# Web storage

## TEMA 9.2

## LOCALSTORAGE Y SESSIONSTORAGE EN HTML5

# Características tanto para localStorage como para sessionStorage son:

- La capacidad es de 5Mb mientras que la de las cookies era de tan solo 4Kb.
- La información no es incluida en cada petición al servidor como sucede con las cookies, sino que solo se obtiene cuando se quiere usar.
- La información es almacenada en pares clave-valor por lo que se puede usar como si se tratase de variables.
- Una página web únicamente puede acceder a la información que ha almacenado ella por lo que es más seguro.

# Diferencias

La diferencia entre localStorage y sessionStorage es que la información almacenada con localStorage se guarda indefinidamente hasta que sea eliminada mediante código o bien borrada desde el navegador y con sessionStorage la información se guarda hasta que se cierra el navegador. Por lo tanto como la única diferencia entre am.bas es el tiempo

# Almacenamiento y recuperación de datos

- **sessionStorage y localStorage, almacenan datos como ítems.**
- **Los ítems están formados por un par** clave/valor, y cada valor será convertido en una cadena de texto antes de ser almacenado.
- Los ítems son como si fueran variables, con un nombre y un valor, que pueden ser creadas, modificadas o eliminadas.

# Métodos para crear y leer

- Existen dos nuevos métodos específicos de esta API incluidos para crear y leer un valor en el espacio de almacenamiento:
- **setItem(clave, valor)**
- **Este es el método que tenemos que llamar para crear un ítem.**
- El ítem será creado con una clave y un valor de acuerdo a los atributos especificados. Si ya existe un ítem con la misma clave, será actualizado al nuevo valor, por lo que este método puede utilizarse también para modificar datos previos.
- **getItem(clave)**
- **Para obtener el valor de un ítem, debemos llamar a este método especificando la clave del ítem** que queremos leer. La clave en este caso es la misma que declaramos cuando creamos al ítem con **setItem().**

# Sintaxis abreviada

Sintaxis abreviada para crear y leer ítems desde el espacio de almacenamiento.

Podemos usar la clave del ítem como una propiedad y acceder a su valor de esta manera.

Este método usa en realidad dos tipos de sintaxis diferentes de acuerdo al tipo de información que estamos usando para crear el ítem.

Podemos encerrar una variable representando la clave entre corchetes (por ejemplo,

- **sessionStorage[clave]=valor) o podemos usar directamente el nombre de la propiedad (por ejemplo,**
- **sessionStorage. imitem=valor).**

# Ejercicio 1

Vamos a ver un ejemplo de localStorage y de session en el que mediante un “formulario” solicitamos el nombre y e-mail y los almacenamos (botón guardar) .Después de haberlos almacenado una vez ,si pulsas en el botón Recuperar datos obtendrás los datos que habías puesto aunque hayas cerrado el navegador. Realizarlo con los dos objetos y analizar las diferencias.

# Storage en HTML5

---

**Nombre:**

**Email:**

Session Storage

Guardar | Obtener | Eliminar nombre

Local Storage

Guardar | Obtener | Eliminar nombre

# Leyendo datos

- **length**

**Esta propiedad retorna el número de ítems guardados por esta aplicación en el espacio de almacenamiento**. Trabaja exactamente como la propiedad length usada normalmente en Javascript para procesar arrays, y es útil para lecturas secuenciales.

- **key(índice)**

**Los ítems son almacenados secuencialmente, enumerados con un índice automático que comienza por 0.**

# Borrar datos almacenados

localStorage.removeItem(“clave")

sessionStorage.removeitem(“clave”)

**clear()---borra todos los items almacenados**

# Evento storage

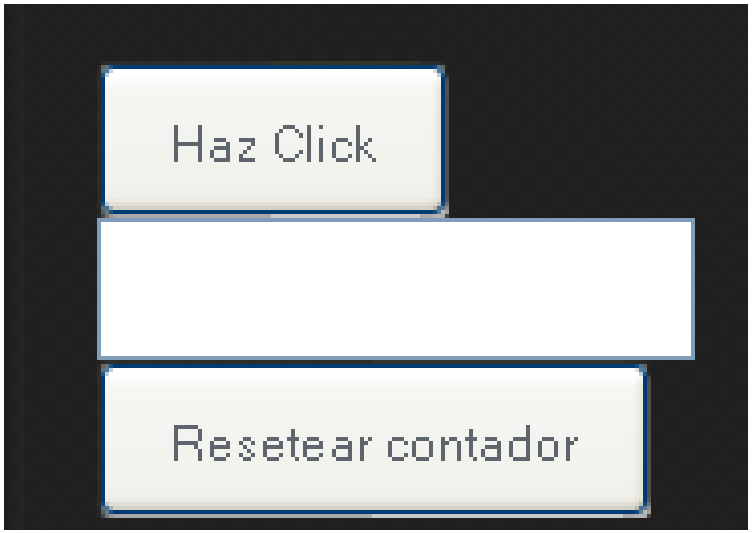
## storage

- Este evento será disparado por la **ventana** cada vez que un cambio ocurra en el espacio de almacenamiento.

- window.addEventListener('storage', funcionalmacen, false);

# Ejercicio

- Contador de clicks. Realizar este pequeño formulario y cuando se ejecute el evento storage enviar un alert. Se trata de ir contando los clicks que damos en el botón, se acumulan de una sesión a otra.
- También podremos resetear en una misma sesión, pero siempre se acumulan.



# Ejercicio cont.

- Añadiremos también un contador de visitas que mostraremos al cargar la página.
- Implementaremos también una opción para que el usuario elija su color de fondo de la página y al volver a entrar guarde esa configuración.
- Por último implementaremos también un botón que nos muestre todos los nombres y los valores almacenados que tenemos en esa página.

```
function compruebaCompatibilidad() {

if (window.sessionStorage && window.localStorage) {
alert('Tu navegador acepta almacenamiento local');
}
else {
 alert('Lo siento, pero tu navegador no acepta almace
namiento local'); }
 }
```

# Almacenando algo mas que strings

# json

- Una manera de almacenar objetos es utilizando JSON. Como la representación de los objetos en JSON puede realizarse a través de texto, podemos almacenar estas cadenas de texto y recuperarlas posteriormente para convertirlas en objetos.

# ejemplo

- Tenemos el siguiente objeto:

```
var contact = new Object();
contact.nombre= “Luisa";
contact.apellidos = “Garcia";
contact.tlf = [“942-344567", “942-897898"];
```

# Ejemplo de guardar y recuperar

- sessionStorage.setItem(‘contacto1', JSON.stringify(contact) ); lo guardamos como cadena

- JSON.parse(sessionStorage.getItem(contact) );
- Lo recuperamos como json