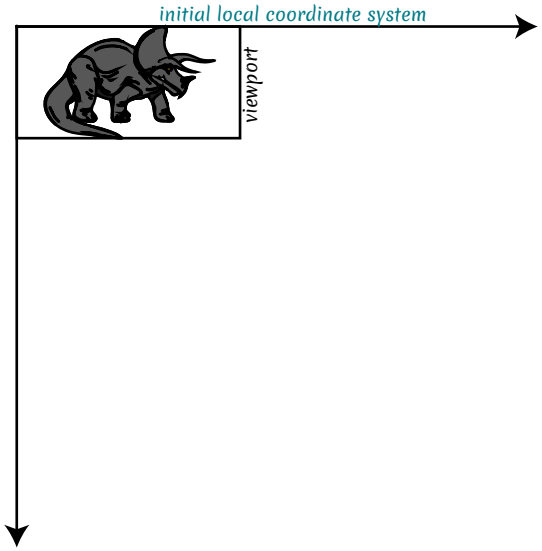
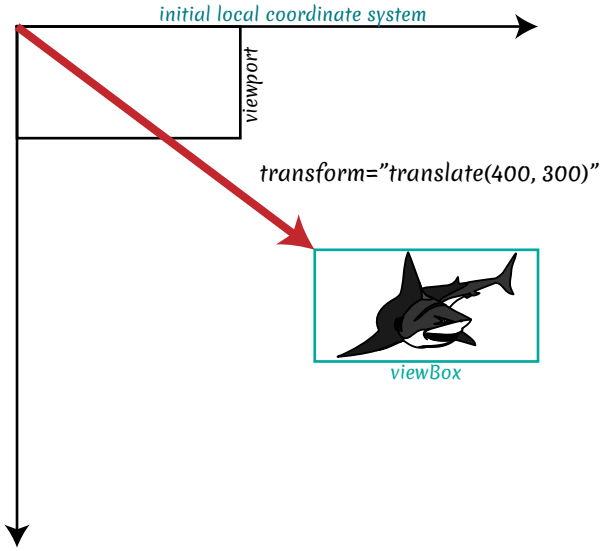
lesson 3

VIEWBOX

min-x and min-y revision

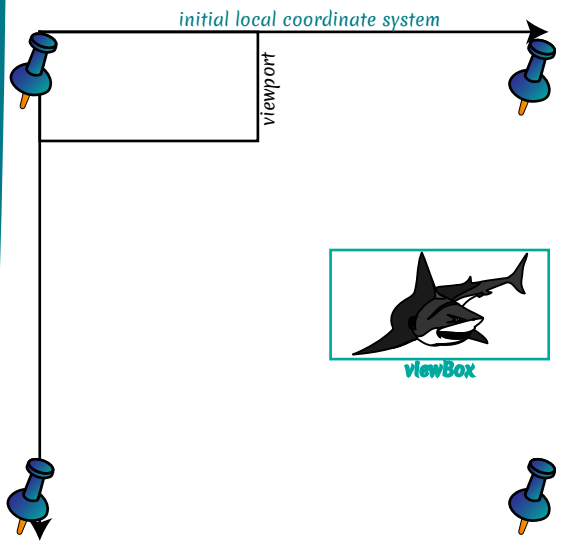


viewBox="0 0 400 300"



viewBox="400 300 400 300"

