**Lesson 2. Drawing in SVG**

Objective

Draw a house, a car and a smiley

Contents

[Introduction 1](#_Toc67401253)

[Drawing circles 2](#_Toc67401254)

[Drawing broken lines and polygons 4](#_Toc67401255)

[Practice 5](#_Toc67401256)

[Finally 6](#_Toc67401257)

[**Homework** 6](#_Toc67401258)

What we repeat

1. SVG coordinates
2. <rect>, <line> tags

What's new

1. <circle>, <ellipse> tags
2. <polyline>, <polygon> tags

Links to materials and personal account

[Working materials](https://hwschool.bitrix24.ru/bitrix/tools/disk/focus.php?folderId=355497&action=openFolderList&ncc=1)(for the teacher).

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

Methodical material

Introduction

In the last lesson, we learned how to draw lines and rectangles using SVG tags. Today we will learn how to make circles and polygons. Let's create a separate folder and in it two files, as usual index.html and style.css. Immediately write the <svg> tag in the body of the site, set its width and height (let's say 500 to 500) and move on.

Drawing circles

Let's create a **<circle>** tag inside the svg and let's think about what attributes the circle might have.

We give the student to think, if there are no ideas, then we explain.

A circle has center coordinates and a radius, given by the attributes **cx, cy, r (center x, center y, radius)**

**<! DOCTYPE html>**

**<html>**

**<head>**

**<title> SVG Lesson 2 </title>**

**<link rel = "stylesheet" type = "text / css" href = "style.css"> </head>**

**<body>**

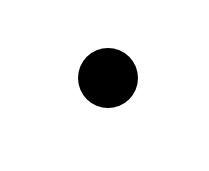
**<svg width = "500" height = "500">**

**<circle cx = "100" cy = "100" r = "30" />**

**</svg>**

**</body>**

**</html>**



Once everything is written we see a black circle, but you can change the color of the fill and borders. This is done in the same way as in the previous tags that we learned. Let's write something like this in CSS:

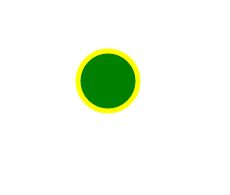
**circle {**

**fill: green;**

**stroke: yellow;**

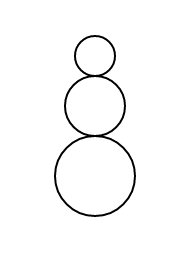
**stroke-width: 5px;**

**}**



We see the result, and now, to fix it, I propose to create a snowman from circles using the <circle> tag

The task seems simple, but it will help the student to calculate the coordinates.



It turned out something like this, if the student cannot calculate the coordinates of the center and radius, then we help him

**x** does not change, because all the circles are on the same line, y will change when we make a circle lower, then y is larger, and higher is less (do not forget about the SVG coordinates, point (0.0) is in the upper left corner) To calculate the coordinate, let's use the obvious formula:

C***y***(new) = C***y***(previous) ± (**r**(old) + **r**(new)),

Where ***Cy – Y center coordinate, and r is the radius***

The image is in the folder with the lesson materials for the teacher.

Finished code:

**HTML**

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

**<circle cx = "100" cy = "100" r = "30" />**

**<circle cx = "100" cy = "170" r = "40" />**

**<circle cx = "100" cy = "50" r = "20" />**

**</svg>**

**CSS**

**circle {**

**fill: white;**

**stroke: black;**

**stroke-width: 2px;**

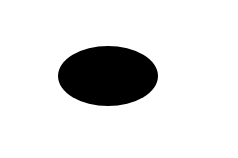
**}**

Now let's look at a very similar **<ellipse>** tag, it differs only in that it has two **x** and **y** radii. You can create a new **svg** tag, or you can write that the previous one.

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

**<ellipse cx = "100" cy = "100" rx = "50" ry = "30" />**

**</svg>**



Obviously, if you set the same radii, you get a circle.

Drawing broken lines and polygons

We already know how to draw lines, now we will consider broken lines - this is the polyline tag. With it, we indicate not the start and end points with the attribute, but all the points through the points attribute. We will offer the student to enter their coordinates or take from an example.

**<svg width = "299" height = "200">**

**<polyline points = "0,0 100,100 0,200" stroke = "black" stroke-width = "2">**

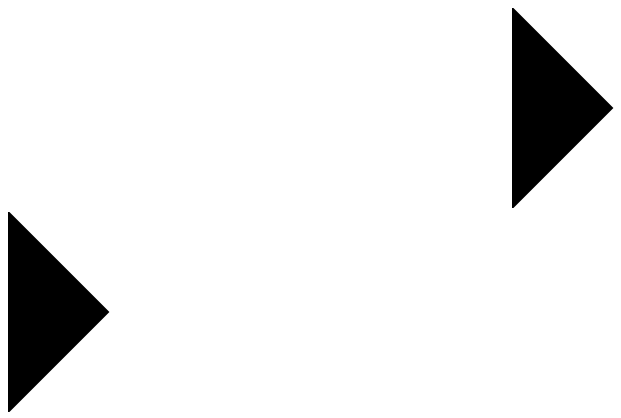
**</svg>**

That is, the **x** and **y** coordinates of one point are written separated by commas, and a space is made between the points. The polygon is created in the same way. Let's create the exact same SVG tag and inside it a polygon with the same coordinates.

