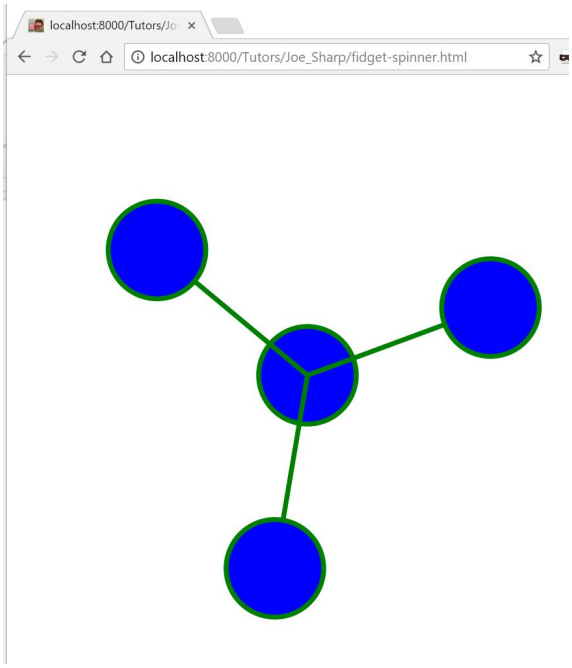
JavaScript SVG Fidget Spinner

They are all the rage at the moment, everywhere you look: Fidget Spinners. We are going to create one in a web page. A real fidget spinner looks like this:

Our one will look like this:



This project will show you how to do two important things:

- Group together SVG elements to create composite graphics.
- React to button presses and mouse events and drive the animation.

First we must create our HTML file.

Create HTML file

I suggest that you take a copy of the SVG game template found under

/KidsProjects/Exercises/JavaScript/svg-game-template.html

This should give you the following:

fidget-spinner-tutorial.html | fidget-spinner.html

```
<html>
  <head>
    <script type='text/javascript'>

window.onload = function() {
    // Get the SVG element
    const svgMain = document.getElementById('svgMain');
    const svgMainRect = svgMain.getBoundingClientRect();

    // Required to create new SVG elements
    const SVG_NS = 'http://www.w3.org/2000/svg';

    console.log('SVG Found ' + svgMainRect);

    //
    // Your code goes here
    //

};
    </script>
  </head>
  <body>
    <svg id='svgMain' width='100%' height='100%'>
      <!-- Custom SVG goes here -->
    </svg>
  </body>
</html>
```

We will be building our fidget spinner inside the <svg> tag near the bottom. We can give the various tags **id** values that our JavaScript can use to maniplate them.

Time to start drawing!

Draw the Fidget Spinner Centre

This work will all go inside the <body><svg> tag. It looks like this:

```
22      <body>
23        <svg id='svgMain' width='100%' height='100%'>
24          <!-- Custom SVG goes here -->
25        </svg>
26      </body>
```

First add a <g> element. This is an SVG grouping element which will contain all the parts required to build our spinner. We must give the <g> element a value for **id** for use later.

```
23      <svg id='svgMain' width='100%' height='100%'>
24        <!-- Custom SVG goes here -->
25        <g id='fidgetSpinner' stroke=green fill=blue stroke-width=5>
26
27        </g>
28
29      </svg>
```

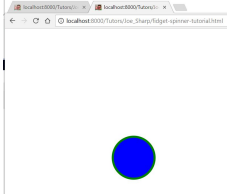
I have also taken this opportunity to specify the colours and width of the pen strokes.

The spinner has a central element, we will use a circle. Add the following.

```
25        <g id='fidgetSpinner' stroke=green fill=blue stroke-width=5>
26          <!-- Central Element-->
27          <circle cx=300 cy=300 r=50 />
28
29        </g>
```

Check it Works:

Load this page up in a browser and you should have a single circle like this:



Draw the first Orbitals

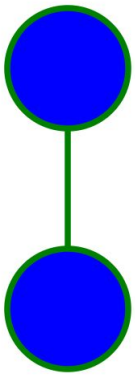
Each orbital element will be made up of a circle, and a line that joins the orbital to the centre.

