Tutorial 7

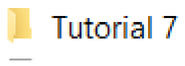
Editor – Brackets

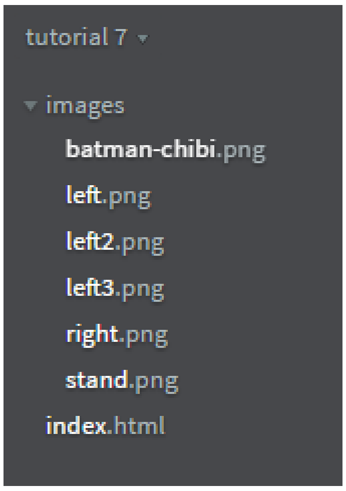
Goals

Create a website showcasing the following techniques

- Animated backgrounds
- Animated game elements
- 3 pages with differing game elements

Website

- Create a folder on the desktop called tutorial 7
    - 
      Tutorial 7
- Open Brackets
- Create the following structure and index.html file

  ```
  tutorial 7 ▾

  ▾ images
      batman-chibi.png
      left.png
      left2.png
      left3.png
      right.png
      stand.png
    index.html
  ```
-

Download the resources: http://www.amgelshadowx.com/adv/Tutorial%207.zip

Extract the images and place in an images folder.

Notice that the script is now below the body, this is so the elements required for the scripting are loaded into memory just before the scripts are loaded.

Timer Page

Let’s start with the canvas and the styles required for the page.

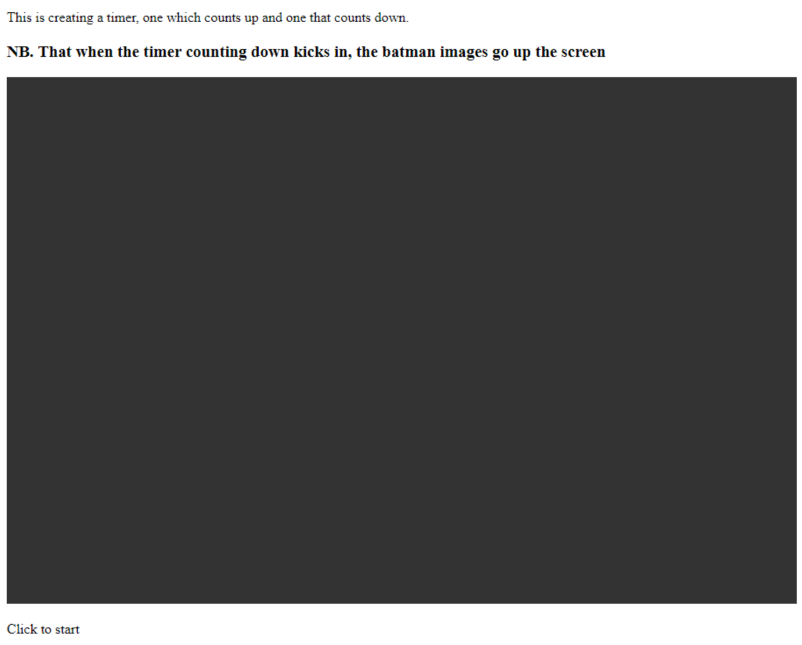
Body Code

```
<body>
    <p>This is creating a timer, one which counts up and one that counts down.</p>
    <h3>NB. That when the timer counting down kicks in, the batman images go up the screen</h3>
    <canvas id="myCanvas" width="900" height="600" class="silver"></canvas>
    <p onclick="start()">Click to start</p>
</body>
```

Style

```
<style type="text/css">
    .silver {background-color:#333333;}
    p {cursor: pointer;}
</style>
```

Result



This is creating a timer, one which counts up and one that counts down.

NB. That when the timer counting down kicks in, the batman images go up the screen

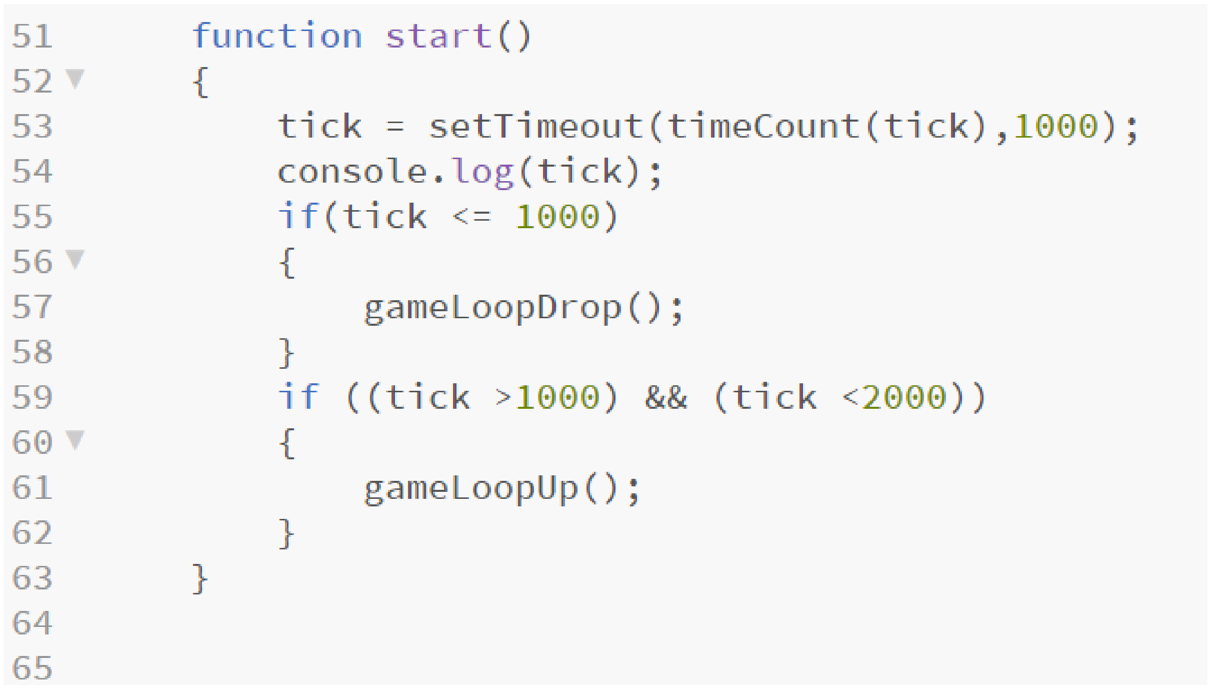
Click to start

From here, we start adding in variables to hold the batman image and an array for calculating random starting positions.

```
<script>
    var canObj = document.getElementById("myCanvas");
    var ctx = canObj.getContext('2d');
    var gameWidth = canObj.width;
    var gameHeight = canObj.height;
    var cUp = 1;
    var cDown = 90;
    var tick = 0;

    //Batman Chibi
    var batman = new Image();
    batman.src = "images/batman-chibi.png";

    // array of differing positions
    var maxArray = 4; // Arrays start at 0, so 4 is actually 5 images.
    positionX = new Array();
    positionY = new Array();
    for (x=0;x<=maxArray;x++)
        positionX[x] = Math.random()*800;
    for (x=0;x<=maxArray;x++)
        positionY[x] = Math.random()*100;

    //console.log("ctx: ",ctx);
    //console.log("canObj: ",canObj);
    var fps = 60;
    window.requestAnimFrame = (function(){
      return  window.requestAnimationFrame       ||
              window.webkitRequestAnimationFrame ||
              window.mozRequestAnimationFrame    ||
              window.msRequestAnimationFrame    ||
              window.oRequestAnimationFrame     ||
              function( callback ){
                window.setTimeout(callback, 1000 / fps);
              };
    })();
</script>
```

Saving and testing this should result in no errors being present on the screen. Next we need to start adding the functions that will be used to move the chibi batmans up and down the screen. The first function we call is start().

