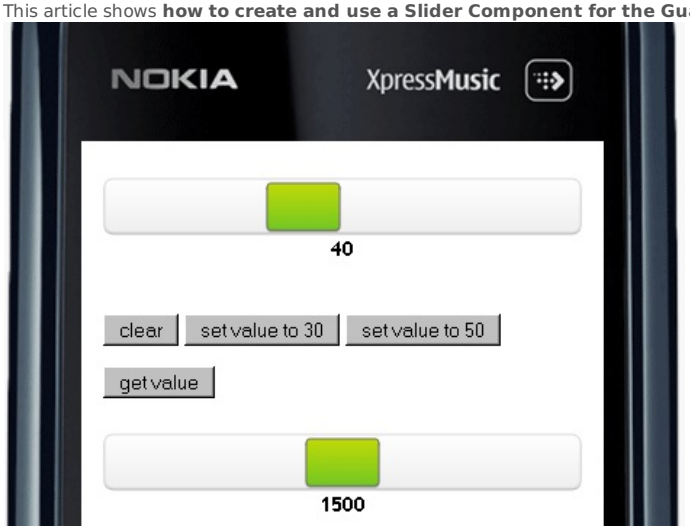
Creating a Slider Component for Guarana UI

This article shows **how to create and use a Slider Component for the Guarana UI Library**.



The Slider Component is visible on the **Guarana UI online Components Browser**: Guarana UI Slider Component.

Introduction

The Guarana UI Library is a **jQuery-Based UI Library for Nokia WRT**, that provides a full set of Components ready to be used within Web Runtime widgets.

Guarana UI also allows to build new UI Components, and integrate them with the Guarana UI structure. Before proceeding with this article, it is so necessary to view the general process behind the creation of a new Guarana UI Component, described here: How to create a new Component for the Guarana UI Library.

Building the empty Component

First thing to do, is to **create the Component skeleton**. This process is described in the Setup the Component source files section of the How-to article.

The following steps must be performed:

- **create an empty slider.js file** in the "<GUARANA_BASE_PATH>/src" folder
- **create an empty slider.css file** in the "<GUARANA_BASE_PATH>/themes/nokia/base/" folder
- add the **dependency declarations** to both the "<GUARANA_BASE_PATH>/src/defaults.js" and "<GUARANA_BASE_PATH>/build/build/js/defaults.js" (for the version 1.1 "<GUARANA_BASE_PATH>/build/tools/js/defaults.js") files

```
/*
 * Slider
 */

'slider': {
 getPath: function() { return NOKIA_PATH_JAVASCRIPT + 'slider.js'; },
 requires: ['widget', 'slider-css']
},

'slider-css': {
 getPath: function() { return NOKIA_PATH_STYLE_ROOT + 'slider.css'; },
 type: 'css',
 requires: ['base-css']
}
```

- **launch the Guarana UI build process**

Now, the skeleton files are ready to be used, so it's possible to implement the new component.

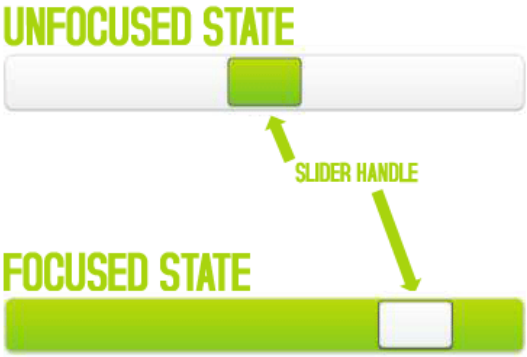
Writing the Component

In order to write the component, let's **create an empty WRT project**. First thing to do, is to move the "<GUARANA_BASE_PATH>/build/Guarana/lib/" and "<GUARANA_BASE_PATH>/build/Guarana/themes/" to the just created empty WRT project.

Define the Component's layout

The Slider component will have the following **look-and-feel** (actually, the same of the Slider component for Mobile Web Templates):

(C) Copyright Nokia 2013. All rights reserved.



