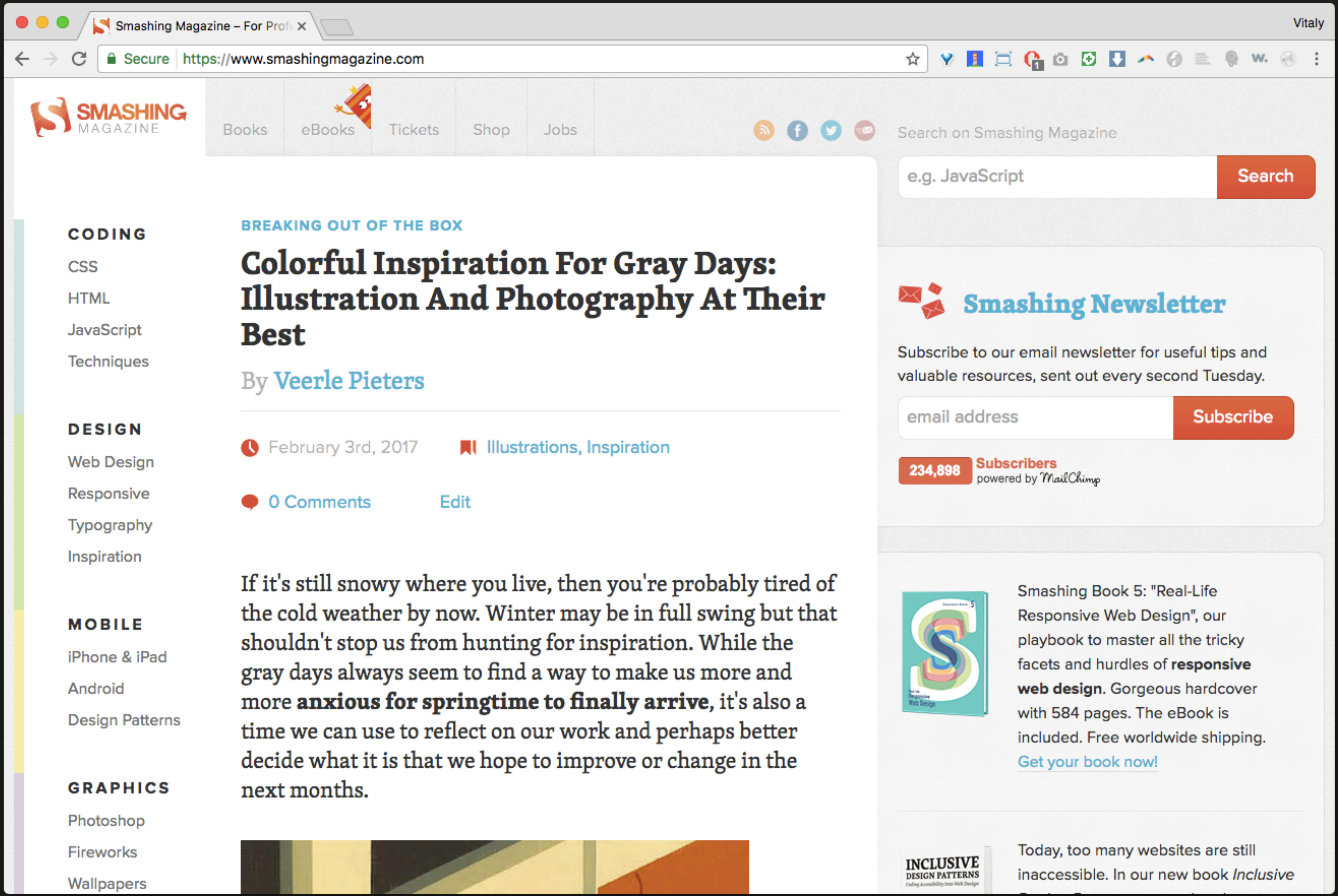
New Adventures In Responsive Web Design

Moscow, Russia @ HolyJS
December 10, 2017

Vitaly Friedman, ex-editor-in-chief and co-founder of SmashingMag


Smashing Magazine – For Prof
Vitaly
Secure
https://www.smashingmagazine.com
SMASHING MAGAZINE
Books
eBooks
Tickets
Shop
Jobs
Search on Smashing Magazine
e.g. JavaScript
Search

CODING

- CSS
- HTML
- JavaScript
- Techniques

DESIGN

- Web Design
- Responsive
- Typography
- Inspiration

MOBILE

- iPhone & iPad
- Android
- Design Patterns

GRAPHICS

- Photoshop
- Fireworks
- Wallpapers

BREAKING OUT OF THE BOX

Colorful Inspiration For Gray Days: Illustration And Photography At Their Best

By Veerle Pieters

February 3rd, 2017 Illustrations, Inspiration

0 Comments Edit

If it's still snowy where you live, then you're probably tired of the cold weather by now. Winter may be in full swing but that shouldn't stop us from hunting for inspiration. While the gray days always seem to find a way to make us more and more **anxious for springtime to finally arrive**, it's also a time we can use to reflect on our work and perhaps better decide what it is that we hope to improve or change in the next months.

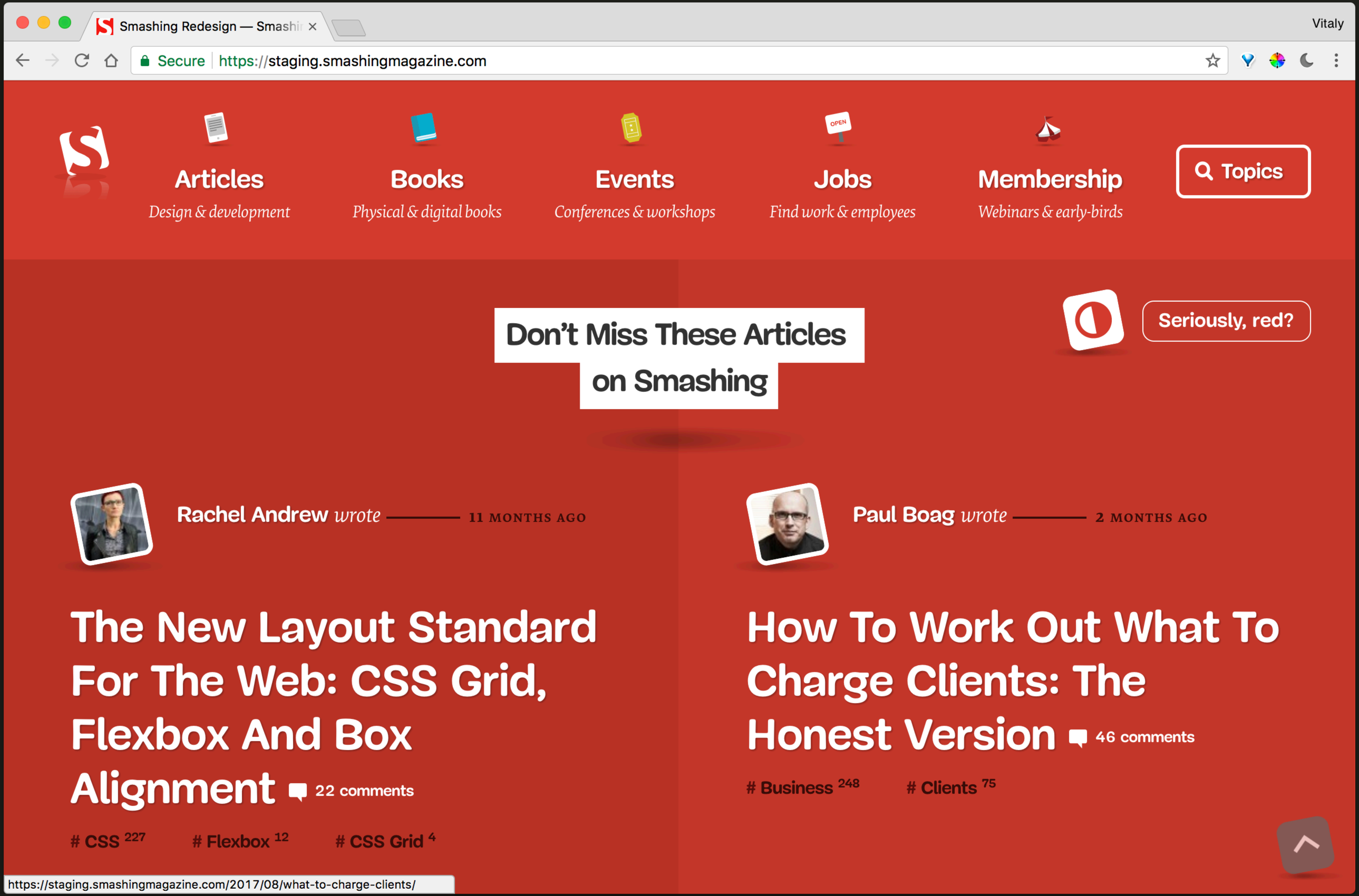
Smashing Newsletter

Subscribe to our email newsletter for useful tips and valuable resources, sent out every second Tuesday.

234,898 Subscribers powered by MailChimp

Smashing Book 5: "Real-Life Responsive Web Design", our playbook to master all the tricky facets and hurdles of **responsive web design**. Gorgeous hardcover with 584 pages. The eBook is included. Free worldwide shipping. Get your book now!

Today, too many websites are still inaccessible. In our new book *Inclusive*



Smashing Redesign — Smashi ×
Vitaly
Secure | https://staging.smashingmagazine.com

Articles
Design & development

Books
Physical & digital books

Events
Conferences & workshops

Jobs
Find work & employees

Membership
Webinars & early-birds

Don't Miss These Articles on Smashing

Rachel Andrew *wrote* — 11 MONTHS AGO

The New Layout Standard For The Web: CSS Grid, Flexbox And Box Alignment 22 comments

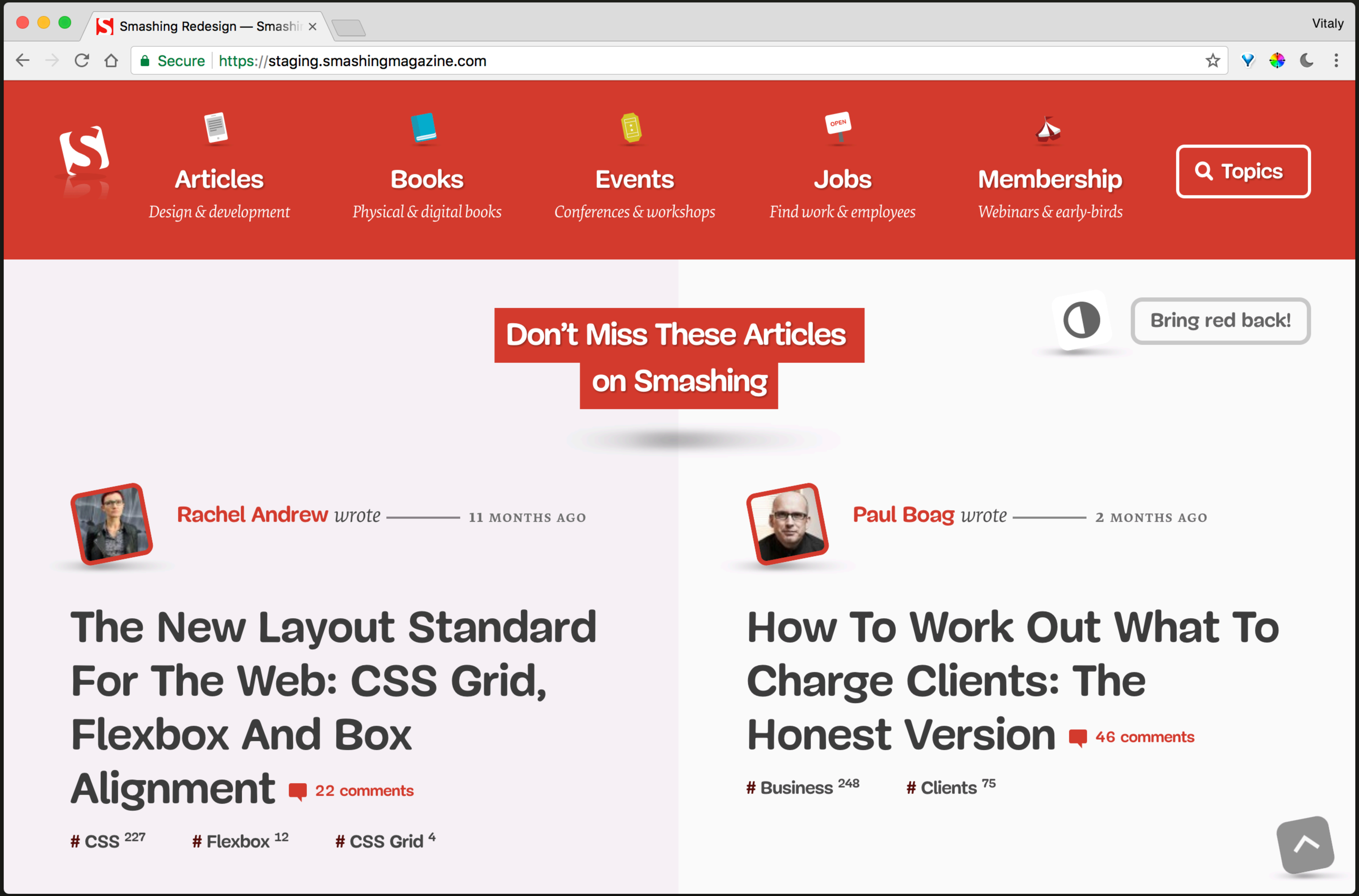
CSS 227 # Flexbox 12 # CSS Grid 4

Paul Boag *wrote* — 2 MONTHS AGO

How To Work Out What To Charge Clients: The Honest Version 46 comments

Business 248 # Clients 75

https://staging.smashingmagazine.com/2017/08/what-to-charge-clients/



Don't Miss These Articles on Smashing

Rachel Andrew *wrote* — 11 MONTHS AGO

The New Layout Standard For The Web: CSS Grid, Flexbox And Box Alignment 22 comments

CSS 227 # Flexbox 12 # CSS Grid 4

Paul Boag *wrote* — 2 MONTHS AGO

How To Work Out What To Charge Clients: The Honest Version 46 comments

Business 248 # Clients 75

RESPONSIVE ADVENTURES

FRONT-END CHALLENGES YOU'VE NEVER EXPERIENCED BEFORE. _

RESPONSIVE ADVENTURES

FRONT-END CHALLENGES YOU'VE NEVER EXPERIENCED BEFORE. _

CHOOSE
ADVENTURE
EASY
MEDIUM
HARDCORE
PRESS ENTER TO CONTINUE

LEVEL 1

COMPRESSION

Compression matters. What’s the best strategy to compress assets/content these days? Essentially, we want to *minimize* bandwidth and speed up *delivery*. How?

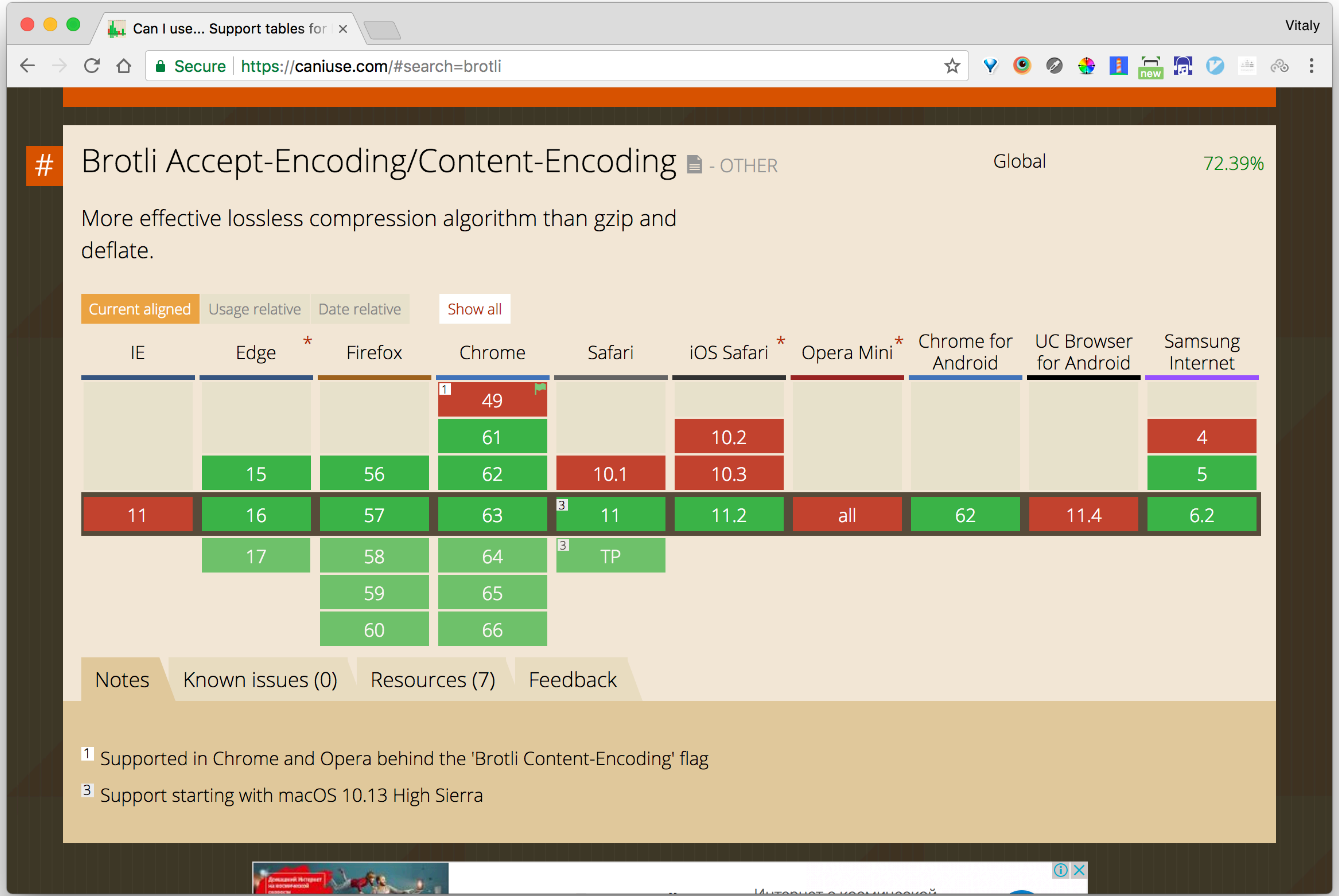
gzip is the most common compression format on the web; its most common implementation is *zlib*, and it uses a combination of LZ77 and Huffman encoding algorithms (called *deflate*).

Each compression library (like *zlib*) has *preset quality settings*, ranging from *fast* compression (levels 1–3) to *slow* compression (levels 4–9).

As developers, we care about the *transferred file size* and *compression* / *decompression* speed — for *both* static and dynamic web content.

> *Zopfli* can be thought of as a way to do a "very good, but slow, deflate or zlib compression". High compression ratio at the cost of a higher overhead for compressing. *Backwarts-compatible* for browsers that support only gzip.
>
> — *Cody Ray Hoeft*
>
> *https://www.quora.com/What-is-Brotli-How-is-it-different-from-Zopfli*

> “*Brotli* is a whole new compression and decompression format. For Brotli, browser support has to be built into the browser. *Future-compatible* with the next generation of browsers.
>
> — *Cody Ray Hoeft*
>
> *https://www.quora.com/What-is-Brotli-How-is-it-different-from-Zopfli*



Can I use... Support tables for
Vitaly
Secure
https://caniuse.com/#search=brotli

Brotli Accept-Encoding/Content-Encoding - OTHER

Global 72.39%

More effective lossless compression algorithm than gzip and deflate.

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			[1] 49						
			61		10.2				4
	15	56	62	10.1	10.3				5
11	16	57	63	[3] 11	11.2	all	62	11.4	6.2
	17	58	64	[3] TP					
		59	65						
		60	66						

[1] Supported in Chrome and Opera behind the 'Brotli Content-Encoding' flag

[3] Support starting with macOS 10.13 High Sierra

> “*Brotli* is a whole new lossless compression and decompression format. For Brotli, browser support has to be built into the browser. *Future-compatible* with the next generation of browsers.
>
> — *Cody Ray Hoeft*
>
> *https://www.quora.com/What-is-Brotli-How-is-it-different-from-Zopfli*

Brotli and Zopfli

- Compared to *gzip, Brotli* is significantly slower at compressing data, but provides much better savings.

  — *Brotli* is an open-sourced, lossless compression format,
  — *Brotli* shows significant improvements for static content,
  — *Brotli's* decompression is fast: comparable to *zlib,*
  — *Brotli* has an advantage for large fiels on *slow connections,*
  — Expect 14-39% file savings on text-based assets (level 4),
  — Ideal for HTML, CSS, JavaScript, SVG — anything text-based.
  — Brotli support is restricted to *HTTPS* connections.

Capable browsers advertise their ability to accept Brotli-compressed resources in the `Accept-Encoding` request header.

```
Accept-Encoding: gzip, deflate, sdch, br
```

If both browser and server support Brotli, the server uses Brotli compression and sets the `Content-Encoding` header of the response to **br** accordingly.

```
Content-Encoding: br
```

```
brotli_static on;
```

Enabling this will cause Brotli to look for files that have already been compressed and serve those files directly without performing any compression itself. What we also did, was to remove the `brotli_types` directive that instructs the server to compress **all** file types. This makes Brotli fall back to its default of only compressing the default mime type **text/html**. `brotli_static` then makes sure that Nginx returns the statically compressed version for any resource that it can match with a brotlified counterpart.

