

## Introduction to HTML5 Canvases

There are many online tutorials for HTML5, as you have probably already seen. You can select the one you want for learning about HTML5 canvases. I like this one:

- [http://www.w3schools.com/html5/html5\\_canvas.asp](http://www.w3schools.com/html5/html5_canvas.asp)

The code below illustrates how to create a “canvas” in HTML5:

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
window.onload=function() {
  <!-- get graphics context -->
  var c = document.getElementById( "myCanvas" );
  var cntxt = c.getContext( "2d" );
  <!-- draw line -->
  cntxt.strokeStyle = "#999999";
  cntxt.moveTo( 100, 100 );
  cntxt.lineTo( 400, 400 );
  cntxt.stroke();
}
</script>
</head>
<body>
<canvas id="myCanvas" width="500" height="500" style="border:5px solid #000000;">
Your browser does not support the canvas element.
</canvas>
</body>
</html>
```

The <canvas> tag is invoked in the HTML body. It has two attributes: width and height, which specify the width and the height of the canvas, in pixels. The style attribute specifies that a 5-pixel black border should be drawn around the border of the canvas. An HTML5 canvas creates a container for drawing graphics inside it.

In the HTML head, JavaScript code is defined which draws inside the canvas. The above example invokes the JavaScript function window.onload(), so that the script is executed when the HTML page loads.

The first two statements in the window.onload() function grab the graphics context and store it in a variable called cntxt:

```
var c = document.getElementById( "myCanvas" );
var cntxt = c.getContext( "2d" );
```

The remaining lines in the example demonstrate drawing a line inside that graphics context:

```
cntxt.strokeStyle = "#999999";
cntxt.moveTo( 100, 100 );
cntxt.lineTo( 400, 400 );
cntxt.stroke();
```

Note that these functions are invoked as part of the graphics context object, cntxt, which was initialized above.

The first line sets the color of the line to be drawn as a medium grey color (#999999).

The second line, which invokes the `moveTo()` function, essentially places the drawing pen at the specified (x,y) location in the canvas so that any subsequent drawing commands will take effect from that point.

The third line, which invokes the `lineTo()` function, draws a line from the current location of the drawing pen to the location specified by the (x,y) arguments to the function.

Taken together, the `moveTo( 100, 100 )` and `lineTo( 400, 400 )` functions draw a line diagonally across the canvas from position (100,100) to position (400,400).

Note that the coordinate system of the graphics context follows computer science conventions, i.e., the upper left corner of the graphics context is (0,0), *x* increases going to the left and *y* increases going downward.

## Lab Exercises

*Note that this lab is an exercise and will not be submitted.*

Example code is posted on the class syllabus page, under today's date.

1. Begin by typing in the example, above, to create an HTML canvas and draw a line inside it. Save your code in a file named something like `ex1.html` (i.e., make sure that your filename ends in `.html`). Test your code and make sure it works by loading it in your browser.
2. Modify the example by drawing additional lines using different colors.
3. Now create a new HTML file and copy into it the contents of your first example, above. Replace the code for drawing lines with code for writing text, like this:

```
cntxt.fillStyle = "#0000ff";
cntxt.font = "20px Arial";
cntxt.fillText( "abracadabra", 20, 20 );
```

As above, test your code and make sure it works by loading it in your browser.

4. Now create a new HTML file and copy into it the contents of your first example, above. Replace the code for drawing lines with code for drawing shapes, such as rectangles, circles and arcs.

Here is sample code for drawing a rectangle, with upper-left corner at (100,100), extending for a width of 100 and a height of 100 pixels:

```
cntxt.strokeStyle = "#00ff00";
cntxt.strokeRect( 100, 100, 100, 100 );
```

Here is sample code for drawing a circle, with center at (100,400) and radius of 100 pixels:

```
cntxt.beginPath();
cntxt.arc( 100, 400, 100, 0, Math.PI*2, true );
cntxt.closePath();
```

Refer to the "Reference" section, below, for information on other drawing functions you could try.

As always, test your code before moving on to the next step.

5. Now create a new HTML file and copy into it the contents of your first example, above. Replace the code for drawing lines with code for drawing an image.

Here is sample code for drawing an image that is stored in the file named "myimage.jpg". The image is displayed with its upper-left corner at (10,10) in the canvas, scaled to a size of 100 × 100 pixels.

```

var img = new Image();
img.src = "myimage.jpg";
cntxt.drawImage( img, 10, 10, 100, 100 );

```

Note that there are additional functions for displaying and manipulating images. See the “Reference” section below, as well as the URLs listed for more information.

6. The final example shows how to use an HTML5 canvas to do animation and to connect HTML form elements (e.g., buttons) to the behavior of the canvas. Look at the example code posted on the class web page (syllabus) for today’s date.

The key elements are as follows:

- JavaScript function to animate a circle:

```

function animate(){
    // get graphics context
    var c = document.getElementById( "myCanvas" );
    var cntxt = c.getContext( "2d" );
    cntxt.clearRect( 0, 0, c.width, c.height );
    // draw circle
    cntxt.fillStyle = "#ff0000";
    cntxt.beginPath();
    cntxt.arc( x, y, r, 0, Math.PI*2, true );
    cntxt.closePath();
    cntxt.fill();
    // move circle
    x++;
    if ( x > c.width ) { x = 0; }
    y++;
    if ( y > c.height ) { y = 0; }
    // request new frame
    if ( more == 1 ) {
        requestFrame( function() { animate(); } );
    }
}

```

- JavaScript “callback” function for animation, which essentially causes the animation() function to be called at regular intervals:

```

window.requestFrame = ( function( callback ) {
    return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        window.oRequestAnimationFrame ||
        window.msRequestAnimationFrame ||
        function( callback ) { window.setTimeout( callback, 1000 / 60 ); };
} ) ( ) ;

```

- JavaScript function that is called when the page loads, which initiates the animation:

```

window.onload = function() {
    animate();
};

```

Try entering the animation code and running it.

Then try modifying the code in some way. For example, add buttons to change the color of the animated circle. Or animate a rectangle instead of a circle.