/JavaScript/the Yahoo UI library

One of the most important weapons in the arsenal of any front-end developer is a good JavaScript Framework. **Gavin Montague** takes us on a tour of his personal favourite

Knowledge needed	Basic HTML and JavaScript
Requires	A browser, preferably Firefox+Firebug or Safari 4
Project time	One hour

The Yahoo User Interface library powers the swishy stuff on most of Yahoo's properties including Upcoming, Delicious and Flickr. In 2006, it was publicly released under a BSD licence and remains in active development by Yahoo and the community of YUI developers.

Generally, JavaScript frameworks offer the same core features: a cross-browser interface to common functions, Ajax support and animation. YUI is no exception, but it also comes with a library of rich UI elements such as dialogs, sliders, calendars, WYSIWIG editors and DataTables as standard. Along with documentation and demos, it can be found at developer.yahoo.com/yui/.

Is speed an issue?

On disk, Prototype weighs around 50KB. YUI comes crashing in at 232KB. Yikes! However, all is not as it first seems. A lot of developers forget about minification, removing comments and white space from files, and on-the-wire Gzip compression, where the server compresses files before sending them to the browser. In this Gzipped and minified format, Prototype is a 20KB download and YUI a manageable 30KB. Much better!

Furthermore, that 30KB is for the whole of YUI and that covers a range of functionality any single page is unlikely to need. The YUI is structured as a set of 'Component' libraries that enable developers to selectively load the parts they require. I generally find myself including less than 15KB YUI Components on all but the most complicated pages.

Compared to most other Frameworks, in particular jQuery, YUI is quite verbose. The following snippet of jQuery, and its YUI equivalent, both add a class to all link tags in the document.

```
# in jQuery
$(".date").addClass("highlight");
# ..and in YUI
var dates = YAHOO.util.Dom.getElementsByClassName("date");
YAHOO.util.Dom. batch(dates, YAHOO.util.Dom.addClass, ' highlight')}
```

The first code is certainly shorter and more readable to someone who isn't familiar with the code, but we can use the trick described in 'Alias Common Functions', above right, to cut the YUI example down substantially:

```
$each($class("date"), $D.addClass, 'highlight')}
```

That's smaller, but still not as readable. However, I'd argue jQuery's very readable chaining technique comes at a greater cost, best expressed by Bradley Wright (intranation.com): "jQuery encourages developers to write JavaScript which doesn't look like JavaScript any more. This is dangerous for big projects and throws all prospects of coding standards out the window." I'm fond of jQuery, but I shy away from using it in large projects as it hides so much of the underlying language. In contrast, YUI is just well-written, well-tested JavaScript and that makes it very easy to learn and build upon.

Now to the examples. Please note that I'm pulling in the YUI Components directly from Yahoo's servers. This saves end-users from having to download the same files repeatedly.

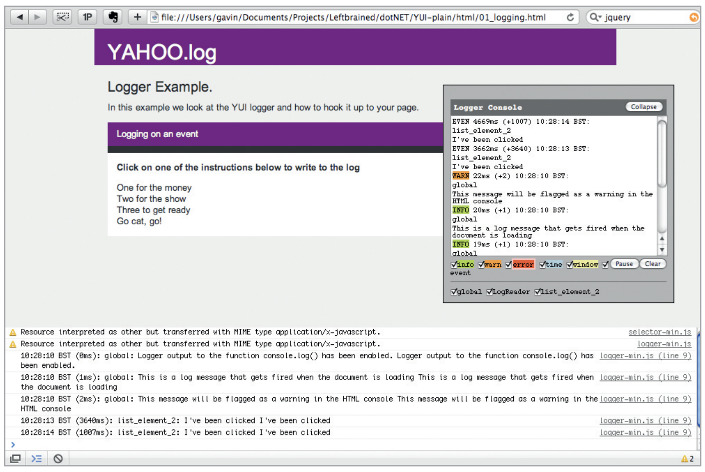
One of the most annoying tasks in vanilla JavaScript is debugging. Often it comes down to peppering **alert("Here")** or **alert("foo="+foo)** statements throughout the program to check on variables or the outcome of decisions: not very informative and very slow! YUI has a better way of observing your code: the Logger. You'll see an example in **01_logger.html** on the CD. Opening the file will display an HTML-based LogReader Widget floating over the page and automatically write a few log messages out to it. The code required to set

