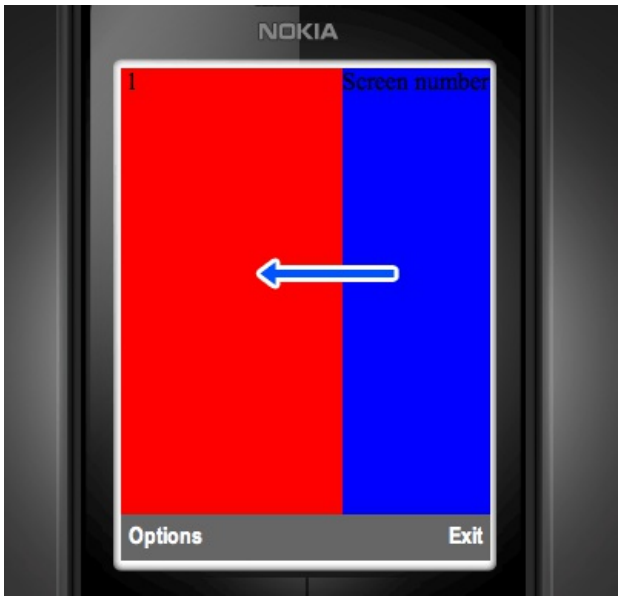
Custom screen transitions in Symbian Web Runtime

This article explains how to implement custom screen transitions in Symbian Web Runtime. A screen is a HTML element that covers the whole widget's interface. Screen Transitions are useful to give visual feedback to the user about the movement to another view of an application.



Screen Transitions design pattern is explained in detail in this Nokia Developer Wiki article: Mobile Design Patterns: Screen Transitions. The transitions implemented in this article are described in the "Screen Push in and out Transition" section.

WRT widget: single canvas

WRT widgets, unlike websites, can contain a single HTML page. This means that, if you want to implement different views or screens within a WRT widget, the only way to do it is by manipulating its DOM structure.

Usually, there are 2 different approaches to this task:

- write all your Widget HTML structure statically, hiding and showing elements by using JavaScript and CSS
- write only the basic HTML structure statically, and then dynamically build, append and remove DOM elements by using JavaScript

Usually, the 2 approaches are mixed within the same Widget.

WRT inbuilt transitions

Web Runtime already supports transitions between views, with the `prepareForTransition()` and `performTransition()` methods. Current WRT versions have support for "fade" transitions.

For more details about the standard WRT transitions, check out these pages:

- prepareForTransition()
- performTransition()

Custom screen transitions

The transitions presented in this article are classical "slide" horizontal transitions from a screen to another, performed from left to right or from right to left.

Basic HTML structure

The sample DOM structure used in this article to perform custom transitions consists of 2 DIV elements, representing the 2 screens.

```
<body>

 <div class="screen" id="screen_1" onclick="gotoScreen2()">
```


(C) Copyright Nokia 2014. All rights reserved.

```
  Screen number 1
 </div>

 <div class="screen" id="screen_2" onclick="gotoScreen1()" style="display: none;">
  Screen number 2
 </div>

</body>
```

Only one screen (screen_1) is initially visible, while the other one (screen_2) is hidden through the inline CSS styling, with the display: none rule.

CSS styling

In order to allow the single screens to cover the whole widget interface, some CSS styling is required.

```
body, html {
 margin: 0;
 padding: 0;
 width: 100%;
 height: 100%;
}

body {
 position: relative;
}

.screen {
 position: absolute;
 width: 100%;
 height: 100%;
}

#screen_1 {
 background: red;
}

#screen_2 {
 background: blue;
}
```

Absolute positioning is used in order to allow exact positioning of single screens, and so to perform the sliding transitions.

Performing the transition with MooTools

MooTools is a JavaScript framework, freely available here: http://mootools.net/.

So, let's see how the previously defined `gotoScreen1()` and `gotoScreen2()` methods can be implemented by using the MooTools framework. Let's start with `gotoScreen2()`, that must bring the user from screen_1 to screen_2. In order to do this, these steps have to be performed:

- screen_2 has to be placed in its initial transition position
- then, it has to be shown, modifying its CSS "display" property
- finally, both screen_1 and screen_2 have to be animated, to perform the transition

This is done by the following code:

```
function gotoScreen2()
{
```


(C) Copyright Nokia 2014. All rights reserved.

```
 document.getElementById('screen_2').style.left = screen.availWidth + 'px';

 document.getElementById('screen_2').style.display = '';

 new Fx.Tween(document.getElementById('screen_1')).start('left', '0px', ( -
screen.availWidth) + 'px');

 new Fx.Tween(document.getElementById('screen_2')).start('left', '0px');
}
```

Similarly, the `gotoScreen1()` function can be implemented as follows:

```
function gotoScreen1()
{
 document.getElementById('screen_1').style.left = - screen.availWidth + 'px';

 document.getElementById('screen_1').style.display = '';

 new Fx.Tween(document.getElementById('screen_2')).start('left', '0px',
screen.availWidth + 'px');

 new Fx.Tween(document.getElementById('screen_1')).start('left', '0px');
}
```

Implementing the transition from scratch: the SlidingScreenTransition object

This section explains 'how to implement the sliding transition from scratch, without using any third party JavaScript framework. In order to build reusable code, a JavaScript will be defined: SlidingScreenTransition, that will allow to easily perform sliding screen transitions in Symbian Web Runtime.

