Topic: Browser Object Model (BOM)

The **Browser Object Model (BOM)** is a set of objects provided by web browsers to interact with the browser itself, beyond just manipulating the content of a web page. It provides JavaScript access to various components of the browser environment, such as the browser window, history, location, and more.

1. Window Object

The window object represents the browser's window. All global JavaScript objects, functions, and variables automatically become members of the window object.

2. **Window Object**:

Get the width and height of the browser window

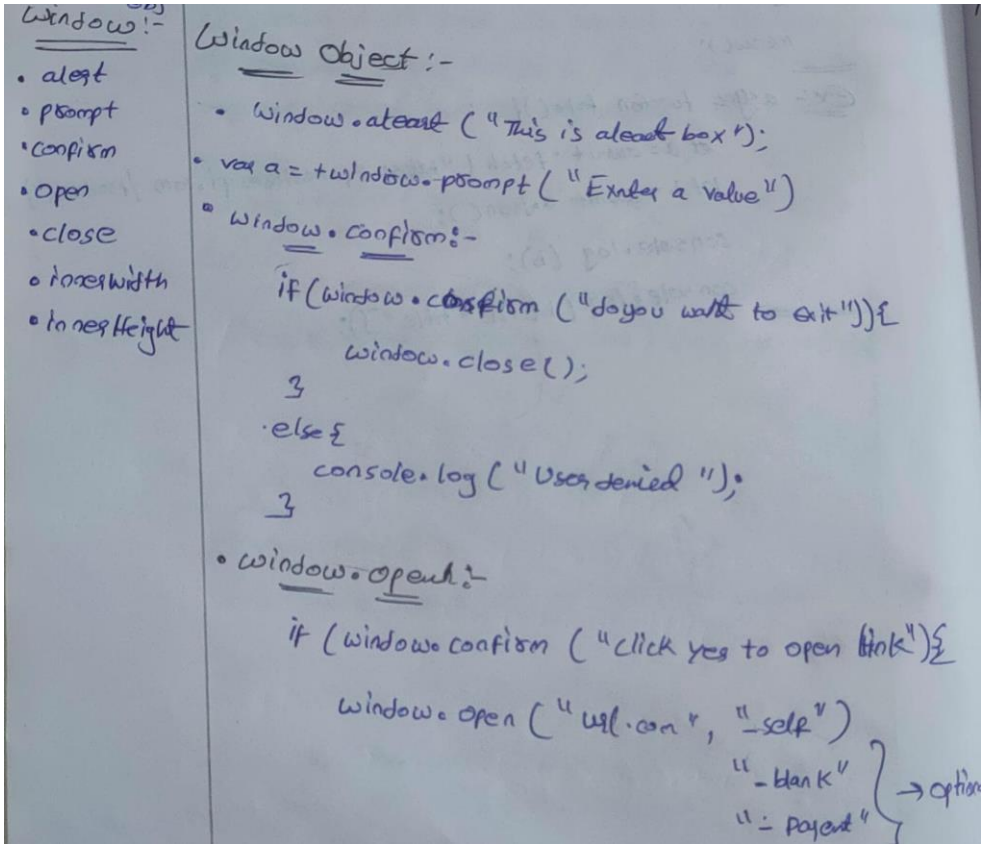
```
console.log(window.innerWidth);
console.log(window.innerHeight);
window.open("https://example.com", "_blank", "width=600,height=400");
```

3. Navigator objects

The Navigator object in JavaScript provides information about the browser's name, version, platform, and capabilities.

4. Location objects

It provides properties that allow you to access and manipulate different parts of the URL



- innerHeight & innerWidth :-

```
console.log (window.innerHeight);
console.log (window.innerWidth);
```

Ex:-
```
if (window.innerWidth < 600){
    document.body.style.backgroundColor = "red";
}
```

~~consolelog (window.innerW~~

Navigator objects:-

It provides the information about the browser's name, version, platform and capabilities.

Navigator obj:-
- geolocation
- appName
- appCodeName
- appVersion
- userAgent
- platform
- online
- cookieEnabled

- Navigator.online (say online or offline)

Ex:-
```
console.log (navigator.online);
if (navigator.online){
    console.log ("browser is online");
}
else{
    console.log ("browser is offline");
}
```

- ~~navigator.appCod~~ appCodeName appName AppVersion, userAgent

```
console.log(navigator.appCodeName); // Mozilla
console.log (navigator.appName); // Netscape
console.log (navigator.appVersion); // 5.0 (Linux ---)
console.log (navigator.userAgent); // Mozilla /5.0 version
console.log (navigator.platform); // ~~Win~~ Win32
```

- geolocation :-

```
if (navigator.geolocation){
    navigator.geolocation.getCurrentPosition(function(position){
        console.log("latitude:" + position.coords.latitude);
        console.log("longitude:" + position.coords.longitude);
    });
}
else{
    console.log("geolocation is not supported in this browser")
}
```

- cookieEnabled :-

checks cookies are Enabled or not.

