Uso del LocalStorage

La llegada de html5 trajo consigo muchas novedades interesantes y muy útiles, entre las cuales se encuentra localstorage o cómo sería su traducción en castellano, *almacenamiento local*. Es una forma de almacenar datos en el navegador del usuario, muy útil para guardar información para ser usada en cualquiera de las páginas de nuestra web, como un carrito de la compra. Pero si ya existen las cookies para almacenar información, para que usar localstorage?.

Diferencias entre Cookies y LocalStorage.

Cookies	LocalStorage
• Espacio limitado: Una cookie sólo puede ocupar 4kb de espacio.Es por eso que las cookies suelen utilizarse sólo para almacenar un hash o un identificador que será utilizado por el servidor para identificar la visita. • Cada vez que se realiza una petición al servidor, toda la información que está almacenada en las cookies es enviada y también es recibida nuevamente con la respuesta del servidor. O sea, en los intercambios de información entre el navegador web y el servidor siempre van pegadas las cookies. • Las cookies tienen una caducidad.	• Espacio menos limitado: localStorage puede ocupar entre 5 y 10MB dependiendo del navegador web. Con 5 o 10 megas ya podemos tener algo más de información • La información almacenada con localStorage no es enviada al servidor en cada petición. • No existe una caducidad para localStorage, la información quedará almacenada hasta que se elimine expresamente. Aunque se cierre el navegador.

Como guardar, consultar y eliminar datos almacenados con LocalStorage.

Guardar

Para utilizar el proceso de guardar un dato sea nombres, numeros telefonicos, correos, fechas o cualquier otro, recordando que solo se puede guardar string, no se pueden guardar vectores, ni matrices, ni booleans, ni floats, solo strings.
Entonces para que guardemos datos en el almacenamiento local pongamos el siguiente ejemplo: tenemos un input, tipo texto, el cual pensamos escribir un nombre, le asignamos un identificador al input llamado nombretxt, entonces procedemos a capturar lo escrito en el campo de texto y guardarlo en una variable para luego guardarlo en el localstorage.

```
1. var nom = document.getElementById("nombretxt").value;
2.  localStorage.setItem("Nombre",nom);
```

De esta manera capturamos el dato y lo guardamos en el localstorage.

Consultar

Ahora para consultar solo debemos llamar la variable que le dimos al guardar el dato, en este caso se le asigno la vaiable Nombre, y el proceso se realiza de la siguiente manera.

```
1. var nombre = localStorage.getItem("Nombre");
2. alert(nombre);
```

Mediante este proceso consultamos y mostramos mediante un alert los datos, aclarando que podemos mostrar los datos donde nosotros deseemos, sea un div, o donde queramos.

Eliminar

Si deseamos eliminar los datos guardados solo debemos hacer lo siguiente.

```
1. localStorage.removeItem("Nombre");
```

Y de esta manera ya hemos eliminado los datos que deseemos.

json-Intercambio de datos

Al intercambiar datos entre un navegador y un servidor, los datos sólo pueden ser de tipo texto. JSON es texto, y podemos convertir cualquier objeto JavaScript en JSON, y enviar JSON al servidor. También podemos convertir cualquier JSON recibido del servidor en objetos JavaScript. De esta manera podemos trabajar con los datos como objetos JavaScript, sin complicados análisis y traducciones.

La diferencia entre un Json o objeto JavaScript es que las “llaves” van entrecomilladas en un Json y sin entrecomillar para JavaScript. Para convertir un Json a objeto puro JavaScript existe la función JSON.parse(objeto_Json);. Pero como JavaScript entiende directamente el JSON como un objeto no hace falta convertirlo para recorrerlo.

Objeto JavaScript

var data = [{Id: 10004, PageName: "club"}, {Id: 10040, PageName: "qaz"}, {Id: 10059, PageName: "jjjjjjj"}];

Objeto JSON

var data = [{"Id": 10004, "PageName": "club"}, {"Id": 10040, "PageName": "qaz"}, {"Id": 10059, "PageName": "jjjjjjj"}];

¿Qué es JSON?

JSON significa JavaScript Object Notation JSON.
Es un formato ligero de intercambio de datos en formato de texto JSON
Es "auto-descriptivo" y fácil de entender

¿Por qué utilizar JSON?

Dado que el formato JSON es sólo texto, puede ser fácilmente enviado y recibido desde un servidor y utilizado como formato de datos por cualquier lenguaje de programación. JavaScript tiene una función incorporada para convertir una cadena, escrita en formato JSON, en objetos JavaScript nativos: JSON.parse () Por lo tanto, si recibe datos de un servidor, en formato JSON, puede usarlo como cualquier otro objeto JavaScript.

