Desarrollo Web en Entorno Cliente

UD 07. Almacenamiento en Javascript

Actualizado Noviembre 2020

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

📖 **Importante**

❕ **Atención**

💬 **Interesante**

ÍNDICE DE CONTENIDO

UD07. ALMACENAMIENTO EN JAVASCRIPT

1. INTRODUCCIÓN

Javascript aplicado a navegadores en principio no está pensado para almacenar grandes cantidades de datos. Existen dos formas de almacenar información en Javascript:

- **Cookies**: para guardar pequeñas variables con información. Usualmente suelen guardar información de logins de usuarios. Las cookies se pueden guardar en el cliente y ser generadas por el cliente, pero también pueden ser enviadas por el servidor.
    - En desuso, en el tema se comentan como curiosidad.
- **LocalStorage**: al crecer Javascript y las aplicaciones asociadas a ellos, los navegadores más modernos implementan LocalStorage. Es más parecido a guardar datos en una aplicación de escritorio. El límite de información a almacenar puede variar según la implementación del navegador pero está definido en torno a los 5 MB.

Estas formas de almacenar información pueden combinarse con JSON para almacenar estructuras complejas como objetos o arrays.

2. RECORDANDO JSON

JSON (siglas de JavaScript Object Notation) es un formato por el cual en formato texto se puede representar la estructura de elementos Javascript. JSON nos permite convertir variables, arrays y objetos en cadenas de texto y así poder facilitar la comunicación entre distintos programas, enviándole el contenido de un objeto como cadena de texto.

Por su simplicidad, su uso no se ha quedado solo en Javascript, sino que se ha convertido en un estándar "de facto" de formato para intercambiar datos entre computadores, utilizandose en multitud de lenguajes de programación.

Los métodos estudiados en este tema (cookies y LocalStorage) permiten guardar cadenas de texto. La forma de almacenar datos más complejos es pasarlos a formato JSON (que es texto al fin y al cabo) y así almacenarlos como cadenas.

Más información sobre el formato en http://www.w3schools.com/js/js_json.asp

En Javascript ES6 Para convertir un objeto a texto siguiendo el formato JSON usamos:

```
textoJSON=JSON.stringify(objeto);
```

Y para convertir una cadena de texto en JSON a un objeto (operación inversa a la anterior) usamos:

```
var objeto = JSON.parse(textoJSON);
```

Ejemplo:

```
let miArray=new Array();
miArray[0]="HOLA";
let textoJSON=JSON.stringify(miArray);
let arrayReconstruido=JSON.parse(textoJSON);
alert(arrayReconstruido[0]);
```

3. COOKIES

Una cookie es una información enviada por un sitio web (y asociada a ese dominio Web) que el navegador se encarga de almacenar. (es decir, se almacenan en el cliente y no en el servidor).

Generalmente suelen estar guardadas en fichero de texto, aunque esto nos da igual ya que nosotros las utilizaremos con comandos específicos de Javascript que nos abstraen de como son almacenadas.

Básicamente, esta información son una o varias variables con su contenido asociado.

Un ejemplo: una página de "midominio.com" crea una cookie. Esta cookie contiene una variable "usuario" y su contenido es "pepe". Esta cookie solo es accesible desde el navegador desde "midominio.com". Una página del dominio "pepe.com" no podría modificarla ni leerla y si creara una cookie con la variable usuario, sería una cookie independiente.

Más información en https://es.wikipedia.org/wiki/Cookie_(inform%C3%A1tica)

Muchas veces las cookies son utilizadas para temas de autentificación de usuarios, sobre todo lo relacionado con "autentificación automática".

