Simplified JSXGraph

JSXGraph with VSCode scaffolding

Tom Berend

board.create() constructs 120+ elements

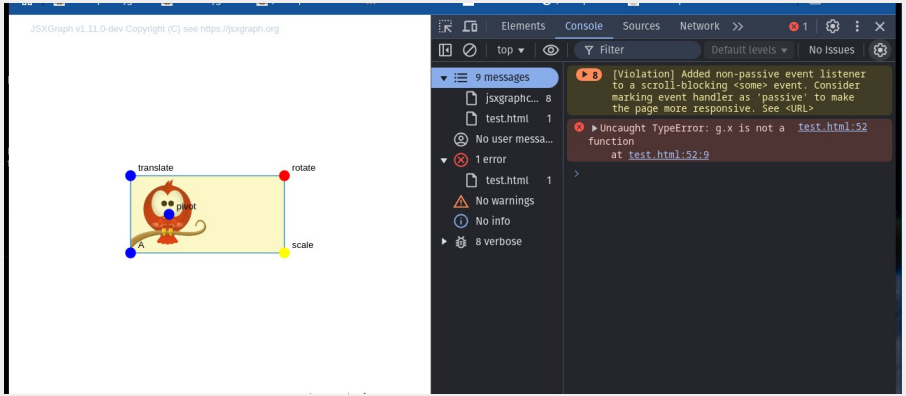
Many of these also have overloads

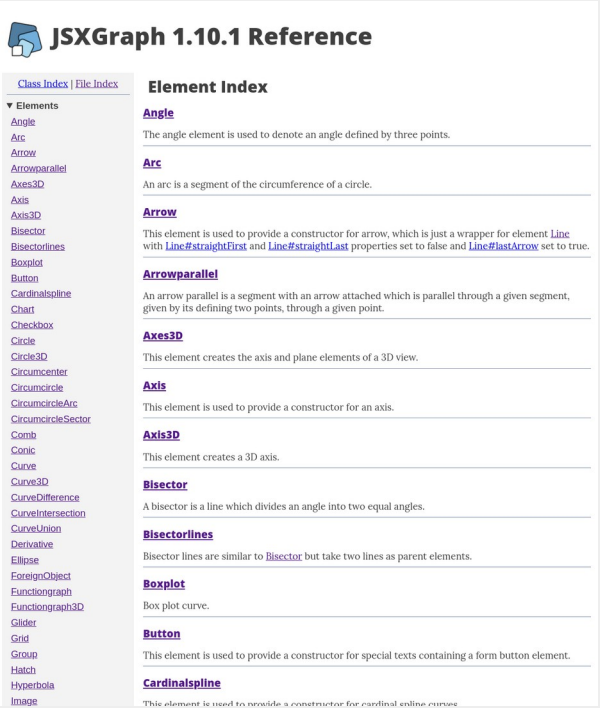
```
11
12  var f1 = board.create('glider', [-2, 0, board.defaultAxes.x], {name:"f'"});
13  var f2 = board.create('glider', [2,  0, board.defaultAxes.x], {name:"f"});
14  var ell = board.create('ellipse', [f1, f2, [0,3]]);
15
16  var P = board.create('glider', [-1, 2, ell], {name: 'p'});
17  var s1 = board.create('segment', [f1,P]);
18  var s2 = board.create('segment', [f2,P]);
19
20  var txt = board.create('text', [0.2, 4,
21      () => "|pf| + |pf'| = " + P.Dist(f1).toFixed(2) + ' + ' +  P.Dist(f2).toFixe
22  ]);
```

Workflow Requires Editor, Console, & API

How we did things 20 years ago...

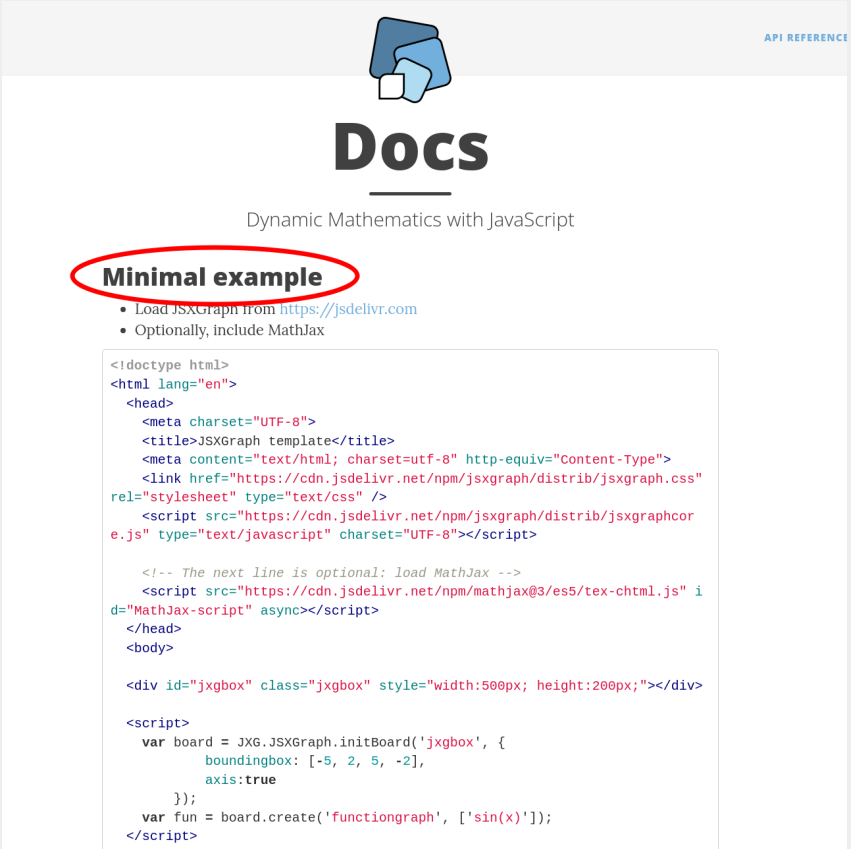
```
<script>
    let board = JXG.JSXGraph.initBoard('jxgbox',{boundingBox:[-5,5,5,-5]})

// Add an image and use the group tools to manipulate it
    let urlImg = "https://jsxgraph.org/distrib/images/uccellino.jpg";

    let col = 'blue';
    let p = [];
    p.push(board.create('point', [-2, -1], { size: 5, strokeColor: col, fillColor: col }));
    p.push(board.create('point', [2, -1], { size: 5, strokeColor: 'yellow', fillColor: 'yellow', name: 's
    p.push(board.create('point', [2, 1], { size: 5, strokeColor: 'red', fillColor: 'red', name: 'rotate'
    p.push(board.create('point', [-2, 1], { size: 5, strokeColor: col, fillColor: col, name: 'translate'

    let pivot = board.create('point',[-1, 0], {size: 5, strokeColor:col, fillColor:col,name: 'pivot'});

    let im = board.create('image', [urlImg, [-2, -1], [2, 2]]);
    let pol = board.create('polygon', p, { hasInnerPoints: true });

    let g = board.create('group', p.concat(im).concat(pivot))
```





