Canvas HTML5

Week 6 Session 1

```
<canvas id="myCanvas" width="400" height="400" >
</canvas>
```

- The HTML5 <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).
- The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics.
- Canvas drawing produces a bitmap

Canvas surface

- Bitmap surface; rasterised graphic image
  - Each pixel represented by bits in an array
- Canvas is flat. Drawing to canvas updates array bits. No layers; no undo

3	3	3	3	3	3	3	3
0	1	4	1	4	1	4	0
0	4	1	4	1	4	1	0
0	5	5	5	5	5	5	0
0	5	5	5	5	5	5	0
0	1	4	1	4	1	4	0
0	4	1	4	1	4	1	0
2	2	2	2	2	2	2	2

0	000000
1	FF0000
2	00FF00
3	0000FF
4	FFFFFF
5	FFFF00
6	FF00FF
7	00FFFF
8	FF0080
9	FF8040
A	804000
B	008080
C	800000
D	800080
E	8080FF

How to use Canvas

HTML:

```
<canvas id="myCanvas" width="300" height="150">
    Your browser does not support the HTML5 canvas tag.
</canvas>
```

JavaScript

```
const canvas = document.getElementById("myCanvas");
const ctx = canvas.getContext("2d");

ctx.beginPath();
ctx.moveTo(0,0);
ctx.lineTo(300,150);
ctx.stroke();
```

Canvas draw types

- Direct Draw: Single function call to draw
  - clearRect(); fillRect(); strokeRect();
- Path Creation: Begin Path, add path points, close Path
  - Line; Arc; Bezier Curve, Quadratic Curve, Rect
- Other:
  - Text, Image, Video

Drawing Style

Try to set different colour:

ctx.strokeStyle = "red"; //rgba(255,0,0,1)

ctx.strokeStyle =”#00FF00”

ctx.strokeStyle ="rgba(144,0,0, 0.5)";

//red with 0.5 opacity

//RGB colors with opacity

Other property for line:

ctx.lineWidth = 20;

Text on canvas

```
ctx.font = “50px Verdana”;
ctx.direction = “ltr” or “rtl”;
ctx.textAlign = “left”;


ctx.fillText("My Text", 0, 0);
or
ctx.strokeText(“My Text”, 0, 0);
```

Draw a Rectangle

```
ctx.fillStyle='rgb(200,200,200)';
ctx.fillRect(50,50,400,400);
```

or

```
ctx.beginPath();
ctx.rect(x, y, w, h);
ctx.closePath();
ctx.fill();
```

ARC

arc(x, y, r, sAngle, eAngle,counterclockwise);

Definition and Usage

The arc() method creates an arc/curve (used to create circles, or parts of circles).

Tip: To create a circle with arc(): Set start angle to 0 and end angle to 2*Math.PI.

Tip: Use the stroke() or the fill() method to actually draw the arc on the canvas.

Center

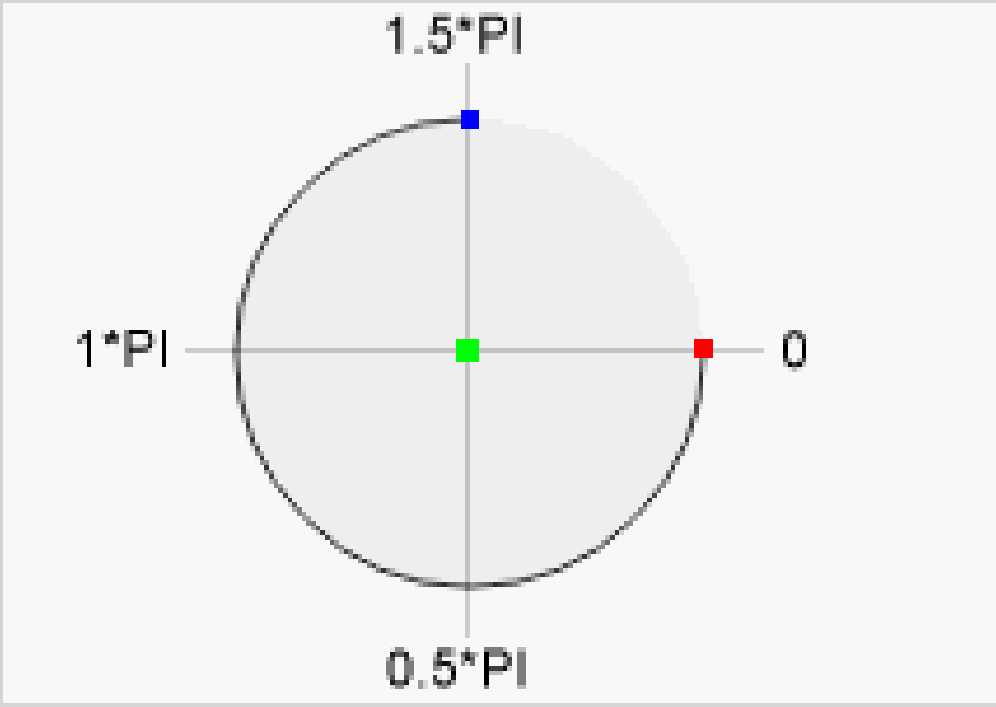
arc(**100,75**,50,0*Math.PI,1.5*Math.PI)

