Below is a **clean, structured, and internally explained rewrite** of **Browser Object Model (BOM)**, written as **high-quality study notes** suitable for **NoteLM, exams, interviews, and real-world understanding**.

I've **kept your content**, but:

- clarified **internals**
- removed ambiguity
- added **mental models**
- highlighted **exam + interview traps**
- separated **DOM vs BOM clearly**

🌐 Browser Object Model (BOM) in JavaScript — Complete Guide

1️⃣ What is the Browser Object Model (BOM)?

The **Browser Object Model (BOM)** is a collection of **objects provided by the browser** that allows JavaScript to interact **with the browser itself**, not the web page content.

📌 BOM lets JavaScript:

- Control browser windows
- Read browser information
- Navigate URLs
- Access history
- Get screen details
- Log debugging information

👉 Key Point

BOM deals with **browser environment**, while DOM deals with **HTML content**.

2️⃣ Is BOM Standardized?

❌ No

- BOM is **not officially standardized**
- Behavior may differ slightly across browsers (Chrome, Firefox, Safari)
- However, **most BOM APIs are de-facto standard** and widely supported

📌 DOM is standardized by W3C

📌 BOM is implemented by browsers

3️⃣ BOM vs DOM (Very Important)

Feature	BOM	DOM
Purpose	Browser interaction	Page (HTML) interaction

Feature	BOM	DOM
Scope	Window, history, location	Elements, attributes
Root Object	`window`	`document`
Example	`location.href`	`document.getElementById()`

👉 **DOM is part of BOM** because `document` lives inside `window`.

4️⃣ BOM Object Hierarchy (Mental Model)

```javascript
window
 ├── document  (DOM)
 ├── screen
 ├── history
 ├── navigator
 ├── location
 ├── console
```

📌 **Everything in BOM starts from `window`**

5️⃣ The `window` Object (Root of BOM)

What is `window`?

- Represents the **browser window or tab**
- Global object in browsers
- All **global variables & functions become window properties**

```js
var x = 10;
console.log(window.x); // 10
```

```js
function greet() {}
window.greet(); // valid
```

Accessing Window Properties

```js
```

```
window.innerWidth
window.innerHeight
```

or simply:

```
js

innerWidth
innerHeight
```

📌 `window.` prefix is optional in browsers.

6️⃣ The `document` Object (DOM Root)

What is `document`?

- Represents the **HTML page**
- Root of the **DOM tree**
- Allows reading & manipulating HTML elements

```
js

document === window.document // true
```

Example: DOM Manipulation

```
html

<div id="text">Original Text</div>

<script>
  const el = document.getElementById("text");
  el.innerHTML = "Text Changed!";
</script>
```

📌 DOM operations do **not** affect browser behavior

📌 BOM operations affect browser behavior

7️⃣ The `screen` Object

Purpose

Provides information about the **user's physical screen**, not browser window.

```
js
```

```
screen.width
screen.height
```

📌 Values depend on:

- Device
- Resolution
- OS scaling

Example

```
js

console.log(screen.width + " x " + screen.height);
```

⚠️ Interview trap

`screen.width ≠ window.innerWidth`

8️⃣ The `history` Object

Purpose

Manages the **session history** (pages visited in the tab)

```
js

history.back();    // previous page
history.forward(); // next page
history.go(-1);    // same as back
```

📌 You **cannot read URLs directly** for security reasons.

Why restricted?

To prevent:

- History spying
- Privacy leaks

9️⃣ The `navigator` Object

Purpose

Provides information about the **browser & device**

Common properties:

```
js
```

```
navigator.userAgent
navigator.language
navigator.cookieEnabled
navigator.onLine
```

📌 Used for:

- Feature detection
- Language preferences
- Online/offline checks

⚠️ **Avoid browser sniffing**

Use **feature detection** instead.

🔟 The `location` Object (Very Important)

Purpose

Represents the **current URL**

```
js

location.href     // full URL
location.host     // domain + port
location.hostname  // domain
location.protocol  // http / https
location.pathname  // path
```

Redirecting User

```
js

location.href = "https://example.com";
```

📌 This **reloads the page**

1️⃣1️⃣ The `console` Object

Purpose

Used for **debugging & logging**

```
js

console.log("Hello");
console.warn("Warning");
```

```
console.error("Error");
console.table({a:1, b:2});
```

📌 Exists only for developers

📌 Should not be relied on for production logic

1️⃣2️⃣ How JavaScript Resolves Global Access

```
js

document
```

Internally becomes:

```
js

window.document
```

Same for:

- `console`
- `location`
- `history`

This is called **global object binding**.

1️⃣3️⃣ Security Restrictions in BOM

JavaScript **cannot**:

- Read full browsing history
- Access cookies marked `HttpOnly`
- Get device hardware details
- Control browser UI deeply

📌 These restrictions protect users.

1️⃣4️⃣ Common Interview Questions

Q1. Is BOM part of JavaScript?

➡️ No, it is provided by the browser.

Q2. Is document part of BOM or DOM?

➡️ DOM object, but accessed through BOM (`window.document`).

Q3. What is the root of BOM?

➡️ `window`

Q4. Can BOM differ across browsers?

➡️ Yes (not standardized).

1️⃣5️⃣ One-Line Summary (Perfect for Exams)

> **The Browser Object Model (BOM) provides browser-specific objects like window, location, history, and navigator that allow JavaScript to interact with the browser environment rather than the web page content.**

⏭️ What Should We Do Next?

Choose one:

1️⃣ Window Object — methods, lifecycle, events (deep)
2️⃣ Location vs History (navigation internals)
3️⃣ Navigator object & feature detection
4️⃣ DOM vs BOM vs Web APIs (architecture view)

Reply with **1 / 2 / 3 / 4** and we continue **one topic at a time, deeply**.