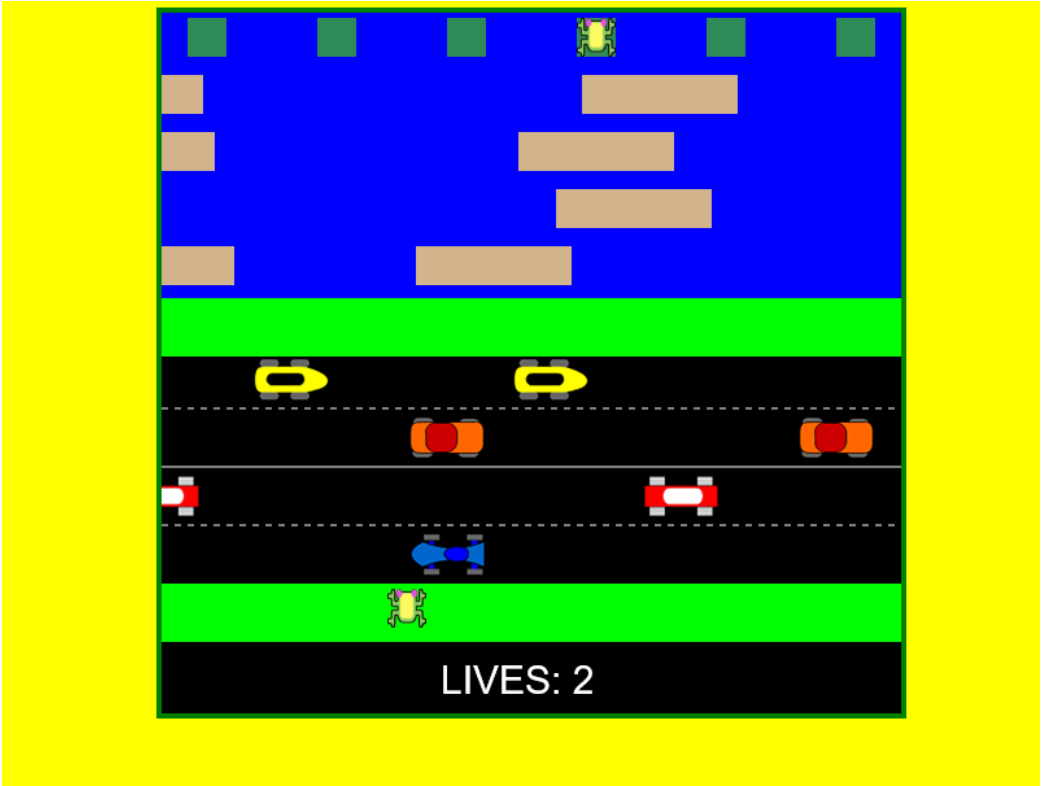
FROGGER TUTORIAL



Frogger Lesson 1

In this step-by-step tutorial we create a simple frogger type game written in JavaScript (without jQuery or any other JavaScript library) and rendered on HTML5 <canvas>.

You will learn the basics of using the <canvas> element to implement fundamental game mechanics like rendering and moving images, collision detection, control mechanisms, and winning and losing states.

To get the most out of this series of articles you should already have basic to intermediate JavaScript knowledge. After working through this tutorial you should be able to build your own simple Web games.

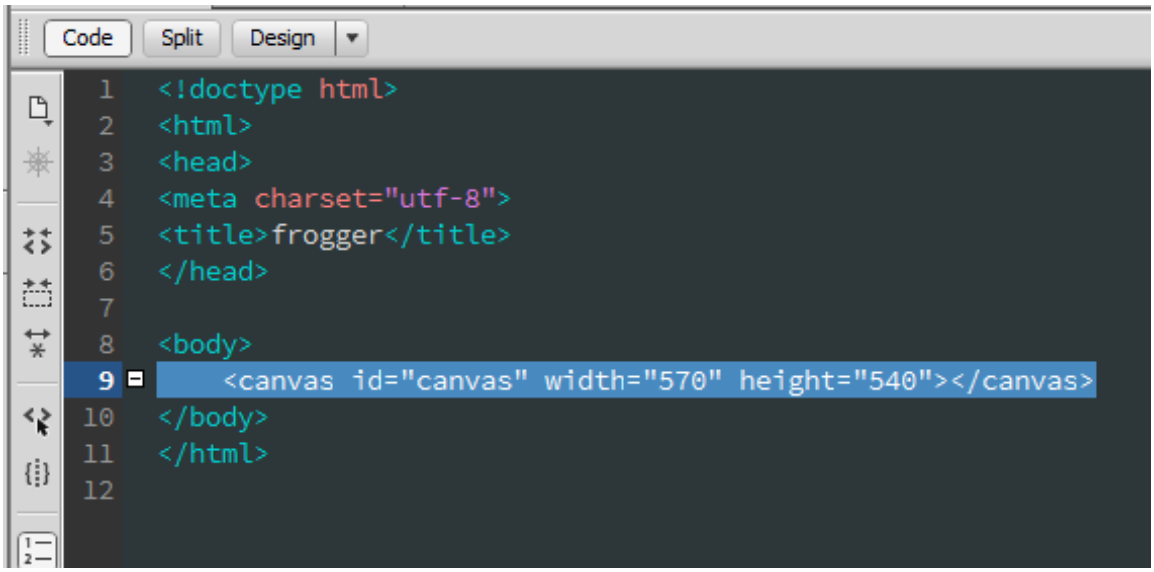
Starting with pure JavaScript is the best way to get a solid knowledge of web game development. After that, you can pick any framework you like and use it for your projects. Frameworks are just tools built with the JavaScript language; so even if you plan on working with them, it's good to learn about the language itself first to know what exactly is going on under the hood. Frameworks speed up development time and help take care of boring parts of the game, but if something is not working as expected, you can always try to debug that or just write your own solutions in pure JavaScript.

Create the Canvas

Before we can start writing the game's functionality, we need to create a basic structure to render the game inside. This can be done using HTML and the <canvas> element. (W3 SUMMARY OF CANVAS ATTRIBUTES)

The game's HTML

The HTML document structure is quite simple, as the game will be rendered entirely on the <canvas> element. Open Dreamweaver, create a new HTML document, save it as frogger.html in a sensible location, and add the following code to create a canvas element.



Next we will add a bit of styling to our page. Typically, you will want to link your html file to a css file that would contain all the css styles to be applied to your elements. Since we have very little css to deal with, I am going to include them in the head element of our html file. To easily center our canvas within the body element, set the body element's display style to flex and justify-content to center. Use whatever colors you like.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>frogger</title>
<style>
body{
    background: yellow;
    display: flex;
    justify-content: center;
}
canvas{
    background: black;
    border: 4px solid green;
}
</style>
</head>

<body>
<canvas id="canvas" width="570" height="540"></canvas>
</body>
</html>
```

At this point we can preview our html file in Goggle Chrome and see the styling we have applied.

Create a new JavaScript file, name it frogger.js and save in the same location as your html file.

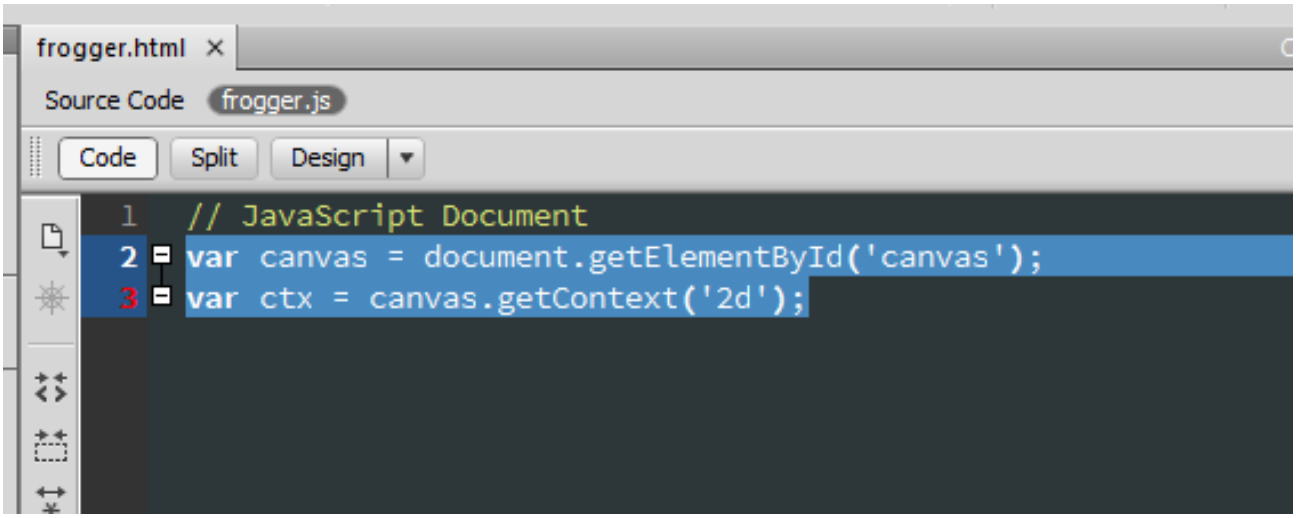
To actually be able to render graphics on the <canvas> element, first we have to grab a reference to it in JavaScript. Add the following in your js file. Here we're storing a reference to the <canvas> element to the canvas variable. Then we're creating the ctx variable to store the 2D rendering context — the actual tool we can use to paint on the Canvas.

```
// JavaScript Document
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');
```

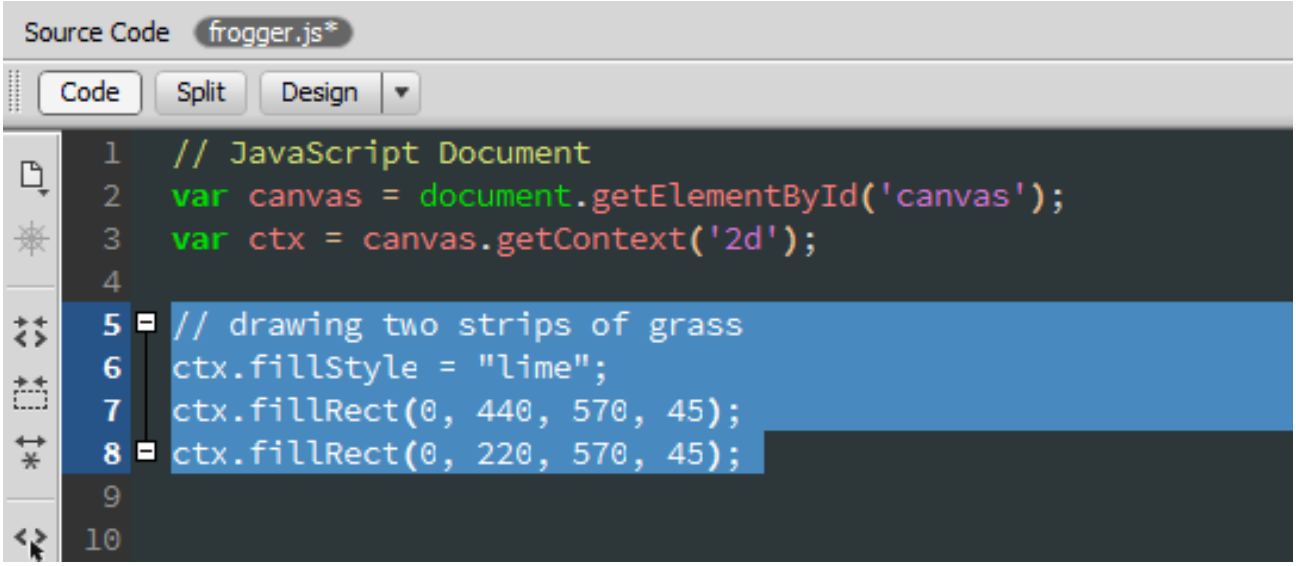
Go back to your html file and add a link to the javascript file as shown below.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>frogger</title>
<style>
body{
    background: yellow;
    display: flex;
    justify-content: center;
}
canvas{
    background: black;
    border: 4px solid green;
}
</style>
</head>

<body>
<canvas id="canvas" width="570" height="540"></canvas>
<script src="frogger.js"></script>
</body>
</html>
```

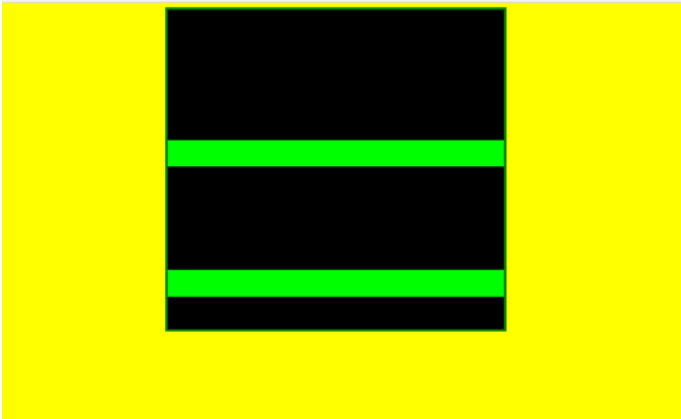
We will add our code in our javascript file, but we will be previewing our html file as we work. The easiest way to do this in Dreamweaver is to close the js tab and access the js file as shown below. Then we can preview in browser simply by pressing our F12 key.



We will start by drawing two rectangles to represent strips of grass on our play screen. We are defining the rectangles using the fillRect method. The first two values specify the x and y coordinates of the top left corner of the rectangle on the canvas, while the second two specify the width and height of the rectangle.



Preview in browser should look like the screenshot below now.



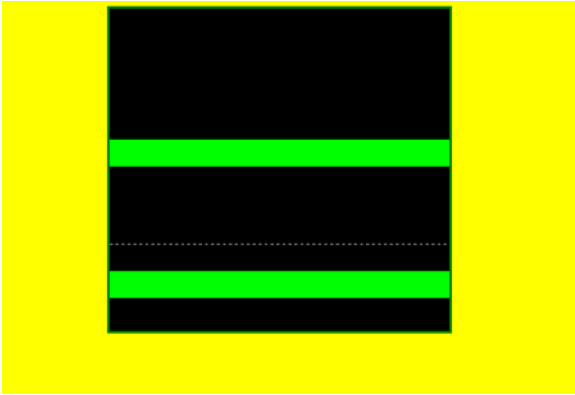
Next we will add a dashed horizontal line to represent a lane boundary for our cars. The line starts at x = 0 and y = 395 and ends at x = 570 and y = 395.

Source Code frogger.js

Code Split Design

```
// JavaScript Document
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');

// drawing two strips of grass
ctx.fillStyle = "lime";
ctx.fillRect(0, 440, 570, 45);
ctx.fillRect(0, 220, 570, 45);

ctx.beginPath();
ctx.moveTo(0,395);
ctx.lineTo (570,395);
ctx.strokeStyle = "white";
ctx.setLineDash([5]);
ctx.strokeWidth = 2;
ctx.stroke();
```

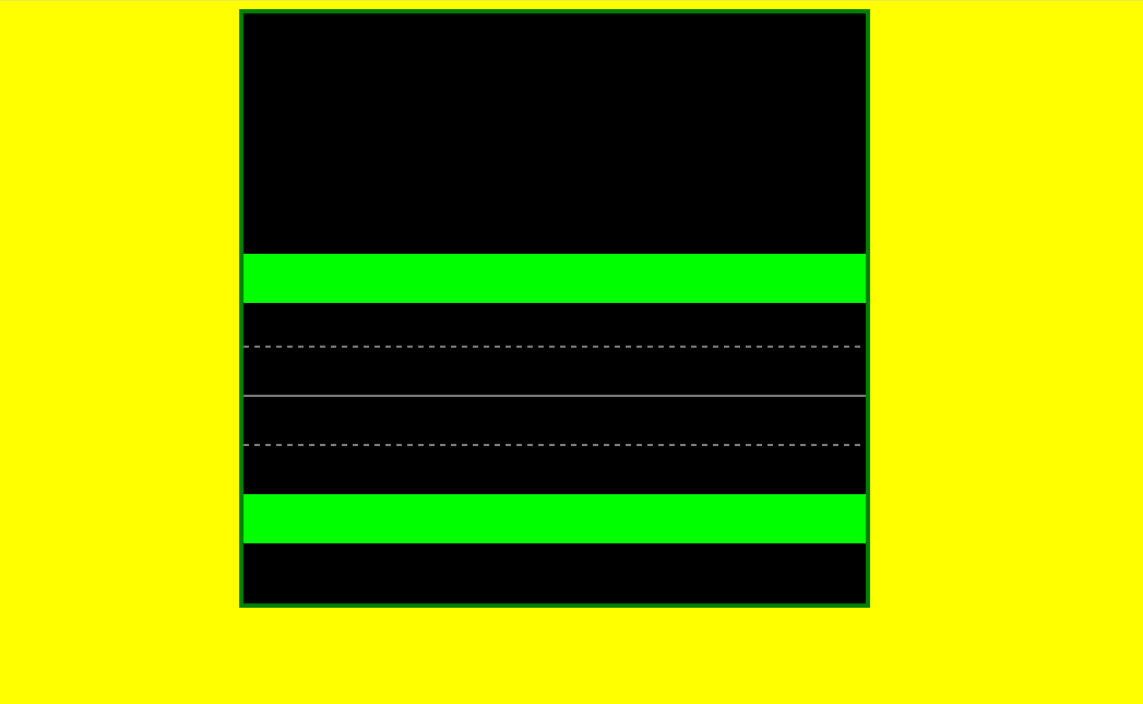
Preview in browser to make sure you line appears like the screenshot below.