Envío de datos

Si tiene datos almacenados en un objeto JavaScript, puede convertir el objeto en JSON y enviarlo a un servidor:

```
var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Recepción de datos

Si recibe datos en formato JSON en texto puro (todo entrecomillado), puede convertirlos en un objeto JavaScript como el ejemplo que hay a continuación. Para convertir un Json a objeto puro JavaScript existe la función JSON.parse(objeto_Json); pero recordar que JavaScript no necesita convertir un Json a array ya que es capaz de gestionar el objeto:

```
var myJSON = '{ "name":"John", "age":31, "city":"New York" }';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Recorrer objeto Json

Recorrer un objeto Json sirve tanto para mostrar toda la información como para buscar un elemento (key/value) añadiendo un if en el bucle.

Con JavaScript:

Con un simple for se puede recorrer un objeto Json con JavaScript.

```
var data = [ {"Id": 10004, "PageName": "club"}, {"Id": 10040, "PageName": "qaz"}, {"Id": 10059, "PageName": "jjjjjjj"} ];

for(i in data) {console.log(data[i].PageName); } // (o el campo que necesites)
```

Con jQuery

```
var data = [ {"Id": 10004, "PageName": "club"}, {"Id": 10040, "PageName": "qaz"}, {"Id": 10059, "PageName": "jjjjjjj"} ];

//Forma 1

$.each(data, function(i, item) {

        console.log(data[i].PageName); })
```

```
//Forma 2

$.each(data, function(i, item) {

      console.log(item.PageName); })
```

Modificar objeto JSON

Para modificar información, hay que hacer referencia a la llave y valor antiguo para asignarle el valor que se quiere modificar.
Ejemplo:
Dado el siguiente objeto:

```
var json = [{ "id": "5001", "type": "None" }, { "id": "5002", "type":
"Glazed" }, { "id": "5005", "type": "Sugar" }, { "id": "5003", "type":
"Chocolate" }, { "id": "5004", "type": "Maple" }, { "id": "5009", "type":
"Juice" }];
```

Si el objetivo es cambiar el id 5001 por el id 6001, como es habitual, primero hay que recorrer todo el objeto y cuando coincida con el valor a buscar entonces se le asigna el nuevo valor.

```
for(i in json) {

    if (json[i].id=="5001") json[i].id="6001";

 }
```

Versión mejorada donde se crea función para modificar objeto pasando a la función el campo llave, valor antiguo (a buscar) y valor nuevo.

```
function replaceByValue( field, oldvalue, newvalue ) {

for( var k = 0; k < json.length; ++k ) {

      if( oldvalue == json[k][field] ) {json[k][field] = newvalue ; }

} return json; } /** * Let's test */

console.log(json);

replaceByValue('id','5001','5010');

console.log(json);

replaceByValue('type','Chocolate','only water');

console.log(json);
```

Almacenamiento de datos Json en localstorage

Al almacenar datos, los datos tienen que ser un cierto formato, e independientemente de donde usted elija almacenar, el texto es siempre uno de los formatos legales. JSON permite almacenar objetos JavaScript como texto.

```
//Guardando información:
myObj = { "name":"John", "age":31, "city":"New York" };
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);


//Obtener variable:
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```

Obtener y enviar los datos de un formulario a través de una llamada Ajax

Obtener datos formulario con JavaScript/jQuery

Antes de poder realizar una llamada de tipo Ajax de Jquery, necesitamos almacenar los datos del formulario en una variable de tipo array de pares nombre_campo:valor_campo. Para realizar dicha acción se comentan a continuación las funciones destinadas a recoger la información de un formulario.

serialize()

Función de JS que serializa los campos de un formulario (nombre y valor de cada campo) y lo almacena en una variable de tipo texto separando los campos por el símbolo &.

```
var serializado = $("#id_form").serialize();
```

serializeArray()

Función de JS que serializa los campos de un formulario (nombre y valor de cada campo) y lo almacena en una variable

de tipo array, de las más utilizadas porque es la forma más cómoda de almacenar información.

```
var serializadoArray=$("#id_form").serializeArray();
```

Con estas funciones y conociendo los objetos de datos Json comentados anteriormente, se puede crear un Json con el array creado con la función seriralizeArray().

```
var jsonData= JSON.stringify($("#id_form").serializeArray());
```

Enviar datos formulario realizando una llamada Ajax de jQuery.

Siguiendo el ejemplo con el nombre de variables anteriores, al tener almacenados los datos del formulario en un Json, se pueden enviar al archivo PHP encargado de gestionar/almacenar la información recibida a través de una llamada jQuery Ajax.

```
   $.ajax({
        url: 'php/recibeJson.php',
        type: 'POST',
        dataType: 'json',
        data: jsonData,
        success : function(result){
          console.log(result);

        },
        error: function(result){
          alert("errorrrrrr!!!");
        }
      })
```

PHP

Ejemplo de archivo PHP que comprueba si recibe información para tratar la información recibida, preparar lo que sería la sentencia MySql para almacenar la información (INSERT).

