Tutorial 3

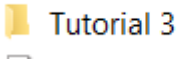
Editor – Brackets

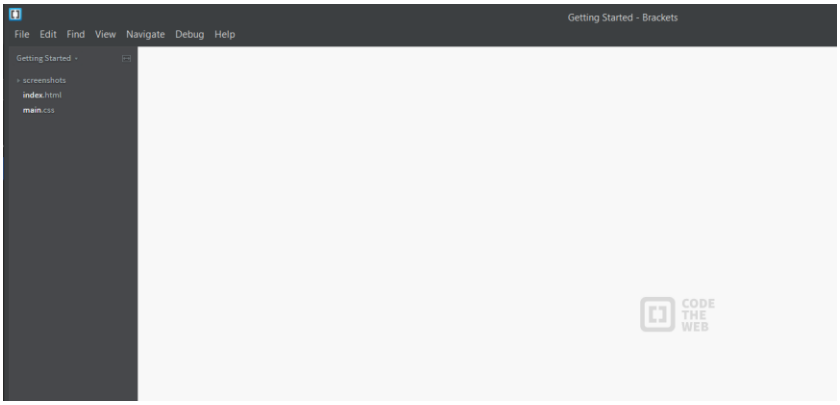
Goals

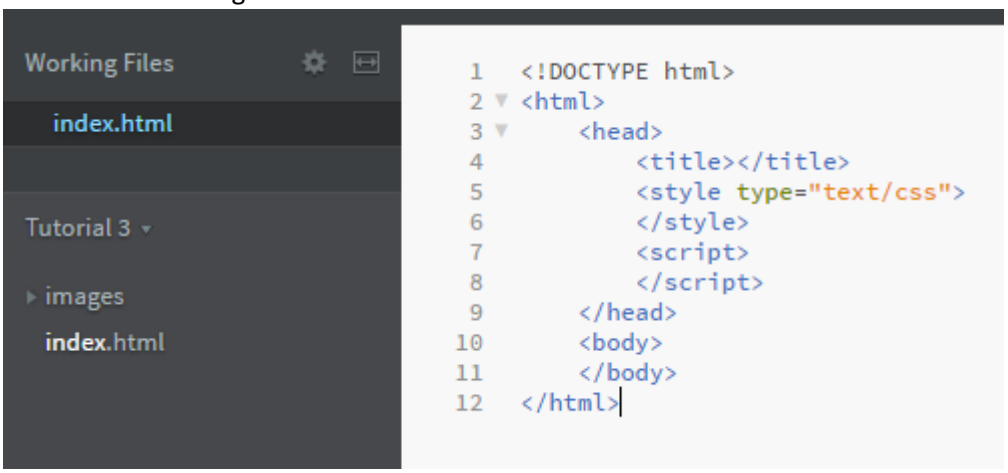
Create a website showcasing the following techniques

- Create a website with a canvas, examine JavaScript technique for controlling elements on the canvas.

Website

- Create a folder on the desktop called tutorial 3
  - 

- Open Brackets
  - 

- Create the following structure and index.html file
  - 

Index Page

Open up index.html, we will be working on this to create the page. To start with we will create a canvas on the page.

