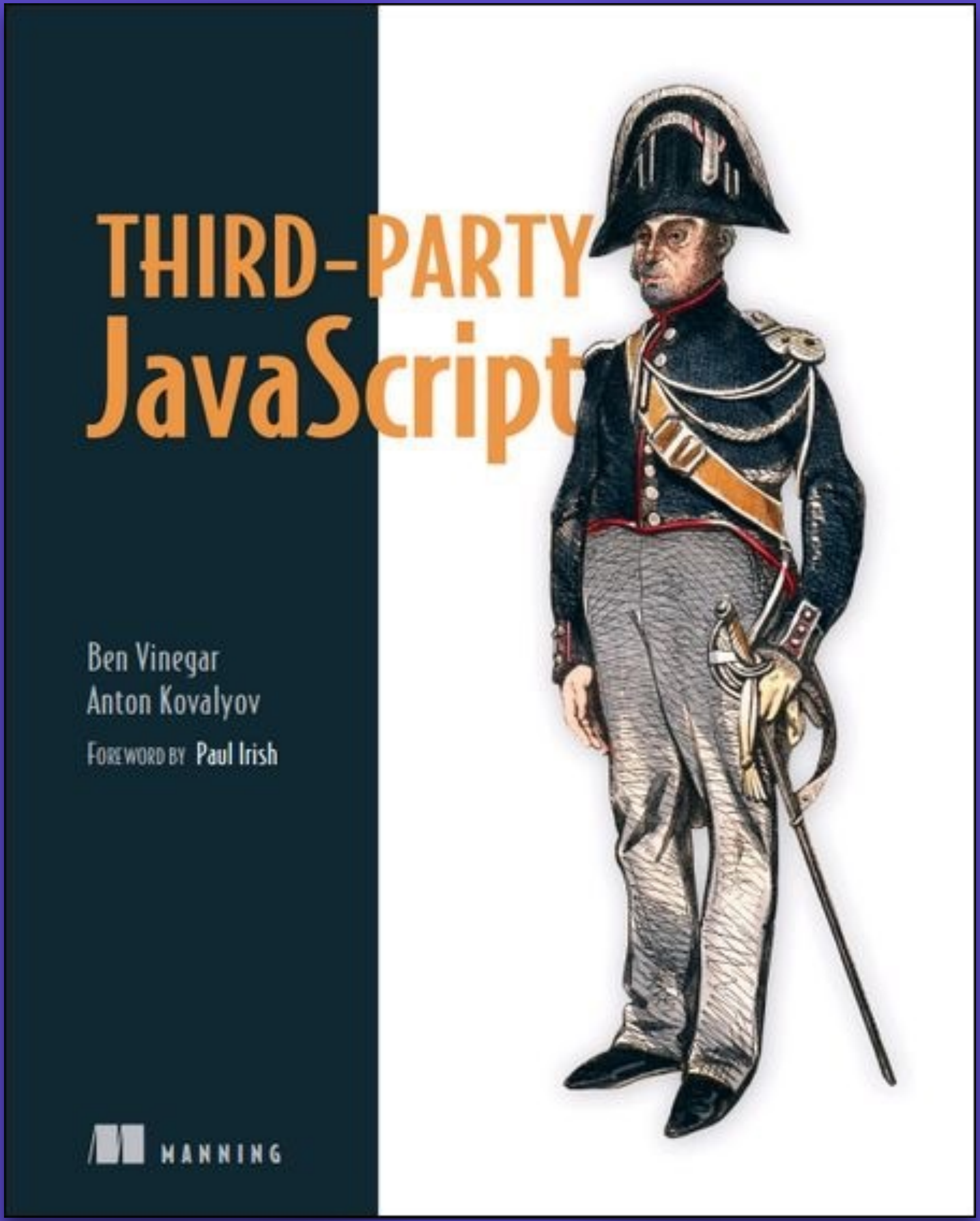
re-evaluating front-end performance best practices

@bentlegen

SHaPe



THIRD-PARTY JavaScript

Ben Vinegar
Anton Kovalyov

Foreword by Paul Irish

MANNING

car ▪ go cult

cargo cult
web performance

how does it happen?

deprecated …

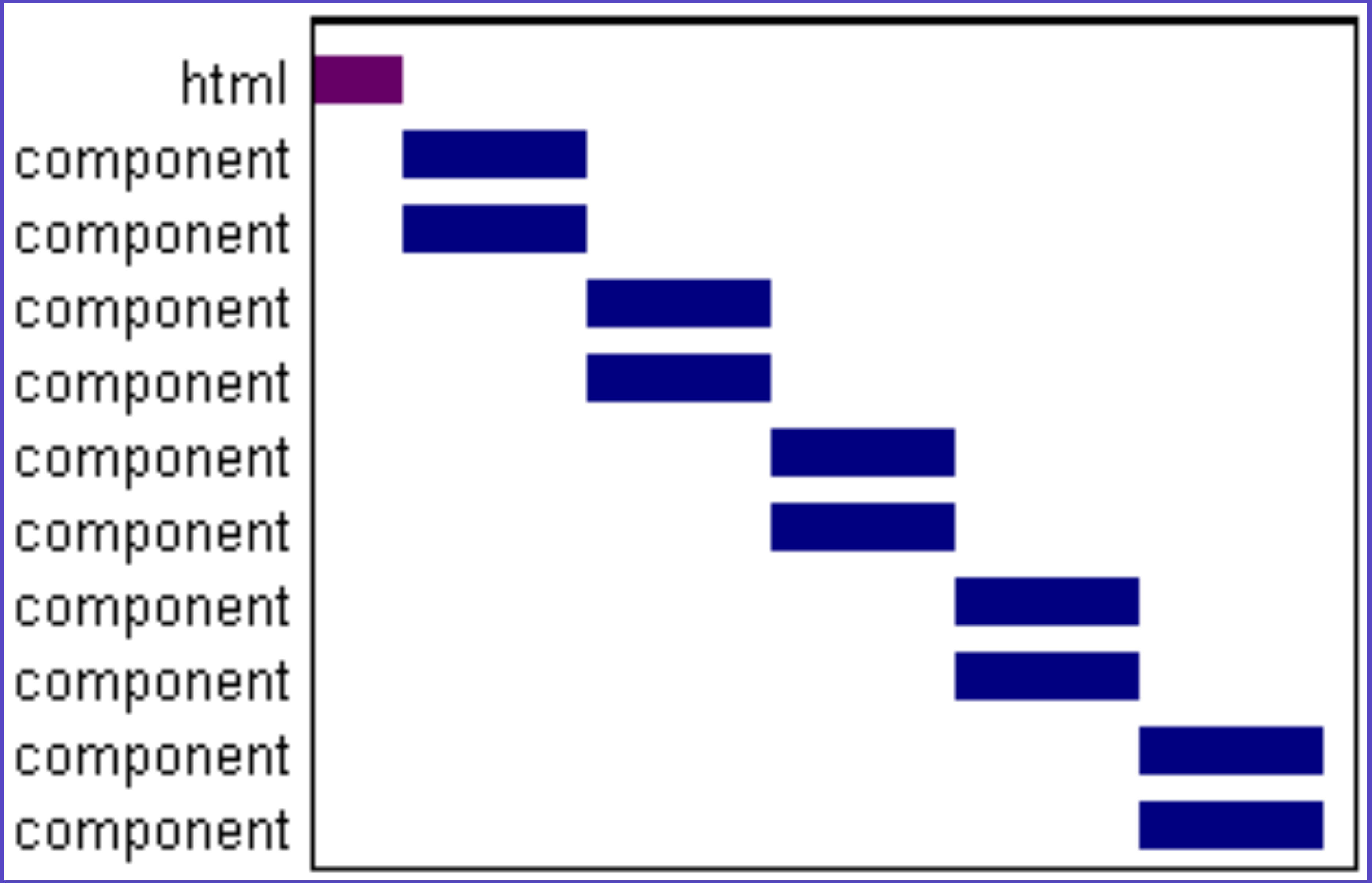
- books
- blog posts
- best practice guides
- StackOverflow answers
- performance analysis tools
- practices on live websites
- conference talks like this one

the agenda

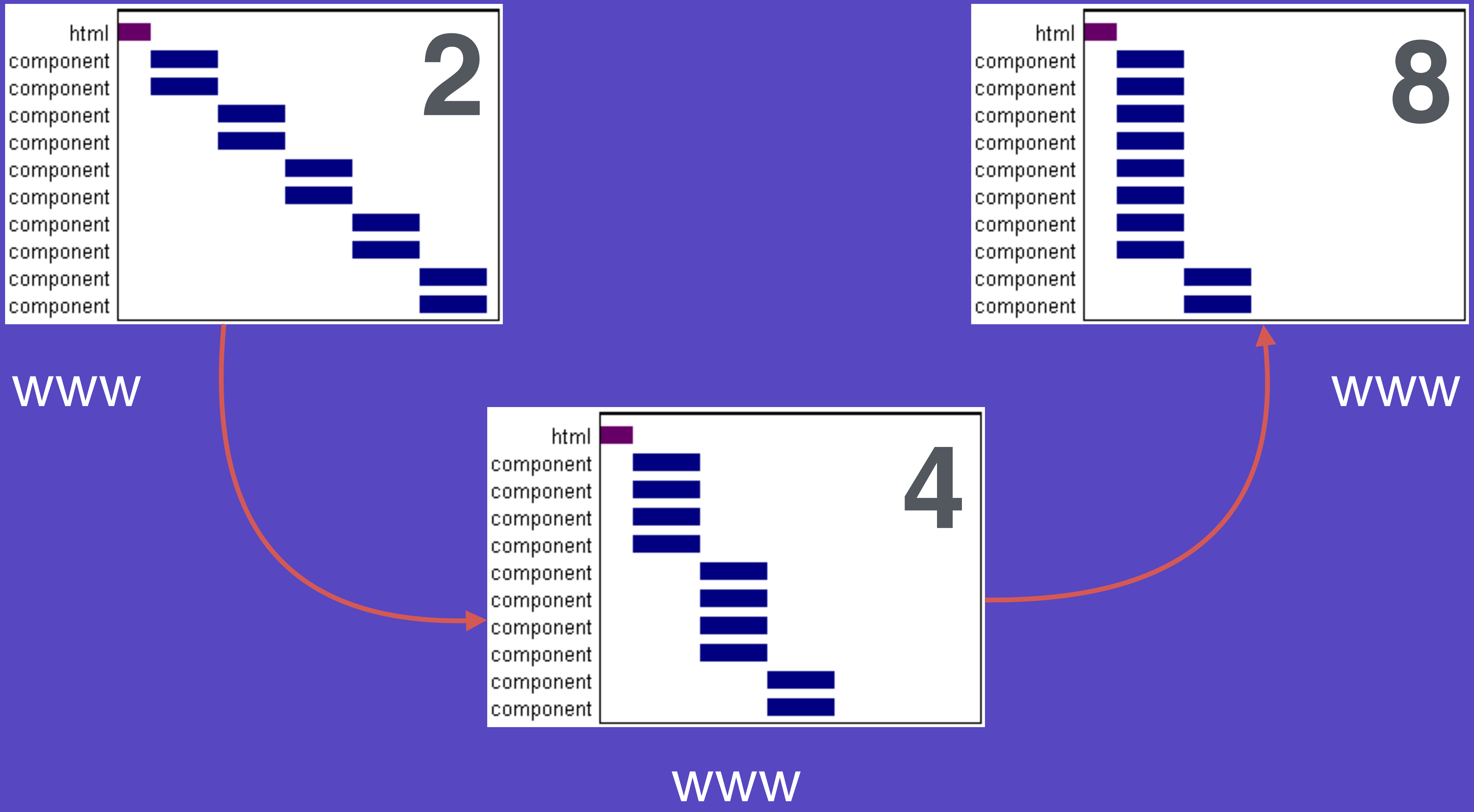
- hostname sharding
- for-loop array length caching
- dynamic script insertion

