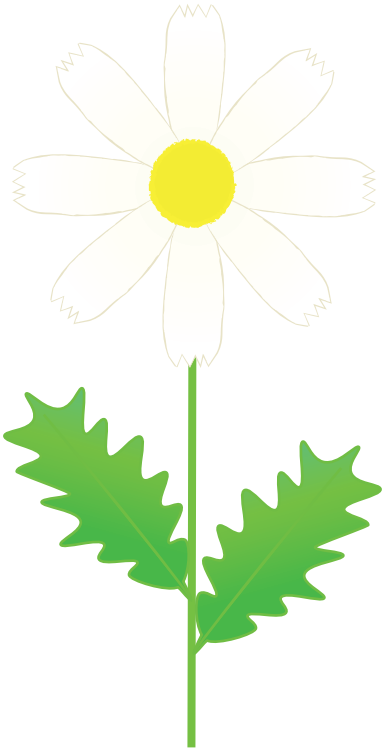
lesson 4

REUSABLE CONTENT

the begining

the end



In the first module, you can see the flower that has only one leaf and only one petal as it described on the image 1. You have to fix the code to create the flower that you can see on the image 2.

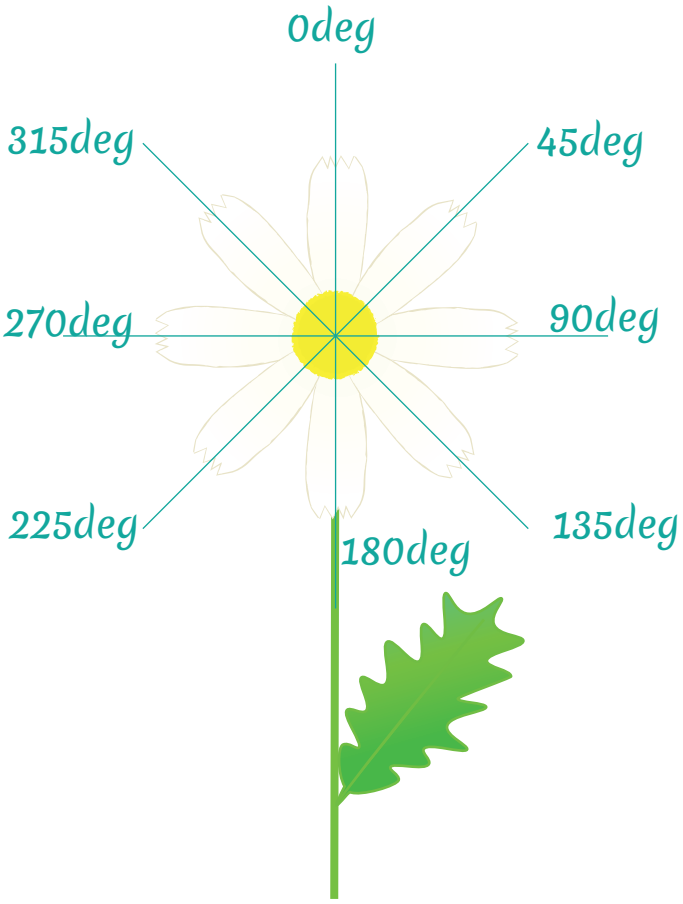
1.1 You can see the group element with id="petal" that contains the path. You should use the petal for 8 times.

So you should make this content reusable to do so you should wrap the g element with id= "petal" by the defs element. Let's check our image in the browser. The petal disappears because all elements that are inside the defs element have the dislplay property that set to none. From now on, You can use the group with the id="petal" to create the petals.

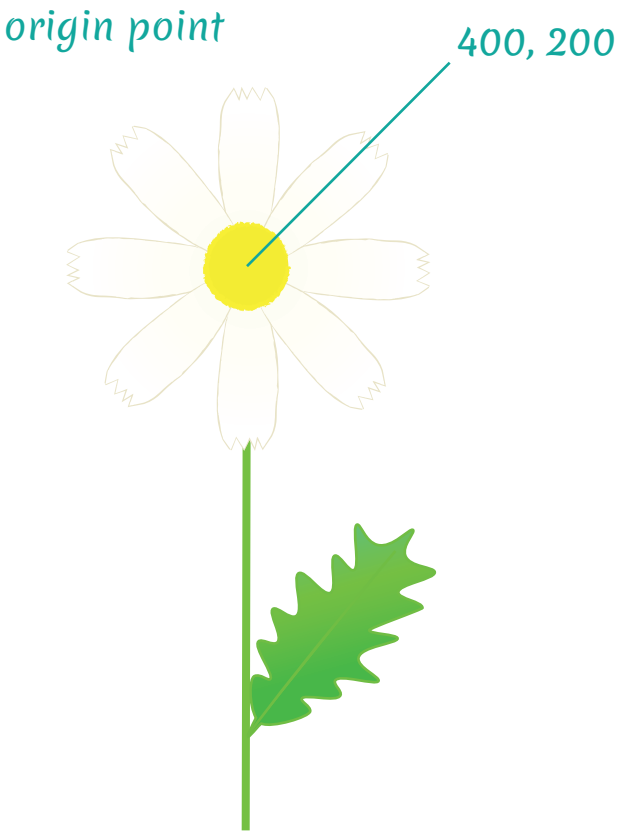
1.2 To do so you should add eight use elements after the line element with id="trunk" and add to each use element the href attribute with the link to the referenced element. The value of the href should be equal to "#petal" that will mean that we should use an element with id="petal" that is inside the current document, in this case, it's the g element that contains petal image.

