2025

UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

PROGRAMACIÓN WEB

GÍIA DE USO Y APLICACIÓN

LocalStorage Y Cookies

CUARTO SEMESTRE
NRC: 2260

CONTENIDO

GUIA DE USO Y APLICACIÓN

Local Storage y las cookies son dos mecanismos clave para el almacenamiento de datos en el navegador. Mientras que Local Storage permite guardar información de manera persistente en el dispositivo del usuario sin enviarla con cada solicitud HTTP, las cookies se utilizan principalmente para la gestión de sesiones y la comunicación con el servidor. Cada tecnología tiene sus ventajas y usos específicos, por lo que comprender sus diferencias y aplicaciones es fundamental para el desarrollo web.

1. LocalStorage (Almacenamiento Local)

1.1. Introducción

El *localStorage* es una funcionalidad de los navegadores que permite a los desarrolladores guardar información en el navegador del usuario. Es parte de Web Storage API [API de Almacenamiento Web] junto a *session storage*. Funciona tomando información en formato de pares clave-valor y la conserva aun cuando el usuario actualiza la página o cierra la pestaña o el navegador.

La API de almacenamiento web es un conjunto de mecanismos que permite a los navegadores almacenar pares clave-valor. Está diseñada para ser mucho más intuitiva que el uso de cookies. El objeto de almacenamiento local ofrece distintos métodos que puede utilizar para interactuar con él. Con estos métodos, puede agregar, leer y eliminar datos del almacenamiento local.

1.2. localStorage en JavaScript

Es una propiedad que permite que los sitios y las aplicaciones de JavaScript guarden pares clave-valor en un navegador web sin fecha de vencimiento. Esto significa que los datos almacenados persisten incluso después de que el usuario cierra el navegador o reinicia la computadora.

localStorage es una *window* propiedad de objeto, lo que la convierte en un objeto global que puede interactuar con la ventana del navegador y manipularla. También se puede utilizar en combinación con otras window propiedades y métodos.

localStorage también permite a sus clientes acceder a datos específicos rápidamente sin la sobrecarga de una base de datos. Existen otras ventajas y desventajas que se deben comprender al trabajar con *localStorage*.

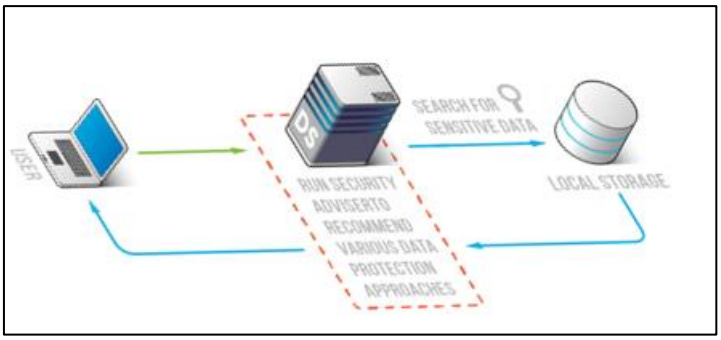

Imagen 1 *Ciclo de vida del TDA*

1.3. *Window.localStorage*, ¿Qué es?

El mecanismo *localStorage* está disponible a través de la propiedad *Window.localStorage*. que es parte de la interfaz en JavaScript y representa una ventana que contiene un documento DOM.

La interfaz *Window* presenta una amplia gama de funciones, constructores, objetos y espacios de nombres. *Window.localStorage* es una propiedad de solo lectura que devuelve una referencia al objeto de almacenamiento local utilizado para almacenar datos a los que solo puede acceder el origen que los creó.

.

1.4. ¿Cuándo usar *localStorage?*

Se utiliza para almacenar y recuperar datos. Si bien puede almacenar pequeñas cantidades de datos con *localStorage*, no es adecuado para grandes conjuntos de datos.

localStorage es accesible para cualquier persona que use el dispositivo, por lo que no debe usarlo para almacenar información confidencial. Puede usarlo para almacenar preferencias del usuario, como idioma o tema. También puede usarlo para almacenar datos en caché si lo usa con frecuencia. *localStorage* puede almacenar datos de formularios que no se perderán si el usuario cierra el navegador. Si tiene una aplicación que requiere que inicie sesión, *localStorage* puede usarla para guardar los datos de su sesión.