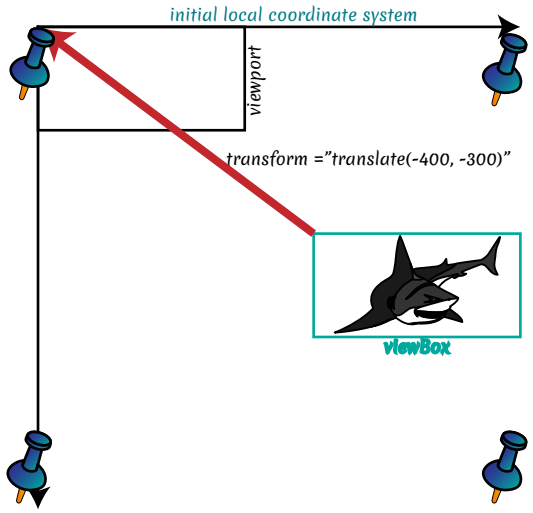
1



We already knew that if we don't specify the viewbox attribute it will be equal to the size of the viewport. We also knew that the second pair of the viewbox's parameters are width and height parameters, and now we will figure out how the first pair of the parameters work. The first pair of the viewbox's parameters are min-x and min-y. Min-x and min-y are the offset of the viewBox's left top edge from the start point of the initial local coordinate system. That is the same if we apply transform="translate()" function with min-x and min-y as the arguments.
We can see that the dinosaur disappear and the shark appears. It happened because the viewbox shows another area of the initial local coordinate system. Despite dinosaur lay under the viewport it's invisible because the mask layer hides it.

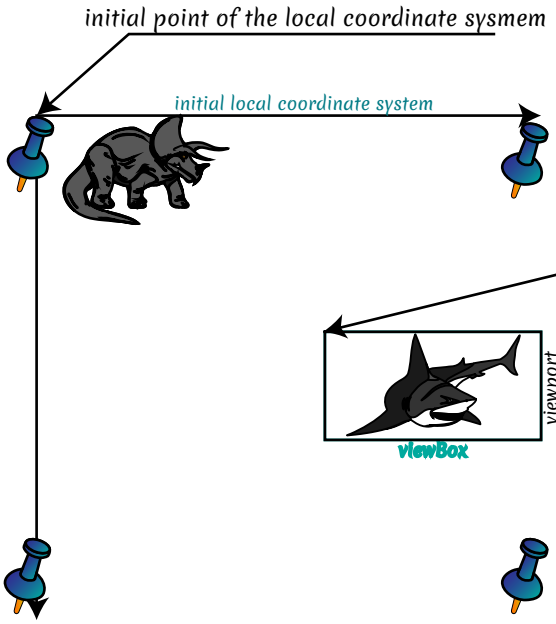
lesson 3

VIEWBOX



2

After the viewBox caught a shark, user agent binds local coordinate system layers with a mask layer. From now on, the user agent will process these bound layers as a whole.
To show the shark within the boundaries of the viewport user agent should move the bound layers to viewport's position.
Note, that user agent moves the bound layers, not viewBox itself. That is why the initial points of the viewport's coordinate system and the initial local's coordinate system don't coincide anymore. We could say that the user agent freezes the viewport layer and moves the bound layers in the background, and that means that the position of the viewport doesn't change inside the HTML document.



1.1 In module one, you should imitate the user-agent algorithm.
You can see a lot of code, but don't worry you should focus on three elements the rect element with an id="viewBox", the g element with an id="transform-backward" and the rect element with an id="viewport" in the very bottom of the file.

Actually, the rect with id="viewBox" and rect with id="viewport" have the same size and positions. The group element(g) let us

VIEWBOX

process a large number of elements as a whole. We will cover it in the next chapter.

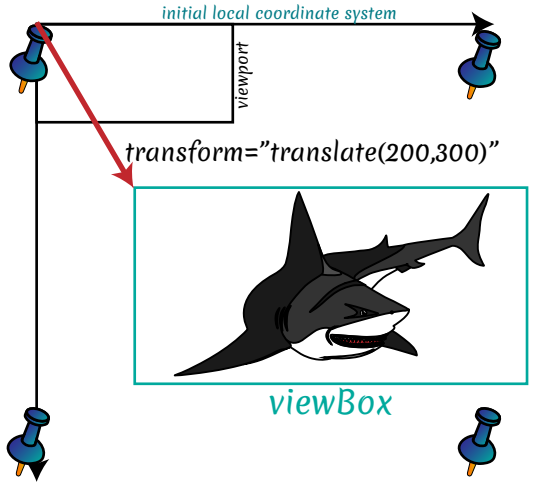
To imitate the viewBox behavior add transform="translate()" function, where x is equal to min-x of the viewBox and y is equal to min-y of the viewBox. ViewBox's min-x is equal to 400 and viewBox's min-y is equal to 300. You should see that the rect with id=" viewBox" moves to the area where the shark is located.

