

# TWaver Flex Training

Serva Software LLC

# TWaver Flex Introduction

TWaver™

- **What is TWaver?**
- **Flex Basics**
  - Flex Language Features
  - Flex & Flash ...
- **TWaver Flex Basics**
  - Hello TWaver!
  - MVC Pattern
  - Data Items/Data Models/Views
- **Comparison with TWaver Java**

# What is TWaver?

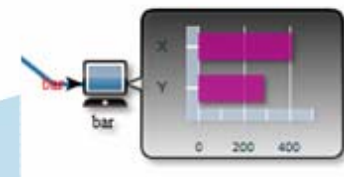
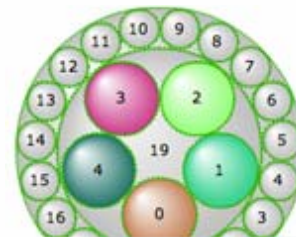
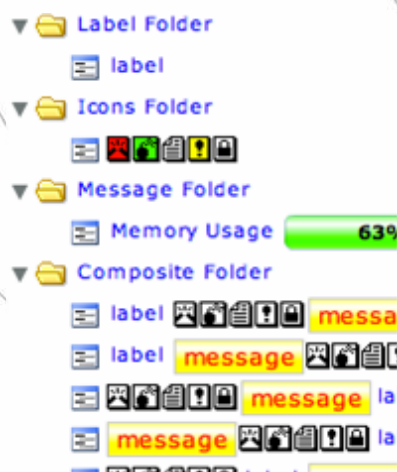
- High efficiency and lightweight graphical component library
- Multi-platform solutions
- TWaver Flex package contents
- TWaver license

TWaver™

# TWaver is Graphical Component Library

TWaver™

- TWaver focus on graphical display of data
- TWaver is for developers, needs secondary development
- TWaver provides documentation, license, training and technical supports

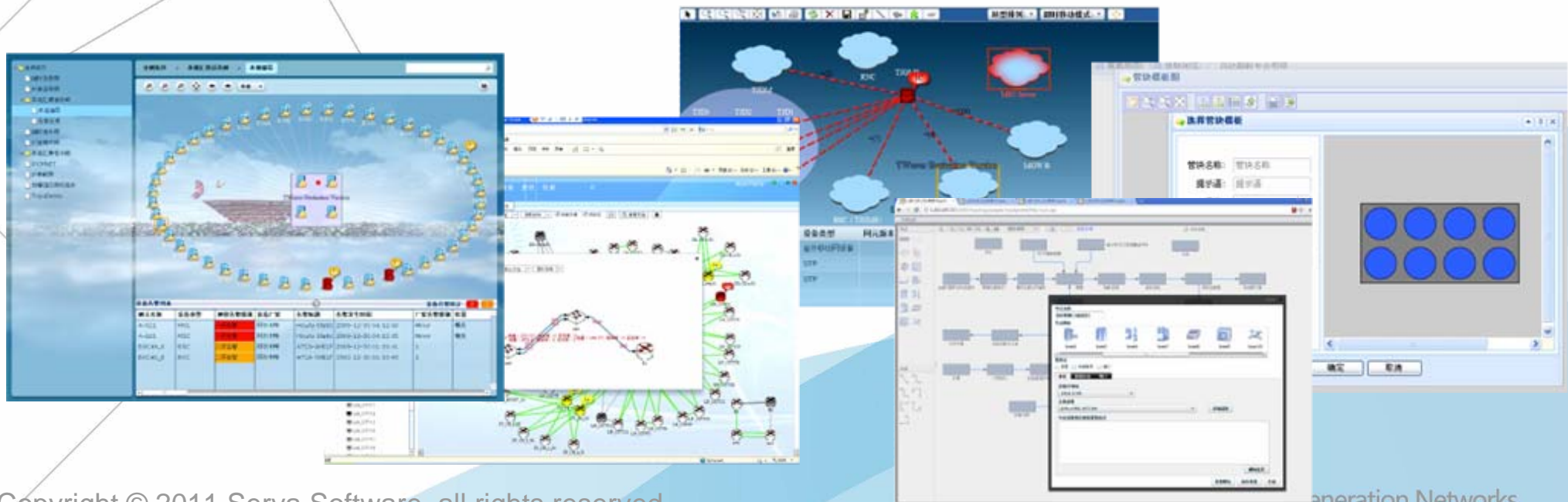


element	tree label	network label
 twaver.Node	boy	boy
 twaver.Node	iphone	iphone
 twaver.Node	Internet	Internet
 twaver.Node	computer	computer

# Focus on Telecom Network Management Components

- Provides telecommunications business models, equipment chassis, alarm transformation, etc.
- Have a large number of telecommunication application cases
- But not limited in telecommunications

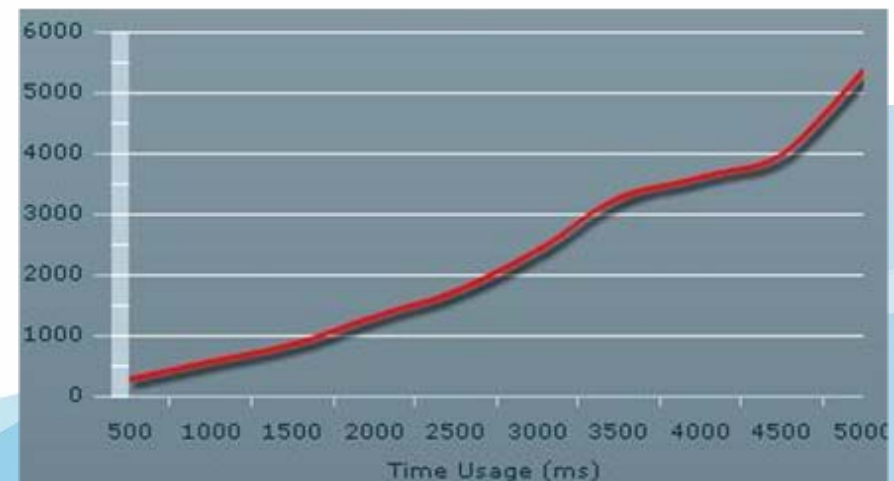
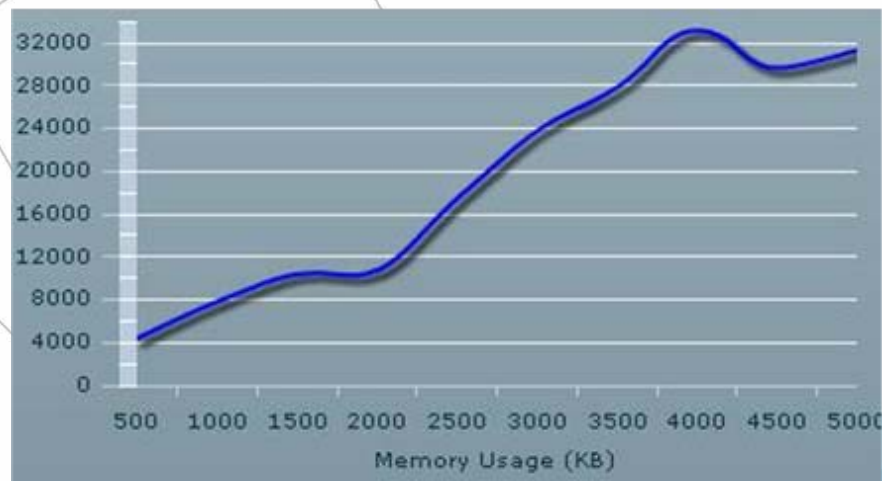
TWaver™





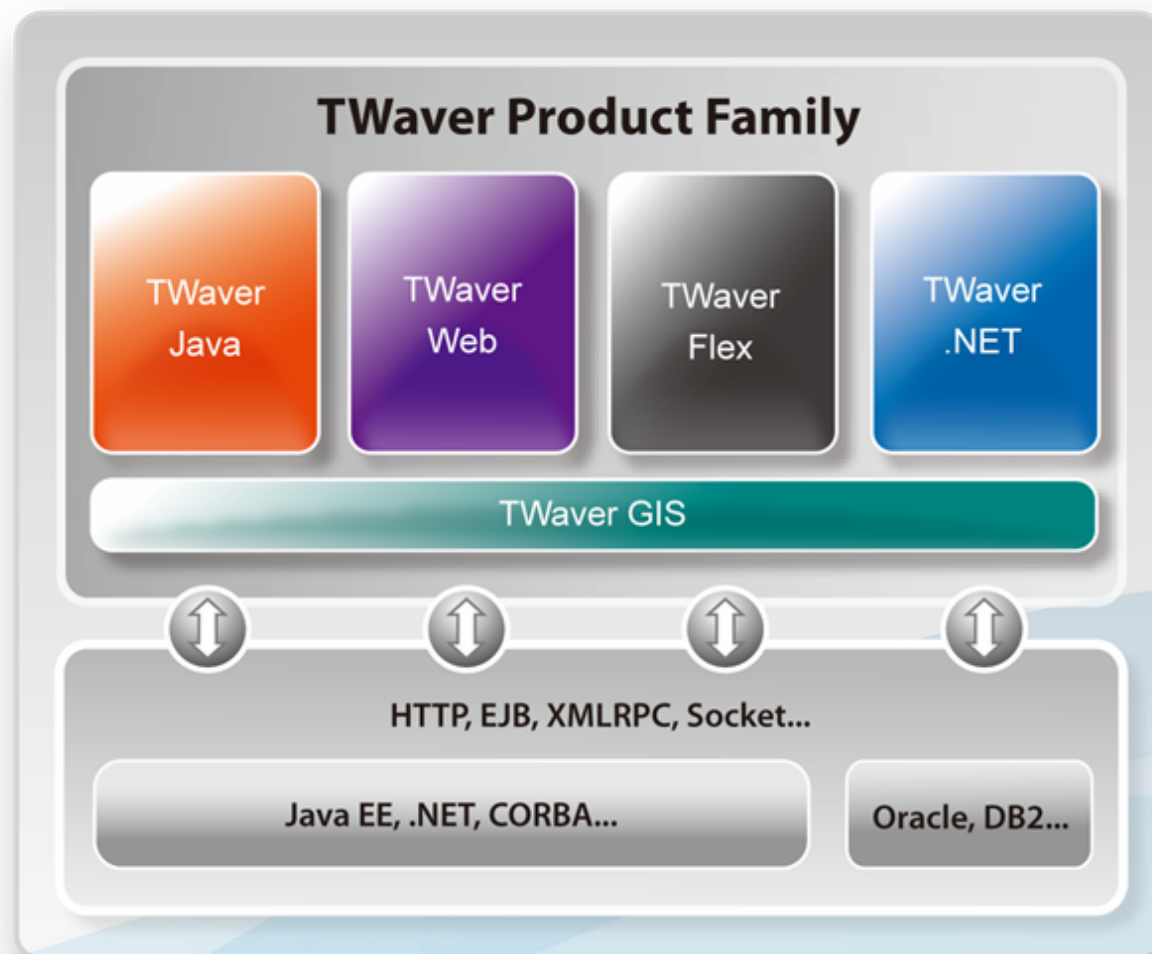
# High Efficiency and Lightweight

- TWaver.swc 475kb
- Less than 2000 nodes or links are suggested
- Load 2000 nodes within 1.5s



# Multi-Platform Solutions

- Java
- Web
- **Flex**
- .NET
- HTML5...



# Customers

TM  
waver





# TWaver Flex Package Contents

TWaver™

twaver-flex-1.4	--
asdocs	--
Blue.swf	29 KB
demo.html	4 KB
demo.swf	2.1 MB
documents	--
TWaver Flex 1.4 Developer Guide.pdf	1.7 MB
lib	--
TWaver.swc	475 KB
License.txt	16 KB
README.txt	4 KB
src	--

API Docs

demo run files

developer guider

TWaver.swc

demo source

# TWaver License

- TWaver has three types of licenses: Evaluation License, Development License, and Runtime License. Click Ctrl+Shift+T to check license information on Network interface.
- **Evaluation** - Used for prophecy on the initial stage or a period of technical selection, component interface will display watermark of "Twaver Evaluation Version" within 5 minutes
- **Development** - Grant you the right to use TWaver to develop your software product or project. The watermark will show after 2 hours.
- **Runtime** - Used for your final project/product deployment. No watermark

TWaver™

# How to Use License File

Twaver™

- license.xml is a plain-text format file, it contains license informations.
- How to use:
  - `<mx:XML source="demo/license.xml" id="licenseXML"/>`
  - `twaver.Utls.validateLicense(this.licenseXML);`

# Flex Basics

- Flex language features
  - get & set
  - .mxml & .as
- Flex & Flash
- Resource embed
- Others

Twaver™



# Flex Language Features

- ActionScript
- It's a scripting language, but use similarly to JAVA, it's an OOP language
- Function Overload
- Doesn't support function overload, one name just on function, supports default parameter values
- Scripting Language Features
- supports arguments, function, prototype, call, apply, length, object['propertyA'] .....and dynamic object

TM  
waver

# get & set

TWaver™

```
/twaver/Data.as
package twaver{
    import flash.events.EventDispatcher;
    ...
    public class Data extends EventDispatcher implements IData{
        public function Data(name:String = null){
            this.name = name;
        }
        private var _name:String;
        public function set name(name:String):void{
            this._name = name;
        }
        public function get name():String{
            return this._name;
        }
    }
}
```

Example:

```
var data:Data = new Data();
data.name = "001";
trace(data.name);
```

# .mxml & .as

- Default application must be .mxml
- .mxml compiled into ActionScript classes finally
- Both .mxml and .as can defined class
- .mxml can instead by .as
- .mxml class cannot pass constructor parameters

TM  
T  
W  
a  
v  
e  
r  
™

# Flex Using Habits

Twaver™

- var, function, “:”,quotation mark, get / set
- No Color, Double, Float, instead by int & Number
- children & toChildren
- forEach(callback)
- setStyle(name, value)
- addEventListener(type, listener, ... )
- CSS
- namespace, manifest.xml, xmlns



# Flex & Flash

- Flex - open、 free
  - Flex SDK
  - “Flex is a highly productive, free, open source framework for building expressive mobile, web, and desktop applications.”
- Flash - closed、 paid for
  - Flash Builder
  - Flash Player

TM  
Twaiver

# Resource Embed

TWaver™

- Embedded resources are packed into a SWF file
- Image embed

```
[Embed(source="resource/images/alarm.png")]  
private static const alarmIcon:Class;
```

- Font embed

```
[Embed(source='Helvetica.ttf', fontName="demoFont",  
        advancedAntiAliasing="false",    mimeType="application/x-font")]  
private var demoFont:Class;
```

- XML embed

```
<mx:XML source="demo/states.xml" id="statesXML"/>
```

- CSS embed

```
<mx:Style source="Default.css" />
```

# Others

- Graphics, Filter, BitmapData ...
- .swc / .swf / .air
- Security sandboxes, local domain, network domain, security domain, crossdomain.xml
- ImageLoader, URLLoader, ModuleLoader
- HTTPService
- Debug, publish

TM  
waver

# TWaver Basics

TWaver™

- Hello TWaver
- MVC design pattern
- Data items and data models
- Views



# Hello TWaver

TWaver™

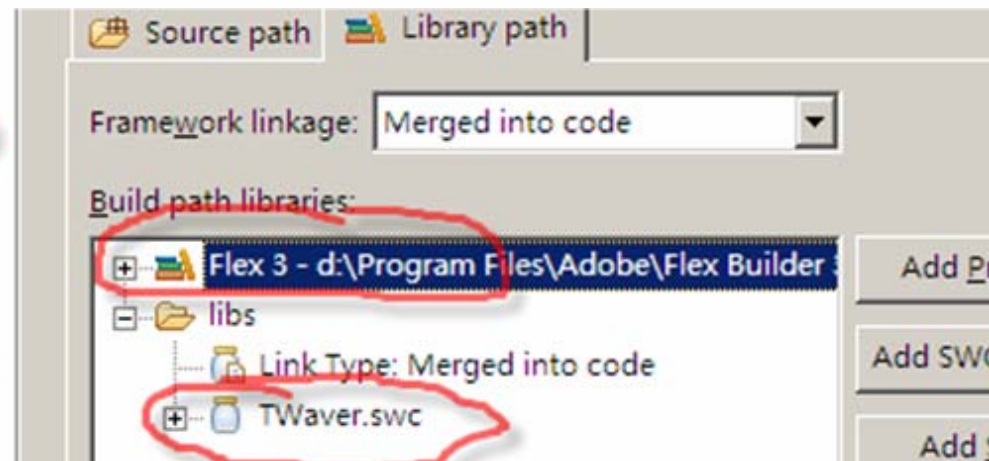
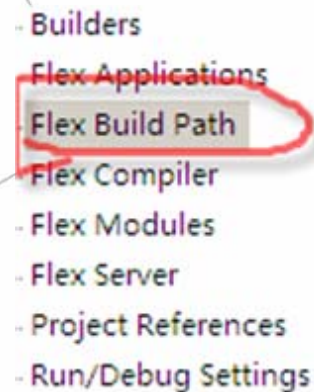
## TWaver Flex development environment

- TWaver.swc
- Flex SDK 3.4.2+
- Flash Builder .....
- Flash Player 9+

# Hello TWaver

- Create Flex Project
- Import TWaver.swc
- Add Flex SDK (3.4.2+)

Refer to section "TWaver Flex development environment" from developer guide



TWaver™

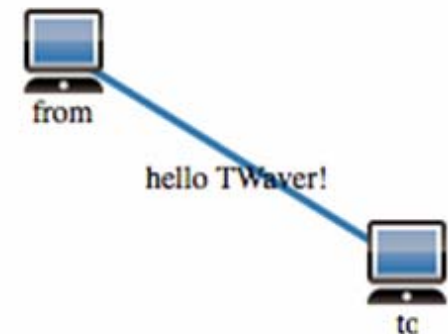
# Hello TWaver

TWaver™

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" pageTitle='Hello TWaver !'
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;
      private function init():void{
        var box:ElementBox = network.elementBox;
        var from:Node = new Node();
        from.name = "from";
        from.location = new Point(20, 20);
        box.add(from);
        var to:Node = new Node();
        to.name = "to";
        to.location = new Point(150, 60);
        box.add(to);
        var link:Link = new Link(from,to);
        link.name = "hello TWaver!";
        box.add(link);
      }
    ]]>
  </mx:Script>
  <twaver:Network id="network" backgroundColor="0xffffff" width="100%" height="100%"/>
</mx:Application>

```



# Adding Tree, Table

TWaver™

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex"
  pageTitle="Hello TWaver!"
  applicationComplete="init()">
  <mx:Script>
    <![CDATA[
      import twaver.*;
      import twaver.network.Network;

      private var box:ElementBox;

      private function init():void{
        box=network.elementBox;
        var from:Node = new Node();
        from.name = "from";
        from.location = new Point(20, 20);
        box.add(from);
        var to:Node = new Node();
        to.name = "to";
        to.location = new Point(150, 100);
        box.add(to);
        var link:Link = new Link(from,to);
        link.name = "hello TWaver!";
        box.add(link);
      }
    ]]>
  </mx:Script>
```

```
tree.dataBox=box;
table.dataBox=box;
```

Set data box to tree & table

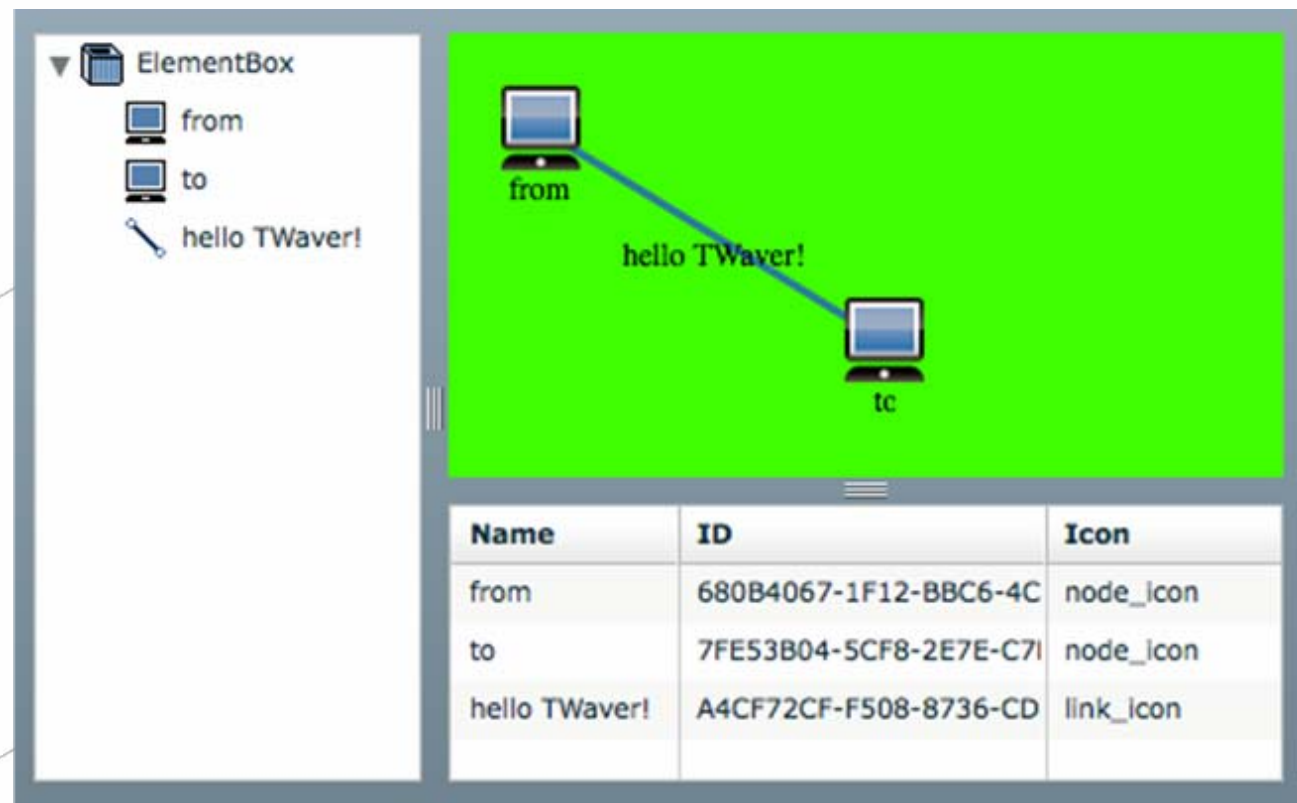
Configure table in MXML

```
<mx:HDividedBox width="100%" height="100%">
  <twaver:Tree id="tree" width="30%" height="100%" />
  <mx:VDividedBox width="100%" height="100%">
    <twaver:Network id="network" backgroundColor="0x00ff00" width="100%" height="70%" />
    <twaver:Table width="100%" height="30%" id="table" editable="true">
      <twaver:columns>
        <twaver:TableColumn dataField="name" headerText="Name" />
        <twaver:TableColumn dataField="id" headerText="ID" />
        <twaver:TableColumn dataField="icon" headerText="Icon" />
      </twaver:columns>
    </twaver:Table>
  </mx:VDividedBox>
</mx:HDividedBox>
</mx:Application>
```



