COOKIE

Cookie 是儲存在瀏覽器的一小段文字資料，通常由伺服器透過 `Set-Cookie` header 傳遞給瀏覽器。瀏覽器收到後會將 cookie 儲存起來，並在之後的請求回傳 cookie 至同樣的伺服器。而它常見的用途像是認證身份，例如登入狀態、購物車等，也被應用於追蹤使用者及廣告上。Cookie 也被用於客戶端的儲存方式，但由於 cookie 會被附加在每一次的 request 之中，可能會影響效能，所以如果是不需要記錄在 server 的資訊，可以改用 storage API。

而要如何使用 Cookie?它的語法如下:

```
Set-Cookie: [cookie 名稱]=[cookie 值]
```

瀏覽器看到 `Set-Cookie` header 便會將 cookie 儲存起來，之後對同一個 domain 發送 HTTP request 的時候，瀏覽器就會將 cookie 帶在 HTTP request 的 `Cookie` header 裡。Request 中的 cookie header 會是 `[cookie 名稱]=[cookie 值]` 的形式，用分號串接之後的結果：

```
Cookie: [cookie1]=[value1]; [cookie2]=[value2]
```

若我們是要使用 JavaScript 讀取 Cookie，便需要輸入以下格式:

```
console.log(document.cookie);
```

讀取出來的 `document.cookie` 會得到一個字串，這個字串是將這個網域底下所有 cookie 用分號串接以後的結果，其中每個 cookie 都是 `[cookie 名稱]=[cookie 值]` 的形式，例如：`name=John; gender=male` 表示這個網域底

下有兩個 cookie：`name` 和 `gender`，其中 `name` 的值是 `John`，而 `gender` 的值是 `male`。

若要寫入 Cookie，格式為: `document.cookie = 'key=value;'`。注意雖然我們用 `document.cookie = ...`，但是並不會整個 cookie 都被覆蓋掉，只有我們指定的 key 會被更新。如下面的例子，`cookie3` 會被新增的同時，原本的 cookie1 和 cookie2 都還會被保留。如下圖:

```
console.log(document.cookie); // cookie1=value1;
cookie2=value2;
document.cookie = 'cookie3=value3';
console.log(document.cookie); // cookie1=value1;
cookie2=value2; cookie3=value3;
```

參數:

Cookie 除了名稱和值之外，通常還需要設定其他額外參數，下面會一一介紹。新增參數的方式是用分號區隔各個參數，例如：

```
user=John; path=/; expires=Tue, 19 Jan 2038 03:14:07 GMT
```

簡單地說，我們會用 `Domain` 和 `Path` 指定 cookie 的可用範圍，用 `Expires` 和 `Max-Age` 控制 cookie 的有效期限，而 `HttpOnly`、`Secure`、和 `SameSite` 則是和安全性相關的參數。

Domain:

```
domain=example.com
```

`domain` 用來指定哪些網域可以存取這個 cookie。

Path:

```
path=/admin
```

`path` 參數用來指定哪些路徑可以存取這個 cookie。

Expires, Max-age:

`expires`, `max-age` 參數的作用是設定 cookie 的有效期限。如果沒有額外設定 `expires` 或是 `max-age` 參數，當瀏覽器關閉之後，儲存在瀏覽器的 cookie 便會消失，這就是所謂的 session cookie。如果我們希望瀏覽器關掉之後 cookie 還是會被儲存下來，那就必須設定 `expires` 或是 `max-age`。

Secure:

`Secure` 參數的作用是讓 cookie 只能透過 https 傳遞。

Cookie 預設是不區分 http 或是 https 的。換句話說，當我們設定 http://example.com 的 cookie 時，https://example.com 也能看得到同樣的 cookie。

如果 cookie 設了 `secure` 參數，只有透過 https 存取這個網站才能存取這個 cookie；透過 http 存取這個網站會看不到這個 cookie。

SameSite:

`Samesite` 的作用是防止 cookie 以跨站方式傳送，可以幫助避免 CSRF (Cross-Site Request Forgery，跨站請求偽造) 攻擊。要理解 SameSite 如何幫助防止 CSRF 攻擊之前，我們需要先理解 CSRF 攻擊。

CSRF 攻擊是什麼呢？簡單地說，他會在受害者已登入的狀態下，假借受害者的身份進行惡意操作，例如把受害者銀行裡的錢轉到攻擊者自己的帳戶中。

參考網站: https://shubo.io/cookies/#samesite