

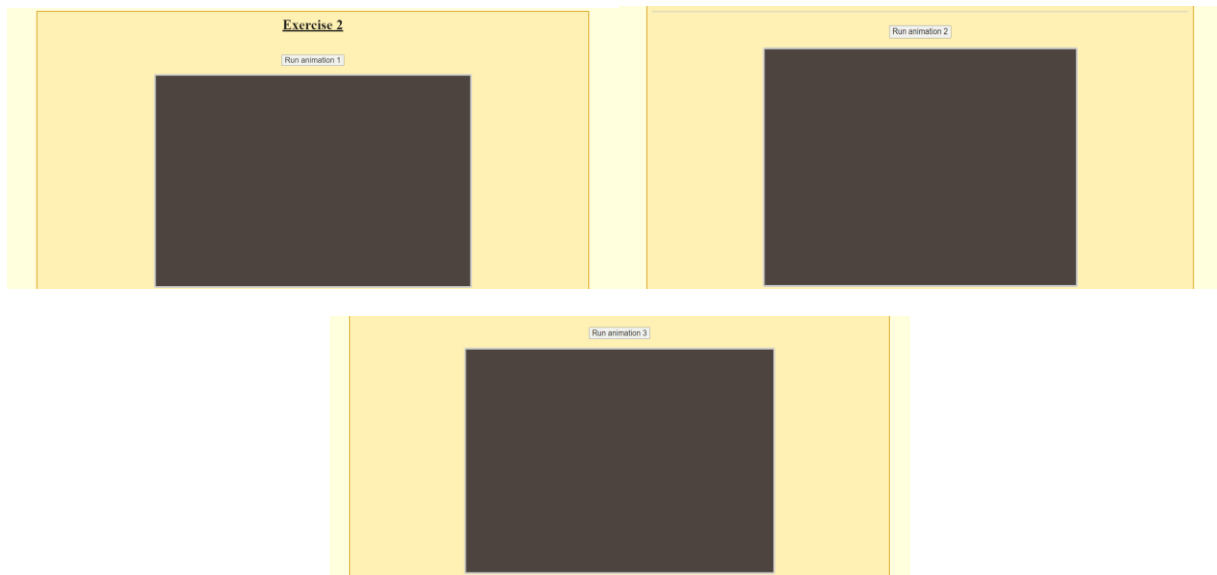
# Foes - Workshop 2

**Below are the detailed instructions intended to help guide you through the set-up and creation of Workshop 2. This workshop will help users focus solely on JavaScript and how it can be used to create animations on 2D objects. These concepts are meant to mimic some of the functions used to build the Foe's final game.**

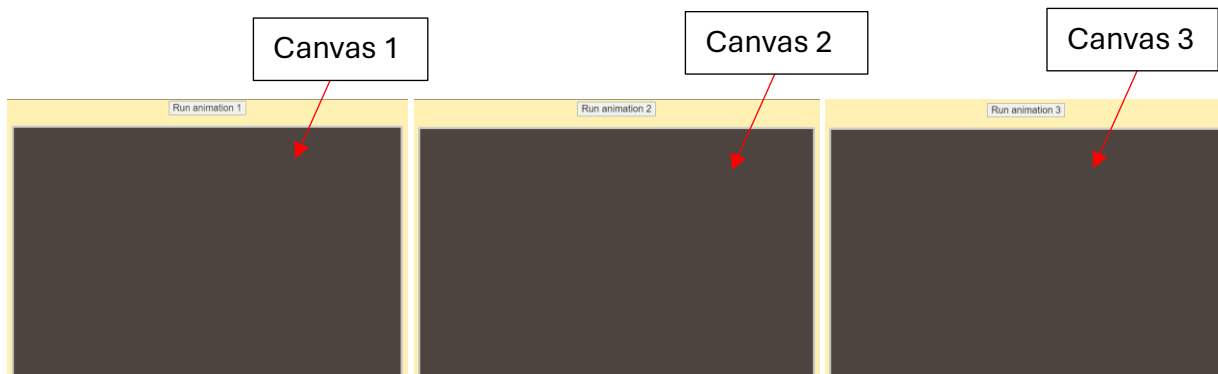
Important: Before starting the workshop, follow the instructions on the bottom of the Foes README.md section labeled “Workshop 1 and 2 via Cloning Repo” to properly gain access to the copy of Foes.