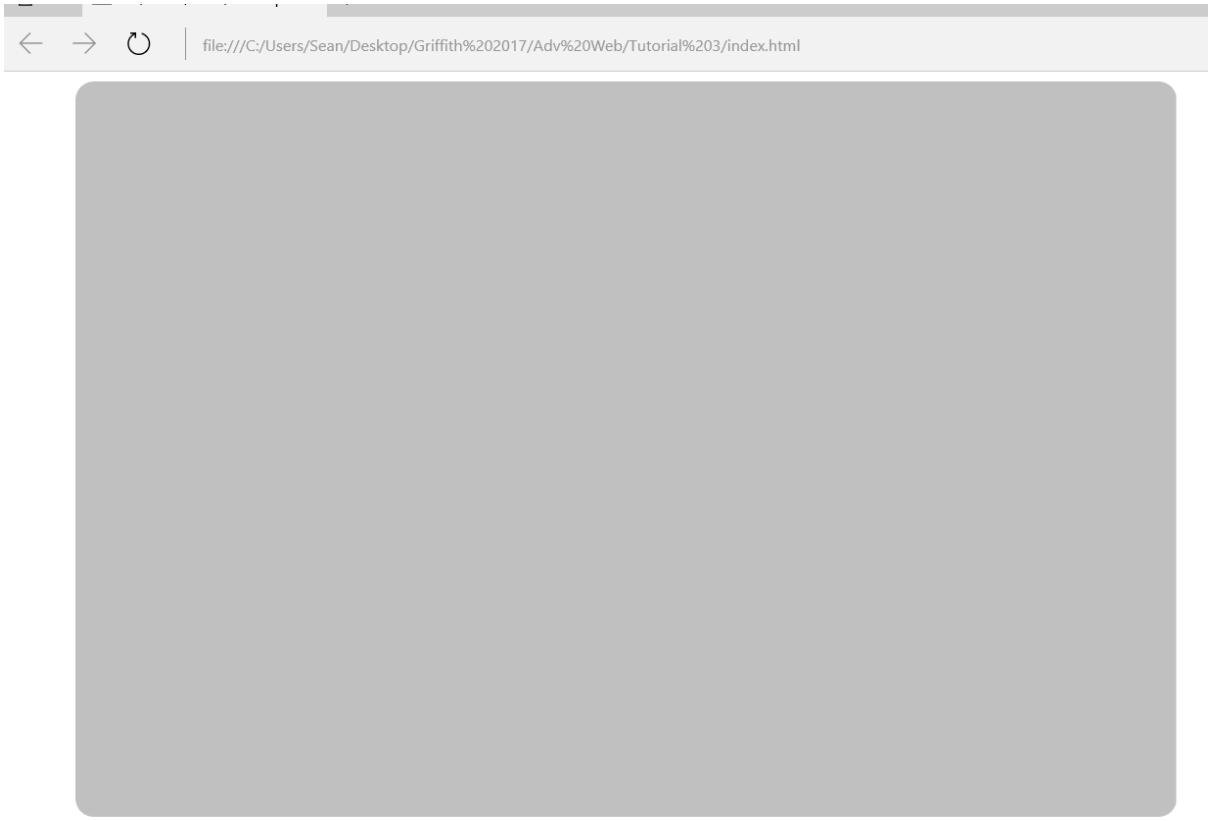
Div Code

```
<div id="wrapper">
    <canvas id="myCanvas" width="900" height="600"
    class="canvasColour"></canvas>
</div><!-- eo wrapper -->
```

Css Code

```
<style type="text/css">
    .canvasColour{background-color: silver; margin-left: 50px;
    border-radius: 15px;}
</style>
```

Result



Next we are going to draw some elements on the canvas. To do this, we will position a div box to contain all of the links that we are going to create:

Div Code

```
</div><!-- eo wrapper -->
<div id="linkBox">
    <p onClick="drawBox()">Draw a Box</p>
</div><!-- eo linkBox -->
body>
```

Css Code

```
#linkBox{margin-left: 50px; margin-top: 10px; font-
size:1.2em;}
#linkBox p{cursor: pointer; float: left; margin-left:
20px;}
```

Result

Draw a Box

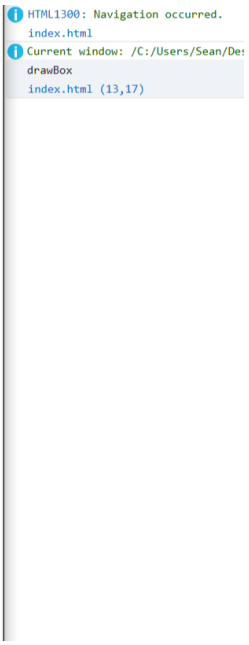
If you run the mouse over the ‘draw a box’ text, you will see that the cursor changes to make it look like a link. We have an onclick function applied to the <p> tag, so now, we need to create a javascript function to allow it to actually do something.

Javascript code

```
<script>
    function drawBox()
    {
        console.log("drawBox");
        var mCanvas = document.getElementById("myCanvas");
        var ctx = mCanvas.getContext('2d');
        ctx.fillStyle = "blue";
        ctx.fillRect(50,50,150,200);
    }
</script>
```

Save and test, with the console open (edge – F12/Developer tools, Chrome ctrl+shift+j, Firefox ctrl+shift +k) you should see the following when you click on the text, draw a box. You will see the console open on the right hand side of the browser in these documents.

