



This article explains **how to automatically save user input**, and how to **reload the saved input values** within a Web Runtime widget.

Contents

- [1 Description](#)
- [2 How to](#)
 - ◆ [2.1 The HTML code](#)
 - ◆ [2.2 Saving the user input](#)
 - ◆ [2.3 Loading the saved data](#)
- [3 Downloads](#)

Description

User input on mobile devices is slower and more difficult than on desktop devices, due to different factors, as:

- **users in mobility**, that are usually performing other actions (e.g.: walking) while entering data
- **different input methods** (e.g.: small keyboards, touchscreen keyboards)
- **smaller display resolutions**

So, mobile applications should always provide a **functionality to allow the user not to lose the entered data**, when some kind of interruptions happen, as:

- **the application is closed** (both for a **user interaction**, or for **another application's interruption** causing the current one to close)
- **device turns off** (e.g.: due to empty battery)
- **user wants to go to another screen** of the same application, and could continue to input data later

In these and other cases it's a good practice to **allow the user to continue editing the previous entered data**, when he goes back to the previous input screen. This greatly **improves the final user experience**, since it **avoids unnecessary actions** and inputs from the user, and allows a **more flexible usage of the mobile application**.

How to

This example shows how to **automatically save and load user input from 2 text fields**, without requiring any extra actions to the user. The same approach can be used with more complex forms, and with different kinds of fields.



The HTML code

For this example, a **basic SMS form is built**, with the classic user input fields:

- a **field for the receiver's phone number**
- a **field for the message body**

```
<html>
  [...]
<body onLoad="javascript:init();">
  <div id="sms_form">
    <label for="receiver">Phone Number</label>
    <input type="text" name="receiver" id="receiver" />
    <label for="message">Message</label>
    <textarea name="message" id="message"></textarea>
    <input type="submit" value="Send">
  </div>
</body>
</html>
```

Saving the user input

In order to save the user input, **different approaches** are possible:

- save it at **predefined time intervals**: this option **could cause more savings than necessary**, since there's no guarantee that the input values will change within the specified interval

Automatically_save_and_load_user_input_in_Web_Runtime_widgets

- save it when the **user press some buttons (e.g.: a "Save draft" button)**: this option require an **explicit interaction from the user**, so it is no optimal from a usability point of view
 - save it when **the value of a input field changes**: this option is both **optimal for the user**, since it has not to perform any action, and from a **performance point of view**, since the value of an input field will be actually saved only when actually changed
-

So, the last proposed approach will be used, and entered data will be saved only when actually changed. In order to be **notified of any changes of an input field**, the **onchange** event handler is used. The following **startAutosave()** method takes a field and a key as arguments, and **starts listening to the change events of that field**. When this event is fired, the **saveValue()** function is called, that performs the actual saving of the input field value:

```
function startAutosave(field, key)
{
  var changeHandler = function()
  {
    saveValue(
      field, key);
  };

  onchange = changeHandler;
}
function saveValue(field, key)
{
  widget.preferenceForKey(field.value, key);
}
```

So, the the widget's **init()** method, the **startAutosave()** method is called for both the receiver and the message inputs:

```
function init()
{
  startAutosave(widget.getElementById('receiver'), 'sms_receiver');

  startAutosave(widget.getElementById('message'), 'sms_message');
}
```

Loading the saved data

Once the user exits and reopen the widget, or simply goes back to the previous form view, the **previously saved input values have to be restored**, and this can be easily done by reading the preferences stored by the previous **saveValue()** function. The following **loadSavedValue()** method takes a field and a key as arguments and, if available, restores the saved value in the input field:

```
function loadSavedValue(field, key)
{
  var value = widget.preferenceForKey(key);

  if(value)
  {
    value = value;
  }
}
```

In order to restore the values as soon as the user enters the widget, the **loadSavedValue()** can be called in the

Automatically_save_and_load_user_input_in_Web_Runtime_widgets

widget's init() method:

```
function init()
{
    loadSavedValue(widget.getElementById('receiver'), 'sms_receiver');

    loadSavedValue(widget.getElementById('message'), 'sms_message');
}
```

Downloads

A sample widget, that shows the auto-save and auto-load functionalities, is available for download here:

[Media:TextFieldAutoSaveWidget.zip](#)