Storing more than strings? Use JSON!

INTRODUCTION

Storing strings is all well and good, but it quickly becomes limiting: you may want to store more complex data with at least a modicum of structure.

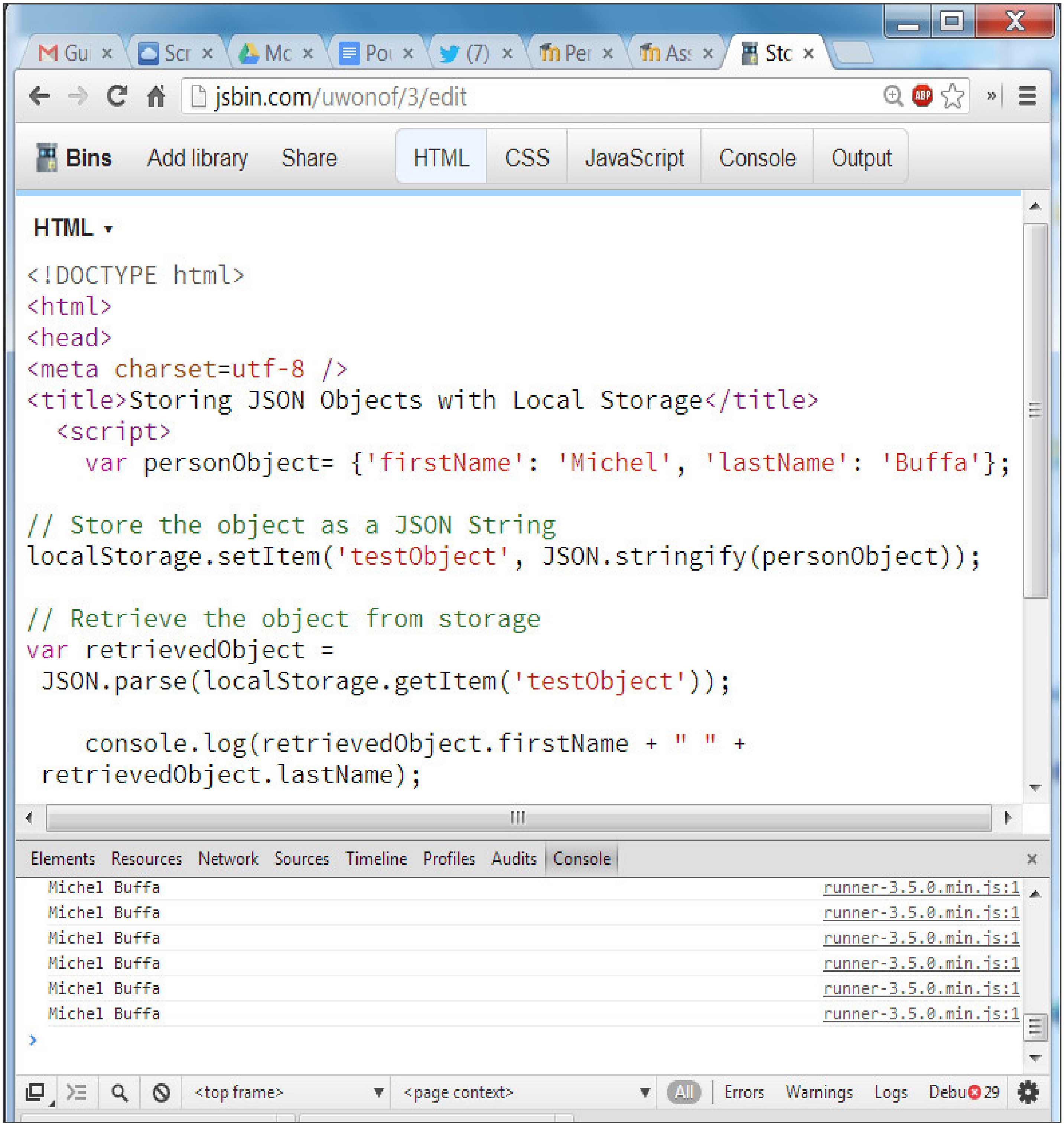
There are some simple approaches, such as creating your own minimal record format (e.g. a string with fields separated with a given character, using join() on store and split() upon retrieval) or using multiple keys (e.g.post_17_title, post_17_content, post_17_author, etc.). But these are really hacks. Thankfully, there's a better way, JSON.`stringify()` and JSON.`parse()` methods.

