# VOODOO'S INTRODUCTION TO JAVASCRIPT



# Part 8: The Image-object

## Images on a web-page

Now we are going to have a look at the Image-object which is available since JavaScript 1.1 (i.e. since Netscape Navigator 3.0). With the help of the Image-object you can change images on a web-page. This allows us for example to create animations.
Please note that users of older browsers (like Netscape Navigator 2.0 or Microsoft Internet Explorer 3.0 - they use JavaScript 1.0) cannot run the scripts shown in this part - or at least they cannot see the whole effect. First, let's see how the images in a web-page can be addressed through JavaScript. All images are represented through an array. This array is called images. It is a property of the document-object. Every image on a web-page gets a number. The first image gets the number 0, the second image gets the number 1 and so on. So we can address the first image through document.images[0].
Every image in an HTML-document is considered as an Image-object. An Image-object has got certain properties which can be accessed through JavaScript. You can for example see which size an image has with the properties *width* and *height*. *document.images[0].width* gives you the width (in pixel) of the first image on the web-page.
Especially if you have many images on one page it gets hard to keep count of all images. Giving names to the different images solves this problem. If you declare an image with this tag

*<img src="img.gif" name="myImage" width=100 height=100>*

you can address it through *document.myImage* or *document.images["myImage"]*.

## Loading new images

Although it is nice to know how to get the size of an image on a web-page this is not what we wanted to know. We want to change images on a web-page. For this purpose we need the *src* property. As in the *<img>* tag the *src* property represents the address of the displayed image. With JavaScript 1.1 you can now assign new addresses to an already loaded image on a web-page. The result is that the image located at the new address is being loaded.
This new image replaces the old image on the web-page. Look at this example:

*<img src="img1.gif" name="myImage" width=100 height=100>*

The image *img1.gif* is being loaded and gets the name *myImage*. The following line of code replaces the old image *img1.gif* with the new image *img2.gif*:

*document.myImage.src= "img2.src";*

The new image has always got the same size as the old image. You cannot change the size of

the area in which the image is being displayed.

*(The online version lets you test this script immediately)*

## Preloading images

One drawback might be that the new image gets loaded after assigning a new address to the src property. As the image is not preloaded it takes some time until the new image is retrieved through the Internet. In some situations this is ok - but often these delays are not acceptable. So what can we do about this? Yes, preloading the image is the solution. For this purpose we have to create a new Image-object. Look at these lines of code:

```
hiddenImg= new Image();
hiddenImg.src= "img3.gif";
```

The first line creates a new Image-Object. The second line defines the address of the image which shall be represented through the object hiddenImg. We have already seen that assigning a new address to the *src* attribute forces the browser to load the image the address is pointing at. So the image img2.gif gets loaded when the second line of this code is being executed. As the name hiddenImg implies the image is not being displayed after the browser finished loading it. It is just kept in the memory (or better in the cache) for later use. In order to display this image we can now use this line:

```
document.myImage.src= hiddenImg.src;
```

Now the image is being taken from the cache and displayed immediately. We have managed to preload the image. Of course the browser must have finished the preloading for being able to display an image without delay. So if you have many images specified for preloading there might be a delay nevertheless because the browser has been busy to download all the other pictures. You always have to consider the speed of the Internet - the downloading of the images doesn't go faster with this code shown here. We only try to start the downloading of the images earlier - so the user can see them earlier. This makes the whole process much smoother.
If you have a fast Internet connection you might wonder what all this talk is about. Which delay is this guy talking about all the time? Well, there are still some people sitting behind a 14.4 modem (No, not me. I just upgraded to 33.6 - oh yes...).

## Changing images on user-initiated events

You can create nice effects through changing images as a reaction to certain events. You can for example change images when the mouse cursor is being moved over a certain area.

*(The online version lets you test this script immediately)*

The source code for this example looks like this:

```
<a href="#"
  onMouseOver="document.myImage2.src='img2.gif'"
  onMouseOut="document.myImage2.src='img1.gif'">
<img src="img1.gif" name="myImage2" width=160 height=50 border=0></a>
```

This code causes some problems though:

- The user might not use a JavaScript 1.1 browser.
- The second image is not being preloaded.
- We have to rewrite the code for every image on a web-page.
- We want to have a script which can be used in many web-page over and over again without large changes.