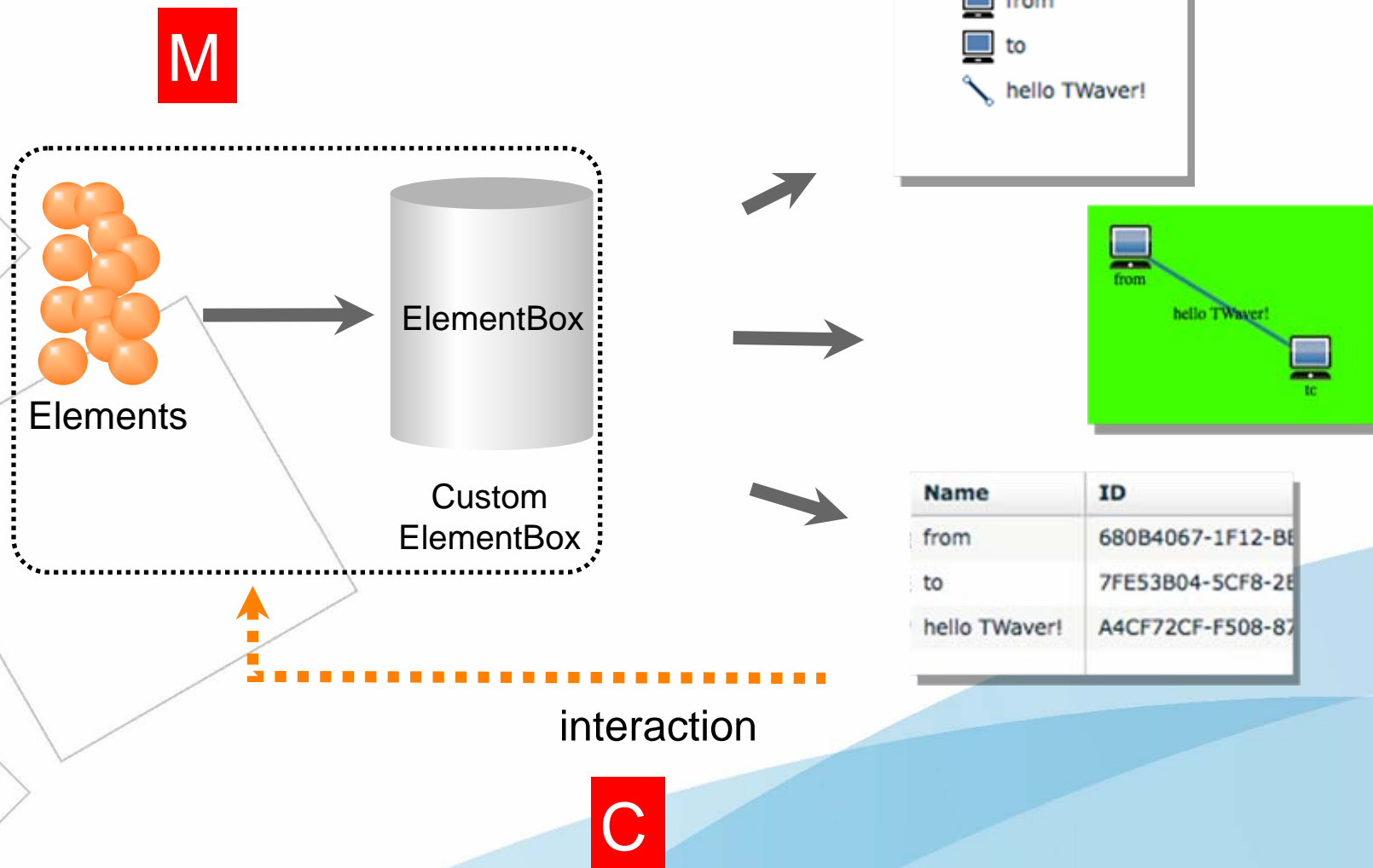
# Hello TWaver

TWaver™



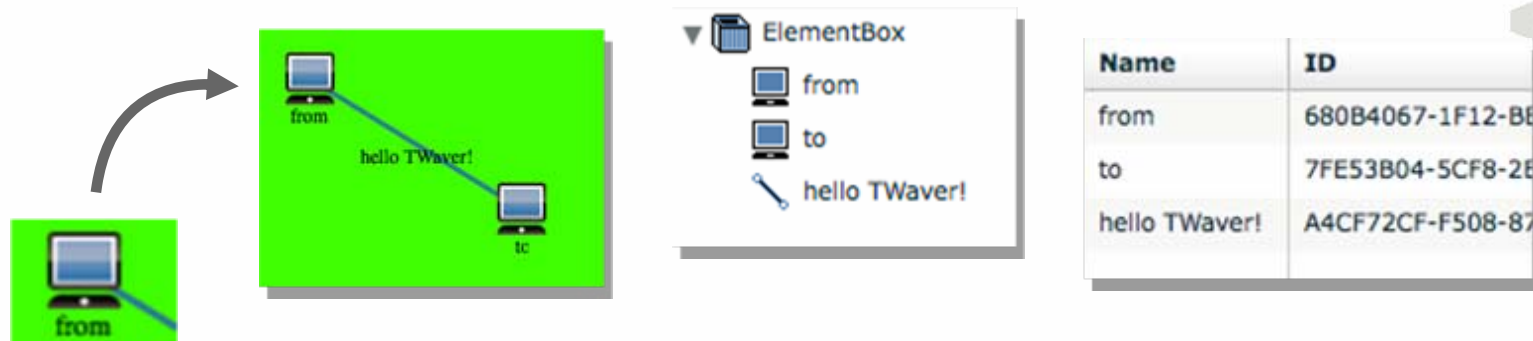
# MVC Design Pattern

TWaver™



# Model & Views

Views(V)



ElementUI

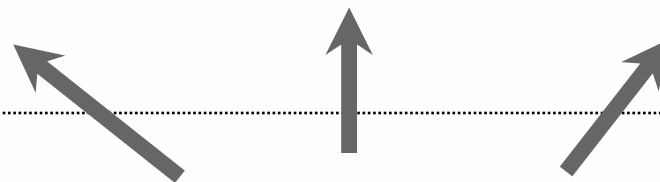
Element

Model(M)

Node  
Link  
Group  
...



ElementBo  
X



TWaver™

# Event-Driven

Twaver™

**Element** `node.setStyle(Styles.INNER_COLOR, 0xFF0000)`

`dispatchPropertyChangeEvent`

**ElementBox**

`dataPropertyChangeListener`

`dispatchEvent`

**View**

network, tree, chart ...

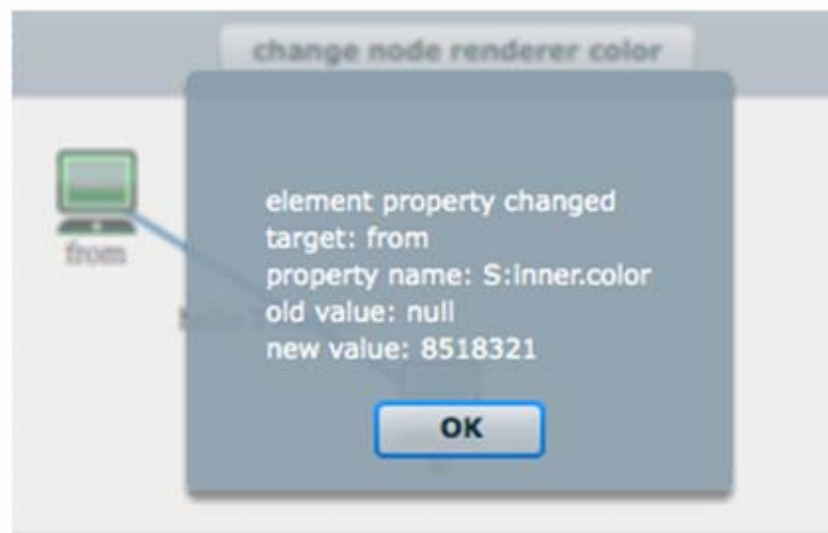
`dataPropertyChangeListener`

`invalidate(element)`



# Event Listeners

```
box.addDataPropertyChangeListener(function(evt:PropertyChangeEvent):void{  
    Alert.show("element property changed\ttarget: " + evt.source +  
        "\nproperty name: " + evt.property +  
        "\nold value: " + evt.oldValue +  
        "\nnew value: " + evt.newValue);  
});
```

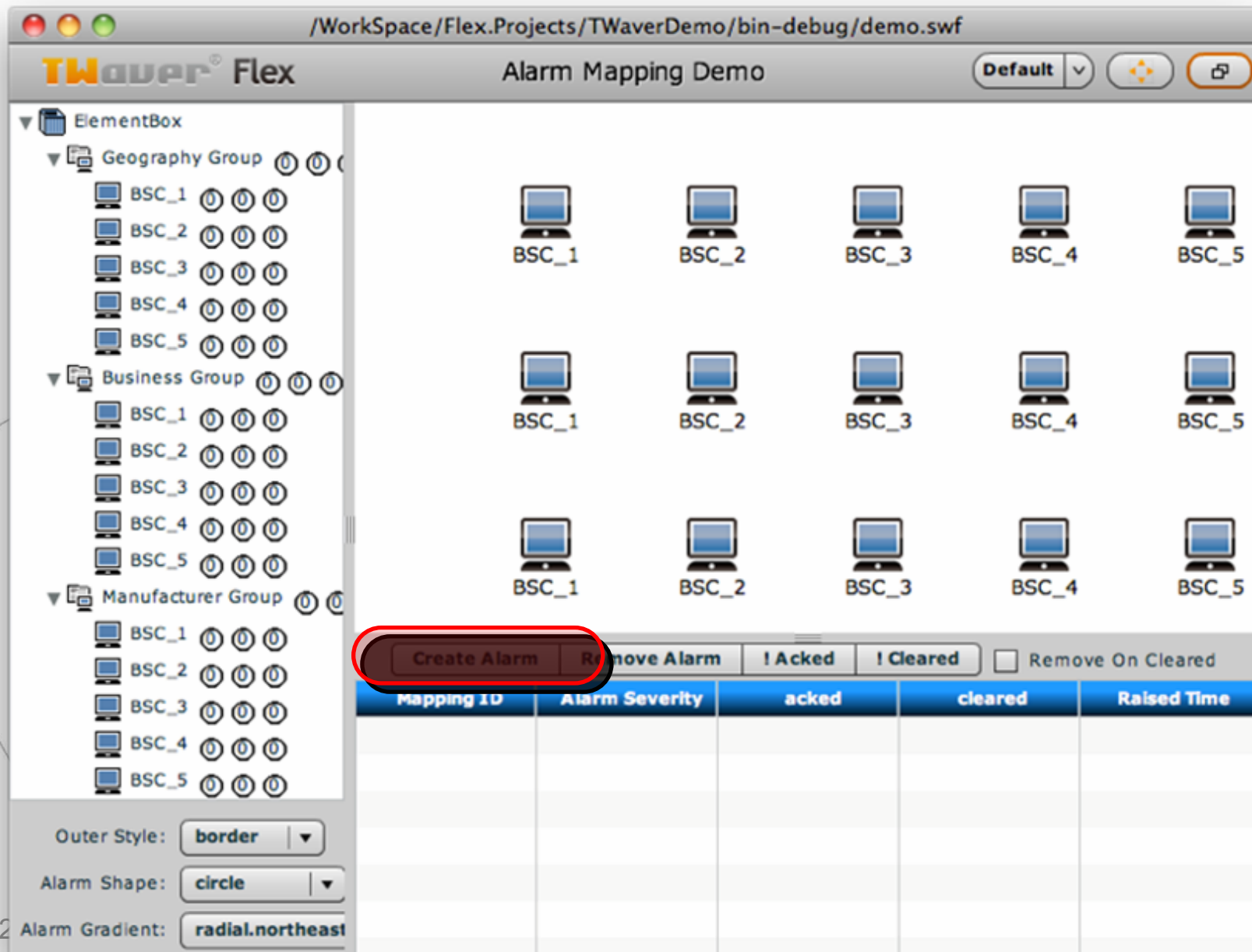


# Event Listeners

Target	Listener
DataBox	addDataBoxChangeListener addDataPropertyChangeListener addHierarchyChangeListener addPropertyChangeListener
ElementBox	extends DataBox addIndexChangeListener
AlarmBox	extends DataBox
LayerBox	extends DataBox
SelectionModel	addSelectionChangeListener
Network	addInteractionListener addPropertyChangeListener
AlarmSeverity	addAlarmSeverityChangeListener



# Views Synchronize Update



Alarm Mapping Demo

Default

ElementBox

- Geography Group
  - BSC\_1
  - BSC\_2
  - BSC\_3
  - BSC\_4
  - BSC\_5
- Business Group
  - BSC\_1
  - BSC\_2
  - BSC\_3
  - BSC\_4
  - BSC\_5
- Manufacturer Group
  - BSC\_1
  - BSC\_2
  - BSC\_3
  - BSC\_4
  - BSC\_5

Create Alarm Remove Alarm Acked Cleared Remove On Cleared

Mapping ID	Alarm Severity	acked	cleared	Raised Time

Outer Style: border

Alarm Shape: circle

Alarm Gradient: radial.northeast

TWaver™

# Views Synchronize Update

The screenshot shows the TWaver Flex Alarm Mapping Demo application. The interface includes a tree view on the left, a grid view in the center, and a table at the bottom.

**Tree View:**

- Geography Group
  - BSC\_1
  - BSC\_2
  - BSC\_3
  - BSC\_4 (highlighted with a red circle)
- Business Group
  - BSC\_1
  - BSC\_2
  - BSC\_3
  - BSC\_4
  - BSC\_5
- Manufacturer Group
  - BSC\_1
  - BSC\_2
  - BSC\_3
  - BSC\_4
  - BSC\_5

**Grid View:**

The grid displays BSC\_1 through BSC\_5 for each group. BSC\_4 in the Geography Group is highlighted with a red circle and a '1M' label. BSC\_4 in the Business Group is also highlighted with a red circle and a '1M' label.

**Table:**

Mapping ID	Alarm Severity	acked	cleared	Raised Time
4	Major	<input type="checkbox"/>	<input type="checkbox"/>	Thu Jun 9 16:51

**Buttons:**

- Create Alarm
- Remove Alarm
- ! Acked
- ! Cleared
- ☐ Remove On Cleared

**Settings:**

- Outer Style: border
- Alarm Shape: circle
- Alarm Gradient: radial.northeast

# Data & Data Models

**Data Item** : The basic unit of data

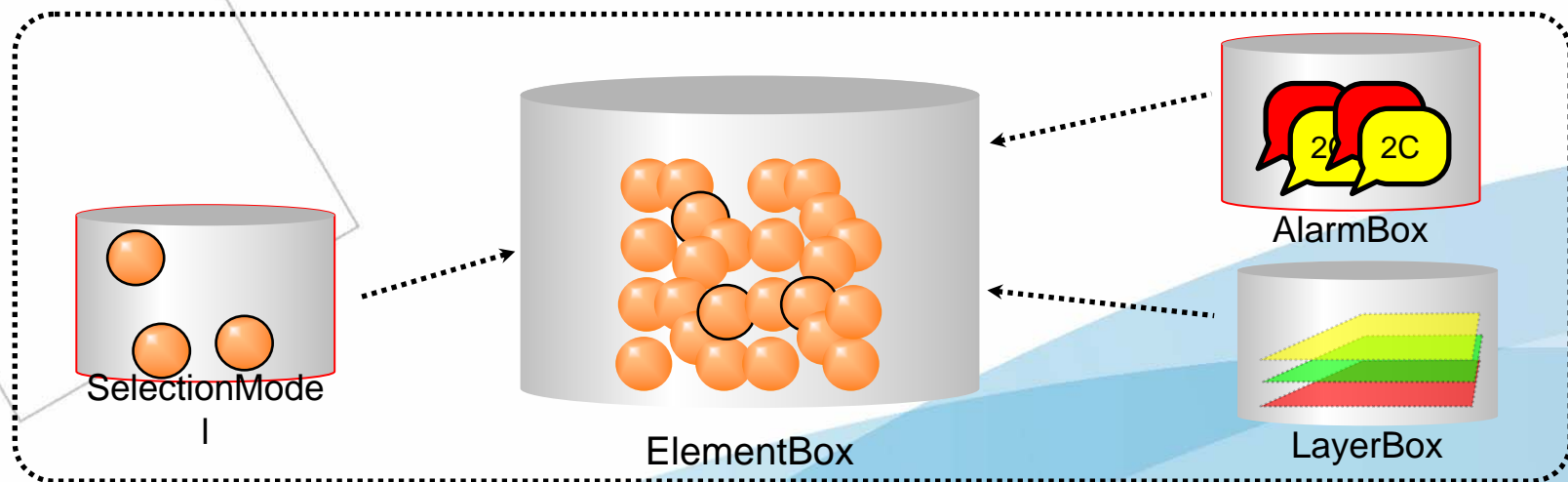
**Data Model** : the data management container, provides add/remove/clear operations, can monitor the property change of data

Data Items	Data Models
IData	DataBox
IElement	ElementBox
IArm	AlarmBox
ILayer	LayerBox

# The Relationship Among Data Models

TWaver™

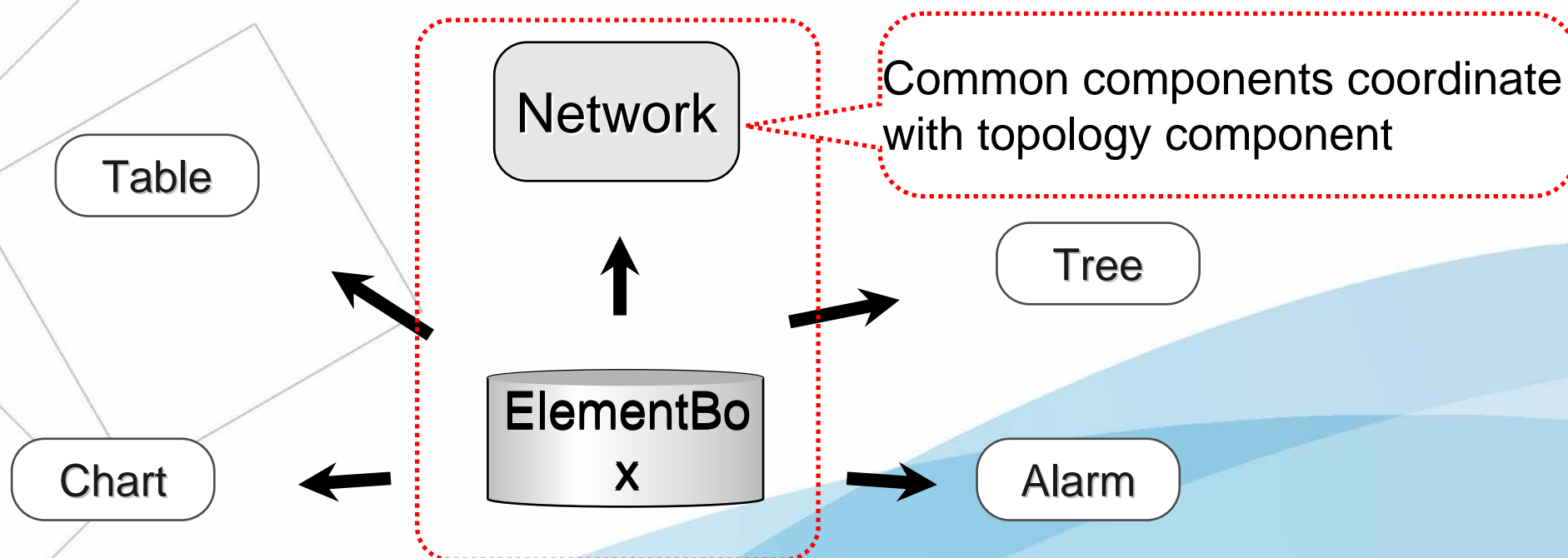
- ElementBox is a data container for managing network elements, it includes a alarm box, a layer box and a selection model



