Building Faster Websites

crash course on web performance

Ilya Grigorik - @igrigorik
Make The Web Fast
Google

Make the Web Fast team at Google:

- Kernel, Networking, Infrastructure, Chrome, Mobile...
- Research & drive performance web standards (W3C, etc)
- Build open source tools, contribute to existing projects
- Optimize Google, optimize the web...

developers.google.com/speed

Goal: make the entire web *faster*

Our agenda for today...

1. **The problem...**
   - ○ Trends on the web
   - ○ Networking in the browser (HTTP, and beyond)
   - ○ Mobile networks
2. **Browser architecture under the hood...**
   - ○ Measuring performance
   - ○ Networking, DOM, Rendering, HW acceleration
3. **Best practices, with context...**
   - ○ Optimizing load time
   - ○ Optimizing apps (FPS, memory, etc)
   - ○ Automating optimization...

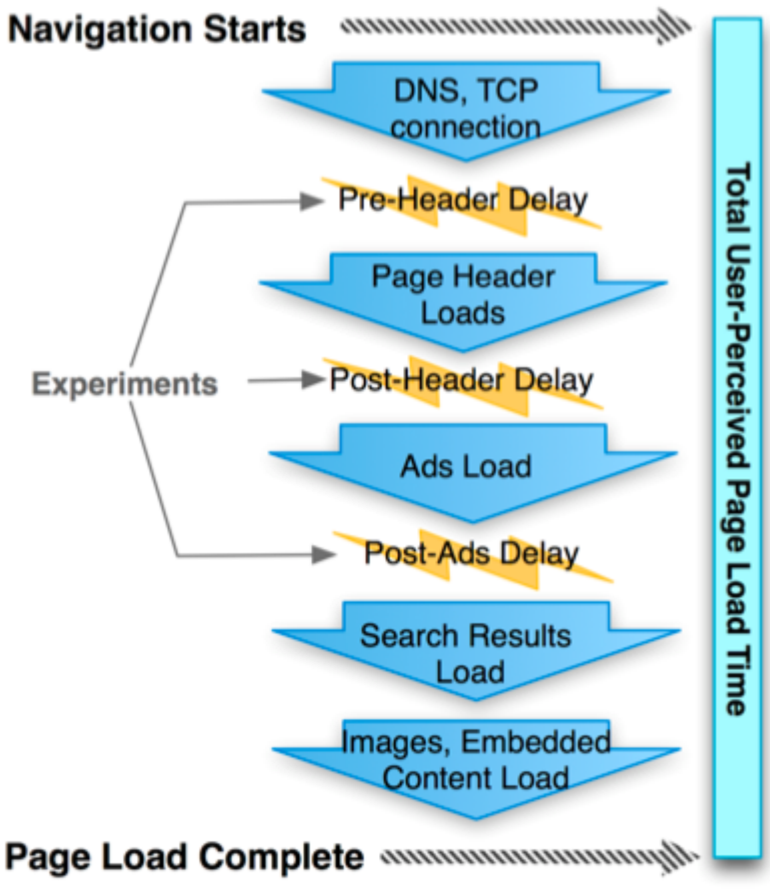
Trends & Technologies...

What do we mean by fast? Why? Won't the networks save us? Mobile?

What's the impact of slow sites?

Lower conversions and engagement, higher bounce rates...

Google Web Search Delay Experiment



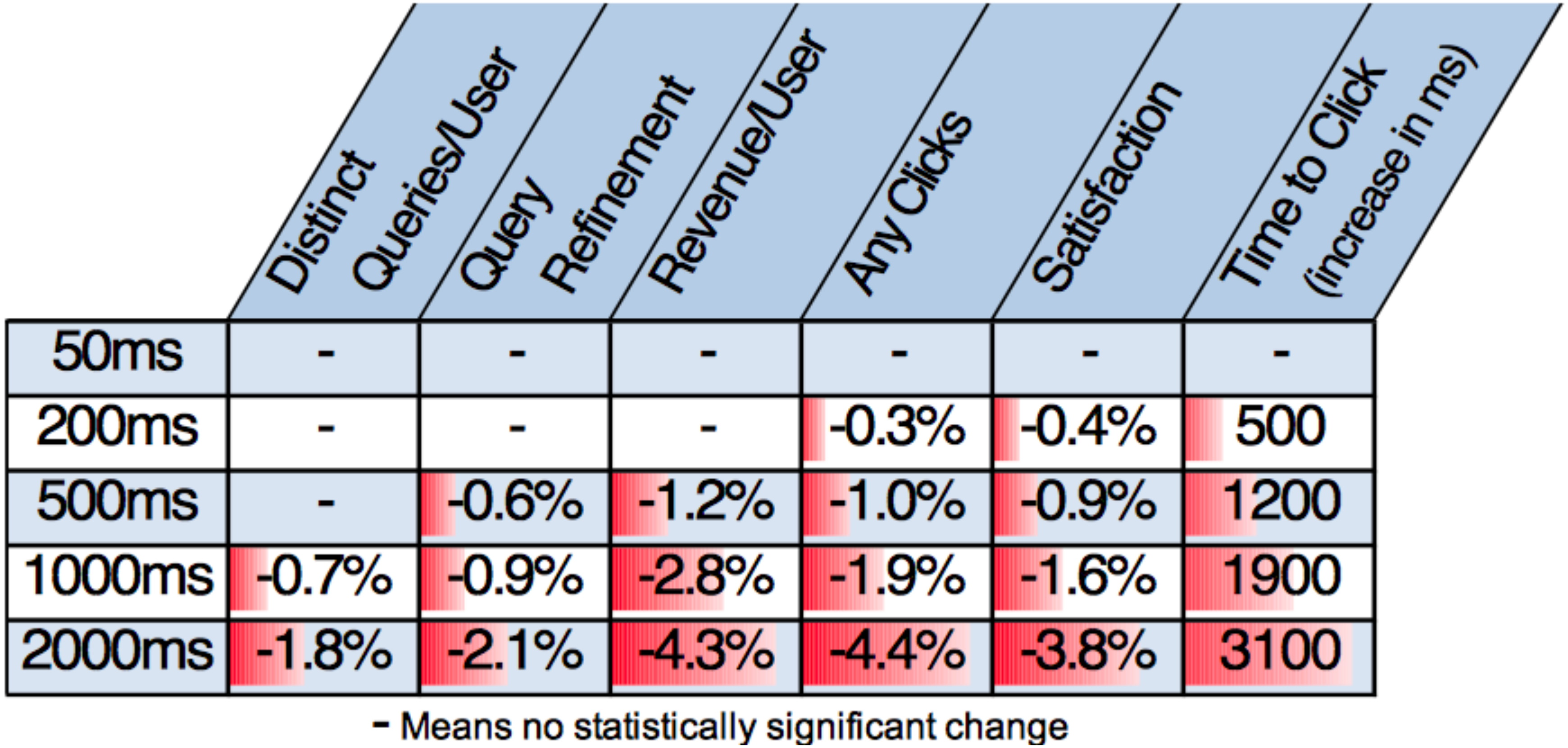
Type of Delay	Delay (ms)	Duration (weeks)	Impact on Avg. Daily Searches
Pre-header	50	4	Not measurable
Pre-header	100	4	-0.20%
Post-header	200	6	-0.59%
Post-header	400	6	-0.59%
Post-ads	200	4	-0.30%

- The cost of delay increases over time and persists
- Delays under half a second impact business metrics
- "Speed matters" is not just lip service

Performance Related Changes and their User Impact

@igrigorik

Server Delays Experiment



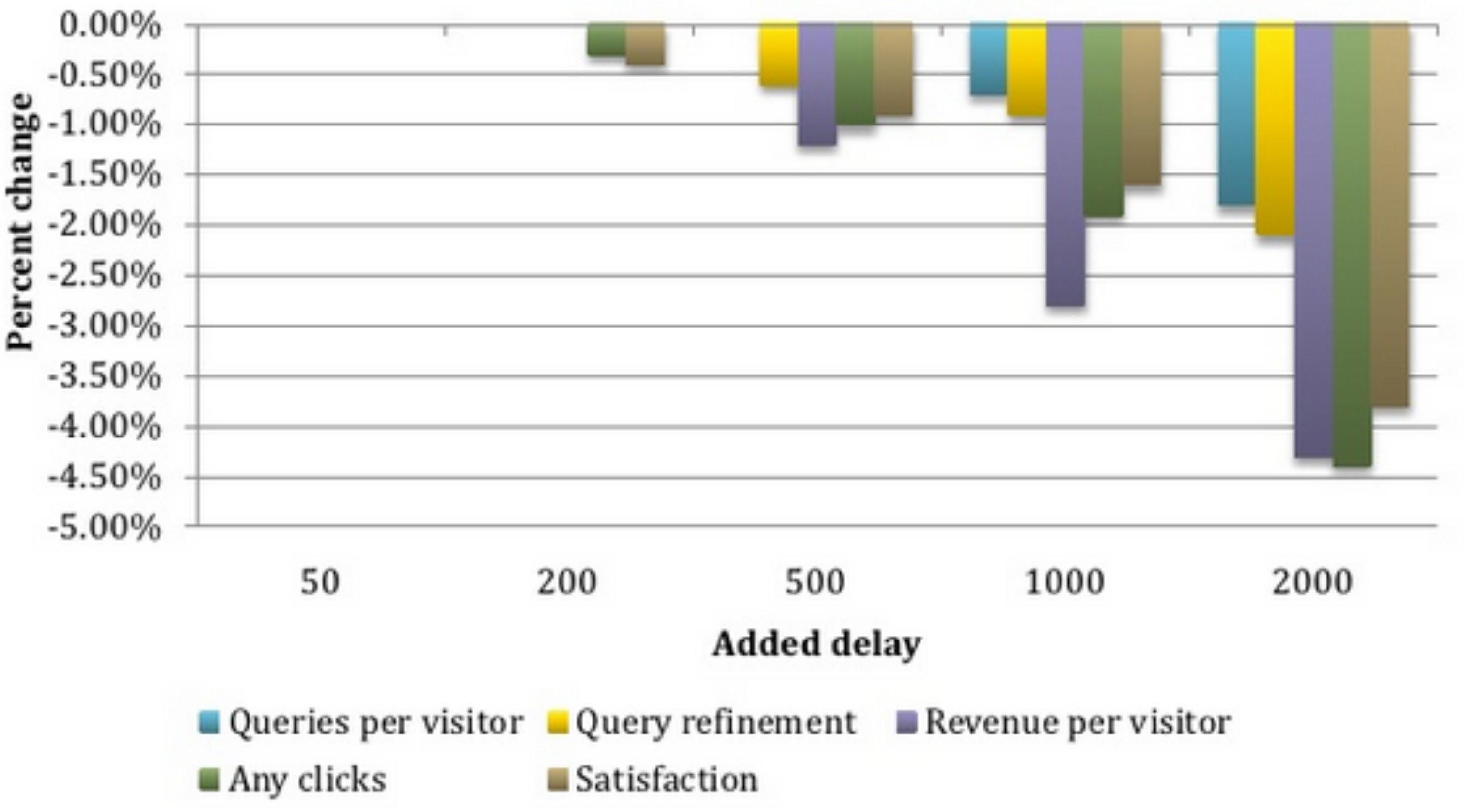
	Distinct Queries/User	Query Refinement	Revenue/User	Any Clicks	Satisfaction	Time to Click (increase in ms)
50ms	-	-	-	-	-	-
200ms	-	-	-	-0.3%	-0.4%	500
500ms	-	-0.6%	-1.2%	-1.0%	-0.9%	1200
1000ms	-0.7%	-0.9%	-2.8%	-1.9%	-1.6%	1900
2000ms	-1.8%	-2.1%	-4.3%	-4.4%	-3.8%	3100

- Means no statistically significant change

- Strong negative impacts
- Roughly linear changes with increasing delay
- Time to Click changed by roughly double the delay

Performance Related Changes and their User Impact

Server Delays Experiment

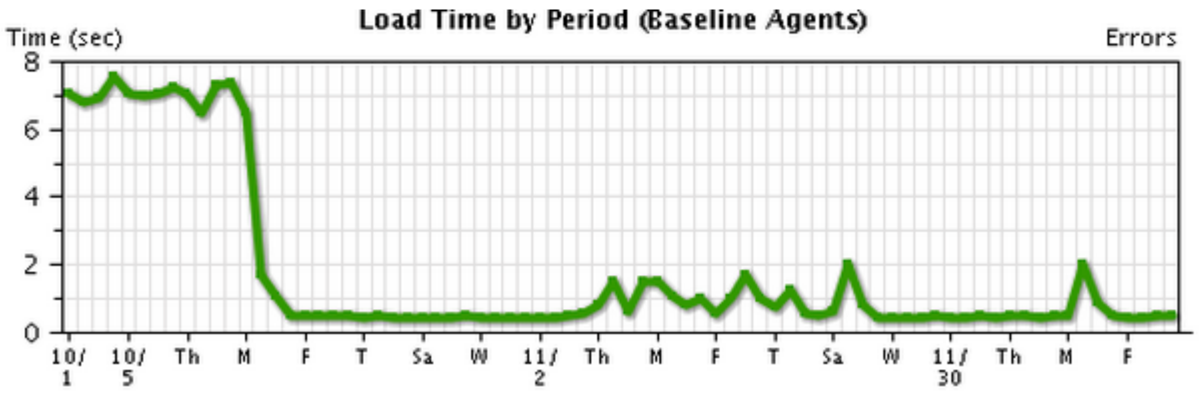


- Strong negative impacts
- Roughly linear changes with increasing delay
- Time to Click changed by roughly double the delay

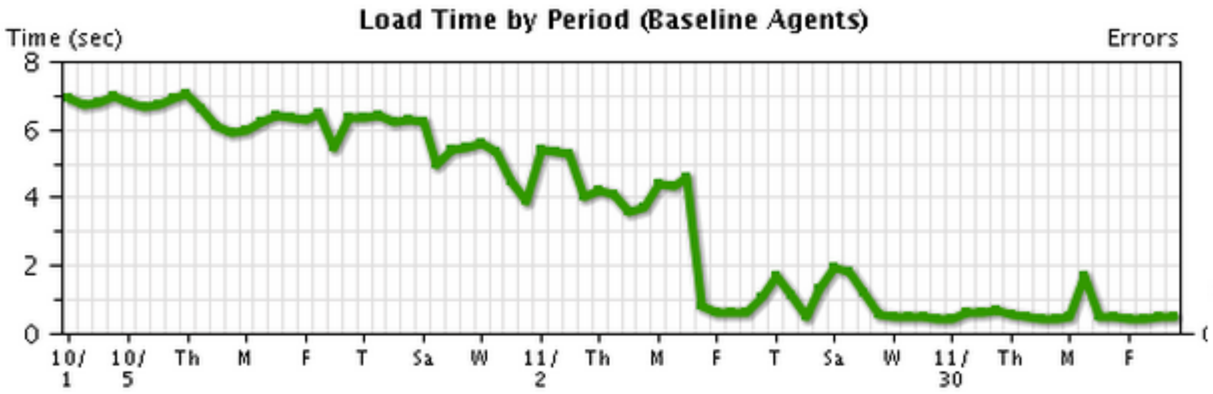
[Impact of web latency on conversion rates]

Impact of PLT on bottom line

shopzilla.com



bizrate.co.uk



shopzilla
the smarter way to shop

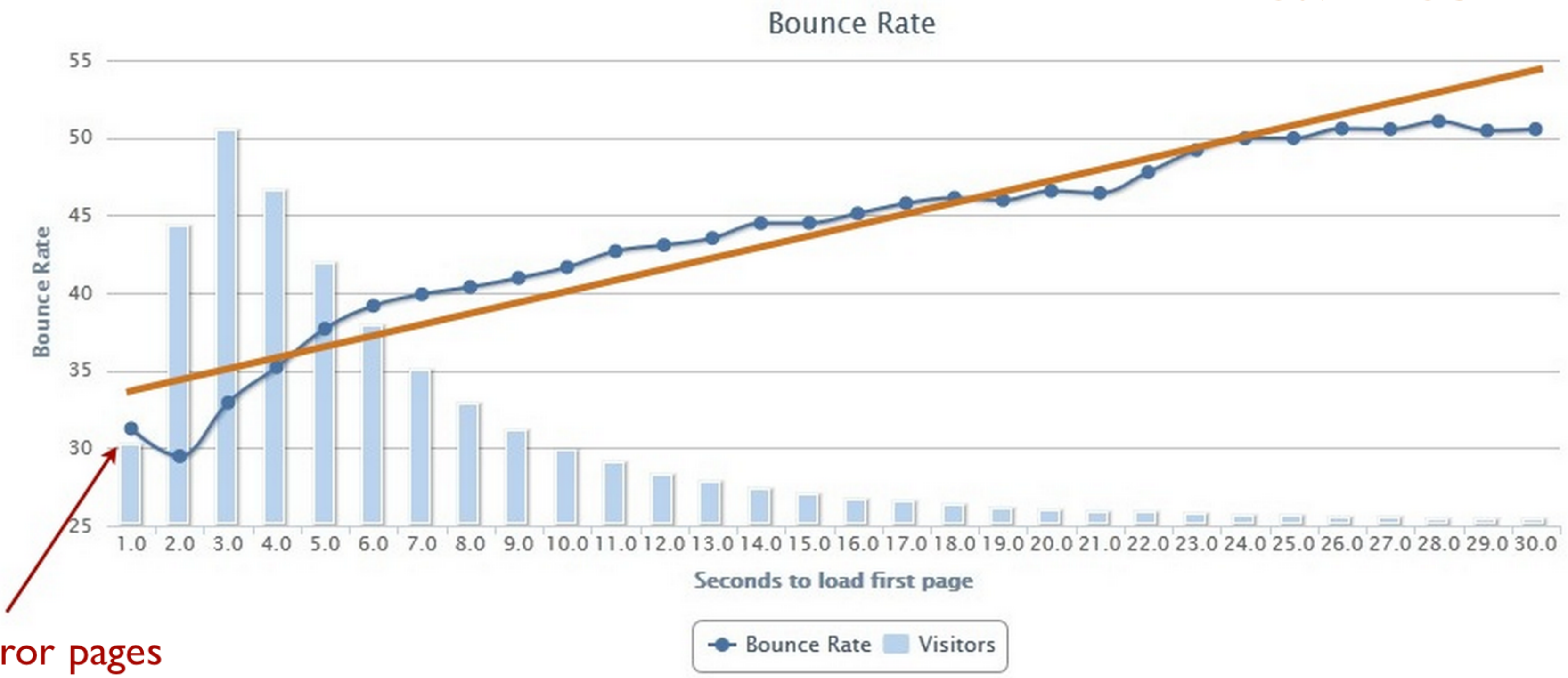
Conversion Rate	+7~12%
Pageviews	+25%
US SEM sessions	+8%
Bizrate.co.uk SEM sessions	+120%

Shopzilla's Site Redo

How speed affects bounce rate

$$y = 0.6517x + 33.682$$

$$R^2 = 0.91103$$

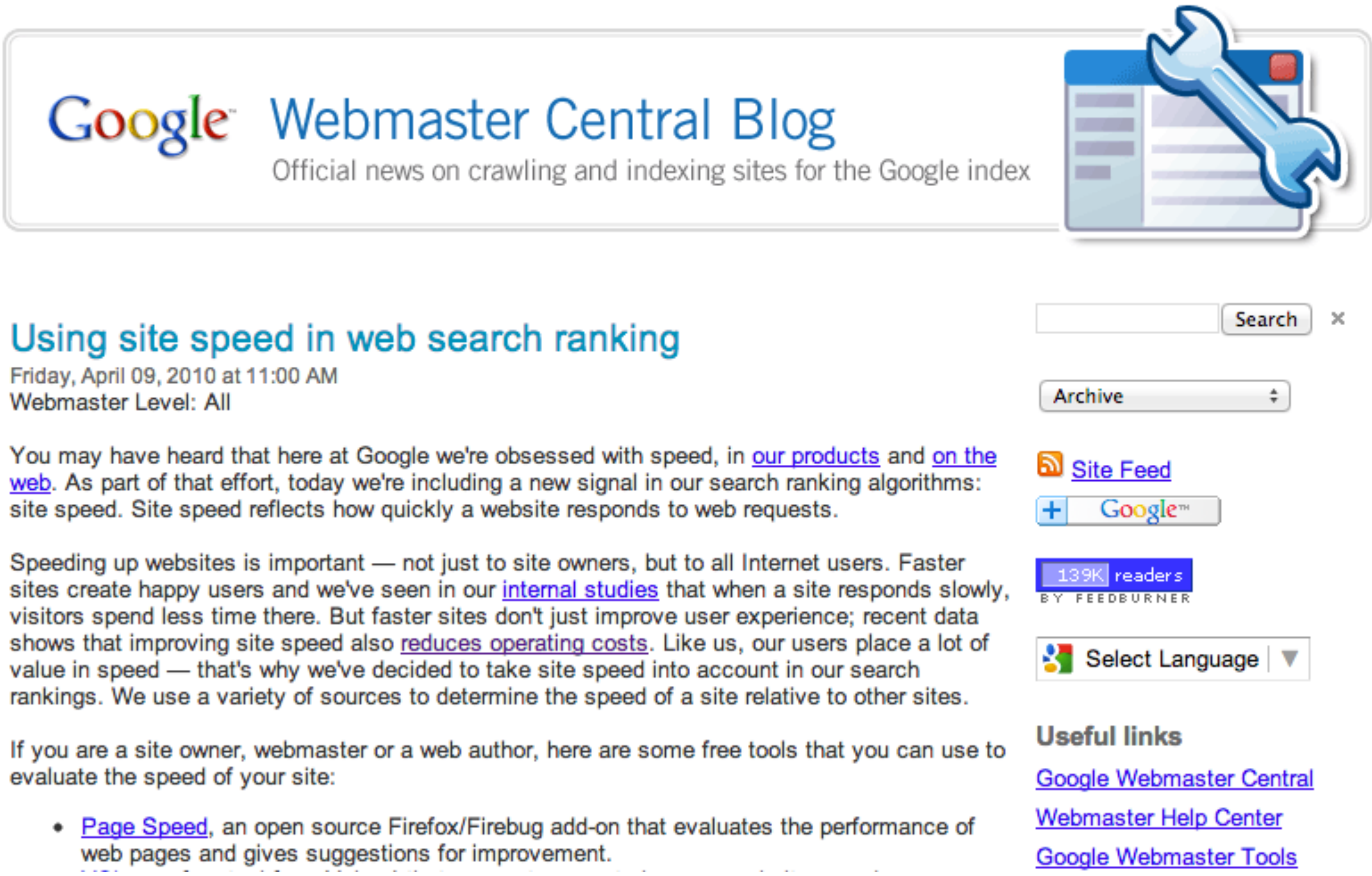


Error pages

Every second = 0.65 increase in bounce rate

[Yo ho ho and a few billion pages of RUM](#)

Site speed is a signal for search



Using site speed in web search ranking

Friday, April 09, 2010 at 11:00 AM
Webmaster Level: All

You may have heard that here at Google we're obsessed with speed, in our products and on the web. As part of that effort, today we're including a new signal in our search ranking algorithms: site speed. Site speed reflects how quickly a website responds to web requests.

Speeding up websites is important — not just to site owners, but to all Internet users. Faster sites create happy users and we've seen in our internal studies that when a site responds slowly, visitors spend less time there. But faster sites don't just improve user experience; recent data shows that improving site speed also reduces operating costs. Like us, our users place a lot of value in speed — that's why we've decided to take site speed into account in our search rankings. We use a variety of sources to determine the speed of a site relative to other sites.

If you are a site owner, webmaster or a web author, here are some free tools that you can use to evaluate the speed of your site:

- Page Speed, an open source Firefox/Firebug add-on that evaluates the performance of web pages and gives suggestions for improvement.

Useful links

Google Webmaster Central
Webmaster Help Center
Google Webmaster Tools

> ***"We encourage you to start looking at your site's speed — not only to improve your ranking in search engines, but also to improve everyone's experience on the Internet."***
>
> *Google Search Quality Team*

Using site speed in web search ranking

*If you want to succeed with web-performance, **don't view it as a technical metric**. Instead, **measure and correlate it's impact on your business metrics**.*

How do you do that? With analytics and real user monitoring.

So, how are we doing today?

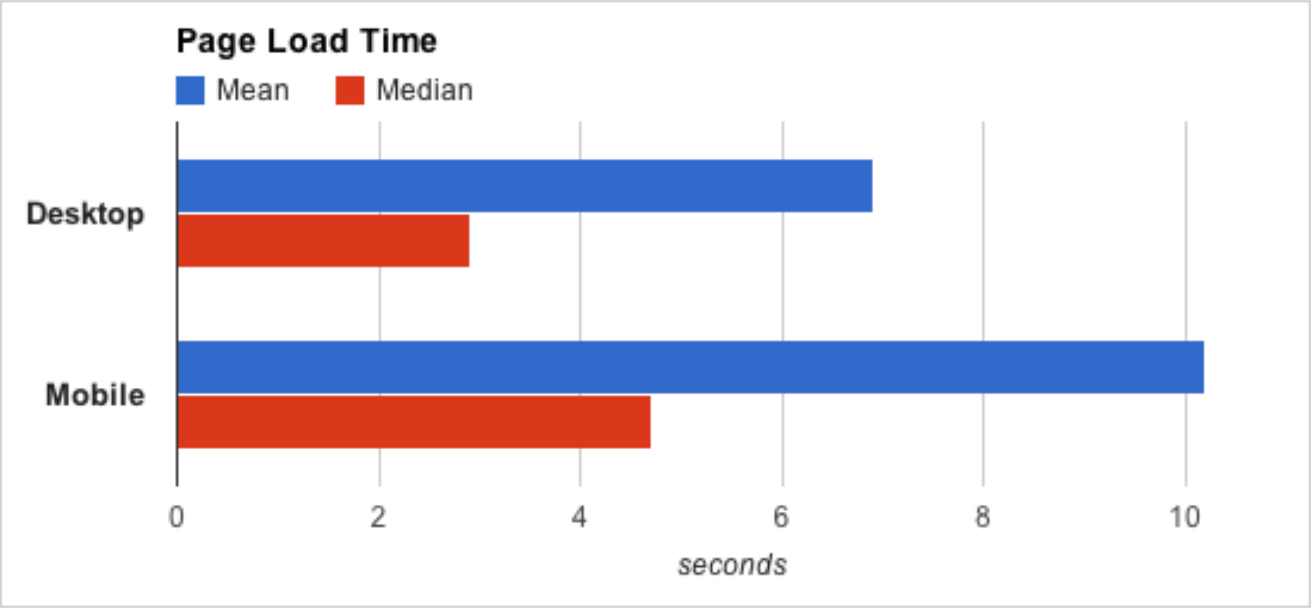
Okay, I get it, speed matters... but, are we there yet?

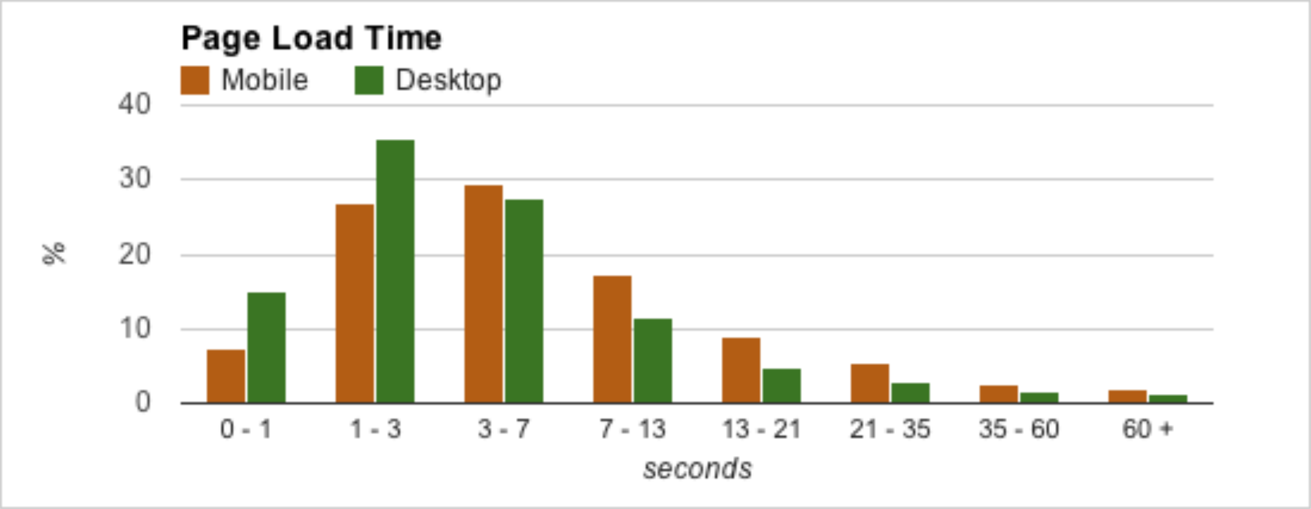
Usability Engineering 101

Delay	User reaction
0 - 100 ms	Instant
100 - 300 ms	*Feels sluggish*
300 - 1000 ms	Machine is working...
1 s+	Mental context switch
10 s+	I'll come back later...

Rule of thumb:

Stay under 250 ms to feel "fast".





Desktop

Median: ~2.7s

Mean: ~6.9s

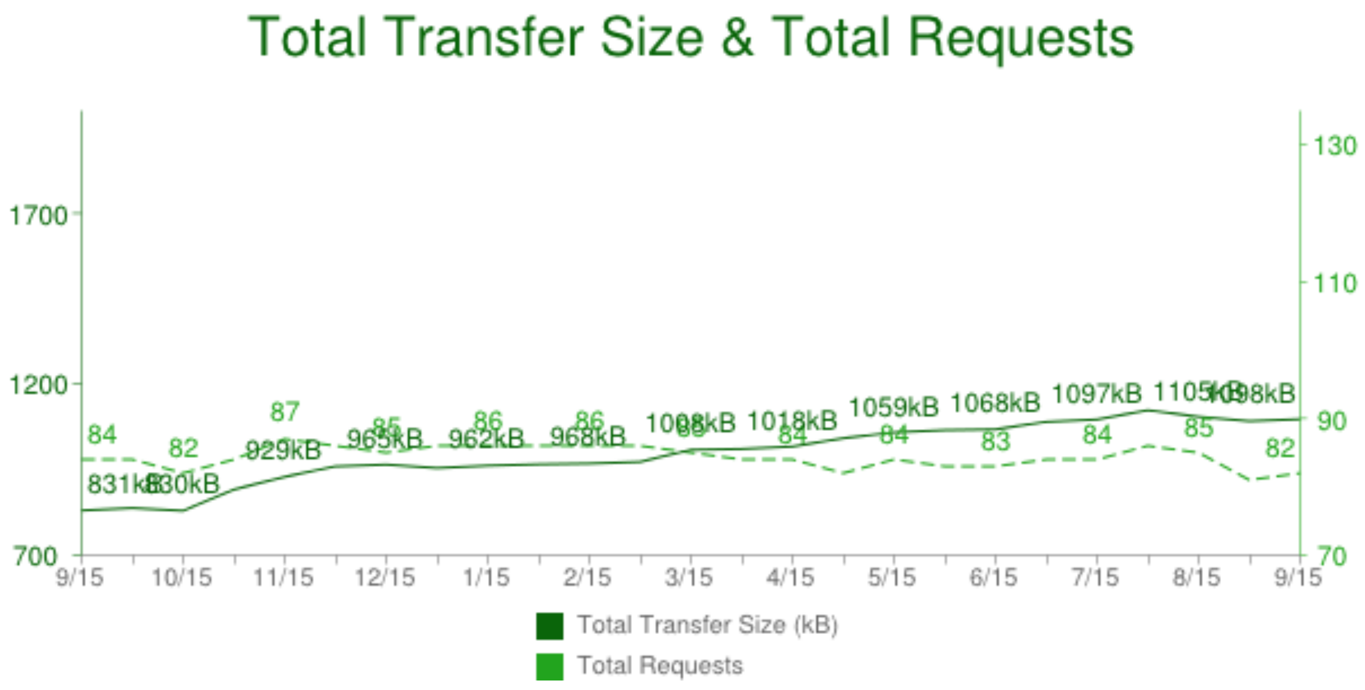
Mobile *

Median: ~4.8s

Mean: ~10.2s

* *optimistic*

How Fast Are Websites Around The World? - Google Analytics Blog

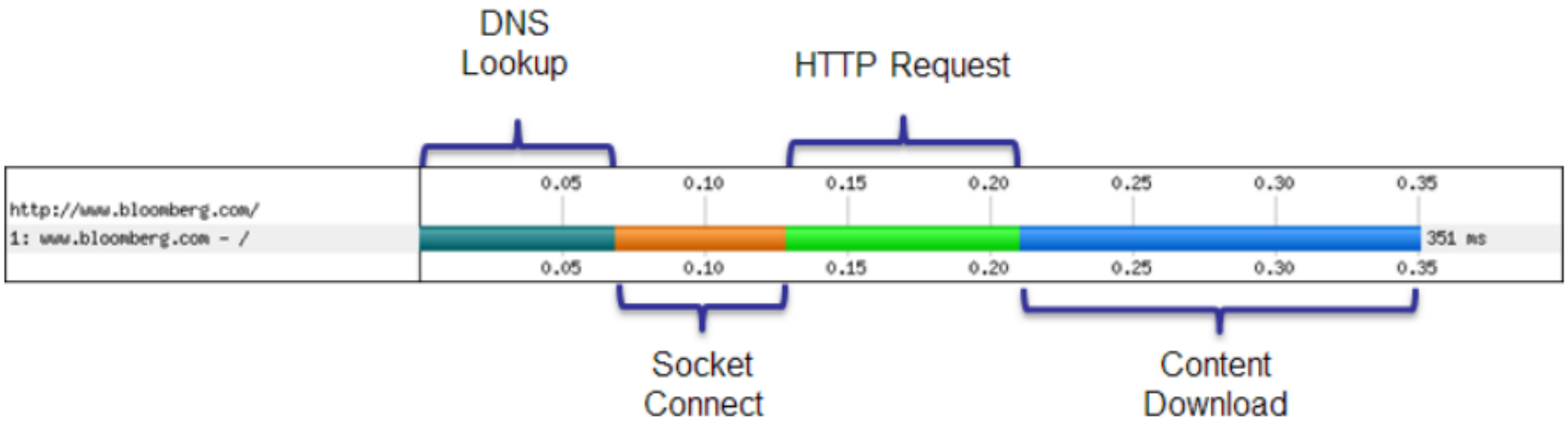


Total Transfer Size & Total Requests

Content Type	Avg # of Requests	Avg size
HTML	8	44 kB
Images	**53**	**635 kB**
Javascript	**14**	**189 kB**
CSS	5	35 kB

HTTP Archive - Trends (Sept, 2012)

Life of an HTTP Request



DNS Lookup

HTTP Request

Socket Connect

Content Download

http://www.bloomberg.com/

1: www.bloomberg.com - /

Let's talk about DNS

A very brief, but important detour...

