Workshop: Advanced JSXGraph

Vol. 2

Alfred Wassermann

16-12-2020
Contents

Preliminaries 3
  Include JSXGraph .................................................. 3

Axes / Grid 3
  Axis element ......................................................... 4
  Default axes ......................................................... 5
  Non-standard scaling .............................................. 5

Highlighting of elements 6
  Disable highlighting for specific elements ......................... 6
  Combine a group of elements for highlighting ..................... 6

Groups 7

How is a construction interpreted? 7

Accessing elements in one layer 8

Discussion and suggestion of further topics 8

Next webinar 9
Preliminaries

Include JSXGraph

- JSXGraph skeleton page:

```html
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>JSXGraph template</title>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<link href="https://cdn.jsdelivr.net/npm/jsxgraph@1.1.0/distrib/jsxgraph.css" rel="stylesheet" type="text/css" />
<script src="https://cdn.jsdelivr.net/npm/jsxgraph@1.1.0/distrib/jsxgraphcore.js" type="text/javascript" charset="UTF-8"></script>
<!-- The next line is optional: MathJax -->
<script src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-chtml.js" id="MathJax-script" async"></script>
</head>
<body>
<div id="jxgbox" class="jxgbox" style="width:500px; height:200px;"></div>
<script>
var board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox: [-5, 2, 5, -2]});
</script>
</body>
</html>
```

- See JSXGraph handbook (in development): https://ipesek.github.io/jsxgraphbook/

Axes / Grid

- Grid is mostly obsolete
- `axis: true` creates default axes.
- There is also the element `axis`. It allows to create individual axes (in every direction).
- See also Murray Bourne's take on axes and grids: https://www.intmath.com/cg3/jsxgraph-axes-ticks-grids.php
- More information about ticks: https://jsxgraph.org/wiki/index.php/Ticks
Axis element

- \texttt{https://jsfiddle.net//two.pnumLwdxvtf/2/}

```javascript
const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-5, 5, 5, -5], axis: false
});

board.create('axis', [[0,0], [0,1]], {
    visible: true,
    scalable: true,
    ticks: {
        type: 'linear',
        visible: true,
        drawZero: true,
        majorHeight: -1,
        minorHeight: 0,
        minorTicks: 0,
        label: {
            visible: true,
            anchorX: 'right',
            anchorY: 'middle',
            fontSize: 12,
            offset: [-6, 0]
        }
    },
    name: 'y',
    withLabel: true,
    label: {
        position: 'rt',
        offset: [-20, -10]
    }
});
```

- The axis looks like a grid by setting:

```javascript
board.create('axis', [[0,0], [0,1]], {
    visible: false,
    ticks: {
        visible: true,
        drawZero: true,
        majorHeight: -1,
        minorHeight: 0,
        minorTicks: 0,
        label: {
            visible: false
        }
    }
});
```
Default axes

The default axes which are generated with the attribute `axis: true` can be styled using the board attributes `defaultAxes.x` and `defaultAxes.y`:

```javascript
const board = JXG.JSXGraph.initBoard('jxgbox', {
  boundingbox: [-5, 5, 5, -5],
  axis: true,
  defaultAxes: {
    x: {
      ticks: {
        minorTicks: 0
      }
    },
    y: {
      ticks: {
        tickEndings: [1, 1],
        minorTicks: 4
      }
    }
  }
});
```

- See [https://jsfiddle.net/7jv9z4Ld/](https://jsfiddle.net/7jv9z4Ld/)

Non-standard scaling

- The most prominent example here is to have major ticks (and labels) for multiples of \( \pi \). This can be realized by using the attributes `scale` and `scaleSymbol`, see [https://jsfiddle.net/tqf8vwoc/](https://jsfiddle.net/tqf8vwoc/).

```javascript
const board = JXG.JSXGraph.initBoard('jxgbox', {
  boundingbox: [-15, 5, 15, -5],
  axis: true,
  defaultAxes: {
    x: {
      ticks: {
        scale: Math.PI,
        scaleSymbol: '\u03c0',
        ticksDistance: 1,
        insertTicks: false
      }
    }
  }
});
```

The symbol \( \pi \) is generated by using the UTF-16 code \( 0x03C0 \) for \( \pi \) in the form \( \backslash u03c0 \).
Highlighting of elements

- If the mouse pointer / pen is close to a JSXGraph element, this element will be highlighted or can be dragged.
- The precision can be set with the attribute `JXG.Options.precision`, see https://jsxgraph.org/docs/symbols/JXG.Options.html#precision.
- Starting from version 1.2.0 the precision can be set individually for every element.
- Among the attributes for highlighting are `highlightStrokeWidth`, `highlightStrokeColor`, `highlightStrokeOpacity`, `highlightFill`..., `highlightCssClass`. The latter attribute is for texts and images.