Function start()

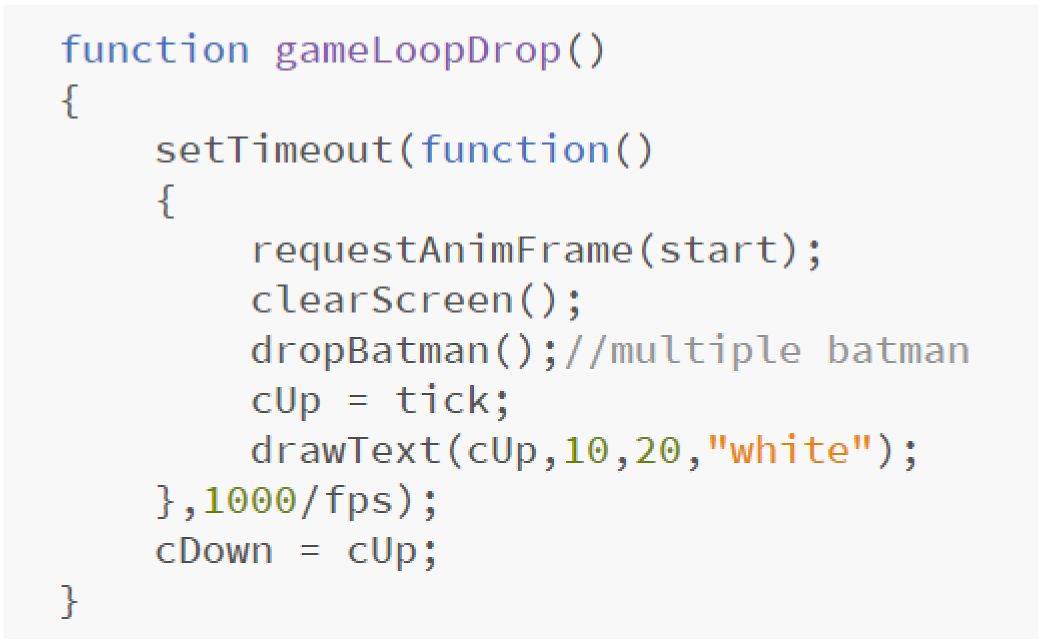


```
    function start()
    {
        tick = setTimeout(timeCount(tick),1000);
        console.log(tick);
        if(tick <= 1000)
        {
            gameLoopDrop();
        }
        if ((tick >1000) && (tick <2000))
        {
            gameLoopUp();
        }
    }


```

This function is designed to count and, depending on what value the variable tick has, it will call the drop loop or the up loop. Each of those particular loops will move the batman image up or down.

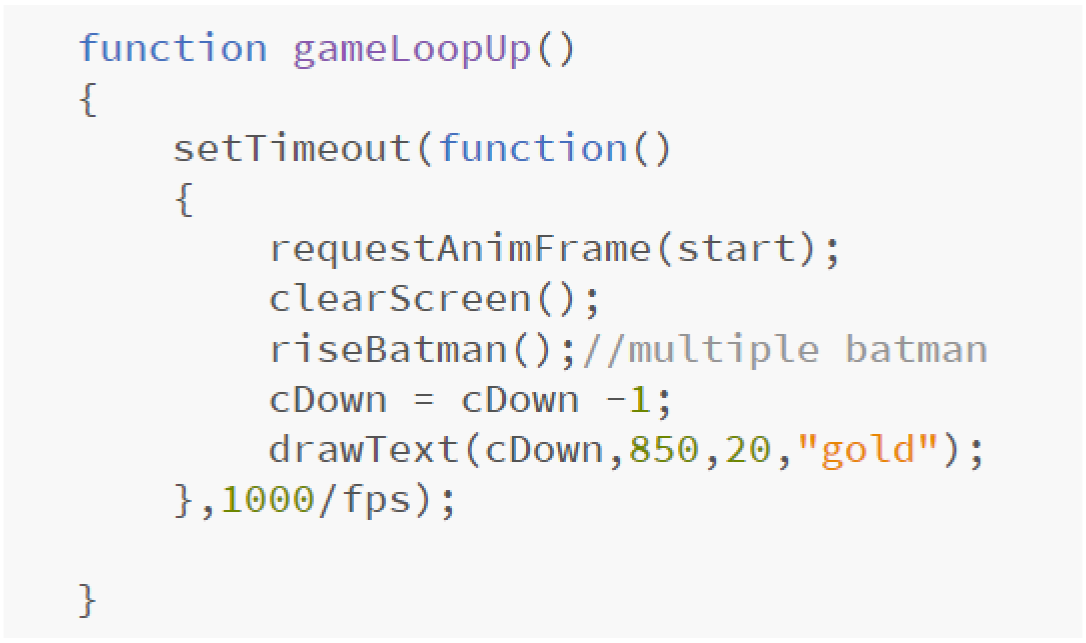
Let’s write these up

Function gameLoopDrop()



```
function gameLoopDrop()
{
    setTimeout(function()
    {
        requestAnimFrame(start);
        clearScreen();
        dropBatman();//multiple batman
        cUp = tick;
        drawText(cUp,10,20,"white");
    },1000/fps);
    cDown = cUp;
}
```

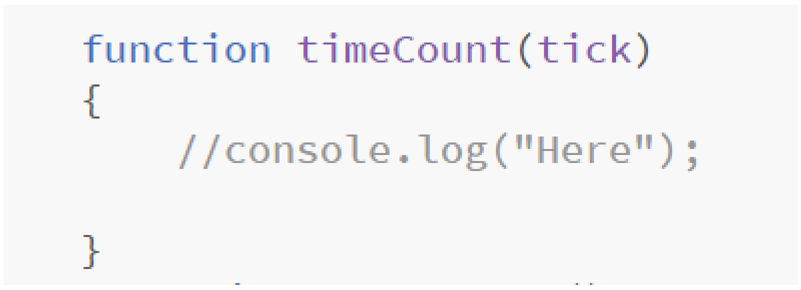
Function gameLoopUp()



```
function gameLoopUp()
{
    setTimeout(function()
    {
        requestAnimFrame(start);
        clearScreen();
        riseBatman();//multiple batman
        cDown = cDown -1;
        drawText(cDown,850,20,"gold");
    },1000/fps);

}
```

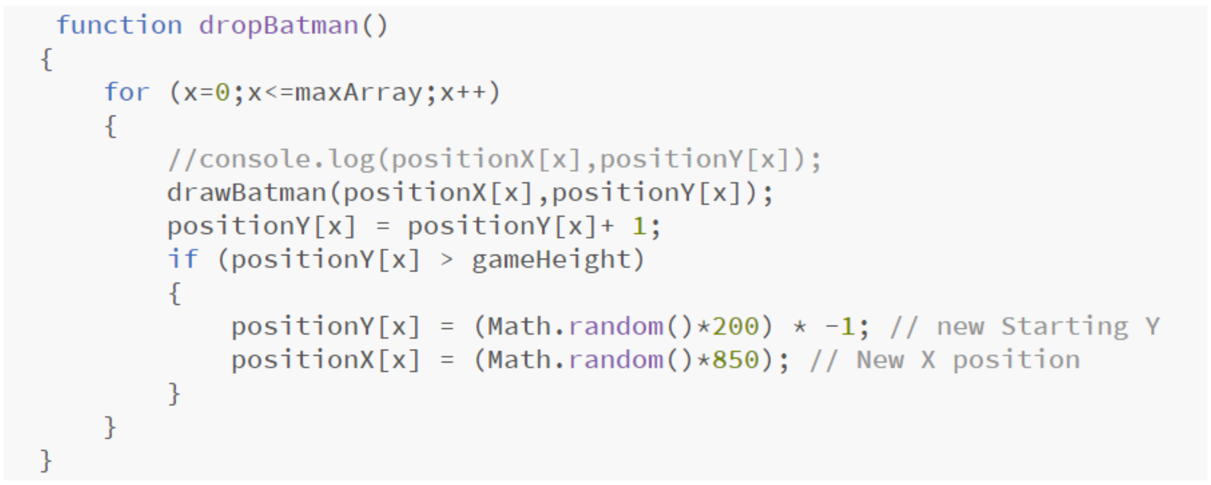
As you can see each loop is designed to clear the screen and position the batman image. Now we need to add the remaining functions and test it. These functions are:

Function timeCount()



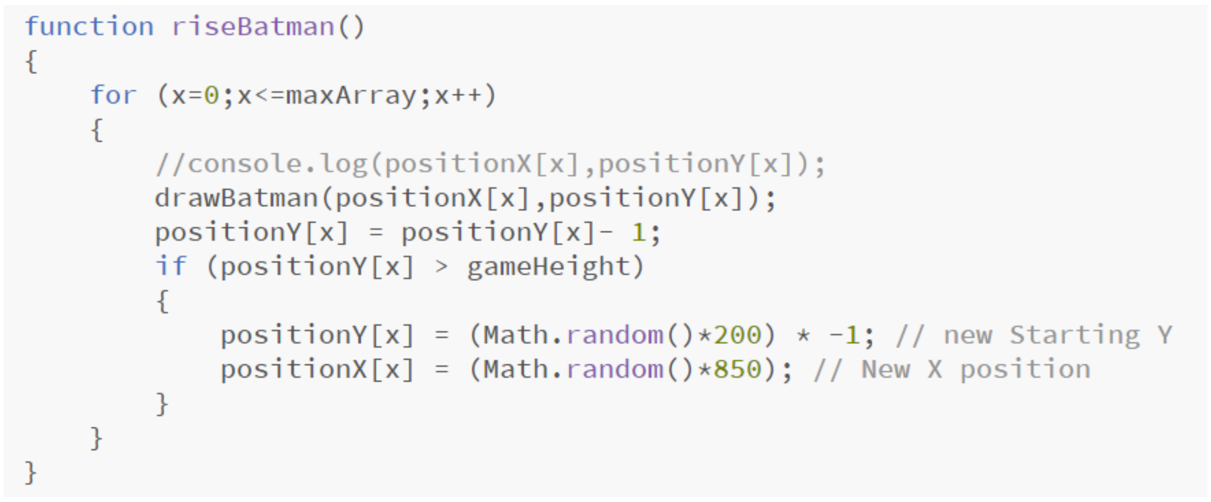
```
function timeCount(tick)
{
    //console.log("Here");

}
```

Function dropBatman()



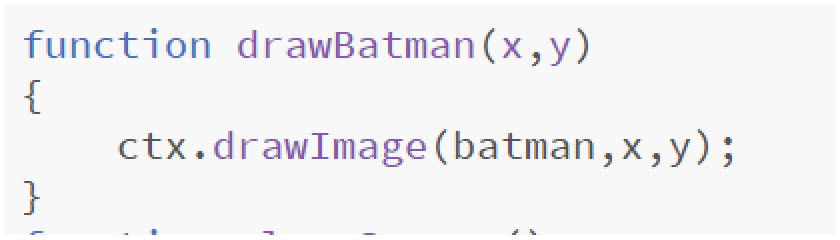
```
 function dropBatman()
{
    for (x=0;x<=maxArray;x++)
    {
        //console.log(positionX[x],positionY[x]);
        drawBatman(positionX[x],positionY[x]);
        positionY[x] = positionY[x]+ 1;
        if (positionY[x] > gameHeight)
        {
            positionY[x] = (Math.random()*200) * -1; // new Starting Y
            positionX[x] = (Math.random()*850); // New X position
        }
    }
}
```

Function riseBatman()



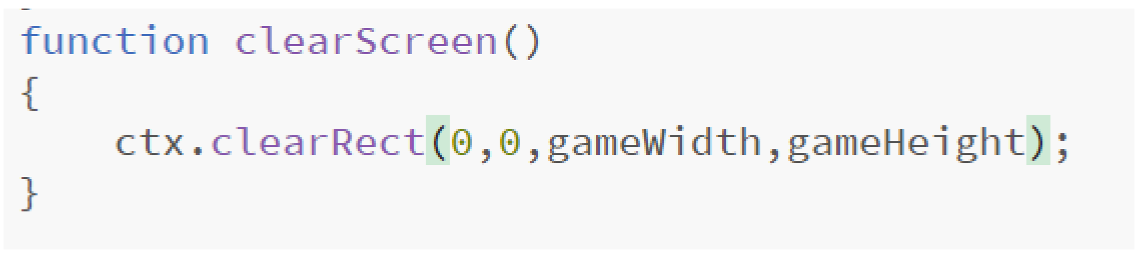
```
function riseBatman()
{
    for (x=0;x<=maxArray;x++)
    {
        //console.log(positionX[x],positionY[x]);
        drawBatman(positionX[x],positionY[x]);
        positionY[x] = positionY[x]- 1;
        if (positionY[x] > gameHeight)
        {
            positionY[x] = (Math.random()*200) * -1; // new Starting Y
            positionX[x] = (Math.random()*850); // New X position
        }
    }
}
```

Function drawBatman()



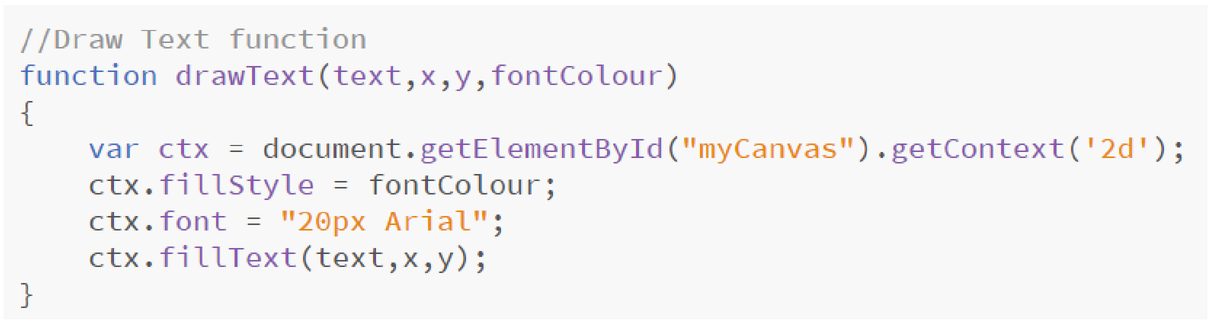
```
function drawBatman(x,y)
{
    ctx.drawImage(batman,x,y);
}
```

Function clearScreen()



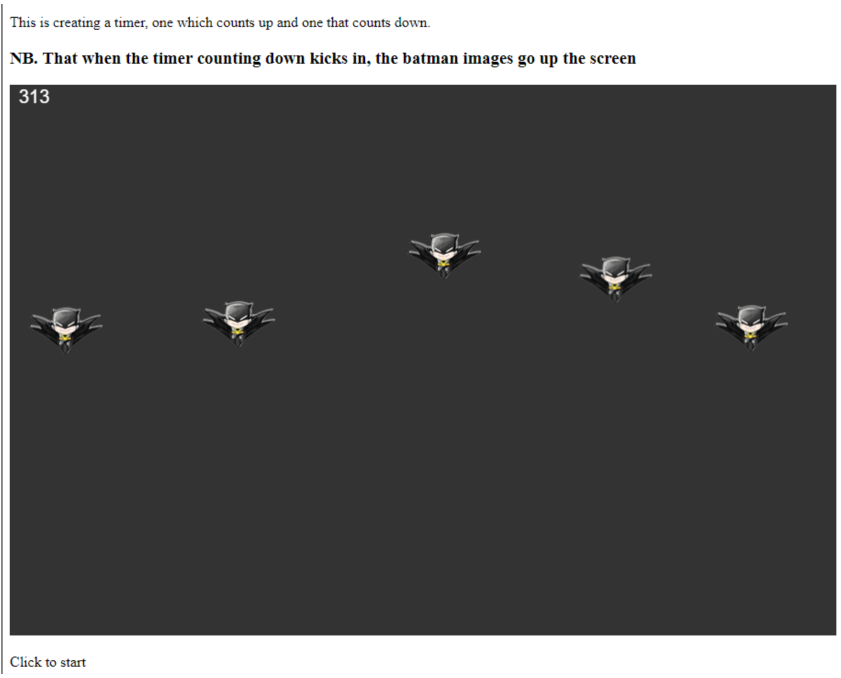
```
function clearScreen()
{
    ctx.clearRect(0,0,gameWidth,gameHeight);
}
```