Most DNS servers are...

- Under provisioned
- Not monitored well
- Susceptible to attacks
- ...

- Poor cache hit rate
- Intermittent failures
- DDOS, cache poisoning, ...

*"Operating the Googlebot web crawler, we have observed an **average resolution time of 130 ms** for nameservers that respond. However, a **full 4-6% of requests simply time out**, due to UDP packet loss and servers being unreachable. If we take into account failures such as packet loss, dead nameservers, DNS configuration errors, etc., the **actual average end-to-end resolution time is 300-400 ms**."*

[Public DNS: Performance Benefits](#)

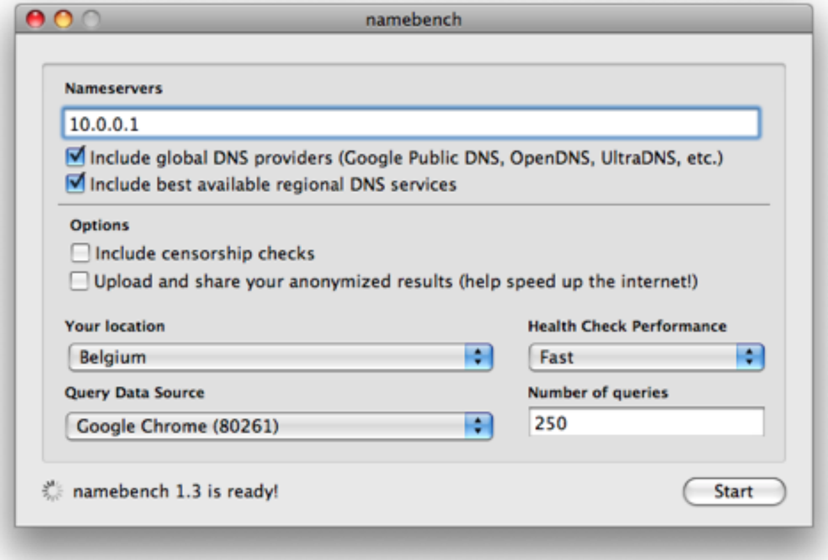
8.8.4.4

8.8.8.8

Google Public DNS

free, no redirects, etc.

namebench



namebench

Nameservers

10.0.0.1

☑ Include global DNS providers (Google Public DNS, OpenDNS, UltraDNS, etc.)

☑ Include best available regional DNS services

Options

☐ Include censorship checks

☐ Upload and share your anonymized results (help speed up the internet!)

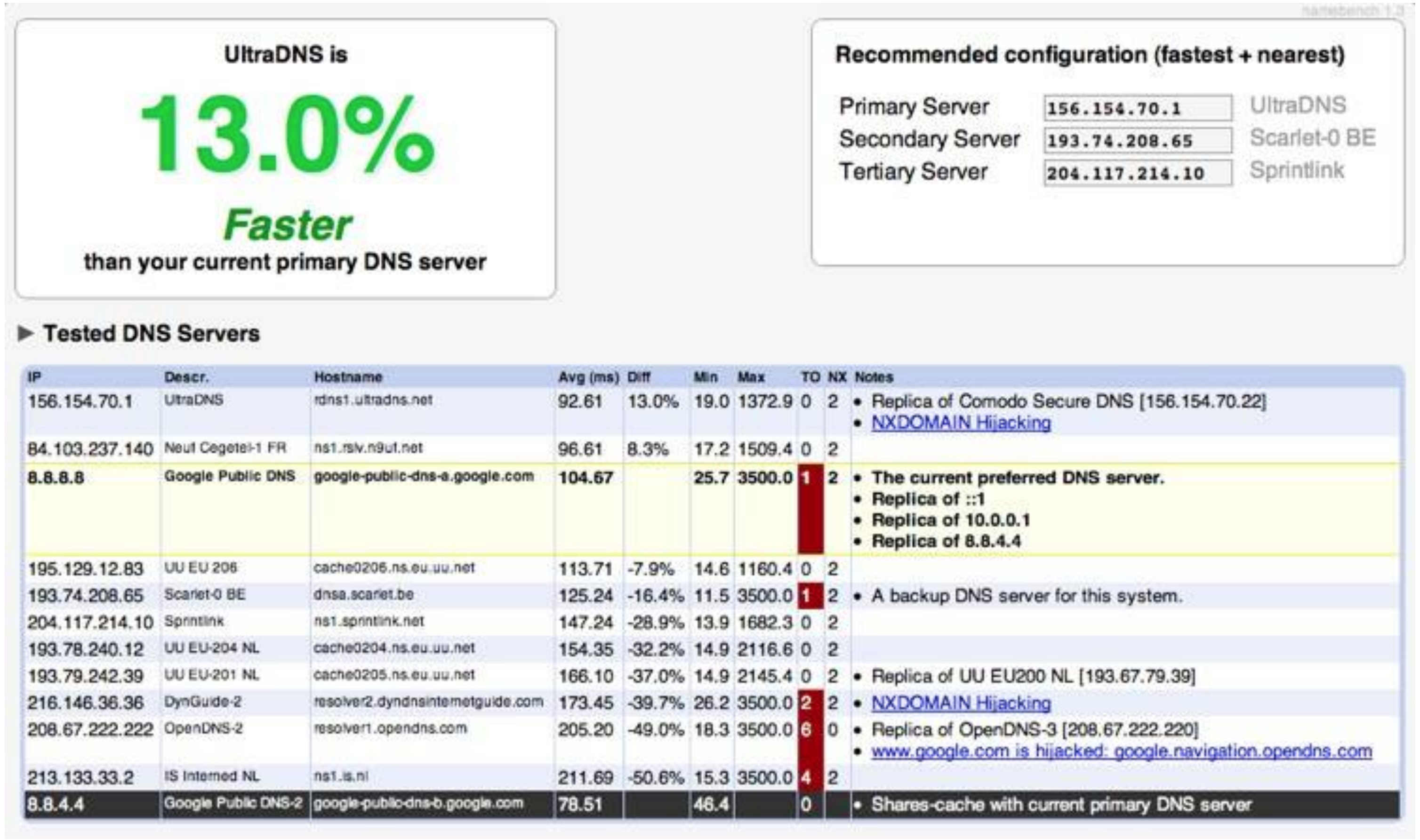
Your location: Belgium

Health Check Performance: Fast

Query Data Source: Google Chrome (80261)

Number of queries: 250

namebench 1.3 is ready!



namebench 1.3

UltraDNS is

13.0%

Faster

than your current primary DNS server

Recommended configuration (fastest + nearest)

Primary Server	156.154.70.1	UltraDNS
Secondary Server	193.74.208.65	Scarlet-0 BE
Tertiary Server	204.117.214.10	Sprintlink

▶ **Tested DNS Servers**

IP	Descr.	Hostname	Avg (ms)	Diff	Min	Max	TO	NX	Notes
156.154.70.1	UltraDNS	rdns1.ultradns.net	92.61	13.0%	19.0	1372.9	0	2	• Replica of Comodo Secure DNS [156.154.70.22] • NXDOMAIN Hijacking
84.103.237.140	Neuf Cegetel-1 FR	ns1.rslv.n9uf.net	96.61	8.3%	17.2	1509.4	0	2	
8.8.8.8	**Google Public DNS**	**google-public-dns-a.google.com**	**104.67**		**25.7**	**3500.0**	1	**2**	• **The current preferred DNS server.** • **Replica of ::1** • **Replica of 10.0.0.1** • **Replica of 8.8.4.4**
195.129.12.83	UU EU 206	cache0206.ns.eu.uu.net	113.71	-7.9%	14.6	1160.4	0	2	
193.74.208.65	Scarlet-0 BE	dnsa.scarlet.be	125.24	-16.4%	11.5	3500.0	1	2	• A backup DNS server for this system.
204.117.214.10	Sprintlink	ns1.sprintlink.net	147.24	-28.9%	13.9	1682.3	0	2	
193.78.240.12	UU EU-204 NL	cache0204.ns.eu.uu.net	154.35	-32.2%	14.9	2116.6	0	2	
193.79.242.39	UU EU-201 NL	cache0205.ns.eu.uu.net	166.10	-37.0%	14.9	2145.4	0	2	• Replica of UU EU200 NL [193.67.79.39]
216.146.36.36	DynGuide-2	resolver2.dyndnsinternetguide.com	173.45	-39.7%	26.2	3500.0	2	2	• NXDOMAIN Hijacking
208.67.222.222	OpenDNS-2	resolver1.opendns.com	205.20	-49.0%	18.3	3500.0	6	0	• Replica of OpenDNS-3 [208.67.222.220] • www.google.com is hijacked: google.navigation.opendns.com
213.133.33.2	IS Internet NL	ns1.is.nl	211.69	-50.6%	15.3	3500.0	4	2	
8.8.4.4	Google Public DNS-2	google-public-dns-b.google.com	78.51		46.4		0		• Shares-cache with current primary DNS server

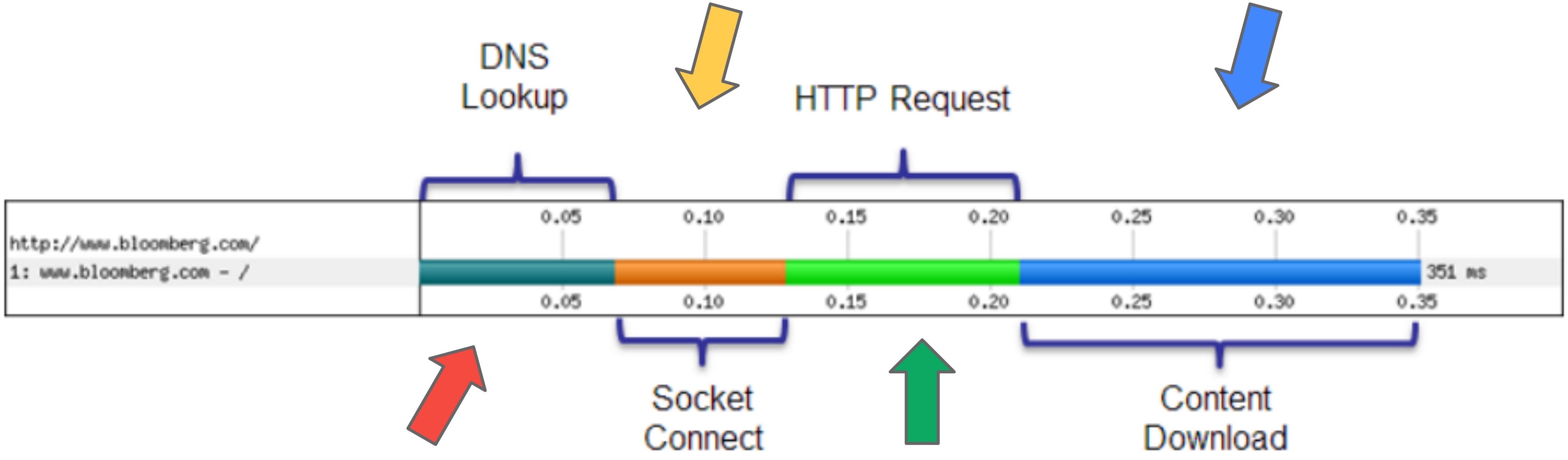
"namebench runs a fair and ***thorough benchmark using your web browser history, tcpdump output, or standardized datasets*** *in order to provide an* ***individualized recommendation****. namebench is completely free and does not modify your system in any way. This project began as a 20% project at Google."*

namebench - Google Code

Life of an HTTP Request

Can we move the server closer?

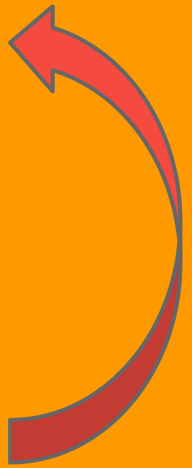
Did you compress, minify, etc?



- *Benchmark your site DNS provider*
- *Benchmark your ISP DNS provider...*

Can we make the server respond faster?

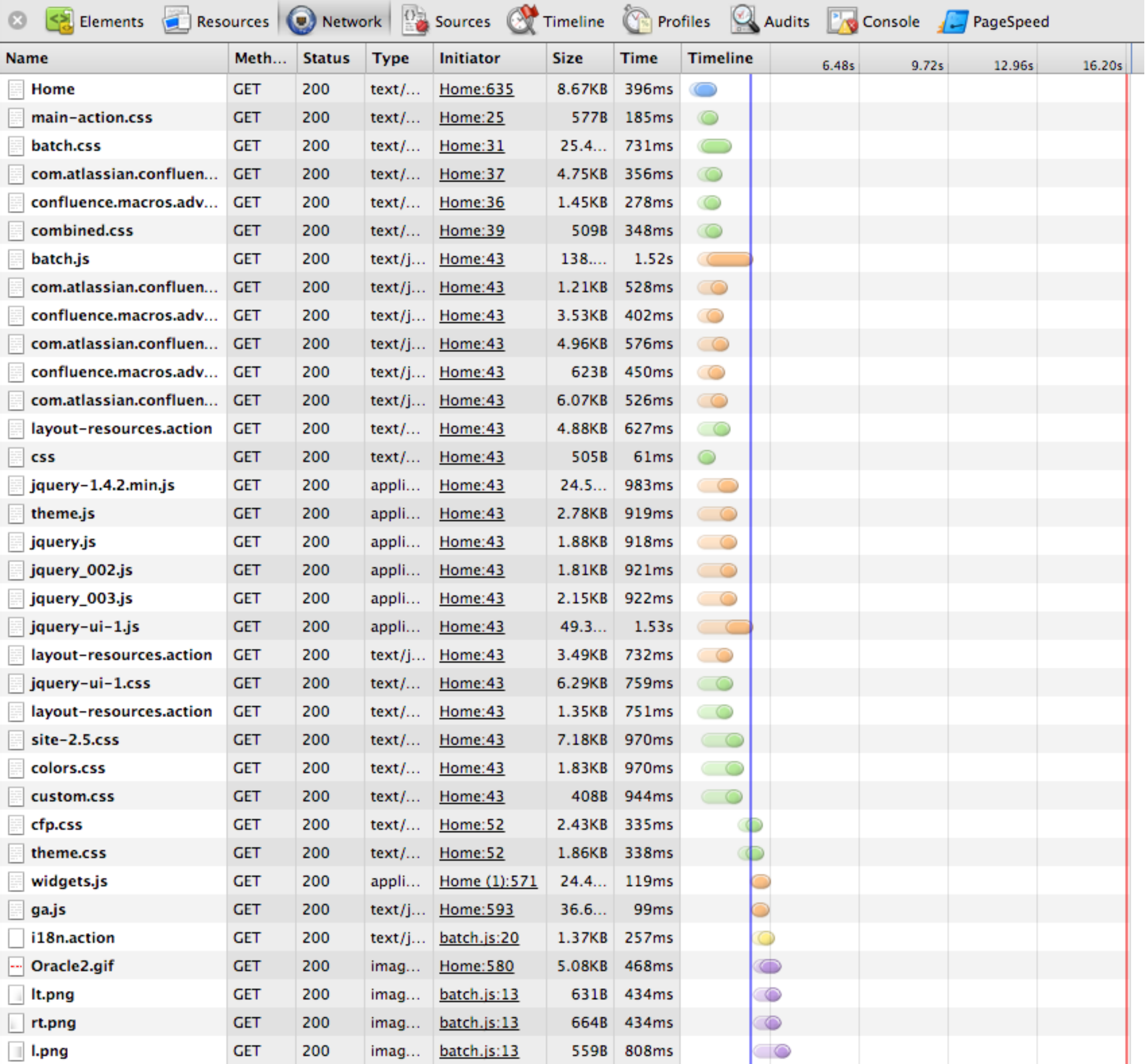
1. Unload the DOM
2. DNS resolution
3. Connection & TCP handshake
4. Send request, wait for response
5. Parse response
6. **Request sub-resources (see step 1)**
7. Execute scripts, apply CSS rules



x 84

(doh)

What does it take to load a web-page?



Name	Meth...	Status	Type	Initiator	Size	Time	Timeline
Home	GET	200	text/...	Home:635	8.67KB	396ms	
main-action.css	GET	200	text/...	Home:25	577B	185ms	
batch.css	GET	200	text/...	Home:31	25.4...	731ms	
com.atlassian.confluen...	GET	200	text/...	Home:37	4.75KB	356ms	
confluence.macros.adv...	GET	200	text/...	Home:36	1.45KB	278ms	
combined.css	GET	200	text/...	Home:39	509B	348ms	
batch.js	GET	200	text/j...	Home:43	138....	1.52s	
com.atlassian.confluen...	GET	200	text/j...	Home:43	1.21KB	528ms	
confluence.macros.adv...	GET	200	text/j...	Home:43	3.53KB	402ms	
com.atlassian.confluen...	GET	200	text/j...	Home:43	4.96KB	576ms	
confluence.macros.adv...	GET	200	text/j...	Home:43	623B	450ms	
com.atlassian.confluen...	GET	200	text/j...	Home:43	6.07KB	526ms	
layout-resources.action	GET	200	text/...	Home:43	4.88KB	627ms	
css	GET	200	text/...	Home:43	505B	61ms	
jquery-1.4.2.min.js	GET	200	appli...	Home:43	24.5...	983ms	
theme.js	GET	200	appli...	Home:43	2.78KB	919ms	
jquery.js	GET	200	appli...	Home:43	1.88KB	918ms	
jquery_002.js	GET	200	appli...	Home:43	1.81KB	921ms	
jquery_003.js	GET	200	appli...	Home:43	2.15KB	922ms	
jquery-ui-1.js	GET	200	appli...	Home:43	49.3...	1.53s	
layout-resources.action	GET	200	text/j...	Home:43	3.49KB	732ms	
jquery-ui-1.css	GET	200	text/...	Home:43	6.29KB	759ms	
layout-resources.action	GET	200	text/...	Home:43	1.35KB	751ms	
site-2.5.css	GET	200	text/...	Home:43	7.18KB	970ms	
colors.css	GET	200	text/...	Home:43	1.83KB	970ms	
custom.css	GET	200	text/...	Home:43	408B	944ms	
cfp.css	GET	200	text/...	Home:52	2.43KB	335ms	
theme.css	GET	200	text/...	Home:52	1.86KB	338ms	
widgets.js	GET	200	appli...	Home (1):571	24.4...	119ms	
ga.js	GET	200	text/j...	Home:593	36.6...	99ms	
i18n.action	GET	200	text/j...	batch.js:20	1.37KB	257ms	
Oracle2.gif	GET	200	imag...	Home:580	5.08KB	468ms	
lt.png	GET	200	imag...	batch.js:13	631B	434ms	
rt.png	GET	200	imag...	batch.js:13	664B	434ms	
l.png	GET	200	imag...	batch.js:13	559B	808ms	

devoxx.com

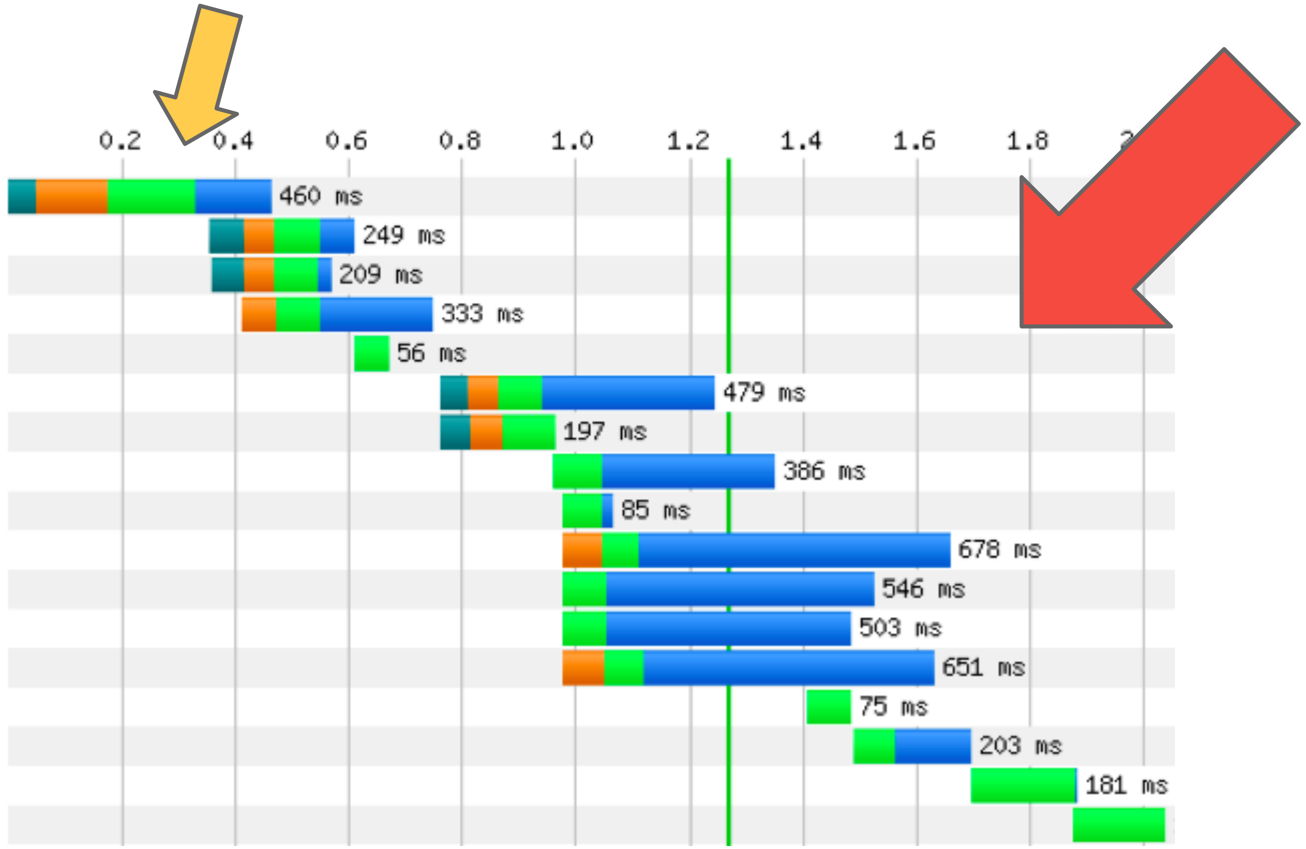
- 67 requests
- 3.83MB transferred

- DomContentLoaded: 2.48s
- onload: 16.20s

@igrigorik

What do we mean by "frontend" performance?

Page HTML



"Waterfall" of associated resources required to compose the page.

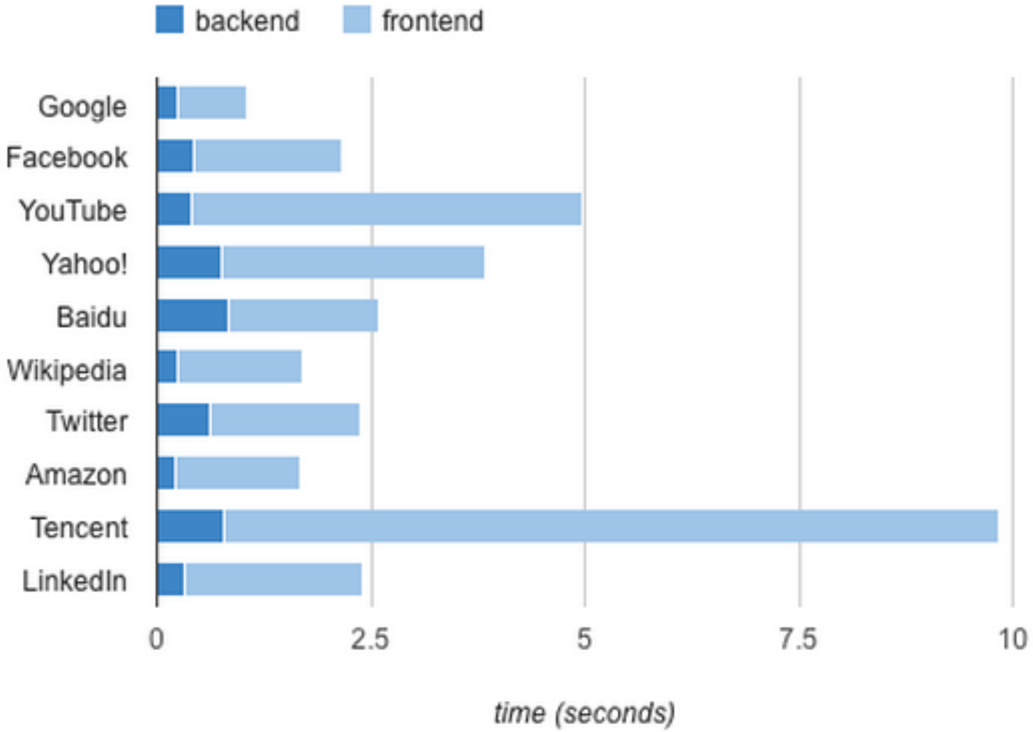
- **~84** requests
- **~1 MB** transferred

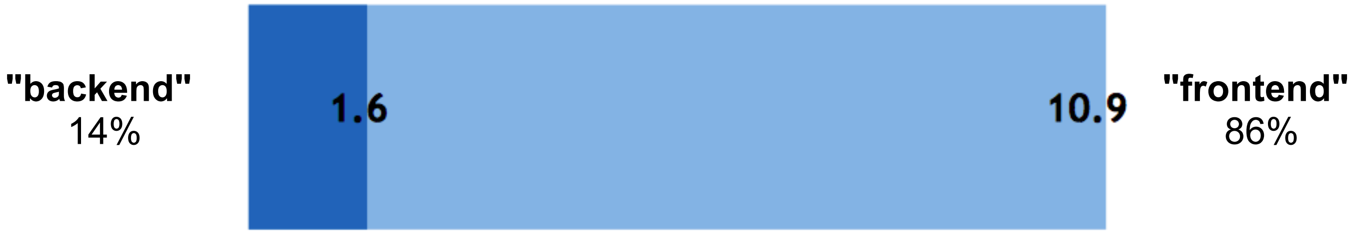
- Scheduled by the browser
- ... **"front-end" performance**

- Can we make the waterfall...
  - **Shorter? Thinner?**

@igrigorik

What do we mean by "frontend" performance?





Frontend this... backend that...

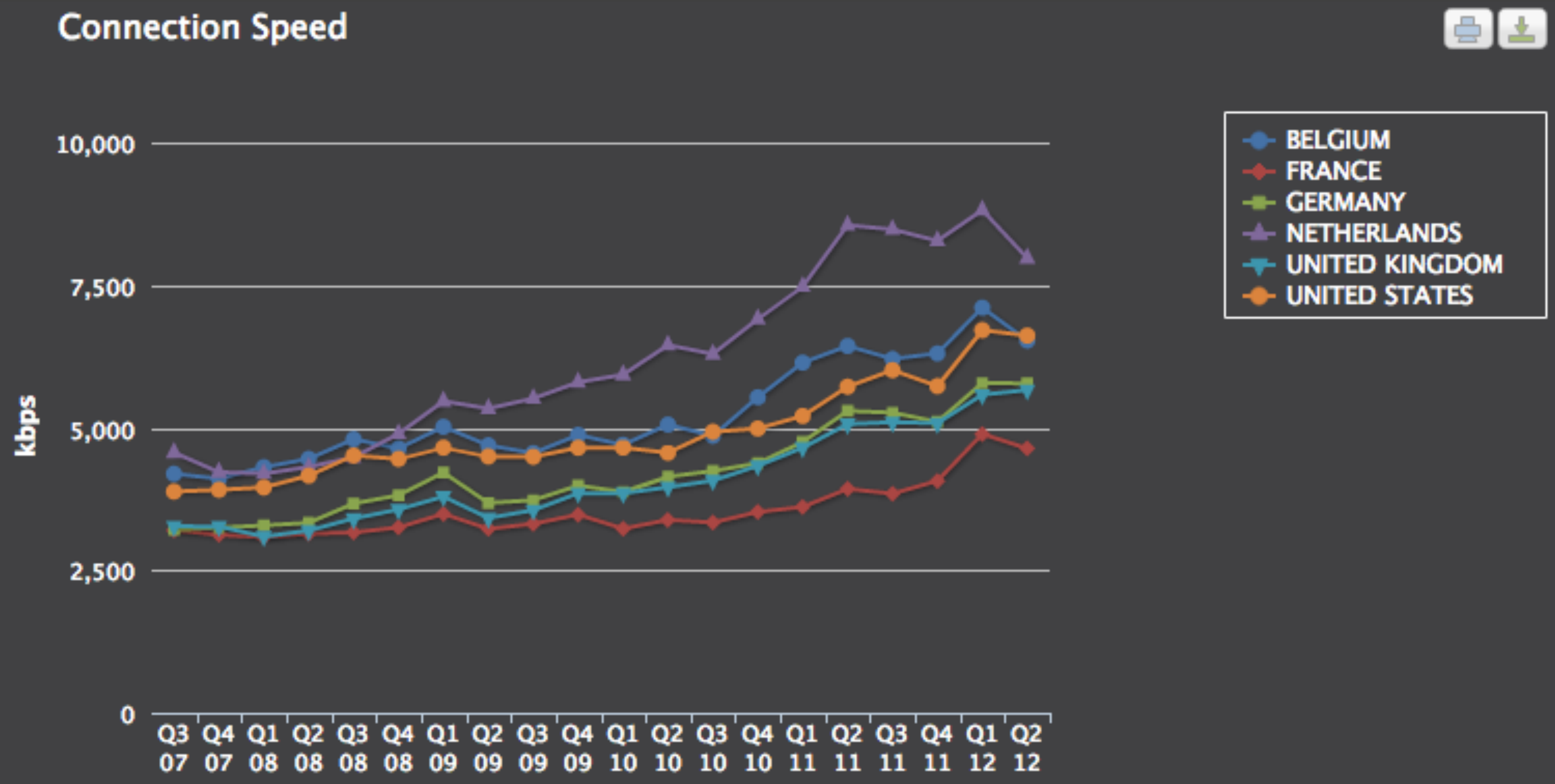
Focus on the lifetime of the page.

It just so happens that our pages are growing in complexity, and many resources are now scheduled by the browser. Not surprisingly, that's where you will find many optimization opportunities.

gLearn class - Steve Souders

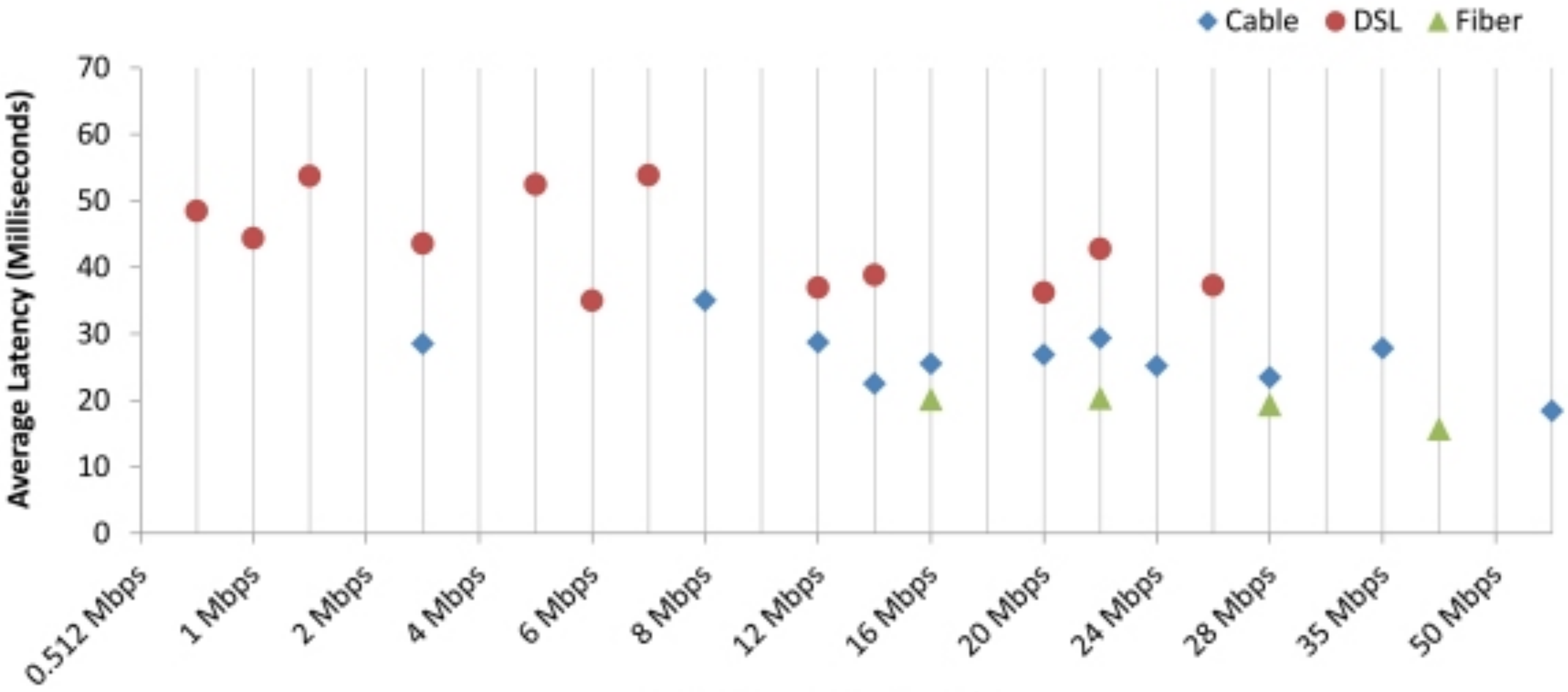
The network will save us?

