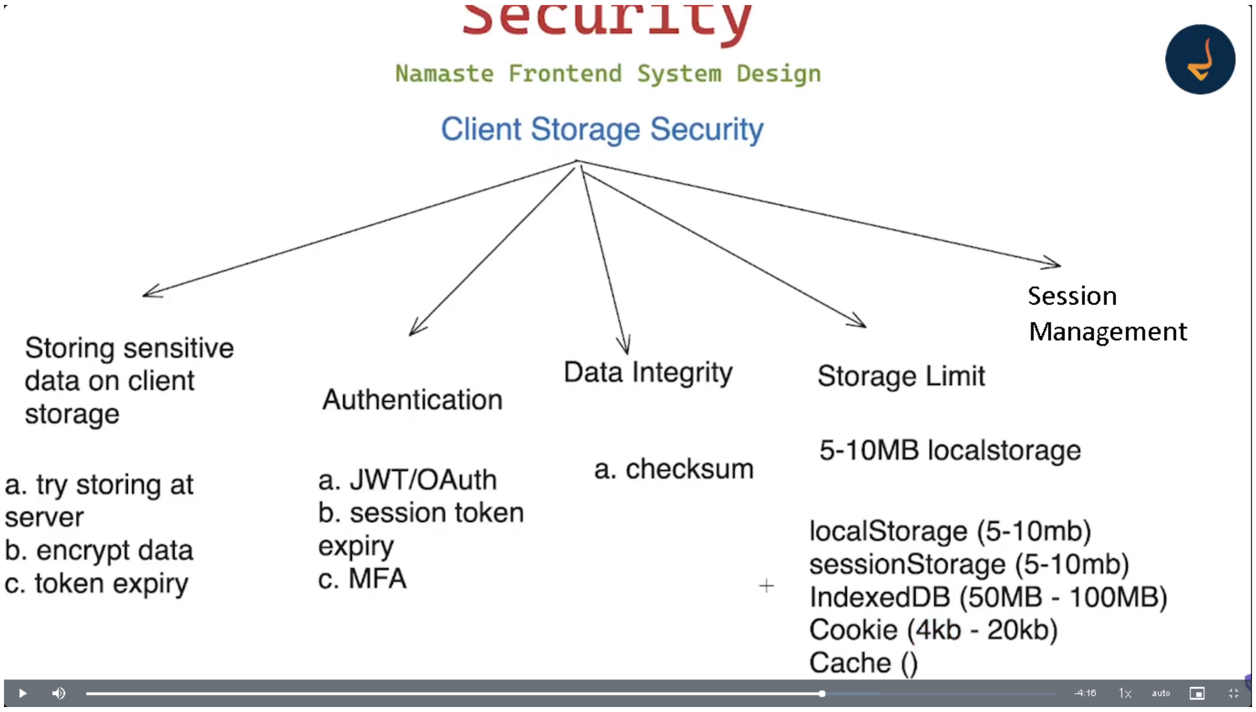
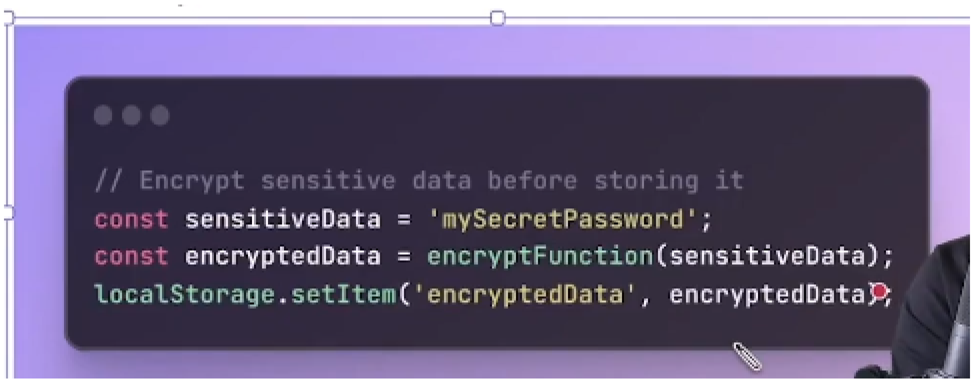
Client-side security