SlidingScreenTransition costructor

In order to perform a sliding transition, 3 parameters are usually needed:

- the DOM element to be "pushed out"
- the DOM element to be "pulled in"
- the direction of the movement

So, the following SlidingScreenTransition constructor will accept these 3 parameters:

```
function SlidingScreenTransition(currentScreen, nextScreen, direction)
{
 /* DOM element to be "pushed out" */
 this.currentScreen = currentScreen;

 /* DOM element to be "pulled in" */
 this.nextScreen = nextScreen;

 /* direction of the sliding transition */
 this.direction = direction;
}
```

Also, some useful properties are defined:

```
/* constant value for right-to-left transitions */
SlidingScreenTransition.DIRECTION_LEFT = 1;
```


(C) Copyright Nokia 2014. All rights reserved.

```
/* constant value for left-to-right transitions */
SlidingScreenTransition.DIRECTION_RIGHT = -1;

/* total number of steps of the sliding transition */
SlidingScreenTransition.TRANSITION_STEPS = 5;
```

Starting the sliding transition

The transition is performed by scrolling both the DOM elements in the same direction, for a number of steps equal to the `SlidingScreenTransition.TRANSITION_STEPS` property value.

Before actually starting the animation, the nextScreen DOM element is positioned, according to the transition direction, and then displayed, by modifying its CSS display property.

```
SlidingScreenTransition.prototype.start = function()
{
 var self = this;

 this.nextScreen.style.top = '0px';
 this.nextScreen.style.left = (this.direction * screen.availWidth) + 'px';
 this.nextScreen.style.display = '';

 this.transitionStep = 0;

 this.transitionInterval = setInterval(
  function()
  {
   self.doTransitionStep();
  },
  100
 );
}
```

In order to stop the transition, it's enough to clear the interval started from the `start()` method:

```
SlidingScreenTransition.prototype.stop = function()
{
 clearInterval(this.transitionInterval);
}
```

The single transition step

After the transition has been started, the following `doTransitionStep()` method is called for each transition step that has to be performed:

```
SlidingScreenTransition.prototype.doTransitionStep = function()
{
 this.transitionStep++;

 if(this.transitionStep <= SlidingScreenTransition.TRANSITION_STEPS)
 {
  this.nextScreen.style.left =
   (screen.availWidth *
   (SlidingScreenTransition.TRANSITION_STEPS - this.transitionStep) *
   this.direction / SlidingScreenTransition.TRANSITION_STEPS)
```


(C) Copyright Nokia 2014. All rights reserved.

```
    + 'px';

  this.currentScreen.style.left =
   (- screen.availWidth * this.transitionStep * this.direction /
   SlidingScreenTransition.TRANSITION_STEPS)
   + 'px';
 }
 else
 {
  this.stop();

  this.currentScreen.style.display = 'none';
 }
}
```

When the transition ends (so, `this.transitionStep > SlidingScreenTransition.TRANSITION_STEPS`) the transition itself is stopped, and the pulled out DOM element is hidden by modifying its CSS display property.

Performing the transitions

It is now possible to use the `SlidingScreenTransition` object to perform the needed transitions. So, these are the `gotoScreen1()` and `gotoScreen2()` versions that use the new custom `SlidingScreenTransition` object:

```
function gotoScreen1()
{
 new SlidingScreenTransition(
  document.getElementById('screen_2'),
  document.getElementById('screen_1'),
  SlidingScreenTransition.DIRECTION_RIGHT
 ).start();
}

function gotoScreen2()
{
 new SlidingScreenTransition(
  document.getElementById('screen_1'),
  document.getElementById('screen_2'),
  SlidingScreenTransition.DIRECTION_LEFT
 ).start();
}
```

So, as seen in the MooTools version, clicking on a screen will bring the user to the other screen, with a nice sliding transition effect.

Downloads

- A sample Web Runtime widget showing custom transitions: Media:CustomScreenTransitionsWidget.zip
- Source code of the SlidingScreenTransition JavaScript object: Media:Wrt SlidingScreenTransition.zip

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.