**Lesson 3. Animated DVD screensaver**

Objective

Create animated DVD screensaver

Contents

[Introduction 1](#_Toc67673473)

[Index.html file 2](#_Toc67673474)

[Style.css file 3](#_Toc67673475)

[Script.js file 4](#_Toc67673476)

[Preparatory stage 4](#_Toc67673477)

[Picture movement 5](#_Toc67673478)

[Correcting a data type error 5](#_Toc67673479)

[Timer and continuous movement 7](#_Toc67673480)

[Pushing off the edges 8](#_Toc67673481)

[**Homework** 12](#_Toc67673482)

What we repeat

1. SVG coordinates
2. JS methods for working with attributes (setAttribute, getAttribute)
3. Variables and Functions
4. Timer operation

What's new

1. **<image>** tag
2. Converting to number **parseInt**

Links to materials and personal account

[Working materials](https://hwschool.bitrix24.ru/bitrix/tools/disk/focus.php?folderId=375969&action=openFolderList&ncc=1)(for the teacher).

[Materials (edit)](https://hwschool.bitrix24.ru/~HBc5w)(we send this link to the student at the beginning of the lesson).

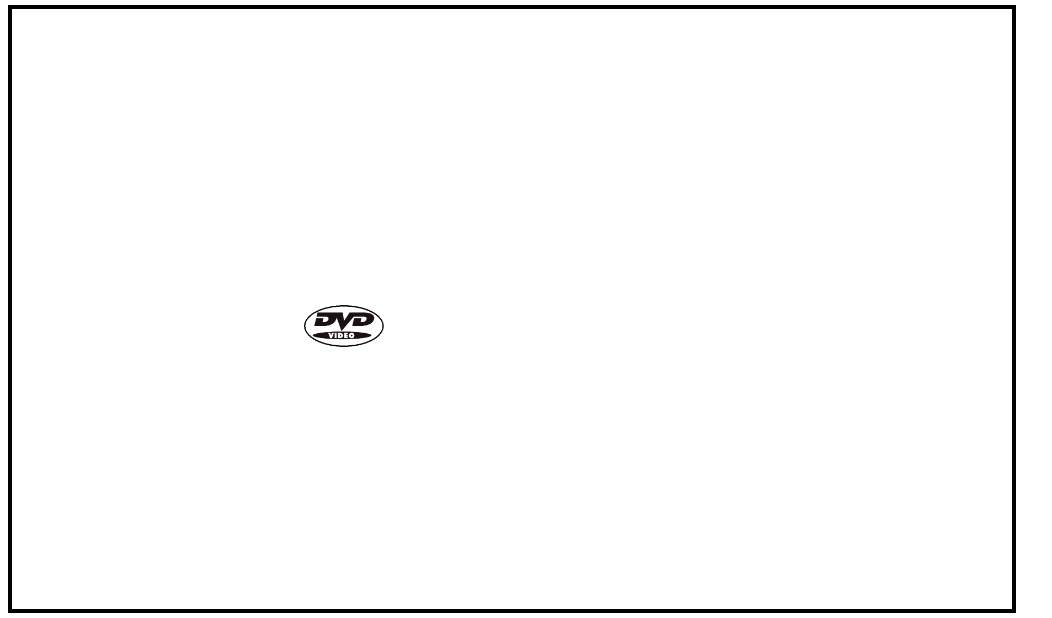
[Video presentation of the finished project](https://youtu.be/bvgBBzHyTdA)

Methodical material

Introduction

Today we will learn how to link **SVG** and **JavaScript** and create a DVD splash screen that bounces off the edges.