hostname sharding

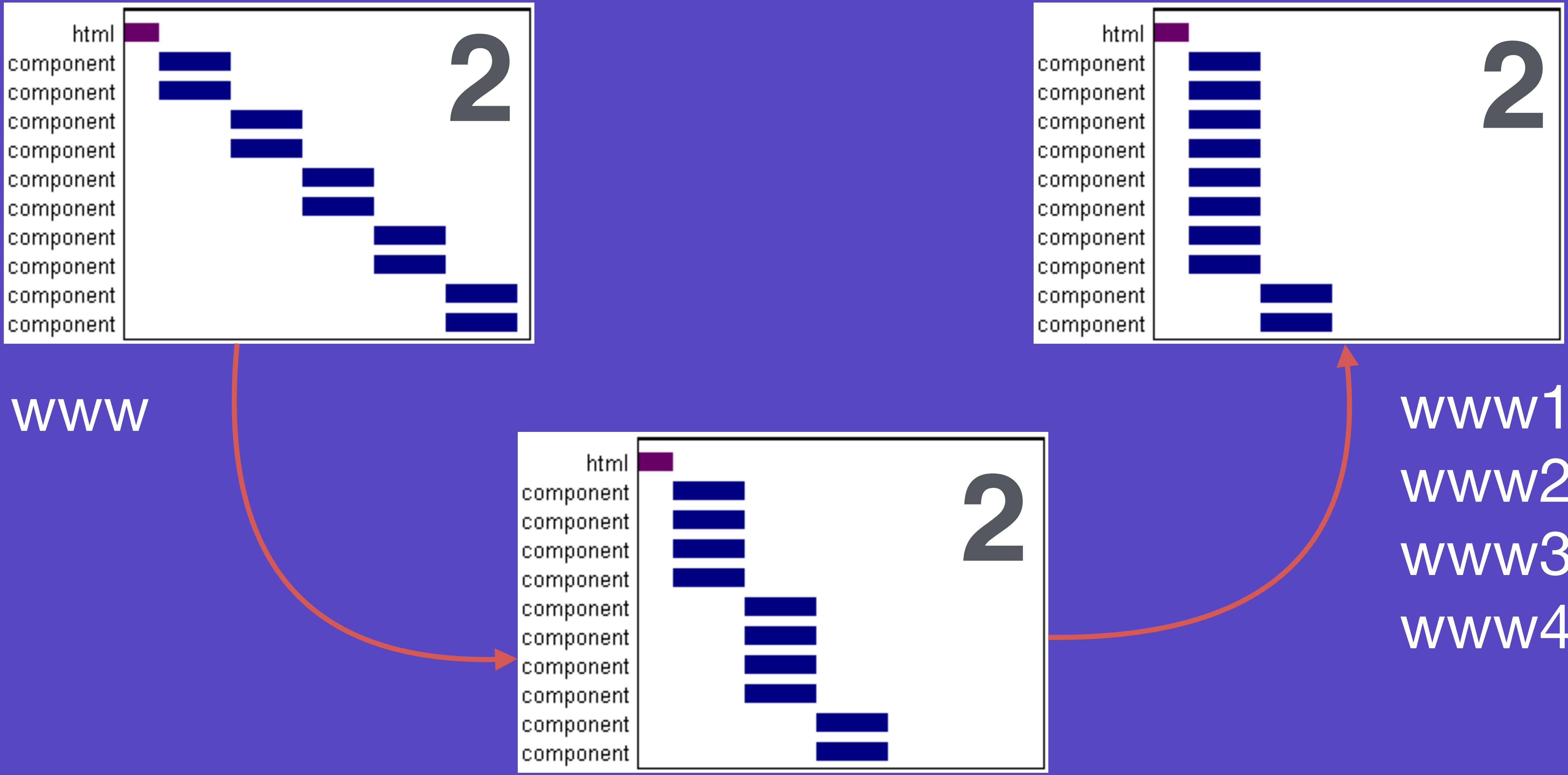
in the beginning (HTTP 1.0)



with more connections



with hostname sharding



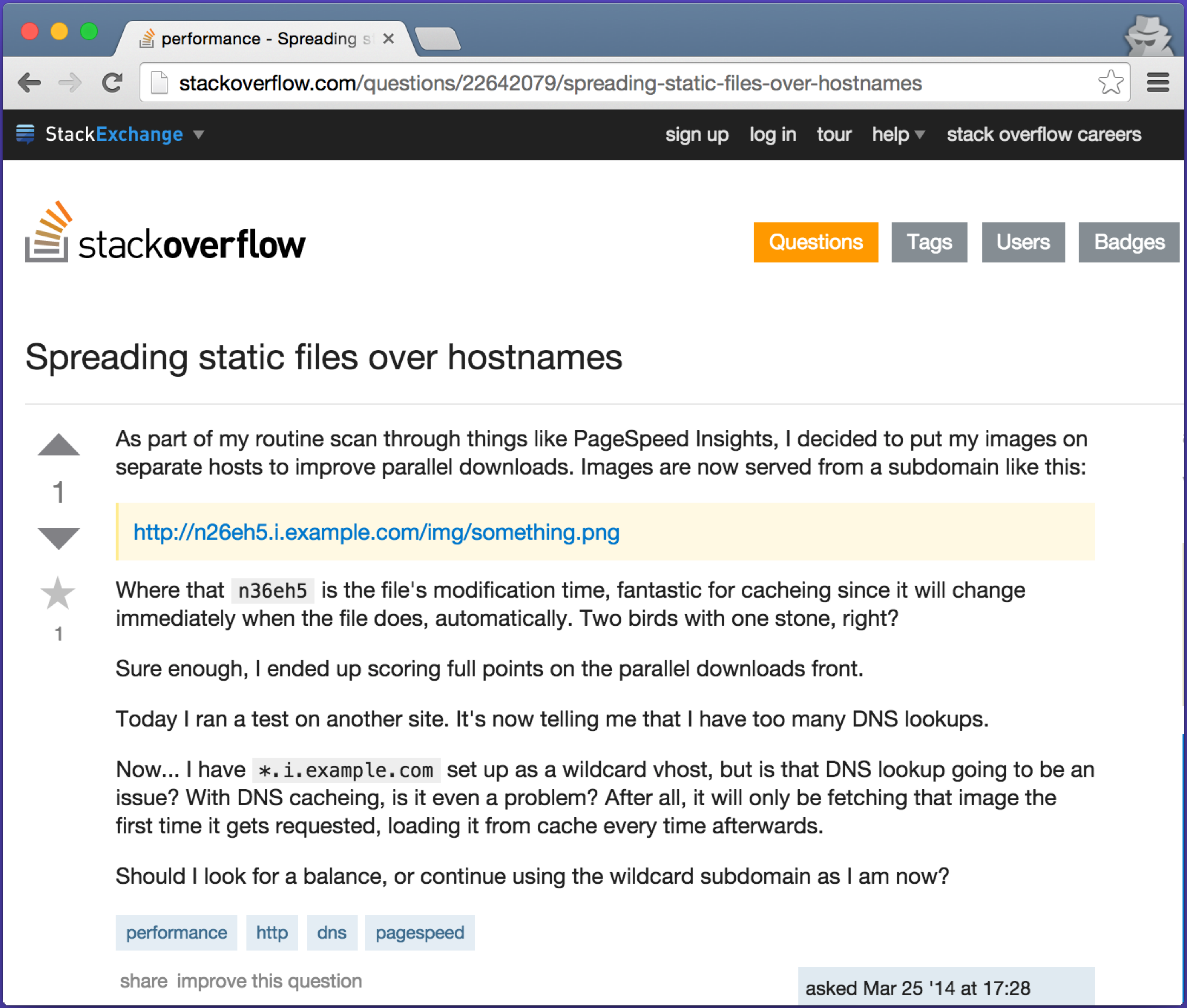
www

www1, www2

www1
www2
www3
www4

still common

Website	# Static Hosts
plus.google.com	4
tumblr.com	4
alibaba.com	4
theverge.com	4
ebay.com	6
businessinsider.com	6
netflix.com	11!


performance - Spreading s ×
stackoverflow.com/questions/22642079/spreading-static-files-over-hostnames
StackExchange
sign up
log in
tour
help
stack overflow careers
stackoverflow
Questions
Tags
Users
Badges
1
1

Spreading static files over hostnames

As part of my routine scan through things like PageSpeed Insights, I decided to put my images on separate hosts to improve parallel downloads. Images are now served from a subdomain like this:

> http://n26eh5.i.example.com/img/something.png

Where that `n36eh5` is the file's modification time, fantastic for cacheing since it will change immediately when the file does, automatically. Two birds with one stone, right?

Sure enough, I ended up scoring full points on the parallel downloads front.

Today I ran a test on another site. It's now telling me that I have too many DNS lookups.

Now... I have `*.i.example.com` set up as a wildcard vhost, but is that DNS lookup going to be an issue? With DNS cacheing, is it even a problem? After all, it will only be fetching that image the first time it gets requested, loading it from cache every time afterwards.

Should I look for a balance, or continue using the wildcard subdomain as I am now?

asked Mar 25 '14 at 17:28

not a big deal anymore

browser connections/origin

Browser	# HTTP / origin
Chrome 42	6
Firefox 37	8
Safari 7	6
IE 8, 9	6
IE 10	8
IE 11	13

bit.ly/rprf-bscope

chrome caps parallel image requests at 10 anyways

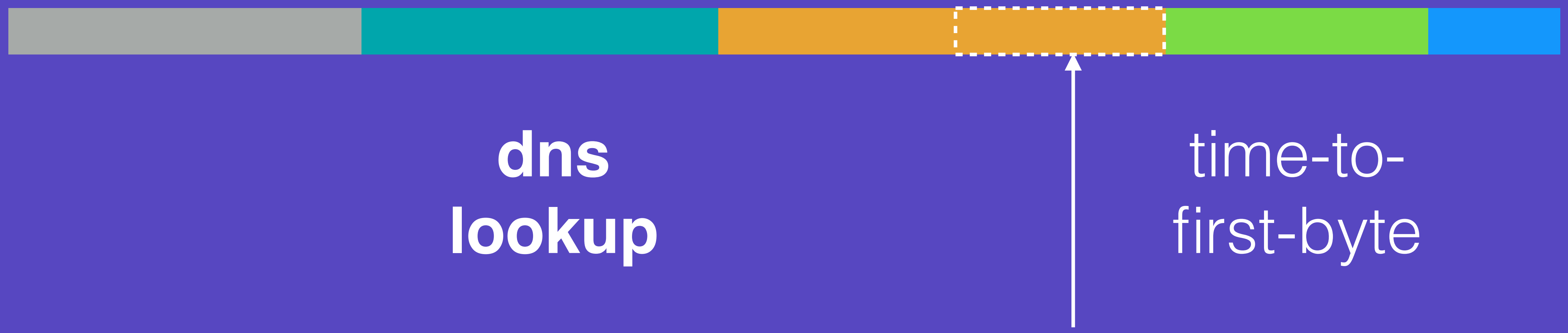
bit.ly/rprf-bscope

rise of https

stalled/
proxy negotiation

connection
+ tls/ssl

download



**dns
lookup**

time-to-
first-byte

tls/ssl handshake

and http/2 will make all this irrelevant anyways

“2 domains for non-SPDY modern browsers”

