

Practical example 2: save/restore user's preferences

INTRODUCTION

Local stores are also useful for saving/restoring user preferences of Web Applications. For example, the JS Bin tool you have been using since the beginning of this course uses `localStorage` to store the list of tabs you open, and their width:

Try to drag this vertical bar and you will see that the **splitterSettings** key in **localStorage** changes its value!

```
<!DOCTYPE html>
<html>
<body onload="init()">
  <canvas id="canvas" width=
height="400" style="border:solid 2px
black"></canvas>
  <ul id="sliders"></ul>
</body>
</html>
```

```
var canvas;
var width = 150, height=150;
// The origin, bottom left corner,
relative to the origin in the canvas
var x = 40, y = 190;
```

Key	Value
password	tutu
personne	{"nom":"buffa","prenom":"michel"}
phone	234
phone_nr	06 62 65 93 45
postalCode	06410
prenom	michel
settings	{"panels":["html","javascript","live"],"editor":{},"font":14,"addons":{"c...
splitterSettings	[{"x":439}, {"x":654}, {"x":624}, {"x":669.587524414063}]
street_number	542
testclients	[{"nom":"Michel","prenom":"Buffa","email":"buffa@unice.fr","dateDe...
user	tutu
username	Michel

In this way, the next time you come back to JS Bin, "it will remember your last settings".

Another example is a guitar FX processor / amp simulator your instructor is writing with some of his students. It uses `localStorage` to save/restore presets values:

file:///Users/mbuffa/Documents/Cours/2014-2015/Miage%20NDP/Nice/ProjetNourielAzri...

Applications https://vegas.univ-tl... Films et Serie en Str... Coming Soon SpriteMe manifestR Manifesto Advanced Rest Clie... Autres favoris

Gain

Sauvegarder Restaurer

This graph is also saved in the localStorage

Preset values

Amp

Drive Bass Mid Treb Presence Boost Master Type

3 5 5 5 5 ☒ 8 Brit Man

Elements Network Sources Timeline Profiles Resources Audits Console NetBeans

Frames Web SQL IndexedDB Local Storage file:// Session Storage file:// Cookies Application Cache

Key	Value
best	3
clients	[{"nom": "Mihel", "prenom": "Buffa", "email": "toto@toto.fr", "date...}
contacts	[{"nom": "Buffa", "prenom": "Michel", "adresse": "12 rue des fleu...}
gPresets	[{"m_nom": "GPresetDebut", "m_id": "jsPlumb_2_1", "m_pr...}
is	undefined
resultat	0,mic;15,tt;
settings	{"panels":["html","live"],"editor":{"simple":false},"font":14,"ad...}
spectrum.demo	rab(238. 204. 204)

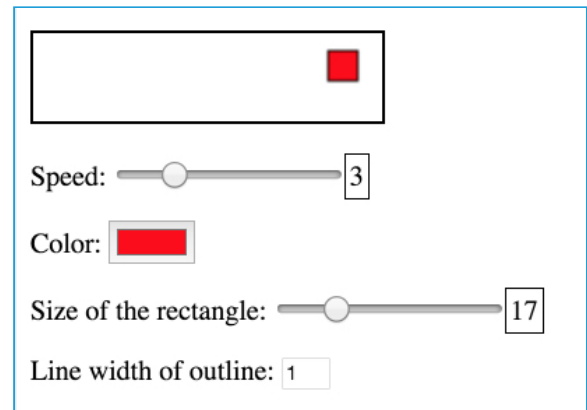
ps4html5.jpg xboxone1.jpg ChromeDevelopersTool.png colorpicketyosemite.png

Tout afficher

PRACTICAL EXAMPLE: SAVE/RESTORE PREFERENCES OF AN
EXAMPLE YOU HAVE ALREADY SEEN

[Original example on JS Bin](#): we can change the color, size and speed of the animated rectangle. However, each time we come back to the page, default values are restored.

We would like to save the current values and find them back as they were when we come back to the page.



[Here is a modified example that saves/restores its state, you can try it at JS Bin](#). In this modified version of the animated rectangle example, you can set the color, size, speed, etc. And if you reload the page, the state of the different input field is restored, but also the internal variables. Check the source code in the JS Bin example and read the following explanations.

We used the same generic code for saving/restoring input fields' values we saw in the first example that used `localStorage`. The only difference is that we renamed the two generic functions so that they correspond better to their role here (instead of `saveFormContent` we called the function `restorePreferences`).

The function `initPreferences` is executed when the page is loaded.

Source code extract:

```
function initPreferences() {  
    console.log("Adding input listener to all input fields");  
    // add an input listener to all input fields  
  
    var listOfInputsInForm = document.querySelectorAll("input");  
    for(var i= 0; i <listOfInputsInForm.length; i++) {  
        addInputListener(listOfInputsInForm[i]);  
    }  
    // restore preferences  
10.    restorePreferences();  
        applyGUIvalues(); // Use the input fields' values we just  
        restored to set internal
```

```
        // size, incX, color, lineWidth
```

```
variables
```

```
}
```

```
function addInputListener(inputField) {
```

```
    // same as before
```

```
}
```

```
function restorePreferences() {
```

```
21.    // same as old restoreFormContent
```

```
}
```

```
function applyGUIvalues() {
```

```
    // Check restored input field content to set the size of  
the rectangle
```

```
    var sizeWidget =document.getElementById("size");
```

```
    size =Math.sign(incX)*parseInt(sizeWidget.value);
```

```
    // also update the outline element's value
```

```
    document.getElementById("sizeValue").innerHTML= size;
```

```
    // Check restored input field content to set the color of  
the rectangle
```

```
    var colorWidget =document.getElementById("color");
```

```
    ctx.fillStyle = colorWidget.value;
```

```
34.
```

```
    // Check restored input field content to set the speed of  
the rectangle
```

```
    var speedWidget =document.getElementById("speed");
```

```
    incX =Math.sign(incX)*parseInt(speedWidget.value);
```

```
    // also update the outline element's value
```

```
    document.getElementById("speedValue").innerHTML= Math.abs(incX);
```

```
    // Check restored input field content to set  
the lineWidth of the rectangle
```

```
    var lineWidthWidget =document.getElementById("lineWidth");
```

```
    ctx.lineWidth =parseInt(lineWidthWidget.value);
```

```
}
```