Function drawText()



```
//Draw Text function
function drawText(text,x,y,fontColour)
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    ctx.fillStyle = fontColour;
    ctx.font = "20px Arial";
    ctx.fillText(text,x,y);
}
```

Result – batman going down – NB White counter

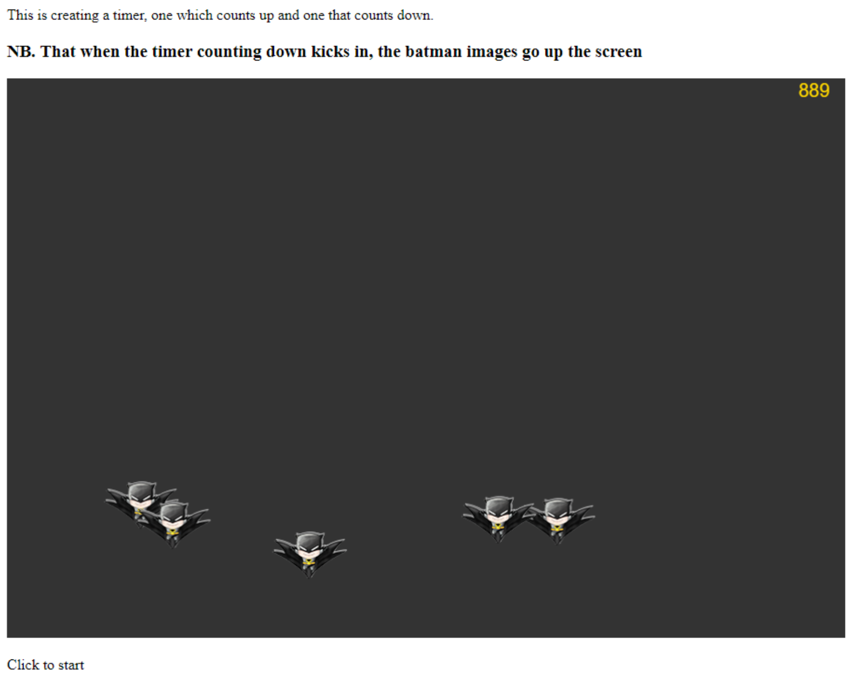


This is creating a timer, one which counts up and one that counts down.

NB. That when the timer counting down kicks in, the batman images go up the screen

Click to start

Result – batman going up – NB gold counter



This is creating a timer, one which counts up and one that counts down.

NB. That when the timer counting down kicks in, the batman images go up the screen

Click to start

Mouse Move

This page will show you how to track an image and restrain it within the canvas.

Start a new page. Let’s start with the canvas and the styles required for the page.

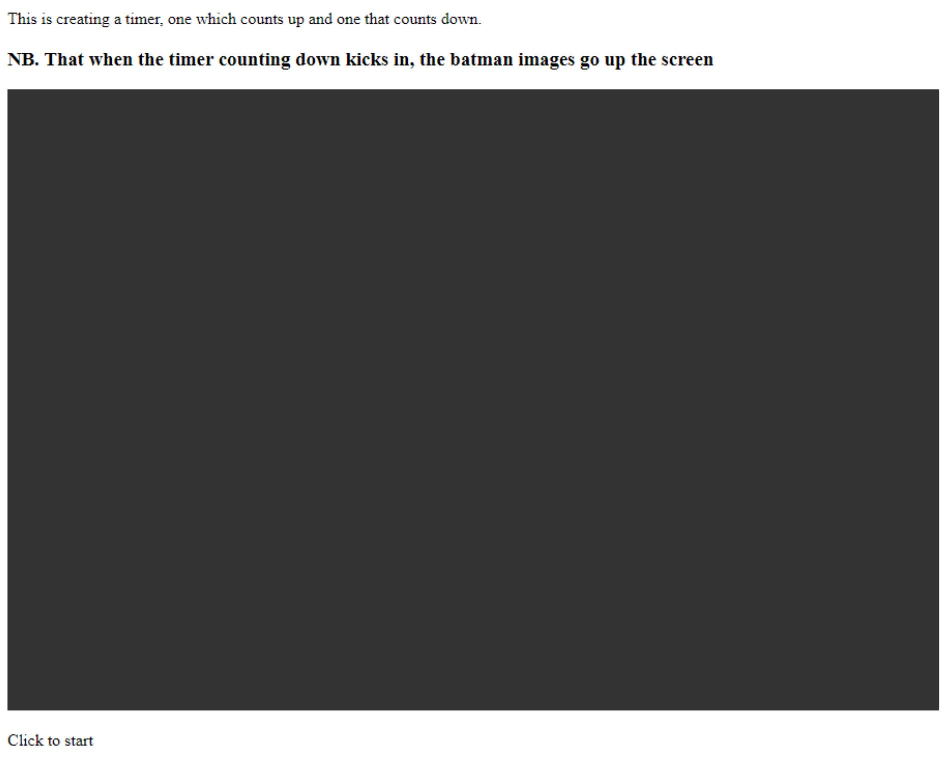
Body Code

```
<body>
    <p>The image will stick to the mouse cursor and be restrained to the canvas</p>
    <canvas id="myCanvas" width="900" height="600" class="silver" onmousemove="playerPos(event)"></canvas>
    <p onclick="start()">Click to start</p>
</body>
```

Style

```
<style type="text/css">
    .silver {background-color:#333333;}
    p {cursor: pointer;}
</style>
```

