

GeoGebra – Javascript – Lesson 1

InterActivity – Change point on number line


Author: Linda Fahlberg-Stojanovska

Thanks to my friends at the GeoGebra Forum

www.mathcasts.org/mtwiki

math247.pbwiki.com

Key Concepts from GeoGebra

1. Work with axes to get a number line.
2. Create number a .
3. Create point  with x-coordinate a and y-coordinate 0.

Key Concepts from HTML

4. Html, Head, Body
5. Table, Form, Input
6. Applet

Key Concepts from Javascript

7. Javascript functions.
8. Generate a random sign (+ or -).
9. Generate a random natural number.
10. Evaluate an expression.
11. Javascript/GeoGebra: `document.ggbapplet.evalCommand`

Key Concepts from Mathematics

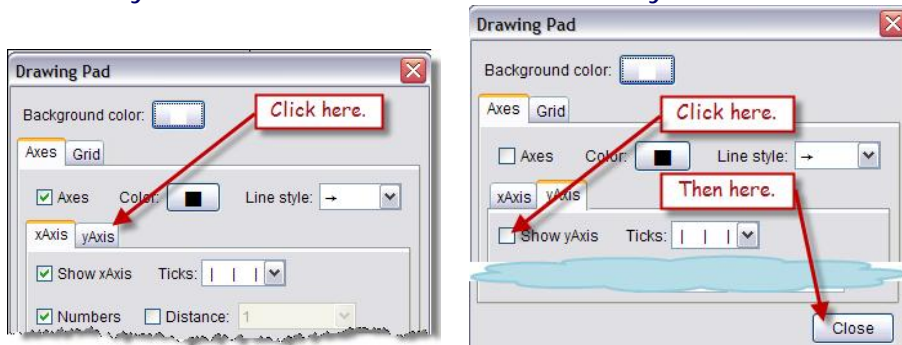
12. Number Line


Disclaimer ☺: I am no expert at anything here – but give me your feedback and together I think we can do really cool things to help our kids learn math.

By its nature, this operation involves 4 different aspects of technology – geogebra, html, javascript and mathematics itself – each can be used at varying levels of expertise. Particularly for these first 2 lessons I have tried to make all 4 aspects as basic as possible and describe each part in detail since that is how I got started.

Script-o-matic
GeoGebra

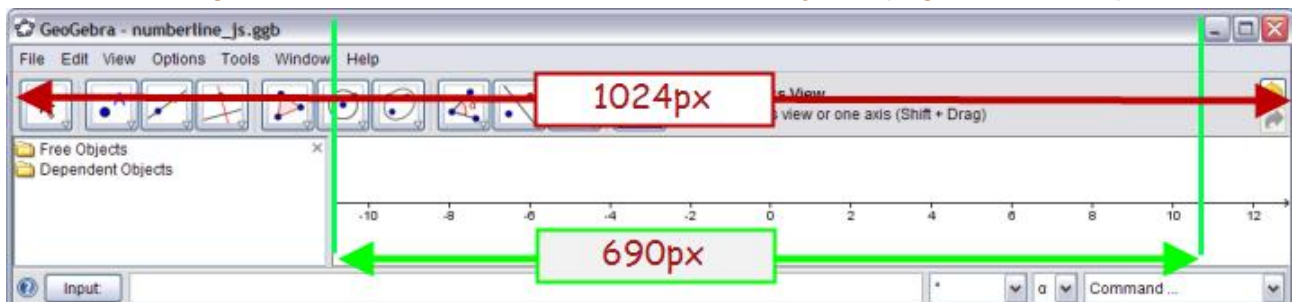
1. Create geogebra file with number line.
 - a. Open new geogebra file.
 - b. If not open, open the algebra window (View -> Algebra Window).
 - c. We want to turn off the y-axis.
 - i. Right-click in graphics view (a.k.a. drawing pad) and choose Properties.
 - ii. Drawing box dialog box will open.
Click on yAxis tab and then deselect Show yAxis. Click on Close.



