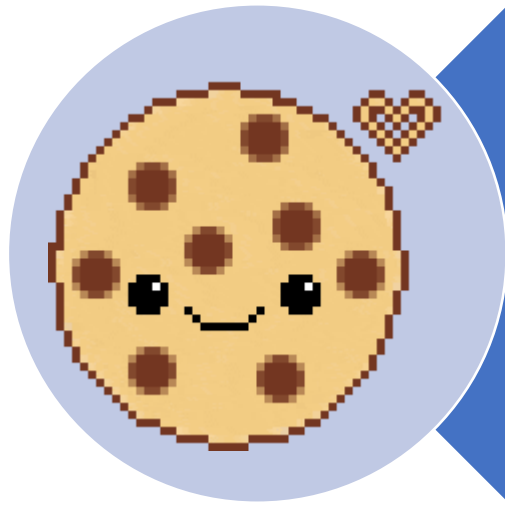


Desarrollo Web en Entorno Cliente

Tema 12





Cookies



[cookies]

Las *cookies* (o galletas en español), más correctamente *HTTP cookies*, es una tecnología que se inventó para el navegador Netscape en 1994.

Actualmente, está definida en el estándar RFC 6265 y es, básicamente, una tecnología consistente en almacenar y recuperar información a través de las cabeceras HTTP. Las *cookies* se almacenan localmente, en el ordenador del cliente, durante un tiempo determinado. Esta información, que queda almacenada en el dispositivo del usuario, se envía hacia y desde el servidor web en las cabeceras HTTP.

El nombre de *cookies* proviene de las galletas de la suerte chinas que albergan en su interior un mensaje al igual que las *HTTP cookies*, que guardan una información.



[cookies]

Existen varios tipos de *cookies*:

Zombie cookies tienen la capacidad de regenerarse una vez han sido borradas del sistema ya que se alojan en el sistema y no directamente en el navegador.

Secure cookies usan métodos de cifrado para almacenar la información, con el fin de impedir que los datos que almacenan puedan ser objetos de ataques. Este tipo de cookies solo se usan en conexiones HTTPS.

Session cookies son las cookies que se generan cuando es necesario seguir la sesión en un dominio o ruta web, como la validación de un usuario, etc. Son borradas al cerrar el navegador o cerrar la sesión.

Persistent cookies son las más frecuentes y almacenan información relativa a cuestiones relativas a un usuario y un sitio web. Caducan cuando ha pasado un tiempo determinado fijado al crear la *cookie*.

[cookies]

Cuando un usuario solicita una página web o cualquier otro recurso, el servidor envía el documento y cierra la conexión. Si el mismo usuario vuelve a solicitar la misma u otra página al servidor, será tratado como si fuera la primera solicitud que realiza.

Esta situación puede suponer un problema, en muchas situaciones y las cookies son una técnica que permite solucionarlo de las otras muchas que existen.

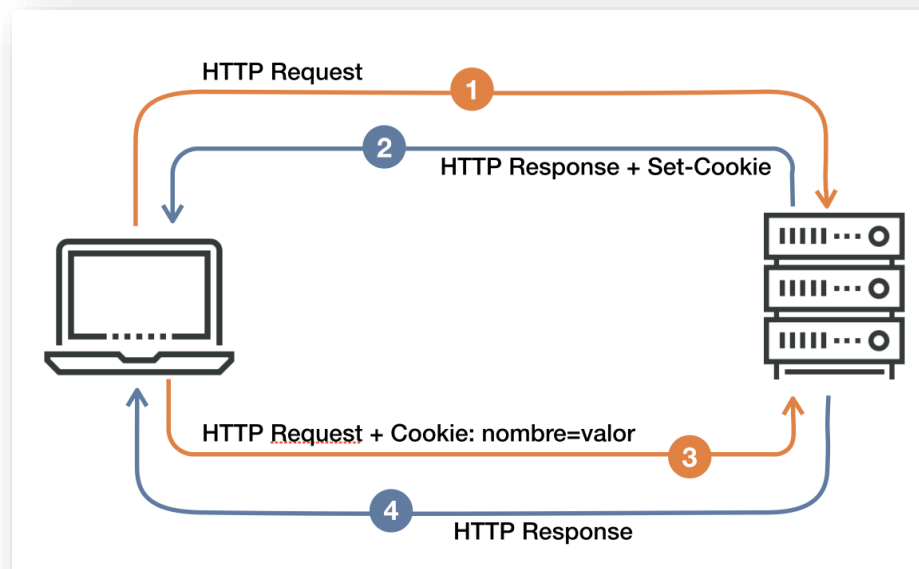
Mediante la técnica de las cookies, el servidor puede enviar información al usuario en las cabeceras HTTP de respuesta y esta información queda almacenada en el dispositivo del usuario.