1.5. Métodos de LocalStorage

Como se había mencionado antes *localStorage* almacena datos. Y, si lo haces, significa que es posible que necesites recuperarlos más adelante. En esta sección, exploraremos exactamente cómo localStoragefunciona

1.5.1. Almacenamiento de datos, localStorage.setItem()

El método *setItem()* permite almacenar valores en *localStorage*. Toma dos parámetros: una clave y un valor. Se puede hacer referencia a la clave más adelante para obtener el valor asociado a ella. Así es como debería verse.

```
localStorage.setItem('name', 'Obaseki Nosa');
```

En el código anterior, puedes ver que *name* es la clave y *Obaseki Nosa* es el valor. Como ya hemos señalado, *localStorage* solo se pueden almacenar cadenas. Para almacenar matrices u objetos en *localStorage*, tendrías que convertirlos en cadenas.

Para eso usamos JSON.stringify() pasando el método before a setItem(), así:

```
const userArray = ["Obaseki",25]
localStorage.setItem('user', JSON.stringify(userArray));
```

1.5.2. Recuperando datos, localStorage.getItem()

El método *setItem()* permite acceder a los datos almacenados en el *localStorage* del navegador . Este método acepta solo un parámetro, el *key*, y devuelve el *value* como una cadena:

```
localStorage.getItem('name');
```

Esto devuelve una cadena con un valor de "Obaseki Nosa". Si la clave especificada no existe en *localStorage*, devolverá *null*. En el caso de la matriz, utilizamos el método *JSON.parse()*, que convierte una cadena JSON en un objeto JavaScript:

```
JSON.parse(localStorage.getItem('name'));
```

Usando la matriz que creamos arriba, aquí se explica cómo recuperarla con localStorage:

```
const userData = JSON.parse(localStorage.getItem('usuario')); console.log(userData);
```

Este método devolverá la matriz . Puedes inspeccionar la página web y buscarla en la consola, de esta manera:["Obaseki", 25]

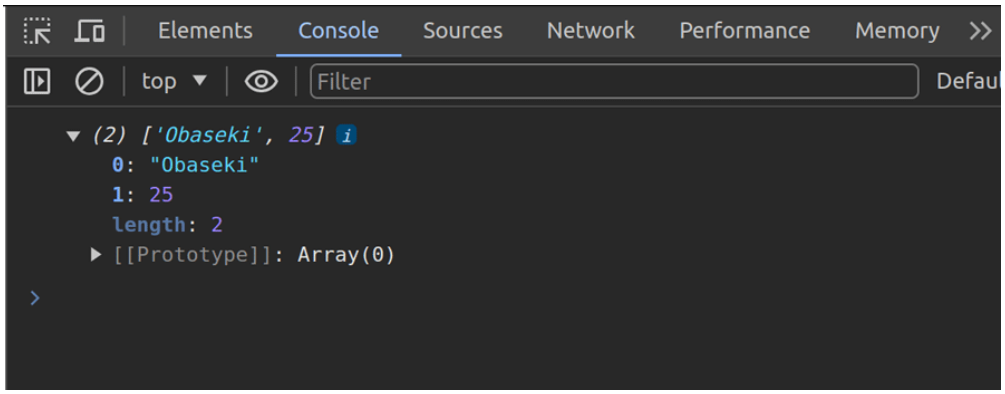

Ilustración 1 *Captura en navegador Firefox*

1.5.3. Eliminar datos, localStorage.removeItem()

Para eliminar un elemento de *localStorage*, deberá utilizar el método *removeItem()*. Al pasar una clave *name*, el método *removeItem()* elimina la clave existente del almacenamiento. Si no hay ningún elemento asociado con la clave dada, este método no hará nada. Este es el código.

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

1.5.4. Eliminar todos los elementos, localStorage.clear ()

Para eliminar todos los elementos de *localStorage*, deberá utilizar el método *clear()*. Cuando se invoca este método, se borra todo el almacenamiento de todos los registros de ese dominio. No recibe ningún parámetro. Se utiliza el siguiente código:

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

1.6. Ejemplos de Uso localStorage

1.6.1. localStorage en un formulario

Véase en: https://github.com/JuDav-093/Local-Storage-y-Cookies/tree/main/Local%20Storage/1localStorage_en_Formulario

1.6.2. Eliminar todos los registros

Véase en: https://github.com/JuDav-093/Local-Storage-y-Cookies/tree/main/Local%20Storage/2Eliminar_todos_los_Resgistros

2. COOKIES

2.1. Introducción

Las cookies son un elemento omnipresente en nuestra navegación por internet. Cada vez que visitamos una página web, es común encontrar un aviso que nos informa sobre el uso de cookies y nos solicita consentimiento para su almacenamiento en nuestros dispositivos.

2.2. ¿Qué es una Cookie?

