Tema 7

Cookies

Cookies

Cada vez que pedimos una página web en el servidor se crear una conexión nueva y el servidor no sabe que hicimos anteriormente.

Esto se puede evitar con las cookies. Las cookies son unos ficheros que se crean en cliente y contiene información que es útil para la navegación por todo el sitio web. Este fichero tiene una limitación de 4 KB y sólo se pueden guardar 300 cookies. Por cada dominio o servidor puede almacenar 20 cookies, este dato depende de los navegadores.

Ideas equivocadas

Desde su introducción en Internet han circulado ideas equivocadas acerca de las cookies. En 2005 Jupiter Research publicó los resultados de un estudio, según el cual un importante porcentaje de entrevistados creían cierta alguna de las siguientes afirmaciones:

- Las cookies son similares a gusanos y virus en que pueden borrar datos de los discos duros de los usuarios.
- Las cookies son un tipo de spyware porque pueden leer información personal almacenada en el ordenador de los usuarios.
- Las cookies generan popups.
- Las cookies se utilizan para generar spam.
- Las cookies sólo se utilizan con fines publicitarios.

En realidad, las cookies son sólo datos, no código, luego no pueden borrar ni leer información del ordenador de los usuarios. Sin embargo, las cookies permiten detectar las páginas visitadas por un usuario en un sitio determinado o conjunto de sitios. Esta información puede ser recopilada en un perfil de usuario. Estos perfiles son habitualmente anónimos, es decir, no contienen información personal del usuario (nombre, dirección, etc). De hecho, no pueden contenerla a menos que el propio usuario la haya comunicado a alguno de los sitios visitados.

Pero aunque anónimos, estos perfiles han sido objeto de algunas preocupaciones relativas a la privacidad.

Según el mismo informe, un gran porcentaje de los usuarios de Internet no saben cómo borrar las cookies.

Fuente: wikipedia

Para trabajar las cookies con jQuery necesitamos un pluging que lo podemos descargar desde la pagina de jQuery. Nosotros no lo vamos a utilizar al tratarse de un pluging.

Creación de cookies

Para crear las cookies en javascript utilizaremos el objeto cookie del objeto document.

```
document.cookie="variable=valor;.....parametros. ....";
```

Por cada valor que introducimos en la cookie hay que crear una cookie nueva.

```
<html>
<head>
<title> cookies01.html</title>
<meta charset="UTF-8" />
<script>
window.onload=function(){
   document.cookie="variable=valor";
   document.cookie="variable2=valor2";
   document.cookie="todo=var1\=valor1:var2\=valor2"; // ojo con esta línea
 };
 </script>
</head>
<body>
<div id="capa">
<p id="parrafo1"> Hemos creado una cookie. pulsar F12 - Application - cookies -
localhost</p>

</div>
</body>
</html>
```

Parámetros de la cookie

name: nombre de la cookie. El nombre debe contener letras de la a-z, A-Z números de 0-9 y el carácter de subrayado. Es el nombre de la variable de la cookie.

expires: Indica cuando debe expirar la cookie. La cadena asignada a este parámetro debe ser UTC el cual puede establecerse utilizando el método toUTCString() o toGMTString() del objeto Date.

max-age: Indicamos en segundos cuanto queremos que dure nuestra cookie.

domain: Nombre del dominio al que pertenece la cookie. Normalmente no se utiliza. Por omisión pondrá el nombre del servidor que creó la cookie.

path: Ruta o directorio que utiliza la cookie. Por omisión se toma la ruta de la página que creó la cookie. Al establecer el camino para una cookie utilice la barra inclinada a derecha (/) en lugar de la barra inclinada a la izquierda (\). No termine la cadena con la barra. `/documento/ver`. Las cookies se reconocen desde la posición de la página hacia el raíz

secure: Para utilizar la cookie con seguridad SSL por medio del protocolo https:. La cookie sólo es válida en conexiones encriptadas.

El único parámetro que es obligatorio es “name” todos los demás parámetros son opcionales.

```
<html>
<head>
<title> cookies02.html</title>
<meta charset="UTF-8" />
<script>
window.onload=function(){
```

```
      var fecha=new Date()
      console.log(fecha.toUTCString());
      fecha.setTime(fecha.getTime()+60*60*1000);
      console.log(fecha.toUTCString());
      document.cookie="local=valorlocal";
     document.cookie="variable=valor;max-age="+60*60*24*30+";path=/curso2021";

   };
   </script>
  </head>
  <body>
  <div id="capa">
  <p id="parrafo1"> Hemos creado una cookie. pulsar F12 - Application - cookies -
  localhost</p>

  </div>
  </body>
  </html>
```

Crear una cookie.

