Tutorial 6

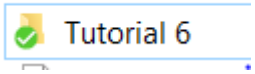
Editor – Brackets

Goals

Create a website showcasing the following techniques

- Animated backgrounds
- Animated game elements

Website

- Create a folder on the desktop called tutorial 6
  - 

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

Download the resources: https://1drv.ms/u/s!AskBHXkgcX44pe5brlf_ln5rA9H92g

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.

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

Body Code

```
<body>
    <canvas id="myCanvas" width="900" height="600" class="canvasColour">
    </canvas>
    <p onclick="nav()">Click to restart</p>
</body>
```

Style

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

Result

Click to restart

Next we add want to get the background up and running, this background will be moving left to right instead of up and down as in previous tutorials.

So, to start with we'll get a static background and then start adding animation; in addition a reload for the page, ie function nav().

Notice that once the code has all been loaded, the image is automatically loaded once the page is run. This is done by the addEventListener and function init().

Global Var

```
var canvasObject = document.getElementById("myCanvas");
var ctx = canvasObject.getContext('2d');
var gameWidth = canvasObject.width;
var gameHeight = canvasObject.height;
var bgImage = new Image();
bgImage.src = "images/background.jpg";
bgImage.addEventListener('load',init,false);
var fps =120;
var drawX = 0;
var drawY = 0;
```

Function init

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

Function gameLoop

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

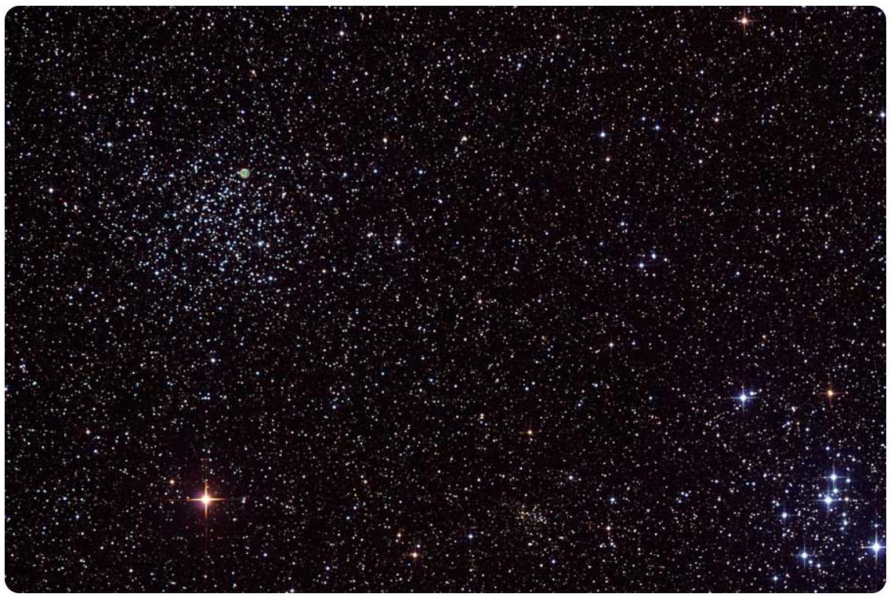
Function nav

```
function nav()
{
    window.document.location.href = "index.html";
}
```

Function drawbackground

```
function drawBackground()
{
      ctx.drawImage(bgImage,drawX,drawY);
}
```

Result



Click to restart

Now that the background is loaded, it’s time to add animation.

Global Javascript

```
var drawX = 0;
var drawY = 0;
window.requestAnimFrame = (function(){
    return  window.requestAnimationFrame       ||
  window.webkitRequestAnimationFrame ||
  window.mozRequestAnimationFrame    ||
  window.msRequestAnimationFrame     ||
  window.oRequestAnimationFrame      ||
  function( callback ){
    window.setTimeout(callback, 1000 / fps);
  };
})();
```

