servasoftware

TWaver® HTML5

开发手册

Version 5.0

Nov 2014
Serva Software
info@servasoftware.com
http://www.servasoftware.com
PO Box 8143, Wichita Falls, Texas, USA 76307

For more information about Serva Software and TWaver please visit the web site at:

http://www.servasoftware.com

Or send e-mail to:

info@servasoftware.com

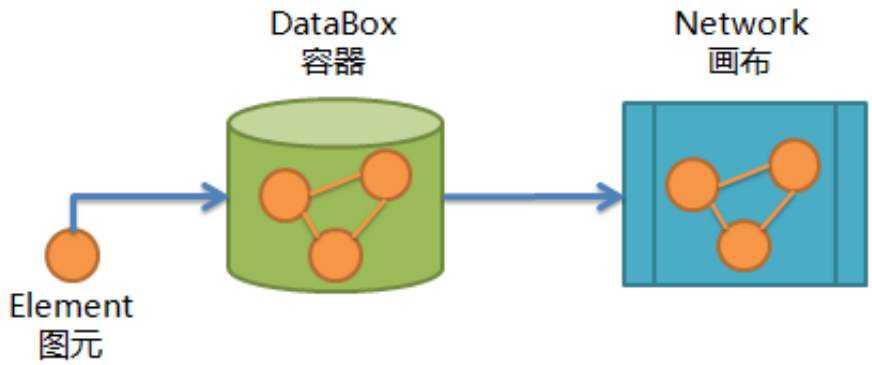
Nov, 2014

基本概念

TWaver HTML5（以下简称TWaver）使用HTML5技术和javascript语言，可在支持HTML5的浏览器上进行绘图。

使用TWaver前，需熟悉几个基本概念：图元（Element）、容器（DataBox）和画布（Network）。

- 图元：图形中的各种基本元素，如节点（Node）、连线（Link）等；
- 容器：图元都统一放置在一个容器（DataBox）中进行管理，如同“装鸡蛋的篮子”一样。它负责图元的增/删/改/查等管理操作；
- 画布：图元最终都绘制在画布（Network）组件上。Network是最终用户看到的图形组件，负责图形画面的具体绘制和交互；



Note:

以上概念是熟练使用TWaver需要掌握的最基本概念，请熟练掌握。

第一个例子

TWaver最常用的几个类：

- 图元：twaver.Node（节点）、twaver.Link（连线），都是twaver.Element图元基础类的子类；
- 容器：twaver.ElementBox，是twaver.DataBox容器类的子类；
- 画布：twaver.vector.Network；

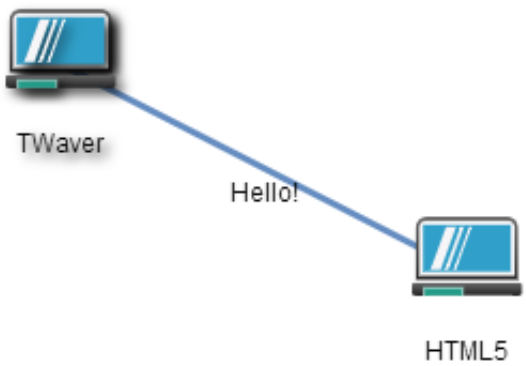
绘制图形的过程是：

1. new一个容器对象ElementBox、一个画布对象Network；
2. 把Network放置在页面中并设置大小；

3. new若干图元（Node、Link等）并add到Box容器中；

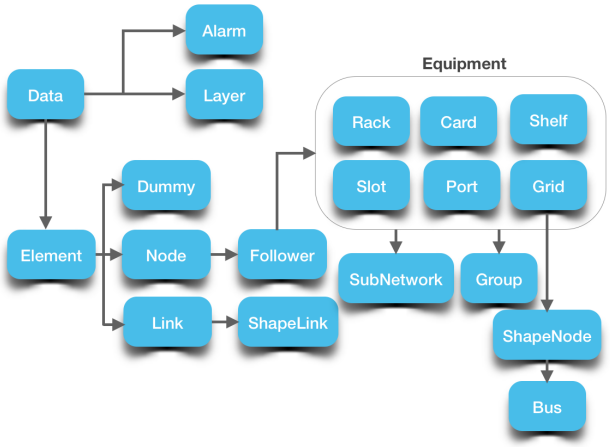
下面的HTML页面创建了一个“两点一线”的简单图形：

```
<html>
<head>
  <title>TWaver HTML5</title>
  <script src="twaver.js"></script>
  <script>
  function init() {
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);

    document.body.appendChild(network.getView());
    network.adjustBounds({x:0, y:0, width:500, height:500});

    var node1 = new twaver.Node();
    node1.setName("TWaver");
    node1.setLocation(100, 100);
    box.add(node1);

    var node2 = new twaver.Node();
    node2.setName("HTML5");
    node2.setLocation(300, 200);
    box.add(node2);

    var link = new twaver.Link(node1, node2);
    link.setName("Hello!");
    link.setToolTip("<b>Hello!</b>");
    box.add(link);
  }
  </script>
</head>
<body onload="init()">
</body>
</html>
```



概述

数据元素是数据模型的基本要素，用于描述图形网元，业务网元，或者纯数据。TWaver HTML5中所有数据元素都继承自twaver.Data。为不同功能的需求，预定义了三类数据类型：twaver.Element,twaver.Alarm,twaver.Layer,分别用来描述拓扑的网元，告警和图层。其中拓扑网元扩展定义了十几种网元类型，用以描述丰富的拓扑网元特性，其中最常用的几类拓扑网元包括：Node、Link、Group、SubNetwork、Grid等，TWaver中网元的继承关系如下图，本章将详细介绍这些网元以及其他数据类型的特性，使用以及扩展应用。



基本数据元素(Data)

twaver.Data是TWaver HTML5中最基本的数据元素，默认定义了id,name,name2,icon,toolTip,parent,children等基本属性，支持事件派发和监听，由它扩展出来还有twaver.Element,twaver.Alarm,twaver.Layer等数据类型。

事件派发与监听

twaver.Data继承于twaver.PropertyChangeDispatcher类,内部包含一个twaver.EventDispatcher实例对象，这使得Data具有派发事件和监听事件的功能，可以通过调用下面的方法派发事件或者添加监听器：

```
//派发属性变化事件
firePropertyChange:function(property,oldValue,newValue)
//添加属性变化监听器
addPropertyChangeListener:function(listener,scope,ahead)
//删除属性变化监听器
removePropertyChangeListener:function(listener)
//属性发生变化的处理方法
onPropertyChanged:function(listener)
```

基本属性

定义了一些基本属性，包括id,name,name2,icon,toolTip等。

```
/**
 *id:网元的id,如果未设置，TWaver内部会分配唯一的Id。
 */
twaver.Data:function(id)

getId:function()
setName:function(value)
getName:function()
setName2:function(value)
getName2:function()
setIcon:function(value)
getIcon:function()
setToolTip:function(value)
getToolTip:function()
```

Note:

name2为新增加的属性,增大了label设置的灵活性。

如果需要设置其他属性，可以通过setClient()/getClient()方法设置自定义的属性(包括对象)。

```
//设置Client属性;
```

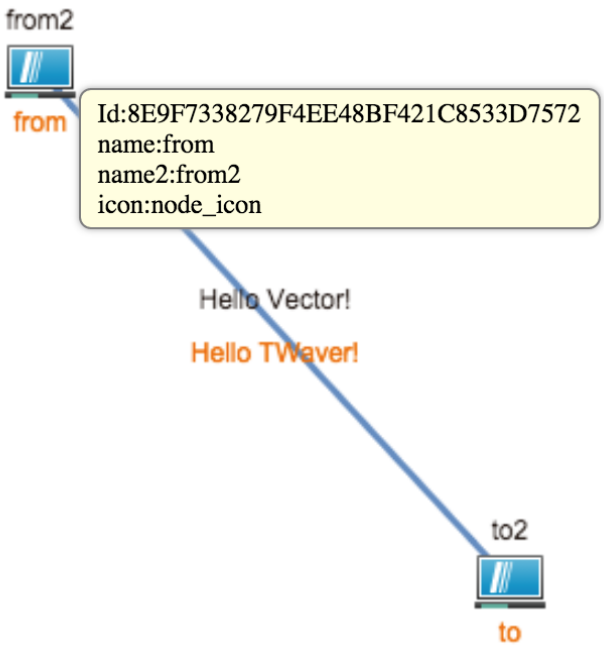
```
setClient = function(clientProp,newValue)
//获取Client属性
getClient = function(clientProp)
```

Note:

setClient()存放数据类似于Java中的HashMap。

示例:

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <title>Alarm</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});

            network.getToolTip = function (element) {
                var Id = element.getId();
                var name = element.getName();
                var name2 = element.getName2();
                var icon = element.getIcon();
                var clientProperty = element.getClient('clientProperty');
                return 'Id:' + Id + '<br>' + 'name:' + name + '<br>' + 'name2:' + name2+
                '<br>' + 'icon:' + icon;
            }
            twaver.Styles.setStyle('label.color','#ec6c00');
            twaver.Styles.setStyle('label2.color','#57ab9a');
            twaver.Styles.setStyle('select.color','#ef8200');
        }

        function initDataBox() {
            var node = new twaver.Node({
                name: 'from',
                name2: 'from2',
                location: {
                    x: 300,
                    y: 200
                }
            });
            box.add(node);

            var node2 = new twaver.Node({
                name: 'to',
                name2: 'to2',
                location: {
                    x: 500,
                    y: 250
                }
            });
            box.add(node2);

            var link = new twaver.Link(node, node2);
            link.setName('Hello TWaver!');
            link.setName2('Hello Vector!');
            link.setClient('clientProperty',node);
            box.add(link);
        }
    </script>

</head>
<body onload="init()">
</body>
</html>
```



Note:从示例中我们学会了什么？

- 统一设置样式，如:twaver.Styles.setStyle('label.color','#ec6c00')
- 自定义toolTip的提示语:重写network.getToolTip方法
- Client属性的使用
- TWaver内部内置一些颜色，twaver.Colors.***

在这我们特别列举出icon的布局问题，icon作为网元的附件可以围绕在网元的周围，呈现一些特殊信息。TWaver支持多组icon同时存在，且摆放的位置可以分别设置。下面列举使用方法：

```
function registerImage(){
  registerNormalImage('./images/list_view.png','list_view');
  registerNormalImage('./images/settings1.png','setting1');
  registerNormalImage('./images/ic_mech_wheel.png','wheel');
  registerNormalImage('./images/ic_search_archive.png','archive');
  registerNormalImage('./images/super_mono.png','mono');
  registerNormalImage('./images/twitter_04.png','twitter');
}
function registerNormalImage(url, name) {
  var image = new Image();
  image.src = url;
  image.onload = function() {
    twaver.Util.registerImage(name, image, image.width, image.height);
    image.onload = null;
    network.invalidateElementUIs();
  };
}
function initNode() {
    var node2 = new twaver.Node("node2");
    node2.setLocation(450, 205);
    box.add(node2);
    node2.setName('name1');
    node2.setName2('name2');
    node2.setSize(300,300);
    node2.setStyle('icons.names', [["mono","wheel",'archive'],["wheel",'archive','mono'],["archive",'mono'],["mono",'wheel']]);
    node2.setStyle('icons.position', ['topleft.topleft','topright.topright','bottomleft.bottomleft',
      'bottomright.bottomright']);
    node2.s('icons.orientation',['top','left','right','bottom']);
}
```

其他功能函数

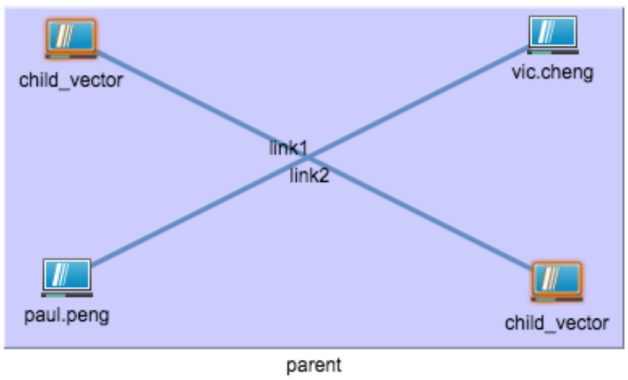
此外，Data中还定义了其他功能函数

```
//获取所有子网元
getChildren:function()
getChildSize:function()
//获取符合macthFunction的所有childs组成的List
toChildren:function(macthFunction,scope)
addChild:function(child,index)
removeChild:function(child)
getChildAt:function(index)
clearChildren:function()
getParent:function()
setParent:function(parent)
hasChildren:function()
isRelatedTo:function(data)
isParentOf:function(data)
isDescendantOf:function(data)
```

```
//toChildren:function(macthFunction,scope)的使用方法
parent.toChildren(function(e){
    return e.getName() === 'vector';
});
```

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <title>Child-RelationShip</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            registerImage();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            twaver.Styles.setStyle('select.color', '#57ab9a');
        }

        function registerImage() {
            //register shadow
            twaver.Util.registerImage('shadow', {
                w: 37,
                h: 29,
                shadowOffsetX: 0,
                shadowOffsetY: 0,
                shadowBlur: 5,
                shadowColor: '#ec6c00',
                v: [
                    {
                        shape: 'vector',
                        name: 'node_image',
                        x: 0,
                        y: 0
                    }
                ]
            });

        }

        function initDataBox() {
            var parent = new twaver.Group({
                name: 'parent',
                location: {x: 300, y: 400 },
            });
            box.add(parent);

            var node1 = new twaver.Node({
                name: 'Jeff.fu',
                location: {
                    x: 200,
                    y: 200
                }
            });
            node1.setClient('vector', true);
            box.add(node1);

            var node2 = new twaver.Node({
                name: 'alex.dong',
                location: {
                    x: 500,
                    y: 350
                }
            });
            node2.setClient('vector', true);
            box.add(node2);

            var node3 = new twaver.Node({
```

```
                name: 'paul.peng',
                location: {
                    x: 200,
                    y: 350
                }
            });
            box.add(node3);

            var node4 = new twaver.Node({
                name: 'vic.cheng',
                location: {
                    x: 500,
                    y: 200
                }
            });
            box.add(node4);

            var link = new twaver.Link(node1, node2);
            link.setName('link1');
            link.setStyle('label.position','topleft.topleft');
            box.add(link);

            var link2 = new twaver.Link(node3, node4);
            link2.setName('link2');
            box.add(link2);

            parent.addChild(node1);
            parent.addChild(node2);
            parent.addChild(node3);
            parent.addChild(node4);

            matchFunction = function (e) {
                if (e.getClient('vector')) {
                    return true;
                }
            }

            var childrenMatch = parent.toChildren(matchFunction);
            childrenMatch.forEach(function (element) {
                element.setImage('shadow');
                element.setName('child_vector');
            });

        }
    </script>

</head>
<body onload="init()">
</body>
</html>
```



Note:上述示例我们学会了什么？

- toChildren()方法的使用
- shadow阴影的设置

告警元素(Alarm)

TWaver中定义了告警，每个告警有告警级别，用以反映告警的紧急程度，告警使用AlarmBox进行管理，将告警与拓扑网元相关联。网元本身不存储具体的告警，而只存储当前告警状态信息。

告警(Alarm)

用来表示网管系统中设备故障或者网络异常的数据模型，与Element关联以反映网元的告警信息，Alarm中预定义了告警级别、告警是否已清除，告警是否已确认以及发出告警的网元id，用户也可以通过setClient()方法添加自己的属性。
告警中定义的属性如下：

```
/**
```

```
 * id:告警的Id
 * elementId:告警网元的Id
 * alarmSeverity:告警级别
 * isAcked:告警是否确认
 * isCleared:告警是否清除
 */
twaver.Alarm: function(id, elementId, alarmSeverity, isAcked, isCleared)

getElementId:function()
isAcked:function()
setAcked:function(value)
isCleared:function()
setCleared:function(value)
getAlarmSeverity:function()
setAlarmSeverity:function(value)
```

告警级别(AlarmSeverity)

告警级别用以反映告警的紧急程度，TWaver HTML5中预定义了六中告警级别，告警级别的value属性表示告警的严重程度，默认值越大告警越严重。

```
/**
 * value:
 * name:
 * nickName:
 * color:
 * displayName:
 */
twaver.AlarmSeverity:function(value,name,nickName,color,displayName)

//TWaver内部预定义六中告警级别
twaver.AlarmSeverity.CRITICAL = twaver.AlarmSeverity.add(500,'Critical','C',"#FF0000");
twaver.AlarmSeverity.MAJOR = twaver.AlarmSeverity.add(400,'Major','M',"#FFA000");
twaver.AlarmSeverity.MINOR = twaver.AlarmSeverity.add(300,'Minor','m',"#FFFF00");
twaver.AlarmSeverity.WARNING = twaver.AlarmSeverity.add(200,'Warning','W',"#00FFFF");
twaver.AlarmSeverity.INDETERMINATE = twaver.AlarmSeverity.add(100,'Indeterminate','N',"#C800FF");
twaver.AlarmSeverity.CLEARED = twaver.AlarmSeverity.add(0,'Cleared','R',"#00FF00");
```

AlarmSeverity中的级别都是静态变量，用户也可以全局注册或者卸载自己的告警级别，此外还提供清除所有告警级别的方法。

```
//添加告警级别
twaver.AlarmSeverity.add:function(value,name,nickName,colour,displayName)
//删除告警级别
twaver.AlarmSeverity.remove:function(name)
//清空告警级别
twaver.AlarmSeverity.clear:function()
```

Note:

因为告警级别是全局变量，删除告警级别会对整个程序产生影响，所以请谨慎操作。

告警状态(AlarmState)

实际应用中，告警的出现可能成千上万，面对告警风暴，直接与告警关联是沉重的,所以TWaver采用网元与告警分离，由告警容器去管理所有告警，网元本身只存储告警状态信息，如当前有多少条告警，最高级别是什么，而对于每条告警具体的信息存放在告警容器(AlarmBox)中。
网元告警状态用twaver.AlarmState来定义，用来反映新发告警的级别和数量。告警状态属性包括:确认告警的最高级别、新发告警的最高级别、该网元所有告警的最高级别、新发告警次高级别、资深告警最高级别，传递告警级别以及各级告警的数量。

```
getHighestAcknowledgedAlarmSeverity:function()
getHighestNewAlarmSeverity:function()
getHighestOverallAlarmSeverity:function()
hasLessSevereNewAlarms:function()
getAcknowledgedAlarmCount:function(severity)
getAlarmCount:function(severity)
getNewAlarmCount:function(severity)
setNewAlarmCount:function(severity,count)
getPropagateSeverity:function()
setPropagateSeverity:function(propagateSeverity)
isEmpty:function()
isEnablePropagation:function()
setEnablePropagation:function(enablePropagation)
```

修改告警状态的相关方法：确认告警，清除告警，增加/减少确认告警，删除告警...

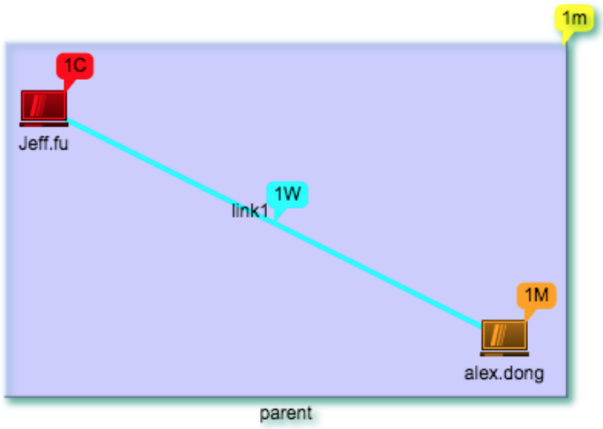
```
acknowledgeAlarm:function(severity)
acknowledgeAllAlarms:function(severity)
increaseAcknowledgesAlarm:function(severity,increment)
increaseNewAlarm:function(severity,increment)
decreaseAcknowledgedAlarm:function(severity,decrement)
decreaseNewAlarm: function(severity,decrement)
removeAllNewAlarms:function(severity)
setAcknowledgedAlarmCount:function(severity,count)
removeAllAcknowledgedAlarms:function(severity)
clear:function()
```

告警的使用

在使用告警时需要注意一点，告警增删都要通过alarmBox来操作，这点与网元需要在elementBox中增删是一致的。

示例：

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <title>Let me tell you how to use vector?</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            registerImage();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            twaver.Styles.setStyle('select.color', '#57ab9a');
        }

        function registerImage() {
            //register shadow
            twaver.Util.registerImage('shadow', {
                w: 37,
                h: 29,
                shadowOffsetX: 0,
                shadowOffsetY: 0,
                shadowBlur: 5,
                shadowColor: '#ec6c00',
                v: [
                    {
                        shape: 'vector',
                        name: 'node_image',
                        x: 0,
                        y: 0
                    }
                ]
            });

        }

        function initDataBox() {
            var parent = new twaver.Group({
                name: 'parent',
                location: {x: 300, y: 400 },
            });
            addAlarm("alarm 0",parent.getId(),twaver.AlarmSeverity.MINOR,box.getAlarmBox());
            box.add(parent);

            var node1 = new twaver.Node({
                name: 'Jeff.fu',
                location: {
                    x: 200,
                    y: 200
                }
            });
            addAlarm("alarm 1", node1.getId(), twaver.AlarmSeverity.CRITICAL, box.getAlarmBox());
            node1.setClient('vector', true);
            box.add(node1);

            var node2 = new twaver.Node({
                name: 'alex.dong',
                location: {
                    x: 500,
                    y: 350
                }
            });
            node2.setClient('vector', true);
            addAlarm("alarm 2", node2.getId(), twaver.AlarmSeverity.MAJOR, box.getAlarmBox());
            box.add(node2);

            var link = new twaver.Link(node1, node2);
            link.setName('link1');
            link.setStyle('label.position', 'topleft.topleft');
            addAlarm("alarm 3",link.getId(),twaver.AlarmSeverity.WARNING,box.getAlarmBox());
            box.add(link);

            parent.addChild(node1);
            parent.addChild(node2);
        }

        function addAlarm(alarmID, elementID, alarmSeverity, alarmBox) {
            var alarm = new twaver.Alarm(alarmID, elementID, alarmSeverity);
            alarmBox.add(alarm);
        }
    </script>
</head>
<body onload="init()">
</body>
</html>
```

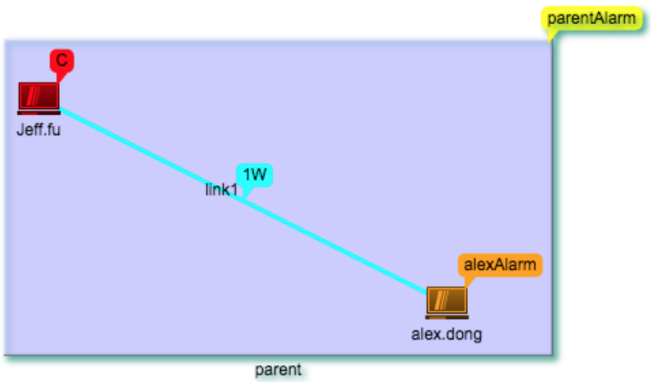


Note:

Alarm的冒泡中显示内容"1C",前面的数字是网元告警的数量。如为Node创建两个twaver.AlarmSeverity.CRITICAL级别的告警，冒泡中则显示"2C"，如果不想显示数字或者自定义告警显示内容，可参考如下的代码。

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <title>Let me tell you how to use vector?</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            registerImage();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            twaver.Styles.setStyle('select.color', '#57ab9a');

            network.getAlarmLabel = function (element) {
                var severity = element.getAlarmState().getHighestNewAlarmSeverity();
               if (severity) {
                    if (element.getName() === 'Jeff.fu') {
                        var label = severity.nickName;
                    } else if (element.getName() === 'alex.dong') {
                        var label = "alexAlarm";
                    } else if (element.getChildrenSize() !== 0) {
                        var label = "parentAlarm";
                    }else {
                        var label = element.getAlarmState().getNewAlarmCount(severity) + severity.nickName;
                    }
                    if (element.getAlarmState().hasLessSevereNewAlarms()) {
                        label += "+";
                    }
                    return label;
                }
                return null ;

            }
        }

        function registerImage() {
            //register shadow
            twaver.Util.registerImage('shadow', {
                w: 37,
                h: 29,
                shadowOffsetX: 0,
                shadowOffsetY: 0,
                shadowBlur: 5,
                shadowColor: '#ec6c00',
                v: [
                    {
                        shape: 'vector',
                        name: 'node_image',
                        x: 0,
                        y: 0
                    }
                ]
            });

        }

        function initDataBox() {
```

```
            var parent = new twaver.Group({
                name: 'parent',
                location: {x: 300, y: 400 },
            });
            addAlarm("alarm 0", parent.getId(), twaver.AlarmSeverity.MINOR, box.getAlarmBox());
            box.add(parent);

            var node1 = new twaver.Node({
                name: 'Jeff.fu',
                location: {
                    x: 200,
                    y: 200
                }
            });
            addAlarm("alarm 1", node1.getId(), twaver.AlarmSeverity.CRITICAL, box.getAlarmBox());
            node1.setClient('vector', true);
            box.add(node1);

            var node2 = new twaver.Node({
                name: 'alex.dong',
                location: {
                    x: 500,
                    y: 350
                }
            });
            node2.setClient('vector', true);
            addAlarm("alarm 2", node2.getId(), twaver.AlarmSeverity.MAJOR, box.getAlarmBox());
            box.add(node2);

            var link = new twaver.Link(node1, node2);
            link.setName('link1');
            link.setStyle('label.position', 'topleft.topleft');
            addAlarm("alarm 3", link.getId(), twaver.AlarmSeverity.WARNING, box.getAlarmBox());
            box.add(link);

            parent.addChild(node1);
            parent.addChild(node2);
        }

        function addAlarm(alarmID, elementID, alarmSeverity, alarmBox) {
            var alarm = new twaver.Alarm(alarmID, elementID, alarmSeverity);
            alarmBox.add(alarm);
        }

    </script>

</head>
<body onload="init()">
</body>
</html>
```



Note:

学会重写network.getAlarmLabel方法，实现Alarm告警提示语的自定义，在TWaver的开发中经常遇到,务必掌握。

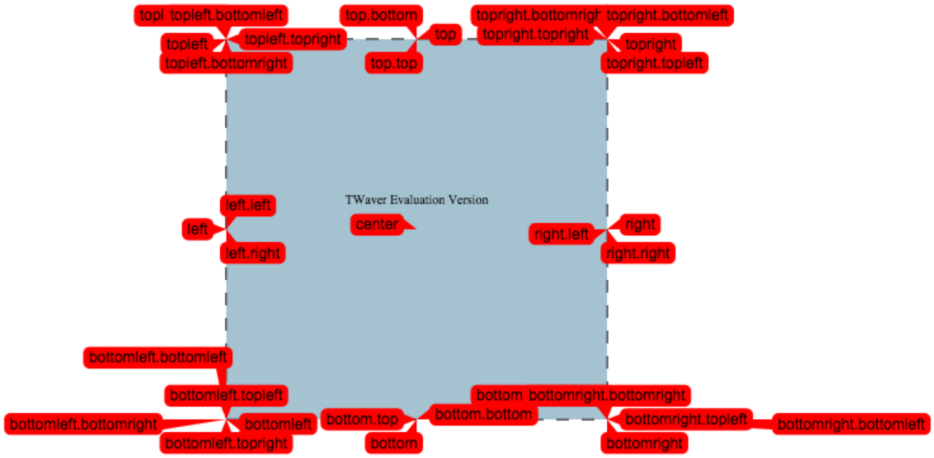
告警位置

在此重点介绍Alarm的两个参数,alarm.xoffset、alarm.yoffset和alarm.position。twaver可以为每个网元对象添加Alarm信息,默认位置为hotspot。将Alarm的位置分为两种情况，一种是相对于Node、ShapeNode、Group、Subnetwork等网元的位置,一种是相对于Link、ShapeLink等网元的位置。alarm除了position的属性可以更改位置信息外，还有两个参数可以更改alarm的位置信息，这两个属性为alarm.xoffset和alarm.yoffset。这两个属性使用起来有些特殊，在此特意强调下。

Node等网元

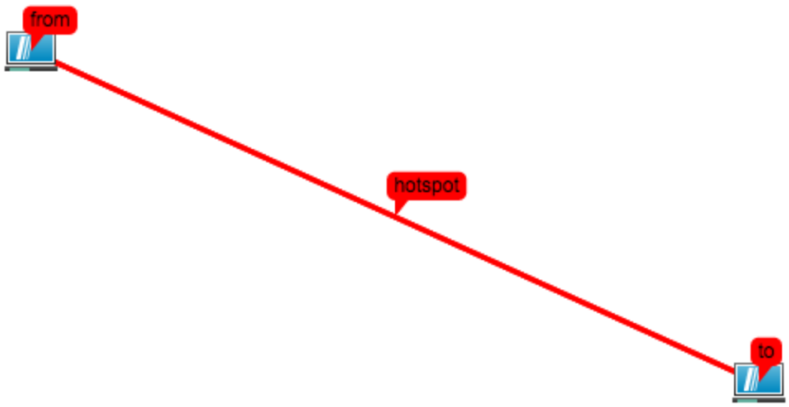
网元中Alarm的位置可以参考Alarm样式属性alarm.position一栏。具体位置可与下图一一对应。注意Node网元中Alarm的Position位置属性没有from和to选项，因为概念上是讲不通的。

对于Node网元alarm.xoffset和alarm.yoffset分别代表了相对于alarm当前位置的x轴和y轴偏移量。



Link等网元

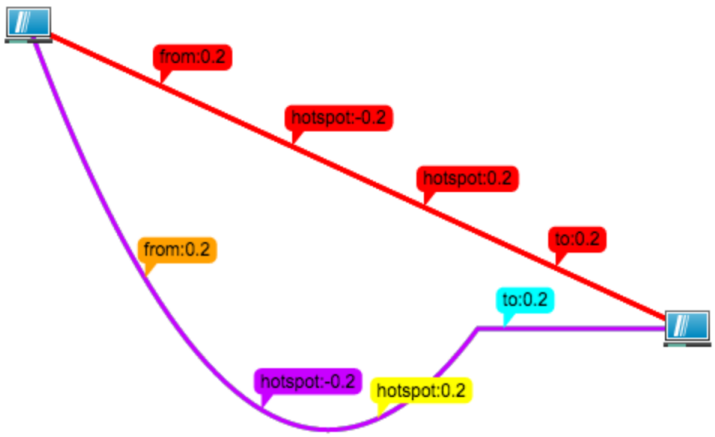
Link网元中alarm.position除了具有fron和to两个特殊的位置外，其他和上面的Node网元完全是一致的。在此着重介绍hotspot、from、to三个位置。



对于Link网元的alarm.xoffset和alarm.yoffset有个特殊的含义,如果将xoffset的值设置为正负(0-1)之间的值，则代表了alarm在Link上的百分比位置。当alarm.position设置为"from"，xoffset 取值为"0 – 1"则表示百分比，否则按实际值计算，偏移量均相对于起始from点，position为"to"，则反之。position为其他任何值，hotspot点为所有link长度一半的点，xoffset可取整数(0 – 1)、也可取负数(0-1)，都则按实际值计算。下面给出一些示例，可以认真体会。

```
positions = ["from","hotspot","to","hotspot"];
alarmSeverity = [twaver.AlarmSeverity.CRITICAL,twaver.AlarmSeverity.MAJOR,twaver.AlarmSeverity.MINOR,twaver.AlarmSeverity.WARNING,
for(var i=0;i<positions.length;i++){
  var node1 = new twaver.Node();
  node1.setCenterLocation(100,100);
  this.box.add(node1);

  var node2 = new twaver.Node();
  node2.setCenterLocation(540,300);
  this.box.add(node2);
  var link = new twaver.Link(node1, node2);
  this.box.add(link);
  link.s('alarm.position',positions[i]);
  if(i === 3){
    link.s('alarm.xoffset', -0.2);
  }else{
    link.s('alarm.xoffset', 0.2);
  }
  this.addAlarm("link" + i, link.getId(), twaver.AlarmSeverity.CRITICAL, this.box.getAlarmBox());

  var link = new twaver.ShapeLink(node1,node2);
  this.box.add(link);
  link.addPoint({
    x : 100 ,
   y : 100
 });

  var list = new twaver.List();
  list.add({
    x : 250,
    y : 500
  });
  list.add({
    x : 400,
    y : 300
  });
  link.addPoint(list);
  link.s('alarm.position',positions[i]);
  if(i === 3){
```

```
    link.s('alarm.xoffset', -0.2);
  }else{
    link.s('alarm.xoffset', 0.2);
  }
  this.addAlarm("shapelink" + i, link.getId(), alarmSeverity[i+1], this.box.getAlarmBox());
}
```



常见问题

网元告警同时闪烁

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
    <title>TWaver HTML5 Demo - Alarm</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
    var network = new twaver.vector.Network();
    var box = network.getElementBox();

    function init() {
        initNetwork();
        initBox();
    }

    function initNetwork(){
       var view = network.getView();
       document.body.appendChild(view);
       network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
   }

   function initBox() {
    var self = this;

    var node1 = new twaver.Node("node1");
    node1.setName("node1");
    node1.setLocation(100, 100);
    box.add(node1);

    addAlarm("alarm 1", node1.getId(), twaver.AlarmSeverity.CRITICAL, box.getAlarmBox());

    var oldAlarmLabel = network.getAlarmLabel;
    var newAlarmLabel = function(element){
        if(element.getName() == 'node1' || element.getName() == 'BSC_2' || element.getName() == 'BSC_3' ){
            return null;
        }
        return Network.prototype.getAlarmLabel(element);
    }

    setInterval(function(){
        if(self.network.getAlarmLabel == oldAlarmLabel){
            self.network.getAlarmLabel = newAlarmLabel;
            self.network.getInnerColor  = function(data){
                return "#FF00FF";
            }
        } else if(self.network.getAlarmLabel === newAlarmLabel){
            self.network.getAlarmLabel = oldAlarmLabel;
            self.network.getInnerColor  = function(data){
                return "#FFF00F";
            }
        }
        self.network.invalidateElementUIs();
    },500);
}

function addAlarm(alarmID, elementID, alarmSeverity, alarmBox) {
    var alarm = new twaver.Alarm(alarmID, elementID, alarmSeverity);
    alarmBox.add(alarm);
}
</script>
</head>
```

```
<body onload="init()" style="margin:0; ">
</input>
</body>
</html>
```

定制告警Attachment

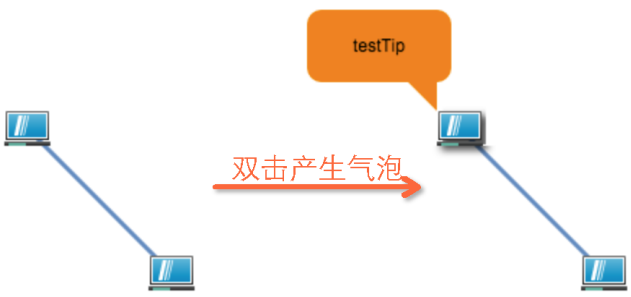
双击网元之后弹出自定义的告警图标，当然我们也可以使用相同的方法定制其他的Attachment。

```
/**
 * 自定义的网元以及附件attachment
 */
CNode = function(id) {
    CNode.superClass.constructor.call(this, id);
}

twaver.Util.ext("CNode", twaver.Node, {
    getVectorUIClass: function (){
        return CNodeUI;
    }
});

CNodeUI = function(network, element) {
    CNodeUI.superClass.constructor.call(this, network, element);
}

twaver.Util.ext('CNodeUI', twaver.vector.NodeUI, {
    checkAttachments: function() {
        CNodeUI.superClass.checkAttachments.call(this);
        this.checkCAttachment();

    },
    checkCAttachment: function () {
        var tip = this._element.getClient("tip");
        var showTip = this._element.getClient("show.tip");
    if (tip != null && tip !== "" && showTip !=null && showTip !=false) {
            if (!this._CAttachment) {
                this._CAttachment = new CAttachment(this);
                this.addAttachment(this._CAttachment);
            }
        } else {
            if (this._CAttachment) {
                this.removeAttachment(this._CAttachment);
                this._CAttachment = null;
            }
        }
    }
});

CAttachment = function(elementUI, showInAttachmentDiv) {
    CAttachment.superClass.constructor.call(this, elementUI, showInAttachmentDiv);
};

twaver.Util.ext('CAttachment', twaver.canvas.BasicAttachment, {
    paint: function (ctx) {
        CAttachment.superClass.paint.apply(this, arguments);
        _twaver.g.drawText(ctx, this.text, this._contentRect, this.font, this.getStyle('alarm.color'));
    },
    validate: function () {
        this.font = null;
        if(this._element.getClient("tip.font")) {
            this.font = this._element.getClient("tip.font");
        }
        this.text = this._element.getClient("tip");
        this._textSize = _twaver.g.getTextSize(this.font, this.text);
        this._fillColor = this._element.getClient("tip.fill.color");
        twaver.canvas.LabelAttachment.superClass.validate.call(this);
    },
    getContentWidth: function () {
        if(this._element.getClient("tip.width")) {
            return this._element.getClient("tip.width");
        }
        return this._textSize ? this._textSize.width : 0;
    },
    getContentHeight: function () {
        if(this._element.getClient("tip.height")) {
            return this._element.getClient("tip.height");
        }
        return this._textSize ? this._textSize.height : 0;
    },
    getCornerRadius: function () {
        if(this._element.getClient('tip.corner.radius')) {
            return this._element.getClient('tip.corner.radius');
        }
        return 5;
    },
    getPointerLength: function () {
        if(this._element.getClient('tip.pointer.length')) {
            return this._element.getClient('tip.pointer.length');
```

```
        return 10;
    },
    getPointerWidth: function () {
        if(this._element.getClient('tip.pointer.width')) {
            return this._element.getClient('tip.pointer.width');
        }
        return 8;
    },
    getPosition: function () {
        if(this._element.getClient('tip.position')) {
            return this._element.getClient('tip.position');
        }
        return "topleft.topleft";// 'topleft.topleft', 'top.top', 'topright.topright', 'right.right', 'left.left', 'bottom.bottom
    },
    getXOffset: function () {
        if(this._element.getClient('tip.xoffset')) {
            return this._element.getClient('tip.xoffset');
        }
        return 0;
    },
    getYOffset: function () {
        if(this._element.getClient('tip.yoffset')) {
            return this._element.getClient('tip.yoffset');
        }
        return 0;
    },
    getPadding: function () {
        if(this._element.getClient('tip.padding.left')) {
            return this._element.getClient('tip.padding.left');
        }
        return 0;
    },
    getPaddingLeft: function () {
        if(this._element.getClient('tip.padding.left')) {
            return this._element.getClient('tip.padding.left');
        }
        return 0;
    },
    getPaddingRight: function () {
        if(this._element.getClient('tip.padding.right')) {
            return this._element.getClient('tip.padding.right');
        }
        return 0;
    },
    getPaddingTop: function () {
        if(this._element.getClient('tip.padding.top')) {
            return this._element.getClient('tip.padding.top');
        }
        return 0;
    },
    getPaddingBottom: function () {
        if(this._element.getClient('tip.padding.bottom')) {
            return this._element.getClient('tip.padding.bottom');
        }
        return 0;
    },
    getDirection: function () {
        if(this._element.getClient('tip.direction')) {
            return this._element.getClient('tip.direction');
        }
        return "aboveleft";
    },
    isFill: function () {
        return this._fillColor != null;
    },
    getFillColor: function () {
        return this._fillColor;
    },
    getGradient: function () {
        if(this._element.getClient('tip.gradient')) {
            return this._element.getClient('tip.gradient');
        }
        return "none";
    },
    getGradientColor: function () {
        if(this._element.getClient('tip.gradient.color')) {
            return this._element.getClient('tip.gradient.color');
        }
        return "#FFFFFF";
    },
    getOutlineWidth: function () {
        if(this._element.getClient('tip.outline.width')) {
            return this._element.getClient('tip.outline.width');
        }
        return -1;
    },
    getOutlineColor: function () {
        if(this._element.getClient('tip.outline.color')) {
            return this._element.getClient('tip.outline.color');
        }
        return "#000000";
    },
    getCap: function () {
        if(this._element.getClient('tip.cap')) {
            return this._element.getClient('tip.cap');
        }
        return 'butt';
    },
    getJoin: function () {
        if(this._element.getClient('tip.join')) {
            return this._element.getClient('tip.join');
        }
```

```
        return 'miter';
    },
    getAlpha: function () {
        if(this._element.getClient('tip.alpha')) {
            return this._element.getClient('tip.alpha');
        }
        return 1;
    },
    isShadowable: function () {
        return false;
    }
});
```

```
/**
 * 使用自定义的网元
 */
<!DOCTYPE html >
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="../../twaver.js"></script>
    <script src="./CNode.js"></script>
</head>
<script type="text/javascript">
var box = new twaver.ElementBox();
var network = new twaver.vector.Network(box);
function init() {
    initNetwork();
    initDataBox();
}

function initNetwork(){
 document.body.appendChild(network.getView());
 network.getView().style.background = '#E9E9E9';
 network.adjustBounds({x: 10, y: 10, width: 500, height: 300});

 network.addInteractionListener(function(e) {
    console.log(e.kind);
    if(e.kind == "doubleClickElement") {
        if(e.element.getClient("show.tip")) {
            e.element.setClient("show.tip", false);
        } else {
            e.element.setClient("show.tip", true);
        }
    }
});
}

function initDataBox(){
 var node1 = new CNode();
 node1.setClient("show.tip", false);
 node1.setClient("tip","testTip");
 node1.setClient("tip.fill.color","rgba(255,255,0,0.8)");
 node1.setClient("tip.width",80);
 node1.setClient("tip.height",50);
 node1.setClient("tip.corner.radius",10);
 node1.setClient("tip.pointer.length",20);
 node1.setClient("tip.pointer.width",20);

 var node2 = new CNode();
 node2.setClient("show.tip", false);
 node2.setClient("tip","testTip2");
 node2.setClient("tip.fill.color","rgba(255,255,0,0.8)");
 node2.setClient("tip.width",80);
 node2.setClient("tip.height",50);
 node2.setClient("tip.corner.radius",10);
 node2.setClient("tip.pointer.length",20);
 node2.setClient("tip.pointer.width",20);

 var link = new twaver.Link(node1, node2);

 node1.setLocation(200, 100);
 node2.setLocation(300, 200);

 box.add(node1);
 box.add(node2);
 box.add(link);
}
</script>
<body onload="init();">
</body>
</html>
```



图层元素(Layer)

图层的属性

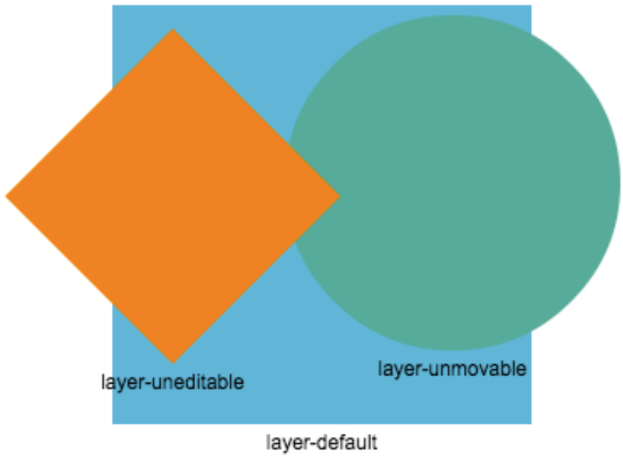
twaver.Layer(图层)继承于twaver.Data,用于描述拓扑网元的图层信息，Layer有四个特殊属性：visible,editable,movable,rotatable。

```
setVisible:function()
getVisible:function()
setMovable:function()
getMovable:function()
setEditable:function()
getEditable:function()
setRotatable:function()
getRotatable:function()
```

图层的使用

TWaver DataBox可用于存放所有的Element,Element上包含Layer属性,默认在TWaver中预定义的一个图层(默认图层),这个图层不可更改。TWaver HTML5中的层次关系由LayerBox来管理，默认的层次顺序由父子关系和加入的先后顺序决定,先加入则在下面。拓扑图中，每个Element通过设置LayerID与某个Layer相关联以控制网元的显示层次。下面的例子展示了图层的使用以及图层三个属性的作用：图层的上下移动等更多说明请参考LayerBox。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test Zoom</title>
    <script src="../twaver.js"></script>
    <script>
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initLayer();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            twaver.Styles.setStyle('select.color', '#57ab9a');
            network.setEditInteractions();
        }

        function initLayer() {
            var layerBox = box.getLayerBox();
            var layer1 = new twaver.Layer('unmovable', 'unmovable layer');
            layer1.setMovable(false);
            var layer2 = new twaver.Layer('uneditable', 'uneditable layer');
            layer2.setEditable(false);
            var layer3 = new twaver.Layer('unvisible', 'unvisible Layer');
            layer3.setVisible(false);

            layerBox.add(layer1);
            layerBox.add(layer2);
            layerBox.add(layer3);

            createNode(layer1, 'circle', 300, 100, 200, 200, '#57ab9a');
            createNode(layer2, 'diamond', 350, 200, 200, 200, '#ef8200');
            createNode(layer3, 'rectangle', 200, 200, 200, 200, '#ec6c00');
            createNode(layerBox.getDefaultLayer(), 'rectangle', 100, 100, 250, 250, '#61b6d8');
        }

        function createNode(layer, shape, x, y, width, height, fillColor) {
            var node = new twaver.Node();
            node.setLayerId(layer.getId());
            node.setName('layer-' + layer.getId());
            node.setStyle('body.type', 'vector');
            node.setStyle('vector.fill.alpha', 0.7);
            node.setStyle('vector.shape', shape);
            node.setSize(width, height);
            node.setLocation(x, y);
            node.setStyle('vector.fill.color', fillColor);
            box.add(node);
            return node;

        }
    </script>
</head>
<body onload="init()">
</body>
</html>
```



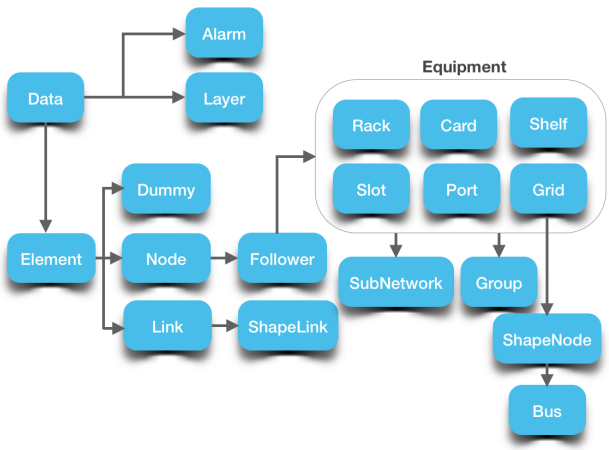
Note:

我们也许会遇到这样的问题，想永远保持Link位于Node的下方，当然可以使用Layer来管理，但不是很方便，这里提供一种简单的方法，twaver.Link.prototype.isAdjustedToBottom = function () {
return true;
};

拓扑元素(Element)

概述

TWaver HTML5中的twaver.Element表示拓扑网元，是TWaver中最重要的数据元素类型。拓扑元素用于拓扑图，主要分为三大类，Dummy、Node、Link。其中Dummy在拓扑图中是不可见的，在树组件上可见，通常设置为其他节点的父节点，表示类别或网元组，如将所有的Link类型网元放在一个Dummy节点下，表示Link分类。Node节点是最常用的网元类型，表示实体对象，包括节点、网元组、子网、设备等。连线表示节点之间的链接关系，ShapeLink继承于Link，可以用于表示不规则走向的连线。下图中Element部分为拓扑元素的继承结构，其中设备面板元素类型在TWaver HTML5的Demo中提供。



网元(Element)

Element继承于Data接口,扩展定义了告警状态(alarmState)、图层编号(layerID)、视图类型(elementUIClass)等属性。

```
setLayerID:function(layerId)
getLayerID:function()
getAlarmState:function()
//elementUI
getElementUIClass:function()
//canvasUI
getCanvasUIClass:function()
//vectorUI
getVectorUIClass:function()
```

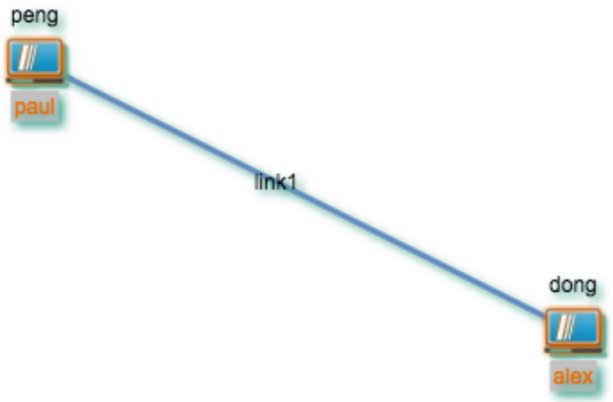
其中的vectorUIClass为VectorUI类型及其扩展类型，不同类型网元同城对应不同类型的VectorUI类。VectorUI是网元在拓扑图中的视图组件，两者构成一个数据与视图分离的模型结构，关于VectorUI将在后面的视图章节详细介绍。Element继承了IStyle接口，定义了get/setStyle()方法或s(style,value)方法用于设置、获

取网元样式，包括颜色、边框、背景、对齐方式等，通常不同的网元的有不同的样式属性，样式列表可参考附录。

```
//设置网元样式
setStyle:function(styleProp,newValue)
//获取网元样式
getStyle:function(styleProp,returnDefaultIfNull);
//样式设置
s:function(style,value)
```

下面的网元分别设置了name、name2的颜色：

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            registerImage();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            twaver.Styles.setStyle('select.color', '#57ab9a');
        }

        function registerImage() {
            //register shadow
            twaver.Util.registerImage('shadow', {
                w: 37,
                h: 29,
                shadowOffsetX: 0,
                shadowOffsetY: 0,
                shadowBlur: 5,
                shadowColor: '#ec6c00',
                v: [
                    {
                        shape: 'vector',
                        name: 'node_image',
                        x: 0,
                        y: 0
                    }
                ]
            });
        }

        function initDataBox() {
            //样式设置方式一
            var node1 = new twaver.Node();
            node1.setName('paul');
            node1.setName2('peng');
            node1.setLocation(200,200);
            node1.setImage('shadow');
            node1.setStyle('label.color','#ef6c00');
            node1.setStyle('label.fill','#ef8200');
            box.add(node1);