Creemos nuestra primera cookie.

```
document.cookie="usuario1=Manuel;expires=Sat, 10 Aug 2021 12:15:00 GMT";

var valor="Tomas";
document.cookie="usuario2="+valor+";expires=Sat, 6 Aug 2021 12:20:00 GMT";
```

Para poder probar las cookies estamos obligados a trabajar sobre un servidor web.

Leer una cookie.

Para leer una cookie, solamente tenemos que acceder a la cookie y asignarlo a una variable.

```
cadena=document.cookie;
```

Esto nos devuelve una cadena de todas las cookies de este sitio separadas por punto y coma ";"

```
prueba0=0; prueba1=1; prueba2=2; prueba3=3
```

Esta cadena la tendremos que manipular con métodos del objeto String para obtener los valores de las cookies.

```
array_cookies=cadena.split(";");
```

Creamos un array con la pareja `variable=valor` con todas las cookies del dominio y del path. Si deseamos, luego podemos crear un array bidimensional donde cada item del array contenga otro array con la pareja variable,valor.

```
<html>
<head>
<title> cookies04.html</title>
<meta charset="UTF-8" />
<script>
window.onload=function(){
    miscookies=getcookies();
    if(miscookies!==null){
```

```
        for(a in miscookies){
           console.log(a + " =  "+ miscookies[a]);
        }
     } else console.log("no hay cookies");

  };
  function getcookies(){
     var micoo=null;
     var cadena=document.cookie;
     var miarray="";
     var mimini;
     if(cadena.trim().length>0){
        miarray=cadena.split(";");
        micoo={};
        for(a=0;a<miarray.length;a++){
           mimini=miarray[a].split("=");
           micoo[mimini[0].trim()]=mimini[1].trim();
        }
    }
    return micoo;
  }
  </script>
</head>
<body>3
<div id="capa">
<p id="parrafo1"> Hemos creado una cookie. pulsar F12 - Resource - cookies -
localhost</p>

</div>
</body>
</html>
```