Finish the street markings by adding two more lines as shown below.

```
ctx.fillRect(0, 220, 570, 45);

ctx.beginPath();
ctx.moveTo(0,395);
ctx.lineTo (570,395);
ctx.strokeStyle = "white";
ctx.setLineDash([5]);
ctx.strokeWidth = 2;
ctx.stroke();

ctx.beginPath();
ctx.moveTo(0,350);
ctx.lineTo (570,350);
ctx.strokeStyle = "white";
ctx.setLineDash([0]);
ctx.strokeWidth = 4;
ctx.stroke();

ctx.beginPath();
ctx.moveTo(0,305);
ctx.lineTo (570,305);
ctx.strokeStyle = "white";
ctx.setLineDash([5]);
ctx.strokeWidth = 2;
ctx.stroke();
```

Preview in browser. Save your files for the next lesson.

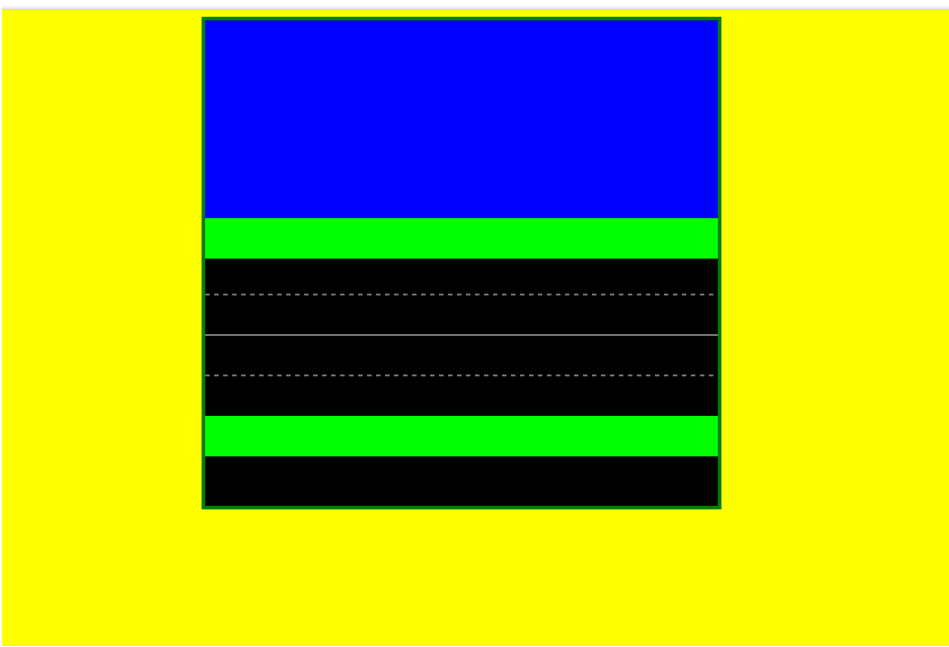


Frogger Lesson 2

We can finish our background by adding javascript code to create a rectangle to represent water. Note that line 34 is a non-executing comment for the purpose of keeping the code somewhat organized and understandable.

```
26  ctx.beginPath();
27  ctx.moveTo(0,305);
28  ctx.lineTo (570,305);
29  ctx.strokeStyle = "white";
30  ctx.setLineDash([5]);
31  ctx.strokeWidth = 2;
32  ctx.stroke();
33
34  //drawing water
35  ctx.fillStyle = "blue";
36  ctx.fillRect(0, 0, 570, 220);
37
```

Preview in Google Chrome. If your screen doesn't look like the one below, work with a classmate to debug your code before proceeding.



Defining a drawing loop for animation.

To keep constantly updating the canvas drawing on each frame, we need to define a drawing function that will run over and over again, with a different set of variable values each time to change sprite positions, etc. We can run a function over and over again using the JavaScript timing function requestAnimationFrame().

Let's start by enclosing the JavaScript we used to draw our background in a function drawBackground(). Add lines 5 and 38 shown below to define the function.

Source Code frogger.js*

Code Split Live file:///C|/Users/John/Desk

```
1  // JavaScript Document
2  var canvas = document.getElementById('canvas');
3  var ctx = canvas.getContext('2d');
4
5  function drawBackground(){
6  // drawing two strips of grass
7  ctx.fillStyle = "lime";
8  ctx.fillRect(0, 440, 570, 45);
9  ctx.fillRect(0, 220, 570, 45);
10
11 ctx.beginPath();
12 ctx.moveTo(0,395);
13 ctx.lineTo (570,395);
14 ctx.strokeStyle = "white";
15 ctx.setLineDash([5]);
16 ctx.strokeWidth = 2;
17 ctx.stroke();
18
19 ctx.beginPath();
20 ctx.moveTo(0,350);
21 ctx.lineTo (570,350);
22 ctx.strokeStyle = "white";
23 ctx.setLineDash([0]);
24 ctx.strokeWidth = 4;
25 ctx.stroke();
26
27 ctx.beginPath();
28 ctx.moveTo(0,305);
29 ctx.lineTo (570,305);
30 ctx.strokeStyle = "white";
31 ctx.setLineDash([5]);
32 ctx.strokeWidth = 2;
33 ctx.stroke();
34
35 //drawing water
36 ctx.fillStyle = "blue";
37 ctx.fillRect(0, 0, 570, 220);
38 }
```

You will notice if you preview in your browser, the background has disappeared because we are not calling for the execution of our function anywhere in our program. We now need to create a draw() function that we will use to call the drawBackground function. With the requestAnimationFrame loop, the draw() function will be executed every time your screen refreshes. Because most screens have a refresh rate of 60Hz, the fastest frame rate you should aim for is 60 frames per second. The loop will sync the framerate accordingly and render the shapes only when needed.

```
ctx.setLineDash([0]);
ctx.strokeWidth = 4;
ctx.stroke();

ctx.beginPath();
ctx.moveTo(0,305);
ctx.lineTo (570,305);
ctx.strokeStyle = "white";
ctx.setLineDash([5]);
ctx.strokeWidth = 2;
ctx.stroke();

//drawing water
ctx.fillStyle = "blue";
ctx.fillRect(0, 0, 570, 220);
}

function draw(){
    drawBackground();

    requestAnimationFrame(draw);
}
draw();
```

Now if you preview your file in Google Chrome, your background should show up as before. The only difference is that the background is now actually being redrawn nearly 60 times per second.

In this next step we will use the drawImage() method to draw an image to represent our frog. The drawImage() method draws an image, canvas, or video onto the canvas.The drawImage() method can also draw parts of an image, and/or increase/reduce the image size.

JavaScript Syntax

Position the image on the canvas:

JavaScript syntax:	*ctx*.drawImage(*img,x,y*);

Position the image on the canvas, and specify width and height of the image:

JavaScript syntax:	*ctx*.drawImage(*img,x,y,width,height*);

Clip the image and position the clipped part on the canvas:

JavaScript syntax:	*ctx*.drawImage(*img,sx,sy,swidth,sheight,x,y,width,height*);

Parameter Values

Parameter	**Description**
img	Specifies the image, canvas, or video element to use
sx	Optional. The x coordinate where to start clipping
sy	Optional. The y coordinate where to start clipping
swidth	Optional. The width of the clipped image
sheight	Optional. The height of the clipped image
x	The x coordinate where to place the image on the canvas
y	The y coordinate where to place the image on the canvas
width	Optional. The width of the image to use (stretch or reduce the image)
height	Optional. The height of the image to use (stretch or reduce the image)

Save the image below, or copy the frogger.png file from the Arrington folder and paste into the same location as your frogger.html and frogger.js files.

Create a new variable for frog with the image source specified as shown below.

Source Code frogger.js*

Code Split Live file:///C|/Users/John/Desktop/fro

```
// JavaScript Document
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');

var frog = new Image(); frog.src = "frogger.png";

function drawBackground(){
// drawing two strips of grass
ctx.fillStyle = "lime";
ctx.fillRect(0, 440, 570, 45);
ctx.fillRect(0, 220, 570, 45);

ctx.beginPath();
ctx.moveTo(0,395);
ctx.lineTo (570,395);
ctx.strokeStyle = "white";
ctx.setLineDash([5]);
ctx.strokeWidth = 2;
ctx.stroke();

ctx.beginPath();
ctx.moveTo(0,350);
ctx.lineTo (570,350);
ctx.strokeStyle = "white";
ctx.setLineDash([0]);
```

Create a drawFrog() function below the drawBackground() function and above the draw() function as shown.

```
35  ctx.stroke();
36
37  //drawing water
38  ctx.fillStyle = "blue";
39  ctx.fillRect(0, 0, 570, 220);
40  }
41
42  function drawFrog(){
43      ctx.drawImage(frog, sx, sy, swidth, sheight, x, y, width, height);
44  }
45
46  function draw(){
47      drawBackground();
48
49      requestAnimationFrame(draw);
```

After creating the drawFrog() function we will need to call its execution from within our animated draw function.

```
40  }
41
42  function drawFrog(){
43      ctx.drawImage(frog, sx, sy, swidth, sheight, x, y, width, height);
44  }
45
46  function draw(){
47      drawBackground();
48      drawFrog();
49
50      requestAnimationFrame(draw);
51  }
52  draw();
```

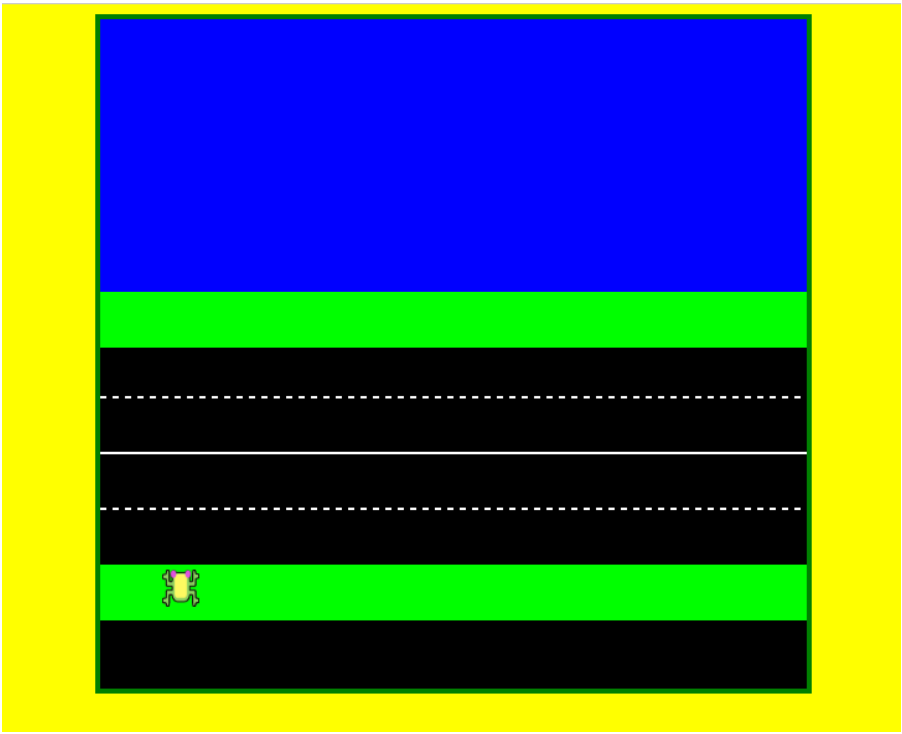
Finally, to actually draw the frog, we need to initialize all the variables for our drawImage method.

Source Code frogger.js

Code Split Live file:///C|/Users/John/Desktop/frogger.html

```javascript
// JavaScript Document
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');

var frog = new Image(); frog.src = "frogger.png";
var sx = 0;
var sy = 0;
var swidth = 40;
var sheight = 40;
var x = 50;
var y = 444;
var width = 30;
var height = 30;

function drawBackground(){
// drawing two strips of grass
ctx.fillStyle = "lime";
ctx.fillRect(0, 440, 570, 45);
```

Preview in browser.



Frogger Lesson 3

We can our frog just sit there…it is time to implement some keyboard controls. We will need:

- Four variables for storing information on whether the up, down, left or right control button is pressed.
- Two event listeners for keydown and keyup events — we want to run some code to handle the paddle movement when the buttons are pressed.
- Two functions handling the keydown and keyup events the code that will be run when the buttons are pressed.
- The ability to move the frog up, down, left and right

Pressed buttons can be defined and initialized with boolean variables, like so. Add these lines under the rest of your variables:

Source Code frogger.js*

Code Split Live file:///C|/Users/John/

```
// JavaScript Document
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');

var frog = new Image(); frog.src = "frogger.png";
var sx = 0;
var sy = 0;
var swidth = 40;
var sheight = 40;
var x = 50;
var y = 444;
var width = 30;
var height = 30;

var rightPressed = false;
var leftPressed = false;
var upPressed = false;
var downPressed = false;

function drawBackground(){
// drawing two strips of grass
ctx.fillStyle = "lime";
ctx.fillRect(0, 440, 570, 45);
```

The default values are false because at the beginning the control buttons are not pressed. To listen for key presses, we will set up two event listeners. Add the following lines below your variables.

```
var y = 444;
var width = 30;
var height = 30;

var rightPressed = false;
var leftPressed = false;
var upPressed = false;
var downPressed = false;

document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);

function drawBackground(){
// drawing two strips of grass
ctx.fillStyle = "lime";
```

When the keydown event is fired on any of the keys on your keyboard (when they are pressed), the keyDownHandler() function will be executed. The same pattern is true for the second listener: keyup events will fire the keyUpHandler() function (when the keys stop being pressed). Add these to your code now, below the addEventListener() lines:

```
var downPressed = false;

document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);

function keyDownHandler(e)
{
  if(e.keyCode == 39) {rightPressed = true;}
  if(e.keyCode == 37) {leftPressed = true;}
  if(e.keyCode == 38) {upPressed = true;}
  if(e.keyCode == 40) {downPressed = true;}
}

function keyUpHandler(e)
{
  if(e.keyCode == 39) {rightPressed = false;}
  if(e.keyCode == 37) {leftPressed = false;}
  if(e.keyCode == 38) {upPressed = false;}
  if(e.keyCode == 40) {downPressed = false;}
}

function drawBackground(){
// drawing two strips of grass
ctx.fillStyle = "lime";
```

When we press a key down, this information is stored in a variable. The relevant variable in each case is set to true. When the key is released, the variable is set back to false.

Both functions take an event as a parameter, represented by the e variable. From that you can get useful information: the keyCode holds the information about the key that was pressed. For example key code 37 is the left cursor key and 39 is the right cursor. If the left cursor is pressed, then the leftPressed variable is set to true, and when it is released the leftPressed variable is set to false.

We don't want our frog to continuously move if a key is held down. We want the player to have to press then release then press again for movement. Add the variables shown below that we will use to force the player to press, release, and press again for movement.

```
var y = 444;
var width = 30;
var height = 30;

var rightPressed = false;
var leftPressed = false;
var upPressed = false;
var downPressed = false;
var up = true;
var down = true;
var right = true;
var left = true;

document.addEventListener("keydown", keyDownHanc
document.addEventListener("keyup", keyUpHandler,
```

The Frog moving logic

We now have the variables for storing the info about the pressed keys, event listeners and relevant functions set up. Now we'll get onto the actual code to use all that and move the frog on the screen. Inside the draw() function, we will check if the up key is pressed when each frame is rendered.

If the up arrow is pressed, the frog will move 44 pixels towards the top of the canvas. We will add a Boolean variable to only allow movement for one keypress in the first if statement and reset the ability to move in the second if statement.

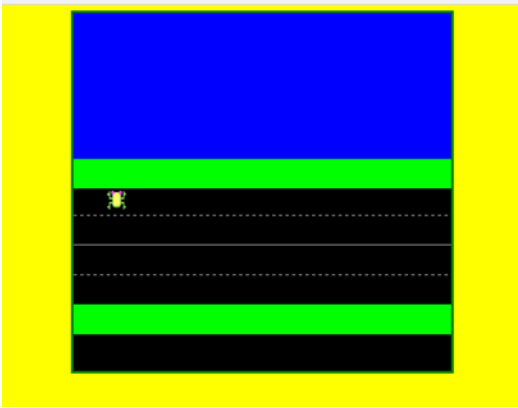
Enter the highlighted code in your draw() function.

```
80  }
81
82  function draw(){
83      drawBackground();
84      drawFrog();
85
86  if (upPressed==true && up==true) {
87      y = y - 44;
88      up = false;
89      }
90  if (upPressed==false) {
91      up = true;
92      }
93
94      requestAnimationFrame(draw);
95  }
96  draw();
```

If you preview in browser you will notice that we need to clear our canvas and redraw our frog or we will leave a trail of frogs when we move. Add the clearRect method as shown to clear your canvas with each execution of your draw function.

```
ctx.fillRect(0, 0, 570, 220);
}

function drawFrog(){
    ctx.drawImage(frog, sx, sy, swidth, sheight, x, y, width,
}

function draw(){
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawBackground();
    drawFrog();

if (upPressed==true && up==true) {
    y = y - 44;
    up = false;
    }
if (upPressed==false) {
    up = true;
    }

    requestAnimationFrame(draw);
}
draw();
```

You should now be able to move your frog up the canvas by pressing the up arrow key. Try to figure out how to move down, left and right before the next lesson.



