JavaScript – cookies y document.cookie

Introducción:

Una cookie es un archivo que una página web guarda en el ordenador del cliente, y proporciona un sistema para que desde el servidor se pueda identificar al cliente sin necesidad de registrarse en el servidor.

En esta web hay buena información sobre los tipos de cookies: https://www.xataka.com/basics/que-cookies-que-tipos-hay-que-pasa-desactivas

Persistencia de la información

Durante la navegación por una aplicación web, el usuario podrá visitar numerosos documentos HTML que se corresponderán con diferentes urls. Como las variables en JavaScript tienen una vida limitada al propio documento HTML en el que van insertas o enlazadas y su información desaparece cuando se carga un nuevo documento HTML, a menudo se necesita un sistema para disponer del valor establecido en algunas variables desde otro documento.

Cargar un archivo común donde definamos unas variables nos puede servir para disponer de dichas variables con su valor inicial en diferentes urls, pero si el valor de esa variable se modifica y continúa la navegación, no dispondremos de información sobre dicha modificación en el resto de urls, ya que “se perderán”.

Lo que vamos buscando es la persistencia de la información, es decir, que no desaparezca la información cuando cambiamos de url.

Hay diferentes maneras de abordar este problema que trataremos en éste y otros capítulos:

- Desde el lado cliente:
  - Utilizando cookies.

  - Utilizando almacenamiento en el navegador (localStorage y sesionStorage).
- Desde el lado servidor:
  - Transmitiendo y almacenando información en una base de datos para recuperarla posteriormente.

Concepto de cookie

Las cookies fueron creadas por trabajadores de la empresa Netscape como forma de dar una respuesta sencilla a la necesidad de almacenar información relacionada con la identificación de usuarios y acciones que un usuario desarrolla durante la navegación. Por ejemplo, se planteaba la siguiente cuestión: si un usuario accede a una tienda web y queremos que pueda ir agregando productos a un carrito de compras, ¿cómo saber qué productos ha almacenado cuando cambia de url? Sin el uso de alguna forma de dotar de persistencia a la información, la información se perdía. Y cierta información sería muy costoso manejarla con herramientas como bases de datos.

La forma de solucionarlo fue inventar las cookies. Las cookies son información que se almacena en el navegador de forma persistente, es decir, que no desaparece cuando el usuario navega a través de diferentes urls. Además, las cookies son enviadas al servidor cuando el usuario hace una petición, de modo que, el servidor puede conocer esa información y actuar en consecuencia.

Las cookies pueden ser creadas de diferentes maneras, por ejemplo:

a) **Ser creadas por el servidor**, y enviarlas al navegador para que las almacene. Supongamos que entramos en una página web de compras por internet. En ese momento somos usuarios anónimos, pero una vez introduzcamos nuestro nombre de usuario y password (por ejemplo, supongamos que somos el usuario Albert Einstein), el servidor envía una cookie al navegador que podría ser:
session_id_= 6n4465736gf9863b52e641757fa0b7db, donde **session_id** es el nombre la cookie y la cadena de letras y números el valor que tiene la cookie, lo que permitirá saber al servidor que quien hace una petición es la misma persona (el mismo computador cliente) que había hecho una petición anteriormente. La comprobación de que sea el mismo computador cliente (realmente se refiere al mismo navegador cliente) se basa en comparar el valor de esa cadena larga: si coincide, es el mismo usuario, si es diferente, es otro usuario.

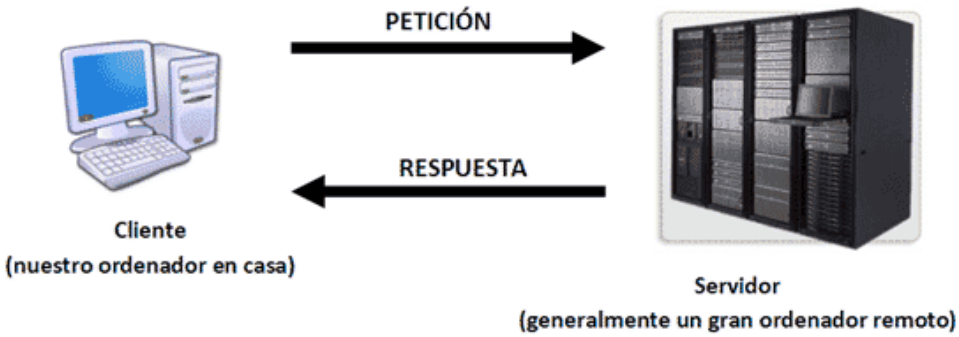
   Un usuario no puede conocer el valor de la cookie que tiene otro debido a que existen prácticamente infinitas posibilidades de combinación de letras y números. Si se usan cadenas arbitrariamente largas, es prácticamente imposible averiguar por casualidad o mediante pruebas el contenido de una cookie.

