



jQuery EasyUI

官方 API 文档中文版

-- version 1.5.1 Build 1 --

作者: 王锦阳
CSDN 账号: richie696
更新日期: 2016-12-26

目录

jQuery EasyUI 官方 API 文档中文版	1
目录	2
前言 1: 汉化说明	10
v1.0.0 版汉化说明	10
v1.1.0 版汉化说明	10
v1.2.0 版汉化说明	11
v1.3.0 版汉化说明	12
v1.3.6 版汉化说明	12
v1.4.0 Build 1 版汉化说明	12
v1.4.1 Build 2 版汉化说明	13
v1.4.2 Build 1 版汉化说明	13
v1.4.3 Build 1 版汉化说明	13
v1.4.4 Build 1 版汉化说明	14
v1.4.5 Build 1 版汉化说明	14
v1.5.0 Build 1 版汉化说明	14
v1.5.1 Build 1 版汉化说明	15
前言 2: 更新说明	16
jQuery EasyUI 1.3.3 版本更新内容	16
jQuery EasyUI 1.3.4 版本更新内容	16
jQuery EasyUI 1.3.5 版本更新内容	17
jQuery EasyUI 1.3.6 版本更新内容	18
jQuery EasyUI 1.4.0 版本更新内容	19
jQuery EasyUI 1.4.1 版本更新内容	19
jQuery EasyUI 1.4.2 版本更新内容	20
jQuery EasyUI 1.4.3 版本更新内容	21
jQuery EasyUI 1.4.4 版本更新内容	21
jQuery EasyUI 1.4.5 版本更新内容	22
jQuery EasyUI 1.5.0 版本更新内容	22
jQuery EasyUI 1.5.1 版本更新内容	23
前言 3: 文档说明	24
属性解释	24
事件解释	24
方法解释	24
jQuery EasyUI 入门指南	25
第一章 Base (基础)	26
Parser(解析器)	27
使用案例	27
属性	27
事件	27
方法	27
EasyLoader (简单加载)	28
使用案例	28

属性	28
事件	29
方法	29
Draggable (拖动)	29
使用案例	30
属性	30
事件	31
方法	31
Droppable (放置)	31
使用案例	32
属性	32
事件	32
方法	32
Resizable (调整大小)	32
使用案例	33
属性	33
事件	33
方法	33
Pagination (分页)	34
依赖关系	35
使用案例	35
属性	35
事件	37
方法	38
SearchBox (搜索框)	38
依赖关系	39
使用案例	39
属性	40
方法	40
ProgressBar (进度条)	41
使用案例	42
属性	42
事件	42
方法	43
Tooltip (提示框)	43
使用案例	44
属性	44
事件	45
方法	45
Mobile (移动)	45
属性	45
方法	46
第二章 Layout (布局)	47
Panel (面板)	47
使用案例	48
属性	49

事件	52
方法	53
Tabs (选项卡)	54
依赖关系	55
使用案例	55
属性	56
事件	58
方法	59
选项卡面板	60
Accordion (分类选项卡)	61
依赖关系	62
使用案例	62
容器属性	63
面板属性	63
事件	63
方法	63
Layout (布局)	64
依赖关系	65
使用案例	65
布局属性	67
布局事件	67
区域面板属性	67
方法	68
第三章 Menu and Button (菜单和按钮)	70
Menu (菜单)	70
使用案例	71
属性	72
事件	73
方法	73
LinkButton (按钮)	75
使用案例	76
属性	76
事件	77
方法	77
MenuButton (菜单按钮)	78
依赖关系	79
属性	80
方法	80
SplitButton (分割按钮)	80
依赖关系	81
属性	82
方法	82
SwitchButton (开关按钮)	82
使用案例	83
属性	83
事件	83

方法	84
第四章 Form（表单）	85
Form（表单）	85
使用案例	86
属性	87
事件	88
方法	88
ValidateBox（验证框）	89
依赖关系	90
用法	90
验证规则	91
属性	91
事件	92
方法	92
TextBox（文本框）	93
依赖关系	94
使用案例	94
属性	94
事件	95
方法	96
PasswordBox（密码框）	97
依赖关系	97
使用案例	98
属性	98
事件	98
方法	98
Combo（自定义下拉框）	99
依赖关系	100
用法	100
属性	100
事件	101
方法	101
ComboBox（下拉列表框）	102
依赖关系	103
使用案例	103
属性	104
事件	107
方法	107
ComboTree（树形下拉框）	108
依赖关系	109
使用案例	109
属性	109
事件	110
方法	110
ComboGrid（数据表格下拉框）	111
依赖关系	112

使用案例	112
属性	113
事件	114
方法	114
ComboTreeGrid（树形表格下拉框）	115
依赖关系	116
使用案例	116
属性	117
事件	117
方法	117
TagBox（文本框）	118
依赖关系	119
使用案例	119
属性	119
事件	120
方法	120
NumberBox（数值输入框）	120
依赖关系	121
用法	121
属性	121
事件	122
方法	122
DateBox（日期输入框）	122
依赖关系	123
使用案例	123
属性	123
事件	125
方法	125
DateTimeBox（日期时间输入框）	126
依赖关系	127
使用案例	127
属性	127
方法	128
DateTimeSpinner（日期时间微调框）	128
依赖关系	129
使用案例	129
属性	129
事件	129
方法	130
Calendar（日历）	130
使用案例	131
属性	131
事件	133
方法	133
Spinner（微调）	133
依赖关系	134

使用案例	134
属性	134
事件	135
方法	135
NumberSpinner (数字微调)	136
依赖关系	136
使用案例	136
属性	136
事件	137
方法	137
TimeSpinner (时间微调)	138
依赖关系	138
使用案例	138
属性	138
事件	139
方法	140
Slider (滑动条)	141
依赖关系	141
使用案例	141
属性	141
事件	142
方法	143
FileBox (文件框)	144
依赖关系	144
使用案例	144
属性	144
事件	145
方法	145
第五章 Window (窗口)	146
Window (窗口)	147
依赖关系	147
使用案例	147
属性	148
事件	149
方法	149
Dialog (对话框窗口)	150
依赖关系	150
使用案例	150
属性	151
事件	153
方法	153
Messenger (消息窗口)	154
依赖关系	154
使用案例	154
属性	154
方法	154

第六章 DataGrid and Tree (表格和树)	157
DataGrid (数据表格)	158
依赖关系	158
使用案例	158
DataGrid 属性	159
列属性	163
编辑器	166
DataGrid 视图	166
事件	167
方法	169
DataList (数据列表)	173
依赖关系	173
使用案例	173
属性	174
事件	174
方法	174
PropertyGrid (属性表格)	175
依赖关系	175
使用案例	175
行数据	176
属性	176
方法	177
Tree (树)	178
依赖关系	178
使用案例	178
属性	181
事件	182
方法	184
TreeGrid (树形表格)	188
依赖关系	188
使用案例	188
属性	189
事件	190
方法	191
第七章 扩展 (Extension)	194
Portal (门户)	195
属性	196
事件	196
方法	196
DataGrid View (数据表格展示)	196
DataGrid DetailView (数据表格详细展示)	197
DataGrid GroupView (数据表格分组展示)	200
DataGrid BufferView (数据表格缓存视图)	202
DataGrid VirtualScrollView (数据表格虚拟滚动视图)	203
Editable DataGrid (可编辑数据表格)	205
使用案例	205

属性	206
事件	206
方法	207
Cell Editing in DataGrid (单元格编辑表格)	208
属性	209
事件	209
方法	209
Columns Extension for DataGrid (列扩展表格)	211
事件	211
方法	212
Editable Tree (可编辑树)	213
方法	214
DataGrid Filter Row (可过滤行的数据表格)	215
属性	215
事件	216
方法	216
Drag and Drop Rows in DataGrid (可拖放行的数据表格)	218
属性	219
事件	219
方法	220
Drag and Drop Rows in TreeGrid (可拖放的树形表格)	221
属性	222
事件	222
方法	222
PivotGrid (数据分析表格)	223
使用案例	223
属性	224
事件	225
方法	225
DWR Loader (DWR 装载器)	226
RTL suport for jQuery EasyUI (jQuery EasyUI RTL 支持)	228
Ribbon (Ribbon 界面)	229
属性	230
事件	230
方法	231

前言 1：汉化说明

v1.0.0 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：v1.0

EasyUI 版本：1.3.2

汉化时间：2013 年 02 月 15 日

本汉化文档系本人原创翻译制作，在过年期间花费了我数个日夜进行名词抽取、翻译、润色、译者注等工作，在文档中我对一些重要的方法、属性、事件等都做了详细的批注以及示例展示，这是英文原版中没有的内容。为了让大家更好的理解我对翻译完成后的文档进行了二次校验和润色。由于时间较紧，可能里面翻译的某些地方存在一些问题或错别字，如果你发现了问题可及时与我联系，E-mail:richie696@vip.qq.com QQ:115198807，欢迎大家能协助我一起完善该 API 文档，我也会对该文档持续进行维护和更新，如有任何疑问请到我的博客中提出 (<http://blog.sina.com.cn/richie696>)。英文原版 API 版权归版权人所有，汉化版版权归本人所有。未经本人许可请勿对本文档进行复制或再发行，谢谢合作！

v1.1.0 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：v1.1

EasyUI 版本：1.3.4

汉化时间：2013 年 09 月 24 日

未能兑现之前的承诺翻译出 1.3.3 的 API，实在是抱歉了。因为工作太忙实在无法抽身，马上要国庆了，现在也相对清闲一点，当我准备开始翻译的时候发现 1.3.4 版也出了，索性直接上 1.3.4 好了，从本版开始，我会将 EasyUI 每个版本更新的内容也放到 API 当中来，明确的告诉大家最新版的 EasyUI 到底更新了哪些东西，并且在 API 当中我也会特别注明哪些内容是新版本 API 新增的，凡是没有特别注明的都代表是 1.3.2 版之前就支持的内容。除此之外，中文 API 中将有如下约定，虽未在正文中提及，但是您得知道他们的含义：

1. API 中所有的“事件”和“方法”的参数中，如果其值为“none”，则代表没有参数；
2. 所有控件的事件参数里面“e”代表事件对象 event，你可以从中获取所有需要的内容。

v1.2.0 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：v1.2

EasyUI 版本：1.3.5

汉化时间：2013 年 12 月 18 日

个人博客：<http://blog.sina.com.cn/richie696>

EasyUI 又更新了，这次更新的内容还是蛮多的，新特性也有，Bug 修复也有，总体来说越来越完善了！在翻译最新的 API 的时候发现，官方的 API 和更新说明经常描述的很罗嗦，但意思却很简单，不知道是不是不同的人写的更新说明或者 API 解释，看着感觉有点误导人。翻译过来的中文 API 做了最通俗易懂的说明，另外就是很多朋友建议我加上例子，我的建议是大家去下载最新的包，然后解压开以后看 demo 文件夹，在里面有所有控件的例子（看下图），你需要的示例代码直接打开示例文件，里面一个字不少，我个人感觉没必要把这些东西再做到 API 里面，我只想保证最原汁原味的 API 文档，而不是掺杂了乱七八糟东西，然后把 API 文档文件做得老大老大的，实在没必要。

C:\Users\richie696\Desktop\jquery-easyui-1.3.5\demo				
名称	修改日期	类型	大小	
accordion	2013/12/10 15:20	文件夹		
calendar	2013/2/18 11:12	文件夹		
combo	2013/11/18 15:47	文件夹		
combobox	2013/12/10 15:12	文件夹		
combogrid	2013/11/12 11:14	文件夹		
combotree	2013/4/23 15:21	文件夹		
datagrid	2013/12/6 16:56	文件夹		
datebox	2013/12/11 16:44	文件夹		
datetimebox	2012/10/9 16:34	文件夹		
dialog	2012/10/10 10:49	文件夹		
draggable	2013/5/31 11:17	文件夹		
droppable	2012/10/13 14:54	文件夹		
easyloader	2013/10/31 15:12	文件夹		
form	2013/10/17 14:44	文件夹		
layout	2013/11/28 15:16	文件夹		
linkbutton	2013/12/6 21:53	文件夹		
menu	2013/11/12 11:33	文件夹		
menubutton	2013/12/10 9:00	文件夹		
messenger	2013/1/23 0:25	文件夹		
numberbox	2013/10/10 22:16	文件夹		
numberspinner	2012/10/11 11:39	文件夹		
pagination	2013/12/9 16:39	文件夹		
panel	2013/10/10 16:58	文件夹		
progressbar	2013/5/23 15:26	文件夹		
propertygrid	2013/11/26 11:29	文件夹		
resizable	2013/3/12 14:27	文件夹		
searchbox	2013/8/22 15:10	文件夹		
slider	2013/7/22 11:03	文件夹		
...		

所有例子都在easyui包的demo文件夹中，需要拷贝代码的时候直接右键编辑html文件，从里面拷贝出示例代码就可以了，官方写的示例非常全面，总有你能用得上的。

v1.3.0 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：v1.3

EasyUI 版本：1.3.6

汉化时间：2014 年 4 月 11 日

个人博客：<http://blog.sina.com.cn/richie696>

老规矩了，废话不多说，这次更新内容还是比较多的，特别是新增内容。估计等到 EasyUI 1.3.7 出来的时候肯定又是一堆 1.3.6 版本新增功能的 Bug 修复。因为统计了一下本次更新只修复了 2 个 Bug，剩下的 27 个变更内容都是新增内容，具体可参考更新说明里面。

v1.3.6 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.3.6

EasyUI 版本：1.3.6

汉化时间：2014 年 4 月 20 日

个人博客：<http://blog.sina.com.cn/richie696>

本次更新除了新增的 API 内容外，还增加了官方提供的所有扩展插件的中文 API 以及插件的下载地址给所有需要的同学。本中文 API 也将会不断的完善，在我业余时间允许的前提条件下，在后续版本中将会陆续加入各个控件的 Demo（非官方包中的，都是我本人日常项目开发中写的实战代码）敬请期待！

v1.4.0 Build 1 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.4 Build 1

EasyUI 版本：1.4

汉化时间：2014 年 8 月 6 日

个人博客：<http://blog.sina.com.cn/richie696>

时隔 4 个月之久，EasyUI 终于迎来大版本更新了，本次更新内容诸多，除了常规维护外，还新增了 3 个新组件，都很实用。详细的可以阅读更新说明，里面给了详细的解读。另外，从该版本开始我将会逐步的将 EasyUI 官方以及第三方较好的插件 API 整合到 API 文档当中，并且会对这些插件做一些简单的 Demo 实现，存放到配套提供的程序包 demo 文件夹下，以便大家学习和使用。本期文档中将官方提供的所

有附加插件的 API 都整理并存放到 Extension 节点下了，这些扩展的 demo 在附带的程序包中已经提供，可以用于参考使用。

v1.4.1 Build 2 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.4.1 Build 2

EasyUI 版本：1.4.1

汉化时间：2014 年 11 月 15 日

个人博客：<http://blog.sina.com.cn/richie696>

本次更新相对于 1.4 只是一些 BUG 修复和功能性修改（比如：增加了一些 API 接口），没有过多的新内容。所以也就不多废话了，具体更新内容请看更新说明章节。

v1.4.2 Build 1 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.4.2 Build 1

EasyUI 版本：1.4.2

汉化时间：2015 年 03 月 24 日

个人博客：<http://blog.sina.com.cn/richie696>

本次更新内容较多，并且首次加入了移动端开发框架，可谓相当给力，新东西自然意味着更多的 BUG 即将诞生，我个人预计在未来的 1~2 个版本里会有不少 BUG 修复的更新内容，今后 EasyUI 定会发力移动开发方向，所以也算是一个好的开头，虽然目前东西还不是很全，但是基本的也都够用了，希望 EasyUI 今后会越来越好吧！由于现在工作过于繁忙所以本次 API 更新延后了快 2 周，抱歉了！

v1.4.3 Build 1 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.4.3 Build 1

EasyUI 版本：1.4.3

汉化时间：2015 年 07 月 15 日

个人博客：<http://blog.sina.com.cn/richie696>

本次更新内容主要是 BUG 修复和性能优化，只有一个新增组件，相隔 4 个月又发布了新版本，如果目前正在用 1.4.2 版本的朋友可以升级一下，其他人请自行测试后再决定是否要升级。

v1.4.4 Build 1 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.4.4 Build 1

EasyUI 版本：1.4.4

汉化时间：2015 年 11 月 25 日

个人博客：<http://blog.sina.com.cn/richie696>

本次更新内容主要是 BUG 修复和功能改进，相隔 4 个月又发布了新版本，这次更新更像是例行公事，不过好在修复了很多 BUG，另外需要说一下，EasyUI 框架当中其实官方还隐藏了不少 API 没有开放出来，有些朋友建议我把整理一下，将一些好用的 API 及其用法也更新到中文 API 中，这里我想说的是，有些 API 或许是因为不稳定、尚有 BUG、未完全实现或者是未经过完整测试的，所以官方并未公布出来，因此我也不建议大家大面积的去使用，这样会带来很多不稳定因素，甚至是致命的 BUG，这也是我没有将这些 API 写入中文 API 文档的原因，所以有能力并且需要的人就自行去源代码中挖掘吧，我这里只同步官网的 API（外加少许的变动或者不影响稳定性和安全性的新增内容）。

v1.4.5 Build 1 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.4.5 Build 1

EasyUI 版本：1.4.5

汉化时间：2016 年 4 月 8 日

个人博客：<http://blog.sina.com.cn/richie696>

快半年了，EasyUI 又更新了。这次依然以 BUG 修复和功能改进为主，EasyUI 发展至今，主体功能已经基本完善。即便以后还有新组件，那也是在现有功能完善的基础上新增一些小组件了，类似 Datagrid 这类的大型复合组件应该比较少了，相比新组件我更期待官方能对 EasyUI 的性能优化多下下功夫。

v1.5.0 Build 1 版汉化说明

汉化人：王锦阳

汉化语言：简体中文

汉化版本：1.5 Build 1

《jQuery EasyUI 简体中文 API 文档》

EasyUI 版本: 1.5

汉化时间: 2016 年 8 月 8 日

个人博客: <http://blog.sina.com.cn/richie696>

例行更新, 不过本次有新组件加入, 感觉这次的组件早就应该有了, 居然到现在才加入进来, 不管怎么说有总比没有好。这次还是以改进为主, 改进项占了大多数。废话不多说具体内容大家看更新说明吧!

v1.5.1 Build 1 版汉化说明

汉化人: 王锦阳

汉化语言: 简体中文

汉化版本: 1.5.1 Build 1

EasyUI 版本: 1.5.1

汉化时间: 2016 年 12 月 26 日

个人博客: <http://blog.sina.com.cn/richie696>

最近比较忙, 抽空做了最新版的 API, 本次的主要精力就是放在了 pdf 版的文档上面, 废话就不多说了。更新内容自己看更新说明吧!

前言 2：更新说明

jQuery EasyUI 1.3.3 版本更新内容

Bug (修复)

- datagrid: 修复列 styler 函数不支持某些 CSS 样式的问题;
- datagrid: 修复 IE 浏览器对单个文档最多只能使用 31 个样式表限制的问题;
- treegrid: 修复列 styler 函数不支持某些 CSS 样式的问题;
- menu: 修复在 IE6 版本下自动列宽不正确的问题;
- combo: 修复在 combo 区域以外点击无法触发“onHidePanel (隐藏面板)”事件函数的问题;

Improvement (改进)

- datagrid: 添加 “scrollTo” 和 “highlightRow” 方法;
- treegrid: 允许 treegrid 从元素解析数据;
- combo: 添加 “selectOnNavigation” 和 “readonly” 属性;
- combobox: 添加 “loadFilter” 属性, 允许用户在加载数据之前将数据格式化为 combobox 空间所需的类型;
- tree: 添加 “onBeforeDrop” 回调事件;
- validatebox: 验证框的消息提示现在开始依赖 “tooltip” 组件, 并增加 “deltaX” 属性;
- numberbox: “filter” 属性用于设置允许按下哪些键;
- linkbutton: 按钮支持分组;
- layout: 为各个区域面板增加了 ‘minWidth’, ‘maxWidth’, ‘minHeight’, ‘maxHeight’ 和 ‘collapsible’ 属性。

New Plugins (新插件)

tooltip: 当鼠标移动到一个元素上时显示一个弹出消息框。

jQuery EasyUI 1.3.4 版本更新内容

Bug (修复)

- combobox: 修复在解析空的本地数据时也会触发 onLoadSuccess 事件的问题;
- form: 修复调用 “reset” 方法的时候无法重置 datebox 编辑器控件里面值的问题。

Improvement (改进)

- 移动设备: 移动设备支持右键菜单和双击功能;
- combobox: 新增的 “groupField” 和 “groupFormatter” 属性可以在组中显示指定的项目;
- tree: 在追加或者新增节点的时候, “data” 参数允许接收一个或多个节点的数据;

《jQuery EasyUI 简体中文 API 文档》

- tree: “getChecked” 方法允许接收一个或多个状态值参数 (‘checked’, ‘unchecked’, ‘indeterminate’) ;
- tree: 新增 “scrollTo” 方法;
- datagrid: 新增 “multiSort” 属性, 支持多列排序;
- datagrid: datagrid 的 “rowStyler” 属性和列的 “styler” 属性可以返回 CSS 类名或行内样式 (译者注: 以前不可以返回 CSS 类名) ;
- treegrid: 新增 “load” 方法用于加载数据并返回至首页;
- tabs: 新增 “tabWidth” 和 “tabHeight” 属性;
- validatebox: 新增 “novalidate” 属性来控制是否使用验证功能 (译者注: true: 关闭验证, false: 启用验证, 默认: false) ;
- validatebox: 新增 “enableValidation” 和 “disableValidation” 方法 (译者注: 用于在代码中控制开启和关闭验证的功能。) ;
- form: 新增 “enableValidation” 和 “disableValidation” 方法 (译者注: 用于在 form 提交时控制开启和关闭整个表单验证功能。) ;
- slider: 新增 “onComplete” 事件 (译者注: 在进度条加载完毕时触发的事件。) ;

pagination: “buttons” 属性允许使用已存在的元素 (译者注: 1.3.4 版之前 buttons 属性只接受一个数组形式的参数传入, 并且只能给按钮指定 iconCls 和 handler 属性, 改为 DOM 以后可以给按钮设置更多的属性及样式了, 使用方法同 datagrid 的 “toolbar” 属性, 即: buttons: ‘#btnDiv’) 。

jQuery EasyUI 1.3.5 版本更新内容

Bug (修复)

- searchbox: 修复 “searcher” 函数提供的 “name” 参数值错误的问题;
- combo: 修复 “isValid” 方法无法返回布尔值的问题;
- combo: 修复点击页面某一个 combo 组件的下拉列表时触发的 “onHidePanel” 事件会导致页面上其他 combo 组件的下拉列表被关闭的问题;
- combogrid: 修复某些从 combo 组件继承来的方法无法使用的问题。

Improvement (改进)

- datagrid: 改进检查行时候的性能;
- menu: 允许追加菜单分隔符;
- menu: 新增 “hideOnUnHover” 属性用于在鼠标离开菜单的时候指示是否需要隐藏菜单;
- slider: 新增 “clear” 和 “reset” 方法;
- tabs: 新增 “unselect” 方法、 “onUnselect” 事件;
- tabs: 新增 “selected” 属性, 用于指定的默认打开的面板;
- tabs: Tab Panel (Tab 页) 新增 “collapsible” 属性, 用于设置是否允许摺叠面板;
- tabs: 新增 “showHeader” 属性、 “showHeader” 方法和 “hideHeader” 方法;
- combobox: 允许 “disabled” 属性禁用下拉列表选项;
- tree: 改进数据加载时候的性能;
- pagination: 新增 “layout” 属性, 用于自定义控件的样式布局;
- accordion: 新增 “unselect” 方法、 “onUnselect” 事件;
- accordion: 新增 “select” 和 “multiple” 属性;

- accordion: 新增 “getSelections” 方法;
- datebox: 新增 “sharedCalendar” 属性, 允许多个 datebox 控件共享使用同一个 calendar 控件;
- datebox: 新增 “buttons” 属性, 用于自定义日历下方的按钮。
(译者注: 该点更新内容官方更新公告上没有注明, 具体内容和用法请看 datebox 的 API。)

jQuery EasyUI 1.3.6 版本更新内容

Bug (修复)

- treegrid: 修复 “getChecked” 方法不能正确的返回被选择的行的问题;
- tree: 修复在 “onlyLeafCheck” 属性为 true 时, 复选框无法在异步树种正确显示的问题。

Improvement (改进)

- treegrid: 所有的选择和选中的方法都扩展自 datagrid 组件;
- linkbutton: 添加图标对齐功能的完整支持, 可用值有: “top”、“bottom”、“left”、“right”;
- linkbutton: 添加 “size” 属性, 可用值有: “small”、“large”;
- linkbutton: 添加 “onClick” 事件;
- menubutton: 添加 “menuAlign” 属性, 该属性允许用户设置顶级菜单对齐;
- combo: 添加 “panelAlign” 属性, 可用值有: “left”、“right”;
- calendar: 添加 “formatter”、“styler” 和 “validator” 属性, 这些属性允许用于自定义日历日期;
- calendar: 添加 “onChange” 事件;
- panel: 添加 “method”、“queryParams” 和 “loader” 选项;
- panel: 添加 “onLoadError” 事件;
- datagrid: 添加 “onBeginEdit” 事件, 该事件在一个行进入编辑模式时触发;
- datagrid: 添加 “onEndEdit” 事件, 该事件在完成编辑但是编辑器尚未销毁之前触发;
- datagrid: 添加 “sort” 方法和 “onBeforeSortColumn” 事件;
- datagrid: 将 “combogrid” 编辑器集成到 datagrid 中;
- datagrid: 添加 “ctrlSelect” 属性, 在启用多行选择的时候允许使用 Ctrl 键+鼠标点击的方式进行多选操作;
- slider: 添加 “converter” 属性, 该属性允许用户决定如何将一个值转换为进度条位置或进度条位置值;
- searchbox: 添加 “disabled” 属性;
- searchbox: 添加 “disable”、“enable”、“clear”、“reset” 方法;
- spinner: 添加 “readonly” 属性、“readonly” 方法和 “onChange” 事件。

jQuery EasyUI 1.4.0 版本更新内容

Bug (修复)

- menu: 修复在删除一个菜单项的时候该菜单无法正确自适应高度的问题;
- datagrid: 修复在 datagrid 宽度太小的时候“fitColumns”方法无法正常工作的问题。

Improvement (改进)

- EasyUI 的所有组件已经支持非固定/百分比大小的尺寸设置;
- menu: 添加“showItem”、“hideItem”和“resize”方法;
- menu: 基于窗体大小自动调整高度;
- menu: 添加“duration”属性, 该属性允许用户自定义隐藏菜单动画的持续时间, 以毫秒为单位;
- validatebox: 添加“onBeforeValidate”和“onValidate”事件;
- combo: 从该版本开始 combo 组件扩展自 textbox 组件 (textbox 是 1.4 中新增的组件);
- combo: 添加“panelMinWidth”、“panelMaxWidth”、“panelMinHeight”和“panelMaxHeight”属性;
- searchbox: 从该版本开始 searchbox 组件扩展自 textbox 组件 (textbox 是 1.4 中新增的组件);
- tree: 添加“getRoot”方法, 用于返回通过“nodeEl”参数指定的节点的顶部父节点元素 (注意: 官网的英文 API 中该函数的说明有误, 其说明是 none (无参数), 实际这里是需要参数的);
- tree: 添加“queryParams”属性;
- datetimebox: 添加“spinnerWidth”属性;
- panel: 添加“doLayout”方法, 用于控制面板内组件的大小;
- panel: 添加“clear”方法, 用于清除面板内的内容;
- datagrid: 允许用户设置百分比宽度的列 (该功能真是千呼万唤始出来啊!);
- form: 添加“ajax”, “novalidate”和“queryParams”属性;
- linkbutton: 添加“resize”方法。

New Plugin (新组件)

- textbox: 该组件是一个增强的输入字段, 它可以让用户非常简单的构建一个表单;
- datetimespinner: 该组件是一个日期和时间的微调组件, 它允许我们选择一个特定的日期或时间;
- filebox: filebox 该组件表单元素中用于上传文件的文件框工具组件。

jQuery EasyUI 1.4.1 版本更新内容

Bug (修复)

- combogrid: 修复 combogrid 组件和其他 combo 组件高度不一致的问题;
- datagrid: 修复在 datagrid 行元素调用“updateRow”方法的时候丢失某些类样式的问题;

- menubutton: 修复在被禁用的按钮上使用“enable”方法无效的问题;
- form: 修复在 form 组件中调用“clear”方法以后导致 firebox 组件失效的问题。

Improvement (改进)

- tabs: “update”方法增加“type”参数, 允许用户更新表头、表体或整个 tab 控件;
- panel: 添加“openAnimation”、“openDuration”、“closeAnimation”和“closeDuration”属性用来设置面板打开和关闭时的动画效果;
- panel: 添加“footer”属性用来定义在页脚展示的页脚栏;
- datagrid: 调用“endEdit”方法可正确获取编辑值(这应该是一个已有功能的改进, 具体内容不得而知);
- datagrid: 添加“onBeforeSelect”、“onBeforeCheck”、“onBeforeUnselect”和“onBeforeUncheck”事件;
- propertygrid: 允许用户使用“beginEdit”方法进行行编辑;
- datebox: 添加“cloneFrom”方法来快速创建“datebox”组件;
- datetimesbox: 添加“cloneFrom”方法来快速创建“datetimesbox”组件。

jQuery EasyUI 1.4.2 版本更新内容

Bug (修复)

- treegrid: 修复重建 treegrid 之后列会恢复原始大小的问题。

Improvement (改进)

- draggable: 添加“delay”属性, 允许用户延迟拖动操作;
- tree: 添加“filter”属性和“doFilter”方法;
- tabs: “add”方法允许用户在指定的索引位上插入选项卡面板;
- tabs: 用户可以决定哪些选项卡面板可以被选择;
- tabs: 添加“justified”, “narrow”和“pill”属性;
- layout: 添加“unsplit”和“split”方法;
- messenger: 支持键盘导航功能;
- form: 添加“onChange”事件;
- combobox: 添加“queryParams”属性;
- slider: 添加“range”属性;
- menu: 添加“itemHeight”, “inline”, “noline”和“align”属性;
- panel: 添加“header”属性, 允许用户自定义面板标题栏;
- menubutton: 添加“hasDownArrow”属性。

New Plugin (新组件)

- datalist: 该组件是展示列表数据的组件, 用户可以对列表数据进行分组、单选、多选等各种操作;
- navpanel: 该组件是移动端框架的根组件。

jQuery EasyUI 1.4.3 版本更新内容

Bug (修复)

- textbox: 修复 “setText” 方法不接受值为 0 的问题;
- timespinner: 修复在使用 IE11 时点击空文本框时出错的问题;
- tabs: 修复 “update” 方法只能更新面板正文的问题。

Improvement (改进)

- combobox: 提升显示下拉框的性能;
- combogrid: 在下拉数据表格跳转到其它页面的时候记住显示的文本;
- combogrid: “setValue” 和 “setValues” 方法接受一个键值对象;
- window: 内联窗体的遮罩层可以自动伸展来填补父容器;
- tabs: “showTool” 和 “hideTool” 方法提供用户显示或隐藏工具栏;
- layout: 允许用户覆盖 “cls”、“headerCls” 和 “bodyCls” 属性值。

New Plugin (新组件)

switchbutton: 新增开关按钮组件, 状态: “开” 和 “关”。

jQuery EasyUI 1.4.4 版本更新内容

Bug (修复)

- filebox: 修复 “clear” 和 “reset” 方法在 IE9 下无法正常工作的问题;
- messenger: 修复调用无参的 \$.messenger.progress() 方法之后, 再调用 \$.messenger.progress('close') 方法时无法正常工作的问题;
- timespinner: 修复在 IE8 中点击微调按钮时无法正确显示值的问题;
- window: 修复在 “onMove” 事件中调用 “options” 方法时无法正常显示的问题;
- treegrid: 修复 “getLevel” 方法无法接受为 0 的参数值的问题。

Improvement (改进)

- layout: 改进后的 “collapsedContent”、“expandMode” 和 “hideExpandTool” 属性可以支持区域面板;
- layout: 改进后的 “hideCollapsedContent” 属性可以在折叠面板上设置显示垂直标题栏;
- layout: 新增 “onCollapse”、“onExpand”、“onAdd”、“onRemove” 事件;
- datagrid: 在排序列的标题上显示 ↑ ↓ 图标;
- datagrid: 新增 “gotoPage” 方法;
- propertygrid: 新增 “groups” 方法, 以允许获取所有数据组;

- messenger: 在显示长消息的时候支持对消息进行自动滚动;
- tabs: “disabled” 属性支持定义一个被禁用的选项卡面板;
- tabs: 支持百分比大小。

jQuery EasyUI 1.4.5 版本更新内容

Bug (修复)

- datagrid: 修复在调用 updateRow 方法之后使用 getChanges 方法无法返回被更新的行的 BUG;
- treegrid: 修复在追加或插入新行的时候触发 onLoadSuccess 事件的 BUG;
- tree: 修复在追加或插入新节点的时候触发 onLoadSuccess 事件的 BUG。

Improvement (改进)

- window: 可以自定义显示样式了;
- window: 新增 “border” 属性允许用户设置不同的边框样式;
- navpanel: 新增 “href” 属性用以从远程服务器加载显示内容;
- combotree: “setValue” 和 “setValues” 方法增加 “id” 和 “text” 形参;
- combobox: 新增 “showItemIcon” 属性;
- combobox: 在 “groupPosition” 属性值设置为 “sticky” 时, 将会将选项分组标签固顶在下拉栏中;
- messenger: 当敲击回车键时将默认触发消息框的第一个按钮;
- validatebox: 新增 “editable”、“disabled”、“readonly”、“validateOnCreate” 和 “validateOnBlur” 属性;
- validatebox: 新增 “enable”、“disable”、“readonly” 和 “resetValidation” 方法;
- validatebox: 允许用户来决定如何显示错误消息;
- filebox: 新增 “accept” 和 “multiple” 属性;
- treegrid: 新增复选框的选择;
- treegrid: 新增 “getCheckedNodes”、“checkNode” 和 “uncheckNode” 方法;
- form: 新增 “iframe” 属性;
- form: 新增 “onProgress” 事件;
- form: 新增 “resetValidation” 方法。

jQuery EasyUI 1.5.0 版本更新内容

Bug (修复)

- combobox: 修复在加载包含所选项数据的时候不会触发 “onSelect” 事件的 BUG;
- datagrid: 修复在字段设置为一个空值的时候导致在某些情况下 “updateRow” 方法无法正常工作的 BUG。

Improvement (改进)

- 一个 label 标签可以被关联到任意表单的字段上；
- combobox: 改进在下拉项中 “select” 和 “unselect” 的规则；
- combobox: 添加 “limitToList” 属性来限制只能输入在列表项中的内容；
- combogrid: 允许用户快速克隆组件；
- form: 添加 “dirty” 属性，允许用户只发送变更的字段内容；
- form: 添加 “resetDirty” 方法；
- datagrid: 允许用户在没有数据的时候显示一条消息（比如：无记录）；
- textbox: 添加 “label”、“labelWidth”、“labelPosition” 和 “labelAlign” 属性；
- spinner: 添加 “spinAlign” 属性；
- calendar: 允许用户在日历组件上显示周数（今年的第几周）；
- window: 添加 “constrain” 属性。

New Plugin (新组件)

- passwordbox: 该插件允许用户在具有更好交互功能的输入框中输入密码；
- combotreegrid: 该插件结合了 combobox 和 treegrid 组件。

jQuery EasyUI 1.5.1 版本更新内容

Bug (修复)

- datagrid: 修复在调用 “updateRow” 方法之后选中和复选行标志丢失的问题；
- tabs: 修复在调用 “update” 方法的时候导致标签栏工具错位的问题；
- window: 修复在窗体高度设置为 “auto” 时，当移动窗体后窗体会丢失的问题；
- messenger: 修复在现实进度消息窗口后立即关闭该窗口会导致程序发生异常的问题；
- form: 修复 “clear” 方法无法清除 combobox 组件选择的下拉项的问题。

Improvement (改进)

- textbox: 可以用 “cls” 属性添加自定义样式；
- numberbox: 允许用户使用意大利货币格式；
- combo: 添加 “multivalue” 属性，允许用户决定如何提交多个值；
- combobox: 添加 “reversed” 属性；
- combobox: 添加 “onClick” 事件；
- combogrid: 添加 “reversed” 属性；
- treegrid: 使用 Shift 键启用多值选择。

New Plugin (新组件)

- tagbox: 允许用户在表单字段上添加标签。

前言 3：文档说明

EasyUI 每个组件的属性，方法和事件。用户可以很容易地扩展他们。

属性解释

所有的属性都定义在 `jQuery.fn. {plugin}.defaults` 里面。例如，对话框属性定义在 `jQuery.fn. dialog.defaults` 里面。

事件解释

所有的事件（回调函数）也都定义在 `jQuery.fn. {plugin}.defaults` 里面。

方法解释

调用方法的语法：`$('selector').plugin('method' , parameter);`

解释：

- *selector* 是 jQuery 对象选择器。
- *plugin* 是插件的名称。
- *method* 是相应插件现有的方法。
- *parameter* 是参数对象，可以是一个对象、字符串等。

所有方法都定义在 `jQuery.fn. {plugin}.methods`。每个方法都有 2 个参数：`jq` 和 `param`。第一个参数 '`jq`' 是必须的，这是指的 jQuery 对象。第二个参数 '`param`' 是指传入方法的实际参数。例如，为 `dialog` 组件扩展一个方法名为 '`mymove`'，代码如下：

```
1. $.extend($.fn.dialog.methods, {  
2.     mymove: function(jq, newposition){  
3.         return jq.each(function() {  
4.             $(this).dialog('move', newposition);  
5.         });  
6.     }  
7. });
```


现在你可以调用 'mymove' 方法将对话框移动到指定位置：

```
1. $('#dd').dialog('mymove', {
2.     left: 200,
3.     top: 100
4. });
```

jQuery EasyUI 入门指南

下载程序库并导入 EasyUI 的 CSS 和 Javascript 文件到您的页面。

```
1. <link rel="stylesheet" type="text/css" href="easyui/themes/default/easyui.css">
2. <link rel="stylesheet" type="text/css" href="easyui/themes/icon.css">
3. <script type="text/javascript" src="easyui/jquery-1.7.2.min.js"></script>
4. <script type="text/javascript" src="easyui/jquery.easyui.min.js"></script>
```

