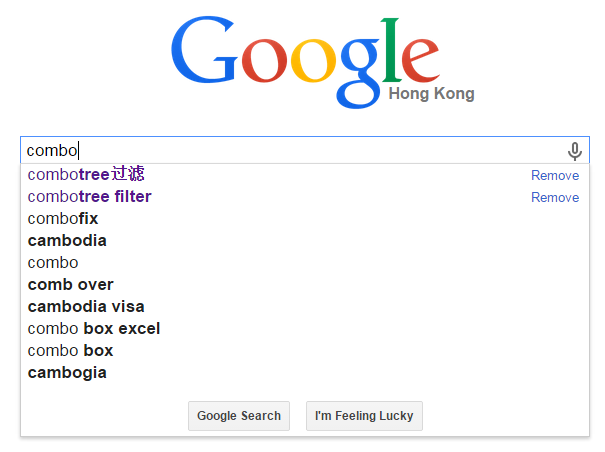
[**【EasyUI】combotree和combobox模糊查询**](http://www.cnblogs.com/ckf1988/p/5715501.html)

这里说的模糊查询指在输入框输入，然后自动在下拉框中显示匹配结果，类似Google搜索提示



EasyUI库已经实现了combobox的查询过滤功能，但只能从头匹配，原因是EasyUI库的代码限制：

1. filter: **function(q, row){**
2. **var opts = $(this).combobox('options');**
3. **return row[opts.textField].indexOf(q) == 0;**
4. }

combobox有一个filter属性，通过这个属性来实现查询效果，在EasyUI库或本地combobox控件中修改这个filter方法就可以实现自定义查询效果

1. $('#businessCityNo').combobox({
2. valueField : 'businessCityNo',
3. textField : 'businessCityName',
4. editable:**true ,**
5. required: **true,**
6. filter: **function(q, row){**
7. **var opts = $(this).combobox('options');**
8. **return row[opts.textField].indexOf(q) >= 0;//这里改成>=即可在任意地方匹配**
9. },
10. data : d.rows,
11. });

当然，直接在生成combobox的地方来添加filter，有多少个就得添加多少次，很麻烦，简单一点，在本地js文件中覆盖这个filter：

1. $.fn.combobox.defaults.filter = **function(q, row){**
2. **var opts = $(this).combobox('options');**
3. **return row[opts.textField].indexOf(q) >= 0;**
4. }

效果如下：



combobox可以通过重写filter方法来实现自定义匹配方式，是因为EasyUI库有对filter属性的底层支持，EasyUI的官方文档中明确提到combobox的属性列表，其中就有filter：  
The properties extend from combo, below is the added properties for combobox.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Default** |
| valueField | string | The underlying data value name to bind to this ComboBox. | value |
| textField | string | The underlying data field name to bind to this ComboBox. | text |
| groupField | string | Indicate what field to be grouped. Available since version 1.3.4. | null |
| groupFormatter | function(group) | return group text to display on group item. Available since version 1.3.4.  Code example:  $('#cc').combobox({  groupFormatter: function(group){  return '<span style="color:red">' + group + '</span>';  }  }); |  |
| mode | string | Defines how to load list data when text changed. Set to 'remote' if the combobox loads from server. When set to 'remote' mode, what the user types will be sent as the http request parameter named 'q' to server to retrieve the new data. | local |
| url | string | A URL to load list data from remote. | null |
| method | string | The http method to retrieve data. | post |
| data | array | The list data to be loaded.  Code example:  <input class="easyui-combobox" data-options="  valueField: 'label',  textField: 'value',  data: [{  label: 'java',  value: 'Java'  },{  label: 'perl',  value: 'Perl'  },{  label: 'ruby',  value: 'Ruby'  }]" /> | null |
| filter | function | Defines how to filter the local data when 'mode' is set to 'local'. The function takes two parameters: q: the user typed text. row: the list row data. Return true to allow the row to be displayed.  Code example:  $('#cc').combobox({  filter: function(q, row){  var opts = $(this).combobox('options');  return row[opts.textField].indexOf(q) == 0;  }  }); |  |
| formatter | function | Defineds how to render the row. The function takes one parameter: row.  Code example:  $('#cc').combobox({  formatter: function(row){  var opts = $(this).combobox('options');  return row[opts.textField];  }  }); |  |
| loader | function(param,success,error) | Defines how to load data from remote server. Return false can abort this action. This function takes following parameters: param: the parameter object to pass to remote server. success(data): the callback function that will be called when retrieve data successfully. error(): the callback function that will be called when failed to retrieve data. | json loader |
| loadFilter | function(data) | Return the filtered data to display. Available since version 1.3.3. |  |