- d. We want to move the x-axis up towards the top and zoom out.
 - i. Click on the "Move graphics view" icon  and then click and drag the x-axis towards the top.
 - ii. With this icon selected, you should be able to use your mouse wheel to zoom in/out. We want the number line to be from -10 to 10 since we will generate this "size" of numbers in our script.

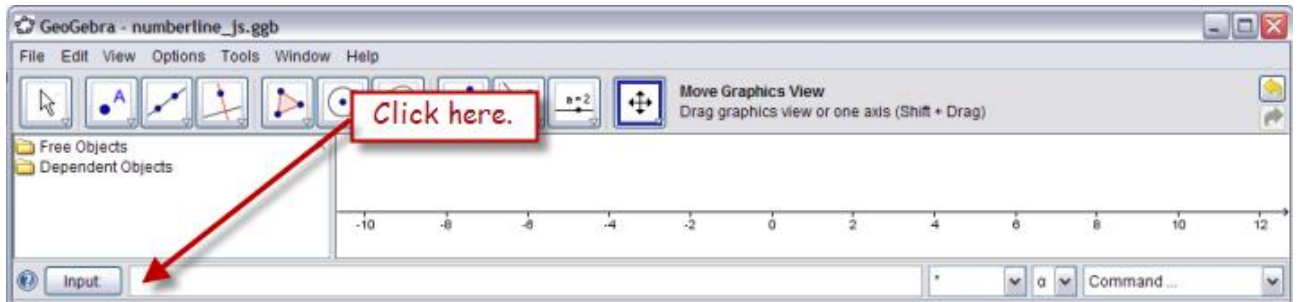
Note:

- When programming, we want the Algebra window to be visible so we can check the interaction. When done, we will usually close the Algebra window which gives the user a larger graphics view.
- For now we will "use" a ggb window of standard width=1024px (I use a freebie program called sizer to get this) with an open Algebra window and plan my width to under the minimize button. (Then when the Algebra window is closed, this will fit in my webpage width 690px.)



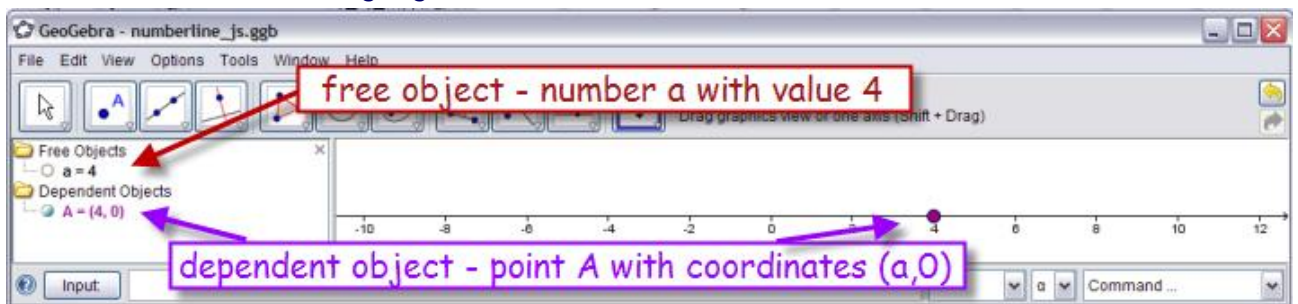
- iii. You can fix the increments, tick marks, etc. of the x-axis.
However, if you want your user to be able to zoom in/out, don't do this because geogebra cannot then automatically adjust.

2. Make a starter number a and point $A=(a,0)$ on number line.
 - a. Click in the input field, type or copy/paste $a=4$ and hit enter.
(If you don't see Input, command View -> Input bar.)



- b. Your cursor is still in input field. Type or copy/paste $A=(a,0)$ and hit enter.
3. Decorate point A

- a. Turn off the label on this point and make it bigger and a distinctive color.
(Right-click on the point and choose properties from drop-down menu.
On the *basic* tab, deselect "Show label", on the *style* tab, move the slider to 5 and on the *color* tab, choose your favorite color.)
 - b. Here is what our geogebra file looks like so far.



Note:

- From the webpage we will be changing the value of the number a . This will automatically change the value of the point A and its position on the number line.