lesson 4

REUSABLE CONTENT



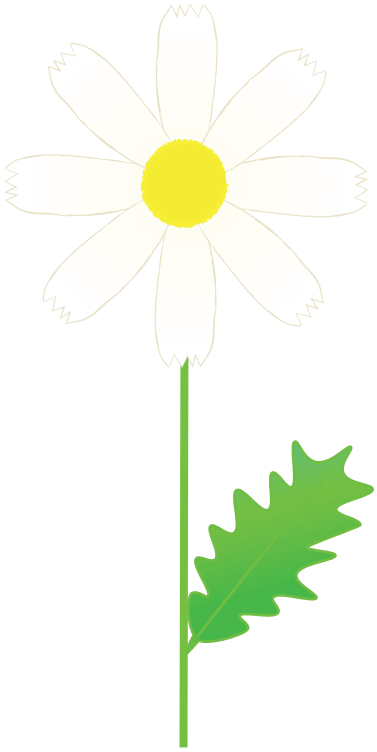
1.3 As you may have noticed It looks like there is only one petal, but it isn't so, actually the petals lay one on top of another. That's not we want, so you should apply transform="rotate()" function to seven use elements to create the flower of the chamomile.
If you are familiar with rotate function in CSS you should know that it works differently in SVG, that means that the rotate function accept three parameters: <angle>, [< x>,<y>]. The x and y parameters set the coordinates of the origin point, that could be anywhere on the local coordinate system, unlike in the CSS, where the origin-point(transform-origin property) can be only within the boundaries of that particular element.
So, add the transform="rotate()" function with the parameters from the images to the seven use elements. to get the result from the images.



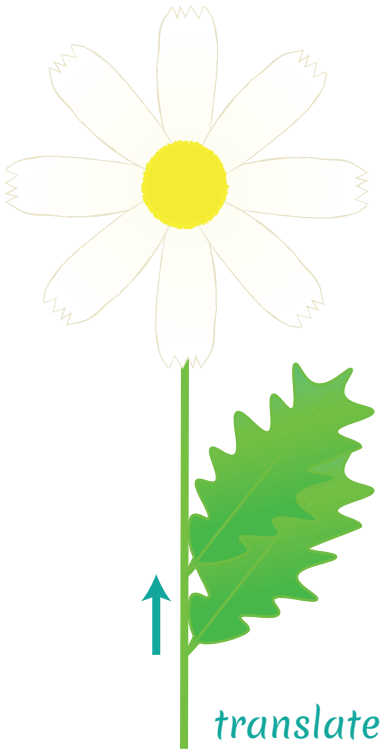
lesson 4

REUSABLE CONTENT

1

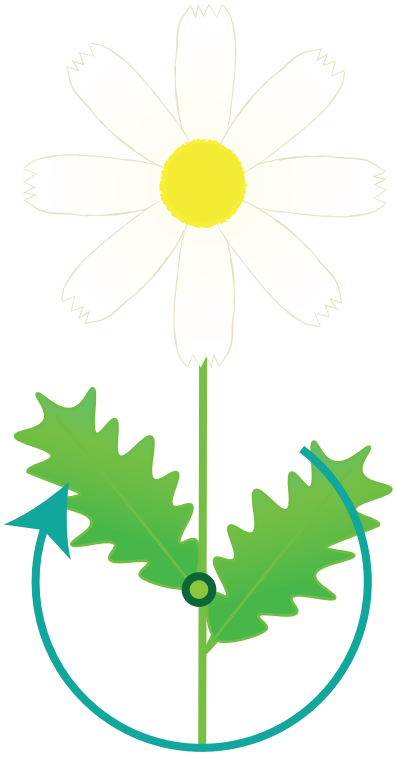


2



translate(0, -50)

