Workshop: JSXGraph for beginners

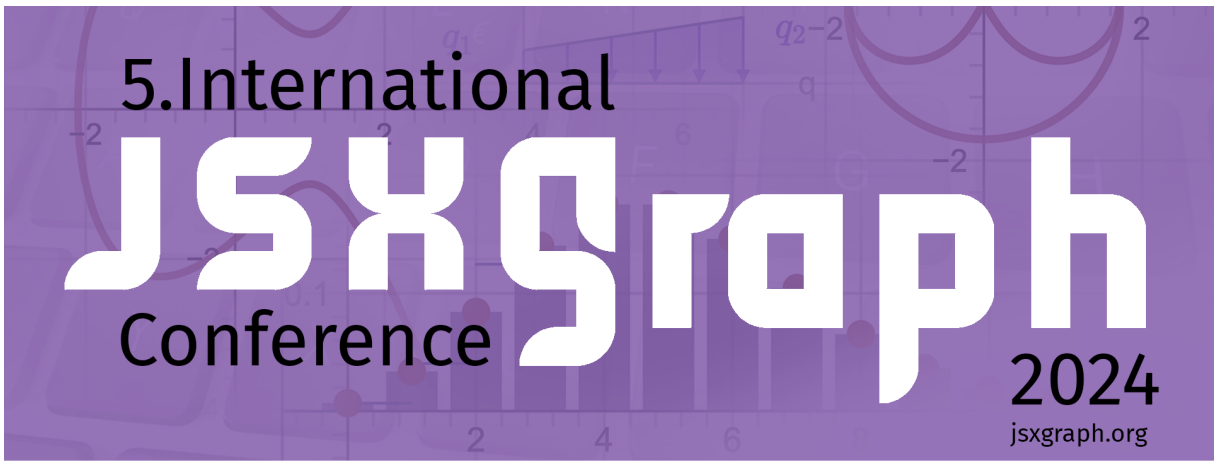
5. International JSXGraph Conference

Carsten Miller, Alfred Wassermann

UNIVERSITÄT
BAYREUTH

08-10-2024, 3:10pm

Contents



October 8 – 10, 2024

UNIVERSITÄT BAYREUTH

1 Preparation

1.1 Minimum requirements needed for JSXGraph

- A **web browser**: Firefox, Google Chrome, Microsoft Edge, Apple Safari. There is no big difference.
- A **text editor**: today we use *Microsoft Visual Studio Code* (its free), but there are countless other free alternatives. However, it should at least support *syntax highlighting*.
  - Download and install *Visual Studio Code* from https://code.visualstudio.com/download
- **Internet connection**

1.2 Helpful but not mandatory

- Basic knowledge about **HTML**
- Basic knowledge about **JavaScript**

> Of course: the more, the better

1.3 More information for beginners

- JSXGraph introduction on youtube
- JSXGraph handbook
- JSXGraph wiki
- JSXGraph examples database
- API documentation

2 Let's start

Our first steps are:

- Download the HTML template file `page1.html` from https://jsxgraph.org/moodle.
- Open the file `page1.html` in your editor. The file should look like this:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>JSXGraph page1</title>
    <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"></
        script>
  </head>
  <body>

  <div id="box" class="jxgbox" style="width:600px; height:600px;"></div>

  <script>
  </script>

  </body>
</html>
```

- Open the file `page1.html` in your web browser

Workflow: type text in editor – save file – reload web page in browser

Online development: https://jsfiddle.net/0k7f8ed1/

3 Use JSXGraph

```
  <body>

  <div id="box" class="jxgbox" style="width:600px; height:600px;"></div>

  <script>
    var board = JXG.JSXGraph.initBoard('box', {boundingbox: [-5, 6, 6, -5]});
  </script>

  </body>
```

- Book section

4 Our first geometry objects: points and lines

4.1 Creating points

```
board.create('point', [-2, 1]);
```

Another point:

```
board.create('point', [-2, 1]);
var q = board.create('point', [3, 0]);
```

- Book section

4.2 Creating lines

```
var line1 = board.create('line', [[-3, 1], [3, -1]]);

var p = board.create('point', [-2, -1]);
var q = board.create('point', [3, 1]);
var line2 = board.create('line', [p, q]);
```

- Book section

4.3 Adding attributes to the objects

```
var p = board.create('point', [-2, -1], {name: 'first', size:5, color: 'blue'});
var q = board.create('point', [3, 1], {name: 'last', fixed:true, face: '[]'});

var line2 = board.create('line', [p, q], {straightLast:false, dash:4 });
```

- Book section

5 Circles

page2.html:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>JSXGraph page2</title>
    <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"></
        script>
  </head>
  <body>

  <div id="box" class="jxgbox" style="width:600px; height:600px;"></div>

  <script>
    var board = JXG.JSXGraph.initBoard('box', {boundingbox: [-5, 6, 6, -5], axis:
        true});
  </script>

  </body>
</html>
```

- Drawing circles:

```
var p = board.create('point', [-1, -1], {name:'A'});
var q = board.create('point', [0, 0], {name:'B'});

var circle1 = board.create('circle', [p, q], {strokeColor:'red', strokeWidth:4});
var circle2 = board.create('circle', ['B', 1.8], {fillColor:'gray', fillOpacity
    :0.2});
```

- Book section

6 Function plotting

page3.html:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>JSXGraph page3</title>
    <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"></
        script>
```

```
  </head>
  <body>

  <div id="box" class="jxgbox" style="width:600px; height:600px;"></div>

  <script>
    var board = JXG.JSXGraph.initBoard('box', {boundingbox: [-5, 6, 6, -5], axis:
        true});
  </script>

  </body>
</html>
```

6.1 Static function graph

We have several possibilities to input a function term.

- Function graph given as string:

```
board.create('functiongraph', ['sin(x)'], {strokeColor: 'black'});
```

The content of the string is in the language *JessieCode* that has been developed specifically for JSXGraph. JessieCode is a full fledged programming language, but it's main purpose is to provide mathematical expressions and enable some symbolic manipulations, see Wigand's talk on Thursday. For more information, see the JessieCode github repository.

- Function graph given as JavaScript function:

```
board.create('functiongraph', [(x) => x * x], {strokeColor: 'red'});
```

In JavaScript, `(x)=> x * x` is called *arrow function*. It is easier to read than the old `function(x){ return x * x; }`.

Input of a function as string is convenient if the input comes from student. The JavaScript way to plot $sin(x)$ would be `Math.sin(x)`, which is inaccessible to students. Example: online function potter.

- See Book chapter

6.2 "Dynamic" function graph

In the next step we want to allow the students to manipulate the graph. For this we need a `slider` element.

```
var s = board.create('slider', [[-4, 5], [-1, 5], [-10, 1, 10]], {name: 'a'});
board.create('functiongraph', [
    (x) => s.Value() * x * x
], {strokeColor: 'red'});
```

We can use sliders also in function graphs given as strings. But now we have to use the `name` of the slider (`a`), not the JavaScript variable name `s`.

```
var s = board.create('slider', [[-4, 5], [-1, 5], [-10, 1, 10]], {name: 'a'});
board.create('functiongraph', [
    'a.Value() * x * x'
], {strokeColor: 'red'});
```

- See Book chapter

6.3 Text elements – static and dynamic

```
board.create('text', [4, 3, 'Text'], {fontSize: 16});

board.create('text', [4, -3,
    () => 'f(2) = ' + s.Value() * 2 * 2
], {fontSize: 16});
```

JSXGraph texts also support MathJax and KaTeX.