Tutorial 5

Editor – Brackets

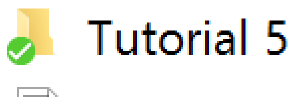
Goals

Create a website showcasing the following techniques

- Animated backgrounds
- Animated game elements

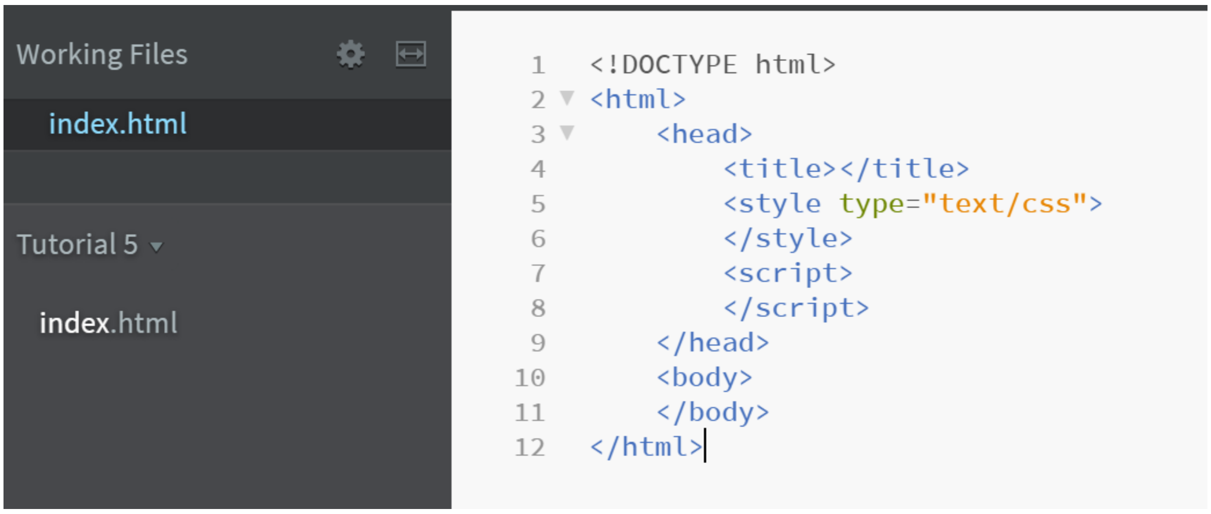
Website

- Create a folder on the desktop called tutorial 5



○

- Open Brackets
- Create the following structure and index.html file

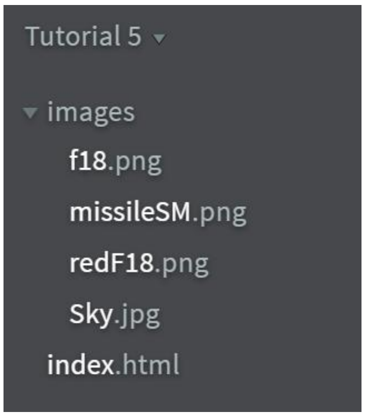


```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <script>
        </script>
    </head>
    <body>
    </body>
</html>
```

-

Index Page

To start with we need to download the resources files form l@G, this is called week6files.zip, extract the files, then copy all of the images into an image folder inside your tutorial 5 folder. You should have the following structure



```
Tutorial 5
  images
    f18.png
    missileSM.png
    redF18.png
    Sky.jpg
  index.html
```

Now let's start with a canvas that we can put our graphics on.

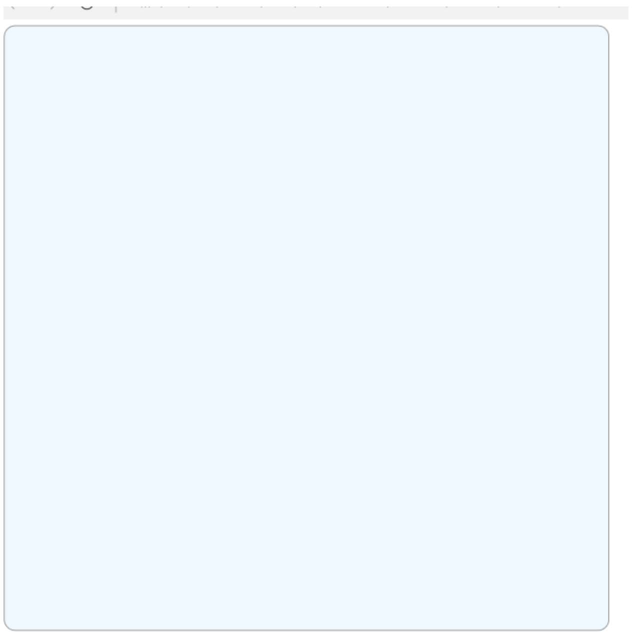
Div code

```
<body>
    <canvas id="myCanvas" width="800" height="800" onClick="handleClick(event)" class="colCanvas"></canvas>
</body>
```

