Even Faster Web Sites

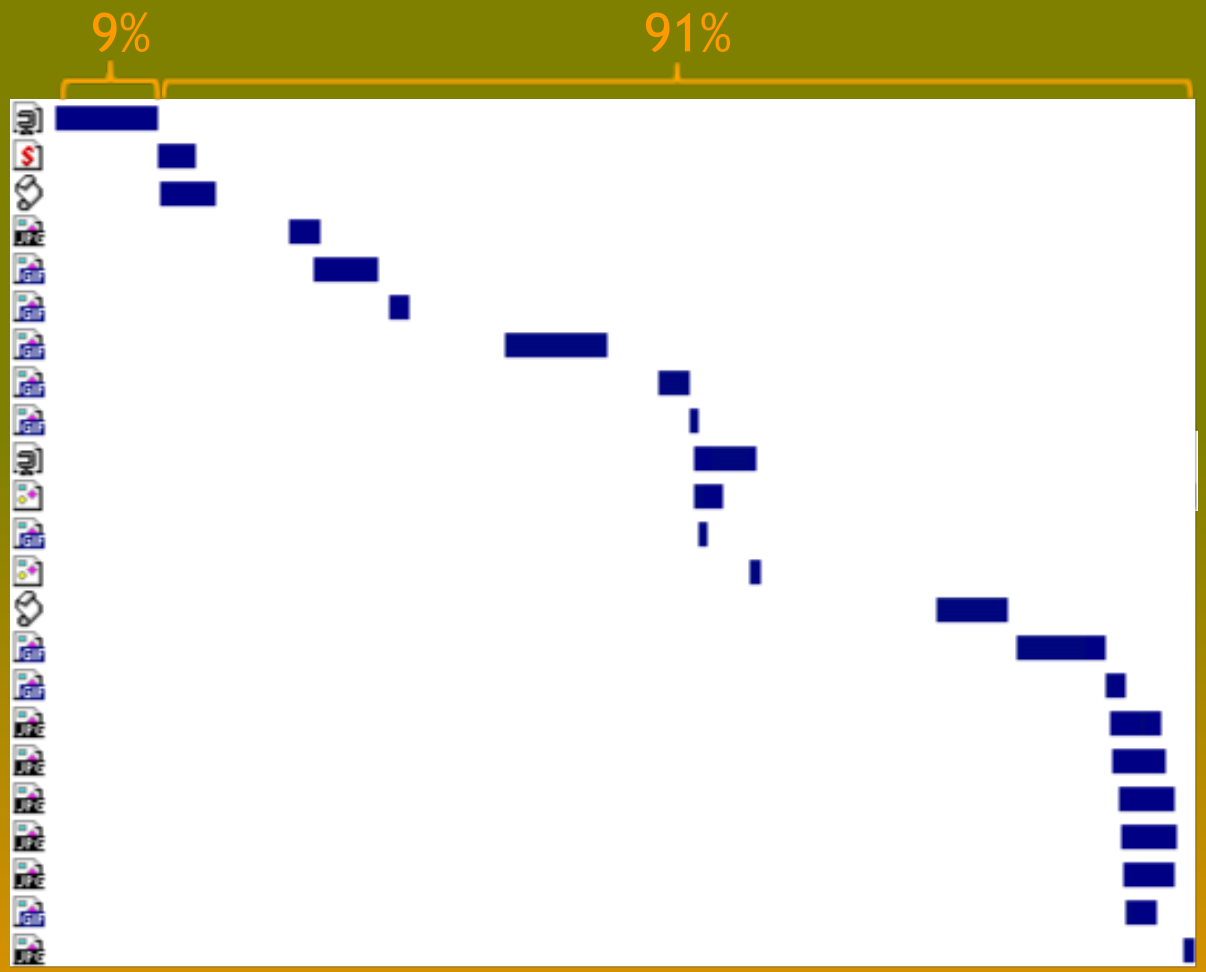
Steve Souders

souders@google.com

http://stevesouders.com/docs/googleio-20090527.ppt

Disclaimer: This content does not necessarily reflect

the importance of frontend performance



iGoogle, empty cache

time spent on the frontend

	Empty Cache	Primed Cache
www.aol.com	97%	97%
www.ebay.com	95%	81%
www.facebook.com	95%	81%
www.google.com/search	47%	0%
search.live.com/results	67%	0%
www.msn.com	98%	94%
www.myspace.com	98%	98%
en.wikipedia.org/wiki	94%	91%
www.yahoo.com	97%	96%
www.youtube.com	98%	97%

April 2008

The Performance Golden Rule

80-90% of the end-user response time is spent on the frontend. Start there.

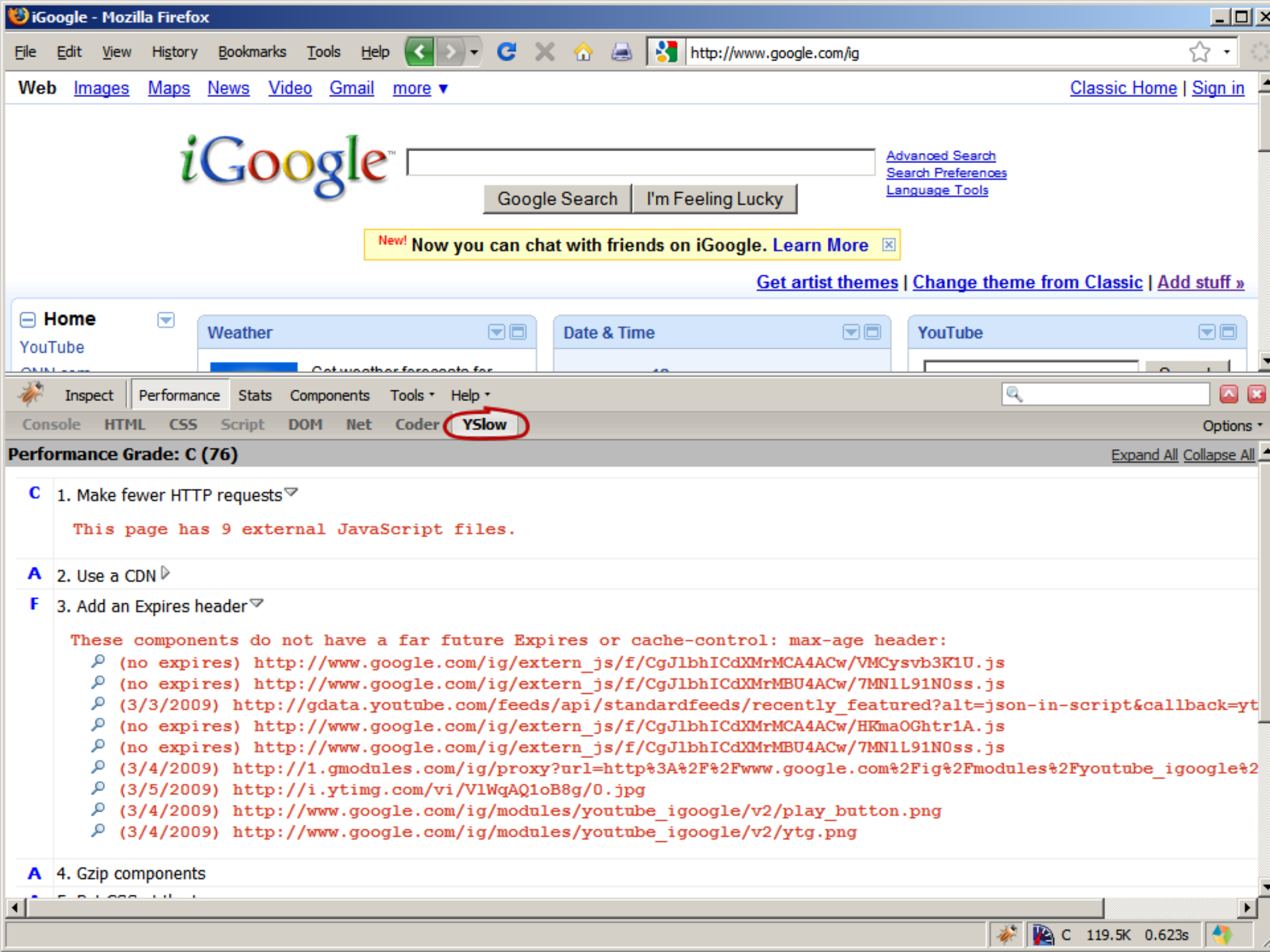
greater potential for improvement

simpler

proven to work

14 RULES

1. MAKE FEWER HTTP REQUESTS
2. USE A CDN
3. ADD AN EXPIRES HEADER
4. GZIP COMPONENTS
5. PUT STYLESHEETS AT THE TOP
6. PUT SCRIPTS AT THE BOTTOM
7. AVOID CSS EXPRESSIONS
8. MAKE JS AND CSS EXTERNAL
9. REDUCE DNS LOOKUPS
10. MINIFY JS
11. AVOID REDIRECTS
12. REMOVE DUPLICATE SCRIPTS
13. CONFIGURE ETAGS
14. MAKE AJAX CACHEABLE


iGoogle - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.google.com/ig
Web Images Maps News Video Gmail more ▾
Classic Home | Sign in
iGoogle
Google Search
I'm Feeling Lucky
Advanced Search
Search Preferences
Language Tools
New! Now you can chat with friends on iGoogle. Learn More
Get artist themes | Change theme from Classic | Add stuff »
Home
YouTube
Weather
Date & Time
YouTube
Inspect Performance Stats Components Tools Help
Console HTML CSS Script DOM Net Coder YSlow
Options
Performance Grade: C (76)
Expand All Collapse All
C 1. Make fewer HTTP requests
This page has 9 external JavaScript files.
A 2. Use a CDN
F 3. Add an Expires header
These components do not have a far future Expires or cache-control: max-age header:
(no expires) http://www.google.com/ig/extern_js/f/CgJlbhICdXMrMCA4ACw/VMCysvb3K1U.js
(no expires) http://www.google.com/ig/extern_js/f/CgJlbhICdXMrMBU4ACw/7MNlL91NOss.js
(3/3/2009) http://gdata.youtube.com/feeds/api/standardfeeds/recently_featured?alt=json-in-script&callback=yt
(no expires) http://www.google.com/ig/extern_js/f/CgJlbhICdXMrMCA4ACw/HKmaOGhtr1A.js
(no expires) http://www.google.com/ig/extern_js/f/CgJlbhICdXMrMBU4ACw/7MNlL91NOss.js
(3/4/2009) http://1.gmodules.com/ig/proxy?url=http%3A%2F%2Fwww.google.com%2Fig%2Fmodules%2Fyoutube_igoogle%2
(3/5/2009) http://i.ytimg.com/vi/VlWqAQ1oB8g/0.jpg
(3/4/2009) http://www.google.com/ig/modules/youtube_igoogle/v2/play_button.png
(3/4/2009) http://www.google.com/ig/modules/youtube_igoogle/v2/ytg.png
A 4. Gzip components
C 119.5K 0.623s

O'REILLY

Velocity

Web Performance and Operations Conference

15% discount code: ve109cmb

JUNE 22–24 | 2009
SAN JOSE | CA

<FAST> <SCALABLE> <EFFICIENT> <AVAILABLE>


Jeremy Zawodny
craigslist.org


Marissa Mayer
Google


Douglas Crockford
Yahoo! Inc.