## Guided Steps:

1. In your chosen IDE open the Foes Repo.
2. To start, open the Workshop Starter Code folder and click the starter2.html file.
3. This is the file that contains Workshop 2's starter code. To view changes made in this workshop, open your file explorer, locate your Foes folder, and open the Workshop Starter Code folder. Under the folder you should see three files underneath.
4. Double tap the starter2.html file and you should be directed to a new webpage which will pull up your webpage. This will be the page we will be changing during the duration of this Workshop.



5. Here we will be working more with JavaScript to make objects animate in the three canvases on our webpage and as you scroll you will see the other canvases. Canvases are block areas where we will run animations.



6. During this workshop you will need to refresh your webpage constantly to see your changes.

7. For the first part of our workshop we will focus on animation 1.

8. Right now when we click the "Run animation 1" button a red box appears, but the red box has no movement.



9. Go back to your IDE and let's look at the code we will be using.

Our code can be broken into three main sections which all help to build our webpage. The first section is located at the top of the code in the `<style></style>` section. Everything under here is our CSS code, which is used to give our webpage an artistic feel through font-styles and a background-color. The first part of the `<body></body>` tags is where our HTML code is, giving the webpage its content, such as the paragraph we see in the middle of our

webpage. The last part of the code is contained in the `<script></script>` tags and give our webpage the dynamic feel allowing the user to click on buttons and so on.

10. The first part of the workshop will be to add some “motion” to our red box and to do so in the `<script></script>` tags find the method **runAnimationFrames()** on line 169. This function controls the animations for all three canvases.

11. To make the first box move we need to change some of the code under the first if statement.

```
169 function runAnimationFrames(){
170     if(animation1Running)
171     {
172         box1X += 1;
173
174         //ctx1.clearRect(0, 0, canvas1.width, canvas1.height);
175         ctx1.fillStyle = "red";
176         //Values: X, Y, width, height - where X and Y are the top left corner of the shape.
177         ctx1.fillRect(box1X, 10, boxwidth, boxHeight);
178     }
```

12. First we will change the augmented value of box1X to **box1X += 2.0;**

```
169 function runAnimationFrames(){
170     if(animation1Running)
171     {
172         box1X += 2.0;
173
174         //ctx1.clearRect(0, 0, canvas1.width, canvas1.height);
175         ctx1.fillStyle = "red";
176         //Values: X, Y, width, height - where X and Y are the top left corner of the shape.
177         ctx1.fillRect(box1X, 10, boxwidth, boxHeight);
178     }
```

13. By changing the value to 2, we are letting JS know that every time this method is called for the first animation, we are going to change the x position of the boxes' coordinate system by 2, thus allowing the box to “move” across the screen. So right now, the box is at the coordinates (20, 10) and when we call runAnimationFrames the X coordinate will change by 2. So, the new coordinates will become (22, 10). If we call the runAnimationFrames function again our coordinates would change from (22, 10) to (24, 10). This will repeat continuously for as long as the program continues to run.

14. When you are done press CTRL+S to save your changes, refresh your webpage and click the "Run animation 1" button again. What happened to the box?

