HTML

Canvas Programming

WJGuo

2012-05-15

QQ:411704005

An overview of Canvas

<canvas> </canvas>

* Openly developed as a W3C spec
  - *First created by Apple for dashboard widgets*
* 2D & 3D drawing platform within the browser
  - *Before that, developers can only use Flash or SVG to draw in browser*
  - *3D drawing supported by most modern browsers except for IE*
* No plugins
  - *Uses nothing more than JavaScript and HTML*
* Extensible through a Javascript API
  - *WebGL & other functional Javascript libraries*
* A bitmap system
  - *Everything is drawn as a single, flat, picture*
  - *Changes require the whole picture to be redrawn*

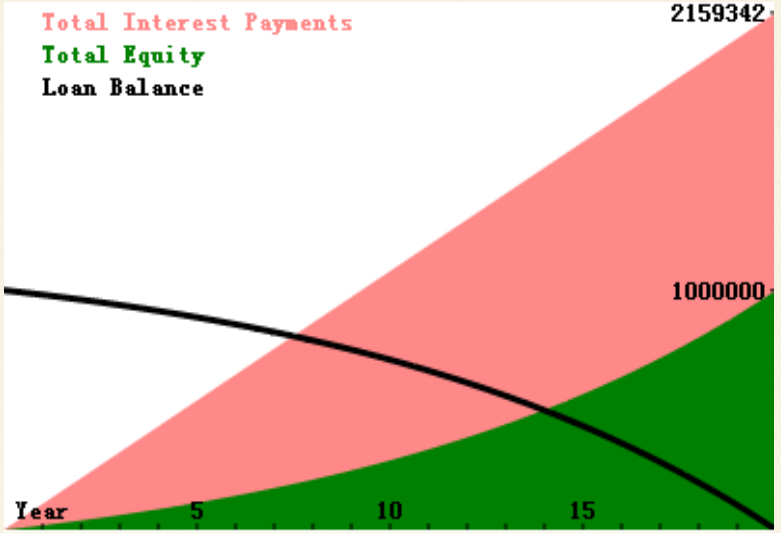
What is Canvas for?

* Data visualization
* Animated graphics

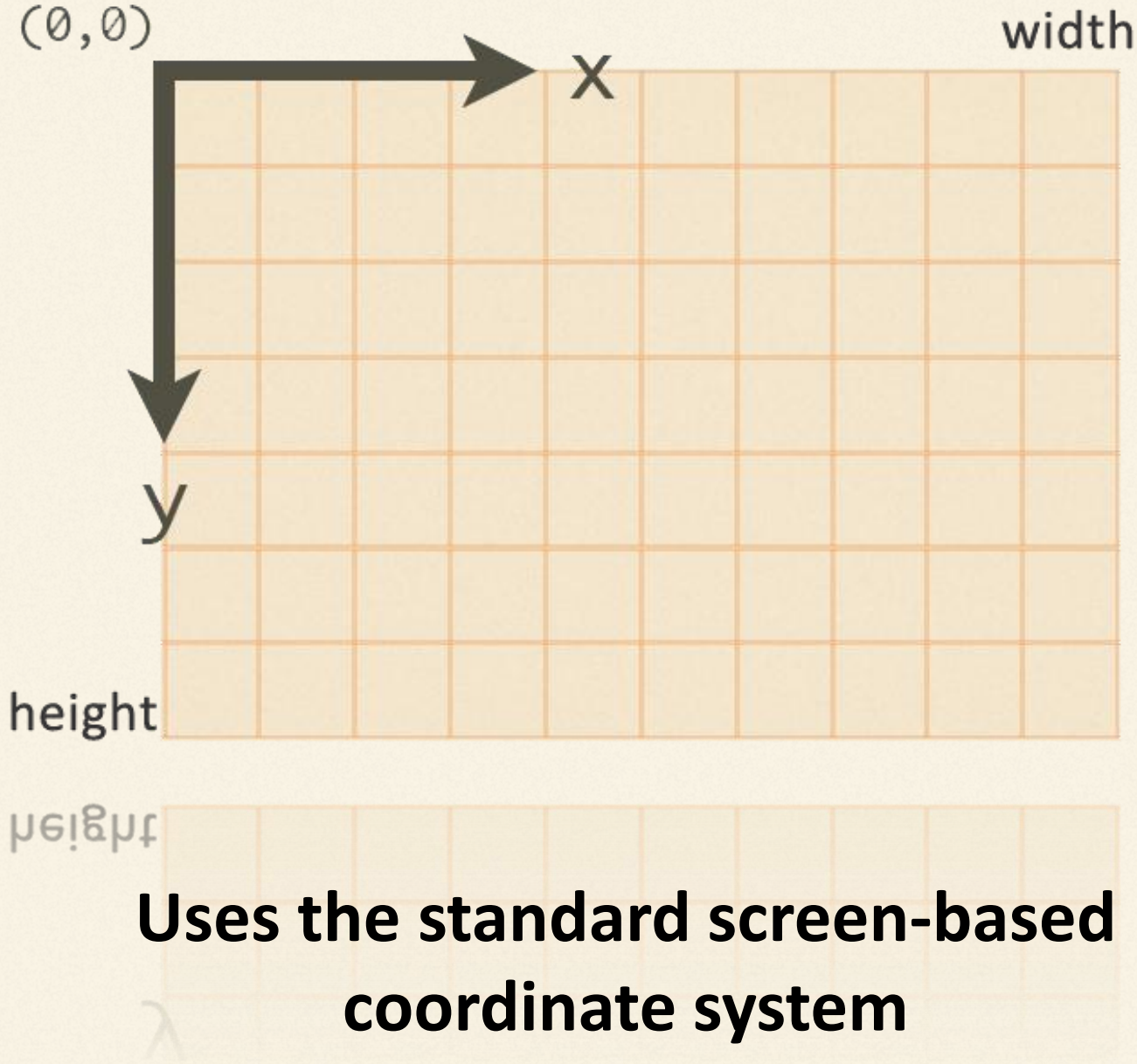
  http://workshop.chromeexperiments.com/
* Web applications

