Il Modello di Oggetto del Browser (BOM)

Il **Browser Object Model (BOM)** è una struttura che permette di interagire con il browser web. Esso include una serie di oggetti che rappresentano l'ambiente del browser, come la finestra del browser stesso (`window`), il documento caricato al suo interno, la posizione dell'URL corrente, la cronologia, i cookie, i timer, ecc.

A differenza del **Document Object Model (DOM)**, che si occupa della struttura e del contenuto del documento HTML o XML, il BOM si concentra sull'ambiente circostante il documento, come le funzionalità fornite dal browser.

Principali Componenti del BOM

1. `window`: rappresenta la finestra del browser e include tutte le altre proprietà del BOM.
2. `navigator`: contiene informazioni sul browser e sul sistema operativo.
3. `location`: fornisce informazioni sull'URL corrente e permette di navigare ad altre pagine.
4. `history`: consente di interagire con la cronologia del browser.
5. `screen`: fornisce informazioni sullo schermo del dispositivo.
6. `console`: per la registrazione di messaggi diagnostici.
7. **Timer**: funzioni come `setTimeout` e `setInterval` per eseguire codice dopo un determinato intervallo di tempo.

1. Oggetto `window`

L'oggetto `window` è l'oggetto globale nel BOM e include tutte le proprietà e i metodi del BOM.

Esempio: Apertura di una nuova finestra

```
// Aprire una nuova finestra
const newWindow = window.open("https://www.example.com", "_blank", "width=600,height=400");

// Chiudere la finestra dopo 5 secondi
setTimeout(() => {
    if (newWindow) {
        newWindow.close();
        console.log("Finestra chiusa.");
    }
}, 5000);
```

2. Oggetto `navigator`

L'oggetto `navigator` contiene informazioni sul browser e sull'utente, come il tipo di browser, la lingua preferita, e se i cookie sono abilitati.

Esempio: Rilevamento delle informazioni del browser

```
console.log("Nome del browser: " + navigator.appName);
console.log("Versione del browser: " + navigator.appVersion);
console.log("Lingua preferita: " + navigator.language);
console.log("Cookie abilitati: " + navigator.cookieEnabled);
```

3. Oggetto `location`

L'oggetto `location` permette di accedere all'URL corrente e di modificarlo.

Esempio: Navigazione verso un'altra pagina

```
// Mostrare l'URL corrente
console.log("URL corrente: " + location.href);

// Reindirizzare a un'altra pagina
location.href = "https://www.google.com";
```

Esempio: Estrarre i componenti dell'URL

```
console.log("Protocollo: " + location.protocol); // es. 'https:'
console.log("Host: " + location.host); // es. 'www.example.com'
console.log("Pathname: " + location.pathname); // es. '/path/to/page'
console.log("Query string: " + location.search); // es. '?param=value'
```

4. Oggetto `history`

L'oggetto `history` consente di navigare avanti e indietro nella cronologia del browser.

Esempio: Navigazione nella cronologia

```
// Tornare indietro di una pagina
history.back();

// Andare avanti di una pagina
history.forward();

// Spostarsi di un certo numero di pagine
history.go(-2); // Torna indietro di due pagine
```

5. Oggetto `screen`

L'oggetto `screen` fornisce informazioni sullo schermo, come la risoluzione.

Esempio: Informazioni sullo schermo

```
console.log("Larghezza dello schermo: " + screen.width + "px");
console.log("Altezza dello schermo: " + screen.height + "px");
console.log("Larghezza disponibile: " + screen.availWidth + "px");
console.log("Altezza disponibile: " + screen.availHeight + "px");
```

6. Timer: `setTimeout` e `setInterval`

Questi metodi permettono di eseguire codice dopo un certo intervallo di tempo o a intervalli regolari.

