Slug: HTML5 for Mobile Web Applications, ISBN number, 10
kyrnin hour10-code.doc

Hour 10

Code

The <canvas> element is easy to add to your HTML documents. Just write:

```
<canvas></canvas>
```

Most of the time, you will also want to specify a width and height and give your canvas an ID so you can reference it in your scripts.

```
<canvas width="350" height="450" id="canvas1"></canvas>
```

You should also include text inside your <canvas> element to display if the element isn't supported by the browser.

```
<canvas>
This page requires an HTML5 compliant browser to render correctly.
Other browsers may not see anything.
</canvas>
```

When a user clicks on your canvas, it will draw something. For example:

```
function drawSquare() {
    var canvas = document.getElementById("canvas1");
    var context = canvas.getContext("2d");
    context.fillStyle = "rgb(13, 118, 208)";
    context.fillRect(30, 30, 150, 150);
}
```

This draws a blue square on the canvas called "canvas1". And you call that function on the <canvas> element in your document.

```
<canvas id="canvas1" width="200" height="200" onclick="drawSquare();"></canvas>
```

You can modify the above script to make the square appear and disappear.

1. Add the <canvas> element to your document

```
<canvas id="canvas1" width="200" height="200" onclick="drawSquare();">
</canvas>
```

2. Put the JavaScript in to draw the square on click

```
function drawSquare() {
    var canvas = document.getElementById("canvas1");
    var context = canvas.getContext("2d");
    context.fillStyle = "rgb(13, 118, 208)";
    context.fillRect(30, 30, 140, 140);
}
```

3. Then create an erase function in the JavaScript

```
function eraseSquare() {
    var canvas = document.getElementById("canvas1");
    canvas.width = canvas.width;
}
```

4. Call the erase function on double click in the <canvas> element

```
ondblclick="eraseSquare();"
```

this script would create a red box in the upper left, and a faded blue box in the lower right, with the overlap being purple:

```
context.fillStyle = "#ff0000";
context.fillRect(10,10, 300,300);
context.fillStyle = "rgba(0,0,255,0.5)";
context.fillRect(190,190, 300,300);
```

Here is an example of how to draw a box with a two-colored linear gradient across the diagonal:

```
var linGrad = context.createLinearGradient(0,0, 500,500);
linGrad.addColorStop(0,"red");
linGrad.addColorStop(1,"orange");

// draw gradient box
context.fillStyle = linGrad;
context.fillRect(10,10, 490,490);
```

Here is an example of how to draw a circle with a radial gradient:

```
var radGrad = context.createRadialGradient(100,150,0, 100,150,300);
radGrad.addColorStop(0.9, "rgb(105,138,72)");
radGrad.addColorStop(0, "rgba(171,235,108,1)");
radGrad.addColorStop(1, "rgba(105,138,72,0)");
```

To draw a line, you write:

```
context.beginPath();
context.moveTo(0,0);
context.lineTo(60,60);
context.stroke();
```

And to draw a triangle, you write:

```
context.beginPath();
context.moveTo(20,30);
context.lineTo(500,100);
context.lineTo(250,300);
context.fill();
```

This won't draw a perfect octagon but it will create an 8-sided figure.

1. Start your path and move your drawing point to 20,0

```
context.beginPath();
context.moveTo(200,0);
```

2. Draw a line to 40,0

```
context.lineTo(400,0);
```

3. Draw seven more lines to 60,20; 60,40; 40,60; 20, 60; 0,40; and 0,20.

```
context.lineTo(600,200);
context.lineTo(600,400);
context.lineTo(400,600);
context.lineTo(200,600);
context.lineTo(0,400);
context.lineTo(0,200);
```

4. Close your path

```
context.closePath();
```

5. Change the fill color to red

```
context.fillStyle = "#ff0000";
```

6. Fill in your octagon

```
context.fill();
```

To draw a line 2 units wide, you write:

```
context.lineWidth = "2";
```

Build a Line with the Three Join Types

1. Set your line width and begin the path

```
context.lineWidth = 15;
context.beginPath();
```

2. Move the starting point to slightly inside the canvas and draw a line

```
context.moveTo(10,20);
context.lineTo(150,200);
```

3. Draw three more lines to create a “W” shape

```
context.lineTo(290,20);
context.lineTo(430,200);
context.lineTo(570,20);
```

4. Change the line join style

```
context.lineJoin = "round";
```

5. Stroke the path

```
context.stroke();
```

The method looks like this:

```
arc(x,y,radius,startAngle,endAngle,clockwise);
```

To draw a circle, write:

```
var startPoint = (Math.PI/180)*0;
var endPoint = (Math.PI/180)*360;
context.beginPath();
```

```
context.arc(200,200,100,startPoint,endPoint,true);
context.fill();
```

Use the arc tool to build a wave pattern.