YUI comes with dialogs, sliders, calendars, WYSIWIG editors and DataTables

up the Logger is minimal; in the header I include the core YUI files, the logger Component and the CSS skin for the HTML logger widget, then call:

```
// First, let's set up an HTML log reader
new YAHOO.widget.LogReader();
// Now, let's write some messages to the log.
YAHOO.log("This is a log message that gets fired when the document is
loading");
```

The LogReader provides constant feedback on the messages being fired by the page and gives filtering options for cutting out chaff. As an alternative


file:///Users/gavin/Documents/Projects/Leftbrained/dotNET/YUI-plain/html/01_logging.html
jquery
YAHOO.log
Logger Example.
In this example we look at the YUI logger and how to hook it up to your page.
Logging on an event
Click on one of the instructions below to write to the log
One for the money
Two for the show
Three to get ready
Go cat, go!
Logger Console
Collapse
✓info ✓warn ✓error ✓time ✓window ✓event
Pause
Clear
✓global ✓LogReader ✓list_element_2

```
EVEN 4669ms (+1007) 10:28:14 BST:
list_element_2
I've been clicked
EVEN 3662ms (+3640) 10:28:13 BST:
list_element_2
I've been clicked
WARN 22ms (+2) 10:28:10 BST:
global
This message will be flagged as a warning in the
HTML console
INFO 20ms (+1) 10:28:10 BST:
global
This is a log message that gets fired when the
document is loading
INFO 19ms (+1) 10:28:10 BST:
global
```

```
Resource interpreted as other but transferred with MIME type application/x-javascript.   selector-min.js
Resource interpreted as other but transferred with MIME type application/x-javascript.   logger-min.js
10:28:10 BST (0ms): global: Logger output to the function console.log() has been enabled. Logger output to the function console.log() has   logger-min.js (line 9)
been enabled.
10:28:10 BST (1ms): global: This is a log message that gets fired when the document is loading This is a log message that gets fired when   logger-min.js (line 9)
the document is loading
10:28:10 BST (2ms): global: This message will be flagged as a warning in the HTML console This message will be flagged as a warning in the logger-min.js (line 9)
HTML console
10:28:13 BST (3640ms): list_element_2: I've been clicked I've been clicked   logger-min.js (line 9)
10:28:14 BST (1007ms): list_element_2: I've been clicked I've been clicked   logger-min.js (line 9)
>
```

Bust those bugs The Yahoo User Interface logger provides a non-invasive way of observing and debugging your code

Alias common functions

Save the strain on your fingers

Typing **YAHOO.util.Dom.getElementsByClassName** gets tedious but it's easy to save your fingers by using aliasing functions.

In JavaScript, functions are objects. You can declare and pass a function as you would a string, DOM element, or any other variable:

```
var adder = function(x, y) { return x+y }
function function_runner(a_function) {
  a_function(2, 2);
}
function_runner(adder); #returns 4
```

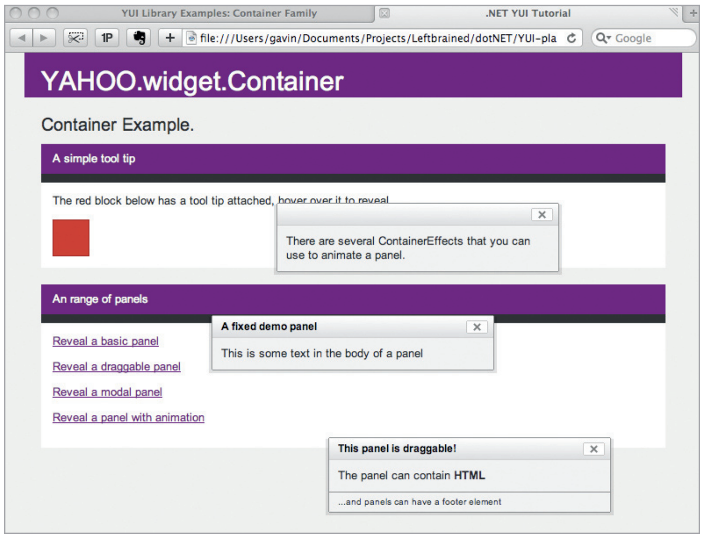
This makes for a lot of interesting code patterns in JavaScript, but one of the simplest is the aliasing of long function names:

```
var $class = YAHOO.util.Dom.getElementsByClassName;
var elements = $class("some-class");
```

