LINB3.0 Cookbook - Getting Started Guide

Preface

"jsLinb" is a Cross-Browser JavaScript framework with cutting-edge functionality for rich web application. And "Visual JS" is a web based tool for AJAX RIA application UI rapid design and involved scripts programming. With this powerful builder, developers can build your web application just like what you do in VB or Delphi.

Features & Resource

1. WYSIWYG GUI builder. Do everything by drag & drop.   Significant development time reduction.
2. Source code editor Integrated (Code Intellisense, Code folding, Syntax Check and Undo/Redo).
3. More than 40 common components, including Tabs, Dialog, TreeGrid, TimeLine and many other web GUI components.
4. Rich client-side API, works with any backend (php, .Net, Java, python) or static HTML pages. Extremely easily to build php application with php server side wrappers.
5. Wide cross-browser compatibility, IE6+, firefox1.5+, opera9+, safari3+ and Google Chrome.
6. Detailed Manual and Full API Documentation with tons of samples. Ever increasing code snippets.
7. Compatible with jquery, prototype, mootools and other frameworks.
8. Dual licenses - Commercial License and LGPL license both available.

Cookbooks are expected to include three parts: Getting Started Guide, Improved Tutorial, and Advanced Tutorial.

This tutorial is the first part: Getting Started Guide.

Go to http://www.linb.net for the latest information.

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

Chapter 1. Preparation

First of all, note that all instances of this tutorial are based on version 3.0. Therefore, our first task is to download the 3.0 release package, and to establish the local environment.

1.1. Download the package

3.0 Release package can be downloaded from http://groups.google.com/group/linb/files. It's the latest stable 3.0 version, but not the latest code. I suggest you get the latest 3.0 code from SNV. For those who are not familiar with SNV, should learn how to use SNV first. After all, a lot of open-source projects use SNV to manage code. SNV requires a client program to connect to what is called a "repository" where the files are stored. On commonly used SNV client is called TortoiseSNV, which is freely available. Other clients exist, but TortoiseSNV is recommended due to its simplicity of use.

3.0 version "repository" URL: http://linb.googlecode.com/svn/trunk/jsLinb3.0/ .

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.
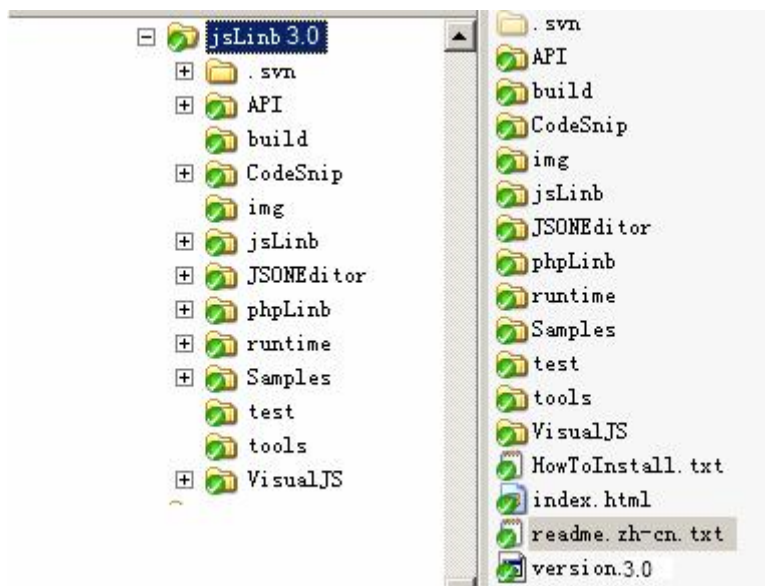
The contents of the package folder

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). At last, 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/jsLinb/), you should be able to open the page with your browser: http://localhost/jsLinb/.

You can browse http://localhost/jsLinb/Examples/ for examples, and http://localhost/jsLinb/API/ for API.
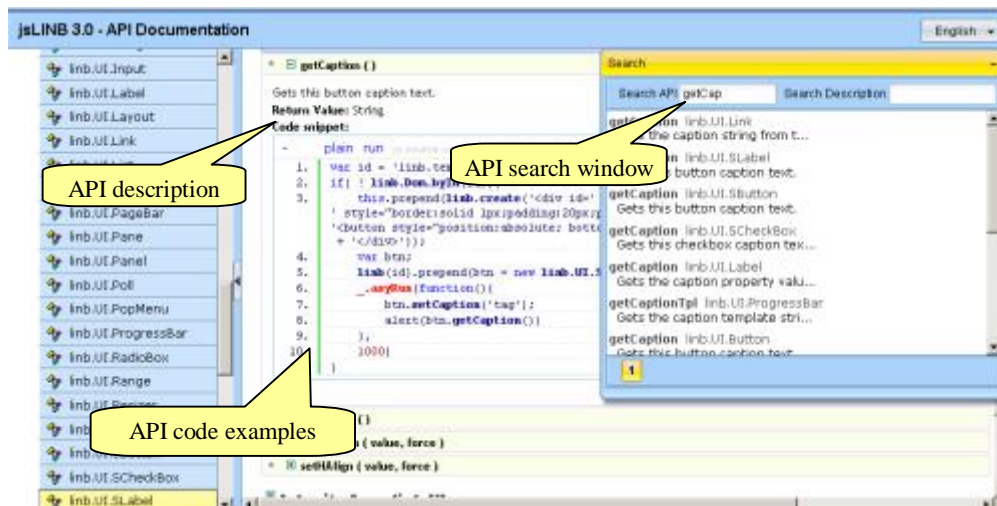
A simple glance at API is strongly recommended. Learn about how to search a specific API, and how to run the inner code examples.

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/jsLinb/), add a sub folder "chapter1" in it, and create a file named "helloworld.html" in "chapter1". Enter the following code:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<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/jsLinb/js/linb-all.js"></script>
    <title>jsLinb Case</title>
</head>
    <body>
        <script type="text/javascript">
            linb.main(function(){
                linb.alert("Hi", "Hello World");
            });
        </script>
</body>
</html>
```

strict mode is recommended

Include lib file

main function

cases/chapter1/helloworld.html

There is a cases folder in the package folder. You can find the source code for each example in this tutorial.

You can double-click the helloworld.htm1 to open the file.
Or open URL http://localhost/jsLinb/cases/chapter1/helloworld.html in your browser (firefox or chrome is recommended here). And you can see the following result:
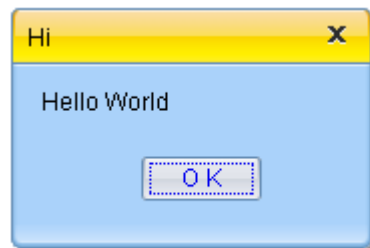
You may have noticed, no lib CSS file was included in the html file. Yes, jsLinb generate CSS automatically, no lib CSS.
File "linb-all.js" contains all codes except several advanced controls (Button, Input, CombInput,

Tabs, TreeBar, and TreeGrid etc.). This file can be found in "runtime/js" folder.
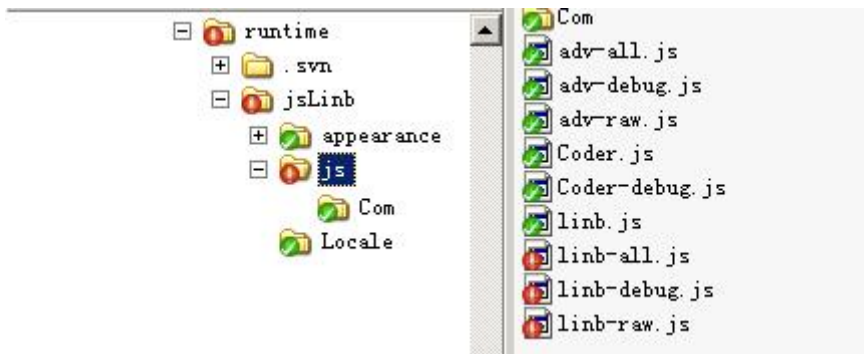
2.2. Render onto a html node

"Replace an html node (such as a div) with an advanced control." 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 PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<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/jsLinb/js/linb-debug.js"></script>
    <title>jsLinb Case</title>
</head>
    <body>
        <div id="grid" style="position:absolute;left:100px;top:100px;width:300px;height:200px;"></div>
                                                              The DIV with id "grid"
        <script type="text/javascript">
            linb.main(function(){
                var grid = new linb.UI.TreeGrid();
                grid.setGridHandlerCaption('grid')          Sets grid caption
                    .setRowNumbered(true)
                    .setHeader(['col 1','col 2','col 3'])    Show line number
                    .setRows([
                        ['a1','a2','a3'],                    Sets columns
                        ['b1','b2','b3'],
                        ['c1','c2','c3'],
                        ['d1','d2','d3'],                    Sets rows
                        ['e1','e2','e3'],
                        ['f1','f2','f3']
                    ]);
                grid.renderOnto('grid');
            });
        </script>
                     Render onto that DIV
</body>
</html>
```

cases/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 :

```
linb.main(function(){
    (new linb.UI.TreeGrid({          Gives properties directly
        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');
});
```

cases/chapter1/renderonto2.html

renderonto3.html :

```
linb.main(function(){
    linb.create('TreeGrid',{
        gridHandlerCaption:'grid',
        rowNumbered:true,           Using linb.create
        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');
});
```

cases/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 Designer

There are two types of designer in jsLinb: One is a simplified version, the other one is an advanced version integrated with document management features. The purposes of the two designers are to reduce development time on UI layout. We only use the simplified version in this introduction

tutorial.

Go to http://localhost/jsLinb/VisualJS/UIBuilder.html for the simplified one.
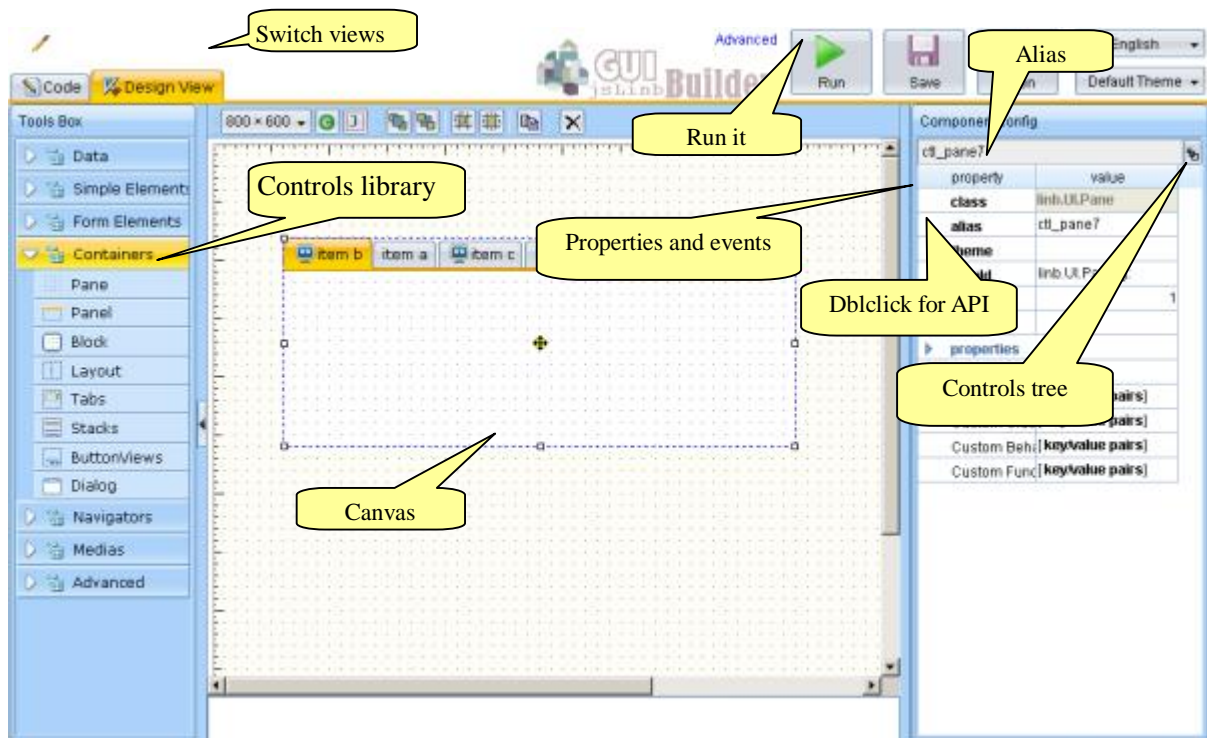
Now, we are trying to create the previous section's grid example in Designer.

1.   Open the navigators group in "Tools Box", and drag the "TreeGrid" control to the Canvas area.
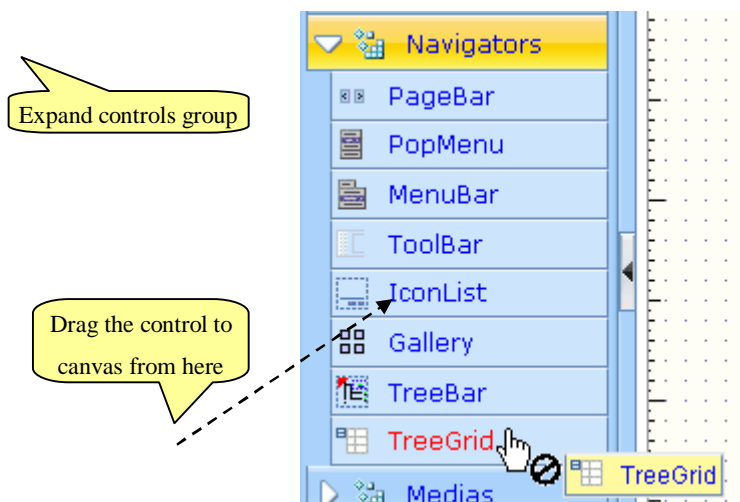
2.   Click to select the "treegrid"
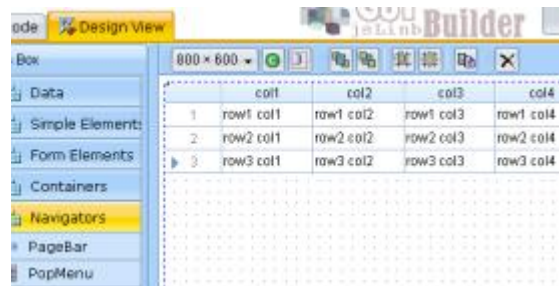
3. Sets this grid's properties according to the following picture.

- Sets dock to 'none';
- Sets rowNumbered to false;
- Sets gridHandlerCapion to 'gird'.

4. Sets header and rows

Sets header data

[] treegrid4 => rows			×

```
1   [
2       ['a1','a2','a3'],
3       ['b1','b2','b3'],
4       ['c1','c2','c3'],
5       ['d1','d2','d3'],
6       ['e1','e2','e3'],
7       ['f1','f2','f3']
8   ]
9
10
11
12
```

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	

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

Code | Design View

Class Outline | Syntax Check

```
1     // The default code is a com class (inherited from linb.Com)
2     Class('{className}', 'linb.Com',{
3         // Ensure that all the value of "key/value pair" does not refer to external variables
4         Instance:{
5             // To initialize instance(e.g. properties)
6             initialize : function(){
7                 // To determine whether or not the com will be destroyed, when the first UI control be destroyed
8                 this.autoDestroy = true;
9                 // To initialize properties
10                this.properties = {};
11            },
12            // To initialize internal components (mostly UI c_____
13            // *** If you're not a skilled, dont modif_____ function manually ***
14            iniComponents : function(){
15                // [[code created by jsLinb UI Builder
16                var host=this, children=[], append=function(child){children.push(child.get(0))};
17
18                append(
19                    (new linb.UI.TreeGrid)
20                    .setHost(host,"ctl_treegrid2")
21                    .setDock("none")
22                    .setLeft(0)
23                    .setTop(0)
24                    .setGridHandlerCaption("grid")
25                    .setHeader([{"id":"col1", "width":80, "type":"label", "caption":"col1"}, {"id":"col2", "width'
26                    .setRows([{"cells":[{"value":"row1 col1", "id":"c_a"}, {"value":"row1 col2", "id":"c_b"}, {"v
27                );
28
```

Code created by jsLinb UI Builder

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

7. Click "Run" 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 PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<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 LINB framework</title>
</head>
    <body>
        <div id='loading'><img src="../runtime/loading.gif" alt="Loading..." /></div>
        <script type="text/javascript" src="../../runtime/jsLinb/js/linb-all.js"></script>
        <script type="text/javascript">
            Class('App', 'linb.Com',{
                Instance:{
                    iniComponents:function(){
                        // [[code created by jsLi
                        var host=this, children=[], append=function(child){children.push(child.get(0))};