Storing sensitive data on client storage

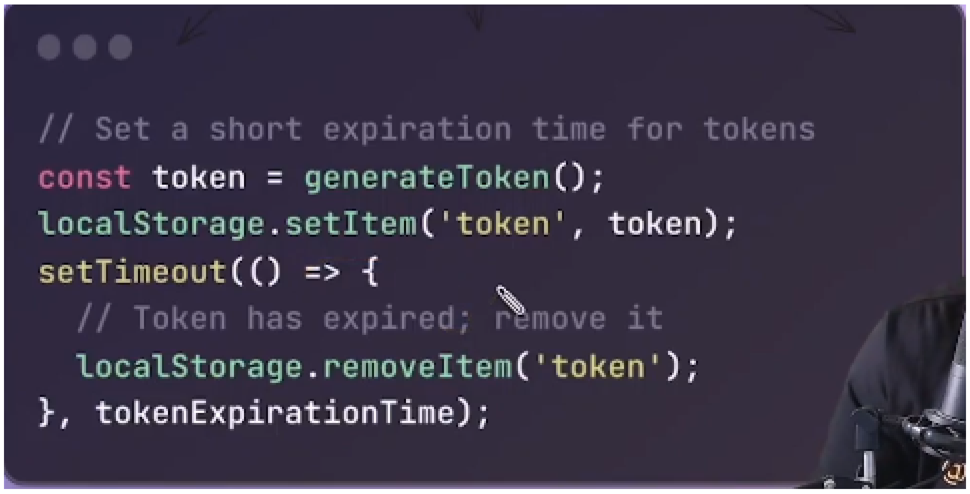
try storing at server

encrypt data example



```
// Encrypt sensitive data before storing it
const sensitiveData = 'mySecretPassword';
const encryptedData = encryptFunction(sensitiveData);
localStorage.setItem('encryptedData', encryptedData);
```

set token expiry example



```
// Set a short expiration time for tokens
const token = generateToken();
localStorage.setItem('token', token);
setTimeout(() => {
  // Token has expired; remove it
  localStorage.removeItem('token');
}, tokenExpirationTime);
```