Jonathan Heiliger
Facebook

Join us at Velocity, the O'Reilly Web Performance and Operations Conference

June 22-24, 2009 at The Fairmont San Jose

Web companies of all sizes face many of the same challenges: sites must be faster, infrastructure must scale, and everything must be available to customers at all times, no matter what. Velocity is the place to obtain the crucial skills and knowledge to build successful websites that are fast, scalable, resilient, and highly available. It's a unique opportunity to learn from your peers, exchange ideas with experts, and share best practices that will help maximize your company's efficiency, performance, and responsiveness.

A full day of workshops and two days of sessions, offered in dedicated Web Performance and Operations tracks, will help you and your

REGISTER NOW

VELOCITY CONFERENCE KUDOS

"Velocity '08 was the first conference dedicated to Internet and Web operators. I was extremely impressed by the technical depth of the tracks, presenters, and attendees. Velocity is now one of my three 'must go' yearly events even though it is only in its second year of operation. I'm very excited for this year's event and look forward to further engaging With other web scale players."
—*Randy Bias, GoGrid*

VELOCITY CONFERENCE BROCHURE

Premium Diamond Sponsor
keynote

Diamond Sponsor
Google

Platinum Sponsor
shopzilla

Gold Sponsors
aptimize

facebook

Home
Getting Started
Degrees & Certificates
Courses & Seminars
Resources
Calendar
Announcements
Student Spotlights
Member Companies
About Us
Help/Contact Us

Search:

Site
Courses
Degrees/Certificates

Join Our Mailing List

High Performance Web Sites

XCS193H

Delivery Options: Online

Course Description

People love fast web sites, but up until now developers have been focusing on the wrong area. Backend (web server, database, etc.) performance is important for reducing hardware costs and improving efficiency, but for most pages 80% of the load time is spent on the frontend (HTML, CSS, JavaScript, images, iframes, and others). Steve Souders, who works at Google on web performance, will introduce best practices for making web pages faster, provide case studies from top web sites, and introduce the tools he uses for researching performance. In addition to learning how to improve web performance, students will gain an understanding of the fundamentals of how the Internet works including DNS, HTTP, and browsers.

Topics Include

- Backend performance.
- Best practices for making web pages faster.
- Fundamentals of how the Internet works.

Course Notes

Videotaped lectures and course materials for this course are the same as those for the CS193H graduate course. The on-demand format for XCS193H makes it possible to enroll in the course at anytime. The course is self-paced and includes an exam to demonstrate an understanding of the course content. Students who successfully complete the exam may request Continuing Education Units. Those students seeking Stanford University graduate credit need to enroll in CS193H when it is next taught at Stanford and complete all course requirements as part of the class cohort.

Course Preview

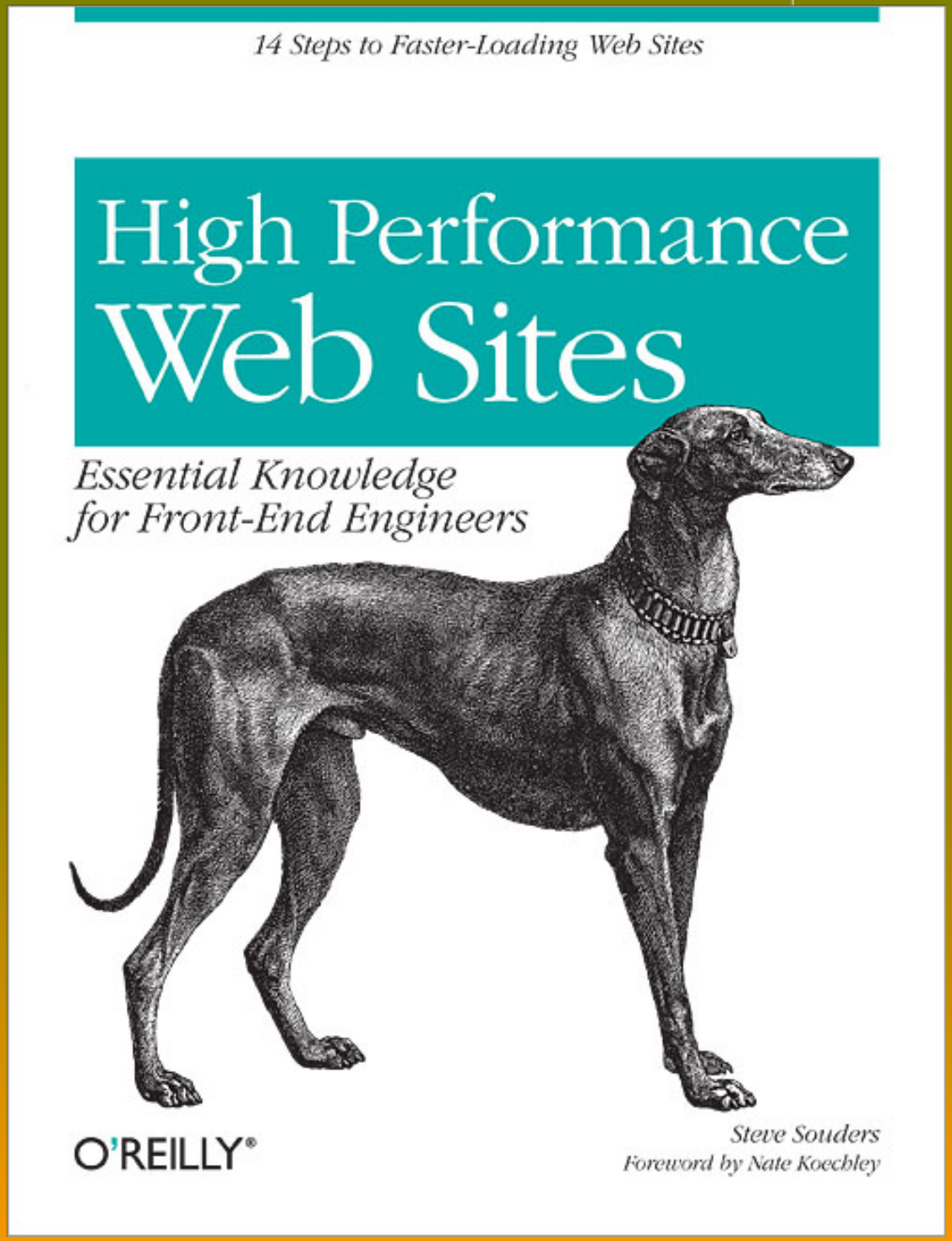
We also highly recommend watching the course preview to ensure you have the requisite background and understand the scope of material covered.

Prerequisite(s)

- Javascript
- CSS
- HTML

Recommended

Professor Souders book, High Performance Web Sites: Essential Knowledge for Front-End Engineers is referenced throughout the course.


14 Steps to Faster-Loading Web Sites
High Performance Web Sites
Essential Knowledge for Front-End Engineers
O'REILLY®
Steve Souders
Foreword by Nate Koechley

Sept 2007

Essential Knowlege for Frontend Engineers

Even Faster Web Sites

O'REILLY®

Steve Souders

June 2009

Even Faster Web Sites

Splitting the initial payload
Loading scripts without blocking
Coupling asynchronous scripts
Positioning inline scripts
Sharding dominant domains
Flushing the document early
Using iframes sparingly
Simplifying CSS Selectors

Understanding Ajax performance.....*Doug Crockford*
Creating responsive web apps.....*Ben Galbraith, Dion Almaer*
Writing efficient JavaScript.....*Nicholas Zakas*
Scaling with Comet.....*Dylan Schiemann*
Going beyond gzipping.....*Tony Gentilcore*
Optimizing images.....*Stoyan Stefanov, Nicole Sullivan*