Draw a Box



So this is programmatically drawing the box, now we are going to change the function to be a bit more dynamic in what it does.

So, we are going to add in X and Y positioning, colour and size. To do this we need to pass the information to the function, this is done via parameters that we can set in the div code. So, we need to make the following changes

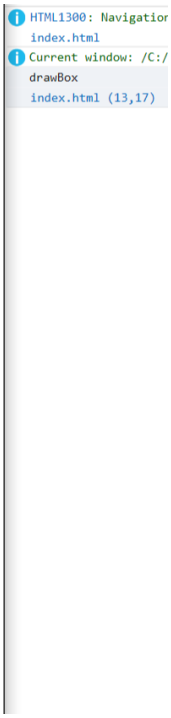
Div code

```
<div id="linkBox">
    <p onClick="drawBox(50,50,'red',20)">Draw a Box</p>
</div><!-- eo linkBox -->
```

Javascript

```
function drawBox(x,y,boxColour,size)
{
    console.log("drawBox");
    var mCanvas = document.getElementById("myCanvas");
    var ctx = mCanvas.getContext('2d');
    ctx.fillStyle = boxColour;
    ctx.fillRect(x,y,size,size);
}
```

Result



Draw a Box

Next we will draw a line, so we need to add a new piece of code to the linkBox area and to the JavaScript area.

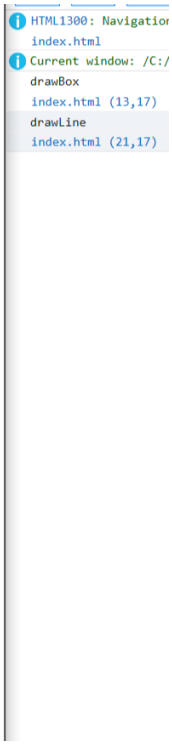
Div Code

```
<div id="linkBox">
    <p onClick="drawBox(50,50,'red',20)">Draw a Box</p>
    <p onClick="drawLine(40,80,80,80)">Draw a Line</p>
</div><!-- eo linkBox -->
```

Javascript

```
function drawLine(x1,y1,x2,y2)
{
    console.log("drawLine");
    var mCanvas = document.getElementById("myCanvas");
    var ctx = mCanvas.getContext('2d');
    ctx.beginPath();
    ctx.moveTo(x1,y1);
    ctx.lineTo(x2,y2);
    ctx.closePath();
    ctx.stroke();
}
```

Result (Click both drawbox and drawline)



Draw a Box Draw a Line

Next we'll draw a circle.

Div code

```
  <p onClick="drawLine(40,80,80,80)">Draw a Line</p>
  <p onClick="drawCircle(20,100,40)">Draw a Circle</p>
div><!-- eo linkBox -->
```

JavaScript

```
function drawCircle(radius,x,y)
{
    console.log("drawCircle");
    var mCanvas = document.getElementById("myCanvas");
    var ctx = mCanvas.getContext('2d');
    ctx.beginPath();
    ctx.arc(x,y,radius, Math.PI*2,0,true);
    ctx.closePath();
    ctx.stroke();
}
```

Result

Next, we'll put text on the canvas

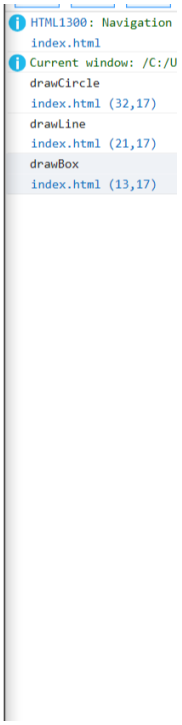
Div Code

```
<p onCLick="drawText('This is text',20,100)">Write
text</p>
div><!-- eo linkBox -->
```

JavaScript

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

Result



Draw a Box Draw a Line Draw a Circle Write text

Putting it all together, this is where we re-use the functions and show that by just changing the parameters feed into the function, how these elements of code can be dynamic.