15. Instead of moving across the screen the box just grows wider and wider.



16. To fix that go back to your starter 2 code and remove the two slashes on line 174 to uncomment our code.

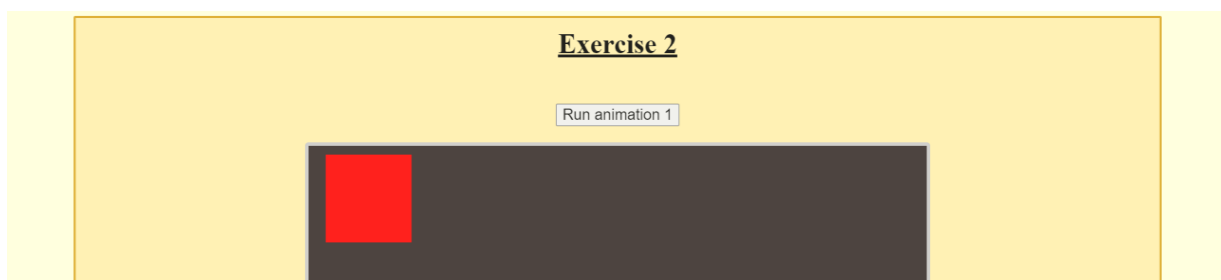
```

169     function runAnimationFrames(){
170         if(animation1Running)
171         {
172             box1X += 2.0;
173
174             //ctx1.clearRect(0, 0, canvas1.width, canvas1.height);
175             ctx1.fillStyle = "red";
176             //Values: X, Y, width, height - where X and Y are the top left corner of the shape.
177             ctx1.fillRect(box1X, 10, boxWidth, boxHeight);
178         }
  
```

```

169     function runAnimationFrames(){
170         if(animation1Running)
171         {
172             box1X += 2.0;
173
174             ctx1.clearRect(0, 0, canvas1.width, canvas1.height);
175             ctx1.fillStyle = "red";
176             //Values: X, Y, width, height - where X and Y are the top left corner of the shape.
177             ctx1.fillRect(box1X, 10, boxWidth, boxHeight);
178         }
  
```

17. Save your changes, go back to your webpage, and click the "Run animation 1" button again. Now what happened to the box?



## Exercise 2

Run animation 1



18. It moved! This is because of the clearRect JavaScript method. ClearRect constantly erases the previous location of the red box on our canvas, so that when we redraw the box it's as if the box is moving across the screen. So now every time we see the box it's in a new position.

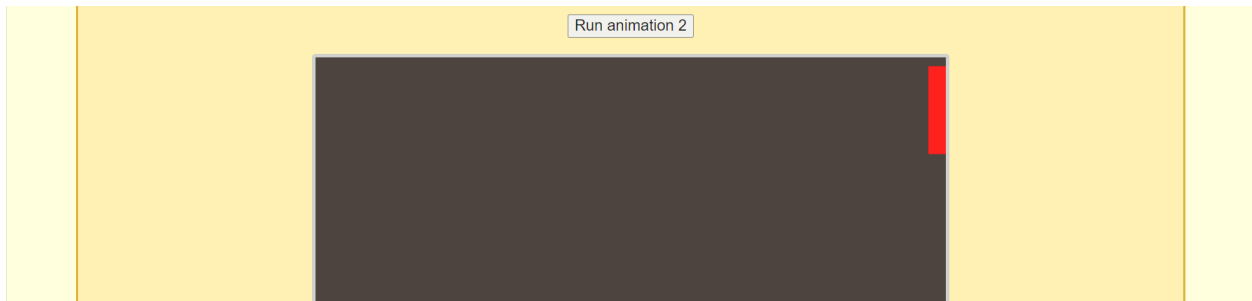
```
169 function runAnimationFrames(){
170     if(animation1Running)
171     {
172         box1X += 2.0;
173
174         ctx1.clearRect(0, 0, canvas1.width, canvas1.height);
175         ctx1.fillStyle = "red";
176         //Values: X, Y, width, height - where X and Y are the top left corner of the shape.
177         ctx1.fillRect(box1X, 10, boxwidth, boxHeight);
178     }
```

19. Let's move on to the second part of our workshop - animation 2.

20. Under our script tags scroll down to our second if statement under the runAnimationFrames method on line 180. This controls our second canvases motion.

```
180     if(animation2Running)
181     {
182         box2X += 5;
183
184         ctx2.clearRect(0, 0, canvas2.width, canvas2.height);
185         ctx2.fillStyle = "red";
186         ctx2.fillRect(box2X, 10, boxwidth, boxHeight);
187
188         if(box2X+boxwidth >= canvas2.width)
189             box2X = canvas2.width;
190     }
```

21. Click the "Run animation 2" button on your webpage and watch as a red box quickly moves across the page just like our first program. But now we want to make sure our box does not go off our canvas.