Css Code

```
<style type="text/css">
    .colCanvas{background-color: aliceblue; border: 1px solid gray; border-radius: 15px;}
</style>
```

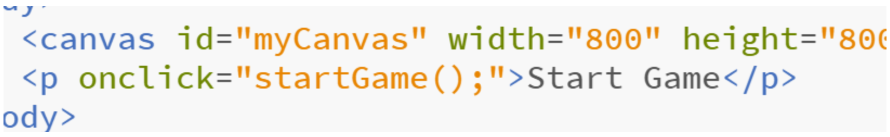
Result



Next we add the background image to the canvas, we will start off by getting the image to show, then we will add in some animation to make the background scroll, so it looks as though there is movement on the screen.

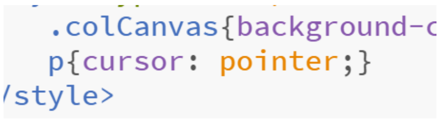
To make this happen we will use multiple functions, we'll also put in the shell for handleClick as even though we aren't using right now, it will be needed later on. So, we'll make a piece of text clickable, then load up all the elements we need.

Div



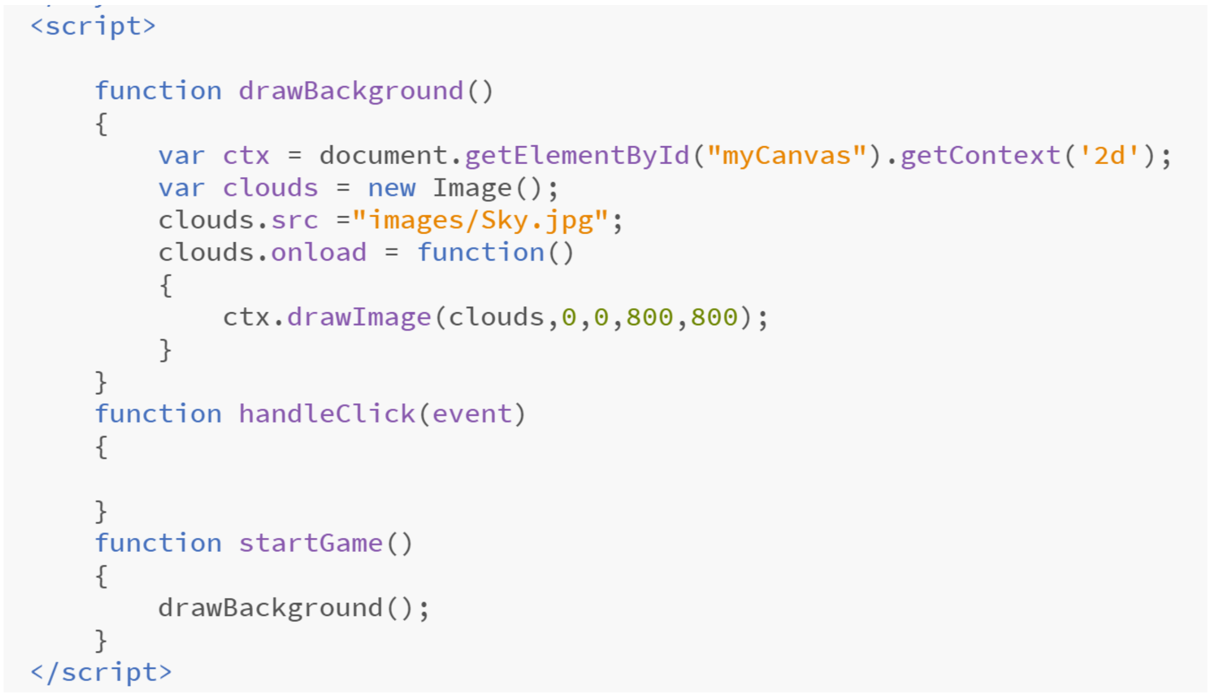
```
 <canvas id="myCanvas" width="800" height="80
 <p onclick="startGame();">Start Game</p>
ody>
```

