

ISSN 2942-1373

9

October 7 – 9, 2025

6th International JSXGraph Conference

Book of Abstracts

University of Bayreuth
Center for Mobile Learning with Digital Technology

CMLE publications

Publisher	University of Bayreuth Center for Mobile Learning with Digital Technology (CMLDT) Universitätsstraße 30 95447 Bayreuth Germany +49 921 55 3266
Series	CMLDT publications
ISSN	2942-1373
Editor	Carsten Miller, carsten.miller@uni-bayreuth.de Alfred Wasserman, alfred.wassermann@uni-bayreuth.de
License	CMLDT publications © 2023 by University of Bayreuth – Center for Mobile Learning with Digital Technology is licensed under CC BY-ND 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/
Layout	Carsten Miller, carsten.miller@uni-bayreuth.de
Internet	https://mobile-learning.uni-bayreuth.de/CMLDTpublications
Issue	9
Year of publication	2025
Title	6 th International JSXGraph Conference – Book of Abstracts



6th International JSXGraph Conference 2025

University of Bayreuth
Center for Mobile Learning with Digital Technology
95440 Bayreuth
Germany

Conference

The 6th International JSXGraph Conference took place from 7th until 9th of October 2025. The online format encouraged fruitful discussion and collaboration among users from all over the world. The schedule respected the location and the timezone of the speakers. The conference was organized by Carsten Miller and Alfred Wassermann from the University of Bayreuth, Germany.

Conference topics

- Usage of JSXGraph
 - for learning / teaching
 - e-Learning environments: moodle, ilias, STACK, ...
 - dynamic visualizations
- Best practices
- Tools
- Presentation of new JSXGraph developments

Website

The abstracts of the talks at the 6th International JSXGraph Conference are also available on the JSXGraph website:

<https://jsxgraph.org/conf2025>

Videos

The recorded videos of the talks can be found on JSXGraph's YouTube Channel:

<https://www.youtube.com/@jsxgraph4224>

Playlist “6th International JSXGraph Conference”

<https://www.youtube.com/playlist?list=PLr10cPSXxWJf-o5YKPjnR13sUPmxGTxop>

Talks at the 6th International JSXGraph Conference

Konstantina Zerva

- 7 [HELM – STACK workbooks: adding interactivity on vector calculus topics](#)

Andreas Maurischat

- 8 [Exercise pool of the HM4mint consortium](#)

David Flenner

- 9 [Utilizing Browser Debugging Tools with JSXGraph Applications](#)

Carsten Miller, Alfred Wassermann

- 10 [Workshop: JSXGraph for beginners](#)

Wigand Rathmann

- 11 [How a Boat Rotates in a River: Visualizing Rotation in Flow Fields](#)

Frauke Sprengel

- 12 [Interactive Graph Algorithm Learning with JSXGraph and Moodle STACK](#)

Pedro A. García-Sánchez, Mattia Ducci, Lukas Schölch

- 13 [Audiofunctions+: exploring math functions through sonification](#)

Alfred Wassermann

- 17 [Advanced workshop I & II](#)

Tom Berend

- 18 [Simplified JSXGraph – JSXGraph with VSCode scaffolding](#)

André Dietrich, Sebastian Zug

- 21 [JSXGraph & LiaScript ... a perfect Match](#)

Sebastian Tabares Amaya

23 [Sr Smith: A Web-Based RF Smith Chart App using JSXGraph and JessieCode](#)

HELM – STACK workbooks: adding interactivity on vector calculus topics

Konstantina Zerva

k.zerva@ed.ac.uk

The University of Edinburgh, UK

HELM (Helping Engineers Learn Mathematics) is a collection of 50 workbooks developed by five English universities between 2002 and 2005. HELM resources cover the basic engineering mathematics and statistics teaching for first and second year mathematics courses for engineering undergraduates.

