

---

# **Workshop on dynamic graphics in JSXGraph**

Workshop

Alfred Wassermann, Andrey Chesnokov

25-4-2023

# Contents

<b>1</b>	<b>Presentation</b>	<b>1</b>
1.1	Getting support . . . . .	1
1.1.1	Information for beginners . . . . .	1
1.1.2	Information for intermediate / advanced users . . . . .	1
1.2	Feature tour . . . . .	1
<b>2</b>	<b>Workshop - JSXGraph standalone</b>	<b>3</b>
2.1	Workshop environment . . . . .	3
2.2	Include JSXGraph – skeleton page . . . . .	3
2.3	Basic concepts in JavaScript . . . . .	4
2.3.1	Variables . . . . .	5
2.3.2	Math functions . . . . .	5
2.3.3	Functions . . . . .	5
2.3.4	Arrays and objects . . . . .	6
2.3.5	Events . . . . .	6
2.4	JSXGraph . . . . .	6
2.4.1	The board . . . . .	7
2.4.2	Generate a point . . . . .	7
2.4.3	More on attributes . . . . .	7
2.4.4	Some dynamic geometry . . . . .	8
2.4.5	Using MathJax in JSXGraph . . . . .	9
2.4.6	Tangent example . . . . .	9
2.4.7	Math syntax and dynamic MathJax . . . . .	10
<b>3</b>	<b>Workshop - JSXGraph in STACK</b>	<b>11</b>
3.1	Move point to given coordinates . . . . .	11
3.2	Locate stationary point on function graph . . . . .	12



# 1 Presentation

## 1.1 Getting support

### 1.1.1 Information for beginners

- Video tutorial: <https://youtu.be/0wQPASnq86Y>
- JSXGraph programming book: <https://ipesek.github.io/jsxgraphbook/>
- JSXGraph wiki <https://jsxgraph.org/wiki> with many examples together with source code

### 1.1.2 Information for intermediate / advanced users

- API docs: <https://jsxgraph.org/docs/>
- Webinar series (7 sessions) in 2020 / 2021: <https://jsxgraph.org/wp/docs/>
- Conferences talks and workshops (with lots of information about *JSXGraph in STACK*)
  - 2020: <https://jsxgraph.org/conf>
  - 2021: <https://jsxgraph.org/conf2021>
  - 2022: <https://jsxgraph.org/conf2022>
  - Upcoming 2023: <https://jsxgraph.org/conf2023>

## 1.2 Feature tour



All these examples come with source code

- [Static plots](#)
- [Interactive \(dynamic\) plots](#)
- [Calculus](#)
  - [functions](#)
  - [Lagrange polynomial](#)

- Taylor polynomial
  - curves
  - sequences and series
  - Riemann sums
  - differential equations
  - splines
  - polygonal chain
- Geometry
  - Euclidean geometry
  - Analytic geometry
  - Projective geometry (offside line in football)
- Statistics:
  - boxplots,
  - Regression I
  - Regression II
- Turtle graphics
- Dynamic MathJax
- Video embedding
- Clipping
- Animations
- 3D
- Upcoming: vector and slope fields

## 2 Workshop - JSXGraph standalone

### 2.1 Workshop environment

Recommended development process

- 1) use a text editor (like [visual studio code](#)) to create an HTML file and open it in a web browser
  - save the file with ending `.html`
  - to open the file in a web browser, click on the file name in the *windows file explorer* or *Mac OS finder*
  - or install the *open in browser* plug-in of visual studio code
- 2) or develop online in [jsfiddle](#)
  - Click on [save](#) to get a permanent URL of the construction. (do not forget to store the URL somewhere)
  - A simple example: <https://jsfiddle.net/tzsy184q/>



Always keep the *web console* (developer tools) open in the web browser to be informed about errors.