            //样式设置方式二
            var node2 = new twaver.Node({
                name: 'alex',
                name2:'dong',
                location: {
                    x: 500,
                    y: 350
                },
                image:'shadow'
            });
            node2.s('label.color','#ef6c00');
            node2.s('label.fill','#ef8200');

            box.add(node2);

            var link = new twaver.Link(node1, node2);
            link.setName('link1');
            link.s('label.position', 'topleft.topleft');
            box.add(link);
        }
    </script>
</head>
<body onload="init()">
</body>
</html>
```



此外Element还定义了isAdjustedToBottom()方法，表示是否置底显示，默认为false。

```
isAdjustedToBottom:function(){
    return false;
}
```

twaver.Group类型网元重写了这个方法，返回true，表示置底显示，这样就不会覆盖其孩子节点，而是位于孩子节点之下显示，如图效果：

节点(Node)

twaver.Node继承于twaver.Element,表示拓扑图上的节点。Node通常表示实体对象，可以作为连线的端点，通过x,y,width,height属性确定位置和尺寸,能设置图形等属性。

twaver.Node的图标显示并不是那么的单调，Node不仅支持传统的位图，而且对矢量图也是完全支持的(详细可参考矢量图与动态属性)。

```
//Node的属性和方法
get/setX:function(x)
get/setY:function(y)
get/setCenterLocation:function(point)
get/setLocation:function(point)
get/setWidth:function(width)
get/setHeight:function(height)
translate:function(x,y)
```

获取与节点相连的连线，如果没有连线与节点相连，返回的是null。

```
getLoopedLinks:function()
getLinks:function()
getAgentLinks:function()
getFromLinks:function()
getToLinks:function()
hasAgentLinks:function()
getFromAgentLinks:function()
getToAgentLinks:function()
```

此外节点上还可以添加跟随者跟随者能随节点的移动而移动，我们将在Follower章节详细的介绍。

```
getFollowers:function()
```

Node的使用在TWaver中最为常见，也最为重要，下面介绍正确使用Node的基本步骤。

(1).创建DataBox和Network,并绑定Network和DataBox之间的关系,DataBox用于存放所有网元数据，Network用于显示网元;

(2).如果需要自定义图形，则需要先注册图形，图形包括位图和矢量图等。

注册方法：twaver.Util.registerImage:function(name,source,width,height)。具体的使用方法可以参考自定义图形模块。

(3).创建一个新Node:var node = new twaver.Node();

(4).设置Node的属性:名称、位置、样式等等属性;

(5).将新创建的node添加到DataBox中:box.add(node);

这样刚创建的Node将显示在Network中,这不仅仅是使用Node的一般方法，也是使用TWaver中网元的一般方法。

HTML节点(HTMLNode)

twaver.HTMLNode继承于twaver.Node,是一种专门用于呈现HTML元素的网元,在HTMLNode上，您可以将任何HTML元素添加到该网元上,包括div、image、svg、table等等。与之对应使用的是HTMLLink,用于在Link上呈现HTML元素，具体使用和HTMLNode完全一致，不再详细介绍。

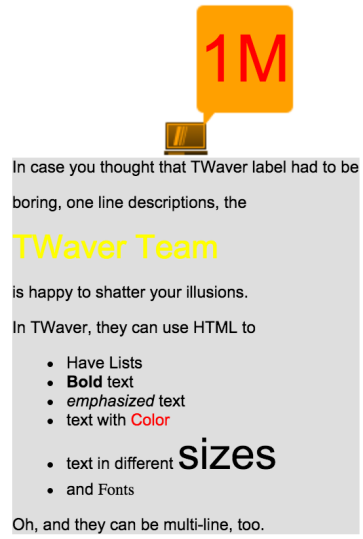
既然HTMLNode继承于Node,那么自然HTMLNode会具有Node的所有特点,唯一的区别在于HTMLNode的LabelAttachment和AlarmAttachment可以支持HTML样式。twaver默认情况下注册了"attachment.label.style"和"attachment.alarm.style"为"html",当选择值为"none"的时候，HTMLNode和Node完全一致。

```
node.s('attachment.label.style','html');//'none', 'html'
```

```
node.s('attachment.alarm.style','html');//'none', 'html'
```

示例一:呈现简单的HTML风格的字符串

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv = Content-Type content = "text/html; charset=utf-8">
  <title></title>
  <script type="text/javascript" src="../libs/twaver.js"></script>
  <script type="text/javascript">
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);

    function init() {
      initNetwork();
      initDataBox();
    }

    function initNetwork(){
      var networkDom = network.getView();
      document.getElementById("main").appendChild(networkDom);
      network.adjustBounds({x:0,y:0,width:1600,height:800});
      network.getAlarmLabel = function(element){
        var severity = element.getAlarmState().getHighestNewAlarmSeverity();
        if (severity) {
          var label = element.getAlarmState().getNewAlarmCount(severity) + severity.nickName;
          if (element.getAlarmState().hasLessSevereNewAlarms()) {
            label += "+";
          }
          label = "<div><p><font color=red size=+6>" + label + "</font></div>";
          return label;
        }
        return null;
      }
    }

    function initDataBox(){
      var htmlNode = new twaver.HTMLNode({
        name:"<div style='background:rgba(200,200,200,0.6)'>In case you thought that TWaver label had to be" +
        "<p>boring, one line descriptions, the " +
        "<p><font color=yellow size=+2>TWaver Team</font>" +
        "<p> is happy to shatter your illusions.<p>" +
        "In TWaver, they can use HTML to " +
        "<ul><li>Have Lists<li><b>Bold</b> " +
        "<li><em>emphasized</em> " +
        "<li>text with <font color=red>Color</font>" +
        "<li>text in different <font size=+3>sizes</font>" +
        "<li>and <font face=AvantGarde>Fonts</font></ul>" +
        "Oh, and they can be multi-line, too.</div>",
        location:{x: 600, y:200}
      });
      box.add(htmlNode);
      addAlarm("alarm 2", htmlNode.getId(), twaver.AlarmSeverity.MAJOR, box.getAlarmBox());
    }

    function addAlarm(alarmID, elementID, alarmSeverity, alarmBox) {
      var alarm = new twaver.Alarm(alarmID, elementID, alarmSeverity);
      alarmBox.add(alarm);
    }

  </script>
</head>
<body onload="init()">
  <div id="main" style="position:relative;">
  </div>
</body>
```

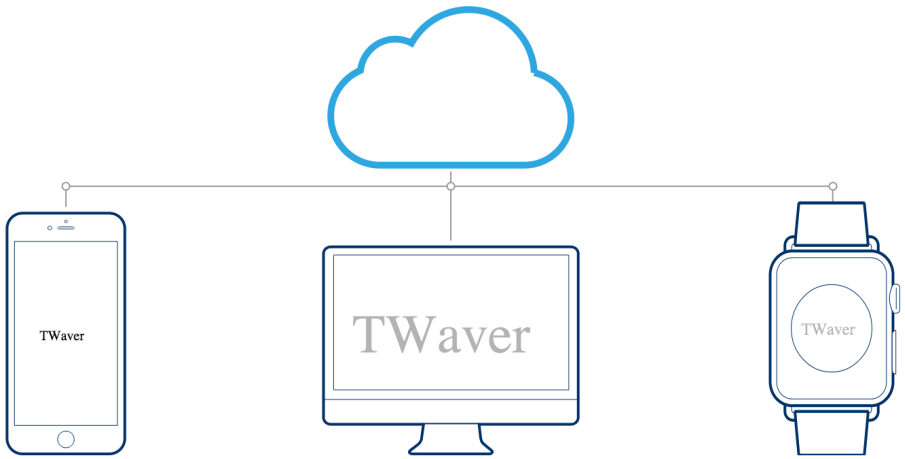


示例二:呈现GIF图片

```
for(var i=0;i<10;i++){
        var htmlNode = new twaver.HTMLNode({
          width:95,
          height:95,
          name:"<image src='../images/run.gif' style='width:100px;height:100px'/>",
          location:{x: 100 + 50*i, y:200+ i%2 * 100}
        });
        box.add(htmlNode);
        htmlNode.setStyle('label.position','center');
      }
```

示例三:呈现SVG资源

```
var svgNode = new twaver.HTMLNode();
      svgNode.setWidth(660);
      svgNode.setHeight(330);
      svgNode.setImage("rect");
      svgNode.setName("<svg id='svg2' xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'' xmlns='http://www.w3.org/2000/svg' h
      svgNode.setLocation(300,150);
      box.add(svgNode);
```



示例四:嵌入Table

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv = Content-Type content = "text/html; charset=utf-8">
  <title></title>
  <script type="text/javascript" src="../libs/twaver.js"></script>
  <script type="text/javascript">
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    var table = new twaver.controls.Table(box);
    var node,freshTableTimerId ;

    function init() {
      initNetwork();
      initDataBox();
      initTable();
    }

    function initNetwork(){
      var networkDom = network.getView();
      document.getElementById("main").appendChild(networkDom);
      network.adjustBounds({x:0,y:0,width:1300,height:800});
      network.setMaxZoom(100);
```

```
    }

    function initDataBox(){
      node = new twaver.HTMLNode();
      node.setImage(null);
      var html = "";
      html += "" +
      "<div style='width:400px;'> " +
      "    <div id='tableBoxTitle' style='width:400px;height:20px;text-align:center;background-color:#ccc;-moz-border-radius-topri
      "        <span>表格标题栏</span>" +
      "    </div> " +
      "    <div id='tableBox' style='width:400px;'> " +
      "    </div> " +
      "</div>";
      node.setName(html);
      node.setLocation(200, 0);
      box.add(node);

      var i = 10;
      while (i-- > 0) {
        var data = new twaver.Node();
        data.setName("TWaver-" + i);
        data.setLocation(Math.random()*600 + 400, Math.random()*600 + 100);
        data.s('inner.color', randomColor());
        box.add(data);
      }
    }

    function initTable(){
      network.addViewListener(function(e){
        if(e.kind === 'validateEnd'){
          var tableBox = document.getElementById('tableBox');
          var tablePane = new twaver.controls.TablePane(table);
          var tableDom = tablePane.getView();
          tableDom.style.width = "400px";
          tableDom.style.height = "200px";
          tableBox.appendChild(tableDom);
        }
      });

      window.onresize = function () {
        tablePane.invalidate();
      }
      createColumn(table, 'Name', 'name', 'accessor', 'string');
      createColumn(table, 'Id', 'id', 'accessor', 'string');
      createColumn(table, 'Icon', 'icon', 'accessor');
      var column = createColumn(table, 'Inner Color', 'inner.color', 'style', 'color');
      column.setSortFunction(function (color1, color2) {
        if (!color1) {
          return -1;
        }
        if (!color2) {
          return 1;
        }
        var number1 = parseInt(color1.substring(1), 16);
        var r1 = (number1 >> 16) & 0xff;
        var g1 = (number1 >> 8) & 0xff;
        var b1 = number1 & 0xff;
        var number2 = parseInt(color2.substring(1), 16);
        var r2 = (number2 >> 16) & 0xff;
        var g2 = (number2 >> 8 ) & 0xff;
        var b2 = number2 & 0xff;
        return (r1 + g1 + b1) - (r2 + g2 + b2);
      });
    }

    function zoomIn(){
      network.zoomIn();
    }

    function zoomOut(){
      network.zoomOut();
    }

    function registerImage(){
    }

    function registerNormalImage(url, name) {
      var image = new Image();
      image.src = url;
      image.onload = function() {
        twaver.Util.registerImage(name, image, image.width, image.height);
        image.onload = null;
        network.invalidateElementUIs();
      };
    }

    function createColumn(table, name, propetyName, propertyType, valueType) {
      var column = new twaver.Column(name);
      column.setName(name);
      column.setPropertyName(propetyName);
      column.setPropertyType(propertyType);
      if (valueType) {
        column.setValueType(valueType);
      }
      table.getColumnBox().add(column);
      return column;
    }

    function randomColor() {
      var r = randomInt(255);
      var g = randomInt(255);
```

```
      var b = randomInt(255);
      return '#' + formatNumber((r << 16) | (g << 8) | b);
    }

    function randomInt(n) {
      return Math.floor(Math.random() * n);
    }

    function formatNumber(value) {
      var result = value.toString(16);
      while (result.length < 6) {
        result = '0' + result;
      }
      return result;
    }
  </script>
</head>
<body onload="init()">
  <div>
    <button onclick="zoomIn()">
      Zoom In
    </button>
    <button onclick="zoomOut()">
      Zoom Out
    </button>
    <button onclick="removeNode()">
      ClearBox
    </button>
  </div>
  <br/>
  <div id="main" style="position:relative;">
  </div>
</body>
</html>
```

形状节点(ShapeNode)

TWaver提供了一个特殊的节点:twaver.ShapeNode。这是一个节点扩展,与普通节点(Node)之间主要区别是:普通节点主要通过图片来表示一个物体,而形状节点通过一个任意形状来表示一个物体。形状节点由一系列点组成,这些点连接成一个形状。通过千变万化的形状,可以用来表示很多特殊的数据,比如一个国家或者地区、一个总线等等。

添加删除控制点相关方法：

```
get/setPoint:function(points)
addPoint:function(point)
addPointAt:function(point,index)
setPointAt:function(point,index)
removePoint:function(point)
```

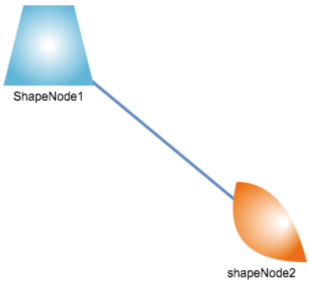
此外还可以指定片段的绘制方式(segments),使ShapeNode表现更丰富的效果，如曲线(QuadTo)，间断(moveTo)等。绘制方式有三种，移动到(moveTo)、直线(lineTo)和曲线(quadto)。

```
"moveto"、"lineto"、"quadto"
```

```
//设置segments
get/setSegments:function(list)
```

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

```

```
        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            twaver.Styles.setStyle('select.color', '#57ab9a');
        }

        function initDataBox() {
            var shapeNode1 = new twaver.ShapeNode({
                name:'ShapeNode1',
            });
            shapeNode1.s('vector.fill.color','#61b6d8');
            shapeNode1.s('vector.gradient','radial.center');

            shapeNode1.addPoint({
                x : 30,
                y : 10
            });
            shapeNode1.addPoint({
                x : 80,
                y : 10
            });
            shapeNode1.addPoint({
                x : 100,
                y : 90
            });
            shapeNode1.addPoint({
                x : 10,
                y : 90
            });
            shapeNode1.addPoint({
                x : 30,
                y : 10
            });
            box.add(shapeNode1);

            var shapeNode2 = new twaver.ShapeNode();
            shapeNode2.setName('shapeNode2');
            shapeNode2.addPoint({
                x : 130,
                y : 110
            });
            shapeNode2.addPoint({
                x : 180,
                y : 110
            });
            shapeNode2.addPoint({
                x : 200,
                y : 190
            });
            shapeNode2.addPoint({
                x : 110,
                y : 190
            });
            shapeNode2.addPoint({
                x : 130,
                y : 110
            });
            shapeNode2.s('vector.fill.color','#ec6c00');
            shapeNode2.s('vector.gradient','radial.east');
            var segments = new twaver.List();
            segments.add("moveto");
            segments.add("quadto");
            segments.add("quadto");
            shapeNode2.setSegments(segments);
            box.add(shapeNode2);

            var link = new twaver.Link(shapeNode1, shapeNode2);
            box.add(link);

        }
    </script>

</head>
<body onload="init()">
</body>
</html>
```



下图为使用ShapeNode绘制地图的效果：

Note:

上面使用ShapeNode来绘制地图说明了一个问题，TWaver最大的价值不在于使用内部的网元搭建应用场景,而是我们都会提供接口供您绘制任何网元形状,做到个性化定制,只要您敢于去构想，敢于去创新,TWaver也将会随着您的能力而发挥出无穷的潜力。

连线(Link)

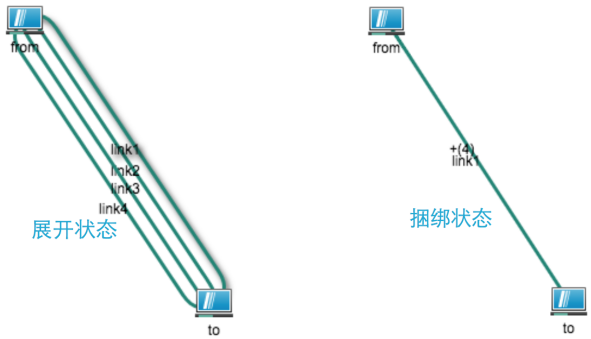
连线通常表示为节点间的连接关系，需要指定起始结束节点，TWaver支持自环，即起始结束节点为同一个节点。直接与连线相连的节点我们称之为fromNode,toNode,如果起始或者结束节点放在网元组中，网元组合并状态时，外观上网元组和连线相连，这个时候此网元组为代理节点。当节点之间有多条连线时，TWaver支持连线的捆绑和展开，默认双击其中一条连线即可实现两种状态的切换，绑定时显示的连线称为绑定代理，默认将第一条连线设为代理，用户也可以指定代理。

```
//获取、设置起始点
get/setFromNode:function(fromNode)
//获取、设置结束点
get/setToNode:function(toNode)
//获取代理起始点
getFromAgent:function()
//获取代理结束点
getToAgent:function()
getBundleLinks:function()
getBundleCount:function()
getBundleIndex:function()
reverseBundleExpanded:function()
isBundleAgent:function()
```

示例：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test Zoom</title>
    <script src="../twaver.js"></script>
    <script>
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }

        function initDataBox() {
            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                }
            });
            box.add(from);

            var to = new twaver.Node({
                name:'to',
                location:{
                    x:300,
                    y:300
                }
            });
            box.add(to);

            var link = new twaver.Link(from,to);
```

```
            link.setName('link1');
            link.s('link.color','#238475');
            link.s('label.xoffset',-10);
            link.s('label.yoffset',-10);
            box.add(link);
            var link = new twaver.Link(from,to);
            link.setName('link2');
            link.s('link.color','#238475');
            box.add(link);
            var link = new twaver.Link(from,to);
            link.setName('link3');
            link.s('label.xoffset',10);
            link.s('label.yoffset',10);
            link.s('link.color','#238475');
            box.add(link);
            var link = new twaver.Link(from,to);
            link.setName('link4');
            link.s('label.xoffset',10);
            link.s('label.yoffset',20);
            link.s('link.color','#238475');
            box.add(link);
        }
    </script>
</head>
<body onload="init()">
</body>
</html>
```

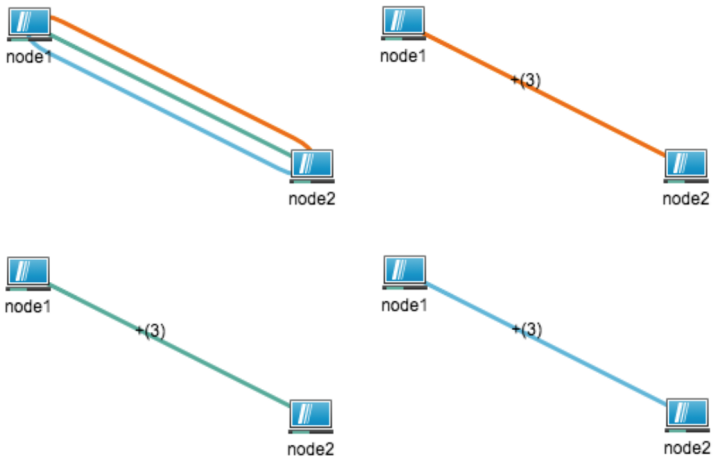


我们经常会遇到自定义代理Link,下面这个示例说明如何自定义代理。

示例：

```
<!DOCTYPE html>
<html>
<head>
    <title>TWaver HTML5 Demo</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initBox();
        }
        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 20, width: 1300, height: 600});
        }

        function initBox() {
            var node1 = new twaver.Node({
                name:'node1',
                location:{x:100,y:100}
            });
            box.add(node1);

            var node2 = new twaver.Node({
                name:'node2',
                location:{x:300,y:200}
            });
            box.add(node2);

            var link1 = new twaver.Link("link1",node1,node2);
            link1.s('link.color',"#ec6c00");
            box.add(link1);

            var link2 = new twaver.Link("link2",node1,node2);
            link2.s('link.color',"#57ab9a");
            box.add(link2);

            var link3 = new twaver.Link("link3",node1,node2);
            link3.s('link.color',"#61b6d8");
            box.add(link3);

        }

        function changeBundleLink(linkID) {
            twaver.Defaults.LINK_BUNDLE_AGENT_FUNCTION = function (links) {
                return box.getDataById(linkID); //在此处修改index
```

```
            }

            box.forEach(function(data){
                if(data instanceof  twaver.Link){
                    var fromAgent = data.getFromAgent();
                    var toAgent = data.getToAgent();
                    _twaver.element.resetBundleLinks(fromAgent,toAgent);
                }
            });
        }

    </script>
</head>
<body onload="init()" style="margin:0;">
<button onclick="changeBundleLink('link1')">SetLink1</button>
<button onclick="changeBundleLink('link2')" >SetLink2</button>
<button onclick="changeBundleLink('link3')" >SetLink3</button>
<div id="main">
</body>
</html>
```



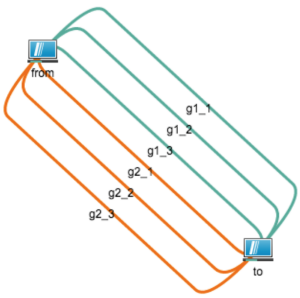
twaver.Link章节我们简单的介绍了连线的绑定与展开，本章将详细介绍分组绑定、自环绑定、绑定与展开以及展开间隙等属性。TWaver定义了下面几种连线绑定相关参数：

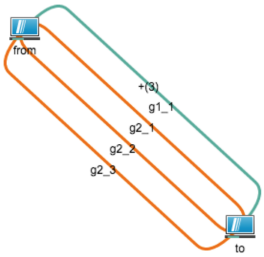
```
"link.bundle.id": 绑定分组编号，同一编号的连线组成一组。
"link.bundle.independent": 分组是否独立，存在多个连线分组时，是否独立绑定
"link.bundle.gap": 连线之间的间隙
"link.bundle.offset": 连线离端点的偏移量
"link.bundle.enable": 是否参与绑定
"link.bundle.expanded": 是否为展开状态,false表示状态为绑定
```

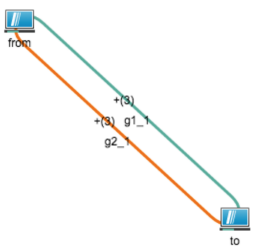
下面的例子中,定义了两组连线,分别设置bundlerID为”1”和”2”。

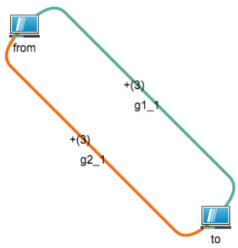
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test Zoom</title>
    <script src="../twaver.js"></script>
    <script>
        var box = new twaver.ElementBox();
        var network = new twaver.canvas.Network(box);

        function init() {
            initNetwork();
            initBox();
        }

        function initNetwork() {
            network.getView().style.backgroundColor = 'rgba(255,255,255,1)';
            document.getElementById('main').innerHTML = "";
            main.appendChild(network.getView());
            network.adjustBounds({
                x: 0,
                y: 0,
                width: 1300,
                height: 600
            });
        }

        function initBox() {
            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                }
            });
            box.add(from);

            var to = new twaver.Node({
                name:'to',
                location:{
                    x:300,
```

```
                    y:300
                }
            });
            box.add(to);
            createLink(from,to,'g1_1',1,"#57ab9a");
            createLink(from,to,'g1_2',1,"#57ab9a");
            createLink(from,to,'g1_3',1,"#57ab9a");
            createLink(from,to,'g2_1',2,"#ec6c00");
            createLink(from,to,'g2_2',2,"#ec6c00");
            createLink(from,to,'g2_3',2,"#ec6c00");

        }

        function createLink(from,to,name,groupID,color,type,groupIndependent,gap,offset,bundleEnable){
            var link = new twaver.Link(from,to);
            link.setName(name);
            if(type){
                link.setStyle('link.type',type);
            }
            if(color){
                link.setStyle('link.color',color);
            }
            if(groupID >= 0){
                link.setStyle('link.bundle.id',groupID);
            }
            if(gap > 0){
                link.setStyle('link.bundle.gap',gap);
            }
            if(offset >0){
                link.setStyle('link.bundle.offset',offset);
            }
            link.setStyle('link.bundle.independent',groupIndependent);
            link.setStyle('link.bundle.enable',bundleEnable);
            box.add(link);
            return link;
        }
    </script>
</head>
<body onload="init()">
<div id='main' style="position:relative;"></div>
</body>
</html>
```









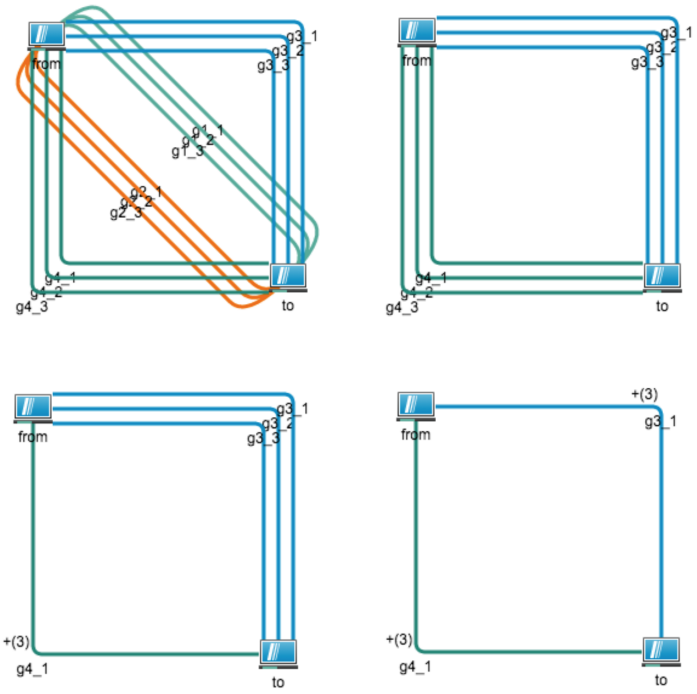
Note:

另外我们也可以控制各group之间的距离，设置样式link.setStyle('link.bundle.group.gap',20)(默认值为0)即可，效果如上图。

下面我们设置两组连线各自独立绑定，并分别设置不同的连线类型。

```
createLink(from,to,'g3_1',1,'#0089c1','orthogonal.H.V');
createLink(from,to,'g3_2',1,'#0089c1','orthogonal.H.V');
createLink(from,to,'g3_3',1,'#0089c1','orthogonal.H.V');
createLink(from,to,'g4_1',2,'#238475','orthogonal.V.H',true);
createLink(from,to,'g4_2',2,'#238475','orthogonal.V.H',true);
createLink(from,to,'g4_3',2,'#238475','orthogonal.V.H',true);
```

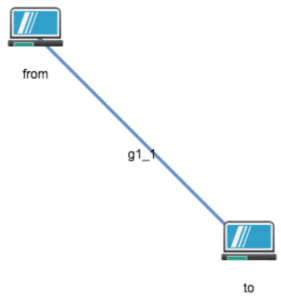
效果如下：



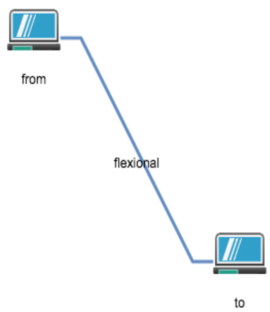
连线用于连接两个节点，可表示为链路，管线等业务对象，TWaver HTML5中的连线类为twaver.Link，由之扩展出来的ShapeLink,这种连线是由一系列控制点决定连线走向的。如果起始和结束点为同一节点，也就是节点A连向节点A,这种情况我们称之为自环。

普通的连线(Link)提供了几种走向类型,主要为三类：

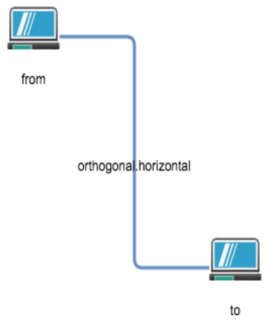
直线：直接连接起始结束节点,常用的样式有：'arc'、'triangle'、'parallel'。



延伸直线:先从端点的水平或者垂直方向延伸，然后再直线连接，常用的样式有: 'flexional', 'flexional.horizontal', 'flexional.vertical'。

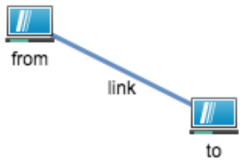
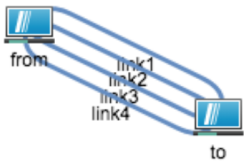
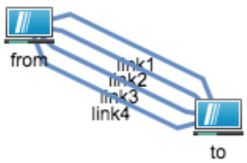
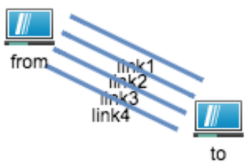
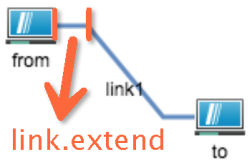
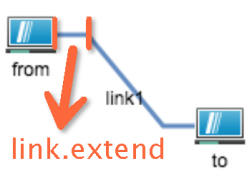


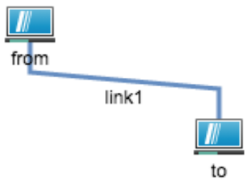
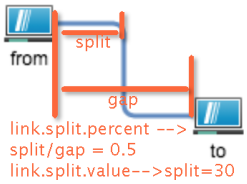
正交直线:这种连线最为丰富，可以通过一个控制点控制正交连线,常用的样式有:'orthogonal', 'orthogonal.vertical', 'orthogonal.H.V','orthogonal.V.H','extend.top','extend.left','extend.bottom','extend.right'。

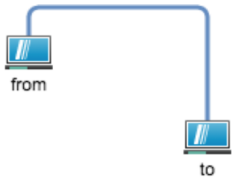


下面我们详细介绍这几种连线类型的使用方法和特点。设置连线类型的方法如下,各种连线类型以及相应的控制参数可参考下表。

```
//方式一
link.setStyle('link.type','orthogonal');
//方式二
link.s('link.type','orthogonal');
```

Link Type	呈现	控制参数
基本类型 直接连接起始结束点,两节点间存在多条连线时, 按连线展开效果又分为下面三中类型:arc,triangle,parallel. 这些类型只有在两节点之间存在多条连线时，呈现有区别		link.from.position 默认值center link.from.xoffset 默认值0 link.from.yoffset 默认值0 link.to.position 默认值center link.to.xoffset 默认值0 link.to.yoffset 默认值0
arc 连线展开时,拐点呈圆弧状		link.bundle.offset 拐点偏移量,默认值20 link.bundle.gap 连线间距,默认值12
triangle 连线展开时,拐点处成直线相连		同上
parallel 连线展开时,呈平行线		同上
flexional 此类连线如右图所示，按方向分为三种： 自动方向：若两节点间X方向间隙大，则取水平方向 水平方向：flexional.horizontal 垂直方向：flexional.vertical		link.from.at.edge 连接到起始节点的边缘 默认值true link.to.at.edge 连接到结束节点的边缘 默认值为true link.extend 展开距离,默认值20
flexional.horizontal		同上

flexional.vertical		同上
正交直线连线类型		
orthogonal 正交直线 按方向分三类： 自动方向：若两节点X方向间隙大，则取水平方向 水平方向：orthogonal.horizontal 垂直方向：orthogonal.vertical	 	link.from.at.edge 连接到起始节点的边缘 默认值true link.to.at.edge 连接到结束节点的边缘 默认值true link.split.by.percent 是否按照百分比劈分，默认为true link.split.percent 劈分点距起始节点的百分比位置 默认值0.5 若按百分比劈分，请设置此属性 link.split.value 劈分点距起始节点的偏移量 默认值20 若按偏移量劈分，请设置此属性
orthogonal.horizontal		同上
orthogonal.vertical		同上
orthogonal.vertical 从起始节点开始，先沿水平方向， 后垂直方向		link.from.at.edge 连接到起始节点的边缘 默认值true link.to.at.edge 连接到结束节点的边缘 默认值为true
vertical.horizontal 从起始节点开始，先沿垂直方向，后水平方向		同上
extend.top 向上扩展		link.extend 扩展量，默认值20

扩展量为劈分点到连线起始结束最顶端的距离		
extend.left 向上扩展 扩展两为劈分点到连线起始结束最顶端的距离		同上
extend.bottom 向底扩展 扩展量为劈分点到连线起始结束最底端的距离		同上
extend.right 向右扩展 扩展两为劈分点到连线起始结束最右端的距离		同上
上面的正交直线类型都可以按照控制点确定走向 如果连线设置了控制点，有限使用控制点来决定劈分点位置		link.control.point 控制点 连线劈分点位置由此控制点决定 连线方向由连线类型决定 自动方向： orthogonal 水平方向： orthogonal.horizontal 垂直方向： orthogonal.vertical

下面我们通过一些示例展示Link更多样式的设置。

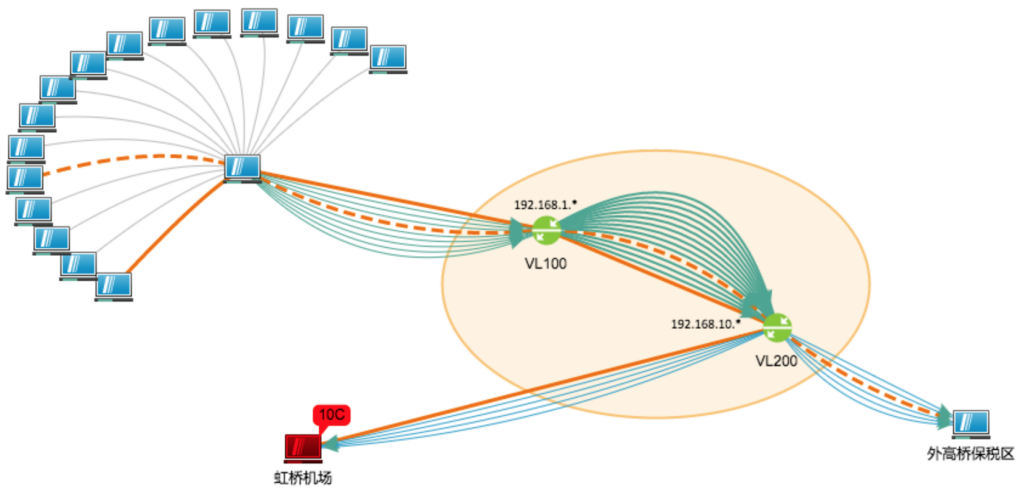
示例一：

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv=Content-Type content="text/html;charset=utf-8">
    <title>TWaver Test</title>
    <script type="text/javascript" src="../lib/twaver.js"></script>
    <script type="text/javascript">

    twaver.Util.registerImage('arrow', {
      w: 8,
      h: 8,
      clip: false,
      origin: { x: 0, y: 1 },
      v: [
      {
        shape: 'line',
        lineWidth: 2,
        lineColor: 'white',
        x1: 5,
        y1: -5,
        x2: 1,
        y2: -1,
      },
      {
        shape: 'path',
          data: [ { x: 6, y: 0 }, { x: 0, y: 0 }, { x: 0, y: -6 },  ],
        lineColor: 'white',
        fill: 'white',
      },
      ]
    });
```

```
    twaver.Util.registerImage('port', {
      w: 35,
      h: 35,
      clip: false,
      origin: { x: 0.5, y: 0.5 },
      v: [
      {
        shape: 'circle',
        cx: 0,
        cy: 0,
        r: 12,
        fill: '#86C63A',
      },
      {
        shape: 'line',
        lineWidth: 2,
        lineColor: 'white',
        x1: -12,
        y1: 0,
        x2: 12,
        y2: 0,
      },
      {
        shape: 'vector',
        name: 'arrow',
        x: 7,
        y: -7
      },
      {
        shape: 'vector',
        name: 'arrow',
        x: -4,
        y: 4
      },
      ]
    });

    MyLink = function(){
            MyLink.superClass.constructor.apply(this, arguments);
            this.setStyle('shadow.blur',10);
            this.setStyle('shadow.xoffset',6);
            this.setStyle('shadow.yoffset',6);
            this.setStyle('link.width',1);
            this.setStyle('link.color','#4DA492');
            this.setStyle('outer.width', 0);
      this.setStyle('arrow.from.color', '#4DA492');
      this.setStyle('arrow.to.color', '#4DA492');
      this.setStyle('arrow.to', true);
      this.setStyle('curve_type','bundle');
        }
        twaver.Util.ext('MyLink', twaver.Link,{
            getVectorUIClass : function(){
                return MyLinkUI;
            }
        });

        MyLinkUI = function(){
            MyLinkUI.superClass.constructor.apply(this, arguments);
        }
        twaver.Util.ext('MyLinkUI', twaver.vector.LinkUI, {
            getLinkPoints: function () {
        MyLinkUI.superClass.getLinkPoints.call(this);

        var f = this.getFromPoint();
        var t = this.getToPoint();
        var angle=Math.atan((t.y-f.y)/(t.x-f.x));

        var points = new twaver.List();
        points.add(f);
        points.add(t);

        if(this.getElement().getStyle('curve_type')=='bundle'){
          var index=this.getElement().getBundleIndex();
          this._lineLength = _twaver.math.calculateLineLength(points);
          var offset = 10*index;
          var offsetX=offset*Math.sin(angle);
          var offsetY=offset*Math.cos(angle);
          var backward=this.getElement().getStyle('backward');
          var factor=1;
          if(backward){
            factor=-1;
          }
          var m = {
            x: (f.x+t.x)/2 + offsetX * factor,
            y: (f.y+t.y)/2 - offsetY * factor,
          }
          var cps = new twaver.List();
          cps.add(m);
          cps.add(t);
          points.removeAt(1);
          points.add(cps);
        }

        if(this.getElement().getStyle('curve_type')=='float'){
          var m = {
            x: (f.x+t.x)/2 + 20,
            y: (f.y+t.y)/2 - 30
          }
          var cps = new twaver.List();
          cps.add(m);
          cps.add(t);
          points.removeAt(1);
```

```
          points.add(cps);
        }

        this._linkPoints = points;

                return this._linkPoints;
            }
        });

        function load(){
            this.box = new twaver.ElementBox();
            this.network = new twaver.vector.Network(this.box);

            document.getElementById('main').appendChild(network.getView());
            network.adjustBounds({x:0,y:0,width:document.documentElement.clientWidth,height:document.documentElement.clientHeight

      var group1=new twaver.Group();
      group1.setLocation(500,500);
      box.add(group1);

      var node1=new twaver.Node();
      node1.setLocation(450,350);
      box.add(node1);

      var count=16;
      for(var i=0;i<count;i++){
        var node=new twaver.Node();
        var angle=Math.PI*1.1/count*i+Math.PI*0.7;
        var r=150;
        var x=node1.getLocation().x+r*1.2*Math.cos(angle);
        var y=node1.getLocation().y+r*0.8*Math.sin(angle);
        node.setLocation(x,y);
        box.add(node);

        var link=new MyLink(node1,node);
        link.setStyle('link.color','#BBBBBB');
        if(i==0 || i==4){
          link.setStyle('link.width',3);
          link.setStyle('link.color','#EC6C00');
          if(i==4){
            link.setStyle('link.pattern', [10, 5]);
          }
        }
        link.setStyle('arrow.to',false);
        link.setStyle('curve_type','float');
        box.add(link);
      }

      var node2=new twaver.Node();
      node2.setLocation(700,400);
      node2.setImage('port');
      node2.setName('VL100');
      node2.setName2('192.168.1.*');
      node2.setStyle('label2.yoffset',5);
      node2.setStyle('label2.font','11px Calibri');
      node2.setParent(group1);
      box.add(node2);

      var count=10;
      for(var i=0;i<count;i++){
        var link=new MyLink(node1,node2);
        link.setStyle('backward',true);
        if(i==0 || i==4){
          link.setStyle('link.width',3);
          link.setStyle('link.color','#EC6C00');
          if(i==4){
            link.setStyle('link.pattern', [10, 5]);
          }
        }
        box.add(link);
      }

      var node3=new twaver.Node();
      node3.setLocation(890,480);
      node3.setParent(group1);
      node3.setImage('port');
      node3.setName('VL200');
      node3.setName2('192.168.10.*');
      node3.setStyle('label2.xoffset',-58);
      node3.setStyle('label2.yoffset',23);
      node3.setStyle('label2.font','11px Calibri');
      box.add(node3);

      var count=15;
      for(var i=0;i<count;i++){
        var link=new MyLink(node2,node3);
        link.setStyle('link.width',2);
        if(i==0 || i==4){
          link.setStyle('link.width',3);
          link.setStyle('link.color','#EC6C00');
          if(i==4){
            link.setStyle('link.pattern', [10, 5]);
          }
        }
        box.add(link);
      }

      var node4=new twaver.Node();
      node4.setLocation(1050,560);
      node4.setName('外高桥保税区');
      node4.setStyle('label.font','12px "Microsoft Yahei"');
      box.add(node4);
```

```
237
238       var count=6;
239       for(var i=0;i<count;i++){
240         var link=new MyLink(node3,node4);
241         link.setStyle('backward',true);
242         link.setStyle('link.width',1);
243         link.setStyle('link.color','#309FC9');
244         if(i==2){
245           link.setStyle('link.width',3);
246           link.setStyle('link.color','#EC6C00');
247           link.setStyle('link.pattern', [10, 5]);
248         }
249         box.add(link);
250       }
251
252       var node5=new twaver.Node();
253       node5.setLocation(500,580);
254       node5.setName('虹桥机场');
255       node5.setStyle('label.font','12px "Microsoft Yahei"');
256       node5.getAlarmState().setNewAlarmCount(twaver.AlarmSeverity.CRITICAL,10);
257       box.add(node5);
258
259       var count=5;
260       for(var i=0;i<count;i++){
261         var link=new MyLink(node3,node5);
262         link.setStyle('backward',true);
263         link.setStyle('link.width',1);
264         link.setStyle('link.color','#309FC9');
265         if(i==0){
266           link.setStyle('link.width',3);
267           link.setStyle('link.color','#EC6C00');
268         }
269         box.add(link);
270       }
271
272       group1.setExpanded(true);
273       group1.s('group.fill.color','#FEF3E2');
274             group1.s('group.alpha',0.1);
275             group1.s('group.shape','oval');
276       group1.s('group.outline.color','#F8C374');
277       group1.s('group.outline.width','2');
278         }
279
280     </script>
281 </head>
282 <body onload="load()">
283     <div id='main' style = 'width:100%;height:100%'>
284     </div>
285 </body>
286 </html>
```



形状连线(ShapeLink)

ShapeLink继承于Link,使用方法和ShapeNode类似，添加删除控制点相关方法：

```
1 addPoint:function(point,index)
2 setPoint:function(point,index)
3 getPoints:function()
4 setPoints:function(value)
5 removePoint:function(point)
6 removeAt:function(index)
```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Test Zoom</title>
6     <script src="../twaver.js"></script>
7     <script>
```

```
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
//            twaver.Styles.setStyle('link.color',"#57ab9a");
        }

        function initDataBox() {
            var nodeA = new twaver.Node();
            nodeA.setLocation(100, 100);
            var nodeB = new twaver.Node();
            nodeB.setLocation(200, 200);
            var link = new twaver.ShapeLink(nodeA, nodeB);

            box.add(nodeA);
            box.add(nodeB);
            box.add(link);

            link.addPoint({
                x : 150,
                y : 150
            });

            var list = new twaver.List();
            list.add({
                x : 150,
                y : 300
            });
            list.add({
                x : 170,
                y : 170
            });
            link.addPoint(list);
        }
    </script>
</head>
<body onload="init()">
</body>
</html>
```

跟随者(Follower)

跟随者跟随宿主移动而移动，具有依附的功能，通常用于设备面板，如端口依附在板卡上。

Follower常用的方法如下：

```
//设置宿主
get/setHost: function(host)
//是否依附在某个节点上
isHostOn: function(node)
//是否是相互依附关系
isLoopedHostOn: function(node)
```

示例：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test Zoom</title>
    <script src="../twaver.js"></script>
    <script>
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
```

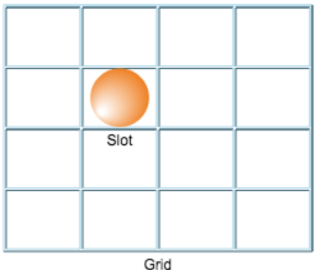
```
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
//            twaver.Styles.setStyle('link.color',"#57ab9a");
        }

        function initDataBox() {
            var Host = new twaver.Node({
                name:'host',
                location:{
                    x:100,
                    y:200
                }
            });
            box.add(Host);

            var Follower = new twaver.Follower({
                name:'Follower',
                location:{
                    x:300,
                    y:300
                }
            });
            box.add(Follower);

            Follower.setHost(Host);


        }
    </script>
</head>
<body onload="init()">
</body>
</html>
```

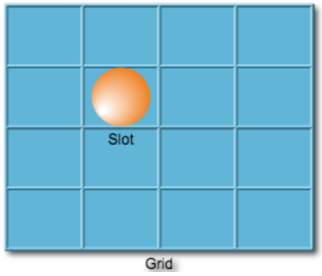
网格(Grid)

twaver.Grid网元在拓扑图中表现为网格，实现了类似于java Swing的TabelLayout,通过指定行列号以及行列跨度实现网元的定位布局。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test Zoom</title>
    <script src="../twaver.js"></script>
    <script>
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }

        function initDataBox() {
            var grid = new twaver.Grid();
            grid.setName("Grid");
            grid.s('grid.border', 1);
            grid.s('grid.deep', 1);
            grid.s('grid.column.count', 4);
            grid.s('grid.row.count', 4);
            grid.setSize(250, 200);
            grid.setLocation(20, 30);
            grid.s("grid.fill", true);
            grid.s("grid.fill.color", '#61b6d8');
            box.add(grid);

            var slot = new twaver.Grid();
            slot.setName("Slot");
            slot.s('body.type', 'vector');
            slot.s('vector.shape', 'circle');
            slot.s('vector.gradient', 'radial.southwest');
            slot.s('vector.fill.color', '#ec6c00');
            slot.setHost(grid);
            slot.setParent(grid);
            slot.s('follower.row.index', 1);
            slot.s('follower.column.index', 1);
            box.add(slot);
```

```
    </script>
</head>
<body onload="init()">
</body>
</html>
```





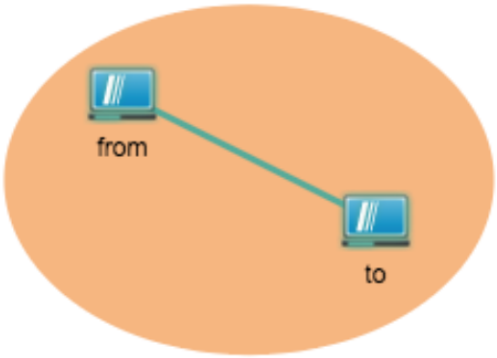
网元组(Group)

通常网元组中包含多个孩子网元,网元组展开时,同时显示网元组和其下的孩子网元,合并时显示网元组网元图标。

下面是网元组嵌套以及展开合并的效果图，默认双击展开或合并网元组。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Test Zoom</title>
    <script src="../twaver.js"></script>
    <script>
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            registerImage();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }

        function registerImage() {
            //register shadow
            twaver.Util.registerImage('shadow', {
                w: 37,
                h: 29,
                shadowOffsetX: 0,
                shadowOffsetY: 0,
                shadowBlur: 5,
                shadowColor: '#57ab9a',
                v: [
                    {
                        shape: 'vector',
                        name: 'node_image',
                        x: 0,
                        y: 0
                    }
                ]
            });

        }

        function initDataBox() {
            var group = new twaver.Group();
            group.s('group.fill.color','#ec6c00');
            group.s('whole.alpha',0.5);
            group.s('group.shape','oval');
            box.add(group);

            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                },
                image:'shadow'
            });
            box.add(from);

            var to = new twaver.Node({
                name:'to',
                location:{
                    x:220,
                    y:160
                },
                image:'shadow'

            });
            box.add(to);

```

```
            var link = new twaver.Link(from,to);
            link.s('link.color','#57ab9a');
            box.add(link);

            group.addChild(from);
            group.addChild(to);
        }
    </script>
</head>
<body onload="init()">
</body>
</html>
```



Note:

Group默认情况下是闭合的，如果想默认情况下为展开，可以设置group.setExpanded(true);

子网(SubNetwork)

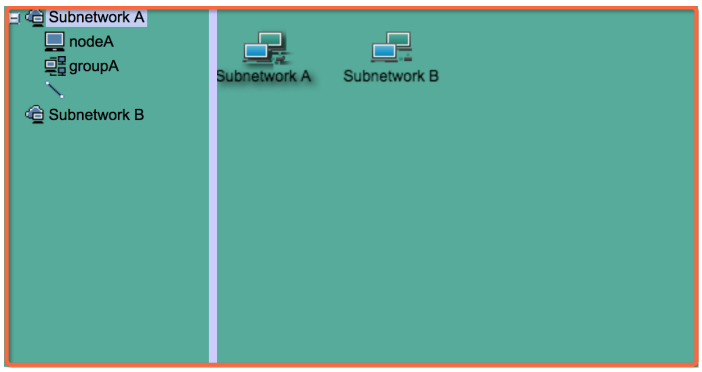
子网接口,表示大型网络中的一小部分网络，TWaver HTML5中拓扑图可以切换子网,呈现的即是当前子网中的网元。子网接口的默认实现类是twaver.SubNetwork,拓扑图中默认双击进入子网,双击背景返回上层子网,当前子网为Null时表示是最顶层子网。

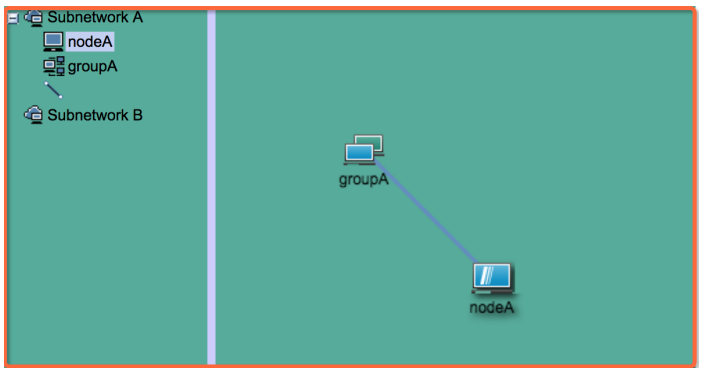
Vector中切换子网的方法：

