Making Use of HTML5 Storage

March 6, 2012 *7 Comments*

HTML5 offers lots of significant advantages to developers, but browser support is still pretty low. There's no reason not to start inserting HTML5 functions into sites now, as long as you take the necessary steps to check for browser support and provide alternative content for everyone else.

In this tutorial we'll go over the basics of using HTML5 and JavaScript to exploit the enhanced storage facilities on offer. With HTML5 you can store more data – and store it more efficiently.

The two main data storage objects are **localStorage** and **sessionStorage**. The**localStorage** data persists between sessions, while data stored using the **sessionStorage**object expires with the current user session. HTML5 storage data is not passed along with each browser request, so you can store large amounts of data while retaining a good level of efficiency.

HTML5 Page Outline

Use the following outline for your HTML5 page:

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
/*Functions here*/
</script>
</head>
<body>
</body>
</html>
```

We will add our storage functions within the script section in the head area and also within the page body, where we will place a few simple elements for demonstration.

Local Storage

The JavaScript **localStorage** object allows HTML5 pages to store data items at client side, then access these items during future sessions. To demonstrate, add the following to the body section of your page:

```
<input id="userText" type="text"/>
<input type="button" value="store" onclick="storeText('userText')"/>
<div id="result">
Result here
</div>
```

Here we have added a text-field, a button calling a function, which it passes the ID of the text-field input element, and a **div** element with some placeholder text content. When the user presses the button, any text they have entered in the text-field is going to be stored using the **localStorage** object. Add the following function to your script section in the page head:

```
function storeText(inputID) {

//check to see if the localStorage item already exists
var userInput = localStorage.userInfo;

//set it up for new data to be appended
if(userInput!=null) userInput+=" ";
else userInput="";

//add the new data
userInput += document.getElementById(inputID).value;

//set the localStorage field with the updated data
localStorage.userInfo = userInput;

//write it to the page
document.getElementById("result").innerHTML = localStorage.userInfo + "
- entered to date";
}
```

This function first checks the **localStorage** object for a specific item of data, named “userInfo” in this case. If the item has not been created yet, we start with an empty string. The function then concatenates the initial string with the new user data from the text-field, then writes the result back to the **localStorage** object field. Finally, the function writes the data into the HTML result element.
For an additional demonstration, you can optionally add the following inside your **div**result element, instead of the placeholder text:

```
<script type="text/javascript">
if(localStorage.userInfo==null) document.write("no info so far");
else document.write(localStorage.userInfo + " - entered before");
</script>
```

Save your page and upload it, then browse to it to test it. Each time you add new data it should be appended to the text stored and displayed.

Session Storage

If you have data items you want to store on the client but don't need after the session ends, it's best to use the **sessionStorage** object. Alter or add to your HTML body section as follows:

```
<input type="button" value="store" onclick="pressedIt()"/>
<div id="sresult">
Result here
</div>
```

Again, we're testing the function using a button, with a result section to demonstrate the stored data. Add the following function to your script section:

```
function pressedIt() {

//check to see if the sessionStorage data item already exists
var userPresses = Number(sessionStorage.userPressesSoFar);

//if not instantiate it to zero
if(userPresses==null) userPresses=0;

//increment
userPresses++;

//set the sessionStorage field with the updated data
sessionStorage.userPressesSoFar = userPresses;

//write it to the page
document.getElementById("sresult").innerHTML =
sessionStorage.userPressesSoFar + " presses so far";
}
```

The function first attempts to read the data item we are concerned with, parsing it as a Number as it is stored as a text string. If it has not yet been set, i.e. if the user has not yet pressed the button this session, the function starts the counter at zero. The code then increments the counter variable to reflect the button click that has just occurred, then stores this new value in the **sessionStorage** object. Finally the code writes the result into the HTML **div** element.

As with the **localStorage** example, you can add an additional demo to your page by replacing the text content of your result **div** element in the page body with the following:

```
<script type="text/javascript">
if(sessionStorage.userPressesSoFar==null) document.write("no presses so
far");
```

```
else document.write(sessionStorage.userPressesSoFar + " presses so
far");
</script>
```

Save and upload your page again to test it. Make sure it works by clicking a few times, then closing the browser window and opening it again – it should reset to zero for each session.

Alternatives

There is an alternative method for using both the **localStorage** and **sessionStorage**objects, in which you call the **getItem** and **setItem** methods. The data items are stored as key-value pairs, but rather than accessing them directly by name as in the above examples, you can use the API methods as follows:

```
//var userInput = localStorage.userInfo;
var userInput = localStorage.getItem("userInfo");

//localStorage.userInfo = userInput;
localStorage.setItem("userInfo", userInput);

//var userPresses = Number(sessionStorage.userPressesSoFar);
var userPresses = Number(sessionStorage.getItem("userPressesSoFar"));

//sessionStorage.userPressesSoFar=userPresses;
sessionStorage.setItem("userPressesSoFar", userPresses);
```

In these examples we have called on HTML5 functionality using buttons for demonstration, but you can of course make your own scripts execute on page loads or other events.

Remember not to rely on HTML5 functions for essential features of your sites, and to provide alternatives for user's whose browsers do not yet support the standard. Developers are using various techniques and even third party libraries like Modernizr to check for browser support with varying degrees of success. At the most basic level, you can check if the browser window object has the **localStorage** or **sessionStorage** variable as follows:

```
if(localStorage) {
//implement HTML5 localStorage functions
}
else {
//implement alternative
}
```

HTML5 will ultimately give developers the tools to create more robust, engaging and interactive Web applications using facilities such as improved storage. In the meantime we just have to be patient.

“Great introductory post on localStorage and sessionStorage.

I'd recommend checking against all falsies instead of just "null", eg.

"if (!userPresses) userPresses = 0;" as opposed to "if(userPresses==null) userPresses=0;".

"if (userInput) userInput+=" ";" as opposed to "if(userInput!=null) userInput+=" ";"

That way, you are checking against 0, "", NaN, false, null, and undefined.”