  http://mudcu.be/sketchpad/
* Games

  http://www.spacegoo.com/maze/
* Others



Getting started



Uses the standard screen-based coordinate system

Basic usage

HTML code:

```
<canvas id="graph" width="150" height="150"></canvas>

<canvas id="clock" width="150" height="150">
        <!– code or text for unsupported browsers -->
        Update you browser to enjoy canvas!
</canvas>
```

Javascript code:


```
var canvas = document.getElementById('graph');
if (canvas.getContext){
        var ctx = canvas.getContext('2d');
        // drawing code here
} else {
        // canvas-unsupported code here
}
```

Skeleton template

```
<html>
  <head>
    <title>Canvas graph</title>
    <script type="text/javascript">
      function draw(){
        var canvas = document.getElementById('graph');
        if (canvas.getContext){
          var ctx = canvas.getContext('2d');
          // your drawing code here
        }
      }
    </script>
    <style type="text/css">  canvas { border: 1px solid black; }  </style>
  </head>
  <body onload="draw();">
    <canvas id="graph" width="150" height="150"></canvas>
  </body>
</html>
```

Simple shapes

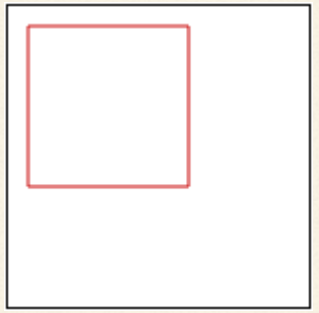
```
// style of shapes to be drawn
ctx.fillStyle = 'rgb(255, 0, 0)';
ctx.strokeStyle = 'rgba(0, 255, 0, 0.5)';
```

Method	Action
fillRect(x, y, w, h**)**	Draws a rectangle using the current fill style
strokeRect(x, y, w, h**)**	Draws the outline of a rectangle using the current stroke style
clearRect(x, y, w, h**)**	Clears all pixels within the given rectangle

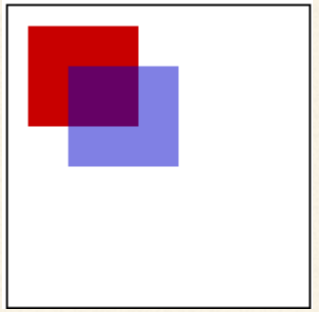
Simple shapes are drawn without effecting the current path

examples

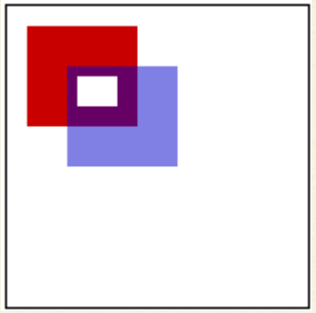
```
ctx.strokeStyle = "rgb(200,0,0)";
ctx.strokeRect (10, 10, 80, 80);
```



```
ctx.fillStyle = "rgb(200,0,0)";
ctx.fillRect (10, 10, 55, 50);
ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
ctx.fillRect (30, 30, 55, 50);
```



```
ctx.fillStyle = "rgb(200,0,0)";
ctx.fillRect (10, 10, 55, 50);
ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
ctx.fillRect (30, 30, 55, 50);
ctx.clearRect (30, 30, 20, 15);
```