Start angle

arc(100,75,50,**0**,1.5*Math.PI)

End angle

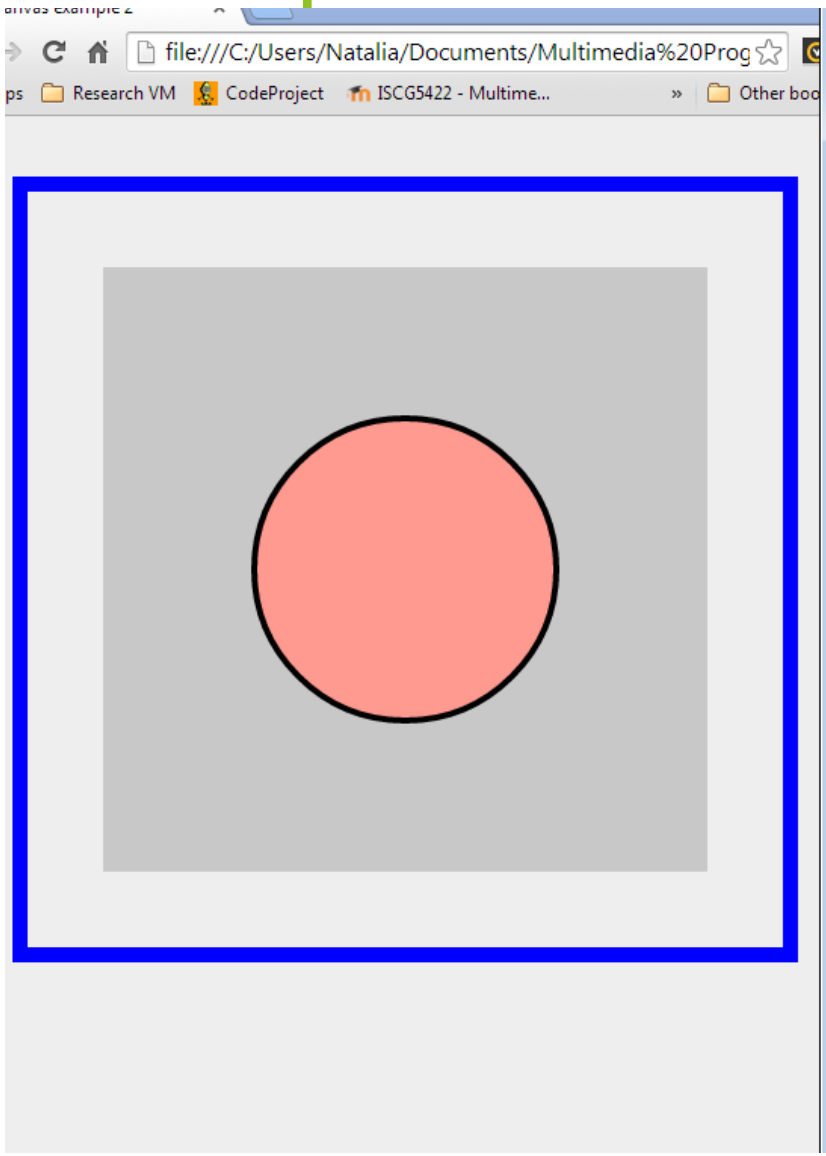
arc(100,75,50,0*Math.PI,**1.5*Math.PI**)



Circle

```
ctx.beginPath();
ctx.arc(250, 250, 100, 0, Math.PI * 2, true);
ctx.fillStyle='rgb(255,154,145)';
ctx.fill();
ctx.closePath();
ctx.lineWidth="4";
ctx.strokeStyle="black";
ctx.stroke();
```

Example:

