CrossUI Cookbook
Getting Started Guide

Preface 7
Chapter 1. Preparation 8
1.1. Download the package 8
1.2. The package folder 8
1.3. Glance at examples and API 9
Chapter 2. Hello World 10
2.1. The first application 10
2.2. Render onto a html node 11
2.3. Do it in CrossUI RAD Tools 13
2.4. Application loading process 19
2.5. Code Editor 21
2.5.1. Highlight code from Outline window 22
2.5.2. Code Folding 22
2.5.3. Code Intellisense 23
2.5.3.1. When context doesn’t recognize the input string 23
2.5.3.2. Type dot after a recognizable variable 25
2.5.3.3. When use dbclick 25
2.5.4. Find the object definition code 26
2.5.5. Generate event code automatically 26
Chapter 3. Controls Facebook 27
3.1. Script testing environment 27
3.2. “Hello world” in env.html 29
3.3. Control creation and runtime update 30
3.4. Button related 31
3.4.1. onClick event 31
3.4.2. Boolean Controls 32
3.4.3. Link Control 32
3.5. Label related 33
3.6. Input related 34
3.6.1. setValue/setValue/getUIValue/setUIValue 34
3.6.2. Dirty Mark 34
3.6.3. Password Input 35
3.6.4. Multi-lines 35
3.6.5. Input validation 36
3.6.5.1. valueFormat property 36
3.6.5.2. beforeFormatCheck event 36
3.6.6. Dynamic input validation 37

3.6.7. Error Mark..37
3.6.7.1. Default Error Mark..37
3.6.7.2. Validation Tips 37
3.6.7.3. Binding Validation..38
3.6.7.4. Custom Error Mark 38
3.6.8. Mask Input 39
3.6.9. xui.UI.ComboInput 40
3.6.9.1. Pop list for selection 40
3.6.9.2. combobox, listbox and helpinput 40
3.6.9.3. Date Piker..41
3.6.9.4. Time Picker 42
3.6.9.5. Color Picker 42
3.6.9.6. File Picker 43
3.6.9.7. Getter..43
3.6.9.8. Custom Pop Window..44
3.6.9.9. Command Buttons..45
3.6.10. RichEditor 45
3.7. List related 46
3.7.1. A Simple one..46
3.7.2. A little bit complicated..47
3.7.3. RadioBox 48
3.7.4. IconList and Gallery..48
3.7.5. Item selection..49
3.7.6. Container related..49
3.7.7. Pane and Panel 50
3.7.8. Block..51
3.8. Dialog related..51
3.8.1. Normal state..51
3.8.2. Min and Max status..53
3.8.3. Modal Mode..53
3.9. Layout Control..54
3.10. Multi-pages Controls..55
3.10.1. noPanel property 56
3.10.2. ButtonViews types 57
3.10.3. Page selection..57
3.10.4. Pages 58
3.10.4.1. Close and options Button 58
3.10.4.2. Add/Remove Pages 60
3.10.5. Dynamic content loading 61
3.10.5.1. onIniPanelView..61
3.10.5.2. beforeUIValueSet/afterUIValueSet 61
3.11. Menus and toolbars..62
3.11.1. Pop Menu 62
3.11.2. MenuBar..63

3.11.3. Toolbars 64
3.12. TreeBar and TreeView 65
3.12.1. Three selection mode 65
3.12.1.1. No-selection 65
3.12.1.2. Single-selection 66
3.12.1.3. Multi-selection 66
3.12.2. Group Item 67
3.12.3. Expand all nodes by default 67
3.12.4. Mutex Expand 68
3.12.5. Dynamic Destruction 69
3.12.6. Dynamically loading 69
3.13. TreeGrid 70
3.13.1. Header and Rows 70
3.13.1.1. Sets standard format 71
3.13.1.2. Sets simplified format 72
3.13.2. getHeader 72
3.13.3. getRows 73
3.13.4. Active Modes 74
3.13.4.1. non-active appearance 74
3.13.4.2. row-active appearance 75
3.13.4.3. cell-active appearance 75
3.13.5. Selection Mode 76
3.13.5.1. Non-selection 76
3.13.5.2. Single row selection 77
3.13.5.3. Multi-row selection 77
3.13.5.4. Single cell selection 78
3.13.5.5. Multi-cells selection 78
3.13.6. The Tree Grid 79
3.13.7. Column config 80
3.13.7.1. The first column 80
3.13.7.2. Column width 81
3.13.7.3. Drag&Drop to modify column width 81
3.13.7.4. Drag&Drop to modify column position 82
3.13.7.5. Default Sorting 82
3.13.7.6. Custom Sorting 83
3.13.7.7. Hide columns 83
3.13.7.8. Setting Cell Types in column header 84
3.13.7.9. column header style 84
3.13.7.10. column header icon 85
3.13.7.11. Update column header dynamically 86
3.13.8. Row config 86
3.13.8.1. Row height 86
3.13.8.2. Drag&Drop to modify row height 87
3.13.8.3. Setting cell type in row 87

3.13.8.4. Row style 88
3.13.8.5. Row numbers 88
3.13.8.6. Custom row numbers 89
3.13.8.7. Alternate Row Colors 90
3.13.8.8. Group 90
3.13.8.9. Preview and Summary region 91
3.13.8.10. Update row dynamically 92
3.13.9. Cell config 93
3.13.9.1. Cell types 93
3.13.9.2. Cell style 94
3.13.9.3. Update cell dynamically 95
3.13.10. Editable 95
3.13.10.1. Editable TreeGrid 96
3.13.10.2. Editable column 96
3.13.10.3. Editable row 97
3.13.10.4. Editable cell 97
3.13.10.5. The Editor 98
3.13.10.6. Custom the editor 99
3.13.11. Add/Remove rows 100
3.14. Other standard controls 101
3.14.1. ProgressBar 101
3.14.2. Slider 102
3.14.3. Image 103
3.14.4. PageBar 103
Chapter 4. Data exchanging(Ajax) 104
4.1. Fiddler 105
4.2. To get the contents of the file 105
4.3. Synchronous data exchange 105
4.4. Cross-domain 106
4.4.1. To monitor SAjax 106
4.4.2. To monitor IAjax 107
4.5. File Upload 108
4.5.1. Selecting upload file with ComboInput 108
4.5.2. Upload by IAjax 109
4.6. A request wrapper for real application 109
4.7. XML Data 110
4.8. An overall example 110
Chapter 5. Distributed UI 112
5.1. Shows dialog from a remote file 112
5.2. xui.Com and xui.ComFactory 113
5.2.1. xui.ComFactory config 113
5.2.2. xui.Com.Load 114
5.2.3. newCom and getCom 115
5.2.4. xui.UI.Tag 116

5.2.5. Destroy com 116
5.2.6. If com exists in memory....116
Chapter 6. Some fundamental things....117
6.1. Pop-up window 117
6.1.1. alert window....117
6.1.2. confirm window 117
6.1.3. prompt window 118
6.1.4. pop window....118
6.2. Asynchronous execution....118
6.2.1. asyRun....118
6.2.2. resetRun 119
6.3. Skin switcher....119
6.3.1. Switch skin for whole application....119
6.3.2. Change skin for a single control....119
6.4. Locale switcher 120
6.5. DOM Manipulation....120
6.5.1. Node generation and insertion....121
6.5.2. Attributes and CSS....121
6.5.3. className 122
6.5.4. Dom events 122
6.5.5. Node Drag&Drop....123
6.5.5.1. Drag&Drop profile....124
6.5.5.2. Events in Drag&Drop....125
6.6. xui.Template....126
6.6.1. example 1 126
6.6.2. example 2 127
6.6.3. A SButton based on xui.Template....128
6.7. About Debugging....129
6.7.1. The code package for debugging 129
6.7.2. Debugging Tools 129
6.7.3. Monitor Tools....130
Chapter 7. Some typical issues 131
7.1. Layout 131
7.1.1. Docking....131
7.1.2. xui.UI.Layout....131
7.1.3. Relative Layout....132
7.2. UI Control’s Drag&Drop 134
7.2.1. Drag&Drop control among containers....134
7.2.2. List sorting 1 134
7.2.3. List sorting 2 135
7.3. Form....136
7.3.1. Form 1 136
7.3.2. DataBinder 137
7.4. Custom UI Styles 139

7.4.1. Custom one instance only - 1 139
7.4.2. Custom one instance only - 2 140
7.4.3. Custom one instance only - 3 140
7.4.4. Custom one instance only - 4 140
7.4.5. Custom one instance only - 5 141
7.4.6. Custom one instance only - 6 141
7.4.7. Custom style for an UI Class 142
7.4.8. Custom style for all UI Class - skin 142
7.4.8.1. First: Copy....142
7.4.8.2. Second: Little by little, modify pictures and CSS 143
The end 144

Preface

CrossUI is a Cross-Browser JavaScript framework with cutting-edge functionality for rich web application.

CrossUI RAD Tools enables developers to rapidly develop and package the exactly same code and UI into Web Apps, Native Desktop Apps for Windows, OS X, Linux and UNIX on 32-bit and 64-bit architectures as well as Mobile Apps for iPhone, iPad, Windows Phone, webOS, BlackBerry, and Android devices. With this powerful RAD Tools, developers can build cross-platform applications just like what they do in VB or Delphi.

Develop Once, Deploy Anywhere!

Features & Resource :

1. Rich client-side API, works with any backend or static HTML pages.
2. Web services (JSON/XML/SOAP) can be directly bound.
3. More than 40 common components, including Tabs, Dialog, TreeGrid, TimeLine and many other web GUI components.
4. Wide cross-browser compatibility, IE6+, firefox1.5+, opera9+, safari3+ and Google Chrome.
5. Full API Documentation with tons of samples.
6. Ever Increasing Code Snippets.
7. PHP/C #/JAVA Back-end service codes are available.
8. CrossUI is Open Source under LGPL3 license;
9. CrossUI RAD (commercial license) can reduce development time significantly.

This guide focuses on CrossUI Framework itself, and contains some info about CrossUI RAD Tools . In this guide book, all the examples will be demoed in browsers. But those examples are cross-platform, you can package them with CrossUI RAD Tools, and deploy them anywhere.

If you have any good suggestions, you can contact me at linb[at]crossui.com.

Go to http://www.crossui.com/Forum for the more information.

Chapter 1. Preparation

First of all, note that all instances of this tutorial are based on CrossUI version 1.0. Therefore, our first task is to download the 1.0 release package, and to establish the local environment.

1.1. Download the package

CrossUI framework zip package can be downloaded from
http://www.crossui.com/download.html or
http://code.google.com/p/crossui/downloads/list.

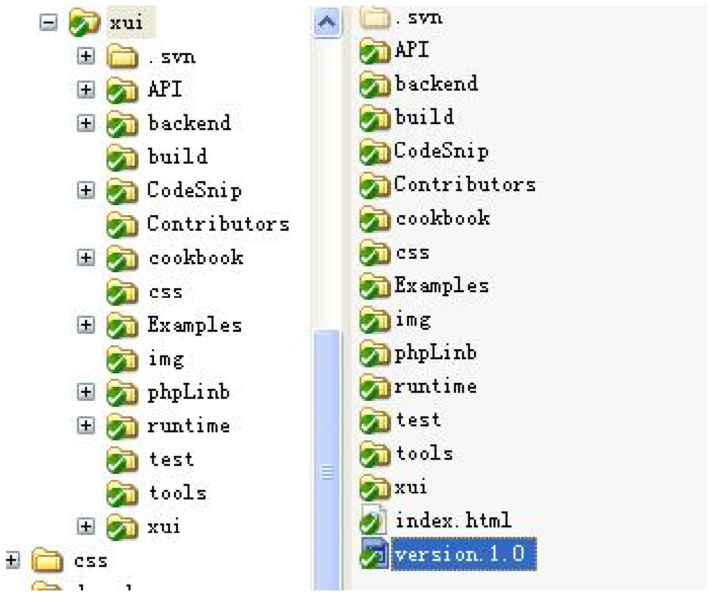
It's the latest stable version, but not the latest code. I suggest you get the latest code from our SVN.

For those who are not familiar with SVN, should learn how to use SVN first. After all, a lot of open-source projects use SVN to manage code. SVN requires a client program to connect to what is called a "repository" where the files are stored. On commonly used SVN client is called TortoiseSVN, which is freely available. Other clients exist, but TortoiseSVN is recommended due to its simplicity of use.

Version 1.0 repository URL: http://crossui.googlecode.com/svn/trunk/xui1.0/ **.**

1.2. The package folder

If you downloaded package from google group, extract the package to a local folder. If you fetch the code from SVN, does not need to extract.

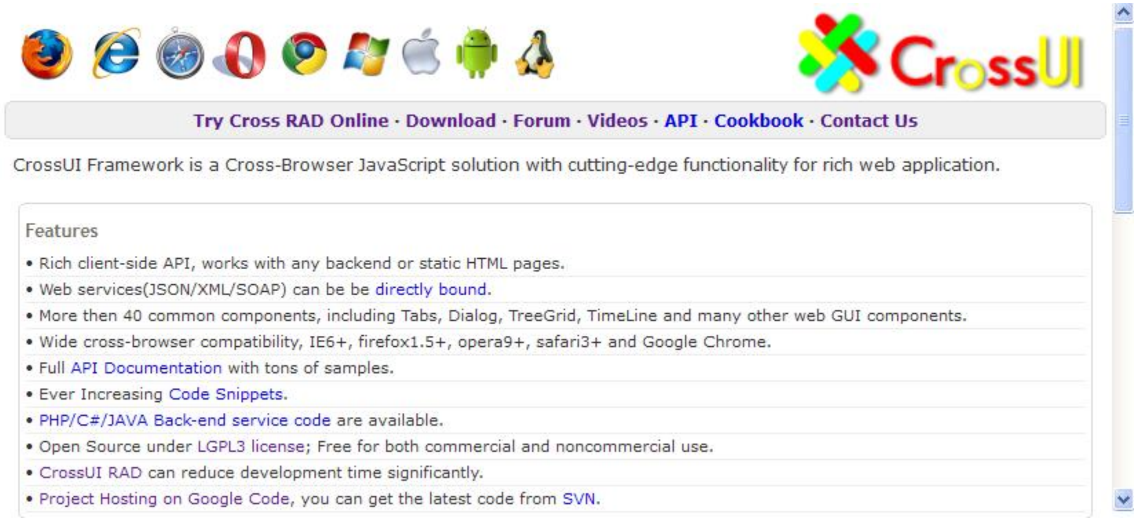


The contents of the package folder

By default, most of the examples in the package can be run in local disk directly, but a small number of examples need php background environment, or mysql database. In this case, you need to prepare Apache server (version 2 and above), php (version 5 and above) and mysql (version 5 and above). And, copy the package to apache web directory.

1.3. Glance at examples and API

If your Apach/php environment works well, after you copied the package folder to Apache's web directory (this tutorial assumes that your root directory is http://localhost/CrossUI/), you should be able to open the page with your browser: http://localhost/CrossUI/.

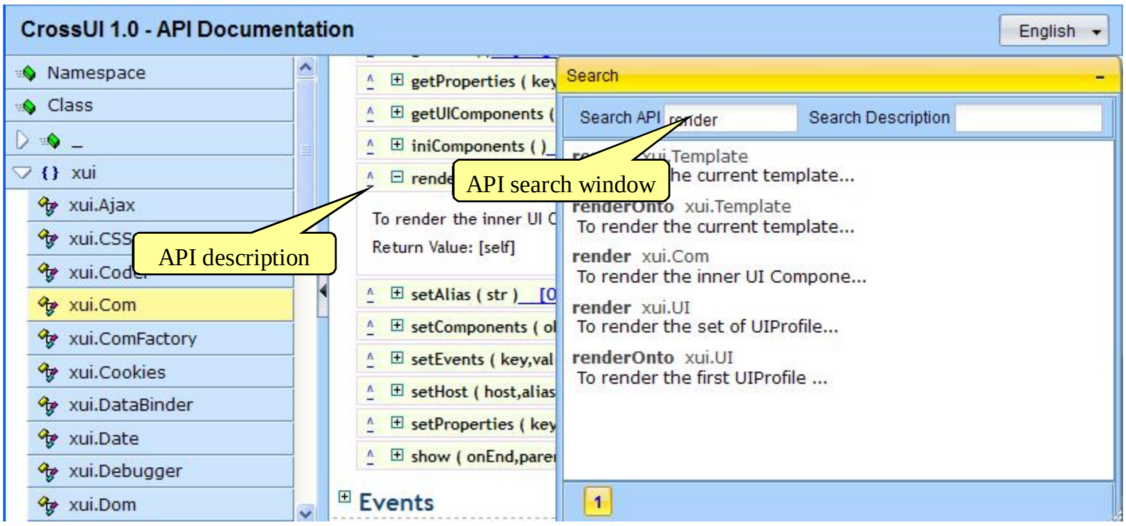

CrossUI
Try Cross RAD Online · Download · Forum · Videos · API · Cookbook · Contact Us

CrossUI Framework is a Cross-Browser JavaScript solution with cutting-edge functionality for rich web application.

Features

- Rich client-side API, works with any backend or static HTML pages.
- Web services(JSON/XML/SOAP) can be be directly bound.
- More then 40 common components, including Tabs, Dialog, TreeGrid, TimeLine and many other web GUI components.
- Wide cross-browser compatibility, IE6+, firefox1.5+, opera9+, safari3+ and Google Chrome.
- Full API Documentation with tons of samples.
- Ever Increasing Code Snippets.
- PHP/C#/JAVA Back-end service code are available.
- Open Source under LGPL3 license; Free for both commercial and noncommercial use.
- CrossUI RAD can reduce development time significantly.
- Project Hosting on Google Code, you can get the latest code from SVN.

You can browse http://localhost/CrossUI/Examples/ for examples, and http://localhost/CrossUI/API/ for API Documentation.



To render the inner UI C
Return Value: [self]

Search API: render

render xui.Template
To render the current template...
renderOnto xui.Template
To render the current template...
render xui.Com
To render the inner UI Compone...
render xui.UI
To render the set of UIProfile...
renderOnto xui.UI
To render the first UIProfile ...

A simple glance at API is strongly recommended. Learn about how to search a specific API,

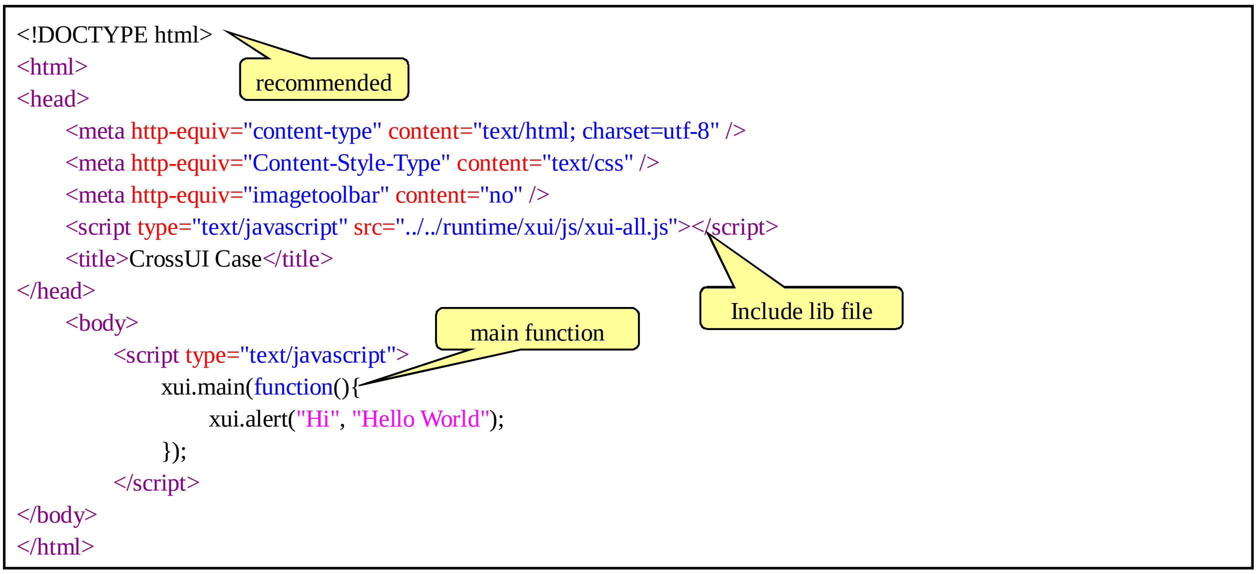
and how to run the inner code snippet.

Chapter 2. Hello World

2.1. The first application

As many would expect or not expect, the first example is "Hello World".

Now, create a new folder “mycases” in the package folder (again, this tutorial assumes that your root directory is http://localhost/CrossUI/), add a sub folder “chapter1” in it, and create a file named “helloworld.html” in “chapter1”. Enter the following code:



```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
    <meta http-equiv="imagetoolbar" content="no" />
    <script type="text/javascript" src="../../runtime/xui/js/xui-all.js"></script>
    <title>CrossUI Case</title>
</head>
    <body>
        <script type="text/javascript">
            xui.main(function(){
                xui.alert("Hi", "Hello World");
            });
        </script>
</body>
</html>
```

recommended
Include lib file
main function

cookbook/chapter1/helloworld.html

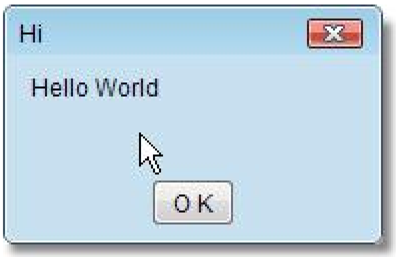
You can find all the source code for each example in this tutorial in the zip package. The CrossUI Cookbook Zip Package with examples can be downloaded from

http://www.crossui.com/download.html , or

http://code.google.com/p/crossui/downloads/list .

You can double-click the helloworld.htm1 to open the file.

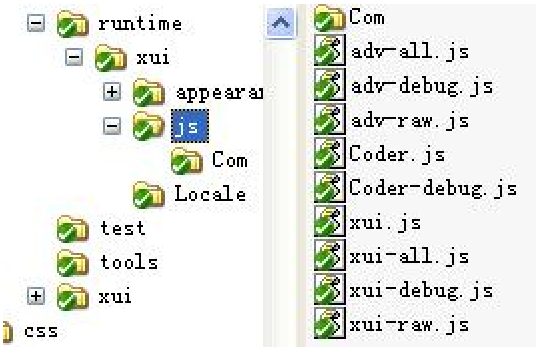
Or open URL http://localhost/CrossUI/cookbook/chapter1/helloworld.html in your browser (firefox or chrome is recommended here). And you can see the following result:



Hi

Hello World

File “xui-all.js” contains all standard controls (Button, Input, CombInput, Tabs, TreeBar, and TreeGrid etc.). This file can be found in “runtime/js” folder.



2.2. Render onto a html node

“Just replace DIVs with your controls.” A project manager said. "Our web page engineer is responsible to design an html file including a DIV with a unique ID, and JavaScript engineer is responsible to build an advanced UI control, and replace that DIV."

The following example in file chapter1/renderonto.html:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
    <meta http-equiv="imagetoolbar" content="no" />
    <script type="text/javascript" src="../../runtime/xui/js/xui-debug.js"></script>
    <title>CrossUI Case</title>
</head>
    <body>
        <div id="grid" style="position:absolute;left:100px;top:100px;width:300px;height:200px;"></div>
        <script type="text/javascript">
            xui.main(function(){
                var grid = new xui.UI.TreeGrid();
                grid.setGridHandlerCaption('grid')
                    .setRowNumbered(true)
                    .setHeader(['col 1','col 2','col 3'])
                    .setRows([
                        ['a1','a2','a3'],
                        ['b1','b2','b3'],
                        ['c1','c2','c3'],
                        ['d1','d2','d3'],
                        ['e1','e2','e3'],
                        ['f1','f2','f3']
                    ]);
                grid.renderOnto('grid');
            });
        </script>
</body>
</html>
```

The DIV with id “grid”

Sets grid caption

Show line number

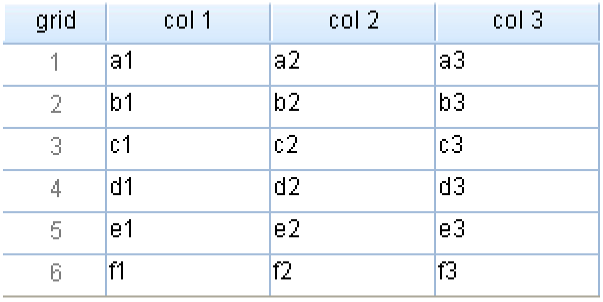
Sets columns

Sets rows

Render onto that DIV

cookbook/chapter1/renderonto.html

The result is:



grid	col 1	col 2	col 3
1	a1	a2	a3
2	b1	b2	b3
3	c1	c2	c3
4	d1	d2	d3
5	e1	e2	e3
6	f1	f2	f3

There are two ways to get the same result; codes were in renderonto2.html and renderonto3.html.

renderonto2.html :

```
xui.main(function(){
    (new xui.UI.TreeGrid({
        gridHandlerCaption:'grid',
        rowNumbered:true,
        header:['col 1','col 2','col 3'],
        rows:[['a1','a2','a3'],['b1','b2','b3'],['c1','c2','c3'],
                ['d1','d2','d3'],['e1','e2','e3'],['f1','f2','f3']]
    })).renderOnto('grid');
});
```

Gives properties directly

cookbook/chapter1/renderonto2.html

renderonto3.html :

```
xui.main(function(){
    xui.create('TreeGrid',{
        gridHandlerCaption:'grid',
        rowNumbered:true,
        header:['col 1','col 2','col 3'],
        rows:[['a1','a2','a3'],['b1','b2','b3'],['c1','c2','c3'],
                ['d1','d2','d3'],['e1','e2','e3'],['f1','f2','f3']]
    }).renderOnto('grid');
});
```

Using xui.create

cookbook/chapter1/renderonto3.html

These three approaches generated the same result. You can use any of those in your project according to your habits. But the first approach (using new and setXX) is recommended.

2.3.Do it in CrossUI RAD Tools

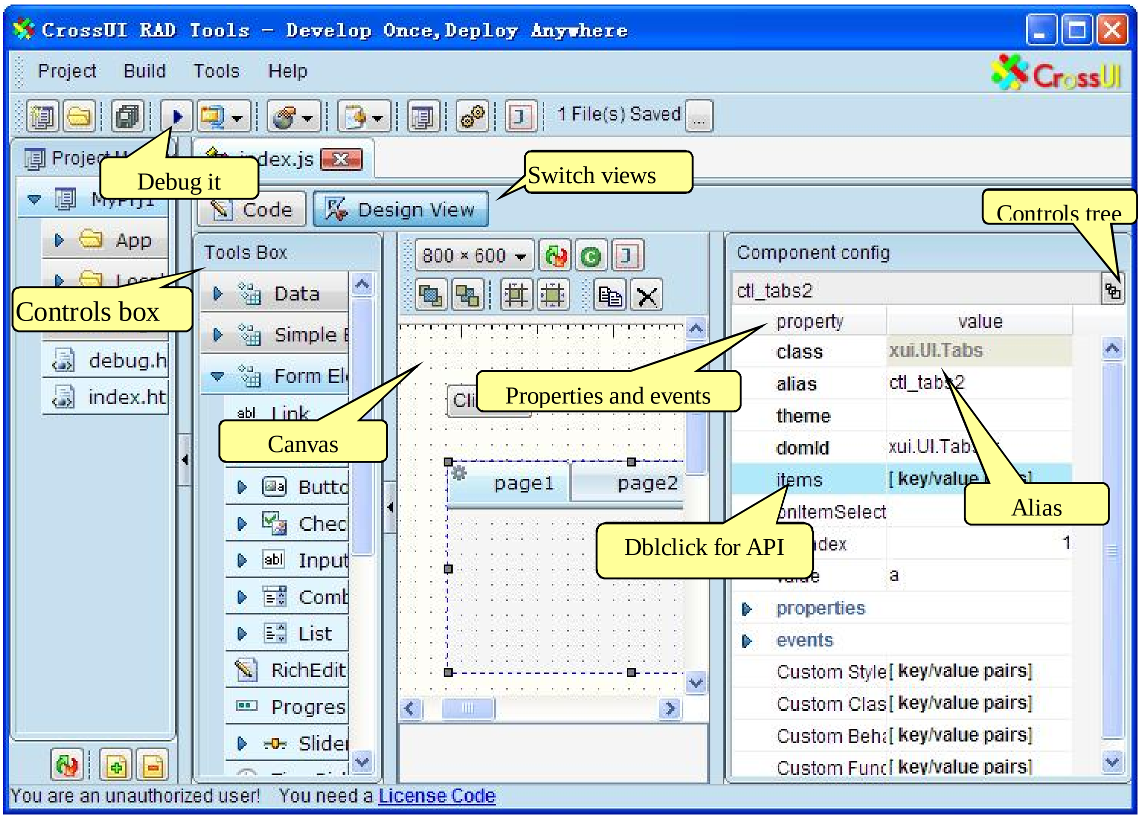
CrossUI RAD can reduce development time significantly, especially on UI layout.

There are two types of designer in CrossUI: online version and desktop version. Desktop version is integrated with many advanced features: document management, package, deployment, and so on.

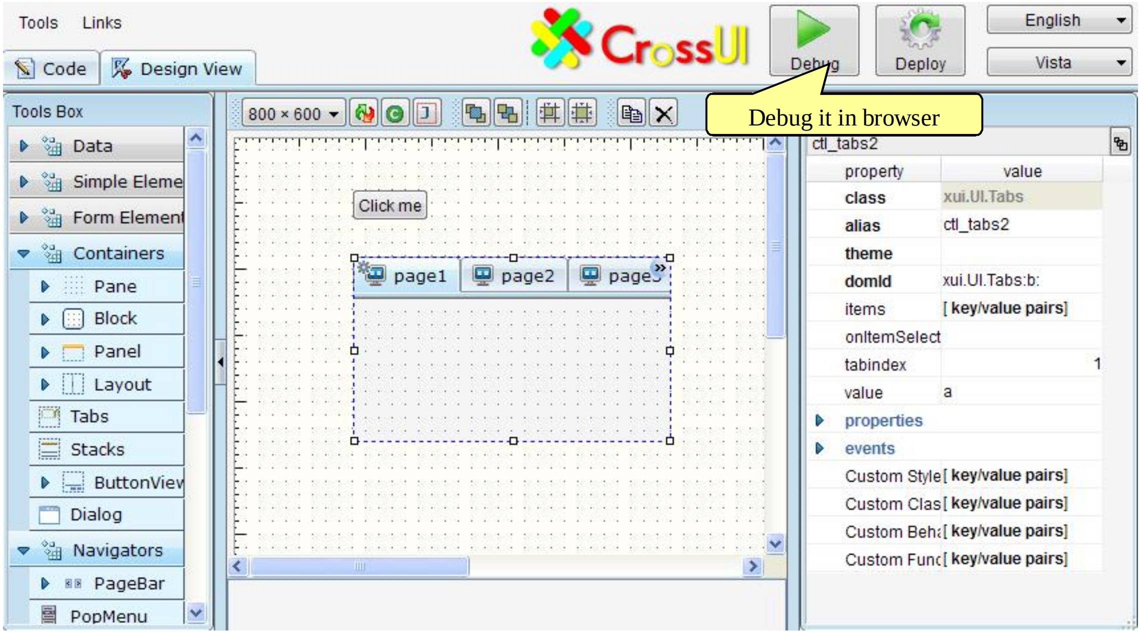
In order to do the following exercises, you need to download CrossUI RAD Tools desktop version from http://www.crossui.com/download.html.

If you don’t want to download that, you can go to http://www.crossui.com/RAD/Builder.html, and do the following exercises online.

RAD Tools desktop:

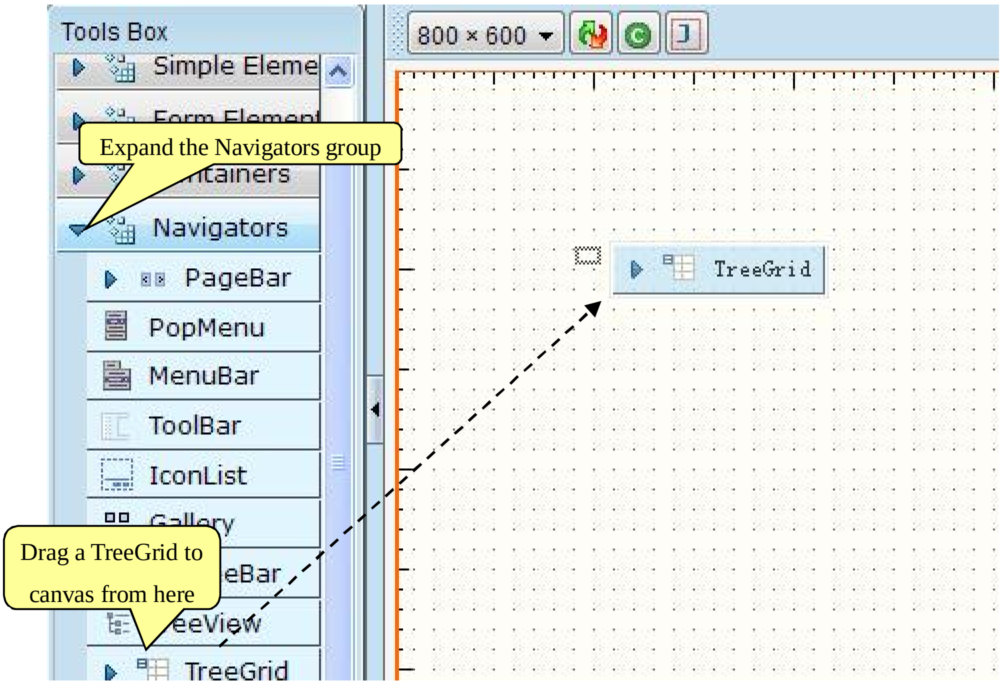


RAD Tools online:



Now, we are trying to create the previous section’s grid example in RAD Tools Designer.

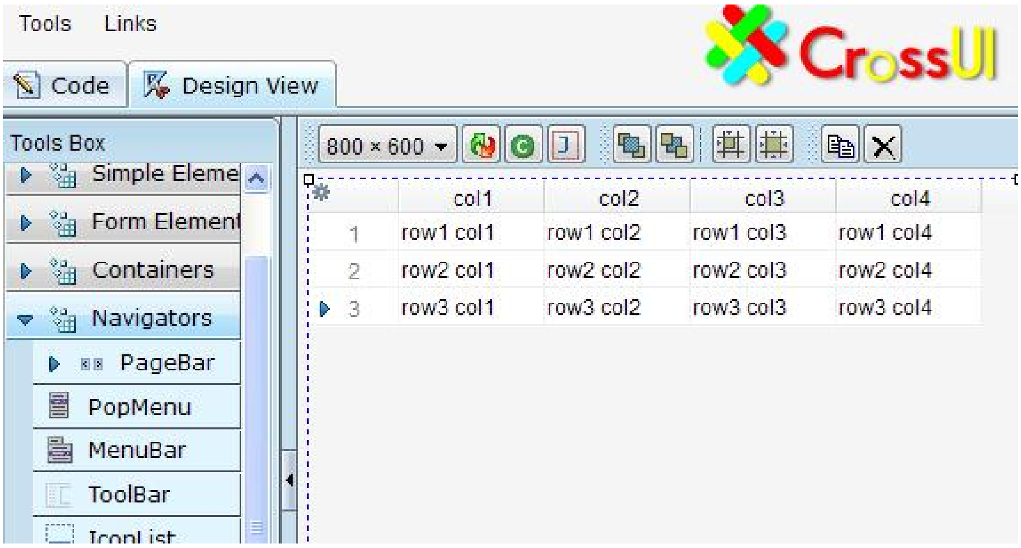
1. Open the navigators group in “Tools Box”, and drag the “TreeGrid” control to the Canvas area.



Expand the Navigators group

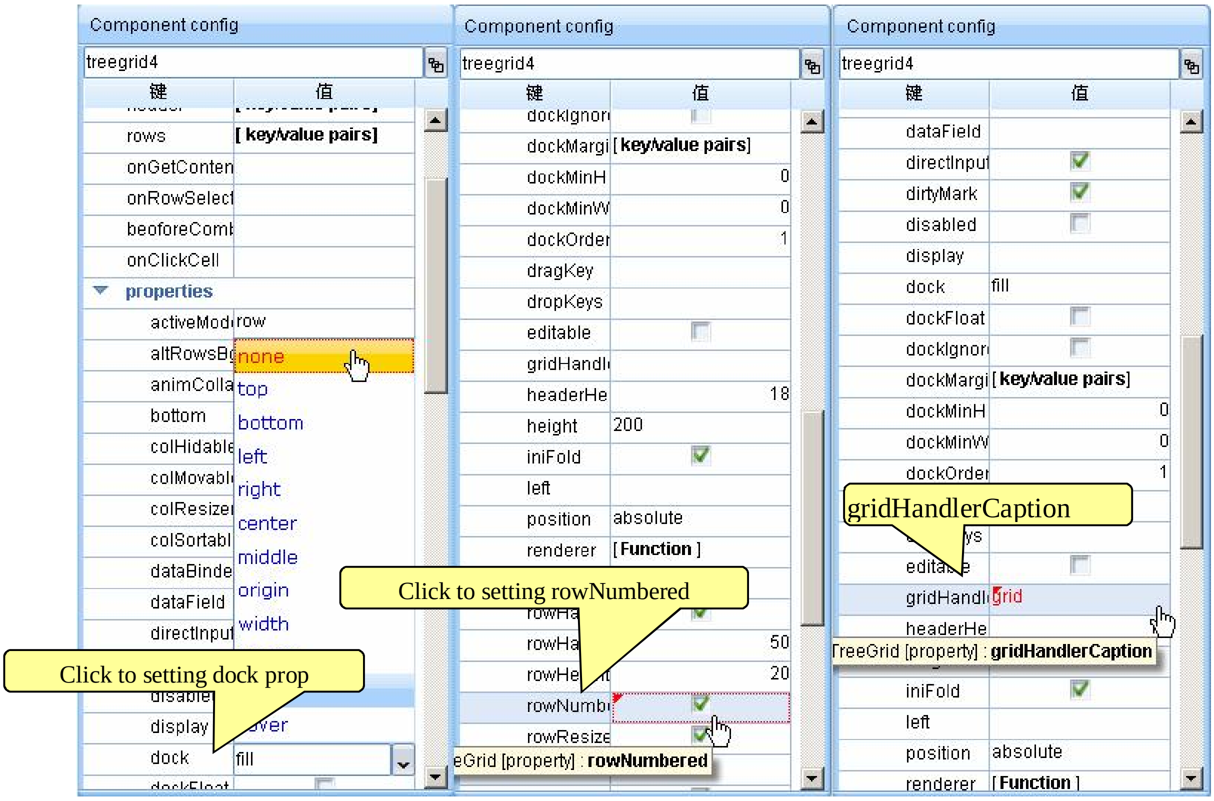
Drag a TreeGrid to canvas from here

2. Click to select the "treegrid" (It's selected by default)



	col1	col2	col3	col4
1	row1 col1	row1 col2	row1 col3	row1 col4
2	row2 col1	row2 col2	row2 col3	row2 col4
3	row3 col1	row3 col2	row3 col3	row3 col4

3. Sets this grid's properties according to the following picture.
   - Sets dock to 'none';
   - Sets rowNumbered to false;
   - Sets gridHandlerCapion to 'grid'.



Component config

treegrid4

键	值
rows	[key/value pairs]
onGetConten	
onRowSelect	
beoforeComb	
onClickCell	
properties	
activeModi	row
altRowsBg	none
animColla	top
bottom	bottom
colHidable	left
colMovabl	right
colResizer	center
colSortabl	middle
dataBinde	origin
dataField	width
directInput	
disable	
display	over
dock	fill

Click to setting dock prop

Component config

treegrid4

键	值
dockIgnore	□
dockMargi	[key/value pairs]
dockMinH	0
dockMinW	0
dockOrder	1
dragKey	
dropKeys	
editable	□
gridHandl	
headerHe	18
height	200
iniFold	☑
left	
position	absolute
renderer	[Function]
rowHa	☑
rowHa	50
rowHe	20
rowNumb	☑
rowResize	☑

Click to setting rowNumbered

eGrid [property] : rowNumbered

Component config

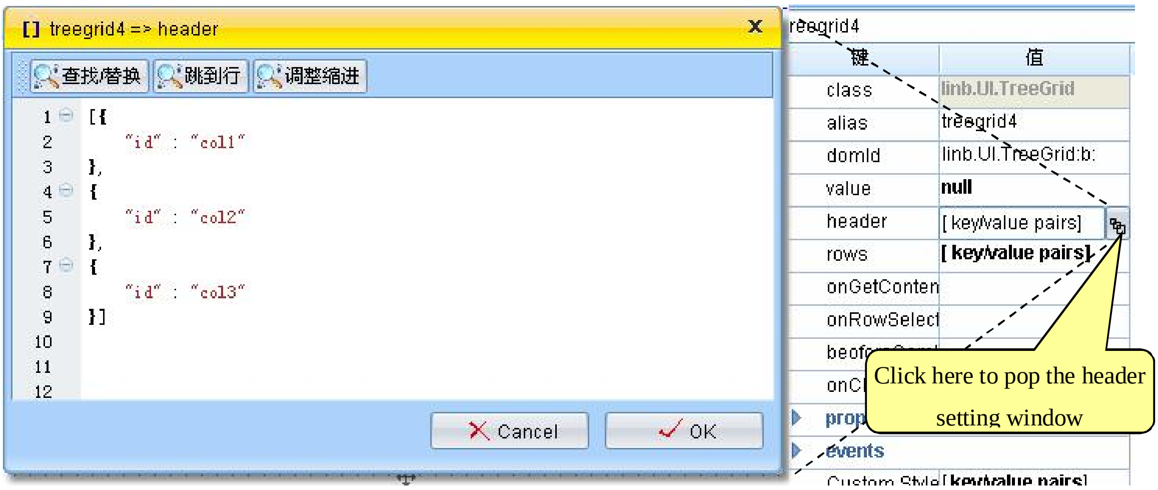
treegrid4

键	值
dataField	
directInput	☑
dirtyMark	☑
disabled	□
display	
dock	fill
dockFloat	□
dockIgnore	□
dockMargi	[key/value pairs]
dockMinH	0
dockMinW	0
dockOrder	1
editable	□
gridHandl	grid
headerHe	
iniFold	☑
left	
position	absolute
renderer	[Function]

gridHandlerCaption

TreeGrid [property] : gridHandlerCaption

4. Sets header and rows



[] treegrid4 => header

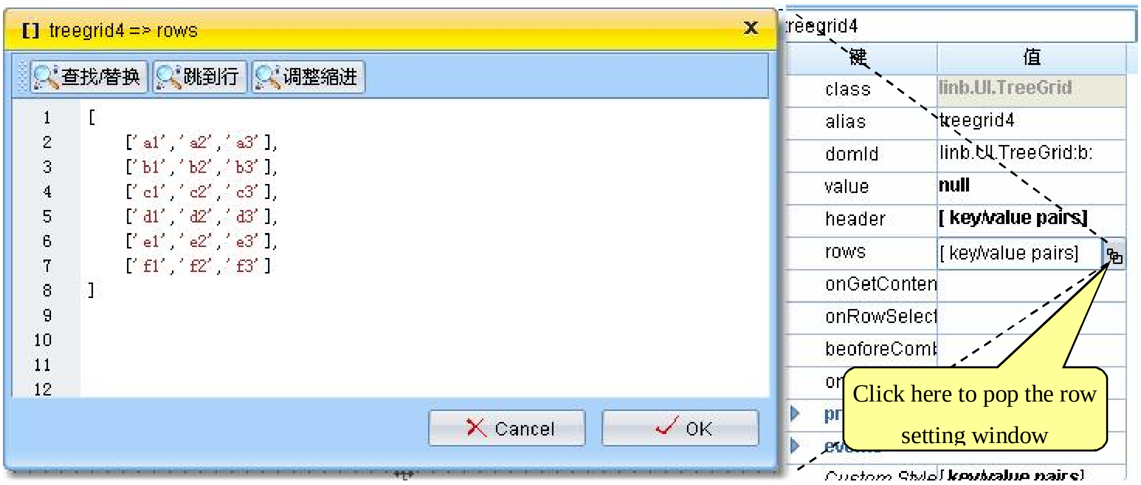
```
[{
    "id" : "col1"
},
{
    "id" : "col2"
},
{
    "id" : "col3"
}]
```

treegrid4

键	值
class	linb.UI.TreeGrid
alias	treegrid4
domId	linb.UI.TreeGrid:b:
value	null
header	[key/value pairs]
rows	[key/value pairs]
onGetConten	
onRowSelect	
beof	
onC	
prop	
events	
Custom Style	[key/value pairs]

Click here to pop the header setting window

Sets header data