Sharding Dominant Domains

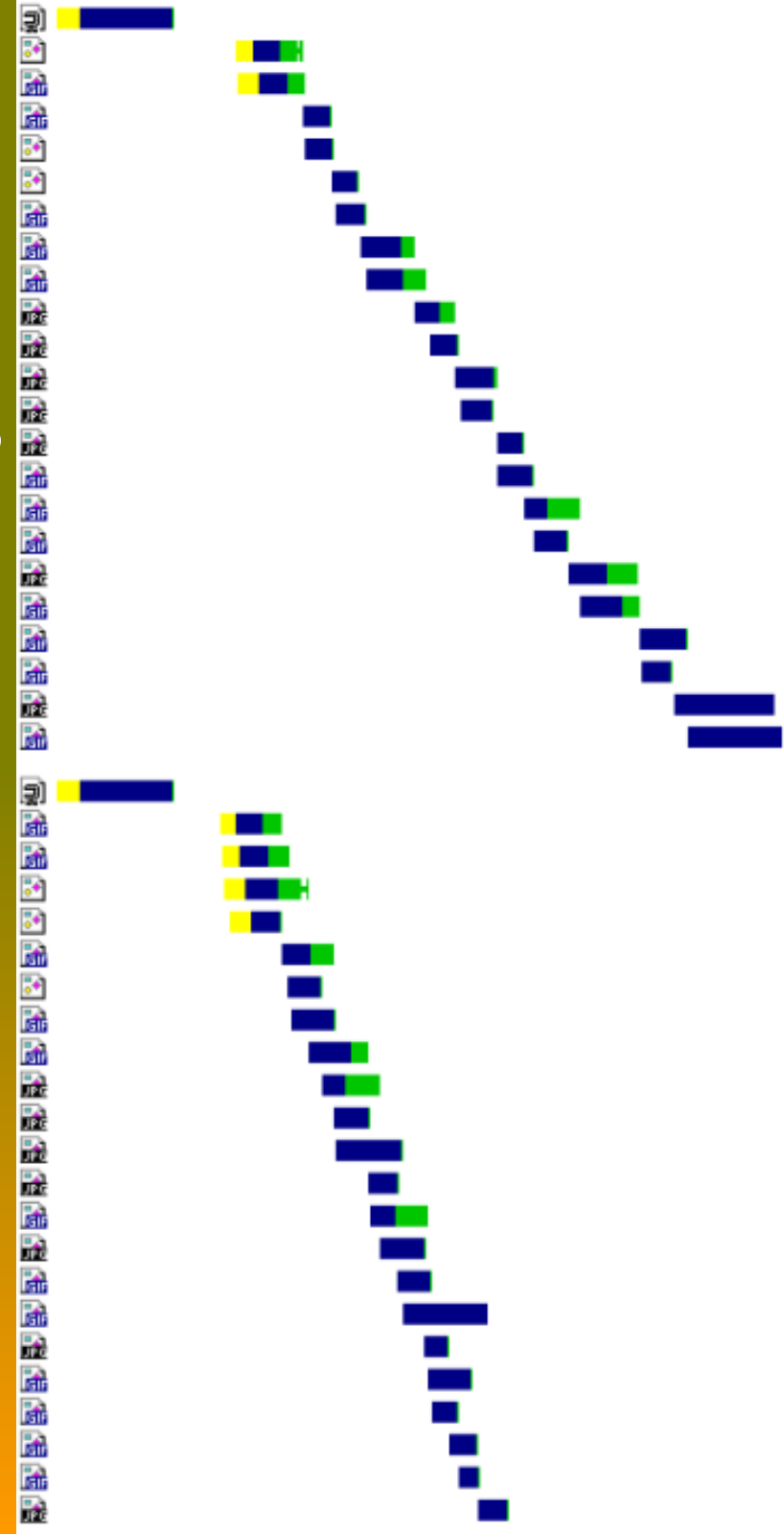
HTTP/1.1 and older browsers

- 2 connections per server
- based on name, not IP
- includes IE 6,7

"domain sharding"

intentionally splitting resources across multiple domains

makes pages load faster



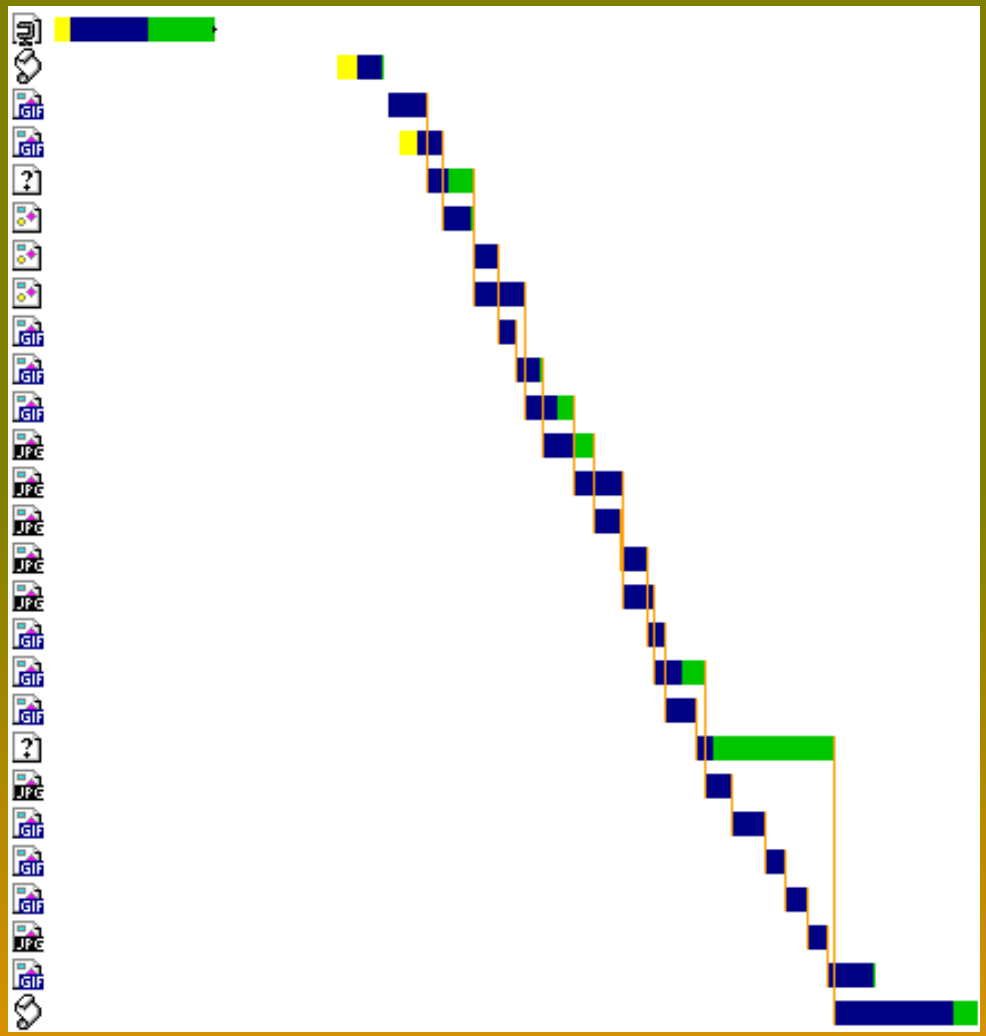
Sharding Dominant Domains

but Rule 9 says "Reduce DNS lookups"?!

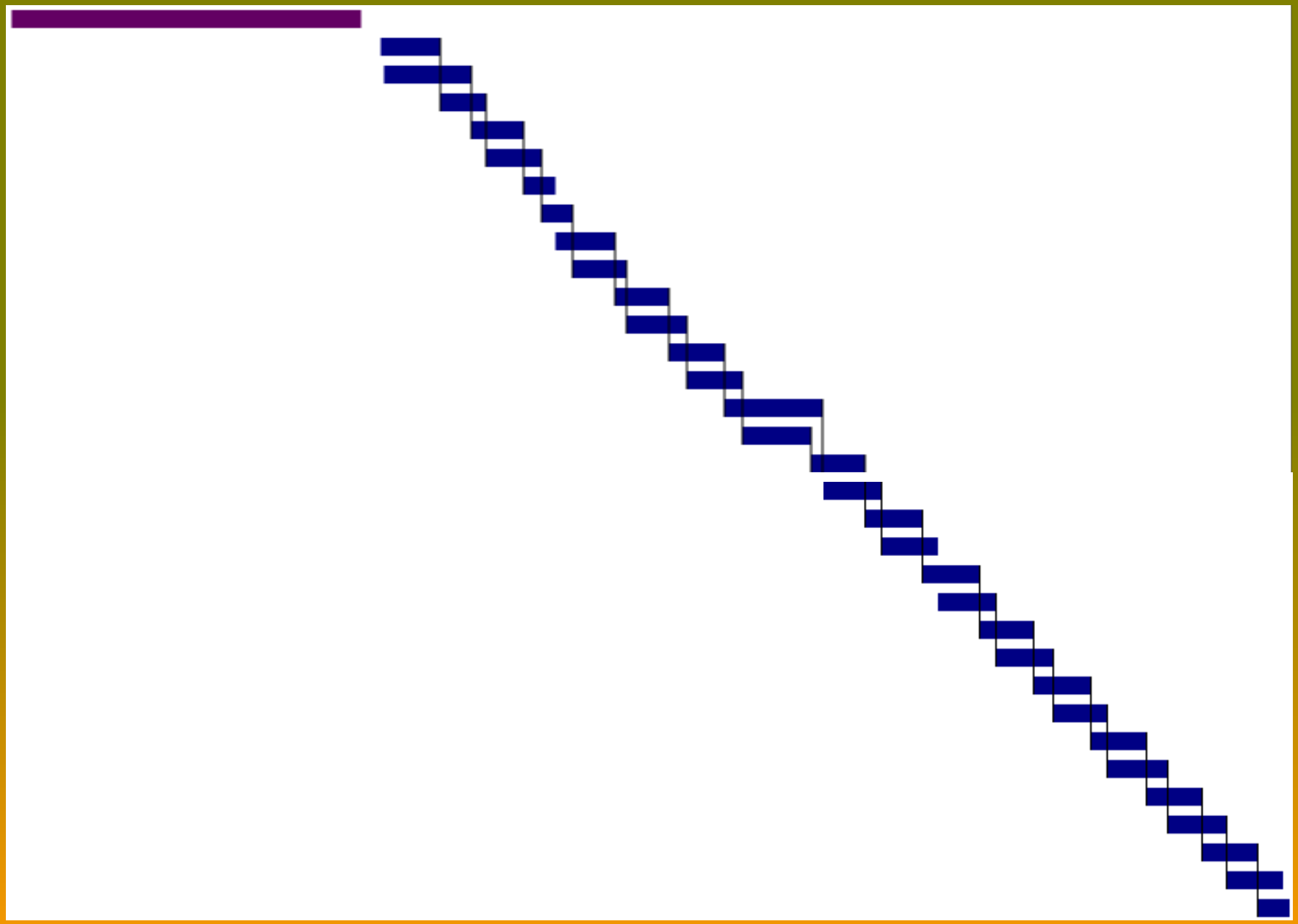
remove DNS lookups that aren't heavily used
split domains that are on the critical path

how find "critical path"?

www.yahoo.com



news.google.com



downgrading to HTTP/1.0

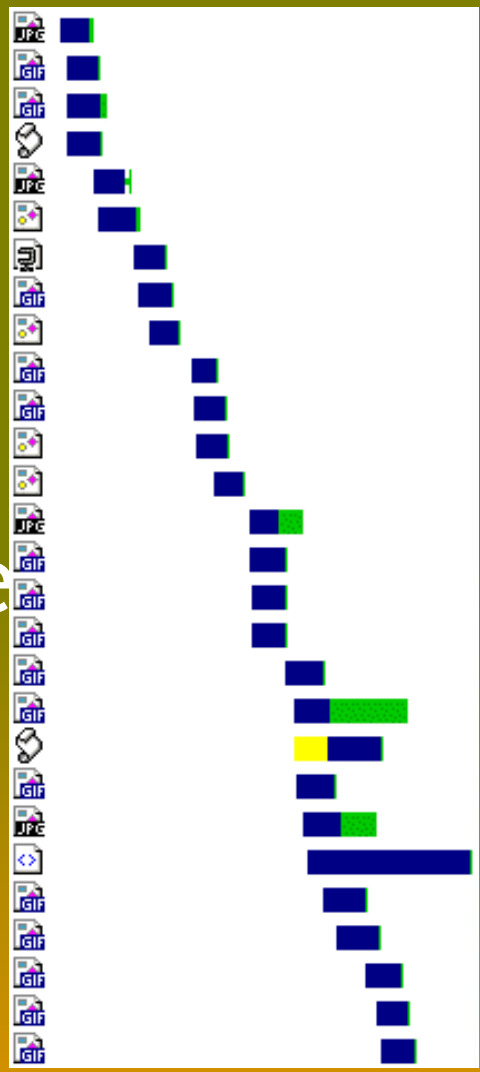
HTTP/1.1 – 2 connections/server

HTTP/1.0 – 4 (IE 6,7), 8 (FF2)

HTTP/1.1 has fewer connections because persistent connections are on by default

best for static content

example: *http://wikipedia.org/*



connections per server by browser

	HTTP/1.1	HTTP/1.0
IE 6,7	2	4
IE 8	6	6
Firefox 1.5, 2	2	8
Firefox 3	6	6
Safari 3,4	4	4
Chrome	6	6
Opera 9	4	4

newer browsers open more connections

http://www.stevesouders.com/blog/2008/03/20/roundup-on-parallel-connections/

how many domains?