Css



```
  .colCanvas{background-c
  p{cursor: pointer;}
/style>
```

Javascript

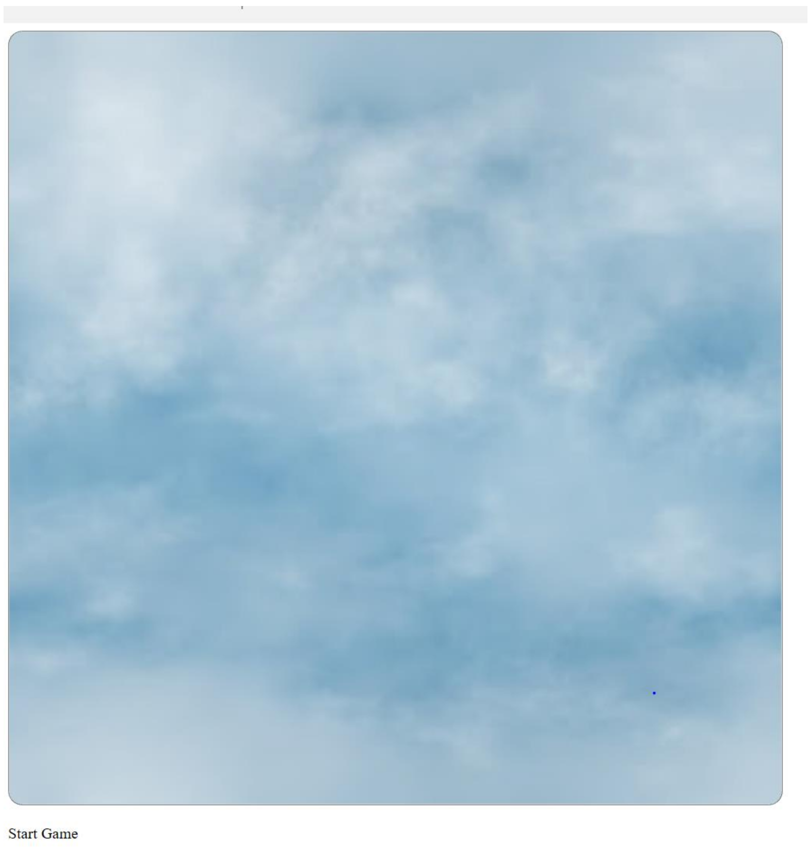


```
 <script>

     function drawBackground()
     {
         var ctx = document.getElementById("myCanvas").getContext('2d');
         var clouds = new Image();
         clouds.src ="images/Sky.jpg";
         clouds.onload = function()
         {
             ctx.drawImage(clouds,0,0,800,800);
         }
     }
     function handleClick(event)
     {

     }
     function startGame()
     {
         drawBackground();
     }
 </script>
```

The following is the result after the "start game" text has been clicked.

Remember to have your console open so you can see if there are any mistakes during coding.

Result



This is a good start but it’s not moving, so now, we are going to make it move. To do this, we will modify the way the information is currently stored. If you notice, the X and Y co-ordinates for the background are classified as 0 and 0 respectively, we’ll want to put this in a variable so we can manipulate them when we start the game loop.

Make the following changes to the javascript.

Javascript

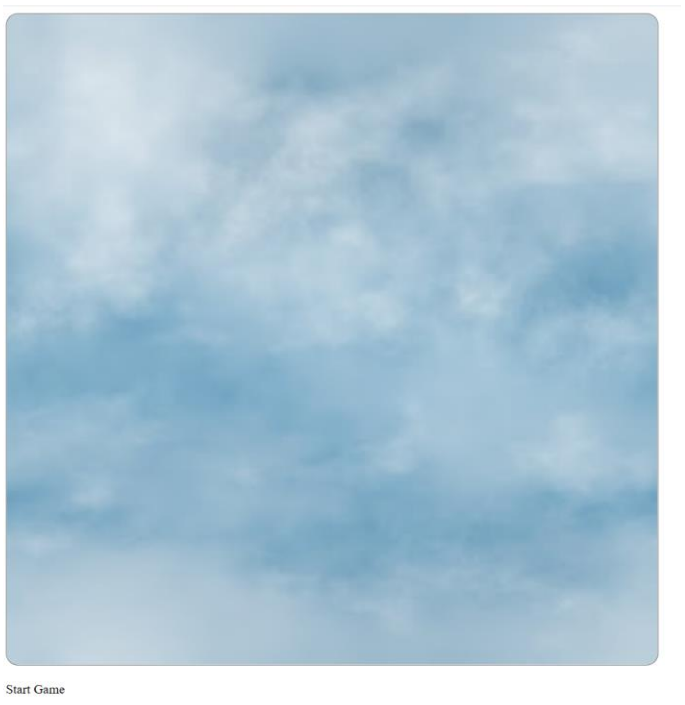
```
<script>
    var cloudx = 0;
    var cloudy = 0;

    function drawBackground()
    {
        var ctx = document.getElementById("myCanvas").getContext('2d');
        var clouds = new Image();
        clouds.src ="images/Sky.jpg";
        clouds.onload = function()
        {
            ctx.drawImage(clouds,cloudx,cloudy,800,800);
        }
    }
```