```
[
    ['a1','a2','a3'],
    ['b1','b2','b3'],
    ['c1','c2','c3'],
    ['d1','d2','d3'],
    ['e1','e2','e3'],
    ['f1','f2','f3']
]
```

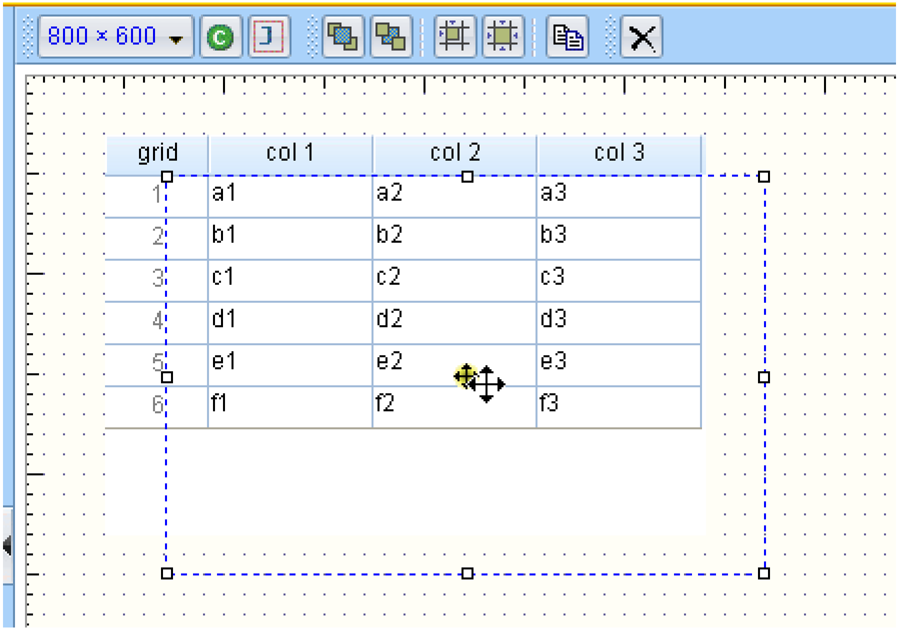
treegrid4

键	值
class	linb.UI.TreeGrid
alias	treegrid4
domId	linb.UI.TreeGrid:b:
value	null
header	[key/value pairs]
rows	[key/value pairs]
onGetConten	
onRowSelect	
beoforeComb	
on	
pr	
eve	
Custom Style	[key/value pairs]

Click here to pop the row setting window

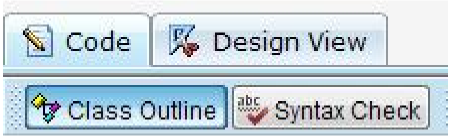
Sets rows data

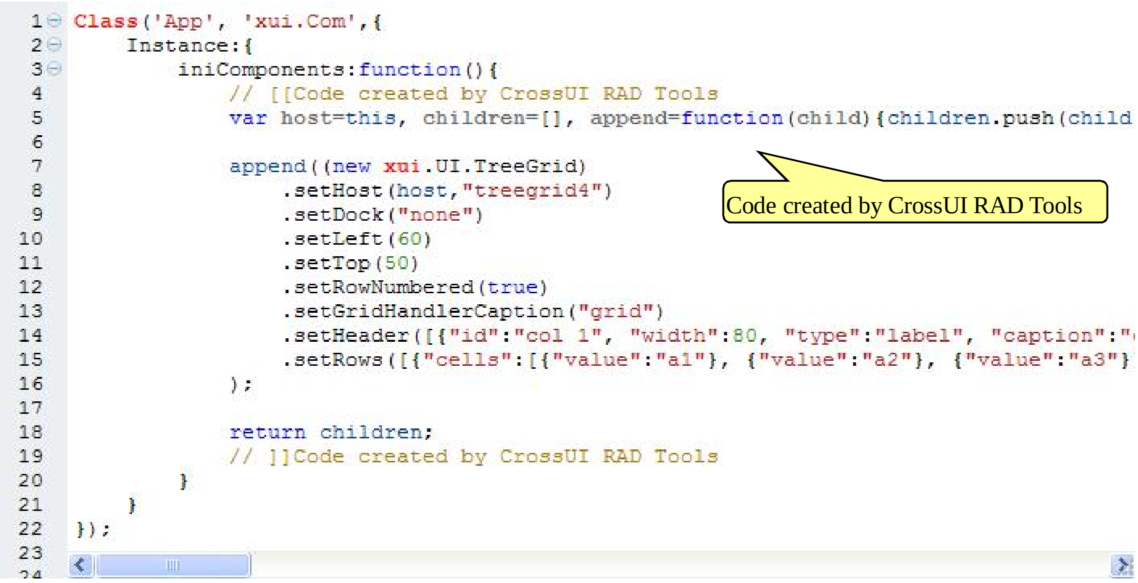
5. Click to select the grid, adjust its position and size



grid	col 1	col 2	col 3
1	a1	a2	a3
2	b1	b2	b3
3	c1	c2	c3
4	d1	d2	d3
5	e1	e2	e3
6	f1	f2	f3

6. Now, switch to “Code” view

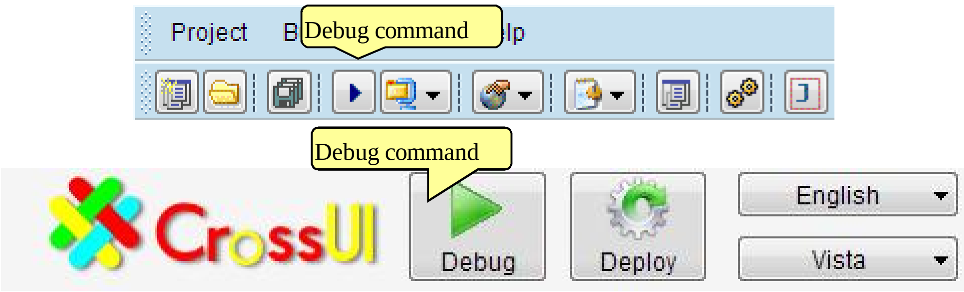




```
Class('App', 'xui.Com',{
    Instance:{
        iniComponents:function(){
            // [[Code created by CrossUI RAD Tools
            var host=this, children=[], append=function(child){children.push(child

            append((new xui.UI.TreeGrid)
                .setHost(host,"treegrid4")
                .setDock("none")
                .setLeft(60)
                .setTop(50)
                .setRowNumbered(true)
                .setGridHandlerCaption("grid")
                .setHeader([{"id":"col 1", "width":80, "type":"label", "caption":"
                .setRows([{"cells":[{"value":"a1"}, {"value":"a2"}, {"value":"a3"}
            );

            return children;
            // ]]Code created by CrossUI RAD Tools
        }
    }
});
```

Above code is serialized by CrossUI RAD. Header data and rows data will not look the same as your setting.

7. Click “Debug” Button to open the test window, you will see the same result with section 2.2.



8. Copy the code from this test page, and paste to a new file designer.grid.html.

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
    <meta http-equiv="imagetoolbar" content="no" />
    <title>Web application powered by XUI framework</title>
</head>
    <body>
        <div id='loading'><img src="../../runtime/loading.gif" alt="Loading..." /></div>
../runtime/loading.gif" alt="Loading..." /></div>
        <script type="text/javascript" src="../../runtime/xui/js/xui-all.js"></script>
        <script type="text/javascript">
            Class('App', 'xui.Com',{
                Instance:{
                    iniComponents:function(){
                        // [[code created by CrossUI RAD Tools
                        var host=this, children=[], append=function(child){children.push(child.get(0))};

                        append((new xui.UI.TreeGrid)
                            .host(host,"treegrid4")
                            .setDock("none")
                            .setLeft(60)
                            .setTop(50)
                            .setRowNumbered(true)
                            .setGridHandlerCaption("grid")
                            .setHeader([{"id":"col 1", "width":80, "type":"label", "caption":"col 1"}, {"id":"col 2", "width":80, "type":"label", "caption":"col 2"}, {"id":"col 3", "width":80, "type":"label", "caption":"col 3"}])
                            .setRows([{"cells":[{"value":"a1"}, {"value":"a2"}, {"value":"a3"}], "id":"j"}, {"cells":[{"value":"b1"}, {"value":"b2"}, {"value":"b3"}], "id":"k"}, {"cells":[{"value":"c1"}, {"value":"c2"}, {"value":"c3"}], "id":"l"}, {"cells":[{"value":"d1"}, {"value":"d2"}, {"value":"d3"}], "id":"m"}, {"cells":[{"value":"e1"}, {"value":"e2"}, {"value":"e3"}], "id":"n"}, {"cells":[{"value":"f1"}, {"value":"f2"}, {"value":"f3"}], "id":"o"}])
                        );

                        return children;
                        // ]]code created by CrossUI RAD Tools
                    }
                }
            });
            xui.Com.load('App', function(){
                xui('loading').remove();
            });
        </script>
</body>
</html>
```

Showing a loading picture

Include lib file in body

Class created by CrossUI RAD.

You can save this part of code to **App/js/index.js**

Load UI in asynchronous mode

If no App Class in memory, by default, CrossUI framewok will load the Class from **App/js/index.js** file .

cookbook/chapter1/designer.grid.html

2.4. Application loading process

In section 2.3, we put all html and JavaScript code in a single file. For a bigger application, it’s not a wise solution. A real application may be include dozens of classes. For a developer, maintaining each class in a separate file is always a must.

OK. Let’s separate “designer.grid.html” into two files → designer.grid.standard.html, and **App/js/index.js**.

designer.grid.standard.html is:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/CSS" />
    <meta http-equiv="imagetoolbar" content="no" />
    <title>Web application powered by XUI framework</title>
</head>
    <body>
        <div id='loading'><img src="../runtime/loading.gif" alt="Loading..." /></div>
        <script type="text/javascript" src="../../runtime/xui/js/xui-all.js"></script>
        <script type="text/javascript">
            xui.Com.load('App', function(){
                xui('loading').remove();
            });
        </script>
</body>
</html>
```

Load App class from App/js/index.js asynchronously

At last, remove loading picture

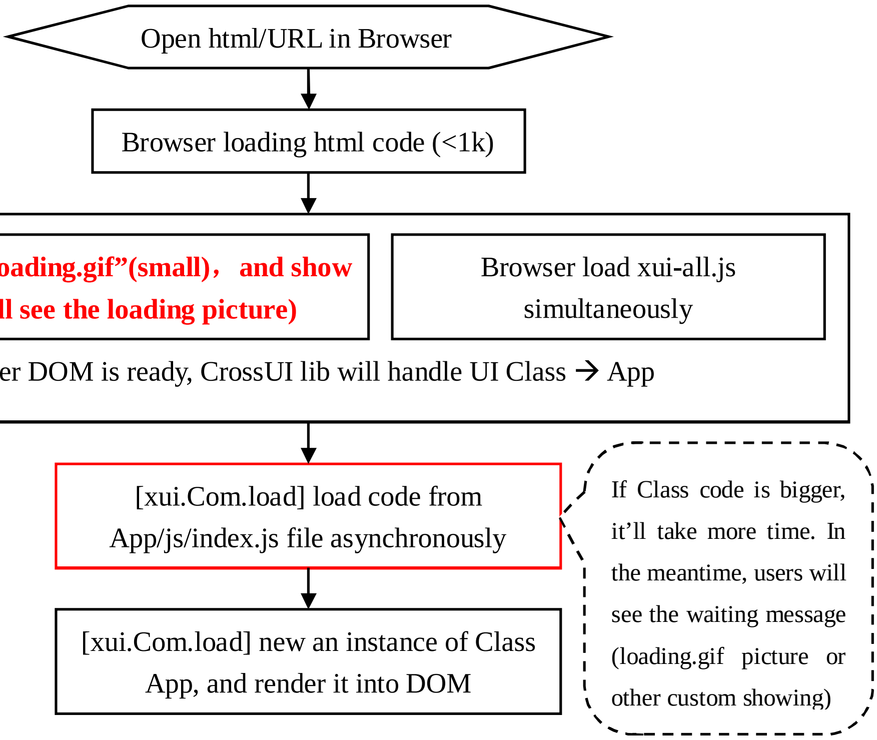
cookbook/chapter1/designer.grid.standard.html

App/js/index.js is:

```
Class('App', 'xui.Com',{
    Instance:{
        iniComponents:function(){
            // [[code created by CrossUI RAD Tools
            var host=this, children=[], append=function(child){children.push(child.get(0))};
            append((new xui.UI.TreeGrid)
                .host(host,"treegrid4")
                .setDock("none")
                .setLeft(60)
                .setTop(50)
                .setRowNumbered(true)
                .setGridHandlerCaption("grid")
                .setHeader([{"id":"col 1", "width":80, "type":"label", "caption":"col 1"}, {"id":"col 2", "width":80,
"type":"label", "caption":"col 2"}, {"id":"col 3", "width":80, "type":"label", "caption":"col 3"}])
                .setRows([{"cells":[{"value":"a1"}, {"value":"a2"}, {"value":"a3"}], "id":"j"}, {"cells":[{"value":"b1"},
{"value":"b2"}, {"value":"b3"}], "id":"k"}, {"cells":[{"value":"c1"}, {"value":"c2"}, {"value":"c3"}], "id":"l"},
{"cells":[{"value":"d1"}, {"value":"d2"}, {"value":"d3"}], "id":"m"}, {"cells":[{"value":"e1"}, {"value":"e2"},
{"value":"e3"}], "id":"n"}, {"cells":[{"value":"f1"}, {"value":"f2"}, {"value":"f3"}], "id":"o"}])
                );
             return children;
             // ]]code created by CrossUI RAD Tools
       }
    }
});
```

cookbook/chapter1/App/js/index.js

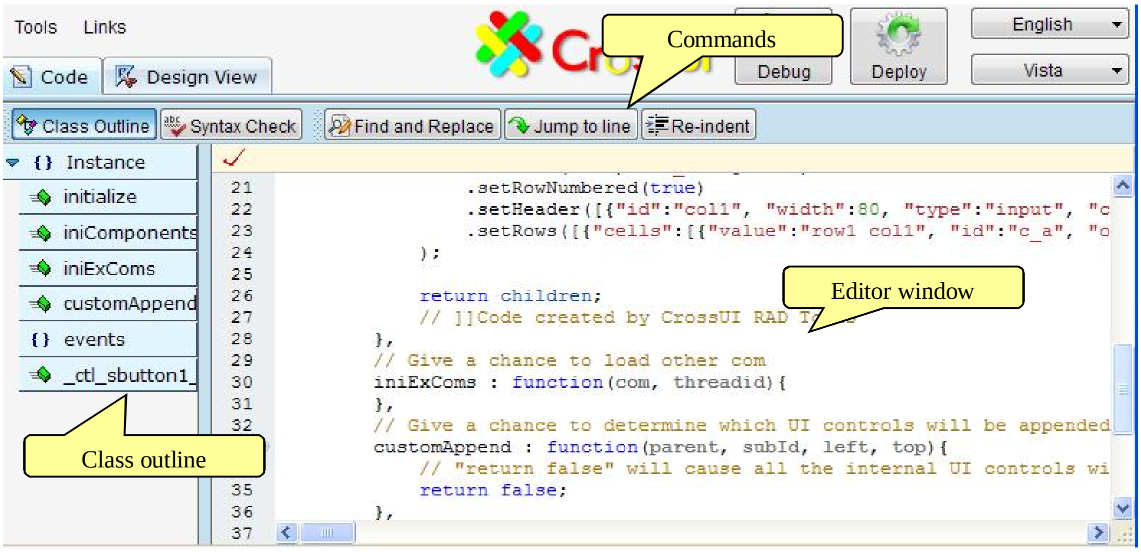
When we open **designer.grid.standard.html** in Browser, the loading process will be:


Open html/URL in Browser
Browser loading html code (<1k)
Browser load “loading.gif”(small)， and show it (Users will see the loading picture)
Browser load xui-all.js simultaneously
After DOM is ready, CrossUI lib will handle UI Class → App
[xui.Com.load] load code from App/js/index.js file asynchronously
[xui.Com.load] new an instance of Class App, and render it into DOM
If Class code is bigger, it’ll take more time. In the meantime, users will see the waiting message (loading.gif picture or other custom showing)

2.5. Code Editor

By the way, if you use CrossUI RAD online, in order to get better performance, firefox and chrome are recommended here.

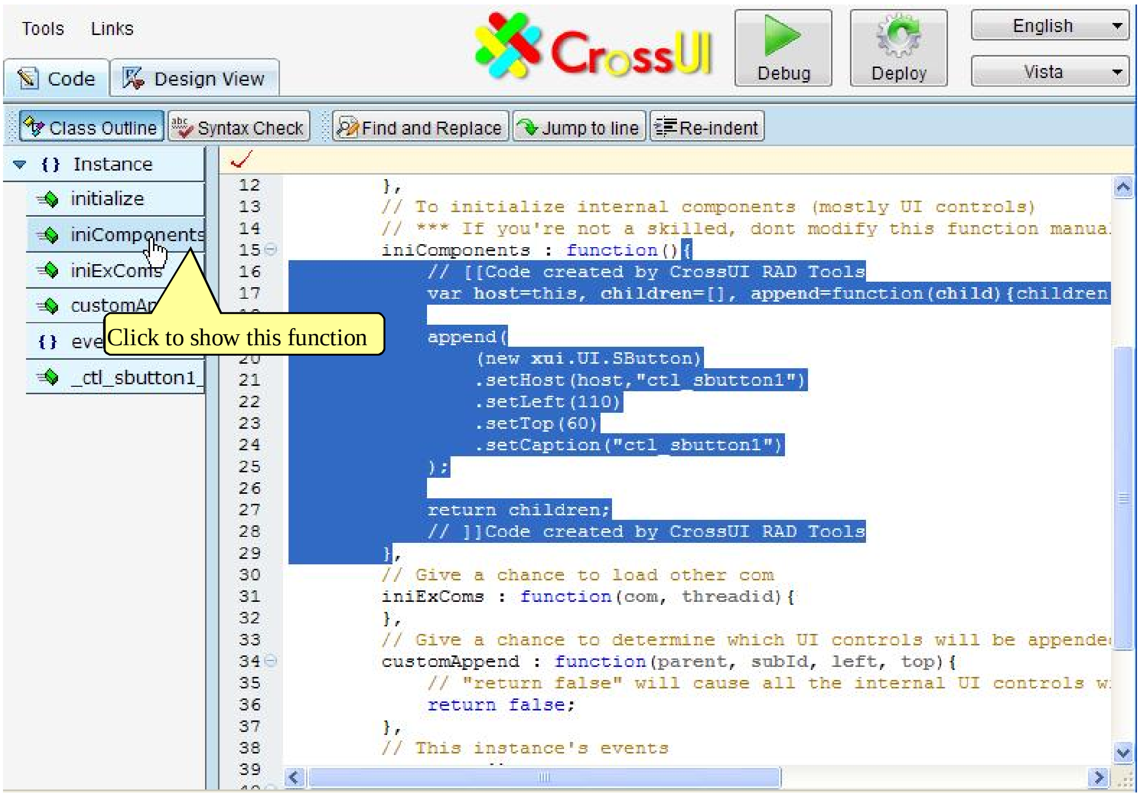
There are two views in Builder: “Design view” and “Code” view. In the online version, the default view is “Design view”. Click “Code” tab to switch to “Code” view.



```
            .setRowNumbered(true)
            .setHeader([{"id":"col1", "width":80, "type":"input", "c
            .setRows([{"cells":[{"value":"row1 col1", "id":"c_a", "o
        );

        return children;
        // ]]Code created by CrossUI RAD T
    },
    // Give a chance to load other com
    iniExComs : function(com, threadid){
    },
    // Give a chance to determine which UI controls will be appended
    customAppend : function(parent, subId, left, top){
        // "return false" will cause all the internal UI controls wi
        return false;
    },
```

2.5.1. Highlight code from Outline window

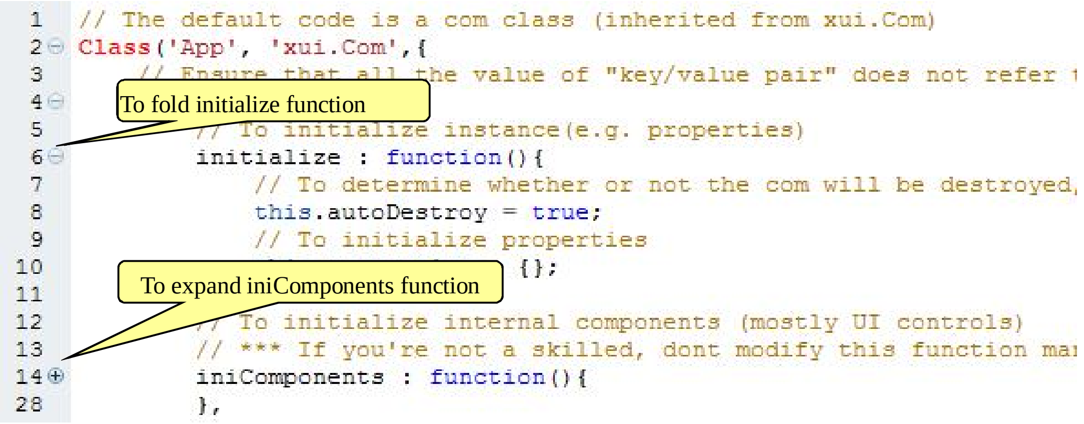
“Class Outline” is located in the left side of “Code” view. By clicking any member or method name in “Class Outline”, RAD Tools will highlight its code in “Editor window”, and scroll “Editor window” to show the code.



```
        },
        // To initialize internal components (mostly UI controls)
        // *** If you're not a skilled, dont modify this function manua
        iniComponents : function(){
            // [[Code created by CrossUI RAD Tools
            var host=this, children=[], append=function(child){children

            append(
                (new xui.UI.SButton)
                .setHost(host,"ctl_sbutton1")
                .setLeft(110)
                .setTop(60)
                .setCaption("ctl_sbutton1")
            );

            return children;
            // ]]Code created by CrossUI RAD Tools
        },
        // Give a chance to load other com
        iniExComs : function(com, threadid){
        },
        // Give a chance to determine which UI controls will be appende
        customAppend : function(parent, subId, left, top){
            // "return false" will cause all the internal UI controls w
            return false;
        },
        // This instance's events
```

2.5.2. Code Folding

To make your code view more clear to read and understand, CrossUI RAD lets you fold certain parts of it. Click the left side “plus” or “minus” will fold or expand the block code.



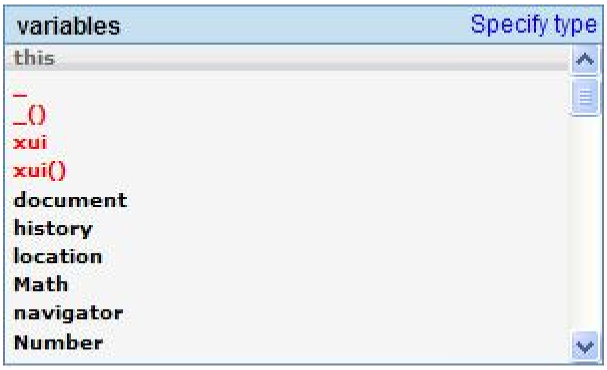
```
// The default code is a com class (inherited from xui.Com)
Class('App', 'xui.Com',{
    // Ensure that all the value of "key/value pair" does not refer t


        // To initialize instance(e.g. properties)
        initialize : function(){
            // To determine whether or not the com will be destroyed,
            this.autoDestroy = true;
            // To initialize properties
                                    {};


        // To initialize internal components (mostly UI controls)
        // *** If you're not a skilled, dont modify this function ma
        iniComponents : function(){
        },
```

Note: Due to some browser’s poor performance, please try not to frequent collapse or expand the large body function or object.

2.5.3. Code Intellisense

Three types Code Intellisense are supported.

- When context does not recognize the input string;
- Type dot after a recognizable variable
- When dbclick a recognizable variable

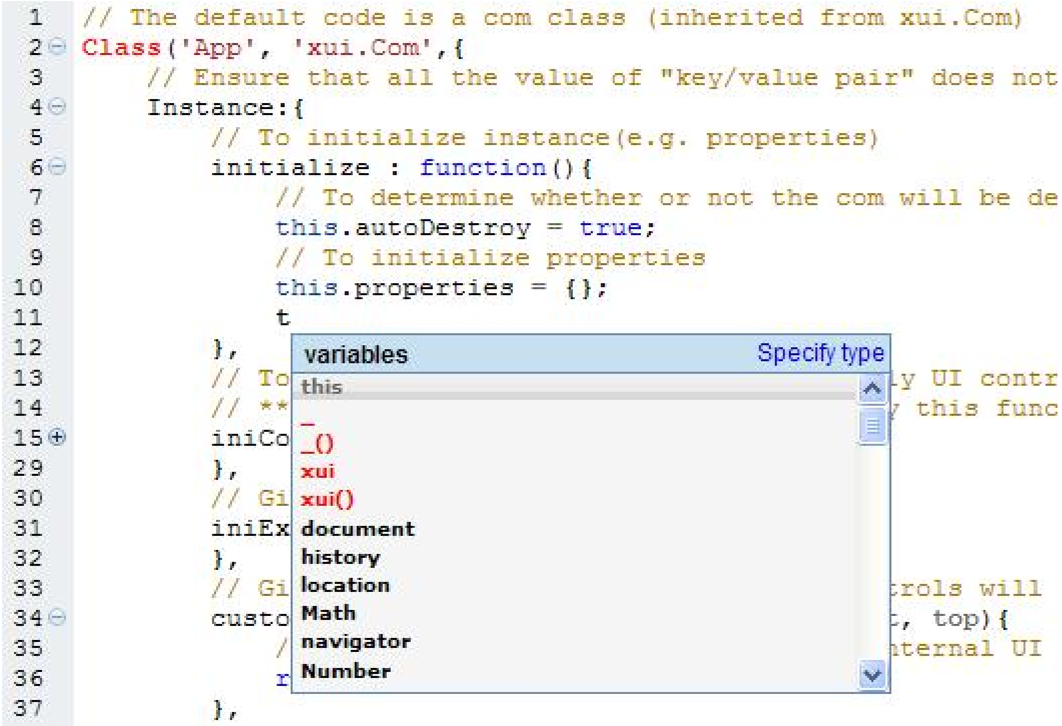


Keyboard actions for Code Intellisense pop Window:

- “up”: Focus to next item in code list
- “down”: Focus to previous item in code list
- “enter”: Select the current focused item, and input to editor window
- “esc”: Close the pop window
- Other visible chars: Find and focus the first matched item

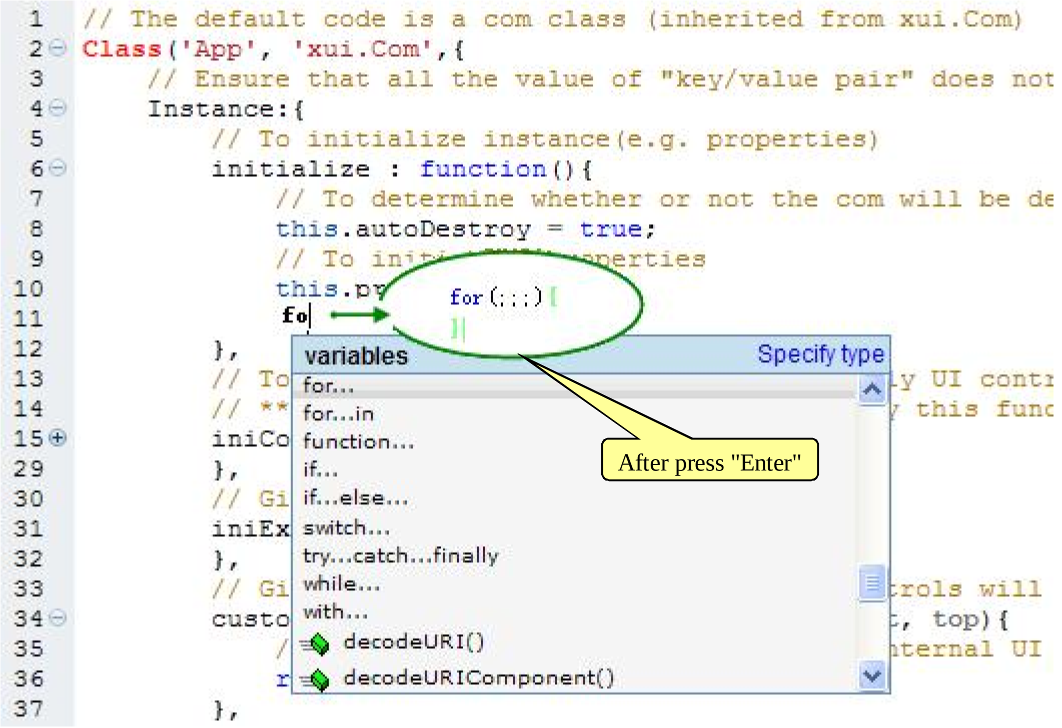
2.5.3.1. When context doesn’t recognize the input string

When you input a string, if editor doesn’t recognize this string, it will pop a list window including local variables, global variables, global functions and JavaScript reserved keywords. In the below picture, type ‘t’ will trigger editor to pop a list window, “this” is the default focused item.



```
// The default code is a com class (inherited from xui.Com)
Class('App', 'xui.Com',{
    // Ensure that all the value of "key/value pair" does not
    Instance:{
        // To initialize instance(e.g. properties)
        initialize : function(){
            // To determine whether or not the com will be de
            this.autoDestroy = true;
            // To initialize properties
            this.properties = {};
            t
        },
        // To                                  ly UI contr
        // **                                  y this func
        iniCo
        },
        // Gi
        iniEx
        },
        // Gi                                  trols will
        custo                                  t, top){
            /                                  nternal UI
            r
        },
```

If the input string is “fo”, the “for loop statement” will be the default focused item.



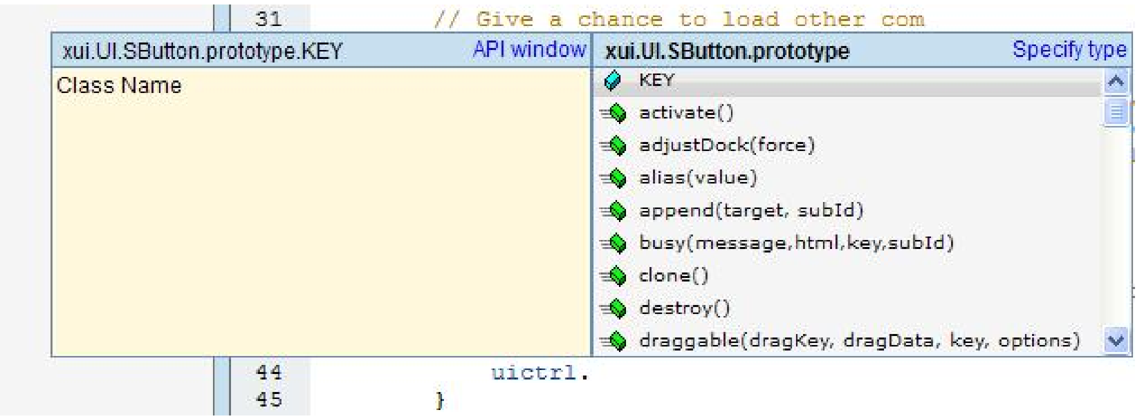
```
// The default code is a com class (inherited from xui.Com)
Class('App', 'xui.Com',{
    // Ensure that all the value of "key/value pair" does not
    Instance:{
        // To initialize instance(e.g. properties)
        initialize : function(){
            // To determine whether or not the com will be de
            this.autoDestroy = true;
            // To ini          perties
            this.pr
            fo
        },
        // To                                  ly UI contr
        // **                                  y this func
        iniCo
        },
        // Gi
        iniEx
        },
        // Gi                                  trols will
        custo                                  t, top){
            /                                  nternal UI
            r
        },
```

```
for(;;){
}
```

In this case, "Enter" keypress will cause “for loop statement” code to be inserted into the editor automatically.

2.5.3.2. Type dot after a recognizable variable

After an editor recognizable variable, if you type char “.”, editor will pop an available members and functions list for the variable.



```
// Give a chance to load other com
```

xui.UI.SButton.prototype.KEY — API window

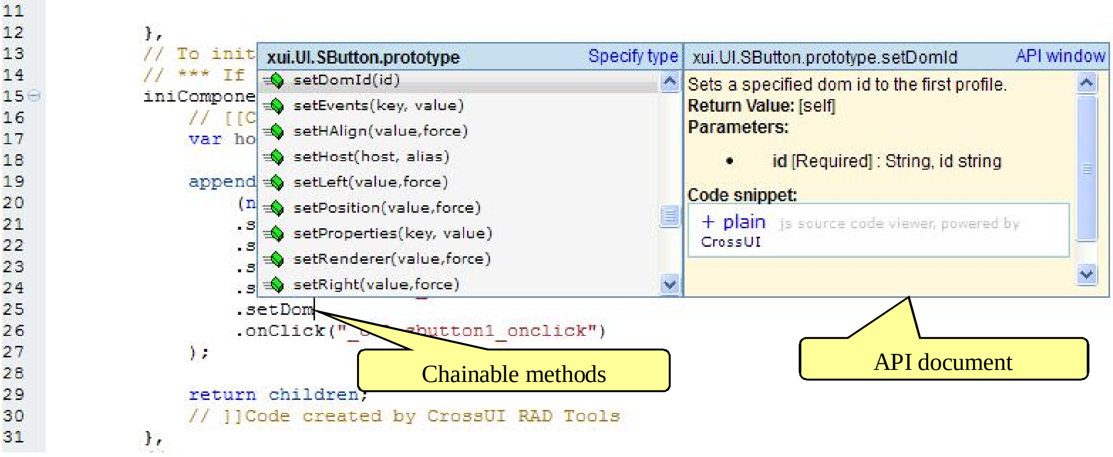
Class Name

xui.UI.SButton.prototype — Specify type

- KEY
- activate()
- adjustDock(force)
- alias(value)
- append(target, subId)
- busy(message,html,key,subId)
- clone()
- destroy()
- draggable(dragKey, dragData, key, options)

```
uictrl.
}
```

Chainable methods can show Code Intellisense window too.



```
},
// To init
// *** If
iniCompone
    // [[C
    var ho

    append
        (n
        .s
        .s
        .s
        .s
        .setDom
        .onClick("_c   sbutton1_onclick")
    );

    return children;
    // ]]Code created by CrossUI RAD Tools
},
```

xui.UI.SButton.prototype — Specify type

- setDomId(id)
- setEvents(key, value)
- setHAlign(value,force)
- setHost(host, alias)
- setLeft(value,force)
- setPosition(value,force)
- setProperties(key, value)
- setRenderer(value,force)
- setRight(value,force)

xui.UI.SButton.prototype.setDomId — API window

Sets a specified dom id to the first profile.
Return Value: [self]
Parameters:

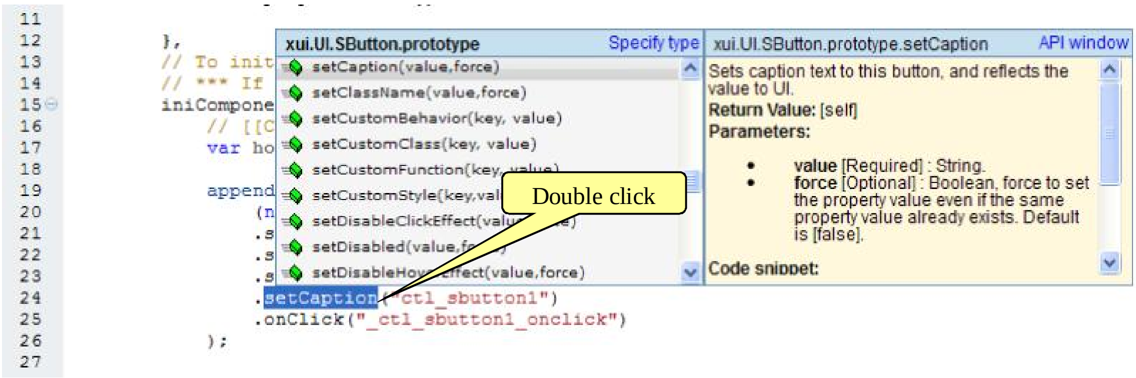
- **id** [Required] : String, id string

Code snippet:

+ plain   js source code viewer, powered by CrossUI

2.5.3.3. When use dbclick

Double click one variable string will trigger editor to pop the Code Intellisense window.