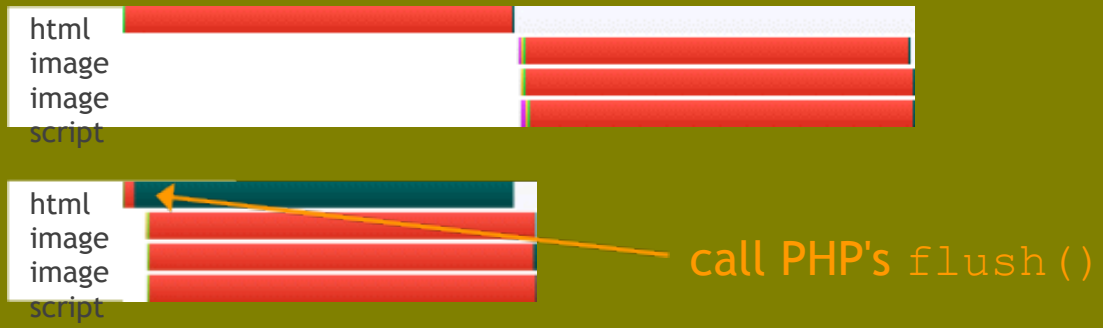
http://yuiblog.com/blog/2007/04/11/performance-research-part-4/

2-4 is optimal

after 4 domains, response time degrades

- more DNS lookups
- thrashing on client

flushing the document early



gotchas:

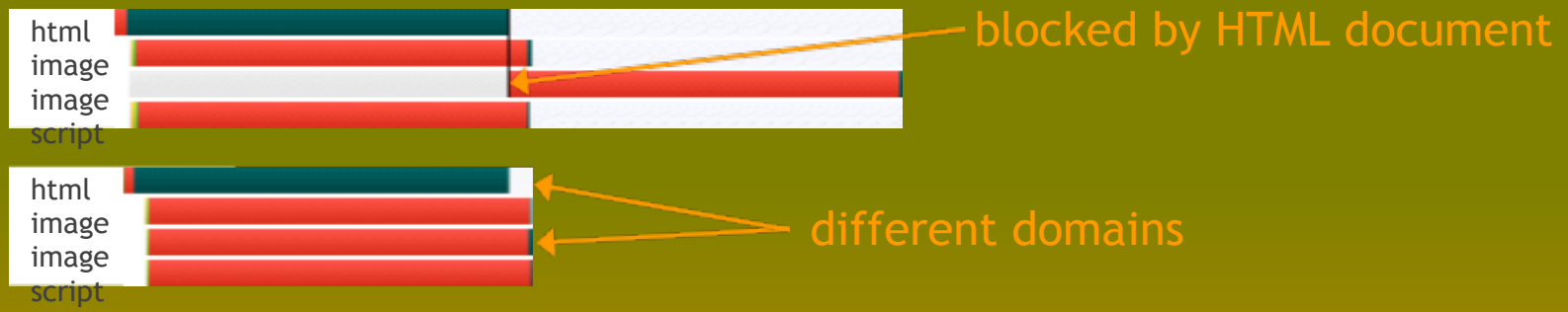
- PHP output_buffering - `ob_flush()`
- Transfer-Encoding: chunked
- gzip - Apache's DeflateBufferSize before 2.2.8
- proxies and anti-virus software
- browsers - Safari (1K), Chrome (2K)

other languages:

- `$|` or `FileHandle autoflush` (Perl), `flush` (Python), `ios.flush` (Ruby)

flushing and domain blocking

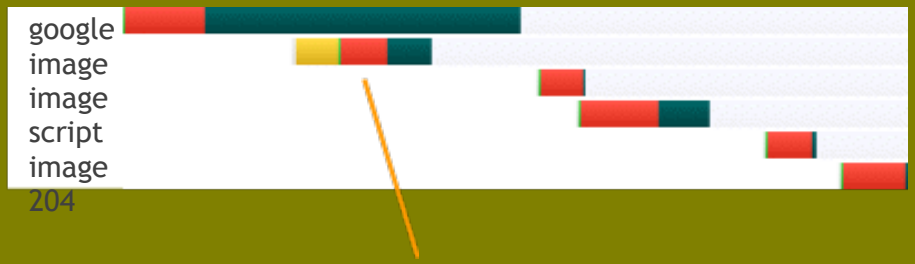
you might need to move flushed resources to a domain different from the HTML doc



case study: Google search

successful flushing

Google Search



http://www.google.com/images/nav_logo4.png

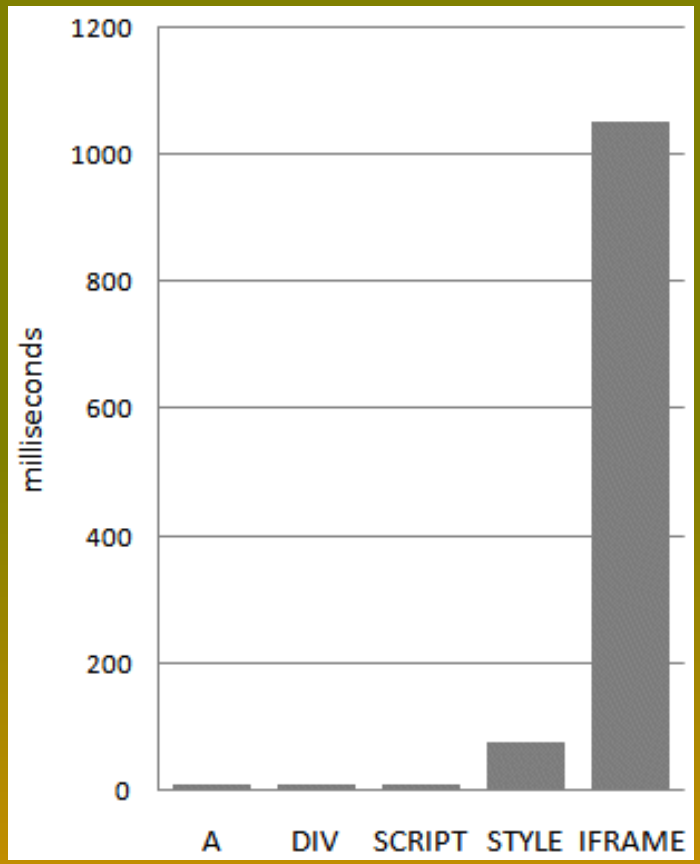
external resource downloaded early

content visible to the user

iframes: most expensive DOM element

load 100 *empty* elements of each type

tested in all major browsers[1]