   Ahora bien, cada vez que cambiemos de url esta cookie es enviada al servidor. El servidor lee la cookie y comprueba: ¡esta cookie pertenece al usuario Albert Einstein!

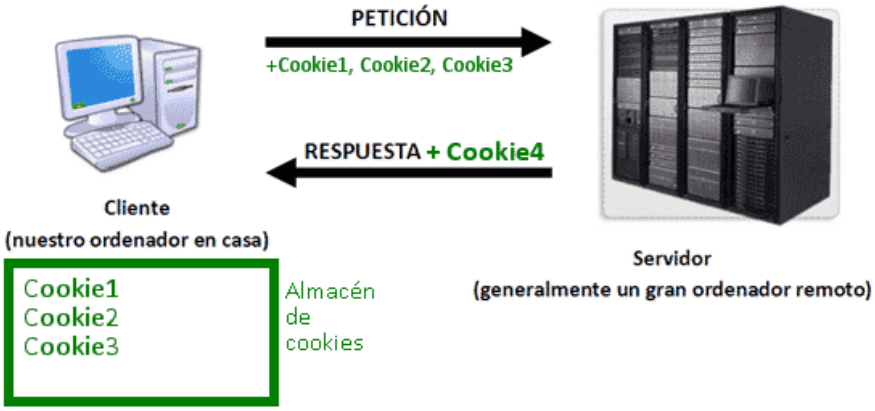
   Aunque el usuario haya cambiado de url, la cookie informa en cada momento que él es Albert Einstein, con lo cual no hace falta estar introduciendo en cada url que se visite el nombre de usuario y password.

b) **Ser creadas mediante JavaScript** en el navegador, almacenarse en el navegador y enviarse posteriormente al servidor en cada comunicación que tenga lugar.

   Simplificadamente podríamos decir que navegar sin cookies sería algo similar a lo que se muestra en este esquema, donde cada petición al servidor equivale a una “petición nueva”:



Mientras que navegar con cookies sería algo similar a lo que se muestra en este esquema, donde el navegador está almacenando información en un almacén de cookies y con cada petición que hace esas cookies son transmitidas (y además pueden irse añadiendo más cookies a las ya existentes previamente):



Resumiendo: las cookies son datos que se almacenan en el navegador del usuario.

Las cookies pueden presentar algunos problemas de seguridad, pero en general se considera que si se utilizan de forma adecuada son un mecanismo que resulta práctico para realizar muchas de las tareas que normalmente se requieren en la navegación web.

Tiempo de vida de las cookies

Las cookies son datos temporales, es decir, su intención no es almacenarse "para siempre", sino almacenarse por un tiempo para facilitar la navegación. Una cookie puede tener asociada una fecha de borrado o expiración, en cuyo caso permanecerá en el navegador del usuario hasta que llegue dicha fecha (a no ser que el usuario decida hacer un borrado de cookies). En este caso, puede ocurrir que el usuario cierre el navegador y lo abra al cabo de unas horas o días, y la información en forma de cookies siga estando ahí. Las cookies con fecha de borrado se suelen llamar cookies persistentes, porque no se destruyen excepto cuando llega la fecha de expiración.

Otras cookies no tienen fecha de borrado o expiración, o si la tienen es muy corta (pongamos que una hora de duración). Si la cookie no tiene fecha de borrado, se destruye cuando se cierra el navegador.

Hay que tener en cuenta que los navegadores pueden almacenar información de otras maneras además de como cookies (por ejemplo, como opciones de configuración, perfiles de usuario, contraseñas, etc.).

Además, las cookies (al igual que JavaScript) pueden desactivarse en los navegadores. La mayoría de los usuarios navegan con cookies activadas (al igual que con JavaScript activado), pero teóricamente un usuario puede deshabilitarlas.