```
},
// To init
// *** If
iniCompone
    // [[C
    var ho

    append
        (n
        .s
        .s
        .s
        .setCaption("ctl_sbutton1")
        .onClick("_ctl_sbutton1_onclick")
    );
```

xui.UI.SButton.prototype — Specify type

- setCaption(value,force)
- setClassName(value,force)
- setCustomBehavior(key, value)
- setCustomClass(key, value)
- setCustomFunction(key, value)
- setCustomStyle(key,val
- setDisableClickEffect(valu
- setDisabled(value,fo
- setDisableHoverEffect(value,force)

xui.UI.SButton.prototype.setCaption — API window

Sets caption text to this button, and reflects the value to UI.
Return Value: [self]
Parameters:

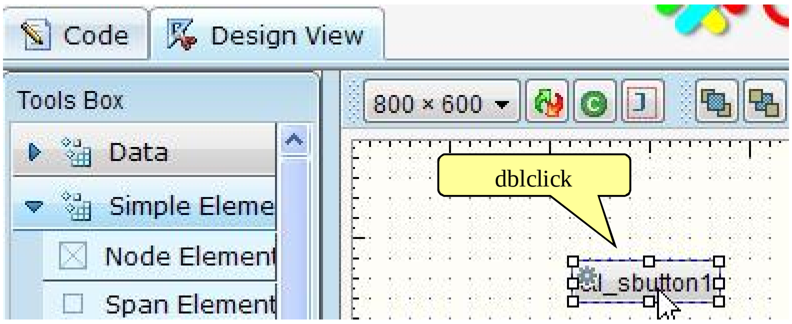
- **value** [Required] : String.
- **force** [Optional] : Boolean, force to set the property value even if the same property value already exists. Default is [false].

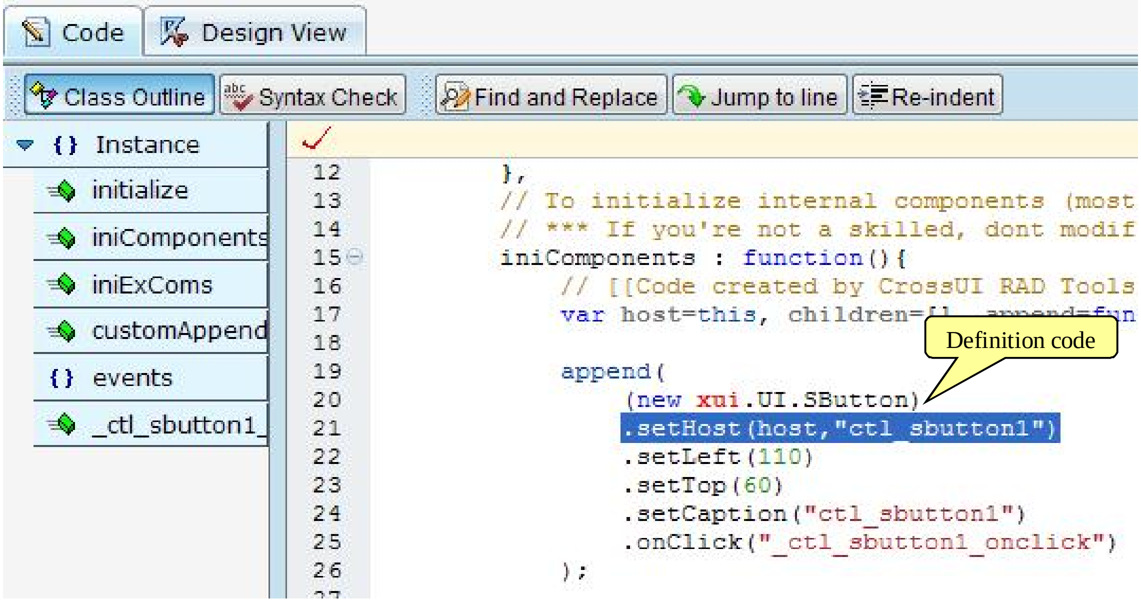
Code snippet:

2.5.4. Find the object definition code

In “Design View”, double click a control will cause:

1) Switch to “Code” view;
2) Highlight the control’s definition code;
3) Scroll the definition code to view.



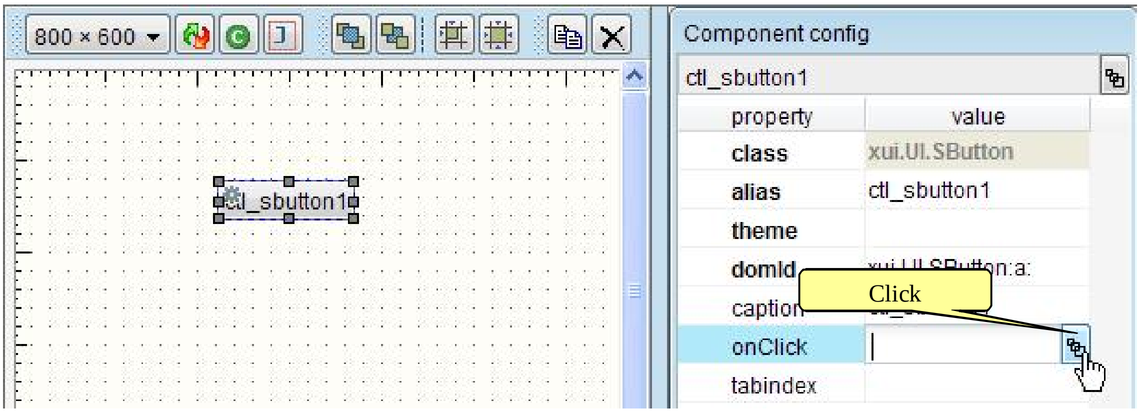


```
        },
        // To initialize internal components (most
        // *** If you're not a skilled, dont modif
        iniComponents : function(){
            // [[Code created by CrossUI RAD Tools
            var host=this, children=[], append=fun

            append(
                (new xui.UI.SButton)
                .setHost(host,"ctl_sbutton1")
                .setLeft(110)
                .setTop(60)
                .setCaption("ctl_sbutton1")
                .onClick("_ctl_sbutton1_onclick")
            );
```

2.5.5. Generate event code automatically

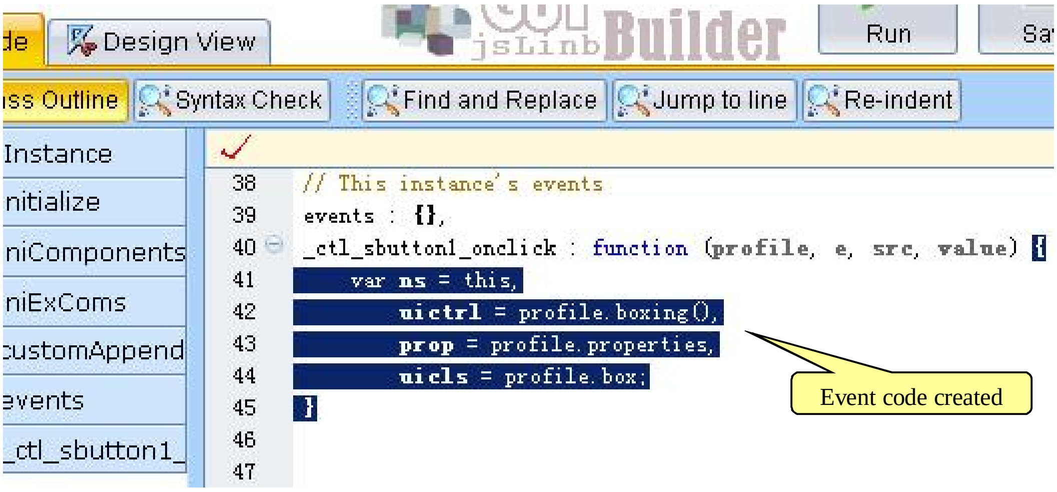
In the “Design View”, select a control; the right side “Component config” window will be refreshed. Find an event (e.g. onClick event), click its event button will cause:

1) Switch to “Code” view;
2) Create event code, and insert into the editor;
3) Scroll the event code to view.



ctl_sbutton1

property	value
class	xui.UI.SButton
alias	ctl_sbutton1
theme	
domId	xui.UI.SButton:a:
caption	
onClick	
tabindex	



```
// This instance's events
events : {},
_ctl_sbutton1_onclick : function (profile, e, src, value) {
    var ns = this,
        uictrl = profile.boxing(),
        prop = profile.properties,
        uicls = profile.box;
}
```

Chapter 3. Controls Facebook

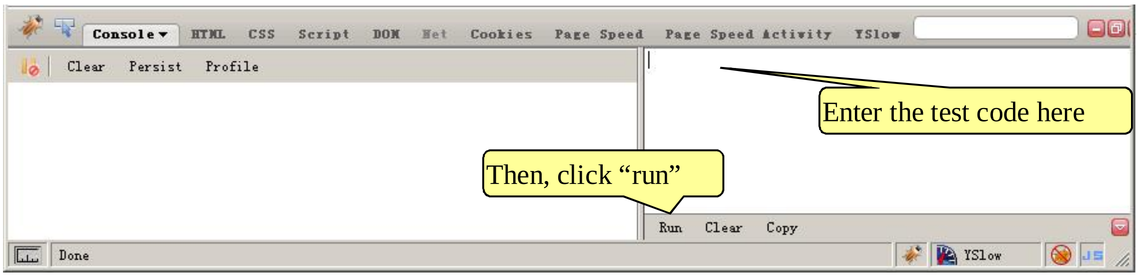
Many beginners are particularly interested in UI controls. In this chapter we'll give a rough look at the basic controls. Since each control has a lot of functions, here is a brief introduction, it is impossible to explain all the functions. You can browse API to understand the specific function of each control in detail!

3.1. Script testing environment

At first, we have to build a testing environment for executing example codes. About Browsers, firefox is recommended, if firefox is not preferred, ie8 or chrome is ok too.

For firefox:

1. You need firefox and firebug;
2. Ensure all files and folders in cookbook package including "**env.html**" are in **cookbook** dir;
3. Open URL **cookbook**/**env.html** in firefox;
4. Open firebug console, switch to the multi-line mode

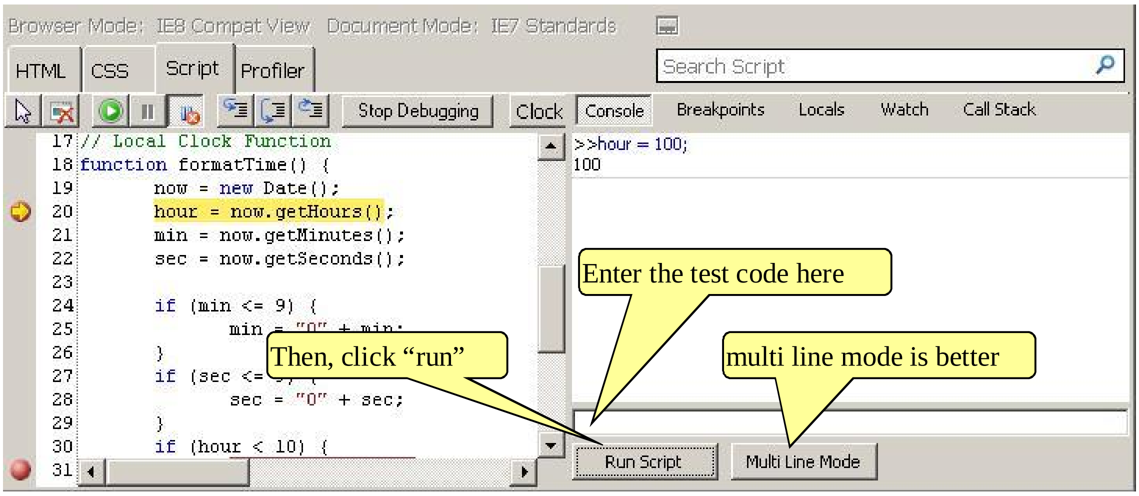


Enter the test code here

Then, click “run”

For IE8+:

1. You need IE8;
2. Open URL **cookbook/env.html** ;
3. Open developer tools, switch to the multi-line mode



```
// Local Clock Function
function formatTime() {
        now = new Date();
        hour = now.getHours();
        min = now.getMinutes();
        sec = now.getSeconds();

        if (min <= 9) {
                min = "0" + min;
        }
        if (sec <= 9) {
                sec = "0" + sec;
        }
        if (hour < 10) {
```

```
>>hour = 100;
100
```

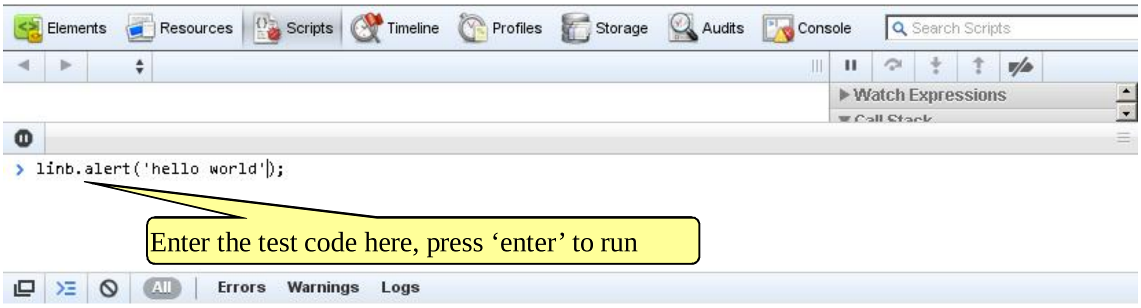
Enter the test code here

Then, click “run”

multi line mode is better

For Chrome:

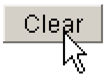
1. You need the latest Chrome;
2. Open URL **cookbook/env.html** ;
3. Open developer tools



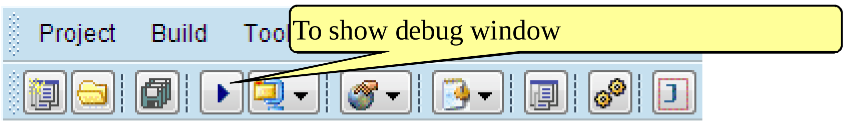
```
> linb.alert('hello world');
```

Enter the test code here, press ‘enter’ to run

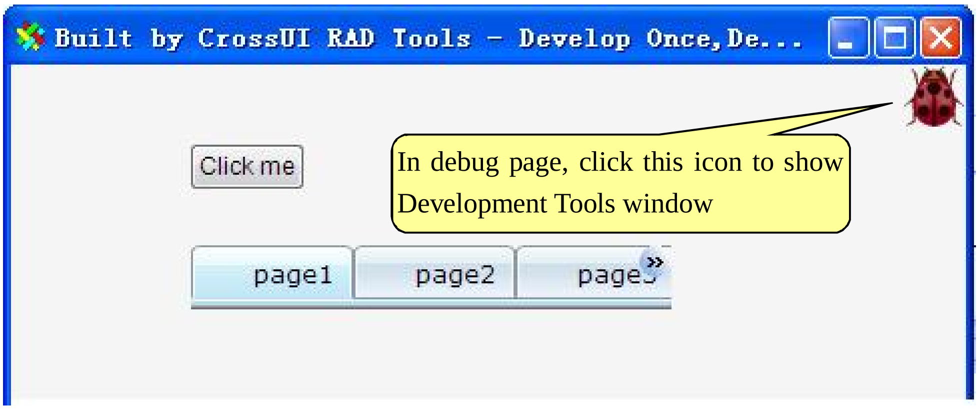
There’s a “Clear” button in **cookbook/env.html**, You can click this button to clean up the current page’s DOM. In some cases, you want to clean up both DOM and memory, press ‘F5’ to refresh your browser.



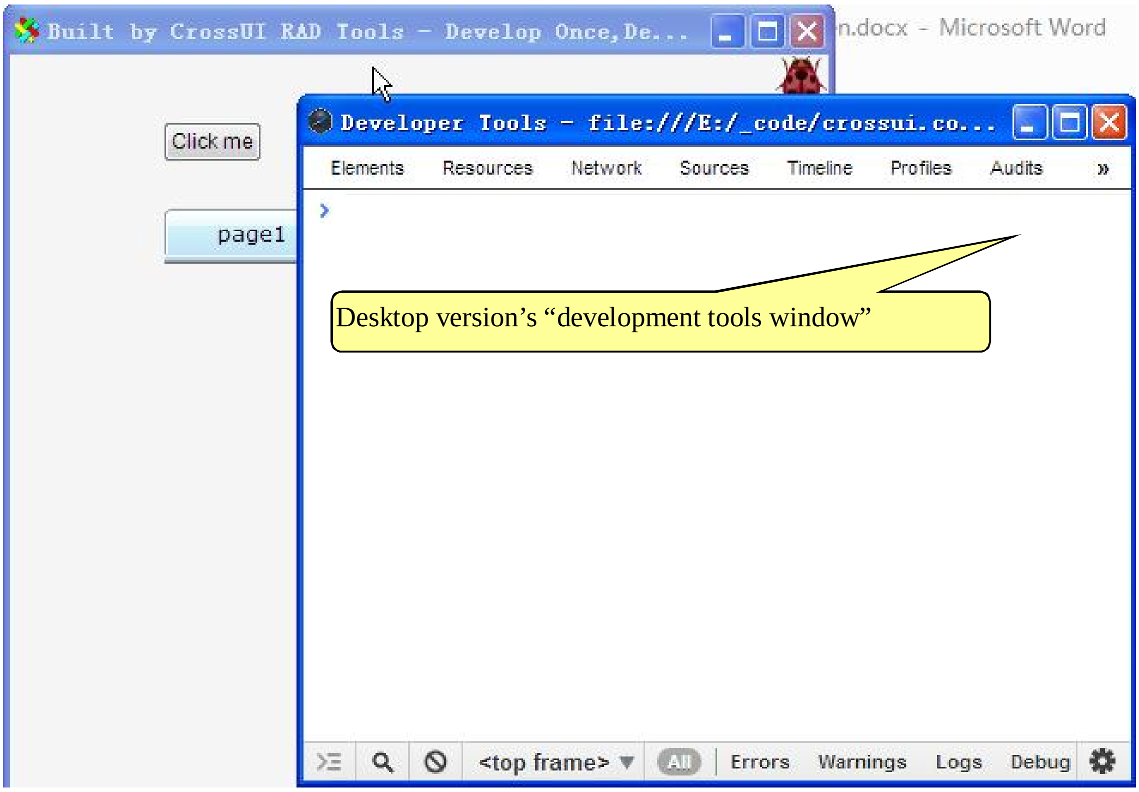
For CrossUI Desktop version:



To show debug window



In debug page, click this icon to show Development Tools window



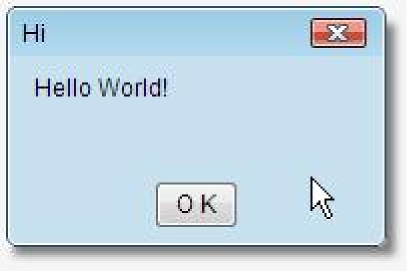
Desktop version's "development tools window"

3.2. "Hello world" in env.html

Input the following code into script window, and run it.

```
xui.alert("Hi", "Hello World!");
```

Output:



Click “Clear” button to clean the DOM.

If you are in CrossUI Desktop, execute the following line in console:

```
xui ("body"). empty();
```

3.3. Control creation and runtime update

There are three approaches to create CrossUI control.

```
// Approach 1
xui.create("SButton", {
        caption: "Using xui.create function",
        position: "relative"}
).show();

// Approach 2
(new xui.UI.SButton({
        caption: "Using new and key/value pairs",
        position: "relative"
})).show();

// Approach 3
(new xui.UI.SButton())
.setCaption("Using new and get/set")
.setPosition("relative")
.show();
```

We use new/setXX mode in RAD Tools

The above three approaches will create entirely consistent UI.

You can use setXXX function to update the control after it was rendered into DOM (runtime update).

```
var dlg=xui.create("Dialog", {caption: "runtime "}).show();
_.asyRun(function(){
      dlg.setCaption("updated");
},500);
_.asyRun(function(){
      dlg.setMaxBtn(false);
},1000);
_.asyRun(function(){
      dlg.setStatus("max");
},1500);
_.asyRun(function(){
      dlg.destroy();
},2000);
```

Create a Dialog

To modify caption

To hide the max button

To modify status

To destroy it

3.4. Button related

This section relates to the following controls: xui.UI.Link, xui.UI.SButton, xui.UI.Button, xui.UI.SCheckBox and xui.UI.CheckBox.

3.4.1. onClick event

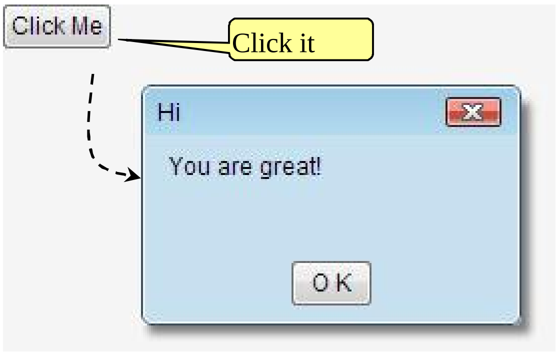
Input:

```
var btn=new xui.UI.SButton();
btn.setCaption("Click Me")
.onClick(function(){
      xui.alert("Hi","You are great!");
});
btn.show();
```

Sets caption

Adds onClick evnet

Output:



You are great!

Input:

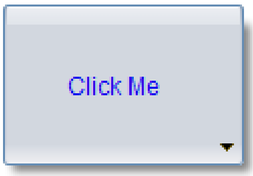
```
var btn=new xui.UI.Button();
btn.setCaption("Click Me")
.onClick(function(){
        xui.alert("Hi","You are great!");
});
btn.show();

_.asyRun(function(){
btn.setHeight(80)
      .setShadow(true)
      .setType("drop")
},1000);
```

Sets caption

Adds onClick event

Execute code after 1 second

Output:



NOTE
xui.UI.SButton / SLabel / SCheckbox are enough for most cases; Only if you need more complex feature, you should use those complex control: **xui.UI.Button / Label / Checkbox**.

3.4.2. Boolean Controls

There are three controls can represent and modify Boolean value:

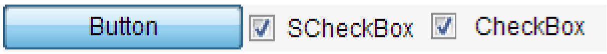
Input:

```
var btn= (new xui.UI.Button({position: "relative", caption:"Button", type:"status"})).show();
var scb= (new xui.UI.SCheckBox({position: "relative", caption:" SCheckBox"})).show();
var cb= (new xui.UI.CheckBox({position: "relative", caption:" CheckBox"})).show();

_.asyRun(function(){
        btn.setValue(true,true);
        scb.setValue(true,true);
        cb.setValue(true,true);
},1000);
```

Sets position to ‘relative’

Sets values to true after 1 second

Output:



3.4.3. Link Control

You can take xui.UI.Link as a simple button.

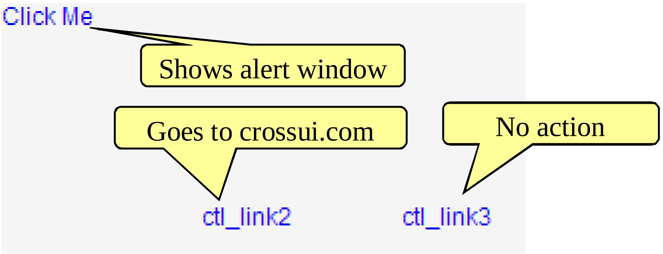
Input:

```
var btn=new xui.UI.Link();
btn.setCaption("Click Me")
.onClick(function(){
        xui.alert("Hi","You are great!");
});
btn.show();

// href property
xui.create("Link",{href: http://www.crossui.com, target: "_blank"}).show(null,null,100,100)

// href was disabled when return false
xui.create("Link",{href: "http://www.longboo.com"}).show(null,null,200,100)
.onClick(function(){
        return false;
});
```

Sets caption

Adds onClick event

Output:



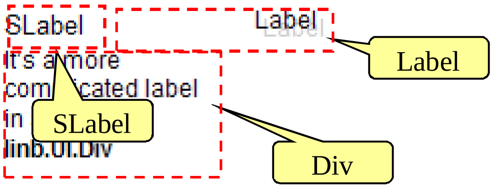
3.5. Label related

This section relates to the following controls: xui.UI.SLabel, xui.UI.Label and xui.UI.Div. These three controls can be used as "label", xui.UI.SLabel is the simplest one, but it's enough for most cases; If you need more complex feature like shadow, resizer or border, you should choose xui.UI.Label; Or if you want to input more complex html code in the control, xui.UI.Div is better.

Input:

```
(new xui.UI.SLabel()).setCaption("SLabel").setPosition("relative").show();
(new xui.UI.Label()).setCaption("Label").setPosition("relative").setShadowText(true).show();
(new xui.UI.Div()).setHtml("It's a more complicated label in a <br /><b>xui.UI.Div</b>")
.setPosition("relative").show();
```

Output:



3.6. Input related

This section relates to the following controls: xui.UI.Input, xui.UI.ComboInput and xui.UI.RichEditor. xui.UI.ComboInput is an enhanced version of xui.UI.Input, it can input/edit value through a pop window; xui.UI.RichEditor is a rich text input/edit control.

3.6.1. setValue/setValue/getUIValue/setUIValue

From the users point of view, value controls (all derived from the xui.absValue control) in CrossUI has two values has two values: the “UI value”(**getUIValue/setUIValue**) and the “control value”(**getValue/setValue**).

“UI value” dose not always equal to “control value”. For example, for an empty input control

1. Keyboard input “**abc**”: “UI value” is “**abc**”, “control value” is **empty**;
2. Calls “updateValue” function: “UI value” is “**abc**”, “control value” is “**abc**”;
3. Calls “setValue(‘**bcd’**)”: “UI value” is “**bcd**”, “control value” is “**bcd**”;
4. Calls “setUIValue(‘**efg’**)”: “UI value” is “**efg’**”, “control value” is “**bcd**”
5. Calls “resetValue(‘**x**’)”: “UI value” is “**x**”, “control value” is “**x**”;

```
var input = (new xui.UI.Input()).show();
      xui.message(input.getUIValue()+":"+input.getValue());
_.asyRun(function(){
      input.setUIValue('uivalue');
      xui.message(input.getUIValue()+":"+input.getValue());
},2000);
_.asyRun(function(){
      input.updateValue();
      xui.message(input.getUIValue()+":"+input.getValue());
},4000);
```

A new Input

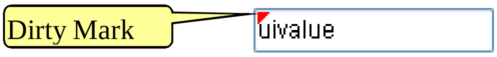
Sets UI value

Updates UI value to control value

You can go to http://www.crossui.com/xui/Examples/comb/DataBinder/index.html for more information about it.

3.6.2. Dirty Mark

If the control’s dirtyMark property is set to true, when “UI value” does not equal to “control value”, a “Dirty Mark” will appear. The “Dirty Mark” will disappear when “UI value” equals to “control value”.



.

```
var input = (new xui.UI.Input()).show();
_.asyRun(function(){
        input.setUIValue('uivalue');
},1000);
_.asyRun(function(){
        input.updateValue();
        input.setDirtyMark (false);
},2000);
_.asyRun(function(){
        input.setUIValue('uivalue 2');
},3000);
```

Dirty Mark appears

Dirty Mark disappears

If DirtyMark is disabled

Nothing happen

3.6.3. Password Input

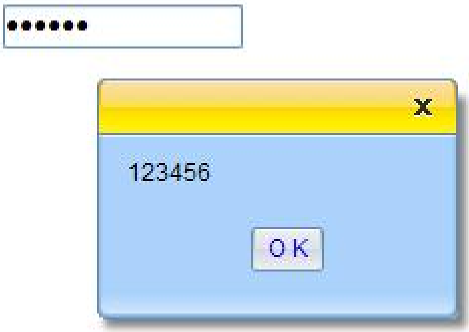
Sets Input's type property to "password".

Input:

```
var input = (new xui.UI.Input({type: 'password'})).show();

_.asyRun(function(){
        input.setUIValue('123456').updateValue();
        xui.pop(input.getValue());
},1000);
```

Sets type

Output:



●●●●●●

123456

3.6.4. Multi-lines

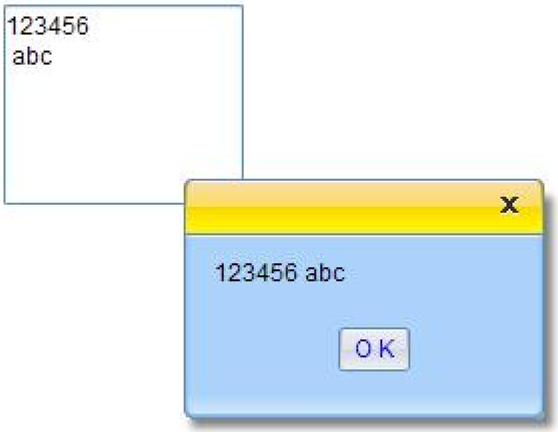
Sets Input's multiLine property to true.

Input:

```
var input = (new xui.UI.Input()).setMultiLines(true).setHeight(100).show();

_.asyRun(function(){
      input.setUIValue('123456 \n abc').updateValue();
      xui.pop(input.getValue());
},1000);
```

Sets multiLine to true

Output:



123456
abc

123456 abc

3.6.5. Input validation

3.6.5.1. valueFormat property

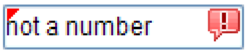
“valueForamt” property represents a regular expression.

Input:

```
var input = (new xui.UI.Input())
      .setValueFormat("^-?\\d\\d*$")
      .show();
```

Number only

Executes the above code, input some charts, and let it lose the mouse focus, the “Error Mark” will appear.



not a number

3.6.5.2. beforeFormatCheck event

Input:

```
    var input = (new xui.UI.Input())
            .beforeFormatCheck(function(profile,value){
                    if(value!=parseFloat(value).toString())
                            return false;
              })
            .show();
```

Number only

In above methods, “beforeFormatCheck” has priority. That means, when "beforeFormatCheck" returns ‘false’, "valueFormat" property will be ignored.

3.6.6. Dynamic input validation

In previous section examples, “Error Mark” appears only when the control loses focus. If you want to a real-time input validation , you need to set dynCheck property to true.

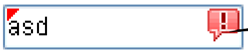
```
var input = (new xui.UI.Input())
        .setDynCheck(true)
        .setValueFormat("^-?\\d\\d*$")
        .show();
```

Sets dynCheck

3.6.7. Error Mark

3.6.7.1. Default Error Mark

The default “Error Mark” is an icon at the right side of Input.



asd

The default Error Icon

3.6.7.2. Validation Tips

There are three tool tips in xui.UI.Input control:

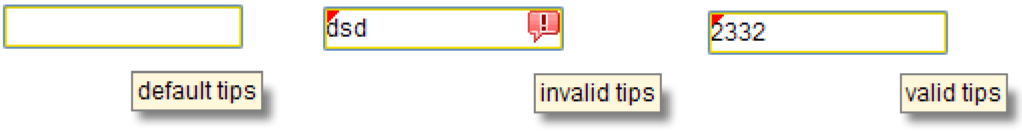
- tips: the default tool tips
- tipsOK: the valid tool tips
- tipsErr: the invalid tool tips

Input:

```
var input = (new xui.UI.Input())
        .setTips("default tips")
        .setTipsErr("invalid tips ")
        .setTipsOK("valid tips")
       .setValueFormat("^-?\\d\\d*$")
       .show();
```

Sets those tips

Output:



3.6.7.3. Binding Validation

You can bind the validation tips to a xui.UI.Div, xui.UI.SLabel or xui.UI.Span.

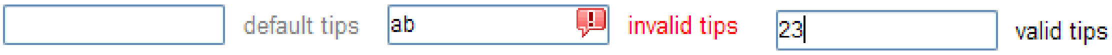
Input:

```
var slbl= (new xui.UI.SLabel({position:'relative'})).setCustomStyle({KEY:'padding-left:10px'});
var input = (new xui.UI.Input({position:'relative'}))
    .setValueFormat("^-?\\d\\d*$")
    .setTipsBinder(slbl)
    .setDynCheck(true)
   .setTips(" default tips")
   .setTipsErr(" invalid tips ")
   .setTipsOK(" valid tips")

input.show();
 slbl.show();
```

Sets tipsBinder

Show SLabel here

Output:



3.6.7.4. Custom Error Mark

We can custom “Error Mark” in beforeFormatMark event.

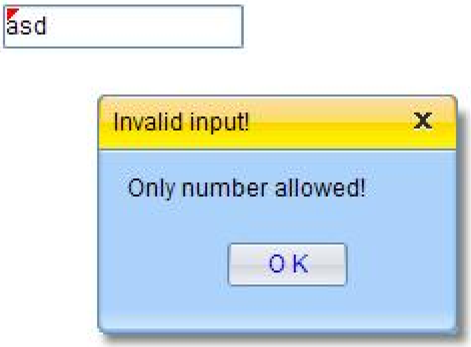
Input:

```
var input = (new xui.UI.Input())
    .setValueFormat("^-?\\d\\d*$")
    .beforeFormatMark(function(profile,err){
       if(err)
          xui.alert("Invalid input!","Only number allowed!",function(){
             profile.boxing().activate();
          });
       return false;
    }).show();
```

Customs information and aciton

Return false to ignore the default action

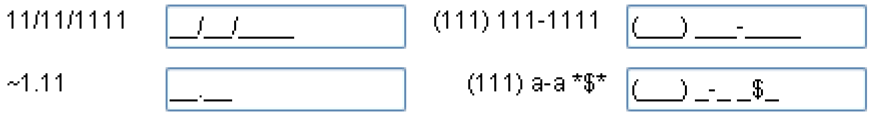
Output:



Only number allowed!

3.6.8. Mask Input

Mask Input examples:



In **chapter2\Input\index.html**

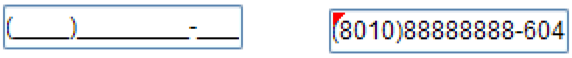
There is a mask property in xui.UI.Input control. It’s a string. In this string,

- ‘~’ represents [+-]
- ‘1’ represents [0-9]
- ‘a’ represents [A-Za-z]
- ‘u’ represents [A-Z]
- ‘l’ represents [a-z]
- ‘*’ represents [A-Za-z0-9]
- Other visible char represents itself

Input:

```
var input = (new xui.UI.Input())
    .setMask("(1111)11111111-11")
    .show();
```

Output:



NOTE

chapter2\Input\index.html is an overall example for Input.

3.6.9. xui.UI.ComboInput

xui.UI.ComboInput is an advanced Input.

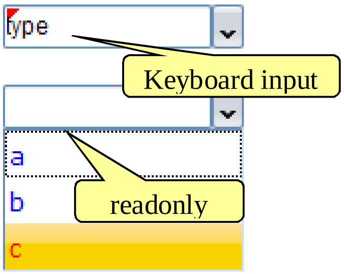
3.6.9.1. Pop list for selection

When type property was set to “combobox”, “listbox” or “helpinput”, click the command button will trigger to pop a list window for selection.

```
(new xui.UI.ComboInput({items:[ "a","b","c"]})).show();
(new xui.UI.ComboInput({type: "listbox", top:"100", items:[ "a","b","c"]})).show();
```

combobox

listbox



3.6.9.2. combobox, listbox and helpinput

There’s an items property in xui.UI.ComboInput (And all list related controls have this property too). Usually, we set items as an simple single layer array (like "[ia’, ‘ib’, ‘ic’]"). Framewok will convert this simple array to inner format:

```
[
   {
       id : "ia",
      caption : "ia"
  },
   {
       id : "ib",
      caption : "ib"
  },
   {
       id : "ic",
      caption : "ic"
  }
]
```

1) combobox: Not readonly. The pop List shows “caption”; Input box shows “caption”; getValue returns “caption”.
2) listbox: Readonly. The pop List shows “caption”; Input box shows “caption”; getValue

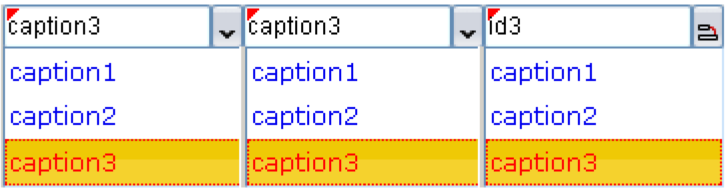
returns “id.

3) helpinput: Not readonly. The pop List shows “caption”; Input box shows “id”; getValue returns “id”.

Input:

```
var items=[
   {
      id : "id1",
      caption : "caption1"
   },{
      id : "id2",
      caption : "caption2"
   },{
      id : "id3",
      caption : "caption3"
   }
];
xui.create('ComboInput',{position:'relative',items:items}).show();
xui.create('ComboInput',{position:'relative',items:items,type:'listbox'}).show();
xui.create('ComboInput',{position:'relative',items:items,type:'helpinput'}).show();
```

Output:



3.6.9.3. Date Piker

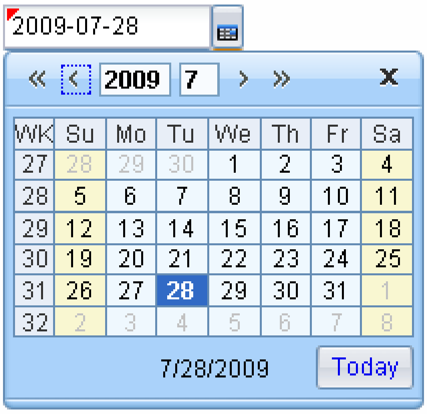
Sets type property to “date”.

Input:

```
var ctrl=xui.create('ComboInput')
.setType('date')
.setValue(new Date)
.show();

_.asyRun(function(){
      alert("The value is a timestamp string:"+ctrl.getValue());
              alert("You can convert it to date object:"+new Date(parseInt(ctrl.getValue())));
});
```

Date object or timestamp string

Output:



2009-07-28

WK	Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	1	2	3	4
28	5	6	7	8	9	10	11
29	12	13	14	15	16	17	18
30	19	20	21	22	23	24	25
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8

7/28/2009

3.6.9.4. Time Picker

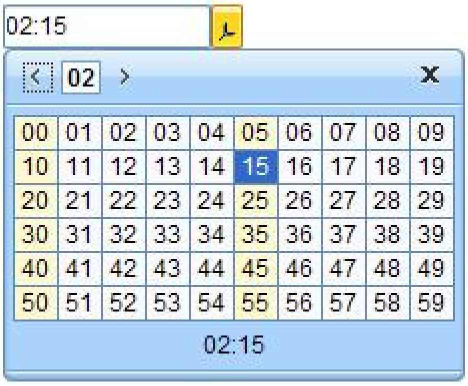
Sets type property to “time”.

Input:

```
var ctrl=xui.create('ComboInput')
.setType('time')
.setValue('2:15')
.show();

xui.alert("The value is a string : "+ctrl.getValue());
```

Sets string

Output:



02:15

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

02:15

3.6.9.5. Color Picker

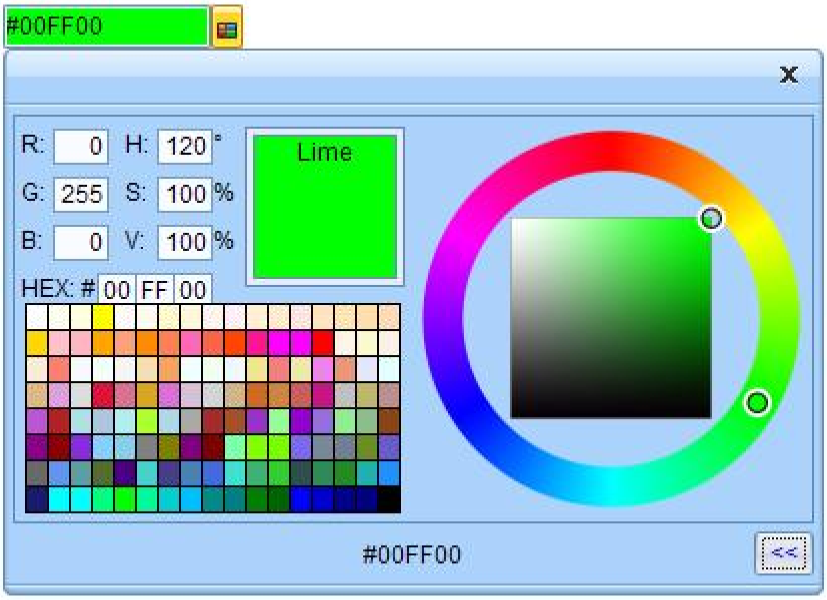
Sets type property to “color”.

Input:

```
var ctrl=xui.create('ComboInput')
.setType('color')
.setValue('#00ff00')
.show();

xui.alert("The value is a string : "+ctrl.getValue());
```

Sets string

Output:



#00FF00

R: 0 H: 120 °

G: 255 S: 100 %

B: 0 V: 100 %

HEX: # 00 FF 00

Lime

#00FF00

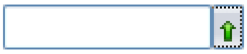
3.6.9.6. File Picker

Sets type property to “upload”.

Input:

```
var ctrl=xui.create('ComboInput')
.setType('file')
.show();
```

Output:



Note: use getUploadObj function to get the file’s handler

```
ctrl.getUploadObj()
```

3.6.9.7. Getter

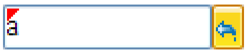
Sets type property to “getter”.

Input:

```
var ctrl=xui.create('ComboInput')
.setType('getter')
.beforeComboPop(function(profile){
        profile.boxing().setUIValue(_.id())
})
.show();
```

Sets value in beforeComboPop event

Output:



3.6.9.8. Custom Pop Window

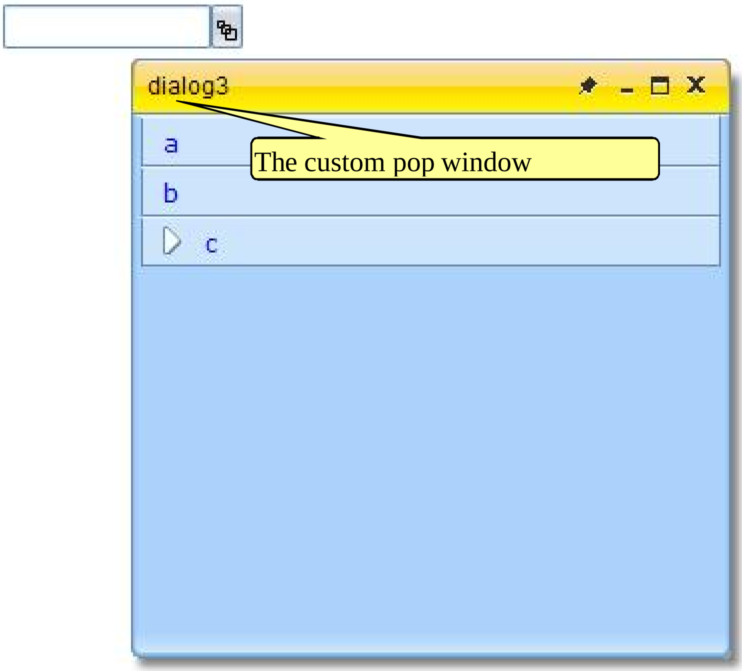
Sets type property to “cmdbox”, or “popbox”.

Input:

```
var ctrl=xui.create('ComboInput')
.setType('popbox')
.beforeComboPop(function(profile){
        var dlg=new xui.UI.Dialog, tb;
        dlg.append(tb=new xui.UI.TreeBar({items:["a","b",{id:"c",sub:["c1","c2","c3"]}]}));
        tb.onItemSelected(function(profile,item){
             ctrl.setUIValue(item.id);
             dlg.destroy();
        });
        dlg.show(null,true,100,100)
})
.show();
```

Shows custom pop window in beforeComboPop event

Output:



The custom pop window

3.6.9.9. Command Buttons

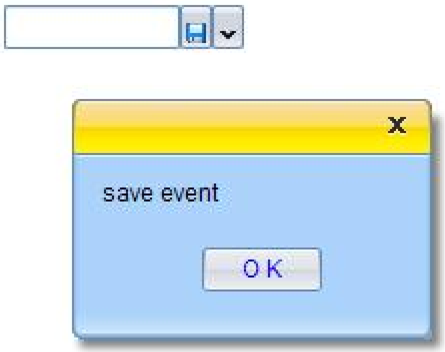
You can use commandBtn property to add an command button into ComboInput control. The following types are available for commandBtn property:

- "none": no command button
- "save": It's a save button
- "add" : It's a add button
- "remove" : It's a remove button
- "delete" : It's a delete button
- "custom" : custom button (sets imageClass or mage,/imagePos to custom it)

Input:

```
(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('none').show();
(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('save').onCommand(function(){ xui.alert('save
event'); }).show();
(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('add').onCommand(function(){ xui.alert('add
event'); }).show();
(new
xui.UI.ComboInput).setPosition('relative').setCommandBtn('remove').onCommand(function(){ xui.alert('remove
event'); }).show();
(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('delete').onCommand(function(){ xui.alert('delete
event'); }).show();
(new xui.UI.ComboInput).setPosition('relative').setCommandBtn('save').onCommand(function(){ xui.alert('save
event'); }).show();
```

Output:



save event

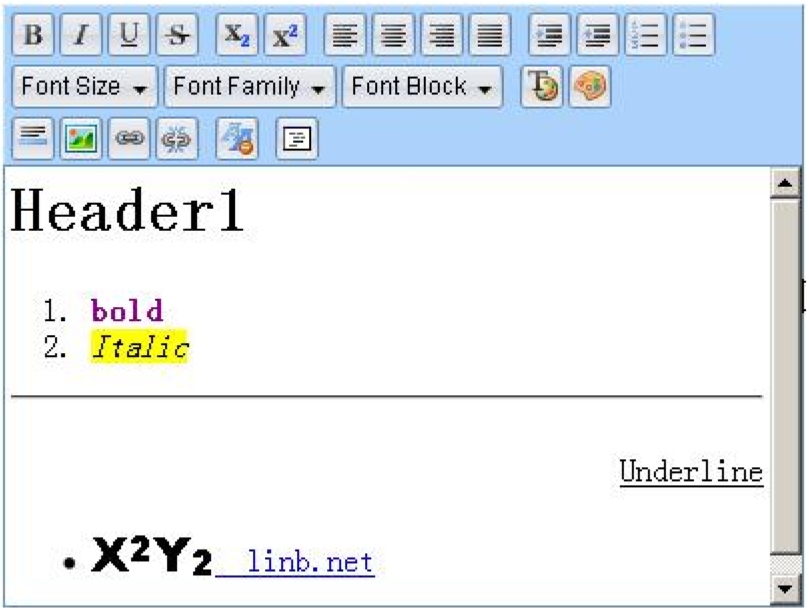
NOTE
chapter2\ComboInput\index.html is an overall example for ComboInput.

3.6.10. RichEditor

Input:

```
(new xui.UI.RichEditor()).show();
```

Output:



Header1

1. bold
2. Italic

Underline

- X^2Y_2 linb.net

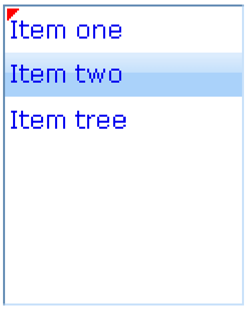
3.7. List related

This section relates to the following controls: xui.UI.List, xui.UI.RadioBox and xui.UIIconList and xui.UI.Gallery.

3.7.1. A Simple one

```
xui.create("List")
.setItems(["Item one","Item two","Item tree"])
.onItemSelected(function(profile,item){
        xui.message(item.id);
})
.show();
```

onItemSelected event



Item one
Item two
Item tree

3.7.2. A little bit complicated

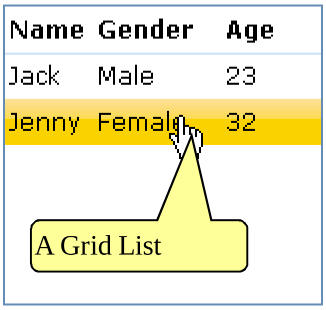
```
var renderer=function(o){
      return '<span style="width:40px">'+o.col1+"</span>" + ' <span style="width:60px">'+o.col2+"</span>" +
' <span style="width:40px">'+o.col3+"</span>";
};

xui.create("List")
.setWidth(160)
.setItems([{
            id:"a",
            col1:'Name',
            col2:'Gender ',
            col3:'Age',
            renderer:renderer,
            itemStyle:'border-bottom:solid 1px #C8E1FA;font-weight:bold;'
      },
      {
            id:"b",
            col1:'Jack',
            col2:'Male',
            col3:'23',
            renderer:renderer
      },
      {
            id:"c",
            col1:' Jenny',
            col2:'Female',
            col3:'32',
            renderer:renderer
      }]
)
.beforeUIValueSet(function (profile, ov, nv){
      return nv!=="a"
})
.show();
```

Gives a render function

Extra variables

For the header item

Result:

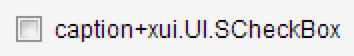


Name	Gender	Age
Jack	Male	23
Jenny	Female	32

A Grid List

The above special render function applies to any control's caption property (e.g. xui.UI.Button, xui.UI.Label); and any control's sub item caption property (e.g. xui.UI.List, xui.UI.TreeBar) .

