Jtopo文档完善以及问题记录

> 该文档只是针对网上的资料，结合个人使用进行知识整理，并非原创。
>
> 关于当前文档整理有问题可交流
>
> QQ：624659494
>
> 请注明来意，谢谢。

目录

- saveAsLocalImage()（导出并下载PNG图片）
- ~~toJson()（序列化）建议弃用~~

- 监听事件
  - addEventListener(eventName, eventHandler)（监听事件）
  - removeEventListener(eventName)（移除监听事件）
  - removeAllEventListener()（移除所有监听事件）
  - click(eventHandler)（监听鼠标单击事件）
  - dbclick(eventHandler)（监听鼠标双击事件）
  - mousedown(eventHandler)（监听鼠标按下事件）
  - mouseup(eventHandler)（监听鼠标松开事件）
  - mouseover(eventHandler)（监听鼠标进入Canvas事件）
  - mouseout(eventHandler)（监听鼠标离开Canvas事件）
  - mousemove(eventHandler)（监听鼠标移动事件）
  - mousedrag(eventHandler)（监听鼠标拖拽事件）
  - mousewheel(eventHandler)（监听鼠标滚轮事件）

Scene（场景对象）

- 基本属性
  - alpha（场景的透明度）
  - backgroundColor（背景颜色）
  - background（背景图片）
  - visible（场景是否可见）
  - areaSelect（是否显示选择矩形框）
  - mode（舞台模式）
  - selectedElements（被选中的元素对象）
  - translateX（水平偏移量）
  - translateY（垂直偏移量）
  - scaleX（水平缩放）
  - scaleY（垂直缩放）
- 基本函数
  - show()（显示）
  - hide()（隐藏）
  - add(element)（添加对象）
  - remove(element)（移除某个元素）
  - clear()（移除所有元素）
  - getDisplayedElements()（获取场景中可见元素）
  - getDisplayedNodes()（获取场景中可见Node对象）
  - findElements(cond)（查找场景中的对象）
- 监听事件

Node（节点对象）

- 基本属性
  - text（节点文本）
  - x（x坐标值）
  - y（y坐标值）
  - visible（设置节点是否可见）
  - shadow（是否显示阴影）
  - shadowOffsetX（阴影水平偏移量）
  - shadowOffsetY（阴影垂直偏移量）

- borderRadius（边框圆角）
- borderWidth（边框宽度）
- borderColor（边框颜色）
- zIndex（层级）
- dragable（设置节点是否可以拖动）
- selected（是否被选中）
- editAble（是否可被编辑）
- font（文本字体）
- fontColor（字体颜色）
- textPosition（文本位置）
- textOffsetX（文字水平偏移）
- textOffsetY（文字垂直偏移）
- showSelected（是否显示选中矩形）
- paintSelected（自定义绘制选中样式）
- rotate（旋转的角度）
- alpha（透明度）
- scaleX（水平缩放）
- scaleY（垂直缩放）
- selected（选中状态）
- fillColor（填充颜色）
- href（超链接）
- target（是否在新窗口打开链接）
- visitedColor（访问过的链接颜色）
- radius（半径）
- paint（绘制）
- alarm（告警文本）
- alarmAlpha（告警透明度）
- alarmColor（告警颜色）
- outLinks（作为起点的线集合）
- inLinks（作为终点的线集合）

基本函数

- setImage(url)（设置节点图片）
- setSize(width, height)（设置节点的宽和高）
- getSize()（获取节点的宽和高）
- setBound(x, y, width, height)（设置节点的坐标、宽、高）
- getBound()（获取节点的坐标、宽、高）
- setLocation(x, y)（设置节点坐标）
- setCenterLocation()（设置节点坐标到场景中心位置）

Link（连线对象）

基本属性

- text（连线文本）
- nodeA（起始节点对象）
- nodeZ（终止节点对象）
- alpha（透明度）
- strokeColor（颜色）
- lineWidth（宽度）
- bundleGap（线段之间的间隔）