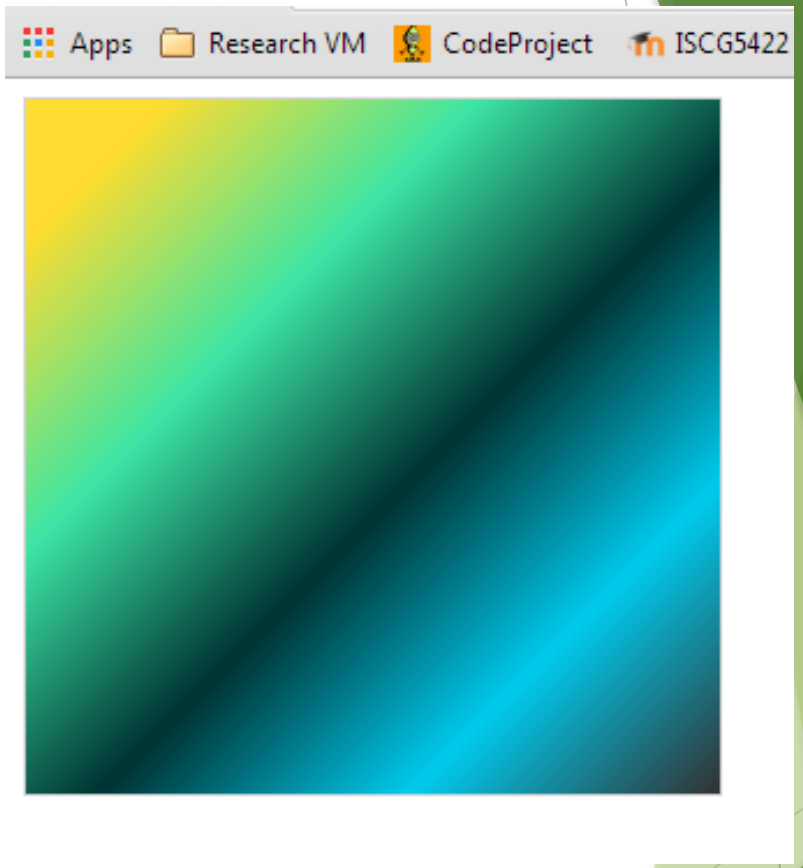


File Edit Format View Help

```
<title>canvas example 2</title>
<style>
body { background-color:#eeeeee; }
#outer {margin-left:40px;
        margin-top:40px;
       }
</style>
</head>
<body>
<div id="outer">
<canvas id="testcanvas" width="500" height="500"
style="border: 10px blue solid">
Your browser doesn't support the canvas! Try another browser.
</canvas>

</div>
<script>
var mycanvas=document.getElementById("testcanvas");
var cntx=mycanvas.getContext('2d');
//draw  rectangle

cntx.fillStyle='rgb(200,200,200)';
cntx.fillRect(50,50,400,400);

// draw  circle
cntx.beginPath();
cntx.arc(250, 250, 100, 0, Math.PI * 2, true);
cntx.fillStyle='rgb(255,154,145)';
cntx.fill();
cntx.closePath();
cntx.lineWidth="4";
cntx.strokeStyle="black";
cntx.stroke();
</script>
```

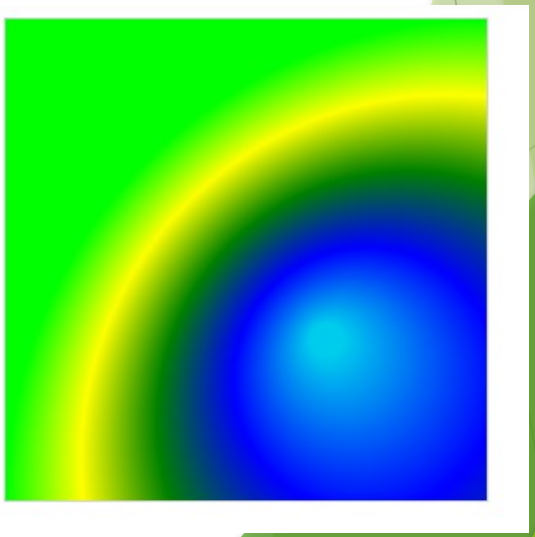
Linear Gradient

```
var mygradient=
cntx.createLinearGradient(30,30,300,300);

mygradient.addColorStop(0,"#ffdd30");
mygradient.addColorStop(0.25,"#3de6a6");
mygradient.addColorStop(0.5,"#003333");
mygradient.addColorStop(0.75,"#00ccee");
mygradient.addColorStop(1,"#333333");

cntx.fillStyle=mygradient;
cntx.fillRect(0,0,400,400);
```



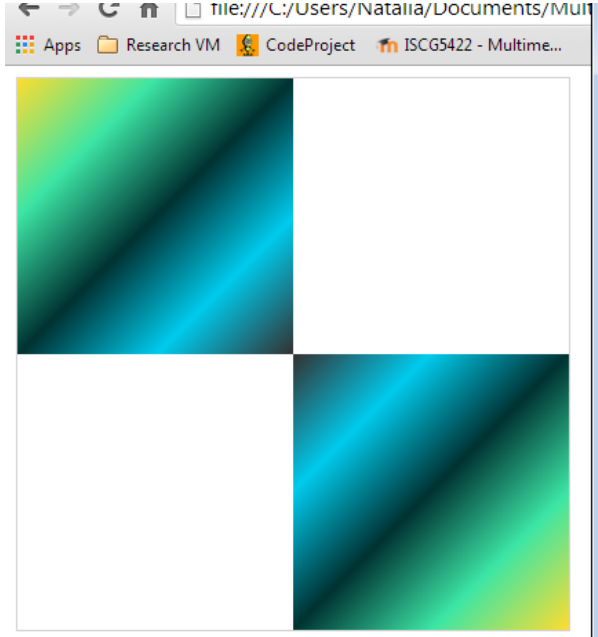
Radial Gradient

```
var mygradient=
cntx.createRadialGradient(200,200,10,300,300,300);

mygradient.addColorStop(0,"#00ccee");
mygradient.addColorStop(0.25,"blue");
mygradient.addColorStop(0.5,"green");
mygradient.addColorStop(0.75,"yellow");
mygradient.addColorStop(1,"#00ff00");
cntx.fillStyle=mygradient;
cntx.fillRect(0,0,400,400);
```



