JavaScript Performance Patterns

@stoyanstefanov

QCon

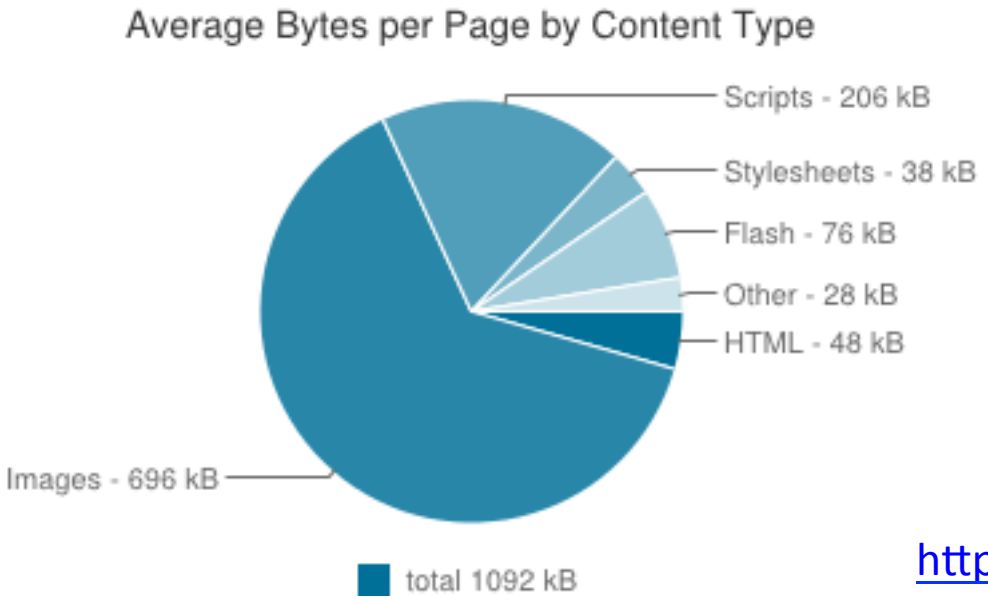
San Francisco, Nov 8, 2012

JavaScript Performance Patterns

Importance of Performance

http://bookofspeed.com

Importance of JavaScript Performance



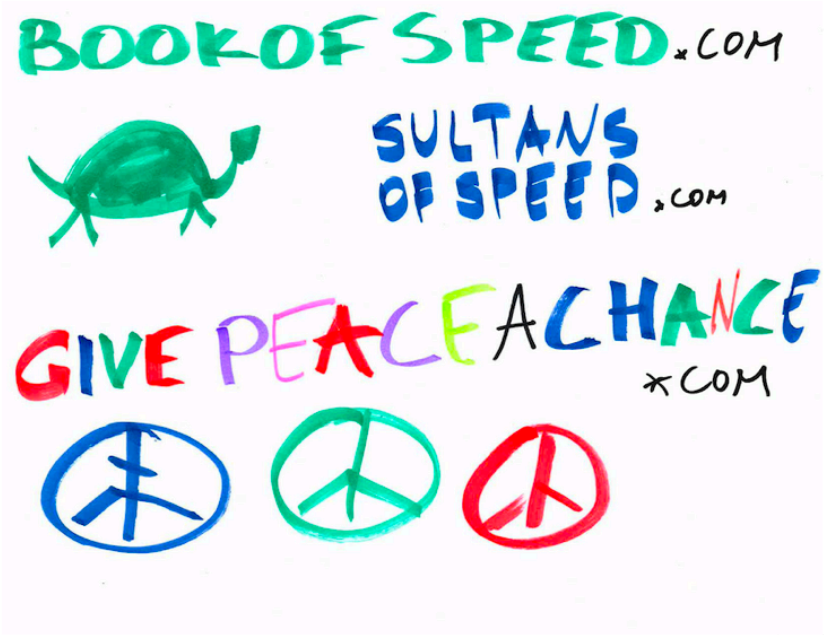
http://httparchive.org

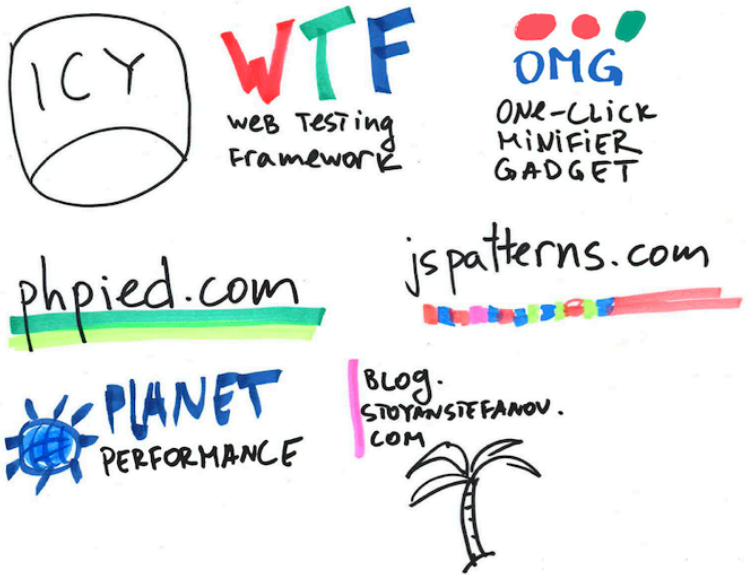
// todo

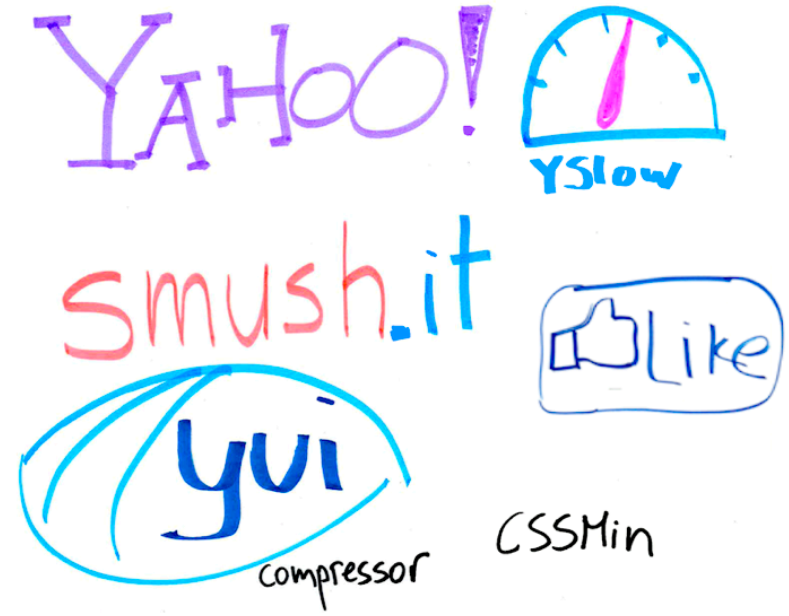
1. Loading JavaScript
2. Runtime / UI / DOM

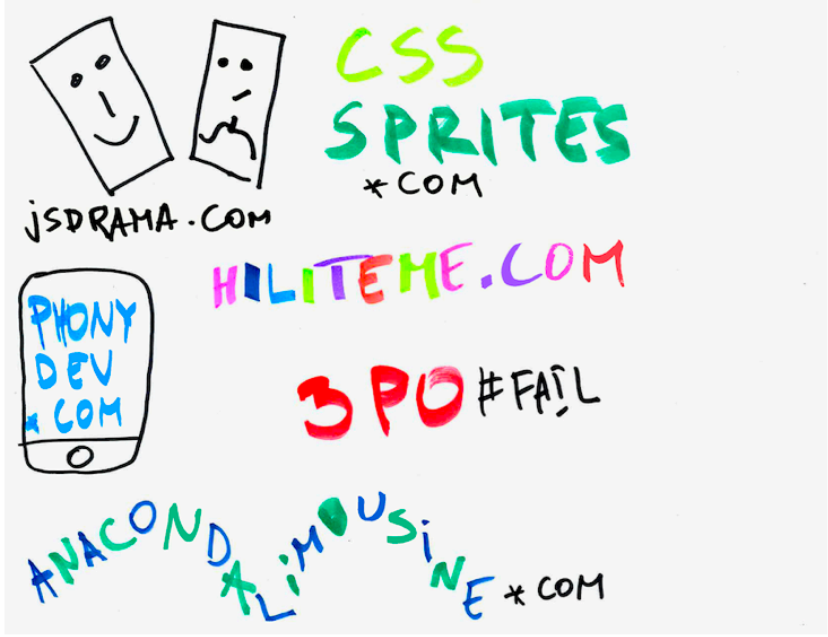
    + benchmarks

    + shims


BOOKOFSPEED.COM
SULTANS OF SPEED.COM
GIVE PEACE A CHANCE *COM


ICY
WTF
web Testing framework
OMG
ONE-CLICK MINIFIER GADGET
phpied.com
jspatterns.com
PLANET PERFORMANCE
BLOG. STOYANSTEFANOV. COM

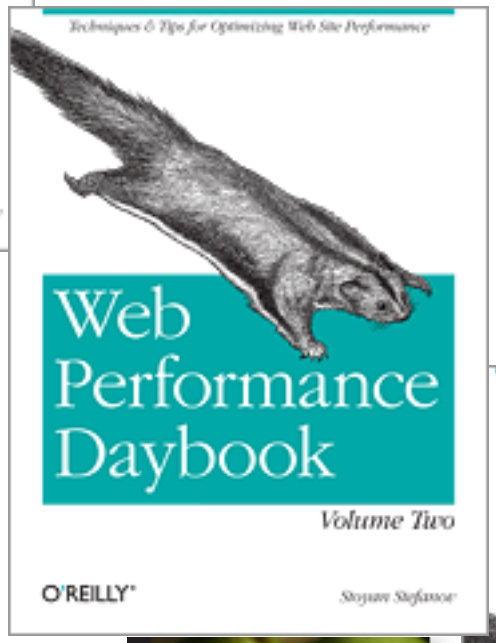

YAHOO!
YSlow
smush.it
Like
yui
compressor
CSSMin


JSDRAMA.COM
CSS SPRITES *COM
PHONY DEV *COM
HILITEME.COM
3PO #FAIL
ANACONDA LIMOUSINE *COM


Build Better Applications with Coding and Design Patterns
JavaScript Patterns
O'REILLY
YAHOO! PRESS
Stoyan Stefanov


Object-Oriented JavaScript


High Performance
JavaScript
O'REILLY
YAHOO! PRESS
Nicholas C. Zakas


Techniques & Tips for Optimizing Web Site Performance
Web Performance Daybook
Volume Two
O'REILLY
Stoyan Stefanov


Essential Knowledge for Frontend Engineers
Even Faster Web Sites
O'REILLY
Steve Souders


Early Release
JavaScript for PHP Developers
O'REILLY
Stoyan Stefanov


phpBB 2


Programming with
PEAR

Loading

First things first

- reduce # of script files
- gzip, shave 70% off
- minify, extra 40-50%
- Expires headers
- CDN

http://yslow.org
PageSpeed
http://webpagetest.org

<script src="http://…">

NOPE!