Now that we have put in variables for positioning, we need to make the game loop and then modify aspects of the background.

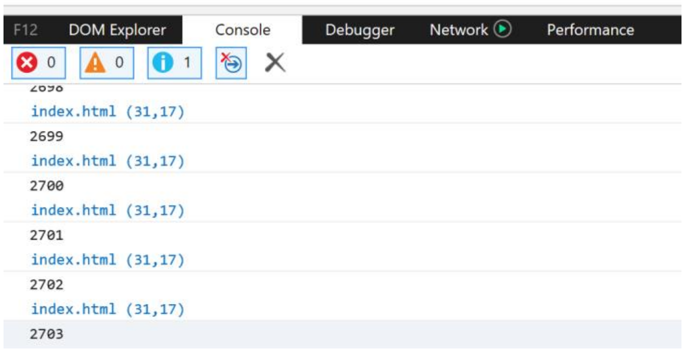
Javascript

```
function handleClick(event)
{

}
function gameLoop()
{
    drawBackground();
    var t=setTimeout("gameLoop()", 1);
    console.log(t);
}
function startGame()
{
    gameLoop();
}
```

Result



Start Game



```
2698
index.html (31,17)
2699
index.html (31,17)
2700
index.html (31,17)
2701
index.html (31,17)
2702
index.html (31,17)
2703
```

As you can see, the variable t is being incremented; 1000 milliseconds = 1 second. So, now that we have a loop occurring, notice how the gameloop calls itself, which is why the variable t increments, we can start to modify the clouds positioning.

So, we will start with moving the cloud down the screen. This is the Y variable that needs to be changed.

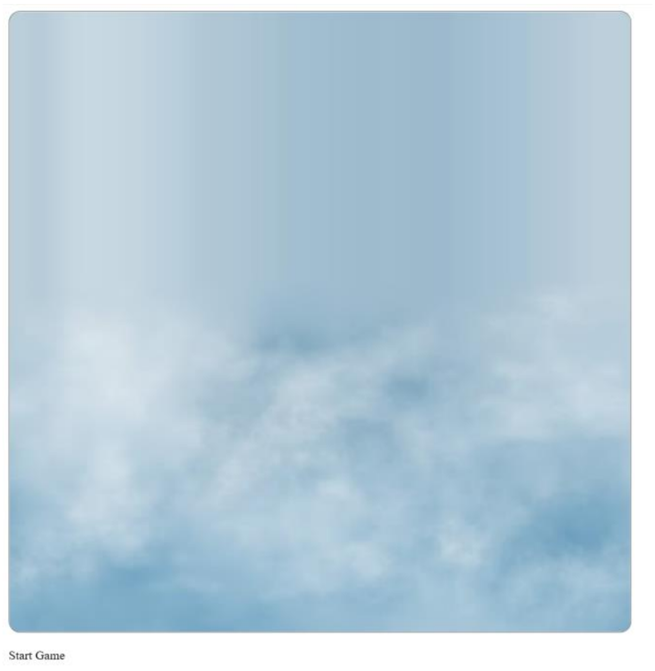
Javascript

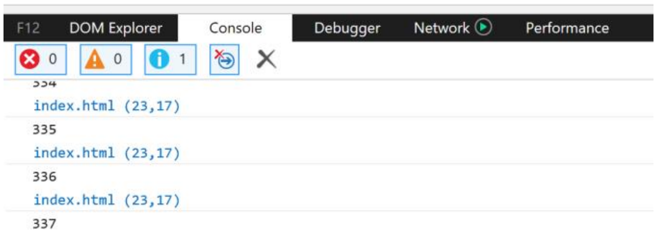
```
function drawBackground()
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    var clouds = new Image();
    clouds.src ="images/Sky.jpg";
    clouds.onload = function()
    {
        ctx.drawImage(clouds,cloudx,cloudy,800,800);
    }
    cloudy = cloudy+1;
    console.log(cloudy);
}

function handleClick(event)
{

}
function gameLoop()
{
    drawBackground();
    var t=setTimeout("gameLoop()", 1);
    //console.log(t);
}
```

Notice that the console log has been commented out in gameLoop, this is because we already know what is happening with variable t.