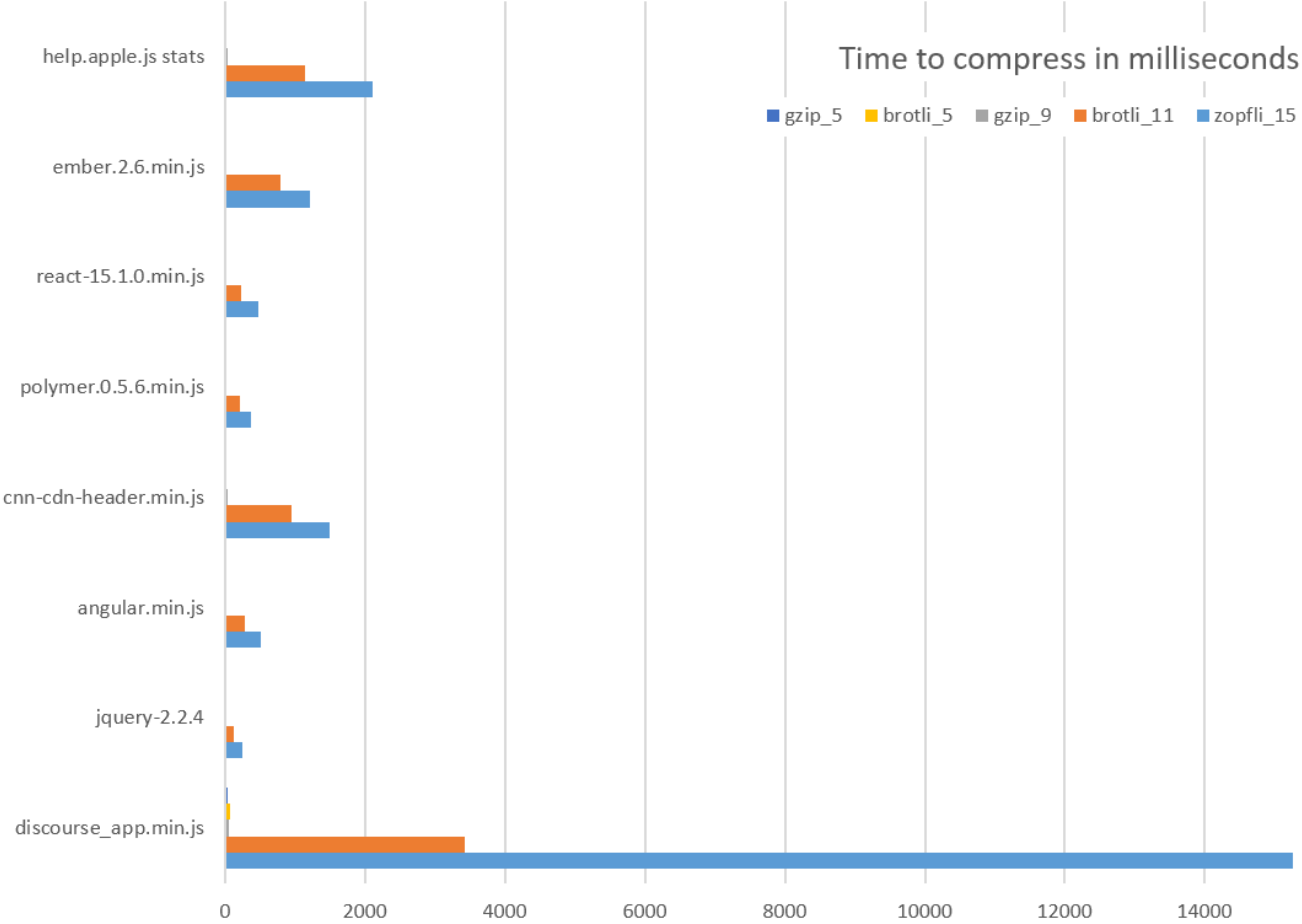
```
dist/assets/css/
  main-7bca136736.css
  main-7bca136736.css.gz
  main-7bca136736.css.br
```

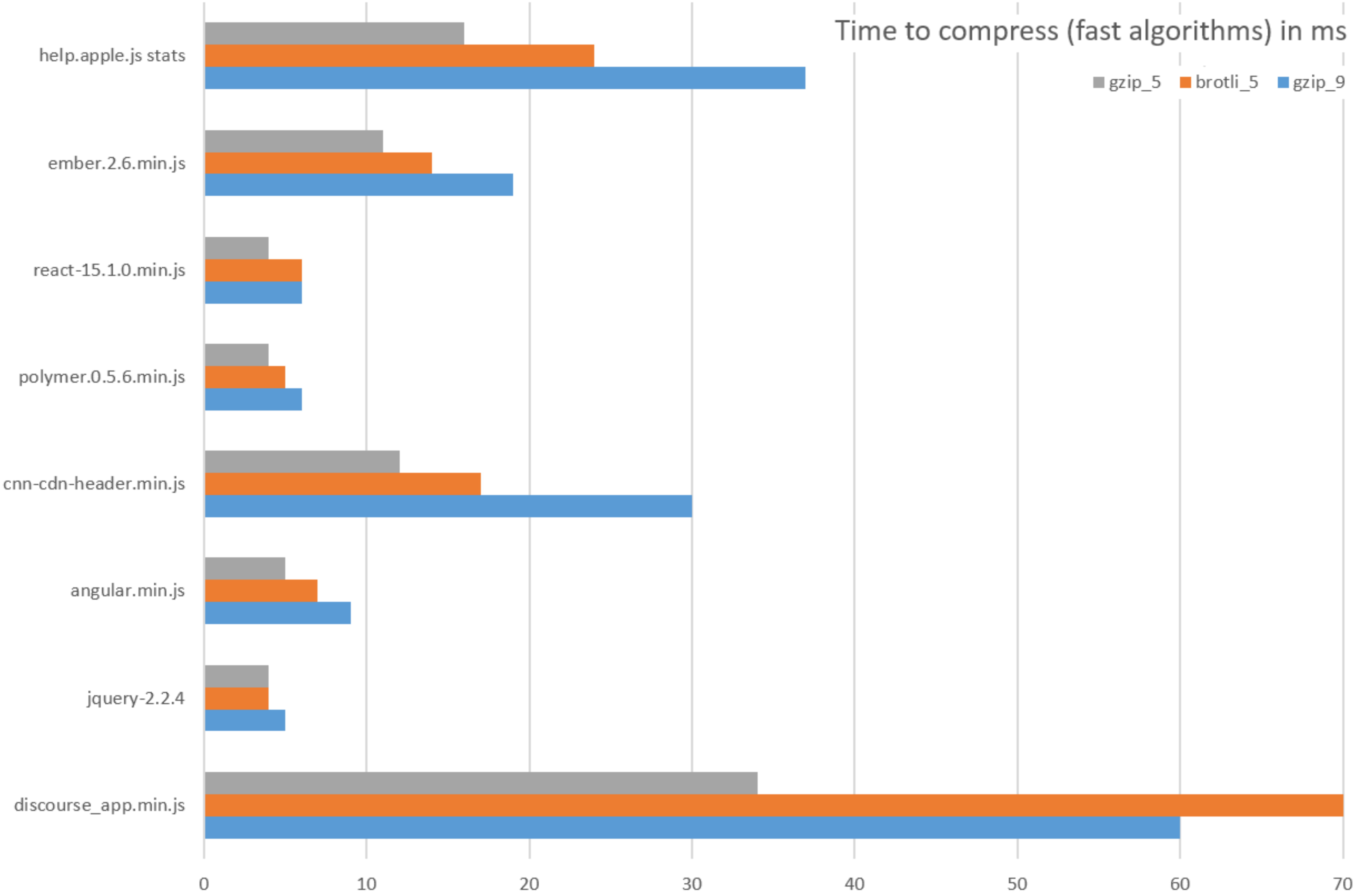
The relevant bit of Nginx configuration now looks like this:

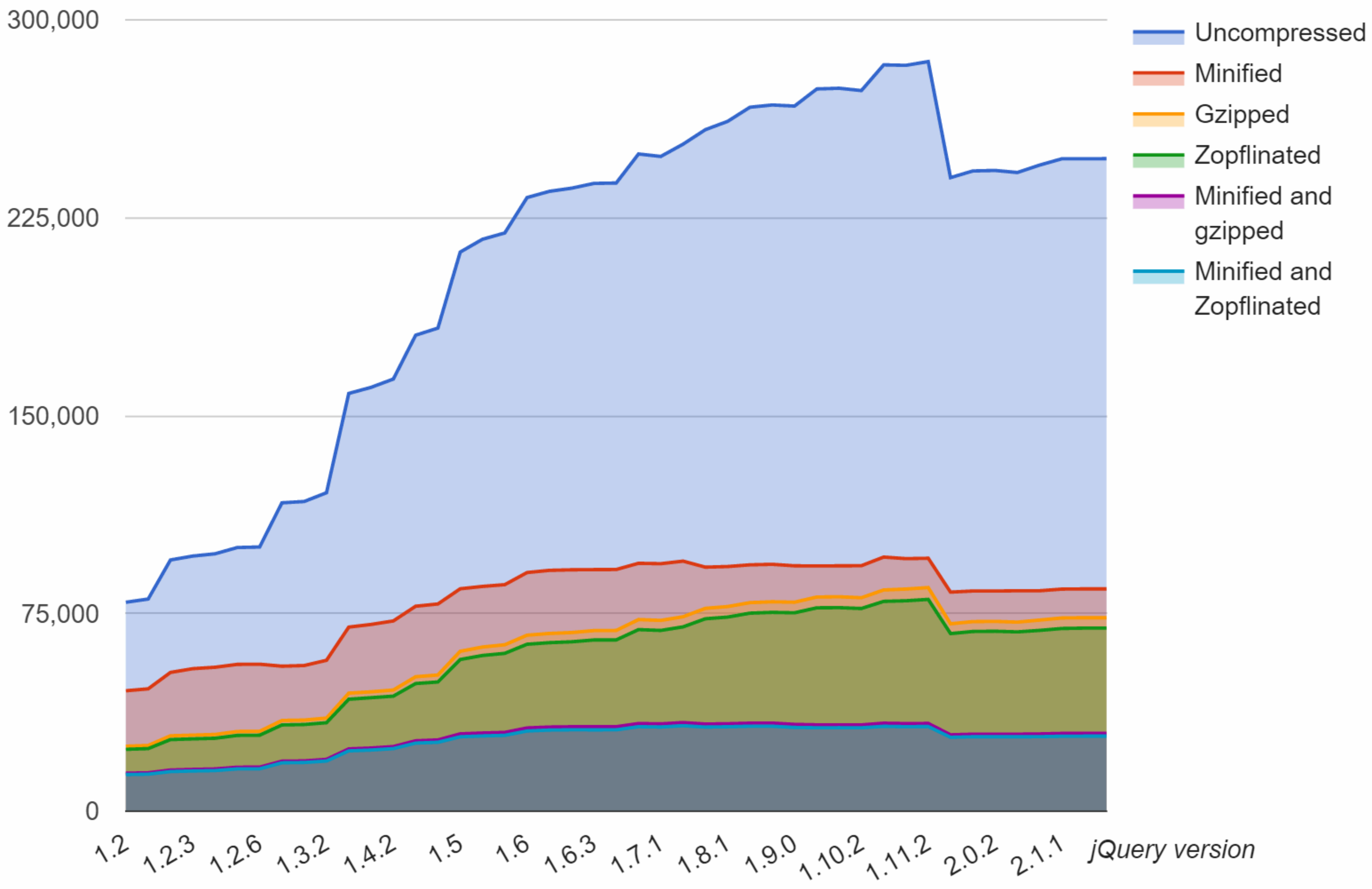
```
http {
  # ...truncated...
  gzip on;
  gzip_static on;
  gzip_vary on;

  brotli on;
  brotli_comp_level 4;
  brotli_static on;
  # ...truncated...
}
```

Brotli and Zopfli

- Compared to *gzip, Brotli* is significantly slower at compressing data, but provides much better savings.
  - — Browsers *advertise support* via *Accept-Encoding* request header: `Accept-Encoding: gzip, deflate, sdch, br`
  - — Servers can *choose to use* Brotli and serve `Content-Encoding: br`
  - — You might need to *recompile your server* to include a Brotli module (available for Apache, Nginx, IIS).
  - — *Zopfli* often not applicable for on-the-fly compression, but a good alternative for one-time compression of static content.


Time to compress in milliseconds
gzip_5
brotli_5
gzip_9
brotli_11
zopfli_15
help.apple.js stats
ember.2.6.min.js
react-15.1.0.min.js
polymer.0.5.6.min.js
cnn-cdn-header.min.js
angular.min.js
jquery-2.2.4
discourse_app.min.js
0
2000
4000
6000
8000
10000
12000
14000


Time to compress (fast algorithms) in ms
gzip_5
brotli_5
gzip_9
help.apple.js stats
ember.2.6.min.js
react-15.1.0.min.js
polymer.0.5.6.min.js
cnn-cdn-header.min.js
angular.min.js
jquery-2.2.4
discourse_app.min.js
0
10
20
30
40
50
60
70


300,000
225,000
150,000
75,000
0
1.2
1.2.3
1.2.6
1.3.2
1.4.2
1.5
1.6
1.6.3
1.7.1
1.8.1
1.9.0
1.10.2
1.11.2
2.0.2
2.1.1
jQuery version
Uncompressed
Minified
Gzipped
Zopflinated
Minified and gzipped
Minified and Zopflinated

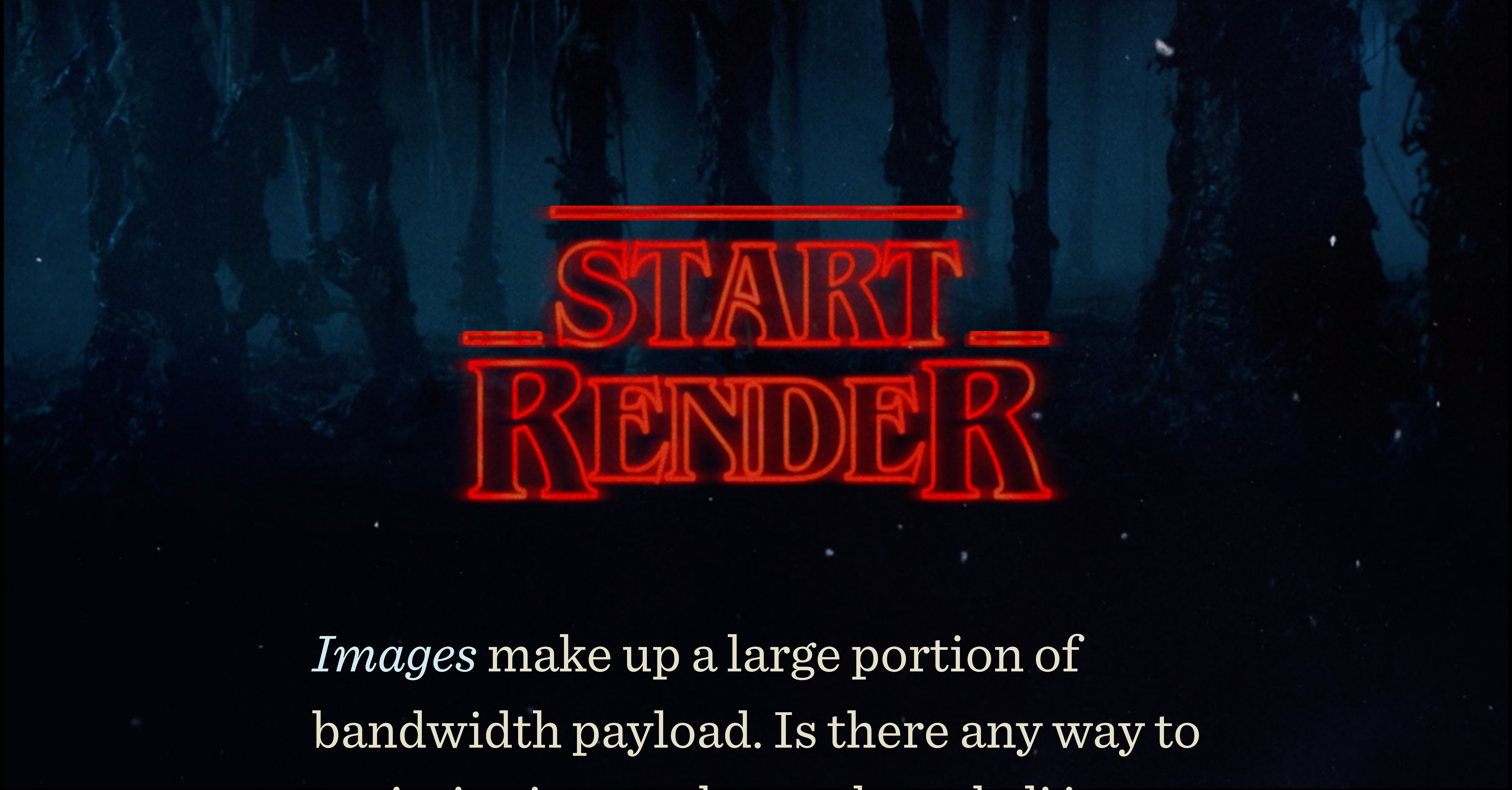
Brotli/Zopfli Compression Strategy

- Compared to *gzip, Brotli* is significantly slower at compressing data, but provides much better savings.

— *Pre-compress* static assets with Brotli+Gzip at the highest level,
— *Compress* (dynamic) HTML *on the fly* with Brotli at level 1–4.
— *Check for Brotli support on CDNs* (KeyCDN, CDN77, Fastly).
— *Server handles content negotiation* for Brotli or gzip.
— *Use Zopfli* if you can't install/maintain Brotli on the server.

"Results of experimenting with Brotli for dynamic web content", https://blog.cloudflare.com/results-experimenting-brotli/,
Tim Kadlec, *"Understanding Brotli's Potential"*, https://blogs.akamai.com/2016/02/understanding-brotlis-potential.html,
"Static site implosion with Brotli and Gzip", https://www.voorhoede.nl/en/blog/static-site-implosion-with-brotli-and-gzip/
"Current state of Brotli compression", https://samsaffron.com/archive/2016/06/15/the-current-state-of-brotli-compression

LEVEL 2 IMAGES



Images make up a large portion of bandwidth payload. Is there any way to optimize images beyond good ol’ image optimization? What if a *hero image has to render fast*, e.g. on **landing pages?**

STATE OF THE WEB ON MOBILE

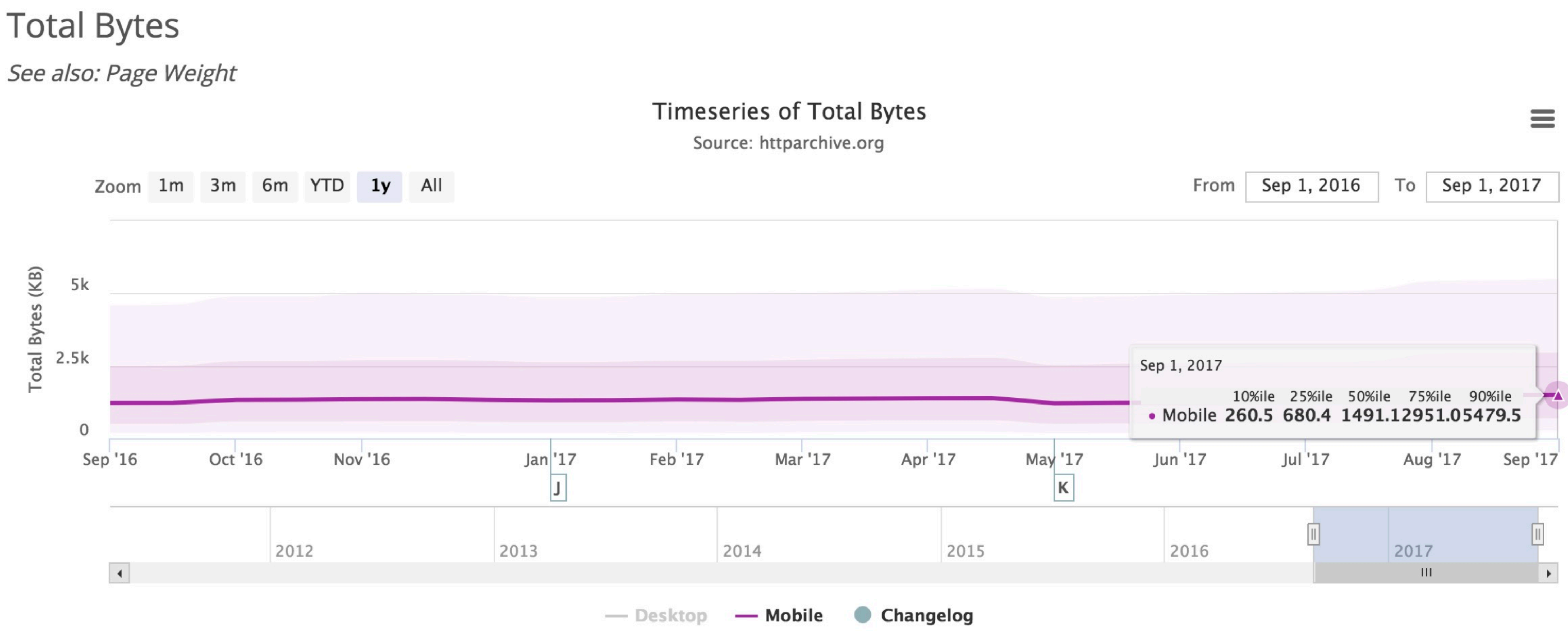
SITES ARE SENDING USERS...

P90: 5.4MB

P75: 2.9MB

P90 3.8MB (70%) of this is images

P90 1MB (18%) of this is JS


Total Bytes

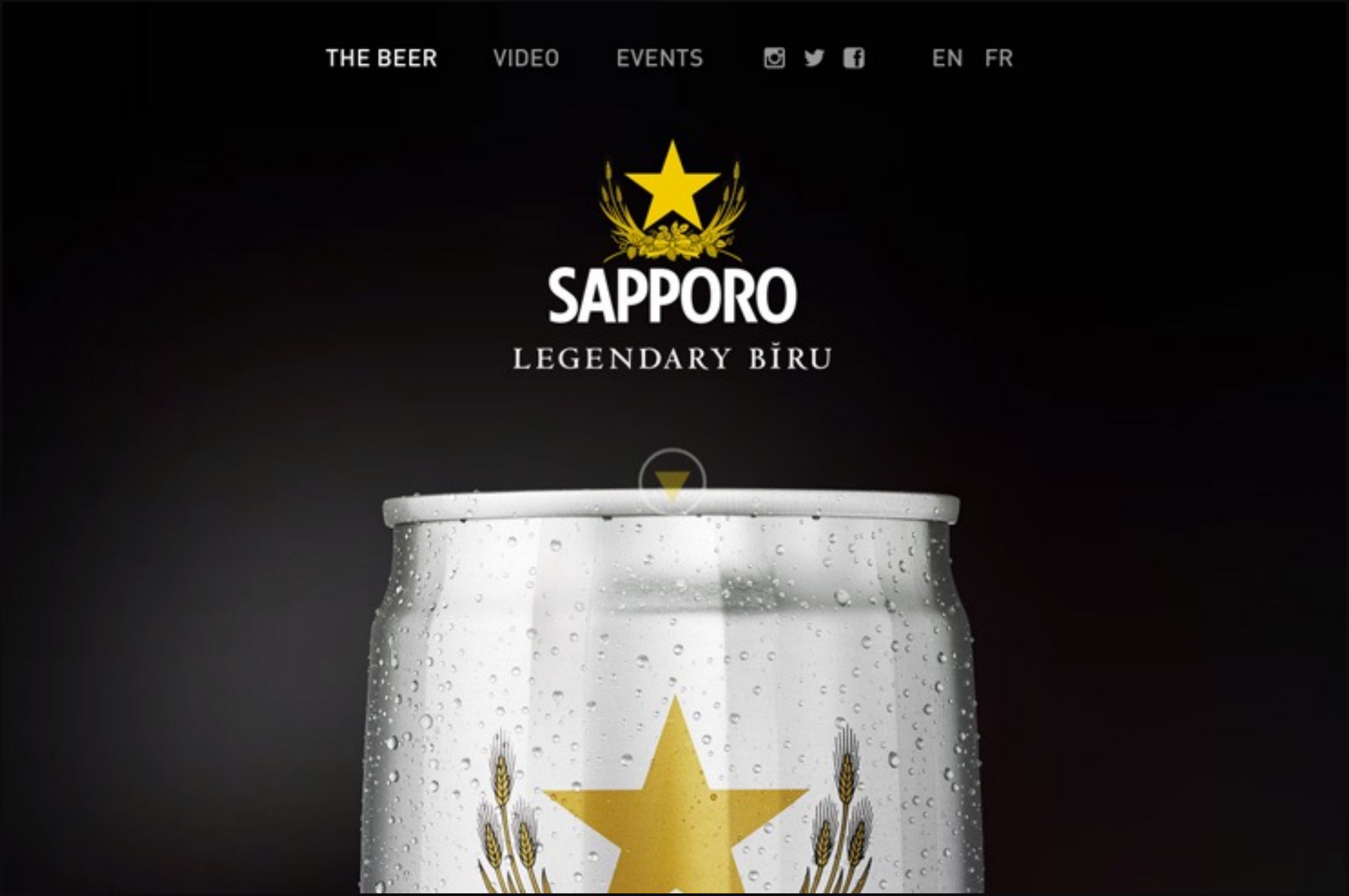
See also: Page Weight

http archive

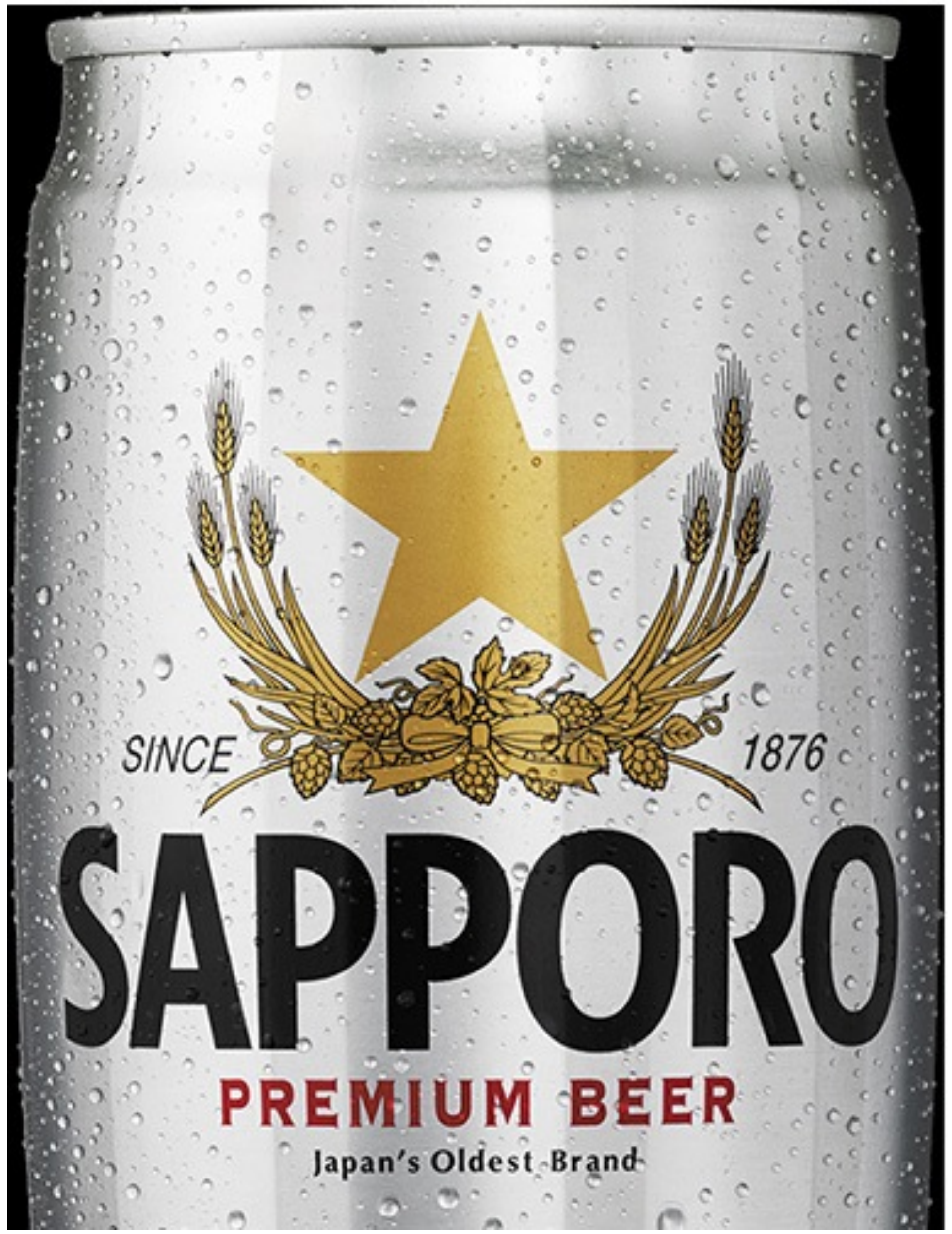
Using Dev Tools mobile emulation, Moto G4 calibrated CPU, Cable (5/1mbps, 28ms)