– Souders in 2013

bit.ly/rprf-2domains

etsy case study

- 4 image domains → 2
- 50-80 ms faster for image heavy pages
- 30-50 ms faster overall
- **up to 500ms faster on mobile**

bit.ly/rprf-etsy

the web's moving on

Website	# Static Hosts
netflix.com*	1
nytimes.com	1
youtube.com	2
twitter.com	2
facebook.com	2
pinterest.com	2
bbc.co.uk	2
etsy.com	3

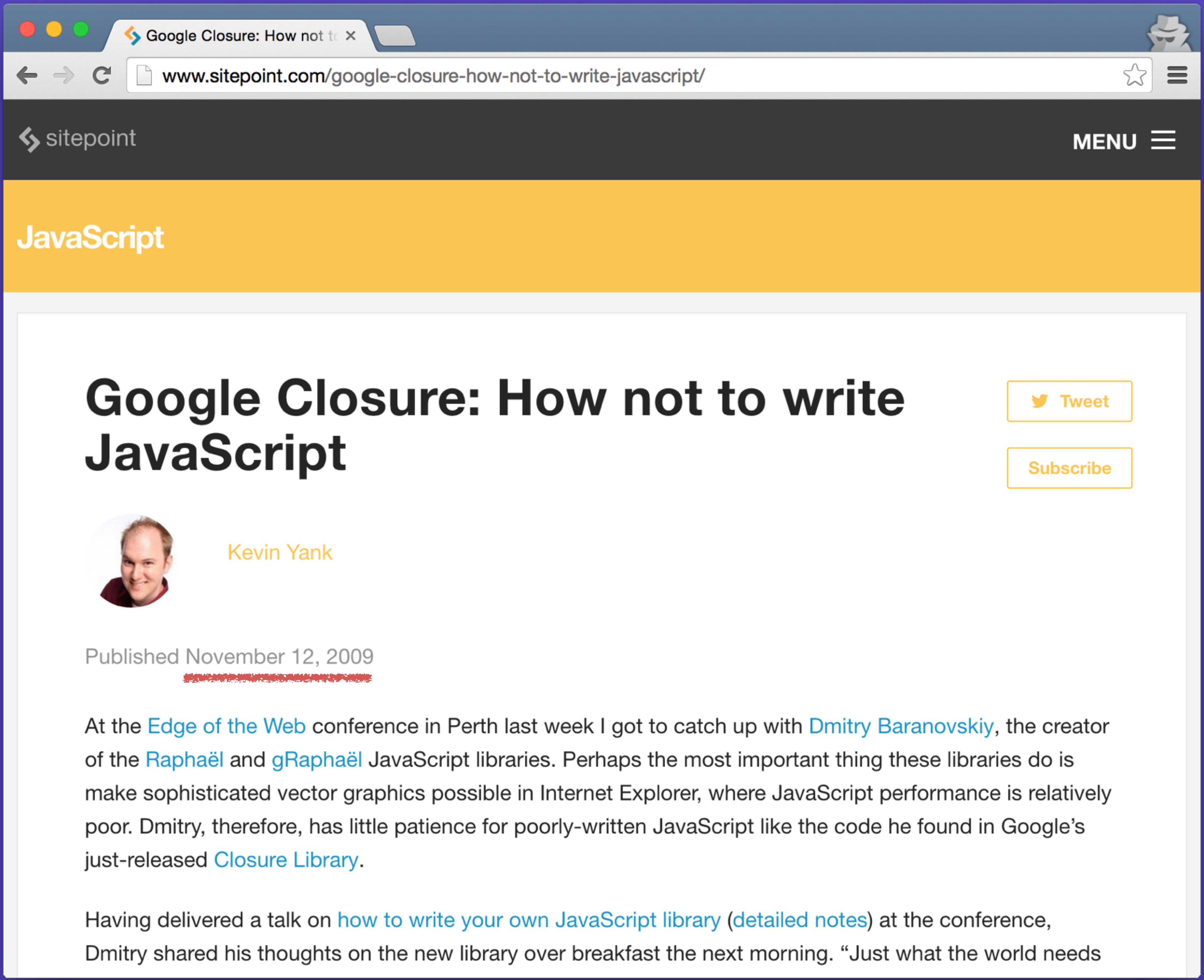
looping

specifically array length caching in for loops

```
for (var i = 0; i < arr.length; i++) {
  // do stuff
}
```

vs

```
for (var i = 0, len = arr.length;
     i < len; i++)
{
   // do stuff
}
```


Google Closure: How not t
www.sitepoint.com/google-closure-how-not-to-write-javascript/
sitepoint
MENU
JavaScript
Tweet
Subscribe

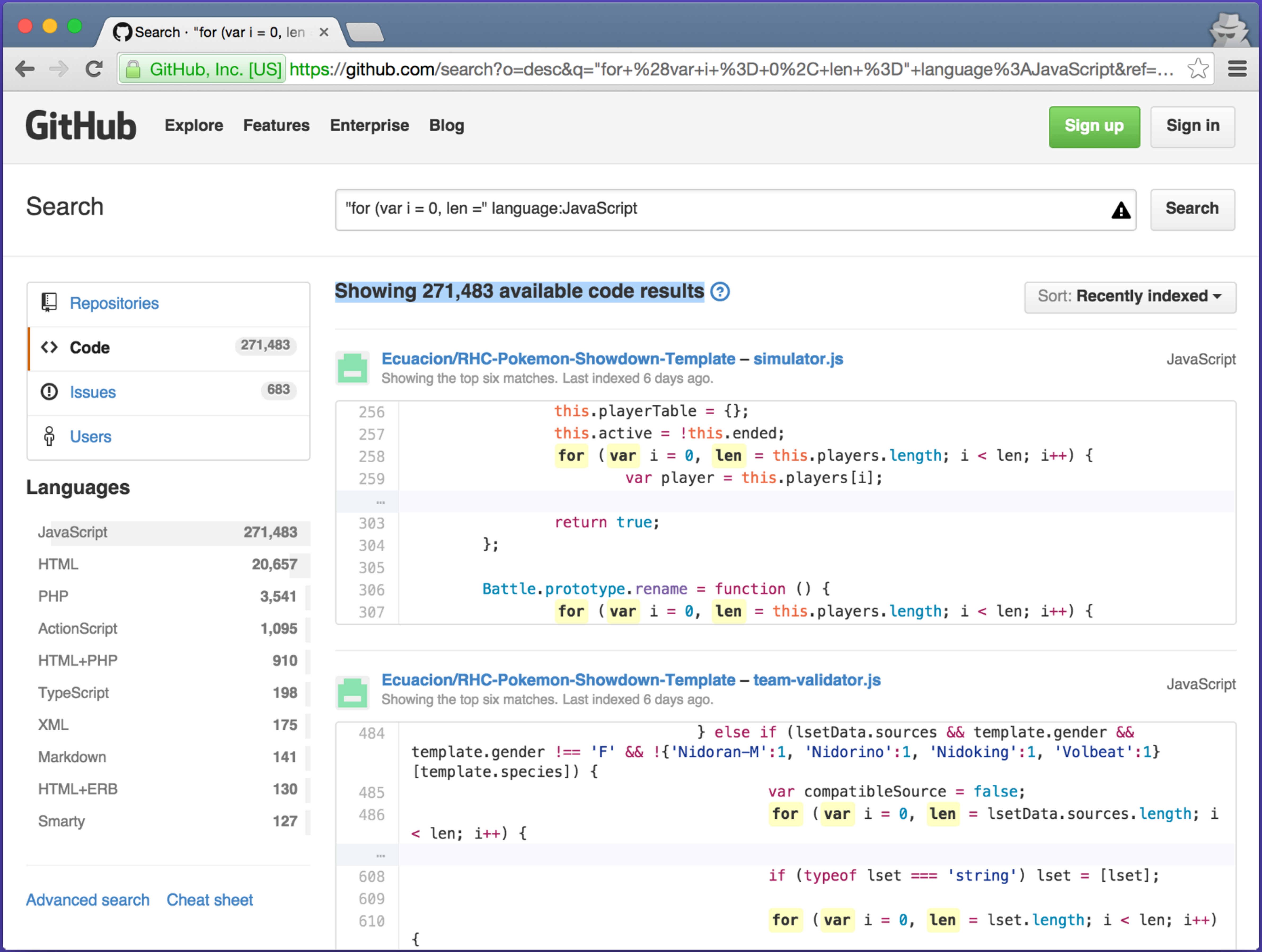
Google Closure: How not to write JavaScript

Kevin Yank

Published November 12, 2009

At the Edge of the Web conference in Perth last week I got to catch up with Dmitry Baranovskiy, the creator of the Raphaël and gRaphaël JavaScript libraries. Perhaps the most important thing these libraries do is make sophisticated vector graphics possible in Internet Explorer, where JavaScript performance is relatively poor. Dmitry, therefore, has little patience for poorly-written JavaScript like the code he found in Google's just-released Closure Library.

Having delivered a talk on how to write your own JavaScript library (detailed notes) at the conference, Dmitry shared his thoughts on the new library over breakfast the next morning. "Just what the world needs


Search · "for (var i = 0, len
GitHub, Inc. [US] https://github.com/search?o=desc&q="for+%28var+i+%3D+0%2C+len+%3D"+language%3AJavaScript&ref=…
GitHub
Explore
Features
Enterprise
Blog
Sign up
Sign in

Search

"for (var i = 0, len =" language:JavaScript

Repositories	
Code	271,483
Issues	683
Users	

Languages

JavaScript	271,483
HTML	20,657
PHP	3,541
ActionScript	1,095
HTML+PHP	910
TypeScript	198
XML	175
Markdown	141
HTML+ERB	130
Smarty	127

Advanced search
Cheat sheet

Showing 271,483 available code results

Ecuacion/RHC-Pokemon-Showdown-Template – simulator.js

Showing the top six matches. Last indexed 6 days ago.