When this is run, you should see something like the following

Result





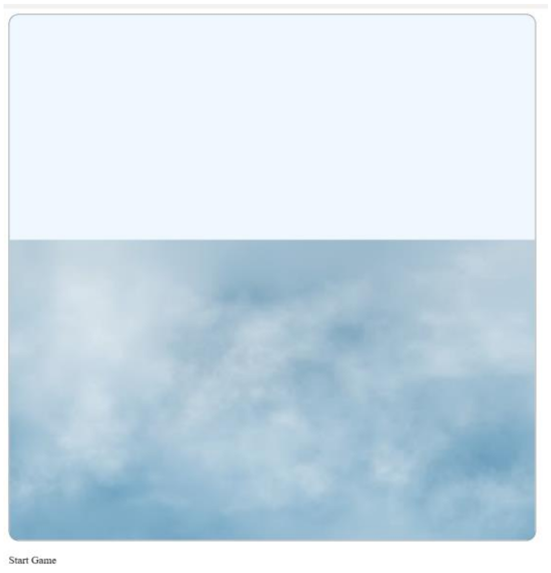
```
334
index.html (23,17)
335
index.html (23,17)
336
index.html (23,17)
337
```

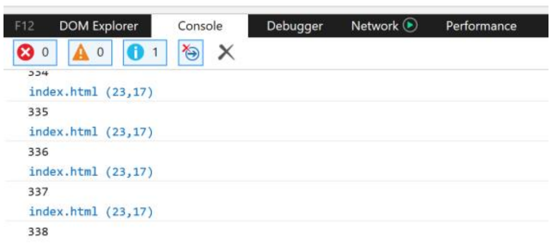
The cloud image is being stretched out leaving a trail of cloud behind it. So, we have moment but it is not clean, to fix this we will put a clear screen function in place. So, using the resetCanvas function that we used in Week 3, we can add this to the page and then turn it on by calling the function in the gameloop function, just before the drawbackground.

Javascript

```
function resetCanvas()
{
    var canvas = document.getElementById("myCanvas");
    context = canvas.getContext('2d');
    context.clearRect(0,0,canvas.width,canvas.height);
}
function handleClick(event)
{

}
function gameLoop()
{
    resetCanvas();
    drawBackground();
    var t=setTimeout("gameLoop()", 1);
    //console.log(t);
}
```

Result

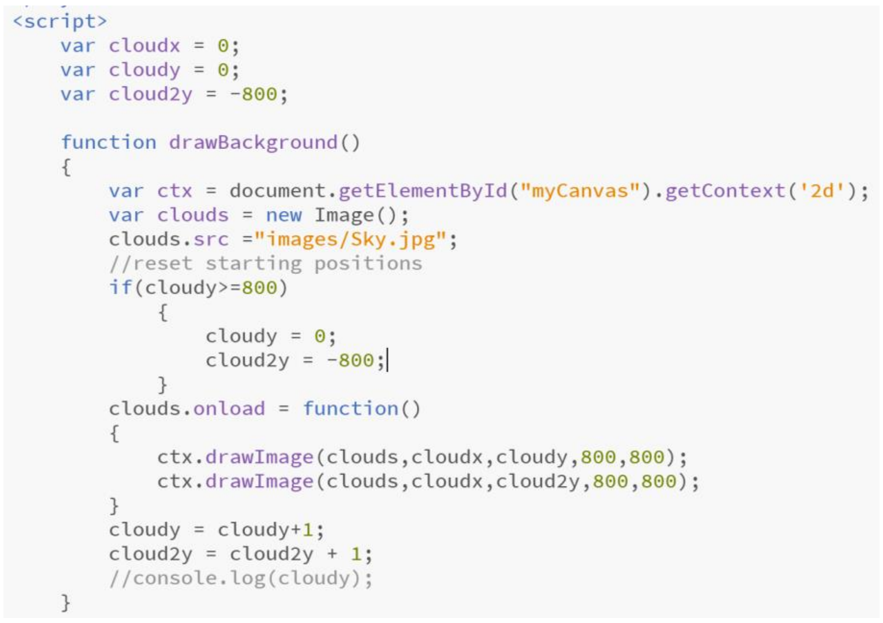




```
index.html (23,17)
335
index.html (23,17)
336
index.html (23,17)
337
index.html (23,17)
338
```

Now you can clearly see the image moving down the screen, though, it is not repeating, to make the background seem constant, we will need to modify the drawbackground to allow for an additional image to load and move. This will be done by using the same load capability, but we will add an if statement to reset the images, so we don't end up with a clear blue screen.

