COOKIES

¿Qué son las cookies?

Las cookies, de nombre más exacto **HTTP cookies**, es una tecnología que en su día inventó el navegador Netscape, actualmente definida en el estándar RFC 6265, y que consiste básicamente en información enviada o recibida en las cabeceras HTTP y que **queda almacenada localmente client-side** durante un tiempo determinado. En otras palabras, es información que queda almacenada en el dispositivo del usuario y que se envía hacia y desde el servidor web en las cabeceras HTTP.

Cuándo un usuario solicita una página web (o cualquier otro recurso), el servidor envía el documento, **cierra la conexión y se olvida del usuario**. 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 muchas técnicas que hay).

Con 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. En el servidor podemos leer esta información y **así «recordar» al usuario e información asociada a él**.

¿Cómo funcionan?

Una cookie consiste en una cadena de texto (string) con varios pares `key=value` cada uno separado por `;`:

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

Un respuesta HTTP puede contener múltiples cabeceras `Set-Cookie`, una por cada cookie. Por ejemplo, una cabecera de respuesta con creación de dos cookies podría ser:

```
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>
```

Cuándo se envían varias cookies, se envían todas separadas por `;` en una única cabecera `Cookie`. Por ejemplo, una cabecera HTTP de solicitud que envía las dos cookies anteriores podría ser:

```
GET /ejemplo.html HTTP/1.1

Host: www.mysite.com

Cookie: colorPreference=blue; sessionToken=48745487
```

Limitaciones

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.

Parámetros

Veamos cada uno de los parámetros de una cookie en más detalle.

<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 (más adelante veremos como las cookies sin valor o con valor vacío son tratadas de forma diferente en PHP y JavaScript).

Como `<nombre>` y `<valor>` se puede utilizar cualquier valor arbitrario que deseemos, sólo hay que tener en cuenta que:

– `<nombre>`: es definido como un **token** y como tal solo puede contenter caracteres alfanuméricos más `!#$%&'*+-.^_`|~`. No puede contenter ningún espacio, coma o punto y coma. Tampoco están permitidos los caracteres de control (`\x00`, `\x1F` más `\x7F`) y no se debería utilziar el signo `=` que es utilizado como separador.

– `<valor>` puede contenter cualquier valor alfanumérico excepto espacios, comas, punto y coma, caracteres de control, barra invertida y comillas dobles. En caso de que sea necesario el uso de estos caracteres, el valor de la cookie **deberá ser codificado** (reemplazar esos caracteres por su código ASCII); en JavaScript lo podemos hacer con `encodeURIComponent()`; en PHP se podría hacer con `urlenconde()`; al leer el valor habría que descodificarlo con `decodeURIComponent()` o `urldecode()`.

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

Opcional. Ambos parámetros especifican el tiempo de validez de la cookie. `expires` establece una fecha, que ha de estar en formato GMT/UTC (por ejemplo, Mon, 03 Jul 2006 21:44:38 UTC). `max-age` establece una duración máxima en segundos (si segundos). Si se especifican ambos, `max-age` toma preferencia (`max-age` no es soportado por IE 8 e inferior).

Si no se específica ninguno de los dos, la cookie sólo es válida para la sesión actual, lo que se conoce como **session cookie** (por ejemplo, hasta

que el usuario cierre el navegador). Si `max-age` es cero la cookie se elimina, al igual que si `expires` es una fecha pasada.

path=<ruta>

Opcional. Establece la ruta para la cuál la cookie es válida. Algo así como los «directorios» o «secciones» de la web. Por defecto, si no se especifica ningún valor, una cookie sólo es **válida para el path actual** (el directorio que contiene la página actual). Por ejemplo, si la cookie se establece en «https://ejemplo.com/noticias/pagina.html», la cookie será válida para cualquier otra página del directorio «https://ejemplo.com/noticias/». Utilizando el parámetro path podemos 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 «http://dominio.com/blog/».

domain=<dominio>

Opcional. Por defecto, las cookies son **válidas sólo para el subdominio actual** en el que se crea la cookie (ten en cuenta que `www.ejemplo.com` se considera el subdominio `www`).

Esto quiere decir que si el atributo domain está vacío y estamos en `ejemplo.com`, la cookie es válida sólo para `ejemplo.com`, si estamos en `www.ejemplo.com`, la cookie será válida sólo para `www.ejemplo.com`, si estamos en `sub.ejemplo.com`, la cookie es válida sólo para `sub.ejemplo.com`, o sí estamos en `foo.sub.ejemplo.com`, la cookie es válida sólo para `foo.sub.ejemplo.com`.