4. Save your geogebra file as numberline_js.ggb.

Webpage

5. Our webpage has 3 main parts.
 - a. It has a *text-html* part with instructions and "form" for user input.
 - b. It has a *geogebra* part with the "applet" part that looks for and then opens and displays the geogebra file.
 - c. It has a *javascript* part with the "function" for generating a new point on the number line and for communicating with the geogebra file. (In the next lesson, we will add another function to check whether the user has correctly identified the point value.)

Parts a and b are visible - that is, the user will see the text and the geogebra file.
Part c is invisible to the user.

Webpage - html part

The text in the box is non-visible html code. All browsers (firefox, internet explorer) know how to read this code.

Open a new file in a text editor (notepad) and copy and paste this text into it.

Or, if you use an html editor (dreamweaver, alleycode), first switch to code-view, erase all the code there (most editors automatically generate the required parts of this code plus some of their own) and then copy and paste in this code.

Save the file as numberline_js.html in the same folder as numberline_js.ggb.

All none-shaded lines in the box are REQUIRED.

== You may change the **green text** – this is the title of the *webpage* itself .

== The shaded stuff between the <style> tags is optional.

It is formatting directions to make the text small enough so there is space for the geogebra file and to make sure that Firefox changes its cursor to a hand over the input button.

The actual content (**parts a, b and c**) will come between the <body> tags.

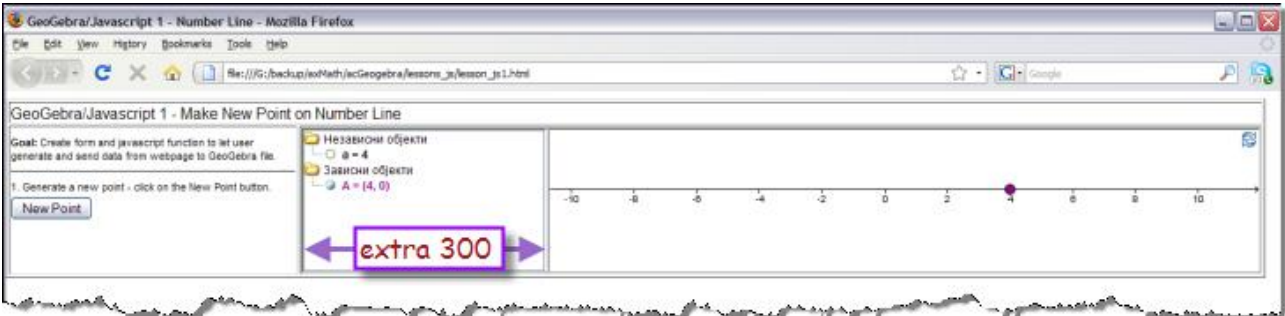
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>GeoGebra/Javascript 1 - Number Line</title>
    <style type="text/css">
      <!--
      body
      {font-family:Helvetica,sans-serif; margin-left:0px; }
      p,ul,ol
      {font-size:.7em; margin-top:3px; margin-bottom:3px;}
      /* cursor changer for Mozilla Firefox */
      .navCurOff
      {cursor: default;}
      .navCurOn
      { cursor: pointer; cursor: hand;}
      -->
    </style>
  </head>
  <body>
    HERE WE PUT CONTENT OF PAGE - IN THE BODY
  </body>
</html>
```

If you open this file in your browser, it will only have the 1 line: **HERE WE PUT ...**

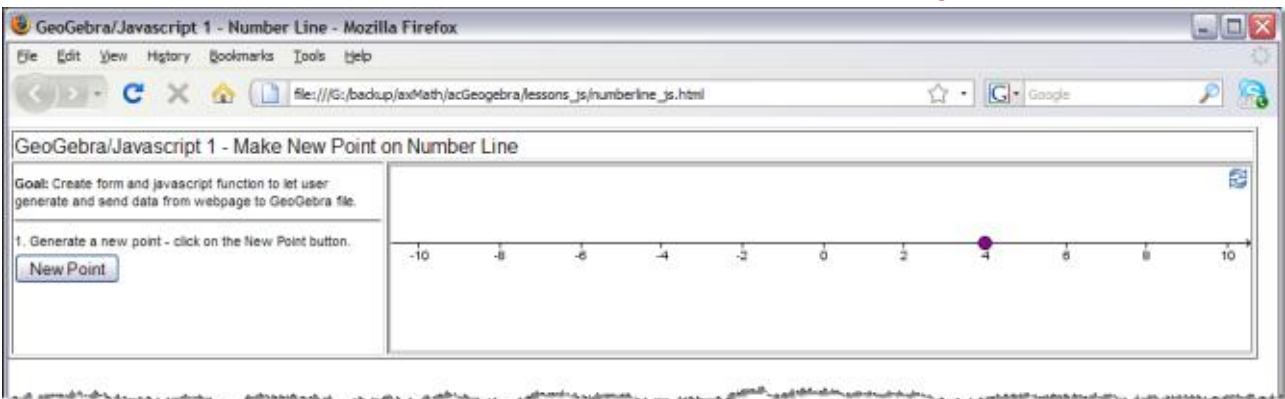
BODY Part a and Part b – Visible text

What will our page look like?

While we are working (open Algebra window so extra width of 300):



When we are done (fits on 1024x790 resolution with closed Algebra window):



Notice the table on this page:



A table is a good way to organize the content of a webpage

Here the top row spans two columns and contains text-title.

The second row has two columns – the instruction column and the ggb page column.

Instructions has a paragraph, a horizontal line, a paragraph, and an Input Button.

Teal text - notice it spans 2 columns	
Blue text Orange text Input Button	GeoGebra file

So - the visible part of page is put in a table between <table> tags.

Idea is a simple table of width 1000px so it fits on a standard monitor resolution in a standard browser (firefox or internet explorer). I usually make my instruction column 300px wide and my GeoGebra column 690px wide with 10px leftover for borders.

While we are working we will add 300px to 1000px and to 690px (2 places!) for the algebra window. When we are done, we go back to 1000 and 690.

