APRENDERAPROGRAMAR.COM

SESSIONSTORAGE Y LOCALSTORAGE. DIFERENCIAS. GUARDAR DATOS EN CACHÉ Y PERSISTENCIA CON JAVASCRIPT (CU01198E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº98 del Tutorial básico “JavaScript desde cero”.

Autor: César Krall

UN MUNDO POR DELANTE CON JAVASCRIPT

En la actualidad la programación JavaScript constituye un área de especialización dentro de los desarrollos web. Las grandes empresas tienen programadores exclusivamente dedicados a JavaScript. En este curso hemos visto los fundamentos del lenguaje y de algunas apis útiles. Vamos a citar algunas apis que no hemos nombrado por el momento.

WEB STORAGE O ALMACENAMIENTO WEB JAVASCRIPT

Se han desarrollado funcionalidades que permiten el almacenamiento de información a través de JavaScript. Desde los momentos iniciales de internet, en que las webs se limitaban a mostrar información, hoy día cada vez se produce un mayor flujo de información entre usuarios y servidores y procesos donde intervienen datos que fluyen de un lado a otro. De la idea o forma de trabajo inicial en el que todo el trabajo de datos recaía en el servidor, se está pasando cada vez más a dar un mayor peso al manejo de datos del lado del cliente. Y ahí JavaScript adquiere un papel fundamental.

Inicialmente el almacenamiento de datos usando JavaScript era relativamente conflictivo: las variables desaparecían cuando se pasaba de una página web a otra. Hoy día existen herramientas como las que vamos a ver a continuación que facilitan el trabajo a los programadores permitiendo la persistencia de la información sin necesidad de almacenarla en el servidor.

Anteriormente en este curso hemos visto cómo se pueden manejar cookies con JavaScript. Las cookies presentan limitaciones por su naturaleza, de modo que se restringen al manejo de cadenas de texto sin integración en un sistema de datos bien estructurado.

En este caso vamos a referirnos a algunas tecnologías como Web Storage (no vamos a tratar aquí Indexed Database). Algunas de sus posibilidades todavía no están estandarizadas, pero otras están siendo ya usadas por la comunidad de programadores.

Con la api Web Storage se ha pretendido hacer una reformulación y potenciación del manejo de datos con algo más potente que las cookies. Con Web Storage se utiliza el ordenador del usuario (su memoria o su disco duro) para almacenar información útil para la navegación web. Esa información almacenada se recupera cuando es necesaria. Las posibilidades de almacenamiento se dividieron en dos partes, una destinada a almacenar información con tiempo de vida la duración de una sesión web (sessionStorage, por ejemplo para mantener los productos en un carrito de compra) y otra destinada a mantener los datos de forma permanente (localStorage, por ejemplo para mantener un informe de negocios), de forma similar a como haría una aplicación de escritorio.

La información almacenada queda relacionada con un sitio o aplicción web, de forma que no puede ser utilizada por un sitio o aplicación distinta a la que creó los datos.

Algunos navegadores pueden no admitir esta funcionalidad, en especial los más antiguos.

SESSION STORAGE

Escribe el siguiente código y guárdalo en un archivo html:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Web Storage API ejemplo aprenderaprogramar.com</title>
<script type="text/javascript">
function addProducto(){
var producto=document.getElementById('producto').value;
var precio=document.getElementById('precio').value;
sessionStorage.setItem(producto,precio); //ó sessionStorage[producto]=precio
mostrarDatos(producto);
}

function mostrarDatos(){
var datosDisponibles=document.getElementById('datosDisponibles');
datosDisponibles.innerHTML='';
for(var i=0;i<sessionStorage.length;i++){
		var producto=sessionStorage.key(i);
		var precio=sessionStorage.getItem(producto);
		datosDisponibles.innerHTML += '<div>'+producto+' - '+precio+'</div>';
		}
}

function limpiarVista() {
var datosDisponibles=document.getElementById('datosDisponibles');
datosDisponibles.innerHTML='Limpiada vista. Los datos permanecen.';
}