```
xui.create("SCheckBox")
.setCaption("caption")
.setRenderer(function(prop){return prop.caption+"+"+this.key})
.show();
```



□ caption+xui.UI.SCheckBox

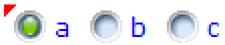
3.7.3. RadioBox

xui.UI.RadioBox is derived from xui.UI.List.

Input:

```
xui.create("RadioBox")
.setItems(["a","b","c"])
.onItemSelected(function(profile,item){
        xui.message(item.id);
})
.show();
```

Output:



● a ○ b ○ c

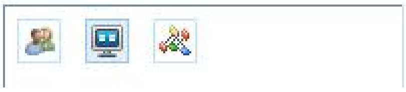
3.7.4. IconList and Gallery

Both are derived from xui.UI.List.

Input:

```
xui.create("IconList")
.setItems([{id:'a',image:'img/a.gif'},{id:'b',image:'img/b.gif'},{id:'c',image:'img/c.gif'}])
.onItemSelected(function(profile,item){
        xui.message(item.id);
})
.show();
```

Output:

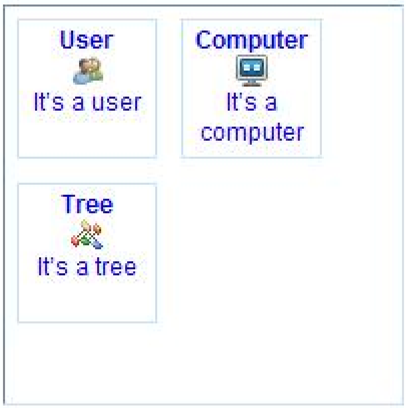


Input:

```
xui.create("Gallery")
.setItemWidth(64).setItemHeight(64)
.setItems([{id:'a',image:'img/a.gif',caption:'User',comment:'It's a
user'},{id:'b',image:'img/b.gif',caption:'Computer',comment:'It's a
computer'},{id:'c',image:'img/c.gif',caption:'Tree',comment:'It's a tree'}])
.onItemSelected(function(profile,item){
        xui.message(item.id);
})
.show();
```

Item's height

Output:



3.7.5. Item selection

You can use "**setUIValue**" function to select a item in List, or use "**fireItemClickEvent**" function to get the same result. "fireItemClickEvent" function will trigger "onItemSelected" event, "setUIValue" won't.

```
var ctrl=xui.create("List")
.setItems(["Item one","Item two","Item tree"])
.onItemSelected(function(profile,item){
        xui.message(item.id);
})
.show();

_.asyRun(function(){
        ctrl.fireItemClickEvent("Item two");
},1000);

_.asyRun(function(){
                ctrl.setUIValue("Item one");
},2000);
```

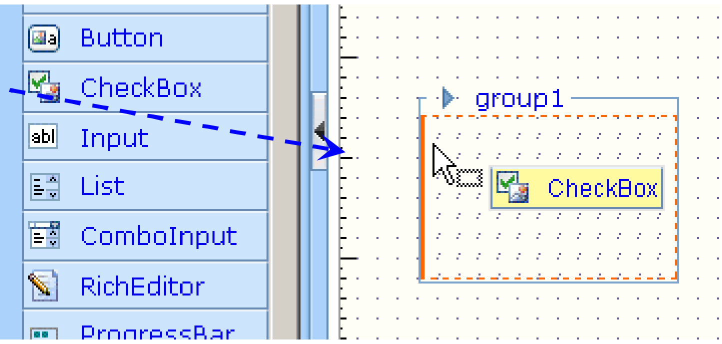
Trigger onItemSelected event

3.7.6. Container related

This section relates to the following controls: xui.UI.Group, xui.UI.Pane , xui.UI.Panel, xui.UI.Block.

xui.UI.Dialog, xui.UI.Layout and xui.UI.Tabs /Stacks/ButtonViews are container controls too, we will give examples of these controls in separate sections.

Container is those controls that can have child controls. In CrossUI RAD Designer, you can drag a child control and drop it into a container control. Just like this,



Input 1:

```
(new xui.UI.Group)
.append(new xui.UI.SButton)
.show();
```

append

Input 2:

```
var con = new xui.UI.Group;
con.show();
(new xui.UI.SButton).show(con);
```

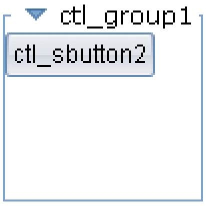
show

Input 3:

```
xui.create({
      key:"xui.UI.Group",
      children:[[{key:"xui.UI.SButton"}]]
}).show();
```

In children object

Output:



3.7.7. Pane and Panel

xui.UI.Pane is a single node control. It's derived from xui.UI.Div. xui.UI.Panel has a border and a title bar.

Input:

```
(new xui.UI.Pane)
.append(new xui.UI.SButton)
.show()
```

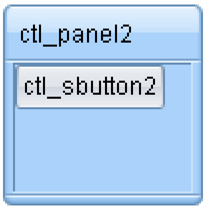
You can’t see output, It’s transparent

Input:

```
(new xui.UI.Panel)
.setDock("none")
.append(new xui.UI.SButton)
.show()
```

Sets dock to ‘none’

Output:

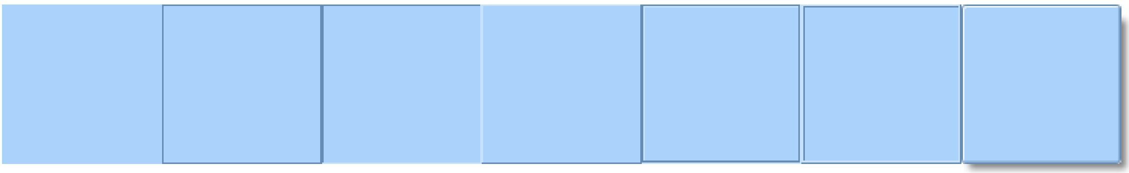


3.7.8. Block

Input:

```
xui.create("Block",{position:'relative',borderType:'none'}).show()
xui.create("Block",{position:'relative',borderType:'flat'}).show()
xui.create("Block",{position:'relative',borderType:'inset'}).show()
xui.create("Block",{position:'relative',borderType:'outset'}).show()
xui.create("Block",{position:'relative',borderType:'groove'}).show()
xui.create("Block",{position:'relative',borderType:'ridge'}).show()
xui.create("Block",{position:'relative',borderType:'none',border:true,shadow:true,resizer:true}).show()
```

Output:



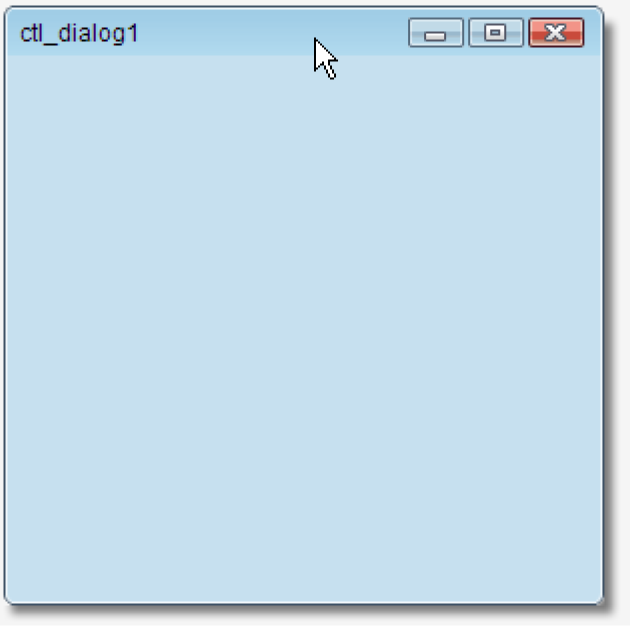
3.8. Dialog related

3.8.1. Normal state

Input:

```
(new xui.UI.Dialog).show()
```

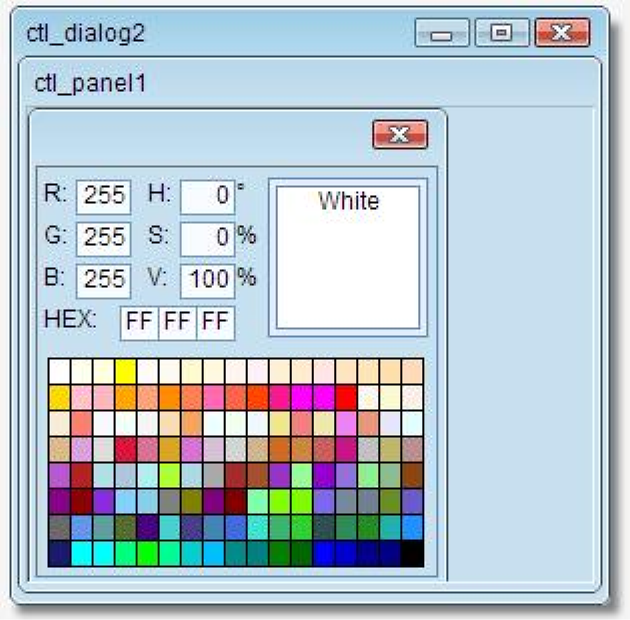
Output:



Input:

```
var dlg = (new xui.UI.Dialog).show();
var panel;
_.asyRun(function(){
    dlg.append(panel=new xui.UI.Panel)
},1000);
_.asyRun(function(){
    panel.append(new xui.UI.ColorPicker)
},2000);
```

Output:

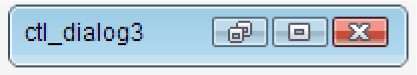


3.8.2. Min and Max status

Input:

```
var dlg = (new xui.UI.Dialog). setStatus("min").show();
_.asyRun(function(){
   dlg.setStatus("normal");
},1000);
_.asyRun(function(){
   dlg.setStatus("max");
},2000);
```

Output:



3.8.3. Modal Mode

Input:

```
var dlg = (new xui.UI.Dialog).show();
dlg.append(panel=new xui.UI.SButton({
     caption: "Pop a modal dialog"
},
{onClick:function(){
        xui.create("Dialog",{
                width:200,
                height:100,
                html:"The second modal dialog"
        }).showModal(dlg);
}}
))

(new xui.UI.Dialog)
.setHtml("The first modal dialog")
.show(null,true);
```

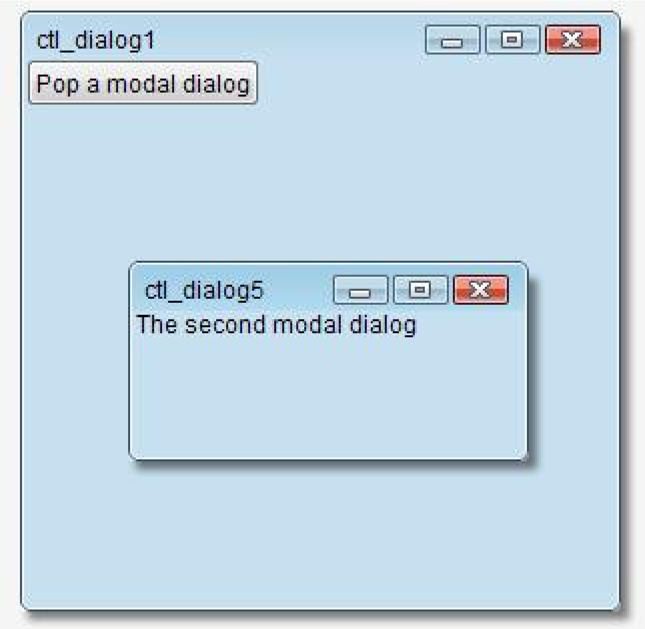
Sets caption

onClick event

Parent is dlg

Parent is html body

Output:



3.9. Layout Control

Input:

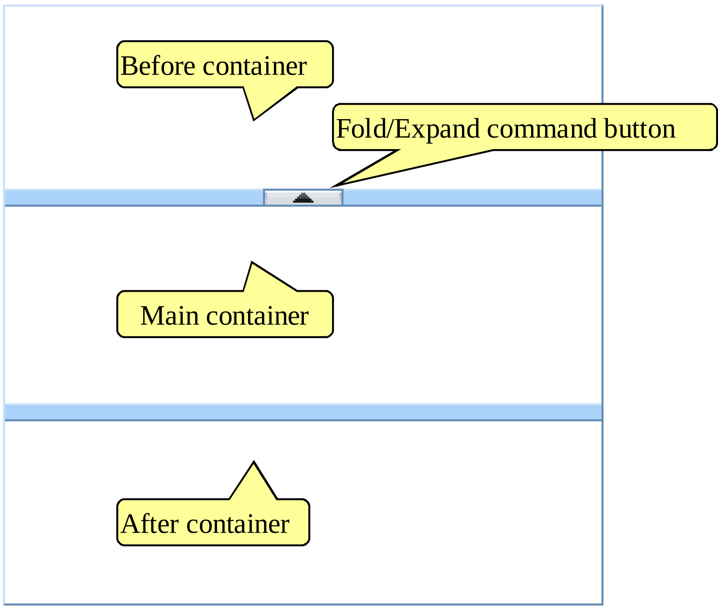
```
var block=xui.create("Block").setWidth(300).setHeight(300);
var layout=xui.create("Layout",{items:[
        {id:'before',
          pos:'before',
          size:100,
          cmd:true
        },{id:'after',pos:'after',size:100}
]});
block.append(layout).show();
```

Append into a block

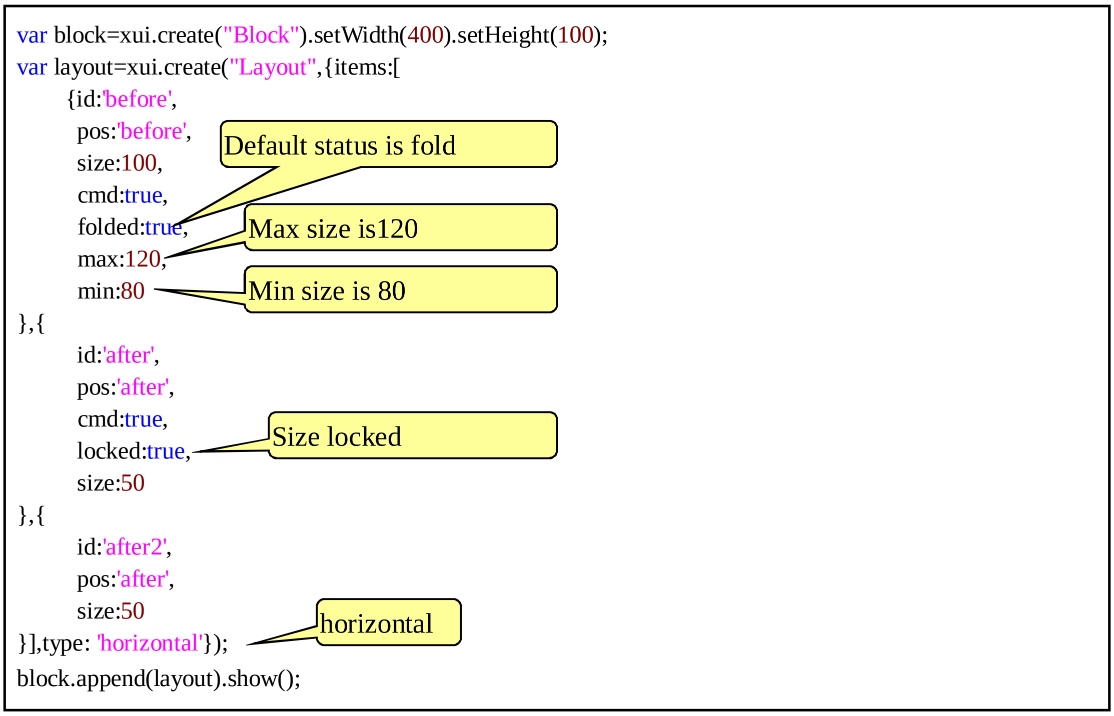
Size to 100

Has a command button

Output:

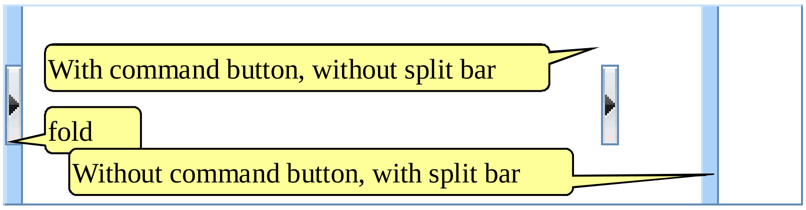


Input:



```
var block=xui.create("Block").setWidth(400).setHeight(100);
var layout=xui.create("Layout",{items:[
        {id:'before',
          pos:'before',
          size:100,
          cmd:true,
          folded:true,
          max:120,
          min:80
},{
          id:'after',
          pos:'after',
          cmd:true,
          locked:true,
          size:50
},{
          id:'after2',
          pos:'after',
          size:50
}],type: 'horizontal'});
block.append(layout).show();
```

Output:



With command button, without split bar
fold
Without command button, with split bar

NOTE

chapter2\Layout\index.html is an overall example for Layout.

3.10. Multi-pages Controls

Three multi-pages controls: xui.UI.Tabs, xui.UI.Stacks and xui.UI.ButtonViews.

Input:

```
var block=xui.create("Block").setWidth(400).setHeight(100);
var pages=xui.create("Tabs",{
        items:["page1","page2","page3"],
        value:"page2"
});
block.append(pages).show();

_.asyRun(function(){
        pages.append(new xui.UI.SButton,"page2")
},1000);
```

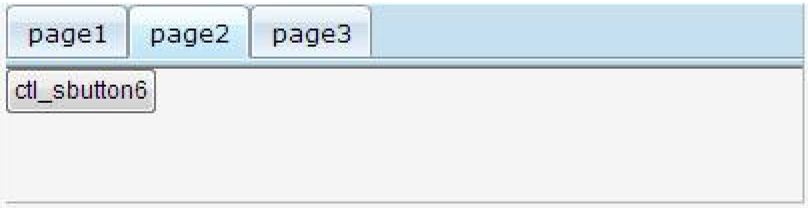
3 pages

Append to a block

The default page

Adds a SButton to 2th page

Output:



3.10.1. noPanel property

For xui.UI.Tabs and xui.UI.ButtonViews, when “noPanel” property was set to true, they no longer are the container control. So, don’t append any children control to tabs in this case.

Input:

```
var block=xui.create("Block").setWidth(400).setHeight(300).show();
var items=["page1","page2","page3"];
xui.create("Tabs",{
        items:items,
        value:"page2",
        position:'relative',
        width:'auto',
        height:'auto',
        dock:'none',
        noPanel:true
}).show(block);

xui.create("ButtonViews",{
        items:items,
        value:"page2",
        position:'relative',
        width:'auto',
        height:32,
        barSize:30,
        dock:'none',
        noPanel:true
}).show(block);
```

Set position to ‘relative’

Auto width

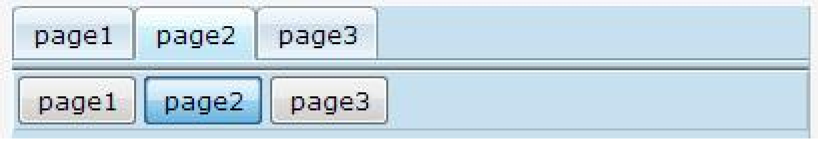
Auto height

No container

Set height to buttonview

No container

Output:

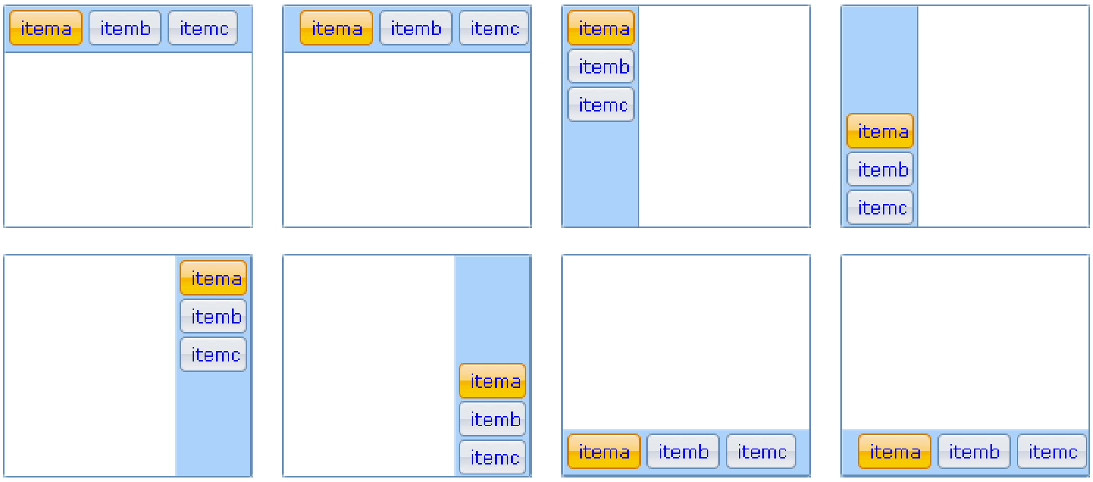


3.10.2. ButtonViews types

There are three properties used to define the ButtonViews' layout:

- **barLocation**: Used to set the location of the command button bar.
  In 'top','bottom','left','right'.
- **barHAlign**: Used to set command buttons horizontal alignment
  In 'left', 'right'. Only for **barLocation** is 'top' or 'bottom'
- **barVAlign**: Used to set command buttons vertical alignment
  In 'left', 'right'. Only for **barLocation** is 'left' or 'right'

The below picture shows all the eight possible ButtonViews layouts:



In **chapter2\ButtonViews\index.html**

NOTE
chapter2\ButtonViews\index.html is an overall example for ButtonViews.

3.10.3. Page selection

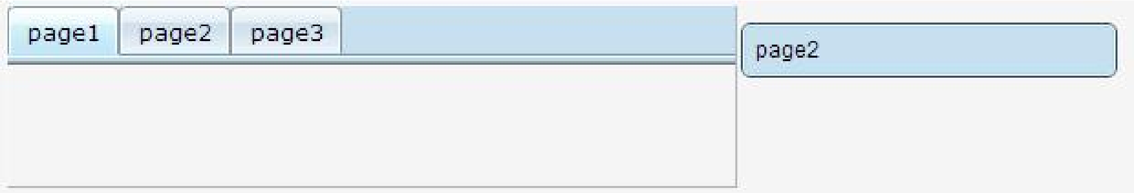
You can use "**setUIValue**" function to select a page, or use "**fireItemClickEvent**" function to get the same result. "fireItemClickEvent" function will trigger "onItemSelected" event, "setUIValue" won't.

Input:

```
var block=xui.create("Block").setWidth(400).setHeight(100).show();
var pages=xui.create("Tabs",{
        items:["page1","page2","page3"]
})
.onItemSelected(function(profile,item){
        xui.message(item.id);
})
.show(block);

_.asyRun(function(){
        pages.fireItemClickEvent("page2");
},1000);

_.asyRun(function(){
        pages.setUIValue("page1");
},2000);
```

Trigger onItemSelected event

Output:



3.10.4. Pages

3.10.4.1. Close and options Button

Each page can hold a “close” button and a “options” button. Click this button will close the page.

Input:

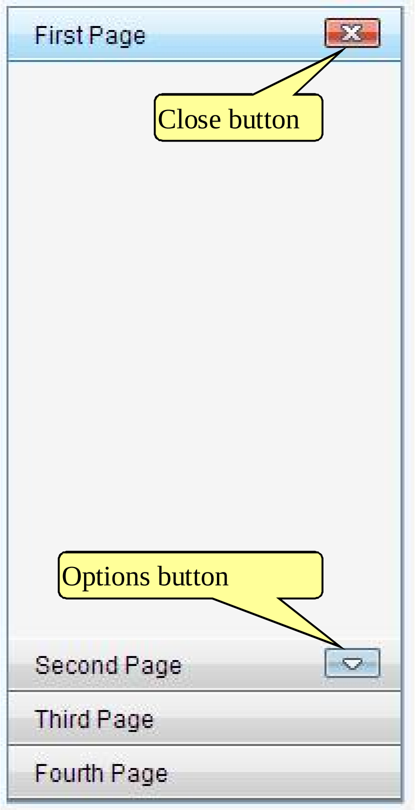
```
var block=xui.create("Block").setWidth(200).setHeight(400).show(), stacks;
block.append(stacks=new xui.UI.Stacks({
        value:'a',
        items:[{
                id:'a',
                caption:'First Page',
                closeBtn:true
        },{
                id:'b',
                caption:'Second Page',
                optBtn:true
        },{
                id:'c',
                caption:'Third Page'
        },{
                id:'d',
                caption: 'Fourth Page'
        }]
}));
stacks.onShowOptions(function(profile,item){
        xui.message(" You clicked "+item.caption)
});
```

Close button

Options button

Click options to trigger onShowOptions event

Output:



Close button

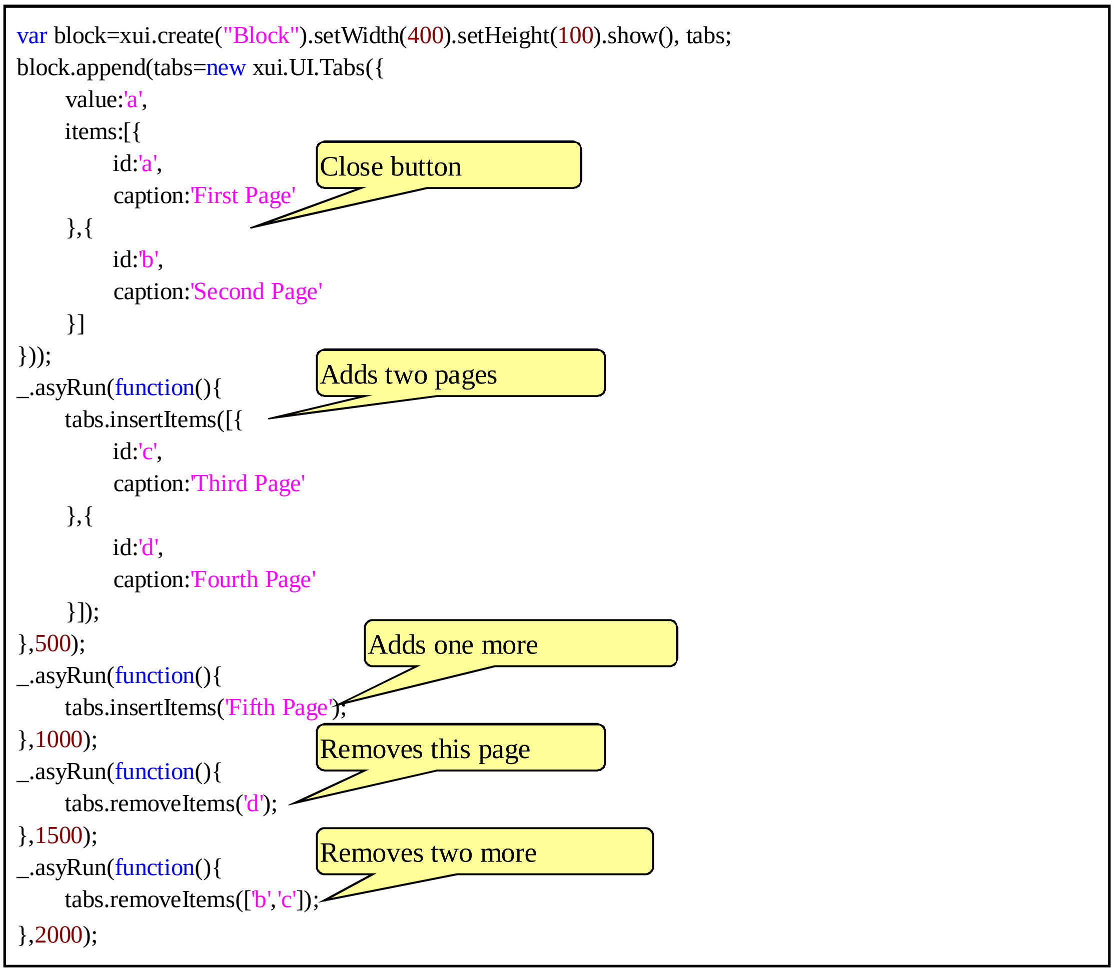
Options button

Two events can be fired when “close” button was clicked:

- beforePageClose: Fired before user clicked the close button on a page. If returns false, the page won’t be closed.
- afterPageClose: Fired after user clicked the close button on a page.

3.10.4.2. Add/Remove Pages

Input:



```
var block=xui.create("Block").setWidth(400).setHeight(100).show(), tabs;
block.append(tabs=new xui.UI.Tabs({
    value:'a',
    items:[{
        id:'a',
        caption:'First Page'
    },{
        id:'b',
        caption:'Second Page'
    }]
}));
_.asyRun(function(){
    tabs.insertItems([{
        id:'c',
        caption:'Third Page'
    },{
        id:'d',
        caption:'Fourth Page'
    }]);
},500);
_.asyRun(function(){
    tabs.insertItems('Fifth Page');
},1000);
_.asyRun(function(){
    tabs.removeItems('d');
},1500);
_.asyRun(function(){
    tabs.removeItems(['b','c']);
},2000);
```

Close button

Adds two pages

Adds one more

Removes this page

Removes two more

3.10.5. Dynamic content loading

3.10.5.1. onIniPanelView

```
    var block=xui.create("Block").setWidth(400).setHeight(100).show(),
    tabs=new xui.UI.Tabs({
        value:'a',
        items:[{
            id:'a',
            caption:'First Page'
        },{
            id:'b',
            caption:'Second Page'
        },{
            id:'c',
            caption:'Third Page'
        }]
    });
    tabs.onIniPanelView(function(profile,item){
        profile.boxing().getPanel(item.id).append(new xui.UI.SButton)
    });
block.append(tabs);
```

3.10.5.2. beforeUIValueSet/afterUIValueSet

It’s a fine-grained mechanism.