- bundleOffset（线段拐角处长度）
- offsetGap（控制节点与节点间的最小间距）
- rotate（弧度）
- selected（选中状态）
- direction（线段的文本位置）
- arrowsRadius（箭头半径）

基本函数

- getStartPosition()（获得起始位置的坐标）
- getEndPosition()（获得结束位置的坐标）

Container（容器对象）

基本属性

- text（容器文本）
- x（x坐标值）
- y（y坐标值）
- width（容器宽度）
- height（容器高度）
- alpha（透明度）
- dragble（是否可以拖动）

Effect.Animate（动画效果）

基本函数

- gravity()（重力效果）
- stepByStep(通用动画)

JTopo.util（自带的工具方法）

内置方法

- randomColor()（随机颜色）
- clone(node)（克隆对象）
- cloneEvent(node)（克隆对象事件）

相关工具

这部分放前面，是为了方便查找，一些工具结合jtopo更效率的开发。

Color-hex调色板： **https://www.color-hex.com/**

> Color-hex提供有关**颜色的**信息，包括颜色模型（RGB，HSL，HSV和CMYK），三重颜色，单色和在彩色页面中计算的类似颜色。Color-hex.com还为所选颜色生成简单的css代码。Html元素样本也显示在颜色详细信息页面下方。

jTopo是什么?

- **jTopo（Javascript Topology library)**是一款完全基于HTML5 Canvas的关系、拓扑图形化界面开发工

具包。
- jTopo关注于数据的图形展示，它是面向开发人员的，需要进行二次开发。
- 使用jTopo很简单，可以快速创建一些关系图、拓扑等相关图形化的展示。只要您的数据之间存在有关联关系，都可以使用jTopo来进行人性化、图形化的展示。
- jTopo的目标：1.简单好用 2.灵活扩展 3.轻松开发出类似Visio、在线脑图、UML建模等类似工具 4.为大数据可视化提供解决方案

为什么需要jTopo

1. 随着软件**用户体验**的要求不断提高，界面图形化展示、操作往往能给用户带来印象深刻、用户体验度提升的同时也会增加对软件更加认可。
2. 一般的软件系统图形界面开发需要花费大量的时间和人力，运行效率低下，界面不够专业美观。如果采用jTopo组件来开发软件界面，可以非常简单快速地创建精美、专业、高效的图形化界面，以便提高开发效率、增加客户满意度、提高软件竞争力。
3. 基于Html5的拓扑图函数库比较少，好用的更是寥寥无几，好用又跨浏览器平台的目前就只有jTopo。
4. 跨行业：可应用到电信、电力、金融、制造、交通、教育等多个行业的软件开发中。

jTopo特点

1. 完全基于HTML5 Canvas开发，始终站在开发者的角度设计，API平易近人、几乎简单到了极致。
2. 不依赖任何其他库、执行仅需一个Canvas，不污染你的页面、Dom结构和代码命名空间。
3. 功能异常强大、灵活，可扩展性极强（为扩展而生），包装一下，就是一款很专业的图形化软件。
4. 体积小，压缩后仅几十KB。
5. 性能十分优异，可流畅地展示大量数据(经过专业优化过甚至可以展示几十万、百万级别的数据)
6. **免费**
7. 不足：目前文档不够详细，主要通过Demo来熟悉。**(本文档逐步完善中)**

对于jtopo的个人理解

> 一个jtopo实例比喻成看一次演唱会（Canvas比喻场馆，用于容纳这次演出），演出有可能是一个舞台（Stage）甚至多个舞台（Stage）组合进行，不同的时间段或者按照演出顺序的不同，会出现不同的场景（Scene），每个场景中会出现不同的人和物（节点：Node，容器：Container），人与物之间表演会存在关系（Link）。
>
> 以上就是我对Jtopo的基本组成要素的一个比较通俗的理解