1.2 Now, we should move the rect and the shark in the opposite direction. We could see that shark and the rect with id="viewBox" are inside the g element with an id="transform-backward". Let's add the transform=" translate(-400, -300)" function to move the shark and the rect to place them at the position of the rect with id="viewport".

lesson 3

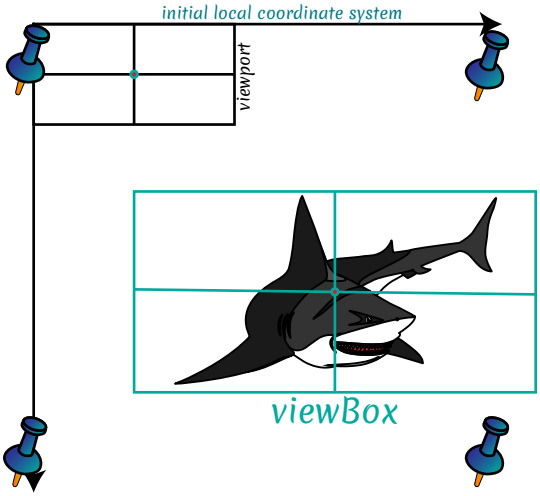
VIEWBOX

VIEWBOX > VIEWPORT OR VIEWBOX < VIEWPORT




```
<svg width="300" height="150"
     viewBox="200 300 600 300">
</svg>
```

1




```
<svg width="300" height="150"
     viewBox="200 300 600 300">
</svg>
```

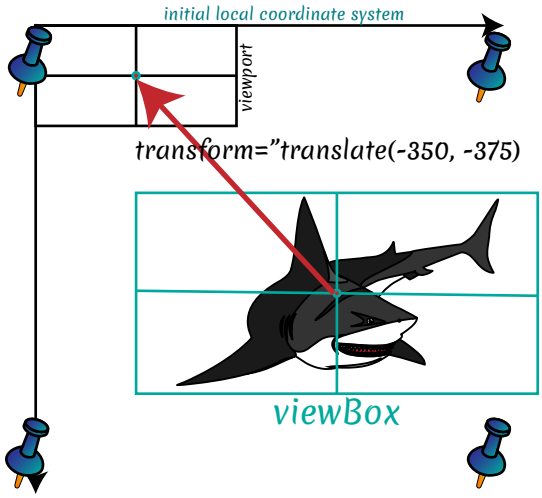
2

2.0 In the previous module, the size of the viewport and the viewBox were equal, but what if they aren't? In the image, you can see that viewBox is larger than the viewport. I added intersecting lines to mark the centres of the viewbox and viewport. You will understand why I did this later on.

In this module, we will assume that viewbox and viewport have the same aspect ratio. Previously you may notice, that when we add the min-x and min-y agruments to viewBox attribute, it was the same as we applied to a viewBox layer the transform =" translate()" function at the first step and applied it to the bound layers at the second step by using the left top points of the viewBox and the viewport as an origin point. For the first step, this algorithm is always correct. We should consider the top left points as the origin points of the viewBox and viewport when viewBox's min-x and min-y aren't equal to zero. You couldn't reassign this behavior of the user agent manually for the first step, but you could reassign the origin points what user agent will use for the viewBox and viewport at the second step by using preserveAspectRatio attribute. I won't cover this attribute until the next lesson, but a thing you should know that the user agent uses the viewport's and viewBox's centers as origin points by default at the second step.