3

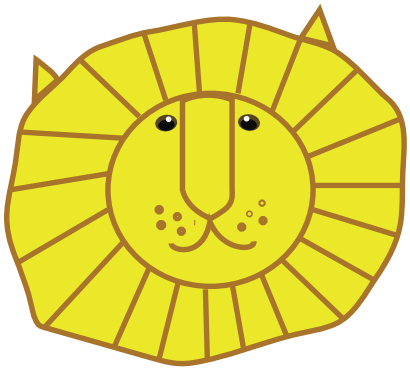


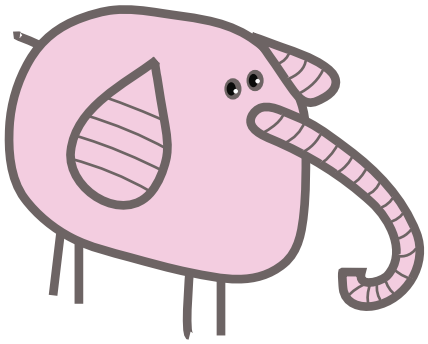
rotate(270, 400, 375)

1.4 Now do the same with a leaf. Wrap g element with id="leaf" by defs element, then add two leaves by adding two use elements with a href attribute that referencing to g element with an id="leaf".
Then apply the transform translate function with the values from the second image and then apply the rotate() function from the third image. The result of the transform function should be the following: transform="translate(values) rotate(values)"
Now we get the result we want.

lesson 4

REUSABLE CONTENT





In the second module, you can see two images. The first one has wrapped by the g element with an id="lion" that is inside the defs element. The second one is inside the symbol element with an id="elephant". The symbol element creates its local coordinate system, it has a viewBox and a viewport so we can consider it as a never-rendered svg element with some additional options.
2.1 So let's create the instance of each image by using the use element. Now we can see both images. So let's imagine that we need to move both elements down to 200 pixels. So we can use the transform="translate()" function to both element.
Make a conclusion. Does the transform="translate()" function works for both images?

Write your answers in the fields

2.2 And now, just imagine that you want to place both of these images in a particular position. We know that translate() function can help, but to use transform="translate()" function we should do a lot of calculation. To find the left top point of the bounding box(we will cover this subject in Section 6) and then to calculate the offset from that point. But it's a lot of unnecessary work. we could do it in a more simple and elegant way. Let's place the elephant to the point (x="500" y="300") by adding x and y geometry property to the symbol element the same way we did it in the previous chapter with nested svg elements.
Make a conclusion. Will the x and y properties affect the element position?

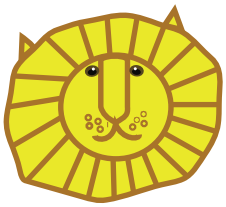
Write your answers in the fields

g element

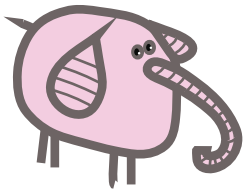
symbol element

lesson 4

REUSABLE CONTENT

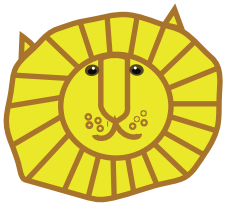


To move elements we've applied the transform="translate()" function. Actually, the attributes of the use element allow us to supersede the transform function. We could add additional transformation by adding x and y attributes. You should remember that unlike the x and y geometry properties of the svg and symbol elements that set the position of the element, the x and y attributes of the use element works the same if we add the transform=" translate()" function with x and y as the parameters

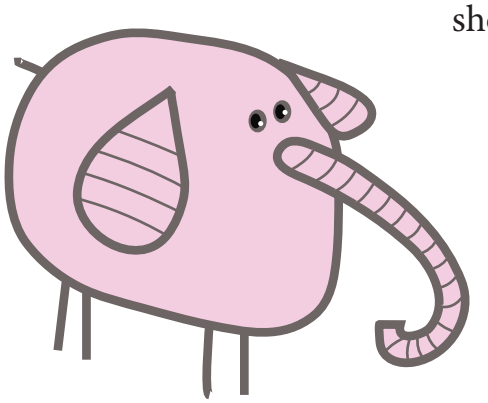


2.3 Delete the transform=" translate()" function from the g element with id="lion" and add x and y attributes to the use element that referencing on the g element. The result shouldn't change.