```
//设置当前子网
get/setCurrentSubnetwork: function(subnetwork,animate)
//返回上级子网
upSubNetwork: function(animate)
```

```
<!DOCTYPE html>
<html>
<head>
    <title>TWaver HTML5 Demo</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);
        var tree = new twaver.controls.Tree(box);
        var table = new twaver.controls.Table(box);

        function init() {
            var tablePane = new twaver.controls.TablePane(table);
            var rightSplit = new twaver.controls.SplitPane(network, tablePane, 'vertical', 0.7);
            var mainSplitPane = new twaver.controls.SplitPane(tree, rightSplit, 'horizontal', 0.3);

            var networkDom = mainSplitPane.getView();
            networkDom.style.width = "100%";
            networkDom.style.height = "100%";
            document.body.appendChild(networkDom);
            network.getView().style.backgroundColor = "#f3f3f3";
            network.getView().style.cursor = "hand";
            window.onresize = function () {
                mainSplitPane.invalidate();
            }
            initDataBox();
        }

        function initDataBox() {
            var subNetworkA = new twaver.SubNetwork();
            subNetworkA.setName("Subnetwork A");
            subNetworkA.setLocation(20, 20);
            box.add(subNetworkA);

            var groupA = new twaver.Group({
                name: 'groupA',
                location: {x: 100, y: 100}
            });
            box.add(groupA);
```

```
40
41              var nodeA = new twaver.Node({
42                  name: 'nodeA',
43                  location: {x: 200, y: 200}
44              });
45              box.add(nodeA);
46
47              var link = new twaver.Link(groupA,nodeA);
48              box.add(link);
49
50              subNetworkA.addChild(nodeA);
51              subNetworkA.addChild(groupA);
52              subNetworkA.addChild(link);
53
54              var subNetworkB = new twaver.SubNetwork();
55              subNetworkB.setName("Subnetwork B");
56              subNetworkB.setLocation(120, 20);
57              box.add(subNetworkB);
58
59          }
60
61      </script>
62  </head>
63  <body onload="init()" style="margin:0;">
64  </body>
65  </html>
```





总线(Bus)

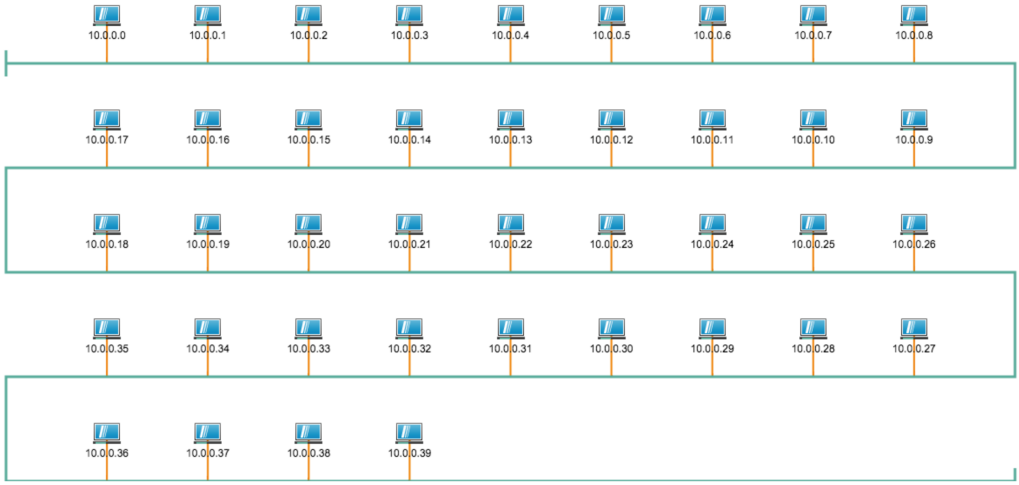
总线(Bus)是计算机各种功能部件之间传送信息的公共通信干线,TWaver可以使用Bus网元进行模拟,twaver.Bus继承于twaver.ShapeNode,使用方法可以参考下面的示例。

示例:

```
1   <!DOCTYPE html >
2   <head>
3       <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
4       <script src="../lib/twaver.js"></script>
5   </head>
6   <script type="text/javascript">
7       var box = new twaver.ElementBox();
8       var network = new twaver.vector.Network(box);
9       var bus = new twaver.Bus();
10      var busCount = 40;
11
12      function init() {
13          initNetwork();
14          initListener();
15          initDataBox();
16      }
17
18      function initNetwork() {
19          document.body.appendChild(network.getView());
20          network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
21          twaver.Styles.setStyle('group.expanded', true);
22      }
23      function initListener() {
24
25
```

```
    function initDataBox() {
        bus.setStyle('vector.outline.width', 3);
        bus.setStyle('vector.outline.color', twaver.Colors.green_light);
        bus.setStyle('bus.style', 'south');
        bus.setLocation(500,500);
        box.add(bus);


        for (var i = 0; i < busCount; i++) {
            var id = "10.0.0." + i;
            var node = new twaver.Node(id);
            node.setName(id);
            box.add(node);

            var link = new twaver.Link(node, bus);
            link.setStyle('link.width', 2);
            link.setStyle('link.color', twaver.Colors.orange_light);
            box.add(link);
        }
        refreshBus();

    }

    function refreshBus() {
        var lastWidth = document.documentElement.clientWidth;
        var margin = 40;
        var wGap = 110;
        var hGap = 120;

        var zoom = network.getZoom();
        var width = Math.max(margin * 2, lastWidth / zoom - margin * 2);
        var columnCount = Math.max(3, Math.floor(width / wGap));
        wGap = width / columnCount;
        var rowCount = Math.floor(busCount / (columnCount - 1));
        if (rowCount * (columnCount - 1) < busCount) {
            rowCount++;
        }

        bus.getPoints().clear();

        var count = 0;
        var x = 0;
        var y = 0;

        bus.getPoints().add({
            x: margin,
            y: hGap + 15
        });
        bus.getPoints().add({
            x: margin,
            y: hGap - 15
        });
        for (var i = 0; i < rowCount; i++) {
            y = hGap * (i + 1);
            if (i % 2 == 0) {
                bus.getPoints().add({
                    x: margin,
                    y: y
                });
                bus.getPoints().add({
                    x: margin + width,
                    y: y
                });
            } else {
                bus.getPoints().add({
                    x: margin + width,
                    y: y
                });
                bus.getPoints().add({
                    x: margin,
                    y: y
                });
            }
            y -= hGap / 2 - 5;
            for (var k = 1; k < columnCount; k++) {
                var node = box.getDataById("10.0.0." + count++);
                if (node == null) {
                    continue;
                }
                if (i % 2 == 0) {
                    x = margin + k * wGap;
                } else {
                    x = margin + width - k * wGap;
                }
                node.setCenterLocation(x, y);
            }
        }
        var point = bus.getPoints().get(bus.getPoints().size() - 1);
        bus.getPoints().add({
            x: point.x,
            y: point.y + 15
        });
        bus.getPoints().add({
            x: point.x,
            y: point.y - 15
        });
        bus.firePointsChange();
    }

</script>
<body onload="init();">
</body>
```

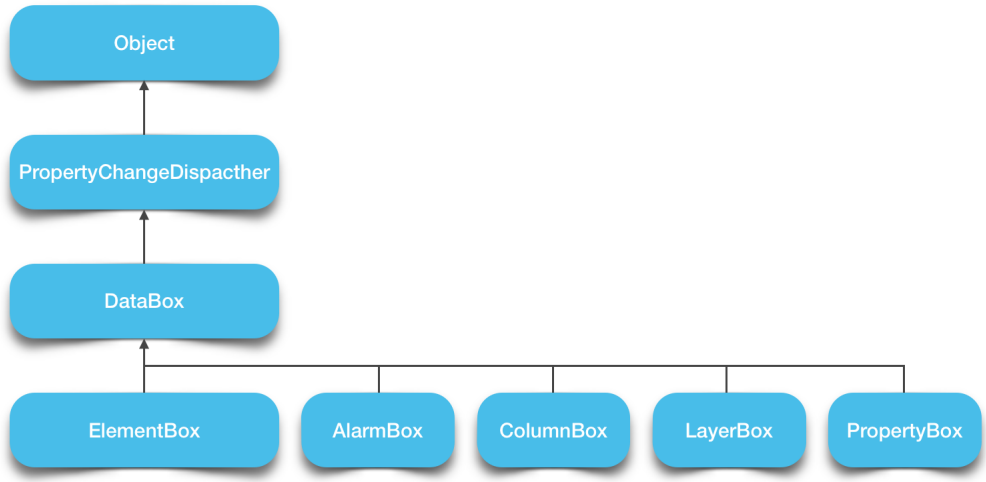
```
129 </html>
```


10.0.0.0
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4
10.0.0.5
10.0.0.6
10.0.0.7
10.0.0.8
10.0.0.17
10.0.0.16
10.0.0.15
10.0.0.14
10.0.0.13
10.0.0.12
10.0.0.11
10.0.0.10
10.0.0.9
10.0.0.18
10.0.0.19
10.0.0.20
10.0.0.21
10.0.0.22
10.0.0.23
10.0.0.24
10.0.0.25
10.0.0.26
10.0.0.35
10.0.0.34
10.0.0.33
10.0.0.32
10.0.0.31
10.0.0.30
10.0.0.29
10.0.0.28
10.0.0.27
10.0.0.36
10.0.0.37
10.0.0.38
10.0.0.39

概述

在TWaver中,所有的网元对象均可以加入到DataBox中进行管理。DataBox是一个数据管理容器，用于管理所有的预定义网元对象，当添加、删除、移动网元或者更改网元属性的时候，DataBox均可以监听到。TWaver DataBox和后台数据也有很好的接口,它可以通过数据流的方式和后台相连，也就是说可以通过XML文件或者Json文件进行更改DataBox中的网元。DataBox在数据承载上也有一个很好的设计，它的优势在于很方便的管理Network上庞大数据而仅占用很少的内存，很少的加载时间。

twaver.DataBox是TWaver中所有Box的父类，即所有的Box均继承于DataBox。主要包括ElementBox、LayerBox、AlarmBox、ColumnBox、PropertyBox、TabBox等。



数据容器(DataBox)

DataBox是用户经常操作的一个类，含有的方法也非常的丰富，用户可以参考API文档深入学习DataBox容器。在这我们介绍一些常用的方法。

增删查改

```
/**
 *功能:往数据容器中添加一个数据
 *data:数据对象
 *index:添加的数据次序,为空时，就将这个数据添加在最后的位置
 */
add:function(data,index)

/**
 *功能:从数据容器中删除某个数据
 *data:删除的数据对象
 */
remove:function(data)

/**
 *功能:根据Id删除网元数据
 */
removeById: function (id)

/**
 *功能:清空DataBox
 */
clear: function ()

/**
 *功能:获取数据容器中某个序号上的数据
 *index:Number,数据的序号
 *Returns:twaver.Data,数据对象
 */
getDataAt:function(index)

/**
 *功能:根据网元ID值获取数据
 */
getDataById:function (id)


/**
 *功能:获取指定数据的所有兄弟数据
 *data: 指定网元
 */
getSiblings: function (data)

```

```
 *功能:获取指定数据在兄弟数据中的序号
 *data:指定网元
 */
getSiblingIndex: function (data)

/**
 *功能:获取数据容器中所有根下的数据。根下的数据指的是没有父亲的数据
 *returns:twaver.List
 */
getRoots: function ()

/**
 *功能:获取DataBox中所有网元
 *returns:twaver.List
 */
 getDatas: function ()

/**
 *功能:获取DataBox中随机一个网元
 *type:可以设置随机网元的类型,如:twaver.Node,twaver.Link等
 */
getRandomData: function(type)
```

遍历数据容器

```
/**
 *功能:遍历数据容器中的所有数据
 *callback:回调函数
 *scope:作用于范围,默认为window
 */
forEach: function (callback, scope)

/**
 *功能:反序遍历数据容器中的所有数据
 *callback:回调函数
 *scope:作用于范围,默认为window
 */
forEachReverse: function (f, scope)
```

包含网元判断

```
/**
 *功能:根据Id判断当前数据是否存在于数据容器中
 *returns: Boolean
 */
containsById: function (id)

/**
 *功能:判断当前数据是否存在于数据容器中
 *returns: Boolean
 */
contains: function (data)
```

数据序列化

数据序列化 (Serialization)将对象的状态信息转换为可以存储或传输的形式的过程。在序列化期间，对象将其当前状态写入到临时或持久性存储区。以后，可以通过从存储区中读取或反序列化对象的状态，重新创建该对象。TWaver支持数据序列化和反序列化，便于数据的传输、以及数据的重建。但是TWaver并没有设计任何加密处理，为了安全起见，数据可以适当的进行加密处理，确保数据传输的安全。

TWaver支持序列化格式有xml和json,使用方法如下:

Xml序列化与反序列化

```
//将DataBox序列化成xml数据
var datas = new twaver.XmlSerializer(box).serialize();
//反序列化，并将数据与DataBox建立关系
new twaver.XmlSerializer(box).deserialize(datas);
```

示例:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="GB2312">
    <title>Hello TWaver Full</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var network = new twaver.network.Network();
        var box = network.getElementBox();
        var alarmBox = box.getAlarmBox();

        function loadJSON(path)
        {
            var result;
            var xhr = new XMLHttpRequest();
            xhr.onreadystatechange = function()
            {
                if (xhr.readyState === 4) {
                    if (xhr.status === 200) {
                        //console.log(xhr.responseText);
```

```
                        new twaver.XmlSerializer(box).deserialize(xhr.responseText);
                    }
                }
            };
            xhr.open("GET", path, true);
            xhr.send();
        }

        function init() {
            var networkDom = network.getView();
            networkDom.style.width = "100%";
            networkDom.style.height = "100%";
            document.body.appendChild(networkDom);
            loadJSON("data.xml");
        }
    </script>
</head>
<body onload="init()" style="margin:0;">
</body>
</html>
```

Json序列化与反序列化

```
//将DataBox序列化成Json数据
 var datas = new twaver.JsonSerializer(box).serialize();
//反序列化，并将数据与DataBox建立关系
 new twaver.JsonSerializer(box).deserialize(datas);
```

示例:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="pragma" content="no-cache">
    <title></title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">

        var box = new twaver.ElementBox();
        var network;
        var datas;

        function init() {
            initCanvas();
            initBox();
        }

        function initCanvas() {
            network = new twaver.canvas.Network(box);
            document.body.appendChild(network.getView());
            network.getView().style.background = '#E9E9E9';
            network.adjustBounds({x: 0, y: 30, width: 1300, height: 600});
        }

        function initBox() {
            var from = new twaver.Node();
            from.setName('From');
            from.setLocation(Math.random() * 1000, Math.random() * 500);
            box.add(from);

            for (var i = 0; i < 20; i++) {
                var to = new twaver.Node();
                to.setName('To');
                to.setLocation(Math.random() * 1200, Math.random() * 600);
                box.add(to);

                var link = new twaver.Link(from, to);
                link.setName('Link');
                box.add(link);
            }
            datas = new twaver.JsonSerializer(box).serialize();
        }

        function xmlRead() {
            box.clear();
            if(datas) {
                new twaver.JsonSerializer(box).deserialize(datas);
            }
            box.forEach(function(data){
                if(data instanceof twaver.Link){
                    data.setStyle('whole.alpha',0);
                }
            })
        }
    </script>
</head>
<body onload="init()">
<button onclick="xmlRead()">
    xmlRead
</button>
</body>
</html>
```

用户类型数据序列化

TWaver默认并不是任何数据类型都会序列化出去的,如当用户设置了用户属性,然后序列化却发现序列化的结果中没有值,或者序列化之后，再反序列化所得到的结果不一样，这是为什么呢？原来TWaver为了实现高效率、通用性、可插拔特性,需要先对用户属性的序列化类型进行注册，注册方式如下：

```
twaver.SerializationSettings.setClientType('text','string');
```

这样注册之后，用户设置node.setClient('text','node')就也可以序列化出去了。

网元管理容器(ElementBox)

TWaver的数据容器不仅包含元素集合,还管理着元素的层次关系,以及选中模型。当容器中的元素变化都会派发出相应的变化事件，如元素增减变化派发DataBoxChangeEvent，元素属性变化派发PropertyChangeEvent,元素层次变化派发出HiberarchyChangeEvent,元素选中变化由SelectionModel管理，在"选中模型"章节将详细介绍。

变化事件	触发原因	捕获事件对象	监听方法

容器属性变化
(PropertyChange)
	databox.setName('name');
	databox.setClient(propertyName,value);
	databox.setStyle(styleName,value)
		(1).属性名称(property)
		(2).属性的新值(newValue)
		(3).属性的原值(oldValue)
		(4).发生变化的对象(source)。
			//添加databox属性监听事件
			databox.addPropertyChangeListener(function(e){});

容器元素变化
(DataBoxChange)
	databox.add(node);
	databox.remove(node);
	databox.clear();
		(1).kind:事件的类型(add、remove、clear)
		(2).data:事件发生的对象
		(3).datas:目标对象是一个集合
			//添加DataBoxChange
			databox.addDataBoxChangeListener(function(e){});
			//

元素属性变化
(DataPropertyChange)
	element.setName('name');
	element.setClient(propertyName,value);
	element.setStyle(styleName,value)
		元素的属性变化通过数据容器转发，避免对每个元素添加属性监听器
		(1).属性名称(property)
		(2).属性的新值(newValue)
		(3).属性的原值(oldValue)
		(4).发生变化的对象(source)。
			//data property change listener
			databox.addDataPropertyChangeListener(function(e){})

数据层次变化
(HiberarchyChange)
	databox.moveToTop(node);
	databox.moveToBottom(node)
		(1).原始次序位置(oldIndex)
		(2).新的次序位置(newIndex)
			//hiberarchy change listener
			databox.addHierarchyChangeListener(function(e){});

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <title>Alarm</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initListener();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }


        function initDataBox() {
            box.setName('databox');

            var node1 = new twaver.Node({
                name: 'node1',
                location: {
                    x: 300,
                    y: 200
                }
            });
            box.add(node1);

            var node2 = new twaver.Node({
                name: 'node2',
                location: {
                    x: 200,
                    y: 200
                }
            });
            box.add(node2);

            box.moveToTop(node2);
            box.clear();
        }

        function initListener() {
            box.addPropertyChangeListener(function (e) {
```

```
                console.log('PropertyChange:');
                console.log(e);
            });

            box.addDataBoxChangeListener(function (e) {
                console.log('DataBoxChange:');
                console.log(e);
            });

            box.addDataPropertyChangeListener(function (e) {
                console.log('DataPropertyChange:');
                console.log(e);
            });

            box.addHierarchyChangeListener(function (e) {
                console.log('HierarchyChange:');
                console.log(e);
            })
        }
    </script>

</head>
<body onload="init()">
</body>
</html>
```

```
PropertyChange:
▶ Object {property: "name", oldValue: "ElementBox", newValue: "databox", source: twaver.ElementBox}
DataBoxChange:
▶ Object {kind: "add", data: $Node}
DataBoxChange:
▶ Object {kind: "add", data: $Node}
HierarchyChange:
▶ Object {data: $Node, oldIndex: 1, newIndex: 0}
DataBoxChange:
▶ Object {kind: "clear", datas: $List}
```

```
PropertyChange:
▼ Object {property: "name", oldValue: "ElementBox", newValue: "databox", source: twaver.ElementBox}
    newValue: "databox"
    oldValue: "ElementBox"
    property: "name"
  ▶ source: twaver.ElementBox
  ▶ __proto__: Object
DataBoxChange:
▼ Object {kind: "add", data: $Node}
  ▶ data: $Node
    kind: "add"
  ▶ __proto__: Object
DataBoxChange:
▶ Object {kind: "add", data: $Node}
HierarchyChange:
▼ Object {data: $Node, oldIndex: 1, newIndex: 0}
  ▶ data: $Node
    newIndex: 0
    oldIndex: 1
  ▶ __proto__: Object
DataBoxChange:
▼ Object {kind: "clear", datas: $List}
  ▶ datas: $List
    kind: "clear"
  ▶ __proto__: Object
```

Note:

在发生DataPropertyChange事件的时候，欲想做一些处理，可以重新实现databox#onDataPropertyChanged方法。函数原型为:onDataPropertyChanged:function(data,e)。

层管理容器(LayerBox)

LayerBox的主要作用是用来管理Layer,得到LayerBox的方式有两种，一种是直接通过var layerBox = databox.getLayerBox();一种是创建:var layerBox = new twaver.LayerBox(box);

```
var layerBox = box.getLayerBox();
var layer1 = new twaver.Layer('unmovable', 'unmovable layer');
layer1.setMovable(false);
var layer2 = new twaver.Layer('uneditable', 'uneditable layer');
layer2.setEditable(false);
var layer3 = new twaver.Layer('unvisible', 'unvisible Layer');
layer3.setVisible(false);
layerBox.add(layer1);
layerBox.add(layer2);
layerBox.add(layer3);
```

Note:

LayerBox继承于DataBox,使用方法类似,可参考API文档进行深入学习。

告警管理容器(AlarmBox)

AlarmBox自然是用来管理告警(Alarm)的,获取AlarmBox的方法也有两种,一种是var alarmBox = databox.getAlarmBox();另外一种是:var alarmBox = new AlarmBox(box);

```
var alarm = new twaver.Alarm(alarmID, elementID, alarmSeverity);
alarmBox.add(alarm);
```

列管理容器(ColumnBox)

列管理容器主要用于Table表格中，管理表格中的列(twaver.Column)对象。

```
var table = new twaver.controls.Table(box);
...
var column = new twaver.Column(name);
column.setName(name);
column.setPropertyName(propetyName);
column.setPropertyType(propertyType);
if (valueType) {
    column.setValueType(valueType);
}
table.getColumnBox().add(column);
```

属性管理容器(PropertyBox)

属性管理容器主要用来管理属性(twaver.Property)对象。

```
var sheet = new twaver.controls.PropertySheet(box);
...
var sheetBox = sheet.getPropertyBox();
var property = new twaver.Property();
property.setCategoryName(category);
    if (!name) {
        name = demo.Util._getNameFromPropertyName(propertyName);
    }
    property.setName(name);
    property.setEditable(true);
    property.setPropertyType(proprtyType);
    property.setPropertyName(propertyName);
    sheetBox.add(property);
```

页管理容器(TabBox)

页管理容器主要用来管理Tab(twaver.Tab)页。

```
var tablePane = new twaver.controls.TablePane(table);
...
var tab = new twaver.Tab(name);
tab.setName(name);
tab.setView(view);
tabPane.getTabBox().add(tab);
```

选中模型(SelectionModel)

选中模型服务于DataBox,用于管理元素选中信息、元素的选中、清除选中都通过SelectionModel实现。那么如何操作SelectionModel呢？

步骤：

(1).构建选中模型

默认情况下DataBox自带一个选中模型,获取方式:databox.getSelectionModel()。视图对象中的默认就是data box的选中模型,可通过view.getSelectionModel()获取,如：network.getSelectionModel()或者tree.getSelectionModel();如果用户想设置自己的选中模型,可通过下面方式设置。

```
//创建模型对象
var myselectionModel = new waver.SelectionModel(databox);
//构造参数传入需要绑定的databox即可，设置到databox或者视图组件
databox.setSelectionModel(mySelectionModel);
network.setSelectionModel(mySelectionModel);
tree.setSelectionModel(mySelectionModel);
...
```

(2).使用SelectionModel功能

```
//追加选中元素,传入参数可以是单个元素,也可以是元素集合
appendSelection:function(datas)
//设置选中元素,与追加选中不同,此方法会清除原始选中状态
setSelection():function(datas)
//选中databox中所有元素
selectAll:function()
//取消元素选中状态,传入参数可以是单个元素,也可以是元素集合
removeSelection:function(datas)
//清除所有元素的选中状态
clearSelection:function()
//获取选中元素集合,注意此方法返回的是SelectionModel内部选中元素集合对象的引用,直接对这个集合操作会影响到选中模型,所以不要对这个集合做修改操作
```

```
getSelection:function()
/**获取当前选中元素集合,注意此方法与上个函数有区别,此方法返回的是心构建的集合类,而不是SelectionModel中原始的选中元
  *素集合对象引用
  *matchFunction:匹配函数，传入IData,返回true或者false,false表示排除
  */
toSelection:function(matchFunction,scope)
//选中数量
size:function()
//元素是否选中
contains:function(data)
//选中集合中的最后一个元素
getLastData:function()
//选中集合中的第一个元素
getFirstData:function()
//是否允许选中
isSelectable:function(data)
```

选中模型管理元素的选中状态,当选中状态发生变化时，派发相应的选中变化事件，如调用追加元素选中函数会派发类型为”append”的选种变化事件。

```
selectionModel.addSelectionChangeListener(function(e){
    console.log("Kind:"+e.kind+',datas:'+e.datas.toString());
});
```

事件对象中包含两个属性’kind’,’datas’，分别代表选中事件类型，事件数据，其中选中变化事件有五种子类型:append、set、remove、all、clear，分别对应选中模型上的五种操作元素选中状态的函数:appendSelection、setSelection、removeSelection、selectAll、clearSelection。
另外TWaver还提供了三中选择模式:多选(mutipleSelection)、单选(‘singleSelection’)、不可选(‘noneSelection’),用于控制选中效果,也为视图组件选择模式提供了数据层支持,默认为多选模式。

```
//设置选择模式
selectionModel.setSelectionMode('singleSelection');
```

Note:

当选择模式发生切换的时候，TWaber内部会首先调用清除所有元素的选中状态。

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <title>Alarm</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);
        var selectionModel = box.getSelectionModel();

        function init() {
            initNetwork();
            initListener();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }


        function initDataBox() {
            for (var i = 0; i < 100; i++) {
                var node = new twaver.Node(i);
                node.setName('node-' + parseInt(i / 4));
                node.setClient('NO', i % 4);
                box.add(node);
            }

            selectionModel.appendSelection(box.getDataById(0));
            selectionModel.appendSelection(box.getDataById(1));
            selectionModel.removeSelection(box.getDataById(0));
            selectionModel.setSelection([box.getDataById(2), box.getDataById(6)]);
            selectionModel.appendSelection([box.getDataById(2), box.getDataById(3), box.getDataById(4)]);

            //single selection mode
            console.log('设置单选模式,首先清除当前的选中状态,以后每次只选择最多一个元素');
            selectionModel.setSelectionMode('singleSelection');
            selectionModel.appendSelection([box.getDataById(2), box.getDataById(3), box.getDataById(4)]);
            console.log('selection size:' + selectionModel.size());

            //none selection mode
            console.log('设置不可选模式,会清除当前的选中状态');
            selectionModel.setSelectionMode('noneSelection');
            selectionModel.appendSelection([box.getDataById(2), box.getDataById(3), box.getDataById(4)]);

            //multiple selection mode
            console.log('默认是多选模式');
            selectionModel.setSelectionMode('multipleSelection');

            console.log('\n设置过滤器,所有id大于5的都不可选');
            selectionModel.setFilterFunction(function (data) {
                return data.getId() > 5;
            });
            selectionModel.selectAll();
```

```
        }

        function initListener() {
            selectionModel.addSelectionChangeListener(function (e) {
                console.log(e);
//                console.log('kind:' + e.kind + ',datas:' + e.datas.toString());
            });
        }
    </script>

</head>
<body onload="init()">
</body>
</html>
```

```
▶ Object {kind: "append", datas: $List}
▶ Object {kind: "append", datas: $List}
▶ Object {kind: "remove", datas: $List}
▶ Object {kind: "set", datas: $List}
▶ Object {kind: "append", datas: $List}
设置单选模式,首先清除当前的选中状态,以后每次只选择最多一个元素
▶ Object {kind: "clear", datas: $List}
▶ Object {kind: "append", datas: $List}
selection size:1
设置不可选模式,会清除当前的选中状态
▶ Object {kind: "clear", datas: $List}
默认是多选模式

设置过滤器,所有id大于5的都不可选
▶ Object {kind: "all", datas: $List}
```

快速查找(QuickFinder)

快速查找用于快速查找指定属性值的所有元素，如查找某个告警级别的所有告警，特定类型的所有网元等等。

```
/**
 * dataBox:数据容器,查找器查找的对象
 * propertyName:属性名称,查找条件
 * propertyType:属性类型,默认为'accessor',表示可直接获取的属性,属性类型支持三种:accessor、client、style等
 * valueFunction:值获取函数，默认根据上面三种方式获取数据，用户也可以指定值的获取方式，通过设置valueFunction实现
 * filterFunction:过滤函数，用于控制哪些元素是否参与查找
 */
twaver.QuickFinder = function(dataBox,propertyName,propertyType,valueFunction,filterFunction)

/**
 *filterFunction:传入data，返回true/false,表示这个元素是否参与查找
 */
filterFunction = function(data){}
```

```
//返回第一个查找到的元素
findFirst: function (value)
//返回找到的所有元素的集合
find: function(value)
```

使用QuickFinder步骤,以查找指定名称元素为例:

(1).首先创建一个与name属性绑定的查找器

```
/**
 * databox:数据容器
 * 'name':查找条件
 */
var nameFinder = new waver.QuickFinder(databox,'name');
```

(2).开始查找

```
/**
 * 返回值:返回查找到的元素集合
 */
var datas = nameFinder.find('group-1');
```

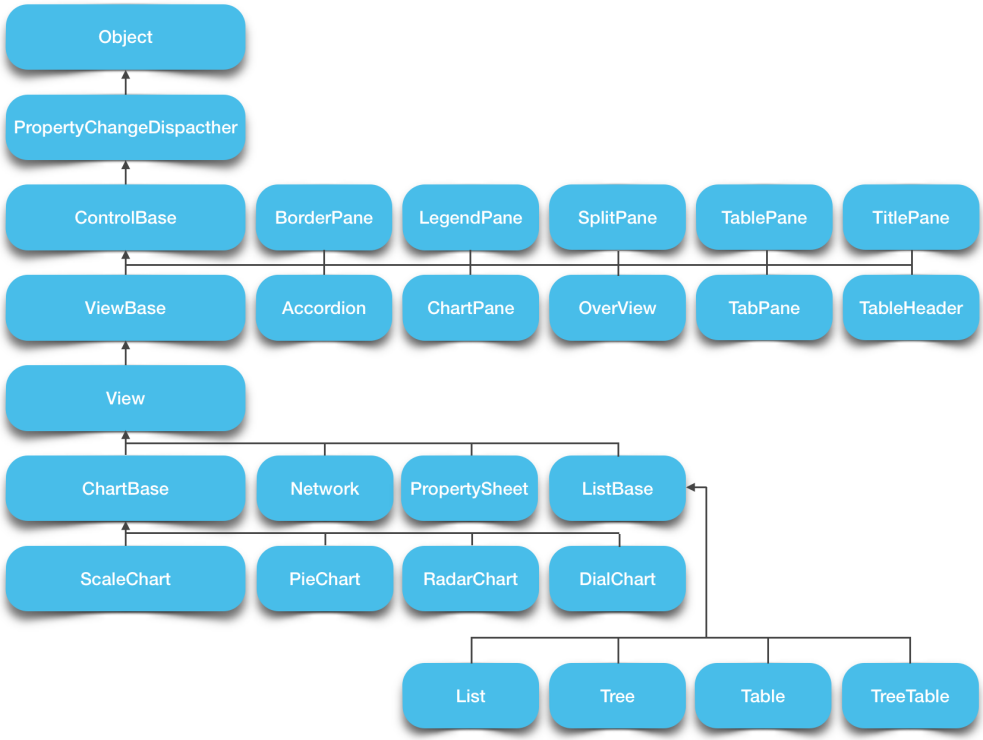
```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <title>Alarm</title>
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }
```

```
22
23          function initDataBox() {
24              for (var i = 0; i < 100; i++) {
25                  var node = new twaver.Node();
26                  node.setName('node-' + parseInt(i / 4));
27                  node.setClient('NO', i % 4);
28                  box.add(node);
29              }
30
31              var nameFinder = new twaver.QuickFinder(box, 'name');
32              var noFinder = new twaver.QuickFinder(box, 'NO', 'client');
33
34              var data = nameFinder.findFirst('node-1');
35              console.log("nameFinder.findFirst('node-1')",data);
36
37              var datas = nameFinder.find('node-1');
38              console.log("nameFinder.find('node-1')",datas);
39
40              data = noFinder.findFirst(1);
41              console.log("noFinder.findFirst(1)",data);
42
43              datas = noFinder.find(1);
44              console.log("noFinder.find(1)",datas);
45
46          }
47      </script>
48
49  </head>
50  <body onload="init()">
51  </body>
52  </html>
```

```
nameFinder.findFirst('node-1') ▼ $Node {_location: Object, _styleMap: Object, _alarmState: twaver.AlarmState, _dispatcher: twaver.EventDispatcher, _childList: $List…}
                                 ▶ _alarmState: twaver.AlarmState
                                 ▶ _childList: $List
                                 ▶ _childMap: Object
                                 ▶ _clientMap: Object
                                 ▶ _dispatcher: twaver.EventDispatcher
                                   _id: "D1AD59EBBFA84ACB8D5B827FBB8DE462"
                                 ▶ _location: Object
                                   _name: "node-1"
                                 ▶ _styleMap: Object
                                 ▶ __proto__: F
                                                                                                                                  qui
nameFinder.find('node-1') ▼ $List {_as: Array[4], constructor: function, getClassName: function, size: function, isEmpty: function…}                qui
                            ▶ _as: Array[4]
                            ▶ __proto__: F
noFinder.findFirst(1) ▼ $Node {_location: Object, _styleMap: Object, _alarmState: twaver.AlarmState, _dispatcher: twaver.EventDispatcher, _childList: $List…}
                        ▶ _alarmState: twaver.AlarmState
                        ▶ _childList: $List
                        ▶ _childMap: Object
                        ▶ _clientMap: Object
                        ▶ _dispatcher: twaver.EventDispatcher
                          _id: "32FB35CCC8844B05ADCF935ADDB4E6C6"
                        ▶ _location: Object
                          _name: "node-0"
                        ▶ _styleMap: Object
                        ▶ __proto__: F
                                                                                                                                  qui
noFinder.find(1) ▼ $List {_as: Array[25], constructor: function, getClassName: function, size: function, isEmpty: function…}                qui
                   ▶ _as: Array[25]
                   ▶ __proto__: F
```

概述

可视化视图组件，顾名思义就是用来呈现数据的组件,是TWaver产品中非常重要的一部分，TWaver支持多种可视化视图组件,常用的有Network组件、Tree组件、Table组件、Chart组件等，它们都继承于Object类，其继承关系可以参考下图。



下面我们将逐一介绍这些可视化组件。

网元可视化视图组件(Network)

Network组件是TWaver产品中最重要的组件之一，本章将详细介绍拓扑图组件的设计和使用，包括拓扑图组件的层次结构，背景的添加，各种过滤器的使用以及交互模式的切换。

层次结构

Network分为三个层次，最底层是view(div),在此基础上放置两个Canvas,分别是rootCanvas和topCanvas。rootCanvas用于绘制背景和网元,topCanvas可用于绘制附件、告警、编辑框等元素。

1.view：是最底层div元素，rootCanvas和topCanvas均放置在这个div上。

获取方式:network.getView()

示例:如修改背景颜色

```
network.getView().style.backgroundColor = '#094AB2';
```

2.rootCanvas用户绘制一些比较底层的元素，如背景、网元、自定义其他元素。

获取方式:Network#getRootCanvas();

绘制顺序:

(1)绘制背景,内部调用$backgroundUI.draw(ctx, this);

(2)拦截绘制paintBottom内容:内部调用this.paintBottom(ctx, dirtyRect);

(3)绘制网元:ui.paint(ctx);

示例1:添加背景图片

TWaver默认支持添加背景图片功能,设置Network的背景图片可以直接调用下面代码实现。

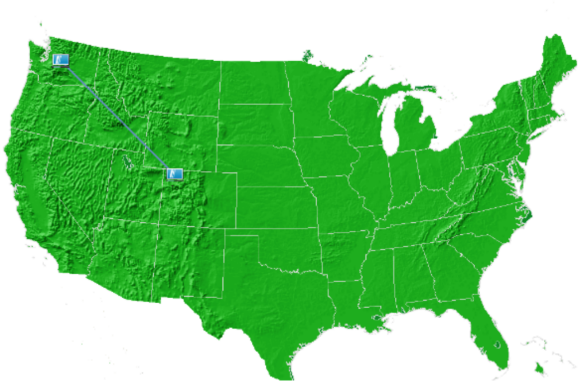
```
box.setStyle('background.type', 'image');
box.setStyle('background.image', 'image_name');
```

其中image_name是用twaver.Util.registerImage方式注册的图片。

代码如下:

```
//使用示例
function registerImage(){
```

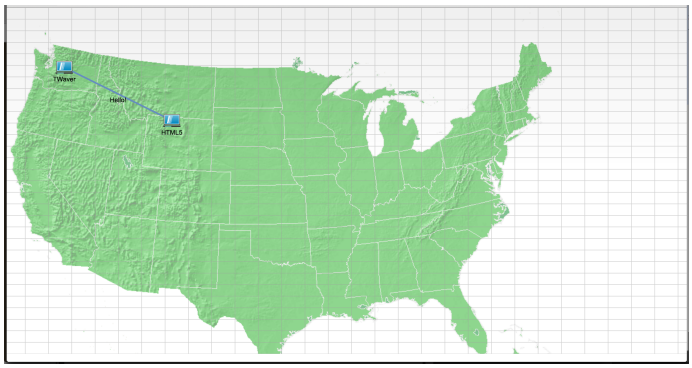
```
  registerNormalImage('../images/bg.png','bg');
}
function registerNormalImage(url, name) {
  var image = new Image();
  image.src = url;
  image.onload = function() {
    twaver.Util.registerImage(name, image, image.width, image.height);
    image.onload = null;
    network.invalidateElementUIs();
  };
}
box.setStyle('background.type', 'image');
box.setStyle('background.image', 'bg');
```



示例2:在背景图片和网元之间绘制一张网状表格

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>TWaver HTML5</title>
  <script src="../lib/twaver.js"></script>
  <script>
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    function init() {
      registerImage();
      document.body.appendChild(network.getView());
      network.adjustBounds({ x: 0, y: 0, width: 1300, height: 650 });
      //network.setBackgroundImage('welcome');
      box.setStyle('background.type', 'image');
      box.setStyle('background.image', 'usa');
      box.setStyle('background.image.stretch','none');
      // network.setDragToPan(false);

      // network._view.style.backgroundColor = '#094AB2';
      var node1 = new twaver.Node();
      node1.setName('TWaver');
      node1.setLocation(100, 100);
      box.add(node1);

      var node2 = new twaver.Node();
      node2.setName('HTML5');
      node2.setLocation(300, 200);
      box.add(node2);

      var link = new twaver.Link(node1, node2);
      link.setName('Hello!');
      box.add(link);

      network.addViewListener(function(e) {
        if (e.kind === 'validateEnd' ) {
        }
      });

      network.paintBottom = function (ctx, dirtyRect){
        var rootCanvas = network.getRootCanvas();
        var gradient = ctx.createLinearGradient(0,0,0,300);
        gradient.addColorStop(0,"#e0e0e0");
        gradient.addColorStop(1,"#ffffff");
        ctx.globalAlpha=0.5;
        ctx.fillStyle = gradient;
        ctx.fillRect(0,0,rootCanvas.width,rootCanvas.height);

          // 描绘边框
          var grid_cols = 60;
          var grid_rows = 60;
          var cell_height = rootCanvas.height / grid_rows;
          var cell_width = rootCanvas.width / grid_cols;
          ctx.lineWidth = 1;
          ctx.strokeStyle = "#a0a0a0";

          // 结束边框描绘
          ctx.beginPath();
          // 准备画横线
          for (var col = 0; col <= grid_cols; col++) {
            var x = col * cell_width;
            ctx.moveTo(x,0);
            ctx.lineTo(x,rootCanvas.height);
          }
          // 准备画竖线
```

```
            for(var row = 0; row <= grid_rows; row++){
              var y = row * cell_height;
              ctx.moveTo(0,y);
              ctx.lineTo(rootCanvas.width, y);
            }
            ctx.stroke();
          }
        }

        function registerImage(){
          registerNormalImage('./usa.gif','usa');
        }

        function registerNormalImage(url, name) {
          var image = new Image();
          image.src = url;
          image.onload = function() {
            twaver.Util.registerImage(name, image, image.width, image.height);
            image.onload = null;
            network.invalidateElementUIs();
          };
        }
      </script>
    </head>
    <body onload="init()">
    </body>
    </html>
```



2.topCanvas

获取方式:Network#getTopCanvas();

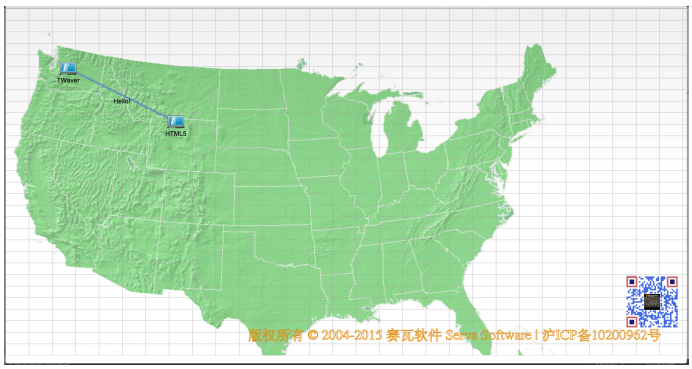
绘制顺序:

(1)绘制Marker等元素:this.paintMarker(ctx);

(2)拦截绘制paintTop内容:内部调用this.paintTop(ctx,dirtyRect);

示例：在topCanvas层绘制公司信息

```
  network.paintTop = function (ctx, dirtyRect) {
          var img=new Image();
          img.src="./twaver.png";
          ctx.drawImage(img,1150,500,100,100);
          ctx.font="26px 宋体";
          ctx.strokeStyle="#E38A0A";
          ctx.strokeText("版权所有 © 2004-2015 赛瓦软件 Serva Software | 沪ICP备10200962号",450,620);
        }
}
```



过滤器

延续TWaver的一贯风格，TWaver HTML5提供了一系列过滤器，包括可见过滤器、可移动过滤器、可编辑过滤器，通过设置过滤器可以实现同一数据模型，不同信息的显示以及不同的操作模式，通常用于按用户权限或者网元类型产生不同的交互和视图。

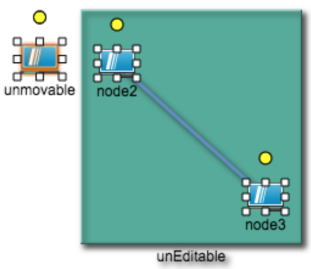
Network的过滤器包括

```
//可见过滤器
get/setVisibleFunction:function(filter)
//可移动过滤器
get/setMovableFunction:function(filter)
```

```
//可编辑过滤器
get/setEditableFunction:function(filter)
```

过滤器的使用示例，注意传入参数类型是Element,返回参数为Boolean。

```
<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});

            network.setEditInteractions();

            network.setVisibleFunction(function (element) {
                if (element.getClient("visible") == false) {
                    return false;
                }
                return true;
            });

            network.setEditableFunction(function (element) {
                if (element.getClient('editable') == false) {
                    return false;
                }
                return true;
            });

            network.setMovableFunction(function (element) {
                if (element.getClient('movable') == false) {
                    return false;
                }
                return true;
            });
        }

        function initDataBox() {
            var node = new twaver.Node();
            node.setName("unmovable");
            node.setLocation(50, 60);
            node.setClient('movable',false);
            box.add(node);

            var node1 = new twaver.Node();
            node1.setName("unvisible");
            node1.setLocation(60, 90);
            node1.setClient("visible", false);
            box.add(node1);

            var node2 = new twaver.Node();
            node2.setName("node2");
            node2.setLocation(80, 100);
            box.add(node2);

            var node3 = new twaver.Node();
            node3.setName("node3");
            node3.setLocation(120, 210);
            box.add(node3);

            var link = new twaver.Link(node2,node3);
            box.add(link);

            var group = new twaver.Group();
            group.setName('unEditable');
            group.isExpanded = function(){
                return true;
            }
            group.addChild(node3);
            group.addChild(node2);
            group.setClient('editable',false);
            group.s('group.fill.color','#ef8200');
            box.add(group);
        }

// ]]></script>
```



工具条

创建工具条示例：

```
<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);
        var toolbar = document.createElement('div');

        function init() {
            initToolbar();
            initNetwork();
            initBox();
        }

        function initNetwork() {
            var networkPane = new twaver.controls.BorderPane(network, toolbar);
            networkPane.setTopHeight(25);
            var view = networkPane.getView();
            view.style.left = '0px';
            view.style.top = '0px';
            view.style.right = '0px';
            view.style.bottom = '0px';
            document.body.appendChild(view);
            window.onresize = function () {
                networkPane.invalidate();
            };
        }

        function initToolbar() {
            addButton(toolbar, 'Pan', 'pan', function () {
            });

            addButton(toolbar, 'Zoom In', 'zoomIn', function () {
                network.zoomIn();
            });
            addButton(toolbar, 'Zoom Out', 'zoomOut', function () {
                network.zoomOut();
            });
            addButton(toolbar, 'Zoom Overview', 'zoomOverview', function () {
                network.zoomOverview();
            });
            addButton(toolbar, 'Zoom Reset', 'zoomReset', function () {
                network.zoomReset();
            });
        }

        function addButton(toolbar, label, src, handler) {
            var button = document.createElement('input');
            button.value = label;
            button.onclick = handler;
            button.setAttribute('type', src ? 'image' : 'button');
            if (src) {
                button.style.padding = '4px 4px 4px 4px';
                if (src.indexOf('/') < 0) {
                    src = '../images/toolbar/' + src + '.png';
                }
                button.setAttribute('src', src);
            } else {
                button.value = label;
            }
            toolbar.appendChild(button);
        }

        function initBox() {
            var from = new twaver.Node();
            from.setName('From');
            from.setLocation(100, 100);
            box.add(from);

            var to = new twaver.Node();
            to.setName('To');
            to.setLocation(300, 300);
            box.add(to);

            var link = new twaver.Link(from, to);
            link.setName('Link');
            box.add(link);
        }

// ]]></script>
```

Note:如何自定义工具条、派发事件、监听事件？

下面的例子展示如何创建清除所有网元Clear按钮。

```
addButton(toolbar, 'Clear', null, function () {
    box.clear();
    //可以派发自定义事件
     network.fireInteractionEvent({ kind: 'ClearAllElement', element: null });
    });
```

操作Clear按钮执行box.clear()方法，并派发ClearAllElement事件,这个事件可以在network.addInteractionListener(function(e){console.log(e)})中监听到,输出e.kind为”ClearAllElement”。

与此类似的,通过toolbar创建网元的时候，可监听到网元创建成功的事件,可执行回调。

```
addButton(toolbar, 'Rack', 'rack_icon', function () {
network.setCreateElementInteractions(Rack); });
addButton(toolbar, 'Shelf', 'shelf_icon', function () {
network.setCreateElementInteractions(Shelf); });
addButton(toolbar, 'Slot', 'slot_icon', function () {
network.setCreateElementInteractions(Slot); });
addButton(toolbar, 'Card', 'card_icon', function () {
network.setCreateElementInteractions(Card); });
addButton(toolbar, 'Port', 'port_icon', function () {
network.setCreateElementInteractions(Port); });
addButton(toolbar, 'LED', 'led_icon', function () {
network.setCreateElementInteractions(LED); });
addButton(toolbar, 'Text', 'text_icon', function () {
network.setCreateElementInteractions(Text); });
```

当创建成功后，可以监听到事件”createElement”。

界面交互

拓扑图上的鼠标、键盘、触屏交互通过一堆Interaction组成，每个Interaction包含独立的事件监听和卸载操作，其中最基本的实现类是twaver.network.interaction.BaseInteraction，其他交互类都是从它继承衍生出来的，下面是*BaseInteraction{*}的实现代码，其中的setUp函数用于添加监听事件，tearDown函数卸载监听，实现交互监听类时，通常需要重写这两个方法。

```
twaver.vector.interaction.BaseInteraction = function (network) {
    this.network = network;
};
_twaver.ext('twaver.vector.interaction.BaseInteraction', Object, {
    setUp: function () {
    },
    tearDown: function () {
    },
    addListener: function () {
        for (var i = 0; i &lt; arguments.length; i++) {
            var type = arguments[i];
            $html.addEventListener(type, 'handle_' + type, this.network.getView(), this);
        }
    },
    removeListener: function () {
        for (var i = 0; i &lt; arguments.length; i++) {
            $html.removeEventListener(arguments[i], this.network.getView(), this);
        }
    },
});
```

Network中预定义了一些常用的几种交互模式，包括默认模式、编辑模式、创建连线模式、创建多点连线、多边形网元以及触控交互模式，此外放大镜模式也很容易实现。

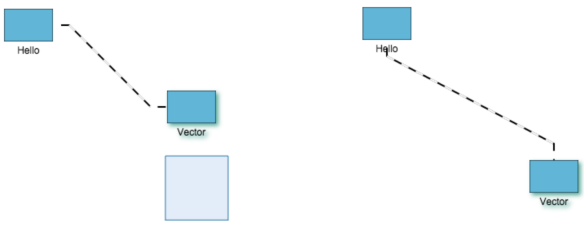
```
//设置默认模式
setDefaultInteractions:function(lazyMode)
//设置编辑模式
setEditInteractions:function(lazyMode)
//设置创建网元模式，type为可为网元类型或者function,如:twaver.Node或者MyElement(自定义交互模式)
setCreateElementInteractions:function(type)
//设置创建连线模式
setCreateLinkInteractions:function(type)
//设置创建形状连线模式
setCreateShapeLinkInteractions:function(type)
//设置创建形状网元模式
setCreateShapeNodeInteractions:function(type)
//设置触屏模式
setTouchInteraction:function()
```

默认交互模式(正常模式):

```
network.setDefaultInteractions(false)
```

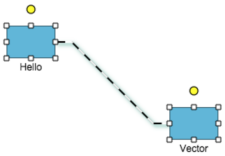
默认交互模式(lazy模式):

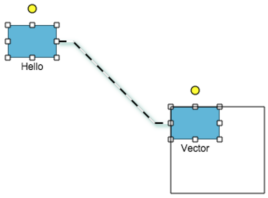
```
network.setDefaultInteractions(true)
```