```
<?php
//Para comprobar si se recibe un post desde un ajax
if ($_SERVER['REQUEST_METHOD']==='POST'){
     //Para almacenar los datos JSON recibidos en una variable
     $request= file_get_contents('php://input');
     //Para convertir un Json en un array de php
     $datos = json_decode($request,true);
     $valores='"';
     $campos="";
     foreach ($datos as $key => $value){
     $campos .= $value['name'].',';
     $valores.= $value['value'].'","';
     }
     $campos = substr($campos,0, -1);
     $valores = substr($valores,0, -2);

     $sql = "INSERT INTO enterprise ($campos) VALUES
($valores)";
     $mysqli = new mysqli('127.0.0.1', 'root', '', 'crm');
     mysqli_set_charset($mysqli,"utf8");
     if ($mysqli) {
          $query=$mysqli->query($sql);
          $mysqli->close();
     }

     if ($query) {
```

```
        echo json_encode([
            "campos"  => $campos,
            "error"        => 0,
            "valores" => $valores,
            "sql"          => $sql,
            "resultado" => "se ha grabado"
            ]);
        }
        else {
            echo json_encode([
            "campos"  => $campos,
            "error"        => 1,
            "valores" => $valores,
            "sql"          => $sql,
            "resultado" => "NO se ha grabado!!"
            ]);
        }
}
else {
echo json_encode([
            "campos" => "KO",
            "error"        => 1,
            "valores" => "no hay"
        ]);
}
?>
```

Obtener información desde base de datos

Partiendo del siguiente HTML

```
<div class="container">
    <h2>Formularios de contacto recibidos</h2>
    <table id="listado">
    <thead>
    <tr>
      <th>ID</th>
      <th>Nombre</th>
      <th>Email</th>
      <th>Mensaje</th>
      <th>Fecha</th>
    </tr>
    </thead>
    <tbody>
    </tbody>
    </table>
 </div>
```

Para obtener información de base de datos, el método de la llamada Ajax pasa de ser POST a ser GET. En este ejemplo, existe una tabla de la base de datos con los formularios de contacto recibidos (contact_form), se obtiene la información, se recorren los datos para crear la estructura de la tabla html que mostrará por pantalla el resultado y una vez recorrido todos los datos recibidos en formato Json y con la ayuda de jQuery se hace un $("#listado tbody").html(tbl_body); para insertar las filas creadas y almacenadas en la variable tbl_body.

JavaScript/jQuery

```
$(document).ready(function() {
    var debug=true;
    $.ajax({
      url: '../php/getListPlatos.php',
      type: 'GET',
      dataType: 'json',
      success : function(result){
        console.log(result);
        var tbl_body = "";
        //Recorrer el array de la query que manda el php
        $.each(result.query, function() {
            if (debug) console.log("Pintando");
            if (debug) console.log(result.cats);

            var tbl_row = "";
            $.each(this, function(campo , valor) {
              if(campo=="foto") {
                tbl_row += "<td>"
                        +"<img class='z-depth-3' src='../"
                        +valor
                        +"' width='90px'>"
                        +"</td>";
              }
              else {
                tbl_row += "<td>"+valor+"</td>";
              }

            });
            tbl_body += "<tr>"+tbl_row+"</tr>";
        });
        if (debug) console.log(tbl_body);
        $("#listado tbody").html(tbl_body);
      },
      error: function(result){
        alert("errorrrrrr!!!");
      }
    });
});
```

PHP getListPlatos.php

El código PHP muy parecido al de insertar datos pero con la diferencia que se hace un “SELECT” en vez de un insert y el resultado se pasa por Json como respuesta. A destacar la línea $rows = $query->fetch_all(MYSQLI_ASSOC); que es la que pasa el resultado de la select a una variable.

```
<?php
//Para comprobar si se recibe un post desde un ajax
if ($_SERVER['REQUEST_METHOD']==='GET'){
	$sql = "SELECT * FROM contactform";
	$mysqli = new mysqli('127.0.0.1', 'root', '', 'takeaway');
	mysqli_set_charset($mysqli,"utf8");
	if ($mysqli) {
		$query=$mysqli->query($sql);
		$mysqli->close();
		$rows = $query->fetch_all(MYSQLI_ASSOC);
		}
	if ($query) {
	echo json_encode([
		"query"		=> $rows,
		"error"		=> 0,
		"resultado" => "se ha cargado"
		]);
	}
	else {
		echo json_encode([
		"query"		=> $query,
		"error"		=> 1,
		"resultado" => "NO se ha cargado!!"
		]);
	}
}
else {
echo json_encode([
		"query" => "KO",
		"error"		=> 1,
		"resultado"	=> "no hay"
	]);
}
?>
```