SPOF

- Single point of failure
- JS blocks

http://phpied.com/3po-fail

SPOF-O-Matic:
https://chrome.google.com/webstore/detail/plikhggfbplemddobondkeogomgoodeg

Off the critical path

Asynchronous JS

- `<script defer>`
- `<script async>`
- until then…

Dynamic script node

```
1 var js = document.createElement('script');
2 js.src = 'http://cdn.com/my.js';
3 document.getElementsByTagName('head')[0].
4           appendChild(js);
```

http://calendar.perfplanet.com/2011/the-art-and-craft-of-the-async-snippet/

But…, butt…, button?

Q: `<button onclick="…"?`

A: To hell with it

Q: Dependencies?

A: `onload` event and `js.onreadystatechange`

```
1 load('jquery.js', 'mystuff.js',
2   function () {
3     mystuff.go();
4   }
5 );
```

Unblocking `onload`

- Async JS blocks `window.onload` in !IE
- May or may not be a problem
- There's a solution: FIF

<fif>

frame-in-frame aka friendly frames
aka this Meebo thing

FIF

1) create `iframe src="js:false"`
2) in the frame `doc.write` a `<body onload …`
3) …that loads JS

FIF (snippet)

```
1  var iframe = document.createElement('iframe');
2  document.body.appendChild(iframe);
3  var doc = iframe.contentWindow.document;
4  doc.open().write('<body onload="'+
5    'var js = document.createElement(\'script\');'+
6    'js.src = \'http://example.org/js.js\';'+
7    'document.body.appendChild(js);">');
8  doc.close();
```

FIF

- unblocks `onload`, but…
- more complex
- requires JS changes

your script (before)

```
// fun with window
// and document
```

your script (before)

```
(function() {

  // fun with window
  // and document
}());
```

FIF (after)

```
(function(window) {
  var document = window.document;
  // fun with window
  // and document
}(parent.window));
```

FIF in the wild

- experimental support in FB JS SDK
- http://jsbin.com/axibow/10/edit

</fif>

Load JS but not execute

- Use cases:
  - preload in anticipation
  - lazy

Preload, then eventually execute

1. fetch the script, but don’t run it
2. run it at some point (same as async JS)

Fetching

- IE: dynamic script node, not in the DOM
- All others: CORS (XHR2)
  - your CDN should let you specify `Access-Control-Allow-Origin` header or else!

Preload, then execute

```
1  // preload
2  var js = document.createElement('script');
3  if (!js.readyState || js.readyState !== 'uninitialized') {
4    // non IE
5    var xhr = new XMLHttpRequest();
6    if ('withCredentials' in xhr) { // XHR2
7      xhr.open('GET', url, false);
8      xhr.send(null);
9    }
10 }
11 js.src = url; // IE preloads! Thanks @getify
12
13 // execute
14 document.getElementsByTagName('head')[0].appendChild(js);
```

// todo

1. ~~Loading JavaScript~~
2. Runtime / UI / DOM

   + benchmarks

   + shims

Benchmarks

- Lies, damn lies and performance advice
- Test the wrong thing
- Measure the wrong thing
- Even if not, still draw the wrong conclusions

Your first benchmark

```
var start = new Date();
// loop 100000 times
var took = new Date() - start;
```

NOPE!

Benchmark.js

- by John-David Dalton
- used in http://jsperf.com
  - calibrating the test
  - end time (ops/second)
  - statistical significance
  - margin of error

Performance Calendar

The speed geek's favorite time of the year

2011 2010 2009

http://calendar.perfplanet.com/2010/bulletproof-javascript-benchmarks/

23rd Dec 2010

Bulletproof JavaScript benchmarks

by Mathias Bynens and John-David Dalton

Writing JavaScript benchmarks isn't as simple as it seems. Even without touching the subject of potential cross-browser issues, there are a lot of pitfalls — booby traps, even — to look out for.

This is part of the reason why I created jsPerf, a simple web interface that allows you to very easily create and share test cases comparing the performance of different code snippets. There's no need to worry about anything; just enter the code you would like to benchmark and have jsPerf create a test case for you which can be run across different browsers and devices.

Behind the scenes, jsPerf was initially using a JSLitmus-based benchmarking library which I named Benchmark.js. More and more features were added, and recently, John-David Dalton rewrote the whole thing from scratch. Benchmark.js has been getting better ever since.

This article will shed some light on the various gotchas in writing *and* running JavaScript benchmarks.

Benchmarking patterns

There are a lot of ways to run benchmarks on JavaScript snippets to test their performance. The most common pattern is the following:

Pattern A

```
var totalTime,
    start = new Date,
    iterations = 6;
while (iterations--) {
```

ABOUT THE AUTHOR

Mathias Bynens (@mathias) works as a freelance web developer in Belgium. He likes HTML, CSS, JavaScript and WPO. To help with those last two things, he created jsPerf a while ago.

John-David Dalton (@jdalton): my first JavaScript project was a Super Mario Bros. game engine I made in high school. I have always been drawn to JavaScript and other ECMAScript based languages. I spend most of my

Benchmarking browsers?

No, thanks

Let's test!

String concat?

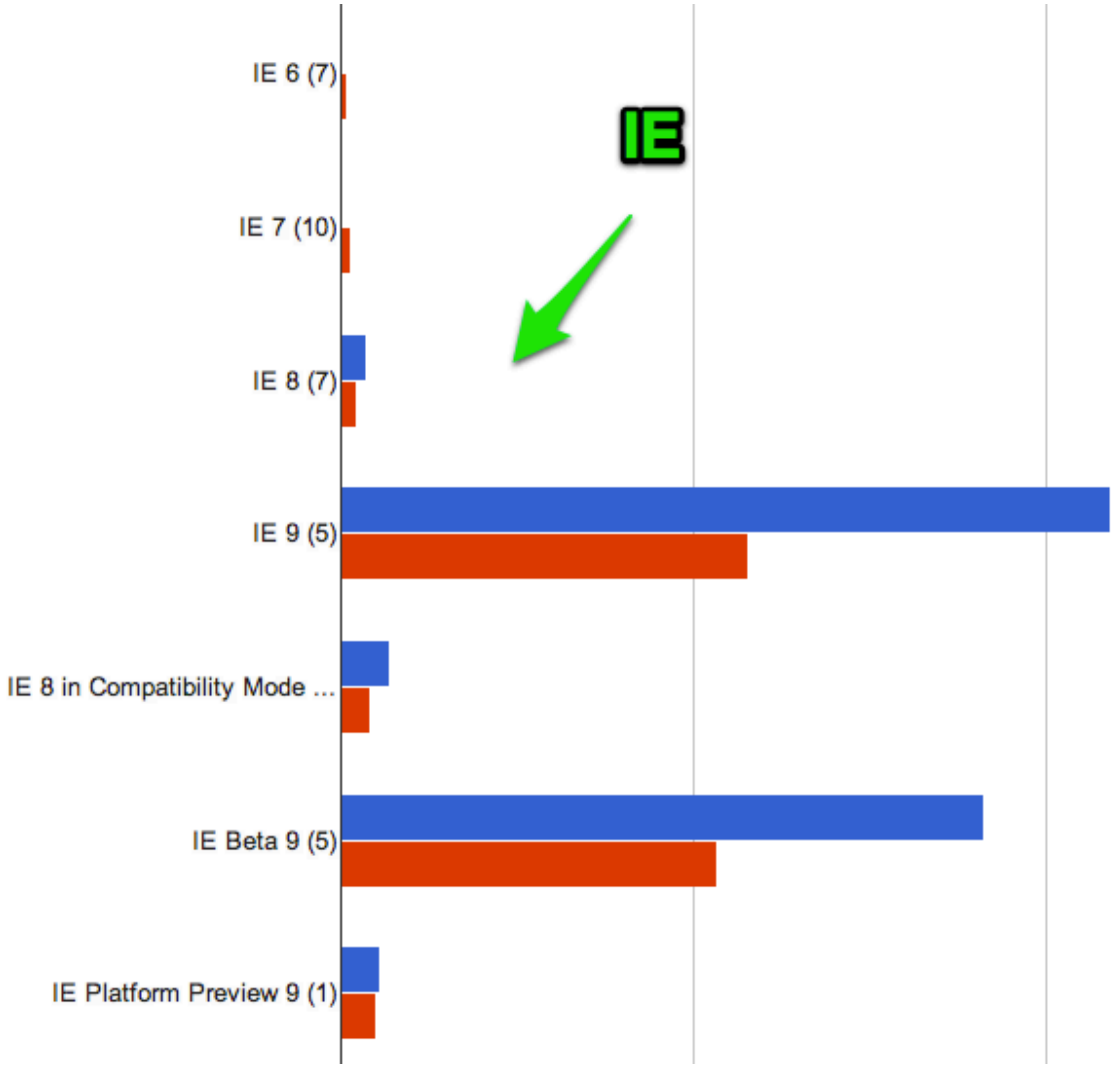
```
var text = "";
text += "moar";
```

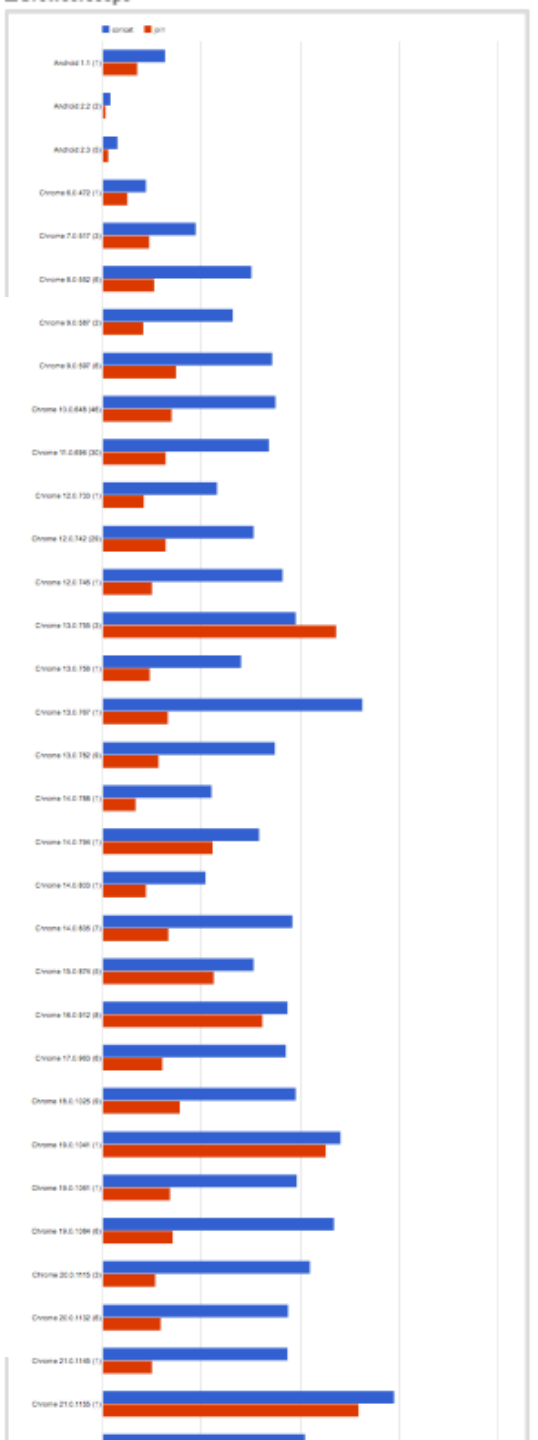
vs.

```
var parts = [];
parts.push('moar');
var text = push.join('');
```