Complex shapes & paths

* Paths are a list of subpaths
* Subpaths are one or more points connected by straight or curved lines
* Rendering context always has a current path
* A new path should be created for each individual shape

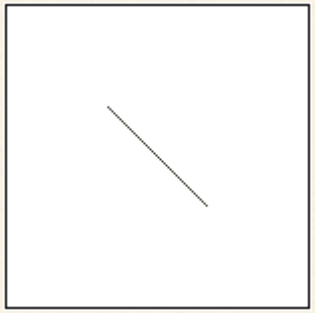
Method	Action
beginPath()	Resets the current path
closePath()	Closes the current subpath and starts a new one
moveTo(x, y**)**	Creates a new subpath at the given point
fill()	Fills the subpaths with the current fill style
stroke()	Outlines the subpaths with the current stroke style

Complex shapes & paths

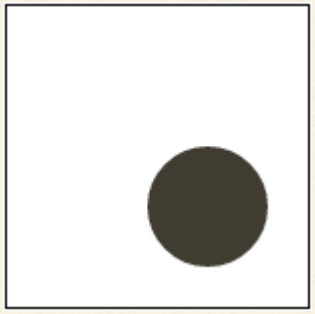
Method	Action
lineTo(x, y**)**	Draws a straight line from the previous point
rect(x, y, w, h**)**	Adds a new closed rectangular subpath
arc(x, y, radius, startAngle, endAngle, anticlockwise**)**	Adds a subpath along the circumference of the described circle, within the angles defines
arcTo(x1, y1, x2, y2, radius**)**	Adds a subpath connecting two points by an arc of the defined radius
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y**)**	Adds a cubic Bézier curve with the given control points
quadraticCurveTo(cpx, cpy, x, y**)**	Adds a quadratic Bézier curve with the given control points

examples

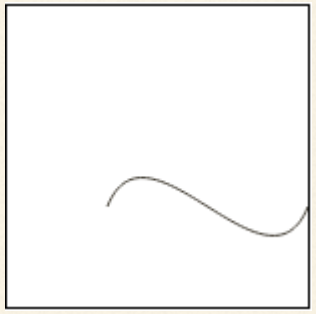
```
ctx.strokeStyle = 'rgb(65, 60, 50)';
ctx.beginPath();
ctx.moveTo(50, 50);
ctx.lineTo(100, 100);
ctx.stroke();
```



```
ctx.fillStyle = 'rgb(65, 60, 50)';
ctx.beginPath();
ctx.arc(100, 100, 30, 0, Math.PI*2, true);
ctx.fill();
```



```
ctx.strokeStyle = 'rgb(65, 60, 50)';
ctx.beginPath();
ctx.moveTo(50, 100);
ctx.arc(70, 50, 130, 150, 150, 100);
ctx.fill();
```