I normally set up the following shortcuts in every YUI-based project.

```
var $Y = YAHOO;
var $D = $Y.util.Dom;
var $ = $D.getlElementByID
var $class = $D.getElementsByClassName
var $each = $D.batch;
var $E = $Y.util.Event
var $on = $E.on;
```

Try not to apply too many aliases; although they make code more compact they can reduce portability because anyone else calling your library of functions will need to have the same shortcuts in place.


YUI Library Examples: Container Family
.NET YUI Tutorial
file:///Users/gavin/Documents/Projects/Leftbrained/dotNET/YUI-pla
Google

YAHOO.widget.Container

Container Example.

A simple tool tip

The red block below has a tool tip attached, hover over it to reveal.

There are several ContainerEffects that you can use to animate a panel.

An range of panels

Reveal a basic panel
Reveal a draggable panel
Reveal a modal panel
Reveal a panel with animation

A fixed demo panel

This is some text in the body of a panel

This panel is draggable!

The panel can contain **HTML**

...and panels can have a footer element

Building block The Container component forms the basis of most of the rich YUI components. It can be used to build a range of floating windows, panels and tooltips

interface, the Logger can also write directly to the debuggers of some browsers including Safari 4 and the FireBug extension to Firefox.

Handling event

The various event-handling models of browsers are notoriously complicated and divergent. When are events called? How are they scoped? How best to attach events? Will they override each other? YUI deals with all of these problems and more. Open up **02_events.html** on the CD for some examples.

Attaching an event handler is easy: simply call the **YAHOO.util.Event. addListener** function, passing in the element you want to target, the type of event you're interested in and the function to be called. In the first demo we watch for clicks to the link or change events on the select box.

Note that I've passed in the ID of the element rather than a DOM reference. If I wanted to target multiple elements with the same event handler I could have passed in an array of IDs or references, as shown in demo 2.

```
// Demo 1
function first_demo_handler(e){ alert("first demo was triggered!"); }
var link = YAHOO.util.Dom.get('demo1-1');
YAHOO.util.Event.addListener(link,¬†"click",¬† first_demo_handler);
YAHOO.util.Event.addListener("demo1-2",¬†"change",¬† first_demo_handler);
// Demo 2
function second_demo_handler(e){ alert("Something in demo 2 was
clicked!"); }
YAHOO.util.Event.addListener(["demo2-1","demo2-2","demo2-3"],¬†"click",¬†
second_demo_handler);
```

The alert boxes aren't very informative: if only they told us what element had triggered the event. Regardless of the browser's event model, YUI will always scope the eventHandler function to the element that triggered the event. As a result, we can use 'this' in the third demo to reference the element that triggered the event. Or, if we pass in a fourth argument to the addListener function as shown in the final line, the event will be scoped to that argument.

```
function third_demo_handler(e){ alert("third demo was triggered by "+this. id); }
YAHOO.util.Event.addListener("demo3-1",¬†"click",¬† third_demo_handler);
YAHOO.util.Event.addListener("demo3-2",¬†"change",¬† third_demo_handler);
YAHOO.util.Event.addListener("demo3-3",¬†"click",¬† function third_demo_
handler(e){ alert(this); }, null, "A string that will be the scope for the event");
```

The YUI also takes care of stacking multiple event listeners: this enables you to watch as many events of the same, or different type on any element without fear of interfering with the other behaviours.

```
function fourth_demo_clicked_a(e){ alert("An event handler on "+this.id); }
function fourth_demo_clicked_b(e){ alert("Another event handler on"+this.id);
}
YAHOO.util.Event.addListener("demo8",¬†"click",¬† fourth_demo_
clicked_a);
YAHOO.util.Event.addListener("demo8",¬†"click",¬† fourth_demo_
clicked_b);
```

Notice I've declared the event handler before the elements appear. This would be problematic with vanilla JavaScript, but YUI cleverly deals with this by delaying the set-up of event listeners for up to 15 seconds if the element isn't immediately available. This generally removes the need for "onload" functions that exist only to set up event listeners after the page has finished loading.