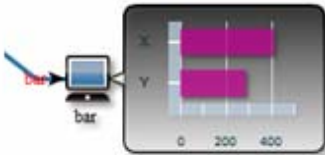
# View Types

TWaver™

- **Topology component - Network**
- **Common components - Tree, Table, Charts**

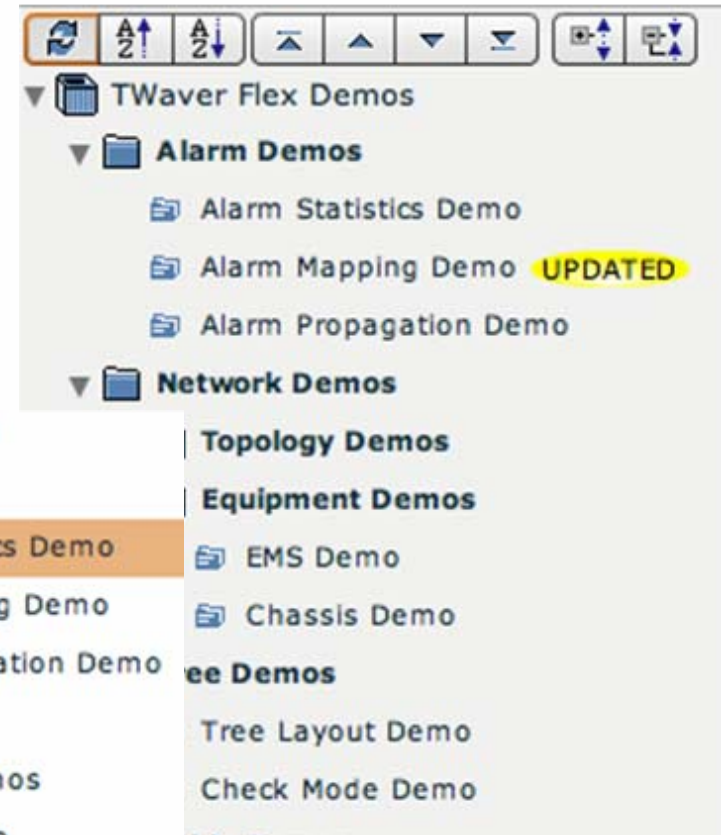
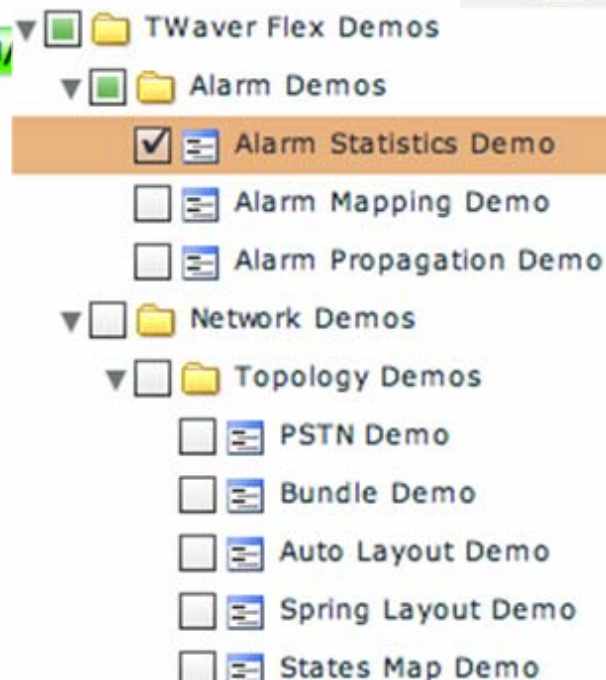








# Tree













TWaver™

# Table

Twaver™

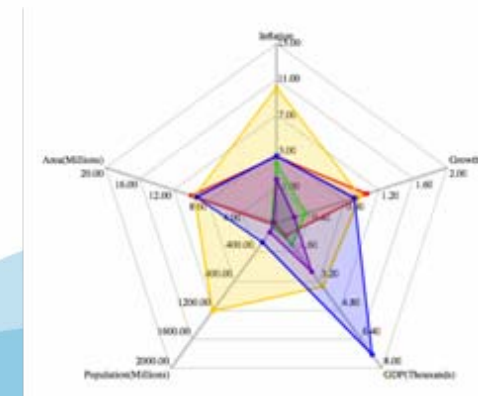
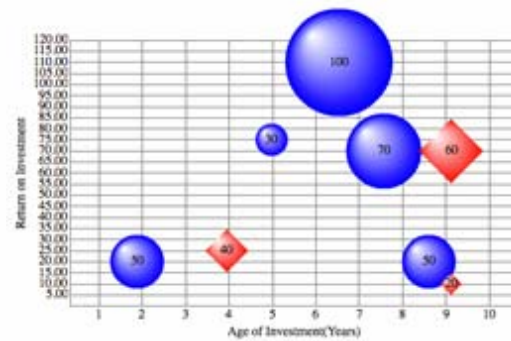
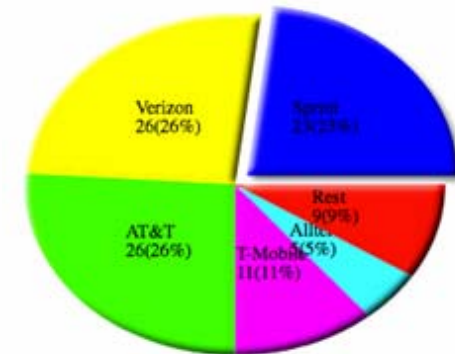
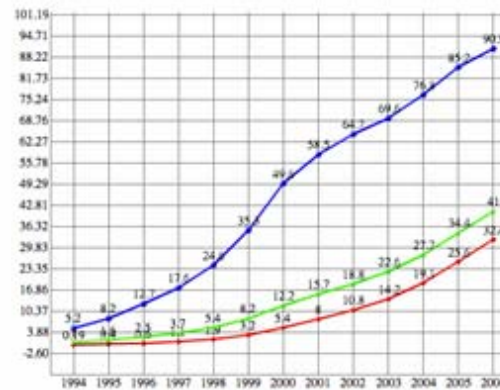
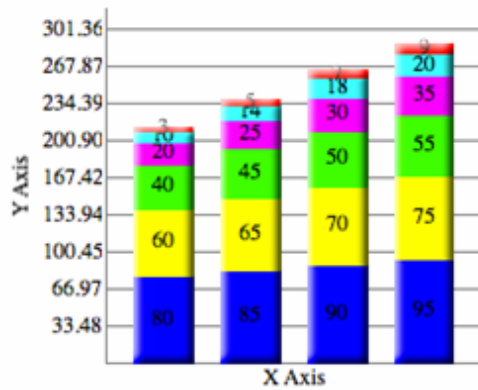
state	employed	unemploy	male	female
District of Columbia	303994	23442	282970	323930
Mississippi	1028773	94712	1230617	1342599
Alabama	1741794	128587	1936162	2104425
New York	8498119	636280	8739138	9496769
Pennsylvania	5434532	344795	5694265	6187378

Mapping ID	Alarm Severity	acked	cleared	Raised Time
1	Indeterminate	<input type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
2	Warning	<input type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
3	Minor	<input type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
4		<input type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
5	Cleared	<input type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
5	Indeterminate	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
4	Warning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
3	Minor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
2	Minor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011
1	Critical	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 10 14:25:51 GMT+0800 2011

element	tree label	network label
 twaver.Node	boy	boy
 twaver.Node	iphone	iphone
 twaver.Node	internet	internet
 twaver.Node	computer	computer
 twaver.Node	girl	girl
 twaver.Link	boy --> iphone	
 twaver.Link	boy -- iphone	
 twaver.Link	boy <-- iphone	
 twaver.Link	iphone --> internet	
 twaver.Link	iphone -- internet	

# Chart

TM  
waver





# Comparison with TWaver Java

TWaver™

- There are many similarities between TWaver Flex and TWaver Java
- Class comparison
- Element properties comparison
- Network component comparison

# Classes Comparison

TWaver™

TWaver Flex	TWaver Java	Description
ElementBox	TDataBox	Element container
AlarmBox	AlarmModel	Alarm container
LayerBox	LayerModel	Layer container
Network	TNetwork	Topology component
Tree	TTree	Tree component
Table	TElementTable	Table component

# Element Properties Comparison

TWaver™

TWaver Flex	TWaver Java	Description
<code>node.name = "001"</code>	<code>node.setName("001")</code>	properties
<code>node.setStyle</code>	<code>node.putClientProperty</code>	styles
<code>node.setClient</code>	<code>node.putUserProperty</code>	client properties



# Network Using Comparison

TWaver Flex	TWaver Java	Description
alarmLabelFunction	alarmLabelGenerator	alarm bubble label
visibleFunction	visibleFilter	visible filter
movableFunction	movableFilter	movable filter
editableFunction	elementLabelEditableFilter	label editable filter
labelFunction	elementLabelGenerator	label text generator
toolTipFunction	elementToolTipTextGenerator	tooltip text generator
innerColorFunction	elementBodyColorGenerator	body color generator
outerColorFunction	elementOutlineColorGenerator	outline color generator
selectColorFunction	elementSelectColorGenerator	selection color generator
alarmFillColorFunction	alarmColorGenerator	alarm bubble color



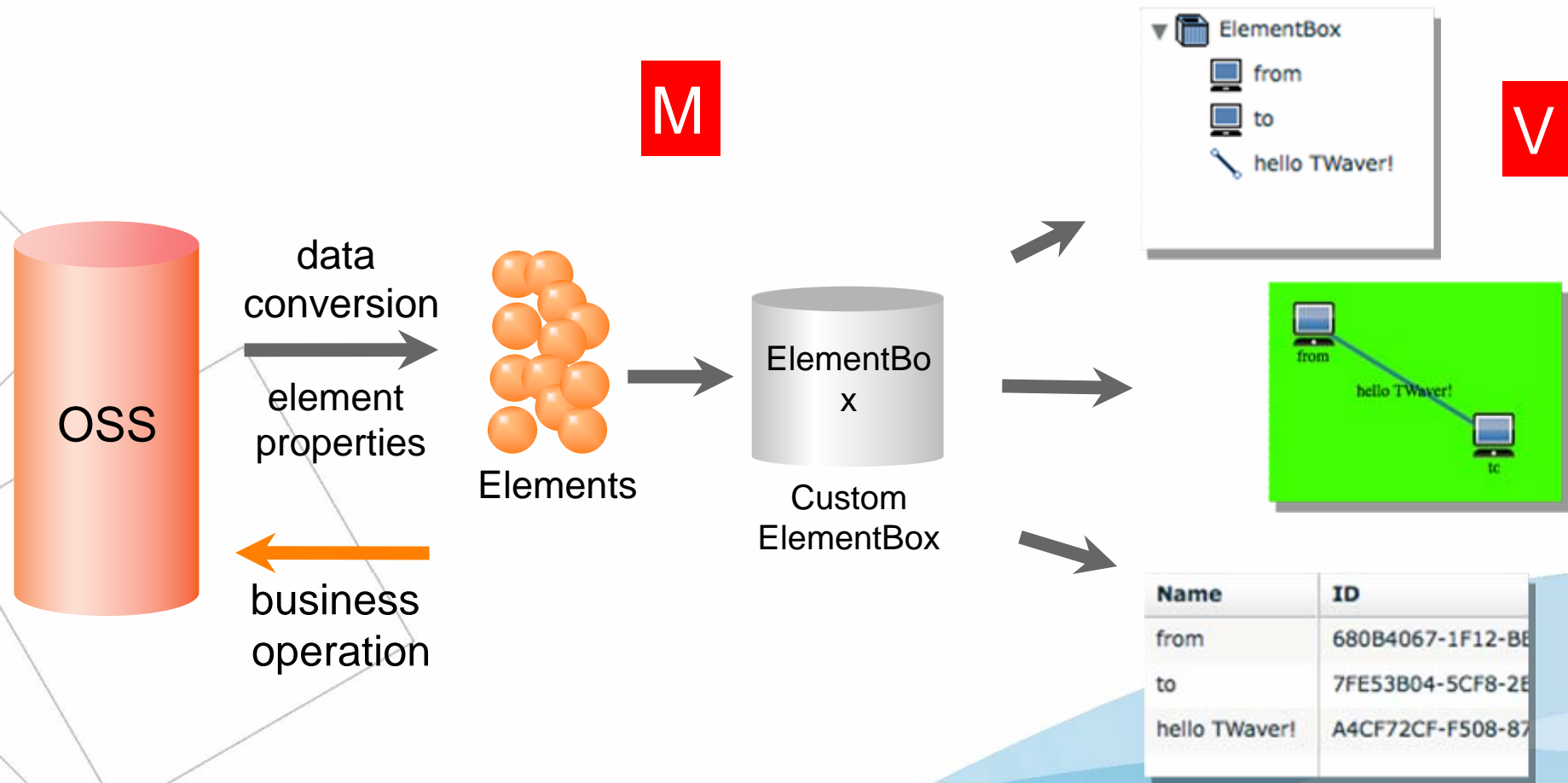
- Home - ServaSoftware.com
- Email - [tw-service@servasoft.com](mailto:tw-service@servasoft.com)

# TWaver Flex Core Components

TWaver™

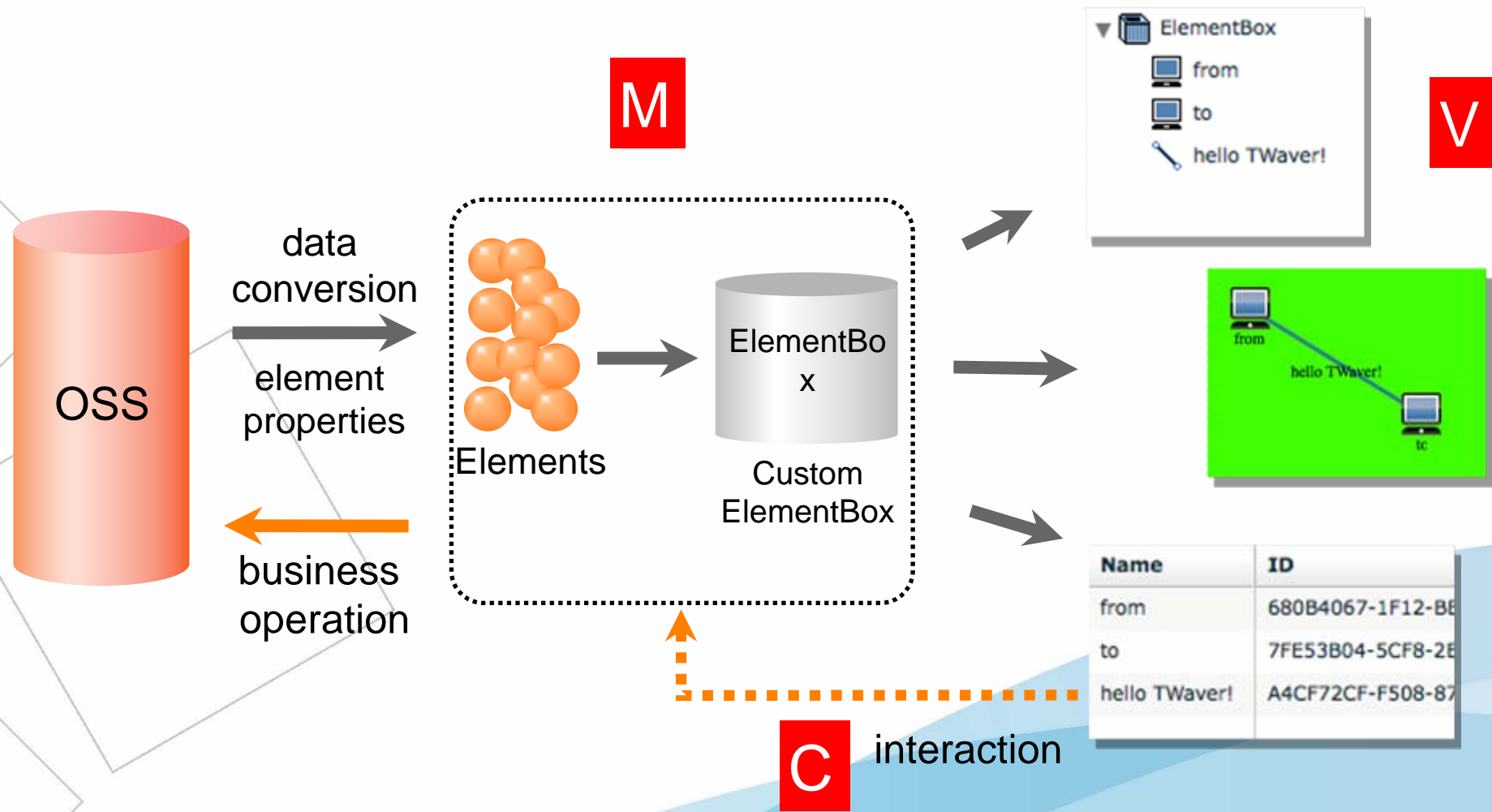
- Development process
- Using DataBox & ElementBox
- Predefined elements
- Using Network

# TWaver Flex Development Process



TWaver™

# TWaver Flex Development Process



# TWaver Flex Development Process

TWaver™

```
box = network.elementBox;
```

```
//data acquisition
```

```
var devices:Array = getDevicesFromOSS();
```

```
var relationships:Array = getDevicesRelationshipFromOSS();
```

```
//data conversion
```

```
translateToTWaverNode(box, devices, relationships);
```

```
var layouter:AutoLayouter = new AutoLayouter(network);
```

```
layouter.animate = false;
```

```
layouter.doLayout(Consts.LAYOUT_SYMMETRY);
```

```
//add interaction
```

```
network.addListener(function(evt:InteractionEvent):void{
```

```
    if(evt.kind != InteractionEvent.DOUBLE_CLICK_ELEMENT){
```

```
        return;
```

```
    }
```

```
    var element:IElement = evt.element;
```

```
    InputDialog.show(network, "Rename", element.name, function(text:String):void{
```

```
        element.name = text;
```

```
        //to storage
```

```
    }, evt.mouseEvent.stageX, evt.mouseEvent.stageY);
```

```
});
```



# TWaver Flex Development Process

TWaver™

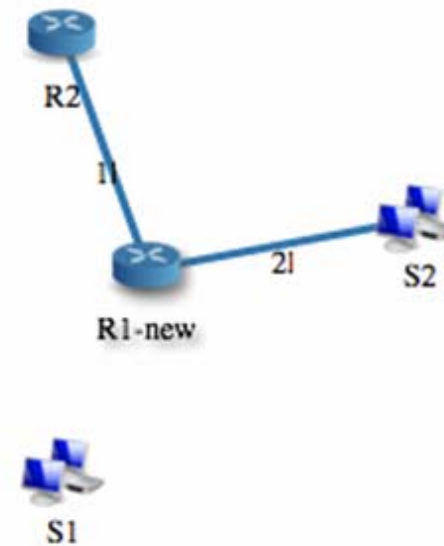
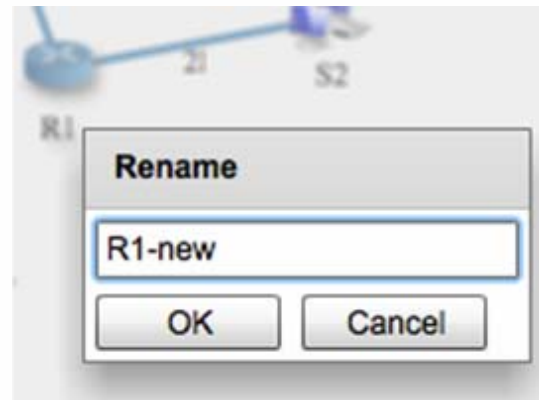
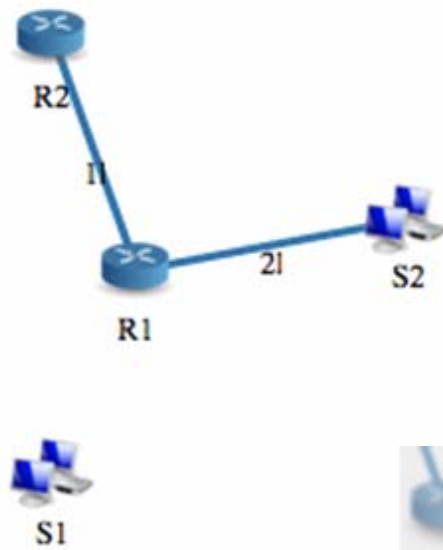
```
private function getDevicesFromOSS():Array{
    return [{name:'R1', type: 'router'}, {name:'R2', type: 'router'}, {name:'S1', type: 'device'}, {name:'S2', type: 'device'}] ;
}

private function getDevicesRelationshipFromOSS():Array{
    return [{name:'1I', from:'R1', to:'R2'}, {name:'2I', from:'R1', to:'S2'}];
}

private function translateToTWaverNode(box:ElementBox, devices:Array, relationships:Array):void{
    devices.forEach(function(device:*, index:int, arr:*)void{
        var node:Node = new Node(device.name);
        node.name = device.name;
        node.image = device.type;
        box.add(node);
    });
    relationships.forEach(function(relationship:*, index:int, arr:*)void{
        var link:Link = new Link(box.getElementByID(relationship.from) as Node, box.getElementByID(relationship.to)
as Node);
        link.name = relationship.name;
        box.add(link);
    });
}
```

# TWaver Flex Development Process

TWaver™

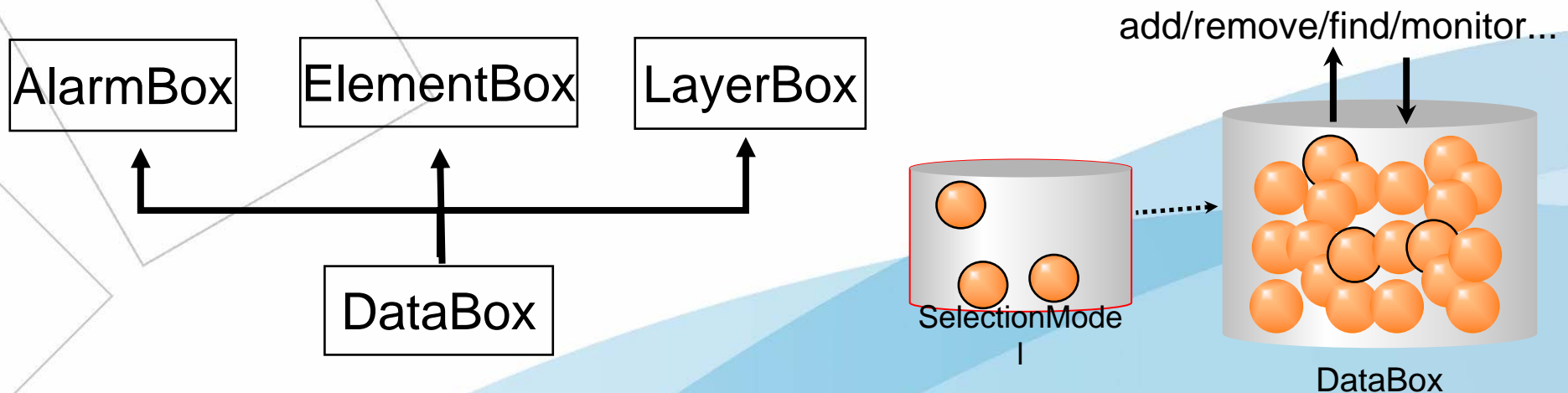


# Using DataBox

DataBox - Data container for managing data items.