During the summers of 2020 and 2021, the universities of Edinburgh and Loughborough undertook an effort of embedding the HELM materials into STACK. The result was a collection of quizzes, each serving as interactive workbooks complete with textbook-style content, detailed worked examples, tasks of varying complexity, and practice questions. Almost 2/3 of the HELM workbooks have been embedded into STACK, mainly the ones covering the first-year engineering teaching. In the previous year, I was awarded funding through the Principal's Teaching Award Scheme at the university of Edinburgh, to continue the translation of HELM into STACK. The project was carried out during the summer of 2025 by two undergraduate interns, who mainly worked on 3 workbooks related to vector calculus.

This talk provides an update on the ongoing translation of the HELM workbooks into STACK. It highlights the developments that took place over the summer and shares the experiences of undergraduate students engaging with JSXGraph for the first time.

Exercise Pool of the HM4mint consortium

Andreas Maurischat

maurischat@combi.rwth-aachen.de

RWTH Aachen University

Aachen, Germany

Besides the online course hm4mint.nrw, the consortium HM4mint has initiated a pool for electronic exercises including several graphical exercises.

In this talk, we present examples of such exercises, we explain how you can import these exercises into your local LMS, and how you can adapt the exercises to your individual style.

Utilizing Browser Debugging Tools with JSXGraph Applications

David Flenner

flennerdr@charleston.edu

College of Charleston

Department of Mathematics

66 George St Charleston SC 29424

Bug-free code is something we all strive for, but there are bound to be unexpected situations that we don't account for when developing a JSXGraph app for the first time. This talk will be a continuation of my presentation from the JSXGraph conference last year, which focused on best practices when first getting started with creating applications using JSXGraph. This year, I will focus on how to utilize tools within your browser to debug problems that exist in your code, as it is inevitable that bugs will exist. Knowing how to best utilize the available tools provided in your browser can help you find these bugs and eliminate them. I will demonstrate the creation of a basic app that has a bug and show how these browser tools can help identify where the bug exists and correct the code. The intended audience is mathematics instructors at all levels who are interested in app development for their classes, but it can be useful for any developer who is unfamiliar with these tools.

Workshop: JSXGraph for beginners

Carsten Miller

carsten.miller@uni-bayreuth.de

Alfred Wassermann

alfred.wassermann@uni-bayreuth.de

University of Bayreuth, Germany

This workshop will cover the basics of working with JSXGraph. The nice thing about the JSXGraph is that you do not need to be a programmer to use it. In the workshop, we will talk about the tools and how to prepare the working environment. By constructing the examples, we will cover some basic objects as points, lines, circles, intersections, and angles.

Throughout the workshop, we will use the [JSXGraph book](#), which has been created in the European Erasmus+ project [ITEMS](#) and is freely available on the internet.

How a Boat Rotates in a River: Visualizing Rotation in Flow Fields

Wigand Rathmann

wigand.rathmann@fau.de

Friedrich-Alexander-Universität Erlangen-Nürnberg

Department Mathematik, Erlangen, Germany

and

Center of Mobile Learning with Digital Technology,

University of Bayreuth, Bayreuth, Germany

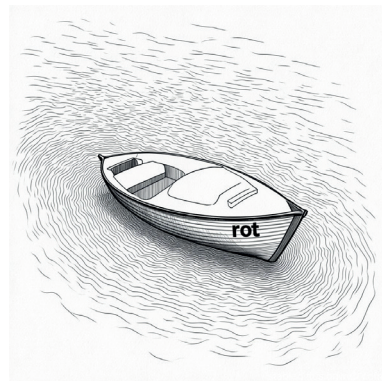
We all know rivers and can imagine, to sit a boat enjoying sun and clouds at the sky and we are drifting along a river not using any paddel.

Sounds like vacation and the rotation of the boat brings as the `curl`-Operator. Thus we wake up back having vector fields in mind thinking about a boat became a single point and moving as a particle

through the vector field along a curve k with an angle velocity induced by the fluid.