```
var block=xui.create("Block").setWidth(400).setHeight(100).show(), tabs;
block.append(tabs=new xui.UI.Tabs({
    value:'a',
    items:[{
        id:'a',
        caption:'First Page'
    },{
        id:'b',
        caption:'Second Page'
    },{
        id:'c',
        caption:'Third Page'
    }]
}));
tabs.beforeUIValueSet(function(profile,ovalue,value){
    if(value=='b')
        return false;
});
tabs.afterUIValueSet(function(profile,ovalue,value){
    if(value=='c'){
        var item=profile.getItemByItemId(value);
        if(!item.$ini){
            profile.boxing().append(new xui.UI.SButton);
            item.$ini=true;
        }
    }
});
```

Cancel selection

Gets item object

Checks flag

Sets a flag

3.11. Menus and toolbars

3.11.1. Pop Menu

Input:

```
var pm=xui.create('PopMenu')
.setItems([
        {"id":"itema", "caption":"itema", "tips":"item a"},
        {"type":"split"},
        {"id":"itemb", "type":"checkbox", value:true,"caption":"itemb", "tips":"item b"},
        {"id":"itemc", "caption":"itemc", "type":"checkbox", "tips":"item c"},
        {"id":"itemd", "caption":"itemd", "tips":"item d",sub:[
                {"id":"itemd1", "caption":"itemd1"},
                {"id":"itemd2", "caption":"itemd2"}
]},
        {"id":"iteme", "caption":"iteme", "tips":"item d", disabled:true}
])
.onMenuSelected(function(profile,item){
        xui.message(item.id + (item.type=="checkbox"?" : " + item.value:""))
});

xui.create('SButton')
.onClick(function(profile){
        pm.pop(profile.getRoot())
})
.show();
```

Checkbox type

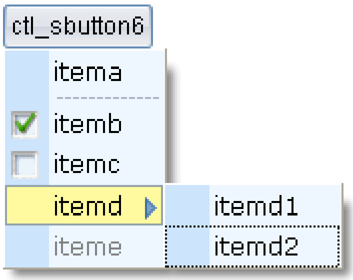
Sub pop menu

event

Disabled it

For position

Output:



3.11.2. MenuBar

Input:

```
var pm=xui.create('MenuBar')
.setItems([{
        "id" : "file", "caption" : "File",
        "sub" : [{
                "id" : "newproject",
                "caption" : "New Project"
        },
        {       "id" : "openproject", "caption" : "Open Project",
                "add" : "Ctrl+Alt+O",
                "image" : "img/b.gif",
                "sub":["option 1","option 2"]
        },
        {       "id" : "closeproject",    "caption" : "Close Project"
        },
        {"type" : "split"},
        {       "id" : "save", "caption" : "Save",
                "image" : "img/a.gif"
        },
        {       "id" : "saveall",    "caption" : "Save All",
                "add" : "Ctrl+Alt+S",
                "image" : "img/c.gif"
        }]
},
{ "id" : "tools", "caption" : "Tools",
        "sub" : [{      "id" : "command",    "caption" : "Command Window"
        },
        {       "id" : "spy", "caption" : "Components Spy"
        }]
},
{       "id" : "build", "caption" : "Build",
        disabled:true,
        "sub" : [{      "id" : "debug",
                "caption" : "Debug"
        }]
}]).show()
```

Pop menu data

Extra data

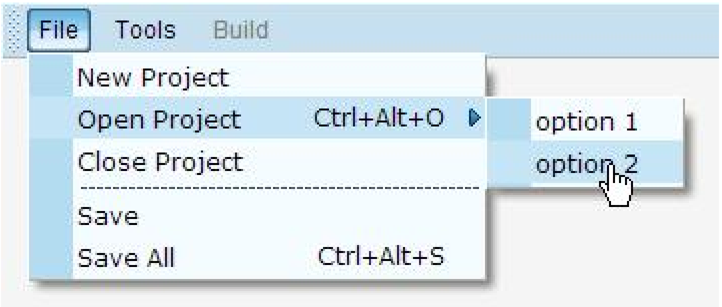
Sub pop menu

A split

An icon

Disabled it

Output:



3.11.3. Toolbars

Input:

```
xui.create('ToolBar',{items:[{
        "id" : "align",
        "sub" : [
                {"id" : "left","caption" : "left"},
                {"id" : "center","caption" : "center"},
                {type:'split'},
                {"id" : "right","caption" : "center"}
        ]
    },{
        "id" : "code",
        "sub" : [{
                "id" : "format","caption" : "format",
                label:"label",
                image:"img/a.gif",
                "dropButton" : true
        }]
    }]
})
.onClick(function(profile,group,item){
        xui.message(group.id + " : " +item.id)
})
.show();
```

Button group data

Button data

A split

With a label

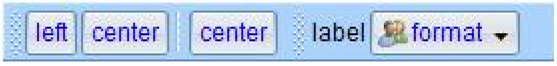
With an icon

A drop button

Group object

Button

Output:



3.12. TreeBar and TreeView

3.12.1. Three selection mode

All controls derived from xui.UI.absList have three options mode.

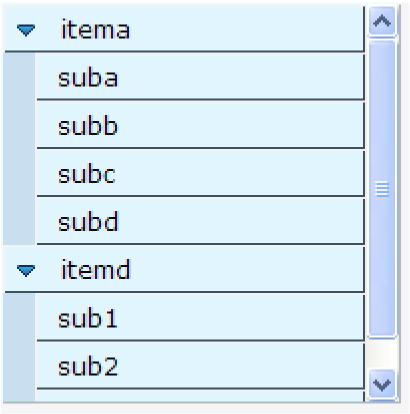
3.12.1.1. No-selection

Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{items:[{ id : "itema", sub : ["suba","subb","subc","subd"]},
{id : "itemd", sub : ["sub1","sub2","sub3"]}]})
.setSelMode("none")
.onItemSelected(function(profile,item){
      xui.message(item.id);
}).show(block);
```

Sets to ‘none’

Output:



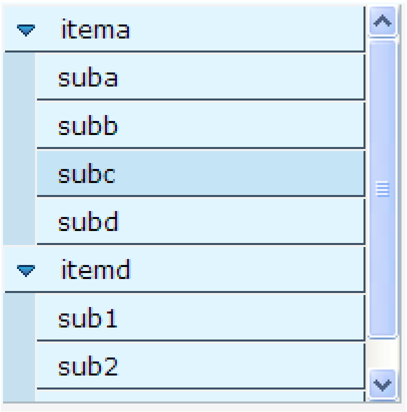
3.12.1.2. Single-selection

Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{items:[{ id : "itema", sub : ["suba","subb","subc","subd"]},
{id : "itemd", sub : ["sub1","sub2","sub3"]}]})
.setSelMode("single")
.onItemSelected(function(profile,item){
    xui.message(item.id);
}).show(block);
```

Sets to single

Output:



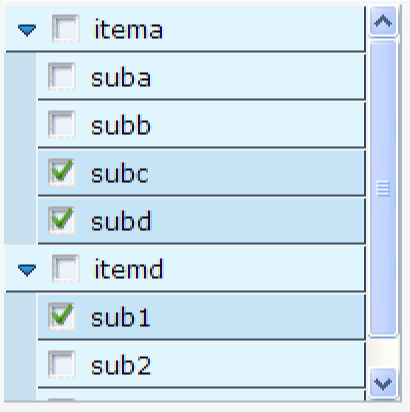
3.12.1.3. Multi-selection

Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{items:[{ id : "itema", sub : ["suba","subb","subc","subd"]},
{id : "itemd", sub : ["sub1","sub2","sub3"]}]})
.setSelMode("multi")
.onItemSelected(function(profile,item){
    xui.message(item.id);
}).show(block);
```

Sets to ‘multi’

Output:



3.12.2. Group Item

Input:

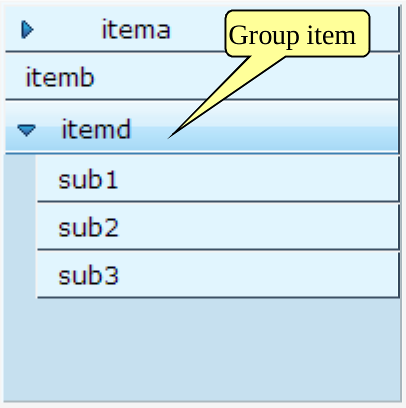
```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{items:[{
        id : "itema",
        image : "img/a.gif",
        sub : ["suba","subb","subc","subd"]
},
{id : "itemb"},
{
         id : "itemd",
        group:true,
        sub : ["sub1","sub2","sub3"]

}]}).show(block);
```

With an icon

Sub items

It's a group

Output:



Group item

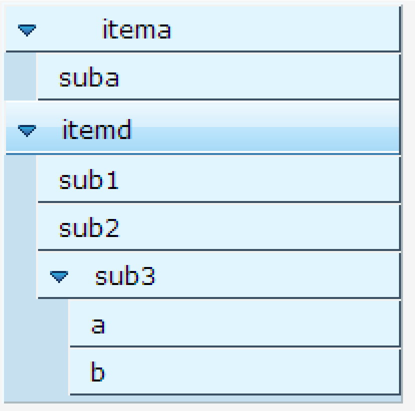
3.12.3. Expand all nodes by default

Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{
iniFold:false,
items:[{
        id : "itema",
        image : "img/a.gif",
        sub : ["suba"]
},
{
id : "itemd",
        group:true,
        sub : ["sub1","sub2",{
                id:"sub3",sub:["a","b"]
        }]
}]}).show(block);
```

Sets iniFold property to false

Output:



3.12.4. Mutex Expand

```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{
singleOpen:true,
items:[{
        id : "itema",
        image : "img/a.gif",
        sub : ["suba"]
},
{
id : "itemd",
        group:true,
        sub : ["sub1","sub2",{
                id:"sub3",sub:["a","b"]
        }]
}]}).show(block);
```

Mutex Expand

3.12.5. Dynamic Destruction

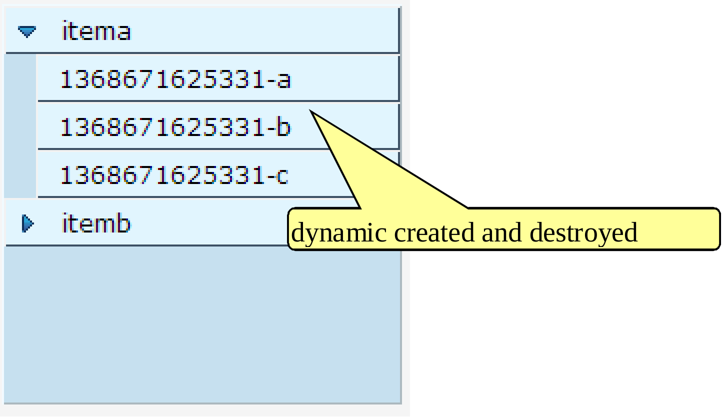
```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{
dynDestory:true,          Dynamic Destruction
items:[{
        id : "itema",
        image : "img/a.gif",
        sub : ["suba"]
},
{
id : "itemd",
        group:true,
        sub : ["sub1","sub2",{
                id:"sub3",sub:["a","b"]
        }]
}]}).show(block);
```

3.12.6. Dynamically loading

Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
xui.create("TreeBar",{
singleOpen:true,          Mutex Expand
dynDestory:true,          Dynamic Destruction
items:[{
        id : "itema",
        sub : true        Wants to load children dynamically
},
{
id : "itemb",
        sub : true
}]})
. onGetContent(function(profile,item,callback){
        if(item.id=="itema"){
                var rnd=_();          Takes time stamp as a random string
                callback([rnd+"-a",rnd+"-b",rnd+"-c"]);   Asynchronous or synchronous callback
        }
        if(item.id=="itemb")
         return ["itembsub1","itembsub2","itembsub3"];   Can also be returned directly
})
.show(block);
```

Output:



dynamic created and destroyed

3.13. TreeGrid

3.13.1. Header and Rows

The header property and rows property in TreeGrid are Array of key/value pairs, like,

```
[
      {id : "xxx1", caption : "xxx1" …,
       sub: []
      },
      {id : "xxx2", caption : "xxx2" …},
      …
]
```

[key/value pairs]

The sub [key/value pairs]

If no id specified, will create one automatically

It can be written as a simplified format,

```
[
      "xx1",
      "xx2",
      {
            id : "xxx3",
            sub: ["sub1","sub2"]
      }
]
```

Only id string

Only id string

When call setHeader/setRows, the simplified format can be convert to,

```
[
        {id:"xx1",caption:"xx1"},
        {id:"xx2",caption:"xx2"}
        {
                id : "xxx3", caption : "xxx3",
                sub: [
                        {id:"sub1",caption:"sub1"},
                        {id:"sub2",caption:"sub2"}
                ]
        }
]
```

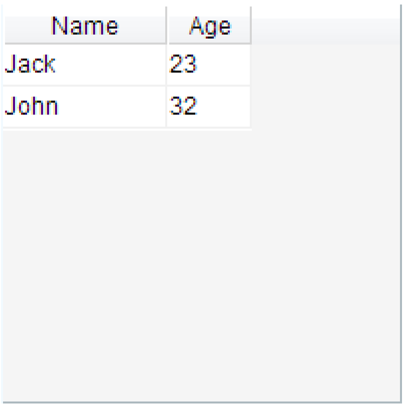
3.13.1.1. Sets standard format

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader([
        {id:"col1", caption:"Name"},
        {id:"col2", caption:"Age", width:40}
]).setRows([
        {id:"row1",cells:[{
                value:'Jack',caption:'Jack'
        },{
                value:23,caption:'23'
        }]},
        {id:"row2",cells:[{
                value:'John',caption:'John'
        },{
                value:32,caption:'32'
        }]}
]).show(block);
```

No row handler

Column's width

Cells data



Name	Age
Jack	23
John	32

3.13.1.2. Sets simplified format

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.show(block);
```

Only id input

Only value input

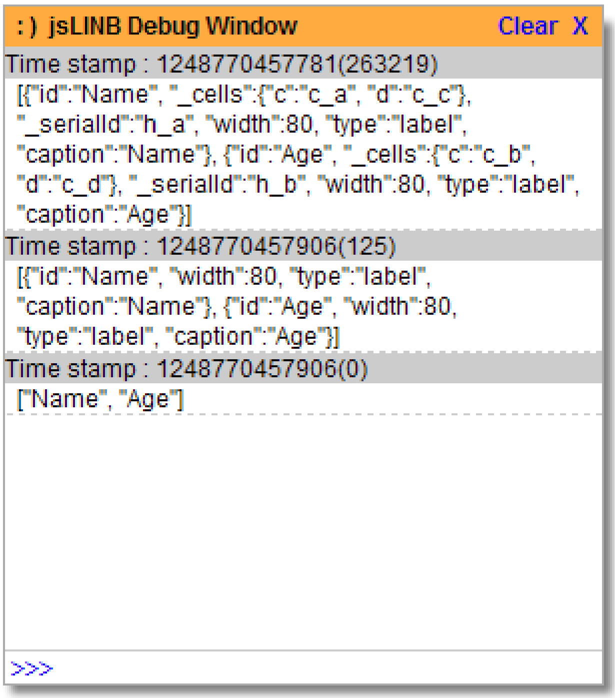
3.13.2. getHeader

Calls getHeader function to return the header data. There are three format,

- getHeader(): returns memory data;
- getHeader("data"): returns the standard format data;
- getHeader("min"): returns the simplified format data;

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.show(block);
xui.log(tg.getHeader());
xui.log(tg.getHeader("data"));
xui.log(tg.getHeader("min"));
```

Comparing these three formats



```
Time stamp : 1248770457781(263219)
[{"id":"Name", "_cells":{"c":"c_a", "d":"c_c"},
"_serialId":"h_a", "width":80, "type":"label",
"caption":"Name"}, {"id":"Age", "_cells":{"c":"c_b",
"d":"c_d"}, "_serialId":"h_b", "width":80, "type":"label",
"caption":"Age"}]
Time stamp : 1248770457906(125)
[{"id":"Name", "width":80, "type":"label",
"caption":"Name"}, {"id":"Age", "width":80,
"type":"label", "caption":"Age"}]
Time stamp : 1248770457906(0)
["Name", "Age"]
```

3.13.3. getRows

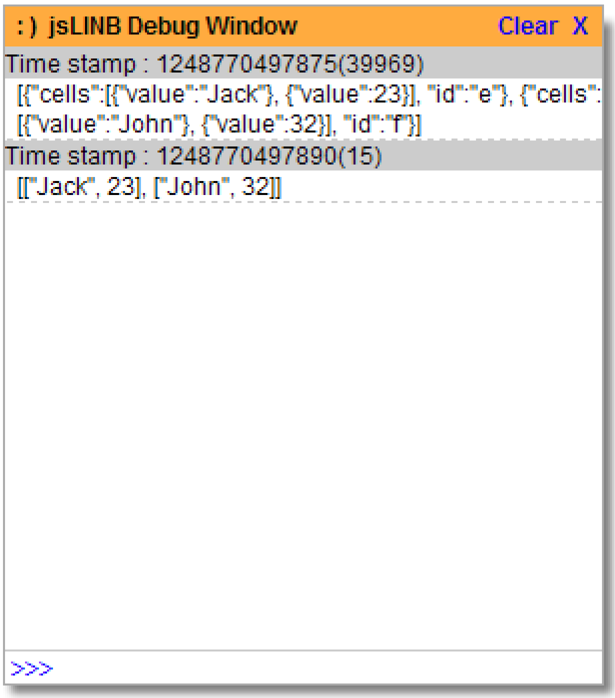
Calls getRows function to return the rows data. Similarly, there are three format,

- getRows (): returns memory data;
- getRows ("data"): returns the standard format data;
- getRows ("min"): returns the simplified format data;

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.show(block);
xui.log(tg.getRows("data"));
xui.log(tg.getRows("min"));
//console.log(tg.getRows());
```

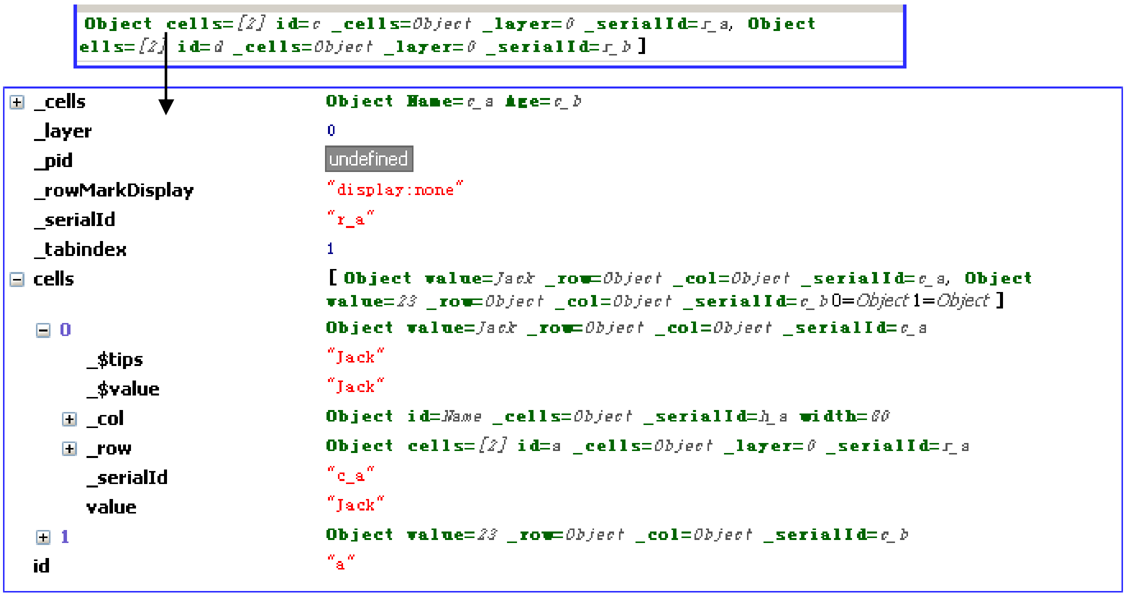
Comparing these three formats

There is circular reference in memory data, can‘t be directlv serialized



```
Time stamp : 1248770497875(39969)
 [{"cells":[{"value":"Jack"}, {"value":23}], "id":"e"}, {"cells":
 [{"value":"John"}, {"value":32}], "id":"f"}]
Time stamp : 1248770497890(15)
 [["Jack", 23], ["John", 32]]
```

The rows memory data in firebug:



```
Object cells=[2] id=c _cells=Object _layer=0 _serialId=r_a, Object
ells=[2] id=d _cells=Object _layer=0 _serialId=r_b ]

⊞ _cells                 Object Name=c_a Age=c_b
  _layer                 0
  _pid                   undefined
  _rowMarkDisplay        "display:none"
  _serialId              "r_a"
  _tabindex              1
⊟ cells                  [ Object value=Jack _row=Object _col=Object _serialId=c_a, Object
                         value=23 _row=Object _col=Object _serialId=c_b 0=Object 1=Object ]
  ⊟ 0                    Object value=Jack _row=Object _col=Object _serialId=c_a
      _$tips             "Jack"
      _$value            "Jack"
    ⊞ _col               Object id=Name _cells=Object _serialId=h_a width=80
    ⊞ _row               Object cells=[2] id=a _cells=Object _layer=0 _serialId=r_a
      _serialId          "c_a"
      value              "Jack"
  ⊞ 1                    Object value=23 _row=Object _col=Object _serialId=c_b
  id                     "a"
```

3.13.4. Active Modes

There are three active modes for TreeGrid:

- non-active appearance : activeMode is “none”;
- the row-active appearance: activeMode is “row” ;
- the cell-active appearance: activeMode is “cell”;

3.13.4.1. non-active appearance

Input:

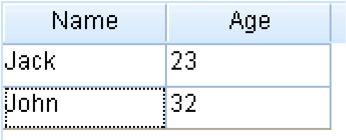
```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setActiveMode("none")
.show(block)


. afterRowActive (function(profile,row){
      xui.message(row.id);
})
. afterCellActive (function(profile,cell){
      xui.message(cell.value);
})
```

No row handler

Sets to ‘none’

Does not trigger events

Output:



Name	Age
Jack	23
John	32

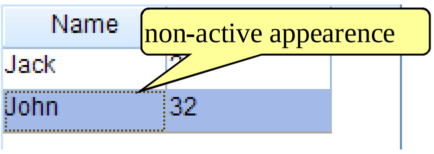
3.13.4.2. row-active appearance

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setActiveMode("row")
.show(block)

.afterRowActive (function(profile,row){
      xui.message(row.id);
})
.afterCellActive (function(profile,cell){
      xui.message(cell.value);
})
```

No row handler

Sets to “row”

Will be fired



Name	
Jack	2
John	32

non-active appearence

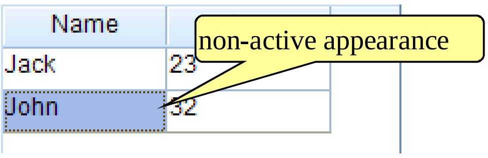
3.13.4.3. cell-active appearance

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setActiveMode("cell")
.show(block)

.afterRowActive (function(profile,row){
      xui.message(row.id);
})
.afterCellActive (function(profile,cell){
      xui.message(cell.value);
})
```

No row handler

Sets to “row”

Will be fired



Name	
Jack	23
John	32

non-active appearance

3.13.5. Selection Mode

There are five selection modes for TreeGrid:

- Non-selection: activeMode is “none”, or selMode is ‘none’
- Single row selection: activeMode is “row”, and selMode is ‘single’
- Multi-rows selection: activeMode is “row”, and selMode is ‘multi’
- Single cell selection: activeMode is “cell”, and selMode is ‘single’
- Multi-cells selection: activeMode is “cell”, and selMode is ‘multi’

3.13.5.1. Non-selection

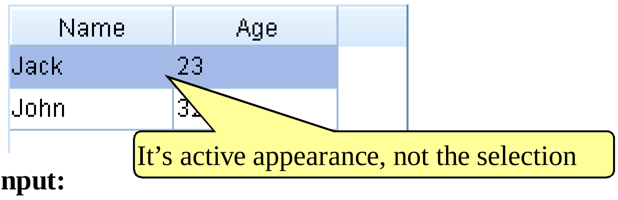
Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setSelMode("none")
.show(block)

.afterUIValueSet(function(profile,ovalue,value){
        xui.message(value);
});
```

Non-selection

Won’t be fired

Output:



Name	Age
Jack	23
John	32

It’s active appearance, not the selection

Input:

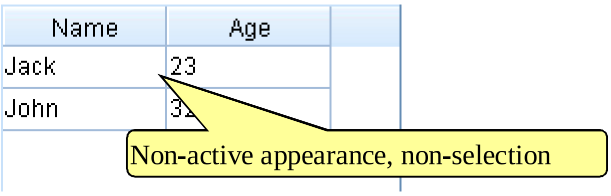
```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setActiveMode("none")
.setSelMode("none")
.show(block)

.afterUIValueSet(function(profile,ovalue,value){
       xui.message(value);
});
```

Non-active

Non-selection

Won’t be fired

Output:



Name	Age
Jack	23
John	32

Non-active appearance, non-selection

3.13.5.2. Single row selection

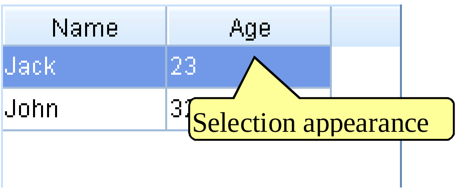
Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setSelMode("single")
.show(block)

.afterUIValueSet(function(profile,ovalue,value){
       xui.message(value);
});
```

Sets to ‘single’ mode

Will be fired

Output:



Name	Age
Jack	23
John	32

Selection appearance

3.13.5.3. Multi-row selection

Input:

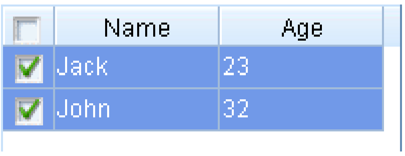
```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(24)
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setSelMode("multi")
.show(block)

.afterUIValueSet(function(profile,ovalue,value){
        xui.message(value);
});
```

Row handler's width

Sets to 'multi' mode

Will be fired

Output:



☐	Name	Age
☑	Jack	23
☑	John	32

3.13.5.4. Single cell selection

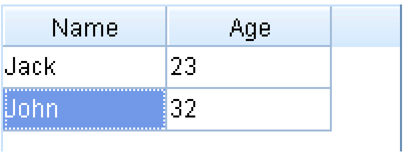
Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setActiveMode("cell")
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setSelMode("single")
.show(block)

.afterUIValueSet(function(profile,ovalue,value){
        xui.message(value);
});
```

Sets to 'cell' mode

Sets to 'single' mode

Output:



Name	Age
Jack	23
John	32

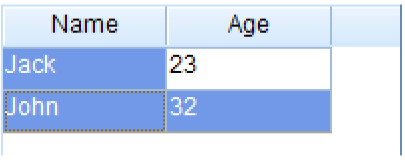
3.13.5.5. Multi-cells selection

Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setActiveMode("cell")
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])
.setSelMode("multi")
.show(block)

.afterUIValueSet(function(profile,ovalue,value){
        xui.message(value);
});
```

Sets to ‘multi’ mode

Output:



Name	Age
Jack	23
John	32

3.13.6. The Tree Grid

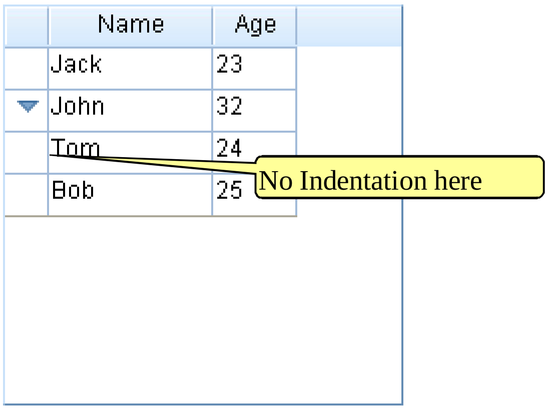
Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(20)
.setHeader([
        {id:"col1", caption:"Name"},
        {id:"col2", caption:"Age", width:40}
]).setRows([
        {id:"row1",cells:['Jack',23]},
        {id:"row2",cells:['John',32],
         sub:[{id:"row21",cells:['Tom',24]},
                {id:"row22",cells:['Bob',25]}
        ]}
]).show(block)
```

Row header is a must for tree grid

A row has sub rows

Output:



	Name	Age
	Jack	23
▼	John	32
	Tom	24
	Bob	25

No Indentation here

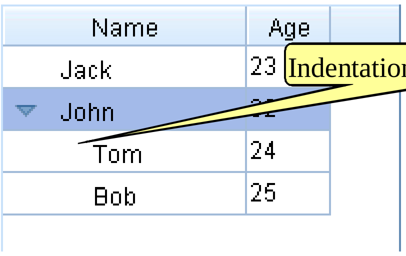
Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(20)
.setGridHandlerCaption("Name")
.setRowHandlerWidth(80)
.setHeader([
        {id:"col2", caption:"Age", width:40}
]).setRows([
        {id:"row1",caption: 'Jack',cells:[23]},
        {id:"row2",caption: 'John',cells:[32],
          sub:[{id:"row21",caption: 'Tom',cells:[24]},
                {id:"row22", caption: 'Bob',cells:[25]}
        ]}
]).show(block)
```

Grid handler caption

Row’s caption

Output:



Name	Age
Jack	23
John	32
Tom	24
Bob	25

Indentation in row handler

3.13.7. Column config

3.13.7.1. The first column

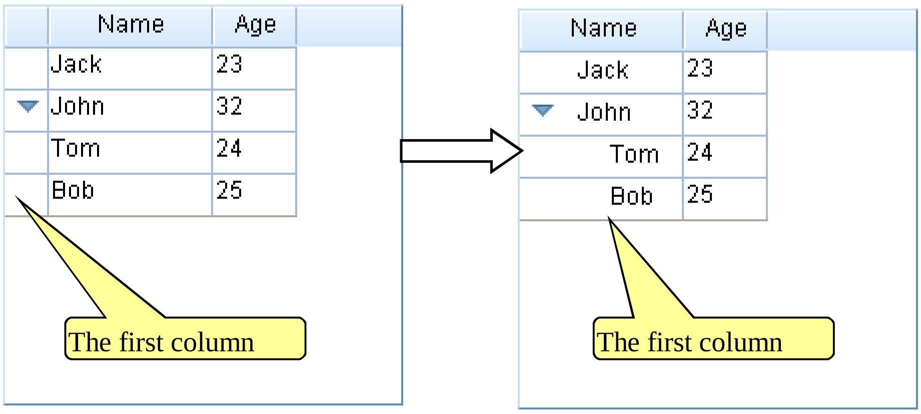
In order to show the first column, you have to set rowHandler to [true].

Input:

```
var block=new xui.UI.Block({width:200,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23]},
        {id:"row2",caption:'John',cells:[32],
          sub:[{id:"row21",caption:'Tom',cells:[24]},
                {id:"row22",caption:'Bob',cells:[25]}
        ]}
]).show(block)
```

Sets row handler’s width

Output:



Name	Age
Jack	23
John	32
Tom	24
Bob	25

Name	Age
Jack	23
John	32
Tom	24
Bob	25

上一节中缩进的例子

3.13.7.2. Column width

Input:

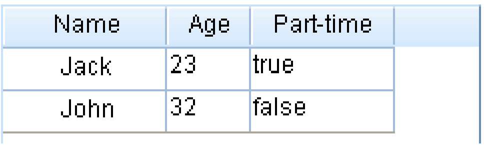
```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40},
        {id:"col2", caption:"Part-time", width:90}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false]}
]).show(block)
_.asyRun(function(){
        tg.updateHeader("col2", {width:70});
},1000);
```

The first column’s width

Column’s width

Modify column width dynamically

Output:



Name	Age	Part-time
Jack	23	true
John	32	false

3.13.7.3. Drag&Drop to modify column width

“ colResizer” property in TreeGrid determines whether the column width can be modified with Drag&Drop. Each column can include a “colResizer” property too. The “colResizer” property in column has higher priority than in TreeGrid.

In CrossUI, "fine-grained Setting has higher priority than coarse-grained" is a base rule.

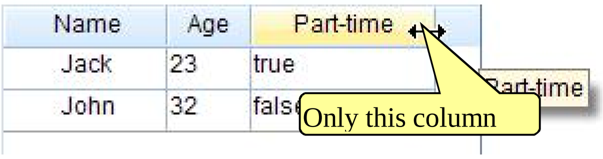
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColResizer(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40},
        {id:"col2", caption:"Part-time", width:90,colResizer:true}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false]}
]).show(block)
```

coarse-grained setting

fine-grained setting

Output:



Name	Age	Part-time
Jack	23	true
John	32	false

Only this column

3.13.7.4. Drag&Drop to modify column position

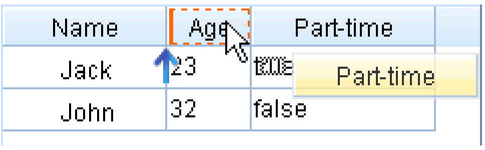
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColMovable(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40},
        {id:"col2", caption:"Part-time", width:90,colMovable:true}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false]}
]).show(block)
```

coarse-grained setting

fine-grained setting

Output:



Name	Age	Part-time
Jack	23	true
John	32	false

3.13.7.5. Default Sorting

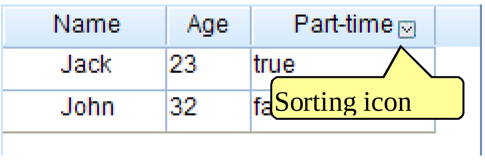
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColSortable(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40},
        {id:"col2", caption:"Part-time", width:90,colSortable:true}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false]}
]).show(block)
```

coarse-grained setting

fine-grained setting

Output:



Name	Age	Part-time
Jack	23	true
John	32	fa

3.13.7.6. Custom Sorting

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40},
        {id:"col2", caption:"Part-time", width:90,sortby:function(x,y){return -1}}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false]}
]).show(block)
```

Custom sorting function

3.13.7.7. Hide columns

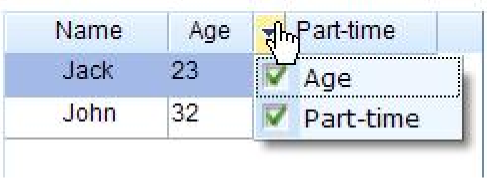
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColHidable (false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, colHidable:true },
        {id:"col2", caption:"Part-time", width:90, colHidable:true}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false]}
]).show(block)
```

Global setting

These 2 columns can be hidden

Output:



Name	Age	Part-time
Jack	23	
John	32	

3.13.7.8. Setting Cell Types in column header

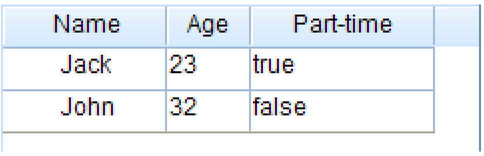
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColSortable(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Part-time", width:90, type: "checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false]}
]).show(block)
```

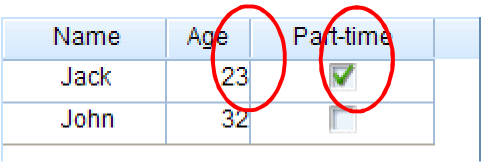
Number

Checkbox type

Output:



Name	Age	Part-time
Jack	23	true
John	32	false



Name	Age	Part-time
Jack	23	☑
John	32	☐

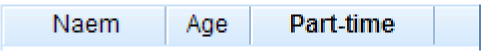
3.13.7.9. column header style

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColSortable(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Part-time", width:90, type: "checkbox", headerStyle: "font-weight:bold;"}
]).show(block)
```

Sets bold

Output:



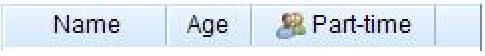
3.13.7.10. column header icon

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColSortable(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Part-time", width:90, type: "checkbox", renderer: function(h){
                return "<img style='vertical-align:middle' src='img/a.gif'> "+ h.caption;
        }}
]).show(block)
```

Renderer function

Output:



3.13.7.11. Update column header dynamically

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setGridHandlerCaption("Name")
.setHeader([
	{id:"col1", caption:"Age", width:40, type: "number"},
	{id:"col2", caption:"Part-time", width:90, type: "checkbox"}
]).setRows([
	{id:"row1",caption:'Jack',cells:[23, true]},
	{id:"row2",caption:'John',cells:[32, false]}
]).show(block)

_.asyRun(function(){
	tg.updateHeader('col2','Full-time')
},1000)

_.asyRun(function(){
	tg.updateHeader('col2',{caption:'Part-time', width:40, headerStyle:'font-weight:bold', colResizer:false, colSortable:false, colMovable:true, colHidable:true})
},2000)
```

Updates caption only

Those properties are updatable

3.13.8. Row config

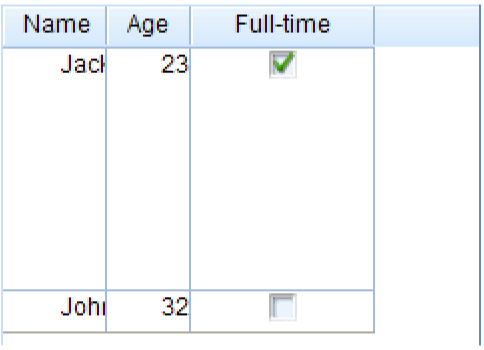
3.13.8.1. Row height

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setGridHandlerCaption("Name")
.setHeader([
	{id:"col1", caption:"Age", width:40, type: "number"},
	{id:"col2", caption:"Full-time", width:90, type: "checkbox"}
]).setRows([
	{id:"row1",caption:'Jack',cells:[23, true], height:120},
	{id:"row2",caption:'John',cells:[32, false]}
]).show(block)
```

Sets row height

Output:



Name	Age	Full-time
Jack	23	☑
John	32	□

3.13.8.2. Drag&Drop to modify row height

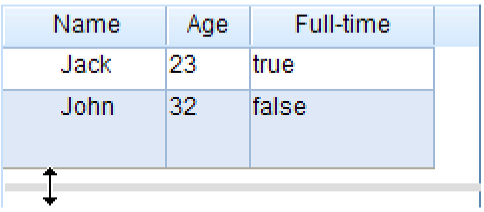
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setRowResizer(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40},
        {id:"col2", caption:"Full-time", width:90}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false],rowResizer:true}
]).show(block)
```

Global disabled rowResizer

This row has rowResizer

Output:



Name	Age	Full-time
Jack	23	true
John	32	false

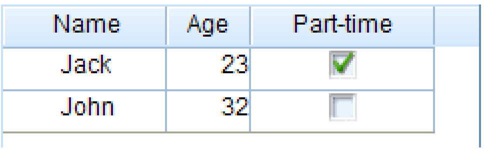
3.13.8.3. Setting cell type in row

Input:

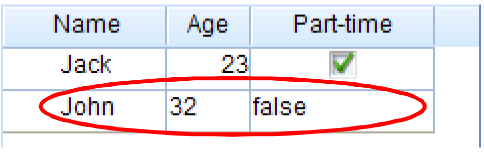
```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColSortable(false)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Part-time", width:90, type: "checkbox"
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false],type: "label"}
]).show(block)
```

Sets all cells in this row type to ‘'label'

Output:



Name	Age	Part-time
Jack	23	☑
John	32	□



Name	Age	Part-time
Jack	23	☑
John	32	false

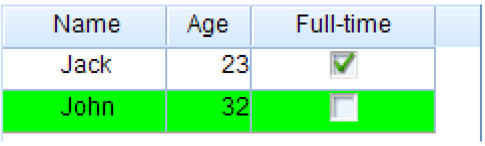
3.13.8.4. Row style

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Full-time", width:90, type: "checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32, false],rowStyle: "background-color:#00ff00"}
]).show(block)
```

Sets styles

Output:



Name	Age	Full-time
Jack	23	☑
John	32	□

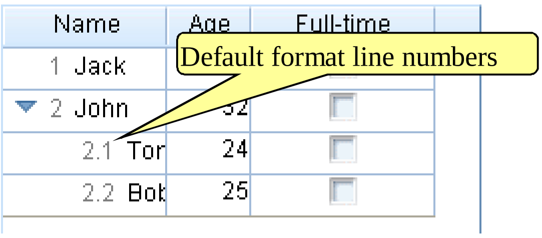
3.13.8.5. Row numbers

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setRowNumbered(true)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Full-time", width:90, type: "checkbox", width:90, type: "checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23]},
        {id:"row2",caption:'John',cells:[32],
         sub:[{id:"row21",caption:'Tom',cells:[24]},
                {id:"row22",caption:'Bob',cells:[25]}
        ]}
]).show(block)
```

To show row numbers

Output:



Name	Age	Full-time
1 Jack		
2 John	32	□
2.1 Tor	24	□
2.2 Bob	25	□

Default format line numbers

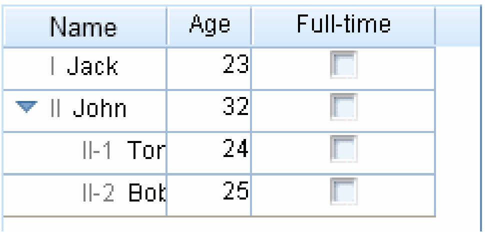
3.13.8.6. Custom row numbers

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setRowNumbered(true)
.setGridHandlerCaption("姓名")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Full-time", width:90, type: "checkbox", width:90, type: "checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23]},
        {id:"row2",caption:'John',cells:[32],
         sub:[{id:"row21",caption:'Tom',cells:[24]},
                {id:"row22",caption:'Bob',cells:[25]}
        ]}
])
.setCustomFunction('getNumberedStr',function(no){
        var a=no.split('.');
        a[0]={1:'I',2:'II'}[a[0]];
        return a.join('-')
})
.show(block)
```

Custom function

Output:



Name	Age	Full-time
I Jack	23	□
II John	32	□
II-1 Tor	24	□
II-2 Bob	25	□

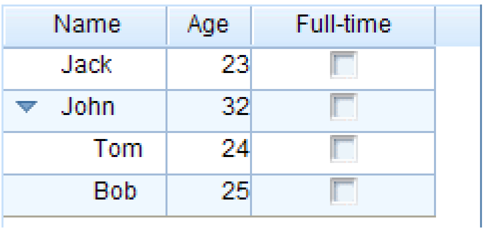
3.13.8.7. Alternate Row Colors

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setAltRowsBg (true)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Full-time", width:90, type: "checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23]},
        {id:"row2",caption:'John',cells:[32],
          sub:[{id:"row21",caption:'Tom',cells:[24]},
                {id:"row22",caption:'Bob',cells:[25]}
        ]}
]).show(block)
```

Sets alternate bg color

Output:



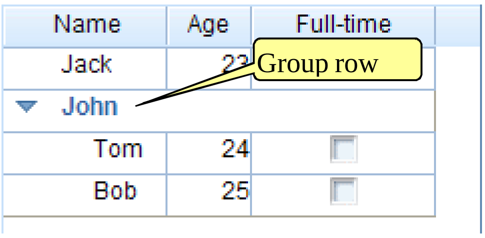
Name	Age	Full-time
Jack	23	□
John	32	□
Tom	24	□
Bob	25	□

3.13.8.8. Group

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Full-time", width:90, type: "checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23]},
        {id:"row2",caption:'John',cells:[32],group:true,
          sub:[{id:"row21",caption:'Tom',cells:[24]},
                    {id:"row22",caption:'Bob',cells:[25]}
        ]}
]).show(block)
```

It's a group row



Name	Age	Full-time
Jack	23	
John		
Tom	24	☐
Bob	25	☐

Group row

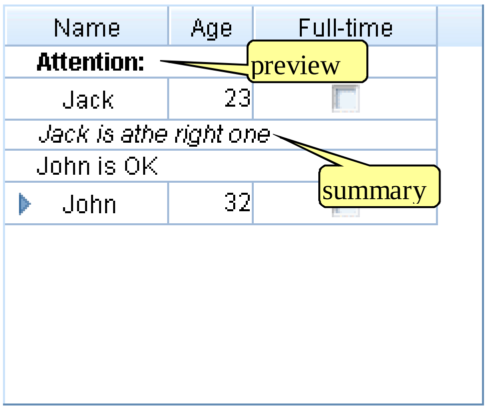
3.13.8.9. Preview and Summary region

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setGridHandlerCaption("Name")
.setHeader([
        {id:"col1", caption:"Age", width:40, type: "number"},
        {id:"col2", caption:"Full-time", width:90, type: "checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23], preview: '<strong>Attention:</strong>',summary: '<em>Jack is athe
right one</em>'} ,
        {id:"row2",caption:'John',cells:[32], preview: 'John is OK',
          sub:[{id:"row21",caption:'Tom',cells:[24]},
                    {id:"row22",caption:'Bob',cells:[25]}
        ]}
]).show(block)
```

preview

summary



Name	Age	Full-time
Attention:		
Jack	23	□
Jack is athe right one		
John is OK		
John	32	

3.13.8.10. Update row dynamically

Input:

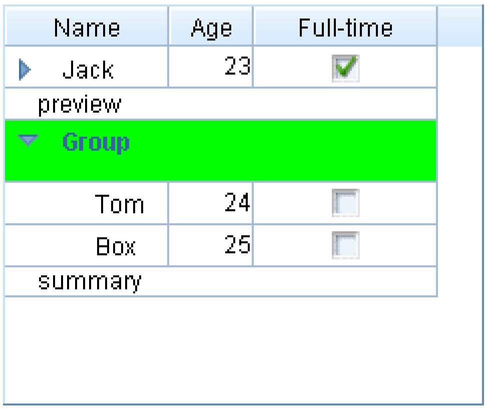
```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80).setGridHandlerCaption("Name")
.setHeader([
{id:"col1", caption:"Age", width:40, type: "number"},{id:"col2", caption:"Full-time", width:90, type:
"checkbox"}
]).setRows([
        {id:"row1",caption:'Jack',cells:[23, true]},
        {id:"row2",caption:'John',cells:[32],
         sub:[{id:"row21",caption:'Tom',cells:[24]},
                 {id:"row22",caption:'Box',cells:[25]}
        ]}
]).show(block)

_.asyRun(function(){
        tg.updateRow('row2', 'Jerry')
},1000)

_.asyRun(function(){
        tg.updateRow('row2',{caption:'Group', height:30, rowStyle:'background-color:#00ff00;', rowResizer:false,
group:true, preview:'preview', summary:'summary'})
},2000)

_.asyRun(function(){
        tg.updateRow('row1', {sub:[{value:"Kate",cells:[24,true]}]})
},3000)
```

Updates row caption only

These properties are updatable

Updates all sub rows

Output:



Name	Age	Full-time
Jack	23	☑
preview		
Group		
Tom	24	□
Box	25	□
summary		

3.13.9. Cell config

3.13.9.1. Cell types

These types are support:

- ‘label’: readonly text;
- ‘button’: the button;
- ‘input’: single line input;
- ‘textarea’: multi lines input;
- ‘number’: number only input;
- ‘currency’: currency only input;
- ‘progress’: the progress appearance;
- ‘combobox’: combo input;
- ‘listbox’: readonly combo input;
- ‘getter’: for getting data;
- ‘helpinput’: help data input;
- ‘cmdbox’: command box input;
- ‘popbox’: pop box input;
- ‘time’: time input;
- ‘date’: date input;
- ‘color’: color input;

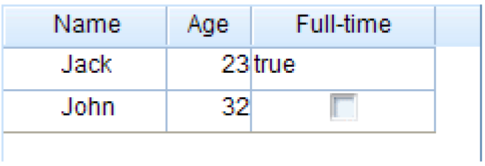
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColSortable(false)
.setGridHandlerCaption("Name")
.setHeader([
    {id:"col1", caption:"Age", width:40, type: "number"},
    {id:"col2", caption:"Full-time", width:90, type: "label"}
]).setRows([
    {id:"row1",caption:'Jack',cells:[23, true]},
    {id:"row2",caption:'John',cells:[32, {value:false, type: "checkbox"}] }
]).show(block)
```

Setting in column header data

Setting in cell (has priority)

Output:



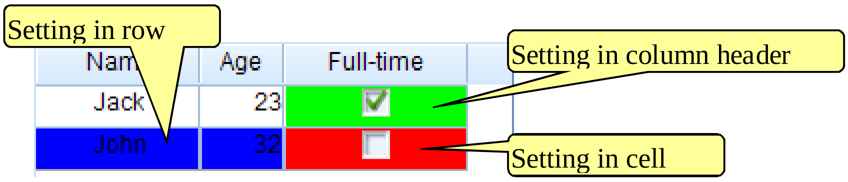
Name	Age	Full-time
Jack	23	true
John	32	☐

3.13.9.2. Cell style

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setColSortable(false)
.setGridHandlerCaption("Name")
.setHeader([
    {id:"col1", caption:"Age", width:40, type: "number"},
    {id:"col2", caption:"Full-time", width:90, type: "checkbox", cellStyle: "background-color:#00ff00;"}
]).setRows([
    {id:"row1",caption:'Jack', cells:[23, true]},
    {id:"row2",caption:'John', cellStyle: "background-color:#0000ff;",cells:[
        32,
        {value:false, cellStyle: "background-color:#ff0000;"}
     ] }
]).show(block)
```

Output:



Nam	Age	Full-time
Jack	23	☑
John	32	☐

3.13.9.3. Update cell dynamically

Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandlerWidth(80).setGridHandlerCaption("Name")
.setHeader([
{id:"col1", caption:"Age", width:40, type: "number"},{id:"col2", caption:"Full-time", width:90, type:
"checkbox"}
]).setRows([
    {id:"row1",caption:'Jack',cells:[23, true]},
    {id:"row2",caption:'John',cells:[32]}
]).show(block)

_.asyRun(function(){
    tg.updateCellByRowCol('row2','col1', 18)
},1000)

_.asyRun(function(){
    tg.updateCellByRowCol('row2','col1',{value:18,cellStyle:'background-color:#00ff00;'})
},2000)

_.asyRun(function(){
    tg.updateCellByRowCol ('row2','col1', {type:"listbox",value:"20", editorListItems:["20","30","40"],
editable:true})
},3000)
```

Updates value only

These properties are updatable

Updates cell type

3.13.10. Editable

“ editable” property in TreeGrid determines whether the TreeGrid is editable or not . Each column / row / cell has this property too. Those setting follow “Fine-grained priority principle”.

