
Elegans Documentation

Release 0.1.0

Naoki Nishida

April 29, 2014

Description

Elegans is a 3D plotting library written in JavaScript. With Elegans, you can generate charts in JavaScript, and show them on your browser.

Contents:

1.1 Getting Started

1.1.1 Installation

Download the latest version of Elegans from [here](#). Then copy and paste code below to your html file.

```
<script type="text/javascript" src="http://cdnjs.cloudflare.com/ajax/libs/three.js/r66/three.min.js">
<script type="text/javascript" src="http://cdnjs.cloudflare.com/ajax/libs/d3/3.4.4/d3.min.js"></script>
<script type="text/javascript" src="your_link_to_elegans.min.js"></script>
```

1.1.2 Preparing Data

There are two input formats of vertexes, Matrix and Array.

Matrix

Matrix format is looks like return value of `Numpy.meshgrid`. Surface and Wireframe support this format.

```
var data =
{x: [
  [1,1,1],
  [2,2,2],
  [3,3,3]
],
y: [
  [1,2,3],
  [1,2,3],
  [1,2,3]
],
z: [
  [1,2,3],
  [4,5,6],
```

```
[7, 8, 9]
];
```

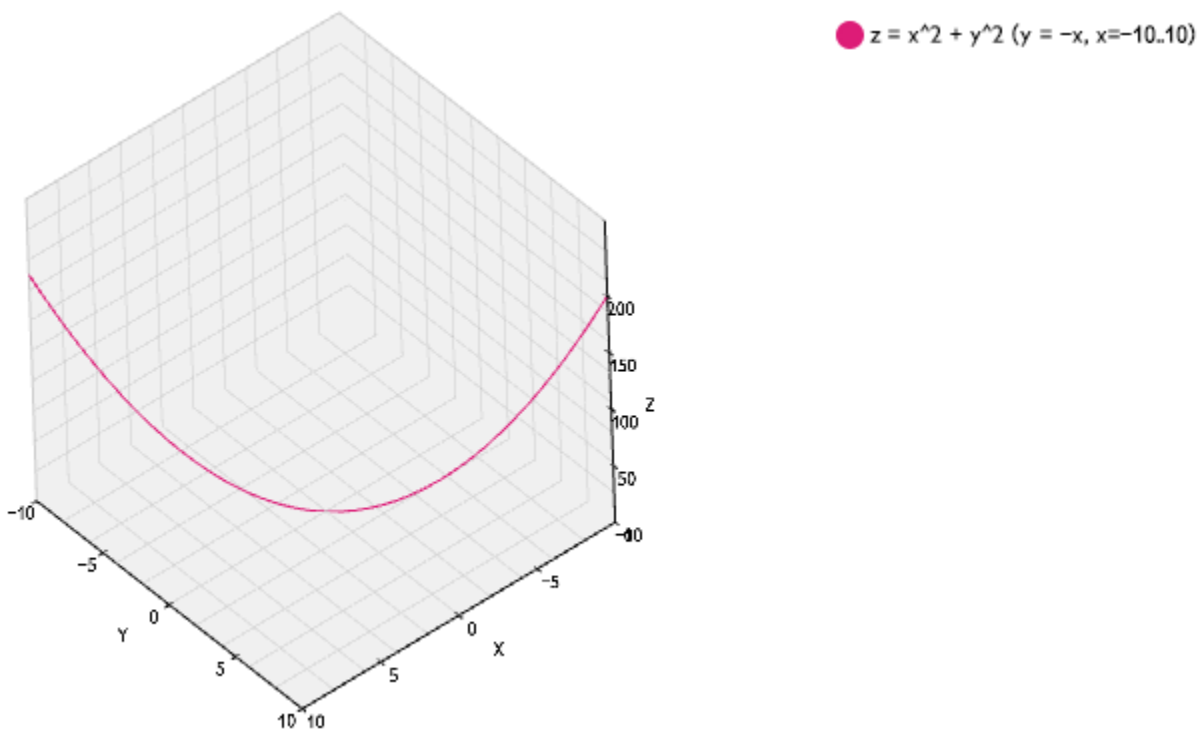
Array

In Array format, the set of 3 values that have the same index in x,y,z array indicates 1 vertex. Line, Particles and Scatter support this format.

```
var data = {x: [1, 2, 3], y: [1, 2, 3], z: [1, 2, 3]};
```

1.1.3 Quick plotting

You can quickly generate plots in method-chain style, with d3.js selector.