http://beta.httparchive.org

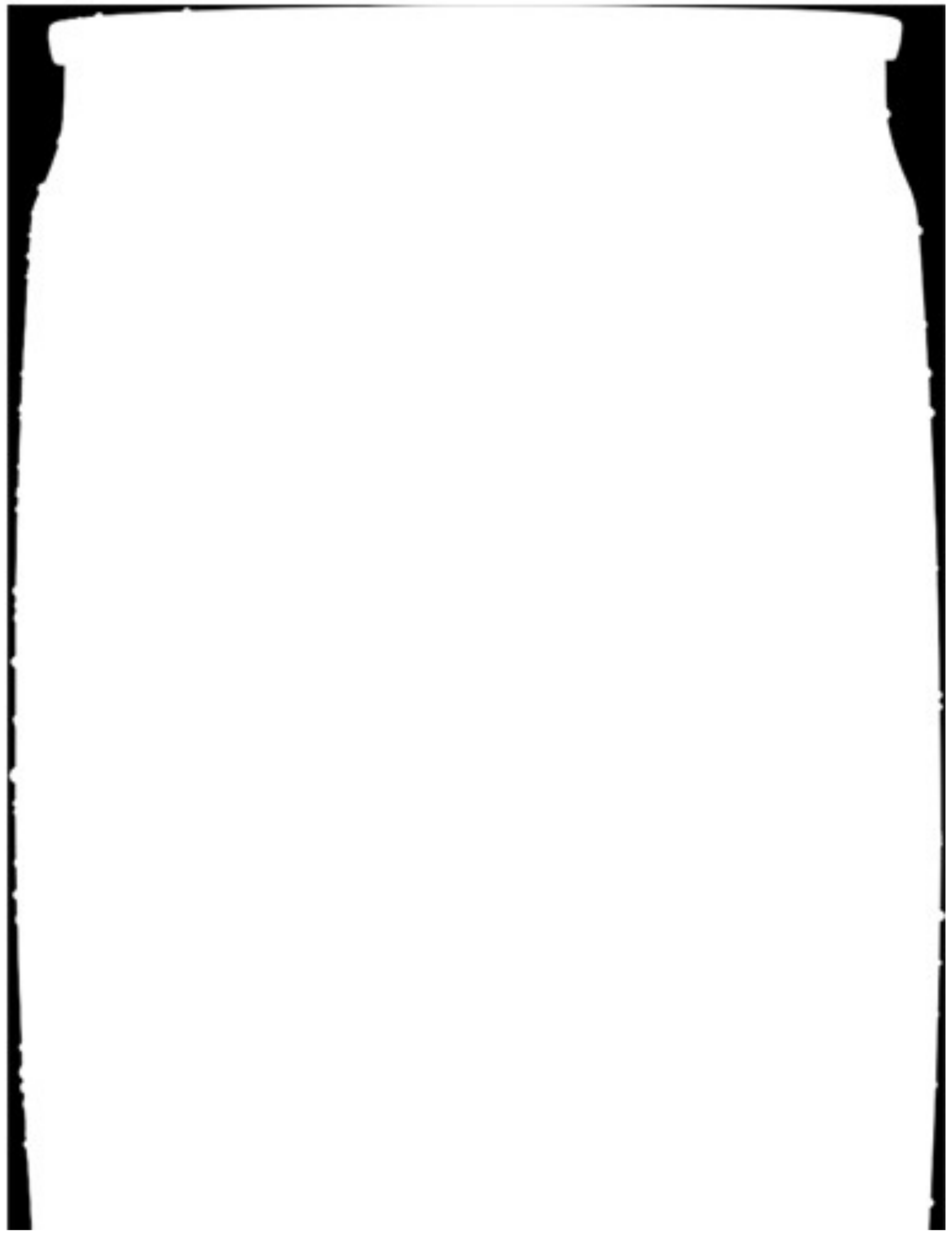
> “What if you have a large photo that requires a transparent shadow? PNG is way too large in file size, and JPEG isn’t good enough in quality because of the gradient in the background. What do you do?


THE BEER
VIDEO
EVENTS
EN
FR
SAPPORO
LEGENDARY BĪRU

CAN-TOP.JPG (260KB)



CAN-TOP-ALPHA.PNG (11KB)



- hero-image.svg:

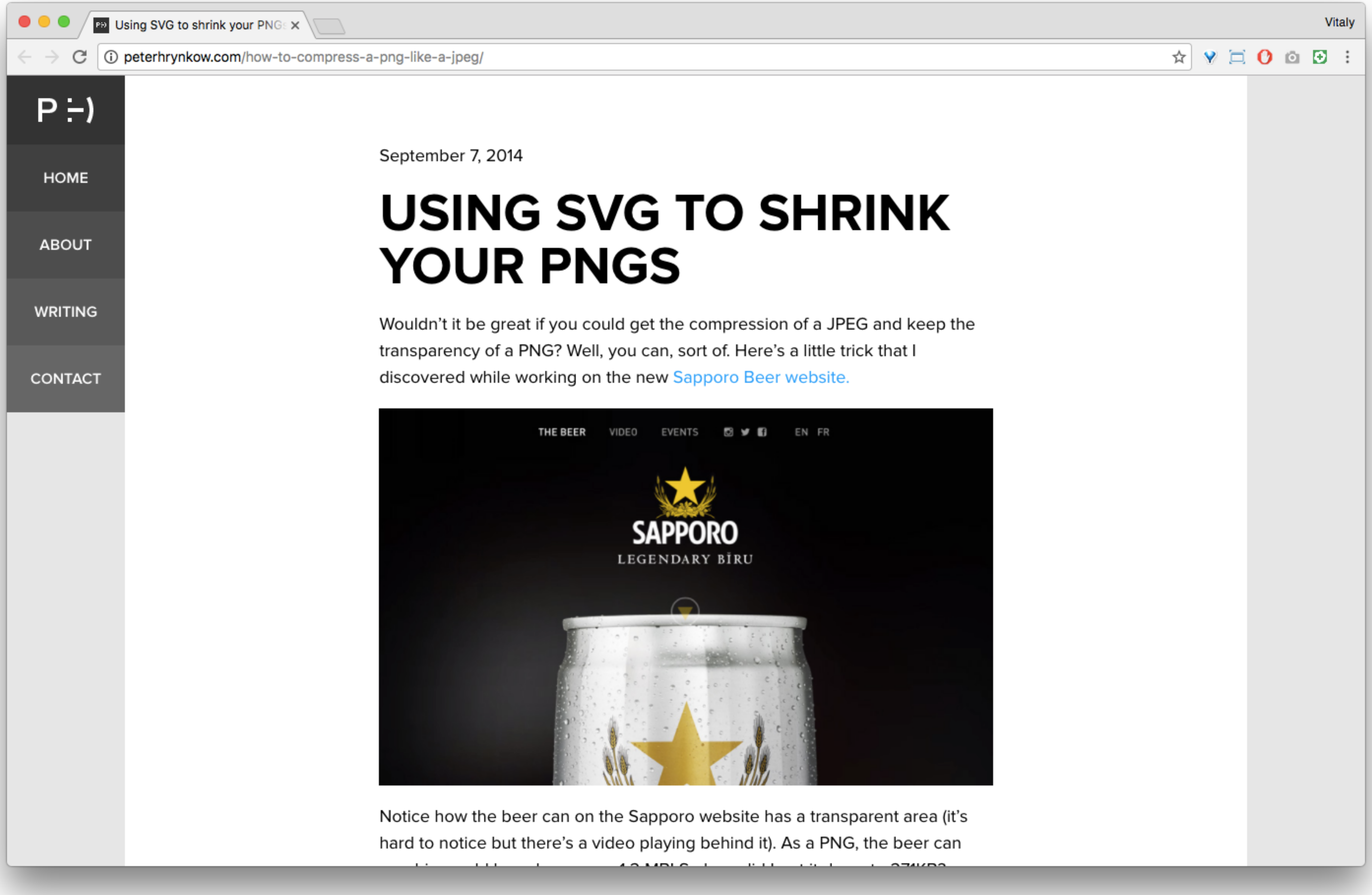
```
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" viewbox="0 0 560 1388">

<defs>

    <mask id="canTopMask">

        <image width="560" height="1388" xlink:href="can-top-alpha.png">
        </image>

    </mask>

</defs>

    <image mask="url(#canTopMask)" id="canTop" width="560" height="1388"
           xlink:href="can-top.jpg"></image>
</svg>
```

- hero-image.svg:

```
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" viewbox="0 0 560 1388">
<defs>
    <mask id="canTopMask">
        <image width="560" height="1388" xlink:href="can-top-alpha.png">
        </image>
    </mask>
</defs>
    <image mask="url(#canTopMask)" id="canTop" width="560" height="1388"
          xlink:href="can-top.jpg"></image>
</svg>
```

- HTML/CSS:

```
<img src="hero-image.svg" />, background: url("hero-image.svg")
```


Using SVG to shrink your PNG
Vitaly
peterhrynkow.com/how-to-compress-a-png-like-a-jpeg/
P :-)
HOME
ABOUT
WRITING
CONTACT

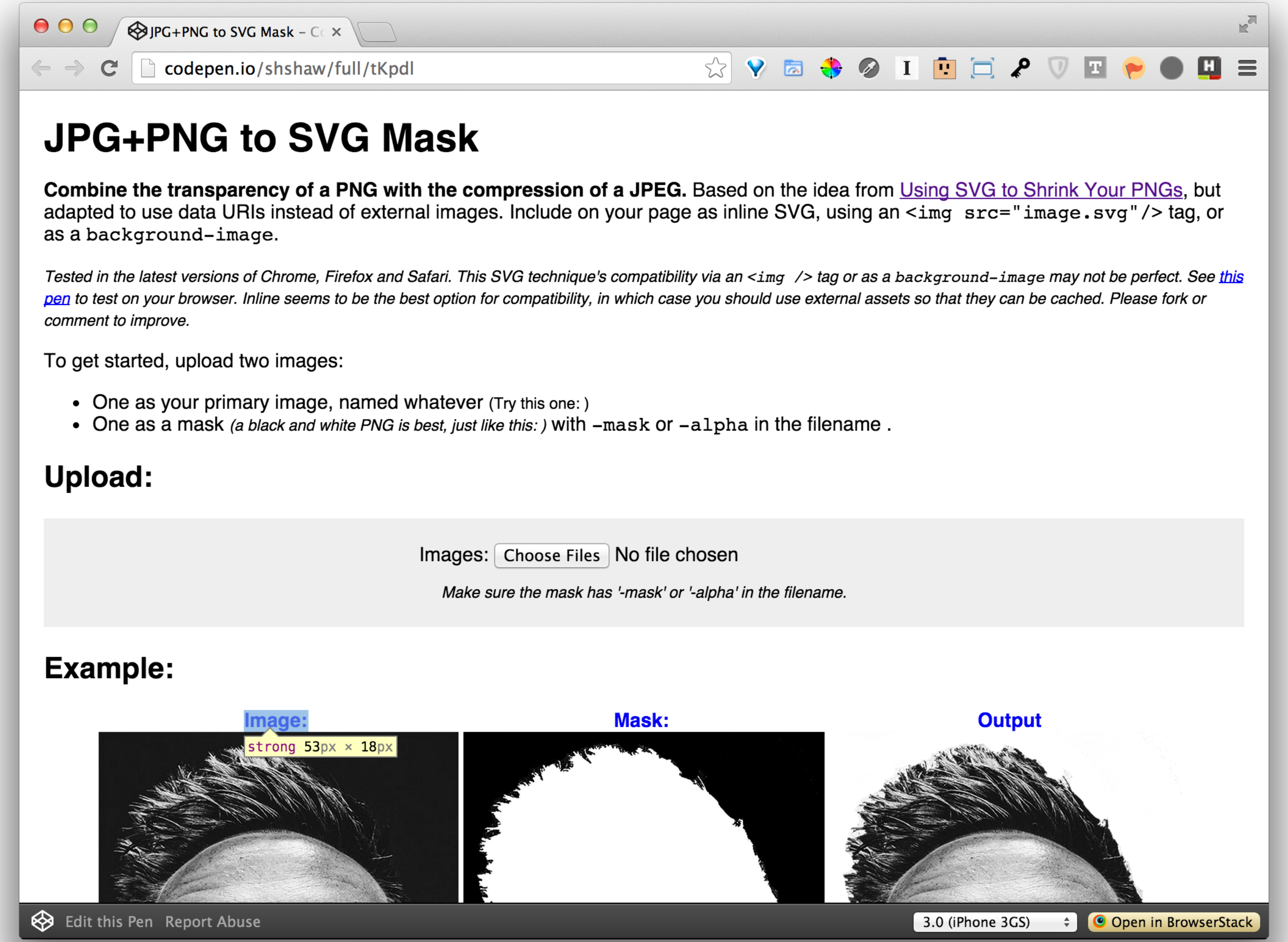
September 7, 2014

USING SVG TO SHRINK YOUR PNGS

Wouldn't it be great if you could get the compression of a JPEG and keep the transparency of a PNG? Well, you can, sort of. Here's a little trick that I discovered while working on the new Sapporo Beer website.

THE BEER
VIDEO
EVENTS
EN FR
SAPPORO
LEGENDARY BIRU

Notice how the beer can on the Sapporo website has a transparent area (it's hard to notice but there's a video playing behind it). As a PNG, the beer can



JPG+PNG to SVG Mask

Combine the transparency of a PNG with the compression of a JPEG. Based on the idea from Using SVG to Shrink Your PNGs, but adapted to use data URIs instead of external images. Include on your page as inline SVG, using an `<img src="image.svg"/>` tag, or as a `background-image`.

Tested in the latest versions of Chrome, Firefox and Safari. This SVG technique's compatibility via an `<img />` tag or as a `background-image` may not be perfect. See this pen to test on your browser. Inline seems to be the best option for compatibility, in which case you should use external assets so that they can be cached. Please fork or comment to improve.

To get started, upload two images:

- One as your primary image, named whatever (Try this one:)
- One as a mask *(a black and white PNG is best, just like this:)* with `-mask` or `-alpha` in the filename .

Upload:

Images: Choose Files No file chosen

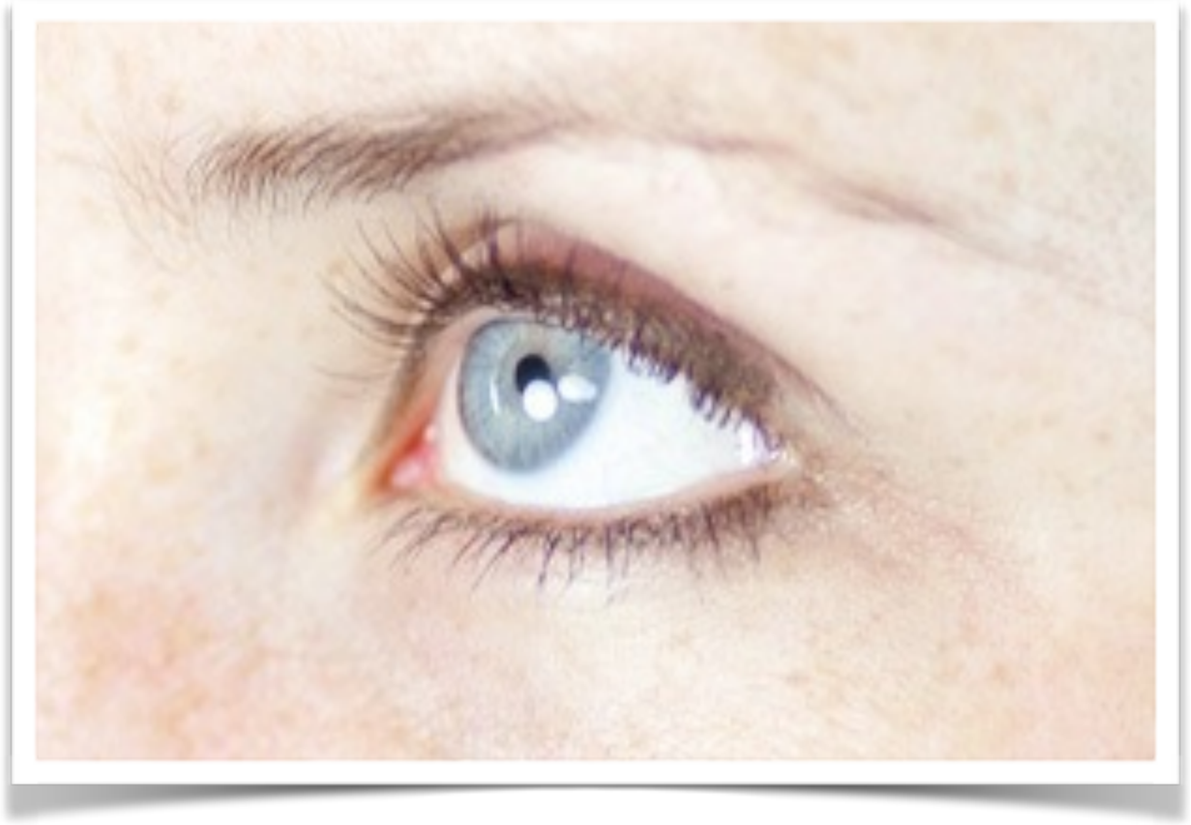
Make sure the mask has '-mask' or '-alpha' in the filename.

Example:

> “…Given two *identical* images that are displayed *at the same size* on a website, one can be dramatically smaller than the other in file size if it’s *highly compressed* and *dramatically larger* in dimensions than it is displayed in.
>
> — *Daan Jobsis*

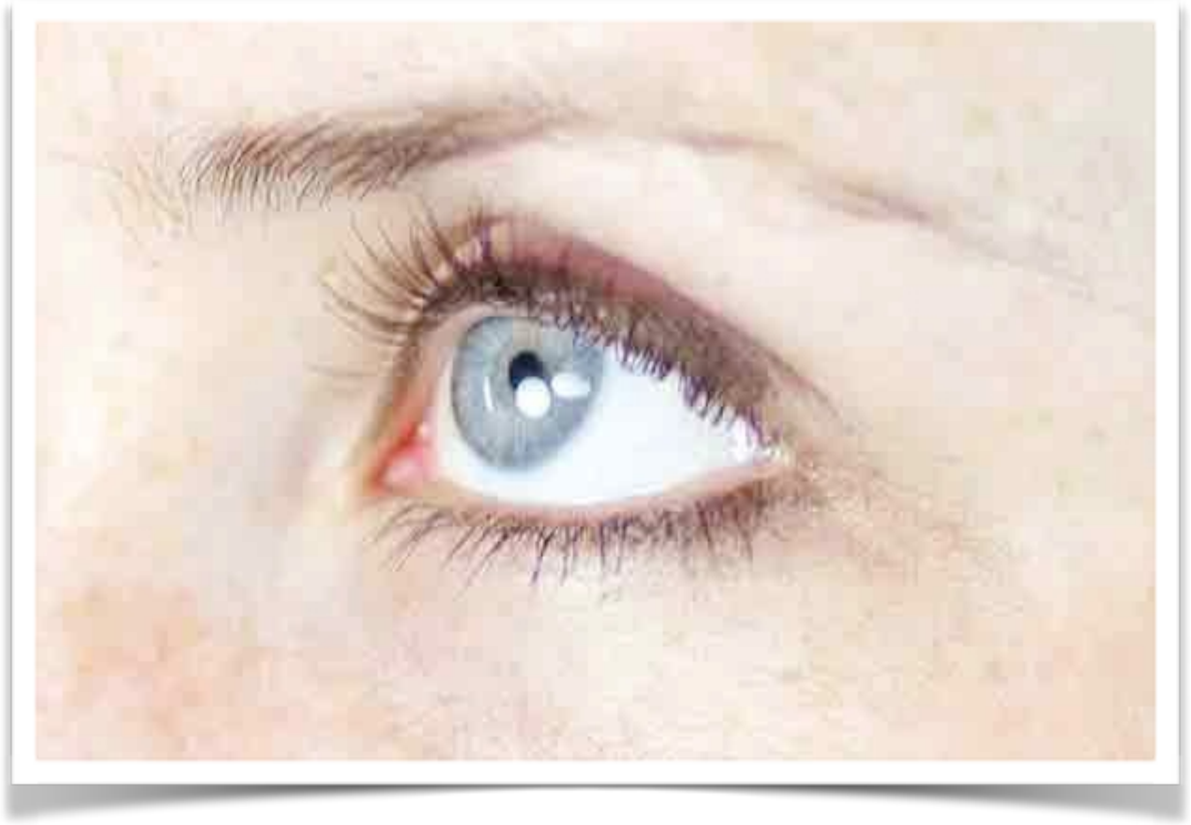
600×400px file, 0% JPEG quality,
displayed in **600×400** (file size 7 Kb)

600×400px file, 0% JPEG quality,
displayed in **300×200** (file size 7 Kb)