22. Under the function **runAnimationFrames** we are going to change some code.

23. Go back to line 188. Right now, our code wants the box to stop when the box2X coordinates becomes greater than or equal to canvas2.width, but the problem we run into is that the box is still appearing off the screen.

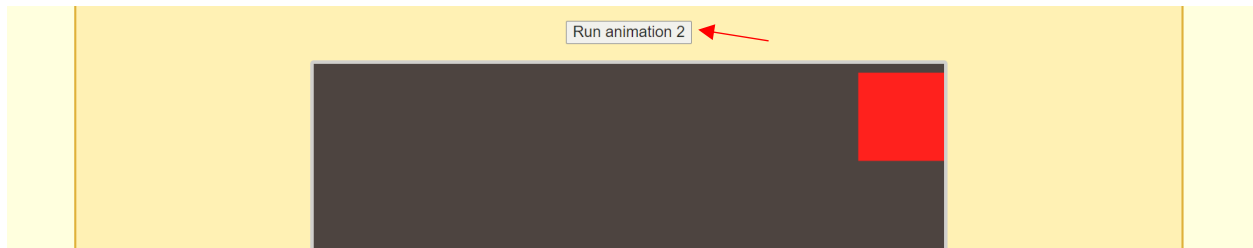
```
180     if(animation2Running)
181     {
182         box2X += 5;
183
184         ctx2.clearRect(0, 0, canvas2.width, canvas2.height);
185         ctx2.fillStyle = "red";
186         ctx2.fillRect(box2X, 10, boxWidth, boxHeight);
187
188         if(box2X+boxWidth >= canvas2.width)
189             box2X = canvas2.width;
190     }
```

24. So, we need to adapt our program so that the red box stops before going off the screen.

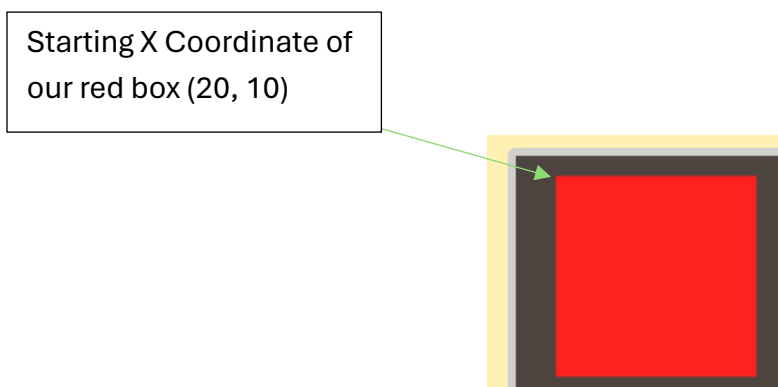
25. In the parentheses of our if statement after canvas2.width add (- **boxWidth**). This will help fix our problem of the box running off the screen.

```
180     if(animation2Running)
181     {
182         box2X += 5;
183
184         ctx2.clearRect(0, 0, canvas2.width, canvas2.height);
185         ctx2.fillStyle = "red";
186         ctx2.fillRect(box2X, 10, boxWidth, boxHeight);
187
188         if(box2X+boxWidth >= canvas2.width - boxWidth)
189             box2X = canvas2.width - boxWidth;
190     }
```