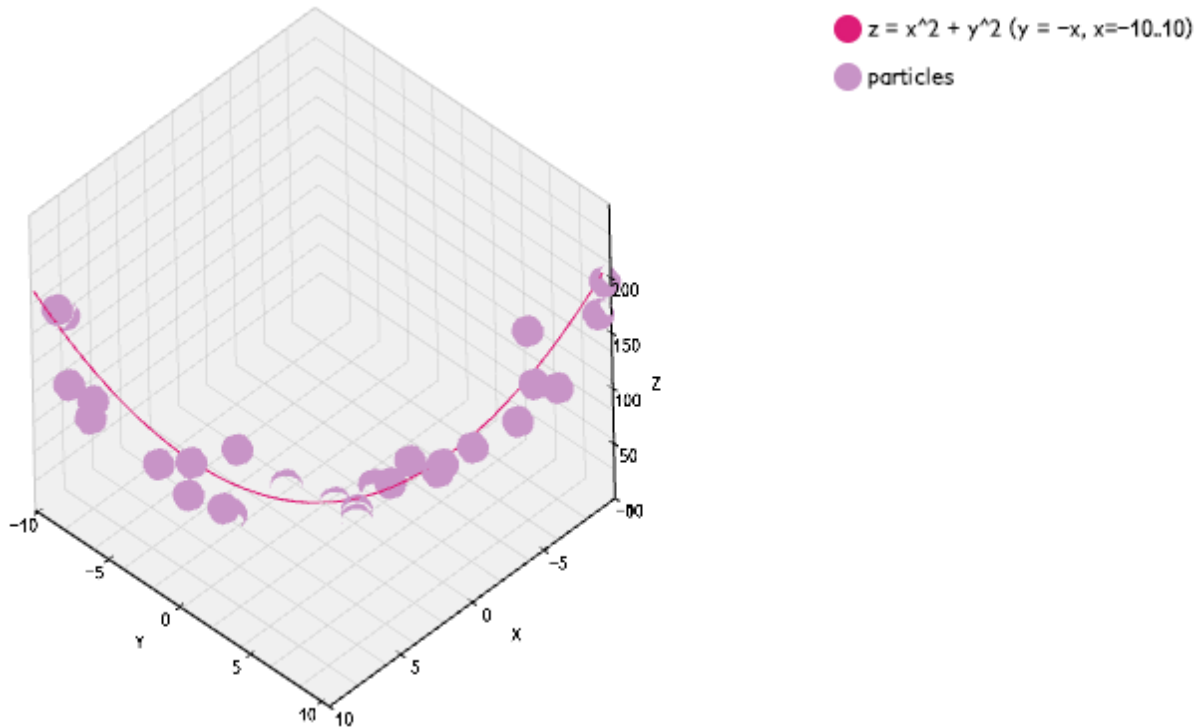


```
d3.select("#vis").datum(data).call(Elegans.LinePlot.thickness(5).colors(["#dd1c77", "#dd1c77"]));
```

[Sample code here](#)

1.1.4 Multiple plotting

When you need to put multiple plots in one 3-dimentional space, you can write code like below.



```

var stage = new Elegans.Stage(d3.select("#vis")[0][0]);
var line = new Elegans.Line(line_data, {thickness: 5, colors: ["#dd1c77", "#dd1c77"]});
var particles = new Elegans.Particles(particles_data, {size: 0.8, color: "#c994c7"});
stage.add(line);
stage.add(particles);
stage.render();

```

[Sample code here](#)

1.1.5 Embedding Elegans into your library

Elegans has API to make it easier to embed it into various environments except browsers, like IPython notebook. What you need to do is only to generate simple JSON object, and embed it into static html templates like below.

