Battery JavaScript component for Symbian Web Runtime

This article explains how to create and use a JavaScript Battery component in Symbian Web Runtime.

It includes the following features:

- customizable battery images
- support for both WRT 1.0 and WRT 1.1
- animation during battery charging phase

Battery component: how to use it

To use this Battery component within a WRT Widget, **these steps are necessary**:

- Import the **battery_component.js** JavaScript library (source code available here: Wrt battery component.txt).

```
<script language="javascript" type="text/javascript"
src="battery_component.js"></script>
```

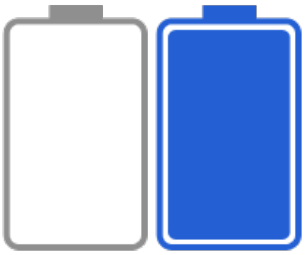
- **Define a parent element** within your WRT HTML code:

```
<body>
[...]
 <div id="battery_holder"></div>
[...]
</body>
```

- If you're using **WRT 1.0**, embed this HTML code snippet in your widget HTML code (as explained here):

```
<embed type="application/x-systeminfo-widget" hidden="yes"></embed>
```

- Add to your WRT project 2 images of the same size, one representing the empty battery, the other one representing the full one. Sample images used in this article are the following:



- **Initialize a Battery instance**, passing these arguments:
  - the empty battery image path
  - the full battery image path

(C) Copyright Nokia 2014. All rights reserved.

- the battery images width
- the battery images height

Once instantiated, append the battery instance to the parent element defined above:

```
var battery = new Battery('images/battery_empty.png', 'images/battery_full.png', 96,
168);

battery.appendTo(document.getElementById('battery_holder'));
```

Source code: the Battery component implementation

Battery constructor and DOM structure

Let's start defining the **Battery object constructor**, with some first **properties and methods related to the component DOM structure**.

```
function Battery(emptyBatterySrc, fullBatterySrc, batteryWidth, batteryHeight)
{

 this.batteryHeight = batteryHeight;
 this.batteryWidth = batteryWidth;

 this.emptyBatteryElement = null;
 this.domElement = null;

 this.init(emptyBatterySrc, fullBatterySrc);
}
Battery.prototype.init = function(emptyBatterySrc, fullBatterySrc)
{
 var el = document.createElement('div');
 el.style.position = 'relative';

 var fullImage = document.createElement('img');
 fullImage.style.position = 'absolute';
 fullImage.src = fullBatterySrc;

 this.emptyBatteryElement = document.createElement('div');
 this.emptyBatteryElement.style.position = 'absolute';
 this.emptyBatteryElement.style.overflow = 'hidden';
 this.emptyBatteryElement.style.width = this.batteryWidth + 'px';
 this.emptyBatteryElement.style.height = '0px';

 var emptyImage = document.createElement('img');
 emptyImage.src = emptyBatterySrc;

 this.emptyBatteryElement.appendChild(emptyImage);

 el.appendChild(fullImage);
 el.appendChild(this.emptyBatteryElement);

 this.domElement = el;
}
```

The Battery constructor' actually stores the battery width and height, passed as arguments, and then call the `init()` method', that creates the component DOM structure, storing some HTML element references in instance variables.

(C) Copyright Nokia 2014. All rights reserved.

Different initialization for WRT 1.0 and WRT 1.1

Now, let's define 2 Battery properties that will hold references to the WRT 1.0 and 1.1 service objects, respectively. So, let's modify the Battery constructor as follows.

```
function Battery(emptyBatterySrc, fullBatterySrc, batteryWidth, batteryHeight)
{
 [...]

 //for WRT 1.1
 this.serviceObject = null;

 //for WRT 1.0
 this.sysInfo = null;
}
```

WRT Service(s) initialization

WRT 1.0 and 1.1 SystemInfo Service APIs must be handled with different code, so we define a method that, after appending the Battery DOM element to a specific parent element, calls the initialization method specific to the WRT version:

```
Battery.prototype.appendTo = function(parentElement)
{
 parentElement.appendChild(this.domElement);

 if(device)
 {
  this.startService11();
 }
 else
 {
  this.startService10();
 }
}
```