JavaScript

```
                this.playerTable = {};
                this.active = !this.ended;
                for (var i = 0, len = this.players.length; i < len; i++) {
                        var player = this.players[i];
…
                return true;
        };

        Battle.prototype.rename = function () {
                for (var i = 0, len = this.players.length; i < len; i++) {
```

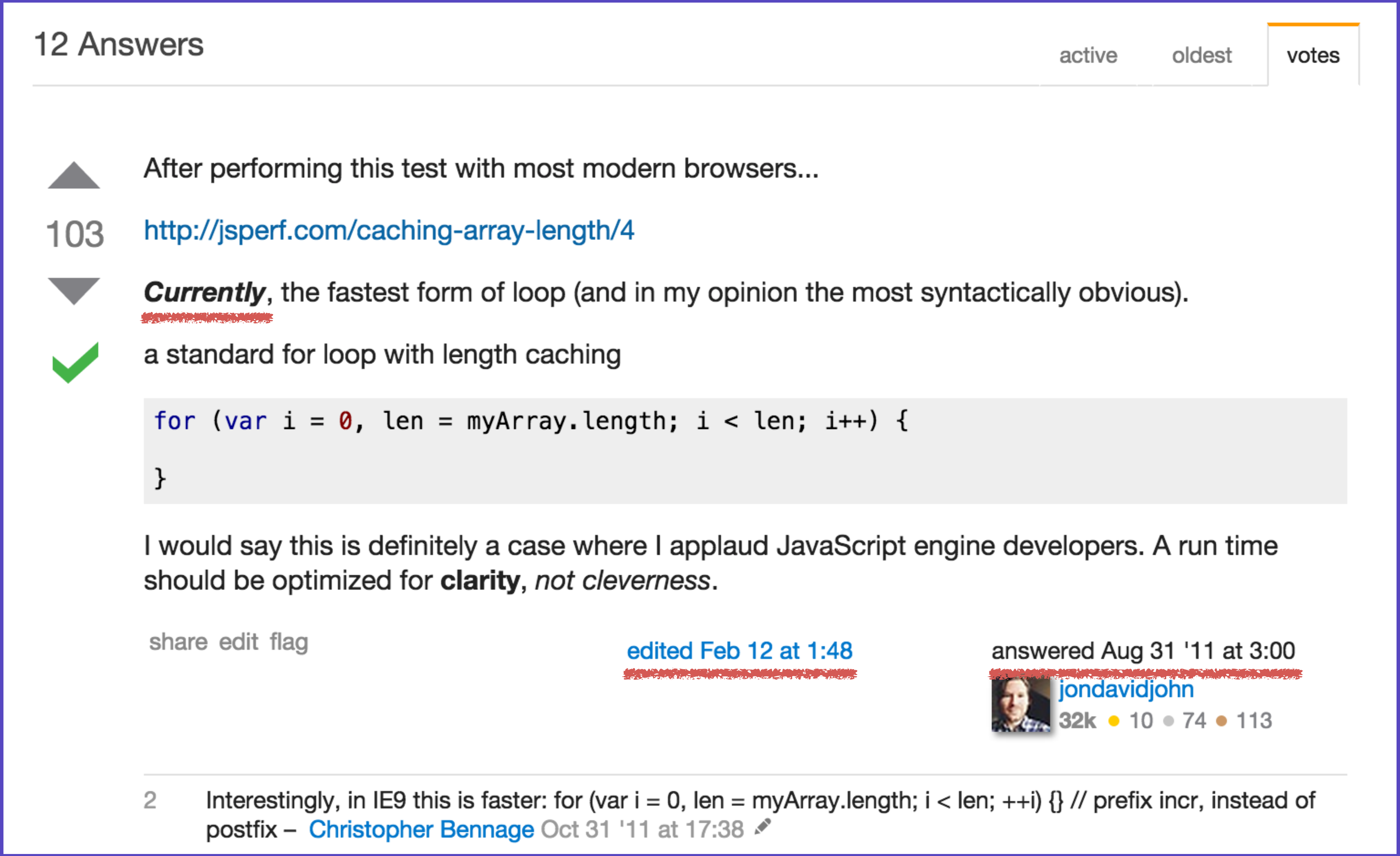
Ecuacion/RHC-Pokemon-Showdown-Template – team-validator.js

Showing the top six matches. Last indexed 6 days ago.

JavaScript

```
                                } else if (lsetData.sources && template.gender &&
template.gender !== 'F' && !{'Nidoran-M':1, 'Nidorino':1, 'Nidoking':1, 'Volbeat':1}
[template.species]) {
                                        var compatibleSource = false;
                                        for (var i = 0, len = lsetData.sources.length; i
< len; i++) {
…
                                        if (typeof lset === 'string') lset = [lset];

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



12 Answers

103

After performing this test with most modern browsers...

http://jsperf.com/caching-array-length/4

Currently, the fastest form of loop (and in my opinion the most syntactically obvious).

a standard for loop with length caching

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

}
```

I would say this is definitely a case where I applaud JavaScript engine developers. A run time should be optimized for **clarity**, *not cleverness*.

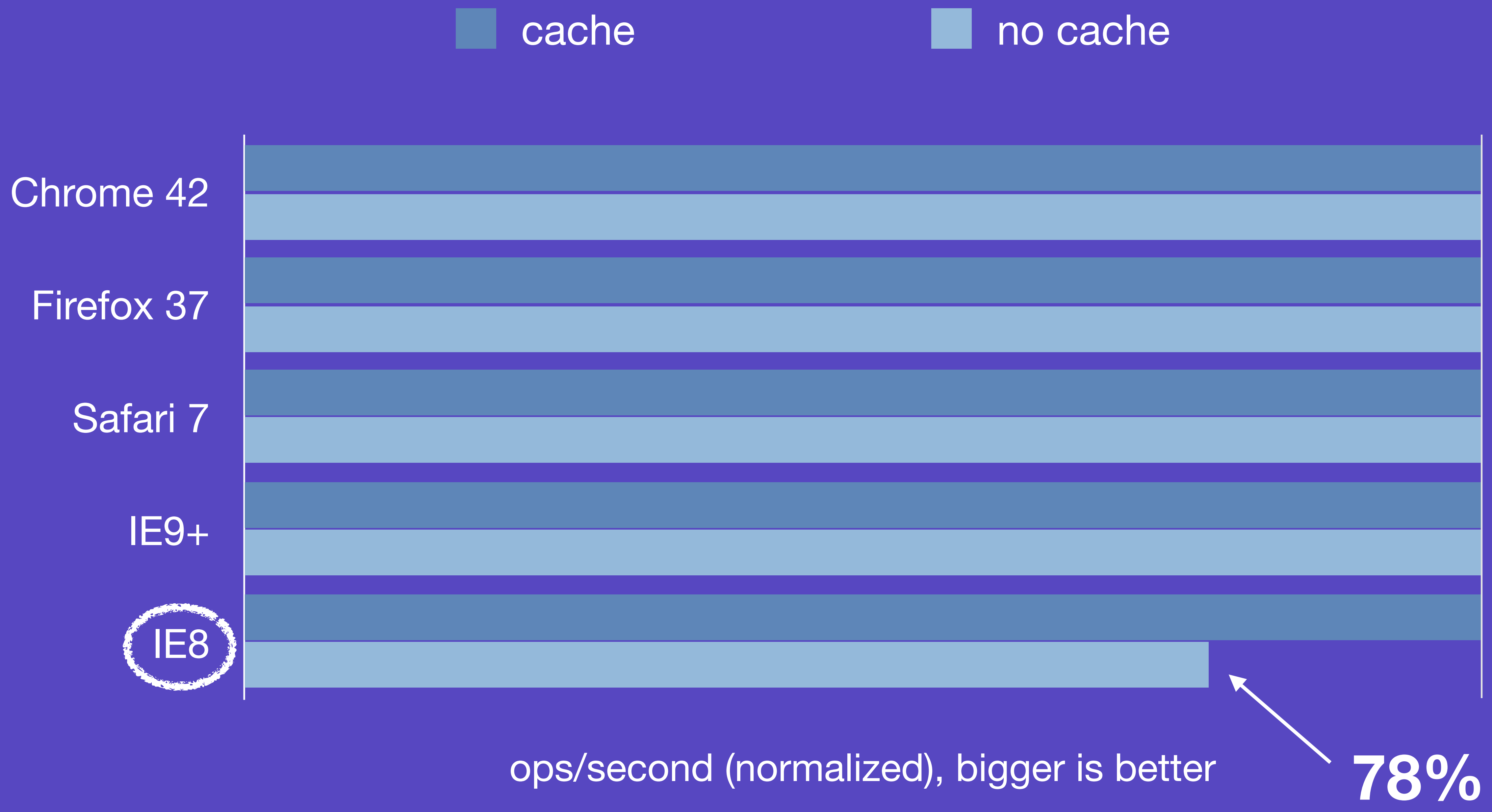
edited Feb 12 at 1:48

answered Aug 31 '11 at 3:00
jondavidjohn
32k ● 10 ● 74 ● 113

2 Interestingly, in IE9 this is faster: for (var i = 0, len = myArray.length; i < len; ++i) {} // prefix incr, instead of postfix – Christopher Bennage Oct 31 '11 at 17:38

does it still hold?

