

关于	参数说明	事件
zTree 介绍	zTreeNodes 详解	callback 回调函数
提交bug及获取更新	checked click icon iconSkin isParent name nodes open target url *自定义*	* beforeClick * beforeRightClick * beforeMouseDown * beforeMouseUp * beforeChange * beforeDrag * beforeDrop * beforeRename * beforeRemove * beforeExpand * beforeCollapse * beforeAsync * nodeCreated # click * rightClick * mouseDown * mouseUp # change # drag # drop * rename * remove * expand * collapse # asyncSuccess # asyncError
核心		
zTree 核心函数		
zTree(setting, [zTreeNodes])		
skin 皮肤说明		
zTreeStyle.css		
zTreeIcons.css		
参数说明		
setting 详解		
async asyncUrl asyncParam # asyncParamOther * isSimpleData * rootPID * treeNodeKey * treeNodeParentKey checkable checkType # checkStyle checkRadioType editable * keepParent * keepLeaf * edit_renameBtn * edit_removeBtn root showLine * showIcon * fontCss * nameCol * nodesCol * checkedCol * expandSpeed * addHoverDom * removeHoverDom * addDiyDom * callback [curTreeNode] * [curEditTreeNode] [dragNodeShowBefore] [dragStatus] * [expandTriggerFlag] [treeObjId] [checkRadioCheckedList]	* [checkedOld] * [checkboxFocus] * [check_True_Full] * [check_False_Full] * [editNameStatus] * [isAjaxing] * [isHover] [isFirstNode] [isLastNode] [level] [parentNode] [tId]	
方法		
获取	# getSelectedNode() getNodeByTId(tID) * getNodeIndex(treeNode) getNodes() # getCheckedNodes(checked) * getChangeCheckedNodes() * getSetting() * getNodeByParam(key, value) * getNodesByParam(key, value, parentNode) * getNodesByParamFuzzy(key, value, parentNode) * transformTozTreeNodes(simpleTreeNodes) * transformToArray(treeNodes)	
操作	addNodes(parentNode, newNodes, isSilent) expandAll(expandSign) # expandNode(treeNode, expandSign, sonSign) # moveNode(targetNode, treeNode, moveType) refresh() removeNode(treeNode) # selectNode(treeNode) * checkAllNodes(checked) setEditable(editable) * cancelSelectedNode() * updateNode(treeNode, checkTypeFlag) * updateSetting(setting) * reAsyncChildNodes(parentNode, reloadType)	
常量		
事件相关		* ZTREE_NODECREATED ZTREE_CLICK # ZTREE_CHANGE * ZTREE_RENAME * ZTREE_REMOVE ZTREE_DRAG ZTREE_DROP ZTREE_ASYNC_SUCCESS ZTREE_ASYNC_ERROR
ID命名相关		IDMark_Switch IDMark_Icon * IDMark_Check * IDMark_Edit * IDMark_Remove IDMark_A * IDMark_Span * IDMark_Input IDMark_UI

常量

背景线条相关

LineMark_Root
LineMark_Roots
LineMark_Center
LineMark_Bottom
LineMark_NoLine
LineMark_Line

文件夹图标相关

FolderMark_Open
FolderMark_Close
FolderMark_Docu

className 相关

Class_CurSelectedNode
* Class_CurSelectedNode_Edit
Class_TmpTargetTree
Class_TmpTargetNode

CheckBox & Radio 相关

Check_Style_Box
Check_Style_Radio
CheckBox_Default
CheckBox_False
CheckBox_True
CheckBox_Full
CheckBox_Part
CheckBox.Focus
Radio_Type_All
Radio_Type_Level

其他

* MinMoveSize
* MoveType_Inner
* MoveType_Before
* MoveType_After

概述

最近对 **JQuery** 进行了入门，一时兴起写了一个**Tree**插件，供大家学习和使用，毕竟是本人第一个公开的组件，肯定有许多问题和不足之处，请大家把发现的问题，或好的想法及时与我沟通，在这里特别要感谢[独上太行](#) 的大力支持，架构方面给了我很多关键性的建议。



同时欢迎利用此版制作其他发行版以方便广大 **zTree** 爱好者，转载请保留版权信息，谢谢。

下面介绍一下**zTree** 的主要功能：(演示[Demo](#) 请访问 [个人站点](#))

- 1、兼容 **IE**、**FireFox**、**Chrome** 等浏览器
- 2、在一个页面内可同时生成多个 **Tree** 实例
- 3、支持 **JSON** 数据
- 4、支持一次性静态生成 和 **Ajax** 异步加载 两种方式
- 5、支持多种事件响应及反馈
- 6、支持 **Tree** 的节点移动、编辑、删除
- 7、支持极其灵活的 **checkbox & radio** 选择功能
- 8、支持任意更换皮肤 / 个性化图标 (依靠**css**)
- 9、简单的参数配置 实现 灵活多变的功能

本手册由 [Hunter.z](#) 整理编辑，并保持长期更新，最新版本请从 [zTree官网](#) 或 [个人站点](#) 获取

zTree v2.5 更新记录

概述

- 01、【修订】允许异步加载与编辑状态共存，无需特殊代码
- 02、【修改】拖拽操作让叶子节点变为父节点后，展开/折叠按钮事件异常的 Bug
- 03、【修改】关于 [[treeNodeParentKey](#)] 属性值在拖拽后没有被修正的Bug（增加 [rootPID](#)）
- 04、【修改】简单数据模式中，如果 [[treeNodeKey](#)] 属性值为 0 的节点无法成为父节点的 Bug
- 05、【增加】锁定父节点的参数 [keepParent](#)，默认false
- 06、【增加】锁定叶子节点的参数 [keepLeaf](#)，默认false
- 07、【增加】[beforeAsync](#) 的回调函数，由用户控制是否允许进行异步加载
- 08、【增加】[getNodesByParamFuzzy\(key, value, parentNode\)](#) 方法支持模糊查找（只能针对String的属性）
- 09、【增加】提供一个不显示图标的参数设置 [showIcon](#)

概述

如果大家使用过程中发现了什么Bug，或者有不同的想法、建议都可以到项目地址：

<http://code.google.com/p/jquerytree/issues/list> 或 <http://hi.baidu.com/ztreeapi/home> 或直接发Email来反馈。

[查看最新版本](#)

想查看最新版本，请从项目地址：

<http://code.google.com/p/jquerytree/downloads/list> 进行下载。

概述

目录: [zTree 核心函数]

这个函数接受一个 JSON 格式的数据对象 `setting` 和一个 JSON 格式的数据对象 `zTreeNodes`, 从而建立 Tree。

对于用户在 Web 页面上建立 Tree, 就是通过这个函数实现的, 对于后期的代码控制, 则通过返回的 `zTreePlugin` 对象操作即可。

需要显示 Tree 的 Web 页面需要加载 `jquery-1.4.2.js / jquery-ztree-2.5.js / zTreeStyle.css` 这三个文件。

请注意对于 Tree 的容器增加 `class="tree"`。

需要使用系列图标请加载 `zTreeIcons.css`。

页面需要进行 W3C 申明, 例如: `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`

参数

setting Json

zTree 的参数配置数据, 详细请参见 [参数说明 -> setting 详解](#)

zTreeNodes (可选) JSON

zTree 的参数配置数据, 详细请参见 [参数说明 -> zTreeNodes 详解](#);

如果将节点数据直接放在 `setting.root.nodes` 下, 或者 全部从异步获取节点数据, 则不需要传递此参数。

示例

描述:

简单创建 zTree 演示

`setting` 举例:

```
var setting = {  
    showLine: true,  
    checkable: true  
};
```

普通 `zTreeNodes` 举例:

```
var zTreeNodes = [  
    { "name":"google", "url":"http://g.cn", "target":"_blank"},  
    { "name":"baidu", "url":"http://baidu.com", "target":"_blank"},  
    { "name":"sina", "url":"http://www.sina.com.cn", "target":"_blank"}  
];
```

带有父子关系的标准 `zTreeNodes` 举例:

```
var zTreeNodes = [  
    {"id":1, "name":"test1", "nodes": [  
        {"id":11, "name":"test11", "nodes": [  
            {"id":111, "name":"test111"}  
        ]},  
        {"id":12, "name":"test12"}  
    ]},  
    .....  
];
```

带有父子关系的简单 Array 格式的 `zTreeNodes` 举例 (使用简单 Array 格式的数据请参考 [isSimpleData](#)) :

```
var treeNodes = [
```

```
{"id":1, "pId":0, "name":"test1"},  
 {"id":11, "pId":1, "name":"test11"},  
 {"id":12, "pId":1, "name":"test12"},  
 {"id":111, "pId":11, "name":"test111"},  
.....  
];
```

Html 对象:

```
<ul id="tree" class="tree" style="width:300px; overflow:auto;"></ul>
```

初始化 Tree:

```
var zTree = $("#tree").zTree(setting, zTreeNodes);
```

zTree的样式设置, 如果想自定义皮肤, 请根据以下简要说明, 设计个性化的图标以及css样式

请注意对于 Tree 的容器增加 `class="tree"`。

简要说明

`.tree{...}`

zTree内部li的总体样式设定 (zTree采用ul li的结构显示Tree)

`.tree li{...}`

zTree内部li内的ul的总体样式设定

`.tree li ul{...}`

zTree内部节点间连线样式设定

`.tree li ul.line{...}`

zTree内节点超级链接的样式设定

`.tree li a{...}`
`.tree li a:hover{...}`

zTree内节点被选择时的样式设定

`.tree li a.curSelectedNode{...}`

zTree内节点被选择后, 处于编辑状态时的样式设定

`.tree li a.curSelectedNode_Edit{...}`

zTree内节点成为正被拖拽目标的父节点时的样式设定

`.tree li a.tmpTargetNode{...}`

zTree内节点编辑name时输入框的样式设定

`.tree li a input.rename{...}`

zTree内部展开折叠图标以及节点个性化图标的总体样式设定

`.tree li button{...}`

zTree内部个性化图标的总体样式设定

`.tree li button.ico{...}`

zTree内部编辑按钮样式设定

`.tree li button.edit{...}`

zTree 内部删除按钮样式设定

```
.tree li button.del{...}
```

zTree 内部 checkbox & radio 图标样式设定

```
.tree li button.chk{...}  
.tree li button.chk.checkbox_false_full{...}  
.tree li button.chk.checkbox_false_full_focus{...}  
.tree li button.chk.checkbox_false_part{...}  
.tree li button.chk.checkbox_false_part_focus{...}  
.tree li button.chk.checkbox_true_full{...}  
.tree li button.chk.checkbox_true_full_focus{...}  
.tree li button.chk.checkbox_true_part{...}  
.tree li button.chk.checkbox_true_part_focus{...}  
  
.tree li button.chk.radio_false_full{...}  
.tree li button.chk.radio_false_full_focus{...}  
.tree li button.chk.radio_false_part{...}  
.tree li button.chk.radio_false_part_focus{...}  
.tree li button.chk.radio_true_full{...}  
.tree li button.chk.radio_true_full_focus{...}  
.tree li button.chk.radio_true_part{...}  
.tree li button.chk.radio_true_part_focus{...}
```

zTree 内部父节点展开折叠图标样式设定

```
.tree li button.switch_root_open{...}  
.tree li button.switch_root_close{...}  
.tree li button.switch_roots_open{...}  
.tree li button.switch_roots_close{...}  
.tree li button.switch_center_open{...}  
.tree li button.switch_center_close{...}  
.tree li button.switch_bottom_open{...}  
.tree li button.switch_bottom_close{...}  
.tree li button.switch_noLine_open{...}  
.tree li button.switch_noLine_close{...}
```

zTree 内部叶子节点连线图标样式设定

```
.tree li button.switch_root_docu{...}  
.tree li button.switch_roots_docu{...}  
.tree li button.switch_center_docu{...}  
.tree li button.switch_bottom_docu{...}  
.tree li button.switch_noLine_docu{...}
```

zTree 内部个性化节点图标样式设定

```
.tree li button.ico_loading{...} //loading 图标, v2.2 增加  
.tree li button.ico_open{...}  
.tree li button.ico_close{...}  
.tree li button.ico_docu{...}
```

zTree 内部 CheckBox 输入框样式设定

```
.tree INPUT.checkbox{...}
```

zTree 的根成为拖拽的目的地时样式设定

```
.tmpTargetTree{...}
```

zTree节点拖拽时指示目标的箭头图标样式设定

```
button.tmpzTreeMove_arrow{...} //拖拽移动节点时的位置图标, v2.2 增加
```

zTree的节点拖拽图层样式设定

```
.zTreeDragUL{...}
```

zTree的iframe遮罩样式设定 (iFrame遮罩能有效避免页面上的iframe导致拖拽停滞的影响)

```
.zTreeMask{...}
```

zTree的样式个性化系列图标设置，因为这部分弹性很大，可能会很多，故专门提取出来。

简要说明

zTree内部个性化节点图标系列样式设定 (从 v1.02 版本开始支持此功能)

```
.tree li button.sim1.ico_open{...}  
.tree li button.sim1.ico_close{...}  
.tree li button.sim2.ico_docu{...}  
.tree li button.sim3.ico_docu{...}  
.....
```

