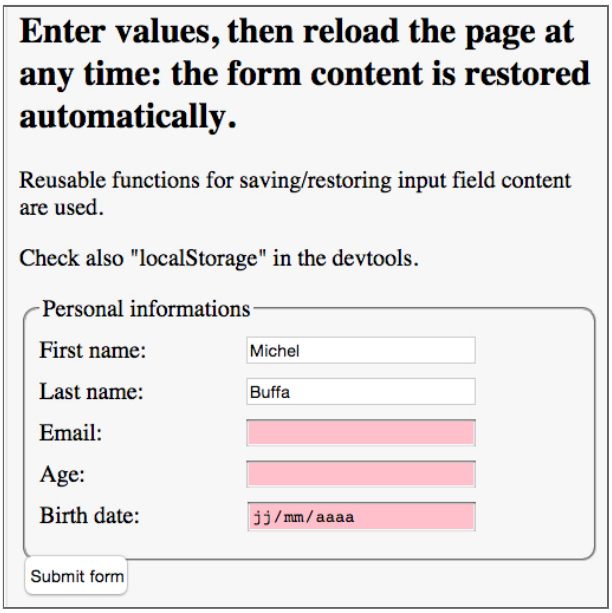
Practical example 3: example 1 on steroids

TRY THE EXAMPLE!

Online example at JS Bin



This time, using the`setItem` and `getItem`method we saw in the previous page of the course, we could write some generic functions for saving/restoring input fields's content, without knowing in advance 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 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;
```

```
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();
}
```

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). a`ddEventListener` will not replace existing `oninput` definitions and keep all existing listeners unchanged.

RESTORE ALL INPUT FIELDS' CONTENT USING A GENERIC FUNCTION

We saw how 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 by 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)
  // - 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) {
      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 FUNCTION CAN BE USED IN MANY DIFFERENT PROJECTS

Indeed, if you look carefully, you will see that these functions are really useful. You may embed them easily in your own projects, maybe adapt them to some particular needs (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.