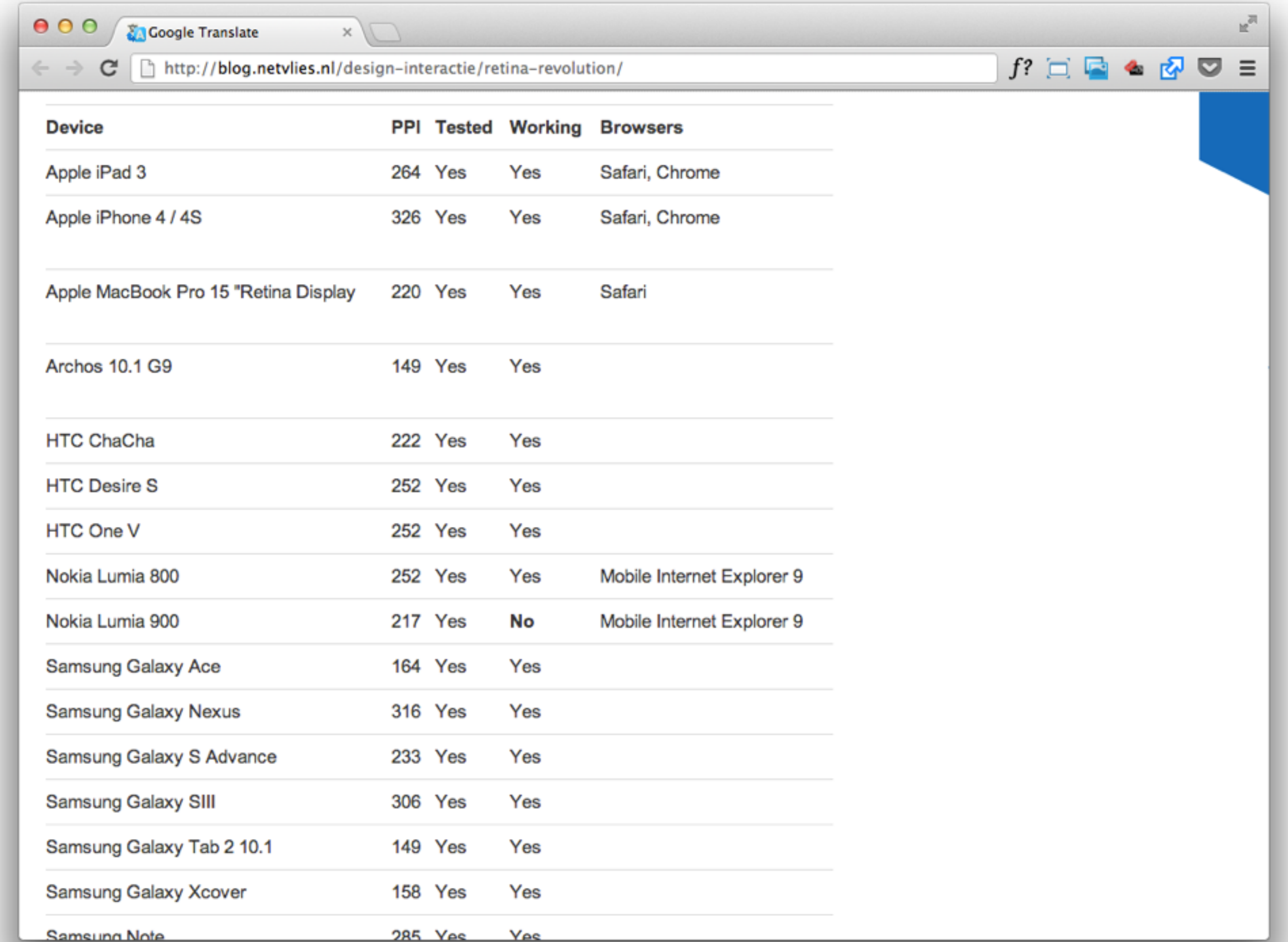
300×200px file **(21 Kb)**

80% JPEG quality
displayed in **300×200**

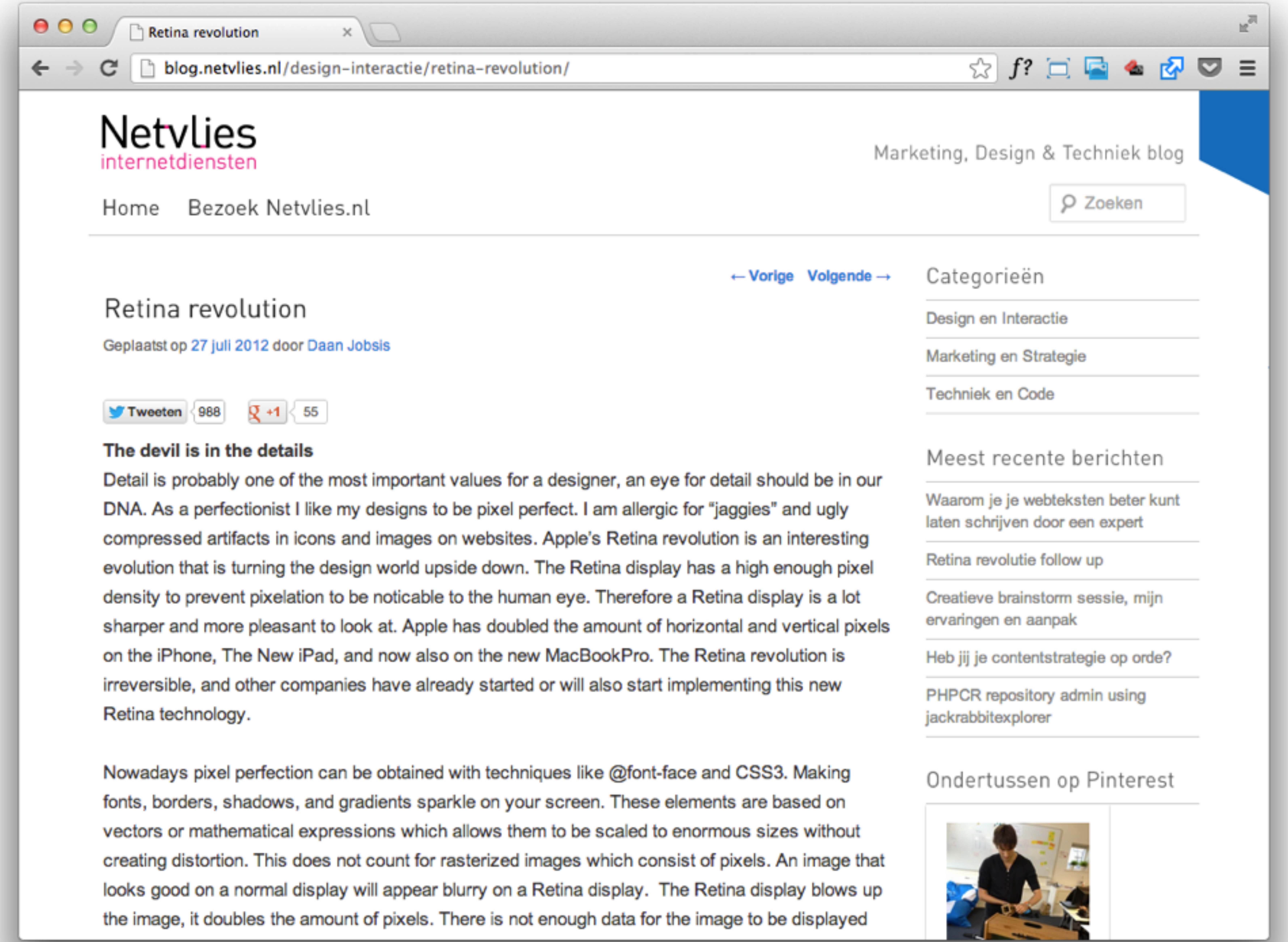


600×400px file **(7 Kb)**

0% JPEG quality
displayed in **300×200**


Google Translate
http://blog.netvlies.nl/design-interactie/retina-revolution/
f?

Device	PPI	Tested	Working	Browsers
Apple iPad 3	264	Yes	Yes	Safari, Chrome
Apple iPhone 4 / 4S	326	Yes	Yes	Safari, Chrome
Apple MacBook Pro 15 "Retina Display	220	Yes	Yes	Safari
Archos 10.1 G9	149	Yes	Yes	
HTC ChaCha	222	Yes	Yes	
HTC Desire S	252	Yes	Yes	
HTC One V	252	Yes	Yes	
Nokia Lumia 800	252	Yes	Yes	Mobile Internet Explorer 9
Nokia Lumia 900	217	Yes	**No**	Mobile Internet Explorer 9
Samsung Galaxy Ace	164	Yes	Yes	
Samsung Galaxy Nexus	316	Yes	Yes	
Samsung Galaxy S Advance	233	Yes	Yes	
Samsung Galaxy SIII	306	Yes	Yes	
Samsung Galaxy Tab 2 10.1	149	Yes	Yes	
Samsung Galaxy Xcover	158	Yes	Yes	
Samsung Note	285	Yes	Yes	



Categorieën

Design en Interactie

Marketing en Strategie

Techniek en Code

Meest recente berichten

Waarom je je webteksten beter kunt laten schrijven door een expert

Retina revolutie follow up

Creatieve brainstorm sessie, mijn ervaringen en aanpak

Heb jij je contentstrategie op orde?

PHPCR repository admin using jackrabbitexplorer

Ondertussen op Pinterest

Retina revolution

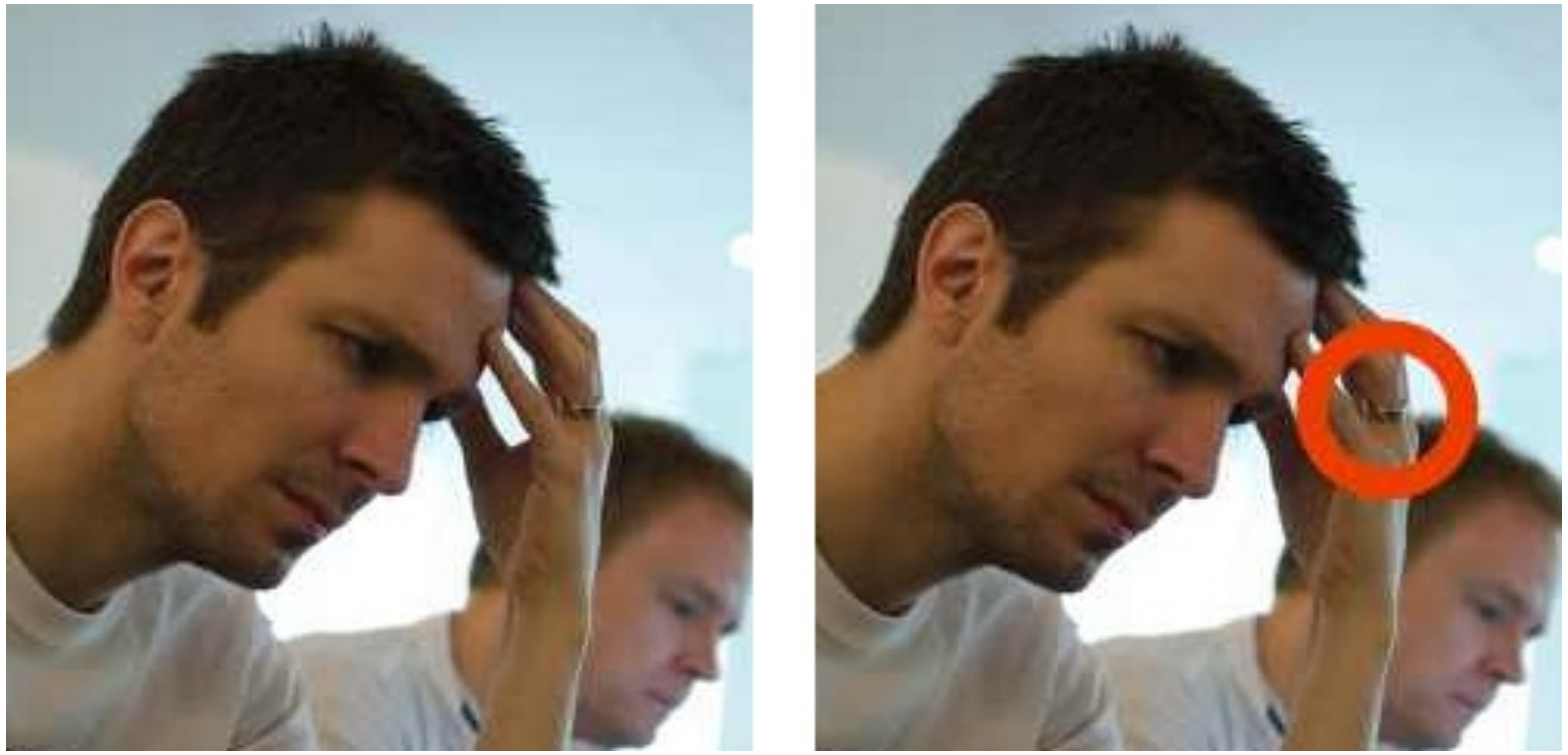
Geplaatst op 27 juli 2012 door Daan Jobsis

The devil is in the details
Detail is probably one of the most important values for a designer, an eye for detail should be in our DNA. As a perfectionist I like my designs to be pixel perfect. I am allergic for "jaggies" and ugly compressed artifacts in icons and images on websites. Apple's Retina revolution is an interesting evolution that is turning the design world upside down. The Retina display has a high enough pixel density to prevent pixelation to be noticable to the human eye. Therefore a Retina display is a lot sharper and more pleasant to look at. Apple has doubled the amount of horizontal and vertical pixels on the iPhone, The New iPad, and now also on the new MacBookPro. The Retina revolution is irreversible, and other companies have already started or will also start implementing this new Retina technology.

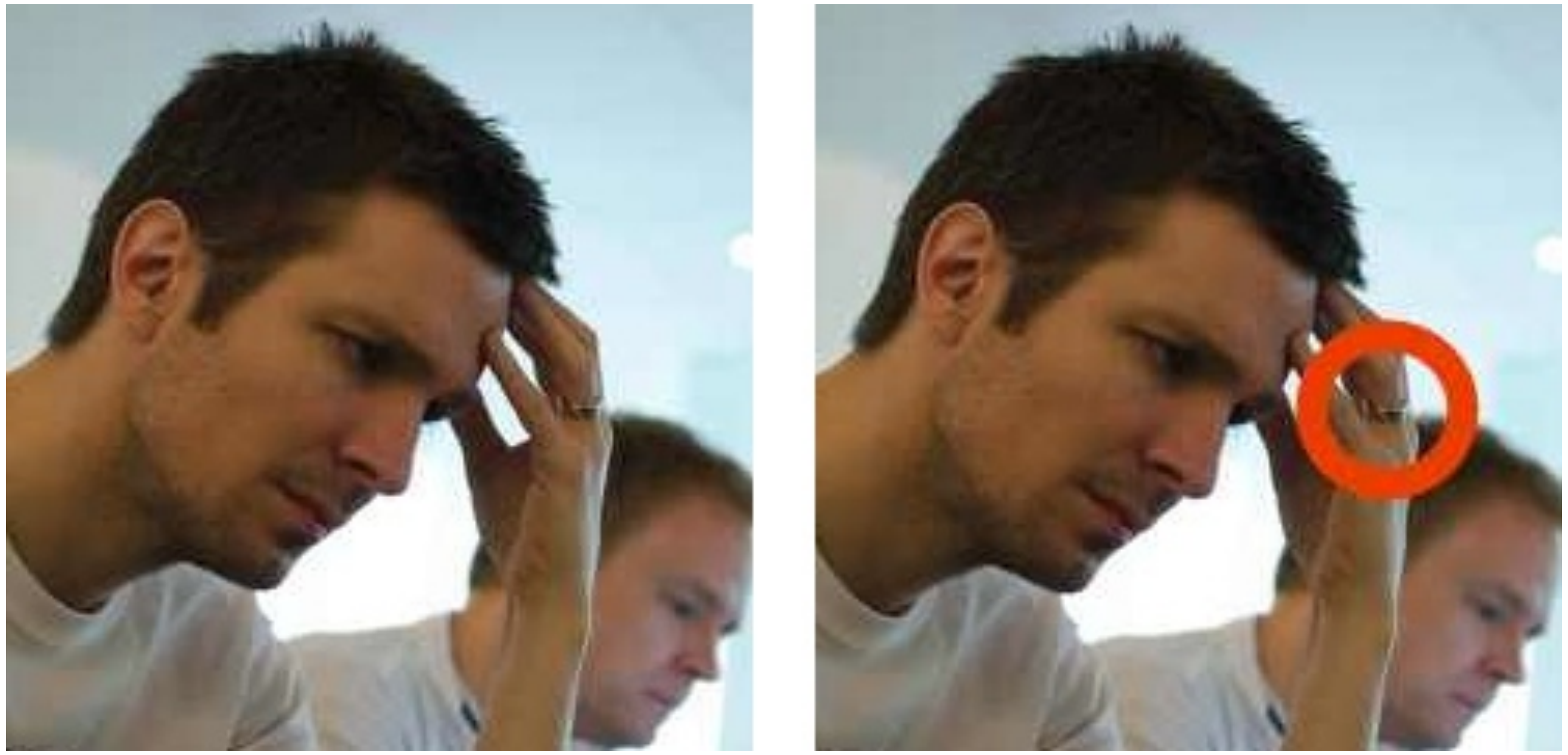
Nowadays pixel perfection can be obtained with techniques like @font-face and CSS3. Making fonts, borders, shadows, and gradients sparkle on your screen. These elements are based on vectors or mathematical expressions which allows them to be scaled to enormous sizes without creating distortion. This does not count for rasterized images which consist of pixels. An image that looks good on a normal display will appear blurry on a Retina display. The Retina display blows up the image, it doubles the amount of pixels. There is not enough data for the image to be displayed

Aftonbladet’s Images Strategy

- Design specification defined *main requirements:*
  - Optimization of the mobile version,
  - The pages should be easy to cache,
  - A single HTML file to be served to all users,
  - All images on a content delivery network (CDN),
  - No complexity in the image-serving logic,
  - Serving different image versions to different devices.
- *Solution:* Loading images with JavaScript after HTML and CSS have fully loaded.



- *30% JPEG quality*: bright-red areas don’t compress well.
- Editors *can select* compression rates, but aggressive compression is a default.



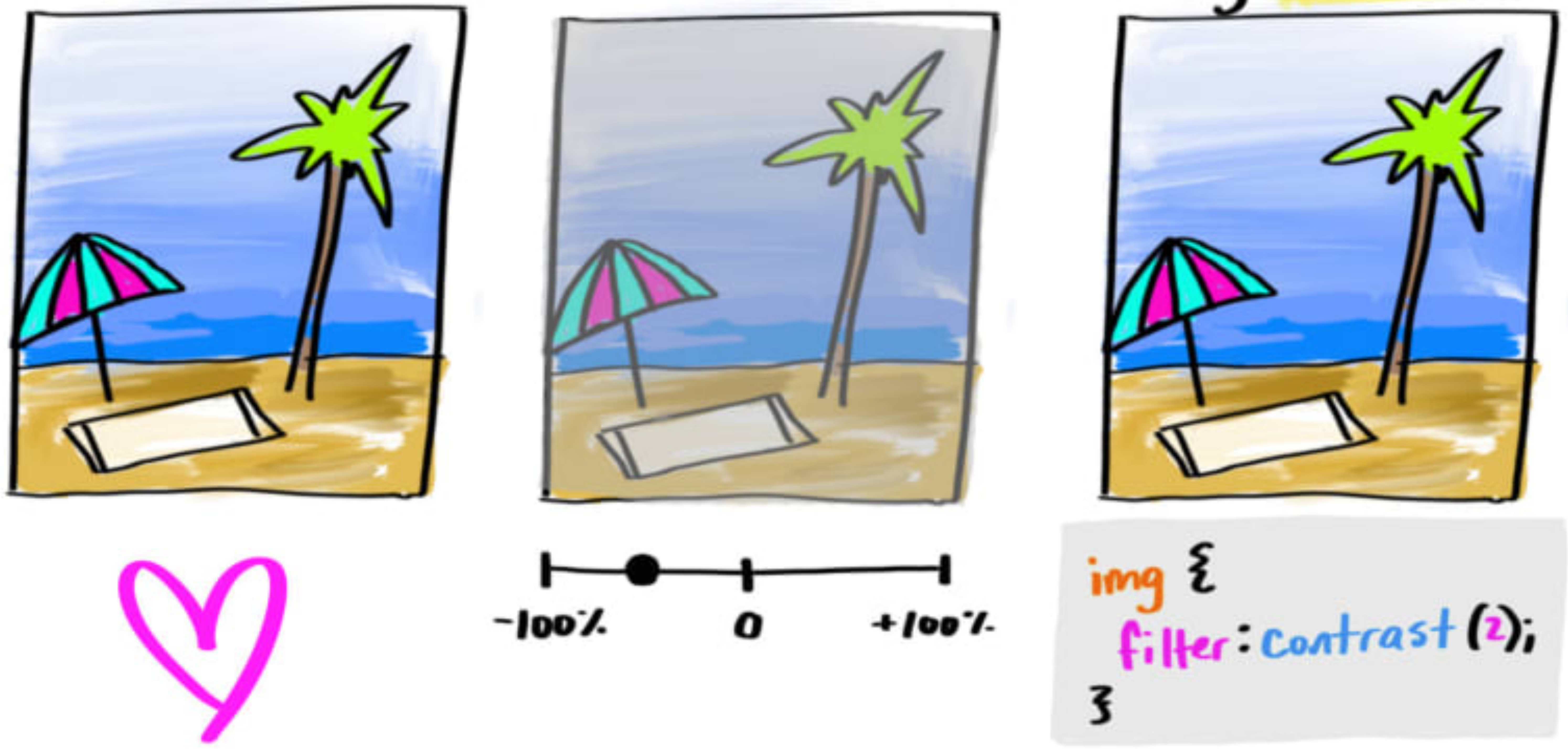
- The homepage on a mobile device has *40 images.*
- On average, the “large” screen has *650 Kb*, “medium” — 570 Kb, “small” — 450 Kb.