[1] IE 6, 7, 8; FF 2, 3.0, 3.1b2; Safari 3.2, 4; Opera 9.63, 10; Chrome 1.0, 2.0

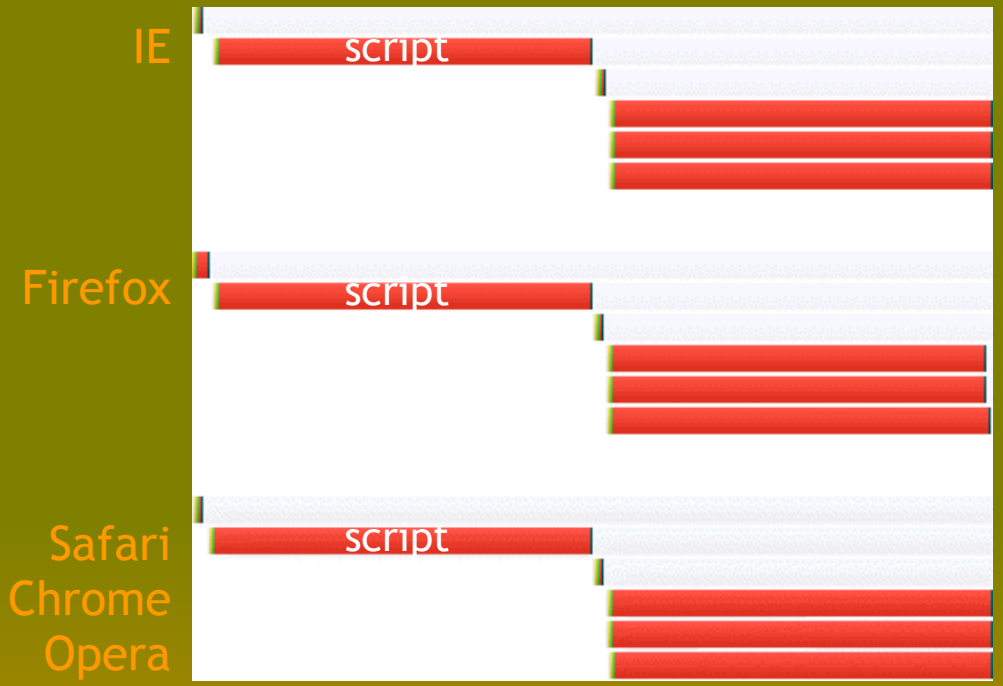
iframes block onload

parent's onload doesn't fire until iframe and all its components are downloaded

workaround for Safari and Chrome: set iframe src in JavaScript

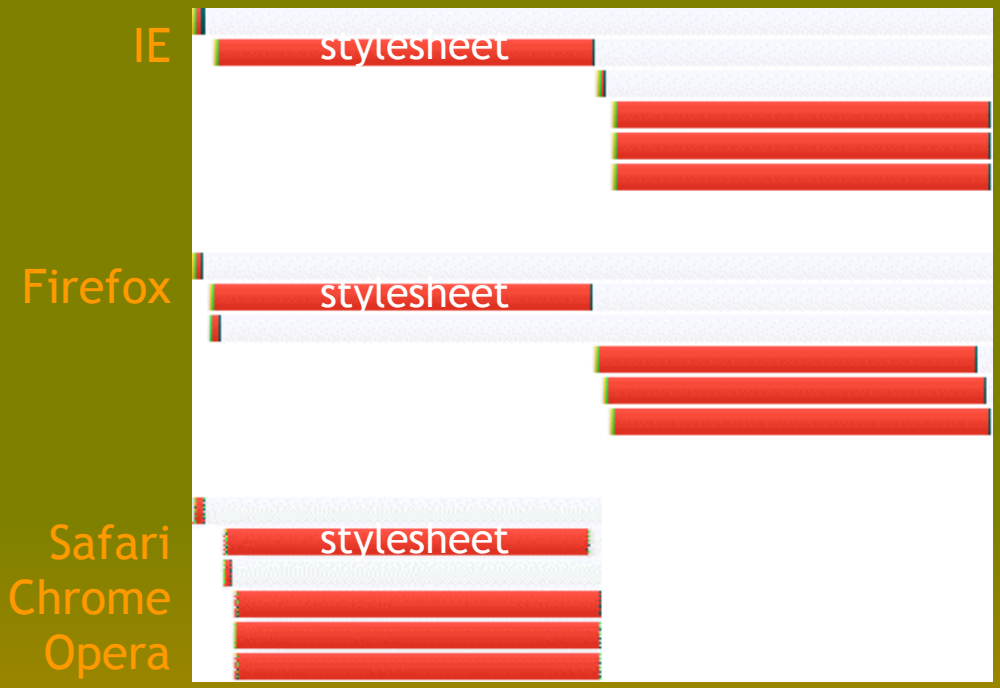
```
<iframe id=iframe1 src=""></iframe>
<script type="text/javascript">
document.getElementById('iframe1').src="url";
</script>
```

scripts block iframe



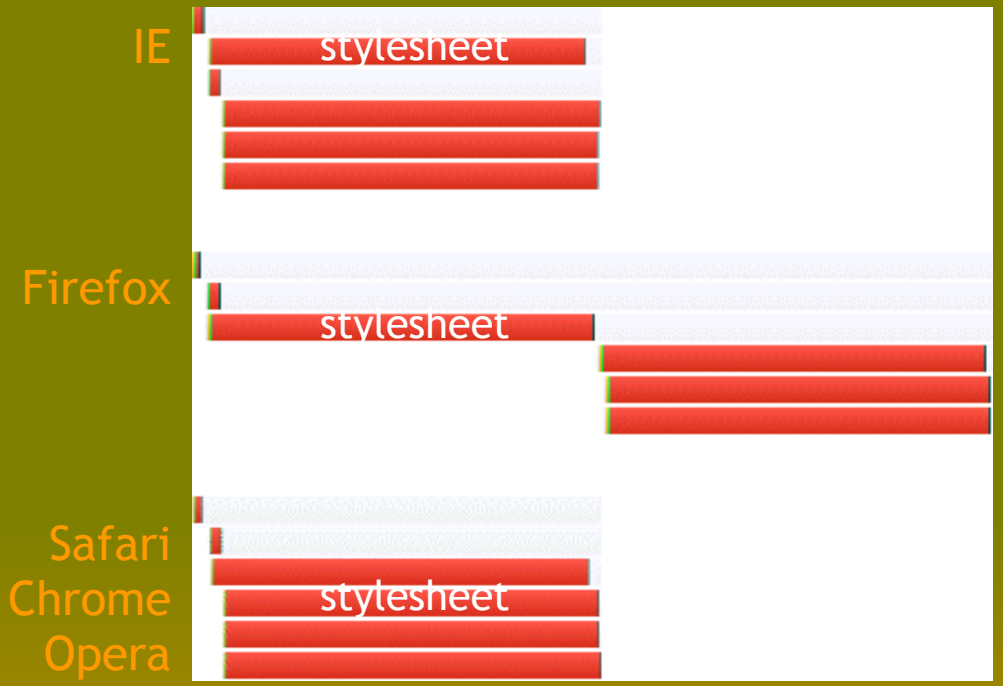
no surprise – scripts in the parent block the iframe from loading

stylesheets block iframe (IE, FF)



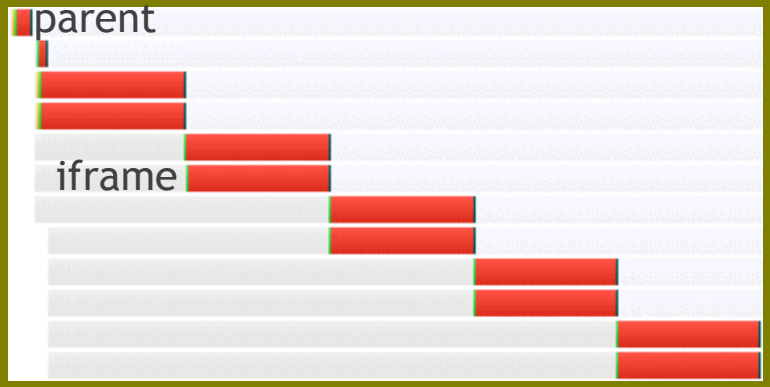
surprise – stylesheets in the parent block the iframe or its resources in IE & Firefox

stylesheets after iframe still block (FF)



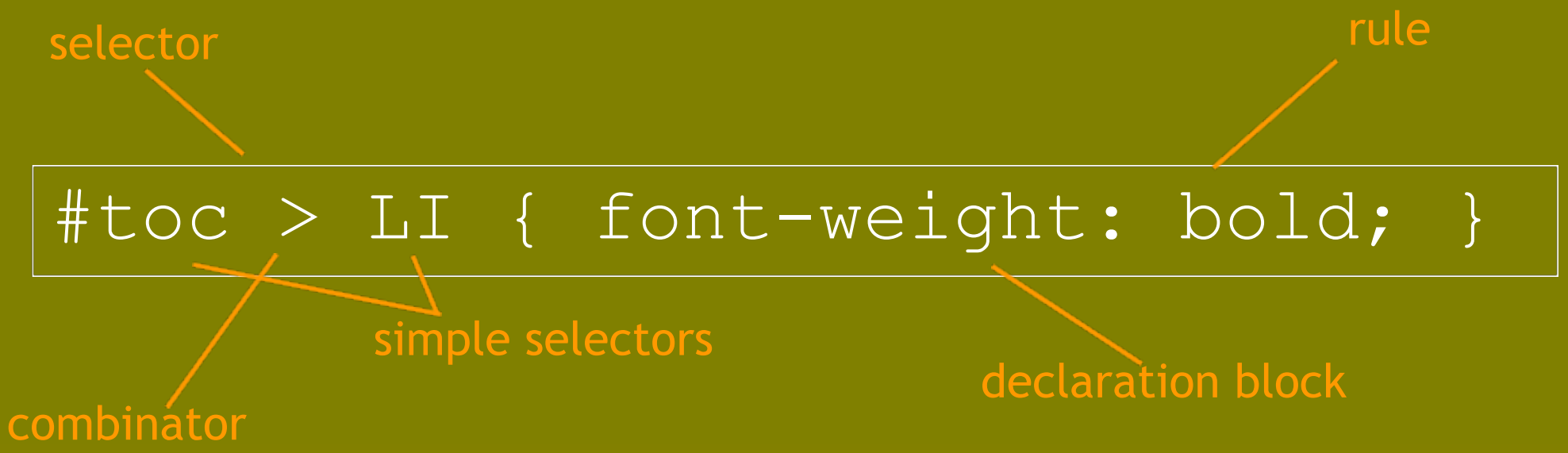
surprise – even moving the stylesheet *after* the iframe still causes the iframe's resources to be blocked in Firefox

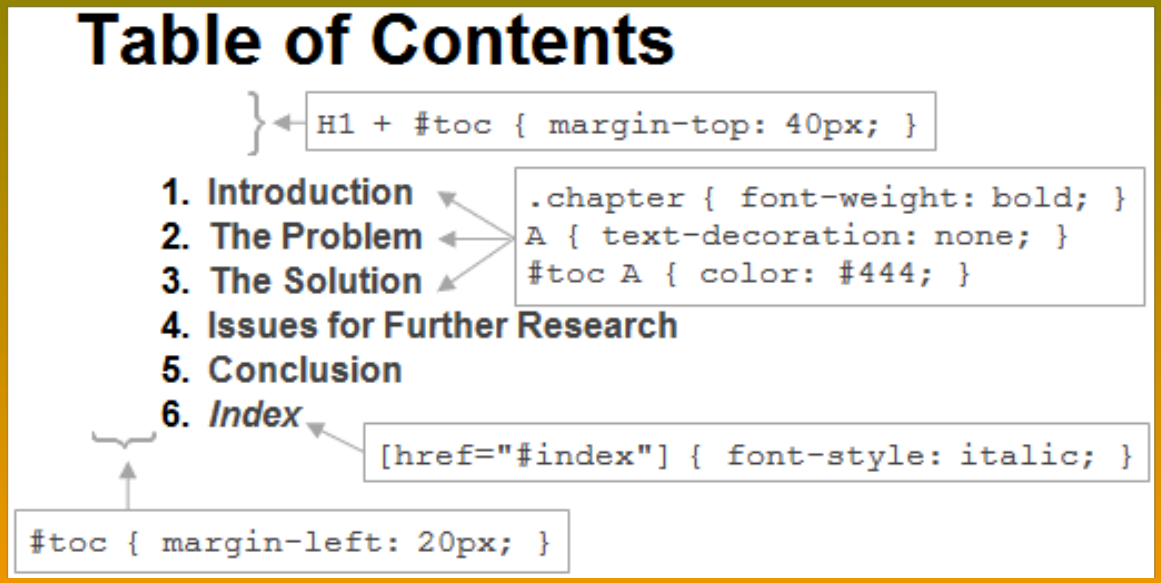
iframes: no free connections



iframe shares connection pool with parent
(here – 2 connections per server in IE 7)

