**Practical-05 - Canvas and JavaScript**

Canvas is a new tag added in HTML5 and is used to draw images, shapes, and is often used for HTML5 games. All the properties of the canvas (size, colour, position, etc) can be hardcoded into the HTML. However, the more common approach is to use JavaScript to modify the canvas at runtime – generally based on some form of user input. Follow the steps below and show your prac supervisor once you’re done.

1.) Create a blank html page with an empty and a <title> of “My First Canvas”

2.) Inside the create a new canvas by using the tag.

3.) Set the “id” attribute of the to “mycvs” and the “height” and “width” attributes to “200”

4.) Within the or (it doesn’t matter which one) create a new <script> of type “text/javascript”

5.) We will need this script to run only once the page is loaded so enter the following at the top of the <script> a. function drawOnLoad(){} b. inside the tag add the attribute onload=”drawOnLoad()”

6.) To begin working on the canvas we first need to get it from the DOM and assign it to a new variable, enter the following code within the drawOnLoad() var canvas = document.getElementById(“mycvs”);

7.) Now that we have the document element we need to get the graphical context which we will use to draw images, text, etc. We do this by using the getContext function on the canvas variable as follows a. Note we’re using 2d context, 3d can be achieved using var ctx = canvas.getContext(“2d”);

8.) Now that we have the context we can start drawing on the canvas! a. See <https://www.w3schools.com/tags/ref_canvas.asp> for a full list of available functions

9.) First we will draw ourselves a blue rectangle. To do this we first set the fillStyle to the colour blue ctx.fillStyle = ‘rgb(0,0,255)’; //This is just an rgb color value

10.) Now that we have set the fillStyle we need to draw the rectangle. This is as simple as using the fillRect function. a. ctx.fillRect(x,y, height, width); // where x and y are the top left co-ordinates of the rect in regards to the top left of the canvas

11.) Draw a rectangle at the coordinates at 20,20 with a height of 100 and width of 50

12.) Open the html by using the lightning symbol and see if this works.

13.) We can draw many other shapes using the Path aspect of canvas. We will be drawing a circle!

14.) First, let’s set the color to a semi-transparent red a. ctx.fillStyle = ‘rgba(255, 0, 0, 0.5) //The ‘a’ stands for alpha and refers to the level on transparency 0 = invisible 1 = solid

15.) Next we need to begin the path a. ctx.beginPath()

16.) Then we use the arc() function to create the path a. ctx.arc(x, y, radius, startAngle, endAngle) // x & y are the centre point, radius is the size, the angles are in Radians not degrees, 0 radians is the 3 oclock position of the circle, 1 radians is the 9 oclock position b. Create a path at x,y of 120, 50, with a radius of 10, a start angle of 0 and an end angle of 2\*Math.PI c. <https://www.w3schools.com/tags/canvas_arc.asp> see this for further understanding

17.) This will create a path but nothing will be drawn yet. To now draw the path we need to call fill() or stroke(). Fill draws a solid circle while stroke draws the outline. a. ctx.fill();

18.) Test your file and you should see a semi-transparent red circle over your rectangle.

19.) Our canvas is getting dirty now and we want to erase all our drawings. We do this by using the clearRect(x,y,width, height) function. This function works in the exact same way as fillRect() except it sets everything inside the rect to be empty/transparent. We can easily clear the whole canvas by: a. ctx.clearRect(0, 0, canvas.width, canvas.height);

20.) Now we know how to draw rects and circles and many other shapes on the canvas. But they don’t do anything yet, lets make a rectangle that moves horizontally across the screen.

21.) First thing is we will want to create a new function called drawRect()

22.) Within the same <script> tag but below the drawOnLoad() function create this new function and copy and paste our rectangle drawing code into it. Modify the fillRect to start at 0,100 and have a size of 10 in both directions Function drawRect(){ ctx.fillStyle = ‘rgb(0,0,255)’; ctx.fiilRect(0, 100, 10, 10); }

23.) Next we need to create an Update() function that will be called at a set time Interval (we’ll get to this). The update function should clear the canvas completely everytime and then call the drawRect(). a. Function update(){ ctx.clearRect(0,0, canvas.width, canvas.height); drawRect(); b. }

24.) Everytime this function gets called it will clear the canvas and redraw the rectangle. At the moment the rectangle doesn’t move so we need to do that next.

25.) We do can do this by creating two new variables. var x =0; and var xSpeed = 2; Place these variables just above the drawRect() function definition.

26.) Then inside the update function after the call to drawRect() we will need to update the x variable by the speed a. x += xSpeed;

27.) Now we have an updating x position but the rectangle doesn’t use this currently. We need to modify the drawRect() function so that is uses the x variable as the drawing location for the rectangle. a. ctx.fillRect(x, 100, 10, 10);

28.) We’re almost there! The final step is to call the update() function at whatever time interval we want. Javascript has a handy built in function for this called setInterval(functionToCall, timeToCall).

29.) Below the update() function place the following line: a. setInterval(update, 1000/60) //The time there is for 60FPS.

30.) Save and test that your code works! If it does then show your prac supervisor to complete your prac.

31.) BONUS CHALLENGE: Get the rectangle to bounce back the other way when it hits the right hand edge of the canvas.