Drawing with HTML 5 & JavaScript

Creating CANVAS animation

HTML 5 has introduced a tag called CANVAS that allows visual element to be displayed in the browser. The CANVAS tag is very much the same as any other element used in HTML. Basically the <canvas> tag is used to rendering graphs, UI elements, and other custom graphics on the client.
The example below demonstrates the basic structure of implementing the canvas:

```
<html>
 <head>
  <script type="application/x-javascript">
   function draw() {
    var canvas = document.getElementById("myCanvas");
    if (canvas.getContext) {
     var ctx = canvas.getContext("2d");

     ……
     Shape Drawing Section
     ……

    }
   }
  </script>
 </head>
 <body onload="draw();">
  <canvas id="myCanvas" width="200" height="200"></canvas>
 </body>
</html>
```

The tag uses the new HTML5 element CANVAS as the opening and closing tag. Width and height attributes specify the size of the CANVAS space on the screen. It is the ID that is important. Here the ID is named "myCanvas." Using JavaScript, you can now program the illustration that will appear in the CANVAS tag. The examples below demonstrate shapes that may be created using this tag alongside JavaScript.

Drawing a Box

The draw function [function draw()] gets the canvas element, then obtains the 2d context. The ctx object can then be used to actually render to the canvas. The example simply fills a rectangle; by setting fillStyle to a colors using CSS color specifications and calling fillRect.

You may want to check http://color.shawnolson.net/ which allows you to select CSS valued colors.

The example below draws a blue box

```
<html>
 <head>
  <script type="application/x-javascript">
   function draw() {
    var canvas = document.getElementById("myCanvas");
    if (canvas.getContext) {
     var ctx = canvas.getContext("2d");
      ctx.fillStyle = "rgb(0,5,196)";
      ctx.fillRect (10, 20, 100, 100);
    }
   }
  </script>
 </head>
 <body onLoad="draw();">
  <canvas id="myCanvas" width="250" height="250"></canvas>
 </body>
</html>
```

Syntax

```
ctx.fillStyle = "rgb(R,G,B)";
ctx.fillRect (left, top, width, height)
```

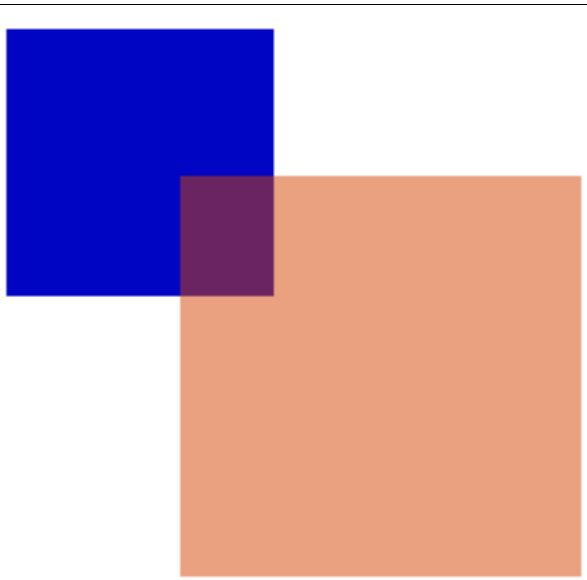
Let's introduce another box, but with opacity of 50%, the syntax for this is

```
ctx.fillStyle = "rgba(213,69,0, 0.5);";
ctx.fillRect (75, 75, 150, 150);
```

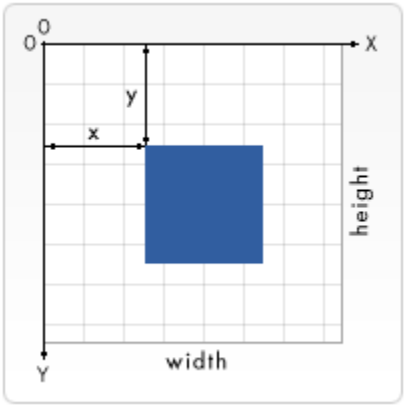
fillStyle uses rgba() to specify an alpha value along with the color (0.5).

The final code for this would be:

```
<html>
 <head>
  <script type="application/x-javascript">
   function draw() {
    var canvas =
document.getElementById("myCanvas");
    if (canvas.getContext) {
     var ctx = canvas.getContext("2d");
     ctx.fillStyle = "rgb(0,5,196)";
     ctx.fillRect (10, 20, 100, 100);
     ctx.fillStyle = "rgba(213,69,0, 0.5);";
     ctx.fillRect (75, 75, 150, 150);
    }
   }
  </script>
 </head>
 <body onLoad="draw();">
  <canvas id="myCanvas" width="250"
height="250"></canvas>
 </body>
```



Imagine creating a face? You would most likely first need to sketch the face on a grid, similar to that shown below:



The code you would need to create would be similar to that shown below.

```
ctx.arc(75,75,50,0,Math.PI*2,true); // Face
ctx.moveTo(110,75);
ctx.arc(75,75,35,0,Math.PI,false);  // Mouth
 ctx.moveTo(65,65);
ctx.arc(60,65,5,0,Math.PI*2,true);  // Left eye
ctx.moveTo(95,65);
ctx.arc(90,65,5,0,Math.PI*2,true);  // Right eye
ctx.stroke();}
```

In this example the move function takes care of moving from one coordinate to another.