- TreeGrid’s editable =>false; cell’s editable=>true: only this cell is editable
- TreeGrid’s editable =>false; column header’s editable=>true: only this column is editable
- TreeGrid’s editable =>false; row’s editable=>true: only this row is editable
- TreeGrid’s editable =>true; cell’s editable=>true: only this cell is uneditable
- TreeGrid’s editable =>true; column header’s editable=>false: only this column is uneditable
- TreeGrid’s editable =>true; row’s editable=> false: only this row is uneditable

It should be noted that, cells in Row handler are uneditalbe; cells with ‘label’ or ‘button’ type are uneditable.

3.13.10.1. Editable TreeGrid

Input:

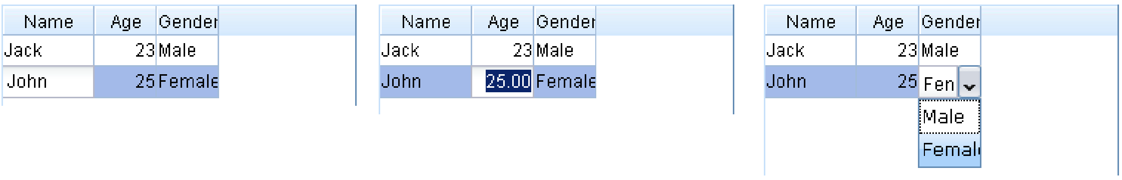
```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(true)
.setHeader([
{id:"col1", caption:"Name", width:60, type: 'input'},
{id:"col2", caption:"Age", width:40, type: "number"},
{id:"col3", caption:"Gender", width:40, type: "listbox", editorListItems:[{id:'male',caption:'Male'},{id:'female',
caption:'Female'}]}
]).setRows([
    ['Jack',23, {value:'male',caption:'Male'}],
    ['John',25, {value:'female',caption:'Female'}]
]).show(block)
```

Sets editable

List for editor

Value and caption

Output:



Name	Age	Gender
Jack	23	Male
John	25	Female

Name	Age	Gender
Jack	23	Male
John	25.00	Female

Name	Age	Gender
Jack	23	Male
John	25	Fen

3.13.10.2. Editable column

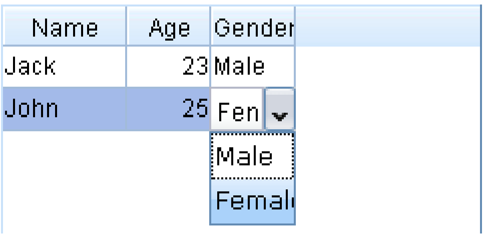
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(false)
.setHeader([
{id:"col1", caption:"Name", width:60, type: 'input'},
{id:"col2", caption:"Age", width:40, type: "number"},
{id:"col3", caption:"Gender", width:40, type: "listbox", editable:true,
editorListItems:[{id:'male',caption:'Male'},{id:'female', caption:'Female'}]}
]).setRows([
    ['Jack',23, {value:'male',caption:'Male'}],
    ['John',25, {value:'female',caption:'Female'}]
]).show(block)
```

Sets uneditable

This column is editable

Output:



Name	Age	Gender
Jack	23	Male
John	25	Fen
		Male
		Femal

3.13.10.3. Editable row

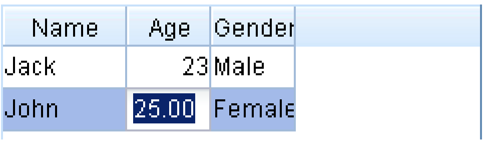
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(false)
.setHeader([
{id:"col1", caption:"Name", width:60, type: 'input'},
{id:"col2", caption:"Age", width:40, type: "number"},
{id:"col3", caption:"Gender", width:40, type: "listbox", editorListItems:[{id:'male',caption:'Male'},{id:'female',
caption:'Female'}]}
]).setRows([
      ['Jack',23, {value:'male',caption:'Male'}],
      {cells:['John',25, {value:'female',caption:'Female'}],editable:true}
]).show(block)
```

Sets uneditable

This row is editable

Output:



Name	Age	Gender
Jack	23	Male
John	25.00	Female

3.13.10.4. Editable cell

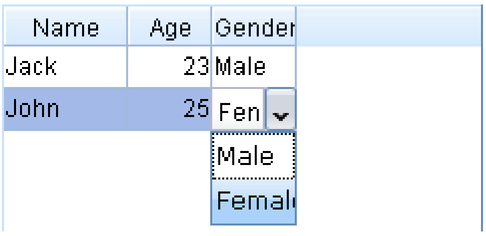
Input:

```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(false)
.setHeader([
{id:"col1", caption:"Name",    width:60,    type: 'input'},
{id:"col2", caption:"Age",    width:40,    type: "number"},
{id:"col3", caption:"Gender", width:40, type: "listbox", editorListItems:[{id:'male', caption:'Male'}, {id:'female',
{id:'female', caption:'Female'} ]}
]).setRows([
      ['Jack',23, {value:'male', caption:'Male'}],
      ['John',25, {value:'female', caption:'Female',    editable:true }]
]).show(block)
```

Sets uneditable

Only this cell is editable

Output:



Name	Age	Gender
Jack	23	Male
John	25	Fen

3.13.10.5. The Editor

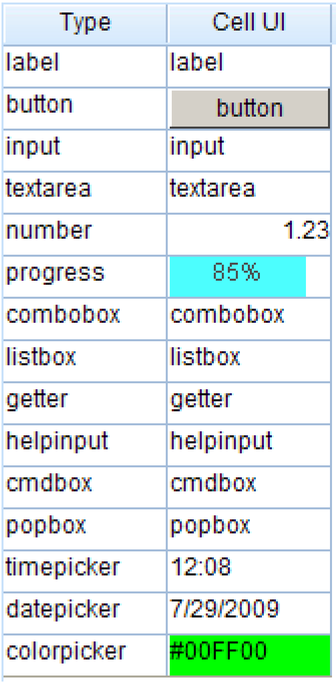
When a cell is set to editable, “active this cell” will show a corresponding editor. There are the following editors for different cell types.

- ‘label’: readonly; no editor
- ‘button’: readonly; no editor
- ‘input’: normal xui.UI.Input control
- ‘textarea’: multi lines xui.UI.Input control
- ‘number’: number only xui.UI.Input control
- ‘currency’: currency only xui.UI.Input control
- ‘progress’: xui.UI.ComboInput control, spin
- ‘combobox’: xui.UI.ComboInput control, combobox
- ‘listbox’: xui.UI.ComboInput control, listbox
- ‘getter’: xui.UI.ComboInput control, getter
- ‘helpinput’: xui.UI.ComboInput control, helpinput
- ‘cmdbox’: xui.UI.ComboInput control, cmdbox
- ‘popbox’: xui.UI.ComboInput control, popbox
- ‘time’: xui.UI.ComboInput control, time
- ‘date’: xui.UI.ComboInput control, date
- ‘color’: xui.UI.ComboInput control, color

Input:

```
var block=new xui.UI.Block({width:300,height:340}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(true)
.setHeader(["Type","Cell UI"]).setRows([
        {cells:['label',{type:'label',value:'label'}]},
        {cells:['button',{type:'button',value:'button'}]},
        {cells:['input',{type:'input',value:'input'}]},
        {cells:['textarea',{type:'textarea',value:'textarea'}]},
        {cells:['number',{type:'number',value:'1.23'}]},
        {cells:['currencty',{type:'number',value:'21.23'}]},
        {cells:['progress',{type:'progress',value:'0.85'}]},
        {cells:['combobox',{type:'combobox',value:'combobox'}]},
        {cells:['listbox',{type:'listbox',value:'listbox'}]},
        {cells:['getter',{type:'getter',value:'getter'}]},
        {cells:['helpinput',{type:'helpinput',value:'helpinput'}]},
        {cells:['cmdbox',{type:'cmdbox',value:'cmdbox'}]},
        {cells:['popbox',{type:'popbox',value:'popbox'}]},
        {cells:['time',{type:'time',value:'12:08'}]},
        {cells:['date',{type:'date',value:(new Date).getTime()}]},
        {cells:['color',{type:'color',value:'#00ff00'}]}
]).show(block)
```

Output:



Type	Cell UI
label	label
button	button
input	input
textarea	textarea
number	1.23
progress	85%
combobox	combobox
listbox	listbox
getter	getter
helpinput	helpinput
cmdbox	cmdbox
popbox	popbox
timepicker	12:08
datepicker	7/29/2009
colorpicker	#00FF00

3.13.10.6. Custom the editor

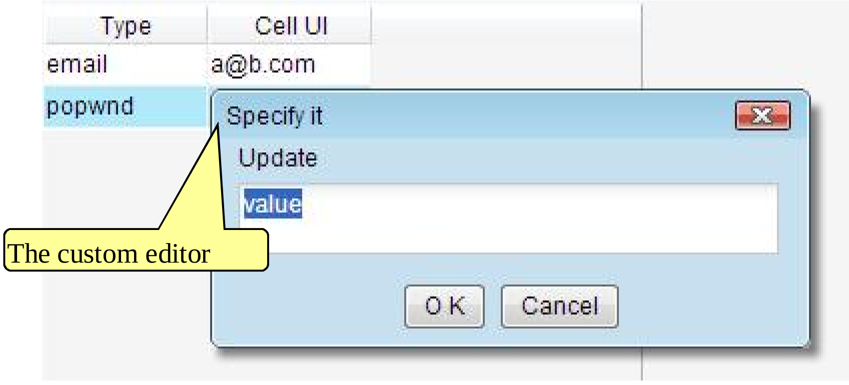
Input:

```
var block=new xui.UI.Block({width:300,height:340}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(true)
.setHeader(["Type","Cell UI "]).setRows([
        {cells:['email',{type:'email',value:'a@b.com'}]},
        {cells:['popwnd',{type:'popwnd',value:'value'}]}
])
.beforeIniEditor(function(profile, cell, cellNode){
        var t=cell.type;
        if(t=='email'){
                var editor = new xui.UI.Input({valueFormat:"^[\\w\\.=-]+@[\\w\\.-]+\\.[\\w\\.-]{2,4}$"});
                return editor;
        }

        if(t=='popwnd'){
                var dlg=xui.prompt('Specify it','Update',cell.value, function(value){
                        if(cell.value!==value)
                                profile.boxing().updateCell(cell, value);
                });
                dlg.getRoot().cssPos(cellNode.offset());
                return false;
        }
}).show(block);
```

Return the custom editor
xui.UI.Input or CombInput

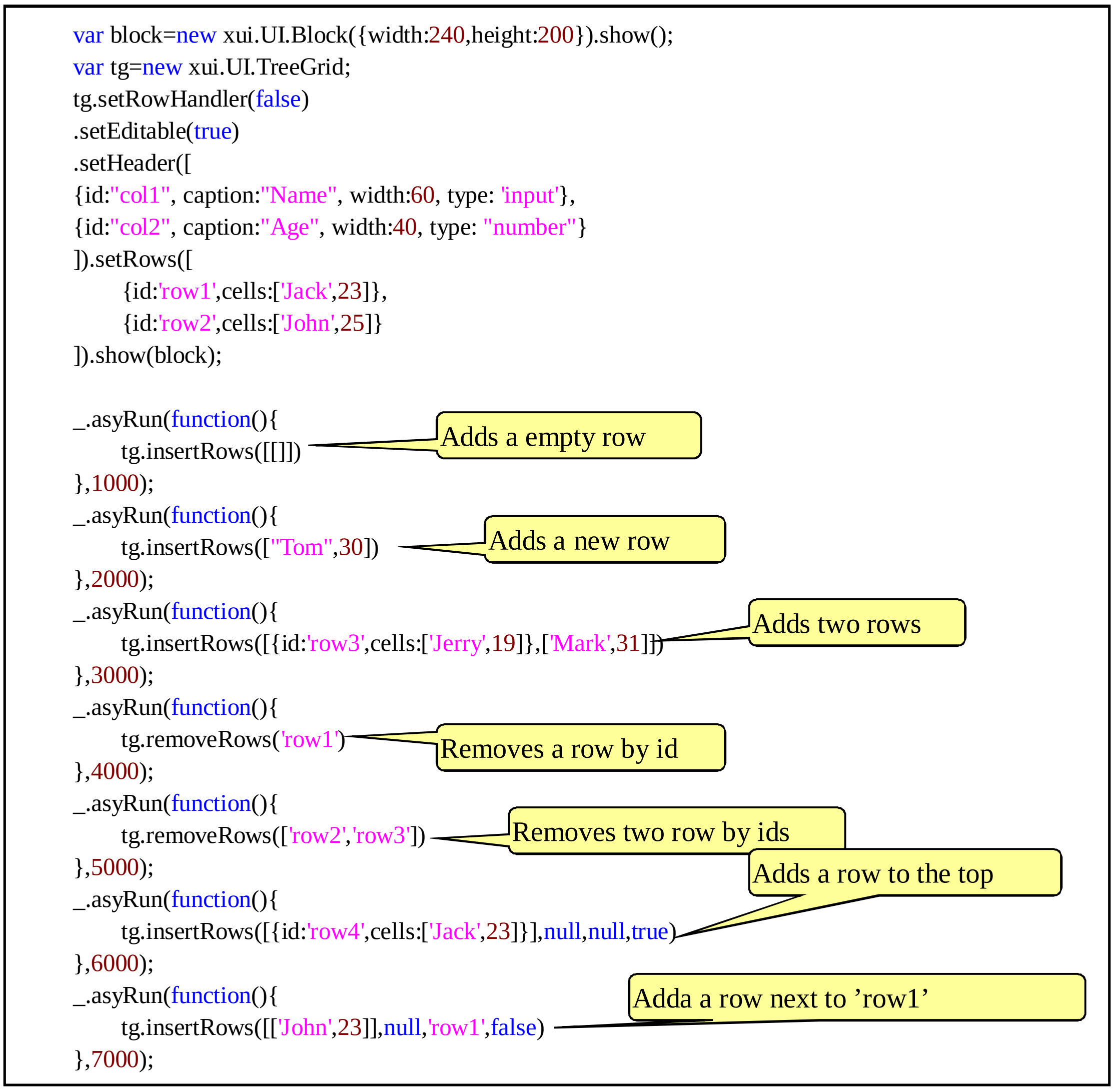
Return false for advanced custom editor

Output:



3.13.11. Add/Remove rows

Input:



```
var block=new xui.UI.Block({width:240,height:200}).show();
var tg=new xui.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(true)
.setHeader([
{id:"col1", caption:"Name", width:60, type: 'input'},
{id:"col2", caption:"Age", width:40, type: "number"}
]).setRows([
        {id:'row1',cells:['Jack',23]},
        {id:'row2',cells:['John',25]}
]).show(block);

_.asyRun(function(){
        tg.insertRows([[]])
},1000);
_.asyRun(function(){
        tg.insertRows(["Tom",30])
},2000);
_.asyRun(function(){
        tg.insertRows([{id:'row3',cells:['Jerry',19]},['Mark',31]])
},3000);
_.asyRun(function(){
        tg.removeRows('row1')
},4000);
_.asyRun(function(){
        tg.removeRows(['row2','row3'])
},5000);
_.asyRun(function(){
        tg.insertRows([{id:'row4',cells:['Jack',23]}],null,null,true)
},6000);
_.asyRun(function(){
        tg.insertRows([['John',23]],null,'row1',false)
},7000);
```

NOTE

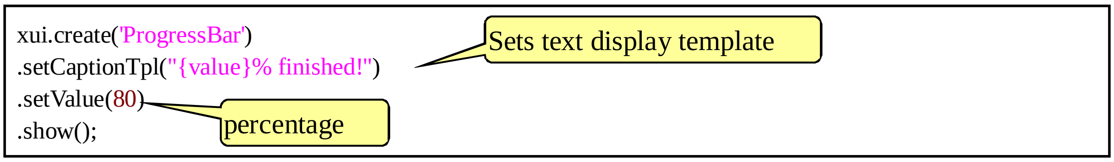
chapter2/TGDynamic\index.html is an overall example for ThreeGrid

chapter2/TreeGrid.Paging\index.html is another example for multi pages

3.14. Other standard controls

3.14.1. ProgressBar

Input:

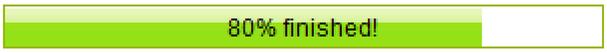


```
xui.create('ProgressBar')
.setCaptionTpl("{value}% finished!")
.setValue(80)
.show();
```

Sets text display template

percentage

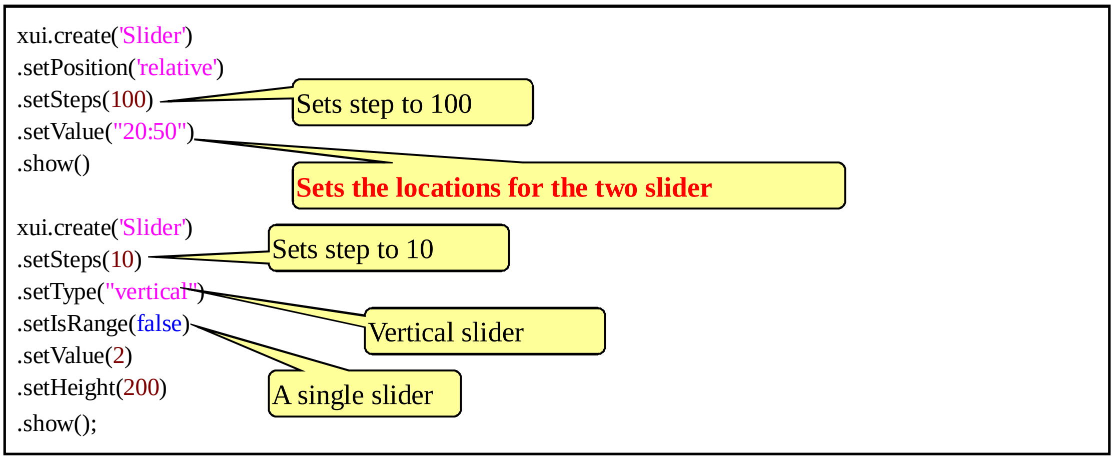
Output:



80% finished!

3.14.2. Slider

Input:



```
xui.create('Slider')
.setPosition('relative')
.setSteps(100)
.setValue("20:50")
.show()

xui.create('Slider')
.setSteps(10)
.setType("vertical")
.setIsRange(false)
.setValue(2)
.setHeight(200)
.show();
```

Sets step to 100

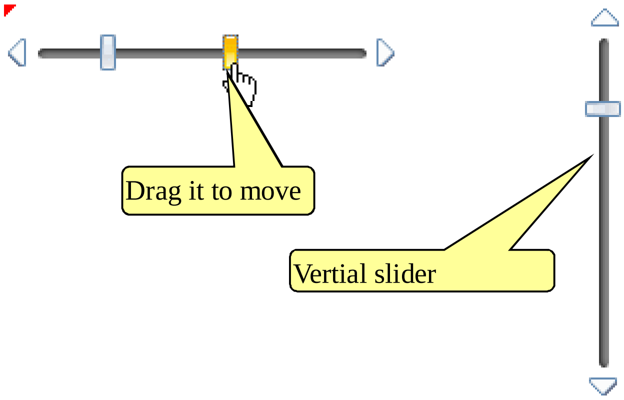
Sets the locations for the two slider

Sets step to 10

Vertical slider

A single slider

Output:



Drag it to move

Vertial slider

3.14.3. Image

```
xui.create("Image")
.setSrc("img/a.gif")
.afterLoad(function(){
        xui.message("The picture is loaded.");
})
.show();

xui.create("Image")
.setSrc("img/b.gif")
.beforeLoad(function(){
        return false;
})
.show()
```

The image was loaded successful

Cancel loading process

3.14.4. PageBar

Input:

```
var onclick=function(profile,page){
        profile.boxing().setPage(page);
};
// a PageBar
xui.create('PageBar')
.setValue("1:5:12")
.onClick(onclick)
.show();
// another PageBar
xui.create('PageBar')
.setValue("1:5:12")
.setTop(100)
.setCaption("")
.setPrevMark("<<")
.setNextMark(">>")
.setTextTpl("[ * ]")
.onClick(onclick)
.show();
```

Set current page

1.Min page; 2.current page; 3.max page

onClick event

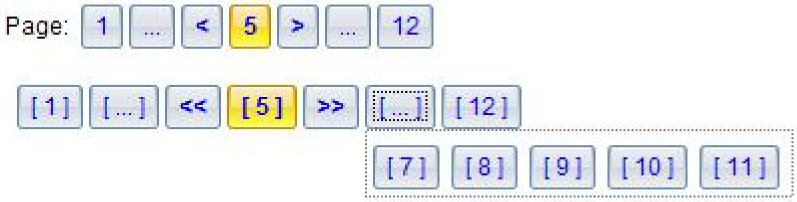
Caption label

Prev command label

Next command label

Page label templates, * is the variable value

Output:



Chapter 4. Data exchanging(Ajax)

CrossUI is a client-side solution, it can work with any backend (php, .Net, Java, python) or static HTML pages. Client-side and backend is completely decoupled. Client-side does not need to care what kind of technique is used in the backend. Client-side sends request to, and gets response from a given backend service(e.g. JSON service, REST service) .

There are three IO class in CrossUI:

- xui.Ajax: An AJAX wrapper for xmlHttp object. It's features:
  - Can only access the same domain by default;
  - Works both synchronous and asynchronous;
  - Works both 'get' and 'post' methods;
  - Returns string.
- xui.SAjax: An AJAX wrapper for "script tag". It's features:
  - Cross domain;
  - Asynchronous only;
  - Can not post data;
  - Returned content is packaged as javascript's Object

    inb.SAjax send request data includes a "callback" parameter (default is "*xui.SAjax.NO._1*").

    Server's return data must be the following format:

    ```
    xui.SAjax.NO._1 ({/*JSON */})
    ```

- xui.IAjax: An AJAX wrapper for "iframe". It's features:
  - Cross domain;
  - Asynchronous only;
  - Can update file;
  - Works both 'get' and 'post' methods;
  - Returned content is packaged as javascript's Object

    inb.IAjax send request data includes a "callback" parameter (default is "window.name").

    Server's return data must be the following format:

    ```
    <script type='text' id='json'>{/*JSON*/}</script>
    <script type='text/javascript'>
    window.name=document.getElementById('json').innerHTML;
    </script>
    ```

"xui.request" function can choose an appropriate class from xui.Ajax, xui.SAjax or xui.IAjax automatically, according to requested domain, 'GET/POST' method and other information.

NOTE
Examples in this chapter works only as a http url, do not double-click directly to open.

4.1. Fiddler

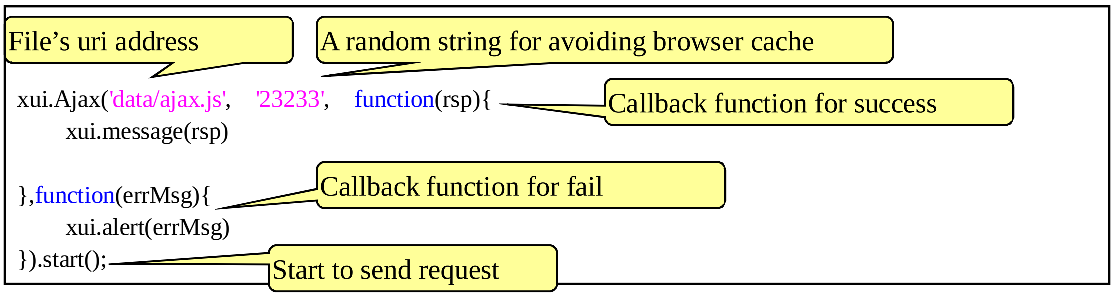
In order to understand the data exchanges process better, you need a tool like Fiddler to monitor network traffic.

Go to http://www.fiddler2.com/fiddler2/ to get Fiddler.

Fiddler can configure IE proxy automatically, but if you are in firefox, chrome or opera, you need to configure the proxy by manual (Fiddler proxy: 127.0.0.1:8888). Of course, you can find some firefox proxy plug-ins to help you.

4.2. To get the contents of the file

xui.Ajax can get file contents easily.



```
xui.Ajax('data/ajax.js',   '23233',   function(rsp){
        xui.message(rsp)

},function(errMsg){
        xui.alert(errMsg)
}).start();
```

File’s uri address

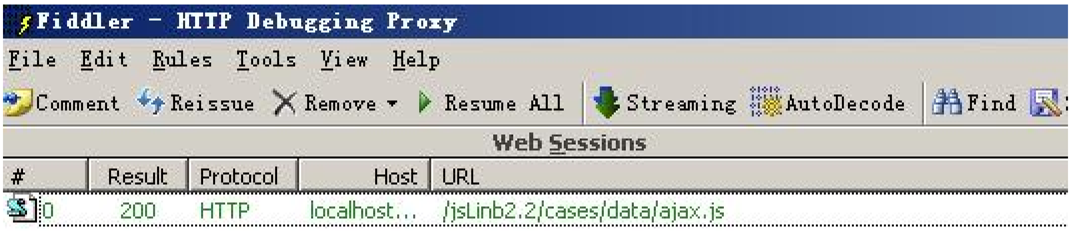
A random string for avoiding browser cache

Callback function for success

Callback function for fail

Start to send request

In Fiddler:



Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Find
Web Sessions

#	Result	Protocol	Host	URL
0	200	HTTP	localhost...	/jsLinb2.2/cases/data/ajax.js

4.3. Synchronous data exchange

Only xui.Ajax support synchronous data exchanging.

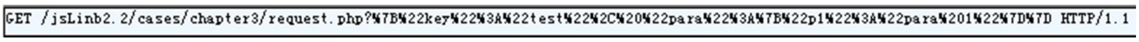
```
var url="chapter3/request.php";
xui.Ajax(url, {
        key:'test',
        para:{p1:'para 1'}
},function(rsp){
        xui.log(rsp);
},function(errMsg){
        xui.alert(errMsg)
}, null, {
      asy:false
}).start();
```

Request data

synchronous

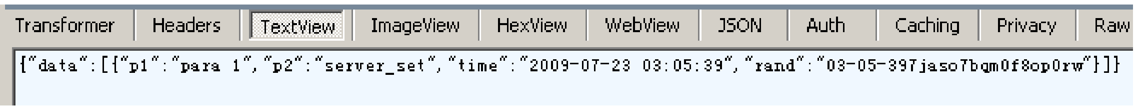
In fiddler:

The request:



```
GET /jsLinb2.2/cases/chapter3/request.php?%7B%22key%22%3A%22test%22%2C%20%22para%22%3A%7B%22p1%22%3A%22para%201%22%7D%7D HTTP/1.1
```

The response:



```
{"data":[{"p1":"para 1","p2":"server_set","time":"2009-07-23 03:05:39","rand":"03-05-397jaso7bqm0f8op0rw"}]}
```

This is an asynchronous request:

```
var url="chapter3/request.php";
xui.Ajax(url, {
        key:'test',
        para:{p1:'para 1'}
},function(rsp){
        xui.log(rsp);
},function(errMsg){
        xui.alert(errMsg)
}).start();
```

4.4. Cross-domain

xui.SAjax and xui.IAjax can be used for calling Cross Domain Web Services. But only xui.IAjax can post data and upload file.

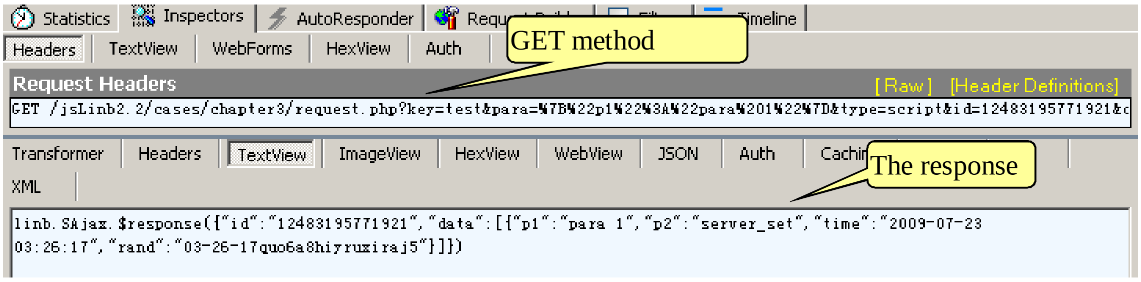
4.4.1. To monitor SAjax

Code:

```
var url="chapter3/request.php";
xui.SAjax(url, {
        key:'test',
        para:{p1:'para 1'}
},function(rsp){
        xui.log(rsp);
},function(errMsg){
        xui.alert(errMsg)
}).start();
```

Request data

In Fiddler:

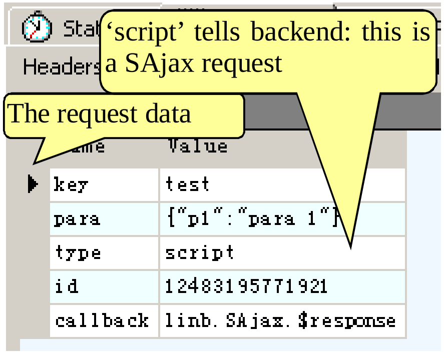


GET method

```
GET /jsLinb2.2/cases/chapter3/request.php?key=test&para=%7B%22p1%22%3A%22para%201%22%7D&type=script&id=12483195771921&c
```

The response

```
linb.SAjax.$response({"id":"12483195771921","data":[{"p1":"para 1","p2":"server_set","time":"2009-07-23
03:26:17","rand":"03-26-17quo6a8hiyruxiraj5"}]})
```



'script' tells backend: this is a SAjax request

The request data

Name	Value
key	test
para	{"p1":"para 1"}
type	script
id	12483195771921
callback	linb.SAjax.$response

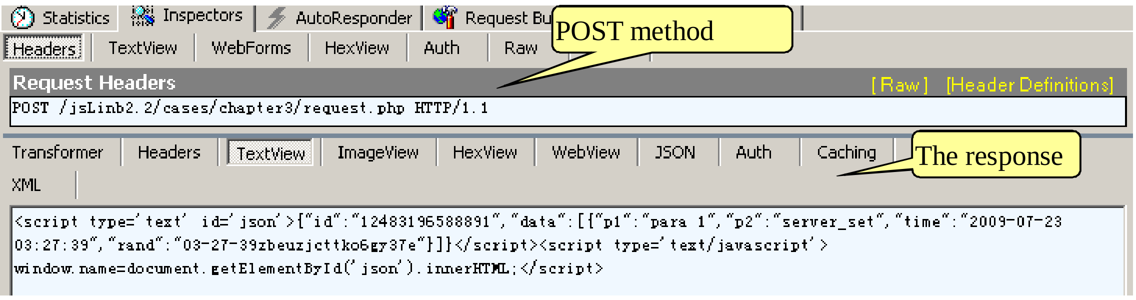
4.4.2. To monitor IAjax

Code:

```
var url="chapter3/request.php";
xui.IAjax(url, {
        key:'test',
        para:{p1:'para 1'}
},function(rsp){
        xui.log(rsp);
},function(errMsg){
        xui.alert(errMsg)
}).start();
```

Request data

In Fiddler:

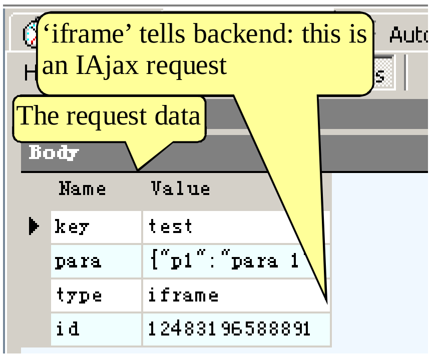


```
POST /jsLinb2.2/cases/chapter3/request.php HTTP/1.1
```

POST method

```
<script type='text' id='json'>{"id":"12483196588891","data":[{"p1":"para 1","p2":"server_set","time":"2009-07-23
03:27:39","rand":"03-27-39zbeuzjcttko6gy37e"}]}</script><script type='text/javascript'>
window.name=document.getElementById('json').innerHTML;</script>
```

The response



Name	Value
key	test
para	{"p1":"para 1
type	iframe
id	12483196588891

'iframe' tells backend: this is an IAjax request

The request data

By default, IAajax use "POST" method, you can specify method in options.

```
var url="chapter3/request.php";
xui.IAjax(url, {
        key:'test',
        para:{p1:'para 1'}
},function(rsp){
        xui.log(rsp);
},function(errMsg){
        xui.alert(errMsg)
},null,{
        method: 'get'
}).start();
```

Switch to GET method

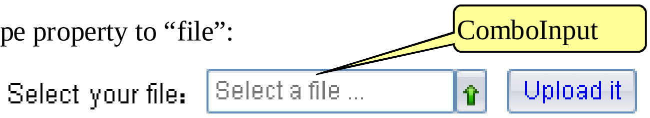
4.5. File Upload

Only xui.UI.IAjax can upload file.

This code in this section is in "chapter3/upload/".

4.5.1. Selecting upload file with ComboInput

Sets ComboInput's type property to "file":



chapter3/upload/index.html

4.5.2. Upload by IAjax

```
Class('App', 'xui.Com',{
    Instance:{
        iniComponents:function(){
            // [[code created by CrossUI UI Builder
            var host=this, children=[], append=function(child){children.push(child.get(0))};

            append((new xui.UI.SLabel)
                .setHost(host,"slabel1")
                .setLeft(40)
                .setTop(44)
                .setCaption("Select your file: ")
            );

            append((new xui.UI.ComboInput)
                . setHost(host,"upload")
                .setLeft(140)
                .setTop(40)
                .setWidth(140)
                .setReadonly(true)
                .setType("upload")
                .setValue("Select a file ...")
            );

            append((new xui.UI.SButton)
                . setHost(host,"sbutton3")
                .setLeft(290)
                .setTop(40)
                .setCaption("Upload it")
                .onClick("_sbutton3_onclick")
            );

            return children;
            // ]]code created by CrossUI UI Builder
        },
        _sbutton3_onclick:function (profile, e, src, value) {
            var file=this.upload.getUploadObj();
            if(file){
                xui.IAjax('../request.php',{key:'upload',para:{},file:file},function(rsp){
                    xui.alert(rsp.data.message);
                },function(errMsg){
                    xui.alert(errMsg)
                }).start();
            }
        }
    }
});
```

Created by Designer

Upload control

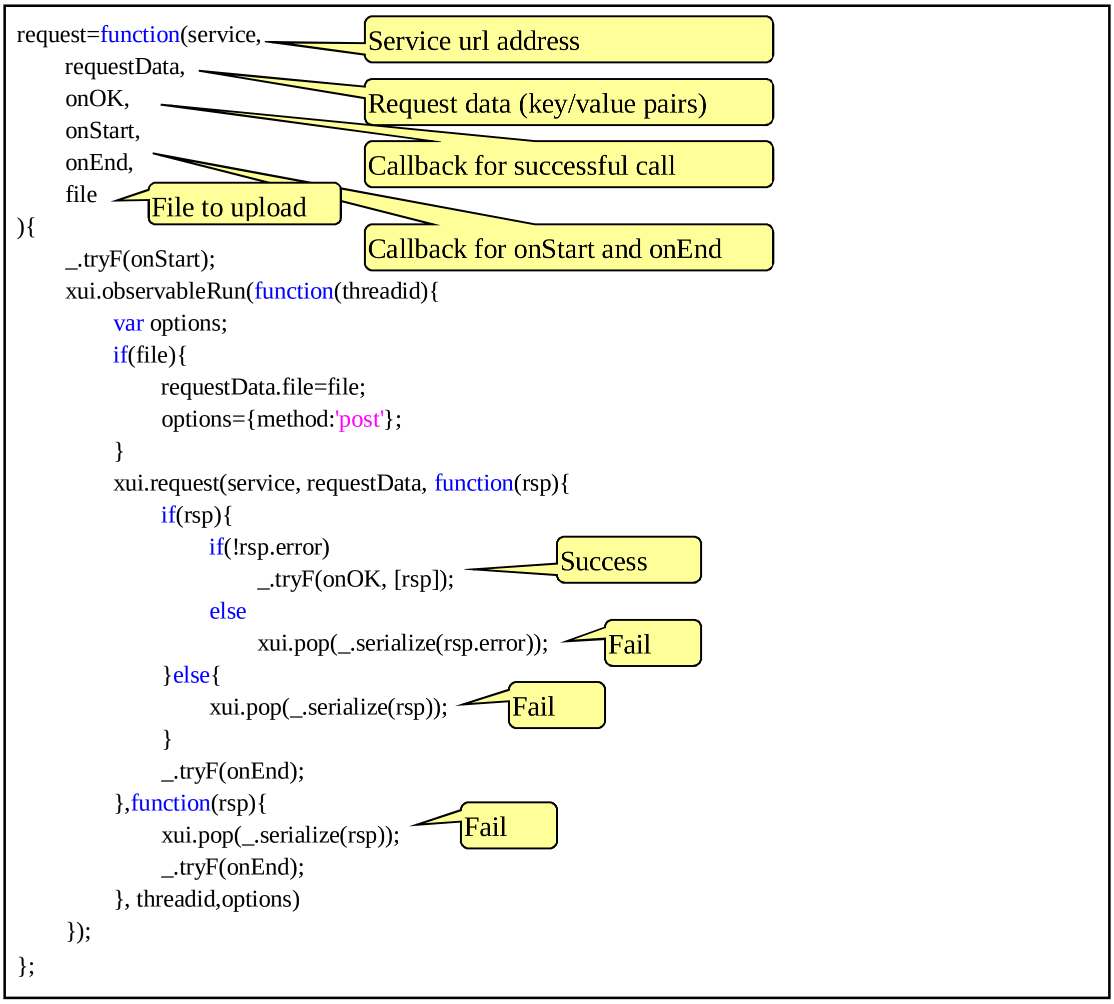
Getting file content

IAjax upload

Successful return

4.6. A request wrapper for real application

In practical applications, you can choose xui.Ajax, xui.SAjax and xui.IAjax according to the actual situation. Usually, we will wrap a common function or class to handle all data interaction with the backend service. This is an example wrapper. Just for your reference.