En la siguiente solicitud que realice el usuario, la cookie es enviada de vuelta al servidor en las cabeceras HTTP de solicitud. El servidor puede leer esta información y así “recordar” al usuario y la información asociada con él.

[cookies]

En realidad, hay dos mecanismos diferentes para crear *cookies*, desde el servidor o desde el navegador del usuario usando JavaScript.

Desde el servidor se realiza cuando se accede a una url desde el navegador, el navegador realiza una petición HTTP al servidor donde están alojados los contenidos. El mensaje de contestación a esa petición HTTP tiene varias partes, una de ellas es la cabecera (*header*). En esta cabecera el servidor utiliza la instrucción *Set-Cookie* para enviar la *cookie*.



[cookies]

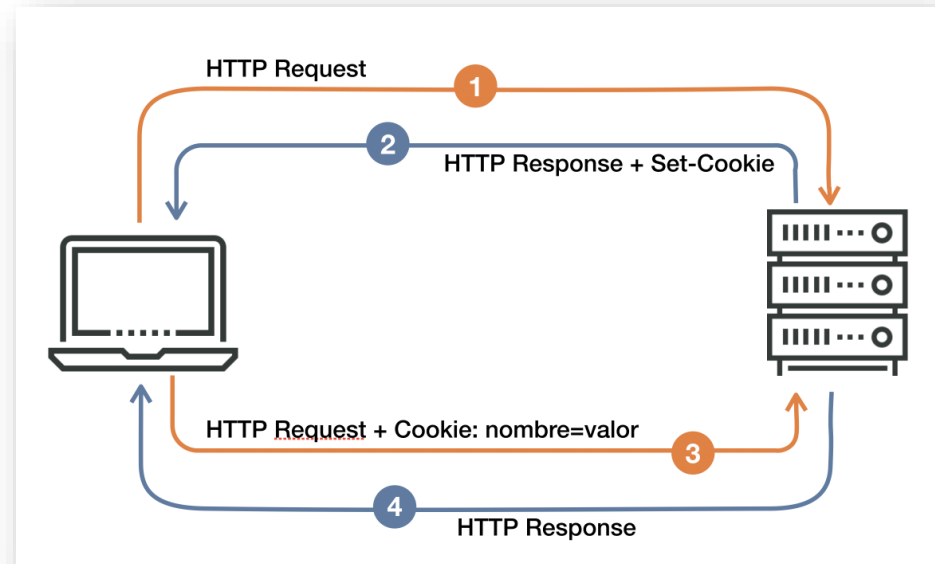
La otra forma de crear una cookie es a través de JS mediante el uso de una propiedad llamada *cookie* del objeto *document*. Esta se almacena desde código JS que se ejecuta directamente en el navegador del cliente.

Una vez que la *cookie* se ha almacenado en el ordenador que ejecuta el navegador la próxima vez que el navegador haga una petición a uno de los servidores del mismo dominio, modificará la cabecera de la petición (*HTTP Request*) con los siguientes datos:

Content-type: text/html

Cookie: nombre=valor

De esta manera el servidor será consciente de la presencia de una cookie llamada *nombre* con valor *valor*.