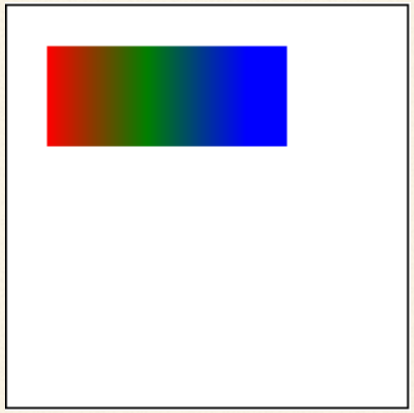


Gradient & Shadow

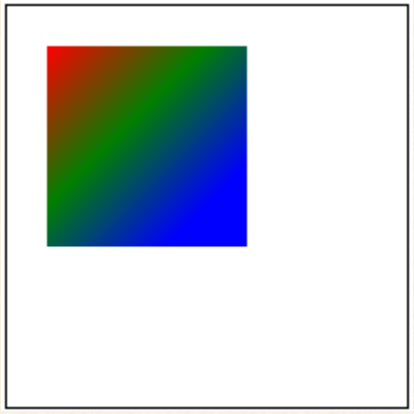
Method	Action
createLinearGradient(x1,y1,x2,y2**);**	Create a linear gradient from point (x1,y1) to point (x2,y2)
createRadialGradient(x1,y1,r1,x2,y2,r2**)**	Create a radical gradient from point (x1,y1,r1) to point (x2,y2,r2)
addColorStop(x,"green"**);**	Set coordinate offset and color offset. x can take any value between 0 and 1.
shadowOffsetX **shadowOffsetY** **shadowColor** **shadowBlur**	All drawing operations are affected by the four global shadow attributes. Shadows are only drawn if the **opacity component of the alpha component of the color of shadowColor is non-zero** and **either the shadowBlur is non-zero, or the shadowOffsetX is non-zero, or the shadowOffsetY is non-zero**.

examples

```
var lgt = ctx.createLinearGradient(20,20,120,20);
lgt.addColorStop("0","magenta");
lgt.addColorStop(".50","green");
lgt.addColorStop("1.0","red");
ctx.fillStyle = lgt;
ctx.fillRect(20,20,120,50);
```

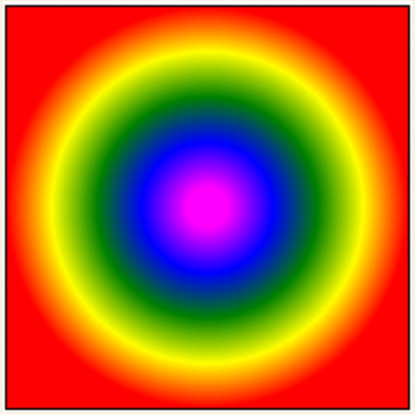


```
var lgt = ctx.createLinearGradient(50,50,100,100);
lgt.addColorStop("0","magenta");
lgt.addColorStop(".50","green");
lgt.addColorStop("1.0","red");
ctx.fillStyle = lgt;
ctx.fillRect(50,50,100,100);
```



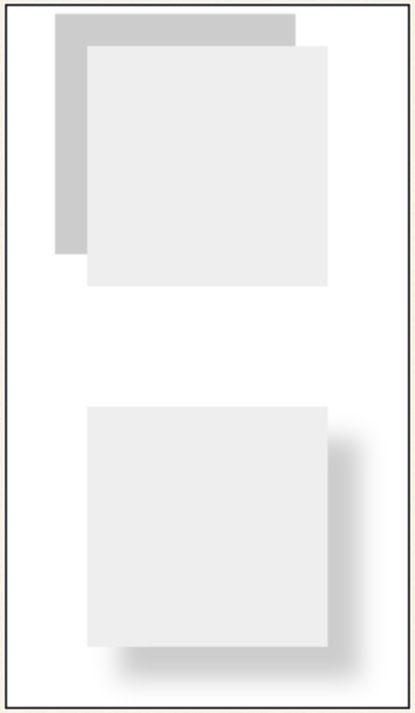
examples

```
var x = canva.width/2;
var y = canva.height/2;
var r = (x<y)?x:y;
// create gradient object
var rgt = ctx.createRadialGradient(x,y,10,x,y,r);
rgt.addColorStop("0","magenta");
rgt.addColorStop(".25","blue");
rgt.addColorStop(".50","green");
rgt.addColorStop(".75","yellow");
rgt.addColorStop("1.0","red");
ctx.fillStyle = rgt;
ctx.fillRect(0,0,canva.width,canva.height);
```



examples

```
// shadow to top left
ctx.shadowOffsetX = -16;
ctx.shadowOffsetY = -16;
ctx.shadowColor = "#ccc";
ctx.shadowBlur = 1;
ctx.fillStyle = "#eee";
ctx.fillRect(40,20,120,120);

// shadow to bottom right
ctx.shadowOffsetX = 16;
ctx.shadowOffsetY = 16;
ctx.shadowColor = "#ccc";
ctx.shadowBlur = 20;
ctx.fillStyle = "#eee";
ctx.fillRect(40,200,120,120);
```