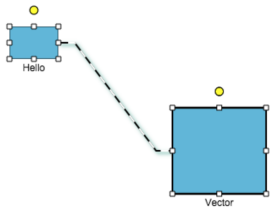


编辑交互模式:

```
network.setEditorInteractions(true)
```





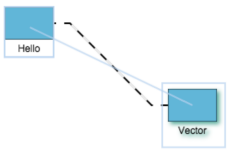


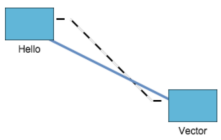
创建网元交互模式:

```
network.setCreateElementInteractions(Node)
```

创建Link交互模式:

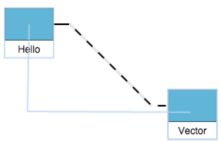
```
network.setCreateLinkInteractions(Node)
```

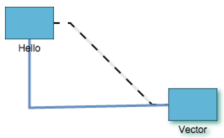




创建ShapeLink交互模式:

```
network.setCreateShapeLinkInteractions(twaver.ShapeLink);
```





Note:如何设置两网元之间只允许创建一条Link？

```
/**
 * node:鼠标下当前网元
 * fromNode:当前创建link的起始网元
 */
var  linkableFunction = function(node,fromNode){
            if(node instanceof twaver.Node){
            if(fromNode){//isToNode
                var links = twaver.Util.getSharedLinks(node,fromNode);
                if(!links){
                    return true;
                }else{
                    if(links.size() == 0){
                        return true;
                    }
                    var l = links.get(0);
                    var fNode = l.getFromNode();
                    var tNode = l.getToNode();
                    if(node === tNode){
                        return !(fromNode === fNode);
                    }else if(node === fNode){
                        return !(fromNode === tNode);
                    }
                }
            }
            return true;
        }
    }
network.setLinkableFunction(linkableFunction);
```

创建ShapeNode交互模式:

```
network.setCreateShapeNodeInteractions();
```

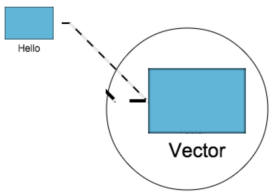
Note: Network初始化的时候会检测设备，判断当前设备支持的交互方式，设置对应的交互模式。

```
if ($ua.isMSToucheable) {
    this.setMSTouchInteractions();
}else if ($ua.isTouchable) {
    this.setTouchInteractions();
} else {
    this.setDefaultInteractions(false);
}
```

放大镜交互模式:

```
/**
 * network:Network
 * scale:放大倍数,默认值为2
 * xRadius:放大镜的宽度,默认值为100
 * yRadius:放大镜的高度,默认值为100
 */
twaver.vector.interaction.MagnifyInteraction = function(network, scale, xRadius, yRadius) {}
```

```
network.setInteractions([
     new twaver.vector.interaction.DefaultInteraction(network),
     new twaver.vector.interaction.MagnifyInteraction(network,2,100,100)
]);
```



下面的例子可以切换各种交互模式，并简单实现了拖拽按钮创建网元的功能，用户可以参考。

```
TWaver HTML5 Demo<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
        var network;
        function init(){
            network = new twaver.vector.Network();

            var box = network.getElementBox();
            var node = new twaver.Node();
            node.setName("from");
            node.setLocation(100, 100);
            box.add(node);
            var node2 = new twaver.Node();
            node2.setName("to");
            node2.setLocation(300, 300);
            box.add(node2);

            var link = new twaver.Link(node, node2);
            link.setName("Hello TWaver");
            link.setToolTip("<b>Hello TWaver</b>");
            box.add(link);

            var view = network.getView();
       document.getElementById("network").appendChild(view);
        network.adjustBounds({x: 20, y: 20, width: 1300, height: 400});
        }
        function setMagnifyInteraction(){
            network.setInteractions([
                // new twaver.network.interaction.SelectInteraction(network),
                // new twaver.network.interaction.MoveInteraction(network),
                new twaver.network.interaction.MagnifyInteraction(network),
                new twaver.network.interaction.DefaultInteraction(network)]);
        }

// ]]></script>
```

Default Edit Create Element Create Link Create ShapeLink Create ShapeNode Magnify

定制交互模式

定制交互监听器需要继承BaseInteraction类，然后重写setUp和tearDown方法，分别执行添加鼠标、键盘、触屏监听器和卸载这些监听动作。

下面我们以一个实例来说明，我们将实现鼠标移动到Node网元上，网元图标发生变化的功能,监听了mousemove事件。

(1)继承BaseInteraction，实现一个阴影交互监听器。

```
function shadowInteraction(network){
    shadowInteraction.superClass.constructor.call(this,network);
}

twaver.Util.ext('shadowInteraction',twaver.vector.interaction.BaseInteraction,{
    setUp:function(){
        this.addListener('mousemove');
    },
    tearDown:function(){
        this.removeListener('mousemove');
    },
    handle_mousemove:function(e){
        var element  = this.network.getElementAt(e);
        if(element){
            this.shadow(element);
        }else{
            this.reset();
        }
    },
    shadowElement:null,
    shadow:function(element){
        this.reset();
        if(element instanceof  twaver.Node){
            this.shadowElement = element;
            element.setImage('shadow');
        }
    },
    reset:function(){
         if(this.shadowElement){
             this.shadowElement.setImage('node_image');
         }
    }
});
```

(2)使用上面定义的交互器，通过调用Network#setInteractions(array)方法，给拓扑图设置一组交互监听器，这样各个交互监听器都可以各司其职。

```
network.setInteractions([
    new twaver.vector.interaction.SelectInteraction(network),
    new twaver.vector.interaction.MoveInteraction(network),
    new shadowInteraction(network),
    new twaver.vector.interaction.DefaultInteraction(network)
]);
```

```
Test Zoom<script src="../twaver.js"></script><script src="./shadowInteraction.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);

        function init() {
            initNetwork();
            initDataBox();
            registerImage();
        }

        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            network.setInteractions([
                new twaver.vector.interaction.SelectInteraction(network),
                new twaver.vector.interaction.MoveInteraction(network),
                new shadowInteraction(network),
                new twaver.vector.interaction.DefaultInteraction(network)
            ]);
        }
        function registerImage() {
            //register shadow
            twaver.Util.registerImage('shadow', {
                w: 37,
                h: 29,
                shadowOffsetX: 0,
                shadowOffsetY: 0,
                shadowBlur: 5,
                shadowColor: '#ec6c00',
                v: [
                    {
                        shape: 'vector',
                        name: 'node_image',
                        x: 0,
                        y: 0
                    }
                ]
            });

        }

        function initDataBox() {
            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
```

```
                }
            });
            box.add(from);

            var to = new twaver.Node({
                name: 'to',
                location: {
                    x: 220,
                    y: 160
                }
            });
            box.add(to);

            var link = new twaver.Link(from, to);
            box.add(link);
        }

// ]]></script>
```

示例:实现一个可以实现创建指定网元的按键。

```
addButton(toolbar, 'Rack', 'rack_icon', function () {
network.setCreateElementInteractions(MyElement); });
```

```
MyElement = function (id) {
    MyElement.superClass.constructor.call(this, id);

    this.setStyle('body.type', 'vector');
    this.setStyle('vector.shape', 'rectangle');
    this.setStyle('vector.gradient', 'none');
    this.setStyle('vector.fill.color', '#C0C0C0');
    this.setStyle('vector.deep', 8);
    this.setWidth(160);
    this.setHeight(160);
};
twaver.Util.ext('MyElement', twaver.Follower);
```

监听事件

```
/**
 *addViewListener主要用于监听视图绘制的事件
 *network.addViewListener(function(e){console.log(e);});
 *
 */
 addViewListener: function (listener, scope, ahead) {
        this._viewDispatcher.add(listener, scope, ahead);
 },
 removeViewListener: function (listener, scope) {
        this._viewDispatcher.remove(listener, scope);
 },
 fireViewEvent: function (evt) {
        this._viewDispatcher.fire(evt);
 },
```

一般在创建Network和刷新Network的时候会派发视图事件，可以监听到"invalidate"、"validateStart"、"validateEnd"等。

```
Object {kind: "validateStart"}
Object {kind: "validateEnd"}
Object {kind: "invalidate"}
Object {kind: "validateStart"}
Object {kind: "validateEnd"}
```

```
/**
 *addInteractionListener主要用于监听视图操作的事件
 *network.addInteractionListener(function(e){console.log(e);});
 *
 */
 addInteractionListener: function (listener, scope, ahead) {
        this._interactionDispatcher.add(listener, scope, ahead);
 },
 removeInteractionListener: function (listener, scope) {
        this._interactionDispatcher.remove(listener, scope);
 },
 fireInteractionEvent: function (evt) {
        this._interactionDispatcher.fire(evt);
 },
```

在Network上执行的操作都可以通过addInteractionListener监听到,TWaver内部常见的派发事件有:

```

```

```
/**
* 添加交互事件监听器，用于监听用户各种操作
* @id twaver.controls.ViewBase.addInteractionListener
* @param {Function} listener 回调函数
* @param {Object} [scope] 可选，回调函数的作用域，默认为null，即全局
* @param {Boolean} [ahead] 可选，是否将此监听放在最前面，默认为false
* @example viewBase.addInteractionListener(function (e) {
* console.log(e.kind, e.element);
* });
*
* kind可以为下列值:
* createElement 创建网元
* clickElement 单击网元
* doubleClickElement 双击网元
* clickBackground 单击背景
* doubleClickBackground 双击背景
* removeElement 删除网元
* selectAll 选中所有网元
* upSubNetwork 进入上一层子网
* enterSubNetwork 进入子网
* bundleLink 展开或合并连线捆绑
* expandGroup 展开组
* liveMoveStart 开始实时移动网元
* liveMoveBetween 正在实时移动网元
* liveMoveEnd 结束实时移动网元
* lazyMoveStart 开始延迟移动网元
* lazyMoveBetween 正在延迟移动网元
* lazyMoveEnd 结束延迟移动网元
* liveResizeStart 开始实时改变网元大小
* liveResizeBetween 正在实时改变网元大小
* liveResizeEnd 结束实时改变网元大小
* lazyResizeStart 开始延迟改变网元大小
* lazyResizeBetween 正在延迟改变网元大小
* lazyResizeEnd 结束延迟改变网元大小
* selectStart 开始框选
* selectBetween 正在框选
* selectEnd 结束框选
* liveMovePointStart 开始移动ShapeLink或ShapeNode的点
* liveMovePointBetween 正在移动ShapeLink或ShapeNode的点
* liveMovePointEnd 结束移动ShapeLink或ShapeNode的点
*/
twaver.controls.ViewBase.prototype.addInteractionListener = function (listener, scope, ahead) {}
松开鼠标的事件为 liveMoveEnd或lazyMoveEnd，获取网元的坐标可以用
network.getMovableSelectedElements().forEach(e){
    console.log(e.getLocation);
};
```

```
/**
 *network.getView.addEventListener(MouseEvent', function (e) {});
 *监听鼠标事件: mousedown、mousemove、mouseup、keydown等。
 */
```

```
network.getView().addEventListener('mousemove', function (e) {
            var target = self.network.hitTest(e);
            if (target) {
                if (target instanceof twaver.network.ElementUI) {
                    console.log('滑过一次！！');
                var testnode = self.network.getElementAt(e);
                alert(testnode.getName());
                 }
                 // if (target instanceof twaver.network.LabelAttachment) {
                     // console.log('clicked LabelAttachment');
                 // }
            }
        });
```

```
//获取鼠标下的网元
network.getView().addEventListener('mousemove',function(e){
    var element = network.getElementAt(e);
})
```

Network样式

TWaver HTML5中使用规则函数的方式设置组件的样式，包括树图或拓扑图中节点颜色渲染、边框、冒泡等。本节主要介绍拓扑图中的相关规则。拓扑图中网元的颜色渲染、边框颜色、告警颜色、告警文本、连线捆绑文本、工具条提示文本等都是通过规则函数的方式实现的，用户也可以设置自己的规则函数实现定制。

Network中的样式规则

getLabel：网元文本标签函数,用于设置网元的标签

getToolTip：工具体提示

getInnerColor：网元内渲染色函数

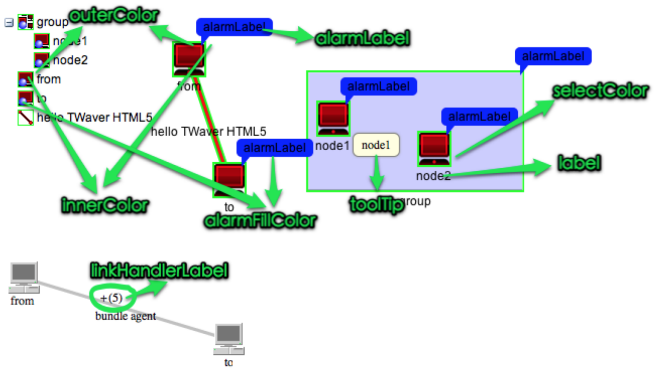
getOuterColor：网元边框颜色

getSelectColor：网元选中颜色

getAlarmFillColor：告警冒泡颜色

getAlarmLabel：告警冒泡文本

getLinkHandlerLabel：绑定连线文本



默认实现

TWaver的默认实现列举如下:

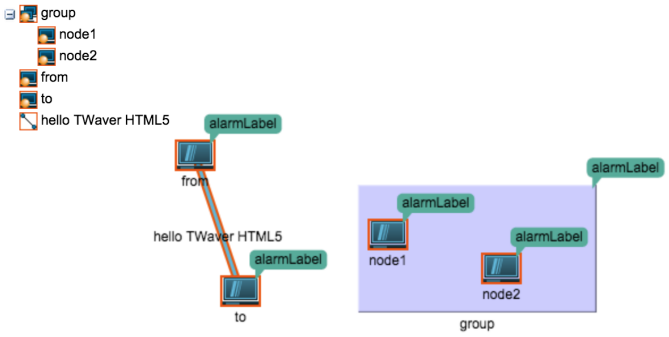
```
//label
Network#getLabel: function (data) {
        return data.getStyle('network.label') || data.getName();
    };
Tree#getLabel = function (data) {
        return data.getName();
    };
//toolTip
Network#getToolTip = function (data) {
        return data.getToolTip();
    };

//innerColor
Network/Tree#
getInnerColor = function (data) {
        if (data.IElement) {
            var severity = data.getAlarmState().getHighestNativeAlarmSeverity();
            if (severity) {
                return severity.color;
            }
            return data.getStyle('inner.color');
        }
        return null;
    };

//outerColor
Network/Tree#
getOuterColor = function (data) {
        if (data.IElement) {
            var severity = data.getAlarmState().getPropagateSeverity();
            if (severity) {
                return severity.color;
            }
            return data.getStyle('outer.color');
        }
        return null;
    };

//selectColor
Network#getSelectColor: function (element) {
        return element.getStyle('select.color');
    };

//alarmFillColor
Network/Tree#
getAlarmFillColor = function (data) {
        if (data.IElement) {
            var severity = data.getAlarmState().getHighestNewAlarmSeverity();
            if (severity) {
                return severity.color;
            }
        }
        return null;
    };

//alarmLabel
getAlarmLabel: function (element) {
        var severity = element.getAlarmState().getHighestNewAlarmSeverity();
        if (severity) {
            var label = element.getAlarmState().getNewAlarmCount(severity) + severity.nickName;
            if (element.getAlarmState().hasLessSevereNewAlarms()) {
                label += "+";
            }
            return label;
        }
        return null;
    }

//linkHandlerLabel
Network#
getLinkHandlerLabel: function (link) {
        if (link.isBundleAgent()) {
            return "+(" + link.getBundleCount() + ")";
        }
        return null;
    };
```

定制规则

TWaver的默认实现列举如下:

```
TWaver HTML5 Demo<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
  var box;
  var number;
  function init() {
    number = 0;
    var network = new twaver.vector.Network();
    network.setToolTipEnabled(true);
    box = network.getElementBox();
    var tree = new twaver.controls.Tree(box);

    var group = new twaver.Group();
    group.setName("group");
    box.add(group);
    group.addChild(createTWaverNode("node1", 200, 100));
    group.addChild(createTWaverNode("node2", 300, 130));
    group.setExpanded(true);

    var from = createTWaverNode("from", 30, 30);
    var to = createTWaverNode("to", 70, 150);
    var link = new twaver.Link(from, to);
    link.setName("hello TWaver HTML5");
    box.add(link);

    // 树节点自身渲染颜色
    tree.getInnerColor = function(data) {
      // return "#ff0000";
      return twaver.Colors.blue_dark;
    };

    // 树节点边框颜色
    tree.getOuterColor = function(data) {
      // return "#00ff00";
      return twaver.Colors.orange_dark;
    };

    // 左下方小球的颜色，如果为null表示不显示
    tree.getAlarmFillColor = function(data) {
      if (data instanceof twaver.Element && !data.getAlarmState().isEmpty()) {
        // return "#0000ff";
        return twaver.Colors.orange_light;
      }
      return null;
    };

    // network是同样的原理，同样的默认实现，
    // innerColor - 节点渲染色
    // outerColor - 边框颜色
    // alarmFillColor - 告警冒泡颜色，如果下面的告警文本为空，告警冒泡不显示
    // alarmLabel - 告警文本
    // Body 渲染色
    network.getInnerColor = function(data) {
      // return "#ff0000";
      return twaver.Colors.blue_light;
    };

    // 节点边框颜色
    network.getOuterColor = function(data) {
      // return "#00ff00";
      return twaver.Colors.orange_dark;
    };

    // 告警冒泡的颜色，如果下面相应的alarmLabel返回null或者颜色为null不显示
    network.getAlarmFillColor = function(data) {
      if (data instanceof twaver.Element && !data.getAlarmState().isEmpty()) {
        // return "#0000ff";
        return twaver.Colors.green_light;
      }
      return null;
    };

    network.getAlarmLabel = function(element) {
      if (!element.getAlarmState().isEmpty()) {
        return "alarmLabel";
      }
      return null;
    };

    network.getSelectColor = function(element) {
      // return "#ffff00";
      return twaver.Colors.gray_dark;
    };

    var treeDom = tree.getView();
    treeDom.style.width = "150px";
    treeDom.style.height = "100%";
    // var networkDom = network.getView();
    // networkDom.style.left = "150px";
    // networkDom.style.width = "100%";
    // networkDom.style.height = "100%";

    var view = network.getView();
    document.body.appendChild(network.getView());
    network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
    document.body.appendChild(treeDom);
}

function createTWaverNode(name, x, y) {
  var node = new twaver.Node();
```

```
    node.setName(name);
    node.setToolTip(name);
    node.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.MAJOR);
    node.setClient("number", number++);
    node.setLocation(x, y);
    box.add(node);
    return node;
}
// ]]></script>
```



Network中含有很多样式设置，详细可以参考API#twaver.vector.Network。

自动布局

拓扑应用有时需要一些常规的布局规则，TWaver Network组件支持布局模型和使用方法分别如下:

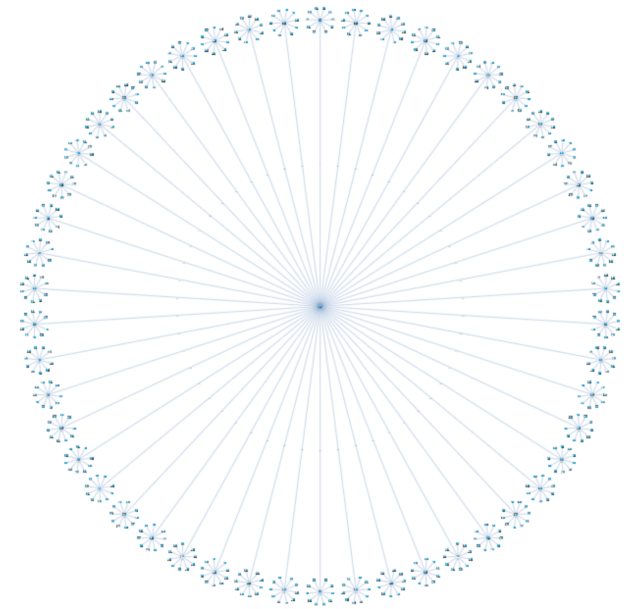
```
//创建Autolayouter并绑定DataBox
var autoLayouter = new twaver.layout.AutoLayouter(box);
//开始布局,参数分别为布局类型和回掉函数
autoLayouter.doLayout(type,callback);
```

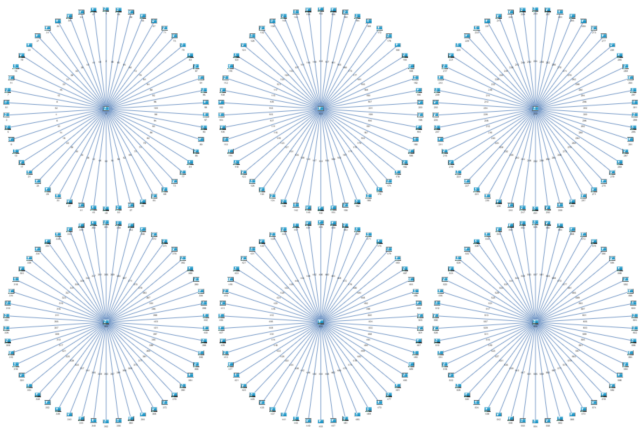
示例:

```
<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    var toolbar = document.createElement('div');
    var autoLayouter = new twaver.layout.AutoLayouter(box);
    var bounds = null;

    function init() {
        initToolbar();
        initNetwork();
        initBox();
    }

    function initNetwork(){
        var pane = new twaver.controls.BorderPane(network, toolbar);
        pane.setTopHeight(25);
        var view = pane.getView();
        view.style.left = '0px';
        view.style.top = '0px';
        view.style.right = '0px';
        view.style.bottom = '0px';
        document.body.appendChild(view);
        window.onresize = function () {
            split.invalidate();
        };
    }
    function initToolbar() {
        var zoomToOverview = addCheckBox(toolbar, false, "Overview", function () {
            doLayout(zoomToOverview.value,autoLayouterType.value);
        });

        var autoLayouterType = document.createElement('select');
        var items = ['round', 'symmetry', 'topbottom', 'bottomtop', 'leftright', 'rightleft', 'hierarchic'];
        items.forEach(function (item) {
            var option = document.createElement('option');
            option.appendChild(document.createTextNode(item));
            option.setAttribute('value', item);
            autoLayouterType.appendChild(option);
        });
        autoLayouterType.addEventListener('change', function () { doLayout(zoomToOverview.value,autoLayouterType.value); }, false);
        toolbar.appendChild(autoLayouterType);
    }

    function doLayout (overview,type) {
       if (overview) {
        autoLayouter.doLayout(type, function () {
            network.zoomOverview(false);
        });
    } else {
        autoLayouter.doLayout(type);
    }
}

function addCheckBox (div, checked, name, callback) {
    var checkBox = document.createElement('input');
```

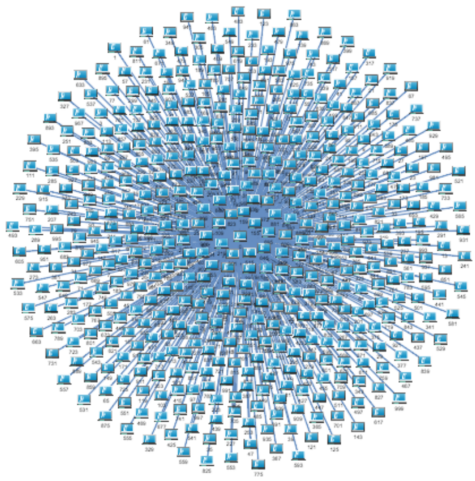
```
    checkBox.id = name;
    checkBox.type = 'checkbox';
    checkBox.style.padding = '4px 4px 4px 4px';
    checkBox.checked = checked;
    if (callback) checkBox.addEventListener('click', callback, false);
    div.appendChild(checkBox);
    var label = document.createElement('label');
    label.htmlFor = name;
    label.innerHTML = name;
    div.appendChild(label);
    return checkBox;
}

function initBox() {
    for (var i = 0, n = 1; i < n; i++) {
        var group = new twaver.Node({name: '' + box.size()});
        box.add(group);
        for (var j = 0, c = 25; j < c; j++) {
            var node = new twaver.Node({name: '' + box.size()});
            group.addChild(node);
            box.add(node);
            var link = new twaver.Link({name: '' + box.size()}, group, node);
            group.addChild(link);
            box.add(link);
        }
    }
    autoLayouter.doLayout('symmetry');
}
// ]]></script>
```

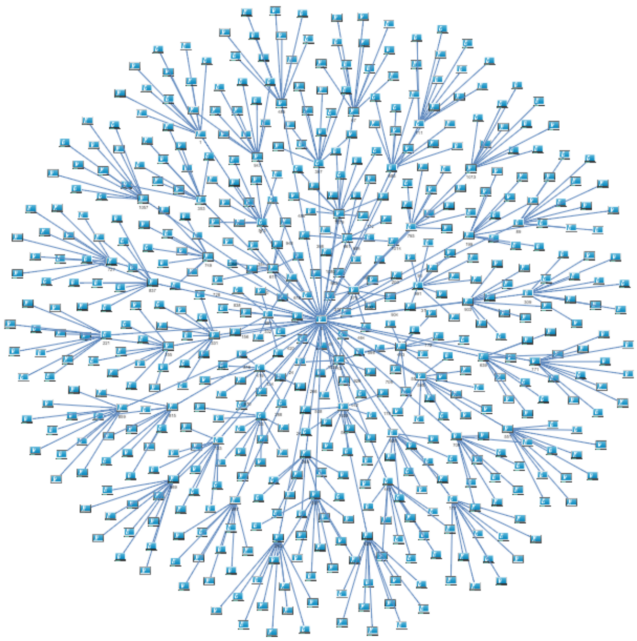
圆形布局(Round)



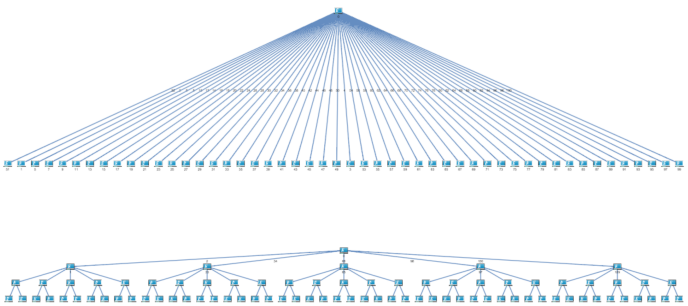


对称布局(symmetry)

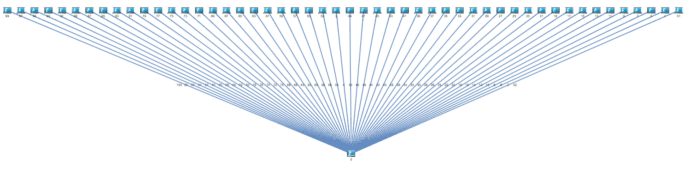




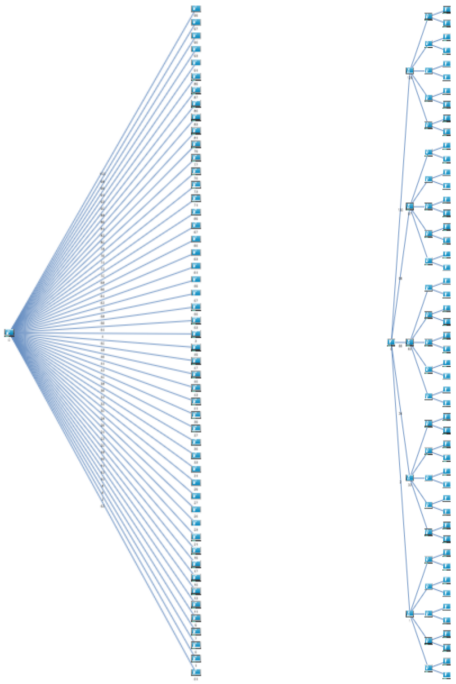
上下布局(topbottom)



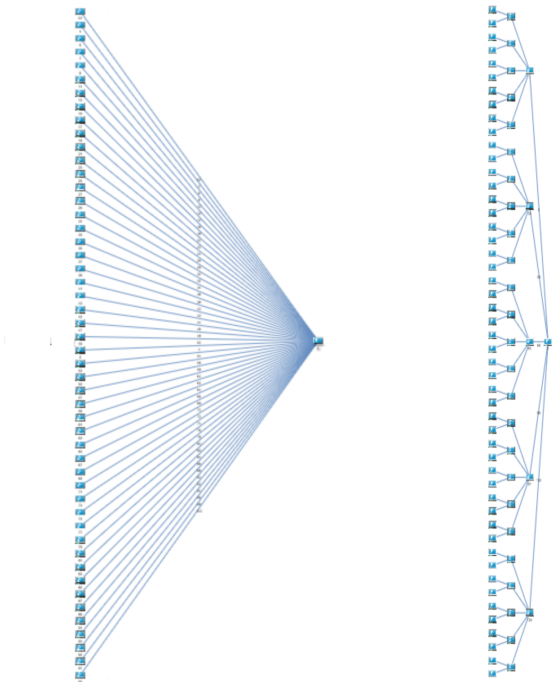
下上布局(bottomtop)



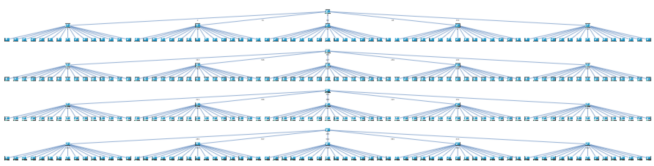
左右布局(leftright)



右左布局(rightleft)



分层布局(hierarchic)

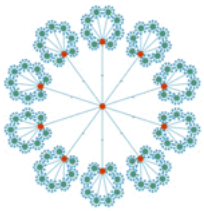


弹簧布局(SpringLayout)

```
AutoLayouter<script src="./twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    var toolbar = document.createElement('div');
    var autoLayouter = new twaver.layout.AutoLayouter(box);
    var springLayouter = new twaver.layout.SpringLayouter(network);

    function init() {
        registerImage();
        initToolbar();
        initNetwork();
        initBox();
    }

    function registerImage() {
        twaver.Util.registerImage('circle', {
            w: 50,
            h: 50,
            line: {
                width: 1,
                color: twaver.Colors.orange_dark
            },
            fill:twaver.Colors.orange_dark,
            v: [
            {
                shape: 'circle',
                cx: 0,
                cy: 0,
                r: 25
            }
            ]
        });

        twaver.Util.registerImage('circle2', {
            w: 50,
            h: 50,
            line: {
                width: 1,
                color: twaver.Colors.orange_dark
            },
            fill:twaver.Colors.green_light,
            v: [
            {
                shape: 'circle',
                cx: 0,
                cy: 0,
                r: 25
            }
            ]
        });
        twaver.Util.registerImage('circle3', {
            w: 20,
            h: 20,

            fill:twaver.Colors.blue_light,
            v: [
            {
             shape: 'circle',
             cx: 0,
             cy: 0,
             r: 10
         }
         ]
     });
    }

    function initNetwork() {
        var pane = new twaver.controls.BorderPane(network, toolbar);
        pane.setTopHeight(25);
        var view = pane.getView();
        view.style.left = '0px';
        view.style.top = '0px';
        view.style.right = '0px';
        view.style.bottom = '0px';
        document.body.appendChild(view);
        window.onresize = function () {
            pane.invalidate();
        };
        twaver.Styles.setStyle('link.color',twaver.Colors.blue_dark);
        autoLayouter.setRepulsion(0.1);
        network.setMinZoom(0.0001);
    }

    function initToolbar() {
        var repulsion = addInput("Repulsion");
        toolbar.appendChild(repulsion);
        repulsion.oninput=function(){
            autoLayouter.setRepulsion(repulsion.value);
            autoLayouter.doLayout(autoLayouterType.value);
        }
        var animation = addCheckBox(toolbar, false, "Animation", function () {
            autoLayouter.setAnimate(animation.checked);
            doLayout(zoomToOverview.value, autoLayouterType.value);
        });
        animation.checked = autoLayouter.isAnimate();

        var zoomToOverview = addCheckBox(toolbar, false, "Overview", function () {
            doLayout(zoomToOverview.value, autoLayouterType.value);
        });

        var autoLayouterType = document.createElement('select');
        var items = ['round', 'symmetry', 'topbottom', 'bottomtop', 'leftright', 'rightleft', 'hierarchic'];
        items.forEach(function (item) {
```

```
            var option = document.createElement('option');
            option.appendChild(document.createTextNode(item));
            option.setAttribute('value', item);
            autoLayouterType.appendChild(option);
        });
        autoLayouterType.addEventListener('change', function () {
            doLayout(zoomToOverview.value, autoLayouterType.value);
        }, false);
        toolbar.appendChild(autoLayouterType);

        var start = addButton(toolbar, 'Start', null, function () {
            var isRunning = springLayouter.isRunning();
            if (isRunning) {
                springLayouter.stop();
                start.value = 'Start';
            } else {
                springLayouter.start();
                start.value = 'Stop';
            }
        });
    }

    function doLayout(overview, type) {
        if (document.getElementById("Animation").checked) {
            autoLayouter.setAnimate(true);
        } else {
            autoLayouter.setAnimate(true);
        }
        if (overview) {
            autoLayouter.doLayout(type, function () {
                network.zoomOverview(false);
            });
        } else {
            autoLayouter.doLayout(type);
        }
    }

    function addCheckBox(div, checked, name, callback) {
        var checkBox = document.createElement('input');
        checkBox.id = name;
        checkBox.type = 'checkbox';
        checkBox.style.padding = '4px 4px 4px 4px';
        checkBox.checked = checked;
        if (callback) checkBox.addEventListener('click', callback, false);
        div.appendChild(checkBox);
        var label = document.createElement('label');
        label.htmlFor = name;
        label.innerHTML = name;
        div.appendChild(label);
        return checkBox;
    }

    function addButton(div, name, src, callback) {
        var button = document.createElement('input');
        button.setAttribute('type', src ? 'image' : 'button');
        button.setAttribute('title', name);
        button.style.verticalAlign = 'top';
        if (src) {
            button.style.padding = '4px 4px 4px 4px';
            if (src.indexOf('/') < 0) {
                src = '../images/toolbar/' + src + '.png';
            }
            button.setAttribute('src', src);
        } else {
            button.value = name;
        }
        button.addEventListener('click', callback, false);
        div.appendChild(button);
        return button;
    }
    function initBox() {
        for (var i = 0, n = 1; i < n; i++) {
            var center = new twaver.Node({name: '' + box.size(),id:"ID"+i});
            center.setImage("circle");
            box.add(center);
            for (var j = 0, c = 10; j < c; j++) {
                var node = new twaver.Node({name: '' + box.size()});
                node.setImage('circle');
                node.setClient('edge',true);
                box.add(node);
                var link = new twaver.Link({name: '' + box.size()}, center, node);
                box.add(link);
                for(var k=0;k<8;k++){
                    var node2 = new twaver.Node();
                    node2.setImage('circle2');
                    box.add(node2);
                    var link = new twaver.Link(node,node2);
                    box.add(link);
                    for(var m = 0;m<10;m++){
                        var node3 = new twaver.Node();
                        node3.setImage('circle3');
                        box.add(node3);
                        var link = new twaver.Link(node2,node3);
                        box.add(link);
                    }
                }
            }
        }
        autoLayouter.doLayout('round',function(){
        });
    }

    function addInput(name) {
        var input = document.createElement('input');
```

```
        if (!twaver.Util.isIE) {
            input.type = 'number';
        }
        input.id = name;
        input.min = 0;
        input.max = 10;
        input.step = 0.1;
        input.value = 0.1;

        var label = document.createElement('label');
        label.htmlFor = name;
        label.innerHTML = name;
        this.toolbar.appendChild(label);
        this.toolbar.appendChild(input);

        return input;
    }

// ]]></script>
```



自定义布局算法

Note：
自动布局的使用与数据源之间的逻辑具有一定的关系，所以并不是任何数据关系都能实现上面的布局效果，但是我们可以自定义布局算法以实现目的，下面我们介绍如何自定义布局？

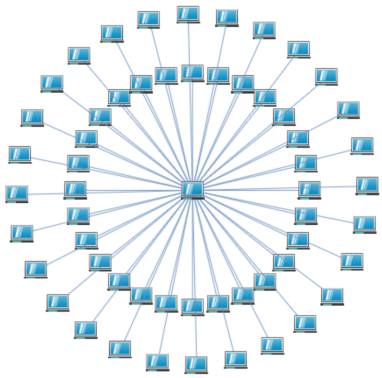
下面示例实现双层圆形布局和双层树形布局的效果。

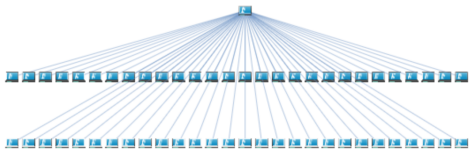
```
TWaver HTML5 - Auto Layout Demo<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDAT
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    var autoLayouter = new twaver.layout.AutoLayouter(box);

    function init() {
        var main = document.getElementById('main');
        main.appendChild(network.getView());

        network.adjustBounds({
            x: 0,
            y: 30,
            width: 1500,
            height: 700
        });

        twaver.Styles.setStyle("link.bundle.expanded", false);
        twaver.Styles.setStyle("link.width", 1);

        var self = this;
        var callback = function (e) {
            if (e.kind === 'validateEnd') {
                self.network.removeViewListener(callback);
                roundLayout(function(){network.zoomOverview()};);
            }
        };
        network.addViewListener(callback);
        initBox();
        initListener();
    }

    function initListener() {
        network.addInteractionListener(function (e) {
            if (e.kind == 'lazyMoveEnd' || e.kind == 'liveMoveEnd') {
                var currentNode = network.getElementBox().getSelectionModel().getLastData();
                var center = box.getDataById('center');
                console.log(center);
                var cx = center.getCenterLocation().x;
                var cy = center.getCenterLocation().y;
                var x = currentNode.getCenterLocation().x;
                var y = currentNode.getCenterLocation().y;


                var distance = Math.sqrt(Math.pow((cx - x), 2) + Math.pow((cy - y), 2));
                var level = currentNode.getClient('level');
                box.setClient("radius", distance / Math.pow(1.5, level));
                roundLayout();
            }
        });
    }

    function initBox() {
        var center = new twaver.Node("center");
```

```
        center.setClient('center', 'center');
        this.box.add(center);
        for (var i = 0; i < 56; i++) {
            var node = new twaver.Node();
            this.box.add(node);

            var link = new twaver.Link(node, center);
            this.box.add(link);
        }

        var sizes = [];
        this.box.forEach(function (element) {
            if (element instanceof  twaver.Node) {
                sizes.push(element.getLinks().size());
            }
        });

        Array.max = function (array) {
            return Math.max.apply(Math, array);
        }


        this.box.forEach(function (element) {
            if (element instanceof  twaver.Node) {
                if (Array.max(sizes) == element.getLinks().size()) {
                    element.setClient('center', 'center');
                }
            }
        });
    }

    function random(min, max) {
        return Math.floor(min + Math.random() * (max - min));
    }

    function getCRound(N) {
// 获取窗口高度
if (window.innerHeight)
    winHeight = window.innerHeight;
else if ((document.body) && (document.body.clientHeight))
    winHeight = document.body.clientHeight;
    //var height = network.viewRect.height;
    return parseInt(30 * N / winHeight);
}

function roundLayout(callback) {
    var datas = box.getDatas().toArray();
    var size = box.getDatas().toArray().length;
    if (window.innerWidth)
        winWidth = window.innerWidth;
    else if ((document.body) && (document.body.clientWidth))
        winWidth = document.body.clientWidth;
// 获取窗口高度
if (window.innerHeight)
    winHeight = window.innerHeight;
else if ((document.body) && (document.body.clientHeight))
    winHeight = document.body.clientHeight;
    var centerX = winWidth / 2;//圆心坐标
    var centerY = winHeight / 2;
    var radius = 0;//半径
    var N = 0; //total number of node

    box.forEach(function (data) {
        if (data.getClient('center') !== undefined) {
            data.setCenterLocation(centerX, centerY);
        }
        if (data instanceof  twaver.Node && data.getClient('center') == undefined) {
            N++;
        }
    });

    var c = getCRound(N);
    var i = 0;
    var n = parseInt(N / c);
    radius = network.viewRect.height / 2 / c - 30 / c;
    if (box.getClient("radius")) {
        radius = parseInt(box.getClient("radius"));
    }
    box.forEach(function (data) {
        if (data instanceof  twaver.Node && data.getClient('center') == undefined) {
            var e = parseInt(i / n);
            var x = centerX + (radius * Math.pow(1.5, e) * Math.cos(Math.PI * 2 / n * i + 0.2 * e));
            var y = centerY + (radius * Math.pow(1.5, e) * Math.sin(Math.PI * 2 / n * i + 0.2 * e));
            i++;
            data.setCenterLocation(x, y);
            data.setClient('level', e);
        }
    });
    if(callback){
       _twaver.callLater(callback);
   }
}

function treeLayout() {
    var datas = box.getDatas().toArray();
    var size = box.getDatas().toArray().length;
    if (window.innerWidth)
        winWidth = window.innerWidth;
    else if ((document.body) && (document.body.clientWidth))
        winWidth = document.body.clientWidth;
// 获取窗口高度
if (window.innerHeight)
    winHeight = window.innerHeight;
else if ((document.body) && (document.body.clientHeight))
```

```
    winHeight = document.body.clientHeight;
    var centerX = winWidth / 2;//圆心坐标
    var centerY = 30;
    var radius = 0;//半径
    var N = 0; //total number of node

    box.forEach(function (data) {
        if (data.getClient('center') !== undefined) {
            data.setCenterLocation(centerX, centerY);
        }
        if (data instanceof  twaver.Node && data.getClient('center') == undefined) {
            N++;
        }
    });

    var c = getCRound(N);
    var i = 0;
    var n = parseInt(N / c);
    radius = network.viewRect.height / 2 / c - 30 / c;
    box.forEach(function (data) {
        if (data instanceof  twaver.Node && data.getClient('center') == undefined) {
            var e = parseInt(i / n);
            var u = i % n;
            var x = centerX + (u - n / 2) * 40;
            var y = centerY + (radius * Math.pow(2, e));
            i++;
            data.setCenterLocation(x, y);
        }
    });
}
// ]]></script>
```

Round Tree





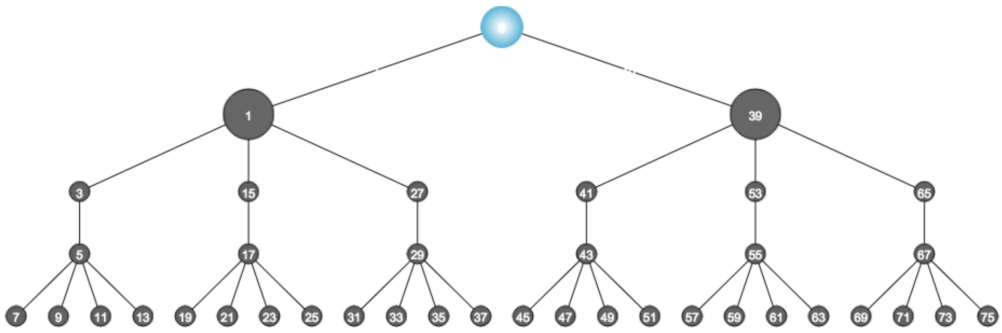
布局多样化

在布局的时候采用单一的布局算法未必会达到理想的效果，所以本节主要讲解如何使用TWaver内部特性实现布局的多样化。

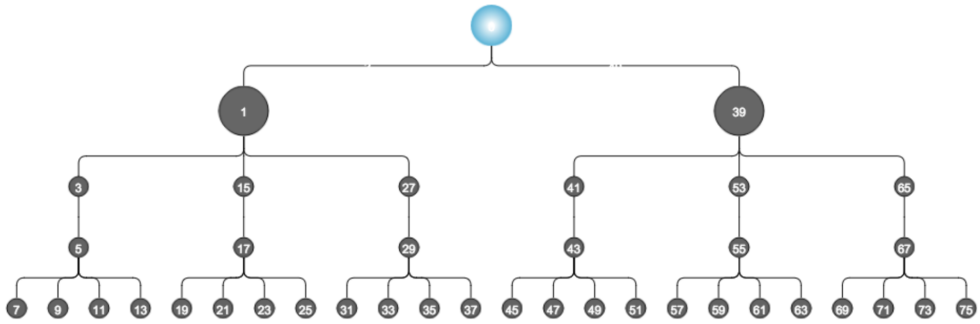
思路一:结合Link样式实现多样化

布局过程中使用单一的链路样式未免显得极为单调，我们可以尝试使用不同的连线样式，看看能实现怎样的效果？

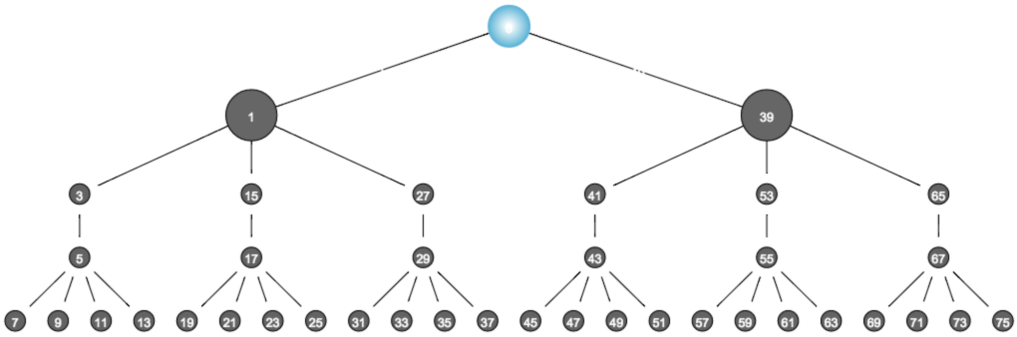
首先来看看使用link的默认样式实现布局的效果,即link.setStyle('link.type','arc');



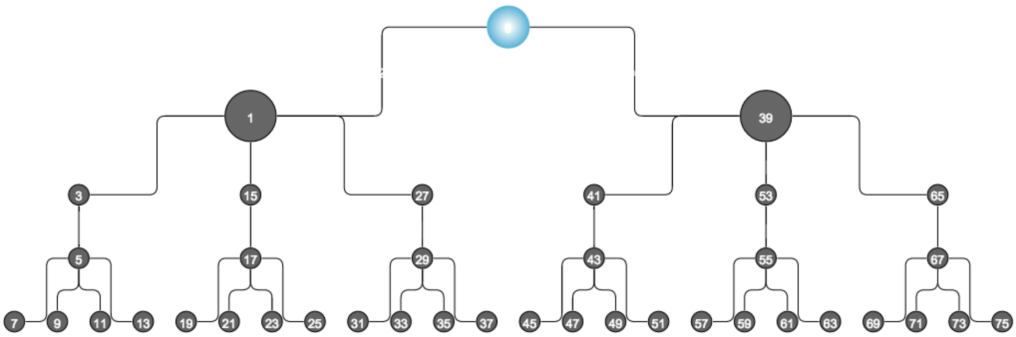
如果将link的样式更改为link.setStyle('link.type','orthogonal.vertical'),看看这样的效果:



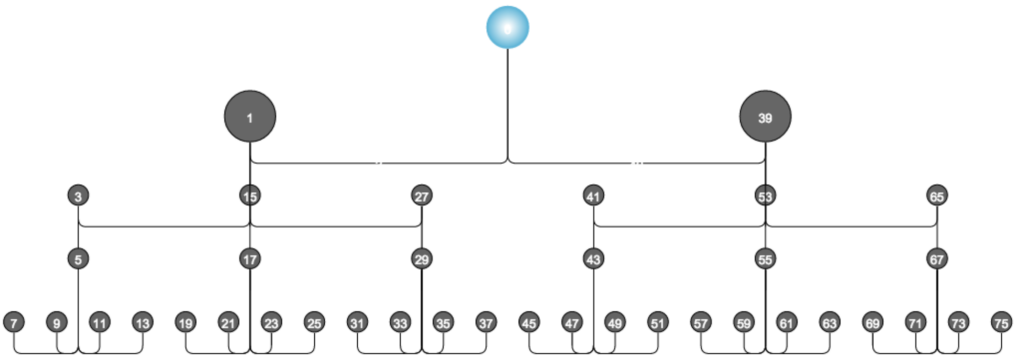
设置link.setStyle('link.type','parallel')样式:

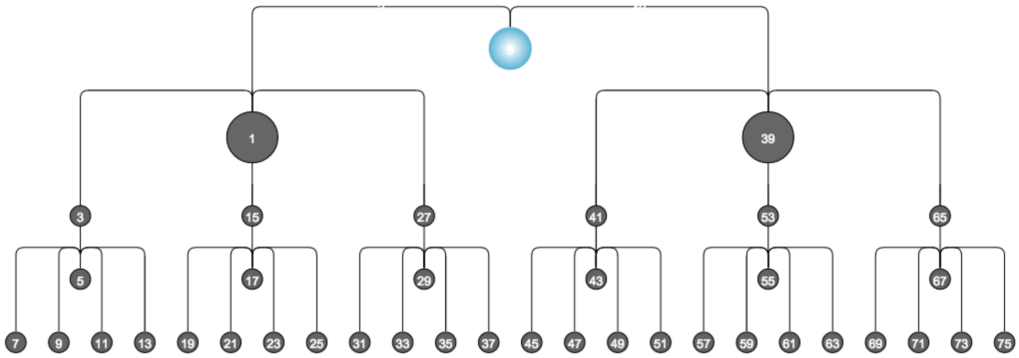


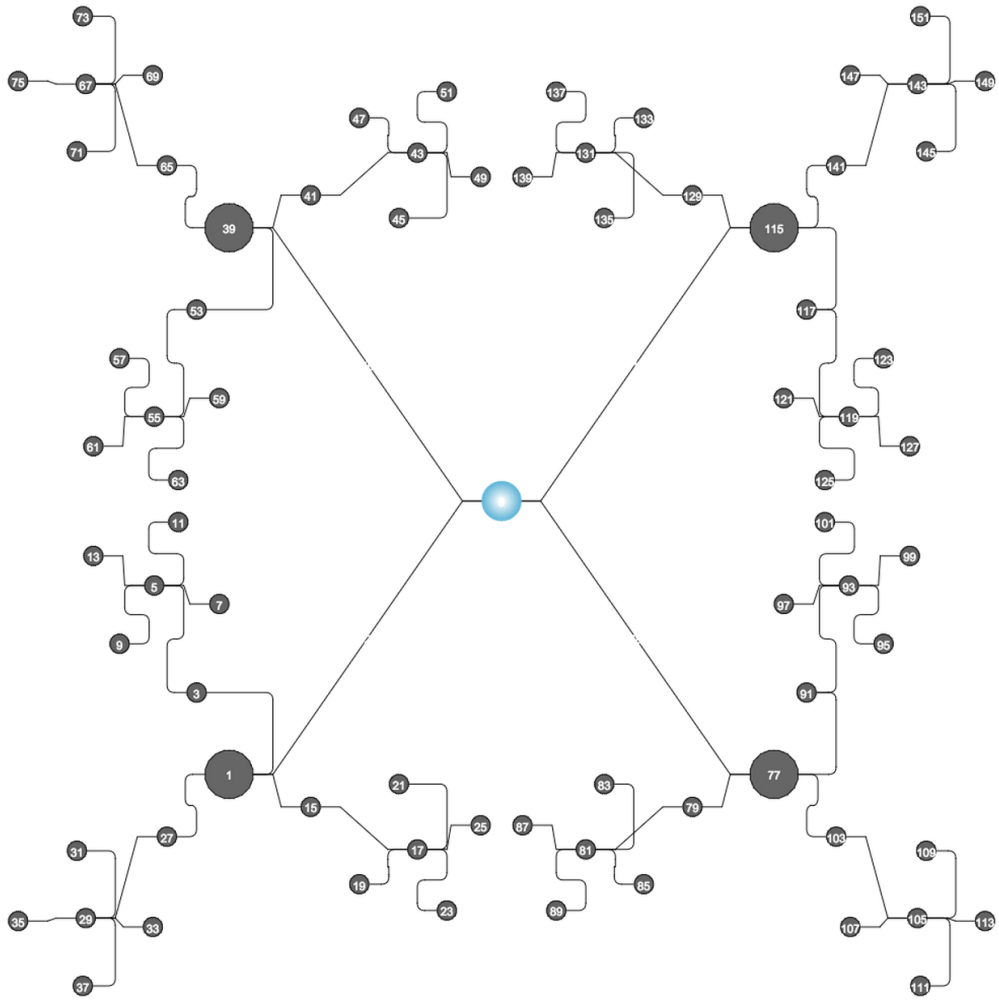
设置link.setStyle('link.type','orthogonal')样式:



还有其他效果，我们不妨可以试试：







其实我们还可以使用混合的link样式实现更加复杂的效果，总之结合link的样式可以实现非常丰富的布局效果。

思路二:局部布局

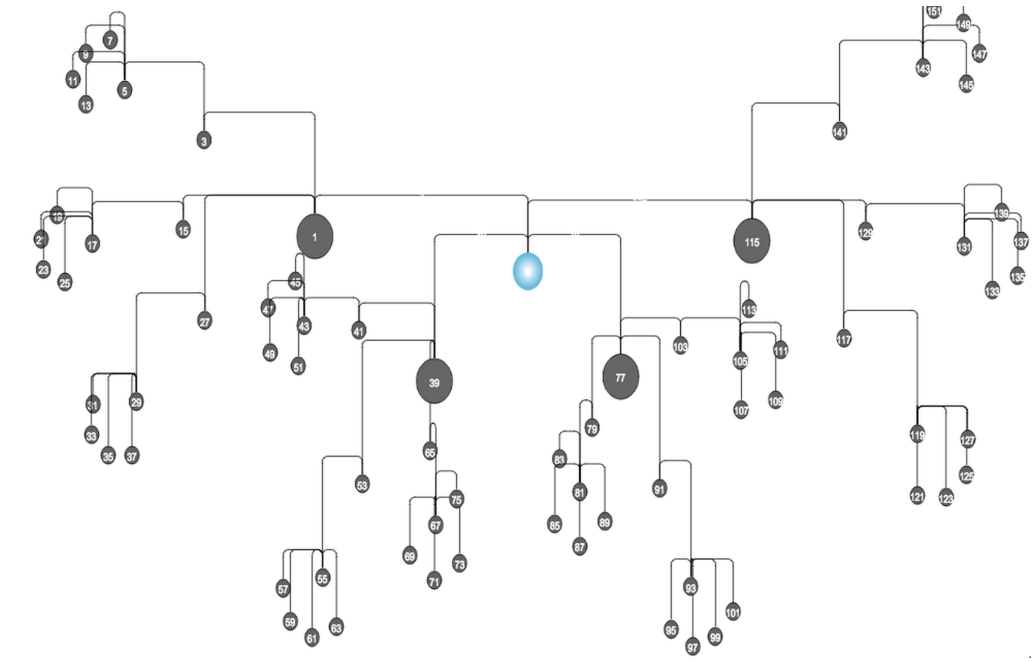
另外TWaver默认是支持局部布局的，如何使用局部布局将整体布局效果做到最优也是我们关注的重点，下面我们来试试如何使用局部布局。

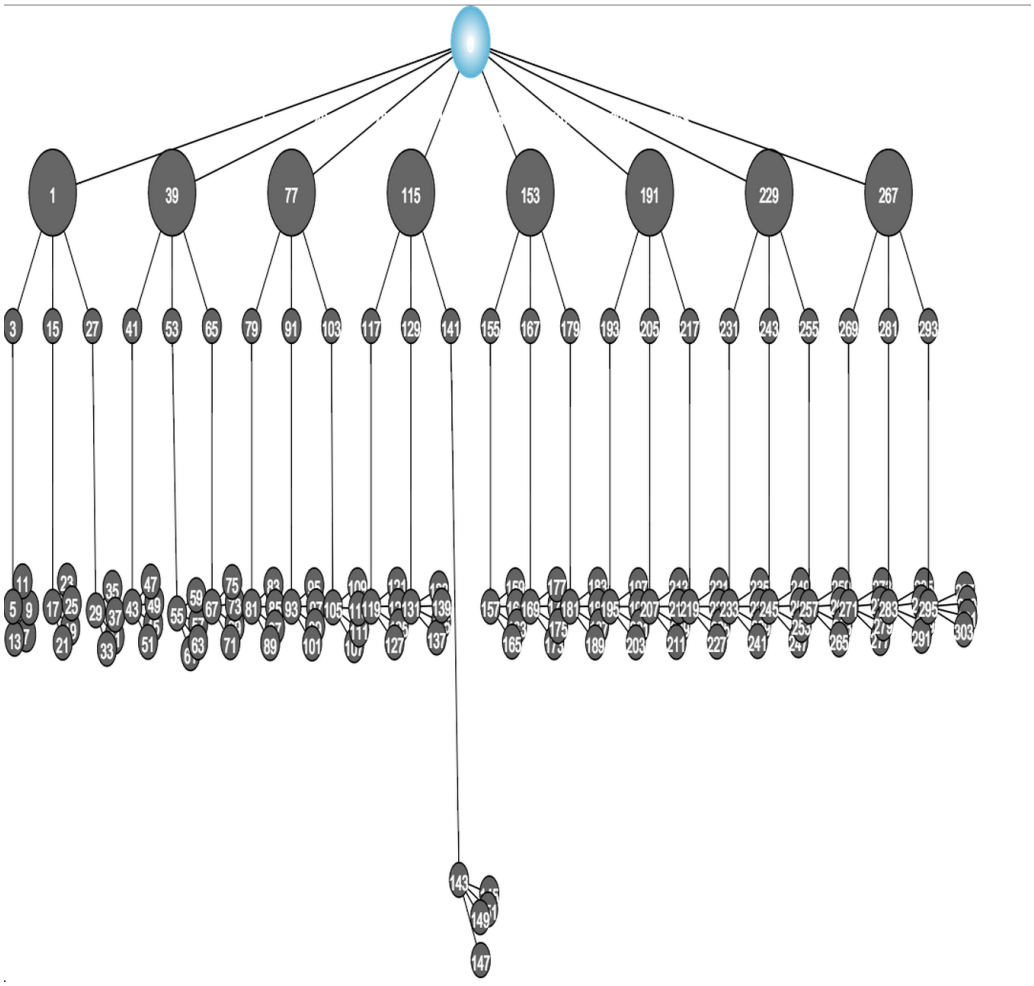
局部布局的实现原理就是设置特定的网元参加特定的自动布局算法，实现接口如下:

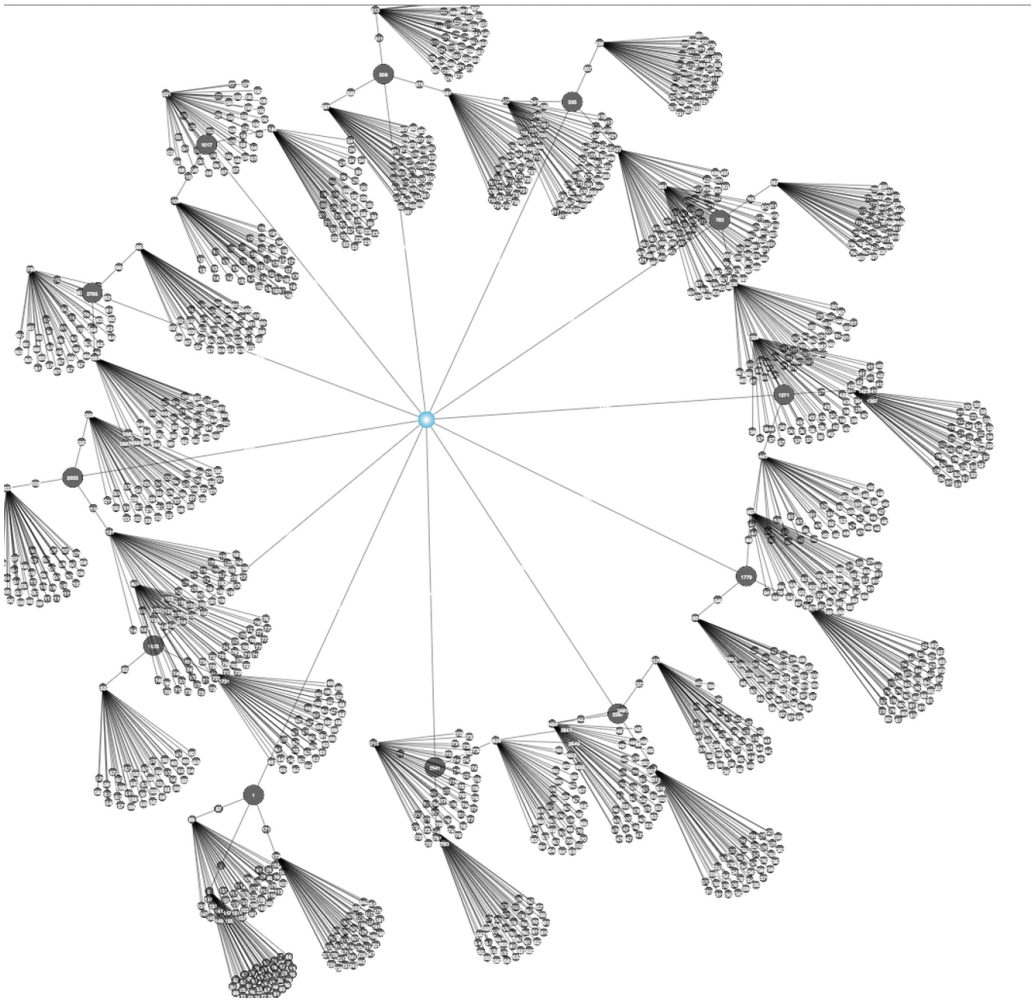
```
 autoLayouter.getElements = function () {
//过滤条件可自行设置
            var elements = new twaver.List();
            box.forEach(function(element){
                if(element.getClient('autoLayouter')){
                    elements.add(element);
                }
            })
            return elements;
        }
```

思路三:结合多种布局算法

TWaver不仅支持局部布局，而且支持同一场景中使用多种布局算法，最常用的莫过于Autolayouter算法和Spring布局算法相结合，首先来看看效果。







使用方法自然很简单，就是同时打开多个布局算法，如:

```
var autoLayouter = new twaver.layout.AutoLayouter(box);
    var autoLayouter2 = new twaver.layout.AutoLayouter(box);
    var springLayouter = new twaver.layout.SpringLayouter(network);
...
 autoLayouter2.doLayout(evt.target.value,function(){
    autoLayouter.doLayout('round',function(){
       network.zoomOverview(true);
    });
});
springLayouter.start();
```

布局的样式各种各样，实现方法也多种多样，将上述三种思路混合，也许又可以产生更加丰富的布局效果。

常见问题

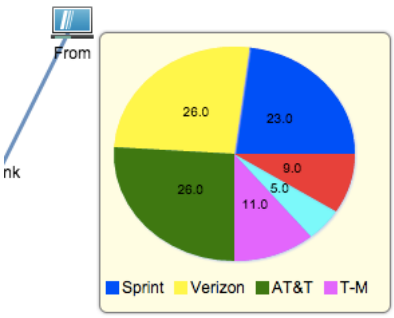
显示ToolTip

```
network.setToolTipEnabled(true);
```

Network中ToolTip显示Chart

```
<script src="../../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[

      var box = new twaver.ElementBox();
      var network = new twaver.network.Network(box);
      var toolbar = document.createElement('div');

      function init () {
         initToolbar();
         var pane = new twaver.controls.BorderPane(network, toolbar);
         pane.setTopHeight(25);
         var view = pane.getView();
         view.style.left = '0px';
         view.style.top = '0px';
         view.style.right = '0px';
         view.style.bottom = '0px';
         document.body.appendChild(view);
         window.onresize = function () {
            pane.invalidate();
         };

         network.getToolTip = function (data) {
            if (!data.getToolTip()) {
```

```
                var pieChart = new twaver.charts.PieChart();
                pieChart.formatValueText = function (value, data) {
                   return value.toFixed(1);
                };
                createPieChartElement("Sprint", 23, 'BLUE', pieChart);
                createPieChartElement("Verizon", 26, 'YELLOW', pieChart);
                createPieChartElement("AT&T", 26, 'GREEN', pieChart);
                createPieChartElement("T-Mobile", 11, 'MAGENTA', pieChart);
                createPieChartElement("Alltel", 5, 'CYAN', pieChart);
                createPieChartElement("Rest", 9, 'RED', pieChart);
                var pieChartPane = new twaver.charts.ChartPane(pieChart);
                pieChartPane.adjustBounds({ x: 0, y: 0, width: 200, height: 200 });
                data.setToolTip(pieChartPane.getView());
                twaver.Util.getToolTipDiv().style.width = '200px';
                twaver.Util.getToolTipDiv().style.height = '200px';
            }
            return data.getToolTip();
         };
         initBox();
      }

      function createPieChartElement (name, value, color, pieChart) {
        var element = new twaver.Element();
        element.setName(name);
        element.setStyle('chart.value', value);
        element.setStyle('chart.color', color);
        pieChart.getDataBox().add(element);
        return element;
    }

      function initBox () {
          from = new twaver.Node();
          from.setName('From');
          from.setLocation(400, 100);
          box.add(from);

          var to = new twaver.Node();
          to.setName('To');
          to.setLocation(300, 300);
          box.add(to);

          var link = new twaver.Link(from, to);
          link.setName('Link');
          box.add(link);
      }

      function initToolbar() {
         addButton(toolbar, 'Zoom In', function () {
            network.zoomIn();
         });
         addButton(toolbar, 'Zoom Out', function () {
            network.zoomOut();
         });
         addButton(toolbar, 'Zoom Overview', function () {
            network.zoomOverview();
         });
         addButton(toolbar, 'Zoom Reset', function () {
            network.zoomReset();
         });
      }

      function addButton(toolbar, label, handler) {
        var button = document.createElement('input');
         button.type = 'button';
         button.value =  label;
         button.onclick = handler;
         toolbar.appendChild(button);
      }


// ]]></script>
```



Link显示流动效果

```
network.setLinkFlowEnabled(true);
```

显示滚动条

```
network.setHScrollBarVisible(true);
network.setVScrollBarVisible(true);
```

判断鼠标下网元是否为附件？

```
 network.getView().addEventListener('mousedown', function (e) {
     var target = self.network.hitTest(e);
     if (target) {
         if (target instanceof twaver.network.ElementUI) {
             console.log('clicked ElementUI');
         }
         if (target instanceof twaver.network.LabelAttachment) {
             console.log('clicked LabelAttachment');
         }
     }
 });
```

Network导出成图片

```
 var canvas;
     if (network.getCanvasSize) {
         //canvas = network.toCanvas(network.getCanvasSize().width/5, network.getCanvasSize().height/5);
         canvas = network.toCanvas(1024,1024);
     }else{
         canvas = network.toCanvas(network.getView().scrollWidth, network.getView().scrollHeight);
     }
     if(twaver.Util.isIE) {
         var w = window.open();
         w.document.open();
         w.document.write("<img src="&amp;quot; + canvas.toDataURL() + &amp;quot;" alt="" />");
         w.document.close();
     }else{
         window.open(canvas.toDataURL(), 'network.png');
     }
```

图片不渲染告警颜色

```
//network.getInnterColor返回网元的渲染颜色,默认如下:
getInnerColor: function (data) {
    if (data.IElement) {
        var severity = data.getAlarmState().getHighestNativeAlarmSeverity();
        if (severity) {
            return severity.color;
        }
        return data.getStyle('inner.color');
    }
    return null;
}
//实现不渲染告警颜色
getInnerColor: function (data) {
    return null;
}
```

增加网元后，保持原布局位置不变

这块内容可以参考我们的社区论坛。

鹰眼可视化视图组件(OverView)

所谓鹰眼就是一个缩略图，内嵌矩形框，显示当前Network中的数据，拖动矩形框可以改变当前Network显示的位置，改变矩形框的大小，可以改变当前Network视图区域大小，从而起到导航的作用。

使用方法:

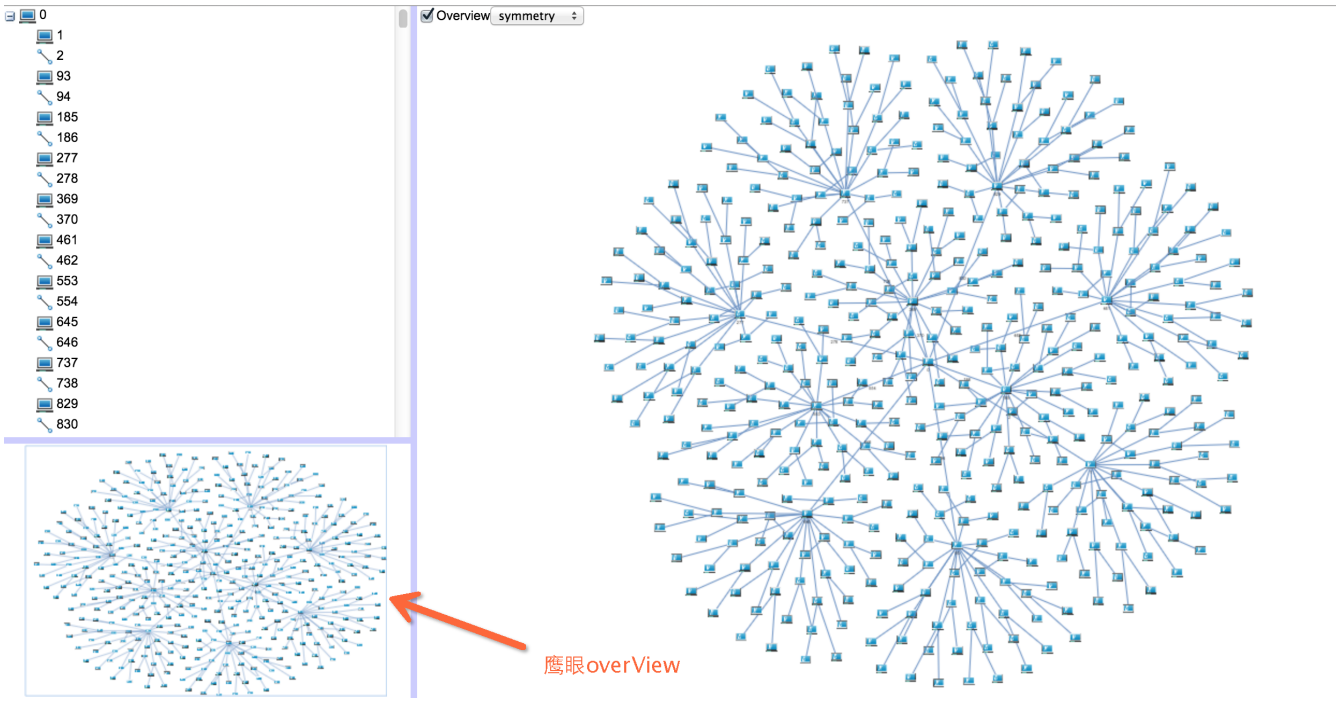
```
var overview = new twaver.vector.Overview(network);
var networkPane = new twaver.controls.BorderPane(network, toolbar);
```

```
        networkPane.setTopHeight(25);
        var leftPane = new twaver.controls.SplitPane(tree, overview, 'vertical', 0.7);
        var pane = new twaver.controls.SplitPane(leftPane, networkPane, 'horizontal', 0.3);
        var view = pane.getView();
        view.style.left = '0px';
        view.style.top = '0px';
        view.style.right = '0px';
        view.style.bottom = '0px';
        document.body.appendChild(view);
```

示例:

```
<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    var tree = new twaver.controls.Tree(box);
    var toolbar = document.createElement('div');
    var autoLayouter = new twaver.layout.AutoLayouter(box);
    var bounds = null;
    var overview = new twaver.vector.Overview(network);

    function init() {
        initToolbar();
        var networkPane = new twaver.controls.BorderPane(network, toolbar);
        networkPane.setTopHeight(25);
        var leftPane = new twaver.controls.SplitPane(tree, overview, 'vertical', 0.7);
        // var split = new twaver.controls.SplitPane(tree, pane, null, 0.3);
        var pane = new twaver.controls.SplitPane(leftPane, networkPane, 'horizontal', 0.3);
        var view = pane.getView();
        view.style.left = '0px';
        view.style.top = '0px';
        view.style.right = '0px';
        view.style.bottom = '0px';
        document.body.appendChild(view);
        window.onresize = function () {
            split.invalidate();
        };
        initBox();
        tree.expandAll();
    }

    function initToolbar() {
        var zoomToOverview = addCheckBox(toolbar, false, "Overview", function () {
            doLayout(zoomToOverview.value,autoLayouterType.value);
        });

        var autoLayouterType = document.createElement('select');
        var items = ['round', 'symmetry', 'topbottom', 'bottomtop', 'leftright', 'rightleft', 'hierarchic'];
        items.forEach(function (item) {
            var option = document.createElement('option');
            option.appendChild(document.createTextNode(item));
            option.setAttribute('value', item);
            autoLayouterType.appendChild(option);
        });
        autoLayouterType.addEventListener('change', function () { doLayout(zoomToOverview.value,autoLayouterType.value); }, false);
        toolbar.appendChild(autoLayouterType);

    }

    function doLayout (overview,type) {
         if (overview) {
            autoLayouter.doLayout(type, function () {
                network.zoomOverview(false);
            });
        } else {
            autoLayouter.doLayout(type);
        }
    }

    function addCheckBox (div, checked, name, callback) {
        var checkBox = document.createElement('input');
        checkBox.id = name;
        checkBox.type = 'checkbox';
        checkBox.style.padding = '4px 4px 4px 4px';
        checkBox.checked = checked;
        if (callback) checkBox.addEventListener('click', callback, false);
        div.appendChild(checkBox);
        var label = document.createElement('label');
        label.htmlFor = name;
        label.innerHTML = name;
        div.appendChild(label);
        return checkBox;
    }
    function initBox() {
        for (var i = 0, n = 1; i < n; i++) {
            var group = new twaver.Node({name: '' + box.size()});
            box.add(group);
            for (var j = 0, c = 10; j < c; j++) {
                var node = new twaver.Node({name: '' + box.size()});
                group.addChild(node);
                box.add(node);
                var link = new twaver.Link({name: '' + box.size()}, group, node);
                group.addChild(link);
                box.add(link);
                for(var k=0;k<15;k++){
                    var child1 = new twaver.Node();
                    box.add(child1);
                    var link1 = new twaver.Link(node,child1);
                    box.add(link1);
                    for(var m=0;m<2;m++){
                    var child2 = new twaver.Node();
                    box.add(child2);
```

```
                    var link2 = new twaver.Link(child2,child1);
                    box.add(link2);
                }
                }
            }
        }
        autoLayouter.doLayout('symmetry');
    }


// ]]></script>
```



常见问题

设置鹰眼视图的尺寸

```
    //如果是将overview放在布局容器中，大小会自动调整
    var overview = new twaver.network.Overview(this.network);
    var leftSplit = new SplitPane(this.tree, overview, 'vertical', 0.7);
    //否则调用adjustBounds();
    overview.adjustBounds({x:0, y:0, width:200, height:200});
```

树可视化视图组件(Tree)

twaver.controls.Tree组件用于展示DataBox中数据元素的层次结构，树图上节点的父子关系由数据元素的父子关系决定，书节点的先后位置由数据元素在其父节点的位置所决定。

创建树组件

树组件的创建与拓扑图类似，也是通过API创建，构造函数可以传入databox，实现与这个数据容器的关联。

```
var tree = new Twaver.controls.Tree(data box);
```

```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var tree = new twaver.controls.Tree(box);

        function init() {
            initTreeView();
            initDataBox();
            registerImage();
        }

        function initTreeView() {
           var treeDom = tree.getView();
            treeDom.style.width = "100%";
            treeDom.style.height = "100%";
            document.body.appendChild(treeDom);
            tree.expandAll();
        }

        function initDataBox() {
            var parent = new twaver.Group();
            parent.setName('group');
```

```
            box.add(parent);

            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                }
            });
            from.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.CRITICAL);
            box.add(from);

            var to = new twaver.Node({
                name: 'to',
                location: {
                    x: 220,
                    y: 160
                }
            });
            to.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.WARNING);
            box.add(to);

            var link = new twaver.Link(from, to);
            link.setName('link');
            box.add(link);

            parent.addChild(from);
            parent.addChild(to);
            parent.addChild(link);
        }

// ]]></script>
```

树节点的样式

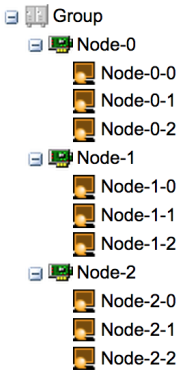
TWaver HTML5的树节点由图标和文字组成，其中图标支持染色和冒泡挂载的功能。

树节点样式设置方法：

```
//设置图标
data.setIcon('iconName')
//设置节点标签
data.setName('001')
```

```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var tree = new twaver.controls.Tree(box);

        function init() {
            initTreeView();
            initDataBox();
        }

        function initTreeView() {
            var treeDom = tree.getView();
            treeDom.style.width = "100%";
            treeDom.style.height = "100%";
            document.body.appendChild(treeDom);
            tree.expandAll();
        }

        function initDataBox() {
            var group = new twaver.Group();
            group.setName('Group');
            group.setIcon('../images/group.png');
            box.add(group);

            for (var i = 0; i < 3; i++) {
                var node1 = new twaver.Node({
                    name:'Node-'+i,
                    location:{
                        x:100,
                        y:200
                    },
                    icon:'../images/node1.png'
                });
                node1.setParent(group);
                box.add(node1);
                for(var j=0;j<3;j++){
                    var node2 = new twaver.Node();
                    node2.setName('Node-'+i+'-'+j);
```

```
                    node2.getAlarmState().setNewAlarmCount(twaver.AlarmSeverity.MAJOR,1);
                    node2.setParent(node1);
                    box.add(node2);
                }
            }

        }

// ]]></script>
```



排序

树节点的层次结构由节点的父子关系，以及孩子的次序所决定，我们可以改变节点的父节点，也可以调整孩子的先后次序，从而实现树结构的调整。我们也可以通过设置比较器进行排序。

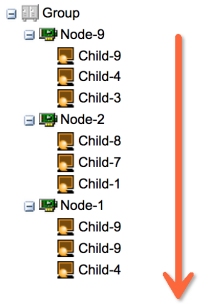
```
tree.setSortFunction(compareFunction);
```

```
tree.setSortFunction(function(d1,d2){
    return d1.getName() &gt; d2.getName() ? -1:1;
});
```

下面的例子我们将创建多个随机名称的树节点，并对其进行降序排序。

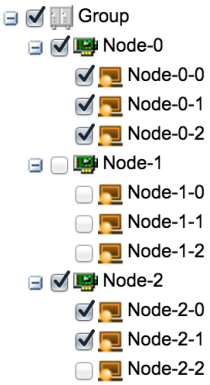
```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var tree = new twaver.controls.Tree(box);

        function init() {
            initTreeView();
            initDataBox();
        }

        function initTreeView() {
            var treeDom = tree.getView();
            treeDom.style.width = "100%";
            treeDom.style.height = "100%";
            document.body.appendChild(treeDom);
            tree.expandAll();

            tree.setSortFunction(function (d1, d2) {
                return d1.getName() > d2.getName() ? -1 : 1;
            });
        }

        function initDataBox() {
            var group = new twaver.Group();
            group.setName('Group');
            group.setIcon('../images/group.png');
            box.add(group);

            for (var i = 0; i < 3; i++) {
                var node1 = new twaver.Node({
                    name: 'Node-' + Math.floor(Math.random() * 10),
                    location: {
                        x: 100,
                        y: 200
                    },
                    icon: '../images/node1.png'
                });
                node1.setParent(group);
                box.add(node1);
                for (var j = 0; j < 3; j++) {
                    var node2 = new twaver.Node();
                    node2.setName('Child-' + Math.floor(Math.random() * 10));
                    node2.getAlarmState().setNewAlarmCount(twaver.AlarmSeverity.MAJOR, 1);
                    node2.setParent(node1);
                    box.add(node2);
                }
            }

        }

```

```
// ]]></script>
```



另外树组件还支持勾选模式，通过树节点前面的勾选框表示节点的选中状态。设置勾选模式的方法如下：

```
/**
 * Tree提供四种勾选模式:
 * default:默认勾选模式,所有节点均可以勾选
 * children:孩子勾选模式,所有的分支节点可以勾选
 * descendant:子孙勾选模式,父节点被勾选,则其子孙节点自动被勾选
 * descendantAncestor:子孙祖先勾选模式,父节点被勾选，则子孙节点自动被勾选，子孙节点被勾选，则其祖先节点也自动勾选
 */
Tree.setCheckMode(v);//如tree.setCheckMode('descendantAncestor');
```



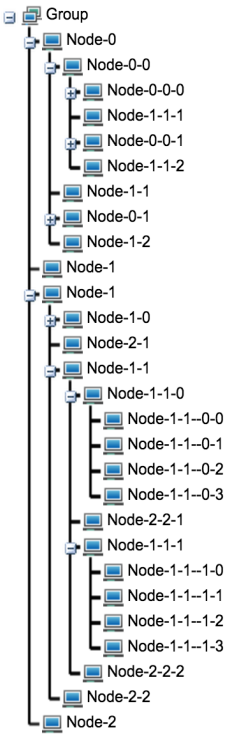
Tree引导线

在树中显示引导线，不仅美观，而且使得层次感清晰。TWaver中的Tree加入引导线的方法如下：

```
tree.setLineType(type)//type的值为'none'、'dotted'、'solid'
```

设置线条样式方法有：

```
tree.setLineType('solid');//线条类型
tree.setLineColor('#000000');//线条颜色
tree.setLineAlpha(1);//线条透明度
tree.setLineThickness(1);//线条厚度
tree.setLineDash([10,2]);//线条虚线样式，在linetype为'dotted'情况下有效。
```



tree	Name	Id	Icon	Inner Color
Group	Group	EC1146461…	group_icon	
Node-0	Node-0	DBBAED71…	node_icon	
Node-0-0	Node-0-0	76BD49EE…	node_icon	
Node-0-0-0	Node-0-0-0	B4D6AA0C…	node_icon	
Node-1-1-1	Node-1-1-1	ECD1A241…	node_icon	
Node-0-0-1	Node-0-0-1	263E6B4E3…	node_icon	
Node-1-1-2	Node-1-1-2	8A205FC6…	node_icon	
Node-1-1	Node-1-1	99CFFBAF…	node_icon	
Node-0-1	Node-0-1	18544D4D5…	node_icon	
Node-1-2	Node-1-2	309360C2F…	node_icon	
Node-1	Node-1	6CE0B8408…	node_icon	
Node-1	Node-1	0F6E84D3…	node_icon	
Node-1-0	Node-1-0	52CB1B6C…	node_icon	
Node-2-1	Node-2-1	843BDD7A…	node_icon	
Node-1-1	Node-1-1	A305AC472…	node_icon	
Node-1-1-0	Node-1-1-0	715C69C96…	node_icon	
Node-1-1--0-0	Node-1-1--0…	DD7D9596…	node_icon	
Node-1-1--0-1	Node-1-1--0…	FE27F2595…	node_icon	
Node-1-1--0-2	Node-1-1--0…	0E59CBF7…	node_icon	
Node-1-1--0-3	Node-1-1--0…	AC9BF329…	node_icon	
Node-2-2-1	Node-2-2-1	35E254A49…	node_icon	
Node-1-1-1	Node-1-1-1	0E933CDC…	node_icon	
Node-1-1--1-0	Node-1-1--1…	6CA27ADC…	node_icon	
Node-1-1--1-1	Node-1-1--1…	1C35F1B09…	node_icon	
Node-1-1--1-2	Node-1-1--1…	4325D4FF4…	node_icon	
Node-1-1--1-3	Node-1-1--1…	A60802460…	node_icon	
Node-2-2-2	Node-2-2-2	0B1F0AB86…	node_icon	
Node-2-2	Node-2-2	7BCB51CE…	node_icon	
Node-2	Node-2	527886F5B…	node_icon	

常见问题

修改tree展开闭合图标

twaver默认情况下，tree的toggle图标定义如下:

```
tree.getToggleImage = function (data) {
        if (data.getChildrenSize() &gt; 0) {
            return this.isExpanded(data) ? this._expandIcon : this._collapseIcon;
        }
        return null;
    };
```

我们欲实现自定义的图标样式，以及控制网元是否显示Toggle图标，只需要重新实现该方法。

示例：tree中所有行均显示Toggle图标。

```
p.getToggleImage = function (data) {
        return twaver.Defaults.TREE_COLLAPSE_ICON;
    };
```

当然我们也可以使用自己的图标，记得要先注册图标。

修改tree选中颜色

```
tree.getSelectColor = function() {
     return "#FF0000";
}
```

tree监听事件

```
tree.addInteractionListener(function(evt) {
     console.log(evt);
     if(evt.kind == "doubleClick") {
          console.log(evt.data.getName());
     }
   });
```

控制显示复选框

```
//重写isCheckabke方法实现自定义复选框显示状态
tree.isCheckable = function(element) {
     if(element.getChildrenSize() == 0) {
          return true;
     }
     return false;};
```

是否显示当前被选中的网元

```
tree.setMakeVisibleOnSelected(false);
```

表格可视化视图组件(Table)

twaver.controls.Table组件实现了与DataBox的数据绑定，用以展示容器内元素属性等信息，每行对应一个数据元素。本章节主要介绍表格组件的使用，表格列的定义，过滤器以及排序功能的使用。众所周知，一张完整的表格要包括表头和内容两部分,所以TWaver的表格也要包含这两部分，由于TWaver的强大封装，简化创建接口，所以创建一张表格只需要执行下面几步:

```
//1.创建table并绑定DataBox
var table = new twaver.controls.Table(box);
//2.创建tablepane并绑定table数据内容,第二个参数为表头，默认不需要传入参数
 var tablePane = new twaver.controls.TablePane(table,tableHeader);
//3.布局表格面板
var tableDom = tablePane.getView();
tableDom.style.width = "100%";
tableDom.style.height = "100%";
document.body.appendChild(tableDom);
//4.向ColumnBox中添加数据,ColumnBox继承于DataBox,用来管理表格中的数据,详细可参考数据容器一章
var column = new twaver.Column(name);
column.setName(name);
column.setPropertyName(propetyName);
column.setPropertyType(propertyType);
table.getColumnBox().add(column);
```

使用表格

```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);
        var table = new twaver.controls.Table(box);

        function init() {
            initTable();
            initDataBox();
        }

        function initTable(){
            var tablePane = new twaver.controls.TablePane(table);
            var tableDom = tablePane.getView();
            tableDom.style.width = "100%";
            tableDom.style.height = "100%";
            document.body.appendChild(tableDom);
            window.onresize = function(){
                tablePane.invalidate();
            }
            createColumn(table,'Name','name','accessor','string');
            createColumn(table,'Id','id','accessor','string');
            createColumn(table,'Icon','icon','accessor');
        }
        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }

        function initDataBox() {
            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                }
            });
            box.add(from);

            var to = new twaver.Node({
                name: 'to',
                location: {
                    x: 220,
                    y: 160
                }
            });
            box.add(to);

            var link = new twaver.Link(from, to);
            box.add(link);

            var i=100;
            while(i-- > 0){
                var data = new twaver.Node();
                data.setName("TWaver-"+i);
                data.setParent(from);

                box.add(data);
            }

        }

        function createColumn(table,name,propetyName,propertyType,valueType){
            var column = new twaver.Column(name);
            column.setName(name);
            column.setPropertyName(propetyName);
            column.setPropertyType(propertyType);
            if(valueType){
                column.setValueType(valueType);
            }
            table.getColumnBox().add(column);
            return column;
```

```
        }

// ]]></script>
```

Name	Id	Icon
from	2CE914E9E8F1482A8449DBAA2C7C0DC7	node_icon
TWaver-99	267DB086D4C04377AB43BA9737599A1F	node_icon
TWaver-98	2EAE0E4BF7A642F59E295D31928B3C19	node_icon
TWaver-97	3D8ABD0C2F9B419F9E8072433BDC204E	node_icon
TWaver-96	9DC6A4C3E0604F7D8B7342026DD87A23	node_icon
TWaver-95	83098F68230D420082D6A04EEC2635CE	node_icon
TWaver-94	1C7A542387964B5E9A2F5766755EEB58	node_icon
TWaver-93	E811F0905B3D4113979389ED4E4BB997	node_icon
TWaver-92	D9D555566682486191FFC14097BC957B	node_icon
TWaver-91	64048BD0E53C48AF8F8239E4559E241A	node_icon
TWaver-90	20EC839F6465466F8C36DB25B904CA50	node_icon
TWaver-89	5A96CBD059EB421382B8F11ADB4EC4BE	node_icon
TWaver-88	5B7A33F771C7411E873B5D427BC635EB	node_icon
TWaver-87	3804BD80E74143298703784659A09737	node_icon
TWaver-86	75E8385E85D24863873B2536650BF288	node_icon

表格列的设置

TWaver HTML5中的表格组件用于展示DataBox中元素属性，每行对应了一个数据元素，每一列对应元素的一种属性。表格列的实现类是twaver.Column,首先我们需要设置该列关联数据元素的哪个属性,余下还可以设置列头名称,列头呈现方式,单元格如何呈现等等。

```
/**
 *Column表格列的设置
 * 列头名称:
 * 属性名称:
 * 属性类型: field(如node.name)、accessor(如node.getName(),node.setName())、style(node.getStyle()、
 *           node.setStyle()、node.s())、client(node.getClient()、node.setClient());
 * 属性值类型:
 */
var column = new twaver.Column();
column.setName("Name");//设置列头名称
column.setPropertyName('name');//设置属性名
column.setPropertyType('accessor');//设置属性类型,默认就是'accessor'类型
column.setValueType('string');//设置值类型,默认就是'string'类型
table.getClomunBox().add(column);//设置到表格列容器
```

```
//再举个style属性列的例子，定义一列显示数据元素的'inner.color'样式属性，并设置不可编辑
var column = new twaver.Column();
column.setName('Inner Color');
column.setPropertyName('inner.color');
column.setPropertyType('style');
column.setValueType('color');
table.getColumnBox().add(column);
```

```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);
        var table = new twaver.controls.Table(box);

        function init() {
            initTable();
            initDataBox();
        }

        function initTable() {
            var tablePane = new twaver.controls.TablePane(table);
            var tableDom = tablePane.getView();
            tableDom.style.width = "100%";
            tableDom.style.height = "100%";
            document.body.appendChild(tableDom);
            window.onresize = function () {
                tablePane.invalidate();
            }
            createColumn(table, 'Name', 'name', 'accessor', 'string');
            createColumn(table, 'Id', 'id', 'accessor', 'string');
            createColumn(table, 'Icon', 'icon', 'accessor');
            createColumn(table, 'Inner Color', 'inner.color', 'style','color');
        }
        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }

        function initDataBox() {
            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                }
            });
            box.add(from);
```

```
40
41              var to = new twaver.Node({
42                  name: 'to',
43                  location: {
44                      x: 220,
45                      y: 160
46                  }
47              });
48              box.add(to);
49
50              var link = new twaver.Link(from, to);
51              box.add(link);
52
53              var i = 100;
54              while (i-- > 0) {
55                  var data = new twaver.Node();
56                  data.setName("TWaver-" + i);
57                  data.setParent(from);
58                  data.s('inner.color', randomColor());
59                  box.add(data);
60              }
61
62          }
63
64          function createColumn(table, name, propetyName, propertyType, valueType) {
65              var column = new twaver.Column(name);
66              column.setName(name);
67              column.setPropertyName(propetyName);
68              column.setPropertyType(propertyType);
69              if (valueType) {
70                  column.setValueType(valueType);
71              }
72              table.getColumnBox().add(column);
73              return column;
74          }
75
76          function randomColor() {
77              var r = randomInt(255);
78              var g = randomInt(255);
79              var b = randomInt(255);
80              return '#' + formatNumber((r << 16) | (g << 8) | b);
81          }
82
83          function randomInt(n) {
84              return Math.floor(Math.random() * n);
85          }
86
87          function formatNumber(value) {
88              var result = value.toString(16);
89              while (result.length < 6) {
90                  result = '0' + result;
91              }
92              return result;
93          }
94
95  // ]]></script>
```

Name	Id	Icon	Inner Color
from	EA50CF2FC28F4A138418C354E764AA06	node_icon	
TWaver-99	C16C35DA72F240EBA971E0566AD02ADE	node_icon	
TWaver-98	ED4A23ED009A455997C75BED5ADA27A7	node_icon	
TWaver-97	1F5D806377694F2981C1CB942AC87858	node_icon	
TWaver-96	4A4D875A97CD4CDD8F948CFCE3159EFD	node_icon	
TWaver-95	79B15084398A4D1A87FE24D33A0EBF57	node_icon	
TWaver-94	E64039D72A9D41FB80F947C46D877D5B	node_icon	
TWaver-93	CBF6B283035E4F0187A22336F8D741F7	node_icon	
TWaver-92	A5FCFD76F590416C8FBF3A275EAEC9A1	node_icon	
TWaver-91	4DC0E848AEEE49F98F2DB43E3AA3B221	node_icon	
TWaver-90	34A2311407A44C738BD2539225A130FA	node_icon	
TWaver-89	2CEB00E10B4A45698363BF30EDA777B0	node_icon	
TWaver-88	AFA9EA33085240D19F4BE6570CF0BB44	node_icon	
TWaver-87	3E15C20940FD49408272388AE28BEDB7	node_icon	
TWaver-86	0DA2C0C91A5E442DA43E17A5D24005EA	node_icon	
TWaver-85	0C96431751E243768B0B80582F1DA8F8	node_icon	
TWaver-84	AF51046BD0254A19883033DABAED3BA4	node_icon	
TWaver-83	CFCDCE91196E49DD867B55C0A480D04B	node_icon	
TWaver-82	3C0F6F7073FC4E4A84E62C52799E03A8	node_icon	
TWaver-81	6D8800FDE2EA42A3888D2D448581BEBA	node_icon	
TWaver-80	EA93B18673D44EF89E5036BD53D61013	node_icon	
TWaver-79	C1954C5DAFA040279063395A073414E1	node_icon	

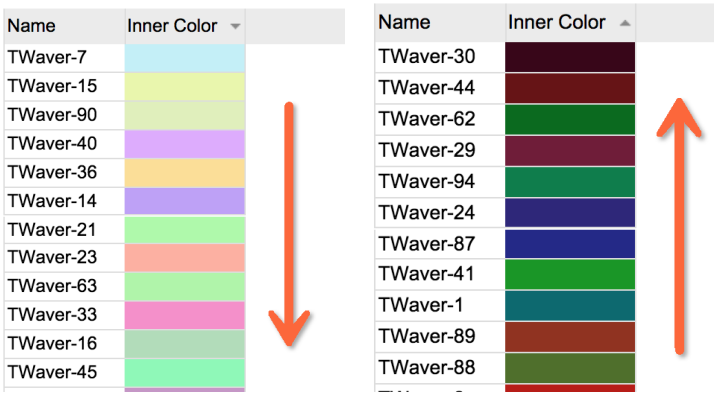
表格数据排序

表格中的数据可以通过调用dataBox.move***(element)来作调整，实现表格行的上下移动。

此外表格组件的每一列都支持排序功能，默认点击列头切换排序状态：升序、降序、不排序，默认的排序比较器使用字符串比较，特殊数据类型的可以定制排序比较器，下面我们将以颜色亮度为例，介绍表格排序的使用。排序比较器设置在表格列(Column)上，使用方法Column#setSortFunction(...)。

```
1  Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
2          var box = new twaver.ElementBox();
3          var network = new twaver.vector.Network(box);
4          var table = new twaver.controls.Table(box);
```

```
5
6           function init() {
7               initTable();
8               initDataBox();
9           }
10
11          function initTable() {
12              var tablePane = new twaver.controls.TablePane(table);
13              var tableDom = tablePane.getView();
14              tableDom.style.width = "100%";
15              tableDom.style.height = "100%";
16              document.body.appendChild(tableDom);
17              window.onresize = function () {
18                  tablePane.invalidate();
19              }
20              createColumn(table, 'Name', 'name', 'accessor', 'string');
21  //            createColumn(table, 'Id', 'id', 'accessor', 'string');
22  //            createColumn(table, 'Icon', 'icon', 'accessor');
23              var column = createColumn(table, 'Inner Color', 'inner.color', 'style', 'color');
24              column.setSortFunction(function (color1, color2) {
25                  if (!color1) {
26                      return -1;
27                  }
28                  if (!color2) {
29                      return 1;
30                  }
31                  var number1 = parseInt(color1.substring(1), 16);
32                  var r1 = (number1 >> 16) & 0xff;
33                  var g1 = (number1 >> 8) & 0xff;
34                  var b1 = number1 & 0xff;
35                  var number2 = parseInt(color2.substring(1), 16);
36                  var r2 = (number2 >> 16) & 0xff;
37                  var g2 = (number2 >> 8 ) & 0xff;
38                  var b2 = number2 & 0xff;
39                  return (r1 + g1 + b1) - (r2 + g2 + b2);
40              });
41          }
42          function initNetwork() {
43              var view = network.getView();
44              document.body.appendChild(view);
45              network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
46          }
47
48          function initDataBox() {
49              var i = 100;
50              while (i-- > 0) {
51                  var data = new twaver.Node();
52                  data.setName("TWaver-" + i);
53                  data.s('inner.color', randomColor());
54                  box.add(data);
55              }
56
57          }
58
59          function createColumn(table, name, propetyName, propertyType, valueType) {
60              var column = new twaver.Column(name);
61              column.setName(name);
62              column.setPropertyName(propetyName);
63              column.setPropertyType(propertyType);
64              if (valueType) {
65                  column.setValueType(valueType);
66              }
67              table.getColumnBox().add(column);
68              return column;
69          }
70
71          function randomColor() {
72              var r = randomInt(255);
73              var g = randomInt(255);
74              var b = randomInt(255);
75              return '#' + formatNumber((r << 16) | (g << 8) | b);
76          }
77
78          function randomInt(n) {
79              return Math.floor(Math.random() * n);
80          }
81
82          function formatNumber(value) {
83              var result = value.toString(16);
84              while (result.length < 6) {
85                  result = '0' + result;
86              }
87              return result;
88          }
89
90  // ]]></script>
```



常见问题

在Table上实现拖拽到Network创建网元

```
/**
 *1.首先让Network可以接受drop
 *
 */
network.getView().addEventListener('dragover', function (e) {
            if (e.preventDefault) {
                e.preventDefault();
            } else {
                e.returnValue = false;
            }
            e.dataTransfer.dropEffect = 'copy';
            return false;
        }, false);
        network.getView().addEventListener('drop', function (e) {
            if (e.stopPropagation) {
                e.stopPropagation();
            }
            if (e.preventDefault) {
                e.preventDefault();
            } else {
                e.returnValue = false;
            }
            var text = e.dataTransfer.getData('Text');
            if (!text) {
                return false;
            }
            if (text &amp;amp;&amp;amp; text.indexOf('className:') == 0) {
                demo.Util._createElement(network, text.substr(10, text.length), network.getLogicalPoint(e));
            }
            return false;
        }, false);
/**
 *2.然后让Table可以实现drag
 *
 */
table.onCellRendered = function (params) {
                if (params.column.getName() === 'Name') {
                    params.div.style.backgroundColor = params.data.getClient('stateColor');
                    if(params.div.getAttribute('draggable') != 'true') {
                        params.div.setAttribute('draggable', 'true');
                        params.div.addEventListener('dragstart', function (e) {
                            e.dataTransfer.effectAllowed = 'copy';
                            e.dataTransfer.setData('Text', 'className');
                        }, false);
                    }
                }
            };
/**
 *3.TWaver中对事件传播进行了控制,在此处需要重写以下方法
 */
_twaver.html.createView = function (overflow, keepDefault) {
                var e = document.createElement('div');
                e.style.position = twaver.Defaults.VIEW_POSITION;
                e.style.fontSize = twaver.Defaults.VIEW_FONT_SIZE;
                e.style.fontFamily = twaver.Defaults.VIEW_FONT_FAMILY;
                e.style.cursor = 'default';
                e.style.outline = 'none';
                e.style.textAlign = "left";
                e.style.msTouchAction = "none";
                e.tabIndex = 0;

                if (keepDefault) {
                    // do nothing
                } else {
                   // e.onmousedown = $html.preventDefault;
                }
                if (e.style.setProperty) {
                    e.style.setProperty("-khtml-user-select", "none", null);
                    e.style.setProperty("-webkit-user-select", "none", null);
                    e.style.setProperty("-moz-user-select", "none", null);
                    e.style.setProperty("-webkit-tap-highlight-color", "rgba(0, 0, 0, 0)", null);

                }
                if (overflow) {
```

```
                    e.style.overflow = overflow;
                }
                return e;
            }
```

