```
<style type="text/css">
/*margin and padding on body element
  can introduce errors in determining
  element position and are not recommended;
  we turn them off as a foundation for YUI
  CSS treatments. */
body {
  margin:0;
  padding:0;
}
</style>

<link rel="stylesheet" type="text/css" href="yui_2.7.0b-lib/fonts/fonts-min.css" />
<link rel="stylesheet" type="text/css" href="yui_2.7.0b-
lib/treeview/assets/skins/sam/treeview.css" />
<script type="text/javascript" src="yui_2.7.0b-lib/yahoo-dom-event/yahoo-dom-
event.js"></script>
<script type="text/javascript" src="yui_2.7.0b-lib/connection/connection-
min.js"></script>
<script type="text/javascript" src="yui_2.7.0b-lib/treeview/treeview-
min.js"></script>


<!--begin custom header content for this example-->
<style>
#treeDiv1 {background: #fff; margin-top:1em; padding:1em; min-height:7em;}
</style>
<!--end custom header content for this example-->

</head>

<body class=" yui-skin-sam">


<h1>Dynamically Loading Node Data</h1>

<div class="exampleIntro">
  <p>In many cases, you'll want to avoid rendering your <a
href="http://developer.yahoo.com/yui/treeview/">TreeView Control</a> with a
full dataset.  Rather, you'll want to load all visible nodes immediately and
then retrieve data only when needed for nodes that aren't visible when the
control is first loaded.  This example shows you how to do that.</p>

<p>In the TreeView instance below, we've loaded all "top-level" nodes into the
page as soon as the page loads; these nodes contain the names of many Indian
states.  When a node is expanded, we use <a
href="http://developer.yahoo.com/yui/connection/">Connection Manager</a> to
access <a
href="http://developer.yahoo.com/search/web/V1/relatedSuggestion.html">a Yahoo!
Search web service that will return a list of "related suggestions."</a>  So
when the page loads, we know nothing about our top-level nodes' children.  And
while the resulting TreeView instance could grow quite large through user
interaction, we need only load a very light set of nodes to begin with.
</p>

<p>This example also shows the two label styles for childless nodes.  The first
(default) style maintains the expand/collapse icon style even when the node has
no children; the second style shows childless nodes as leaf nodes with no
expand/collapse icon.  Note: this is for dynamically loaded nodes after the
```

```
dynamic load process has produced zero children.  You can also force the leaf
node presentation for any node by setting the isLeaf property to true (this
also disables dynamic loading).</p>

</div>

<!--BEGIN SOURCE CODE FOR EXAMPLE  -->

<h3>Childless Node Style:</h3>
<dd><label for="mode0"><input type="radio" id="mode0" name="mode" value ="0"
checked />
    Expand/Collapse</label>
</dd>
<dd><label for="mode1"><input type="radio" id="mode1" name="mode" value ="1" />
Leaf Node</label></dd>

<div id="treeDiv1"></div>

<script type="text/javascript">

YAHOO.example.treeExample = function() {

    var tree, currentIconMode;

    function changeIconMode() {
        var newVal = parseInt(this.value);
        if (newVal != currentIconMode) {
            currentIconMode = newVal;
        }
        buildTree();
    }

        function loadNodeData(node, fnLoadComplete)  {

            //We'll load node data based on what we get back when we
            //use Connection Manager topass the text label of the
            //expanding node to the Yahoo!
            //Search "related suggestions" API.  Here, we're at the
            //first part of the request -- we'll make the request to the
            //server.  In our success handler, we'll build our new children
            //and then return fnLoadComplete back to the tree.

