Workshop

4. International JSXGraph Conference

Alfred Wassermann

UNIVERSITÄT
BAYREUTH

10-10-2023, 9:00am

Contents

October 10 – 12, 2023

UNIVERSITÄT BAYREUTH

1 Introduction

- I'm the JSXGraph lead developer, University of Bayreuth, Germany
- The plan of this workshop is to summarize what happened since October 2022 and - maybe - learn some lesser known features of JSXGraph.

2 Preliminaries

2.1 Information for beginners

- https://youtu.be/0wQPASnq86Y
- JSXGraph handbook: https://ipesek.github.io/jsxgraphbook/
- JSXGraph wiki: https://jsxgraph.org/wiki

2.2 Information for intermediate / advanced users

- Previous conferences:
  - 2020: https://jsxgraph.org/conf

  - 2021: https://jsxgraph.org/conf2021
  - 2022: https://jsxgraph.org/conf2022

- Webinar series (7 sessions) in 2020 / 2021: https://jsxgraph.org/wp/docs/
- Examples with source code: https://jsxgraph.org/share/ and https://jsxgraph.org/wiki/
- API docs: https://jsxgraph.org/docs/

2.3 Include JSXGraph - skeleton page

How to include JSXGraph:

- Local copy of `jsxgraphcore.js` and `jsxgraph.css` (download or npm)
- https://cdn.jsdelivr.net/npm/jsxgraph/distrib/jsxgraphcore.js
- https://cdnjs.cloudflare.com/ajax/libs/jsxgraph/1.4.6/jsxgraphcore.js

```
<!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/distrib/jsxgraph.css" rel="
        stylesheet" type="text/css" />
    <script src="https://cdn.jsdelivr.net/npm/jsxgraph/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>
```

3 Releases since the last conference

- JSXGraph v1.5.0 was released in January 2023
  - Move code base to ES6
  - Keyboard support
  - Individual shadows
  - board logging
- JSXGraph v1.6.0 was released in August 2023
  - Vector fields and slope fields
  - smartlabel
  - Internationalization
  - HTML texts can now be rotated (finally!)
  - `board.setAttribute()`

4 Internationalization

- New in v1.6.0: JSXGraph supports *internationalization of numbers*
- Overview: MDN on Intl.NumberFormat
- All attributes of *Intl.NumberFormat* are supported
- Internationalization can be configured board-wise and individual per element
- *Example:* see https://jsfiddle.net/49z3rdut/8/

```
const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-0.5, 0.5, 0.5, -0.5],
    intl: {
        enabled: false,
        locale: 'de-DE'
    },
    keepaspectratio: true,
    axis: true,
    defaultAxes: {
        x: {
            ticks: {
                intl: {
                        enabled: true,
                        options: {
                            style: 'unit',
                            unit: 'kilometer-per-hour',
                            unitDisplay: 'narrow'
                        }
                }
            }
```

```
        }
    },
    infobox: {
        fontSize: 20,
        intl: {
            enabled: true,
            options: {
                minimumFractionDigits: 4,
                maximumFractionDigits: 5
            }
        }
    }
});

var p = board.create('point', [0.1, 0.1], {name: 'P'});
var txt = board.create('text', [0.1, 0.05, 'X_0(P) = <value>X(P)</value>'], {
    intl: {
        enabled: true,
        locale: 'it-IT'
    }
});

var t = board.create('text', [0.05, 0.2, -Math.PI*100], {
  intl: {
          enabled: true,
          options: {
              style: 'unit',
              unit: 'celsius'
          }
    }
});

var s = board.create('slider', [[-0.45, 0.2], [-0.3, 0.2], [0, 1, 360]], {
    name: '&alpha;',
    snapWidth: 1,
    intl: {
        enabled: true,
        options: {
            style: 'unit',
            unit: 'degree',
        }
    }
});

var t = board.create('text', [0.3, -0.3, ''], {
    intl: {
        enabled: true,
        locale: 'it-IT',
        options: {
            style: 'unit',
            unit: 'kilometer-per-hour',
            unitDisplay: 'narrow',
            maximumFractionDigits: 2
        }
    }
});
```

```
// Set dynamic text consisting of text and number.
t.setText(function() {
    var txt = 'Speed: ',
        number = t.X();

    // Add formated number to variable txt
    if (t.useLocale()) {
        txt += t.formatNumberLocale(number);
    } else {
        txt += JXG.toFixed(number, 2);
    }
    return txt;
});
```

Use with care! It might lead to confusion if number format for input fields is different from the displayed format.

