HTML5 Notes

HTML5 CANVAS:

* <canvas> element:
* Used to draw graphics, on the fly, via JavaScript
* The element is only a container for graphics
* Must use JS to actually draw the graphics
* Canvas has several methods for drawing paths, boxes, circles, text, & adding images
* A canvas is a rectangular area on an HTML page
* Has no border and no content by default
* Must have an id attribute so it can be referred to by JS
* Must have a width & height attribute to define the size of the canvas
* Markup: <canvas id="myCanvas" width="200" height="100"></canvas>

Drawing:

* All drawing on the HTML canvas must done with JavaScript (or jQuery)
* Steps:

1. Find the Canvas Element:

var canvas = document.getElementById("myCanvas");

$(“.myCanvas).

1. Create a drawing Object:

* getContext() 🡪 built-in HTML object w/ properties and methods for drawing

var ctx = canvas.getContext("2d");

1. Draw on the Canvas:

* **fillRect(x, y, width, height)** method:
* Sets the fill style property of the drawing object
* can be a CSS color, gradient, or pattern; default is black

Coordinates:

* The HTML canvas is a two-dimensional grid.
* The upper-left corner of the canvas has the coordinates (0,0)
* Therefore, **fillRect(0,0,150,75)** means:

🡪 Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle

* To Draw a Line:
* **moveTo(x, y)** - defines the starting point of the line
* **lineTo(x, y)** - defines the ending point of the line
* To actually draw the line 🡪 must use one of the "ink" methods, like **stroke()**
* **ex:**

var canvas = document.getElementById("myCanvas");  
var context = canvas.getContext("2d");  
context.moveTo(0,0);  
context.lineTo(200,100);  
context.stroke();

* Draw a Circle:
* **beginPath()** - begins a path
* define circle w/ the **arc()** method - creates an arc/curve
* Set start angle to 0 and end angle to 2\*Math.PI. The x and y parameters define the x- and y-coordinates of the center of the circle. The r parameter defines the radius of the circle
* use the **stroke()** method to actually draw the circle
* **ex:**

var canvas = document.getElementById("myCanvas");  
var context = canvas.getContext("2d");  
context.beginPath();  
context.arc(95,50,40,0,2\*Math.PI);  
context.stroke();

Gradients:

 \*Two different types of gradients:

* **createLinearGradient(*x,y,x1,y1*)** - creates a linear gradient
* **createRadialGradient(*x,y,r,x1,y1,r1*)** - creates a radial/circular gradient

\*Once you’ve created a gradient object 🡪 add 2+ color stops:

* **addColorStop()** method - specifies the color stops & its position along the gradient
* gradient positions can be anywhere between 0 to 1
* to use the gradient:
* set the fillStyle or strokeStyle property to the gradient
* then draw the shape (rectangle, text, or a line)
* **ex - Linear:**

var c = document.getElementById("myCanvas");   
var ctx = c.getContext("2d");  
  
var grd = context.createLinearGradient(0,0,200,0); //Create gradient  
grd.addColorStop(0,"red");  
grd.addColorStop(1,"white");  
  
context.fillStyle = grd; //Fill w/ gradient  
context.fillRect(10,10,150,80);

* **ex – Radial/Circular:**

var c = document.getElementById("myCanvas");   
var ctx = c.getContext("2d");  
  
var grd = context.createRadialGradient(75,50,5,90,60,100); //Create gradient  
grd.addColorStop(0,"red");  
grd.addColorStop(1,"white");  
  
context.fillStyle = grd; //Fill w/ gradient  
context.fillRect(10,10,150,80);

Drawing Text:

\*Most important property & methods are:

* font - defines the font properties for the text
* **fillText(text, x, y)** - draws "filled" text on the canvas
* **strokeText(text, x, y)** - draws text on the canvas (no fill)
* **ex. – Sets font to 30px Arial & writes a filled in text on the canvas:**

var canvas = document.getElementById("myCanvas");  
var ctx = canvas.getContext("2d");  
context.font = "30px Arial";  
context.fillText("Hello World",10,50);

* **ex. – Sets font to 30px Arial & writes text, w/ no fill, on the canvas:**

var canvas = document.getElementById("myCanvas");  
var ctx = canvas.getContext("2d");  
context.font = "30px Arial";  
context.strokeText("Hello World",10,50);

Article Notes <http://www.informit.com/articles/article.aspx?p=2424802>

Canvas generally corresponds to the functionality of Flash in earlier browsers. Canvas is intended to be standard across all browsers, as opposed to Flash, which is proprietary

**// Global scope and variables**

var Main = {};

Main.Canvas = document.getElementById('myCanvas');

Main.Context = Main.Canvas.getContext('2d');

**// Mouse coordinates**

Main.MX = 0;

Main.MY = 0;

**// Keep track of the mouse**

Main.Canvas.onmousemove = function(event)

{

if (event.offsetX) {

mouseX = event.offsetX;

mouseY = event.offsetY;

}

else if (event.layerX) {

mouseX = event.layerX;

mouseY = event.layerY;

}

Main.MX = mouseX;

Main.MY = mouseY;

}

**// Do the animation**

Main.Animate = function()

{

Main.Context.clearRect(0, 0, Main.Canvas.width, Main.Canvas.height);

**// Draw the rectangle**

Main.Context.fillStyle = "#FF0000";

Main.Context.fillRect(0, 0, 150, 75);

**// Draw the circle**

Main.Context.beginPath();

Main.Context.arc(395, 150, 60, 0, 2 \* Math.PI);

Main.Context.stroke();

**// Draw the current position of the mouse**

Main.Context.font = "30px Arial";

Main.Context.fillText("X: " + Main.MX + " Y: " + Main.MY, 100, 150);

requestAnimFrame(function() { Main.Animate(); });

}

**// Browser compatibility**

window.requestAnimFrame = (function(callback)

{

return window.requestAnimationFrame

|| window.webkitRequestAnimationFrame

|| window.mozRequestAnimationFrame

|| window.oRequestAnimationFrame

|| window.msRequestAnimationFrame

|| function(callback) { window.setTimeout(callback, 1000 / 60); };

})();

**// When page is loaded**

$(document).ready(function()

{

Main.Animate();

});

* var Main = {};
* creates a sort-of-global scope called Main, which allows us to reference components at the application (global) level
* Main.Canvas = document.getElementById(‘myCanvas’);
* Assigns the canvas element from the HTML to Main.Canvas
* Could be done w/ JQuery

\*Canvas is a predefined object that comes with methods defined by the specification, such as getContext(). If you use the getContext() method with the parameter '2d', you can acquire a context object:

* Main.Context = Main.Canvas.getContext(‘2d’);
* Main.MX = 0; & Main.MY = 0;
* it's good practice to create globally scoped MX and MY variables to keep track of the mouse's location
* Main.Context.clearRect(0, 0, Main.Canvas.width, Main.Canvas.height);
* Clears the entire context (screen) using the globally defined coordinates

**The Framework**

The following code is basic, but consider it a framework to be included in all animations:

Main.Animate = function()

{

**// Animation Code Goes Here**

requestAnimFrame(function() { Main.Animate(); });

}

window.requestAnimFrame = (function(callback)

{

return window.requestAnimationFrame

|| window.webkitRequestAnimationFrame

|| window.mozRequestAnimationFrame

|| window.oRequestAnimationFrame

|| window.msRequestAnimationFrame

|| function(callback) { window.setTimeout(callback, 1000 / 60); };

})();

$(document).ready(function()

{

Main.Animate();

});