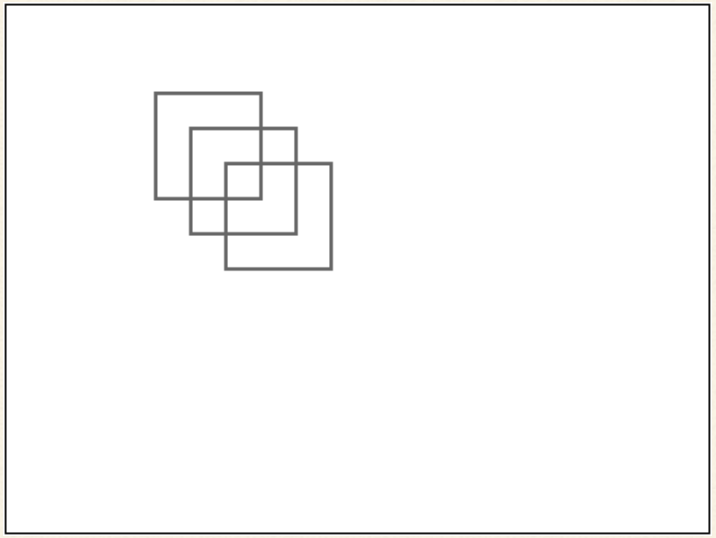
Text drawing

```
function textDraw(ctx,font,x,y,fill){
  var text = "Canvas";
  ctx.font =font;
  ctx.textAlign = "center";
  ctx.textBaseline ="bottom";
  if(fill){
    ctx.fillStyle = "#ccc";
    ctx.fillText(text,x,y);
  } else {
    ctx.strokeStyle = "#666";
    ctx.strokeText(text,x,y);
  }
}
```

```
textDraw(ctx,"bold 35px impact",90,70,false);
textDraw(ctx,"bold 35px arial blank",130,110,true);
textDraw(ctx,"bold 35px comic sans ms",170,150,true);
```

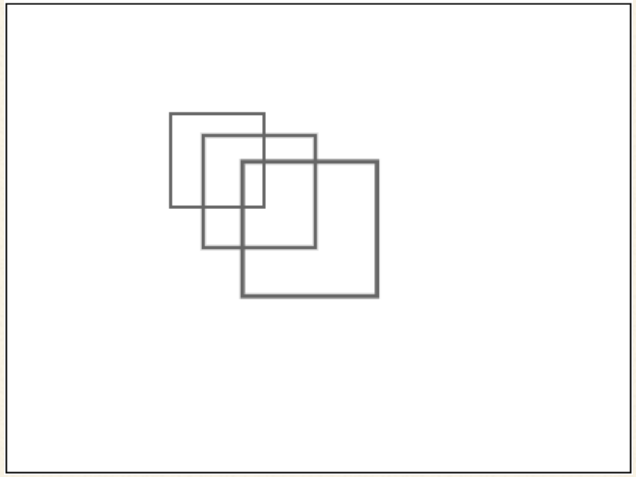
Transformations

```
// draw rectangle
function drawRect(){
  canva = document.getElementById("graph");
  ctx = canva.getContext("2d");
  ctx.strokeStyle = "#666";
  ctx.lineWidth = 2;
  ctx.strokeRect(105,70,60,60);
}

// move rectangle
function translateRect(){
  canva = document.getElementById("graph");
  ctx = canva.getContext("2d");
  drawRect();
  ctx.translate(-20,-20);
  drawRect();
  ctx.translate(40,40);
  drawRect();
}
```

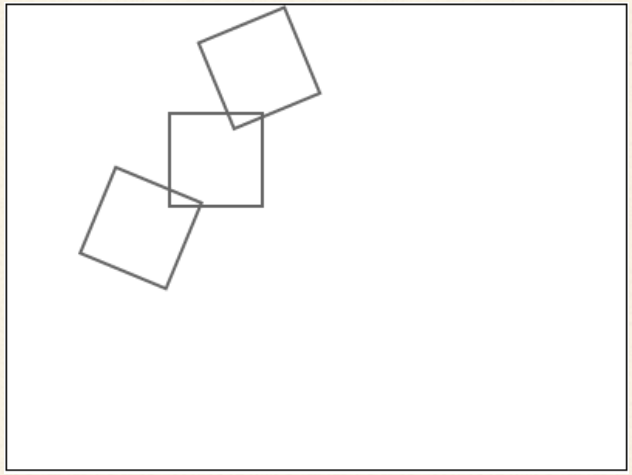


Transformations

```
// scale rectangle
function scaleRect(){
  canva = document.getElementById("graph");
  ctx = canva.getContext("2d");
  drawRect();
  ctx.scale(1.2,1.2);
  drawRect();
  ctx.scale(1.2,1.2);
  drawRect();
}
```



```
// rotate rectangle
function rotateRect(){
  canva = document.getElementById("graph");
  ctx = canva.getContext("2d");
  drawRect();
  ctx.rotate(Math.PI/8);
  drawRect();
  ctx.rotate(-Math.PI/4);
  drawRect();
}
```