http://jsperf.com/join-concat/

String concat





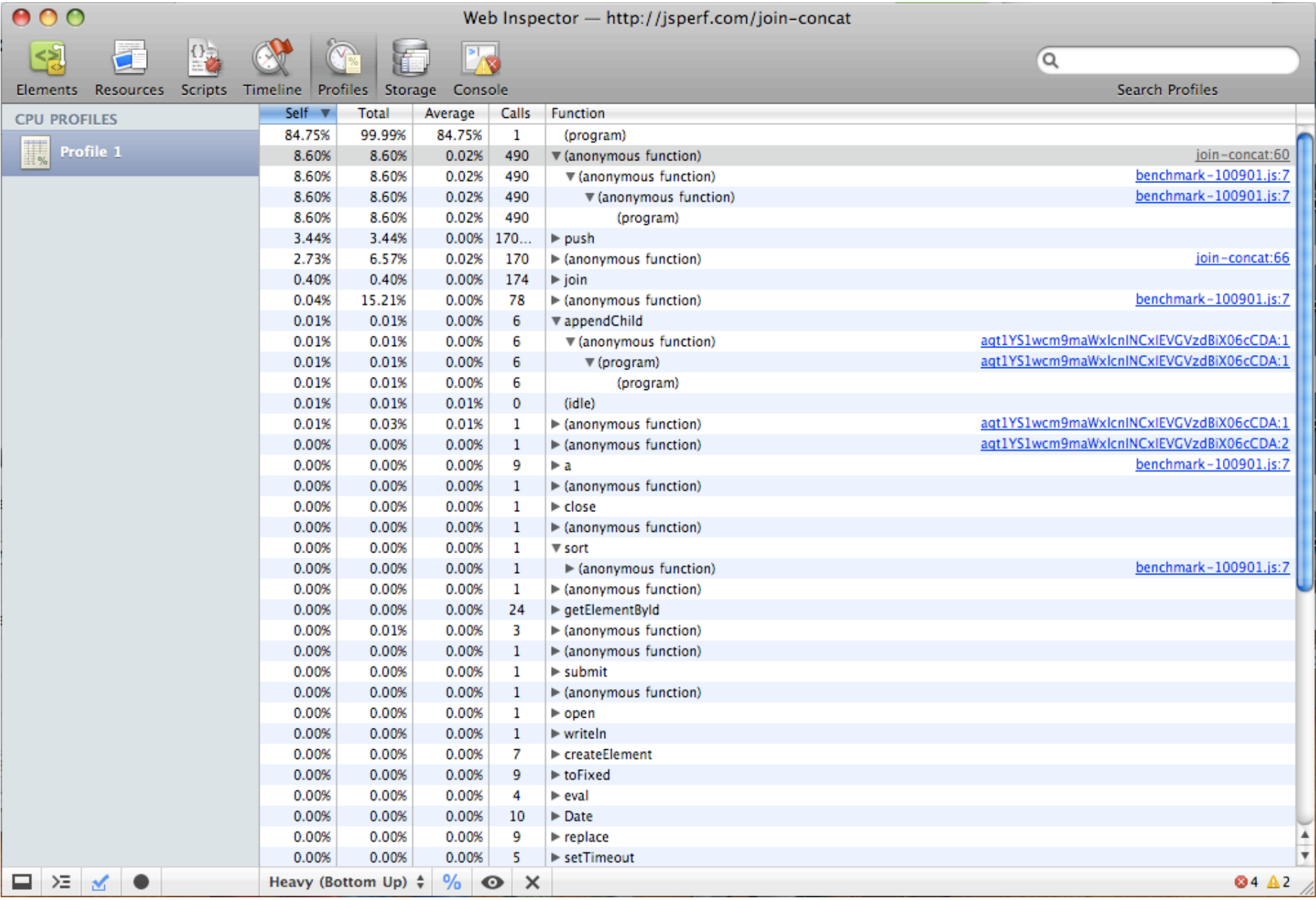
The pen is mightier than the sword! *

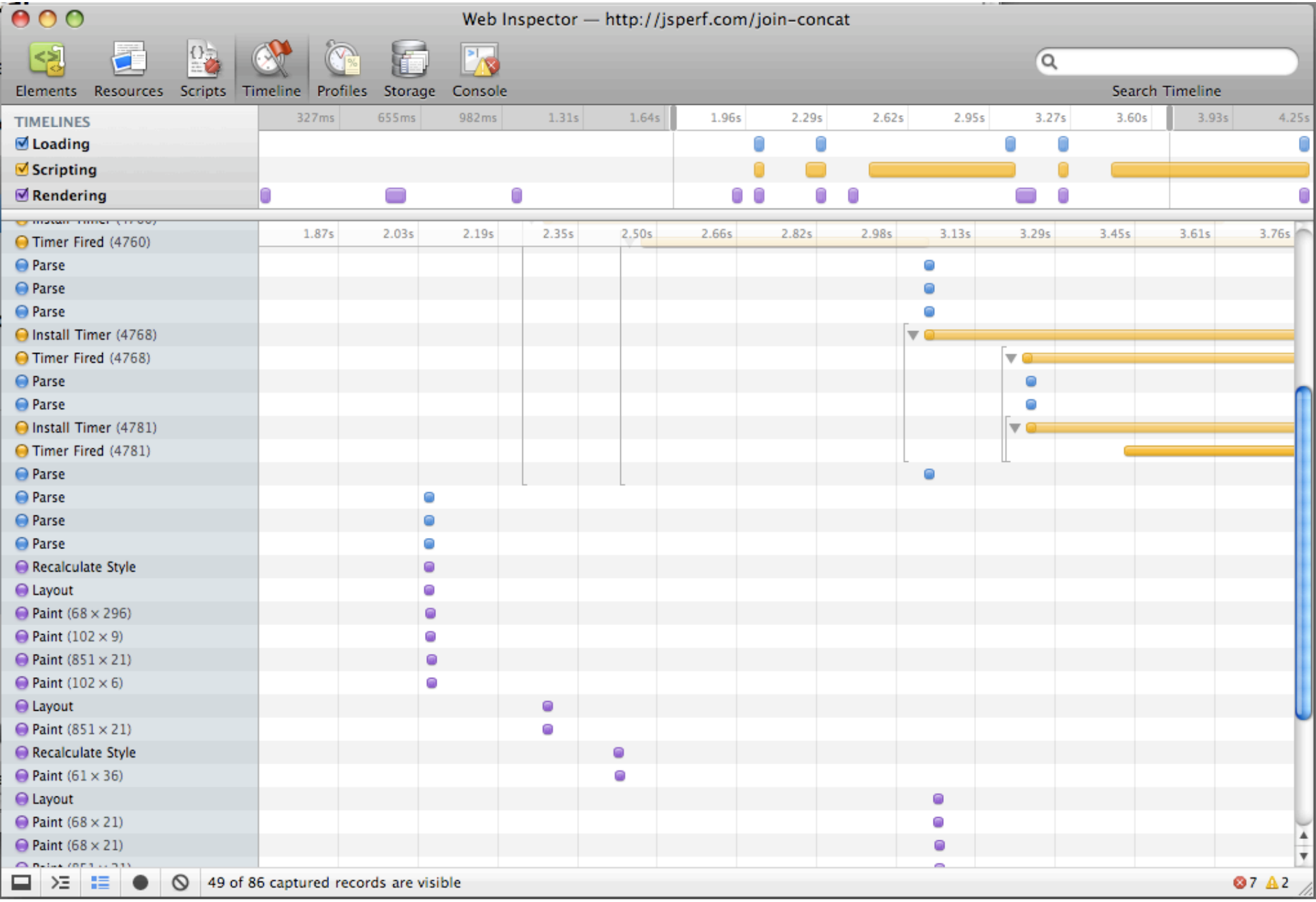
* Only if the sword is very small and the pen very sharp

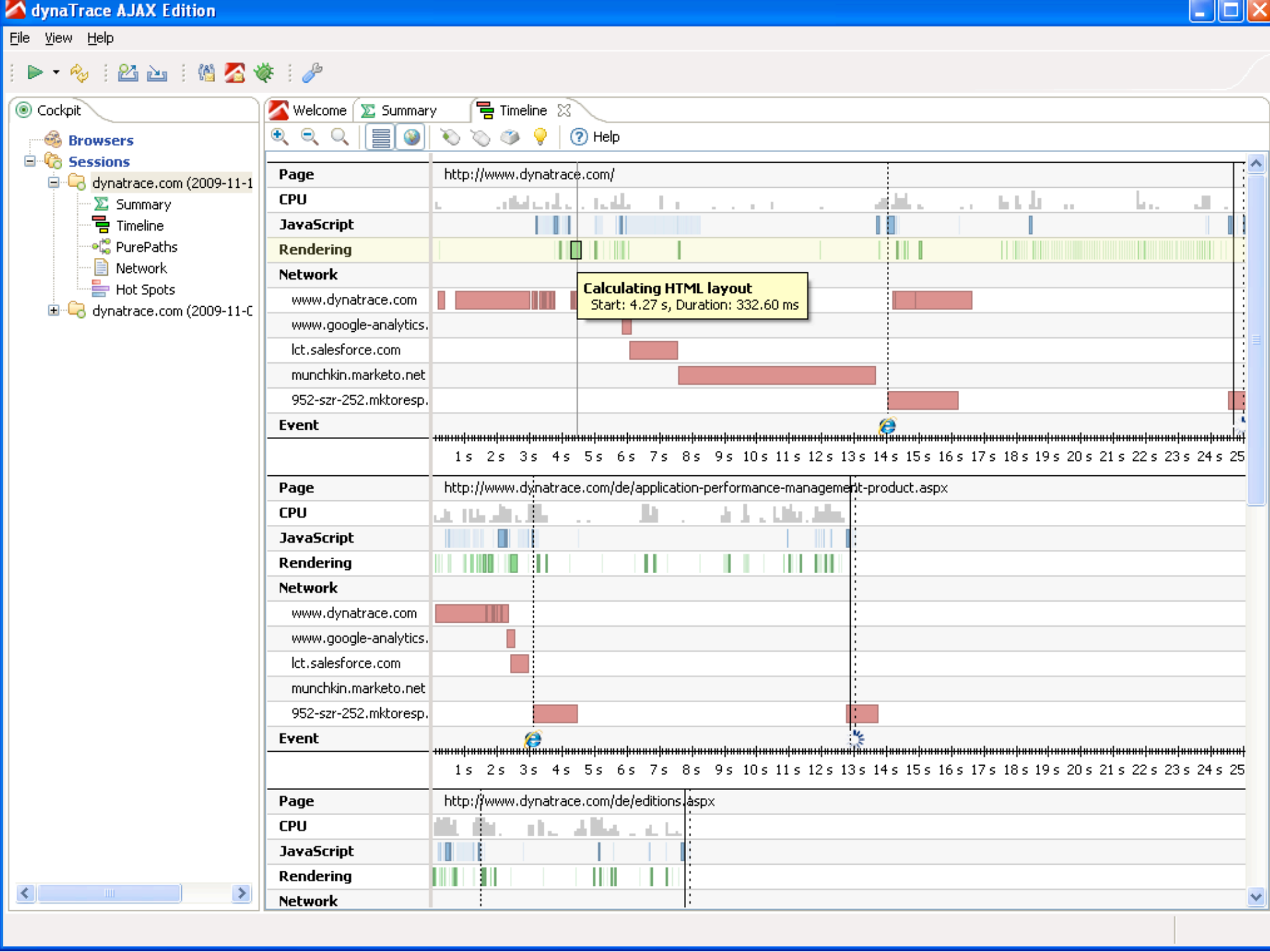
"Don't A, B is so much faster!"