In engineering maths vector fields and vector analysis are demanding concepts. Vector fields are considered in \mathbb{R}^2 and \mathbb{R}^3 and are (easy) to visualize using JSXGraph. To find a way to visualize $\text{curl } V(\mathbf{x}) = \nabla \times V(\mathbf{x})$ of a vector field $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ is very demanding.

In this talk I consider 2D vector fields $V : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ and will demonstrate a visualization of the rotating angle of particle/point along a curve $k : \mathbb{R} \rightarrow \mathbb{R}^2$.



Generated using Stable Diffusion

Interactive Graph Algorithm Learning with JSXGraph and Moodle STACK

Frauke Sprengel

frauke.sprengel@hs-hannover.de

Department of Computer Science

Faculty IV - Business and Computer Science

Hannover University of Applied Sciences and Arts

Hannover, Germany

Teaching fundamental graph algorithms such as Kruskal's minimum spanning tree, Dijkstra's shortest path, and depth-first/breadth-first search often relies on static materials that do not adequately convey the interactive nature of algorithmic decision-making. This presentation describes a framework for creating interactive graph algorithm exercises using JSXGraph integrated with Moodle's STACK question type system.

The implementation generates randomized graph instances with configurable parameters: vertex count, edge density, and weights. JSXGraph is used to visualize the graph, for mouse interactions, hover effects, and undo functionality. Students interact with the graph visualization by clicking edges in the correct algorithmic sequence and receive visual feedback through color-coded selections and progression numbering.

The assessment system validates both the resulting spanning tree structure and the order of edge selections. Both of them are not unique, allowing for multiple correct solutions. The randomized questions allow repeated practice while preventing solution sharing.

This is joint work with Shahab Abtahi, B.Sc.

Audiofunctions+: exploring math functions through sonification

Pedro A. García-Sánchez

pedro@ugr.es

joint work with Mattia Ducci, Ondřej Nečas, Lukas Schölch

Departamento de Álgebra and IMAG

University of Granada, E-18071, Spain

Audiofunctions+ is a tool that facilitates the learning and teaching of mathematical functions for visually impaired students through sonification. It is one of the deliverables of the Erasmus+ Key Action 2 project SONAIRGRAPH, coordinated by the University of Turin, with participation from the Karlsruhe Institute of Technology (KIT), the GIS Community Association of Timisoara, and the Universities of Masaryk, Granada, and Valle d'Aosta.

Audiofunctions was created in 2014 at the University of Milan as a native iOS app to allow the exploration of functions on touch screens. The Polin laboratory of the University of Turin and the EveryWare Lab at the University of Milano, based on their research on the educational needs of students with visual disabilities in schools, extended Audiofunctions to Audiofunctions.web as a web application to be used in an educational context. This new version was made available at <https://ewserver.di.unimi.it/audiofunctions>. Further research, interviews, and user feedback, gave rise to Audiofunctions.web 2.0, which is accessible at <https://audiofunctions.netlify.app>. The first beta version of Audiofunctions+ is available at <https://audiofunctions-plus.netlify.app>.

As its predecessor, Audiofunctions+ is an open source and freely available web application that is meant for teaching mathematical functions in secondary school. Research will be conducted on both the usability and methodological aspects of this tool, in order to determine the impact in teaching and learning mathematical concepts related to the study of functions for visually impaired students. Audiofunctions+ can be used in any computer via a browser.

The teacher can prepare working sessions for their students, and can be shared via link (or a json file).

Functions are inserted in text fields via an algebraic expression. Piecewise functions are described via algebraic expressions and inequalities where these expressions apply. The user can change the default view (the box in which the function will be represented) by using the WASD-keys for pan, and Z or shift-Z for zoom. The app plays a sound depending on the position of the cursor and the value of the function at the x-coordinate of the cursor.