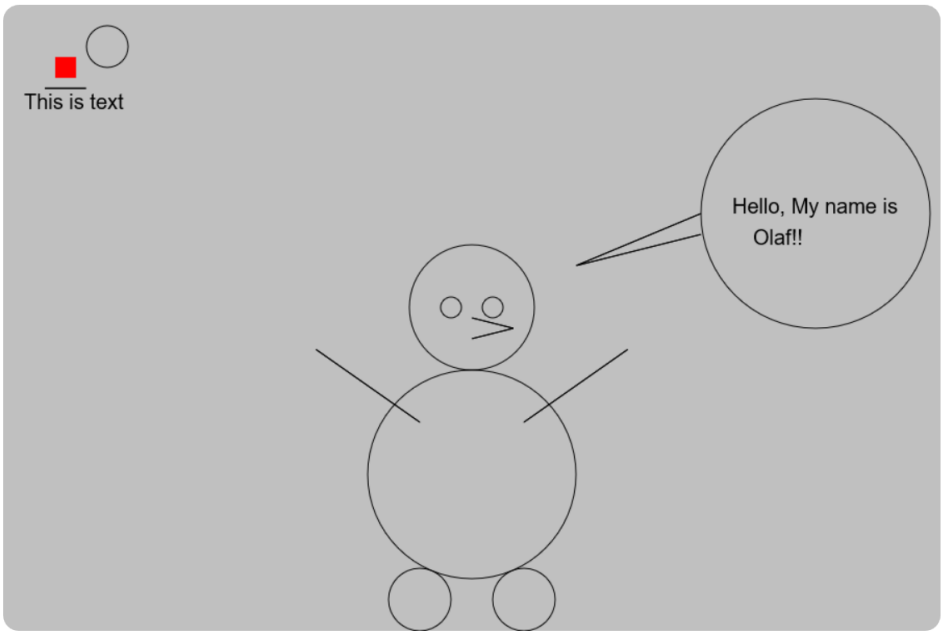
Div Code

```
  <p onClick="drawSnowman()">Draw a Snowman</p>
/div><!-- eo linkBox -->
```

JavaScript

```
function drawSnowman()
{
    drawCircle(100,450,450);
    drawCircle(60,450,290);
    drawCircle(10,430,290);
    drawCircle(10,470,290);
    drawLine(450,300,490,310);
    drawLine(450,320,490,310);
    drawLine(400,400,300,330);
    drawLine(500,400,600,330);
    drawCircle(30,400,570);
    drawCircle(30,500,570);
    drawCircle(110,780,200);
    drawText("Hello, My name is",700,200);
    drawText("Olaf!!",720,230);
    drawLine(670,200,550,250);
    drawLine(670,220,550,250);
}
```

Result



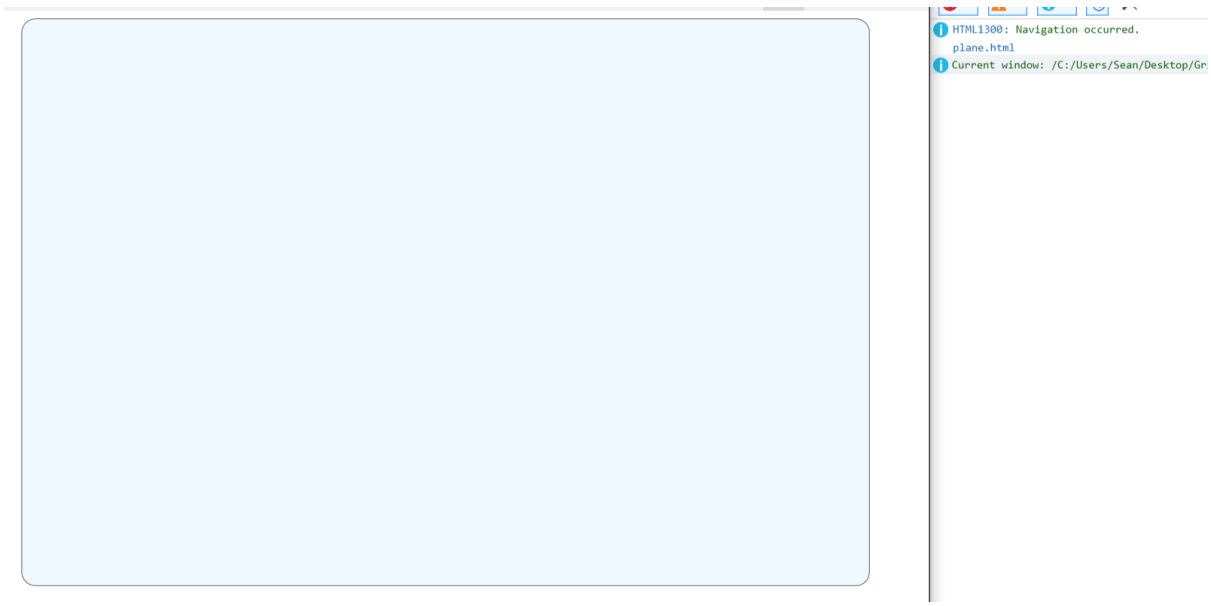
Draw a Box Draw a Line Draw a Circle Write text Draw a Snowman