Right, right? Or maybe not...



Average connection speed in Q1 2012: ***5000 kbps+***

State of the Internet - Akamai - 2007-2012



Fiber-to-the-home *services provided* ***18 ms*** *round-trip latency on average, while* ***cable-based*** *services averaged* ***26 ms****, and* ***DSL-based*** *services averaged* ***43 ms****. This compares to 2011 figures of 17 ms for fiber, 28 ms for cable and 44 ms for DSL.*

Measuring Broadband America - July 2012 - FCC

Worldwide: ~100ms

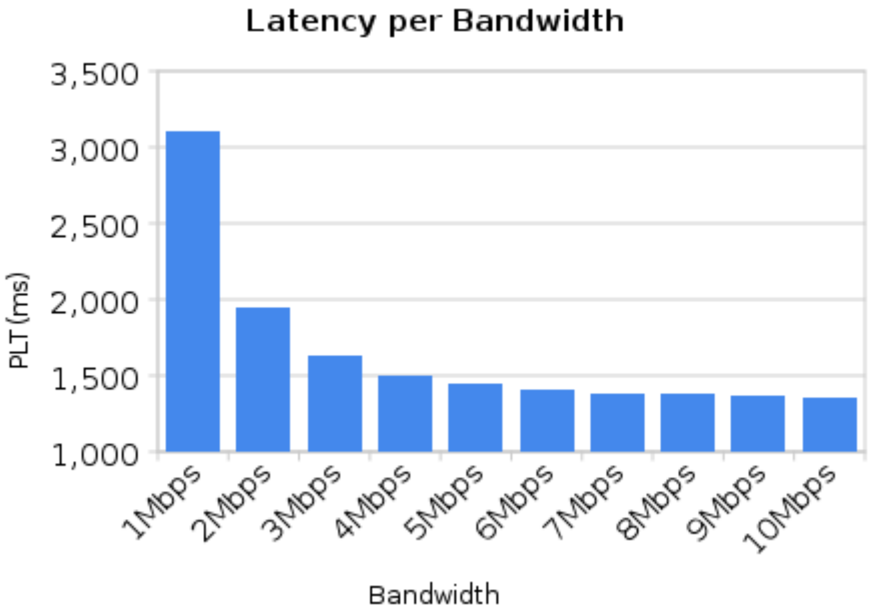
US: ~50~60ms

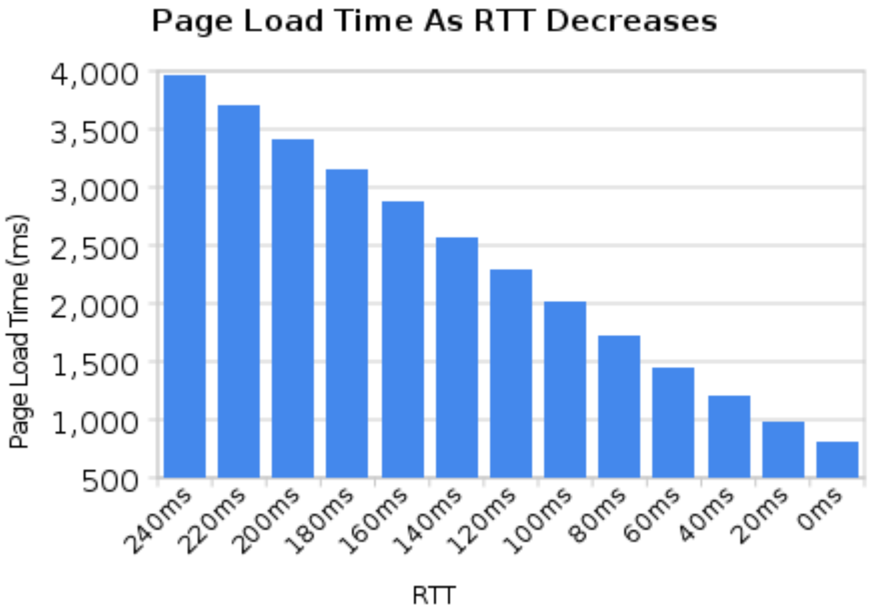
Average RTT to Google in 2012 is...

Bandwidth doesn't matter *(much)*

It's the latency, dammit!

PLT: latency *vs.* bandwidth





Average household in is running on a **5 mbps+** connection. Ergo, **average consumer would not see an improvement in page loading time by upgrading their connection.** (doh!)

Bandwidth doesn't matter (much) - Google

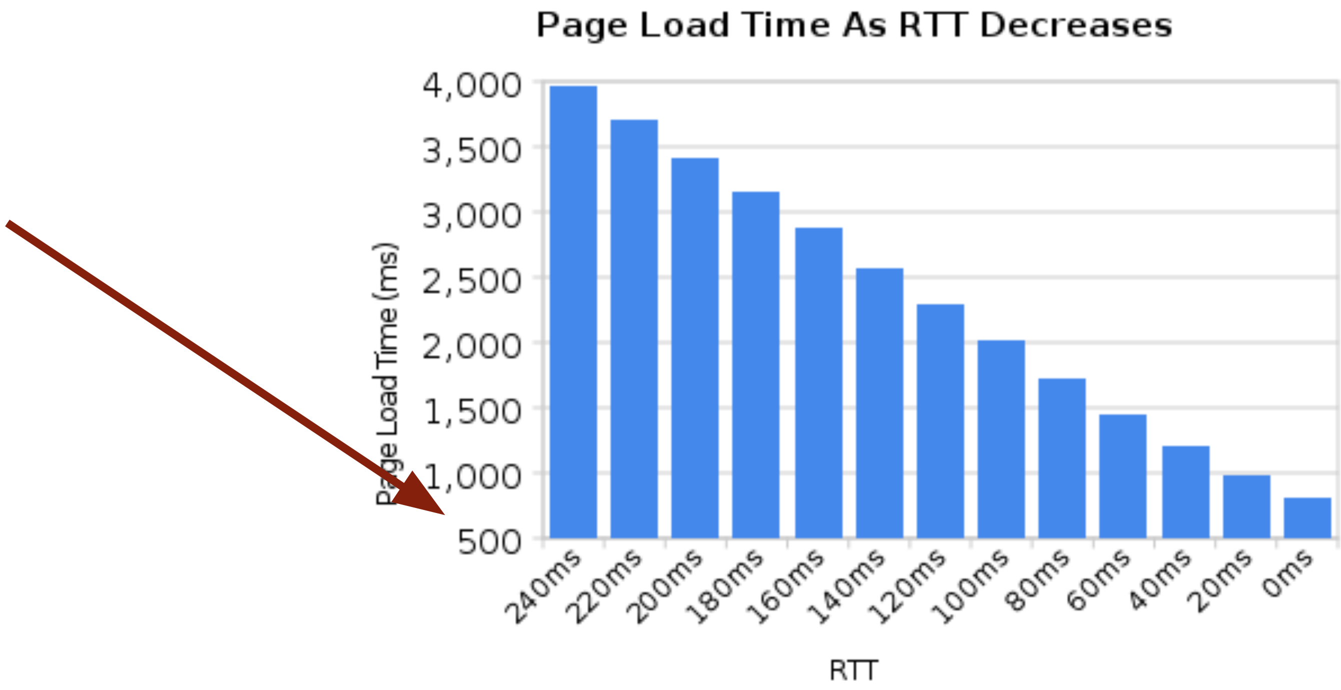
@igrigorik

Mobile, oh Mobile...

*Users of the **Sprint 4G network** can expect to experience average speeds of 3Mbps to 6Mbps download and up to 1.5Mbps upload with an **average latency of 150ms**. On the **Sprint 3G** network, users can expect to experience average speeds of 600Kbps - 1.4Mbps download and 350Kbps - 500Kbps upload with an **average latency of 400ms**.*

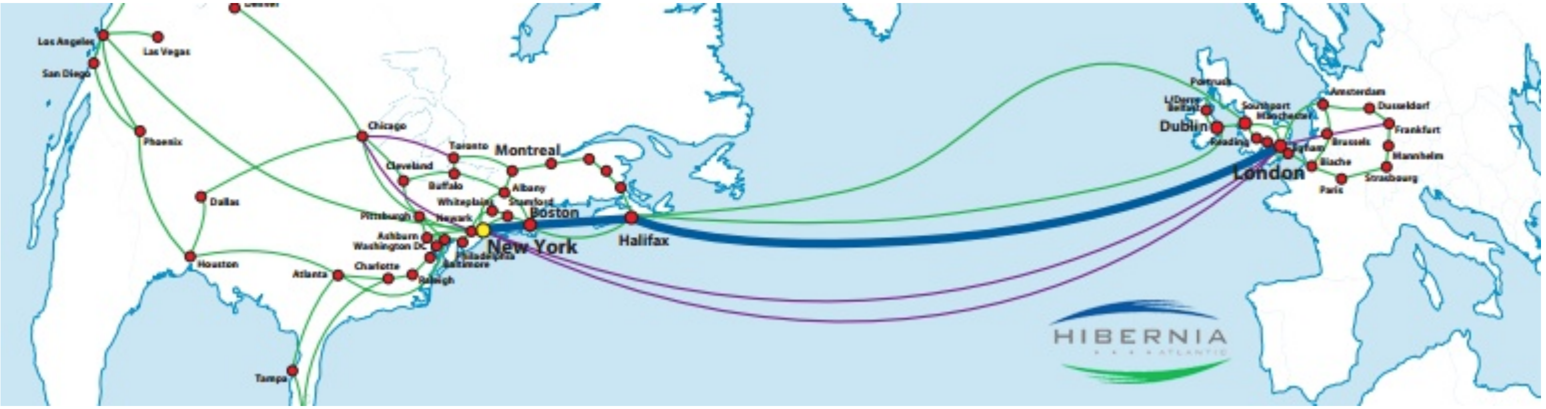
We stopped at 240ms!

(facepalm meme goes here...)



Virgin Mobile FAQ

- **Improving bandwidth is easy... ******
  - Still lots of unlit fiber
  - 60% of new capacity through upgrades
  - "Just lay more cable" ...
- **Improving latency is expensive... impossible?**
  - Bounded by the speed of light
  - We're already within a small constant factor of the maximum
  - Lay **shorter** cables!



$80M / ms

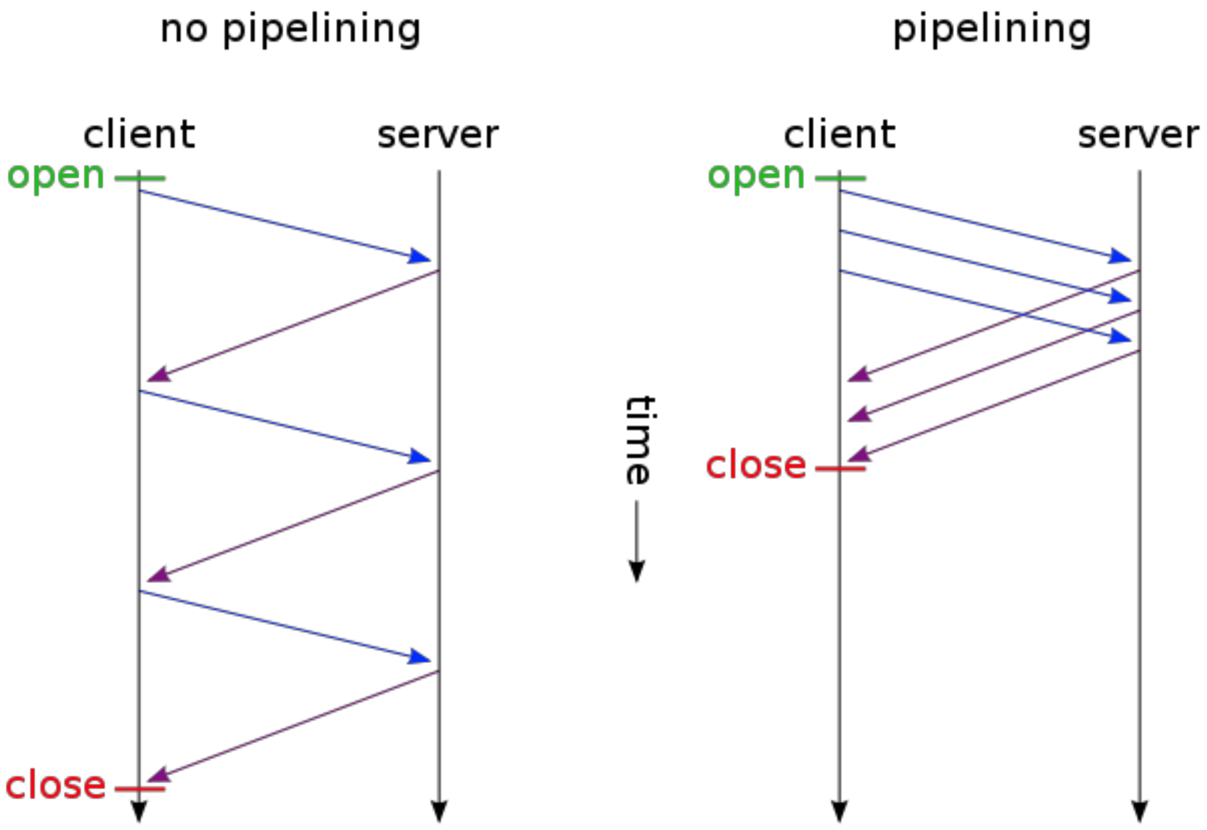
[Latency is the new Performance Bottleneck](#)

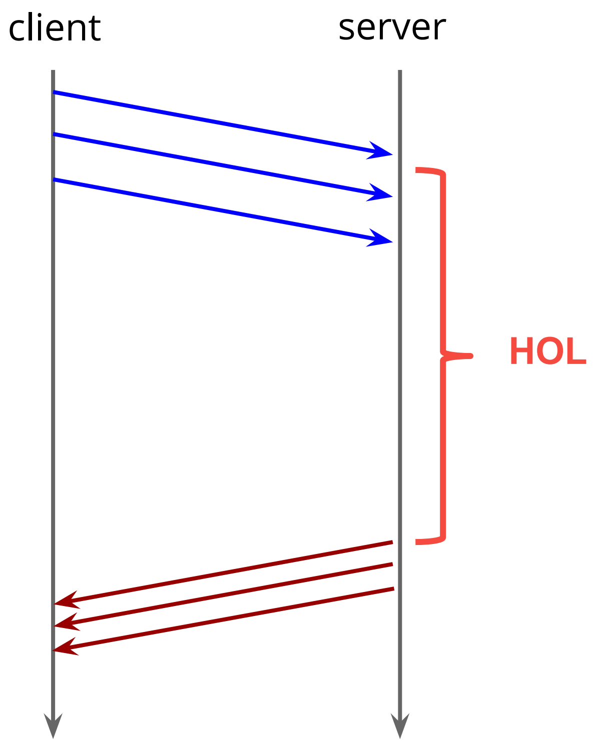
@igrigorik

Why is latency the problem?

Remember that HTTP thing... yeah...

HTTP doesn't have multiplexing!

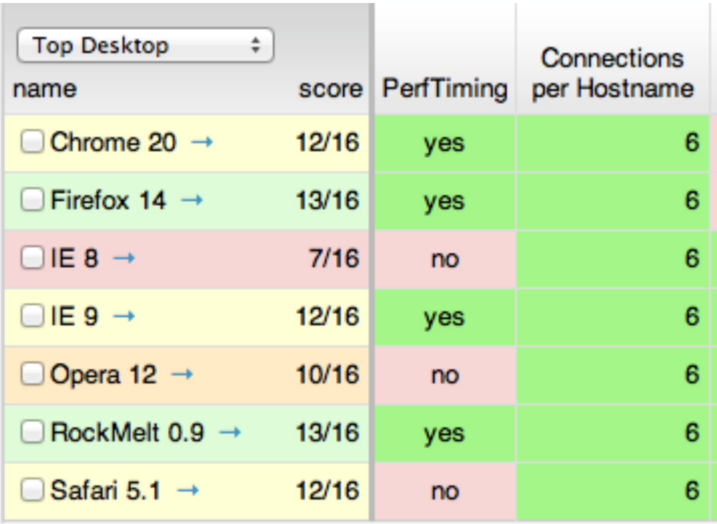




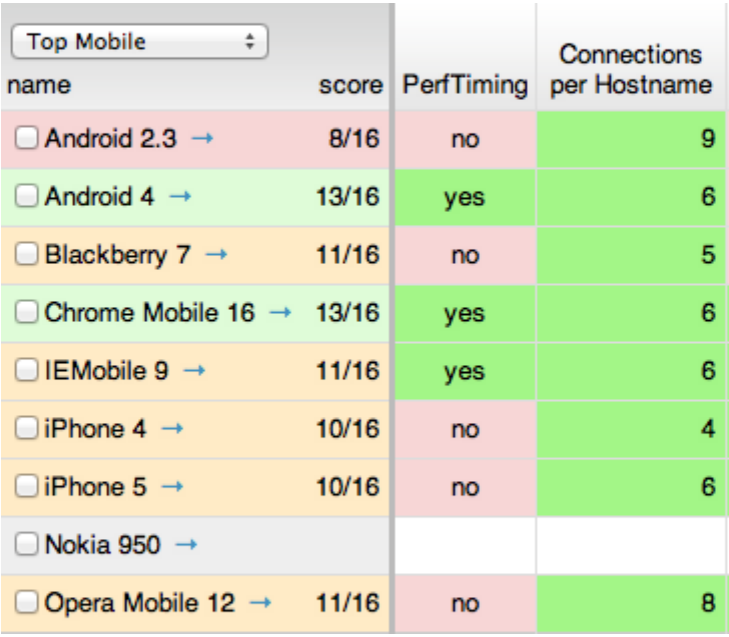
- **No pipelining:** request queuing
- **Pipelining*:** response queuing

- **Head of Line blocking**
    - It's a guessing game...
    - Should I wait, or should I pipeline?

Open multiple TCP connections!!!



Top Desktop name	score	PerfTiming	Connections per Hostname
☐ Chrome 20 →	12/16	yes	6
☐ Firefox 14 →	13/16	yes	6
☐ IE 8 →	7/16	no	6
☐ IE 9 →	12/16	yes	6
☐ Opera 12 →	10/16	no	6
☐ RockMelt 0.9 →	13/16	yes	6
☐ Safari 5.1 →	12/16	no	6



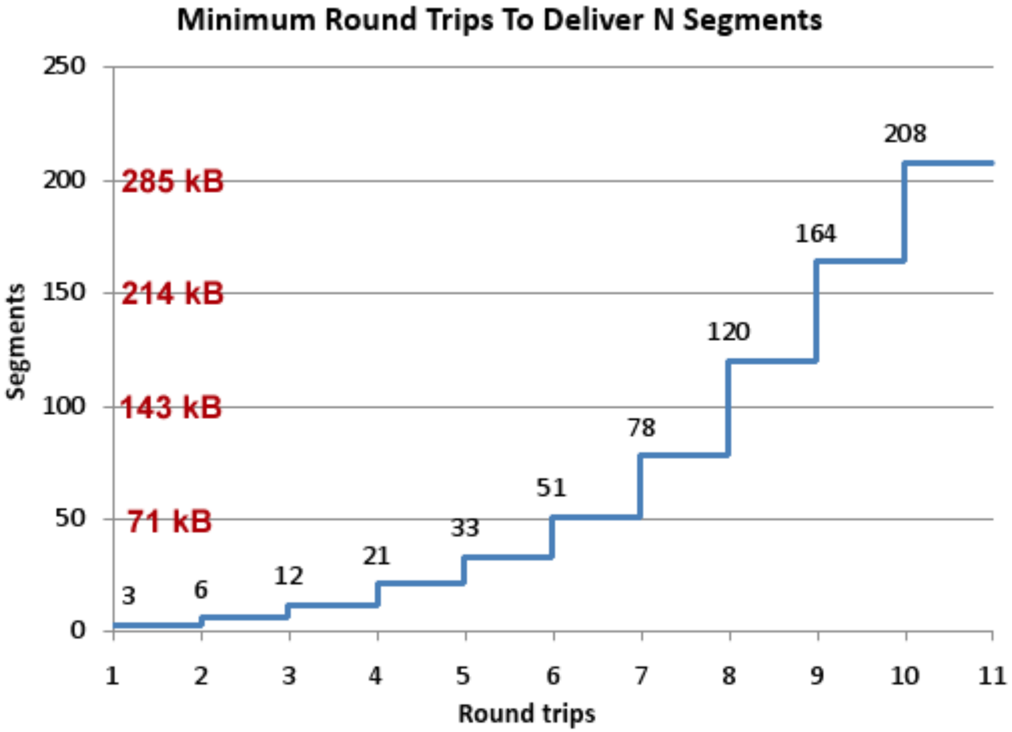
Top Mobile name	score	PerfTiming	Connections per Hostname
☐ Android 2.3 →	8/16	no	9
☐ Android 4 →	13/16	yes	6
☐ Blackberry 7 →	11/16	no	5
☐ Chrome Mobile 16 →	13/16	yes	6
☐ IEMobile 9 →	11/16	yes	6
☐ iPhone 4 →	10/16	no	4
☐ iPhone 5 →	10/16	no	6
☐ Nokia 950 →			
☐ Opera Mobile 12 →	11/16	no	8

- **6 connections per host** on Desktop
- **6 connections per host** on Mobile (recent builds)

So what, what's the big deal?

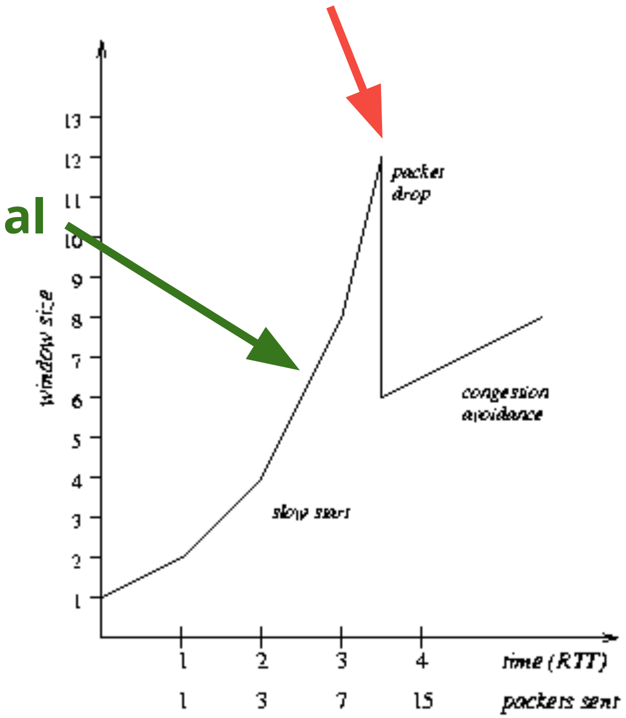
TCP Congestion Control & Avoidance...

- TCP is designed to probe the network to figure out the available capacity
- **TCP Slow Start** - feature, not a bug



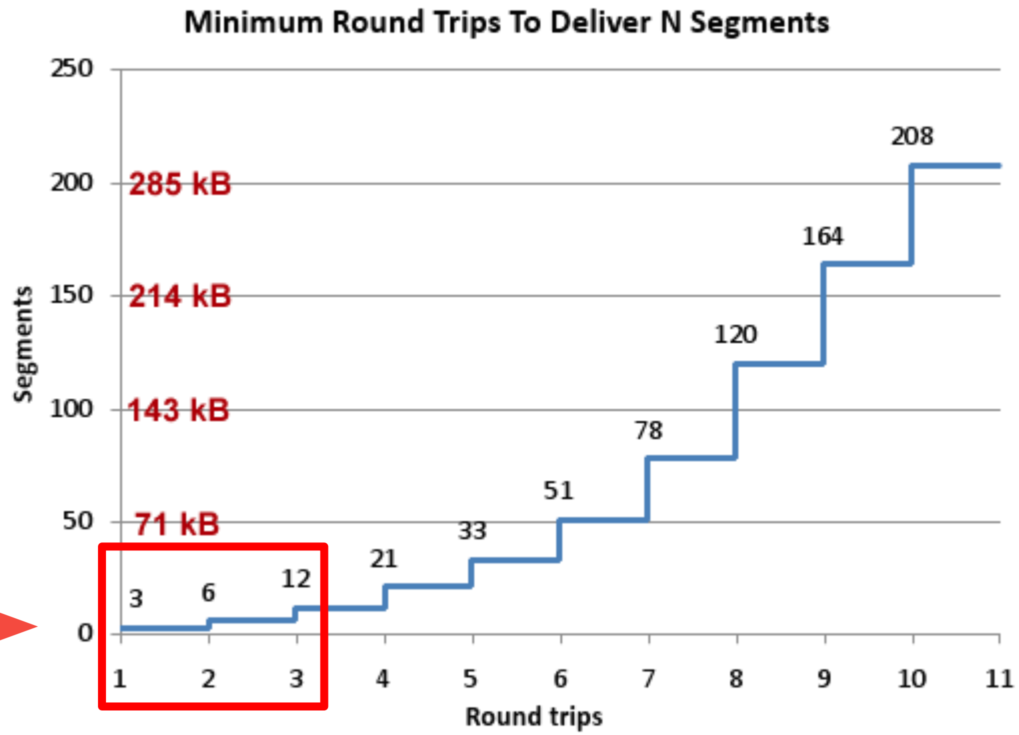
Packet Loss

Exponential growth



HTTP Archive says...

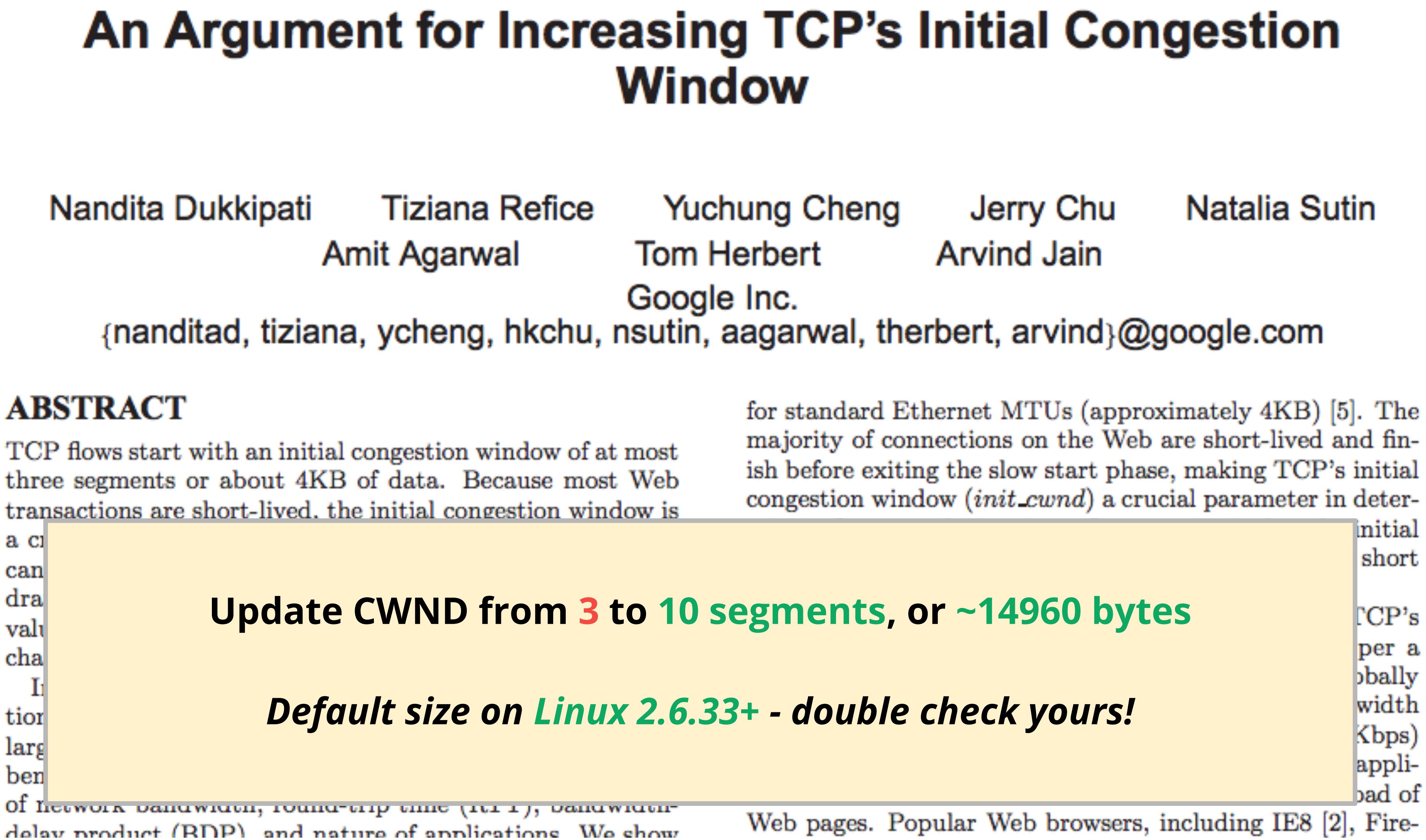
- 1098kb, 82 requests, ~30 hosts... ~**14kb per request!**
- Most HTTP traffic is composed of small, bursty, TCP flows



Where we want to be

You are here

1-3 RTT's



An Argument for Increasing TCP's Initial Congestion Window

Nandita Dukkipati Tiziana Refice Yuchung Cheng Jerry Chu Natalia Sutin
Amit Agarwal Tom Herbert Arvind Jain
Google Inc.
{nanditad, tiziana, ycheng, hkchu, nsutin, aagarwal, therbert, arvind}@google.com

ABSTRACT

TCP flows start with an initial congestion window of at most three segments or about 4KB of data. Because most Web transactions are short-lived, the initial congestion window is
a cr
can
dra
valu
cha
In
tion
larg
ben
of network bandwidth, round-trip time (RTT), bandwidth-
delay product (BDP), and nature of applications. We show

for standard Ethernet MTUs (approximately 4KB) [5]. The majority of connections on the Web are short-lived and finish before exiting the slow start phase, making TCP's initial congestion window (*init_cwnd*) a crucial parameter in deter-
initial
short
TCP's
per a
bally
width
Kbps)
appli-
oad of
Web pages. Popular Web browsers, including IE8 [2], Fire-

Update CWND from 3 to 10 segments, or ~14960 bytes

Default size on Linux 2.6.33+ - double check yours!

[An Argument for Increasing TCP's initial Congestion window](#)

It's here!

Let's talk about HTTP 2.0 / SPDY

Yes, it's coming!

SPDY is HTTP 2.0... *sort of...*

- HTTPBis Working Group met in Vancouver in late July
- Adopted **SPDY v2 as starting point** for HTTP 2.0

HTTP 2.0 Charter

1. **Done** Call for Proposals for HTTP/2.0
2. **Nov 2012** First WG draft of HTTP/2.0, based upon draft-mbelshe-httpbis-spdy-00
3. **Apr 2014** Working Group Last call for HTTP/2.0
4. **Nov 2014** Submit HTTP/2.0 to IESG for consideration as a Proposed Standard

http://lists.w3.org/Archives/Public/ietf-http-wg/2012JulSep/0971.html

> *It’s important to understand that SPDY isn’t being adopted as HTTP/2.0; rather, that it’s the* ***starting point*** *of our discussion, to avoid a laborious start from scratch.*

- Mark Nottingham (chair)

It is expected that HTTP/2.0 will...

- Substantially and measurably improve end-user perceived latency over HTTP/
- Address the "head of line blocking" problem in HTTP
- Not require multiple connections to a server to enable parallelism, thus improving its use of

Make things better

- Retain the semantics of HTTP/1.1, including (but not limited to)
    - HTTP methods
    - Status Codes
    - URIs
    - Header fields
- Clearly define how HTTP/2.0 interacts with HTTP/1.x
    - especially in intermediaries (both 2->1 and 1->2)

Build on HTTP 1.1

- Clearly identify any new extensibility points and policy for their appropriate use

Be extensible

… we're not replacing all of HTTP — the methods, status codes, and most of the headers you use today will be the same. Instead, we're ***re-defining how it gets used "on the wire" so it's more efficient****, and so that it is more gentle to the Internet itself ….*

- Mark Nottingham (chair)

A litany of problems.. and "workarounds"...

1. **Concatenating files**
   - JavaScript, CSS
   - Less modular, large bundles
2. **Spriting images**
   - What a pain...
3. **Domain sharding**
   - Congestion control who? 30+ parallel requests --- *Yeehaw!!!*
4. **Resource inlining**
   - TCP connections are expensive!
5. ...

All due to flaws in HTTP 1.1

So, what's a developer to do?

Fix HTTP 1.1! Use SPDY in the meantime...

