

ISCG6420 Semester 1 2024

Exercise: Animating with JavaScript

This exercise will take you through creating animations with JavaScript as an alternative to CSS animations.

A JavaScript-based animation is required for Assignment 1 Part 2. The learning outcomes from this exercise will prepare you for this assignment task.

Prepare the animation surface

Open a new or existing project in your code editor. Create a new webpage with the html:5 template:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

In the <body>, create a <div> with ID: animWrapper.

Inside animWrapper, create a child <div> with ID: animBlock.

```
<body>
  <div id="animWrapper">
    <div id="animBlock"></div>
  </div>
</body>
```

Create a CSS file and link it to the HTML file:

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>

  <link rel="stylesheet" href="./style.css">
</head>
```

Add the following styles to the CSS file:

```
#animWrapper {  
    /* Position wrapper in centre */  
    display: block;  
    margin-left: auto;  
    margin-right: auto;  
  
    /* Child positioning and visibility */  
    position: relative;  
    overflow: hidden;  
  
    /* Wrapper size and style */  
    width: 800px;  
    height: 600px;  
    border: 2px solid black;  
}  
  
#animBlock {  
    /* position with Top and Left */  
    position: absolute;  
  
    /* Block size and style */  
    width: 30px;  
    height: 30px;  
    background-color: green;  
}
```

Save and open the HTML file in a web browser. It should appear like this:



Animating with JavaScript

Create a JS file and link it to the HTML at the bottom of the <body>

```
    </div>
    <script src="adAnimation.js"></script>
</body>
```

inside the JS file, create a function called "animate".

Inside the function, create 6 variables:

```
function animate() {
  const element = document.getElementById("animBlock");
  let x = 0;
  let y = 0;
  let opacity = 1;
  let phase = 0;
  let animId;
```

const element is a reference to the green block <div>.

X and Y are numbers to represent the position of the <div>

Opacity is a number to represent the opacity style value of the <div>

Phase is a number that will determine which code block we run

AnimId has no value yet, but will hold the ID of an interval timer

Inside the animate() function, under the variables, create a new function called "doAnimation".

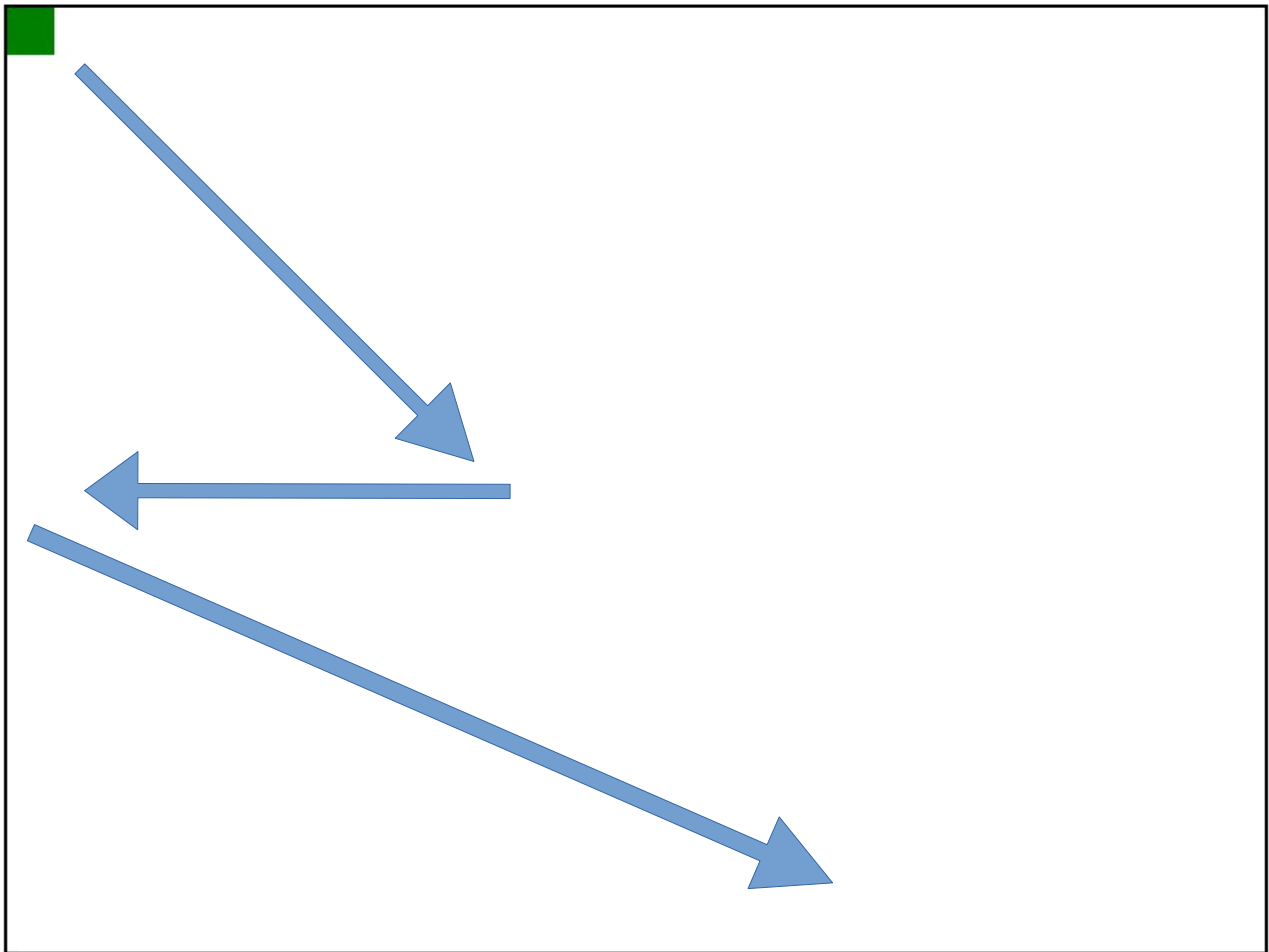
```
    let animId;