创建一个Jtopo

每一个Jtopo都是由以下几个基本步骤组成：

- 引入js：

```
<script src="jtopo-0.4.8-min.js"></script>
```

- 创建一个canvas对象：

```
<canvas id="jtopoCanvas" ></canvas>
```

- 在canvas中创建一个或多个stage(舞台对象)：

```
var canvas = document.getElementById("jtopoCanvas"); // canvas节点
var stage1 = new JTopo.Stage(canvas);// 舞台对象1
//var stage2 = new JTopo.Stage(canvas);// 舞台对象2
//var stage3 = new JTopo.Stage(canvas);// 舞台对象3
```

- 在stage中创建一个或多个scene(场景对象)：

```
var scene1 = new JTopo.Scene(stage1);// 舞台对象1中的场景对象1
//var scene2 = new JTopo.Scene(stage1);// 舞台对象1中的场景对象2
//var scene3 = new JTopo.Scene(stage2);// 舞台对象2中的场景对象3
```

- 在scene中创建一个或多个node(节点对象)：

```
var node1 = new JTopo.Node("节点1名称");
//对点进行个性化设置，具体详情查看node的章节
node1.setLocation(x, y); // 节点的坐标位置 x，y为变量
node1.setSize(width, height); // 节点大小 width，height为变量
//生成点
scene1.add(node1);

var node2 = new JTopo.Node("节点2名称");
//对点进行个性化设置，具体详情查看node的章节
node2.setLocation(x, y); // 节点的坐标位置 x，y为变量
node2.setSize(width, height); // 节点大小 width，height为变量
//生成点
scene1.add(node2);
```

- 在scene中创建节点1与节点2之间连线：

```
var link1 = new JTopo.Link(node1, node2, "线1的名称"); //这是有方向的线 node1为起始，node2为终止
//还可以对线进行个性化设置，具体详情查看link的章节
link1.lineWidth = 3; // 线宽
link1.dashedPattern = dashedPattern; // 虚线
link1.bundleOffset = 60; // 折线拐角处的长度
link1.bundleGap = 20; // 线条之间的间隔
link1.textOffsetY = 3; // 文本偏移量（向下3个像素）
link1.strokeColor = '0,200,255';
//生成线
scene.add(link1);
```

代码到此，已经实现了一个最基本的jtopo实例，如需要继续深入，请继续阅读后面章节。

Stage（舞台对象）

一个抽象的舞台对象,对应一个Canvas和多个场景对象(Scene)

```
var canvas = document.getElementById("jtopoCanvas"); // canvas节点
var stage = new JTopo.Stage(canvas);
```

基本属性

frames（设置当前舞台播放的帧数/秒）

> 设置当前舞台播放的帧数/秒
>
> 默认为:24
>
> frames可以为0，表示：不自动绘制，由用户手工调用Stage对象的paint()方法来触发。
>
> 如果小于0意味着：只有键盘、鼠标有动作时才会重绘，例如：stage.frames = -24。

例子:

```
stage.frames = -24
```

canvas（所属Canvas对象）

> 对应的Canvas对象

width（舞台宽度）

> 舞台宽度(Canvas的宽度)

例子:

```
stage.width = 500
```

height（舞台高度）

> 舞台高度(Canvas的高度)

例子：

```
stage.height = 500
```

mode（舞台的表现模式）

> 舞台模式，不同模式下有不同的表现:
>
> 设置舞台模式，例如：stage.mode = "drag";
>
> normal[默认]：可以点击选中单个节点（按住Ctrl可以选中多个），点中空白处可以拖拽整个画面
>
> drag: 该模式下不可以选择节点，只能拖拽整个画面
>
> select: 可以框选多个节点、可以点击单个节点
>
> edit: 在默认基础上增加了：选中节点时可以通过6个控制点来调整节点的宽、高