### 2.2 Include JSXGraph – skeleton page

**Listing 2.1:** How to include JSXGraph

```
<!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">

    <!-- Include JSXGraph -->
    <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>

<!-- Include MathJax - optional -->
<script src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-ctml.js" id="
  MathJax-script" async></script>
</head>
<body>
<h1>My first JSXGraph page</h1>
<div id="jxgbox" class="jxgbox" style="width:600px; height:600px;"></div>
<script>
  var board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox: [-5, 2, 5, -2]});
</script>

</body>
</html>

```

- The template file at <https://jsfiddle.net/tzsy184q/1/>

Alternative locations of 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/1.5.0/jsxgraphcore.js>



Embedding JSXGraph directly from <https://jsxgraph.org/distrib/jsxgraphcore.js> is possible but not recommended

There are two essential parts:

1. The HTML element that contains a JSXGraph construction:

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

2. The program logic:

```

<script>
  var board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox: [-5, 2, 5, -2]});
</script>

```

## 2.3 Basic concepts in JavaScript

- JavaScript is embedded with `<script> ... </script>`
- Syntax is related to C.
- Data types:

- Boolean (**true**, **false**)
  - String
  - Number (floats and integers are not distinguished)
  - Function
  - Object, Array
- Strings can be enclosed in '...' or "..."

### 2.3.1 Variables

#### Listing 2.2: Declaration of variables

```
var a = 1,
    b, c; // Scope is the surrounding function

let i = 1;
// Scope is the surrounding block
for (let i = 0; i < 10; i++) {
    console.log(i, i * i);
}
```



Using neither `var` nor `let` to declare a variable is possible. The variable will then be global. This should be avoided and may be suppressed in some environments. Put the expression `"use strict";` at the beginning of the JavaScript code to be warned (in the console) about global variables and other problems with JavaScript.

### 2.3.2 Math functions

- Math functions, see [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Math](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math)

#### Listing 2.3: Math functions

```
console.log(Math.cos(0));
```

### 2.3.3 Functions

- Functions can be treated like any other variables

```
var square = function(x) {
    return x * x;
}
```



```
var cube = (x) => x * x * x;  
console.log(square(3), cube(3));
```

### 2.3.4 Arrays and objects

- Arrays are zero based

```
var arr = [1, 2, 3, 4];  
console.log(arr[0], arr.length);
```

- Objects can be compared to *associative arrays* in PHP or *dictionaries* in python.

```
var obj = {  
  color: 'green',  
  fillColor: 'yellow'  
};  
  
console.log(obj.color, obj['fillColor']);
```

### 2.3.5 Events

- JavaScript is *event based*
- Events can be supplied with eventlistener / eventhandler, see <https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener>
- Available events: <https://developer.mozilla.org/en-US/docs/Web/Events>

```
<button id="mybutton">Click me</button>
```

```
document.getElementById('mybutton').addEventListener('click', function(evt) {  
  alert('hello');  
});
```

## 2.4 JSXGraph

- Documentation: <https://jsxgraph.org/docs/>
- Basic example: <https://jsfiddle.net/WigandR/hgyz32oe/>

### 2.4.1 The board

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

### 2.4.2 Generate a point

```
var p = board.create('point', [2, 1]);
```

```
var p = board.create('point', [2, 1]);
var p2 = board.create('point', [1, 2], {
  name: 'p_2',
  color: 'blue',
  label: {
    color: '#0000ff',
    fontSize: 32
  },
  showInfobox: false
});
```

### 2.4.3 More on attributes

#### Listing 2.4: Local change

```
var p = board.create('point', [2, 2], {showInfobox: false});
```

#### Listing 2.5: Board wide change

```
board.options.point.showInfobox = false;
var p = board.create('point', [2,2]);
```

#### Listing 2.6: Page wide change

```
JXG.Options.point.showInfobox = false;

const board = JXG.JSXGraph.initBoard('jxgbox', {
  boundingbox: [-5, 5, 5, -5], axis:true
});
var p = board.create('point', [2, 2]);
```

### 2.4.4 Some dynamic geometry

- point

```
var p = board.create('point', [-2, 1]);
var p2 = board.create('point', [
    () => p.X(),
    function(x) { return p.Y() + 1; }
], {
    name: 'p_2',
    color: 'blue',
    label: {
        color: '#0000ff',
        fontSize: 32
    },
    showInfobox: false
});
```

