

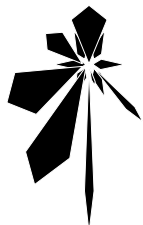
Introduction to JavaScript Animation

Revision 1.1

June 07, 2010

by

Aloysius Indrayanto



(C) 2010 AnemoneSoft.com

This document is multi-licensed under the Creative Commons Attribution Share-Alike (CC-BY-SA) license version 3.0 and the GNU Free Documentation License (GNU FDL) version 1.3 or later.

1. Introduction

Animation provides an optical illusion of motion. In computer graphics, it is done by rapidly displaying a sequence of 2D images or 3D models. Animation in JavaScript is mostly related to these four functions:

Function	Arguments	Return Value
<code>setTimeout(function, timeout)</code>	<i>function</i> : a function or a piece of JavaScript expression to be executed when the timeout occurred. <i>timeout</i> : the number of milliseconds for the timeout to occur.	<i>timeoutID</i> : a numeric timeout ID corresponds to the newly set timeout.
<code>clearTimeout(timeoutID)</code>	<i>timeoutID</i> : a numeric timeout ID that was returned by <code>setTimeout()</code> .	N/A
<code>setInterval(function, interval)</code>	<i>function</i> : a function or a piece of JavaScript expression to be executed periodically. <i>interval</i> : the number of milliseconds for the periodical execution of the function.	<i>intervalID</i> : a numeric timeout ID corresponds to the newly set interval.
<code>clearInterval(intervalID)</code>	<i>intervalID</i> : a numeric interval ID that was returned by <code>setInterval()</code> .	N/A

1.1. Choice of Function

The `setInterval()` function has the advantage that for a particular animation, it will only need to be called once and then the animation will be updated automatically with the given interval. Therefore, this function will be more suitable for simple animations with minimum calculation and processing time. In this case, we assume that the PC will be able to finish all the calculation and processing needed to update the current animation frame before the next interval is due. If the PC is old and slow, or if the calculation and processing are too extensive, the resulting animation can be no longer updated at the specified interval.

The `setTimeout()` function, despite the fact that it will need to be recalled for every animation frame, has the advantage users can dynamically adjust the updating (timeout) rate of the animation. If the system detect that the animation has run too slow, it can reduce the timeout delay so that the animation speed can be increased; or, it may skip the current animation frame altogether and proceed directly with the next frame. Therefore, this function can be more suitable for complicated animations with extensive calculation and processing; especially if the time needed to update the animation can vary greatly between frames.

1.2. Image Format Consideration

JavaScript is able to manipulate the position, size, and visibility of almost all HTML elements, including the image element ``. All modern browsers supports animated GIF (Graphics Interchange Format). Therefore if there is a need to display an animated character, it can be easier to use animated GIF rather than animating the character frame by frame (changing its image) using JavaScript. In this case, JavaScript will be only used to update the position and size of the character's image. However, Animated GIF uses a simple image format, it only supports 256 colors and one level of transparency (fully opaque or fully transparent). Usually this is enough for simple a animated character. Thus, if the character needs more than 256 colors or a smoother transparency support, animated GIF cannot be used.

Another popular image format for the internet is PNG (Portable Network Graphics). This formats supports true color (RGB) and a full range of transparency (256 levels). However, this format does not support animation. Hence, the JavaScript will need to update the character's image frame by frame. Note that, Internet Explorer lower than version 7.0 does not have a native support for PNG transparency (transparency needs to be implemented as CSS hack). Basically, instead defining the image element as:

```

```

define it as:

```

```

Please refer to the website in the reference section for the details on how to perform this CSS hack.

A newer PNG format, called animated PNG (APNG) supports animation. However, it is currently only supported by FireFox 3.0+ and Opera 9.5+. Although, there is a JavaScript hack to make APNG cross browser (please refer to the website in the reference section), it still causes much more additional complexity. Hence, APNG may still not a good choice for image format. In conclusion, use animated GIF whenever possible and switch to PNG if necessary (with CSS hack for Internet Explorer lower than version 7.0).

2. Requirements

A basic knowledge in programming using JavaScript and Prototype.JS will be needed to understand the topic discussed in this tutorial. A supported browser (Internet Explorer 6.0+, Mozilla Firefox 1.5+, Google Chrome 1.0+, Apple Safari 2.04+, or Opera 9.25+) will be also needed to run the code snippets.