例子：

```
stage.mode = "drag"
```

childs（场景对象列表）

> 场景对象列表，其中包含了该舞台（stage）下所有的场景（scene）

eagleEye（鹰眼对象）

> 鹰眼对象（右下角的全景小窗口）
>
> 显示鹰眼：stage.eagleEye.visible = true
>
> 隐藏鹰眼：stage.eagleEye.visible = false

例子：

```
stage.eagleEye.visible = true
```

wheelZoom（鼠标滚轮缩放操作比例）

> 鼠标滚轮缩放操作比例，默认为null，不显示鹰眼
>
> 启用鼠标滚轮缩放：stage.wheelZoom = 0.85; //缩放比例为0.85
>
> 禁用用鼠标滚轮缩放: stage.wheelZoom = null;

例子：

```
stage.wheelZoom = 0.85

// 设置场景缩放 Firefox缩放bug
stage.wheelZoom = Browser == "FF" ? 0.8 : 1.2;
```

cursor（拖动时鼠标图形设置）

> 拖动时鼠标图形可以自定义成其他图片。

例子：

```
stage.cursor = "url(./img/cur/closedhand.cur) 8 8, default"; // 拖动鼠标图形设置
```

基本函数

add(Scene)（添加舞台）

> 将一个Scene场景加入到舞台中（只有加入舞台才可以显示出现）

例子：

```
var scene = new JTopo.Scene();
stage.add(scene);
```

remove(Scene)（移除舞台）

> 将一个Scene场景从舞台中移除（不再显示）

例子：

```
stage.remove(scene);
```

clear()（移除舞台）

> 将所有Scene场景从舞台中移除

paint()（执行一次绘制）

> 执行一次绘制, 如果frames设置为0，可以手工调用此方法来通知jtopo进行一次重绘。

zoom(scale)（缩放）

> 缩放，scale取值范围[0-1]，实际上本操作是调用了舞台中所有Scene对象的zoom函数。

zoomOut(scale)（放大）

> 放大，scale取值范围[0-1], 调用zoom实现。

zoomIn(scale)（缩小）

> 缩小，scale取值范围[0-1], 调用zoom实现。

centerAndZoom(scale)（缩放并居中显示所有元素）

> 缩放并居中显示所有元素，scale为缩放比例，取值范围[0-1]

setCenter(x, y)（设置当前舞台的中心坐标）

> 设置当前舞台的中心坐标（舞台平移）

getBound()（得到边界大小）

> 得到舞台中所有元素位置确定的边界大小（left、top、right、bottom）

saveImageInfo()（导出成PNG图片）

> 导出成PNG图片（在新打开的浏览器Tab页中）

saveAsLocalImage()（导出并下载PNG图片）

> 导出成PNG图片（直接弹出另存为对话框或者用下载软件下载）

~~toJson()（序列化）建议弃用~~

> 把当前对象的属性序列化成json数据

监听事件

> **jtopo中监听事件的理解：在jtopo中，事件的监听是针对整个舞台的，并不是针对独立对象，如：**
>
> **例题：**实现节点的右键菜单。

> **思路：**
>
> 1、监听舞台的鼠标右键点击事件；
>
> 2、事件触发后，检查当前被选中的对象属性，是否为节点对象；
>
> 3、获取当前鼠标位置或者节点坐标，弹出右键菜单。

addEventListener(eventName, eventHandler)（监听事件）

> 可以监听的事件有：
> click,dbclick,mousedown,mouseup,mouseover,mouseout,mousemove,mousedrag,mousewheel

例子：

```
stage.addEventListener("mousedown", function(event){});
```

removeEventListener(eventName)（移除监听事件）

> 移除监听事件和addEventListener相对应

例子：

```
stage.removeEventListener("mousedown", function(event){});
```

removeAllEventListener()（移除所有监听事件）

> 移除所有监听事件

例子：

```
stage.removeAllEventListener()
```

click(eventHandler)（监听鼠标单击事件）

> 监听鼠标单击事件（鼠标按下并松开），等价于：stage.addEventListener("click", eventHandler);

