Chapter 10 - Network Requests & Storing Data

JavaScript can be used to send and retrieve information from the network when needed (AJAX)

Fetch API

fetch is used to get data over the network

```
let promise = fetch(url, [options])
```

↳ without options, a get request is sent

Getting a response is a 2-stage process

1. An object of Response class containing "status" & "ok" properties

   status - The http status code, eg. 200
   ok - boolean, true if the HTTP status code is 200 - 299

2. After that we need to call another method to access the body in different formats:

   response.text() → Read & return the text
   response.json() → parse the response as JSON

   Other methods include response.formData(), response.blob(), response.arrayBuffer() etc.

Note - We can use only one body reading method. example if we have already got the response with response.text() then response.json() wont work

Response headers

The response headers are available in response.headers

Request headers

To set a request header in fetch, we can use the headers option.

```
let res = fetch(url, {
            headers: {
              Authentication: 'secret'
            }
          });
```

POST requests

To make a POST request, we need to use fetch options

1) method → HTTP-method, e.g. POST
2) body → the request body

```
let response = await fetch('/url', {
                 method: 'POST',
                 headers: {
                   'Content Type': 'application/json'
                 },
                 body: '{"a": "harry"}'
               });
let result = await response.json()
```

summary: browser le request gryo server lai tehha browser ma cookies ko rup ma key vallue pair save vako hunxa jun server le nae browser lai pathako hunxa
ani next time browser le tesmae request grdae sangae cookies ni janxa ani server le paila kae manxe ko req ho ki haenw ki sab chinxa

JavaScript Cookies

Cookies are small strings of data stored directly in the browser.
In JavaScript, document.cookie provides access to cookies.

Cookies are set by a web server using the Set-Cookie HTTP-header. Next time when the request is sent to the same domain, the browser sends the cookie using the Cookie HTTP-header. That way the server knows who sent the request

We can also access cookies using document.cookie property:

```
alert ( document.cookie)
```

↳ Contains key=value pairs delimited by a ;

Writing to Cookie

An assignment to document.cookie is treated specially in a way that a write operation doesn't touch other cookies

key value pairs rahexa tala chai key user value harry

```
document.cookie = "user=Harry"
```

↳ updates only cookie named user to Harry

Quick Quiz: Print all the cookies on twitter.com

encodeURIComponent

encode uri component lai chai key or value ma jun data dinxam tyo jun format accept grna help grxa jasto ki key ma rosh+=+; esto ni accept k ani value ma roshni#4342 esto ni accept je symbol rakheni sab accept grna chai encodeURIComponent

This function helps keep the valid formatting. It is used like this:

```
document.cookie = encodeURIComponent(name) + '=' +
                  encodeURIComponent(value)
```

This way, the special characters are encoded

Cookie options

Cookies have several options which can be provided after key=value to a set call like this:

```
document.cookie = "user=John; path=/a; expires=
                   Tue, 29 March 2041 03:18:22 GMT"
```

path option makes the cookie visible at /a, /a/b etc.
expires sets the cookie expiration time

Note:

1. The name=value pair, after encodeURIComponent, should not exceed 4KB
2. Total no of cookies per domain is limited to around 20+ (Exact number is browser dependent)

localStorage

localStorage is a web storage object which are not sent to server with each request
This data survives a full page refresh and even a full browser restart

These are the methods provided by localStorage

1. setItem(key, value) → store key/value pair
2. getItem(key) → get the value by key
3. removeItem(key) → remove the key with its value.
4. clear() → delete everything
5. key(index) - get the key on a given position.
6. length - the number of stored items

We can get and set values like an object

```
localStorage.one = 1

alert(localStorage.one)

delete localStorage.one
```

Important Note

1. Both key and values must be strings
2. We can use the two JSON methods to store objects in localStorage:

```
JSON.stringify(object) → Converts objects to JSON strings
JSON.parse(string) → Converts string to objects
                     (must be a valid JSON)
```

Session Storage

Used less often than localStorage. Properties and methods are same as localStorage but:

1. The sessionStorage exists only within the current browser tab. Another tab with same page will have a different storage.
2. The data survives page refresh, but not closing/opening the tab.

Storage Event

When the data gets updated in localStorage or sessionStorage, storage event triggers with these properties:

1. key → The key
2. oldValue → Previous value
3. newValue → New value
4. url → Page URL
5. storageArea → local or sessionStorage

We can listen the onstorage event of window which is triggered when updates are made to the same storage from other documents.