‘**Minimal example**’ is terrifying

But I just want…



API REFERENCE

Docs

Dynamic Mathematics with JavaScript

Minimal example

- Load JSXGraph from https://jsdelivr.com
- Optionally, include MathJax

```
<!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/jsxgraphcor
e.js" type="text/javascript" charset="UTF-8"></script>

    <!-- The next line is optional: load MathJax -->
    <script src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-chtml.js" i
d="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],
            axis:true
        });
    var fun = board.create('functiongraph', ['sin(x)']);
  </script>
```

Requires webserver setup.

Requires a programmer editor.

Requires understanding file system and terminal commands.

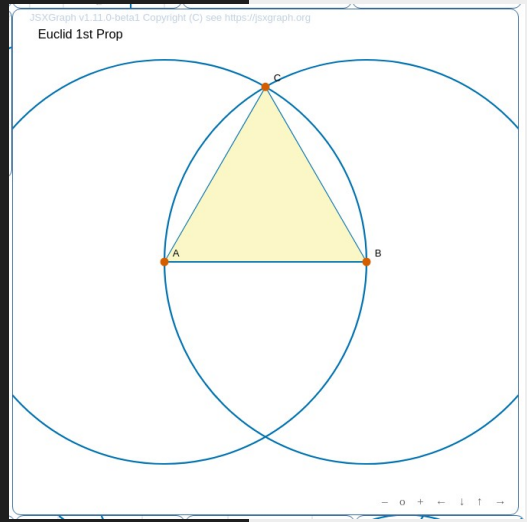
Requires JavaScript and HTML

Requires knowing how the internet works

Same Engine – Only Replacing **create()**

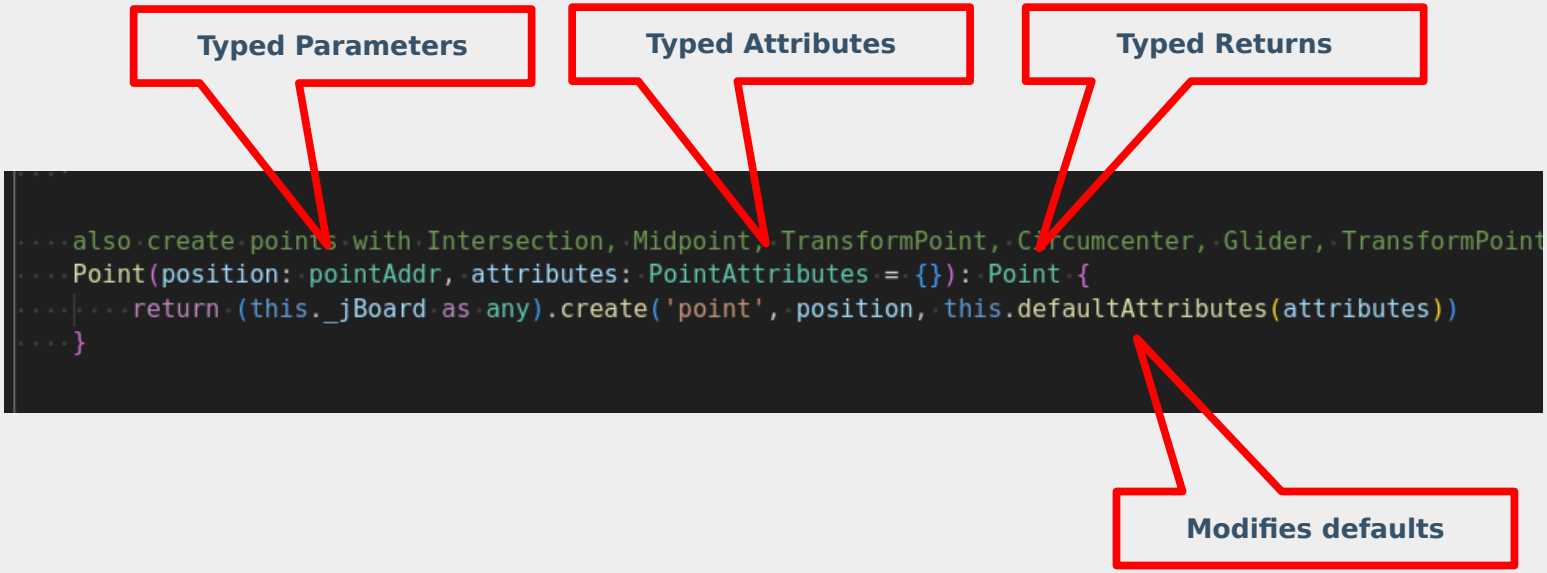
Constructors for each element

```
let TSX = new TSXBoard('html03',)

TSX.Text([-4.5, 4.5], 'Euclid 1st Prop', { fontSize: 15 })

// problem - create equlateral 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' })
```