            //Get the node's label and urlencode it; this is the word/s
            //on which we'll search for related words:
            var nodeLabel = encodeURI(node.label);

            //prepare URL for XHR request:
            var sUrl = "yui_2.7.0b-assets/treeview-assets/ysuggest_proxy.php?
query=" + nodeLabel;

            //prepare our callback object
            var callback = {

                //if our XHR call is successful, we want to make use
                //of the returned data and create child nodes.
                success: function(oResponse) {
                    YAHOO.log("XHR transaction was successful.", "info",
"example");
                    //YAHOO.log(oResponse.responseText);
                    var oResults = eval("(" + oResponse.responseText + ")");
                    if((oResults.ResultSet.Result) &&
(oResults.ResultSet.Result.length)) {
                        //Result is an array if more than one result, string
otherwise
                        if(YAHOO.lang.isArray(oResults.ResultSet.Result)) {
                            for (var i=0, j=oResults.ResultSet.Result.length; i<j;
i++) {
                                var tempNode = new
YAHOO.widget.TextNode(oResults.ResultSet.Result[i], node, false);
                            }
                        } else {
                            //there is only one result; comes as string:
                            var tempNode = new
YAHOO.widget.TextNode(oResults.ResultSet.Result, node, false)
                        }
                    }

                    //When we're done creating child nodes, we execute the node's
                    //loadComplete callback method which comes in via the argument
                    //in the response object (we could also access it at
node.loadComplete,
                    //if necessary):
                    oResponse.argument.fnLoadComplete();
                },

                //if our XHR call is not successful, we want to
                //fire the TreeView callback and let the Tree
                //proceed with its business.
                failure: function(oResponse) {
                    YAHOO.log("Failed to process XHR transaction.", "info",
"example");
                    oResponse.argument.fnLoadComplete();
                },

                //our handlers for the XHR response will need the same
                //argument information we got to loadNodeData, so
                //we'll pass those along:
                argument: {
                    "node": node,
                    "fnLoadComplete": fnLoadComplete
                },

                //timeout -- if more than 7 seconds go by, we'll abort
                //the transaction and assume there are no children:
                timeout: 7000
            };

            //With our callback object ready, it's now time to
            //make our XHR call using Connection Manager's
            //asyncRequest method:
            YAHOO.util.Connect.asyncRequest('GET', sUrl, callback);
        }

        function buildTree() {
           //create a new tree:
           tree = new YAHOO.widget.TreeView("treeDiv1");

           //turn dynamic loading on for entire tree:
```

```
           tree.setDynamicLoad(loadNodeData, currentIconMode);

           //get root node for tree:
           var root = tree.getRoot();

           //add child nodes for tree; our top level nodes are
           //all the states in India:
           var aStates = ["Andhra Pradesh","Arunachal
Pradesh","Assam","Bihar","Chhattisgarh","Goa","Gujarat","Haryana","Himachal
Pradesh","Jammu and Kashmir","Jharkhand","Karnataka"/* we're only using the first
half of this list, to keep the tree smaller,"Kerala","Madhya
Pradesh","Maharashtra","Manipur","Meghalaya","Mizoram","Nagaland","Orissa","Punjab"
,"Rajasthan","Sikkim","Tamil Nadu","Tripura","Uttaranchal","Uttar","Pradesh","West
Bengal"*/];

           for (var i=0, j=aStates.length; i<j; i++) {
                var tempNode = new YAHOO.widget.TextNode(aStates[i], root, false);
           }

           // Use the isLeaf property to force the leaf node presentation for a
given node.
           // This disables dynamic loading for the node.
           var tempNode = new YAHOO.widget.TextNode('This is a leaf node', root,
false);
           tempNode.isLeaf = true;

           //render tree with these toplevel nodes; all descendants of these nodes
           //will be generated as needed by the dynamic loader.
           tree.draw();
        }


    return {
        init: function() {
            YAHOO.util.Event.on(["mode0", "mode1"], "click", changeIconMode);
            var el = document.getElementById("mode1");
            if (el && el.checked) {
                currentIconMode = parseInt(el.value);
            } else {
                currentIconMode = 0;
            }

            buildTree();
        }

    }
} ();

//once the DOM has loaded, we can go ahead and set up our tree:
YAHOO.util.Event.onDOMReady(YAHOO.example.treeExample.init,
YAHOO.example.treeExample,true)

</script>


<!--END SOURCE CODE FOR EXAMPLE  -->

</body>
</html>
```