Since this **component uses images**, it is necessary to **place them in the right folder**, in order for Guarana UI to correctly recognize them. Component's images must be placed inside the **"themes/nokia/base/images/" folder of the WRT project**.

The JavaScript code

This section shows and explains the Slider JavaScript code.

Initialization

The **most important** part of the component, and the **first to be executed**, is the **initialize() function**. It must take care of initializing the Component's properties, by using default options when explicit ones are not used, and then of creating the slider:

```
initialize: function(options) {
        var instance = this;

 var defaults = {
  minValue: 0,
  maxValue: 100,
  increment: 1,
  value: 0,
  readonly: false
        };

        instance.options = jQuery.extend(defaults, options);

        instance.element = jQuery( instance.options.element );

        instance._super.apply(instance, [instance.options]);

        instance.registerData("slider");

        instance._create();

 instance.minValue = instance.options.minValue;
 instance.maxValue = instance.options.maxValue;
 instance.increment = instance.options.increment;

        instance.setValue(instance.options.value);

 instance.registerVibrationOn([ ]);

 instance.fireCallback('create');
}
```

DOM creation

The **_create() function takes care of creating the Slider DOM structure**, binding the necessary events to make the Component fully interactive. Before defining the _create function, let's define which **CSS styles** will be used by the Component, by using the Styles property:

```
Styles: {
 slider: 'slider',
 sliderItem: 'slider-item',
 sliderValue: 'slider-value'
}
```

Below, it is possible to see the _create function:

```
_create: function() {
 var instance = this;
 var options = instance.options;
 var element = instance.element;
 var Styles = instance.Styles;

 element.each(function(i, el) {
  Nokia.util.addClass(el, Styles.slider);
 });

 instance.slider = Nokia.dom.parseHTML('<div class="' + Styles.sliderItem + '"></div>');
 instance.sliderItem = Nokia.dom.parseHTML('<a href="javascript:void(0)"></a>');
 instance.sliderHandle = Nokia.dom.parseHTML('<span></span>');
 instance.sliderValue = Nokia.dom.parseHTML('<div class="' + Styles.sliderValue + '"></div>');



 if (!options.readonly) {

  var clickAction = function(e) {
   instance.handleClicked(e);
   return false;
  };
  var keyAction = function(e) {
   instance.handleKeyDown(e);
   return false;
  };

  instance.sliderItem.bind('click.slider', clickAction);
```


(C) Copyright Nokia 2013. All rights reserved.

```
  instance.sliderItem.bind('keypress.slider', keyAction);

 }

 instance.sliderItem.append(instance.sliderHandle);
 instance.slider.append(instance.sliderItem);
 element.append(instance.slider);
 element.append(instance.sliderValue);

},
```

Setting/Getting Slider's value

Being fully interactive, the Component needs **functions that allow setting and getting the Slider's value**. The setting function must also take care of **updating the Component's appearance**, and this operation will be performed by the update() function in the code below:

```
update: function()
{
 var positionX = 100 * (this.value - this.minValue) / (this.maxValue - this.minValue);

 this.sliderHandle.css('background-position', positionX + '% center');
},
setValue: function(_value)
{
 this.sliderValue.text(_value);

 this.value = _value;

 this.update();
},

getValue: function(_value)
{
 return this.value;
}
```

Handling events

As the Component must be **usable with both touch and key-based navigation**, it is necessary to define two different event handlers, that will handle a specific event. The handleKeyDown() function will handle the key-based events, while the handleClicked() function will handle the touch-based ones.

```
handleKeyDown: function(e)
{
 var key = e.keyCode;

 var newValue = this.value;

 if(key == 37 && this.value > this.minValue)
  newValue = Math.max(this.minValue, this.value - this.increment);
 else if(key == 39 && this.value < this.maxValue)
  newValue = Math.min(this.maxValue, this.value + this.increment);

 if(newValue != this.value)
 {
  this.setValue(newValue);

  this.fireCallback('change', e, this.value);
 }
},
handleClicked: function(e)
{
 var mouseX = e.clientX;

 var relativeX = mouseX - this.sliderHandle.position().left;

 var newValue = this.minValue + Math.round((this.maxValue - this.minValue) * relativeX / (this.sliderHandle.width() * this.increment

 this.setValue(newValue);

 this.fireCallback('change', e, this.value);
},
```