一旦你导入了 EasyUI 必须的文件，你就可以通过标记或 Javascript 定义一个 EasyUI 组件。例如：定义一个带折叠功能的面板，你需要写的 HTML 代码如下：

```
1. <div id="p" class="easyui-panel" style="width:500px;height:200px;padding:10px;"
2.     title="My Panel" iconCls="icon-save" collapsible="true">
3.     The panel content
4. </div>
```

当通过标记创建一个组件的时候，从 EasyUI 1.3 版开始可以用 HTML5 标准的 'data-options' 属性来改写上面的代码为：

```
1. <div id="p" class="easyui-panel" style="width:500px;height:200px;padding:10px;"
2.     title="My Panel" data-options="iconCls:'icon-save',collapsible:true">
3.     The panel content
4. </div>
```

下面的代码演示了如何创建一个组合框，并绑定 onSelect 事件。

```
1. <input class="easyui-combobox" name="language"
2.     data-options="
3.         url:'combobox_data.json',
4.         valueField:'id',
5.         textField:'text',
6.         panelHeight:'auto',
7.         onSelect:function(record) {
8.             alert(record.text)
5.         }" />
```

第一章

Base

(基础)

Parser (解析器)

使用案例

1. `$.parser.parse();` // 解析所有页面
2. `$.parser.parse('#cc');` // 解析指定节点

属性

属性名	类型	描述	默认值
<code>\$.parser.auto</code>	boolean	定义是否自动解析 EasyUI 组件。	true

事件

事件名	事件参数	描述
<code>\$.parser.onComplete</code>	context	在解析器完成解析操作的时候触发。

方法

方法名	方法参数	描述
<code>\$.parser.parse</code>	context	解析 EasyUI 组件。

EasyLoader（简单加载）

使用案例

加载 EasyUI 模块

```
1. easyloader.base = '../'; // 设置 easyui 基础目录
2. easyloader.load('messenger', function() { // 加载指定模块
3. $.messenger.alert('Title', 'load ok');
4. });
```

加载来自绝对路径的脚本

```
1. using('http://code.jquery.com/jquery-1.4.4.min.js', function() {
2. // ...
3. });
```

加载来自相对路径的脚本

```
1. // 脚本路径相对于 easyui 目录
2. using('./myscript.js', function() {
3. // ...
4. });
```

属性

属性名	类型	描述	默认值
modules	object	预定义模块。	
locales	object	预定义区域。	
base	string	easyui 基础目录，必须用 '/' 结束。	基本目录被预设为相对路径下的 easyload.js 文件
theme	string	主题的名称预定义在 themes 目录下。	default
css	boolean	定义在加载模块的时候加载 CSS 文件。	true
locale	string	区域名称	null
timeout	number	超时的值以毫秒为单位，载入如果超时则重载。	2000

预定义区域

- af // 阿富汗
- am // 亚美尼亚
- ar // 阿根廷
- bg // 保加利亚语
- ca // 加拿大
- cs // 捷克语
- cz // 捷克语（捷克共和国）
- da // 丹麦语

- de // 德语
- el // 希腊
- en // 英语
- es // 西班牙语
- fr // 法语
- it // 意大利
- jp // 日本
- ko // 韩国
- nl // 荷兰
- pl // 波兰
- pt // 葡萄牙
- ru // 俄罗斯
- sv // 萨尔瓦多
- tr // 土耳其
- zh_CN // 简体中文
- zh_TW // 繁体中文

事件

事件名	事件参数	描述
onProgress	name	当一个模块加载成功的时候触发。
onLoad	name	当一个模块以及他的依赖加载成功的时候触发。

方法

方法名	方法参数	描述
load	module, callback	加载指定模块。当加载成功的回调函数被调用。 模块参数有效的类型包括： <ul style="list-style-type: none"> ● 一个单一的模块名称 ● 模块数组 ● “.css”后缀结尾的 CSS 文件 ● “.js”后缀结尾的 JS 文件

Draggable（拖动）

使用\$.fn.draggable.defaults 重写默认值对象。

使用案例

通过标签创建一个可拖动的元素。

```
1. <div id="dd" class="easyui-draggable" data-options="handle:'#title'" style="width:100px; height:100px;">
2.     <div id="title" style="background:#ccc;">title</div>
3. </div>
```

使用 Javascript 创建一个可拖动的元素。

```
1. <div id="dd" style="width:100px; height:100px;">
2. <div id="title" style="background:#ccc;">title</div>
3. </div>
1. $('#dd').draggable({
2.     handle:'#title'
3. });
```

属性

属性名	类型	描述	默认值
proxy	string, function	<p>在拖动的时候使用的代理元素，当使用 'clone' 的时候，将使用该元素的一个复制元素来作为替代元素。如果指定了一个函数，它将返回一个 jquery 对象。</p> <p>下面的例子显示了如何创建一个简单的代理对象。</p> <pre>\$('.dragitem').draggable({ proxy: function(source) { var p = \$('<div style="border:1px solid #ccc; width:80px"></div>'); p.html(\$(source).html()).appendTo('body'); return p; } });</pre>	null
revert	boolean	如果设置为 true，在拖动停止时元素将返回起始位置。	false
cursor	string	拖动时的 CSS 指针样式。	move
deltaX	number	被拖动的元素对应于当前光标位置 x。	null
deltaY	number	被拖动的元素对应于当前光标位置 y。	null
handle	selector	开始拖动的句柄。	null
disabled	boolean	如果设置为 true，则停止拖动。	false
edge	number	可以在其中拖动的容器的宽度。	0
axis	string	定义元素移动的轴向，可用值有：'v' 或 'h'，当没有设置或设置	null

		为 null 时可同时在水平和垂直方向上拖动。	
delay	number	定义元素在多少毫秒后开始移动。 (该属性自 1.4.2 版开始可用)	100

事件

事件名	参数	描述
onBeforeDrag	e	在拖动之前触发，返回 false 将取消拖动。
onStartDrag	e	在目标对象开始被拖动时触发。
onDrag	e	在拖动过程中触发，当不能再拖动时返回 false。
onStopDrag	e	在拖动停止时触发。

方法

方法名	参数	描述
options	none	返回属性对象。
proxy	none	如果代理属性被设置则返回该拖动代理元素。
enable	none	允许拖动。
disable	none	禁止拖动。

Draggable (放置)

使用\$.fn.draggable.defaults 重写默认值对象。

使用案例

通过标签创建一个放置区。

```
1. <div id="dd" class="easyui-droppable" data-options="accept:'#d1,#d3'" style="width:100px;height:100px;"></div>
```

使用 Javascript 创建一个放置区。

```
1. <div id="dd" style="width:100px;height:100px;"></div>
1. $(' #dd').droppable({
2.   accept:'#d1,#d3'
3. });
```

属性

属性名	类型	描述	默认值
accept	selector	确定哪些可拖拽元素将被接受。	null
disabled	boolean	如果为 true，则禁止放置。	false

事件

事件名	事件参数	描述
onDragEnter	e, source	在被拖拽元素到放置区内的时候触发，source 参数表示被拖拽的 DOM 元素。
onDragOver	e, source	在被拖拽元素经过放置区的时候触发，source 参数表示被拖拽的 DOM 元素。
onDragLeave	e, source	在被拖拽元素离开放置区的时候触发，source 参数表示被拖拽的 DOM 元素。
onDrop	e, source	在被拖拽元素放入到放置区的时候触发，source 参数表示被拖拽的 DOM 元素。

方法

方法名	方法参数	描述
options	none	返回属性对象。
enable	none	启用放置功能。
disable	none	禁用放置功能。

Resizable (调整大小)

使用\$.fn.resizable.defaults 重写默认值对象。

使用案例

使用标签创建可变大小的窗口。

```
1. <div id="rr" class="easyui-resizable" data-options="maxWidth:800,maxHeight:600" style="width:100px;height:100px;border:1px solid #ccc;"></div>
```

使用 Javascript 创建可变大小的窗口。

```
1. <div id="rr" style="width:100px;height:100px;border:1px solid #ccc;"></div>
1. $('#rr').resizable({
2.   maxWidth:800,
3.   maxHeight:600
4. });
```

属性

属性名	类型	描述	默认值
disabled	boolean	如果为 true，则禁用大小调整。	false
handles	string	声明调整方位，'n' = 北，'e' = 东，'s' = 南等。	n, e, s, w, ne, se, sw, nw, all
minWidth	number	当调整大小时候的最小宽度。	10
minHeight	number	当调整大小时候的最小高度。	10
maxWidth	number	当调整大小时候的最大宽度。	10000
maxHeight	number	当调整大小时候的最大高度。	10000
edge	number	边框边缘大小。	5

事件

事件名	事件参数	描述
onStartResize	e	在开始改变大小的时候触发。
onResize	e	在调整大小期间触发。当返回 false 的时候，不会实际改变 DOM 元素大小。
onStopResize	e	在停止改变大小的时候触发。

方法

方法名	方法参数	描述
options	none	返回调整大小属性。
enable	none	启用调整大小功能。
disable	none	禁用调整大小功能。

Pagination (分页)

使用 `$.fn.pagination.defaults` 重写默认值对象

该分页控件允许用户导航页面的数据。它支持页面导航和页面长度选择的选项设置。用户可以在分页控件上添加自定义按钮，以增强其功能。

依赖关系

- linkbutton

使用案例

使用标签创建分页控件。

```
1. <div id="pp" class="easyui-pagination" data-options="total:2000, pageSize:10" style="background:#efefef;border:1px solid #ccc;"></div>
```

使用 Javascript 创建分页控件。

```
1. <div id="pp" style="background:#efefef;border:1px solid #ccc;"></div>
1. $('#pp').pagination({
2.   total:2000,
3.   pageSize:10
4. });
```

让我们使用面板和分页插件来创建一个 ajax 分页。当用户选择一个新页面的时候，面板将显示指定页面的内容。

```
1. <div id="content" class="easyui-panel" style="height:200px"
2.   data-options="href:'show_content.php?page=1'">
3. </div>
4. <div class="easyui-pagination" style="border:1px solid #ccc;"
5.   data-options="
6.     ).panel('refresh', 'show_content.php?page=' + pageNumber);
7.   ">
8. </div>
```

面板上默认显示第一页的内容。当用户导航页面的时候，'onSelectPage' 事件将被触发，将会根据一个新的 URL 参数获取对应页面的新内容，并通过 'refresh' 方法将内容刷新到内容面板上。

属性

属性名	类型	描述	默认值
total	number	总记录数，在分页控件创建时初始的值。	1
pageSize	number	页面大小。	10
pageNumber	number	在分页控件创建的时候显示的页数。	1

pageList	array	<p>用户可以改变页面大小。pageList 属性定义了页面导航显示的页码。</p> <p>代码示例：</p> <pre>\$('#pp').pagination({ pageList: [10, 20, 50, 100] });</pre>	[10, 20, 30, 50]
loading	boolean	定义数据是否正在载入。	false
buttons	array	<p>自定义按钮，可用值有：</p> <p>①. 每个按钮都有 2 个属性： iconCls: 显示背景图片的 CSS 类 ID handler: 当按钮被点击时调用的一个句柄函数。</p> <p>②. 页面已存在元素的选择器对象 (例如: buttons: '#btnDiv') (该属性值自 1.3.4 版开始可用)</p> <p>自定义按钮可以通过标签创建：</p> <pre><div class="easyui-pagination" style="border:1px solid #ccc" data-options="total: 114, buttons: [{ iconCls:'icon-add', handler:function() {alert(' add')}}],'-',{ iconCls:'icon-save', handler:function() {alert(' save')}}]"]> </div></pre> <p>自定义按钮也可以通过 Javascript 创建：</p> <pre>\$('#pp').pagination({ total: 114, buttons: [{ iconCls:'icon-add', handler:function() {alert(' add')}}],'-',{ iconCls:'icon-save', handler:function() {alert(' save')}}] });</pre>	null
layout	array	<p>分页控件布局定义。(该属性值自 1.3.5 版开始可用)</p> <p>布局选项可以包含一个或多个值：</p> <ol style="list-style-type: none"> 1) list: 页面显示条数列表。 2) sep: 页面按钮分割线。 3) first: 首页按钮。 4) prev: 上一页按钮。 	

		5) next: 下一页按钮。 6) last: 尾页按钮。 7) refresh: 刷新按钮。 8) manual: 手工输入当前页的输入框。 9) links: 页面数链接。 示例代码: <pre>\$('#pp').pagination({ layout: ['first', 'links', 'last'] });</pre>	
links	number	该链接数仅在“links”项包含在“layout”中的时候有效。 (该属性值自 1.3.5 版开始可用)	10
showPageList	boolean	定义是否显示页面导航列表。	true
showRefresh	boolean	定义是否显示刷新按钮。	true
beforePageText	string	在输入组件之前显示一个 label 标签。	Page
afterPageText	string	在输入组件之后显示一个 label 标签。	of {pages}
displayMsg	string	显示页面信息。	Displaying {from} to {to} of {total} items

事件

事件名	事件参数	描述
onSelectPage	pageNumber , pageSize	用户选择一个新页面的时候触发。回调函数包含 2 个参数: pageNumber: 新的页数。 pageSize: 页面大小 (每页显示的条数)。 代码示例: <pre>\$('#pp').pagination({ onSelectPage: function (pageNumber, pageSize) { \$(this).pagination('loading'); alert('pageNumber:' + pageNumber + ', pageSize:' + pageSize); \$(this).pagination('loaded'); } });</pre>
onBeforeRefresh	pageNumber , pageSize	在点击刷新按钮刷新之前触发, 返回 false 可以取消刷新动作。

onRefresh	pageNumber , pageSize	刷新之后触发。
onChangePageSize	pageSize	在页面更改页面大小的时候触发。

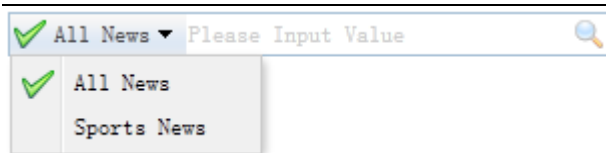
方法

方法名	方法参数	描述
options	none	返回参数对象。
loading	none	提醒分页控件正在加载中。
loaded	none	提醒分页控件加载完成。
refresh	options	刷新并显示分页栏信息。 (该方法自 1.3 版开始可用) 代码示例： <pre>\$('#pp').pagination('refresh'); // 刷新分页栏信息 \$('#pp').pagination('refresh',{ // 改变选项并刷新分页栏信息 total: 114, pageNumber: 6 });</pre>
select	page	选择一个新页面，页面索引从 1 开始。 (该方法自 1.3 版开始可用) 代码示例： <pre>\$('#pp').pagination('select'); // 刷新当前页 \$('#pp').pagination('select', 2); // 选择第二页</pre>

SearchBox（搜索框）

使用\$.fn.searchbox.defaults 重写默认值对象。

搜索框提示用户需要输入搜索的值。它可以结合一个菜单，允许用户选择不同的搜索类别。在用户按下回车键或点击组件右边的搜索按钮的时候会执行搜索操作。



依赖关系

- textbox
- menubutton

使用案例

创建查询框

1. 使用标签创建。添加 'easyui-searchbox' 类 ID 到 <input/> 标签。

```

1. <script type="text/javascript">
2.   function qq(value,name) {
3.     alert(value+"："+name)
4.   }
5. </script>
6.
7. <input id="ss" class="easyui-searchbox" style="width:300px"
8.   data-options="searcher:qq,prompt:'Please Input Value',menu:'#mm'"></input>
9.
10. <div id="mm" style="width:120px">
11.   <div data-options="name:'all',iconCls:'icon-ok'">All News</div>
12.   <div data-options="name:'sports'">Sports News</div>
13. </div>

```

2. 创建程序。

```

1. <input id="ss"></input>
2. <div id="mm" style="width:120px">
3.   <div data-options="name:'all',iconCls:'icon-ok'">All News</div>
4.   <div data-options="name:'sports'">Sports News</div>
5. </div>
6. $(' #ss').searchbox({
7.   searcher:function(value,name) {
8.     alert(value + "，" + name)
9.   },
10.   menu:'#mm',
11.   prompt:'请输入值'
12. });

```

属性

属性名	类型	描述	默认值
width	number	组件宽度。	auto
height	number	组件高度。 (该属性自 1.3.2 版开始可用)	22
prompt	string	在输入框中显示提示消息。	''
value	string	输入的值。	''
menu	selector	<p>搜索类型菜单。每个菜单项都具备一下属性：</p> <p>name: 搜索类型名称。</p> <p>selected: 自定义默认选中的搜索类型名称。</p> <p>如下示例定义了一个选择搜索类型名称的搜索框：</p> <pre><input class="easyui-searchbox" style="width:300px" data-options="menu:'#mm'" /> <div id="mm" style="width:150px"> <div data-options="name:'item1'">Search Item1</div> <div data- options="name:'item2',selected:true">Search Item2</div> <div data-options="name:'item3'">Search Item3</div> </div></pre>	null
searcher	function(value, name)	在用户按下搜索按钮或回车键的时候调用 searcher 函数。	null
disabled	boolean	定义是否禁用搜索框。 (该属性自 1.3.6 版开始可用)	false

方法

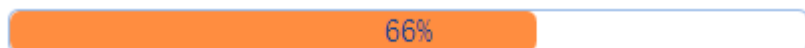
方法名	方法参数	描述
options	none	返回属性对象。
menu	none	<p>返回搜索类型菜单对象。下面的例子显示了更改菜单项图标。</p> <pre>var m = \$('#ss').searchbox('menu'); // 获取菜单项 var item = m.menu('findItem', 'Sports News'); // 查找菜单项 // 更改菜单项图标 m.menu('setIcon', { target: item.target, iconCls: 'icon-save'</pre>

		}); // 选择搜索类型名 \$('#ss').searchbox('selectName', 'sports');
textbox	none	返回文本框对象。
getValue	none	返回当前搜索值。
setValue	value	设置一个新的搜索值。
getName	none	返回当前搜索类型名。
selectName	name	选择当前搜索类型名。 代码示例： \$('#ss').searchbox('selectName', 'sports');
destroy	none	销毁该控件。
resize	width	重置组件宽度。
disable	none	禁用搜索框。（该方法自 1.3.6 版开始可用）
enable	none	启用搜索框。（该方法自 1.3.6 版开始可用）
clear	none	清除搜索框。（该方法自 1.3.6 版开始可用）
reset	none	重置搜索框。（该方法自 1.3.6 版开始可用）

ProgressBar（进度条）

使用\$.fn.progressbar.defaults 重写默认值对象。

进度条提供了一个反馈显示一个长时间运行的操作进展。可以更新的进展条，让用户知道当前正在执行操作。



使用案例

创建进度条

使用 HTML 标签或程序创建进度条组件。从标签创建更加简单，添加 'easyui-progressbar' 类 ID 到 <div/> 标签。

```
1. <div id="p" class="easyui-progressbar" data-options="value:60" style="width:400px;"></div>
```

使用 Javascript 创建进度条。

```
1. <div id="p" style="width:400px;"></div>
1. $('#p').progressbar({
2.     value: 60
3. });
```

获取值和设置值

获取当前值和设置一个新的值到进度条控件。

```
1. var value = $('#p').progressbar('getValue');
2. if (value < 100) {
3.     value += Math.floor(Math.random() * 10);
4.     $('#p').progressbar('setValue', value);
5. }
```

属性

属性名	类型	描述	默认值
width	string	设置进度条宽度。	auto
height	number	设置进度条高度。(该属性自 1.3.2 版开始可用)	22
value	number	进度值	0
text	string	在组件上显示的进度值模板	{value}%

事件

事件名	事件参数	描述
onChange	newValue, oldValue	<p>在值更改的时候触发。</p> <p>代码示例：</p> <pre>\$('#p').progressbar({ onChange: function(value) { alert(value) } });</pre>

		}
		});

方法

方法名	方法参数	描述
options	none	返回属性对象。
resize	width	组件大小。 代码示例： \$('#p').progressbar('resize'); // 更改进度条到原始宽度 \$('#p').progressbar('resize', 350); // 更改进度条到新的宽度
getValue	none	返回当前进度值。
setValue	value	设置一个新的进度值。

Tooltip（提示框）

使用 `$.fn.tooltip.defaults` 重写默认值对象。（译者注：1.3.3 版中新增的 plugin）

当用户将鼠标移动到元素上的时候，将会显示一个消息提示框。提示框的内容可以是页面中任何一个 HTML 元素或者通过 Ajax 发送后台请求以获取提示框内容。

移动到这里.....

这是上面控件的提示信息。

使用案例

创建提示框

1. 通过标签创建提示框，给元素添加一个“easyui-tooltip”的类名，无需任何 Javascript 代码。

```
1. <a href="#" title="This is the tooltip message." class="easyui-
   tooltip">Hover me</a>
```

2. 通过 Javascript 创建提示框。

```
1. <a id="dd" href="javascript:void(0)">Click here</a>

1. $('#dd').tooltip({
2.     position: 'right',
3.     content: '<span style="color:#fff">This is the tooltip
   message.</span>',
4.     onShow: function() {
5.         $(this).tooltip('tip').css({
6.             backgroundColor: '#666',
7.             borderColor: '#666'
8.         });
9.     }
10.});
```

属性

属性名	类型	描述	默认值
position	string	消息框位置。可用值有：“left”, “right”, “top”, “bottom”	bottom
content	string	消息框内容。	null
trackMouse	boolean	为 true 时，允许提示框跟着鼠标移动。	false
deltaX	number	水平方向提示框的位置。	0
deltaY	number	垂直方向提示框的位置。	0
showEvent	string	当激发什么事件的时候显示提示框。	mouseenter
hideEvent	string	当激发什么事件的时候隐藏提示框。	mouseleave
showDelay	number	延时多少秒显示提示框。	200
hideDelay	number	延时多少秒隐藏提示框。	100

事件

事件名称	事件参数	描述
onShow	e	在显示提示框的时候触发。
onHide	e	在隐藏提示框的时候触发。
onUpdate	content	在提示框内容更新的时候触发。
onPosition	left, top	在提示框位置改变的时候触发。
onDestroy	none	在提示框被销毁的时候触发。

方法

方法名	方法参数	描述
options	none	返回控件属性对象。
tip	none	返回 tip 元素对象。
arrow	none	返回箭头元素对象。
show	e	显示提示框。
hide	e	隐藏提示框。
update	content	更新提示框内容。
reposition	none	重置提示框位置。
destroy	none	销毁提示框。

Mobile（移动）

使用\$.mobile.defaults 重写默认值对象。（该组件自 1.4.2 版开始可用）

属性

属性名	类型	描述	默认值
-----	----	----	-----

animation	string	导航面板的动画效果类型。	slide
direction	string	导航面板的动画效果方向。	left

方法

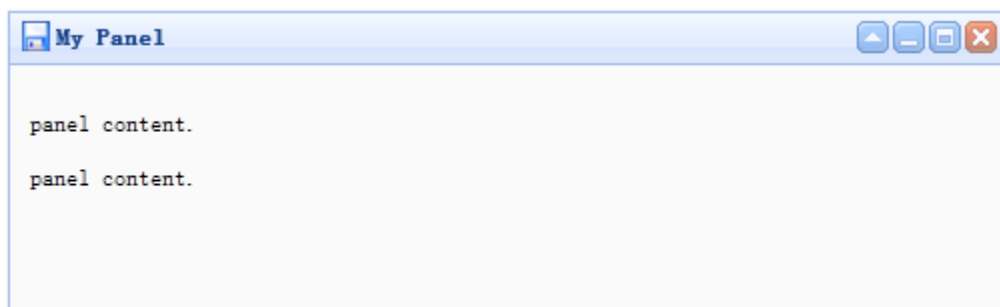
方法名	方法参数	描述
\$.mobile.init	none	初始化移动面板。
\$.mobile.nav	from, to, animation, direction	从一个面板导航到另外一个面板。 代码示例: \$.mobile.nav('#p1', '#p2', 'slide', 'left');
\$.mobile.go	panel, animation, direction	导航到指定面板。 代码示例: \$.mobile.go('#p2'); \$.mobile.go('#p3', 'slide', 'right');
\$.mobile.back	none	导航回到上一个面板。

第二章 Layout (布局)

Panel (面板)

使用 `$.fn.panel.defaults` 重写默认值对象。

面板作为承载其它内容的容器。这是构建其他组件的基础（比如：layout, tabs, accordion 等）。它还提供了折叠、关闭、最大化、最小化和自定义行为。面板可以很容易地嵌入到 web 页面的任何位置。



使用案例

创建面板

1. 通过标签创建面板

通过标签创建更简单。添加 'easyui-panel' 类 ID 到 <div/> 标签。

```

1. <div id="p" class="easyui-panel" title="My Panel"
2.     style="width:500px;height:150px;padding:10px;background:#fafafa;"
3.     data-options="iconCls:'icon-save',closable:true,
4.         collapsible:true,minimizable:true,maximizable:true">
5.     <p>panel content.</p>
6.     <p>panel content.</p>
7. </div>

```

2. 创建面板程序

让我们创建面板右上角的工具栏。

```

1. <div id="p" style="padding:10px;">
2.     <p>panel content.</p>
3.     <p>panel content.</p>
4. </div>
5.
6. $('#p').panel({
7.     width:500,
8.     height:150,
9.     title: 'My Panel',
10.    tools: [{
11.        iconCls:'icon-add',
12.        handler:function() {alert('new')}
13.    }, {
14.        iconCls:'icon-save',
15.        handler:function() {alert('save')}
16.    }]

```



```
17. });
```

移动面板

调用 'move' 方法移动面板到新的位置。

```
1. $('#p').panel('move', {
2.     left:100,
3.     top:100
4. });
```

读取内容

当加载成功的时候让我们通过 ajax 加载面板内容并显示一些消息。

```
1. $('#p').panel({
2.     href:'content_url.php',
3.     onLoad:function() {
4.         alert('loaded successfully');
5.     }
6. });
```

属性

属性名	类型	描述	默认值
id	string	面板的 ID 属性。	null
title	string	在面板头部显示的标题文本。	null
iconCls	string	设置一个 16x16 图标 CSS 类 ID 显示在面板左上角。	null
width	number	设置面板宽度。	auto
height	number	设置面板高度。	auto
left	number	设置面板距离左边的位置 (即 X 轴位置)。	null
top	number	设置面板距离顶部的位置 (即 Y 轴位置)。	null
cls	string	添加一个 CSS 类 ID 到面板。	null
headerCls	string	添加一个 CSS 类 ID 到面板头部。	null
bodyCls	string	添加一个 CSS 类 ID 到面板正文部分。	null
style	object	添加一个当前指定样式到面板。 如下代码示例更改面板边框宽度： <pre><div class="easyui-panel" style="width:200px;height:100px" data-options="style:{borderWidth:2}"> </div></pre>	{}
fit	boolean	当设置为 true 的时候面板大小将自适应父容器。下面的例子显示了一个面板，可以自动在父容器的最大范围内调整大小。	false

		<pre> <div style="width:200px;height:100px;padding:5px"> <div class="easyui-panel" style="width:200px;height:100px" data-options="fit:true,border:false"> Embedded Panel </div> </div> </pre>	
border	boolean	定义是否显示面板边框。	true
doSize	boolean	如果设置为 true，在面板被创建的时候将重置大小和重新布局。	true
noheader	boolean	如果设置为 true，那么将不会创建面板标题。	false
content	string	面板主体内容。	null
collapsible	boolean	定义是否显示可折叠按钮。	false
minimizable	boolean	定义是否显示最小化按钮。	false
maximizable	boolean	定义是否显示最大化按钮。	false
closable	boolean	定义是否显示关闭按钮。	false
tools	array, selector	<p>自定义工具菜单，可用值：</p> <ol style="list-style-type: none"> 1) 数组，每个元素都包含 'iconCls' 和 'handler' 属性。 2) 指向工具菜单的选择器。 <p>面板工具菜单可以声明在已经存在的<div>标签上：</p> <pre> <div class="easyui-panel" style="width:300px;height:200px" title="My Panel" data-options="iconCls:' icon-ok',tools:'#tt'"> </div> <div id="tt"> </div> </pre> <p>面板工具菜单也可以通过数组定义：</p> <pre> <div class="easyui-panel" style="width:300px;height:200px" title="My Panel" data-options="iconCls:' icon-ok', tools:[{ iconCls:' icon-add', handler:function() {alert(' add')}} </pre>	[]

		<pre> }, { iconCls: 'icon-edit', handler: function() {alert('edit')} }]"]> </div> </pre>	
header	selector	<p>定义面板标题。(该属性自 1.4.2 版开始可用)</p> <p>代码示例:</p> <pre> <div class="easyui-panel" style="width:300px;height:200px" title="我的面板"> <header>面板标题</header> </div> </pre>	
footer	selector	<p>定义面板页脚。(该属性自 1.4.1 版开始可用)</p> <p>代码示例:</p> <pre> <div class="easyui-panel" style="width:300px;height:200px" title="我的面板" data-options="iconCls:'icon-ok',tools:[{ iconCls:'icon-add',handler:function(){ alert('添加') } }, { iconCls:'icon-edit',handler:function(){ alert('编辑') } }]"]> </div> </pre>	null
openAnimation	string	定义打开面板的动画, 可用值有: 'slide', 'fade', 'show'。 (该属性自 1.4.1 版开始可用)	
openDuration	number	定义打开面板的持续时间。 (该属性自 1.4.1 版开始可用)	400
closeAnimation	string	定义关闭面板的动画, 可用值有: 'slide', 'fade', 'show'。 (该属性自 1.4.1 版开始可用)	
closeDuration	number	定义关闭面板的持续时间。 (该属性自 1.4.1 版开始可用)	400
collapsed	boolean	定义是否在初始化的时候折叠面板。	false
minimized	boolean	定义是否在初始化的时候最小化面板。	false
maximized	boolean	定义是否在初始化的时候最大化面板。	false
closed	boolean	定义是否在初始化的时候关闭面板。	false
href	string	<p>从 URL 读取远程数据并且显示到面板。注意: 内容将不会被载入, 直到面板打开或扩大, 在创建延迟加载面板时是非常有用的:</p> <pre> <div id="pp" class="easyui-panel" style="width:300px;height:200px" data-options="href='get_content.php',closed:true"> </div> </pre>	null

		<code>Open</code>	
cache	boolean	如果为 true，在超链接载入时缓存面板内容。	true
loadingMessage	string	在加载远程数据的时候在面板内显示一条消息。	Loading ...
extractor	function	定义如何从 ajax 应答数据中提取内容，返回提取数据。 extractor: function(data) { var pattern = /<body[^\>]*>((. [\n\r])*>\</body>/im; var matches = pattern.exec(data); if (matches) { return matches[1]; // 仅提取主体内容 } else { return data; } }	
method	string	使用 HTTP 的哪一种方法读取内容页。可用值： 'get', 'post'。(该属性自 1.3.6 版开始可用)	get
queryParams	object	在加载内容页的时候添加的请求参数。(该属性自 1.3.6 版开始可用)	{}
loader	function	定义了如何从远程服务器加载内容页，该函数接受以下参数： param: 参数对象发送给远程服务器。 success(data): 在检索数据成功的时候调用的回调函数。 error(): 在检索数据失败的时候调用的回调函数。 (该属性自 1.3.6 版开始可用)	

事件

事件名	事件参数	描述
onBeforeLoad	none	在加载内容页之前触发，返回 false 将忽略该动作。(该事件自 1.3.6 版开始可用)
onLoad	none	在加载远程数据时触发。
onLoadError	none	在加载内容页发生错误时触发。(该事件自 1.3.6 版开始可用)
onBeforeOpen	none	在打开面板之前触发，返回 false 可以取消打开操作。
onOpen	none	在打开面板之后触发。
onBeforeClose	none	在关闭面板之前触发，返回 false 可以取消关闭操作。下列的面板将不能关闭。 <code><div class="easyui-panel" style="width:300px;height:200px;" title="My Panel" data-options="onBeforeClose:function() {return false}"> 面板将不能关闭 </div></code>
onClose	none	在面板关闭之后触发。
onBeforeDestroy	none	在面板销毁之前触发，返回 false 可以取消销毁操作。

onDestroy	none	在面板销毁之后触发。
onBeforeCollapse	none	在面板折叠之前触发，返回 false 可以终止折叠操作。
onCollapse	none	在面板折叠之后触发。
onBeforeExpand	none	在面板展开之前触发，返回 false 可以终止展开操作。
onExpand	none	在面板展开之后触发。
onResize	width, height	在面板改变大小之后触发。 width: 新的宽度。 height: 新的高度。
onMove	left, top	在面板移动之后触发。 left: 新的左边距位置。 top: 新的上边距位置。
onMaximize	none	在窗口最大化之后触发。
onRestore	none	在窗口恢复到原始大小以后触发。
onMinimize	none	在窗口最小化之后触发。

方法

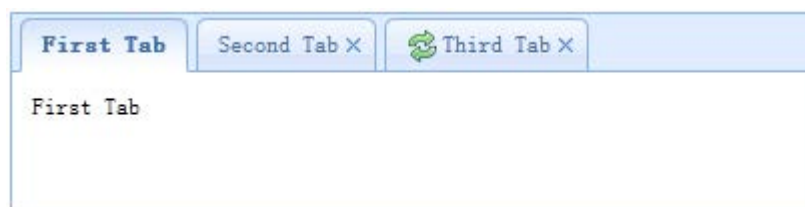
方法名	方法参数	描述
options	none	返回属性对象。
panel	none	返回面板对象。
header	none	返回面板头对象。
footer	none	返回面板页脚对象。 (该方法自 1.4.1 版开始可用)
body	none	返回面板主体对象。
setTitle	title	设置面板头的标题文本。
open	forceOpen	在 'forceOpen' 参数设置为 true 的时候，打开面板时将跳过 'onBeforeOpen' 回调函数。
close	forceClose	在 'forceClose' 参数设置为 true 的时候，关闭面板时将跳过 'onBeforeClose' 回调函数。
destroy	forceDestroy	在 'forceDestroy' 参数设置为 true 的时候，销毁面板时将跳过 'onBeforeDestroy' 回调函数。
clear	none	清除面板内容。 (该方法自 1.4 版开始可用)
refresh	href	刷新面板来装载远程数据。如果 'href' 属性有了新配置，它将重写旧的 'href' 属性。 代码示例： // 打开面板且刷新其内容。 <code>\$('#pp').panel('open').panel('refresh');</code> // 刷新一个新的 URL 内容 <code>\$('#pp').panel('open').panel('refresh', 'new_content.php');</code>
resize	options	设置面板大小和布局。参数对象包含下列属性： width: 新的面板宽度。 height: 新的面板高度。 left: 新的面板左边距位置。 top: 新的面板上边距位置。

		代码示例： <pre>\$('#pp').panel('resize',{ width: 600, height: 400 });</pre>
doLayout	none	设置面板内子组件的大小。 (该方法自 1.4 版开始可用)
move	options	移动面板到一个新位置。参数对象包含下列属性： left: 新的左边距位置。 top: 新的上边距位置。
maximize	none	最大化面板到容器大小。
minimize	none	最小化面板。
restore	none	恢复最大化面板回到原来的大小和位置。
collapse	animate	折叠面板主题。
expand	animate	展开面板主体。

Tabs（选项卡）

使用\$.fn.tabs.defaults 重写默认值对象。

选项卡显示一批面板。但在同一个时间只会显示一个面板。每个选项卡面板都有头标题和一些小的按钮工具菜单，包括关闭按钮和其他自定义按钮。



依赖关系

- panel
- linkbutton

使用案例

创建面板

1. 通过标签创建选项卡

通过标签可以更容易的创建选项卡，我们不需要写任何 Javascript 代码。只需要给<div/>标签添加一个类 ID 'easyui-tabs'。每个选项卡面板都通过子<div/>标签进行创建，用法和 panel(面板)相同。

```

1. <div id="tt" class="easyui-tabs" style="width:500px;height:250px;">
2.     <div title="Tab1" style="padding:20px;display:none;">
3.         tab1
4.     </div>
5.     <div title="Tab2" data-options="closable:true" style="overflow:auto;padding:20px;dis
   isplay:none;">
6.         tab2
7.     </div>
8.     <div title="Tab3" data-options="iconCls:'icon-reload',closable:true" style="paddin
   g:20px;display:none;">
9.         tab3
10.    </div>
11. </div>

```

2. 通过 Javascript 创建选项卡

下面的代码演示如何使用 Javascript 创建选项卡，当该选项卡被选择时将会触发 'onSelect' 事件。

```

1. $('#tt').tabs({
2.     border:false,
3.     onSelect:function(title){
4.         alert(title+' is selected');
5.     }
6. });

```

添加新的选项卡面板

添加一个新的包含小工具菜单的选项卡面板，小工具菜单图标(8x8)被放置在关闭按钮之前。

```

1. // add a new tab panel
2. $('#tt').tabs('add', {
3.     title:'New Tab',
4.     content:'Tab Body',
5.     closable:true,
6.     tools:[{
7.         iconCls:'icon-mini-refresh',
8.         handler:function() {
9.             alert('refresh');
10.        }
11.    }]
12. });

```

获取选择的选项卡

```

1. // get the selected tab panel and its tab object
2. var pp = $('#tt').tabs('getSelected');
3. var tab = pp.panel('options').tab; // the corresponding tab object

```

属性

属性名	类型	描述	默认值
width	number	选项卡容器宽度。	auto
height	number	选项卡容器高度。	auto
plain	boolean	设置为 true 时，将不显示控制面板背景。	false
fit	boolean	设置为 true 时，选项卡的大小将铺满它所在的容器。	false
border	boolean	设置为 true 时，显示选项卡容器边框。	true
scrollIncrement	number	选项卡滚动条每次滚动的像素值。	100
scrollDuration	number	每次滚动动画持续的时间，单位：毫秒。	400
tools	array, selector	工具栏添加在选项卡面板头的左侧或右侧。可用的值有： <ol style="list-style-type: none"> 一个工具菜单数组，每个工具选项都和 linkbutton 相同。 一个指向<div/>容器工具菜单的选择器。 代码示例： 通过数组定义工具菜单。	null