lesson 3

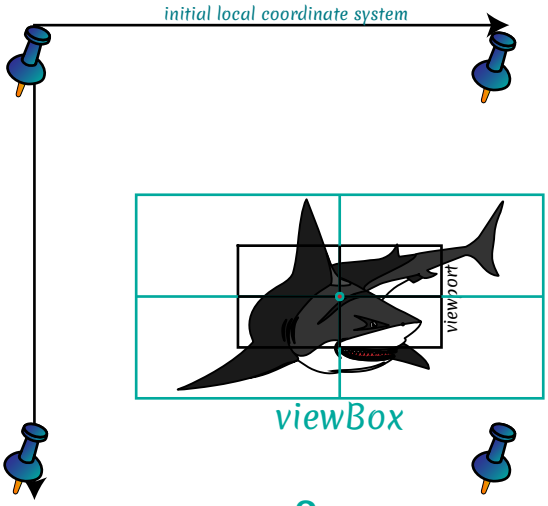
VIEWBOX



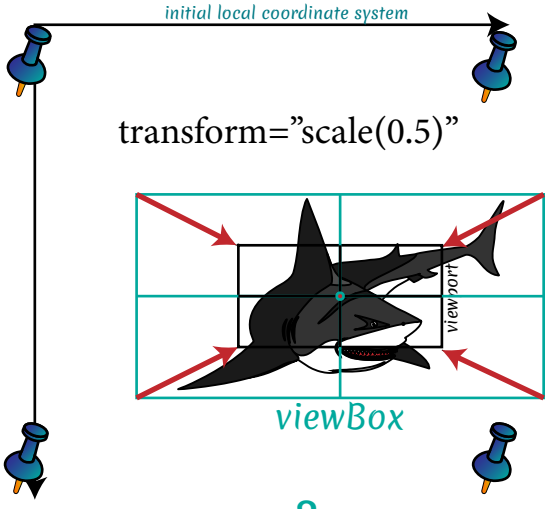

```
<svg width="300" height="150"
     viewBox="200 300 600 300">
</svg>
```

2

At the second step, user agent apply transform=" translate()" function to move the origin point of the viewBox to the position of the origin point of the viewport.



2

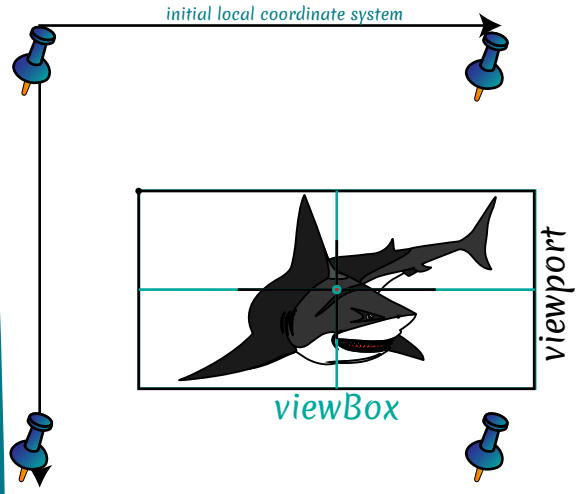


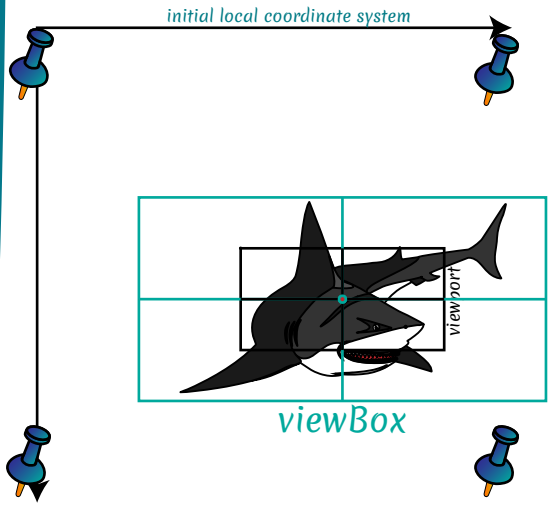
2

When the center points coincide the user agent applies the transform=" scale()" function to a bound layers, and that means that the size of the viewBox will decrease until it is equal to the size of the viewport. As I said, the scale function would apply to a local coordinate system also, so the coordinate system itself and all the graphics that located on it will change their sizes. When I say it will change its size, I mean that the user agent will recalculate the measurement units by multiplying them on the scale coefficient. If the viewBox is larger then viewport the scale coefficient is smaller than one. And if viewBox is smaller than the viewport the scale coefficient is larger than one.

lesson 3

VIEWBOX





"translate(-350, -375)"

2.1 The second module has small changes in comparison with the first module. First of all, the shark is larger than in the previous module. I have already added the transform=" translate()" function to rect element with id="viewBox", that will imitate the behavior of the min-x and min-y attributes of the viewBox. So we will continue from the second step.

As I said earlier, the user agent uses the centers of the viewBox and the viewport as the origin points. So, we should find the coordinates of the centers to calculate the values of the transform=" translate()" function arguments. The viewport starts at the point(0,0), and its width is equal to 300, and its height is equal to 150. So we should use the value of width divided by 2 to get the central point coordinate on the x-axis, and we should use the value of width divided by 2 to get the central point coordinate on the x-axis. Thus, the coordinates of viewport's central point are 150 and 75.

