

## Simple Use Case

```
var tree = new YAHOO.widget.TreeView("treeDiv1");
var root = tree.getRoot();
var tmpNode = new YAHOO.widget.TextNode("mylabel",
    root, false);
tree.draw();
```

Places a Tree control in the HTML element whose ID attribute is "treeDiv1"; adds one node to the top level of the Tree and renders.

## Constructor: YAHOO.widget.TreeView

```
YAHOO.widget.TreeView(str | element target);
```

Arguments:

- (1) **Element id or reference:** HTML ID or element reference for the element being into which the Tree's DOM structure will be inserted.

## Nodes: TextNode, MenuNode, HTMLNode

### TextNode (for simple labeled nodes)

```
YAHOO.widget.TextNode(obj | str oData, Node obj
    oParent[, b expanded]);
```

Arguments:

- (1) **Associated data:** A string containing the node label or an object containing str `label`, str `href`, and any other custom members desired. If no `oData.href` is provided, clicking on the TextNode's intrinsic `<a>` tag will invoke the node's `expand` method.
- (2) **Parent node:** The node object of which the new node will be a child; for top-level nodes, the parent is the Tree's root node.
- (3) **Expanded state:** A boolean indicating whether the node is expanded when the Tree is rendered.

### MenuNode (for auto-collapsing node navigation)

MenuNodes are identical to TextNodes in construction and behavior, except that only one MenuNode can be open at any time for a given level of depth.

### HTMLNode

```
YAHOO.widget.HTMLNode(obj | str HTML, Node obj
    oParent[, b expanded, b hasIcon]);
```

Arguments:

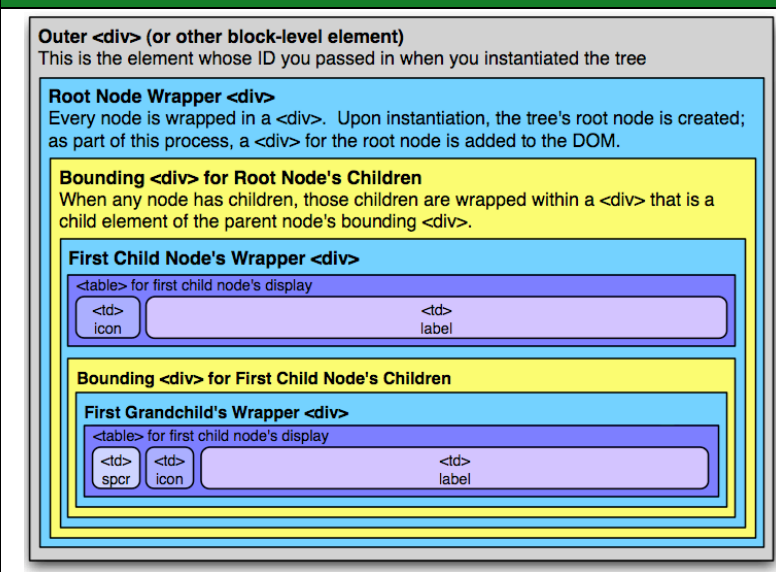
- (1) **HTML:** A string containing markup for the node's label; no event handlers are provided by default for this markup.
- (2) **Parent node:** See TextNode.
- (3) **Expanded state:** See TextNode.
- (4) **Has Icon:** Stipulates whether the expanded/contracted icon (and its horizontal space) should be rendered for this node.

## Interesting Moments in TreeView

| Event      | Fires...  | Arguments                          |
|------------|---|------------------------------------|
| onExpand   | ...before a node expands;<br>return false to cancel.  | Node obj<br><i>expanding node</i>  |
| onCollapse | ...before a node collapses;<br>return false to cancel | Node obj<br><i>collapsing node</i> |

These are methods of TreeView's prototype object; you will most commonly use them by overriding them on your instance: `tree.onExpand = myFn;`

## TreeView DOM Structure



## Solutions:

**Dynamically load child nodes:**

```
fnLoadData = function(oNode, fnCallback) {
    //create child nodes for oNode
    var tmp = new YAHOO.widget.TextNode("lbl", oNode);
    fnCallback(); //then fire callback
    var tree = new Yahoo.widget.TreeView(targetEl);
    tree.setDynamicLoad(fnLoadData);
    var root = tree.getRoot();
    var node1 = new YAHOO.widget.TextNode("1st", root);
    tree.draw();
}
```

## Dependencies

TreeView requires the YAHOO global object.

YAHOO.widget.  
TreeView: Properties

id (str)  
nodeCount (int)

YAHOO.widget.  
TreeView: Methods

collapseAll()  
draw()  
expandAll()  
getNodesByProperty()  
getRoot()  
popNode(node) returns detached node, which can then be reinserted  
removeChildren(node)  
removeNode(node, b  
 autorefresh)  
setDynamicLoad(fn)

YAHOO.widget.Node:  
Properties

Inherited by Text, Menu, & HTML nodes

data (obj)  
expanded (b)  
hasIcon (b)  
href (str)  
iconMode (i)  
nextSibling (node obj)  
parent (node obj)  
previousSibling (node  
 obj)  
target (str)  
tree (TreeView obj)

YAHOO.widget.Node:  
Method

Inherited by Text, Menu, & HTML nodes

appendTo()  
collapse()  
collapseAll()  
expand()  
expandAll()  
getEl()  
getHTML() includes children  
getNodeHTML() sans children  
hasChildren()  
insertBefore()  
insertAfter()  
isDynamic()  
isRoot()  
setDynamicLoad()  
toggle()