**Part 1:**

HTML5

**CANVAS**

To view all properties & methods associated with HTML5 canvas – type HTMLCanvasElement.prototype in to the console.

Rectangle methods and the 2d context object

Var ctx = document.getElementById(‘canvas’).getContext(‘2d’);

*Ctx refers to the CanvasRenderingContext2d object. Write this in the console to view members (properties & methods).*

Ctx.fillRect(x, y, w, h); //create rectangle using x & y co-ordinates, and width & height of rect.

Ctx.strokeRect(x,y,w,h); // non filled rectangle, same parameters as fillRect().

Ctx.fillStyle = “color”; // rgba || hex || color || gradient || pattern

Ctx.clearRect(x,y,w,h); //clear rectangles in the parameters given

\**use ctx.canvas.width (and/or height) to get full co-ordinates of canvas. You do not need to use static dimensions*\*

Fill & stroke styles color gradient pattern

Ctx.fillyStyle = ‘color’; ctx.strokeStyle = ‘color’;

Ctx.fillStyle = ‘rgba(0,200,0,1)’;

Ctx.strokeStyle = ‘red’;

Ctx.fillRect(100, 100, 150, 150);

Ctx.strokeRect(100, 100, 150, 150);

//this will create a green filled rectangle with a red border/stroke

Ctx.lineWidth = 10; //sets thickness of stroke

Var gradient = ctx.createLinearGradient(x0, y0, x1, y1);

Gradient.addColorStop(0, ‘magenta’);

Gradient.addColorStop(2, ‘black’);

Ctx.fillStyle = gradient;

Var gradientR = ctx.createRadialGradient(x0, y0, r0, x1, y1, r1); //create radial gradient

Var leather = new image(); leather.src = ‘leather.jpg’; //for creating patterns

Var pattern = ctx.createPattern(leather, ‘repeat’); // repeatx || repeaty || … (similar to CSS)

Ctx.fill();

Line styles for lines & strokes

Ctx.beginPath(); //begin creating line

Ctx.moveTo(150,150); // line start position

Ctx.lineTo(240,240); //where to move line from start position

Ctx.lineCap = “round”; //butt || round || square

Ctx.lineJoin = “miter”; //miter || round || bevel

Ctx.miterLimit = 5;

Ctx.stroke(); //draw line

Ctx.setLineDash([20,10]); //give dash effect

Ctx.getLineDash(); //gets copy of lineDash pattern

Building & Drawing paths with Javascript

Ctx.moveTo(x,y); //creates new subpath at the given points

Ctx.closePoint(); //close current path

Ctx.isPointInPath(x,y); //determines if specified point is in a path (console log this)

Ctx.clip(); //clips outline if not within shape

Ctx.arc(x,y,r,sAngle,eAngle, [true/false]); //s=start, e=end. True/false = clockwise/counter

Ctx.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y);

Ctx.quadraticCurveTo(cpx, cpy, x, y);

//cp1 = first control point, cp2 = second control point, x/y = end point

Ctx.arcTo(x,y);

Drawing images & videos with Javascript

Var my\_pic = new Image();

My\_pic.src = “location.jpg”;

My\_pic.onload = function() {  
 var ctx = document.getElementById(‘canvas’).getContext(‘2d’);

Ctx.drawImage(my\_pic, x, y, w, h); //there are option parameters if clipping

}

Drawing text

Ctx.fillStyle = ‘color’;

Ctx.font = “font properties”; ///italic bold 16px Arial, sans-serif etc…

Ctx.fillText(‘text to input’, x, y); //input text fill

Ctx.textAlign = “Start”; //start || end || left || right || center

Ctx.textBaseline = “hanging”; //top || middle || bottom || hanging || alphabetic || ideographic

Ctx.lineWidth = 3; //line thickness of text

Ctx.strokeText(“input text”, x, y); //input text stroke

Shadows

Ctx.fillStyle = “color”;

shadowColor = color || hex || rgba();

shadowOffsetX = //positive or negative number

shadowOffsetY = //positive or negative number

shadowBlur = //positive number

ctx.fillRect(x,y,w,h);

Transformation effects

Ctx.rotate(angle); //rotates current drawing by specified degrees

Ctx.transform(xScale, ySkew, xSkew, yScale, xTrans, yTrans); //scale, rotate, move & show the context

//default transform matrix is (1, 0, 0, 1, 0, 0);

Ctx.setTransform(1, 0, 0, 1, 0, 0); //resets the current transformation to the identity matrix & then invokes a transformation described by the arguments of this method

Pixel manipulation

//draw 2 rectangles first

Ctx.fillStyle = “magenta”;

Ctx.fillRect(20,20,100,100);

Ctx.fillStyle = “orange”;

Ctx.fillRect(40,40,100,100);

Var src = ctx.getImageData(x,y,w,h); //returns image data in pixels

Ctx.putImageData(src, x, y); //paints data from the getImageData() on to the canvas

OR

Var copy = ctx.createImageData(src.width, src.height);

For(var i = 0; i<copy.data.length; i++) { //where length = no. of pixels

Copy.data[i] = src.data[i];

}

Ctx.putImageData(copy,250,250);

Compositing

Ctx.globalAlpha = .5; //global transparency – values between 0 and 1

Ctx.globalCompositeOperation = source-atop || source-in || source-out || source-over(default) || destination-atop || destination-in || destination-out || destination-over || lighter || copy ||xor

//practice these by putting these lines of code between two different colour rectangles

Dynamic centering & alignment

Var rectW = 100;

Var rectH = 100;

Var rectX = (ctx.canvas.width / 2) – (rectW / 2);

Var rectY = (ctx.canvas.height / 2) – (rectH / 2);

Ctx.fillRect(rectX, rectY, rectW, rectH);