HTML5 canvas

<canvas>

HTML5 canvas, intro

What is canvas?

The canvas element is a "drawable region defined in HTML code with height and width attributes. JavaScript code may access the area through a full set of drawing functions similar to other common 2D APIs, thus allowing for dynamically generated graphics. Some anticipated uses of canvas include building graphs, animations, games, and image composition."

Canvas markup:

```
<canvas id="canv" width="400" height="200">
    <!-- message for legacy browsers -->
    <p>Hey, time to update your browser!</p>
</canvas>
```

HTML5 canvas, support

Do not forget Modernizr

Load the latest version of modernizr in the head tag:

```
<script src="assets/js/modernizr-latest.js"></script>
```

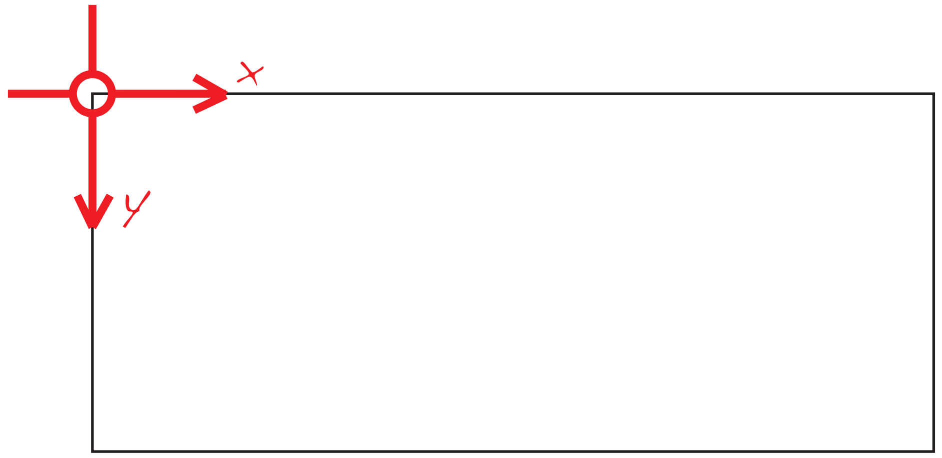
Check for support:

```
//checking function
var supportCanvas = (function(){return !!Modernizr.canvas})();

//usage:
if(supportCanvas){
    //can use canvas api
}
```

HTML5 canvas, stage and context

Canvas stage

The canvas space has something in common with stage in flash: the origin is the top-left corner.



Get canvas and context:

```
//get the canvas element
var canvas = document.getElementById('canvas');

//you need to get the canvas context before you start drawing:
var context = canvas.getContext('2d');
```

HTML5 canvas, draw rectangles

Rectangles:

```
ctx.beginPath();                //starts a new path
ctx.rect(x, y, w, h);           //creates a rectangle path
ctx.fillStyle = col;            //sets fill colour
ctx.fill();                     //fills in the path
ctx.lineWidth = str;            //sets stroke width
ctx.strokeStyle = strokeCol;    //sets stroke colour
ctx.stroke();                   //draws the stroke
```

rect() or **fillRect()** will ask for the position of the top left corner (x, y) then the size of the rectangle (width, height)

fill() fills in the shape with the colour set before with **fillStyle()**

stroke() will set the draw the border/outline of the shape with the colour previously set with **strokeStyle()**

**if not set, default color is black*

HTML5 canvas, draw circles and arcs

Circles:

```
context.beginPath();                              //begins a new path
context.arc(x, y, w, 0, 2*Math.PI, false);        //draws an arc
context.fillStyle = col;                          //sets fill colour
context.fill();                                   //fills in the path
context.lineWidth = str;                          //sets stroke width
context.strokeStyle = strokeCol;                  //sets stroke colour
context.stroke();                                 //draws the stroke
```

arc() will ask for the position of the centre (x, y), the radius, the starting and ending points in radiants, the direction of filling:

context.**arc(** **x:**centreX,
y:centreY,
radius:circleRadius,
startAngle:startsFrom,
endAngle:endsTo,
antiClockwise:bool**)**

v

HTML5 canvas, shadows and canvas clear

Shadows:

```
context.shadowColor = "#999";     //colour of the shadow
context.shadowBlur = 20;          //blur
context.shadowOffsetX = 15;       //horizontal offset
context.shadowOffsetY = 15;       //vertical offset
```