Notes! That the x, y geometry propertyes have different behavior for Firefox and for the others browsers. They add additional transformation matrix for the elements that do not create the viewport. if the elements create the viewport then firefox still adds the additional transformation but the others browsers will reset the x and y geometry propertyies. The Opera, Chome, Edge ignore the x and y values of the symbol element so <use> element uses the default value of it that are equal to 0,0. Actually, we should reasign the x,y, height and width properties inside the use element to get the proper result in Chrome, Edge and Opera



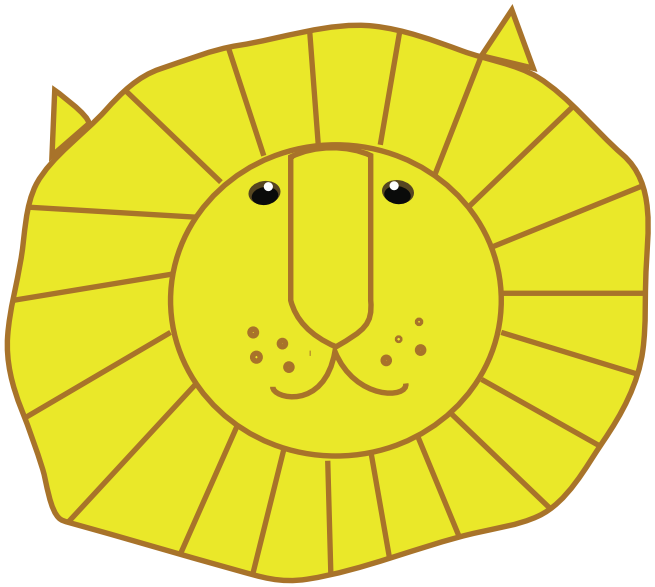
Moreover, the use element allows to redefine the viewport of the elements that have a viewport(svg, symbol). Actually this works properly only in Firefox. You should define the height and width propeties manually inside the use element if you want cross-browser compatibility.

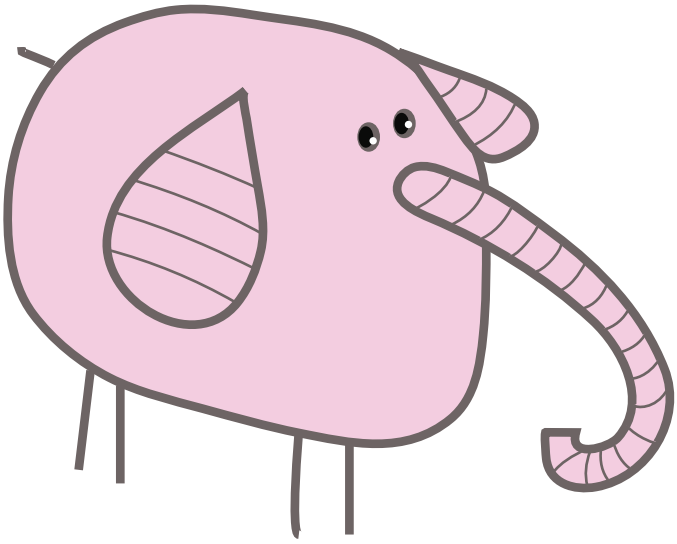


2.4 Add width and height attributes to the use element that referencing to the symbol element. The width value is equal to 780, and the height value is equal to 600.

lesson 4 REUSABLE CONTENT

2.5 Apply trasform="scale()" function with the value equal to 3 to <use> element that referencing to the lion group element to get the same result for the lion.



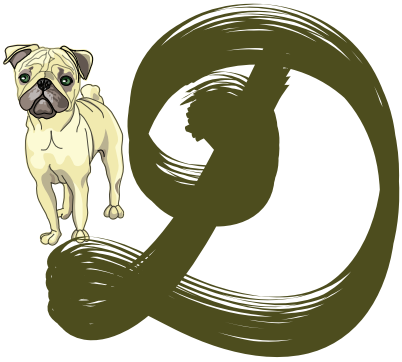


As you can see the scale() function not only increases the size of the group but also it increases the values of the x and y attributes. So the lion changed its position. It was not what we want. we can fix this to move the scale() function from <use> element to the group element with an id="lion". Now it is work just fine.
If we had used symbol element from the beginning to avoid this problem.
But to do this we can can get to another trap. As you may have noticed the symbol element has the transform=”translate()” function that works just fine in the firefox but have no effect when you launch this code in the Chrome, Edge or Opera.
If you move the transform=”translate()” function from the symbol element to use elenent it will work properly in every browser. .

lesson 4

REUSABLE CONTENT

In the third module rewrite the code. It should work the same way in all browsers.



lesson 4

REUSABLE CONTENT

Fill the table with values.
1) If the property adds an additional transformation matrix or sets the value that can be applied without redefining it in the <use> element then add "+" sign in the field.
2) If the property doesn't add an additional transformation matrix or doesn't set the value that can be applied without redefining it in the <use> element, but it adds the default value instead, then add "-" in the field.

properties:	Firefox symbol	Google Chrome, Microsoft Edge, Opera symbol
x		
y		
width		
height		
translate()		

lesson 4

REUSABLE CONTENT

Describe the strategy that allows us to avoid the problems with cross-browser compatibility when we using the symbol and the <use> elements

lesson 4

REUSABLE CONTENT