                        append((new linb.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 jsLinb UI Builder
                    }
                }
            });
            linb.Com.load('App', function(){
                linb('loading').remove();
            });
        </script>
</body>
</html>
```

Showing a loading picture

Places js lib files in body

Class created in Designer.
You can save this part of code to
App/js/index.js

Load UI in asynchronous mode
If no App Class in memory, lib will load the
Class from **App/js/index.js** file first.

cases/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 à designer.grid.standard.html, and

App/js/index.js.

designer.grid.standard.html is:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<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 LINB framework</title>
</head>
    <body>
        <div id='loading'><img src="../runtime/loading.gif" alt="Loading..." /></div>
        <script type="text/javascript" src="../../runtime/jsLinb/js/linb-all.js"></script>
        <script type="text/javascript">
            linb.Com.load('App', function(){
                linb('loading').remove();
            });
        </script>
</body>
</html>
```
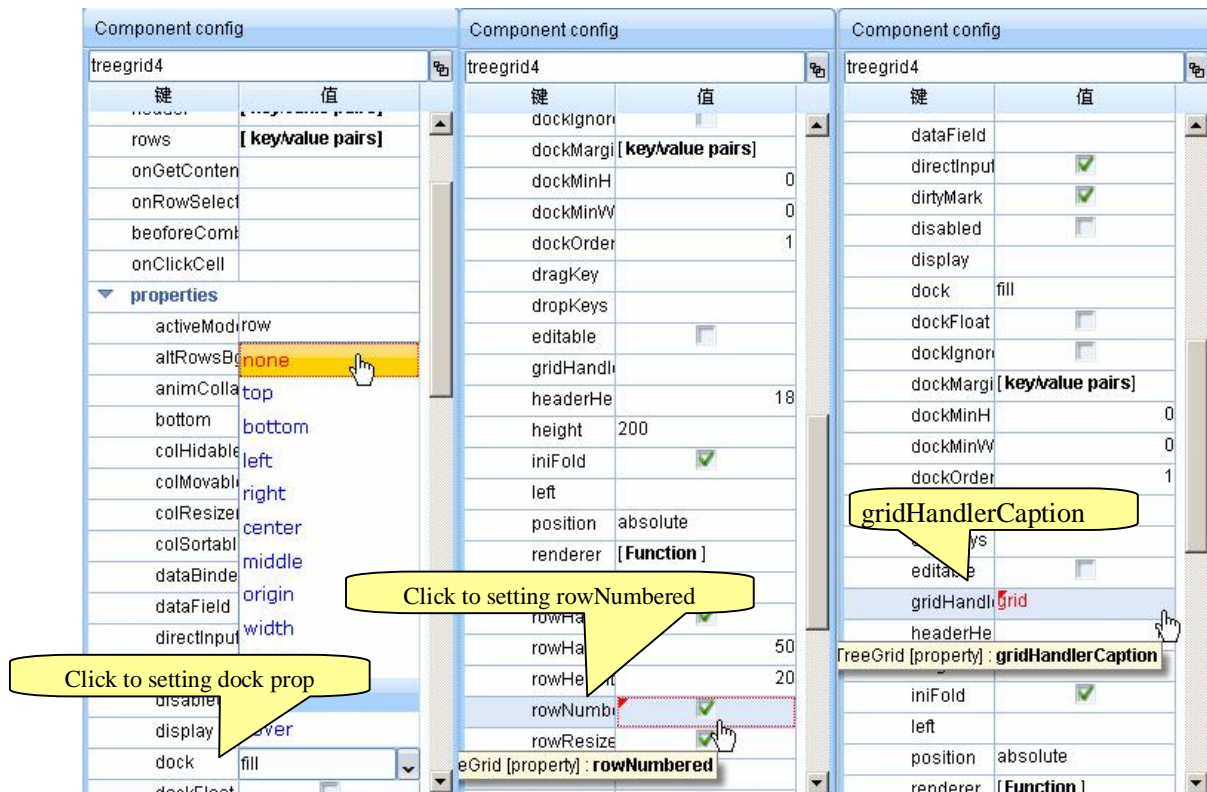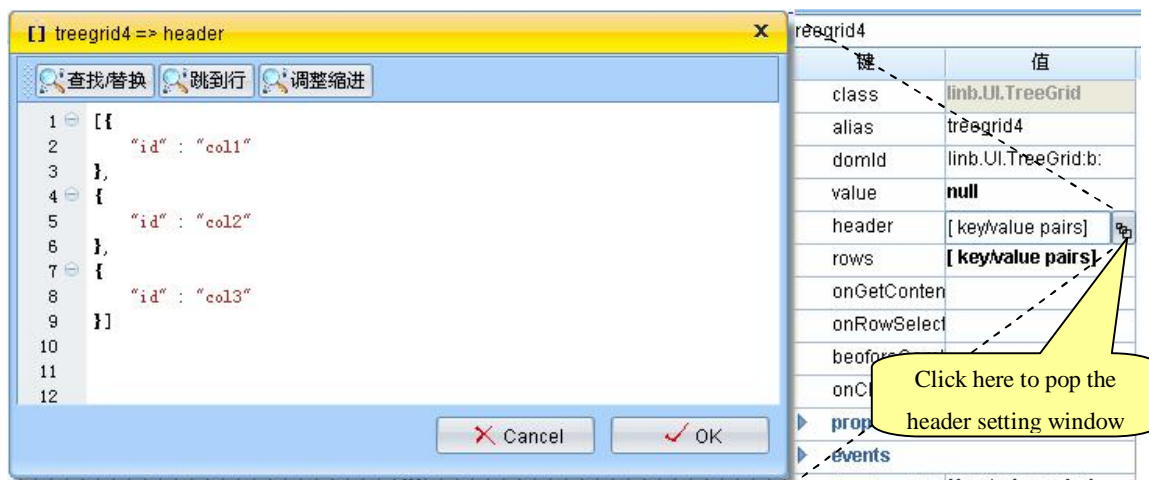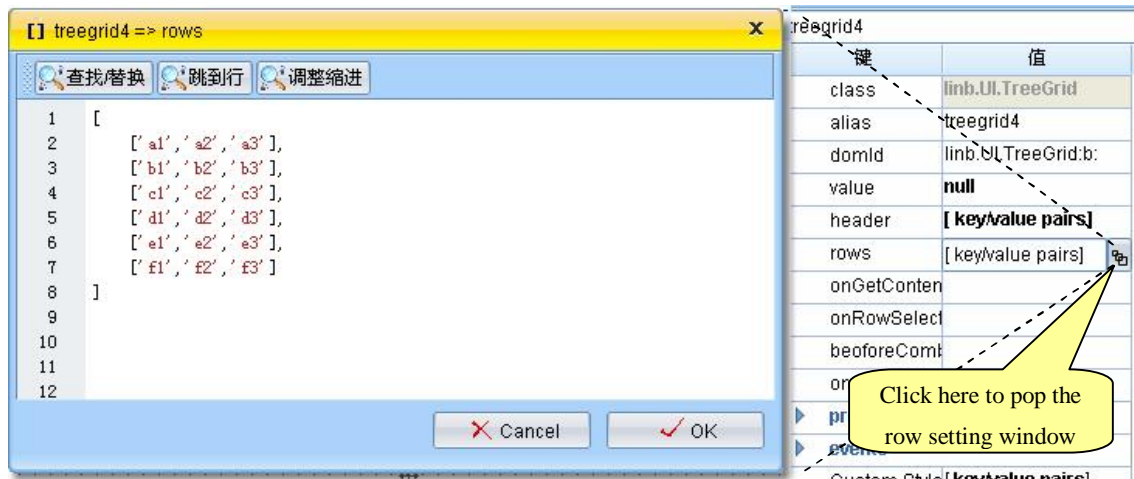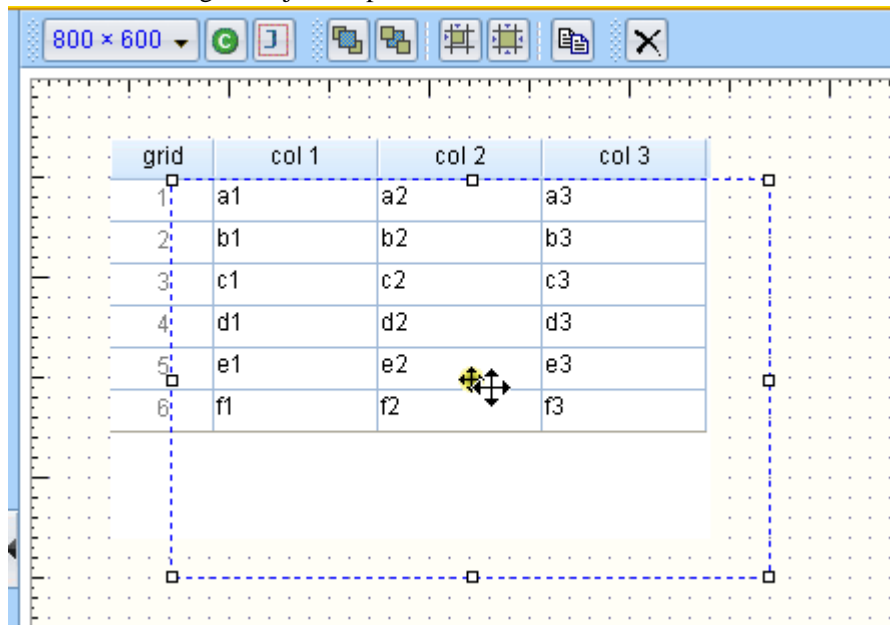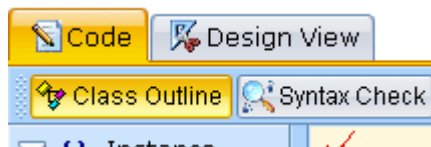
Load App class from App/js/index.js asynchronously

At last, remove loading picture

cases/chapter1/designer.grid.standard.html

App/js/index.js is:

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

cases/chapter1/App/js/index.js

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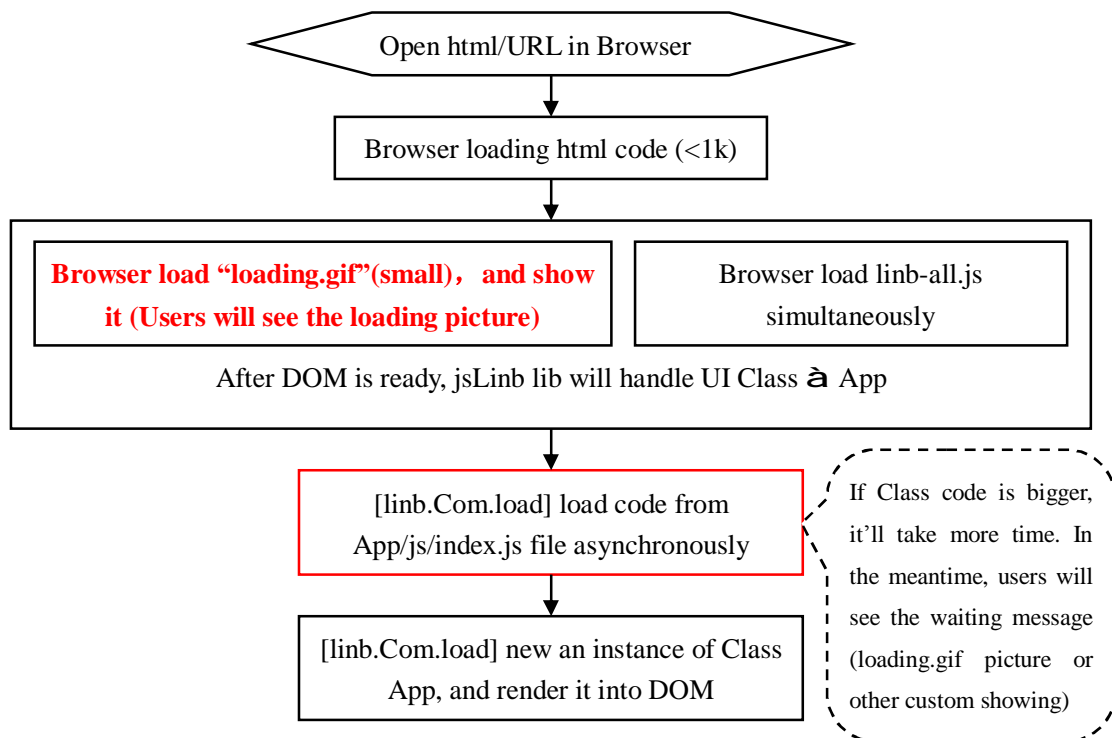
2.5. Code Editor

By the way, in order to get better performance, firefox and chrome are recommended here.

There are two views in Builder: "Design view" and "Code" view. The default view is "Design view". Click "Code" tab to switch to "Code" view.
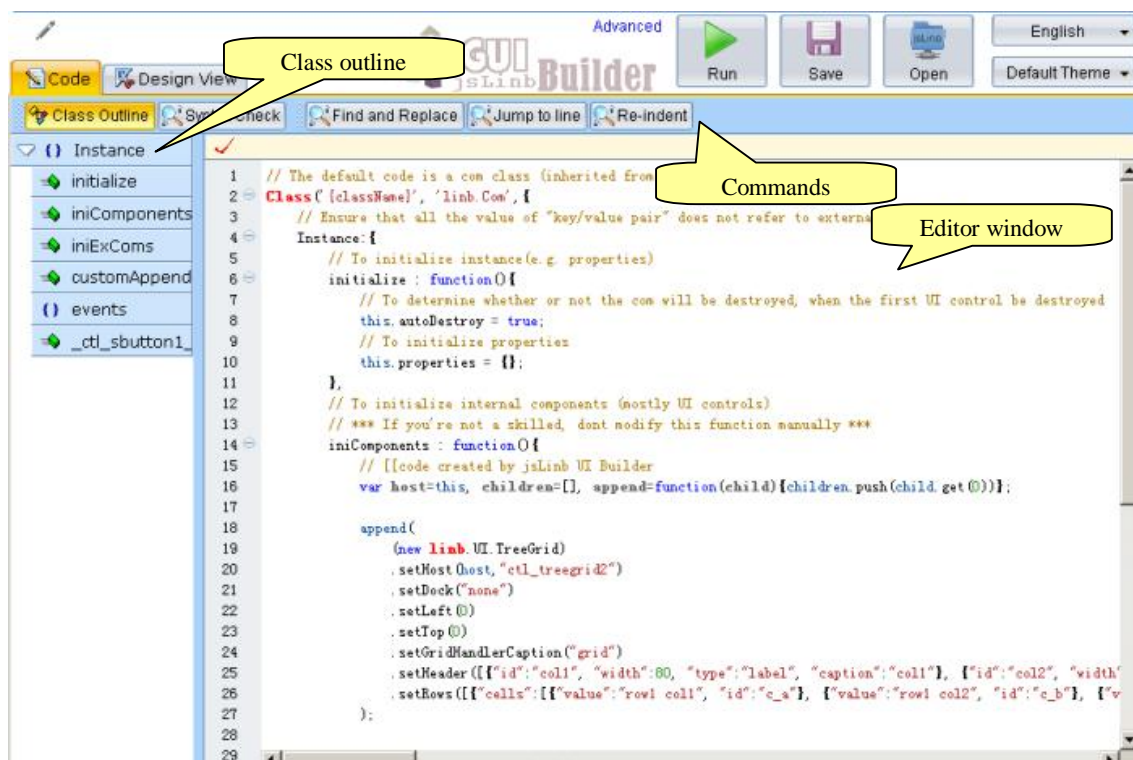
2.5.1. Highlight code from Outline window

"Class Outline" is located in the left side of "Code" view. By clicking any member or method name in "Class Outline", Builder will highlight its code in "Editor window", and scroll "Editor window" to show the code.
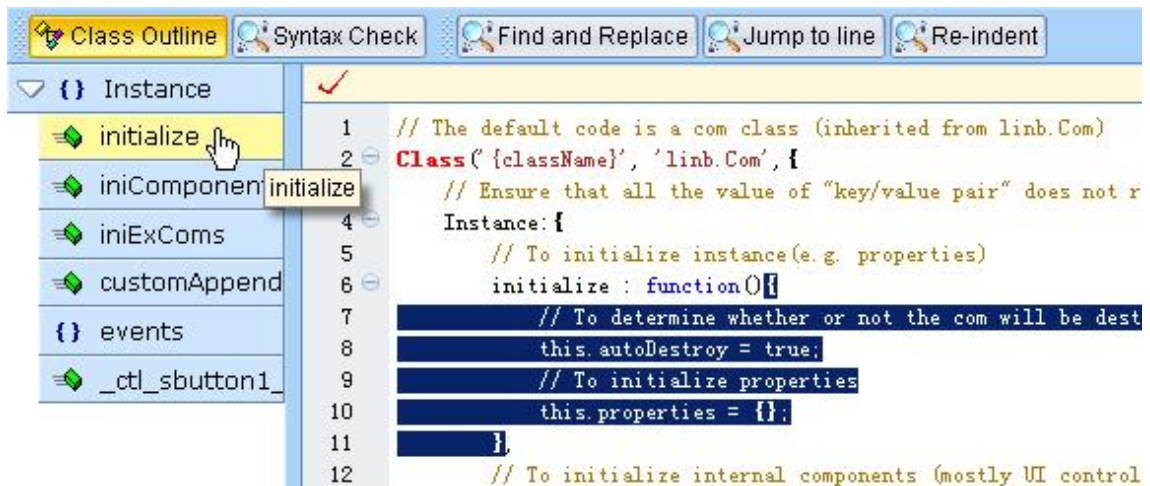
2.5.2. Code Folding

To make your code view more clear to read and understand, jsLinb Builder lets you fold certain parts of it. Click the left side "plus" or "minus" will fold or expand the block code.
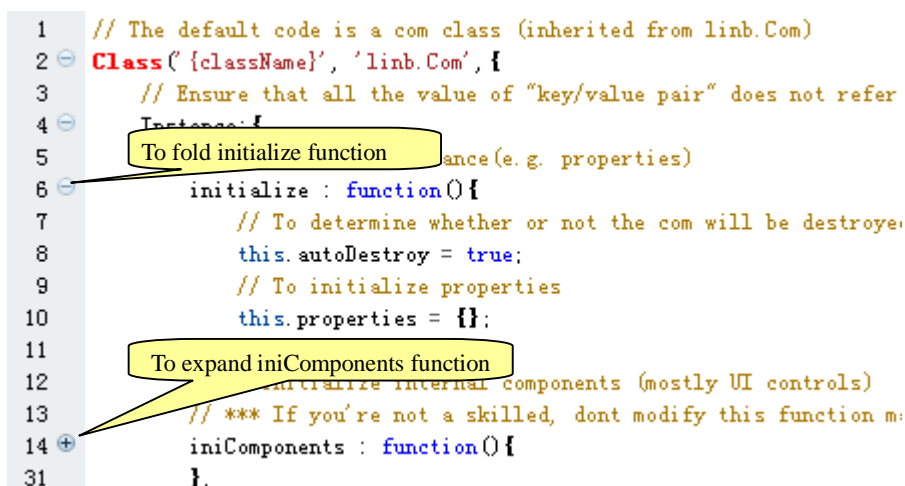
Note: Due to the browser's poor performance on iframe, please try not to frequent collapse or expand the large body function or object. Chrome's performance is better than others.

2.5.3. Code Intellisense

Three types Code Intellisense are supported.

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

ι When use shortcut [Alt+1], or dbclick a recognizable variable
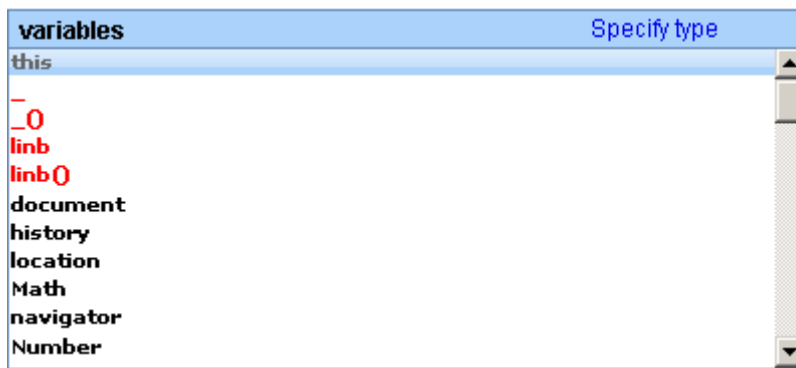
Keyboard actions for Code Intellisense pop Window:

ι "up": Focus to next item in code list
ι "down": Focus to previous item in code list
ι "enter": Select the current focused item, and input to editor window
ι "esc": Close the pop window
ι 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.
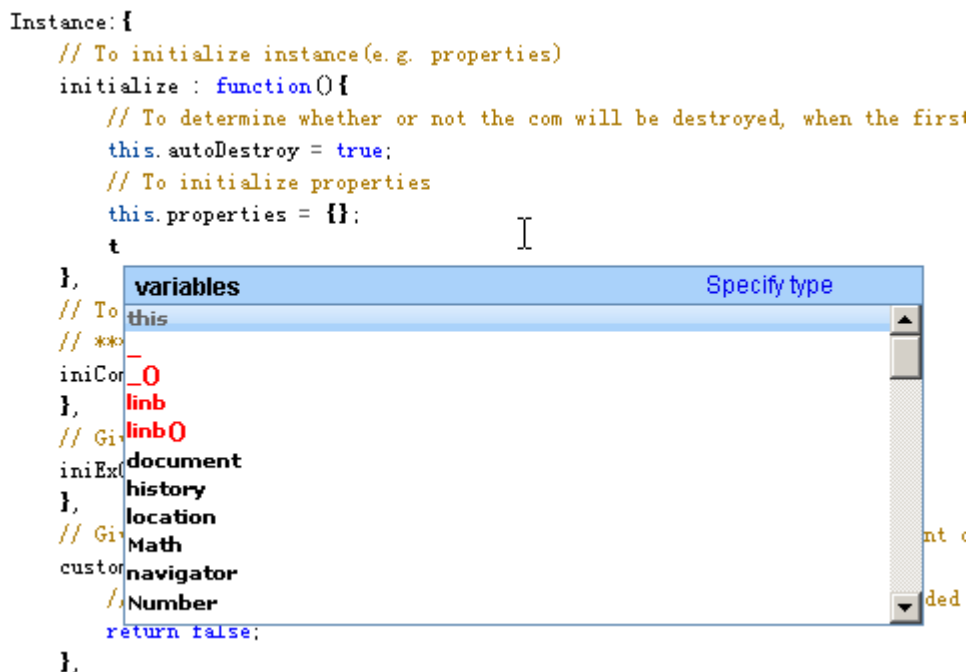
If the input string is "fo", the "for loop statement" will be the default focused item.

