Painting with patterns

PRINCIPLE

The principle of "pattern" drawing consists in repeating an image (if the image is smaller than the surface of the shape you are going to draw) for filling the surface of objects to be drawn (either filled or stroked).

To illustrate this principle, in the next examples, we are going to draw rectangles using this pattern:

We will need to do some steps before drawing any shapes with this pattern:

1. **Create a JavaScript image object**

```
var pattern = context.createPattern(image, "repeat|repeat-x|repeat-y|no-repeat");
```

2. **Define a callback function that will be called once the image has been fully loaded** in memory, we cannot draw before the image has been loaded.

```
imageObj.onload = function(){
```

```
...
}
```

3. **Set the source of this image to the URL of the pattern** (in our with url of the pattern),

```
imageObj.src = "http://www.myserver.com/myRepeatablePattern.png";
```

4. As soon as step 3 is executed, an HTTP request is sent in background by the browser, and when the image is loaded in memory, the callback defined at step 2 is called. We create a pattern object inside, from the loaded image:

```
// callback called asynchronously, after the src attribute of imageObj is set
imageObj.onload = function(){
    // We enter here when the image is loaded, we create a pattern object.
    // a good practice is to set this as a global variable, easier to share
    pattern1 = ctx.createPattern(imageObj, "repeat");
};
```

5. **Inside the callback function (or inside a function called from inside the callback) we can draw**.

```
// callback called asynchronously, after the src attribute of imageObj is set
imageObj.onload = function(){
    pattern1 = ctx.createPattern(imageObj, "repeat");
    // Draw a textured rectangle
    ctx.fillStyle = pattern1;
    ctx.fillRect(10, 10, 500, 800);
};
```

EXAMPLE 1: DRAW TWO RECTANGLES WITH A PATTERN (ONE FILLED, ONE STROKED)

Online example: http://jsbin.com/qezojo/1/edit

Here we have two rectangles drawn using a pattern (an image that can be repeated along the X and Y axis). The first one is a filled rectangle while the second one is "stroked" with a `lineWidth` of 10 pixels.

HTML source code:

```
<!DOCTYPE html>
<html>
<body onload="init();">
  <canvas id="myCanvas" width="500" height="400">
     Your browser does not support the canvas tag. </canvas>
  </body>
</html>
```

JavaScript source code:

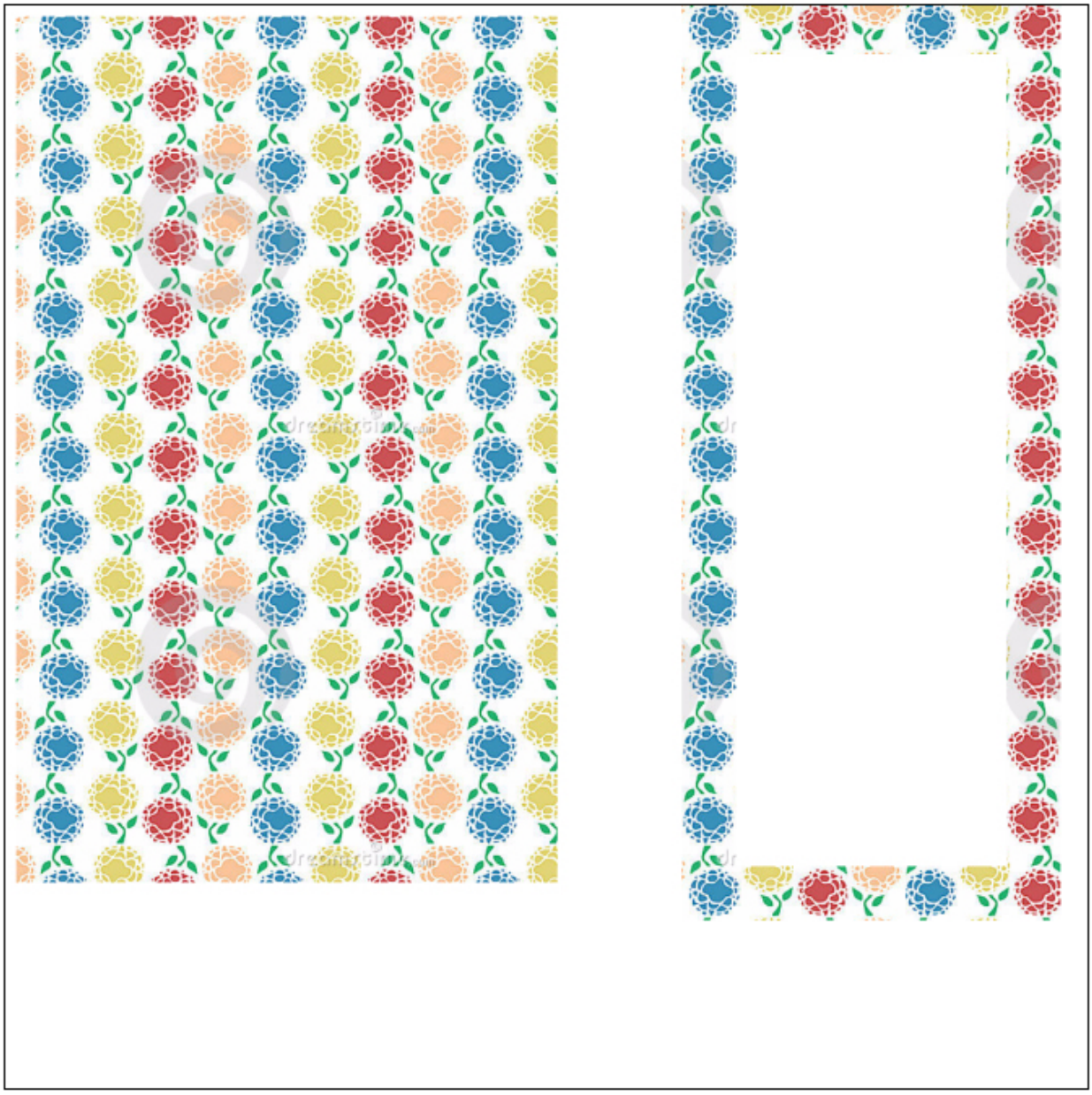
```
var canvas, ctx, pattern1;

function init() {
  canvas = document.querySelector('#myCanvas');
  ctx = canvas.getContext('2d');
  // We need 1) to create an empty image object, 2) to set a callback function
  // that will be called when the image is fully loaded, 3) to create a
  // pattern object, 4) to set the fillStyle or the strokeStyle property of
  // the context with this pattern, 5) to draw something
  // WE CANNOT DRAW UNTIL THE IMAGE IS FULLY LOADED -> draw from inside the
  // onload callback only !
  // 1 - Allocate an image
  var imageObj = new Image();

  // 2 - callback called asynchronously, after the src attribute of imageObj
```

```
  // is set
  imageObj.onload = function(){
    // We enter here only when the image has been loaded by the browser
    // 4 - Pattern creation using the image object
    // Instead of "repeat", try different values : repeat-x, repeat-y,
    // or no-repeat, You may draw larger shapes in order to see
    // different results
    // It is a good practice to leave this as a global variable if it
    // will be reused by other functions
    pattern1 = ctx.createPattern(imageObj, "repeat");
    // 5 - Draw things. Here a textured rectangle
    ctx.fillStyle = pattern1;
    ctx.fillRect(10, 10, 200, 200);
    // ... And a wireframe one
    ctx.lineWidth=20;
    ctx.strokeStyle=pattern1;
    ctx.strokeRect(230, 20, 150, 100);
  };
  // 3 - Send the request for loading the image
  // Settingthe src attribute will tell the browser to send an asynchronous
  // request.
  // When the browser will get an answer, the callback above will be called
  imageObj.src = "http://www.dreamstime.com/colourful-flowers-repeatable-pattern-thumb18692760.jpg";
}
```

EXAMPLE 2: THE REPEATABILITY OF A PATTERN

To "better" see the repeatability of the pattern, here is the same example with a 1000x1000 pixel wide canvas.

Online version here: http://jsbin.com/befiti/3/edit, and here is the result:



You can change the way the pattern is repeated by modifying the second parameter of this method:

```
pattern1 = ctx.createPattern(imageObj, "repeat");
```

Please try: `repeat-x`, `repeat-y` or `no-repeat` as acceptable values. Just change this line in the online example and you will see live results.

KNOWLEDGE CHECK 3.5.4

Patterns are images that can be used to "fill" shapes, eventually repeating themselves?

- Yes
- No
- Yes, but only with filled shapes, patterns cannot be used with the `strokeStyle` property of the context.