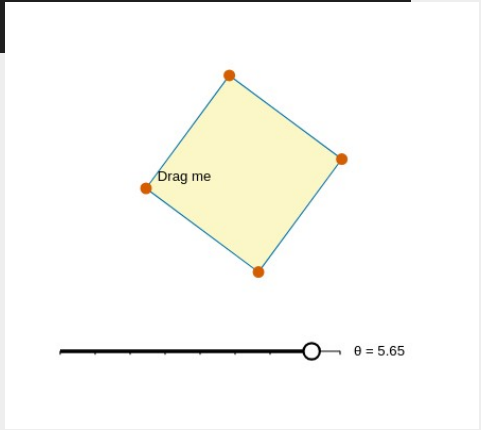
Multiple signatures

```
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

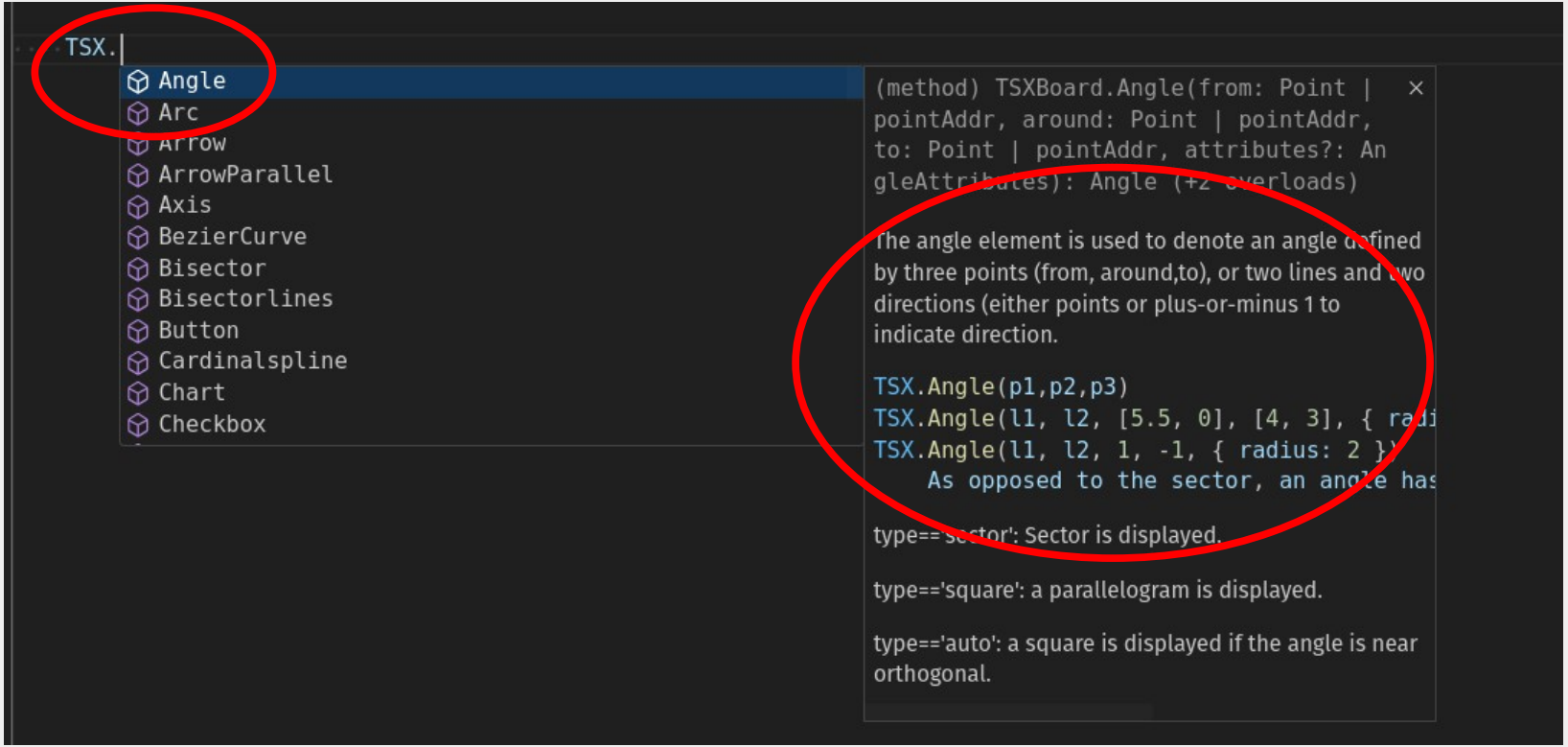
Some decisions didn't age well.