SPDY in a Nutshell

- One TCP connection
- Request = Stream

- Streams are multiplexed
- Streams are prioritized

- Binary framing
- Length-prefixed

- Control frames
- Data frames

Control Frame:

```
+----------------------------------+
|C| Version(15bits) | Type(16bits) |
+----------------------------------+
| Flags (8)  |  Length (24 bits)   |
+----------------------------------+
|               Data               |
+----------------------------------+
```

Data Frame:

```
+----------------------------------+
|D|       Stream-ID (31bits)       |
+----------------------------------+
| Flags (8)  |  Length (24 bits)   |
+----------------------------------+
|               Data               |
+----------------------------------+
```

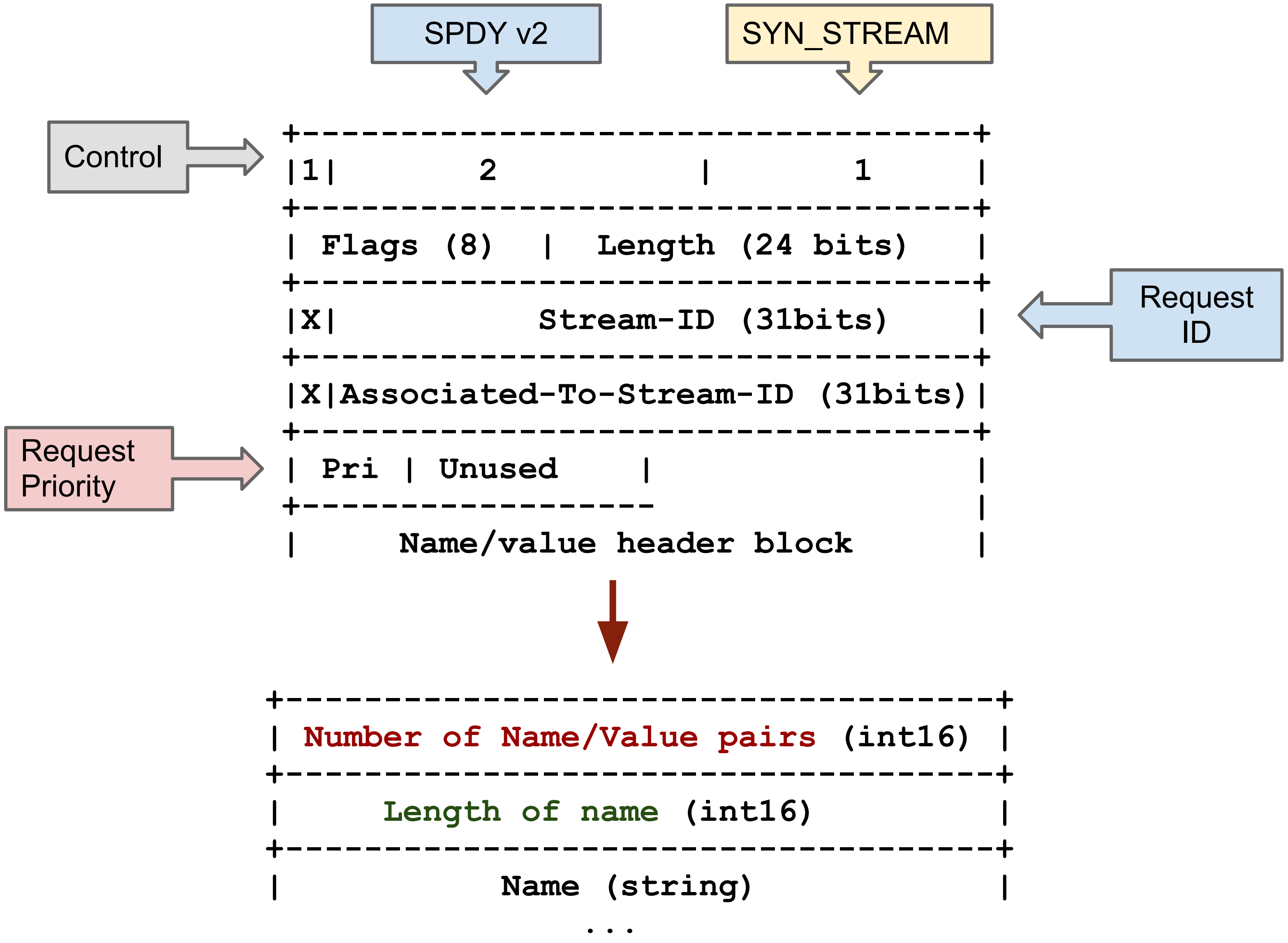
SYN_STREAM

- Server SID: even
- Client SID: odd

- Associated-To: push *
- Priority: higher, better

- Length prefixed headers

**** Much of this may (will, probably) change*



SPDY v2

SYN_STREAM

Control

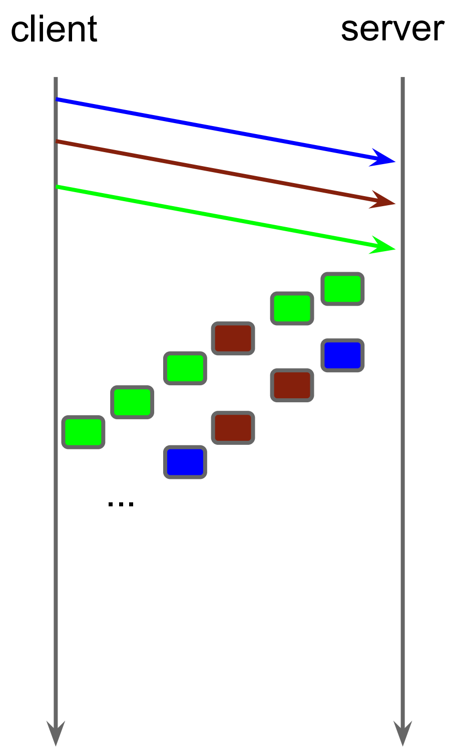
Request ID

Request Priority

```
+----------------------------------+
|1|       2          |      1      |
+----------------------------------+
| Flags (8)  |  Length (24 bits)   |
+----------------------------------+
|X|          Stream-ID (31bits)    |
+----------------------------------+
|X|Associated-To-Stream-ID (31bits)|
+----------------------------------+
| Pri | Unused    |                |
+------------------                |
|     Name/value header block      |
```

```
+------------------------------------+
| Number of Name/Value pairs (int16) |
+------------------------------------+
|     Length of name (int16)         |
+------------------------------------+
|           Name (string)            |
                 ...
```

SPDY in action



- Full request & response multiplexing
- Mechanism for request prioritization

- Many small files? No problem
- Higher TCP window size
- More efficient use of server resources
- TCP Fast-retransmit for faster recovery

Anti-patterns

- Domain sharding
    - *Now we need to unshard - doh!*

Speaking of HTTP Headers...

```
curl -vv -d'{"msg":"oh hai"}' http://www.igvita.com/api

> POST /api HTTP/1.1
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0)
libcurl/7.24.0 OpenSSL/0.9.8r zlib/1.2.5
> Host: www.igvita.com
> Accept: */*
> Content-Length: 16
> Content-Type: application/x-www-form-urlencoded

< HTTP/1.1 204
< Server: nginx/1.0.11
< Content-Type: text/html; charset=utf-8
< Via: HTTP/1.1 GWA
< Date: Thu, 20 Sep 2012 05:41:30 GMT
< Expires: Thu, 20 Sep 2012 05:41:30 GMT
< Cache-Control: max-age=0, no-cache
....
```

- Average request / response header overhead: **800 bytes**

- No compression for headers in HTTP!
- Huge overhead

- **Solution:** compress the headers!
  - gzip all the headers
  - header registry
  - connection-level *vs.* request-level

- **Complication:** intermediate proxies **

SPDY Server Push

Premise: server can push resources to client

- ***Concern: but I don't want the data! Stop it!***
  - Client can cancel SYN_STREAM if it doesn't the resource
- Resource goes into browsers cache (no client API)

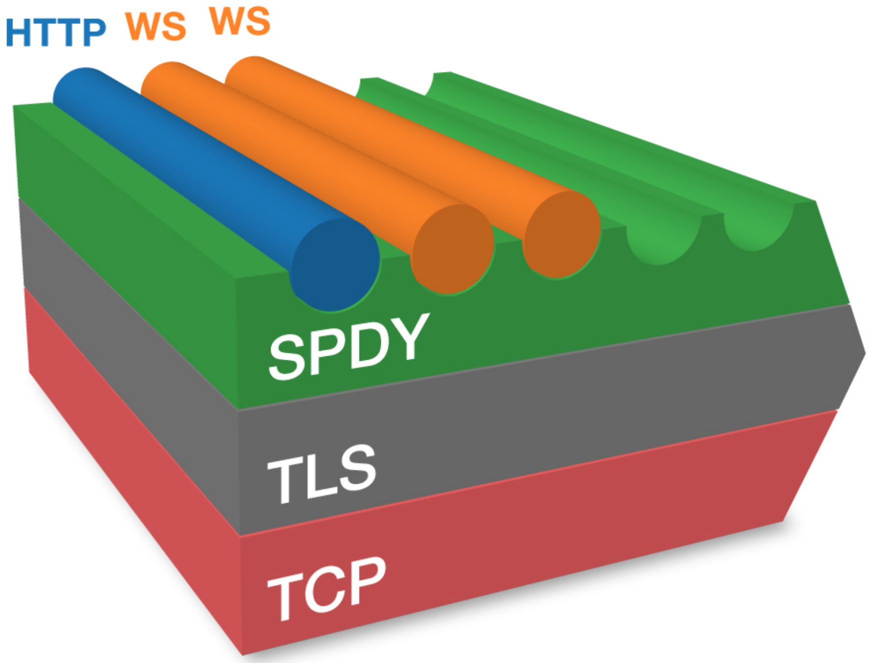
Newsflash: we are already using "server push"

- Today, we call it "inlining"
- Inlining works for unique resources, bloats pages otherwise

Advanced use case: forward proxy (ala Amazon's Silk)

- Proxy has full knowledge of your cache, can intelligently push data to the client

Encrypt all the things!!!



SPDY runs over TLS

- Philosophical reasons
- Political reasons
- **Pragmatic + deployment reasons - Bing!**

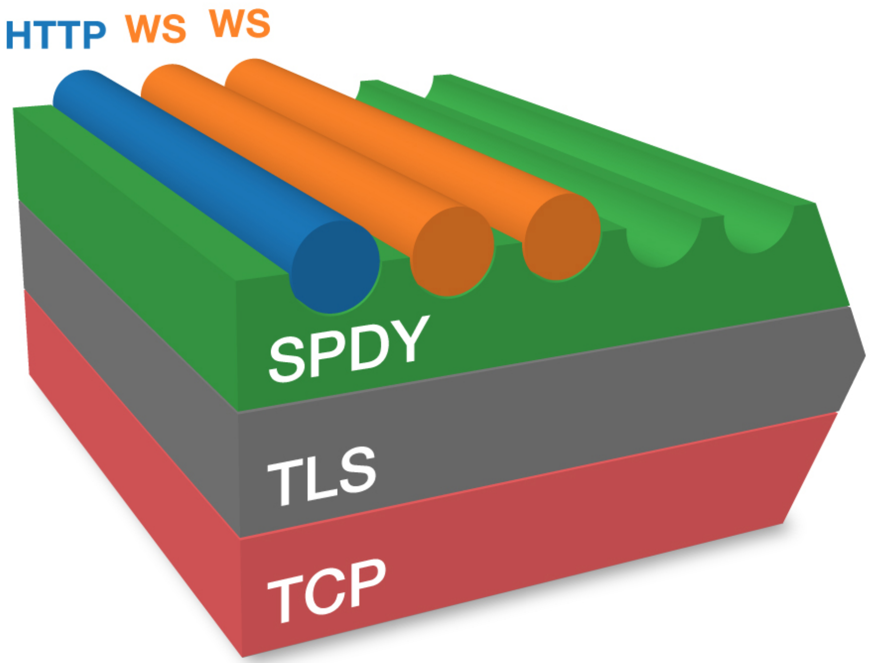
Observation: intermediate proxies get in the way

- Some do it intentionally, many unintentionally
- *Ex: Antivirus / Packet Inspection / QoS / ...*

SDHC / WebSocket: No TLS works.. in *80-90% of cases*

- 10% of the time things fail for no discernable reason
- In practice, any large WS deployments run as WSS

But isn't TLS *slow?*



CPU

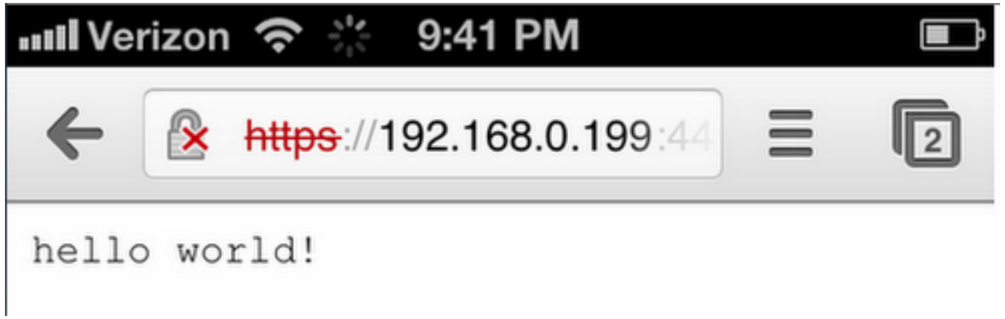
> *"On our production frontend machines, **SSL/TLS accounts for less than 1% of the CPU load**, less than 10KB of memory per connection and less than 2% of network overhead."*
>
> \- Adam Langley (Google)

Latency

- TLS Next Protocol Negotiation
  - Protocol negotiation as part of TLS handshake
- TLS False Start
  - reduce the number of RTTS for full handshake from two to one
- TLS Fast Start
  - reduce the RTT to zero
- Session resume, ...

Who supports SPDY?

- **Chrome**, since forever..
  - Chrome on Android + iOS
- **Firefox 13+**
- **Opera 12.10+**



hello world!

Server

- mod_spdy (Apache)
- nginx
- Jetty, Netty
- node-spdy
- ...

3rd parties

- Twitter
- Wordpress
- Facebook*

- Akamai
- Contendo
- F5 SPDY Gateway
- Strangeloop
- ...

All Google properties

- Search, GMail, Docs
- GAE + SSL users
- ...

SPDY FAQ

- **Q: Do I need to modify my site to work with SPDY / HTTP 2.0?**
- **A:** No. But you can optimize for it.

- **Q: How do I optimize the code for my site or app?**
- **A:** "Unshard", stop worrying about silly things (like spriting, etc).

- **Q: Any server optimizations?**
- **A:** Yes!
  - CWND = 10
  - Check your SSL certificate chain (length)
  - TLS resume, terminate SSL connections closer to the user
  - Disable TCP slow start on idle

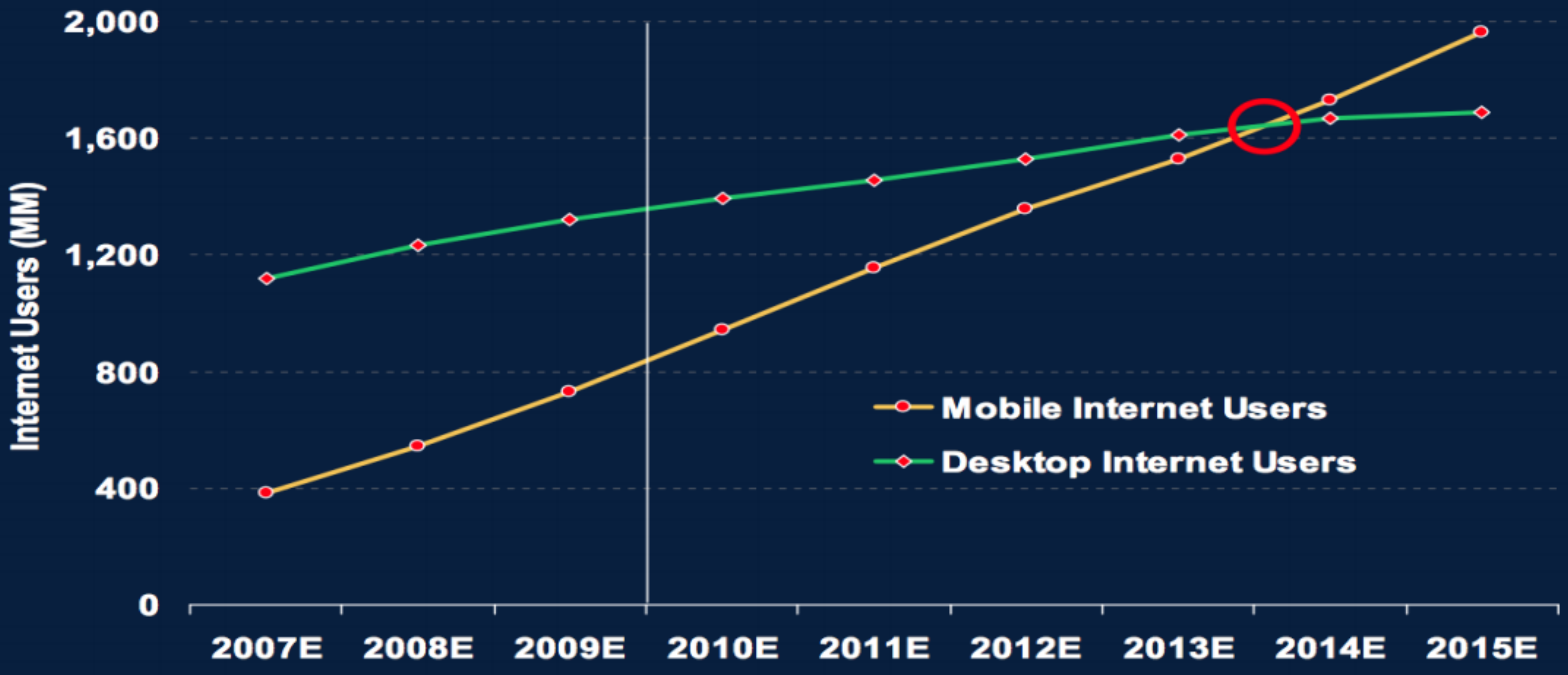
- **Q: Sounds complicated...**
- **A:** mod_spdy, nginx, GAE!

Mobile... oh mobile...

We still have a lot to learn when it comes to mobile

Mobile Users > Desktop Internet Users Within ~~5 Years~~ 2 Years

Global Mobile vs. Desktop Internet User Projection, 2007 – 2015E



Source: Morgan Stanley Research.

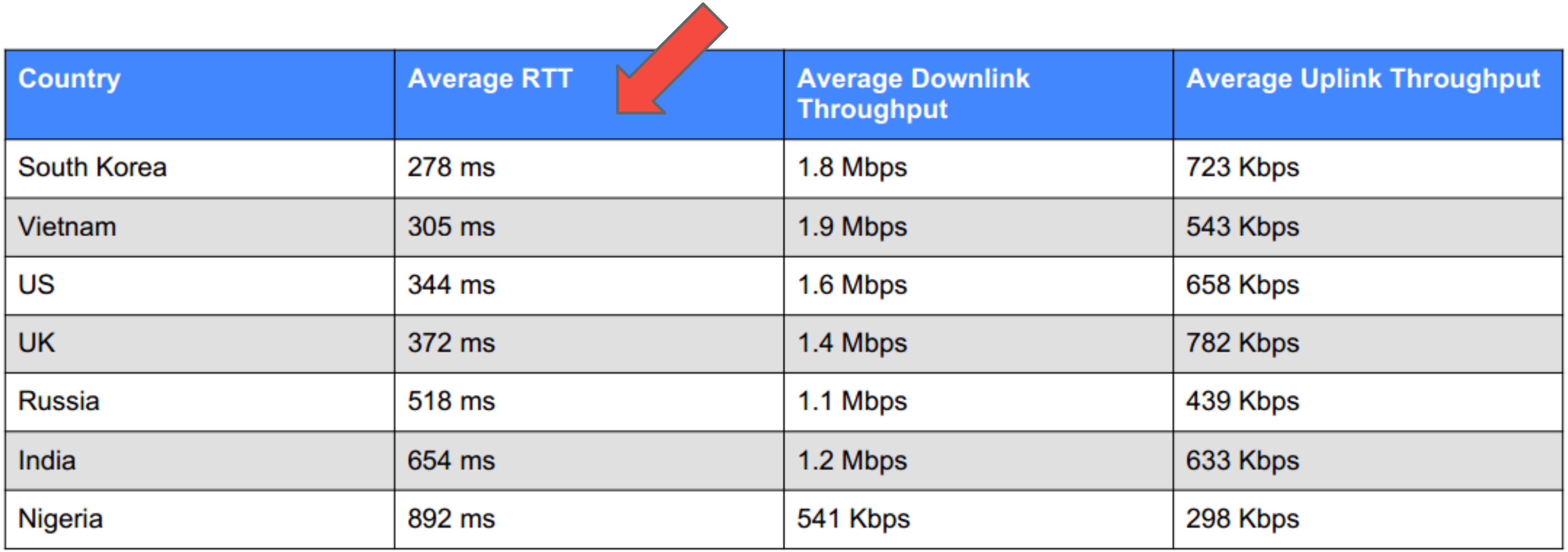
For many, mobile is the one and only internet device

Country	Mobile-only users
Egypt	70%
India	59%
South Africa	57%
Indonesia	44%
United States	25%

onDevice Research

Average RTT & downlink / uplink speeds

Ouch!



Country	Average RTT	Average Downlink Throughput	Average Uplink Throughput
South Korea	278 ms	1.8 Mbps	723 Kbps
Vietnam	305 ms	1.9 Mbps	543 Kbps
US	344 ms	1.6 Mbps	658 Kbps
UK	372 ms	1.4 Mbps	782 Kbps
Russia	518 ms	1.1 Mbps	439 Kbps
India	654 ms	1.2 Mbps	633 Kbps
Nigeria	892 ms	541 Kbps	298 Kbps

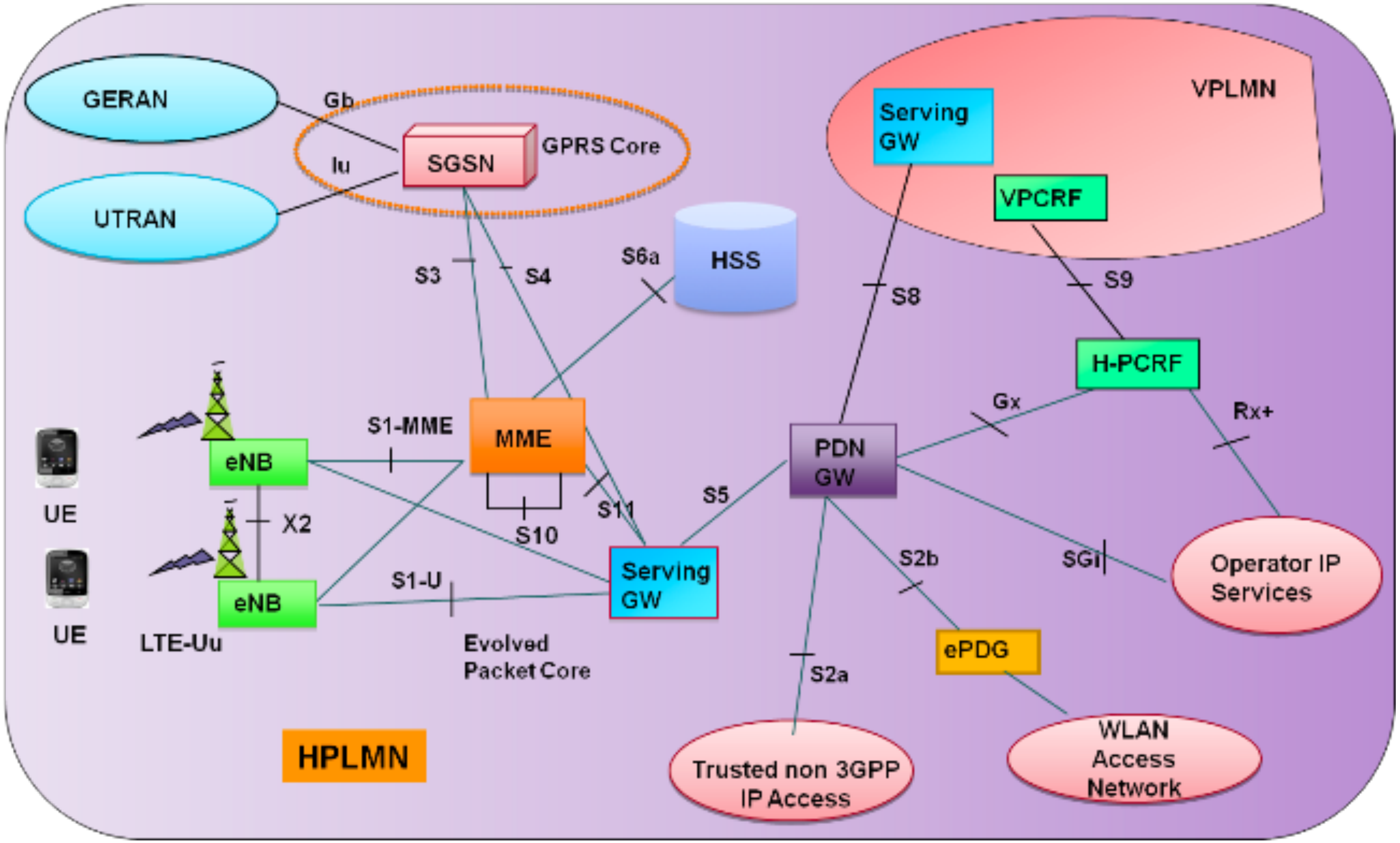
These numbers don't look that much different from the Sprint / Virgin latency numbers we saw earlier! Hmm...

Mobile is a land of contradictions...

We want point-to-point links	**But** we broadcast to everyone via a shared channel
We want to pretend mobile networks are no different	**But** the physical layer and delivery is completely different
We want "always on" radio performance	**But** we want long battery life from our devices
We want ubiquitous coverage	**But** we need to build smaller cells for high throughput
...	...

And the list goes on, and on, and on...

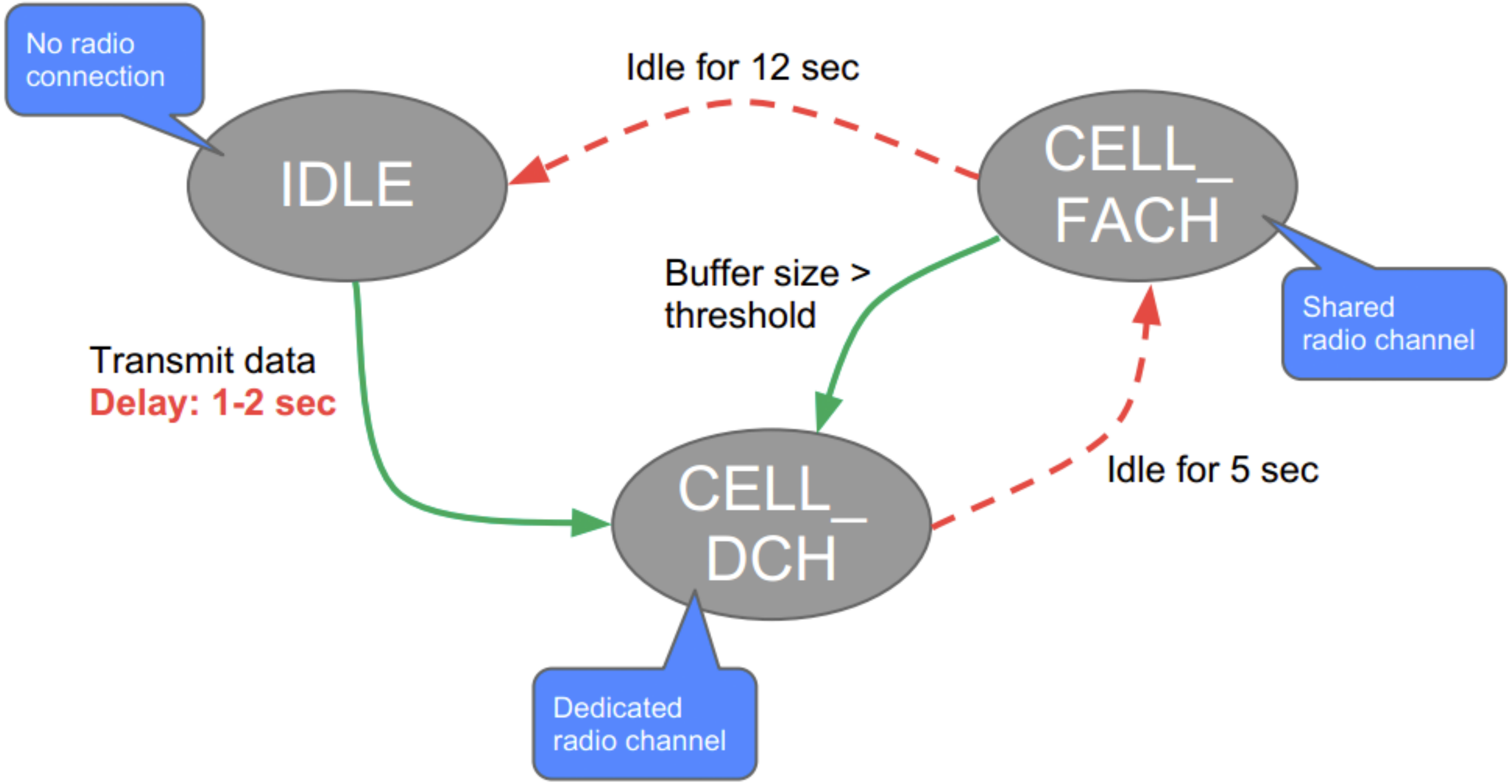
4G Network under the hood...



It's complicated... and we don't have all day. ***BUT****, the point is, we can't ignore it.*

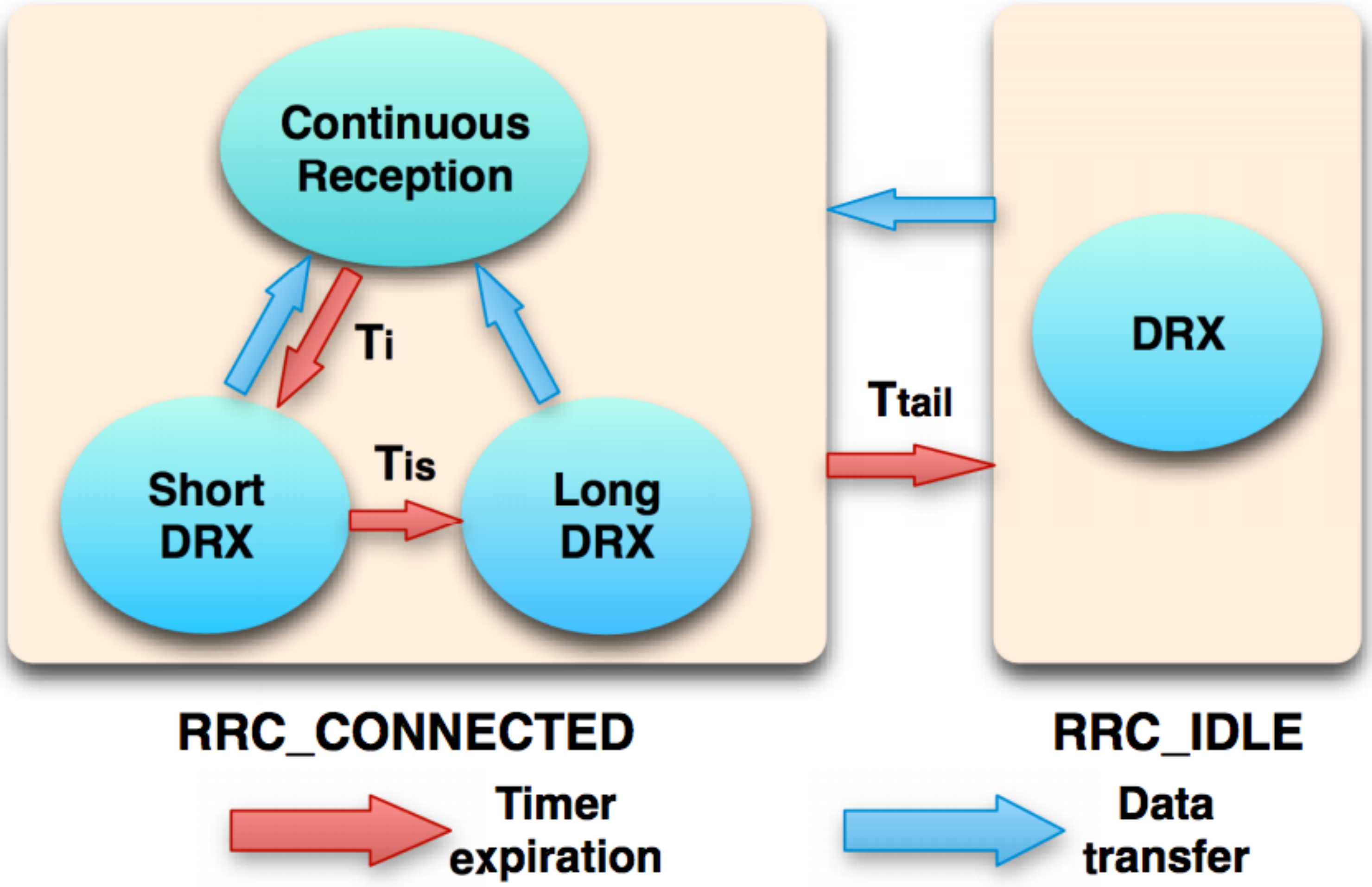
Designing a great mobile applications requires that you think about how to respect the limits, restrictions (and advantages) of a mobile device.