5 JSXGraph in mobile devices

- Many thanks to Murray Bourne who pointed out most of the problems discussed here.
- **Problem:** Scrolling a page inside of a JSXGraph construction on a mobile browser. There might be no margins around the JSXgraph construction and the student is trapped inside of the construction.
- Intermediate solution: board attribute `browserPan` and double tap
- Open https://jsfiddle.net/1uowbvm6/2/ (`browserPan:true`) and https://jsfiddle.net/1uowbvm6/3/ (with `browserPan:false`)

```
const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-5, 5, 5, -5], axis:true,
    pan: {
        enabled: true,
        needTwoFingers: true,
    },
    browserPan: true,
    zoom: {
        enabled: false
    }
});
```

- Recently, mobile browsers introduced `pointerCapture`
- This allows to “drop” points outside of the board
- Example: again https://jsfiddle.net/1uowbvm6/3/ on a mobile device, drag `A` outside of the board and release.

- This is a controversial feature for JSXGraph!
- In v1.6.1 it will be prevented by default and allowed by board attribute `moveTarget:document`
- v1.6.1: Fullscreen mode will react to orientation change

6 JSXGraph as ES6 module

- Version 1.5.0 is a major refactoring of the JSXGraph code base.
- JSXGraph moved from AMD module system to ES6 module system (thanks to Sam Ritchie)
- **Are there any changes for users?**
- If you include JSXGraph by loading `jsxgraphcore.js`: no changes
- If you include / pack JSXGraph as module:
  - JSXGraph is now exported as *unnamed AMD module*
  - CommonJS does work now (again), i.e. in nodejs
  - ES6 import with `jsxgraphcore.mjs` or `src/index.js`
- See https://jsxgraph.org/wp/2023-01-27-release-of-version-1.5.0/

7 Enable users to sketch curves

- This little example was created during the 1st northern e-assessment meeting in Trondheim (2023) in collaboration with Michael Kallweit, Bochum, Germany

Task: Let users sketch a curve or function

- See https://jsfiddle.net/6ckhtzqm/

Add new handling mode

- Three board modes are predefined:
  - `board.BOARD_MODE_NONE = 0x0000`
  - `board.BOARD_MODE_DRAG = 0x0001`
  - `board.BOARD_MODE_MOVE_ORIGIN = 0x0002`
- New user modes can be defined. Please use numbers larger than `0x00f0`!
- These modes can be used in event listeners
- Please, *capture* the event only if mode equals `board.BOARD_MODE_NONE`

```
board.BOARD_MODE_SKETCH = 0x0100;
board.on('down', function(evt) {
    if (board.mode !== board.BOARD_MODE_NONE) {
        return;
    }
    board.mode = board.BOARD_MODE_SKETCH;
    sketch = board.create('curve', [[], []], {
        strokeColor: '#bbbbbb',
        lineCap: 'round',
        strokeWidth: 10
    });
});
```

Collect sketch points

```
// Collect data points for the sketch curve
board.on('move', function(evt, mode) {
    var pos, c;

    if (mode !== board.BOARD_MODE_SKETCH) {
        return;
    }

    pos = board.getMousePosition(evt);
    c = new JXG.Coords(JXG.COORDS_BY_SCREEN, pos, board);
    sketch.dataX.push(c.usrCoords[1]);
    sketch.dataY.push(c.usrCoords[2]);
    board.update();
});
```

Create curve from data points

```
board.on('up', function(evt) {
    var coords = [], i, p;

    if (board.mode !== board.BOARD_MODE_SKETCH) {
        return;
    }

    // Remove previous curve if it exists
    if (JXG.exists(curve)) {
        board.removeObject(curve);
        board.removeObject(points);
    }

    board.mode = board.BOARD_MODE_NONE;

    // Get list of coordinates from sketch curve
    for (i = 0; i < sketch.dataX.length; i++) {
        coords.push(new JXG.Coords(JXG.COORDS_BY_USER, [sketch.dataX[i], sketch.
            dataY[i]], board));
    }

    // Reduce the number of coordinate points to 6 internal coordinate points
```

```
    coords = JXG.Math.Numerics.Visvalingam(coords, 6);

    // Convert the output of Visvalingam to JSXGraph points
    points = [];
    for (i = 0; i < coords.length; i++) {
        points.push(
            board.create('point', [coords[i].usrCoords[1], coords[i].usrCoords[2]],
                 {
                withLabel: false,
                visible: false
            })
        );
    }

    // Create cardinal spline from JSXGraph points
    curve = board.create('curve',
        JXG.Math.Numerics.CardinalSpline(points, () => tau.Value()), {
            strokeColor: '#000000',
            strokeWidth: 3,
            lineCap: 'round',
            fixed: false
        });

    // Remove the sketch curve
    board.removeObject(sketch);
    board.removeObject(points);
});
```

