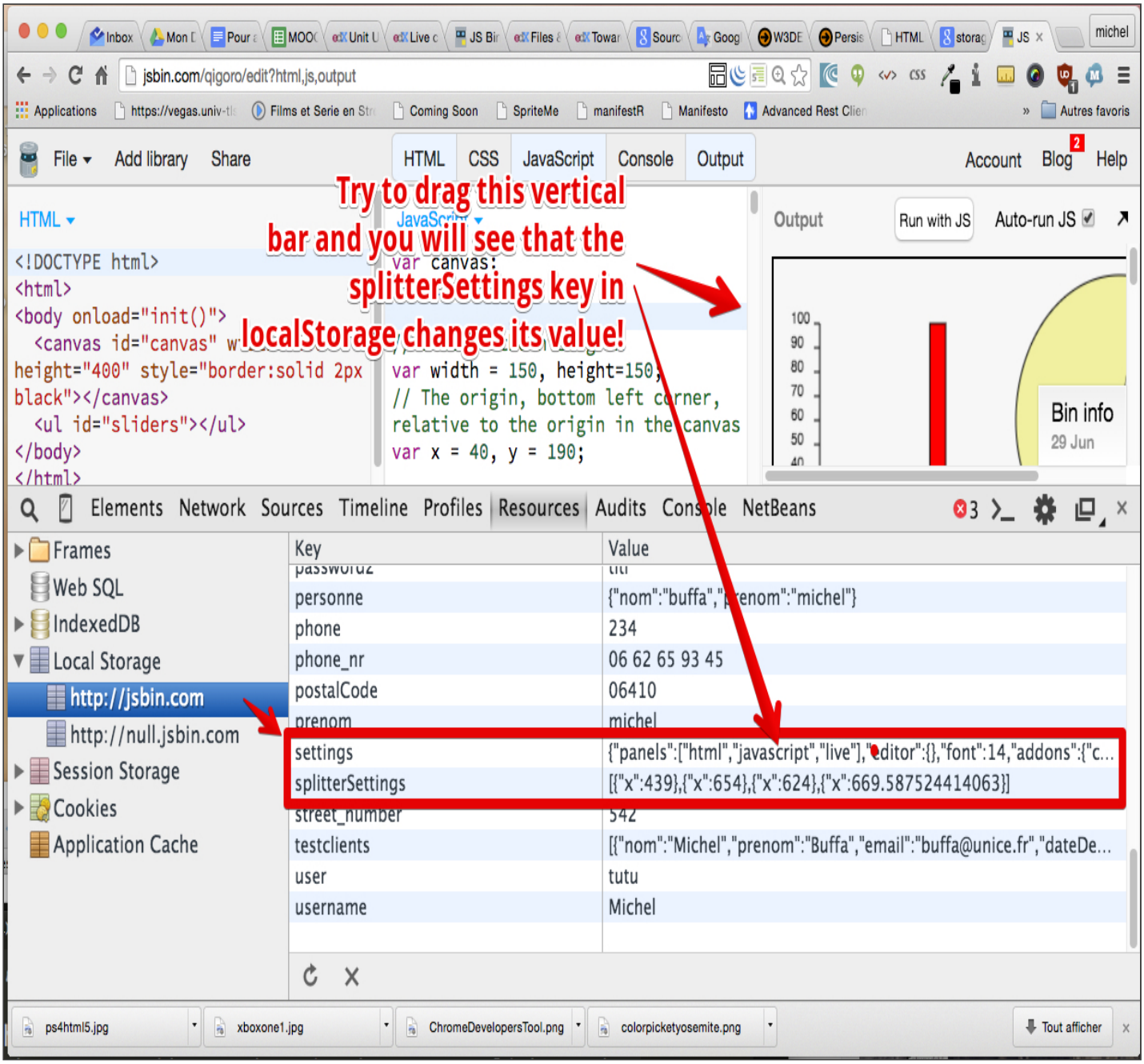
Practical example 2: save/restore user's preferences