概述

目录: [参数说明]

setting 是 zTree 的全部设置参数集合, 采用 JSON 结构, 便于灵活配置。

属性

[async](#) | [asyncUrl](#) | [asyncParam](#) | [asyncParamOther](#)
[isSimpleData](#) | [rootPID](#) | [treeNodeKey](#) | [treeNodeParentKey](#)
[checkable](#) | [checkType](#) | [checkStyle](#) | [checkRadioType](#)
[editable](#) | [keepParent](#) | [keepLeaf](#) | [edit_renameBtn](#) | [edit_removeBtn](#)
[root](#) | [showLine](#) | [showIcon](#) | [fontCss](#)
[nameCol](#) | [nodesCol](#) | [checkedCol](#) | [expandSpeed](#)
[addHoverDom](#) | [removeHoverDom](#) | [addDiyDom](#) | [callback](#)

[[curTreeNode](#)] | [[curEditTreeNode](#)]
[[dragNodeShowBefore](#)] | [[dragStatus](#)] | [[expandTriggerFlag](#)]
[[treeObjId](#)] | [[checkRadioCheckedList](#)]

async

Boolean

概述

目录: [参数说明] --> [setting 详解]

确定 zTree 是否通过异步方式获取 isParent = true, 且没有子节点数据的父节点的子节点数据

默认值: false

示例

描述:

需要采用异步方式获取子节点数据

setting 举例:

```
var setting = {  
    async : true,  
    .....  
};
```

相关参数

[asyncUrl](#) | [asyncParam](#) | [asyncParamOther](#)

相关事件

[asyncSuccess\(event, treeId, treeNode, msg\)](#) | [asyncError\(event, treeId, treeNode, XMLHttpRequest, textStatus, errorThrown\)](#)

asyncUrl

String / Function

概述

目录: [参数说明] --> [setting 详解]

当 `async = true` 时, 设置异步获取节点的 URL 地址

默认值: ""

v2.2版扩展此参数功能, 允许接收 `function` 的引用, 以便于用户根据节点动态变换异步加载的url。

`function` 格式举例: `function getAsyncUrl(treeNode) { ... }` 注意: 对于root根节点异步加载时, `treeNode = null`

示例

描述:

设置异步获取节点的 URL 为 `nodes.php`

`setting` 举例:

```
var setting = {
    asyncUrl : "nodes.php",
    .....
};
```

描述:

设置异步获取节点的 URL 为 `function` 动态获取

`setting` 举例:

```
function getAsyncUrl(treeNode) {
    var url = "";
    .....
    return url;
};
var setting = {
    asyncUrl : getAsyncUrl,
    .....
};
```

相关方法

[reAsyncChildNodes\(parentNode, reloadType\)](#)

相关参数

[async](#) | [asyncParam](#) | [asyncParamOther](#)

概述

目录: [参数说明] --> [setting 详解]

当 `async = true` , 且访问 `asyncUrl` 指定的地址时, 提交的与节点数据相关的必需属性。 例如: `name`

默认值: []

示例

描述:

设置异步获取数据时, 必需传递父节点数据"name"属性的值
setting 举例:

```
var setting = {  
    asyncParam : ["name"],  
    .....  
};
```

相关参数

[async](#) | [asyncUrl](#) | [asyncParamOther](#)

概述

目录: [参数说明] --> [setting 详解]

当 `async = true` , 且访问 `asyncUrl` 时, 提交的固定键值对。

默认值: []

可以为空[], 如果有 `key`, 则必须存在 `value`。 例如: `[key, value]`

从 v2.5 版本开始可以支持 JSON 对象, 例如: `{key1:value1, key2:value2}`

示例

描述:

设置异步获取数据时, 传递 Array 格式的 `key, value`
setting 举例:

```
var setting = {  
    asyncParamOther : ["key", "value"],  
    ....  
};
```

设置异步获取数据时, 传递 JSON 格式的 `key, value`
setting 举例:

```
var setting = {  
    asyncParamOther : {"key":"value"},  
    ....  
};
```

相关参数

[async](#) | [asyncUrl](#) | [asyncParam](#)

概述

目录: [参数说明] --> [setting 详解]

确定 zTree 初始化时的节点数据、异步加载时的节点数据、或 [addNodes\(parentNode, newNodes, isSilent\)](#) 方法中输入的 newNodes 数据是否采用简单 Array 格式

不需要用户再把数据库中取出的 List 强行转换为复杂的 JSON 嵌套格式

如果设置为 true, 请务必设置节点唯一标识属性名称 [treeNodeKey](#) 和 父节点唯一标识属性名称 [treeNodeParentKey](#), 并且让数据满足父子关系。

默认值: false

示例

描述:

使用简单 Array 格式的数据

setting 举例:

```
var setting = {  
    isSimpleData : true,  
    treeNodeKey : "id",  
    treeNodeParentKey : "pId",  
    .....  
};
```

简单 Array 数据 举例:

```
var treeNodes = [  
    {"id":1, "pId":0, "name":"test1"},  
    {"id":11, "pId":1, "name":"test11"},  
    {"id":12, "pId":1, "name":"test12"},  
    {"id":111, "pId":11, "name":"test111"},  
    .....  
];
```

相关参数

[treeNodeKey](#) | [treeNodeParentKey](#)

相关方法

[transformTozTreeNodes\(simpleTreeNodes\)](#)

rootPID

任意类型

概述

目录: [参数说明] --> [setting 详解]

当 [isSimpleData](#) 设置为 true, zTree 需要在增加、移动节点时修正其 [treeNodeParentKey](#) 对应属性值。

rootPID 用于通知 zTree, 对于根节点 [treeNodeParentKey](#) 对应属性值

rootPID 可以是任何类型, 请尽可能与 [treeNodeParentKey](#) 对应属性值一致

默认值: null

示例

描述:

使用简单 Array 格式的数据

setting 举例:

```
var setting = {  
    isSimpleData : true,  
    rootPID : -1,  
    treeNodeKey : "id",  
    treeNodeParentKey : "pId",  
    .....  
};
```

简单 Array 数据 举例:

```
var treeNodes = [  
    {"id":1, "pId":0, "name":"test1"},  
    {"id":11, "pId":1, "name":"test11"},  
    {"id":12, "pId":1, "name":"test12"},  
    {"id":111, "pId":11, "name":"test111"},  
    .....  
];
```

相关参数

[isSimpleData](#) | [treeNodeKey](#) | [treeNodeParentKey](#)

相关方法

[transformTozTreeNodes\(simpleTreeNodes\)](#)

treeNodeKey

String

概述

目录: [参数说明] --> [setting 详解]

设置节点唯一标识属性名称, 转换数据格式时使用, 例如: 当 [isSimpleData](#) 设置为 true, 或 调用 [transformTozTreeNodes\(simpleTreeNodes\)](#) 方法。

默认值: ""

示例

描述:

使用简单 Array 格式的数据

setting 举例:

```
var setting = {  
    isSimpleData : true,  
    rootPID : -1,  
    treeNodeKey : "id",  
    treeNodeParentKey : "pId",  
    .....  
};
```

简单 Array 数据 举例:

```
var treeNodes = [  
    {"id":1, "pId":0, "name":"test1"},  
    {"id":11, "pId":1, "name":"test11"},  
    {"id":12, "pId":1, "name":"test12"},  
    {"id":111, "pId":11, "name":"test111"},  
    .....  
];
```

相关参数

[isSimpleData](#) | [rootPID](#) | [treeNodeParentKey](#)

相关方法

[transformTozTreeNodes\(simpleTreeNodes\)](#)

treeNodeParentKey

String

概述

目录: [参数说明] --> [setting 详解]

设置节点的父节点唯一标识属性名称, 转换数据格式时使用, 例如: 当 [isSimpleData](#) 设置为 `true`, 或 调用 [transformTozTreeNodes\(simpleTreeNodes\)](#) 方法。

默认值: ""

示例

描述:

使用简单 `Array` 格式的数据

setting 举例:

```
var setting = {  
    isSimpleData : true,  
    rootPID : -1,  
    treeNodeKey : "id",  
    treeNodeParentKey : "pId",  
    .....  
};
```

简单 `Array` 数据 举例:

```
var treeNodes = [  
    {"id":1, "pId":0, "name":"test1"},  
    {"id":11, "pId":1, "name":"test11"},  
    {"id":12, "pId":1, "name":"test12"},  
    {"id":111, "pId":11, "name":"test111"},  
    .....  
];
```

相关参数

[isSimpleData](#) | [rootPID](#) | [treeNodeKey](#)

相关方法

[transformTozTreeNodes\(simpleTreeNodes\)](#)

checkable

Boolean

概述

目录: [参数说明] --> [setting 详解]

确定 zTree 的节点上是否显示 CheckBox

默认值: **false**

示例

描述:

需要显示 CheckBox

setting 举例:

```
var setting = {  
    checkable : true,  
    .....  
};
```

相关参数

[checkType](#)

相关事件

[change](#)

概述

目录: [参数说明] --> [setting 详解]

当 `checkable = true` 时, `checkType` 配置点击 `CheckBox` 后, 对于父子节点的影响规则。

规则说明: Y 表明 `CheckBox` 被选中后的情况; N 表明 `CheckBox` 取消选中后的情况; p 表示选择会影响父级节点; s 表明会影响子级节点。

请注意大小写, 不要改变

请注意在取消选择的操作时, 如果设置影响父级节点, 只有在父节点下的子节点全部取消选择时才会生效。

默认值: { "Y": "ps", "N": "ps" }

示例

描述:

`CheckBox` 被选中后, 只影响父级节点; 取消选中后, 只影响子级节点

setting 举例:

```
var setting = {
  checkable : true,
  checkType : { "Y": "p", "N": "s" },
  .....
};
```

相关参数

[checkable](#)

概述

目录: [参数说明] --> [setting 详解]

当 `checkable = true` 时, `checkStyle` 配置选择框类型为 `checkbox` 还是 `radio`。

在2.0中, `radio` 只能选择单个节点, 并且自动选中最新节点, 完全满足`radio`标准, 删除最大数量限制功能。

规则说明:

`checkStyle = "checkbox"` 时, 显示为 `checkbox` 选择框, 没有选择数量限制, [checkType](#) 属性有效。

`checkStyle = "radio"` 时, 显示为 `radio` 选择框, [checkRadioType](#) 属性有效。

请注意大小写, 不要改变

默认值: `checkbox`

checkbox 状态说明

- 当前节点没有被选择; 如果是父节点, 则没有被选择的子节点。鼠标移到该节点上显示为:
- 当前节点没有被选择; 存在被选择的子节点 (只有父节点存在此状态)。鼠标移到该节点上显示为:
- 当前节点被选择; 如果是父节点, 则全部子节点都被选择。鼠标移到该节点上显示为:
- 当前节点被选择; 存在没有被选择的子节点 (只有父节点存在此状态)。鼠标移到该节点上显示为:

radio 状态说明

- 当前节点没有被选择; 如果是父节点, 则没有被选择的子节点。鼠标移到该节点上显示为:
- 当前节点没有被选择; 存在被选择的子节点 (只有父节点存在此状态)。鼠标移到该节点上显示为:
- 当前节点被选择; 如果是父节点, 则没有被选择的子节点。鼠标移到该节点上显示为:
- 当前节点被选择; 且有被选择的子节点 (只有父节点存在此状态)。鼠标移到该节点上显示为:

示例

描述:

选择框显示为 `radio`

setting 举例:

```
var setting = {  
    checkable : true,  
    checkStyle : "radio",  
    ....  
};
```

相关参数

[checkable](#) | [checkType](#) | [checkRadioType](#)

checkRadioType

String

概述

目录: [参数说明] --> [setting 详解]

当 checkable = true 且 checkStyle = "radio" 时, checkRadioType 配置 radio 的分组范围。

规则说明:

checkRadioType = "level" 时, 在每一级节点范围内当做一个分组。

checkRadioType = "all" 时, 在整棵树范围内当做一个分组。

请注意大小写, 不要改变

默认值: level

示例

描述:

在整棵树内限制可选择节点的个数

setting 举例:

```
var setting = {  
    checkable : true,  
    checkStyle : "radio",  
    checkRadioType : "all",  
    .....  
};
```

相关参数

[checkable](#) | [checkStyle](#)

editable

Boolean

概述

目录: [参数说明] --> [setting 详解]

确定 zTree 是否处于编辑状态。

editable = true 时:

- 1、将不会对节点指定的 [url](#) 地址进行响应
- 2、从 v2.5 版本开始支持 编辑 与 异步加载 状态共存
- 3、可以对节点进行拖拽 从 v2.3 版本开始支持多棵树之间进行拖拽
- 4、可以通过编辑按钮修改 name 属性
- 5、可以通过删除按钮删除节点

默认值: false

示例

描述:

允许拖拽节点

setting 举例:

```
var setting = {  
    editable : true,  
    .....  
};
```

相关参数

[edit_renameBtn](#) | [edit_removeBtn](#)

相关事件

[beforeDrag\(treeId, treeNode\)](#) | [beforeDrop\(treeId, treeNode, targetNode, moveType\)](#)
[drag\(event, treeId, treeNode\)](#) | [drop\(event, treeId, treeNode, targetNode, moveType\)](#)

keepParent

Boolean

概述

目录: [参数说明] --> [setting 详解]

是否锁定 zTree 的节点父节点属性，是否始终保持 `isParent=true`

对于 zTree 内所有 `isParent=true` 的节点，即使该节点的子节点被全部删除或移走，依旧保持父节点状态。

默认值: `false`

示例

描述:

需要锁定父节点状态

setting 举例:

```
var setting = {  
    keepParent : true,  
    .....  
};
```

相关参数

[keepLeaf](#)

keepLeaf

Boolean

概述

目录: [参数说明] --> [setting 详解]

是否锁定 zTree 的节点叶子节点属性，是否始终保持 isParent=false

对于 zTree 内所有 isParent=false 的节点，则无法给该节点添加子节点。

默认值: false

示例

描述:

需要锁定叶子节点状态

setting 举例:

```
var setting = {  
    keepLeaf : true,  
    .....  
};
```

相关参数

[keepParent](#)

edit_removeBtn

Boolean

概述

目录: [参数说明] --> [setting 详解]

设定当 zTree 处于编辑状态时，是否显示节点删除按钮。

当点击某节点的删除按钮时：

- 1、首先触发 [beforeRemove](#) 回调函数，用户可利用此回调函数进行删除确认等自定义操作。

默认值: true

示例

描述:

不显示删除按钮

setting 举例：

```
var setting = {  
    editable : true,  
    edit_removeBtn : false,  
    .....  
};
```

相关参数

[editable](#) | [edit_renameBtn](#)

edit_renameBtn

Boolean

概述

目录: [参数说明] --> [setting 详解]

设定当 **zTree** 处于编辑状态时，是否显示节点编辑按钮。

当点击某节点的编辑按钮，进入编辑状态 时：

- 1、在输入框内无法触发拖拽事件，可通过节点图标进行拖拽。
- 2、点击其他节点 或 当前节点的图标时，自动退出节点编辑状态。

默认值: **true**

示例

描述:

不显示编辑按钮

setting 举例：

```
var setting = {  
    editable : true,  
    edit_renameBtn : false,  
    .....  
};
```

相关参数

[editable](#) | [edit_removeBtn](#)

概述

目录: [参数说明] --> [setting 详解]

zTree 数据节点的根，全部节点数据都处于 `root.nodes` 内。

初始化**zTree**时，如果直接把节点数据放在 `setting.root.nodes` 内，则 `zTreeNodes` 参数可以省略。

默认值: { `nodes:[]` }

请注意如果修改了[nodesCol](#) 属性后，初始化时 `root` 内的 `nodes` 和 `zTreeNodes` 数据中全部的 `nodes` 属性 也需要被更换为修改后的名称

示例

描述:

初始化之前将数据节点放在 `setting` 内

`setting` 举例:

```
var setting = {
  root : {
    nodes: [
      { "name":"google", "url":"http://g.cn", "target":"_blank"}, 
      { "name":"baidu", "url":"http://baidu.com", "target":"_blank"}, 
      { "name":"sina", "url":"http://www.sina.com.cn", "target":"_blank"} 
    ]
  },
  .....
};
```

相关内容

[zTreeNodes](#)

showLine

Boolean

概述

目录: [参数说明] --> [setting 详解]

设置 zTree 是否显示节点之间的连线。

默认值: true

示例

描述:

设置 zTree 不显示节点之间的连线

setting 举例:

```
var setting = {  
    showLine : false,  
    .....  
};
```

概述

目录: [参数说明] --> [setting 详解]

设置 zTree 是否显示节点的图标。

默认值: true

允许接收 function 的引用, 以便于用户独立设置每一个节点是否显示图标。

function 格式举例: function showIconForTree(treeId, treeNode) {...}

示例

描述:

设置 zTree 不显示图标

setting 举例:

```
var setting = {  
    showIcon : false,  
    .....  
};
```

设置 zTree 仅仅 level=2 的节点不显示图标

setting 举例:

```
function showIconForTree(treeId, treeNode) {  
    return treeNode.level != 2;  
}  
var setting = {  
    showIcon : showIconForTree,  
    .....  
};
```

概述

目录: [参数说明] --> [setting 详解]

设置个性化文字样式，只针对 zTree 在节点上显示的<A>对象。

默认值: {}

JSON 格式为 JQuery css 方法中的 JSON 对象格式，例如: `{color:"#ff0011", background:"blue"}`

function 格式举例: `function setzTreeFont(treeId, treeNode) {...}` 返回值同上

示例

描述:

不修改CSS，设置全部节点 name 显示为红色

setting 举例:

```
var setting = {  
    fontCss : {color:"red"},  
    ....  
};
```

描述:

设置 level=0 的节点 name 显示为红色

setting 举例:

```
function setFontCss(treeId, treeNode) {  
    if (treeNode.level == 0) {  
        return {color:"red"};  
    }  
    return {};  
};  
var setting = {  
    fontCss : setFontCss,  
    ....  
};
```

nameCol

String

概述

目录: [参数说明] --> [setting 详解]

设置 zTree 显示节点名称的属性名称。

默认值: "name"

示例

描述:

设置 zTree 显示节点时, 将 treeNode 的 ename 属性当做节点名称

setting 举例:

```
var setting = {  
    nameCol : "ename",  
    .....  
};
```

nodesCol

String

概述

目录: [参数说明] --> [setting 详解]

设置 zTree 中保存子节点数据的属性名称。

默认值: "nodes"

请注意如果修改了此属性后, 初始化时 `root` 内的 `nodes` 和 `zTreeNode` 数据中全部的 `nodes` 属性 也需要被更换为修改后的名称

示例

描述:

设置 zTree 显示节点时, 将 `treeNode` 的 `child` 属性当做节点名称

setting 举例:

```
var setting = {  
    nodesCol : "child",  
    root:{ child:[....] },  
    .....  
};
```

checkedCol

String

概述

目录: [参数说明] --> [setting 详解]

设置 zTree 中保存 check 状态的属性名称。

默认值: "checked"

请勿与 **zTreeNode** 默认的参数冲突, 例如: [checkedOld](#)

示例

描述:

设置 zTree 显示节点时, 将 **treeNode** 的 **child** 属性当做节点名称
setting 举例:

```
var setting = {  
    checkedCol : "checked",  
    .....  
};
```

概述

目录: [参数说明] --> [setting 详解]

设置 zTree 节点展开、折叠时的动画速度 或 取消动画，设置方法同 JQuery 动画效果中 speed 参数。

设置为 "" 时，不显示动画效果，三种预定速度之一的字符串("slow", "normal", or "fast") 或 表示动画时长的毫秒数值(如: 1000)

默认值: "fast"

示例

描述:

设置为慢速显示动画效果

setting 举例:

```
var setting = {  
    expandSpeed : "slow",  
    .....  
};
```

概述

目录: [参数说明] --> [setting 详解]

鼠标移动到节点上时，显示用户自定义控件，显示隐藏状态同 zTree 内部的编辑、删除按钮

请务必与 [removeHoverDom](#) 同时使用；属于高级应用，使用时请确保对 zTree 比较了解。

Function 参数

treeId	String
---------------	--------

对应 zTree 的 [treeObjId](#)，便于用户操控

treeNode	JSON
-----------------	------

需要显示自定义控件的节点 JSON 数据对象

示例

描述：

设置鼠标移到节点上，在后面显示一个按钮

setting & function 举例：

```
var setting = {
    addHoverDom: addHoverDom,
    removeHoverDom: removeHoverDom,
    ....
};

function addHoverDom(treeId, treeNode) {
    var aObj = $("#" + treeId + " .a");
    if ($("#diyBtn_" + treeNode.id).length > 0) return;
    var editStr = "<span id='diyBtn_space_" + treeNode.id + "'> </span>" +
        "<button type='button' class='diyBtn1' id='diyBtn_" + treeNode.id +
        "' title='" + treeNode.name + "' onfocus='this.blur();'></button>";
    aObj.append(editStr);
    var btn = $("#diyBtn_" + treeNode.id);
    if (btn) btn.bind("click", function() { alert("diy Button for " + treeNode.name); });
}

function removeHoverDom(treeId, treeNode) {
    $("#diyBtn_" + treeNode.id).unbind().remove();
    $("#diyBtn_space_" + treeNode.id).unbind().remove();
}
....
```

相关参数

[removeHoverDom](#) | [addDiyDom](#)

概述

目录: [参数说明] --> [setting 详解]

鼠标移出节点时，隐藏用户自定义控件，显示隐藏状态同 zTree 内部的编辑、删除按钮

请务必与 [addHoverDom](#) 同时使用；属于高级应用，使用时请确保对 zTree 比较了解。

Function 参数

treeId	String
---------------	--------

对应 zTree 的 [treeObjId](#)，便于用户操控

treeNode	JSON
-----------------	------

需要隐藏自定义控件的节点 JSON 数据对象

示例

描述：

设置鼠标移到节点上，在后面显示一个按钮

setting & function 举例：

```
var setting = {
    addHoverDom: addHoverDom,
    removeHoverDom: removeHoverDom,
    ....
};

function addHoverDom(treeId, treeNode) {
    var aObj = $("#" + treeNode.tId + "_a");
    if ($("#diyBtn_" + treeNode.id).length > 0) return;
    var editStr = "<span id='diyBtn_space_" + treeNode.id + "'></span>" +
        "<button type='button' class='diyBtn1' id='diyBtn_" + treeNode.id +
        "' title='" + treeNode.name + "' onfocus='this.blur();'></button>";
    aObj.append(editStr);
    var btn = $("#diyBtn_" + treeNode.id);
    if (btn) btn.bind("click", function() { alert("diy Button for " + treeNode.name); });
}

function removeHoverDom(treeId, treeNode) {
    $("#diyBtn_" + treeNode.id).unbind().remove();
    $("#diyBtn_space_" + treeNode.id).unbind().remove();
}
....
```

相关参数

[addHoverDom | addDiyDom](#)

概述

目录: [参数说明] --> [setting 详解]

在节点上固定显示用户自定义控件

属于高级应用，使用时请确保对 zTree 比较了解。

Function 参数

treeId	String
---------------	--------

对应 zTree 的 [treeObjId](#)，便于用户操控

treeNode	JSON
-----------------	------

需要显示自定义控件的节点 JSON 数据对象

示例

描述:

设置节点后面显示一个按钮

setting & function 举例:

```
var setting = {
    addDiyDom: addDiyDom,
    ....
};

function addDiyDom(treeId, treeNode) {
    var aObj = $("#" + treeNode.tId + "_a");
    if ($("#diyBtn_" + treeNode.id).length > 0) return;
    var editStr = "<span id='diyBtn_space_" + treeNode.id + "'> </span>" +
        "<button type='button' class='diyBtn1' id='diyBtn_" + treeNode.id +
        "' title='" + treeNode.name + "' onfocus='this.blur();'></button>";
    aObj.append(editStr);
    var btn = $("#diyBtn_" + treeNode.id);
    if (btn) btn.bind("click", function() {alert("diy Button for " + treeNode.name);});
}
....
```

相关参数

[removeHoverDom](#) | [addDiyDom](#)