Improvement: sketch polynomial of given degree

- See https://jsfiddle.net/c03qu1ok/
- The only difference is that the Visvalingam algorithm is used to reduce the curve to `degree + 1` points and instead of cardinal spline Lagrange interpolation is used. Further, the defining points are not removed.

```
board.on('up', function(evt) {
    var coords = [],
        i, p;

    if (board.mode !== board.BOARD_MODE_SKETCH) {
        return;
    }

    // Remove previous curve if it exists
    if (JXG.exists(curve)) {
        board.removeObject(curve);
        board.removeObject(points);
    }

    board.mode = board.BOARD_MODE_NONE;

    // Get list of coordinates from sketch curve
    for (i = 0; i < sketch.dataX.length; i++) {
```

```
        coords.push(new JXG.Coords(JXG.COORDS_BY_USER, [sketch.dataX[i], sketch.
            dataY[i]], board));
    }

    // Reduce the number of coordinate points to `degree + 1 - 2` internal
        coordinate points plus start point and end point.
    coords = JXG.Math.Numerics.Visvalingam(coords, degree.Value() - 1);

    // Convert the output of Visvalingam to JSXGraph points
    points = [];
    for (i = 0; i < coords.length; i++) {
        points.push(
            board.create('point', [coords[i].usrCoords[1], coords[i].usrCoords[2]],
                 {
                size: 5,
                withLabel: false,
                visible: true
            })
        );
    }

    // Create Lagrange polynomial from JSXGraph points
    curve = board.create('functiongraph', [JXG.Math.Numerics.lagrangePolynomial(
        points)], {
        strokeColor: '#000000',
        strokeWidth: 3,
        lineCap: 'round',
        fixed: false
    });

    // Remove the sketch curve
    board.removeObject(sketch);
    // board.removeObject(points);
});
```

See also

- https://jsxgraph.org/share/example/sketch-curve
- https://jsxgraph.org/share/example/sketch-polynomial

8 Security

Who needs to be secured?

1. Securing the JSXGraph construction from accidentally overwriting CSS / JavaScript

- The hosting website may be a problem, i.e. there might be a namespace conflict with variables
- **Example:** `<video>` tag can not be changed from outside
- **Solution:** `shadowDOM`, see talk by Holger Engels on Thursday

2. Securing the hosting website from JSXGraph code

Scenario:

- JSXGraph is displayed in a `<div>` element.
- JavaScript code may access user data, i.e. launch a *Cross site scripting attack* (XSS)
- Example: access moodle session cookie, see conference course
- The JSXGraph / JavaScript code of the construction is loaded from an *untrusted site* or students are allowed to upload code
- That is: a malicious content developer may inject dangerous JavaScript into a JSXGraph construction

> JSXGraph in a `div` is secure as long as you trust the authors of the JSXGraph construction. No need to worry then.

Solutions:

- Restrict student input to strings which are parsed by JessieCode
  - Example: https://jsxgraph.org/share/example/function-plotter
- Sandbox the content into an `iframe` element
- Realized in new latest version of moodle /STACK, see talk by Hans Jakob Rivertz on Thursday
- Will be realized in upcoming moodle JSXGraph filter (maybe optional)

8.1 JSXGraph in iframe

- See example https://jsfiddle.net/L71e0cua/2/
- Create secure iframe which executes JSXGraph

```
const iframe = document.createElement('iframe');
iframe.setAttribute('name', attr.iframe.name);
iframe.setAttribute('id', attr.iframe.id);
iframe.setAttribute('class', attr.iframe.class);
iframe.setAttribute('style', attr.iframe.style);
iframe.setAttribute('scrolling', 'no');
iframe.setAttribute('srcdoc', src)
// Security
sandbox = 'allow-scripts';
allow = "fullscreen *;";
csp = 'navigate-to \'none\'; ' +
    'connect-src \'none\'; ' +
    'worker-src \'none\'; ' +
```

```
    'script-src \'unsafe-inline\' \'self\';'
iframe.setAttribute('sandbox', sandbox);
iframe.setAttribute('allow', allow);
iframe.setAttribute('csp', csp);
document.body.appendChild(iframe);
```

- MDN article about iframe sandboxing
- The above example is a solution for *paranoids*: in the iframe it is even forbidden to load JavaScript libraries, JSXGraph is injected via `srcdoc`.
- Allow `allow-scripts` to run JavaScript
- Forbid everything else, in particular `allow-same-origin`

Sandboxing with `allow-scripts` and `allow-same-origin` is no sandboxing!

- Communication with hosting website: message system
- Message handling in iframe:

```
// Receive messages
window.addEventListener("message", (evt) => {
    jsx_iframe_handleMsg(JSON.parse(evt.data));
}, false);

// Send messages
const sendMessage = function(msg_obj) {
    msg_obj.uuid = uuid;
    window.parent.postMessage(JSON.stringify(msg_obj), '*');
    delete msg_obj.uuid;
};
```

- Message handling in host web site:

```
// Receive messages
window.addEventListener("message", (e) => {
  var msg = JSON.parse(e.data);
  ...
});

// Send messages
var win = window.frames[attr.iframe.name];
win.postMessage(JSON.stringify(msg[idx]), '*');
```