JavaScript Performance

Velocity 2012

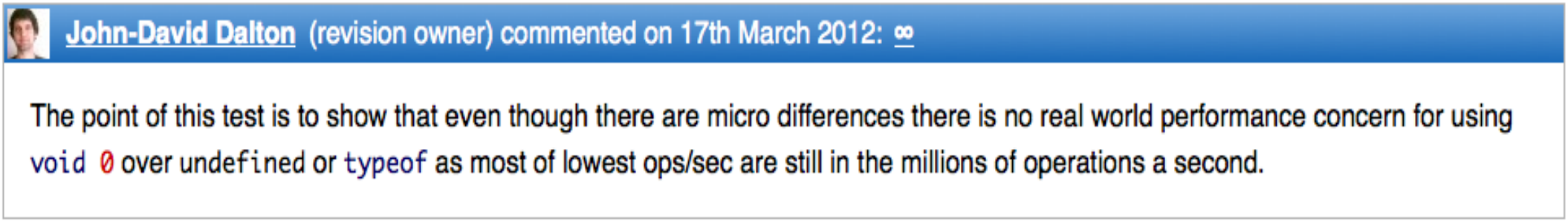
jsPerf

Crowdsourcing the results

- no dedicated test resources
- project runs in perpetuity
- real world test conditions
- aggregating results reduces bias
- new browsers show up immediately

Interpreting JSPerf results

- More is better
- Diminishing value with large numbers
  - 1M ops/sec vs 8M ops/sec is kinda meaningless ;)



- Don't forget about hardware bias

Test Time!

String concatenation is slow, use Array.join() instead.

jsperf.com/string-concatenation-vs-array-join-2/8

Results table

String concatenation is slow, use Array.join() instead.

IE <= 7

Operations which require an implicit primitive-to-object cast/conversion will be slower.