Result

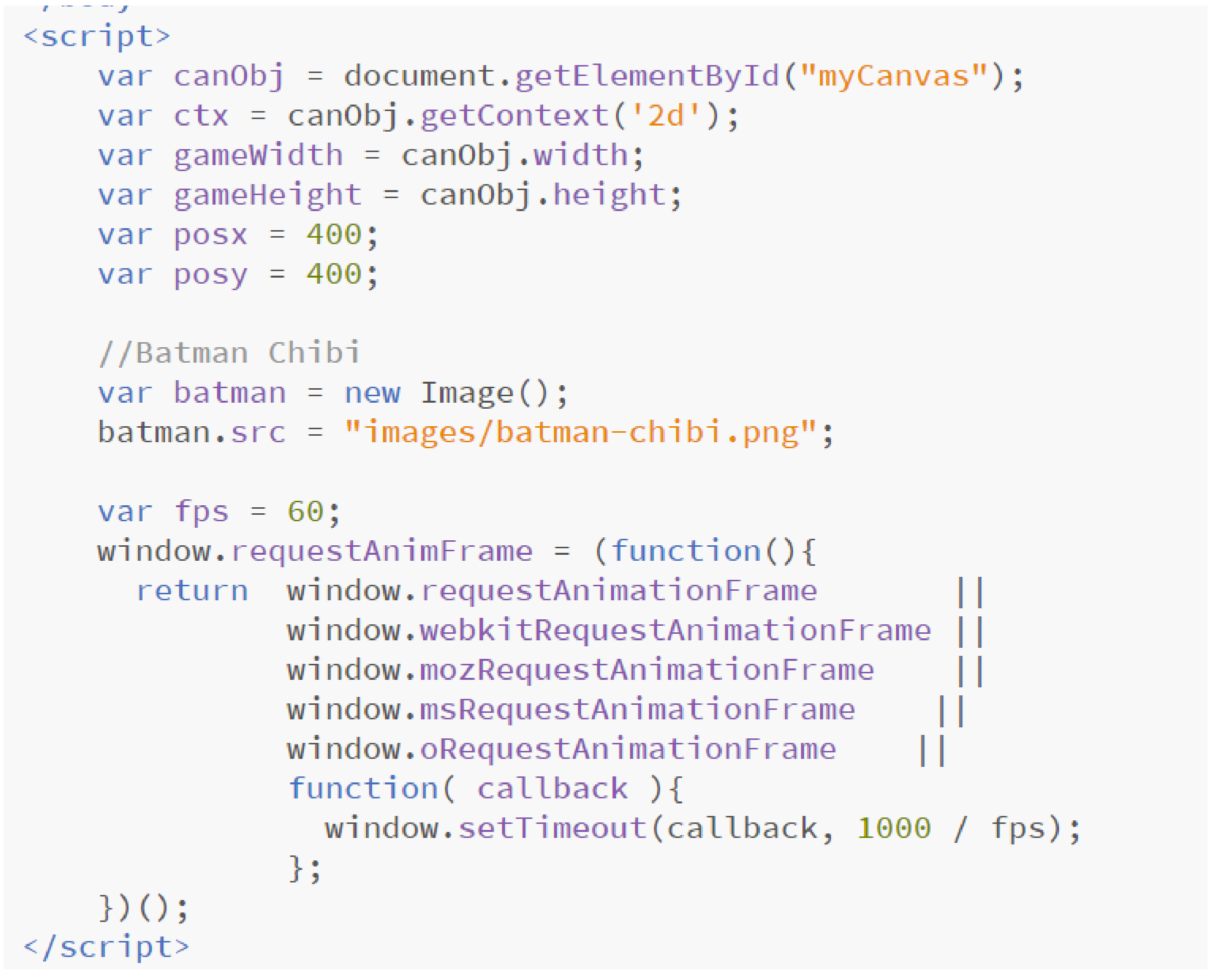


This is creating a timer, one which counts up and one that counts down.

NB. That when the timer counting down kicks in, the batman images go up the screen

Click to start

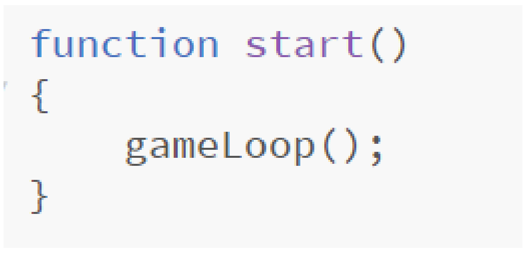
Now we add the starting global variables required for the program.



```
<script>
    var canObj = document.getElementById("myCanvas");
    var ctx = canObj.getContext('2d');
    var gameWidth = canObj.width;
    var gameHeight = canObj.height;
    var posx = 400;
    var posy = 400;

    //Batman Chibi
    var batman = new Image();
    batman.src = "images/batman-chibi.png";

    var fps = 60;
    window.requestAnimFrame = (function(){
      return  window.requestAnimationFrame       ||
              window.webkitRequestAnimationFrame ||
              window.mozRequestAnimationFrame    ||
              window.msRequestAnimationFrame     ||
              window.oRequestAnimationFrame      ||
              function( callback ){
                window.setTimeout(callback, 1000 / fps);
              };
    })();
</script>
```

Next we start to add the functions required to make it work.

Function start()



```
function start()
{
    gameLoop();
}
```

Function gameLoop

```
function gameLoop()
{
    setTimeout(function()
    {
        requestAnimFrame(start);
        clearScreen();
        drawBatman(posx,posy);//multiple batman
        drawText('X Pos',10,15,"white");
        drawText('Y Pos',60,15,"white");
        drawText(posx,10,35,"white");
        drawText(posy,60,35,"white");
    },1000/fps);
}
```

Function playerPos()

```
function playerPos(event)
{
    x = event.clientX;
    y = event.clientY;
    var winOffsetX = canObj.offsetLeft - canObj.scrollLeft;
    var winOffsetY = canObj.offsetTop - canObj.scrollTop;
    var bw = 100/2;
    var bh = 50/2;
   // console.log(bw,bh);
    posx = x - winOffsetX - bw;
    posy = y - winOffsetY - bh;
}
```

Function drawBatman()

```
function drawBatman(x,y)
{
    //restraining batman in the canvas
    y = restrainCanvasHeight(y);
    x = restrainCanvasWidth(x);
    ctx.drawImage(batman,x,y);
}
```

Function clearScreen()

```
function clearScreen()
{
    ctx.clearRect(0,0,gameWidth,gameHeight);
}
```

Function restrainCanvasHeight()

```
function restrainCanvasHeight(y)
{
    if (y> (gameHeight-50)) //ground
    {
        y = gameHeight-50;
    }
    if (y<=0) // ceiling
    {
            y = 0;
    }
    return y;
}
```

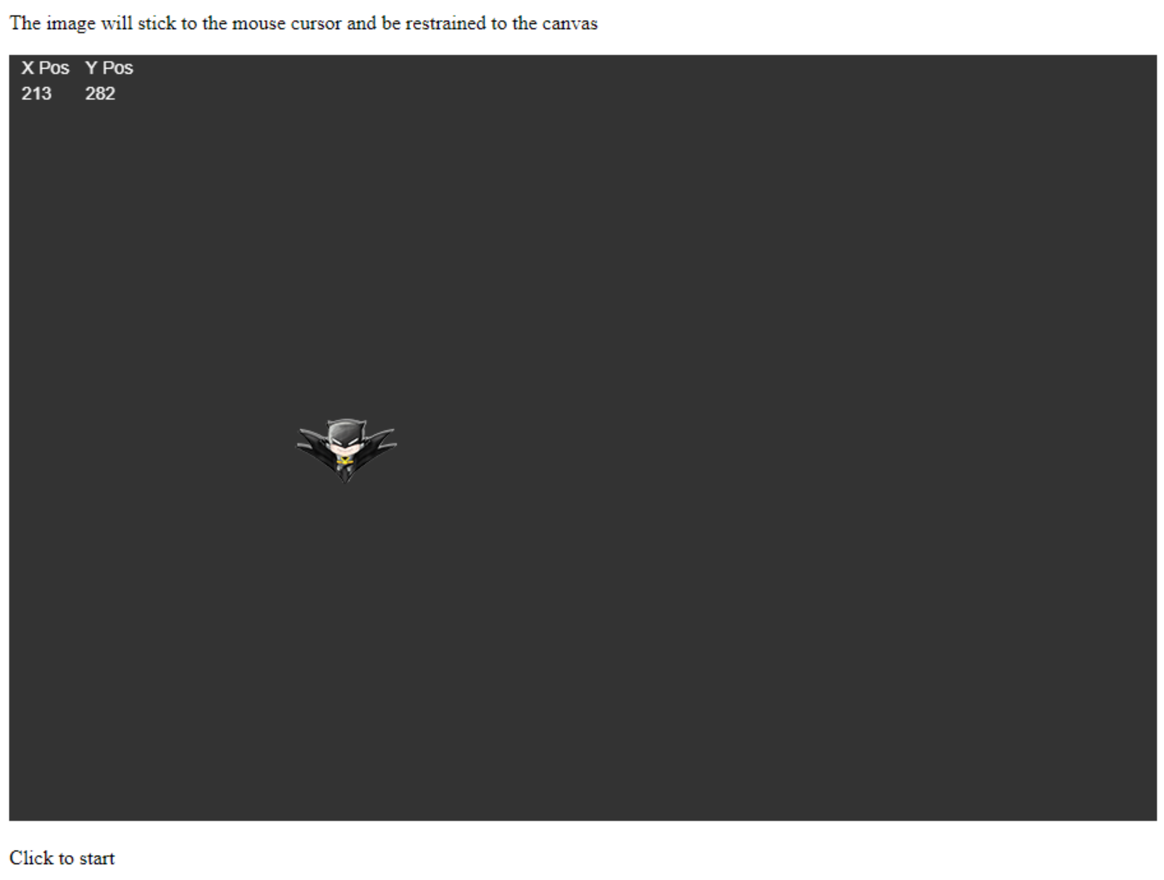
Function restrainCanvasWidth()

```
function restrainCanvasWidth(x)
{
    if(x<=0) //left side
    {
        x = 0;
    }
    if(x>= (gameWidth-100)) //right side
    {
        x= gameWidth-100;
    }
return x;
}
```

Function drawText()

```
function drawText(text,x,y,fontColour)
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    ctx.fillStyle = fontColour;
    ctx.font = "14px Arial";
    ctx.fillText(text,x,y);
}
```

Result



The image will stick to the mouse cursor and be restrained to the canvas

```
X Pos   Y Pos
213     282
```

Click to start

Character direction with changing sprites

This page will draw a chibi character that will change the direction it faces depending on where the mouse is. There is also 'gravity' pulling the character down but, with a left click the character jumps up/

Start a new page. Let's start with the canvas and the styles required for the page.

