

Practical example 3: example 1 on steroids

TRY THE EXAMPLE!

[Online example at JS Bin](#)

This time, using the `setItem` and `getItem` method we saw earlier in the course, we could write some generic functions for saving/restoring input fields' content, without having advance knowledge about the number of fields in the form, their types, their ids, etc.

Furthermore, we removed all input listeners in the HTML, making it cleaner (no more `oninput="localStorage.firstName = this.value; '...)"`

DEFINE LISTENERS + RESTORE OLD VALUES AFTER THE PAGE IS LOADED, USE GENERIC FUNCTIONS

We start writing an `init()` function that is called when the page is loaded. This function will:

1. Define `input` listeners for all input fields
2. Restore the last saved value for each input field, if present.

Source code:

```
// Called when the page is loaded
window.onload = init;
```

Enter values, then reload the page at any time: the form content is restored automatically.

Reusable functions for saving/restoring input field content are used.

Check also "localStorage" in the devtools.

Personal informations

First name:

Last name:

Email:

Age:

Birth date:

```

function init() {
    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 form content with previously saved values
    restoreFormContent();
}
10.

```

And here is the `addInputListener(inputField)` function. It takes an input field as parameter and attaches an `oninput` listener to it, that will save the field's content each time a value is entered. The key will be the id of the input field (line 3):

```

function addInputListener(inputField) {
    inputField.addEventListener('input',function(event) {
        localStorage.setItem(inputField.id,inputField.value);
    }, false);
}

```

Note that at line 2 we use `addEventListener` (that is not using the `oninput` property here). `addEventListener` will not replace existing `oninput` definitions and keep all existing listeners unchanged.

RESTORE ALL INPUT FIELDS' CONTENT USING A GENERIC FUNCTION

We have seen how to save all input fields' content on the fly. Now, let's see how we can restore saved values and update the form. This is done using the function `restoreFormContent()`:

```

function restoreFormContent() {

```

```

    console.log("restoring form content from localStorage");
    // get the list of all input elements in the form

    var listOfInputsInForm =document.querySelectorAll("input");
    // For each input element,
    // - get its id (that is also the key for it's saved
content
    // in the localStorage)
10. // - get the value associated with the id/key in the local
    // storage
    // - If the value is not undefined, restore the value
    // of the input field
    for(var i= 0; i <listOfInputsInForm.length; i++) {
        var fieldToRestore =listOfInputsInForm[i];
        var id = fieldToRestore.id;
        var savedValue =localStorage.getItem(id);
        if(savedValue !== undefined) {
20.     fieldToRestore.value =savedValue;
        }
    }
}

```

In this function, we first get the list of input fields (line 5), then iterate on it (line 14). For each input field, we get its `id`, which value is the key in `localStorage` for the previous data saved for this field (lines 15-16). Then if the value is not undefined, we restore it by setting the value of the input field (lines 19-20).

THESE GENERIC FUNCTIONS CAN BE USED IN MANY DIFFERENT PROJECTS

Indeed, if you look carefully, you will see that these functions are really useful. You may easily embed them in your own projects, or perhaps adapt them for a particular need (i.e. for saving input type="checkboxes" that work a bit differently), etc. Later in the course, we will show how to reuse them with another example: the animated red rectangle.