- Provides basic operations like add, delete, edit
- Monitors elements changes and other events
- It's the base class for ElementBox, AlarmBox, LayerBox
- It contains selection model

DataBox#**public function get** selectionModel():SelectionModel



# Basic Operations

- Add data

**public function** add(data:IData, index:int = -1):**void**

- Delete data

**public function** remove(data:IData):**void**

**public function** removeSelection():**void**

**public function** removeByID(id:Object):**void**

- Clear data

**public function** clear():**void**

TM  
T  
W  
a  
v  
e  
r

# Getting Elements

TWaver™

- Getting data

**public function** getDataByID(id:Object):IData

**public function get** datas():ICollection

**public function** toDatas(matchFunction:Function = **null**):ICollection

**public function get** roots():ICollection

**public function** getSiblings(data:IData):ICollection

- Traverse data

**public function** forEach(callbackFunction:Function):**void**

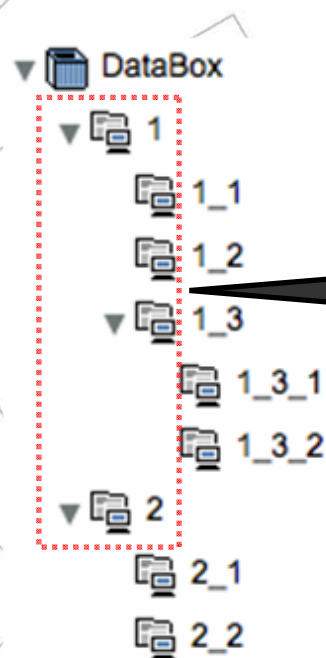
**public function** forEachByDepthFirst(callbackFunction:Function, data:IData = **null**):**void**

**public function** forEachByBreadthFirst(callbackFunction:Function, data:IData = **null**):**void**

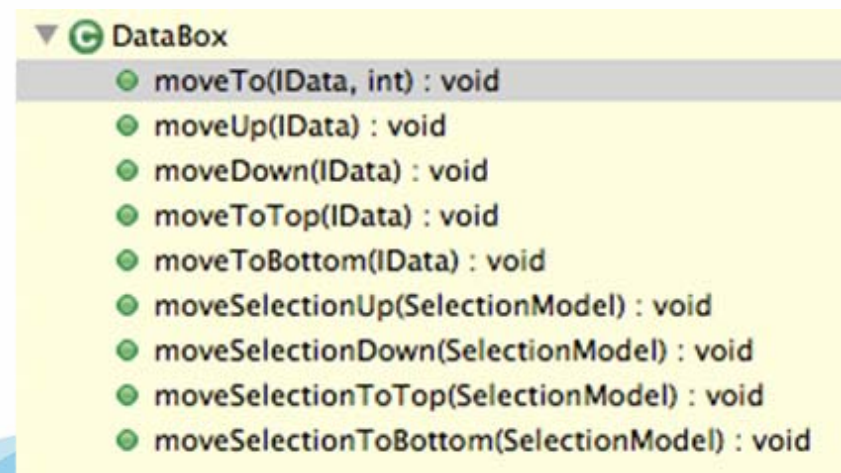
# Hierarchy of Elements

Element hierarchy is built by parent-child relationships  
move\*\*\*methods of DataBox are used to adjust  
element index

TWaver™



node 1,2 at the root  
level (rootElements)





# Quick Finder

## QuickFinder:

To find elements by name or other property of element, for example: `var result:Array = new QuickFinder(box, "name").find("PC");`

## Create a quick finder:

```
public function QuickFinder(dataBox:DataBox, propertyName:String,  
    propertyType:String = "accessor",  
    valueFunction:Function = null,  
    filterFunction:Function = null)
```

## Quick finder types:

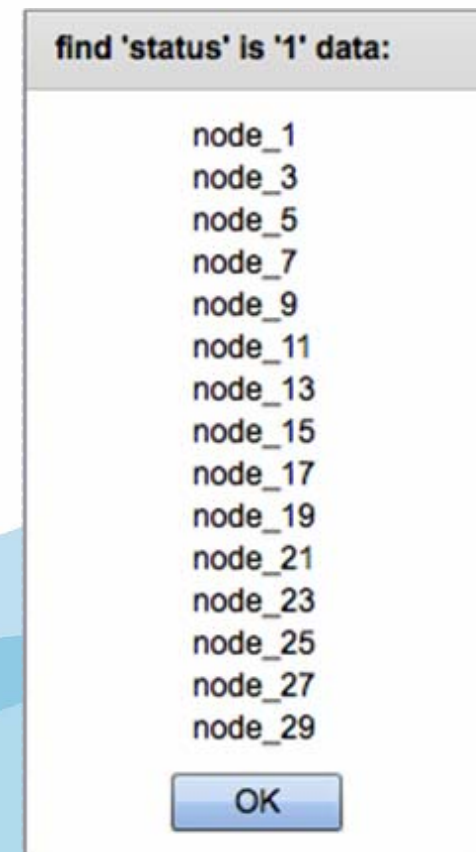
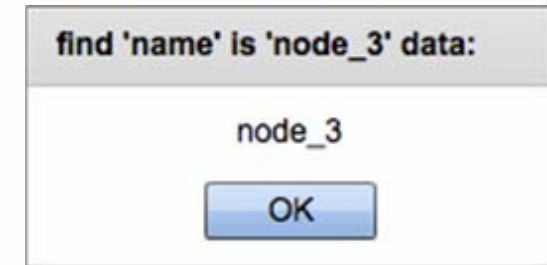
- `Consts.PROPERTY_TYPE_ACCESSOR` — find by accessor
- `propertyConsts.PROPERTY_TYPE_CLIENT` — find by client
- `propertyConsts.PROPERTY_TYPE_STYLE` — find by style property

# Quick Finder Example

```
private function init():void{
    var box:DataBox = new DataBox();
    for (var i:int = 0; i < 30; i++) {
        var data:Data = new Data();
        data.name = "node_" + i;
        data.setClient("status", i%2);
        box.add(data);
    }

    var nameFinder:QuickFinder = new QuickFinder(box, 'name');
    var result:Array = nameFinder.find("node_3");
    alertArray(result, function():void{
        var clientFinder:QuickFinder = new QuickFinder(box, "status",
Consts.PROPERTY_TYPE_CLIENT);
        alertArray(clientFinder.find(1), null);
    });
}

private function alertArray(result:Array, closeHandler:Function):void{
    var sResult:String = "";
    result.forEach(function(item:IData, index:*, arr:*)void{
        sResult += item.name + "\n";
    });
    Alert.show(sResult, null, 4, null, closeHandler);
}
```



# Listeners in DataBox

Add listener - DataBox#add\*\*\*Listener

Delete listener - DataBox#remove\*\*\*Listener

```
box.addDataPropertyChangeListener(function(evt:PropertyChangeEvent):void{  
    trace((evt.source as Data).name + "'s property changed");  
});
```

Listener	Description
addDataBoxChangeListener	detect element's add, remove, clear
addPropertyChangeListener	detect dataBox's property change
addDataPropertyChangeListener	detect element's property change
addHierarchyChangeListener	detect dataBox's sequence change

# Listener Example

```
private function init():void{
    var box:DataBox = new DataBox();
    box.addDataBoxChangeListener(function(evt:DataBoxChangeEvent):void{
        trace("data " + evt.kind + ": " + evt.data.name);
    });

    box.addDataPropertyChangeListener(function(evt:PropertyChangeEvent):void{
        trace((evt.source as Data).name + "'s property changed");
    });

    var data:Data = new Data();
    data.name = '001';
    //add data
    box.add(data);
    //modified data property
    data.name = '002';
    //remove data
    box.remove(data);
}
```

output:  
data add: 001  
002's property changed  
data remove: 002

# Import & Export DataBox

Twaver™

- DataBox can both import and export xml
- The export xml contains properties of all element and DataBox' s own properties
- Import and export by XMLSerializer



# Import & Export DataBox

- XMLSerializer - import & export

```
public function XMLSerializer(dataBox:DataBox, settings:SerializationSettings  
    = null)
```

```
public function serialize():String
```

```
public function deserialize(xmlString:String, rootParent:IData = null):void
```

- SerializationSettings - import & export setting

```
public function registerProperty(property:String, type:String,  
    cdata:Boolean=false):void
```

```
public function registerStyle(styleProp:String, type:String,  
    cdata:Boolean=false):void
```

```
public function registerClient(clientProp:String, type:String,  
    cdata:Boolean=false):void
```

# DataBox Export Example

```
var box:DataBox = new DataBox();  
box.name = 'box001';
```

```
var data:Data = new Data();  
data.name = '001';  
data.setClient('age', 27);  
box.add(data);  
var child:Data = new Data();  
child.name = '001-1';  
child.parent = data;  
box.add(child);
```

```
var settings:SerializationSettings = new SerializationSettings();  
settings.registerClient('age', Consts.TYPE_INT);  
settings.registerProperty('name', Consts.TYPE_STRING);  
settings.registerProperty("parent", Consts.TYPE_DATA);
```

```
var serializer:XMLSerializer = new XMLSerializer(box, settings);  
trace(serializer.serialize());
```

output:

```
<twaver v='1.4' p='flex'>  
<dataBox type='twaver.DataBox'>  
  <p n='name'>box001</p>  
</dataBox>  
<data type='twaver.Data' ref='0'>  
  <c n='age'>27</c>  
  <p n='name'>001</p>  
</data>  
<data type='twaver.Data' ref='1'>  
  <p n='name'>001-1</p>  
  <p n='parent' ref='0'/>  
</data>  
</twaver>
```

Twaver™

# DataBox Import Example

Twaver™

```
private var xml:XML=<twaver v='1.4' p='flex'>
<dataBox type='twaver.DataBox'>
  <p n='name'>box001</p>
</dataBox>
<data type='twaver.Data' ref='0'>
  <c n='age'>27</c>
  <p n='name'>001</p>
</data>
<data type='twaver.Data' ref='1'>
  <p n='name'>001-1</p>
  <p n='parent' ref='0'/>
</data>
</twaver>
```

```
private function init():void{
  var box:DataBox = new DataBox();

  var settings:SerializationSettings = new SerializationSettings();
  settings.registerClient('age', Consts.TYPE_INT);
  settings.registerProperty('name', Consts.TYPE_STRING);
  settings.registerProperty("parent", Consts.TYPE_DATA);

  var serializer:XMLSerializer = new XMLSerializer(box, settings);

  serializer.deserialize(xml.toXMLString());

  trace('box name is ' + box.name);
  trace('box count is ' + box.count);
  box.forEach(function(data:Data):void{
    trace('data name is ' + data.name);
    if(data.getClient("age")){
      trace('age is ' + data.getClient('age'));
    }
    if(data.parent){
      trace('parent is ' + data.parent.name);
    }
  })
}
```

output:

```
box name is box001
box count is 2
data name is 001
age is 27
data name is 001-1
parent is 001
```

# Using ElementBox

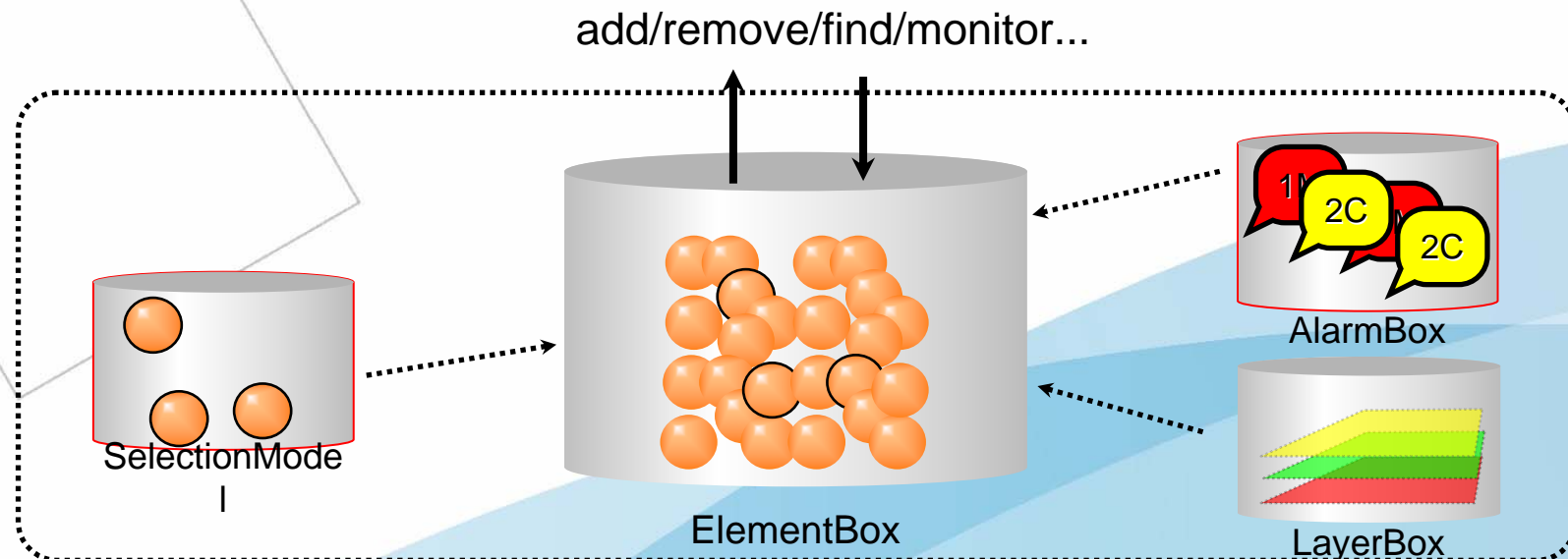
**ElementBox** - the data container for topology elements management

Inherited from DataBox, retain the functions of DataBox, increase layer box and alarm box

ElementBox#

```
public function get layerBox():LayerBox
```

```
public function get alarmBox():AlarmBox
```



# Using ElementBox

- Increase layerBox、alarmBox

```
public function get layerBox():LayerBox
```

```
public function get alarmBox():AlarmBox
```

- Traverse elements by layer

```
public function forEachByLayer(callbackFunction:Function, layer:ILayer =  
    null):void
```

```
public function forEachByLayerReverse(callbackFunction:Function, layer:ILayer =  
    null):void
```

- Increase layer change listener

```
public function addIndexChangeListener(listener:Function, priority:int = 0,  
    useWeakReference:Boolean = false):void
```



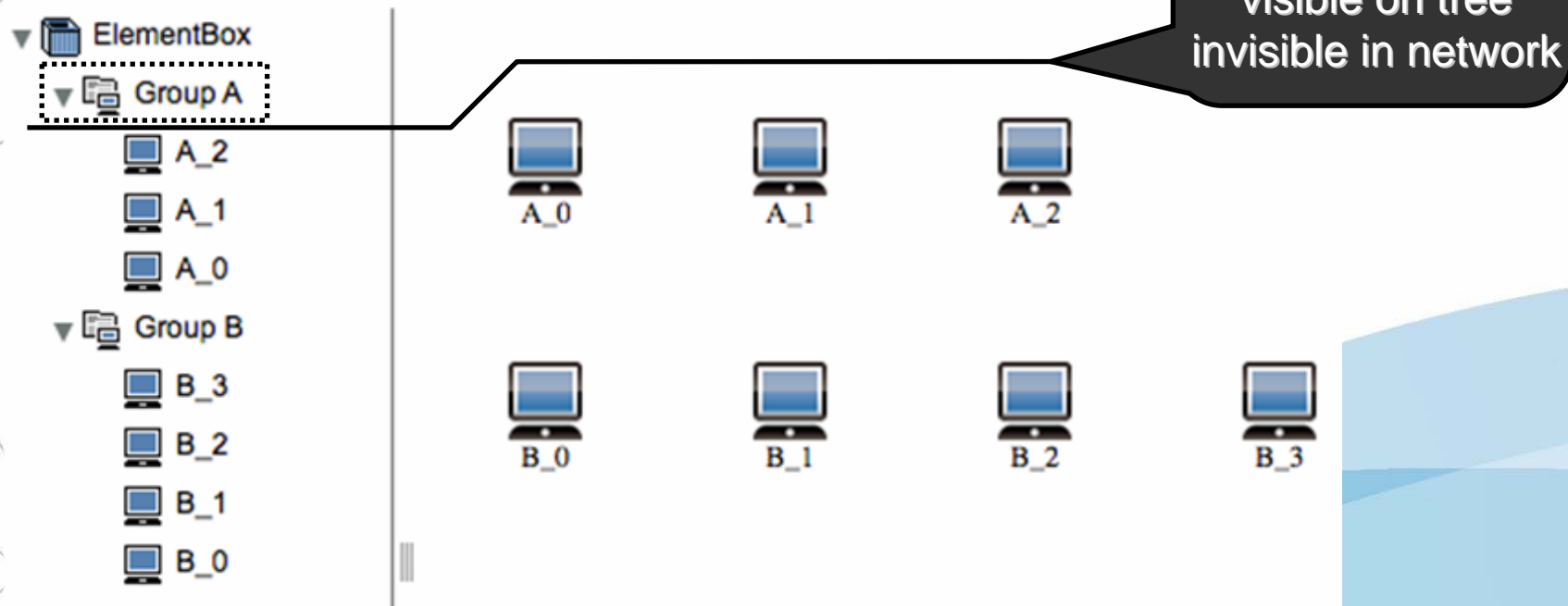
# Elements

## Data

- |-- Alarm
- |-- Layer
- |-- Element
  - |-- Dummy
  - |-- Node
    - |-- Follower
    - |-- Group
    - |-- Grid
    - |-- ShapeNode
    - |-- Bus
    - |-- SubNetwork
  - |-- Link
    - |-- ShapeLink
    - |-- LinkSubNetwork

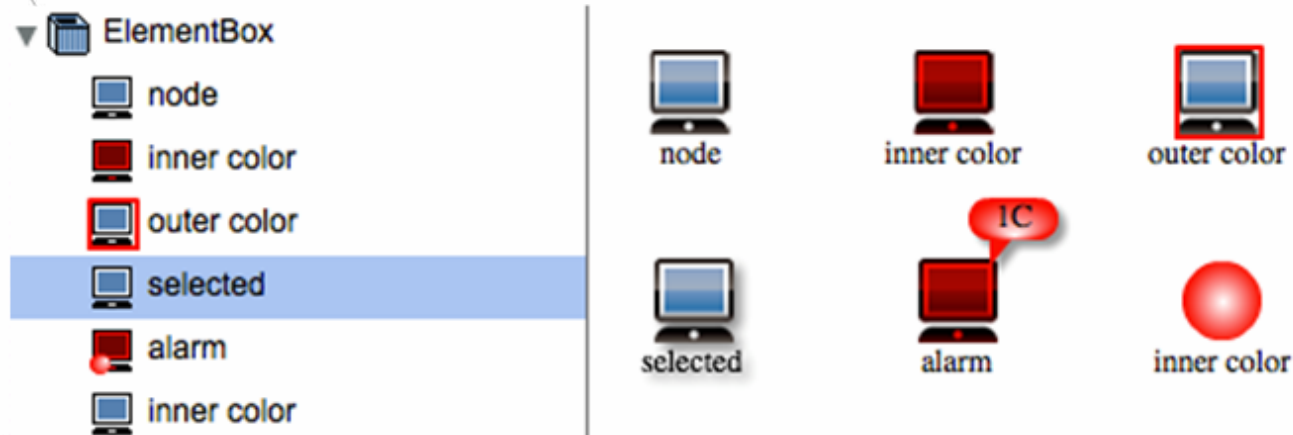
# Dummy

- Dummy is invisible on network, but visible on Tree and Table
- Dummy can be used to reorganize tree structure and avoid to affect network



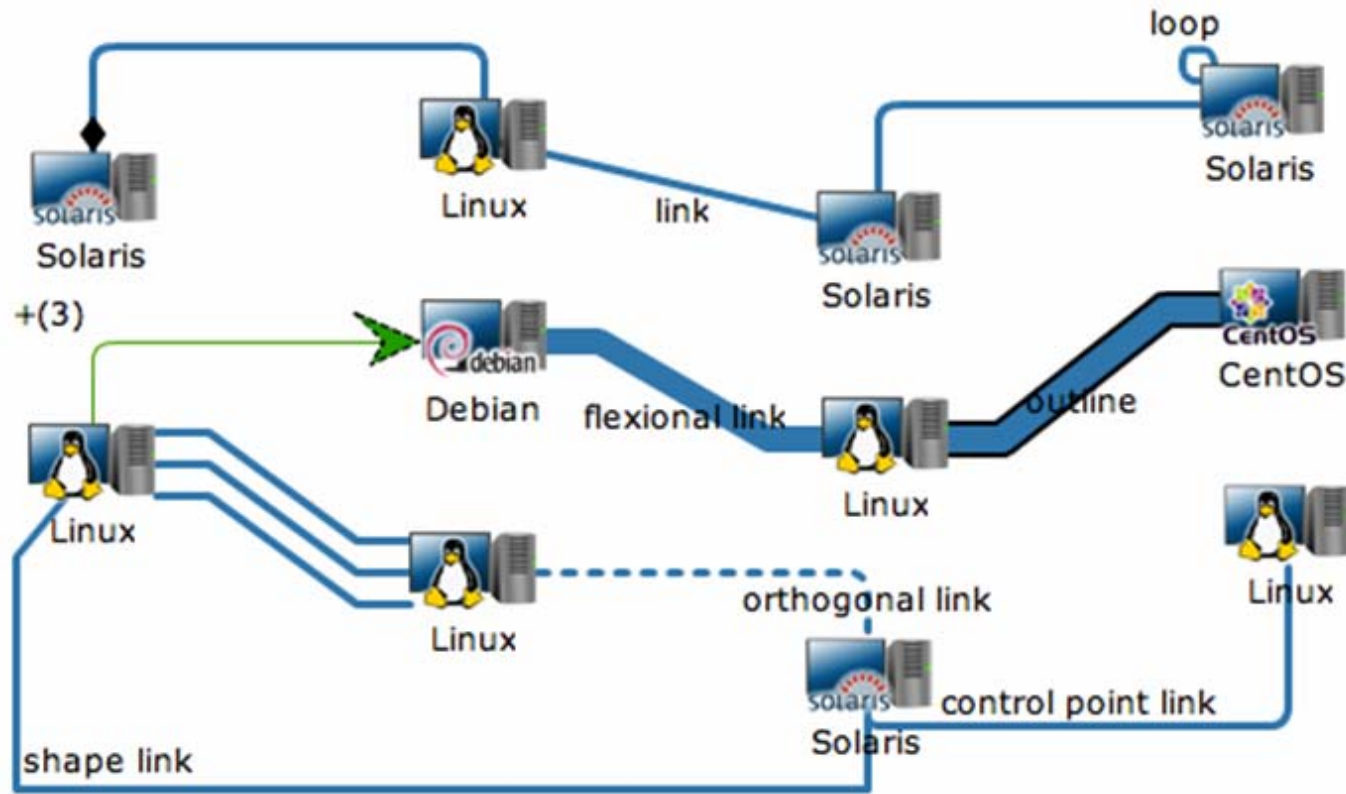
# Node

- Basic node type, can set image, outline, render color ...
- twaver.Node is the base class for other elements



# Link

- Link, linking two nodes, it has many types and styles, it provides arrows, loop, flowing etc.

