Lab 8

Problem to solve

I want you to create a Breakout-game (see figure 1) using the canvas element.
In this lab you will create the first part; a ball bouncing off the left-, top- and right walls and a paddle that you can control using your mouse cursor.
Next week you will add bricks to destroy and a score.

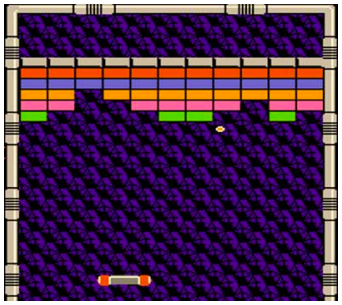

Figure 1 - Arkanoid, 1986

1. download and unzip *Lab8-firstname_lastname.zip*
2. rename the folder to match your name
3. define the ball's properties using an object:
   x, y, dx*, dy, w, h and speed
4. add two methods to the ball object:
   move, which changes the ball's x and y values `x = x + (speed * dx)`
   draw, which draws a filled rectangle on canvas using the objects properties
5. Create a global function called **update** in which you call the ball's **move** and **draw** functions. What happens if you don't clear the canvas first?
   As the last statement in the function call requestAnimationFrame(update) so that the function will be called again in 1/60 of a second.
6. Create a function that checks if the ball is hitting a wall (if its spatial properties are no longer within the canvas' dimensions) and changes its dx or dy.
   Add a function call for it in the update function.
7. Create a paddle object, similar to the ball object.
   It only needs the x, y, w and h properties plus a draw method.
8. Position it at the bottom of the canvas.
9. Make sure the paddle is drawn on the canvas as well.

* x-direction is either 1 or -1 and will control the direction

Star level

Make the paddle follow the x-position of the mouse cursor when the mouse is over the canvas.
Make the ball bounce off the paddle back in to the game area.
Hints:
There is an event called *mousemove.*
The mouse position is available as properties in event objects and they're called *clientX* and *clientY*. They report the mouse position relative to the browser window (not the canvas) at the time of the event.
offsetLeft is a property on HTML elements reporting the distance between the element and the browser window's left edge.

To get the extra point, your solution must work at first upload and be handed in no later than 4 days after your lab.
Add a comment to the upload "aiming for a star"

Upload your solution

Zip your folder and name your file *Lab8-firstname_lastname.zip*
Upload it on pingpong.

Canvas 2D context properties and methods

++
Setting color and style
++

`fillStyle = color` - change fill color
`strokeStyle = color` - change stroke color
`lineWidth = number` - width in pixels
`lineCap = "butt" / "round" / "square"`
`lineJoin = "miter" / "bevel" / "round" –corners`
`miterLimit = integer` - higher numbers allow sharper corners
`save()` saves the current stroke and color state
`restore()` resets any changes made to stroke and color, to last saved state

++
Drawing rectangles
++

`fillRect(startX, startY, width, height)` -a filled rectangle
`strokeRect(startX, startY, width, height)` -an outlined rectangle
`clearRect(x, y, w, h)` -erases everything within

++
Animating canvas
++

Canvas Animation is all about drawing, erasing, drawing, erasing....

Erase a rectangle on canvas:

```
ctx.clearRect(x, y, w, h)
```

If you have a reference to the canvas element you can find its width and height like this:

```
ctx.clearRect(0, 0, canvas.width, canvas.height);
```

Animating:
There are two new methods in HTML5 for animating:
-requestAnimationFrame
-cancelAnimationFrame
Usage:

```
ID = requestAnimationFrame(functionName);
cancelAnimationFrame(ID)
```

For an example see this pen: http://codepen.io/jkohlin/pen/ZeMRBx?editors=0010