[cookies]

Una *cookie* no es más que un par clave-valor, al que le acompañan opcionalmente una serie de atributos, también en la forma, clave-valor que sirven para definir cómo y cuándo se usará esa cookie.

Así, la *cookie* consiste en una cadena de texto (*string*), con un contenido de varios pares clave-valor, *key=value* , cada uno separado por punto y coma ;.

```
<nombre>=<valor>; expires=<fecha>; max-age=<segundos>; path=<ruta>; domain=<dominio>;  
secure; httponly;
```

Desde el servidor, las cookies son creadas mediante la cabecera de respuesta *HTTP Set-Cookie*:

```
Set-Cookie: <nombre>=<valor>; expires=<fecha>; max-age=<segundos>; path=<ruta>;  
domain=<dominio>; secure; httponly;
```


[cookies]

Una respuesta HTTP puede contener múltiples cabeceras *Set-Cookie*, una por cada cookie.

El siguiente ejemplo representa una cabecera de respuesta con creación de dos cookies:

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: colorPreference=blue
Set-Cookie: sessionToken=48745487; Expires=Thu, 01 Jan 2031 19:22:10 GMT
```

Una vez que la cookie ha sido creada, cada nueva solicitud que realice el usuario enviará la cookie en la cabecera de solicitud *HTTP Cookie*. En esta cabecera sólo se envían las cookies que sean válidas según los parámetros establecidos al crearla (*expires*, *path*, etc.) y sólo se envía el par *<nombre>=<valor>*

```
Cookie: <nombre>=<valor>
```

[cookies]

Cuando se envían varias cookies, se envían todas separadas por punto y coma ; en una única cabecera Cookie.

El siguiente ejemplo representa una cabecera HTTP de solicitud que envía las dos cookies mostradas en el ejemplo anterior:

```
GET /ejemplo.html HTTP/1.1  
Host: www.iesvenancioblanco.es  
Cookie: colorPreference=blue; sessionId=48745487
```

En total, cada cookie puede tener un tamaño máximo de 4.096 bytes, se permiten hasta 50 cookies por dominio y 3000 cookies por navegador/dispositivo.

[cookies]

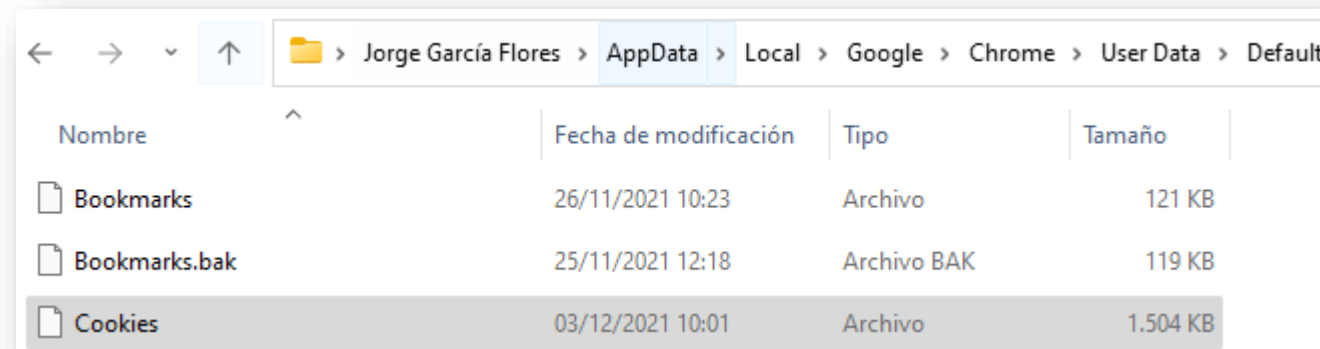
Cada navegador y sistema operativo almacena las cookies en un sitio diferente del ordenador cliente.

Google Chrome, bajo el entorno Windows, almacena todas las cookies en un archivo llamado *cookies* ubicado en el siguiente directorio:

%LOCALAPPDATA%\Google\Chrome\User Data\Default

o también:

C:\Users\<nombre_usuario>\AppData\Local\Google\Chrome\User Data\Default



Jorge García Flores > AppData > Local > Google > Chrome > User Data > Default				
Nombre	Fecha de modificación	Tipo	Tamaño	
Bookmarks	26/11/2021 10:23	Archivo	121 KB	
Bookmarks.bak	25/11/2021 12:18	Archivo BAK	119 KB	
Cookies	03/12/2021 10:01	Archivo	1.504 KB	

[cookies]

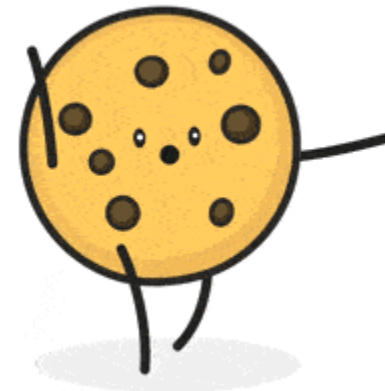
El navegador Firefox lo hace en:

`%AppData%\Roaming\Mozilla\Firefox\Profiles`

o también:

`C:\Users\<nombre_usuario>\AppData\Roaming\Mozilla\Firefox\Profiles`

Y dentro del perfil del usuario en un archive llamado archivo `cookies.sqlite`.



[cookies]

A continuación, se muestran cada uno de los parámetros de una cookie:

`<nombre>=<valor>`

`<nombre>` es el nombre (*key*) que identifica a la cookie y `<valor>` es su valor. El `<nombre>` es obligatorio para poder crear una cookie mientras que el `<valor>` es opcional. Las cookies sin valor o con valor vacío son tratadas de forma diferente en PHP y JavaScript.

`<nombre>` es definido como un *token* y por ello solo puede contener caracteres alfanuméricos más `!#$%&'*+-.^_`/~`. No puede contener ningún espacio, coma o punto y coma. Tampoco caracteres de control (`\x00`, `\x1F` más `\x7F`) y no debería utilizarse el carácter igual `=` que es utilizado como separador.

`<valor>` puede contener cualquier valor alfanumérico excepto espacios, comas, punto y coma, caracteres de control, barra invertida y comillas dobles. Si es necesario el uso de estos caracteres, el valor de la cookie deberá ser codificado (reemplazar esos caracteres por su código ASCII); en JS se puede hacer con `encodeURIComponent()`, en PHP con `urlencode()`, pero al leer el valor habrá que decodificarlo con `decodeURIComponent()` o `urldecode()`.



expires=<fecha> y max-age=<segundos>

Ambos parámetros especifican el tiempo de validez de la cookie.

Estos parámetros son opcionales y si no se especifica ninguno de los dos, la cookie sólo es válida para la sesión actual, lo que se conoce como *session cookie*, hasta que el usuario cierre el navegador.

expires establece una fecha, que ha de estar en formato *GMT/UTC*, por ejemplo: *Mon, 03 Jul 2021 21:44:38 UTC*.

max-age establece una duración máxima en segundos. Si se especifican ambos, *max-age* tiene preferencia sobre *expires*. *max-age* no es soportado por IE 8 o sus versiones anteriores.

Si *max-age* es cero la cookie se elimina, al igual que si *expires* si contiene una fecha ya pasada.

[cookies]

path=<ruta>

Este parámetro es opcional y establece la ruta para (*path*) la cuál la cookie es válida.

Por defecto, si no se especifica ningún valor, una *cookie* sólo es válida para la ruta (*path*) actual, esto es, el directorio que contiene la página actual.

Por ejemplo, si la cookie se establece en *https://www.iesvenancioblanco.es/noticias/pagina.html*, la cookie también será válida para cualquier otra página del directorio *https://www.iesvenancioblanco.es/noticias/*.

Utilizando el parámetro *path* se puede establecer un directorio diferente.

Por ejemplo, */* sería para todos los directorios del dominio, incluyendo el directorio raíz; */blog* sería sólo para páginas bajo el directorio *https://www.iesvenancioblanco.es/blog*.

[cookies]

domain=<dominio>

Este parámetro es opcional.

Por defecto, las *cookies* son válidas sólo para el subdominio actual donde se crea la cookie (Para la validez de las *cookies*, el subdominio *www.iesvenancioblanco.es* no se considera parte del dominio *www*).

Así, si el atributo *domain* está vacío y se está en *iesvenancioblanco.es*, la *cookie* es válida sólo para *iesvenancioblanco.es*, pero si se está en *www.iesvenancioblanco.es*, la *cookie* será válida sólo para *www.iesvenancioblanco.es*, si se está en *sub.iesvenancioblanco.es*, la *cookie* es válida sólo para *sub.iesvenancioblanco.es*.