例子：

```
stage.click(function(event){})
//或
stage.addEventListener("click", function(event){});
```

dbclick(eventHandler)（监听鼠标双击事件）

> 监听鼠标双击事件（鼠标按下并松开），等价于：stage.addEventListener("dbclick", eventHandler);

例子：

```
stage.dbclick(function(event){})
//或
stage.addEventListener("dbclick", function(event){});
```

mousedown(eventHandler)（监听鼠标按下事件）

> 监听鼠标按下事件，等价于：stage.addEventListener("mousedown", eventHandler);

例子：

```
stage.mousedown(function(event){})
//或
stage.addEventListener("mousedown", function(event){});
```

mouseup(eventHandler)（监听鼠标松开事件）

> 监听鼠标松开事件，等价于：stage.addEventListener("mouseup", eventHandler);

例子：

```
stage.mouseup(function(event){})
//或
stage.addEventListener("mouseup", function(event){});
```

mouseover(eventHandler)（监听鼠标进入Canvas事件）

> 监听鼠标进入Canvas事件，等价于：stage.addEventListener("mouseover", eventHandler);

例子：

```
stage.mouseover(function(event){})
//或
stage.addEventListener("mouseover", function(event){});
```

mouseout(eventHandler)（监听鼠标离开Canvas事件）

> 监听鼠标离开Canvas事件，等价于：stage.addEventListener("mouseup", eventHandler);

例子：

```
stage.mouseout(function(event){})
//或
stage.addEventListener("mouseout", function(event){});
```

mousemove(eventHandler)（监听鼠标移动事件）

> 监听鼠标移动事件，等价于：stage.addEventListener("mousemove", eventHandler);

例子：

```
stage.mousemove(function(event){})
//或
stage.addEventListener("mousemove", function(event){});
```

mousedrag(eventHandler)（监听鼠标拖拽事件）

> 监听鼠标拖拽事件，等价于：stage.addEventListener("mousedrag", eventHandler);

例子：

```
stage.mousedrag(function(event){})
//或
stage.addEventListener("mousedrag", function(event){});
```

mousewheel(eventHandler)（监听鼠标滚轮事件）

> 监听鼠标滚轮事件，等价于：stage.addEventListener("mousewheel", eventHandler);

例子：

```
stage.mousewheel(function(event){})
//或
stage.addEventListener("mousewheel", function(event){});
```

Scene（场景对象）

创建一个场景对象，场景对象，概念上同很多图形系统中的Layer。

```
var scene = new JTopo.Scene(stage);
```

基本属性

alpha（场景的透明度）

> 场景的透明度0~1，默认为0，即：完全透明。所以有时候即使设置了背景颜色却不起作用）

例子：

```
scene.alpha = 0.5
```

backgroundColor（背景颜色）

> 背景颜色，设置的时候请注意alpha属性。

例子：

```
// 只支持输入rgb值，设置背景颜色是必须设置透明度
scene.alpha = 1
scene.backgroundColor = '80,100,80'
```

background（背景图片）

> 设置场景的背景图片,与backgroundColor冲突，一旦设置了该属性，backgroundColor属性将失效

例子：

```
//与backgroundColor冲突，一旦设置了图片，backgroundColor属性将失效
scene.background = "./img/bg.png";
```

visible（场景是否可见）

> 得到、设置场景是否可见，默认为：true

例子：

```
//设置场景是否可见
scene.visible = true
//查看场景是否可见
var visible = scene.visible
```

areaSelect（是否显示选择矩形框）

> 在select模式中，是否显示选择矩形框

例子：

```
scene.areaSelect = true
```

mode（舞台模式）

> 舞台模式，不同模式下有不同的表现: 设置舞台模式，例如：stage.mode = "drag";
>
> normal[默认]：可以点击选中单个节点（按住Ctrl可以选中多个），点中空白处可以拖拽整个画面
>
> drag: 该模式下不可以选择节点，只能拖拽整个画面
>
> select: 可以框选多个节点、可以点击单个节点
>
> edit: 在默认基础上增加了：选中节点时可以通过6个控制点来调整节点的宽、高