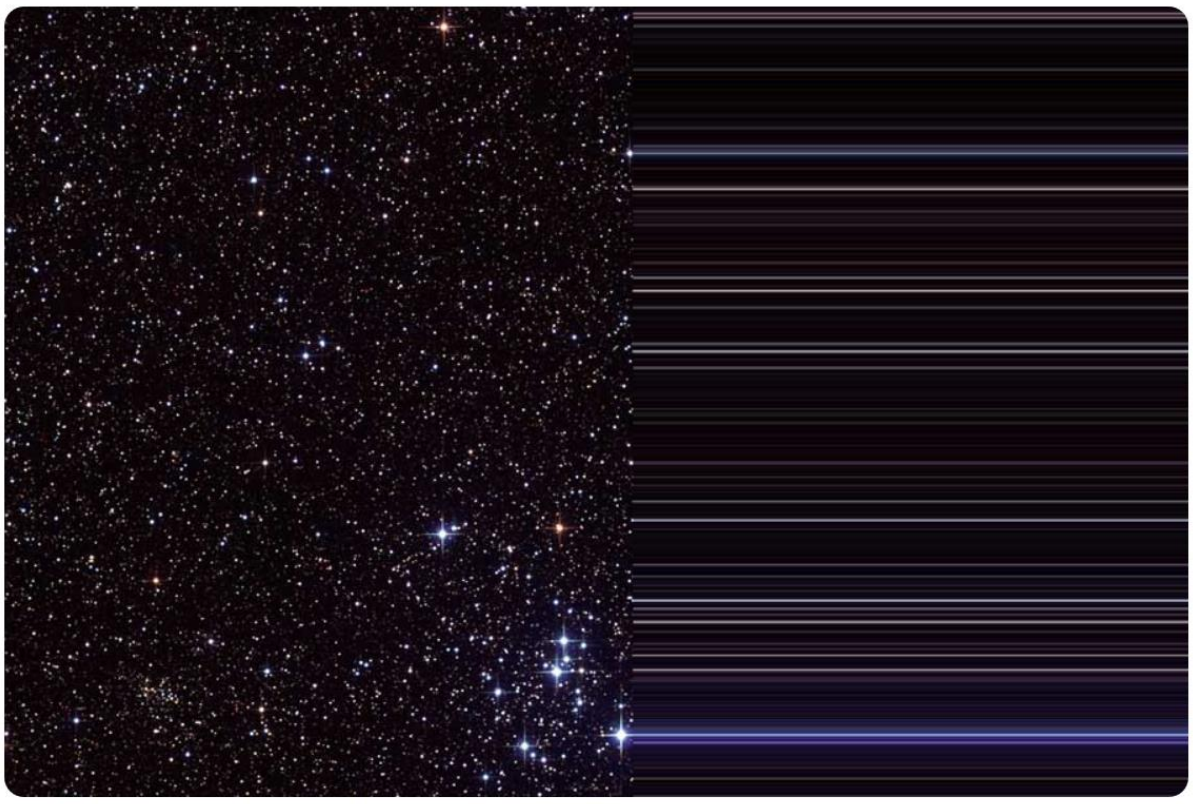
Function gameLoop

```
function gameLoop()
{
    setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            drawBackground();
        },1000/fps);
}
```

Function drawBackground

```
function drawBackground()
{
  ctx.drawImage(bgImage,drawX,drawY);
  drawX--;
}
```

Result



Click to restart

As before, we have a moving image which is leaving a streak path as the canvas is not being cleaned up after the image has been drawn. This can be fixed by creating a clearScren function and applying it to the gameLoop()

Function clearScreen

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

Function GameLoop

```
function gameLoop()
{
    setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            clearScreen();
            drawBackground();
        },1000/fps);
}
```

Result

Click to restart

Now that we have the screen being drawn cleanly, it’s time to loop the background image.

Global Var

```
var drawX = 0;
var drawY = 0;
var drawX_2 = gameWidth;
window.requestAnimFrame = (fu
```

Function drawBackground

```
function drawBackground()
  {
    if (drawX_2 <= 0)
    {
        drawX = 0;
        drawX_2 = gameWidth;
    }
      ctx.drawImage(bgImage,drawX,drawY);
      ctx.drawImage(bgImage,drawX_2,drawY);
    drawX--;
    drawX_2--;
  }
```

Result

Click to restart

Next we need to draw the player on the screen

Global Var

```
//player
var player = new Image();
player.src = "images/x-wing.png";
var posX = 20;
var posY = 400;
```

Function drawPlayer

```
function drawPlayer()
{
    ctx.drawImage(player,posX,posY);
}
```

Function gameLoop

```
function gameLoop()
{
    setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            clearScreen();
            drawBackground();
            drawPlayer();
        },1000/fps);
}
```

Result

Click to restart

Now, we will get the player ship to move, instead of using the mouse to move, we will implement keyboard commands, the commands used will be the WASD keyboard layout with the space bar, so WASD for movement and Space to shoot.

