Programador JavaScript Avanzado

Unidad 8: WebStorage

Contactos: tecnologia@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 8: WebStorage

Objetivos

Que el alumno logre:

- Almacenar datos de forma local con webStorage

WebStorage

Web Storage es una API en JavaScript que consta de dos mecanismos de almacenamiento: localStorage y sessionStorage.

localStorage: Almacena datos sin límite de expiración. Los datos persisten incluso después de cerrar el navegador y se mantienen hasta que se borran explícitamente o el usuario borra el caché del navegador.

sessionStorage: Almacena datos para una sesión específica. Los datos se mantienen solo durante la duración de la sesión del navegador. Se borran cuando se cierra la pestaña o el navegador.

Estos mecanismos están disponibles mediante las propiedades **window.sessionStorage** y **window.localStorage** (dicho con más precisión, en navegadores con soporte, el objeto Window implementa los objetos WindowLocalStorage y WindowSessionStorage, en los cuales se basan las propiedades localStorage y sessionStorage).

Al invocar uno de éstos, se creará una instancia del objeto Storage, a través del cual los datos pueden ser creados, recuperados y eliminados. sessionStorage y localStorage utilizan un objeto de almacenamiento diferente según su origen — funcionan y son controlados por separado.

```
// localStorage
localStorage.setItem('nombre', 'Juan'); // Guarda un dato en localStorage
const nombre = localStorage.getItem('nombre'); // Obtiene un dato de localStorage
localStorage.removeItem('nombre'); // Elimina un dato de localStorage
localStorage.clear(); // Limpia todo el contenido de localStorage

// sessionStorage
sessionStorage.setItem('color', 'azul'); // Guarda un dato en sessionStorage
const color = sessionStorage.getItem('color'); // Obtiene un dato de sessionStorage
sessionStorage.removeItem('color'); // Elimina un dato de sessionStorage
sessionStorage.clear(); // Limpia todo el contenido de sessionStorage
```

Tanto localstorage como sessionstorage comparten métodos similares (setItem, getItem, removeItem, clear) para administrar los datos almacenados. La principal diferencia radica en la persistencia de los datos.

Es importante tener en cuenta que estos métodos solo almacenan datos como cadenas de texto. Si deseas guardar objetos JavaScript, debes convertirlos a cadenas de texto usando JSON.stringify() al guardarlos y JSON.parse() al recuperarlos, tal como se mencionó en el ejemplo de localStorage anteriormente.

Navegación Privada / Modo incógnito

La mayoría de los navegadores de hoy en día soportan una opción de privacidad llamada **'Modo incógnito**', **'Navegación privada'**, o algo similar, que básicamente se asegura de que la sesión de navegación no deje rastros después de que el navegador se cierra. **Esto es fundamentalmente incompatible con el almacenamiento web** por obvias razones. Por ello, muchos navegadores están experimentando con diferentes escenarios para lidiar con esta incompatibilidad.

La mayoría de los navegadores han optado por una estrategia donde las API de almacenamiento siguen disponibles y aparentemente completamente funcionales, con la única gran diferencia de que todos los datos almacenados son eliminados después de cerrar el navegador.

Para estos navegadores aún hay diferentes interpretaciones sobre qué debería hacerse con los datos almacenados existentes (de una sesión de navegación normal). ¿Deberían de estar disponibles para lectura cuando esté en modo privado? Entonces, hay algunos navegadores, sobre todo Safari, que han optado por una solución donde el almacenamiento está disponible, pero vacío, y tiene un cupo de 0 bytes asignado, por lo que se vuelve imposible usar esta memoria para escribir datos.

Es importante tener en cuenta estas diferentes implementaciones a la hora de desarrollar aplicaciones web que depende de la API de almacenamiento web.

Resumen

En esta Unidad…

Trabajamos con WebSstorage