		<pre> \$(' #tt').tabs({ tools:[{ iconCls:' icon-add', handler:function() {alert(' 添加')}}],{ iconCls:' icon-save', handler:function() {alert(' 保存')}}] }); 通过存在的 DOM 容器定义工具菜单。 \$(' #tt').tabs({ tools:' #tab-tools' }); <div id="tab-tools"> </div> </pre>	
toolPosition	string	工具栏位置。可用值: 'left', 'right'。(该属性自 1.3.2 版开始可用)	right
tabPosition	string	选项卡位置。可用值: 'top', 'bottom', 'left', 'right'。(该属性自 1.3.2 版开始可用)	top
headerWidth	number	选项卡标题宽度, 在 tabPosition 属性设置为 'left' 或 'right' 的时候才有效。(该属性自 1.3.2 版开始可用)	150
tabWidth	number	标签条的宽度。(该属性自 1.3.4 版开始可用)	auto
tabHeight	number	标签条的高度。(该属性自 1.3.4 版开始可用)	27
selected	number	初始化选中一个标签页。(该属性自 1.3.5 版开始可用)	0
showHeader	boolean	设置为 true 时, 显示标签页标题。(该属性自 1.3.5 版开始可用)	true
justified	boolean	<p>设置为 true 时, 生成等宽标题选项卡。</p> <p>为 true 时:</p>  <p>为 false 时:</p>  <p>(该属性自 1.4.2 版开始可用)</p>	false
narrow	boolean	设置为 true 时, 删除选项卡标题之间的空间。	false

		<p>为 true 时: </p> <p>为 false 时: </p> <p>(该属性自 1.4.2 版开始可用)</p>	
pill	boolean	<p>设置为 true 时, 选项卡标题样式改为气泡状。</p> <p>为 true 时: </p> <p>为 false 时: </p> <p>(该属性自 1.4.2 版开始可用)</p>	false

事件

事件名	事件参数	描述
onLoad	panel	在 ajax 选项卡面板加载完远程数据的时候触发。
onSelect	title, index	用户在选择一个选项卡面板的时候触发。
onUnselect	title, index	用户在取消选择一个选项卡面板的时候触发。(该属性自 1.3.5 版开始可用)
onBeforeClose	title, index	<p>在选项卡面板关闭的时候触发, 返回 false 取消关闭操作。下面的例子展示了在关闭选项卡面板之前以何种方式显示确认对话框。</p> <pre> \$('#tt').tabs({ onBeforeClose: function(title) { return confirm('您确认想要关闭 ' + title); } }); // 使用异步确认对话框 \$('#tt').tabs({ onBeforeClose: function(title, index) { var target = this; \$.messager.confirm('确认', '你确认想要关闭' + title, function(r) { if (r) { var opts = \$(target).tabs('options'); var bc = opts.onBeforeClose; opts.onBeforeClose = function() {}; // 允许现在关闭 \$(target).tabs('close', index); } }); return bc ? bc.call(target, title, index) : true; } }); </pre>

		<pre> opts.onBeforeClose = bc; // 还原事件函数 } }); return false; // 阻止关闭 } }); </pre>
onClose	title, index	在用户关闭一个选项卡面板的时候触发。
onAdd	title, index	在添加一个新选项卡面板的时候触发。
onUpdate	title, index	在更新一个选项卡面板的时候触发。
onContextMenu	e, title, index	在右键点击一个选项卡面板的时候触发。

方法

方法名	方法参数	描述
options	none	返回选项卡属性。
tabs	none	返回所有选项卡面板。
resize	none	调整选项卡容器大小和布局。
add	options	<p>添加一个新选项卡面板，选项参数是一个配置对象，查看选项卡面板属性的更多细节。在添加一个新选项卡面板的时候它将变成可选的。</p> <p>添加一个非选中状态的选项卡面板时，记得设置' selected' 属性为 false。</p> <p>// 添加一个未选中状态的选项卡面板</p> <pre> \$('#tt').tabs('add', { title: '新选项卡面板', selected: false //... }); </pre> <p>(该功能自 1.4.2 版开始可用)</p> <p>// 在索引为 2 的位置上插入一个选项卡面板</p> <pre> \$('#tt').tabs('add', { title: '插入的选项卡面板', index: 2 }); </pre>
close	which	关闭一个选项卡面板，' which' 参数可以是选项卡面板的标题或者索引，以指定要关闭的面板。
getTab	which	获取指定选项卡面板，' which' 参数可以是选项卡面板的标题或者索引。
getTabIndex	tab	获取指定选项卡面板的索引。
getSelected	none	<p>获取选择的选项卡面板。下面的例子展示了如何获取选择的选项卡面板索引。</p> <pre> var tab = \$('#tt').tabs('getSelected'); var index = \$('#tt').tabs('getTabIndex', tab); alert(index); </pre>
select	which	选择一个选项卡面板，' which' 参数可以是选项卡面板的标题或者索引。
unselect	which	<p>取消选择一个选项卡面板，' which' 参数可以是选项卡面板的标题或者索引。</p> <p>(该方法自 1.3.5 版开始可用)</p>

showHeader	none	显示选项卡的标签头。 (该方法自 1.3.5 版开始可用)
hideHeader	none	隐藏选项卡的标签头。 (该方法自 1.3.5 版开始可用)
showTool	none	显示工具栏。 (该方法自 1.4.3 版开始可用)
hideTool	none	隐藏工具栏。 (该方法自 1.4.3 版开始可用)
exists	which	表明指定的面板是否存在，'which' 参数可以是选项卡面板的标题或索引。
update	param	<p>更新指定的选项卡面板，'param' 参数包含 2 个属性：</p> <p>tab: 要更新的选项卡面板。</p> <p>type: 更新类型，可用值有：'header'，'body'，'all'。(该参数自 1.4.1 版开始可用)</p> <p>options: 面板的属性。</p> <p>代码示例：</p> <pre>// 更新选择的面板的新标题和内容 var tab = \$('#tt').tabs('getSelected'); // 获取选择的的面板 \$('#tt').tabs('update', { tab: tab, options: { title: '新标题', href: 'get_content.php' // 新内容的 URL } }); // 调用 'refresh' 方法更新选项卡面板的内容 var tab = \$('#tt').tabs('getSelected'); // 获取选择的的面板 tab.panel('refresh', 'get_content.php');</pre>
enableTab	which	<p>启用指定的选项卡面板，'which' 参数可以是选项卡面板的标题或索引。(该方法自 1.3 版开始可用)</p> <p>代码示例：</p> <pre>\$('#tt').tabs('enableTab', 1); // 启用第二个选项卡面板 \$('#tt').tabs('enableTab', 'Tab2'); // 启用标题为 Tab2 的选项卡面板</pre>
disableTab	which	<p>禁用指定的选项卡面板，'which' 参数可以是选项卡面板的标题或索引。(该方法自 1.3 版开始可用)</p> <p>代码示例：</p> <pre>\$('#tt').tabs('disableTab', 1); // 禁用第二个选项卡面板</pre>
scrollBy	deltaX	<p>滚动选项卡标题指定的像素数量，负值则向右滚动，正值则向左滚动。(该方法自 1.3 版开始可用)</p> <p>代码示例：</p> <pre>// 滚动选项卡标题到左边 \$('#tt').tabs('scroll', 10);</pre>

选项卡面板

选项卡面板属性与 panel 组件属性的定义类似，下面是 2 个组件的一些公共属性。

属性名	类型	描述	默认值
id	string	选项卡面板的 ID 属性。	null
title	string	选项卡面板的标题文本。	
content	string	选项卡面板的内容。	
href	string	从 URL 加载远程数据内容填充到选项卡面板。	null
cache	boolean	如果为 true，在 'href' 属性设置了有效值的时候缓存选项卡面板。	true
iconCls	string	定义了一个图标的 CSS 类 ID 显示到选项卡面板标题。	null
width	number	选项卡面板宽度。	auto
height	number	选项卡面板高度。	auto
collapsible	boolean	如果为 true，则允许选项卡摺叠。	false

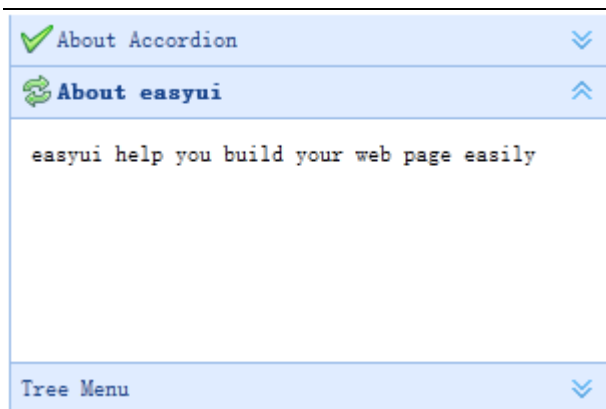
下面的是选项卡面板新增且独有的属性。

属性名	类型	描述	默认值
closable	boolean	在设置为 true 的时候，选项卡面板将显示一个关闭按钮，在点击的时候会关闭选项卡面板。	false
selected	boolean	在设置为 true 的时候，选项卡面板会被选中。	false
disabled	boolean	在设置为 true 的时候，选项卡面板会被禁用。 (该属性自 1.4.4 版开始可用)	false

Accordion (分类选项卡)

使用 `$.fn.accordion.defaults` 重写默认值对象。

分类空间允许用户使用多面板，但在同一时间只会显示一个。每个面板都内建支持展开和折叠功能。点击一个面板的标题将会展开或折叠面板主体。面板内容可以通过指定的 'href' 属性使用 ajax 方式读取面板内容。用户可以定义一个被默认选中的面板，如果未指定，那么第一个面板就是默认的。



依赖关系

- panel

使用案例

创建分类

通过标签创建分类，给<div/>标签添加一个名为'easyui-accordion'的类 ID。

```

1. <div id="aa" class="easyui-accordion" style="width:300px;height:200px;">
2.     <div title="Title1" data-options="iconCls:'icon-save'" style="overflow:auto;padding:10px;">
3.         <h3 style="color:#0099FF;">Accordion for jQuery</h3>
4.         <p>Accordion is a part of easyui framework for jQuery.
5.         It lets you define your accordion component on web page more easily.</p>
6.     </div>
7.     <div title="Title2" data-options="iconCls:'icon-reload',selected:true" style="padding:10px;">
8.         content2
9.     </div>
10.    <div title="Title3">
11.        content3
12.    </div>
13. </div>

```

我们可以更改或修改面板的一些功能以后再重新创建它。

```

1. $('#aa').accordion({
2.     animate:false
3. });

```

刷新分类面板内容

调用'getSelected'方法获取当前面板，此外我们还可以调用'refresh'方法重新载入新内容。

```

1. var pp = $('#aa').accordion('getSelected'); // 获取选择的面板
2. if (pp) {
3.     pp.panel('refresh', 'new_content.php'); // 调用'refresh'方法刷新
4. }

```

容器属性

属性名	类型	描述	默认值
width	number	分类容器的宽度。	auto
height	number	分类容器的高度。	auto
fit	boolean	如果设置为 true，分类容器大小将自适应父容器。	false
border	boolean	定义是否显示边框。	true
animate	boolean	定义在展开和折叠的时候是否显示动画效果。	true
multiple	boolean	如果为 true 时，同时展开多个面板。 (该属性自 1.3.5 版开始可用)	false
selected	number	设置初始化时默认选中的面板索引号。 (该属性自 1.3.5 版开始可用)	0

面板属性

分类面板属性继承自 panel (面板)，分类面板新增的属性如下：

属性名	类型	描述	默认值
selected	boolean	如果设置为 true 将展开面板。	false
collapsible	boolean	如果设置为 true 将显示折叠按钮。	true

事件

事件名	事件参数	描述
onSelect	title, index	在面板被选中的时候触发。
onUnselect	title, index	在面板被取消选中的时候触发。 (该方法自 1.3.5 版开始可用)
onAdd	title, index	在添加新面板的时候触发。
onBeforeRemove	title, index	在移除面板之前触发，返回 false 可以取消移除操作。
onRemove	title, index	在面板被移除的时候触发。

方法

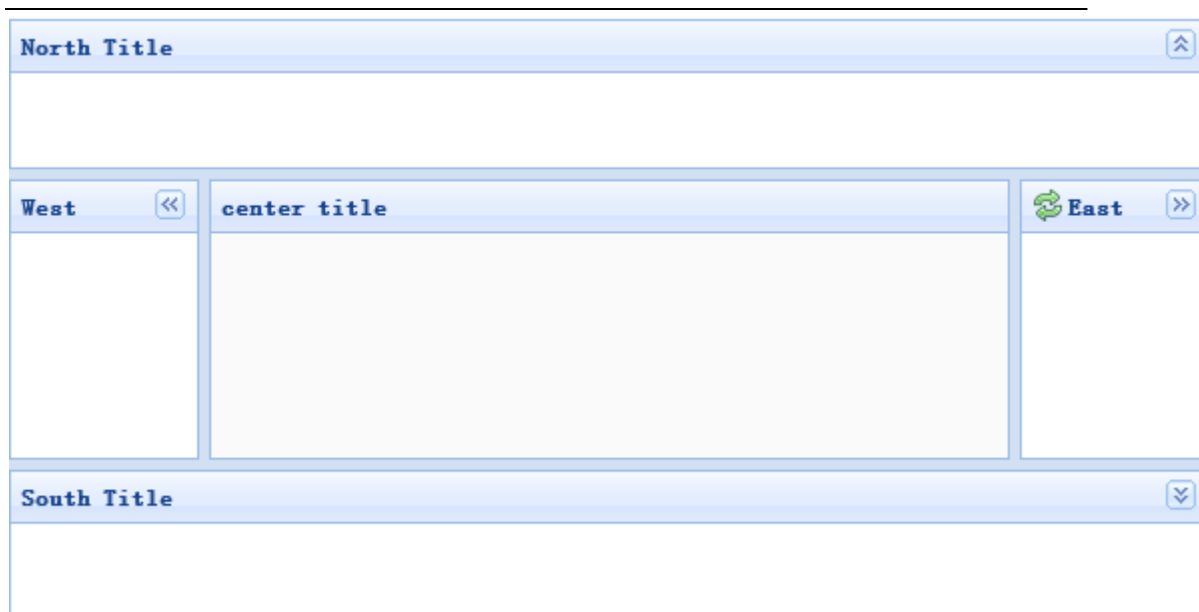
方法名	方法参数	描述
options	none	返回分类组件的属性。
panels	none	获取所有面板。
resize	none	调整分类组件大小。
getSelected	none	获取选中的面板。

getSelections	none	获取所有选中的面板。 (该方法自 1.3.5 版开始可用)
getPanel	which	获取指定的面板，'which' 参数可以是面板的标题或者索引。
getPanelIndex	panel	获取指定面板的索引。 (该方法自 1.3 版开始可用) 以下示例显示如何获取选中面板的索引。 <pre>var p = \$('#aa').accordion('getSelected'); if (p) { var index = \$('#aa').accordion('getPanelIndex', p); alert(index); }</pre>
select	which	选择指定面板。'which' 参数可以是面板标题或者索引。
unselect	which	取消选择指定面板。'which' 参数可以是面板标题或者索引。 (该方法自 1.3.5 版开始可用)
add	options	添加一个新面板。在默认情况下，新增的面板会变成当前面板。如果要添加一个非选中面板，不要忘记将'selected' 属性设置为 false。 代码示例： <pre>\$('#aa').accordion('add', { title: '新标题', content: '新内容', selected: false });</pre>
remove	which	移除指定面板。'which' 参数可以使面板的标题或者索引。

Layout（布局）

使用\$.fn.layout.defaults 重写默认值对象。

布局容器有 5 个区域：北、南、东、西和中间。中间区域面板是必须的，边缘的面板都是可选的。每个边缘区域面板都可以通过拖拽其边框改变大小，也可以点击折叠按钮将面板折叠起来。布局可以进行嵌套，用户可以通过组合布局构建复杂的布局结构。



依赖关系

- panel
- resizable

使用案例

创建布局

1. 通过标签创建布局

为<div/>标签增加名为'easyui-layout'的类 ID。

```

1. <div id="cc" class="easyui-layout" style="width:600px;height:400px;">
2.     <div data-options="region:'north',title:'North Title',split:true" style="height:100px;"></div>
3.     <div data-options="region:'south',title:'South Title',split:true" style="height:100px;"></div>
4.     <div data-options="region:'east',iconCls:'icon-reload',title:'East',split:true" style="width:100px;"></div>
5.     <div data-options="region:'west',title:'West',split:true" style="width:100px;"></div>
6.     <div data-options="region:'center',title:'center title'" style="padding:5px;background:#eee;"></div>
7. </div>

```

2. 使用完整页面创建布局

```

1. <body class="easyui-layout">

```

```

2.     <div data-options="region:'north',title:'North Title',split:true" style="height:100px;"></div>
3.     <div data-options="region:'south',title:'South Title',split:true" style="height:100px;"></div>
4.     <div data-options="region:'east',iconCls:'icon-reload',title:'East',split:true" style="width:100
    px;"></div>
5.     <div data-options="region:'west',title:'West',split:true" style="width:100px;"></div>
6.     <div data-options="region:'center',title:'center title'" style="padding:5px;background:#eee;"></
    div>
7. </body>

```

3. 创建嵌套布局

注意：嵌套在内部的布局面板的左侧（西面）面板是折叠的。

```

1. <body class="easyui-layout">
2.     <div data-options="region:'north'" style="height:100px"></div>
3.     <div data-options="region:'center'">
4.         <div class="easyui-layout" data-options="fit:true">
5.             <div data-options="region:'west',collapsed:true" style="width:180px"></div>
6.             <div data-options="region:'center'"></div>
7.         </div>
8.     </div>
9. </body>

```

4. 通过 ajax 读取内容

布局是以面板为基础创建的。所有的布局面板都支持异步加载 URL 内容。使用异步加载技术，用户可以使自己的布局页面显示的内容更多更快。

```

1. <body class="easyui-layout">
2.     <div data-options="region:'west',href:'west_content.php'" style="width:180px" ></div>
3.     <div data-options="region:'center',href:'center_content.php'" ></div>
4. </body>

```

折叠布局面板

```

1. $('#cc').layout();
2. // collapse the west panel
3. $('#cc').layout('collapse','west');

```

添加西侧区域面板和工具菜单按钮

```

1. $('#cc').layout('add',{
2.     region: 'west',
3.     width: 180,
4.     title: 'West Title',
5.     split: true,
6.     tools: [{

```

```

7.         iconCls: 'icon-add',
8.         handler: function() {alert(' add')}
9.     }, {
10.        iconCls: 'icon-remove',
11.        handler: function() {alert(' remove')}
12.    }]
13. });

```

布局属性

属性名	类型	描述	默认值
fit	boolean	如果设置为 true，布局组件将自适应父容器。当使用 'body' 标签创建布局的时候，整个页面会自动最大。	false

布局事件

事件名	事件参数	描述
onCollapse	region	在折叠区域面板的时候触发。 (该事件自 1.4.4 版开始可用)
onExpand	region	在展开区域面板的时候触发。 (该事件自 1.4.4 版开始可用)
onAdd	region	在新增区域面板的时候触发。 (该事件自 1.4.4 版开始可用)
onRemove	region	在移除区域面板的时候触发。 (该事件自 1.4.4 版开始可用)

区域面板属性

区域面板属性定义与 [panel](#) 组件类似，下面的是公共的和新增的属性：

属性名	类型	描述	默认值
title	string	布局面板标题文本。	null
region	string	定义布局面板位置，可用的值有：north, south, east, west, center。	
border	boolean	为 true 时显示布局面板边框。	true
split	boolean	为 true 时用户可以通过分割栏改变面板大小。	false
iconCls	string	一个包含图标的 CSS 类 ID，该图标将会显示到面板标题上。	null
href	string	用于读取远程站点数据的 URL 链接	null
collapsible	boolean	定义是否显示折叠按钮。 (该属性自 1.3.3 版开始可用)	true
minWidth	number	最小面板宽度。 (该属性自 1.3.3 版开始可用)	10
minHeight	number	最小面板高度。 (该属性自 1.3.3 版开始可用)	10

maxWidth	number	最大面板宽度。 (该属性自 1.3.3 版开始可用)	10000
maxHeight	number	最大面板高度。 (该属性自 1.3.3 版开始可用)	10000
expandMode	string	在点击折叠面板时候的扩展模式。可用值有：“float”、“dock”和 null float：区域面板将展开并浮动在顶部，在鼠标焦点离开面板时会自动隐藏。 dock：区域面板将展开并钉在面板上，在鼠标焦点离开面板时不会自动隐藏。 null：什么也不会发生。 (该属性自 1.4.4 版开始可用)	float
collapsedSize	number	折叠后的面板大小。 (该属性自 1.4.4 版开始可用)	28
hideExpandTool	boolean	为 true 时隐藏折叠面板上的扩展面板工具。 (该属性自 1.4.4 版开始可用)	false
hideCollapsedContent	boolean	为 true 时隐藏折叠面板上的标题栏。 (该属性自 1.4.4 版开始可用)	true
collapsedContent	string, function(title)	定义在折叠面板上要显示标题内容。 1. 标题字符串； 2. 通过函数返回标题内容。 (该方法自 1.4.4 版开始可用) 代码示例： <pre>collapsedContent: function(title) { var region = \$(this).panel('options').region; if (region == 'north' region == 'south') { return title; } else { return '<div class="mytitle">' + title + '</div>'; } }</pre>	

方法

方法名	方法属性	描述
resize	param	<p>设置布局大小。param 对象包含如下属性：</p> <p>width：布局宽度。</p> <p>height：布局高度。</p> <p>代码示例：</p> <pre>\$('#cc').layout('resize', { width: '80%',</pre>

		<pre>height:300 });</pre>						
panel	region	返回指定面板。 'region' 参数可用值有: 'north','south','east','west','center'。						
collapse	region	折叠指定面板。'region' 参数可用值有: 'north','south','east','west'。						
expand	region	展开指定面板。'region' 参数可用值有: 'north','south','east','west'。						
add	options	添加指定面板。属性参数是一个配置对象, 更多细节请查看选项卡面板属性。						
remove	region	移除指定面板。'region' 参数可用值有: 'north','south','east','west'。						
split	region	分割区域面板。'region' 参数可用值有: 'north','south','east','west'。 (该方法自 1.4.2 版开始可用) 代码示例: <pre>\$("#layout").layout("split", "west");</pre> <table><tr><td>West</td><td><<</td><td>✓ Main Title</td></tr><tr><td></td><td></td><td>Item ID Produc</td></tr></table>	West	<<	✓ Main Title			Item ID Produc
West	<<	✓ Main Title						
		Item ID Produc						
unsplit	region	移除指定面板。'region' 参数可用值有: 'north','south','east','west'。 (该方法自 1.4.2 版开始可用) 代码示例: <pre>\$("#layout").layout("unsplit", "west");</pre> <table><tr><td>West</td><td><<</td><td>✓ Main Title</td></tr><tr><td></td><td></td><td>Item ID Proc</td></tr></table>	West	<<	✓ Main Title			Item ID Proc
West	<<	✓ Main Title						
		Item ID Proc						

第三章

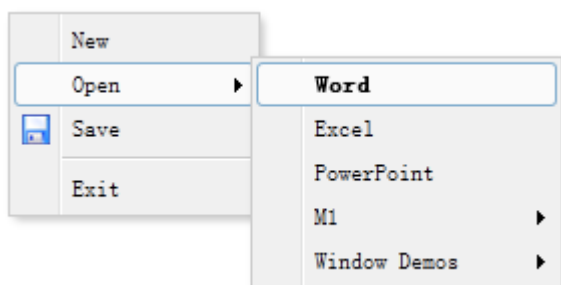
Menu and Button

(菜单和按钮)

Menu (菜单)

使用`$.fn.menu.defaults` 重写默认值对象。

菜单组件通常用于快捷菜单。他是构建其他菜单组件的必备基础组件。比如：`menubutton` 和 `splitbutton`。它还可以用于导航和执行命令。



使用案例

创建菜单

通过标签创建菜单，给<div/>标签添加一个名为'easyui-menu'的类 ID。每个菜单项都通过<div/>标签创建。我们可以添加'iconCls'属性来给菜单项定义一个图标显示到菜单项的左侧。添加'menu-sep'类 ID 菜单项将会生成一个菜单分割线。

```

1. <div id="mm" class="easyui-menu" style="width:120px;">
2.     <div>New</div>
3.     <div>
4.         <span>Open</span>
5.         <div style="width:150px;">
6.             <div><b>Word</b></div>
7.             <div>Excel</div>
8.             <div>PowerPoint</div>
9.         </div>
10.    </div>
11.    <div data-options="iconCls:'icon-save'">Save</div>
12.    <div class="menu-sep"></div>
13.    <div>Exit</div>
14. </div>

```

使用 Javascript 创建菜单项并监听'onClick'事件。

```

1. $('#mm').menu({
2.     onClick:function(item) {
3.         //...
4.     }
5. });

```

显示菜单

在菜单被创建的时候它是隐藏和不可见的。调用 'show' 方法显示菜单。

```
1. $('#mm').menu('show', {
2.     left: 200,
3.     top: 100
4. });
```

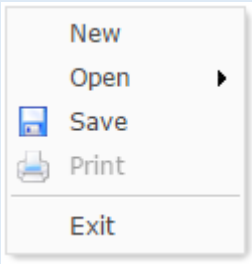
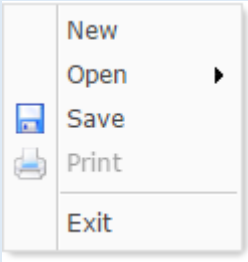
菜单项

菜单项代表显示在菜单上的一个单独的项目。它包含以下属性：

属性名	类型	描述	默认值
id	string	菜单项 ID 属性。	
text	string	菜单项文本。	
iconCls	string	显示在菜单项左侧的 16x16 像素图标的 CSS 类 ID。	
href	string	设置点击菜单项时候的页面位置。	
disabled	boolean	定义是否显示菜单项。	false
onclick	function	在点击菜单项的时候调用的函数。	

属性

属性名	类型	描述	默认值
zIndex	number	菜单 z-index 样式，增加它的值。	110000
left	number	菜单的左边距位置。	0
top	number	菜单的上边距位置。	0
align	string	菜单对齐方式，可用值有：'left' 和 'right'。 (该属性自 1.4.2 版开始可用)	left
minWidth	number	菜单的最小宽度。(该属性自 1.3.2 版开始可用)	120
itemHeight	number	菜单项高度。(该属性自 1.4.2 版开始可用)	22
duration	number	该属性允许用户自定义隐藏菜单动画的持续时间，以毫秒为单位。(该属性自 1.4 版开始可用)	100
hideOnUnhover	boolean	当设置为 true 时，在鼠标离开菜单的时候将自动隐藏菜单。 (该属性自 1.3.5 版开始可用)	true
inline	boolean	当设置为 true 时，菜单将相对于父级标签进行定位，否则相对于 body 进行定位。(该属性自 1.4.2 版开始可用)	false
fit	boolean	当设置为 true 时，菜单尺寸将自动适配父容器。 (该属性自 1.4.2 版开始可用)	false
noline	boolean	当设置为 true 时，菜单将不显示图标和文字之间的分割线。 (该属性自 1.4.2 版开始可用)	false

		为 true 时:		
		为 false 时:		

事件

事件名	参数	描述
onShow	none	在菜单显示之后触发。
onHide	none	在菜单隐藏之后触发。
onClick	item	<p>在菜单项被点击的时候触发。下面的例子显示了如何处理所有菜单项的点击：</p> <pre> <div class="easyui-menu" data-options="onClick:menuHandler" style="width:120px;"> <div data-options="name:'new'">新建</div> <div data-options="name:'save',iconCls:'icon-save'">保存</div> <div data-options="name:'print',iconCls:'icon-print'">打印</div> <div class="menu-sep"></div> <div data-options="name:'exit'">退出</div> </div> <script type="text/javascript"> function menuHandler(item){ alert(item.name) } </script> </pre>

方法

方法名	方法参数	描述
options	none	返回属性对象。
show	pos	<p>显示菜单到指定的位置。'pos' 参数有 2 个属性：</p> <p>left: 新的左边距位置。</p> <p>top: 新的上边距位置。</p>
hide	none	隐藏菜单。
destroy	none	销毁菜单。

getItem	itemEl	<p>获取指定的菜单项。得到的是一个菜单项的 DOM 元素。下面的例子展示了如何根据 ID 获取指定的项：</p> <pre><div id="mm" class="easyui-menu" style="width:120px"> <div>New</div> <div id="m-open">Open</div> <div>Save</div> </div> var itemEl = \$('#m-open')[0]; // 获取菜单项 var item = \$('#mm').menu('getItem', itemEl); console.log(item);</pre>
setText	param	<p>设置指定菜单项的文本。'param' 参数包含 2 个属性：</p> <p>target: DOM 对象，要设置值的菜单项。</p> <p>text: 字符串，要设置的新文本值。</p> <p>代码示例：</p> <pre>var item = \$('#mm').menu('findItem', '保存'); \$('#mm').menu('setText', { target: item.target, text: '修改' });</pre>
setIcon	param	<p>设置指定菜单项图标。'param' 参数包含 2 个属性：</p> <p>target: DOM 对象，要设置的菜单项。</p> <p>iconCls: 新的图标 CSS 类 ID。</p> <p>代码示例：</p> <pre>\$('#mm').menu('setIcon', { target: \$('#m-open')[0], iconCls: 'icon-closed' });</pre>
findItem	text	<p>查找的指定菜单项，返回的对象和 getItem 方法是一样的。</p> <p>代码示例：</p> <pre>// 查找“打开”项并禁用它 var item = \$('#mm').menu('findItem', '打开'); \$('#mm').menu('disableItem', item.target);</pre>
appendItem	options	<p>追加新的菜单项，'options' 参数代表新菜单项属性。默认情况下添加的项在菜单项的顶部。追加一个子菜单项，'parent' 属性应该设置指定的父项元素，并且该父项元素必须是已经定义在页面上的。</p> <p>代码示例：</p> <pre>// 追加一个顶部菜单 \$('#mm').menu('appendItem', { text: '新菜单项', iconCls: 'icon-ok',</pre>

		<pre> onclick: function() {alert('提示: 新菜单项!')} }); // 追加一个子菜单项 var item = \$('#mm').menu('findItem', '打开'); // 查找“打开”项 \$('#mm').menu('appendItem', { parent: item.target, // 设置父菜单元素 text: '打开 Excel 文档', iconCls: 'icon-excel', onclick: function() {alert('提示: 打开 Excel 文档!')} }); </pre>
removeItem	itemEl	移除指定的菜单项。
enableItem	itemEl	启用菜单项。
disableItem	itemEl	禁用菜单项。
showItem	itemEl	显示菜单项。(该方法自 1.4 版开始可用)
hideItem	itemEl	隐藏菜单项。(该方法自 1.4 版开始可用)
resize	menuEl	重置指定的菜单项。(该方法自 1.4 版开始可用)

LinkButton (按钮)

使用 \$.fn.linkbutton.defaults 重写默认值对象。

按钮组件使用超链接按钮创建。它使用一个普通的 <a> 标签进行展示。它可以同时显示一个图标和文本, 或只有图标或文字。按钮的宽度可以动态和折叠/展开以适应它的文本标签。



使用案例

创建按钮

使用标签创建按钮更加简单。

```
1. <a id="btn" href="#" class="easyui-linkbutton" data-options="iconCls:'icon-search'">easyui</a>
```

也可以使用 Javascript 创建按钮。

```
1. <a id="btn" href="#">easyui</a>
2. $('#btn').linkbutton({
3.     iconCls: 'icon-search'
4. });
```

处理按钮的点击

点击按钮会将用户引导到其他页面。

```
1. <a href="otherpage.php" class="easyui-linkbutton" data-options="iconCls:'icon-search'">easyui</a>
```

下面的示例提示了一个警告信息。

```
1. <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-search'"
2.     onclick="javascript:alert('easyui')">easyui</a>
```

使用 jQuery 绑定点击事件。

```
1. <a id="btn" href="#" class="easyui-linkbutton" data-options="iconCls:'icon-search'">easyui</a>
1. $(function() {
2.     $('#btn').bind('click', function() {
3.         alert('easyui');
4.     });
5. });
```

属性

属性名	类型	描述	默认值
width	number	组件的宽度。 (该属性自 1.4 版开始可用)	null
height	number	组件的高度。 (该属性自 1.4 版开始可用)	null
id	string	组件的 ID 属性。	null
disabled	boolean	为 true 时禁用按钮。	false
toggle	boolean	为 true 时允许用户切换其状态是被选中还是未选择，可实现 checkbox 复选效果。 (该属性自 1.3.3 版开始可用)	false

selected	boolean	定义按钮初始的选择状态，true 为被选中，false 为未选中。 (该属性自 1.3.3 版开始可用)	false
group	string	指定相同组名称的按钮同属于一个组，可实现 radio 单选效果。 (该属性自 1.3.3 版开始可用)	null
plain	boolean	为 true 时显示简洁效果。	false
text	string	按钮文字。	''
iconCls	string	显示在按钮文字左侧的图标(16x16)的 CSS 类 ID。	null
iconAlign	string	按钮图标位置。可用值有： 'left', 'right' (该属性自 1.3.2 版开始可用) 'top', 'bottom' (该属性自 1.3.6 版开始可用)	left
size	string	按钮大小。可用值有：'small', 'large'。 (该属性自 1.3.6 版开始可用)	small

事件

事件名	事件参数	描述
onClick	none	在点击按钮的时候触发。 (该事件自 1.3.6 版开始可用)

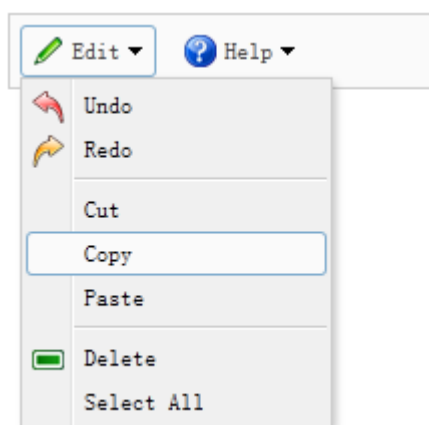
方法

方法名	方法参数	描述
options	none	返回属性对象。
resize	param	重置按钮。 (该方法自 1.4 版开始可用) 代码示例： <pre>\$('#btn').linkbutton('resize', { width: '100%', height: 32 });</pre>
disable	none	禁用按钮。 代码示例： <pre>\$('#btn').linkbutton('disable');</pre>
enable	none	启用按钮。 代码示例： <pre>\$('#btn').linkbutton('enable');</pre>
select	none	选择按钮。 (该方法自 1.3.3 版开始可用)
unselect	none	取消选择按钮。 (该方法自 1.3.3 版开始可用)

MenuButton（菜单按钮）

扩展自\$.fn.linkbutton.defaults。使用\$.fn.menubutton.defaults 重写默认值对象。

菜单按钮是下拉菜单的一部分。它伴随着 linkbutton 和 menu 组件。在用户点击 linkbutton 之前菜单是隐藏的，当用户用鼠标点击或移动到 linkbutton 上面的时候菜单才会显示。



依赖关系

- menu
- linkbutton

用法

通常菜单按钮通过标签创建。

```

1. <a href="javascript:void(0)" id="mb" class="easyui-menubutton"
2.     data-options="menu:'#mm', iconCls:'icon-edit'">Edit</a>
3. <div id="mm" style="width:150px;">
4.     <div data-options="iconCls:'icon-undo'">Undo</div>
5.     <div data-options="iconCls:'icon-redo'">Redo</div>
6.     <div class="menu-sep"></div>
7.     <div>Cut</div>
8.     <div>Copy</div>
9.     <div>Paste</div>
10.    <div class="menu-sep"></div>
11.    <div data-options="iconCls:'icon-remove'">Delete</div>
12.    <div>Select All</div>
13. </div>

```

使用 Javascript 创建菜单按钮。

```

1. <a href="javascript:void(0)" id="mb">Edit</a>
2. <div id="mm" style="width:150px">
3. ...
4. </div>
1. $('#mb').menubutton({
2.     iconCls: 'icon-edit',
3.     menu: '#mm'
4. });

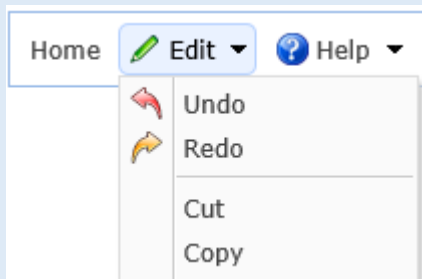
```

属性

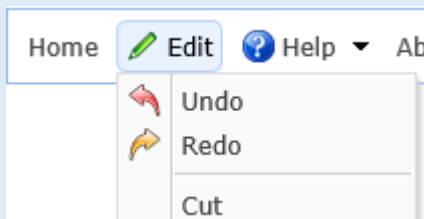
菜单按钮属性扩展自 linkbutton(按钮)，菜单按钮新增的属性如下：

属性名	类型	描述	默认值
plain	boolean	为 true 时显示简易效果。	true
menu	string	用来创建一个对应菜单的选择器。	null
menuAlign	string	允许用户设置顶级菜单对齐方式。可用值有：'left', 'right'。(该属性自 1.3.6 版开始可用)	null
duration	number	定义鼠标划过按钮时显示菜单所持续的时间，单位为毫秒。	100
hasDownArrow	boolean	定义是否显示向下箭头图标。(该属性自 1.4.2 版开始可用)	true

为 true 时：



为 false 时：



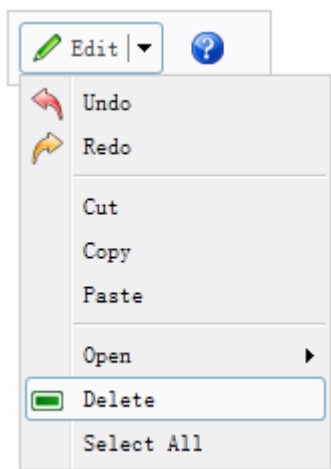
方法

方法名	方法参数	描述
options	none	返回属性对象。
disable	none	禁用菜单按钮。
enable	none	启用菜单按钮。
destroy	none	销毁菜单按钮。

SplitButton (分割按钮)

扩展自\$.fn.linkbutton.defaults。用于\$.fn.splitbutton.defaults 重写默认值对象。

类似菜单按钮，分割按钮也与 linkbutton 和 菜单 有关系。menubutton 和 splitbutton 之间的区别是，splitbutton 分为两部分。它只会在鼠标移动到 splitbutton 按钮右边的时候才会显示出“分割线”。



依赖关系

- menu
- linkbutton

用法

使用标签创建分割按钮。

```

1. <a href="javascript:void(0)" id="sb" class="easyui-splitbutton"
2.     data-options="menu:'#mm', iconCls:'icon-ok'" onclick="javascript:alert('ok')">Ok</a>
3. <div id="mm" style="width:100px;">
4.     <div data-options="iconCls:'icon-ok'">Ok</div>
5.     <div data-options="iconCls:'icon-cancel'">Cancel</div>
6. </div>

```

使用 Javascript 创建分割按钮。

```

1. <a href="javascript:void(0)" id="sb" onclick="javascript:alert('ok')">Ok</a>
2. <div id="mm" style="width:100px;">
3. ...
4. </div>
1. $('#sb').splitbutton({
2.     iconCls: 'icon-ok',
3.     menu: '#mm'
4. });

```

属性

分割按钮属性扩展自 linkbutton，分割按钮新增的属性如下：

属性名	类型	描述	默认值
plain	boolean	设置为 true 将显示简洁效果。	true
menu	string	用来创建一个对应菜单的选择器。	null
duration	number	定义鼠标划过按钮时显示菜单所持续的时间，单位为毫秒。	100

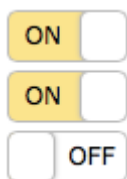
方法

方法名	方法参数	描述
options	none	返回属性对象。
disable	none	禁用分割按钮。 代码示例： <code>\$('#sb').splitbutton('disable');</code>
enable	none	启用分割按钮。 代码示例： <code>\$('#sb').splitbutton('enable');</code>
destroy	none	销毁分割按钮。

SwitchButton（开关按钮）

用于\$.fn.switchbutton.defaults 重写默认值对象。（该组件自 1.4.3 版开始可用）

用在“form”表单中的开关按钮。按钮有 2 个状态：“开”和“关”，用户可以点击或轻敲来切换，标签状态是可定制的。



使用案例

通过标签创建开关按钮。

1. `<input class="easyui-switchbutton" checked>`
2. `<input class="easyui-switchbutton" data-options="onText:'Yes',offText:'No'">`

使用 Javascript 创建开关按钮。

1. `Ok`
2. `<div id="mm" style="width:100px;">`
3. ...
4. `</div>`

属性

属性名	类型	描述	默认值
width	number	开关按钮宽度。	60
height	number	开关按钮高度。	26
handleWidth	number	开关把手宽度。	auto
checked	boolean	定义按钮是否开启。	false
disabled	boolean	定义按钮是否禁用。	false
readonly	boolean	定义按钮是否只读。	false
reversed	boolean	设置为 true 时，反转开关文本。	false
onText	string	左边文本值（反转后是右边）。	ON
offText	string	右边文本值（反转后是左边）。	OFF
handleText	string	开关把手文本值。	''
value	string	给按钮绑定默认值。	on

事件

事件名	事件参数	描述
onChange	checked	在更改控件值的时候触发。

方法

方法名	方法参数	描述
options	none	返回属性对象。
resize	param	调整开关按钮大小。
disable	none	禁用开关按钮。 代码示例： <code>\$('#sb').switchbutton('disable');</code>
enable	none	启用开关按钮。 代码示例： <code>\$('#sb').switchbutton('enable');</code>
readonly	mode	启用/禁用只读模式。
check	none	启用开关按钮。
uncheck	none	禁用开关按钮。
clear	none	清除开关按钮的值。
reset	none	重置开关按钮的值。
setValue	value	设置开关按钮的值。

第四章

Form

(表单)

Form (表单)

使用 `$.fn.form.defaults` 重写默认值对象

`form` 提供了各种方法来操作执行表单字段，比如：`ajax` 提交，`load`，`clear` 等等。当提交表单的时候可以调用 `validate` 方法检查表单是否有效。

《jQuery EasyUI 简体中文 API 文档》

使用案例

创建一个简单的 HTML 表单。构建一个包含 id、action 和 method 值的表单元素。

```

1. <form id="ff" method="post">
2.     <div>
3.         <label for="name">名字:</label>
4.         <input class="easyui-validatebox" type="text" name="name" data-options="required:true" />
5.     </div>
6.     <div>
7.         <label for="email">邮箱:</label>
8.         <input class="easyui-validatebox" type="text" name="email" data-options="validType:'email'" />
9.     </div>
10.    ...
11. </form>

```

使普通表单成为 ajax 提交方式的表单。

```

1. $('#ff').form({
2.     url:...,
3.     onSubmit: function() {
4.         // do some check
5.         // return false to prevent submit;
6.     },
7.     success:function(data) {
8.         alert(data);
9.     }
10. });
11. // submit the form
12. $('#ff').submit();

```

做一个提交操作。

```

1. // call 'submit' method of form plugin to submit the form
2. $('#ff').form('submit', {
3.     url:...,
4.     onSubmit: function() {
5.         // do some check
6.         // return false to prevent submit;
7.     },
8.     success:function(data) {
9.         alert(data)
10.    }
11. });

```

提交额外的参数。

```
1. $('#ff').form('submit', {
2.     url:...,
3.     onSubmit: function(param) {
4.         param.p1 = 'value1';
5.         param.p2 = 'value2';
6.     }
7. });
```

处理提交响应

提交一个 ajax 表单是非常简单的。用户可以在提交完成后获取响应数据。注意，响应的数据是来自服务器的原始数据。A parse action to the response data is required to get the correct data.