3. Basic Techniques

3.1. Using the `setTimeout()` and `setInterval()` Functions

The code snippets below demonstrate the use of `setTimeout()` and `setInterval()`. The content of the 'divOne' element will be updated once, while the content of the 'divTwo' element will be updated for about every one seconds. Note that all the code snippets in this tutorial will utilize the Prototype.JS JavaScript framework.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

  <head>
    <script type="text/javascript" src="prototype-c161.js"></script>
  </head>

  <body>
    <div id="divOne">I will change after 5 seconds</div>
    <div>&nbsp;</div>
    <div id="divTwo">0</div>
  </body>

</html>

<script type="text/javascript">
  setTimeout(
    function()
    {
      $('divOne').innerHTML = 'See, I told you ;)';
    },
    5000
  );
  setInterval(
    function()
    {
      $('divTwo').innerHTML = parseInt($('divTwo').innerHTML) + 1;
    },
    1000
  );
</script>
```

tut01.html: Using `setTimeout()` and `setInterval()`.

For simplicity, in all the remaining sections, only the content of the `<body></body>` and `<script></script>` tags that will be shown in all code snippets.

The next code snippet demonstrate the use of `setTimeout()` for continuous animation. This is the standard construct for animation using `setTimeout()` that will be used through the rest of the tutorial.

```
...
...
<body>
  <div id='divTwo'>0</div>
</body>
...
...
<script type='text/javascript'>
  setTimeout(
    function()
    {
      $('divTwo').innerHTML = parseInt($('divTwo').innerHTML) + 1;
      setTimeout(arguments.callee, 1000);
    },
    1000
  );
</script>
```

tut02.html: Using `setTimeout()` for continuous animation.

3.2. Loading Images

In order to create a smooth animation using multiple image files, it is necessary to load all the images before starting the animation. The code snippet below demonstrate the use of `Image` class for this purpose:

```
...
...
<body>
  <div id='divOne'>Loading image ...</div>
</body>
...
...
<script type='text/javascript'>
  var image = new Image();
  image.src = 'image/javascript-logo.gif';

  setTimeout(
    function()
    {
      if(!image.complete) {
        setTimeout(arguments.callee, 25);
        return;
      }

      var d = $('divOne');
      d.insert('<div>&nbsp;</div>');
      d.insert('<div>Loaded a ' + image.width + 'x' + image.height + ' image.</div>');
      d.insert('<div>&nbsp;</div>');
      d.insert(image);
    },
    25
  );
</script>
```

tut03.html: Loading image using the `Image` class.

Explanation:

- An object of a type Image will behave the same with the HTML image element ``.
- Loading an image file from an URL is as simple as setting the URL in the `src` property of the image object.
- The `complete` property will contain a value that evaluate to `true` if the image has been fully loaded. Using this property, the script periodically check for the completion of the loading process. Try to change the image URL to an internet address pointing to a large enough image; there should be some delay before the complete message and the image are shown.
- The `width` and `height` properties will contain the width and height of the loaded image in pixels. These values will be zero if the image is not yet loaded, the image URL is not valid, the image source (file) is corrupted, the image format is not supported, etc. Due to the `complete` property will also evaluate to `true` even if the image cannot be loaded (because of error), it may be also necessary to check if the `width` or `height` properties still contains zero after the system reports that the image has been loaded.
- The `insert()` method from Prototype.JS can append a new HTML code or a new element object to an existing element. In case adding a new element object, the function behaves identical with the `appendChild()` function from JavaScript.

This image loading method can be extended for multiple images by using an array to store the image objects. Hence, one will need to loop through the items in the array and check if all of their `complete` properties evaluate to `true`.

3.2. Opacity Setting (Transparency)

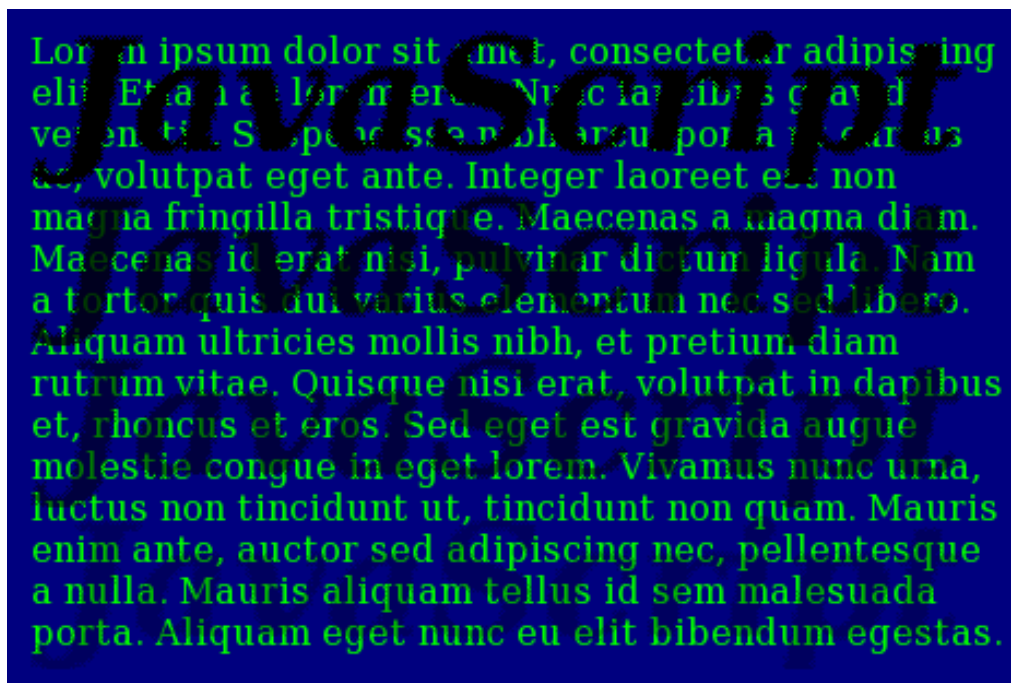
Almost all HTML elements supports opacity setting (transparency). The opacity of an element can be set through its CSS, as demonstrated by the code snippet below:

```
...
...
<body style='background-color:#00007F;'>
  <div style='position:absolute; top:10px; left:10px; width:446px; color:#00FF00;'>
    Lorem ipsum dolor sit amet, ...
  </div>
  <div style='position:absolute; top:10px; left:10px; width:426px;'>
    <div><img id='imgA' src='image/javascript-logo.gif'/></div>
    <div><img id='imgB' src='image/javascript-logo.gif'/></div>
    <div><img id='imgC' src='image/javascript-logo.gif'/></div>
    <div><img id='imgD' src='image/javascript-logo.gif'/></div>
  </div>
</body>
...
```

```
...  
<script type='text/javascript'  
  function setOpacity(object, procent)  
  {  
    object.style.opacity    = procent / 100;  
    object.style.MozOpacity = procent / 100;  
    object.style.filter     = 'alpha(opacity=' + procent + ')';  
  }  
  
  setOpacity($('imgA'), 100);  
  setOpacity($('imgB'), 75);  
  setOpacity($('imgC'), 50);  
  setOpacity($('imgD'), 25);  
</script>
```

tut04.html: Opacity (Transparency).

It is necessary to set multiple CSS values because of inconsistency between browsers. The above code would generate output similar to:



By combining the `setTimeout()` and `setOpacity()` functions, one can create nice fading effect. Try it yourself!

4. A Simple Animated Scenery

The code snippet below will generate a simple animated scenery containing land, grasses, trees, clouds, and chickens. The clouds will be semi transparent and moving trough the sky while some of the chickens will be walking from left to right. In this code snippet, both static GIF and animated GIF files are used.

