

IT452 Advanced Web and Internet Systems

Fall 2009

## Set 5: Web Development Toolkits

### Why Use a Toolkit?

- Choices

- Yahoo! UI Library (YUI) <http://developer.yahoo.com/yui>
- Google Web Toolkit
- Dojo
- Prototype
- ... more

## Preliminaries

```
<!-- Make auto-complete font a little nicer -->
<link type="text/css" rel="stylesheet"
      href="http://www.usna.edu/Users/cs/lmcadowel/courses/it452/yui/fonts/fonts-min.css" />

<!-- Load the YUI Loader script: -->
<script type="text/javascript"
       src="http://www.usna.edu/Users/cs/lmcadowel/courses/it452/yui/yuiloader/yuiloader-min.js"></script>

<script type="text/javascript">
    // Instantiate and configure Loader:
    var loader = new YAHOO.util.YUILoader({

        // Get YUI from local copy instead of Yahoo server
        base: "http://www.usna.edu/Users/cs/lmcadowel/courses/it452/yui/",

        // Identify the components you want to load.
        require: ["connection", "logger", "autocomplete"],

        // Configure loader to pull in optional dependencies.
        loadOptional: true,

        // The function to call when all script/css resources have been loaded
        onSuccess: function() { init(); }

        // optional: , filter: "debug" // load debug version of modules

    });

    // Tell the loader to get started
    loader.insert();
</script>

<style type="text/css">
.autocomplete { padding-bottom:2em; width:300px; }/* set width of widget here*/
</style>

<body class="yui-skin-sam" onload="init()"> // Note 'class' here: needed for logger
```

## yui1\_2.html – YUI-style AJAX

```
// Make a-synchronous call to server to get data for a new row
function handleQuery() {

    var callback = {
        success: handleSuccess,
        failure: handleFailure,
        argument: { myuser:"lkm", myquery:"cats" }
    };

    var request = YAHOO.util.Connect.asyncRequest("GET", "dummy.cgi", callback);

    YAHOO.log("Initiating request; tId: " + request.tId + ".", "info", "example");

    return false; // false prevents the form from actually submitting
}

var handleSuccess = function(o) {
    var data = o.responseText;
    var elems = data.split(",")

    // Make new row with this data
    insertRow(elems);

    window.alert ("Success! BTW, user was "+o.argument.myuser);
}

var handleFailure = function(o) {
    if(o.responseText !== undefined){
        window.alert ("HTTP status: " + o.status + " message: "+o.statusText);
    }
}
```

## yui1\_2.html – Logger

```
// Setup things once page is loaded
function init() {
    myLogReader = new YAHOO.widget.LogReader();
}
```

Don't forget the 'class' on <body>!!!!

## yui1\_3.html – Auto-complete (HTML)

```
<table> <tr>

<td> User name: </td>

<td>
<div id="my_auto_complete1" class="autocomplete">
    <input name="query_str" id="query_str" type="text" />
    <div id="query_str_box"> </div>
</div>
</td>
</td>
</tr>
</table>
```

## yui1\_3.html – Auto-complete (JS)

```
function init() {
    myLogReader = new YAHOO.widget.LogReader();

    /*
     * Setup DataSource for our auto-complete
     */

    var myDataSource = new YAHOO.util.XHRDataSource("restricted/query_autocomplete.cgi");

    myDataSource.responseType    = YAHOO.util.XHRDataSource.TYPE_TEXT;
    //myDataSource.responseType    = YAHOO.util.XHRDataSource.TYPE_XML; // alternative

    // newlines separate records, commas separate different parts of same entry
    myDataSource.responseSchema  = {
        recordDelim: "\n",
        fieldDelim: ","
    }

    myDataSource.maxCacheEntries = 10;

    /*
     * We have DataSource - now connect it to the input box
     */
    myAutoComp = new YAHOO.widget.AutoComplete('query_str','query_str_box', myDataSource);
}
```

## yui1\_3.html – Auto-complete (Fake Perl)

```
use CGI ":standard";
print "Content-Type: text/plain; charset=UTF-8\n\n";

print "tailor,extraStuff1\n";
print "this,extraStuff2\n";
print "tunic,extraStuff3\n";
print "typhoid,extraStuff4\n";
```

**What do you need to change?**

## **What else can we do?**

- DataTable
- Calendar
- TabView
- Button
- ... many more

## **How do I start?**