Detecting orientation changes in Symbian Web Runtime

Overview

Because of the vast variety in screen sizes on Nokia devices, it is important that the developer pays attention to the scalability of his or her widget. In addition, the orientation of the screen can change. There are a couple of things which need to be taken into account in this respect. The first is detecting when the screen size changes. The second is what to do when the screen size changes (for more information about this, see Reacting to the changes in screen size in Symbian Web Runtime.) This snippet covers the first aspect.

The easiest way to detect when the screen size changes is to use the `onresize` JavaScript function. It is called automatically when the screen is resized. This happens, for example, when the orientation of the device changes. However, some devices, such as the first product release of the Nokia 5800 XpressMusic, do not support the `onresize` function. This requires additional measures which are explained later on in this snippet. First though, let's consider the simplest case.

Source: Using the onresize event

Here is the JavaScript code which assigns an event handler (a callback function, if you want) to the `onresize` event:

```
window.onresize = windowResized;
```

The function itself is called whenever the screen size changes:

```
// Called when the window is resized
function windowResized() {
    // TODO: Add code for different screen sizes (i.e. orientations)
}
```

It may also be necessary to call the `windowResized()` function as soon as the page is loaded. This ensures that the correct orientation is recognised and the screen is laid out correctly when the widget is started. Here's the code for calling the function during startup:

```
// Register an event handler for the onload event
// (i.e. call the init function as soon as the page has been loaded)
window.onload = init;
```

```
// Does initialization tasks for the widget
function init(){
    // Call windowResized manually here to ensure the correct layout
    windowResized();
}
```

Source: Using the timer

There are some devices that do not support the `onresize` function. As a workaround for this known issue, a timer can be started to poll for changes in the screen size. Here's how:

```
var timer = null;

window.onload = init;

// Does initialization tasks for the widget
```


(C) Copyright Nokia 2014. All rights reserved.

```
function init(){
    // Call the tick function at one second intervals
    timer = setInterval("tick()", 1000);
}

// Called when the timer's interval elapses
function tick() {
    // Call the windowResized function to achieve the same effect as if it were
    // triggered by WRT
    windowResized();
}
```

This code can be further enhanced by preventing unnecessary calls to `windowResized()`. This can be done by detecting the new resolution and comparing it to the current one. If the resolution hasn't changed from the last tick of the timer, there's no need to do anything.

```
// Device resolutions. Can be used for device-specific customization.
var RESOLUTION_UNDEFINED = 0;
var RESOLUTION_QVGA_LANDSCAPE = 1;  // 320x240
var RESOLUTION_QVGA_PORTRAIT = 2;   // 240x320
var RESOLUTION_NHD_LANDSCAPE = 3;   // 640x360
var RESOLUTION_NHD_PORTRAIT = 4;    // 360x640
var RESOLUTION_HOME_SCREEN = 5;     // less than 75 % of the resolutions above
var resolution = RESOLUTION_UNDEFINED;

// Called when the timer's interval elapses
function tick() {
    var prevResolution = resolution;

    // Detect the new resolution.
    detectResolution();

    // If the resolution hasn't changed, there's no need to do anything.
    if (resolution == prevResolution) {
        return;
    }

    // Call the windowResized function to achieve the same effect as if it were
    // triggered by WRT
    windowResized();
}

// Detects the resolution of the device
function detectResolution() {
    var screenWidth = screen.width;
    var screenHeight = screen.height;
    var windowWidth = window.innerWidth;
    var windowHeight = window.innerHeight;

    if (windowHeight < (0.75 * screenHeight) ||
        windowWidth < (0.75 * screenWidth)) {
        // If the window width or height is less than 75 % of the screen width
        // or height, we assume the home screen view should be active
        resolution = RESOLUTION_HOME_SCREEN;
    } else if (screenWidth == 240 && screenHeight == 320) {
        resolution = RESOLUTION_QVGA_PORTRAIT;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    } else if (screenWidth == 320 && screenHeight == 240) {
        resolution = RESOLUTION_QVGA_LANDSCAPE;
    } else if (screenWidth == 360 && screenHeight == 640) {
        resolution = RESOLUTION_NHD_PORTRAIT;
    } else if (screenWidth == 640 && screenHeight == 360) {
        resolution = RESOLUTION_NHD_LANDSCAPE;
    } else {
        resolution = RESOLUTION_UNDEFINED;
    }
}
```

Postconditions

Changes in screen size are detected.

See also

- Reacting to the changes in screen size in Symbian Web Runtime
- "STEW: supporting screen rotation" in Nokia Developer Web Developer's Library

Supplementary material

This code snippet is part of the stub concept, which means that it has been patched on top of a template application in order to be more useful for developers. The version of the WRT stub application used as a template in this snippet is v1.2.

- The patched, executable application that can be used to test the features described in this snippet is available for download at Media:detecting orientation changes.zip.
- You can view all the changes that are required to implement the above-mentioned features. The changes are provided in unified diff and colour-coded diff (HTML) formats in Media:detecting orientation changes.diff.zip.
- For general information on applying the patch, see Using Diffs.
- For unpatched stub applications, see Example app stubs with logging framework.

(C) Copyright Nokia 2014. All rights reserved.