Javascript



```
<script>
    var cloudx = 0;
    var cloudy = 0;
    var cloud2y = -800;

    function drawBackground()
    {
        var ctx = document.getElementById("myCanvas").getContext('2d');
        var clouds = new Image();
        clouds.src ="images/Sky.jpg";
        //reset starting positions
        if(cloudy>=800)
            {
                cloudy = 0;
                cloud2y = -800;
            }
        clouds.onload = function()
        {
            ctx.drawImage(clouds,cloudx,cloudy,800,800);
            ctx.drawImage(clouds,cloudx,cloud2y,800,800);
        }
        cloudy = cloudy+1;
        cloud2y = cloud2y + 1;
        //console.log(cloudy);
    }
```

Notice that there is a new global variable called clouds2y, this is going to be the y position for the second cloud image. See how, the same image is drawn twice within the onload function, this because we feed the two images directly after one another.

The if statement used, basically says that when the first cloud image increases the y value from 0 to 800, then reset both of them back to their starting values and continue the loop. In this way, we give the illusion that it is a constant moving background.

Now that we have a background moving, we can draw in the planes. We will draw in the plane that the user will use and an enemy red plane.

Javascript

There are multiple areas in which the javascript needs to be modified, this is broken down below.

Global Vars

```
<script>
    var cloudx = 0;
    var cloudy = 0;
    var cloud2y = -800;
    var playerx = 400;
    var playery = 600;
    var redPlaneX = 100;
    var redPlaneY = 0;
```

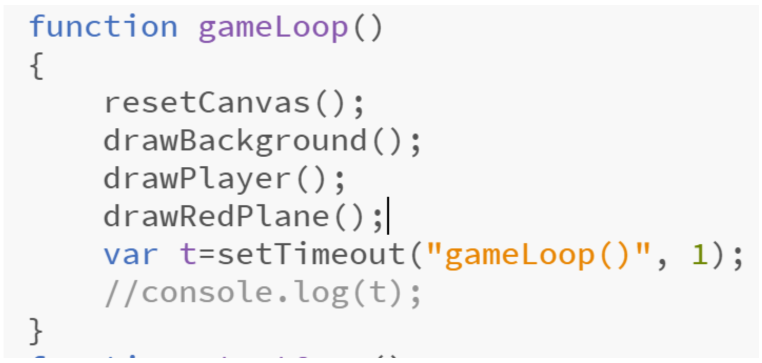
Function Player

```
function drawPlayer()
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    var plane = new Image();
    plane.src = "images/f18.png";
    plane.onload = function()
    {
        ctx.drawImage(plane,playerx,playery);
    }
}
```

Function redPlane

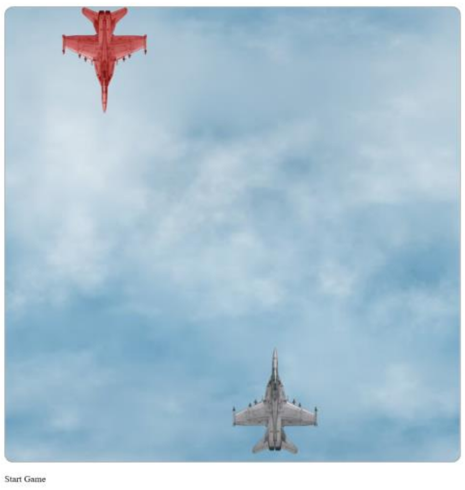
```
function drawRedPlane()
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    var redPlane = new Image();
    redPlane.src = "images/redF18.png";
    redPlane.onload = function()
    {
        ctx.drawImage(redPlane,redPlaneX, redPlaneY);
    }
}
```