JSON provides a great way of encoding and decoding data that is a really good match for JavaScript. You have to be careful not to use circular data structures or non-serializable objects, but in the vast majority of cases, plugging JSON support into your local store is straightforward.

TYPICAL USAGE

```
locaStorage.key = JSON.stringify(object); // or...
localStorage.setItem(key,JSON.stringify(object));
```

Let's try a simple toy example (online at JS Bin). The example below saves a JavaScript object in JSON, then restores it and checks that the object properties are still there!



```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>Storing JSON Objects with Local Storage</title>
  <script>
    var personObject= {'firstName': 'Michel', 'lastName': 'Buffa'};

// Store the object as a JSON String
localStorage.setItem('testObject', JSON.stringify(personObject));

// Retrieve the object from storage
var retrievedObject =
 JSON.parse(localStorage.getItem('testObject'));

    console.log(retrievedObject.firstName + " " +
 retrievedObject.lastName);
```

```
Michel Buffa                                      runner-3.5.0.min.js:1
Michel Buffa                                      runner-3.5.0.min.js:1
Michel Buffa                                      runner-3.5.0.min.js:1
Michel Buffa                                      runner-3.5.0.min.js:1
Michel Buffa                                      runner-3.5.0.min.js:1
Michel Buffa                                      runner-3.5.0.min.js:1
>
```

Source code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
```

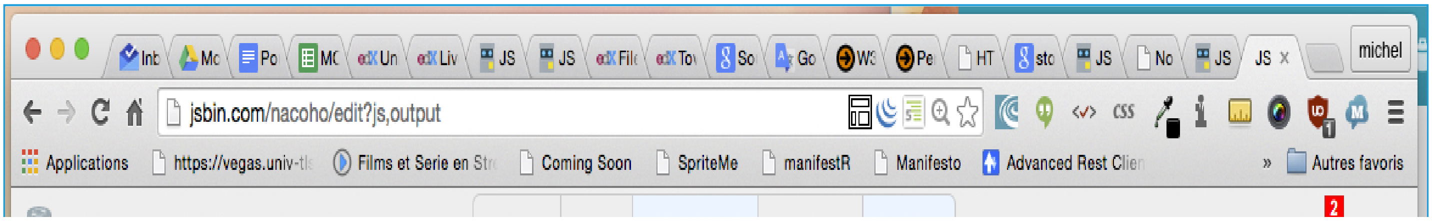
```
     <title>Storing JSON Objects with Local Storage</title>
      <script>

       var personObject= {'firstName':'Michel', 'lastName': 'Buffa'};
         // Store the object as a JSON String
10.

       localStorage.setItem('testObject',JSON.stringify(personObject));
         // Retrieve the object from storage

