Achieving Performance Zen with YUI 3

Ryan Grove
YUI Team
“Every 100ms delay costs 1% of sales.”
– Greg Linden, Amazon.com
The economics of web performance are simple.
Users prefer fast websites.
Better user experience means more users.
More users means more money.
“Premature optimization is the root of all evil.”

– Lazy programmers
“Must go faster!”
– “Rock star” programmers

“Premature optimization is the root of all evil.”
– Lazy programmers
“There is no doubt that the grail of efficiency leads to abuse. Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered.”

– Donald Knuth
“We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil.”

– Donald Knuth (cont’d)
“Yet we should not pass up our opportunities in that critical 3%. A good programmer will not be lulled into complacency by such reasoning, he will be wise to look carefully at the critical code; but only after that code has been identified.”

– Donald Knuth (cont’d)

http://pplab.snu.ac.kr/courses/adv_pl05/papers/p261-knuth.pdf
Just right!
YUI 3 uses careful abstractions to provide the best possible balance between user efficiency and developer efficiency.
Effort vs. gains

Sweet spot

Low / High

High / Low
In a single, highly modular library, we provide support for **all capable browsers**, including mobile browsers.
Capability based loading allows us to deliver streamlined code to all browsers.

Mobile Safari shouldn’t have to suffer for IE6’s sins.
As with any abstraction layer, it’s important to understand what’s being abstracted.
Certain tradeoffs are made in order to increase developer efficiency.

If you find that this impacts user efficiency, you can shift the balance.
Bootstrapping
Seed bootstrap

<script src="http://yui.yahooapis.com/combo?3.2.0pr2/build/yui/yui-min.js"></script>

<script>
YUI().use('node', function (Y) {

});
</script>
Seed + Loader bootstrap

```html
<script src="http://yui.yahooapis.com/combo?3.2.0pr2/build/yui/yui-min.js&3.2.0pr2/build/loader/loader-min.js"></script>

<script>
YUI().use('node', function (Y) {
});
</script>
```
Seed + Loader bootstrap

```
<script src="http://yui.yahooapis.com/combo?3.2.0pr2/build/yui/yui-min.js&3.2.0pr2/build/loader/loader-min.js"></script>

<script>
YUI().use('node', function (Y) {
});
</script>
```
177ms (-62ms)
No bootstrap

<script src="http://yui.yahooapis.com/combo?3.2.0pr2/build/yui/yui-min.js&3.2.0pr2/build/oop/oop-min.js&3.2.0pr2/build/dom/dom-min.js&3.2.0pr2/build/event-custom/event-custom-base-min.js&3.2.0pr2/build/event/event-base-min.js&3.2.0pr2/build/pluginhost/pluginhost-min.js&3.2.0pr2/build/node/node-min.js&3.2.0pr2/build/event/event-delegate-min.js"></script>

<script>YUI({bootstrap: false}).use('*', function (Y) {
});</script>
No bootstrap
118ms (-121ms)
Build-time concatenation

```ruby
#!/usr/bin/env ruby
require 'open-uri'

ROOT = 'http://yui.yahooapis.com/combo?'
PATH = '3.2.0pr2/build/

MODULES = [
  'yui/yui-min.js',
  'oop/oop-min.js',
  'dom/dom-min.js',
  'event-custom/event-custom-base-min.js',
  'event/event-base-min.js',
  'pluginhost/pluginhost-min.js',
  'node/node-min.js',
  'event/event-delegate-min.js'
]

open(ROOT + PATH + MODULES.join("&#{PATH}")) do |yui|
  puts yui.read
end

$ ./yui-concat.rb > local-yui-min.js
```
Lazy bootstrap (use responsibly)

```html
<script src="http://yui.yahoouapis.com/combo?3.2.0pr2/build/yui/yui-min.js&3.2.0pr2/build/loader/loader-min.js"></script>

<script>
window.onload = function () {
    YUI().use('node', function (Y) {

    });
};
</script>
```
Lazy bootstrap (use responsibly)

<script src="http://yui.yahooapis.com/combo?3.2.0pr2/build/yui/yui-min.js&3.2.0pr2/build/loader/loader-min.js"></script>