We will now have a look at a complete script which solves these problems. The script gets much longer - but once it is written you do not have to bother about it anymore. There are two requirements for keeping the script flexible:

- Undefined number of images - it should not matter if 10 or 100 images are used
- Undefined order of images - it should be possible to change the order of the images without changing the code

*(The online version lets you test this script immediately)*

Have a look at the code (I have added some comments):

```
<html>
<head>

<script language="JavaScript">
<!-- hide

// ******************************************************
// Script from Stefan Koch - Voodoo's Intro to JavaScript
//     http://rummelplatz.uni-mannheim.de/~skoch/js/
//       JS-book: http://www.dpunkt.de/javascript
//    You can use this code if you leave this message
// ******************************************************

  // ok, we have a JavaScript browser
  var browserOK = false;
  var pics;

// -->
</script>

<script language="JavaScript1.1">
<!-- hide

  // JavaScript 1.1 browser - oh yes!
  browserOK = true;
  pics = new Array();

// -->
</script>
```

```
<script language="JavaScript">
<!-- hide

var objCount = 0; // number of (changing) images on web-page

function preload(name, first, second) {

  // preload images and place them in an array

  if (browserOK) {
    pics[objCount] = new Array(3);
    pics[objCount][0] = new Image();
    pics[objCount][0].src = first;
    pics[objCount][1] = new Image();
    pics[objCount][1].src = second;
    pics[objCount][2] = name;
    objCount++;
  }
}

function on(name){
  if (browserOK) {
     for (i = 0; i < objCount; i++) {
      if (document.images[pics[i][2]] != null)
        if (name != pics[i][2]) {
          // set back all other pictures
            document.images[pics[i][2]].src = pics[i][0].src;
        } else {
            // show the second image because cursor moves across this image
              document.images[pics[i][2]].src = pics[i][1].src;
        }
    }
  }
}

function off(){
  if (browserOK) {
     for (i = 0; i < objCount; i++) {
      // set back all pictures
      if (document.images[pics[i][2]] != null)
            document.images[pics[i][2]].src = pics[i][0].src;
    }
  }
}

// preload images - you have to specify which images should be preloaded
// and which Image-object on the wep-page they belong to (this is the first
// argument). Change this part if you want to use different images (of course
// you have to change the body part of the document as well)
```

```
preload("link1", "img1f.gif", "img1t.gif");
preload("link2", "img2f.gif", "img2t.gif");
preload("link3", "img3f.gif", "img3t.gif");

// -->
</script>
<head>


<body>
<a href="link1.htm" onMouseOver="on('link1')"
 onMouseOut="off()">
<img name="link1" src="link1f.gif"
 width="140" height="50" border="0"></a>

<a href="link2.htm" onMouseOver="on('link2')"
 onMouseOut="off()">
<img name="link2" src="link2f.gif"
 width="140" height="50" border="0"></a>

<a href="link3.htm" onMouseOver="on('link3')"
 onMouseOut="off()">
<img name="link3" src="link3f.gif"
 width="140" height="50" border="0"></a>
</body>
</html>
```

This script puts all images in an array pics. The *preload()* function which is called in the beginning builds up this array. You can see that we call the *preload()* function like this:

```
preload("link1", "img1f.gif", "img1t.gif");
```

This means that the script should load the two images *img1f.gif* and *img1t.gif*. The first image is the image which should be displayed when the mousecursor isn't inside the image area. When the user moves the mousecursor across the image area the second image is shown. With the first argument *"img1"* of the call of the *preload()* function we specify which Image-object on the web-page the two preloaded images belong to. If you look into the <*body*> part of our example you will find an image with the name *img1*. We use the name of the image (and not its number) in order to be able to change the order of the pictures without changing the script.
The two functions *on()* and *off()* are being called through the event-handlers *onMouseOver* and *onMouseOut*. As images cannot react to the events *MouseOver* and *MouseOut* we have to put a link around the images.
As you can see the *on()* function sets back all other images. This is necessary because it could happen that several images are highlighted (the event *MouseOut* does not occur for example when the user moves the cursor from an image directly outside the window).

Images are certainly a great way for enhancing your web-page. The Image-object lets you create really sophisticated effects. But please notice not every image and JavaScript program enhances your page. If you surf around the net you can see many examples where images are used in a horrible way. It's not the quantity of images that makes your web-page look good - it's the quality. It is really annoying to download 50 kB of bad graphics.

Keep this in mind when creating image-effects with JavaScript and your visitors/customers will come back more likely.


e-mail:skoch@rumms.uni-mannheim.de
http://rummelplatz.uni-mannheim.de/~skoch/
My JavaScript-book: http://www.dpunkt.de/javascript