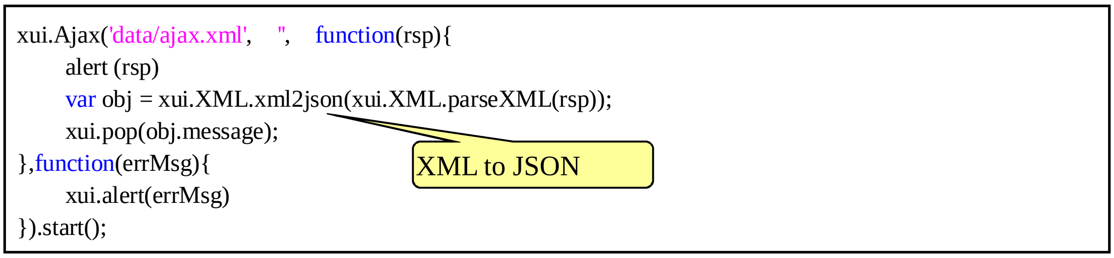


```
request=function(service,
    requestData,
    onOK,
    onStart,
    onEnd,
    file
){
    _.tryF(onStart);
    xui.observableRun(function(threadid){
        var options;
        if(file){
            requestData.file=file;
            options={method:'post'};
        }
        xui.request(service, requestData, function(rsp){
            if(rsp){
                if(!rsp.error)
                    _.tryF(onOK, [rsp]);
                else
                    xui.pop(_.serialize(rsp.error));
            }else{
                xui.pop(_.serialize(rsp));
            }
            _.tryF(onEnd);
        },function(rsp){
            xui.pop(_.serialize(rsp));
            _.tryF(onEnd);
        }, threadid,options)
    });
};
```

Service url address
Request data (key/value pairs)
Callback for successful call
File to upload
Callback for onStart and onEnd
Success
Fail
Fail
Fail

4.7. XML Data

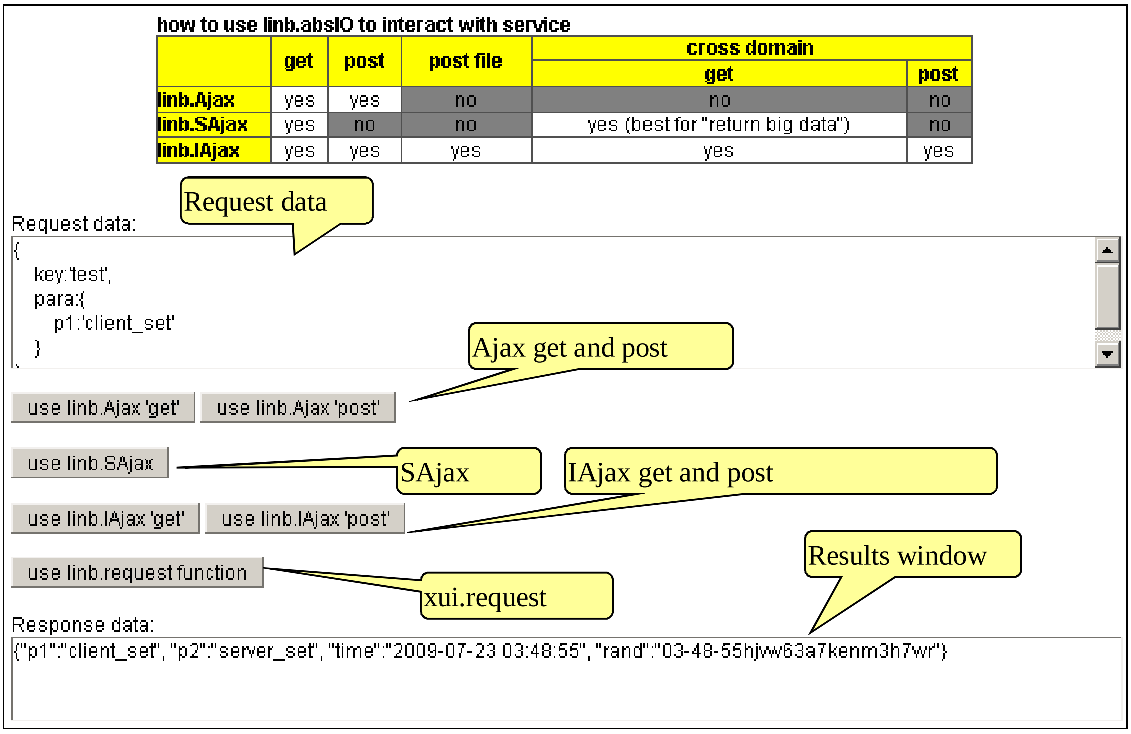
If the server returns xml data, we can use xui.XML to convert the XML data into JSON data.



```
xui.Ajax('data/ajax.xml', '', function(rsp){
    alert (rsp)
    var obj = xui.XML.xml2json(xui.XML.parseXML(rsp));
    xui.pop(obj.message);
},function(errMsg){
    xui.alert(errMsg)
}).start();
```

4.8. An overall example

The following is an overall example for data exchanging.



how to use linb.absIO to interact with service

	get	post	post file	cross domain get	cross domain post
linb.Ajax	yes	yes	no	no	no
linb.SAjax	yes	no	no	yes (best for "return big data")	no
linb.IAjax	yes	yes	yes	yes	yes

Request data:

```
{
   key:'test',
   para:{
      p1:'client_set'
   }
```

Response data:

```
{"p1":"client_set","p2":"server_set","time":"2009-07-23 03:48:55","rand":"03-48-55hjvw63a7kenm3h7wr"}
```

Chapter3/io/index.html

Chapter 5. Distributed UI

Sometimes, especially in larger applications, we maybe save a large “not frequently used” UI Class into a separate file. This file will not be loaded at the beginning.

When the application needs to show the UI, the program will automatically load code from the "separate file". It is so called "distributed UI". This "distributed UI" file can be in your server, or in different domain remote servers.

5.1. Shows dialog from a remote file

There’s a file “Module3.js” in folder “chapter4\distributed\App\js\”, “Module3.js” includes a Class named “App.Module3”. Let’s try to call it.

Input:

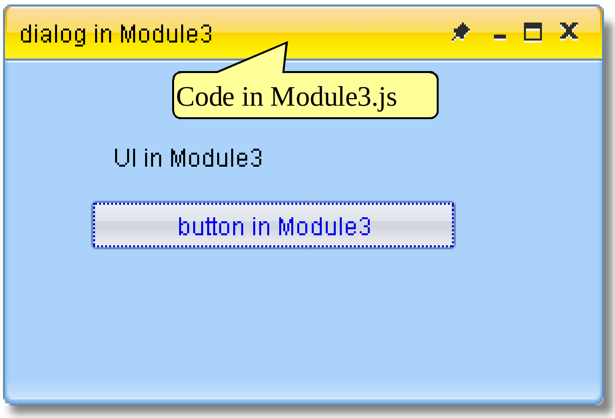
```
Namespace("App");
xui.include("App.Module3",
        xui.getPath("chapter4/distributed/App/js/","Module3.js"),
        function(){
                var ins=new App.Module3();
                ins.show();
        },function(){
                xui.alert("fail");
        }
);
```

Namespace is “App”

Dynamic asynchronous remote file loading

Create instance and show it

Output:



Code in Module3.js

And try to load code and create UI from a difference domain.

```
Namespace("App");
xui.include("App.Module3",
"",
function(){
            var ins=new App.Module3();
            ins.show();
      },function(){
            xui.alert("fail");
      }
);
```

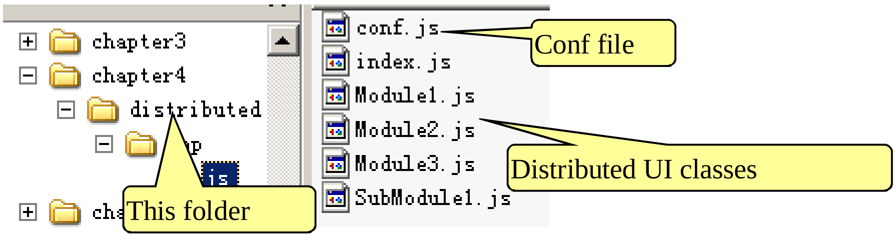
A difference domain

5.2. xui.Com and xui.ComFactory

In fact, most of the actual business applications will not load code from a foreign domain. From another perspective, most of "Distributed UI" files are put in the application directory.
In this case, we can use xui.Com and xui.ComFactory to load those “distributed UI”. In order to use this approach, all those Classes must be derived from the xui.Com, named according to specified rules, and put into the specified directory.

xui.ComFactory implements a management mechanism for the xui.Com. It can follow a specified rule (finding file path from the class name) to load code from a remote file.

There ‘s an overall example in “chapter4/distributed”, we can browse it for detail.



5.2.1. xui.ComFactory config

In conf.js:

```
CONF={ComFactoryProfile:
{
        module1:{
                cls:'App.Module1',
                children:{
                        tag_SubModule1:'submodule1'
                }
        },
        submodule1:{
                cls:'App.SubModule1'
        }
}}
```

module1 is an “Aoo.Module1” Class

module1 has a child submodule1

submodule1 is a “Aoo.SubModule1” Class

Loading this configuration to xui.ComFactory:

```
        xui.ComFactory.setProfile(CONF.ComFactoryProfile);
```

5.2.2. xui.Com.Load

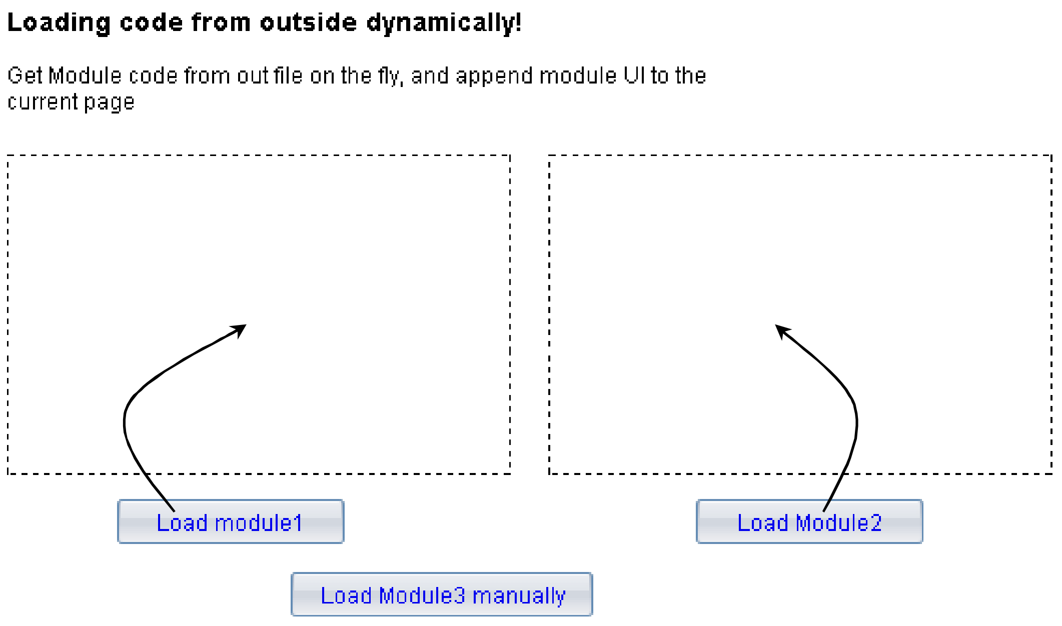
In file index.html,

```
        xui.Com.load ('App');
```

To load and show the firs UI Class

The above code will try to find file named “index.js” from “distributed/App/js/”, create an instance (new App), and show the instance to DOM.

Output:

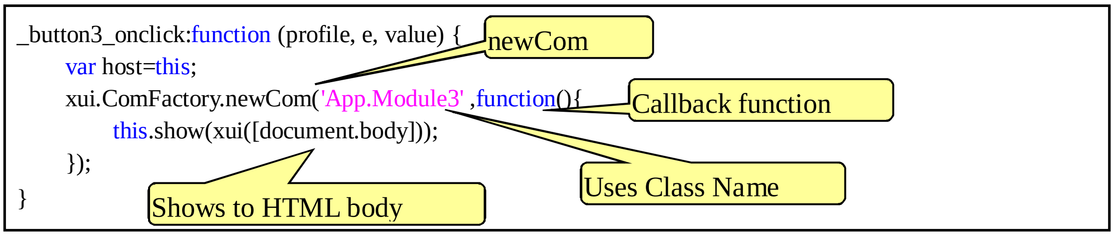


Loading code from outside dynamically!

Get Module code from out file on the fly, and append module UI to the current page

5.2.3. newCom and getCom

In index.js, onClick event for “Load module3 manually” button is:



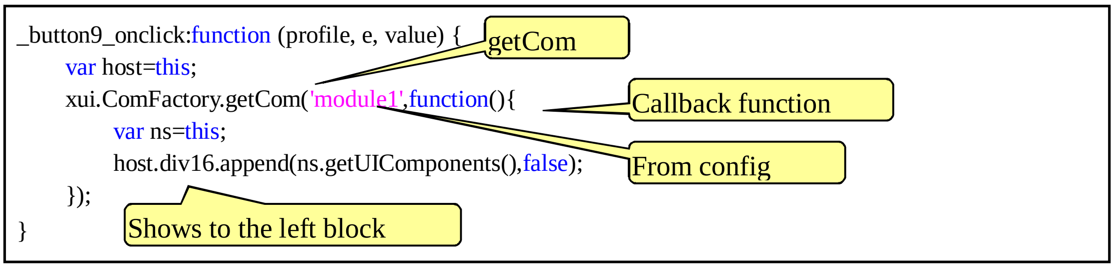
```
_button3_onclick:function (profile, e, value) {
      var host=this;
      xui.ComFactory.newCom('App.Module3' ,function(){
            this.show(xui([document.body]));
      });
}
```

[xui.ComFactory.newCom(“App.Module3”..], will:

- find file “Module3.js” in “distributed/App/js/”
- load code from file “Module3.js” ;
- create new instance,;
- call the callback function.

Note: newCom use “Class Name” to load code.

onClick event for “Load module1” button is:

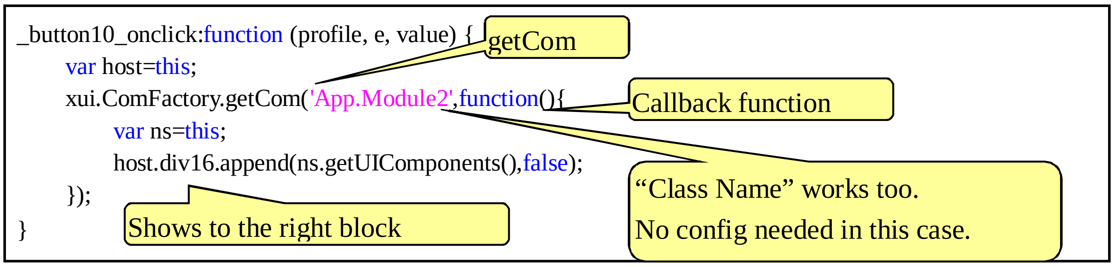


```
_button9_onclick:function (profile, e, value) {
      var host=this;
      xui.ComFactory.getCom('module1',function(){
            var ns=this;
            host.div16.append(ns.getUIComponents(),false);
      });
}
```

[xui.ComFactory.newCom(“module1”..], will:

- find config from xui.ComFactory
- find file “Module1.js” in “distributed/App/js/”
- load code from file “Module1.js” ;
- create new instance,;
- call the callback function.

onClick event for “Load module2” button is:



```
_button10_onclick:function (profile, e, value) {
      var host=this;
      xui.ComFactory.getCom('App.Module2',function(){
            var ns=this;
            host.div16.append(ns.getUIComponents(),false);
      });
}
```

By default, the instance created by "getCom" is singleton, and will be cached in inb.ComFactory.

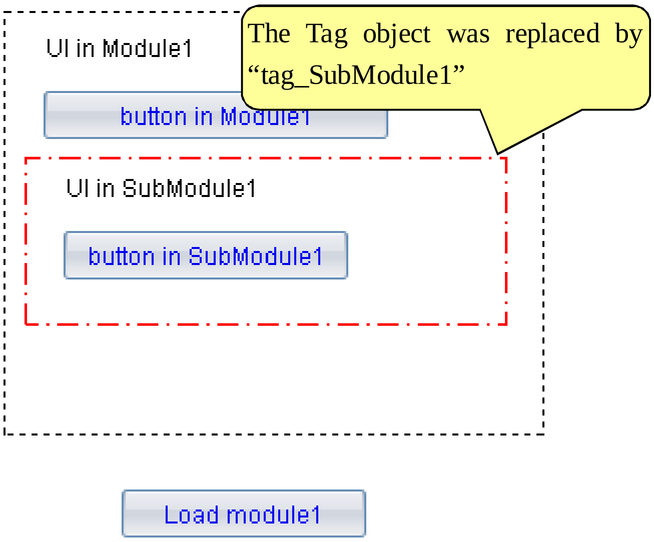
5.2.4. xui.UI.Tag

There’s a xui.UI.Tag object in file Module1.js:

```
host.panelMain.append((new xui.UI.Tag)
        .host(host,"tag2")
        .setLeft(20)
        .setTop(70)
        .setWidth(218)
        .setHeight(98)
        .setTagKey("tag_SubModule1")
);
```

Module name in configuration

Here, this Tag object configures size and position properties for module “tag_SubModule1”. When the instance of Module1 was created, according to the Tag object’ info, system will load the “tag_SubModule1” automatically, and set size and position properties to it. Then, system will replace the Tag object with “tag_SubModule1” object, and destroy the Tag object.



The Tag object was replaced by “tag_SubModule1”

5.2.5. Destroy com

Call com’s **destroy()** function to destroy the Class instance;
Call **Class.destroy(“class name”)** to destroy the Class itself.
If you used “getCom(‘module name’)” to create an com instance, you have to call “**xui.ComFactory.setCom (‘module name’, null)**” to clear that cache.

5.2.6. If com exists in memory

If a com exists in memory already, we can call it directly:

```
xui('body').append(new App.Acom);
```

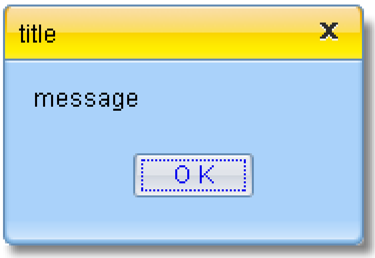
Chapter 6. Some fundamental things

6.1. Pop-up window

6.1.1. alert window

```
xui.alert('title','message',function(){
      xui.message('You close this window!')
}, 'O K', 50, 100);
```

Fired after user close the window



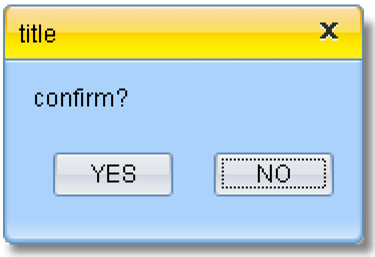
a

6.1.2. confirm window

```
xui.confirm('title','confirm?',function(){
            xui.message('You confirmed it')
      },
      function(type){
            xui.message(" You didn’t cofirm it -" + type)
      },'YES', 'NO',50,100);
```

Fired when user click “YES”

Fired when user click “NO” or click close button

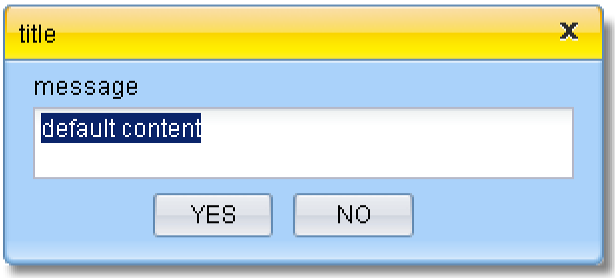


6.1.3. prompt window

```
xui.prompt('title', 'message','default content',function(msg){
        xui.message('You input - ' + msg)
    },function(){
        xui.message(" You cancel it")
    },'YES', 'NO',50,100);
```

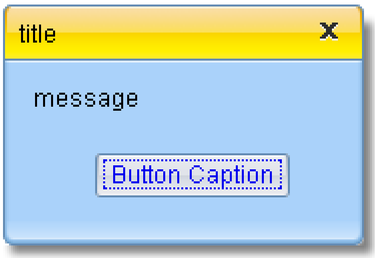
Fired when user click "YES"

Fired when user click "NO" or click close button



6.1.4. pop window

```
xui.pop('title', 'message','Button Caption',50,100);
```



6.2. Asynchronous execution

6.2.1. asyRun

_.asyRun is a wrapper for settimeout.

```
_.asyRun(function(arg1,arg2){
    xui.pop("this===xui:"+(this===xui), arg1+":"+arg2)
},
500,
['arg1','arg2'],
xui)
```

Function body

Delay 500 ms

parameters

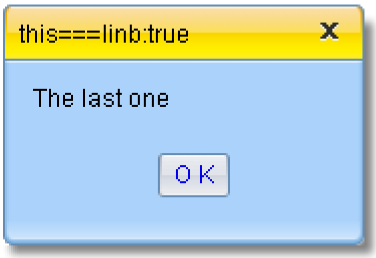
scope

6.2.2. resetRun

_.asyRun is a wrapper for set timeout too. But it has an unique id. When you set another function with the same id, the latter will cover the former.

```
_.resetRun('key1',function(arg){
        xui.pop("this===xui:"+(this===xui), arg)
},500,['The previous one'],xui)

_.resetRun('key1',function(arg){
        xui.pop("this===xui:"+(this===xui), arg)
},500,['The last one'],xui)
```

The latter one will be executed



this===linb:true

The last one

6.3. Skin switcher

6.3.1. Switch skin for whole application

There are three default system skins in CrossUI : default, vista and aqua. You can use xui.setTheme to switch the skin.

```
xui.setTheme("vista")
```

Dynamically (no page refresh)

6.3.2. Change skin for a single control

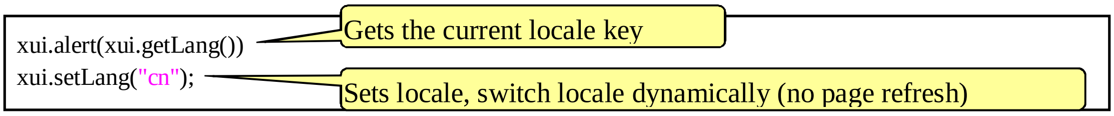
It’s a fine-grained mechanism.

```
ctl_button.setTheme("custom")
```

Only for “ctl_button”

In this case, developer needs to define CSS class for this "custom".

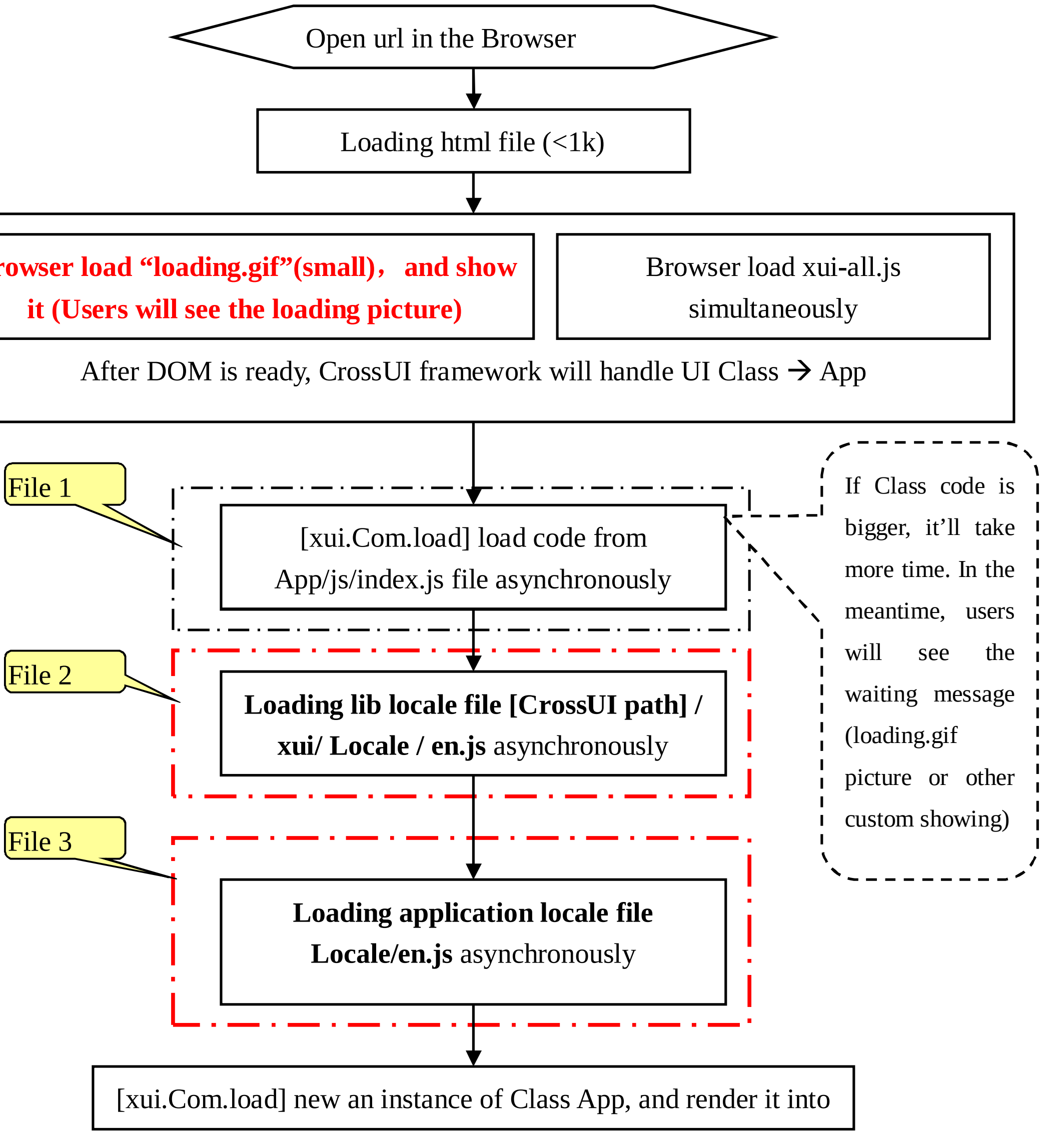
6.4. Locale switcher



```
xui.alert(xui.getLang())
xui.setLang("cn");
```

Gets the current locale key
Sets locale, switch locale dynamically (no page refresh)

Example “chapter5\lang” loading process:



Open url in the Browser
Loading html file (<1k)
Browser load “loading.gif”(small)， and show it (Users will see the loading picture)
Browser load xui-all.js simultaneously
After DOM is ready, CrossUI framework will handle UI Class → App
File 1
[xui.Com.load] load code from App/js/index.js file asynchronously
If Class code is bigger, it’ll take more time. In the meantime, users will see the waiting message (loading.gif picture or other custom showing)
File 2
Loading lib locale file [CrossUI path] / xui/ Locale / en.js asynchronously
File 3
Loading application locale file Locale/en.js asynchronously
[xui.Com.load] new an instance of Class App, and render it into

6.5. DOM Manipulation

Class “xui.Dom” is a wrapper for cross-browser DOM Manipulation. It can: create / remove elements; manage elements’ attributes; manage elements’ CSS; manage elements’ events.

6.5.1. Node generation and insertion

Input:

```
var firstNode;
xui('body').append(firstNode=xui.create('<div style="border:solid 1px">the first div</div>'));
firstNode.append('input')

.last().attr('value','append')

.parent()

.prepend('<button>prepend</button>')

.addPrev('<div style="border:solid 1px">prev div</div>')

.addNext('<div style="border:solid 1px"> div</div>')
```

It’s the fist node

Appends to firstNode

Finds input, and sets

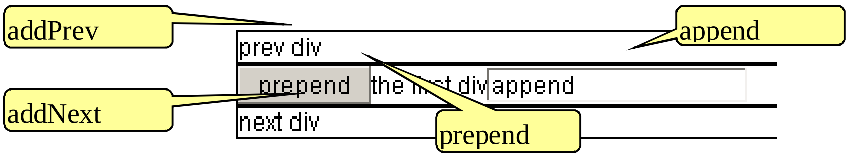
Returns to firstNode

Prepend a button

Adds to previous sibling

Adds to next sibling

output:



6.5.2. Attributes and CSS

Input:

```
var node;
xui('body').append(node=xui.create('div'));

node.html('content<input value="ini">');

_.asyRun(function(){
    node.css('border','solid 1px');
},1000);

_.asyRun(function(){
    xui.message(node.css('fontSize'))
    node.css({background:'#00ff00',fontSize:'16px'});
},2000);

_.asyRun(function(){
    node.attr('style','border:none;font-size:18px;')
},3000);

_.asyRun(function(){
    xui.message(node.last().attr('value'))
    node.last().attr('value','updated');
},4000);
```

Sets contents

Updates CSS border

Gets CSS fontSize

Updates fontSize and backgorund

Updates all style

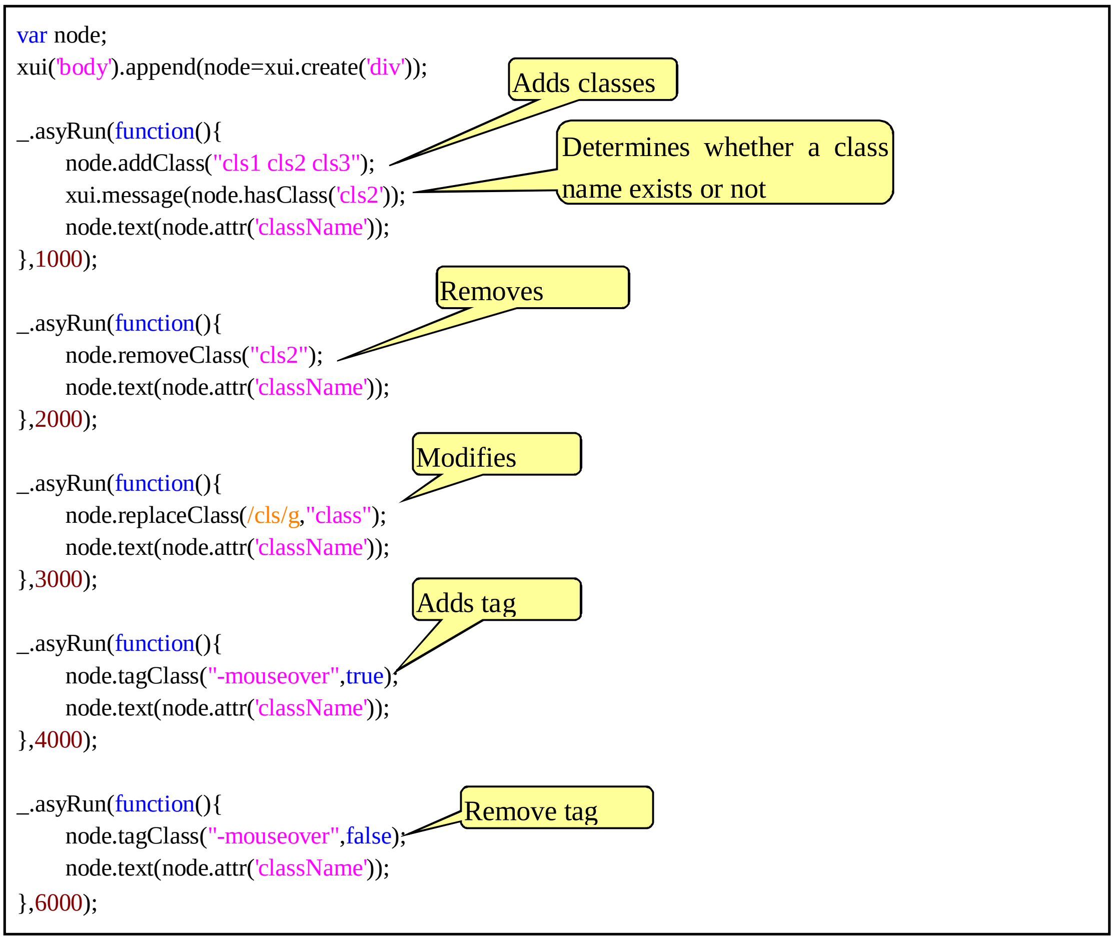
Gets input’ value attr

Updated input’s value attr

6.5.3. className

There are five function to handle CSS className:

- hasClass: Determines whether a specified class exists or not
- addClass: Adds classes to the current DOM nodes
- removeClass: Removes classes from the current DOM nodes
- replaceClass: Replaces classes for the current DOM nodes
- tagClass: Adds/Removes a tag to all classes of the current DOM nodes



```
var node;
xui('body').append(node=xui.create('div'));

_.asyRun(function(){
    node.addClass("cls1 cls2 cls3");
    xui.message(node.hasClass('cls2'));
    node.text(node.attr('className'));
},1000);

_.asyRun(function(){
    node.removeClass("cls2");
    node.text(node.attr('className'));
},2000);

_.asyRun(function(){
    node.replaceClass(/cls/g,"class");
    node.text(node.attr('className'));
},3000);

_.asyRun(function(){
    node.tagClass("-mouseover",true);
    node.text(node.attr('className'));
},4000);

_.asyRun(function(){
    node.tagClass("-mouseover",false);
    node.text(node.attr('className'));
},6000);
```

6.5.4. Dom events

There are three groups of event functions are designed for a DOM event in CrossUI: [before-], [on-] and [after-].

- xui(/**/).onClick([function], 'label') => adds the [function] to [onclick]group;
- xui(/**/).onClick([function]) => removes all event functions in [onclick] group, and adds the [function] to [onclick] group;
- xui(/**/).onClick(null, 'label') => removes the event function labeled with 'label' from the [onclick] group;
- xui(/**/).onClick(null) => removes all event functions in [onclick] group;

- xui(/**/).onClick(null,null,true) => removes all event functions in [beforeclick] group, [onclick] group and [afterclick] group;
- xui(/**/).onClick() => fire event, executes all event functions in [onclick] group in order. If any of those functions returns [false], the remaining functions will be ignored;
- xui(/**/).onClick(true) => fire event, executes all event functions in [beforeclick] group, [onclick] group and [afterclick] group in order;

```
var node;
xui('body').append(node=xui.create("<button>click me</button>"));

node.onClick(function(){
        alert('onClick');
        return false;
})
.beforeClick(function(){
        alert('beforeClick');
})
.afterClick(function(){
        alert('afterClick');
});

node.onClick(true);

_.asyRun(function(){
        node.onClick(null);
        node.onClick(true);
},2000);
```

Adds a onClick event

Adds a beforeClick event

Adds an afterClick event

Fires all click events. Since onClick returns false, afterClick will not be fired.

Removes onClick event;

Fires all click events. Since onClick was removed, afterClick will be fired this time.

6.5.5. Node Drag&Drop

Input:

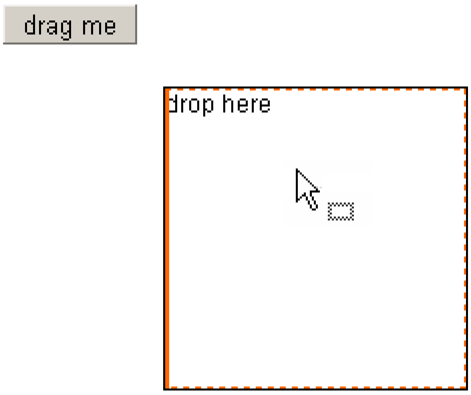
```
var btn,div;
xui('body').append(btn=xui.create("<button>drag me</button>"))
.append(div=xui.create("<div style='position:absolute;left:100px;top:100px;border:solid 1px;width:150px;
height:150px;display:block;'> drop here </button>"))

btn.draggable(true,{dragType:'icon'},'dragkey','dragdata');
div.droppable(true,'dragkey')
.onDrop(function(){
        xui.alert(xui.DragDrop.getProfile().dragData);
});
```

Sets draggable

Sets droppable

onDrop event

Output:



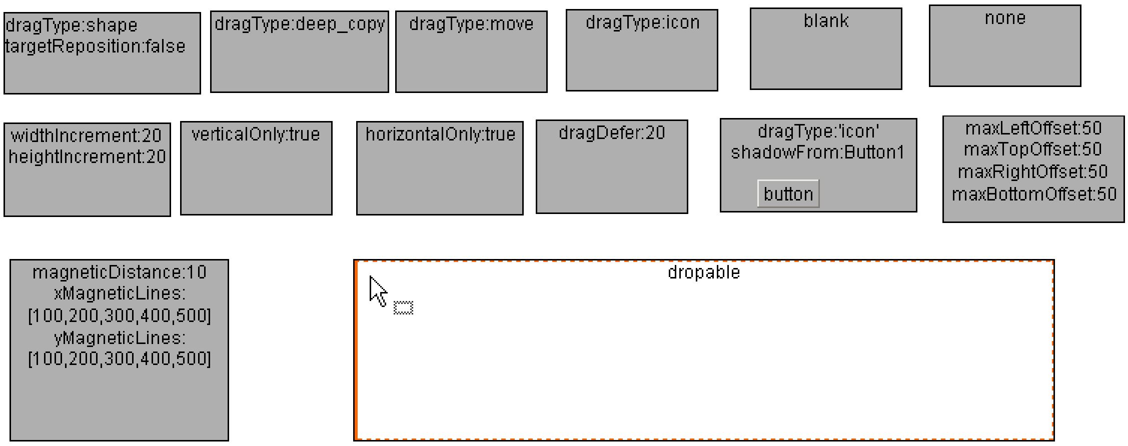
6.5.5.1. Drag&Drop profile

The "draggable" function's second parameter is Drag&Drop profile object. It's a key/value pairs. In dragging process, the Drag&Drop profile object can be got by xui.DragDrop.getProfile(). The profile object:

- dragType: 'move','copy','deep_copy','shape','icon', 'blank' or 'none', Default is 'shape';
- shadowFrom: DOM element or xui.Dom Object. It's valid when dragType=='icon';
- targetReposition: Boolean, does dd reset the target position, Default is [true];
- dragIcon: String, the drag icon image path, Default is [xui.ini.path+'ondrag.gif'];
- magneticDistance: Number, the magnetic distance, Default is 0;
- xMagneticLines: Array of Number, the magnetic line values in horizontal dir, Default is [];
- yMagneticLines: Array of Number, the magnetic line values in vertical dir, Default is [];
- widthIncrement: Number, the width increment in horizontal dir, Default is 0;
- heightIncrement: Number, the height increment in vertical dir, Default is 0;
- dragDefer: Number, when [xui.DragDrop.startDrag] is called, the real drag action will be triggered after [document.onmousemove] runs [dragDefer] times, Default is 0;
- horizontalOnly:Boolean, drag horizontal dir only, Default is [false];
- verticalOnly: Boolean, drag vertical dir only, Default is [false];
- maxBottomOffset</strong>:Number, the offset between [the restricted bottom] and [the current mouse Y], for mouse restricted region, Default is [null];
- maxLeftOffset</strong>:Number, the offset between [the restricted left] and [the current mouse X], for mouse restricted region, Default is [null];
- maxRightOffset</strong>:Number, the offset between [the restricted right] and [the current mouse X], for mouse restricted region, Default is [null];
- maxTopOffset: Number, the offset between [the restricted top] and [the current mouse Y], for mouse restricted region, Default is [null];
- targetNode: DOM element or xui.Dom Object, the drag target node;
- targetCSS: Number, the drag target node's CSS key/value Object, Default is [null];
- dragKey: String, the drag key, Default is [null];
- dragData: Object, the drag data, Default is [null];
- targetLeft: Number, the drag target node's CSS left, Default is [null];
- targetTop: Number, the drag target node's CSS top, Default is [null];
- targetWidth: Number, the drag target node's CSS width, Default is [null];

- targetHeight: Number, the drag target node's CSS height, Default is [null];
- targetOffsetParent: xui.Dom Object, the drag target node offsetParent node, Default is [null];
- dragCursor: 'none', 'move', 'link', or 'add', the drag cursor key; [readonly]
- x: Number, current X value of mouse; [readonly]
- y: Number, current Y value of mouse; [readonly]
- ox: Number, original X value of mouse; [readonly]
- oy: Number, original Y value of mouse; [readonly]
- curPos: {left:Number,top:Number}, current CSS pos of the dragging node [readonly]
- offset: {x:Number,y:Number}, offset from now to origin [readonly]
- isWorking: Boolean, is dd working or not? [readonly]
- restrictedLeft: Number, the calculated restricted left value; [readonly]
- restrictedRight: Number, the calculated restricted right value; [readonly]
- restrictedTop: Number, the calculated restricted top value; [readonly]
- restrictedBottom: Number, the calculated restricted bottom value; [readonly]
- proxyNode: xui.Dom Object, the proxy Object; [readonly]
- dropElement: String, the target drop element DOM id. [readonly]

There is an DD overall example in **chapter3/dd/ddProfile.html.**



dragType:shape
targetReposition:false

dragType:deep_copy

dragType:move

dragType:icon

blank

none

widthIncrement:20
heightIncrement:20

verticalOnly:true

horizontalOnly:true

dragDefer:20

dragType:'icon'
shadowFrom:Button1

maxLeftOffset:50
maxTopOffset:50
maxRightOffset:50
maxBottomOffset:50

magneticDistance:10
xMagneticLines:
[100,200,300,400,500]
yMagneticLines:
[100,200,300,400,500]

6.5.5.2. Events in Drag&Drop

For that node in dragging,

- onDragbegin
- onDrag
- onDragstop

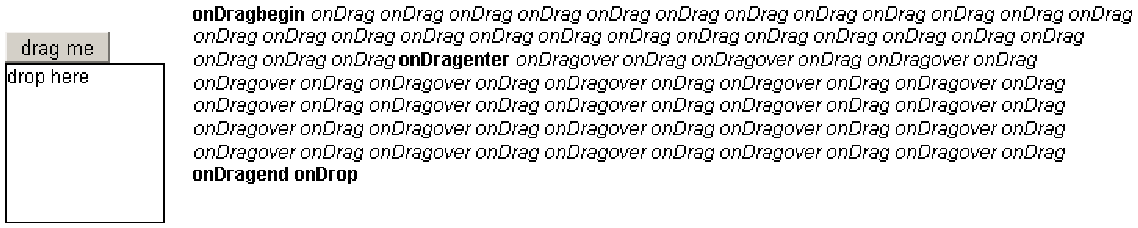
For that droppable node,

- onDragenter
- onDragleave
- onDragover
- onDrop

Input:

```
var btn,div,elist;
xui('body').append(btn=xui.create("<button>drag me</button>"))
.append(div=xui.create("<div style='border:solid 1px;width:100px;height:100px;'>drop here</button>"))

xui('body').append(elist=xui.create('<div
style="position:absolute;left:140px;top:40px;width:600px;height:400px;overflow:auto;"></div>'))

btn.dragable(true,{dragType:'icon'},'dragkey','dragdata')
.onDragbegin(function(){
        elist.append('<strong>onDragbegin </strong>');
})
.onDrag(function(){
        elist.append('<em>onDrag </em>');
})
.onDragstop(function(){
        elist.append('<strong>onDragend </strong>');
})

div.dropable(true,'dragkey')
.onDragenter(function(){
        elist.append('<strong>onDragenter </strong>');
})
.onDragover(function(){
        elist.append('<em>onDragover </em>');
})
.onDragleave(function(){
        elist.append('<strong>onDragleave </strong>');
})
.onDrop(function(){
        elist.append('<strong>onDrop </strong>');
});
```

Output:



```
onDragbegin onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag
onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag onDrag
onDrag onDrag onDrag onDragenter onDragover onDrag onDragover onDrag onDragover onDrag
onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag
onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag
onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag
onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag onDragover onDrag
onDragend onDrop
```

6.6. xui.Template

Xui.Template is a completely independent UI wrapper. It doesn’t depend on xui.UI Class and all its derived Classes.

