

1 Introducción

Javascript aplicado a navegadores en principio no está pensado para almacenar grandes cantidades de datos. Existen dos formas de almacenar información en Javascript:

- **Cookies:** para guardar pequeñas variables con información. Usualmente suelen guardar información de logins de usuarios. Las cookies se pueden guardar en el cliente y ser generadas por el cliente, pero también pueden ser enviadas por el servidor.
- **LocalStorage:** al crecer Javascript y las aplicaciones asociadas a ellos, los navegadores más modernos implementaron LocalStorage. Es más parecido a guardar datos en una aplicación de escritorio. El límite de información a almacenar puede variar según la implementación del navegador, pero está definido en torno a los 5 MB.

2 Cookies

Una cookie es una información enviada por un sitio web (y asociada a ese dominio Web) que el navegador se encarga de almacenar, es decir, se almacenan en el cliente y no en el servidor.

Generalmente suelen estar guardadas en fichero de texto, aunque se utilizarán comandos específicos de Javascript que abstraen de cómo son almacenadas. Básicamente, esta información consiste en una o varias variables con su contenido asociado.

Un ejemplo: una página de “midominio.com” crea una cookie. Esta cookie contiene una variable “usuario” y su contenido es “pepe”. Esta cookie sólo es accesible desde el navegador desde “midominio.com”. Una página del dominio “pepe.com” no podría modificarla ni leerla y si creara una cookie con la variable usuario, sería una cookie independiente.

Más información en [https://es.wikipedia.org/wiki/Cookie_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Cookie_(inform%C3%A1tica))

3.1 Formato de cookies

Para crear una cookie, usamos document.cookie. Esto es una string especial que tiene el siguiente formato:

“variable=valor;expires=fecha expiración;path=/”

Donde variable es la variable a establecer, valor su valor, expires es la fecha de expiración (la forma de borrar cookies es cambiarles la fecha de expiración a una ya pasada) y path el lugar del dominio donde son válidas.

Ejemplo:

```
document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013  
12:00:00 UTC; path=/";
```

Para una explicación detallada del formato y uso de cookies desde Javascript, se puede acudir aquí http://www.w3schools.com/js/js_cookies.asp

A efectos prácticos, se recomienda el uso de estas funciones ya establecidas para crear, consultar y eliminar cookies, obtenidas de la página citada anteriormente.

setCookie: establece una cookie indicandole variable, valor y días para la expiración

```
function setCookie(cname, cvalue, exdays) {    Si son 7 días, en parámetro pones exdays=7  
var d = new Date();  
  
d.setTime(d.getTime() + (exdays*24*60*60*1000));  
var expires = "expires=" + d.toUTCString(); En la construcción de la cookie, el formato para  
document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/"; la expiración es expires=fecha.  
}  
Se añade path=/ para hacer que la cookie sea accesible en todo el sitio.
```

getCookie: recibe el nombre de la variable y devuelve su valor

```
function getCookie(cname) {Crees una variable name con el nombre de la cookie que buscas, junto con el  
signo igual =. Lo haces para poder buscar esa cadena exacta en las cookies  
almacenadas.  
var name = cname + "=";  
var ca = document.cookie.split(';'); Aquí tomas todas las cookies del navegador (que están en  
document.cookie. Cada elemento de este array es una cookie completa, por ejemplo "nombre=Juan"  
for(var i = 0; i < ca.length; i++) {  
  
var c = ca[i];  En cada vuelta del bucle, tomas una cookie del array y la guardas en la varia  
while (c.charAt(0) == ' ') { Este while elimina los espacios en blanco al principio de la cookie c. Es  
decir, si la cookie empieza con un espacio, lo quita para facilitar la  
c = c.substring(1); búsqueda.  
  
} Aquí compruebas si la cookie c empieza exactamente con el texto qu  
buscas (name). La función indexOf devuelve 0 si la cadena name está  
if (c.indexOf(name) == 0) { comienzo. Esto indica que la cookie que estás mirando es la que quie  
  
return c.substring(name.length, c.length);  
} La función substring extrae desde el final del nombre  
(name.length) hasta el final de la cookie, que es el valor que  
quieres obtener. Es decir, elimina el nombre y el signo igual,  
dejando solo el valor de la cookie.  
}  
return "";  
}
```

OJO CUANDO LLAMAS ES getCookie("cname");

Elimina la cookie de la variable establecida

```
function deleteCookie(cname) {  
  
document.cookie = cname+'='; expires=Thu, 01 Jan 1970 00:00:01  
GMT;path='';  
}
```

2

OJO Cuando haces document.cookie, el navegador te devuelve todas las cookies disponibles para esa página web en una sola cadena de texto, no solo la que creaste. Esta cadena contiene todos los pares nombre=valor separados por punto y coma (;). Por eso hacemos el split ; y buscamos el nombre de lo que nos interesa

3.2 Probando el funcionamiento de las cookies.

Se puede probar usando esta función, que intenta obtener una cookie "username". Si existe, muestra un mensaje. Si no existe, la establece. Lógicamente este ejemplo se puede combinar con todo lo aprendido anteriormente de Javascript.

```
function checkCookie() {  
    var user = getCookie("username");  
    if (user != "") {  
        alert("Welcome again " + user);  
    } else {  
        user = prompt("Please enter your name:", "");  
        if (user != "" && user != null) {  
            setCookie("username", user, 365);  
        }  
    }  
}  
el set y get cookie es genérico, cuando construyo la cookie la llamo ejemplo  
setCookie("nombre",valorNombre,30);  
igual con el var usuario=getCookie("nombre");
```

3 LocalStorage

El texto localStorage es una tecnología de almacenamiento existente en los navegadores más modernos, siendo incompatible con navegadores antiguos. La información se almacena en el cliente y generalmente posee al menos 5MB para guardar información.

Para más información http://www.w3schools.com/html/html5_webstorage.asp

A efectos prácticos es importante conocer:

- Hay dos objetos: localStorage y sessionStorage. La diferencia entre uno y otro es que localStorage almacena la información indefinidamente y sessionStorage lo hace sólo mientras la ventana de la página este abierta. Por el resto de los detalles ambos objetos funcionan igual.
- Las funciones a utilizar son 3: setItem, getItem, removeItem.

Ejemplo setItem, getItem y removeItem

```
localStorage.setItem("apellido", "Garcia");  
alert(localStorage.getItem("apellido"));  
localStorage.removeItem("apellido");  
alert(localStorage.getItem("apellido"));
```