The image of the final project and the splash screen itself are in the "For student" folder



Let's immediately create 3 files: **index.html, style.css, script.js**

Index.html file

Let's write a standard template from the main tags, while do not forget to include styles, scripts, and the SVG tag (do not forget to prescribe arbitrary sizes with attributes, for example 1000 by 600)

**<! DOCTYPE html>**

**<html>**

**<head>**

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

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

**<body>**

**<svg width = "1000" height = "600">**

**</svg>**

**<script type = "text / javascript" src = "script.js"> </script>**

**</body>**

**</html>**

Now let's learn how to insert bitmaps inside **SVG** (be sure to check with the student if he remembers the difference between vector and raster graphics), the **SVG image** tag will help us with this. Attention! It has **href** as an attribute for the path to the image, and not **src** as in the long familiar **img**) You can also set its initial position and dimensions

**<svg width = "1000" height = "600">**

**<image x = "40" y = "40" href = "DVD.png" height = '50 'width = '80' />**

**</svg>**



Done, now we offer the student to independently move the logo to the left, bottom and any other part of the SVG tag, changing the initial x and y coordinates, thus fixing the coordinates



Style.css file

Nothing new here, just give the **SVG** tag a border to see the borders. For example:

**svg {**

**border: 4px solid black;**

**}**

Script.js file

Preparatory stage

Now let's get down to **JavaScript**, first of all we create the move function and call it in the file

**function move () {**

**}**

**move ()**

Now we will pull out the svg tag from the html, put it in a variable and output it to the console

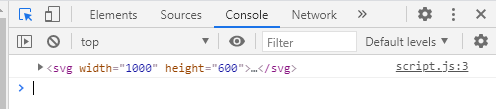
**function move () {**

**svg = document.querySelector ('svg');**

**console.log (svg)**

**}**

**move ()**



We will do the same with the image tag, leaving the output to the console at the end of the function.

**function move () {**

**svg = document.querySelector ('svg');**

**let image = document.querySelector ('image');**

**console.log (svg)**

**}**

**move ()**

Now is the time to “pull out” the x and y coordinates of the picture, so that later we can make its movement. First, let's remember how to find out any attribute of a tag.

If the student does not succeed, then we remind him about getAttribute, pull out the attributes and print them to the console

**let svg = document.querySelector ('svg');**

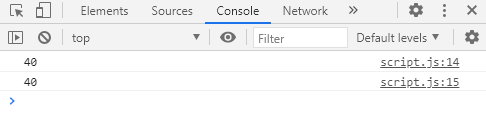
**let image = document.querySelector ('image');**

**let x = image.getAttribute ('x');**

**let y = image.getAttribute ('y');**

**console.log (x);**

**console.log (y);**



Picture movement

It's time to make our picture move. To do this, we will change its x and y coordinates. Let's add some number (arbitrary number) to the current coordinate and accumulate the value in a variable

**x + = 100;**

**y + = 100;**

So far, nothing is moving, because we need to return its value back to the attribute, remember how to do this?

If not, then we give a hint that you need to remember the to-do list and how we set the path for the basket picture there. If it doesn't work, then just talk about setAttribute. This function has two arguments: the first is what attribute we set, the second, what value. Let's write at the end of the function:

**image.setAttribute ('x', x);**

**image.setAttribute ('y', y);**

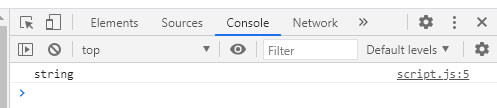
Correcting a data type error

We see a problem with the fact that our picture has disappeared somewhere altogether. Let's look at its coordinates through the element code



The numbers are huge, let's think why this happened. If there are no ideas, then we tell. Now, inside the variables, the results of which we got from the attributes, are numbers or strings? Let's check with **typeof**(we haven't used this thing for a long time, but don't forget)

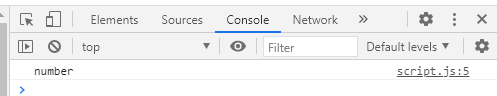
**console.log (typeof (x))**



We find out that there is a line, and this needs to be fixed, the parseInt function will help us with this, we will use it when assigning a value to a variable