<script>
window.onload = function () {
  YUI().use('node', function (Y) {

  });
};
</script>
72ms (-167ms)
Native vs. non-native
YUI uses native APIs when they’re available and emulates them when they aren’t.
Y.one('.foo');

Selectors

Y.Node → Y.Selector

Native?

emulated querySelector

No

About 1.5x as fast

native querySelector

Yes
```javascript
Y.JSON.parse(jsonString);
Y.JSON.stringify(someObject);
```

**JSON**

- **emulated JSON**
  - No
- **Native?**
  - Yes
- **native JSON**
  - Parse: about 22x as fast
  - Stringify: about 19x as fast
Transitions

```javascript
YUI().use('transition', function (Y) {
    Y.one('#demo').transition({
        duration: 1,
        easing: 'ease-out',
        height: '10px',
        width: '10px',

        // Per-property duration and easing.
        opacity: {
            value: 0,
            duration: 2,
            easing: 'ease-in'
        }
    }, function () {
        Y.log('transition finished!');
    });
});
```
YUI()

use('transition', function(Y) {
  Y.one('#demo').transition({
    duration: 1,
    easing: 'ease-out',
    height: '10px',
    width: '10px',
    // Per-property duration and easing.
    opacity: {
      value: 0,
      duration: 2,
      easing: 'ease-in'
    }
  }, function() {
    Y.log('transition finished!');
  });
});

Y.Transition

Native?

emulated transitions

Native?

native transitions

About 32% less CPU
Performance helpers
Y.cached()

Provides simple **function memoization**. Included in the YUI seed, so it’s always available.

```javascript
var factorial = Y.cached(function (n) {
    return (n === 0 || n === 1) ? 1 : n * factorial(n - 1);
});

factorial(150); // 150 function calls
factorial(150); // 1 function call
factorial(100); // 1 function call
factorial(160); // 11 function calls
```
Cache Utility

Provides a **fixed-size memory cache** for key/value pairs.

```javascript
YUI().use('cache', function (Y) {
    var cache = new Y.Cache({max: 10});

    function expensive(n) {
        var cachedResult = cache.retrieve('result' + n),
            result;

        if (cachedResult) {
            return cachedResult.response;
        }

        result = 0;

        while (result < n) {
            result += 1;
        }

        cache.add('result' + n, result);
        return result;
    }

});
```
```javascript
YUI().use('cache', function (Y) {
    var cache = new Y.Cache({max: 10});

    function expensive(n) {
        var cachedResult = cache.retrieve('result' + n),
            result;

        if (cachedResult) {
            return cachedResult.response;
        }

        result = 0;

        while (result < n) {
            result += 1;
        }

        cache.add('result' + n, result);
        return result;
    }
});
```

```
expensive(10000000); // about 360ms
expensive(10000000); // <1ms
```
CacheOffline Utility

Extends Cache to use **localStorage**.

```javascript
YUI().use('cache', function (Y) {
  var cache = new Y.CacheOffline({expires: 1800000}); // 30 minutes

  function expensive(n) {
    var cachedResult = cache.retrieve('result' + n),
        result;

    if (cachedResult) {
      return cachedResult.response;
    }

    result = 0;

    while (result < n) {
      result += 1;
    }

    cache.add('result' + n, result);
    return result;
  }

});
```
YUI().use('cache', function (Y) {
    var cache = new Y.CacheOffline({expires: 1800000}); // 30 minutes

    function expensive(n) {
        var cachedResult = cache.retrieve('result' + n),
            result;

        if (cachedResult) {
            return cachedResult.response;
        }

        result = 0;

        while (result < n) {
            result += 1;
        }

        cache.add('result' + n, result);
    return result;
    }
});
YUI().use('cache', function (Y) {
    var cache = new Y.CacheOffline({expires: 1800000}); // 30 minutes

    function expensive(n) {
        var cachedResult = cache.retrieve('result' + n), result;

        if (cachedResult) {
            return cachedResult.response;
        }

        result = 0;

        while (result < n) {
            result += 1;
        }

        cache.add('result' + n, result);
        return result;
    }
});