```
<table border="1" width="1300">
  <tbody>
    <tr>
      <td colspan="2">GeoGebra/Javascript 1 - Make New Point on Number Line </td>
    </tr>
    <tr>
      <td width="300"><p><b>Goal:</b> Create form and javascript function to let user
generate and send data from webpage to GeoGebra file. </p>
      <hr>
      <p>1. Generate a new point - click on the New Point button. </p>
      <form>
        <input name="button" type="button" class="navCurOff"
onClick="gensetInt('a',10);" onMouseOver="this.className='navCurOn';"
onMouseOut="this.className='navCurOff';" value="New Point">
      </form>
      <p>&nbsp;</p>
      <p>&nbsp;</p></td>
      <td width="990">
        <applet name="ggbApplet" code="geogebra.GeoGebraApplet" codebase="./"
archive="http://www.geogebra.org/webstart/geogebra.jar" height="150"
width="990">
          <param name="filename" value="numberline_js.ggb">
          <param name="showAlgebraInput" value="false">
          <param name="showMenuBar" value="false">
          <param name="showToolBar" value="false">
          <param name="showResetIcon" value="true">
          Sorry, the GeoGebra Applet could not be started. Please make sure that Java
1.4.2 (or later) is installed and activated. (<a
href="http://java.sun.com/getjava">click here to install Java now</a>)
        </applet></td>
    </tr>
  </tbody>
</table>
```

You can copy and paste this text between the <body> tags in your numberline_js.html file and save it again. (Delete the **HERE WE PUT ...** .)

If you now open this file in your browser, you should see the "working version".

Don't click on the Input Button - we haven't defined the javascript function.

The two "important" parts here are the **FORM** and the **APPLET**

1. The form


```
<form>
  <input name="button" type="button" class="navCurOff"
onClick="gensetInt('a',10);" onMouseOver="this.className='navCurOn';"
onMouseOut="this.className='navCurOff';" value="New Point">
</form>
```

Most of this stuff is "formatting", but it uses the html style stuff (from the first part of the html code) to make the cursor change to a pointer or hand when the mouse rolls over the button so that one "knows" it is clickable.

- **type** says that input is a **button** you click on (and not e.g. text you type in).
- **value** says button name is "New Point" (this is text that you can change).
- **onClick** says - when the user clicks on the button - the page should run the **javascript** function "gensetInt('A',10)".

2. The applet

