Cookies are small pieces of data stored in the user's web browser by websites they visit. They are commonly used to remember information about the user, track their interactions with the website, and maintain stateful information across multiple requests and sessions. Cookies are an essential part of web development and play a crucial role in providing a personalised and seamless user experience.

1. Creating and setting a cookie in JavaScript:

To create a cookie, you use the `document.cookie` property. It allows you to set a string representing the cookie's key-value pair, along with optional attributes such as expiration date, path, domain, and security settings.

```
// Function to set a cookie
function setCookie(name, value, daysToExpire) {
  const expirationDate = new Date();
  expirationDate.setTime(expirationDate.getTime() + (daysToExpire * 24 * 60 * 60 *
1000));
  const expires = `expires=${expirationDate.toUTCString()}`;
  document.cookie = `${name}=${value};${expires};path=/`;
}

// Usage example
setCookie('username', 'JohnDoe', 7);
```

2. Retrieving a cookie in JavaScript:

To retrieve a cookie's value, you need to parse the `document.cookie` string.

```
// Function to get the value of a cookie by its name
function getCookie(name) {
  const cookies = document.cookie.split(';');
  for (const cookie of cookies) {
    const [cookieName, cookieValue] = cookie.trim().split('=');
    if (cookieName === name) {
      return decodeURIComponent(cookieValue);
    }
  }
  return null;
}
```

```
// Usage example
const username = getCookie('username');
console.log('Username:', username);
```

3. Deleting a cookie in JavaScript:

To delete a cookie, you need to set its expiration date to a past date, effectively making it expire immediately.

```
// Function to delete a cookie by its name
function deleteCookie(name) {
  document.cookie = `${name}=;expires=Thu, 01 Jan 1970 00:00:00 UTC;path=/`;
}

// Usage example
deleteCookie('username');
```

4. Cookie Attributes:

- `**name=value**`: The key-value pair representing the data you want to store.
- `**expires**`: Sets an expiration date for the cookie. After this date, the cookie will be deleted.
- `**path**`: Specifies the path on the server to which the cookie will be sent.
- `**domain**`: Specifies the domain for which the cookie is valid.
- `**secure**`: If set to true, the cookie will only be sent over secure (HTTPS) connections.
- `**HttpOnly**`: If set to true, the cookie cannot be accessed by client-side scripts, enhancing security.

5. Use Cases of Cookies:

a. **User Authentication:** Cookies are commonly used to store a session identifier after a user logs in, allowing the server to recognize the user across multiple requests.
b. **Remember Me Functionality:** Websites use cookies to remember users who selected "Remember Me" during login, avoiding the need to log in on every visit.
c. **Shopping Carts:** E-commerce websites use cookies to store information about the items in the user's shopping cart.
d. **Personalization:** Cookies are used to remember user preferences, such as language settings or layout preferences.

e. **Tracking and Analytics:** Cookies help track user behaviour, interactions, and navigation patterns for analytics and marketing purposes.

6. Limitations and Concerns:

a. **Security:** Cookies can be vulnerable to attacks like cross-site scripting (XSS) and cross-site request forgery (CSRF). Secure and HttpOnly attributes should be used when appropriate.

b. **Privacy:** Cookies can track user activities, leading to privacy concerns. Web developers should be transparent about their use and provide users with options to manage cookie preferences.

c. **Size Limit:** Cookies are limited in size (usually around 4KB), which may restrict the amount of data that can be stored.

d. **Browser Support:** While cookies are widely supported, users can disable or delete them, affecting the functionality of certain websites.

Prepared By :
Divyansh Singh (LinkedIn : rgndunes)