How to dynamically change font size in Web Runtime widgets

This article explains how to implement dynamic font sizes in a Symbian Web Runtime widget.

Description

This approach is useful in situations where **the user wants to change the widget's font size** according to its own preferences and needs. Sample scenarios include:

- visually impaired users, who can't read small texts
- users in mobility that want to easily read the widget's contents, may want to increase the font size
- users that want to display more information at once, so decreasing the overall font size

This article presents three different approaches allowing the user to dynamically choose and set his preferred font size. Approaches can be used singularly, or be combined to obtain more complex results.

Widget's base HTML code

The sample widget used to show the different techniques contains the following HTML code:

```
<html>
 [...]

 <body>
  <h1>Page heading</h1>
  <div>
   <p>This <strong>Web Runtime widget</strong>
explains how to dynamically change <em>font size</em>.</p>
  </div>
 </body>
</html>
```

Techniques for dynamic font size change

Changing the base font size

The first presented technique consists in changing the base font size of a widget. Typically, this means changing the font size of the body element. In order for this technique to correctly work, all font sizes must be expressed in relative units of measure, so using em or percentages values. If absolute units of measures (e.g.: **pt or px**) are used for some elements, the font size of these elements will not be changed using this technique.

(C) Copyright Nokia 2014. All rights reserved.

CSS code

The following CSS style sheet is applied to the HTML code seen above:

```
h1 {
 font-size: 2em;
}
h2 {
 font-size: 1.5em;
}

p {
 font-size: 1em;
}

input {
 font-size: 0.8em;
}
```

So, the CSS contains all font-sizes specified using **relative units of measure**.

The JavaScript code

In order to change the base font size of this widget, it's enough to change the font size of the body element. Since this has to be done dynamically, the following **setBaseFontSize()** JavaScript function is defined:

```
function setBaseFontSize(sizeIndex)
{
 document.body.style.fontSize = (100 + sizeIndex * 50) + '%';
}
```

Adding user interaction

In order to allow the users to change the font size, **3 buttons are added to the widget's user interface**, so adding this HTML snippet to widget's code:

```
[...]

   <h2>Select one of these buttons to modify base stylesheet:</h2>

   <input type="submit" onclick="setBaseFontSize(0)" value="Small" />

   <input type="submit" onclick="setBaseFontSize(1)" value="Medium" />

   <input type="submit" onclick="setBaseFontSize(2)" value="Large" />
```

Result

This technique allow to **easily change all the font sizes expressed by using relative unit of measures**, by simply changing the body element's font size. With this approach, **proportions among font sizes of different elements are always maintained**, since relative units are used.

Switching the active style sheet

This technique **works by using multiple style sheets**, one for each different set of font-sizes, and allows the user to **dynamically switch the currently active style sheet**. This approach does not need font sizes to be expressed using relative unit of measures, even if it is always good practice to use them.

(C) Copyright Nokia 2014. All rights reserved.

CSS code

Since this approach works by using multiple style sheets, let's build **3 different CSS files**, that will be dynamically switched.

- **Style sheet for Small font size (fontsize_0.css)**

```
h1 {
 font-size: 2em;
}
h2 {
 font-size: 1.5em;
}

p {
 font-size: 1em;
}

input {
 font-size: 0.8em;
}
```

- Style sheet for Medium font size (fontsize_1.css)

```
h1 {
 font-size: 3em;
}
h2 {
 font-size: 2em;
}

p {
 font-size: 1.5em;
}

input {
 font-size: 1em;
}
```

- Style sheet for Large font size (fontsize_2.css)

```
h1 {
 font-size: 4em;
}
h2 {
 font-size: 3em;
}

p {
 font-size: 2em;
}

input {
 font-size: 1.5em;
}
```


(C) Copyright Nokia 2014. All rights reserved.

