J A V A S C R I P T

‘ ‘ T H E W O R L D ’ S M O S T M I S U N D E R S T O O D P R O G R A M M I N G L A N G U A G E ’ ’

D O U G L A S C R O C K F O R D

THE JSXGRAPH DEVELOPERS

JSXGRAPH

UNIVERSITY OF BAYREUTH

PUBLISHED BY UNIVERSITY OF BAYREUTH

JSXGRAPH.ORG

First printing, July 2009

Contents

Dedicated to those who appreciate JavaScript

Introduction

This booklet shows how to use the open-source library JSXGraph (http://jsxgraph.org).

JSXGraph — what?

Interactive Geometry, plotting, visualization

JSXGraph is a cross-browser library for interactive geometry, function plotting, graphs, and data visualization in a web browser. It is implemented completely in JavaScript and uses SVG and VML. JSXGraph is easy to embed and has a small footprint: only about 70 kByte if embedded in a web page. No plug-ins are required!

JSXGraph uses the JavaScript libraries/frameworks Prototype or jQuery.

JSXGraph is developed at the Lehrstuhl für Mathematik und ihre Didaktik, University of Bayreuth, Germany.

Available at http://jsxgraph.org

Features

- Euclidean Geometry: Points, lines, circles, intersections, perpendicular lines, angles
- Curve plotting: Graphs, parametric curves, polar curves, data plots
- Turtle graphics
- Lindenmayer systems
- Interaction via sliders
- Animations
- Polynomial interpolation, spline interpolation
- Tangents, normals
- Charts
- Vectors

License JSXGraph is released under the LGPL - Lesser GNU General Public License. So, everybody is encouraged to use it.

Include JSXGraph into web pages

Additional files

For including JSXGraph into HTML, three files are necessary:

- prototype.js from http://www.prototypejs.org
- jsxgraphcore.js
- jsxgraph.css

You can either download these three files and use the local copy or you can use the online version. Then, the beginning of the HTML file should start like this:

```
<html>
<head>
 <link rel="stylesheet" type="text/css" href="jsxgraph.css" />
 <script type="text/javascript" src="prototype.js"></script>
 <script type="text/javascript" src="jsxgraphcore.js"></script>
</head>
<body>
...
</body>
</html>
```

If you want to include the online of JSXGraph in your HTML file then you have to write the following lines into the document head:

```
<html>
<head>
 <link rel="stylesheet" type="text/css" href="http://jsxgraph.uni-bayreuth.de/distrib/jsxgraph.css" />
 <script type="text/javascript" src="http://jsxgraph.uni-bayreuth.de/distrib/prototype.js"></script>
 <script type="text/javascript" src="http://jsxgraph.uni-bayreuth.de/distrib/jsxgraphcore.js"></script>
</head>
<body>
...
</body>
</html>
```

Alternatively, the framework jQuery from http://jquery.com can be used. The following text has to be included in the head part of the HTML file.

```
<head>
 <link rel="stylesheet" type="text/css" href="jsxgraph.css" />
 <script type="text/javascript" src="jquery.min.js"></script>
 <script type="text/javascript" src="jsxgraphcore.js"></script>
</head>
```

What to use? prototype.js or jquery.js? The choice completely depends on you. JSXGraph works with both packages equally good. Choose the package you're most familiar with.

The drawing panel

The geometric construction which is displayed by JSXGraph resides in an HTML element. Usually, a div-element is taken. This division needs an ID. Using this ID, we declare this element to be a drawing panel of JSXGraph.

The following code has to be placed into the body part of an HTML file:

```
<div id="box" class="jxgbox"  style="width:500px;␣height:500px;"></div>
<script type="text/javascript">
 var board = JXG.JSXGraph.initBoard('box', {boundingbox:[-5,5,5,-5], axis:true});
</script>
```

We can use as many different drawing panels as we like in one HTML file. The class jxgbox sets "position:relative" which seems to be mandatory for the Internet Explorer 7.

Then, the web browser should display an element like the one shown in Figure 1.

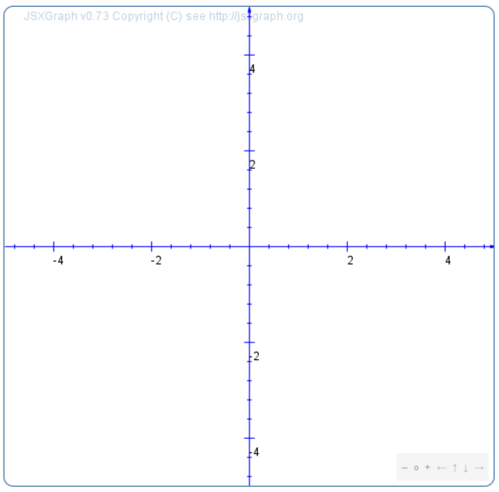

Figure 1: The first JSXGraph construction.

The complete HTML file then looks like this:

```
<html>
<head>
 <link rel="stylesheet" type="text/css" href="jsxgraph.css" />
 <script type="text/javascript" src="prototype.js"></script>
 <script type="text/javascript" src="jsxgraphcore.js"></script>
</head>
<body>
<div id="box" class="jxgbox" style="width:500px;␣height:500px;"></div>
<script type="text/javascript">
 var board = JXG.JSXGraph.initBoard('box', {boundingbox:[-5,5,5,-5], axis:true});
</script>
</body>
</html>
```

Connect JSXGraph with the HTML div tag (usually at the end of the document body) and call the method initBoard() of the global object JXG

```
<script type="text/javascript">
 var board = JXG.JSXGraph.initBoard('box',
                {boundingbox:[-5,5,5,-5], axis:true});
</script>
```

The method initBoard() may have two arguments:

- Parameter 1: the id of the div tag in the HTML page which will contain the JSXGraph construction.
- (optional) Parameter 2: additional properties of the board.