Then we should find the central point of the viewBox. We should use the value of the min-x that is equal to 200 and add it to width divided by 2 that is equal to 300. So the 500 is the coordinate of the central point by x-axis. We should use the value of the min-y that is equal to 300 and add it to width divided by 2 that is equal to 150. So the 450 is the coordinate of the central point of the viewBox by y-axis.

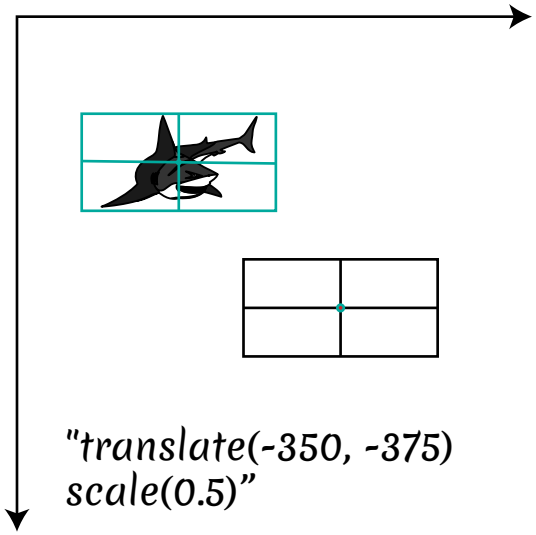
We should subtract the viewBox's central point coordinates from the viewport central point coordinates. Then x equal 150 minus 500 and it is equal to -350. The y equals 75 minus 450, and it is equal to -375.

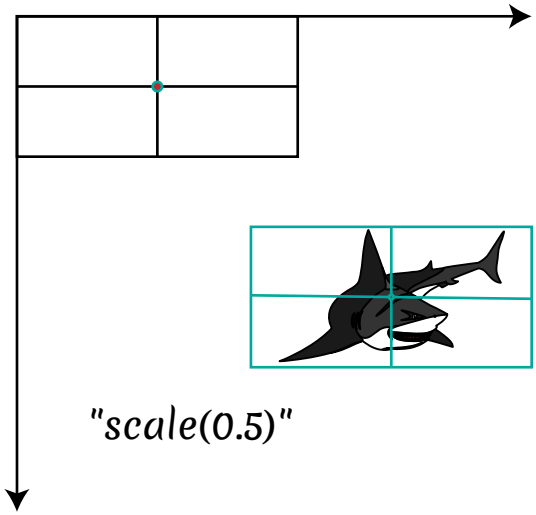
So we should add the transform=" translate(-350,-375)" to the g element with an id="bound_layers". We can see that the centers of the viewBox and the viewport are coincides

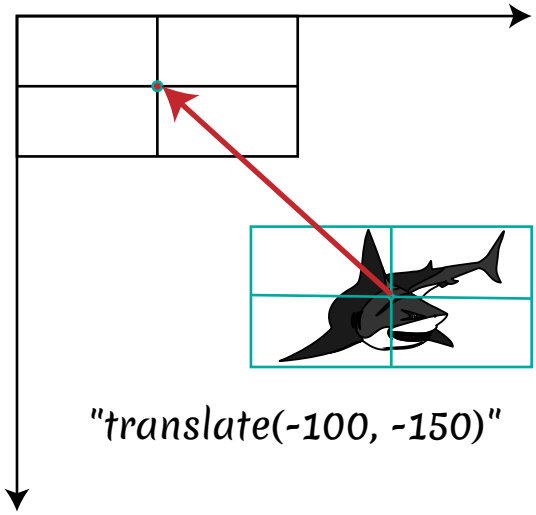
Now, we should reduce the size of the viewBox, to do this we should change transform=" translate(-350,-375)" to transform=" translate(-350,-375) scale(0.5)". When