**Do not use shadows and outline at the same time*

Clear the canvas:

```
function clearCanvas(canv, context){
   // area to clear (rectangle)
   context.clearRect(0, 0, canv.width, canv.height);
   // restore ctx
   context.restore();
}//end clearCanvas
```

**This will clear a rectangle area starting from (0,0) with same size of canvas*

HTML5 canvas, text

Text:

```
//same order as css: style, weight, size, and family
context.font = "italic bold 60px Helvetica";   //set font style
context.textAlign = "center";                  //set alignment
context.fillStyle = col;                       //set text colour
context.fillText(txt, x, y, [maxWidth]);       //draw text

//or

context.strokeStyle = col;                  //set text outline colour
ctx3.strokeText(txt, x, y, [maxWidth]);     //draw text outline only
```

**using maxWidth, the text will scale automatically to fit the max size*

HTML5 canvas, custom shapes

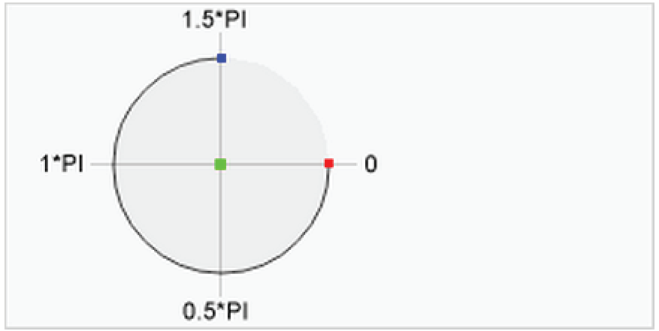
Custom shapes:

```
ctx.beginPath()                 //starts a new path
ctx.moveTo(x,y)                 //moves to starting point
ctx.[ lineTo(x,y) ],            //draws a line to a new point

//draws an arc between two points, with given radius
ctx.[ arcTo(x1,y1,x2, y2,radius) ],

//draws a curve to new point using a control point (cp)
ctx.[ quadraticCurveTo(cpx,cpy,x,y) ],

//draws a curve to new point using two control point (cp1, cp2)
ctx.[ bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y) ]

ctx.closePath();                //close the path and styles the shape
->ctx.lineWidth = *;
->ctx.fillStyle = *;
->ctx.fill();
->ctx.strokeStyle = *;
->ctx.stroke();
```

HTML5 canvas, curves

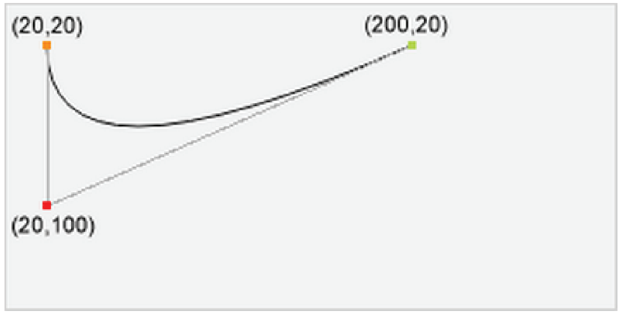
Curves:

ctx.arc()



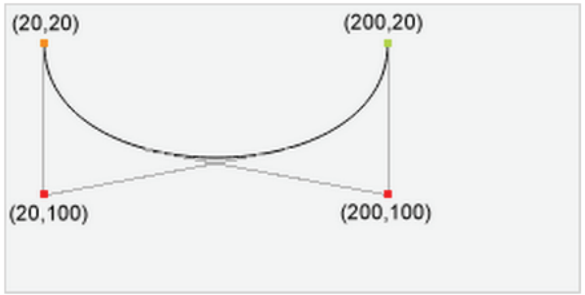
🟥	Startangle	arc(100,75,50,**0*Math.PI**,1.5*Math.PI)
🟦	Endangle	arc(100,75,50,0*Math.PI,**1.5*Math.PI**)
🟩	Center	arc(**100,75**,50,0*Math.PI,1.5*Math.PI)

ctx.quadraticCurveTo()



🟧	Current point	moveTo(**20,20**)
🟩	CurveTo point	quadraticCurveTo(20,100,**200,20**)
🟥	Controllpoint	quadraticCurveTo(**20,100**,200,20)

ctx.bezierCurveTo()