```
// 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
```



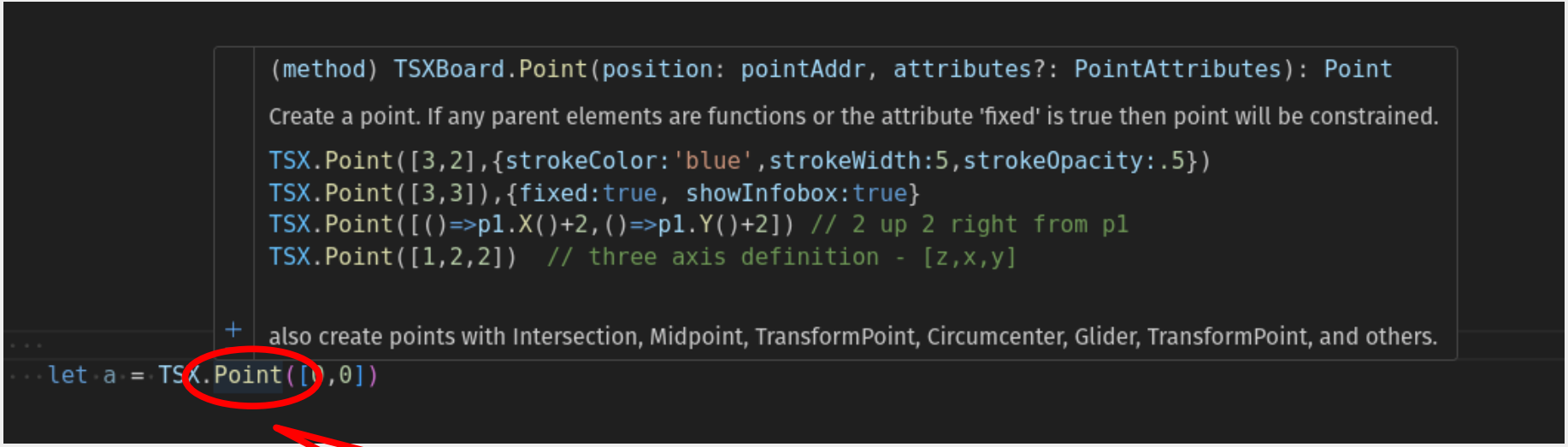
Completions expose JSXGraph Elements

Explore beyond the basic elements...



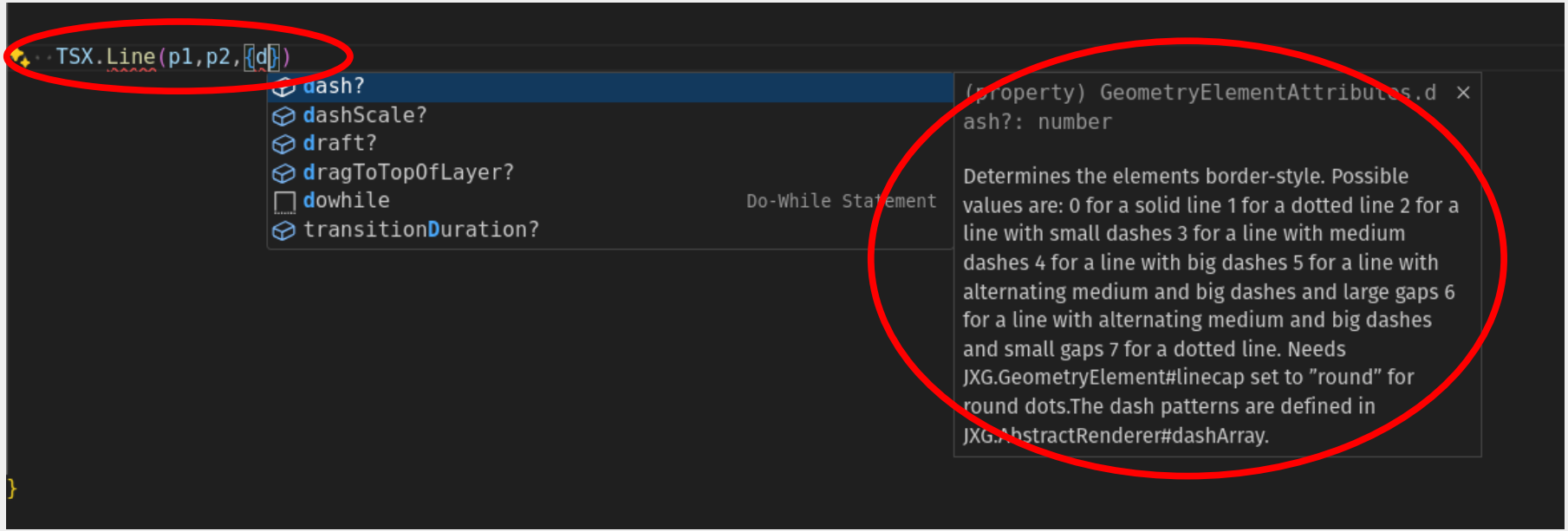
Hover for Context Help with Examples

Also: Hover over variables to see their type



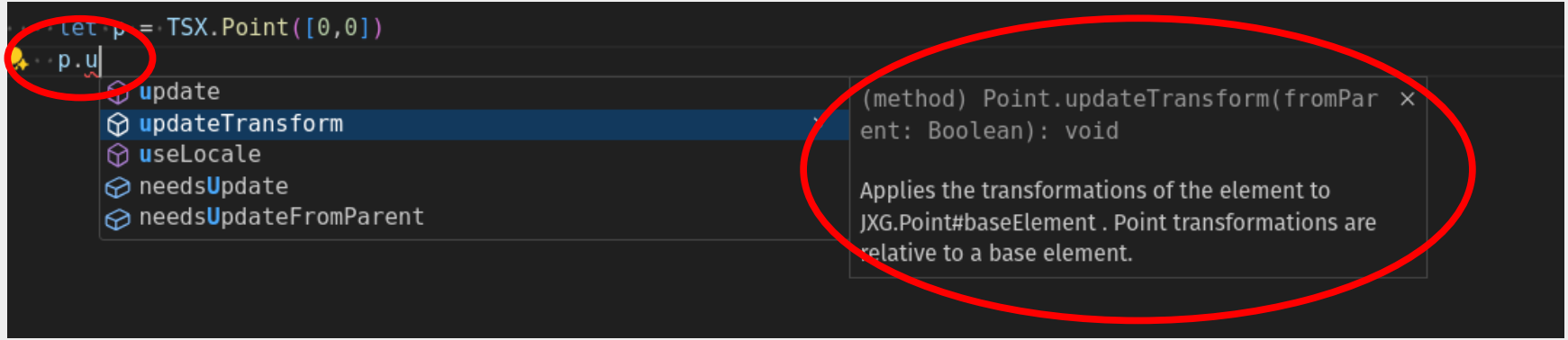
Completions expose Attributes

Was it ‘dashscale’ or ‘dashScale’?