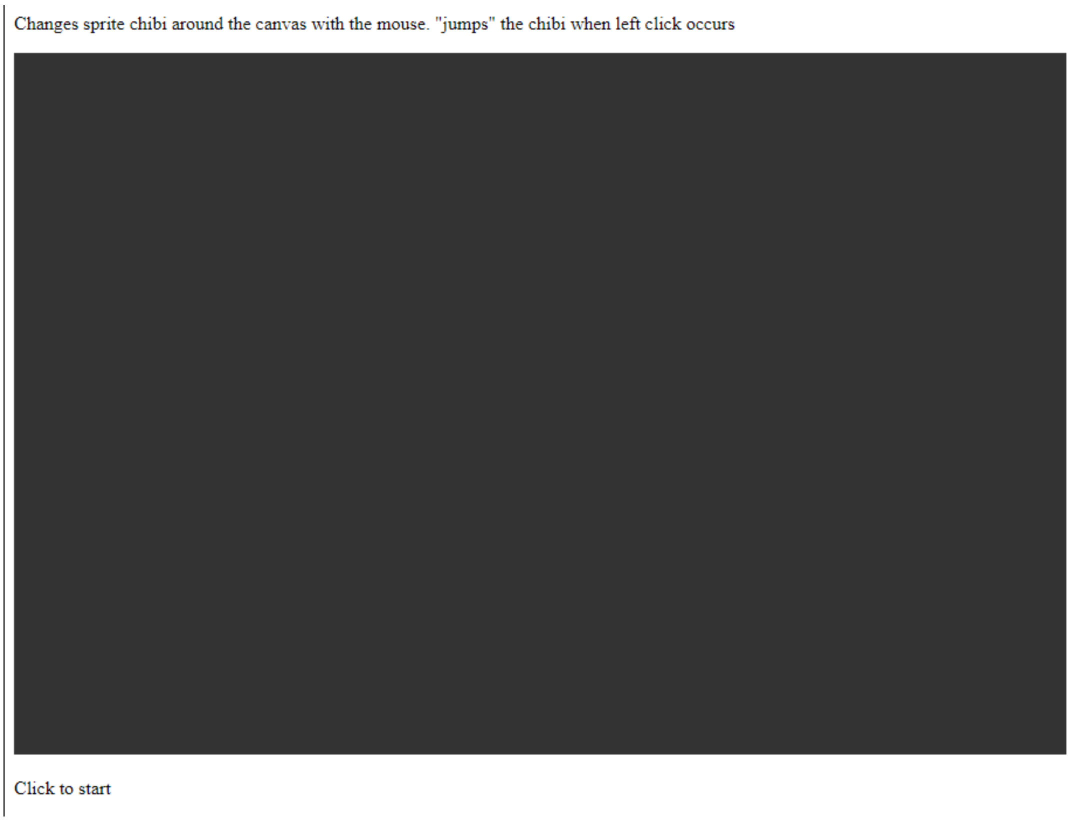
Body Code

```
<body>
    <p>
    Changes sprite chibi around the canvas with the mouse.
    "jumps" the chibi when left click occurs
    </p>
    <canvas id="myCanvas" width="900" height="600" class="silver" onmousemove="mousePos(event)"></canvas>
    <p onclick="start()">Click to start</p>
</body>
```

Style

```
<style type="text/css">
    .silver {background-color:#333333;}
    p {cursor: pointer;}
</style>
```

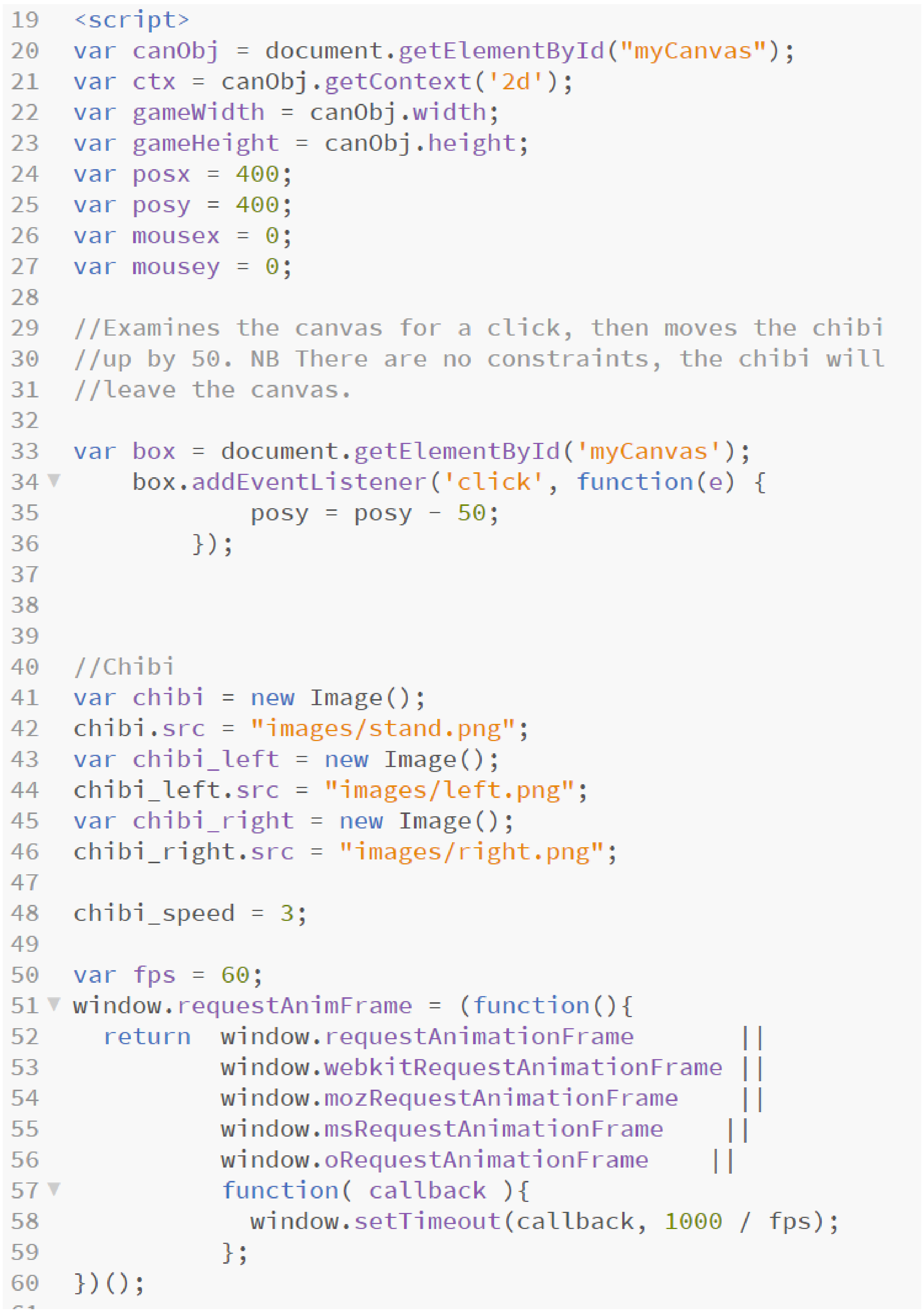
Result



Changes sprite chibi around the canvas with the mouse. "jumps" the chibi when left click occurs

