https://blog.csdn.net/liushuiziyouliu/article/details/68059860

**easyui combotree 模糊搜索**

2017年03月29日 09:46:54 [liu---shui](https://me.csdn.net/liushuiziyouliu" \t "_blank) 阅读数 1560

只需把此js引入到项目中即可

(function ($) {  
    //combotree可编辑，自定义模糊查询    
    $.fn.combotree.defaults.editable = true;  
    $.extend($.fn.combotree.defaults.keyHandler, {  
        query: function (q) {  
            var t = $(this).combotree('tree');  
            t.tree("search", q);  
        }  
    });  
    $.extend($.fn.tree.methods, {           /\*\*     
         \* 扩展easyui tree的搜索方法   
         \* @param tree easyui tree的根DOM节点(UL节点)的jQuery对象          \* @param searchText 检索的文本   
         \* @param this-context easyui tree的tree对象          \*/  
        search: function (jqTree, searchText) {  
            //easyui tree的tree对象。可以通过tree.methodName(jqTree)方式调用easyui tree的方法   
            var tree = this;  
            //获取所有的树节点   
            var nodeList = getAllNodes(jqTree, tree);  
            //如果没有搜索条件，则展示所有树节点         
            searchText = $.trim(searchText);  
            if (searchText == "") {  
                for (var i = 0; i < nodeList.length; i++) {  
                    $(".tree-node-targeted",  
                nodeList[i].target).removeClass("tree-node-targeted");  
                    $(nodeList[i].target).show();  
                }  
                //展开已选择的节点（如果之前选择了）              
                var selectedNode = tree.getSelected(jqTree);  
                if (selectedNode) {  
                    tree.expandTo(jqTree, selectedNode.target);  
                }  
                return;  
            }  
            //搜索匹配的节点并高亮显示              
            var matchedNodeList = [];  
            if (nodeList && nodeList.length > 0) {  
                var node = null;  
                for (var i = 0; i < nodeList.length; i++) {  
                    node = nodeList[i];  
                    if (isMatch(searchText, node.text)) {  
                        matchedNodeList.push(node);  
                    }  
                }  
                //隐藏所有节点   
                for (var i = 0; i < nodeList.length; i++) {  
                    $(".tree-node-targeted", nodeList[i].target).removeClass("tree-node-targeted");  
                    $(nodeList[i].target).hide();  
                }  
                //折叠所有节点   
                tree.collapseAll(jqTree);  
                //展示所有匹配的节点以及父节点   
  