lesson 3

VIEWBOX



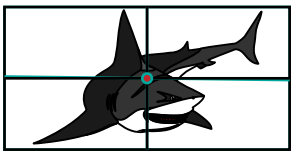




we add the scale(0.5) that mean that we reduce the size by half.
But the result isn't that we expected. We get in "measure units recalculation trap." When we apply the scale function, it recalculates the value of every attribute, for example, the width and the height change its size. The scale function also influences every property of the element and also every descendant element. Thus, transform="translate(200,300)" of the rect element with id="viewBox" changes to transform="translate(100,150)". The same happens with sharks path coordinates.
2.2.To fix this issue, we should reverse the second step. So we will apply the scale function first, and then we will use the transform translate function. I will do for a better understanding only.
So let's add the transform="scale(0.5)" to g element with an id = "bound_layers". From now on, the rect with id ="viewBox" has the width that is equal to 300 and the height that is equal to 150. We know it doesn't have x and y attributes, but instead, we added transform= " translate(200, 300) attribute, that changes to transform = " translate(100, 150) after recalculation. Now, the coordinates of the central point of the viewBox are different. They are equal to 250 for x-axis and 225 for the y-axis.
Now we should calculate the values for transform="translate()".
x = 150-250=-100
y= 75 - 225 = -150.
Add the "translate(-100, -150)" to transformation-list to get the following result transform="scale(0.5) translate(-100,-150)".
We get the result we want.

lesson 3

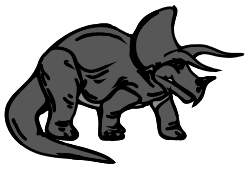
VIEWBOX



"translate(-100, -150)
scale(0.5)"

lesson 3

VIEWBOX

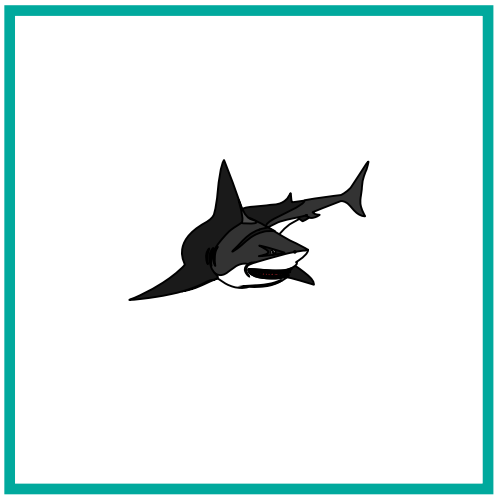


3.1 In the third module, there is an svg element that contains two images the shark and the triceratops prorsus. The viewport and the viewBox are equal. Width and height are both equal to 800.
Change the viewBox width and height to 400.
What animal do you see?

_____

Change min-x and min-y to make the visible.
The viewBox min-x is equal to _____

The viewBox min-y is equal to _____



viewBox="0 0 800 800"
width="800" height="800

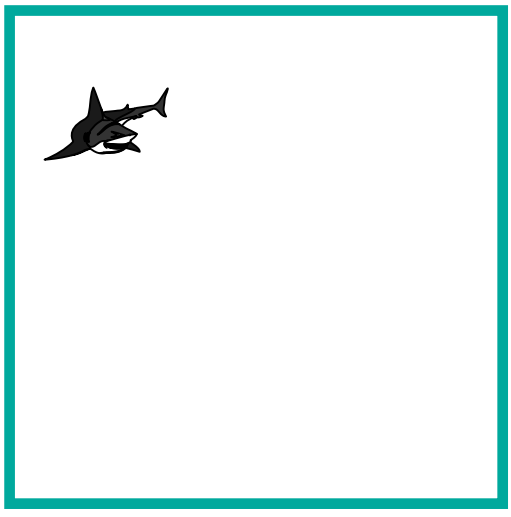
4.1 In the fourth module, we will focus on the word "scale" of the SVG. As you can mention in the previous module the differences in size between the viewport and viewBox enforce the user agent to use scale function. The viewBox and the viewport relationship is the cornerstone in the scalable vector graphics.
In this module, we could see the svg element that contains the width, height, and viewBox attribute. The width and height are both equal to 800, and the viewBox is equal to "0 0 800 800".

lesson 3

VIEWBOX

viewBox=”200 250 400 400”
width=”800” height=”800

4.2 Change the viewBox value to "200 250 400 400". What happens with the shark size? Why does it happen?



viewBox=”0 0 1600 1600”
width=”800” height=”800

4.3 Change the viewBox value to "0 0 1600 1600".
What happens with the shark size?
Why does it happen?