- <https://jsfiddle.net/tzsy184q/3/>

```
for (let i = 0; i < 100; i++) {
    board.create('point', [
        (Math.random() - 0.5) * 8, (Math.random() - 0.5) * 8
    ], {name: ''});
}

for (let i = 0; i < 100; i++) {
    board.create('point', [
        function() {return (Math.random() - 0.5) * 8; },
        () => (Math.random() - 0.5) * 8
    ], {name: '', color: 'blue'});
}
```

- <https://jsfiddle.net/tzsy184q/4/>
- line, intersections, circle, polygon, clipping

```
var p = board.create('point', [-2, 1]);
var p2 = board.create('point', [
    () => p.X(),
    function(x) { return p.Y() + 1; }
], {
    name: 'p_2',
    color: 'blue',
    visible: function() {
        if (p.X() < 0) {
            return false;
        } else {
            return true;
        }
    },
    label: {
        color: '#0000ff',
```

```

        fontSize: 32
    },
    showInfobox: false
});

var li = board.create('line', [p, p2]);
var ci = board.create('circle', [p, p2]);
var p3 = board.create('glider', [0, 3, ci]);

var pol = board.create('polygon', [p, p2, [5, 2], [3, -1]]);
var ci2 = board.create('circle', [[0, -2], 3]);
var fill = board.create('curveintersection', [pol, ci2], {fillColor: 'yellow'})
;

var but = document.getElementById('mybutton');
but.addEventListener('click', function(evt) {
    board.removeObject(p2);
});

```

- <https://jsfiddle.net/tzsy184q/5/>

### 2.4.5 Using MathJax in JSXGraph

- Add <https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-ctrl.js>
- Attribute `useMathjax: true` and double backslash e.g. `\\(\\frac{a}{b}\\)`

### 2.4.6 Tangent example

- See <https://jsfiddle.net/WigandR/hgyz32oe/>

```

const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-1, 4, 5, -4],
    keepaspectratio: false,
    axis: true
});

var a = board.create('slider', [
    [0, -2.5],
    [4, -2.5],
    [0, 2, 3]
], {
    name: 'a;'
});

var c = board.create('slider', [
    [0, -3.5],
    [4, -3.5],
    [0, 1, 5]
], {name: '&omega;'});

```

```

var fun = function(x) {
  return a.Value() * Math.sin(c.Value() * x );
}

var graph = board.create('functiongraph', [fun, -10, 10], {
  strokeColor: '#00ff00'
});

var p1 = board.create('glider', [0, 0, graph], {name: 'P'});
var t = board.create('tangent', [p1]);

```

### 2.4.7 Math syntax and dynamic MathJax

- If supplied as string, usual math syntax can be used for function terms
- See <https://jsfiddle.net/fhx804rv/>

```

var board = JXG.JSXGraph.initBoard('jxgbox', {
  boundingbox: [-5, 5, 8, -5],
  axis: true
});

var a = board.create('slider', [
  [0, -2],
  [5, -2],
  [0, 2, 3]
], {
  name: 'a'
});

var c = board.create('slider', [
  [0, -3],
  [4, -3],
  [0, 1, 5]
], {name: 'c'});

var graph = board.create('functiongraph', ['a * sin(c * x)']);
var p1 = board.create('glider', [0.5, 0, graph], {name: 'P'});
var t = board.create('tangent', [p1]);

var txt = board.create('text', [-3, 3, function() { return p1.X(); }]);
var txt2 = board.create('text', [-4, -3,
  function() { return 'MathJax: \\(\\frac{' + p1.X().toFixed(2) + '{' +
    p1.Y().toFixed(2) + '}\\)'; },
  {useMathjax: true}]);

```

## 3 Workshop - JSXGraph in STACK

Two possible assessment questions.

