Using Swipe Events with dynamic content in Nokia Asha Web Apps

Swipe events can only be attached at a <script> if this tag is the last element of the <body>. This article explains how to bypass this limitation to add swipe events to dynamically created elements anywhere in the code.

08 Sep 2013

Note: This article was a winner in the Asha Touch Competition 2012Q3. It has since been updated to work in the Asha 501.

Introduction

The Asha Web App [1] states that the code that adds handlers to Swipe Events must be in the last tag of `<body>`. The events simply don't work when they are bound in other places of the code. This implies that it is impossible, or at least not easy, to add swipe events to elements created dynamically. This article presents a trick to bypass this apparent limitation and add swipe events to any element in the page at any time: dynamically creating the final `<script>` element using DOM manipulation.

As an example, we build a "PhoneBook" application. The user swipes right on top of a contact to get details and swipes left on the details screen to go back to the contact list. To illustrate the creation of new elements, we add a button that adds more items to the phonebook and updates all the event handlers.

Swipe events working with dynamically created content.

Adding a basic swipe event

Create a new project in Nokia Web Tools with the basic template and open **index.html**. Put the following in the `<body>` tag.

```
<div id="phonebook_panel">
 <ul id="phonebook">
 </ul>
</div>
<div id="details_panel"  style="display: none;">
 <div id="details">></div>
 <br></br><br></br>
 <i>Swipe Right to go back to the contact list.</i>
</div>
```

Then add this script to the `<head>` element. The code basically defines an object that contains names and phones and populates the *phonebook* list. It builds a HTML string with <li> elements with ids `contact#` and assigns it to the `innerHTML` property of the `#phonebook` element. Notice that the `addSwipe*` Listener functions are called inside our showContent code. This works as long as our function is called at a <script> tag right before </body>. We also need to be extra careful with the quotes in the mwl commands, as they will give a parse error if placed incorrectly.

```
<script type="text/javascript">
 var phonebook = {
  names: ['AAA', 'BBB', 'CCC'],
  phones: ['+44112453', '+234234', '+1232344']
 };

 function showContent() {
  var pb = document.getElementById("phonebook");
  pb.innerHTML = "";
  var details = document.getElementById("details");
  details.innerHTML = "";
  for (var i = 0; i < phonebook.phones.length; i++) {
```


(C) Copyright Nokia 2013. All rights reserved.

```
    pb.innerHTML += "<li id=\"contact" + i + "\">" + phonebook.names[i] + "</li>";

                        var html = "<div id='details" + i + "'> ";
     html += phonebook.names[i] + "<br>";
     html +=  phonebook.phones[i] + "";
     html += "</div>";
    details.innerHTML += html;
   }

   mwl.addSwipeLeftListener("#contact0", "mwl.hide('#phonebook_panel');mwl.show('#details_panel');mwl.hide('#details1');mwl.hide('#de
   mwl.addSwipeLeftListener("#contact1", "mwl.hide('#phonebook_panel');mwl.show('#details_panel');mwl.hide('#details0');mwl.hide('#de
   mwl.addSwipeLeftListener("#contact2", "mwl.hide('#phonebook_panel');mwl.show('#details_panel');mwl.hide('#details0');mwl.hide('#de

   mwl.addSwipeRightListener("#details_panel", "mwl.hide('#details_panel'); mwl.show('#phonebook_panel');");

  }
 </script>
```

Finally, add this `<script>` tag before the `</body>` tag to call our function.

```
<script type="text/javascript">
 showContent();
</script>
```

Run the app to test the swipe events.

Adding items Dynamically

Add a button after the phone list:

```
<div class="ui-content">
       <div id="phonebook_panel">
  <ul id="phonebook">
  </ul>
  <input type="button" name="button1" class="ui-button" value="More" onclick="addMore();" />
 </div>
 <div id="details_panel"  style="display: none;">
  <div id="details"></div>
  <br></br><br></br>
  <i>Swipe Right to go back to the contact list.</i>
 </div>
</div>
```

Then add the following two functions to our `<head>` script. `randomEntry()` adds a new entry to the phone book `refresh()` shows the updated list.

```
function randomEntry() {
 var letters = ['a', 'b', 'c', 'd', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'x', 'y', '
 var name = '';
 for (var i = 0; i < 10; i++) {
  name += letters[Math.floor(Math.random() * letters.length)]
 }
 var phone = 0;
 for (var i = 0; i < 10; i++) {
  phone = Math.floor(Math.random() * 9) + phone * 10;
 }
 phonebook.names.push(name);
 phonebook.phones.push(phone);
}

function addMore() {
 randomEntry();
 showContent();
}
```

If we run the program now, clicking `addMore()` will add more elements to our list. However, swipe events won't work on them. We'll fix this on the next section.

Binding everything together

Now comes the tricky part: the swipe events can only be registered in a `<script>` tag positioned at the end of the `<body>` element. They have no effect when used in other parts of the page. The solution is to dynamically generate the `<script>` tag every time we update the list. To do this we'll use DOM manipulation functions.

The `document.createElement` function creates a HTML element of the type of its first parameter. This element is not added to the document. To add the created element we use the `appendChild()` method of any HTML element. The most common way to obtain HTML elements is using the `document.getElementById`, but since we need the `<body>` tag, we use can use the `document.body` variable. It is important to notice that different HTML elements have different methods and attributes.

Add the following function to the `<head>` element.

```
function updateScript() {
 var src = document.createElement("script");
 src.id = "swipes";
 src.type  = "text/javascript";
 var hide_all = "mwl.hide('#phonebook_panel');";
 for (i = 0; i < phonebook.names.length; i++) {
  hide_all += "mwl.hide('#details" + i + "');"
 }
 for (i = 0; i < phonebook.names.length; i++) {
  src.text += "mwl.addSwipeLeftListener(\"#contact" + i + "\", \"" + hide_all +"mwl.show('#details_panel');mwl.show('#details" + i +
```


(C) Copyright Nokia 2013. All rights reserved.

```
  }
  src.text += "mwl.addSwipeRightListener(\"#details_panel\", \"mwl.hide('#details_panel'); mwl.show('#phonebook_panel');\");";
  document.body.appendChild(src);
 }
```

This function creates a new `<script>` tag with statements that add the event listeners to all elements in phonebook. Since the `appendChild()` adds the created element to the last position of the `<body>` element, the `<script>` tag is the last element and is processed correctly.

The `hide_all` variable contains the MWL code to hide all with contact details. Then we just call *mwl.show* on the current contact.

To finish, update the `showContent()` as follows:

```
function showContent() {
 var pb = document.getElementById("phonebook");
 pb.innerHTML = "";
 var details = document.getElementById("details");
 details.innerHTML = "";
 for (var i = 0; i < phonebook.phones.length; i++) {
  pb.innerHTML += "<li id=\"contact" + i + "\">" + phonebook.names[i] + "</li>";

                var html = "<div id='details" + i + "'> ";
   html += phonebook.names[i] + "<br>";
   html +=  phonebook.phones[i] + "";
   html += "</div>";
  details.innerHTML += html;
 }
 updateScript();
}
```

Finally, remove the old event code from the last `<script>` tag. Now the events should be working on every entry of the list. Run the app to check that the events are processed correctly even after the creation of new elements.

Conclusion

This article showed how to use swipe events with dynamic lists. It uses DOM manipulation to create a `<script>` tag as the last element of the `<body>` to be able to call the swipe event listeners. Check the attached code example to see the complete Web App.

(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.