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.

Chainable methods can show Code Intellisense window too.

Chainable methods
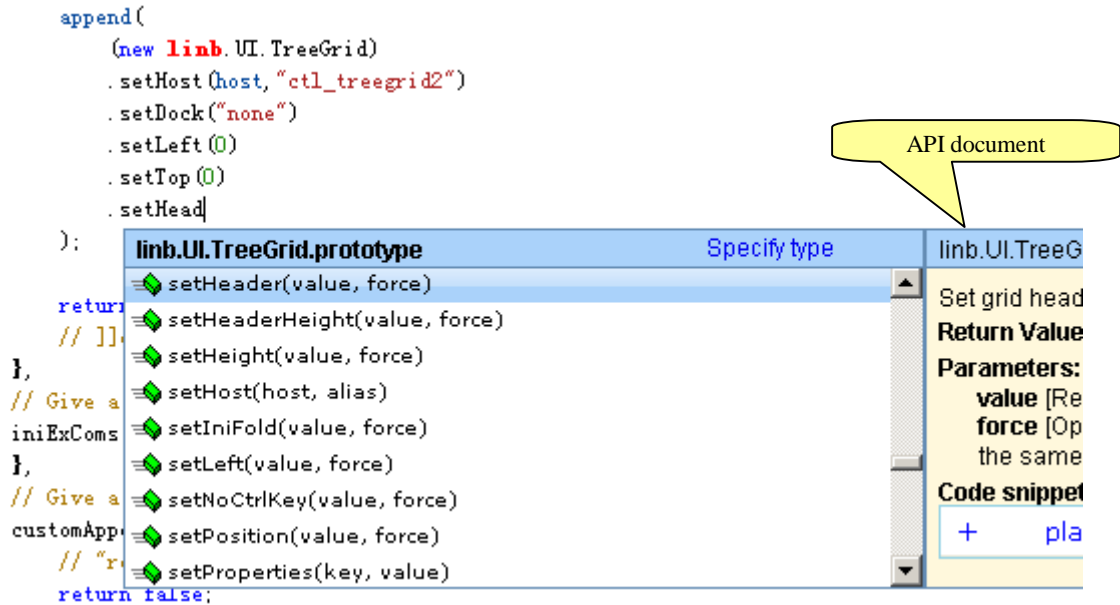
2.5.3.3. When use shortcut [Alt+1], or dbclick

When a variable was focused, press shortcut [Alt+1] will trigger editor to pop the Code Intellisense window. Double click this variable string will get the same result.
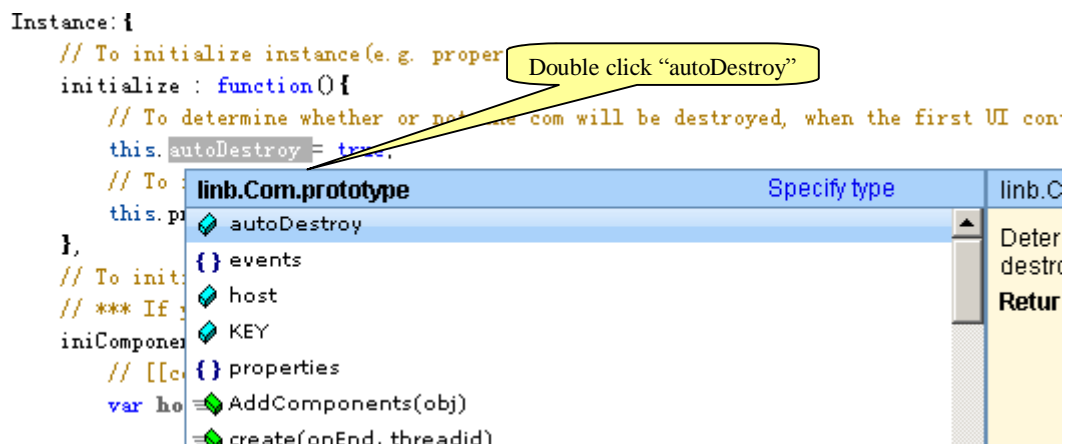
2.5.4. Find the object definition code

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.
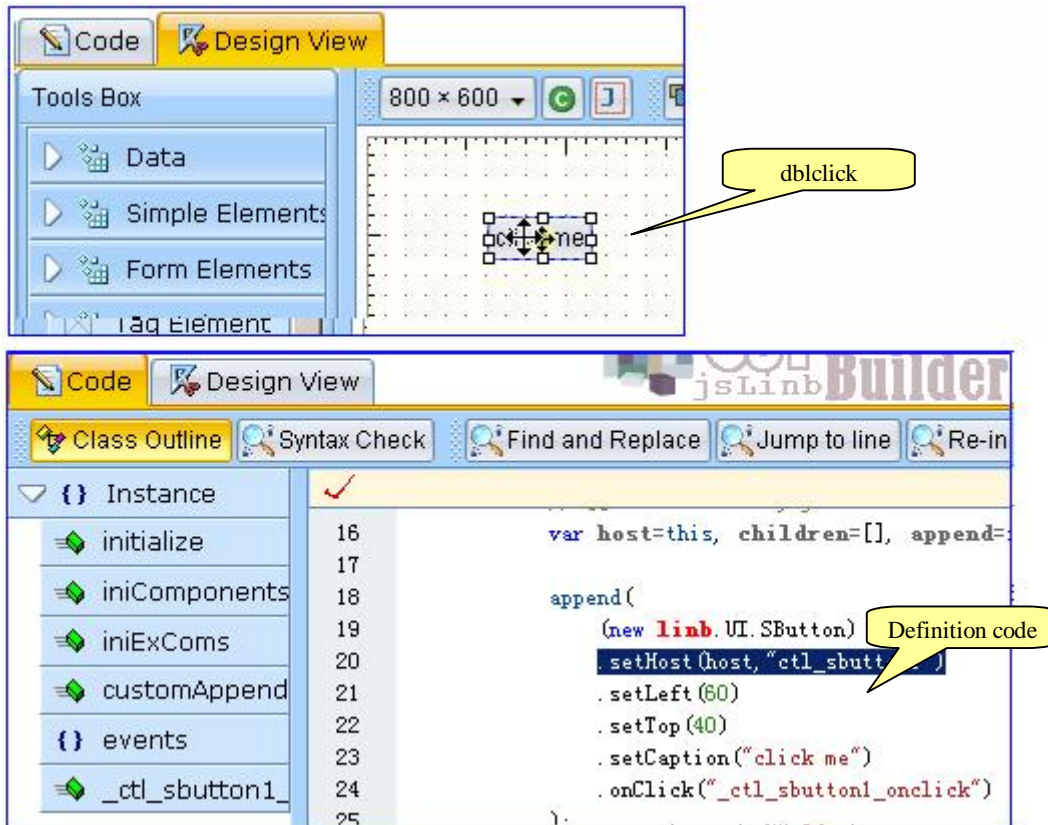
2.5.5. Generate event code automatically

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.

Event code created
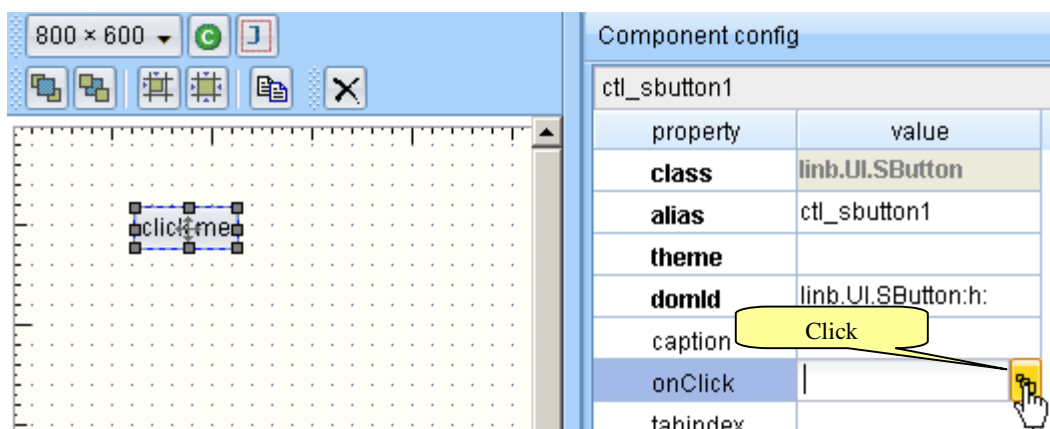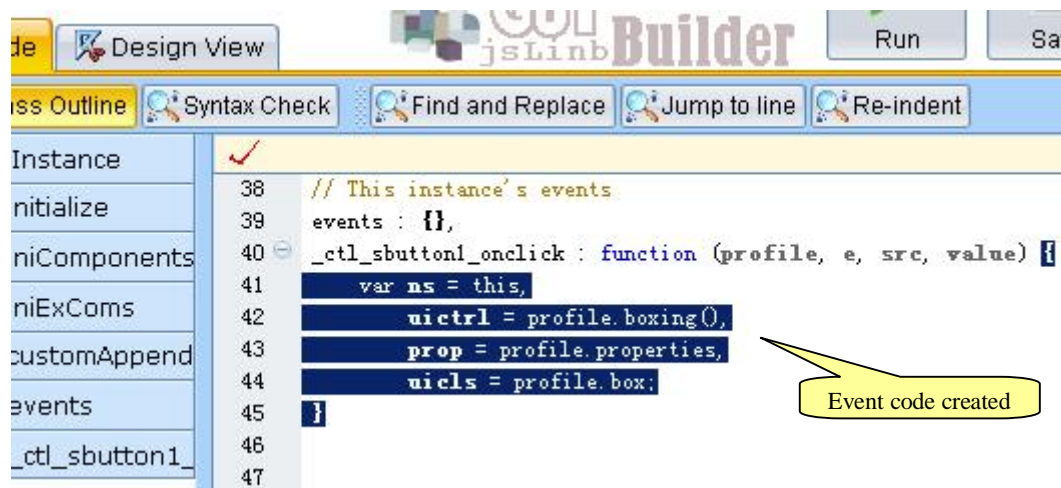
Chapter 3. Controls Facebook

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 it does not like firefox, ie8 or chrome is ok too.

For firefox:
1. You need firefox and firebug;
2. Open URL cases/**env.html** in firefox;
3. Open firebug console, switch to the multi-line mode

Enter the test code here

Then, click "run"
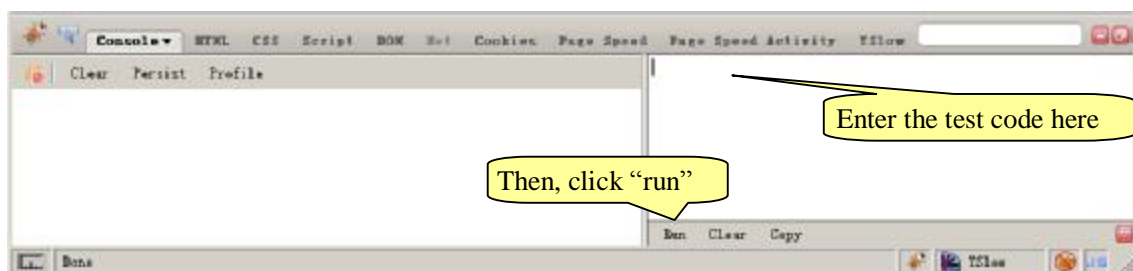
For IE8:
1. You need IE8;
2. Open URL cases/**env.html** ;
3. Open developer tools, switch to the multi-line mode
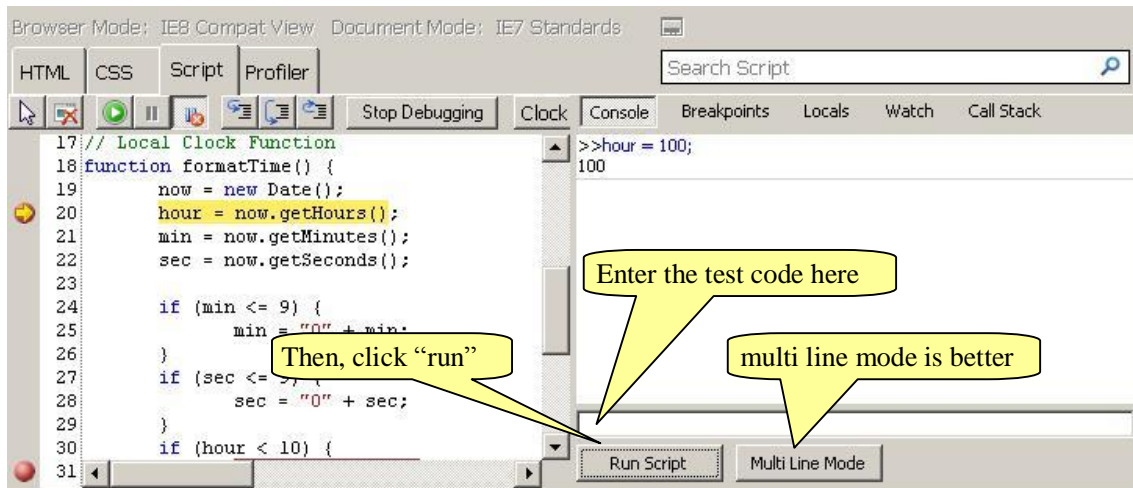
For IE8:

4. You need the latest Chrome;
5. Open URL cases/**env.html** ;
6. Open developer tools
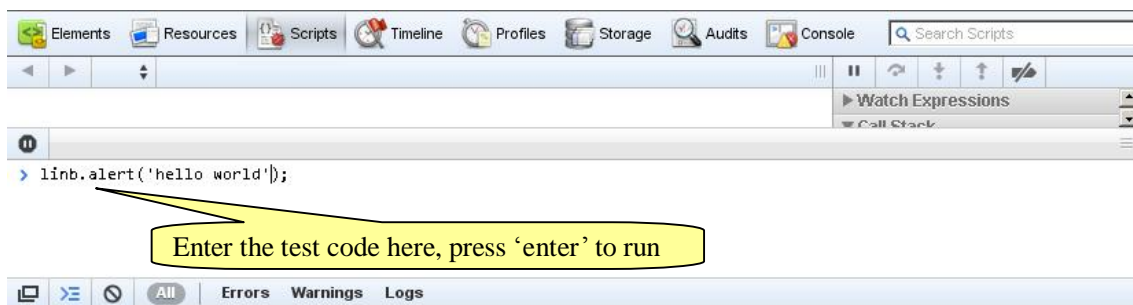
There's a "Clear" button in cases/**env.html**, You can click this button to clean up the current page's DOM. In some cases, you want to clean up both DOM and memory, press 'F5' to refresh your browser.
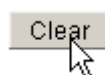
3.2. "Hello world" in env.html

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

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

Output:
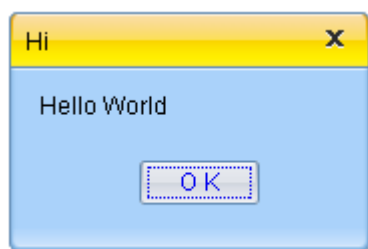
Click "Clear" button to clean the DOM.

3.3. Control creation and runtime update

There are three approaches to create jsLinb control.

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

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

// Approach 3
(new linb.UI.SButton())
.setCaption("Using new and get/set")        We use new/setXX mode in Desinger
.setPosition("relative")
.show();
```

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=linb.create("Dialog", {caption: "runtime "}).show();        Create a Dialog
_.asyRun(function(){
    dlg.setCaption("updated");        To modify caption
},500);
_.asyRun(function(){
    dlg.setMaxBtn(false);        To hide the max button
},1000);
_.asyRun(function(){
    dlg.setStatus("max");        To modify status
},1500);
_.asyRun(function(){
    dlg.destroy();
},2000);        To destroy it
```

3.4. Button related

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

3.4.1. onClick event

Input:

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

Output:

Click Me   Click it

Hi   ✕
You are great!
OK

Input:

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

_.asyRun(function(){
btn.setHeight(80)
    .setShadow(true)
    .setType("drop")
},1000);                                 Execute code after 1 second
```

Output:

Click Me
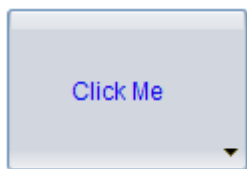
NOTE

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

3.4.2. Boolean Controls

There are three controls can represent and modify Boolean value:

Input:

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

Button ☑ SCheckBox ☑ CheckBox
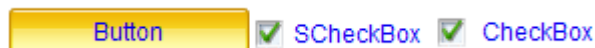
3.4.3. Link Control

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

Input:

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

// href property
linb.create("Link",{href: "http://www.linb.net"}).show(null,null,100,100)

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

Sets caption

Adds onClick event

Output:

Click Me

Goes to linb.net

No action

link2   link3

Shows alert window
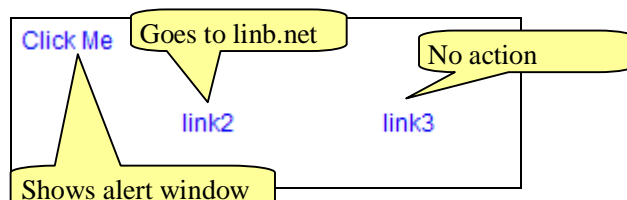
3.5. Label related

This section relates to the following controls: linb.UI.SLabel, linb.UI.Label and linb.UI.Div. These three controls can be used as "label", linb.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 linb.UI.Label; Or if you want to input more complex html code in the control, linb.UI.Div is better.

Input:

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

Output:
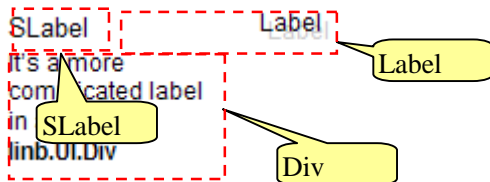
3.6. Input related

This section relates to the following controls: linb.UI.Input, linb.UI.ComboInput and linb.UI.RichEditor. linb.UI.ComboInput is an enhanced version of linb.UI.Input, it can input/edit value through a pop window; linb.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 linb.absValue control) in jsLinb 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 inputs "**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 linb.UI.Input()).show();          A new Input
    linb.message(input.getUIValue()+":"+input.getValue());
_.asyRun(function(){
    input.setUIValue('uivalue');          Sets UI value
    linb.message(input.getUIValue()+":"+input.getValue());
},2000);
_.asyRun(function(){          Updates  UI  value  to  control
    input.updateValue();
    linb.message(input.getUIValue()+":"+input.getValue());
},4000);
```

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".

Dirty Mark   uivalue   .

```
var input = (new linb.UI.Input()).show();
_.asyRun(function(){          Dirty Mark appears
    input.setUIValue('uivalue');
},1000);
_.asyRun(function(){          Dirty Mark disappears
    input.updateValue();
    input.setDirtyMark (false);          If DirtyMark is disabled
},2000);
_.asyRun(function(){
    input.setUIValue('uivalue 2');          Nothing happen
},3000);
```

3.6.3. Password Input

Sets Input's type property to "password".

Input:

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

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

Output:

3.6.4. Multi-lines

Sets Input's multiLine property to true.