Si no se utiliza este parámetro la cookie solo es válida para el subdominio actual.

[cookies]

Si se utiliza el parámetro *domain*, entonces, se especifica el dominio y todos sus subdominios en los que la *cookie* es válida.

Por ejemplo, *domain=sub.iesvenancioblanc.es* crea una *cookie* válida sólo para *sub.iesvenancioblanc.es* y sus subdominios como *alumnos.iesvenancioblanc.es* pero si se está en otro subdominio, por ejemplo, en *www.educajcy.es*, y se intenta leer esa *cookie*, no se podrá, ya que sólo es válida para *sub.iesvenancioblanc.es*.

Si se está en *iesvenancioblanc.es* y se quiere que una *cookie* sea válida sólo para *iesvenancioblanc.es*, no se consigue estableciendo *domain=.iesvenancioblanc.es* (nótese el punto después del igual) ya que el primer punto será ignorado, según establece la especificación RFC 6265, por lo que *domain=.iesvenancioblanc.es* y *domain=iesvenancioblanc.es* tienen el mismo efecto y crean una *cookie* válida para cualquier subdominio.

De esta manera, si se está en *iesvenancioblanc.es* y se quiere que una *cookie* sea válida sólo para *iesvenancioblanc.es*, la única opción es dejar el atributo *domain* vacío ya que si se establece *domain=iesvenancioblanc.es* la *cookie* será válida para todos los subdominios, no sólo para *iesvenancioblanc.es*.

[cookies]

En la especificación RFC 2965 se indica, sobre la cabecera *Set-Cookie2*, que debe llevar un punto precedente para crear una *cookie* válida para todos los subdominios.

Además, en la especificación RFC 2109, ya obsoleta, *domain=iesvenancioblanc.es* y *domain=.iesvenancioblanc.es* no eran tratados como lo mismo.

Por lo que es recomendable utilizar siempre el punto precedente si se quiere crear una *cookie* válida para todos los subdominios ya que garantiza el mismo comportamiento en implementaciones antiguas, que puedan seguir especificaciones obsoletas, como en implementaciones modernas.

Por último, por motivos de seguridad, no se permite crear *cookies* para dominios diferentes al que crea la *cookie* (*same-origin policy*).

[cookies]

secure

Opcional. Este parámetro no tiene ningún valor. Si está presente la cookie sólo es válida para conexiones encriptadas (por ejemplo, mediante protocolo HTTPS).

HttpOnly

Opcional. Este parámetro no tiene ningún valor. Si está presente, la cookie solo es accesible mediante protocolo HTTP (o HTTPS). Estas *cookies* no pueden ser leídas ni creadas mediante otros protocolos o APIs, como, por ejemplo, JS.

[cookies]

Es importante entender que las *cookies* son enviadas y pueden ser recibidas con cualquier solicitud, no necesariamente una página web.

Por ejemplo, al solicitar una imagen o un script pueden enviarse *cookies*.

Esto hace que se pueda estar en una determina página que carga un recurso desde otro dominio y que se haya recibido *cookies* de ambos dominios.

La *cookie* de otro dominio se conoce como *third-party cookie* y, como se ha creado al solicitar un recurso a ese dominio, no significa que se haya saltado la política de seguridad de *same-origin*.

No obstante, para evitar que las *cookies* sean leídas desde dominios externos, llamados *cross-site*, se creó el parámetro *samesite*.

[cookies]

samesite

Opcional, añadida con Chrome 51. Este parámetro impide al navegador enviar *cookies* a través de peticiones *cross-site*. Los valores posibles son *Lax*, *strict* o *none*.

strict impide que la *cookie* sea enviada por el navegador al sitio destino en contexto de navegador *cross-site*, incluso cuando se sigue un enlace.

Lax solo las *cookies* del mismo sitio se envían en solicitudes secundarias, esto es, aquellas que no modifican la URL en la barra del navegador como llamadas para cargar imágenes o iframes, pero si se envían cuando un usuario navegue a la URL desde un sitio externo, por ejemplo, siguiendo un enlace.

none establece de manera explícita que no se aplicarán restricciones. La cookie se enviará en todas las solicitudes, tanto en *cross-site* como en *samesite*.

[document.cookie]

La propiedad *cookie* de *document* permite utilizar *cookies client-side* desde JS.

Esta propiedad *cookie* permite crear, leer, modificar y eliminar *cookies*.

Para crear una *cookie* se crea una cadena de texto, que define la *cookie*, y se asigna a la propiedad *document.cookie*.

Ejemplo:

```
document.cookie = "nombrecookie=valorcookie; max-age=3600; path="/;
```

[document.cookie]

Si se quieren crear varias cookies se tienen que asignar cada una de ellas a la propiedad *document.cookie*.

Cada vez que se le asigna una nueva *cookie*, no se sobrescriben las *cookies* anteriores, sino que la nueva se añade a la colección de *cookies* del documento, siempre que la *cookie* tenga un nombre único.

Este comportamiento se debe a que *document.cookie* no es un dato con un valor (*data property*), sino una propiedad de acceso, con métodos *set* y *get* nativos (*accessor property*), esto lo veremos en breve en próximos temas.

El siguiente ejemplo crea dos cookies *nombre* y *puntuacion*:

```
document.cookie = "nombre=jorge; max-age=3600; path="/";  
document.cookie = "puntuacion=1200";
```

[document.cookie]

Como las *cookies* se envían en cabeceras HTTP deben estar correctamente codificadas.

Es conveniente utilizar la función *encodeURIComponent()* para evitar errores inesperados.

El siguiente ejemplo funciona por el uso de la función *encodeURIComponent()*, si no se hubiera utilizado el espacio incluido en el valor daría error.

```
const testvalue = "Hola mundo!";  
document.cookie = "testcookie=" + encodeURIComponent(testvalue);
```


[document.cookie]

Si no se especifica fecha de caducidad la cookie será válida sólo para la sesión actual.

Para utilizar el parámetro *expires*, la fecha tiene que estar en formato UTC. Para ello, puede utilizarse el método *Date.toUTCString()*, que devuelve una cadena de texto en formato UTC (*Www, dd Mmm yyyy hh:mm:ss GMT*) a partir de una fecha y hora, en formato americano (años, días, mes), pasada como argumento.

Ejemplo de cookie con caducidad para el 1 de febrero del año 2068 a las 11:20:

```
const expiresdate = new Date(2068, 1, 02, 11, 20);  
const cookievalue = "Hola Mundo!";  
document.cookie = "testcookie=" + encodeURIComponent(cookievalue) +  
    "; expires=" + expiresdate.toUTCString();
```

[document.cookie]

La cookie debe tener un identificador único dentro del dominio o/y dentro de un *path* determinado.

Si se añade a la colección de cookies del documento otra, pero cuyo nombre ya existe, en este caso se modifica la cookie existente en lugar de añadir una cookie más.

En el siguiente ejemplo, la primera vez que se asigna un valor a la propiedad *document.cookie* se crea la cookie *nombre*, pero en la segunda línea, como ya existe una cookie con el nombre *nombre* se modifica dicha cookie:

```
document.cookie = "nombre=Miguel";  
document.cookie = "nombre=Jorge";
```

[document.cookie]

Si una cookie se crea para un dominio o/y para un *path* determinado y se quiere modificar, el dominio y el *path* han de coincidir con la cookie a modificar, de lo contrario, se crearán dos cookies diferentes válidas para cada *path* y dominio.

Ejemplo, se está en *iesvenancioblanco.es/blog*, el valor predeterminado del *path* es, en este caso, */blog*:

```
// Se crea la cookie para el path "/"  
document.cookie="nombre=Jorge; path=/";
```

```
// Se crea una nueva cookie para el path "/blog"  
document.cookie="nombre=Pedro";
```

```
// Se modifica la cookie "nombre" del path "/"  
document.cookie="nombre=Juan; path=/";
```

[document.cookie]

Para eliminar una cookie desde JS se debe modificar dicha cookie, asignándole una fecha de caducidad (*expires*) ya pasada o un *max-age* igual a cero.

En ambos casos da igual el valor que se le asigne a la cookie porque el navegador eliminará la cookie.

Ejemplo:

```
document.cookie = "nombre=Jorge";
```

```
document.cookie = "nombre=; expires=Thu, 01 Jan 1970 00:00:00 UTC";
```

```
// Otra forma con max-age
```

```
document.cookie = "nombre=; max-age=0";
```

[document.cookie]

No hay un método de lectura directo para cada cookie individual en JS.

A través de la propiedad *document.cookie* se puede obtener una cadena de texto con todas las cookies válidas para el documento y, manipulando dicha cadena se puede encontrar el nombre y valor de la cookie que se desea leer.

La cadena tiene la forma o estructura siguiente:

"cookie1=valor1[;cookie2=valor2;cookie3=valor3;...]"

Ejemplo:

```
const cadenaCookies = document.cookie;
```

[document.cookie]

La cadena no proporciona otros parámetros como *expires...*, sólo contiene pares de nombre de la cookie y su valor.

Además, solo se pueden leer las cookies válidas para el documento actual.

Las cookies para otros *path*, dominios o cookies caducadas no se pueden leer.

Aunque en una página pueden crearse cookies para otros subdominios y *paths*, sólo se pueden leer las que son válidas para el subdominio y *path* actual.

[document.cookie]

Ejemplo:

```
document.cookie="nombre=Jorge; SameSite=None; Secure";  
document.cookie="color_fondo=azul; SameSite=None; Secure";  
  
function mostrarCookies() {  
    const cookies = document.cookie;  
    return cookies;  
}
```

Bibliografía y recursos online

- https://www.w3schools.com/js/js_cookies.asp
- <https://developer.mozilla.org/en-US/docs/Web/API/document/cookie>
- <https://cybmeta.com/que-son-las-cookies-y-como-funcionan>
- <https://cybmeta.com/cookies-en-javascript>
- <https://ayudaleyprotecciondatos.es/cookies/javascript/>

Bibliografía y recursos online

- <http://www.ietf.org/rfc/rfc6265.txt>
- <https://www.doctormetrics.com/samesite-cookies-ataques-csrf/>
- <https://errequeerre.es/que-son-realmente-las-cookies-y-su-papel-en-la-medicion-cross-site/>
- <https://raidboxes.io/es/blog/security/cross-site-request-forgery-cookies-als-gefahr/>
- <https://javascript.info/localstorage#sessionstorage>