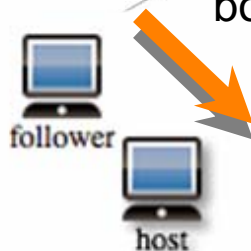


# Follower

- Follower, it can set a host node, follower will follow it when host is moving.

```
var node:Follower = new Follower();  
node.location = new Point(100, 100);  
node.name = 'follower';  
box.add(node);
```

```
var host:Node = new Node();  
host.location = new Point(140, 140);  
node.host = host;  
host.name = 'host';  
box.add(host);
```



host move, and follower move, too.

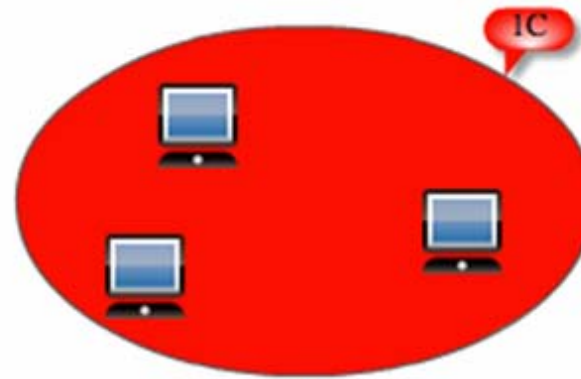


# Group

- Group, contains other nodes, it can close or open, it has many types of shapes



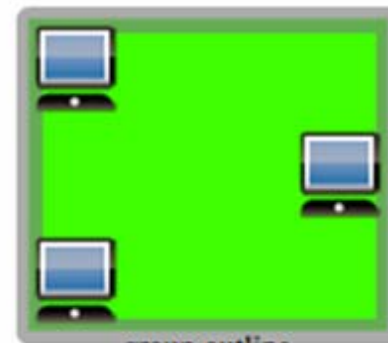
group



group with alarm



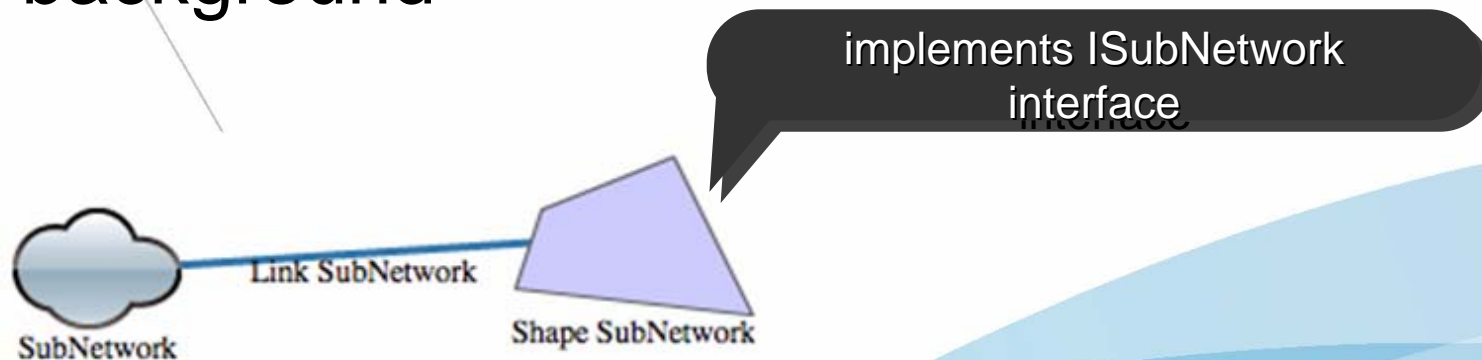
group gradient



group outline

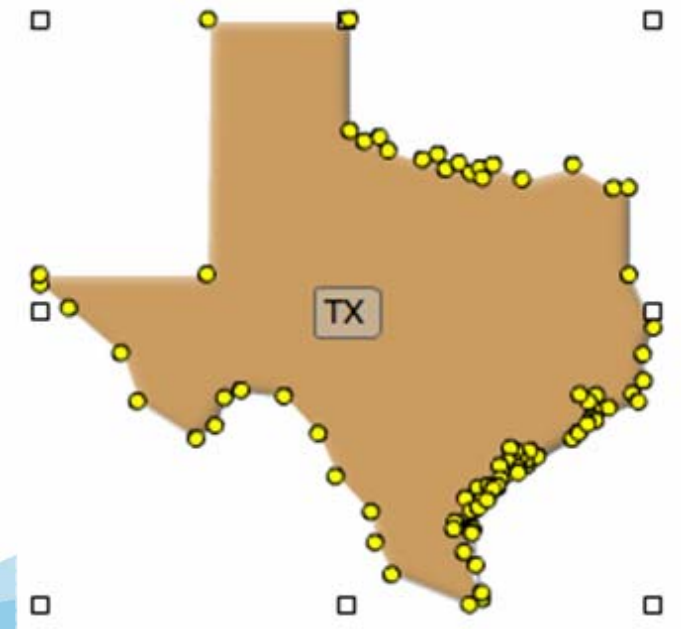
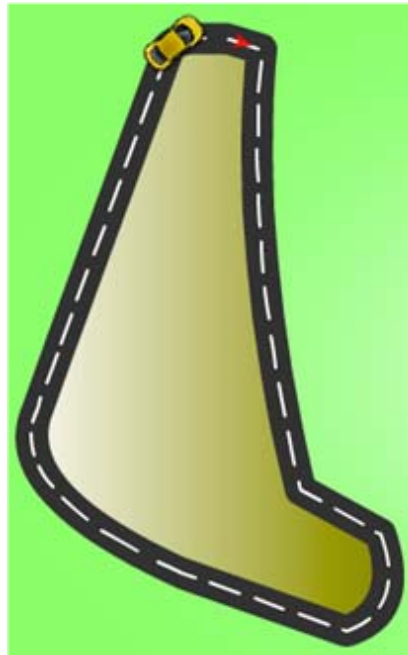
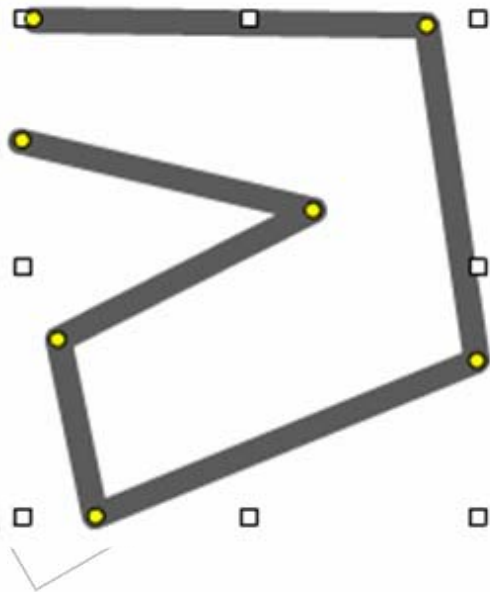
# SubNetwork

- Subnetwork, you can drop in or out by double click, when you enter into a subnetwork, the children of the subnetwork will be displayed, Subnetwork has its own background



# ShapeNode

- ShapeNode, surrounded by a number of control points.



# Bus

- Inherited from ShapeNode, connected by a set of control points into line, the links that connected to it are orthogonal distribution



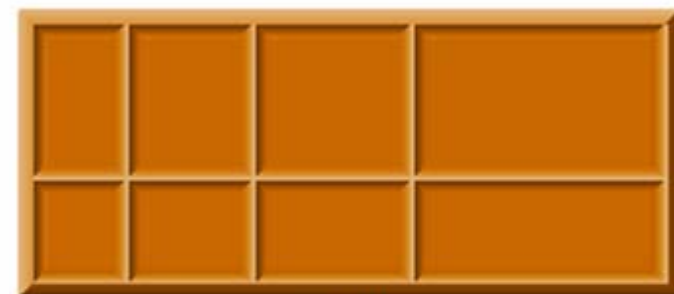
# Grid

- Grid, displays child elements in rows and columns, one grid can embed into an other grid, can be used for creating equipment chassis

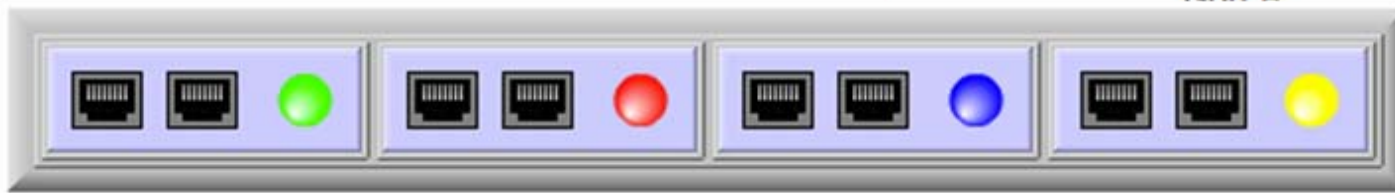
```
var grid:Grid = new Grid();
grid.name = 'Grid B';
grid.setStyle(Styles.GRID_BORDER, 5);
grid.setStyle(Styles.GRID_COLUMN_COUNT, 4);
grid.setStyle(Styles.GRID_COLUMN_PERCENTS,
[0.15,0.2,0.25,0.4]);
grid.setStyle(Styles.GRID_ROW_COUNT, 2);
grid.setStyle(Styles.GRID_ROW_PERCENTS, [0.6,0.4]);
grid.setStyle(Styles.GRID_FILL_COLOR, 0xCC6600);
grid.setStyle(Styles.GRID_DEEP, 11);
grid.setStyle(Styles.GRID_CELL_DEEP, -4);
grid.setStyle(Styles.GRID_PADDING, 0);
grid.width = 250;
grid.height = 106;
```



Grid A



Grid B





# Element Properties

twaver.Consts#

```
public static const PROPERTY_TYPE_ACCESSOR:String = "accessor";  
public static const PROPERTY_TYPE_CLIENT:String = "client";  
public static const PROPERTY_TYPE_STYLE:String = "style";
```

Element

**accessor** - name, id ...

node.name = '001';

**style** - style properties

node.setStyle(Styles.INNER\_COLOR, 0xFF0000);

**client** - client properties

node.setClient('age', 27);

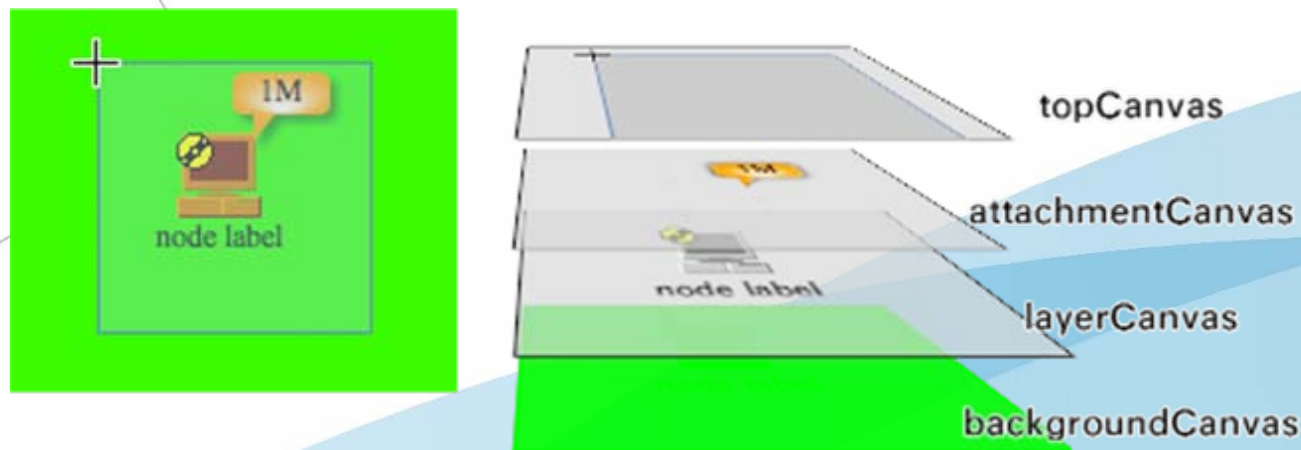
# Using Network

- Network hierarchy
- Network background
- Network interactions
- Network filters
- Network generators
- Network auto layout

Twaver™

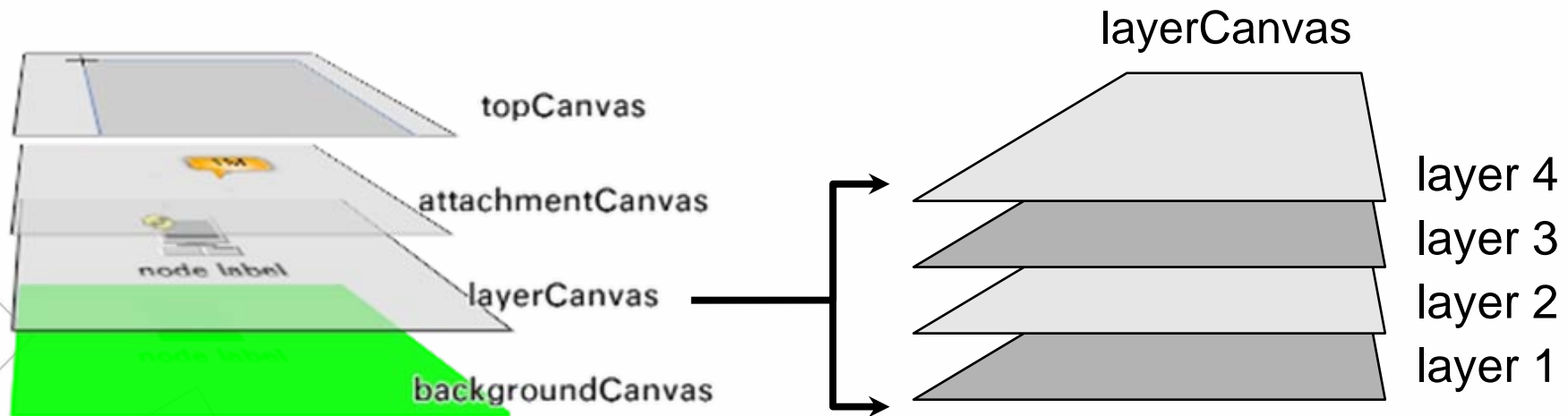
# Network Hierarchy

- rootCanvas — main canvas, contains all topology components
  - topCanvas — — can be used for painting selection rectangle
  - attachmentCanvas — — contains top attachments, like alarm bubble
  - layerCanvas — — contains all nodes and links
    - layer n
    - layer ...
    - default layer
  - bottomCanvas — — bottom canvas, painting shading under the background
  - backgroundCanvas — — color background or image background



# Layer Management

TM  
waver



```
var box:ElementBox = network.elementBox;
var layerBox:LayerBox=box.layerBox;

var layerTop:Layer=new Layer("top", 'top');
layerBox.add(layerTop, layerBox.count);

var node:Node = new Node();
box.add(node);
var node2:Node = new Node();
box.add(node2);
var link:Link = new Link(node, node2);
link.layerID = layerTop.id;
link.name = 'link at top';
link.setStyle(Styles.LINK_COLOR, 0x000000);
box.add(link);
```



# Adding Background

TWaver™

- Can set the background to ElementBox and subnetworks, supports color, image or gradient background.

```
box.setStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_IMAGE);  
box.setStyle(Styles.BACKGROUND_IMAGE, "background");
```

- Flex component also supports its own background styles

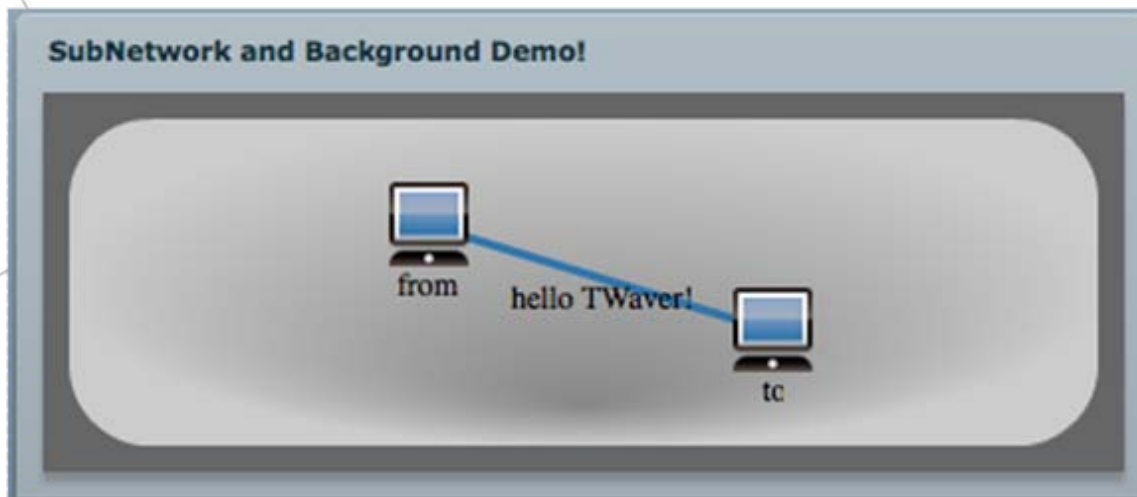
```
<twaver:Network id="network" backgroundColor="0x666666" width="100%"  
height="100%" />
```



# Background Example

```
<twaver:Network id="network" backgroundColor="0x666666" width="100%" height="100%" />
```

```
var subnetwork:SubNetwork=new SubNetwork();  
subnetwork.setStyle(Styles.BACKGROUND_TYPE, Consts.BACKGROUND_TYPE_VECTOR);  
subnetwork.setStyle(Styles.BACKGROUND_VECTOR_GRADIENT, Consts.GRADIENT_RADIAL_SOUTH);  
subnetwork.setStyle(Styles.BACKGROUND_VECTOR_FILL_COLOR, 0xCCCCCC);  
subnetwork.setStyle(Styles.BACKGROUND_VECTOR_GRADIENT_COLOR, 0x888888);  
subnetwork.setStyle(Styles.BACKGROUND_VECTOR_SHAPE, Consts.SHAPE_ROUNDSRECT);  
subnetwork.setStyle(Styles.BACKGROUND_VECTOR_PADDING, 10);
```



# Network Interaction Mode

- **InteractionHandler** - single listener
- **Interaction Mode** - several input handlers combined into an interaction mode

Network#

public function set

interactionHandlers(interactionHandlers:ICollection):void

- **Predefined Modes** - Network provides several predefined interaction modes, such as the default mode, edit mode, creating link mode ...

Network#

public function setDefaultInteractionHandlers(lazyMode:Boolean = false):void

public function setEditInteractionHandlers(lazyMode:Boolean = false):void

public function setCreateLinkInteractionHandlers(linkClass:Class = null):void

...