Disable highlighting for specific elements

```javascript
// Turn off highlighting
var line = board.create('line', [[-2, -3], [3, 4]], {highlight: false});
```

See https://jsfiddle.net/ujor/four.pnumxbf/

Combine a group of elements for highlighting

- Combined highlighting of objects seems to be impossible if triggered from board events.
- However, triggering combined highlighting from outside is possible, see https://jsfiddle.net/92re5zo6/1/

```javascript
var p1 = board.create('point', [-3, -2]);
var p2 = board.create('point', [-1, 3]);
var p3 = board.create('point', [0, 1]);
var p4 = board.create('point', [2, -2]);
var line = board.create('line', [p1, p3]);

// JXG.Composition
var c = new JXG.Composition({
  p1: p1,
  p2: p2,
  p3: p3,
  p4: p4
});

function doit(highlight) {
  if (highlight) {
    c.highlight(true);
  } else {
```
• Also possible without Composition element

Groups

• Combine a group of points with the element group.
• Then, dragging one point affects the other points, too.
• There are the following special points:
  – translation points: each point is translation point by default
  – scale points
  – rotation points
• Then there is a
  – scaleCenter
  – rotationCenter
• Centers are given by points, the string 'centroid', or an array with coordinates.

```javascript
var p1 = board.create('point', [-1, -2]);
var p2 = board.create('point', [2, -2]);
var p3 = board.create('point', [2, 2]);
var p4 = board.create('point', [-1, 2]);

var pol = board.create('polygon', [p1, p2, p3, p4], {fillColor: 'yellow'});
var G = board.create('group', [p1, p2, p3, p4]);
G.setRotationCenter(p2).setRotationPoints([p1]);
G.setScaleCenter('centroid').setScalePoints([p3, p4]);
G.setAttribute({size: 8});
```

• See https://jsfiddle.net/jw81koze/1/

How is a construction interpreted?

• Updates a triggered on every pointer move / down / up event.
• All updates run in a single thread.
• Exceptions, where functions are called asynchronously are:
- determine the size of a text element
- animations
- dumpToCanvas
- transitions, see https://jsfiddle.net/jgdwpho/eight.pnum/:

```javascript
var p1 = board.create('point', [-1, -2]);
var p2 = board.create('point', [2, -2]);
var p3 = board.create('point', [2, 2]);
var p4 = board.create('point', [-1, 2]);
var pol = board.create('polygon', [p1, p2, p3, p4], {
    fillColor: 'yellow',
    highlightFillColor: 'blue',
    transitionDuration: 2000,
    hasInnerPoints: true
});
```

- Usually, it is suggested to load MathJax asynchronously. This might lead to problems, if JSXGraph constructions are loaded synchronously.

### Accessing elements in one layer

- Elements in one layer are ordered chronological. That means, the last element is on top.
- This may be changed with the attribute `dragToTopOfLayer`, which places an element on top if dragged.
- See https://jsfiddle.net/y1gf4de8/

```javascript
var li1 = board.create('line', [1, 1, 1], {strokeWidth: 20, dragToTopOfLayer: true});
var li2 = board.create('line', [1, -1, 1], {strokeWidth: 20, strokeColor: 'red', dragToTopOfLayer: true});
```

### Discussion and suggestion of further topics

- Please, make suggestion for a new element `vectorfield` at https://github.com/jsxgraph/jsxgraph/issues/333
- Topics of the January webinar: new features of v1.2.0
- Alpha blendings of colors: use hex rgba string like '#ff000054' or `strokeOpacity: 0.5'.
Next webinar

The next webinar will be **Wednesday, January 20th, 2021 at 4 pm CET**