**let x = parseInt (image.getAttribute ('x'));**

**console.log (typeof (x))**



Now it's different, let's fix it for **y** too

**function move () {**

**let svg = document.querySelector ('svg');**

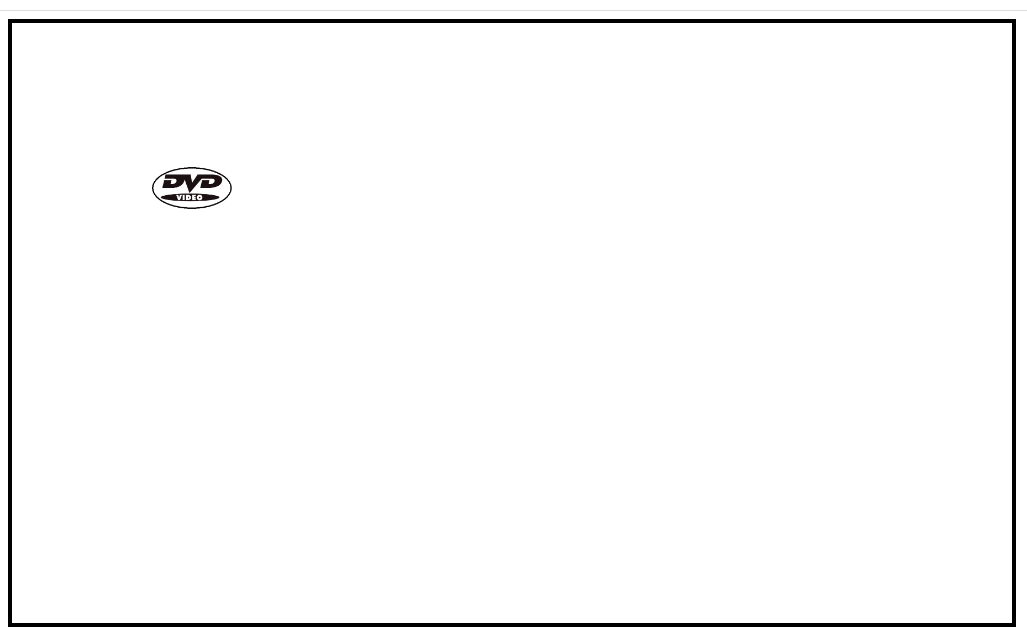
**let image = document.querySelector ('image');**

**let x = parseInt (image.getAttribute ('x'));**

**let y =parseInt(image.getAttribute ('y'));**

**}**

We look at the result



Timer and continuous movement

Our image has moved exactly 100 pixels down and to the right, but we won't write a huge number to move it further.

To solve this problem we need a timer(perhaps the student got to know him while creating a splash screen in JS).

We have 2 functions

1.**setInterval** - continuously calls something after a certain interval of time

2. **setTimeout** - executed once after the specified delay. The first function suits us, let's call move at intervals, let's say every 10 milliseconds.

Note that inside setInterval we do not call the function, that is, we do not write move (), because we do not execute the function, but pass it, and setInterval will execute it for us)

**setInterval (move, 10)**

But now the image flew away very quickly into the corner, I propose to slow down the speed somewhere down to 2-3, as a result we get

**function move () {**

**let svg = document.querySelector ('svg');**

**let image = document.querySelector ('image');**

**let x = parseInt (image.getAttribute ('x'));**

**let y = parseInt (image.getAttribute ('y'));**

**x + =2;**

**y + =2;**

**image.setAttribute ('x', x);**

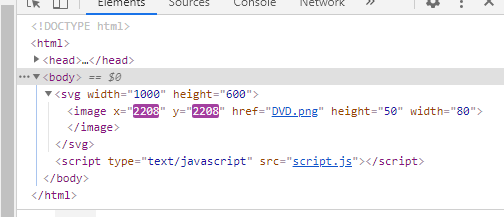
**image.setAttribute ('y', y);**

**}**

**setInterval (move, 10)**

Pushing off the edges

Now the splash screen flies to the lower right corner and continues to move. We can verify this by opening the element code and looking at the coordinates of the **image** tag



To fix the situation, we need to find out the coordinates of the edges of the **SVG** tag and change the direction of movement when they are touched. Let's take out our step in **x** and **y** into variables and find out the width and height of the **SVG** tag (don't forget about ParseInt), write them into the variables **width** and **height**.