Pero si el atributo domain no está vació, podemos especificar otros subdominios para los que la cookie es válida. Por ejemplo, `domain=sub.ejemplo.com` creará una cookie válida sólo para `sub.dominio.com`; si estamos en otro subdominio, por ejemplo `foo.ejemplo.com`, e intentamos leer esa cookie, no podremos, ya que sólo era válida para `sub.ejemplo.com`.

También podemos hacer que una cookie sea **válida para todo el dominio y sus subdominios** estableciendo `domain=.ejemplo.com` (nota el punto

delante del nombre del dominio). La especificación RFC 6265 antes mencionada, dice que **el primer punto será ignorado**, es decir, `domain=.ejemplo.com` **y** `domain=ejemplo.com` **tienen el mismo efecto y crean una cookie válida para cualquier subdominio**.

Lo anterior tiene una importante implicación. Si estamos en `ejemplo.com` y queremos una cookie válida sólo para `ejemplo.com`, la única opción es dejar el atributo `domain` vacío ya que si establecemos `domain=ejemplo.com` hacemos la cookie válida para todos los subdominios, no sólo para `ejemplo.com`.

Sin embargo, la especificación RFC 2965 sobre la cabecera `Set-Cookie2`, **requiere el punto precedente** para crear una cookie válida para todos los subdominos. Además, en la especificación RFC 2109 ya obsoleta, `domain=.ejemplo.com` y `domain=ejemplo.com` no eran tratados como lo mismo.

Por estos motivos, en mi opinión, **es recomendable utilizar siempre el punto precedente si queremos crear una cookie válida para todos los subdominios** ya que garantiza el mismo comportamiento en implementaciones antiguas que puedan serguir especificaciones obsoletas pero también en implementaciones modernas.

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

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 y APIs, por ejemplo, JavaScript.

Third-party cookies

Es importante aclarar que las cookies son enviadas y 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 haya cookies de ambos dominios. La cookie del 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 regla del same-origin(política del mismo origen** es una medida importante de seguridad para scripts en la parte cliente**)**.

Como utilizar cookies en JavaScript

La propiedad `document.cookie`

La propiedad `document.cookie` es todo lo que se necesita para trabajar con cookies client-side desde JavaScript. A través de ella podemos crear, leer, modificar y eliminar cookies. Veamos cada uno de los casos.

Crear cookies

Establecer una cookie en JavaScript es tan fácil como crear el string que define la cookie y asignarlo a `document.cookie`. Por ejemplo:

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

Si queremos crear varias cookies, tenemos que hacer este paso una vez para cada una. Por ejemplo, con el siguiente código se crearían las cookies `comida_favorita` y `color_favorito`:

```
document.cookie = "comida_favorita=arroz; max-age=3600; path=/";

document.cookie = "color_favorito=amarillo";
```

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*). 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**.

Recuerda que las cookies se envían en las cabeceras HTTP y, por tanto, deben estar correctamente codificadas. Puedes utilizar `encodeURIComponent()`, acostúmbrate a utilizarlo siempre para evitarte sorpresas:

```
var testvalue = "Hola mundo!";

document.cookie = "testcookie=" + encodeURIComponent( testvalue );
```

Si vas a utilizar el parámetro `expires`, recuerda que ha de ser una fecha en formato UTC. Te puede ser de ayuda el método `Date.toUTCString()`. Por ejemplo, una cookie con caducidad para el 1 de Febrero del año 2068 a las 11:20:

```
var expiresdate = new Date(2068, 1, 02, 11, 20);

var cookievalue = "Hola Mundo!";

document.cookie = "testcookie=" + encodeURIComponent( cookievalue ) + ";
expires=" + expiresdate.toUTCString();
```

Recuerda también que **si no se específica fecha de caducidad la cookie será válida sólo para la sesión actual**.

Modificar cookies

En el punto anterior se dijo que cada vez que se asigna una cookie a `document.cookie`, esta es añadida a la colección de cookies del documento. Esto es verdad excepto si la cookie asignada tiene un identificador que ya existe. En este caso se modifica la cookie existente en lugar de añadir una más.

Por ejemplo, podemos crear la siguiente cookie con identificador `nombre` y valor `Miguel`:

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

Si queremos modificar el valor, por ejemplo cambiarlo por Juan:

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

Es importante tener en cuenta que si una cookie se crea para un dominio o para un path determinado y se quiere modificar, **el dominio y el path han de coincidir**. De lo contrario se crearán dos cookies diferentes válidas para cada path y dominio. Por ejemplo, imaginemos que estamos en «miweb.com/blog» (el valor predeterminado del path es en este caso `/blog`):

```
// Supongamos que estamos en "miweb.com/blog"

// y creamos las siguientes cookies



// Creamos la cookie para el path "/"

document.cookie = "nombre=Miguel; path=/";



// Con la siguiente linea se crea una nueva cookie para el path "/blog"
(valor por defecto)

// pero no se modifica la cookie "nombre" anterior porque era para un
path diferente

document.cookie = "nombre=Juan";



// Con la siguiente línea SI se modifica la cookie "nombre" del path "/"
correctamente

document.cookie = "nombre=Juan; path=/";
```

Eliminar cookies

Para eliminar una cookie desde JavaScript se debe **asignar una fecha de caducidad (`expires`) pasada o un `max-age` igual a cero**. En ambos casos da igual el valor que se le asigne a la cookie porque se eliminará pero ha de darse el nombre de la cookie aunque sea sin valor.