       var retrievedObject =JSON.parse(localStorage.getItem('testObject'));
         console.log(retrievedObject.firstName+ "
     " + retrievedObject.lastName);
        // then you can use retrievedObject.firstName,
     retrievedObject.lastName...
      </script>
     </head>
20.  <body>
     </body>
     </html>
```

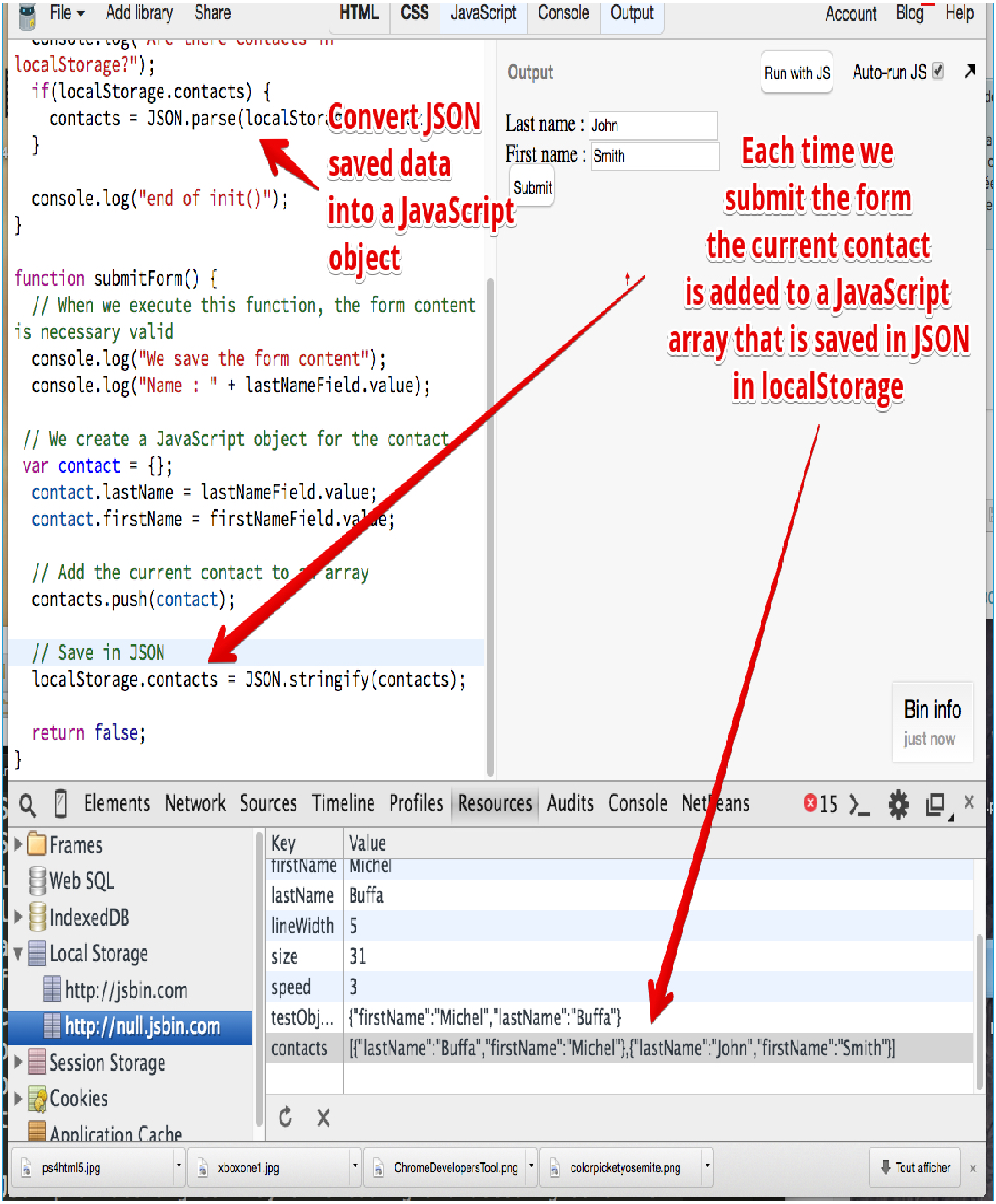
Explanations:

- **Line 7:** we built a JavaScript object that contains a person.
- **Line 10:** we store it in `localStorage` as a JSON string object, with a key equal to `testObject`.
- **Line 13:** we restore it from `localStorage` as a string, and the `JSON.parse` methods turns it back into a JavaScript object.
- **Line 15:** we print the values of the object properties.

MORE COMPLETE EXAMPLE THAT SHOWS HOW WE CAN SAVE A FORM'S CONTENT IN JSON

Online example on JS Bin that saves in localStorage an array of contacts in JSON