Also implemented is a global variable for determining if the x-wing shoots and the speed of the x-wing.

Global Javascript

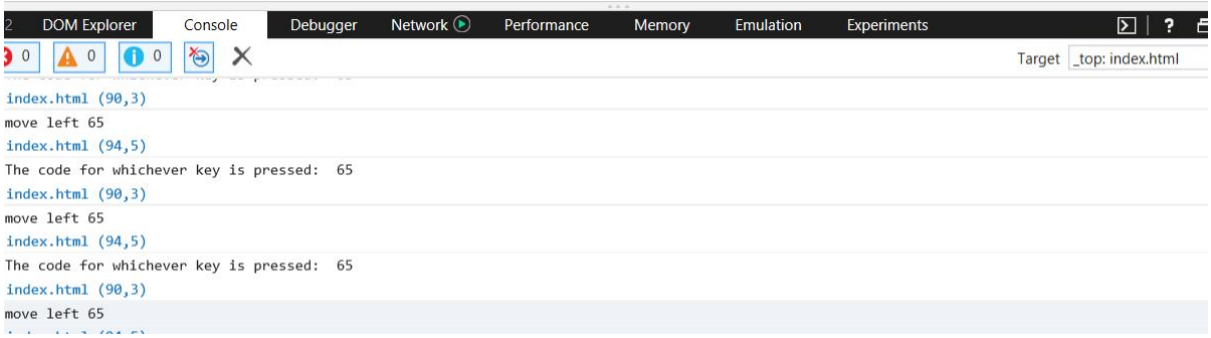
```
//player
var player = new Image();
player.src = "images/x-wing.png";
var posX = 20;
var posY = 400;
var xwingSpeed = 5;
var shoot = false;
//check if the keys are pushed
document.onkeydown = KeyCheck;
```

Function KeyCheck

```
function KeyCheck(e)
{
    var KeyID = (window.event) ? event.keyCode : e.keyCode;
    console.log("The code for whichever key is pressed: ",KeyID);
    switch(KeyID)
    {
        case 65: //key a Left
            console.log("move left",KeyID);
            posX = posX - xwingSpeed;
        break;
        case 87: // w Up
            console.log("move Up",KeyID);
            posY = posY - xwingSpeed;
        break;
        case 68: //key d Right
            console.log("move Right",KeyID);
            posX = posX + xwingSpeed;
        break;
        case 83: // key s Down
            console.log("move Down",KeyID);
            posY = posY + xwingSpeed;
        break;
        case 32: // key spacebar
            console.log("This is where we shoot",KeyID);
            shoot = true;
            gb_posX = posX+player.width - 10;;
            gb_posX2 = gb_posX;
            gb_posY = posY;
            gb_posY2 = posY + player.height;
          //  console.log(gb_posY,posY,gb_posY2);
        break;
    }
}
```

Result

Click to restart



Now that we have a player who can shoot, we need to give him something to fire, so we will add the laser bolts that will shoot off once the space bar has been hit.

Global Vars

```
//Player bolt
var greenBolt = new Image();
greenBolt.src = "images/greenBolt.png";
var gb_posX = 0;
var gb_posY = 0;
var gb_posX2 = 0;
var gb_posY2 = 0;
var shoot = false;
var boltSpeed = 7;
```