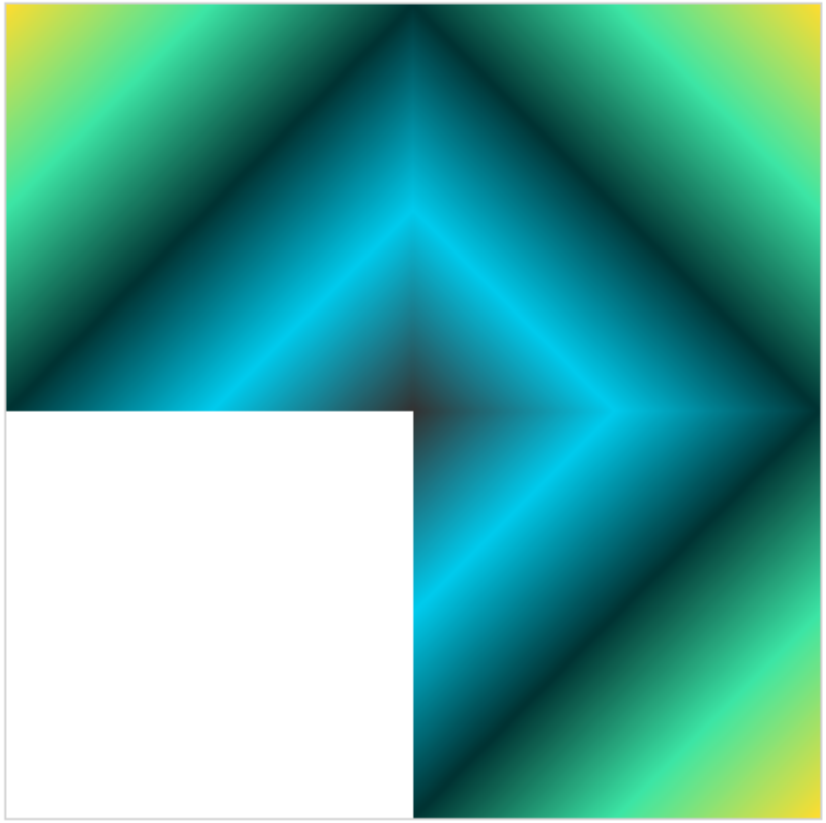
createRadialGradient(*x0,y0,r0,x1,y1,r1*);

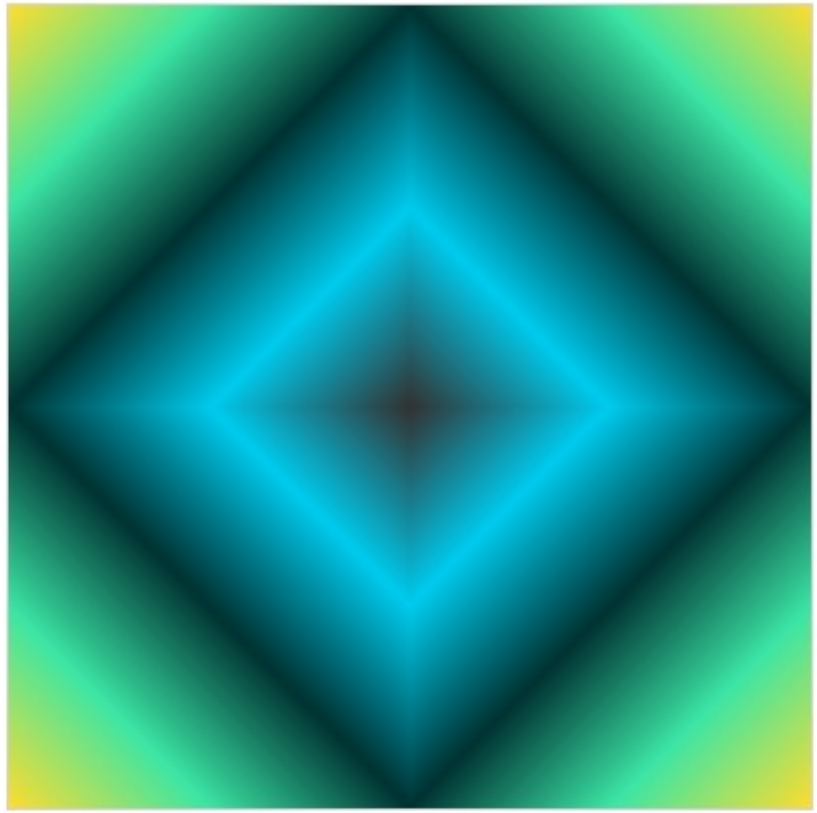
If we do two….