单击表格时获得点击的某列表头名称及值

```
table.getView().addEventListener("mousedown", function(e) {
        var column = table.getColumnAt(e);
        console.log(column.getName());
});
```

如何响应表格某列的值变化事件

```
//表格中的每一列都对应一个网元的一个属性。可以直接对table.getDataBox()进行监听即可：
box.addDataPropertyChangeListener(function(e) {
});
```

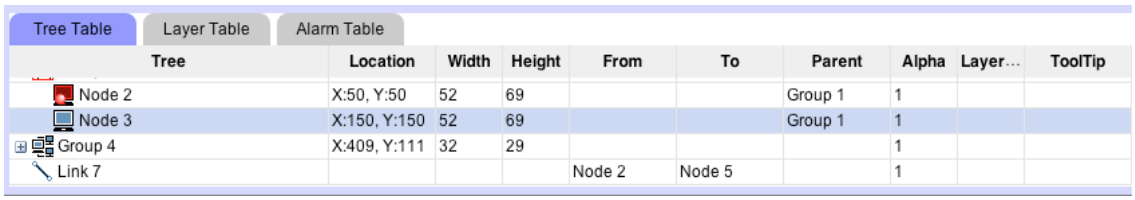
获取选中表格的数量

```
table.getSelectionModel().getSelection().size();
```

如何让表格列占满剩余空间

```
//实现思路是在表格大小更改时(adjustBounds)，重新计算剩余空间，分配给需要最大化的列（client属性pack为true）。
demo.AutoPackTreeTable = function (dataBox) {
    demo.AutoPackTreeTable.superClass.constructor.call(this, dataBox);
};
twaver.Util.ext('demo.AutoPackTreeTable', twaver.controls.TreeTable, {
    _minPackWidth: 100,
    getMinPackWidth: function () {
        return this._minPackWidth;
    },
    setMinPackWidth: function (v) {
        this._minPackWidth = v;
    },
    adjustBounds: function (rect) {
        demo.AutoPackTreeTable.superClass.adjustBounds.call(this, rect);
        this.packColumns(rect.width);
    },
    packColumns: function (width) {
        var packCoumns = new twaver.List(),
            packWidth;
        this.getColumnBox().getRoots().forEach(function (column) {
            if (column.getClient('pack')) {
                packCoumns.add(column);
            } else {
                width -= column.getWidth();
            }
        });
        if (packCoumns.size() === 0) {
            return;
        }
        packWidth = width / packCoumns.size();
        if (packWidth &lt; this._minPackWidth) {
            packWidth = this._minPackWidth;
        }
        packCoumns.forEach(function (column) {
            column.setWidth(packWidth);
        });
    }
});
```

比如要让treetable的第一列最大，那么就：this.treeTable.getTreeColumn().setClient('pack', true)。



获取Table中每行每列的值

```
//获取每一行数据。
table.getDataBox()
//获取表格的列,对应twaver.Column
table.getColumnBox()
//获取第一行第一列的值
var node =table. getDatabox.getDataAt(0);
var column = table.getColumnBox().getDataAt(0);
var value = table.getValue(node,column);
```

树表可视化视图组件(TreeTable)

所谓树表组件，就是在普通表格的基础上增加父子关系,可支持展开、合并等功能。使用方法和Table一样。

```
//1.创建treetable并绑定DataBox
var treetable = new twaver.controls.TreeTable(box);
//2.创建tablePane并绑定treetable
var tablePane = new twaver.controls.TablePane(table);
//3.布局tablepane
 var tableDom = tablePane.getView();
            tableDom.style.width = "100%";
            tableDom.style.height = "100%";
            document.body.appendChild(tableDom);
            window.onresize = function () {
                tablePane.invalidate();
            }
//4.填充数据
  var column = new twaver.Column(name);
            column.setName(name);
            column.setPropertyName(propetyName);
            column.setPropertyType(propertyType);
            if (valueType) {
                column.setValueType(valueType);
            }
            table.getColumnBox().add(column);
```

示例:

```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);
        var table = new twaver.controls.TreeTable(box);

        function init() {
            initTable();
            initDataBox();
        }

        function initTable() {
            var tablePane = new twaver.controls.TablePane(table);
            var tableDom = tablePane.getView();
            tableDom.style.width = "100%";
            tableDom.style.height = "100%";
            document.body.appendChild(tableDom);
            window.onresize = function () {
                tablePane.invalidate();
            }
            createColumn(table, 'Name', 'name', 'accessor', 'string');
            createColumn(table, 'Id', 'id', 'accessor', 'string');
            createColumn(table, 'Icon', 'icon', 'accessor');
            createColumn(table, 'Inner Color', 'inner.color', 'style','color');
        }
        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }

        function initDataBox() {
            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                }
            });
            box.add(from);

            var to = new twaver.Node({
                name: 'to',
                location: {
                    x: 220,
                    y: 160
                }
            });
            box.add(to);

            var link = new twaver.Link(from, to);
            box.add(link);

            var i = 10;
            while (i-- > 0) {
                var data = new twaver.Node();
                data.setName("TWaver-" + i);
                data.setParent(from);
                data.s('inner.color', randomColor());
                box.add(data);
            }

        }

        function createColumn(table, name, propetyName, propertyType, valueType) {
            var column = new twaver.Column(name);
            column.setName(name);
            column.setPropertyName(propetyName);
            column.setPropertyType(propertyType);
            if (valueType) {
                column.setValueType(valueType);
            }
            table.getColumnBox().add(column);
            return column;
        }

        function randomColor() {
            var r = randomInt(255);
            var g = randomInt(255);
```

```
            var b = randomInt(255);
            return '#' + formatNumber((r << 16) | (g << 8) | b);
        }

        function randomInt(n) {
            return Math.floor(Math.random() * n);
        }

        function formatNumber(value) {
            var result = value.toString(16);
            while (result.length < 6) {
                result = '0' + result;
            }
            return result;
        }

// ]]></script>
```

tree	Name	Id	Icon	Inner Color
from	from	57B2D6679…	node_icon	
TWaver-9	TWaver-9	858BC4D2…	node_icon	
TWaver-8	TWaver-8	0D2B5ED8…	node_icon	
TWaver-7	TWaver-7	48A00771B…	node_icon	
TWaver-6	TWaver-6	02BB2EAF…	node_icon	
TWaver-5	TWaver-5	000D21DB…	node_icon	
TWaver-4	TWaver-4	574A5F06A…	node_icon	
TWaver-3	TWaver-3	1DCC069E…	node_icon	
TWaver-2	TWaver-2	61323B41F…	node_icon	
TWaver-1	TWaver-1	FD88E4449…	node_icon	
TWaver-0	TWaver-0	BA7E12421…	node_icon	
to	to	AEF889B5…	node_icon	
		85D848CA…	link_icon	

列表可视化视图组件(List)

列表组件List是一个可滚动的单选或多选列表框。List组件使用基于零的索引,其中索引为0的项目就是显示在顶端的项目。当使用List类的方法和属性添加、删除或替换列表项时，您可能需要指定该列表项的索引,另外我们也可以使用键盘方向键控制它。

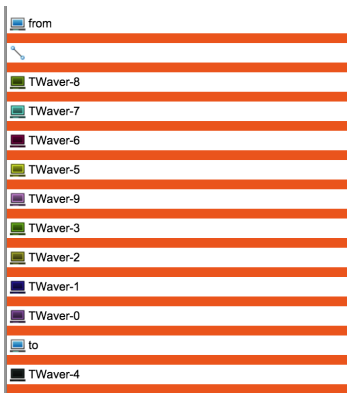
创建列表的步骤如下:

```
//1.创建list并绑定DataBox
var list = new twaver.controls.List(box);
//2.布局List组件
 var listPan = list.getView();
 listPan.style.width = "100%";
 listPan.style.height = "100%";
 document.body.appendChild(listPan);
//3.设置样式或排序规则
list.setRowLineWidth(10);
list.setRowHeight(30);
list.setRowLineColor(twaver.Colors.blue_dark);
list.setSortFunction(function (node1, node2) {
    if (!node1) {
        return -1;
    }
    if (!node2) {
        return 1;
    }

    return node2.getId() - node1.getId();
});
```

示例:

```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
        var box = new twaver.ElementBox();
        var network = new twaver.vector.Network(box);
        var list = new twaver.controls.List(box);

        function init() {
            initTable();
            initDataBox();
        }

        function initTable() {
            var listPan = list.getView();
            listPan.style.width = "100%";
            listPan.style.height = "100%";
            document.body.appendChild(listPan);

            list.setRowLineWidth(10);
            list.setRowHeight(30);
            list.setRowLineColor(twaver.Colors.blue_dark);
            list.setSortFunction(function (node1, node2) {
                if (!node1) {
                    return -1;
                }
                if (!node2) {
                    return 1;
                }
```

```
                return node2.getId() - node1.getId();
            });
        }
        function initNetwork() {
            var view = network.getView();
            document.body.appendChild(view);
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
        }

        function initDataBox() {
            var from = new twaver.Node({
                name: 'from',
                location: {
                    x: 100,
                    y: 100
                }
            });
            box.add(from);

            var to = new twaver.Node({
                name: 'to',
                location: {
                    x: 220,
                    y: 160
                }
            });
            box.add(to);

            var link = new twaver.Link(from, to);
            box.add(link);

            var i = 10;
            while (i-- > 0) {
                var data = new twaver.Node();
                data.setName("TWaver-" + i);
                data.setParent(from);
                data.s('inner.color', randomColor());
                box.add(data);
            }

        }

        function randomColor() {
            var r = randomInt(255);
            var g = randomInt(255);
            var b = randomInt(255);
            return '#' + formatNumber((r << 16) | (g << 8) | b);
        }

        function randomInt(n) {
            return Math.floor(Math.random() * n);
        }

        function formatNumber(value) {
            var result = value.toString(16);
            while (result.length < 6) {
                result = '0' + result;
            }
            return result;
        }

// ]]></script>
```



属性可视化视图组件(PropertySheet)

PropertySheet属性表组件是一个表格，通过"属性—-值"两列显示一个管理对象的属性。属性表支持对管对对象的属性发现与遍历、显示、编辑、渲染等。在使用属性组件之前，我们应该了解TWaver的另外一个网元对象twaver.Property，它继承于twaver.Data，用来创建一个属性表中的每行属性对象。

属性(Property)

twaver.Property的使用类似于twaver.Node的使用,创建对象、设置属性、添加到DataBox中,下面来看看Property的使用方法。

```
    //new 一个Property对象
    var property = new twaver.Property();
```

```
    //设置分类名称
    property.setCategoryName(category);
        if (!name) {
            name = _getNameFromPropertyName(propertyName);
        }

    property.setName(name);
    //设置是否可编辑
    property.setEditable(true);
    //设置属性类型
    property.setPropertyType(proprtyType);
    //设置属性名称
    property.setPropertyName(propertyName);

    var valueType;
    if (proprtyType === 'style') {
        valueType = twaver.SerializationSettings.getStyleType(propertyName);
    } else if (proprtyType === 'client') {
        valueType = twaver.SerializationSettings.getClientType(propertyName);
    } else {
        valueType = twaver.SerializationSettings.getPropertyType(propertyName);
    }
    if (valueType) {
        property.setValueType(valueType);
    }

    box.add(property);
```

属性面板(PropertySheet)

PropertySheet的使用类似于Network,创建并绑定数据源(DataBox),然后设置布局即可。创建属性表的一般步骤如下:

```
//1.创建PropertySheet,并与DataBox绑定
 var propertysheet = new twaver.controls.PropertySheet(box);
//2.布局属性面板
 var pane = new twaver.controls.SplitPane(propertysheet, network, 'horizontal', 0.3);
        var view = pane.getView();
        view.style.left = '0px';
        view.style.top = '0px';
        view.style.right = '0px';
        view.style.bottom = '0px';
        document.body.appendChild(view);
//3.创建属性信息并添加到DataBox中
var property = new twaver.Property();
...
box.add(property);
```

另外值得一提的是，如何监听通过属性表格修改网元属性所派发的事件？其实很简单，因为属性表与DataBox绑定，属性表所操作的依然是DataBox中的网元，所以监听的方法和DataBox监听网元属性变化的方法是一致的，代码如下:

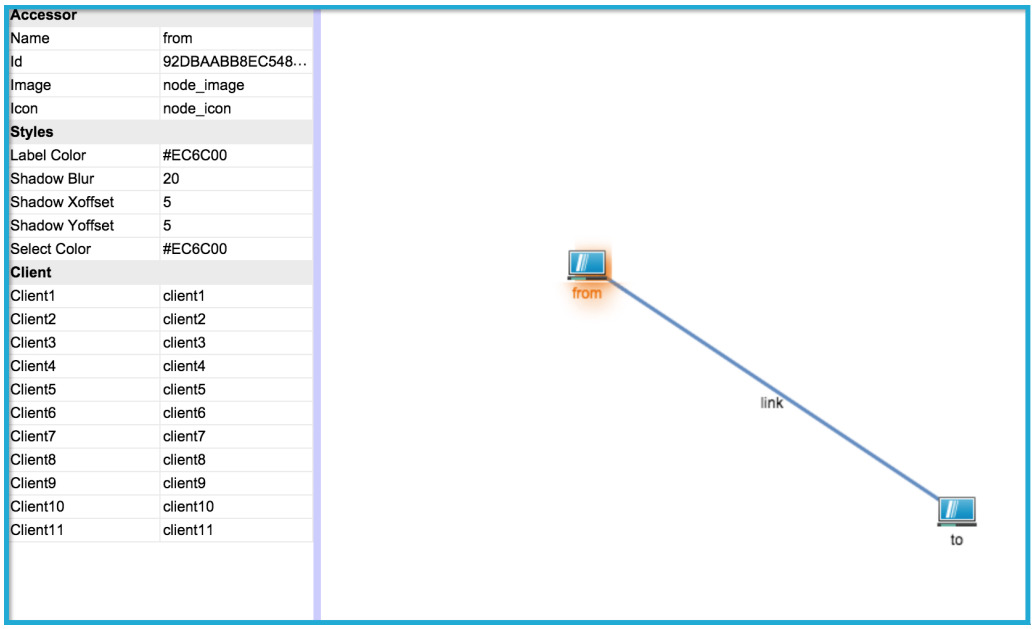
```
    propertysheet.getDataBox().addDataPropertyChangeListener(function(e){
        console.log(e);
    });
```

示例：

```
<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    var propertysheet = new twaver.controls.PropertySheet(box);

    function init() {
        initNetwork();
        initBox();
        initSheet();
    }

    function initNetwork(){
        var networkPane = network.getView();
        var pane = new twaver.controls.SplitPane(propertysheet, network, 'horizontal', 0.3);
        var view = pane.getView();
        view.style.left = '0px';
        view.style.top = '0px';
        view.style.right = '0px';
        view.style.bottom = '0px';
        document.body.appendChild(view);
        window.onresize = function () {
            pane.invalidate();
        };
    }

    function initSheet(){
        var propertyBox = propertysheet.getPropertyBox();
        addAccessorProperty(propertyBox,"Name","Accessor");
        addAccessorProperty(propertyBox,"Id","Accessor");
        addAccessorProperty(propertyBox,"Image","Accessor");
        addAccessorProperty(propertyBox,"Icon","Accessor");

        addStyleProperty(propertyBox,"label.color","Styles");
        addStyleProperty(propertyBox,"shadow.blur","Styles");
        addStyleProperty(propertyBox,"shadow.xoffset","Styles");
        addStyleProperty(propertyBox,"shadow.yoffset","Styles");
        addStyleProperty(propertyBox,"select.color","Styles");

        addClientProperty(propertyBox,"client1","Client").setValueType('string');
        addClientProperty(propertyBox,"client2","Client").setValueType('string');
```

```
        addClientProperty(propertyBox,"client3","Client").setValueType('string');
        addClientProperty(propertyBox,"client4","Client").setValueType('string');
        addClientProperty(propertyBox,"client5","Client").setValueType('string');
        addClientProperty(propertyBox,"client6","Client").setValueType('string');
        addClientProperty(propertyBox,"client7","Client").setValueType('string');
        addClientProperty(propertyBox,"client8","Client").setValueType('string');
        addClientProperty(propertyBox,"client9","Client").setValueType('string');
        addClientProperty(propertyBox,"client10","Client").setValueType('string');
        addClientProperty(propertyBox,"client11","Client").setValueType('string');


        propertysheet.setEditable(true);
        // propertysheet.setBorderColor("#06387a");
        propertysheet.setIndent(0);
        propertysheet.setSelectColor();
        // propertysheet.setPropertyNameWidth(60);
    }

    function addStyleProperty (box, propertyName, category, name) {
        return _addProperty(box, propertyName, category, name, 'style');
    }
    function addClientProperty (box, propertyName, category, name) {
        return _addProperty(box, propertyName, category, name, 'client');
    }
    function addAccessorProperty (box, propertyName, category, name) {
        return _addProperty(box, propertyName, category, name, 'accessor');
    }
    function _addProperty(box, propertyName, category, name, proprtyType) {
        var property = new twaver.Property();
        property.setCategoryName(category);
        if (!name) {
            name = _getNameFromPropertyName(propertyName);
        }
        property.setName(name);
        property.setEditable(true);
        property.setPropertyType(proprtyType);
        property.setPropertyName(propertyName);

        var valueType;
        if (proprtyType === 'style') {
            valueType = twaver.SerializationSettings.getStyleType(propertyName);
        } else if (proprtyType === 'client') {
            valueType = twaver.SerializationSettings.getClientType(propertyName);
        } else {
            valueType = twaver.SerializationSettings.getPropertyType(propertyName);
        }
        if (valueType) {
            property.setValueType(valueType);
        }

        box.add(property);
        return property;
    }

    function _getNameFromPropertyName (propertyName) {
        var names = propertyName.split('.');
        var name = '';
        for (var i = 0; i < names.length; i++) {             if (names[i].length > 0) {
                name += names[i].substring(0, 1).toUpperCase() + names[i].substring(1, names[i].length);
            }
            if (i < names.length - 1) {
                name += ' ';
            }
        }
        return name;
    }


    function initBox() {
        var from = new twaver.Node();
        from.setLocation(200,200);
        from.setName("from");
        from.setClient('client1','client1');
        from.setClient('client2','client2');
        from.setClient('client3','client3');
        from.setClient('client4','client4');
        from.setClient('client5','client5');
        from.setClient('client6','client6');
        from.setClient('client7','client7');
        from.setClient('client8','client8');
        from.setClient('client9','client9');
        from.setClient('client10','client10');
        from.setClient('client11','client11');
        from.setStyle('label.color',twaver.Colors.orange);
        from.setStyle('shadow.blur',20);
        from.setStyle('shadow.xoffset',5);
        from.setStyle('shadow.yoffset',5);
        from.setStyle('select.color',twaver.Colors.orange);


        box.add(from);

        console.log(from);
        var to = new twaver.Node();
        to.setName("to");
        to.setLocation(500,400);
        box.add(to);

        var link = new twaver.Link(from,to);
        link.setName("link");
        box.add(link);

        box.getSelectionModel().setSelection(from);
        box.addDataPropertyChangeListener(function(e){
```

```
            console.log(e);
        });
        // propertysheet.getDataBox().addDataPropertyChangeListener(function(e){
        //     console.log(e);
        // });
}
// ]]></script>
```



常见问题

设置属性表高度

```
sheet.setRowHeight(30);
sheet.setRowLineHeight(20);
```

弹出框可视化视图组件(PopupMenu)

PopupMenu直接继承于Object类，邮件弹出对话框，实现与系统交互功能。

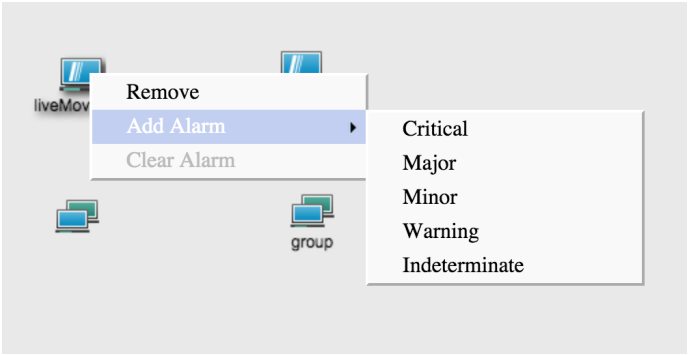
使用示例:

```
<script src="../twaver.js" type="text/javascript"></script><script type="text/javascript">// <![CDATA[
        var box = new twaver.ElementBox();
        var network;
        var popupMenu;

        function init() {
```

```
            initNetwork();
            initDataBox();
            initPopupMenu();
            initListener();
        }

        function initNetwork(){
            network = new twaver.canvas.Network(box);
            document.body.appendChild(network.getView());
            network.getView().style.background = '#E9E9E9';
            network.adjustBounds({x: 0, y: 0, width: 1300, height: 600});
            network.setLimitViewInCanvas(false);
//            network.setTouchInteractions();
            popupMenu = new twaver.controls.PopupMenu(network);
        }

        function initListener(){
            var node = new twaver.Node();
            node.setLocation(200,200);
            box.add(node);

            network.addInteractionListener(function(e){
                 node.setName(e.kind);
            });
        }

        function initDataBox() {
            var group = new twaver.Group();
            box.add(group);

            var from = new twaver.Node();
            from.setName('From');
            from.setLocation(Math.random()*1000,Math.random()*500);
            box.add(from);

            var node1 = new twaver.Node();
            node1.setName('node1');
            node1.setLocation(100,100);
            box.add(node1);


            var node2 = new twaver.Node();
            node2.setName('node2');
            node2.setLocation(300,300);
            box.add(node2);

            var link = new twaver.Link(node1,node2);
            box.add(link);

            var group = new twaver.Group();
            group.setName('group');
            box.add(group);

            group.addChild(node1);
            group.addChild(node2);
        }

        function initPopupMenu() {
            var lastData, lastPoint, magnifyInteraction;
            popupMenu.onMenuShowing = function (e) {
                lastData = network.getSelectionModel().getLastData();
                lastPoint = network.getLogicalPoint(e);
                magnifyInteraction = null;
                network.getInteractions().forEach(function (interaction) {
                    if (interaction instanceof twaver.network.interaction.MagnifyInteraction
                            || interaction instanceof twaver.canvas.interaction.MagnifyInteraction) {
                        magnifyInteraction = interaction;
                    }
                });
                return true;
            };
            popupMenu.onAction = function (menuItem) {
                if (menuItem.label === 'Expand Group'
                        || menuItem.label === 'Collapse Group') {
                    lastData.reverseExpanded();
                }
                if (menuItem.label === 'Enter SubNetwork') {
                    network.setCurrentSubNetwork(lastData);
                }
                if (menuItem.label === 'Up SubNetwork') {
                    network.upSubNetwork();
                }
                if (menuItem.label === 'Expand LinkBundle'
                        || menuItem.label === 'Collapse LinkBundle') {
                    lastData.reverseBundleExpanded();
                }
                if (menuItem.label === 'Critical') {
                    lastData.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.CRITICAL, 1);
                }
                if (menuItem.label === 'Major') {
                    lastData.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.MAJOR, 1);
                }
                if (menuItem.label === 'Minor') {
                    lastData.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.MINOR, 1);
                }
                if (menuItem.label === 'Warning') {
                    lastData.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.WARNING, 1);
                }
                if (menuItem.label === 'Indeterminate') {
                    lastData.getAlarmState().increaseNewAlarm(twaver.AlarmSeverity.INDETERMINATE, 1);
                }
                if (menuItem.label === 'Clear Alarm') {
                    lastData.getAlarmState().clear();
                }
```

```
            if (menuItem.label === 'Remove') {
                box.remove(lastData);
            }
            if (menuItem.label === 'Add Node') {
                var node = new twaver.Node();
                node.setParent(network.getCurrentSubNetwork());
                node.setCenterLocation(lastPoint);
                box.add(node);
            }
        };
        popupMenu.isVisible = function (menuItem) {
            if (magnifyInteraction) {
                return menuItem.group === 'Magnify';
            } else {
                if (lastData) {
                    if (lastData instanceof twaver.SubNetwork && menuItem.group === 'SubNetwork') {
                        return true;
                    }
                    // if (lastData instanceof twaver.Group && menuItem.group === 'Group') {
                    //     return true;
                    // }
                    if (lastData instanceof twaver.Link && menuItem.group === 'Link') {
                        return true;
                    }
                    return menuItem.group === 'Element';
                } else {
                    return menuItem.group === 'none';
                }
            }
        };
        popupMenu.isEnabled = function (menuItem) {
            if (lastData) {
                if (lastData instanceof twaver.SubNetwork) {
                    return true;
                }
                if (lastData instanceof twaver.Group && menuItem.group === 'Group') {
                    var expanded = lastData.isExpanded();
                    return menuItem.expand ? !expanded : expanded;
                }
                if (lastData instanceof twaver.Link && menuItem.group === 'Link') {
                    var expanded = lastData.getStyle("link.bundle.expanded");
                    return menuItem.expand ? !expanded : expanded;
                }
                if (menuItem.label === 'Clear Alarm') {
                    return !lastData.getAlarmState().isEmpty();
                }
            } else {
                if (menuItem.label === 'Up SubNetwork') {
                    return network.getCurrentSubNetwork() != null;
                }
            }
            return true;
        };
        popupMenu.setMenuItems([
            { label: 'Remove', group: 'Element' },
            { label: 'CNode Item', group: 'CNode' },
            { label: 'Add Alarm', group: 'Element',
                items:[{label:'Critical', group:'Element'},
                    {label:'Major', group:'Element'},
                    {label:'Minor', group:'Element'},
                    {label:'Warning', group:'Element'},
                    {label:'Indeterminate', group:'Element'}] },
            { label: 'Clear Alarm', group: 'Element' },
            { separator: true, group: 'Element' },
            { label: 'Expand LinkBundle', group: 'Link', expand: true },
            { label: 'Collapse LinkBundle', group: 'Link' },
            { separator: true, group: 'Link' },
            { label: 'Expand Group', group: 'Group', expand: true },
            { label: 'Collapse Group', group: 'Group' },
            { separator: true, group: 'Group' },
            { label: 'Enter SubNetwork', group: 'SubNetwork' },

            { label: 'Add Node', group: 'none'},
            { label: 'Up SubNetwork', group: 'none' },
            { label: 'Shape', group: 'Magnify', items: [
                { label: 'rectangle', type: 'radio', groupName: 'Shape', group: 'Magnify', action: function () {
                    magnifyInteraction.setShape('rectangle');
                    magnifyInteraction.setYRadius(magnifyInteraction.getXRadius());
                } },
                { label: 'roundrect', type: 'radio', groupName: 'Shape', group: 'Magnify', action: function () {
                    magnifyInteraction.setShape('roundrect');
                    magnifyInteraction.setYRadius(magnifyInteraction.getXRadius());
                } },
                { label: 'oval', type: 'radio', groupName: 'Shape', group: 'Magnify', action: function () {
                    magnifyInteraction.setShape('oval');
                    magnifyInteraction.setYRadius(magnifyInteraction.getXRadius() * 0.75);
                } },
                { label: 'round', type: 'radio', groupName: 'Shape', selected: true, group: 'Magnify', action: function () {
                    magnifyInteraction.setShape('round');
                    magnifyInteraction.setYRadius(magnifyInteraction.getXRadius());
                } },
                { label: 'star', type: 'radio', groupName: 'Shape', group: 'Magnify', action: function () {
                    magnifyInteraction.setShape('star');
                    magnifyInteraction.setYRadius(magnifyInteraction.getXRadius());
                } },
            ] },
            { label: 'Zoom', group: 'Magnify', items: [
                { label: 2, type: 'radio', groupName: 'Zoom', selected: true, group: 'Magnify', action: function () { magnifyIn
                { label: 3, type: 'radio', groupName: 'Zoom', group: 'Magnify', action: function () { magnifyInteraction.setZoo
                { label: 4, type: 'radio', groupName: 'Zoom', group: 'Magnify', action: function () { magnifyInteraction.setZoo
            ] },
            { label: 'Size', group: 'Magnify', items: [
                { label: 50, type: 'radio', groupName: 'Size', group: 'Magnify', action: function () { magnifyInteraction.setXR
                { label: 100, type: 'radio', groupName: 'Size', selected: true, group: 'Magnify', action: function () { magnify
```

```
                    { label: 200, type: 'radio', groupName: 'Size', group: 'Magnify', action: function () { magnifyInteraction.setX
                ] },
                { label: 'BorderWidth', group: 'Magnify', items: [
                    { label: '1', type: 'radio', groupName: 'BorderWidth', selected: true, group: 'Magnify', action: function () {
                    { label: '2', type: 'radio', groupName: 'BorderWidth', group: 'Magnify', action: function () { magnifyInteracti
                    { label: '3', type: 'radio', groupName: 'BorderWidth', group: 'Magnify', action: function () { magnifyInteracti
                ] },
                { label: 'BorderColor', group: 'Magnify', items: [
                    { label: 'black', type: 'radio', groupName: 'BorderColor', selected: true, group: 'Magnify', action: function (
                    { label: 'green', type: 'radio', groupName: 'BorderColor', group: 'Magnify', action: function () { magnifyInter
                    { label: 'blue', type: 'radio', groupName: 'BorderColor', group: 'Magnify', action: function () { magnifyIntera
                ] },
                { label: 'BackgroundColor', group: 'Magnify', items: [
                    { label: 'white', type: 'radio', groupName: 'BackgroundColor', selected: true, group: 'Magnify', action: functi
                    { label: 'transparent', type: 'radio', groupName: 'BackgroundColor', group: 'Magnify', action: function () { ma
                    { label: 'black', type: 'radio', groupName: 'BackgroundColor', group: 'Magnify', action: function () { magnifyI
                ] }
            ]);
        }

// ]]></script>
```



样式面板

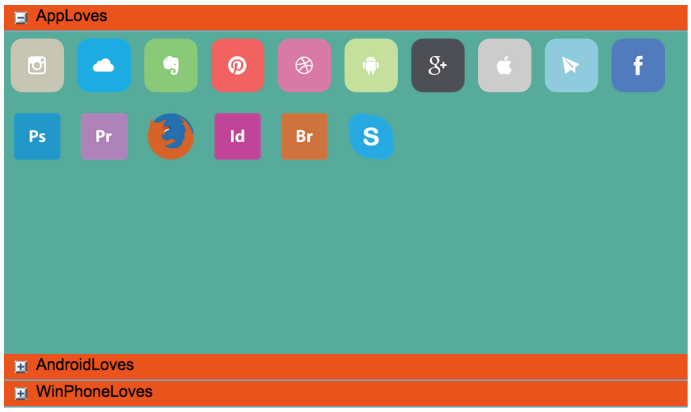
健全的功能加上合理的布局才是一套完美的系统，所以布局对于产品而言尤为重要，TWaver内置了一些常用的布局面板，我们可以直接拿来使用，方便快捷，而且好看。下面我们详细介绍几种常用的布局面板。

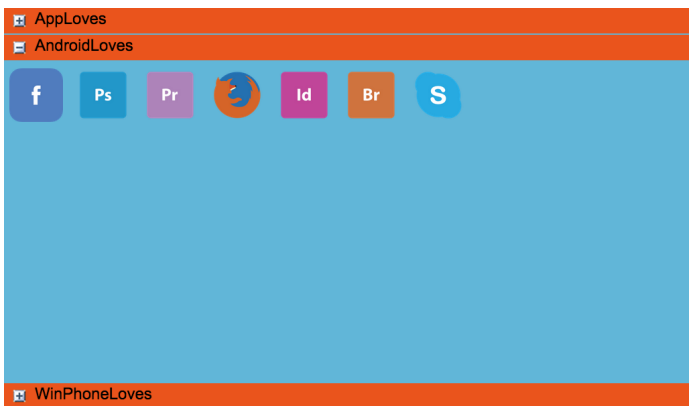
可折叠面板(Accordion)

示例:

```
Test Zoom<script src="../twaver.js"></script><script>// <![CDATA[
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);
    var accordion = new twaver.controls.Accordion();
    var appLoveDiv = document.createElement('div');
    var androidLoveDiv = document.createElement('div');
    var winPhoneDiv = document.createElement('div');
    function init() {
        initAccordion();
        initDataBox();
    }

    function initAccordion(){
        accordion.adjustBounds({x: 0, y: 0, width: 512, height: 300});
        document.body.appendChild(accordion.getView());
        accordion.setTitleBackground(twaver.Colors.orange_dark);
        accordion.setBorderBottomColor(twaver.Colors.blue_light);
        accordion.add('AppLoves', appLoveDiv);
        accordion.add('AndroidLoves', androidLoveDiv);
        accordion.add('WinPhoneLoves', winPhoneDiv);

        appLoveDiv.style.background=twaver.Colors.green_light;
        androidLoveDiv.style.background=twaver.Colors.blue_light;

        addLove(appLoveDiv,'../images/icons/icon01.png');
        addLove(appLoveDiv,'../images/icons/icon02.png');
        addLove(appLoveDiv,'../images/icons/icon03.png');
        addLove(appLoveDiv,'../images/icons/icon04.png');
        addLove(appLoveDiv,'../images/icons/icon05.png');
        addLove(appLoveDiv,'../images/icons/icon06.png');
        addLove(appLoveDiv,'../images/icons/icon07.png');
        addLove(appLoveDiv,'../images/icons/icon08.png');
        addLove(appLoveDiv,'../images/icons/icon09.png');
        addLove(appLoveDiv,'../images/icons/icon10.png');
        addLove(appLoveDiv,'../images/icons/icon11.png');
        addLove(appLoveDiv,'../images/icons/icon12.png');
        addLove(appLoveDiv,'../images/icons/icon13.png');
        addLove(appLoveDiv,'../images/icons/icon14.png');
        addLove(appLoveDiv,'../images/icons/icon15.png');
        addLove(appLoveDiv,'../images/icons/icon16.png');

        addLove(androidLoveDiv,'../images/icons/icon10.png');
        addLove(androidLoveDiv,'../images/icons/icon11.png');
        addLove(androidLoveDiv,'../images/icons/icon12.png');
        addLove(androidLoveDiv,'../images/icons/icon13.png');
        addLove(androidLoveDiv,'../images/icons/icon14.png');
```

```
        addLove(androidLoveDiv,'../images/icons/icon15.png');
        addLove(androidLoveDiv,'../images/icons/icon16.png');
    }

    function addLove(div,src) {
        var imageName = getImageName(src);
        var button = document.createElement('input');
        button.setAttribute('type', 'image');
        button.setAttribute('title', imageName);
        button.style.padding = '5px 5px 5px 5px';
        button.style.width = '40px';
        button.style.height = '40px';
        button.setAttribute('src', src);
        var self = this;
        button.addEventListener('click', function () {
            self.network.setCreateElementInteractions(function (point) {
                var element = new Node();
                element.setImage(imageName);
                element.setCenterLocation(point);
                return element;
            });
        }, false);
        div.appendChild(button);
        return button;
    }

    function getImageName (url) {
        var index = url.lastIndexOf('/');
        var name = url;
        if (index >= 0) {
            name = url.substring(index + 1);
        }
        index = name.lastIndexOf('.');
        if (index >= 0) {
            name = name.substring(0, index);
        }
        return name;
    }

// ]]></script>
```





边框面板(BorderPane)

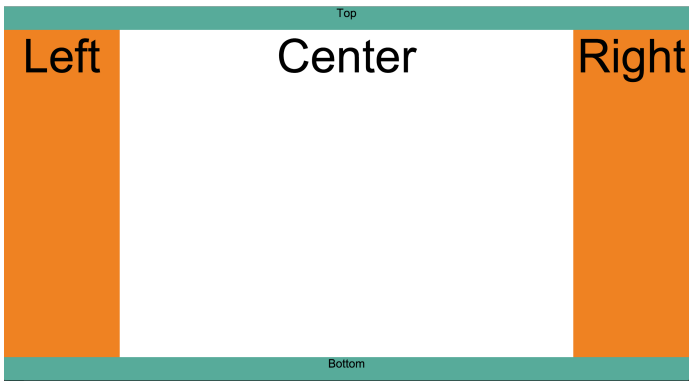
```
//BorderPane的方法定义如下:
BorderPane = function (center, top, right, bottom, left)
```

使用步骤:

```
//1.创建borderPane
var borderPane = new twaver.controls.BorderPane(centerPane,topPane,rightPane,bottomPane,leftPane);
//2.设置borderPane边距属性
 borderPane.setTopHeight(40);
 borderPane.setLeftWidth(200);
 borderPane.setBottomHeight(40);
 borderPane.setRightWidth(200);
```

示例:

```
<script src="../twaver.js"></script><script>// <![CDATA[
    var borderPane;
    var splitPane ;

    function init() {
        initBorderPane();
    }

    function initBorderPane(){
        var centerPane = document.createElement('div');
        var topPane = document.createElement('div');
        var bottomPane = document.createElement('div');
        var leftPane = document.createElement('div');
        var rightPane = document.createElement('div');
        // centerPane.style.background=twaver.Colors.blue_light;
        topPane.style.background=twaver.Colors.green_light;
        leftPane.style.background=twaver.Colors.orange_light;
        bottomPane.style.background=twaver.Colors.green_light;
        rightPane.style.background=twaver.Colors.orange_light;

        centerPane.innerHTML="Center";
        centerPane.style.textAlign="center";
        centerPane.style.fontSize = "80px";

        topPane.innerHTML="Top";
        topPane.style.textAlign="center";
        topPane.style.fontSize = "20px";

        leftPane.innerHTML="Left";
        leftPane.style.textAlign="center";
        leftPane.style.fontSize = "80px";

        bottomPane.innerHTML="Bottom";
        bottomPane.style.textAlign="center";
        bottomPane.style.fontSize = "20px";

        rightPane.innerHTML="Right";
        rightPane.style.textAlign="center";
        rightPane.style.fontSize = "80px";

        borderPane = new twaver.controls.BorderPane(centerPane,topPane,rightPane,bottomPane,leftPane);
        borderPane.setTopHeight(40);
        borderPane.setLeftWidth(200);
        borderPane.setBottomHeight(40);
        borderPane.setRightWidth(200);

        borderPane.getView().style.position = 'absolute';
        borderPane.getView().style.left = '0px';
        borderPane.getView().style.top = '0px';
        borderPane.getView().style.right = '0px';
        borderPane.getView().style.bottom = '0px';

        document.body.appendChild(borderPane.getView());
        window.onresize = function (e) { borderPane.invalidate(); };
    }

// ]]></script>
```

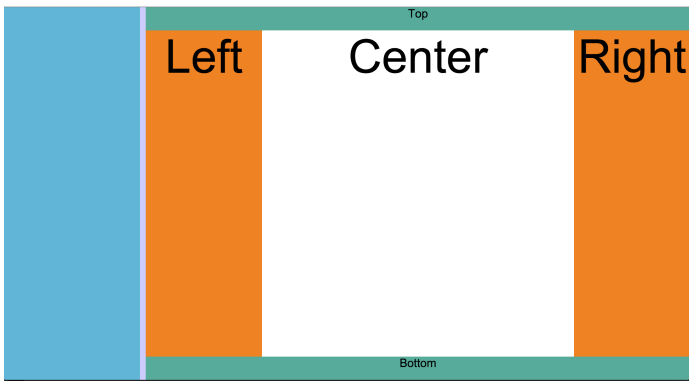


分割面板(SplitPane)

示例

```
<script src="../twaver.js"></script><script>// <![CDATA[
    var borderPane;
    var splitPane ;

    function init() {
        initPane();
    }

    function initPane(){
        var centerPane = document.createElement('div');
        var topPane = document.createElement('div');
        var bottomPane = document.createElement('div');
```

```
        var leftPane = document.createElement('div');
        var rightPane = document.createElement('div');
        // centerPane.style.background=twaver.Colors.blue_light;
        topPane.style.background=twaver.Colors.green_light;
        leftPane.style.background=twaver.Colors.orange_light;
        bottomPane.style.background=twaver.Colors.green_light;
        rightPane.style.background=twaver.Colors.orange_light;

        centerPane.innerHTML="Center";
        centerPane.style.textAlign="center";
        centerPane.style.fontSize = "80px";

        topPane.innerHTML="Top";
        topPane.style.textAlign="center";
        topPane.style.fontSize = "20px";

        leftPane.innerHTML="Left";
        leftPane.style.textAlign="center";
        leftPane.style.fontSize = "80px";

        bottomPane.innerHTML="Bottom";
        bottomPane.style.textAlign="center";
        bottomPane.style.fontSize = "20px";

        rightPane.innerHTML="Right";
        rightPane.style.textAlign="center";
        rightPane.style.fontSize = "80px";

        borderPane = new twaver.controls.BorderPane(centerPane,topPane,rightPane,bottomPane,leftPane);
        borderPane.setTopHeight(40);
        borderPane.setLeftWidth(200);
        borderPane.setBottomHeight(40);
        borderPane.setRightWidth(200);

        var splitPaneLeft = document.createElement('div');
        splitPaneLeft.style.background=twaver.Colors.blue_light;

        splitPane =  new twaver.controls.SplitPane(splitPaneLeft,borderPane);
        splitPane.getView().style.position = 'absolute';
        splitPane.getView().style.left = '0px';
        splitPane.getView().style.top = '0px';
        splitPane.getView().style.right = '0px';
        splitPane.getView().style.bottom = '0px';
        splitPane.setPosition(0.2);
        splitPane.setDividerWidth(10);

        document.body.appendChild(splitPane.getView());
        window.onresize = function (e) { splitPane.invalidate(); };
    }


// ]]></script>
```



TabPane

TablePane

TitlePane

ChartPane

LegendPane

图表可视化视图组件(Chart)

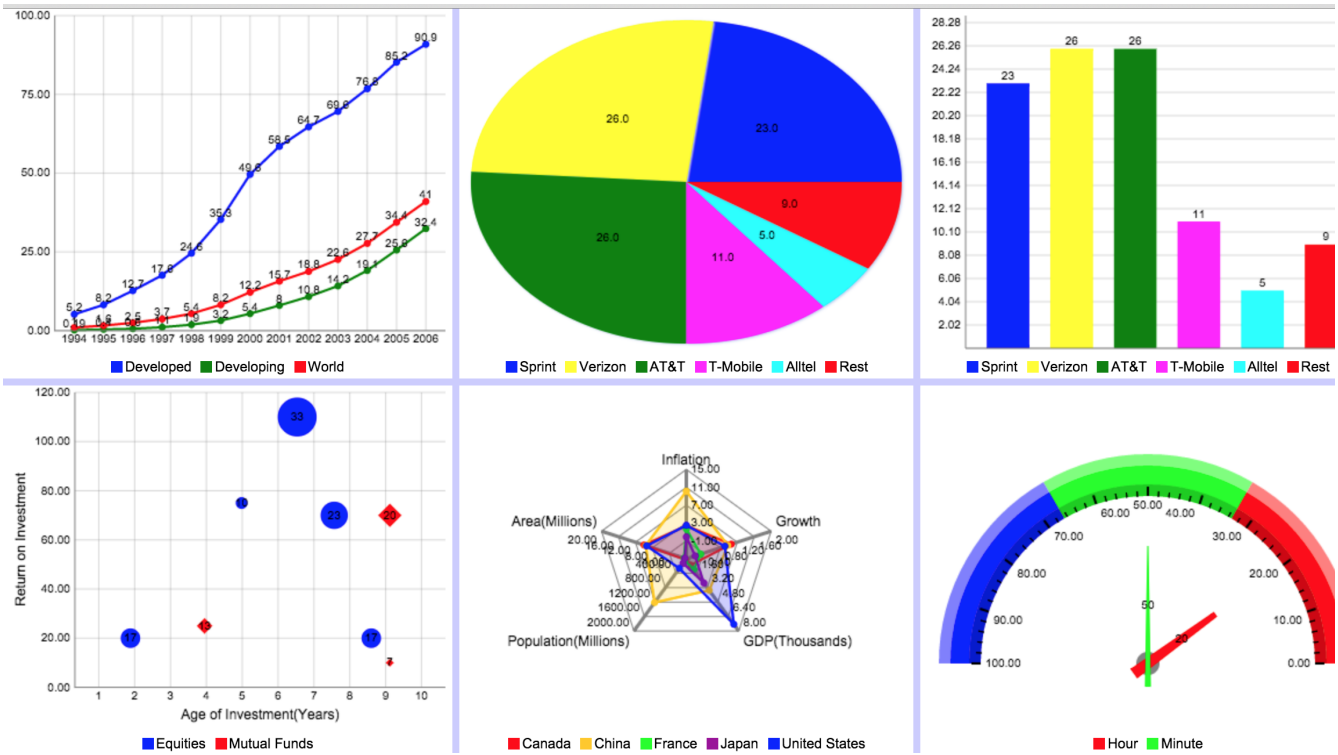
TWaver的图表可视化视图组件主要包括DialChart、PieChart、RadarChart以及继承ScaleChart的barChart、bubbleChart、lineChart,下面示例展示了所有的图表组件。

```
ChartDemo = function () {
    this.lineChart = new twaver.charts.LineChart();
```

```
    this.pieChart = new twaver.charts.PieChart();
    this.barChart = new twaver.charts.BarChart();
    this.bubbleChart = new twaver.charts.BubbleChart();
    this.radarChart = new twaver.charts.RadarChart();
    this.dialChart = new twaver.charts.DialChart();

    this.developedElement = this.createLineChartElement('Developed', 'blue', 'circle');
    this.developingElement = this.createLineChartElement('Developing', 'green', 'triangle');
    this.worldElement = this.createLineChartElement('World', 'red', 'rectangle');

    this.testList = new twaver.List();
    this.developedList = new twaver.List();
    this.developingList = new twaver.List();
    this.worldList = new twaver.List();

    this.equitiesValues = [];
    this.equitiesXAxisValues = [];
    this.equitiesYAxisValues = [];
    this.equitiesNames = [];

    this.mutualFundsValues = [];
    this.mutualFundsXAxisValues = [];
    this.mutualFundsYAxisValues = [];
    this.mutualFundsNames = [];

    this.countries = ['Canada', 'China', 'France', 'Japan', 'United States'];
    this.colors = ['#FF0000', '#FFC800', '#00FF00', '#990099', '#0000FF'];
    this.inflations = [2.4, 10.1, 1.7, -0.1, 2.5];
    this.areas = [9976140, 9596960, 547030, 377835, 9372210];
    this.populations = [28860271, 1210004956, 58317450, 125449703, 266476278];
    this.gdps = [694000, 3500000, 1173000, 2679200, 7247700];
    this.growths = [1.06, 0.98, 0.34, 0.21, 0.91];
    this.axisList = new twaver.List([
          { text: 'Inflation', min: -5, max: 15 },
          { text: 'Area(Millions)', min: 0, max: 20000000 / 1000000 },
          { text: 'Population(Millions)', min: 0, max: 2000000000 / 1000000 },
          { text: 'GDP(Thousands)', min: 0, max: 8000000 / 1000000 },
          { text: 'Growth', min: 0, max: 2 }
        ]);
};


twaver.Util.ext('ChartDemo', Object, {
    init: function () {
        this.initChart();

        var lineChartPane = new twaver.charts.ChartPane(this.lineChart);
        var pieChartPane = new twaver.charts.ChartPane(this.pieChart);
        var barChartPane = new twaver.charts.ChartPane(this.barChart);
        var bubbleChartPane = new twaver.charts.ChartPane(this.bubbleChart);
        var radarChartPane = new twaver.charts.ChartPane(this.radarChart);
        var dialChartPane = new twaver.charts.ChartPane(this.dialChart);

        var topRightSplit = new SplitPane(pieChartPane, barChartPane, 'horizontal', 0.5);
        var topSplit = new SplitPane(lineChartPane, topRightSplit, 'horizontal', 0.33);
        var bottomRightSplit = new SplitPane(radarChartPane, dialChartPane, 'horizontal', 0.5);
        var bottomSplit = new SplitPane(bubbleChartPane, bottomRightSplit, 'horizontal', 0.33);
        var mainSplit = new SplitPane(topSplit, bottomSplit, 'vertical', 0.5);
        demo.Util.appendChild(mainSplit.getView(), document.getElementById('main'), 0, 0, 0, 0);

        window.onresize = function (e) { mainSplit.invalidate(); };
    },
    initChart: function () {
        this.initLineChart();
        this.initPieChart();
        this.initBarChart();
        this.initBubbleChart();
        this.initRadarChart();
        this.initDialChart();
    },
    initLineChart: function () {
        this.lineChart.setYScaleMinTextVisible(true);
        this.lineChart.setYScaleValueGap(25);
        this.lineChart.setLowerLimit(0);
        this.lineChart.setUpperLimit(100);

        this.addValue("1994", 5.2, 1.0, 0.19);
        this.addValue("1995", 8.2, 1.6, 0.4);
        this.addValue("1996", 12.7, 2.5, 0.6);
        this.addValue("1997", 17.6, 3.7, 1.1);
        this.addValue("1998", 24.6, 5.4, 1.9);
        this.addValue("1999", 35.3, 8.2, 3.2);
        this.addValue("2000", 49.6, 12.2, 5.4);
        this.addValue("2001", 58.5, 15.7, 8.0);
        this.addValue("2002", 64.7, 18.8, 10.8);
        this.addValue("2003", 69.6, 22.6, 14.2);
        this.addValue("2004", 76.8, 27.7, 19.1);
        this.addValue("2005", 85.2, 34.4, 25.6);
        this.addValue("2006", 90.9, 41.0, 32.4);

        this.lineChart.setXScaleTexts(this.testList);
        this.developedElement.setStyle('chart.values', this.developedList);
        this.developingElement.setStyle('chart.values', this.developingList);
        this.worldElement.setStyle('chart.values', this.worldList);
    },
    initPieChart: function () {
        this.pieChart.formatValueText = function (value, data) {
            return value.toFixed(1);
        };
        this.createPieChartElement("Sprint", 23, 'BLUE');
        this.createPieChartElement("Verizon", 26, 'YELLOW');
        this.createPieChartElement("AT&amp;T", 26, 'GREEN');
        this.createPieChartElement("T-Mobile", 11, 'MAGENTA');
        this.createPieChartElement("Alltel", 5, 'CYAN');
```

```
        this.createPieChartElement("Rest", 9, 'RED');
    },
    initBarChart: function () {
        this.createBarChartElement("Sprint", 23, 'BLUE', [80, 85, 90, 95]);
        this.createBarChartElement("Verizon", 26, 'YELLOW', [60, 65, 70, 75]);
        this.createBarChartElement("AT&amp;T", 26, 'GREEN', [40, 45, 50, 55]);
        this.createBarChartElement("T-Mobile", 11, 'MAGENTA', [20, 25, 30, 35]);
        this.createBarChartElement("Alltel", 5, 'CYAN', [10, 14, 18, 20]);
        this.createBarChartElement("Rest", 9, 'RED', [3, 5, 7, 9]);
    },
    initBubbleChart: function () {
        this.bubbleChart.setXAxisText('Age of Investment(Years)');
        this.bubbleChart.setYAxisText('Return on Investment');
        this.bubbleChart.setXScaleTexts(new twaver.List(['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']));
        this.bubbleChart.setXAxisLowerLimit(0);
        this.bubbleChart.setXAxisUpperLimit(10);
        this.bubbleChart.setLowerLimit(0);
        this.bubbleChart.setUpperLimit(120);
        this.bubbleChart.setYScaleValueGap(20);
        this.bubbleChart.setYScaleMinTextVisible(true);
        this.bubbleChart.formatValueText = function (value, data) {
            return value.toFixed(0);
        };

        this.addEquitiesValue('A', 6, 110, 100);
        this.addEquitiesValue('B', 1.5, 20, 50);
        this.addEquitiesValue('C', 4.5, 75, 30);
        this.addEquitiesValue('D', 7, 70, 70);
        this.addEquitiesValue('E', 8, 20, 50);
        this.createBubbleChartElement('Equities', 'blue', 'circle',
            this.equitiesValues, this.equitiesXAxisValues, this.equitiesYAxisValues, this.equitiesNames);

        this.addMutualFundsValue('P', 3.5, 25, 40);
        this.addMutualFundsValue('Q', 8.5, 70, 60);
        this.addMutualFundsValue('R', 8.5, 10, 20);

        this.createBubbleChartElement('Mutual Funds', 'red', 'diamond',
            this.mutualFundsValues, this.mutualFundsXAxisValues, this.mutualFundsYAxisValues, this.mutualFundsNames);
    },
    initRadarChart: function () {
        this.radarChart.setAxisStartAngle(90);
        this.radarChart.setAxisList(this.axisList);
        for (var i = 0; i &lt; this.countries.length; i++) {
            this.createRadarChartElement(
                this.countries[i], this.colors[i], this.inflations[i], this.areas[i], this.populations[i], this.gdps[i], this.growt
        }
    },
    initDialChart: function () {
        var colorRangeList = new twaver.List(['#FF0000', '#00FF00', '#0000FF']);
        this.dialChart.setStartAngle(0);
        this.dialChart.setEndAngle(180);
        this.dialChart.setColorRangeList(colorRangeList);
        this.createDialChartElement('Hour', '#FF0000', 20, 0.5, 15, 5, 10);
        this.createDialChartElement('Minute', '#00FF00', 50, 0.7, 20, 1, 5);
    },
    createLineChartElement: function (name, color, shape) {
        var element = new twaver.Element();
        element.setName(name);
        element.setStyle('chart.color', color);
        element.setStyle('chart.mark.shape', shape);
        this.lineChart.getDataBox().add(element);
        return element;
    },
    createPieChartElement: function (name, value, color) {
        var element = new twaver.Element();
        element.setName(name);
        element.setStyle('chart.value', value);
        element.setStyle('chart.color', color);
        this.pieChart.getDataBox().add(element);
        return element;
    },
    createBarChartElement: function (name, value, color, values) {
        var element = new twaver.Element();
        element.setName(name);
        element.setStyle('chart.value', value);
        element.setStyle('chart.color', color);
        element.setStyle('chart.values', values);
        this.barChart.getDataBox().add(element);
        return element;
    },
    createBubbleChartElement: function (name, color, shape, values, xAxisValues, yAxisValues, names) {
        var element = new twaver.Element();
        element.setName(name);
        element.setStyle('chart.color', color)
          .setStyle('chart.bubble.shape', shape)
          .setStyle('chart.values', values)
          .setStyle('chart.xaxis.values', xAxisValues)
          .setStyle('chart.yaxis.values', yAxisValues)
          .setStyle('chart.names', names);
        this.bubbleChart.getDataBox().add(element);
        return element;
    },
    createRadarChartElement: function (name, color, inflation, area, population, gdp, growth) {
        var element = new twaver.Element();
        element.setName(name);
        element.setStyle('chart.color', color)
          .setStyle('chart.values', [inflation, area / 1000000, population / 1000000, gdp / 1000000, growth]);
        this.radarChart.getDataBox().add(element);
        return element;
    },
    createDialChartElement: function (name, color, value, raduis, rearExtension, topWidth, baseWidth) {
        var element = new twaver.Node();
        element.setName(name);
        element.setStyle('chart.color', color)
```

```
            .setStyle('chart.value', value)
            .setStyle('dialchart.radius', raduis)
            .setStyle('dialchart.rear.extension', rearExtension)
            .setStyle('dialchart.top.width', topWidth)
            .setStyle('dialchart.base.width', baseWidth);
        this.dialChart.getDataBox().add(element);
        return element;
    },
    addEquitiesValue: function (name, ageOfInvestment, returnOnInvestment, investment) {
        this.equitiesValues.splice(this.equitiesValues.length, 0, investment / 3);
        this.equitiesXAxisValues.splice(this.equitiesXAxisValues.length, 0, ageOfInvestment);
        this.equitiesYAxisValues.splice(this.equitiesYAxisValues.length, 0, returnOnInvestment);
        this.equitiesNames.splice(this.equitiesNames.length, 0, name);
    },
    addMutualFundsValue: function (name, ageOfInvestment, returnOnInvestment, investment) {
        this.mutualFundsValues.splice(this.mutualFundsValues.length, 0, investment / 3);
        this.mutualFundsXAxisValues.splice(this.mutualFundsXAxisValues.length, 0, ageOfInvestment);
        this.mutualFundsYAxisValues.splice(this.mutualFundsYAxisValues.length, 0, returnOnInvestment);
        this.mutualFundsNames.splice(this.mutualFundsNames.length, 0, name);
    },
    addValue: function (year, developed, world, developing) {
        this.testList.add(year);
        this.developedList.add(developed);
        this.developingList.add(developing);
        this.worldList.add(world);
    }
});
```