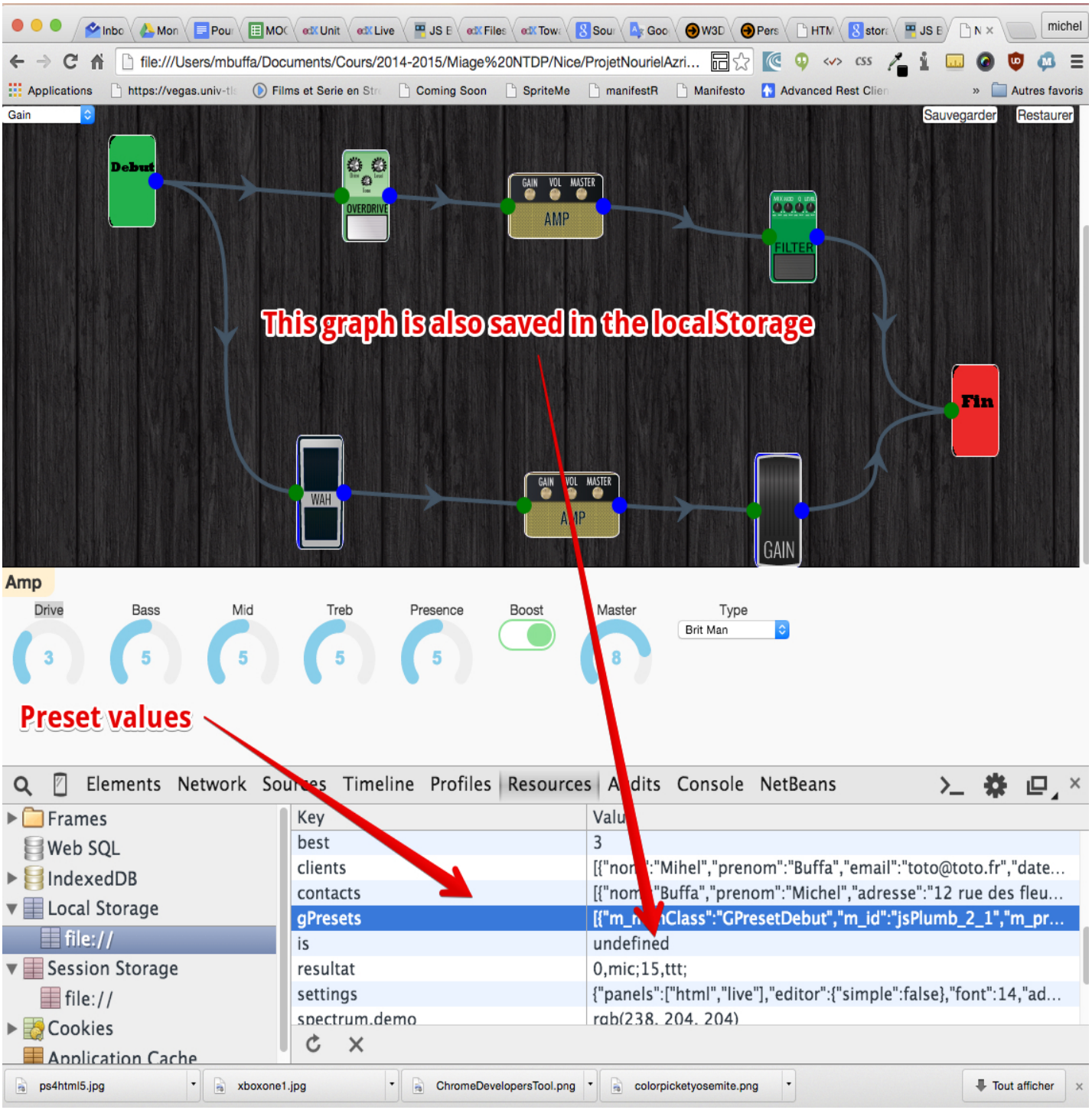
INTRODUCTION

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:



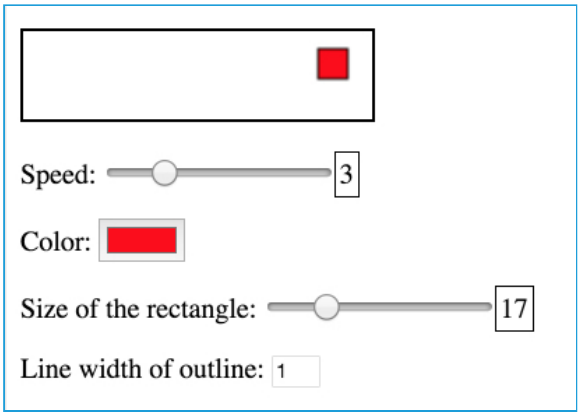
In this way, the next time you come back to JS Bin, "it will remember your last settings".

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:



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
   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() {
 // 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;

   // 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);
}
```