Function gameLoop



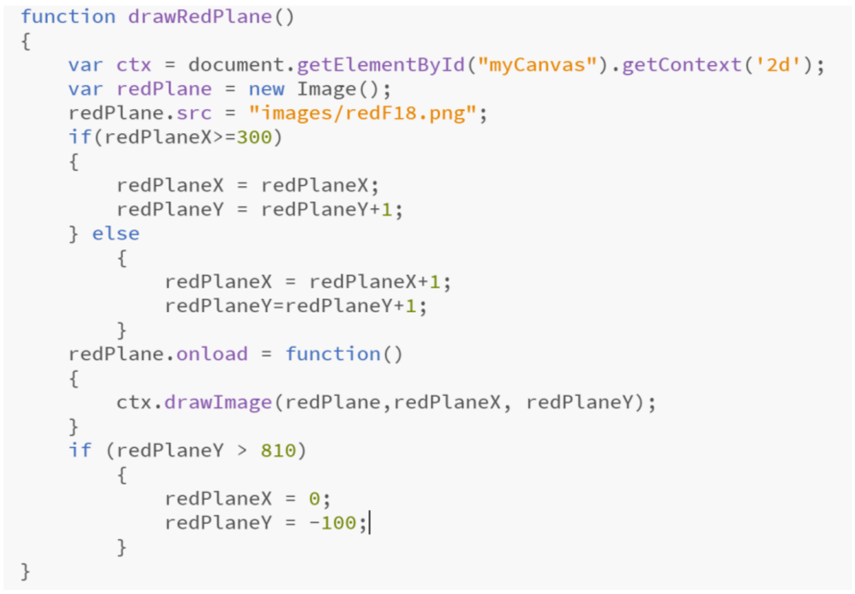
```
function gameLoop()
{
    resetCanvas();
    drawBackground();
    drawPlayer();
    drawRedPlane();
    var t=setTimeout("gameLoop()", 1);
    //console.log(t);
}
```

Result



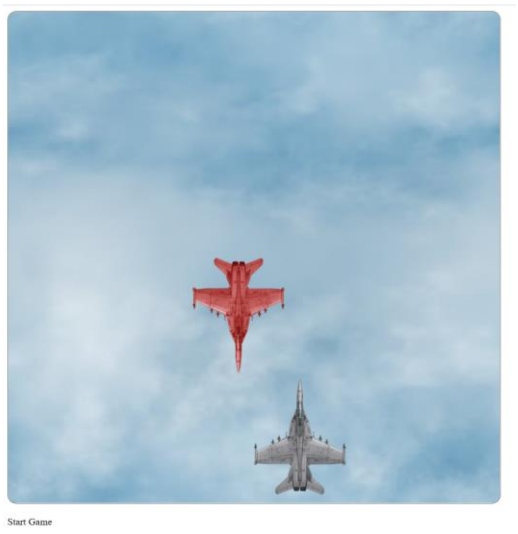
Now that we have our elements on the screen, we need to add some movement. This will be similar to how we manipulated the clouds, let's do the red plane first. What we want to do is to have it fly down on an angle until about halfway and then fly straight off the bottom of the screen. Once the red plane has left the screen, we will make it reappear and fly down again.

Javascript



```
function drawRedPlane()
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    var redPlane = new Image();
    redPlane.src = "images/redF18.png";
    if(redPlaneX>=300)
    {
        redPlaneX = redPlaneX;
        redPlaneY = redPlaneY+1;
    } else
        {
            redPlaneX = redPlaneX+1;
            redPlaneY=redPlaneY+1;
        }
    redPlane.onload = function()
    {
        ctx.drawImage(redPlane,redPlaneX, redPlaneY);
    }
    if (redPlaneY > 810)
        {
            redPlaneX = 0;
            redPlaneY = -100;
        }
}
```

Result



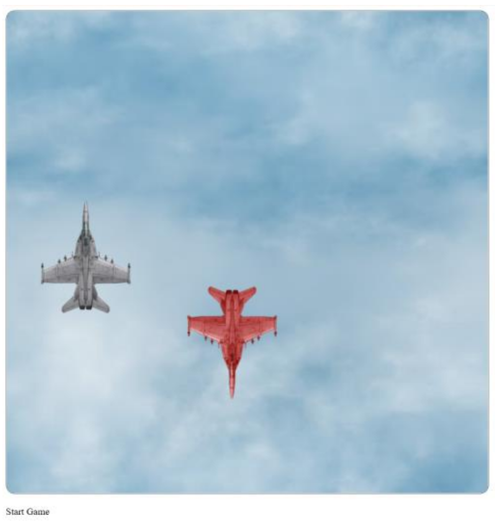
Now that we have a moving red plane, it's time to give the player the chance to control their own plane. To do this, we will modify the handleClick function

Javascript

```
function handleClick(event)
{
    var x = event.clientX;
    var y = event.clientY;
    //adding the maths below, the cursor is centered on the image
    //remember playerx and playery are global vars which is why
    // we can make the change here to effect the players position.
    playerx = x - 75; // 75 is half the width
    playery = y - 93; // 50 is half the height
}
```

If you run this, this then gives you the ability to click on the canvas and have the plane appear where you click.