Esempio: Utilizzo di `setTimeout`

```
setTimeout(() => {
```

```
    console.log("Questo messaggio appare dopo 3 secondi.");
}, 3000);
```

Esempio: Utilizzo di **setInterval**

```
let counter = 0;
const intervalId = setInterval(() => {
    counter++;
    console.log("Contatore: " + counter);
    if (counter === 5) {
        clearInterval(intervalId); // Ferma l'intervallo dopo 5 iterazioni
        console.log("Intervallo fermato.");
    }
}, 1000);
```

7. Interazione con i Cookie (parte del BOM)

I cookie possono essere gestiti direttamente tramite `document.cookie`.

Esempio: Creare, leggere e cancellare cookie

```
// Creare un cookie
document.cookie = "username=JohnDoe; expires=Fri, 31 Dec 2024 23:59:59 GMT; path=/";

// Leggere tutti i cookie
console.log("Tutti i cookie: " + document.cookie);

// Cancellare un cookie (impostare una data di scadenza passata)
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path=/";
console.log("Cookie eliminato: " + document.cookie);
```

Conclusione

Il BOM è una parte fondamentale di JavaScript per interagire con il browser e migliorare l'esperienza utente. Con il BOM, possiamo controllare finestre, gestire URL, cronologia, cookie e altro ancora. È importante capire la differenza tra BOM e DOM: mentre il DOM si occupa del contenuto del documento, il BOM si occupa dell'ambiente del browser.

1. Esempio di apertura e chiusura di una finestra con **window**

In questo esempio, una nuova finestra viene aperta quando l'utente clicca su un pulsante, e dopo 5 secondi, la finestra si chiude automaticamente.

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Finestra del Browser</title>
</head>
<body>
    <h1>Esperimento con il Modello di Oggetto del Browser (BOM)</h1>
    <button id="apriFinestra">Apri Finestra</button>
```

```
    <script>
        let nuovaFinestra;

        document.getElementById("apriFinestra").addEventListener("click", function() {
            // Apri una nuova finestra
            nuovaFinestra = window.open("https://www.example.com", "_blank",
"width=600,height=400");
            console.log("Finestra aperta");

            // Chiudi la finestra dopo 5 secondi
            setTimeout(function() {
                if (nuovaFinestra) {
                    nuovaFinestra.close();
                    console.log("Finestra chiusa.");
                }
            }, 5000);
        });
    </script>
</body>
</html>
```

Descrizione:

- Quando l'utente clicca sul pulsante "Apri Finestra", viene aperta una nuova finestra con un URL specificato.
- Dopo 5 secondi, la finestra si chiude automaticamente.

2. Esempio di utilizzo di `navigator` per visualizzare informazioni sul browser

In questo esempio, mostriamo informazioni sul browser e sul sistema operativo dell'utente.

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Informazioni del Browser</title>
</head>
<body>
    <h1>Informazioni sul Browser</h1>
    <p><strong>Nome del browser:</strong> <span id="browserName"></span></p>
    <p><strong>Versione del browser:</strong> <span id="browserVersion"></span></p>
    <p><strong>Lingua preferita:</strong> <span id="userLanguage"></span></p>

    <script>
        // Rilevare informazioni sul browser
        document.getElementById("browserName").textContent = navigator.appName;
        document.getElementById("browserVersion").textContent = navigator.appVersion;
        document.getElementById("userLanguage").textContent = navigator.language;
    </script>
</body>
</html>
```

Descrizione:

- Il codice mostra il nome del browser, la sua versione e la lingua preferita, prelevando queste informazioni dall'oggetto `navigator`.

3. Esempio di navigazione tramite `location`

In questo esempio, l'utente può cliccare un pulsante per navigare verso una nuova pagina, ed è anche possibile visualizzare i dettagli dell'URL corrente.