  
                for (var i = 0; i < matchedNodeList.length; i++) {  
                    showMatchedNode(jqTree, tree, matchedNodeList[i]);  
                }  
            }  
        },  
        /\*\*     
        \* 展示节点的子节点（子节点有可能在搜索的过程中被隐藏了）          
        \* @param node easyui tree节点           
        \*/  
        showChildren: function (jqTree, node) {  
            //easyui tree的tree对象。可以通过tree.methodName(jqTree)方式调用easyui tree的方法   
            var tree = this;  
            //展示子节点   
            if (!tree.isLeaf(jqTree, node.target)) {  
                var children = tree.getChildren(jqTree, node.target);  
                if (children && children.length > 0) {  
                    for (var i = 0; i < children.length; i++) {  
                        if ($(children[i].target).is(":hidden")) {  
                            $(children[i].target).show();  
                        }  
                    }  
                }  
            }  
        },  
        /\*\*     
        \* 将滚动条滚动到指定的节点位置，使该节点可见（如果有滚动条才滚动，没有滚动条就不滚动）   
        \* @param param {     
        \* treeContainer: easyui tree的容器（即存在滚动条的树容器）。如果为null，则取easyui tree的根UL节点的父节点。   
        \* argetNode:  将要滚动到的easyui tree节点。如果targetNode为空，则默认滚动到当前已选中的节点，如果没有选中的节点，则不滚动          \* }             \*/  
        scrollTo: function (jqTree, param) {  
            //easyui tree的tree对象。可以通过tree.methodName(jqTree)方式调用easyui tree的方法   
            var tree = this;  
            //如果node为空，则获取当前选中的node     
            var targetNode = param && param.targetNode ? param.targetNode : tree.getSelected(jqTree);  
            if (targetNode != null) {  
                //判断节点是否在可视区域                 var root = tree.getRoot(jqTree);     
                var $targetNode = $(targetNode.target);  
                var container = param && param.treeContainer ? param.treeContainer : jqTree.parent();  
                var containerH = container.height();  
                var nodeOffsetHeight = $targetNode.offset().top - container.offset().top;  
                if (nodeOffsetHeight > (containerH - 30)) {  
                    var scrollHeight = container.scrollTop() + nodeOffsetHeight - containerH + 30;  
                    container.scrollTop(scrollHeight);  
                }  
            }  
        }  
    });  
    /\*\*     
         \* 展示搜索匹配的节点      \*/  
    function showMatchedNode(jqTree, tree, node) {  
        //展示所有父节点   
        $(node.target).show();  
        $(".tree-title", node.target).addClass("tree-node-targeted");  
        var pNode = node;  
        while ((pNode = tree.getParent(jqTree, pNode.target))) {  
            $(pNode.target).show();  
        }  
        //展开到该节点   
        tree.expandTo(jqTree, node.target);  
        //如果是非叶子节点，需折叠该节点的所有子节点        
        if (!tree.isLeaf(jqTree, node.target)) {  
            tree.collapse(jqTree, node.target);  
        }  
    }  
    /\*\*     
     \* 判断searchText是否与targetText匹配   
     \* @param searchText 检索的文本      \* @param targetText 目标文本   
     \* @return true-检索的文本与目标文本匹配；否则为false.          
    \*/  
    function isMatch(searchText, targetText) {  
        return $.trim(targetText) != "" && targetText.indexOf(searchText) != -1;  
    }  
    /\*\*     
     \* 获取easyui tree的所有node节点      \*/  
    function getAllNodes(jqTree, tree) {  
        var allNodeList = jqTree.data("allNodeList");  
        if (!allNodeList) {  
            var roots = tree.getRoots(jqTree);  
            allNodeList = getChildNodeList(jqTree, tree, roots);  
            jqTree.data("allNodeList", allNodeList);  
        }  
        return allNodeList;  
    }  
    /\*\*     
     \* 定义获取easyui tree的子节点的递归算法      \*/  
    function getChildNodeList(jqTree, tree, nodes) {  
        var childNodeList = [];  
        if (nodes && nodes.length > 0) {  
            var node = null;  
            for (var i = 0; i < nodes.length; i++) {  
                node = nodes[i];  
                childNodeList.push(node);  
                if (!tree.isLeaf(jqTree, node.target)) {  
                    var children = tree.getChildren(jqTree, node.target);  
                    childNodeList = childNodeList.concat(getChildNodeList(jqTree, tree, children));  
                }  
            }  
        }  
        return childNodeList;  
    }  
})(jQuery);

**[Easyui实现combotree,输入文本及时筛选功能](https://blog.csdn.net/qq_35624642/article/details/53931111" \o "Easyui实现combotree,输入文本及时筛选功能" \t "_blank)**

[阅读数 5823](https://blog.csdn.net/qq_35624642/article/details/53931111" \o "Easyui实现combotree,输入文本及时筛选功能" \t "_blank)

[easyui提供的ComboTree组件没有查找功能，如果结点多的话很不好找,想要的功能是在ComboTree中输入文字就显示出相应关键字的结点，有点像自动补全。本来的路是用ComboTreer的on...](https://blog.csdn.net/qq_35624642/article/details/53931111)博文[来自： QC班长的博客](https://blog.csdn.net/qq_35624642)