

How to create animated images in Web Runtime widgets

This article explains how to programmatically create animated images in Web Runtime widgets.

Description

First Web Runtime releases have limited support for GIF images, so alternative approaches are needed in order to have animated images within a WRT widget. Moreover, the techniques described in this article will allow for a greater level of flexibility and control over the animated images.



First approach: using multiple images

The first technique works by using a standard IMG tag, and by using multiple images that are used to create the animation effect. In order to create the desired animation effect, the IMG src attribute is periodically replaced, by using the `setInterval()` JavaScript function.

JavaScript code: the `MultipleImagesAnimator` class

The following class uses the technique described above to periodically modify the src attribute of an HTML Image element in order to create the animation effect.

The constructor

The constructor takes 3 parameters:

- the HTML Image element to be used for the animation
- an Array containing the image paths
- the interval, in milliseconds, between the different animation frames

```
function MultipleImagesAnimator(imageElement, imagePaths, frameInterval)
{
    this.imageElement = imageElement;
    this.imagePaths = imagePaths;
    this.frameInterval = frameInterval;
    this.currentFrameIndex = 0;
    this.interval = null;
}
```

Starting and stopping the animation

The following two methods will respectively start and stop the animation, by using the `setInterval()` and `clearInterval()` JavaScript

functions:

```
MultipleImagesAnimator.prototype.start = function()
{
  if(this.interval == null)
  {
    var self = this;

    this.interval = setInterval(
      function()
      {
        self.nextFrame();
      },
      this.frameInterval
    );
  }
}

MultipleImagesAnimator.prototype.stop = function()
{
  if(this.interval != null)
  {
    clearInterval(this.interval);

    this.interval = null;
  }
}
```

Performing the animation

The actual animation effect is performed by the nextFrame() method, that takes care of replacing the current image with the next one taken from the Array passed to the MultipleImagesAnimator constructor:

```
MultipleImagesAnimator.prototype.nextFrame = function()
{
  this.imageElement.src = this.imagePaths[this.currentFrameIndex];

  this.currentFrameIndex = (this.currentFrameIndex + 1) % this.imagePaths.length;
}
```

The HTML code

In order to use the MultipleImagesAnimator class, it is enough to define an HTML Image element in the Widget code:

```
<html>
<head>
  <title> Sample Widget</title>
  <script language="javascript" type="text/javascript" src="basic.js"></script>
  <script language="javascript" type="text/javascript"
src="MultipleImagesAnimator.js"></script>
</head>
<body onLoad="javascript:init();">
  <h2>Image animation Widget</h2>
```

```
<img id="multiple_images" />  
  
</body>  
</html>
```

Using the MultipleImagesAnimator

In order to use the `MultipleImagesAnimator` class defined above, it's necessary to include in the WRT widget the images to be used for the animation, and then to instantiate it with a code similar to the following:

```
function init()  
{  
    var imageFrames = new Array(  
        'images/loader0.png',  
        'images/loader1.png',  
        'images/loader2.png',  
        'images/loader3.png',  
        'images/loader4.png',  
        'images/loader5.png'  
    );  
  
    new MultipleImagesAnimator(  
        document.getElementById('multiple_images'),  
        imageFrames,  
        200  
    ).start();  
}
```

The above code creates an Array containing the paths of the animation images, and then instantiate a new `MultipleImagesAnimator` object by passing the HTML Image element defined above, the paths Array, and an interval of 200 milliseconds

Second approach: using CSS sprites

CSS sprites are a commonly used Web technique that uses multiple images combined in a single, larger image in order to minimize and optimize performances and network traffic. The single images are then shown by using appropriate CSS properties, in order to show only the relevant portion of the master image. CSS sprites are described in detail in this Nokia Developer Wiki article: [Mobile Design Pattern: High Performance Widgets: CSS Sprites](#)

JavaScript code: the ImageSpriteAnimator class

The following class works by adjusting the background properties of a given HTML element, in order to show only the current animation frame. This code assumes that single animation frames are positioned vertically within the master image, and all frames have the same width and height.

The constructor

The constructor takes 6 parameters:

- the HTML element to be used for the animation (any HTML element can be good, e.g.: DIVs, SPANs, etc..)
- the path of the image containing the animation sprites
- the total number of sprites
- the image width
- the height of a single animation frame
- the interval, in milliseconds, between the different animation frames

```
function ImageSpriteAnimator(imageElement, imagePath, totalFrames, imageWidth,  
    frameHeight, frameInterval)
```

```
{  
    this.imageElement = imageElement;  
    this.imagePath = imagePath;  
    this.totalFrames = totalFrames;  
    this.frameInterval = frameInterval;  
    this.imageWidth = imageWidth;  
    this.frameHeight = frameHeight;  
    this.currentFrameIndex = 0;  
    this.interval = null;  
}
```

Starting and stopping the animation

The following two methods will respectively start and stop the animation, by using the `setInterval()` and `clearInterval()` JavaScript functions. The `start()` method initializes the `backgroundImage` property of the given HTML element, by using the path of the image containing the animation frames, and sets the element's width and height properties, by using the values passed to `ImageSpriteAnimator`'s constructor.

```
ImageSpriteAnimator.prototype.start = function()  
{  
    if(this.interval == null)  
    {  
        this.imageElement.style.backgroundImage = "url('" + this.imagePath + "')";  
        this.imageElement.style.height = this.frameHeight + 'px';  
        this.imageElement.style.width = this.imageWidth + 'px';  
        var self = this;  
        this.interval = setInterval(  
            function()  
            {  
                self.nextFrame();  
            },  
            this.frameInterval  
        );  
    }  
}  
  
ImageSpriteAnimator.prototype.stop = function()  
{  
    if(this.interval != null)  
    {  
        clearInterval(this.interval);  
  
        this.interval = null;  
    }  
}
```

Performing the animation

The actual animation effect is performed by the `nextFrame()` method, that periodically moves the background position in order to show the current animation frame.

```
ImageSpriteAnimator.prototype.nextFrame = function()  
{  
    this.imageElement.style.backgroundPosition = "0px " + (- this.frameHeight *  
    this.currentFrameIndex) + "px";
```

```
    this.currentFrameIndex = (this.currentFrameIndex + 1) % this.totalFrames;
}
```

The HTML code

In order to use the `ImageSpriteAnimator` class, the following DIV element will be used:

```
<html>
<head>
<title> Sample Widget</title>
<script language="javascript" type="text/javascript" src="basic.js"></script>
<script language="javascript" type="text/javascript"
src="ImageSpriteAnimator.js"></script>
</head>
<body onLoad="javascript:init();">
<h2>Image animation Widget</h2>

<div id="image_sprite"></div>

</body>
</html>
```

Using the `ImageSpriteAnimator`

In order to use the `ImageSpriteAnimator` class defined above, it's first necessary to include in the WRT widget the image containing all the animation frames. Then, it is possible to instantiate it as follows:

```
function init()
{
    new ImageSpriteAnimator(
        document.getElementById('image_sprite'),
        'images/loader_sprite.png',
        6,
        16,
        16,
        200
    ).start();
}
```

The above code creates a new `ImageSpriteAnimator` instance by using the given HTML element and image path, and so creating an animation with 6 frames, a width and height of 16 pixels, and an interval between frames of 200 milliseconds.

Downloads

The following widget shows both the techniques described in this article: [Media:AnimatedImageWidget.zip](#)