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Utilizzo di Location</title>
</head>
<body>
    <h1>Modifica URL con location</h1>
    <p><strong>URL corrente:</strong> <span id="currentUrl"></span></p>
    <button id="goToGoogle">Vai a Google</button>

    <script>
        // Mostra l'URL corrente
        document.getElementById("currentUrl").textContent = location.href;

        // Naviga verso Google quando si clicca il pulsante
        document.getElementById("goToGoogle").addEventListener("click", function() {
            location.href = "https://www.google.com";
        });
    </script>
</body>
</html>
```

Descrizione:

- La pagina visualizza l'URL corrente della pagina web.
- Quando l'utente clicca sul pulsante "Vai a Google", la pagina naviga verso il sito di Google.

4. Esempio di navigazione nella cronologia con `history`

In questo esempio, l'utente può navigare avanti o indietro nella cronologia del browser.

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Utilizzo di History</title>
</head>
<body>
    <h1>Navigazione nella cronologia</h1>
    <p><strong>Clicca i pulsanti per navigare avanti o indietro nella cronologia.</strong></p>
    <button id="goBack">Indietro</button>
    <button id="goForward">Avanti</button>
```

```
    <script>
        // Funzionalità per tornare indietro nella cronologia
        document.getElementById("goBack").addEventListener("click", function() {
            history.back();
        });

        // Funzionalità per andare avanti nella cronologia
        document.getElementById("goForward").addEventListener("click", function() {
            history.forward();
        });
    </script>
</body>
</html>
```

Descrizione:

- L'utente può cliccare su "Indietro" per tornare alla pagina precedente o su "Avanti" per andare alla pagina successiva nella cronologia del browser.

5. Esempio di utilizzo dei Timer con `setTimeout` e `setInterval`

In questo esempio, vediamo come utilizzare `setTimeout` per eseguire una funzione dopo un certo periodo di tempo, e `setInterval` per eseguire una funzione ripetutamente.

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Utilizzo dei Timer</title>
</head>
<body>
    <h1>Timer con setTimeout e setInterval</h1>
    <p id="timeoutMessage"></p>
    <p id="intervalCounter"></p>

    <script>
        // Utilizzare setTimeout per eseguire una funzione dopo 3 secondi
        setTimeout(function() {
            document.getElementById("timeoutMessage").textContent = "Questo messaggio appare
dopo 3 secondi!";
        }, 3000);

        // Utilizzare setInterval per contare ogni secondo
        let counter = 0;
        const intervalId = setInterval(function() {
            counter++;
            document.getElementById("intervalCounter").textContent = "Contatore: " + counter;
            if (counter === 5) {
                clearInterval(intervalId); // Fermare l'intervallo dopo 5 secondi
                document.getElementById("intervalCounter").textContent += " (Intervallo
fermato)";
            }
        }, 1000);
    </script>
```

```
</body>
</html>
```

Descrizione:

- Dopo 3 secondi dall'apertura della pagina, viene visualizzato un messaggio.
- Viene mostrato un contatore che incrementa ogni secondo fino a fermarsi dopo 5 secondi.

6. Esempio di gestione dei Cookie con `document.cookie`

In questo esempio, creiamo, leggiamo e cancelliamo un cookie.

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gestione dei Cookie</title>
</head>
<body>
    <h1>Gestione dei Cookie</h1>
    <button id="createCookie">Crea Cookie</button>
    <button id="readCookie">Leggi Cookie</button>
    <button id="deleteCookie">Elimina Cookie</button>
    <p id="cookieMessage"></p>

    <script>
        // Creare un cookie
        document.getElementById("createCookie").addEventListener("click", function() {
            document.cookie = "user=JohnDoe; expires=Fri, 31 Dec 2024 23:59:59 GMT; path=/";
            document.getElementById("cookieMessage").textContent = "Cookie creato!";
        });

        // Leggere i cookie
        document.getElementById("readCookie").addEventListener("click", function() {
            let cookie = document.cookie;
            document.getElementById("cookieMessage").textContent = "Cookie: " + cookie;
        });

        // Eliminare un cookie
        document.getElementById("deleteCookie").addEventListener("click", function() {
            document.cookie = "user=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path=/";
            document.getElementById("cookieMessage").textContent = "Cookie eliminato!";
        });
    </script>
</body>
</html>
```

Descrizione:

- L'utente può cliccare su "Crea Cookie" per creare un cookie con il nome "user" e valore "JohnDoe".
- Può cliccare su "Leggi Cookie" per visualizzare tutti i cookie salvati.
- Può cliccare su "Elimina Cookie" per cancellare il cookie creato.