You should check it again

Profiling


Web Inspector — http://jsperf.com/join-concat
Elements
Resources
Scripts
Timeline
Profiles
Storage
Console
Search Profiles
CPU PROFILES
Profile 1
Self
Total
Average
Calls
Function
84.75% 99.99% 84.75% 1 (program)
8.60% 8.60% 0.02% 490 ▼(anonymous function) join-concat:60
8.60% 8.60% 0.02% 490 ▼(anonymous function) benchmark-100901.js:7
8.60% 8.60% 0.02% 490 ▼(anonymous function) benchmark-100901.js:7
8.60% 8.60% 0.02% 490 (program)
3.44% 3.44% 0.00% 170... ▶push
2.73% 6.57% 0.02% 170 ▶(anonymous function) join-concat:66
0.40% 0.40% 0.00% 174 ▶join
0.04% 15.21% 0.00% 78 ▶(anonymous function) benchmark-100901.js:7
0.01% 0.01% 0.00% 6 ▼appendChild
0.01% 0.01% 0.00% 6 ▼(anonymous function) aqt1YS1wcm9maWxlcnINCxIEVGVzdBiX06cCDA:1
0.01% 0.01% 0.00% 6 ▼(program) aqt1YS1wcm9maWxlcnINCxIEVGVzdBiX06cCDA:1
0.01% 0.01% 0.00% 6 (program)
0.01% 0.01% 0.01% 0 (idle)
0.01% 0.03% 0.01% 1 ▶(anonymous function) aqt1YS1wcm9maWxlcnINCxIEVGVzdBiX06cCDA:1
0.00% 0.00% 0.00% 1 ▶(anonymous function) aqt1YS1wcm9maWxlcnINCxIEVGVzdBiX06cCDA:2
0.00% 0.00% 0.00% 9 ▶a benchmark-100901.js:7
0.00% 0.00% 0.00% 1 ▶(anonymous function)
0.00% 0.00% 0.00% 1 ▶close
0.00% 0.00% 0.00% 1 ▶(anonymous function)
0.00% 0.00% 0.00% 1 ▼sort
0.00% 0.00% 0.00% 1 ▶(anonymous function) benchmark-100901.js:7
0.00% 0.00% 0.00% 1 ▶(anonymous function)
0.00% 0.00% 0.00% 24 ▶getElementById
0.00% 0.01% 0.00% 3 ▶(anonymous function)
0.00% 0.00% 0.00% 1 ▶(anonymous function)
0.00% 0.00% 0.00% 1 ▶submit
0.00% 0.00% 0.00% 1 ▶(anonymous function)
0.00% 0.00% 0.00% 1 ▶open
0.00% 0.00% 0.00% 1 ▶writeln
0.00% 0.00% 0.00% 7 ▶createElement
0.00% 0.00% 0.00% 9 ▶toFixed
0.00% 0.00% 0.00% 4 ▶eval
0.00% 0.00% 0.00% 10 ▶Date
0.00% 0.00% 0.00% 9 ▶replace
0.00% 0.00% 0.00% 5 ▶setTimeout
Heavy (Bottom Up)
4 2


Web Inspector — http://jsperf.com/join-concat
Elements
Resources
Scripts
Timeline
Profiles
Storage
Console
Search Timeline
TIMELINES
327ms
655ms
982ms
1.31s
1.64s
1.96s
2.29s
2.62s
2.95s
3.27s
3.60s
3.93s
4.25s
Loading
Scripting
Rendering
1.87s
2.03s
2.19s
2.35s
2.50s
2.66s
2.82s
2.98s
3.13s
3.29s
3.45s
3.61s
3.76s
Timer Fired (4760)
Parse
Parse
Parse
Install Timer (4768)
Timer Fired (4768)
Parse
Parse
Install Timer (4781)
Timer Fired (4781)
Parse
Parse
Parse
Parse
Recalculate Style
Layout
Paint (68 × 296)
Paint (102 × 9)
Paint (851 × 21)
Paint (102 × 6)
Layout
Paint (851 × 21)
Recalculate Style
Paint (61 × 36)
Layout
Paint (68 × 21)
Paint (68 × 21)
49 of 86 captured records are visible
7
2


dynaTrace AJAX Edition
File View Help
Cockpit
Browsers
Sessions
dynatrace.com (2009-11-1
Summary
Timeline
PurePaths
Network
Hot Spots
dynatrace.com (2009-11-C
Welcome
Summary
Timeline
Help
Page
http://www.dynatrace.com/
CPU
JavaScript
Rendering
Network
www.dynatrace.com
www.google-analytics.
lct.salesforce.com
munchkin.marketo.net
952-szr-252.mktoresp.
Event
Calculating HTML layout
Start: 4.27 s, Duration: 332.60 ms
1 s 2 s 3 s 4 s 5 s 6 s 7 s 8 s 9 s 10 s 11 s 12 s 13 s 14 s 15 s 16 s 17 s 18 s 19 s 20 s 21 s 22 s 23 s 24 s 25
Page
http://www.dynatrace.com/de/application-performance-management-product.aspx
CPU
JavaScript
Rendering
Network
www.dynatrace.com
www.google-analytics.
lct.salesforce.com
munchkin.marketo.net
952-szr-252.mktoresp.
Event
1 s 2 s 3 s 4 s 5 s 6 s 7 s 8 s 9 s 10 s 11 s 12 s 13 s 14 s 15 s 16 s 17 s 18 s 19 s 20 s 21 s 22 s 23 s 24 s 25
Page
http://www.dynatrace.com/de/editions.aspx
CPU
JavaScript
Rendering
Network

Picking battles

DOM

DOM

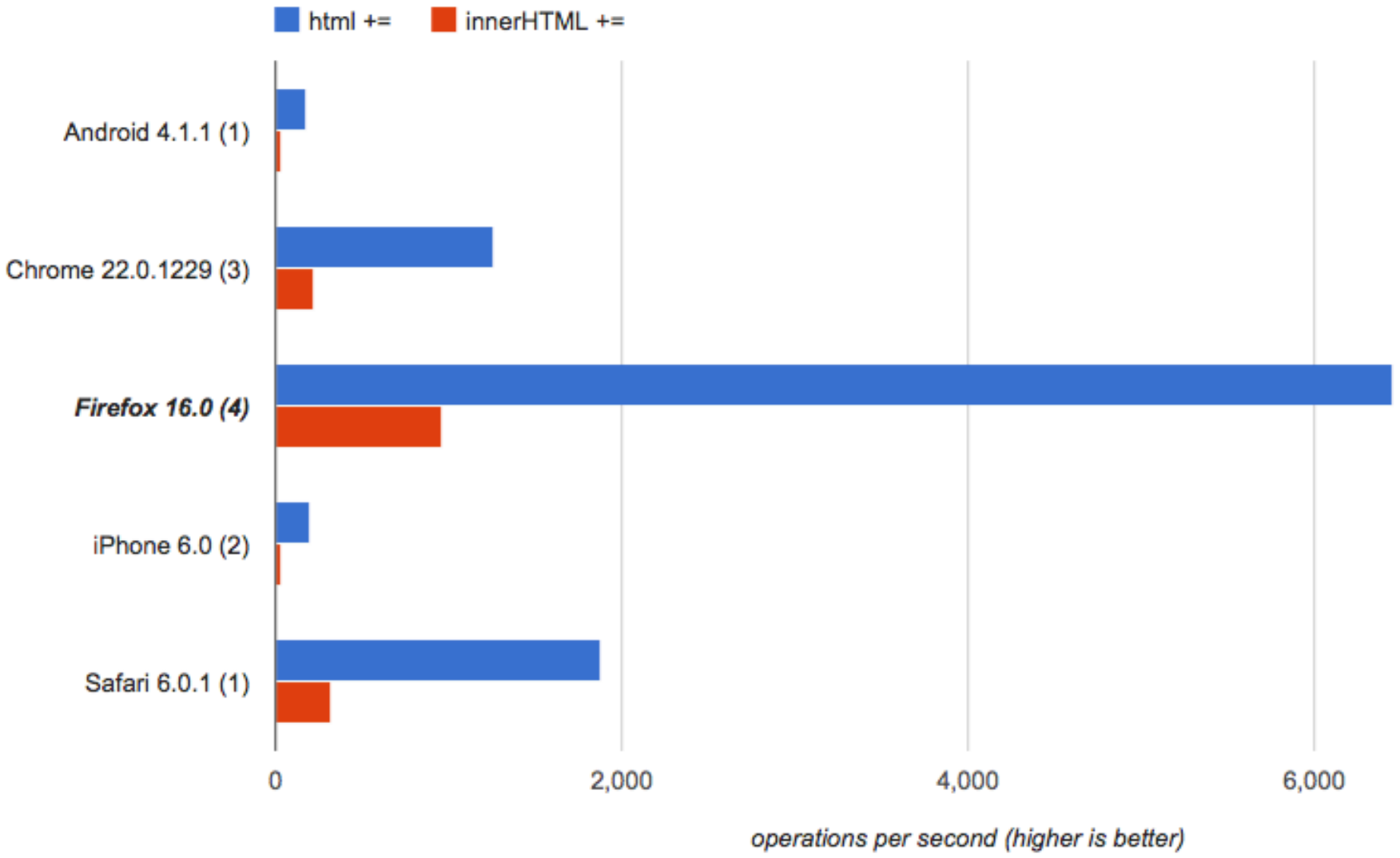
- DOM is slow
- How slow?
- http://jsperf.com/dom-touch

DOM

```
// DOM
div.innerHTML  = 'a';
div.innerHTML += 'b';

// string
var html = '';
html += 'a';
html += 'b';
div.innerHTML = html;
```