the Contrast Swap Technique

Start with your image

Remove image contrast

Reapply contrast using **CSS filters**



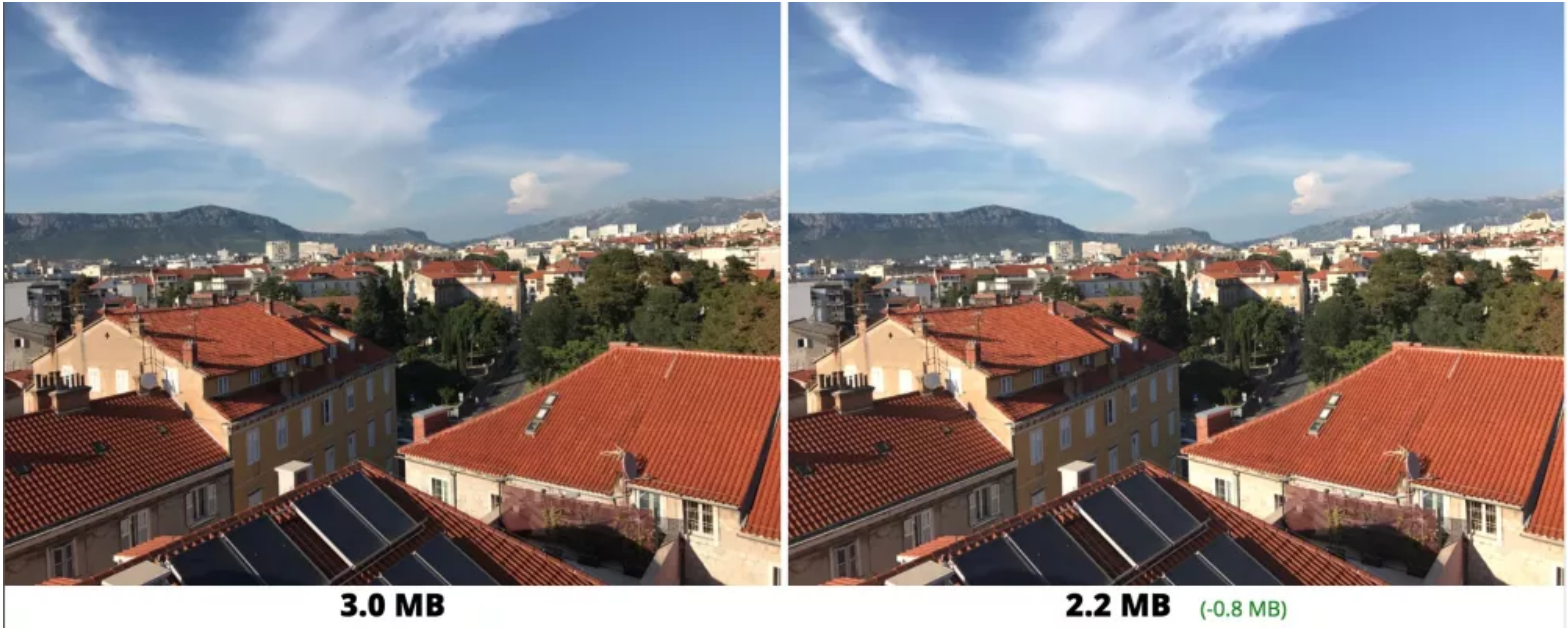
```
img {
  filter: contrast(2);
}
```

3.2 MB

2.3 MB (-0.9 MB)

2.2 MB

1.7 MB (-0.5 MB)



3.0 MB
2.2 MB
(-0.8 MB)

1.9 MB

1.4 MB (-0.5 MB)

3.2 MB

2.3 MB

3.0 MB

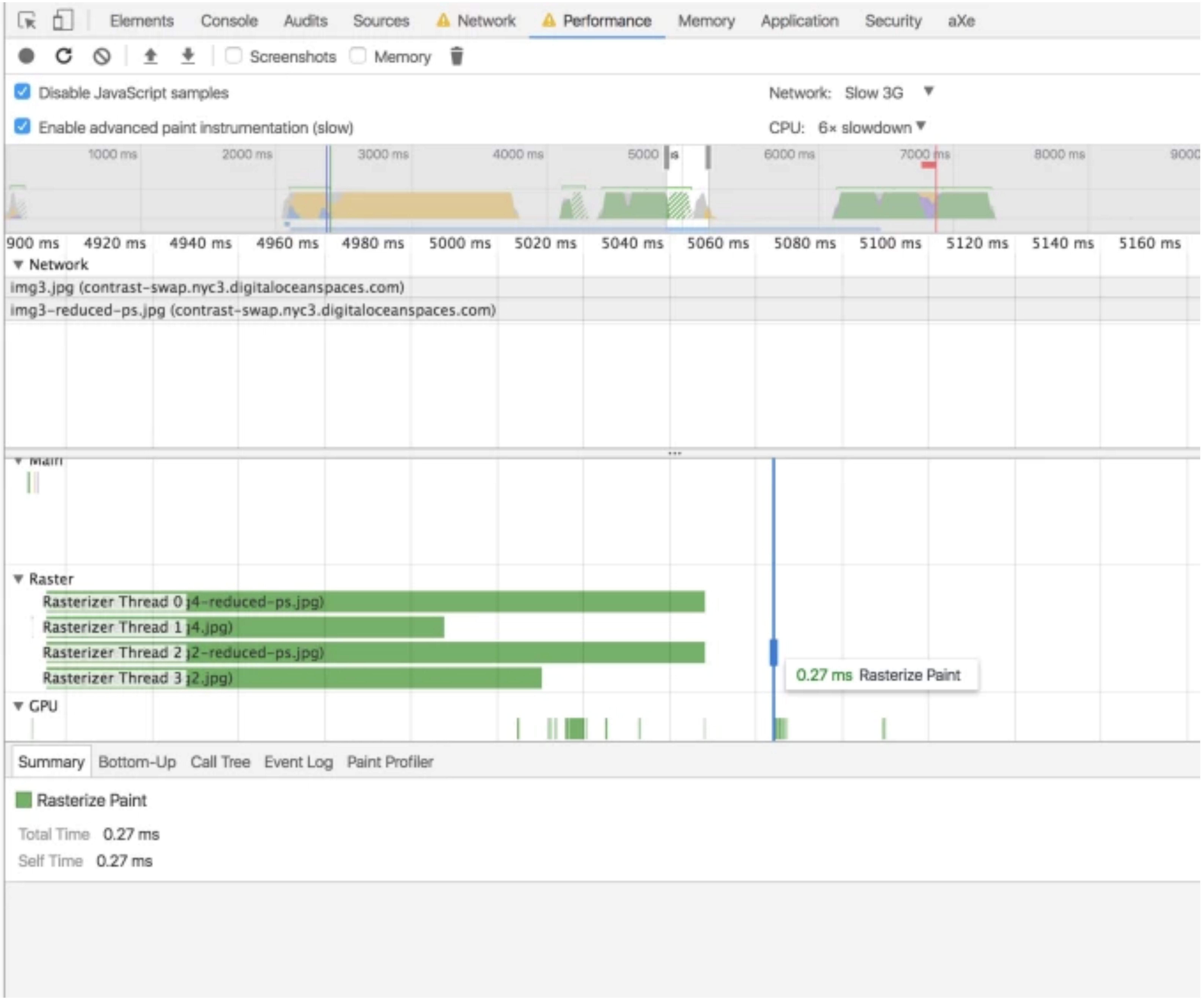
2.2 MB

2.2 MB

1.7 MB

1.9 MB

1.4 MB



☑ Disable JavaScript samples

☑ Enable advanced paint instrumentation (slow)

Network: Slow 3G ▼

CPU: 6× slowdown ▼

▼ Network

img3.jpg (contrast-swap.nyc3.digitaloceanspaces.com)

img3-reduced-ps.jpg (contrast-swap.nyc3.digitaloceanspaces.com)

▼ Raster

Rasterizer Thread 0 j4-reduced-ps.jpg)

Rasterizer Thread 1 j4.jpg)

Rasterizer Thread 2 j2-reduced-ps.jpg)

Rasterizer Thread 3 j2.jpg)

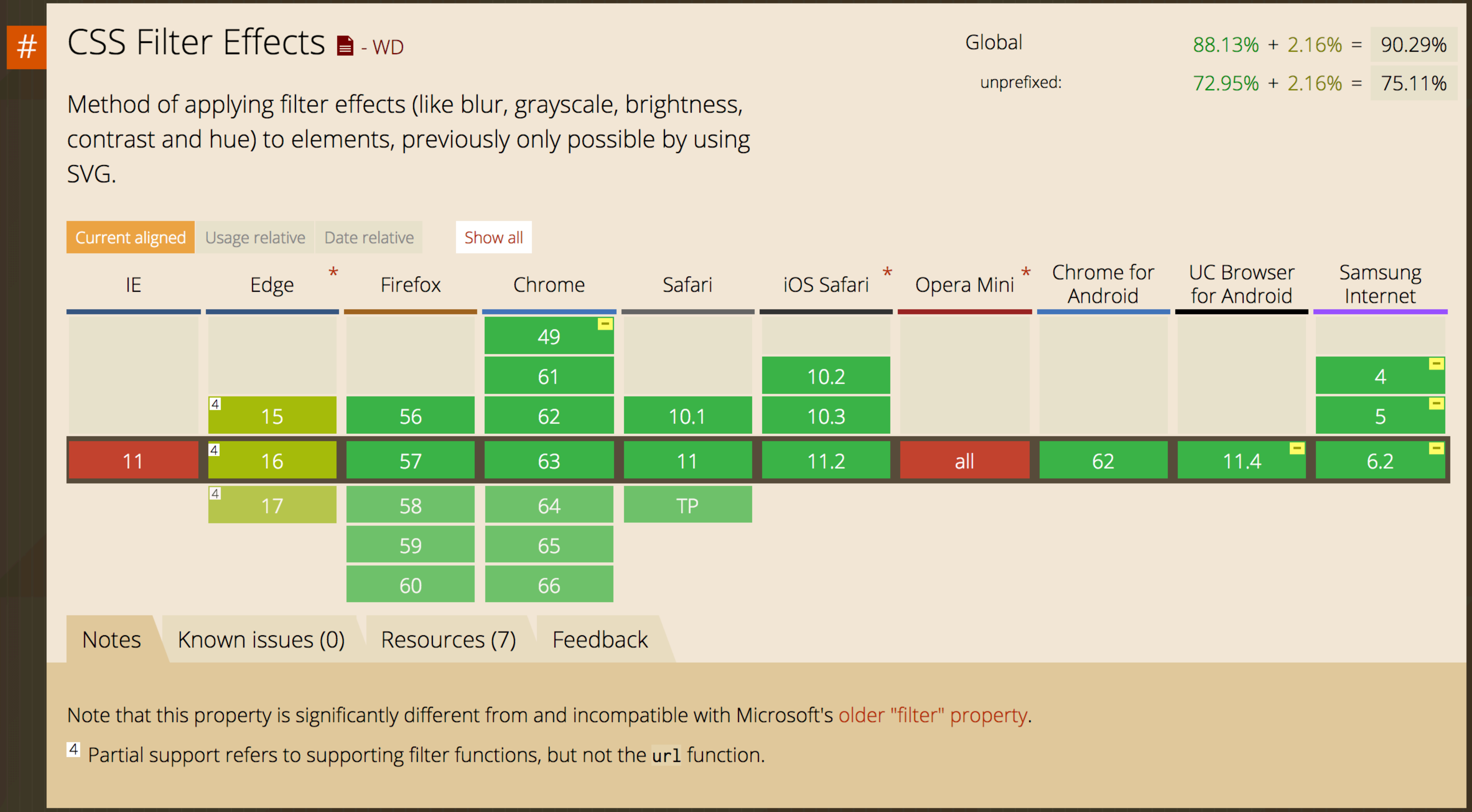
0.27 ms Rasterize Paint

▼ GPU

🟩 Rasterize Paint

Total Time 0.27 ms

Self Time 0.27 ms



CSS Filter Effects - WD

Global 88.13% + 2.16% = 90.29%

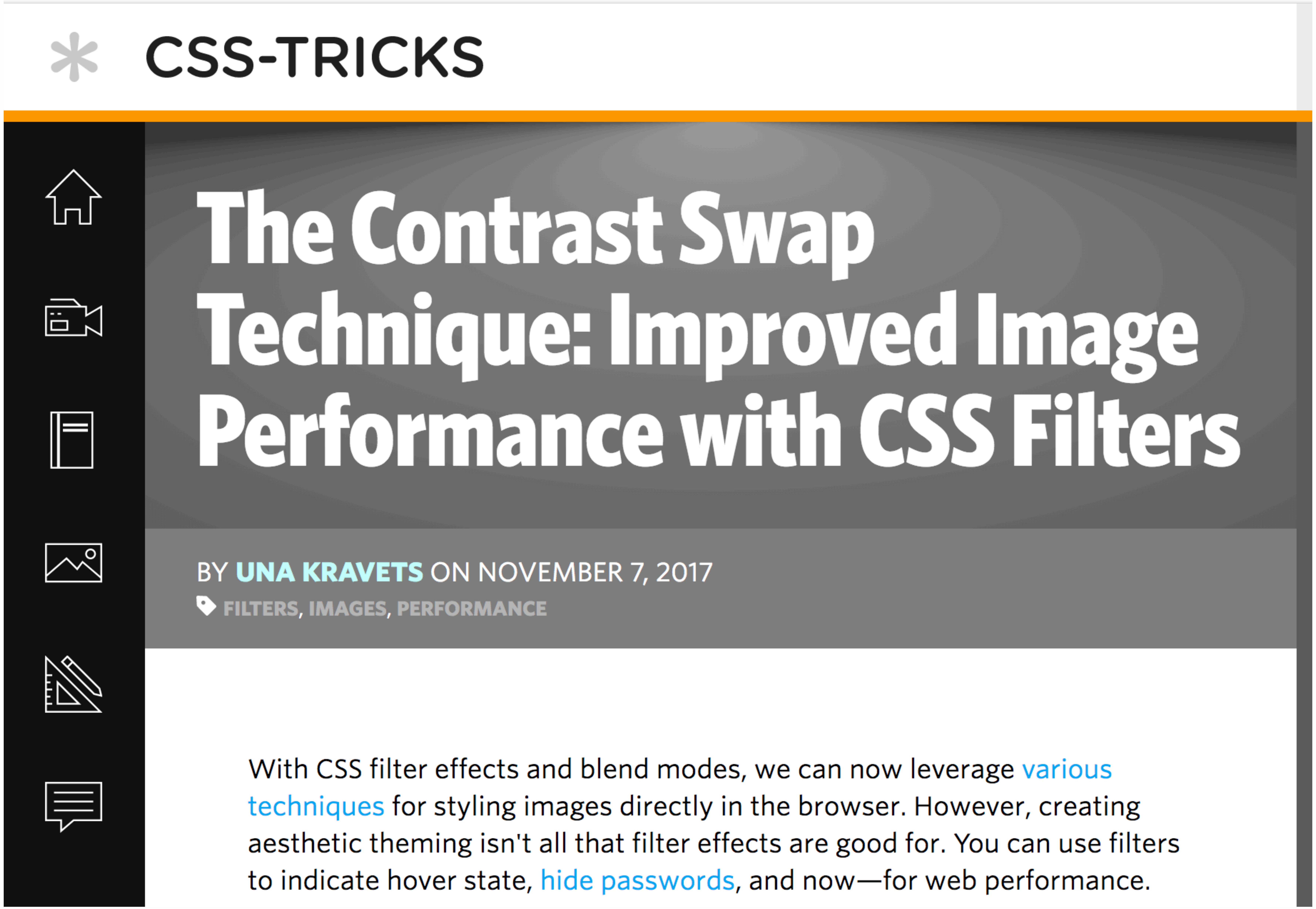
unprefixed: 72.95% + 2.16% = 75.11%

Method of applying filter effects (like blur, grayscale, brightness, contrast and hue) to elements, previously only possible by using SVG.

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49 −						
			61		10.2				4 −
	[4] 15	56	62	10.1	10.3				5 −
11	[4] 16	57	63	11	11.2	all	62	11.4 −	6.2 −
	[4] 17	58	64	TP					
		59	65						
		60	66						

Note that this property is significantly different from and incompatible with Microsoft's older "filter" property.

[4] Partial support refers to supporting filter functions, but not the `url` function.



CSS-TRICKS

The Contrast Swap Technique: Improved Image Performance with CSS Filters

BY UNA KRAVETS ON NOVEMBER 7, 2017

FILTERS, IMAGES, PERFORMANCE

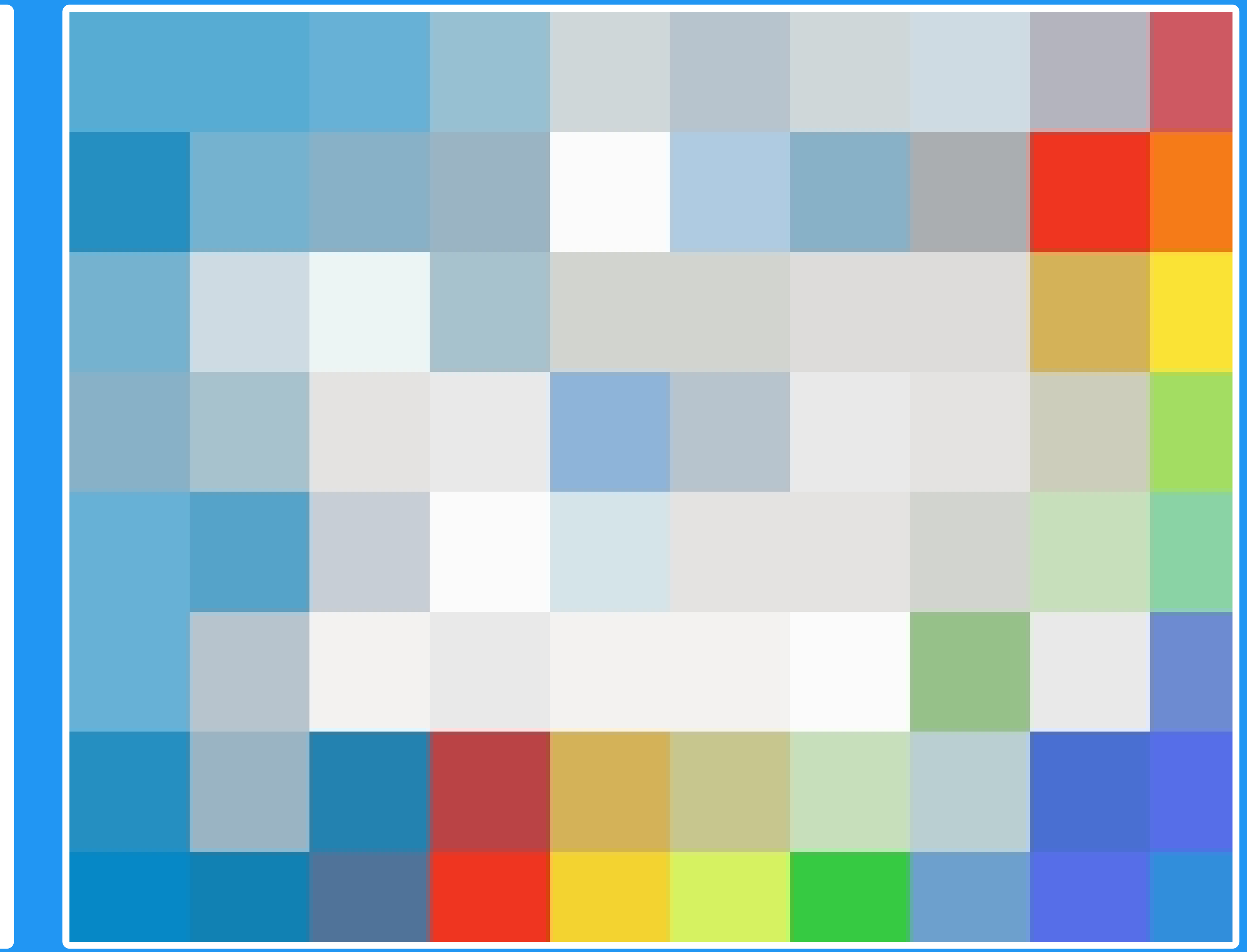
With CSS filter effects and blend modes, we can now leverage various techniques for styling images directly in the browser. However, creating aesthetic theming isn't all that filter effects are good for. You can use filters to indicate hover state, hide passwords, and now—for web performance.

- The original photo has 1600px width, *971 Kb.*
  Quality *60* brings the size down to *213 Kb.*

- Blurring unimportant parts of the photo brings the size down to *147 Kb.*

Sequential JPEG

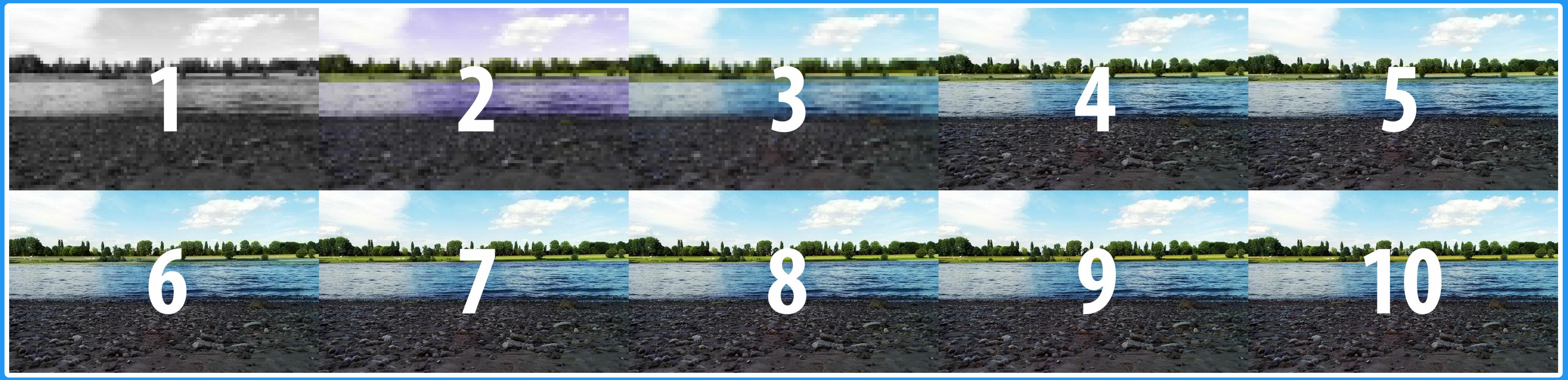
Progressive JPEG



Images taken from http://www.pixelstech.net/article/1374757887-Use-progressive-JPEG-to-improve-user-experience

Scans

Default Scan Levels



Thanks to Frédéric Kayser for creating 'jsk': http://encode.ru/threads/1800-JSK-JPEG-Scan-Killer-progressive-JPEG-explained-in-slowmo

```
# Initial DC scan for Y,Cb,Cr (lowest bit not sent)
0,1,2: 0-0,   0, 1 ;

# First AC scan: send first 5 Y AC coefficients, minus 2 lowest bits:
0:     1-5,   0, 2 ;

# Send all Cr,Cb AC coefficients, minus lowest bit:
# (chroma data is usually too small to be worth subdividing further;
#  but note we send Cr first since eye is least sensitive to Cb)
2:     1-63,  0, 0 ;
1:     1-63,  0, 0 ;

# Send remaining Y AC coefficients, minus 2 lowest bits:
0:     6-63,  0, 2 ;
# Send next-to-lowest bit of all Y AC coefficients:
0:     1-63,  2, 1 ;

# At this point we've sent all but the lowest bit of all coefficients.
# Send lowest bit of DC coefficients
0,1,2: 0-0,   1, 0 ;

# Send lowest bit of AC coefficients
2:     1-63,  1, 0 ;
1:     1-63,  1, 0 ;

# Y AC lowest bit scan is last; it's usually the largest scan
0:     1-63,  1, 0 ;
```

```
# Initial DC scan for Y,Cb,Cr (lowest bit not sent)
0,1,2: 0-0,   0, 1 ;

# First AC scan: send first 5 Y AC coefficients, minus 2 lowest bits:
0:     1-5,   0, 2 ;

# Send all Cr,Cb AC coefficients, minus lowest bit:
# (chroma data is usually too small to be worth subdividing further;
#  but note we send Cr first since eye is least sensitive to Cb)
2:     1-63,  0, 0 ;
1:     1-63,  0, 0 ;

# Send remaining Y AC coefficients, minus 2 lowest bits:
0:     6-63,  0, 2 ;
# Send next-to-lowest bit of all Y AC coefficients:
0:     1-63,  2, 1 ;

# At this point we've sent all but the lowest bit of all coefficients.
# Send lowest bit of DC coefficients
0,1,2: 0-0,   1, 0 ;

# Send lowest bit of AC coefficients
2:     1-63,  1, 0 ;
1:     1-63,  1, 0 ;

# Y AC lowest bit scan is last; it's usually the largest scan
0:     1-63,  1, 0 ;
```

Y (brightness / luminance)

Cr & Cb (red & blue / chrominance)

0-63: Matrix index

1st Scan Layer Has Small Byte Size

Ships Fast

&

Shows Soon

```
# Interleaved DC scan for Y,Cb,Cr:
0,1,2: 0-0, 0, 1 ;

# AC scans:
0: 1-27,  0, 0 ;
2: 1-63,    0, 0 ;
1: 1-63,  0, 0 ;

# Remaining Y coefficients
0: 28-63, 0, 0 ;
```

