عنوان: **تعدادی از توابع kendo UI Treeview**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۱۳ ۱:۱۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Kendo UI

kendo ui یکی از جذاب‌ترین و بهترین فریم ورک‌های HTML5 است که استفاده از آن بین برنامه نویسان جا افتاده است و تلریک هم پشتیبانی خوبی از آن به عمل آورده است. من هم به تازگی از شیء treeview آن استفاده کردم و موقعیکه کارم با شیء Treeview به پایان رسید، یک فایل کوچک جاوااسکریپت به کار اضافه شد که شامل تعدادی از توابع چون حذف گره و ... بود که تصمیم گرفتم بر اساس مستندات و نیازهای عمومی، تعداد این توابع را بالا ببرم که برای استفاده در صفحات و پروژه‌های دیگر و بالا بردن امتیاز استفاده مجدد از کد از آن استفاده کنم. سورس آن در گیت هاب موجود است.
بعد از صدا زدن فایل‌های مورد نیاز کندو، این فایل جاوااسکریپتی را هم به پروژه اضافه کنید. بیشتر توابع تست شده و جواب گرفته و فکر نکنم مشکلی باشد؛ هر چند عیب یابی آن هم ساده است و مشکلی برای برطرف کردن آن وجود ندارد و هر تابع دو یا سه خط بیشتر نیست.

معرفی توابع

```
var treeview;

    function FindTreeViewObj(objName) {
        treeview = $(objName).data("kendoTreeView");
    }
```

اولین و مهمترین تابع می‌باشد که باید قبل از همه در زمان لود پروژه بعد از ایجاد درخت آن را صدا بزنید، در صورتی که صدا زده نشود، بقیه‌ی توابع کار نخواهند کرد. این تابع وظیفه دارد شیء درخت معرفی شده را پیدا کرده و داخل یک متغیر عمومی به اسم treeview قرار دهد:

```
FindTreeViewObj("#treeview");
```

```
function GetSelectedNode() {
        return treeview.select();
    }
```

گره انتخابی را باز می‌گرداند:

```
GetSelectedNode();
```

```
function DisableSelectedNode() {
        treeview.enable(GetSelectedNode(), false);
    }
```

گره انتخابی را غیرفعال می‌کند:

```
DisableSelectedNode();
```

```
function EnableSelectedNode() {
        treeview.enable(GetSelectedNode(), true);
    }
```

گره انتخابی را فعال می‌کند:

```
EnableSelectedNode();
```

```
function EnableAllNodes() {
        treeview.enable(".k-item");
    }
```

همه‌ی گره‌ها را فعال می‌کند:

```
EnableAllNodes();
```

```
function ExpandAllNodes() {
        treeview.expand(".k-item");
    }
```

همه‌ی گره‌ها را به سمت فرزندان باز می‌کند:

```
ExpandAllNodes();
```

```
function CollapseAllNodes() {
        treeview.collapse(".k-item");
    }
```

همه گره‌ها به سمت والد را جمع می‌کند:

```
CollapseAllNodes();
```

```
function RemoveSelectedNode() {
        treeview.remove(GetSelectedNode());
    }
```

گره انتخابی را حذف می‌کند:

```
RemoveSelectedNode()
```

```
function FilterTreeView(filterText) {
        if (filterText !== "") {
            treeview.dataSource.filter({
                field: "text",
                operator: "contains",
                value: filterText
            });
        } else {
            treeview.dataSource.filter({});
        }
    }
```

گره‌هایی که عبارت مد نظر در آن‌ها باشند، نمایش میابند:

```
FilterTreeView('my node')
```

```
function SortTreeView(sortType) {
        treeview.dataSource.sort({
            field: "text",
            dir: sortType
        });
    }
```

گره‌ها را به صورت صعودی یا نزولی مرتب می‌کند. مقادیر پارامتر ورودی آن یا "asc" است یا "desc"

```
SortTreeView('asc');
SortTreeView('desc');
```

```
function GetSelectedDataItem() {
        return treeview.dataItem(GetSelectedNode());
    }
```

قسمت اطلاعاتی یک گره که شامل مواردی چون عنوان یا Id است را باز می‌گرداند:

```
GetSelectedDataItem();
```

```
function GetSelectedNodeId() {
        var data = GetSelectedDataItem();
        return data.id;
    }
```

Id گره را بر میگرداند:

```
GetSelectedNodeId();
```

```
function GetSelectedNodeText() {
        var data = GetSelectedDataItem();
        return data.Name;
    }
```

متن یا مقدار گره را باز میگرداند:

```
GetSelectedNodeText();
```

```
function SetSelectedNodeText(value) {
        var node = GetSelectedNode();
        treeview.text(node, value);
    }
```

مقدار گره را به مقدار جدید تغییر می‌دهد:

```
SetSelectedNodeText('new value');
```

```
function GetNodeByText(text) {
        return treeview.findByText(text);
    }
```

اولین گره با این متن را باز میگرداند:

```
GetNodeByText('mynode');
```

```
function GetNodeByText(id) {
        return treeview.findByUid(id);
    }
```

گره ای با این Id را باز میگرداند:

```
GetNodeByText(4);
```

```
function InsertAfter(item, nextItem) {
        treeview.insertAfter({ text: "item" }, GetNodeByText(nextItem));
    }
```

متن دو گره را دریافت می‌کند، گره‌ای را بر اساس متن پارامتر دوم پیدا کرده و بعد از آن گره اول را با مقدار پارامتر اول درج می‌کند.

```
InsertAfter('new item', 'old item')
```

```
function MoveToAfter(firstItem, secondItem) {
        treeview.insertAfter(GetNodeByText(firstItem), GetNodeByText(secondItem));
    }
```

مقدار گره‌ها را دریافت می‌کند و بعد از پیدا کردن آن‌ها، گره اول را به بعد از گره دوم انتقال می‌دهد:

```
MoveToAfter('firstItem', 'secondItem');
```

```
function InsertBefore(item, nextItem) {
        treeview.insertBefore({ text: "item" }, GetNodeByText(nextItem));
    }

    function MoveToBefore(firstItem, secondItem) {
        treeview.insertBefore(GetNodeByText(firstItem), GetNodeByText(secondItem));
    }
```

دو تابع بعد مثل توابع بالا هستند و فقط به قبل آن اضافه می‌کند یا انتقال می‌دهد.

```
InsertBefore('new item', 'old item');
MoveToBefore('firstItem', 'secondItem');
```

```
function GetParent(node) {
        return treeview.parent(node);
    }
```

والد گره انتخابی را بر میگرداند:

```
GetParent(GetSelectedNode());
```

```
function Toggle(node) {
        treeview.toggle(node);
    }
```

گره را (در صورت داشتن فرزند) باز و بسته می‌کند:

```
Toggle(GetSelectedNode());
```

```
function NewNode(nodeText, nodeValue, selectedNode) {
        treeview.append({
            Name: nodeText,
            Id: nodeValue
        }, selectedNode);
    }
```

گره جدیدی را اضافه می‌کند. پارامتر اول مقدار گره، پارامتر دوم Id گره و پارامتر سوم در صورتی که بخواهید یک زیر گره باشد، باید گره والد را به آن پاس کنید و درصورتی که زیر گره نیست آن را نال مقدار دهی کنید.

```
NewNode('new node', 1, null);
NewNode('new sub node', 2, GetSelectedNode());
```