Mobile radio 101: 3G Radio Resource Control (RRC)



- *RRC state controlled by the network*
- *Gateway schedules your uplink & downlink intervals*
- *Radio cycles between 3 power states*
  - *Idle*
  - *Low TX power*
  - *High TX power*

Mobile radio 101: 4G Radio Resource Control (RRC)

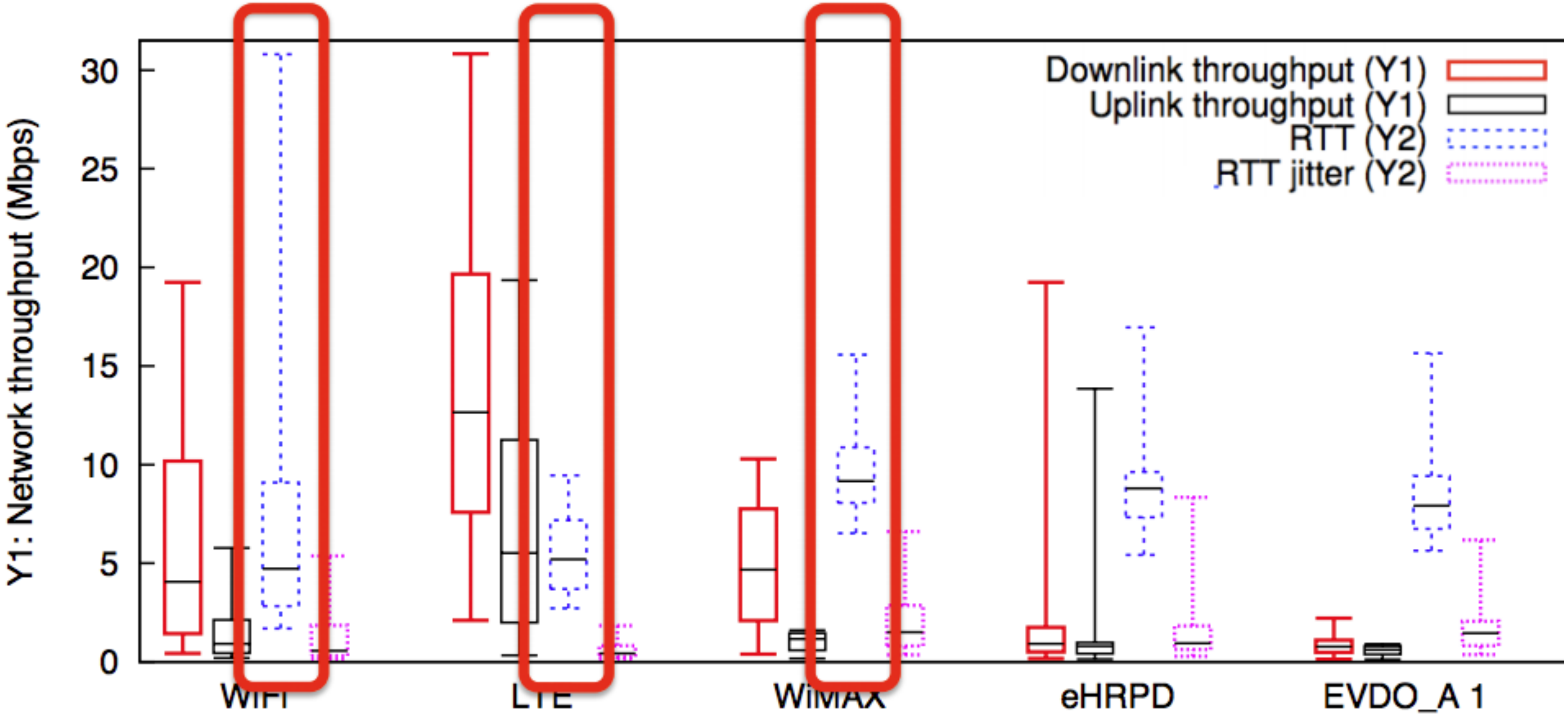


- *Similar to 3G, but different*

- *Connected & Idle states*
- *DRX cycles change receive timeouts*

- *4G Goals*
  - *faster state transitions*
  - ***aka, lower latency***
  - *better throughput*

Mobile radio 101: 4G Radio Resource Control (RRC)



- ***LTE median RTT is 70 ms***
- *Similar RTT profile to WiFi networks*

[Performance characteristics of 4G LTE Networks]

@igrigorik

Uh huh... Yeah, tell me more...

1. **Latency and variability are both very high on mobile networks**

2. **4G networks will improve latency, but...**
   a. We still have a long way to go until everyone is on 4G
   b. And 3G is definitely not going away anytime soon
   c. Ergo, latency and variability in latency ***is*** your problem

3. **What can we do about it?**
   a. Think back to TCP / SPDY...
   b. Re-use connections, use pipelining
   c. Download resources in bulk, avoid waking up the radio
   d. Compress resources
   e. Cache

@igrigorik

The browser *is* trying to help you!

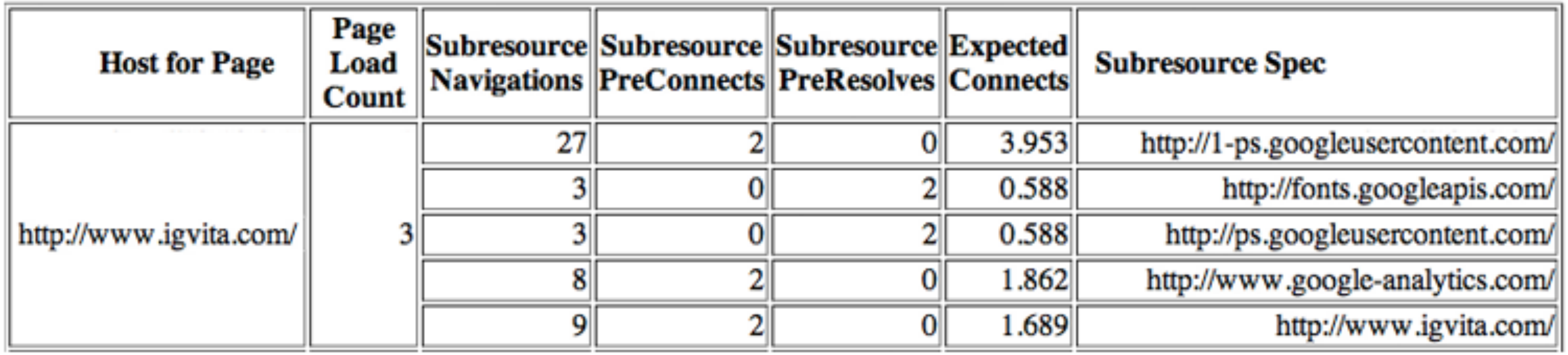
It is trying really hard... help it, help you!

(Chrome) Network Stack

An average page has grown to **1059 kB** (over 1MB!) and is now composed of **80+ subresources**.

- **DNS prefetch** - pre-resolve hostnames before we make the request
- **TCP preconnect** - establish connection before we make the request
- **Pooling & re-use** - leverage keep-alive, re-use existing connections (6 per host)
- **Caching** - fastest request is request not made (sizing, validation, eviction, etc)

Ex, Chrome learns subresource domains:



Host for Page	Page Load Count	Subresource Navigations	Subresource PreConnects	Subresource PreResolves	Expected Connects	Subresource Spec
http://www.igvita.com/	3	27	2	0	3.953	http://1-ps.googleusercontent.com/
		3	0	2	0.588	http://fonts.googleapis.com/
		3	0	2	0.588	http://ps.googleusercontent.com/
		8	2	0	1.862	http://www.google-analytics.com/
		9	2	0	1.689	http://www.igvita.com/

Chrome Networking: DNS Prefetch & TCP Preconnect

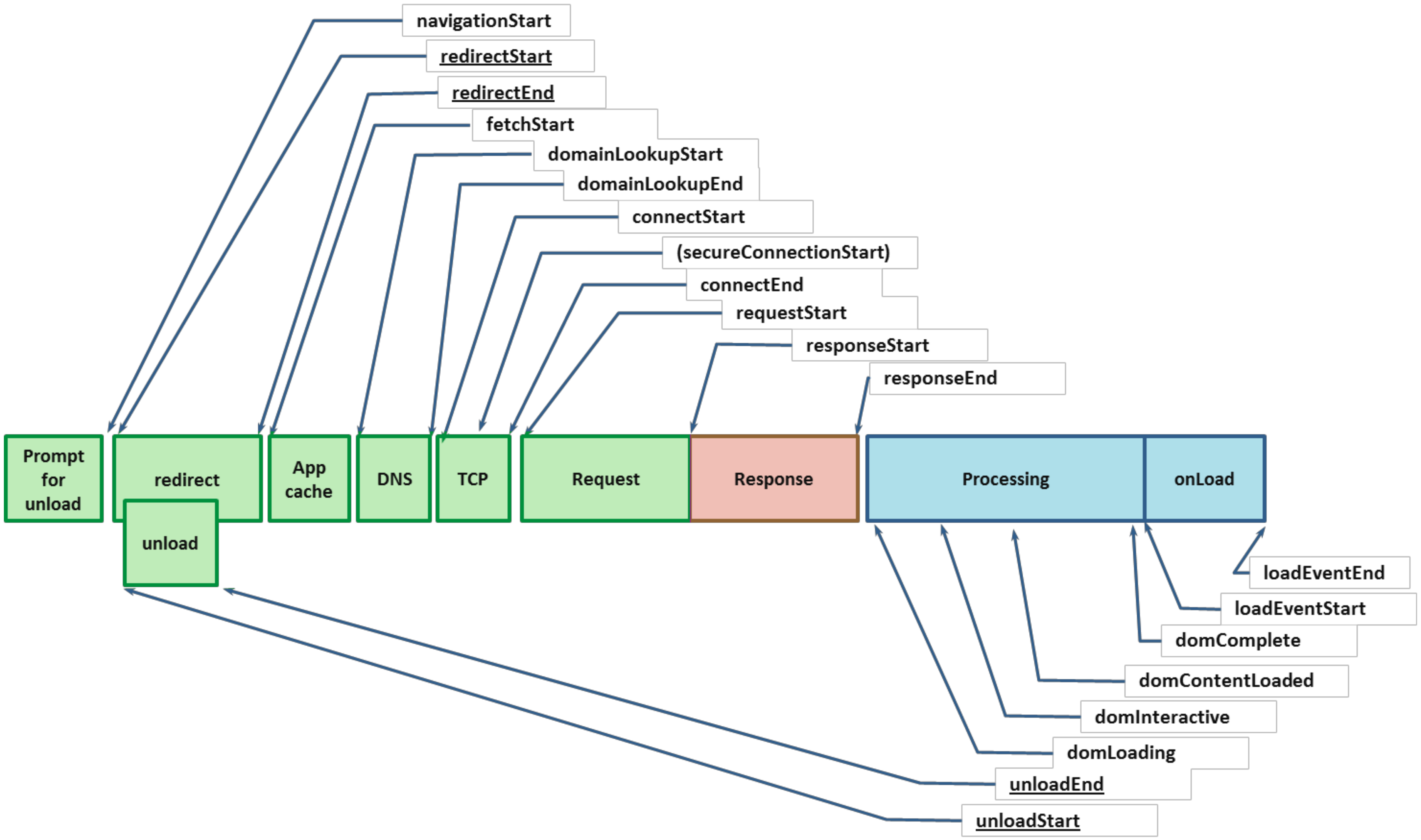
(Chrome) Network Stack

- **chrome://predictors** - omnibox predictor stats (check 'Filter zero confidences')
- **chrome://net-internals#sockets** - current socket pool status
- **chrome://net-internals#dns** - Chrome's in-memory DNS cache
- **chrome://histograms/DNS** - histograms of your DNS performance
- **chrome://dns** - startup prefetch list and subresource host cache

```
enum ResolutionMotivation {
    MOUSE_OVER_MOTIVATED,      // Mouse-over link induced resolution.
    PAGE_SCAN_MOTIVATED,       // Scan of rendered page induced resolution.
    LINKED_MAX_MOTIVATED,      // enum demarkation above motivation from links.
    OMNIBOX_MOTIVATED,         // Omni-box suggested resolving this.
    STARTUP_LIST_MOTIVATED,    // Startup list caused this resolution.
    EARLY_LOAD_MOTIVATED,      // In some cases we use the prefetcher to warm up the connection
    STATIC_REFERAL_MOTIVATED,  // External database suggested this resolution.
    LEARNED_REFERAL_MOTIVATED, // Prior navigation taught us this resolution.
    SELF_REFERAL_MOTIVATED,    // Guess about need for a second connection.
    // ...
};
```

Chrome Networking: DNS Prefetch & TCP Preconnect

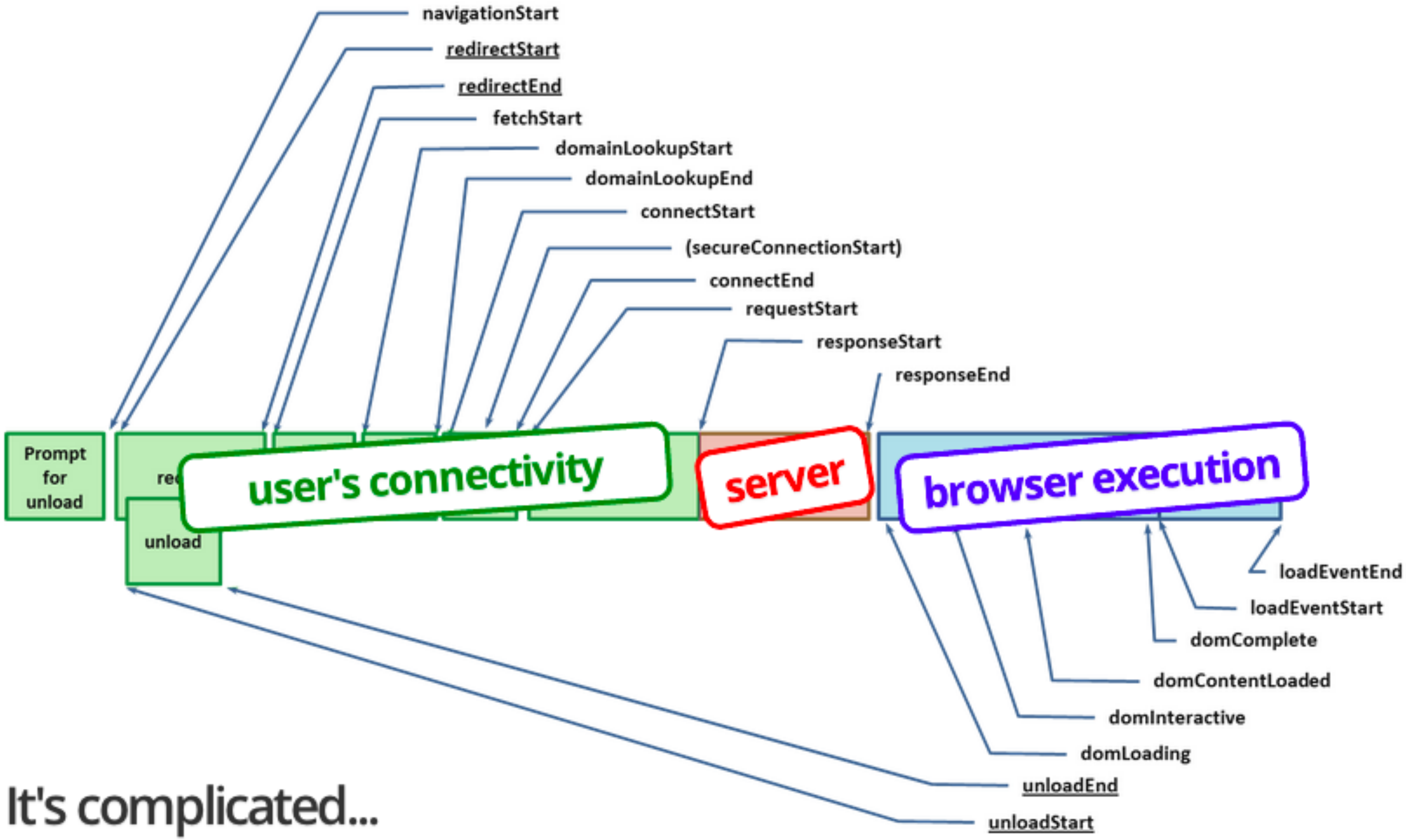
Navigation Timing (W3C)



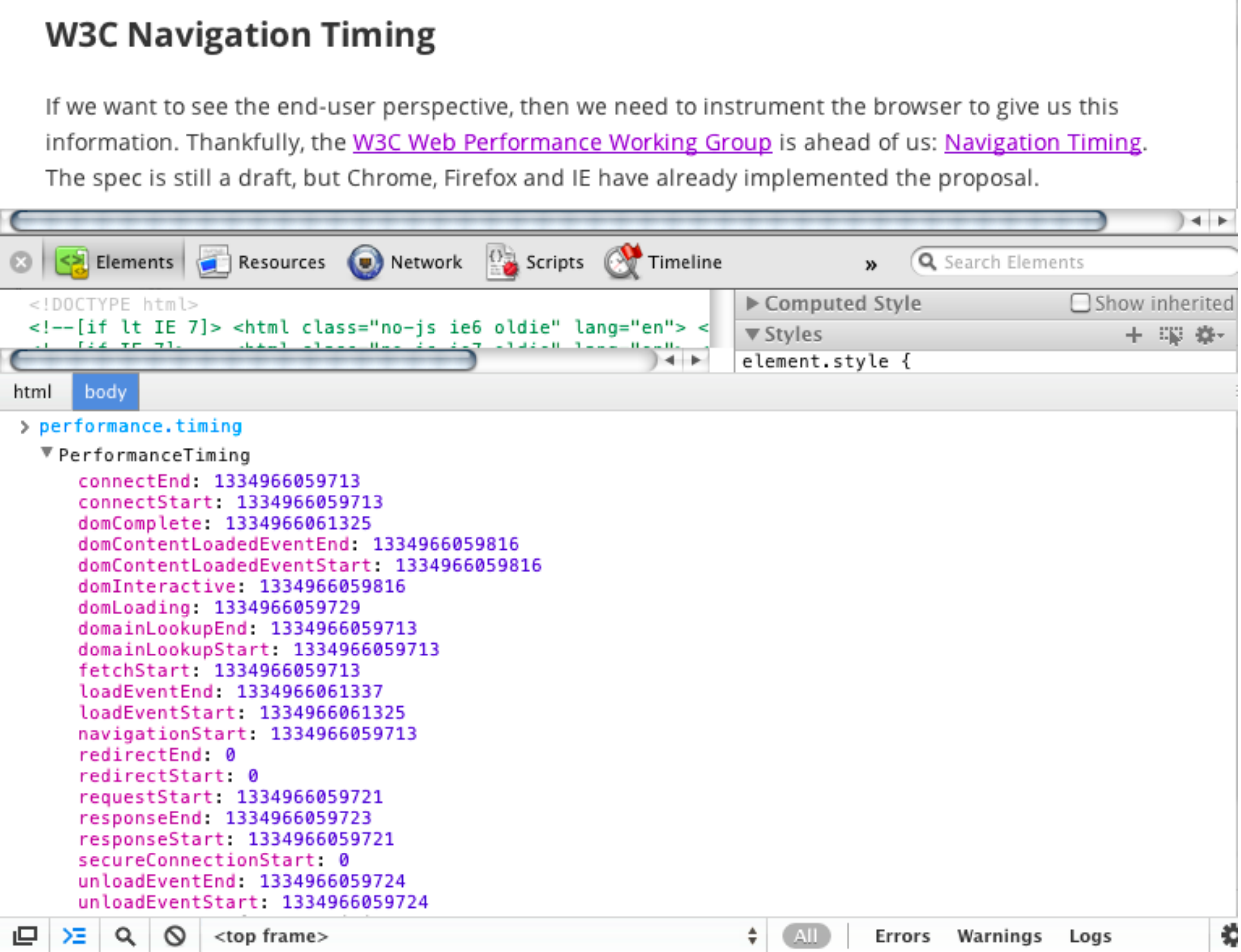
[Navigation Timing spec](#)

@igrigorik

Navigation Timing (W3C)



It's complicated...



W3C Navigation Timing

If we want to see the end-user perspective, then we need to instrument the browser to give us this information. Thankfully, the W3C Web Performance Working Group is ahead of us: Navigation Timing. The spec is still a draft, but Chrome, Firefox and IE have already implemented the proposal.

```
<!DOCTYPE html>
<!--[if lt IE 7]> <html class="no-js ie6 oldie" lang="en"> <
```

```
element.style {
```

```
> performance.timing
  ▼ PerformanceTiming
      connectEnd: 1334966059713
      connectStart: 1334966059713
      domComplete: 1334966061325
      domContentLoadedEventEnd: 1334966059816
      domContentLoadedEventStart: 1334966059816
      domInteractive: 1334966059816
      domLoading: 1334966059729
      domainLookupEnd: 1334966059713
      domainLookupStart: 1334966059713
      fetchStart: 1334966059713
      loadEventEnd: 1334966061337
      loadEventStart: 1334966061325
      navigationStart: 1334966059713
      redirectEnd: 0
      redirectStart: 0
      requestStart: 1334966059721
      responseEnd: 1334966059723
      responseStart: 1334966059721
      secureConnectionStart: 0
      unloadEventEnd: 1334966059724
      unloadEventStart: 1334966059724
```

Available in...

- IE 9+
- Firefox 7+
- Chrome 6+
- Android 4.0+

Real User Measurement (RUM) with Google Analytics

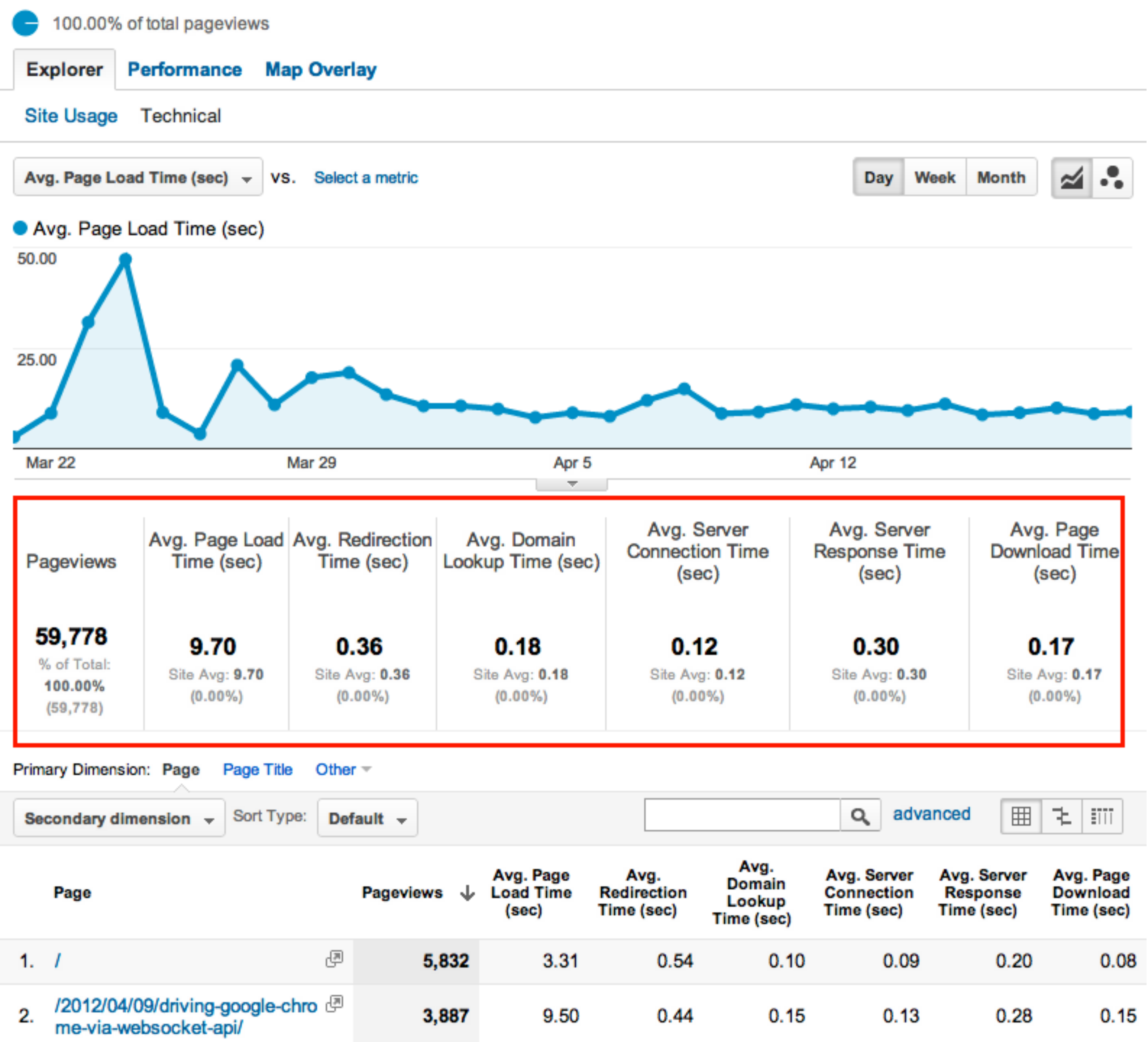
```
<script>
  _gaq.push(['_setAccount','UA-XXXX-X']);
  _gaq.push(['_setSiteSpeedSampleRate', 100]); // #protip
  _gaq.push(['_trackPageview']);
</script>
```

Google Analytics > Content > Site Speed

- Automagically collects this data for you - defaults to 1% sampling rate
- Maximum sample is 10k visits/day
- You can set custom sampling rate

You have all the power of Google Analytics! Segments, conversion metrics, ...

setSiteSpeedSampleRate docs



Avg. Page Load Time (sec) vs. Select a metric

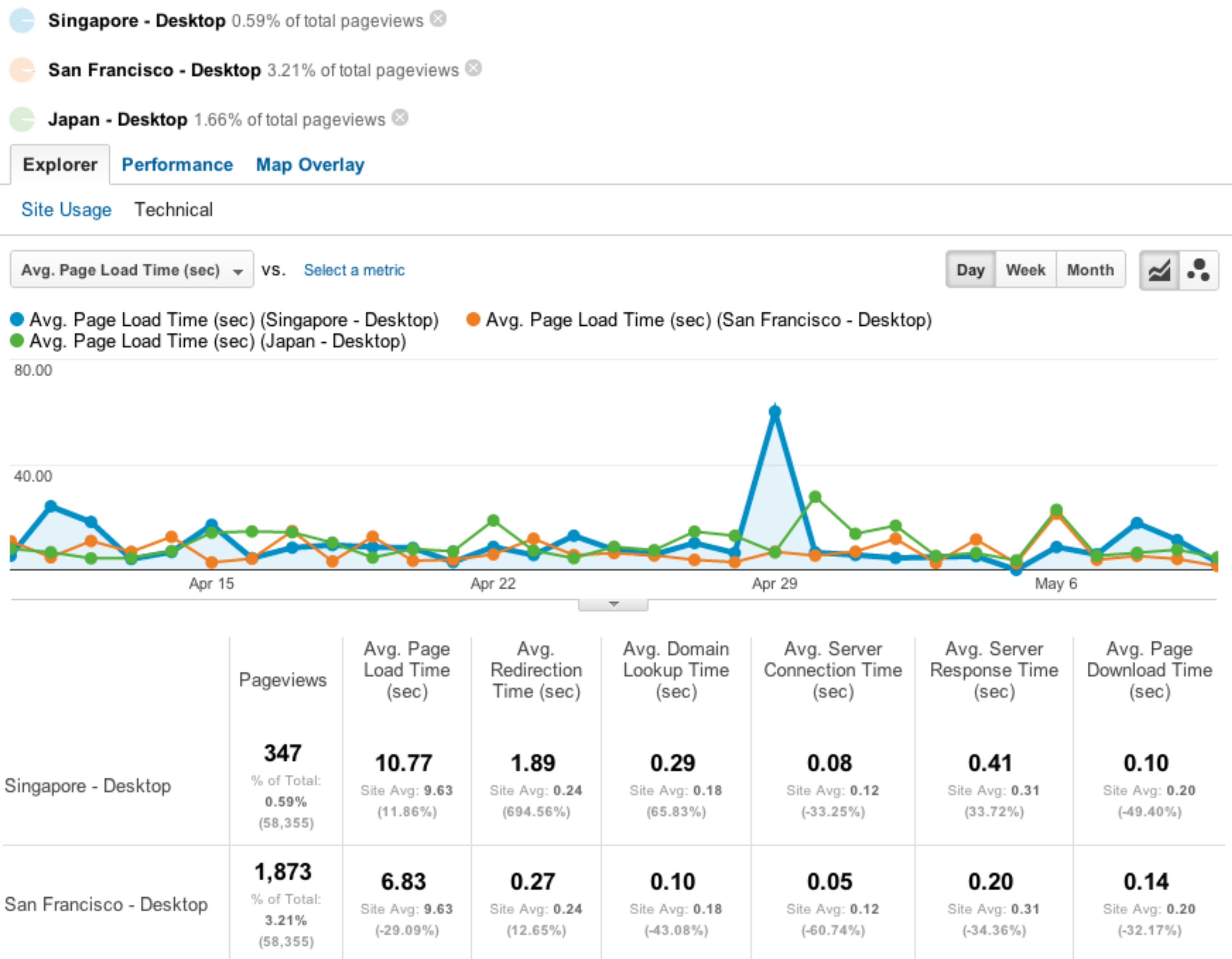
Pageviews	Avg. Page Load Time (sec)	Avg. Redirection Time (sec)	Avg. Domain Lookup Time (sec)	Avg. Server Connection Time (sec)	Avg. Server Response Time (sec)	Avg. Page Download Time (sec)
59,778 % of Total: 100.00% (59,778)	9.70 Site Avg: 9.70 (0.00%)	0.36 Site Avg: 0.36 (0.00%)	0.18 Site Avg: 0.18 (0.00%)	0.12 Site Avg: 0.12 (0.00%)	0.30 Site Avg: 0.30 (0.00%)	0.17 Site Avg: 0.17 (0.00%)

Primary Dimension: Page Page Title Other

Secondary dimension Sort Type: Default

	Page	Pageviews ↓	Avg. Page Load Time (sec)	Avg. Redirection Time (sec)	Avg. Domain Lookup Time (sec)	Avg. Server Connection Time (sec)	Avg. Server Response Time (sec)	Avg. Page Download Time (sec)
1.	/	5,832	3.31	0.54	0.10	0.09	0.20	0.08
2.	/2012/04/09/driving-google-chrome-via-websocket-api/	3,887	9.50	0.44	0.15	0.13	0.28	0.15

Performance data from real users, on real networks

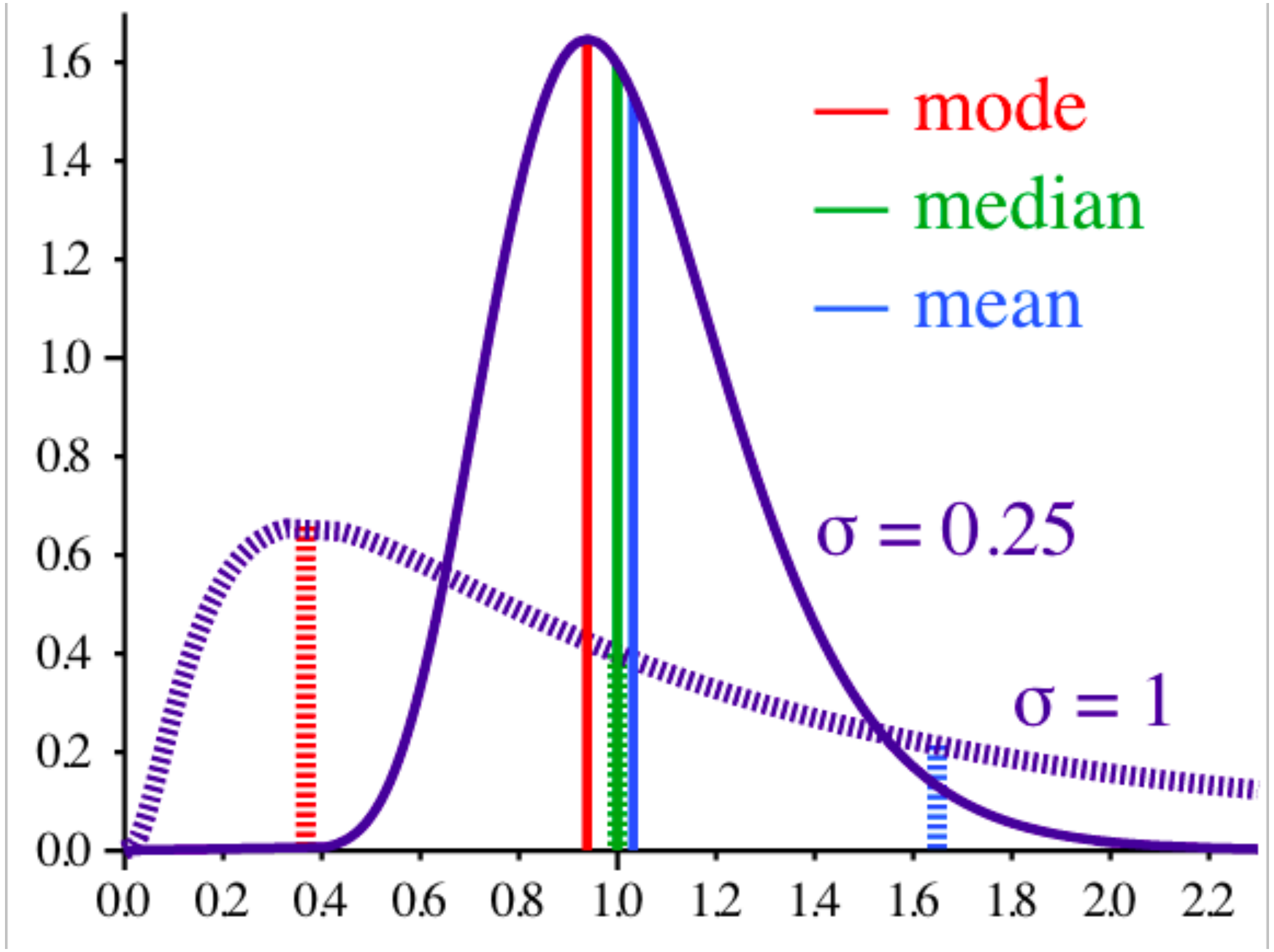


Singapore - Desktop 0.59% of total pageviews
San Francisco - Desktop 3.21% of total pageviews
Japan - Desktop 1.66% of total pageviews
Explorer
Performance
Map Overlay
Site Usage
Technical
Avg. Page Load Time (sec)
vs.
Select a metric
Day
Week
Month
● Avg. Page Load Time (sec) (Singapore - Desktop)
● Avg. Page Load Time (sec) (San Francisco - Desktop)
● Avg. Page Load Time (sec) (Japan - Desktop)
80.00
40.00
Apr 15
Apr 22
Apr 29
May 6

	Pageviews	Avg. Page Load Time (sec)	Avg. Redirection Time (sec)	Avg. Domain Lookup Time (sec)	Avg. Server Connection Time (sec)	Avg. Server Response Time (sec)	Avg. Page Download Time (sec)
Singapore - Desktop	**347** % of Total: **0.59%** (58,355)	**10.77** Site Avg: **9.63** (11.86%)	**1.89** Site Avg: **0.24** (694.56%)	**0.29** Site Avg: **0.18** (65.83%)	**0.08** Site Avg: **0.12** (-33.25%)	**0.41** Site Avg: **0.31** (33.72%)	**0.10** Site Avg: **0.20** (-49.40%)
San Francisco - Desktop	**1,873** % of Total: **3.21%** (58,355)	**6.83** Site Avg: **9.63** (-29.09%)	**0.27** Site Avg: **0.24** (12.65%)	**0.10** Site Avg: **0.18** (-43.08%)	**0.05** Site Avg: **0.12** (-60.74%)	**0.20** Site Avg: **0.31** (-34.36%)	**0.14** Site Avg: **0.20** (-32.17%)

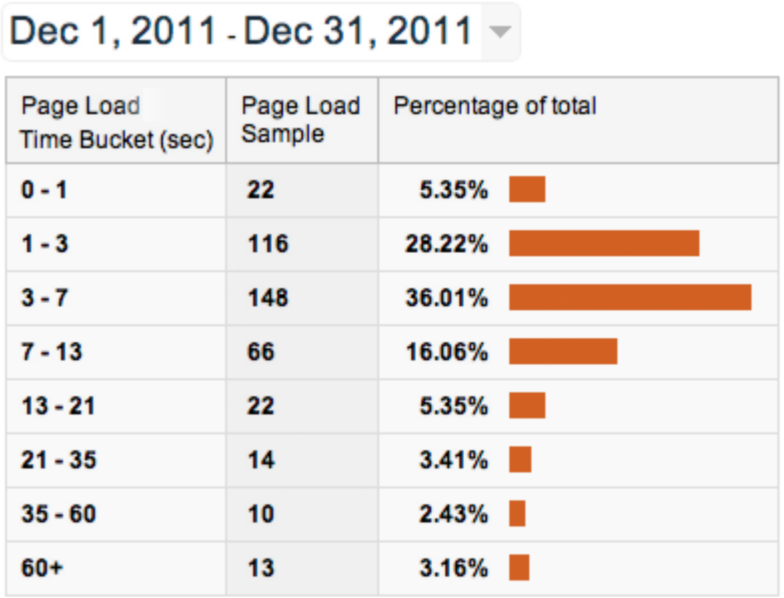
Full power of GA to segment, filter, compare, ...