```
<applet name="ggbApplet" code="geogebra.GeoGebraApplet" codebase="./"
archive="http://www.geogebra.org/webstart/geogebra.jar" height="150"
width="690">
  <param name="filename" value="numberline_js.ggb">
  <param name="showAlgebraInput" value="false">
  <param name="showMenuBar" value="false">
  <param name="showToolBar" value="false">
  <param name="showResetIcon" value="true">
  Sorry, the GeoGebra Applet could not be started. Please make sure that
  Java 1.4.2 (or later) is installed and activated. (<a
  href="http://java.sun.com/getjava">click here to install Java now</a>)
</applet>
```

- **archive** says where to find the files that will run geogebra - when 3.2 is released (in February 2009), [dev/unpacked/](#) should be deleted. (This is typical, but it also means one must be online to run the applet. More about this later...)
- **width** is the width of the display of the ggb file. Match width of table column.
- **height** is the height of the display of the ggb file. (I added to blank rows in the left column `<p> </p>` to compensate for this height.)
- **filename** is the name of the ggb file. Notice no path means that this html file and the ggb file must be in the SAME folder.
- **showAlgebraInput**, **showMenuBar**, **showToolBar** are whether input field, command menu and icon tool bar are shown with the geogebra file. (Showing these on the page is a matter of space and whether you want your students to use them. *False* is default value so all three lines can be deleted.)
- **showResetIcon** is whether you get the reset icon  to be able to reload ggb.

BODY Part c - Javascript

The javascript part of the page is put between <script> tags.