If f is the function to be represented, then the pitch of the sound depends on $f(x)$, the value of f at the x coordinate of the cursor. The x coordinate itself influences the placement of the sound in a stereo panorama (the sound will be reproduced on the left ear if we are at the leftmost position of the screen and will increasingly move to the right ear as we move to the rightmost position in the screen). If the exploration is made via the mouse, the volume of the sound depends on the distance between y and $f(x)$, where y is the y -coordinate of the mouse cursor. Keyboard exploration can be either stepwise or smoothly with the arrow keys. The user can play the whole representation of the function in the current view box by pressing the key “b”.

The app offers two kinds of sonification: continuous and discrete. While the continuous sonification produces a sound constantly, the discrete sonification will only play a sound if the y -value of the function varies significantly to the last sonified position. A guitar sound is used for discrete sonification, while a clarinet is used by default for continuous sonification.

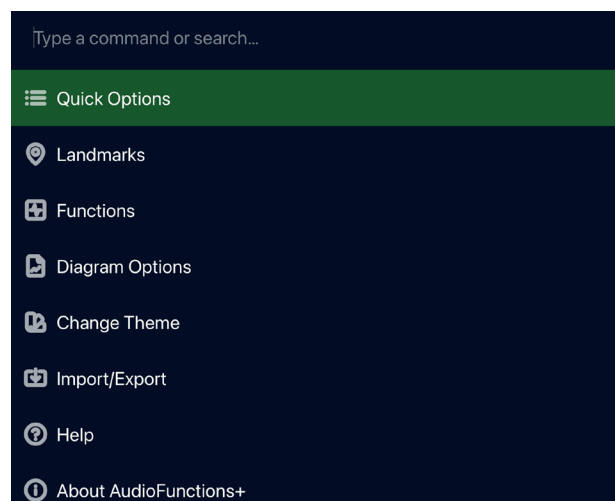
The app offers several earcons to help the navigation. When the user reaches the end of the view box (either at the right or left), a sound is played. Another sound is produced if, for a given x -coordinate, $(x, f(x))$ is not in the current view box. When the user crosses the y -axis with the cursor, another earcon is produced. A gentle noise is added to the sound when the value of the function is negative.

To ensure that important function points, such as isolated points, are not accidentally missed during sonification, there is the “Point of Interest” feature. This

ensures that the corresponding point cannot simply be skipped over with the cursor. When a point is skipped during step-by-step navigation, a notification earcon is played to alert the user that a “Point of Interest” has been missed. In smooth navigation, gentle snapping has been implemented for these points. The “Points of Interest” are recognized and set automatically by AudioFunctions+. Currently, these are limited to isolated points and undefined points of a function. In principle, however, teachers can define any desired points of interest in the JSON file.

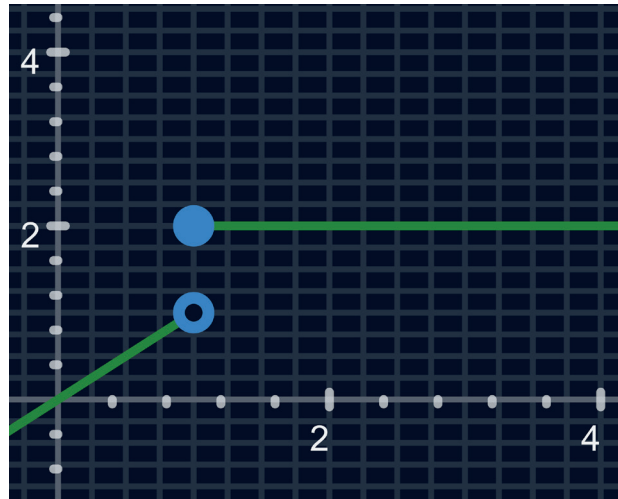
With this functionality, an auditory equivalent to visual representations of specific points is to be achieved. This is because visual representations are usually clearly recognizable with a big dot, even though they represent only a very specific point in the function. The snapping plays that functionality for the sonification.