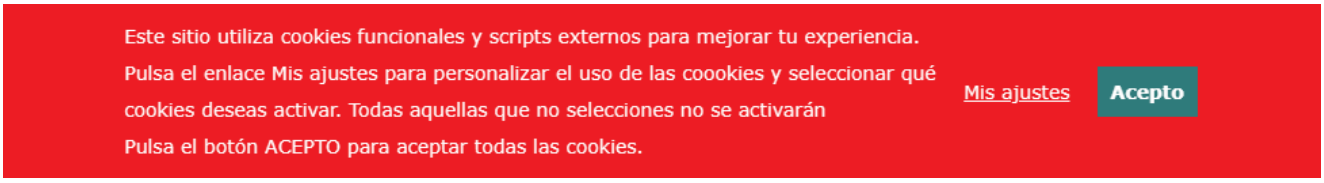
Borrar una cookie

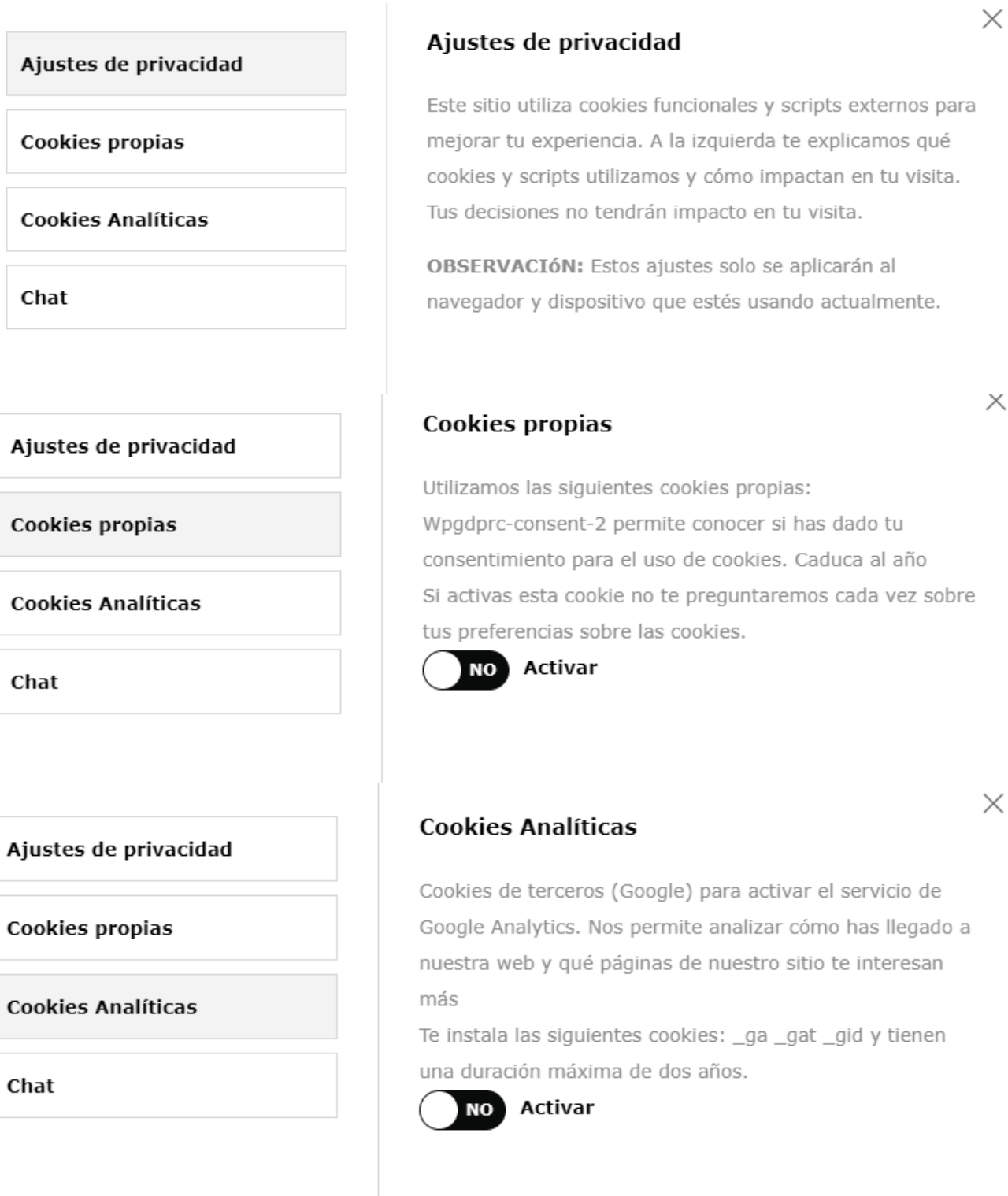
Para borrar una cookie sólo tenemos que reescribir la cookie con un valor nulo y poner una fecha ya pasada en la entrada expires (max-age).

Guía sobre el uso de las cookies (Agencia Española de Protección de Datos) Julio 2020

https://www.aepd.es/sites/default/files/2020-07/guia-cookies.pdf

Aviso cookie en web




Ajustes de privacidad
Cookies propias
Cookies Analíticas
Chat
Ajustes de privacidad
Este sitio utiliza cookies funcionales y scripts externos para mejorar tu experiencia. A la izquierda te explicamos qué cookies y scripts utilizamos y cómo impactan en tu visita. Tus decisiones no tendrán impacto en tu visita.
OBSERVACIóN: Estos ajustes solo se aplicarán al navegador y dispositivo que estés usando actualmente.
Ajustes de privacidad
Cookies propias
Cookies Analíticas
Chat
Cookies propias
Utilizamos las siguientes cookies propias:
Wpgdprc-consent-2 permite conocer si has dado tu consentimiento para el uso de cookies. Caduca al año
Si activas esta cookie no te preguntaremos cada vez sobre tus preferencias sobre las cookies.
NO
Activar
Ajustes de privacidad
Cookies propias
Cookies Analíticas
Chat
Cookies Analíticas
Cookies de terceros (Google) para activar el servicio de Google Analytics. Nos permite analizar cómo has llegado a nuestra web y qué páginas de nuestro sitio te interesan más
Te instala las siguientes cookies: _ga _gat _gid y tienen una duración máxima de dos años.
NO
Activar

Tema 9

localStorage, sessionStorage

localStorage y sessionStorage

Para trabajar con estos objetos tenemos ciertas diferencias según los navegadores. Dependiendo del navegador y de la versión del mismo, tenemos que trabajar sobre un servidor web o podemos trabajar abriendo la página desde el escritorio. Para que no haya ningún problema se aconseja trabajar siempre sobre un servidor web.

sessionStorage es un objeto global de javascript. Cuando abrimos una página web se mantiene un área de almacenamiento que está disponible durante la sesión de página. Si abrimos una página en otra pestaña o ventana se creará una sesión nueva. Transcurrido un periodo de tiempo, normalmente 15 minutos, sin realizar actividad los datos de sesión se borrarán.

localStorage es exactamente igual a sessionStorage con la diferencia que los datos se guardan en cliente, por tanto, funciona de una forma muy parecida a las cookies y esto quiere decir, que los datos no desaparecen al cerrar el navegador.