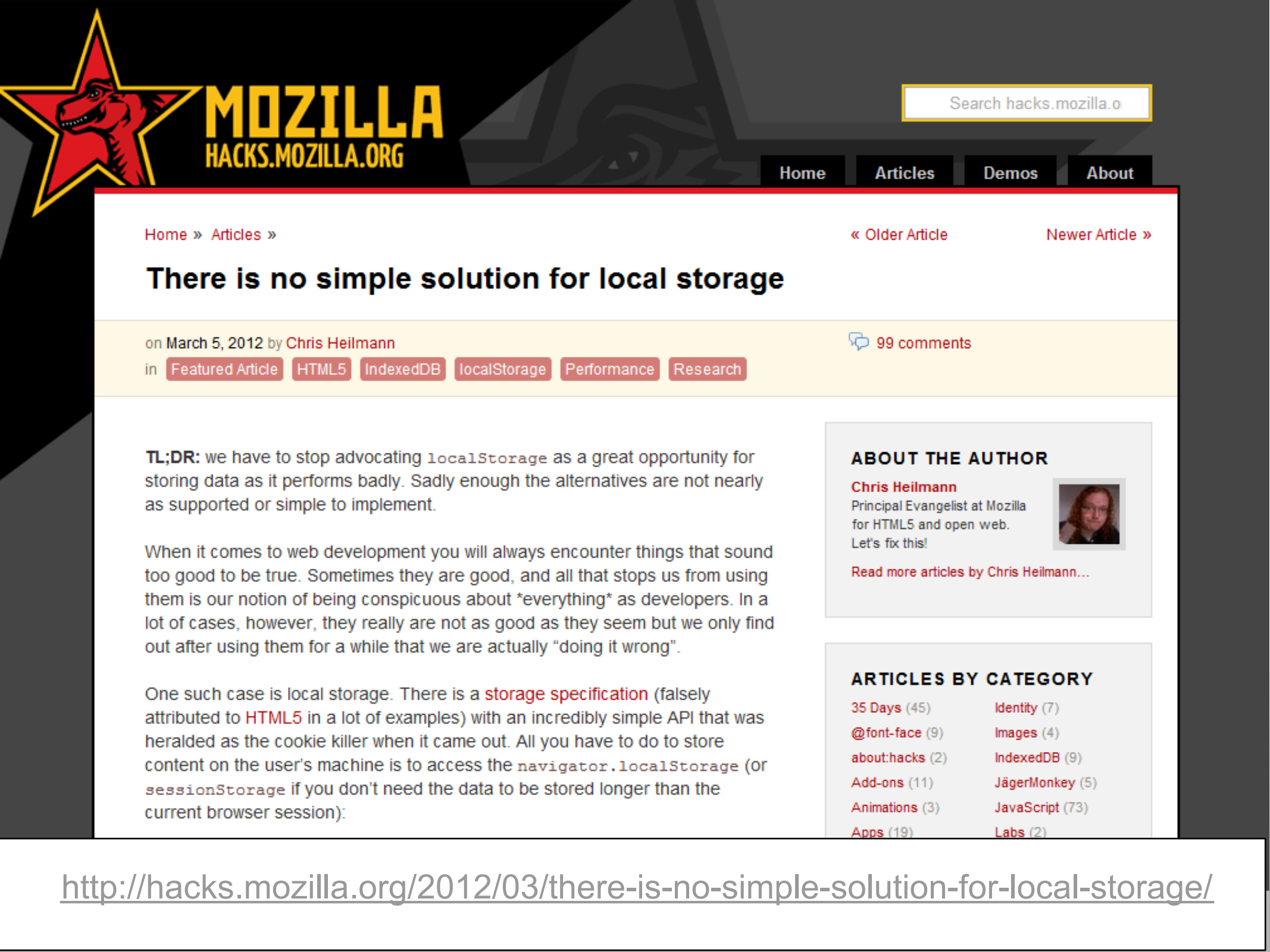
udw.geti.fi/jsperf/object-casting

Operations which require an implicit primitive-to-object cast/conversion will be slower.

localStorage is too slow to use

jsperf.com/localstorage-vs-objects/19

Results table



http://hacks.mozilla.org/2012/03/there-is-no-simple-solution-for-local-storage/

localStorage is too slow to use

ish

"`eval` is evil", or in other words, `eval` is too slow and quirky to be considered useful.

jsperf.com/more-practical-eval/3

jsperf.com/eval-kills

"`eval` is evil", or in other words, `eval` is too slow and quirky to be considered useful.

You can rely on feature detection.

(aka hashbang routing is lame,
use history.pushState!)

jquery.bassistance.de/historytest/
HTML5 History test results

danwebb.net/2011/5/28/it-is-about-the-hashbangs

You can rely on feature detection.

(aka hashbang routing is lame,
use history.pushState!)

* but you should still prefer it

Frameworks/libraries (jQuery, etc) are performance-tuned ***enough*** by js gurus.

4vk.geti.fi/jsperf/jquery-this

Frameworks/libraries (jQuery, etc) are performance-tuned ***enough*** by js gurus.

querySelectorAll() is faster than getElementsByTagName()

jsperf.com/queryselectorall-vs-getelementsbytagname

Results table

querySelectorAll() is faster than getElementsByTagName()

Converting a NodeList to an array is expensive, so you shouldn't do it.

jsperf.com/lists-of-nodes

Converting a NodeList to an array is expensive, so you shouldn't do it.

