COOKIE

In JavaScript, a "cookie" is a small piece of data that a website can store on a user's device. Cookies are often used to remember information about the user with the website. They are commonly utilized for various purposes such as session management, tracking user behavior, and personalization.

Cookie	
Capacity	4kb
Browsers	HTML/HTML5
Accessible from	Any window/tab
Expires	Manual
Storage Location	Brower and Server
Send with request	Yes

Usually cookies are created by the server program and sent to the user's browser (Client program) when the user visits a website for the first time. The browser then stores this cookie locally on the user's device. Whenever the user visits the same website again, the browser sends the stored cookie data back to the server along with the HTTP requests, allowing the server to recognize the user and retrieve stored information associated with that user.

Cookies typically consist of **key-value pairs** and may have additional attributes like expiration date, domain, and path.

1. **Setup cookie**

To set a cookie using JavaScript, you can use the `document.cookie` property. Here's an example of how to set a cookie with a name-value pair:

e.g.-1

```
// Set a cookie named "username" with the value "JohnDoe"
document.cookie = "username=JohnDoe; path=/";
```

e.g.-2

```
<!DOCTYPE html>
<html>
<head>
  <title>Storage Menu</title>
</head>
<body>
  <script>
  const expirationDate = new Date("2024-12-31");
  const formattedExpiration = expirationDate.toUTCString();
  console.log(formattedExpiration)
  document.cookie =`username = bhoomika; path= / ; expires = ${formattedExpiration}`
  </script>

</body>
</html>
```

2. **Read cookie**

    To read a cookie in JavaScript, you can access the document.cookie property, which returns a string containing all the cookies associated with the current domain. However, extracting specific values from this string requires some parsing.

e.g.

```
function getCookie(name) {
  const cookies = document.cookie.split('; ');
  for (const cookie of cookies) {
    const [cookieName, cookieValue] = cookie.split('=');
    if (cookieName === name) {
      return cookieValue;
    }
  }
  return null; // Return null if the cookie with the specified name is not found
}

// Usage example:
const username = getCookie("username");
if (username) {
  console.log(`Username cookie value: ${username}`);
} else {
  console.log("Username cookie not found.");
}
```

3. **Modify cookie**

    To modify a cookie in JavaScript, you essentially overwrite the existing cookie with a new value, potentially with updated attributes like expiration time or other properties

```
function modifyCookie(name, newValue, expirationDays) {
  const date = new Date();
  date.setTime(date.getTime() + (expirationDays * 24 * 60 * 60 * 1000));
  const expires = "expires=" + date.toUTCString();

  document.cookie = `${name}=${newValue};${expires};path=/`;
}

// Usage example:
modifyCookie("username", "NewUsername", 7);
```

4. **Delete cookie**

    To delete a cookie in JavaScript, you can set its expiration time to a date in the past. This essentially instructs the browser to remove the cookie immediately. Here's how you can delete a cookie using the `document.cookie` property:

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Storage Menu</title>
</head>
<body>
    <script>
    const expirationDate = new Date("2023-12-31");
    const formattedExpiration = expirationDate.toUTCString();
    console.log(formattedExpiration)
    document.cookie =`username = bhoomika; path= / ; expires = ${formattedExpiration}`
    </script>

</body>
</html>
```

Note:

1. *Cookies are part of the browser's local storage, and changes made to cookies using JavaScript are limited to the browser session and the user's device. Deleting* ***a cookie doesn't guarantee the removal of all traces of that information from external servers or tracking mechanisms.***

2. Cookies are widely used for various web-related tasks, there are also privacy concerns associated with them. Modern web development often employs more advanced techniques like using the **Web Storage API *(localStorage and sessionStorage*)** or browser technologies like HTTP-only cookies and SameSite attribute to provide better control over data storage and security.

Akhilesh Kumar Gupta
(Technical Training specialist)
TechCarrel LLP