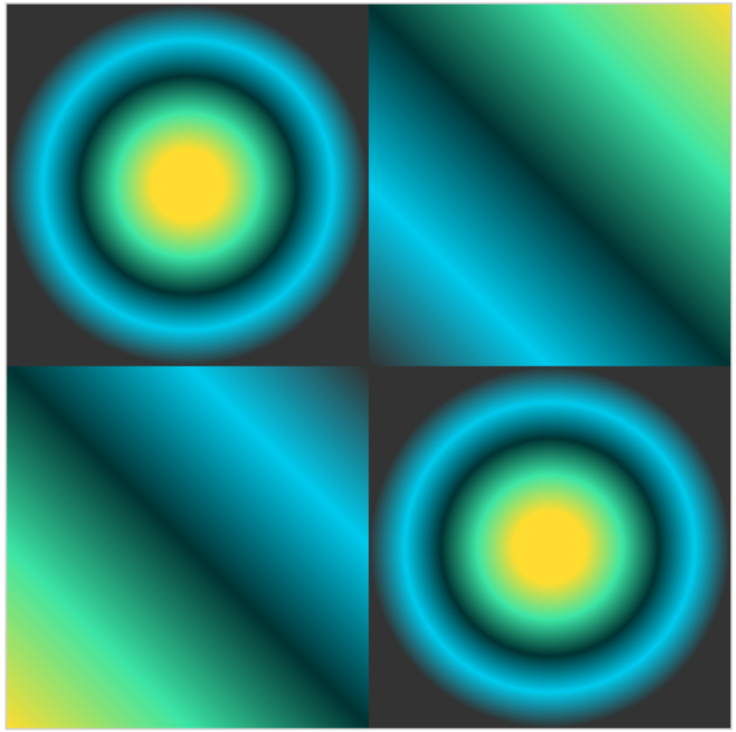

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="400" height="400" style="border:1px sol
Your browser does not support the HTML5 canvas tag.</canvas>

<script>
var c=document.getElementById("myCanvas");

var cntx=c.getContext('2d');
var mygradient=cntx.createLinearGradient(0,0,200,200);

mygradient.addColorStop(0,"#ffdd30");
mygradient.addColorStop(0.25,"#3de6a6");
mygradient.addColorStop(0.5,"#003333");
mygradient.addColorStop(0.75,"#00ccee");
mygradient.addColorStop(1,"#333333");
cntx.fillStyle=mygradient;
cntx.fillRect(0,0,200,200);

var mygradient2=cntx.createLinearGradient(400,400,200,200);
mygradient2.addColorStop(0,"#ffdd30");
mygradient2.addColorStop(0.25,"#3de6a6");
mygradient2.addColorStop(0.5,"#003333");
mygradient2.addColorStop(0.75,"#00ccee");
mygradient2.addColorStop(1,"#333333");
cntx.fillStyle=mygradient2;
cntx.fillRect(200,200,200,200);

</script>

</body>
</html>
```

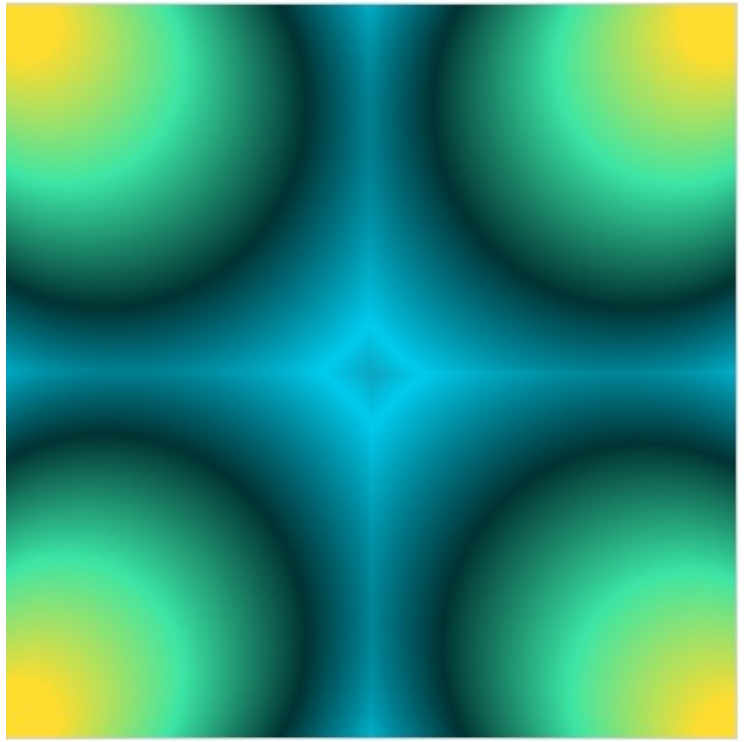
Three squares …and Four squares





Challenge: Build that patterns





For the first square:
cntx.createRadialGradient
(100,100,20,100,100,100);

For the first square:
cntx.createRadialGradient
(10,10,20,100,100,200);

Allow user to input colours as well:

Canvas:

X1: 0 Y1: 0

X2: 300 Y2: 300

Do

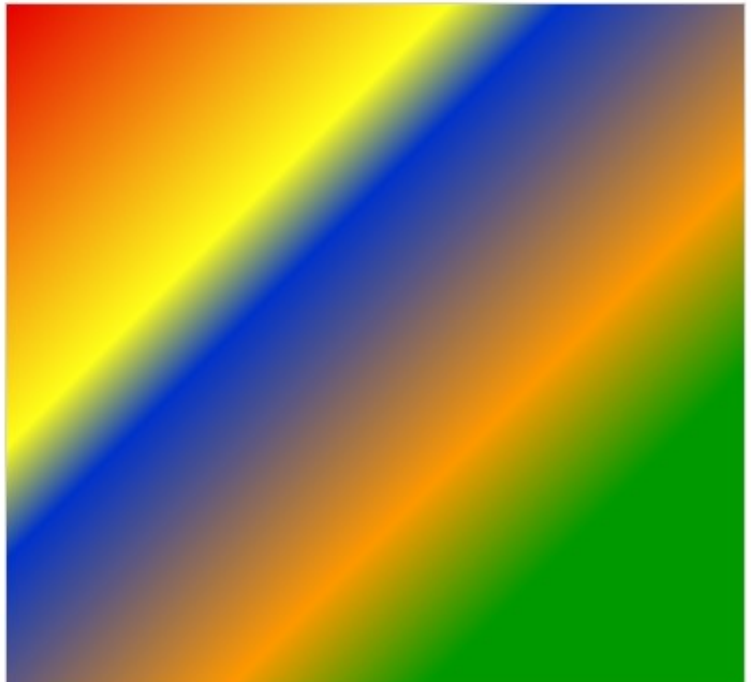
A colour picker 1:

A colour picker 2:

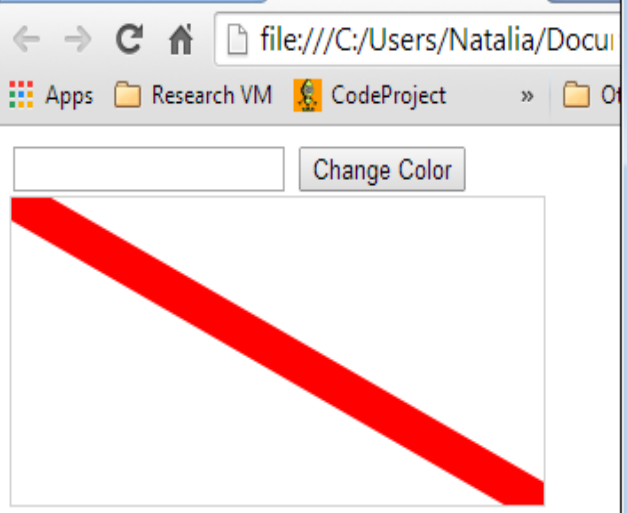
A colour picker 3:

A colour picker 4:

A colour picker 5:



```
A colour picker 3: <input type="color" id="color3" value="#0033cc"><br/>
A colour picker 4: <input type="color" id="color4" value="#ff9900"><br/>
A colour picker 5: <input type="color" id="color5" value="#009900"><br/><br/>
<canvas id="myCanvas" width="400" height="400" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>
</form>
<script>
function setColous() {
    document.getElementById("color1").value="#e60000";
    document.getElementById("color2").value="#ffff1a";
    document.getElementById("color3").value="#0033cc";
    document.getElementById("color4").value="#ff9900";
    document.getElementById("color5").value="#009900";
}
function drawR()
{

var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
//ctx.clearRect(0,0,c.width,c.height);

var x = document.getElementById("myX1").value;
var y = document.getElementById("myY1").value;
var x2 = document.getElementById("myX2").value;
var y2 = document.getElementById("myY2").value;

var c1 = document.getElementById("color1").value;
var c2 = document.getElementById("color2").value;
var c3 = document.getElementById("color3").value;
var c4 = document.getElementById("color4").value;
var c5 = document.getElementById("color5").value;
  var mygradient=ctx.createLinearGradient(x,y,x2,y2);
  mygradient.addColorStop(0,c1);
  mygradient.addColorStop(0.4,c2);
  mygradient.addColorStop(0.5,c3);
  mygradient.addColorStop(0.82,c4);
  mygradient.addColorStop(1,c5);
  ctx.fillStyle=mygradient;
  ctx.fillRect(0,0,400,400);
}
</script>
```



File Edit Format View Help

```
<!DOCTYPE html>
<html>
<body>
<input type="text" id="color1" />

<button onclick="ChangeColor();" href="javascript:;">Change Color</button>
<canvas id="myCanvas" width="300" height="150" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<script>
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.moveTo(0,0);
ctx.lineTo(300,150);
ctx.lineWidth = 20;

ctx.strokeStyle = "red";
ctx.stroke();


 function ChangeColor(){
      var newColor = document.getElementById('color1');
      alert(newColor.value);
      ctx.strokeStyle = newColor.value;
        ctx.stroke();
  }
</script>