```
<script type="text/javascript">
//
function gensetInt(objName,rang) {
    // generates random plus and minus sign
    var sgn1 = 2*Math.floor(Math.random()*2)-1;
    // generates random integer between 1 and rang - use floor to get 0 to rang-1
    var n1 = Math.ceil(Math.random()*rang);
    var z=sgn1*n1;
    document.ggbApplet.evalCommand(objName + " = " + z);
}
//
</script>
```

Usually one puts the javascript at the the end – just above the </body> of the html.
Copy and paste this text into your numberline_js.html file and save it.

6. When the user clicks on the New Point button, the above javascript function will be called using the command `gensetInt('a',10)`.

Two parameters are passed into the function.

The 1st is `objName='a'` (text) and the 2nd is `rang=10` (number).

- javascript math functions:
 - `Math.random()` generates a random number between 0 and 1
 - `Math.floor(number)` finds the largest integer \leq number
 - `Math.ceil(number)` finds the smallest integer \geq number
- geogebra/javascript function:
 - `document.ggbApplet.evalCommand(legitimate ggb input)`
where javascript variables are separated by + and text by “.
Mixing of text and numbers is allowed.

Here, let's assume we generated `z=-3`. We were given `objName = 'a'`.

`objName + " = " + z` means `a=-3`

So the function `gensetInt(objName,rang)` will

- randomly generate `sgn1` a sign (+ or -),
- randomly generate a natural number `n1` between 1 and `rang=10`,
- multiply these to get `z` and
- set `a=z` in `numberline_js.ggb`

In `numberline_js.ggb` automatically,

- a. the number `a` will take on the value `z`
- b. the point `A` will change to `(z,0)` and
- c. the purple point will move to `z` on the number line.

While “working”, we keep the Algebra window open so we can see all of these changes.

7. Let's try it!

- Open `numberline_js.html` in your browser.
- Roll your mouse over the **New Point** button. Check cursor changes to a pointer.
- Now click on the **New Point** button. In the algebra window, you should see a `a` and (then) `A` change and in the graphics view, the purple point should move. (First time may take a moment.)
- Click on the **New Point** button again to see it generate a new point.

When you are satisfied that all is working,

- in GEOGEBRA close the Algebra window and re-save `numberline_js.ggb`
- in text/HTML editor, change `1300->1000` and `990->690` (2 times) and re-save `numberline_js.html`

Reload `numberline_js.html` in your browser.

If the algebra window still shows open, you may need to close all instances of your browser to clear your cache (so it reloads the newly saved ggb file).

Now, when you click on the **New Point** button only `"c"` will be visible to the user – that is, the user see a new point on the number line.

A couple of javascript hints:

- `//` (two forward slashes) is a comment line in javascript. Comment everything.
- Sections of code are separated by `{ }` and individual lines of code in a section end with a semi-colon `;`; indenting is a good way to keep sections separate.
- We can and will have more than one javascript function between script tags.
- Javascript is case sensitive. Try and be consistent about capitalization.
- Javascript is extremely unforgiving of missing punctuation.
- Save yourself a lot of time. Write and check each section of code individually.
- I use the code line: `alert("Here1");` to see where I am stuck.
- Unlike most programming languages, it is hard to get parameters out of a javascript function. We will use global variables.

A couple of html hints:

- A good free html editor – <http://www.alleycode.com/download.htm> (use `Ctrl+D` to switch between code view and wysiwyg).
- Blanks (spaces or rows) mean NOTHING in html. (This text will only have 1 space between each word. The extras are thrown away.)
 - To add blank spaces, use the code for a "non-breaking space" ` `;
 - To add blank row, use the code for "break row" `
`
- Learn to close all your tags – e.g. `<tr>` starts a table row. When you are done with the row, type `</tr>`. Most good editors have code checkers.

What the html file looks like in Dreamweaver (color-coded code)

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>GeoGebra/Javascript 1 - Number Line</title>
6 <style type="text/css">
7 <!--
8 body
9 {font-family:Helvetica,sans-serif; margin-left:0px}
10 p,ul,ol
11 {font-size:.7em; margin-top:3px; margin-bottom:3px;}
12 /* cursor changer for Mozilla Firefox */
13 .navCurOff
14 {cursor:default;}
15 .navCurOn
16 {cursor:pointer; cursor:hand;}
17 -->
18 </style>
19 </head>
20 <body>
21 <table border="1" width="1300px">
22 <tbody>
23 <tr>
24 <td colspan="2">GeoGebra/Javascript 1 - Make New Point on Number Line </td>
25 </tr>
26 <tr>
27 <td width="300px"><p><b>Goal:</b> Create form and javascript function to let user generate and
28 send data from webpage to GeoGebra file. </p>
29 <hr>
30 <p>1. Generate a new point - click on the New Point button. </p>
31 <form>
32 <input name="button" type="button" class="navCurOff" onClick="gensetInt('a',10);" onMouseOver=
33 "this.className='navCurOn';" onMouseOut="this.className='navCurOff';" value="New Point">
34 </form>
35 <p>&nbsp;</p>
36 <p>&nbsp;</p></td>
37 <td width="990px">
38 <applet name="ggbApplet" code="geogebra.GeoGebraApplet" codebase="." archive=
39 "http://www.geogebra.org/webstart/dev/unpacked/geogebra.jar" height="150" width="990">
40 <param name="filename" value="numberline_js.ggb">
41 <param name="showAlgebraInput" value="false">
42 <param name="showMenuBar" value="false">
43 <param name="showToolBar" value="false">
44 <param name="showResetIcon" value="true">
45 Sorry, the GeoGebra Applet could not be started. Please make sure that Java 1.4.2 (or later)
46 is installed and activated. (<a href="http://java.sun.com/getjava">click here to install Java now</a>)
47 </applet></td>
48 </tr>
49 </tbody>
50 </table>
51 <script type="text/javascript">
52 //
53 function gensetInt(objName,rang) {
54 // generates random plus and minus sign
55 var sgn1 = 2*Math.floor(Math.random()*2)-1;
56 // generates random integer between 1 and rang - use floor to get 0 to rang-1!
57 var n1 = Math.ceil(Math.random()*rang);
58 var z=sgn1*n1;
59 document.ggbApplet.evalCommand(objName + " = " + z);
60 }
61 //
62 </script>
63 </body>
64 </html>
65
```