Por ejemplo, en lenguajes de servidor como PHP, se combinan con las sesiones (que guarda información en el servidor) para permitir una autentificación adecuada y relativamente segura http://php.net/manual/es/session.examples.basic.php

3.1 Formato de cookies

Para crear una cookie, usamos document.cookie. Esto es una string especial que tiene el siguiente formato:

```
"variable=valor;expires=fecha expiración;path=/"
```

Donde variable es la variable a establecer, valor su valor, expires es la fecha de expiración (la forma de borrar cookies es cambiarles la fecha de expiración a una ya pasada) y path el lugar del dominio donde son válidas

Ejemplo:

```
document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013
12:00:00 UTC; path=/";
```

Para una explicación detallada del formato y uso de cookies desde Javascript, podéis acudir aquí http://www.w3schools.com/js/js_cookies.asp

A efectos prácticos si vamos a operar con Cookies (desaconsejado, ya que están obsoletas), recomiendo el uso de estas funciones ya establecidas para crear, consultar y eliminar cookies, obtenidas de la página citada anteriormente.

setCookie: establece una cookie indicando variable, valor y días para la expiración

```
function setCookie(cname, cvalue, exdays) {
    let d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    let expires = "expires="+d.toUTCString();
    document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```

getCookie: recibe el nombre de la variabley devuelve su valor

```
function getCookie(cname) {
    let name = cname + "=";
    let  ca = document.cookie.split(';');
    for(let i = 0; i < ca.length; i++) {
        let c = ca[i];
        while (c.charAt(0) == ' ') {
            c = c.substring(1);
        }
        if (c.indexOf(name) == 0) {
            return c.substring(name.length, c.length);
        }
    }
    return "";
}
```

deleteCookie: Elimina la cookie de la variable establecida

```
function deleteCookie(cname) {
     document.cookie = cname+'=; expires=Thu, 01 Jan 1970 00:00:01
GMT;path=/';
}
```

3.2 Probando el funcionamiento de las cookies

Podéis probarlas usando esta función, que intenta obtener una cookie “username”. Si existe, muestra un mensaje. Si no existe, la establece.

Lógicamente este ejemplo podéis combinarlo con todo lo aprendido anteriormente de Javascript.

```
function checkCookie() {
    let user = getCookie("username");
    if (user !=) "") {
        alert("Welcome again " + user);
    } else {
        user = prompt("Please enter your name:", "");
        if (user !=="" && user !==null) {
            setCookie("username", user, 365);
        }
    }
}
```

4. LOCALSTORAGE

El llamado "localStorage" es una tecnología de almacenamiento existente en los navegadores más modernos, siendo incompatible con navegadores antiguos. La información se almacena en el cliente y generalmente posee al menos 5MB para guardar información.

Para más información http://www.w3schools.com/html/html5_webstorage.asp

4.1 Operar con localStorage

A efectos prácticos debéis conocer lo siguiente:

- Hay dos objetos: localStorage y sessionStorage. La diferencia entre uno y otro es que localStorage almacena la información indefinidamente y sessionStorage lo hace solo mientras la ventana de la página esté abierta. Por el resto de detalles ambos objetos funcionan igual y en los ejemplos nos referiremos únicamente a localStorage.
- Las funciones a utilizar son 3: setItem, getItem, removeItem.

Ejemplo setItem, getItem y removeItem:

```
localStorage.setItem("apellido", "Garcia");
alert(localStorage.getItem("apellido"));
localStorage.removeItem("apellido");
alert(localStorage.getItem("apellido"));
```

4.2 Vaciar localStorage

Si queremos vaciar por completo (es decir, eliminar todas las entradas) de nuestro localStorage, lo podemos hacer simplemente usando clear();

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

4.3 JSON y localStorage

Como hemos comentado anteriormente localStorage permite almacenar texto. Entonces, si queremos almacenar además de texto, elementos de nuestro programa como un array, un objeto, etc. debemos convertirlos a un formato de texto, pero que mantenga toda su información.

Para ello en Javascript tenemos JSON (Javascript Object Notation).

Para este propósito utilizaremos JSON.stringify para convertir un elemento a texto y JSON.parse para convertir texto de nuevo al elemento.

```
let miArray=[1,2,3]
let miArray2;
// Guardamos en localStorage el array con stringify
localStorage.setItem("valorArray",JSON.stringify(miArray));
//Recuperamos el texto y lo convertimos de nuevo a array con parse
miArray2=JSON.parse(localStorage.getItem("valorArray"));
```

5. BIBLIOGRAFÍA

[1] Javascript Mozilla Developer https://developer.mozilla.org/es/docs/Web/JavaScript

[2] Javascript ES6 W3C https://www.w3schools.com/js/js_es6.asp

6. AUTORES (EN ORDEN ALFABÉTICO)

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- García Barea, Sergi