🟧	Current point	moveTo(**20,20**)
🟩	CurveTo point	bezierCurveTo(20,100,200,100,**200,20**)
🟥	Controllpoint 1	bezierCurveTo(**20,100**,200,100,200,20)
🟥	Controllpoint 2	bezierCurveTo(20,100,**200,100**,200,20)

HTML5 canvas, utilities

Make a snapshot:

```
function(){  window.open(canv.toDataURL('image/png')); }
```

Convert degrees to radians:

```
radians = degrees*(Math.PI/180)
```

Convert radians to degrees:

```
degrees = radians / (Math.PI/180)
```

x

HTML5 canvas, transformations - 1

Rotation:

```
// translate context to the center of the shape
ctx.translate(cx, cy);
// rotate 45 degrees clockwise
ctx.rotate(45*Math.PI /180);
//rectangle sizes
var w = 150;
var h = 80;
// begin rotated shape
ctx.beginPath();
ctx.moveTo(0-w/2, 0-h/2);
ctx.lineTo(0+w/2, 0-h/2);
ctx.lineTo(0+w/2, 0+h/2);
ctx.lineTo(0-w/2, 0+h/2);

// complete rotated shape
ctx.closePath();
```

```
/**
* to rotate a shape around its
* centre, you must:
* a) translate the origin of the
*    context
* b) rotate the context around new
*    origin (in radians)
* c) draw the shape with its centre
*    on the new origin
*/
```

HTML5 canvas, transformations - 2

Scale:

```
// draw a rectangle
ctx.strokeRect(0,0,50,20);

// scale the context (3 times bigger)
ctx.scale(3,3);

// draw the same shape again
ctx.strokeRect(0,0,50,20);
```

```
/**
*        ** CAREFUL **
* just like rotate, scale (and other
* transformations) will
* affect the whole context,
* not the shape only.
*/
```

HTML5 canvas, gradients - 1

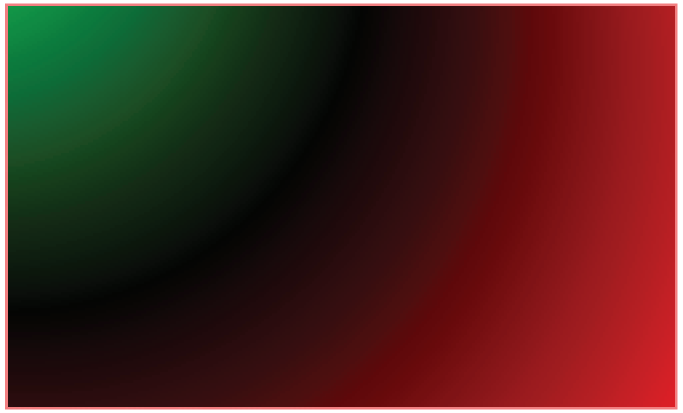
Linear gradients:

```
// create a linear gradient (top left x, y - bottom right x,y)
var gr=ctx.createLinearGradient(200,0,450,0);        //horizontal
//var gr=ctx.createLinearGradient(0,100,0,250)       //vertical
//var gr=ctx.createLinearGradient(200,100,450,250);  //diagonal

// create colorStops: addColorStop(pos:[0->1], col:rbg)
gr.addColorStop(0,'rgb(255,0,0)');
gr.addColorStop(1,'rgb(0,0,255)');

//set the gradient to fill the shape
ctx.fillStyle = gr;

//draw the shape
ctx.fillRect(200, 100, 250, 150);
```

HTML5 canvas, gradients - 2

Radial gradients:

```
//createRadialGradient(cp1x, cp1y, cp2x, cp2y, c1radius, c2radius)
var gr = ctx.createRadialGradient(250,150,250,150,10,20);

gr.addColorStop(0, 'rgb(255, 0, 0)');
gr.addColorStop(0.5, 'rgb(0, 0, 0)');
gr.addColorStop(1, 'rgb(0, 255, 0)');

ctx.beginPath();
ctx.fillStyle = gr;
ctx.fillRect(200, 100, 250, 150);
ctx.closePath();
```



HTML5 canvas, images

Add images:

```
//getting canvas and context
var canvas = document.querySelector("#myCanvas");
var ctx = canvas.getContext("2d");

//creating new image
var img = new Image();

//setting image source
img.src = 'assets/imgs/pic.png';

//ctx.drawImage(img, x, y, w, h);
ctx.drawImage(img, 200, 200, 450, 450);
```