We will define the orbitals before the centre element so that the lines go underneath both circles. This looks better. To add the first orbital, add the code from lines 26 to 28. Note that I have put them **before** the central element.

```
25      <g id='fidgetSpinner' stroke=green fill=blue stroke-width=5>
26        <!-- Orbital 1 -->
27        <line x1=300 y1=300 x2=300 y2=500 />
28        <circle cx=300 cy=500 r=50 />
29
30        <!-- Central Element-->
31        <circle cx=300 cy=300 r=50 />
```

Check it Works:

Refresh your page, the spinner should look like this:



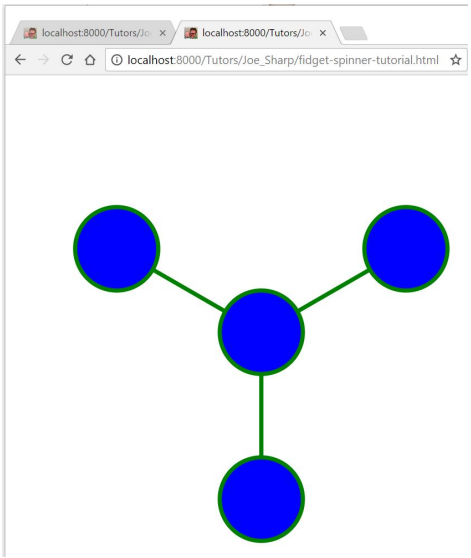
Draw the Other Two Orbitals

Add the other two orbital elements, you can do a lot of copying and pasting, then just change the values of cx/cy/x2/y2 as required.

```
30          <!-- Orbital 2 -->
31          <line x1=300 y1=300 x2=126 y2=200 />
32          <circle cx=126 cy=200 r=50 />
33
34          <!-- Orbital 3 -->
35          <line x1=300 y1=300 x2=474 y2=200 />
36          <circle cx=474 cy=200 r=50 />
```

Check it Works:

Refresh your browser, the spinner should look like this:



Get it Spinning

Now that our drawing is complete, we should get it rotating. First we will spin it at a constant speed with no input from the user. This will establish how to animate the frame. Later we can modify the spin based on user input.

The following code will all go inside the JavaScript part of this page. The template gives you a section that looks like this:

```
5   window.onload = function() {
6       // Get the SVG element
7       const svgMain = document.getElementById('svgMain');
8       const svgMainRect = svgMain.getBoundingClientRect();
9
10      // Required to create new SVG elements
11      const SVG_NS = 'http://www.w3.org/2000/svg';
12
13      console.log('SVG Found ' + svgMainRect);
14
15      //
16      // Your code goes here
17      //
18
19  };
```

Take note of the comment lines, your code will start after those, but keep it inside the onload function.

First we must get a reference to the spinner <g> element. Add the following line.

```
16      // Your code goes here
17      //
18      const fidgetSpinner = document.getElementById('fidgetSpinner');
```

Create variables to store the current angle and speed of rotation.

```
20      var spinSpeed = 0.2;
21      var angle = 0;
```

Write a function called **animateFrame** and call it every 20 milliseconds using the built-in **setInterval** function.

```
23      function animateFrame() {
24      }
25
26      setInterval(animateFrame, 20);
```

In our **animateFrame** function we will calculate the **transform** to apply to our spinner. We will use the **rotate** function, but we must also specify the centre point of the rotation. This will be the location of our central element x=300, y=300. Write the function as shown:

```
23      function animateFrame() {
24        angle += spinSpeed;
25        var transform = 'rotate(' + angle + ',300,300)';
26        fidgetSpinner.setAttribute('transform', transform);
27      }
```

Check it Works:

Refresh your browser page and your spinner should now slowly rotate. You can play around with the value of **spinSpeed** to see how it changes the behaviour.