Las cookies son pequeños archivos de texto que se guardan en el dispositivo de un usuario cuando visita un sitio web. Estos archivos contienen datos que el sitio puede leer durante la visita del usuario o en visitas posteriores. Esto permite a los sitios web recordar acciones y preferencias del usuario, lo que resulta esencial para proporcionar una experiencia de navegación personalizada y eficiente.

Ilustración 2 Representación gráfica de una cooki

Es decir, aunque leyendo ciertos medios parezca que las cookies son "*El Mal* ", en realidad fueron creadas para facilitar la navegación al usuario; por ejemplo, permiten recordar en qué idioma queremos navegar en una web o si nos hemos identificado con nuestro usuario y contraseña, para no tener que repetir estas acciones una y otra vez. Otro tema es que el uso de cookies de terceros haya llegado, en algunos casos, a límites que rozan el espionaje del usuario.

2.3. Cookies según su plazo de tiempo

2.3.1. Cookies de sesión

Son cookies temporales que se utilizan para recordar la actividad del usuario durante la duración de su visita a la web. Estas se borran automáticamente cuando el usuario cierra el navegador. Son útiles para funciones como mantener una sesión de usuario activa y asegurar que las operaciones realizadas se recuerden durante la navegación en una sesión.
Se puede implementar de la siguiente manera:

```
document.cookie = "sesionID=12345; path=/";
```

Esta cookie almacena un identificador de sesión (sesionID=12345). La opción **path=/** indica que la cookie estará disponible en todas las rutas del dominio.

Estas cookies de sesión se utilizan generalmente en un sitio de banca en línea para recordar que el usuario ha iniciado sesión. Cuando el usuario cierra el navegador, la sesión termina y la cookie se elimina automáticamente, lo que ayuda a proteger la seguridad de la cuenta.

2.3.2. Cookies persistentes

A diferencia de las cookies de sesión, las cookies persistentes permanecen en el dispositivo del usuario durante un periodo específico o hasta que sean eliminadas manualmente. Estas cookies son utilizadas para recordar las preferencias del usuario y diversas configuraciones entre sesiones, facilitando el acceso y la personalización del sitio en futuras visitas.

Se puede implementar de la siguiente manera:

```
let fechaExpiracion = new Date();
fechaExpiracion.setTime(fechaExpiracion.getTime() + (7 * 24 * 60 * 60 * 1000));
document.cookie = "usuario=Anabel; expires=" + fechaExpiracion.toUTCString() + "; path=/";
```

Se crea una cookie llamada *usuario* con el valor Anabel. La fecha de expiración se establece a 7 días en el futuro, asegurando que persista más allá de la sesión actual.

Las cookies persistentes podrían ser utilizadas en un ecommerce para recordar el carrito de compras del usuario o sus preferencias de idioma y región, incluso si cierra el navegador y vuelve días después.

2.3.3. Cookies seguras

Las cookies seguras tienen ciertos atributos o indicadores que mejoran su seguridad, como el indicador "Seguro" que indica que la cookie solo debe enviarse a través de conexiones HTTPS, no HTTP, se usan para proteger información sensible, como credenciales de usuario.

Se puede implementar de la siguiente manera:

```
document.cookie = "autenticationToken=abc123; Secure; path=/*;
```

Se define una cookie autenticacionToken con el valor abc123. La opción Secure garantiza que solo se envíe en conexiones HTTPS.

2.4. Cookies según su finalidad

2.4.1. Cookies Tecnicas u Obligatorias

Estas cookies son esenciales para el funcionamiento básico del sitio web. Permiten a los usuarios navegar por el sitio y usar sus funciones fundamentales, como acceder a áreas privadas.

Para saber si el usuario tiene activa la sesión se puede implementar de la siguiente manera:

```
document.cookie = "autenticationToken=abc123; Secure; path=/*;
```

Un ejemplo puede ser una cookie que mantiene a un usuario conectado a su cuenta mientras navega de una página a otra, asegurando que no tenga que

iniciar sesión en cada página nueva.

2.4.2. Cookies de preferencia o personalización

Permiten que una web recuerde las elecciones que hace un usuario —como su nombre de usuario, idioma o la región en la que se encuentra— con el objetivo de proporcionar funcionalidades mejoradas y personalizar la experiencia de los usuarios.
Para almacenar la preferencia del usuario sobre el tema del sitio puedes hacerlo de la siguiente manera:

```
document.cookie = "tema=oscuro; path=/";
```

Un ejemplo puede ser una cookie que recuerda el idioma seleccionado por el usuario para que el sitio web se muestre automáticamente en ese idioma en futuras visitas.