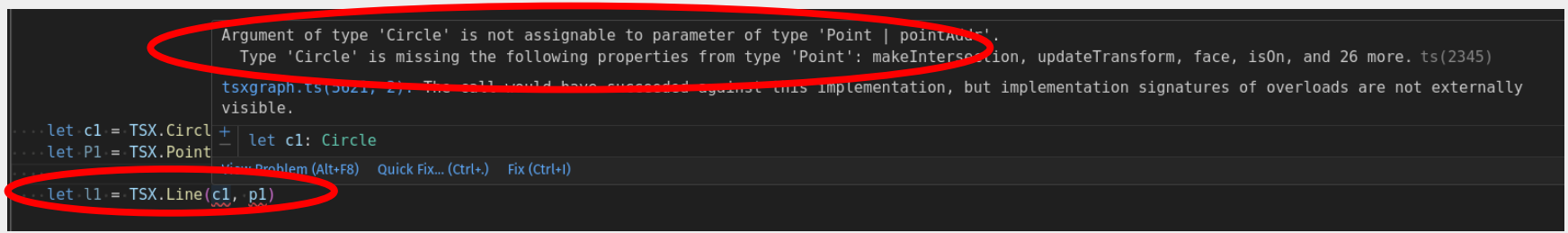
Completions expose Methods and Fields

The function you need is already there



Catch Most Errors during Edit

Almost never need console debug

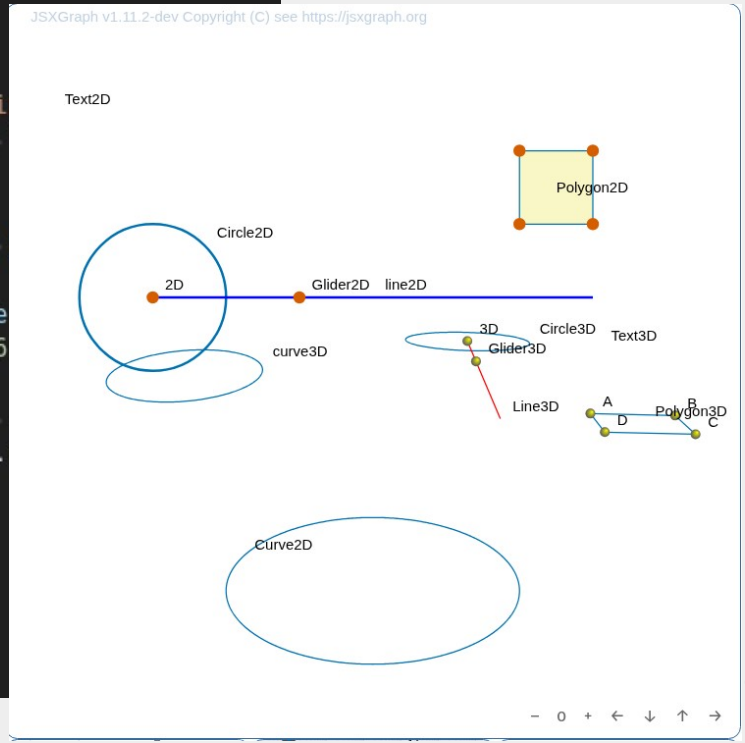


Interactive syntax checking and type checking.
Also linting, refactoring, prettifying.

Unified 2D and 3D views

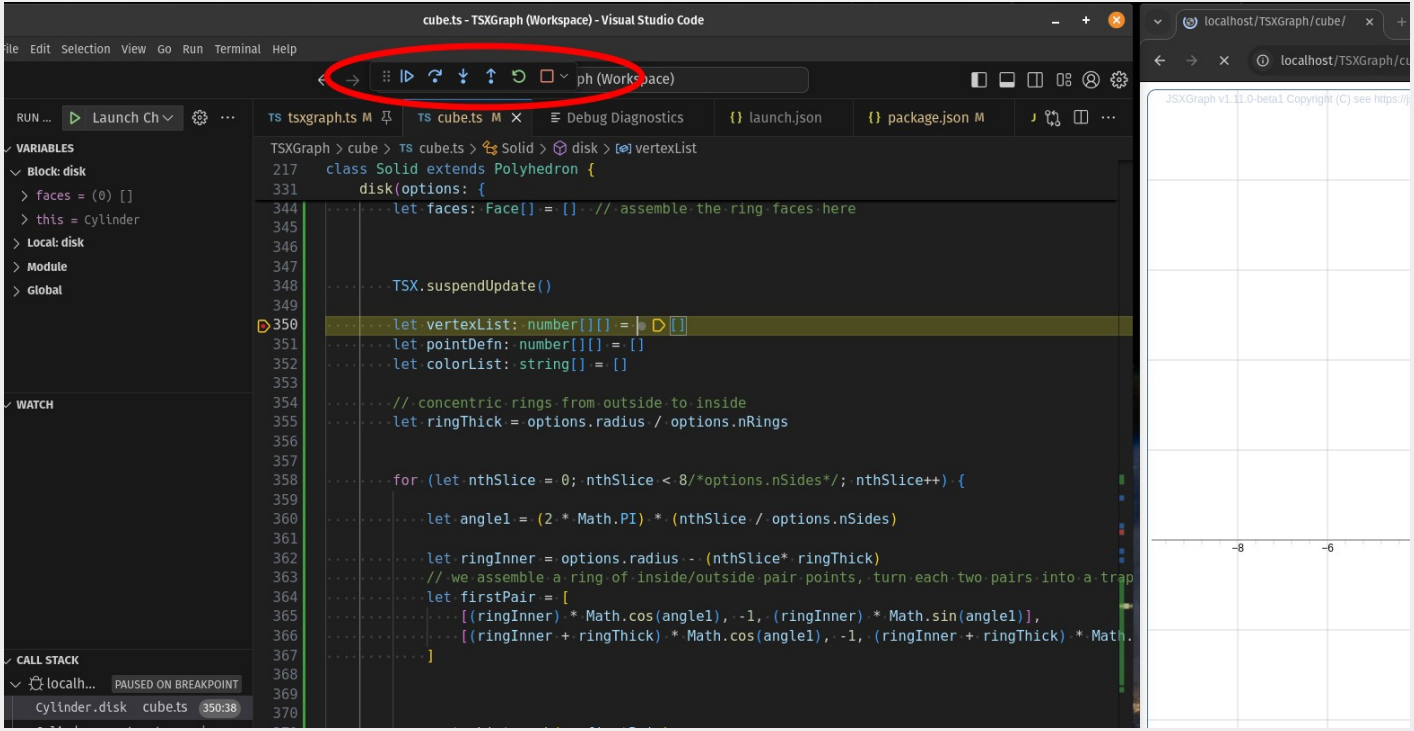
Don’t need ‘View’ anymore