Input:

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

_.asyRun(function(){                    Sets multiLine to true
    input.setUIValue('123456 \n abc').updateValue();
    linb.pop(input.getValue());
},1000);
```

Output:
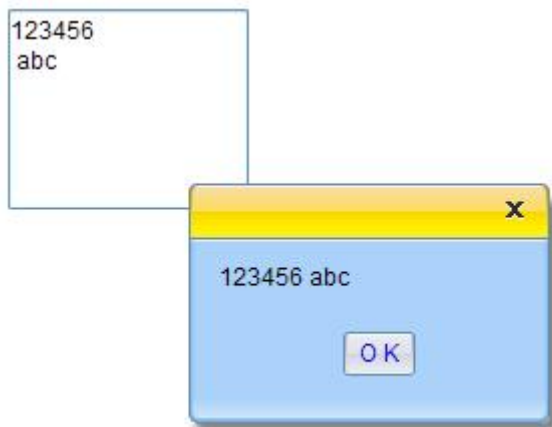
3.6.5. Input validation

3.6.5.1.   valueFormat property

"valueForamt" property represents a regular expression.

Input:

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

Executes the above code, inputs some charts, and let it lose the mouse focus, the "Error Mark" will appear.

not a number
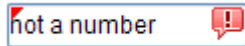
3.6.5.2. beforeFormatCheck event

Input:

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

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 linb.UI.Input())
    .setDynCheck(true)         Sets dynCheck
    .setValueFormat("^-?\\d\\d*$")
    .show();
```

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 linb.UI.Input control:

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

Input:

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

Output:
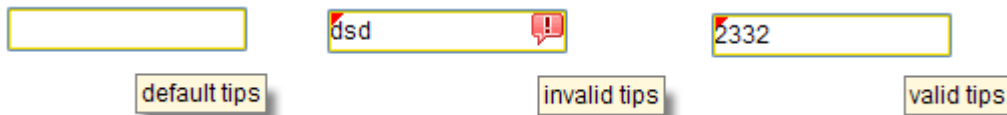
3.6.7.3. Binding Validation

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

Input:

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

input.show();                    Show SLabel here
slbl.show();
```

Output:

3.6.7.4. Custom Error Mark

We can custom "Error Mark" in beforeFormatMark event.

Input:

```
var input = (new linb.UI.Input())
    .setValueFormat("^-?\\d\\d*$")
    .beforeFormatMark(function(profile,err){
        if(err)
            linb.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:

asd

Invalid input!   X

Only number allowed!

O K

3.6.8. Mask Input

Mask Input examples:

| 11/11/1111 | __/__/____ | (111) 111-1111 | (__) ___-____ |
| ~1.11 | __.__ | (111) a-a *$* | (__) _-_ _$_ |

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

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

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

Input:

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

Output:

(____)_____-___   (8010)88888888-604

> **NOTE**
> **chapter2\Input\index.html** is an overall example for Input.

3.6.9. linb.UI.ComboInput

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

type   Keyboard input

a
b   readonly
c
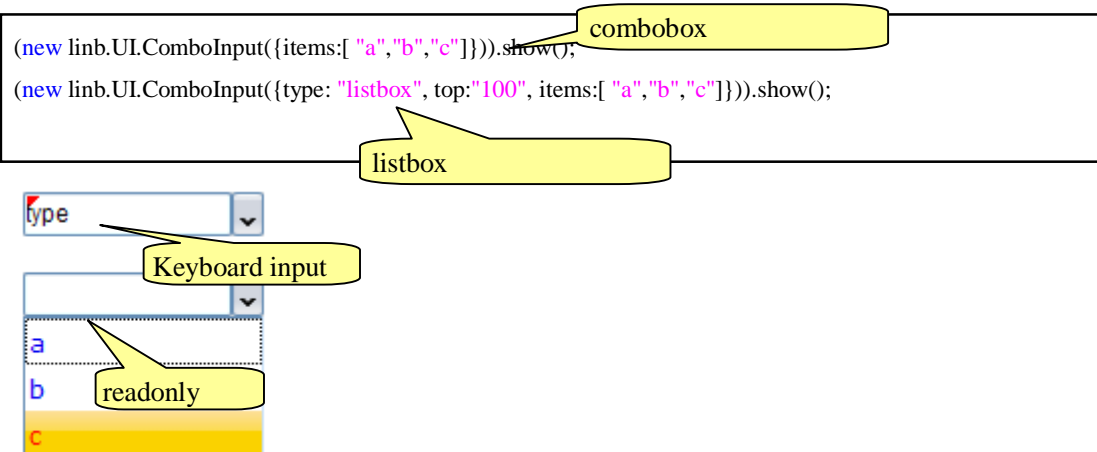
3.6.9.2. combobox, listbox and helpinput

There's an items property in linb.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']"). The lib? 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"
  }
];
linb.create('ComboInput',{position:'relative',items:items}).show();
linb.create('ComboInput',{position:'relative',items:items,type:'listbox'}).show();
linb.create('ComboInput',{position:'relative',items:items,type:'helpinput'}).show();
```

Output:

3.6.9.3. Date Piker

Sets type property to "date".

Input:

```
var ctrl=linb.create('ComboInput')
.setType('date')
.setValue(new Date)          Date  object  or
.show();                     timestamp string

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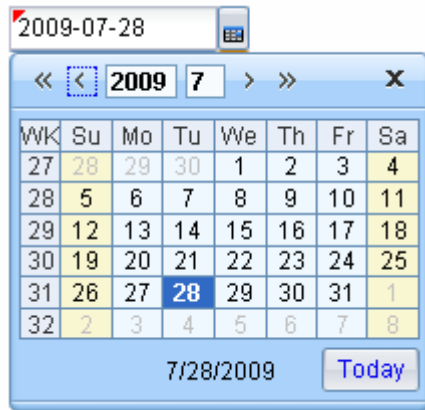
Output:

3.6.9.4. Time Picker

Sets type property to "time".

Input:

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

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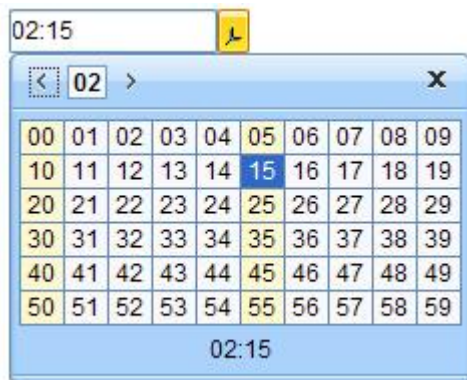
Output:

3.6.9.5. Color Picker

Sets type property to "color".

Input:

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

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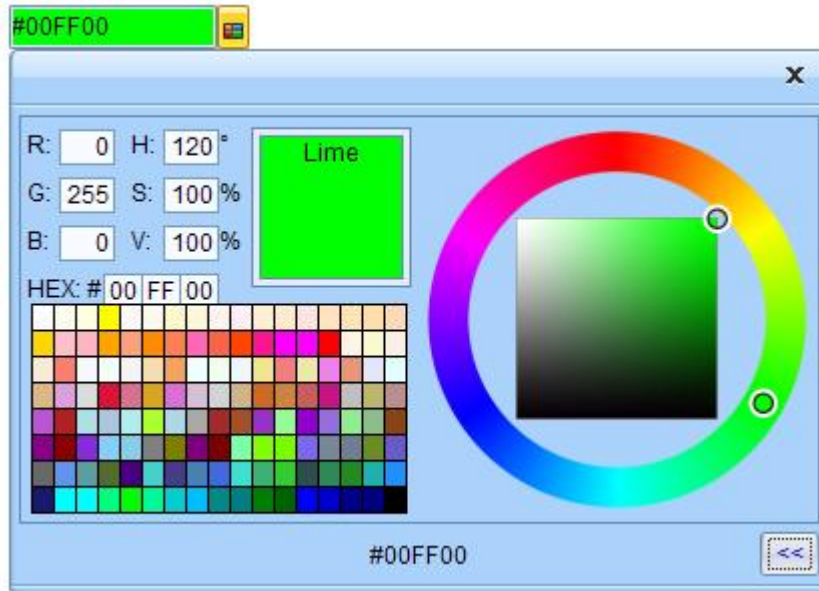
Output:

3.6.9.6. File Picker

Sets type property to "upload".

Input:

```
var ctrl=linb.create('ComboInput')
.setType('upload')
.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=linb.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=linb.create('ComboInput')
.setType('popbox')
.beforeComboPop(function(profile){
    var dlg=new linb.UI.Dialog, tb;
    dlg.append(tb=new linb.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:

dialog3

a

The custom pop window

b

▷ c
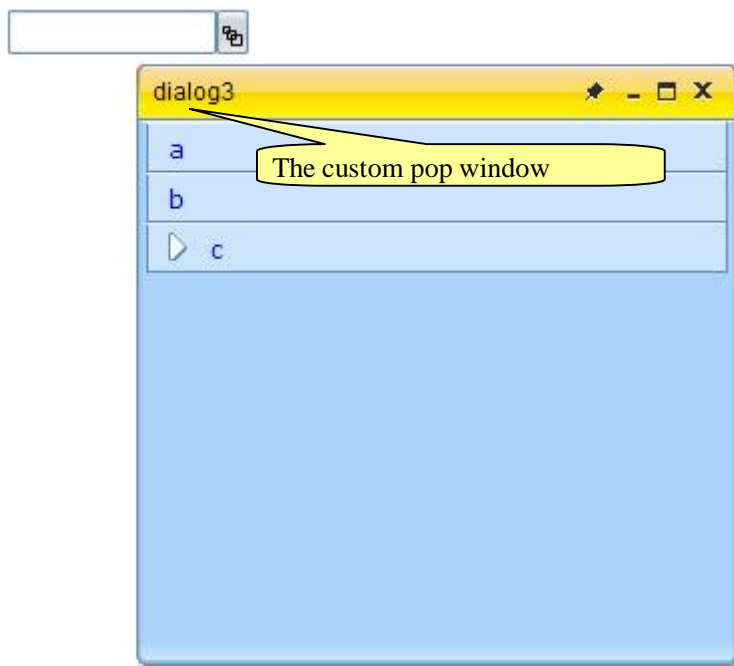
3.6.9.9.　　Command Buttons

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

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

Input:

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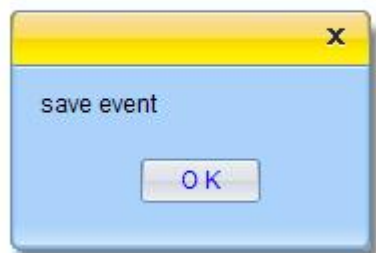
Output:

3.6.10.　　RichEditor

Input:

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

Output:
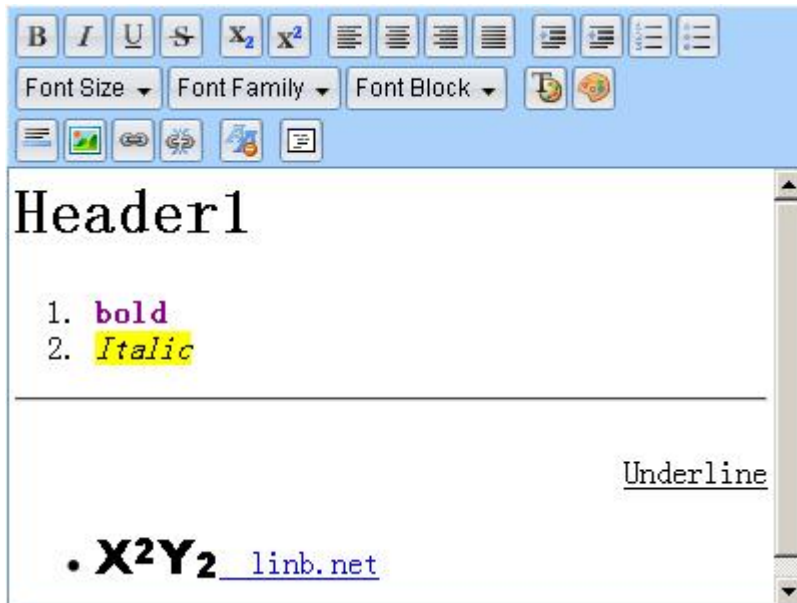
3.7. List related

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

3.7.1. A Simple one

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

onItemSelected event
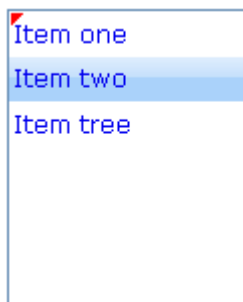
3.7.2. A little bit complicated

```
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>";
};

linb.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:

A Grid List

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

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


caption+linb.UI.SCheckBox

3.7.3. RadioBox

linb.UI.RadioBox is derived from linb.UI.List.

Input:

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

Output:

3.7.4. IconList and Gallery

Both are derived from linb.UI.List.

Input:

```
linb.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){
    linb.message(item.id);
})
.show();
```

Output:

Input:

```
linb.create("Gallery")                    Item's height
.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){
      linb.message(item.id);
})
.show();
```

Output:

User
It's a user

Computer
It's a computer

Tree
It's a tree
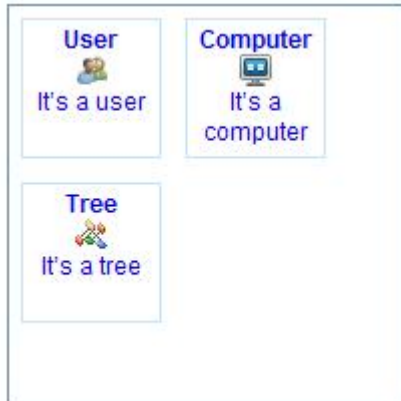
3.7.5. Item selection

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

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

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

3.7.6. Container related

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

linb.UI.Block.

linb.UI.Dialog, linb.UI.Layout and linb.UI.Tabs /Stacks/ButtonViews are container controls too, we will give an example? these controls in separate sections.

Container is those controls that can have child controls. In jsLinb Designer, you can drag a child control and drop it into a container control. Just like this,
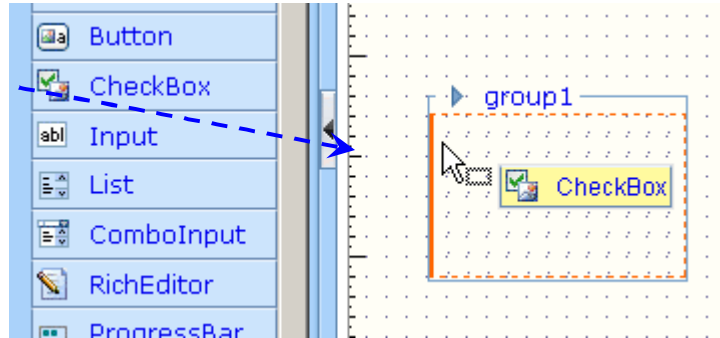
Input 1:

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

Input 2:

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

Input 3:

```
linb.create({                       In children object
    key:"linb.UI.Group",
    children:[[{key:"linb.UI.SButton"}]]
}).show();
```
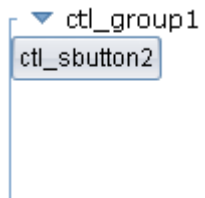
Output:

3.7.7. Pane and Panel

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

title bar.

Input:

```
(new linb.UI.Pane)
.append(new linb.UI.SButton)          You can't see output, It's transparent
.show()
```

Input:

```
(new linb.UI.Panel)        Sets dock to 'none'
.setDock("none")
.append(new linb.UI.SButton)
.show()
```

Output:

3.7.8. Block

Input:

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

Output:
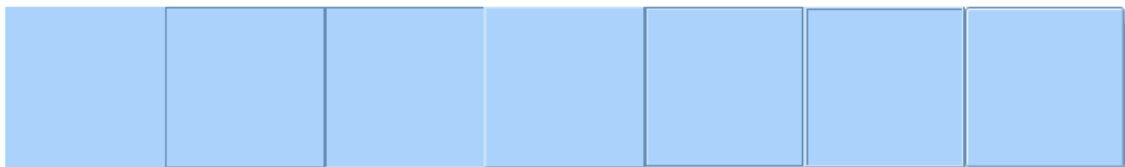
3.8. Dialog related

3.8.1. Normal state

Input:

```
(new linbUI.Dialog).show()
```

Output:
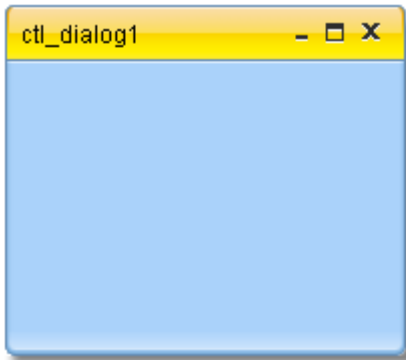
Input:

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

Output:
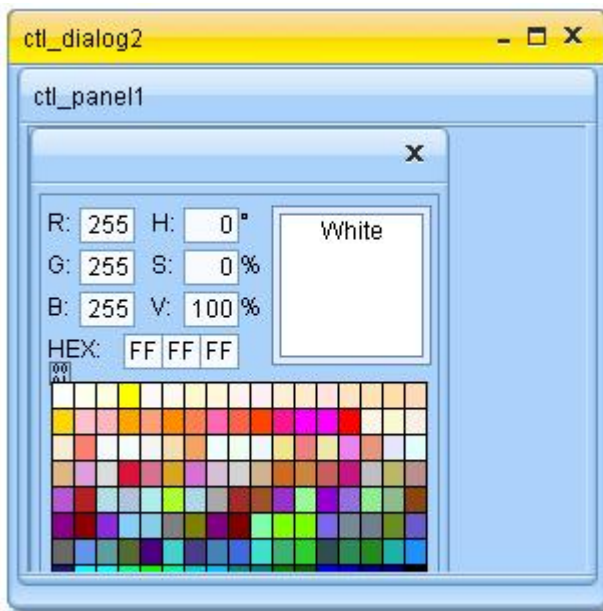
3.8.2. Min and Max status

Input:

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

Output:

ctl_dialog3

3.8.3. Modal Mode

Input:

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

(new linb.UI.Dialog)
.setHtml("The first modal dialog")
.show(null,true);                           Parent is html body
```

Output:
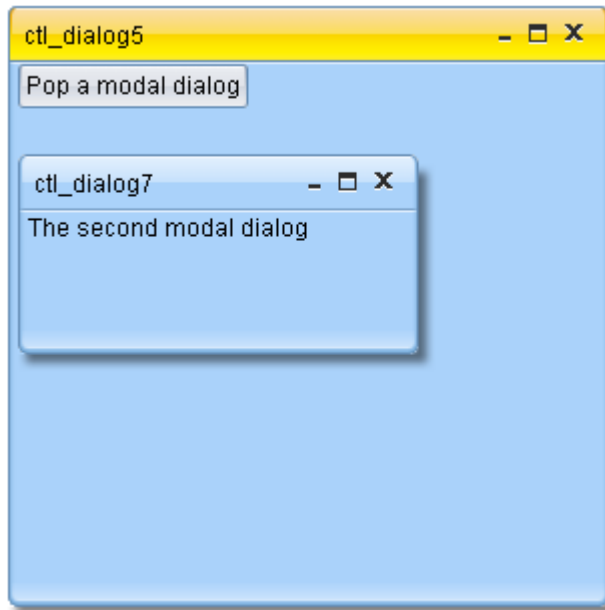
3.9. Layout Control

Input:

```
var block=linb.create("Block").setWidth(300).setHeight(300);
var layout=linb.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:

Before container

Fold/Expand command button

Main container

After container
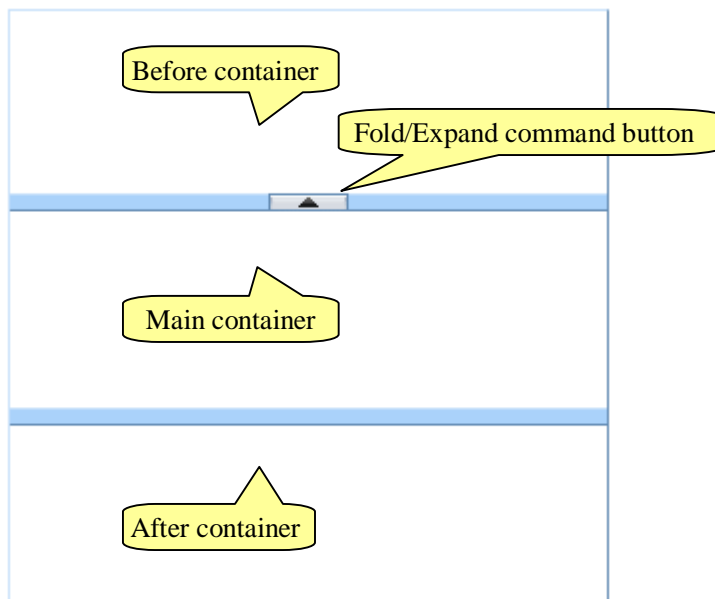
Input:

```
var block=linb.create("Block").setWidth(400).setHeight(100);
var layout=linb.create("Layout",{items:[
    {id:'before',
     pos:'before',
     size:100,            Default status is fold
     cmd:true,
     folded:true,         Max size is120
     max:120,
     min:80               Min size is 80
},{
     id:'after',
     pos:'after',
     cmd:true,
     locked:true,         Size locked
     size:50
},{
     id:'after2',
     pos:'after',
     size:50
}],type: 'horizontal'});    horizontal
block.append(layout).show();
```

Output:

With command button, without split bar

fold

Without command button, with split bar
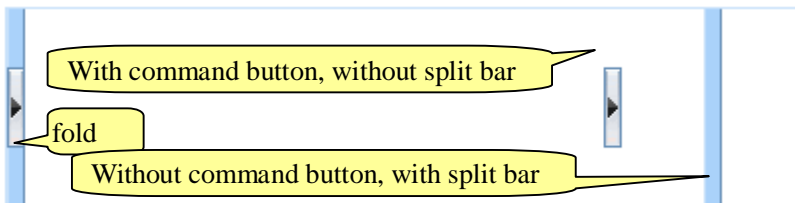
3.10. Multi-pages Controls

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

Input:

```
var block=linb.create("Block").setWidth(400).setHeight(100);
var pages=linb.create("Tabs",{
    items:["page1","page2","page3"],        3 pages        Append to a block
    value:"page2"
});                          The default page
block.append(pages).show();

                                            Adds a SButton to 2th page
_.asyRun(function(){
    pages.append(new linb.UI.SButton,"page2")
},1000);
```

Output:

| page1 | page2 | page3 |

ctl_sbutton1
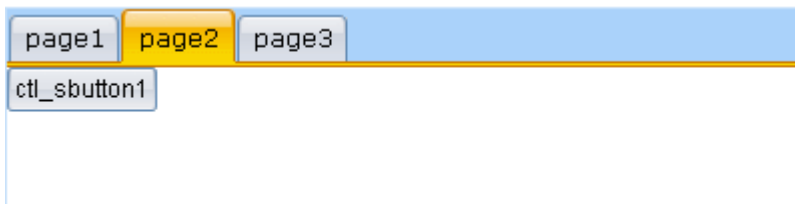
3.10.1. noPanel property

For linb.UI.Tabs and linb.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=linb.create("Block").setWidth(400).setHeight(300).show();
    var items=["page1","page2","page3"];
    linb.create("Tabs",{
        items:items,
        value:"page2",
        position:'relative',       Set position to 'relative'
        width:'auto',              Auto width
        height:'auto',             Auto height
        dock:'none',
        noPanel:true               No container
    }).show(block);

    linb.create("ButtonViews",{
        items:items,
        value:"page2",
        position:'relative',
        width:'auto',
        height:32,                 Set height to buttonview
        barSize:30,
        dock:'none',
        noPanel:true               No container
    }).show(block);
```

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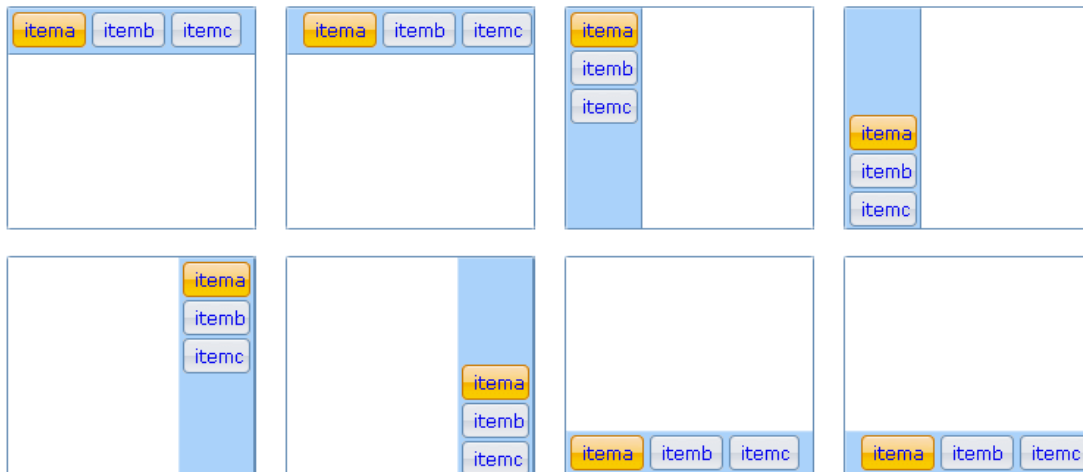
In **chapter2\ButtonViews\index.html**

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

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

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

Output:

page1	page2	page3
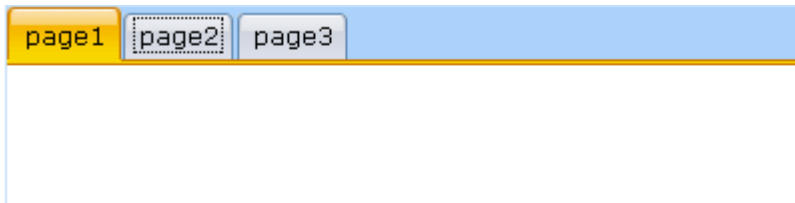
3.10.4.   Pages

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

Output:
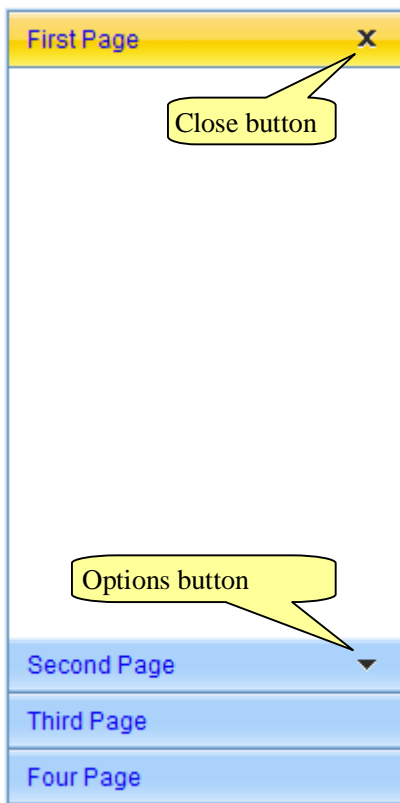
Two events can be fired when "close" button was clicked:

- 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=linb.create("Block").setWidth(400).setHeight(100).show(), tabs;
block.append(tabs=new linb.UI.Tabs({
    value:'a',
    items:[{
        id:'a',                    Close button
        caption:'First Page'
    },{
        id:'b',
        caption:'Second Page'
    }]
}));
_.asyRun(function(){           Adds two pages
    tabs.insertItems([{
        id:'c',
        caption:'Third Page'
    },{
        id:'d',
        caption:'Fourth Page'
    }]);
},500);                        Adds one more
_.asyRun(function(){
    tabs.insertItems('Fifth Page');
},1000);                       Removes this page
_.asyRun(function(){
    tabs.removeItems('d');
},1500);                       Removes two more
_.asyRun(function(){
    tabs.removeItems(['b','c']);
},2000);
```

3.10.5.   Dynamic content loading

3.10.5.1. onIniPanelView

```
    var block=linb.create("Block").setWidth(400).setHeight(100).show(),
    tabs=new linb.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 linb.UI.SButton)
    });
block.append(tabs);
```

3.10.5.2. beforeUIValueSet/afterUIValueSet

It's a fine-grained mechanism.

```
var block=linb.create("Block").setWidth(400).setHeight(100).show(), tabs;
block.append(tabs=new linb.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.$ini){
            profile.boxing().append(new linb.UI.SButton);
            item.$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=linb.create('PopMenu')
.setItems([
    {"id":"itema", "caption":"itema", "tips":"item a"},        Checkbox type
    {"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"},        Sub pop menu
        {"id":"itemd2", "caption":"itemd2"}
]},
    {"id":"iteme", "caption":"iteme", "tips":"item d", disabled:true}
])
.onMenuSelected(function(profile,item){        event        Disabled it
    linb.message(item.id + (item.type=="checkbox"?" : " + item.value:""))
});

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

Output:
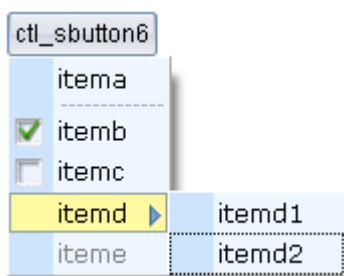
3.11.2.   MenuBar

Input:

```
var pm=linb.create('MenuBar')
.setItems([{
    "id" : "file", "caption" : "File",
    "sub" : [{                          ← Pop menu data
        "id" : "newproject",
        "caption" : "New Project"
    },
    {    "id" : "openproject", "caption" : "Open Project",
        "add" : "Ctrl+Alt+O",           ← Extra data
        "image" : "img/b.gif",
        "sub":["option 1","option 2"]   ← Sub pop menu
    },
    {    "id" : "closeproject",    "caption" : "Close Project"
    },
    {"type" : "split"},                 ← A split
    {    "id" : "save", "caption" : "Save",
        "image" : "img/a.gif"           ← An icon
    },
    {    "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,                      ← Disabled it
    "sub" : [{    "id" : "debug",
        "caption" : "Debug"
    }]
}]).show()
```

Output:
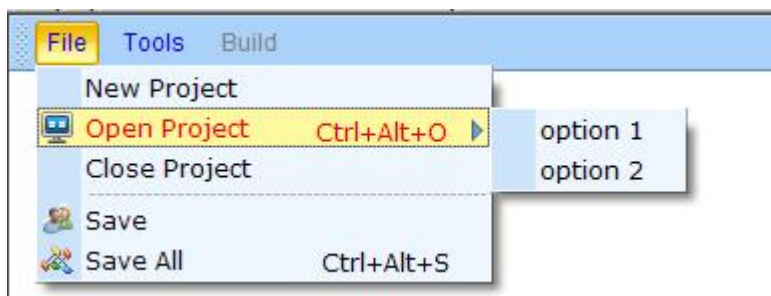
3.11.3.   Toolbars

Input:

```
linb.create('ToolBar',{items:[{
    "id" : "align",                          Button group data
    "sub" : [
        {"id" : "left","caption" : "left"},      Button data
        {"id" : "center","caption" : "center"},
        {type:'split'},                      A split
        {"id" : "right","caption" : "center"}
    ]
},{
    "id" : "code",
    "sub" : [{
        "id" : "format","caption" : "format",    With a label
        label:"label",
        image:"img/a.gif",                   With an icon
        "dropButton" : true
    }]                                       A      drop
}]                Group object    Button
})
.onClick(function(profile,group,item){
    linb.message(group.id + " : " +item.id)
})
.show();
```

Output:

left | center | center | label | format ▾

3.12. TreeBar and TreeView

3.12.1.   Three selection mode

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

3.12.1.1. No-selection

Input:

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

Output:
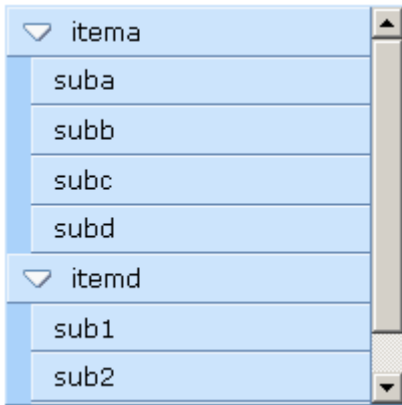
3.12.1.2. Single-selection

Input:

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

Output:
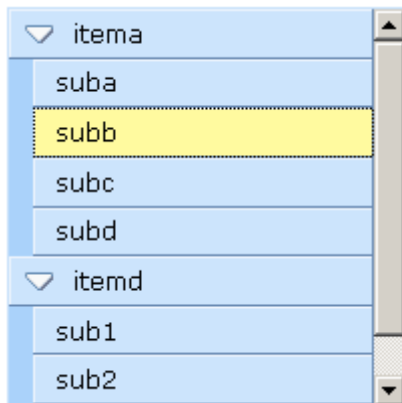
3.12.1.3. Multi-selection

Input:

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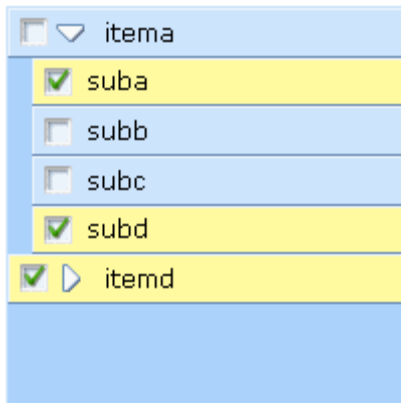
Output:

3.12.2. Group Item

Input:

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

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

Output:
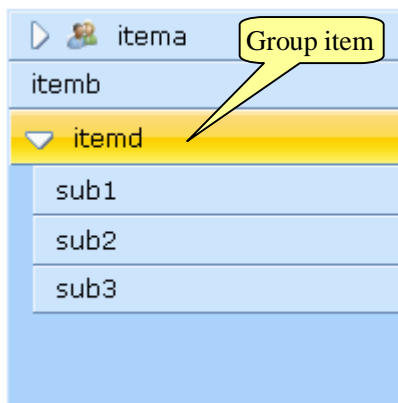
3.12.3. Expand all nodes by default

Input:

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

Output:
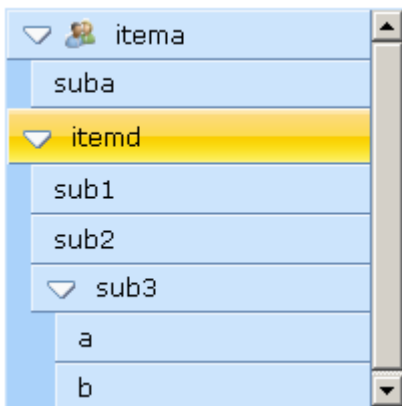
3.12.4. Mutex Expand

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

3.12.5.   Dynamic Destruction

```
var block=new linb.UI.Block({width:200,height:200}).show();
linb.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 linb.UI.Block({width:200,height:200}).show();
linb.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:

itema
  1278994481020-a
  1278994481020-b
  1278994481020-c
itemb

dynamic created and destroyed
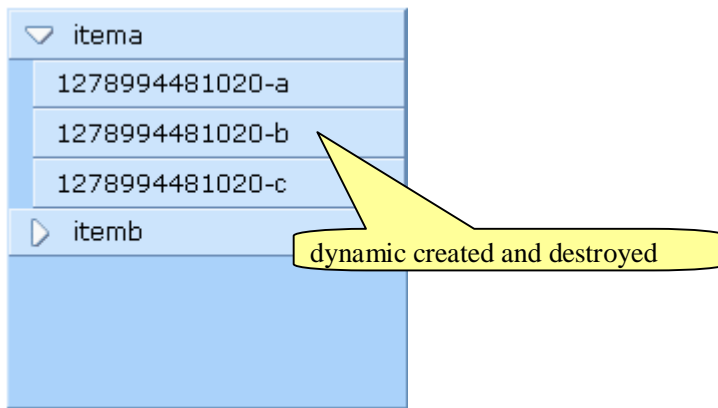
3.13. TreeGrid

3.13.1. Header and Rows

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

```
[                    [key/value pairs]
    {id : "xxx1", caption : "xxx1" …,
     sub: []                The sub [key/value pairs]
    },
    {id : "xxx2", caption : "xxx2" …},
    …
]           If no id specified, will create one automatically
```
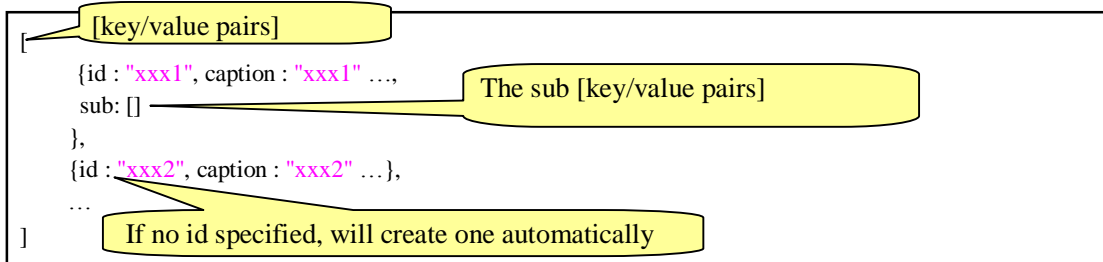
It can be written as a simplified format,

```
[
    "xx1",
    "xx2",          Only id string
    {
        id : "xxx3",
        sub: ["sub1","sub2"]
    }
]          Only id string
```
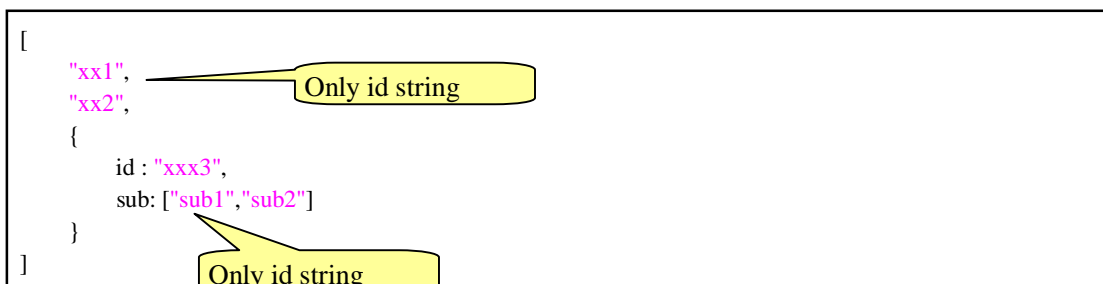
When call setHeader/setRows, the simplified format can be convert to,

```
[
    {id:"xx1",caption:"xx1"},
    {id:"xx2",caption:"xx2"}
    {
        id : "xxx3", caption : "xxx3",
        sub: [
            {id:"sub1",caption:"sub1"},
            {id:"sub2",caption:"sub2"}
        ]
    }
]
```
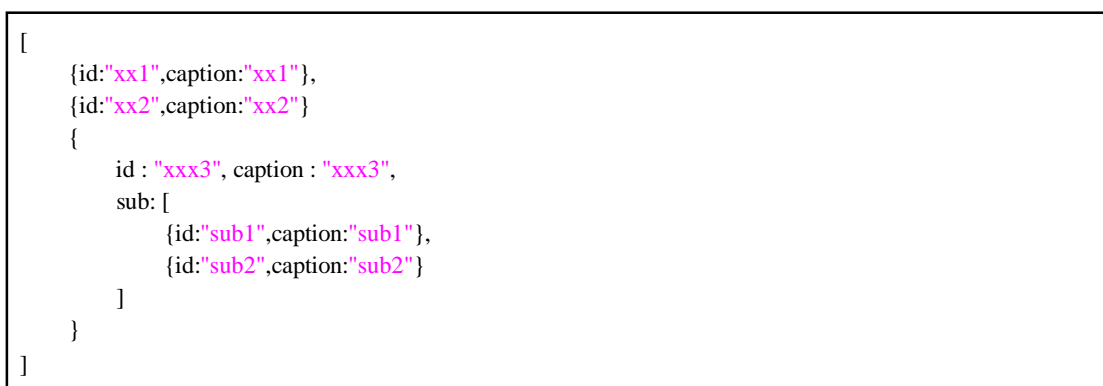
3.13.1.1. Sets standard format

```
var block=new linb.UI.Block({width:200,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)        No row handler
.setHeader([
                                Column's width
    {id:"col1", caption:"Name"},
    {id:"col2", caption:"Age", width:40}
]).setRows([
    {id:"row1",cells:[{          Cells data
        value:'Jack',caption:'Jack'
    },{
        value:23,caption:'23'
    }]]},
    {id:"row2",cells:[{
        value:'John',caption:'John'
    },{
        value:32,caption:'32'
    }]]}
]).show(block);
```