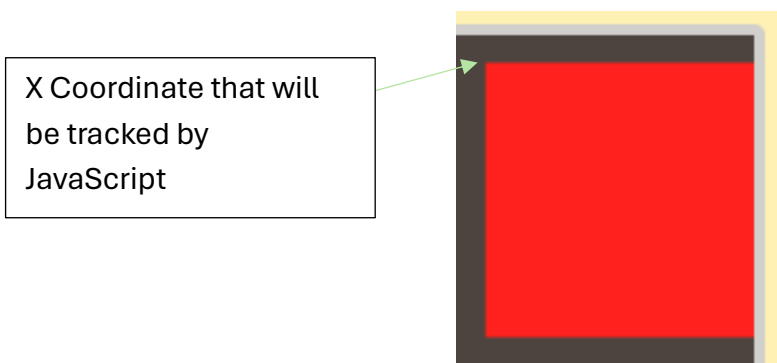
26. Save your changes, go back to your webpage, and click the "Run animation 2" button again. Now what happened to the box?



27. This is because when we created our box on the canvas, we didn't create it at the coordinates (0, 0) but instead at (20, 10), and the height and width of the box was created to be 100 pixels. When doing animations, we must consider not only the initial positions of our shapes but also the size of them.



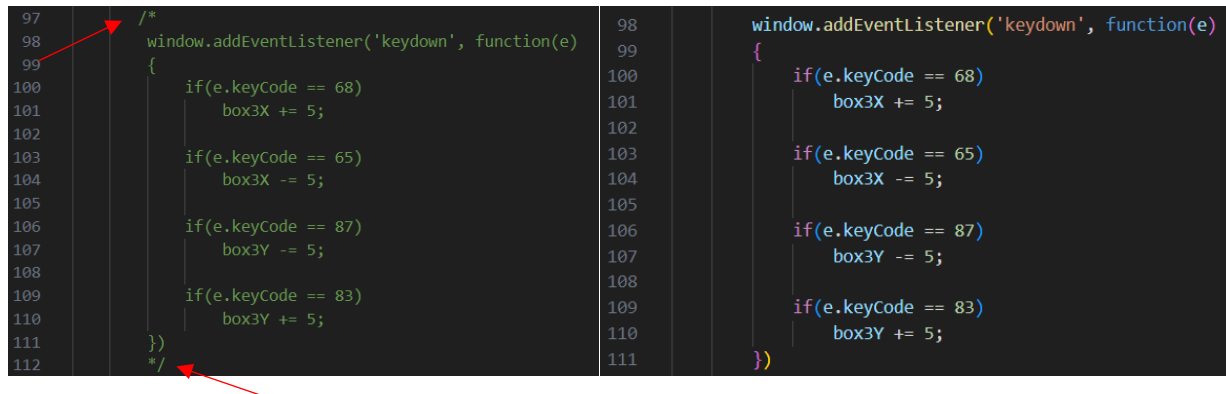
IMPORTANT: When we run an animation the x coordinate in the upper left corner is the one that matters the most to JavaScript. So, before we changed our code by the time our box2XWidth's equaled to canvas2.width the rest of our box was already off the page.



28. Lastly let's move on to the last part of our workshop, which allows the user to change the box's location using our keyboard.

29. On your webpage scroll to view the third animation and its canvas. Click the "Run animation 3" button. What happens? Nothing! This is because our third animation has no code to control it. Let's change that.

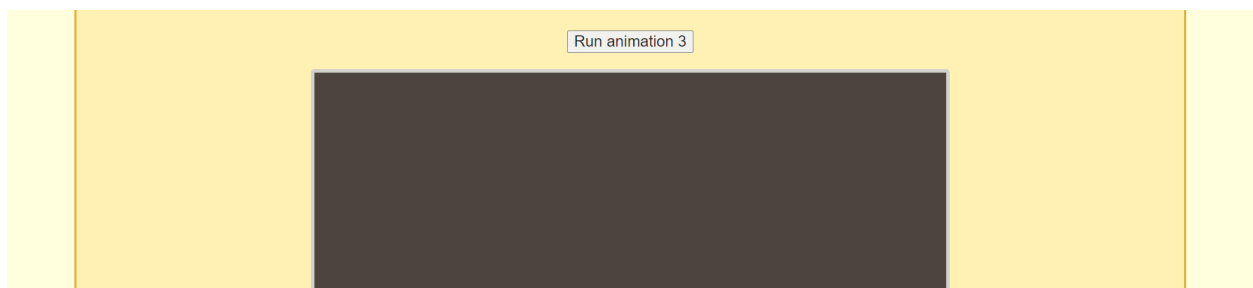
30. In our starter2.html code scroll to our `<script></script>` tags and find the huge portion of blocked/commented code. To uncomment it place your cursor on line 97 after the star and simply delete it. Do the same on line 112.