Animation

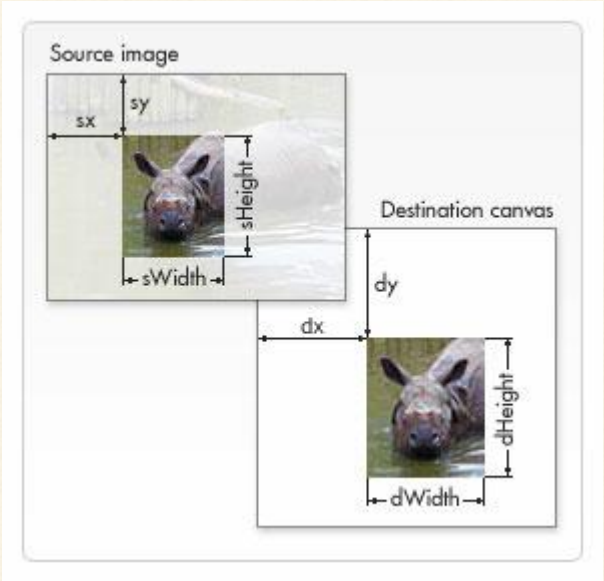
```
var ii,jj,xx;

function simpleAnimation(){
  canva = document.getElementById("graph");
  ctx = canva.getContext("2d");
   drawFace(ctx);
  ii = 1;
  jj = 21;
  setInterval(moveFace,100);
}
 // move face
function moveFace(ctx,color,x,y,r,h,fill){
  var canva = document.getElementById("graph");

  ctx = canva.getContext("2d");
   ctx.clearRect(0,0,280,190);
  if(ii<20) { ii += 1; xx = ii;
  } else {    if(jj>0){ jj -= 1; xx = -jj;  }   }
  ctx.translate(xx,0);
  drawFace(ctx);
}
```

```
// draw face element
function drawCirc(ctx,color,x,y,r,h,fill){
  ctx.beginPath();
  ctx.arc(x,y,r,0,Math.PI*h,fill);
  if(fill){     ctx.fillStyle = color;     ctx.fill();
  } else {  ctx.lineWidth = 2;
    ctx.strokeStyle = color;     ctx.stroke();   }
  ctx.closePath();
}
// draw face
function drawFace(ctx){
  drawCirc(ctx,"#666",30,80,30,2,true);
  drawCirc(ctx,"#fff",20,70,5,2,true);
  drawCirc(ctx,"#fff",40,70,5,2,true);
  drawCirc(ctx,"#fff",30,80,18,1,false);
}
```

Image drawing

```
var img = new Image();
img.onload = function(){
  drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight);
};
```

```
// images in same page
var img = document.getElementById('frame') ;
// creating images  from scratch
var img = new Image();
img.src = 'images/rhino.jpg';
// use data URL
img.src = '
AIAAAAAA3pn/ZiH5BAEAAAEALAAAAAALAAsAA
AIUhA+hkcuO4lmNVindo7qyrIXiGBYAOw==';
// use images from other domain
```



Other image operate

Image fillstyle:

```
var img=new Image();
img.src="img.hippo.png";
var pattern=ctx.createPattern(img,repeat-x);
img.onload=function(){
  ctx,fillStyle=pattern;
  ctx.fillRect(0,0,canva.width,canva.height);
}
```

Operate pixels:

```
ctx.getImageData(sx,sy,sw,sh);
ctx.putImageData(imgData,dx,dy);
```

Save to base64:

```
canva.toDataURL("image/png");
```

Save & restore:

```
ctx.save();
ctx.restore();
```

Reference sites

http://www.w3.org/TR/2dcontext/

https://developer.mozilla.org/cn/HTML/Canvas

http://dev.opera.com/articles/view/html-5-canvas-the-basics/

http://msdn.microsoft.com/en-us/library/ie/hh771733%28v=vs.85%29.aspx