Ex:-

```
console.log(navigator.cookieEnabled);
// true.
```

Location obj's:-
- href
- pathname
- hostname
- protocol

Location Objects:-

```
console.log(location.href); // https://127.0.0.1:5500/day22/main.html
console.log(location.pathname); // 127.0.0.1
                                //  ↑   ↓
console.log(location.hostname); // day22/main.html
console.log(location.protocol); // http
```

```
if (window.confirm("redirect")){
    location.href = "day22.html";
}
```

→ this will redirects to another page.

Screen Objects:-

screen objects in javascript provides information about the users screen or display, such as width, height, colors, depth and pixel density.

screen objs:-
- height
- width
- pixelDepth
- colorDepth

Ex:-

```
console.log (screen.height);      // 1080
console.log (screen.width);       // 1920
console.log (screen.pixelDepth);  // 24
console.log (screen.colorDepth);  // 24
```

History Objects:-

History objs:-
- history.back();
- history.forward();
- history.go();

History object in Javascript represents the user's navigation history for the current browser window it allows you to navigate back and forward through the history stack.

Ex:-

```
<button onclick="this()"> click me </button>

<script>
function this(){
    window.open("1.html","_self");
}
</script>
```

1.html:-

```
<button onclick="gb()"> click me </button>

<script>
function gb(){
    window.history.back();
}
</script>
```

```
history.back();      // moves the browser back one page
history.forward();   // moves the browser forward one page
history.go(-2);      // moves the browser back two pages.
```

Cookie Objects:-

cookies are small piece of data stored in the users web browser. They are Typically used by websites to remember users preferences, authentication status and other information related to their browsing session.

Timing functions:-

Timing functions:-

setTimeout()
setInterval()
clearInterval()

Timing functions are crucial for managing when and how often certain blocks of code execute.

- setTimeout() // takes time to execute
- setInterval() // Executes for every time period

Ex:- `<button onclick="hello">click me</button>`

```
function hello(){
    console.log("hello world");
}
setTimeout(hello,5000);
```

Ex:-

```
function hello(){
    console.log("hello world");
}
setInterval(hello,1000);
```

- clearInterval:-

```
function hello(){
    console.log("Hello world");
}
let a = setInterval(hello,1000)
function hik(){
    clearInterval(a);
}
```

**Example:-

```
for (var i=0; i<10; i++) {
    setTimeOut ( ()=>{
        console.log(i);
    }, 1000)
}
```

o/p:- 10 (10 times)

→ difference is Scoping.
var has global scope
let has block scope.

**Example :-

```
for (let i=0; i<10; i++){
    setTimeOut (()=>{
        console.log(i)
    };0)
}
```

o/p:- 0 to 9

Session storage & Local storage:-

Session storage:- is a part of web storage API in web browsers that provides a way to key value pairs locally on the client-side.

- Data stored in session storage is cleared when the page session ends (until browser closes).

Ex:- 1.html:-

```
let a = "john";
window.localStorage.setItem ("store", a);
```

2.html:-

```
let a = localStorage.getItem ("store");
console.log (a); //john
```

(A) setItem ("store", "something")
// something

Ex:- 1.html:-

```
let a={
    name: "john";
    age: 25
};
localStorage.setItem("store", JSON.stringify(a));
```

↳ to store object in local storage

2.html:-

```
let aa = localStorage.getItem('store');
console.log(JSON.parse(aa));
```

↳ to convert JSON to object.

Example:-

a.html:-

```
<input type="text" id="inp1"><br>
<input type="text" id="inp2"><br>
<button onclick="fun()"> Submit Information </button>
<script>
    function fun(){
        var inp1 = document.getElementById("inp1").value;
        var inp2 = document.getElementById("inp2").value;
        localStorage.setItem("inp1", inp1)
        localStorage.setItem("inp2", inp2)
        window.open("day22.html", "_self")
    }
```

b.html:-

```
let aa1 = localStorage.getItem('inp1');
let aa2 = localStorage.getItem("inp2");
var h1 = document.createElement("h1");
h1.innerText = aa1 + aa2;
document.body.appendChild(h1)
console.log(aa1, aa2);
```