例如，响应数据假设为 JSON，一个典型的响应数据格式如下：

```
1. {
2.     "success": true,
3.     "message": "Message sent successfully."
4. }
```

现在在 'success' 回调函数中处理 JSON 字符串。

```
1. $('#ff').form('submit', {
2.     success: function(data) {
3.         var data = eval('(' + data + ')'); //将 JSON 字符串转换为 JS 对象
4.         if (data.success) {
5.             alert(data.message)
6.         }
7.     }
8. });
```

属性

属性名	类型	描述	默认值
novalidate	boolean	定义是否验证表单的字段，true:验证，false:不验证。 (该属性自 1.4 版开始可用)	false
iframe	boolean	定义是否使用 iframe 模式提交表单，true:使用，false:不使用。 (该属性自 1.4.5 版开始可用)	true
ajax	boolean	定义是否使用 ajax 提交表单，true:使用，false:不使用。 (该属性自 1.4 版开始可用)	true

dirty	boolean	定义是否只提交变更字段, true: 是, false: 不是。(该属性自 1.5 版开始可用)	false
queryParams	object	当表单被提交到服务器的时候增加的额外参数列表。(该属性自 1.4 版开始可用)	{}
url	string	提交表单动作的 URL 地址	null

事件

事件名	参数	描述
onSubmit	param	在提交之前触发, 返回 false 可以终止提交。
onProgress	percent	在上传进度数据有效时触发。在 “iframe” 属性设置为 true 时该事件不会被触发。(该事件自 1.4.5 版开始可用)
success	data	在表单提交成功以后触发。
onBeforeLoad	param	在请求加载数据之前触发。返回 false 可以停止该动作。
onLoadSuccess	data	在表单数据加载完成后触发。
onLoadError	none	在表单数据加载出现错误的时候触发。
onChange	target	在表单数据发生变化的时候触发。(该事件自 1.4.2 版开始可用)

方法

方法名	参数	描述
submit	options	<p>执行提交操作, 该选项的参数是一个对象, 它包含以下属性:</p> <ul style="list-style-type: none"> url: 请求的 URL 地址。 onSubmit: 提交之前的回调函数。 success: 提交成功后的回调函数。 <p>下面的例子演示了如何提交一个有效并且避免重复提交的表单。</p> <pre>\$.messenger.progress(); // 显示进度条 \$('#ff').form('submit', { url: ..., onSubmit: function() { var isValid = \$(this).form('validate'); if (!isValid){ // 如果表单无效则隐藏进度条 \$.messenger.progress('close'); } return isValid; // 返回 false 终止表单提交 }, success: function() { \$.messenger.progress('close'); // 如果提交成功则隐藏进度条 } });</pre>

load	data	<p>读取记录填充到表单中。数据参数可以是一个字符串或一个对象类型，如果是字符串则作为远程 URL，否则作为本地记录。</p> <p>代码示例：</p> <pre> \$(' #ff').form(' load', 'get_data.php'); // 读取表单的 URL \$(' #ff').form(' load', { name: 'name2', email: 'mymail@gmail.com', subject: 'subject2', message: 'message2', language: 5 }); </pre>
clear	none	清除表单数据。
reset	none	重置表单数据。 (该方法自 1.3.2 版开始可用)
validate	none	做表单字段验证，当所有字段都有效的时候返回 true。该方法使用 validatebox (验证框) 插件。
enableValidation	none	启用验证。 (该方法自 1.3.4 版开始可用)
disableValidation	none	禁用验证。 (该方法自 1.3.4 版开始可用)
resetValidation	none	重置验证设置为默认值。 (该方法自 1.4.5 版开始可用)
resetDirty	none	重置只提交变更字段的设置为默认值。 (该方法自 1.5 版开始可用)

ValidateBox (验证框)

使用 `$.fn.validatebox.defaults` 重写默认值对象。

validatebox (验证框) 的设计目的是为了验证输入的表单字段是否有效。如果用户输入了无效的值，它将会更改输入框的背景颜色，并且显示警告图标和提示信息。该验证框可以结合 form (表单) 插件并防止表单重复提交。

User Name:

This field is required.

Email:

依赖关系

- tooltip

用法

通过标签创建验证框。

```
1. <input id="vv" class="easyui-validatebox" data-options="required:true,validType:'email'" />
```

使用 Javascript 创建验证框。

```
1. <input id="vv" />
1. $('#vv').validatebox({
2.     required: true,
3.     validType: 'email'
4. });
```

检查密码和确认密码是否相同。

```
1. // extend the 'equals' rule
2. $.extend($.fn.validatebox.defaults.rules, {
3.     equals: {
4.         validator: function(value, param) {
5.             return value == $(param[0]).val();
6.         },
7.         message: 'Field do not match.'
8.     }
9. });
1. <input id="pwd" name="pwd" type="password" class="easyui-validatebox" data-options="required:true" />
2. <input id="rpwd" name="rpwd" type="password" class="easyui-validatebox"
3.     required="required" validType="equals['#pwd']" />
```

验证规则

验证规则是根据使用需求和验证类型属性来定义的，这些规则已经实现：

- email：匹配 E-Mail 的正则表达式规则。
- url：匹配 URL 的正则表达式规则。
- length[0, 100]：允许在 x 到 x 之间个字符。
- remote['http://.../action.do', 'paramName']：发送 ajax 请求需要验证的值，当成功时返回 true。

自定义验证规则，需要重写 \$.fn.validatebox.defaults.rules 中定义的验证器函数和无效消息。例如，定义一个最小长度(minLength)的自定义验证：

```
1. $.extend($.fn.validatebox.defaults.rules, {
2.     minLength: {
3.         validator: function(value, param){
4.             return value.length >= param[0];
5.         },
6.         message: 'Please enter at least {0} characters.'
7.     }
8. });
```

现在你可以在输入框中限制最小长度为 5 的自定义最小长度验证了：

```
1. <input class="easyui-validatebox" data-options="validType:'minLength[5]'">
```

属性

属性名	类型	描述	默认值
required	boolean	定义为必填字段。	false
validType	string, array	定义字段验证类型，比如：email，url 等等。可用值有： <ol style="list-style-type: none"> 1). 一个有效类型字符串运用一个验证规则。 2). 一个有效类型数组运用多个验证规则。(多验证规则验证一个字段在 1.3.2 或更高版本中才可以使用) 代码示例： <pre><input class="easyui-validatebox" data-options="required:true,validType:'url'"> <input class="easyui-validatebox" data-options=" required:true,</pre>	null

		<code>validType:['email','length[0,20]']</code>	
<code>delay</code>	number	延迟到最后验证输入值。(该属性自 1.3.2 版开始可用)	200
<code>missingMessage</code>	string	当文本框未填写时出现的提示信息。	This field is required.
<code>invalidMessage</code>	string	当文本框的内容被验证为无效时出现的提示。	null
<code>tipPosition</code>	string	定义当文本框内容无效的时候提示消息显示的位置，有效的值有：'left','right'。(该属性自 1.3.2 版开始可用)	right
<code>deltaX</code>	number	提示框在水平方向上位移。(该属性自 1.3.3 版开始可用)	0
<code>novalidate</code>	boolean	为 true 时关闭验证功能。(该属性自 1.3.4 版开始可用)	false
<code>editable</code>	boolean	为 true 时用户可以在文本域中输入内容。(该属性自 1.4.5 版开始可用)	true
<code>disabled</code>	boolean	为 true 时禁用验证框（在表单提交时不会被提交）。(该属性自 1.4.5 版开始可用)	false
<code>readonly</code>	boolean	为 true 时将验证框设为只读（在表单提交时会被提交）。(该属性自 1.4.5 版开始可用)	false
<code>validateOnCreate</code>	boolean	为 true 时在创建该组件时就进行验证。(该属性自 1.4.5 版开始可用)	true
<code>validateOnBlur</code>	boolean	为 true 时在该组件失去焦点的时候进行验证。(该属性自 1.4.5 版开始可用)	false

事件

事件名	参数	描述
<code>onBeforeValidate</code>	none	在验证一个字段之前触发。(该事件自 1.4 版开始可用)
<code>onValidate</code>	valid	在验证一个字段的时候触发。(该事件自 1.4 版开始可用)

方法

方法名	方法属性	描述
<code>options</code>	none	返回属性对象。
<code>destroy</code>	none	移除并销毁组件。
<code>validate</code>	none	验证文本框的内容是否有效。
<code>isValid</code>	none	调用 <code>validate</code> 方法并且返回验证结果，true 或者 false。
<code>enableValidation</code>	none	启用验证。(该方法自 1.3.4 版开始可用)
<code>disableValidation</code>	none	禁用验证。(该方法自 1.3.4 版开始可用)
<code>resetValidation</code>	none	重置验证。(该方法自 1.4.5 版开始可用)
<code>enable</code>	none	启用该组件。(该方法自 1.4.5 版开始可用)
<code>disable</code>	none	禁用该组件。(该方法自 1.4.5 版开始可用)

readonly	mode	启用/禁用只读模式。 (该方法自 1.4.5 版开始可用) 代码示例： <pre> \$('#tt').validatebox('readonly'); // 启用只读模式 \$('#tt').validatebox('readonly', true); // 启用只读模式 \$('#tt').validatebox('readonly', false); // 禁用只读模式 </pre>
----------	------	--

TextBox (文本框)

扩展自\$.fn.validatebox.defaults, 使用\$.fn.textbox.defaults 重写默认值对象。 **(该组件自 1.4 版开始可用)**

TextBox (文本框) 是一个增强的输入字段组件, 它允许用户非常简单的创建一组表单。它是一个用于构建其他组合控件的基础组件, 如: combo, databox、spinner 等

依赖关系

- validatebox
- linkbutton

使用案例

通过标签创建文本框。

```
1. <input class="easyui-textbox" data-options="iconCls:'icon-search'" style="width:300px">
```

使用 Javascript 创建文本框。

```
1. <input id="tb" type="text" style="width:300px">
1. $('#tb').textbox({
2.     buttonText:'Search',
3.     iconCls:'icon-man',
4.     iconAlign:'left'
5. })
```

属性

属性名	类型	描述	默认值
width	number	组件的宽度。	auto
height	number	组件的高度。	22
cls	string	允许给组件添加一个自定义 css 类名。 (该属性自 1.5.1 版开始可用)	22
prompt	string	在输入框显示提示消息。	''
value	string	默认值	
type	string	文本框类型。可用值有："text"和"password"。	text
label	string, selector	文本框标签。 (该属性自 1.5 版开始可用) 代码示例: <pre>\$('#email').textbox({ label: 'Email:' labelPosition: 'top', prompt: 'Ent' });</pre> 	null

labelWidth	number	标签宽度。(该属性自 1.5 版开始可用)	auto
labelPosition	string	标签位置。可用值: 'before', 'after', 'top' (该属性自 1.5 版开始可用)	before
labelAlign	string	标签对齐方式。可用值: 'left', 'right' (该属性自 1.5 版开始可用)	left
multiline	boolean	定义是否是多行文本框。	false
editable	boolean	定义用户是否可以直接在该字段内输入文字。	true
disabled	boolean	定义是否禁用该字段。	false
readonly	boolean	定义是否将该控件设为只读。	false
icons	array	<p>在文本框删贡献是图标。每一项都有以下属性:</p> <p>iconCls: 类型 string, 图标的 class 名称;</p> <p>disabled: 类型 boolean, 指明是否禁用该图标;</p> <p>handler: 类型 function, 用于处理点击该图标以后的动作。</p> <p>代码示例:</p> <pre> \$(' #tb').textbox({ icons: [{ iconCls: 'icon-add', handler: function(e) { \$(e.data.target).textbox(' setValue', 'Something added!'); } }, { iconCls: 'icon-remove', handler: function(e) { \$(e.data.target).textbox(' clear'); } }] }) </pre>	[]
iconCls	string	在文本框显示背景图标。	null
iconAlign	string	背景图标的位置。可用值有: "left", "right"。	right
iconWidth	number	图标宽度。	18
buttonText	string	文本框附加按钮显示的文本内容。	
buttonIcon	string	文本框附加按钮显示的图标。	null
buttonAlign	string	附加按钮的位置。可用值有: "left", "right"。	right

事件

事件扩展自 validatebox, 以下是新增的文本框事件。

事件名	参数	描述
onChange	newValue, oldValue	在字段值更改的时候触发。
onResize	width, height	在文本框大小改变的时候触发。
onClickButton	none	在用户点击按钮的时候触发。
onClickIcon	index	在用户点击图标的时候触发。

方法

方法扩展自 validatebox，以下是新增的文本框方法。

方法名	方法属性	描述
options	none	返回属性对象。
textbox	none	返回文本框对象。 代码示例： <pre>var t = \$('#tt'); t.textbox('textbox').bind('keydown', function(e){ if (e.keyCode == 13){ // 当按下回车键时接受输入的值。 t.textbox('setValue', \$(this).val()); } });</pre>
button	none	返回按钮对象。
destroy	none	销毁文本框组件。
resize	width	调整文本框组件宽度。
disable	none	禁用组件。
enable	none	启用组件。
readonly	mode	启用/禁用只读模式。 代码示例： <pre>\$('#tb').textbox('readonly'); // 启用只读模式 \$('#tb').textbox('readonly', true); // 启用只读模式 \$('#tb').textbox('readonly', false); // 禁用只读模式</pre>
clear	none	清除组件中的值。
reset	none	重置组件中的值。
initValue	value	初始化组件值。调用该方法不会触发“onChange”事件。 (该方法自 1.4.5 版开始可用)
setText	text	设置显示的文本值。
getText	none	获取显示的文本值。
setValue	value	设置组件的值。
getValue	none	获取组件的值。
getIcon	index	获取指定图标对象。

PasswordBox (密码框)

扩展自`$.fn.textbox.defaults`，使用`$.fn.passwordbox.defaults` 重写默认值对象。 **(该组件自 1.5 版开始可用)**

该插件允许用户在具有更好交互功能的输入框中输入密码。 密码框会通过显示圆点的方式来保护您输入的密码文本，同时输入框中会提供一个眼睛的图标来通过点击的动作查看您输入的密码，来确保您输入的密码正确无误。



依赖关系

- `textbox`

使用案例

通过标签创建密码框。

```
1. <input class="easyui-passwordbox" prompt="Password" iconWidth="28" style="width:100%;height:34px;padding:10px">
```

使用 Javascript 创建密码框。

```
1. <input id="pb" type="text" style="width:300px">
2. $(function() {
3.     $('#pb').passwordbox({
4.         prompt: 'Password',
5.         showEye: true
6.     });
7. });
```

属性

属性扩展自 `textbox`，下面的属性是密码框添加或者覆盖了文本框的属性。

属性名	类型	描述	默认值
passwordChar	string	在文本框中显示的密码字符。	%u25CF
checkInterval	number	在间隔的时间后检查并转换输入的字符为密码字符。	200
lastDelay	number	在延迟的时间后转换最后输入的字符为密码字符。	500
revealed	boolean	定义是否默认显示真实密码。	false
showEye	boolean	定义是否显示眼睛图标。	true

事件

事件扩展自 `textbox`。

方法

方法扩展自 `textbox`，以下是新增的文本框方法。

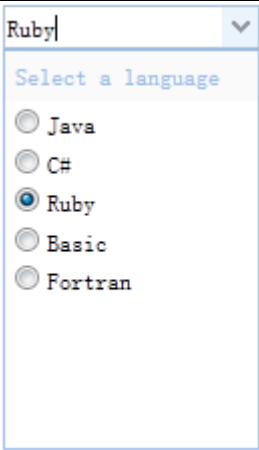
方法名	方法属性	描述
options	none	返回属性对象。
showPassword	none	显示密码。

hidePassword	none	隐藏密码。
--------------	------	-------

Combo（自定义下拉框）

扩展自\$.fn.validatebox.defaults。使用\$.fn.combo.defaults 重写默认值对象。

自定义下拉框显示一个可编辑的文本框和下拉面板在 html 页面。这是构建其他复杂的组合部件（如：combobox, combotree, combogrid 等）之前需要构建的最基本的组件。



依赖关系

- [textbox](#)
- [panel](#)

用法

自定义下拉框使用 Javascript 创建一个<select>或<input>元素。注意：使用自定义下拉框不能通过标签的方式进行创建。

```
1. <input id="cc" value="001">
1. $(' #cc').combo({
2.     required:true,
3.     multiple:true
4. });
```

属性

属性扩展自 validatebox(验证框)，自定义下拉框新增的属性如下。

属性名	类型	描述	默认值
width	number	组件的宽度。	auto
height	number	组件的高度。(该属性自 1.3.2 版开始可用)	22
panelWidth	number	下拉面板宽度。	null
panelHeight	number	下拉面板高度。	200
panelMinWidth	number	下拉面板最小宽度。(该属性自 1.4 版开始可用)	null
panelMaxWidth	number	下拉面板最大宽度。(该属性自 1.4 版开始可用)	null
panelMinHeight	number	下拉面板最小高度。(该属性自 1.4 版开始可用)	null
panelMaxHeight	number	下拉面板最大高度。(该属性自 1.4 版开始可用)	null

t			
panelAlign	string	面板对齐方式。可用值有: 'left', 'right'。 (该属性自 1.3.6 版开始可用)	200
multiple	boolean	定义是否支持多选。	false
multivalue	boolean	定义是否支持多值提交。(该属性自 1.5.1 版开始可用)	true
reversed	boolean	定义在失去焦点的时候是否恢复原始值。 (该属性自 1.5.1 版开始可用)	false
selectOnNavigation	boolean	定义是否允许使用键盘导航来选择项目。 (该属性自 1.3.3 版开始可用)	true
separator	string	在多选的时候使用何种分隔符进行分割。	,
editable	boolean	定义用户是否可以直接输入文本到字段中。	true
disabled	boolean	设置启用/禁用字段。	false
readonly	boolean	设置该字段为读写/只读模式。(该属性自 1.3.3 版开始可用)	false
hasDownArrow	boolean	定义是否显示向下箭头按钮。	true
value	string	字段的默认值。	
delay	number	最后一次输入事件与执行搜索之间的延迟间隔 (执行自动完成功能的延迟间隔)	200
keyHandler	object	在用户按下键的时候调用一个函数。该按键处理器被定义为: keyHandler: { up: function(e) {}, down: function(e) {}, left: function(e) {}, right: function(e) {}, enter: function(e) {}, query: function(q, e) {} }	

事件

事件名	事件属性	描述
onShowPanel	none	当下拉面板显示的时候触发。
onHidePanel	none	当下拉面板隐藏的时候触发。
onChange	newValue, oldValue	当字段值改变的时候触发。

方法

自定义下拉框的方法扩展自 validatebox(验证框)。自定义下拉框新增的方法如下:

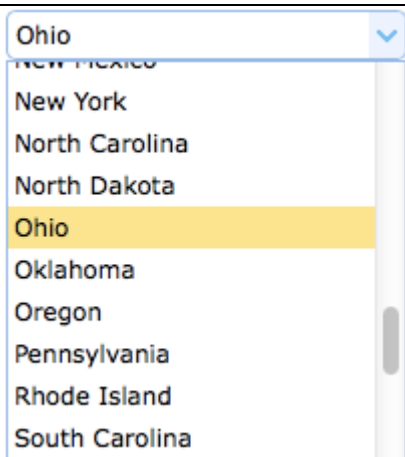
方法名	方法参数	描述
options	none	返回属性对象。
panel	none	返回下拉面板对象。

textbox	none	返回文本框对象。
destroy	none	销毁该组件。
resize	width	调整组件宽度。
showPanel	none	显示下拉面板。
hidePanel	none	隐藏下拉面板。
disable	none	禁用组件。
enable	none	启用组件。
readonly	mode	启用/禁用只读模式。 (该方法自 1.3.3 版开始可用) 使用案例: <pre> \$('#cc').combo('readonly'); // 启用只读模式 \$('#cc').combo('readonly', true); // 启用只读模式 \$('#cc').combo('readonly', false); // 禁用只读模式 </pre>
validate	none	验证输入的值。
isValid	none	返回验证结果。
clear	none	清除控件的值。
reset	none	重置控件的值。 (该方法自 1.3.2 版开始可用)
getText	none	获取输入的文本。
setText	text	设置输入的文本。
getValues	none	获取组件值的数组。
setValues	values	设置组件值的数组。
getValue	none	获取组件的值。
setValue	value	设置组件的值。

ComboBox (下拉列表框)

扩展自\$.fn.combo.defaults。使用\$.fn.combobox.defaults 重写默认值对象。

下拉列表框显示一个可编辑文本框和下拉式列表，用户可以选择一个值或多个值。用户可以直接输入文本到列表顶部或选择一个或多个当前列表中的值。



依赖关系

- combo

使用案例

通过<select>元素创建一个预定义结构的下拉列表框。

```
1. <select id="cc" class="easyui-combobox" name="dept" style="width:200px;">
2.     <option value="aa">aitem1</option>
3.     <option>bitem2</option>
4.     <option>bitem3</option>
5.     <option>ditem4</option>
6.     <option>eitem5</option>
7. </select>
```

通过<input>标签创建下拉列表框。

```
1. <input id="cc" class="easyui-combobox" name="dept"
2.     data-options="valueField:'id',textField:'text',url:'get_data.php' "/>
```

使用 Javascript 创建下拉列表框。

```
1. <input id="cc" name="dept" value="aa">
1. $('#cc').combobox({
2.     url:'combobox_data.json',
3.     valueField:'id',
4.     textField:'text'
5. });
```

创建 2 个有依赖关系的下拉列表框。

```
1. <input id="cc1" class="easyui-combobox" data-options="
2.     valueField: 'id',
3.     textField: 'text',
4.     url: 'get_data1.php',
5.     onSelect: function(rec) {
6.         var url = 'get_data2.php?id=' + rec.id;
7.         $('#cc2').combobox('reload', url);
8.     }" />
9. <input id="cc2" class="easyui-combobox" data-options="valueField:'id',textField:'text'" />
```

JSON 数据格式化例子:

```
1. [{
2.     "id":1,
3.     "text":"text1"
4. }, {
5.     "id":2,
6.     "text":"text2"
7. }, {
8.     "id":3,
9.     "text":"text3",
10.    "selected":true
11. }, {
12.     "id":4,
13.     "text":"text4"
14. }, {
15.     "id":5,
16.     "text":"text5"
17. }]
```

属性

下拉列表框属性扩展自 combo(自定义下拉框)，下拉列表框新增的属性如下：

属性名	类型	描述	默认值
valueField	string	基础数据值名称绑定到该下拉列表框。	value
textField	string	基础数据字段名称绑定到该下拉列表框。	text
groupField	string	指定分组的字段名称（译者注：分组的字段由数据源决定）。 (该属性自 1.3.4 版开始可用)	null
groupFormatter	function (group)	返回格式化后的分组标题文本，以显示分组项（该属性自 1.3.4 版开始可用） 代码示例： \$('#cc').combobox({	

		<pre>groupFormatter: function(group) { return '' + group + ''; } });</pre> 	
mode	string	定义了当文本改变时如何读取列表数据。设置为'remote'时，下拉列表框将会从服务器加载数据。当设置为“remote”模式时，用户输入将被发送到名为'q'的HTTP请求参数到服务器检索新数据。	local
url	string	通过URL加载远程列表数据。	null
method	string	HTTP方法检索数据(POST / GET)。	post
data	array	<p>数据列表加载。</p> <p>代码示例：</p> <pre><input class="easyui-combobox" data-options=" valueField: 'label', textField: 'value', data: [{ label: 'java', value: 'Java' }, { label: 'perl', value: 'Perl' }, { label: 'ruby', value: 'Ruby' }]"/></pre>	null
queryParams	object	<p>在向远程服务器请求数据的时候可以通过该属性像服务器端发送额外的参数。 (该属性自 1.4.2 版开始可用)</p> <p>代码示例：</p> <pre>\$('#cc').combobox({ url: "127.0.0.1/xxx", queryParams: {</pre>	post

		<pre> "age" : 25, "order" : "desc" } }); </pre>	
limitToList	boolean	设置为 true 时，输入的值只能是列表框中的内容。 (该属性自 1.5 版开始可用)	false
showItemIcon	boolean	设置为 true 时，显示选中项的图标。 (该属性自 1.4.5 版开始可用)	false
groupPosition	string	定位分组选项。可用值有：“static”和“sticky”。当设置为“sticky”时会将分组选项标签固顶在下拉栏中。 (该属性自 1.4.5 版开始可用)	static
filter	function	<p>定义当‘mode’设置为‘local’时如何过滤本地数据，函数有 2 个参数：</p> <p>q：用户输入的文本。</p> <p>row：列表行数据。</p> <p>返回 true 的时候允许行显示。</p> <p>代码示例：</p> <pre> \$('#cc').combobox({ filter: function(q, row) { var opts = \$(this).combobox('options'); return row[opts.textField].indexOf(q) == 0; } }); </pre>	
formatter	function	<p>定义如何渲染行。该函数接受 1 个参数：row。</p> <p>代码示例：</p> <pre> \$('#cc').combobox({ formatter: function(row) { var opts = \$(this).combobox('options'); return row[opts.textField]; } }); </pre>	
loader	function (param, success, error)	<p>定义了如何从远程服务器加载数据。返回 false 可以忽略该动作。该函数具备如下参数：</p> <p>param：传递到远程服务器的参数对象。</p> <p>success(data)：在检索数据成功的时候调用该回调函数。</p> <p>error()：在检索数据失败的时候调用该回调函数。</p>	json loader
loadFilter	function (data)	返回过滤后的数据并显示。 (该属性自 1.3.3 版开始可用)	

事件

下拉列表框事件继承自 combo(自定义下拉框)，下拉列表框新增的事件如下：

事件名	事件参数	描述
onBeforeLoad	param	<p>在请求加载数据之前触发，返回 false 取消该加载动作。</p> <p>代码示例：</p> <pre>// 在加载服务器数据之前改变 http 请求参数的值 \$('#cc').combobox({ onBeforeLoad: function(param) { param.id = 2; param.language = 'js'; } });</pre>
onLoadSuccess	none	在加载远程数据成功的时候触发。
onLoadError	none	在加载远程数据失败的时候触发。
onChange	newValue, oldValue	在字段值发生更改的时候触发。 (该事件自 1.4.5 版开始可用)
onClick	record	在用户点击列表项的时候触发。 (该事件自 1.5.1 版开始可用)
onSelect	record	在用户选择列表项的时候触发。
onUnselect	record	在用户取消选择列表项的时候触发。

方法

下拉列表框扩展自 combo(自定义下拉框)，下拉列表框新增或重写的方法如下：

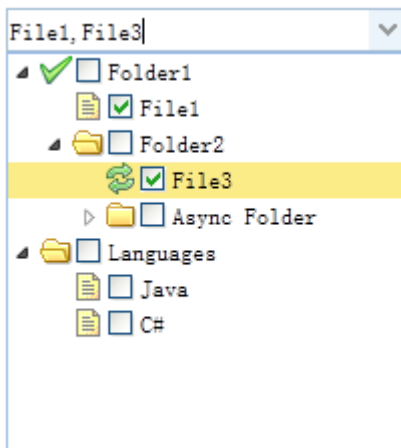
方法名	方法参数	描述
options	none	返回属性对象。
getData	none	返回加载数据。
loadData	data	读取本地列表数据。
reload	url	<p>请求远程列表数据。通过 'url' 参数重写原始 URL 值。</p> <p>代码示例：</p> <pre>\$('#cc').combobox('reload'); // 使用旧的 URL 重新载入列表数据 \$('#cc').combobox('reload', 'get_data.php'); // 使用新的 URL 重新载入列表数据</pre>
setValues	values	<p>设置下拉列表框值数组。</p> <p>代码示例：</p> <pre>\$('#cc').combobox('setValues', ['001', '002']);</pre>
setValue	value	设置下拉列表框的值。

		代码示例： <code>\$('#cc').combobox('setValue', '001');</code>
clear	none	清除下拉列表框的值。
select	value	选择指定项。
unselect	value	取消选择指定项。

ComboTree（树形下拉框）

扩展自\$.fn.combo.defaults 和\$.fn.tree.defaults。使用\$.fn.combotree.defaults 重写默认值对象。

树形下拉框结合选择控件和下拉树控件。它与 combobox(下拉列表框)类似,但是将下拉列表框的列表替换成了树形控件。该控件支持树状态复选框,方便多选操作。



依赖关系

- combo
- tree

使用案例

使用标签创建树形下拉框。

```
1. <select id="cc" class="easyui-combotree" style="width:200px;"
2.     data-options="url:'get_data.php',required:true"></select>
```

使用 Javascript 创建树形下拉框。

```
1. <input id="cc" value="01">
1. $('#cc').combotree({
2.     url: 'get_data.php',
3.     required: true
4. });
```

属性

树形下拉框属性扩展自 combo(自定义下拉框)和 tree(树形控件),树形下拉框重写的属性如下:

属性名	类型	描述	默认值
editable	boolean	定义用户是否可以直接输入文本到字段中。	false

事件

该控件的事件完全继承自 combo(自定义下拉框)和 tree(树形控件)。

方法

树形下拉框方法扩展自 combo(自定义下拉框)，树形下拉框新增和重写的方法如下：

方法名	方法参数	描述
options	none	返回属性对象。
tree	none	返回树形对象。以下的例子显示了如何得到选择的树节点。 <pre>var t = \$('#cc').combotree('tree'); // 获取树对象 var n = t.tree('getSelected'); // 获取选择的节点 alert(n.text);</pre>
loadData	data	读取本地树形数据。 代码示例： <pre>\$('#cc').combotree('loadData', [{ id: 1, text: 'Languages', children: [{ id: 11, text: 'Java' }, { id: 12, text: 'C++' }] }]);</pre>
reload	url	再次请求远程树数据。通过' url' 参数重写原始 URL 值。
clear	none	清空控件的值。
setValues	values	设置组件值。值可以是节点的“id”值或“id”和“text”键值对。 (该方法自 1.4.5 版开始可用，之前的旧方法[单值参数]自 1.4.5 开始作废) 代码示例： <pre>\$('#cc').combotree('setValue', 6); // set value with {id,text} pairs \$('#cc').combotree('setValue', { id: 61, text: 'text61' });</pre>
setValue	value	设置组件值数组。 (该方法自 1.4.5 版开始可用，之前的旧方法[单值参数]自

1.4.5 开始作废)

代码示例:

```
$('#cc').combotree('setValues', [1, 3, 21]);  
$('#cc').combotree('setValues', [1, 3, 21, {id:73, text:'text73'}]);
```

ComboGrid (数据表格下拉框)

扩展自\$.fn.combo.defaults 和\$.fn.datagrid.defaults。使用\$.fn.combogrid.defaults 重写默认值对象。

数据表格下拉框结合了可编辑文本框控件和下拉数据表格面板控件，该控件允许用户快速查找和选择，并且该控件提供了键盘导航支持，对行进行筛选。

Name 6			
Code	Name	Address	Col41
001	Name 1	Address 11	col4 data
002	Name 2	Address 13	col4 data
003	Name 3	Address 87	col4 data
004	Name 4	Address 63	col4 data
005	Name 5	Address 45	col4 data
006	Name 6	Address 16	col4 data
007	Name 7	Address 27	col4 data
008	Name 8	Address 81	col4 data

依赖关系

- combo
- datagrid

使用案例

创建数据表格下拉框

1. 使用标签创建一个数据表格下拉框。

```

1. <select id="cc" class="easyui-combogrid" name="dept" style="width:250px;"
2.     data-options="
3.         panelWidth:450,
4.         value:'006',
5.         idField:'code',
6.         textField:'name',
7.         url:'datagrid_data.json',
8.         columns:[[
9.             {field:'code',title:'Code',width:60},
10.            {field:'name',title:'Name',width:100},
11.            {field:'addr',title:'Address',width:120},
12.            {field:'col4',title:'Col41',width:100}
13.        ]]
14.     "></select>

```

2. 使用 Javascript 通过已经定义的<select>或<input>标签来创建数据表格下拉框。

```

1. <input id="cc" name="dept" value="01" />
1. $(' #cc').combogrid({
2.     panelWidth:450,
3.     value:'006',
4.     idField:'code',

```

《jQuery EasyUI 简体中文 API 文档》


```

5.     textField:'name',
6.     url:'datagrid_data.json',
7.     columns:[[
8.         {field:'code',title:'Code',width:60},
9.         {field:'name',title:'Name',width:100},
10.        {field:'addr',title:'Address',width:120},
11.        {field:'col4',title:'Col41',width:100}
12.    ]]
13. });

```

自动完成功能

让我们为数据表格下拉框控件添加高级的自动完成功能。下拉数据表格会根据用户输入显示适合的结果。

```

1.  $('#cc').combogrid({
2.     delay: 500,
3.     mode: 'remote',
4.     url: 'get_data.php',
5.     idField: 'id',
6.     textField: 'name',
7.     columns: [[
8.         {field:'code',title:'Code',width:120,sortable:true},
9.         {field:'name',title:'Name',width:400,sortable:true}
10.    ]]
11. });