# Custom Interaction

## interaction handler class

```
public class HighlightInputHandler extends BasicInteractionHandler{
    public function HighlightInputHandler(network:Network){
        super(network);
    }
    override public function installListeners():void{
        this.network.addEventListener(MouseEvent.CLICK, handleMouseMove);
    }
    override public function uninstallListeners():void{
        this.network.removeEventListener(MouseEvent.CLICK, handleMouseMove);
    }
    ...
}
```

## setting interaction mode

```
network.interactionHandlers = new Collection([
    new SelectInteractionHandler(network),
    new MoveInteractionHandler(network),
    new DefaultInteractionHandler(network),
    new HighlightInputHandler(network)
]);
```



mouse over,  
node highlight

```

public class HighlightInputHandler extends BasicInteractionHandler{
    public function HighlightInputHandler(network:Network){
        super(network);
    }
    override public function installListeners():void{
        this.network.addEventListener(MouseEvent.MOUSE_MOVE,
handleMouseMove);
    }
    override public function uninstallListeners():void{
        this.network.removeEventListener(MouseEvent.MOUSE_MOVE, handleMouseMove);
    }
    private function handleMouseMove(e:MouseEvent):void{
        var element:IElement = network.getElementByMouseEvent(e);
        if(element != null){
            highlight(element);
        }else{
            reset();
        }
    }
    private var highlightElement:IElement;
    private var highLightColor:uint = 0xFF8888;
    private var oldColor:int = -1;

    [Embed(source="hand_cursor.png")]
    private var handCursor:Class;

    private function highlight( element:IElement):void {
        if(element == null || element == highlightElement){
            return;
        }
        reset();
        highlightElement = element;
        oldColor = element.getStyle(Styles.INNER_COLOR);
        CursorManager.setCursor(handCursor, 2, -9, -3);
        element.setStyle(Styles.INNER_COLOR, highLightColor);
    }
    private function reset():void {
        if (highlightElement != null) {
            if(oldColor){
                highlightElement.setStyle(Styles.INNER_COLOR, oldColor);
            }else{
                highlightElement.setStyle(Styles.INNER_COLOR, null);
            }
            highlightElement = null;
            CursorManager.removeAllCursors();
        }
    }
}

```

TWaver™

# Network Filters

Filters are used for global control, such as control elements to show or hide, move or not move

Network#

//visible filter

`public` function get/set visibleFunction():Function

//movable filter

`public` function get/set movableFunction():Function

//editable filter

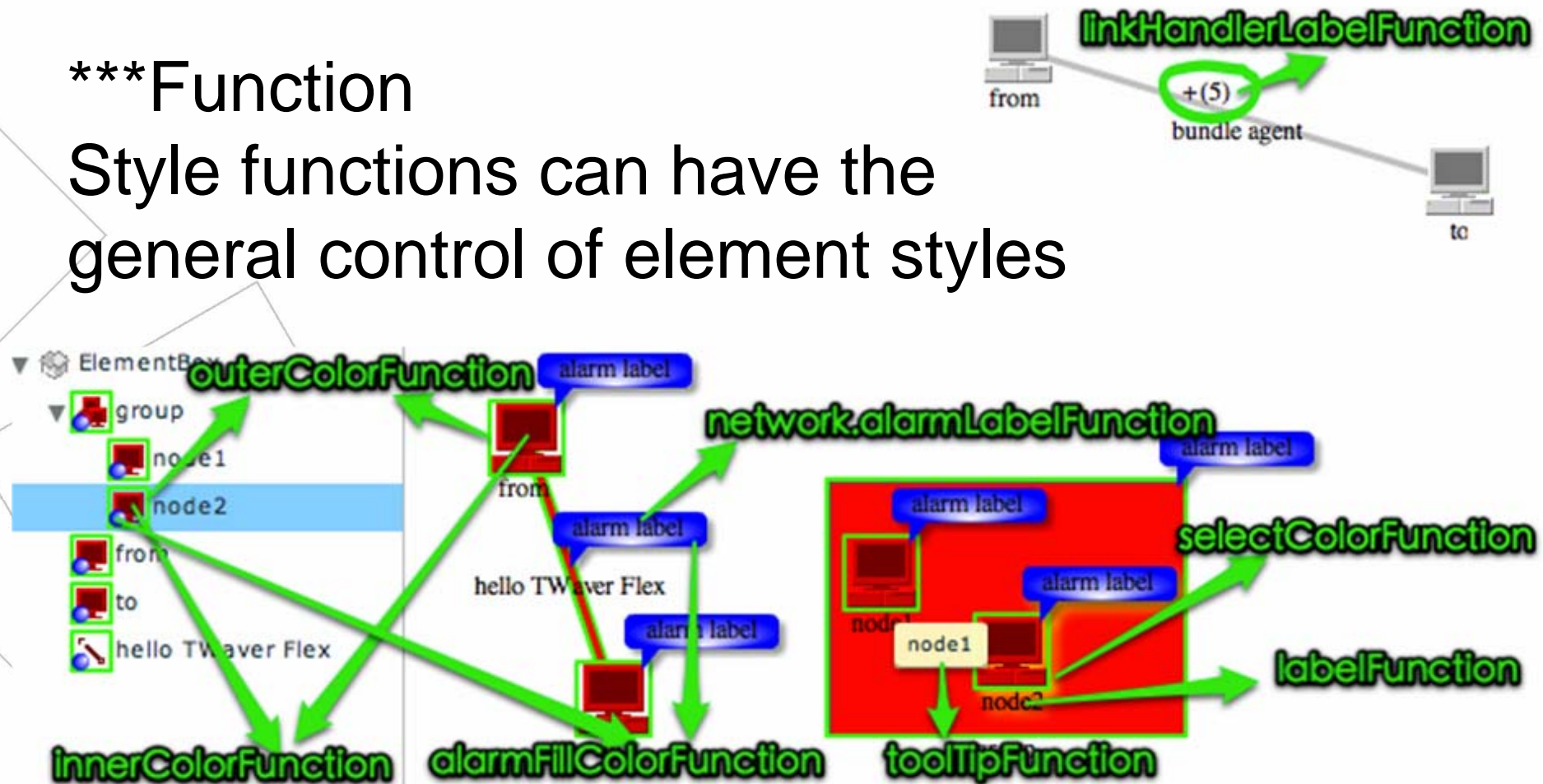
`public` function get/set editableFunction():Function



# Network Style Functions

\*\*\*Function

Style functions can have the general control of element styles

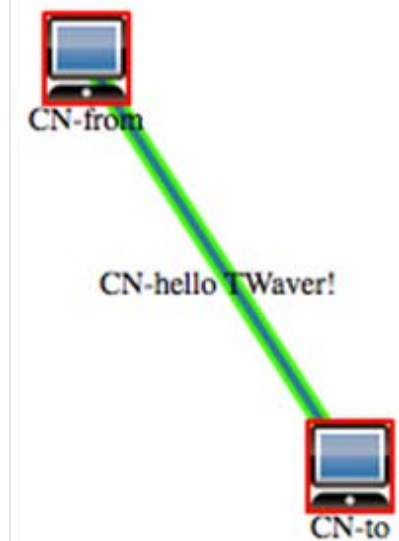


# Style Function Example

Example for adding prefix “CN-”  
to element label, and  
customizing outline color

```
network.labelFunction = function(element:IElement):String{  
    return "CN-" + element.name;  
};
```

```
network.outerColorFunction = function(element:IElement):uint{  
    return (element is Node) ? 0xFF0000 : 0x00FF00;  
};
```

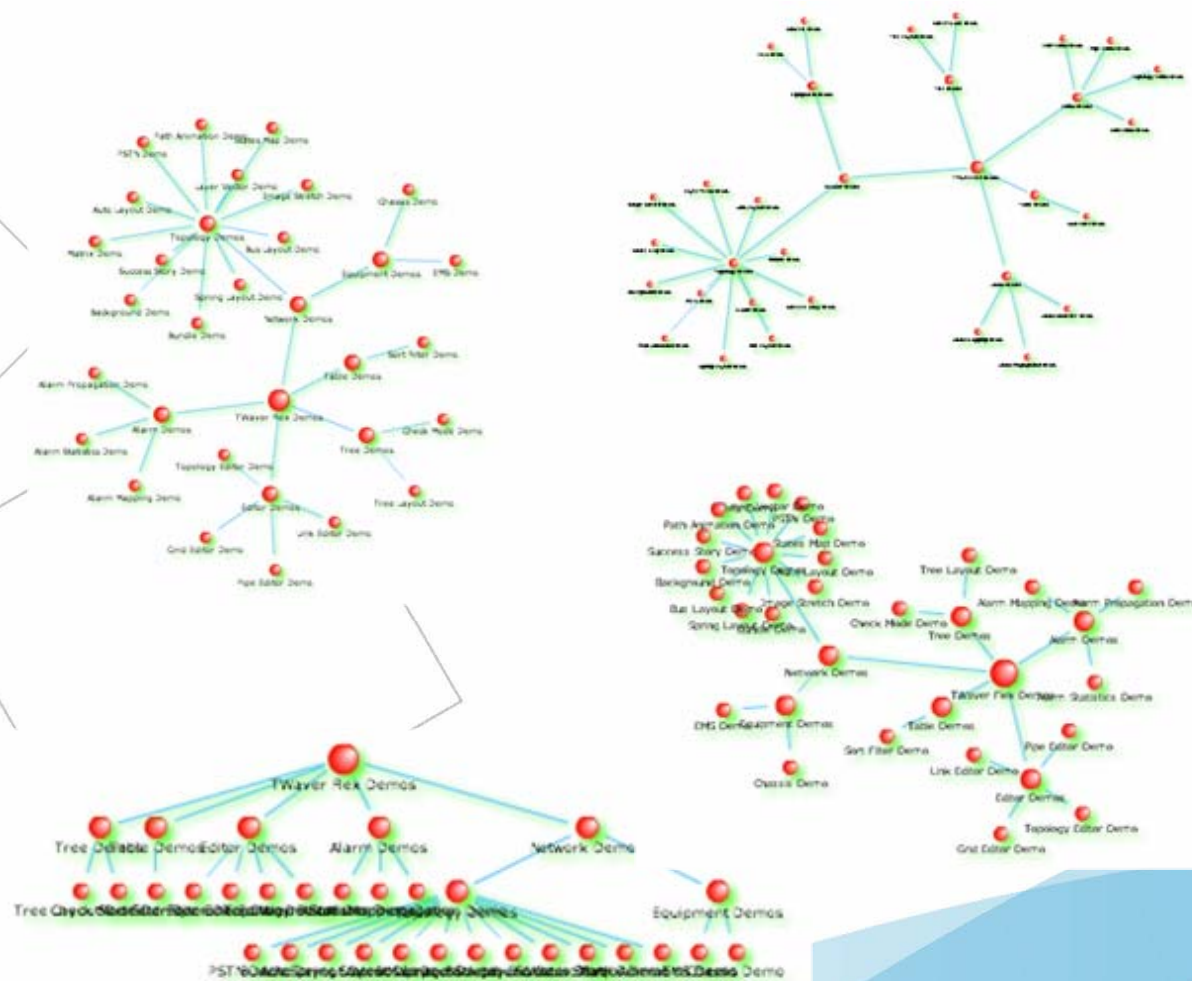


# Network Auto Layouts

- Automatic positioning nodes and links, make them spread out, and showing good.



# Default Layouts







- Home - ServaSoftware.com
- Email - [tw-service@servasoft.com](mailto:tw-service@servasoft.com)



# Common Components & Alarms

TM  
waver

- **Common components**

- Tree / Table / Chart

- **Alarm**

- Alarm / AlarmSeverity / AlarmBox / AlarmState / AlarmStateStatistics / Alarm views

- **Flex component expansion**

- CSS configuration / skin customization / component extension

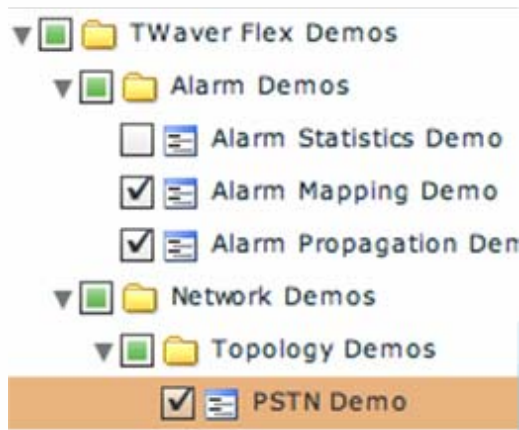
# Using Tree

TWaver™

twaver.controls.Tree - provides a hierarchical view of elements contained in a DataBox

```
var tree:Tree = new Tree(box);
```

```
<twaver:Tree id="tree" width="100%" height="100%" />
```



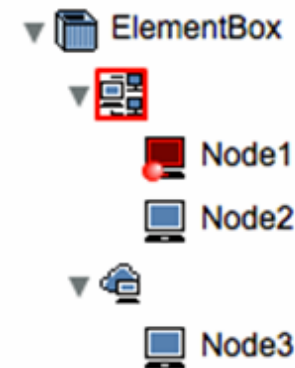
# Creating a Tree

```
var box:DataBox = new ElementBox();  
tree.dataBox = box;
```

```
var group:Group = new Group();  
box.add(group);  
var subnetwork:SubNetwork = new SubNetwork();  
box.add(subnetwork);
```

```
var node1:Node = new Node();  
node1.name = "Node1";  
node1.parent = group;  
node1.alarmState.increaseNewAlarm(AlarmSeverity.CRITICAL);  
box.add(node1);  
var node2:Node = new Node();  
node2.name = "Node2";  
node2.parent = group;  
box.add(node2);  
var node3:Node = new Node();  
node3.name = "Node3";  
node3.parent = subnetwork;  
box.add(node3);
```

```
tree.callLater(function():void{  
    tree.expandAll();  
});
```



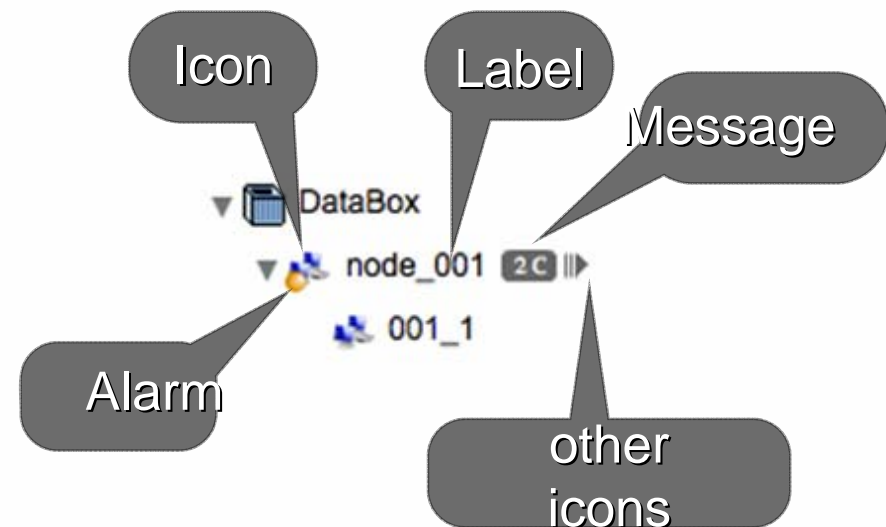
sets element's parent-child relationship

# Customizing Tree Node

- **Icon**  
`data.icon = 'iconName';`

- **Label**  
`data.name = '001';`  
`element.setStyle(Styles.TREE_LABEL, '002');`

- **Other styles**  
`element.setStyle(Styles.TREE_LABEL_***, ***);`  
`element.setStyle(Style.TREE_MESSAGE_***, ***);`  
`element.setStyle(Style.TREE_ICONS_***, ***);`



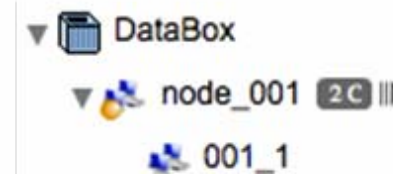
# Tree Node Style Example

```
[Embed(source="images/device.png")]
private static const devicelcon:Class;
[Embed(source="images/out.gif")]
private static const outlcon:Class;
private function init():void{
    Utils.registerImageByClass("devicelcon",devicelcon);
    Utils.registerImageByClass("outlcon",outlcon);

    var box:DataBox = tree.dataBox;

    var node:Element = new Element();
    node.name = '001';
    node.icon = 'devicelcon';
    node.setStyle(Styles.TREE_LABEL, 'node_' + node.name);
    node.setStyle(Styles.TREE_ICONS_NAMES, ['outlcon']);
    node.setStyle(Styles.TREE_LAYOUT_GAP, 3);
    node.setStyle(Styles.TREE_MESSAGE, '2 C');
    node.setStyle(Styles.TREE_MESSAGE_FILL_COLOR, 0x666666);
    node.setStyle(Styles.TREE_MESSAGE_GRADIENT, Consts.GRADIENT_NONE);
    node.setStyle(Styles.TREE_MESSAGE_SIZE, 9);
    node.setStyle(Styles.TREE_MESSAGE_YPADDING, -3);
    node.setStyle(Styles.TREE_MESSAGE_COLOR, 0xFFFFFFFF);
    node.setStyle(Styles.TREE_LAYOUT, Consts.TREE_LAYOUT_LABEL_MESSAGE_ICONS);
    node.setStyle(Styles.TREE_ALARM_FILL_COLOR, 0xE08000);
    box.add(node);

    ...
}
```





# Node Hierarchy and Order

Twaver™

- Parent-child hierarchy  
`node.parent = parent;`  
`node.addChild(child);`
- Element order  
`box.move***(node);`
- Sort by setting comparer  
`tree.compareFunction = compareFunction;`

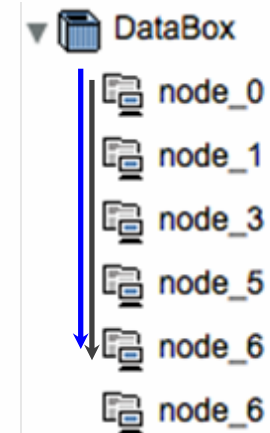


# Tree Sort Example

```
var box:DataBox = tree.dataBox;

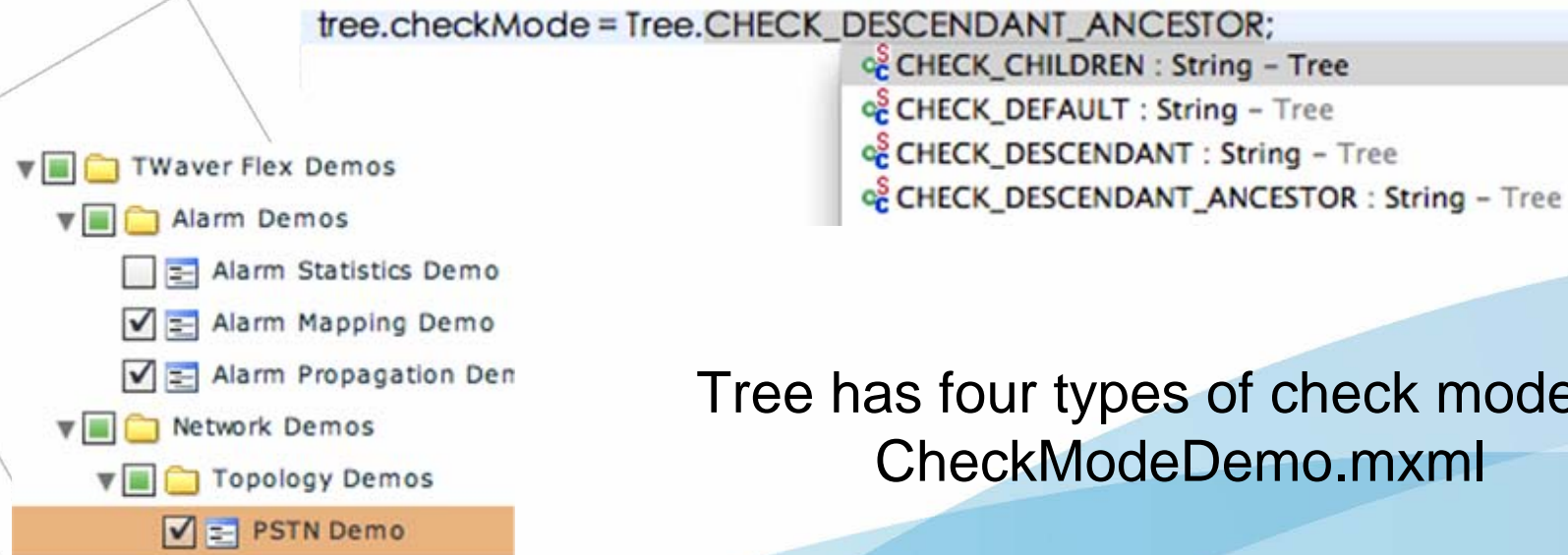
for(var i:int = 0; i < 10; i++){
    var data:Data = new Data();
    data.name = 'node_' + Utils.randomInt(10);
    box.add(data);
}

tree.compareFunction = function(d1:IData, d2:IData):int{
    return d1.name.localeCompare(d2.name);
};
```



# Tree Check Mode

- Tree supplies check mode, setting as follow:
- *Tree.checkMode = Tree.CHECK\_\*\*\**



Tree has four types of check modes:  
CheckModeDemo.mxml

# Using Table

- Bind with DataBox, list all data that in the DataBox, each row is one data
- Support the display for three types of data properties: accessor、 style、 client
- Data property change and the table update automatically
- Table inherited from mx.controls.DataGrid, supports renderer & editor customization

TM  
Twaiver

# Creating a Table

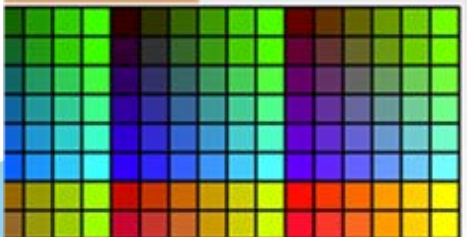
TWaver™

twaver.controls.Table - List all data that in the DataBox, each row is one data

```
var table:Table = new Table(box);
```

```
<twaver:Table id="table" width="100%" height="100%" />
```

Name ▼	Number	Inner Color
node_5	5	16229533
node_4	4	5731022
node_3	3	4504295
node_2	2	4997537
node_1	1	8846834

Fill	Fill Color	Fill Alpha
<input checked="" type="checkbox"/>	C6D578	1
C6D578		1
		1
		1
		1
		1
		1
		1
		1

# Create Table by MXML

Supports three types of properties:  
accessor、style、client

```
<twaver:Table id="table1" editable="true">
  <twaver:columns>
    <twaver:TableColumn dataField="name" headerText="Name"/>
    <twaver:TableColumn dataField="C:number" headerText="Number"
      itemEditor="{new ClassFactory(mx.controls.NumericStepper)}"
      editorDataField="value"/>
    <twaver:TableColumn dataField="S:{Styles INNER_COLOR}"
      headerText="Inner Color" editable="false" />
  </twaver:columns>
</twaver:Table>
var box:DataBox = table1.dataBox;
var i:int=0;
while(i++<5){
  var node:Node = new Node();
  node.name = "node_"+i;
  node.setClient("number",i);
  node.setStyle(Styles.INNER_COLOR, Utils.randomColor);
  box.add(node);
}
```

Name ▼	Number	Inner Color
node_5	5	16229533
node_4	4	5731022
node_3	3	4504295
node_2	2	4997537
node_1	1	8846834



# Create Table by API

TWaver™

```
var table2:Table = new Table(box);  
table2.editable = true;
```

```
var nameColumn:TableColumn = new TableColumn("Name");  
nameColumn.dataField = 'name';
```

```
var clientColumn:TableColumn = new TableColumn("Number");  
clientColumn.client = 'number';  
clientColumn.itemEditor = new ClassFactory(mx.controls.NumericStepper);  
clientColumn.editorDataField = 'value';
```

```
var colorColumn:TableColumn = new TableColumn("Inner Color");  
colorColumn.style = Styles.INNER_COLOR;  
colorColumn.editable = false;
```

```
table2.columns = [nameColumn, clientColumn, colorColumn];
```

```
table1.parent.addChild(table2);
```



# Table Rows Order

- Table has three types of orders - reverse, forward and roots

```
table.iterateMode = Consts. ITERATE_MODE_***;
```