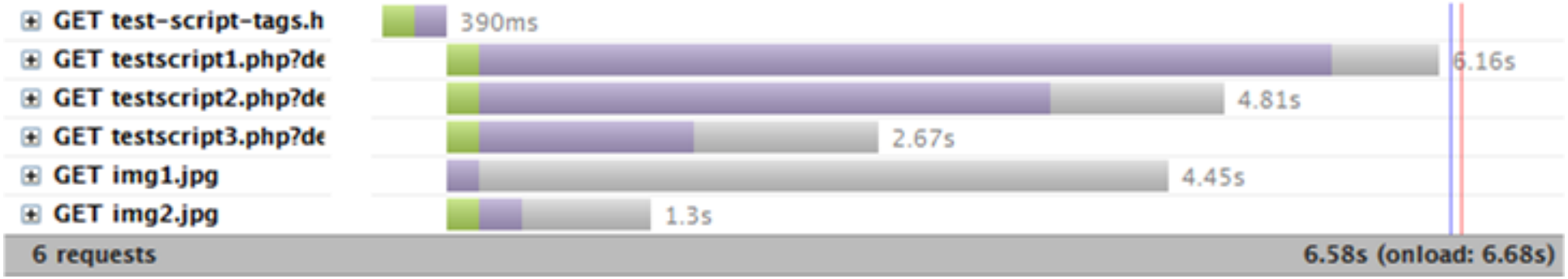
Script concatenation and/or <script defer> is ***all*** you need to load JS performantly.

(aka "Issue 28")

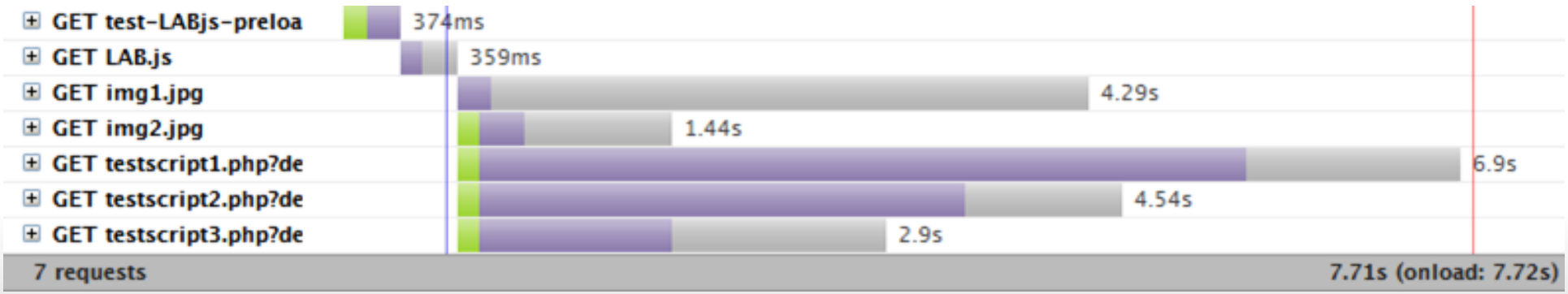
github.com/h5bp/html5-boilerplate/issues/28

<script defer>

- horribly buggy across browsers (esp IE<=9!)
  github.com/paulirish/lazyweb-requests/issues/42
- still would delay the DOM-ready event (bad!)...and btw, this is also buggy cross-browser!



<script> or <script defer>



dynamic script loading

<script "one-big-file.js">

- at some point, "big" is big enough to overcome HTTP connection overhead, meaning a second parallel request for half the file will be faster.

  NOTE: ***do*** concat your files first, ***then*** (if size >=100k) chunk into 2-3 ~equal-sized parts.