```

在服务器端，参数'q'必须先检索。用户可以查询数据库，然后返回一个 SQL 查询结果的 JSON 格式给浏览器。

get_data.php:

```

1.  $q = isset($_POST['q']) ? $_POST['q'] : ''; // the request parameter
2.  // query database and return JSON result data
3.  $rs = mysql_query("select * from item where name like '$q%'");
4.  echo json_encode(...);

```

属性

数据表格下拉框的属性扩展自 combo(自定义下拉框)和 datagrid(数据表格)，树形下拉框新增属性如下：

属性名	类型	描述	默认值
loadMsg	string	在数据表格加载远程数据的时候显示消息。	null

idField	string	ID 字段名称。	null
textField	string	要显示在文本框中的文本字段。	null
mode	string	定义在文本改变的时候如何读取数据网格数据。设置为 'remote'，数据表格将从远程服务器加载数据。 当设置为 'remote' 模式的时候，用户输入将会发送到名为 'q' 的 http 请求参数，向服务器检索新的数据。	local
filter	function(q, row)	定义在 'mode' 设置为 'local' 的时候如何选择本地数据，返回 true 时则选择该行。 代码示例： <pre> \$('#cc').combogrid({ filter: function(q, row) { var opts = \$(this).combogrid('options'); return row[opts.textField].indexOf(q) == 0; } }); </pre>	

事件

数据表格下拉框事件完全扩展自 combo(自定义下拉框)和 datagrid(数据表格)。

方法

数据表格下拉框的方法扩展自 combo(自定义下拉框)，数据表格下拉框新增或重写的方法如下：

方法名	方法参数	描述
options	none	返回属性对象。
grid	none	返回数据表格对象。下面的例子显示了如何获取选择的行： <pre> var g = \$('#cc').combogrid('grid'); // 获取数据表格对象 var r = g.datagrid('getSelected'); // 获取选择的行 alert(r.name); </pre>
setValues	values	设置组件值数组。 代码示例： <pre> \$('#cc').combogrid('setValues', ['001', '007']); \$('#cc').combogrid('setValues', '001', '007', {id: '008', name: 'name008'}]); </pre>
setValue	value	设置组件值。 代码示例： <pre> \$('#cc').combogrid('setValue', '002'); \$('#cc').combogrid('setValue', {id: '003', name: 'name003'}]); </pre>

clear	none	清除组件的值。
-------	------	---------

ComboTreeGrid（树形表格下拉框）

扩展自\$.fn.combo.defaults 和\$.fn.treegrid.defaults。使用\$.fn.combotreegrid.defaults 重写默认值对象。（该组件自 1.5 版开始可用）

树形表格下拉框结合了可编辑文本框控件和下拉树形表格面板控件，该控件允许用户快速从树形表格中选择一条或多条记录。

eclipse.exe			
Name	Size	Date	
▲ 文件夹 C		02/19/2010	
▲ 文件夹 Program Files	120 MB	03/20/2010	
▶ 文件夹 Java		01/13/2010	
▶ 文件夹 MySQL		01/13/2010	
▲ 文件夹 eclipse		01/20/2010	
文件 eclipse.exe	56 KB	05/19/2009	
文件 eclipse.ini	1 KB	04/20/2010	

依赖关系

- combo
- treegrid

使用案例

创建树形表格下拉框

1. 使用标签创建一个数据表格下拉框。

```

1. <input class="easyui-combotreegrid" data-options="
2.     width:'100%',
3.     panelWidth:500,
4.     label:'Select Item:',
5.     labelPosition:'top',
6.     url:'treegrid_data1.json',
7.     idField:'id',
8.     treeField:'name',
9.     columns:[[
10.         {field:'name',title:'Name',width:200},
11.         {field:'size',title:'Size',width:100},
12.         {field:'date',title:'Date',width:100}
13.     ]]">

```

2. 使用 Javascript 通过已经定义的<select>或<input>标签来创建数据表格下拉框。

```

1. <input id="cc" name="name">
2. $(function() {
3.     $('#cc').combogrid({
4.         value:'006',
5.         width:'100%',
6.         panelWidth:500,

```

```

7.      label:'Select Item:',
8.      labelPosition:' top',
9.      url:'treegrid_data1.json',
10.     idField:' id',
11.     treeField:' name',
12.     columns:[[
13.         {field:' name',title:' Name',width:200},
14.         {field:' size',title:' Size',width:100},
15.         {field:' date',title:' Date',width:100}
16.     ]]
17. });
18. });

```

属性

树形表格下拉框的属性扩展自 `combo` (自定义下拉框) 和 `treegrid` (树形表格), `combotreegrid` 新增属性如下:

属性名	类型	描述	默认值
idField	string	ID 字段名称。	null
textField	string	要显示在文本框中的文本字段。	null
limitToGrid	boolean	限制输入的值只能是树形表格中的值。	false

事件

数据表格下拉框事件完全扩展自 `combo` (自定义下拉框) 和 `datagrid` (数据表格)。

方法

数据表格下拉框的方法扩展自 `combo` (自定义下拉框), 数据表格下拉框新增或重写的方法如下:

方法名	方法参数	描述
options	none	返回属性对象。
grid	none	返回数据表格对象。下面的例子显示了如何获取选择的行: <pre>var g = \$('#cc').combogrid('grid'); // 获取数据表格对象 var r = g.datagrid('getSelected'); // 获取选择的行 alert(r.name);</pre>
setValue	value	设置组件值。

		代码示例： \$('#cc').combogrid('setValue', '002'); \$('#cc').combogrid('setValue', { id: '003', name: 'name003' });
setValues	values	设置组件值数组。 代码示例： \$('#cc').combogrid('setValues', ['001', '007']); \$('#cc').combogrid('setValues', ['001', '007', { id: '008', name: 'name008' }]);
clear	none	清除组件的值。

TagBox（文本框）

扩展自\$.fn.combobox.defaults，使用\$.fn.tagbox.defaults 重写默认值对象。（该组件自 1.5.1 版开始可用）

TagBox(标签框)扩展自 combobox， 他包含 combobox 的所有功能。TagBox 允许用户将输入内容显示在标签框中，而不是显示在输入框中。

Ruby x Java x Select a Language v

依赖关系

- combobox

使用案例

通过标签创建文本框。

```
1. <input class="easyui-tagbox" value="Apple, Orange" label="Add a tag" style="width:100%">
```

使用 Javascript 创建文本框。

```
1. <input id="tb" type="text" style="width:300px">
```

```
1. $('#tb').tagbox({
2.     label:'添加一个标签',
3.     value:['苹果', '橘子']
4. })
```

属性

标签框属性扩展自 combobox(标签框)，标签框重写的属性如下：

属性名	类型	描述	默认值
hasDownArrow	boolean	定义是否显示向下的箭头	false
tagFormatter	function(value, row)	<p>该格式化器用于格式化返回值。</p> <p>代码示例：</p> <pre>\$('#tb').tagbox({ tagFormatter: function(value, row) { var opts = \$(this).tagbox('options'); return row ? row[opts.textField] : value; } });</pre>	

tagStyler	function(value, row)	<p>该格式化器用于格式化标签样式，返回自定义标签样式字符串。</p> <p>代码示例：</p> <pre> \$('#tb').tagbox({ tagStyler: function(value) { if (value == 3) { return 'background:#ffd7d7; color:#c65353'; } else if (value == 4) { return 'background:#b8eecf; color:#45872c'; } } }); </pre>	
------------------	----------------------	--	--

事件

事件扩展自 combobox，以下是新增的文本框事件。

事件名	参数	描述
onClickTag	value	在点击标签框的时候触发该事件。
onBeforeRemoveTag	value	在移除一个标签框之前触发，返回 false 则取消移除操作。
onRemoveTag	value	在移除标签框时触发。

方法

方法扩展自 combobox。

NumberBox（数值输入框）

扩展自 \$.fn.validatebox.defaults。使用 \$.fn.numberbox.defaults 重写默认值对象。

数值输入框是用来限制用户只能输入数值型数据的。他可以转换一个输入的元素到其他类型，比如：数字、百分比、货币等。更多的输入类型定义依赖于'formatter'和'parser'函数。

依赖关系

- `textbox`

用法

使用标签创建数值输入框。

```
1. <input type="text" class="easyui-numberbox" value="100" data-options="min:0,precision:2"></input>
```

使用 Javascript 创建数值输入框。

```
1. <input type="text" id="nn"></input>
1. $('#nn').numberbox({
2.     min:0,
3.     precision:2
4. });
```

属性

数值输入框的属性扩展自 validatebox(验证框)，数值输入框新增的属性如下：

属性名	类型	描述	默认值
disabled	boolean	是否禁用该字段。	false
value	number	默认值。	
min	number	允许的最小值。	null
max	number	允许的最大值。	null
precision	number	在十进制分隔符之后显示的最大精度。(即小数点后的显示精度)	0
decimalSeparator	string	使用哪一种十进制字符分隔数字的整数和小数部分。	.

groupSeparator	string	使用哪一种字符分割整数组，以显示成千上万的数据。（比如：99,999,999.00 中的','就是该分隔符设置。）	
prefix	string	前缀字符。（比如：金额的\$或者¥）	
suffix	string	后缀字符。（比如：后置的欧元符号€）	
filter	function(e)	定义如何过滤按键，当返回 true 时则允许输入，反之禁止。 (该属性自 1.3.3 版开始可用)	
formatter	function(value)	用于格式化数值的函数。返回字符串值以显示到输入框中。	
parser	function(s)	用于解析字符串的函数。返回数值。	

事件

事件名	事件参数	描述
onChange	newValue, oldValue	当字段值更改的时候触发。

方法

数值输入框的方法扩展自 validatebox(验证框)，数值输入框新增或重写的方法如下：

方法名	方法参数	描述
options	none	返回数值输入框属性。
destroy	none	销毁数值输入框对象。
disable	none	禁用字段。
enable	none	启用字段。
fix	none	将输入框中的值修正为有效的值。
setValue	value	设置数值输入框的值。 代码示例： \$('#nn').numberbox('setValue', 206.12);
getValue	none	获取数值输入框的值。 代码示例： var v = \$('#nn').numberbox('getValue'); alert(v);
clear	none	清楚数值输入框的值。
reset	none	重置数值输入框的值。(该方法自 1.3.2 版开始可用)

DateBox（日期输入框）

扩展自\$.fn.combo.defaults。使用\$.fn.datebox.defaults 重写默认值对象。

日期输入框结合了一个可编辑的文本框控件和允许用户选择日期的下拉日历面板控件。选择的日期会自动转变为一个有效的日期然后填充到文本框中。选定的日期也可以被格式化为预定格式。



依赖关系

- combo
- calendar

使用案例

使用标签创建日期输入框。

```
1. <input id="dd" type="text" class="easyui-datebox" required="required"> </input>
```

使用 Javascript 创建日期输入框。

```
1. <input id="dd" type="text"></input>
1. $('#dd').datebox({
2.     required:true
3. });
```

属性

日期输入框扩展自 combo (自定义下拉框)，日期输入框新增的属性如下：

属性名	类型	描述	默认值
panelWidth	number	下拉日历面板宽度。	180
panelHeight	number	下拉日历面板高度。	auto
currentText	string	显示当天按钮。	Today
closeText	string	显示关闭按钮。	Close
okText	string	显示 OK 按钮。	Ok
disabled	boolean	该属性值为 true 时禁用该字段。	false
buttons	array	<p>在日历下面的按钮。(该属性自 1.3.5 版开始可用)</p> <p>代码示例:</p> <pre>var buttons = \$.extend([], \$.fn.datebox.defaults.buttons); buttons.splice(1, 0, { text: 'MyBtn', handler: function(target){ alert('click MyBtn'); } }); \$('#dd').datebox({ buttons: buttons });</pre>	
sharedCalendar	string, selector	<p>将一个日历控件共享给多个 datebox 控件使用。(该属性自 1.3.5 版开始可用)</p> <p>代码示例:</p> <pre><input class="easyui-datebox" sharedCalendar="#sc"> <input class="easyui-datebox" sharedCalendar="#sc"> <div id="sc" class="easyui-calendar"></div></pre>	null
formatter	function	<p>该函数用于格式化日期, 它有一个 'date' 参数并且会返回一个字符串类型的值。下面的一个例子展示了如何重写默认的 formatter 函数。</p> <pre>\$.fn.datebox.defaults.formatter = function(date) { var y = date.getFullYear(); var m = date.getMonth()+1; var d = date.getDate(); return m+'/' +d+'/' +y; }</pre>	
parser	function	<p>该函数用于解析一个日期字符串, 它有一个 'date' 字符串参数并且会返回一个日期类型的值。下面的一个例子展示了如何重写默认的 parser 函数。</p> <pre>\$.fn.datebox.defaults.parser = function(s) { var t = Date.parse(s); if (!isNaN(t)) {</pre>	

		<pre> return new Date(t); } else { return new Date(); } } </pre>	
--	--	--	--

事件

事件名	事件参数	描述
onSelect	date	<p>当用户选择一个日期的时候触发。</p> <p>代码示例：</p> <pre> \$('#dd').datebox({ onSelect: function(date) { alert(date.getFullYear()+"-"+ (date.getMonth()+1)+"-"+date.getDate()); } }); </pre>

方法

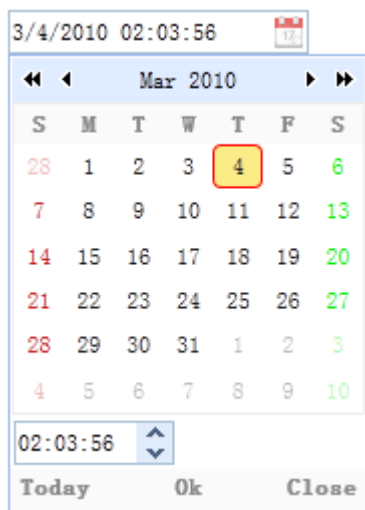
方法扩展自 `combo` (自定义下拉框)，日期输入框重写的方法如下：

方法名	方法参数	描述
options	none	返回属性对象。
calendar	none	<p>获取日历对象。下面的例子显示了如果获取日历对象并重新创建它。</p> <pre> // 获取日历对象 var c = \$('#dd').datebox('calendar'); // 设置一周的第一天是星期几 (0 是周日, 1 是周一) c.calendar({ firstDay: 1 }); </pre>
setValue	value	<p>设置日期输入框的值。</p> <p>代码示例：</p> <pre> \$('#dd').datebox('setValue', '6/1/2012'); // 设置日期输入框的值 var v = \$('#dd').datebox('getValue'); // 获取日期输入框的值 </pre>
cloneFrom	from	<p>克隆一个 <code>datebox</code> 控件。(该方法自 1.4.1 版开始可用)</p> <p>代码示例：</p> <pre> <input id="from" class="easyui-datebox"> // 克隆一个存在的 datebox 组件 \$('#.dt').datebox('cloneFrom', '#from'); </pre>

DateTimeBox（日期时间输入框）

扩展自\$.fn.datebox.defaults，使用\$.fn.datetimebox.defaults 重写默认值对象。

和日期输入框类似，日期时间输入框允许用户选择日期和指定的时间并按照指定的输出格式显示。相比日期输入框，它在下拉面板中添加了一个时间微调器。



依赖关系

- datebox
- timespinner

使用案例

使用标签创建日期时间输入框。

```
1. <input class="easyui-datetimebox" name="birthday"
2.     data-options="required:true,showSeconds:false" value="3/4/2010 2:3" style="width:150px">
```

使用 Javascript 创建日期时间输入框。

```
1. <input id="dt" type="text" name="birthday"></input>
1. $('#dt').datetimebox(
2.     value: '3/4/2010 2:3',
3.     required: true,
4.     showSeconds: false
5. );
```

属性

日期时间输入框扩展自 datebox (日期输入框)，日期时间输入框新增的属性如下：

属性名	类型	描述	默认值
-----	----	----	-----

currentText	string	文本显示为当前天按钮。 (该属性自 1.4 版开始可用)	Today
closeText	string	文本显示为关闭按钮。 (该属性自 1.4 版开始可用)	Close
okText	string	文本显示为确定按钮。 (该属性自 1.4 版开始可用)	Ok
spinnerWidth	number	定义 datetimetype 组件嵌入的时间微调器的宽度。	100%
showSeconds	boolean	定义是否显示秒钟信息。	true
timeSeparator	string	定义在小时、分钟和秒之间的时间分割字符。 (该属性自 1.3 版开始可用)	:

方法

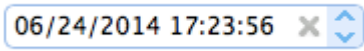
日期时间输入框的方法扩展自 datebox(日期输入框)，日期时间输入框重写的方法如下：

方法名	方法参数	描述
options	none	返回属性对象。
spinner	none	返回时间微调器对象。
setValue	value	<p>设置日期时间输入框值。</p> <p>代码示例：</p> <pre>\$('#dt').datetimebox('setValue', '6/1/2012 12:30:56'); // 设置日期时间输入框的值 var v = \$('#dt').datetimebox('getValue'); // 获取日期时间输入框的值 alert(v);</pre>
cloneFrom	from	<p>克隆一个 datetimebox 控件。(该方法自 1.4.1 版开始可用)</p> <p>代码示例：</p> <pre><input id="from" class="easyui-datetimebox"></pre> <p>// 克隆一个存在的 datebox 组件</p> <pre>\$('#dt').datetimebox('cloneFrom', '#from');</pre>

DateTimeSpinner（日期时间微调框）

扩展自 \$.fn.timespinner.defaults，使用 \$.fn.datetimespinner.defaults 重写默认值对象。**(该组件自 1.4 版开始可用)**

DateTimeSpinner (日期时间微调框)扩展自 timespinner(时间微调框)，它允许用户使用微调按钮调整指定的字段。



依赖关系

- timespinner

使用案例

通过标签创建验证框。

```
1. <input class="easyui-datetimepicker" style="width:300px">
```

使用 Javascript 创建验证框。

```
1. <input id="dt" type="text" style="width:300px">
1. $('#dt').datetimepicker({
2.     //...
3. })
```

属性

属性扩展自 timespinner，以下是新增的日期时间微调框属性。

属性名	类型	描述	默认值
selections	array	选择高亮部分的值 (该值必须设置正确，否则微调的时候会出问题)。	[[0, 2], [3, 5], [6, 10], [11, 13], [14, 16], [17, 19]]

事件

事件扩展自 timespinner。

方法

方法扩展自 `timespinner`。

Calendar（日历）

使用 `$.fn.calendar.defaults` 重写默认值对象。

日历控件显示一个月的日历，允许用户选择日期和移动到下一个或上一个月。默认情况下，一周的第一天是周日。它可以通过设置 `firstDay` 属性的值来更改设置。

八月 2016							
周数	日	一	二	三	四	五	六
30	31	1	2	3	4	5	6
31	7	8	9	10	11	12	13
32	14	15	16	17	18	19	20
33	21	22	23	24	25	26	27
34	28	29	30	31	1	2	3
35	4	5	6	7	8	9	10

使用案例

使用标签创建日历。

```
1. <div id="cc" class="easyui-calendar" style="width:180px;height:180px;"></div>
```

使用 Javascript 创建日历。

```
1. <div id="cc" style="width:180px;height:180px;"></div>
```

```
1. $('#cc').calendar({
2.     current:new Date()
3. });
```

属性

属性名	类型	描述	默认值
width	number	日历控件宽度。	180
height	number	日历控件高度。	180
fit	boolean	当设置为 true 的，将设置日历控件大小自适应父容器。	false
border	boolean	定义是否显示边框。	true
showWeek	boolean	当设置为 true 时，将显示周数。 (该属性自 1.5 版开始可用)	false
weekNumberHeader	string	周数的标签显示在头部。 (该属性自 1.5 版开始可用)	
getWeekNumber	function (date)	该函数用于返回周数值。 (该属性自 1.5 版开始可用)	
firstDay	number	定义一周的第一天是星期几。0=星期日、1=星期一 等。	0
weeks	array	显示的周列表内容。	['S', 'M', 'T', 'W', 'T', 'F', 'S']

]
months	array	显示的月列表内容。	[' Jan', ' Feb', ' Mar', ' Apr', ' May', ' Jun', ' Jul', ' Aug', ' Sep', ' Oct', ' Nov', ' Dec']
year	number	年日历。下面的例子显示了如何使用指定的年份和月份创建一个日历。 <div class="easyui-calendar" data-options="year:2012,month:6" />	当前年份（4 位数）
month	number	月日历。	当前月份（从 1 开始）
current	Date	当前日期。	当前日期
formatter	function (date)	日期格式化函数，返回日期值。 （该属性自 1.3.6 版开始可用） 代码示例： <pre> \$('#cc').calendar({ formatter: function(date) { return date.getDate(); } }) </pre>	
styler	function (date)	日历天的样式函数，返回行内样式或 CSS 样式表的 Class 名称。 （该属性自 1.3.6 版开始可用） 代码示例： <pre> \$('#cc').calendar({ styler: function(date) { if (date.getDay() == 1) { return 'background-color:#ccc'; // 函数可以返回预定义的 // css class 和预定义的行内样式 // return {class:'r1', style:{'color:#fff'}}; } else { return ''; } } }) </pre>	
validator	function (date)	验证器函数用于确定是否可以选择日历上的某一天，返回 false 将阻止选择当前天。 （该属性自 1.3.6 版开始可用）	

		<p>代码示例:</p> <pre> \$('#cc').calendar({ validator: function(date) { if (date.getDay() == 1) { return true; } else { return false; } } }) </pre>	
--	--	---	--

事件

事件名	事件参数	描述
onSelect	date	<p>在用户选择一天的时候触发。</p> <p>代码示例:</p> <pre> \$('#cc').calendar({ onSelect: function(date) { alert(date.getFullYear()+"-"+ +(date.getMonth()+1)+"-"+date.getDate()); } }); </pre>
onChange	newDate, oldDate	<p>在用户更改日期的时候触发。(该事件自 1.3.6 版开始可用)</p>

方法

方法名称	方法参数	描述
options	none	返回参数对象。
resize	none	调整日历大小。
moveTo	date	<p>移动日历到指定日期。</p> <p>代码示例:</p> <pre> \$('#cc').calendar('moveTo', new Date(2012, 6, 1)); </pre>

Spinner (微调)

扩展自\$.fn.validatebox.defaults。使用\$.fn.spinner.defaults 重写默认值对象。

微调控件结合了一个可编辑文本框和 2 个小按钮让用户选择一个值的范围。和下拉列表框类似，微调控件允许用户输入值，但是没有下拉列表。微调控件是创建其他高级微调控件的基础控件，比如：numberspinner, timespinner 等。

依赖关系

- textbox

使用案例

只能使用 Javascript 创建微调控件。标签创建是无效的。

```
1. <input id="ss" value="2">
1. $('#ss').spinner({
2.     required:true,
3.     increment:10
4. });
```

属性

微调控件的属性扩展自 validatebox(验证框)，微调控件新增或重写的属性如下：

属性名	类型	描述	默认值
width	number	组件宽度。	auto
height	number	组件高度。 (该属性自 1.3.2 版开始可用)	22
value	string	默认值。	
min	string	允许的最小值。	null
max	string	允许的最大值。	null
increment	number	在点击微调按钮的时候的增量值。	1
editable	boolean	定义用户是否可以直接输入值到字段。	true
disabled	boolean	定义是否禁用字段。	false
readonly	boolean	定义控件是否为只读。 (该属性自 1.3.6 版开始可用)	false
spinAlign	string	定义控件的对齐方式。可用值：'left', 'right', 'horizontal', 'vertical' (该属性自 1.5 版开始可用)	right
spin	function (down)	在用户点击微调按钮的时候调用的函数。'down' 参数对应用户点击的向下按钮。	

事件

事件名	事件参数	描述
onSpinUp	none	在用户点击向上微调按钮的时候触发。
onSpinDown	none	在用户点击向下微调按钮的时候触发。

方法

微调控件的方法扩展自 validatebox (验证框)，微调控件新增的方法如下：

方法名	方法参数	描述
options	none	返回属性对象。
destroy	none	销毁微调组件。
resize	width	返回组件宽度。通过'width'参数重写原始宽度。 代码示例： <pre>\$('#ss').spinner('resize'); // 调整到原始宽度 \$('#ss').spinner('resize', 200); // 调整到新宽度</pre>
enable	none	启用组件。
disable	none	禁用组件。
getValue	none	获取组件值。
setValue	value	设置组件值。
readonly	mode	启用/禁用只读模式。 (该方法自 1.3.6 版开始可用) 代码示例： <pre>\$('#ss').spinner('readonly'); // 启用只读模式 \$('#ss').spinner('readonly', true); // 启用只读模式 \$('#ss').spinner('readonly', false); // 禁用只读模式</pre>
clear	none	清空组件值。
reset	none	重置组件值。 (该方法自 1.3.2 版开始可用)

NumberSpinner（数字微调）

扩展自\$.fn.spinner.defaults和\$.fn.numberbox.defaults。使用\$.fn.numberspinner.defaults重写默认值对象。

数字微调控件的创建是基于微调控件和数值输入框控件的。他可以转换输入的值，比如：数值、百分比、货币等。它也允许使用上/下微调按钮调整到用户的期望值。



依赖关系

- spinner
- numberbox

使用案例

使用标签创建数字微调组件。

```
1. <input id="ss" class="easyui-numberspinner" style="width:80px;"
2.     required="required" data-options="min:10,max:100,editable:false">
```

使用 Javascript 创建数字微调组件。

```
1. <input id="ss" required="required" style="width:80px;">
1. $('#ss').numberspinner({
2.     min: 10,
3.     max: 100,
4.     editable: false
5. });
```

创建数字微调组件并将数值格式化为货币字符串。

```
1. <input class="easyui-numberspinner" value="1234567890" style="width:150px;"
2.     data-options="required:true,precision:2,groupSeparator:',',decimalSeparator:'.',prefix:'$'"/>
```

属性

本组件的属性完整继承自 [spinner](#) (微调) 和 [numberbox](#) (数值输入框)。

事件

本组件的事件完整继承自 [spinner](#) (微调)。

方法

本组件的方法扩展自 [spinner](#) (微调)，本组件新增的方法如下：

方法名称	方法参数	描述
options	none	返回属性对象。
setValue	value	设置数字微调控件的值。 代码示例： <pre>\$('#ss').numberspinner('setValue', 8234725); // 设置值 var v = \$('#ss').numberspinner('getValue'); // 获取值 alert(v);</pre>

TimeSpinner (时间微调)

扩展自\$.fn.spinner.defaults。使用\$.fn.timespinner.defaults 重写默认值对象。

时间微调组件的创建基于微调组件。它和数字微调类似，但是显示的时间值。时间微调组件允许用户点击组件右侧的小按钮来增加或减少时间。



依赖关系

- spinner

使用案例

使用标签创建时间微调组件。

```
1. <input id="ss" class="easyui-timespinner" style="width:80px;"
2.     required="required" data-options="min:'08:30',showSeconds:true" />
```

使用 Javascript 创建时间微调组件。

```
1. <input id="ss" style="width:80px;">
1. $('#ss').timespinner({
2.     min: '08:30',
3.     required: true,
4.     showSeconds: true
5. });
```

属性

该组件属性扩展自 spinner(微调)，该组件新增的属性如下：

属性名	属性类型	描述	默认值
separator	string	定义在小时、分钟和秒之间的分隔符。	:
showSeconds	boolean	定义是否显示秒钟信息。	false
highlight	number	初始选中的字段 0=小时, 1=分钟...	0
formatter	function(date)	格式化日期函数，该函数接受 date 对象型参数并返回一个字符串值。 (该属性自 1.4 版开始可用)	

		<p>以下的示例代码展示了如何覆盖默认格式化器的方法。</p> <pre>\$.fn.timespinner.defaults.formatter = function(date) { if (!date) {return '';} var opts = \$(this).timespinner('options'); var tt = [formatN(date.getHours()), formatN(date.getMinutes())]; if (opts.showSeconds) { tt.push(formatN(date.getSeconds())); } return tt.join(opts.separator); function formatN(value) { return (value < 10 ? '0' : '') + value; } }</pre>	
parser	function(s)	<p>解析日期/时间字符串的函数，该函数接受 date 字符串类型的参数并返回一个 date 对象值。(该属性自 1.4 版开始可用)</p> <p>以下的示例代码展示了如何覆盖默认日期解析器的方法。</p> <pre>\$.fn.timespinner.defaults.parser = function(s) { var opts = \$(this).timespinner('options'); if (!s) {return null;} var tt = s.split(opts.separator); return new Date(1900, 0, 0, parseInt(tt[0], 10) 0, parseInt(tt[1], 10) 0, parseInt(tt[2], 10) 0); }</pre>	
selections	array	<p>高亮选择部分的值，突出显示每一部分。例如：将字符从 0 点到 2 则高亮小时部分。(该属性自 1.4 版开始可用)</p>	<pre>[[0, 2], [3, 5], [6, 8]]</pre>

事件

该组件事件完全继承自 [spinner](#) (微调)。

方法

该组件的方法扩展自 [spinner](#) (微调)，该组件重写的方法如下：

方法名称	方法参数	描述
options	none	返回属性对象。
setValue	value	设置时间微调组件的值。 代码示例： <pre>\$('#ss').timespinner('setValue', '17:45'); // 设置时间微调组件的值 var v = \$('#ss').timespinner('getValue'); // 获取时间微调组件的值 alert(v);</pre>
getHours	none	获取当前的小时数。
getMinutes	none	获取当前的分钟数。
getSeconds	none	获取当前的秒数。

Slider (滑动条)

使用 `$.fn.slider.defaults` 重写默认值对象。

滑动条允许用户从一个有限的范围内选择一个数值。当滑块控件沿着轨道移动的时候，将会显示一个提示来表示当前值。用户可以通过设置其属性自定义滑块。



依赖关系

- draggable

使用案例

当使用一个表单字段时，使用 `<input>` 标签创建一个滑动条。

```
1. <input class="easyui-slider" value="12" style="width:300px"
2.     data-options="showTip:true,rule:[0,'|',25,'|',50,'|',75,'|',100]" />
```

也允许使用 `<div/>` 创建滚动条，但是 `'value'` 属性是无效的。

```
1. <div class="easyui-slider" data-options="min:10,max:90,step:10" style="width:300px"></div>
```

使用 Javascript 创建滑动条。

```
1. <div id="ss" style="height:200px"></div>
2. $('#ss').slider({
3.     mode: 'v',
4.     tipFormatter: function(value) {
5.         return value + '%';
6.     }
7. });
```

属性

属性名	属性类型	描述	默认值
width	number	滑动条宽度。	auto
height	number	滑动条高度。	auto
mode	string	声明滚动条类型。可用值有：'h'（水平）、'v'（垂直）。	h

reversed	boolean	设置为 true 时，最小值和最大值将对调他们的位置。 (该属性自 1.3.2 版开始可用)	false
showTip	boolean	定义是否显示值信息提示。	false
disabled	boolean	定义是否禁用滑动条。	false
range	boolean	定义是否显示滑块范围。(该属性自 1.4.2 版开始可用) 为 true 时：  为 false 时： 	false
value	number	默认值。	0
min	number	允许的最小值。	0
max	number	允许的最大值。	100
step	number	值增加或减少。	1
rule	array	显示标签旁边的滑块，' ' 只显示一行。	[]
tipFormatter	function	该函数用于格式化滑动条。返回的字符串值将显示提示。	
converter	function	该转换器函数允许用户决定如何将一个值转换为进度条位置或进度条位置值。(该属性自 1.3.6 版开始可用) 代码示例：	

事件

事件名	事件参数	描述
onChange	newValue, oldValue	在字段值更改的时候触发。
onSlideStart	value	在开始拖拽滑动条的时候触发。
onSlideEnd	value	在结束拖拽滑动条的时候触发。
onComplete	value	在滑块值被用户改变的时候触发，无论是拖动还是点击滑块。(该事件自 1.3.4 版开始可用)

方法

方法名	方法参数	描述
options	none	返回滑动条属性。
destroy	none	销毁滑动条对象。
resize	param	设置滑动条大小。'param' 参数包含一下属性： width: 新滑动条宽度。 height: 新滑动条高度。
getValue	none	获取滑动条的值。
getValues	none	获取滑动条的值数组。 (该方法自 1.4.2 版开始可用)
setValue	value	设置滑动条的值。
setValues	value	设置滑动条的值数组。 (该方法自 1.4.2 版开始可用)
clear	none	清除滑动条的值。 (该方法自 1.3.5 版开始可用)
reset	none	重置滑动条的值。 (该方法自 1.3.5 版开始可用)
enable	none	启用滑动条控件。
disable	none	禁用滑动条控件。

FileBox（文件框）

扩展自\$.fn.textbox.defaults，使用\$.fn.filebox.defaults 重写默认值对象。（该组件自 1.4 版开始可用）

FileBox(文件框)组件在表单当中表示一个文件上传的字段。它扩展自 textbox(文本框)，大部分的属性、事件和方法都继承自文本框。但是由于浏览器的安全问题，其中的某些方法（如：“setValue”）则不能用于 filebox 组件。



依赖关系

- textbox

使用案例

通过标签创建验证框。

```
1. <input class="easyui-filebox" style="width:300px">
```

使用 Javascript 创建验证框。

```
1. <input id="fb" type="text" style="width:300px">
2. $('#fb').filebox({
3.     buttonText: '选择文件',
4.     buttonAlign: 'left'
5. })
```

属性

属性扩展自 textbox，以下是新增或重写的文件框属性。

属性名	类型	描述	默认值
buttonText	string	在文本框上附加的按钮显示的文本。	Choose File
buttonIcon	string	在文本框上附加的按钮显示的图标。	null
buttonAlign	string	附加按钮位置。可用值有：“left”，“right”。	right
accept	string	指定接受的文件类型。	
multiple	boolean	指定是否接受多文件选择。	false
separator	string	指定多个文件名称之间的分隔符。	,

事件

事件扩展自 `textbox`。

方法

方法扩展自 `textbox`。

第五章

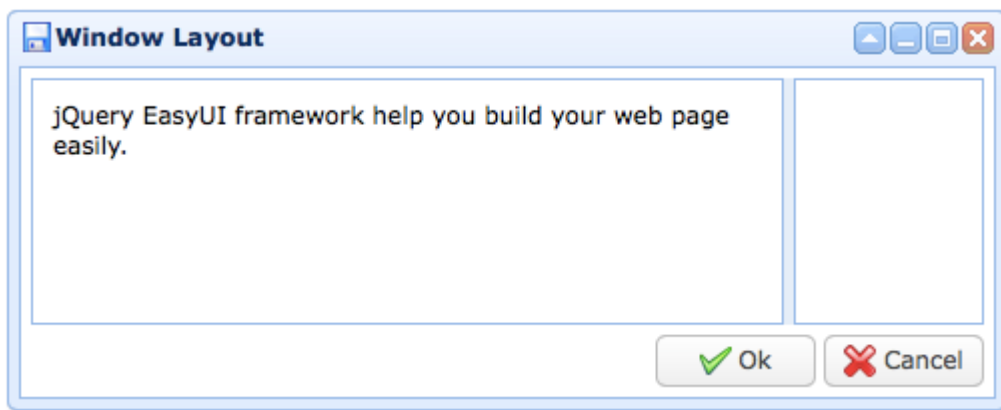
Window

(窗口)

Window (窗口)

扩展自\$.fn.panel.defaults。使用\$.fn.window.defaults 重写默认值对象。

窗口控件是一个浮动和可拖拽的面板可以用作应用程序窗口。默认情况下, 窗口可以移动, 调整大小和关闭。它的内容也可以被定义为静态 html 或要么通过 ajax 动态加载。



依赖关系

- draggable
- resizable
- panel

使用案例

创建窗口

1. 通过标签窗口窗口。

```
1. <div id="win" class="easyui-window" title="My Window" style="width:600px;height:400px"
2.     data-options="iconCls:'icon-save',modal:true">
3.     Window Content
4. </div>
```

2. 通过 Javascript 创建窗口。

```
1. <div id="win"></div>
2. $('#win').window({
3.     width:600,
4.     height:400,
5.     modal:true
6. });
```

3. 创建复合布局窗口。

像往常一样定义窗口布局。下面的例子显示了如何将窗体分为两部分：北部和中间。

```

1. <div id="win" class="easyui-window" title="My Window" style="width:600px;height:400px"
2.     data-options="iconCls:'icon-save',modal:true">
3.     <div class="easyui-layout" data-options="fit:true">
4.         <div data-options="region:'north',split:true" style="height:100px"></div>
5.         <div data-options="region:'center'">
6.             The Content.
7.         </div>
8.     </div>
9. </div>

```

窗口的一些动作

打开和关闭窗口。

```

1. $('#win').window('open'); // open a window
2. $('#win').window('close'); // close a window

```

通过 ajax 读取窗口内容。

```

1. $('#win').window('refresh', 'get_content.php');

```

属性

窗口的属性扩展自 panel(面板)，窗口新增或重新定义的属性如下：

属性名	类型	描述	默认值
title	string	窗口的标题文本。	New Window
collapsible	boolean	定义是否显示可折叠按钮。	true
minimizable	boolean	定义是否显示最小化按钮。	true
maximizable	boolean	定义是否显示最大化按钮。	true
closable	boolean	定义是否显示关闭按钮。	true
closed	boolean	定义是否可以关闭窗口。	false
zIndex	number	窗口 Z 轴坐标。	9000
draggable	boolean	定义是否能够拖拽窗口。	true
resizable	boolean	定义是否能够改变窗口大小。	true
shadow	boolean	如果设置为 true，在窗体显示的时候显示阴影。	true
inline	boolean	定义如何布局窗口，如果设置为 true，窗口将显示在它的父容器中，否则将显示在所有元素的上面。	false
modal	boolean	定义是否将窗体显示为模式化窗口。	true
border	boolean,	定义窗体边框的样式。可用值：true, false, 'thin',	true

	string	'thick'。(该方法自 1.4.5 版开始可用)	
constrain	boolean	定义是否限制窗体的位置。(该方法自 1.5 版开始可用)	false