专门用于用户自定义各种 callback 回调函数。

默认值:

```
var setting = {  
    callback : {  
        beforeClick:null,           //详情参考: beforeClick  
        beforeRightClick:null,      //详情参考: beforeRightClick  
        beforeMouseDown:null,       //详情参考: beforeMouseDown  
        beforeMouseUp:null,        //详情参考: beforeMouseUp  
        beforeChange:null,         //详情参考: beforeChange  
        beforeDrag:null,           //详情参考: beforeDrag  
        beforeDrop:null,           //详情参考: beforeDrop  
        beforeRename:null,          //详情参考: beforeRename  
        beforeRemove:null,          //详情参考: beforeRemove  
        beforeExpand:null,          //详情参考: beforeExpand  
        beforeCollapse:null,        //详情参考: beforeCollapse  
        beforeAsync:null,           //详情参考: beforeAsync  
  
        nodeCreated:null,           //详情参考: nodeCreated  
        click:null,                 //详情参考: click  
        rightClick:null,             //详情参考: rightClick  
        mouseDown:null,              //详情参考: mouseDown  
        mouseUp:null,                //详情参考: mouseUp  
        change:null,                 //详情参考: change  
        drag:null,                  //详情参考: drag  
        drop:null,                  //详情参考: drop  
        rename:null,                 //详情参考: rename  
        remove:null,                 //详情参考: remove  
        expand:null,                  //详情参考: expand  
        collapse:null,                //详情参考: collapse  
        asyncSuccess:null,             //详情参考: asyncSuccess  
        asyncError:null,               //详情参考: asyncError  
    },  
    ....  
};
```

概述

目录: [参数说明] --> [setting 详解]

当 `checkable = true && checkStyle = "radio" && checkRadioType = "all"` 时, `checkRadioCheckedList` 用于记录当前被选择的节点。

不需要用户进行初始化, 属于内部参数。

概述

目录: [参数说明] --> [setting 详解]

zTree 用于记录当前被选中的节点数据 JSON 对象。

不需要用户进行初始化，属于内部参数。

可通过 [getSelectedNode\(\)](#) 方法获取

可通过 [selectNode\(treeNode\)](#) 方法设置某节点被选择

概述

目录: [参数说明] --> [setting 详解]

zTree 用于记录当前处于编辑状态的节点数据 JSON 对象。

不需要用户进行初始化，属于内部参数。

当前 zTree 节点是否正在被拖拽时, 如果该节点是父节点, 且展开状态, 则将其临时折叠; 此参数就是用于记录这个状态。

不需要用户进行初始化, 属于内部参数。

正在拖拽时, 如果被拖拽节点在拖拽操作前为展开状态的父节点, 设置为 `true`, 拖拽结束后恢复为 `false`

概述

目录: [参数说明] --> [setting 详解]

标识当前 zTree 节点是否正在被拖拽。

不需要用户进行初始化，属于内部参数。

默认值: **false**

正在拖拽时: **true**

expandTriggerFlag

Boolean

概述

目录: [参数说明] --> [setting 详解]

用于确定 zTree 何时触发 expand 或 collapse 事件。

不需要用户进行初始化，属于内部参数。

概述

目录: [参数说明] --> [setting 详解]

zTree 的唯一标识，初始化后，等于 用户定义的 **zTree** 容器的 **id** 属性值。

不需要用户进行初始化，属于内部参数。

概述

目录: [参数说明]

zTreeNodes 是 zTree 的全部节点数据集合, 采用由 JSON 对象组成的数据结构。

`zTreeNodes = setting.root.nodes`

属性

[checked](#) | [click](#) | [icon](#) | [iconSkin](#) | [isParent](#) | [name](#) | [nodes](#) | [open](#) | [target](#) | [url](#) | [*自定义*](#)
[\[checkedOld\]](#) | [\[checkboxFocus\]](#) | [\[check_True_Full\]](#) | [\[check_False_Full\]](#)
[\[editNameStatus\]](#) | [\[isAjasing\]](#) | [\[isHover\]](#) | [\[isFirstNode\]](#) | [\[isLastNode\]](#)
[\[level\]](#) | [\[parentNode\]](#) | [\[tId\]](#)

相关参数

[root](#)

checked

Boolean

概述

目录: [参数说明] --> [zTreeNodes 详解]

当 setting.[checkable](#) = true 时有效, 设定节点的 CheckBox 是否被选中

默认值: false

示例

描述:

需要对某节点初始化时选中其 CheckBox

zTreeNodes 举例:

```
var zTreeNodes = [{  
    checked : true,  
    .....  
}];
```

相关参数

[checkedOld](#)

click

String

概述

目录: [参数说明] --> [zTreeNodes 详解]

设定节点在鼠标点击后做的事情，相当于 `onclick="...."` 的内容，可用于一些简单操作，如果过于复杂的，建议通过 [click](#) 事件进行控制处理

默认值: 空

示例

描述:

点中某节点后，弹出该节点名称

zTreeNodes 举例:

```
var zTreeNodes = [{  
    click : "alert('myName')",  
    .....  
}];
```

相关事件

[click](#)

icon

String

概述

目录: [参数说明] --> [zTreeNodes 详解]

设定节点的自定义图标，以替换 `css` 样式中配置的普通图标。（设定时请注意指定图标的相对路径是否正确）

对于整体图标更换，请参考 [zTreeStyle.css](#) 说明，整体替换

默认值：空

示例

描述：

设定某节点自定义图标

zTreeNodes 举例：

```
var zTreeNodes = [{  
    icon : "folder.gif",  
    .....  
}];
```

设定节点的自定义图标在 CSS 中对应的自定义 ClassName (从 v1.02 版本开始支持此功能)

对于 CSS 图标具体定义方法, 请参考 [zTreeStyle.css](#) 说明

因为 CSS 选择器功能的支持问题, 因此不兼容 IE6, 做项目必须兼容 IE6 的朋友, 请不要使用此功能

按照 CSS 的优先规则, 为避免异常, 建议将自定义的 CSS 属性放在 ".tree li button.ico_open"、".tree li button.ico_close"、".tree li button.ico_docu" 这几个定义之后。

示例

描述:

设定某节点自定义图标为 sim 系列中的图标

zTreeNodes 举例:

```
var zTreeNodes = [{  
    iconSkin : "sim1",  
    .....  
}];
```

isParent

Boolean

概述

目录: [参数说明] --> [zTreeNodes 详解]

设置某节点是否为父节点。

当 `setting.async = true` 且 `isParent = true`、该节点的 `nodes` 不存在 或 `length = 0`, 当点击该节点时会触发异步获取子节点的事件。

默认值: 如果用户未设置该属性, 则根据节点是否有子节点进行自动设置

示例

描述:

需要对某节点被点击时触发异步获取子节点的事件

zTreeNodes 举例:

```
var zTreeNodes = [{  
    isParent : true,  
    ....  
}];
```

相关参数

[nodes](#) | [parentNode](#)

name

String

概述

目录: [参数说明] --> [zTreeNodes 详解]

节点显示的名称。

示例

描述:

设置节点显示的名称为: Test

zTreeNodes 举例:

```
var zTreeNodes = [{  
    name : "Test",  
    .....  
}];
```

概述

目录: [参数说明] --> [zTreeNodes 详解]

某节点的子节点集合。

当 `setting.async = true` 且 `isParent = true`、该节点的 `nodes` 不存在 或 `length = 0`, 当点击该节点时会触发异步获取子节点的事件。

如果不使用 `nodes` 作为子节点的属性, 请修改 `nodesCol` 属性

示例

描述:

设置某节点的子节点数据

zTreeNodes 举例:

```
var zTreeNodes = [{  
    nodes : [ {...}, {...}, ...],  
    ....  
}];
```

相关参数

[isParent](#) | [parentNode](#)

概述

目录: [参数说明] --> [zTreeNodes 详解]

设置有子节点的节点初始化展开状态。

对于不需要异步获取子节点信息的父节点有效。

默认值: `true`

示例

描述:

设置某父节点初始化时展开

`zTreeNodes` 举例:

```
var zTreeNodes = [{  
    open : true,  
    .....  
}];
```

target

String

概述

目录: [参数说明] --> [zTreeNodes 详解]

示例

描述:

对于存在 [url](#) 属性的节点，点击后跳转的目标，同 超链接的 target属性 (_blank, _self等)

zTreeNodes 举例:

```
var zTreeNodes = [{  
    target : "_blank",  
    .....  
}];
```

相关参数

[url](#)

url

String

概述

目录: [参数说明] --> [zTreeNodes 详解]

指定节点被点击后的跳转页面 URL 地址

示例

描述:

设置某节点点击时，跳转到 g.cn

zTreeNodes 举例:

```
var zTreeNodes = [{  
    url : "http://g.cn",  
    .....  
}];
```

相关参数

[target](#)

自定义

String

概述

目录: [参数说明] --> [zTreeNodes 详解]

示例

描述:

设置某节点用户自定义信息

zTreeNodes 举例:

```
var zTreeNodes = [{  
    id : "001",  
    .....  
}];
```

checkedOld

Boolean

概述

目录: [参数说明] --> [zTreeNodes 详解]

用户点击节点的 `checkBox` 或 `radio` 时，用于保留初始化的 [checked](#) 属性

放弃了v2.2版本之前的 `checkedNew` 参数，不需要用户进行初始化，属于内部参数。

相关参数

[checked](#)

概述

目录: [参数说明] --> [zTreeNodes 详解]

用于设置节点的 `checkBox` 或 `radio` 的样式

用于设置节点的 `checkBox` 或 `radio` 的样式

用于设置节点的 `checkBox` 或 `radio` 的样式

editNameStatus

Boolean

概述

目录: [参数说明] --> [zTreeNodes 详解]

记录节点是否处于名称编辑状态，且只有当 `editable` 属性为 `true` 时有效。

不需要用户进行初始化，属于内部参数。

相关参数

[editable](#)

zTree内部用来避免节点重复异步加载。

不需要用户进行初始化，属于内部参数。

isFirstNode

Boolean

概述

目录: [参数说明] --> [zTreeNodes 详解]

记录节点是否为同级节点中的第一个节点，主要用于画线使用

不需要用户进行初始化，属于内部参数。

相关参数

[isLastNode](#)

在添加自定义控件时，设置节点 `hover` 状态，对于 `zTree` 本身无作用，仅供用户操作参考。

isLastNode

Boolean

概述

目录: [参数说明] --> [zTreeNodes 详解]

记录节点是否为同级节点中的最后一个节点，主要用于画线使用

不需要用户进行初始化，属于内部参数。

相关参数

[isFirstNode](#)

概述

目录: [参数说明] --> [zTreeNodes 详解]

记录节点处于第几级节点, 根节点 `level = 0`

不需要用户进行初始化, 属于内部参数。

概述

目录: [参数说明] --> [zTreeNodes 详解]

记录节点的父节点数据 node 对象, 根节点 parentNode = null, 主要便于数据计算

不需要用户进行初始化, 属于内部参数。

相关参数

[nodes](#) | [isParent](#)

tId

String

概述

目录: [参数说明] --> [zTreeNodes 详解]

zTree对每个节点自动生成的唯一标识ID，生成规则: [treeObjId](#) + "_" + 计数，请用户在zTree的页面上避免使用此种规则定义其他对象的 ID。

不需要用户进行初始化，属于内部参数。

相关参数

[treeObjId](#)

getSelectedNode()

返回值: JSON Object

概述

目录: [方法] --> [获取]

获取 zTree 当前被选中的节点数据 JSON 对象。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

如果当前没有被选择的节点, 则返回 null

在 1.x 版本中名称为 [getCurNode\(\)](#)

示例

描述:

获取 zTree 当前被选中的节点数据

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var selectedNode = zTreeObj.getSelectedNode();  
.....
```

相关方法

[selectNode\(treeNode\)](#) | [cancelSelectedNode\(\)](#)

相关参数

[curTreeNode](#)

getNodeByTId(tID)

返回值: JSON Object

概述

目录: [方法] --> [获取]

根据某个节点数据的 [tId](#) 属性获取该节点的 JSON 数据对象。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后, 返回的 `zTreePlugin` 对象执行此方法

如果当前没有对应 `tID` 的节点, 则返回 `null`

参数

tID

String

该属性值是 `zTree` 在初始化时, 自动赋予, 详情参考 [tId](#)

示例

描述:

获取某个 `tID` 的节点数据

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
var tID = "abc_1"  
var node = zTreeObj.getNodeByTId(tID);  
....
```

相关方法

[getNodeByParam\(key, value\)](#) | [getNodesByParam\(key, value, parentNode\)](#)