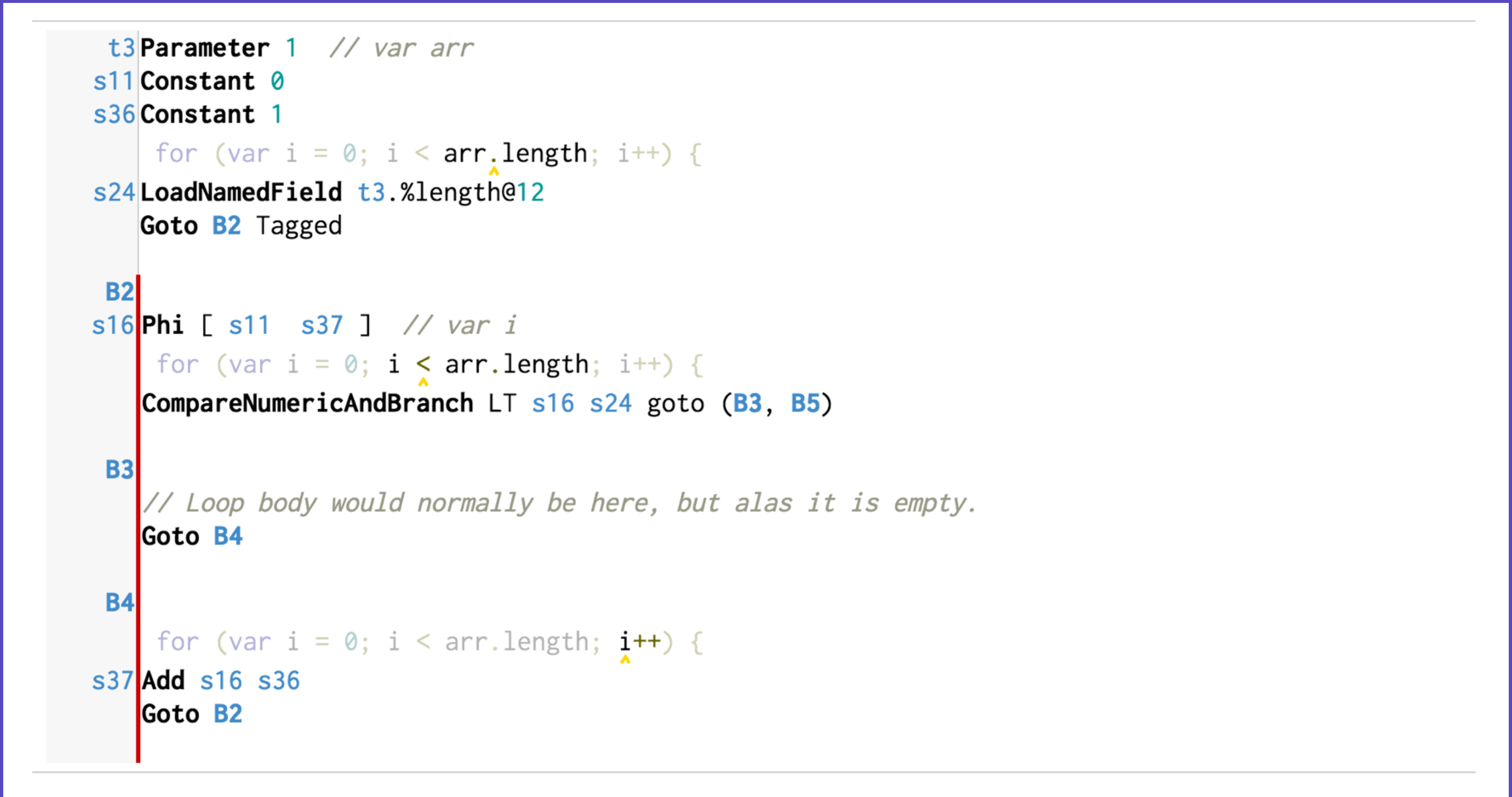
cache vs no cache



V8 (and other browsers) recognize this pattern

bit.ly/rprf-v8opt

uncached version



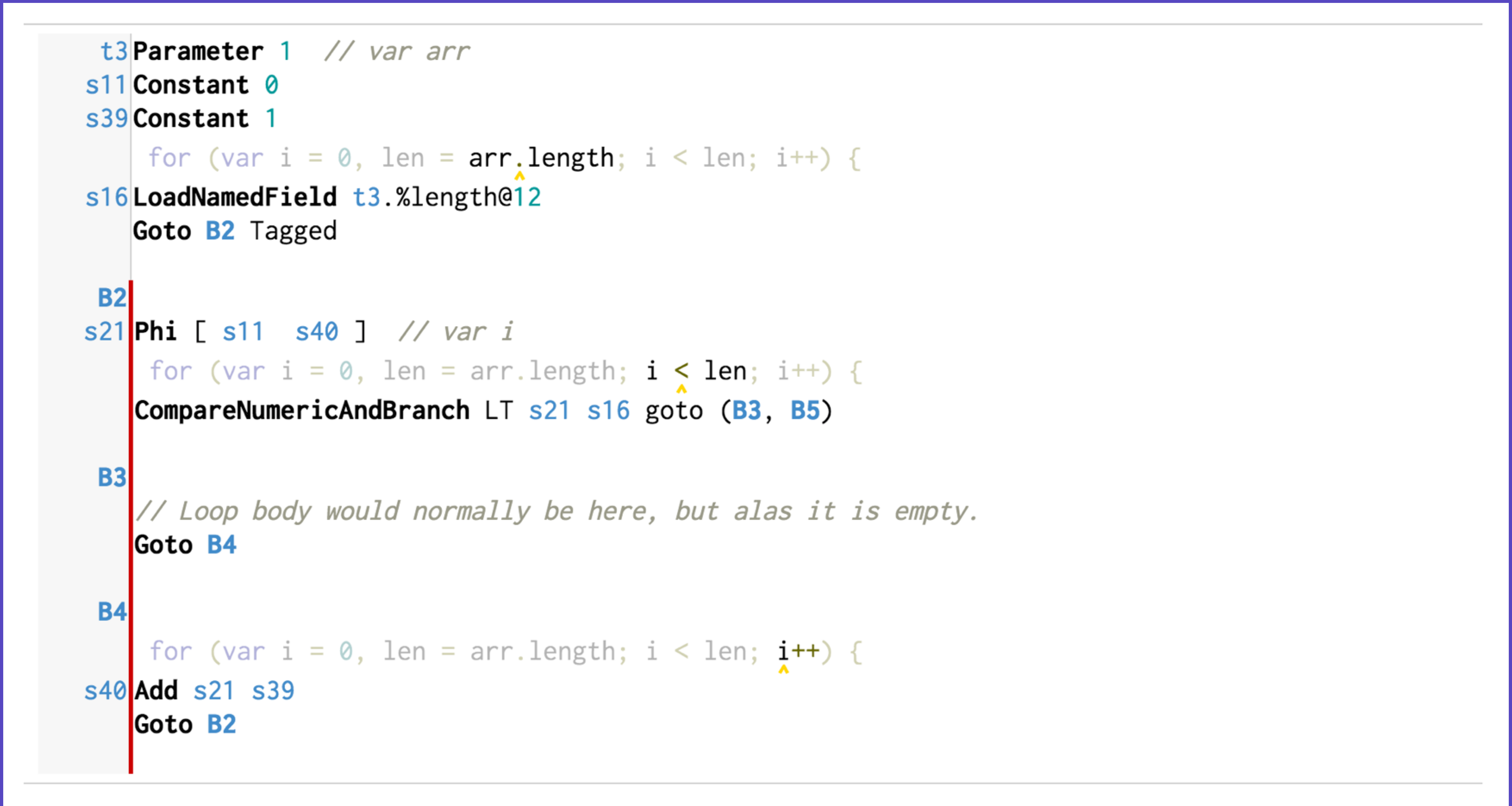
```
 t3 Parameter 1   // var arr
s11 Constant 0
s36 Constant 1
    for (var i = 0; i < arr.length; i++) {
                           ^
s24 LoadNamedField t3.%length@12
    Goto B2 Tagged

 B2
s16 Phi [ s11  s37 ]  // var i
    for (var i = 0; i < arr.length; i++) {
                      ^
    CompareNumericAndBranch LT s16 s24 goto (B3, B5)

 B3
    // Loop body would normally be here, but alas it is empty.
    Goto B4

 B4
    for (var i = 0; i < arr.length; i++) {
                                    ^
s37 Add s16 s36
    Goto B2
```

cached version



```
 t3 Parameter 1   // var arr
s11 Constant 0
s39 Constant 1
     for (var i = 0, len = arr.length; i < len; i++) {
                              ^
s16 LoadNamedField t3.%length@12
    Goto B2 Tagged

 B2
s21 Phi [ s11  s40 ]  // var i
     for (var i = 0, len = arr.length; i < len; i++) {
                                          ^
    CompareNumericAndBranch LT s21 s16 goto (B3, B5)

 B3
    // Loop body would normally be here, but alas it is empty.
    Goto B4

 B4
     for (var i = 0, len = arr.length; i < len; i++) {
                                                ^
s40 Add s21 s39
    Goto B2
```

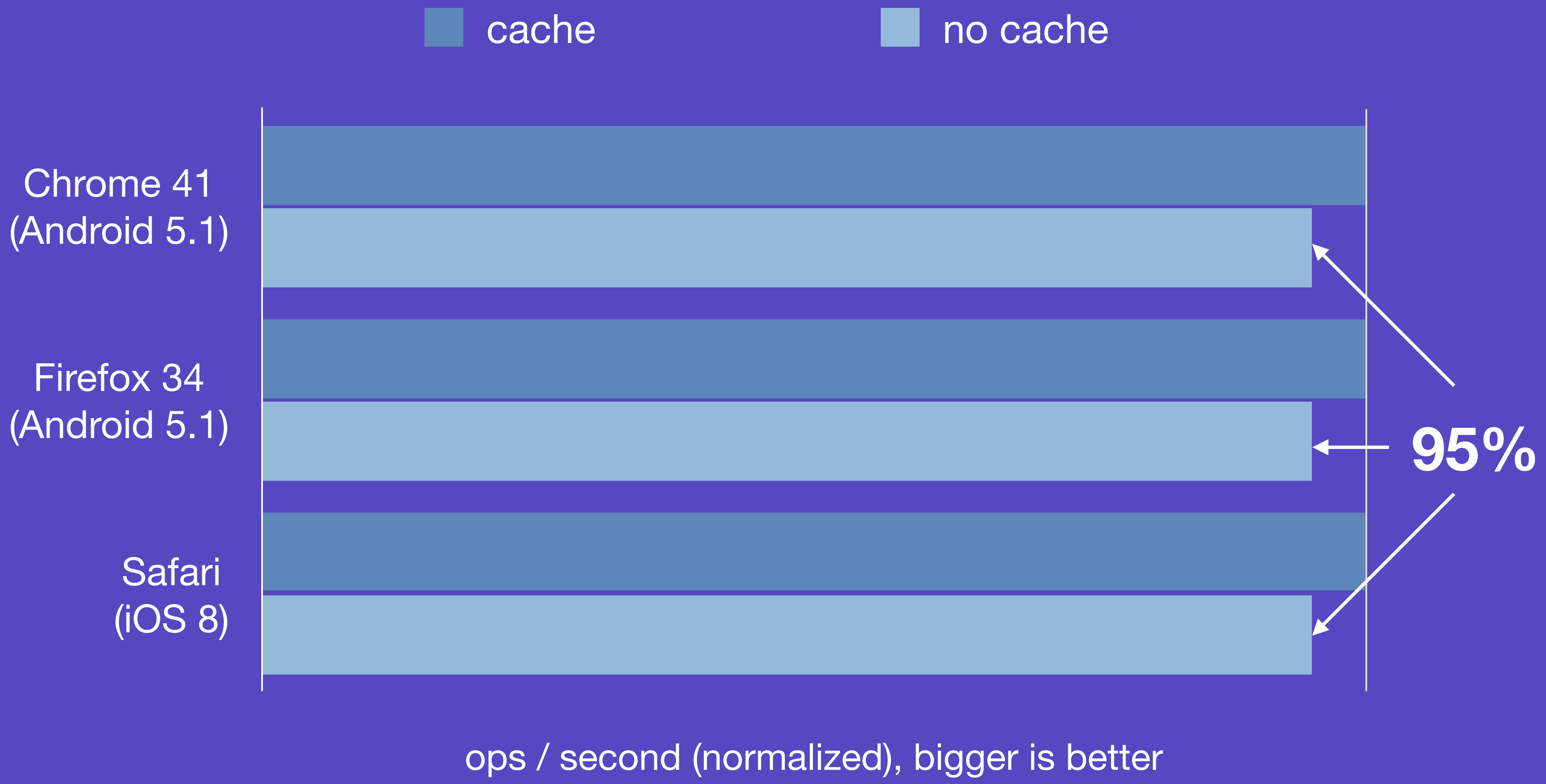
“we should start assuming that our code is optimized”