Por ejemplo, creamos la cookie con el identificador `nombre` y valor `Miguel` igual que antes:

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

Si queremos eliminarla:

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


// O con max-age

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

Al igual que ocurría con la modificación de cookies, **para la eliminación el path y el dominio también tienen que coincidir**:

```
// Se crean dos cookies con el mismo identificador

// para dos paths diferentes

document.cookie = "nombre=Miguel; path=/noticias";

document.cookie = "nombre=Juan; path=/blog";


// Solo se elimina la cookie del path /noticias

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

Leer y obtener el valor de las cookies

Puede que obtener el valor sea el paso más tedioso de trabajar con cookies en JavaScript, y es que **no hay un método de lectura directo para cada cookie individual**. Sólo se puede obtener un string con todas las cookies válidas para el documento y manipular el string hasta encontrar el nombre y valor de la cookie que queremos.

El string con todas las cookies se obtiene del siguiente modo:

```
var lasCookies = document.cookie;
```

Y tiene el siguiente formato:

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

Quiero remarcar los siguientes aspectos:

El string sólo contiene pares de nombre de la cookie y su valor. **No se puede acceder a otros parámetros a través de** `document.cookie`.

Sólo se obtienen las **cookies válidas para el documento actual**. Esto implica que cookies para otros paths, dominios o cookies caducadas no se pueden leer. Aunque en una página puedan crearse cookies para otros subdominios y paths, sólo se pueden leer las que san válidas para el subdominio y path actual.

Por ejemplo, imagina que estamos en la subdominio `noticias.miweb.com`. Aquí podemos crear una cookie para el subdominio `tienda.miweb.com`, pero **esta cookie no es válida para el documento en el que estamos** (`noticias.miweb.com`), por lo que no podemos leer su valor desde aquí, aunque sí hemos podido crearla:

```
// Suponiendo que estamos en noticias.miweb.com



// Se crean dos cookies para dos subdominios diferentes

document.cookie = "cookienoticias=valorcn; domain=noticias.miweb.com";

document.cookie = "cookietienda=valorct; domain=tienda.miweb.com";



var lasCookies = document.cookie;

alert( lasCookies );

// Obtendremos cookienoticias=valorcn

// No podemos acceder a la cookie cookietienda

// porque es válida solo para tienda.miweb.com y estamos en
noticias.miweb.com
```

Ahora que sabemos cómo obtener las cookies, ¿cómo leemos los valores de cookies individuales? Pues tenemos que manipular el string con todas las cookies y dividirlo por cada `;` para separar cada par `nombrecookie=valor`. Luego se divide cada uno de estos pares por `=` para separar el nombre de la cookie y su valor. Se puede conseguir utilizando varios métodos, a continuación algunos de los más comunes:

Ejemplo con split

```
// Adapatado de http://www.quirksmode.org/js/cookies.html#script
function readCookie(name) {
  var nameEQ = name + "=";

  var ca = document.cookie.split(';');
  for(var i=0;i < ca.length;i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1,c.length);
    if (c.indexOf(nameEQ) == 0) {
      return decodeURIComponent( c.substring(nameEQ.length,c.length) );
    }
  }
  return null;

}

// Creamos una cookie
document.cookie = "pais=" + encodeURIComponent( "Uruguay" );

// Leemos la cookie
var miCookie = readCookie( "pais" );
// Muestra "Uruguay"
alert( miCookie );
```

Ejemplo con regex

```
function readCookie(name) {



  return decodeURIComponent(document.cookie.replace(new
RegExp("(?:(?:^|.*;)\\s*" + name.replace(/[\-\.\+\*]/g, "\\$&") +
"\\s*\\=\\s*([^;]*).*$)|^.*$"), "$1")) || null;



}



// Creamos una cookie

document.cookie = "pais=" + encodeURIComponent( "Ecuador" );



// Leemos la cookie

var miCookie = readCookie( "pais" );



// Muestra "Ecuador"

alert( miCookie );
```

Puedes elegir el que quieras o utilizar alguno de los cientos de snippets que circulan por la red que te harán la vida más fácil. Personalmente me gustan el framework que proponen en MDN y JS Cookie.