The possible optional properties of the board are:
* originX, originY (in pixel) * unitX, unitY (in pixel) * zoomX, zoomY * Bounding box * axis (true/false) * grid (true/false) * showNavigation (true/false) * showCopyright (true/false)

More than one boards can be initialised simultaneously in one HTML file.

Creating geometric elements

The next step is to create geometric elements in the drawing panel which can be dragged around. Through the JavaScript variable board in the above listing we have access to the drawing panel and can place objects there. New geometry elements can be added to the board. All elements are added with the method board.createElement(). One example:

```
board.createElement('point',
        [1,3],
        {name:'A', strokecolor:'red'}
        );
```

Another example:

```
board.createElement('point',
        [function(){return s.X();},function(){return t.X()}],
        {trace:true}
        );
```

The parameters of the method board.createElement() are:

```
board.createElement(elementType, parents, attributes);
```

where

- elementType is a string containing the type of the element which is constructed. At the moment, possible types are:
  - primitive elements like points, lines, curves
  - composite elements like bisectors, midpoints
- parents is an array containing the parameters which define the element. This can be parent elements like two points which define a line. It can also consist of JavaScript functions, numbers, and strings containing GEONExT[1] syntax. The possible array elements depend on the element type.
- attributes is an optional argument and has to be a JavaScript object. Usually it is given in the form {key1:value1, key2:value2, ...} , called the "literal object" form.

[1] see http://geonext.de

Construction of a free point

This example in Figure 2 shows how to construct a simple, draggable point. It is produced by the following commands:

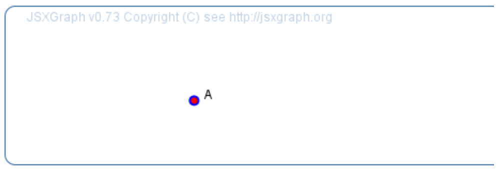

Figure 2: The first JSXGraph point.

```
<div id="box" class="jxgbox"
    style="width:200px;␣height:200px;"></div>
<script type="text/javascript">
 var board = JXG.JSXGraph.initBoard('box',
               {boundingbox:[−2,2,2,−2]});
 var p = board.createElement('point',[1,1]);
</script>
```

The JavaScript code has to be placed *after* the div element which will contain the construction. From now on, we will only show the JavaScript code.

Attributes of a point

Several attributes can be given to change the properties of a point, for example a name or the point style.

```
 var board = JXG.JSXGraph.initBoard('box',
               {boundingbox:[−2,2,2,−2]});
 var p = board.createElement('point',[1,1],{name:'X',style:5});
```

The resulting point in Figure 3 is now labeled with "X".

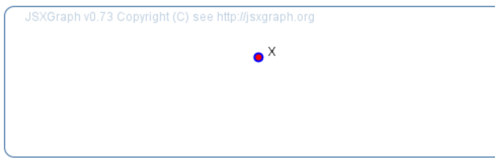

Figure 3: The JSXGraph point "X".

Point styles The layout of a point can be influenced by the property type. It can attain the values $0, 1, \ldots, 12$. Alternatively of these equivalent constants can be used:

Constant	value	description
JXG.POINT_STYLE_X_SMALL	0	Small x
JXG.POINT_STYLE_X	1	Medium x
JXG.POINT_STYLE_X_BIG	2	Big x
JXG.POINT_STYLE_CIRCLE_TINY	3	Tiny circle
JXG.POINT_STYLE_CIRCLE_SMALL	4	Small circle
JXG.POINT_STYLE_CIRCLE	5	Medium circle
JXG.POINT_STYLE_CIRCLE_BIG	6	Big circle
JXG.POINT_STYLE_SQUARE_SMALL	7	Small square
JXG.POINT_STYLE_SQUARE	8	Medium square
JXG.POINT_STYLE_SQUARE_BIG	9	Big square
JXG.POINT_STYLE_PLUS_SMALL	10	Small +
JXG.POINT_STYLE_PLUS	11	Medium +
JXG.POINT_STYLE_PLUS_BIG	12	Big +

In the following example we use a for loop to create 13 points attaining all possible styles. The result can be see in Figure 4.

```
var board = JXG.JSXGraph.initBoard('box',
                {boundingbox:[-2,2,2,-2]});
for (var i=0;i<13;i++) {
  var p = b3.createElement('point',[i,0],
                {name:'P_{'+i+'}', style:i});
}
```

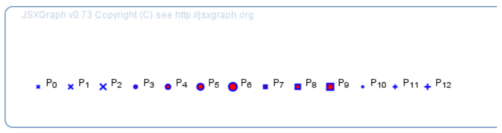

Figure 4: All possible point styles.

Other attributes of a point are

attribute name	value	description
style	0...12	see above
strokeColor	color string	
strokeWidth	color string	
fillColor	color string	
highlightStrokeColor	color string	
highlightFillColor	color string	
labelColor	color string	
visible	**true**,**false**	point and label are visible
fixed	**true**,**false**	dragging possible
draft	**true**,**false**	
trace	**true**,**false**	dragging leaves a trace
withLabel	**true**,**false**	point has a label
name	string	label text this element
id	string	unique id for this element

If not given name and id are chosen automatically.

All properties beside id can be changed during the life time of an object el using the method el.setProperty. There are several formats possible.

```
el.setProperty('key1:value1','key2:value2',...);
el.setProperty([key1:value1],[key2:value2],...);
el.setProperty({key1:value1, key2:value2,...});
```