```
  console.log("Are there contacts in
localStorage?");
  if(localStorage.contacts) {
    contacts = JSON.parse(localStor
  }

  console.log("end of init()");
}

function submitForm() {
  // When we execute this function, the form content
is necessary valid
  console.log("We save the form content");
  console.log("Name : " + lastNameField.value);

 // We create a JavaScript object for the contact
 var contact = {};
  contact.lastName = lastNameField.value;
  contact.firstName = firstNameField.value;

  // Add the current contact to an array
  contacts.push(contact);

  // Save in JSON
  localStorage.contacts = JSON.stringify(contacts);

  return false;
}
```

Convert JSON saved data into a JavaScript object

Last name : John

First name : Smith

Submit

Each time we submit the form the current contact is added to a JavaScript array that is saved in JSON in localStorage

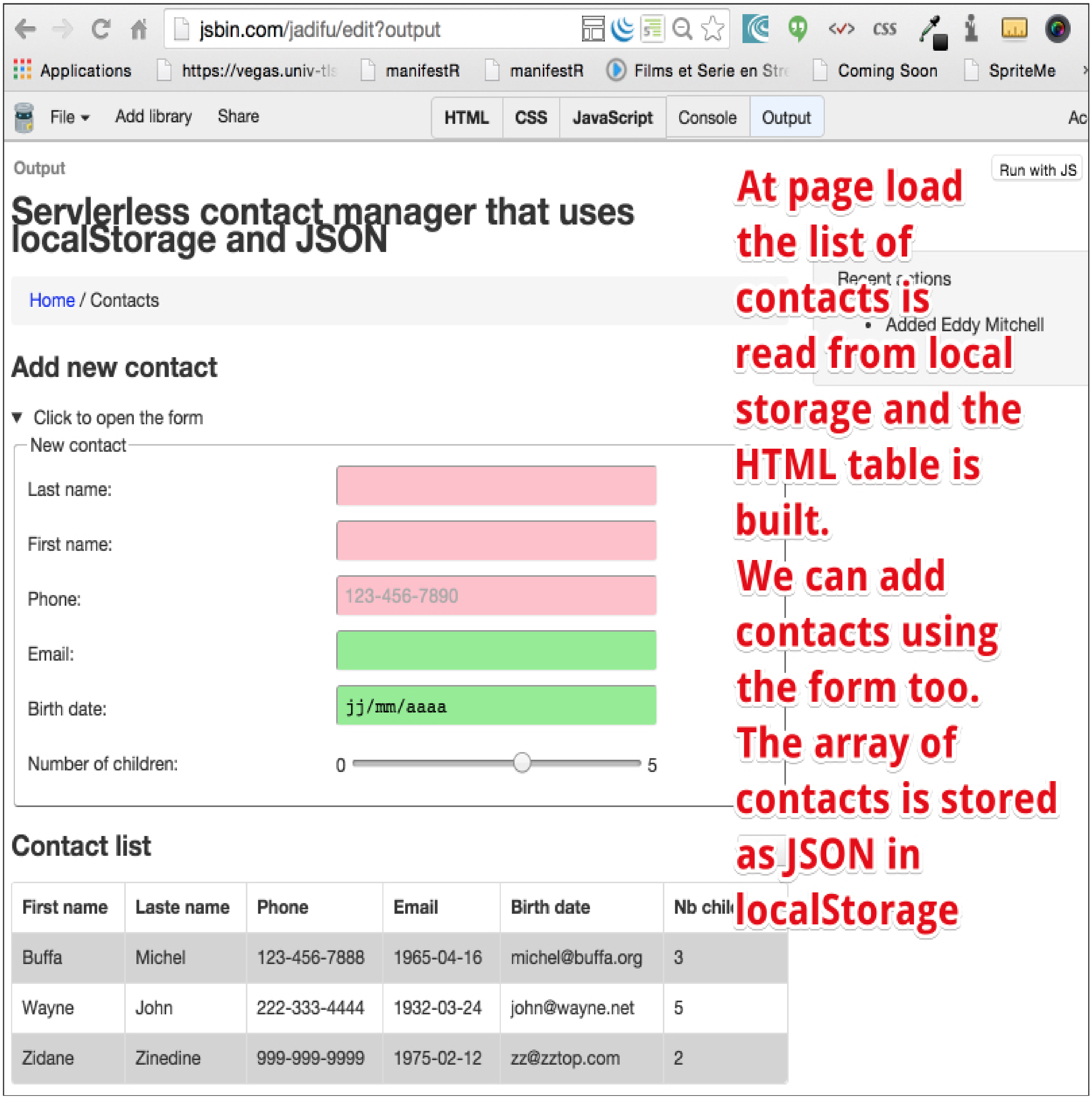
Key	Value
firstName	Michel
lastName	Buffa
lineWidth	5
size	31
speed	3
testObj...	{"firstName":"Michel","lastName":"Buffa"}
contacts	[{"lastName":"Buffa","firstName":"Michel"},{"lastName":"John","firstName":"Smith"}]