事件

窗口的事件完整继承自 panel (面板)。

方法

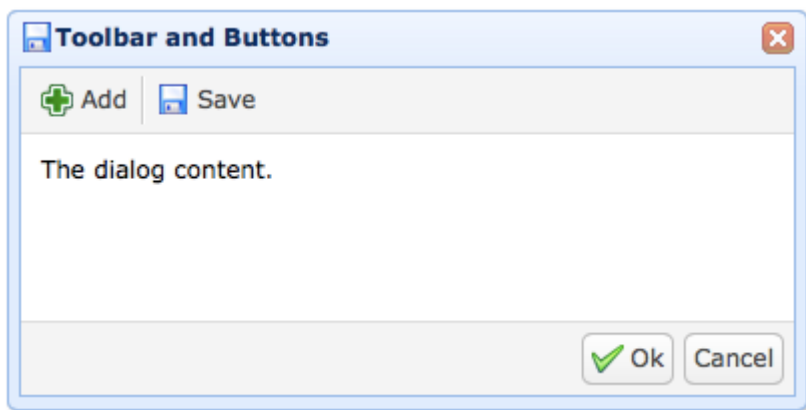
窗口的方法扩展自 panel (面板)，窗口新增的方法如下：

方法名	方法参数	描述
window	none	返回外部窗口对象。
hcenter	none	仅水平居中窗口。(该方法自 1.3.1 版开始可用)
vcenter	none	仅垂直居中窗口。(该方法自 1.3.1 版开始可用)
center	none	将窗口绝对居中。(该方法自 1.3.1 版开始可用)

Dialog (对话框窗口)

扩展自\$.fn.window.defaults。使用\$.fn.dialog.defaults 重写默认值对象。

该对话框是一种特殊类型的窗口，它在顶部有一个工具栏，在底部有一个按钮栏。对话框窗口右上角只有一个关闭按钮用户可以配置对话框的行为显示其他工具，如 collapsible, minimizable, maximizable 工具等。



依赖关系

- window
- linkbutton

使用案例

通过已存在的 DOM 节点元素标签创建。下面的例子显示了一个可变大小的模式窗口。

```
1. <div id="dd" class="easyui-dialog" title="My Dialog" style="width:400px;height:200px;"
2.     data-options="iconCls:'icon-save',resizable:true,modal:true">
3.     Dialog Content.
4. </div>
```

使用 Javascript 创建对话框窗口也是允许的。现在让我们创建一个模式窗口并调用 'refresh' 方法通过 ajax 读取内容。

```
1. <div id="dd">Dialog Content.</div>
1. $(' #dd').dialog({
2.     title: 'My Dialog',
3.     width: 400,
4.     height: 200,
5.     closed: false,
6.     cache: false,
7.     href: 'get_content.php',
```

```

8.     modal: true
9. });
10. $('#dd').dialog('refresh', 'new_content.php');

```

属性

对话框窗口的属性扩展自 window(窗口)，对话框窗口重新定义的属性如下：

属性名	类型	描述	默认值
title	string	对话框窗口标题文本。	New Dialog
collapsible	boolean	定义是否显示可折叠按钮。	false
minimizable	boolean	定义是否显示最小化按钮。	false
maximizable	boolean	定义是否显示最大化按钮。	false
resizable	boolean	定义是否可以改变对话框窗口大小。	false
toolbar	array, selector	<p>设置对话框窗口顶部工具栏，可用值有：</p> <ol style="list-style-type: none"> 1) 一个数组，每一个工具栏中的工具属性都和 linkbutton 相同。 2) 一个选择器指定工具栏。 <p>对话框窗口工具栏可以声明在<div>标签里面：</p> <pre> <div class="easyui-dialog" style="width:600px;height:300px" data-options="title:'我的对话框 ',toolbar:'#tb',modal:true"> 对话框窗口内容。 </div> <div id="tb"> </div> </pre> <p>对话框窗口工具栏也可以通过数组进行定义：</p> <pre> <div class="easyui-dialog" tyle="width:600px;height:300px" data-options="title:'My Dialog',modal:true, toolbar:[{ text:'编辑', iconCls:'icon-edit', handler:function() { alert('edit') } }]"> </pre>	null

		<pre> } }, { text: '帮助', iconCls: 'icon-help', handler: function() { alert('help') } } }]"> 对话框窗口内容。 </div></pre>	
buttons	array, selector	<p>对话框窗口底部按钮，可用值有：</p> <ol style="list-style-type: none">1) 一个数组，每一个按钮的属性都和 linkbutton 相同。2) 一个选择器指定按钮栏。 <p>按钮可以声明在<div>标签里面：</p> <pre><div class="easyui-dialog" style="width:600px;height:300px" data-options="title:' My Dialog', buttons: '#bb', modal:true"> 对话框窗口内容。 </div> <div id="bb"> 保存 关闭 </div></pre> <p>按钮也可以通过数组定义：</p> <pre><div class="easyui-dialog" style="width:600px;height:300px" data-options="title:' 我的对话框 , modal:true, buttons:[{ text:' 保存', handler:function() {...} }, { text:' 关闭', handler:function() {...} }]"> 对话框窗口内容。 </div></pre>	null

事件

对话框窗口事件完全继承自 window(窗口)。

方法

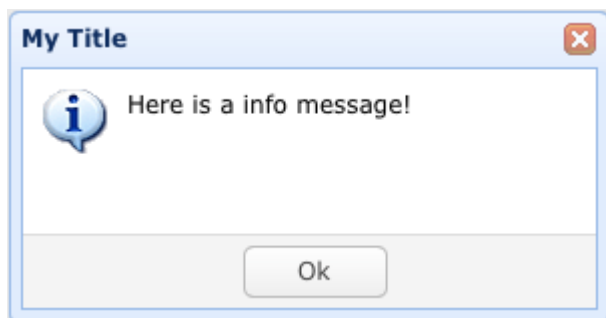
对话框窗口的方法扩展自 window(窗口)，对话框窗口新增的方法如下：

方法名	方法参数	描述
dialog	none	返回外部对话框窗口对象。

Messenger（消息窗口）

使用 `$.messenger.defaults` 重写默认值对象。

消息窗口提供了不同的消息框风格，包含 `alert` (警告框)，`confirm` (确认框)，`prompt` (提示框)，`progress` (进度框) 等。所有的消息框都是异步的。用户可以在交互消息之后使用回调函数去处理结果或做一些自己需要处理的事情。



依赖关系

- window
- linkbutton
- progressbar

使用案例

```
1. $.messenger.alert('警告', '警告消息');
2. $.messenger.confirm('确认', '您确认想要删除记录吗?', function(r) {
3.     if (r) {
4.         alert('确认删除');
5.     }
6. });
```

属性

属性名	类型	描述	默认值
ok	string	确定按钮文本。	Ok
cancel	string	取消按钮文本。	Cancel

方法

方法名	方法参数	描述
<code>\$.messenger.show</code>	options	在屏幕右下角显示一条消息窗口。该选项参数是一个可配置的对

		<p>象:</p> <p>showType: 定义将如何显示该消息。可用值有: null, slide, fade, show。默认: slide。</p> <p>showSpeed: 定义窗口显示的过度时间。默认: 600 毫秒。</p> <p>width: 定义消息窗口的宽度。默认: 250px。</p> <p>height: 定义消息窗口的高度。默认: 100px。</p> <p>title: 在头部面板显示的标题文本。</p> <p>msg: 显示的消息文本。</p> <p>style: 定义消息窗体的自定义样式。</p> <p>timeout: 如果定义为 0, 消息窗体将不会自动关闭, 除非用户关闭他。如果定义成非 0 的数, 消息窗体将在超时后自动关闭。默认: 4 秒。</p> <p>代码示例:</p> <pre>\$.messenger.show({ title:'我的消息', msg:'消息将在 5 秒后关闭。', timeout:5000, showType:'slide' }); // 消息将显示在顶部中间 \$.messenger.show({ title:'我的消息', msg:'消息将在 4 秒后关闭。', showType:'show', style:{ right:'', top:document.body.scrollTop +document.documentElement.scrollTop, bottom:'' } });</pre>
\$.messenger.alert	title, msg, icon, fn	<p>显示警告窗口。参数:</p> <p>title: 在头部面板显示的标题文本。</p> <p>msg: 显示的消息文本。</p> <p>icon: 显示的图标图像。可用值有: error, question, info, warning。</p> <p>fn: 在窗口关闭的时候触发该回调函数。</p> <p>代码示例:</p> <pre>\$.messenger.alert('我的消息', '这是一个提示信息!', 'info');</pre>
\$.messenger.confirm	title, msg, fn	<p>显示一个包含“确定”和“取消”按钮的确认消息窗口。参数:</p> <p>title: 在头部面板显示的标题文本。</p> <p>msg: 显示的消息文本。</p> <p>fn(b): 当用户点击“确定”按钮的时候将传递一个 true 值给回调函数, 否则传递一个 false 值。</p>

		<p>代码示例：</p> <pre>\$.messenger.confirm(' 确认对话框', ' 您想要退出该系统吗?', function(r) { if (r) { // 退出操作; } });</pre>
\$.messenger.prompt	title, msg, fn	<p>显示一个用户可以输入文本的并且带“确定”和“取消”按钮的消息窗体。参数：</p> <p>title: 在头部面板显示的标题文本。</p> <p>msg: 显示的消息文本。</p> <p>fn(val): 在用户输入一个值参数的时候执行的回调函数。</p> <p>代码示例：</p> <pre>\$.messenger.prompt(' 提示信息', ' 请输入你的姓名: ', function(r) { if (r) { alert(' 你的姓名是: ' + r); } });</pre>
\$.messenger.progress	options or method	<p>显示一个进度消息窗体。</p> <p>属性定义为：</p> <p>title: 在头部面板显示的标题文本。默认：空。</p> <p>msg: 显示的消息文本。默认：空。</p> <p>text: 在进度条上显示的文本。默认：undefined。</p> <p>interval: 每次进度更新的间隔时间。默认：300 毫秒。</p> <p>方法定义为：</p> <p>bar: 获取进度条对象。</p> <p>close: 关闭进度窗口。</p> <p>代码示例：</p> <pre>// 显示进度消息窗口 \$.messenger.progress(); // 关闭进度消息窗口 \$.messenger.progress(' close');</pre>

第六章

DataGrid and Tree


(表格和树)

DataGrid（数据表格）

扩展自\$.fn.panel.defaults。使用\$.fn.datagrid.defaults 重写默认值对象。

DataGrid 以表格形式展示数据，并提供了丰富的选择、排序、分组和编辑数据的功能支持。DataGrid 的设计用于缩短开发时间，并且使开发人员不需要具备特定的知识。它是轻量级的且功能丰富。单元格合并、多列标题、冻结列和页脚只是其中的一小部分功能。

DataGrid - Expand Row

	Item ID ▾	Product ID	List Price	Unit Cost	Attribute	Status
⊕	EST-8	K9-PO-02	18.50	12.00	Male Puppy	P
⊕	EST-7	K9-BD-01	18.50	12.00	Female Puppy	P
⊖	EST-6	K9-BD-01	18.50	12.00	Male Adult	P
<div><div></div><div><div>Item ID: EST-6</div><div>List Price: 18.50</div><div>Attribute: Male Adult</div></div><div><div>Product ID: K9-BD-01</div><div>Unit Cost: 12.00</div></div></div>						

10 ▾

⏪ ⏩

Page 1 of 3

⏴ ⏵

🔄

Displaying 1 to 10 of 27 items

依赖关系

- panel
- resizable
- linkbutton
- pagination

使用案例

从现有的表格元素创建 DataGrid，在 HTML 中定义列、行和数据。

```
1. <table class="easyui-datagrid">
2.     <thead>
3.         <tr>
4.             <th data-options="field:'code'">编码</th>
5.             <th data-options="field:'name'">名称</th>
6.             <th data-options="field:'price'">价格</th>
7.         </tr>
8.     </thead>
9.     <tbody>
10.        <tr>
11.            <td>001</td><td>名称 1</td><td>2323</td>
12.        </tr>
13.        <tr>
```

《jQuery EasyUI 简体中文 API 文档》

```

14.         <td>002</td><td>名称 2</td><td>4612</td>
15.     </tr>
16. </tbody>
17. </table>

```

通过<table>标签创建 DataGrid 控件。在表格内使用<th>标签定义列。

```

1. <table class="easyui-datagrid" style="width:400px;height:250px"
2.     data-options="url:'datagrid_data.json',fitColumns:true,singleSelect:true">
3.     <thead>
4.         <tr>
5.             <th data-options="field:'code',width:100">编码</th>
6.             <th data-options="field:'name',width:100">名称</th>
7.             <th data-options="field:'price',width:100,align:'right'">价格</th>
8.         </tr>
9.     </thead>
10. </table>

```

此外，也允许使用 Javascript 去创建 DataGrid 控件。

```

1. <table id="dg"></table>
1. $('#dg').datagrid({
2.     url:'datagrid_data.json',
3.     columns:[[
4.         {field:'code',title:'代码',width:100},
5.         {field:'name',title:'名称',width:100},
6.         {field:'price',title:'价格',width:100,align:'right'}
7.     ]]
8. });

```

使用一些参数查询数据。

```

1. $('#dg').datagrid('load', {
2.     name: 'easyui',
3.     address: 'ho'
4. });

```

改变的数据保存到服务器之后，刷新当前数据。

```

1. $('#dg').datagrid('reload');    // 重新载入当前页面数据

```

DataGrid 属性

该属性扩展自 panel(面板)，下面是 DataGrid 控件添加的属性。

《jQuery EasyUI 简体中文 API 文档》

属性名	类型	描述	默认值
columns	array	DataGrid 列配置对象，详见列属性说明中更多的细节。	undefined
frozenColumns	array	同列属性，但是这些列将会被冻结在左侧。	undefined
fitColumns	boolean	真正的自动展开/收缩列的大小，以适应网格的宽度，防止水平滚动。	false
resizeHandle	string	调整列的位置，可用的值有： 'left', 'right', 'both'。在使用 'right' 的时候用户可以通过拖动右侧边缘的列标题调整列，等等。 (该属性自 1.3.2 版开始可用)	right
autoRowHeight	boolean	定义设置行的高度，根据该行的内容。设置为 false 可以提高负载性能。	true
toolbar	array, selector	<p>顶部工具栏的 DataGrid 面板。可能的值：</p> <ol style="list-style-type: none"> 1) 一个数组，每个工具属性都和 linkbutton 一样。 2) 选择器指定的工具栏。 <p>在<div>标签上定义工具栏：</p> <pre> \$(' #dg').datagrid({ toolbar: '#tb' }); <div id="tb"> </div> </pre> <p>通过数组定义工具栏：</p> <pre> \$(' #dg').datagrid({ toolbar: [{ iconCls: ' icon-edit', handler: function() {alert(' 编辑按钮')} }, '-', { iconCls: ' icon-help', handler: function() {alert(' 帮助按钮')} }] }); </pre>	null
striped	boolean	是否显示斑马线效果。	false
method	string	该方法类型请求远程数据。	post
nowrap	boolean	如果为 true，则在同一行中显示数据。设置为 true 可以提高加载性能。	true
idField	string	指明哪一个字段是标识字段。	null
url	string	一个 URL 从远程站点请求数据。	null
data	array, object	数据加载 (该属性自 1.3.2 版开始可用)	null

		代码示例： <pre> \$('#dg').datagrid({ data: [{f1:'value11', f2:'value12'}, {f1:'value21', f2:'value22'}] }); </pre>	
loadMsg	string	在从远程站点加载数据的时候显示提示消息。	Processing, please wait ...
pagination	boolean	如果为 true, 则在 DataGrid 控件底部显示分页工具栏。	false
rownumbers	boolean	如果为 true, 则显示一个行号列。	false
singleSelect	boolean	如果为 true, 则只允许选择一行。	false
ctrlSelect	boolean	在启用多行选择的时候允许使用 Ctrl 键+鼠标点击的方式进行多选操作。 (该属性自 1.3.6 版开始可用)	false
checkOnSelect	boolean	如果为 true, 当用户点击行的时候该复选框就会被选中或取消选中。 如果为 false, 当用户仅在点击该复选框的时候才会选中或取消。 (该属性自 1.3 版开始可用)	true
selectOnCheck	boolean	如果为 true, 单击复选框将永远选择行。 如果为 false, 选择行将不选中复选框。 (该属性自 1.3 版开始可用)	true
pagePosition	string	定义分页工具栏的位置。可用的值有： 'top', 'bottom', 'both'。 (该属性自 1.3 版开始可用)	bottom
pageNumber	number	在设置分页属性的时候初始化页码。	1
pageSize	number	在设置分页属性的时候初始化页面大小。	10
pageList	array	在设置分页属性的时候初始化页面大小选择列表。	[10, 20, 30, 40, 50]
queryParams	object	在请求远程数据的时候发送额外的参数。 代码示例： <pre> \$('#dg').datagrid({ queryParams: { name: 'easyui', subject: 'datagrid' } }); </pre>	{}
sortName	string	定义哪些列可以进行排序。	null
sortOrder	string	定义列的排序顺序, 只能是 'asc' 或 'desc'。	asc
multiSort	boolean	定义是否允许多列排序。 (该属性自 1.3.4 版开始可用)	false
remoteSort	boolean	定义从服务器对数据进行排序。	true
showHeader	boolean	定义是否显示行头。	true

showFooter	boolean	定义是否显示行脚。	false
scrollbarSize	number	滚动条的宽度(当滚动条是垂直的时候)或高度(当滚动条是水平的时候)。	18
rownumberWidth	number	行号列宽度。 译者注(该属性官方未标注出可用版本号,译者是在 1.5 版本中发现新增的属性,姑且认为是 1.5 新增属性。1.5 之前的版本可自行尝试,如果无效请删除。)	30
editorHeight	number	编辑器默认高度。 译者注(该属性官方未标注出可用版本号,译者是在 1.5 版本中发现新增的属性,姑且认为是 1.5 新增属性。1.5 之前的版本可自行尝试,如果无效请删除。)	24
rowStyler	function	<p>返回样式如'background:red'。带 2 个参数的函数: index: 该行索引从 0 开始 row: 与此相对应的记录行。</p> <p>代码示例:</p> <pre>\$('#dg').datagrid({ rowStyler: function(index,row) { if (row.listprice>80) { return 'background-color:#6293BB; color:#fff;'; } } });</pre> <p>译者注 (1.3.4 新增方式) :</p> <pre>\$('#dg').datagrid({ rowStyler: function(index,row) { if (row.listprice>80) { // rowStyle 是一个已经定义了的 ClassName return 'rowStyle'; } } });</pre>	
loader	function	<p>定义如何从远程服务器加载数据。返回 false 可以放弃本次请求动作。该函数接受以下参数:</p> <p>param: 参数对象传递给远程服务器。</p> <p>success(data): 当检索数据成功的时候会调用该回调函数。</p> <p>error(): 当检索数据失败的时候会调用该回调函数。</p>	json loader
loadFilter	function	<p>返回过滤数据显示。该函数带一个参数'data'用来指向源数据(即:获取的数据源,比如Json对象)。您可以改变源数据的标准数据格式。这个函数必须返回包含'total'和'rows'属性的标准数据对象。</p>	

		<p>代码示例：</p> <pre>// 从 Web Service (asp.net、java、php 等) 输出的 JSON 对象中移除 'd' 对象 \$('#dg').datagrid({ loadFilter: function(data) { if (data.d) { return data.d; } else { return data; } } });</pre>	
editors	object	定义在编辑行的时候使用的编辑器。	预定义编辑器
view	object	定义 DataGrid 的视图。	默认视图

列属性

DataGrid 列是一个数组对象，该元素也是一个数组对象。元素数组里面的元素是一个配置对象，它用来定义每一个列字段。

代码示例：

```
1. columns:[[
2.     {field:'itemid',title:'Item ID',rowspan:2,width:80,sortable:true},
3.     {field:'productid',title:'Product ID',rowspan:2,width:80,sortable:true},
4.     {title:'Item Details',colspan:4}
5. ],[
6.     {field:'listprice',title:'List Price',width:80,align:'right',sortable:true},
7.     {field:'unitcost',title:'Unit Cost',width:80,align:'right',sortable:true},
8.     {field:'attr1',title:'Attribute',width:100},
9.     {field:'status',title:'Status',width:60}
10. ]]
```

属性名称	类型	描述	默认值
title	string	列标题文本。	undefined
field	string	列字段名称。	undefined
width	number	列的宽度。如果没有定义，宽度将自动扩充以适应其内容。	undefined
rowspan	number	指明将占用多少行单元格（合并行）。	undefined
colspan	number	指明将占用多少列单元格（合并列）。	undefined
align	string	指明如何对齐列数据。可以使用的值有：	undefined

		'left','right','center'。	
halign	string	指明如何对齐列标题。可以使用的值有： 'left','right','center'。如果没有指定，则按照 align 属性进行对齐。 (该属性自 1.3.2 版开始可用)	undefined
sortable	boolean	如果为 true，则允许列使用排序。	undefined
order	string	默认排序数序，只能是'asc'或'desc'。 (该属性自 1.3.2 版开始可用)	undefined
resizable	boolean	如果为 true，允许列改变大小。	undefined
fixed	boolean	如果为 true，在"fitColumns"设置为 true 的时候阻止其自适应宽度。	undefined
hidden	boolean	如果为 true，则隐藏列。	undefined
checkbox	boolean	如果为 true，则显示复选框。该复选框列固定宽度。	undefined
formatter	function	单元格 formatter(格式化器)函数，带 3 个参数： value: 字段值。 row: 行记录数据。 index: 行索引。 代码示例： <pre>\$('#dg').datagrid({ columns:[[{field:'userId',title:'User', width:80, formatter: function(value,row,index){ if (row.user){ return row.user.name; } else { return value; } }]] });</pre>	undefined
styler	function	单元格 styler(样式)函数，返回如'background:red' 这样的自定义单元格样式字符串。该函数带 3 个参数： value: 字段值。 row: 行记录数据。 index: 行索引。 代码示例： <pre>\$('#dg').datagrid({ columns:[[{ field:'listprice', title:'List Price', width:80, align:'right',</pre>	undefined

		<pre> styler: function(value, row, index) { if (value < 20) { return 'background-color:#ffee00; color:red;'; } } }]] }); </pre>	
sorter	function	<p>用来做本地排序的自定义字段排序函数，带 2 个参数： a：第一个字段值。 b：第二个字段值。</p> <p>代码示例：</p> <pre> \$('#dg').datagrid({ remoteSort: false, columns: [[{ field:'date', title:'Date', width:80, sortable:true, align:'center', sorter:function(a,b){ a = a.split('/'); b = b.split('/'); if (a[2] == b[2]){ if (a[0] == b[0]){ return (a[1]>b[1]?1:-1); } else { return (a[0]>b[0]?1:-1); } } else { return (a[2]>b[2]?1:-1); } } }]]] }); </pre>	undefined
editor	string, object	<p>指明编辑类型。当字符串指明编辑类型的时候，对象包含 2 个属性：</p> <p>type：字符串，该编辑类型可以使用的类型有：text, textarea, checkbox, numberbox, validatebox, datebox, combobox, combotree。</p> <p>options：对象，object，该编辑器属性对应于编辑类型。</p>	undefined

编辑器

使用 `$.fn.datagrid.defaults.editors` 重写默认值对象。

每一个编辑器都有下面的动作：

名称	参数	描述
init	container, options	初始化编辑器并返回目标对象。
destroy	target	如果有必要销毁编辑器。
getValue	target	从编辑器中获取值。
setValue	target , value	向编辑器中写入值。
resize	target , width	如果有必要调整编辑器的大小。

例如，该文本编辑器定义如下：

```

1.  $.extend($.fn.datagrid.defaults.editors, {
2.      text: {
3.          init: function(container, options){
4.              var input = $('<input type="text" class="datagrid-editable-input">').appendTo(container);
5.              return input;
6.          },
7.          getValue: function(target){
8.              return $(target).val();
9.          },
10.         setValue: function(target, value){
11.             $(target).val(value);
12.         },
13.         resize: function(target, width){
14.             var input = $(target);
15.             if ($.boxModel == true){
16.                 input.width(width - (input.outerWidth() - input.width()));
17.             } else {
18.                 input.width(width);
19.             }
20.         }
21.     }
22. });

```

DataGrid 视图

使用 `$.fn.datagrid.defaults.view` 重写默认值对象。

该视图是一个对象，将告诉 DataGrid 如何渲染行。该对象必须定义下列函数：

《jQuery EasyUI 简体中文 API 文档》

名称	参数	描述
render	target, container, frozen	数据加载时调用。 target: DOM 对象, DataGrid 对象。 container: 行容器。 frozen: 指明如何渲染冻结容器。
renderFooter	target, container, frozen	这是一个选择函数来渲染行页脚。
renderRow	target, fields, frozen, index, row	这是一个属性功能, 将调用 render 函数。
refreshRow	target, index	定义如何刷新指定的行。
onBeforeRender	target, rows	在视图被呈现之前触发。
onAfterRender	target	在视图呈现之后触发。

事件

事件继承自 panel (面板), 以下是 DataGrid 新增的事件。

事件名	参数	描述
onLoadSuccess	data	在数据加载成功的时候触发。
onLoadError	none	在载入远程数据产生错误的时候触发。
onBeforeLoad	param	在载入请求数据之前触发, 如果返回 false 可终止载入数据操作。
onClickRow	index, row	在用户点击一行的时候触发, 参数包括: index: 点击的行的索引值, 该索引值从 0 开始。 row: 对应于点击行的记录。
onDbClickRow	index, row	在用户双击一行的时候触发, 参数包括: index: 点击的行的索引值, 该索引值从 0 开始。 row: 对应于点击行的记录。
onClickCell	index, field, value	在用户点击一个单元格的时候触发。
onDbClickCell	index, field, value	在用户双击一个单元格的时候触发。 代码示例: // 在双击一个单元格的时候开始编辑并生成编辑器, 然后定位到编辑器的输入框上 <pre> \$(' #dg').datagrid({ onDbClickCell: function(index, field, value) { \$(this).datagrid('beginEdit', index); var ed = \$(this).datagrid('getEditor', {index:index, field:field}); \$(ed.target).focus(); } }); </pre>
onBeforeSortCo	sort, order	在用户排序一个列之前触发, 返回 false 可以取消排序。 (该事件

lumn		自 1.3.6 版开始可用)
onSortColumn	sort, order	在用户排序一系列的时候触发，参数包括： sort：排序列字段名称。 order：排序列的顺序(ASC 或 DESC)
onResizeColumn	field, width	在用户调整列大小的时候触发。
onBeforeSelect	index, row	在用户选择一行之前触发，返回 false 则取消该动作。(该事件自 1.4.1 版开始可用)
onSelect	index, row	在用户选择一行时候触发，参数包括： index：选择的行的索引值，索引从 0 开始。 row：对应于取消选择行的记录。
onBeforeUnselect	index, row	在用户取消选择一行之前触发，返回 false 则取消该动作。(该事件自 1.4.1 版开始可用)
onUnselect	index, row	在用户取消选择一行时候触发，参数包括： index：选择的行的索引值，索引从 0 开始。 row：对应于取消选择行的记录。
onSelectAll	rows	在用户选择所有行的时候触发。
onUnselectAll	rows	在用户取消选择所有行的时候触发。
onBeforeCheck	index, row	在用户校验一行之前触发，返回 false 则取消该动作。(该事件自 1.4.1 版开始可用)
onCheck	index, row	在用户勾选一行时候触发，参数包括： index：选中的行索引，索引从 0 开始。 row：对应于所选行的记录。 (该事件自 1.3 版开始可用)
onBeforeUncheck	index, row	在用户取消校验一行之前触发，返回 false 则取消该动作。(该事件自 1.4.1 版开始可用)
onUncheck	index, row	在用户取消勾选一行时候触发，参数包括： index：选中的行索引，索引从 0 开始。 row：对应于取消勾选行的记录。 (该事件自 1.3 版开始可用)
onCheckAll	rows	在用户勾选所有行的时候触发。(该事件自 1.3 版开始可用)
onUncheckAll	rows	在用户取消勾选所有行的时候触发。(该事件自 1.3 版开始可用)
onBeforeEdit	index, row	在用户开始编辑一行时候触发，参数包括： index：编辑行的索引，索引从 0 开始。 row：对应于编辑行的记录。
onBeginEdit	index, row	在一行进入编辑模式的时候触发。(该事件自 1.3.6 版开始可用)
onEndEdit	index, row, changes	在完成编辑但编辑器还没有销毁之前触发。(该事件自 1.3.6 版开始可用)
onAfterEdit	index, row, changes	在用户完成编辑一行时候触发，参数包括： index：编辑行的索引，索引从 0 开始。 row：对应于完成编辑的行的记录。 changes：更改后的字段(键)/值对。
onCancelEdit	index, row	在用户取消编辑一行时候触发，参数包括： index：编辑行的索引，索引从 0 开始。 row：对应于编辑行的记录。
onHeaderContext	e, field	在鼠标右击 DataGrid 表格头的时候触发。

tMenu		
onRowContextMenu	e, index, row	在鼠标右击一行记录的时候触发。

方法

方法名	参数	描述
options	none	返回属性对象。
getPager	none	返回页面对象。
getPanel	none	返回面板对象。
getColumnFields	frozen	返回列字段。如果设置了 frozen 属性为 true，将返回固定列的字段名。 代码示例： <pre>var opts = \$('#dg').datagrid('getColumnFields'); // 获取解冻列 var opts = \$('#dg').datagrid('getColumnFields', true); // 获取冻结列</pre>
getColumnOption	field	返回指定列属性。
resize	param	做调整和布局。
load	param	加载和显示第一页的所有行。如果指定了 'param'，它将取代 'queryParams' 属性。通常可以通过传递一些参数执行一次查询，通过调用这个方法从服务器加载新数据。 <pre>\$('#dg').datagrid('load', { code: '01', name: 'name01' });</pre>
reload	param	重载行。等同于 'load' 方法，但是它将保持在当前页。
reloadFooter	footer	重载页脚行。代码示例： <pre>// 更新页脚行的值并刷新 var rows = \$('#dg').datagrid('getFooterRows'); rows[0]['name'] = 'new name'; rows[0]['salary'] = 60000; \$('#dg').datagrid('reloadFooter');</pre> <pre>// 更新页脚行并载入新数据 \$('#dg').datagrid('reloadFooter', [{name: 'name1', salary: 60000}, {name: 'name2', salary: 65000}]);</pre>
loading	none	显示载入状态。
loaded	none	隐藏载入状态。
fitColumns	none	使列自动展开/收缩到合适的 DataGrid 宽度。
fixColumnSize	field	固定列大小。如果 'field' 参数未配置，所有列大小将都是固定的。

		代码示例： <pre> \$\$('#dg').datagrid('fixColumnSize', 'name'); // 固定' name' 列大小 \$\$('#dg').datagrid('fixColumnSize'); // 固定所有列大小 </pre>
fixRowHeight	index	固定指定列高度。如果' index' 参数未配置，所有行高度都是固定的。
freezeRow	index	冻结指定行，当 DataGrid 表格向下滚动的时候始终保持被冻结的行显示在顶部。 (该方法自 1.3.2 版开始可用)
autoSizeColumn	field	自动调整列宽度以适应内容。 (该方法自 1.3 版开始可用)
loadData	data	加载本地数据，旧的行将被移除。
getData	none	返回加载完毕后的数据。
getRows	none	返回当前页的所有行。
getFooterRows	none	返回页脚行。
getRowIndex	row	返回指定行的索引号，该行的参数可以是一行记录或一个 ID 字段值。
getChecked	none	在复选框选中时返回所有行。 (该方法自 1.3 版开始可用)
getSelected	none	返回第一个被选中的行或如果没有选中的行则返回 null。
getSelections	none	返回所有被选中的行，当没有记录被选中的时候将返回一个空数组。
clearSelections	none	清除所有选择的行。
clearChecked	none	清除所有勾选的行。 (该方法自 1.3.2 版开始可用)
scrollTo	index	滚动到指定的行。 (该方法自 1.3.3 版开始可用)
gotoPage	param	跳转到指定页。 (该方法自 1.4.4 版开始可用) 代码示例： <pre> // 跳转到第 3 页 \$\$('#dg').datagrid('gotoPage', 3); // 跳转到第 3 页并执行回调函数 \$\$('#dg').datagrid('gotoPage', { page: 3, callback: function(page) { console.log(page) } }) </pre>
highlightRow	index	高亮一行。 (该方法自 1.3.3 版开始可用)
selectAll	none	选择当前页中所有的行。
unselectAll	none	取消选择所有当前页中所有的行。
selectRow	index	选择一行，行索引从 0 开始。
selectRecord	idValue	通过 ID 值参数选择一行。
unselectRow	index	取消选择一行。
checkAll	none	勾选当前页中的所有行。 (该方法自 1.3 版开始可用)
uncheckAll	none	取消勾选当前页中的所有行。 (该方法自 1.3 版开始可用)
checkRow	index	勾选一行，行索引从 0 开始。 (该方法自 1.3 版开始可用)
uncheckRow	index	取消勾选一行，行索引从 0 开始。 (该方法自 1.3 版开始可用)
beginEdit	index	开始编辑行。
endEdit	index	结束编辑行。
cancelEdit	index	取消编辑行。

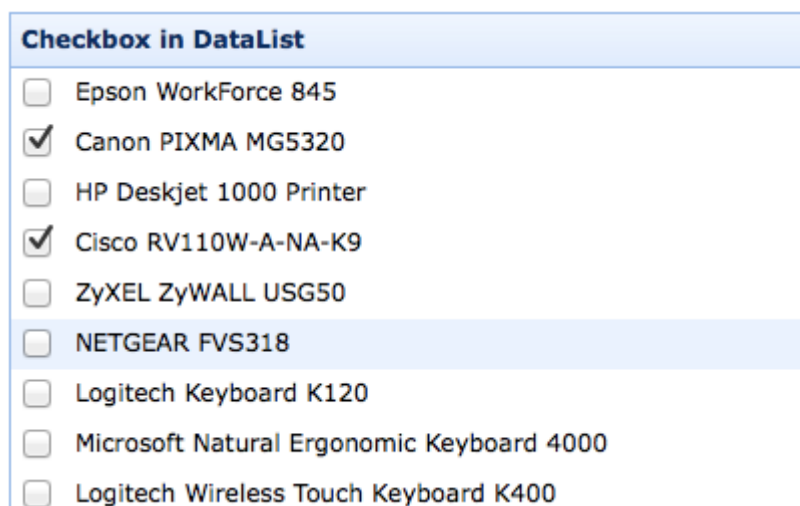
getEditors	index	<p>获取指定行的编辑器。每个编辑器都有以下属性：</p> <p>actions: 编辑器可以执行的动作，同编辑器定义。</p> <p>target: 目标编辑器的 jQuery 对象。</p> <p>field: 字段名称。</p> <p>type: 编辑器类型，比如：'text', 'combobox', 'datebox' 等。</p>
getEditor	options	<p>获取指定编辑器，options 包含 2 个属性：</p> <p>index: 行索引。</p> <p>field: 字段名称。</p> <p>代码示例：</p> <pre>// 获取日期输入框编辑器并更改它的值 var ed = \$('#dg').datagrid('getEditor', {index:1,field:'birthday'}); \$(ed.target).datebox('setValue', '5/4/2012');</pre>
refreshRow	index	刷新行。
validateRow	index	验证指定的行，当验证有效的时候返回 true。
updateRow	param	<p>更新指定行，参数包含下列属性：</p> <p>index: 执行更新操作的行索引。</p> <p>row: 更新行的新数据。</p> <p>代码示例：</p> <pre>\$('#dg').datagrid('updateRow',{ index: 2, row: { name: '新名称', note: '新消息' } });</pre>
appendRow	row	<p>追加一个新行。新行将被添加到最后的位置。</p> <pre>\$('#dg').datagrid('appendRow',{ name: '新名称', age: 30, note: '新消息' });</pre>
insertRow	param	<p>插入一个新行，参数包括一下属性：</p> <p>index: 要插入的行索引，如果该索引值未定义，则追加新行。</p> <p>row: 行数据。</p> <p>代码示例：</p> <pre>// 在第二行的位置插入一个新行 \$('#dg').datagrid('insertRow',{ index: 1, // 索引从 0 开始 row: { name: '新名称', age: 30, note: '新消息' } });</pre>

		} });
deleteRow	index	删除行。
getChanges	type	从上一轮的提交获取改变的所有行。类型参数指明用哪些类型改变的行，可以使用的值有：inserted, deleted, updated 等。当类型参数未配置的时候返回所有改变的行。
acceptChanges	none	提交所有从加载或者上一次调用 acceptChanges 函数后更改的数据。
rejectChanges	none	回滚所有从创建或者上一次调用 acceptChanges 函数后更改的数据。
mergeCells	options	合并单元格，options 包含以下属性： index：行索引。 field：字段名称。 rowspan：合并的行数。 colspan：合并的列数。
showColumn	field	显示指定的列。
hideColumn	field	隐藏指定的列。
sort	param	排序 datagrid 表格。 (该方法自 1.3.6 版开始可用) 代码示例： <pre> \$('#dg').datagrid('sort', 'itemid'); // 排序一个列 \$('#dg').datagrid('sort', { // 指定了排序顺序的列 sortName: 'productid', sortOrder: 'desc' }); </pre>

DataList（数据列表）

扩展自\$.fn.datagrid.defaults。用于\$.fn.datalist.defaults 重写默认值对象。（该组件自 1.4.2 版开始可用）

datalist 组件专用于在列表数据上渲染，这是一个特殊的 datagrid 组件，专用于渲染 1 列数据时使用。您可以定义列当中每一行的展示格式及样式。



依赖关系

- datagrid

使用案例

使用标签创建数据列表。

```

1. <ul class="easyui-datalist" title="DataList" style="width:400px;height:250px">
2.   <li value="AL">Alabama</li>
3.   <li value="AK">Alaska</li>
4.   <li value="AZ">Arizona</li>
5.   <li value="AR">Arkansas</li>
6.   <li value="CA">California</li>
7.   <li value="CO">Colorado</li>
8. </ul>

```

使用 Javascript 创建分割按钮。

```

1. <div id="dl"></div>
1. $(' #dl').datalist({
《jQuery EasyUI 简体中文 API 文档》

```

```
2.    url: 'datalist_data1.json',
3.    checkbox: true,
4.    lines: true
5.  });
```