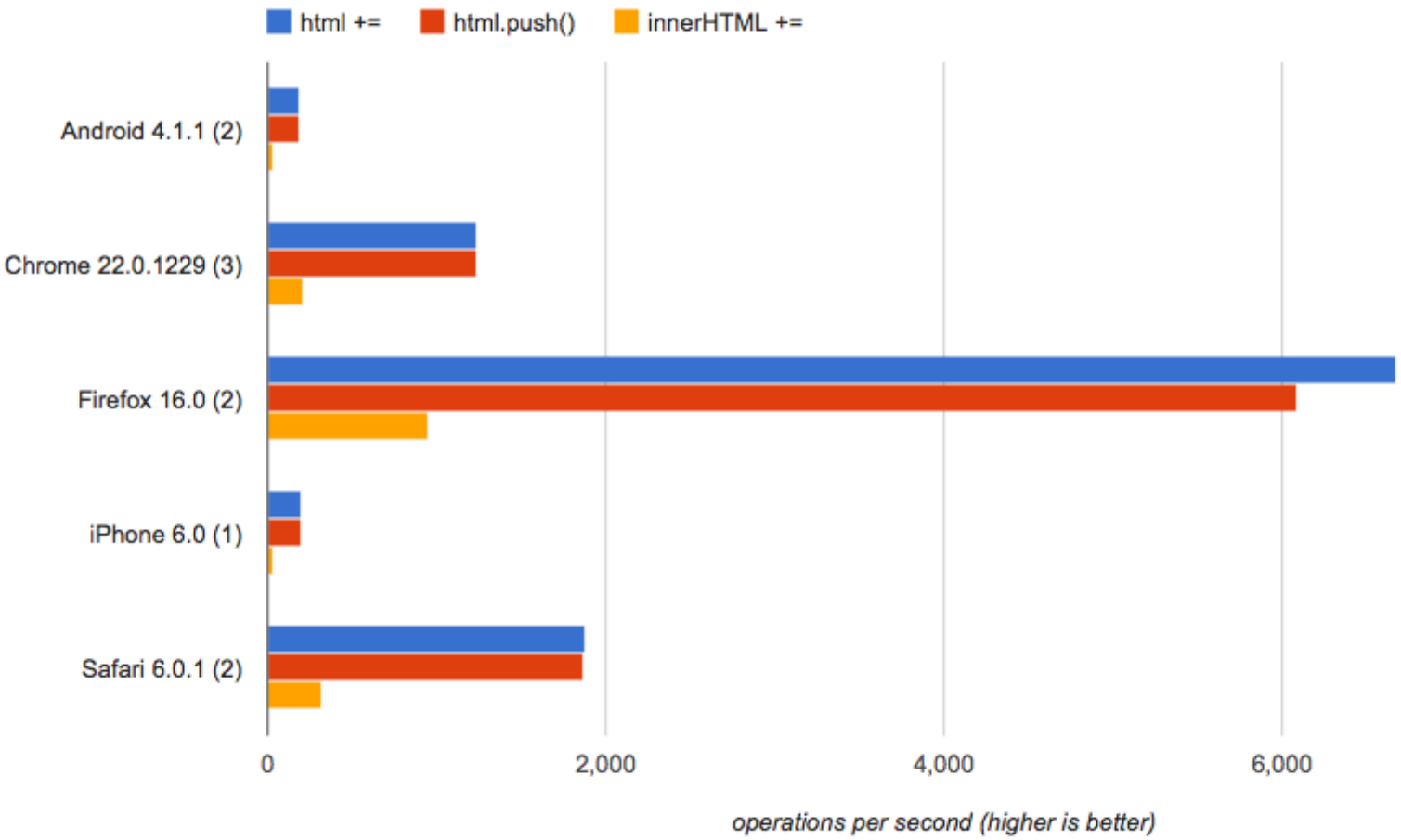
DOM


html +=
innerHTML +=
Android 4.1.1 (1)
Chrome 22.0.1229 (3)
Firefox 16.0 (4)
iPhone 6.0 (2)
Safari 6.0.1 (1)
0
2,000
4,000
6,000
operations per second (higher is better)

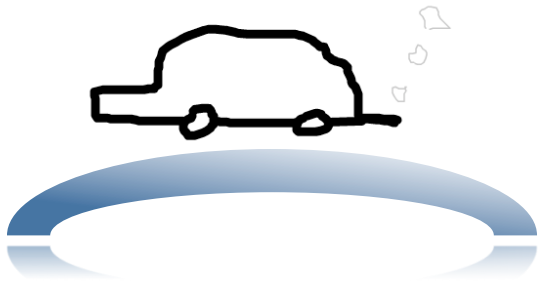
DOM + string concat

- put things in perspective

http://jsperf.com/dom-touch-concat



ECMAland

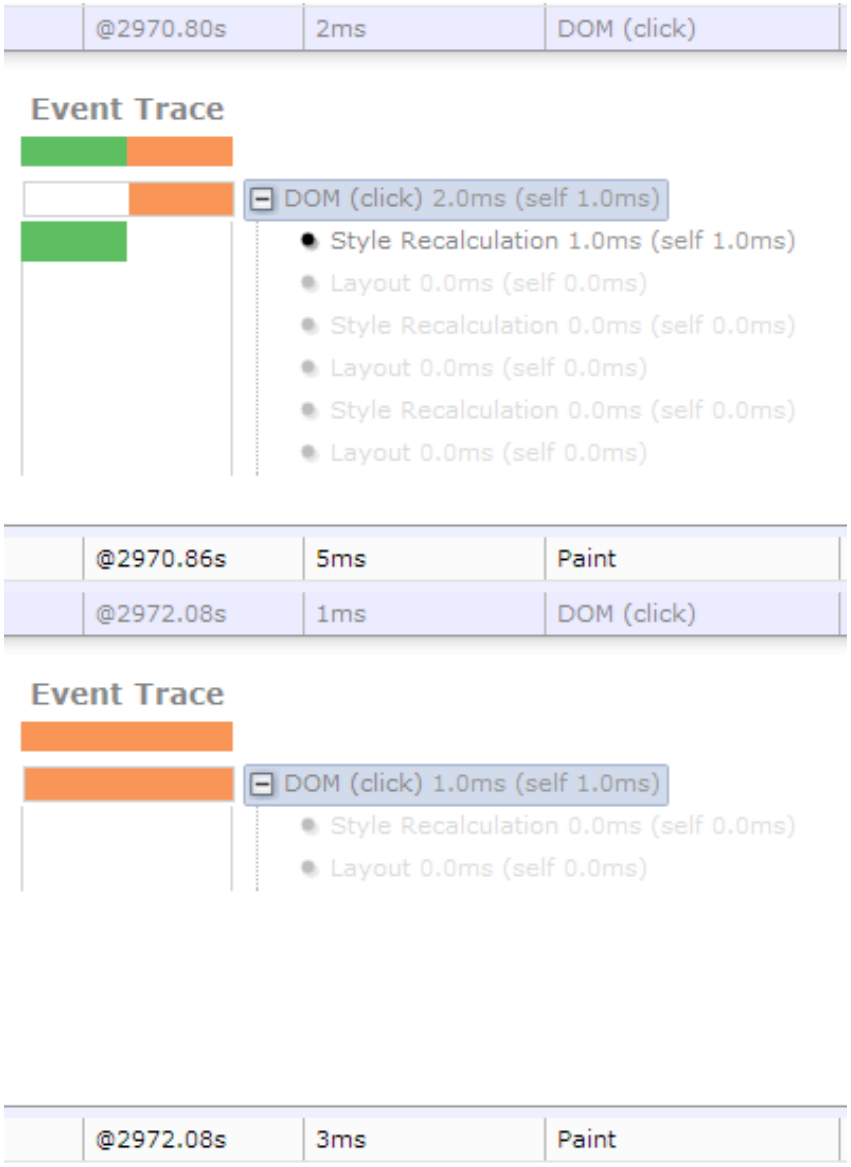


DOMland

DOM

- caching DOM references
- caching `length` in collection loops
- "offline" changes in document fragment
- batch style changes
- reducing reflows and repaints

reflows



`getComputedStyle()`, or `currentStyle` in IE

```
bodystyle.color = 'red';
tmp = computed.backgroundColor;
bodystyle.color = 'white';
tmp = computed.backgroundImage;
bodystyle.color = 'green';
tmp = computed.backgroundAttachment;
```

```
bodystyle.color = 'red';
bodystyle.color = 'white';
bodystyle.color = 'green';
tmp = computed.backgroundColor;
tmp = computed.backgroundImage;
tmp = computed.backgroundAttachment;
```

querySelectorSlow()?

```
<table border="1" id="test-table">
  <thead>
   <!-- ... -->
  </thead>
  <tbody>
    <tr class="rowme">
      <td>1</td><td>John</td><td><!-- ... -->
      <!-- ... -->
```

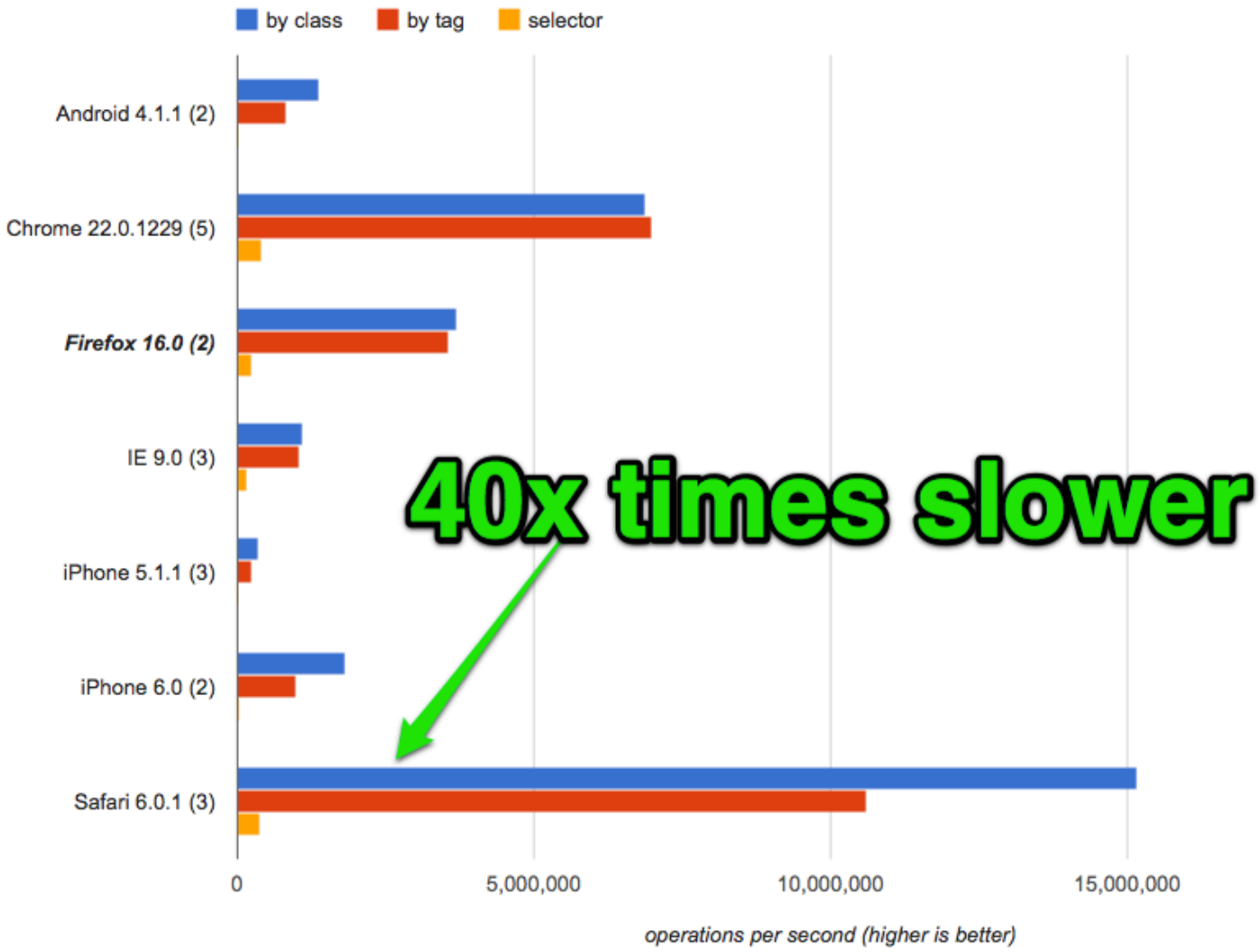
querySelectorSlow()?

```
var trs =
  tbody.getElementsByClassName('rowme');

var trs =
  tbody.getElementsByTagName('tr');

var trs =
  tbody.querySelectorAll('.rowme');
```