- Consts.ITERATE\_MODE\_REVERSE
- Consts.ITERATE\_MODE\_FORWARD
- Consts.ITERATE\_MODE\_ROOTS

- Adjusting row order by method of `dataBox.move***(element)`

# Table Column Sort

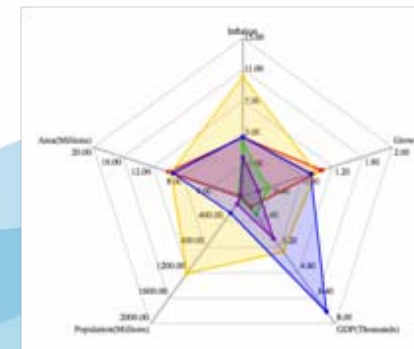
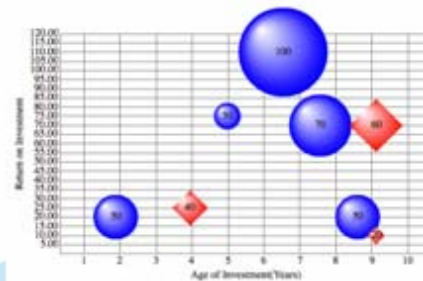
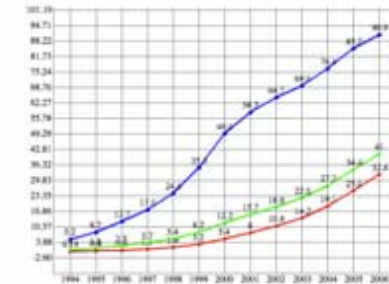
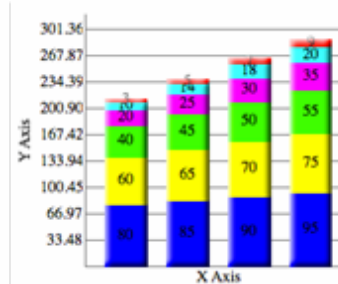
```
var sort:Sort = new Sort();  
//sort.fields = [new SortField("name"), new SortField("C:number")];  
sort.fields = [new SortField("name", false, true)];  
(table1.dataProvider as ArrayCollection).sort = sort;  
(table1.dataProvider as ArrayCollection).refresh();
```

Name ▼	Number	Inner Color
node_5	5	16229533
node_4	4	5731022
node_3	3	4504295
node_2	2	4997537
node_1	1	8846834

Name	Number	Inner Color
node_0	3	1923848
node_0	6	14986603
node_0	9	10550708
node_1	1	1426661
node_1	4	7830315
node_1	7	9846957

# Using Charts

- BarChart
- LineChart
- PieChart
- DialChart
- BubbleChart
- RadarChart



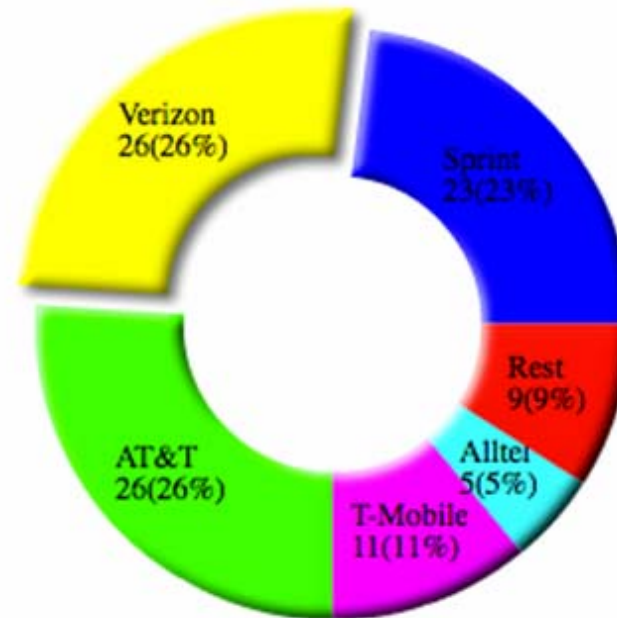
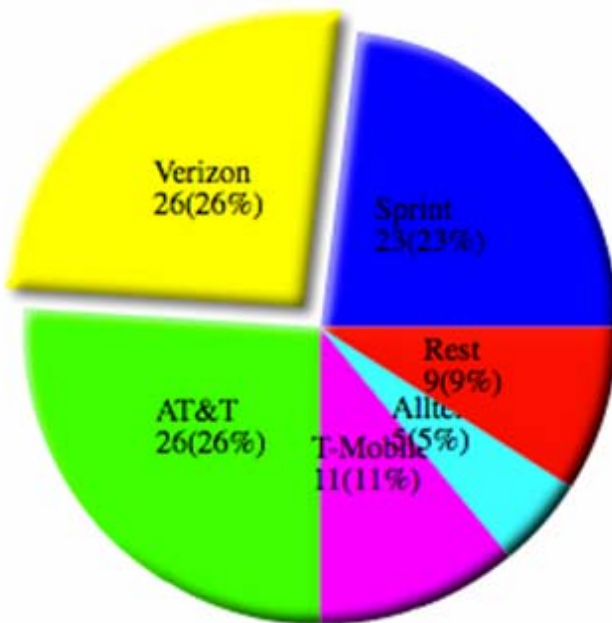
# ChartValue & ChartValues

- Chart associated with the DataBox, graphically display the chart values of elements in the DataBox
- There are two types of chart values:  
***CHART\_VALUE & CHART\_VALUES***
- setting chart values as follows:

```
element.setStyle(Styles.CHART_VALUE, 27);
```

```
element.setStyle(Styles.CHART_VALUES, [27, 37, 48]);
```

# PieChart





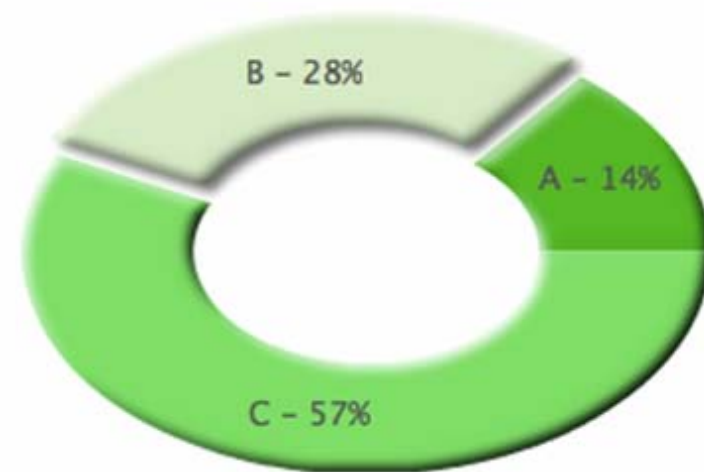
# PieChart

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex" >
    <mx:initialize>
        <![CDATA[
            import twaver.*;
            import twaver.network.Network;

            var box:DataBox = chart.dataBox;
            createData('A', 10, 0x4CB743);
            createData('B', 20, 0xD6EAC9);
            createData('C', 40, 0x77DD77);

            chart.type = Consts.PIECHART_TYPE_OVALDONUT;
            chart.valueTextFunction = function(item:IElement, value:Number):String{
                return item.name + ' - ' + int(value/chart.sum*100) + '%';
            };

            function createData(name:String, value:Number, color:uint):void{
                var data:IElement = new Element();
                data.setStyle(Styles.CHART_VALUE, value);
                data.setStyle(Styles.CHART_COLOR, color);
                data.setStyle(Styles.CHART_VALUE_COLOR, 0x555555);
                data.setStyle(Styles.CHART_VALUE_FONT, 'hevetica');
                data.name = name;
                box.add(data);
            };
        ]]>
    </mx:initialize>
    <twaver:PieChart id="chart" backgroundColor="0xFFFFF" width="100%" height="100%"/>
</mx:Application>
```

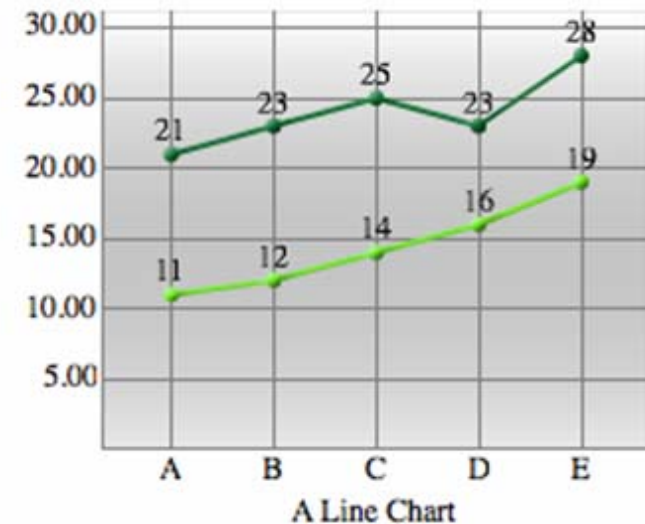


# LineChart

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:twaver="http://www.servasoftware.com/2009/twaver/flex" >
    <mx:initialize>
        <![CDATA[
            import twaver.*;
            import twaver.network.Network;

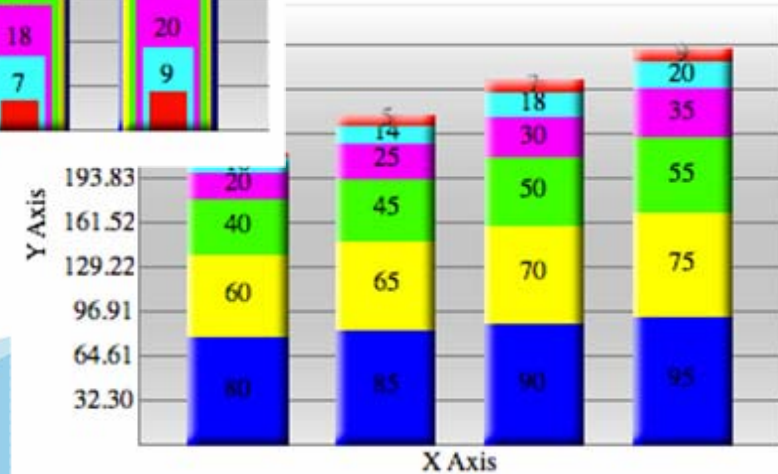
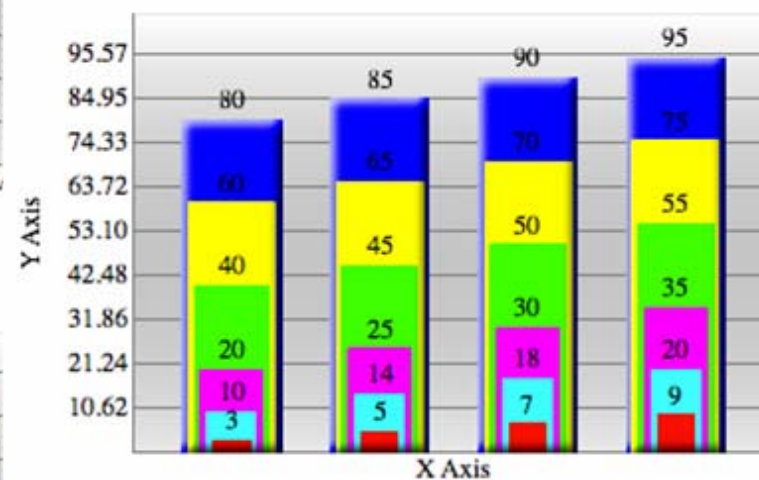
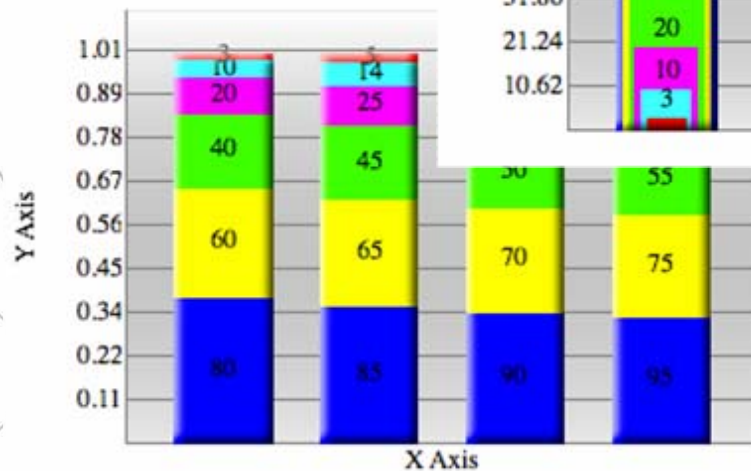
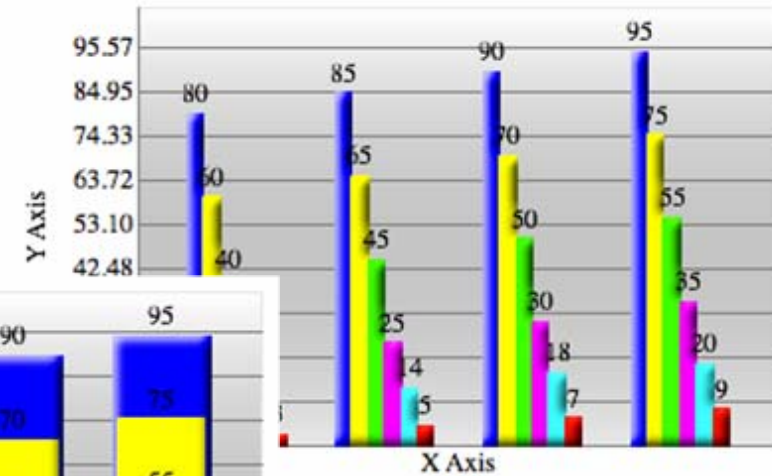
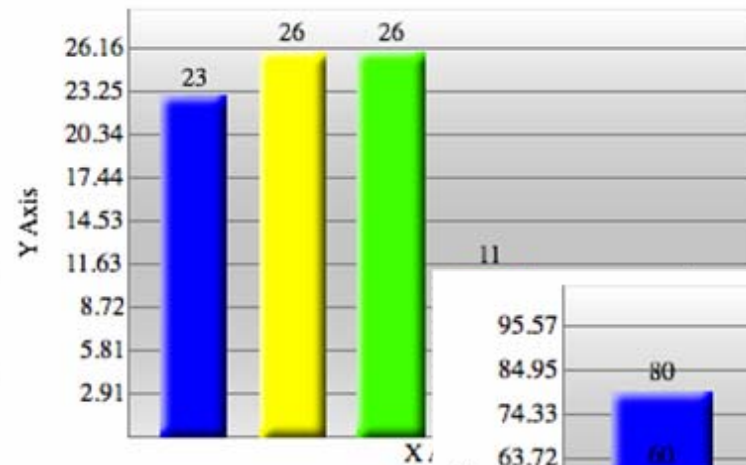
            var box:DataBox = chart.dataBox;
            var data:IElement = new Element();
            data.setStyle(Styles.CHART_VALUES, [11, 12, 14, 16, 19]);
            box.add(data);
            data = new Element();
            data.setStyle(Styles.CHART_VALUES, [21, 23, 25, 23, 28]);
            box.add(data);

            chart.xScaleTexts = ['A', 'B', 'C', 'D', 'E'];
            chart.lowerLimit = 0;
            chart.yScaleValueGap = 5;
            chart.xAxisText = 'A Line Chart';
            chart.backgroundVisible = true;
        ]]>
    </mx:initialize>
    <twaver:LineChart id="chart" backgroundColor="0xFFFFFFFF" width="100%" height="100%"/>
</mx:Application>
```

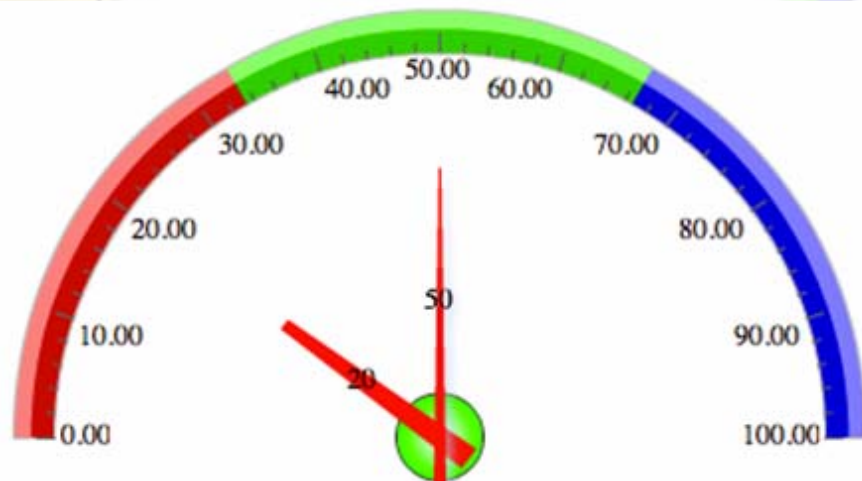
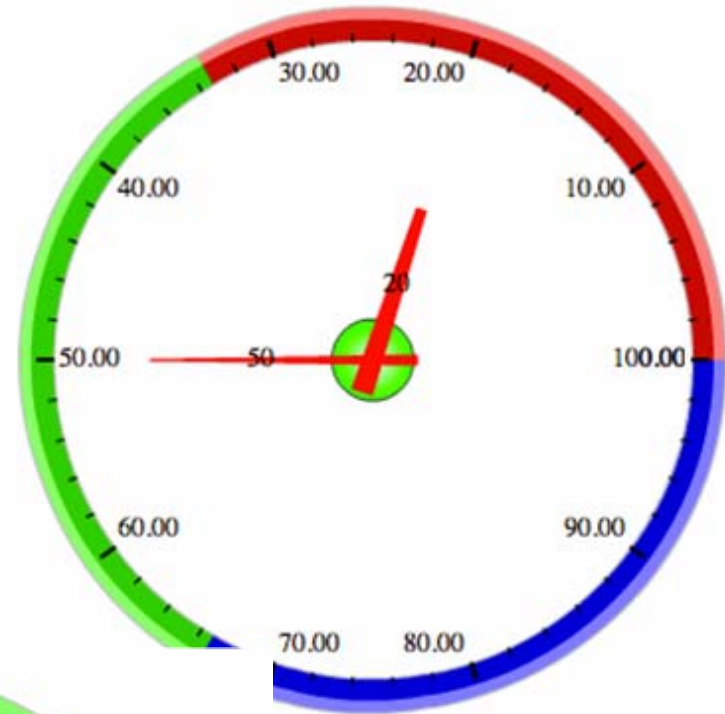


Twaver™

# BarChart

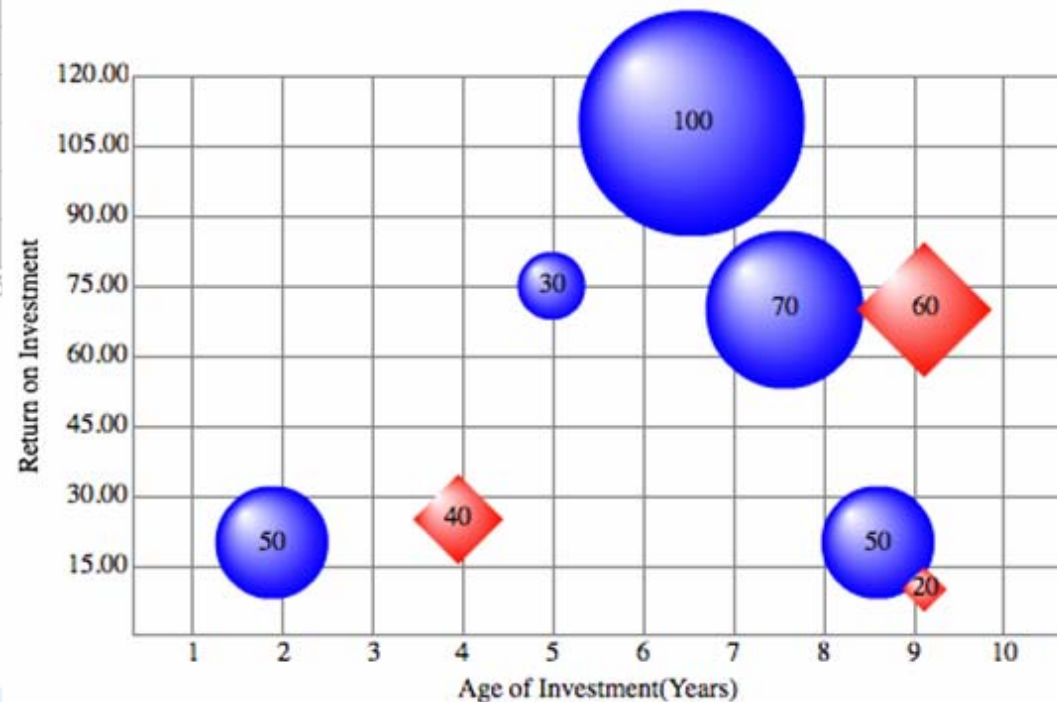
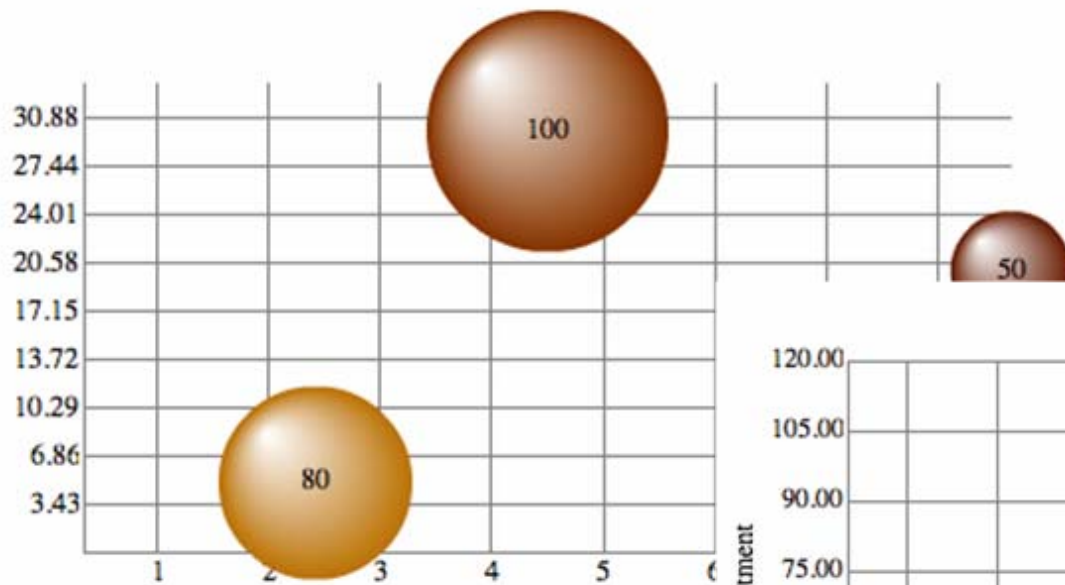