The HTML code

First, the 3 style sheets need to be added to the widget's HTML code, and this is done with the following HTML snippet:

```
<html>
 <head>
  [...]

  <link rel="stylesheet" href="fontsize_0.css" type="text/css">
  <link rel="stylesheet" href="fontsize_1.css" type="text/css">
  <link rel="stylesheet" href="fontsize_2.css" type="text/css">
 </head>

 [...]
</html>
```

Then, user interaction must be added to the widget, to allow the user to switch the active style sheet.

```
   <h2>Select one of these buttons to switch the active stylesheet:</h2>

   <input type="submit" onclick="switchStyleSheet(0)" value="Small" />

   <input type="submit" onclick="switchStyleSheet(1)" value="Medium" />

   <input type="submit" onclick="switchStyleSheet(2)" value="Large" />
```

The JavaScript code

Enabling the default style sheet

Since **multiple style sheets are defined** within this widget, the first thing the widgets need to do is to **choose a default one, and to disable all the others**. This is typically done in the widget's **onload event**, and can be performed with the following JavaScript code:

```
function init()
{
 document.styleSheets[1].disabled = true;
 document.styleSheets[2].disabled = true;
}
```

So, the 2nd and 3rd style sheets are disabled, leaving the first one (**fontsize_0.css**) as the initially active style sheet.

Changing the active style sheet

In order to change the active style sheet, the `switchStyleSheet()` function was defined in the widget's HTML code. This function can be implemented as follows:

```
function switchStyleSheet(sizeIndex)
{
 for(var i = 0; i < 3; i++)
 {
  document.styleSheets[i].disabled = (sizeIndex != i);
 }
}
```


(C) Copyright Nokia 2014. All rights reserved.

The `switchStyleSheet()` function just enable the specified style sheet, disabling all the other ones.

Result

This approach offers a greater level of control on font sizes than the previous presented technique, since it allows to explicitly set the font size of each element in each style sheet included in the widget. For this reason, **font sizes proportions are not automatically maintained**, since it's up to the widget's designer/developer to define the font sizes in each different style sheet.

Changing the font size of specific elements via JavaScript

The previous techniques showed how to change the font size of whole widgets. This technique shows how to change font size of specific elements, by using JavaScript.

CSS code

Let's take back the CSS code of the first presented approach:

```
h1 {
 font-size: 2em;
}
h2 {
 font-size: 1.5em;
}

p {
 font-size: 1em;
}

input {
 font-size: 0.8em;
}
```

The JavaScript code

Now, let's say that the user needs to modify the font size of the input fields, leaving all the other sizes untouched. This can be done by the following JavaScript function:

```
function setJSFontSize(sizeIndex)
{
 var inputs = document.getElementsByTagName('input');

 var fontSize;

 switch(sizeIndex)
 {
  case 0:
   fontSize = '1em';
   break;
  case 1:
   fontSize = '1.5em';
   break;
  case 2:
   fontSize = '2em';
   break;
 }

 for(var i = 0; i < inputs.length; i++)
 {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    inputs[i].style.fontSize = fontSize;
  }
}
```

The `setJSFontSize()` takes all the **input** nodes in the widget's DOM structure, and then applies the desired font size only to these elements.

Adding user interaction

3 buttons are added in order to allow user interaction:

```
    <h2>Select one of these buttons to change buttons' font size via JavaScript:</h2>

    <input type="submit" onclick="setJSFontSize(0)" value="Small" />

    <input type="submit" onclick="setJSFontSize(1)" value="Medium" />

    <input type="submit" onclick="setJSFontSize(2)" value="Large" />
```

Result

This approach offers the **greatest level of control over the font sizes** of single HTML elements, but it is also more complex to implement and maintain, since it uses JavaScript code to explicitly set the specific font sizes. So, it is preferable to use this approach only when really needed, preferring the 2 previous techniques in all the other scenarios

Download

A sample widget, that shows all the techniques presented in this article, can be downloaded from this link: Media:FontSizeChangerWidget.zip

(C) Copyright Nokia 2014. All rights reserved.