常见问题

Chart导出成图片

```
chart.addViewListener(function (evt) {
          if (evt.kind === 'validateEnd') {
               var mycanvas = chart._canvas;
               window.open(mycanvas.toDataURL(), 'network.png');
       }
```

概述

TWaver支持全矢量的图形绘制模式，缩放不失真，也可以同时混合位图和矢量图。Node的Image图片、图元的Icon图标、Attachment附件等均支持矢量模式。

TWaver中，各种图形（位图或矢量图）在使用之前，都需要进行注册，才能使用。使用下面的函数进行注册。

```
/**
* name: 图片的注册名称。后续使用中就使用该name进行引用。例如"preview_icon"、"my_node_image"等;
* source: 图片源。例如: "images/loading.gif";
* width: 图片的注册宽度;
* height: 图片的注册高度;
*/
twaver.Util.registerImage : function (name, source, width, height)
```

使用位图

假设需要使用一张位图'images/preview.png'作为节点图片，则可以用下面的代码进行注册，并使用：

```
//注册图片
twaver.Util.registerImage('preview', 'images/preview.png', 32, 32);

//在node上使用
var node = new twaver.Node();
node.setImage('preview');
```

使用位图的优点是简单方便，位图资源丰富。缺点是缩放会导致图形失真，位图文件的尺寸较大（相比矢量图，一般情况下）。对于一般简单的图形应用，位图可以满足要求。如有"缩放不失真"的要求，请使用矢量图。

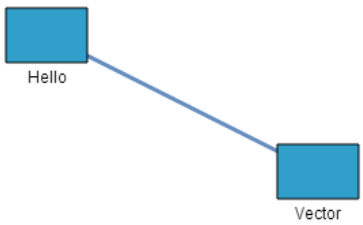
使用矢量图

矢量图是通过描述图形的形状来定义，在绘制时可以根据需要缩放而不会导致失真。SVG、Visio、AutoCAD等文件格式都是典型的矢量格式图形。TWaver支持矢量图形格式，且内置的默认图标均为矢量格式（见下图）：

下面代码注册了一个简单的矢量图，并注册为'test_vector'。Node通过node.setImage('test_vector')即可直接使用：

```
twaver.Util.registerImage('test_vector', {
  w: 60,
  h: 40,
  v: [
  {
    shape: 'rect',
    x: -30,
    y: -20,
```

```
    w: 60,
    h: 40,
    lineColor: 'black',
    fill: '#309FC9',
    lineWidth: 1,
  }
  ]
});

//可以这样创建node
var node1 = new twaver.Node();
node1.setName("Hello");
node1.setLocation(100, 100);
node1.setImage('test_vector');
box.add(node1);

//也可以这样创建node
var node2 = new twaver.Node({
  name: 'Vector',
  location: {
    x: 300,
    y: 200
  },
  image: 'test_vector'
});
box.add(node2);

var link = new twaver.Link(node1, node2);
box.add(link);
```



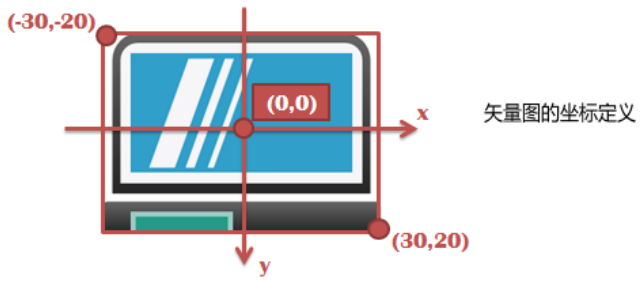
矢量图格式详解

基本结构

TWaver内置了一套简单易用的矢量图形格式，可直接通过json格式进行定义。图形格式是一个js对象，属性主要包括：w（width-宽）、h（height-高）、v（vectors-图形数组）等。基本格式如下：

```
twaver.Util.registerImage('test_vector', {
  w: 60,
  h: 40,
  v: [
  {
    shape: 'rect',
    x: -30,
    y: -20,
    w: 60,
    h: 40,
    lineColor: 'black',
    fill: '#309FC9',
    lineWidth: 1,
    //...更多属性
  },
  //...更多图形
  ]
});
```

矢量图的中心点定义为坐标原点，x、y轴方向分别向右方和下方。如下图：



矢量图形的根属性有：

属性	描述
w	矢量图宽度，如果未提供，则取网元的宽

h	矢量图高度，如果未提供，则取网元的高
cache	是否缓存矢量图，默认值为true。 有动画效果(animate)、状态变化(state)或者动态属性的矢量图无法缓存，为了提高性能建议设置cache为true。 唯一需要设置为false的场景是：如果父矢量引用了无法缓存的子矢量，但子矢量还没有注册时，用这个属性告诉父矢量不要缓存
clip	是否对超出矢量图区域的部分进行裁剪，出于性能考虑，默认值为false。如果为true，则裁剪区域是矢量图片所占的区域；另外clip也可以是一个shape对象，比如{ shape: 'circle', r: 10 }，则裁剪区域是以矢量图片中心为圆心，半径为10的圆。 如果clip为false，而且画出了图形的边界，缩放比例大于1时，网元移动时可能会有残影，这是由于局部刷新导致的。这种情况有2个选择，要么clip为true，要么不要画出边界
origin	矢量图原点坐标，默认值为{ x: 0.5, y: 0.5 }，也即中心点；TWaver也提供转化SVG为TWaver的矢量图形，由于SVG图形的原点为左上角，这时可以设置orgin为{ x: 0, y: 0 }，也即左上角
v	矢量图中包含的图形数组。每个图形的具体定义方法见下方
visible	矢量图是否可见，默认值为true
图形通用属性	矢量图形的全局样式，所有v属性中的图形继承该样式。比如设置fill: 'red'，所有图形除非指定fill属性，否则用'red'填充。详情见下方

clip和origin的用法如下：

```
twaver.Util.registerImage('clip_false', {
  w: 100,
  h: 100,
  clip: false, // optional, default is false
  origin: { x: 0.5, y: 0.5 }, // optional, default is { x: 0.5, y: 0.5 }
  v: [
    {
      shape: 'ellipse',
      cx: 0,
      cy: 0,
      rx: 60,
      ry: 40,
      fill: '#0089C1',
      gradient: {
        type: 'linear.west',
        color: '#61B6D8'
      }
    }
  ]
});

twaver.Util.registerImage('clip_true', {
  w: 100,
  h: 100,
  clip: true,
  origin: { x: 0, y: 0 },
  v: [
    {
      shape: 'ellipse',
      cx: 50,
      cy: 50,
      rx: 60,
      ry: 40,
      fill: '#0089C1',
      gradient: {
        type: 'linear.west',
        color: '#61B6D8'
      }
    }
  ]
});

var clip_false = new twaver.Node();
clip_false.setName('clip: false,\norigin: { x: 0.5, y: 0.5 }');
clip_false.setImage('clip_false');
clip_false.setLocation(100, 80);
box.add(clip_false);

var clip_true = new twaver.Node();
clip_true.setName('clip: true,\norigin: { x: 0, y: 0 }');
clip_true.setImage('clip_true');
clip_true.setLocation(300, 80);
box.add(clip_true);
```

clip: false,
origin: { x: 0.5, y: 0.5 }

clip: true,
origin: { x: 0, y: 0 }

图形通用属性

TWaver提供的基本图形有线条（line）、矩形（rect）、圆形（circle）、椭圆（ellipse）、路径（path）、文字（text）、自定义绘制（draw）和矢量（vector，嵌入其他已定义的矢量图形）。另外TWaver也提供注册自定义图形的功能（twaver.Util.registerShape(name, shapeFunc)）,详情见下方。这些基本图形可以定义线条和填充、渐变、模式填充、阴影、文字以及其他属性：

图形线条和填充属性

属性	描述
lineWidth	线条宽度，默认值为0，也即没有边框
lineColor	线条颜色，默认为黑色(black)
lineCap	线条端点样式，默认值为'butt'，可选值为'butt', 'round', 'square'
lineJoin	线条连接点样式，默认值为'miter'，可选值为'round', 'bevel', 'miter'
lineMiterLimit	线条连接点样式为'miter'时，转角长度，默认值为4
lineDash	线条虚线样式，比如[10, 5]，默认为null，也即实线
lineDashOffset	线条虚线偏移，默认值为0
line	统一设置线条样式，为包含width、color、cap、join、miterLimit、dash和dashOffset属性的对象
fill	填充颜色，默认值空，也即没有填充

```
twaver.Util.registerImage('shape1', {
  w: 100,
  h: 100,
  line: {
    width: 1,
    color: '#EC6C00',
    dash: [ 10, 10 ],
    dashOffset: 10
  },
  fill: '#61B6D8',
  v: [
    {
      shape: 'rect',
      rect: [ -50, -50, 100, 100]
    }
  ]
});

twaver.Util.registerImage('shape2', {
  w: 100,
  h: 100,
  lineWidth: 1,
  lineColor: '#EC6C00',
  lineDash: [ 10, 10 ],
  v: [
    {
      shape: 'rect',
      rect: [ -50, -50, 100, 100]
    }
  ]
});

var shape1 = new twaver.Node();
shape1.setName('shape1');
shape1.setImage('shape1');
shape1.setLocation(100, 80);
box.add(shape1);

var shape2 = new twaver.Node();
shape2.setName('shape2');
shape2.setImage('shape2');
shape2.setLocation(300, 80);
box.add(shape2);
```

图形渐变属性

属性	描述
gradient	渐变类型，可以是字符串（代表预定义的渐变类型，具体参见下表），或者是包含type和color的对象，其中type代表渐变类型gradient，color代表渐变颜色gradientColor，或者是复杂的渐变对象（具体参见下文）
gradientColor	渐变颜色，默认值为'#FFFFFF'

渐变类型描述如下表：

gradient	描述
linear.southwest	从左下角到右上角线性渐变
linear.southeast	从右下角到左上角线性渐变
linear.northwest	从左上角到右下角线性渐变
linear.northeast	从右上角到左下角线性渐变
linear.north	从上到下线性渐变
linear.south	从下到上线性渐变
linear.west	从左到右线性渐变
linear.east	从右到左线性渐变
radial.center	从中心环形渐变
radial.southwest	从左下角环形渐变
radial.southeast	从右下角环形渐变
radial.northwest	从左上角环形渐变
radial.northeast	从右上角环形渐变
radial.north	从上边环形渐变
radial.south	从下边环形渐变
radial.west	从左边环形渐变
radial.east	从右边环形渐变

复杂渐变：

属性	描述
type	渐变类型，可选值为'line'或者'radial'
x1	当渐变类型为'line'时，渐变起始点x坐标
y1	当渐变类型为'line'时，渐变起始点y坐标
x2	当渐变类型为'line'时，渐变结束点x坐标
y2	当渐变类型为'line'时，渐变结束点y坐标
r	当渐变类型为'radial'时，渐变区域的半径
cx	当渐变类型为'radial'时，渐变区域的中心点x坐标
cy	当渐变类型为'radial'时，渐变区域的中心点y坐标
fx	当渐变类型为'radial'时，渐变焦点区域的中心点x坐标
fy	当渐变类型为'radial'时，渐变焦点区域的中心点y坐标
stop	定义渐变的数组，元素为offset和color属性的对象

```
twaver.Util.registerImage('gradient1', {
  w: 100,
  h: 100,
  lineWidth: 1,
  v: [
    {
      shape: 'rect',
      rect: [ -50, -50, 100, 100],
      lineColor: '#EC6C00',
      fill: '#0089C1',
      gradient: 'linear.west',
      gradientColor: '#61B6D8'
    }
  ]
});
twaver.Util.registerImage('gradient2', {
  w: 100,
  h: 100,
  lineWidth: 1,
  v: [
    {
      shape: 'rect',
      rect: { x: -50, y: -50, w: 100, h: 100 },
      lineColor: '#EC6C00',
      gradient: {
        type: 'linear.west',
        color: '#61B6D8'
      },
      fill: '#0089C1'
    }
  ]
});
twaver.Util.registerImage('gradient3', {
  w: 100,
  h: 100,
  lineWidth: 1,
  v: [
    {
      shape: 'rect',
      rect: { x: -50, y: -50, w: 100, h: 100 },
      lineColor: '#EC6C00',
      gradient: {
        type: 'linear',
        x1: -50,
        y1: 0,
        x2: 50,
        y2: 0,
        stop: [
          {
            offset: 1,
            color: '#0089C1'
          },
          {
            offset: 0,
            color: '#61B6D8'
          }
        ]
      }
    }
  ]
});

var gradient1 = new twaver.Node();
gradient1.setImage('gradient1');
gradient1.setLocation(200, 80);
box.add(gradient1);

var gradient2 = new twaver.Node();
gradient2.setImage('gradient2');
gradient2.setLocation(400, 80);
box.add(gradient2);

var gradient3 = new twaver.Node();
gradient3.setImage('gradient3');
gradient3.setLocation(600, 80);
box.add(gradient3);
```



图形模式填充属性

模式填充（pattern）可以用预定义的矢量图形以tile的方式填充其他图形。

属性	描述
pattern	可以是矢量名称或者矢量对象

下面的例子用实心的圆形作为pattern，填充另外一个圆形：

```
var box = new twaver.ElementBox();
var network = new twaver.vector.Network(box);

function init () {
  var view = network.getView();
  document.body.appendChild(view);
  network.adjustBounds({ x: 0, y: 0, width: 1200, height: 700 });
  network.setZoomManager(new twaver.vector.LogicalZoomManager(network, true));

  var img = new Image();
  img.src = './images/floor.png';
  img.onload = function () {
    img.onload = null;
    twaver.Util.registerImage('pattern1', img, img.width, img.height);
    network.invalidateElementUIs();
    setupToolTip();
  };
  twaver.Util.registerImage('pattern2', {
    w: 20,
    h: 20,
    fill: '#57AB9A',
    v: [
      {
        shape: 'circle',
        r: 10
      }
    ]
  });

  twaver.Util.registerImage('shape1', {
    w: 100,
    h: 100,
    origin: {
      x: 0,
      y: 0
    },
    lineWidth: 1,
    lineColor: '#EC6C00',
    pattern: 'pattern1',
    v: [
      {
        shape: 'circle',
        r: 50,
        cx: 50,
        cy: 50
      }
    ]
  });

  twaver.Util.registerImage('shape2', {
    w: 100,
    h: 100,
    lineWidth: 1,
    lineColor: '#EC6C00',
    pattern: 'pattern2',
    v: [
      {
        shape: 'circle',
        r: 50
      }
    ]
  });

  var shape1 = new twaver.Node();
  shape1.setImage('shape1');
  shape1.setLocation(200, 120);
  shape1.setSize(100, 100);
  box.add(shape1);

  var shape2 = new twaver.Node();
  shape2.setImage('shape2');
  shape2.setLocation(400, 120);
  shape2.setSize(100, 100);
  box.add(shape2);

  setupToolTip();
}

function setupToolTip () {
  twaver.Defaults.TOOLTIP_DISMISS_DELAY = 60 * 1000;
  box.forEach(function (element) {
    var image = element.getImage();
    if (typeof image === 'string') {
      image = twaver.Util.getImageAsset(image);
      image = image && image.getImage();
      if (image instanceof Image) {
        image = image.src;
      }
    }
    image = JSON.stringify(image, null, 2);
    element.getImage() && element.setToolTip('<pre>' + image + '</' + 'pre>');
  })
}
```

图形阴影属性

使用阴影可以做出绚丽的光晕效果，具体属性如下：

属性	描述
shadowOffsetX	阴影水平偏移量，默认值为0
shadowOffsetY	阴影垂直偏移量，默认值为0
shadowBlur	阴影模糊度，默认值为0
shadowColor	阴影颜色，默认值为'rgba(0, 0, 0, 0)'
shadow	统一设置阴影样式，为包含offsetX, offsetY, blur和color属性的对象

```
twaver.Util.registerImage('shadow1', {
  w: 41,
  h: 33,
  shadow: {
    offsetX: 2,
    offsetY: 2,
    blur: 5,
    color: '#61B6D8'
  },
  v: [
    {
      shape: 'vector',
      name: 'node_image',
      x: 0,
      y: 0
    }
  ]
});

twaver.Util.registerImage('shadow2', {
  w: 37,
  h: 29,
  shadowOffsetX: 0,
  shadowOffsetY: 0,
  shadowBlur: 5,
  shadowColor: '#61B6D8',
  v: [
    {
      shape: 'vector',
      name: 'node_image',
      x: 0,
      y: 0
    }
  ]
});

var shadow1 = new twaver.Node();
shadow1.setName('shadow1');
shadow1.setImage('shadow1');
shadow1.setLocation(100, 80);
box.add(shadow1);

var shadow2 = new twaver.Node();
shadow2.setName('shadow2');
shadow2.setImage('shadow2');
shadow2.setLocation(300, 80);
box.add(shadow2);
```

图形文字属性

文字属性只对文字有意义。

属性	描述

font	字体
textAlign	水平对齐方式，默认值为'center'，可选值为'left', 'right', 'center'
textBaseline	垂直对齐方式，默认值为'middle'，可选值为'top', 'middle', 'bottom'

```
twaver.Util.registerImage('text', {
  w: 300,
  h: 100,
  font: '20px arial',
  shadow: {
    blur: 5,
    color: '#61B6D8',
  },
  v: [
    {
      shape: 'rect',
      rect: [ -150, -50, 300, 100],
      lineWidth: 1
    },
    {
      shape: 'text',
      text: 'TWaver',
      textAlign: 'left',
      textBaseline: 'top',
      x: -150,
      y: -50
    },
    {
      shape: 'text',
      text: 'HTML5',
      textAlign: 'right',
      textBaseline: 'bottom',
      x: 150,
      y: 50
    }
  ]
});

var text = new twaver.Node();
text.setImage('text');
text.setLocation(300, 80);
box.add(text);
```

图形其他属性

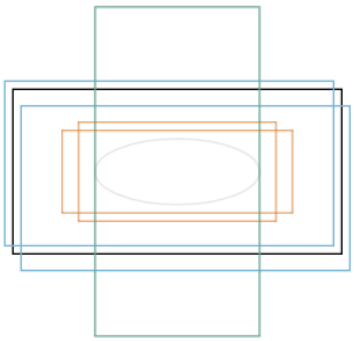
可以控制图形的偏移、旋转、缩放、透明度以及是否可见等。具体属性描述如下：

属性	描述
translate	偏移量，默认为{ x: 0, y: 0 }，x表示水平偏移量，y表示垂直偏移量；如果为数组，则第一个元素表示x，第二个元素表示y；如果为数字，则表示x和y都为相应的值
rotate	旋转角度，单位为度，默认值为0
rotateOrigin	旋转的原点，默认是矢量的原点origin
scale	缩放系数，默认值为{ x: 1, y: 1 }，x表示水平缩放比例，y表示垂直缩放比例；如果为数组，则第一个元素表示x，第二个元素表示y；如果为数字，则表示x和y都为相应的值
alpha	透明度，默认值为1
visible	是否可见，默认值为true
state	状态，可以是字符串或者数组，根据网元的image.state样式决定图形是否可见
rel	坐标和长宽值是否是相对于矢量图形宽高的百分比值，默认是false；如果rel是true，但值大于1或者小于-1，则值是绝对值

```
twaver.Util.registerImage('normal', {
  w: 200,
  h: 200,
  cache: false,
  v: [
    {
      shape: 'rect',
      rel: true,
      rect: [ -0.5, -0.25, 1, 0.5],
      lineWidth: 1
```

```
    }
  ]
});

twaver.Util.registerImage('translate1', {
  w: 200,
  h: 200,
  cache: false,
  translate: -5,
  v: [
    {
      shape: 'rect',
      rect: [ -100, -50, 200, 100],
      lineWidth: 1,
      lineColor: '#61B6D8'
    }
  ]
});

twaver.Util.registerImage('translate2', {
  w: 200,
  h: 200,
  cache: false,
  translate: { x: 5, y: 10 },
  v: [
    {
      shape: 'rect',
      rect: [ -100, -50, 200, 100],
      lineWidth: 1,
      lineColor: '#61B6D8'
    }
  ]
});

twaver.Util.registerImage('rotate', {
  w: 200,
  h: 200,
  cache: false,
  rotate: 90,
  v: [
    {
      shape: 'rect',
      rect: [ -100, -50, 200, 100],
      lineWidth: 1,
      lineColor: '#57AB9A'
    }
  ]
});

twaver.Util.registerImage('scale1', {
  w: 200,
  h: 200,
  scale: 0.6,
  v: [
    {
      shape: 'rect',
      rect: [ -100, -50, 200, 100],
      lineWidth: 1,
      lineColor: '#EC6C00'
    }
  ]
});

twaver.Util.registerImage('scale2', {
  w: 200,
  h: 200,
  scale: { x: 0.7, y: 0.5 },
  v: [
    {
      shape: 'rect',
      rect: [ -100, -50, 200, 100],
      lineWidth: 1,
      lineColor: '#EC6C00'
    }
  ]
});

twaver.Util.registerImage('alhpa', {
  w: 200,
  h: 200,
  alpha: 0.1,
  v: [
    {
      shape: 'ellipse',
      cx: 0,
      cy: 0,
      rx: 50,
      ry: 20,
      lineWidth: 1
    }
  ]
});

twaver.Util.registerImage('visible', {
  w: 200,
  h: 200,
  v: [
    {
      shape: 'circle',
      visible: false,
      cx: 0,
      cy: 0,
```

```
      r: 10,
      lineWidth: 1
    }
  ]
});

[ 'normal', 'translate1', 'translate2', 'rotate', 'scale1', 'scale2', 'alhpa', 'visible' ].forEach(function (shape) {
  var node = new twaver.Node();
  node.setImage(shape);
  node.setLocation(200, 200);
  box.add(node);
});
```



用状态属性控制图形可见：

```
twaver.Util.registerImage('state', {
  w: 100,
  h: 100,
  lineWidth: 1,
  lineColor: '#EC6C00',
  v: [
    {
      shape: 'rect',
      rect: [ -50, -50, 100, 100],
      state: 'rect'
    },
    {
      shape: 'circle',
      r: 50,
      state: 'circle'
    },
    {
      shape: 'ellipse',
      rx: 50,
      ry: 25,
      state: ['rect', 'circle']
    }
  ]
});

var state1 = new twaver.Node();
state1.setImage('state');
state1.setStyle('image.state', 'rect');
state1.setLocation(100, 80);
box.add(state1);

var state2 = new twaver.Node();
state2.setImage('state');
state2.setStyle('image.state', 'circle');
state2.setLocation(300, 80);
box.add(state2);
```

矢量图形

矢量图形的shape属性描述了矢量图形的形状：

shape	描述
line	线条，从起始点（x1,y1或者p1）连一条连线到结束点（x2,y2或者p2）

rect	矩形，从起始点（x,y）画宽为w高为h的矩形
circle	圆形，以圆心（cx,cy）画半径为r的园
ellipse	椭圆，以圆心（cx,cy）画x轴半径为rx、y轴半径为ry的椭圆
path	路径，画数组data描述的轨迹，详细描述见下文
text	文字，在坐标点（x,y）画文字text
draw	自定义绘制，用自定义代码绘制，详细描述见下文
vector	矢量，在坐标点（x,y）画宽为w高为h的矢量vector

线条（line）

属性	描述
x1	起始点x坐标
y1	起始点y坐标
x2	结束点x坐标
y2	结束点y坐标
p1	起始点，可以是包含x和y属性的对象或者是包含两个元素的数组，对应于x1和y1，p1和(x1,y1)二选一
p2	结束点，可以是包含x和y属性的对象或者是包含两个元素的数组，对应于x2和y2，p2和(x2,y2)二选一

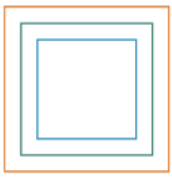
```
1  twaver.Util.registerImage('line', {
2    w: 100,
3    h: 100,
4    line: {
5      width: 1,
6      color: '#EC6C00'
7    },
8    v: [
9      {
10       shape: 'line',
11       x1: -50,
12       y1: -50,
13       x2: 50,
14       y2: 50
15     },
16     {
17       shape: 'line',
18       p1: { x: 50, y: -50 },
19       p2: { x: -50, y: 50 }
20     }
21   ]
22 });
23
24 var line = new twaver.Node();
25 line.setImage('line');
26 line.setLocation(200, 80);
27 box.add(line);
```

矩形（rect）

属性	描述
x	左上角x坐标
y	左上角y坐标
w	宽度
h	高度
rect	和x,y,w及h属性二选一，可以是数组或者带x,y,w和h属性的对象
	圆角半径，可以为Number或者数组。 如果为Number，代表4个角的圆角半径都是指定的值； 如果为数组，元素可以为1、2或者4个。

r	1个元素和Number类似； 2个元素时，第一个值代表左上和右上角的圆角半径，第二个值代表左下和右下角的圆角半径； 4个元素时，分表代表左上、右上、左下和右下角的圆角半径

```
twaver.Util.registerImage('rect', {
  w: 100,
  h: 100,
  v: [
    {
      shape: 'rect',
      x: -50,
      y: -50,
      w: 100,
      h: 100,
      line: {
        width: 1,
        color: '#EC6C00'
      },
    },
    {
      shape: 'rect',
      rect: [ -40, -40, 80, 80 ],
      line: {
        width: 1,
        color: '#238475'
      }
    },
    {
      shape: 'rect',
      rect: { x: -30, y: -30, w: 60, h: 60 },
      line: {
        width: 1,
        color: '#0089C1'
      }
    }
  ]
});

var rect = new twaver.Node();
rect.setImage('rect');
rect.setLocation(200, 80);
box.add(rect);
```



圆形（circle）

属性	描述
cx	圆心x坐标
cy	圆心y坐标
r	半径
startAngle	圆弧开始角度，单位为度，默认值为0度
endAngle	圆弧结束角度，单位为度，默认值为360度
anticlockwise	是否逆时针，默认值为false
close	是否关闭，如果关闭，就会从圆心画两条边到圆弧的两端，默认值为false

```
twaver.Util.registerImage('circle', {
  w: 100,
  h: 100,
  line: {
    width: 1,
    color: '#EC6C00'
  },
  v: [
    {
      shape: 'circle',
      cx: 0,
      cy: 0,
      r: 50
    },
    {
      shape: 'circle',
      r: 30,
```

```
      startAngle: 50,
      endAngle: 130,
      anticlockwise: false
    },
    {
      shape: 'path',
      data: 'M-30,-20h15M10,-20h15'
    }
  ]
});

var circle = new twaver.Node();
circle.setImage('circle');
circle.setLocation(200, 80);
box.add(circle);
```

椭圆（ellipse）

属性	描述
cx	圆心x坐标
cy	圆心y坐标
rx	x轴半径
ry	y轴半径
startAngle	椭圆弧开始角度，单位为度，默认值为0度
endAngle	椭圆弧结束角度，单位为度，默认值为360度
anticlockwise	是否逆时针，默认值为false
close	是否关闭，如果关闭，就会从圆心画两条边到圆弧的两端，默认值为false

```
twaver.Util.registerImage('ellipse', {
  w: 100,
  h: 100,
  v: [
    {
      shape: 'ellipse',
      cx: 0,
      cy: 0,
      rx: 50,
      ry: 25,
      line: {
        width: 1,
        color: '#EC6C00'
      },
    }
  ]
});

var ellipse = new twaver.Node();
ellipse.setImage('ellipse');
ellipse.setLocation(200, 80);
box.add(ellipse);
```

路径（path）

属性	描述
data	可以为数组或者字符串，如果为数组，元素为数字，或者包含x和y属性的对象，，或者包含x、y和c属性的对象，

如果data为数组，数组元素类型可以为：

类型	描述
number	数组元素个数必须为偶数对，data[0]代表点的x坐标，data[1]代表点的y坐标，将这些点用直线连成路径
point	数组中每个元素代表一个点{ x: 0, y: 0 }，将这些点用直线连成路径

{ c: 'command', ... }	数组中每个元素必须有一个c属性，代表命令，根据命令画路径，具体描述见下文

如果数组元素为number或者point，最后一个元素可以为'Z'或者'z'，表示从最后一个点画直线到第一个点

如果data为字符串，格式为：

number,number,... //例如: 10,10,20,20

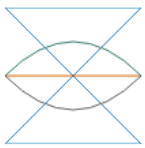
或者

{command}{number},{number}{command}{number},{number}... // 例如: M10,10L20,20

command可选值为：

属性	描述
M	移动到(x, y)坐标点
m	相对上一点，移动到(x, y)坐标点
L	从上一点，画直线到(x, y)坐标点
l	相对上一点，画直线到(x, y)坐标点
H	从上一点，画水平直线到x轴x坐标点
h	相对上一点，画水平直线到x轴x坐标点
V	从上一点，画垂直直线到y轴y坐标点
v	相对上一点，画垂直直线到y轴y坐标点
Q	从上一点，经过控制点(x1, y1)，画二次曲线到(x, y)坐标点
q	相对上一点，经过控制点(x1, y1)，画二次曲线到(x, y)坐标点
C	从上一点，经过控制点(x1, y1)和(x2, y2)，画三次曲线到(x, y)坐标点
c	相对上一点，经过控制点(x1, y1)和(x2, y2)，画三次曲线到(x, y)坐标点
Z或者z	从最后一个点画直线到第一个点

```
twaver.Util.registerImage('path', {
  w: 100,
  h: 100,
  lineWidth: 1,
  v: [
    {
      shape: 'path',
      data: [ -50, -50, 50, -50, -50, 50, 50, 50, 'z' ],
      lineColor: '#0089C1',
    },
    {
      shape: 'path',
      data: [ { x: -50, y: 0 }, { x: 50, y: 0 } ],
      lineColor: '#EC6C00',
    },
    {
      shape: 'path',
      data: [ { c: 'M', x: -50, y: 0 }, { c: 'Q', x1: 0, y1: -50, x: 50, y: 0 } ],
      lineColor: '#238475',
    },
    {
      shape: 'path',
      data: 'M-50,0Q0,50,50,0',
      lineColor: '#666666',
    }
  ]
});

var path = new twaver.Node();
path.setImage('path');
path.setLocation(200, 80);
box.add(path);
```



文字（text）

属性	描述
x	文字x坐标

y	文字y坐标
w	文字最大宽度：如果w>0，文字长度如果超过w，会自动换行，默认值为0
text	文字，用\n可以强制换行

```
twaver.Util.registerImage('text', {
  w: 300,
  h: 100,
  font: '20px arial',
  shadow: {
    blur: 5,
    color: '#61B6D8',
  },
  v: [
    {
      shape: 'rect',
      rect: [ -150, -50, 300, 100],
      lineWidth: 1
    },
    {
      shape: 'text',
      text: 'TWaver',
      textAlign: 'left',
      textBaseline: 'top',
      x: -150,
      y: -50
    },
    {
      shape: 'text',
      text: 'HTML5',
      textAlign: 'right',
      textBaseline: 'bottom',
      x: 150,
      y: 50
    }
  ]
});

var text = new twaver.Node();
text.setImage('text');
text.setLocation(300, 80);
box.add(text);
```

自定义绘制（draw）

属性	描述
draw	可以是function (g, data, view)，或者字符串，字符串可以是通过twaver.Util.registerDraw注册的函数，或者用<%和%>包含的代码

```
twaver.Util.registerDraw('drawFunction', function (g, data, view) {
  g.strokeRect(-20, -20, 40, 40);
});

var draw = new twaver.Node();
draw.setImage({
  w: 300,
  h: 100,
  v: [
    {
      shape: 'draw',
      draw: 'drawFunction'
    },
    {
      shape: 'draw',
      draw: function (g, data, view) {
        g.drawShape({
          shape: 'rect',
          rect: [-10, -10, 20, 20],
          lineColor: '#EC6C00',
          lineWidth: 1
        });
      }
    },
    {
      shape: 'draw',
      draw: '<%g.strokeStyle = "#0089C1";g.strokeRect(-30, -30, 60, 60);%>'
    }
  ]
});
```

```
draw.setLocation(200, 80);
box.add(draw);
```

提示：
网元的图片可以是注册的图片名字外，还可以直接设置为矢量对象，不需要用twaver.Util.registerImage注册，对于只用一次的矢量图片非常方便。

矢量（vector）

shape为vector或image时，可以嵌入其他注册的矢量

属性	描述
x	矢量的x坐标
y	矢量的y坐标
name	矢量的名称

```
twaver.Util.registerImage('X', {
  w: 100,
  h: 100,
  line: {
    width: 1,
    color: '#0089C1'
  },
  v: [
    {
      shape: 'line',
      p1: { x: -50, y: -50 },
      p2: { x: 50, y: 50 }
    },
    {
      shape: 'line',
      p1: { x: 50, y: -50 },
      p2: { x: -50, y: 50 }
    }
  ]
});

var node = new twaver.Node();
node.setImage({
  w: 100,
  h: 100,
  v: [
    {
      shape: 'rect',
      rect: { x: -50, y: -50, w: 100, h: 100 },
      lineWidth: 1,
      lineColor: '#EC6C00'
    },
    {
      shape: 'vector',
      name: 'X'
    }
  ]
});
node.setLocation(200, 80);
box.add(node);
```

自定义图形

可以用twaver.Util.registerShape(name, shapeFunc)注册自定义图形

```
twaver.Util.registerShape('triangle', function (g, shapeData, data, view) {
  var rect = shapeData.rect;
  g.moveTo(rect.x + rect.w / 2, rect.y);
  g.lineTo(rect.x + rect.w, rect.y + rect.h);
```

```
  g.lineTo(rect.x, rect.y + rect.h);
  g.closePath();
});

var shape = new twaver.Node();
shape.setImage({
  w: 100,
  h: 100,
  v: [
    {
      shape: 'triangle',
      rect: { x: -50, y: -50, w: 100, h: 100 },
      lineWidth: 1,
      lineColor: '#EC6C00'
    }
  ]
});
shape.setLocation(200, 80);
box.add(shape);
```

事件

矢量图片可以添加事件，响应鼠标单击（onClick），双击（onDoubleClick），长按（onLongClick），进入（onMouseEnter），移动（onMouseMove）或移出（onMouseLeave）事件。

事件	描述
onClick	function (data, view)，单击事件，单击矢量图片时触发
onDoubleClick	function (data, view)，双击事件，双击矢量图片时触发
onLongClick	function (data, view)，长按事件，长按矢量图片时触发
onMouseEnter	function (data, view)，鼠标进入事件，鼠标进入矢量图片时触发
onMouseMove	function (data, view)，鼠标移动事件，鼠标在矢量图片上移动时触发
onMouseLeave	function (data, view)，鼠标移出事件，鼠标移出矢量图片时触发

下面的例子模拟了鼠标进入时高亮，移出时取消高亮的效果：

```
twaver.Util.registerImage('event', {
  w: 150,
  h: 100,
  rel: true,
  v: [
    {
      shape: 'rect',
      rect: [ -0.5, -0.5, 1, 1 ],
      fill: function (data, view) {
        return data.getClient('focus') ? '#61B6D8' : '#0089C1';
      }
    }
  ],
  onMouseEnter: function (data, view) {
    data.setClient('focus', true);
  },
  onMouseLeave: function (data, view) {
    data.setClient('focus', false);
  }
});

var node1 = new twaver.Node();
node1.setImage('event');
node1.setClient('focus', true);
node1.setName('Focused');
node1.setLocation(200, 80);
box.add(node1);
var node2 = new twaver.Node();
node2.setImage('event');
node2.setLocation(400, 80);
box.add(node2);
```

动态属性

矢量图形中的所有属性，除了可以在定义时给出具体的值，也可以动态读取网元中的属性。

动态属性有三种方式：<%= expression %>、<% code %>和function (data, view)。

<%= expression %>中，expression表示表达式，用表达式的值作为动态属性的值。

<% code %>中，code表示代码，用return语句返回动态属性的值。

这三种方式各有优缺点：
<%= expression %>和<% code %>：有利于图片直接序列化，反序列化时会自动注册序列化字符串中的图片，但这种方式运行效率很低，对效率要求高的场景谨慎使用。
function (data, view)：这种方式注册的图片无法序列化，反序列化时需要再次注册图片，但运行效率高。

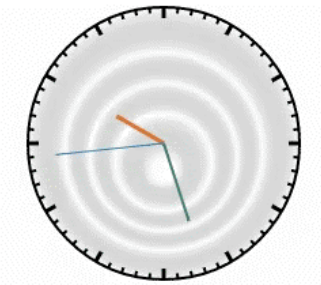
expression和code中能访问的变量如下表：

变量	描述
view	拓扑对象network，比如访问network的当前缩放比例：view.getZoom()
data	网元，可以通过data.getClient('date')或者直接getClient('date')访问网元的属性
this	矢量对象

下面是用动态属性实现的时钟：

```
1  twaver.Util.registerDraw('scale', function (g, data, view) {
2    for (var i = 0; i < 60; i++) {
3      var lineWidth, x;
4      if (i % 5 === 0) {
5        lineWidth = 3;
6        x = 92;
7      } else {
8        lineWidth = 2;
9        x = 96;
10     }
11     g.beginPath();
12     g.lineWidth = lineWidth;
13     g.moveTo(x, 0);
14     g.lineTo(100, 0);
15     g.stroke();
16     g.rotate(Math.PI * 2 / 60);
17   }
18 });
19
20 twaver.Util.registerImage('clock', {
21   w: 200,
22   h: 200,
23   v: [
24     {
25       shape: 'circle',
26       cx: 0,
27       cy: 0,
28       r: 100,
29       lineWidth: 2,
30       gradient: {
31         type: 'radial',
32         cx: 0,
33         cy: 0,
34         fx: 0,
35         fy: 20,
36         r: 100,
37         stop: [
38           {
39             offset: 0,
40             color: 'rgba(102, 102, 102, 0)'
41           },
42           {
43             offset: 0.3,
44             color: '#FFFFFF'
45           },
46           {
47             offset: 0.5,
48             color: 'rgba(102, 102, 102, 0)'
49           },
50           {
51             offset: 0.7,
```

```
            color: '#FFFFFF'
          },
          {
            offset: 1,
            color: 'rgba(102, 102, 102, 0)'
          },
        ]
      }
    },
    {
      shape: 'draw',
      draw: 'scale'
    },
    {
      shape: 'line',
      x1: 0,
      y1: 0,
      x2: 0,
      y2: -40,
      rotate: '<%= getClient("date").getHours() % 12 / 12 * 180 * 2 %>',
      lineWidth: 3,
      lineColor: '#EC6C00'
    },
    {
      shape: 'line',
      x1: 0,
      y1: 0,
      x2: 0,
      y2: -60,
      rotate: '<%= getClient("date").getMinutes() / 60 * 180 * 2 %>',
      lineWidth: 2,
      lineColor: '#238475'
    },
    {
      shape: 'line',
      x1: 0,
      y1: 0,
      x2: 0,
      y2: -80,
      rotate: '<%= getClient("date").getSeconds() / 60 * 180 * 2 %>',
      lineWidth: 1,
      lineColor: '#0089C1'
    }
  ]
});

var clock = new twaver.Node();
clock.setImage('clock');
clock.setLocation(200, 100);
clock.setClient('date', new Date());
box.add(clock);

setInterval(function () {
  clock.setClient('date', new Date());
}, 1000);
```



综合演练：一个动态表盘（Guage）

下面的例子用以上介绍的功能实现了仪表盘的功能：

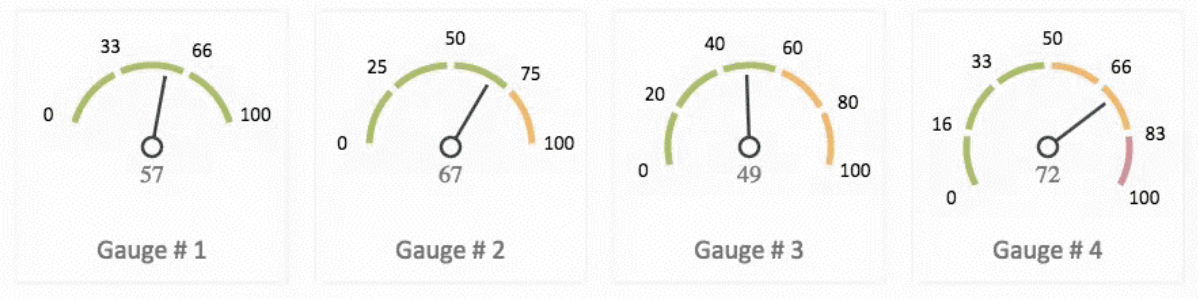
- 可以设置表盘的分段数（client属性section），表盘的圆弧范围（client属性range），表盘的标题（client属性title）以及表盘的值（client属性value）。
- 表盘的分段数由于是变化的，很难用普通的矢量图形描述，所以是用自定义绘制(draw)实现的。

有兴趣的同学可以尝试给这个仪表盘增加更多功能，或者实现其他比如曲线图、柱状图等图表功能。

```
twaver.Util.registerImage('gauge', {
  w: 200,
  h: 200,
  font: '15px Calibri',
  textAlign: 'center',
  textBaseline: 'middle',

  _radius: 60,
  _sectionColors: ['#A6C567', '#A6C567', '#A6C567', '#FCBB69', '#FCBB69', '#E19094', ],
  _sectionGap: 5,
  _min: 0, //gauge min value.
  _max: 100, //gauge max value.

  getSection: function(data) {
    return data.getClient('section');
  },
```

```
  getRange: function(data) {
    return data.getClient('range') || 280;
  },
  getFromAngle: function(data) {
    return (360 - this.getRange(data)) / 2 + 90;
  },
  getToAngle: function(data) {
    return this.getFromAngle(data) + this.getRange(data);
  },
  getValueAngle: function(data) {
    var valueRange = this._max - this._min;
    var value = data.getClient('value') || 0;
    value = value > valueRange ? valueRange : value;
    value = value < 0 ? 0 : value;
    var angle = value / valueRange * this.getRange(data);
    return angle + this.getFromAngle(data);
  },

  v: [
    {
      shape: 'rect',
      rect: [-100, -100, 200, 200],
      lineColor: '#EEEEEE',
      lineWidth: 1,
    },
    {
      shape: 'line',
      x2: '<%= this._radius-7 %>',
      rotate: '<%= this.getValueAngle(data) %>',
      lineColor: '<% if(getClient("value")>=this._max){return "#E19094";}else{return "#43474B";} %>',
      lineWidth: 2.5,
    },
    {
      shape: 'circle',
      r: 7,
      lineColor: '#43474B',
      lineWidth: 2.5,
      fill: 'white',
    },
    {
      shape: 'draw',
      draw: function(g, data, view) {
        g.lineWidth = 5;
        var section = this.getSection(data);
        var sectionAngle = this.getRange(data) / section;
        for (var i = 0; i < section; i++) {
          var from = this.getFromAngle(data) + i * sectionAngle + this._sectionGap / 2;
          var to = from + sectionAngle - this._sectionGap;
          g.strokeStyle = this._sectionColors[i];
          g.beginPath();
          g.arc(0, 0, this._radius, from * Math.PI / 180, to * Math.PI / 180);
          g.stroke();
          g.closePath();

          var textAngle = from * Math.PI / 180;
          var textRadius = this._radius + 20;
          var text = parseInt(this._min + i * (this._max - this._min) / section);
          g.fillText(text, textRadius * Math.cos(textAngle), textRadius * Math.sin(textAngle));

          if (i == section - 1) {
            textAngle = to * Math.PI / 180;
            text = this._max;
            g.fillText(text, textRadius * Math.cos(textAngle), textRadius * Math.sin(textAngle));
          }
        }
      }
    },
    {
      shape: 'text',
      y: 20,
      text: '<%= getClient("value") %>',
      fill: '#777777',
      font: '18px Calibri bold',
    },
    {
      shape: 'text',
      y: 75,
      text: '<%= getClient("title") %>',
      fill: '#777777',
      font: '20px Calibri',
    }
  ]
});

var box = new twaver.ElementBox();
var network = new twaver.vector.Network(box);
document.body.appendChild(network.getView());
network.adjustBounds({
  x: 0,
  y: 0,
  width: 1200,
  height: 700
});
network.setZoomManager(new twaver.vector.LogicalZoomManager(network, true));

for (var i = 0; i < 4; i++) {
  var node = new twaver.Node();
  node.setClient('title', 'Gauge # ' + (i + 1));
  node.setImage('gauge');
  node.setLocation(200 + 220 * i, 200);
  node.setClient('value', parseInt(Math.random() * 100));
  node.setClient('range', 150 + 30 * i);
```

```
    node.setClient('section', i + 3);
    node.setClient('step', 1);
    box.add(node);

    window.node = node;
}

setInterval(function() {
    var value = node.getClient('value');
    var step = node.getClient('step');
    if (value > 120 || value < -20) {
      step = -step;
      node.setClient('step', step);
    }
    value += step;
    node.setClient('value', value);
}, 20);
```