6.6.1. example 1

xui.Template includes three aspects: template, properties and events:

Input:

```
var tpl=new xui.Template;
tpl.setTemplate("<div [event]>{con}</div>")
//tpl.setTemplate({root:"<div [event]>{con}</div>"})
.setProperties({
        con:'click me'
})
.setEvents({
        root:{
                onClick:function(){
                        xui.alert('Hi');
                }
        }
})
.show()
```

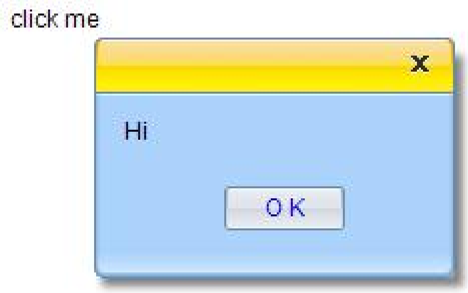
For event

Sets template

Sets properties

Sets events

Output:



Hi

6.6.2. example 2

Input:

```
(tpl=new xui.Template)
.setTemplate({
        root : "<div style='width:200px;border:solid 1px;'><h3>{head}</h3><ul>{items}</ul><div 
style='clear:both;'
        items: "<li [event]   style='padding:4px;border-top:dashed 1px;'><div ><div><a href='{href}'><img 
src='{src}'/><div>{price}</div></a></div></div><div><a 
href='{href}'><h4>{title}</h4><div>{desc}</div></a></div></li>"
})
.setEvents({
        items:{
                onMouseover:function(profile,e,src){
                        xui.use(src).css('backgroundColor','#EEE');
                        //Tips
                        var item=profile.getItem(src),
                                tpl=new xui.Template({"root":"<div style='text-align:center;border:solid 
1px;background:#fff;'><h4>{title}</h4></div><div>{desc}</div>"},item),
                                html=tpl.toHtml();
                        xui.Tips.show(xui.Event.getPos(e),html);
                },
                onMouseout:function(profile,e,src){
                        xui(src).css('backgroundColor','transparent');
                        xui.Tips.hide();
                }
        }
})
.setProperties({
        head:"On sale products",
        items:[{ id:"a",    href:"#", price:"$ 18.99", title:"product #0", desc:"product #0 is on sale now!" },
                {id:"b", href:"#", price:"$ 23.99", title:"product #1", desc:"product #1 is on sale now!" },
                {id:"c", href:"#", price:"$ 23.99", title:"product #2", desc:"product #2 is on sale now!" }]
})
.show()
```

Sub template exists

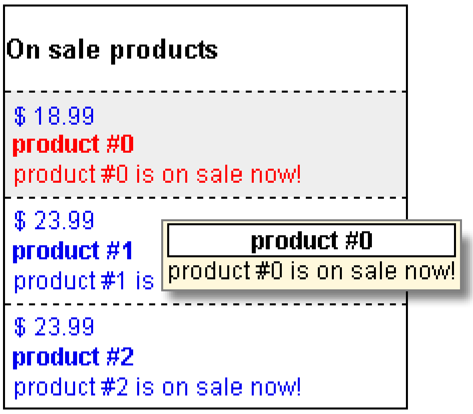
Here, “root” key is a must

Sets events in “items”

Sets properties in “root”

Sets properties in “items”

Output:



On sale products

$ 18.99
product #0
product #0 is on sale now!

$ 23.99
product #1
product #1 is

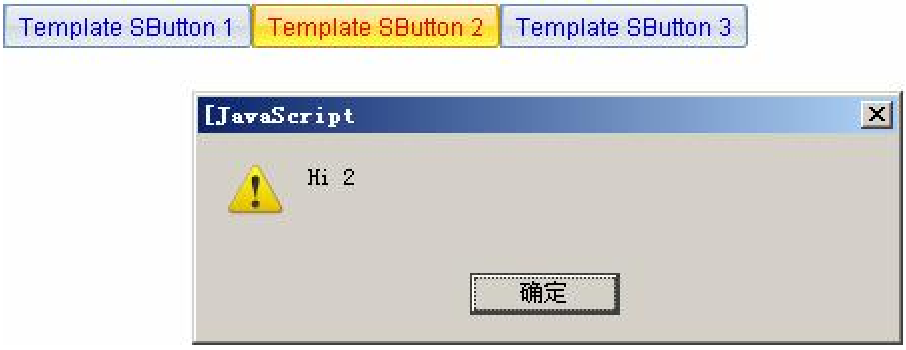
product #0
product #0 is on sale now!

$ 23.99
product #2
product #2 is on sale now!

6.6.3. A SButton based on xui.Template

“chapter5\SButton” is an example for creating a xui.UI.SButton like control based on xui.Template.

Output:

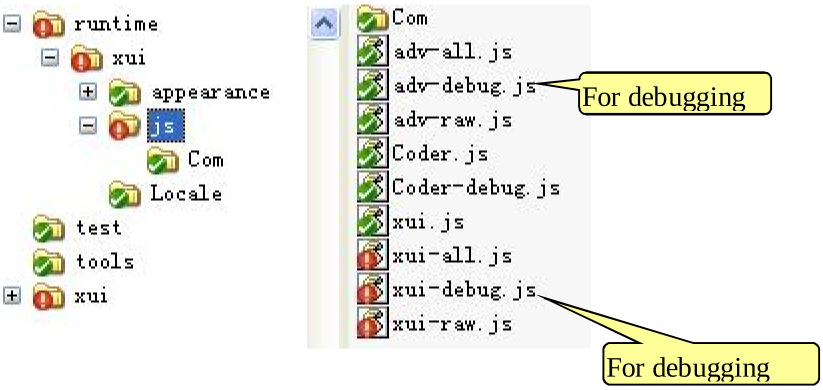


Hi 2

6.7. About Debugging

6.7.1. The code package for debugging

In folder “runtime/xui/js/”, All files ending with ”-debug.js” are for debugging purpose.



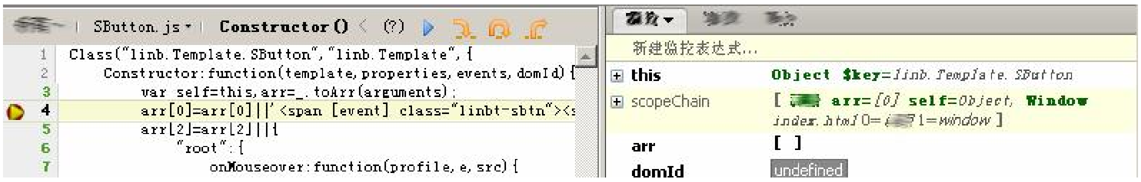
For debugging

For debugging

6.7.2. Debugging Tools

You can use Firebug in Firefox, developer tool in IE8, chrome or opera10 to debug JavaScript.

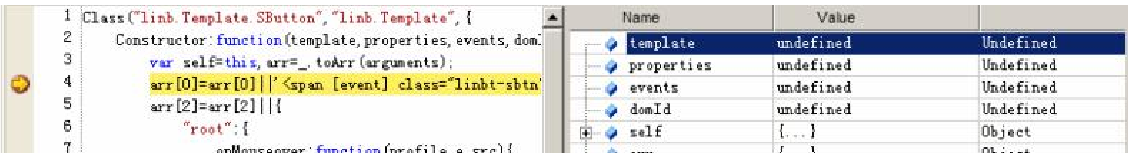
FireBug:



```
Class("linb.Template.SButton","linb.Template",{
    Constructor:function(template,properties,events,domId){
        var self=this,arr=_.toArr(arguments);
        arr[0]=arr[0]||'<span [event] class="linbt-sbtn"><s
        arr[2]=arr[2]||{
            "root":{
                onMouseover:function(profile,e,src){
```

this	Object $key=linb.Template.SButton
scopeChain	[arr=[0] self=Object, Window index.html 0= 1=window]
arr	[ ]
domId	undefined

Developer Tools in IE8:

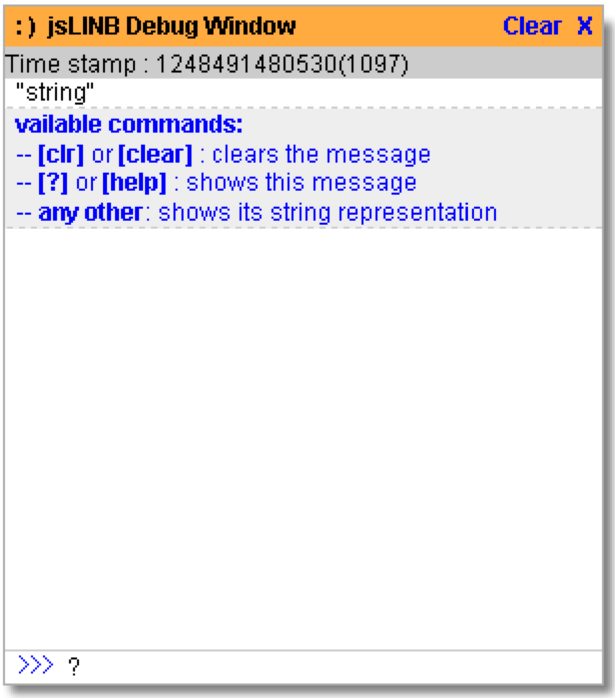


```
Class("linb.Template.SButton","linb.Template",{
    Constructor:function(template,properties,events,dom
        var self=this,arr=_.toArr(arguments);
        arr[0]=arr[0]||'<span [event] class="linbt-sbtn'
        arr[2]=arr[2]||{
            "root":{
                onMouseover:function(profile,e,src){
```

Name	Value	
template	undefined	Undefined
properties	undefined	Undefined
events	undefined	Undefined
domId	undefined	Undefined
self	{...}	Object

6.7.3. Monitor Tools

CrossUI has a variable monitor tools. You can call xui.log(“xxx”) to show the monitor window:



```
Time stamp : 1248491480530(1097)
 "string"
vailable commands:
-- [clr] or [clear] : clears the message
-- [?] or [help] : shows this message
-- any other: shows its string representation

>>> ?
```

Chapter 7. Some typical issues

7.1. Layout

7.1.1. Docking

Input:

```
xui.create('Block',{dock:"top",
        height:80,html:'top'
}).show();
xui.create('Block',{dock:"bottom",
        height:30,html:'bottom'
}).show();
xui.create('Block',{dock:"left",
        width:150,html:'left'
}).show();
xui.create('Block',{dock:"right",
        width:150,html:'right'
}).show();
xui.create('Block',{dock:"fill",
        html:'main',
        dockMargin:{left:10,right:10,top:10,bottom:10}
}).show();
```

At top

At

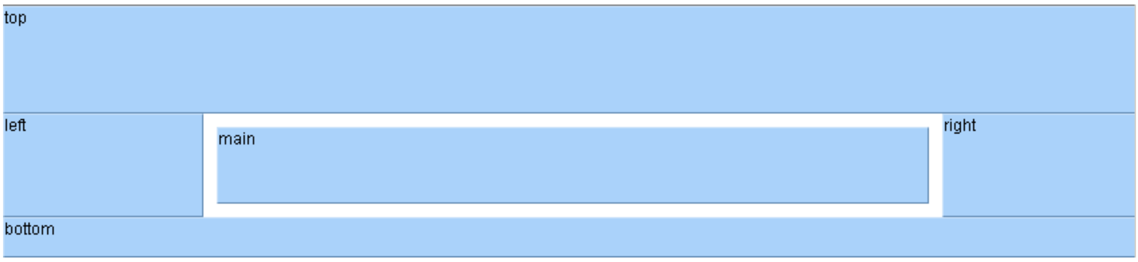
Left side

Right side

The main area

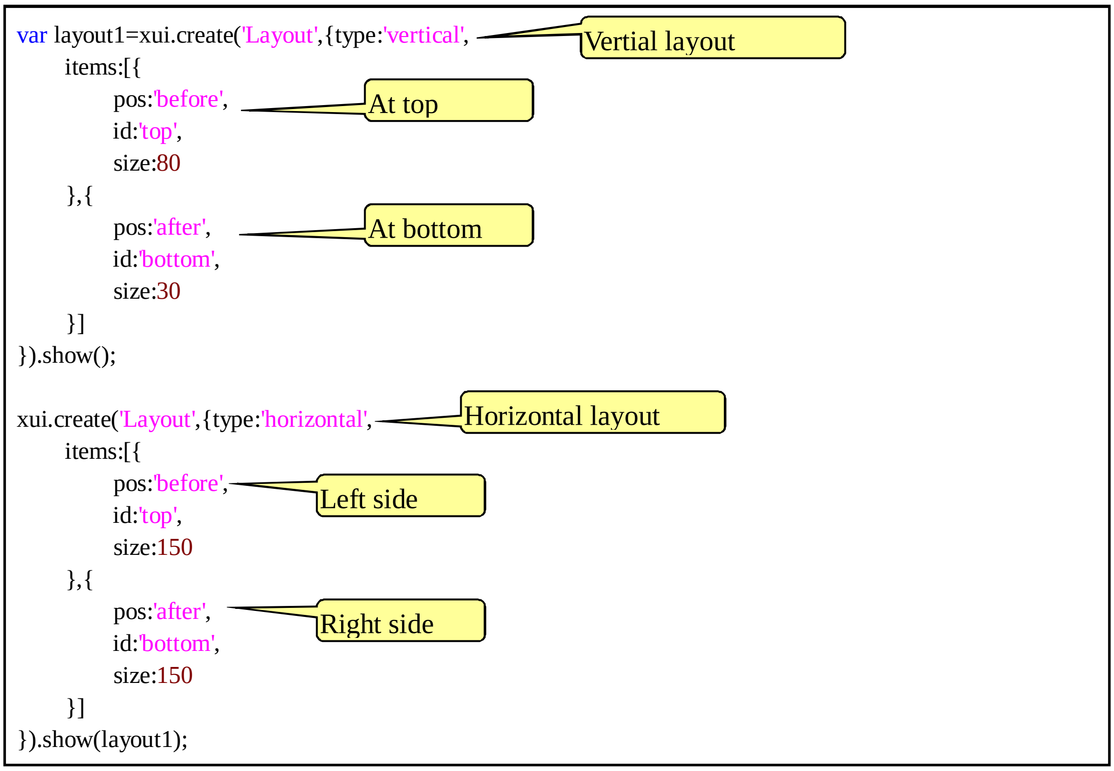
Sets docking margin

Output:



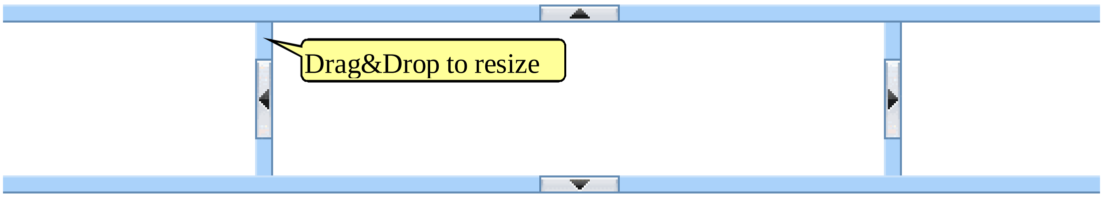
7.1.2. xui.UI.Layout

Input:



```
var layout1=xui.create('Layout',{type:'vertical',
        items:[{
                pos:'before',
                id:'top',
                size:80
        },{
                pos:'after',
                id:'bottom',
                size:30
        }]
}).show();

xui.create('Layout',{type:'horizontal',
        items:[{
                pos:'before',
                id:'top',
                size:150
        },{
                pos:'after',
                id:'bottom',
                size:150
        }]
}).show(layout1);
```

Vertial layout
At top
At bottom
Horizontal layout
Left side
Right side



7.1.3. Relative Layout

Input:

```
xui.create('Pane',{position:'relative',
        width:"auto",height:80,html:"the top div"
})
.setCustomStyle({"KEY":"border:solid 1px #888"})
.show()

var pane=xui.create('Pane',{position:'relative',
        width:"auto",height:"auto",
        html:"<strong>auto height</strong>"
})
.setCustomStyle({"KEY":"border:solid 1px #888"})
.show()

xui.create('Pane',{position:'relative',
        width:"auto",height:100,
        html:"<strong>bottom</strong>"
})
.setCustomStyle({"KEY":"border:solid 1px #888"})
.show()

xui.create("SButton")
.setLeft(140)
.setTop(30)
.setCaption("Add content")
.onClick(function(){
        pane.append(xui.create("Pane").setPosition("relative").setHeight(30).append("Input"))
})
.show()
```

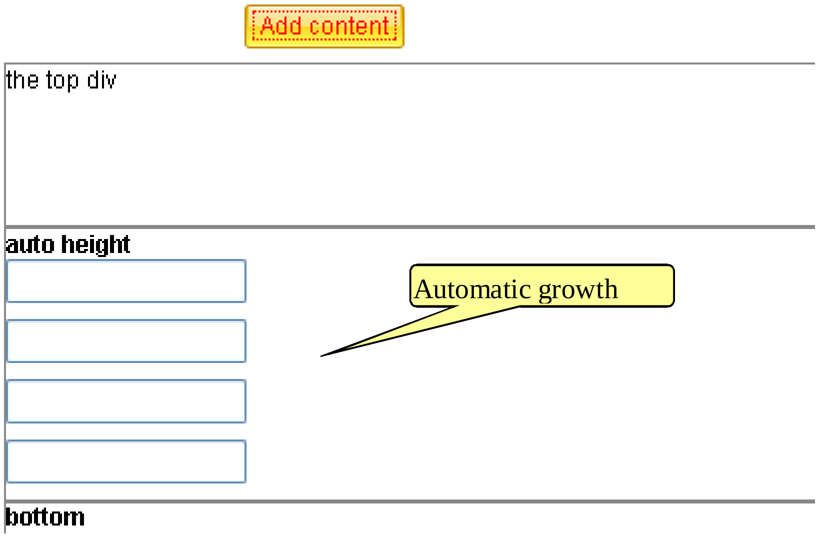
At top

Middle

At bottom

Adds contents

Output:



7.2. UI Control's Drag&Drop

7.2.1. Drag&Drop control among containers

Input:

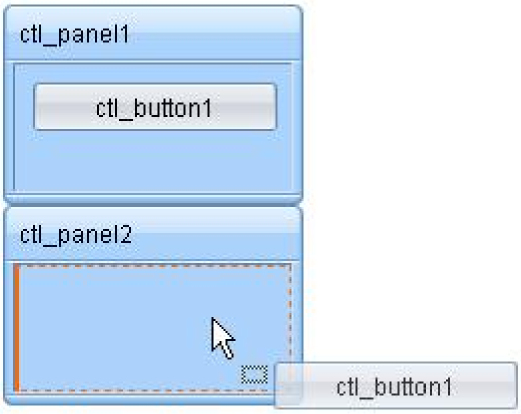
```
var panel1=xui.create('Panel',{position:'relative', dock:'none',width:150}).show();
var panel2=xui.create('Panel',{position:'relative', dock:'none',width:150}).show();
var btn=xui.create('Button',{left:10,top:10}).show(panel1);
var onDrop=function (profile, e, node, key, data) {
        var dd = xui.DragDrop.getProfile(), data = dd.dragData;
        if(data){
                var btn=xui.getObject(data);
                profile.boxing().append(btn.boxing());
        }
};

btn.draggable('iAny',btn.get(0).getId(),null,{shadowFrom:btn.getRoot()});
panel1.setDropKeys('iAny').onDrop(onDrop);
panel2.setDropKeys('iAny').onDrop(onDrop);
```

Sets onDrop function

Sets draggable

Sets droppable

Output:



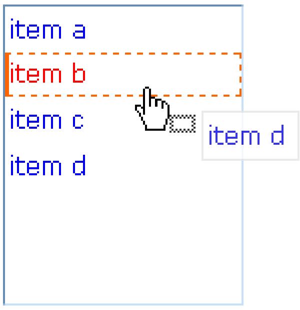
7.2.2. List sorting 1

Input:

```
xui.create("List",{
        items:["item a","item b","item c","item d"]
})
.setDragKey("list")
.setDropKeys("list")
.show()
```

Sets drag key and drop keys.

Output:



7.2.3. List sorting 2

Input:

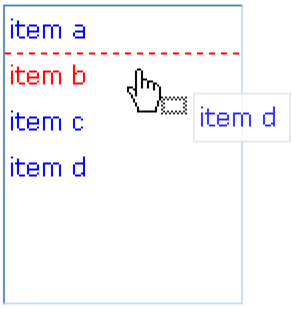
```
xui.create("List",{
        items:["item a","item b","item c","item d"]
})
.setDragKey("list")
.setDropKeys("list")
.onDropMarkShow(function(profile,e,src,key,data,item){
        if(item){
                xui.DragDrop.setDragIcon('move');
                xui.DragDrop.setDropFace(null);
                profile.getSubNodeByItemId('ITEM',item.id).css('borderTop','dashed 1px');
                return false;
        }
})
.onDropMarkClear(function(profile,e,src,key,data,item){
        if(item){
                xui.DragDrop.setDragIcon('none');
                profile.getSubNodeByItemId('ITEM',item.id).css('borderTop','');
                return false;
        }
})
.show()
```

Sets drag key and drop keys

Custom appearance

Restores appearance

Output:



7.3. Form

7.3.1. Form 1

Input:

```
Class.destroy('App');
Class('App', 'xui.Com',{
    Instance:{
        iniComponents:function(){
            // [[code created by CrossUI UI Builder
            var host=this, children=[], append=function(child){children.push(child.get(0))};
            append((new xui.UI.SLabel)
                .setHost(host,"slabel1").setLeft(80).setTop(60).setWidth(44).setCaption("Name:") );
            append((new xui.UI.SLabel)
                . setHost(host,"slabel2").setLeft(80).setTop(90).setCaption("Age:").setWidth(44));
            append((new xui.UI.Input)
                .setHost(host,"iName").setLeft(130).setTop(60).setValueFormat("[^.*]").setValue("Jack"));
            append((new xui.UI.ComboInput)
                .setHost(host,"iAge").setLeft(130).setTop(90).setType("spin").setIncrement(1).setMin(20).setMax(60).setValue("35"));
            append((new xui.UI.SCheckBox)
                .setHost(host,"cFull").setLeft(130).setTop(130).setCaption("Full time"));
            append((new xui.UI.SButton)
                .setHost(host,"submit").setLeft(130).setTop(170).setCaption("SUBMIT").onClick("_submit_onclick"));
            return children;
            // ]]code created by CrossUI UI Builder
        },
        _submit_onclick:function (profile, e, src, value) {
            if(!this.iName.checkValid()){
                xui.alert('You must specify Name');
                return;
            }
            var name=this.iName.updateValue().getValue(), age=this.iAge.updateValue().getValue(),
                full=this.cFull.updateValue().getValue();
            xui.alert(_.serialize({name:name,age:age,full:full}))
    }}
});
(new App).show();
```

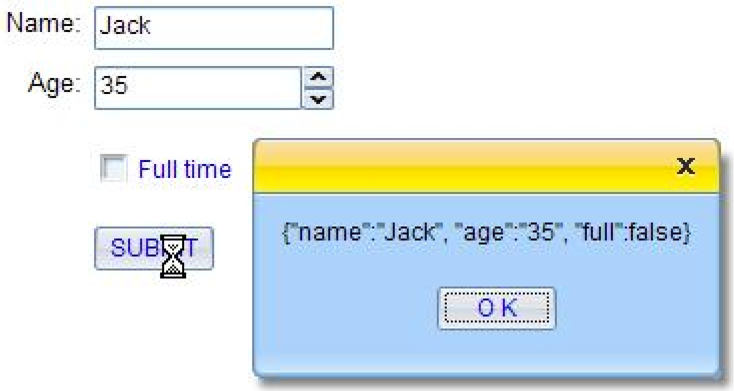
Code created by Designer

event

Form validation

Collects data

Output:



Name: Jack

Age: 35

□ Full time

{"name":"Jack", "age":"35", "full":false}

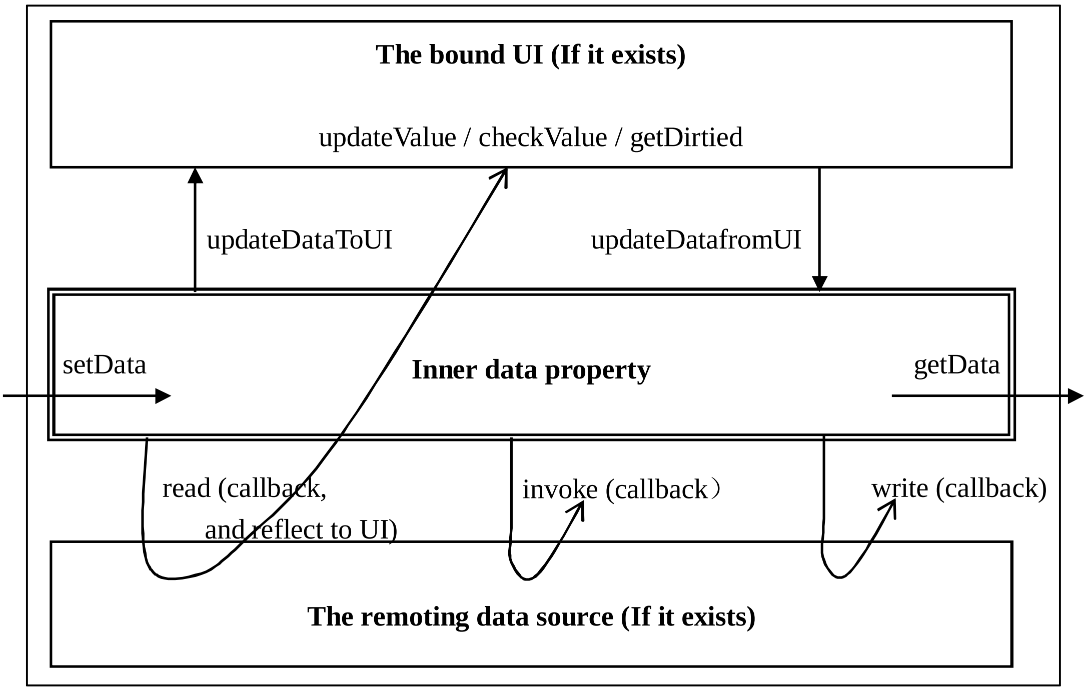
7.3.2. DataBinder

There are three types of data in DataBinder:

1. The inner data property:
   setData function: to get the inner data property;
   getData function: to set the inner data property;

2. The bound UI controls (if it exists):
   updateValue function: to update the bound UI controls "UI value" to inner value, and removed dirty marks;
   checkValue function: to check the bound UI controls;
   getDirtied function: to get the dirtied values from the bound UI controls;
   updateDataToUI function: to update the inner data to the bound UI controls;
   updateDatafromUI function: to update the inner data from the bound UI controls;

3. The remoting data source (if it exists):
   invoke function: invoke the remoting call;
   write function: invoke the writing type remoting call;
   read function: invoke the reading type remoting call; and update result data to the bound UI controls (updateDataToUI)；



Input:

```
Class.destroy('App');
Class('App', 'xui.Com',{
        Instance:{
                iniComponents:function(){
                        // [[code created by CrossUI UI Builder
                        var host=this, children=[], append=function(child){children.push(child.get(0))};
                        append((new xui.DataBinder).setHost(host,"binder").setName("binder"))
                        append((new xui.UI.SLabel)
                                .setHost(host,"slabel1").setLeft(80).setTop(60).se
                        append((new xui.UI.SLabel)
                                .setHost(host,"slabel2").setLeft(80).setTop(90).setCaption("Age:").setWidth(44));
                        append((new xui.UI.Input) .setDataBinder("binder").setDataField("name")
                                .setHost(host,"iName").setLeft(130).setTop(60).setValueFormat("[^.*]").setValue("Jack"));
                        append((new xui.UI.ComboInput) .setDataBinder("binder").setDataField("age")
                                .setHost(host,"iAge").setLeft(130).setTop(90).setType("spin").setIncrement(1).setMin(20).s
etMax(60).setValue("35"));
                        append((new xui.UI.SCheckBox) .setDataBinder("binder").setDataField("isfull")
                                .setHost(host,"cFull").setLeft(130).setTop(130).setCaption("Full time"));
                        append((new xui.UI.SButton)
                                .setHost(host,"submit").setLeft(130).setTop(170).setCaption("SUBMIT").onClick("_submit
_onclick"));
                        return children;
                        // ]]code created by CrossUI UI Builder
                },
                _submit_onclick:function (profile, e, src, value) {
                        if(!this.binder.checkValid()){
                                xui.alert('One or some invalid fields exits!');
                                return;
                        }
                        xui.alert(_.serialize(this.binder.updateDataFromUI().getData()))
        }}
});
(new App).show();
```

Code created by Designer

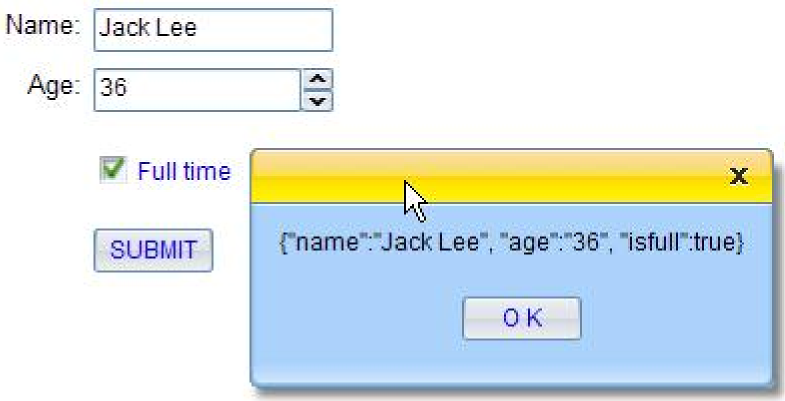
Adds a DataBinder, sets name property

Sets dataBinder and dataField to each control

Form validation

Collects data

Output:

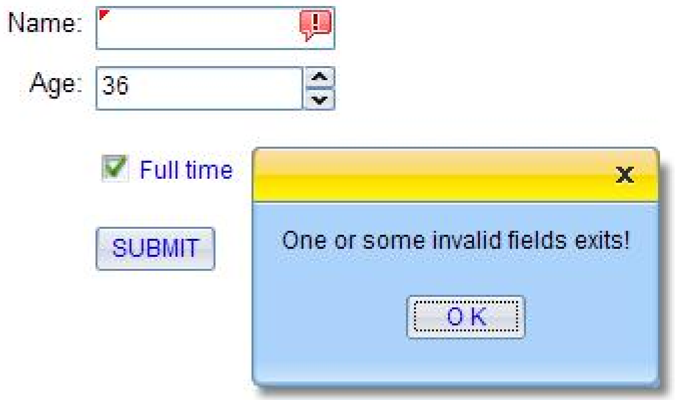


Name: Jack Lee

Age: 36

☑ Full time

{"name":"Jack Lee", "age":"36", "isfull":true}



Name:

Age: 36

☑ Full time

One or some invalid fields exits!

7.4. Custom UI Styles

7.4.1. Custom one instance only - 1

Input:

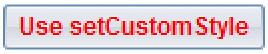
```
xui.CSS.remove("id","my_css");
xui.CSS.addStyleSheet(".xui-sbutton-custom-focus{font-weight:bold;color:#ff0000;}", "my_css");

(new xui.UI.SButton)
.setCaption("Use setCustomClass ")
.setTheme ("custom")
.show();
```

Theme key words

Adds CSS. You should put those into a CSS file in your real application

Sets theme to this instance

Output:



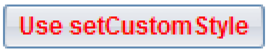
7.4.2. Custom one instance only - 2

Input:

```
(new xui.UI.SButton)
.setCaption("Use setCustomStyle")
.setCustomStyle({
        FOCUS:"font-weight:bold;color:#ff0000;"
})
.show();
```

Custom FOCUS node style

Output:



7.4.3. Custom one instance only - 3

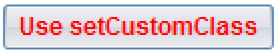
Input:

```
xui.CSS.remove("id","my_css");
xui.CSS.addStyleSheet(".my-class{font-weight:bold;color:#ff0000;}", "my_css");

(new xui.UI.SButton)
.setCaption("Use setCustomClass ")
.setCustomClass({
        FOCUS:"my-class"
})
.show();
```

Adds CSS. You should put those into a CSS file in your real application

Custom FOCUS node className

Output:



7.4.4. Custom one instance only - 4

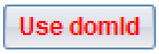
Input:

```
xui.CSS.remove("id","my_css");
xui.CSS.addStyleSheet("#myctrl1 .xui-sbutton-focus{font-weight:bold;color:#ff0000;}", "my_css");

(new xui.UI.SButton)
.setCaption("Use domId")
.setDomId("myctrl1")
.show();
```

Adds CSS. You should put those into a CSS file in your real application

Gives a domId

Output:



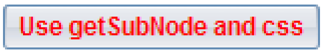
7.4.5. Custom one instance only - 5

Input:

```
(new xui.UI.SButton)
.setCaption("Use getSubNode and css ")
.onRender(function(profile){
        profile.getSubNode('FOCUS').css({
                fontWeight:'bold',
                color:'#ff0000'
        });
})
.show()
```

After it was rendered into DOM

Output:



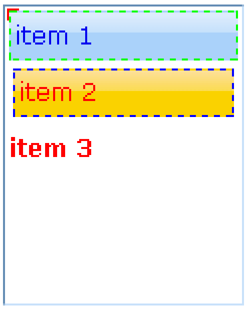
7.4.6. Custom one instance only - 6

Input:

```
xui.CSS.remove("id","my_css");
xui.CSS.addStyleSheet(".my-listitem{font-weight:bold;color:#ff0000;}", "my_css");

xui.create('List',{items:[{
        id:"item 1",
        itemStyle:"border:dashed 1px #00ff00;marg
},{
        id:"item 2",
        itemStyle:"border:dashed 1px #0000ff;margin:4px;"
},{
        id:"item 3",
        itemClass:"my-listitem"
}]}).show()
```

Adds CSS. You should put those into a CSS file in your real application

Output:



7.4.7. Custom style for an UI Class

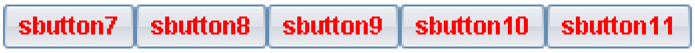
Input:

```
xui.CSS.remove("id","my_css");
xui.CSS.addStyleSheet(".xui-sbutton-focus{font-weight:bold;color:#ff0000;}", "my_css");

(new xui.UI.SButton({position:'relative'})).show();
(new xui.UI.SButton({position:'relative'})).show();
(new xui.UI.SButton({position:'relative'})).show();
(new xui.UI.SButton({position:'relative'})).show();
(new xui.UI.SButton({position:'relative'})).show();
```

Adds CSS. You should put those into a CSS file in your real application

All instances were changed

Output:

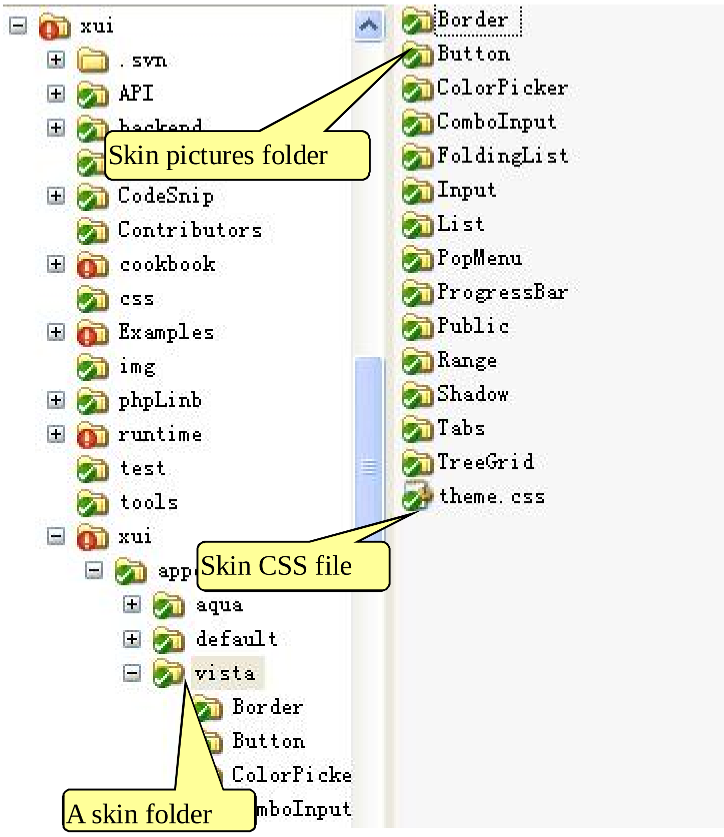


7.4.8. Custom style for all UI Class - skin

There are three system built-in skins in CrossUI: default, vista and aqua. You can use xui.setTheme to switch the skin. You can also add your own custom skin easily. Only two steps:

7.4.8.1. First: Copy

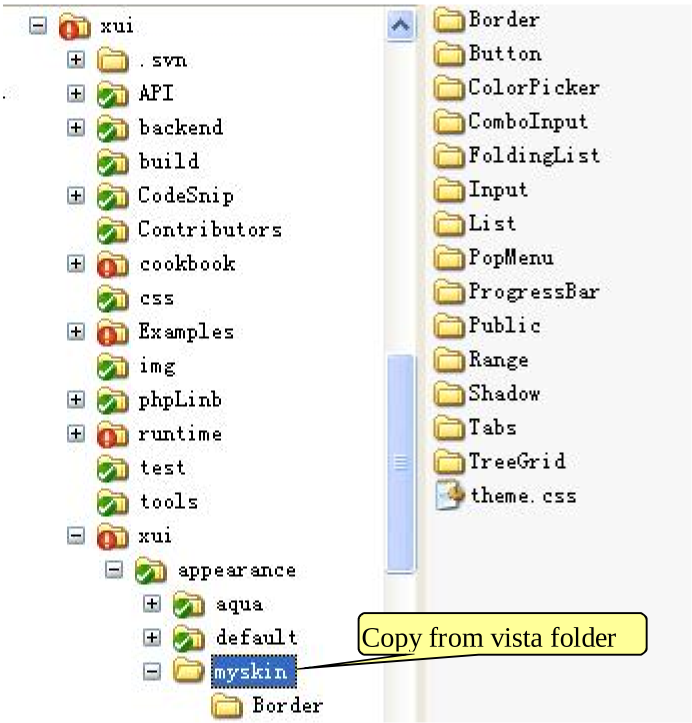
All skins are in “runtime/xui/appearance”, you can create an new folder (e.g. ‘myskin’), and copy all directories and files in an existing skin folder to it.



Skin pictures folder

Skin CSS file

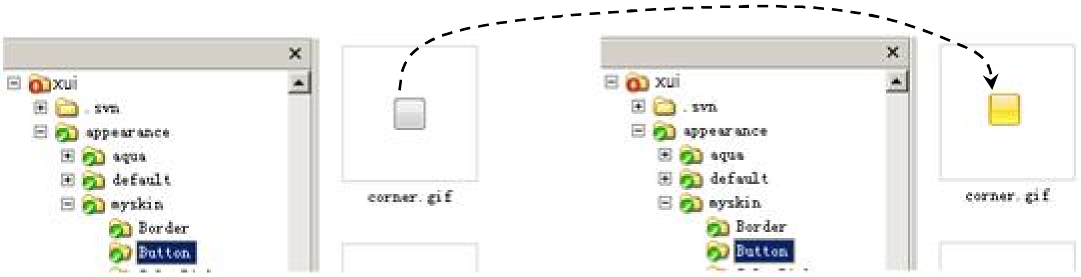
A skin folder



Copy from vista folder

7.4.8.2. Second: Little by little, modify pictures and CSS

For example, we modifies corner.gif file in Button folder.

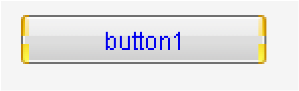


After that,

Input:

```
xui.create('Button').show();
_.asyRun(function(){
    xui.setTheme('myskin')
},2000);
```

Output:



The end