例子：

```
stage.mode = "drag"
```

selectedElements（被选中的元素对象）

> 当前场景中被选中的元素对象

例子：

```
var elements = scene.selectedElements
```

translateX（水平偏移量）

> 场景偏移量（水平方向），随鼠标拖拽变化

例子：

```
var translateX = scene.translateX
//设置偏移量
scene.translateX = 100
```

translateY（垂直偏移量）

> 场景偏移量（垂直方向），随鼠标拖拽变化

例子：

```
var translateY = scene.translateY
//设置偏移量
scene.translateY = 100
```

scaleX（水平缩放）

> 场景水平缩放比例，默认为1（水平）

scaleY（垂直缩放）

> 场景垂直缩放比例，默认为1（垂直）

基本函数

show()（显示）

> 显示

例子：

```
scene.show()
```

hide()（隐藏）

> 隐藏

例子：

```
scene.hide()
```

add(element)（添加对象）

> 添加对象到当前场景中来。

例子：

```
//场景中添加节点
scene.add(new JTopo.Node());
//场景中添加线
scene.add(new JTopo.Link(nodeA, nodeZ))
```

remove(element)（移除某个元素）

> 移除场景中的某个元素。

例子：

```
scene.remove(myNode);
```

clear()（移除所有元素）

> 移除场景中的所有元素

例子：

```
scene.clear()
```

getDisplayedElements()（获取场景中可见元素）

> 获取场景中可见并绘制出来的元素（超过Canvas边界）

例子：

```
var elements = scene.getDisplayedElements()
```

getDisplayedNodes()（获取场景中可见Node对象）

> 获取场景中可见并绘制出来的Node对象（超过Canvas边界）

例子：

```
var nodes = scene.getDisplayedNodes()
```

findElements(cond)（查找场景中的对象）

> 查找场景中的对象。

> **注意：** 入参必须为方法，必须有返回值，返回值为true表示匹配成功，记录到查找结果列表中

例子：

```
//查找所有x坐标大于100的元素
var elements = scene.findElements(function(element){ return element.x > 100; });
```

监听事件

> 监听事件统一在stage中实现。scene不提供单独的监听。

Node（节点对象）

拓扑图中的点对象

节点包含以下几种类型：

```
// 创建节点
var node = new JTopo.Node('节点名称'); // 创建普通节点
var circleNode = new JTopo.CircleNode('节点名称'); // 创建圆形节点
var textNode = new JTopo.TextNode('This is a text node.'); // 创建文本节点
var linkNode = new JTopo.LinkNode('超链接节点');  // 创建链接节点
```

基本属性

text（节点文本）

> 设置节点的名字（显示文本）

x（x坐标值）

> x坐标值

y（y坐标值）

> y坐标值

visible（设置节点是否可见）

> 设置节点是否可见

shadow（是否显示阴影）

> 是否显示阴影, 例如:node.shadow = "true"

shadowOffsetX（阴影水平偏移量）

> 阴影水平偏移量（像素）

shadowOffsetY（阴影垂直偏移量）

> 阴影垂直偏移量（像素）

borderRadius（边框圆角）

> 边框的圆角弧度，如：node.borderRadius = 5;

borderWidth（边框宽度）

> 边框宽度/厚度，如：node.borderWidth = 2;

borderColor（边框颜色）

> 边框颜色，如：node.borderColor = '255,255,255';

zIndex（层级）

> zIndex，大的覆盖小的,范围 [10-999]，10以下保留占用。

dragable（设置节点是否可以拖动）

> 设置节点是否可以拖动

selected（是否被选中）

> 是否被选中

editAble（是否可被编辑）

> 是否可被编辑

font（文本字体）

> 节点字体，例如：node.font = "12px Consolas"