概述

目录: [方法] --> [获取]

根据节点数据的属性精确搜索满足条件的的 JSON 数据对象。

如果有多个同样属性值的节点，则只返回第一个找到的节点，如果需要获取全部满足条件的节点集合，请参考

[getNodesByParam\(key, value, parentNode\)](#)

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后，返回的 `zTreePlugin` 对象执行此方法

如果当前没有满足条件的节点，则返回 `null`

参数

key String

进行搜索的节点数据的属性名称

value

进行搜索的节点数据的属性值，一定要保证数据类型匹配

示例

描述:

获取 `id = 10` 的节点数据

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var node = zTreeObj.getNodeByParam("id", 10);  
.....
```

相关方法

[getNodesByParam\(key, value, parentNode\)](#) | [getNodesByParamFuzzy\(key, value, parentNode\)](#)
[getNodeByTId\(tID\)](#)

概述

目录: [方法] --> [获取]

根据节点数据的属性精确搜索指定节点 `parentNode` 下面的子节点中的 JSON 数据对象集合。

如果只需要一个满足条件的节点, 请参考 [getNodeByParam\(key, value\)](#)

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 `zTreePlugin` 对象执行此方法

简单遍历 `Array` 就能得到全部结果, 如果当前没有满足条件的节点, 则返回长度为 0 的 `Array` 数组

参数

key	String
-----	--------

进行搜索的节点数据的属性名称

value

进行搜索的节点数据的属性值, 一定要保证数据类型匹配

parentNode	JSON Object
------------	-------------

指定的父节点, 如果查找全部节点, 请设置 `parentNode` 为 `null` 即可。

示例

描述:

获取 `level = 1` 的节点数据

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var nodes = zTreeObj.getNodesByParam("level", 1);  
.....
```

获取 当前选择的父节点下子节点名字是"test" 的节点数据

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var pNode = zTreeObj.getSelectedNode();  
var nodes = zTreeObj.getNodesByParam("name", "test", pNode);  
.....
```

相关方法

[getNodesByParamFuzzy\(key, value, parentNode\)](#) | [getNodeByParam\(key, value\)](#) | [getNodeByTId\(tID\)](#)

getNodesByParamFuzzy(key, value, parentNode)

返回值: Array(JSON)

概述

目录: [方法] --> [获取]

根据节点数据的属性模糊搜索指定节点 `parentNode` 下面的子节点中的 JSON 数据对象集合。

注意: 可以进行模糊匹配的仅限于 `string` 类型的数据

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 `zTreePlugin` 对象执行此方法

简单遍历 `Array` 就能得到全部结果, 如果当前没有满足条件的节点, 则返回长度为 0 的 `Array` 数组

参数

key	String
-----	--------

进行搜索的节点数据的属性名称

value

进行搜索的节点数据的属性值, 一定要保证数据类型匹配

parentNode	JSON Object
------------	-------------

指定的父节点, 如果查找全部节点, 请设置 `parentNode` 为 `null` 即可。

示例

描述:

获取 名字包含“测试”字符串 的全部节点数据

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
var nodes = zTreeObj.getNodesByParamFuzzy("name", "测试");  
....
```

获取 当前选择的父节点下子节点名字包含“测试”字符串 的全部节点数据

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
var pNode = zTreeObj.getSelectedNode();  
var nodes = zTreeObj.getNodesByParamFuzzy("name", "测试", pNode);  
....
```

相关方法

[getNodesByParam\(key, value, parentNode\)](#) | [getNodeByParam\(key, value\)](#) | [getNodeByTId\(tID\)](#)

getNodeIndex(treeNode)

返回值: Number

概述

目录: [方法] --> [获取]

获取某节点在同一层级节点中的序号（从0开始）。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后，返回的 `zTreePlugin` 对象执行此方法

如果该节点不存在，则返回 -1

参数

treeNode

JSON

需要获取序号的节点 JSON 数据

示例

描述:

获取当前被选中的节点所在层级中的序号

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
var zIndex = zTreeObj.getNodeIndex(zTreeObj.getSelectedNode());  
....
```

getNodes()

返回值: Array(JSON)

概述

目录: [方法] --> [获取]

返回 zTree 根内部的全部节点数据 (是 zTree 中使用的标准数据, 子节点都存在于父节点的数据中)

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

示例

描述:

获取全部节点数据

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var nodes = zTreeObj.getNodes();  
.....
```

相关方法

[addNodes\(parentNode, newNodes, isSilent\)](#) | [moveNode\(targetNode, treeNode, moveType\)](#)
[updateNode\(treeNode, checkTypeFlag\)](#) | [removeNode\(treeNode\)](#)

相关参数

[root](#)

getCheckedNodes(**checked**)

返回值: Array(JSON)

概述

目录: [方法] --> [获取]

返回 zTree 当前checkBox / radio 输入框被选择 或 未选择的节点集合 (简单遍历 Array 就能得到全部节点)

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

在 1.x 版本中名称为 [getSelectedNodes\(selected\)](#)

参数

selected

Boolean

设置获取节点的类型 --- **true**: 获取被选择的节点(默认); **false**: 获取未选择的节点

示例

描述:

获取全部被选择的节点数据

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
var nodes = zTreeObj.getCheckedNodes(); //或 zTreeObj.getCheckedNodes(true);  
....
```

获取全部未选择的节点数据

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
var nodes = zTreeObj.getCheckedNodes(false);  
....
```

相关方法

[getChangeCheckedNodes\(\)](#) | [checkAllNodes\(checked\)](#)

getChangeCheckedNodes()

返回值: Array(JSON)

概述

目录: [方法] --> [获取]

返回 zTree 当前checkBox / radio 输入框选择状态被改变的节点集合, 即 `checked != checkedOld`

(简单遍历 `Array` 就能得到全部节点)

请通过 zTree 核心函数 `zTree(setting, [zTreeNode])` 运行后, 返回的 zTreePlugin 对象执行此方法

补充: 如果想在不刷新 zTree 的情况下, 获取 zTree 每次点击后被自动转换的节点集合, 可以在每次 `change` 事件后, 使用本方法, 并将所有节点的 `checked` 属性值赋给 `checkedOld` 属性即可。

示例

描述:

获取全部被选择的节点数据

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var nodes = zTreeObj.getChangeCheckedNodes();  
.....
```

相关方法

[getCheckedNodes\(checked\)](#) | [checkAllNodes\(checked\)](#)

概述

目录: [方法] --> [获取]

获取 zTree 当前配置信息的 JSON 对象。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

为了更好的保证配置信息的正常使用, 请注意以下几点:

- 1、得到的 JSON 对象是 zTree setting 的复制
- 2、不返回 root 属性, 获取节点数据, 请参考 [getNodes\(\)](#) 方法
- 3、如果希望修改后的 setting 生效, 请参考 [updateSetting\(setting\)](#) 方法

示例

描述:

获取 zTree 当前配置信息

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var xSetting = zTreeObj.getSetting();  
.....
```

相关方法

[updateSetting\(setting\)](#)

transformTozTreeNodes(simpleTreeNodes)

返回值: Array(JSON)

概述

目录: [方法] --> [获取]

将简单 Array 格式转换为 zTree 使用的标准格式。 (是 zTree 中使用的标准数据，子节点都存在于父节点的数据中)

使用此方法，请务必设置节点唯一标识属性名称 [treeNodeKey](#) 和 父节点唯一标识属性名称 [treeNodeParentKey](#)，并且让数据满足父子关系，注意：简单 Array 数据中必须要让父节点在子节点之前。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后，返回的 zTreePlugin 对象执行此方法

参数

simpleTreeNodes Array(JSON) / JSON

需要被转换的简单 Array 格式数据，如果是一个 JSON 对象，则被简单封装为长度为1的 Array 数组

示例

描述:

将简单 Array 格式转换为zTree使用的标准格式，并加到zTree根节点

js 代码:

```
....  
var setting = {  
    isSimpleData : true,  
    treeNodeKey : "id",  
    treeNodeParentKey : "pId",  
    ....  
};  
var zTreeObj = zTree(setting, zTreeNodes);  
var simpleTreeNodes = [  
    {"id":1, "pId":0, "name":"test1"},  
    {"id":11, "pId":1, "name":"test11"},  
    {"id":12, "pId":1, "name":"test12"},  
    {"id":111, "pId":11, "name":"test111"}  
];  
var treeNodes = zTreeObj.transformTozTreeNodes(simpleTreeNodes);  
zTreeObj.addNode(null, treeNodes);  
....
```

相关方法

[transformToArray\(treeNodes\)](#)

相关参数

[treeNodeKey](#) | [treeNodeParentKey](#)

transformToArray(treeNodes)

返回值: Array(JSON)

概述

目录: [方法] --> [获取]

将 zTree 使用的标准格式转换为简单 Array 格式，便于将数据返回给后台。（简单遍历 Array 就能得到全部节点）

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后，返回的 zTreePlugin 对象执行此方法

参数

treeNodes

Array(JSON) / JSON

需要被转换的标准 zTree 格式数据

示例

描述:

将 zTree 当前全部数据转换为简单 Array 格式

js 代码:

```
.....  
var zTreeObj = zTree(setting, treeNodes);  
var treeNodes = zTreeObj.getNodes();  
var simpleTreeNodes = zTreeObj.transformToArray(treeNodes);  
.....
```

相关方法

[transformTozTreeNodes\(simpleTreeNodes\)](#)

概述

目录: [方法] --> [操作]

在指定节点下增加子节点。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 `zTreePlugin` 对象执行此方法

参数

parentNode	JSON Object
-------------------	-------------

指定的父节点, 如果增加根节点, 请设置 `parentNode` 为 `null` 即可。

newNodes	Array(JSON)
-----------------	-------------

需要增加的节点数据 JSON 对象集合, 支持将节点的n级子节点一次性增加, 只需要符合`zTree`的节点数据结构即可。详情参考 [zTreeNode 详解](#)

isSilent	Boolean
-----------------	---------

设定增加节点后是否展开其父节点。`isSilent = true` 时, 不展开父节点, 其他值或缺省状态都自动展开。

示例

描述:

增加根节点

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
var newNodes = [ {...}, {...}, ...];  
var nodes = zTreeObj.addNodes(null, newNodes);  
.....
```

相关方法

[getNodes\(\)](#) | [moveNode\(targetNode, treeNode, moveType\)](#) | [updateNode\(treeNode, checkTypeFlag\)](#)
[removeNode\(treeNode\)](#)

相关参数

[nodes](#)

expandAll(expandSign)

返回值: 无

概述

目录: [方法] --> [操作]

让 zTree 展开全部节点。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

参数

expandSign

Boolean

展开 (true) 或 折叠 (false) 标识

示例

描述:

展开全部节点

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.expandAll(true);  
.....
```

相关方法

[expandNode\(treeNode, expandSign, sonSign\)](#)

expandNode(treeNode, expandSign, sonSign)

返回值: 无

概述

目录: [方法] --> [操作]

让 zTree 展开指定节点。

在2.0中，此方法增加了将操作节点设定为焦点的功能，避免当节点很多并出现滚动条时，操作的节点在可视区域以外。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后，返回的 zTreePlugin 对象执行此方法

参数

treeNode

JSON

指定要 展开 或 折叠 的节点 JSON 数据

expandSign

Boolean

展开 (true) 或 折叠 (false) 标识

sonSign

Boolean

展开 或 折叠 是否影响子孙级节点标识

示例

描述:

只展开节点的下一级节点

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.expandNode(treeNode, true, false);  
.....
```

相关方法

[expandAll\(expandSign\)](#)

moveNode(targetNode, treeNode, moveType)

返回值: 无

概述

目录: [方法] --> [操作]

将某节点移动到其他节点下。

在2.2中，增加了 moveType 参数，允许指定移动的相对位置。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后，返回的 zTreePlugin 对象执行此方法

参数

targetNode

JSON

指定移动到的目标节点 JSON 数据，如果移入根节点，请设置 targetNode 为 null 即可

treeNode

JSON

指定被移动的节点 JSON 数据

moveType

String

指定移动到目标节点的相对位置

"inner": 成为子节点（默认值），"before": 成为同级前一个节点，"after": 成为同级后一个节点

示例

描述:

将节点1（treeNode1）移动到节点2（treeNode2）下

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.moveNode(treeNode2, treeNode1);  
....
```

将节点1（treeNode1）移动到节点2（treeNode2）同级前一个节点

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.moveNode(treeNode2, treeNode1, "before");  
....
```