expensive(10000000); // about 360ms
expensive(10000000); // 1ms
```javascript
YUI().use('cache', function (Y) {
    var cache = new Y.CacheOffline({expires: 1800000}); // 30 minutes
    function expensive(n) {
        var cachedResult = cache.retrieve('result' + n), result;
        if (cachedResult) {
            return cachedResult.response;
        }
        result = 0;
        while (result < n) {
            result += 1;
        }
        cache.add('result' + n, result);
        return result;
    }
});
```

```
expensive(10000000); // about 360ms
expensive(10000000); // 1ms
New pageview
expensive(10000000); // 1ms
expensive(10000000); // 1ms
```
ImageLoader

```html
<img id="img1">
<img id="img2">
<div style="height: 1000px"></div>
<img id="img3">

<script src="http://yui.yahoapis.com/combo?3.2.0pr2/build/yui/yui.js"></script>

<script>
YUI().use('imageloader', function (Y) {
    var foldGroup = new Y.ImgLoadGroup({foldDistance: 25});

    foldGroup.registerImage({domId: 'img1', srcUrl: 'aardvark.jpg'});
    foldGroup.registerImage({domId: 'img2', srcUrl: 'bumblebee.jpg'});
    foldGroup.registerImage({domId: 'img3', srcUrl: 'cheetah.jpg'});
});
</script>
```
YUI().use('async-queue', function (Y) {
  var q = new Y.AsyncQueue(function () {
    // do something expensive
    }, function () {
      // do another expensive thing
    }, function () {
      // and another!
    });

    q.run();
});
YUI().use('async-queue', function (Y) {
    var q = new Y.AsyncQueue(function () {
        // do something expensive
    }, function () {
        // do another expensive thing
    }, function () {
        // and another!
    });

    q.run();
});
Design code with performance in mind.
Always optimize when it’s easy.
Tackle harder optimizations if it will measurably improve the user experience.
Profiling tools
YUI Profiler

```javascript
YUI().use('profiler', function (Y) {
    Y.Profiler.registerObject('Assist', Y.Search.Assist.Util);

    // ... after a bit of usage ...

    console.log(Y.Profiler.getFullReport());
});
```
WebKit Profiler
WebKit Timeline Panel
## Firebug Profiler