例子：

```
node.font = '14px 微软雅黑'
```

fontColor（字体颜色）

> 字体颜色，例如：node.fontColor = "255,255,0"

例子：

```
node.fontColor = "255,255,0"
```

textPosition（文本位置）

> 节点文本位置，例如：node.textPosition = "Bottom_Center"
>
> 对齐方式可选择：
>
> Top_Left、Top_Center、Top_Right（顶部）
>
> Middle_Left、Middle_Center、Middle_Right（中部）
>
> Bottom_Left、Bottom_Center、Bottom_Right（底部）

textOffsetX（文字水平偏移）

> 文字水平偏移，大于0向右偏移，小于0向左偏移（单位像素）

例子：

```
node.textOffsetX = -30; // 文字向左偏移 30像素
node.textOffsetX = 30; // 文字向右偏移 30像素
```

textOffsetY（文字垂直偏移）

> 文字垂直偏移，大于0向下偏移，小于0向上偏移（单位像素）

例子：

```
node.textOffsetY = -8; // 文字向上偏移8像素
node.textOffsetY = 8; // 文字向下偏移8像素
```

showSelected（是否显示选中矩形）

> 选中时，是否显示表示选中状态的矩形，默认为：true，显示

paintSelected（自定义绘制选中样式）

> 自定义绘制选中样式

例子：

```
node.paintSelected = function(a) { //修改节点选中样式
  0 != this.showSelected &&
    (a.save(), a.beginPath(), a.strokeStyle = "rgba(218,221,221, 0)",
      a.fillStyle = "rgba(218,221,221,0)",
      a.arc(0, 0, this.width / 2 + 3, 0, Math.PI * 2, false),
        a.fill(), a.stroke(), a.closePath(), a.restore());
};
```

rotate（旋转的角度）

> 设置节点旋转的角度（弧度）

alpha（透明度）

> 透明度, 取值范围[0-1]

scaleX（水平缩放）

> 水平缩放

scaleY（垂直缩放）

> 垂直缩放

selected（选中状态）

> 用于判断是否已被选中

fillColor（填充颜色）

> 设置节点的填充颜色

href（超链接）

> 作用于链接节点：var linkNode = new JTopo.LinkNode('超链接节点');

例子：

```
linkNode.href = 'http://www.jtopo.com';
```

target（是否在新窗口打开链接）

> 作用于链接节点：var linkNode = new JTopo.LinkNode('超链接节点');
>
> 是否在新窗口打开链接，默认为非新窗口打开

例子：

```
linkNode.target = '_blank'; // 新窗口打开链接
//或
linkNode.target = null // 当前窗口重定向打开链接
```

visitedColor（访问过的链接颜色）

> 作用于链接节点：var linkNode = new JTopo.LinkNode('超链接节点');
>
> 访问过的链接颜色

例子：

```
linkNode.visitedColor = '0,0,255'; // 访问过的链接为蓝色
```

radius（半径）

> 作用于链接节点：new JTopo.CircleNode('圆形节点');
>
> 单位：像素

例子：

```
circleNode.radius = 24; // 半径
```

paint（绘制）

> 该属性默认用于绘制图形，如果比较熟悉的同学可以尝试自定义图形绘制

例子：

```
var node = new JTopo.Node("自定义绘制");
  node.percent = 0.8; // 百分比，自定义属性
  node.beginDegree = 0; // 开始度，自定义属性
  node.width = node.height = 60; // 设置大小
  node.setLocation(200, 210); //设置位置
  node.paint = function(g) { // 绘制
    g.beginPath();
    g.moveTo(0, 0);
    g.fillStyle = 'rgba(0,255,0,' + this.alpha + ')';
```

```
    g.arc(0, 0, this.width / 2, this.beginDegree, this.beginDegree + 2 * Math.PI *
this.percent);
    g.fill();
    g.closePath();
        g.save();
        g.beginPath();
        g.fillStyle = 'rgba(255,255,0,' + this.alpha + ')';
        g.moveTo(0, 0);
        g.arc(0, 0, this.width / 2 - 10, this.beginDegree, this.beginDegree + 2 *
Math.PI);
        g.fill();
        g.closePath();
        g.restore();
        this.paintText(g);
};
scene.add(node);
```

