Web Storage. Introducción

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Ene
- iPasen - cjaedia071@g.educaand.es

Carmelo José Jaén Díaz

Índice

OBJETIVO

- Descubrir elementos propios de JavaScript como los eventos
- Registrar en el front-end información de navegación, usuario, etc., mediante cookies o mediante el almacenamiento local que ofrece el estándar HTML5.

GLOSARIO

Expresión regular. Secuencia de caracteres que forman un patrón determinado, generalmente un patrón de búsqueda.

Listener. Código responsable de controlar eventos. Están a la escucha (de ahí su nombre) y, cuando ocurre un evento, se ejecuta el código que el programador haya implementado.

INTRODUCCIÓN

Web Storage es una **API de almacenamiento web** que proporciona los mecanismos mediante los cuales el navegador puede almacenar información de tipo clave-valor de una forma mucho **más intuitiva que utilizando *cookies*.**

Cuando hablamos de **Web Storage** disponemos, a su vez, de dos mecanismos de almacenamiento web:

- **`sessionStorage`**: que permite almacenar información mientras el navegador está abierto; es decir, mientras dura la sesión de la página. Todas las páginas que tienen el mismo origen (dominio y protocolo) pueden acceder a estos datos indistintamente.
- **`localStorage`**: similar a `sessionStorage` pero los datos se mantienen aún a pesar de que cerremos el navegador.

INTRODUCCIÓN

A continuación, utilizaremos los diferentes **métodos** que nos permitirán trabajar con este tipo de almacenamiento:

- `setItem`: permite crear un elemento de almacenamiento web.
- `getItem`: permite consultar un elemento.
- `removeItem` y `clear`: permite eliminar información de un elemento o el elemento completo.

WEB STORAGE

Web Storage es implementado por HTML5 y mejora sustancialmente la persistencia de datos respecto al tradicional uso de *cookies*, entre otras funcionalidades permite:

- Permite almacenar datos localmente en el ordenador del usuario.
- Es más seguro y almacena más información que las *cookies*.
- El *web storage* se almacena por origen (dominio y protocolo): no se limita a una página en concreto, todas las páginas del mismo origen pueden acceder a los mismos datos.

WEB STORAGE
Soporte por el navegador

Para comprobar si el navegador soporta el uso de *Web Storage*:

```
if (typeof(Storage) !== "undefined"){
    alert ("El navegador soporta webStorage");
    //Escribir código asociado a web storage aquí
}
else{
    alert("El navegador NO soporta WebStorage");
}
```

window.localStorage
Crear un elemento

`window.localStorage`: almacena datos SIN fecha de expiración, es decir se mantienen hasta que el usuario decide borrarlos. ***Nota:*** *Debido a que el objeto `window` es el objeto padre del resto de objetos y métodos, es posible omitir su declaración.*

Para crear un elemento escribiríamos las siguientes sentencias. ***Recuerda:*** *Estos elementos por defecto son `string`, por lo que si deseamos obtener números tenemos que "parsearlos". Si deseamos obtener `arrays` u objetos, tenemos que pasarlos a JSON.*

```
//Al método setItem, le pasamos como argumento un nombre y un valor
localStorage.setItem("docente", "Carmelo");
//Otra forma de implementación. NO RECOMENDABLE
localStorage.docente = "Carmelo";
```

window.localStorage
Consultar un elemento

```
//Al método getItem, le pasamos como argumento el nombre del elemento cuyo
//valor queremos obtener
localStorage.getItem("docente");

//Otra forma de implementación. NO RECOMENDABLE
localStorage.docente;
```

Para ver su funcionalidad, podríamos mostrarlos mediante un `alert()`, por ejemplo:

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

window.localStorage
Borrar un elemento

```
//Al método removeItem, le pasamos como argumento el nombre del elemento a
//borrar
localStorage.removeItem("docente");

//Al consultar el valor del elemento borrado, devolvería null
alert(localStorage.getItem("docente"));

//Si deseamos borrar TODOS los elementos, empleamos el método clear
localStorage.clear();
```

window.sessionStorage

`window.sessionStorage`: almacena los datos durante una sesión, es decir se mantienen hasta que se cierra la ventana o el navegador, momento en que desaparecen.

`sessionStorage` tiene los mismos métodos y la API se usa de la misma forma que `localStorage`.

window.sessionStorage
LocalStorage vs SessionStorage

Característica	LocalStorage	SessionStorage
Acceso	En todas las pestañas del mismo dominio.	Sólo en la pestaña que se creó.
Duración	Hasta que se eliminen manualmente.	Hasta que se cierra la pestaña.
Capacidad	~10MB en Chrome, aunque puede variar por navegador.	Similar a localStorage.
Modo incógnito	Los datos se borran al cerrar la pestaña.	Igual a LocalStorage.

WEB STORAGE
Ventajas frente al uso de cookies

Como curiosidad, si comparamos el `localStorage`/`sessionStorage` con las clásicas *cookies*, suelen ser un buen reemplazo, ya que evitan el clásico problema de enviar la *cookie* en cada petición web, mientras que la API de *Web Storage* sólo descarga los datos cuando lo solicitas desde Javascript.