<table>
<thead>
<tr>
<th>Function</th>
<th>Calls</th>
<th>Percent</th>
<th>Own Time</th>
<th>Time</th>
<th>Avg</th>
<th>Min</th>
<th>Max</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>plug</td>
<td>138</td>
<td>15.65%</td>
<td>53.717ms</td>
<td>150.116ms</td>
<td>1.088ms</td>
<td>0.623ms</td>
<td>52.994ms</td>
<td>combo?...1.39.js (line 12)</td>
</tr>
<tr>
<td>byId</td>
<td>123</td>
<td>14.8%</td>
<td>50.786ms</td>
<td>50.786ms</td>
<td>0.413ms</td>
<td>0.009ms</td>
<td>41.072ms</td>
<td>combo?...1.39.js (line 6)</td>
</tr>
<tr>
<td>byId</td>
<td>2</td>
<td>6.23%</td>
<td>21.376ms</td>
<td>21.488ms</td>
<td>10.744ms</td>
<td>1.221ms</td>
<td>20.267ms</td>
<td>combo?...1.39.js (line 6)</td>
</tr>
<tr>
<td>fireExternalBeacon</td>
<td>1</td>
<td>5.8%</td>
<td>19.91ms</td>
<td>21.058ms</td>
<td>21.058ms</td>
<td>21.058ms</td>
<td>21.058ms</td>
<td>combo?...0.6.js (line 6)</td>
</tr>
<tr>
<td>h</td>
<td>3965</td>
<td>4.98%</td>
<td>17.108ms</td>
<td>17.108ms</td>
<td>0.004ms</td>
<td>0ms</td>
<td>0.074ms</td>
<td>combo?....2.4.js (line 2)</td>
</tr>
<tr>
<td>ad_embedObj</td>
<td>3</td>
<td>3.79%</td>
<td>13.014ms</td>
<td>286.5ms</td>
<td>95.5ms</td>
<td>13.818ms</td>
<td>258.379ms</td>
<td>ad_eo_1.1.js (line 5)</td>
</tr>
<tr>
<td>f</td>
<td>3</td>
<td>3.63%</td>
<td>12.453ms</td>
<td>12.921ms</td>
<td>4.307ms</td>
<td>0.152ms</td>
<td>12.113ms</td>
<td>combo?...1.39.js (line 13)</td>
</tr>
<tr>
<td>init</td>
<td>138</td>
<td>1.95%</td>
<td>6.679ms</td>
<td>13.874ms</td>
<td>0.101ms</td>
<td>0.089ms</td>
<td>0.172ms</td>
<td>combo?...1.39.js (line 11)</td>
</tr>
<tr>
<td>State</td>
<td>5385</td>
<td>1.92%</td>
<td>6.578ms</td>
<td>6.578ms</td>
<td>0.001ms</td>
<td>0ms</td>
<td>0.009ms</td>
<td>combo?...1.39.js (line 10)</td>
</tr>
<tr>
<td>byId</td>
<td>38</td>
<td>1.85%</td>
<td>6.364ms</td>
<td>6.364ms</td>
<td>0.167ms</td>
<td>0.029ms</td>
<td>4.873ms</td>
<td>combo?...1.39.js (line 10)</td>
</tr>
<tr>
<td>State</td>
<td>276</td>
<td>1.61%</td>
<td>5.518ms</td>
<td>19.548ms</td>
<td>0.071ms</td>
<td>0.018ms</td>
<td>0.182ms</td>
<td>combo?...1.39.js (line 6)</td>
</tr>
<tr>
<td>State</td>
<td>2208</td>
<td>1.43%</td>
<td>4.901ms</td>
<td>17.469ms</td>
<td>0.008ms</td>
<td>0.004ms</td>
<td>0.065ms</td>
<td>combo?...1.39.js (line 10)</td>
</tr>
<tr>
<td>State</td>
<td>139</td>
<td>1.29%</td>
<td>4.421ms</td>
<td>19.595ms</td>
<td>0.141ms</td>
<td>0.127ms</td>
<td>0.247ms</td>
<td>combo?...1.39.js (line 10)</td>
</tr>
<tr>
<td>h</td>
<td>1006</td>
<td>1.09%</td>
<td>3.746ms</td>
<td>19.595ms</td>
<td>0.141ms</td>
<td>0.127ms</td>
<td>0.247ms</td>
<td>combo?...1.39.js (line 10)</td>
</tr>
<tr>
<td>State</td>
<td>5254</td>
<td>1.08%</td>
<td>3.721ms</td>
<td>3.721ms</td>
<td>0.001ms</td>
<td>0ms</td>
<td>0.004ms</td>
<td>combo?...1.39.js (line 10)</td>
</tr>
</tbody>
</table>
**jsPerf** — JavaScript performance playground

What is jsPerf?

jsPerf aims to provide an easy way to create and share test cases, comparing the performance of different JavaScript snippets by running benchmarks. For more information, see [the FAQ](http://jsperf.com/).

### Create a test case

<table>
<thead>
<tr>
<th>Your details (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>E-mail</td>
</tr>
<tr>
<td>(won't be displayed; might be used for Gravatar)</td>
</tr>
<tr>
<td>URL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test case details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title *</td>
</tr>
</tbody>
</table>
User efficiency is most important.

Developer efficiency is important, but less so than user efficiency.

Efficiency for its own sake is least important.
Find your own zen.
Photo & illustration credits

- http://www.flickr.com/photos/formulaphoto/2625359503/
- http://www.flickr.com/photos/paulwoolrich/135550296/
- Laden Swallow by YUI’s own Allen Rabinovich
- http://www.flickr.com/photos/diphthongasaurus_rex/3530738527/
- http://www.flickr.com/photos/vickisnature/2923728081/
- http://www.flickr.com/photos/80835774@N00/2380266276/
- http://www.flickr.com/photos/wwarby/4915969081/