Result



Next we will add the missiles.

Javascript

Global vars

```
var missileX = 600;
var missileY = -100;
var missileSpeed = 3;
```

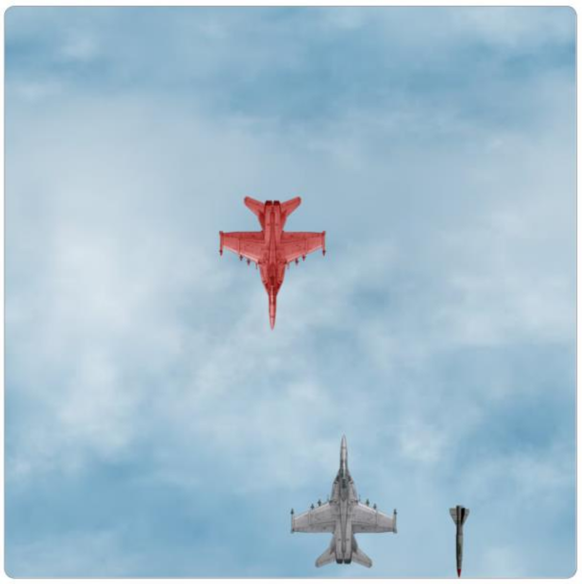
Function drawmissile

```
function drawMissile()
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    var missile = new Image();
    missile.src = "images/missileSM.png";
    missile.onload = function()
    {
        ctx.drawImage(missile,missileX,missileY);
    }
    missileY = missileY + missileSpeed;
}
```

Function gameloop

```
function gameLoop()
{
    resetCanvas();
    drawBackground();
    drawPlayer();
    drawRedPlane();
    drawMissile();
    var t=setTimeout("gameLoop()", 1);
    //console.log(t);
}
```

Result



Start Game

Next we'll add multiple missiles using arrays. Arrays are a way of storing multiple elements of the same type in a single variable. If you think of a variable as a named bucket in memory, you can think of an array as a named shelf, with multiple buckets you can access.

Javascript

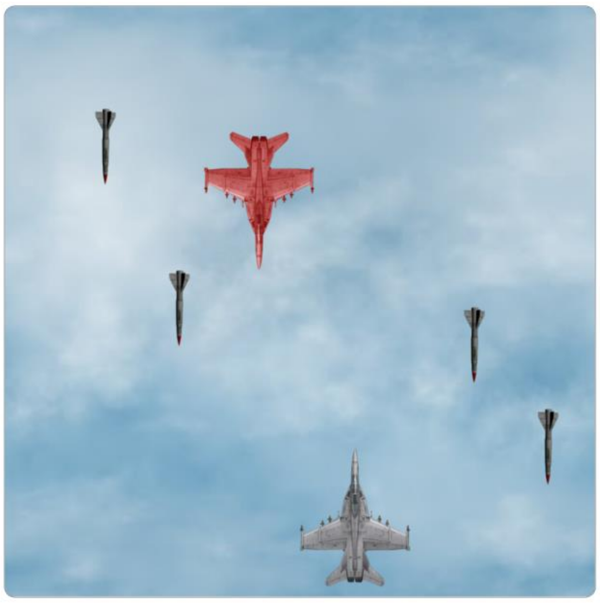
Global Vars

```
var missileX = [600,200,700,100];
var missileY = [-100,-150,-300,-200];
var missileSpeed = [3,3,5,2];
```

Function drawMissile

```
function drawMissile()
{
    var ctx = document.getElementById("myCanvas").getContext('2d');
    var missile = new Image();
    missile.src = "images/missileSM.png";
    missile.onload = function()
    {
        for(x=0;x<=4;x++)
        {
            ctx.drawImage(missile,missileX[x],missileY[x]);
            missileY[x] = missileY[x] + missileSpeed[x];
        }
    }

}
```

Result



Start Game

Backup

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