The Event library provides a lot more functionality than I've covered here, but you should now be able to get started with some of the demos at developer.yahoo.com/yui/event/.

Animation

YUI allows a great deal of control over animation, as shown in **03_animation. html**. In a pattern that's common to several of its other Components, the Anim constructor takes most of its arguments in the form of a JavaScript object passed in JSON notation, which allows a large number of arguments to be passed in a very readable format.

```
YAHOO.util.Event.on('demo1', 'click', function() {
  var anim = new YAHOO.util.Anim(this, { width: { to: 200 } });
  anim.animate();
});
```

Here the second argument to the Anim constructor is actually an object expressed in JSON notation. Think of it as a package of instructions telling the animated object what properties to shift. You can issue multiple sets of instructions through this format. (Notice that I've also passed in a final parameter of '3' to make the animation run over three seconds.)

>>

```
>> YAHOO.util.Event.on('demo2', 'click', function() {
   var anim = new YAHOO.util.Anim(this, { width: { to: 200 }, height: {to:200},
opacity: {to:0.5} }, 3);
   anim.animate();
```

Our animations are looking a little flat: let's make them more interesting by 'easing' the transitions. Rather than moving everything at a constant rate, YUI allows us to apply several paces to our objects. Try swapping out the easing parameter with the following constants:

```
YAHOO.util.Event.on('demo3', 'click', function() {
   var anim = new YAHOO.util.Anim(this, { width: { to: 200 }}, 2, YAHOO.util.
Easing.bounceOut);
   anim.animate();
});
```

Try changing the Easing function for one of the other examples mentioned on the demo page to see the effects you can achieve.

The animation object created by YUI has a rich lifecycle; you can eavesdrop on the events that befall an animation and react accordingly. In the example below we watch for the first animation to complete and then fire a second animation to give a 'chained' effect. Or we could listen out for the start of an animation with onStart, or perform an action on every frame by subscribing to the onTween event. Why not try to add a Logger to this example and have it report the width of the box on every frame of its animation?

```
YAHOO.util.Event.on('demo4', 'click', function() {
   var first_anim = new YAHOO.util.Anim(this, { width: { to: 200 }}, 2, YAHOO.
util.Easing.bounceOut);
   var second_anim = new YAHOO.util.Anim(this, { height: { to: 200 }}, 2,
YAHOO.util.Easing.bounceOut);
   first_anim.onComplete.subscribe(function() { second_anim.animate(); });
   first_anim.animate();
});
```

Yahoo has created libraries of higher function UI elements that can be used to create some pretty spectacular desktop-like interfaces. We could fill an entire issue by talking about the various options of the Slider, Calendar, ColourPicker, Drag-and-Drop, ImageCropper, ImageLoader, Menu, Rich Text Editor, Slider, TabView and TreeView. Here I'll touch on the two I use most frequently:

The Container is one of the most versatile components. These elements form the basis of a wide range of content-based 'containers' ranging from simple tool-tips right up to Ajax dialogue windows. The two base classes of Container are the Module and the Overlay. You'll only ever use these if you're building your own components from the ground up. Mostly, you'll use the higher-level elements that ship with YUI and can be seen in **04_container.html**.

The tooltip is the simplest Container Component – analogous to the OS's own tooltips, which you see from hovering over elements – but still shows the common behaviours of a Container: an absolutely positioned element that can be triggered, and respond to, events. More complex is the Panel Component. This window-like pane can be floated over the rest of your page. It contains a lot of extensible behaviours, some of which you can see on the demos. Panels can be made to animate, call to server-side Ajax functions, appear as modal alerts and more: see developer.yahoo.com/yui/examples/container/index.html.