Now, we start a new page that deals with drawing images onto the canvas.

Page 2

Start with a clean page called plane.html. Download the f18 image from L@G and store in the images folder. It should look like the following:

Result



Now we will create a function to load the image of the f18

First off we will modify the canvas to handle a click event

Then write a function to deal with the click event

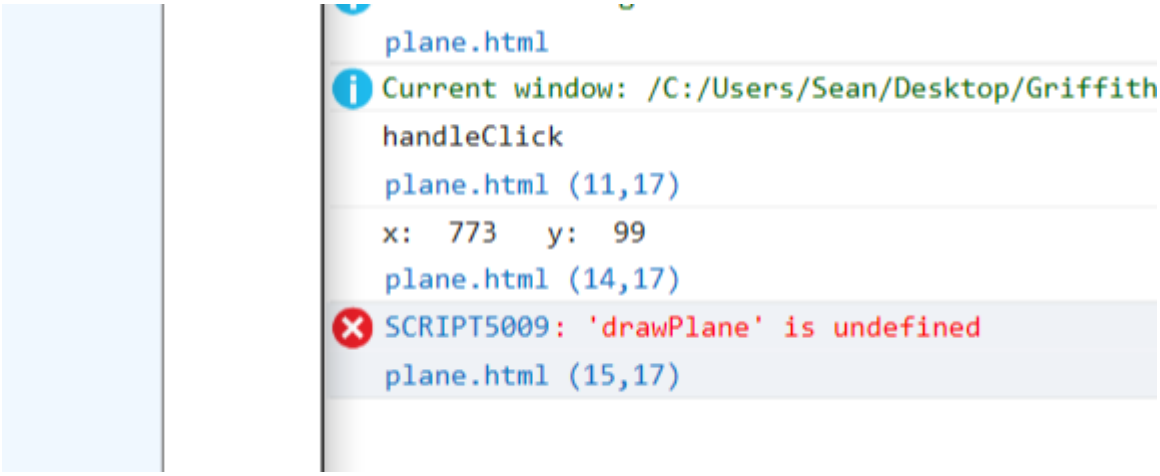
```
function handleClick(event)
{
    console.log("handleClick");
    var clickX = event.clientX - 50; //50 is the margin
    var clickY = event.clientY;
    console.log('x: ',clickX,'  y: ',clickY);
}
```

Test this with the console open, you should see differing numbers appear when you click on the canvas area. For example:

Now, we add the drawPlane function to the handleClick function, so when we click, we draw a plane.

```
function handleClick(event)
{
    console.log("handleClick");
    var clickX = event.clientX - 50; //50 is the margin
    var clickY = event.clientY;
    console.log('x: ',clickX,'  y: ',clickY);
    drawPlane(clickX,clickY);
}
```

