**Lesson 1. Introduction to SVG, line, rect tags**

Objective

Become familiar with SVG graphics and basic SVG tags, learn how to draw squares and lines using svg tags

Contents

[Types of graphics and what SVG is 2](#_Toc67398481)

[Benefits of SVG 2](#_Toc67398482)

[Creating the SVG tag 2](#_Toc67398483)

[<line> Tag and SVG Coordinates 3](#_Toc67398484)

[The <rect> tag and overlay 5](#_Toc67398485)

[Finally 7](#_Toc67398486)

[**Homework** 7](#_Toc67398487)

[№1. Draw plus 7](#_Toc67398488)

[№2: create a mirror effect 7](#_Toc67398489)

What we repeat

Introductory svg tutorial

What's new

1. Raster and vector graphics.
2. SVG Coordinate System.
3. Tags <line>, <rect>.

Links to materials and personal account

[Working materials](https://hwschool.bitrix24.ru/bitrix/tools/disk/focus.php?folderId=304237&action=openFolderList&ncc=1)(for the teacher).

[Materials](https://hwschool.bitrix24.ru/~r5DGQ) (we send this link to the student at the beginning of the lesson).

Methodical material

Types of graphics and what SVG is

There will be a lot of theory in this section, so be careful, if a student gets bored, it is better to move on to practice. But I think 5-7 minutes of introduction will not hurt. Images or graphics are constantly used on our sites. They can be divided into raster and vector, let's try to figure out what their differences are.

1. Raster graphics - pictures are composed of colored dots (pixels). It is possible to make images of any complexity, but at the same time they weigh quite a lot and dots are noticeable when magnified.
2. Vector graphics - pictures are made up of geometric shapes (lines, circles, rectangles, etc.). They are lightweight, do not lose quality when enlarged, but using shapes it is impossible to make complex images.

**SVG**(Scalable Vector Graphics) - a markup language for scalable vector graphics(you can insert bitmaps, but more on that later).

Benefits of SVG

Consider why we are learning SVG

1. Scaling: like any vector graphics, it does not lose quality.
2. Styling: Using CSS, you can change the graphics options on the site, such as background, transparency, or borders.
3. Animation and Editing: Using JavaScript, you can animate SVG, and also edit in a text or graphics editor (InkScape or Adobe Illustrator). We will do this in a text editor (Brackets or Sublime Text)

Creating the SVG tag

The tag for creating SVG graphics is called SVG, let's write the code:

**<! DOCTYPE html>**

**<html>**

**<head>**

**<meta charset = "UTF-8">**

**<title> SVG Introduction </title>**

**</head>**

**<body>**

**<svg> </svg>**

**</body>**

**</html>**

Let's see the size of our tag in the inspector, initially it is 300 to 150



Let's increase it, only we will do it not in CSS, as we are used to, but using attributes, it looks strange, but this will help us in the future for writing animations. Let's make the tag square (sometimes it seems to me that it comes out not quite a square, but still)

**<svgwidth = "500" height = "500">**

**</svg>**

<line> Tag and SVG Coordinates

It's time to create the first line inside our tag, use the line tag and indicate the attributes of the beginning **x1, y1** and the end **x2, y2** (If the student does not remember about the **xOy** coordinate system, then we draw parallels with a scratch or Cartesian coordinate system from school mathematics).

**<svg width = "500" height = "500">**

**<line x1 = "0" y1 = "0" x2 = "100" y2 = "100"> </line>**

**</svg>**

The line appeared in the inspector, but it is still not visible, because it has no thickness, we will set it with one more attribute **stroke-width** (responsible for the thickness) and **stroke** (responsible for the color).

**<svg width = "500" height = "500">**

**<line x1 = "0" y1 = "0" x2 = "100" y2 = "100"stroke-width = "1" stroke = "green"**

**> </line>**

**</svg>**



We see a line that starts from the upper left corner, and not from the middle, although we set the coordinates (0,0). How did this happen? Let's figure out the SVG coordinate system, for this I send this picture to the students



So, we see a number of differences from the standard coordinate system.

1. The dot (0,0) is in the upper left corner
2. **X** increases to the right up to the **width** attribute
3. **Y** grows down to the **height** attribute
4. The bottom right corner will have the **(width, height)** attributes

Caution: The Y coordinate changes downward, which slightly confuses us

Now we will make a cross from two lines, it is desirable that the student tries to calculate the coordinates himself

**<svg width = "200" height = "200">**

**<line x1 = "0" y1 = "0" x2 = "100" y2 = "100" stroke-width = "1" stroke = "green"> </line>**

**<line x1 = "200" y1 = "0" x2 = "0" y2 = "200" stroke-width = "1" stroke = "green"> </line>**

**</svg>**

Done, but we wrote the width of the line and the color twice, so we will take out these two attributes in **CSS**, also make the lines thicker and instead of the closing tag we will use **/** at the end of the opening tag, since nothing can be written inside the line.