CSS code

The **CSS code, contained in the slider.css file**, is the following one:

```
.slider {
 margin: 0;
 padding: 0;
 list-style: none;
}
.slider div.slider-value
{
 display: block;
 clear: both;
 text-align: center;
 font-weight: bold;
 margin-bottom: 10px;
}
.slider div.slider-item {
 float: left;
 width: 100%;
 margin-bottom: 2px;
 margin-top: 10px;
 line-height: 40px;
 background: url(images/sprite-button-rounded-right.png) no-repeat right top;
 text-align: center;
}

.slider div.slider-item:hover {
 background: url(images/sprite-button-rounded-right.png) no-repeat right bottom;
}

.slider div.slider-item a {
 width: 100%;
 display: block;
 padding: 0 0;
 height: 40px;
```


(C) Copyright Nokia 2013. All rights reserved.

```
 background: url(images/sprite-button-rounded-left.png) no-repeat left top;
 text-decoration: none;
 color: #333333;
}
.slider div.slider-item a:hover {
 background: url(images/sprite-button-rounded-left.png) no-repeat left bottom;
 color: #FFF;
 outline: none;
}
.slider div.slider-item a span {
 margin-left: 5px;
 margin-right: 5px;
 display: block;
 height: 40px;
 background-image: url(images/slider-off.png);
 background-repeat: no-repeat;
 background-position: 0 center;
}
.slider div.slider-item a:hover span
{
 background-image: url(images/slider-on.png);
}
```

Using the Component

The Slider can be used by following these steps:

- **Include the Guarana UI CSS dependencies** (as explained here: GuaranaUI-QuickStart):

```
<!-- Themeroller CSS -->
<link rel="stylesheet" href="themes/themeroller/default-theme/Themeroller.css" type="text/css" media="screen">
<!-- Specific Theme/Resolution CSS -->
<link rel="stylesheet" href="themes/nokia/ext-theme/default/360x640/custom.css" type="text/css" media="screen">
```

- Include the **JavaScript dependencies**:

```
<!-- jQuery file -->
<script src="/lib/jquery/jQuery.js" type="text/javascript" charset="utf-8"></script>
<!-- Guarana file -->
<script src="/lib/Guarana.js" type="text/javascript" charset="utf-8"></script>
```

- define a **root element** in the widget's HTML code:

```
<div id="slider01"></div>
```

- define the Guarana UI **path variables**:

```
NOKIA_PATH_JAVASCRIPT = 'lib/';

NOKIA_PATH_STYLE_ROOT = 'themes/nokia/base/';
```

- use the **Nokia Loader** to load the slider resources:

```
Nokia.use('slider', init);
```

- in the init() function, **initialize the Slider**:

```
var slider1 = new Nokia.Slider({
    element: '#slider01',
    value: 40,
 minValue: 0,
 maxValue: 100,
 increment: 5,
    create: function() {
        alert('slider created');
    },
    change: function(event, value) {
        alert('slider change');
    }
});
```

Finalizing the Component

Once finished, some steps have to be performed:

- **move back the slider.js file** to the "<GUARANA_BASE_PATH>/src/" folder
- **move back the slider.css file** to the "<GUARANA_BASE_PATH>/themes/nokia/base/" folder
- **move the Component images** to the "<GUARANA_BASE_PATH>/themes/nokia/base/images/" folder
- **launch the Guarana UI build process**

Done this, the Slider Component is ready to be used in all your WRT widgets.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.