WRT 1.0 SystemInfo Service initialization

The **SystemInfo Service API for WRT 1.0** allows access to some device's properties, by using a **plug-in module that must be embedded** by defining the following HTML code:

```
<embed type="application/x-systeminfo-widget" hidden="yes"></embed>
```

After embedding this HTML code, it is possible to **obtain a reference to it**, and then to **retrieve values of the battery-related properties**.

```
Battery.prototype.startService10 = function()
{
 this.sysInfo = document.embeds[0];

 if(this.sysInfo)
 {
  this.updateBattery(this.sysInfo.chargelevel);

  this.sysInfo.onchargerconnected = "batteryChargerHandler();";
```

 (C) Copyright Nokia 2014. All rights reserved.

```
  this.sysInfo.onchargelevel = "batteryLevelHandler();";
 }
}
```

In the above code, we define 2 custom handlers for the `onchagerconnected` and `onchargelevel` events. Since these functions are not Battery instance methods, they need to grab a reference to the single Battery instance, in order to properly work. To do this, let's first define an array that will hold all the Battery objects (assuming there could be more than one).

```
function Battery(emptyBatterySrc, fullBatterySrc, batteryWidth, batteryHeight)
{
 [...]

 Battery.instances.push(this);
}

Battery.instances = new Array();
```

So, it is now possible to **define the 2 handler functions as follows**:

```
function batteryChargerHandler()
{
 for(var i = 0; i < Battery.instances.length; i++)
 {

Battery.instances[i].setChargerConnected(Battery.instances[i].sysInfo.chargerconnected);
 }
}
function batteryLevelHandler()
{
 for(var i = 0; i < Battery.instances.length; i++)
 {
  Battery.instances[i].updateBattery(Battery.instances[i].sysInfo.chargelevel);
 }
}
```

What these 2 handlers do is to loop the Battery instances, and call the `setChargerConnected(` and `updateBattery()` methods, that will be implemented later, and that will manage charger status and battery level, respectively.

WRT 1.1 SystemInfo Service initialization

The **WRT 1.1 SystemInfo Service API** allows widgets to access and modify system information on a device. The API is integrated into WRT through the device object'.

```
Battery.prototype.startService11 = function()
{
 this.serviceObject = device.getServiceObject("Service.SysInfo", "ISysInfo");

 if (this.serviceObject)
 {
  var self = this;

  /* retrieving battery level information */

  var criteria = {
   'Entity': 'Battery',
```


(C) Copyright Nokia 2014. All rights reserved.

```
    'Key': 'BatteryStrength'
   };

   var callback = function(transId, eventCode, result){
    self.batteryCallback(transId, eventCode, result);
   };

   this.serviceObject.ISysInfo.GetInfo(criteria, callback);

   this.serviceObject.ISysInfo.GetNotification(criteria, callback);

   /* retrieving charging status information */

   var criteria = {
    'Entity': 'Battery',
    'Key': 'ChargingStatus'
   };

   var callback = function(transId, eventCode, result){
    self.chargerCallback(transId, eventCode, result);
   };

   var result = this.serviceObject.ISysInfo.GetInfo(criteria);

   if (result.ErrorCode == 0)
    this.setChargerConnected(result.ReturnValue.Status);

   this.serviceObject.ISysInfo.GetNotification(criteria, callback);
  }
 }
```

The above code uses the **GetInfo()** method to retrieve current **status of battery level** (in asynchronous mode) and **charging status** (in synchronous mode). Also, the **GetNotification()** method is used to be notified of changes of these 2 properties. The async **GetInfo()** and both the **GetNotification()** methods need a callback handler: *batteryCallback()* for the Battery level information, and *chargerCallback()* for the charger-related information. Being asynchronous callback handlers, they have to be defined as described on Nokia Developer Library:

- http://library.developer.nokia.com/index.jsp?topic=/Web_Developers_Library/GUID-C7C3B541-8D28-4DC6-9003-ECF57F26C608.html

```
Battery.prototype.chargerCallback = function(transId, eventCode, result)
{
 if(eventCode != 4 && result.ErrorCode == 0)
 {
  this.setChargerConnected(result.ReturnValue.Status);
 }
}

Battery.prototype.batteryCallback = function(transId, eventCode, result)
{
 if(eventCode != 4 && result.ErrorCode == 0)
 {
  this.batteryValue = result.ReturnValue.Status;

  if(!this.chargerConnected)
   this.updateBattery(this.batteryValue);
```


(C) Copyright Nokia 2014. All rights reserved.

```
 }
}
```

Both these handlers **check if there is no error**, and then call the same functions called by WRT 1.0 service handling methods: `setChargerConnected()` and `updateBattery()`. In case of errors, the component simply does nothing: this is intended to not interfere with the Widget main behavior, so avoiding intrusive alerts or popup messages.

Updating the battery level (and images)

First, let's define 2 properties, that will hold the current **battery level**, and the current **charging status**:

```
function Battery(emptyBatterySrc, fullBatterySrc, batteryWidth, batteryHeight)
{
 [...]

 this.batteryValue = 0;
 this.chargerConnected = false;
}
```

The **updateBattery()** implementation does 2 things:

- updates the `batteryValue` property
- updates the graphical appearance of the Battery component, to show the new battery level

```
/**
 * Updates battery with the given value (0 - 100 range)
 * @param {Number} value
 */
Battery.prototype.updateBattery = function(value)
{
 if(value == undefined || value < 0)
  value = 0;

 if(!this.chargerConnected)
  this.batteryValue = value;

 var height = Math.round(((100 - value) * this.batteryHeight) / 100);

 this.emptyBatteryElement.style.height = height + 'px';
}
```

Managing the charger status

When the charger is connected, it would be nice to have an animation showing the battery in its charging status. To implement the animation, let's define some useful properties:

```
function Battery(emptyBatterySrc, fullBatterySrc, batteryWidth, batteryHeight)
{
 [...]

 this.chargeInterval = null;
 this.chargeStep = 0;
 this.CHARGE_STEPS = 5;
}
```


(C) Copyright Nokia 2014. All rights reserved.

These properties define, respectively:

- `chargeInterval`: will hold a reference to the id returned by setInterval() method
- `chargeStep`': holds the current animation step
- `CHARGE_STEPS`: defined the number of steps of the animation

So, the animation progress will be implemented via `setInterval()`, that will periodically update the battery level, to show the battery in its charging status.

```
Battery.prototype.setChargerConnected = function(connected)
{
 if(this.chargerConnected != connected)
 {
  this.chargerConnected = connected;

  if(connected)
  {
   this.chargeStep = 0;

   this.updateBattery(this.chargeStep * 100 / this.CHARGE_STEPS);

   var self = this;

   this.chargeInterval = setInterval(
    function()
    {
     self.chargeStep++;

     if(self.chargeStep > self.CHARGE_STEPS)
      self.chargeStep = 0;

     self.updateBattery(self.chargeStep * 100 / self.CHARGE_STEPS);
    },
    500
   );
  }
  else
  {
   clearInterval(this.chargeInterval);

   this.updateBattery(this.batteryValue);
  }
 }
}
```

Considerations

- **Error management**: this component manages errors simply skipping them. This was done to not interfere with the Widget main behavior. A possible alternative management could be to **hide the component if something goes wrong**, or to show an according image to the user, notifying him of the unavailability of the battery level.
- **WRT 1.0 support**: the **WRT 1.0 code actually works also on WRT 1.1 runtimes**, so it would have been enough to write code only for WRT 1.0. However, the purpose of this article is to show **how to manage different runtime versions** in JavaScript code, and to implement both versions of the component.

Downloads

Here's a list of related downloads:

(C) Copyright Nokia 2014. All rights reserved.

- Sample WRT Widget using the Battery component: BatteryWidget.zip
- Battery empty image: Wrt battery component empty.png
- Battery full image: Wrt battery component full.png
- WRT Battery component source code: Wrt battery component.txt (just change file extension to ".js")

(C) Copyright Nokia 2014. All rights reserved.