Frogger Lesson 4

We will use the same logic to move our frog with the down key as we did with the up key.

Copy and paste the code segment we wrote earlier to move the frog up the canvas and revise as shown below. Remember that the y value on the canvas gets larger as you move down the canvas. After entering this code segment you should be able to move your frog up and down the canvas.

```
81
82  function draw(){
83      ctx.clearRect(0, 0, canvas.width, canvas.
84      drawBackground();
85      drawFrog();
86
87  if (upPressed==true && up==true) {
88      y = y - 44;
89      up = false;
90      }
91  if (upPressed==false) {
92      up = true;
93      }
94
95  if (downPressed==true && down==true) {
96      y = y + 44;
97      down = false;
98      }
99  if (downPressed==false) {
100     down = true;
101     }
102
103     requestAnimationFrame(draw);
104 }
105 draw();
```

Use the same logic to move the frog in the horizontal (x) direction. After entering this code segment you should be able to move your frog up and down and to the right on the canvas.

```
function draw(){
    ctx.clearRect(0, 0, canvas.width, canvas
    drawBackground();
    drawFrog();

if (upPressed==true && up==true) {
    y = y - 44;
    up = false;
    }
if (upPressed==false) {
    up = true;
    }

if (downPressed==true && down==true) {
    y = y + 44;
    down = false;
    }
if (downPressed==false) {
    down = true;
    }

if (rightPressed==true && right==true) {
    x = x + 44;
    right = false;
    }
if (rightPressed==false) {
    right = true;
    }

    requestAnimationFrame(draw);
```

Finally, add code to move the frog to the left. After entering this code segment you should be able to move your frog up and down and to the left and right on the canvas.

```
        }
    if (rightPressed==false) {
        right = true;
        }

    if (leftPressed==true && left==true) {
        x = x - 44;
        left = false;
        }
    if (leftPressed==false) {
        left = true;
        }

        requestAnimationFrame(draw);
```

We want our frog to face right if he is moving to the right and left if he is moving to the left, so we will need to change the sx value of our image accordingly. When the frog is moving up or down we want the sx value to be 0. When the frog is moving to the right we want sx = 40, and to the left sx = 80. All 4 of the frogs within the image are 40 pixels wide.

Add highlighted line below to always select the first frog in the image when moving up.

```

    if (upPressed==true && up==true) {
        y = y - 44;
        up = false;
        sx=0;
        }
    if (upPressed==false) {
        up = true;
```

Add highlighted line below to always select the first frog in the image when moving down.

```
if (downPressed==true && down==true) {
    y = y + 44;
    down = false;
    sx=0;
    }
if (downPressed==false) {
    down = true;
    }
```

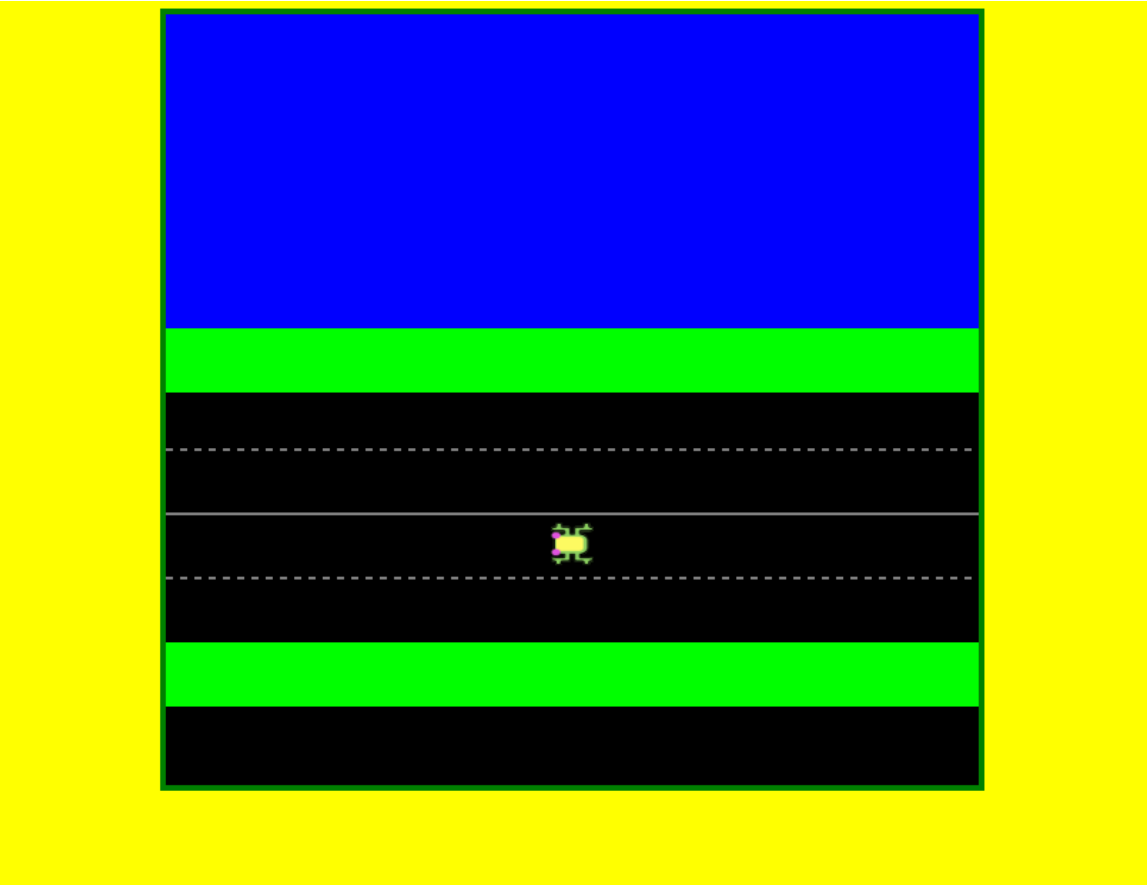
Add highlighted line below to always select the second frog in the image when moving right.

```
    }

if (rightPressed==true && right==true) {
    x = x + 44;
    right = false;
    sx=40;
    }
if (rightPressed==false) {
    right = true;
    }
```

Add highlighted line below to always select the third frog in the image when moving left.

```

if (leftPressed==true && left==true) {
    x = x - 44;
    left = false;
    sx=80;
    }
if (leftPressed==false) {
    left = true;
    }
```

Preview in browser to make sure your frog is facing the correct direction.



Frogger Lesson 5

Just to keep the program somewhat organized, I created a function called moveFrog() containing the code segments handling movement. Cut and paste the if statements shown below so that they are above your draw() function, then define a moveFrog function as shown in the highlighted line below. Remember to add a closing bracket as shown in line 118 below.

Source Code frogger.js

Code Split Live

```
81
82  function moveFrog(){
83      if (upPressed==true && up==true) {
84      y = y - 44;
85      up = false;
86      sx=0;
87      }
88  if (upPressed==false) {
89      up = true;
90      }
91
92  if (downPressed==true && down==true) {
93      y = y + 44;
94      down = false;
95      sx=0;
96      }
97  if (downPressed==false) {
98      down = true;
99      }
100
101 if (rightPressed==true && right==true) {
102     x = x + 44;
103     right = false;
104     sx=40;
105     }
106 if (rightPressed==false) {
107     right = true;
108     }
109
110 if (leftPressed==true && left==true) {
111     x = x - 44;
112     left = false;
113     sx=80;
114     }
115 if (leftPressed==false) {
116     left = true;
117     }
118 }
119
120 function draw(){
```

Now call for the moveFrog() function to be executed within your draw() function. Notice that you can collapse functions in Dreamweaver to help organize your program.

```
24   document.addEventListener("keydown", keyDownHandler, false);
25   document.addEventListener("keyup", keyUpHandler, false);
26
27   function keyDownHandler(e)
28   {...}
34
35   function keyUpHandler(e)
36   {...}
42
43   function drawBackground(){...}
77
78   function drawFrog(){...}
81
82   function moveFrog(){...}
119
120  function draw(){
121      ctx.clearRect(0, 0, canvas.width, canvas.height);
122      drawBackground();
123      drawFrog();
124      moveFrog();
125
126      requestAnimationFrame(draw);
127  }
128  draw();
```

Drawing Cars

Save the image below, or copy the froggercars.png file from the Arrington folder and paste into the same location as your frogger.html and frogger.js files.

Create a variable named car and initialize as your froggercars.png image.

```
18  var downPressed = false;
19  var up = true;
20  var down = true;
21  var right = true;
22  var left = true;
23
24  var car = new Image(); car.src = "froggercars.png";
25
26  document.addEventListener("keydown", keyDownHandler,
27  document.addEventListener("keyup", keyUpHandler, fals
28
29  function keyDownHandler(e)
30  {...}
```

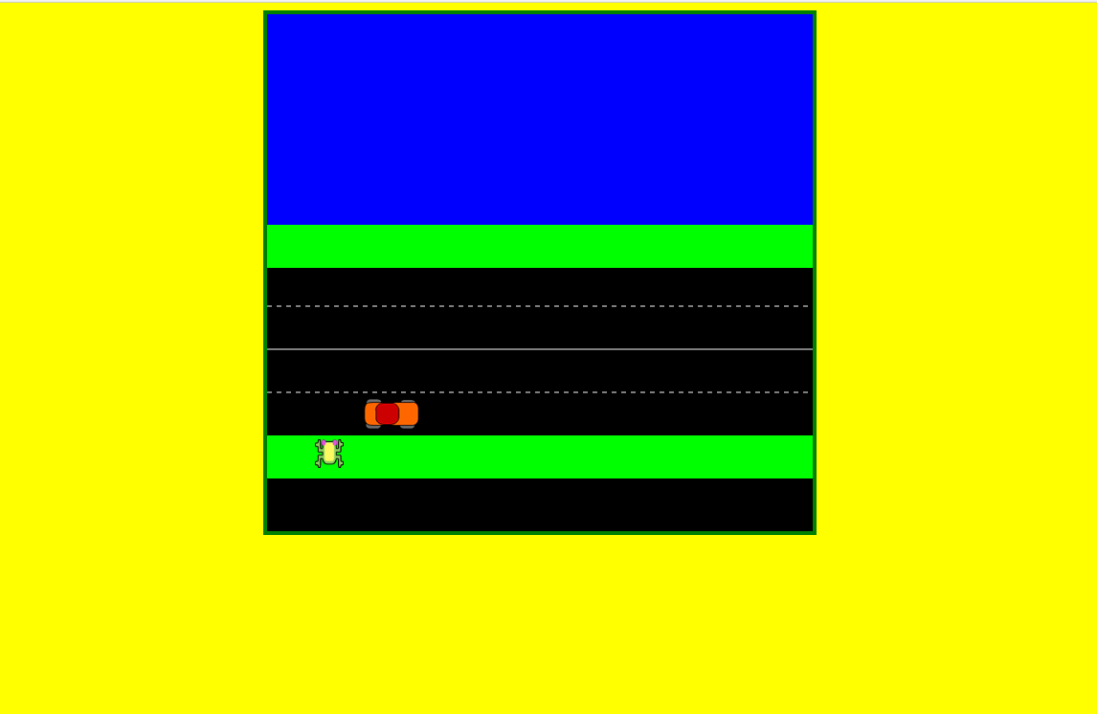
Next create a function named drawCars just above your draw() function using the javascript syntax: ctx.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);. Note we are selecting the car at the far left of the image (sx = 0) and placing it a x = 100 and y = 400.

```
44
45  function drawBackground(){...}
79
80  function drawFrog(){...}
83
84  function moveFrog(){...}
121
122 function drawCars(){
123     ctx.drawImage(car,0, 0, 60, 35, 100, 400,60,35);
124 }
125
126 function draw(){
127     ctx.clearRect(0, 0, canvas.width, canvas.height);
128     drawBackground();
```

Call for your drawCars() function to execute within the draw() function as shown below.

```
function moveFrog(){...}

function drawCars(){
    ctx.drawImage(car,0, 0, 60, 35, 100, 400,60,35);
}

function draw(){
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawBackground();
    drawFrog();
    moveFrog();
    drawCars();

    requestAnimationFrame(draw);
}
draw();
```

Preview in browser to check.



Frogger Lesson 6

Make sure you have a frog that moves with the press of an arrow key and a single car drawn on the canvas before proceeding.

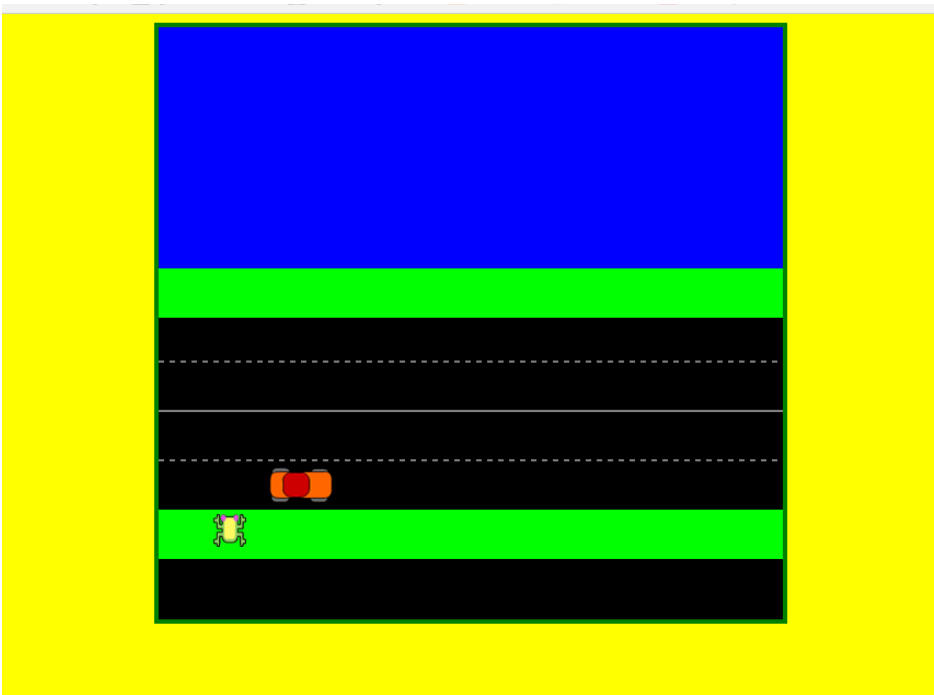
We want to make the car move across the canvas in a continuous loop, so we will need to vary the x position of the car. Inside your drawCars() function, replace the x position of this first car with a variable named carX1.

```
function drawFrog(){...}

function moveFrog(){...}

function drawCars(){
    ctx.drawImage(car,0, 0, 60, 35, carX1, 400,60,35);
}

function draw(){
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawBackground();
    drawFrog();
    moveFrog();
    drawCars();

    requestAnimationFrame(draw);
}
draw();
```

Assign an Initial variable to carX1 of 100.

```
var downPressed = false;
var up = true;
var down = true;
var right = true;
var left = true;

var car = new Image(); car.src = "froggercars.png";
var carX1 = 100;

document.addEventListener("keydown", keyDownHandler,
document.addEventListener("keyup", keyUpHandler, fals
```

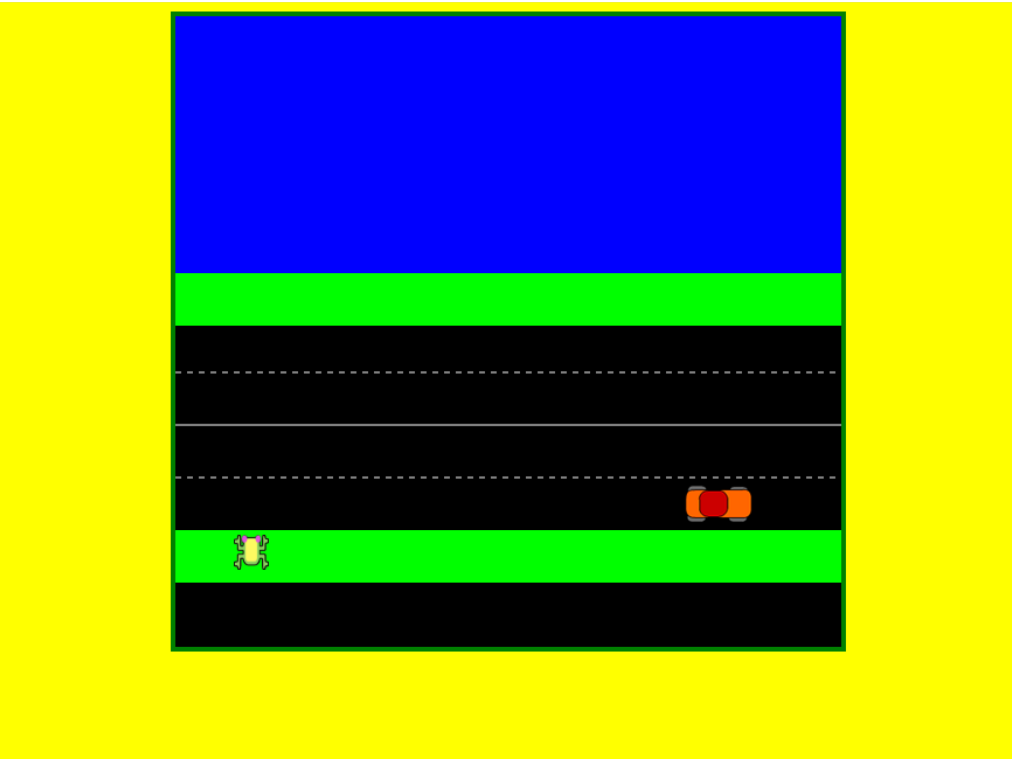
Make sure your car is still shown on your canvas.