Click to start

Now we add the starting global variables required for the program.



```
<script>
var canObj = document.getElementById("myCanvas");
var ctx = canObj.getContext('2d');
var gameWidth = canObj.width;
var gameHeight = canObj.height;
var posx = 400;
var posy = 400;
var mousex = 0;
var mousey = 0;

//Examines the canvas for a click, then moves the chibi
//up by 50. NB There are no constraints, the chibi will
//leave the canvas.

var box = document.getElementById('myCanvas');
    box.addEventListener('click', function(e) {
            posy = posy - 50;
        });



//Chibi
var chibi = new Image();
chibi.src = "images/stand.png";
var chibi_left = new Image();
chibi_left.src = "images/left.png";
var chibi_right = new Image();
chibi_right.src = "images/right.png";

chibi_speed = 3;

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

Now that we have the global vars in place, we need to add the following functions

Function start()

```
function start()
{
        gameLoop();
}
```

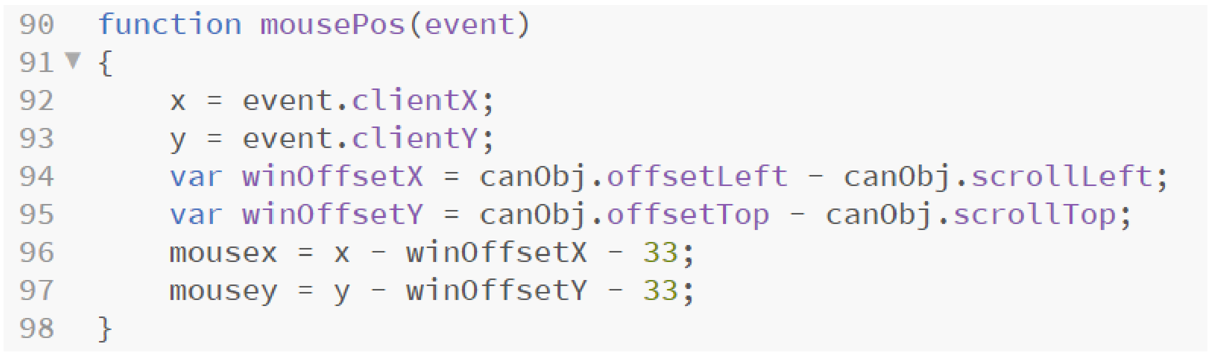
Function gameLoop()

```
function gameLoop()
{
        setTimeout(function()
        {
                requestAnimFrame(start);
                clearScreen();
                drawChibi();
                gravity();
        },1000/fps);
}
```

Function gravity()

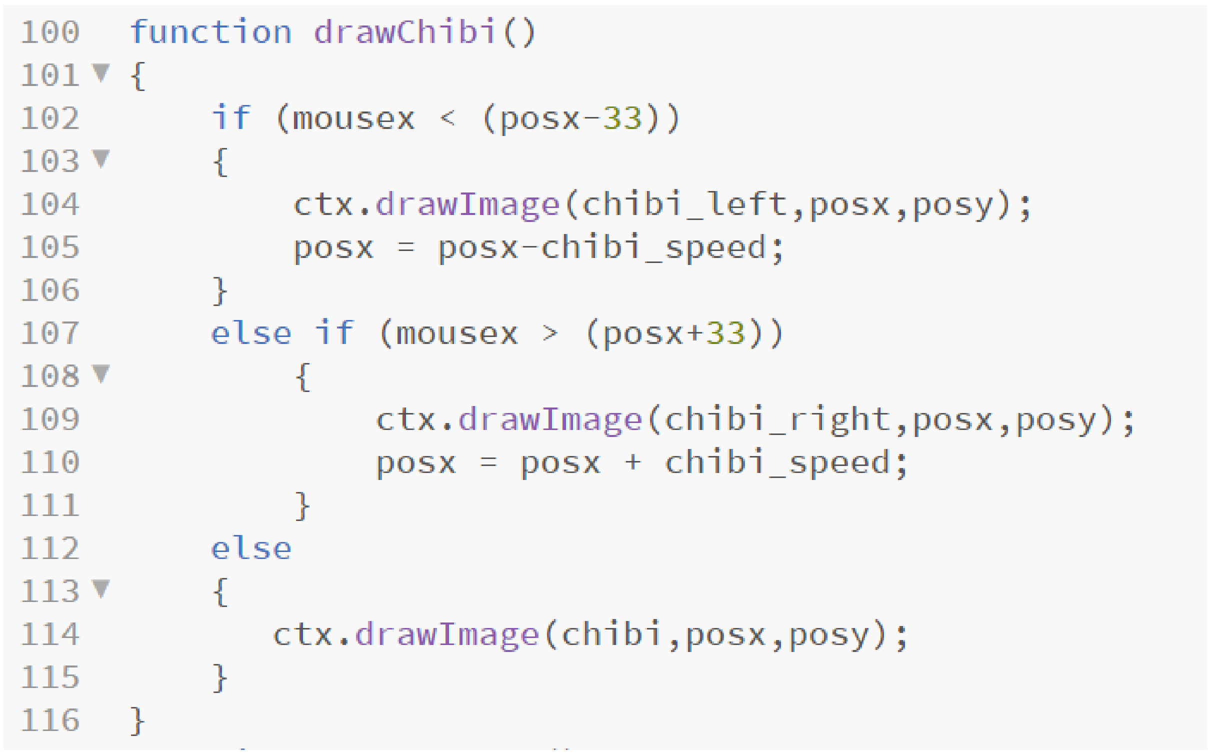
```
function gravity()
{
        if(posy !=500)
                {
                if (posy<=500)
                        { posy++}
        }
        else
        {posy=500;}
      // console.log(posy);

}
```

Function mousePos()



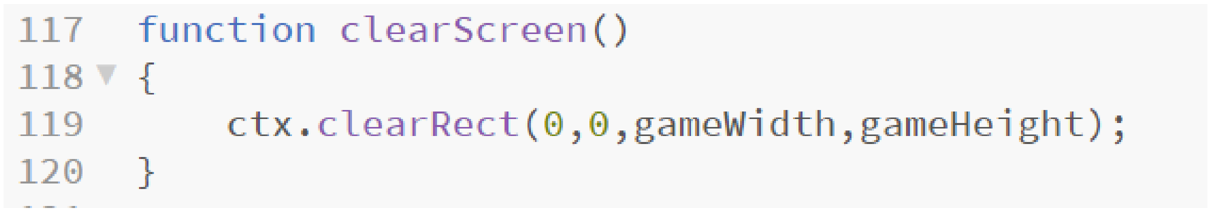
```
function mousePos(event)
{
    x = event.clientX;
    y = event.clientY;
    var winOffsetX = canObj.offsetLeft - canObj.scrollLeft;
    var winOffsetY = canObj.offsetTop - canObj.scrollTop;
    mousex = x - winOffsetX - 33;
    mousey = y - winOffsetY - 33;
}
```

Function drawChibi()