**function move () {**

**let svg = document.querySelector ('svg');**

**let image = document.querySelector ('image');**

**let width = parseInt (svg.getAttribute ('width'));**

**let height = parseInt (svg.getAttribute ('height'));**

**let x = parseInt (image.getAttribute ('x'));**

**let y = parseInt (image.getAttribute ('y'));**

**x + = 2;**

**y + = 2;**

**image.setAttribute ('x', x);**

**image.setAttribute ('y', y);**

**}**

**setInterval (move, 10)**

Variables must be global, that is, outside the function, because if you declare them inside the function, they will be created anew every time, we do not need this. By the way, let's think about what other variables we don't need to recreate every time.

Answer: everything except **x** and **y** of the picture, because they are constantly changing, so the rest can be safely taken out as a function

**let step\_x = 2;**

**let step\_y = 2;**

**let svg = document.querySelector ('svg');**

**let image = document.querySelector ('image');**

**let width = parseInt (svg.getAttribute ('width'));**

**let height = parseInt (svg.getAttribute ('height'));**

**function move () {**

**let x = parseInt (image.getAttribute ('x'));**

**let y = parseInt (image.getAttribute ('y'));**

**x + = step\_x;**

**y + = step\_y;**

**image.setAttribute ('x', x);**

**image.setAttribute ('y', y);**

**}**

**setInterval (move, 10)**

Now let's write that if the y of the image is larger than the bottom edge, then we will change the sign of our **step\_y**. And what is y of the lower border for us? Correct - the height of the **SVG** tag.

Checking before putting numbers back into html

**function move () {**

**x + = step\_x;**

**y + = step\_y;**

**if (y> = height) {**

**step\_y = -step\_y;**

**}**

**image.setAttribute ('x', x);**

**image.setAttribute ('y', y);**

**}**

Everything worked, but the picture fell over the edge, and then pushed off. Why did it happen? The answer lies in the SVG coordinates, we compared the top border of the image tag with the bottom border of the svg tag.

If the student does not understand this point, then we draw in Zoom or remind about the image with SVG coordinates



It is very easy to fix this, find out the height and width of the image from the **html attribute** and add the height to get the coordinates of the bottom border of the image

**let step\_x = 2;**

**let step\_y = 2;**

**let svg = document.querySelector ('svg');**

**let image = document.querySelector ('image');**

**let width = parseInt (svg.getAttribute ('width'));**

**let height = parseInt (svg.getAttribute ('height'));**

**let width\_ball = parseInt (image.getAttribute ('width'));**

**let height\_ball = parseInt (image.getAttribute ('height'));**

**function move () {**

**let x = parseInt (image.getAttribute ('x'));**

**let y = parseInt (image.getAttribute ('y'));**

**x + = step\_x;**

**y + = step\_y;**

**if (y + height\_ball> = height) {**

**step\_y = -step\_y;**

**}**

**image.setAttribute ('x', x);**

**image.setAttribute ('y', y);**

**}**

**setInterval (move, 10)**

Great, now everything is working as it should.

**Homework**

1. Flip off edges at **x** and top edge at **y**
2. Try to fit within two **if** statements and use compound conditions through “and”, “or”.

Teacher code

**if (y + height\_ball> = height || y <= 0) {**

**step\_y = -step\_y;**

**}**

**if (x + width\_ball> = width || x <= 0) {**

**step\_x = -step\_x;**

**}**