Las ventajas de utilizar localStorage frente a las cookies es su capacidad. Con localStorage podemos almacenar hasta 5 MB, en algunos navegadores y en otros incluso hasta 10 MB. También, tanto localStorage como sessionStorage, poseen una serie de métodos que nos facilita el manejo de la información. La especificación de HTML5 recomienda a los fabricantes de navegadores que reserven un mínimo de 5 MB para cada origen.

A continuación explicaremos como trabajar con estos objetos globales. Para simplificar utilizaremos uno de ellos. Los dos objetos tienen las mismas propiedades y métodos. Si existe alguna diferencia ya se comentará en su momento.

Comprobamos si nuestro navegador soporta estos objetos.

Para comprobar si podemos usar o no estos objetos introduciremos el siguiente código.

```
if(localStorage)
    alert("Puedo trabajar con localStorage");
else
    alert("Esto no funciona bien");
```

Ahora veamos este texto dentro de una página html.

storage01.html

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>storage01</title>

<script>
  window.addEventListener("load",function(){
     var boton=document.getElementById("pulsa");
```

```
      boton.addEventListener("click",function(){
        if(localStorage) alert("localStorage existe");
          else  alert("NO existe");
        if(sessionStorage) alert("sessionStorage existe");
          else  alert("NO existe");
      });
    });
  </script>
  </head>
  <body>
       <div><p>Ejemplo para comprobar si podemos usar localStorage
              y sessionStorage</p>
       <p><button id="pulsa">Comprobación</button></p></div>
  </body>
  </html>
```

Almacenar información en storage

Utilizaremos el método ***setItem(clave,valor)***. Es un método que recibe dos parámetros, el primero es para indicar el nombre de la variable a guardar y el segundo es el valor que deseamos guardar.

```
localStorage.setItem("curso","2° DAW");
```

También podemos almacenar datos añadiendo propiedades al objeto global.

```
localStorage.grupo="2DAWA";
```

Otro modo de guardar información es utilizando localStorage como un array asociativo, es decir, que el índice del array sea una cadena.

```
localStorage["numero"]=30;
```

Los datos que guardamos son siempre convertidos a cadenas (string).

```
alert(typeof(localStorage["numero"])); // string
```

Veamos ahora un ejemplos

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>storage02</title>
<script>
  window.onload=function(){
     localStorage.setItem("variable1","valor1");
     localStorage.variable2="valor2";
     localStorage["variable3"]=30;
     alert(typeof localStorage["variable3"]);
  };
</script>
</head>
<body>
       <div><p>Ejemplo para comprobar como se almacena información en
storage</p>
       <p><button id="pulsa">Comprobación</button></p></div>
</body>
</html>
```

Recuperar información de storage

Para leer datos podemos utilizaremos el método ***getItem(clave)***. Este método recibe como parámetro el nombre de la clave que se desea recuperar.

```
alert(localStorage.getItem("numero"));
```

También podemos acceder al dato accediendo como una propiedad del objeto.

```
alert(localStorage.curso);
```

Y también como array asociativo

```
alert(localStorage["grupo"]);
```

Como siempre almacena cadenas, los datos que recuperamos de storage son cadenas.

Obtener el número de variables guardadas

Con la propiedad ***length*** podemos averiguar el número de elementos almacenados.

```
alert(localStorage.length); // 3
```

En el siguiente ejemplo veremos como podemos acceder a los datos almacenados y como averiguar el número de datos almacenados.

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>storage03</title>

<script>
  window.addEventListener("load",function(){
     var boton=document.getElementById("pulsa");
     boton.addEventListener("click",function(){
        alert("existen "+localStorage.length+" elementos grabados");
        alert("variable 1 --> " + localStorage.getItem("variable1"));
        alert("variable 2 --> " + localStorage.variable2);
        alert("variable 3 --> " + localStorage["variable3"]);
     });
  });
</script>
</head>
<body>
      <div><p>Ejemplo para recuperar datos en storage</p>
      <p><button id="pulsa">Comprobación</button></p></div>
</body>
</html>
```

Eliminar una clave

Para eliminar una clave utilizaremos el método ***removeItem(clave)***. Elimina la clave indicada del objeto localStorage.