```

...
...
<body style='text-align:center;'>
  <div id='divMain' style='position:relative; margin-top:20px;
    margin-left:auto; margin-right:auto; width:640px; height:480px;
    background-color:#00FFFF; border:4px #000000 solid;
    overflow-x:hidden; overflow-y:hidden;'>
    <!-- Put the soil -->
    <div style='z-index:0; position:absolute; top:280px; left:0px;
      width:100%; height:35px; background-color:#E08337;'></div>
    <div style='z-index:0; position:absolute; top:305px; left:0px;
      width:100%; height:85px; background-color:#C0702F;'></div>
    <div style='z-index:0; position:absolute; top:380px; left:0px;
      width:100%; height:110px; background-color:#A45514;'></div>
    <!-- Put the tree and grasses -->
    <div style='z-index:1; position:absolute; top:210px; left:130px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:230px; left:50px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:230px; left:100px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:200px; left:500px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:220px; left:450px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:220px; left:560px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:220px; left:590px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:50px; left:500px;'>
      <img src='image/tree.gif' /></div>
    <div style='z-index:1; position:absolute; top:240px; left:475px;'>
      <img src='image/grass.gif' /></div>
    <div style='z-index:1; position:absolute; top:250px; left:530px;'>
      <img src='image/grass.gif' /></div>
    <!-- Put the chickens -->
    <div style='z-index:1; position:absolute; top:330px; left:80px;'>
      <img src='image/chicken-baby.gif' /></div>
    <div style='z-index:1; position:absolute; top:320px; left:120px;'>
      <img src='image/chicken-baby.gif' /></div>
  </div>
  <div style='margin-top:10px;'>
    Cliparts are taken from
    <a href='http://www.clker.com'>http://www.clker.com</a> and
    <a href='http://www.sitevip.net/gifs/chicken'>
      http://www.sitevip.net/gifs/chicken</a>.
  </div>
</body>
...
...
<script type='text/javascript'>
  var totCloud = 5;
  var aryCloud = new Array();
  var imgCloud = new Image();
  var imgCkFam = new Image();
  imgCloud.src = 'image/cloud.gif';
  imgCkFam.src = 'image/chicken-family.gif';

  // Wait until the images are fully loaded and then
  // initialize the cloud object array and chicken-family object
  setTimeout(
    function()
    {
      if(!imgCloud.complete || !imgCkFam.complete) {
        setTimeout(arguments.callee, 25);
        return;
      }
      animInit();
    },
    25
  );

```

```

// Helper function to set the opacity of an element
function setOpacity(object, procent)
{
    object.style.opacity    = procent / 100;
    object.style.MozOpacity = procent / 100;
    object.style.filter     = 'alpha(opacity=' + procent + ')';
}

// Initialize the cloud object array and chicken-family object
function animInit()
{
    // Initialize the cloud objects
    var d = $('divMain');
    for(var i = 0; i < totCloud; ++i) {
        aryCloud[i] = imgCloud.cloneNode(false);
        aryCloud[i].style.zIndex = Math.floor(Math.random() * 3);
        aryCloud[i].style.position = 'absolute';
        aryCloud[i].style.top = Math.floor(Math.random() * (200 - imgCloud.height))
            + 'px';
        aryCloud[i].style.left = Math.floor(Math.random() * (640 - imgCloud.width))
            + 'px';
        aryCloud[i].xSpeed = Math.floor(Math.random() * 3) * 2 + 1;
        setOpacity(aryCloud[i],
            Math.floor(Math.random() * 20) * 2 + (5 * aryCloud[i].style.zIndex) + 30);
        d.insert(aryCloud[i]);
    }

    // Initialize the chicken-family object
    imgCkFam.style.zIndex = 0;
    imgCkFam.style.position = 'absolute';
    imgCkFam.style.top = (280 - imgCkFam.height + 5) + 'px';
    imgCkFam.style.left = '-300px';
    d.insert(imgCkFam);

    // Start animation for the first frame
    setTimeout(animCalback, 40);
}

// Animate the cloud object array and chicken-family object
function animCalback()
{
    // Update the cloud objects' positions
    for(var i = 0; i < totCloud; ++i) {
        var curX = parseInt(aryCloud[i].style.left);
        var newX = curX + aryCloud[i].xSpeed;
        if(newX >= 640) {
            aryCloud[i].style.top = Math.floor(Math.random() * (200 - imgCloud.height))
                + 'px';
            aryCloud[i].style.left = (-imgCloud.width) + 'px';
            aryCloud[i].xSpeed = Math.floor(Math.random() * 3) * 2 + 1;
            setOpacity(aryCloud[i],
                Math.floor(Math.random() * 20) * 2 + (5 * aryCloud[i].style.zIndex) + 30);
        }
        else
            aryCloud[i].style.left = newX + 'px';
    }