1 initial DC for All channels

2 Half of all brightness values

3/4 All remaining color channel values

5 2nd half of brightness channel

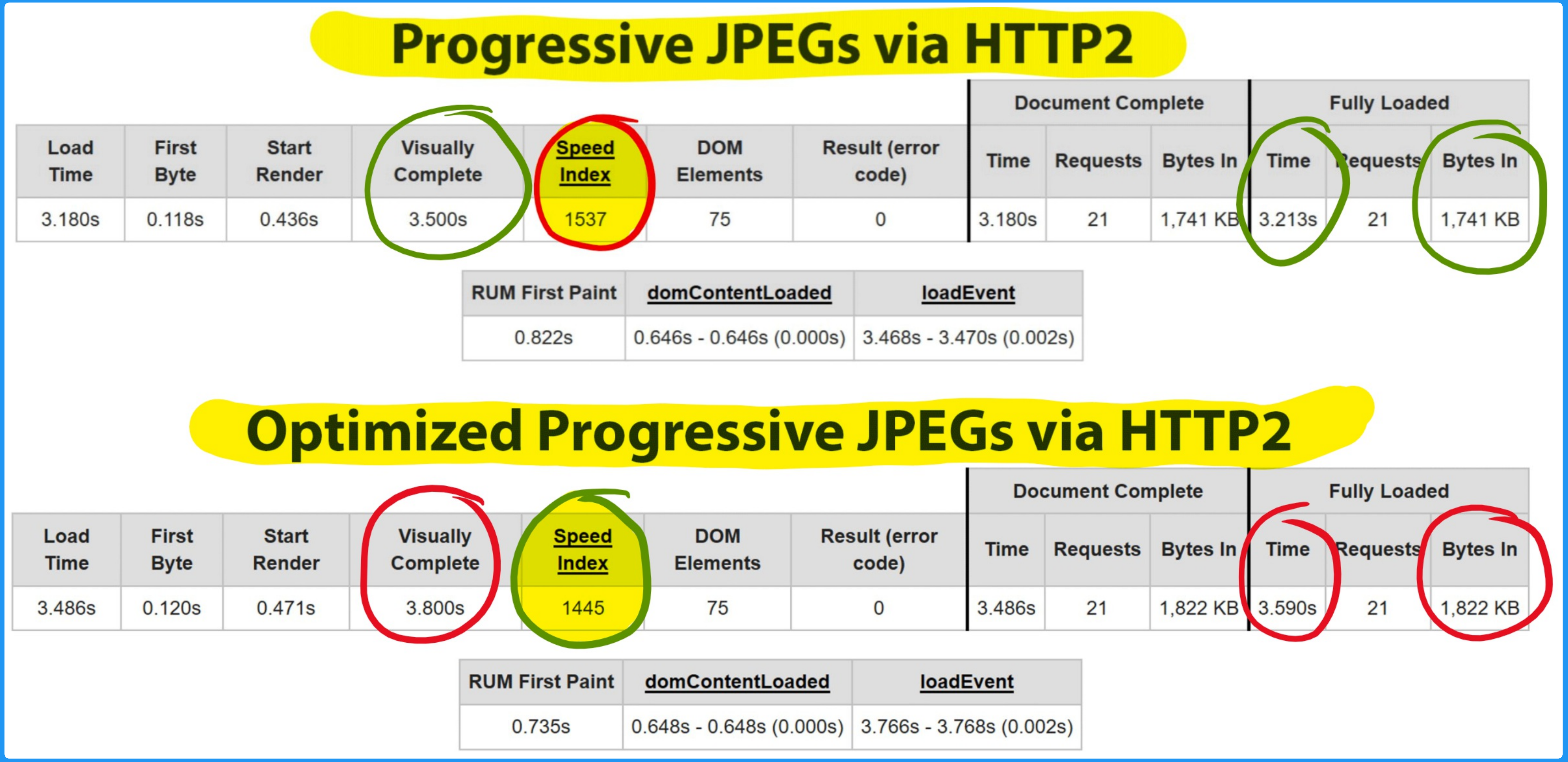
1

2

3

4

5



Progressive JPEGs via HTTP2

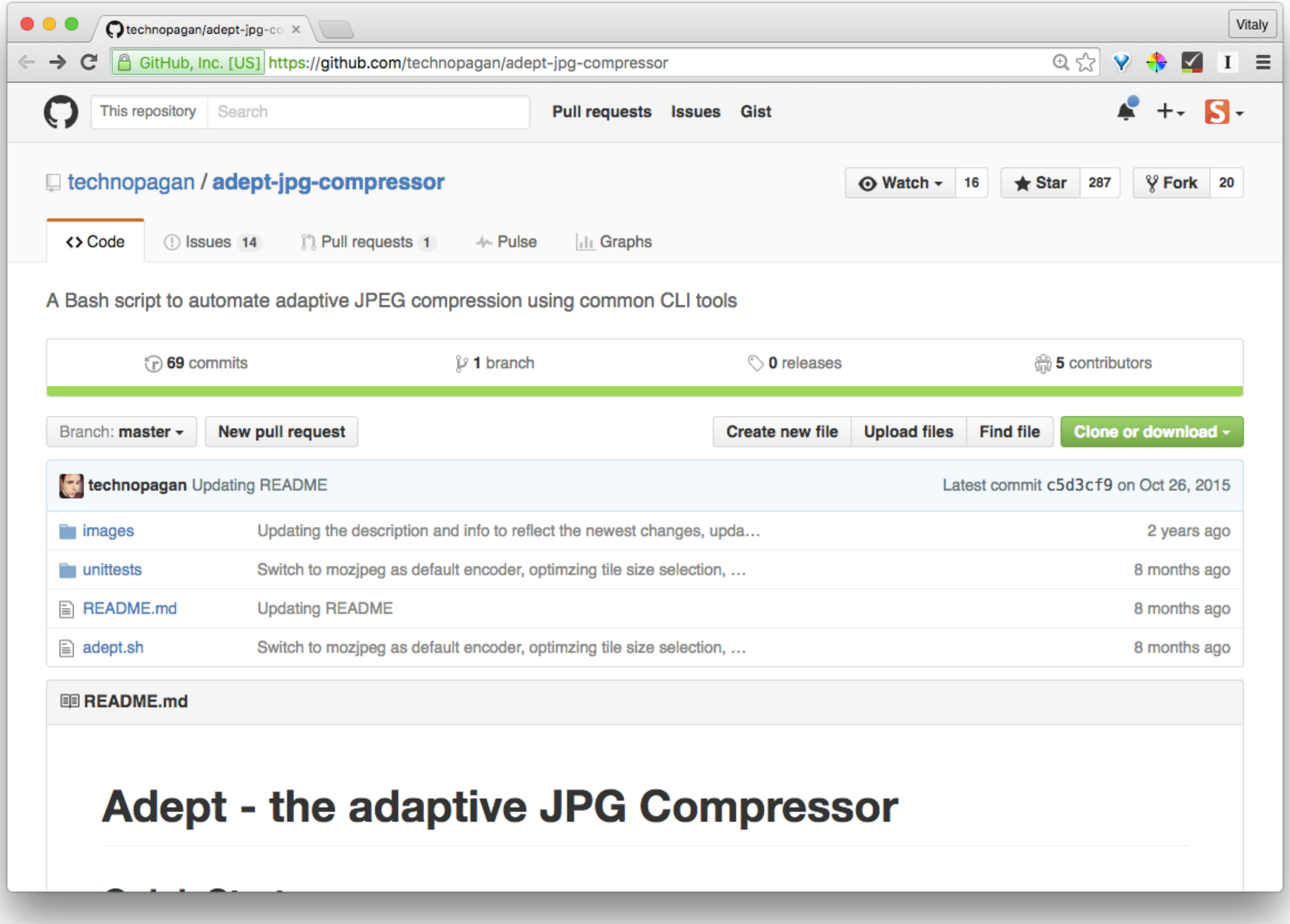
							Document Complete			Fully Loaded		
Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
3.180s	0.118s	0.436s	3.500s	1537	75	0	3.180s	21	1,741 KB	3.213s	21	1,741 KB

RUM First Paint	domContentLoaded	loadEvent
0.822s	0.646s - 0.646s (0.000s)	3.468s - 3.470s (0.002s)

Optimized Progressive JPEGs via HTTP2

							Document Complete			Fully Loaded		
Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
3.486s	0.120s	0.471s	3.800s	1445	75	0	3.486s	21	1,822 KB	3.590s	21	1,822 KB

RUM First Paint	domContentLoaded	loadEvent
0.735s	0.648s - 0.648s (0.000s)	3.766s - 3.768s (0.002s)



technopagan / adept-jpg-compressor

A Bash script to automate adaptive JPEG compression using common CLI tools

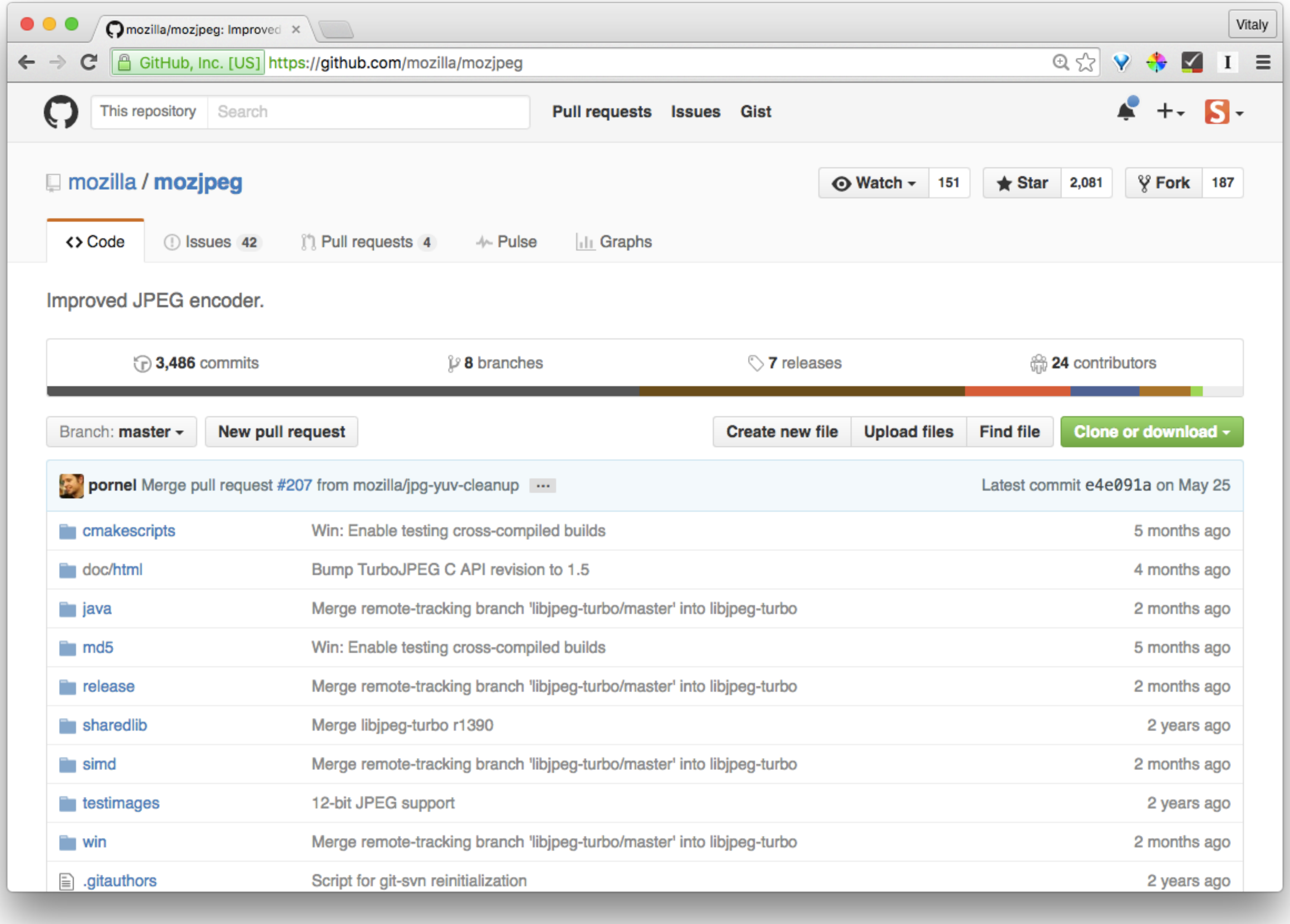
69 commits · 1 branch · 0 releases · 5 contributors

technopagan Updating README — Latest commit c5d3cf9 on Oct 26, 2015

images	Updating the description and info to reflect the newest changes, upda…	2 years ago
unittests	Switch to mozjpeg as default encoder, optimzing tile size selection, …	8 months ago
README.md	Updating README	8 months ago
adept.sh	Switch to mozjpeg as default encoder, optimzing tile size selection, …	8 months ago

README.md

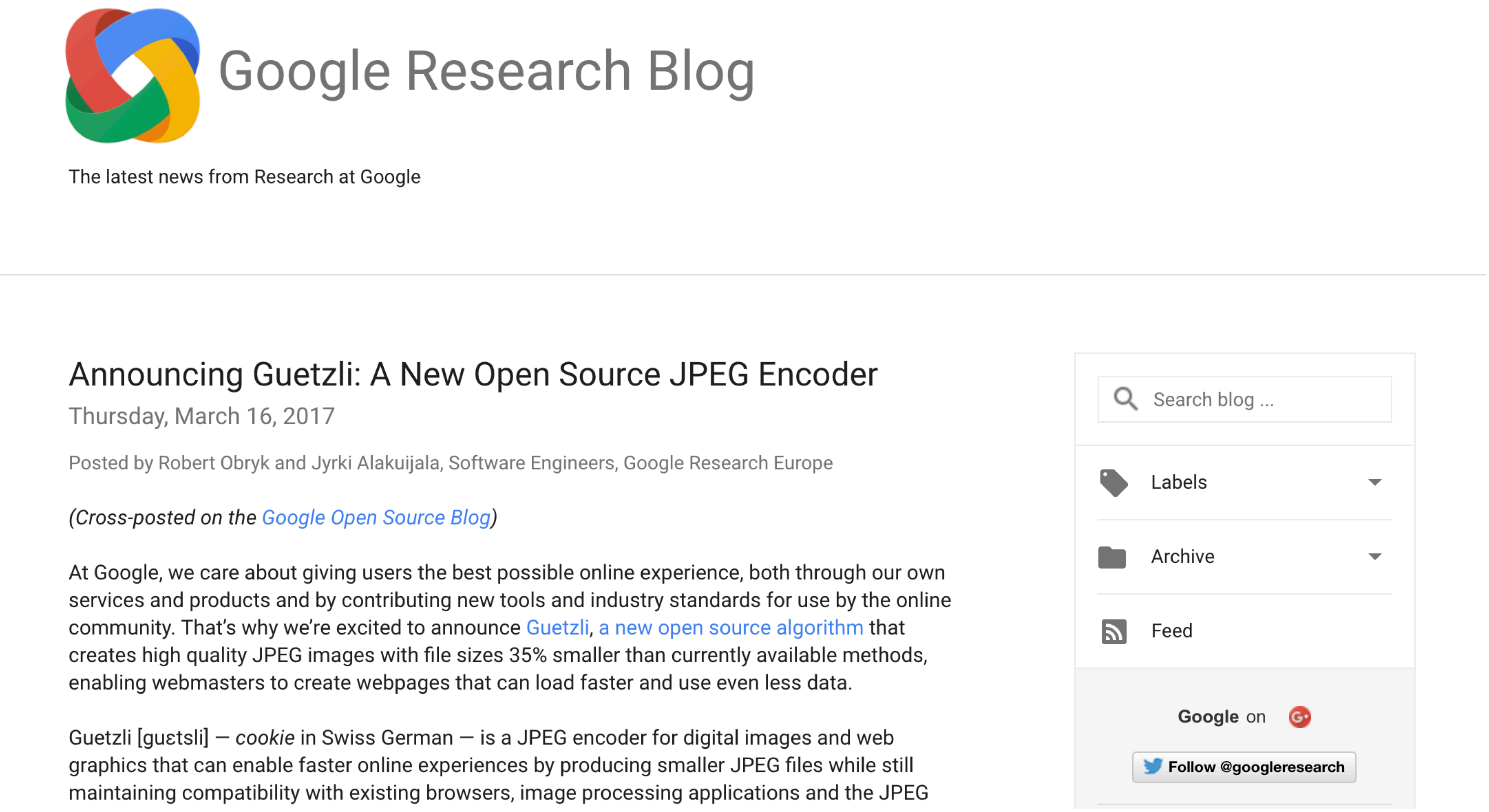
Adept - the adaptive JPG Compressor



Improved JPEG encoder.

pornel Merge pull request #207 from mozilla/jpg-yuv-cleanup … Latest commit e4e091a on May 25

cmakescripts	Win: Enable testing cross-compiled builds	5 months ago
doc/html	Bump TurboJPEG C API revision to 1.5	4 months ago
java	Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo	2 months ago
md5	Win: Enable testing cross-compiled builds	5 months ago
release	Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo	2 months ago
sharedlib	Merge libjpeg-turbo r1390	2 years ago
simd	Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo	2 months ago
testimages	12-bit JPEG support	2 years ago
win	Merge remote-tracking branch 'libjpeg-turbo/master' into libjpeg-turbo	2 months ago
.gitauthors	Script for git-svn reinitialization	2 years ago



Google Research Blog

The latest news from Research at Google

Announcing Guetzli: A New Open Source JPEG Encoder

Thursday, March 16, 2017

Posted by Robert Obryk and Jyrki Alakuijala, Software Engineers, Google Research Europe

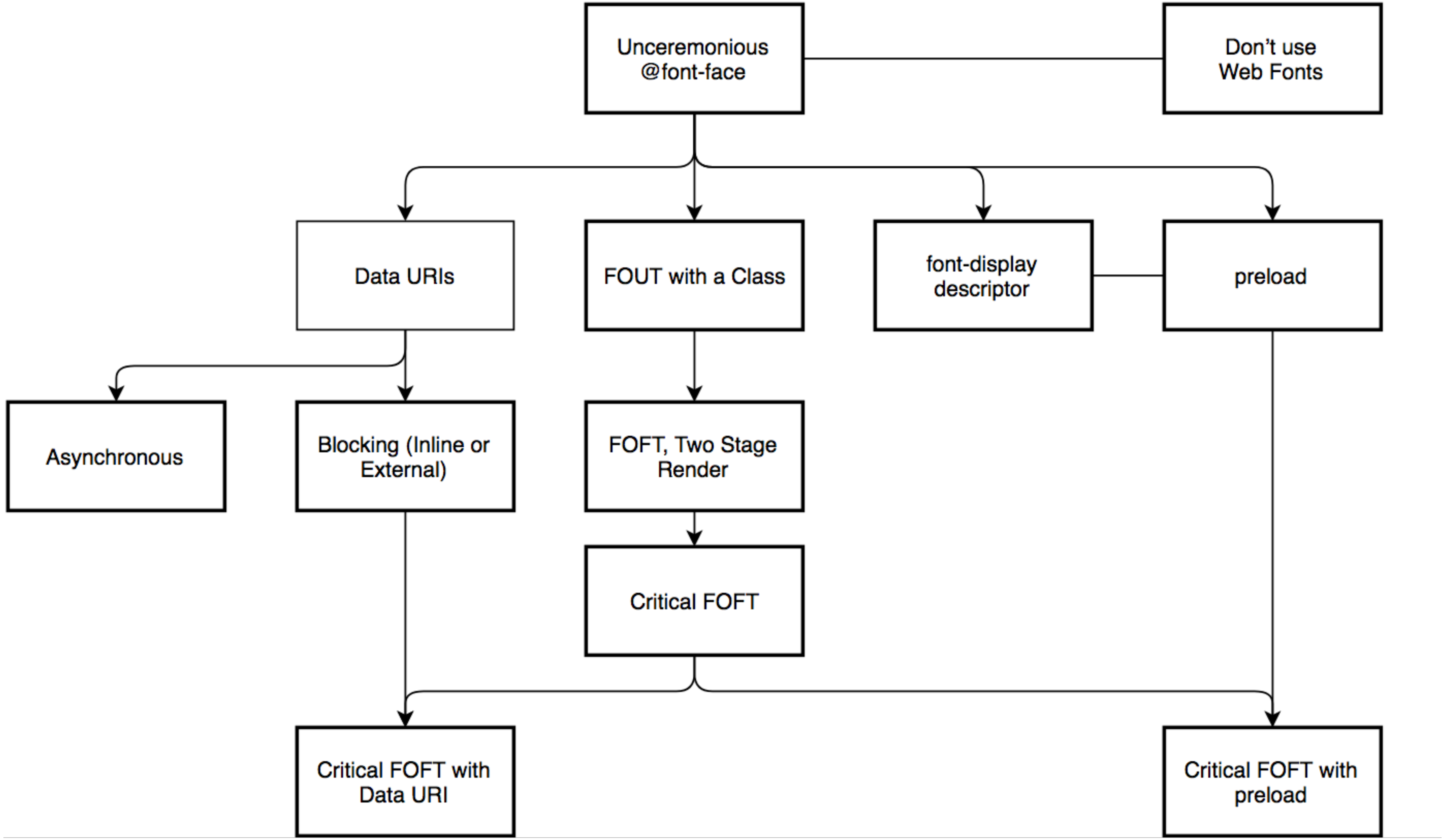
(Cross-posted on the Google Open Source Blog)

At Google, we care about giving users the best possible online experience, both through our own services and products and by contributing new tools and industry standards for use by the online community. That’s why we’re excited to announce Guetzli, a new open source algorithm that creates high quality JPEG images with file sizes 35% smaller than currently available methods, enabling webmasters to create webpages that can load faster and use even less data.

Guetzli [guɛtsli] — *cookie* in Swiss German — is a JPEG encoder for digital images and web graphics that can enable faster online experiences by producing smaller JPEG files while still maintaining compatibility with existing browsers, image processing applications and the JPEG

LEVEL3 WEBFONTS

We want nice type, but **performance** matters, too. You either rely on Typekit/ Google Fonts or self-host the fonts. What is your strategy for *loading web fonts?*


Unceremonious @font-face
Don’t use Web Fonts
Data URIs
FOUT with a Class
font-display descriptor
preload
Asynchronous
Blocking (Inline or External)
FOFT, Two Stage Render
Critical FOFT
Critical FOFT with Data URI
Critical FOFT with preload

Declaring @font-face

- We can use bulletproof @font-face syntax to avoid common traps along the way:

- CSS:

```css
@font-face {
    font-family: 'Elena Regular';
    src: url('elena.eot?#iefix') format('embedded-opentype'),
         url('elena.woff2') format('woff2'),
         url('elena.woff') format('woff'),
         url('elena.otf') format('opentype');
}
```

Declaring @font-face

- If you want only smart browsers (IE9+) to download fonts, declaration can be shorter:

- CSS:

```
@font-face {
   font-family: 'Elena Regular';
   src: url('elena.woff2') format('woff2'),
        url('elena.woff') format('woff'),
        url('elena.otf') format('opentype');
}
```

- CSS:

```
@font-face {
   font-family: 'Elena Regular';
   src: url('elena.woff2') format('woff2'),
        url('elena.woff') format('woff'),
        url('elena.otf') format('opentype');
}
```

- When a font family name is used in CSS, browsers match it against all @font-face rules, download web fonts, display content.

- When a font family name is used in CSS, browsers match it against all @font-face rules, download web fonts, display content.