1. Decide on the diameter of your circle, and define the radius as half of that

```
var radius = 125/2;
```

2. Define the x and y coordinates of your first circle based on the radius

```
var y = radius+10;
var x1 = radius;
```

3. Define the coordinates of the subsequent circles as multiples of the radius

```
var x2 = 3*radius;
var x3 = 5*radius;
var x4 = 7*radius;
```

4. Draw the first half circle using Math.PI as your end point value to get a half circle

```
context.beginPath();
context.arc(x1,y,radius,0,Math.PI,true);
context.stroke();
```

5. Draw the second circle, only change the direction to counter-clockwise

```
context.beginPath();
context.arc(x2,y,radius,0,Math.PI,false);
context.stroke();
```

6. Repeat for as many circles as you want on your canvas

```
context.beginPath();
context.arc(x3,y,radius,0,Math.PI,true);
context.stroke();
context.beginPath();
context.arc(x4,y,radius,0,Math.PI,false);
context.stroke();
```

To draw your text on the canvas, use the fillText() method to define what and where to draw.

```
context.fillText("Hello World", 250, 100);
```

Jennifer Kyrnin
Aug 2, '11, 1:25 PM
Replaced: “

Jennifer Kyrnin
Aug 2, '11, 1:25 PM
Replaced: ”

Here is an example of how to add modified text to a <canvas> element.

```
context.font = "bold 3em/3.5em 'Palatino Linotype', 'Book Antiqua', Palatino, serif";
context.textAlign = "center";
context.textBaseline = "middle";
context.fillStyle = "#f93";
context.fillText("Hello World!", 250, 250);
```

Add a Shadow to Text

1. Add some text to your canvas

```
context.font = "bold 3em/3.5em 'Palatino Linotype', 'Book Antiqua', Palatino, serif";
context.textAlign = "center";
context.textBaseline = "middle";
context.fillStyle = "#f93";
context.fillText("Hello World!", 250, 250);
```

2. Above the text, set the x and y offsets for the shadow

```
context.shadowOffsetX = -2;
context.shadowOffsetY = 2;
```

3. Define the blur

```
context.shadowBlur = 2;
```

4. Choose a shadow color

```
context.shadowColor = "rgba(0,0,0,0.5)";
```

you can create a new image in the JavaScript.

```
var img = new Image();
img.src = "images/mydogs.png";
img.onload = function() {
    context.drawImage(img, 10,10);
}
```

You use four parameters in the drawImage method to do it:

```
context.drawImage(x, y, width, height)
```

To clip an image you need to indicate the coordinates, width and height of the area to be clipped and the coordinates, width and height where it should go on the canvas.

```
context.drawImage(clipx, clipy, clipwidth, clipheight, gox, goy, gowidth, goheight)
```

Here is how I took a photo of my dogs and created clips of their faces on the canvas:

```
var mydogs = new Image();
mydogs.src = "images/mydogs.png";
mydogs.onload = function() {
    context.drawImage(mydogs, 20,50);
    context.drawImage(mydogs, 148, 14, 92, 120, 20,370, 123,160);
    context.drawImage(mydogs, 235, 122, 65, 85, 298,370, 122,160);
}
```

You can create patterns with your images with the canvas method createPattern(). You tell it an image and how you want it patterned in your canvas.

```
context.createPattern(image, repeat-x)
```

Draw a Fancy Border on a Canvas

1. Create a new image object for your pattern

```
var pattern = new Image();
pattern.src = "images/leaf-icon.png";
```

2. Open a function when the image loads

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

3. Create a new pattern

```
    var ptn = context.createPattern(pattern,"repeat-y");
```

4. Set the fill style to the pattern

```
    context.fillStyle = ptn;
```

5. Build a rectangle and fill with the pattern

```
    context.fillRect(0,0,500,500);
```

6. Close the function

```
}
```

Exercise Answers

1. While there are many ways to create a rainbow gradient, one way is like this:

```
var canvas = document.getElementById("c");
var context = canvas.getContext("2d");

var linGrad = context.createLinearGradient(0,0, 500,500);
linGrad.addColorStop(0,"red");
linGrad.addColorStop(0.2,"orange");
linGrad.addColorStop(0.4,"yellow");
linGrad.addColorStop(0.6,"green");
linGrad.addColorStop(0.8,"blue");
linGrad.addColorStop(0.95,"violet");

// draw gradient box
context.fillStyle = linGrad;
context.fillRect(10,10, 490,490);

...
<canvas width="1000" height="500" id="c"></canvas>
```