The most complex of the standard Containers is the Dialog, which integrates with the YUI Ajax libraries to send and retrieve server-side information. Because of the Security Sandbox, JavaScript operates on the Desktop so we can't demonstrate it here. A working example can be found at: developer.yahoo.com/yui/examples/container/dialog-quickstart.html. A dialog-based component we can use here is the SimpleDialog shown in demo 3. This extension to the Dialog component hides most of the configuration options and enables you to build pop-up button based forms. If you've ever felt stymied by the alert() and confirm() JavaScript dialogs, take a look at this.

DataTable

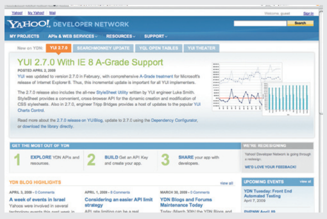
The DataTable Component is perhaps my favourite part of the YUI. In any data-driven website you'll need to display tables of formatted data. In **05_DataTable** I've mapped an HTML table to a YUI DataTable. There's a huge range of features available: column sorting, polling for remote data, editable table cells that can fire back to a server-side update, pagination, the list goes on.

You'll notice that all the columns are sortable in a meaningful fashion: dates are sorted chronologically and the balance column understands it's a floating point number. I've also made the notes column editable; try clicking and updating the values, which will then respond to being sorted. On a live site we could write these changes back to the database via Ajax.

This has only been a whistle-stop tour of some of the YUI's functionality. The place to go from here is the YUI site (developer.yahoo.com/yui/). ●

Yahoo Developer Network

Great resources for client-side development

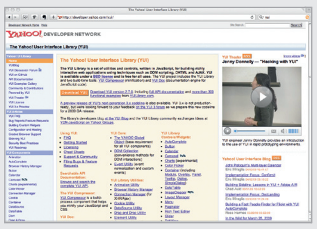


Network first What's Yahoo ever done for us? Er, quite a lot actually

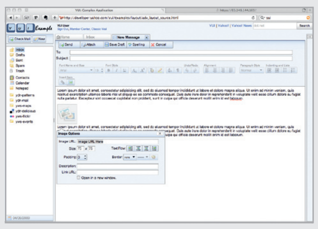
Yahoo's Developer Network (developer.yahoo.com) contains some great development resources:

- Browser Grading Guidelines: Yahoo's guide to which browsers to support (and why) is especially useful when explaining to clients why their site doesn't look the same on their mum's Windows 3.1 PC.
- The Yahoo CSS framework: Working alongside the YUI and the Grading Guidelines are Yahoo's CSS libraries, with some handy baselines for working with fonts and grids in a consistent way between browsers. In fact, the demos that accompany this tutorial are skinned with Tyler Hall's YUIapp theme (clickontyler.com/blog/2009/03/yui-app-theme/)
- YUI Compressor: A Java-based command-line tool for compressing JavaScript. There are other tools out there that do the same thing, but I've yet to find one that yields as high a return as this.
- YSlow: A plug-in for Firebug/Firefox that analyses how web pages load and offers details statistics on what might be making your page lag. You can also get hints about how best to improve user experience.
- YPatterns: What ways to people deal with pagination? What's the optimum way of letting users rate a 'thing'? Read Yahoo's findings here.

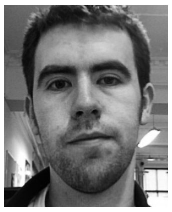
Resources Where to learn more



YUI Homepage
This site is the Bible for YUI developers. You'll find more than 300 examples from the library, video tutorials, cheat sheets and a searchable API.
developer.yahoo.com/yui



YUI examples
For a quick example of the rich interfaces that can be created with YUI, see the complex app example. A very credible email client can be built with YUI Components.
tinyurl.com/4rlfb3



About the author

Name Gavin Montague
Site www.leftbrained.co.uk
Twitter twitter.com/gavinmontague
Areas of expertise Ruby, PHP, front-end development
What was your first computer? A Sinclair ZX81. Harrier Attack remains the finest game in the world.