Authentication

JWT/OAuth

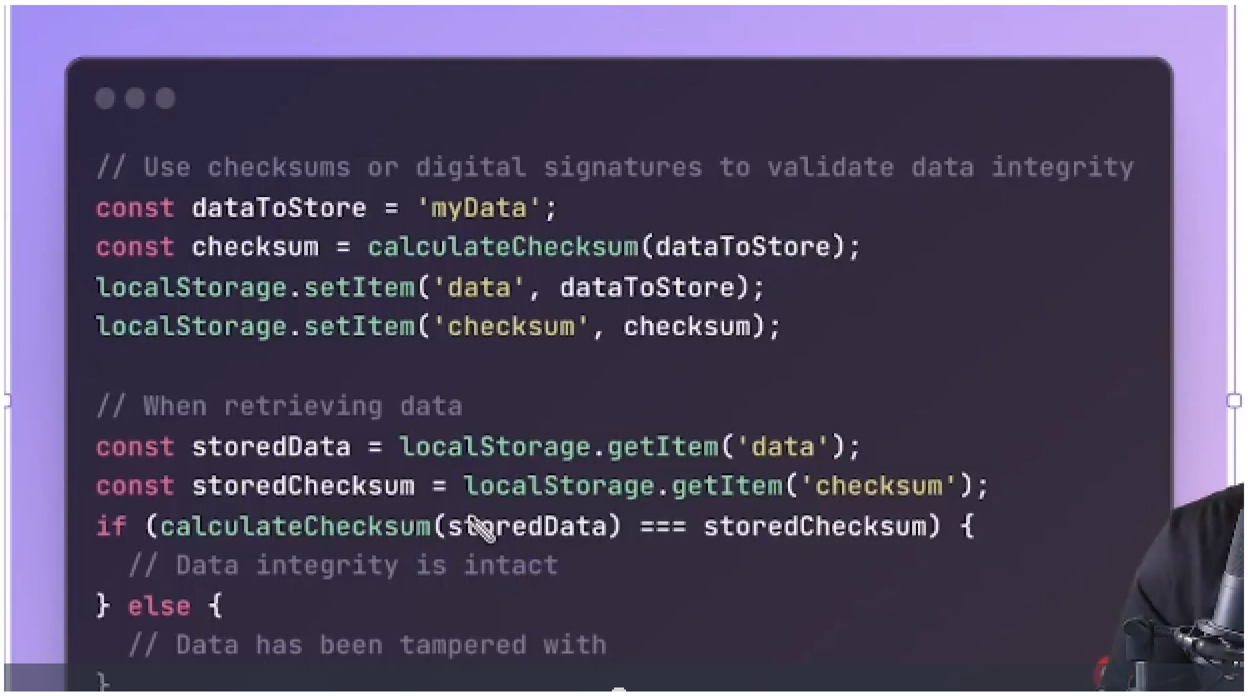
Set session token expiry

MFA - multifactor authentication

Data integrity

The property that data has not been altered in an unauthorized manner. Data integrity covers data in storage, during processing, and while in transit.

Using checksum to ensure data integrity



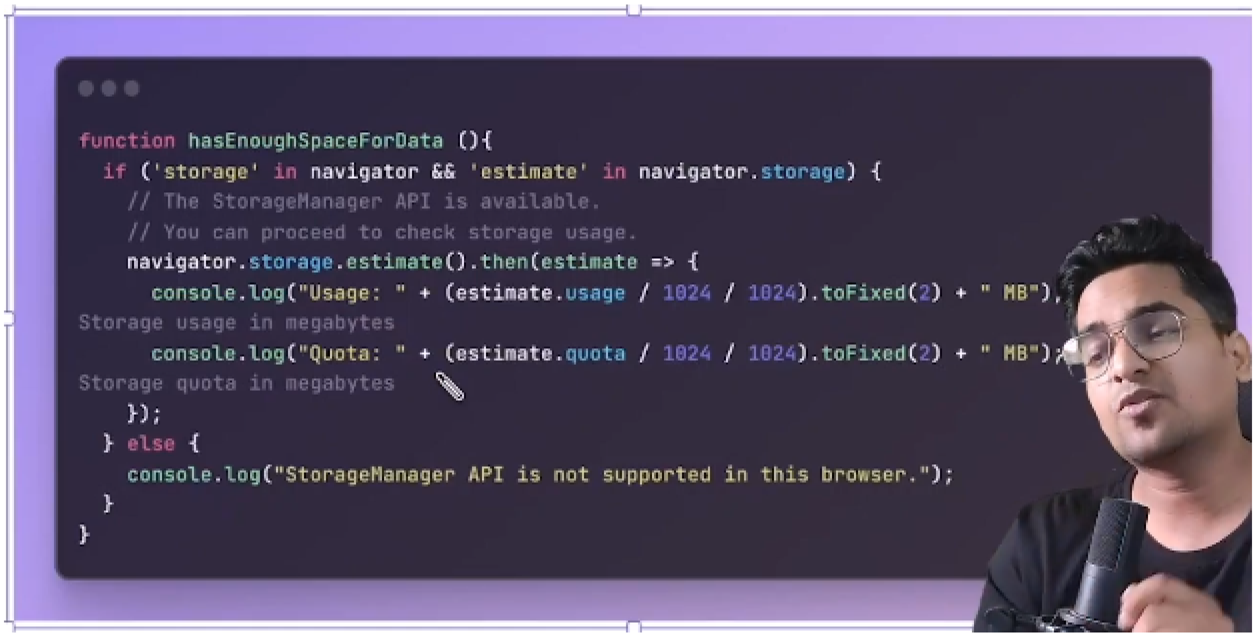
```
// Use checksums or digital signatures to validate data integrity
const dataToStore = 'myData';
const checksum = calculateChecksum(dataToStore);
localStorage.setItem('data', dataToStore);
localStorage.setItem('checksum', checksum);

// When retrieving data
const storedData = localStorage.getItem('data');
const storedChecksum = localStorage.getItem('checksum');
if (calculateChecksum(storedData) === storedChecksum) {
  // Data integrity is intact
} else {
  // Data has been tampered with
```