http://jsperf.com/queryinging/4



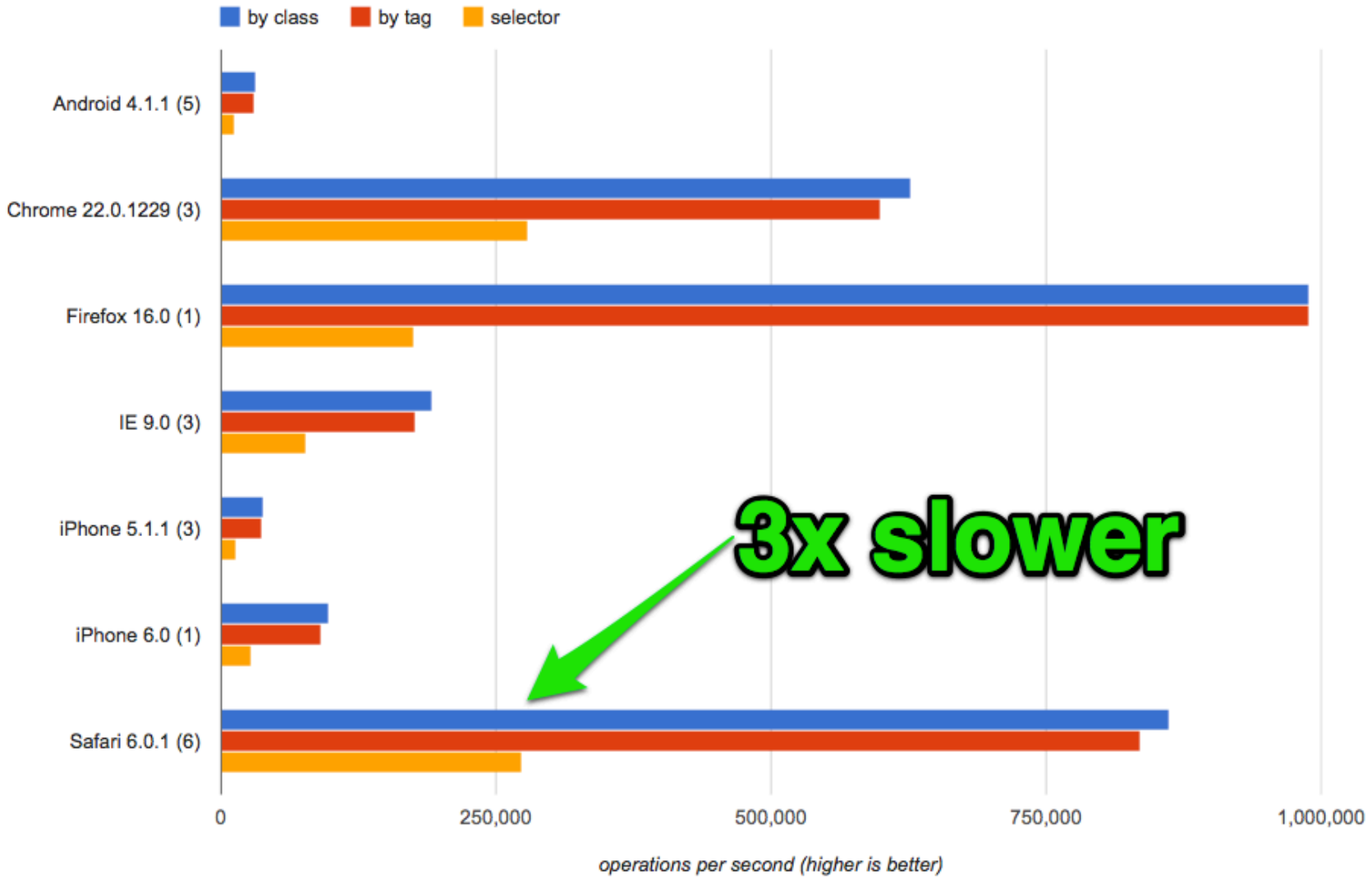
querySelectorSlow()?

```
for (
  var i = 0, len = trs.length;
  i < len;
  i += 2) {

  trs[i].className;

}
```

http://jsperf.com/queryinging/3



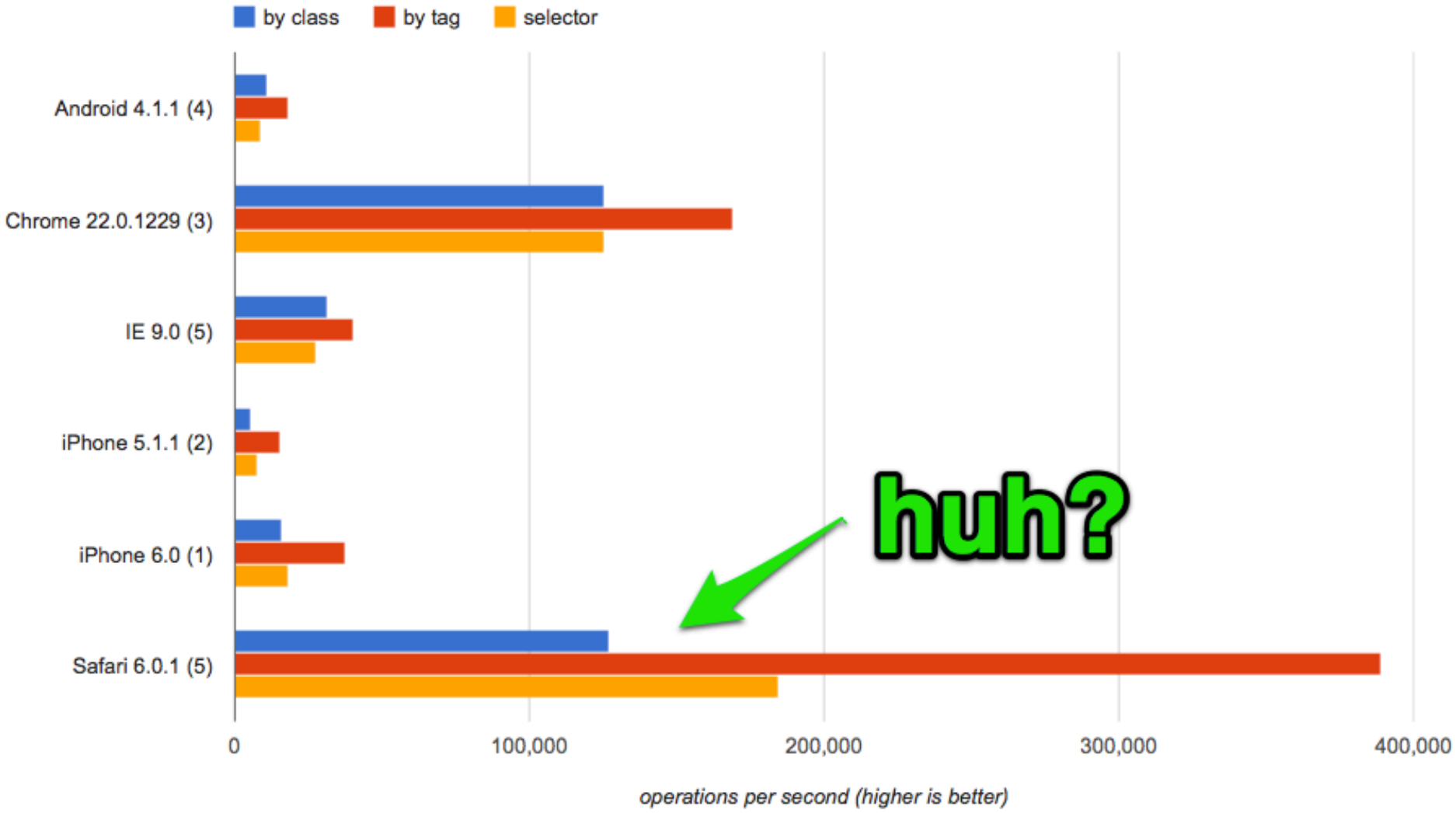
querySelectorSlow()?

```
for (
  var i = 0, len = trs.length;
  i < len;
  i += 2) {

  trs[i].className = "rowme hilite";

}
```