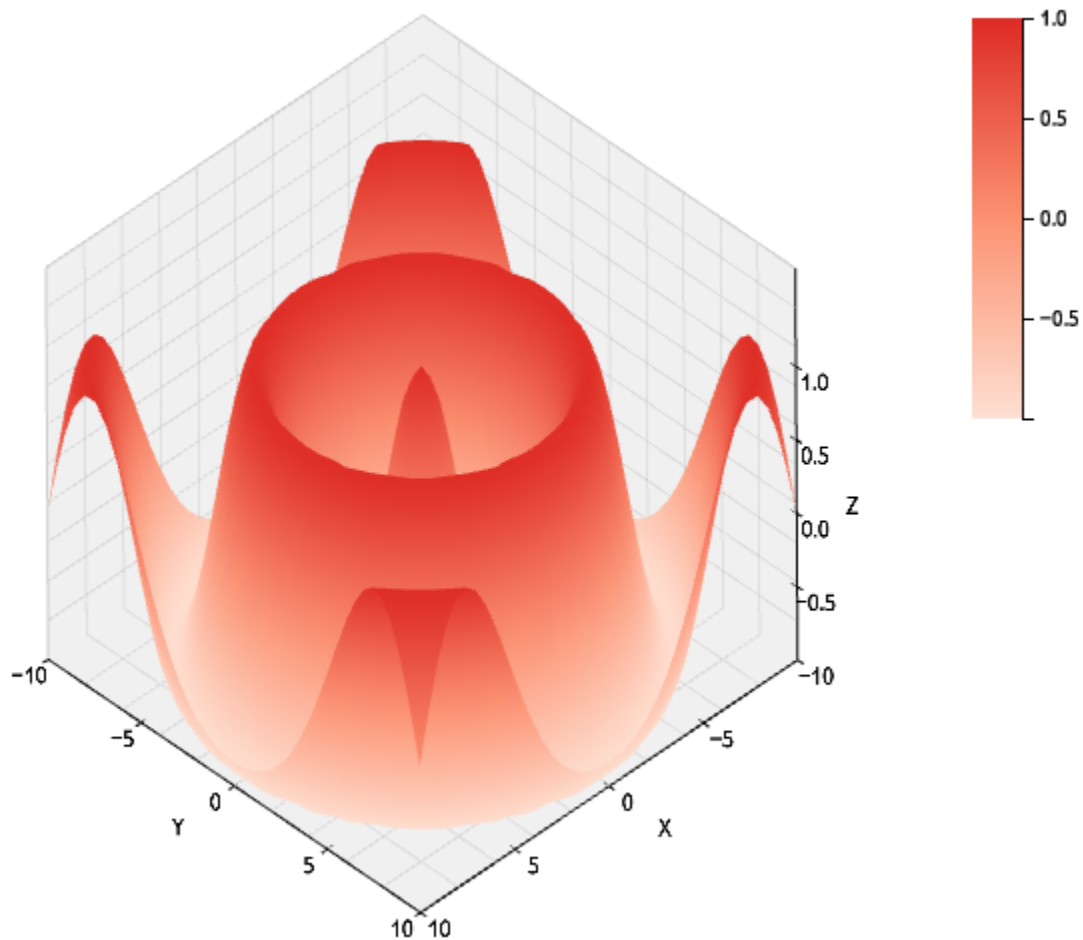
```

var model = {
  charts: [{type: "Particles", data: {x: [1, 2, 3], y: [1, 2, 3], z: [1, 2, 3]}, options: {color: "#000000"}}],
  options: {width: 500, height: 500}
};
Elegans.Embed.parse(model, "#vis");

