Multiple image loader (for advanced users, or just take it as it is: a utility function you can reuse)

See below the 4 rectangles drawn with 4 different patterns.

DRAW WITH MULTIPLE PATTERNS? WE NEED TO LOAD ALL OF THEM BEFORE DRAWING!

We said earlier that we cannot draw before the image used by a pattern is loaded. This can become rapidly complicated if we need to draw using multiple patterns. We need a way to load all images and then, *only when all images have been loaded, start drawing*.

JavaScript is an asynchronous language. When you set the `src` attribute of an image, then an asynchronous request is sent by the browser, and then after a while, the `onload`callback is called... The difficult part to understand for those who are not familiar with JavaScript is that these requests are done in parallel and we do not know when the images will be loaded, and in which order.

SOLUTION: A MULTIPLE IMAGE LOADER THAT COUNTS THE LOADED IMAGES AND CALLS A FUNCTION YOU PASS WHEN DONE!

The trick consists in having an array of URLs that will be used by our multiple image loader, then in the `onload` callback, that will be called once per image loaded, we count the number of images effectively loaded.

When all images have been loaded, then we call a callback function that has been passed to our loader.

A complete example code that produces the result shown at the beginning of this page is at: http://jsbin.com/fufiyu/6/edit

Define the list of images to be loaded

```
// List of images to load, we used a JavaScript object instead of
// an array, so that named indexes (aka properties)
// can be used -> easier to manipulate
var imagesToLoad = {
    flowers:
       'http://cdn2.digitalartsonline.co.uk/images/features/1675/intro.jpg',
    lion: 'http://farm3.static.flickr.com/2319/2486384418_8c031fec76_o.jpg',
    blackAndWhiteLys: 'http://pshero.com/assets/tutorials/0062/final.jpg',
    tiledFloor:

'http://4.bp.blogspot.com/_Rqs7w7m37B4/TETj5rD_QmI/AAAAAAAAADk/qRiwoTO-
zKk/s1600/symmetry:assymetry+repeatable+pattern.jpg'
 };
```

Notice that instead of using a traditional array, we defined this list as a JavaScript object, with properties whose names will be easier to manipulate (flowers, lion,

tiledFloor, etc.).

The image loader function

```
function loadImages(imagesToBeLoaded, drawCallback) {
    var imagesLoaded = {};
    var loadedImages = 0;
    var numberOfImagesToLoad = 0;
    // get num of images to load
    for(var name in imagesToBeLoaded) {
        numberOfImagesToLoad++;
    }

    for(var name in imagesToBeLoaded) {
        imagesLoaded[name] = new Image();
        imagesLoaded[name].onload = function() {
            if(++loadedImages >= numberOfImagesToLoad) {
                drawCallback(imagesLoaded);
            } // if
        }; // function
        imagesLoaded[name].src = imagesToBeLoaded[name];
     } // for
} // function
```

This function takes as a parameter the list of images to be loaded, and a drawCallback function that will be called only once all images have been loaded. This callback takes as a parameter a new object that is the list of images that have been loaded (see line 16).

We first count the number of images to load (lines 7-9), then for each image to be loaded we create a new JavaScript image object (line 12) and set its src attribute (line 19), this will start to load the image.

When an image comes in, the onload callback is called (line 14) and inside, we increment the number of images loaded (line 15) and test if this number is >= of the total number of images that should be loaded. If this is the case, the callback function is called (line 16).

Example of use of this loader

```
loadImages(imagesToLoad, function(imagesLoaded) {
  patternFlowers = ctx.createPattern(imagesLoaded.flowers, 'repeat');
  patternLion    = ctx.createPattern(imagesLoaded.lion, 'repeat');
  patternBW = ctx.createPattern(imagesLoaded.blackAndWhiteLys, 'repeat');
  patternFloor   = ctx.createPattern(imagesLoaded.tiledFloor, 'repeat');
  drawRectanglesWithPatterns();
});
```

- *Line 1* is the call to the image loader, the first parameter is the list of images to be loaded, while the second parameter is the callback function that will be called once all images have been loaded.
- *Lines 2-5*: in this callback we create patterns from the loaded images (note the use of the property names imagesLoaded.flowers, etc. that makes the code easier to read).
- *Line 7*: then we call a function that will draw the rectangles.

Here is the function:

```
function drawRectanglesWithPatterns() {
  ctx.fillStyle=patternFloor;
  ctx.fillRect(0,0,200,200);
  ctx.fillStyle=patternLion;
  ctx.fillRect(200,0,200,200);
  ctx.fillStyle=patternFlowers;
  ctx.fillRect(0,200,200,200);
10.
  ctx.fillStyle=patternBW;
  ctx.fillRect(200,200,200,200);
}
```