Simplifying CSS Selectors





types of CSS selectors

ID selectors

```
#toc { margin-left: 20px; }
```

element whose ID attribute has the value "toc"

class selectors

```
.chapter { font-weight: bold; }
```

elements with class=chapter

type selectors

```
A { text-decoration: none; }
```

all A elements in the document tree

http://www.w3.org/TR/CSS2/selector.html

types of CSS selectors

adjacent sibling selectors

```
H1 + #toc { margin-top: 40px; }
```

an element with ID=toc that immediately follows an H1

child selectors

```
#toc > LI { font-weight: bold; }
```

all LI elements whose parent has id="toc"

descendant selectors

```
#toc A { color: #444; }
```

all A elements that have id="toc" as an ancestor

types of CSS selectors

universal selectors

```
* { font-family: Arial; }
```

all elements

attribute selectors

```
[href="#index"] { font-style: italic; }
```

all elements where the href attribute is "#index"

psuedo classes and elements

```
A:hover { text-decoration: underline; }
```

non-DOM behavior

others: `:visited`, `:link`, `:active`, `:focus`, `:first-child`, `:before`, `:after`

writing efficient CSS

https://developer.mozilla.org/en/Writing_Efficient_CSS

"The style system matches a rule by starting with the rightmost selector and moving to the left through the rule's selectors. As long as your little subtree continues to check out, the style system will continue moving to the left until it either matches the rule or bails out because of a mismatch."

```
#toc > LI { font-weight: bold; }
```

find every LI whose parent is id="toc"

```
#toc A { color: #444; }
```

find every A and climb its ancestors until id="toc" or DOM root (!) is found

writing efficient CSS

1. avoid universal selectors
2. don't qualify ID selectors
   - bad: `DIV #navbar {}`
   - good: `#navbar {}`
3. don't qualify class selectors
   - bad: `LI .tight {}`
   - good: `.li-tight {}`
4. make rules as specific as possible
   - bad: `#navbar A {}`
   - good: `.a-navbar {}`

https://developer.mozilla.org/en/Writing_Efficient_CSS

writing efficient CSS

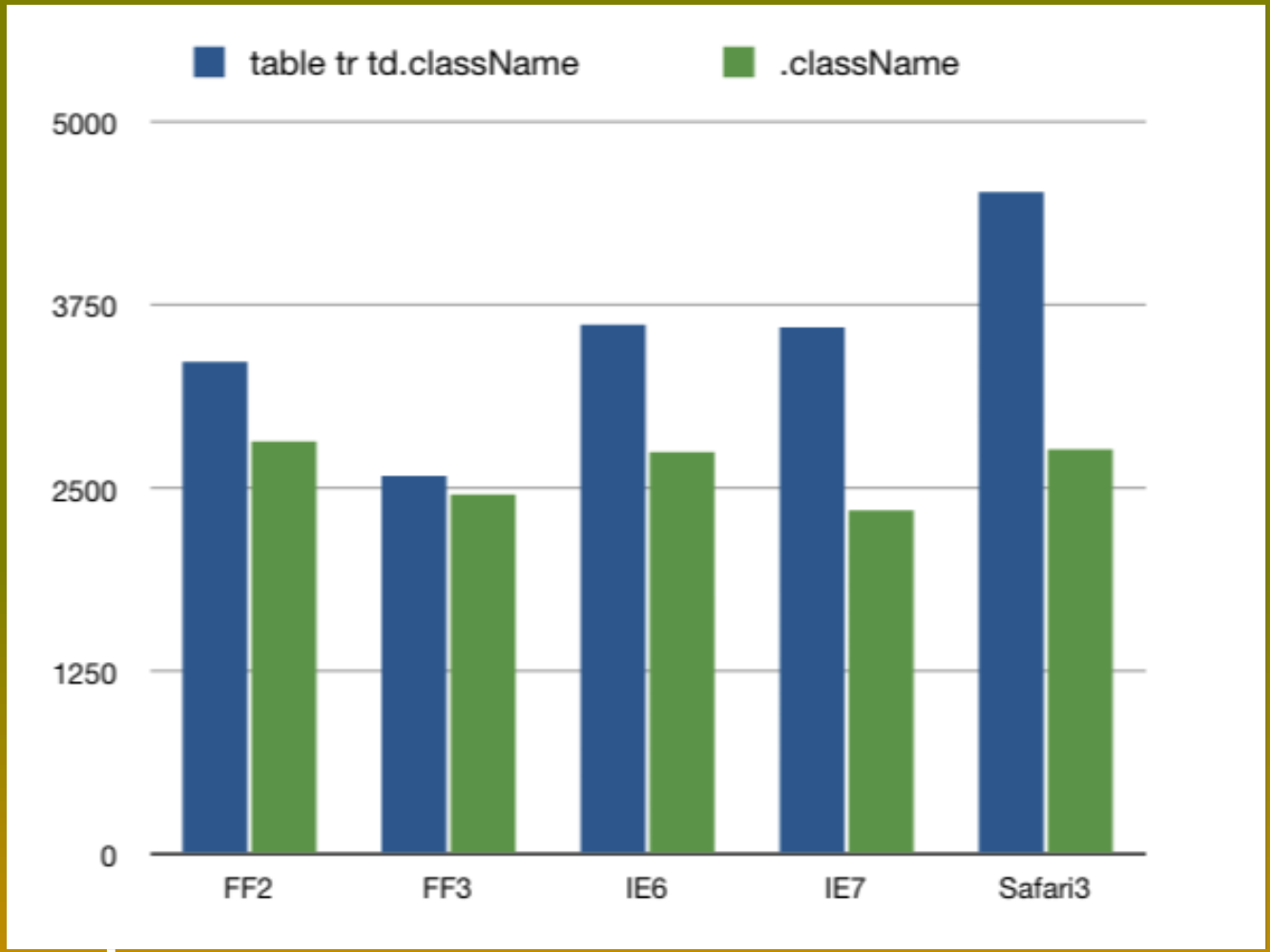
1. avoid descendant selectors
   - bad: `UL LI A {}`
   - better: `UL > LI > A {}`
2. avoid tag-child selectors
   - bad: `UL > LI > A {}`
   - best: `.li-anchor {}`
3. be wary of child selectors
4. rely on inheritance
   - *http://www.w3.org/TR/CSS21/propidx.html*

https://developer.mozilla.org/en/Writing_Efficient_CSS

David Hyatt

4/21/2000

Testing CSS Performance



20K TD elements

http://jon.sykes.me/152/testing-css-performance-pt-2

testing massive CSS

20K `A` elements
no style: control
tag:

```
A {}
```

class:

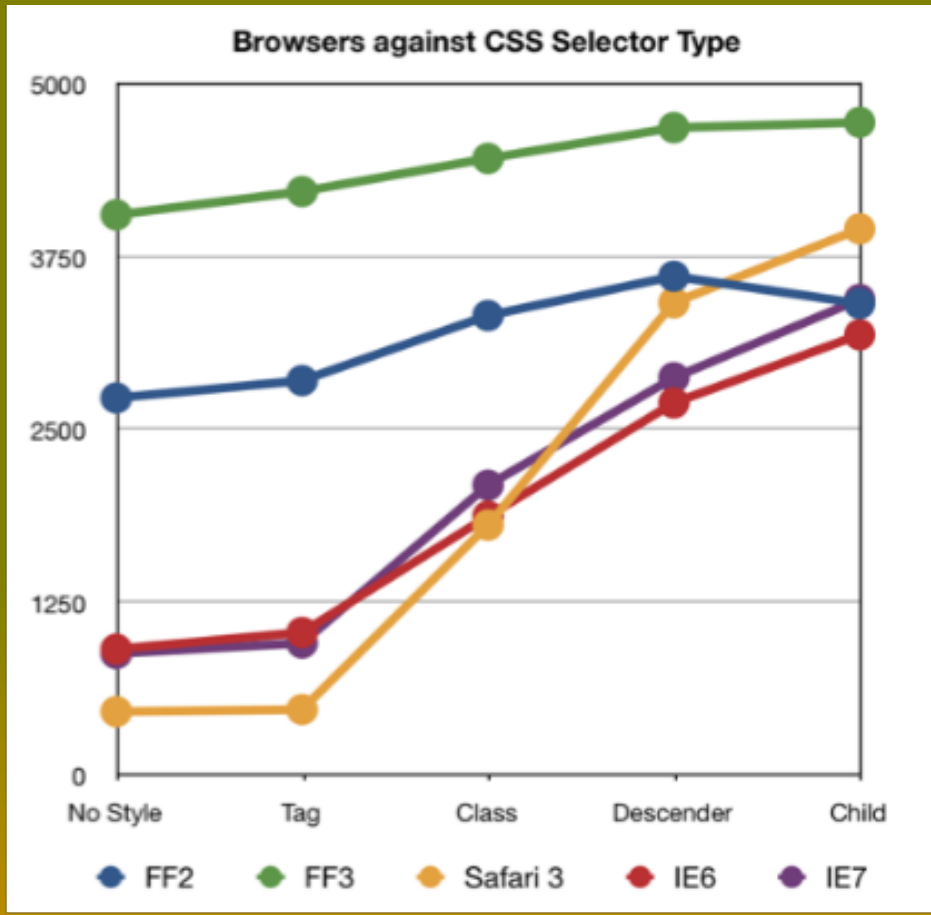
```
.a00001 {}
.a20000 {}
```

descender:

```
DIV DIV DIV P A.a00001 {}
```

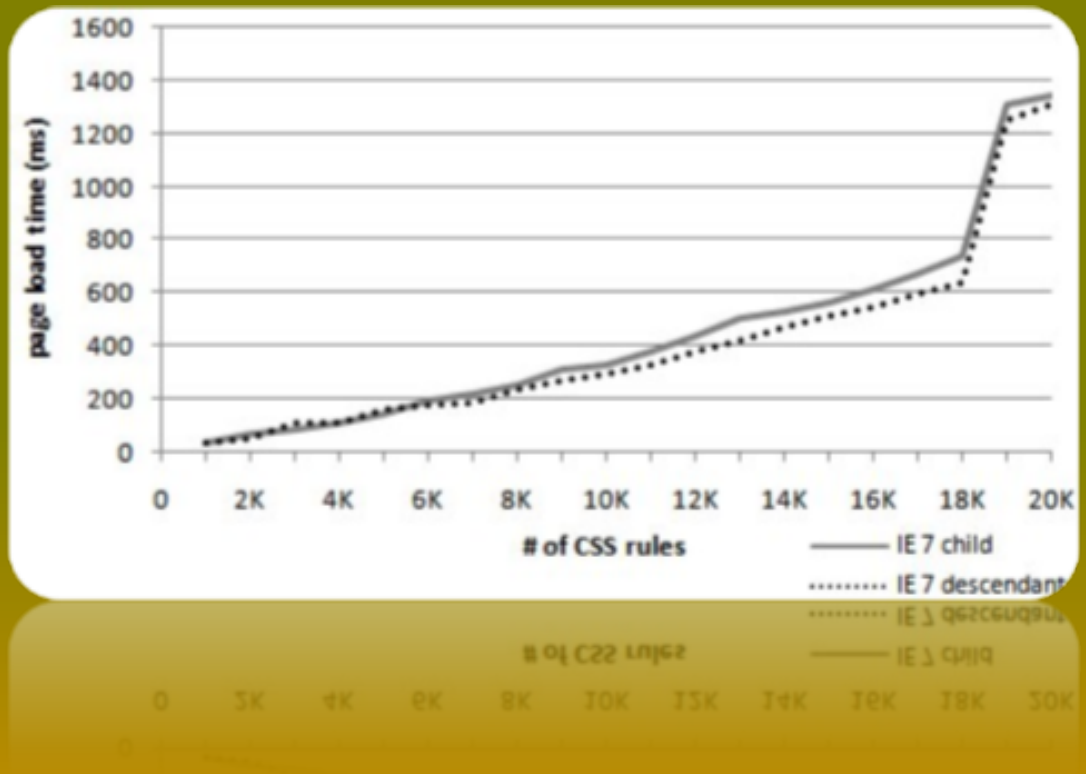
child:

```
DIV > DIV > DIV > P > A.a00001 {}
```