Network缩放概述

TWaver Network组件提供了完整的缩放功能。缩放，是指通过鼠标滚轮、快捷键或API对network画布进行放大或缩小操作时，画布上图形相应的放大或缩小变化。一般而言，缩放操作会使画布上的所有图形做出等比例的放大和缩小。在使用TWaver矢量图的情况下，画布上所有的图形元素在缩放下都不会产生失真。在实际应用中，“按比例缩放”并不能完全满足各种应用场景。例如：一幅“两点一线”的图形，在放大10倍的情况下，连线的粗细是否也放大10倍还是保持不变？节点的图标尺寸是放大10倍还是2倍还是保持不变？TWaver通过“缩放策略”来定义缩放规则，让每个元素的缩放行为交给开发者进行控制。 TWaver缩放策略包括物理缩放模式、逻辑缩放模式、混合缩放模式。

- 物理缩放模式（默认）：是对于画布的直接缩放。呈现结果是：节点的尺寸、连线的线宽，都会按zoom等比例进行变化；
- 逻辑缩放模式：不对画布进行缩放，而是对节点的位置进行逻辑上的缩放。节点的线宽都保持不变，节点的尺寸由sizeChange参数控制是否保持不变；
- 混合缩放模式：物理缩放和逻辑缩放的混合。当zoom>1的时候，采用逻辑缩放策略，反之用物理缩放的策略;

Note:

当前缩放支持居中缩放，左上角缩放，以及鼠标点缩放，默认为鼠标点缩放。

Network缩放策略

调用Network的setZoomManager即可设置不同的缩放策略，物理缩放，逻辑缩放，混合模式，分别对应下面的代码：

```
//set physical zoom policy
network.setZoomManager(new twaver.vector.PhysicalZoomManager(network));

//set logical zoom policy
network.setZoomManager(new twaver.vector.LogicalZoomManager(network,false));

//set mixed zoom policy
network.setZoomManager(new twaver.vector.MixedZoomManager(network,false));
```

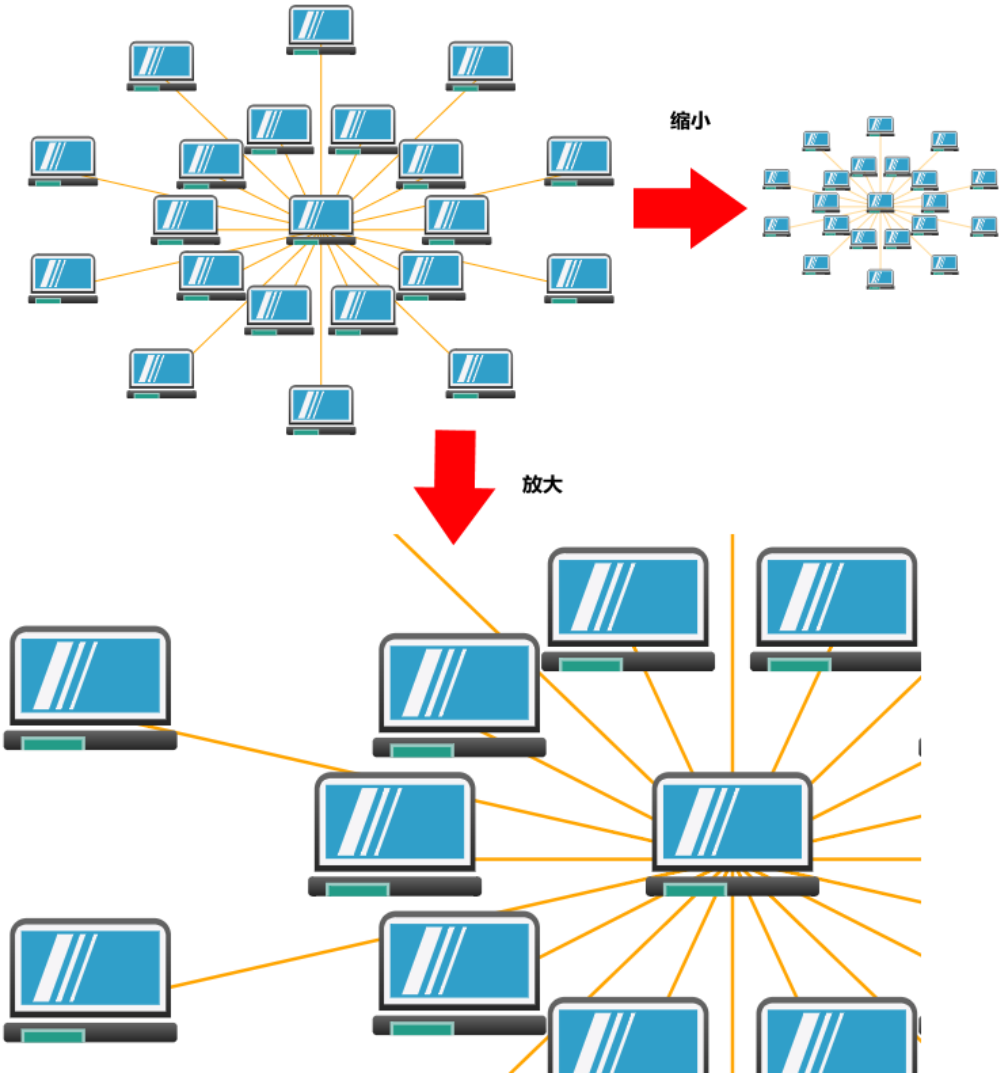
为了演示几种缩放模式的显示效果，下面的demo创建了一个简单的网络图形：

```
TWaver HTML5<script src="twaver.js"></script>
<script>
function init() {
    var box = new twaver.ElementBox();
    var network = new twaver.vector.Network(box);

    document.body.appendChild(network.getView());
    network.adjustBounds({x:0, y:0, width:700, height:700});

    var x=300, y=300, r=200;
    var center = new twaver.Node();
    center.setLocation(x, y);
    box.add(center);

    var count=20;
    for(var i=0;i<count; i++){
        var node = new twaver.Node();
        var angle = i * 2 * Math.PI / count ;
        var radius = r*( i%2 * 0.5 + 0.5);
        node.setLocation(x+radius*Math.cos(angle), y+radius*Math.sin(angle));
        node.setLayerId('node');
        box.add(node);
  var link = new twaver.Link(center, node);
        link.setStyle('link.width', 1);
        link.setStyle('link.color', 'orange');
        link.setLayerId('link');
        box.add(link);
    }
    var linkLayer=new twaver.Layer('link');
    box.getLayerBox().add(linkLayer);
    box.getLayerBox().moveToTop(linkLayer);   }
</script>
```

物理缩放

在默认或使用下面代码进行设置的情况下，物理缩放模式运行效果如下图：

```
nodenetwork.setZoomManager(new twaver.vector.PhysicalZoomManager(network));
```

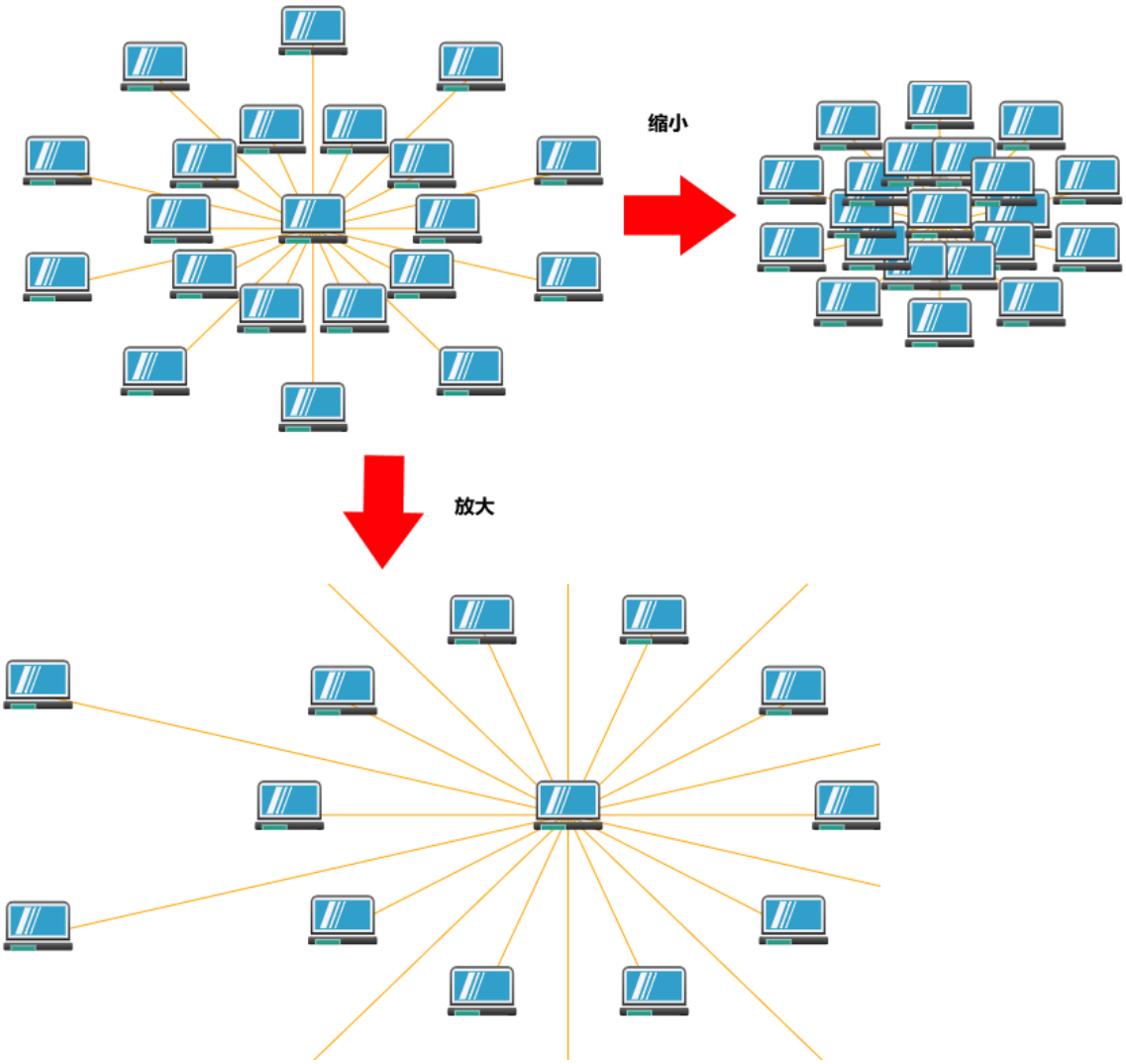


逻辑缩放

下面代码设置逻辑缩放：

```
1 network.setZoomManager(new twaver.vector.LogicZoomManager(network));
```

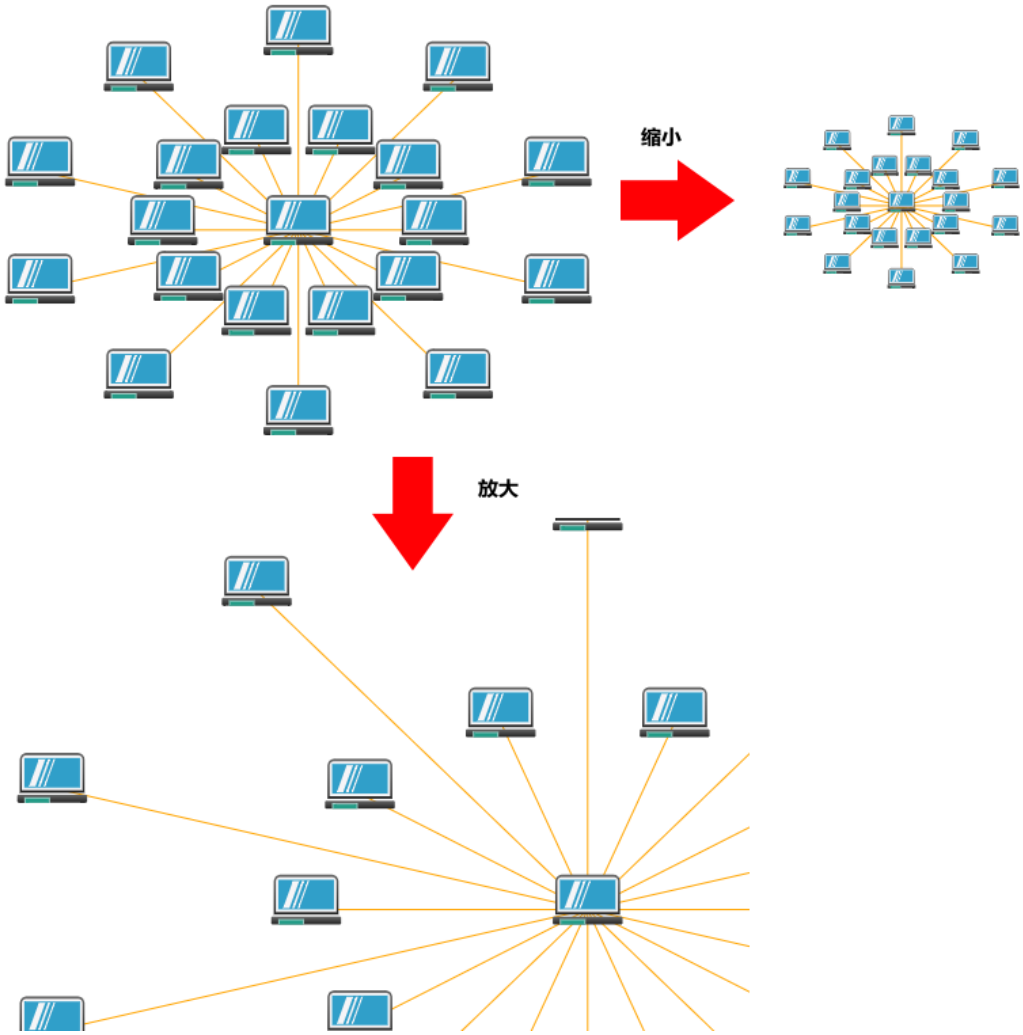
效果如下图：



混合缩放

下面代码设置混合缩放：

```
1 network.setZoomManager(new twaver.vector.MixedZoomManager(network));
```



效果如下图：

缩放值

Network的缩放值，可以通过Network的getZoom方法获取，针对前面不同的缩放模式，我们也提出了相应的不通用的缩放值的概念，包括Graphics Zoom，Size Zoom， Location Zoom，不同缩放模式下，这些值的取值如下：

缩放模式	取值
物理缩放	graphicsZoom = zoom,sizeZoom =1,locationZoom=1
逻辑缩放	graphicsZoom = 1,sizeChange参数为true，sizeZoom = zoom反之sizeZoom =1,locationZoom=zoom

可以通过API获取这几种Zoom：

```
network.getGraphicsZoom();
network.getSizeZoom();
network.getLocationZoom();
```

缩放的可见阀值

缩放的可见阀值（VisibilityThreshold）可以控制元素在某个zoom值以下不可见，这些元素包括连线，Element的Label，Element 的告警冒泡，Element的其他附件。 比如可以这样设置：

```
network.setZoomVisibilityThresholds({
  zoomName : 'sizeZoom',
  link : 0.5,
  label : 0.8,
  label2 : 0.6,
  alarmBallon : 0.5,
});
```

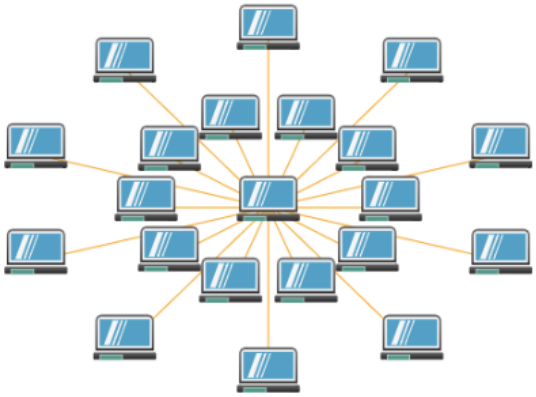
以上代码表示以sizeZoom为标准，当值低于0.8的时候，label不显示，低于0.6时，label2不显示，低于0.5时，告警冒泡不显示，并且link不显示。 我们可以写一个实例来体现这种效果，在上面的例子中，首先给Node和Link设置上Name标签，在原来代码中增加代码如下：

```
node.setName('node' + i);
link.setName('link' + i);
```

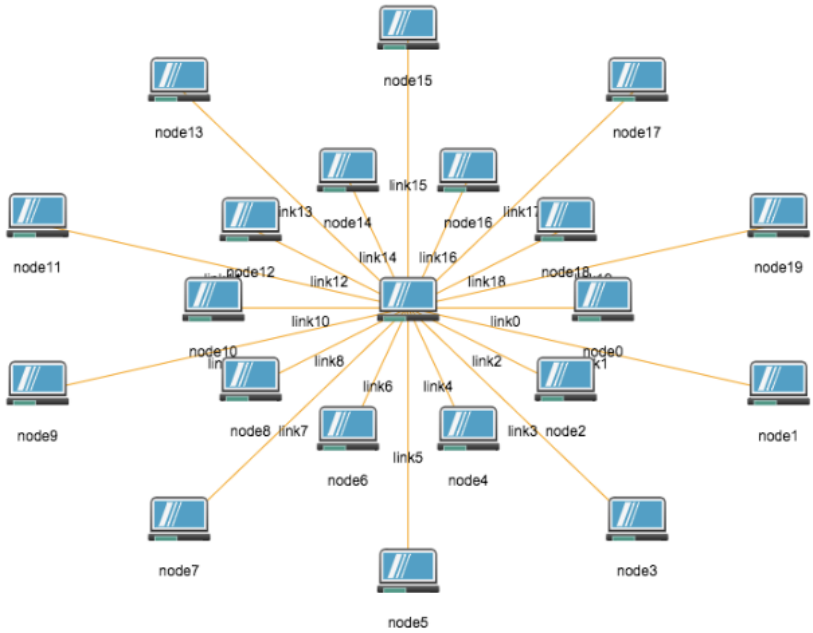
然后设置Network的ZoomVisibilityThreshold，代码如下：

```
network.setZoomVisibilityThresholds({
          label : 1.5,
        });
```

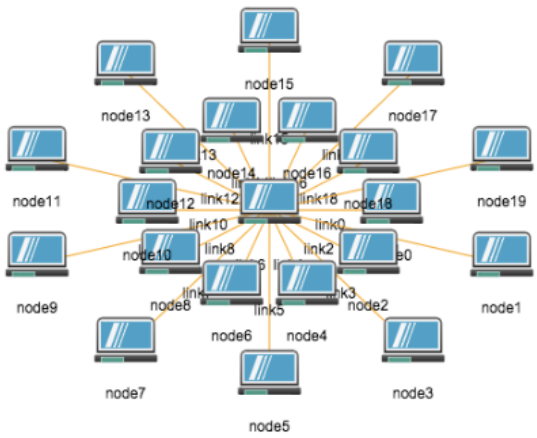
上面代码说明，当zoom小于1.5的时候Name标签不显示，大于1.5的时候Name标签显示下面是截图： zoom为1的情况（也就是原始情况）：



zoom大于1.5的情况：



ZoomVisibilityThreshold的设计目的就是为了让一些元素在zoom比较小的情况下不显示，因为在zoom比较小得情况下，显示太多元素，反而显得比较乱，比如我们不设置ZoomVisibilityThreshold，在来看原始（zoom=1）的情况：



可以看出，很多元素叠加在一起，看起来比较混乱。

缩放交互

在twaver.vector.interaction.DefaultInteraction中，提供了通过鼠标滚轮来改变network的zoom值，而且总是以鼠标点的位置为中心在缩放，而不是以固定的左上角或者network中心点为缩放中心点。

定制SizeZoom

在逻辑缩放的模式下，用一个参数sizeChange可以控制节点的大小是否随着zoom值的变化而变化，但是这种方式是统一控制的，针对所有节点，要么所有节点的SizeZoom随zoom变化而，要么所有节点的SizeZoom保持大小不变，比较单调；LogicZoomManager提供了可以重写的接口函数getSizeZoom，定制节点的Size随zoom的变化：

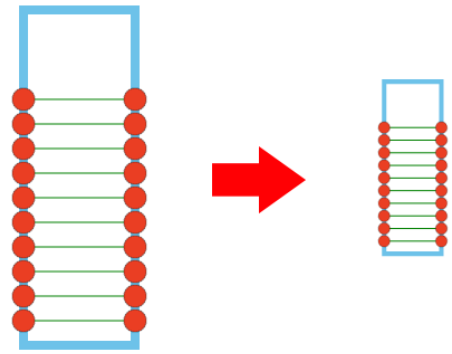
```
  zoomManager.getSizeZoom = function(ui){
      var element = ui.getElement();
      var zoom=this.getZoom();
      if(element.getClient('zoom')===false &amp;amp;&amp;amp; zoom&amp;gt;1){
        return 1;
      }
      return zoom;
    };
```

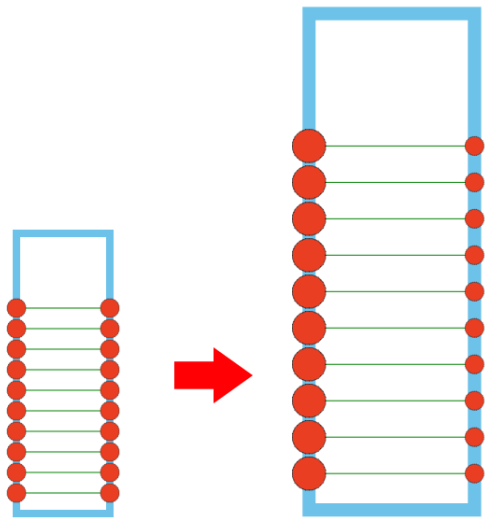
方法可以得到一个ui参数，通过ui可以得到element，这样可以定制不同的element的Size随zoom的变化。
下面是一个完整demo：

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta http-equiv="pragma" content="no-cache" charset="UTF-8">
    <script type="text/javascript" src="../twaver.js"></script>
    <script type="text/javascript">
        function init() {
            twaver.Util.registerImage('chip_body', {
                w: 100,
                h: 300,
                v: [
                    {
                        shape: 'rect',
                        rect: [ -50, -150, 100, 300],
                        lineColor: twaver.Colors.blue_dark,
                        lineWidth: 10,
                    },
                ]
            });

            twaver.Util.registerImage('chip_foot', {
                w: 20,
                h: 20,
                v: [
                    {
                        shape: 'circle',
                        cx: 0,
                        cy: 0,
                        r: 10,
                        lineColor: 'black',
                        fill: twaver.Colors.orange_light,
                        lineWidth: 0.3,
                    },
                ]
            });

            var box = new twaver.ElementBox();
            var network = new twaver.vector.Network(box);
            document.body.appendChild(network.getView());
            network.adjustBounds({x: 0, y: 0, width: 1200, height: 700});
            var zoomManager = new twaver.vector.LogicalZoomManager(network, true);
            network.setZoomManager(zoomManager);
            zoomManager.getSizeZoom = function (ui) {
                if (ui) {
                    var element = ui.getElement();
                    var zoom = this.getZoom();
                    if (element.getClient('zoom') === false && zoom > 1) {
                        return 1;
                    }
                }
                return zoom;
            };

            network.setEditInteractions();
            var body = new twaver.Node();
            body.setImage('chip_body');
            body.setLocation(160, 100);
            box.add(body);

            function createNode(box, x, y, zoom) {
                var node = new twaver.Follower();
                node.setImage('chip_foot');
                node.setClient('zoom', zoom);
                node.setHost(body);
                node.setLocation(x, y);
                box.add(node);
```

```
                return node;
            }

            for (var i = 0; i < 10; i++) {
                var left = createNode(box, 150, 130 + i * 22, true);
                var right = createNode(box, 250, 130 + i * 22, false);
                var link = new twaver.Link(left, right);
                link.setStyle('link.width', 2);
                link.setStyle('link.color', twaver.Colors.green_light);
                box.add(link);
            }
        }

    </script>

</head>
<body onload="init()">
</body>
</html>
```

下面是截图,可以看出，当zoom>1左边的元素随着zoom变化而变化，右边的元素始终大小不变，而当zoom<1的时候，所有元素都随着zoom变化而变化



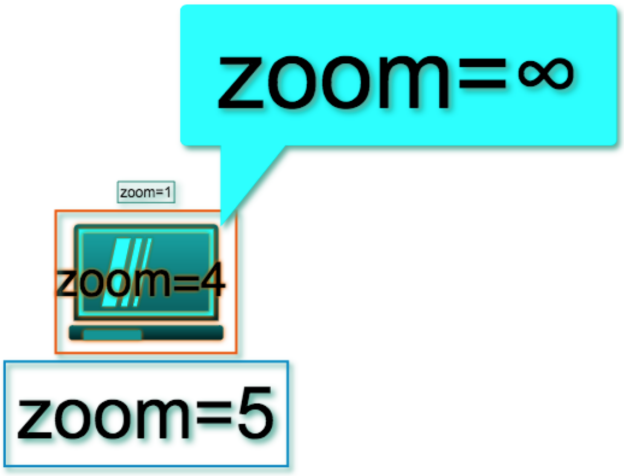


除此以外，还可以控更耕细粒度的附件元素，可以通过重写方法getAttachmentSizeZoom实现：

```
CustomZoomManager = function (network, sizeChange) {
    CustomZoomManager.superClass.constructor.apply(this, arguments);
};

_twaver.ext(CustomZoomManager, twaver.vector.LogicalZoomManager, {
    getSizeZoom: function (ui) {
        var zoom = this.getZoom();
        if (ui) {

            var element = ui.getElement();
            return zoom > 4 ? 4 : zoom;
        }
        return zoom;
    },
    getAttachmentSizeZoom: function (attachment) {
        var zoom = this.getZoom();
        if (attachment instanceof  twaver.vector.AlarmAttachment) {
            return zoom;
        } else if (attachment instanceof  twaver.vector.LabelAttachment) {
            return zoom > 5 ? 5 : zoom;
        } else if (attachment instanceof  twaver.vector.Label2Attachment) {
            return 1;
        }
```

```
25          return zoom;
26      }
27  });
```

效果如下图所示，其中body的zoom>4时候固定为4，name的zoom>5时候固定为5，name2一直保持为1，而告警会一直随着zoom的变化而变化。



概述

TWaver HTML5网元的所有图形属性都是通过设置样式属性来实现的，本章将枚举各种网元类型的样式名，赋值范围和使用说明。

Twaver HTML5中的样式设置方法如下：

```
//1.全局设置网元样式,一次设置,全局使用
twaver.Styles.setStyle(style,value);
twaver.Styles.s(style,value);

//2.局部设置网元样式
node.setStyle(type,value);
node.s(type,value);
link.setStyle(type,value);
link.s(type,value);
```

alarm-告警冒泡相关样式

样式名	赋值范围	说明
alarm.alpha	类型:Number 默认值为:1	告警冒泡透明度
alarm.cap	类型:String 默认值为:butt 取值范围:butt,round,square	边框线端点样式
alarm.color	类型:Color 默认值为:#000000	填充色
alarm.corner.radius	类型:Number 默认值为:5	圆角半径
alarm.direction	类型:String 默认值为:aboveright 取值范围：aboveleft, aboveright, belowleft, belowright, leftabove, leftbelow,rightabove, rightbelow, above, below, left, right	冒泡方向
alarm.gradient	类型:String 默认值为:none 取值范围:linear.east, linear.north, linear.northeast, linear.northwest, linear.south, linear.southeast, linear.southwest, linear.west, none,radial.center, radial.east,radial.north, radial.northeast, radial.northwest,radial.south, radial.southeast,radial.southwest, radial.west, spread.antidiagonal, spread.diagonal,spread.east,spread.horizontal, spread.north, spread.south, spread.vertical, spread.west	渐变填充颜色
alarm.gradient.color	类型:Color 默认值:#FFFFF	渐变填充色
alarm.join	类型:String 默认值: miter 取值范围:miter, round, bevel	边框线拐点样式
alarm.outline.color	类型:Color 默认值:#000000	边框线颜色
alarm.outline.width	类型:Number 默认值:-1	边框线宽度
alarm.padding	类型:Number 默认值:0	间隙宽度
alarm.padding.bottom	类型:Number 默认值:0	底部间隙
alarm.padding.left	类型:Number 默认值:0	左边间隙

alarm.padding.right	类型:Number 默认值:0	右边间隙
alarm.padding.top	类型:Number 默认值:0	顶点间隙
alarm.pointer.length	类型:Number 默认值:10	指针长度
alarm.pointer.width	类型:Number 默认值:8	指针宽度
alarm.position	类型:String 默认值:hotspot 取值范围：topleft.topleft, topleft.topright, top.top, topright.topleft, topright.topright, topleft, top, topright, topleft.bottomleft, topleft.bottomright, top.bottom, topright.bottomleft, topright.bottomright, left.left, left, left.right, center, right.left, right, right.right, bottomleft.topleft, bottomleft.topright, bottom.top, bottomright.topleft, bottomright.topright, bottomleft, bottom, bottomright, bottomleft.bottomleft, bottomleft.bottomright, bottom.bottom, bottomright.bottomleft, bottomright.bottomright	冒泡位置
alarm.shadowable	类型:String 默认值:true	冒泡阴影
alarm.xoffset	类型:Number 默认值:0	x偏移量
alarm.yoffset	类型:Number 默认值:0	y偏移量

arrow-箭头样式

样式名	赋值范围	说明
arrow.from	类型:Boolean 默认值:false	是否需要起始箭头
arrow.from.fill	类型:Boolean 默认值:true	起始箭头是否需要填充
arrow.from.shape	类型:String 默认值:arrow.standard 取值范围:arrow.standard, arrow.delta, arrow.diamond, arrow.short, arrow.slant	起始箭头的形状
arrow.from.color	类型:Color 默认值:#000000	起始箭头的颜色值
arrow.from.xoffset	类型:Number 默认值:#000000	起始箭头X轴上的偏移
arrow.from.yoffset	类型:Number 默认值:0	起始箭头Y轴上的偏移
arrow.from.width	类型:Number 默认值:12	起始箭头的宽度
arrow.from.height	类型:Number 默认值:9	起始箭头的高度
arrow.from.outline.color	类型:Color 默认值:#000000	起始箭头外边框的颜色
arrow.from.outline.width	类型:Number 默认值:-1	起始箭头外边框的宽度
arrow.from.at.edge	类型:Boolean 默认值:true	起始箭头是否在边缘
arrow.to	类型:Boolean	是否需要结束箭头

	默认值:false	
arrow.to.fill	类型:Boolean 默认值:true	结束箭头是否需要填充
arrow.to.shape	类型:String 默认值:arrow.standard 取值范围:arrow.standard, arrow.delta, arrow.diamond, arrow.short, arrow.slant	结束箭头的形状
arrow.to.color	类型:Color 默认值:#000000	结束箭头的颜色值
arrow.to.xoffset	类型:Number 默认值:0	结束箭头X轴上的偏移
arrow.to.yoffset	类型:Number 默认值:0	结束箭头Y轴上的偏移
arrow.to.width	类型:Number 默认值:12	结束箭头的宽度
arrow.to.height	类型:Number 默认值:9	结束箭头的高度
arrow.to.outline.color	类型:Color 默认值:#000000	结束箭头外边框的颜色
arrow.to.outline.width	类型:Number 默认值:-1	结束箭头外边框的高度
arrow.to.at.edge	类型:Boolean 默认值:true	结束箭头是否在边缘

body-网元主体样式

样式名	赋值范围	说明
body.type	类型:String 默认值:default 取值范围：none', 'default', 'vector', 'default.vector', 'vector.default'	网元主体类型

bus-总线样式

样式名	赋值范围	说明
bus.style	类型:String 默认值:nearby 取值范围：'nearby', 'north', 'south', 'west', 'east'	总线类型

chart-图标样式

样式名	赋值范围	说明
chart.bubble.shape	类型:String 默认值:circle 取值范围:rectangle, oval, roundrect, star, triangle, circle, hexagon, pentagon, diamond	气泡形状
chart.line.width	类型:Number 默认值:2	线宽度
chart.marker.shape	类型:String 默认值:circle 取值范围：rectangle, oval, roundrect, star, triangle, circle, hexagon, pentagon, diamond	标记形状
chart.marker.size	类型:Number 默认值:6	标记尺寸

chart.value	类型:Number 默认值:0	图标数值
chart.value.color	类型:Color 默认值:#000000	图标颜色

dialchart-仪表盘样式

样式名	赋值范围	说明
dialchart.radius	类型:Number 默认值:0.8	指针长度比例
dialchart.base.width	类型:Number 默认值:5	指针宽度
dialchart.rear.extension	类型:Number 默认值:0	指针偏移量
dialchart.top.width	类型:Number 默认值:0	指针尖宽度

follower-跟随者相关样式

样式名	赋值范围	说明
follower.column.index	类型:Number 默认值:0	列号
follower.column.span	类型:Number 默认值:1	跨列数
follower.padding	类型:Number 默认值:0	间隙
follower.padding.bottom	类型:Number 默认值:0	底部间隙
follower.padding.left	类型:Number 默认值:0	左边间隙
follower.padding.right	类型:Number 默认值:0	右边间隙
follower.padding.top	类型:Number 默认值:0	顶部间隙
follower.row.index	类型:Number 默认值:0	行号
follower.row.span	类型:Number 默认值:1	跨行数

grid-网格样式

样式名	赋值范围	说明
grid.border	类型:Number 默认值:1	边距宽度
grid.border.bottom	类型:Number 默认值:0	底部边距
grid.border.left	类型:Number 默认值:0	左边边距
grid.border.right	类型:Number	右边边距

	默认值:0	
grid.border.top	类型:Number 默认值:0	顶部边距
grid.cell.deep	类型:Number 默认值:-1	单元格深度
grid.column.count	类型:Number 默认值:1	列数
grid.deep	类型:Number 默认值:1	网格深度
grid.fill	类型:String 默认值:true	是否填充
grid.fill.color	类型:Color 默认值:#C0C0C0	填充色
grid.padding	类型:Number 默认值:1	间隙
grid.padding.bottom	类型:Number 默认值:0	底部间隙
grid.padding.left	类型:Number 默认值:0	左边间隙
grid.padding.right	类型:Number 默认值:0	右边间隙
grid.padding.top	类型:Number 默认值:0	顶部间隙
grid.row.count	类型:Number 默认值:1	行数

group-网元组相关样式

样式名	赋值范围	说明
group.cap	类型:String 默认值:butt 取值范围:butt, round, square	边框线端点样式
group.deep	类型:Number 默认值:1	网元组深度
group.fill	类型:String 默认值:true	是否填充
group.fill.color	类型:Color 默认值:#CCCCFF	填充色
group.gradient	类型:String 默认值为：none 取值范围：linear.east, linear.north, linear.northeast, linear.northwest, linear.south, linear.southeast, linear.southwest, linear.west,none, radial.center, radial.east, radial.north, radial.northeast, radial.northwest, radial.south, radial.southeast, radial.southwest, radial.west, spread.antidiagonal, spread.diagonal, spread.east, spread.horizontal, spread.north, spread.south, spread.vertical, spread.west	渐变填充
group.gradient.color	类型:Color 默认值:#FFFFFF	渐变色
group.join	类型:String 默认值:miter 取值范围:miter,round,bevel	边框线连接点样式

group.outline.color	类型:Color 默认值:#5B5B5B	边框线颜色
group.outline.width	类型:Number 默认值:-1	边框线宽度
group.padding	类型:Number 默认值:5	间隙
group.padding.bottom	类型:Number 默认值:0	底部间隙
group.padding.left	类型:Number 默认值:0	左边间隙
group.padding.right	类型:Number 默认值:0	右边间隙
group.padding.top	类型:Number 默认值:0	顶部间隙
group.shape	类型:String 取值范围：rectangle, oval, roundrect, star, triangle, circle, hexagon, pentagon, diamond	分组形状

icons-图标样式

样式名	赋值范围	说明
icons.orientation	类型:String 默认值:right 取值范围:left, right, top, bottom	图标排列方向
icons.position	类型:String 默认值为：topleft.bottomright 取值范围：topleft.topleft, topleft.topright, top.top, topright.topleft, topright.topright, topleft, top, topright, topleft.bottomleft, topleft.bottomright, top.bottom, topright.bottomleft, topright.bottomright, left.left, left, left.right, center, right.left, right, right.right, bottomleft.topleft, bottomleft.topright,bottom.top, bottomright.topleft, bottomright.topright, bottomleft, bottom, bottomright, bottomleft.bottomleft, bottomleft.bottomright, bottom.bottom, bottomright.bottomleft, bottomright.bottomright	图标位置
icons.xgap	类型:Number 默认值:1	x间隙
icons.xoffset	类型:Number 默认值:0	x偏移量
icons.ygap	类型:Number 默认值:1	y间隙
icons.yoffset	类型:Number 默认值:0	y偏移量

image-图片样式

样式名	赋值范围	说明
image.padding	类型:Number 默认值:0	间隙
image.padding.bottom	类型:Number 默认值:0	底部间隙
image.padding.left	类型:Number 默认值:0	左边间隙
image.padding.right	类型:Number 默认值:0	右边间隙

image.padding.top	类型:Number 默认值:0	顶部间隙

label-标签样式

样式名	赋值范围	说明
label.alpha	类型:Number 默认值:1	透明度
label.cap	类型:String 默认值:butt 取值范围:butt, round, square	边框线端点样式
label.color	类型:Color 默认值:#000000	文本颜色
label.corner.radius	类型:Number 默认值:0	边框圆角半径
label.direction	类型:String 默认值:below 取值范围:aboveleft, aboveright, belowleft, belowright, leftabove, leftbelow, rightabove, rightbelow, above, below, left, right	标签方位，必须设置指针长度时才有效，见: label.pointer.length
label.font	类型:String 默认值:null	'30px arial'等
label.fill	类型:String 默认值:false	背景填充
label.fill.color	类型:Color 默认值:#C0C0C0	背景填充色
label.gradient	类型:String 默认值:none 取值范围:linear.east, linear.north, linear.northeast, linear.northwest, linear.south, linear.southeast, linear.southwest, linear.west, none, radial.center, radial.east, radial.north, radial.northeast, radial.northwest, radial.south, radial.southeast, radial.southwest, radial.west, spread.antidiagonal, spread.diagonal, spread.east, spread.horizontal, spread.north, spread.south, spread.vertical, spread.west	背景渐变
label.gradient.color	类型:Color 默认值:#FFFFFF	背景渐变色
label.join	类型:String 默认值:miter 取值范围:miter, round, bevel	边框线拐点连接样式
label.outline.color	类型:Color 默认值:#000000	边框线颜色
label.outline.width	类型:Number 默认值：-1	边框线宽度
label.padding	类型:Number 默认值:0	间隙
label.padding.bottom	类型:Number 默认值:0	底部间隙
label.padding.left	类型:Number 默认值:0	左边间隙
label.padding.right	类型:Number 默认值:0	右边间隙
label.padding.top	类型:Number	顶部间隙

	默认值:0	
label.pointer.length	类型:Number 默认值:0	指针长度
label.pointer.width	类型:Number 默认值:8	指针宽度
label.position	类型:String 默认值:bottom.bottom 取值范围:topleft.topleft, topleft.topright, top.top, topright.topleft, topright.topright, topleft, top, topright, topleft.bottomleft, topleft.bottomright, top.bottom, topright.bottomleft, topright.bottomright, left.left, left, left.right, center, right.left, right, right.right, bottomleft.topleft, bottomleft.topright, bottom.top, bottomright.topleft, bottomright.topright, bottomleft, bottom, bottomright, bottomleft.bottomleft, bottomleft.bottomright, bottom.bottom, bottomright.bottomleft, bottomright.bottomright,from,to	位置
label.shadowable	类型:String 默认值:true	阴影
label.xoffset	类型:Number 默认值:0	x偏移量
label.yoffset	类型:Number 默认值:2	y偏移量

link-连线样式

样式名	赋值范围	说明
link.bundle.enable	类型:String 默认值:true	是否参与捆绑
link.bundle.expanded	类型:String 默认值:true	展开
link.bundle.gap	类型:Number 默认值:12	捆绑间隙
link.bundle.id	类型:Number 默认值:0	捆绑编号
link.bundle.independent	类型:String 默认值:false	独立捆绑
link.bundle.offset	类型:Number 默认值:20	捆绑偏移量
link.cap	类型:String 默认值:butt 取值范围:butt,round,square	连线端点样式
link.color	类型:Color 默认值:#658DC1	颜色
link.corner	类型:String 默认值:round 取值范围:none,round,bevel	圆角类型
link.extend	类型:Number 默认值:20	延伸量
link.form.at.edge	类型:String 默认值:true	连接到起始节点的边缘
	类型:String 默认值:center 取值范围:topleft.topleft, topleft.topright, top.top, topright.topleft, topright.topright, topleft, top, topright,	

link.from.position	topleft.bottomleft, topleft.bottomright, top.bottom, topright.bottomleft, topright.bottomright, left.left, left, left.right, center, right.left, right, right.right, bottomleft.topleft, bottomleft.topright, bottom.top, bottomright.topleft, bottomright.topright, bottomleft, bottom, bottomright, bottomleft.bottomleft, bottomleft.bottomright, bottom.bottom, bottomright.bottomleft, bottomright.bottomright	起始连接点位置
link.from.xoffset	类型:Number 默认值:0	其实连接点x偏移量
link.from.yoffset	类型:Number 默认值:0	其实连接点y偏移量
link.handler.alpha	类型:Number 默认值:1	连线手柄透明度
Link.handler.cap	类型:String 默认值:butt 取值范围:butt, round, square	连线手柄边框线端点样式
link.handler.color	类型:Color 默认值:#000000	连线手柄颜色
link.handler.corner.radius	类型:Number 默认值:0	连线手柄圆角半径
link.handler.direction	类型:String 默认值:below 取值范围:aboveleft, aboveright, belowleft, belowright, leftabove, leftbelow, rightabove, rightbelow, above, below, left, right	连线手柄方向
link.handler.fill	类型:String 默认值:false	连线手柄是否填充
link.handler.fill.color	类型:Color 默认值:#C0C0C0	连线手柄填充色
link.handler.gradient	类型:String 默认值:none 取值范围:linear.east, linear.north, linear.northeast, linear.northwest, linear.south, linear.southeast, linear.southwest, linear.west, none, radial.center, radial.east, radial.north, radial.northeast, radial.northwest, radial.south, radial.southeast, radial.southwest, radial.west, spread.antidiagonal, spread.diagonal, spread.east, spread.horizontal, spread.north, spread.south, spread.vertical, spread.west	连线手柄填充渐变
link.handler.gradient.color	类型:Color 默认值:#FFFFFF	连线手柄渐变颜色
link.handler.join	类型:String 默认值:miter 取值范围:miter, round, bevel	连线手柄边框线拐点样式
link.handler.outline.color	类型:Color 默认值:#000000	连线手柄边框线颜色
link.handler.outline.width	类型:Number 默认值:-1	连线手柄边框宽度
link.handler.padding	类型:Number 默认值:0	连线手柄间隙
link.handler.padding.bottom	类型:Number 默认值:0	连线手柄底部间隙
link.handler.padding.left	类型:Number 默认值:0	连线手柄左边间隙
link.handler.padding.right	类型:Number 默认值:0	连线手柄右边间隙
link.handler.padding.top	类型:Number	连线手柄顶部间隙

	默认值:0	
link.handler.pointer.length	类型:Number 默认值:0	连线手柄指针长度
link.handler.pointer.width	类型:Number 默认值:8	连线手柄指针宽度
link.handler.position	类型:String 默认值:topleft.topleft 取值范围:topleft.topleft, topleft.topright, top.top, topright.topleft, topright.topright, topleft, top, topright, topleft.bottomleft, topleft.bottomright, top.bottom, topright.bottomleft, topright.bottomright, left.left, left, left.right, center, right.left, right, right.right, bottomleft.topleft, bottomleft.topright, bottom.top, bottomright.topleft, bottomright.topright, bottomleft, bottom, bottomright, bottomleft.bottomleft, bottomleft.bottomright, bottom.bottom, bottomright.bottomleft, bottomright.bottomright	连线手柄位置
link.handler.shadowable	类型:String 默认值:true	连线手柄是否阴影
link.handler.xoffset	类型:Number 默认值:0	连线手柄x偏移量
link.handler.yoffset	类型:Number 默认值:0	连线手柄y偏移量
link.join	类型:String 默认值:miter 取值范围:miter,round,bevel	连线拐点样式
link.looped.direction	类型:String 默认值:northwest 取值范围：north, northeast, east, southeast, south, southwest, west, northwest	自环方向
link.looped.gap	类型:Number 默认值:6	自环间隙
link.looped.type	类型:String 默认值:arc	自环类型
link.split.by.percent	类型:String 默认值:true	按百分比劈分
link.split.percent	类型:Number 默认值:0.5	劈分百分比
link.split.value	类型:String 默认值:20	劈分量
link.to.position	类型:String 默认值:center 取值范围:topleft.topleft, topleft.topright, top.top, topright.topleft, topright.topright, topleft, top, topright, topleft.bottomleft, topleft.bottomright, top.bottom, topright.bottomleft, topright.bottomright, left.left, left, left.right, center, right.left, right, right.right, bottomleft.topleft, bottomleft.topright, bottom.top, bottomright.topleft, bottomright.topright, bottomleft, bottom, bottomright, bottomleft.bottomleft, bottomleft.bottomright, bottom.bottom, bottomright.bottomleft, bottomright.bottomright	结束端连接点位置
link.to.xoffset	类型:Number 默认值:0	结束端连接点x偏移量
link.to.yoffset	类型:Number 默认值:0	结束端连接点y偏移量
link.type	类型:String 默认值:arc 取值范围：arc, triangle, parallel, flexional, flexional.horizontal, flexional.vertical, orthogonal, orthogonal.horizontal, orthogonal.vertical, orthogonal.H.V, orthogonal.V.H, extend.top, extend.left, extend.bottom, extend.right	连线类型

link.width	类型:Number 默认值:3	连线宽度
link.xradius	类型:Number 默认值:8	x圆角半径
link.yradius	类型:Number 默认值:8	y圆角半径

outer-边框

样式名	赋值范围	说明
outer.cap	类型:String 默认值:butt 取值范围:butt,round,square	边框线端点样式
outer.join	类型:String 默认值:miter 取值范围:miter, round, bevel	边框线拐点样式
outer.padding	类型:Number 默认值:1	边框间隙
outer.padding.bottom	类型:Number 默认值:0	边框底部间隙
outer.padding.left	类型:Number 默认值:0	边框左边间隙
outer.padding.right	类型:Number 默认值:0	边框右边间隙
outer.padding.top	类型:Number 默认值:0	边框顶部间隙
outer.shape	类型:String 默认值:rectangle 取值范围：rectangle, oval, roundrect, star, triangle, circle, hexagon, pentagon, diamond	边框形状
outer.width	类型:Number 默认值:2	边框线宽度

select-选中边框样式

样式名	赋值范围	说明
select.cap	类型:String 默认值:butt 取值范围:butt, round, square	边框线端点样式
select.color	类型:String 默认值:rgba(0,0,0,0.7)	颜色
select.join	类型:String 默认值:miter 取值范围:miter,round,bevel	边框线拐点样式
select.padding	类型:Number 默认值:2	间隙
select.padding.bottom	类型:Number 默认值:0	底部间隙
select.padding.left	类型:Number	左边间隙

	默认值:0	
select.padding.right	类型:Number 默认值:0	右边间隙
select.padding.top	类型:Number 默认值:0	顶部间隙
select.shape	类型:String 默认值:rectangle 取值范围:rectangle, oval, roundrect, star, triangle, circle, hexagon, pentagon, diamond	形状
select.style	类型:String 默认值:shadow 取值范围:null, shadow, border	选中边框样式
select.width	类型:Number 默认值:2	选中边框宽度

shadow-阴影样式

样式名	赋值范围	说明
shadow.blur	类型:Number 默认值:6	模糊指数
shadow.xoffset	类型:Number 默认值:3	x偏移量
shadow.yoffset	类型:Number 默认值:3	y偏移量

shapelink

样式名	赋值范围	说明
shapelink.type	类型:String 默认值:lineto 取值范围:lineto,quadto,cubicto	连线类型

shapenode-多边形样式

样式名	赋值范围	说明
shapenode.closed	类型:String 默认值:false	是否闭合

vector-图形样式

样式名	赋值范围	说明
vector.cap	类型:String 默认值:butt 取值范围:butt,round,square	边框线端点样式
vector.deep	类型:Number 默认值:0	深度
vector.fill	类型:String 默认值:true	是否填充
vector.fill.color	类型:Color 默认值:#CCCCFF	填充色
	类型:String	

vector.gradient	默认值:none 取值范围:：linear.east, linear.north, linear.northeast, linear.northwest, linear.south, linear.southeast, linear.southwest, linear.west, none, radial.center, radial.east, radial.north, radial.northeast, radial.northwest, radial.south, radial.southeast, radial.southwest, radial.west, spread.antidiagonal, spread.diagonal, spread.east, spread.horizontal, spread.north, spread.south, spread.vertical, spread.west	填充渐变
vector.gradient.color	类型:Color 默认值:#FFFFFF	渐变色
vector.join	类型:String 默认值:miter 取值范围:miter,round,bevel	边框线拐点样式
vector.outline.color	类型:Color 默认值:#5B5B5B	边框线颜色
vector.outline.width	类型:Number 默认值:-1	边框线宽度
vector.padding	类型:Number 默认值:0	间隙
vector.padding.bottom	类型:Number 默认值:0	底部间隙
vector.padding.left	类型:Number 默认值:0	左边间隙
vector.padding.right	类型:Number 默认值:0	右边间隙
vector.padding.top	类型:Number 默认值:0	顶部间隙
vector.shape	类型:String 默认值:rectangle 取值范围:rectangle, oval, roundrect, star, triangle, circle, hexagon, pentagon, diamond	图形形状

whole-网元整体

样式名	赋值范围	说明
whole.alpha	类型:Number 默认值:1	整体透明度

API使用文档

版本：TWaver HTML5 v5.0.3

在线阅读：点击在新窗口中打开

下载至本地：twaver-html5-5.0.3-api.zip