alarm（告警文本）

> 告警文本信息

例子：

```
//设置告警
node.alarm = '设备故障';

//删除告警
delete node.alarm || node.alarm = null
```

alarmAlpha（告警透明度）

> 告警透明度，如：node.alarmAlpha = 0.8

alarmColor（告警颜色）

> 告警颜色，如：node.alarmColor = "255,255,0"

outLinks（作为起点的线集合）

> 与其它节点连接并以此node为起点的线段的集合

inLinks（作为终点的线集合）

> 与其它节点连接并以此node为终点的线段的集合

基本函数

setImage(url)（设置节点图片）

> 设置节点图片

setSize(width, height)（设置节点的宽和高）

> 设置节点的宽和高

getSize()（获取节点的宽和高）

> 获取节点的宽和高

setBound(x, y, width, height)（设置节点的坐标、宽、高）

> 设置节点的坐标、宽、高

getBound()（获取节点的坐标、宽、高）

> 获取节点的坐标、宽、高

setLocation(x, y)（设置节点坐标）

> 设置节点在场景中的位置坐标(左上角)

setCenterLocation()（设置节点坐标到场景中心位置）

> 设置节点在场景中的位置坐标(中心位置)

Link（连线对象）

拓扑中的线对象

存在以下几种类型：

```
var link = new JTopo.Link(nodeA, nodeZ, text); // 普通线
var foldLink = new JTopo.FoldLink(nodeA, nodeZ, text); // 折线
var flexionalLink = new JTopo.FlexionalLink(nodeA, nodeZ, text); // 二次折线
var curveLink = new JTopo.CurveLink(nodeA, nodeZ, text); // 曲线
```

基本属性

text（连线文本）

> 连线的名字（文本）

nodeA（起始节点对象）

> 起始节点对象

nodeZ（终止节点对象）

> 终止节点对象

alpha（透明度）

> 透明度

strokeColor（颜色）

> 线条的颜色

lineWidth（宽度）

> 线条的宽度（像素）

bundleGap（线段之间的间隔）

> 多条线同时存在与两个节点的时候，线之间的间隔。（单位：像素）

bundleOffset（线段拐角处长度）

> 如果线段存在折位，那么折的位置距离点的长度。（单位：像素）

offsetGap（控制节点与节点间的最小间距）

> 控制节点与节点间的最小间距

rotate（弧度）

> 作用于弧线

selected（选中状态）

> 用于判断是否已被选中

direction（线段的文本位置）

> 线段的文本位置

arrowsRadius（箭头半径）

> 箭头半径

基本函数

getStartPosition()（获得起始位置的坐标）

> 获得起始位置的坐标

getEndPosition()（获得结束位置的坐标）

> 获得结束位置的坐标

Container（容器对象）

基本属性

text（容器文本）

> 名称（文本）, 不会显示

x（x坐标值）

> x坐标值

y（y坐标值）

> y坐标值

width（容器宽度）

> 容器宽度

height（容器高度）

> 容器高度

alpha（透明度）

> 透明度

dragble（是否可以拖动）

> 是否可以拖动

Effect.Animate（动画效果）

基本函数

gravity()（重力效果）

> 给指定元素增加重力效果

stepByStep(通用动画)

> 通用动画效果功能，可以把一个元素对象的某些属性在指定的时间内变化到指定值

JTopo.util（自带的工具方法）

内置方法

randomColor()（随机颜色）

clone(node)（克隆对象）

cloneEvent(node)（克隆对象事件）