除此之外，还有匹配中文或英文的问题，这里不讨论，如果遇到请Google  
但是，EasyUI并没有为combotree提供filter属性，也就是说没有重写filter方法的根据，官方文档提到combotree就一个属性：  
The properties extend from combo and tree, below is the overridden properties for combotree.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Default** |
| editable | boolean | Defines if user can type text directly into the field. | false |

那么，要使combotree实现查询功能，只能通过扩展代码，在EasyUI库或者本地js文件中加入以下代码：

1. (**function(){**
2. $.fn.combotree.defaults.editable = **true;**
3. $.extend($.fn.combotree.defaults.keyHandler,{
4. up:**function(){**
5. console.log('up');
6. },
7. down:**function(){**
8. console.log('down');
9. },
10. enter:**function(){**
11. console.log('enter');
12. },
13. query:**function(q){**
14. **var t = $(this).combotree('tree');**
15. **var nodes = t.tree('getChildren');**
16. **for(var i=0; i<nodes.length; i++){**
17. **var node = nodes[i];**
18. **if (node.text.indexOf(q) >= 0){**
19. $(node.target).show();
20. } **else {**
21. $(node.target).hide();
22. }
23. }
24. **var opts = $(this).combotree('options');**
25. **if (!opts.hasSetEvents){**
26. opts.hasSetEvents = **true;**
27. **var onShowPanel = opts.onShowPanel;**
28. opts.onShowPanel = **function(){**
29. **var nodes = t.tree('getChildren');**
30. **for(var i=0; i<nodes.length; i++){**
31. $(nodes[i].target).show();
32. }
33. onShowPanel.call(**this);**
34. };
35. $(**this).combo('options').onShowPanel = opts.onShowPanel;**
36. }
37. }
38. });
39. })(jQuery);

如果在公共EasyUI库中加入以上代码，就可以在本地重写query方法来实现和combobox一样的自定义查询效果；如果是本地可以直接更改query实现自定义。  
效果如下：



总结：让EasyUI的combobox和combotree同时支持自定义模糊查询，在不更改其他代码的情况下，添加以下代码就行了：

* 1. /\*\*
  2. \* combobox和combotree模糊查询
  3. \*/
  4. (**function(){**
  5. //combobox可编辑，自定义模糊查询
  6. $.fn.combobox.defaults.editable = **true;**
  7. $.fn.combobox.defaults.filter = **function(q, row){**
  8. **var opts = $(this).combobox('options');**
  9. **return row[opts.textField].indexOf(q) >= 0;**
  10. };
  11. //combotree可编辑，自定义模糊查询
  12. $.fn.combotree.defaults.editable = **true;**
  13. $.extend($.fn.combotree.defaults.keyHandler,{
  14. up:**function(){**
  15. console.log('up');
  16. },
  17. down:**function(){**
  18. console.log('down');
  19. },
  20. enter:**function(){**
  21. console.log('enter');
  22. },
  23. query:**function(q){**
  24. **var t = $(this).combotree('tree');**
  25. **var nodes = t.tree('getChildren');**
  26. **for(var i=0; i<nodes.length; i++){**
  27. **var node = nodes[i];**
  28. **if (node.text.indexOf(q) >= 0){**
  29. $(node.target).show();
  30. } **else {**
  31. $(node.target).hide();
  32. }
  33. }
  34. **var opts = $(this).combotree('options');**
  35. **if (!opts.hasSetEvents){**
  36. opts.hasSetEvents = **true;**
  37. **var onShowPanel = opts.onShowPanel;**
  38. opts.onShowPanel = **function(){**
  39. **var nodes = t.tree('getChildren');**
  40. **for(var i=0; i<nodes.length; i++){**
  41. $(nodes[i].target).show();
  42. }
  43. onShowPanel.call(**this);**
  44. };
  45. $(**this).combo('options').onShowPanel = opts.onShowPanel;**
  46. }
  47. }
  48. });
  49. })(jQuery);

标签: [css](http://www.cnblogs.com/ckf1988/tag/css/)