```
let p12D = TSX.Point([-3, 1], { name: '2D' })
let p13D = TSX.Point3D([-3, 1.5, 0], { name: '3D' })

let s12D = TSX.Segment(p12D, [3, 1], { strokeColor: 'blue', name: 'li
let s13D = TSX.Line3D(p13D, [3, 1.5, 0], { strokeColor: 'red', name:

TSX.Circle(p12D, 1, { withLabel: true, name: 'Circle2D' })
TSX.Circle3D(p13D, [0, 0, 1], 1, { withLabel: true, name: 'Circle3D'

TSX.Polygon([[2, 2], [2, 3], [3, 3], [3, 2]], { withLabel: true, name
TSX.Polygon3D([[2.6, 2.6, 0], [2.6, 3.6, 0], [3.6, 3.6, 0], [3.6, 2.6

TSX.Curve((x: number) => 2 * Math.cos(x), (y: number) => Math.sin(y)
TSX.Curve3D((x: number) => 2 * Math.cos(x) + .5, (y: number) => Math.

TSX.Glider(s12D, [-1, 0], { name: 'Glider2D' })
TSX.Glider3D(s13D, [-1, 0, 0], { name: 'Glider3D' })

TSX.Text([-4.2, 3.7], 'Text2D')
TSX.Text3D([-4, 4, 0], 'Text3D')
```



VSCode Debug with Breakpoints

Stop using console.log()



Supports JavaScript and TypeScript

Same syntax check, discovery, completions...

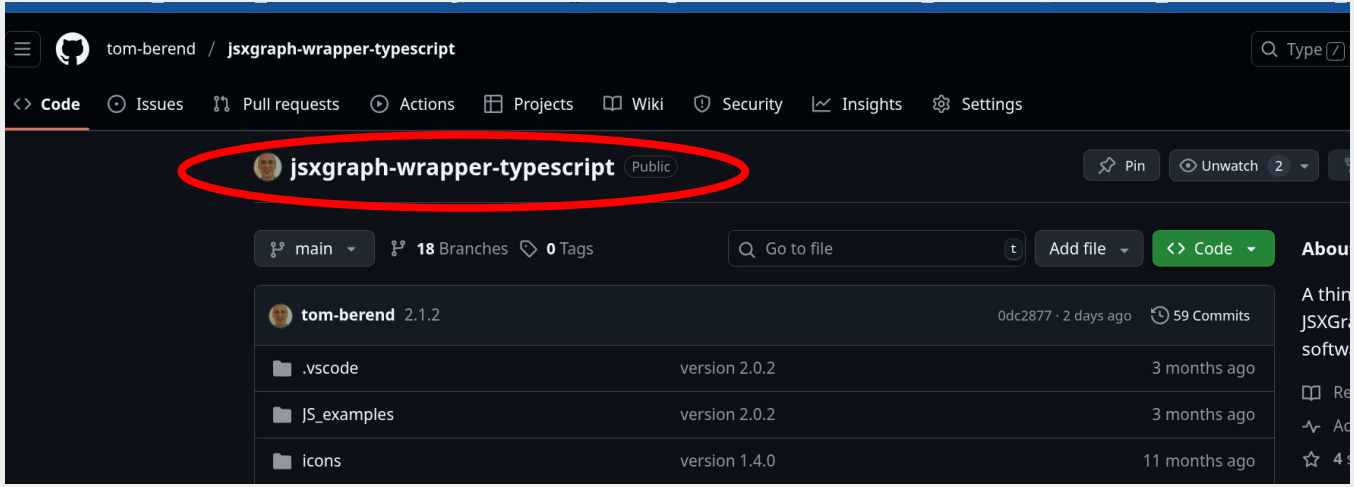
This code is identical in JavaScript and TypeScript

```
import { TSXBoard } from '../src/tsxgraph.js';
const TSX = new TSXBoard('jxgbox', { axis: true })