```

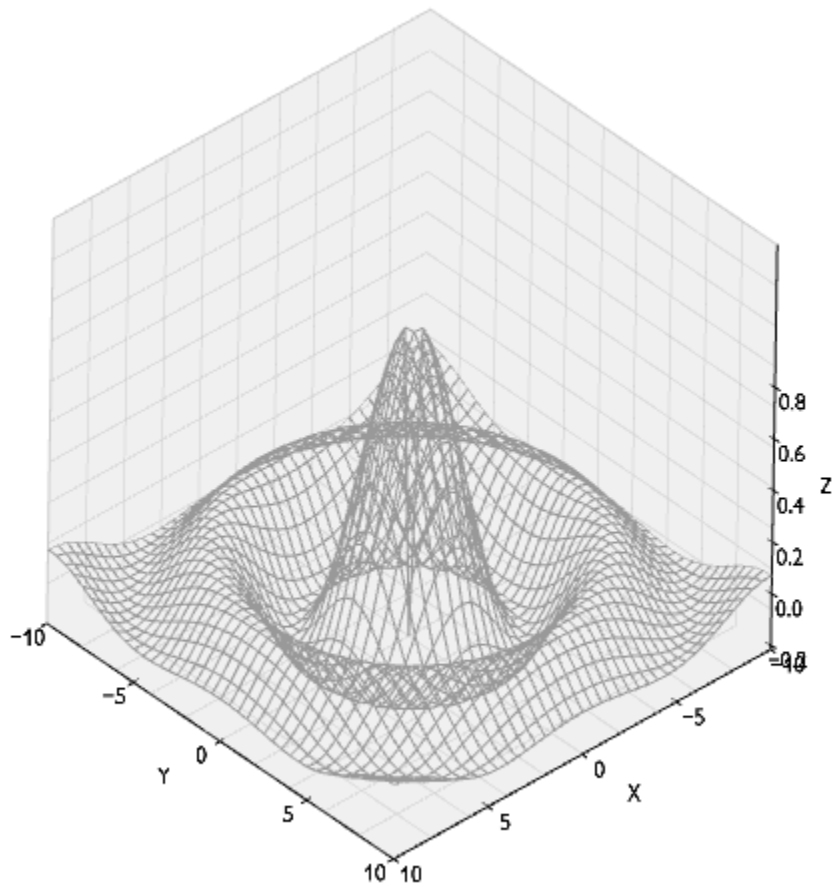
1.2 Supporting Charts

1.2.1 Surface



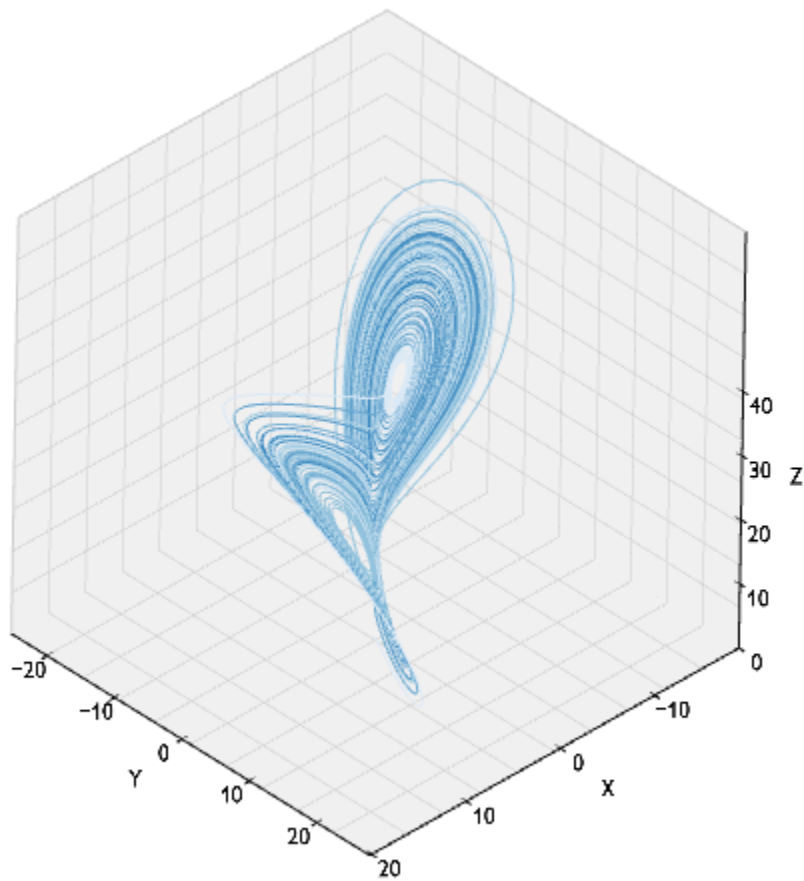
See also

1.2.2 Wireframe



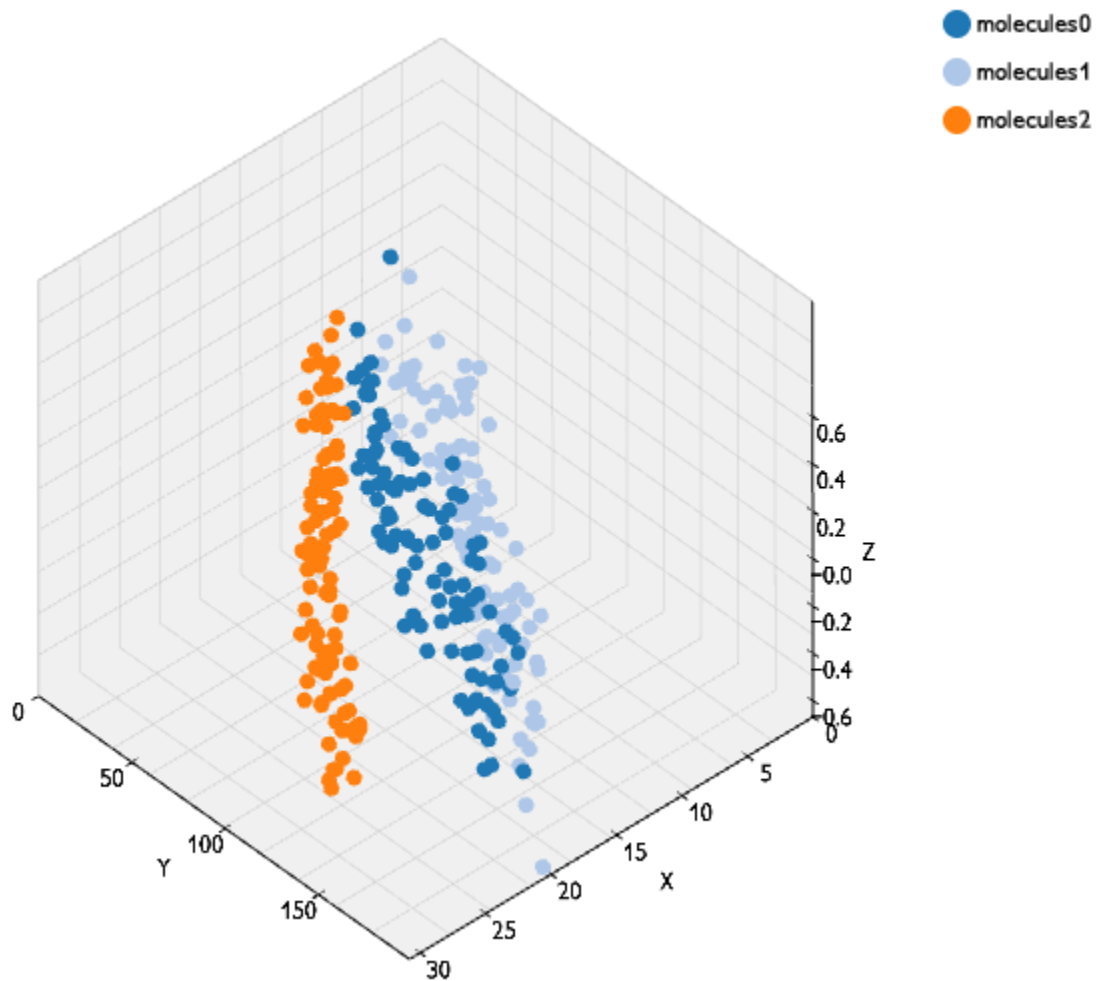
See also

1.2.3 Line



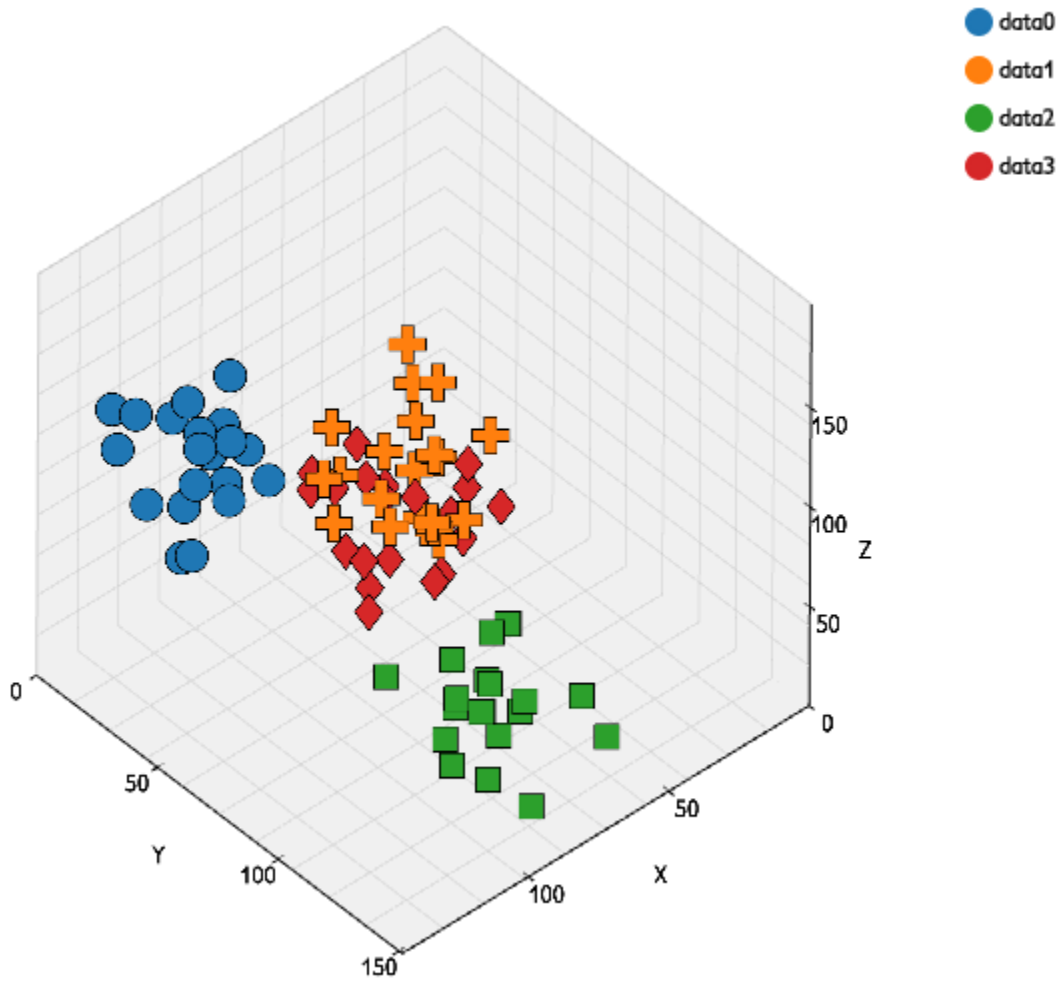
See also

1.2.4 Particles



See also

1.2.5 Scatter



See also

1.3 API Reference

1.3.1 Stage

Constructor

`Stage(element, options)`

- `element` - HTML DOM Element Object
- `options`

Methods

`.add(chart)`

Add charts to 3-dementional space.

- `chart` - instance of Surface, Wireframe, Line, Particles, Scatter

`.render()`

Start rendering charts.

1.3.2 Surface

Constructor

`Surface(data, options)`

- `data` - Matrix type data
- `options`
 - `fill_colors` - Array of String
 - `has_legend` - Boolean

1.3.3 WireFrame

Constructor

`WireFrame(data, options)`

- `data` - Matrix type data
- `options`
 - `name` - String
 - `color` - String
 - `thickness` - Number
 - `has_legend` - Boolean

1.3.4 Line

Constructor

`Line(data, options)`