To enable fast access to all functionality via the keyboard, a CommandPalette and shortcuts were implemented. The CommandPalette can be opened at any time, using the shortcut ctrl+k (cmd+k on Mac) to search for actions. A fuzzy search with alternative keywords is used so that users do not need to memorize exact action names. Additionally, all actions can be navigated like a list with sublists within the CommandPalette.



Associated to constraints in a piecewise function, there is a set of “endpoints”, which are the points in the graph of the function determined by the ends of

the intervals associated with the constraints. For instance, the function $f(x)=x$ if $x<1$ and $f(x)=2$ if $x\geq 1$, produces two endpoints, one hollow and the other one filled out:



Functions and endpoints are drawn using JSXGraph. Sonification is performed with tone.js . The math parsing of the algebraic expressions and constraints for piecewise functions is carried out with the help of math.js . JessieCode is used to evaluate expressions. Pieceswise functions are translated recursively to javascript conditional expressions “ $C ? A : B$ ” and then passed to JessieCode. The user can have several functions defined in one session, but only the active one will be displayed and sonified. There are keyboard shortcuts to switch between functions, so that the student can compare their properties easily.

The authors are supported by the SONification for Accessible and Inclusive Representation of GRAPHS in Education (SONAIRGRAPH) project, which is an Erasmus+ key action 2 (project number 2024-1-IT02-KA220-HED-000244481), funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Advanced workshop I & II

Alfred Wassermann

alfred.wassermann@uni-bayreuth.de

University of Bayreuth, Germany

In two workshops we will cover more advanced topics of JSXGraph programming. The topics will mostly be oriented on changes that happened in JSXGraph since the conference in 2024, that is since version v1.10.0:



- Progress in 3D
- 3D polyhedra
- ARIA
- Extension programming

Simplified JSXGraph – JSXGraph with VSCode scaffolding

Tom Berend

Canada

Simplified JSXGraph
JSXGraph with VSCode scaffolding

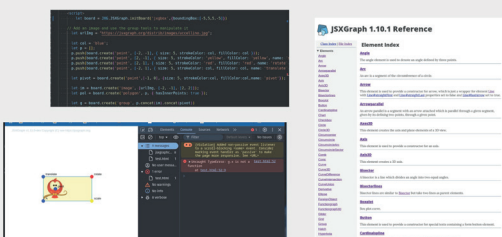


Tom Berend

board.create() constructs 120+ elements
Many of these also have overloads

```
11  
12 var f1 = board.create('glider', [2, 0, board.defaultAxes.x], {name: 'f1'});  
13 var f2 = board.create('glider', [2, 0, board.defaultAxes.x], {name: 'f2'});  
14 var ell = board.create('ellipse', [f1, f2, [0,3]]);  
15  
16 var P = board.create('glider', [-1, 2, ell], {name: 'p'});  
17 var s1 = board.create('segment', [f1,P]);  
18 var s2 = board.create('segment', [f2,P]);  
19  
20 var txt = board.create('text', [0, 2, 4,  
21 () => "|p| + |p'| = " + P.Dist(f1).toFixed(2) + ' + ' + P.Dist(f2).toFixed(2)  
22 ]]);
```

Workflow Requires Editor, Console, & API
How we did things 20 years ago...



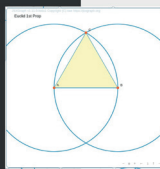
'Minimal example' is terrifying
But I just want...




- Requires webserver setup.
- Requires a programmer editor.
- Requires understanding file system and terminal commands.
- Requires JavaScript and HTML
- Requires knowing how the internet works

Same Engine – Only Replacing create()
Constructors for each element

```
let TSX = new TSXBoard('html03');  
TSX.Text([-4.5, 4.5], 'Euclid 1st Prop', { fontSize: 15 })  
  
// problem - create equilateral on this segment  
let a = TSX.Point([-2, 0], { name: 'A' })  
let b = TSX.Point([2, 0], { name: 'B' })  
TSX.Segment(a, b)  
  
// solution  
let c1 = TSX.Circle(a, b)  
let c2 = TSX.Circle(b, a)  
let c = TSX.Intersection(c2, c1, { name: 'C' })  
TSX.Polygon([a, b, c])
```



A Thin TypeScript Wrapper over JSXGraph
Calls board.create() with types



Keeps JSXGraph Coding Style
Works the way you expect.