But don't trust the averages...



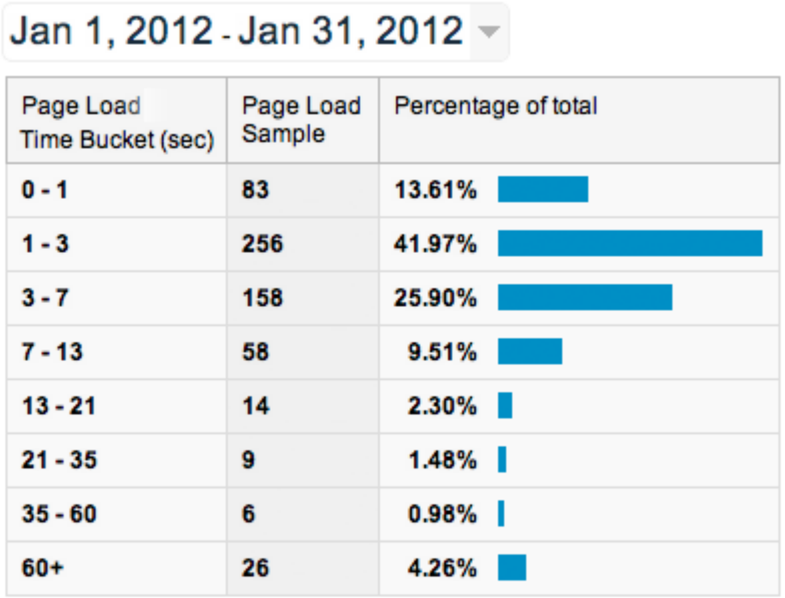
*Head into the **Technical reports** to see the histograms and distributions!*

Case study: igvita.com page load times



Dec 1, 2011 - Dec 31, 2011

Page Load Time Bucket (sec)	Page Load Sample	Percentage of total
0 - 1	22	5.35%
1 - 3	116	28.22%
3 - 7	148	36.01%
7 - 13	66	16.06%
13 - 21	22	5.35%
21 - 35	14	3.41%
35 - 60	10	2.43%
60+	13	3.16%



Jan 1, 2012 - Jan 31, 2012

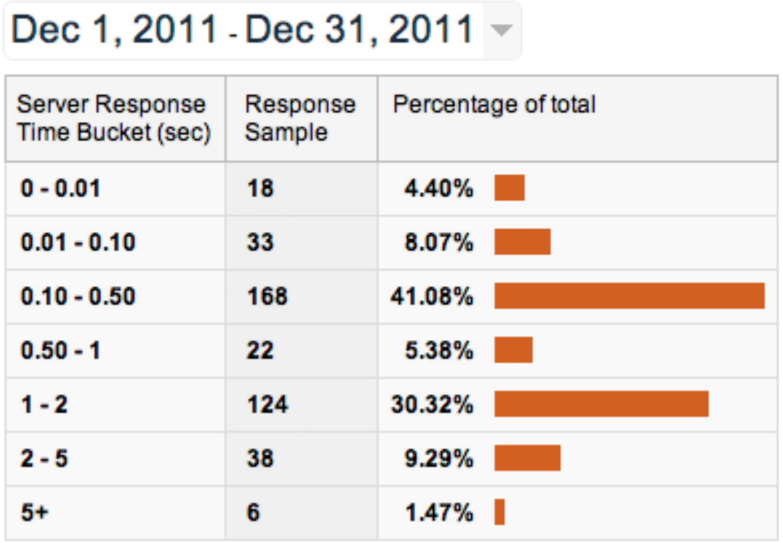
Page Load Time Bucket (sec)	Page Load Sample	Percentage of total
0 - 1	83	13.61%
1 - 3	256	41.97%
3 - 7	158	25.90%
7 - 13	58	9.51%
13 - 21	14	2.30%
21 - 35	9	1.48%
35 - 60	6	0.98%
60+	26	4.26%

Content > Site Speed > Page Timings > Performance

Migrated site to new host, server stack, web layout, and using static generation. Result: noticeable shift in the user page load time distribution.

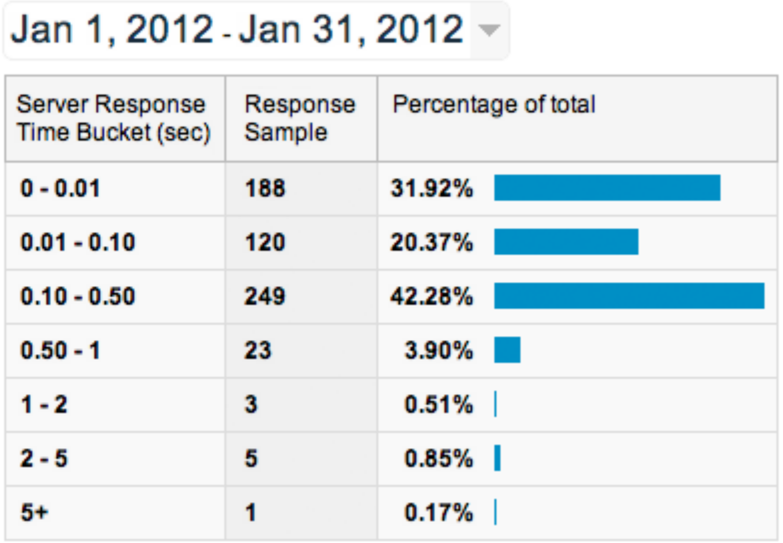
Measuring Site Speed with Navigation Timing

Case study: igvita.com server response times



Dec 1, 2011 - Dec 31, 2011

Server Response Time Bucket (sec)	Response Sample	Percentage of total
0 - 0.01	18	4.40%
0.01 - 0.10	33	8.07%
0.10 - 0.50	168	41.08%
0.50 - 1	22	5.38%
1 - 2	124	30.32%
2 - 5	38	9.29%
5+	6	1.47%



Jan 1, 2012 - Jan 31, 2012

Server Response Time Bucket (sec)	Response Sample	Percentage of total
0 - 0.01	188	31.92%
0.01 - 0.10	120	20.37%
0.10 - 0.50	249	42.28%
0.50 - 1	23	3.90%
1 - 2	3	0.51%
2 - 5	5	0.85%
5+	1	0.17%

Content > Site Speed > Page Timings > Performance

Bimodal response time distribution?
Theory: user cache *vs.* database cache *vs.* full recompute

Measuring Site Speed with Navigation Timing

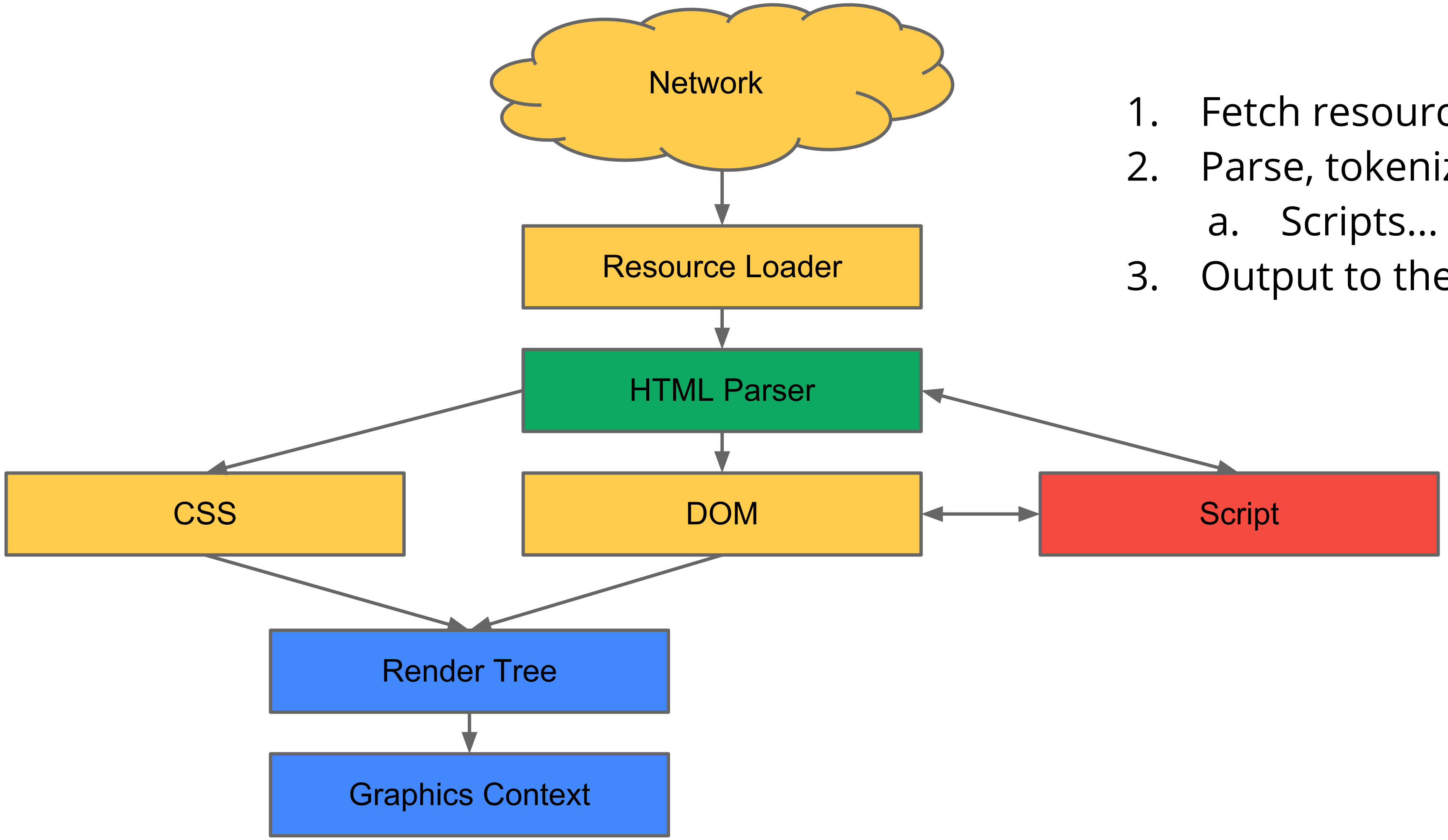
1. *Measure user perceived latency*
2. *Leverage Navigation Timing data*
3. *Use GA's advanced segments (or similar solution)*
4. *Setup {daily, weekly, ...} reports*

Measure, analyze, optimize, repeat...

How do we render the page?

we're getting bytes off the wire... and then what?

Life of a web-page in WebKit

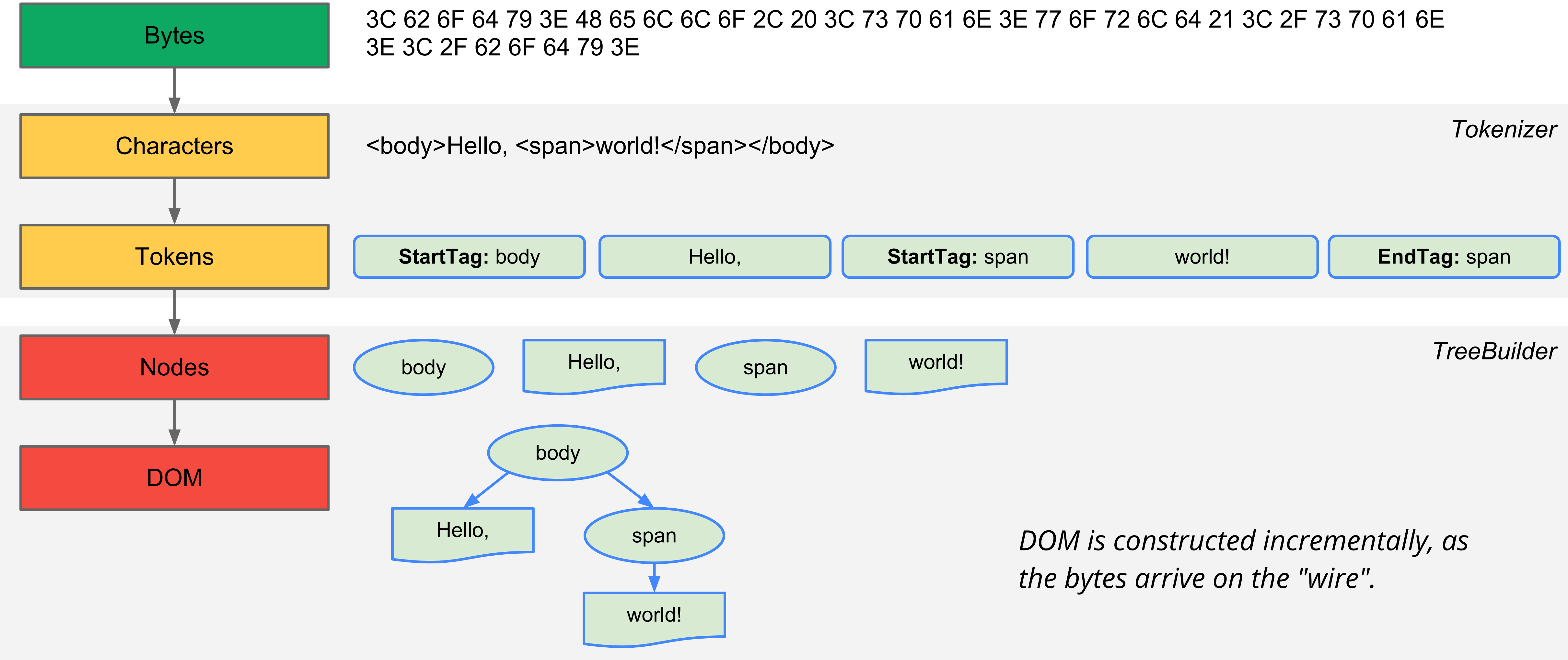


1. Fetch resources from the network
2. Parse, tokenize, construct the OM
   a. Scripts...
3. Output to the screen

How WebKit works - Adam Barth

@igrigorik

The HTML(5) parser at work...



Bytes

3C 62 6F 64 79 3E 48 65 6C 6C 6F 2C 20 3C 73 70 61 6E 3E 77 6F 72 6C 64 21 3C 2F 73 70 61 6E
3E 3C 2F 62 6F 64 79 3E

Tokenizer

Characters

<body>Hello, <span>world!</span></body>

Tokens

StartTag: body | Hello, | **StartTag:** span | world! | **EndTag:** span

TreeBuilder

Nodes

body | Hello, | span | world!

DOM

body
- Hello,
- span
  - world!

DOM is constructed incrementally, as the bytes arrive on the "wire".

How WebKit works - Adam Barth

@igrigorik

The HTML(5) parser at work…

```
<!doctype html>
<meta charset=utf-8>
<title>Awesome HTML5 page</title>

<script src=application.js></script>
<link href=styles.css rel=stylesheet />

<p>I'm awesome.
```

HTMLDocumentParser begins parsing the received data …

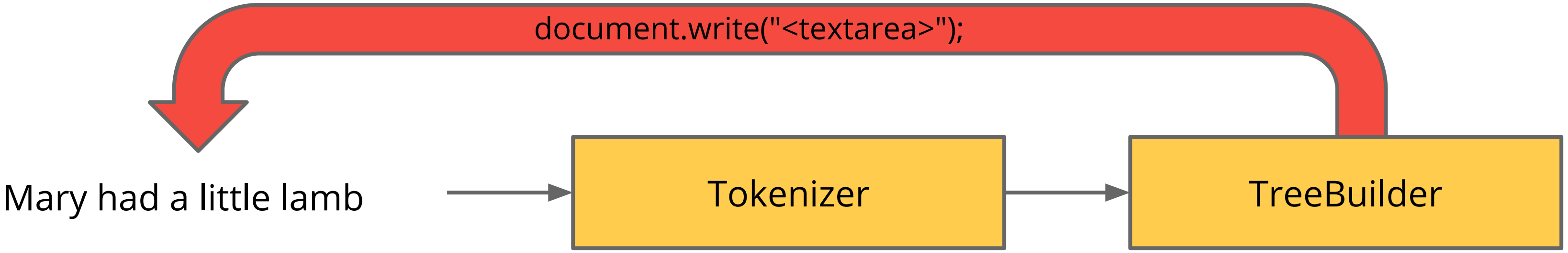
```
HTML
  - HEAD
    - META charset="utf-8"
    - TITLE
      #text: Awesome HTML5 page
    - SCRIPT src="application.js"
      ** stop **
```

Stop. Dispatch request for application.js. Wait…

<script> could doc.write, stop the world!

script "async" and "defer" are your escape clauses

Sync scripts block the parser...



Script execution can change the input stream. Hence we **must wait**.

Sync scripts block the parser...

Sync script **will block** the rendering of your page:

```
<script type="text/javascript" src="https://apis.google.com/js/plusone.js"></script>
```

✖

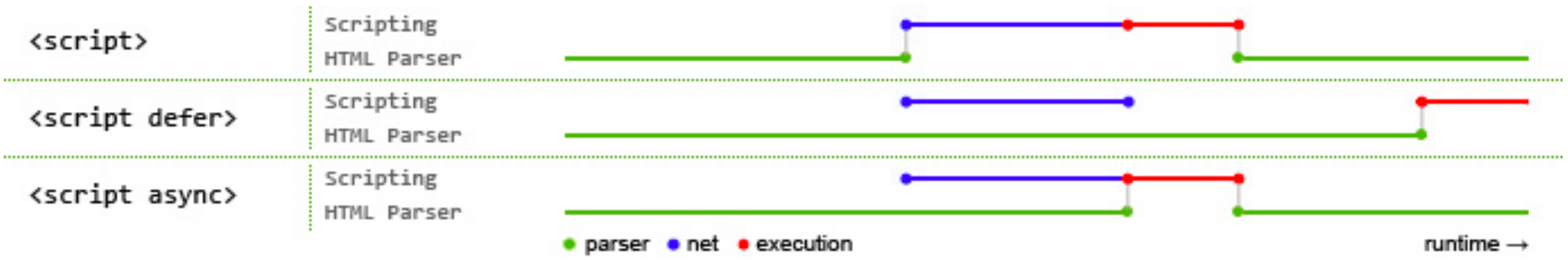
Async script **will not block** the rendering of your page:

```
<script type="text/javascript">
  (function() {
    var po = document.createElement('script'); po.type = 'text/javascript';
    po.async = true; po.src = 'https://apis.google.com/js/plusone.js';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(po, s);
  })();
</script>
```

✚

async *vs.* defer

```
<script src="file-a.js"></script>
<script src="file-b.js" defer></script>
<script src="file-c.js" async></script>
```



- **regular** - wait for request, execute, proceed
- **defer** - download in background, execute in order before DomContentLoaded
- **async** - download in background, execute when ready

async and defer explained

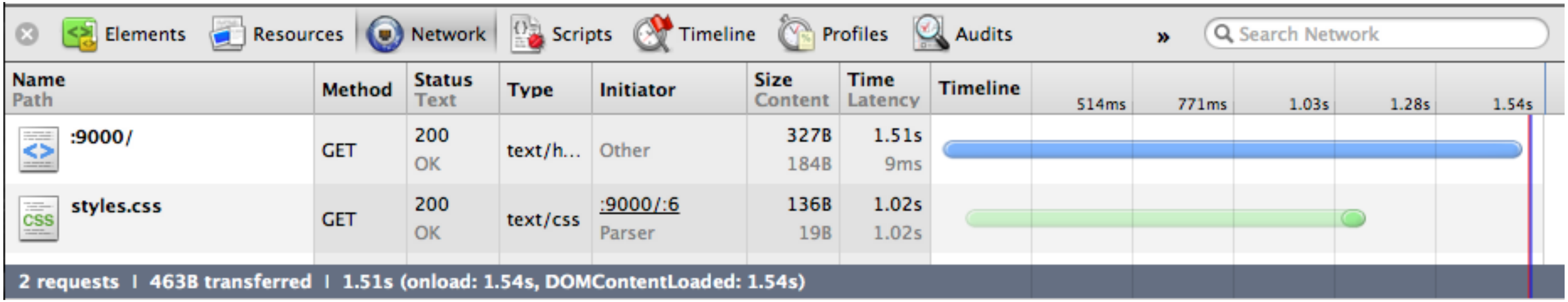
Browser tries to help.. Preload Scanner to the rescue!

```
if (isWaitingForScripts()) {
    ASSERT(m_tokenizer->state() == HTMLTokenizerState::DataState);
    if (!m_preloadScanner) {
        m_preloadScanner = adoptPtr(new HTMLPreloadScanner(document()));
        m_preloadScanner->appendToEnd(m_input.current());
    }
    m_preloadScanner->scan();
}
```

HTMLPreloadScanner tokenizes ahead, looking for blocking resources...

```
if (m_tagName    != imgTag
    && m_tagName != inputTag
    && m_tagName != linkTag
    && m_tagName != scriptTag
    && m_tagName != baseTag)
    return;
```

Flush early, flush often...



Name	Method	Status	Type	Initiator	Size	Time	Timeline
Path		Text			Content	Latency	
:9000/	GET	200 OK	text/h...	Other	327B 184B	1.51s 9ms	
styles.css	GET	200 OK	text/css	:9000/:6 Parser	136B 19B	1.02s 1.02s	

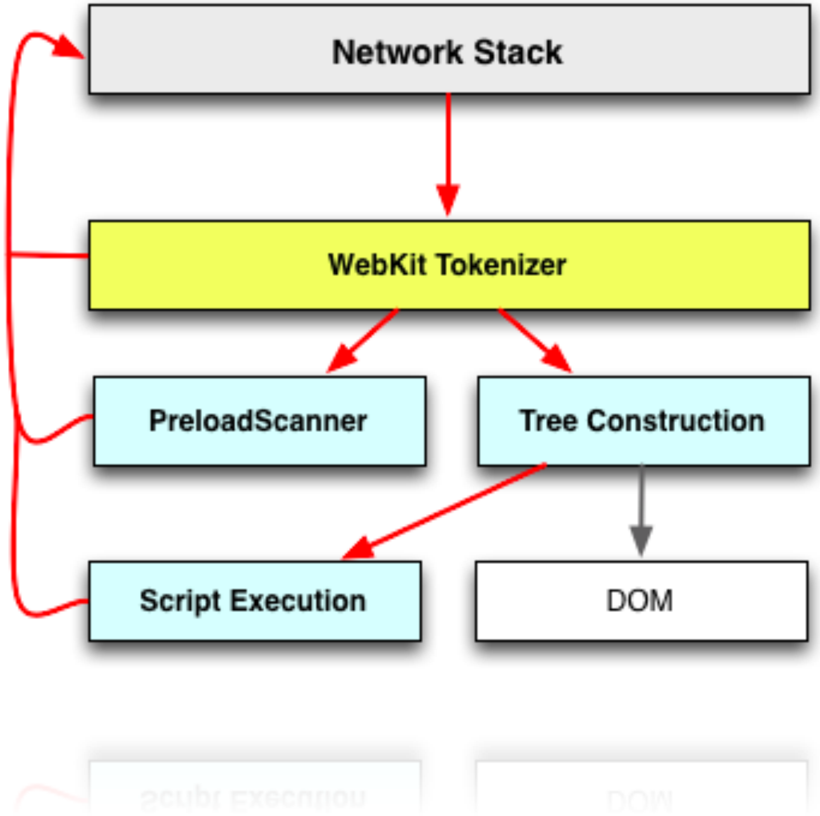
2 requests | 463B transferred | 1.51s (onload: 1.54s, DOMContentLoaded: 1.54s)

Early flush example: https://gist.github.com/3058839

- Time to first byte (**TTFB**) matters when you can deliver useful data in those first bytes!
- Example: flush the header of your page before the rest of your body to kick off resource fetch!

- Network stack can run **DNS prefetch** & **TCP-preconnect**
- **PreloadScanner** can fetch resources while parser is blocked

Let the browser help you...



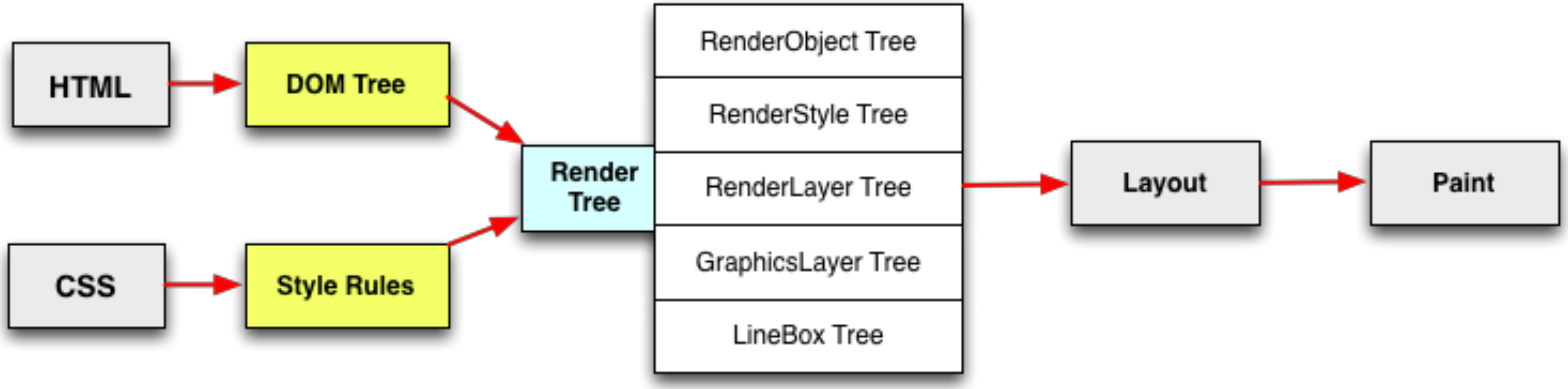
- Flush early, flush often, flush smart
- Time to first packet matters when...
- Content of first packet can tip-off the parser

- Try not to hide resources from the parser!
- CSSPreloadScanner scans for @import's only

Let's build a Render tree

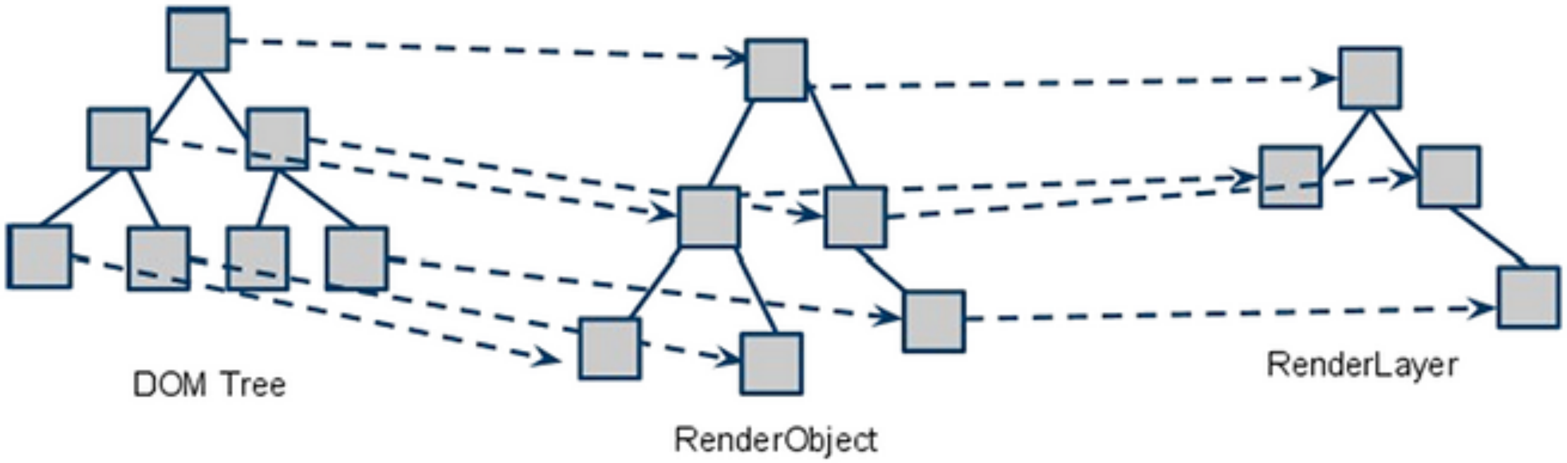
Or, maybe an entire forest?

DOM + CSSOM > Render Tree(s)



- Some trees share objects
- Independently constructed, not 1:1 match
- Lazy evaluation - defer to just before we need to render!