```
function drawChibi()
{
    if (mousex < (posx-33))
    {
        ctx.drawImage(chibi_left,posx,posy);
        posx = posx-chibi_speed;
    }
    else if (mousex > (posx+33))
        {
            ctx.drawImage(chibi_right,posx,posy);
            posx = posx + chibi_speed;
        }
    else
    {
       ctx.drawImage(chibi,posx,posy);
    }
}
```

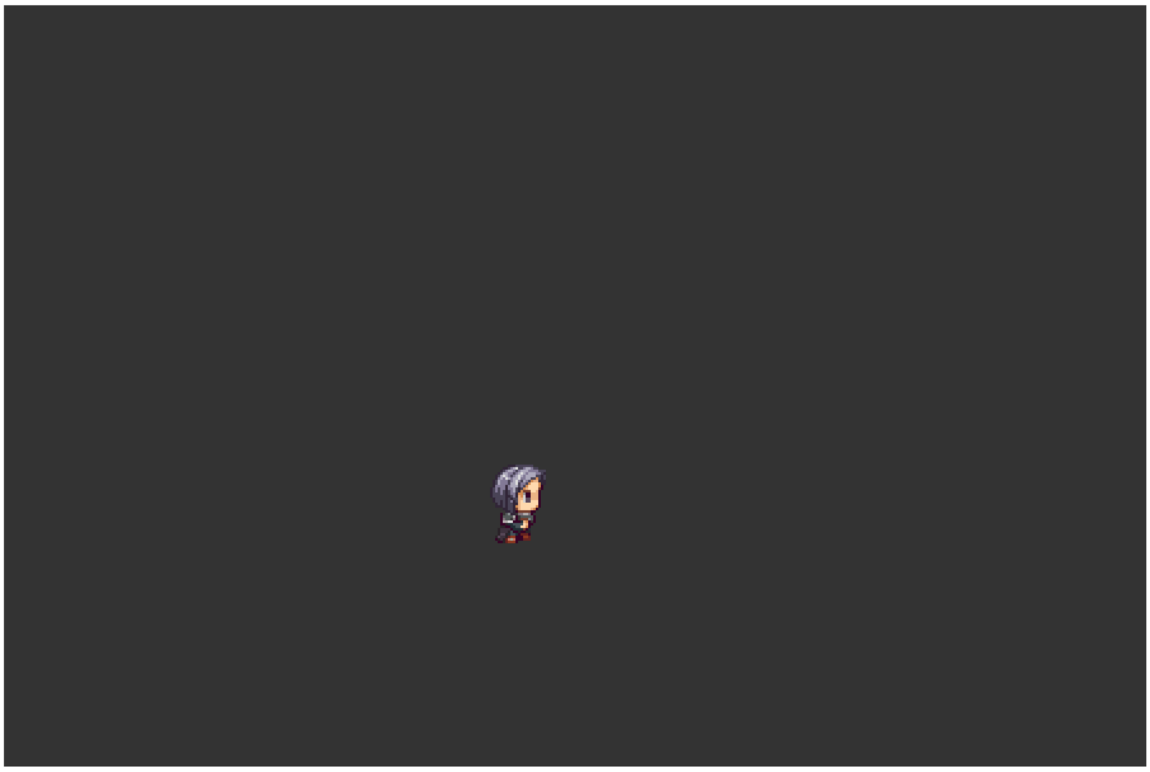
Function clearScreen()



```
function clearScreen()
{
    ctx.clearRect(0,0,gameWidth,gameHeight);
}
```

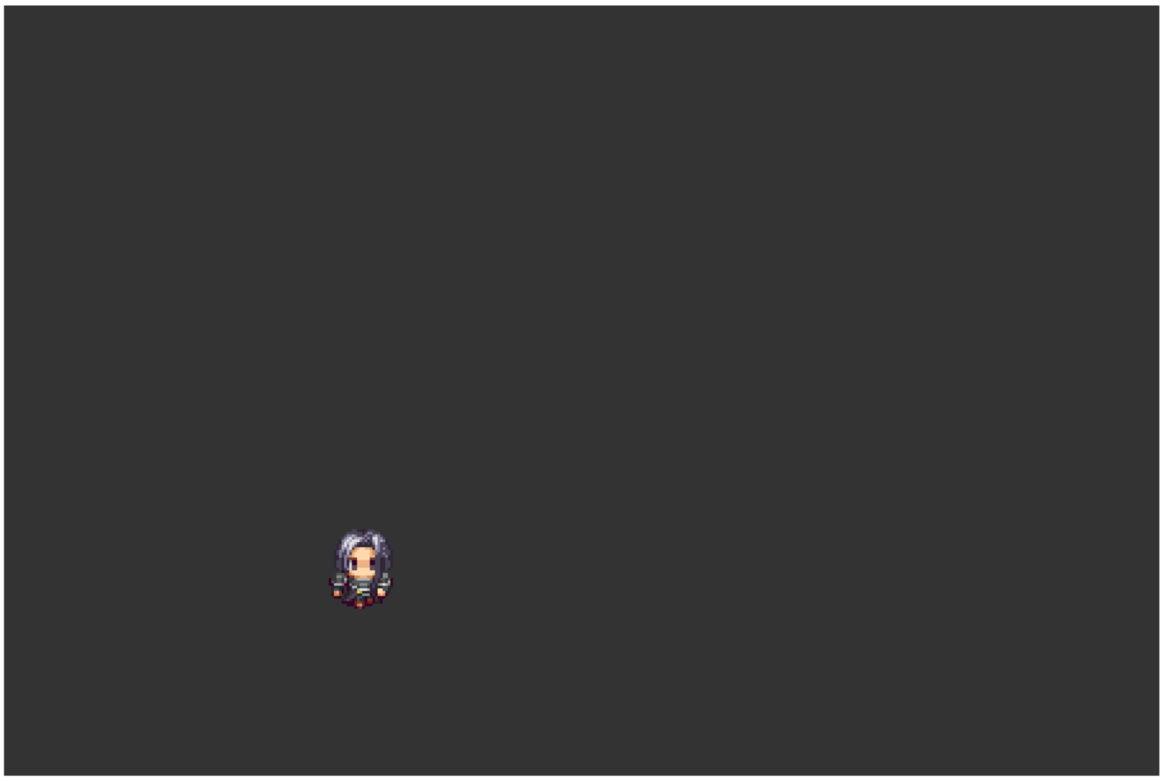
Result

Changes sprite chibi around the canvas with the mouse. "jumps" the chibi when left click occurs



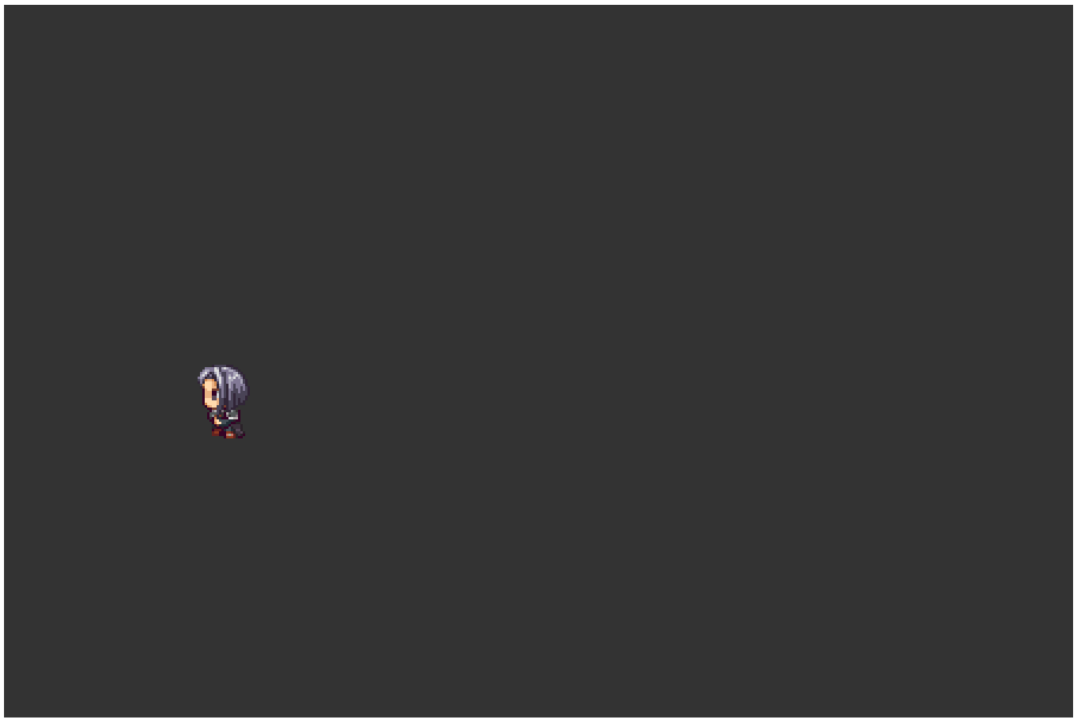
Click to start

Changes sprite chibi around the canvas with the mouse. "jumps" the chibi when left click occurs



Click to start

Changes sprite chibi around the canvas with the mouse. "jumps" the chibi when left click occurs



Click to start

Backup

This concludes tutorial. Save your work to USB, student drive, Onedrive, Dropbox or zip and email the folder to yourself.