http://jon.sykes.me/153/more-css-performance-testing-pt-3

CSS performance isn't linear



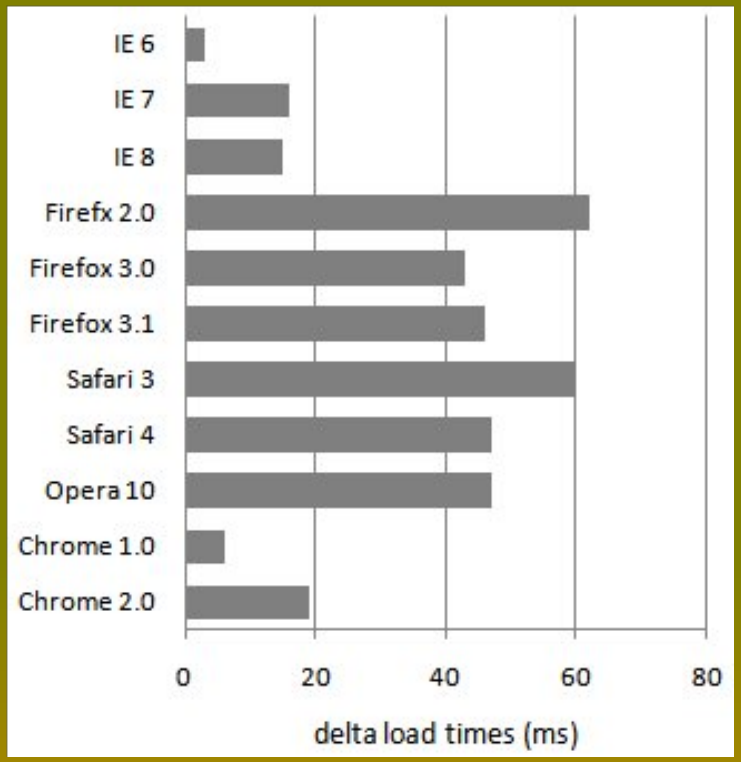
IE 7 "cliff" at 18K rules

real world levels of CSS

	# Rules	# elements	Avg Depth
AOL	2289	1628	13
eBay	305	588	14
Facebook	2882	1966	17
Google Search	92	552	8
Live Search	376	449	12
MSN.com	1038	886	11
MySpace	932	444	9
Wikipedia	795	1333	10
Yahoo!	800	564	13
YouTube	821	817	9
average	**1033**	**923**	**12**

testing typical CSS

1K rules (vs. 20K)
same amount of CSS in all test pages
30 ms avg delta



"costly"selectors aren't always costly (at typical levels)

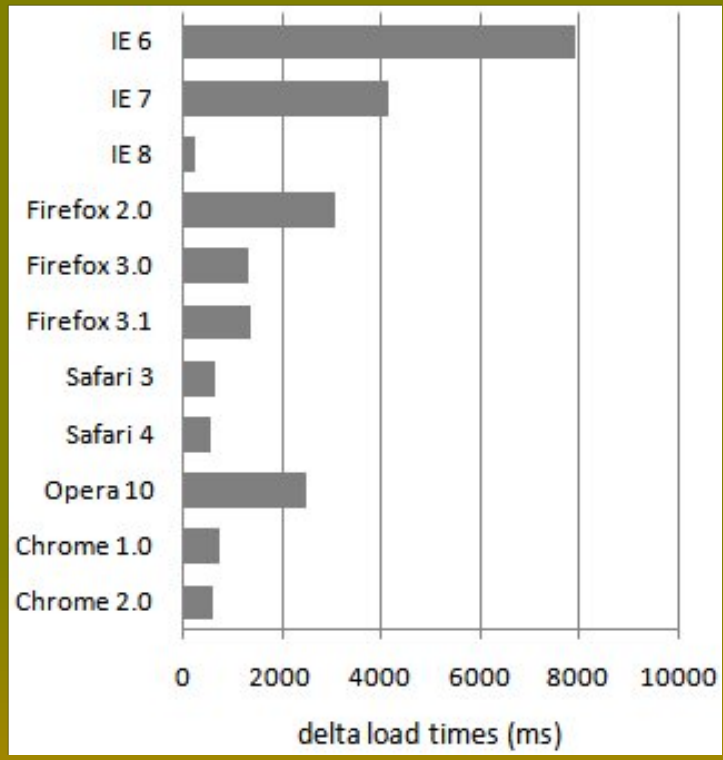
are these selectors "costly"?

```
DIV DIV DIV P A.class0007 { ... }
```

http://www.stevesouders.com/blog/2009/03/10/performance-impact-of-css-selectors/

testing expensive selectors

1K rules (vs. 20K)
same amount of CSS in all test pages
2126 ms avg delta!



truly expensive selector

```
A.class0007 * { ... }
```

compare to:

```
DIV DIV DIV P A.class0007 { ... }
```

the key is the *key selector* - the rightmost argument

CSS3 selectors (bad)

more David Hyatt:

> "The sad truth about CSS3 selectors is that they really shouldn’t be used at all if you care about page performance. Decorating your markup with classes and ids and matching purely on those while avoiding all uses of sibling, descendant and child selectors will actually make a page perform significantly better in all browsers."

selectors to avoid

```
A.class0007 DIV { ... }
#id0007 > A { ... }
.class0007 [href] { ... }
DIV:first-child { ... }
```

reflow time vs. load time

reflow – time to apply CSS, re-layout elements, and repaint

triggered by DHTML:

```
elem.className = "newclass";
elem.style.cssText = "color: red";
elem.style.padding = "8px";
elem.style.display = "";
```

reflow can happen multiple times for long-lasting Web 2.0 apps

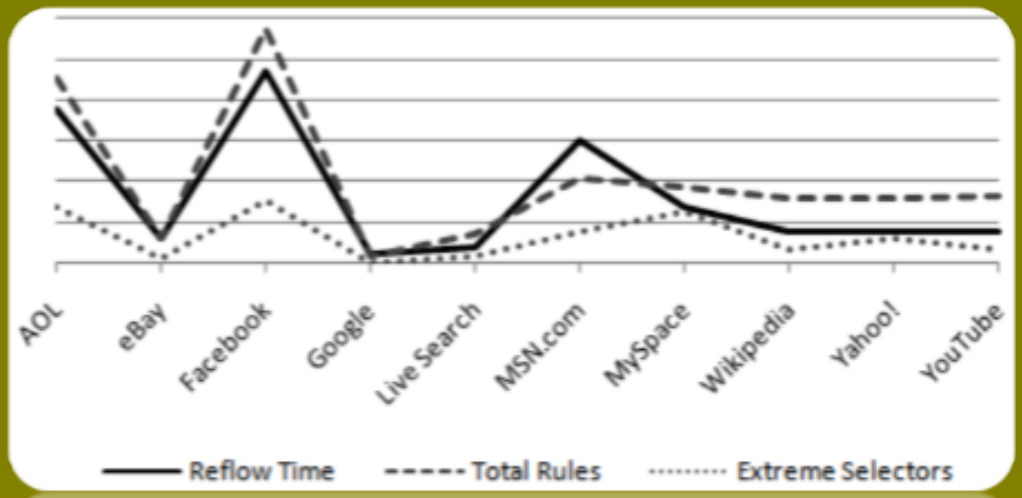
reflow time by browser

DHTML action	Chr1	Chr2	FF2	FF3	IE6,7	IE 8	Op	Saf3	Saf4
className	1x	1x	1x	1x	1x	1x	1x	1x	1x
display none	-	-	-	-	1x	-	-	-	-
display default	1x	1x	1x	2x	1x	1x	-	1x	1x
visibility hidden	1x	1x	1x	1x	1x	1x	-	1x	1x
visibility visible	1x	1x	1x	1x	1x	1x	-	1x	1x
padding	-	-	1x	2x	4x	4x	-	-	-
width length	-	-	1x	2x	1x	1x	-	1x	-
width percent	-	-	1x	2x	1x	1x	-	1x	-
width default	1x	-	1x	2x	1x	1x	-	1x	-
background	-	-	1x	1x	1x	-	-	-	-
font-size	1x	1x	1x	2x	1x	1x	-	1x	1x

reflow performance varies by browser and action

"1x" is 1-6 seconds depending on browser (1K rules)

Simplifying CSS Selectors



efficient CSS comes at a cost – page weight

focus optimization on selectors where the *key selector* matches many elements

reduce the number of selectors

Avoiding @import - @import @import

```
<style>
@import url('stylesheet1.css');
@import url('stylesheet2.css');
</style>
```

no blocking

in fact, improves progressive rendering in IE

http://stevesouders.com/tests/atimport/import-import.php

link @import

```
<link rel='stylesheet' type='text/css'
href='stylesheet1.css'>
<style>
@import url('stylesheet2.css');
</style>
```

blocks in IE

http://stevesouders.com/tests/atimport/link-import.php

link with @import

```
<link rel='stylesheet' type='text/css'
href='stylesheet1.css'>
```

includes

```
@import url('stylesheet2.
css');
```

blocks in all browsers!