Name	Age	
Jack	23	
John	32	

3.13.1.2. Sets simplified format

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

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

linb.log(tg.getHeader());

linb.log(tg.getHeader("data"));

linb.log(tg.getHeader("min"));
```

Comparing these three formats


```
: ) jsLINB Debug Window                  Clear  X
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"]


>>>
```
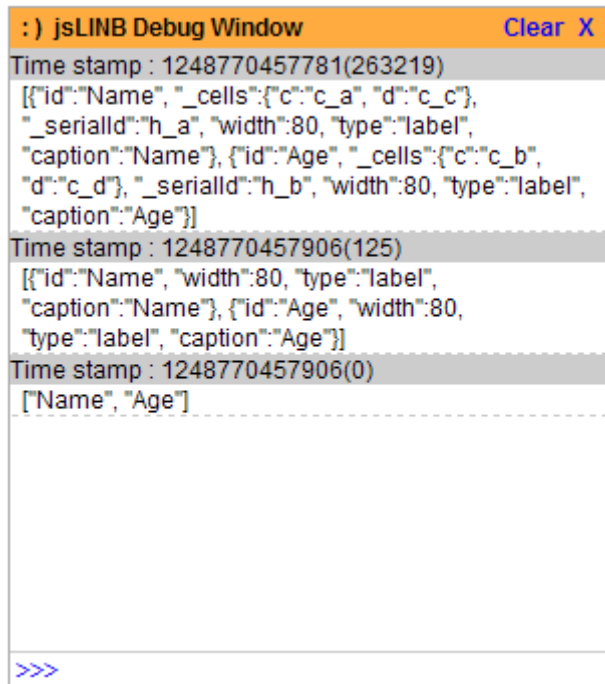
3.13.3.   getRows

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

linb.log(tg.getRows("data"));

linb.log(tg.getRows("min"));

//console.log(tg.getRows());
```

Comparing these three formats

There is circular reference in memory data, can't be directly serialized

The rows memory data in firebug:

3.13.4.   Active Modes

There are three active modes for TreeGrid:

- l   non-active appearance : activeMode is "none";
- l   the row-active appearance: activeMode is "row" ;
- l   the cell-active appearance: activeMode is "cell";

3.13.4.1. non-active appearance

Input:

```
var block=new linb.UI.Block({width:200,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)          No row handler
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])    Sets to 'none'
.setActiveMode("none")
.show(block)


. afterRowActive (function(profile,row){     Does not trigger events
     linb.message(row.id);
})
. afterCellActive (function(profile,cell){
     linb.message(cell.value);
})
```

Output:

non-active appearence

Name	
Jack	2?
John	32

3.13.4.2. row-active appearance

```
var block=new linb.UI.Block({width:200,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)          No row handler
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])    Sets to "row"
.setActiveMode("row")
.show(block)

                                 Will be fired
.afterRowActive (function(profile,row){
     linb.message(row.id);
})
.afterCellActive (function(profile,cell){
     linb.message(cell.value);
})
```

non-active appearence

Name	
Jack	?
John	32

3.13.4.3. cell-active appearance

```
var block=new linb.UI.Block({width:200,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)                    No row handler
.setHeader(["Name", "Age"])
.setRows([['Jack', 23], ['John', 32]])     Sets to "row"
.setActiveMode("cell")
.show(block)


.afterRowActive (function(profile,row){
    linb.message(row.id);
})                                         Will be fired
.afterCellActive (function(profile,cell){
    linb.message(cell.value);
})
```

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


.afterUIValueSet(function(profile,ovalue,value){   Won't be fired
    linb.message(value);
});
```

Output:

It's active appearance, not the selection

Input:

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

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

Non-active

Non-selection

Won't be fired

Output:

Non-active appearance, non-selection

3.13.5.2. Single row selection

Input:

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

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

Sets to 'single' mode

Will be fired

Output:

Selection appearance

3.13.5.3. Multi-row selection

Input:

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


.afterUIValueSet(function(profile,ovalue,value){
    linb.message(value);          Will be fired
});
```

Output:

	Name	Age	
☑	Jack	23	
☑	John	32	

3.13.5.4. Single cell selection

Input:

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


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

Output:

Name	Age	
Jack	23	
John	32	

3.13.5.5. Multi-cells selection

Input:

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


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

Output:

Name	Age	
Jack	23	
John	32	

3.13.6. The Tree Grid

Input:

```
var block=new linb.UI.Block({width:200,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandlerWidth(20)        Row header is a must for tree grid
.setHeader([
    {id:"col1", caption:"Name"},
    {id:"col2", caption:"Age", width:40}
]).setRows([
    {id:"row1",cells:['Jack',23]},
    {id:"row2",cells:['John',32],        A row has sub rows
     sub:[{id:"row21",cells:['Tom',24]},
          {id:"row22",cells:['Bob',25]}
    ]}
]).show(block)
```

Output:

	Name	Age	
	Jack	23	
▼	John	32	
	Tom	24	
	Bob	25	

No Indentation here

Input:

```
var block=new linb.UI.Block({width:200,height:200}).show();
var tg=new linb.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:

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 linb.UI.Block({width:200,height:200}).show();
var tg=new linb.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:

上一节中缩进的例子

3.13.7.2. Column width

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandlerWidth(80)         The first column's width
.setGridHandlerCaption("Name")
.setHeader([                       Column's width
    {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(){              Modify column width dynamically
    tg.updateHeader("col2", {width:70});
},1000);
```

Output:

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

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandlerWidth(80)                    coarse-grained setting
.setColResizer(false)
.setGridHandlerCaption("Name")               fine-grained setting
.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)
```

Output:

Only this column

3.13.7.4. Drag&Drop to modify column position

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandlerWidth(80)           coarse-grained setting
.setColMovable(false)
.setGridHandlerCaption("Name")          fine-grained setting
.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)
```

Output:

3.13.7.5. Default Sorting

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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...

Sorting icon

3.13.7.6. Custom Sorting

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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 linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

3.13.7.8. Setting Cell Types in column header

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

3.13.7.9. column header style

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

Naem	Age	Part-time	

3.13.7.10. column header icon

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

Name	Age	Part-time	

3.13.7.11. Update column header dynamically

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

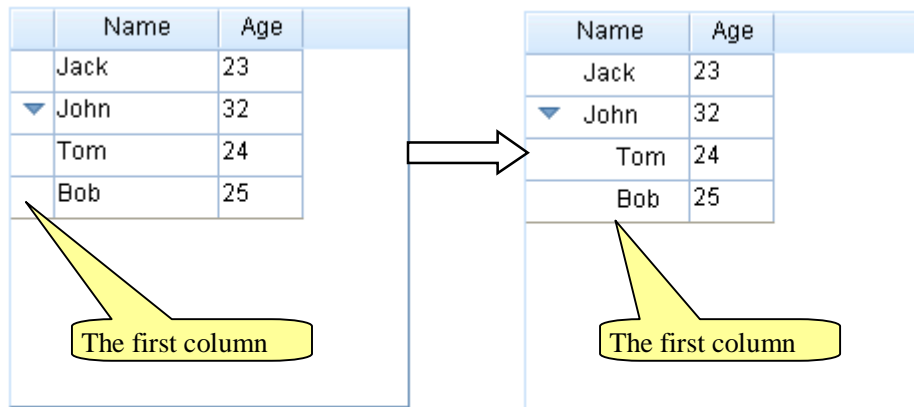```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setGridHandlerCaption("Name")
.setHeader([
    {id:"col1", caption:"Age", width:40, type: "number"},
    {id:"col2", caption:"Full-time", width:90, type: "checkbox"}
]).setRows([
    {id:"row1",caption:'Jack',cells:[23,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

Sets row height

**Output:**

## 3.13.8.2. Drag&Drop to modify row height

**Input:**

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

Output:

3.13.8.3. Setting cell type in row

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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 linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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 linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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 Bot	25	☐

Default format line numbers

3.13.8.6. Custom row numbers

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

3.13.8.7. Alternate Row Colors

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandlerWidth(80)
.setAltRowsBg (true)          Sets alternate bg color
.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)
```

Output:

3.13.8.8. Group

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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

Group row

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

3.13.8.9. Preview and Summary region

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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

3.13.8.10. Update row dynamically

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

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 linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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 linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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:

Setting in row

Setting in column header

Name	Age	Full-time
Jack	23	☑
John	32	☐

Setting in cell

3.13.9.3. Update cell dynamically

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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".

- l   TreeGrid's editable =>false; cell's editable=>true: only this cell is editable
- l   TreeGrid's editable =>false; column header's editable=>true: only this column is editable
- l   TreeGrid's editable =>false; row's editable=>true: only this row is editable
- l   TreeGrid's editable =>true; cell's editable=>true: only this cell is uneditable
- l   TreeGrid's editable =>true; column header's editable=>false: only this column is uneditable
- l   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 linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(true)                    [Sets editable]
.setHeader([                                              [List for editor]
{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'}],          [Value and caption]
    ['John',25, {value:'female',caption:'Female'}]
]).show(block)
```

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 ⌄
		Male
		Femal

3.13.10.2. Editable column

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(false)               [Sets uneditable]
.setHeader([                                       [This column is editable]
{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)
```

Output:

3.13.10.3. Editable row

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(false)          Sets uneditable
.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([                                    This row is editable
    ['Jack',23, {value:'male',caption:'Male'}],
    {cells:['John',25, {value:'female',caption:'Female'}],editable:true}
]).show(block)
```

Output:

3.13.10.4. Editable cell

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.UI.TreeGrid;
tg.setRowHandler(false)
.setEditable(false)          Sets uneditable
.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([                            Only this cell is editable
    ['Jack',23, {value:'male', caption:'Male'}],
    ['John',25, {value:'female', caption:'Female',   editable:true }]
]).show(block)
```

Output:

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 linb.UI.Input control
- 'textarea': multi lines linb.UI.Input control
- 'number': number only linb.UI.Input control
- 'currency': currency only linb.UI.Input control
- 'progress': linb.UI.ComboInput control, spin
- 'combobox': linb.UI.ComboInput control, combobox
- 'listbox': linb.UI.ComboInput control, listbox
- 'getter': linb.UI.ComboInput control, getter
- 'helpinput': linb.UI.ComboInput control, helpinput
- 'cmdbox': linb.UI.ComboInput control, cmdbox
- 'popbox': linb.UI.ComboInput control, popbox
- 'time': linb.UI.ComboInput control, time
- 'date': linb.UI.ComboInput control, date
- 'color': linb.UI.ComboInput control, color

Input:

```
var block=new linb.UI.Block({width:300,height:340}).show();
var tg=new linb.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 linb.UI.Block({width:300,height:340}).show();
var tg=new linb.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 linb.UI.Input({valueFormat:"^[\\w\\.=-]+@[\\w\\.-]+\\.[\\w\\.-]{2,4}$"});
        return editor;
    }

    if(t=='popwnd'){
        var dlg=linb.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
Linb.U.Input or CombInput

Return false for advanced custom editor

Output:

Type	Cell UI
email	a@b.com
popwnd	

Specify it   x

Update

value

OK   Cancel

The custom editor
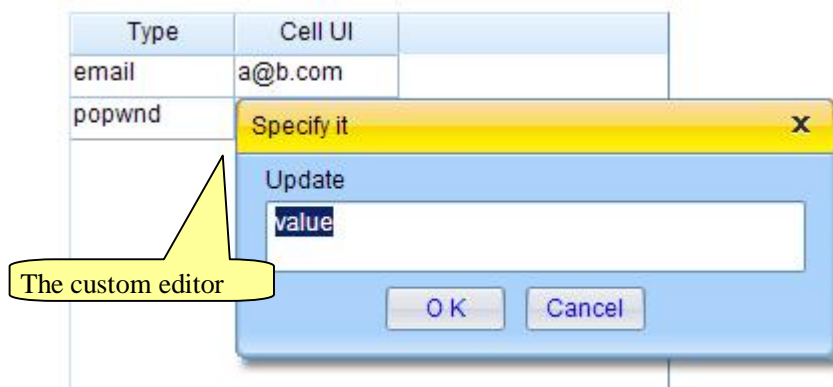
3.13.11. Add/Remove rows

Input:

```
var block=new linb.UI.Block({width:240,height:200}).show();
var tg=new linb.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([[]])         Adds a empty row
},1000);
_.asyRun(function(){
    tg.insertRows(["Tom",30])    Adds a new row
},2000);
_.asyRun(function(){
    tg.insertRows([{id:'row3',cells:['Jerry',19]},['Mark',31]])   Adds two rows
},3000);
_.asyRun(function(){
    tg.removeRows('row1')        Removes a row by id
},4000);
_.asyRun(function(){
    tg.removeRows(['row2','row3'])   Removes two row by ids
},5000);
_.asyRun(function(){            Adds a row to the top
    tg.insertRows([{id:'row4',cells:['Jack',23]}],null,null,true)
},6000);
_.asyRun(function(){           Adda a row next to 'row1'
    tg.insertRows([['John',23]],null,'row1',false)
},7000);
```

3.14. Other standard controls

3.14.1. ProgressBar

Input:

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

Sets text display template

percentage

Output:

80% finished!
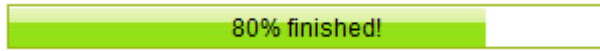
3.14.2.   Slider

Input:

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

linb.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

Vertial slider
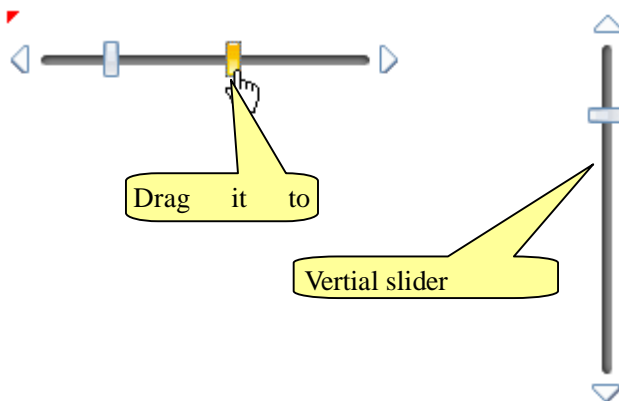
3.14.3. Image

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

linb.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
linb.create('PageBar')
.setValue("1:5:12")
.onClick(onclick)
.show();
// another PageBar
linb.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:

Page: 1 ... < 5 > ... 12

[1] [...] << [5] >> [...] [12]

[7] [8] [9] [10] [11]

Chapter 4. Data exchanging(Ajax)

jsLinb 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 jsLinb:
- linb.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.
- linb.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 "linb.SAjax.$response"), and a "id" parameter (the uniquely identify).

    Server's return data must be the following format:

    ```
    linb.SAjax.$response({id: "12483145855311"/*,data*/})
    ```

- linb.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 an "id" parameter (the uniquely identify).

    Server's return data must be the following format:

    ```
    <script type='text' id='json'>{"id":"12483161278402"/*,data*/}]}</script>
    <script type='text/javascript'>
    window.name=document.getElementById('json').innerHTML;
    </script>
    ```

"linb.request" function can choose an appropriate class from linb.Ajax, linb.SAjax or linb.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

linb.Ajax can get file contents easily.
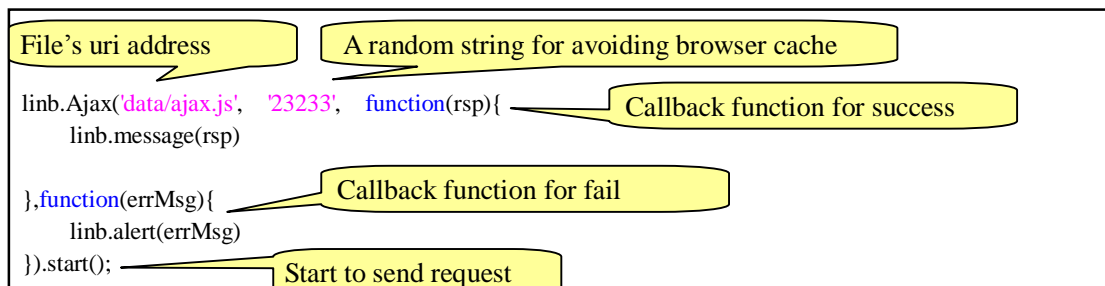
In Fiddler:

4.3. Synchronous data exchange

Only linb.Ajax support synchronous data exchanging.

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

Request data → (points to `key:'test',`)

synchronous → (points to `asy:false`)

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:

```
Transformer  Headers  TextView  ImageView  HexView  WebView  JSON  Auth  Caching  Privacy  Raw
{"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";
linb.Ajax(url, {
     key:'test',
     para:{p1:'para 1'}
},function(rsp){
     linb.log(rsp);
},function(errMsg){
     linb.alert(errMsg)
}).start();
```

4.4. Cross-domain

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

4.4.1. To monitor SAjax

Code:

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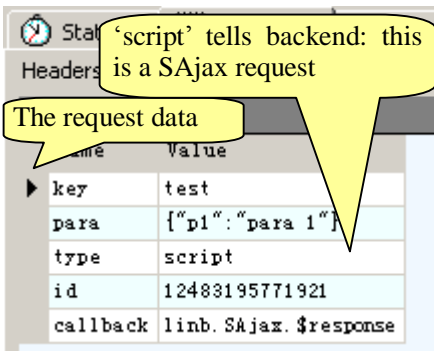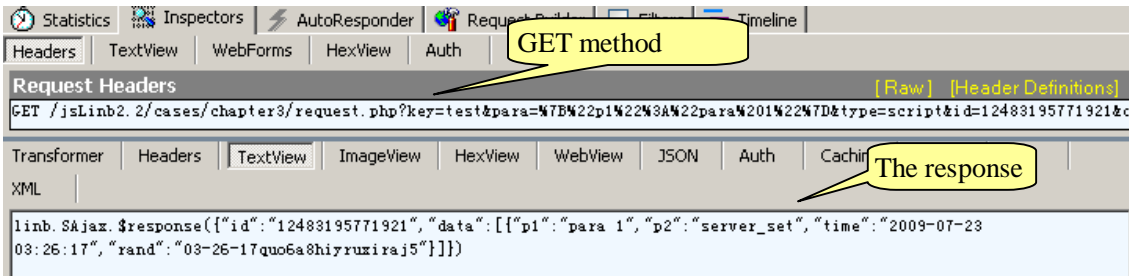
In Fiddler:

4.4.2. To monitor IAjax

Code:

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

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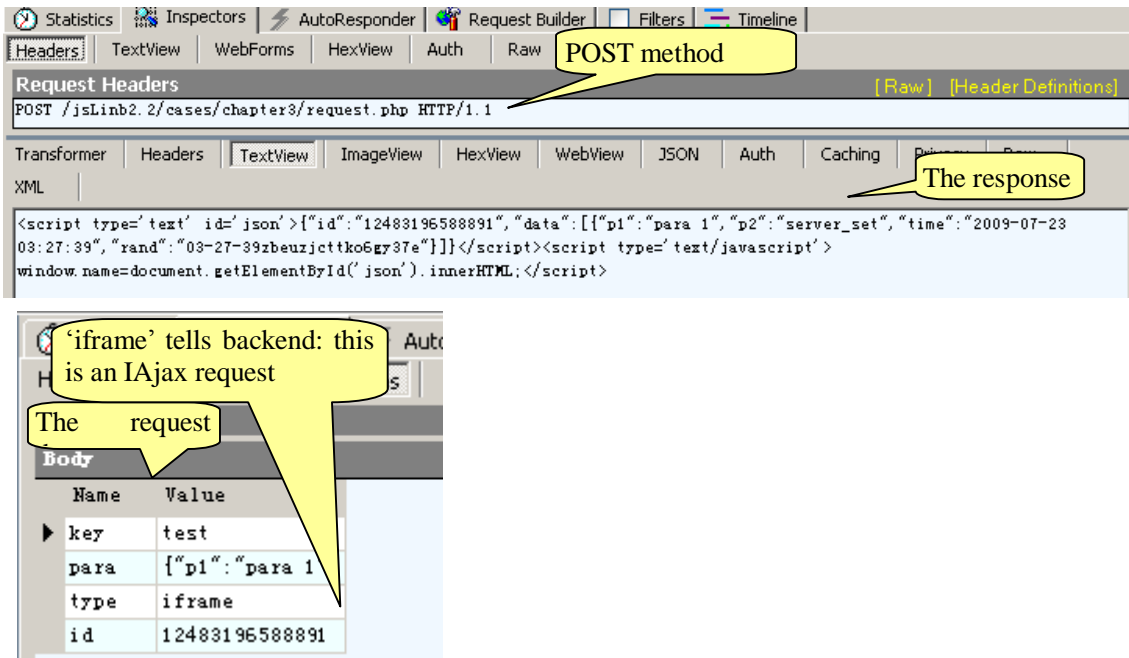
By default, IAajax use "POST" method, you can specify method in options.