Attribute is always last

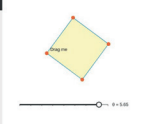
```
let origin = TSX.Point([0, 0], { name: 'Origin' })
```

Multiple signatures

```
TSX.Angle(p1, p2, p3) // angle from 3 points  
TSX.Angle(l1, l2, [5, 0], [5, 5]) // two lines, two directions  
TSX.Angle(l1, l2, 1, -1, { radius: 2 }) // two lines, two +/- values
```

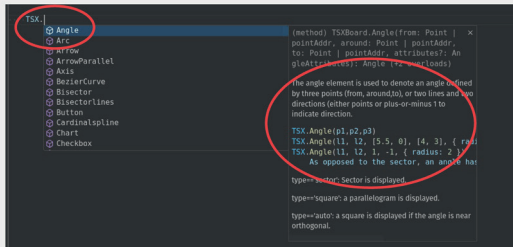
Fix Legacy Design - eg: Translations
Some decisions didn't age well.

```
// create a transformation that rotates around p1  
let t1 = board.create('transform', [() => slider.Value(), p1], { type: 'rotate' })  
let t2 = TSX.Rotate(() => slider.Value(), p1) // new
```



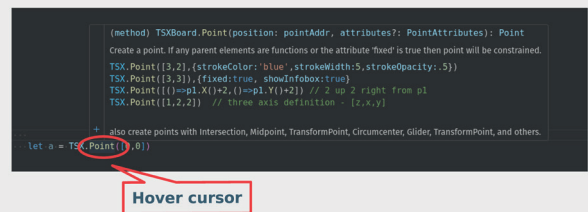
Completions expose JSXGraph Elements

Explore beyond the basic elements...



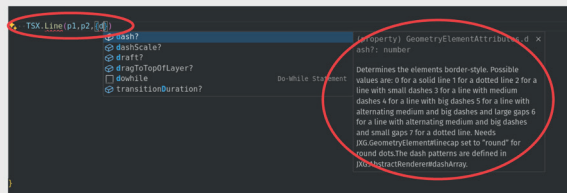
Hover for Context Help with Examples

Also: Hover over variables to see their type



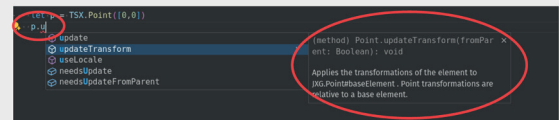
Completions expose Attributes

Was it 'dashscale' or 'dashScale'?



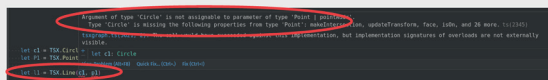
Completions expose Methods and Fields

The function you need is already there



Catch Most Errors during Edit

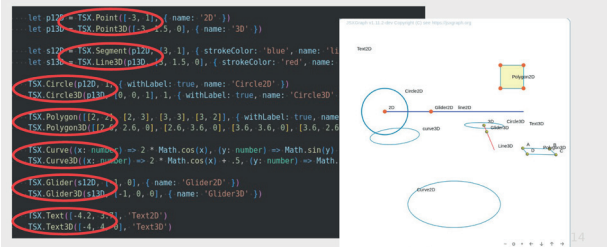
Almost never need console debug



Interactive syntax checking and type checking.
Also linting, refactoring, prettifying.