**<svg width = "200" height = "200">**

**<line x1 = "0" y1 = "0" x2 = "200" y2 = "200"/>**

**<line x1 = "200" y1 = "0" x2 = "0" y2 = "200"/>**

**</svg>**

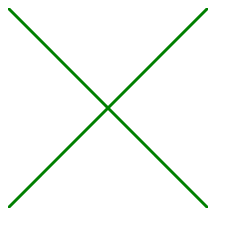
Let's connect the CSS and write the appropriate styles (color and thickness at our discretion)

**line {**

**stroke: green;**

**stroke-width: 3px;**

**}**



The <rect> tag and overlay

Now we will learn how to make rectangles in **SVG**, for this we need the **<rect>** tag, we will also set its width and height

**<line x1 = "0" y1 = "0" x2 = "200" y2 = "200" />**

**<line x1 = "200" y1 = "0" x2 = "0" y2 = "200" />**

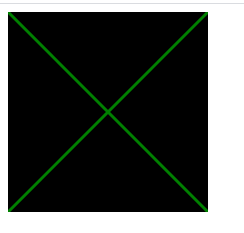
**<rect width = "200" height = "200" />**

But now our lines are not visible, let's swap the tags so that **rect** is at the beginning of the SVG

**<rect width = "200" height = "200" />**

**<line x1 = "0" y1 = "0" x2 = "200" y2 = "200" />**

**<line x1 = "200" y1 = "0" x2 = "0" y2 = "200" />**



Conclusion

Now everything is fine, from this we conclude that in **SVG** elements can overlap each other, as if they had **position** set to **absolute** or **fixed**. You can imagine that inside the **SVG** tag we have layers being formed, with the lowest tag being in the foreground, let's try to remember this.

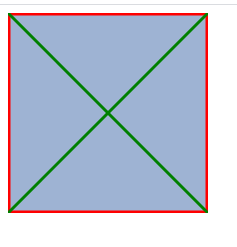
Back to **rect**, we can change the fill color using the **fill** attribute or **CSS** property (any color can be used)

**<rect width = "200" height = "200"fill = "# 9EB3D3"/>**

We can also change the border color and width using the same properties as **stroke** for color and **stroke-width** for width.

**<rect width = "200" height = "200"fill = "# 9EB3D3"stroke = "red" stroke-width = "5"/>**

Let's see the result

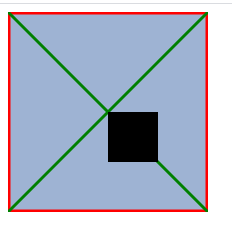


Great, let's also create a small square and move it to the center using the **x** and **y** attributes. To do this, let's take and calculate the middle of our SVG tag (we created a square for simplicity, so as not to count 2 times)

**width (200) / 2 = 100**

**<rect width = "50" height = "50" x = "100" y = "100" />**

They wrote everything correctly, but not quite.

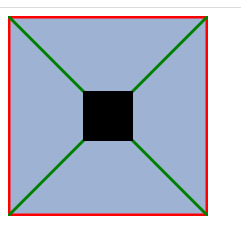


The center is now the top-left corner of our rect tag, remember when we talked about the peculiarities of coordinates inside **SVG**. To set the center of the rectangle to the center of the **SVG** tag, subtract from our middle half the width and height of the square, that’s the result:

**100-50 / 2 = 75**

**<rect width = "50" height = "50" x = "75" y = "75" />**

Now it’s exactly what we need



Finally

We tell the student that in the next lesson we will learn how to make circles, ellipses, polygons and broken lines.

From the point of view of practical application, vector graphics are widely used on many sites, but we will look at the top and the basics of this topic and we will even create simple games, if appropriate, we give an analogy with scratch: ping pong, racing, dinosaur from Google Chrome, etc. .d

For the teacher: I see the main task of the section on SVG in the development of spatial thinking and the ability to apply arithmetic calculations in practice.

**Homework**

№1. Draw plus

Use the lines to make a plus sign

№2: create a mirror effect

Use 3-4 rectangles to make a mirror effect

Examples of work are in the student folder