<script "one-big-file.js">

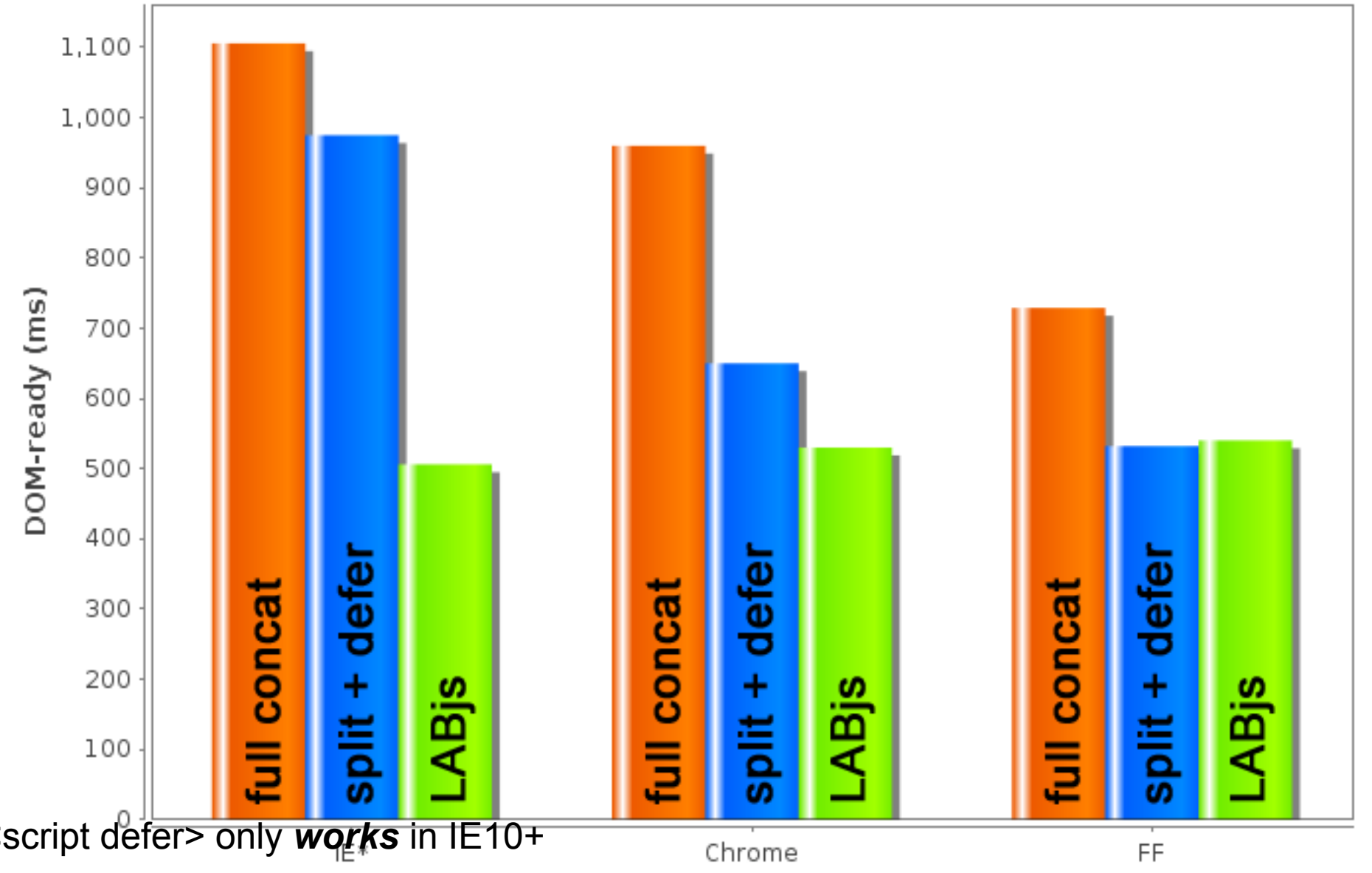
- chunking a big file into 2-3 parts, based on volatility of code (how often it changes) allows for different caching headers for each part (win!).

  NOTE: jquery.js is far more stable than your site's code.

<script "one-big-file.js">

- chunking your concat'd code file allows you to load the chunks separately, like one before load, the next 100ms after DOM-ready, etc.

  NOTE: lazy-loading is an established best-practice!



* <script defer> only ***works*** in IE10+

Experimenting with JS loading techniques on a real site (incl. jquery, jquery-ui, site code)

Script concatenation and/or <script defer> is ***all*** you need to load JS performantly.

(mostly)

Use native methods for better performance.

es5.github.com/#x15.4.4.18

jsperf.com/accessor-perf

jsperf.com/bind-vs-custom

jsperf.com/for-vs-array-foreach

Use native methods for better performance.

Status: 

The string method `indexOf` is much faster than using a regular expression to search a string.

imb.geti.fi/jsperf/regex/indexof

The string method `indexOf` is much faster than using a regular expression to search a string.