</body>
</html>
```

Select element

```
<select id="color1">
  <option value="red">Red</option>
  <option value="green">Green</option>
  <option value="blue">Blue</option>
  <option value="yellow">Yellow</option>
</select>
```

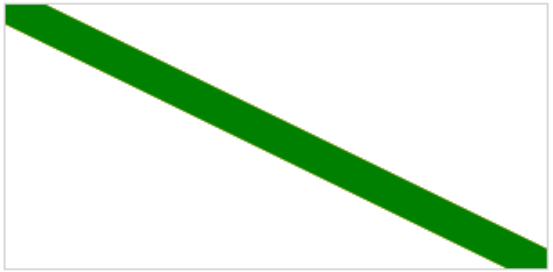
Home Insert Design Transitions Animations Slide Show

canvas_userInput_ canvas_userInput_

file:///C:/Users/Natalia/Docu

Apps Research VM CodeProject » Other bookmarks

20 Green Change Color



canvas_userInput_2.html - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<body>
<input type="text" id="size"  />
<select id="color1">
  <option value="red">Red</option>
  <option value="green">Green</option>
  <option value="blue">Blue</option>
  <option value="yellow">Yellow</option>
</select>

<button onclick="ChangeColor();" href="javascript:;">Change Color</button
<canvas id="myCanvas" width="300" height="150" style="border:1px solid #d
Your browser does not support the HTML5 canvas tag.</canvas>

<script>
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.moveTo(0,0);
ctx.lineTo(300,150);
ctx.lineWidth = 20;

ctx.strokeStyle = "red";
ctx.stroke();


 function ChangeColor(){
        var newColor = document.getElementById('color1');
        var newSize = document.getElementById('size');
        alert(newColor.value);
        ctx.strokeStyle = newColor.value;
        ctx.lineWidth = newSize.value;
        ctx.stroke();
  }
</script>
```

HTML – Canvas- Animation
How does animation work?

- Animation works by changing the properties of the element, or layer, in such small increments, that it looks smooth to the user.

- If you slow down animation, and make these increments larger, you'll actually be able to see the element jumping between the different increments.

Animation Steps

1. **Clear the canvas**
   Unless the shapes you'll be drawing fill the complete canvas (for instance a backdrop image), you need to clear any shapes that have been drawn previously. The easiest way to do this is using the clearRect() method.
2. **Save the canvas state**
   If you're changing any setting (such as styles, transformations, etc) which affect the canvas state and you want to make sure the original state is used each time a frame is drawn, you need to save that original state.
3. **Draw animated shapes**
   The step where you do the actual frame rendering.

Animation – Frame updates

- 3 techniques
  - SetInterval()
  - setTimeout()
  - requestAnimationFrame()

To Animate use Scheduled updates

- First there's the window.setInterval() and window.setTimeout() functions, which can be used to call a specific function over a set period of time.

- setInterval(*function*, *delay*) Starts repeatedly executing the function specified by function every delay milliseconds.
- setTimeout(*function*, *delay*) Executes the function specified by function in delay milliseconds.
- If you **don't want** any user interaction it's best to use the setInterval()

function which repeatedly executes the supplied code.

Animate using requestAnimationFrame

- SetInterval() and SetTimeout() can use accurate time frames, but are susceptible to issues:
  - Heavy web pages disrupt the pace.
  - Continues running when unfocused or minimised – bad for battery life of devices.

- Newer, better way to update your canvas: requestAnimationFrame()
- Called using window.requestAnimationFrame(callback);
- Limited to 1 execution per second when minimised.

- https://flaviocopes.com/requestanimationframe/#optimization

Example

```
window.onload = init;

function init() {
    canvas = document.getElementById('myCanvas');
    context = canvas.getContext('2d');

    canvasWidth = canvas.clientWidth;
    canvasHeight = canvas.clientHeight;

    // Start the first frame request
    window.requestAnimationFrame(gameLoop);
}

function gameLoop(timeStamp) {
    // Calculate fps
    fps = Math.round(1 / secondsPassed);
    // update elements of the animation
    update(secondsPassed);
    // Perform the drawing operation
    draw();
    // The loop function has reached it's end. Keep requesting new frames
    window.requestAnimationFrame(gameLoop);
}
```

Or https://glitch.com/edit/#!/flavio-requestanimationframe-example?path=script.js%3A1%3A0

Updating on user interaction

- Other way to control an animation is user input.
- If we wanted to make a game, we could use keyboard or mouse events to control the animation. By setting EventListeners, we catch any user interaction and execute our animation functions.

Example

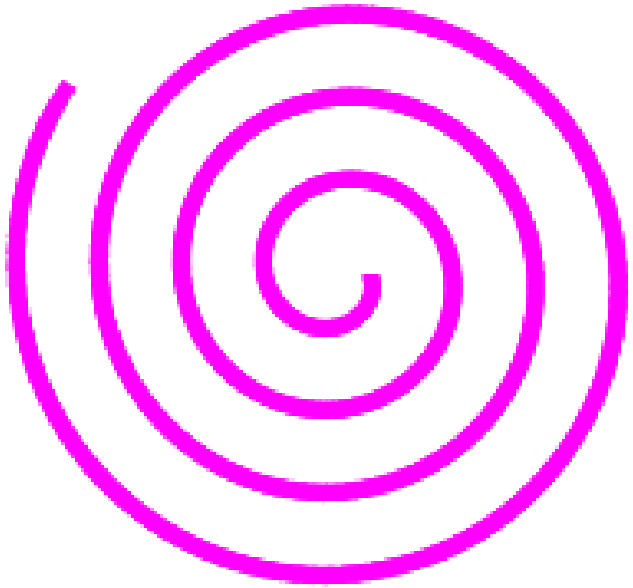
```
<body onLoad="init();">
    <canvas id="myCanvas" width="400" height="400" >
    </canvas>

    <script>
        let context;

        function init() {
            context= myCanvas.getContext('2d');
            ...
        }
    <script>
</body>
```