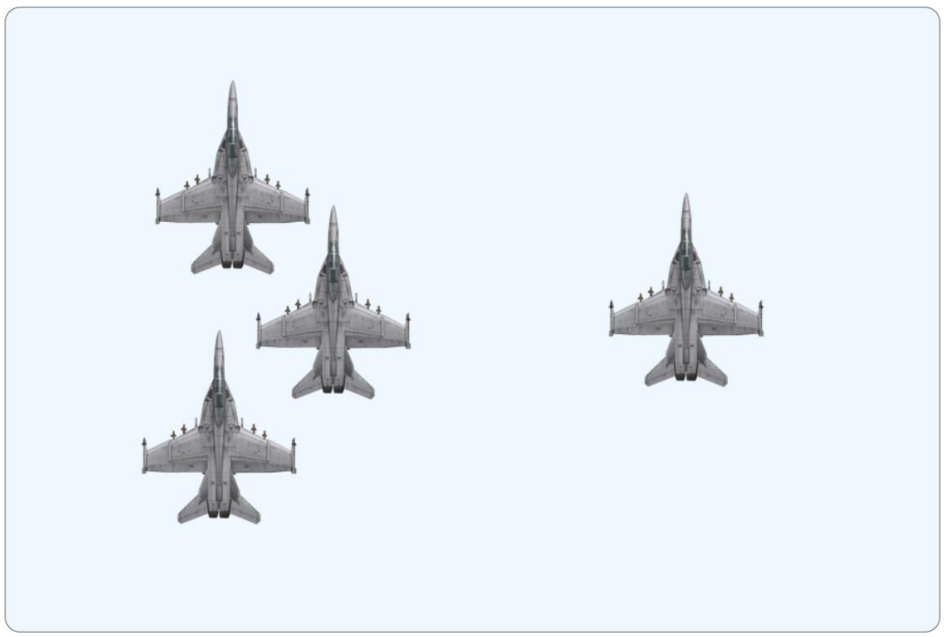
If you save and test it now, you will see the following error:



This is because we haven't written the drawPlane function yet. So write the following JavaScript function; Note that the click event x and y numbers are passed through to the drawPlane function:

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

Test with multiple clicks on the canvas

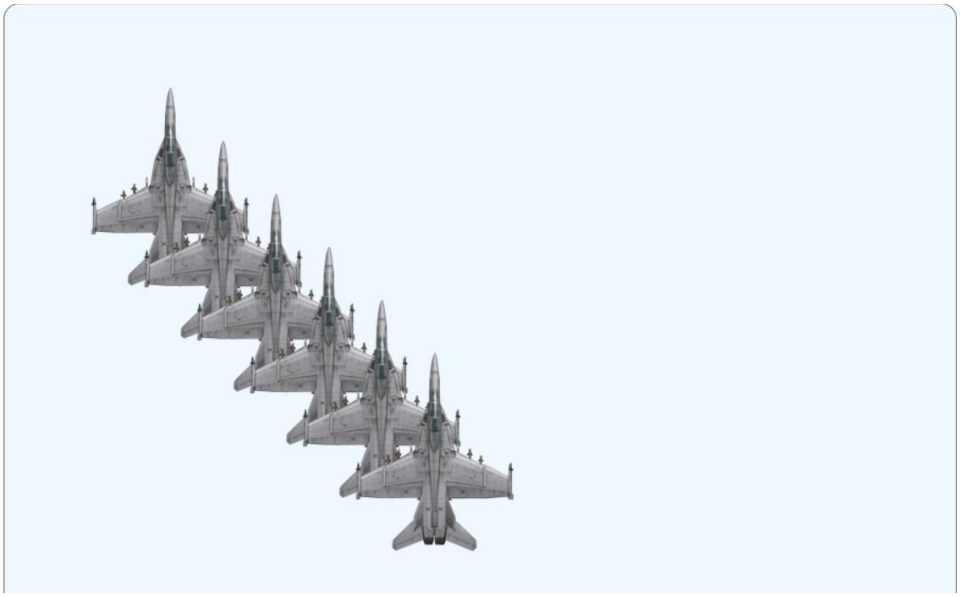


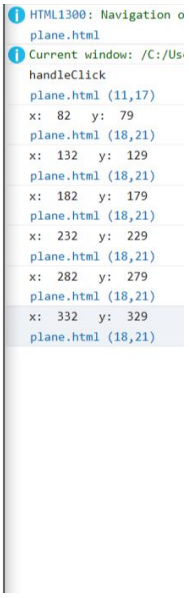
Loops

Let's look at a for loop in action. A for loop is a method of repeating a set of commands for a certain amount of times. In this case, we'll draw the plane multiple times

```
function handleClick(event)
{
    console.log("handleClick");
    var clickX = event.clientX - 50; //50 is the margin
    var clickY = event.clientY;
    for (x=0;x<=5;x++)
    {
        clickX = clickX + 50;
        clickY = clickY + 50;
        console.log('x: ',clickX,'  y: ',clickY);
        drawPlane(clickX,clickY);
    }
}
```