- Vyacheslav Egorov, V8

bit.ly/rprf-v8opt

if you can trivially optimize it, the browser (probably) can too

mobile disagrees, a little



if you think you’re going to get performance gains from optimizing for loops

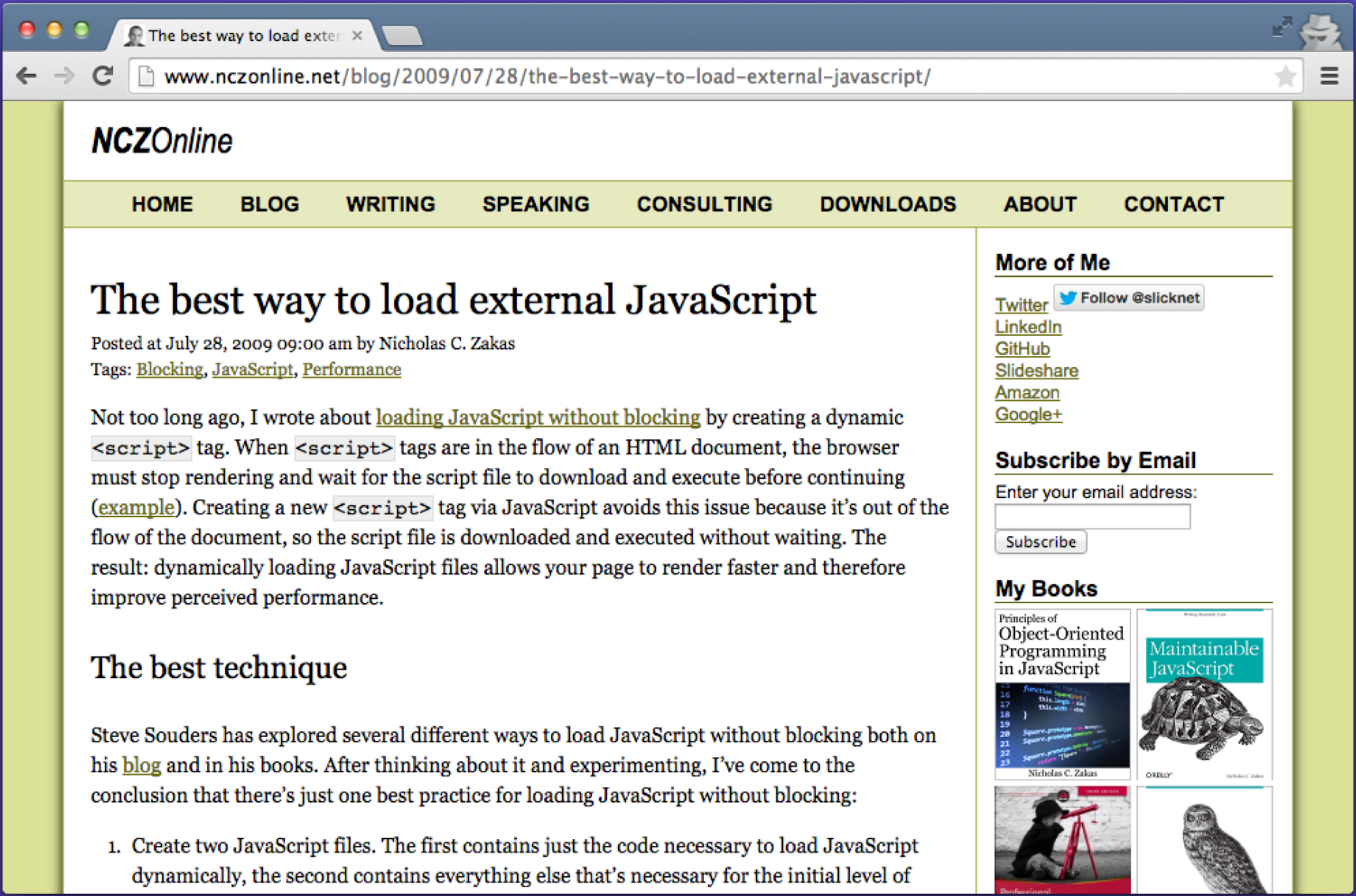
you’re gonna have a bad time

dynamic script insertion

```
<script src="/app.js"></script>
```

VS

```
<script>
  var script = document.createElement('script');
  script.src = '/app.js';
  document.getElementsByTagName('head')[0]
    .appendChild(script);
</script>
```



The best way to load external JavaScript

Posted at July 28, 2009 09:00 am by Nicholas C. Zakas

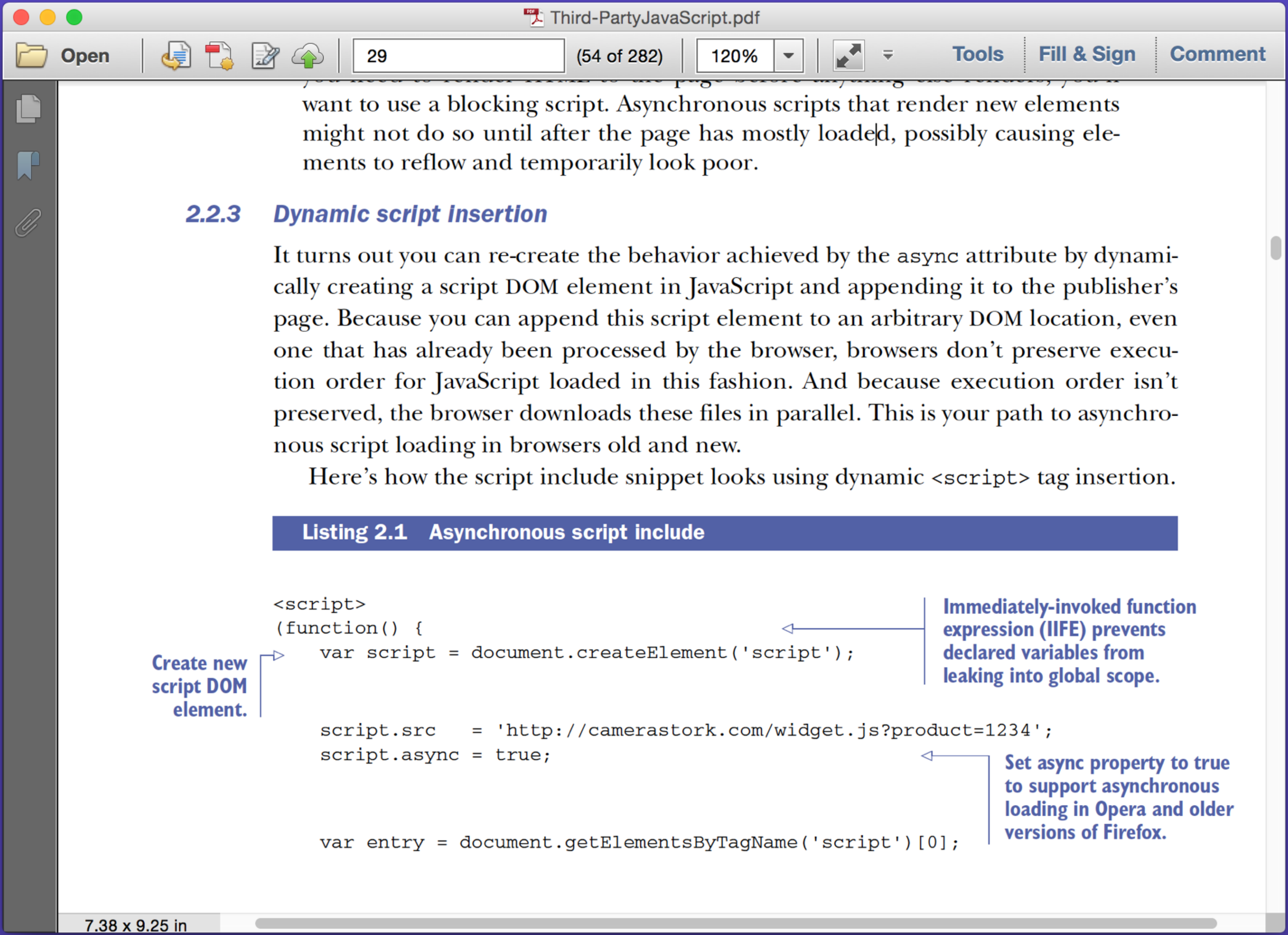
Tags: Blocking, JavaScript, Performance

Not too long ago, I wrote about loading JavaScript without blocking by creating a dynamic `<script>` tag. When `<script>` tags are in the flow of an HTML document, the browser must stop rendering and wait for the script file to download and execute before continuing (example). Creating a new `<script>` tag via JavaScript avoids this issue because it's out of the flow of the document, so the script file is downloaded and executed without waiting. The result: dynamically loading JavaScript files allows your page to render faster and therefore improve perceived performance.

The best technique

Steve Souders has explored several different ways to load JavaScript without blocking both on his blog and in his books. After thinking about it and experimenting, I've come to the conclusion that there's just one best practice for loading JavaScript without blocking:

1. Create two JavaScript files. The first contains just the code necessary to load JavaScript dynamically, the second contains everything else that's necessary for the initial level of



want to use a blocking script. Asynchronous scripts that render new elements might not do so until after the page has mostly loaded, possibly causing elements to reflow and temporarily look poor.

2.2.3 Dynamic script insertion

It turns out you can re-create the behavior achieved by the async attribute by dynamically creating a script DOM element in JavaScript and appending it to the publisher's page. Because you can append this script element to an arbitrary DOM location, even one that has already been processed by the browser, browsers don't preserve execution order for JavaScript loaded in this fashion. And because execution order isn't preserved, the browser downloads these files in parallel. This is your path to asynchronous script loading in browsers old and new.

Here's how the script include snippet looks using dynamic <script> tag insertion.

Listing 2.1 Asynchronous script include

```
<script>
(function() {
    var script = document.createElement('script');

    script.src   = 'http://camerastork.com/widget.js?product=1234';
    script.async = true;

    var entry = document.getElementsByTagName('script')[0];
```

Immediately-invoked function expression (IIFE) prevents declared variables from leaking into global scope.

Create new script DOM element.

Set async property to true to support asynchronous loading in Opera and older versions of Firefox.

who does this?

- analytics: GA, Mixpanel, Chartbeat, Wordpress
- widgets: Disqus, Facebook Comments
- JS module loaders: RequireJS, LabJS
- script managers: Google Tag Manager, Segment

small problem:

CSS Object Model

CSS Object Model (CCSOM)

```
<link type="text/stylesheet" href="/main.css"/>
```

can’t execute until CSS ready

```
<script>
  window.getComputedStyle(document.body).margin;
</script>
```

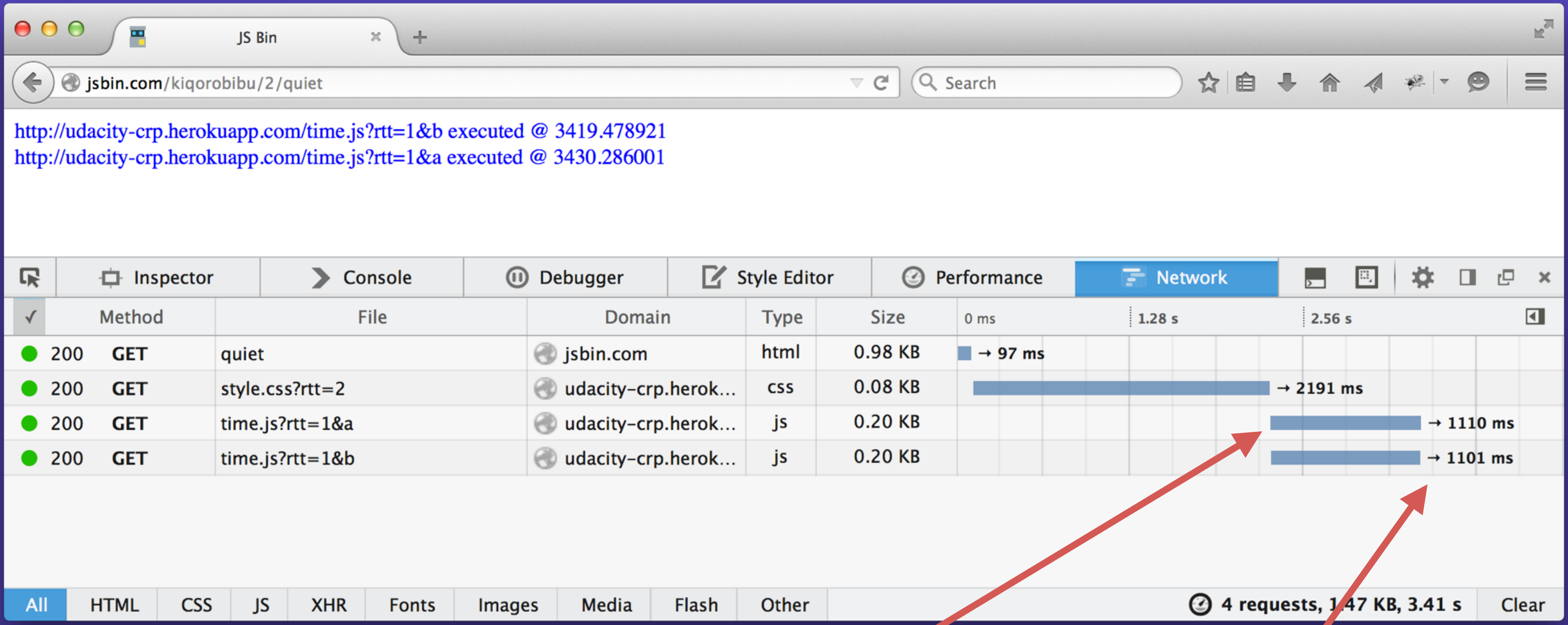
CSSOM + dynamic script insertion

```
<link type="text/stylesheet" href="/main.css"/>
```

can’t execute until CSS ready

```
<script>
  var script = document.createElement('script');
  script.src = '/app.js';
  document.getElementsByTagName('head')[0]
    .appendChild(script);
</script>
```

dynamic script insertion



http://udacity-crp.herokuapp.com/time.js?rtt=1&b executed @ 3419.478921
http://udacity-crp.herokuapp.com/time.js?rtt=1&a executed @ 3430.286001