**<svg width = "299" height = "200">**

**<polygon points = "0,0 100,100 0,200" stroke = "black" stroke-width = "2">**

**</svg>**



Let's look at the result and suggest wrapping them in a generic div tag to separate them from the previous SVG tag and put them on one line.

**<div>**

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

**<poly line points = "0,0 100,100 0,200" stroke = "black" stroke-width = "2" />**

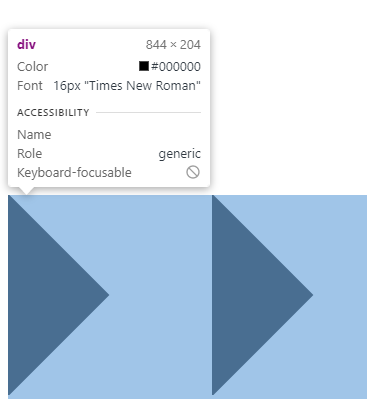
**</svg>**

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

**<polygon points = "0,0 100,100 0,200" stroke = "black" stroke-width = "2" />**

**</svg>**

**</div>**

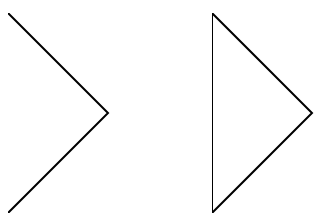


So far, both tags look completely the same, but let's try to remove the fill, since we do this for two tags, I suggest using CSS (I think by this point the student already remembers that this property is **fill**)

**polygon, polyline {**

**fill: none;**

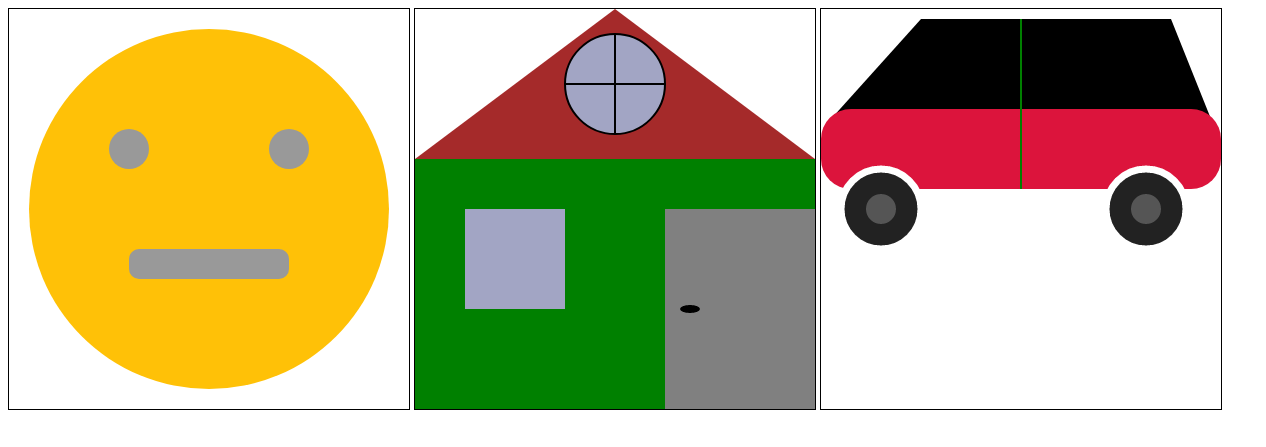
**}**



Now we can see the difference. The polyline itself does not close, but the polygon completes the line to close.

Practice

So, we have learned the basic shapes of SVG graphics and figured out the coordinate system. Now let's practice and create some simple drawings.



We ask the student to choose any of the proposed ones and draw it in a new file (he can put any colors and sizes). The commented code for creating the image is located in the **classPractice.html** file in the teacher’s resource link.

In the process, the student will encounter rounded corners at the rectangles, the **rx** attribute will help in this, like an ellipse, only we apply it to the rectangle (**rect**), thus we get the local analogue of border-radius. Used to create the mouth of the smiley face and the body of the car.

For those who come across old SVG projects: the car in the source used to be more beautiful, but it is more difficult to draw it, so I had to simplify it.

Finally

When there are a few minutes left until the end of the lesson, we tell the student that in the next lesson we will already be using **JavaScript** in conjunction with **SVG** graphics. If the student is curious, here is a link to a free resource on **SVG** fill and stroke

http://css.yoksel.ru/svg-fill-and-stroke/

**Homework**

1. Finish the remaining drawings yourself
2. Repeat **DOM (querySelector**, etc)