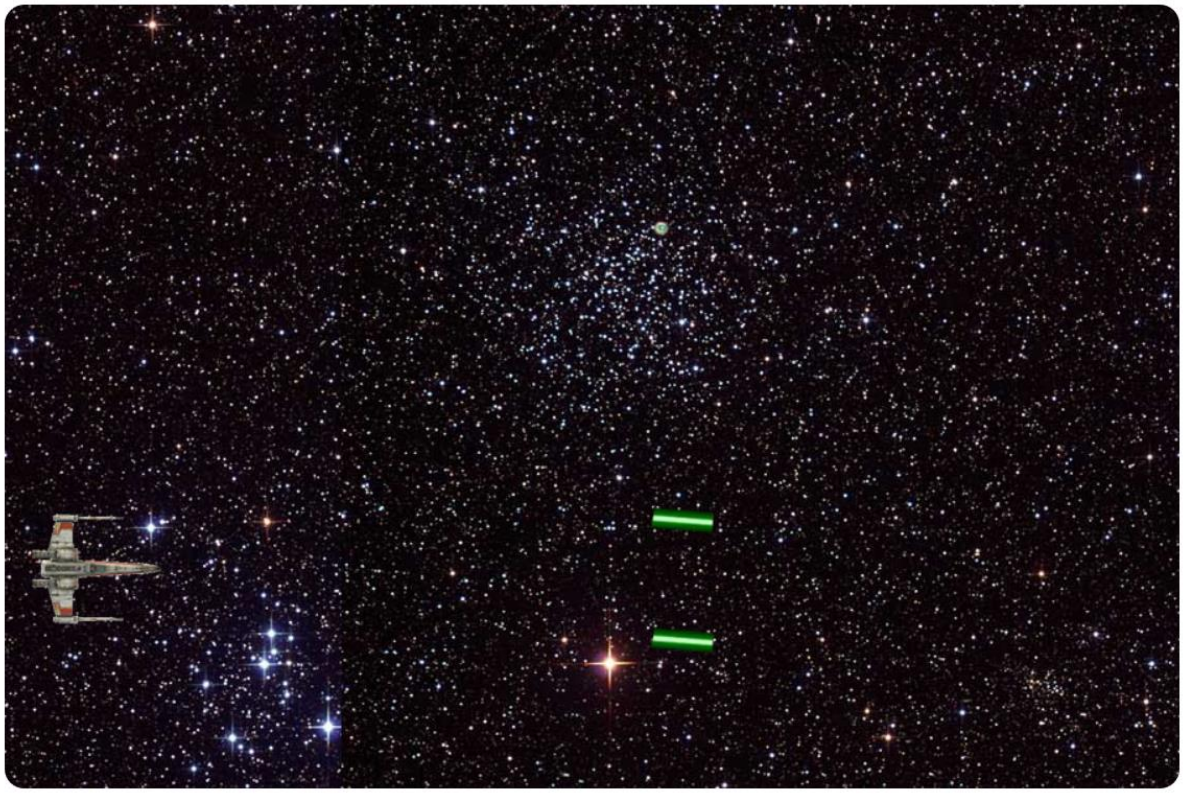
Function drawGreenBolt

```
function drawGreenBolt()
{
    if(shoot == true)
    {
        ctx.drawImage(greenBolt,gb_posX,gb_posY);
        ctx.drawImage(greenBolt,gb_posX2,gb_posY2);
        gb_posX = gb_posX + boltSpeed;
        gb_posX2 = gb_posX2 + boltSpeed;
    }
    if (gb_posX > gameWidth)
    {
        shoot = false;
    }
}
```

Function gameLoop

```
function gameLoop()
{
    setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            clearScreen();
            drawBackground();
            drawPlayer();
            if (shoot == true)
            {
                drawGreenBolt();
            }
        },1000/fps);
}
```

Result



Click to restart

Now we have to give our player something to shoot, so we will draw a tie fighter and have it chase down the x-wing.

Global Var

```
//Tie Fighter
var tie = new Image();
tie.src = "images/tie.png";
var tieSpeed = 6;
var en_posX = 1000;
var en_posY = 400;
```

Function drawTie

```
function drawTie()
{
    ctx.drawImage(tie,en_posX,en_posY);
}
```

Function moveTie

```
function moveTie()
{
    var x = en_posX;
    var y = en_posY;
    en_posX = x - tieSpeed;
    if (y <= posY)
        {y++;}
    else
        {y--;}
    en_posY = y;
    if (en_posX <= -100)
        {en_posX = gameWidth + 200;}
}
```

Function gameLoop

```
function gameLoop()
{
    setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            clearScreen();
            drawBackground();
            drawPlayer();
            drawTie();
            moveTie();
            if (shoot == true)
            {
                drawGreenBolt();
            }
        },1000/fps);
}
```

Result

Click to restart

Next we add collision detection and a visual to showcase that the ships collide, this is all building up to the part where we can program the laser bolts to blow up the tie fighter.