Back in your drawCars() function, add the if else statement to move the car 5 pixels to the right each frame until the car's x position is no longer less than the width of the canvas plus 100 pixels. At that point the car's x position is reset to -100 (100 pixels to the left of the left edge of the canvas).

```
122
123  function drawCars(){
124      ctx.drawImage(car,0, 0, 60, 35, carX1, 400,60,35);
125       if (carX1 < canvas.width + 100) {
126           carX1 = carX1 + 5;
127           }
128           else {
129               carX1 = -100;
130               }
131  }
132
133  function draw(){
134      ctx.clearRect(0, 0, canvas.width, canvas.height);
135      drawBackground();
136      drawFrog();
```

You should have a nice smooth looping animation now for your car.



Instead of the same car looping every time, let's add some code to randomly select one of the four cars on our image. First replace the sx value of 0 with a variable carSX1.

```
function moveFrog(){...}

function drawCars(){
    ctx.drawImage(car, carSX1, 0, 60, 35, carX1, 400,60,35);
     if (carX1 < canvas.width + 100) {
         carX1 = carX1 + 5;
         }
         else {
             carX1 = -100;
             }
}

function draw(){
```

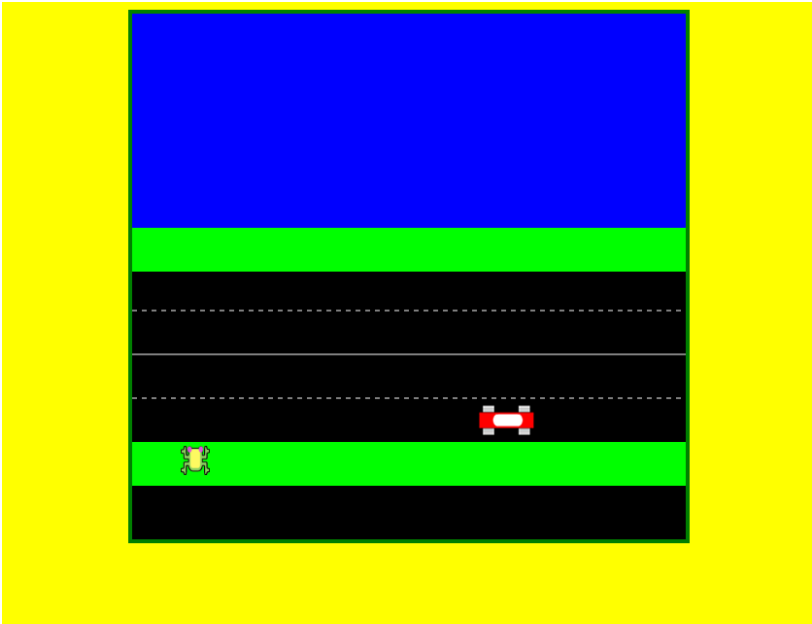
Assign an initial value of 0 to our carSX1 variable.

```
var downPressed = false;
var up = true;
var down = true;
var right = true;
var left = true;

var car = new Image(); car.src = "froggercars.png";
var carX1 = 100;
var carSX1 = 0;

document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);
```

Then add the highlighted code to reset carSX1 to either 0, 60, 120, or 180 each time the car restarts its loop. Math.random generates a number between 0 and 1 and Math.floor rounds down to the nearest whole number, so our code should generate a random integer of 0, 1, 2, or 3. Multiplying by 60 then gives us the sx value of one of our 4 cars.

```
function moveFrog(){...}

function drawCars(){
    ctx.drawImage(car, carSX1, 0, 60, 35, carX1, 400,60,35);
     if (carX1 < canvas.width + 100) {
         carX1 = carX1 + 5;
         }
         else {
             carX1 = -100;
             carSX1 = (Math.floor(Math.random() * 4)) * 60;
             }
}

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

Preview to make sure all four of your car colors show up as intended.



Before we add a collision test between car and frog we need to go back into the moveFrog() function and add a bit of code so that we cannot move the frog off the canvas. First, add the highlighted code to restrict the up movement.

```
82  function drawFrog(){...}
85
86  function moveFrog(){
87      if (upPressed==true && up==true && y > 20) {
88      y = y - 44;
89      up = false;
90      sx=0;
91      }
92  if (upPressed==false) {
93      up = true;
94      }
95
96  if (downPressed==true && down==true) {
97      y = y + 44;
```

Next add the highlighted code to restrict down movement.

```
        sx=0;
        }
    if (upPressed==false) {
        up = true;
        }

    if (downPressed==true && down==true && y + height < canvas.height - 80) {
        y = y + 44;
        down = false;
        sx=0;
        }
    if (downPressed==false) {
        down = true;
```

Next add the highlighted code to restrict movement to the right.

```
        down = true;
        }

    if (rightPressed==true && right==true && x + width < canvas.width-20) {
        x = x + 44;
        right = false;
        sx=40;
        }
    if (rightPressed==false) {
        right = true;
```

Next add the highlighted code to restrict movement to the left.

```
        }
    if (rightPressed==false) {
        right = true;
        }

    if (leftPressed==true && left==true && x > 20) {
        x = x - 44;
        left = false;
        sx=80;
        }
    if (leftPressed==false) {
        left = true;
```

Frogger Lesson 7

Our next step is to create a collision detection between frog and car. First we want to replace some more of the constants used in drawing our car with variables to make the collision detection code more readily understandable.

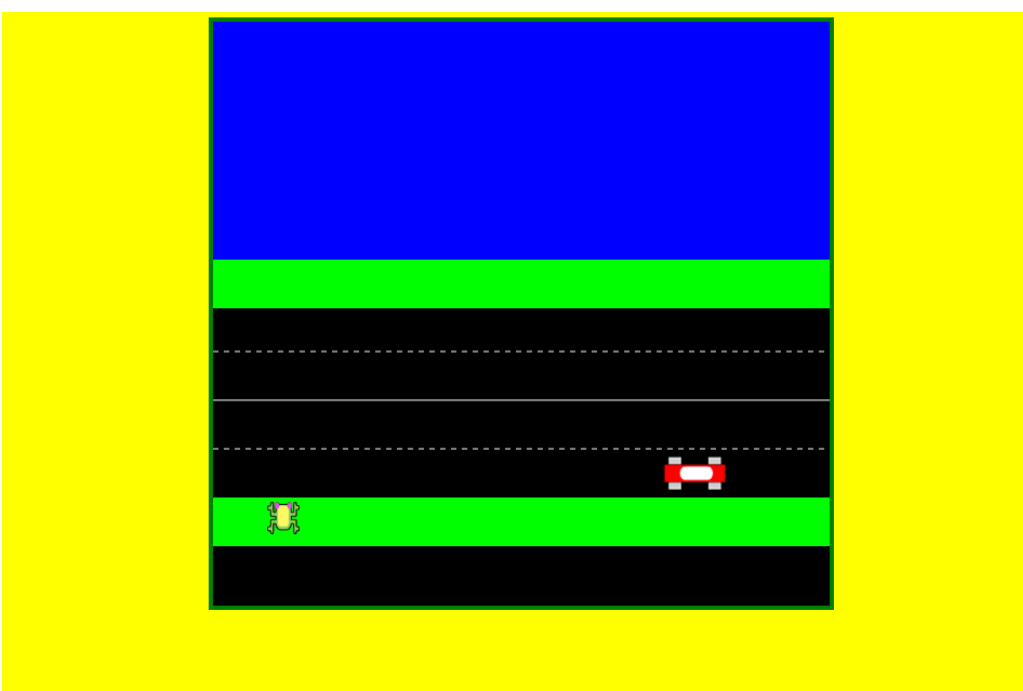
Add the highlighted variables shown below.

```
23
24  var car = new Image(); car.src = "froggercars.png";
25  var carX1 = 100;
26  var carSX1 = 0;
27  var carY1 = 400;
28  var carWidth = 60;
29  var carHeight = 35;
30
31  document.addEventListener("keydown", keyDownHandler, false);
32  document.addEventListener("keyup", keyUpHandler, false);
33
34  function keyDownHandler(e)
35  {...}
41
42  function keyUpHandler(e)
43  {...}
49
50  function drawBackground(){...}
84
85  function drawFrog(){...}
88
89  function moveFrog(){...}
126
127 function drawCars(){
128     ctx.drawImage(car, carSX1, 0, 60, 35, carX1, 400,60,35);
129      if (carX1 < canvas.width + 100) {
130          carX1 = carX1 + 5;
```

In your drawCars() function, replace constants with variables as shown below.

```
var car = new Image(); car.src = "froggercars.png";
var carX1 = 100;
var carSX1 = 0;
var carY1 = 400;
var carWidth = 60;
var carHeight = 35;

document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);

function keyDownHandler(e)
{...}

function keyUpHandler(e)
{...}

function drawBackground(){...}

function drawFrog(){...}

function moveFrog(){...}

function drawCars(){
    ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
     if (carX1 < canvas.width + 100) {
         carX1 = carX1 + 5;
```

Check to make sure the program is still working.



Next create a runOver() function that will check to see if the frog and the car are overlapping. If they are overlapping (colliding) we will simply reset the y position of the frog to near the bottom of the canvas. We will add to this outcome later.

Also, we will revise this function in just a moment so that we can use it for multiple cars, but I wanted to start with this version to try and make it as understandable as possible.

```
function moveFrog(){...}

function drawCars(){...}

function runOver (){
    if (carX1 <= x + width &&
    carX1 + carWidth >= x &&
    carY1 + carHeight >= y &&
    carY1 <= y + height) {
        y= 488;
        }
}

function draw(){
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawBackground();
    drawFrog();
    moveFrog();
```

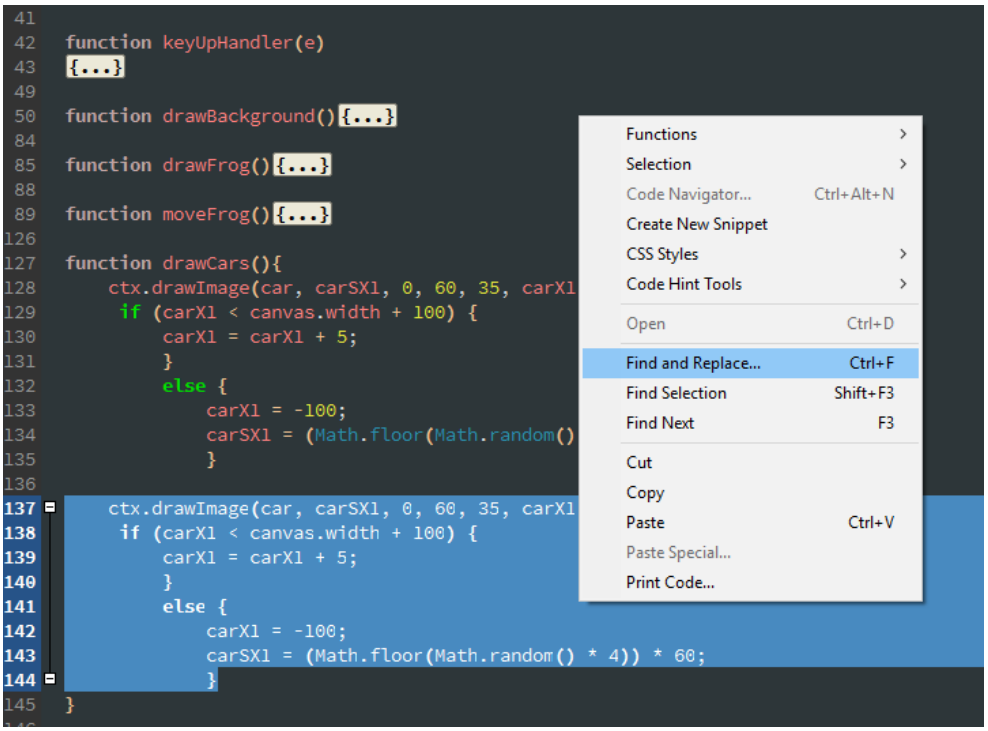
Enter the highlighted code within your draw() function to call for execution of your runOver() function.

```
138 function runOver (){
139     if (carX1 <= x + width &&
140     carX1 + carWidth >= x &&
141     carY1 + carHeight >= y &&
142     carY1 <= y + height) {
143         y= 488;
144         }
145 }
146
147 function draw(){
148     ctx.clearRect(0, 0, canvas.width, canvas.height);
149     drawBackground();
150     drawFrog();
151     moveFrog();
152     drawCars();
153     runOver();
154
155     requestAnimationFrame(draw);
156 }
157 draw();
```

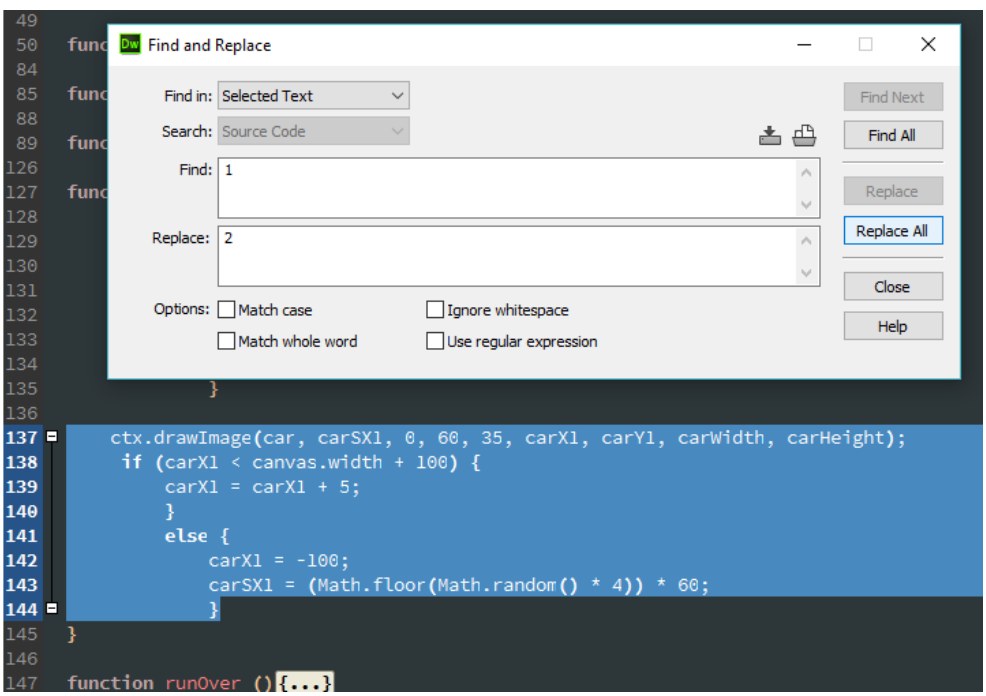
To create a second car, return to your drawCars() function and start by copying and pasting the code for the first car.

```
88
89  function moveFrog(){...}
126
127 function drawCars(){
128     ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
129      if (carX1 < canvas.width + 100) {
130          carX1 = carX1 + 5;
131          }
132          else {
133              carX1 = -100;
134              carSX1 = (Math.floor(Math.random() * 4)) * 60;
135              }
136
137     ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
138      if (carX1 < canvas.width + 100) {
139          carX1 = carX1 + 5;
140          }
141          else {
142              carX1 = -100;
143              carSX1 = (Math.floor(Math.random() * 4)) * 60;
144              }
145 }
146
147 function runOver (){...}
155
```

Make sure the code segment that you just pasted is selected, then right-click and select “Find and Replace…”.



<u>MAKE SURE “Selected Text” IS SELECTED FROM THE “Find in:” pulldown menu</u>. Enter 1 in the “Find” window and 2 in the “Replace” window, then click the “Replace All” button.



Check that all 1's are replaced with 2's. We "accidently" changed a couple of 1's in lines 138 and 142 that we should not have, but we will change those back later.

```
126
127  function drawCars(){
128      ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
129       if (carX1 < canvas.width + 100) {
130           carX1 = carX1 + 5;
131           }
132           else {
133               carX1 = -100;
134               carSX1 = (Math.floor(Math.random() * 4)) * 60;
135               }
136
137      ctx.drawImage(car, carSX2, 0, 60, 35, carX2, carY2, carWidth, carHeight);
138       if (carX2 < canvas.width + 200) {
139           carX2 = carX2 + 5;
140           }
141           else {
142               carX2 = -200;
143               carSX2 = (Math.floor(Math.random() * 4)) * 60;
144               }
145  }
146
147  function runOver (){...}
```

Define and initialize the highlighted variables for our second car.