let p12D = TSX.Point([-3, 1], { name: '2D' })
let p13D = TSX.Point3D([-3, 1.5, 0], { name: '3D' })
```

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

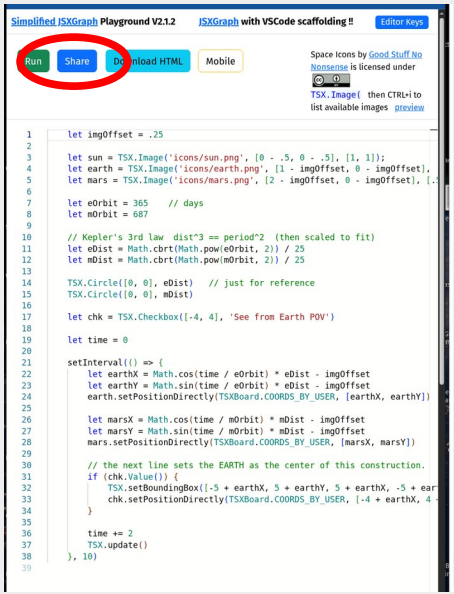
Then browse to **http://localhost:3000**

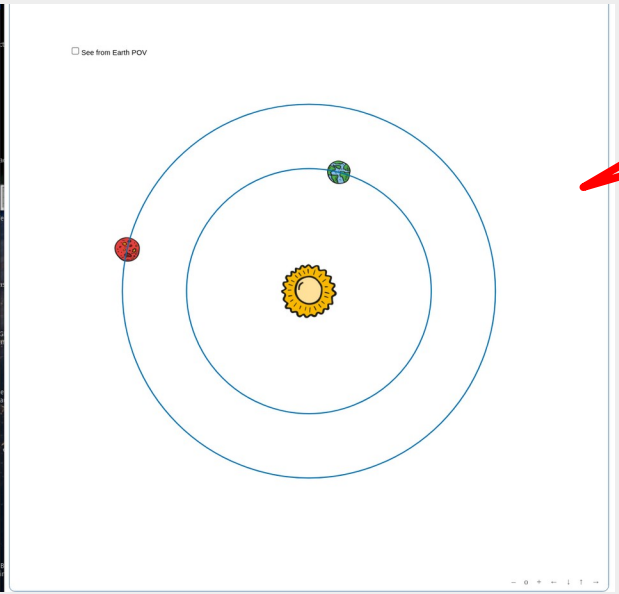
Or Just Run in the Playground

...ready to go in 0 seconds

Explore, Create, Share, Get Feedback

...lets students show off their awesomeness !





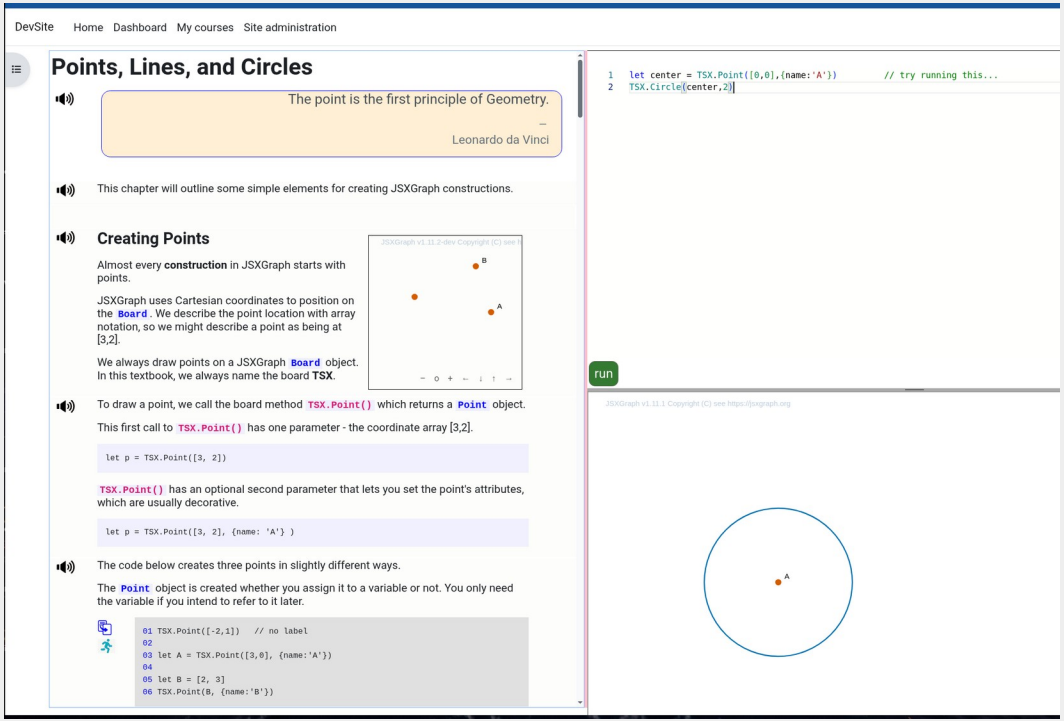
Playful Space Icons

Try it here..

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