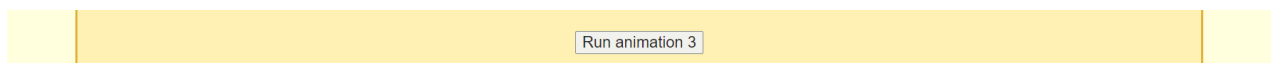
```
97  /*
98  window.addEventListener('keydown', function(e)
99  {
100      if(e.keyCode == 68)
101          box3X += 5;
102
103      if(e.keyCode == 65)
104          box3X -= 5;
105
106      if(e.keyCode == 87)
107          box3Y -= 5;
108
109      if(e.keyCode == 83)
110          box3Y += 5;
111  })
112  */

98  window.addEventListener('keydown', function(e)
99  {
100      if(e.keyCode == 68)
101          box3X += 5;
102
103      if(e.keyCode == 65)
104          box3X -= 5;
105
106      if(e.keyCode == 87)
107          box3Y -= 5;
108
109      if(e.keyCode == 83)
110          box3Y += 5;
111  })
```

31. Once your code is fully commented out. Save your changes then scroll your webpage to show the button “Run Animation 3” and its canvas.



32. Click the “Run animation 3” button on your webpage.

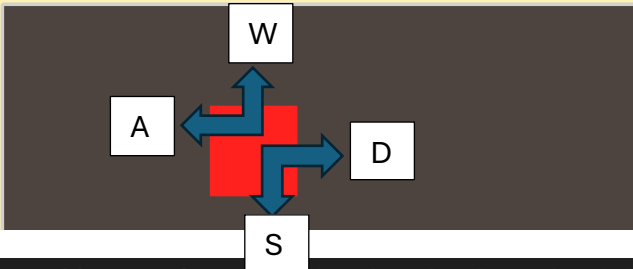


33. To see changes reflected on the canvas you must interact with the webpage by pressing the keys: W, A, S, and D on your keyboard. What do each of the keys do to the box? They move it!

34. The code we uncommented in our code uses the `addEventListener` method which allows us to specify the action we want JavaScript to "listen" for, in this case it's **'keydown'**. Under this code we specify what we want the code to do once they keys are pressed. By pressing the keys JS will either add or subtract 5 from the `box3X` and `box3Y` coordinates allowing the box to move with user interaction.



Run animation 3



A diagram showing a red square box in the center of a dark gray area. Four blue arrows point outwards from the box: up, down, left, and right. Above the box is a white box with the letter 'W', below is 'S', to the left is 'A', and to the right is 'D'. The entire diagram is set against a light yellow background.

```
98 window.addEventListener('keydown', function(e)
99 {
100     if(e.keyCode == 68)
101         box3X += 5;
102
103     if(e.keyCode == 65)
104         box3X -= 5;
105
106     if(e.keyCode == 87)
107         box3Y -= 5;
108
109     if(e.keyCode == 83)
110         box3Y += 5;
111 }
```

A code editor snippet with line numbers 98 to 111. Red arrows point from labels 'A', 'D', 'W', and 'S' to specific lines of code: 'A' points to line 104, 'D' points to line 101, 'W' points to line 107, and 'S' points to line 110.

35. Now you try changing the amount we subtract/add and note the changes you see.

36. Next try changing the keycode values so that the user can press different keys to make the box go up, down, left, and right by using this website:

<https://www.exeideas.com/2014/05/javascript-char-codes-key-codes.html>

37. Happy coding!