```
23
24  var car = new Image(); car.src = "froggercars.png";
25  var carX1 = 100;
26  var carSX1 = 0;
27  var carY1 = 400;
28  var carWidth = 60;
29  var carHeight = 35;
30  var carX2 = 500;
31  var carSX2 = 60;
32  var carY2 = 400;
33
34  document.addEventListener("keydown", keyDownHandler, false);
35  document.addEventListener("keyup", keyUpHandler, false);
36
37  function keyDownHandler(e)
38  {...}
44
```

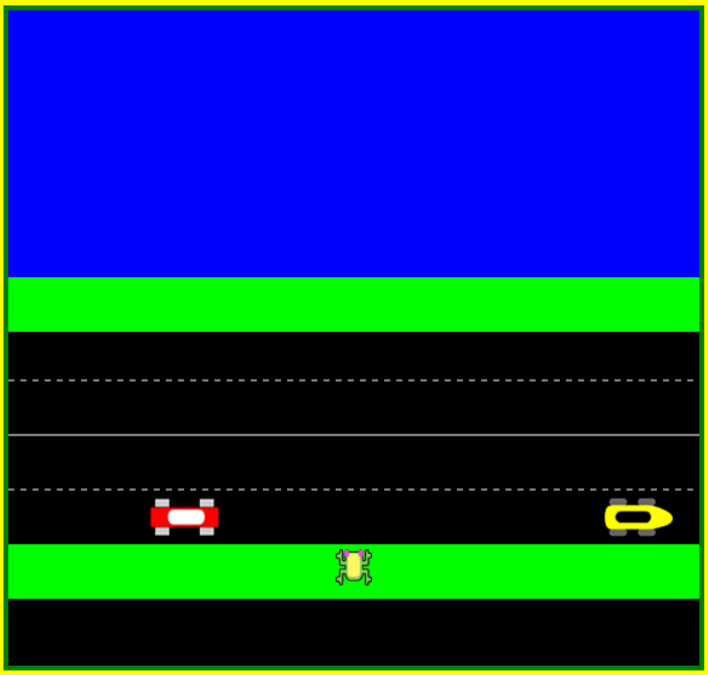
Reset the second cars restart position. (“accidently” changed when we did the Find and Replace all 1’s)

```
129
130  function drawCars(){
131      ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
132       if (carX1 < canvas.width + 100) {
133           carX1 = carX1 + 5;
134           }
135           else {
136               carX1 = -100;
137               carSX1 = (Math.floor(Math.random() * 4)) * 60;
138               }
139
140      ctx.drawImage(car, carSX2, 0, 60, 35, carX2, carY2, carWidth, carHeight);
141       if (carX2 < canvas.width + 200) {
142           carX2 = carX2 + 5;
143           }
144           else {
145               carX2 = -100;
146               carSX2 = (Math.floor(Math.random() * 4)) * 60;
147               }
148  }
149
150  function runOver (){...}
```

Also, reset the point at which the car will return to the restart position.

```
129
130  function drawCars(){
131      ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
132       if (carX1 < canvas.width + 100) {
133           carX1 = carX1 + 5;
134           }
135           else {
136               carX1 = -100;
137               carSX1 = (Math.floor(Math.random() * 4)) * 60;
138               }
139
140      ctx.drawImage(car, carSX2, 0, 60, 35, carX2, carY2, carWidth, carHeight);
141       if (carX2 < canvas.width + 100) {
142           carX2 = carX2 + 5;
143           }
144           else {
145               carX2 = -100;
146               carSX2 = (Math.floor(Math.random() * 4)) * 60;
147               }
148  }
149
150  function runOver (){...}
```

Check to see if you have two cars. Notice that our collision test only works for one of the cars.



REVISE your runOver() function to include a for loop that will check for x and y values entered into arrays for each car created.

After revising your runOver() function check to see if both cars now “collide” with the frog.

```
130  function drawCars(){...}
149
150  function runOver (){
151
152      var carsX = [carX1, carX2];
153      var carsY = [carY1, carY2];
154
155      for (i = 0; i < carsX.length; i++){
156          if (carsX[i] <= x + width &&
157          carsX[i] + carWidth >= x &&
158          carsY[i] + carHeight >= y &&
159          carsY[i] <= y + height) {
160              y= 488;
161          }
162      }
163  }
164
165  function draw(){...}
175  draw();
```

Frogger Lesson 8

Our next step is to add more cars, so in this lesson we will be working on our drawCars() function. We will be drawing a total of 8 cars, so before we actually begin drawing, we will set up a for loop like the one we created for our collision test in the previous lesson.

First, create arrays for the variables that will need to be specified for each separate car.

```
91
92  function moveFrog(){...}
129
130 function drawCars(){
131
132     var carsSX = [carSX1, carSX2];
133     var carsX = [carX1, carX2];
134     var carsY = [carY1, carY2];
135
136 ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth,
137      if (carX1 < canvas.width + 100) {
138          carX1 = carX1 + 5;
139          }
140          else {
141              carX1 = -100;
142              carSX1 = (Math.floor(Math.random() * 4)) * 60;
143              }
144
145     ctx.drawImage(car, carSX2, 0, 60, 35, carX2, carY2, carWic
```

Next, create a for loop as shown below.

```
129
130 function drawCars(){
131
132     var carsSX = [carSX1, carSX2];
133     var carsX = [carX1, carX2];
134     var carsY = [carY1, carY2];
135
136     for (i = 0; i < carsX.length; i++){
137
138     }
139
140 ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, car
141      if (carX1 < canvas.width + 100) {
142          carX1 = carX1 + 5;
143          }
144          else {
145              carX1 = -100;
146              carSX1 = (Math.floor(Math.random() * 4)) *
147              }
```

Select and cut the highlighted drawImage() function.

```
129
130  function drawCars(){
131
132      var carsSX = [carSX1, carSX2];
133      var carsX = [carX1, carX2];
134      var carsY = [carY1, carY2];
135
136      for (i = 0; i < carsX.length; i++){
137
138      }
139
140  ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
141       if (carX1 < canvas.width + 100) {
142           carX1 = carX1 + 5;
143           }
144           else {
145               carX1 = -100;
146               carSX1 = (Math.floor(Math.random() * 4)) * 60;
147               }
148
```

And paste within your for loop.

```
 91
 92  function moveFrog(){...}
129
130  function drawCars(){
131
132      var carsSX = [carSX1, carSX2];
133      var carsX = [carX1, carX2];
134      var carsY = [carY1, carY2];
135
136      for (i = 0; i < carsX.length; i++){
137          ctx.drawImage(car, carSX1, 0, 60, 35, carX1, carY1, carWidth, carHeight);
138      }
139
140
141       if (carX1 < canvas.width + 100) {
142           carX1 = carX1 + 5;
143           }
144           else {
145               carX1 = -100;
```

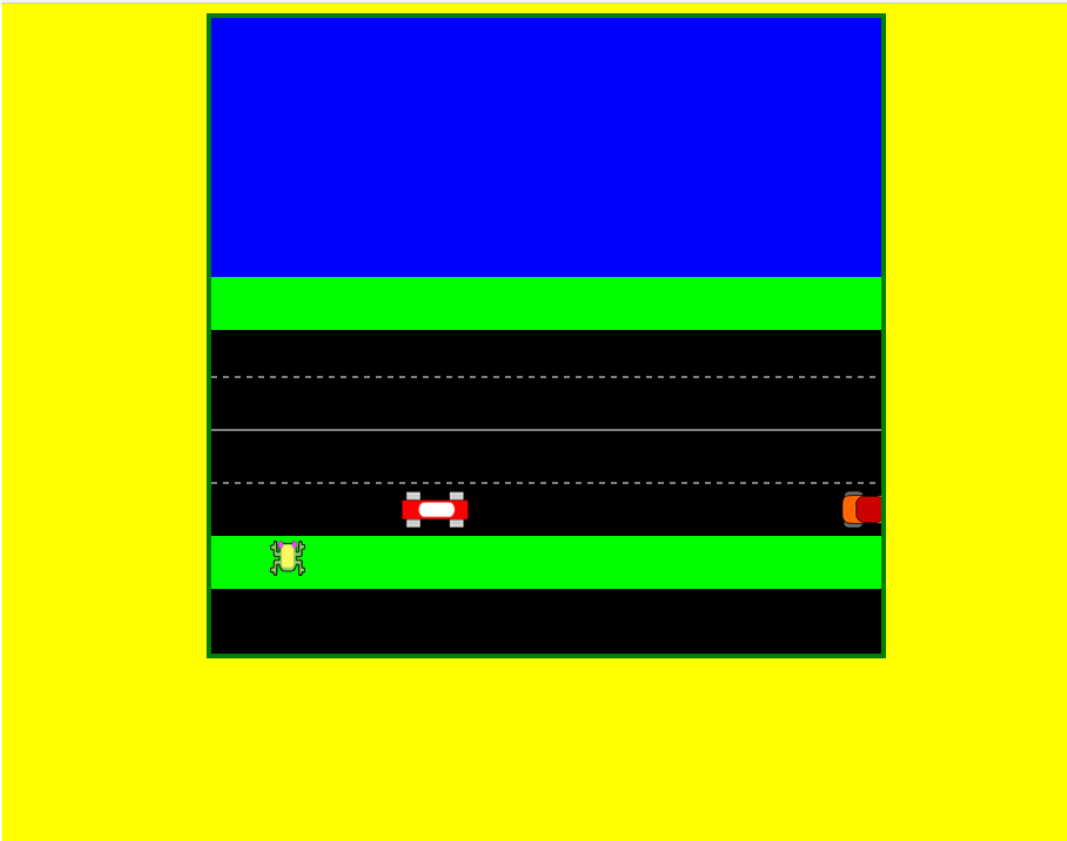
Replace the variable carSX1 with carsSX[i].

```
129
130  function drawCars(){
131
132      var carsSX = [carSX1, carSX2];
133      var carsX = [carX1, carX2];
134      var carsY = [carY1, carY2];
135
136      for (i = 0; i < carsX.length; i++){
137          ctx.drawImage(car, carsSX[i], 0, 60, 35, carX1, carY1, carWidth, carHeight)
     ;
138      }
139
140
141       if (carX1 < canvas.width + 100) {
142           carX1 = carX1 + 5;
143           }
```

Replace the variable carX1 with carsX[i]. Replace the variable carY2 with carsY[i].

```
129
130  function drawCars(){
131
132      var carsSX = [carSX1, carSX2];
133      var carsX = [carX1, carX2];
134      var carsY = [carY1, carY2];
135
136      for (i = 0; i < carsX.length; i++){
137          ctx.drawImage(car, carsSX[i], 0, 60, 35, carsX[i], carsY[i], carWidth, carHeight);
138      }
139
140
141       if (carX1 < canvas.width + 100) {
142           carX1 = carX1 + 5;
```

Check to make sure your program is still working as before.



Delete this line of code which is unneeded now.

```
138     }
139
140
141      if (carX1 < canvas.width + 100) {
142          carX1 = carX1 + 5;
143          }
144          else {
145              carX1 = -100;
146              carSX1 = (Math.floor(Math.random() * 4)) * 60;
147              }
148
149     ctx.drawImage(car, carSX2, 0, 60, 35, carX2, carY2, carWidth, carHeight);
150      if (carX2 < canvas.width + 100) {
151          carX2 = carX2 + 5;
152          }
153          else {
154              carX2 = -100;
155              carSX2 = (Math.floor(Math.random() * 4)) * 60;
156              }
157 }
```

Your drawCars() function should now look like the screenshot below.

```
91
92  function moveFrog(){...}
129
130 function drawCars(){
131
132     var carsSX = [carSX1, carSX2];
133     var carsX = [carX1, carX2];
134     var carsY = [carY1, carY2];
135
136     for (i = 0; i < carsX.length; i++){
137         ctx.drawImage(car, carsSX[i], 0, 60, 35, carsX[i], carsY[i], car
138     }
139
140
141      if (carX1 < canvas.width + 100) {
142          carX1 = carX1 + 5;
143          }
144          else {
145              carX1 = -100;
146              carSX1 = (Math.floor(Math.random() * 4)) * 60;
147              }
148
149      if (carX2 < canvas.width + 100) {
150          carX2 = carX2 + 5;
151          }
152          else {
153              carX2 = -100;
154              carSX2 = (Math.floor(Math.random() * 4)) * 60;
155              }
156 }
157
158 function runOver (){
159
```

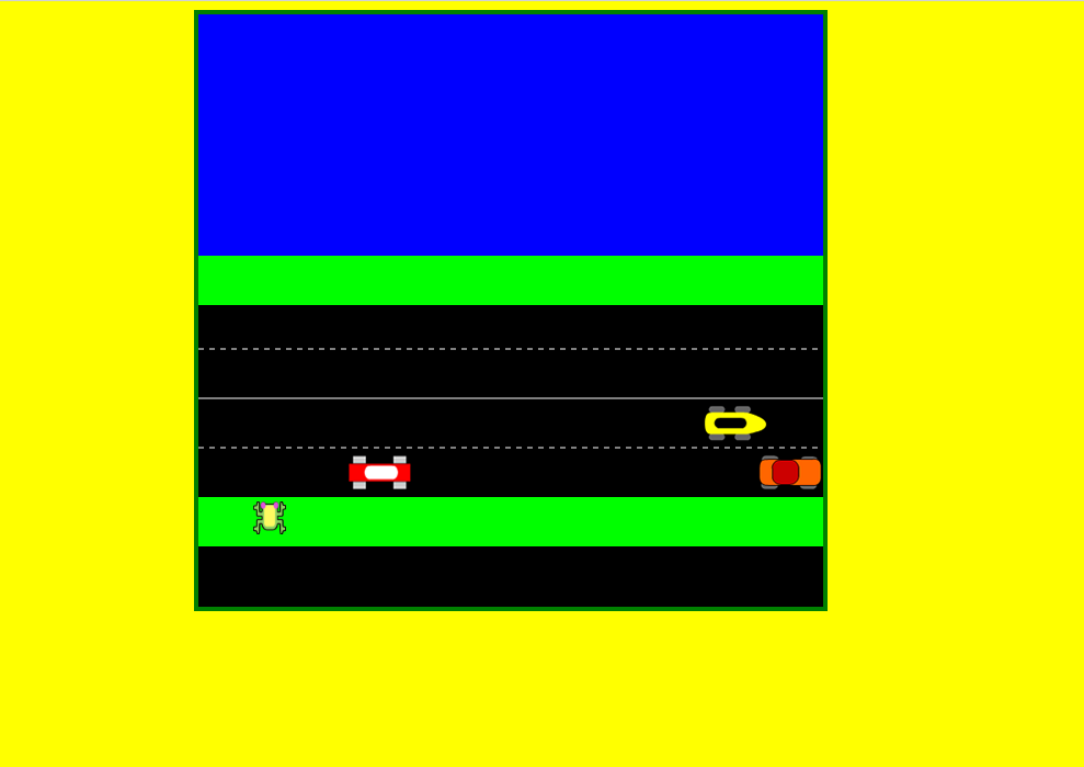
Now we will define and initialize variables for a brand new car.

```
24  var car = new Image(); car.src = "froggercars.png";
25  var carX1 = 100;
26  var carSX1 = 0;
27  var carY1 = 400;
28  var carWidth = 60;
29  var carHeight = 35;
30  var carX2 = 500;
31  var carSX2 = 60;
32  var carY2 = 400;
33  var carX3 = 460;
34  var carSX3 = 120;
35  var carY3 = 355;
36
37  document.addEventListener("keydown", keyDownHandler,
```

Add the new variables to the arrays in your drawCars() function.