http://stevesouders.com/tests/atimport/link-with-import.php

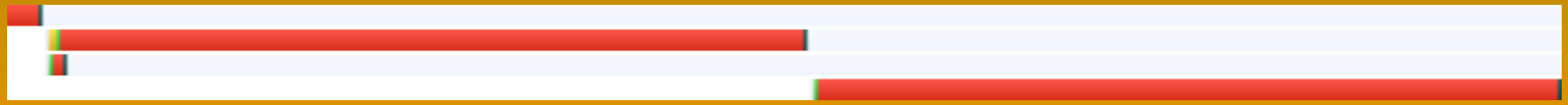
link blocks @import

```
<link rel='stylesheet' type='text/css'
href='stylesheet1.css'>
<link rel='stylesheet' type='text/css'
href='proxy.css'>
```

includes

```
@import url('stylesheet2.
css');
```

blocks in IE

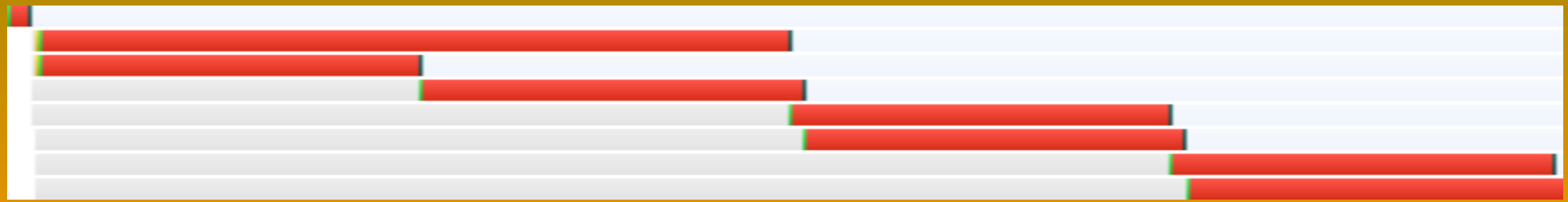


http://stevesouders.com/tests/atimport/link-blocks-import.php

many @imports

```
<style>
@import url('stylesheet1.css');
...
@import url('stylesheet6.css');
</style>
<script src='script1.js'></script>
```

loads script before stylesheets in IE



http://stevesouders.com/tests/atimport/many-imports.php

link link

```
<link rel='stylesheet' type='text/css'
href='stylesheet1.css'>
<link rel='stylesheet' type='text/css'
href='stylesheet2.css'>
```

no blocking in all browsers

http://stevesouders.com/tests/atimport/link-link.php

going beyond gzipping

Tony Gentilcore, Chapter 9, Even Faster Web Sites

Rule 4: Gzip Components from HPWS

HTTP/1.1

- request: `Accept-Encoding: gzip,deflate`
- response: `Content-Encoding: gzip`

Apache 2.x:

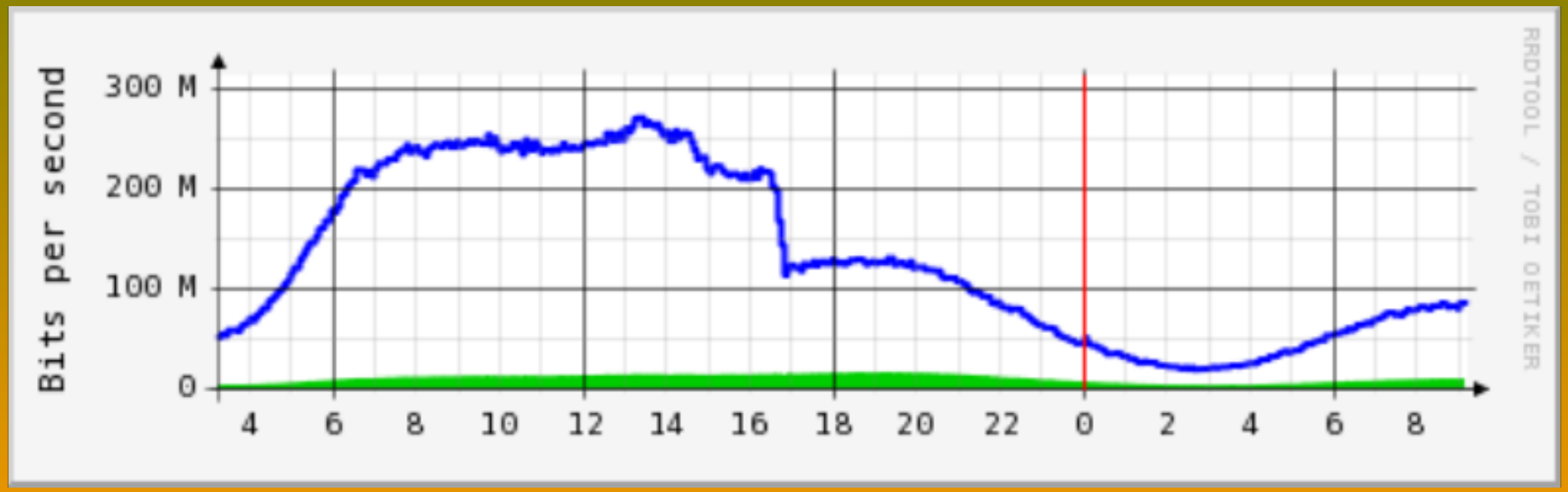
```
AddOutputFilterByType DEFLATE
text/html text/css application/x-
javascript
```

benefits of gzipping

70% reduction in transfer size

not just for HTML!!

- all text: JavaScript, CSS, XML, JSON
- not binary: images, PDF, Flash



so what's the issue?

15% of users get uncompressed responses

surprize! why?

old browsers? no

- Netscape Navigator 3 – 0.0%
- Netscape Communicator 4 – 0.1%
- Opera 3.5 – 0.0%
- IE <3 – 0.01%

clue: most prevalent in the Middle East

proxies and anti-virus software

sometimes, Accept-Encoding is missing

other times, it's obfuscated

```
Accept-EncodXng: gzip, deflate
X-cept-Encoding: gzip, deflate
XXXXXXXXXXXXXXX: XXXXXXXXXXXXX
---------------: -------------
~~~~~~~~~~~~~~~: ~~~~~~~~~~~~~
```

proxies and anti-virus software disable compression for easier response filtering

check your site (http://stevesouders.com)

recorded headers for 500 unique users

14% missing A-E, 1% munged A-E

```
ACCEPT_ENCODING=gzip, deflate
ACCEPT_ENCODXNG=gzip, deflate
                    =----- -------
__________________
```

indicators

	overall	null A-E
VIA	53 (11%)	28 (41%)
PROXY_CONNECTION	12 (2%)	12 (18%)
CONNECTION missing	24 (5%)	15 (22%)
ACCEPT_CHARSET missing	173 (35%)	54 (79%)
SERVER_PROTOCOL = HTTP/1.0	45 (9%)	17 (25%)
UA_CPU = x86	111 (22%)	43 (63%)

what to do

don't assume compression

go the extra mile to reduce response size

- minify HTML, JavaScript, and CSS
- use CSS rules over inline styles
- alias long JavaScript symbol names
- leverage relative URLs

Thanks, Tony!

See Tony's session at Velocity for more details.

takeaways

focus on the frontend

run YSlow: http://developer.yahoo.com/yslow

speed matters

impact on revenue

Google: +500 ms -20% traffic[1]

Yahoo: +400 ms -5-9% full-page traffic[2]

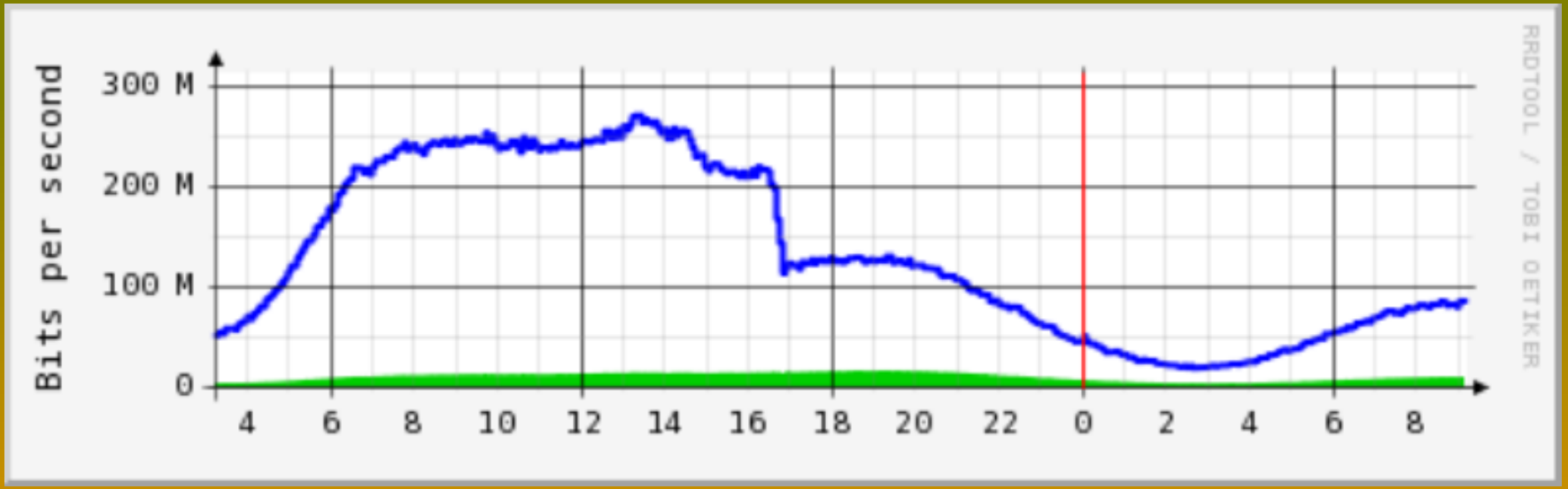
Amazon: +100 ms -1% sales[1]

1 http://home.blarg.net/~glinden/StanfordDataMining.2006-11-29.ppt
2 http://www.slideshare.net/stoyan/yslow-20-presentation

cost savings

hardware – reduced load

bandwidth – reduced response size



http://billwscott.com/share/presentations/2008/stanford/HPWP-RealWorld.pdf

if you want
- better user experience
- more revenue
- reduced operating expenses

the strategy is clear

Even Faster Web Sites

Steve Souders

souders@google.com

http://stevesouders.com/docs/googleio-20090527.ppt