Contenidos de las cookies

Las cookies podemos verlas como pequeños ficheros de texto que se almacenan en el navegador, cuyo contenido es el siguiente:

1. Un par nombre – valor que define la cookie. Por ejemplo, el nombre puede ser **user_name** y el valor **pelaez**.
2. Una fecha de caducidad (en algunos casos, estará indefinida, con lo cual la cookie será borrada cuando se cierre el navegador). La fecha se expresa en tiempo UTC, o como un número de segundos desde el momento actual.
3. El dominio y ruta del servidor donde la cookie fue definida, lo que permite que si existieran dos cookies con el mismo nombre se pudiera saber qué cookie corresponde a cada url que se visita. No está permitido falsear dominios, es decir, si el dominio desde el que se establece una cookie es daw2cliente.com, no se podría poner como información asociada a la cookie que el dominio es microsoft.com, sino que la cookie únicamente puede ir asociada a daw2cliente.com. Esto permite que, si se está navegando por un sitio, no haya necesidad de enviar todas las cookies almacenadas en el navegador al servidor de ese sitio, sino únicamente las cookies relacionadas con ese sitio.

La ruta permite especificar un directorio específico al cual se debe considerar asociada la cookie. De este modo, el navegador no tendría que enviar la cookie a todas las páginas de un dominio, sino solo a las páginas concretas cuya ruta esté definida. Normalmente la ruta definida es simplemente << / >> lo que significa que la cookie es válida en todo el dominio.

Cookies con JavaScript: document.cookie

Las cookies podemos verlas como pequeños ficheros de texto que se almacenan en el navegador pero que desde el punto de vista de Javascript es una cadena de texto que contiene parejas *clave=valor* separadas por ; (punto y coma).

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

Para obtener todas las cookies accesibles desde una localización se utiliza la propiedad:

```
let todasLasCookies = document.cookie;
```

Crear una cookie

Para crear una cookie con JavaScript usaremos la siguiente sintaxis:

document.cookie=nuevaCookie;

donde nuevaCookie es una cadena de texto con pares clave-valor separadas por punto y coma.

Los atributos son:

<nombre>=<valor>

Requerido. `<nombre>` es el nombre (key) que identifica la cookie y `<valor>` es su valor. A diferencia de las cookies en PHP, en JavaScript se puede crear una cookie con un valor vacío (`<nombre>=`).

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

Opcional. Ambos parámetros especifican el tiempo de validez de la cookie. `expires` establece una fecha (ha de estar en formato UTC) mientras que `max-age` establece una duración máxima en segundos. `max-`

`age` toma preferencia sobre `expires`. Si no se especifica ninguno de los dos se creará una **session cookie**. Si es `max-age=0` o `expires=fechaPasada` la cookie se elimina.

path=<ruta>

Opcional. Establece la ruta para la cual la cookie es válida. Si no se especifica ningún valor, la cookie será válida para la ruta la página actual.

domain=<dominio>

Opcional. Dentro del dominio actual, se puede indicar el subdominio para el que la cookie es válida. El valor predeterminado es el subdominio actual. Establecer `domain=.miweb.com` para una cookie que sea válida para cualquier subdominio (nota el punto delante del nombre del dominio). Por motivos de seguridad, los navegadores no permiten crear cookies para dominios diferentes al que crea la cookie (*same-origin policy*).

secure

Opcional. Atributo sin valor. Si está presente la cookie sólo es válida para conexiones encriptadas (por ejemplo, mediante protocolo HTTPS).

Ejemplos de creación de cookie:

```
document.cookie= "nombreCookie=valorCookie; expires=fechaDeExpiración; path=rutaParaLaCookie";
```

Se pueden añadir a la cadena de texto otros parámetros clave-valor.