```
94
95  function moveFrog(){...}
132
133 function drawCars(){
134
135     var carsSX = [carSX1, carSX2, carSX3];
136     var carsX = [carX1, carX2, carX3];
137     var carsY = [carY1, carY2, carY3];
138
139     for (i = 0; i < carsX.length; i++){
140         ctx.drawImage(car, carsSX[i], 0, 60, 35, carsX[i
141     }
142
143
144      if (carX1 < canvas.width + 100) {
145          carX1 = carX1 + 5;
146          }
147          else {
```

You should see a third car now. This car will be parked on the road until we update our code. Also this car won't "collide" with our frog until we add its variables to the array in our runOver() function.



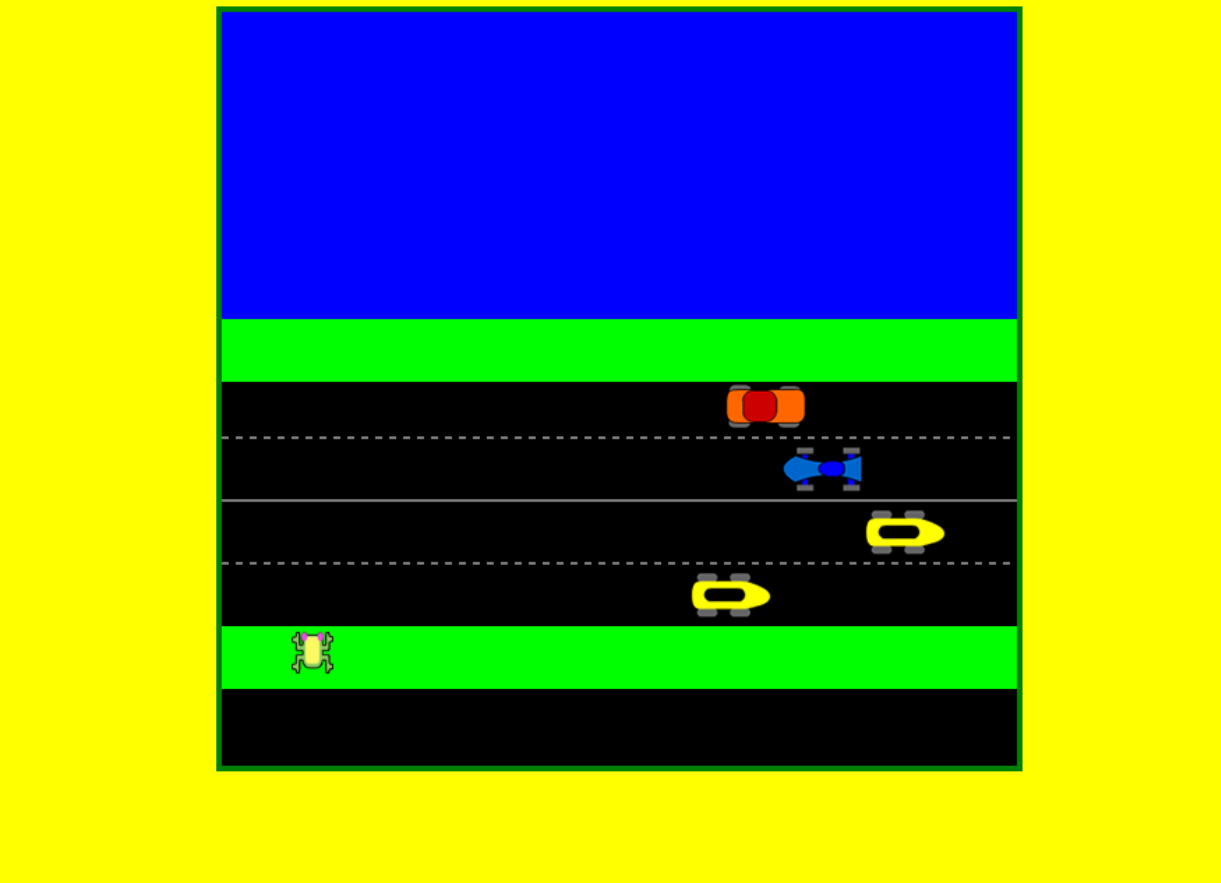
Let's draw a couple more cars before we wrap up the lesson. Define and initialize the new variables highlighted below.

```
33  var carX3 = 460;
34  var carSX3 = 120;
35  var carY3 = 355;
36  var carX4 = 400;
37  var carSX4 = 180;
38  var carY4 = 310;
39  var carX5 = 360;
40  var carSX5 = 0;
41  var carY5 = 265;
42
43  document.addEventListener("keydown", keyDownHandler, false);
44  document.addEventListener("keyup", keyUpHandler, false);
45
46  function keyDownHandler(e)
47  {...}
53
54  function keyUpHandler(e)
55  {...}
```

Add the newly created variables to the arrays in our drawCars() function.

```
function moveFrog(){...}

function drawCars(){

    var carsSX = [carSX1, carSX2, carSX3, carSX4, carSX5];
    var carsX = [carX1, carX2, carX3, carX4, carX5];
    var carsY = [carY1, carY2, carY3, carY4, carY5];

    for (i = 0; i < carsX.length; i++){
        ctx.drawImage(car, carsSX[i], 0, 60, 35, carsX[i], carsY[i], c
    }
```

Preview



Frogger Lesson 9

I want to have two cars on each of the four lanes, so I will create three more cars for a total of eight cars. Two cars at y = 400, two cars at y = 355, two cars at y = 310, and two cars at y = 265. You can set the x position wherever you like.

Add the highlighted code to define and initialize variables to draw three more cars.

```
var carX4 = 400;
var carSX4 = 180;
var carY4 = 310;
var carX5 = 360;
var carSX5 = 0;
var carY5 = 265;
var carX6 = 60;
var carSX6 = 120;
var carY6 = 355;
var carX7 = 100;
var carSX7 = 180;
var carY7 = 310;
var carX8 = 160;
var carSX8 = 0;
var carY8 = 265;

document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);

function keyDownHandler(e)
```

Update the carsSX list of variables..

```
function drawCars(){

    var carsSX = [carSX1, carSX2, carSX3, carSX4, carSX5, carSX6, carSX7, carSX8];
```

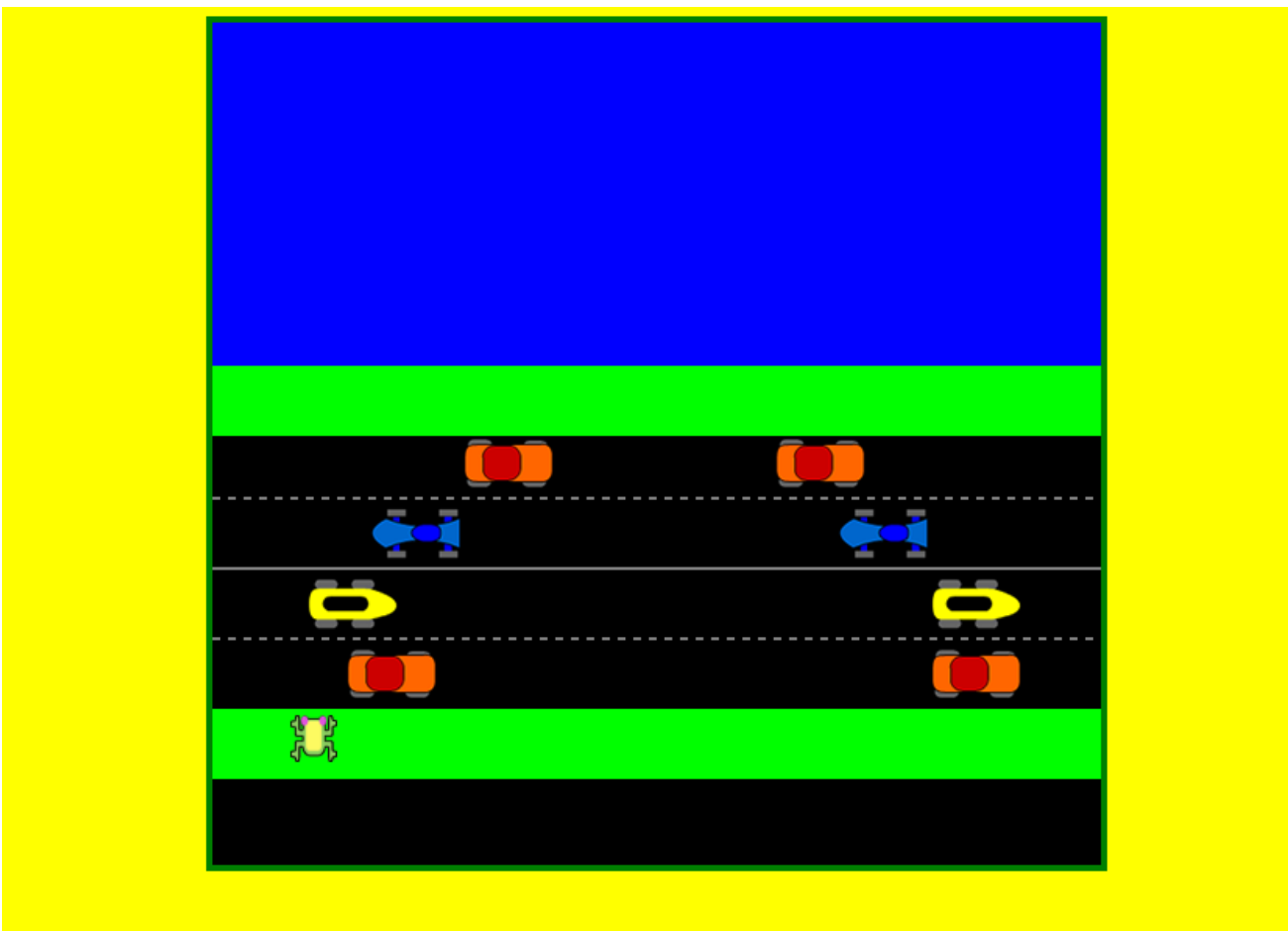
Update the carsX list of variables.

```
function drawCars(){

    var carsSX = [carSX1, carSX2, carSX3, carSX4, carSX5, carSX6, carSX7, carSX8];
    var carsX = [carX1, carX2, carX3, carX4, carX5, carX6, carX7, carX8];
```

Update the carsY list of variables.

```
function drawCars(){

    var carsSX = [carSX1, carSX2, carSX3, carSX4, carSX5, carSX6, carSX7, carSX8];
    var carsX = [carX1, carX2, carX3, carX4, carX5, carX6, carX7, carX8];
    var carsY = [carY1, carY2, carY3, carY4, carY5, carY6, carY7, carY8];

    for (i = 0; i < carsX.length; i++){
        ctx.drawImage(car, carsSX[i], 0, 60, 35, carsX[i], carsY[i], carWidth, carHeight);
    }

```

I now have my 8 cars, but only the two in the first lane are moving.



Define a new function moveCars() between the drawCars() function and the runover() function.

```
            carX2 = carX2 + 5;
            }
            else {
                carX2 = -100;
                carSX2 = (Math.floor(Math.random() * 4)) * 60;
                }
  }

  function moveCars(){

  }


  function runOver (){

      var carsX = [carX1, carX2];
```

Select and cut the two if else statements from your drawCars() function.

```
    var carsSX = [carSX1, carSX2, carSX3, carSX4, carSX5, carSX6, carSX7,
    var carsX = [carX1, carX2, carX3, carX4, carX5, carX6, carX7, carX8];
    var carsY = [carY1, carY2, carY3, carY4, carY5, carY6, carY7, carY8];

    for (i = 0; i < carsX.length; i++){
        ctx.drawImage(car, carsSX[i], 0, 60, 35, carsX[i], carsY[i], carWi
carHeight);
    }


     if (carX1 < can
         carX1 = car
         }
         else {
             carX1 =
             carSX1 =                                  ) * 60;
             }

     if (carX2 < can
         carX2 = car
         }
         else {
             carX2 =
             carSX2 =                                  ) * 60;
             }
}

function moveCars(){

}
```

Functions
Selection
Code Navigator... Ctrl+Alt+N
Create New Snippet
CSS Styles
Code Hint Tools
Open Ctrl+D
Find and Replace... Ctrl+F
Find Selection Shift+F3
Find Next F3
Cut
Copy
Paste Ctrl+V
Paste Special...
Print Code...

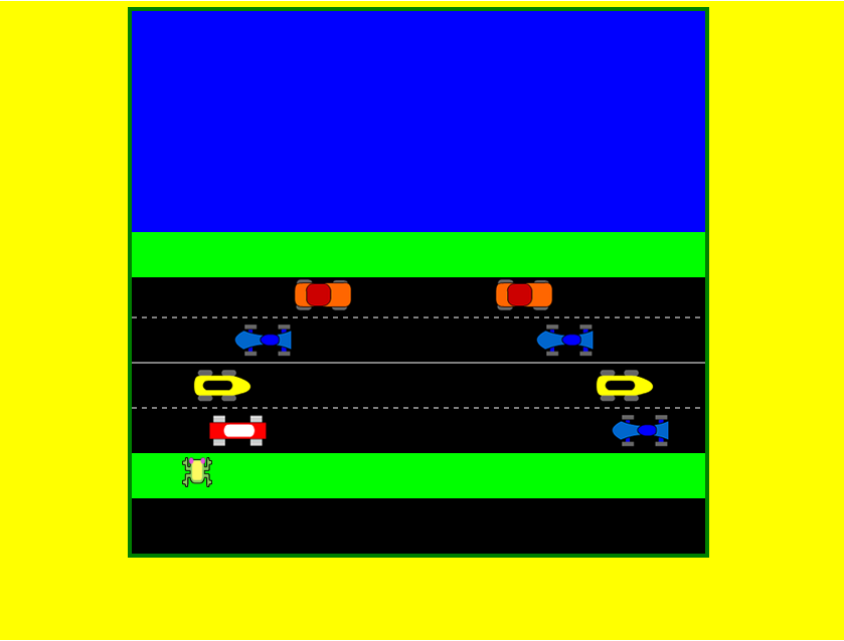
Paste the two if else statements inside your moveCars() function.

```
154     for (i = 0; i < carsX.length; i++){
155         ctx.drawImage(car, carsSX[i], 0, 60, 35, carsX[i], carsY[
    carHeight);
156     }
157 }
158
159 function moveCars(){
160
161  if (carX1 < canvas.width + 100) {
162      carX1 = carX1 + 5;
163      }
164      else {
165          carX1 = -100;
166          carSX1 = (Math.floor(Math.random() * 4)) * 60;
167          }
168
169  if (carX2 < canvas.width + 100) {
170      carX2 = carX2 + 5;
171      }
172      else {
173          carX2 = -100;
174          carSX2 = (Math.floor(Math.random() * 4)) * 60;
175          }
176 }
177
178
179 function runOver (){
```

Add the highlighted line in your draw() function to call the moveCars() function.

```
172         else {
173             carX2 = -100;
174             carSX2 = (Math.floor(Math.random() * 4)) * 60;
175             }
176     }
177
178
179     function runOver (){...}
193
194     function draw(){
195         ctx.clearRect(0, 0, canvas.width, canvas.height);
196         drawBackground();
197         drawFrog();
198         moveFrog();
199         drawCars();
200         moveCars();
201         runOver();
202
203         requestAnimationFrame(draw);
204     }
```

Check to make sure your program is still working.



Copy one of the if else statements and paste. Change the variables to control a third car.

```
function drawCars(){...}

function moveCars(){

 if (carX1 < canvas.width + 100) {
     carX1 = carX1 + 5;
     }
     else {
         carX1 = -100;
         carSX1 = (Math.floor(Math.random() * 4)) * 60;
         }

 if (carX2 < canvas.width + 100) {
     carX2 = carX2 + 5;
     }
     else {
         carX2 = -100;
         carSX2 = (Math.floor(Math.random() * 4)) * 60;
         }

 if (carX3 < canvas.width + 100) {
     carX3 = carX3 + 5;
     }
     else {
         carX3 = -100;
         carSX3 = (Math.floor(Math.random() * 4)) * 60;
         }
}


function runOver (){...}

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

You should have three cars moving now. We want the cars in alternating lanes to move in the opposite direction so we will need to change the code. Revise the if else statement for car 3 to loop from right to left.

```
171         }
172         else {
173             carX2 = -100;
174             carSX2 = (Math.floor(Math.random() * 4)) * 60;
175             }
176
177     if (carX3 > -100) {
178         carX3 = carX3 - 5;
179         }
180         else {
181             carX3 = canvas.width + 100;
182             carSX3 = (Math.floor(Math.random() * 4)) * 60;
183             }
184     }
185
186
187     function runOver (){...}
```

Make another copy of your if else statement and revise to control “car 6” (car 6 has the same y value as car 3 so they are in the same lane). After entering this code you should have 4 cars moving in two lanes.

```
181             carX3 = canvas.width + 100;
182             carSX3 = (Math.floor(Math.random() * 4)) * 60;
183             }
184
185     if (carX6 > -100) {
186         carX6 = carX6 - 5;
187         }
188         else {
189             carX6 = canvas.width + 100;
190             carSX6 = (Math.floor(Math.random() * 4)) * 60;
191             }
192     }
193
194
195     function runOver (){...}
209
```

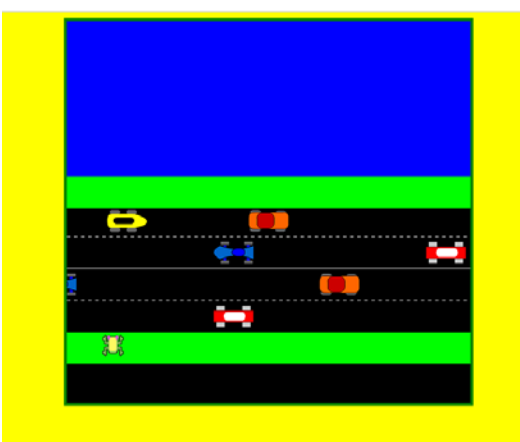
Cars 4 and 7 are in the third lane (Y = 310) and we want those two cars to move from left to right. Paste copies of if else statements and revise as shown below.

```
184
185    if (carX6 > -100) {
186        carX6 = carX6 - 5;
187        }
188        else {
189            carX6 = canvas.width + 100;
190            carSX6 = (Math.floor(Math.random() * 4)) * 60;
191            }
192
193    if (carX4 < canvas.width + 100) {
194        carX4 = carX4 + 5;
195        }
196        else {
197            carX4 = -100;
198            carSX4 = (Math.floor(Math.random() * 4)) * 60;
199            }
200
201    if (carX7 < canvas.width + 100) {
202        carX7 = carX7 + 5;
203        }
204        else {
205            carX7 = -100;
206            carSX7 = (Math.floor(Math.random() * 4)) * 60;
207            }
208    }
209
```

Cars 5 and 8 are in the top lane (Y = 265) and we want those two cars to move from right to left. Paste copies of if else statements and revise as shown below.

```
            carSX7 = (Math.floor(Math.random() * 4)) * 60;
            }

    if (carX5 > -100) {
        carX5 = carX5 - 5;
        }
        else {
            carX5 = canvas.width + 100;
            carSX5 = (Math.floor(Math.random() * 4)) * 60;
            }

    if (carX8 > -100) {
        carX8 = carX8 - 5;
        }
        else {
            carX8 = canvas.width + 100;
            carSX8 = (Math.floor(Math.random() * 4)) * 60;
            }
}


function runOver (){...}
```

You should have four cars moving now.