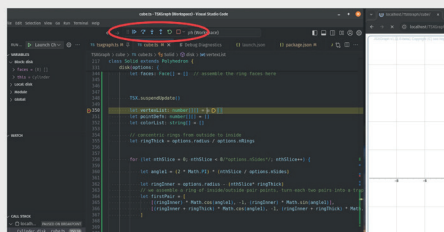
Unified 2D and 3D views

Don't need 'View' anymore



VSCode Debug with Breakpoints

Stop using `console.log()`



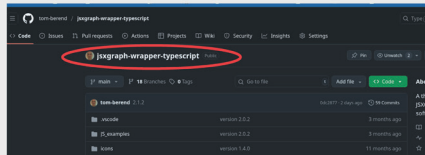
Supports JavaScript and TypeScript

Same syntax check, discovery, completions...



TypeScript adds type safety, interfaces, and improved tooling. Better for larger projects.

Github Quick-Start includes Webserver ...ready to go in 10 seconds



```
> git clone ....
> npm i
> npm run start
```

Then browse to <http://localhost:3000>

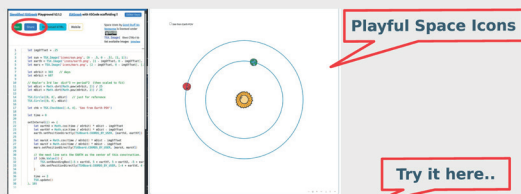
17

Or Just Run in the Playground ...ready to go in 0 seconds



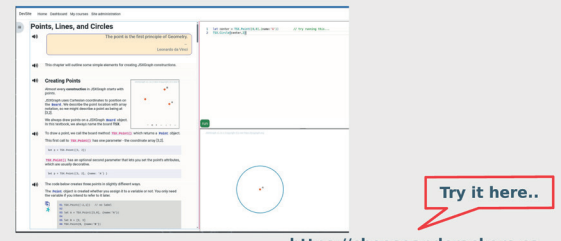
18

Explore, Create, Share, Get Feedback ...lets students show off their awesomeness !



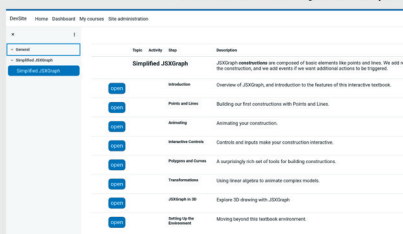
<https://cheeseandcrackers.ca/playground/?script=TPHOG9C07>

Simplified JSXGraph Interactive Textbook Assumes basic JavaScript



<https://cheeseandcrackers.ca>

Simplified JSXGraph Interactive Textbook Assumes basic JavaScript



<https://cheeseandcrackers.ca>

Try Simplified JSXGraph for your next project

- Delivers modern tooling
- 100% compatible - wraps board.create()
- 100% coverage - attributes & methods from documentation
- Works with JavaScript or TypeScript

22

JSXGraph = "LOGO for High-School" Why I wrote this package

- 'Math-like' - leads students to math ideas
- Basic Elements controlled by Functions
- Simple 2D graphics - ideal for writing games
- 'Real Programming' - not Scratch
- Interactive Geometry - unique capabilities
- GUI elements included



Missing: "Low Floor - High Ceiling"