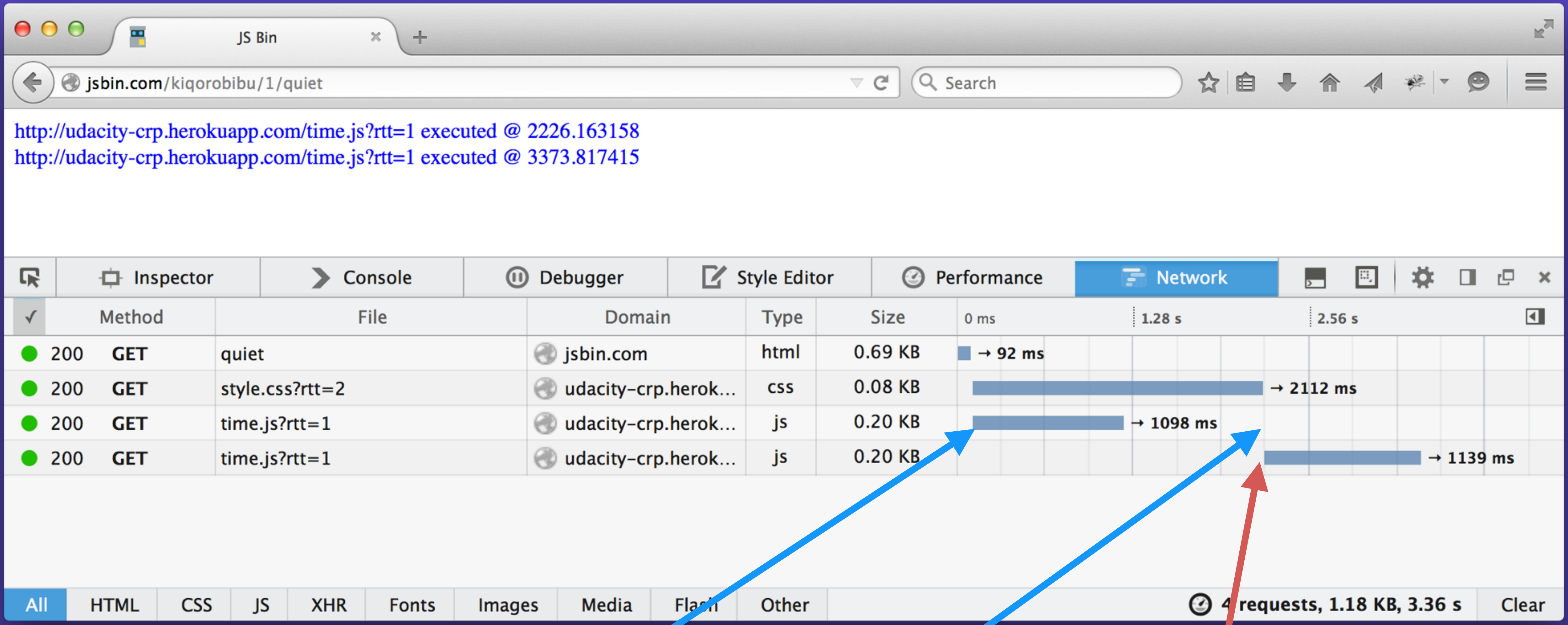
✓	Method	File	Domain	Type	Size	0 ms · 1.28 s · 2.56 s
● 200	GET	quiet	jsbin.com	html	0.98 KB	→ 97 ms
● 200	GET	style.css?rtt=2	udacity-crp.herok...	css	0.08 KB	→ 2191 ms
● 200	GET	time.js?rtt=1&a	udacity-crp.herok...	js	0.20 KB	→ 1110 ms
● 200	GET	time.js?rtt=1&b	udacity-crp.herok...	js	0.20 KB	→ 1101 ms

4 requests, 1.47 KB, 3.41 s

inline scripts can’t execute until CSSOM ready; downloading is delayed

scripts execute

blocking



http://udacity-crp.herokuapp.com/time.js?rtt=1 executed @ 2226.163158
http://udacity-crp.herokuapp.com/time.js?rtt=1 executed @ 3373.817415

✓	Method	File	Domain	Type	Size	0 ms / 1.28 s / 2.56 s
● 200	GET	quiet	jsbin.com	html	0.69 KB	→ 92 ms
● 200	GET	style.css?rtt=2	udacity-crp.herok...	css	0.08 KB	→ 2112 ms
● 200	GET	time.js?rtt=1	udacity-crp.herok...	js	0.20 KB	→ 1098 ms
● 200	GET	time.js?rtt=1	udacity-crp.herok...	js	0.20 KB	→ 1139 ms

4 requests, 1.18 KB, 3.36 s

1st script preloaded

1st script executes

2nd script downloads after 1st executes

“Have we been doing it all wrong?”

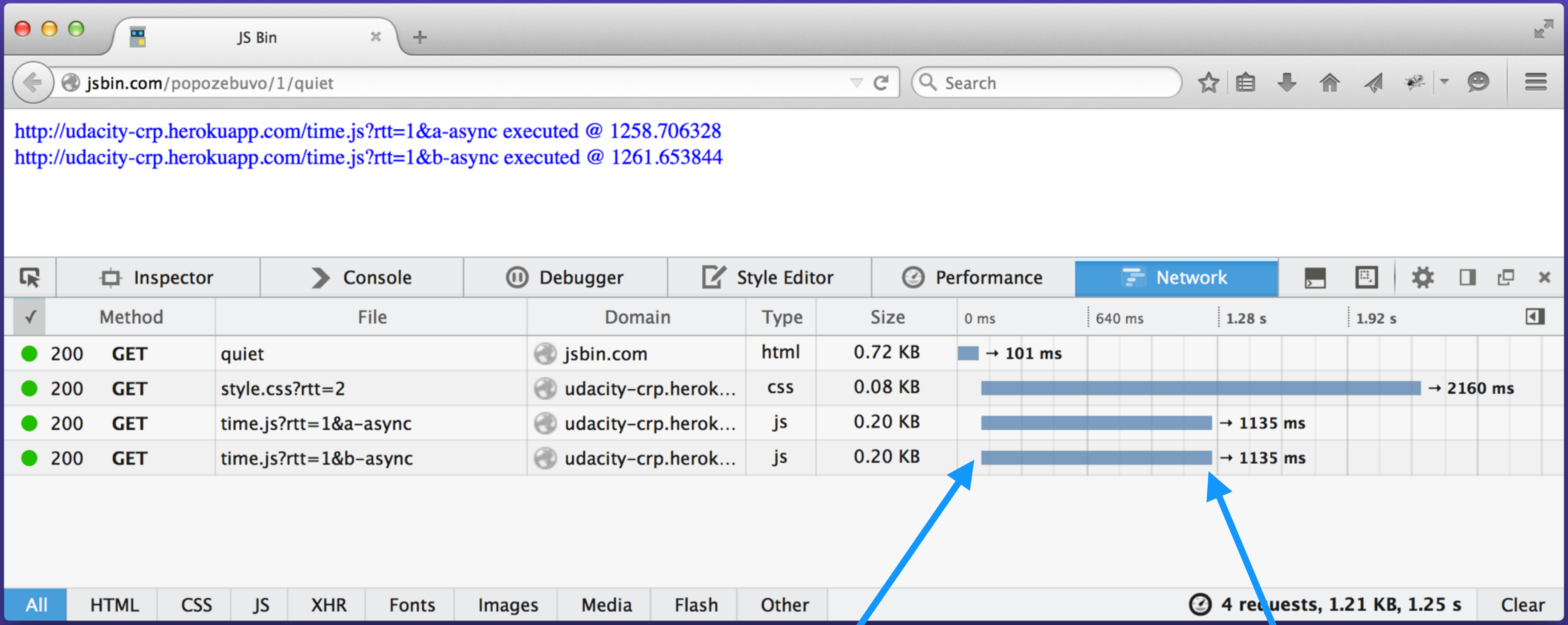
- Ilya Grigorik, 2014

bit.ly/rprf-injected

3rd option: async attribute

```
<script async src="/app.js"></script>
```

ideal: async attribute



http://udacity-crp.herokuapp.com/time.js?rtt=1&a-async executed @ 1258.706328
http://udacity-crp.herokuapp.com/time.js?rtt=1&b-async executed @ 1261.653844

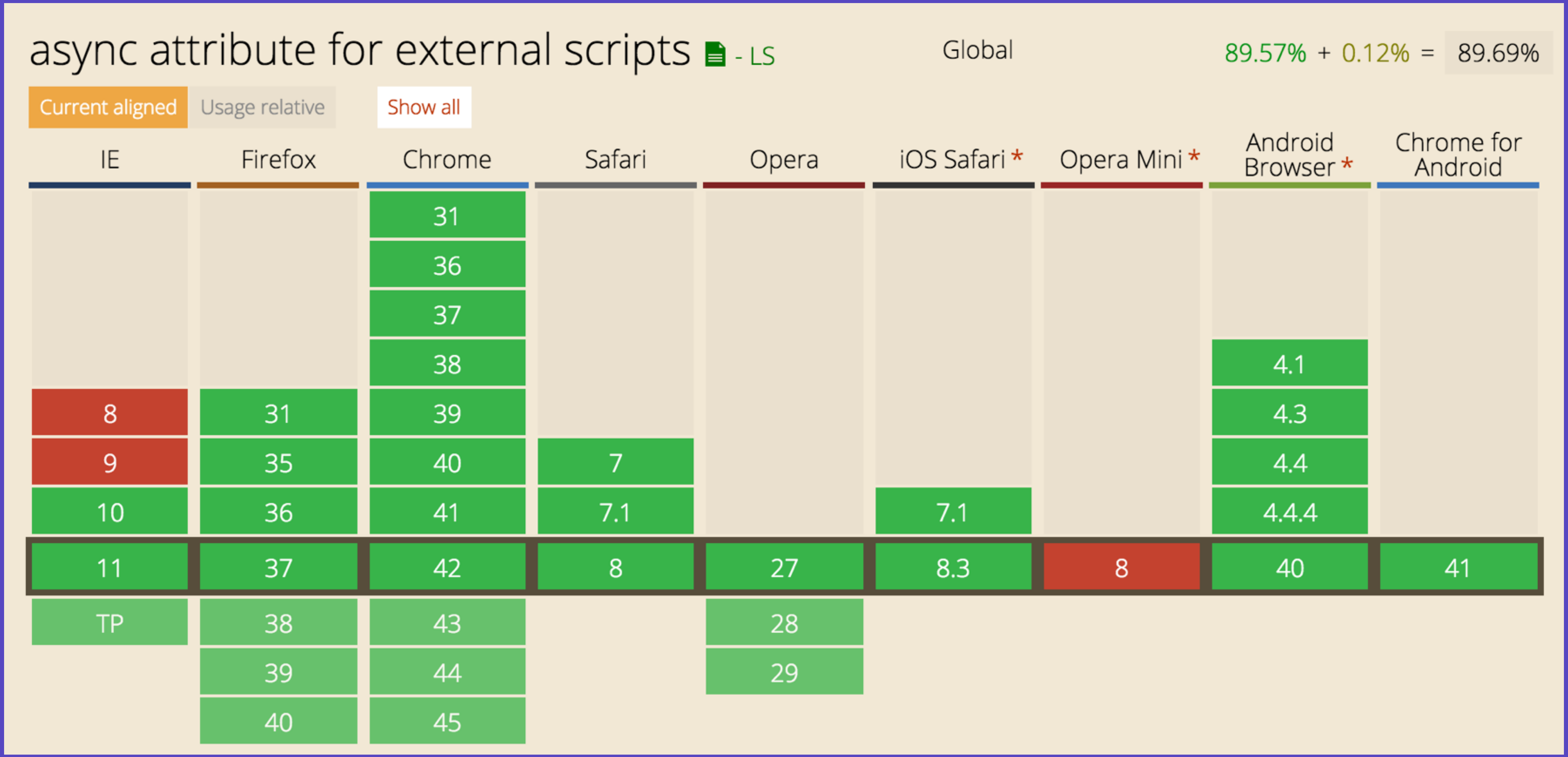
✓	Method	File	Domain	Type	Size	0 ms · 640 ms · 1.28 s · 1.92 s
● 200	GET	quiet	jsbin.com	html	0.72 KB	→ 101 ms
● 200	GET	style.css?rtt=2	udacity-crp.herok...	css	0.08 KB	→ 2160 ms
● 200	GET	time.js?rtt=1&a-async	udacity-crp.herok...	js	0.20 KB	→ 1135 ms
● 200	GET	time.js?rtt=1&b-async	udacity-crp.herok...	js	0.20 KB	→ 1135 ms