Next, add the variables for all of our cars to the arrays in our runOver() function. (quickest way would be to copy and paste the arrays from the drawCars() function.) After entering this code, you should have a working collision test between the frog and all cars.

```
109
110  function moveFrog(){...}
147
148  function drawCars(){...}
158
159  function moveCars(){...}
225
226  function runOver (){
227
228      var carsX = [carX1, carX2, carX3, carX4, carX5, carX6, carX7, carX8];
229      var carsY = [carY1, carY2, carY3, carY4, carY5, carY6, carY7, carY8];
230
231      for (i = 0; i < carsX.length; i++){
232          if (carsX[i] <= x + width &&
233          carsX[i] + carWidth >= x &&
234          carsY[i] + carHeight >= y &&
235          carsY[i] <= y + height) {
236              y= 488;
237          }
238      }
239  }
```

Frogger Lesson 10

In this lesson, we will start drawing the rectangles that the frog will be able to float on to get across the water to reach the top of the canvas. We will call the rectangles logs even though some will move upstream.

First define a drawLogs() function immediately above the draw() function as shown.

```
158
159  function moveCars() {...}
225
226  function runOver () {...}
240
241  function drawLogs(){
242
243  }
244
245  function draw(){
246      ctx.clearRect(0, 0, canvas.width, canvas.height);
247      drawBackground();
```

Use JavaScript's fillStyle and fillRect methods to create a rectangle. Enter variables for the rectangle's x position, y position, width and height as shown.

```
159  function moveCars() {...}
225
226  function runOver () {...}
240
241  function drawLogs(){
242      ctx.fillStyle = "tan";
243      ctx.fillRect(logX1, logY1, logWidth, logHeight);
244  }
245
246  function draw(){
247      ctx.clearRect(0, 0, canvas.width, canvas.height);
248      drawBackground();
```

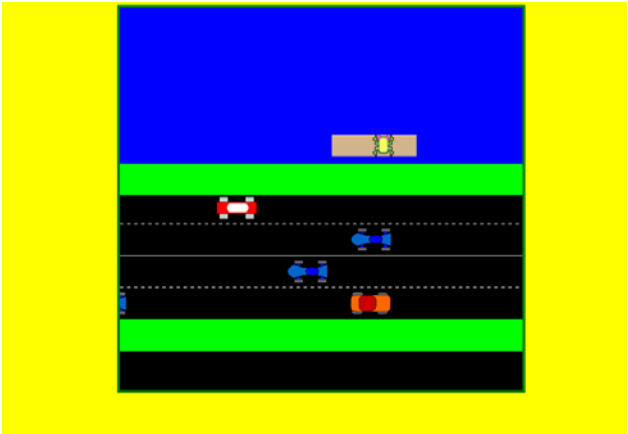
Go to your variable stack and define and assign initial values for the rectangle's x position, y position, width and height as shown. Although the logWidth and logHeight will not vary, assigning descriptive names to these values will help in setting up our collision test between frog and log.

```
49  var carSX8 = 0;
50  var carY8 = 265;
51
52  var logX1 = 300;
53  var logY1 = 180;
54  var logWidth = 120;
55  var logHeight = 30;
56
57  document.addEventListener("keydown", keyDownHan
```

Add the highlighted line below to call your drawLogs() function. Make sure the call for drawLogs() is below drawBackground() and above drawFrog(). The order in which the functions are called will determine which images appear in front or behind the others.

```
246  function drawLogs(){
247      ctx.fillStyle = "tan";
248      ctx.fillRect(logX1, logY1, logWidth, logHeight);
249  }
250
251  function draw(){
252      ctx.clearRect(0, 0, canvas.width, canvas.height);
253      drawBackground();
254      drawLogs();
255      drawFrog();
256      moveFrog();
257      drawCars();
258      moveCars();
259      runOver();
260
261      requestAnimationFrame(draw);
262  }
263  draw();
```

A browser preview should look like the screenshot below now.



Next create a function moveLogs() below your drawLogs() function as shown.

```
231  function runOver (){...}
245
246  function drawLogs(){
247      ctx.fillStyle = "tan";
248      ctx.fillRect(logX1, logY1, logWidth, logHeight);
249  }
250
251  function moveLogs(){
252
253  }
254
255  function draw(){
256      ctx.clearRect(0, 0, canvas.width, canvas.height);
257      drawBackground();
```

Add the highlighted code to your moveLogs() function to create a left to right 2 pixels per frame scrolling animation.

```
function drawLogs(){
    ctx.fillStyle = "tan";
    ctx.fillRect(logX1, logY1, logWidth, logHeight);
}

function moveLogs(){
 if (logX1 < canvas.width + 100) {
     logX1 = logX1 + 2;
     }
     else {
         logX1 = -100;
         }
}

function draw(){
```

Call your moveLogs() function from within your draw() function. Your log should now loop across the canvas.

```
function draw(){
    ctx.clearRect(0, 0, canvas.width, c
    drawBackground();
    drawLogs();
    moveLogs();
    drawFrog();
    moveFrog();
    drawCars();
    moveCars();
    runOver();

    requestAnimationFrame(draw);
}
draw();
```

Create a float() function below your moveLogs() function as shown.

```
function moveLogs(){
 if (logX1 < canvas.width + 100) {
     logX1 = logX1 + 2;
     }
     else {
         logX1 = -100;
         }
}

function float(){

}

function draw(){
    ctx.clearRect(0, 0, canvas.width, canvas.heig
```

Add the highlighted code to your float() function to check if any part of the log rectangle is “overlapping” the frog’s space. We are only checking for y less than 220 because that is where the “water” begins. If y is less than 220, and if the frog is contacting the log, and if the frog remains on the canvas, then its x position is set to advance at the same speed as this log (2 pixels per frame). If the frog’s y position is less than 220 and there is no overlap, the frog is sent to the bottom of the canvas (y = 488).

```
function float(){
    if (y < 220){
        if (logX1 <= x + width &&
        logX1 + logWidth >= x &&
        logY1 + logHeight >= y &&
        logY1 <= y + height) {
            if(x < canvas.width-30){
                x = x +2;
            }
        }
    else {
        y = 488;
        }
    }
}
```

Add code to call the float() function within your draw() function. Now if you preview in browser, the frog should be able to float on the log and return to the bottom if it hits the water.

```
275
276   function draw(){
277       ctx.clearRect(0, 0, canvas.width, canva
278       drawBackground();
279       drawLogs();
280       moveLogs();
281       drawFrog();
282       moveFrog();
283       drawCars();
284       moveCars();
285       runOver();
286       float();
287
288       requestAnimationFrame(draw);
289   }
290   draw();
```

Next add the variables highlighted below that we will use to create another log.

```
53   var logY1 = 180;
54   var logWidth = 120;
55   var logHeight = 30;
56   var logX2 = 40;
57   var logY2 = 180;
58
59   document.addEventListener("keydown", keyDownHandler, fals
60   document.addEventListener("keyup", keyUpHandler, false);
61
62   function keyDownHandler(e)
```

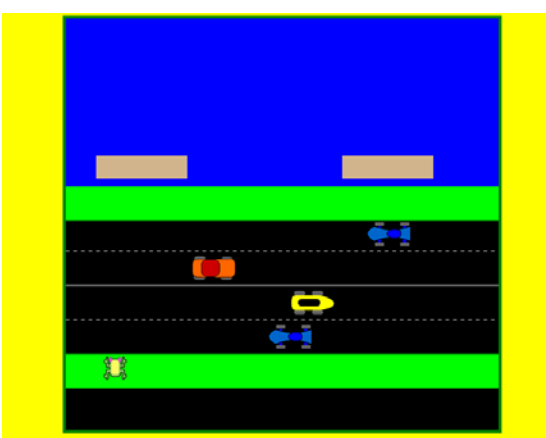
Create two lists of variables within your drawLogs() function as shown below.

```
function drawLogs(){
    ctx.fillStyle = "tan";
    var logsX = [logX1, logX2];
    var logsY = [logY1, logY2]

    ctx.fillRect(logX1, logY1, logWidth, logHeight);
}

function moveLogs(){
 if (logX1 < canvas.width + 100) {
```

Create a for loop to enable us to draw logs by simply adding new variables for the x and y positions of the rectangle. Change logX1 to logsX[i] and logY1 to logsY[i] in your fillRect parameter list as shown.

```
function drawLogs(){
    ctx.fillStyle = "tan";
    var logsX = [logX1, logX2];
    var logsY = [logY1, logY2]

    for (i = 0; i < logsX.length; i++){
    ctx.fillRect(logsX[i], logsY[i], logWidth, logHeight);
    }
}

function moveLogs(){
```

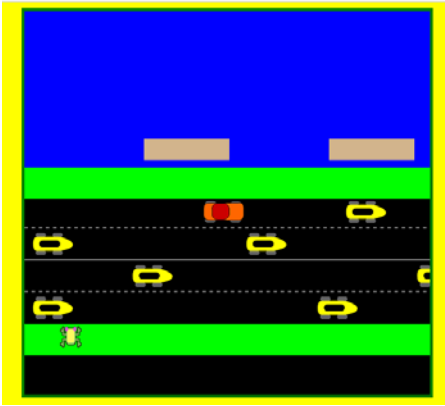
You should now have two logs (only one moving log) after entering the code shown above.



In your moveLogs() function, copy the if else statements that move logX1. Paste the if else statement and replace the logX1 variables with logX2 (in 4 places) as shown below.

```
257
258 function moveLogs(){
259  if (logX1 < canvas.width + 100) {
260      logX1 = logX1 + 2;
261      }
262      else {
263          logX1 = -100;
264          }
265
266  if (logX2 < canvas.width + 100) {
267      logX2 = logX2 + 2;
268      }
269      else {
270          logX2 = -100;
271          }
272 }
273
274 function float(){
275     if (y < 220){
276         if (logX1 <= x + width &&
277             logX1 + logWidth >= x &&
278             logY1 + logHeight >= y &&
```

You should have two moving logs now, but only one log is set up for the frog to float on. We will revise our float() function to handle multiple logs in the next lesson.



Frogger Lesson 11

In this lesson we will create more logs for our frog to float upon in its quest to reach the top of the canvas, but we need to begin by revising our float() function to accommodate multiple logs. After entering the code highlighted below for our float() function, the frog should be able to float on both logs.

```
273
274 function float(){
275     if (logX1 <= x + width &&
276                 logX1 + logWidth >= x &&
277                 logY1 + logHeight >= y &&
278                 logY1 <= y + height) {
279                     if(x < canvas.width - 30){
280                         x = x + 2;
281                     }
282     }
283
284     else if (logX2 <= x + width &&
285                 logX2 + logWidth >= x &&
286                 logY2 + logHeight >= y &&
287                 logY2 <= y + height) {
288                     if(x < canvas.width - 30){
289                         x = x + 2;
290                     }
291     }
292
293     else if (y < 220){
294             y = 488;
295         }
296 }
297
298 function draw(){
299     ctx.clearRect(0, 0, canvas.width, canvas.height);
300     drawBackground();
```

To draw a third log, go inside your drawLogs() function and enter new logX3 and logY3 variables in your two arrays as shown.

```
247
248  function drawLogs(){
249      ctx.fillStyle = "tan";
250      var logsX = [logX1, logX2, logX3];
251      var logsY = [logY1, logY2, logY3];
252
253      for (i = 0; i < logsX.length; i++){
254      ctx.fillRect(logsX[i], logsY[i], logWidth, logHeight);
255      }
256  }
257
258  function moveLogs(){
259   if (logX1 < canvas.width + 100) {
260       logX1 = logX1 + 2;
```

Define and initialize your new variables in the variable stack as shown.

```
55   var logHeight = 30;
56   var logX2 = 40;
57   var logY2 = 180;
58   var logX3 = 100;
59   var logY3 = 136;
60
61   document.addEventListener("keydown", keyD
```

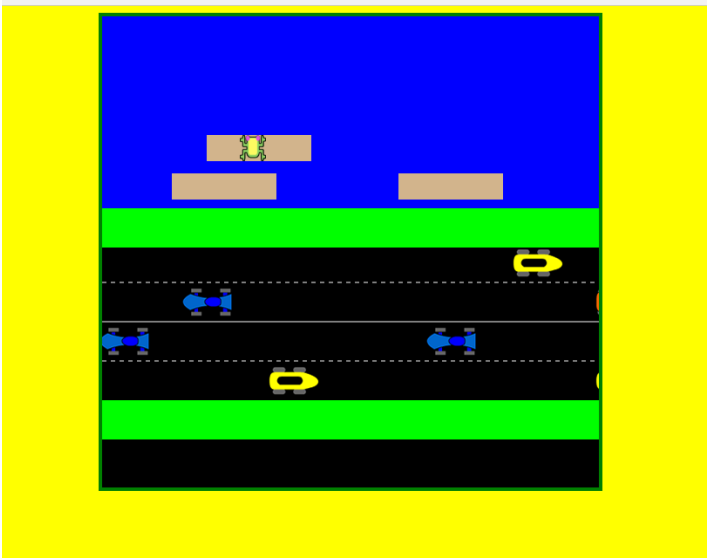
Add highlighted code for the third log in your moveLogs() function as shown. This log will move from right to left at 2 pixels per frame.

```
259
260  function moveLogs(){
261   if (logX1 < canvas.width + 100) {
262       logX1 = logX1 + 2;
263       }
264       else {
265           logX1 = -100;
266           }
267
268   if (logX2 < canvas.width + 100) {
269       logX2 = logX2 + 2;
270       }
271       else {
272           logX2 = -100;
273           }
274
275   if (logX3 > 0-logWidth ) {
276       logX3 = logX3 - 2;
277       }
278       else {
279           logX3 = canvas.width + 100;
280           }
281  }
282
283  function float(){
284      if (logX1 <= x + width &&
285                  logX1 + logWidth >= x &&
```

Add highlighted code for the third log in your float() function as shown. Since this log will move from right to left, the if statement that “floats” the frog is different than that used for logs floating from left to right (lines 306 and 307).

```
282
283  function float(){
284      if (logX1 <= x + width &&
285                  logX1 + logWidth >= x &&
286                  logY1 + logHeight >= y &&
287                  logY1 <= y + height) {
288                      if(x < canvas.width - 30){
289                          x = x + 2;
290                      }
291      }
292
293      else if (logX2 <= x + width &&
294                  logX2 + logWidth >= x &&
295                  logY2 + logHeight >= y &&
296                  logY2 <= y + height) {
297                      if(x < canvas.width - 30){
298                          x = x + 2;
299                      }
300      }
301
302      else if (logX3 <= x + width &&
303                  logX3 + logWidth >= x &&
304                  logY3 + logHeight >= y &&
305                  logY3 <= y + height) {
306                      if(x > 0){
307                          x = x - 2;
308                      }
309      }
310
311      else if (y < 220){
312          y = 488;
313          }
314  }
```

Preview in browser and your frog should be able to float on all three logs.



Simply repeat steps to draw and animate log number 4.

1. Add variables to array in drawLogs() function.

```
function drawLogs(){
    ctx.fillStyle = "tan";
    var logsX = [logX1, logX2, logX3, logX4];
    var logsY = [logY1, logY2, logY3, logY4];

    for (i = 0; i < logsX.length; i++){
    ctx.fillRect(logsX[i], logsY[i], logWidth, logHeight);
    }
}

```

2. Define and initialize variables in variable stack, and you should see a new (stationary) log.

```
var logX2 = 40;
var logY2 = 180;
var logX3 = 100;
var logY3 = 136;
var logX4 = 400;
var logY4 = 136;

document.addEventListener
```

3. Add the highlighted if else code segment in your moveLogs() function. Log 4 will use the same movement as log 3. After entering this code, all logs should be moving.

```
function moveLogs(){
 if (logX1 < canvas.width + 100) {
     logX1 = logX1 + 2;
     }
     else {
         logX1 = -100;
         }

 if (logX2 < canvas.width + 100) {
     logX2 = logX2 + 2;
     }
     else {
         logX2 = -100;
         }

 if (logX3 > 0-logWidth ) {
     logX3 = logX3 - 2;
     }
     else {
         logX3 = canvas.width + 100;
         }

 if (logX4 > 0-logWidth ) {
     logX4 = logX4 - 2;
     }
     else {
         logX4 = canvas.width + 100;
         }
}

function float(){
```

4. In your float() function, add the highlighted else if statement for your 4th log. After entering this code all logs should allow frog floating.

```
310
311         else if (logX3 <= x + width &&
312                     logX3 + logWidth >= x &&
313                     logY3 + logHeight >= y &&
314                     logY3 <= y + height) {
315                         if(x > 0){
316                             x = x - 2;
317                         }
318         }
319
320         else if (logX4 <= x + width &&
321                     logX4 + logWidth >= x &&
322                     logY4 + logHeight >= y &&
323                     logY4 <= y + height) {
324                         if(x > 0){
325                             x = x - 2;
326                         }
327         }
328
329         else if (y < 220){
330             y = 488;
331             }
332     }
333
334     function draw(){
```

Repeat the following four steps to add logs 5 and 6.