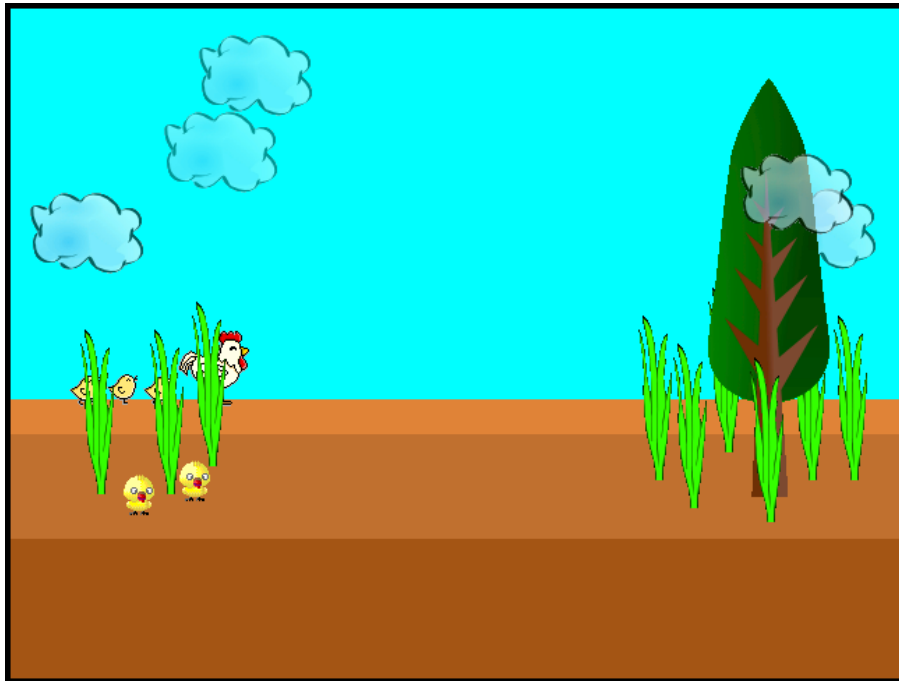
    // Update the chicken-family object's position
    var newX = parseInt(imgCkFam.style.left) + 3;
    if(newX >= 640) imgCkFam.style.left = '-300px';
    else
        imgCkFam.style.left = newX + 'px';

    // Schedule animation for the next frame
    setTimeout(arguments.callee, 40);
}
</script>

```

tut05.html: A simple animated scenery.

The above code would generate output similar to:



Explanation:

- All non-animated objects (soil, tree, grasses, and baby chickens) are simply initialized and hard-coded in using plain HTML.
 - The soil is simply created using three `<div></div>` elements with explicit width, height, and color.
 - The tree, grasses, and baby chickens are created using ``. It is necessary to enclose the `` elements within `<div></div>` elements so that the image can be positioned relative to the parent `<div></div>` element 'divMain'. Basically, a `<div></div>` element with absolute positioning will be positioned relative to its parent's top-left position only if the parent has been positioned using absolute or relative positioning. If there is no specification, `<div></div>` child elements will be positioned relative to the `<body></body>` element.
- All animated objects (clouds and chicken family) are initialized using JavaScript.
 - There is only one chicken family object, so a single Image object is enough. However, there will be more than one clouds, so an array of Image objects will be needed.

- The system then periodically check if the cloud and chicken family images have been loaded using the `setTimeout()` function. If they have been fully loaded, the `animInit()` function will be called to initialize the animation.
 - Basically the clouds will have random starting position, speed, and opacity while the chicken family has a fixed starting position.
 - After all the animated objects have been initialized, the system will start the animation by registering the `animCallback()` function using the `setTimeout()` function.
- In each animation frame, the system update the positions of all the animated-objects.
 - Each of the clouds will be moved to the right as far as its pre-initialized speed. If the cloud has reached beyond the far end of the scene, its position will reset and its speed as well as its opacity will be reinitialized randomly.
 - The chicken family object will be moved to the right with a fixed speed. If it has reached beyond the far end of the scene. Its position will reset to its original starting position.
 - Finally, the system will schedule the animation for the next frame by using the `setTimeout()` function.

References

<http://en.wikipedia.org/wiki/Animation>, June 02, 2010

<http://www.elated.com/articles/javascript-timers-with-settimeout-and-setinterval>, June 02, 2010

<http://24ways.org/2007/supersleight-transparent-png-in-ie6>, June 03, 2010

<http://www.twinhelix.com/css/iepngfix>, June 03, 2010

<http://homepage.ntlworld.com/bobosola/pnginfo.htm>, June 03, 2010

<http://en.wikipedia.org/wiki/APNG>, June 03, 2010

<http://ajaxian.com/archives/apng-class-get-apng-going-on-all-browsers>, June 03, 2010

<http://www.clker.com>, June 03, 2010

<http://www.sitevip.net/gifs/chicken>, June 03, 2010