COOKIES

- Guardan datos en un archivo mediante pares de nombre-valor.
- Asociadas a un único dominio: sólo serán enviadas aquellas que tengan relación sobre el dominio al que se realizan las peticiones.
- Si no se le pone un tiempo de expiración, expira cuando se cierra el navegador.

Cookies Javascript:

```
document.cookie=”nombreEmpleado=Quique”;

todasLasCookies = document.cookie;
document.cookie="username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC"; //para borrar una cookie, establecer el atributo expires a una fecha ya pasada
```

Cookies JQuery:

https://github.com/js-cookie/js-cookie

- Create a cookie, valid across the entire site:

```
Cookies.set('name', 'value');
```

- Create a cookie that expires 7 days from now, valid across the entire site:

```
Cookies.set('name', 'value', { expires: 7 });
```

Create an expiring cookie, valid to the path of the current page:

```
Cookies.set('name', 'value', { expires: 7, path: '' });
```

- Read cookie:

```
Cookies.get('name'); // => 'value'
```

- Read all visible cookies:

```
Cookies.get(); // => { name: 'value' }
```

- Delete cookie:

```
Cookies.remove('name');
```

**Para probarlo, mejor en Firefox, es posible que en Chrome no estén habilitadas las cookies.

WebStorage

localStorage y sessionStorage, como sus nombres indican, se tratan de un espacio de almacenamiento local. La diferencia entre ellos es que uno almacena la información hasta que se termina la sesión en el navegador (se cierra el navegador) y el otro almacena la información permanentemente (se le puede dar un tiempo de expiración) en local.

localStorage vs Cookies

La mejor forma de entender por qué es necesario el **localStorage** es indicando los tres grandes problemas de las cookies:

1. Espacio limitado: Una cookie sólo puede ocupar 4kb de espacio. Es por eso que las cookies suelen utilizarse sólo para almacenar un hash o un identificador que será utilizado por el servidor para identificar la visita.
2. Cada vez que se realiza una petición al servidor, toda la información que está almacenada en las cookies es enviada y también es recibida nuevamente con la respuesta del servidor. O sea, en los intercambios de información entre el navegador web y el servidor siempre van pegadas las cookies.
3. Las cookies tienen una caducidad.

En cuanto a **localStorage**

1. Espacio menos limitado: **localStorage puede ocupar entre 5 y 10MB** dependiendo del navegador web.
2. La información almacenada con localStorage **no es enviada al servidor en cada petición**.
3. **No existe una caducidad para localStorage**, la información quedará almacenada hasta que se elimine expresamente. Aunque se cierre el navegador.

A veces lo que interesa es que la información se elimina una vez se cierre el navegador. Para estos casos, en vez de utilizar localStorage, se debe usar **sessionStorage**.

El sessionStorage es exactamente igual que localStorage, pero con la salvedad de que una vez cerrado el navegador se pierde la información, todo lo demás es lo mismo.

https://msdn.microsoft.com/es-es/library/dn194486.aspx

- Desde la perspectiva del código asociado a la página Web, *Local Storage* y *Session Storage* se comportan de la misma manera. Sólo cambia la disponibilidad temporal de la información.

- Tanto el *Local Storage* como el *Session Storage* están asociados a un dominio de Internet, no a una página concreta. Por ejemplo, si se usó para www.dominio.com/carpeta/hoja1.html, también estará disponible para www.dominio.com/default.html, ya que pertenecen al mismo dominio.

- Dado que está asociado a un determinado ordenador y usuario, si se accede desde otro ordenador o desde el mismo ordenador pero como diferente usuario, los anteriores datos de *Local Storage* no estarán disponibles.

- Los datos se almacenan siempre como texto, así pues han de utilizarse funciones de conversión para acceder a datos de otro tipo –números, fechas, …pero se puede convertir en un objeto JSON para luego trabajar con él.

- Los datos se almacenan en formato de parejas de *nombre/valor*, en dónde el nombre es de la variable almacenada y el valor es el contenido textual que se guarda.

Almacenar un valor localStorage

Para guardar la información utilizamos el método *setItem*:

```
localStorage.setItem('key', 'value');
```

De esta forma, en nuestro espacio de almacenamiento local, tendremos un elemento llamado *key* y que tendrá por valor *value*.

Recuperar un valor localStorage

Para recuperar la información utilizamos el método *getItem*:

```
var value = localStorage.getItem('key');
```

Este código Javascript almacena en la variable *value* el contenido almacenado para *key*.

Cómo saber el número de elementos almacenados en localStorage

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

Al ejecutar este código, nos aparecería una alerta con el número de elementos almacenados en nuestro espacio local.

Borrar un elemento localStorage

Para ello utilizamos el método *removeItem*

```
localStorage.removeItem('key');
```

Borrar todos los elementos localStorage

```
localStorage.clear();
```

Limitaciones del localStorage

Más que limitaciones, deberíamos hablar de la gran limitación: Sólo podemos almacenar cadenas de texto. O sea, no podemos guardar booleanos (true o false), no podemos guardar arrays, objetos, floats…. sólo strings.

Estamos ante una gran limitación… pero podemos superarla con **JSON**.

Gracias a **JSON** podremos convertir un objeto (o lo que esa) en cadena de texto y almacenarlo en nuestro **localStorage**. Al mismo tiempo, con **JSON** podremos transformar la cadena recuperada de **localStorage** al objeto inicial

```
var object = { 'uno' : '1', 'dos' : '2' };
// Lo guardamos en localStorage pasandolo a cadena con JSON.
localStorage.setItem('key', JSON.stringify(object));
// Creamos una nueva variable object2 con el valor obtenido de localStorage usando JSON recuperar el objeto inicial.
var object2 = JSON.parse(localStorage.getItem('key'));
// La alerta mostrará 1 por pantalla.
alert(object2.uno);
```

En Chrome podemos ver los valores guardados desde Más herramientas → Herramientas para desarroladores

WebSQL

Está deprecado, Firefox no lo admite.

Un ejemplo:

https://jsfiddle.net/Trae/76srLbwr/

**En Firefox no funciona

IndexDB

Se escapa al contenido del curso, aquí está explicado bastante bien paso a paso como se maneja:

https://rolandocaldas.com/html5/indexeddb-tu-base-de-datos-local-en-html5