DOM + CSSOM > Render Tree(s)



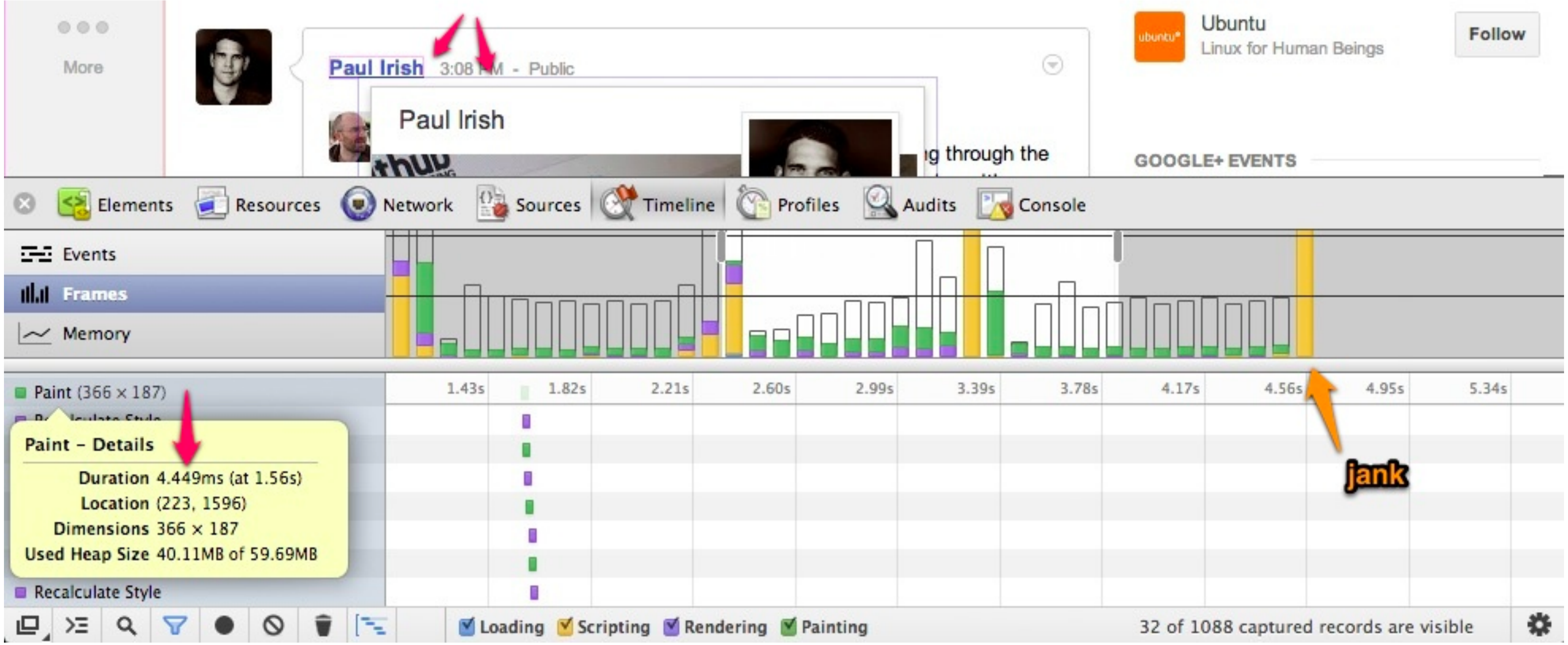
RenderObject Tree	StyleObject Tree	RenderLayer Tree
owned by DOM tree	computed styles for all renderers	"helper" class for rendering
rendered content only	owned by RenderObject tree	used for <video>, <canvas>, ...
responsible for layout & paint	RenderObjects share RenderStyles	Some RenderLayers have GPU layers
answers DOM API measurement requests	RenderStyles share data members	...

Querying layout (ex, **offset{Width,Height}**), forces a full layout flush!

"60 FPS? That's for games and stuff, right?"

Wrong. 60 FPS applies to web pages as well!

What are we painting? How much?



Paint – Details

Duration 4.449ms (at 1.56s)

Location (223, 1596)

Dimensions 366 × 187

Used Heap Size 40.11MB of 59.69MB

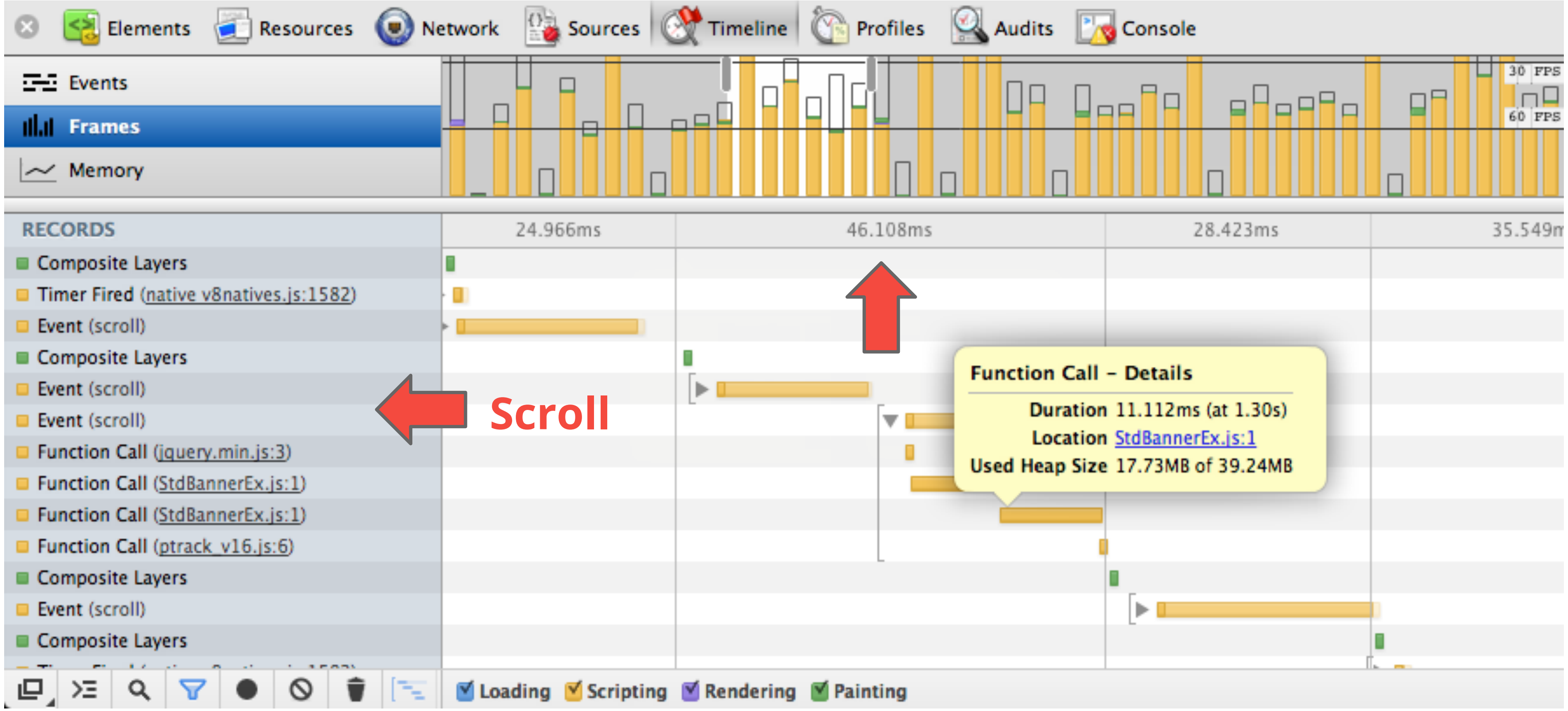
☑ Loading ☑ Scripting ☑ Rendering ☑ Painting

32 of 1088 captured records are visible

- Enable "show paint rectangles" to see painted areas
- Check timeline to see time taken, memory usage, dimensions, and more...
- **Minimize the paint areas whenever possible**

Wait, DevTools could do THAT?

How much time did each frame take?



RECORDS

- Composite Layers
- Timer Fired (native v8natives.js:1582)
- Event (scroll)
- Composite Layers
- Event (scroll)
- Event (scroll)
- Function Call (jquery.min.js:3)
- Function Call (StdBannerEx.js:1)
- Function Call (StdBannerEx.js:1)
- Function Call (ptrack_v16.js:6)
- Composite Layers
- Event (scroll)
- Composite Layers

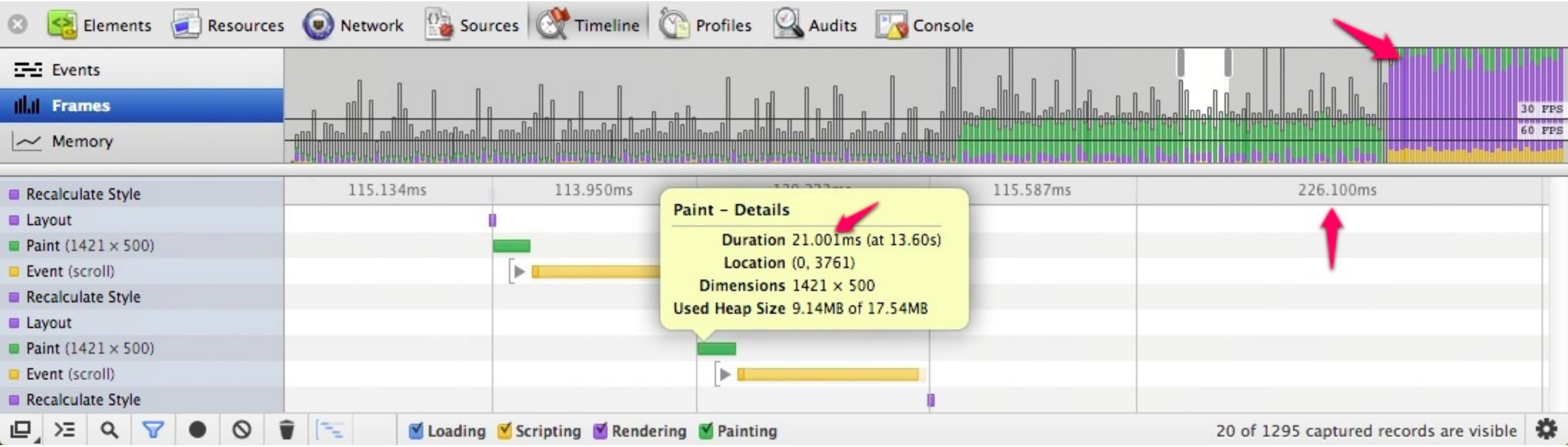
Function Call – Details

Duration 11.112ms (at 1.30s)
Location StdBannerEx.js:1
Used Heap Size 17.73MB of 39.24MB

- **60 FPS** affords you a **16.6 ms budget per frame**
- StdBannerEx.js is executing **20 ms+ of JavaScript** on every scroll event ... *<facepalm />*
- **It's better to be at consistent** than jump between variable frame-rates

Google I/O 2012 - Jank Busters: Building Performant Web Apps

How much time did each frame take?



- Recalculate Style
- Layout
- Paint (1421 × 500)
- Event (scroll)
- Recalculate Style
- Layout
- Paint (1421 × 500)
- Event (scroll)
- Recalculate Style

Paint – Details

Duration 21.001ms (at 13.60s)
Location (0, 3761)
Dimensions 1421 × 500
Used Heap Size 9.14MB of 17.54MB

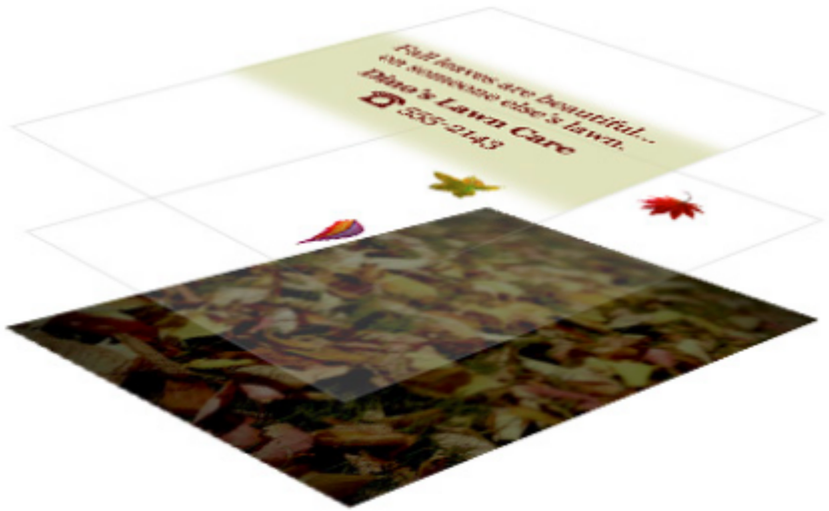
☑ Loading ☑ Scripting ☑ Rendering ☑ Painting

20 of 1295 captured records are visible

Jank demo (open Timeline, hit record, and err.. enjoy)

- **CSS effects** can cause slow(er) paints
- **Style recalculations** can cause slow(er) paints
- **Excessive Javascript** can cause slow(er) paints

Hardware Acceleration 101



- A RenderLayer can have a GPU backing store
- Certain elements are GPU backed automatically (canvas, video, CSS3 animations, ...)
- Forcing a GPU layer: *-webkit-transform:translateZ(0)*
- GPU is **really fast** at **compositing**, **matrix operations** and **alpha blends**

Hardware Acceleration 101

1. The **object is painted** to a buffer (texture)
2. **Texture is uploaded** to GPU
3. Send commands to GPU: **apply op X to texture Y**

- Minimize CPU-GPU interactions
- Texture **uploads are not free**
- No upload: position, size, opacity
- Texture upload: everything else

CSS3 Animations are as close to "free lunch" as you can get **

*** Assuming no texture reuploads and animation runs entirely on GPU...*

CSS3 Animations with no Javascript!

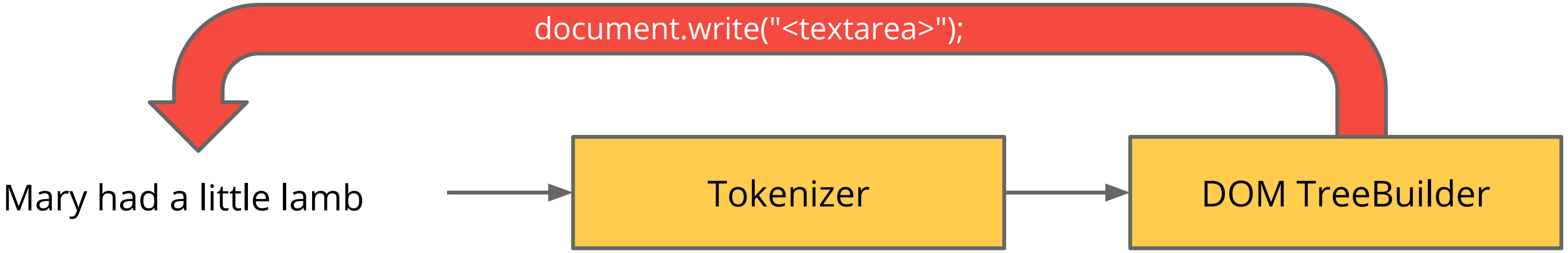
```
<style>
  .spin:hover {
    -webkit-animation: spin 2s infinite linear;
  }

  @-webkit-keyframes spin {
    0% { -webkit-transform: rotate(0deg);}
    100% { -webkit-transform: rotate(360deg);}
  }
</style>

<div class="spin" style="background-image: url(images/chrome-logo.png);"></div>
```

- Look ma, no JavaScript!
- Performance: YMMV, but improving rapidly

DOM, CSSOM & Javascript sitting in a tree...

There is an interesting dependency graph in here...

(1) Scripts can block the document parser...

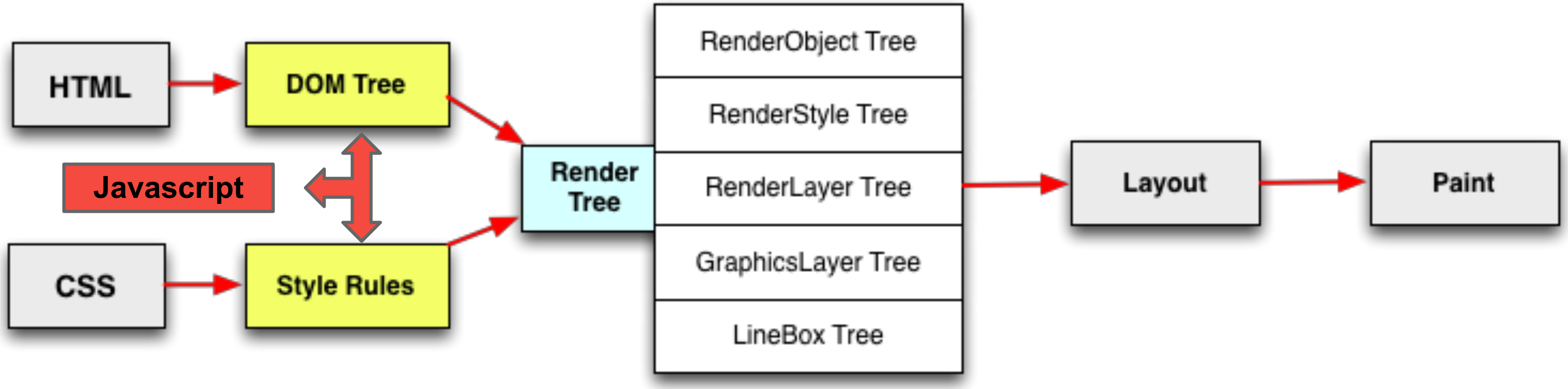


JavaScript can **block the DOM** construction.

Script execution can change the input stream. Hence we **must wait**.

@igrigorik

(2) Javascript can query CSS, which means...



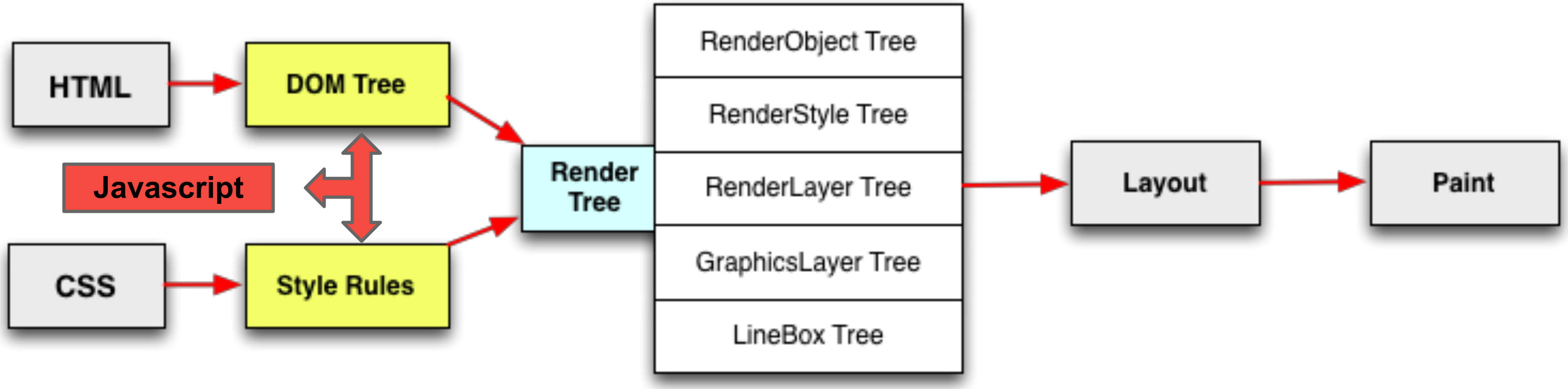
JavaScript can **block on CSS**.

DOM construction can be blocked on Javascript, which can be blocked on CSS

- ○ *ex: asking for computed style, but stylesheet is not yet ready...*

At least CSS can't query javascript.. phew!

(3) Rendering is blocked on CSS...



CSS must be fetched & parsed before Render tree can be painted.

Otherwise, the user will see "flash of unstyled content" + reflow and repaint when CSS is ready

At least CSS can't query javascript.. phew!

Putting it all together...

(1) JavaScript can **block the DOM** construction
(2) JavaScript can **block on CSS**
(3) Rendering is **blocked on CSS**...

Which means...

(1) Get CSS down to the client as fast as you can

- *Unblocks paints, removes potential JS waiting on CSS scenario*

(2) If you can, use async scripts + avoid doc.write at all costs

- *Faster DOM construction, faster DCL and paint!*

Now let's try a fabricated example...

Doesn't mean it's an easy one!

What could be simpler...

```
<html>
  <body>
    <link rel="stylesheet" href="example.css">

    <div>Hi there!</div>

    <script>
      document.write('<script src="other.js"></scr' + 'ipt>');
    </script>

    <div>Hi again!</div>

    <script src="last.js"></script>
  </body>
</html>
```

Actually, it's not simple, at all...

```
<html>
  <body>
    <link rel="stylesheet" href="example.css">

    <div>Hi there!</div>

    <script>...
```

- Parser discovers **example.css** and fetches it from the network
- Parser **continues without blocking** on fetch of example.css

- Parser reaches start of inline script block
  - **Can't execute** because it's blocked on pending stylesheet

- Render tree construction also blocked on stylesheet, so **no paint requested**
- Preload scanner looks ahead in the document, initiates fetch for last.js

Understanding and Optimizing Web Performance Metrics - Bryan McQuade

Actually, it's not simple, at all...

```
  <html>
    <body>
      <link rel="stylesheet" href="example.css">

      <div>Hi there!</div>

      <script>
        document.write('<script src="other.js"></scr' + 'ipt>');
      </script>
```

- Once **example.css** finishes loading, **render tree is constructed**
- After inline script block executes, parser is immediately blocked on **other.js**
  - *Preloader is of no help here, since other.js is scheduled via JS*

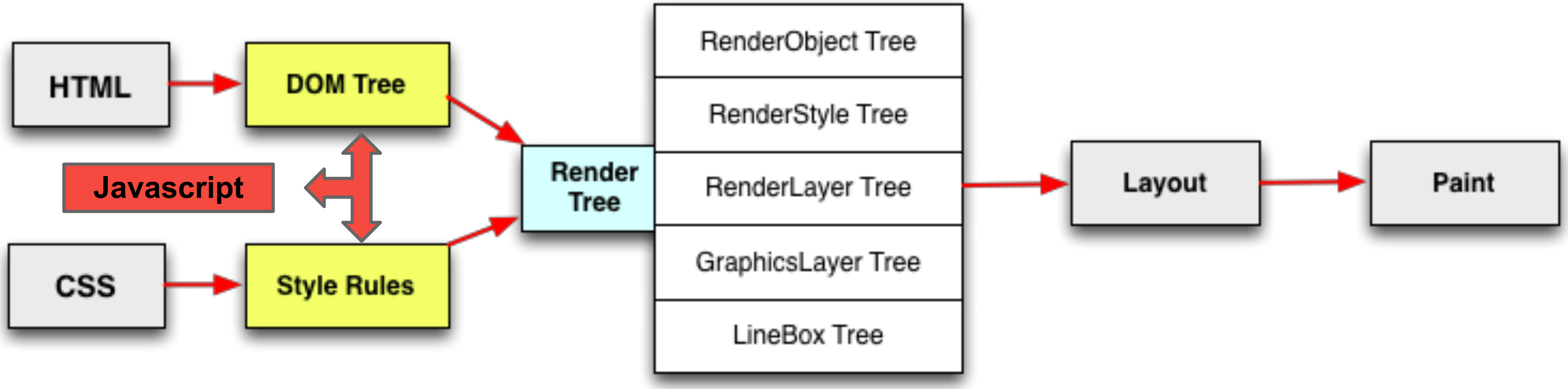
- Once parser is blocked, first paint is requested and **"Hi there!"** is painted to the screen

Actually, it's not simple, at all...

```html
<html>
  <body>
    <link rel="stylesheet" href="example.css">

    <div>Hi there!</div>

    <script>
      document.write('<script src="other.js"></scr' + 'ipt>');
    </script>

    <div>Hi again!</div>

    <script src="last.js"></script>
  </body>
</html>
```

- Parser discovers **last.js**, which, thanks to the speculative loader, is in the browser cache
  - last.js is executed immediately
- Paint is requested and "Hi again!" is painted to the screen
- **Done**

Not to repeat myself, but ...



(1) Get CSS down to the client as fast as you can

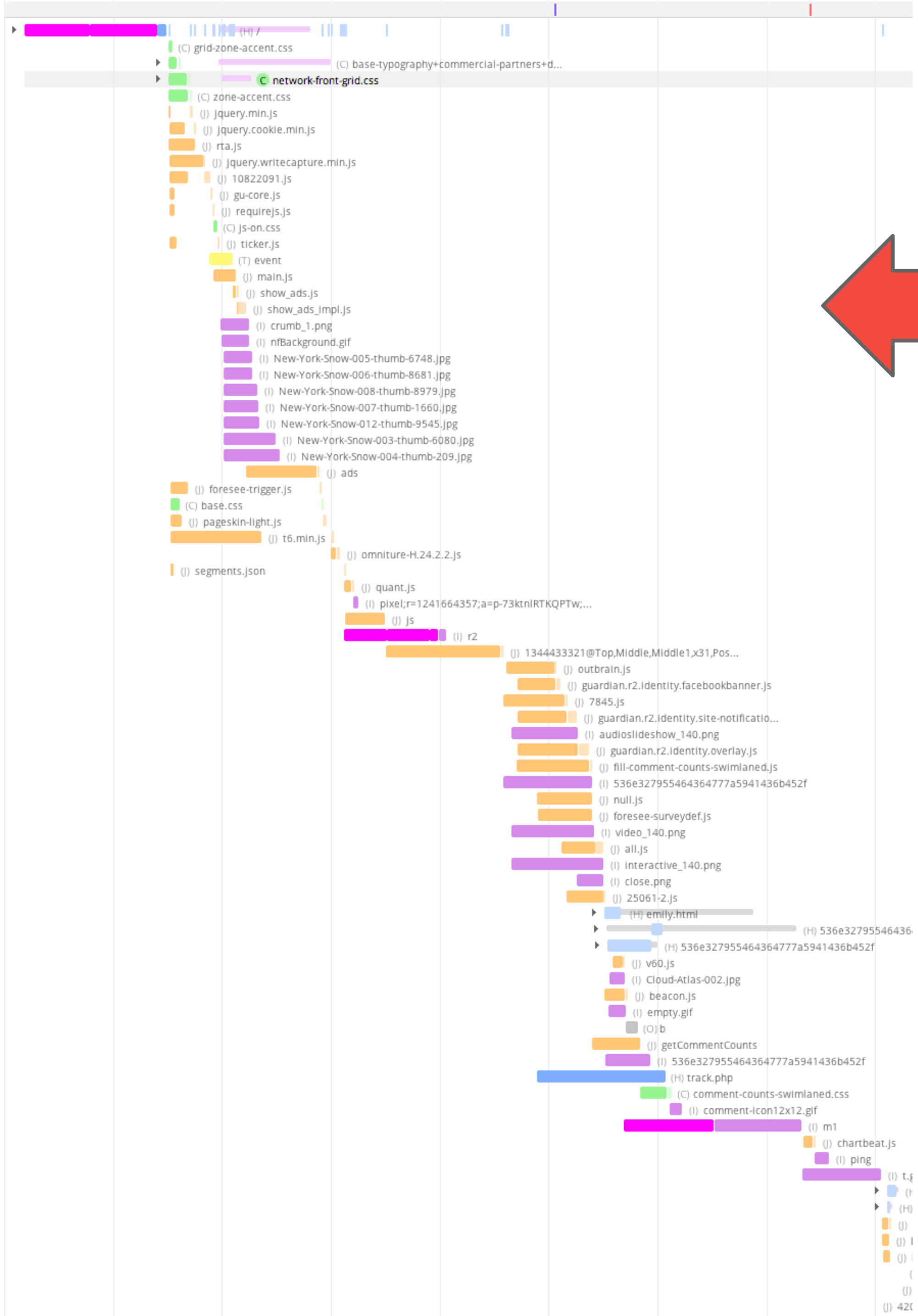
- *Unblocks paints, removes potential JS waiting on CSS scenario*

(2) If you can, use async scripts + avoid doc.write at all costs

- *Faster DOM construction, faster DCL and paint!*

OK. Let's try a real-life example...

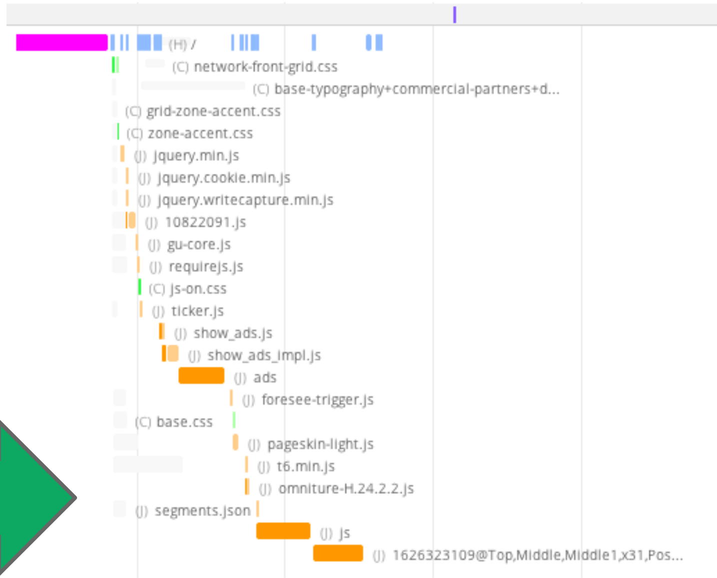
and apply what we've learned so far!


(H) /
(C) grid-zone-accent.css
(C) base-typography+commercial-partners+d...
C network-front-grid.css
(C) zone-accent.css
(J) jquery.min.js
(J) jquery.cookie.min.js
(J) rta.js
(J) jquery.writecapture.min.js
(J) 10822091.js
(J) gu-core.js
(J) requirejs.js
(C) js-on.css
(J) ticker.js
(T) event
(J) main.js
(J) show_ads.js
(J) show_ads_impl.js
(I) crumb_1.png
(I) nfBackground.gif
(I) New-York-Snow-005-thumb-6748.jpg
(I) New-York-Snow-006-thumb-8681.jpg
(I) New-York-Snow-008-thumb-8979.jpg
(I) New-York-Snow-007-thumb-1660.jpg
(I) New-York-Snow-012-thumb-9545.jpg
(I) New-York-Snow-003-thumb-6080.jpg
(I) New-York-Snow-004-thumb-209.jpg
(J) ads
(J) foresee-trigger.js
(C) base.css
(J) pageskin-light.js
(J) t6.min.js
(J) omniture-H.24.2.2.js
(J) segments.json
(J) quant.js
(I) pixel;r=1241664357;a=p-73ktnlRTKQPTw;...
(J) js
(I) r2
(J) 1344433321@Top,Middle,Middle1,x31,Pos...
(J) outbrain.js
(J) guardian.r2.identity.facebookbanner.js
(J) 7845.js
(J) guardian.r2.identity.site-notificatio...
(I) audioslideshow_140.png
(J) guardian.r2.identity.overlay.js
(J) fill-comment-counts-swimlaned.js
(I) 536e327955464364777a5941436b452f
(J) null.js
(J) foresee-surveydef.js
(I) video_140.png
(J) all.js
(I) interactive_140.png
(I) close.png
(J) 25061-2.js
(H) emily.html
(H) 536e327955464364
(H) 536e327955464364777a5941436b452f
(J) v60.js
(I) Cloud-Atlas-002.jpg
(J) beacon.js
(I) empty.gif
(O) b
(J) getCommentCounts
(I) 536e327955464364777a5941436b452f
(H) track.php
(C) comment-counts-swimlaned.css
(I) comment-icon12x12.gif
(I) m1
(J) chartbeat.js
(I) ping
(I) t.g
(J) 420

guardian.co.uk

Full Waterfall

Critical Path

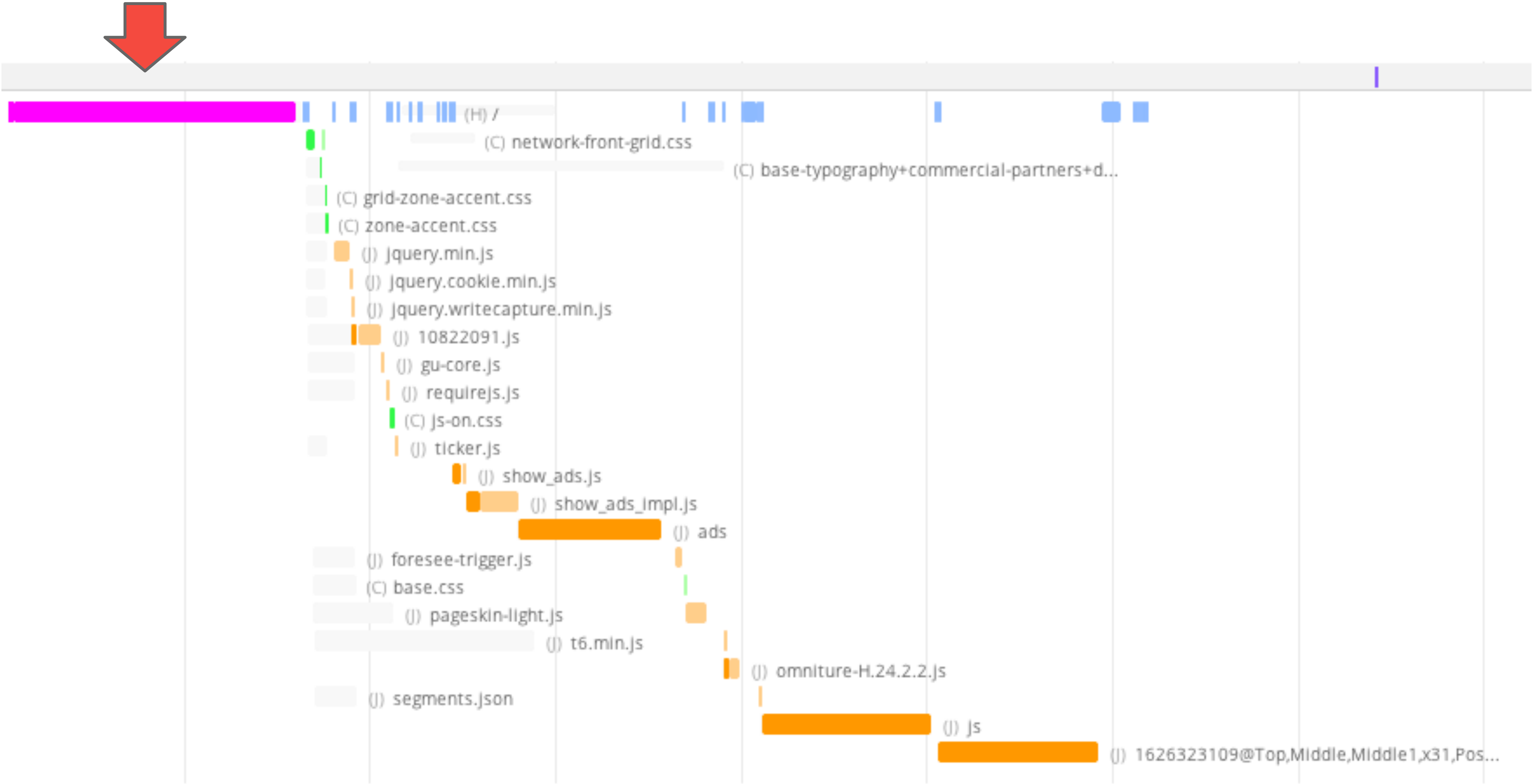


Critical Path Explorer* extracts the subtree** *of the waterfall that is in the* **"critical path"** *of the* ***document parser and the renderer.