属性

数据列表属性扩展自 datagrid，数据列表新增的属性如下：

属性名	类型	描述	默认值
lines	boolean	定义是否显示行号	false
checkbox	boolean	定义是否在每行开头显示复选框。	false
valueField	string	行绑定的字段值名称。	value
textField	string	行绑定的字段名名称。	text
groupField	string	指定分组字段名称。	
textFormatter	function(value, row, index)	字段名称格式化器，返回格式化后的字段内容，共 3 个参数： value: 字段值 row: 行记录数据 index: 行索引	
groupFormatter	function(value, rows)	分组名称格式化器，返回格式化后的分组内容，共 2 个参数： value: 通过“groupField”属性定义的分组字段名称 rows: 指定分组字段名称对应的数据行记录数组	

事件

数据列表事件扩展自 datagrid。


方法

数据列表方法扩展自 datagrid。

PropertyGrid (属性表格)

继承自\$.fn.datagrid.defaults。使用\$.fn.propertygrid.defaults 重写默认值对象。

属性表格提供 The propertygrid provide 给用户浏览和编辑对象属性的一个接口。属性表格是一个行内可编辑数据表格。它使用起来相当简单。用户可以非常简单的创建一个分层的可编辑属性列表和表示任何数据类型的项。属性表格内建排序和分组功能。

Name	Value
ID Settings	
Name	Bill Smith
Address	
Age	40
Birthday	01/02/2012 
SSN	123-456-7890
Marketing Settings	
Email	bill@gmail.com
FrequentBuyer	false

依赖关系

- datagrid

使用案例

使用标签创建一个属性表格。注意：列已经内置不需要再去声明它。

```
1. <table id="pg" class="easyui-propertygrid" style="width:300px"
2.     data-options="url:'get_data.php',showGroup:true,scrollbarSize:0"></table>
```

使用 Javascript 创建一个属性表格。

```
1. <table id="pg" style="width:300px"></table>
1. $('#pg').propertygrid({
2.     url: 'get_data.php',
3.     showGroup: true,
4.     scrollbarSize: 0
5. });
```

追加一个新行到属性表格。

```
1. var row = {
2.     name: 'AddName',
```

```

3.     value:'',
4.     group:'Marketing Settings',
5.     editor:'text'
6.   };
7.   $('#pg').propertygrid('appendRow', row);

```

行数据

属性表格扩展自 datagrid(数据表格)。它的行数据格式和数据表格相同。作为一个属性行，以下字段是必须的：

name：字段名称。

value：字段值。

group：分组字段值。

editor：在编辑属性值的时候使用的编辑器对象。

行数据示例：

```

1.  {"total":4,"rows":[
2.    {"name":"Name","value":"Bill Smith","group":"ID Settings","editor":"text"},
3.    {"name":"Address","value":"","group":"ID Settings","editor":"text"},
4.    {"name":"SSN","value":"123-456-7890","group":"ID Settings","editor":"text"},
5.    {"name":"Email","value":"bill@gmail.com","group":"Marketing Settings","editor":{"
6.      "type":"validatebox",
7.      "options":{"
8.        "validType":"email"
9.      }}
10.  ]}
11. ]}

```

属性

属性表格的属性扩展自 datagrid(数据表格)，属性表格新增的属性如下：

属性名	属性类型	描述	默认值
showGroup	boolean	定义是否显示属性分组。	false
groupField	string	定义分组的字段名。	group
groupFormatter	function(group, rows)	定义如何格式化分组的值。该函数拥有如下参数： group：分组字段值。 rows：属于该分组的所有行。	

方法

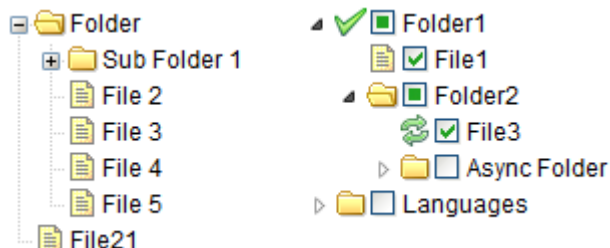
属性表格的方法扩展自 datagrid(数据表格)，属性表格新增的方法如下：

方法名	方法参数	描述
groups	none	返回所有分组。每一组包含如下属性。 (该方法自 1.4.4 版开始可用) value: 分组的字段值。 rows: 属于该分组的行。 startIndex: 当前行相对于所有行的索引。
expandGroup	groupIndex	展开指定分组。如果'groupIndex'参数未指定，则展开所有分组。
collapseGroup	groupIndex	折叠指定分组。如果'groupIndex'参数未指定，则折叠所有分组。

Tree (树)

使用 `$.fn.tree.defaults` 重写默认值对象。

树控件在 web 页面中一个将分层数据以树形结构进行显示。它提供用户展开、折叠、拖拽、编辑和异步加载等功能。



依赖关系

- draggable
- droppable

使用案例

树控件使用 `` 元素定义。标签能够定义分支和子节点。节点都定义在 `` 列表内的 `` 元素中。以下显示的元素将被用作树节点嵌套在 `` 元素中。

```

1. <ul id="tt" class="easyui-tree">
2.   <li>
3.     <span>Folder</span>
4.     <ul>
5.       <li>
6.         <span>Sub Folder 1</span>
7.         <ul>
8.           <li>
9.             <span><a href="#">File 11</a></span>
10.          </li>
11.          <li>
12.            <span>File 12</span>
13.          </li>
14.          <li>
15.            <span>File 13</span>
16.          </li>
17.        </ul>
18.      </li>
19.      <li>

```

```

20.         <span>File 2</span>
21.     </li>
22.     <li>
23.         <span>File 3</span>
24.     </li>
25. </ul>
26. </li>
27. <li>
28.     <span>File21</span>
29. </li>
30. </ul>

```

树控件也可以定义在一个空元素中并使用 Javascript 加载数据。

```

1. <ul id="tt"></ul>
1. $('#tt').tree({
2.     url: 'tree_data.json'
3. });

```

使用 loadFilter 函数处理来自 Web Services 的 JSON 数据。

```

1. $('#tt').tree({
2.     url: ...,
3.     loadFilter: function(data) {
4.         if (data.d) {
5.             return data.d;
6.         } else {
7.             return data;
8.         }
9.     }
10. });

```

树控件数据格式化

每个节点都具备以下属性：

- id: 节点 ID，对加载远程数据很重要。
- text: 显示节点文本。
- state: 节点状态，'open' 或 'closed'，默认：'open'。如果为'closed'的时候，将不自动展开该节点。
- checked: 表示该节点是否被选中。
- attributes: 被添加到节点的自定义属性。
- children: 一个节点数组声明了若干节点。

一些案例：

```

1.  [{
2.      "id":1,
3.      "text":"Folder1",
4.      "iconCls":"icon-save",
5.      "children":[{
6.          "text":"File1",
7.          "checked":true
8.      }],{
9.          "text":"Books",
10.         "state":"open",
11.         "attributes":{
12.             "url":"/demo/book/abc",
13.             "price":100
14.         },
15.         "children":[{
16.             "text":"PhotoShop",
17.             "checked":true
18.         }],{
19.             "id": 8,
20.             "text":"Sub Bookds",
21.             "state":"closed"
22.         }]
23.     }]
24. }, {
25.     "text":"Languages",
26.     "state":"closed",
27.     "children":[{
28.         "text":"Java"
29.     }, {
30.         "text":"C#"
31.     }]
32. }]

```

异步树控件

树控件内建异步加载模式的支持，用户先创建一个空的树，然后指定一个服务器端，执行检索后动态返回 JSON 数据来填充树并完成异步请求。例子如下：

```
1. <ul class="easyui-tree" data-options="url:'get_data.php'"></ul>
```

树控件读取 URL。子节点的加载依赖于父节点的状态。当展开一个封闭的节点，如果节点没有加载子节点，它将会把节点 id 的值作为 http 请求参数并命名为 'id'，通过 URL 发送到服务器上面检索子节点。

下面是从服务器端返回的数据。

```
1.  [{
《jQuery EasyUI 简体中文 API 文档》
```

```

2.     "id": 1,
3.     "text": "Node 1",
4.     "state": "closed",
5.     "children": [{
6.         "id": 11,
7.         "text": "Node 11"
8.     }, {
9.         "id": 12,
10.        "text": "Node 12"
11.    }]
12. }, {
13.     "id": 2,
14.     "text": "Node 2",
15.     "state": "closed"
16. }]

```

节点 1 和节点 2 是封闭的，当展开节点 1 的时候将直接显示它的子节点。当展开节点 2 的时候它将发送值(2)到服务器获取子节点。

属性

属性名	类型	描述	默认值
url	string	检索远程数据的 URL 地址。	null
method	string	检索数据的 HTTP 方法。(POST / GET)	post
animate	boolean	定义节点在展开或折叠的时候是否显示动画效果。	false
checkbox	boolean	定义是否在每一个节点之前都显示复选框。	false
cascadeCheck	boolean	定义是否层叠选中状态。	true
onlyLeafCheck	boolean	定义是否只在末级节点之前显示复选框。	false
lines	boolean	定义是否显示树控件上的虚线。	false
dnd	boolean	定义是否启用拖拽功能。	false
data	array	节点数据加载。 <pre> \$('#tt').tree({ data: [{ text: 'Item1', state: 'closed', children: [{ text: 'Item11' }, { text: 'Item12' }] }, { text: 'Item2' }] }) </pre>	null

		<pre>]] }); </pre>	
queryParams	object	在请求远程数据的时候增加查询参数并发送到服务器。 (该属性自 1.4 版开始可用)	{}
formatter	function (node)	定义如何渲染节点的文本。 代码示例： <pre> \$(' #tt').tree({ formatter:function(node) { return node.text; } }); </pre>	false
filter	function (q, node)	定义如何过滤本地展示的数据，返回 true 将允许节点进行展示。 (该属性自 1.4.2 版开始可用)	json loader
loader	function (param, success, error)	定义如何从远程服务器加载数据。返回 false 可以忽略本操作。 该函数具备以下参数： param：发送到远程服务器的参数对象。 success(data)：当检索数据成功的时候调用的回调函数。 error()：当检索数据失败的时候调用的回调函数。	json loader
loadFilter	function (data, parent)	返回过滤过的数据进行展示。返回数据是标准树格式。该函数具备以下参数： data：加载的原始数据。 parent：DOM 对象，代表父节点。	

事件

很多事件的回调函数都包含 'node' 参数，其具备如下属性：

- id：绑定节点的标识值。
- text：显示的节点文本。
- iconCls：显示的节点图标 CSS 类 ID。
- checked：该节点是否被选中。
- state：节点状态，'open' 或 'closed'。
- attributes：绑定该节点的自定义属性。
- target：目标 DOM 对象。

事件名	事件参数	描述
onClick	node	在用户点击一个节点的时候触发。 代码示例： <pre> \$(' #tt').tree({ onClick: function(node) { alert(node.text); // 在用户点击的时候提示 </pre>

		<pre> } }); </pre>
onDb1Click	node	在用户双击一个节点的时候触发。
onBeforeLoad	node, param	在请求加载远程数据之前触发, 返回 false 可以取消加载操作。
onLoadSuccess	node, data	在数据加载成功以后触发。
onLoadError	arguments	在数据加载失败的时候触发, arguments 参数和 jQuery 的 \$.ajax() 函数里面的 'error' 回调函数的参数相同。
onBeforeExpand	node	在节点展开之前触发, 返回 false 可以取消展开操作。
onExpand	node	在节点展开的时候触发。
onBeforeCollapse	node	在节点折叠之前触发, 返回 false 可以取消折叠操作。
onCollapse	node	在节点折叠的时候触发。
onBeforeCheck	node, checked	在用户点击勾选复选框之前触发, 返回 false 可以取消选择动作。 (该事件自 1.3.1 版开始可用)
onCheck	node, checked	在用户点击勾选复选框的时候触发。
onBeforeSelect	node	在用户选择一个节点之前触发, 返回 false 可以取消选择动作。
onSelect	node	在用户选择节点的时候触发。
onContextMenu	e, node	<p>在右键点击节点的时候触发。</p> <p>代码示例:</p> <pre> // 右键点击节点并显示快捷菜单 \$('#tt').tree({ onContextMenu: function(e, node){ e.preventDefault(); // 查找节点 \$('#tt').tree('select', node.target); // 显示快捷菜单 \$('#mm').menu('show', { left: e.pageX, top: e.pageY }); } }); </pre> <p>// 右键菜单定义如下:</p> <pre> <div id="mm" class="easyui-menu" style="width:120px;"> <div onclick="append()" data-options="iconCls:'icon-add'"> 追加</div> <div onclick="remove()" data-options="iconCls:'icon- remove'">移除</div> </div> </pre>
onBeforeDrag	node	在开始拖动节点之前触发, 返回 false 可以拒绝拖动。 (该事件自 1.3.2 版开始可用)
onStartDrag	node	在开始拖动节点的时候触发。 (该事件自 1.3.2 版开始可用)

onStopDrag	node	在停止拖动节点的时候触发。 (该事件自 1.3.2 版开始可用)
onDragEnter	target, source	在拖动一个节点进入到某个目标节点并释放的时候触发, 返回 false 可以拒绝拖动。 target: 释放的目标节点元素。 source: 开始拖动的源节点。 (该事件自 1.3.2 版开始可用)
onDragOver	target, source	在拖动一个节点经过某个目标节点并释放的时候触发, 返回 false 可以拒绝拖动。 target: 释放的目标节点元素。 source: 开始拖动的源节点。 (该事件自 1.3.2 版开始可用)
onDragLeave	target, source	在拖动一个节点离开某个目标节点并释放的时候触发, 返回 false 可以拒绝拖动。 target: 释放的目标节点元素。 source: 开始拖动的源节点。 (该事件自 1.3.2 版开始可用)
onBeforeDrop	target, source, point	在拖动一个节点之前触发, 返回 false 可以拒绝拖动。 target: 释放的目标节点元素。 source: 开始拖动的源节点。 point: 表示哪一种拖动操作, 可用值有: 'append', 'top' 或 'bottom'。 (该事件自 1.3.3 版开始可用)
onDrop	target, source, point	当节点位置被拖动时触发。 target: DOM 对象, 需要被拖动动的目标节点。 source: 源节点。 point: 表示哪一种拖动操作, 可用值有: 'append', 'top' 或 'bottom'。
onBeforeEdit	node	在编辑节点之前触发。
onAfterEdit	node	在编辑节点之后触发。
onCancelEdit	node	在取消编辑操作的时候触发。

方法

方法名	方法参数	描述
options	none	返回树控件属性。
loadData	data	读取树控件数据。
getNode	target	获取指定节点对象。
getData	target	获取指定节点数据, 包含它的子节点。
reload	target	重新载入树控件数据。
getRoot	nodeEl	获取通过 “nodeEl” 参数指定的节点的顶部父节点元素。 (该方法自 1.4 版开始可用)
getRoots	none	获取所有根节点, 返回节点数组。
getParent	target	获取父节点, 'target' 参数代表节点的 DOM 对象。

getChildren	target	获取所有子节点，'target' 参数代表节点的 DOM 对象。
getChecked	state	<p>获取所有选中的节点。'state' 可用值有：'checked'，'unchecked'，'indeterminate'。如果'state'未指定，将返回'checked'节点。</p> <p>代码示例：</p> <pre>var nodes = \$('#tt').tree('getChecked'); // 获取选中节点 var nodes = \$('#tt').tree('getChecked', 'unchecked'); // 获取未选择节点 var nodes = \$('#tt').tree('getChecked', 'indeterminate'); // 获取不确定的节点</pre> <p>译者注：(1.3.4 新增获取方式)</p> <pre>var nodes = \$('#tt').tree('getChecked', ['unchecked', 'indeterminate']);</pre>
getSelected	none	获取选择节点并返回它，如果未选择则返回 null。
isLeaf	target	判断指定的节点是否是叶子节点，target 参数是一个节点 DOM 对象。
find	id	<p>查找指定节点并返回节点对象。</p> <p>代码示例：</p> <pre>// 查找一个节点并选择它 var node = \$('#tt').tree('find', 12); \$('#tt').tree('select', node.target);</pre>
select	target	选择一个节点，'target' 参数表示节点的 DOM 对象。
check	target	选中指定节点。
uncheck	target	取消选中指定节点。
collapse	target	折叠一个节点，'target' 参数表示节点的 DOM 对象。
expand	target	展开一个节点，'target' 参数表示节点的 DOM 对象。在节点关闭或没有子节点的时候，节点 ID 的值(名为'id'的参数)将会发送给服务器请求子节点的数据。
collapseAll	target	折叠所有节点。
expandAll	target	展开所有节点。
expandTo	target	打开从根节点到指定节点之间的所有节点。
scrollTo	target	滚动到指定节点。 (该方法自 1.3.4 版开始可用)
append	param	<p>追加若干子节点到一个父节点，param 参数有 2 个属性：</p> <p>parent: DOM 对象，将要被追加子节点的父节点，如果未指定，子节点将被追加至根节点。</p> <p>data: 数组，节点数据。</p> <p>代码示例：</p> <pre>// 追加若干个节点并选中他们 var selected = \$('#tt').tree('getSelected'); \$('#tt').tree('append', { parent: selected.target, data: [{ id: 23, text: 'node23' }, { text: 'node24',</pre>

		<pre> state: 'closed', children: [{ text: 'node241' },{ text: 'node242' }] }] }); </pre>
toggle	target	打开或关闭节点的触发器，target 参数是一个节点 DOM 对象。
insert	param	<p>在一个指定节点之前或之后插入节点，'param' 参数包含如下属性：</p> <p>before: DOM 对象，在某个节点之前插入。</p> <p>after: DOM 对象，在某个节点之后插入。</p> <p>data: 对象，节点数据。</p> <p>下面的代码展示了如何将一个新节点插入到选择的节点之前：</p> <pre> var node = \$('#tt').tree('getSelected'); if (node) { \$('#tt').tree('insert', { before: node.target, data: { id: 21, text: 'node text' } }); } </pre> <p>译者注：（1.3.4 新增获取方式）</p> <pre> var node = \$('#tt').tree('getSelected'); if (node) { \$('#tt').tree('insert', { before: node.target, data: [{ id: 23, text: 'node23' }, { text: 'node24', state: 'closed', children: [{ text: 'node241' }, { text: 'node242' }] }]); } </pre>
remove	target	移除一个节点和它的子节点，'target' 参数是该节点的 DOM 对象。

pop	target	移除一个节点和它的子节点，该方法跟 remove 方法一样，不同的是它将返回被移除的节点数据。
update	param	<p>更新指定节点。'param' 参数包含以下属性： target (DOM 对象，将被更新的目标节点)，id, text, iconCls, checked 等。</p> <p>代码示例：</p> <pre>// 更新选择的节点文本 var node = \$('#tt').tree('getSelected'); if (node) { \$('#tt').tree('update', { target: node.target, text: 'new text' }); }</pre>
enableDnd	none	启用拖拽功能。
disableDnd	none	禁用拖拽功能。
beginEdit	target	开始编辑一个节点。
endEdit	target	结束编辑一个节点。
cancelEdit	target	取消编辑一个节点。
doFilter	text	<p>过滤操作，和 filter 属性功能类似。(该方法自 1.4.2 版开始可用)</p> <p>代码示例：</p> <pre>\$('#tt').tree('doFilter', 'easyui'); \$('#tt').tree('doFilter', ''); // 清除过滤器</pre>

TreeGrid (树形表格)

扩展自\$.fn.datagrid.defaults。使用\$.fn.treegrid.defaults 重写默认值对象。

树形表格用于显示分层数据表格。它是基于数据表格、组合树控件和可编辑表格。树形表格允许用户创建可定制的、异步展开行和显示在多列上的分层数据。

	Region	2009			2010			
		2st qrt.	3st qrt.	4st qrt.	1st qrt.	2st qrt.	3st qrt.	4st qrt.
1	Wyoming							
2	Albin	1800	1903	2183	2133	1923	2018	1838
3	Canon	1800	1903	2183	2133	1923	2018	1838
4	Egbert	1800	1903	2183	2133	1923	2018	1838
5	Washington							
6	Bellingham							
	Total	12600	13321	15281	14931	13461	14126	12866

依赖关系

- datagrid

使用案例

使用 HTML 标签创建树形表格。在大多数情况下，树形表格遵循数据表格的结构。

```

1. <table id="tt" class="easyui-treegrid" style="width:600px;height:400px"
2.     data-options="url:'get_data.php',idField:'id',treeField:'name'"
3.     <thead>
4.         <tr>
5.             <th data-options="field:'name',width:180">Task Name</th>
6.             <th data-options="field:'persons',width:60,align:'right'">Persons</th>
7.             <th data-options="field:'begin',width:80">Begin Date</th>
8.             <th data-options="field:'end',width:80">End Date</th>
9.         </tr>
10.    </thead>
11. </table>

```

使用 Javascript 创建树形表格。

```

1. <table id="tt" style="width:600px;height:400px"></table>
《jQuery EasyUI 简体中文 API 文档》

```

```

1. $('#tt').treegrid({
2.     url:'get_data.php',
3.     idField:'id',
4.     treeField:'name',
5.     columns:[[
6.         {title:'Task Name',field:'name',width:180},
7.         {field:'persons',title:'Persons',width:60,align:'right'},
8.         {field:'begin',title:'Begin Date',width:80},
9.         {field:'end',title:'End Date',width:80}
10.    ]]
11. });

```

属性

树形表格扩展自 datagrid(数据表格)，树形表格新增的属性如下：

属性名	类型	描述	默认值
idField	string	定义关键字段来标识树节点。 (必须的)	null
treeField	string	定义树节点字段。 (必须的)	null
animate	boolean	定义在节点展开或折叠的时候是否显示动画效果。	false
checkbox	boolean, function	定义在每一个节点前显示复选框，也可以指定一个函数来动态指定是否显示复选框，但函数返回 true 的时候则显示，否则不显示。 (该属性自 1.4.5 版开始可用) 代码示例： <pre> \$('#tg').treegrid({ checkbox: function(row) { var names = 'Java','eclipse.exe','eclipse.ini']; if (\$.inArray(row.name, names)>=0) { return true; } } }); </pre>	false
cascadeCheck	boolean	定义是否级联检查。 (该属性自 1.4.5 版开始可用)	true
onlyLeafCheck	boolean	定义是否仅在叶子节点前显示复选框。 (该属性自 1.4.5 版开始可用)	false
lines	boolean	定义是否显示 treegrid 行。 (该属性自 1.4.5 版开始可用)	false
loader	function(param, success, error)	定义以何种方式从远程服务器读取数据。返回 false 可以忽略该动作。该函数具有一下参数： param: 传递到远程服务器的参数对象。 success(data): 当检索数据成功的时候调用的回调函数。	json loader

		error()：当检索数据失败的时候调用的回调函数。	
loadFilter	function(data, parentId)	返回过滤后的数据进行展示。	

事件

树形表格的事件扩展自 datagrid(数据表格)，树形表格新增的时间如下：

事件名	事件参数	描述
onClickRow	row	在用户点击节点的时候触发。
onDbClickRow	row	在用户双击节点的时候触发。
onClickCell	field, row	在用户点击单元格的时候触发。
onDbClickCell	field, row	在用户双击单元格的时候触发。
onBeforeLoad	row, param	在请求数据加载之前触发，返回 false 可以取消加载动作。
onLoadSuccess	row, data	数据加载完成之后触发。
onLoadError	arguments	数据加载失败的时候触发，参数和 jQuery 的 \$.ajax() 函数的 'error' 回调函数一样。
onBeforeSelect	row	在用户选择一行之前触发，返回 false 则取消该动作。 (该事件自 1.4 版开始可用)
onSelect	row	在用户选择的时候触发，返回 false 则取消该动作。 (该事件自 1.4 版开始可用)
onBeforeUnselect	row	在用户取消选择一行之前触发，返回 false 则取消该动作。 (该事件自 1.4 版开始可用)
onUnselect	row	在用户取消选择的时候触发，返回 false 则取消该动作。 (该事件自 1.4 版开始可用)
onBeforeCheckNode	row, checked	在用户选中一行节点之前触发，返回 false 则取消该动作。 (该事件自 1.4.5 版开始可用)
onCheckNode	row, checked	在用户选中一行节点的时候触发，返回 false 则取消该动作。 (该事件自 1.4.5 版开始可用)
onBeforeExpand	row	在节点展开之前触发，返回 false 可以取消展开节点的动作。
onExpand	row	在节点被展开的时候触发。
onBeforeCollapse	row	在节点折叠之前触发，返回 false 可以取消折叠节点的动作。
onCollapse	row	在节点被折叠的时候触发。
onContextMenu	e, row	在右键点击节点的时候触发。
onBeforeEdit	row	在用户开始编辑节点的时候触发。
onAfterEdit	row, changes	在用户完成编辑的时候触发。
onCancelEdit	row	在用户取消编辑节点的时候触发。

方法

很多方法都使用 'id' 命名参数，而 'id' 参数代表树节点的值。

方法名	方法参数	描述
options	none	返回树形表格的属性。
resize	options	设置树形表格大小，options 包含 2 个属性： width: 树形表格的新宽度。 height: 树形表格的新高度。
fixRowHeight	id	修正指定的行高。
load	param	读取并显示首页内容。 (该方法自 1.3.4 版开始可用) 代码示例： // 读取并发送请求参数 \$('#tg').treegrid('load', { q: 'abc', name: 'name1' });
loadData	data	读取树形表格数据。
reload	id	重新加载树形表格数据。如果 'id' 属性有值，将重新载入指定树形行，否则重新载入所有行。 代码示例： \$('#tt').treegrid('reload', 2); // 重新载入值为 2 的行 \$('#tt').treegrid('reload'); // 重新载入所有行
reloadFooter	footer	重新载入页脚数据。
getData	none	获取载入数据。
getFooterRows	none	获取页脚数据。
getRoot	none	获取根节点，返回节点对象。
getRoots	none	获取所有根节点，返回节点数组。
getParent	id	获取父节点。
getChildren	id	获取子节点。
getSelected	none	获取选择的节点并返回它，如果没有节点被选中则返回 null。
getSelections	none	获取所有选择的节点。
getLevel	id	获取指定节点等级。
find	id	查找指定节点并返回节点数据。
select	id	选择一个节点。
unselect	id	反选一个节点。
selectAll	none	选择所有节点。
unselectAll	none	反选所有节点。
unselectAll	none	反选所有节点。
unselectAll	none	反选所有节点。
collapse	id	折叠一个节点。

expand	id	展开一个节点。
collapseAll	id	折叠所有节点。
expandAll	id	展开所有节点。
expandTo	id	打开从根节点到指定节点之间的所有节点。
toggle	id	节点展开/折叠状态触发器。
append	param	<p>追加节点到一个父节点，'param' 参数包含如下属性： parent: 父节点 ID，如果未指定则追加到根节点。 data: 数组，节点数据。</p> <p>代码示例： // 追加若干节点到选中节点的后面 var node = \$('#tt').treegrid('getSelected'); \$('#tt').treegrid('append', { parent: node.id, // the node has a 'id' value that defined through 'idField' property data: [{ id: '073', name: 'name73' }] });</p>
insert	param	<p>插入一个新节点到指定节点。'param' 参数包含一下参数： before: 插入指定节点 ID 值之前。 after: 插入指定节点 ID 值之后。 data: 新节点数据。</p> <p>代码示例： // 插入一个新节点到选择的节点之前 var node = \$('#tt').treegrid('getSelected'); if (node) { \$('#tt').treegrid('insert', { before: node.id, data: { id: 38, name: 'name38' } }); }</p> <p>(该方法自 1.3.1 版开始可用)</p>
remove	id	移除一个节点和他的所有子节点。
pop	id	弹出并返回节点数据以及它的子节点之后删除。 (该方法自 1.3.1 版开始可用)
refresh	id	刷新指定节点。
update	param	<p>更新指定节点。'param' 参数包含以下属性： id: 要更新的节点的 ID。 row: 新的行数据。</p>

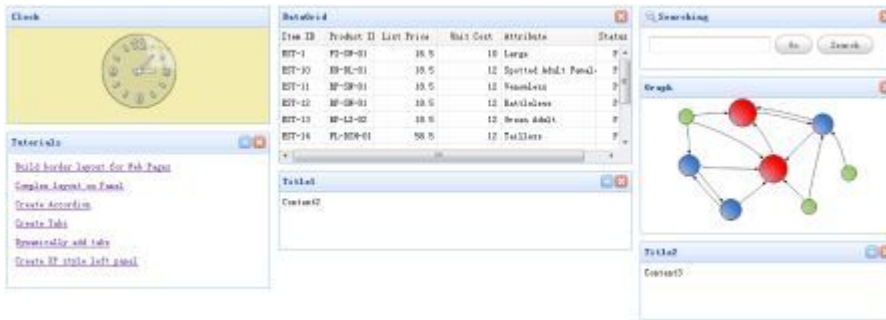
		<p>代码示例：</p> <pre> \$('#tt').treegrid('update', { id: 2, row: { name: '新名称', iconCls: 'icon-save' } }); </pre> <p>(该方法自 1.3.1 版开始可用)</p>
beginEdit	id	开始编辑一个节点。
endEdit	id	结束编辑一个节点。
cancelEdit	id	取消编辑一个节点。
getEditors	id	<p>获取指定行编辑器。每个编辑器都包含以下属性：</p> <p>actions: 编辑器执行的动作。</p> <p>target: 目标编辑器的 jQuery 对象。</p> <p>field: 字段名称。</p> <p>type: 编辑器类型。</p>
getEditor	param	<p>获取指定编辑器，'param' 参数包含 2 个属性：</p> <p>id: 行节点 ID。</p> <p>field: 字段名称。</p>
showLines	none	显示 treegrid 行。

第七章 扩展 (Extension)

Portal (门户)

[Extension](#) > Portal

扩展下载地址: <http://www.jeasyui.com/extension/downloads/jquery-easyui-portal.zip> (我发布的程序包也有提供, 在 extension 目录下)



第 1 步: 创建一个 HTML 页面

1. `<div id="pp" style="width:800px;height:500px;">`
2. `<div style="width:33%"></div>`
3. `<div style="width:33%"></div>`
4. `<div style="width:33%"></div>`
5. `</div>`

第 2 步: 创建一个门户

1. `$('#pp').portal(options);`

第 3 步: 在这个门户中添加面板部件

1. `// create the panel`
2. `var p = $('<div></div>').appendTo('body');`
3. `p.panel({`
4. `title: 'My Title',`
5. `height:150,`
6. `closable: true,`
7. `collapsible: true`
8. `});`
- 9.
10. `// add the panel to portal`
11. `$('#pp').portal('add', {`
12. `panel: p,`
13. `columnIndex: 0`
14. `});`

属性

属性名	类型	描述	默认值
width	number	门户宽度。	auto
height	number	门户高度。	auto
border	boolean	定义是否显示门户边框。	false
fit	boolean	当该属性为 true 时则设置门户大小自适应父容器。	false

事件

事件名	参数	描述
onStateChange	none	在用户拖拽面板的时候触发。
onResize	width, height	在门户大小改变的时候触发。

方法


方法名	参数	描述
options	none	返回属性对象。
resize	param	设置门户大小, 'param' 参数包含以下属性: width: 新的门户宽度。 height: 新的门户高度。
getPanels	columnIndex	获取指定列面板, 当列索引参数未指定的时候则返回所有面板。
add	param	添加一个新面板, 'param' 参数包含以下属性: panel: 添加的面板对象。 columnIndex: 插入的列索引。
remove	panel	移除和销毁指定面板。
disableDragging	panel	禁用面板拖拽功能。
enableDragging	panel	启用面板拖拽功能。

DataGrid View (数据表格展示)

[Extension](#) > DataGrid View

扩展下载地址: <http://www.jeasui.com/extension/downloads/jquery-easyui-datagridview.zip> (我发布的程序包整也有提供, 在 extension 目录下)

DataGrid DetailView (数据表格详细展示)

DataGrid - DetailView						
	Item ID	Product ID	List Price	Unit Cost	Attribute	Status
+	EST-1	FI-SW-01	16.5	10	Large	P
+	EST-2	K9-DL-01	18.5	12	Spotted Adult Female	P
+	EST-3	RP-SN-01	18.5	12	Venomless	P
-	EST-4	RP-SN-01	18.5	12	Rattleless	P
 Attribute: Rattleless Status: P						
+	EST-5	RP-LI-02	18.5	12	Green Adult	P
+	EST-6	FI-DSH-01	58.5	12	Tailless	P

第 1 步: 创建一个 HTML 页面

```

1. <head>
2.
3. <script type="text/javascript" src="datagrid-detailview.js"></script>
4.
5. </head>
6.
7. <body>
8.
9. <table id="tt"></table>
10.
11. </body>

```

第 2 步: 创建数据表格

```

1. $(' #tt').datagrid({
2.
3. title:'DataGrid - DetailView',
4.
5. width:500,
6.
7. height:250,
8.
9. remoteSort:false,
10.
11. singleSelect:true,

```

```
12.
13. nowrap:false,
14.
15. fitColumns:true,
16.
17. url:'datagrid_data.json',
18.
19. columns:[[
20.
21. {field:'itemid',title:'Item ID',width:80},
22.
23. {field:'productid',title:'Product ID',width:100,sortable:true},
24.
25. {field:'listprice',title:'List Price',width:80,align:'right',sortable:true},
26.
27. {field:'unitcost',title:'Unit Cost',width:80,align:'right',sortable:true},
28.
29. {field:'attr1',title:'Attribute',width:150,sortable:true},
30.
31. {field:'status',title:'Status',width:60,align:'center'}
32.
33. ]],
34.
35. view: detailview,
36.
37. detailFormatter: function(rowIndex, rowData) {
38.
39. return '<table><tr>' +
40.
41. '<td rowspan=2'
42. style="border:0"></td>' +
44.
45. '<td'
46. style="border:0">' +
47.
48. '<p>Attribute: ' + rowData.attr1 + '</p>' +
49.
50. '<p>Status: ' + rowData.status + '</p>' +
51.
52. '</td>' +
53.
54. '</tr></table>';
55.
56. }
57.
```

```
58. });
```

属性

属性名	类型	描述	默认值
detailFormatter	function(index, row)	detailFormatter 函数返回行详细内容。	

事件

事件名	参数	描述
onExpandRow	index, row	在展开行的时候触发。
onCollapseRow	index, row	在折叠行的时候触发。

方法

方法名	参数	描述
fixDetailRowHeight	index	修复明细行高度。
getExpander	index	获取行展开对象。
getRowDetail	index	获取明细内容。
expandRow	index	展开一行。
collapseRow	index	折叠一行。
subgrid	conf	<p>创建一个嵌套的子表格。“conf”参数有“option”和“subgrid”属性：</p> <ol style="list-style-type: none"> options: 定义数据表格的展现方式。子表格 options 对象有一个“foreignField”属性。foreign 值将会被发送到服务器端进行数据查询。 subgrid: 用于定义创建嵌套子表格的参数。 <p>代码示例：</p> <pre>var conf = { options: { //主表格参数 }, subgrid: { options: { foreignField: 'orderid', // 外键关联字段 // 其他表格参数 }, subgrid: { options: { foreignField: ..., // 其他表格参数 } } } }</pre>

		<pre> } } }; \$('#dg').datagrid().datagrid('subgrid', conf); </pre>
getParentGrid	none	获取父 datagrid 对象。
getParentRowIndex	none	获取父行索引。

DataGrid GroupView (数据表格分组展示)

DataGrid - GroupView						
	Product ID	List Price	Unit Cost	Attribute	Status	
K9-DL-01 - 1 Item(s)						
2	K9-DL-01	18.5	12	Spotted Adult Female	P	
RP-SN-01 - 2 Item(s)						
3	RP-SN-01	18.5	12	Venomless	P	
4	RP-SN-01	18.5	12	Rattleless	P	
RP-LI-02 - 1 Item(s)						
FL-DSH-01 - 2 Item(s)						
6	FL-DSH-01	58.5	12	Tailless	P	

第 1 步: 创建 HTML 页面

```

1. <head>
2. <script type="text/javascript" src="datagrid-groupview.js"></script>
3. </head>
4. <body>
5. <table id="tt"></table>
6. </body>

```

第 2 步: 创建数据表格

```

1. $('#tt').datagrid({
2.     title:'DataGrid - GroupView',
3.     width:500,
4.     height:250,
5.     rownumbers:true,
6.     remoteSort:false,
7.     nowrap:false,
8.     fitColumns:true,

```



```

9.     url:'datagrid_data.json',
10.     columns:[[
11.         {field:'productid',title:'Product ID',width:100,sortable:true},
12.         {field:'listprice',title:'List Price',width:80,align:'right',sortable:true},
13.         {field:'unitcost',title:'Unit Cost',width:80,align:'right',sortable:true},
14.         {field:'attr1',title:'Attribute',width:150,sortable:true},
15.         {field:'status',title:'Status',width:60,align:'center'}
16.     ]],
17.     groupField:'productid',
18.     view: groupview,
19.     groupFormatter:function(value, rows){
20.         return value + ' - ' + rows.length + ' Item(s)';
21.     }
22. });

```

属性

属性名	类型	描述	默认值
groupField	string	声明哪些字段分组。	
groupFormatter	function(value, rows)	groupFormatter 函数返回分组内容。value 参数指明了分组值定义的 'groupField' 属性。rows 参数指明了指定分组值的数据行。	
groupStyler	function(value, rows)	<p>函数返回 CSS 样式组。</p> <p>value 参数表明了通过 'groupField' 属性定义的分组值。</p> <p>rows 参数表明了根据分组值指定的数据行。</p> <p>代码示例:</p> <pre> \$(' #dg').datagrid({ groupStyler: function(value, rows) { if (value == 'RP-LI-02') { return 'background-color:#6293BB; color:#fff;'; // 返回行内 // the function can return predefined css class and inline style // return {class:'rl', style:{'color:#fff'}}; } } }); </pre>	

方法

方法名	参数	描述
expandGroup	groupIndex	展开一个分组。
collapseGroup	groupIndex	折叠一个分组。
scrollToGroup	groupIndex	滚动一个分组。