### 3.1 Move point to given coordinates

- Documentation: [https://github.com/math/moodle-qtype\\_stack/blob/master/doc/en/Authoring/JSXGraph.md](https://github.com/math/moodle-qtype_stack/blob/master/doc/en/Authoring/JSXGraph.md)
- question variables:

```
/* Coordinates to be found */
xCoord:rand_with_step(-9,9,1);
yCoord:rand_with_step(-9,9,1);
point:[xCoord,yCoord];

/* Initial coordinates of point */
xCoordinit:rand_with_step(-9,9,1);
yCoordinit:rand_with_step(-9,9,1);
point_init:[xCoordinit,yCoordinit];

ta: [xCoord,yCoord];
```

- question text:

```
<p>Place the point P at the coordinates ({@stack_disp_comma_separate(point)}).</p>

[[jsxgraph width="500px" height="500px" input-ref-ans1='ans1Ref']]
var board = JXG.JSXGraph.initBoard(divid,
    {boundingbox : [-6, 6, 6, -6], axis:true, shownavigation : false});

var p1 = board.create('point', {#point_init#}, {name : 'P', snapToGrid: true,
    showInfobox: false});
stack_jxg.bind_point(ans1Ref, p1);
board.update();
[[/jsxgraph]]
<p>[[input:ans1]]</p><p>[[validation:ans1]]</p>
```

- Question note: {@ta@}

**Input ans1:**

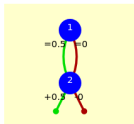
- Input type: numerical
- Model answer: `[xCoordinit, yCoordinit]`

### Potential response tree:

PRT feedback style Standard

Feedback variables

This potential response tree will become active when the student has answered: `ans1`



Node	Answer test	SAns	TAns	Test options	Quiet	No
Node 1	NumRelative	ans1[1]	xCoord	0.1	Quiet	No
Node 1 when true	Mod =	Score 0.5	Penalty	Next	Node 2	Answer note prt1-1-T
Node 1 true feedback	x-coordinate is fine HTML format					
Node 1 when false	Mod =	Score 0	Penalty	Next	Node 2	Answer note prt1-1-F
Node 1 false feedback	Check x-coordinate HTML format					
Delete node 1						
Node 2	NumRelative	ans1[2]	yCoord	0.1	Quiet	No
Node 2 when true	Mod +	Score 0.5	Penalty	Next	[stop]	Answer note prt1-2-T
Node 2 true feedback	y-coordinate is fine HTML format					
Node 2 when false	Mod -	Score 0	Penalty	Next	[stop]	Answer note prt1-2-F
Node 2 false feedback	Check y-coordinate HTML format					
Delete node 2						

Figure 3.1: Response tree

## 3.2 Locate stationary point on function graph

- question variables:

```
omega:rand([1/4,1/2,3/4,1,5/4,3/2,7/4,2]);
fun:3*sin(omega*x);
dfundx:diff(fun,x);
```

- question text:

```
<p>Place the point P at a stationary point at the function graph.</p>
[[jsxgraph width="500px" height="500px" input-ref-ans1='ans1Ref']]
var board = JXG.JSXGraph.initBoard(divid,
  {boundingbox : [-1, 4, 5, -4], axis:true, shownavigation : false});
```

```

var funtxt='{#fun#}';
var graph = board.create('functiongraph', [funtxt, -10, 10], {
    strokeColor: '#00ff00'});

var p1 = board.create('glider', [0,0,graph], {style:6, name:'P'});
var t = board.create('tangent', [p1]);
stack_jxg.bind_point(ans1Ref,p1);

board.update();
[[/jsxgraph]]
<p>[[input:ans1]] </p><p>[[validation:ans1]]</p>

```

- feedback variables: `slope:ev(dfundx,x=ans1[1])`

▼ Potential response tree: prt1

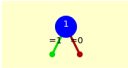
Question value

Auto-simplify

PRT feedback style

Feedback variables

This potential response tree will become active when the student has answered: ans1



Node 1	Answer test	NumAbsolute	SAns	slope	TAns	0	Test options	0.1	Quiet	No
Node 1 when true	Mod	=	Score	1	Penalty		Next	[stop]	Answer note	prt1-1-T
Node 1 true feedback	The slope is close to zero. $\sqrt{f'(x)}= slope $									
	HTML format									
Node 1 when false	Mod	=	Score	0	Penalty		Next	[stop]	Answer note	prt1-1-F
Node 1 false feedback	The chosen point is not close to an stationary point. $\sqrt{f'(x)}= dfundfSAnsVal $									
	HTML format									

**Figure 3.2:** Response tree