- CSS:

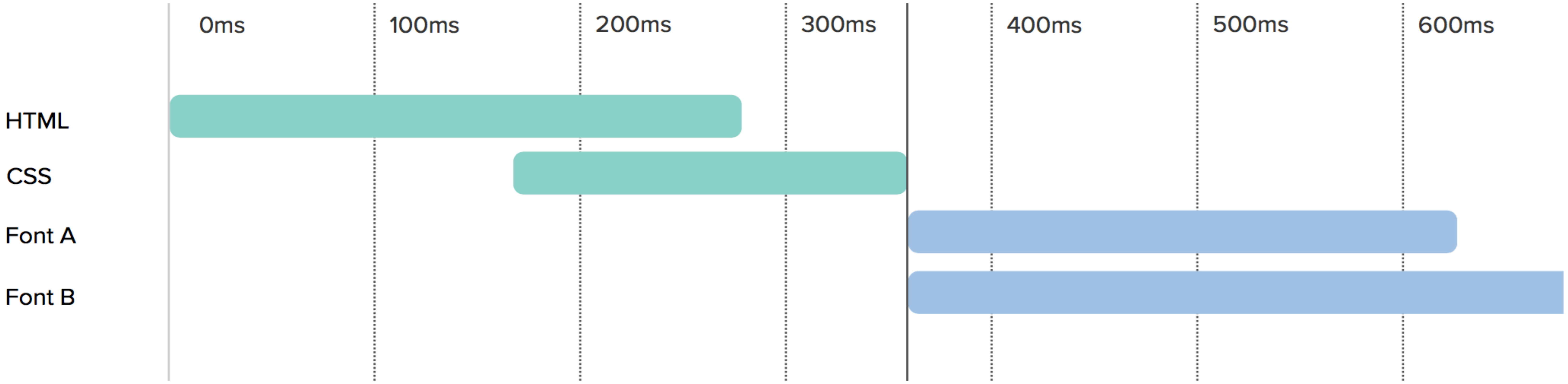
```
body {
    font-family: 'Elena Regular',
    AvenirNext, Avenir,                 /* iOS */
   'Roboto Slab', 'Droid Serif',        /* Android */
   'Segoe UI',                          /* Microsoft  */
    Georgia, 'Times New Roman', serif;  /* Fallback  */
}
```

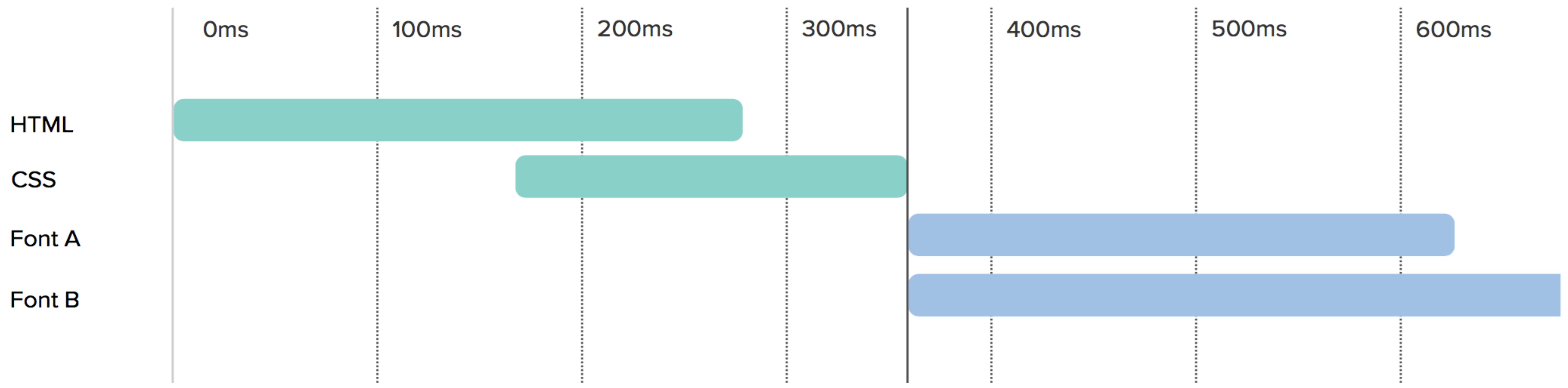
- CSS:

```
body {
   font-family: 'Elena Regular',
   AvenirNext, Avenir,                 /* iOS */
  'Roboto Slab', 'Droid Serif',        /* Android */
  'Segoe UI',                          /* Microsoft  */
   Georgia, 'Times New Roman', serif;  /* Fallback  */
}
```

- HTML:

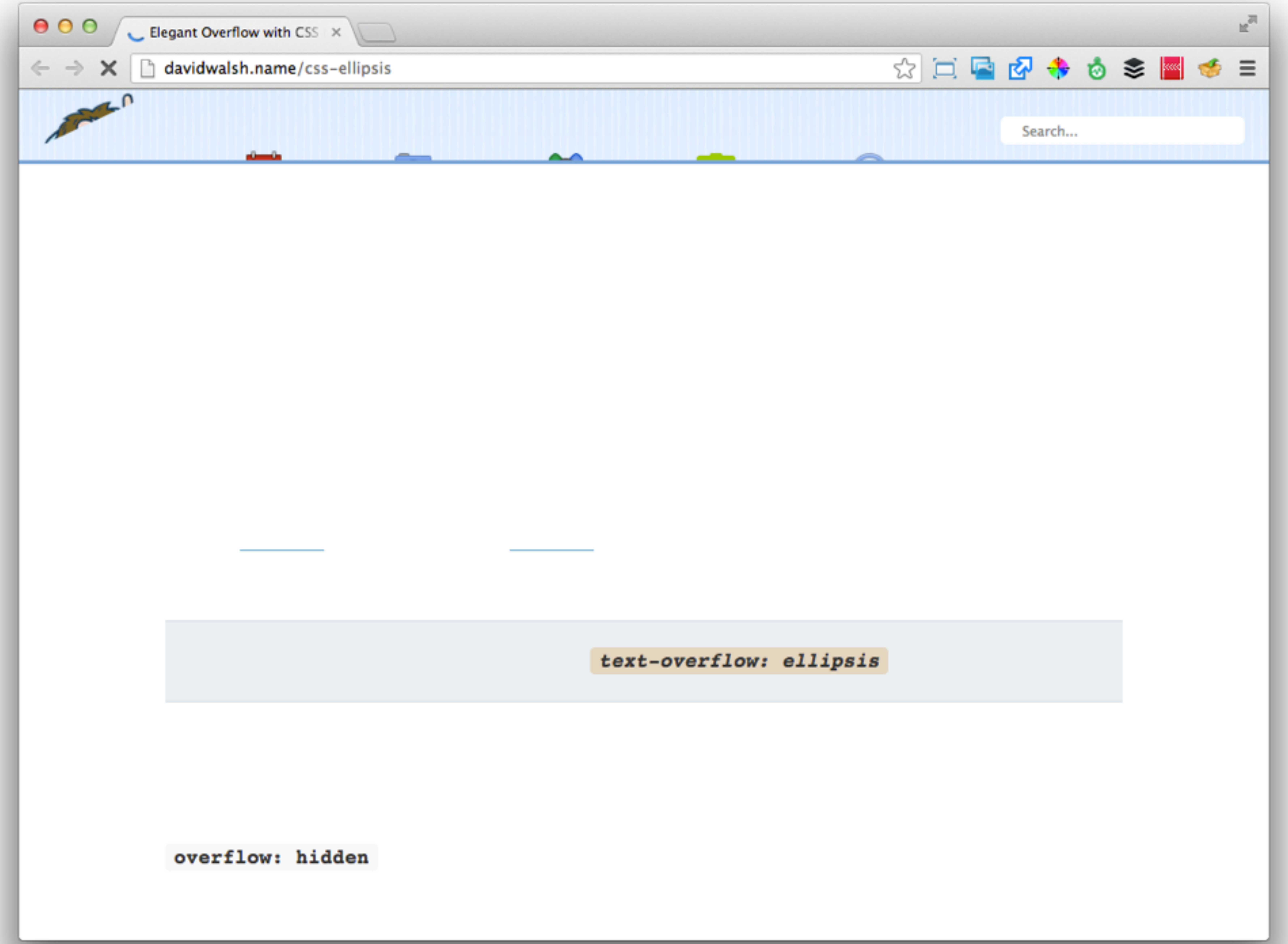
```
<link href='http://fonts.googleapis.com/css?family=Skolar_Reg'
rel='stylesheet' type='text/css'>

<script type="text/javascript"
    src="//use.typekit.net/tbb3uid.js"></script>
<script type="text/javascript">
    try{Typekit.load();}catch(e){}</script>
```


0ms
100ms
200ms
300ms
400ms
500ms
600ms
HTML
CSS
Font A
Font B



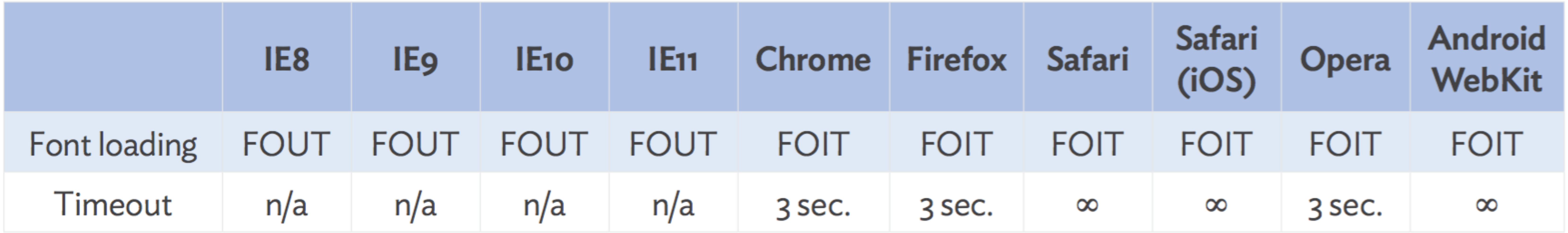
- Once DOM and CSSOM are constructed, if @font-face matches, a font will be required.

- If fonts aren’t cached yet, they will be requested, downloaded and applied, deferring rendering.



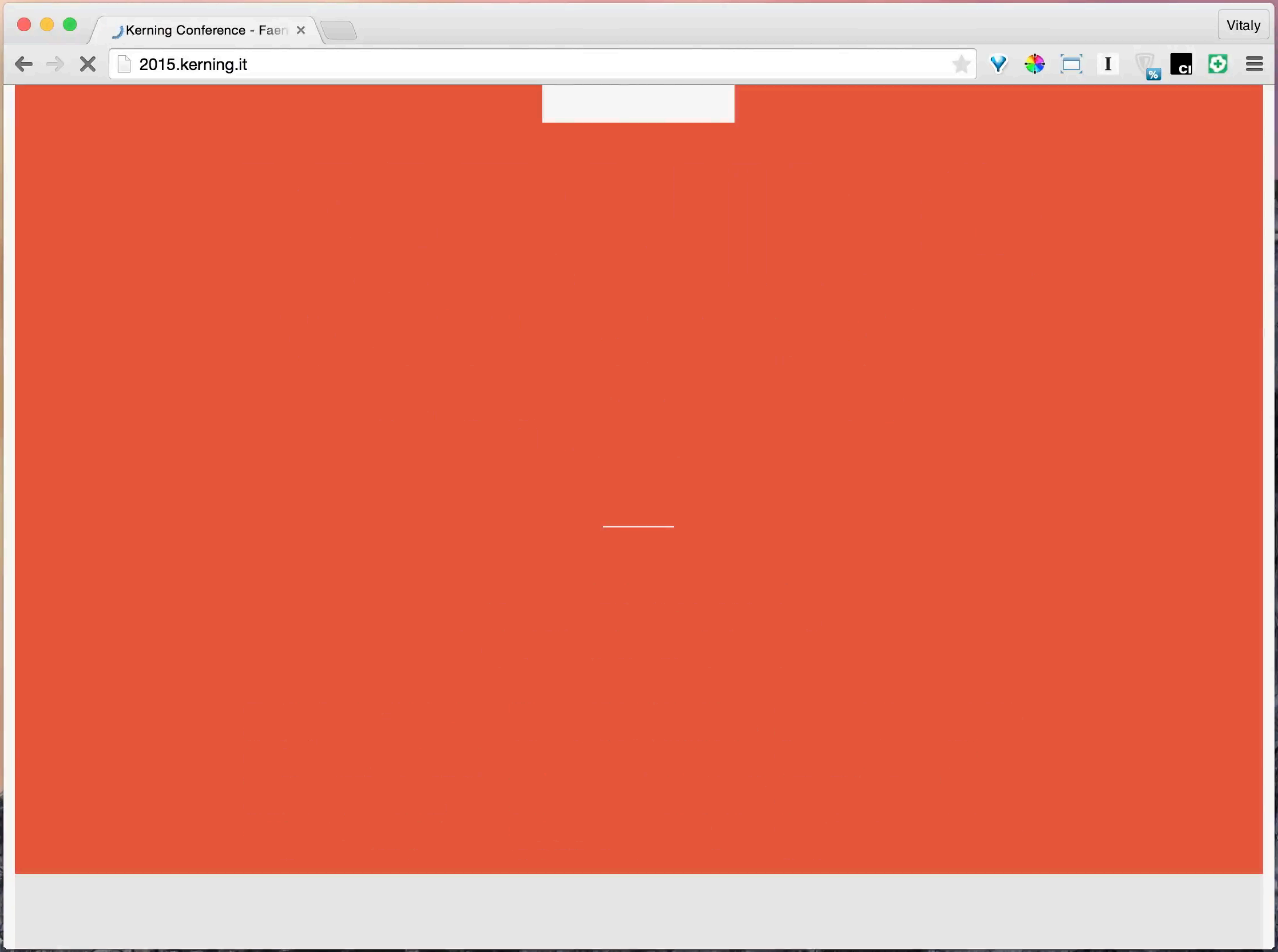
`text-overflow: ellipsis`

`overflow: hidden`



	IE8	IE9	IE10	IE11	Chrome	Firefox	Safari	Safari (iOS)	Opera	Android WebKit
Font loading	FOUT	FOUT	FOUT	FOUT	FOIT	FOIT	FOIT	FOIT	FOIT	FOIT
Timeout	n/a	n/a	n/a	n/a	3 sec.	3 sec.	∞	∞	3 sec.	∞

- FOIT *(Flash Of Invisible Text):* no content displayed until the font becomes available.
- FOUT *(Flash Of Unstyled Text)*: show content in fallback fonts first, then switch to web fonts.


Kerning Conference - Faen
Vitaly
2015.kerning.it

Async Data URI Stylesheet

- To eliminate FOIT, we display fallback right away, and load web fonts async with *loadCSS.*
  - Easy to group requests into a single repaint,
  - Has a noticeable short *FOIT* during parsing,
  - How to choose a format to load? JS loader needed.
- *Verdict:* bare minimum for the web font loading strategy today. Self-hosting required.

CSS Font Loading API

- Native browser API à la Web Font Loader, with a FontFace object representing @font-face rules.
- JavaScript:

```
var elena_reg = new FontFace(
      'Elena Regular',
      'url(elena_reg.woff) format("woff"),' +
      'url(elena_reg.otf) format("otf")',
      { weight: 'regular', unicodeRange: 'U+0-7ff' }
);
```

- JavaScript:

```
var elena_reg = new FontFace(
      'Elena Regular',
      'url(elena_reg.woff) format("woff"),' +
      'url(elena_reg.otf) format("otf")',
      { weight: 'regular', unicodeRange: 'U+0-7ff' }
);
```

- JavaScript:

```
document.fonts.load('1em elena_reg')
.then(function() {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-loaded';
}).catch(function () {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-failed';
});
```

- JavaScript:

```
document.fonts.load('1em elena_reg')
.then(function() {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-loaded';
}).catch(function () {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-failed';
});
```

- CSS:

```
.elena_reg-loaded h1 {
      font-family: "Elena Regular";
}
```

- JavaScript:

```
document.fonts.load('1em elena_reg’)
.then(function() {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-loaded’;
}).catch(function () {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-failed’;
});
```

- CSS:

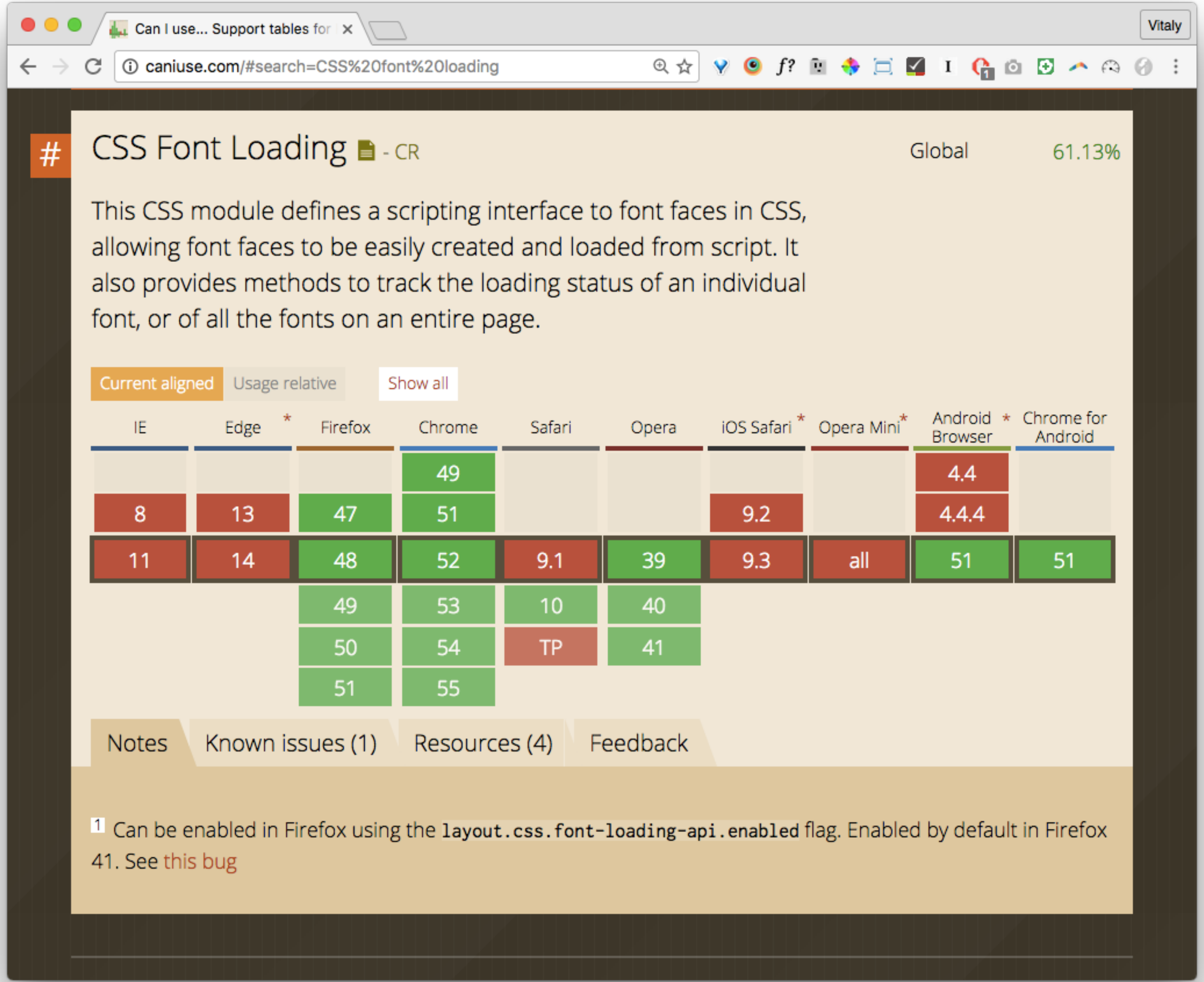
```
.elena_reg-loaded h1 {
      font-family: "Elena Regular";
      font-rendering: "block 0s swap infinite"; // FOUT
      // font-rendering: "block 3s swap infinite"; // FOIT
 }
```

- JavaScript:

```
document.fonts.load('1em elena_reg’)
.then(function() {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-loaded’;
}).catch(function () {
      var docEl = document.documentElement;
      docEl.className += ' elena_reg-failed’;
});
```

- CSS:

```
.elena_reg-loaded h1 {
      font-family: "Elena Regular";
      // font-rendering: "block 0s swap infinite"; // FOUT
      font-rendering: "block 3s swap 3s"; // FOIT, at most 3sec
 }
```



Can I use... Support tables for
Vitaly
caniuse.com/#search=CSS%20font%20loading

CSS Font Loading - CR

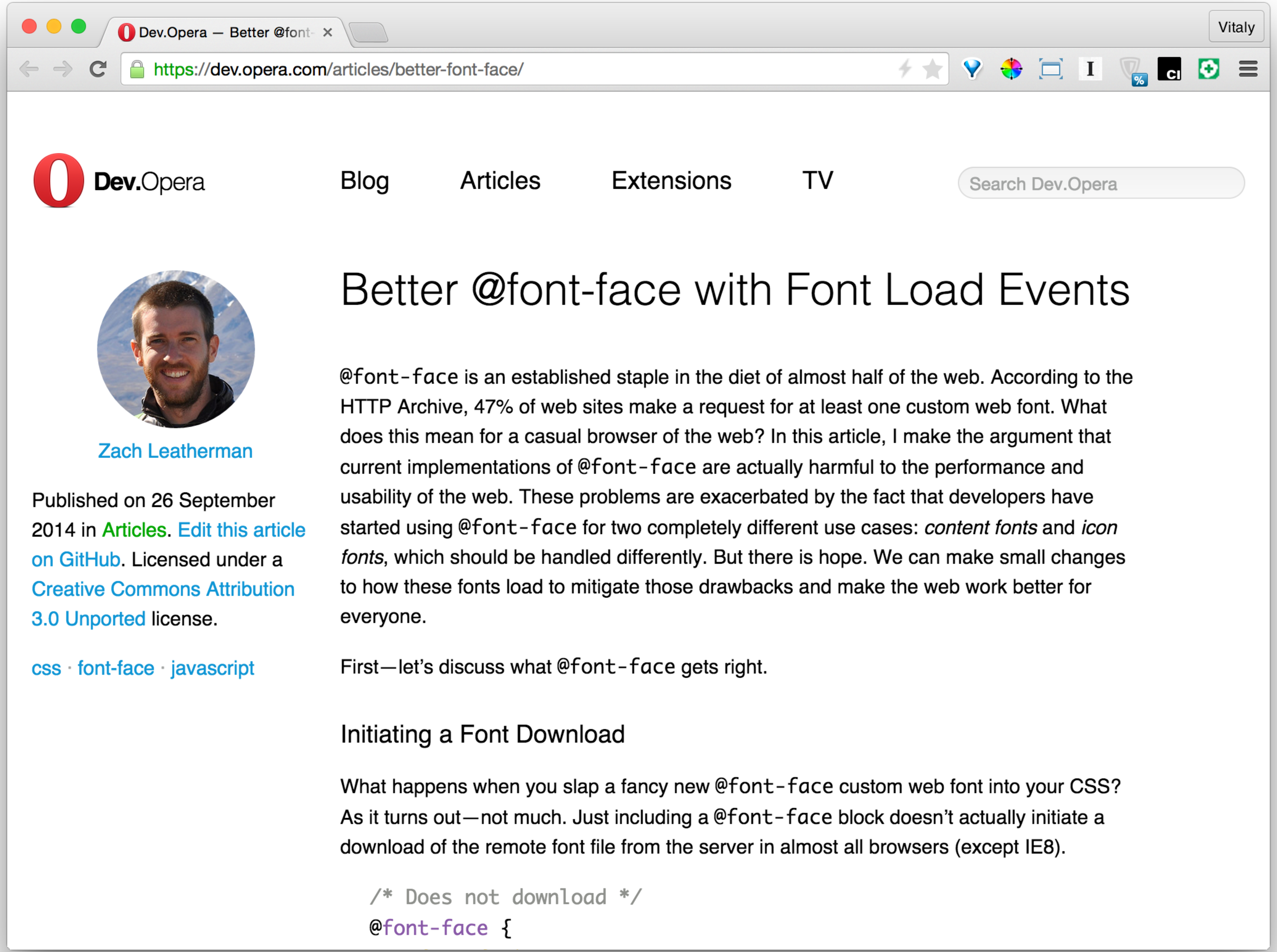
Global 61.13%

This CSS module defines a scripting interface to font faces in CSS, allowing font faces to be easily created and loaded from script. It also provides methods to track the loading status of an individual font, or of all the fonts on an entire page.