DataGrid BufferView（数据表格缓存视图）

DataGrid - BufferView							
	Inv No	Date	Name	Amount	Price	Cost	Note
690	Inv No 690	7/15/2012	Name 690	444	671	297924	Note 690
691	Inv No 691	7/15/2012	Name 691	657	865	568305	Note 691
692	Inv No 692	7/15/2012	Name 692	804	92	73968	Note 692
693	Inv No 693	7/15/2012	Name 693	625	135	84375	Note 693
694	Inv No 694	7/15/2012	Name 694	906	608	550848	Note 694
695	Inv No 695	7/15/2012	Name 695	360	393	141480	Note 695
696	Inv No 696	7/15/2012	Name 696	293	600	175800	Note 696
697	Inv No 697	7/15/2012	Name 697	166	309	51294	Note 697

第1步：导入 bufferview 的 js 文件

```
<head>
<script type="text/javascript" src="datagrid-bufferview.js"></script>
</head>
```

第2步：创建数据表格

```
1. <table id="tt" class="easyui-datagrid" style="width:700px;height:250px"
2. title="DataGrid - BufferView"
3. data-options="url:'get_data.php',view:bufferview,rownumbers:true,
   singleSelect:true,autoRowHeight:false,pagesize:50">
4. <thead>
5. <tr>
6. <th field="inv" width="80">Inv No</th>
7. <th field="date" width="100">Date</th>
8. <th field="name" width="80">Name</th>
9. <th field="amount" width="80" align="right">Amount</th>
10. <th field="price" width="80" align="right">Price</th>
11. <th field="cost" width="100" align="right">Cost</th>
12. <th field="note" width="110">Note</th>
13. </tr>
14. </thead>
15. </table>
```

DataGrid VirtualScrollView (数据表格虚拟滚动视图)

DataGrid - VirtualScrollView							
	Inv No	Date	Name	Amount	Price	Cost	Note
7555	Inv No 7555	7/15/2012	Name 7555	252	613	154476	Note 7555
7556	Inv No 7556	7/15/2012	Name 7556	120	212	25440	Note 7556
7557	Inv No 7557	7/15/2012	Name 7557	969	383	371127	Note 7557
7558	Inv No 7558	7/15/2012	Name 7558	5	325	1625	Note 7558
7559	Inv No 7559	7/15/2012	Name 7559	451	526	237226	Note 7559
7560	Inv No 7560	7/15/2012	Name 7560	886	695	615770	Note 7560
7561	Inv No 7561	7/15/2012	Name 7561	118	677	79886	Note 7561
7562	Inv No 7562	7/15/2012	Name 7562	605	712	430760	Note 7562

第 1 步: 导入滚动视图的 JS 文件

1. `<head>`
2. `<script type="text/javascript" src="datagrid-scrollview.js"></script>`
3. `</head>`

第 2 步: 创建包含虚拟滚动视图的数据表格

1. `<table id="tt" class="easyui-datagrid" style="width:700px;height:250px"`
2. `title="DataGrid -`
3. `VirtualScrollView"`
4. `data-`
`options='url:' get_data.php', view:scrollview, rownumbers:true, singleSelect:true, autoRo`
`wHeight:false, pageSize:50">`
5. `<thead>`
6. `<tr>`
7. `<th field="inv" width="80">Inv No</th>`
8. `<th field="date" width="100">Date</th>`
9. `<th field="name" width="80">Name</th>`
10. `<th field="amount" width="80" align="right">Amount</th>`
11. `<th field="price" width="80" align="right">Price</th>`
12. `<th field="cost" width="100" align="right">Cost</th>`
13. `<th field="note" width="110">Note</th>`
14. `</tr>`
15. `</thead>`
16. `</table>`


方法

方法名	参数	描述
<code>gotoPage</code>	page	跳转到指定页面。
<code>scrollTo</code>	index	滚动视图到指定的行。
<code>fixDetailRowHeight</code>	index	固定明细行行高。
<code>getExpander</code>	index	获取展开对象。
<code>getRowDetail</code>	index	获取明细内容。
<code>expandRow</code>	index	展开一行。
<code>collapseRow</code>	index	折叠一行。

Editable DataGrid (可编辑数据表格)

[Extension](#) > Editable DataGrid

扩展下载地址: <http://www.jeasyui.com/extension/downloads/jquery-easyui-edatagrid.zip> (我发布的程序包整也有提供, 在 extension 目录下)

Editable DataGrid					
Item ID	Product ID	List Price	Unit Cost	Attribute	
EST-1	FI-SW-01	16.5	10	Large	
EST-2	K9-DL-01	18.5	12	Spotted Adult Female	
EST-3	RF-SN-01	18.5	12	Venomless	
EST-5	RP-LI-02	18.5	12	Green Adult	
EST-6	FL-DSH-01	58.5	12	Tailless	
EST-7	FL-DSH-01	23.5	12	With tail	
EST-8	FL-DLH-02	93.5	12	Adult Female	

使用案例

在 HTML 标签里面创建数据表格

```

1. <table id="tt" style="width:600px;height:200px"
2.     title="Editable DataGrid"
3.     singleSelect="true">
4.     <thead>
5.         <tr>
6.             <th field="itemid" width="100" editor="{type:'validatebox',options:{required:true}}">Item ID</th>
7.             <th field="productid" width="100" editor="text">Product ID</th>
8.             <th field="listprice" width="100" align="right" editor="{type:'numberbox',options:{precision:1}}">List Price</th>
9.             <th field="unitcost" width="100" align="right" editor="numberbox">Unit Cost</th>
10.            <th field="attr1" width="150" editor="text">Attribute</th>
11.        </tr>
12.    </thead>
13. </table>

```

创建编辑器

```

1. $('#tt').datagrid({
2.     url: 'datagrid_data.json',
3.     saveUrl: ...,
4.     updateUrl: ...,
5.     destroyUrl: ...
6. });

```

现在你可以通过双击数据表格的行开始编辑操作。你也可以设置 `saveUrl`, `updateUrl`, `destroyUrl` 属性来自动从客户端同步数据到服务器端。

属性

可编辑表格控件的熟悉扩展自 `DataGrid`(数据表格)，可编辑表格控件新增的属性如下：

属性名	类型	描述	默认值
destroyMsg	object	销毁行的时候显示的确认对话框消息。	<pre>destroyMsg:{ norecord:{ // 在没有记录选择的时候执行 title:'Warning', msg:'No record is selected.' }, confirm:{ // 在选择一行的时候执行 title:'Confirm', msg:'Are you sure you want to delete?' } }</pre>
autoSave	boolean	设置为 true 时，在点击表格外部的时候自动保存编辑的行。	false
url	string	通过 URL 向服务器检索数据。	null
saveUrl	string	通过 URL 保存数据到服务器并返回添加的行。	null
updateUrl	string	通过 URL 更新数据到服务器并返回更新的行。	null
destroyUrl	string	通过 URL 将 'id' 参数发送到服务器以销毁行。	null
tree	selector	树选择器指示相对应的树控件。	null
treeUrl	string	通过 URL 检索树控件数据。	null
treeDndUrl	string	通过 URL 处理拖拽操作。	null
treeTextField	string	定义树的文本字段名称。	name
treeParentField	string	定义树的父节点字段名。	parentId

事件

可编辑表格的事件扩展自 `DataGrid`(数据表格)，可编辑表格新增的事件如下：

事件名	参数	描述
onAdd	index, row	在添加新行的时候触发。
onEdit	index, row	在编辑一行数据的时候触发。
onBeforeSave	index	在保存一行记录之前触发，返回 false 可以取消保存操作。
onSave	index, row	在保存一行记录时触发。
onDestroy	index, row	在销毁一行记录时触发。
onError	index, row	<p>在服务器返回错误的时候触发。服务器端将返回一个包含” isError” 属性的 json 字符串，当该参数等于 true 时，则表示请求发生了错误。</p> <p>代码示例：</p> <pre>// 服务器端代码 echo json_encode(array('isError' => true, 'msg' => '请求发生了错误。')); // 客户端代码 \$('#dg').datagrid({ onError: function(index,row){ alert(row.msg); } });</pre>

方法

可编辑表格的方法扩展自 DataGrid(数据表格)，可编辑表格新增或重写的方法如下：

方法名	参数	描述
options	none	返回属性对象。
enableEditing	none	启用数据表格编辑。
disableEditing	none	禁用数据表格编辑。
editRow	index	编辑指定行。
addRow	none	<p>添加一个新的空行。</p> <p>代码示例：</p> <pre>// 追加一个空行 \$('#dg').datagrid('addRow'); // 在第一行追加一个空行 \$('#dg').datagrid('addRow', 0); // 新增一个带默认值的行 \$('#dg').datagrid('addRow', { index: 2, row: { name: 'name1', addr: 'addr1' } });</pre>

		<pre> } }); </pre>
saveRow	none	保存编辑行并发送到服务器。
cancelRow	none	取消编辑行。
destroyRow	none	<p>销毁当前选择的行。</p> <p>代码示例：</p> <pre> // 销毁所有选择的行 \$('#dg').datagrid('destroyRow'); // 销毁首行 \$('#dg').datagrid('destroyRow', 0); // 销毁指定的行 \$('#dg').datagrid('destroyRow', [3, 4, 5]); </pre>

Cell Editing in DataGrid (单元格编辑表格)

[Extension](#) > Cell Editing in DataGrid

扩展下载地址: <http://www.jeasyui.com/extension/downloads/datagrid-cellediting.zip> (我发布的程序包也有提供, 在 extension 目录下)

Cell Editing in DataGrid						
Item ID	Product	List Price	Unit Cost	Attribute	Status	
EST-1	FI-SW-01	36.5	10	Large	P	
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P	
EST-11	RP-SN-01	38.5	12	Venomless	P	
EST-12	RP-SN-01	26.5	12	Rattleless	P	
EST-13	RP-LI-02	35.5	12	Green Adult	P	
EST-14	FL-DSH-01	158.5	12	Tailless	P	

导入 “datagrid-cellediting.js” 文件

```
1. <script type="text/javascript" src="datagrid-cellediting.js"></script>
```

启用单元格编辑器

```

1. $('#dg').datagrid({
2.     data: data
3. }).datagrid('enableCellEditing').datagrid('gotoCell', {
4.     index: 0,
5.     field: 'productid'

```


6. });

属性

属性表格的属性扩展自 datagrid(数据表格)，属性表格新增的属性如下：

属性名	属性类型	描述	默认值
clickToEdit	boolean	设置为 true 时，则会在点击该单元格时启用编辑功能。	true
dblclickToEdit	boolean	设置为 true 时，则会在双击该单元格时启用编辑功能。	false

事件

事件扩展自 datagrid，以下是本插件新增事件。

事件名	参数	描述
onBeforeCellEdit	index, field	在编辑单元格之前的时候触发，返回 false 将取消该动作。
onCellEdit	index, field, value	在编辑单元格的时候触发，返回 false 将取消该动作。参数包含： index: 编辑行索引 field: 编辑行名称 value: 按下键盘上的字符代码初始化值。在按下 Del 或 Backspace 键的时候这个值是一个空字符串。
onSelectCell	index, field	在选择一个单元格的时候触发，返回 false 将取消该动作。
onUnselectCell	index, field	在取消选择一个单元格的时候触发，返回 false 将取消该动作。

方法

方法扩展自 datagrid，以下是本插件新增方法。

方法名	参数	描述
editCell	param	<p>编辑一个单元格。“param”参数包含以下属性：</p> <p>index: 行索引 field: 字段名</p> <p>代码示例：</p> <pre>\$('#dg').datagrid('editCell', { index: 0, field: 'productid' });</pre>

gotoCell	param	<p>跳转到高亮行。可用参数包括：'up'，'down'，'left'，'right' 或者包含 'index' 和 'field' 的对象。</p> <p>代码示例：</p> <pre>\$('#dg').datagrid('gotoCell', 'down'); \$('#dg').datagrid('gotoCell', { index: 0, field: 'productid' });</pre>
enableCellSelecting	none	启用 datagrid 的单元格选择。
disableCellSelecting	none	禁用 datagrid 的单元格选择。
enableCellEditing	none	在数据表格中启用单元格编辑器。
disableCellEditing	none	在数据表格中禁用单元格编辑器。
input	none	返回当前编辑器框对象。
cell	none	返回当前包含 'index' 和 'field' 属性的单元格信息。
getSelectedCells	none	返回所有选择的单元格。

Columns Extension for DataGrid (列扩展表格)

[Extension](#) > Columns Extension for DataGrid

扩展下载地址: <http://www.jeasyui.com/extension/downloads/columns-ext.zip> (我发布的程序包也有提供, 在 `extension` 目录下)

Moving Columns						
	Item ID	List Price	Product	Unit Cost	Attribute	Status
1	EST-1	36.5	FI-SW-01	✓ List Price	large	P
2	EST-10	18.5	K9-DL-01	12	Spotted Adult Female	P
3	EST-11	38.5	RP-SN-01	12	Venomless	P
4	EST-12	26.5	RP-SN-01	12	Rattleless	P
5	EST-13	35.5	RP-LI-02	12	Green Adult	P
6	EST-14	158.5	FL-DSH-01	12	Tailless	P
7	EST-15	83.5	FL-DSH-01	12	With tail	P
8	EST-16	23.5	FL-DLH-02	12	Adult Female	P

导入 'columns-ext.js' 文件

```
1. <script type="text/javascript" src="columns-ext.js"></script>
```

事件

事件扩展自 `datagrid`, 以下是本插件新增事件。

事件名	参数	描述
<code>onBeforeDragColumn</code>	<code>field</code>	在拖拽一列的之前触发, 返回 <code>false</code> 的时候取消该动作。
<code>onStartDragColumn</code>	<code>field</code>	在开始拖拽一列的时候触发, 返回 <code>false</code> 的时候取消该动作。
<code>onStopDragColumn</code>	<code>field</code>	在停止拖拽一列的时候触发, 返回 <code>false</code> 的时候取消该动作。
<code>onBeforeDropColumn</code>	<code>toField</code> , <code>fromField</code> , <code>point</code>	在释放一列之前触发, 返回 <code>false</code> 的时候取消该动作。 <code>toField</code> : 释放的目标列。 <code>fromField</code> : 来源列。 <code>point</code> : 指明释放操作, 可用值包含: 'before' 或 'after'。
<code>onDropColumn</code>	<code>toField</code> , <code>fromField</code> , <code>point</code>	在释放一列的时候触发, 返回 <code>false</code> 的时候取消该动作。 <code>toField</code> : 释放的目标列。 <code>fromField</code> : 来源列。 <code>point</code> : 指明释放操作, 可用值包含: 'before' 或 'after'。

方法

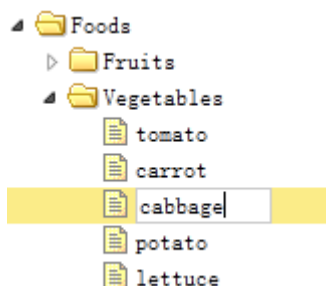
可编辑树扩展自树控件，可扩展树的新增方法如下：

方法名	方法参数	描述
columnMoving	none	启用列移动，该方法允许用户拖拽表格的列。
freezeColumn	field	冻结一个列。
unfreezeColumn	field	解冻一个列。
moveColumn	param	<p>移动一个列。</p> <p>代码示例：</p> <pre>\$('#dg').datagrid('moveColumn', { field: 'itemid', before: 'listprice' // after: 'listprice' });</pre>

Editable Tree (可编辑树)

[Extension](#) > Editable Tree

扩展下载地址: <http://www.jeasyui.com/extension/downloads/jquery-easyui-etree.zip> (我发布的程序包也有提供, 在 extension 目录下)



创建树

```
1. <ul id="tt"></ul>
2. $('#tt').etree({
3.     url: 'tree_data.json',
4.     createUrl: ...,
5.     updateUrl: ...,
6.     destroyUrl: ...,
7.     dndUrl: ...
8. });
```

设置 url, createUrl, updateUrl, destroyUrl 和 dndUrl 属性将自动从客户端同步数据到服务器端。

- url: 返回树控件的数据。
- createUrl: 在创建一个新节点的时候, 树控件将父节点 ID 赋值给一个名为 'parentId' 的参数并发送到服务器。服务器将会返回对应节点数据。下面的代码显示了添加节点数据的例子:
 - {"id":1,"text":"new node"}
- updateUrl: 在更新一个节点的时候, 树控件将会发送 'id' 和 'text' 参数到服务器。服务器会执行更新并返回更新的节点数据。
- destroyUrl: 在销毁一个节点的时候, 树控件将会发送 'id' 参数到服务器。服务器会返回 {"success":true} 这样的 JSON 字符串数据。
- dndUrl: 在拖拽一个节点的时候, 树控件将会发送如下参数到服务器:
 - id - 拖拽的节点 ID。
 - targetId - 拖拽到的节点 ID。
 - point - 指明释放操作, 可用值有: 'append', 'top' 或 'bottom'。
 服务器会做一些动作并返回 {"success":true} 这样的 JSON 字符串数据。

方法

可编辑树扩展自树控件，可扩展树的新增方法如下：

方法名	方法参数	描述
options	none	返回属性对象。
create	none	创建一个新节点。
edit	none	编辑当前选中的节点。
destroy	none	销毁当前选中的节点。

DataGrid Filter Row（可过滤行的数据表格）

[Extension](#) > DataGrid Filter Row

扩展下载地址: <http://www.jeasyui.com/extension/downloads/datagrid-filter.zip>（我发布的程序包整也有提供，在 extension 目录下）

DataGrid						
Item ID	Product	List Price		Unit Cost		Status
		132.0	Y	12.0	Y	All
EST-1	FI-SW-01					P
EST-10	K9-DL-01					P
EST-11	RP-SN-01					P
EST-12	RP-SN-01					P
EST-13	RP-LI-02					P
EST-15	FL-DSH-01	83.5		12	With tail	P
EST-16	FL-DLH-02	23.5		12	Adult Female	P

导入'datagrid-filter.js' 文件

```
1. <script type="text/javascript" src="datagrid-filter.js"></script>
```

启用过滤

```
1. var dg = $('#dg');
2. dg.datagrid(); // 创建数据表格
3. dg.datagrid('enableFilter'); // 启用过滤
```

属性

下列属性扩展自 Datagrid，以下是新增的属性。.

属性名	类型	描述	默认值
filterMenuIconCls	string	用来表示选择了哪个过滤器菜单项的图标 class 名称。	icon-ok
filterBtnIconCls	string	过滤器按钮的图标 class 名称。	icon-filter
filterBtnPosition	string	过滤器按钮的位置。可用值有：'left' 和 'right'	right
filterPosition	string	过滤器栏的可折叠菜单显示位置。可用值有：'top' 和 'bottom'	bottom
showFilterBar	boolean	设置为 true 时，显示过滤器栏。	true
remoteFilter	boolean	设置为 true 时，启用远程过滤。在启用远程过滤的时候	false

		'filterRules' 参数将会发送到远程服务器。'filterRules' 参数值是从 'filterStringify' 函数获取的返回值。	
filterDelay	number	延迟过滤 'text' 过滤器组件中最后一次键盘输入事件。	400
filterRules	array	过滤器规则定义。每个规则都包含 'field', 'op' 和 'value' 属性。	[]
filterMatchingType	string	指定过滤行是否需要匹配全部或部分应用过滤器。可用值有: "all", "any"	all
defaultFilterType	string	默认过滤类型。	text
defaultFilterOperator	string	默认过滤操作器。	contains
defaultFilterOptions	object	默认过滤选项。	
filterStringify	function	字符串化过滤器规则的函数。	<pre>function(data) { returnJSON.stringify(data); }</pre>

事件

下列事件扩展自 Datagrid，以下是新增的事件。

方法名	方法参数	描述
onClickMenu	item, button, field	在点击菜单项的时候触发，返回 false 取消该动作。 item: 点击的菜单项。 button: 绑定到过滤器的过滤按钮。 field: 字段名。

方法

下列方法扩展自 Datagrid，以下是新增的方法。

方法名	方法参数	描述
enableFilter	filters	创建并启用过滤功能。'filters' 参数是一个过滤器配置数组。每一个项目都包含以下属性： 1) field: 自定义规则的字段名。 2) type: 过滤类型，可用值有： label, text, textarea, checkbox, numberbox, validatebox, datebox, combobox, combotree。 3) options: 过滤类型参数。 4) op: 过滤条件：可用值有： contains, equal, notequal, beginwith, endwith, less, lessorequal, greater, greaterorequal。

		<p>代码示例：</p> <pre> \$('#dg').datagrid('enableFilter'); \$('#dg').datagrid('enableFilter', [{ field: 'listprice', type: 'numberbox', options: {precision: 1}, op: ['equal', 'notequal', 'less', 'greater'] }]); </pre>
disableFilter	none	禁用过滤功能。
destroyFilter	none	销毁过滤器栏。
getFilterRule	field	获取过滤规则。
addFilterRule	param	<p>添加一个过滤规则。</p> <pre> \$('#dg').datagrid('addFilterRule', { field: 'desp', op: 'contains', value: 'easyui' }); </pre>
removeFilterRule	field	远程过滤规则。如果 'field' 参数未指定，将删除所有过滤规则。
doFilter	none	基于一些过滤规则进行过滤。
getFilterComponent	field	根据指定的字段获取过滤器。
resizeFilter	field	调整过滤器大小。

Drag and Drop Rows in DataGrid (可拖放行的数据表格)

[Extension](#) > Drag and Drop Rows in DataGrid

扩展下载地址: <http://www.jeasyui.com/extension/downloads/datagrid-dnd.zip> (我发布的程序包也有提供, 在 `extension` 目录下)

DataGrid						
Item ID	Product	List Price	Unit Cost	Attribute	Status	
EST-1	FL-SW-01	36.5	10	Large	P	
EST-10	K9-DL-01	18.5	12	Spotted Adult Female	P	
EST-11	RP-SN-01	38.5	12	Venomless	P	
EST-12	RP-SN-01	26.5	12	Rattleless	P	
EST-13	✓ EST-11	RP-SN-01	38.5	12 Venomless	P	
EST-14	FL-DSH-01	158.5	12	Tailless	P	
EST-15	FL-DSH-01	83.5	12	With tail	P	
EST-16	FL-DLH-02	23.5	12	Adult Female	P	

导入 'datagrid-dnd.js' 文件

```
1. <script type="text/javascript" src="datagrid-dnd.js"></script>
```

启用拖放

```
1. <table class="easyui-datagrid" title="DataGrid" style="width:700px;height:250px"
  data-options="
2.   singleSelect:true,
3.   data:data,
4.   onLoadSuccess:function() {
5.     $(this).datagrid('enableDnd');
6.   }
7. ">
8. <thead>
9. <tr>
10. <th data-options="field:'itemid',width:80">Item ID</th>
11. <th data-options="field:'productid',width:100">Product</th>
12. <th data-options="field:'listprice',width:80,align:'right'">List Price</th>
13. <th data-options="field:'unitcost',width:80,align:'right'">Unit Cost</th>
14. <th data-options="field:'attr1',width:250">Attribute</th>
15. <th data-options="field:'status',width:60,align:'center'">Status</th>
16. </tr>
17. </thead>
18. </table>
```

拖动多个行

```

1. $('#dg').datagrid({
2.   singleSelect: false,
3.   dragSelection: true,
4.   onLoadSuccess: function() {
5.     $(this).datagrid('enableDnd');
6.   }
7. });

```

属性

下列属性扩展自 Datagrid，以下是新增的属性。

事件名	参数	描述	默认值
dropAccept	selector	确定哪些行允许被拖拽。	tr.datagrid-row
dragSelection	boolean	设置为 true 时，拖拽所有选中的行，否则只拖拽单行。	false

事件

下列事件扩展自 Datagrid，以下是新增的事件。

事件名	参数	描述
onBeforeDrag	row	在行开始拖动之前触发，返回 false 拒绝拖动。
onStartDrag	row	在开始拖动行的时候触发。
onStopDrag	row	在停止拖动行的时候触发。
onDragEnter	targetRow, sourceRow	在行被拖动到目标行内触发，返回 false 拒绝拖动。
onDragOver	targetRow, sourceRow	在行悬停在目标行内时触发，返回 false 拒绝拖动。
onDragLeave	targetRow, sourceRow	在行被拖动到目标行内触发。
onBeforeDrop	targetRow, sourceRow, point	在行被释放前触发，返回 false 拒绝释放。 targetRow: 要释放的目标行。 sourceRow: 被拖动的原始行。 point: 指明拖放操作，可用值有: 'top', 'bottom'。
onDrop	targetRow, sourceRow, point	在行被释放的时候触发。 targetRow: 要释放的目标行。 sourceRow: 被拖动的原始行。 point: 指明拖放操作，可用值有: 'top', 'bottom'。

方法

下列方法扩展自 Datagrid，以下是新增的方法。

方法名	方法参数	描述
enableDnd	index	<p>启用拖放行功能。'index' 参数表明什么行被拖放。如果该参数未指定将启用所有行的拖放功能。</p> <p>代码示例：</p> <pre>\$('#dg').datagrid('enableDnd', 1); // 启用第二行的拖放 \$('#dg').datagrid('enableDnd'); // 启用所有行的拖放</pre>

Drag and Drop Rows in TreeGrid (可拖放的树形表格)

[Extension](#) > Drag and Drop Rows in TreeGrid

扩展下载地址: <http://www.jeasyui.com/extension/downloads/treegrid-dnd.zip> (我发布的程序包也有提供, 在 `extension` 目录下)

Folder Browser			
	Name	Size	Modified Date
1	▲ C		02/19/2010
2	▲ Program Files	120 MB	03/20/2010
3	▶ Java		01/13/2010
6	▲ MySQL <input checked="" type="checkbox"/> my-huge.ini		01/13/2010
7	my.ini	10 KB	02/26/2009
8	my-huge.ini	5 KB	02/26/2009
9	my-large.ini	5 KB	02/26/2009
10	eclipse.ini	1 KB	04/20/2010

导入 'treegrid-dnd.js' 文件

```
1. <script type="text/javascript" src="treegrid-dnd.js"></script>
```

启用拖放

```
1. <table title="Folder Browser" class="easyui-treegrid"
   style="width:700px;height:250px"
2. data-options="
3. data: data,
4. rownumbers: true,
5. idField: 'id',
6. treeField: 'name',
7. onSuccess: function(row) {
8. $(this).treegrid('enableDnd', row?row.id:null);
9. }
10. ">
11. <thead>
12. <tr>
13. <th data-options="field:'name' " width="220">Name</th>
14. <th data-options="field:'size' " width="150" align="right">Size</th>
15. <th data-options="field:'date' " width="200">Modified Date</th>
16. </tr>
17. </thead>
18. </table>
```

属性

下列属性扩展自 Datagrid，以下是新增的属性。

事件名	参数	描述	默认值
dropAccept	selector	确定哪些行允许被拖拽。	tr[node-id]

事件

下列事件扩展自 Treegrid，以下是新增的事件。

事件名	参数	描述
onBeforeDrag	row	在行开始拖动之前触发，返回 false 拒绝拖动。
onStartDrag	row	在开始拖动行的时候触发。
onStopDrag	row	在停止拖动行的时候触发。
onDragEnter	targetRow, sourceRow	在行被拖动到目标行内触发，返回 false 拒绝拖动。
onDragOver	targetRow, sourceRow	在行悬停在目标行内时触发，返回 false 拒绝拖动。
onDragLeave	targetRow, sourceRow	在行被拖动到目标行内触发。
onBeforeDrop	targetRow, sourceRow, point	在行被释放前触发，返回 false 拒绝释放。 targetRow: 要释放的目标行。 sourceRow: 被拖动的原始行。 point: 指明拖放操作，可用值有: 'top', 'bottom'。
onDrop	targetRow, sourceRow, point	在行被释放的时候触发。 targetRow: 要释放的目标行。 sourceRow: 被拖动的原始行。 point: 指明拖放操作，可用值有: 'top', 'bottom'。

方法

下列方法扩展自 Treegrid，以下是新增的方法。

方法名	方法参数	描述
enableDnd	id	启用拖放行功能。'id' 参数表明什么行被拖放。如果该参数未指定将启用所有行的拖放功能。

PivotGrid（数据分析表格）

[Extension](#) > PivotGrid

扩展下载地址: <http://www.jeasyui.com/extension/downloads/jquery-easyui-pivotgrid.zip> (我发布的程序包整也有提供, 在 extension 目录下)

PivotGrid						
Layout						
	red		yellow		white	
	Price	Discount	Price	Discount	Price	Discount
▲ 文件夹 Australia	9441610	2055	6780	255	64372	100
📄 Accessories	2088	66	108	98	0	0
📄 Bikes	0	0	0	0	64372	100
📄 Clothing	21443	221	6672	157	0	0
📄 Components	59292	840	0	0	0	0
📄 Cars	9358787	928	0	0	0	0
▲ 文件夹 Canada	249500	2425	22157	203	5166914	3685
📄 Accessories						

使用案例

创建数据分析表格

```

1. <table id="pg"></table>

1. $('#pg').pivotgrid({
2.     title:'PivotGrid',
3.     toolbar:[{
4.         text:'Layout',
5.         handler:function() {
6.             $('#pg').pivotgrid('layout');
7.         }
8.     ]],
9.     width:700,
10.    height:300,
11.    url:'data.json',
12.    method:'get',
13.    pivot:{
14.        rows:['Country','Category'],
15.        columns:['Color'],
16.        values:[
17.            {field:'Price',op:'sum'},
18.            {field:'Discount',op:'sum'}
19.        ]

```

```

20.     },
21.     valueStyler:function(value,row,index){
22.         if (/Discount$/.test(this.field) && value>100 && value<500){
23.             return 'background:#D8FFD8'
24.         }
25.     }
26. });

```

属性

属性扩展自 `treegrid`，以下是 `pivotgrid` 新增属性。

属性名	类型	描述	默认值
forzenColumnTitle	string	冻结列标题。	
valueFieldWidth	number	值字段列宽度。	80
valuePrecision	number	值字段精确度。	0
valueStyler	function	值字段单元格样式函数。	
valueFormatter	function	值字段单元格格式化函数。	
pivot	object	数据分析表格配置，包含如下配置项： rows: 数组，表格纵向展示的行。 columns: 数组，表格横向展示的列。 values: 数组，显示在列的值字段。 filters: 数组，过滤器。 filterRules: 对象，过滤器规则。	
i18n	object	国际化配置，包含如下配置项： fields: 字段的标题。 filters: 过滤器的标题。 rows: 行的标题。 columns: 列的标题。 ok: 确定按钮的标题。 cancel: 取消按钮的标题。	
operators	object	值字段的操作符，可用值有：'sum'，'count'，'max'，'min'。 代码示例： // 实现 'sum' 操作 \$.extend(\$.fn.pivotgrid.defaults.operators, { sum: function(rows, field){ var opts = \$(this).pivotgrid('options'); var v = 0; \$.map(rows, function(row) { v += parseFloat(row[field]) 0; }); return v.toFixed(opts.valuePrecision); } });	

		}	
		});	

事件

事件扩展自 treegrid。

方法

方法扩展自 treegrid，以下是 pivotgrid 新增方法。

方法名	参数	描述
options	none	返回属性对象。
getData	none	获取加载数据。
layout	none	打开布局对话框，允许用户在运行时更改数据立方体。

DWR Loader (DWR 装载器)

[Extension](#) > DWR 装载器

扩展下载地址: <http://www.jeasyui.com/extension/downloads/jquery-easyui-dwrloader.zip> (我发布的程序包整也有提供, 在 extension 目录下)

导入'dwrloader.js'文件

要让 EasyUI 使用 DWR 来获取数据, 首先需要导入 'dwrloader.js' 文件。

```
1. <script type="text/javascript" src="../../jquery-1.7.2.min.js"></script>
2. <script type="text/javascript" src="../../jquery.easyui.min.js"></script>
3. <script type="text/javascript" src="dwrloader.js"></script>
```

指派 DWR 方法给 'url' 属性

作为默认 json 装载器, 'url' 属性指明远程 URL 来获取 JSON 数据。在使用 DWR 装载器的时候, 我们应该指定一个函数来从 DWR 的 'url' 属性检索数据。下面的示例展示如何通过使用 DWR 加载器来显示一个 datagrid。

```
1. <table id="dg"></table>
2. $(function() {
3.     $('#dg').datagrid({
4.         columns: [[
5.             {field:"id", title:'ID', width:80},
6.             {field:"text", title:'Text', width:100}
7.         ]],
8.         singleSelect: true,
9.         autoRowHeight: false,
10.        width: 200,
11.        height: 200,
12.        url: MyTest.getDataGridData
13.    });
14. });
```

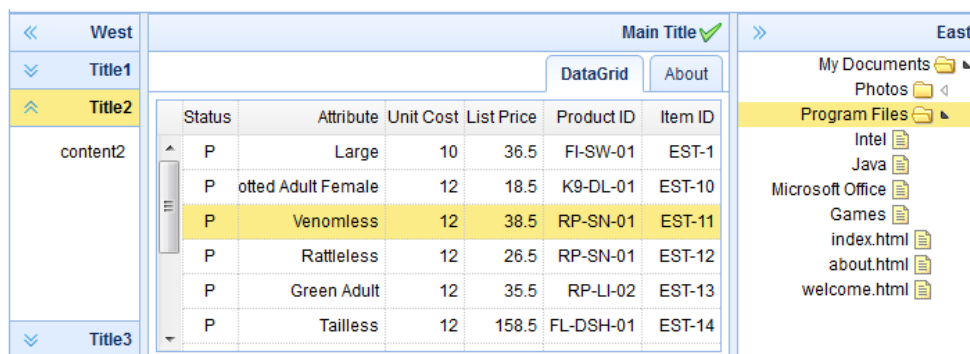
Java 测试代码

```
1. public class Test {
2.     public List<Map<String, Object>> getDataGridData() {
3.         List<Map<String, Object>> items = new ArrayList<Map<String, Object>>();
4.         Map<String, Object> item = new HashMap<String, Object>();
5.         item.put("id", 1);
6.         item.put("text", "text1");
7.         items.add(item);
8.         item = new HashMap<String, Object>();
9.         item.put("id", 2);
10.        item.put("text", "text2");
11.        items.add(item);
12.        return items;
13.    }
14. }
```

RTL suport for jQuery EasyUI (jQuery EasyUI RTL 支持)

[Extension](#) > RTL suport for jQuery EasyUI

扩展下载地址: <http://www.jeasyui.com/extension/downloads/jquery-easyui-rtl.zip> (我发布的程序包整也有提供, 在 extension 目录下)



导入 RTL 文件

启用 RTL 功能, 需要导入 'easyui-rtl.css' 和 'easyui-rtl.js' 文件。

1. `<link rel="stylesheet" type="text/css" href="../../themes/default/easyui.css">`
2. `<link rel="stylesheet" type="text/css" href="easyui-rtl.css">`
3. `<script type="text/javascript" src="../../jquery-1.8.0.min.js"></script>`
4. `<script type="text/javascript" src="../../jquery.easyui.min.js"></script>`
5. `<script type="text/javascript" src="easyui-rtl.js"></script>`

添加 RTL 属性

记住要往<body>标签添加 'dir' 属性, 其值为 'rtl'。

1. `<body dir="rtl">`
2. `<div class="easyui-accordion" data-options="fit:true,border:false">`
3. `<div title="Title1" style="padding:10px;"> content1</div>`
4. `<div title="Title2" data-options="selected:true" style="padding:10px;">content2</div>`
5. `<div title="Title3" style="padding:10px">`
6. `content3`
7. `</div>`
8. `</div>`
9. `</body>`

Ribbon (Ribbon 界面)

[Extension](#) > Ribbon

扩展下载地址: <http://www.jeasyui.com/extension/downloads/jquery-easyui-ribbon.zip> (我发布的程序包也有提供, 在 extension 目录下)



导入 Ribbon 文件

要创建一个 ribbon 组件, 需要导入 'ribbon.css', 'ribbon-icon.css' 和 'jquery.ribbon.js' 文件。

```
1. <link rel="stylesheet" type="text/css"
   href="http://www.jeasyui.com/easyui/themes/default/easyui.css">
2. <link rel="stylesheet" type="text/css"
   href="http://www.jeasyui.com/easyui/themes/icon.css">
3. <link rel="stylesheet" type="text/css" href="ribbon.css">
4. <link rel="stylesheet" type="text/css" href="ribbon-icon.css">
5. <script type="text/javascript" src="http://www.jeasyui.com/easyui/jquery-
   1.8.0.min.js"></script>
6. <script type="text/javascript"
   src="http://www.jeasyui.com/easyui/jquery.easyui.min.js"></script>
7. <script type="text/javascript" src="jquery.ribbon.js"></script>
```

创建 Ribbon

通过标签创建

```
1. <div class="easyui-ribbon" style="width:700px;">
2. <div title="Home">
3. <div class="ribbon-group">
4. <div class="ribbon-toolbar">
5. <a href="#" class="easyui-menubutton" data-options="name:'paste', iconCls:'icon-
   paste-large', iconAlign:'top', size:'large'">Paste</a>
```

```

6. </div>
7. <div class="ribbon-toolbar">
8. <a href="#" class="easyui-linkbutton" data-options="name:'cut',iconCls:'icon-
   cut',plain:true">Cut</a><br>
9. <a href="#" class="easyui-linkbutton" data-options="name:'copy',iconCls:'icon-
   copy',plain:true">Copy</a><br>
10. <a href="#" class="easyui-linkbutton" data-options="name:'format',iconCls:'icon-
    format',plain:true">Format</a>
11. </div>
12. <div class="ribbon-group-title">Clipboard</div>
13. </div>
14. <div class="ribbon-group-sep"></div>
15. <div class="ribbon-group">
16. <div class="ribbon-toolbar" style="width:200px"></div>
17. <div class="ribbon-group-title">other title</div>
18. </div>
19. <div class="ribbon-group-sep"></div>
20. </div>
21. </div>

```

通过 javascript 创建。

```

1. <div id="rr" style="width:700px;"></div>
2. <script>
3. $(function() {
4.   $('#rr').ribbon({
5.     data:data
6.   });
7. });
8. </script>

```

属性

属性扩展自 tabs 控件，下列是 ribbon 新增属性。

属性名	类型	描述	默认值
data	object	Ribbon 描述对象。	undefined

事件

事件扩展自 tabs 控件，下列是 ribbon 新增事件。

事件名	参数	描述
-----	----	----

onClick	name, target	在点击按钮的时候触发。参数包括： name：按钮名称。 target：点击的 DOM 元素。
----------------	--------------	--

方法

方法扩展自 tabs 控件，下列是 ribbon 新增方法。

方法名	方法参数	描述
loadData	data	读取 Ribbon 数据。