相关方法

[getNodes\(\)](#) | [addNodes\(parentNode, newNodes, isSilent\)](#) | [updateNode\(treeNode, checkTypeFlag\)](#)
[removeNode\(treeNode\)](#)

刷新zTree。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 `zTreePlugin` 对象执行此方法

描述:

刷新zTree

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.refresh();  
.....
```

removeNode(treeNode)

返回值: 无

概述

目录: [方法] --> [操作]

删除某节点。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后, 返回的 `zTreePlugin` 对象执行此方法

参数

treeNode

JSON

指定需要被删除的节点 JSON 数据

示例

描述:

将节点1 (treeNode1) 删除

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.removeNode(treeNode1);  
.....
```

相关方法

[getNodes\(\)](#) | [addNodes\(parentNode, newNodes, isSilent\)](#) | [moveNode\(targetNode, treeNode, moveType\)](#)
[updateNode\(treeNode, checkTypeFlag\)](#)

概述

目录: [方法] --> [操作]

将某节点设置为被选中状态。

在2.0中，此方法增加了将节点设定为焦点的功能，避免当节点很多并出现滚动条时，被选中的节点在可视区域以外。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后，返回的 `zTreePlugin` 对象执行此方法

参数

treeNode

JSON

指定需要设置为被选中状态的节点 JSON 数据

示例

描述:

将节点1（treeNode1）设置为被选中状态

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.selectNode(treeNode1);  
....
```

相关方法

[getSelectedNode\(\)](#) | [cancelSelectedNode\(\)](#)

checkAllNodes(**checked**)

返回值: 无

概述

目录: [方法] --> [操作]

在 [checkable](#) 为 true 时, 设置全部节点的选中状态。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

参数

checked

Boolean

true: 全部选中; **false:** 全部取消选中

示例

描述:

将全部节点设置为被选中状态

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.checkAllNodes(true);  
.....
```

相关方法

[getCheckedNodes\(checked\)](#) | [getChangeCheckedNodes\(\)](#)

setEditable(editable)

返回值: 无

概述

目录: [方法] --> [操作]

设置 zTree 是否为可拖拽的编辑状态。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

参数

editable

Boolean

可编辑 (true) 或 不可编辑 (false) 标识

示例

描述:

设置 zTree 为可编辑状态

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.setEditable(true);  
.....
```

相关参数

[editable](#)

cancelSelectedNode()

返回值: 无

概述

目录: [方法] --> [操作]

将被选中的节点设置为未被选中状态。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

v2.2版本将原先拼写错误的cancelSelectedNode 修改为 cancelSelectedNode, 但为了保证以前用户的代码正常, 因此继续保留 cancelSelectedNode 方法。

示例

描述:

将被选中的节点设置为未被选中状态

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
....  
zTreeObj.cancelSelectedNode();  
.....
```

相关方法

[getSelectedNode\(\)](#) | [selectNode\(treeNode\)](#)

updateNode(treeNode, checkTypeFlag)

返回值: 无

概述

目录: [方法] --> [操作]

更新某节点数据，主要用于该节点显示属性的更新。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后，返回的 `zTreePlugin` 对象执行此方法

如果在 `nameCol` 属性内指定了名称数据的属性名称，则更新节点数据时，需要修改对应的属性数据。

v2.2版本中修正了`updateNode`方法，可针对`name`、`target`、`url`、`icon`、`iconSkin`这几个用于显示效果的参数进行更新，其他用于`zTreeNodes`的参数请不要随意更新，对于展开节点，还请调用 `expandNode`方法，因此请勿随意修改`open`属性。

参数

treeNode JSON

指定需要更新的节点 JSON 数据

checkTypeFlag Boolean

由用户决定是否按照`setting.checkType`属性进行父子节点的选中状态联动

默认: `false`

示例

描述:

修改了节点 (`treeNode1`) 的名称后，在 `zTree` 上进行更新

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
.....  
treeNode1.name = "test Name";  
zTreeObj.updateNode(treeNode1, true);  
.....
```

相关方法

[getNodes\(\)](#) | [addNodes\(parentNode, newNodes, isSilent\)](#) | [moveNode\(targetNode, treeNode, moveType\)](#)
[removeNode\(treeNode\)](#)

相关参数

[checkType](#) | [nameCol](#)

概述

目录: [方法] --> [操作]

更新 zTree 当前配置信息。

请通过 zTree 核心函数 [zTree\(setting, \[zTreeNode\]\)](#) 运行后, 返回的 zTreePlugin 对象执行此方法

此方法主要用于 zTree 更为灵活的高级操作, 因此使用中请注意以下几点:

1、更新后配置信息会立即生效, 因此对于 [showLine](#)、[editable](#)、[checkable](#)... 等影响 zTree 显示的配置最好不要随便修改, 如果必须修改, 请在更新后使用 [refresh\(\)](#) 方法

2、建议对以下属性可随时修改: [asyncUrl](#)、[asyncParam](#)、[asyncParamOther](#)、[checkType](#)、[expandSpeed](#)、[callback](#)

3、无法修改 [treeObjId](#) 和 [root](#) 属性, 修改节点数据, 请参考其他 Node 相关方法

4、获取当前配置信息, 请参考 [getSetting\(\)](#) 方法

示例

描述:

更新 zTree 当前配置中异步获取数据的URL (Demo中“CheckBox 演示”已经使用了此方法)

js 代码:

```
....  
var zTreeObj = zTree(setting, zTreeNodes);  
var xSetting = zTreeObj.getSetting();  
xSetting.asyncUrl = "node.jsp";  
zTreeObj.updateSetting(xSetting);  
....
```

相关方法

[getSetting\(\)](#)

reAsyncChildNodes(parentNode, reloadType)

返回值: 无

概述

目录: [方法] --> [操作]

指定父节点进行异步加载子节点（已经加载过的父节点可反复使用此方法重新加载）。

请通过 `zTree` 核心函数 [zTree\(setting, \[zTreeNodes\]\)](#) 运行后，返回的 `zTreePlugin` 对象执行此方法

参数

parentNode

JSON

指定需要异步加载的节点 JSON 数据

`parentNode = null` 时，相当于从根节点 `Root` 进行异步加载

`parentNode.isParent = false` 时，不进行异步加载

reloadType

String

指定是清空后重新加载还是追加子节点

`reloadType = "refresh"` 时，表明清空后重新加载，否则进行追加子节点处理。

示例

描述:

指定父节点1 (`treeNode1`) 重新异步加载

js 代码:

```
.....  
var zTreeObj = zTree(setting, zTreeNodes);  
zTreeObj.reAsyncChildNodes(treeNode1, "refresh");  
.....
```

相关参数

[asyncUrl](#)

beforeClick(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeClick(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 callback 参数内，即可在触发 zTree 的 click 事件之前得到相应信息，并根据自己的需求确定是否该节点可以点击。

该事件在节点被点击后最先触发，如果返回 false，则中断 click 事件，也不会触发 click 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 treeObjId，便于用户操控

treeNode	JSON
-----------------	------

被点击的节点 JSON 数据对象

示例

描述:

禁止 zTree 的 click 操作

js 代码:

```
.....
var setting = {
  callback : {
    beforeClick: zTreeBeforeClick,
    .....
  },
  .....
};

function zTreeBeforeClick(treeId, treeNode) {
  return false;
}
.....
```

相关事件

click(event, treeId, treeNode)
--

beforeRightClick(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeRightClick(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 [callback](#) 参数内，即可在触发 zTree 的 rightClick 事件之前得到相应信息，并根据自己的需求确定是否该节点可以右键点击。

该事件在节点被鼠标右键点击后最先触发，如果返回 false，则中断 rightClick 事件，也不会触发 rightClick 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 treeObjId，便于用户操控

treeNode	JSON
-----------------	------

被点击的节点 JSON 数据对象

示例

描述:

禁止 zTree 的 rightClick 操作

js 代码:

```
.....
var setting = {
    callback : {
        beforeRightClick: zTreeBeforeRightClick,
        .....
    },
    .....
};

function zTreeBeforeRightClick(treeId, treeNode) {
    return false;
}
.....
```

相关事件

rightClick(event, treeId, treeNode)

beforeMouseDown(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeMouseDown(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 callback 参数内，即可在根据自己的需求确定是否该节点可以触发 zTree 的 mouseDown 事件。

该事件在节点被鼠标按键按下后最先触发，如果返回 **false**，仅仅不会触发 mouseDown 回调函数，对于其他事件无任何影响。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 treeObjId，便于用户操控

treeNode	JSON
-----------------	------

触发鼠标按键按下事件的节点 JSON 数据对象，如果 **treeNode = null**，则表明是 zTree 的根节点 Root。

zTree 根据页面元素获取 **treeNode** 的规则与编辑模式下拖拽时定位目标节点的规则相同。

示例

描述:

禁止 zTree 的 mouseDown 操作

js 代码:

```
.....
var setting = {
    callback : {
        beforeMouseDown: zTreeBeforeMouseDown,
        .....
    },
    .....
};

function zTreeBeforeMouseDown(treeId, treeNode) {
    return false;
}
.....
```

相关事件

mouseDown(event, treeId, treeNode)

beforeMouseUp(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `beforeMouseUp(treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可在根据自己的需求确定是否该节点可以触发 zTree 的 `mouseUp` 事件。

该事件在节点被鼠标按键抬起后最先触发，如果返回 `false`，仅仅不会触发 `mouseUp` 回调函数，对于其他事件无任何影响。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode	JSON
-----------------	------

触发鼠标按键抬起事件的节点 JSON 数据对象，如果 `treeNode = null`, 则表明是 zTree 的根节点 Root。

zTree 根据页面元素获取 `treeNode` 的规则与编辑模式下拖拽时定位目标节点的规则相同。

示例

描述:

禁止 zTree 的 `mouseUp` 操作

js 代码:

```
.....
var setting = {
    callback : {
        beforeMouseUp: zTreeBeforeMouseUp,
        .....
    },
    .....
};

function zTreeBeforeMouseUp(treeId, treeNode) {
    return false;
}
.....
```

相关事件

mouseUp(event, treeId, treeNode)
--

beforeChange(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeChange(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 [callback](#) 参数内，即可在触发 zTree 的 change 事件之前得到相应信息，并根据自己的需求确定是否该节点可以点击修改checkbox 或 radio 的选择状态。

该事件在节点的 checkbox 或 radio 被点击后最先触发，如果返回 false，则中断 change 事件，也不会触发 [change](#) 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 [treeObjId](#)，便于用户操控

treeNode	JSON
-----------------	------

被点击 checkbox 或 radio 的节点 JSON 数据对象

示例

描述:

禁止 zTree 的 change 操作

js 代码:

```
.....
var setting = {
    callback : {
        beforeChange: zTreeBeforeChange,
        .....
    },
    .....
};

function zTreeBeforeChange(treeId, treeNode) {
    return false;
}
.....
```

相关事件

[change\(event, treeId, treeNode\)](#)

相关参数

[checkable](#)

beforeDrag(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeDrag(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 callback 参数内，即可在触发 zTree 的 drag 事件之前得到相应信息，并根据自己的需求确定是否该节点可以拖拽。

该事件在节点开始被拖拽时最先触发，如果返回 false，则中断 drag 事件，也不会触发 drag 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 treeObjId，便于用户操控

treeNode	JSON
-----------------	------

被拖拽的节点 JSON 数据对象

示例

描述:

禁止 zTree 的 drag 操作

js 代码:

```
.....
var setting = {
    callback : {
        beforeDrag: zTreeBeforeDrag,
        .....
    },
    .....
};

function zTreeBeforeDrag(treeId, treeNode) {
    return false;
}
.....
```

相关事件

[drag\(event, treeId, treeNode\) | beforeDrop\(treeId, treeNode, targetNode, moveType\)](#)
[drop\(event, treeId, treeNode, targetNode, moveType\)](#)

相关参数

[editable](#)

beforeDrop(treeId, treeNode, targetNode, moveType)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeDrop(treeId, treeNode, targetNode, moveType) 函数（函数名可以自定义），并配置在 setting 的 [callback](#) 参数内，即可在触发 zTree 的 drop 事件之前得到相应信息，并根据自己的需求确定是否此次拖拽结果可以生效。

该事件在节点拖拽操作结束时最先触发，如果返回 **false**，则中断 drop 事件，也不会触发 [drop](#) 回调函数。

参数

treeId String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 [treeObjId](#)，便于用户操控

多个 zTree 之间进行拖拽时，返回目标节点的 [treeObjId](#)

treeNode JSON

被拖拽的节点 JSON 数据对象。

如果没有拖拽到合法节点内，则返回 **null**

targetNode JSON

成为 treeNode 父节点的目标节点 JSON 数据对象。

如果没有拖拽到合法节点内 或 拖拽成为根节点，则返回 **null**

moveType String

指定移动到目标节点的相对位置

"inner": 成为子节点, "before": 成为同级前一个节点, "after": 成为同级后一个节点

示例

描述:

禁止 zTree 的 drop 操作

js 代码:

```
....  
var setting = {  
    callback : {  
        beforeDrop: zTreeBeforeDrop,  
        ....  
    },  
    ....  
};  
....  
function zTreeBeforeDrop(treeId, treeNode, targetNode, moveType) {  
    return false;  
}  
....
```

相关事件

[drop\(event, treeId, treeNode, targetNode, moveType\)](#)
[beforeDrag\(treeId, treeNode\) | drag\(event, treeId, treeNode\)](#)

相关参数

[editable](#)

beforeRename(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `beforeRename(treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可在显示编辑名称的输入框之前得到相应信息，并根据自己的需求确定是否该节点可以修改名称。

该事件在节点的编辑按钮被点击后最先触发，如果返回 `false`，则不会进入名称编辑状态 和 触发 `rename` 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode	JSON
-----------------	------

将要进入名称编辑状态的节点 JSON 数据对象

示例

描述:

禁止节点进入名称编辑状态

js 代码:

```
.....
var setting = {
    callback : {
        beforeRename: zTreeBeforeRename,
        .....
    },
    .....
};

function zTreeBeforeRename(treeId, treeNode) {
    return false;
}
.....
```

相关事件

rename(event, treeId, treeNode)

相关参数

editable edit_renameBtn

beforeRemove(treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `beforeRemove(treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可在删除节点之前得到相应信息，并根据自己的需求确定是否该节点可以被删除。

该事件在节点的删除按钮被点击后最先触发，如果返回 `false`，则中断 `remove` 事件，也不会触发 `remove` 回调函数。

参数

treeId String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode JSON

将要删除的节点 JSON 数据对象

示例

描述：

禁止删除任何节点

js 代码：

```
....  
var setting = {  
    callback : {  
        beforeRemove: zTreeBeforeDel,  
        ....  
    },  
    ....  
};  
....  
function zTreeBeforeDel(treeId, treeNode) {  
    return false;  
}  
....
```

相关事件

[remove\(event, treeId, treeNode\)](#)

相关参数

[editable | edit_removeBtn](#)

beforeExpand(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeExpand(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 callback 参数内，即可在触发 zTree 的 expand 事件之前得到相应信息，并根据自己的需求确定是否该节点可以展开。

该事件在点击处于折叠状态父节点的(+)图标或双击该节点后触发，如果返回 false，则中断 expand 事件，也不会触发 expand 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 treeObjId，便于用户操控

treeNode	JSON
-----------------	------

将要被展开的节点 JSON 数据对象

示例

描述:

禁止 zTree 的 expand 操作

js 代码:

```
.....
var setting = {
    callback : {
        beforeExpand: zTreeBeforeExpand,
        .....
    },
    .....
};

function zTreeBeforeExpand(treeId, treeNode) {
    return false;
}
.....
```

相关事件

expand(event, treeId, treeNode)
beforeCollapse(treeId, treeNode) collapse(event, treeId, treeNode)

beforeCollapse(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeCollapse(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 callback 参数内，即可在触发 zTree 的 collapse 事件之前得到相应信息，并根据自己的需求确定是否该节点可以折叠。

该事件在点击处于展开状态父节点的(+)图标或双击该节点后触发，如果返回 false，则中断 collapse 事件，也不会触发 collapse 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 treeObjId，便于用户操控

treeNode	JSON
-----------------	------

将要被折叠的节点 JSON 数据对象

示例

描述:

禁止 zTree 的 collapse 操作

js 代码:

```
.....
var setting = {
    callback : {
        beforeCollapse: zTreeBeforeCollapse,
        .....
    },
    .....
};

function zTreeBeforeCollapse(treeId, treeNode) {
    return false;
}
.....
```

相关事件

collapse(event, treeId, treeNode)
beforeExpand(treeId, treeNode) expand(event, treeId, treeNode)

beforeAsync(treeId, treeNode)

返回值: Boolean

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 beforeAsync(treeId, treeNode) 函数（函数名可以自定义），并配置在 setting 的 callback 参数内，即可在触发 zTree 的 异步加载事件之前得到相应信息，并根据自己的需求确定是否该节点可以异步加载子节点。

该事件在节点需要异步加载时最先触发，如果返回 false，则中断异步加载事件，也不会触发 asyncSuccess 或 asyncError 回调函数。

参数

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 treeObjId，便于用户操控

treeNode	JSON
-----------------	------

需要异步加载子节点的节点 JSON 数据对象

示例

描述:

禁止 zTree 某节点的异步加载操作

js 代码:

```
.....
var setting = {
  callback : {
    beforeAsync: zTreeBeforeAsync,
    .....
  },
  .....
};

function zTreeBeforeAsync(treeId, treeNode) {
  if (treeNode.id == 1) return false;
  return true;
}
.....
```

相关事件

<u>asyncSuccess(event, treeId, treeNode, msg)</u>
<u>asyncError(event, treeId, treeNode, XMLHttpRequest, textStatus, errorThrown)</u>

nodeCreated(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onNodeCreated(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `nodeCreated` 事件。

该事件在节点被加入到 zTree 内渲染完毕后触发。

参数

event	js event 对象
--------------	-------------

标准的 js event 对象

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode	JSON
-----------------	------

当前渲染完毕的节点 JSON 数据对象

示例

描述:

获取最后一个被渲染的 zTree 节点

js 代码:

```
.....
var setting = {
    callback : {
        nodeCreated: zTreeOnNodeCreated,
        .....
    },
    .....
};

var lastRenderingNode = null;
function zTreeOnNodeCreated(event, treeId, treeNode) {
    lastRenderingNode = treeNode;
}
.....
```

click(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onClick(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 click 事件。

该事件在节点被点击后触发。

如果用户配置了 `beforeClick` 方法，并返回 `false`，将无法触发 click 事件。

在 zTree v1.x 版本中，用户无法配置，只能按照要求编写指定名称的对应函数，比较死板。

参数

event	js event 对象
--------------	-------------

`onClick` 事件返回的标准 `event` 对象

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode	JSON
-----------------	------

被点击的节点 JSON 数据对象

示例

描述:

每次点击 zTree 节点后，弹出该节点的 tId 以及 name 信息

js 代码:

```
.....
var setting = {
  callback : {
    click: zTreeOnClick,
    .....
  },
  .....
};

function zTreeOnClick(event, treeId, treeNode) {
  alert(treeNode.tId + ", " + treeNode.name);
}
.....
```

相关事件

beforeClick(treeId, treeNode)

rightClick(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onRightClick(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `rightClick` 事件。

该事件在节点被鼠标右键点击后触发。

如果用户配置了 `beforeRightClick` 方法，并返回 `false`，将无法触发 `rightClick` 事件。

注意：只要将 `function` 的引用赋给 `rightClick` 属性，则右键点击zTree时，将屏蔽浏览器的右键菜单。

参数

event js event 对象

onClick 事件返回的标准 event 对象

treeId String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode JSON

被右键点击的节点 JSON 数据对象，如果 `treeNode = null`, 则表明右键点击到 zTree 的根节点 Root 上。

zTree 根据页面元素获取 `treeNode` 的规则与编辑模式下拖拽时定位目标节点的规则相同。

示例

描述：

每次右键点击 zTree 节点后，弹出该节点的 tId 以及 name 信息

js 代码：

```
....  
var setting = {  
    callback : {  
        rightClick: zTreeOnRightClick,  
        ....  
    },  
    ....  
};  
....  
function zTreeOnRightClick(event, treeId, treeNode) {  
    if (treeNode)  
        alert(treeNode.tId + ", " + treeNode.name);  
    else  
        alert("is root");  
}  
....
```

相关事件

[beforeRightClick\(treeId, treeNode\)](#)

mouseDown(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onMouseDown(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `mouseDown` 事件。

该事件在节点被鼠标按键按下后触发。

如果用户配置了 `beforeMouseDown` 方法，并返回 `false`，将无法触发 `mouseDown` 事件。

注意：鼠标按键按下对于 zTree 本身不进行任何其他操作，仅仅为了便于用户进行扩展应用而制作的。

参数

event js event 对象

`onMouseDown` 事件返回的标准 `event` 对象

treeId String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便
于用户操控

treeNode JSON

被鼠标按键按下的节点 JSON 数据对象，如果 `treeNode = null`，则表明鼠标按键按下时焦点在 zTree 的根节点 Root 上。

zTree 根据页面元素获取 `treeNode` 的规则与编辑模式下拖拽时定位目标节点的规则相同。

示例

描述：

每次鼠标按键按下后，弹出对应节点的 `tId` 以及 `name` 信息

js 代码：

```
....  
var setting = {  
    callback : {  
        mouseDown: zTreeOnMouseDown,  
        ....  
    },  
    ....  
};  
....  
function zTreeOnMouseDown(event, treeId, treeNode) {  
    if (treeNode)  
        alert(treeNode.tId + ", " + treeNode.name);  
    else  
        alert("is root");  
}  
....
```

相关事件

[beforeMouseDown\(treeId, treeNode\)](#)

mouseUp(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onMouseUp(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `mouseUp` 事件。

该事件在节点被鼠标按键抬起后触发。

如果用户配置了 `beforeMouseUp` 方法，并返回 `false`，将无法触发 `mouseUp` 事件。

注意：鼠标按键抬起对于 zTree 本身不进行任何其他操作，仅仅为了便于用户进行扩展应用而制作的。

参数

event

js event 对象

`onMouseUp` 事件返回的标准 `event` 对象

treeId

String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便
于用户操控

treeNode

JSON

被鼠标按键抬起的节点 JSON 数据对象，如果 `treeNode = null`，则表明鼠标按键抬起时焦点在 zTree 的根节点 Root 上。

zTree 根据页面元素获取 `treeNode` 的规则与编辑模式下拖拽时定位目标节点的规则相同。

示例

描述:

每次鼠标按键抬起后，弹出对应节点的 `tId` 以及 `name` 信息

js 代码:

```
....  
var setting = {  
    callback : {  
        mouseUp: zTreeOnMouseUp,  
        ....  
    },  
    ....  
};  
....  
function zTreeOnMouseUp(event, treeId, treeNode) {  
    if (treeNode)  
        alert(treeNode.tId + ", " + treeNode.name);  
    else  
        alert("is root");  
}  
....
```

相关事件

`beforeMouseUp(treeId, treeNode)`

change(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onChange(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `change` 事件。

该事件在节点的 `CheckBox` 被点击时触发。

如果用户配置了 `beforeChange` 方法，并返回 `false`，将无法触发 `change` 事件。

在 zTree v1.x 版本中，用户无法配置，只能按照要求编写指定名称的对应函数，比较死板。

参数

event

js event 对象

onChange 事件返回的标准 event 对象

treeId

String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode

JSON

被点击 checkbox 或 radio 的节点 JSON 数据对象

示例

描述:

每次点击 zTree 上的 checkbox 或 radio 后，弹出该节点的 `tId` 以及 `name` 信息

js 代码:

```
.....
var setting = {
  callback : {
    change: zTreeOnChange,
    .....
  },
  .....
};

function zTreeOnChange(event, treeId, treeNode) {
  alert(treeNode.tId + ", " + treeNode.name);
}
.....
```

相关事件

[beforeChange\(treeId, treeNode\)](#)

相关参数

[checkable](#)

drag(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onDrag(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `drag` 事件。

该事件在节点开始被拖拽时触发

如果用户配置了 `beforeDrag` 方法，并返回 `false`，将无法触发 `drag` 事件。

在 zTree v1.x 版本中，用户无法配置，只能按照要求编写指定名称的对应函数，比较死板。

参数

event	js event 对象
--------------	-------------

标准的 js event 对象

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode	JSON
-----------------	------

被拖拽的节点 JSON 数据对象

示例

描述:

每次拖拽开始时，弹出该节点的 `tId` 以及 `name` 信息

js 代码:

```
.....
var setting = {
  callback : {
    drag: zTreeOnDrag,
    .....
  },
  .....
};

function zTreeOnDrag(event, treeId, treeNode) {
  alert(treeNode.tId + ", " + treeNode.name);
}
.....
```

相关事件

[beforeDrag\(treeId, treeNode\)](#) | [beforeDrop\(treeId, treeNode, targetNode, moveType\)](#)
[drop\(event, treeId, treeNode, targetNode, moveType\)](#)

相关参数

[editable](#)

drop(event, treeId, treeNode, targetNode, moveType)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onDrop(event, treeId, treeNode, targetNode, moveType)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 drop 事件。

该事件在节点拖拽操作结束时触发

如果用户配置了 `beforeDrop` 方法，并返回 `false`，将无法触发 drop 事件。

在 zTree v1.x 版本中，用户无法配置，只能按照要求编写指定名称的对应函数，比较死板。

参数

event js event 对象

标准的 js event 对象

treeId String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

多个 zTree 之间进行拖拽时，返回目标节点的 `treeObjId`

treeNode JSON

被拖拽的节点 JSON 数据对象。

如果没有拖拽到合法节点内，则返回 `null`

targetNode JSON

成为 `treeNode` 父节点的目标节点 JSON 数据对象。

如果没有拖拽到合法节点内 或 拖拽成为根节点，则返回 `null`

moveType String

指定移动到目标节点的相对位置

"inner": 成为子节点, "before": 成为同级前一个节点, "after": 成为同级后一个节点

示例

描述:

拖拽结束，弹出被拖拽节点和成为父节点的 tId 以及 name 信息

js 代码:

```
....  
var setting = {  
    callback : {  
        drop: zTreeOnDrop,  
        ....  
    },  
    ....  
};  
....  
function zTreeOnDrop(event, treeId, treeNode, targetNode, moveType) {  
    if (treeNode) alert("treeNode = " + treeNode.tId + ", " + treeNode.name);  
    if (targetNode) alert("targetNode = " + targetNode.tId + ", " + targetNode.name);  
}  
....
```

相关事件

[beforeDrop\(treeId, treeNode, targetNode, moveType\)](#)

[beforeDrag\(treeId, treeNode\) | drag\(event, treeId, treeNode\)](#)

相关参数

[editable](#)

rename(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onRename(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `rename` 事件。

该事件在编辑节点名称完毕并生效后触发，即 编辑输入框 `onblur` 事件后触发。

如果用户配置了 `beforeRename` 方法，并返回 `false`，将根本无法进入名称编辑状态。

参数

event js event 对象

标准的 js event 对象

treeId String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便与用户操控

treeNode JSON

将要进入名称编辑状态的节点 JSON 数据对象

示例

描述:

弹出被编辑名称后的节点的 `tId` 以及 `name` 信息

js 代码:

```
....  
var setting = {  
    callback : {  
        rename: zTreeOnRename,  
        ....  
    },  
    ....  
};  
....  
function zTreeOnRename(event, treeId, treeNode) {  
    alert(treeNode.tId + ", " + treeNode.name);  
}  
....
```

相关事件

[beforeRename\(treeId, treeNode\)](#)

相关参数

[editable | edit_renameBtn](#)

remove(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onRemove(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `remove` 事件。

该事件在点击节点的删除按钮后触发。

如果用户配置了 `beforeRemove` 方法，并返回 `false`，将无法触发 `remove` 事件。

参数

event

js event 对象

标准的 js event 对象

treeId

String

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便与用户操控

treeNode

JSON

将要删除的节点 JSON 数据对象

示例

描述:

弹出被删除的节点的 tId 以及 name 信息

js 代码:

```
....  
var setting = {  
    callback : {  
        remove: zTreeOnRemove,  
        ....  
    },  
    ....  
};  
....  
function zTreeOnRemove(event, treeId, treeNode) {  
    alert(treeNode.tId + ", " + treeNode.name);  
}  
....
```

相关事件

[beforeRemove\(treeId, treeNode\)](#)

相关参数

[editable | edit_removeBtn](#)

expand(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onExpand(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 setting 的 `callback` 参数内，即可监听 zTree 的 `expand` 事件。

该事件在节点被鼠标点击导致展开后触发。

如果用户配置了 `beforeExpand` 方法，并返回 `false`，将无法触发 `expand` 事件。

在以下情况展开节点，不会触发此事件： 拖拽节点时、客户端用 js 操作 zTree 进行展开、选中、增加、移动节点操作时...

参数

event	js event 对象
--------------	-------------

标准 event 对象

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 callback 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode	JSON
-----------------	------

被展开的节点 JSON 数据对象

示例

描述:

每次展开 zTree 节点后，弹出该节点的 tId 以及 name 信息

js 代码:

```
.....
var setting = {
    callback : {
        expand: zTreeOnExpand,
        .....
    },
    .....
};

function zTreeOnExpand(event, treeId, treeNode) {
    alert(treeNode.tId + ", " + treeNode.name);
}
.....
```

相关事件

[beforeExpand\(treeId, treeNode\)](#)
[beforeCollapse\(treeId, treeNode\)](#) | [collapse\(event, treeId, treeNode\)](#)

collapse(event, treeId, treeNode)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上，编写 `onCollapse(event, treeId, treeNode)` 函数（函数名可以自定义），并配置在 `setting` 的 `callback` 参数内，即可监听 zTree 的 `collapse` 事件。

该事件在节点被鼠标点击导致折叠后触发。

如果用户配置了 `beforeCollapse` 方法，并返回 `false`，将无法触发 `collapse` 事件。

在以下情况展开节点，不会触发此事件： 拖拽节点时、客户端用 js 操作 zTree 进行折叠节点操作时...

参数

event	js event 对象
--------------	-------------

标准 event 对象

treeId	String
---------------	--------

因为 zTree 支持页面上同时存在多个 zTree 的实例，因此在 `callback` 回调函数内返回对应 zTree 的 `treeObjId`，便于用户操控

treeNode	JSON
-----------------	------

被折叠的节点 JSON 数据对象

示例

描述:

每次折叠 zTree 节点后，弹出该节点的 tId 以及 name 信息

js 代码:

```
.....
var setting = {
    callback : {
        collapse: zTreeOnCollapse,
        .....
    },
    .....
};

function zTreeOnCollapse(event, treeId, treeNode) {
    alert(treeNode.tId + ", " + treeNode.name);
}
.....
```

相关事件

beforeCollapse(treeId, treeNode)
beforeExpand(treeId, treeNode) expand(event, treeId, treeNode)

zTreeOnAsyncSuccess(event, treeId, msg)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上, 编写 zTreeOnAsyncSuccess(event, treeId, msg) 函数, 即可随意监听 zTree 节点的异步获取节点成功 事件。

该事件在异步操作结束并成功时触发。

在 zTree v1.x 版本中, 用户无法配置, 只能按照要求编写指定名称的对应函数, 比较死板。

参数

event

js event 对象

标准的 js event 对象

treeId

String

因为 zTree 支持页面上同时存在多个 zTree 的实例, 因此在 callback 回调函数内返回对应 zTree 的 [treeObjId](#), 便
于用户操控

msg

String

异步获取的节点数据字符串, 主要便于用户调试使用。

示例

描述:

异步获取数据成功后, 弹出得到的数据字符串

js 代码:

```
....  
function zTreeOnAsyncSuccess(event, treeId, msg) {  
    alert(msg);  
}  
.....
```

相关事件

[beforeAsync\(treeId, treeNode\)](#) | [asyncError\(event, treeId, treeNode, XMLHttpRequest, textStatus, errorThrown\)](#)

相关参数

[async](#)

zTreeOnAsyncError(event, treeId, XMLHttpRequest, textStatus, errorThrown)

概述

目录: [事件] --> [callback 回调函数]

用户在使用 zTree 的页面上, 编写 zTreeOnAsyncError(event, treeId, XMLHttpRequest, textStatus, errorThrown) 函数, 即可随意监听 zTree 节点的 异步获取节点失败 事件。

该事件在异步操作结束并失败时触发。

在 zTree v1.x 版本中, 用户无法配置, 只能按照要求编写指定名称的对应函数, 比较死板。

参数

event

js event 对象

标准的 js event 对象

treeId

String

因为 zTree 支持页面上同时存在多个 zTree 的实例, 因此在 callback 回调函数内返回对应 zTree 的 [treeObjId](#), 便
于用户操控

XMLHttpRequest

String

标准 XMLHttpRequest 对象, 请参考 [JQuery API 文档](#)。

textStatus

String

请求状态: success, error, 请参考 [JQuery API 文档](#)。

errorThrown

String

errorThrown 只有当异常发生时才会被传递, 请参考 [JQuery API 文档](#)。

示例

描述:

异步获取数据失败后, 弹出错误信息

js 代码:

```
....  
function zTreeOnAsyncError(event, treeId, XMLHttpRequest, textStatus, errorThrown) {  
    alert(XMLHttpRequest);  
}  
....
```

相关事件

[beforeAsync\(treeId, treeNode\)](#) | [asyncSuccess\(event, treeId, treeNode, msg\)](#)

相关参数

[async](#)

zTree 的 [nodeCreated](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 [click](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 [change](#) 事件 关键字, 核心内部使用, 仅供参考

在 1.x 版本中名称为 ZTREE_CHECK

zTree 的 [rename](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 [remove](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 [drag](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 [drop](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 [asyncSuccess](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 [asyncError](#) 事件 关键字, 核心内部使用, 仅供参考

zTree 的 开关图标 id 后缀命名定义, 核心内部使用, 仅供参考

zTree 的 节点自定义图标 id 后缀命名定义, 核心内部使用, 仅供参考

zTree 的 节点checkbox 或 radio对象 id 后缀命名定义, 核心内部使用, 仅供参考

zTree 的 节点编辑按钮命名定义, 核心内部使用, 仅供参考

zTree 的 节点删除按钮命名定义, 核心内部使用, 仅供参考

zTree 的 节点超链接层 id 后缀命名定义, 核心内部使用, 仅供参考

`zTree` 的 节点用于显示名称 `name` 的容器命名定义，核心内部使用，仅供参考

`zTree` 的 节点用于编辑名称 `name` 的输入框命名定义, 核心内部使用, 仅供参考

zTree 的 节点内 **UL** 层 **id** 后缀命名定义, 核心内部使用, 仅供参考

zTree 的 节点图标 **id** 后缀（节点位置相关）命名定义，核心内部使用，仅供参考

`zTree` 的 节点图标 `id` 后缀（节点位置相关）命名定义，核心内部使用，仅供参考

zTree 的 节点图标 **id** 后缀（节点位置相关）命名定义，核心内部使用，仅供参考

`zTree` 的节点自定义图标 `id` 后缀（文件夹、末级节点区分相关）命名定义，核心内部使用，仅供参考

zTree 的节点自定义图标 `id` 后缀（文件夹、末级节点区分相关）命名定义，核心内部使用，仅供参考

zTree 的节点自定义图标 `id` 后缀（文件夹、末级节点区分相关）命名定义，核心内部使用，仅供参考

`zTree` 的节点被选中状态 `class` 命名定义, 核心内部使用, 仅供参考

zTree 的节点被选中后且处于编辑状态 **class** 命名定义, 核心内部使用, 仅供参考

zTree 的根成为拖拽源节点的目标状态 **class** 命名定义, 核心内部使用, 仅供参考

Class_TmpTargetNode

概述

目录: [常量] --> [className相关]

`zTree` 的节点成为拖拽源节点的目标状态 `class` 命名定义, 核心内部使用, 仅供参考

checkStyle 的 CheckBox 常量, 核心内部使用, 仅供参考

checkStyle 的 Radio 常量, 核心内部使用, 仅供参考

CheckBox_Default

概述

目录: [常量] --> [CheckBox & Radio 相关]

CheckBox 或 Radio 的 class 命名定义, 核心内部使用, 仅供参考

CheckBox_False

概述

目录: [常量] --> [CheckBox & Radio 相关]

CheckBox 或 Radio 的 class 命名定义, 核心内部使用, 仅供参考

Radio 的 选择节点个数限制范围 命名定义, 核心内部使用, 仅供参考

Radio_Type_Level

概述

目录: [常量] --> [CheckBox & Radio 相关]

Radio 的 选择节点个数限制范围 命名定义, 核心内部使用, 仅供参考

zTree 的 节点为避免鼠标微小划动导致的误拖拽操作，而设定的最小移动像素定义，核心内部使用，仅供参考

zTree 的 节点移动到目标节点类型之一（成为子节点），核心内部使用，仅供参考

zTree 的 节点移动到目标节点类型之一（成为同级前一个节点），核心内部使用，仅供参考

zTree 的 节点移动到目标节点类型之一（成为同级后一个节点），核心内部使用，仅供参考

在添加自定义控件时，设置节点 `hover` 状态，对于 `zTree` 本身无作用，仅供用户操作参考。