También se puede dejar sin especificar **expires** (la cookie se borrará al cerrar el navegador) y **path** (la cookie quedará asociada al dominio), con lo que la definición quedaría:

```
document.cookie = 'nombreCookie=valorCookie;';
```

Las cookies se envían en las cabeceras HTTP y, por tanto, deben estar correctamente codificadas. Conviene utilizar `encodeURIComponent()` para evitar sorpresas. Ejemplo:

```
let testvalue = "Hola mundo!";
document.cookie = "nombreCookie=" + encodeURIComponent( testvalue );
```

Si se va a utilizar el atributo `expires`, ha de ser con una fecha en formato UTC. 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:

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

En lugar de **expires** se puede usar **max-age**. En este caso, en lugar de especificar una fecha concreta se especifica el número de segundos desde la creación de la cookie hasta su caducidad. Por ejemplo:

```
document.cookie = "color=blue; max-age=" + 60*60*24*30 + "; path=/; domain=daw2cliente.com; secure"
```

El ejemplo anterior serviría para indicar que el nombre de la cookie es *color*, su valor *blue*, su fecha de expiración 30 días (expresado en segundos resulta 30 días * 24 horas/día * 60 minutos/hora * 60 segundos/minuto").

Si no se especifica max-age ni expires, la cookie expirará al terminar la sesión actual. Si se especifican ambos atributos, tiene prioridad max-age.

La propiedad document.cookie

Cuando se van creando cookies, éstas se van añadiendo a document.cookie (es decir, document.cookie no funciona como una propiedad que se vaya sobrescribiendo, sino que cada definición de document.cookie añade una cookie a la colección de cookies del documento). Este comportamiento se debe a que document.cookie no es un dato con un valor, sino una propiedad de acceso con métodos set y get nativos. 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.

Modificar una cookie

Una cookie se puede redefinir: para ello simplemente hemos de usar document.cookie indicando el mismo nombre de cookie que existiera previamente. Los datos de esa cookie serán sobrescritos, quedando reemplazados los anteriormente existentes por los nuevos.

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 SÍ se modifica la cookie "nombre" del path "/" correctamente
document.cookie = "nombre=Juan; path=/";
```

Eliminar cookies

Una cookie se puede eliminar: para ello hemos de usar document.cookie indicando el nombre de cookie que queremos eliminar y establecer una fecha de expiración ya pasada (por ejemplo del día de ayer, o simplemente indicar **expires=Thu, 01 Jan 1970 00:00:00 UTC;**). El navegador al comprobar que la fecha de caducidad de la cookie ha pasado, la eliminará.

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

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

Tenemos dos formas de eliminarla:

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

// O con max-age
document.cookie = "nombre=; max-age=0”;
```

Recuperar el contenido de cookies

Para recuperar el contenido de cookies hemos de trabajar con document.cookie como si fuera una cadena de texto

Para recuperar el valor de la cookie, hemos de buscar dentro de esa cadena de texto (string). Para ello usaremos las herramientas que nos proporciona JavaScript.

```
let todasLasCookies = document.cookie;
```

En la variable todasLasCookies las cookies se organizan de la siguiente manera:

```
nombreCookieA=valorCookieA; nombreCookieB=valorCookieB; … ; nombreCookieN = valorCookieN;
```

A tener en cuenta:

- 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 sean válidas para el subdominio y path actual.

Por ejemplo, imagina que estamos en el 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";

let 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
```

Para leer el contenido de cookies individuales se hace manipulando 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.

Ejercicio: Realiza y prueba una función que obtenga el valor de una cookie pasando su nombre como parámetro.

Ejercicio: utilizando expresiones regulares, realiza y prueba una función que obtenga el valor de una cookie pasando su nombre como parámetro.

Saber si las cookies están habilitadas o no.

Un usuario puede eliminar las cookies de su navegador. También puede desactivar las cookies. Por ello, puede ser interesante saber si están activadas, se sabe leyendo el objeto **navigator** y la propiedad booleana **cookieEnabled**.

Ejercicio: Realiza una página web que muestre un mensaje en el documento indicando si las cookies están o no activadas. Prueba la función activando y desactivando las cookies de tu navegador.

Ejercicio: HacerAlgoUnaSolaVez

```
/*
Para usar el siguiente código: reemplaza todas las veces la
palabra hacerAlgoUnaSolaVez (el nombre de la cookie) con un nombre personalizado.
*/
function hazUnaVez() {
    if (document.cookie.replace(/(?:(?:^|.*;\s*)hacerAlgoUnaSolaVez\s*\=\s*([^;]*).*$)|^.*$/, "$1") !== "true") {
       alert("Hacer algo aquí!");
       document.cookie = "hacerAlgoUnaSolaVez=true; expires=Fri, 31 Dec 9999 23:59:59 GMT";
     }
  }

  <button onclick="hazUnaVez()">Solo hacer algo una vez</button>
```