```
localStorage.removeItem("curso");
```

Para comprobar si una variable existe en el storage va a depender de como accedamos a dicha variable nos responderá de un modo u otro, es decir, nos puede devolver el valor *null* o *undefined*.

La variable localStorage[“pp”] no existe, por tanto...

```
console.log(localStorage.getItem("pp")); // null
if(localStorage.getItem("pp")===null) alert("pp no existe");

console.log(localStorage.pp); // undefined
if(localStorage.pp===undefined) alert("pp no existe");

console.log(localStorage["pp"]); // undefined
if(localStorage["pp"]===undefined) alert("pp no existe");
```

Si en lugar de usar tres iguales (===) utilizamos dos (==) nos daría igual comparar la existencia de la variable con *null* o con *undefined*.

Recordar que *undefined* es un valor que recibe una variable cuando es declarada y no se le asigna ningún valor. *undefined* es una propiedad del objeto global que representa el valor *undefined*, existe *window.undefined*. null es un objeto y se puede asignar a una variable.

```
typeof localStorage["pp"]; // "undefined"
typeof localStorage.getItem("pp"); // "object"
```

Eliminar todos los elementos de storage

Utilizamos el método clear(). Con este método podemos borrar todas las entradas del objeto.

```
localStorage.clear();
alert(localStorage.length); // 0
```

Ahora veamos un ejemplo

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>storage04</title>

<script>
  window.addEventListener("load",function(){
     document.getElementById("boton").addEventListener("click",function(){
        localStorage.clear();
          alert("ahora comprobar que se ha borrado todo");
     });
     document.getElementById("boton1").addEventListener("click",function(){
          if(localStorage.getItem("variable1")===null)
             alert("variable1 no existe");
          if(localStorage.variable2===undefined)
             alert("variable2 no existe");
          if(localStorage["variable3"]===undefined)
             alert("variable3 no existe");
     });
     document.getElementById("boton2").addEventListener("click",function(){
          localStorage.removeItem("variable1");
        alert("comprobar que se ha borrado variable1");
     });
     document.getElementById("anyadir").addEventListener("click",function(){
          localStorage.setItem("variable1","valor1");
          localStorage.setItem("variable2","valor2");
          localStorage.setItem("variable3",30);
     });
  });
```

```
</script>
</head>
<body>
      <div>localStorage y sessionStorage. Comprobar valores y borrar
valores</div>
      <button id="anyadir">Crear valores</button>
      <button id="boton">Borrar todo</button>
      <button id="boton1">Comprobar</button>
      <button id="boton2">Borrar "variable1"</button>

</body>
</html>
```

Mostrar el nombre de las claves

El método ***key(num)***, nos devuelve el nombre de la clave de la posición pasada como parámetro.

```
localStorage.setItem("curso","2° DAW");
localStorage.grupo="2DAWA";
localStorage["numero"]=30;

alert(localStorage.key(1)); // grupo
```

RECUERDA. Ya hemos estudiando JSON y como convertir un objeto a cadena (JSON.stringify) y como convertir una cadena a objeto (JSON.parse). Por tanto, ahora ya sabes como puedes almacenar una estructura de datos en un objeto localStorage.

Si quiere recorrer todo un objeto localStorage y visualizar todas las claves lo podemos hacer de las siguientes maneras:

```
for(a=0;a<localStorage.length;a++)
   alert("la clave " + localStorage.key(a) + " tiene el valor " +
          localStorage[localStorage.key(a)]);
```

otra forma

```
for(a in localStorage)
  document.write("la clave " +a+"tiene el valor "+localStorage[a]+"<br />");
```

Veamos ahora un ejemplo

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>storage05</title>

<script>
  window.addEventListener("load",function(){
    alert("terminado");
     document.getElementById("boton").addEventListener("click",function(){
         alert(localStorage.length);
         for(a=0;a<localStorage.length;a++)
                alert((a+1)+"/"+localStorage.length+ " " +
                localStorage.key(a) +" =  " +
                localStorage[localStorage.key(a)]);
     });
      document.getElementById("boton1").addEventListener("click",function(){
         for(a in localStorage)
```