23

Simplified JSXGraph JSXGraph with VSCode scaffolding

JSXGraph

Visual Studio Code

Tom Berend

24

JSXGraph & LiaScript ... a perfect Match

André Dietrich, Sebastian Zug

TU Bergakademie Freiberg

Bernhard-von-Cotta-Straße 2

09599 Freiberg

LiaScript is an extension for markdown which was especially developed as a language for easy and decentralized creation of interactive online courses - in short, a language for OER. It adds various features to markdown that were previously missing, such as the integration of multimedia content, integration of Oembed sites, the execution of script-tags, animations in combination with text to speech output to generate a mixture between screencast and interactive textbooks. It does not require an additional compilation step, since the content is directly interpreted within the browser. Moreover, other markdown dialects often lack the capacity to extend the language, in LiaScript this is possible with an additional macro-syntax which allows the integration and mixing of external JavaScript, CSS, plus the LiaScript markdown language. We therefore have added a JSXGraph extension, which is in itself a Markdown file that only has to be imported into the header of a document. This read-me is itself a self describing documentation. Afterwards, it is possible to use different jsx extensions directly within the codeblock, which are indicated by @JSX.Graph.

```
JXG.Options.slider.snapValues = [-5, -2, -1, 0, 1, 2, 5];
JXG.Options.slider.snapValueDistance = 0.2;

var a = board.create('slider', [[2, -5], [7, -5], [-5, 1, 5]], { name: 'a' });
var b = board.create('slider', [[2, -6], [7, -6], [-5, 0, 5]], { name: 'b' });
var c = board.create('slider', [[2, -7], [7, -7], [-5, 0, 5]], { name: 'c' });

var f = board.create('functiongraph', [(x) => a.Value() * x * x + b.Value() * x + c.Value()]);

var txt = board.create('text', [-9, -5,
    () => JXG.Math.Numerics.generatePolynomialTerm([c.Value(), b.Value(), a.Value()], 2, 'x', 2)
], { fontSize: 18 });
```

As an alternative, if you want to allow users to inspect the code, simply use the @JSX.Script macro, which will be highlighted differently. This allows the user to double-click or double-tab on “result of the script”, which will reveal the code that was required to generate the output. Thus, users can inspect, modify and experiment with the code to get a deeper understanding.

JSXGraph delivers robust, low-latency visualizations; LiaScript supplies the narrative glue, assessment scaffolding, and friction-free distribution channel. Together they create living, explorable documents in which theory, code, and visual feedback co-evolve in real time — no build chains, no vendor lock-in, and zero installation overhead. Authors write once; learners tinker anywhere, from a smartphone on the bus to an offline and remote feature-phone. This synergy lowers the entry barrier for content creators, turns static lessons into interactive laboratories, and ultimately bridges the gap between explanation and exploration — exactly the kind of experience we believe the JSXGraph community strives for.

Sr Smith: A Web-Based RF Smith Chart App using JSXGraph and JessieCode

Sebastian Tabares Amaya

me@syta.co

2 Tian Jin Li Gong Da Xue Qing Nian Jiao Shi He Xue

Zhe Gong Yu, Xi Qing Qu, Tian Jin Shi

China, 300387

Sr Smith is a powerful web application that reinvents the classic Smith Chart for the digital age. Built with the high-performance JSXGraph library and the intuitive JessieCode DSL, it provides RF engineers, professors and students with an interactive platform for circuit analysis and design, with a graphical experience similar to GeoGebra.

This app allows you to visualize and design complex RF circuits in a dynamic environment. You can plot and see circuit transformations in real time, making it easy to analyze impedance matching networks, S-parameters, RF amplifiers, transmission lines, and other great RF applications and modern use of the Smith Chart. This goes beyond static old-school diagrams, giving you a live and colorful, visual understanding of your designs.

Here's the cool part: Sr Smith uses JessieCode. With this simple yet powerful language, the app lets you turn your graphical designs into a repeatable, parametric workflow. You can easily tweak a value in the code or in the canvas, and watch your entire design update instantly. It's the best of both worlds? The visual ease of a graphical chart with the power and precision of code. Unlike traditional Smith Chart software that relies on fixed components like capacitors and inductors, the core principle of Sr Smith is purely geometrical. This app is built on a foundation of fundamental primitives like points, lines, circles and curves. This has two advantages: You're no longer limited by predefined

component libraries. You can create any graphical construction on the complex plane, both visually and through code, allowing you to explore design possibilities that are impossible with other tools. And by focusing on the geometric nature of the Smith Chart, Sr Smith promotes a deeper learning experience. It helps users truly grasp the visual and conceptual representations of RF principles, which is the entire point of using a graphical tool like the Smith Chart in the first place. Essentially, Sr Smith treats the Smith Chart as a dynamic canvas for geometric exploration, giving you unmatched freedom and a clearer understanding of your designs. It isn't just a calculator; it's a complete design tool that bridges the gap between traditional graphical methods and modern, code-driven engineering.