# DialChart



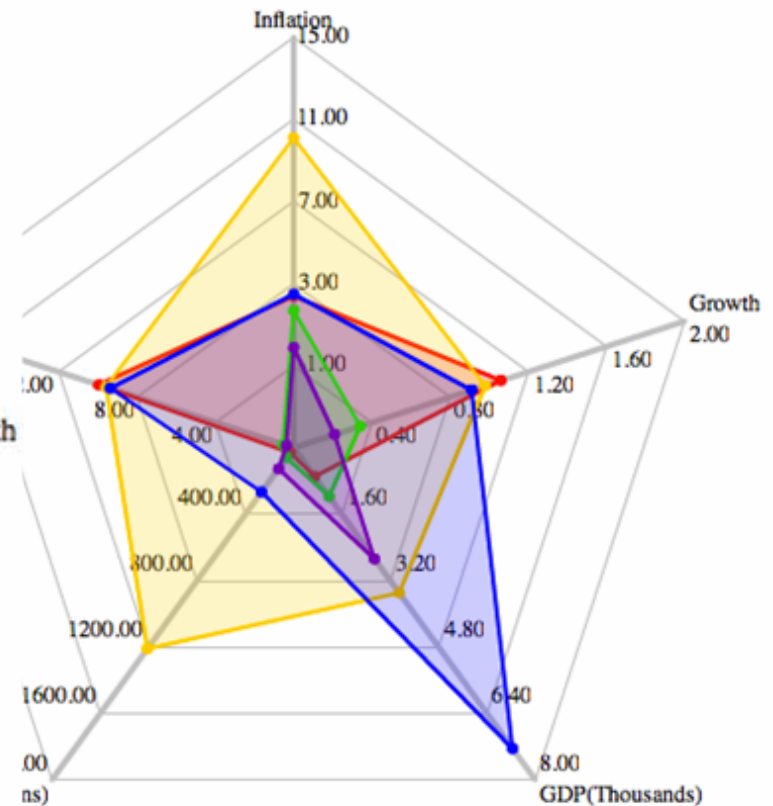
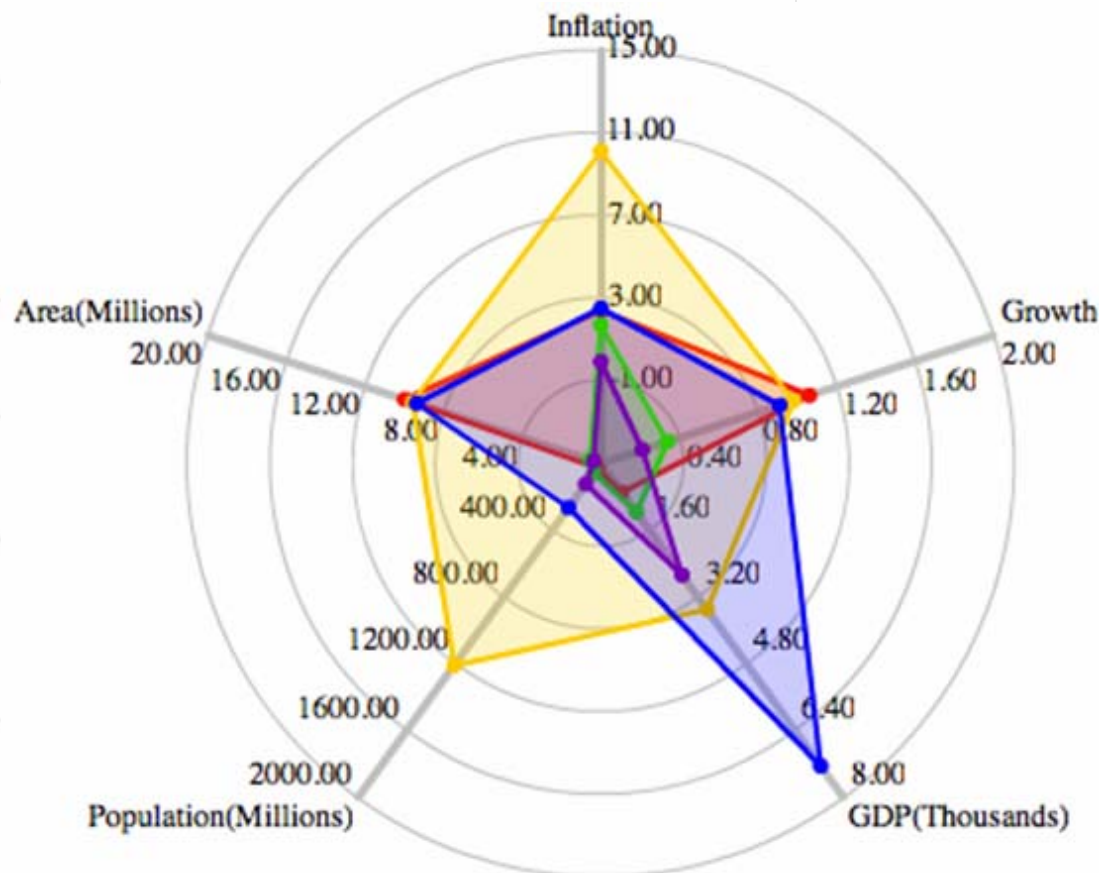


# BubbleChart





# RadarChart



TM  
Tmaver

# Using Alarm

- Alarm - describes the element or device operation status
- Alarm/AlarmSeverity
- AlarmBox
- AlarmState/AlarmStateStatistics
- Alarm display

TM  
waver

# AlarmSeverity

TWaver™

- *twaver. AlarmSeverity*
- **Alarm severity** - Reflect the urgency of the alarm contains name, nickName, value, color and other properties
- **Default alarm severities** - There are six alarm severities predefined.  
*CRITICAL, MAJOR, MINOR, WARNING, INDETERMINATE, CLEARED*
- **Custom alarm severities**  
`AlarmSeverity.clear();`  
`AlarmSeverity.register(1, "a", "a", 0xFF0000, "AAA");`

# Custom Alarm Severity Example

```
private function init():void{
    AlarmSeverity.clear();

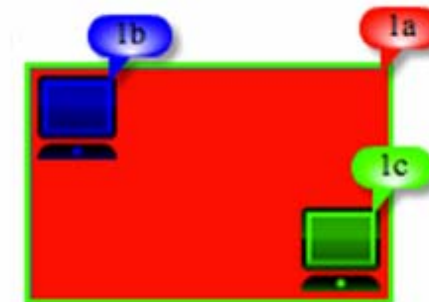
    var a:AlarmSeverity = AlarmSeverity.register(1, "a", "a", 0xFF0000, "AAA");
    var b:AlarmSeverity = AlarmSeverity.register(2, "b", "b", 0x0000FF, "BBB");
    var c:AlarmSeverity = AlarmSeverity.register(3, "c", "c", 0x00FF00, "CCC");

    var box:ElementBox = network.elementBox;

    var node1:Group = new Group();
    node1.expanded = true;
    var node2:Node = new Node();
    node2.setLocation(100, 100);
    var node3:Node = new Node();
    node3.setLocation(200, 150);
    node3.parent = node1;
    node2.parent = node1;
    box.add(node1);
    box.add(node2);
    box.add(node3);

    addAlarm(box.alarmBox,node1,a);
    addAlarm(box.alarmBox,node2,b);
    addAlarm(box.alarmBox,node3,c);
}

private function addAlarm(alarmBox:AlarmBox, element:Element, severity:AlarmSeverity):void{
    var alarm:Alarm=new Alarm(null, element.id, severity);
    alarmBox.add(alarm);
}
```



# Alarm Object

Twaver™

- twaver.Alarm - data element which describes equipment fault and network abnormality.
- Alarm properties:  
id, elementID, alarmSeverity, acked, cleared and other properties by method: alarm.setClient(key, value).



# AlarmBox

TM  
waver

- AlarmBox - Data container for managing alarm objects.
- Basic Operations  
add/remove/clear ...
- Quick finder and listeners



AlarmBox

# AlarmState

Twaver™

- twaver. AlarmState - describes the current alarm status of elements or devices
- Including all statistics information about new alarms, acknowledged alarms and propagated alarms.
- The specific alarm objects(twaver.alarm) are not included

# AlarmStateStatistics

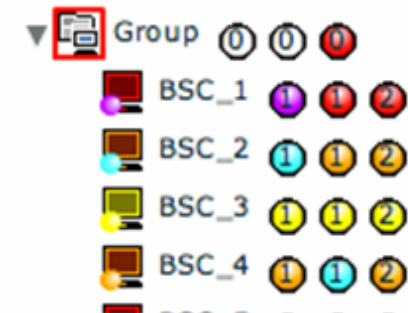
Twaver™

- AlarmStateStatistics - Counts the alarm states of all elements in the ElementBox.
- Includes all quantities of alarms for each alarm severity
- Includes the numbers of all new alarms or acknowledged alarms.
- Supports filter, can statistics for specify elements

# Alarm Display

- Alarm balloon
- Alarm color - fill or outline
- Alarm table
- Alarm statistics charts

Mapping ID	Alarm Severity	acked	cleared	R
5	Indeterminate	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 17 1
4	Warning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 17 1
3	Minor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 17 1
2	Major	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Fri Jun 17 1



Severity	New	Acked	Total ▲
Cleared	2	1	3
Minor	1	3	4
Critical	3	2	5
Major	5	1	6
Indeterminate	0	7	7
Warning	8	1	9
Total	19	15	34

# Flex Component Extension

Twaver™

- There are three ways to extend Flex components
  - CSS configuration
  - Skin customization
  - Component classes extension



# CSS Configuration

```
@namespace s "library://ns.adobe.com/flex/spark";
@namespace mx "library://ns.adobe.com/flex/mx";
```

defines namespace

```
global{
  chrome-color: #FFC000;
}
```

global style

```
#button1{
  fontSize: 22;
}
```

#id

mx button, with 'mx|Button' style

spark button, with 's|Button' style

```
.big{
  fontSize: 18;
}
```

with '.big' style

with '#button1' style

```
s|Button{
  fontSize: 14;
}
```

.styleName

namespace|class

```
mx|Button{
  fontSize: 11;
}
```

```
<mx:Button label="mx button, with 'mx|Button' style" />
<s:Button label="spark button, with 's|Button' style" />
<s:Button styleName="big" label="with '.big' style" />
<s:Button id="button1" label="with '#button1' style" />
```

TM  
Twaiver

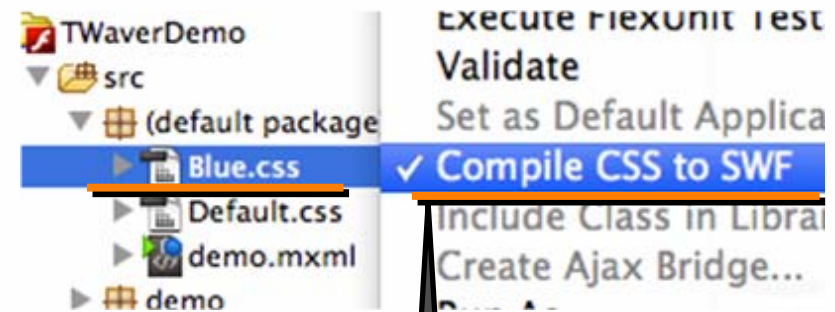
# Dynamic loading CSS

- **Loading CSS**

```
StyleManager.loadStyleDeclarations("Blue.swf",true);
```

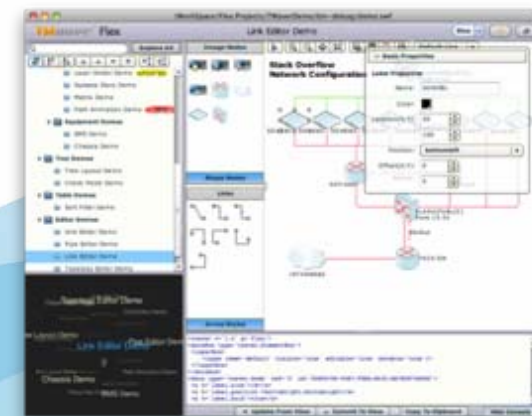
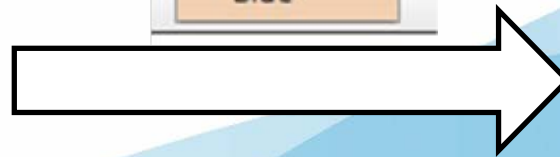
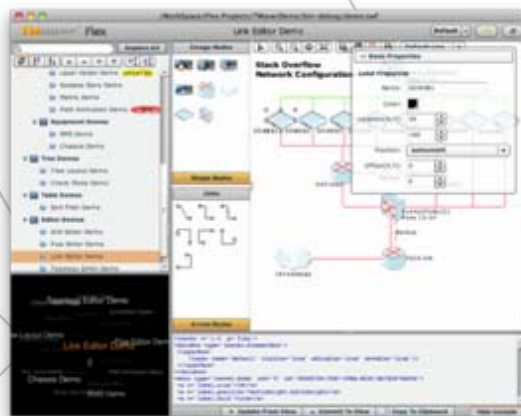
- **Unloading CSS**

```
StyleManager.unloadStyleDeclarations("Blue.swf", true);
```



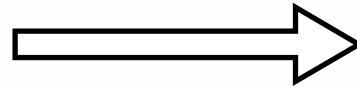
click the item "Compile CSS to SWF"

TWaver™



# Skin Customization

default TextInput



custom skin TextInput



Twaver™

Component

```
public class CustomTextInput extends TextInput{
    public function CustomTextInput(){
        super();
        //...
        this.setStyle("skinClass", CustomTextInputSkin);
    }
}
```

Skin

```
<s:SparkSkin ...>...
  <s:HGroup>
    <!-- icon -->
    <s:Image ... />
    <!-- text -->
    <s:RichEditableText ... />
  </s:HGroup>
</s:SparkSkin>
```

Associate with the custom skin by setting "skinClass" style

# Custom Skin Example

## CustomTextInput.as

```
package component{
import skin.CustomTextInputSkin;
import spark.components.TextInput;

[Style(name="icon", inherit="no", type="Class")]
[Style(name="radius", inherit="true", type="Number")]

public class CustomTextInput extends TextInput{
    [Embed(source="/images/search.png")]
    private const defaultIcon:Class;

    public function CustomTextInput(){
        super();
        this.setStyle('icon', defaultIcon);
        this.setStyle("skinClass", CustomTextInputSkin);
    }
}
```

## CustomTextInputSkin.mxml

```
<s:SparkSkin ...>
...
<s:HGroup gap="0" height="100%" paddingLeft="4"
paddingRight="4">
    <!-- icon -->
    <s:Image id="icon" includeIn="normal" x="0" y="0"
source="{hostComponent.getStyle('icon')}}"
verticalAlign="middle" height="100%"/>
    <!-- text -->
    <s:RichEditableText id="textDisplay"
        verticalAlign="middle"
        widthInChars="10"
        left="1" right="1" top="1" bottom="1"
height="100%"/>
    </s:HGroup>
...
</s:SparkSkin>
```

# Component Classes Extension

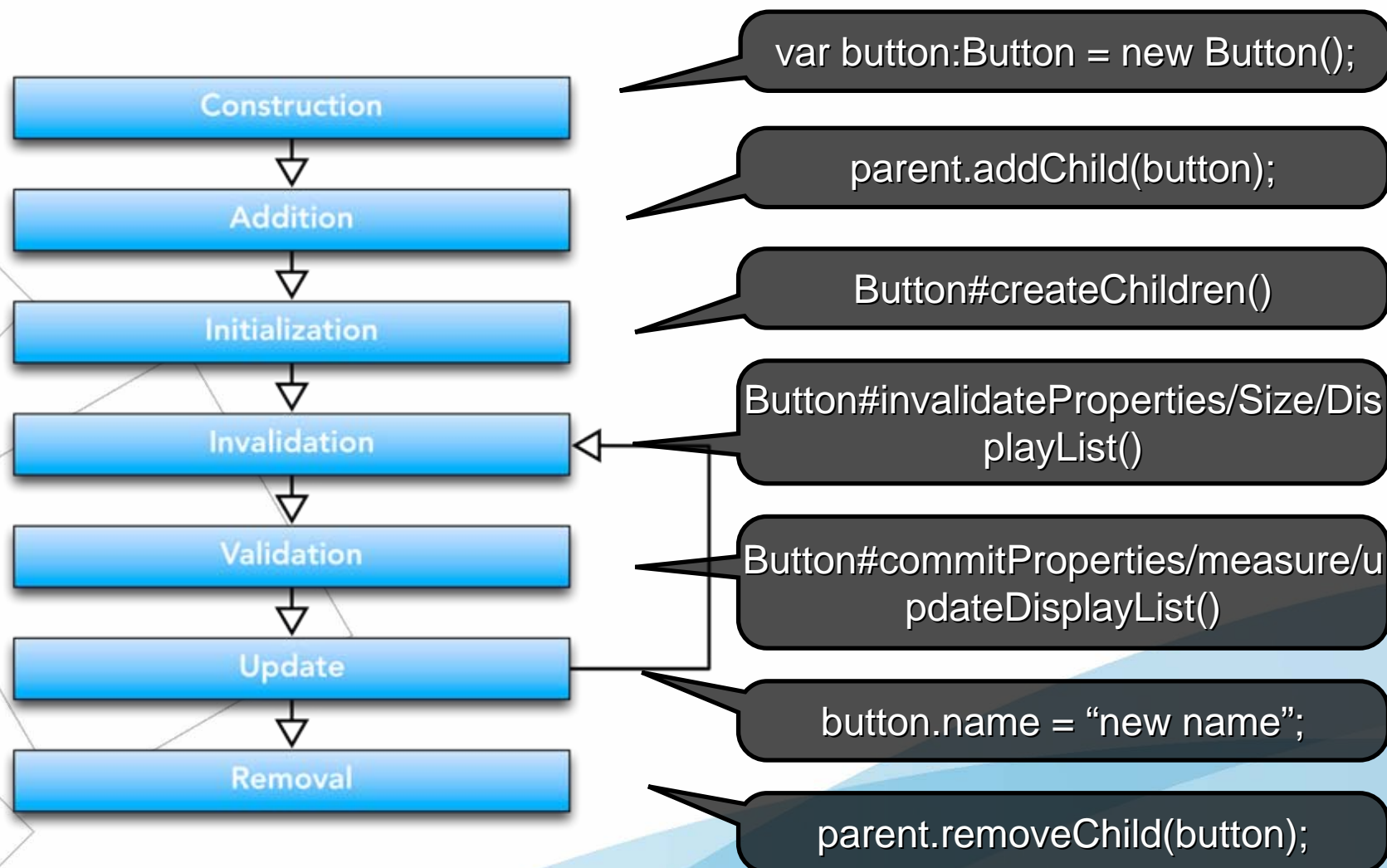
TM  
waver

- All Flex components based on UIComponent
- Custom Flex components, usually need to override methods of UIComponent, such as: *createChildren()/commitProperties()/measure()/updateDisplayList() etc.*
- If you want extend components deeply, you need to know more about **Flex component's life cycle**



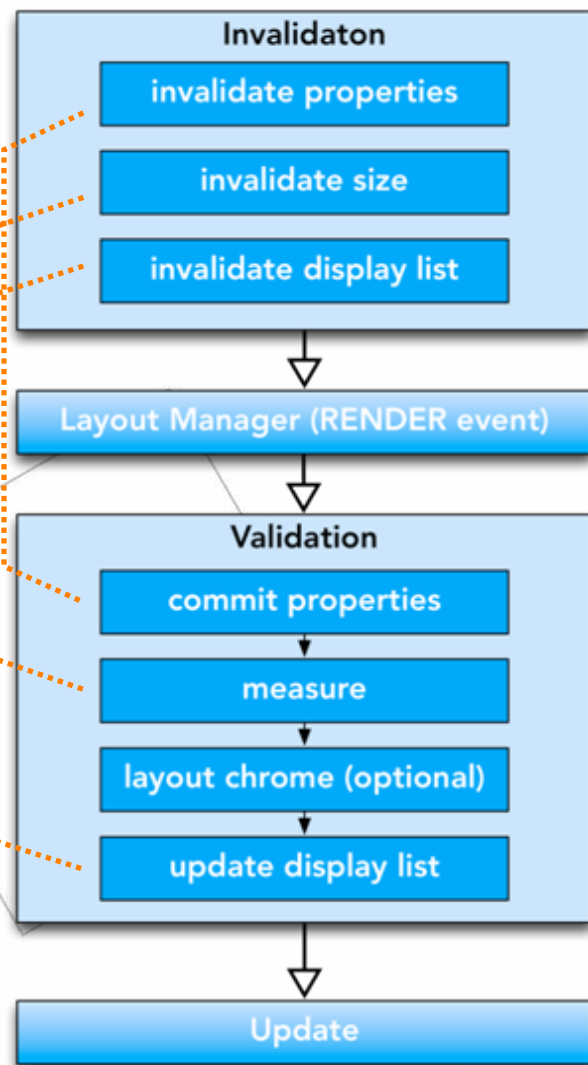
# Flex Component's Life Cycle

TWaver™



# Validate & Invalidate

TWaver™



These three functions are respectively referred to properties, size or display list was changed

The invalidate components will not be redraw until the next RENDER event

The validation also includes three methods relative to the Invalidation methods, plus layoutChrome

Update components in the Validate-Invalidate cycle

# Thank you

- Home - ServaSoftware.com
- Email - [tw-service@servasoft.com](mailto:tw-service@servasoft.com)