4 requests, 1.21 KB, 1.25 s

all 3 resources download in parallel

both scripts execute

we should probably start using async

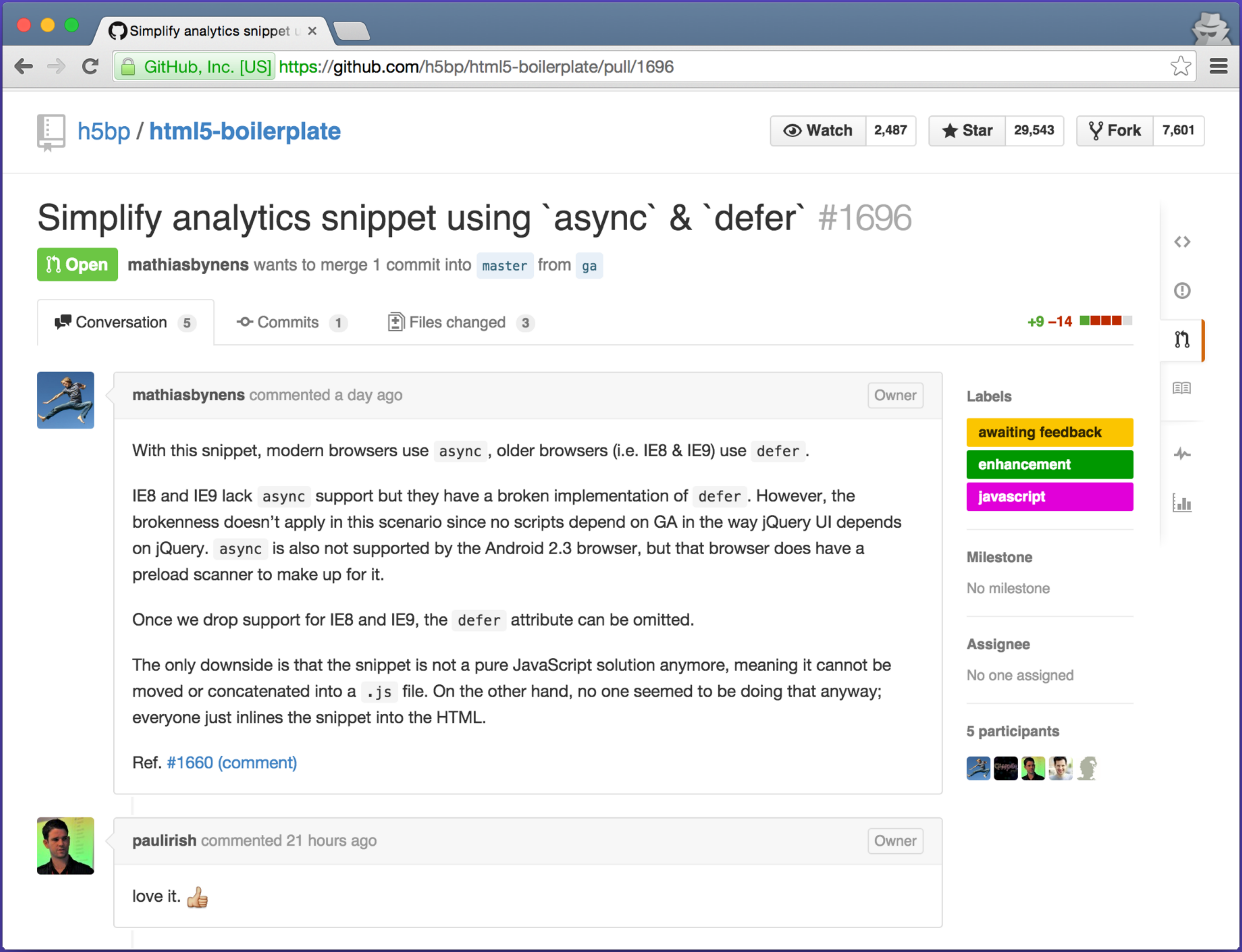


async attribute for external scripts - LS

Global 89.57% + 0.12% = 89.69%

Current aligned | Usage relative | Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		36						
		37						
		38					4.1	
🟥 8	31	39					4.3	
🟥 9	35	40	7				4.4	
10	36	41	7.1		7.1		4.4.4	
11	37	42	8	27	8.3	🟥 8	40	41
TP	38	43		28				
	39	44		29				
	40	45						



Simplify analytics snippet using `async` & `defer` #1696

Open **mathiasbynens** wants to merge 1 commit into `master` from `ga`

mathiasbynens commented a day ago

With this snippet, modern browsers use `async`, older browsers (i.e. IE8 & IE9) use `defer`.

IE8 and IE9 lack `async` support but they have a broken implementation of `defer`. However, the brokenness doesn't apply in this scenario since no scripts depend on GA in the way jQuery UI depends on jQuery. `async` is also not supported by the Android 2.3 browser, but that browser does have a preload scanner to make up for it.

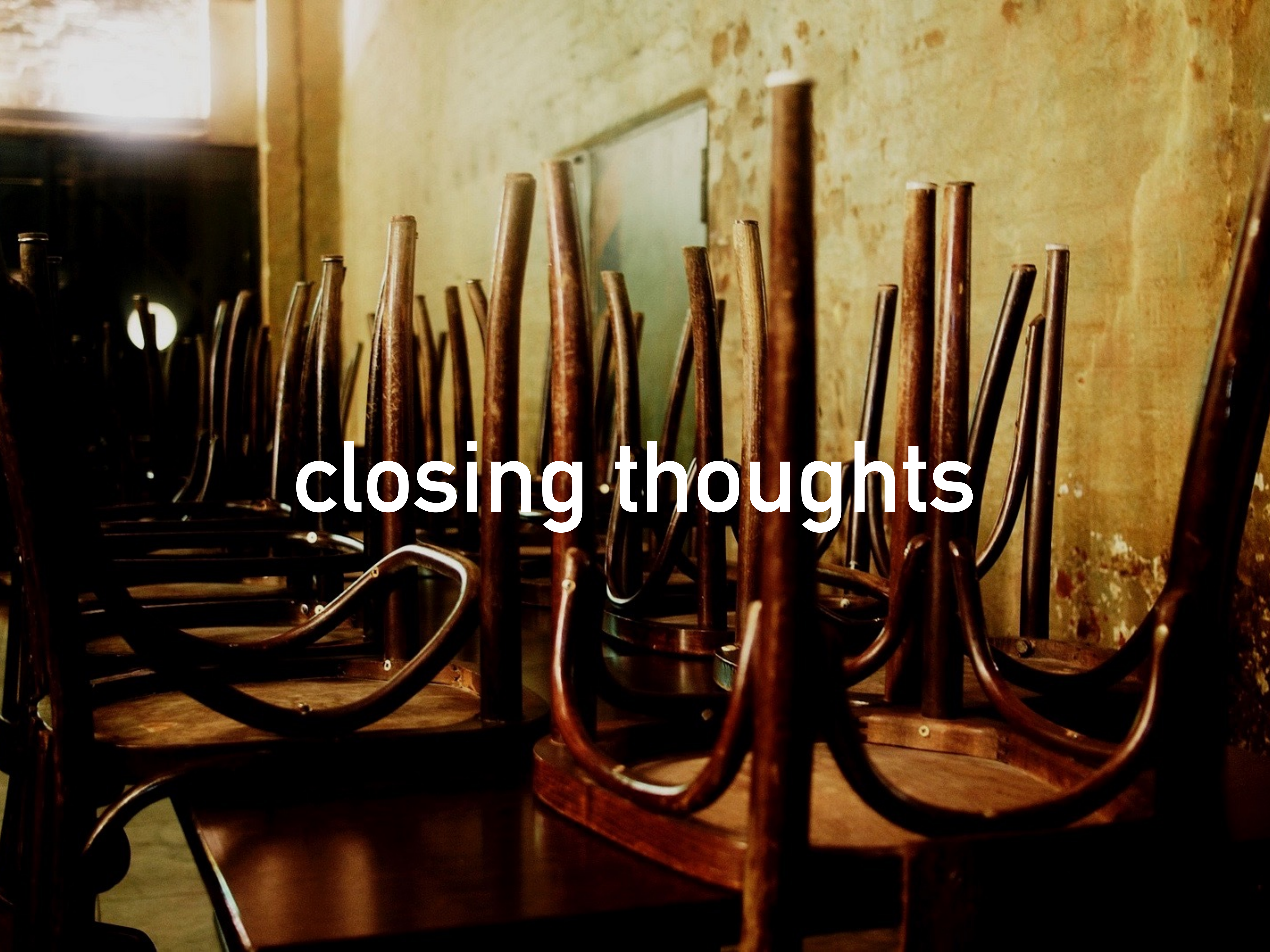
Once we drop support for IE8 and IE9, the `defer` attribute can be omitted.

The only downside is that the snippet is not a pure JavaScript solution anymore, meaning it cannot be moved or concatenated into a `.js` file. On the other hand, no one seemed to be doing that anyway; everyone just inlines the snippet into the HTML.

Ref. #1660 (comment)

paulirish commented 21 hours ago

love it. 👍



closing thoughts

don't always believe what you read on the internet benchmark your own stuff

benchmark your own stuff
every year, apparently

always bet on browsers

(and JS engines)

thanks

acknowledgements

- performance research: Steve Souders, Ilya Grigorik, Guy Podjarny, Vyacheslav Egorov, Jonathan Klein, Paul Irish, Nicholas Zakas
- photos: Christian Junker, André Hofmeister, “My aim is true”
- me: Ben Vinegar (@bentlegen)
- office hours @ 1:30 PM today