- `data` - Array type data
- `options`
 - `name` - String
 - `colors` - Array of String

- thickness - Number
- has_legend - Boolean

1.3.5 Particles

Constructor

`Particles(data, options)`

- data - Array type data
- options
 - name - String
 - color - String
 - size - Number. default value is 0.3
 - has_legend - Boolean

1.3.6 Scatter

Constructor

`Scatter(data, options)`

- data - Array type data
- options
 - name - String
 - shape - String. One of “circle”, “cross”, “rect”, and “diamond.”
 - size - Number. default value is 1.5.
 - stroke_width - Number. default value is 3.
 - stroke_color - String
 - fill_color - String
 - has_legend - Boolean

1.3.7 SurfacePlot

Properties

- fill_colors - Array of String
- has_legend - Boolean

1.3.8 WireFramePlot

Properties

- data_name - String
- color - String
- thickness - Number
- has_legend - Boolean

1.3.9 LinePlot

Properties

- data_name - String
- colors - Array of String
- thickness - Number
- has_legend - Boolean

1.3.10 ParticlesPlot

Properties

- data_name - String
- color - String
- size - Number
- has_legend - Boolean

1.3.11 ScatterPlot

Properties

- data_name - String
- shape - String
- stroke_width - Number
- stroke_color - String
- fill_color - String
- has_legend - Boolean

1.3.12 Embed

Methods

```
.parse(model)
```

Parse JSON model. Return value is a instance of Elegans.Stage.

1.4 Contribution

1.4.1 How to contribute

- Fork it.
- Create your feature branch.
- Commit your changes.
- Push to the branch.
- Create new Pull-request.

1.4.2 How to build Elegans

Before building Elegans, you need to install node.js and npm.

```
sudo apt-get install node
```

After installing, Pull repository from GitHub.

```
git pull https://github.com/domitry/elegans.git
```

Then, run commands below.

```
cd elegans
npm install
grunt release
```

1.5 License

This version of Elegans is licenced under the MIT License.

The MIT License (MIT)

Copyright (c) 2014 Naoki Nishida

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- *search*