So the drawing starts by 1st drawing the face, followed by moving to the end of the mouth and then drawing that, and so forth, as the diagram shows below.

You're probably thinking that it's just too much work, and imagine the time it would take to add a simple tween or animation? Yes! You're quite right, in comparison to other tools such as Flash it is a bit time consuming, but there are libraries available that help in terms of adding Animation to your Canvas. To make your life easier there is a great JavaSript library called CAKE (Canvas Animation Kit Experiment) that can be downloaded from http://code.google.com/p/cakejs/. Using the CAKE library you can easily create CANVAS based animation.

The following code demonstrates a pulsing red circle:

```
<html>
	<head>
		<script type="text/javascript" src="cake.js"></script>
		<script type="text/javascript" src="Circle.js"></script>
		<title>CAKE Programming Tutorial</title>
	</head>
	<body style="margin: 0">
	</body>
</html>
```

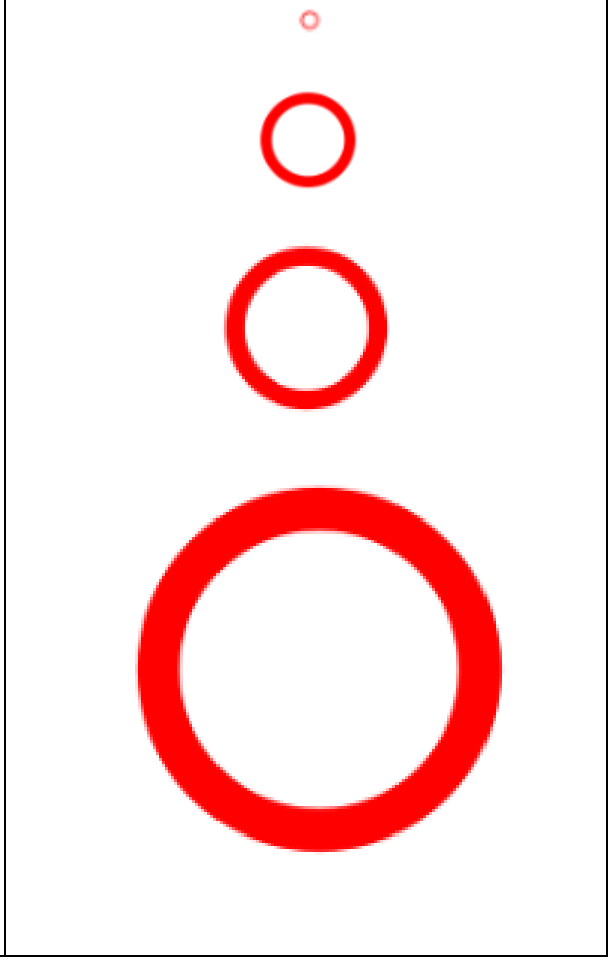
Circle.js

```
window.onload = function()
{
 var CAKECanvas = new
Canvas(document.body, 400, 400);

 var myCircle = new Circle(150,
  {
   id: 'myCircle',
   x: CAKECanvas.width / 2,
   y: CAKECanvas.height / 2,
   stroke: 'red',
   strokeWidth: 40,
   endAngle: Math.PI*2
  }
 );

 myCircle.addFrameListener(
  function(t, dt)
  {
   this.scale = Math.sin(t / 2000);
  }
 );

 CAKECanvas.append(myCircle);
};
```



In addition to this you will need to download cake.js from
http://code.google.com/p/cakejs/

To accomplish the animation you are using the SCALE method.

```
this.scale = Math.sin(t / 2000);
```

The effect is a virtually similar to what you may find from a flash animation, but with one core benefit that no plug-in is required, and more significantly it can be viewed across mobile devices that do not have any plug-in players (iPhone).

So in summary what can you do with the CANVAS?

Drawing tools

- Rectangles
- Arcs
- Paths and line drawing
- Bezier and quadratic curves

Effects

- Fills and strokes
- Shadows
- Linear and radial gradients
- Alpha transparency
- Compositing

Transformations

- Scaling
- Rotation
- Translation
- Transformation matrix

The next step you need to take is to create more complex solutions using CANVAS
Following are resources that would be helpful:

TUTORIALS: https://developer.mozilla.org/en/Canvas_tutorial
CHEAT SHEET:
http://www.nihilogic.dk/labs/canvas_sheet/HTML5_Canvas_Cheat_Sheet.pdf
GOOGLE EXPERIMENTS: http://www.chromeexperiments.com/