Global Vars

```
//collision
var exp = new Image();
exp.src = "images/explosion.png";
var explosion = false;
var hit = false;
```

Function drawExplosion

```
function drawExplosion(a)
{
    if (a == 'tie')
    {
        ctx.drawImage(exp,en_posX,en_posY);
    }
    if (a == 'player')
    {
        ctx.drawImage(exp,posX,posY);
    }
}
```

Function collision

```
function collision()
{
    var playerWidth = posX + player.width;
    var playerHeight = posY + player.height;
    var enemyHeight = en_posY + tie.height;
    if (playerWidth > en_posX)
    {
        console.log("Hit X");
        if(posY >= en_posY && playerHeight <= enemyHeight)
        {
            console.log("Hit Y");
            shoot=false;
            explosion = true;
            drawExplosion('player');
            drawExplosion('tie');
            hit=true;
        }
    }

}
```

Function gameLoop

```
function gameLoop()
{
    setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            clearScreen();
            drawBackground();
            drawPlayer();
            drawTie();
            moveTie();
            collision();
            if (shoot == true)
            {
                drawGreenBolt();
            }
        },1000/fps);
}
```

Result

Click to restart

Now we will add the code to allow for the laser bolt to hit the tie fighter. This will require determining bolt collision, the tie exploding and resetting of the tie after it has exploded.

Function drawGreenBolt

```
function drawGreenBolt()
{
    boltCollision();
    if(shoot == true)
    {
        ctx.drawImage(greenBolt,gb_posX,gb_posY);
        ctx.drawImage(greenBolt,gb_posX2,gb_posY2);
        gb_posX = gb_posX + boltSpeed;
        gb_posX2 = gb_posX2 + boltSpeed;
    }
    if (gb_posX > gameWidth)
    {
        shoot = false;
    }
}
```

Function boltCollision

```
function boltCollision()
{
  var gb_x = gb_posX + greenBolt.width;
  var gb_x2 = gb_posX2 + greenBolt.width;
  var tieH = en_posY + tie.height;
  if (gb_x > en_posX)
  {
      //console.log("Hit X");
      if(gb_posY > en_posY && gb_posY < tieH)
      {
          console.log("Hit Y");
          shoot=false;
          explosion = true;
      }
  }
  if (gb_x2 > en_posX)
  {
      //console.log("Hit X");
      if(gb_posY2 > en_posY && gb_posY2 < tieH)
      {
          console.log("Hit Y");
          shoot=false;
          explosion = true;
      }
  }
}
```

Function gameLoop

```
function gameLoop()
{
    setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            clearScreen();
            drawBackground();
            drawPlayer();
            if (explosion == false)
            {
                drawTie();
                moveTie();
            }
            else
            {
                drawExplosion('tie');
                resetTie();
            }
            collision();
            if (shoot == true)
            {
                drawGreenBolt();
            }
            if (shoot == true)
            {
                drawGreenBolt();
            }
        },1000/fps);
}
```

Function resetTie

```
function resetTie()
{
    en_posX = 1000;
    en_posY = 400;
    explosion = false;
}
```

Result

Click to restart

Two final elements to be added to the game, a score and a way to have the game stop when the player gets hit by the tie fighter.

Global Var

```
//Score
var score = 0;
```

Function updateScore

```
function updateScore()
{
    ctx.font = "20px Arial";
    ctx.fillStyle = "white";
    ctx.fillText("Score: "+score,800,20);

}
```

Function drawExplosion

```
function drawExplosion(a)
{
    if (a == 'tie')
    {
        ctx.drawImage(exp,en_posX,en_posY);
        score++;
    }
    if (a == 'player')
    {
        ctx.drawImage(exp,posX,posY);
    }
}
```

Function gameLoop

```
function gameLoop()
{
    if (hit == false)
    {
        setTimeout(function()
        {
            requestAnimFrame(gameLoop);
            clearScreen();
            drawBackground();
            drawPlayer();
            if (explosion == false)
            {
                drawTie();
                moveTie();
            }
            else
            {
                drawExplosion('tie');
                resetTie();
            }
            collision();
            if (shoot == true)
            {
                drawGreenBolt();
            }
            updateScore();
        },1000/fps);
     }
     else
     {
      console.log("You dead",hit);
     }

}
```

Result

Click to restart

Backup

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