http://jsperf.com/queryinging/2

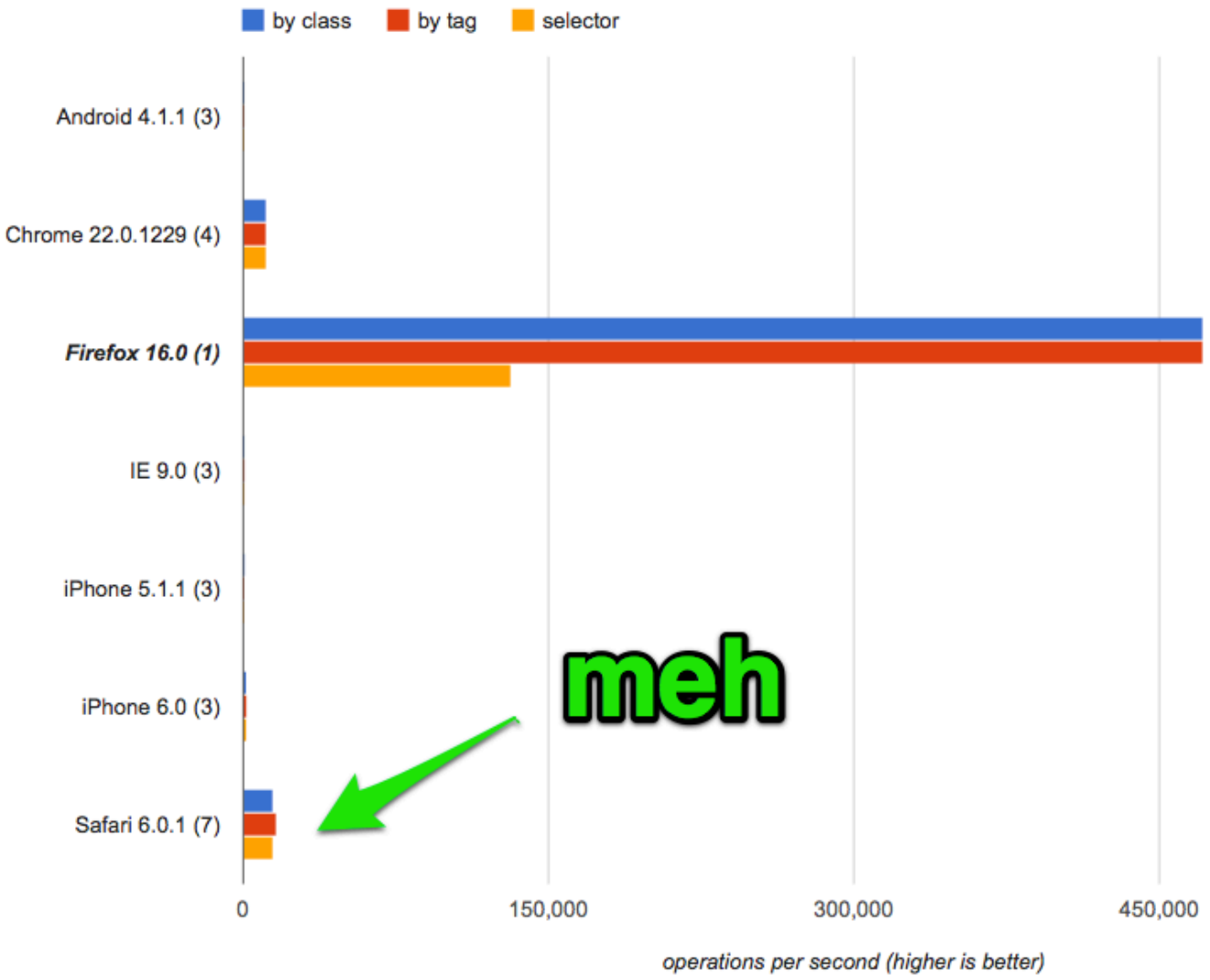


querySelectorSlow()?

```
for (
  var i = 0, len = trs.length;
  i < len;
  i += 2) {

  trs[i].className = "rowme hilite";
}
trs[0].offsetHeight;
```

http://jsperf.com/queryinging/



Priorities

1. Loading – drop everything, fix now
2. Reflows – fix asap
3. Writing DOM
4. Reading DOM
5. Querying DOM
6. ECMALand - later

data attributes

```
<div data-stuff="convenient"></div>
```

- `div.dataset.stuff`
- `div.getAttribute('data-stuff')`
- `Data.get(div).stuff // DIY`

data attributes DIY

```
1  var Data = function() {
2    var warehouse = {};
3    var count = 1;
4    return {
5      set: function (dom, data) {
6        if (!dom.__data) {
7          dom.__data = "hello" + count++;
8        }
9        warehouse[dom.__data] = data;
10     },
11     get: function(dom) {
12       return warehouse[dom.__data];
13     }
14   };
15 }();
```

data attributes

Test		Ops/sec

setAttribute	`div.setAttribute('data-yo', 'yo');`
	`div.setAttribute('data-ma', 'ma');`
	`div.setAttribute('data-la', 'la');`
	`var a = div.getAttribute('data-yo');`
	`var b = div.getAttribute('data-ma');`
	`var c = div.getAttribute('data-la');`
		1,137,453
		±0.98%
		86% slower

Data	`Data.set(div, {yo: 'yo', ma: 'ma', la: 'la'});`
	`var data = Data.get(div);`
	`var a = data.yo;`
	`var b = data.ma;`
	`var c = data.la;`
		8,089,482
		±0.35%
		fastest

dataset	`div.dataset = {yo: 'yo', ma: 'ma', la: 'la'};`
	`var data = div.dataset;`
	`var a = data.yo;`
	`var b = data.ma;`
	`var c = data.la;`
		830,578
		±0.53%
		90% slower

data attributes

http://jsperf.com/data-dataset

Shims and polyfills

Shims

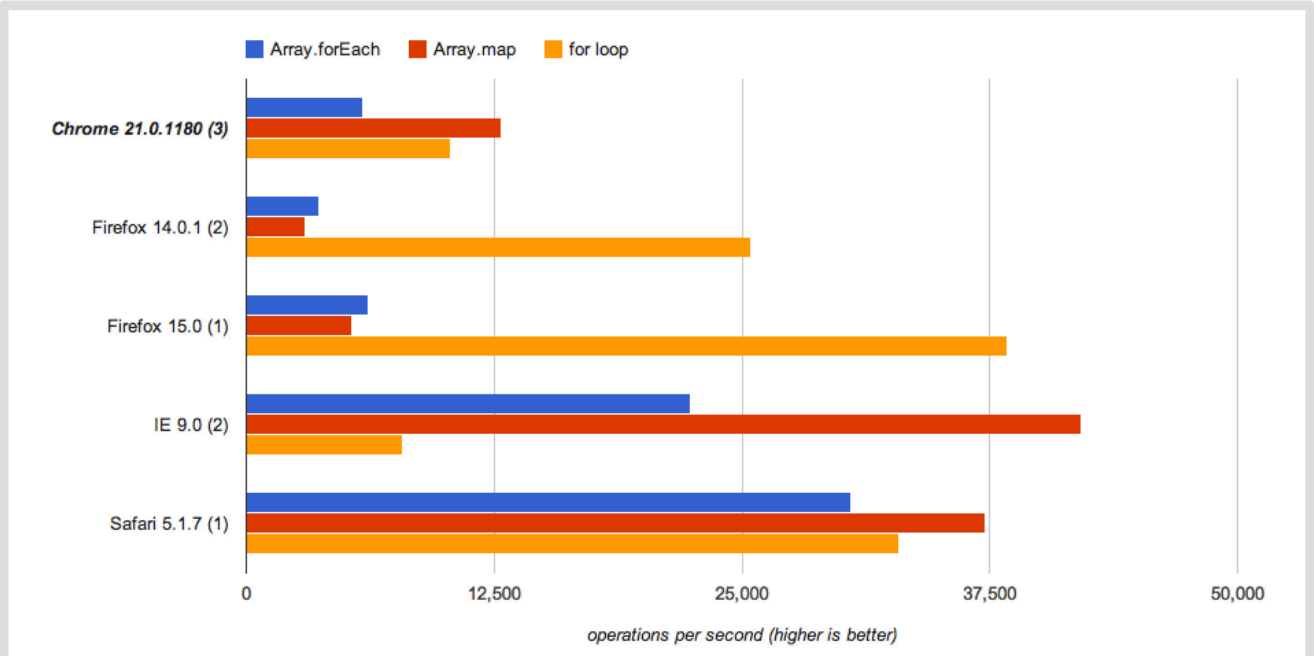
- pick the smaller/optimized one
- one that uses native where available *
- load conditionally

e.g. JSON is non-native only for 8% of users *, why load shim 100% of the time

* http://html5please.us

Fast ECMAScript5 natives?

- JDD: "browsers optimize loops because of benchmarks"
- http://jsperf.com/native-for-loop-vs-array-foreach-and-array-map-vs-lodas/2



jQuery: the most popular polyfill

- not free (perf-wise)
- do you need it?

Performance Calendar

The speed geek's favorite time of the year

2011 2010 2009

http://calendar.perfplanet.com/2011/lazy-evaluation-of-commonjs-modules/

22nd Dec 2011

Lazy evaluation of CommonJS modules

by Tobie Langel

About two years ago, the mobile Gmail team posted an article focused on reducing the startup latency of their HTML5 application. It described a technique which enabled bypassing parsing and evaluation of JavaScript until it was needed by placing it inside comments. Charles Jolley of SproutCore fame was quick to jump on the idea. He experimented with it and found that similar performance gains could be achieved by putting the code inside of a string rather then commenting it. Then, despite promises of building it into SproutCore, this technique pretty much fell into oblivion. That's a shame because it's an interesting alternative to lazy loading that suits CommonJS modules really well.

Close encounters of the text/javascript type

To understand how this technique works, let's look at what happens when the browser's parser encounters a `script` element with a valid `src` attribute. First, a request is sent to the server. Hopefully the server responds and the browser proceeds to download (and cache) the requested file. Once these steps are completed the file still needs to be parsed and evaluated.