Test this

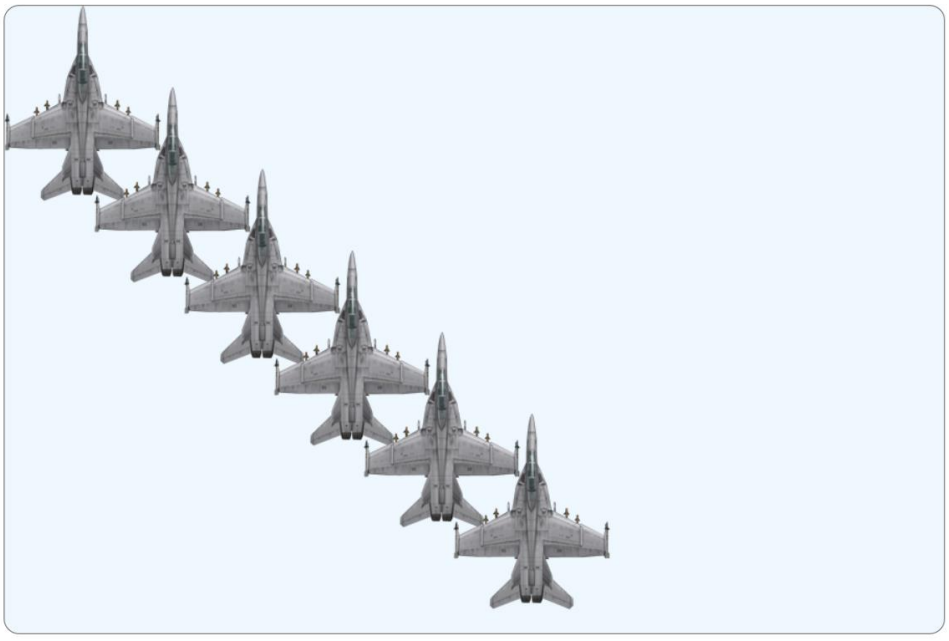




Next we'll look at the while loop by re-writing the handleclick function

```
function handleClick(event)
{
    console.log("handleClick");
    var clickX = event.clientX - 50; //50 is the margin
    var clickY = event.clientY;
    var max = 5;
    var start = 0;
    while (start<= max)
    {
        console.log('x: ',clickX,'  y: ',clickY);
        drawPlane(clickX*start,clickY*start);
        start++;
        console.log(start);
    }
}
```

Test



Next, we'll add a reset to the canvas to clean up multiple drawings, and remove the loops.

So handleClick becomes:

```
function handleClick(event)
{
    console.log("handleClick");
    var clickX = event.clientX - 50; //50 is the margin
    var clickY = event.clientY;
    console.log('x: ',clickX,'  y: ',clickY);
    drawPlane(clickX,clickY);
}
```

A new function called resetCanvas needs to be written and it looks like:

```
function resetCanvas()
{
    console.log("resetCanvas");
    var canvas = document.getElementById("myCanvas");
    context = canvas.getContext('2d');
    console.log("Canvas width before: ",canvas.width);
    context.clearRect(0,0,canvas.width,canvas.height);
    console.log("Canvas width after: ",canvas.width);
}
```

Then, the resetCanvas has to be called by the drawPlane function, so we modify it like this:

```
function drawPlane(x,y)
{
    resetCanvas();
    var mCanvas = document.getElementById("myCanvas");
    var ctx = mCanvas.getContext('2d');
    var plane = new Image();
    plane.src = "images/f18.png";
    plane.onload = function()
    {
        ctx.drawImage(plane,x,y);
    }
}
```

Now test, when you are doing this, you will see that the plane is only drawn once per click.

Backup

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