2.4.3. Cookies de analítica o estadística

Recogen datos sobre el uso del sitio para mejorar su rendimiento, como información sobre cómo navegan los usuarios, las páginas más visitadas o si reciben mensajes de error.
La siguiente implementación contabiliza las veces que el usuario ha visitado el sitio:

```
document.cookie = "visitas=5; path=/";
```

En esta clasificación entrarían las cookies utilizadas por servicios de analítica web, como Google Analytics.

2.4.4. Cookies de publicidad o marketing

Estas cookies recogen información sobre los hábitos de navegación del usuario, con el fin de hacer la publicidad más relevante para el usuario y adaptada a sus intereses. Suelen colocarlas redes publicitarias con el permiso del propietario del sitio web.
La siguiente cookie rastrear los anuncios mostrados al usuario:

```
document.cookie = "ads_tracking=enabled; path=/";
```

Esta cookie rastrea qué productos ha visto el usuario en una tienda online para mostrar anuncios de productos similares en otros sitios web.

2.5. Cookies según el propietario:

2.5.1. Cookies propias

Son aquellas que coloca directamente el sitio web que el usuario está visitando. Estas cookies suelen ser esenciales para el funcionamiento de la web y son más fáciles de gestionar en términos de cumplimiento normativo, ya que la información recopilada es mantenida y controlada por el propietario del sitio.

```
document.cookie = "ads_tracking=enabled; path=/";
```

Es decir El sitio web almacena la información del usuario, cuando un usuario visita un ecommerce y el site utiliza una cookie propia para mantener la sesión activa mientras el usuario añade artículos a su carrito de compra.

2.5.2. Cookies de tercero

Son cookies que coloca un dominio diferente al del sitio web que el usuario está visitando. Estas cookies son utilizadas principalmente para rastrear al usuario a través de varios sitios, ya sea con fines publicitarios, de investigación de mercados o de comportamiento en línea.
A continuación, te presento una cookie de rastreo establecida por un servicio de publicidad.

```
document.cookie = "trackingID=xyz123; path=/”; domain=publicidad.com";
```

Por ejemplo, si un usuario visita un blog y este blog tiene un widget de comentarios de una plataforma de redes sociales, la plataforma puede colocar una cookie de terceros para rastrear la actividad del usuario en la web y mostrar anuncios personalizados.

Este tipo de cookies están en el centro del debate sobre la privacidad y desde hace unos años se han ido restringiendo en muchos navegadores, como Safari de Apple o Mozilla Firefox. También está previsto que Google Chrome deje de utilizarlas en 2025.

2.6. Métodos de Manipulación de Cookies

2.6.1. Crear Cookie

Este método permite establecer una nueva cookie en el navegador. Por ejemplo,

crearemos una cookie llamada “usuario” con el valor "Anabel", válida hasta la fecha especificada.

```
document.cookie = "usuario=Anabel; path=/; expires=Fri, 31 Dec 2025 23:59:59 GMT";
```

2.6.2. Leer Cookie

Este método permite obtener el valor de una cookie almacenada. La función que presentamos a continuación recorre las cookies disponibles y devuelve el valor de la cookie solicitada.

```
function obtenerCookie(nombre) {
    let cookies = document.cookie.split('; ');
    for (let i = 0; i < cookies.length; i++) {
        let [clave, valor] = cookies[i].split('=');
        if (clave === nombre) return valor;
    }
    return null;
}
console.log(obtenerCookie("usuario"));
```

2.6.3. Eliminar Cookie

Con este método se puede eliminar una cookie estableciendo su fecha de expiración en el pasado. Para esto la cookie "usuario" se elimina al establecer una fecha de expiración pasada.

```
document.cookie = "usuario=; path=/; expires=Thu, 01 Jan 1970 00:00:00 GMT";
```

2.7. Ejemplos de Aplicación de Cookies

2.7.1. Ejemplos Cookies según propietario

Véase en: https://github.com/JuDav-093/Local-Storage-y-Cookies/tree/main/Cookies/Cookies%20segun%20el%20propietario

2.7.2. Ejemplos Cookies según el tiempo

Véase en: https://github.com/JuDav-093/Local-Storage-y-Cookies/tree/main/Cookies/Cookies%20segun%20el%20tiempo

2.7.3. Ejemplos Cookies según su finalidad

Véase en: https://github.com/JuDav-093/Local-Storage-y-Cookies/tree/main/Cookies/Cookies%20segun%20su%20finalidad

2.7.4. Ejemplos Métodos de manipulación de Cookies

Véase en: https://github.com/JuDav-093/Local-Storage-y-Cookies/tree/main/Cookies/M%C3%A9todos%20de%20manipulaci%C3%B3n%20de%20Cookies