Step 1. Add variables to array.

```
function runOver (){...}

function drawLogs(){
    ctx.fillStyle = "tan";
    var logsX = [logX1, logX2, logX3, logX4, logX5, logX6];
    var logsY = [logY1, logY2, logY3, logY4, logY5, logY6];

    for (i = 0; i < logsX.length; i++){
    ctx.fillRect(logsX[i], logsY[i], logWidth, logHeight);
    }
}
```

Step 2. Define and initialize variables in variable stack.

```
var logY3 = 136;
var logX4 = 400;
var logY4 = 136;
var logX5 = 480;
var logY5 = 92;
var logX6 = 60;
var logY6 = 92;

document.addEventListene
```

Step 3. Add if else statements for each log in your moveLogs() function. I started by copying and pasting the code segments for log 1 and log 2. I changed the speed of log 5 and log 6 from 2 to 3 pixels per frame.

```
if (logX4 > 0-logWidth ) {
    logX4 = logX4 - 2;
    }
    else {
        logX4 = canvas.width + 100;
        }

if (logX5 < canvas.width + 100) {
    logX5 = logX5 + 3;
    }
    else {
        logX5 = -100;
        }

if (logX6 < canvas.width + 100) {
    logX6 = logX6 + 3;
    }
    else {
        logX6 = -100;
        }
}

function float(){
```

Step 4. Add else if statements for each log in your float() function. Since I changed the speed of log 5 and log 6 from 2 to 3 pixels per frame, I will have to change the frog floating speed here as well (lines 352 and 361 below). All logs should support floating frogs after entering this code.

```
337
338         else if (logX4 <= x + width &&
339                     logX4 + logWidth >= x &&
340                     logY4 + logHeight >= y &&
341                     logY4 <= y + height) {
342                         if(x > 0){
343                             x = x - 2;
344                         }
345         }
346
347         else if (logX5 <= x + width &&
348                     logX5 + logWidth >= x &&
349                     logY5 + logHeight >= y &&
350                     logY5 <= y + height) {
351                         if(x < canvas.width - 30){
352                             x = x + 3;
353                         }
354         }
355
356         else if (logX6 <= x + width &&
357                     logX6 + logWidth >= x &&
358                     logY6 + logHeight >= y &&
359                     logY6 <= y + height) {
360                         if(x < canvas.width - 30){
361                             x = x + 3;
362                         }
363         }
364
365         else if (y < 220){
366             y = 488;
367             }
368     }
369
370     function draw(){
```

Frogger Lesson 12

If your code is not working - you can start this lesson using the code from this file. (right click on the page; view page source; click the link to the js file). YOU WILL PROBABLY NEED TO UPDATE THE LINK TO THE JS FILE IN YOUR HTML FILE, AND THE LINKS TO THE FROG AND CARS IMAGES IN YOUR JS FILE.

In this lesson we will draw a 7th and 8th log, and then draw 6 pads to represent the final destination for our frog at the top of the canvas.

First add variables as shown.

```
var logX5 = 480;
var logY5 = 92;
var logX6 = 60;
var logY6 = 92;
var logX7 = 120;
var logY7 = 48;
var logX8 = 500;
var logY8 = 48;

document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);
```

Inside your drawLogs() function, update your variable lists.

```
function drawLogs(){
    ctx.fillStyle = "tan";
    var logsX = [logX1, logX2, logX3, logX4, logX5, logX6, logX7, logX8];
    var logsY = [logY1, logY2, logY3, logY4, logY5, logY6, logY7, logY8];

    for (i = 0; i < logsX.length; i++){
    ctx.fillRect(logsX[i], logsY[i], logWidth, logHeight);
    }
```

Inside your moveLogs() function, add the two highlighted if else statements to loop log7 and log8 from right to left across the canvas.

```
305
306     if (logX6 < canvas.width + 100) {
307         logX6 = logX6 + 3;
308         }
309         else {
310             logX6 = -100;
311             }
312
313     if (logX7 > 0-logWidth ) {
314         logX7 = logX7 - 2;
315         }
316         else {
317             logX7 = canvas.width + 100;
318             }
319
320     if (logX8 > 0-logWidth ) {
321         logX8 = logX8 - 2;
322         }
323         else {
324             logX8 = canvas.width + 100;
325             }
326 }
327
328 function float(){
329     if (logX1 <= x + width &&
```

Inside your float() function, add the two highlighted else if statements to allow the frog to float on log7 and log8.

```
373
374         else if (logX6 <= x + width &&
375                     logX6 + logWidth >= x &&
376                     logY6 + logHeight >= y &&
377                     logY6 <= y + height) {
378                         if(x < canvas.width - 30){
379                             x = x + 3;
380                         }
381         }
382
383         else if (logX7 <= x + width &&
384                     logX7 + logWidth >= x &&
385                     logY7 + logHeight >= y &&
386                     logY7 <= y + height) {
387                         if(x > 0){
388                             x = x - 2;
389                         }
390         }
391
392         else if (logX8 <= x + width &&
393                     logX8 + logWidth >= x &&
394                     logY8 + logHeight >= y &&
395                     logY8 <= y + height) {
396                         if(x > 0){
397                             x = x - 2;
398                         }
399         }
400
401         else if (y < 220){
402             y = 488;
403             }
404     }
405
406     function draw(){
```

Drawing pads

Add the variables shown below.

```
var logY7 = 48;
var logX8 = 500;
var logY8 = 48;

var padWidth = 30;
var padHeight = 30;
var padX1 = 20;
var padY1 = 4;
var padX2 = 120;
var padY2 = 4;
var padX3 = 220;
var padY3 = 4;
var padX4 = 320;
var padY4 = 4;
var padX5 = 420;
var padY5 = 4;
var padX6 = 520;
var padY6 = 4;

document.addEventListener("keyd
document.addEventListener("keyu
```

Create a drawPads() function as shown below.

```
function moveLogs(){...}

function float(){...}

function drawPads(){
    ctx.fillStyle = "seagreen";
    var padsX = [padX1, padX2, padX3, padX4, padX5, padX6];
    var padsY = [padY1, padY2, padY3, padY4, padY5, padY6];

    for (i = 0; i < padsX.length; i++){
    ctx.fillRect(padsX[i], padsY[i], padWidth, padHeight);
    }
}

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

Add the line to call the drawPads() function from within the draw() function.

```
function draw(){
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawBackground();
    drawLogs();
    moveLogs();
    drawPads();
    drawFrog();
    moveFrog();
    drawCars();
    moveCars();
```

Preview in browser.

Next we will program the game so that if a frog reaches a pad, an image of the frog will remain on the pad and the frog will return to the bottom of the canvas.

First add the variables shown below. We will consider pad1 and pad2 false until a frog reaches them.

```
var padX6 = 520;
var padY6 = 4;

var pad1 = false;
var pad2 = false;

document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);

function keyDownHandler(e)
```

Create an onPad() function as shown below for the first two pads.

```
function onPad(){
    if (padX1 <= x + width &&
        padX1 + padWidth >= x &&
        padY1 + padHeight >= y &&
        padY1 <= y + height) {
            pad1 = true;
            y = 488;
            }

    else if (padX2 <= x + width &&
            padX2 + padWidth >= x &&
            padY2 + padHeight >= y &&
            padY2 <= y + height) {
                pad2 = true;
                y = 488;
                }

    else if (y < 48){
        y = 488;
        }

    var pads = [pad1, pad2];
    var padsX = [padX1, padX2];
    var padsY = [padY1, padY2];

    for (i = 0; i < pads.length; i++){
        if (pads[i] === true) {
            ctx.drawImage(frog, 0, 0, 40, 40, padsX[i], padsY[i], 30, 30);
        }
    }
}

function draw(){
```

Call the onPad() function from within the draw() function.

```
466  function draw(){
467      ctx.clearRect(0, 0, canvas.width, canvas.height);
468      drawBackground();
469      drawLogs();
470      moveLogs();
471      drawPads();
472      onPad();
473      drawFrog();
474      moveFrog();
475      drawCars();
476      moveCars();
```

REVISE THE HIGHLIGTED CODE INSIDE your float() function. We need to add “&& y > 44” to allow the frog to leave a log to reach a pad.

```
413                 logY8 <= y + height) {
414                     if(x > 0){
415                         x = x - 2;
416                     }
417     }
418
419     else if (y < 220 && y > 44){
420         y = 488;
421         }
422 }
423
424 function drawPads(){...}
```

Set the variables shown below.

```
84  var padY6 = 4;
85
86  var pad1 = false;
87  var pad2 = false;
88  var pad3 = false;
89  var pad4 = false;
90  var pad5 = false;
91  var pad6 = false;
92
93  document.addEventListener("keydown", keyDownHandler,
94  document.addEventListener("keyup", keyUpHandler, fal
```

Update the variable lists in the onPad() function.

```
437
438  function onPad(){
439      if (padX1 <= x + width &&
440          padX1 + padWidth >= x &&
441          padY1 + padHeight >= y &&
442          padY1 <= y + height) {
443              pad1 = true;
444              y = 488;
445              }
446
447      else if (padX2 <= x + width &&
448              padX2 + padWidth >= x &&
449              padY2 + padHeight >= y &&
450              padY2 <= y + height) {
451                  pad2 = true;
452                  y = 488;
453                  }
454
455      else if (y < 48){
456          y = 488;
457          }
458
459      var pads = [pad1, pad2, pad3, pad4, pad5, pad6];
460      var padsX = [padX1, padX2, padX3, padX4, padX5, padX6];
461      var padsY = [padY1, padY2, padY3, padY4, padY5, padY6];
462
463      for (i = 0; i < pads.length; i++){
464          if (pads[i] === true) {
465              ctx.drawImage(frog, 0, 0, 40, 40, padsX[i], padsY[i]
466          }
467      }
468  }
469
470  function draw(){
471      ctx.clearRect(0, 0, canvas.width, canvas.height);
```

Add the else if statement for pad 3 inside your onPad() function.

```
function onPad(){
    if (padX1 <= x + width &&
        padX1 + padWidth >= x &&
        padY1 + padHeight >= y &&
        padY1 <= y + height) {
            pad1 = true;
            y = 488;
            }

    else if (padX2 <= x + width &&
            padX2 + padWidth >= x &&
            padY2 + padHeight >= y &&
            padY2 <= y + height) {
                pad2 = true;
                y = 488;
                }

    else if (padX3 <= x + width &&
            padX3 + padWidth >= x &&
            padY3 + padHeight >= y &&
            padY3 <= y + height) {
                pad3 = true;
                y = 488;
                }

    else if (y < 48){
        y = 488;
        }
```

Add if else statements for logs 4 – 6.

```
                padY3 <= y + height) {
                    pad3 = true;
                    y = 488;
                    }

        else if (padX4 <= x + width &&
                padX4 + padWidth >= x &&
                padY4 + padHeight >= y &&
                padY4 <= y + height) {
                    pad4 = true;
                    y = 488;
                    }

        else if (padX5 <= x + width &&
                padX5 + padWidth >= x &&
                padY5 + padHeight >= y &&
                padY5 <= y + height) {
                    pad5 = true;
                    y = 488;
                    }

        else if (padX6 <= x + width &&
                padX6 + padWidth >= x &&
                padY6 + padHeight >= y &&
                padY6 <= y + height) {
                    pad6 = true;
                    y = 488;
                    }

        else if (y < 48){
            y = 488;
            }
```

Frogger Lesson 13

Full credit for this lesson goes to Lexi K. who created functions for losing and winning end states for the frogger game. You can start this lesson using the code from this file. (right click on the page; view page source; click the link to the js file). YOU WILL PROBABLY NEED TO UPDATE THE LINK TO THE JS FILE IN YOUR HTML FILE, AND THE LINKS TO THE FROG AND CARS IMAGES IN YOUR JS FILE.

First create a drawLives() function that will calculate and draw the number of lives the frog has left.

```
428  function drawPads(){...}
437
438  function onPad(){...}
501
502  function drawLives() {
503      // count and display lives left
504      if (lives - livesLost != 0){
505      ctx.fillStyle = "white";
506      ctx.font = "30px Arial";
507      ctx.fillText("LIVES: " + (lives - livesLost), (canvas.width/2)-70, 525);
508      }
509  }
510
511  function draw(){
512      ctx.clearRect(0, 0, canvas.width, canvas.height);
```

Define and initialize the variables used in the drawLives() function.

```
90   var pad5 = false;
91   var pad6 = false;
92
93   var lives = 3;
94   var livesLost = 0;
95
96   document.addEventListener("keydown"
97   document.addEventListener("keyup",
98
99   function keyDownHandler(e)
100  {...}
```

Call for execution of the drawLives() function within your draw() function.

```
504
505  function drawLives() {
506      // count and display lives left
507      if (lives - livesLost != 0){
508      ctx.fillStyle = "white";
509      ctx.font = "30px Arial";
510      ctx.fillText("LIVES: " + (lives - livesLost), (ca
511      }
512  }
513
514  function draw(){
515      ctx.clearRect(0, 0, canvas.width, canvas.height);
516      drawLives();
517      drawBackground();
518      drawLogs();
519      moveLogs();
520      drawPads();
521      onPad();
522      drawFrog();
523      moveFrog();
524      drawCars();
525      moveCars();
526      runOver();
527      float();
528
529      requestAnimationFrame(draw);
530  }
531  draw();
```

Define a gameOver() function which will draw text as shown when the frog runs out of lives.

```
function drawLives() {
    // count and display lives left
    if (lives - livesLost != 0){
    ctx.fillStyle = "white";
    ctx.font = "30px Arial";
    ctx.fillText("LIVES: " + (lives - livesLost), (can
    }
}

function gameOver() {
    //end game if they run out of lives
    if (lives - livesLost == 0){
    play = false;
    ctx.fillStyle = "white";
    ctx.font = "72px Arial";
    ctx.fillText("GAME OVER", 0, 100);
    ctx.font = "28px Arial";
    ctx.fillText("Refresh to try again!", 50, 150);
    }
}

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

Define and initialize variables to be used in an if statement within your draw() function.

```
var pad3 = false;
var pad4 = false;
var pad5 = false;
var pad6 = false;

var lives = 3;
var livesLost = 0;
var play = true;
var victoryCondition = false;

document.addEventListener("keydown", keyDownHa
document.addEventListener("keyup", keyUpHandle

function keyDownHandler(e)
```

Add the If statement in the draw() function to determine whether the gameOver() and drawLIves() functions are executed.

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

    if (victoryCondition === false){
    gameOver();
    drawLives();
    }

    drawBackground();
    drawLogs();
```

Enclose the remaining functions of your draw() function within the if(play) statement as shown.

```
528  function draw(){
529      ctx.clearRect(0, 0, canvas.width, canvas.height);
530
531      if (victoryCondition === false){
532          gameOver();
533          drawLives();
534      }
535
536      if(play){
537          drawBackground();
538          drawLogs();
539          moveLogs();
540          drawPads();
541          onPad();
542          drawFrog();
543          moveFrog();
544          drawCars();
545          moveCars();
546          runOver();
547          float();
548      }
549
550      requestAnimationFrame(draw);
551  }
552  draw();
```

Now that are functions are set up, go back to the code segments which represent the frog losing a life. Add the highlighted line below in the float() function to add 1 to the variable livesLost.

```
                    logY8 <= y + height) {...}

      else if (y < 220 && y > 44){
          y = 488;
          livesLost ++;
          }
  }

  function drawPads(){...}

  function onPad(){...}
```

Add the highlighted line below in the runOver() function to add 1 to the variable livesLost.

```
function runOver (){

    var carsX = [carX1, carX2, carX3, carX4,
    var carsY = [carY1, carY2, carY3, carY4,

    for (i = 0; i < carsX.length; i++){
        if (carsX[i] <= x + width &&
        carsX[i] + carWidth >= x &&
        carsY[i] + carHeight >= y &&
        carsY[i] <= y + height) {
            y= 488;
            livesLost ++;
        }
    }
}

function drawLogs(){...}
```

Add the highlighted line below in the onPad() function to add 1 to the variable livesLost.

```
    else if (padX6 <= x + width &&
            padX6 + padWidth >= x &&
            padY6 + padHeight >= y &&
            padY6 <= y + height) {...}

    else if (y < 48){
        y = 488;
        livesLost++;
        }

    var pads = [pad1, pad2, pad3, pad4, pad5, pad
    var padsX = [padX1, padX2, padX3, padX4, padX
```

Add the victory() function shown below to display if all the pads have been reached by the frog.

```
509 ▶ function drawLives() {...}
517
518 function victory () {
519 if (pad1 && pad2 && pad3 && pad4 && pad5 && pad6){
520 //print "You won!" at (220, 488)
521 ctx.fillStyle = "white";
522 ctx.font = "30px Arial";
523 ctx.fillText("You won!", (canvas.width/2)-60, 525);
524 victoryCondition = true;
525 }
526 }
527
528 ▶ function gameOver() {...}
539
540 ▼ function draw(){
541       ctx.clearRect(0, 0, canvas.width, canvas.height);
542
543 ▼     if (victoryCondition === false){
```