function borrarTodo() {sessionStorage.clear(); mostrarDatos(); }

</script>
</head>
<body>
<form name="formulario">
<p>Nombre del producto:<br><input type="text" name="producto" id="producto"></p>
<p>Precio:<br><input type="text" name="precio" id="precio"></p>
<p><input type="button" onclick="addProducto()" name="guardar" id="guardar" value="guardar"></p>
<p><input type="button" onclick="limpiarVista()" name="limpiar" id="limpiar" value="Limpiar Vista"></p>
<p><input type="button" onclick="borrarTodo()" name="borrar" id="borrar" value="Borrar todo"></p>
</form>
<br/>
<div id="datosDisponibles">No hay información almacenada</div>
</body>
```

El resultado esperado es que se nos permita ir añadiendo productos con su precio a la lista, limpiar la vista de productos y precios pero manteniendo almacenados los mismos, o bien borrarlos completamente pulsando el botón “Borrar todo”.

Resumen de métodos y propiedades de sessionStorage:

Método o propiedad de sessionStorage	Descripción aprenderaprogramar.com
sessionStorage.setItem('clave', 'valor');	Guarda la información valor a la que se podrá acceder invocando a clave. Por ejemplo clave puede ser nombre y valor puede ser Francisco.
sessionStorage.getItem('clave')	Recupera el value de la clave especificada. Por ejemplo si clave es nombre puede recuperar “Francisco”.
sessionStorage[clave]=valor	Igual que setItem
sessionStorage.length	Devuelve el número de items guardados por el objeto sessionStorage actual
sessionStorage.key(i)	Cada item se almacena con un índice que comienza por cero y se incrementa unitariamente por cada item añadido. Con esta sintaxis rescatamos la clave correspondiente al item con índice i.
sessionStorage.removeItem(clave)	Elimina un item almacenado en sessionStorage
sessionStorage.clear()	Elimina todos los items almacenados en sessionStorage, quedando vacío el espacio de almacenamiento.

PERSISTENCIA DE LOS DATOS CON SESSION STORAGE

Los datos con sessionStorage son accesibles mientras dura la sesión de navegación. Los datos no son recuperables si:

a) Se cierra el navegador y se vuelve a abrir.

b) Se abre una nueva pestaña de navegación independiente y se sigue navegando en esa pestaña.

c) Se cierra la ventana de navegación que se estuviera utilizando y se abre otra.

LOCALSTORAGE

Con sessionStorage se puede tratar adecuadamente el flujo de información durante sesiones de navegación (por ejemplo, mantener los datos de un carrito de la compra). ¿Pero qué ocurre si quisiéramos almacenar esa información más allá de una sesión y que estuviera disponible tanto si se cierra el navegador y se vuelve a abrir como si se continúa navegando en una ventana distinta de la inicial?

Esto trata de ser respondido por localStorage. Este objeto tiene las mismas propiedades y métodos que sessionStorage, pero su persistencia va más allá de la sesión.

Sin embargo sessionStorage tiene limitaciones importantes: el almacenamiento de los datos depende, en última instancia, de la voluntad del usuario. Aunque la forma en que se almacenan los datos depende del navegador, podemos considerar que una limpieza de caché del navegador hará que se borren los datos almacenados con localStorage. Si el usuario no limpia la caché, los datos se mantendrán durante mucho tiempo. En cambio hay usuarios que tienen configurado el navegador para que la caché se limpie en cada ocasión en que cierren el navegador. En este caso la persistencia que ofrece localStorage es similar a la que ofrece sessionStorage.

No podemos confiar el funcionamiento de una aplicación web a que el usuario limpie o no limpie caché, por tanto deberemos seguir trabajando con datos del lado del servidor siempre que deseemos obtener una persistencia de duración indefinida.

EL EVENTO STORAGE

Dado que localStorage permite que se reconozcan datos desde distintas ventanas ¿Cómo detectar en una ventana que se ha producido un cambio en los datos? Para ello se definió el evento storage: este evento se dispara cuando tiene lugar un cambio en el espacio de almacenamiento y puede ser detectado por las distintas ventanas que estén abiertas.

Para crear una respuesta a este evento podemos escribir:

window.addEventListener("storage", nombreFuncionRespuesta, false);

Donde nombreFuncionRespuesta es el nombre de la función que se invocará cuando se produzca el evento.

DIFERENCIAS ENTRE COOKIES Y STORAGE

Los objetos storage juegan un papel en alguna medida similar a las cookies, pero por otro lado hay diferencias importantes:

a) Las cookies están disponibles tanto en el servidor como en el navegador del usuario, los objetos storage sólo están disponibles en el navegador del usuario.

b) Las cookies se concibieron como pequeños paquetes de identificación, con una capacidad limitada (unos 4 Kb). Los objetos storage se han concebido para almacenar datos a mayor escala (pudiendo comprender cientos o miles de datos con un espacio de almacenamiento típicamente de varios Mb).

Tener en cuenta que de una forma u otra, ni las cookies ni los objetos storage están pensados para el almacenamiento de grandes volúmenes de información, sino para la gestión de los flujos de datos propios de la navegación web.

MÁS POSIBILIDADES DE JAVASCRIPT CON LOS NAVEGADORES MODERNOS

En este curso hemos hecho un recorrido por los fundamentos del lenguaje JavaScript y muchas de las tecnologías que hay en torno a él, pero extendernos en todas las posibilidades del lenguaje y extensiones al mismo requeriría alargar un curso y ese no era nuestro objetivo. Nuestro objetivo era dar una visión sintética del lenguaje y sus posibilidades.

A modo de referencia para aquellas personas que quieran profundizar en la programación JavaScript vamos a citar algunas extensiones o Apis de interés con las que no hemos trabajado:

Api forms: concebida para facilitar la validación de formularios definiendo eventos, estados, métodos, etc.

Api Indexed Database: concebida para almacenar grandes volúmenes de información estructurada, de forma análoga a una base de datos pero con ciertas particularidades.

Api File: destinadas a facilitar la operación con archivos y directorios.

Cross Document Messaging: concebida para comunicar entre sí diferentes ventanas.

Web sockets: concebida para envío y recepción de información del servidor en periodos cortos de tiempo, útil para aplicaciones de tiempo real (como una conversación a través de chat).

Web workers: concebida para hacer posible procesar múltiples tareas al mismo tiempo (multiprocesamiento).

History: concebida para permitir la navegación por páginas visitadas e incluso por estados dentro de la evolución de una web cuando no ha habido recarga explícita de la misma (por ejemplo debido al uso de Ajax).

Offline: concebida para detectar la falta de conexión a internet y permitir el trabajo sin conexión.

EJERCICIO

Si has seguido el curso hasta aquí darte nuestra enhorabuena. No vamos a plantear aquí un ejercicio, únicamente te propondremos que dejes un comentario en los foros aprenderaprogramar.com indicándonos qué te ha parecido el curso, qué es lo que te ha parecido lo mejor de él y cuáles serían los aspectos mejorables. Preparar este curso ha requerido muchas horas de dedicación, pensamos que pedirte unos minutos para que nos ayudes a mejorar y conocer tu opinión es razonable y será algo que te agradeceremos.

Próxima entrega: CU01199E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206