Storage limit

There are basically 2 major impacts of storing data on client

1. It affects performance
2. Storing more data than browser’s capacity can lead to data loss

Below function is an example how one can get current data usage and available data quota in browsers

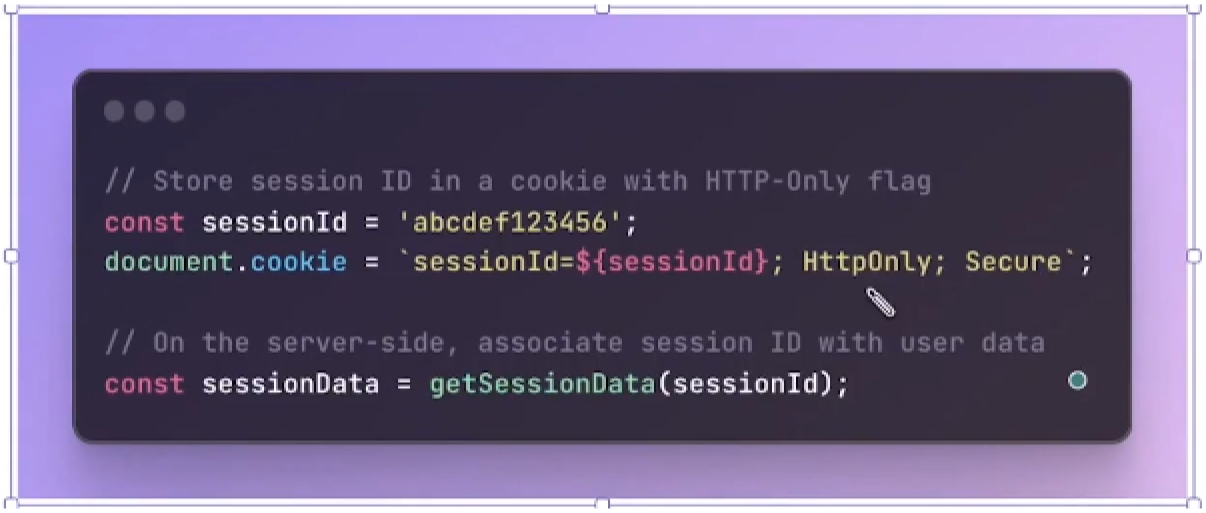


```javascript
function hasEnoughSpaceForData (){
  if ('storage' in navigator && 'estimate' in navigator.storage) {
    // The StorageManager API is available.
    // You can proceed to check storage usage.
    navigator.storage.estimate().then(estimate => {
      console.log("Usage: " + (estimate.usage / 1024 / 1024).toFixed(2) + " MB"),
Storage usage in megabytes
      console.log("Quota: " + (estimate.quota / 1024 / 1024).toFixed(2) + " MB");
Storage quota in megabytes
    });
  } else {
    console.log("StorageManager API is not supported in this browser.");
  }
}
```

Session Management

We should manage sessions and cookies appropriately.

Below is example of setting appropriate headers on cookies. We set the same headers in iFrame protection module as well.



```javascript
// Store session ID in a cookie with HTTP-Only flag
const sessionId = 'abcdef123456';
document.cookie = `sessionId=${sessionId}; HttpOnly; Secure`;

// On the server-side, associate session ID with user data
const sessionData = getSessionData(sessionId);
```