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

4.5. File Upload

Only linb.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 "upload":

chapter3/upload/index.html

4.5.2. Upload by IAjax

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

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

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

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

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

4.6. A request wrapper for real application

In practical applications, you can choose linb.Ajax, linb.SAjax and linb.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,                 Service url address
    requestData,
    onOK,                                 Request data (key/value pairs)
    onStart,
    onEnd,                                Callback for successful call
    file           File to upload
){
                                          Callback for onStart and onEnd
    _.tryF(onStart);
    linb.observableRun(function(threadid){
        var options;
        if(file){
            requestData.file=file;
            options={method:'post'};
        }
        linb.request(service, requestData, function(rsp){
            if(typeof rsp=='string')
                rsp=_.unserialize(rsp);        linb.Ajax returns string, not js object

            if(rsp){
                if(!rsp.error)
                    _.tryF(onOK, [rsp]);       Success
                else
                    linb.pop(_.serialize(rsp.error));   Fail
            }else{
                linb.pop(_.serialize(rsp));    Fail
            }
            _.tryF(onEnd);
        },function(rsp){
            linb.pop(_.serialize(rsp));    Fail
            _.tryF(onEnd);
        }, threadid,options)
    });
};
```

4.7. XML Data

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

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

4.8. An overall example

The following is an overall example for   data exchanging.

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 "cases\chapter4\distributed\App\js\", "Module3.js" includes a Class named "App.Module3". Let's try to call it.

Input:

```
Namespace("App");                              ← Namespace is "App"
linb.include("App.Module3",                    ← Dynamic asynchronous remote file loading
    linb.getPath("chapter4/distributed/App/js/","Module3.js"),
    function(){
        var ins=new App.Module3();             ← Create instance and show it
        ins.show();
    },function(){
        linb.alert("fail");
    }
);
```

Output:

dialog in Module3

Code in Module3.js

UI in Module3

button in Module3
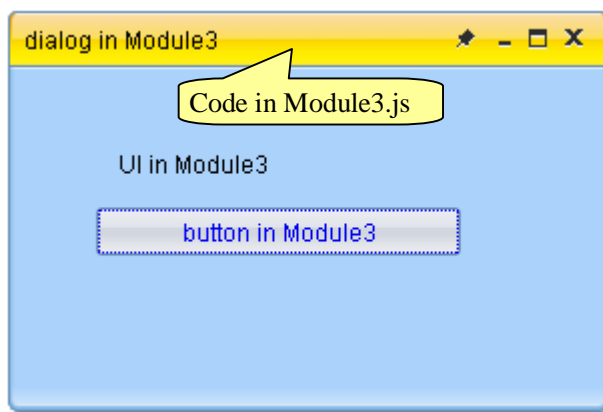
And try to load code and create UI from a difference domain.

```
Namespace("App");
linb.include("App.Module3",
"http://www.linb.net/Samples/linb/app/distributed/App/js/Module3.js",
function(){
        var ins=new App.Module3();
        ins.show();
    },function(){
        linb.alert("fail");
    }
);
```

A difference domain

5.2. linb.Com and linb.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 linb.Com and linb.ComFactory to load those "distributed UI". In order to use this approach, all those Classes must be derived from the linb.Com, named according to specified rules, and put into the specified directory.

linb.ComFactory implements a management mechanism for the linb.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.

Conf file

Distributed UI classes

This folder
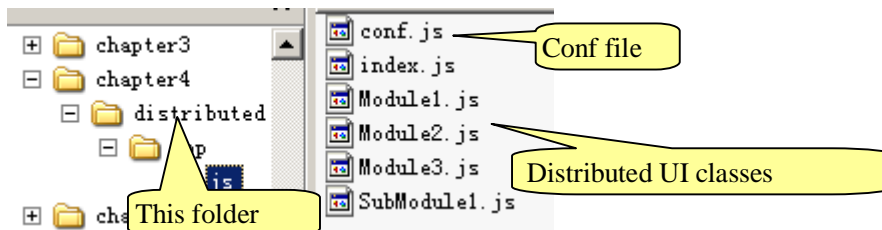
5.2.1. linb.ComFactory config

In conf.js:

```
CONF={ComFactoryProfile:
{
    module1:{                    module1 is an "Aoo.Module1" Class
        cls:'App.Module1',
        children:{               module1 has a child submodule1
            tag_SubModule1:'submodule1'
        }
    },
    submodule1:{                 submodule1 is a "Aoo.SubModule1" Class
        cls:'App.SubModule1'
    }
}}
```

Loading this configuration to linb.ComFactory:

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

5.2.2. linb.Com.Load

In file index.html,

```
linb.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

Load module1

Load Module2

Load Module3 manually
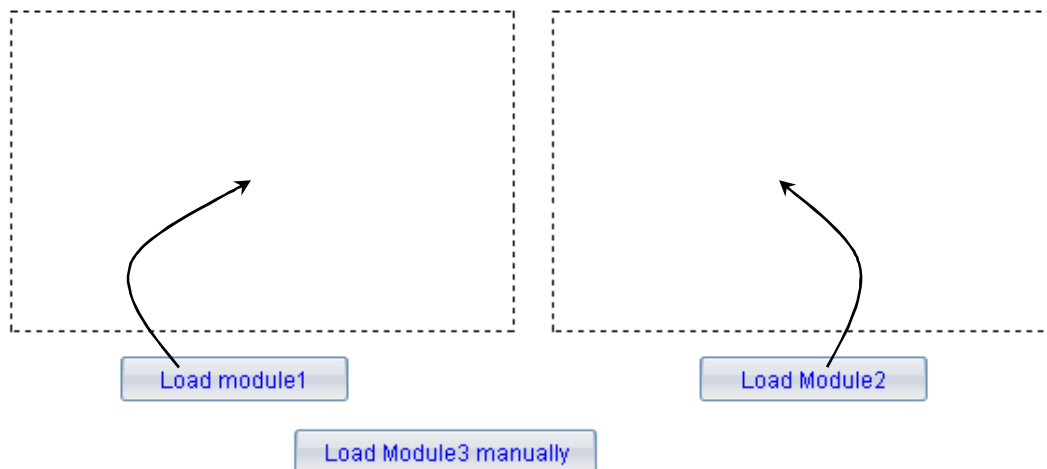
5.2.3. newCom and getCom

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

```
_button3_onclick:function (profile, e, value) {          newCom
    var host=this;
    linb.ComFactory.newCom('App.Module3',function(){          Callback function
        this.show(linb([document.body]));
    });                                                   Uses Class Name
}
         Shows to HTML body
```
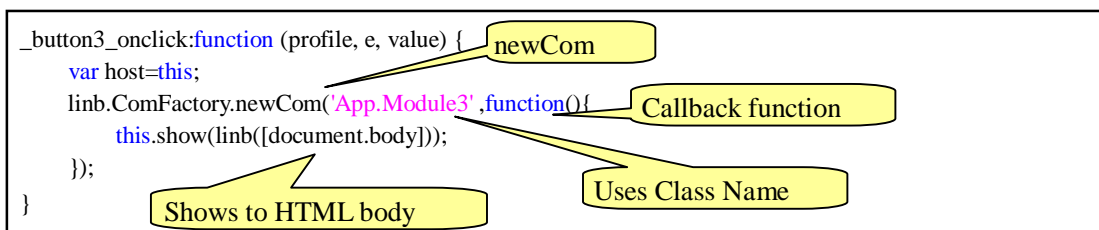
[linb.CombFactory.newCom("App.Module3"..], will:

- 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) {          getCom
    var host=this;
    linb.ComFactory.getCom('module1',function(){          Callback function
        var ns=this;
        host.div16.append(ns.getUIComponents(),false);    From config
    });
}       Shows to the left block
```
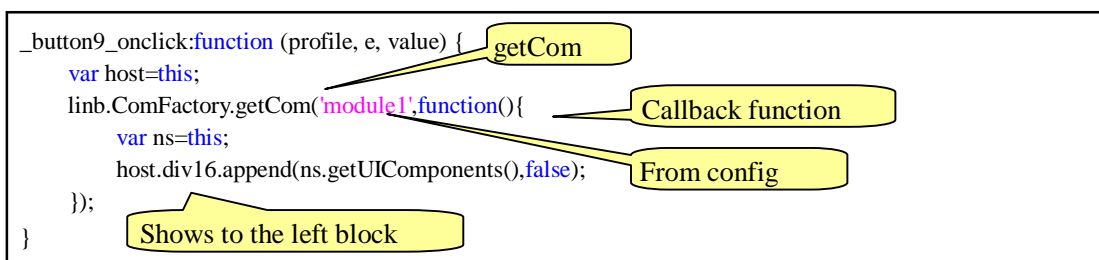
[linb.CombFactory.newCom("module1"..], will:

- find config from linb.CombFactory
- 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) {          getCom
    var host=this;
    linb.ComFactory.getCom('App.Module2',function(){          Callback function
        var ns=this;
        host.div16.append(ns.getUIComponents(),false);
    });                                                   "Class Name" works too.
}       Shows to the right block                          No config needed in this case.
```
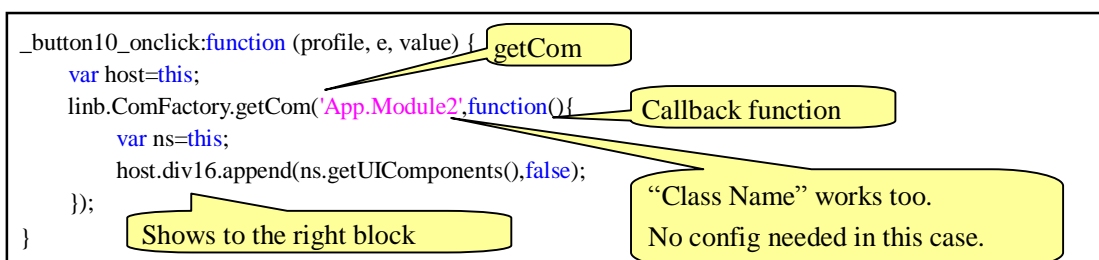
By default, the instance created by "getCom" is singleton, and will be cached in inb.CombFactory.

5.2.4. linb.UI.Tag

There's a linb.UI.Tag object in file Module1.js:

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

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.
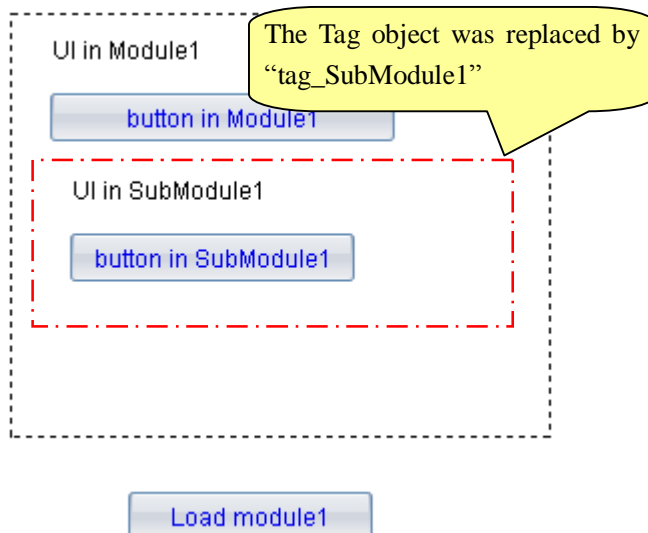
5.2.5. Destroy com

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 "**linb.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:

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

Chapter 6. Some fundamental things

6.1. Pop-up window

6.1.1. alert window

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

Fired after user close the window
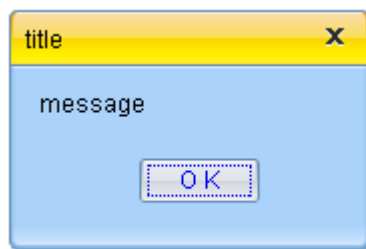
a

6.1.2. confirm window

```
linb.confirm('title','confirm?',function(){
        linb.message('You confirmed it')
    },
    function(type){
        linb.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
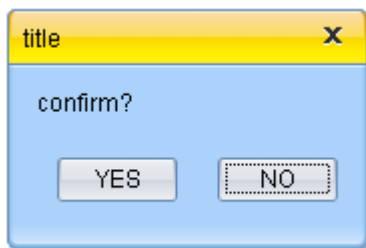
6.1.3. prompt window

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

Fired when user click "YES"

Fired when user click "NO" or click close button
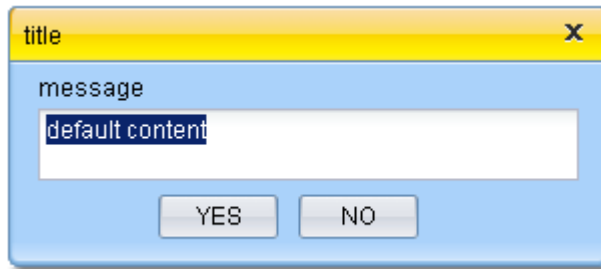
6.1.4. pop window

```
linb.pop('title', 'message','Button Caption',50,100);
```
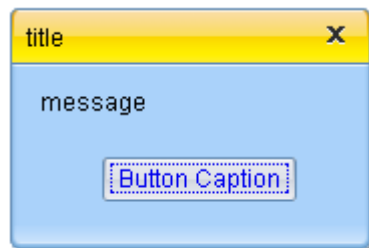
6.2. Asynchronous execution

6.2.1. asyRun

_.asyRun is a wrapper for settimeout.

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

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){
    linb.pop("this===linb:"+(this===linb), arg)
},500,['The previous one'],linb)

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

The latter one will be executed

this===linb:true
The last one
O K
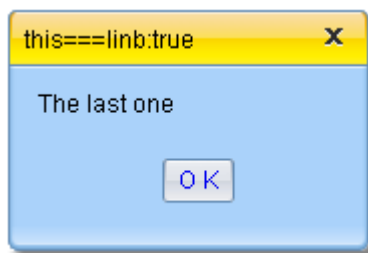
6.3. Skin switcher

6.3.1. Switch skin for whole application

There are three system skins in jsLinb3.0: default, vista and aqua. You can use linb.UI.setTheme to switch the skin.

```
linb.UI.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

linb.alert(linb.getLang())   Gets the current locale key

linb.setLang("cn");   Sets locale, switch locale dynamically (no page refresh)
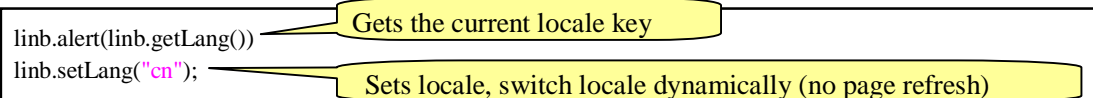
Example "chapter5\lang"   loading process:

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 linb-all.js simultaneously

After DOM is ready, jsLinb lib will handle UI Class à App

File 1

[linb.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 [jsLinb path] / jsLinb / Locale / en.js asynchronously

File 3

Loading application locale file Locale/en.js asynchronously

[linb.Com.load] new an instance of Class App, and

6.5. DOM Manipulation

Class "linb.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;
linb('body').append(firstNode=linb.create('<div style="border:solid 1px">the first div</div>'));   It's the fist node
firstNode.append('input')
                                    Appends to firstNode
.last().attr('value','append')
                            Finds input, and sets
.parent()
                    Returns to firstNode
.prepend('<button>prepend</button>')
                                    Prepend a button
.addPrev('<div style="border:solid 1px">prev div</div>')        Adds to previous sibling
.addNext('<div style="border:solid 1px"> div</div>')            Adds to next sibling
```

output:

```
addPrev                                                         append
                        prev div
                        prepend   the first div append
addNext
                        next div                    prepend
```

6.5.2. Attributes and CSS

Input:

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

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

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

                                                    Gets CSS fontSize
_.asyRun(function(){
     linb.message(node.css('fontSize'))
     node.css({background:'#00ff00',fontSize:'16px'});    Updates fontSize and backgorund
},2000);

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

_.asyRun(function(){
     linb.message(node.last().attr('value'))             Gets input' value attr
     node.last().attr('value','updated');
},4000);                                                 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;
linb('body').append(node=linb.create('div'));

_.asyRun(function(){                         Adds classes
    node.addClass("cls1 cls2 cls3");
    linb.message(node.hasClass('cls2'));      Determines whether a class
    node.text(node.attr('className'));        name exists or not
},1000);

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

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

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

_.asyRun(function(){           Remove tag
    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 jsLinb: [before-], [on-] and [after-].

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

l   linb(/**/).onClick(null,null,true) => removes all event functions in [beforeclick] group, [onclick] group and [afterclick] group;

l   linb(/**/).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;

l   linb(/**/).onClick(true) => fire event, executes all event functions in [beforeclick] group, [onclick] group and [afterclick] group in order;

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

node.onClick(function(){                    Adds a onClick event
    alert('onClick');
    return false;
})
.beforeClick(function(){                    Adds a beforeClick event
    alert('beforeClick');
})
.afterClick(function(){                     Adds an afterClick event
    alert('afterClick');
});                           Fires all click events. Since onClick returns false,
                              afterClick will not be fired.
node.onClick(true);
                              Removes onClick event;
_.asyRun(function(){
    node.onClick(null);
    node.onClick(true);       Fires all click events. Since onClick was
},2000);                      removed, afterClick will be fired this time.
```