Overdraw or Clear

- When drawing more than once, we need to decide whether to add to an existing drawing, or clear the canvas and draw afresh
- context.clearRect(0,0, width, height);
  - Reset Canvas bitmap

Different Animations:

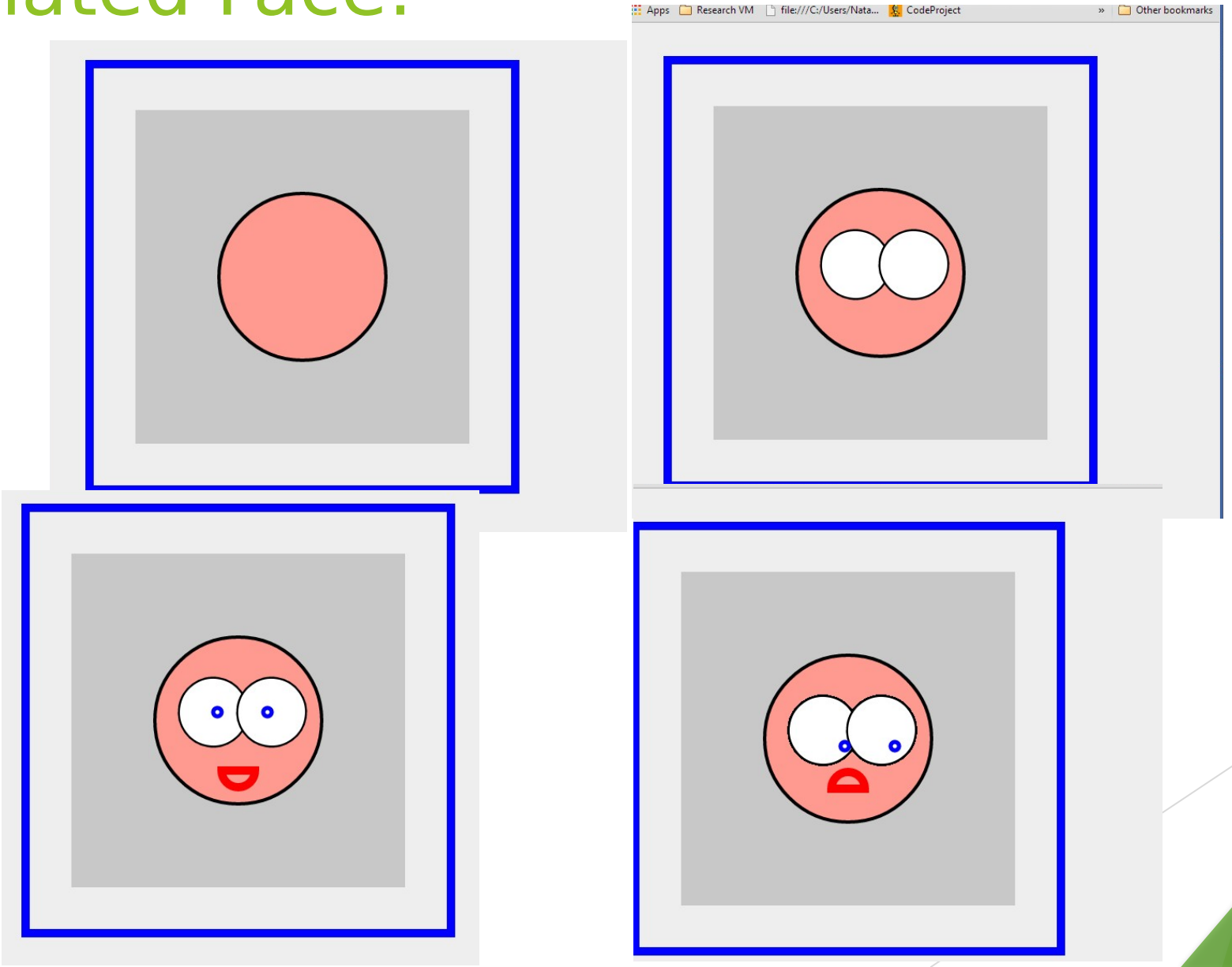
```
function draw()
{
        context.beginPath();
        context.lineWidth=5;
        context.strokeStyle= col;
        context.arc(x, y, rad , c1, c2, false);
        context.stroke();

        rad = rad + radOffset;
        c1 = c1 + cOffset;
        c2 = c2 + cOffset;
        if (rad > 100) {
                clearInterval(myMove);
        }
}
```



If we want to stop animation

```
var myMove;
context= myCanvas.getContext('2d');

function init()
{
  myMove=setInterval(draw,20);
}
function draw()
{
        //......
        //....
        if (rad>100) {
                clearInterval(myMove);
```

Animated Face:



Resources:

- https://testdrive-archive.azurewebsites.net/Graphics/CanvasPad/Default.html
- http://www.w3schools.com/html/html5_canvas.asp
- https://jschollitt.github.io/week6.html