(automation for the win!)

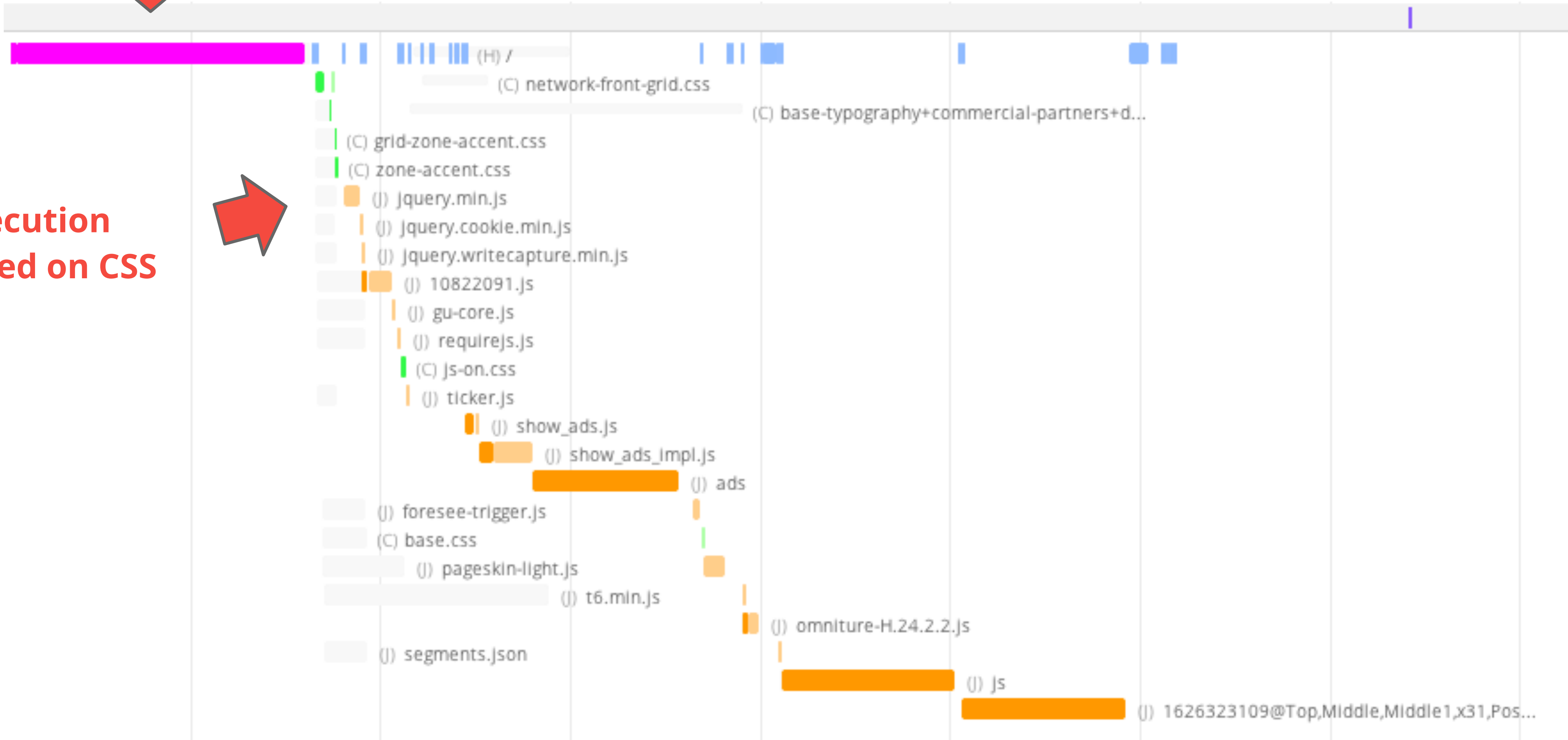
@igrigorik

300 ms redirect!



300 ms redirect!

JS execution blocked on CSS

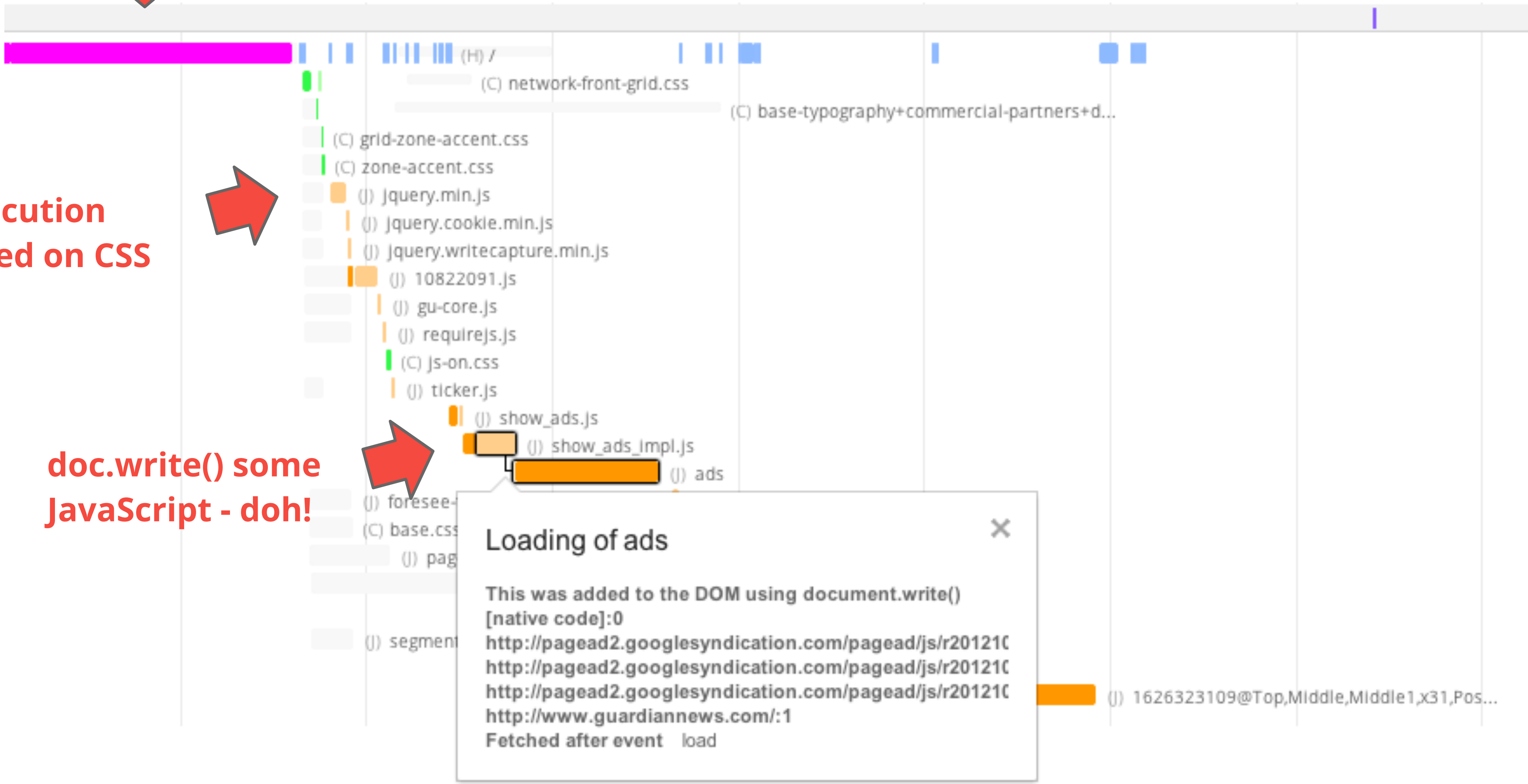


@igrigorik

300 ms redirect!

JS execution blocked on CSS

doc.write() some JavaScript - doh!



Loading of ads

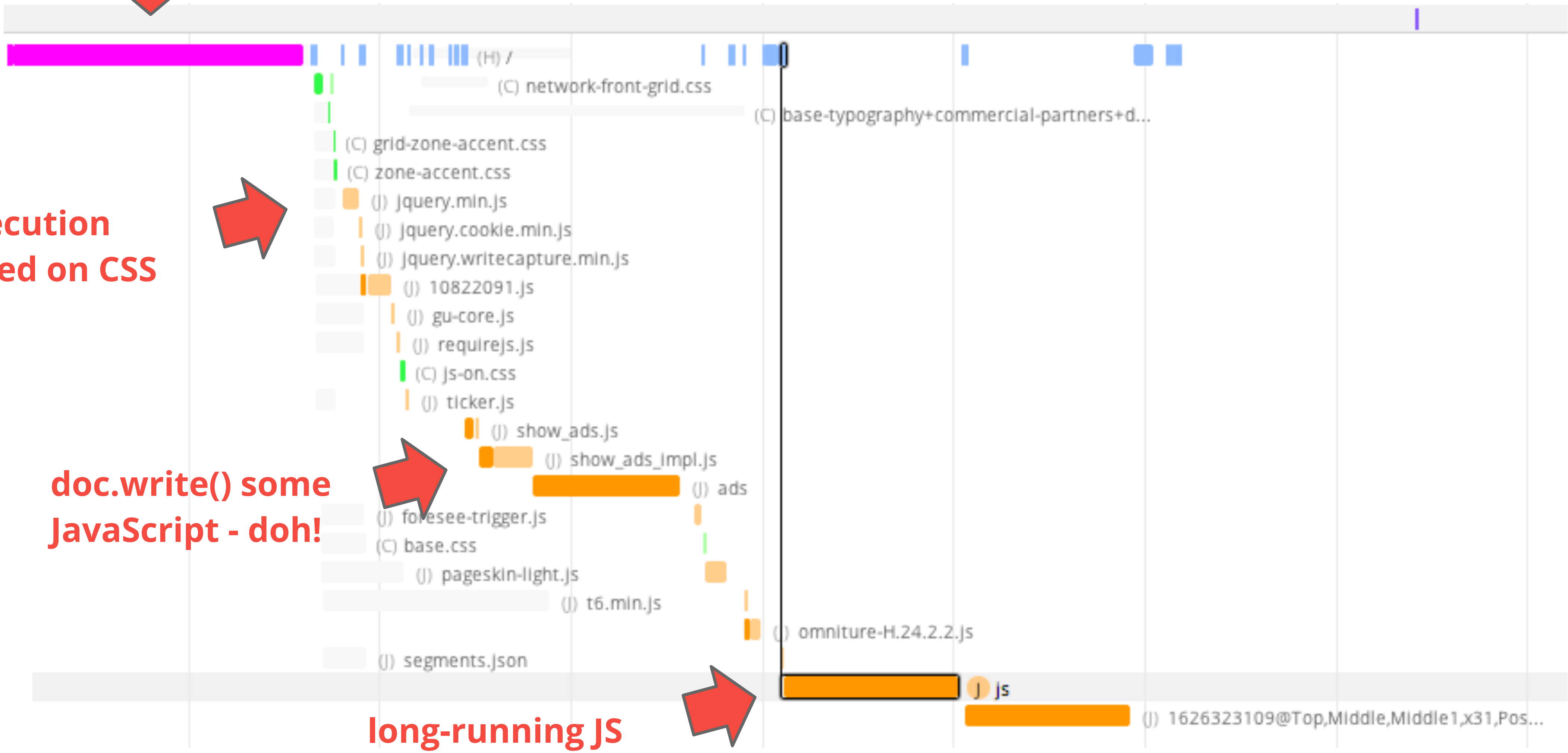
This was added to the DOM using document.write()
[native code]:0
http://pagead2.googlesyndication.com/pagead/js/r20121(
http://pagead2.googlesyndication.com/pagead/js/r20121(
http://pagead2.googlesyndication.com/pagead/js/r20121(
http://www.guardiannews.com/:1
Fetched after event load

@igrigorik

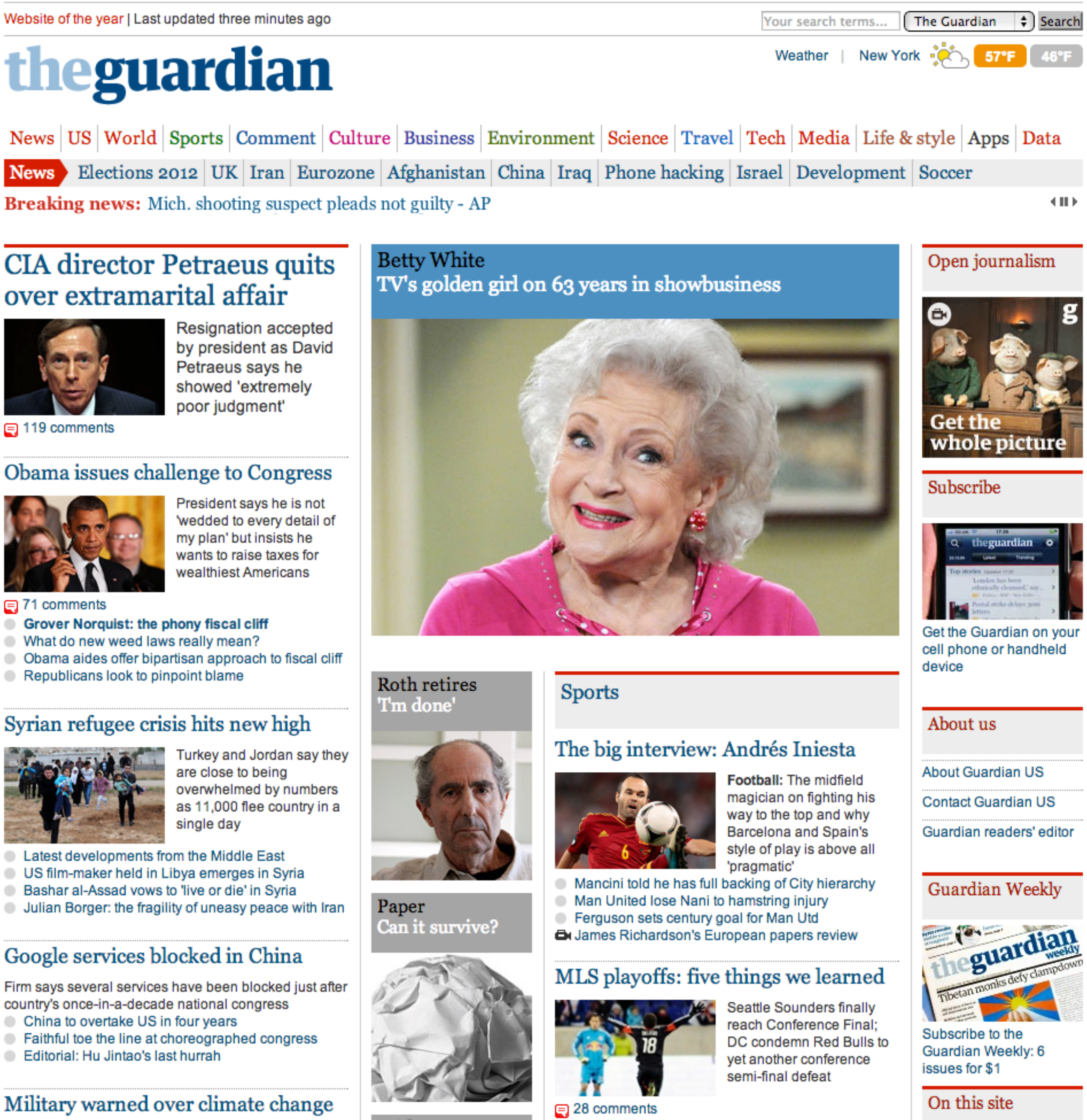
300 ms redirect!

JS execution blocked on CSS

doc.write() some JavaScript - doh!



long-running JS



Website of the year | Last updated three minutes ago

Your search terms... The Guardian Search

theguardian

Weather | New York 57°F 46°F

News | US | World | Sports | Comment | Culture | Business | Environment | Science | Travel | Tech | Media | Life & style | Apps | Data

News Elections 2012 | UK | Iran | Eurozone | Afghanistan | China | Iraq | Phone hacking | Israel | Development | Soccer

Breaking news: Mich. shooting suspect pleads not guilty - AP

CIA director Petraeus quits over extramarital affair

Resignation accepted by president as David Petraeus says he showed 'extremely poor judgment'

119 comments

Obama issues challenge to Congress

President says he is not 'wedded to every detail of my plan' but insists he wants to raise taxes for wealthiest Americans

71 comments

- **Grover Norquist: the phony fiscal cliff**
- What do new weed laws really mean?
- Obama aides offer bipartisan approach to fiscal cliff
- Republicans look to pinpoint blame

Syrian refugee crisis hits new high

Turkey and Jordan say they are close to being overwhelmed by numbers as 11,000 flee country in a single day

- Latest developments from the Middle East
- US film-maker held in Libya emerges in Syria
- Bashar al-Assad vows to 'live or die' in Syria
- Julian Borger: the fragility of uneasy peace with Iran

Google services blocked in China

Firm says several services have been blocked just after country's once-in-a-decade national congress

- China to overtake US in four years
- Faithful toe the line at choreographed congress
- Editorial: Hu Jintao's last hurrah

Military warned over climate change

Betty White
TV's golden girl on 63 years in showbusiness

Roth retires
'I'm done'

Paper
Can it survive?

Sports

The big interview: Andrés Iniesta

Football: The midfield magician on fighting his way to the top and why Barcelona and Spain's style of play is above all 'pragmatic'

- Mancini told he has full backing of City hierarchy
- Man United lose Nani to hamstring injury
- Ferguson sets century goal for Man Utd
- James Richardson's European papers review

MLS playoffs: five things we learned

Seattle Sounders finally reach Conference Final; DC condemn Red Bulls to yet another conference semi-final defeat

28 comments

Open journalism

Get the whole picture

Subscribe

Get the Guardian on your cell phone or handheld device

About us

About Guardian US

Contact Guardian US

Guardian readers' editor

Guardian Weekly

Subscribe to the Guardian Weekly: 6 issues for $1

On this site

- **159** requests
- **844.13 KB** transferred

- DomContentLoaded: **1.99s**
- onload: **3.11s**

Critical Path

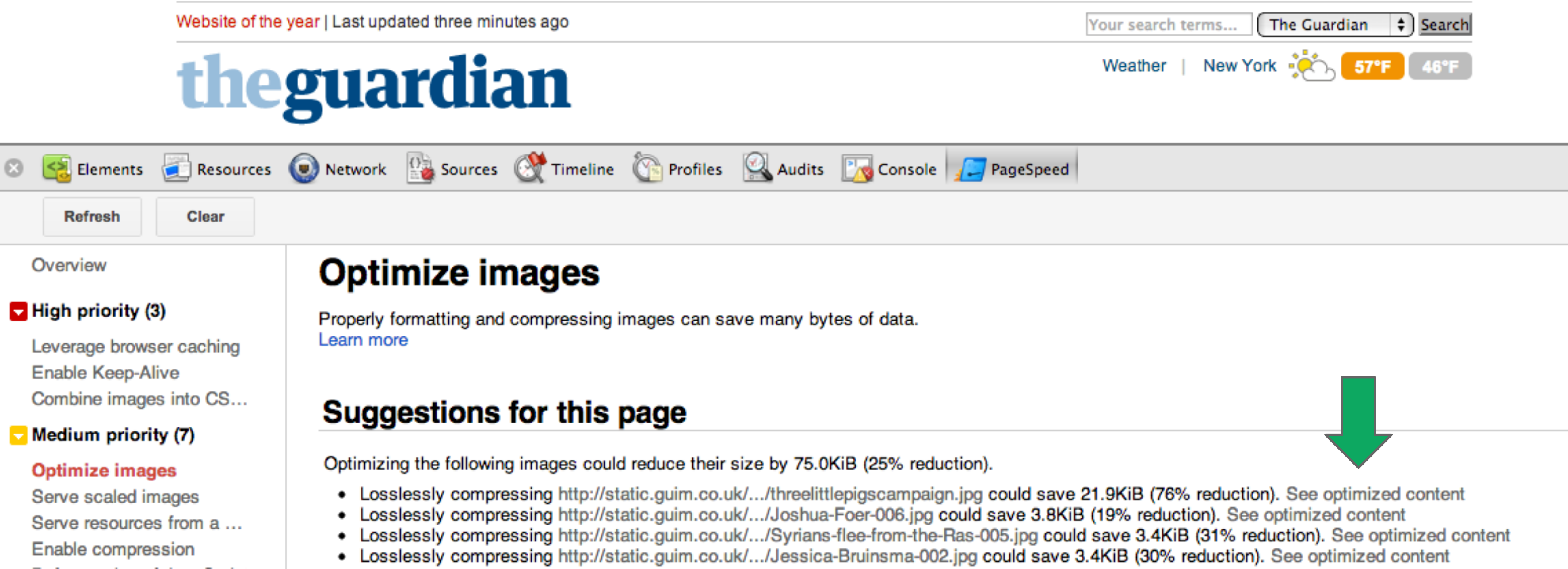
- 23 requests
- 300 ms in redirect latency
- 5 CSS files, mostly Javascript

Optimizing the page...

- Can we eliminate the redirect? Cache it?
- Can we reduce the overall size?
- Can we make fewer requests?
- Can we defer some of the Javascript?
- Can we combine some of the assets?

Analyzing PageSpeed extension...

Looks like we can **remove ~75kb of data** through better image compression!



Optimize images

Properly formatting and compressing images can save many bytes of data.
Learn more

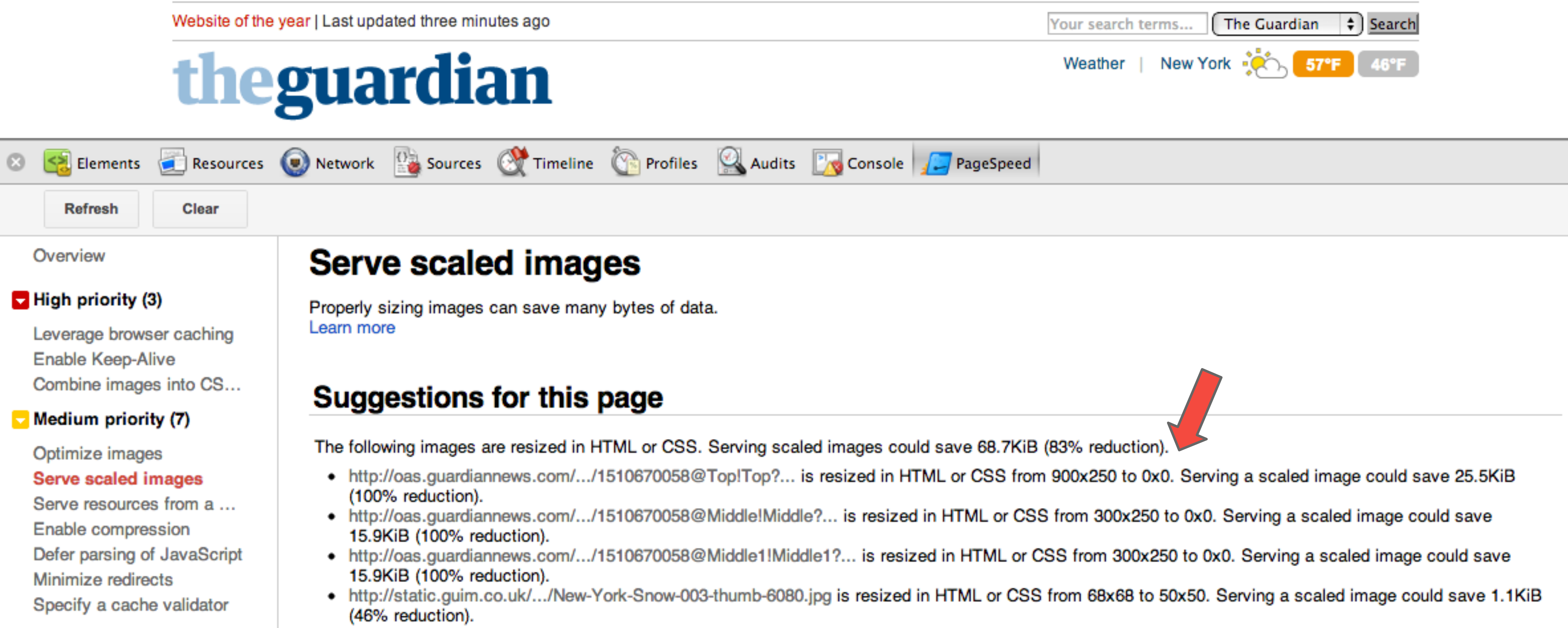
Suggestions for this page

Optimizing the following images could reduce their size by 75.0KiB (25% reduction).

- Losslessly compressing http://static.guim.co.uk/.../threelittlepigscampaign.jpg could save 21.9KiB (76% reduction). See optimized content
- Losslessly compressing http://static.guim.co.uk/.../Joshua-Foer-006.jpg could save 3.8KiB (19% reduction). See optimized content
- Losslessly compressing http://static.guim.co.uk/.../Syrians-flee-from-the-Ras-005.jpg could save 3.4KiB (31% reduction). See optimized content
- Losslessly compressing http://static.guim.co.uk/.../Jessica-Bruinsma-002.jpg could save 3.4KiB (30% reduction). See optimized content

Analyzing PageSpeed extension...

Hmmm... Resizing from **900x250** to **0x0**? Well, that's creative...



Serve scaled images

Properly sizing images can save many bytes of data.
Learn more

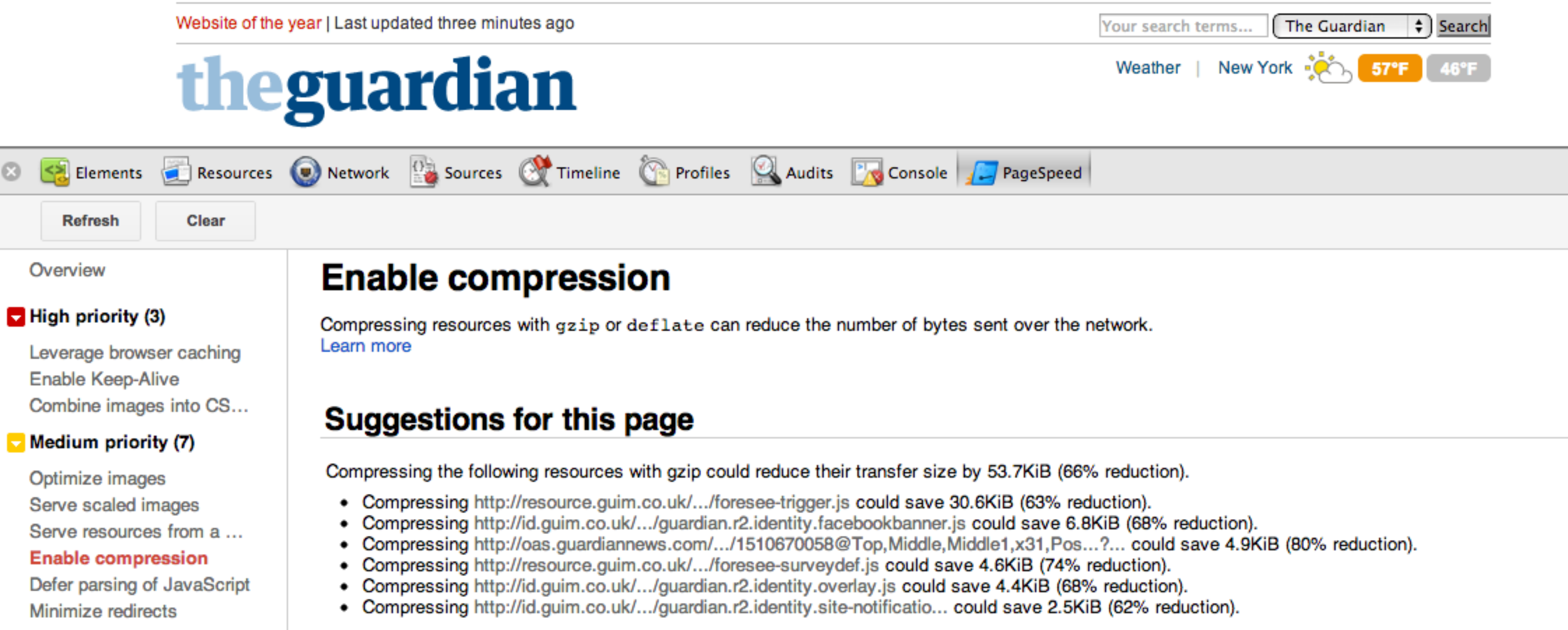
Suggestions for this page

The following images are resized in HTML or CSS. Serving scaled images could save 68.7KiB (83% reduction).

- http://oas.guardiannews.com/.../1510670058@Top!Top?... is resized in HTML or CSS from 900x250 to 0x0. Serving a scaled image could save 25.5KiB (100% reduction).
- http://oas.guardiannews.com/.../1510670058@Middle!Middle?... is resized in HTML or CSS from 300x250 to 0x0. Serving a scaled image could save 15.9KiB (100% reduction).
- http://oas.guardiannews.com/.../1510670058@Middle1!Middle1?... is resized in HTML or CSS from 300x250 to 0x0. Serving a scaled image could save 15.9KiB (100% reduction).
- http://static.guim.co.uk/.../New-York-Snow-003-thumb-6080.jpg is resized in HTML or CSS from 68x68 to 50x50. Serving a scaled image could save 1.1KiB (46% reduction).

Analyzing PageSpeed extension...

Looks like some of the **Javascript assets are not being compressed**! Another 53kb...



Enable compression

Compressing resources with `gzip` or `deflate` can reduce the number of bytes sent over the network.
Learn more

Suggestions for this page

Compressing the following resources with gzip could reduce their transfer size by 53.7KiB (66% reduction).

- Compressing http://resource.guim.co.uk/.../foresee-trigger.js could save 30.6KiB (63% reduction).
- Compressing http://id.guim.co.uk/.../guardian.r2.identity.facebookbanner.js could save 6.8KiB (68% reduction).
- Compressing http://oas.guardiannews.com/.../1510670058@Top,Middle,Middle1,x31,Pos...?... could save 4.9KiB (80% reduction).
- Compressing http://resource.guim.co.uk/.../foresee-surveydef.js could save 4.6KiB (74% reduction).
- Compressing http://id.guim.co.uk/.../guardian.r2.identity.overlay.js could save 4.4KiB (68% reduction).
- Compressing http://id.guim.co.uk/.../guardian.r2.identity.site-notificatio... could save 2.5KiB (62% reduction).

And more... #protip: try PageSpeed Insights.

*And try **Critical Path Explorer** in the online version...*

Performance Best Practices

Yo dawg, I heard you like top {N} lists...

Performance best practices, in context...

- **Reduce DNS lookups**
  - **130 ms** average lookup time! Even slower on mobile..
- **Avoid redirects**
  - Often results in **new handshake** (and maybe even DNS)
- **Make fewer HTTP requests**
  - No request is faster than no request
- **Flushing the document early**
  - Help document parser discover **external resources** early!
- **Use a CDN**
  - Faster RTT == faster page loads
  - Also, terminate SSL closer to the user!

Reduce the size of your pages!

- **GZIP your (text) assets**
  - ~80% compression ratio for text
- **Optimize images, pick optimal format**
  - ~60% of total size of an average page!
- **Add an Expires header**
  - No request is faster than no request
- **Add ETags**
  - Conditional checks to avoid fetching **duplicate content**

Optimize for fast first paint, don't block the parser!

- **Place stylesheets at the top**
  - Rendered, and potentially DOM construction, is blocked on CSS!
- **Load scripts asynchronously, whenever possible**
  - Sync scripts **block** the document parser
- **Place scripts at the bottom**
  - "Unblocks" the document parser (since there is nothing to block)
- **Minify, concatenate**
  - Remove redundant libraries & markup
  - Concatenate files to reduce number of HTTP requests

Hunt down & eliminate jank and memory leaks!

- **Build buttery smooth pages (scroll included)**
  - 60 FPS means 16.6 ms budget per frame
  - Use frames view to hunt down and eliminate jank
- **Leverage hardware acceleration where possible**
  - Let the GPU do what it's good at: alpha, translations
  - Avoid excessive CPU > GPU interaction
- **Eliminate JS and DOM memory leaks**
  - Monitor and diff heap usage to identify memory leaks
- **Test on mobile devices**
  - Emulators won't show you true performance on the device

Use (and learn) the right tools for the job

- **Learn about Developer Tools**
  - Spend some time reading the docs, follow tutorials
    - http://bit.ly/devtools-tips
- **PageSpeed Insights**
  - Install the browser extension for quick diagnostics
  - Leverage Critical Path Explorer to identify the... critical path!
- **WebPageTest.org**
  - Test your pages against multiple browsers
  - Test performance, not just UX acceptance!
- **Test on mobile devices**
  - Test with real mobile networks to get a feel for the differences

- Treat performance as a **business metric**, not a technical one
- Map **Real User Measurement** metrics to business outcomes

- Web performance & optimization is a **process**, not a checklist
- You should **design** with web performance in mind

- Always ask **"why"**, don't just follow a checklist

zomg, you made it.

Slides @ **bit.ly/webperf-crash-course**

Twitter	@igrigorik
G+	gplus.to/igrigorik
Web	igvita.com