MORE COMPLETE EXAMPLE: A FORM AND A TABLE THAT DISPLAYS THE

CONTACTS STORED IN LOCALSTORAGE

Example on JS Bin (uses summary/details, so use a browser that supports it or add a polyfill, as seen in Week 1).

Add contacts using the form, see how the HTML table is updated. Try to reload the page: data are persisted in localStorage.



Servlerless contact manager that uses localStorage and JSON

Home / Contacts

Recent actions

- Added Eddy Mitchell

Add new contact

▼ Click to open the form

New contact

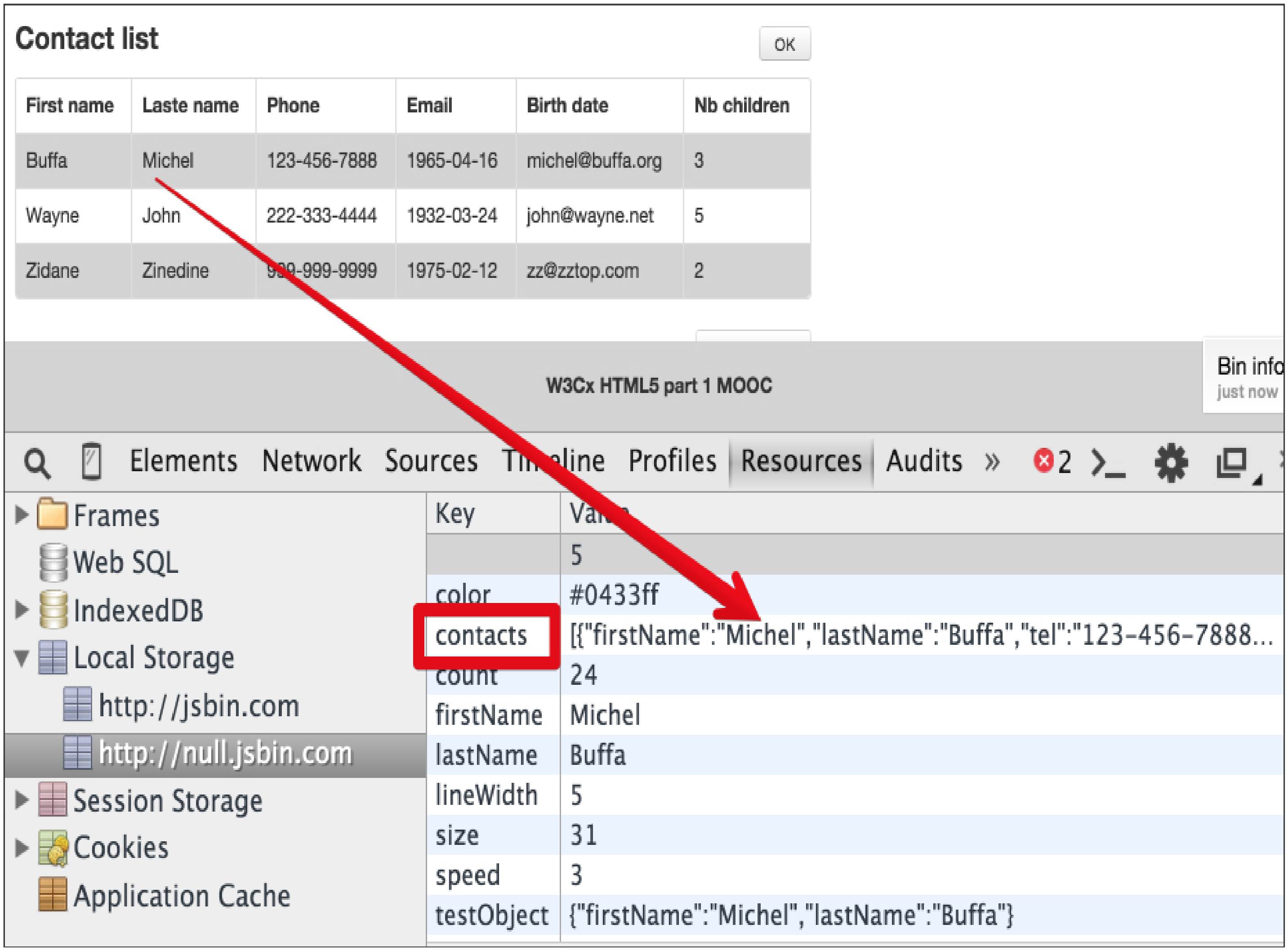
Last name:	
First name:	
Phone:	123-456-7890
Email:	
Birth date:	jj/mm/aaaa
Number of children:	0 5

Contact list

First name	Laste name	Phone	Email	Birth date	Nb chil
Buffa	Michel	123-456-7888	1965-04-16	michel@buffa.org	3
Wayne	John	222-333-4444	1932-03-24	john@wayne.net	5
Zidane	Zinedine	999-999-9999	1975-02-12	zz@zztop.com	2

At page load the list of contacts is read from local storage and the HTML table is built.
We can add contacts using the form too.
The array of contacts is stored as JSON in localStorage

Examine the localStorage:



Contact list

First name	Laste name	Phone	Email	Birth date	Nb children
Buffa	Michel	123-456-7888	1965-04-16	michel@buffa.org	3
Wayne	John	222-333-4444	1932-03-24	john@wayne.net	5
Zidane	Zinedine	999-999-9999	1975-02-12	zz@zztop.com	2

Key	Value
	5
color	#0433ff
contacts	[{"firstName":"Michel","lastName":"Buffa","tel":"123-456-7888...
count	24
firstName	Michel
lastName	Buffa
lineWidth	5
size	31
speed	3
testObject	{"firstName":"Michel","lastName":"Buffa"}

The source code for this example is a bit long, and we suggest that you examine it in the JS Bin tool. We extensively commented it. It uses:

- Well structured page with the new elements seen during Week 1 (section, article, nav, aside, etc.)
- HTML5 form elements with builtin and custom validation (the date cannot be in the past, the firstName and lastName fields do not accept &, #, ! or $ characters),
- localStorage for saving / restoring an array of contacts in JSON
- It shows how to use the DOM API for dynamically updating the page content (build the HTML table from the array of contacts, add a new line when a new contact is submitted, etc.)