6.5.5. Node Drag&Drop

Input:

```
var btn,div;
linb('body').append(btn=linb.create("<button>drag me</button>"))
.append(div=linb.create("<div    style='position:absolute;left:100px;top:100px;border:solid    1px;width:150px;
height:150px;display:block;'> drop here </button>"))
                                            Sets draggable
btn.draggable(true,{dragType:'icon'},'dragkey','dragdata');
div.droppable(true,'dragkey')
.onDrop(function(){                Sets droppable
    linb.alert(linb.DragDrop.getProfile().dragData);     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 linb.DragDrop.getProfile(). The profile object:

- dragType: 'move','copy','deep_copy','shape','icon', 'blank' or 'none', Default is 'shape';
- shadowFrom: DOM element or linb.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 [linb.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 [linb.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:Number, the offset between [the restricted bottom] and [the current mouse Y], for mouse restricted region, Default is [null];
- maxLeftOffset:Number, the offset between [the restricted left] and [the current mouse X], for mouse restricted region, Default is [null];
- maxRightOffset: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 linb.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];

l   targetHeight: Number, the drag target node's CSS height, Default is [null];

l   targetOffsetParent: linb.Dom Object, the drag target node offsetParent node, Default is [null];

l   dragCursor: 'none', 'move', 'link', or 'add', the drag cursor key; [readonly]

l   x: Number, current X value of mouse; [readonly]

l   y: Number, current Y value of mouse; [readonly]

l   ox: Number, original X value of mouse; [readonly]

l   oy: Number, original Y value of mouse; [readonly]

l   curPos: {left:Number,top:Number}, current CSS pos of the dragging node [readonly]

l   offset: {x:Number,y:Number}, offset from now to origin [readonly]

l   isWorking: Boolean, is dd working or not? [readonly]

l   restrictedLeft: Number, the calculated restricted left value; [readonly]

l   restrictedRight: Number, the calculated restricted right value; [readonly]

l   restrictedTop: Number, the calculated restricted top value; [readonly]

l   restrictedBottom: Number, the calculated restricted bottom value; [readonly]

l   proxyNode: linb.Dom Object, the proxy Object; [readonly]

l   dropElement: String, the target drop element DOM id. [readonly]

There is an DD overall example in **chapter3/dd/ddProfile.html.**
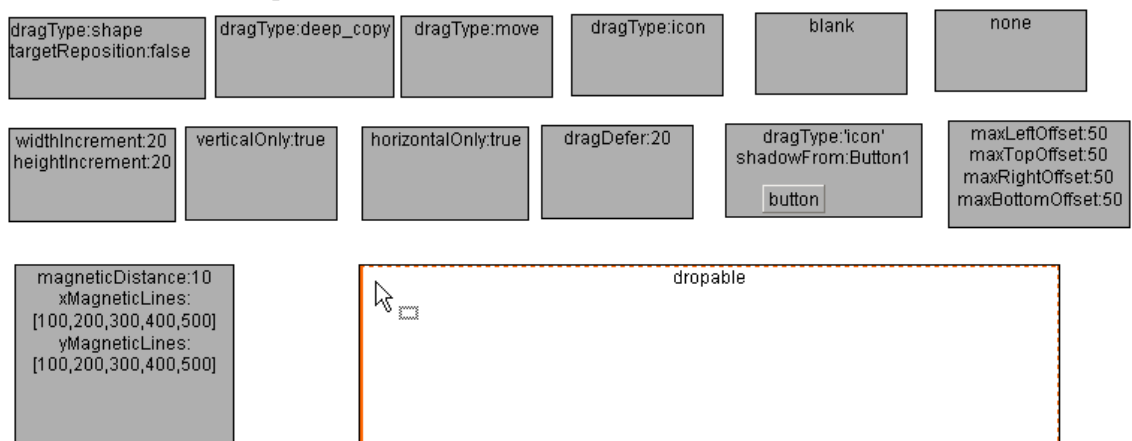
6.5.5.2.   Events in Drag&Drop

For that node in dragging,

l   onDragbegin

l   onDrag

l   onDragstop

For that droppable node,

l   onDragenter

l   onDragleave

l   onDragover

l   onDrop

Input:

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

linb('body').append(elist=linb.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:

6.6. linb.Template

Linb.Template is a completely independent UI wrapper. It doesn't depend on linb.UI Class and all its derived Classes.

6.6.1. example 1

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

Input:

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

Output:

click me

6.6.2. example 2

Input:

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

Output:

6.6.3. A SButton based on linb.Template

"chapter5\SButton" is an example for creating a linb.UI.SButton like control based on linb.Template.

   Output:

6.7. About Debugging

6.7.1. The code package for debugging

In folder "runtime/jsLinb/js/", All files ending with "-debug.js" are for debugging purpose.
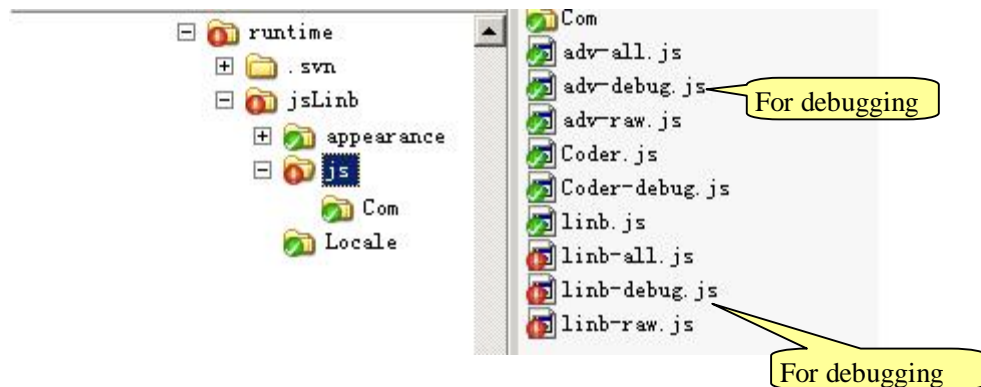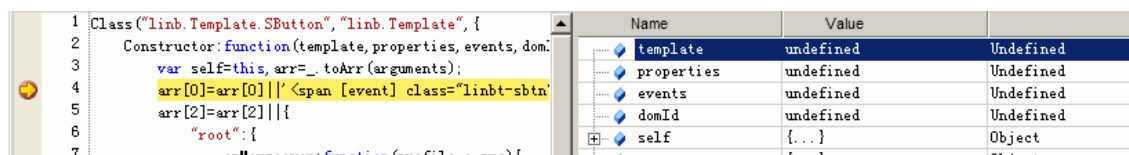

6.7.2. Debugging Tools

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

FireBug:

Developer Tools in IE8:

6.7.3. jsLinb Monitor Tools

jsLinb has a variable monitor tools. It's cross browser. Call linb.log("xxx") to show the monitor window:

Chapter 7. Some typical issues

7.1. Layout

7.1.1. Docking

Input:

```
linb.create('Block',{dock:"top",
    height:80,html:'top'         At top
}).show();
linb.create('Block',{dock:"bottom",    At
    height:30,html:'bottom'
}).show();
linb.create('Block',{dock:"left",    Left side
    width:150,html:'left'
}).show();
linb.create('Block',{dock:"right",    Right side
    width:150,html:'right'
}).show();                              The main area
linb.create('Block',{dock:"fill",
    html:'main',                              Sets docking margin
    dockMargin:{left:10,right:10,top:10,bottom:10}
}).show();
```

Output:

```
top



left      main                                           right



bottom
```

7.1.2. linb.UI.Layout

Input:

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

linb.create('Layout',{type:'horizontal',        Horizontal layout
    items:[{
        pos:'before',                    Left side
        id:'top',
        size:150
    },{
        pos:'after',                     Right side
        id:'bottom',
        size:150
    }]
}).show(layout1);
```

Drag&Drop to resize
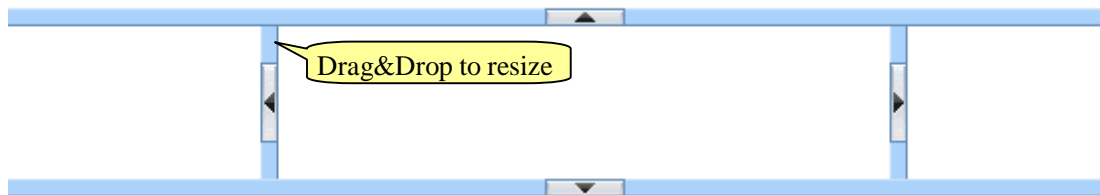
7.1.3. Relative Layout

Input:

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

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

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

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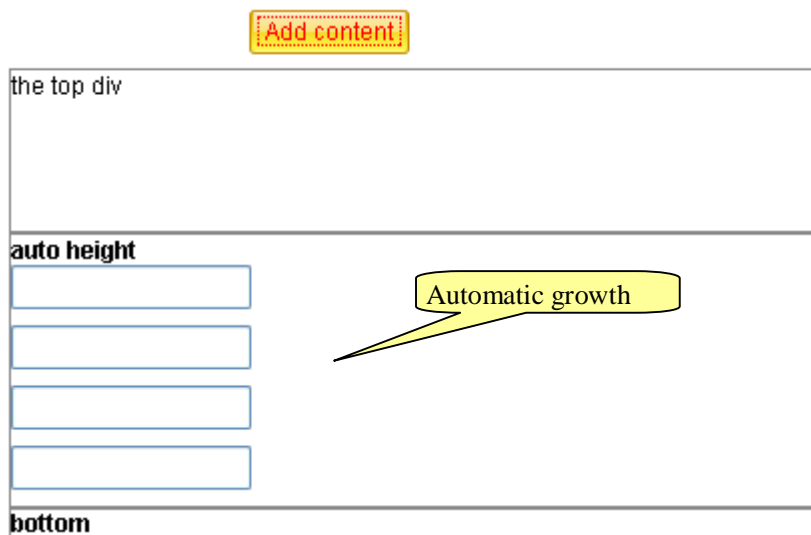
Output:

7.2. UI Control's Drag&Drop

## 7.2.1. Drag&Drop control among containers

Input:

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

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

Output:
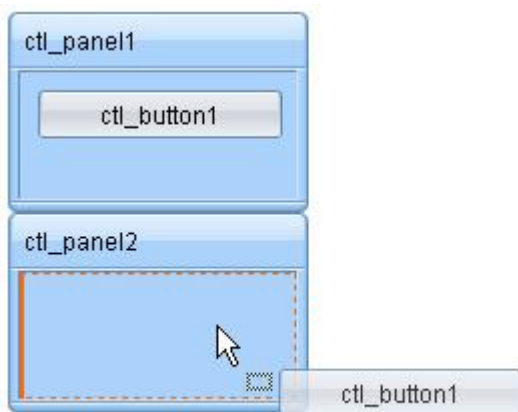
7.2.2. List sorting 1

Input:

```
linb.create("List",{
    items:["item a","item b","item c","item d"]
})
.setDragKey("list")          Sets drag key and drop keys.
.setDropKeys("list")
.show()
```
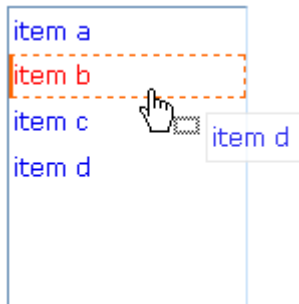
Output:

7.2.3. List sorting 2

Input:

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

Output:
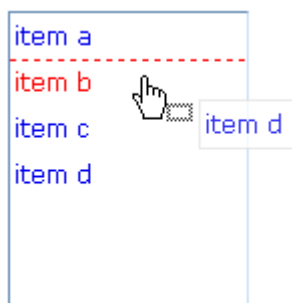
7.3. Form

7.3.1. Form 1

Input:

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

Output:

7.3.2. DataBinder

Input:

```
Class.destroy('App');
Class('App', 'linb.Com',{                    Code created by Designer
    Instance:{
        iniComponents:function(){
            // [[code created by jsLinb UI Builder
            var host=this, children=[], append=function(child){children.push(child).get(0))};    Adds a DataBinder, sets name property
            append((new linb.DataBinder).setHost(host,"binder").setName("binder"))
            append((new linb.UI.SLabel                                    Sets dataBinder and dataField
                .setHost(host,"slabel1").setLeft(80).setTop(60).se          to each control
            append((new linb.UI.SLabel
                .setHost(host,"slabel2").setLeft(80).setTop(90).setCaption("Age:").setWidth(44));
            append((new linb.UI.Input) .setDataBinder("binder").setDataField("name")
                .setHost(host,"iName").setLeft(130).setTop(60).setValueFormat("[^.*]").setValue("Jack"));
            append((new linb.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 linb.UI.SCheckBox) .setDataBinder("binder").setDataField("isfull")
                .setHost(host,"cFull").setLeft(130).setTop(130).setCaption("Full time"));
            append((new linb.UI.SButton)
                .setHost(host,"submit").setLeft(130).setTop(170).setCaption("SUBMIT").onClick("_submit
_onclick"));
            return children;
            // ]]code created by jsLinb UI Builder
        },
        _submit_onclick:function (profile, e, src, value) {          Form validation
            if(!this.binder.checkValid()){                    Collects data
                linb.alert('One or some invalid fields exits!');
                return;
            }
            linb.alert(_.serialize(this.binder.getValue()))
    }}
});
(new App).show();
```

Output:

7.4. Custom UI Styles

7.4.1. Custom only one instance only - 1

Input:

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

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

Theme key words

Sets theme to this instance

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

Output:

7.4.2. Custom only one instance only - 2

Input:

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

Output:

Use setCustomStyle

7.4.3. Custom only one instance only - 3

Input:

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

(new linb.UI.SButton)                Adds CSS. You should put those into a CSS
.setCaption("Use setCustomClass ")   file in your real application
.setCustomClass({
    FOCUS:"my-class"
})                          Custom FOCUS node className
.show();
```

Output:

Use setCustomClass

7.4.4. Custom only one instance only - 4

Input:

```
linb.CSS.remove("id","my_css");
linb.CSS.addStyleSheet("#myctrl1 .linb-sbutton-focus{font-weight:bold;color:#ff0000;}", "my_css");
                            Adds CSS. You should put those into a CSS
(new linb.UI.SButton)       file in your real application
.setCaption("Use domId")
.setDomId("myctrl1")
.show();            Gives a domId
```

Output:

Use domId

7.4.5. Custom only one instance only - 5

Input:

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

After it was rendered into DOM

Output:

Use getSubNode and css

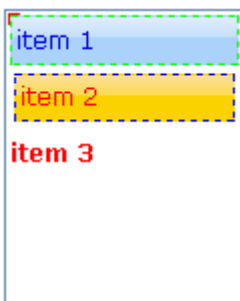## 7.4.6. Custom only one instance only - 6

Input:

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

linb.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:

item 1

item 2

item 3

7.4.7. Custom style for an UI Class

Input:

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

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

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

> All instances were changed

Output:

| sbutton7 | sbutton8 | sbutton9 | sbutton10 | sbutton11 |

7.4.8. Custom style for all UI Class - skin

There are three system skins in jsLinb3.0: default, vista and aqua. You can use linb.UI.setTheme to switch the skin. You can also add your own custom skin easily. Only two steps:
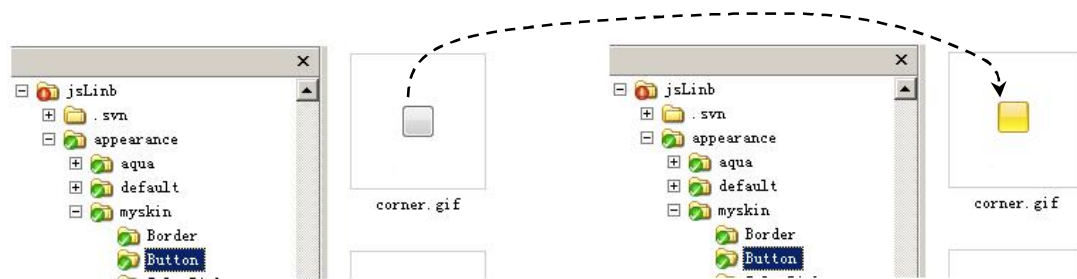
7.4.8.1. First: Copy

All skins are in "runtime/jsLinb/appearance", you can create an new folder (e.g. 'myskin'), and copy all directories and files in an existing skin folder to it.
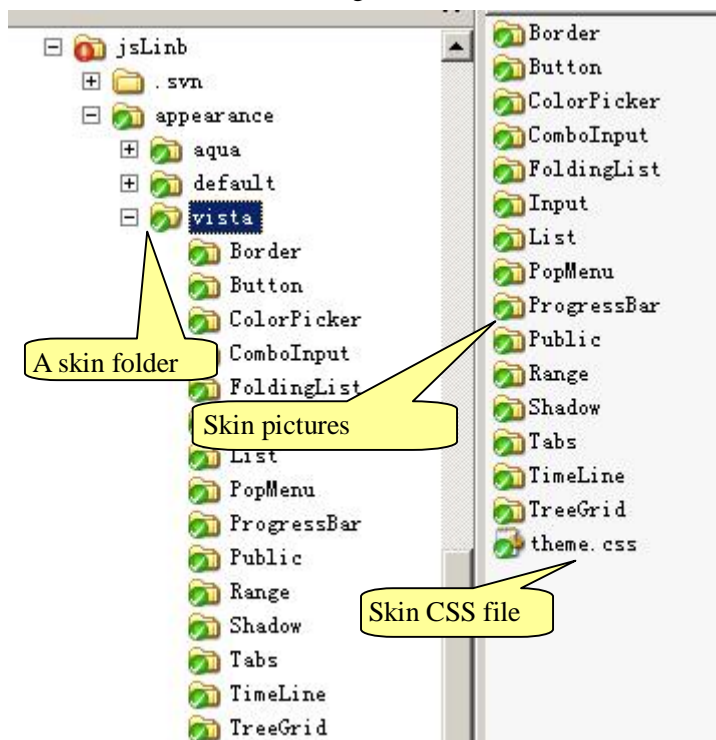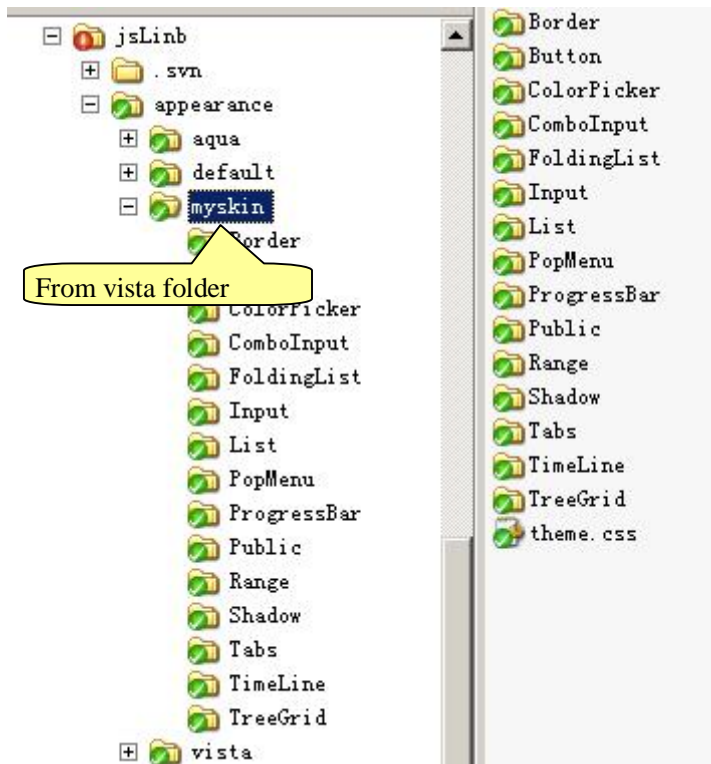
7.4.8.2.   Second: Little by little, modify pictures and CSS

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

After that,

Input:

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

Output:

The end

linb

2010-7-21