Apply Friction

As a last step, the spinner should gradually slow down if left to its own devices. Add line 25 and then refresh your browser to see if the spinner gradually stops.

```
24        angle += spinSpeed;
25        spinSpeed *= 0.995;
```

Use the Mouse

We will be detecting mouse gestures to kick the spinner off. I have devised the following rules:

- A mouse swipe upwards will result in anti-clockwise rotation
- A mouse swipe downwards will result in clockwise rotation

The speed of rotation will be governed as follows:

- The time between the mouse down and up events, shorter time = quicker spin
- The distance travelled (vertical only for simplicity), longer distance = quicker spin

To measure these two differences, we need a couple of variables. Add the following after the call to **setInterval** (but inside the **onload** function).

```
30      setInterval(animateFrame, 20);
31
32      // Using the mouse - detect start points
33      var timeMouseDown = 0;
34      var mouseDownPosY = null;
```

We need to populate these variables when the mouse click goes down. Add the following event handler to do this.

```
36      svgMain.onmousedown = function(e) {
37        mouseDownPosY = e.clientY;
38        timeMouseDown = new Date().getTime();
39      };
```

When the mouse button is released, these values will be used to calculate the new spin speed. Add the following event handler for mouse up.

```
41      svgMain.onmouseup = function(e) {
42        const mouseUpPosY = e.clientY;
43        const timeMouseUp = new Date().getTime();
44
45        const travel = mouseUpPosY - mouseDownPosY;
46        const timeDiff = timeMouseUp - timeMouseDown;
47        spinSpeed = 10 * (travel / timeDiff);
48      };
```

Check it Works:

Refresh your browser, the spinner will have started off spinning. If you click swipe up or down with the mouse, upon releasing the mouse key the spin speed should change.

Use the Keyboard

This will use similar rules to the mouse:

- Pressing the RIGHT arrow key results in clockwise rotation
- Pressing the LEFT arrow key results in anti-clockwise rotation

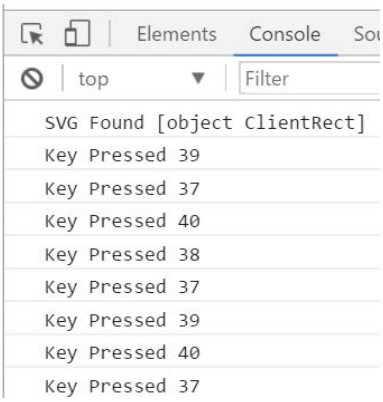
We will measure the time between key presses to govern speed.

We need a variable to keep track of the last key press time, and we will intercept the **keydown** event on the **window**. Add the following code:

```
50      // Using key presses
51      var lastKeyTime = 0;
52      window.onkeyup = function(e) {
53        console.log('Key Pressed ' + e.keyCode);
54      }
```

Check it Works:

Refresh your browser, ensure Developer Tools are open at the **Console** window. Press some keys and you should see messages like this:



Handle Arrow Keys

Measure the time gap between key presses using the following code inside our event handler (lines 54 to 56 are new):

```
52      window.onkeyup = function(e) {
53        console.log('Key Pressed ' + e.keyCode);
54        const now = new Date().getTime();
55        const diff = now - lastKeyTime;
56        lastKeyTime = now;
57      }
```

If you observed the key codes being printed, you should have seen that the RIGHT arrow is keyCode 39 and the LEFT arrow key is keyCode 37.

We will use a switch statement to decide which key was pressed and react accordingly. I am using a fiddle factor of 5000 to get the numbers to line up sensibly.

```
15    window.onkeyup = function(e) {
16      const now = new Date().getTime();
17      const diff = now - lastKeyTime;
18      lastKeyTime = now;
19
20      switch (e.keyCode) {
21        case 39: // RIGHT arrow
22          spinSpeed = (+5000 / diff);
23          break;
24        case 37: // LEFT arrow
25          spinSpeed = (-5000 / diff);
26          break;
27      }
28    }
```

Check it Works:

Refresh your browser, check for errors in the **console** and then try pressing the RIGHT/LEFT arrow keys on your keyboard. You should be able to drive your spinner clockwise and anti-clockwise.

CONGRATULATIONS - YOU HAVE FINISHED THE FIDGET SPINNER