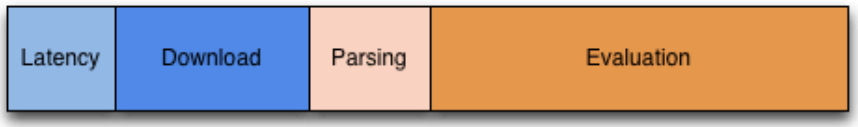
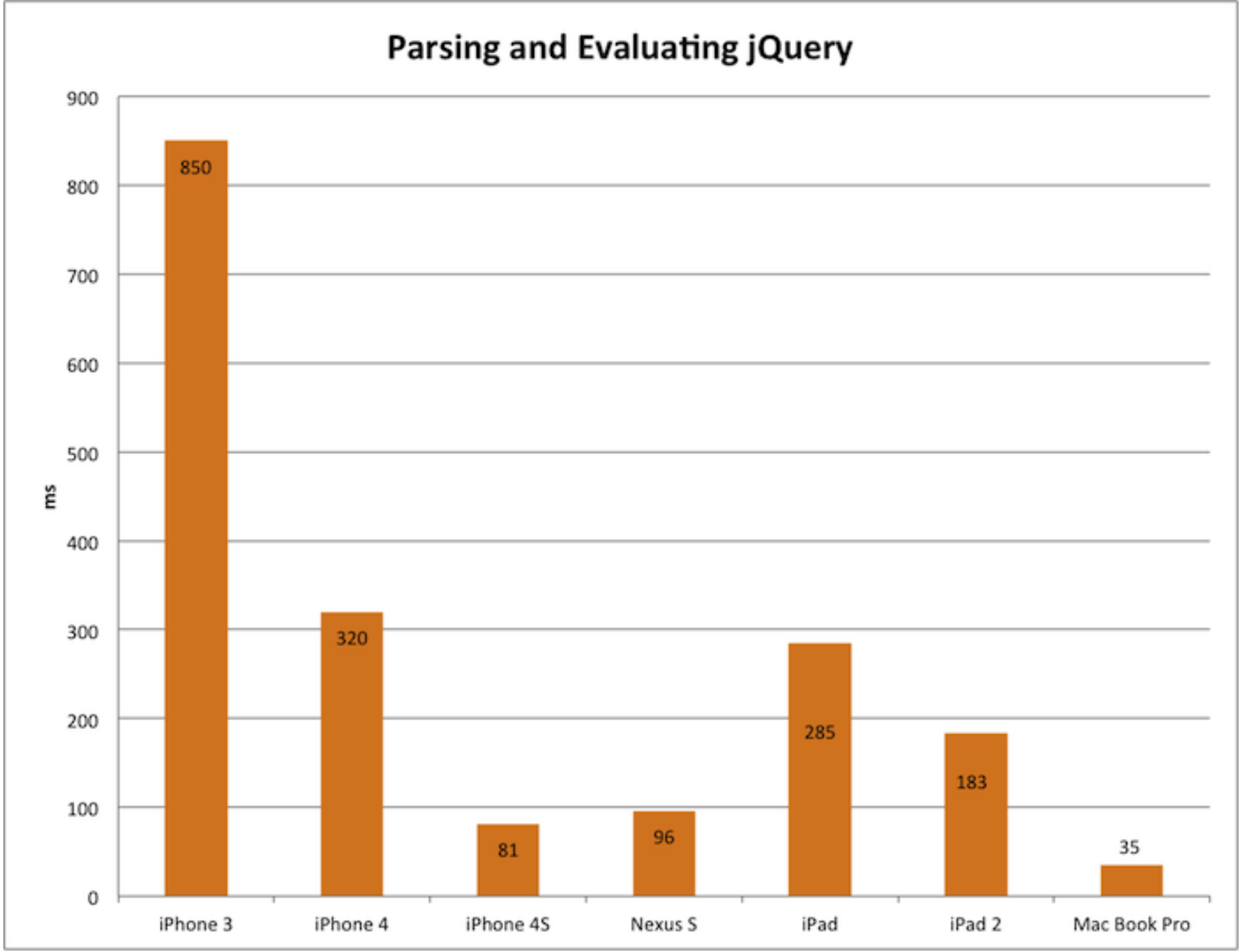


Fig. 1: uncached JavaScript resource fetching, parsing and evaluation.

ABOUT THE AUTHOR

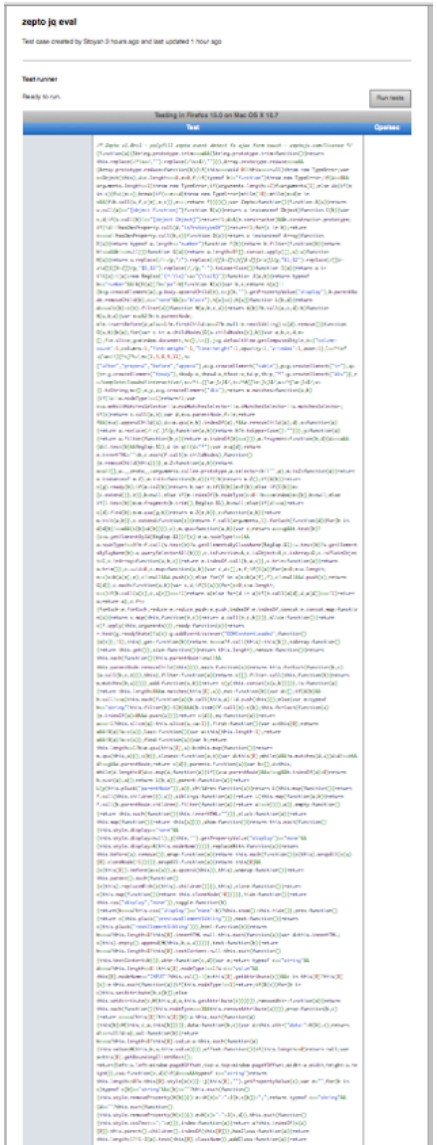
Tobie Langel (@tobie) is a Software engineer at Facebook. He's also Facebook's W3C AC Rep. An avid open-source contributor, he's mostly known for having co-maintained the Prototype JavaScript Framework. Tobie recently picked up blogging again and rants at blog.tobie.me. In a previous life, he was a professional jazz drummer.


Parsing and Evaluating jQuery
900
800
700
600
500
400
300
200
100
0
ms
850
320
81
96
285
183
35
iPhone 3
iPhone 4
iPhone 4S
Nexus S
iPad
iPad 2
Mac Book Pro

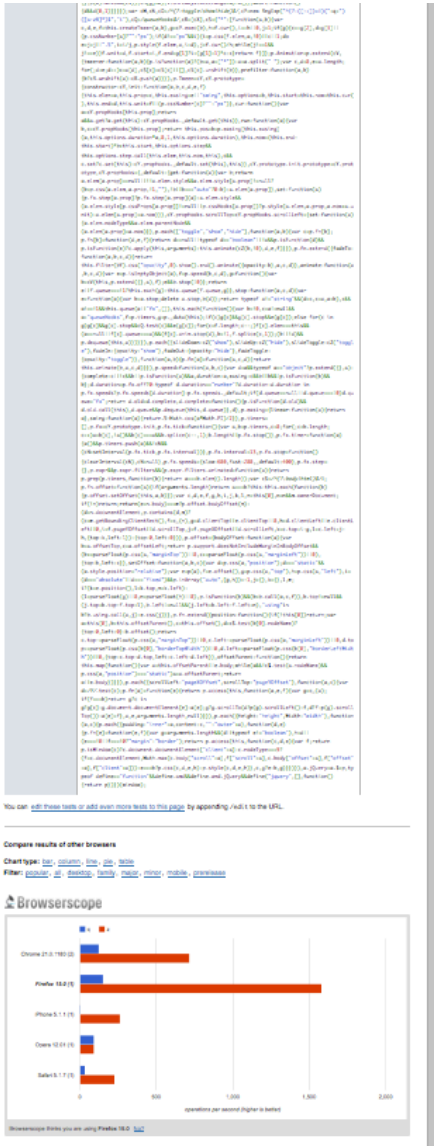
Experiment: jQuery vs. Zepto

What’s the cost of just dropping it on the page?

jsperf.com/zepto-jq-eval

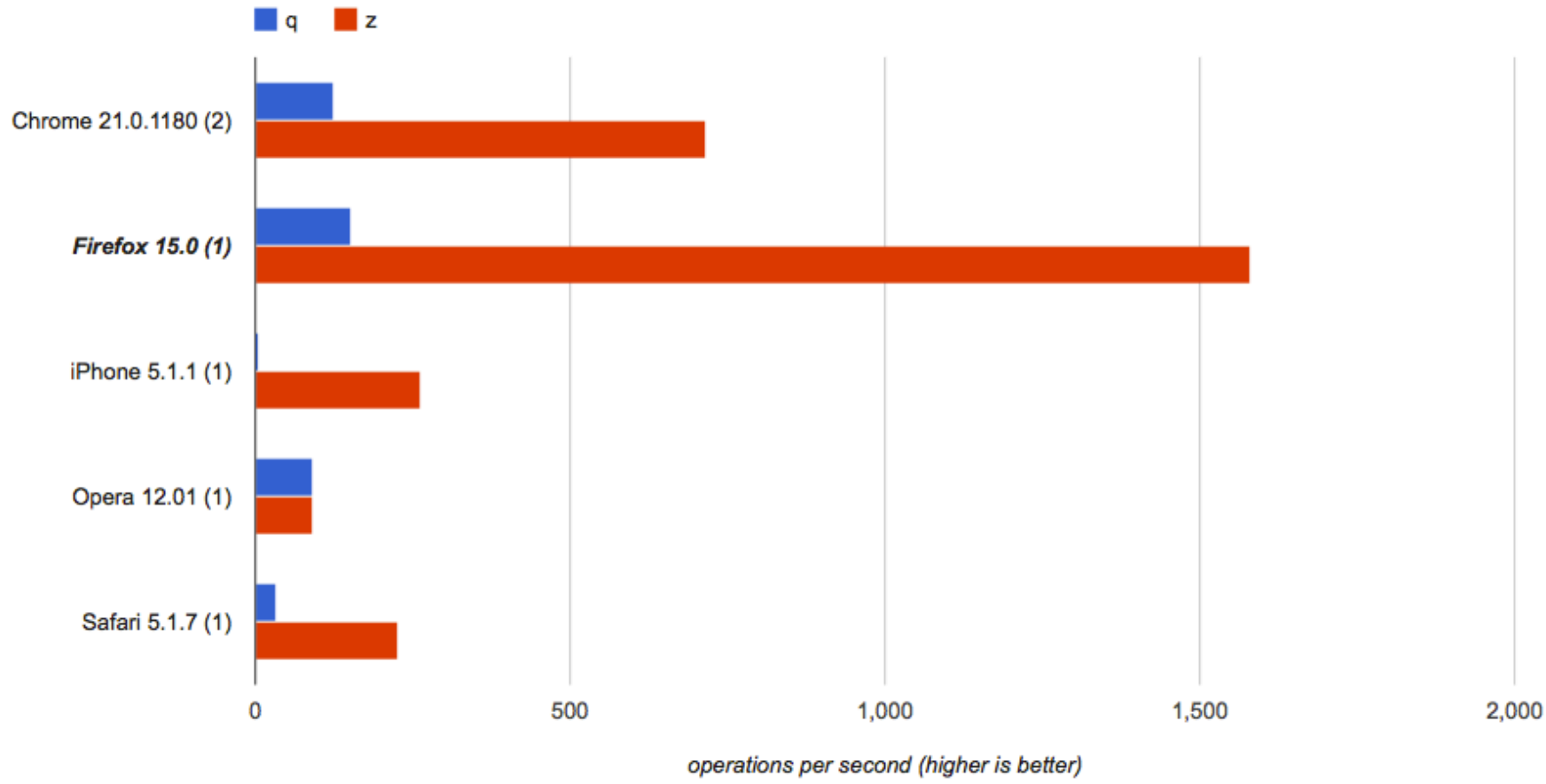


[…]



jsperf.com/zepto-jq-eval

Browserscope



Browserscope thinks you are using **Firefox 15.0** No?

jsperf.com/zepto-jq-eval

Browserscope

UserAgent	q	z	# Tests
Chrome 21.0.1180	124	716	2
Firefox 15.0	153	1,580	1
iPhone 5.1.1	4	262	1
Opera 12.01	91	91	1
Safari 5.1.7	33	227	1

Browserscope thinks you are using **Firefox 15.0** No?

z: 56 times faster on iPhone 4

In closure…

- JS off the critical path
  (async, lazy, preload)
- Practice writing jsperf.com tests
  ("jsperf URL or it didn't happen!")
- Don't touch the DOM (remember the bridge)
- Use the tools (Timeline, CPU/heap profiler, SpeedTracer, Dynatrace)
- Think of poor mobile
  (easy with the shims)

Thank you!

http://slideshare.net/stoyan/