Simplified JSXGraph Interactive Textbook

Assumes basic JavaScript

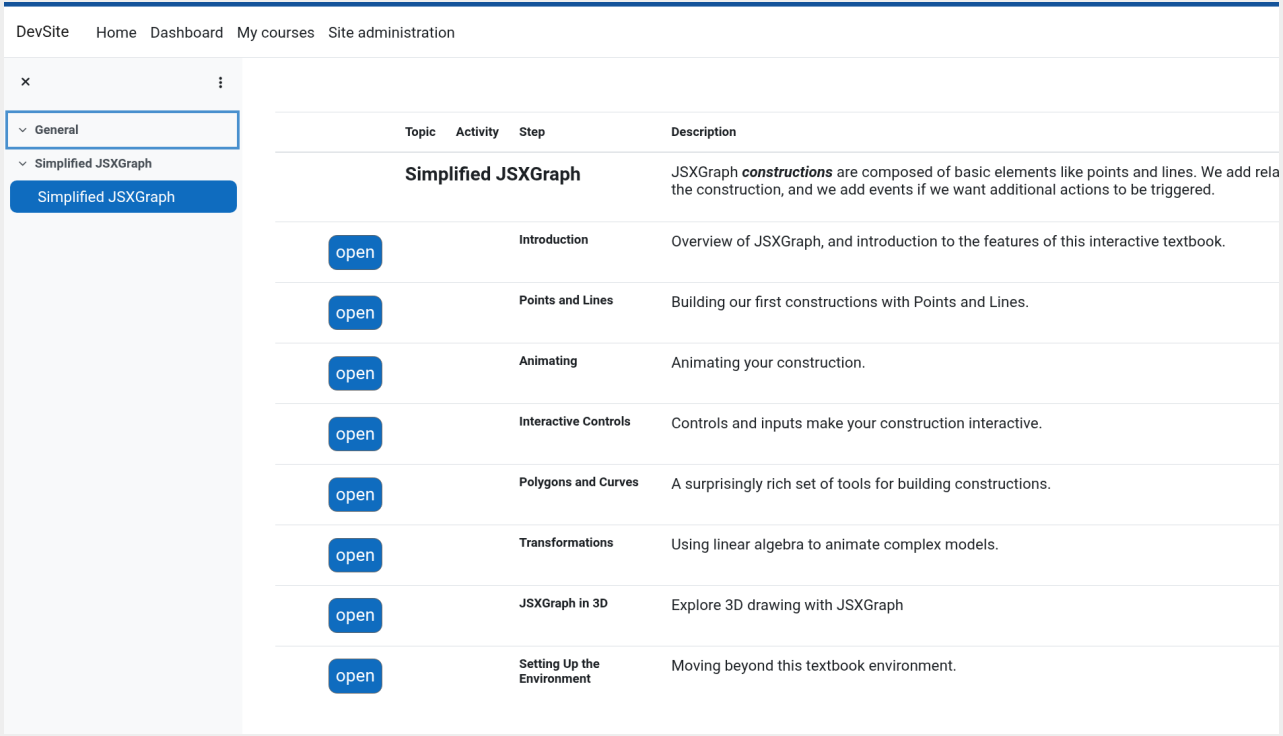


Try it here..

https://cheeseandcrackers.ca

Simplified JSXGraph Interactive Textbook

Assumes basic JavaScript



Try it here..

https://cheeseandcrackers.ca

Try Simplified JSXGraph

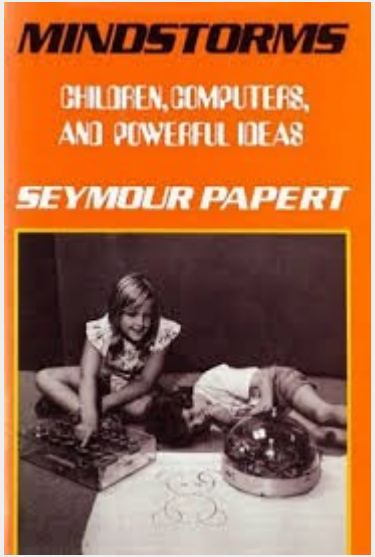
for your next project

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

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”

Simplified JSXGraph

JSXGraph with VSCode scaffolding

Tom Berend