```
                alert(a+ " = " +localStorage[a]);
      });
      document.getElementById("cargar").addEventListener("click",function(){
         localStorage.nia_0001="Roberto Roncero";
         localStorage["nia_0002"]="Sergio Cancedo";
         localStorage.setItem("nia_0003","Rosario Lopez");
      });
  });
</script>
</head>
<body>
      <div>localStorage y sessionStorage</div>
      <button id="boton">Recorrer todo 1</button>
      <button id="boton1">Recorrer todo 2</button>
      <button id="cargar">Cargar datos</button>
</body>
</html>
```

Comprobar que nos hemos excedido en tamaño

Para realizar esta comprobación es conveniente trabajar con try...catch.

```
  try{
      for(a=0;a<9000;a++){
          localStorage.setItem("c"+a,"per la carretera de relleu pasa un carro
carregat de torrat y darrere va un burro en el rabo tallat");
      }
   } catch (e){
      if(e.code==22) alert("te has equivocado");
      if(e.name=="QuotaExceededError") alert("también de esta forma");
      if(e.code==e.QUOTA_EXCEEDED_ERR) alert("y de esta tercera forma");
   }
   alert("final del todo en el número " + a);
```

Esto funciona en Chrome y Firefox (donde se han probado). En algunas páginas de internet indican que los códigos cambian según el navegador. Hay que comprobar en los diferentes navegadores y versiones.

Con el siguiente código pordemos ver cuantas entradas podemos crear con la frase que guardamos.

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>storage06</title>

<script>
  window.onload(function(){

   try{
      for(a=0;a<100000;a++){
          localStorage.setItem("c"+a,"per la carretera de relleu pasa un carro
carregat de torrat y darrere va un burro en el rabo tallat");
      }
   } catch (e){
      if(e.code==22) alert("te has equivocado");
      if(e.name=="QuotaExceededError") alert("también de esta forma");
      if(e.code==e.QUOTA_EXCEEDED_ERR) alert("y de esta tercera forma");
   }
```

```
    alert("final del todo en el número " + a);

  });
</script>
</head>
<body>
      <div>LocalStorage y SessionStorage</div>
</body>
</html>
```

Seguimiento de eventos.

Cuando se modifica un valor en el storage en otra pestaña se produce un evento que puede seguirse con los métodos habituales de escucha de eventos de JavaScript.

En nuestro caso concreto, que se trata un evento seguido del almacenamiento local HTML5, se puede obtener la información devuelta en el parámetro ***Storage Event Object***.

El evento que tenemos que controlar es “storage”. Es un evento que es recogido por el objeto window del navegador. Es un evento que se lanza cuando se cambia el contenido de un valor del storage desde otra ventana. (en la misma máquina)

En esencia, un código de captura de evento del Web Storage debería ser algo así como el siguiente código

```
<script>
   window.addEventListener("storage",function(){
      alert("cambiado");
   });
</script>
```

Para probar el funcionamiento hay que hacerlo desde un servidor web.

Storage Event Object

Este objeto que es transferido a la función de respuesta tiene numerosos métodos y propiedades. Aquí vamos a ver las propiedades que nos interesan para este capítulo.

Propiedad	**Descripción**
key	La clave que ha sido cambiada y es objeto del evento.
newValue	El nuevo valor.
oldValue	El antiguo valor.
url	La dirección desde la cual se ha emitido el evento de cambio.

En donde:

- evt: sería el parámetro pasado a la función de respuesta.
- 'es-ES':determina la localización de la fecha y hora a mostrar.

Veamos un ejemplo

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>storage08</title>

<script>
  window.addEventListener("load",function(){
     document.getElementById("modificar").addEventListener("click",function(){
        localStorage.nia_0003=Math.round(Math.random()*10);
     });
     document.getElementById("cargar").addEventListener("click",function(){
        localStorage.nia_0001="Roberto Roncero";
        localStorage["nia_0002"]="Sergio Cancedo";
        localStorage.setItem("nia_0003","Rosario Lopez");
     });
     window.addEventListener("storage",function(evt){
       alert("key " + evt.key);
       alert("newValue " + evt.newValue);
       alert("oldValue " + evt.oldValue);
       alert("url " + evt.url);
     });

  });
</script>
</head>
<body>
      <div>localStorage y sessionStorage</div>
      <button id="cargar">Cargar datos</button>
      <button id="modificar">Modificar</button>
</body>
</html>
```