    function doAnimation() {

    }
```

Conditional code

Our animation will consist of the green square moving along multiple linear paths sequentially, like so:



Each path of travel will be represented by a **phase** during the animation code.

Inside the inner function, create an if statement to check if **phase** is equal to 0. If it is, do the following steps:

1. Increase X and Y by 1
2. Set the element Top and Left styles to X / Y pixels
3. Check if the value of X is at least 300
 - a. If it is, set the phase to 1

In code, this can be implemented like so:

```

    if (phase == 0) {
        x++;
        y++;

        element.style.left = x + 'px';
        element.style.top = y + 'px';

        if (x >= 300) {
            phase = 1;
        }
    }

```

Create 2 **else if** blocks after the above if block. Set their conditions and code to the following:

Else if (phase is equal to 1)

1. Decrease X by 1
2. Set the element Left to X pixels
3. Check if X is at most 20
 - a. If it is, set the phase to 2

Else if (phase is equal to 2)

1. Increase X by 1
2. Increase Y by 0.5
3. Set the element Left to X pixels
4. Set the element Top to Y pixels
5. Check if X is at most 550
 - a. If it is, set the phase to 3

```

    else if (phase == 1) {
        x--;
        element.style.left = x + 'px';
        if (x <= 20) {
            phase = 2;
        }
    }
    else if (phase == 2) {
        x++;
        y += 0.5;
        element.style.left = x + 'px';
        element.style.top = y + 'px';
        if (x >= 550) {
            phase = 3;
        }
    }

```

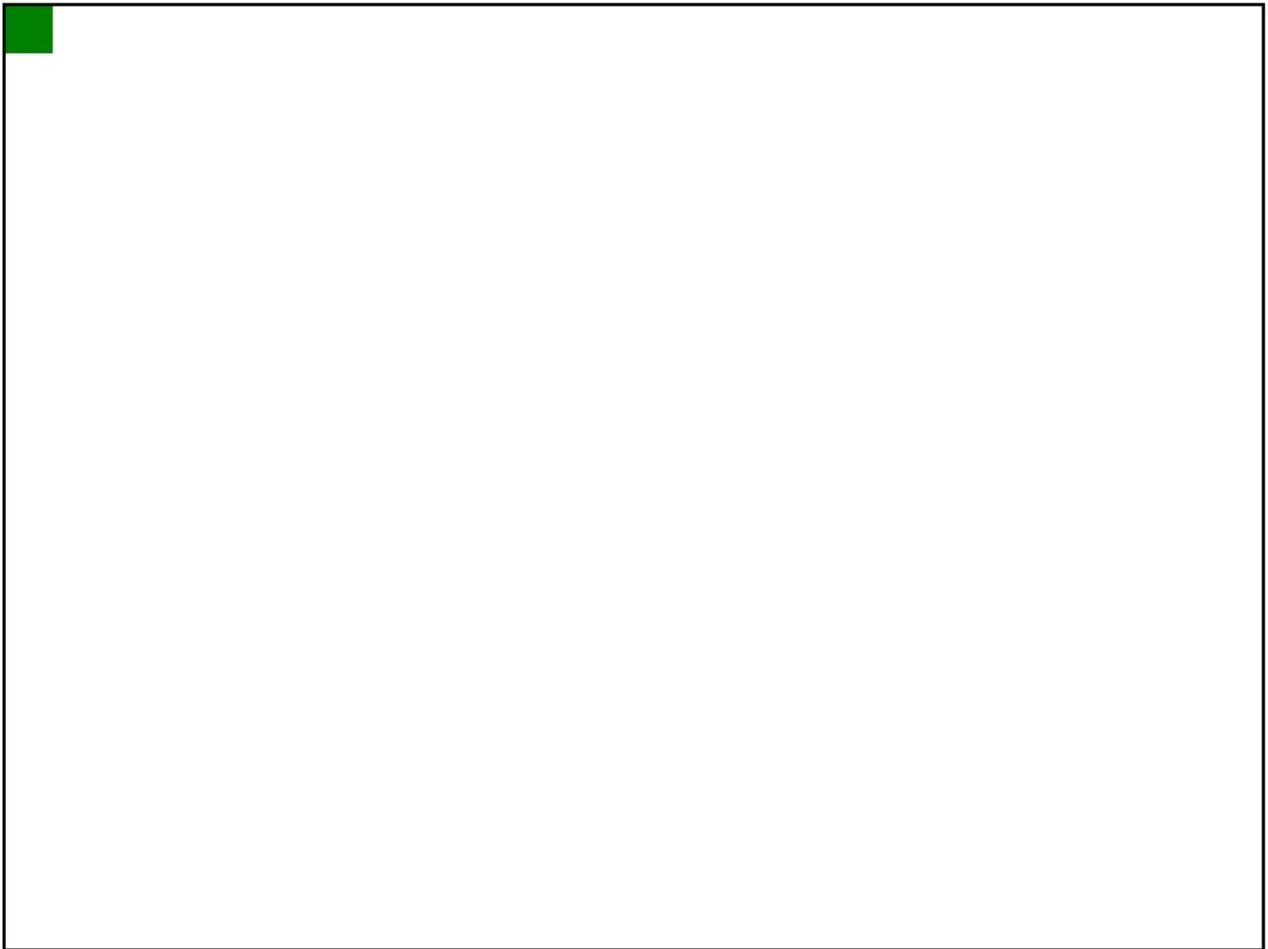
Invoking code

With the animation code complete, the webpage needs a way of calling the code. This can be achieved using an event listener.

At the bottom of the JS file, outside the functions, create an event listener linking the “click” event of the animWrapper with the animate function:

```
document.getElementById("animWrapper").addEventListener("click", animate);
```

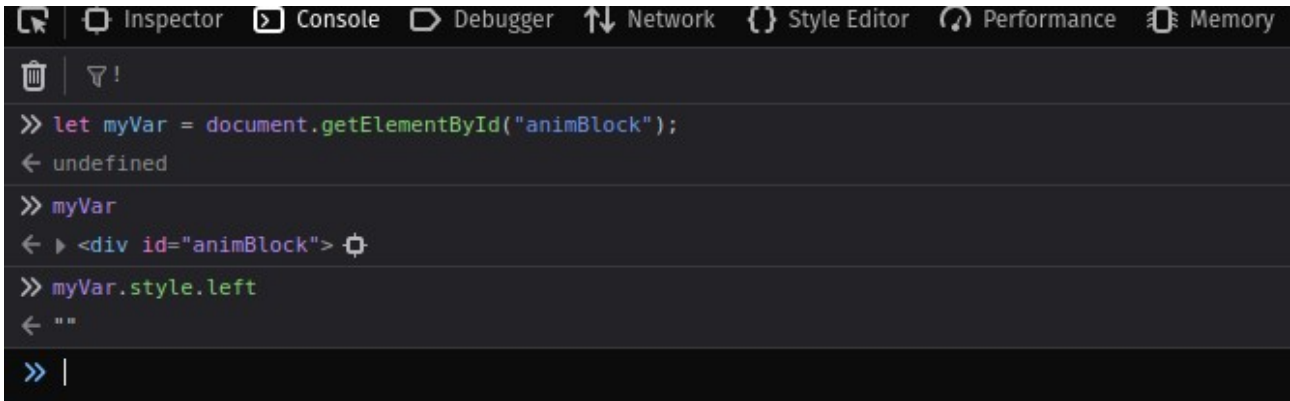
Save and run the page to test the animation.



Something is wrong. The animation doesn't seem to be running.

Debugging

Open the web browser Developer Tools console. In the console, create a variable that references the green <div>. Print the variable to the console, and print the **style.left** value to the console:



```
>> let myVar = document.getElementById("animBlock");
< undefined
>> myVar
< <div id="animBlock">
>> myVar.style.left
< ""
>> |
```

The console output shows the div is there, but it has no value for **style.left**. This indicates our code for phase 1 hasn't run yet.

This is because our function `animate()` has been called, but the function inside it, `doAnimation()`, hasn't been called. For the animation to work, `doAnimation()` needs to be called repetitively.

Repetitive tasks

Running code repeatedly can be done by creating an Interval Timer. An Interval Timer takes the name of a function and an interval time (in milliseconds), and will call the function after each interval delay indefinitely.

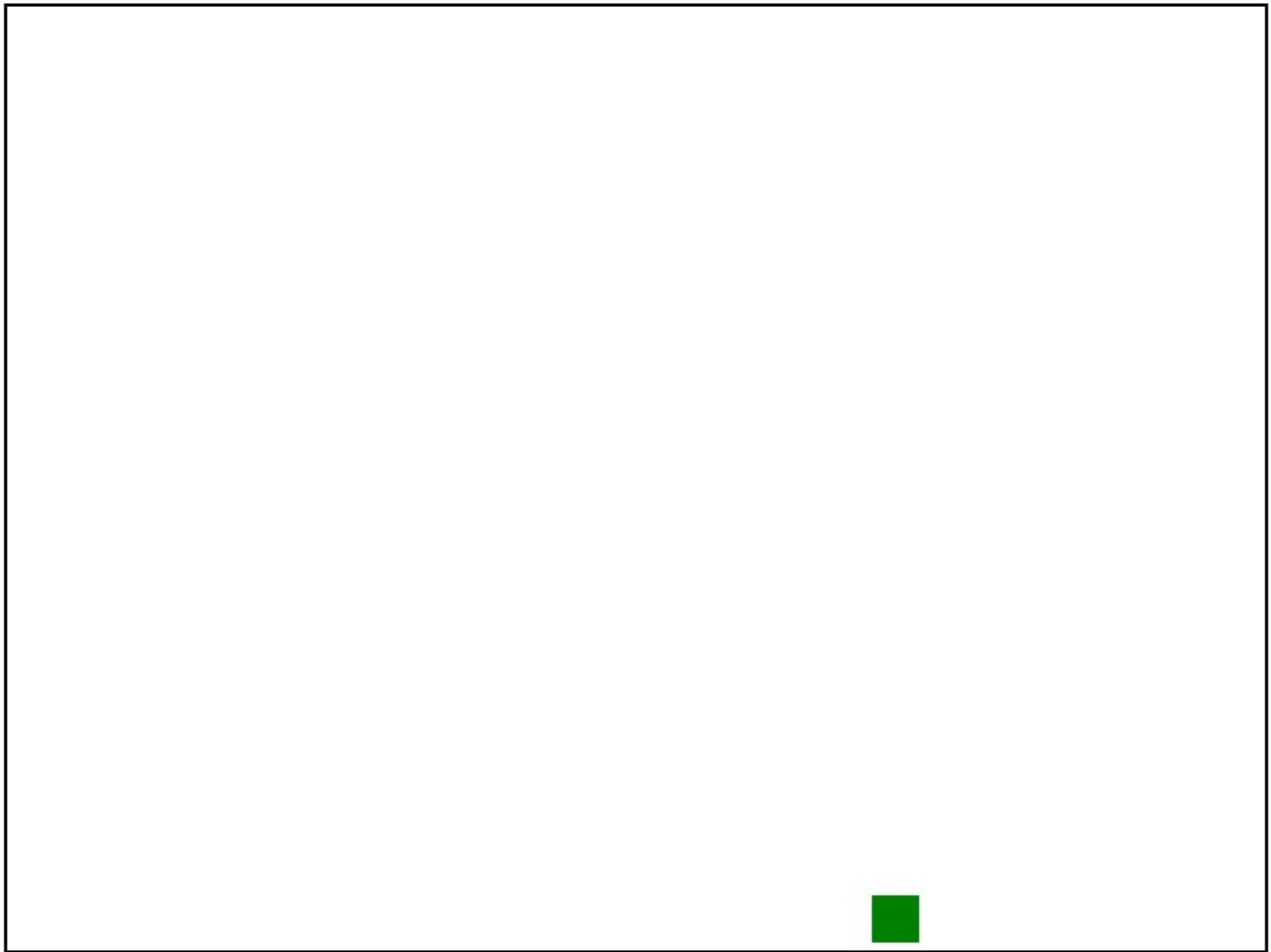
At the bottom of the `animate` function, create an Interval timer that calls "doAnimation" every 10 milliseconds. Set the value of the interval to "animId":

```
    } // doAnimation block end
    animId = setInterval(doAnimation, 10);
} // animate block end
```

To stop the interval running after our animation finishes, clear the interval with its ID in the ending condition of phase 2:

```
    else if (phase == 2) {
        x++;
        y += 0.5;
        element.style.left = x + 'px';
        element.style.top = y + 'px';
        if (x >= 550) {
            phase = 3;
            clearInterval(animId);
        }
    }
```

Save and test the changes:



The green div should now follow the paths and end at the bottom near the right-side.

Next steps

This webpage can be used for testing or creating more complex animations. Try implementing the following changes:

- Add another phase, and adjust the div opacity to fade out
- Add a second element to the animation and follow a different path
- Loop the animations by resetting at the end of each animation pass
- Modify the div to display text or an image
- Combine JS animations with CSS animations

Exercise complete.