Current aligned
Usage relative
Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			🟩 49					🟥 4.4	
🟥 8	🟥 13	🟩 47	🟩 51			🟥 9.2		🟥 4.4.4	
🟥 11	🟥 14	🟩 48	🟩 52	🟥 9.1	🟩 39	🟥 9.3	🟥 all	🟩 51	🟩 51
		🟩 49	🟩 53	🟩 10	🟩 40				
		🟩 50	🟩 54	🟥 TP	🟩 41				
		🟩 51	🟩 55						

[1] Can be enabled in Firefox using the `layout.css.font-loading-api.enabled` flag. Enabled by default in Firefox 41. See this bug


Dev.Opera — Better @font-
Vitaly
https://dev.opera.com/articles/better-font-face/
Dev.Opera
Blog
Articles
Extensions
TV
Search Dev.Opera

Better @font-face with Font Load Events

Zach Leatherman

Published on 26 September 2014 in Articles. Edit this article on GitHub. Licensed under a Creative Commons Attribution 3.0 Unported license.

css · font-face · javascript

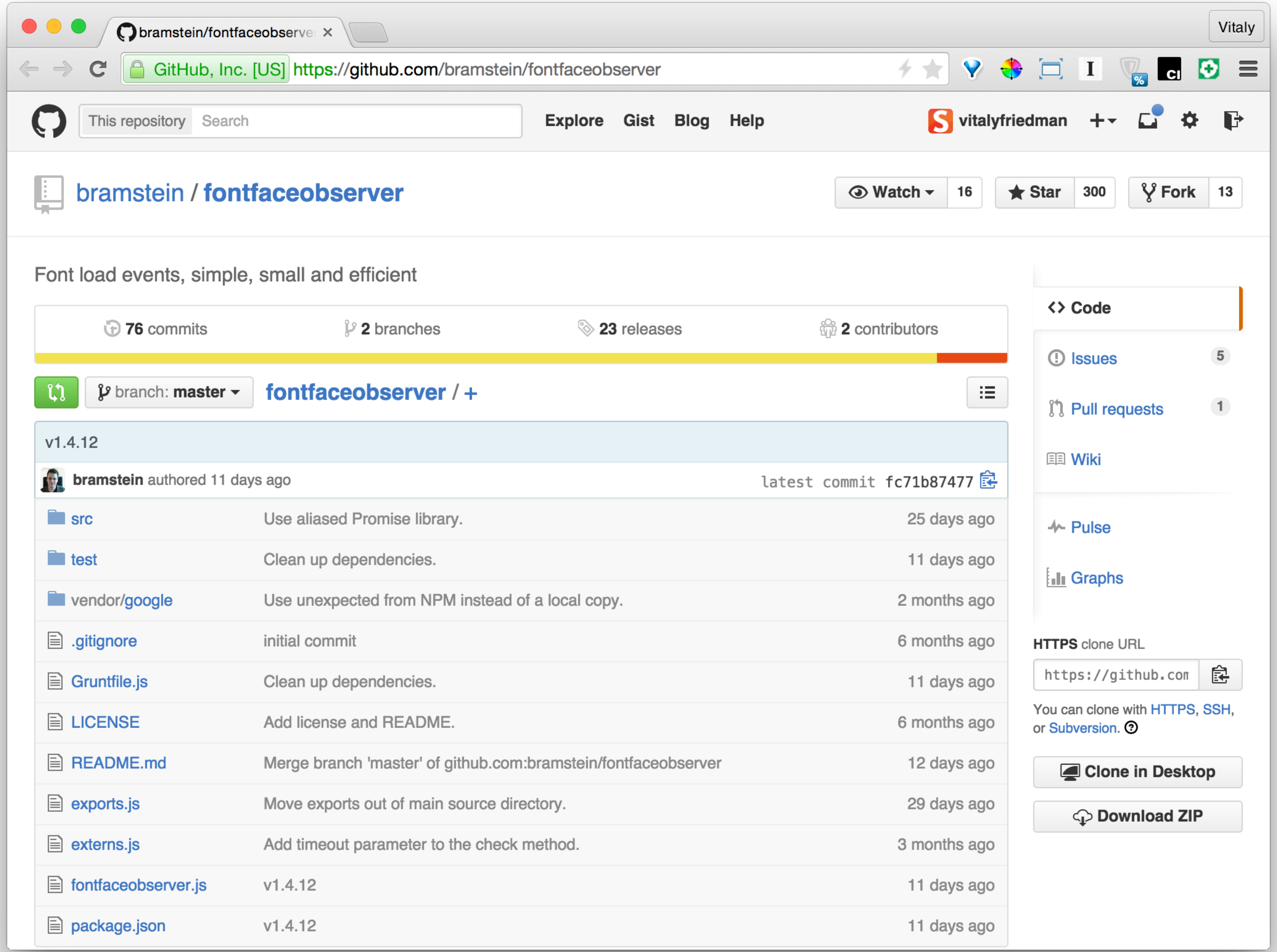
`@font-face` is an established staple in the diet of almost half of the web. According to the HTTP Archive, 47% of web sites make a request for at least one custom web font. What does this mean for a casual browser of the web? In this article, I make the argument that current implementations of `@font-face` are actually harmful to the performance and usability of the web. These problems are exacerbated by the fact that developers have started using `@font-face` for two completely different use cases: *content fonts* and *icon fonts*, which should be handled differently. But there is hope. We can make small changes to how these fonts load to mitigate those drawbacks and make the web work better for everyone.

First—let's discuss what `@font-face` gets right.

Initiating a Font Download

What happens when you slap a fancy new `@font-face` custom web font into your CSS? As it turns out—not much. Just including a `@font-face` block doesn't actually initiate a download of the remote font file from the server in almost all browsers (except IE8).

```
/* Does not download */
@font-face {
```



bramstein / fontfaceobserver

Font load events, simple, small and efficient

v1.4.12

bramstein authored 11 days ago latest commit fc71b87477

src	Use aliased Promise library.	25 days ago
test	Clean up dependencies.	11 days ago
vendor/google	Use unexpected from NPM instead of a local copy.	2 months ago
.gitignore	initial commit	6 months ago
Gruntfile.js	Clean up dependencies.	11 days ago
LICENSE	Add license and README.	6 months ago
README.md	Merge branch 'master' of github.com:bramstein/fontfaceobserver	12 days ago
exports.js	Move exports out of main source directory.	29 days ago
externs.js	Add timeout parameter to the check method.	3 months ago
fontfaceobserver.js	v1.4.12	11 days ago
package.json	v1.4.12	11 days ago

HTTPS clone URL

https://github.com

You can clone with HTTPS, SSH, or Subversion.

Font Load Events

- Use the *CSS Font Loading API* with a polyfill to apply web font only after it has loaded successfully.
    - Toggle a class on *<html>*; with Sass/LESS mixins,
    - Optimize for *repeat* views with sessionStorage,
    - Easy to implement with 3rd-party hosts,
    - Requires strict control of CSS; a single use of a web font font-family will trigger a FOIT.
- *Verdict:* good option for web font loading, to integrate with 3rd-party hosting providers.

Flash of Faux Text

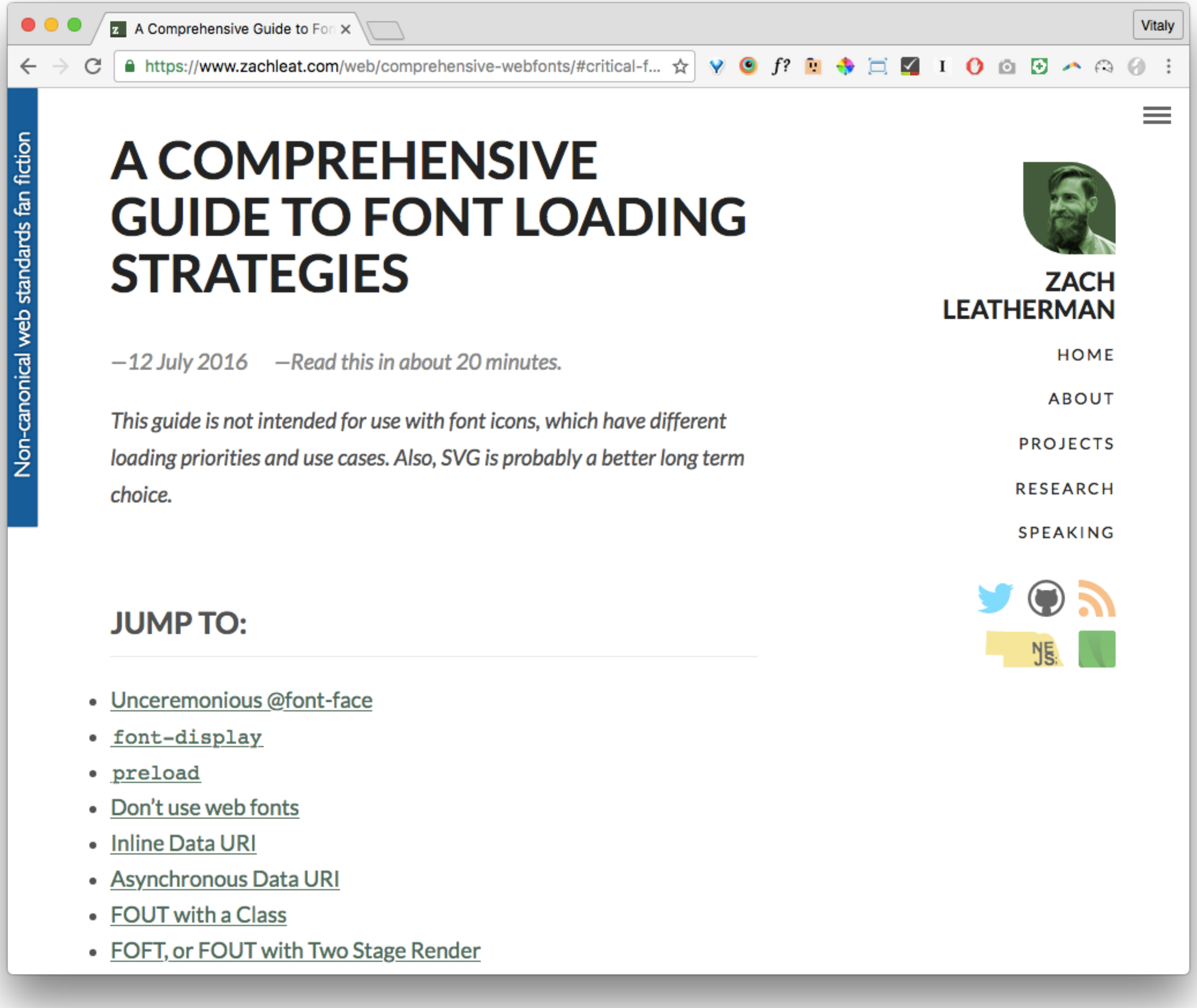
- When using multiple weights, we split web fonts into groups: Roman / *Faux* content.
    - *Two-stage render:* Roman first and rest later,
    - Optimize for *repeat* views with sessionStorage,
    - Font synthesis is a big drawback.
- *Verdict:* good option for great performance, but font synthesis might produce awkward results.

Critical FOFT

- When using multiple weights, we split web fonts into groups: Roman / *Faux* content.
    - *Two-stage render:* Roman first and rest later,
    - Subset fonts to *minimum* *(A–Z, 0–9, punctuation),*
    - Optimize for *repeat* views with sessionStorage,
    - Font synthesis is a big drawback.
    - Subset is duplicated in the full Roman font.
    - Licensing issues: requires subsetting.
- *Verdict:* good option for great performance, but font synthesis might produce awkward results.

Critical FOFT With Data URI

- Instead of loading via a JavaScript API, we *inline* the web font directly in the markup.
    - *Two-stage render:* Roman first and rest later,
    - Subset fonts to *minimum* *(A–Z, 0–9, punctuation),*
    - Load the subsetted font (Roman) *first* inline,
    - Load full fonts with all weights and styles async,
    - Use *sessionStorage* for return visits,
    - Requires self-hosting; data URI blocks rendering.
- *Verdict:* the fastest web font loading strategy as of today. Eliminates FOIT and greatly reduces FOUT.



A Comprehensive Guide to For
https://www.zachleat.com/web/comprehensive-webfonts/#critical-f...
Vitaly
Non-canonical web standards fan fiction
ZACH LEATHERMAN
HOME
ABOUT
PROJECTS
RESEARCH
SPEAKING

A COMPREHENSIVE GUIDE TO FONT LOADING STRATEGIES

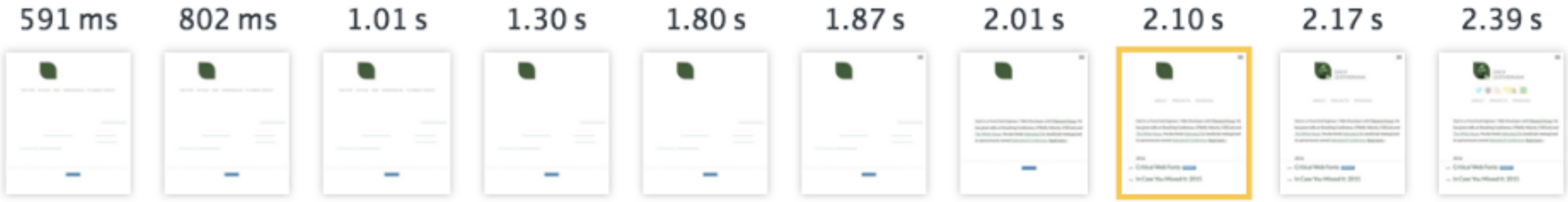
—12 July 2016 —Read this in about 20 minutes.

This guide is not intended for use with font icons, which have different loading priorities and use cases. Also, SVG is probably a better long term choice.

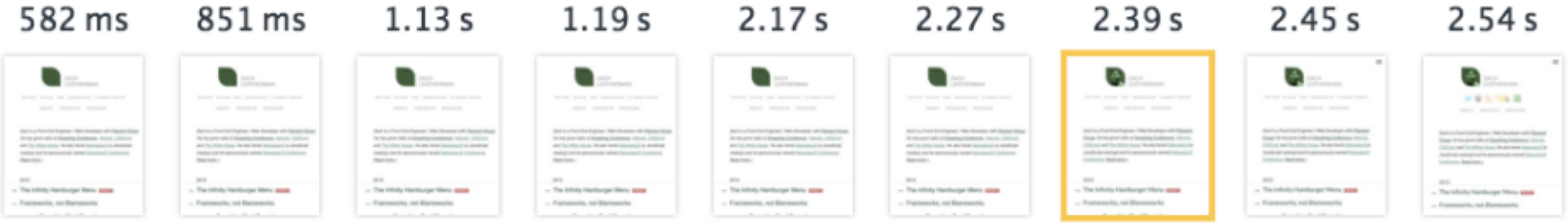
JUMP TO:

- Unceremonious @font-face
- `font-display`
- `preload`
- Don't use web fonts
- Inline Data URI
- Asynchronous Data URI
- FOUT with a Class
- FOFT, or FOUT with Two Stage Render

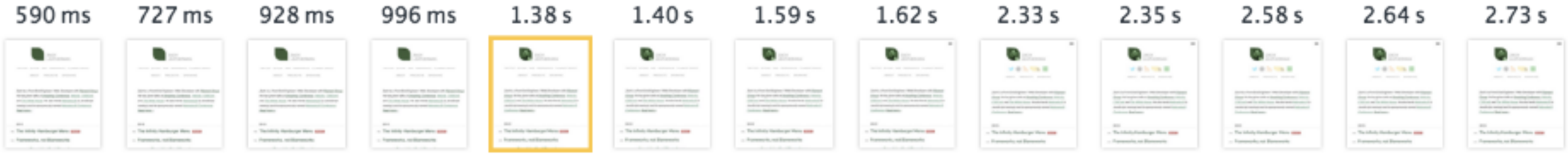
DEFAULT



SCOPED CLASS FOR FOUT

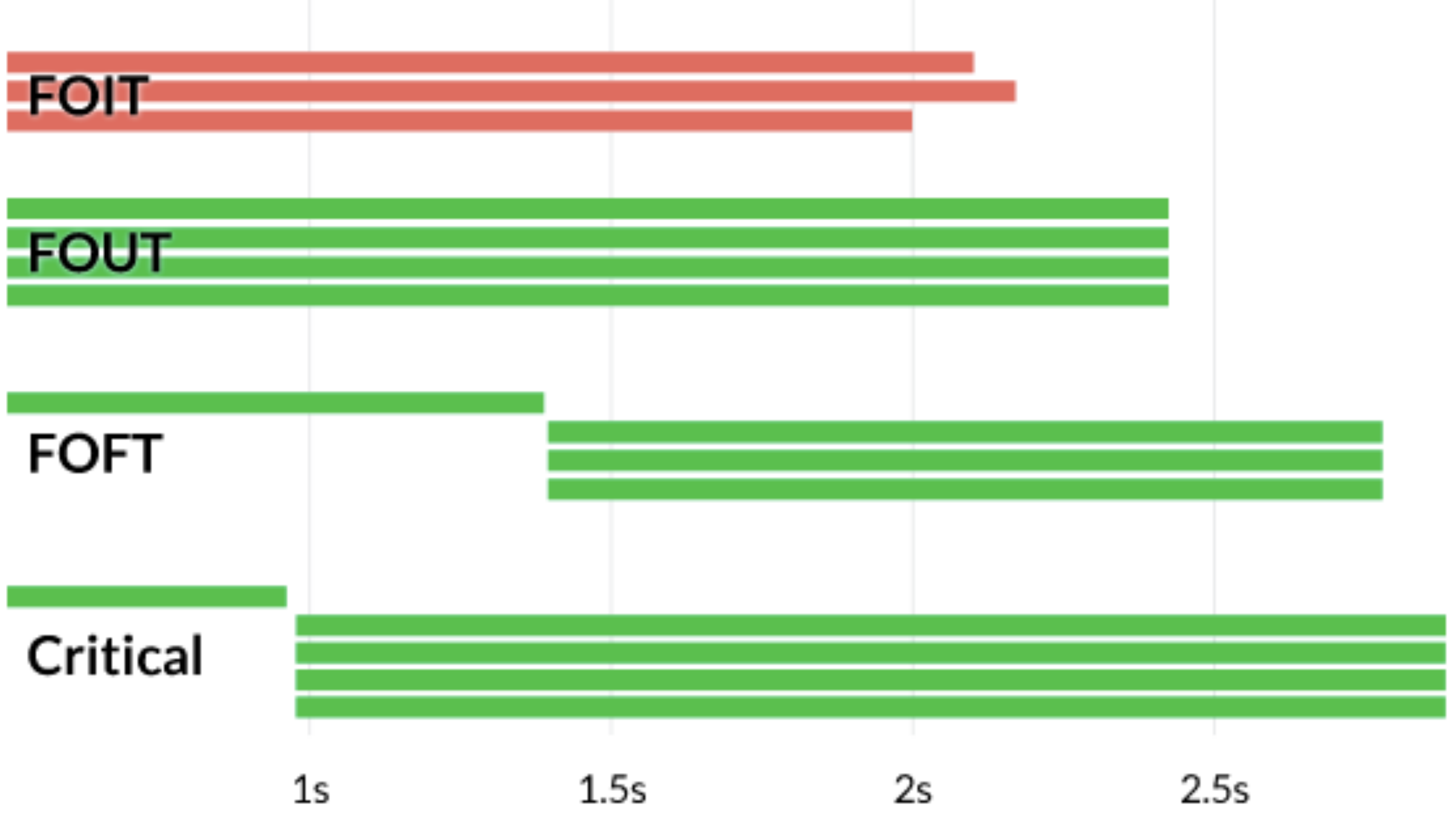


TWO SCOPED CLASSES FOR FOFT



TWO SCOPED CLASSES FOR CRITICAL FOFT

PERFORMANCE COMPARISON

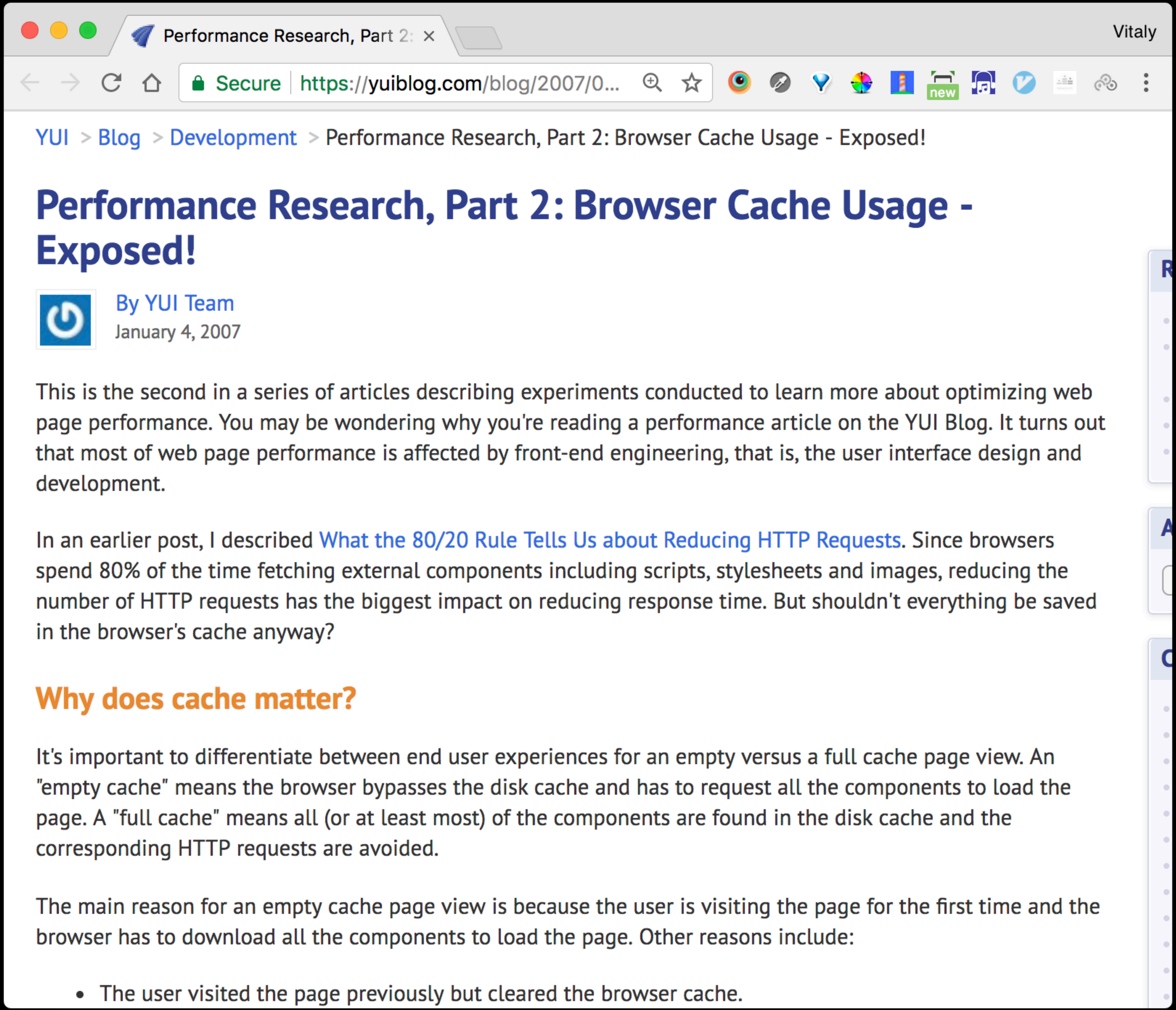


Critical 2-Stage-FOFT-Render With Data URI/ServiceWorker (C2SFOFTRWDURISW)

- Instead of using sessionStorage, we *inline* the web font in the markup *and* use Service Workers cache.
    - *Two-stage render:* Roman first and rest later,
    - Subset fonts to *minimum (A–Z, 0–9, punctuation),*
    - Load the subsetted font (Roman) *first* inline,
    - Load full fonts with all weights and styles async,
    - Use *Service Workers* for return visits,
    - Requires self-hosting/HTTPS; data URI blocks rendering.
- *Verdict:* the fastest web font loading strategy as of today. Eliminates FOIT and greatly reduces FOUT.

Caching

In most cases, when a web page needs a resource, Chrome starts by looking it up in the **Memory cache**. If the Memory cache doesn’t have it, Chrome will then ask the network stack to handle the request. The network stack will eventually process the request and will start by looking for the resource in the **HTTP cache**. If the HTTP cache doesn’t have it, the network stack will then issue an actual **network** request.


Performance Research, Part 2:
Vitaly
Secure
https://yuiblog.com/blog/2007/0...
new

YUI > Blog > Development > Performance Research, Part 2: Browser Cache Usage - Exposed!

Performance Research, Part 2: Browser Cache Usage - Exposed!

By YUI Team
January 4, 2007

This is the second in a series of articles describing experiments conducted to learn more about optimizing web page performance. You may be wondering why you're reading a performance article on the YUI Blog. It turns out that most of web page performance is affected by front-end engineering, that is, the user interface design and development.

In an earlier post, I described What the 80/20 Rule Tells Us about Reducing HTTP Requests. Since browsers spend 80% of the time fetching external components including scripts, stylesheets and images, reducing the number of HTTP requests has the biggest impact on reducing response time. But shouldn't everything be saved in the browser's cache anyway?

Why does cache matter?

It's important to differentiate between end user experiences for an empty versus a full cache page view. An "empty cache" means the browser bypasses the disk cache and has to request all the components to load the page. A "full cache" means all (or at least most) of the components are found in the disk cache and the corresponding HTTP requests are avoided.

The main reason for an empty cache page view is because the user is visiting the page for the first time and the browser has to download all the components to load the page. Other reasons include:

- The user visited the page previously but cleared the browser cache.

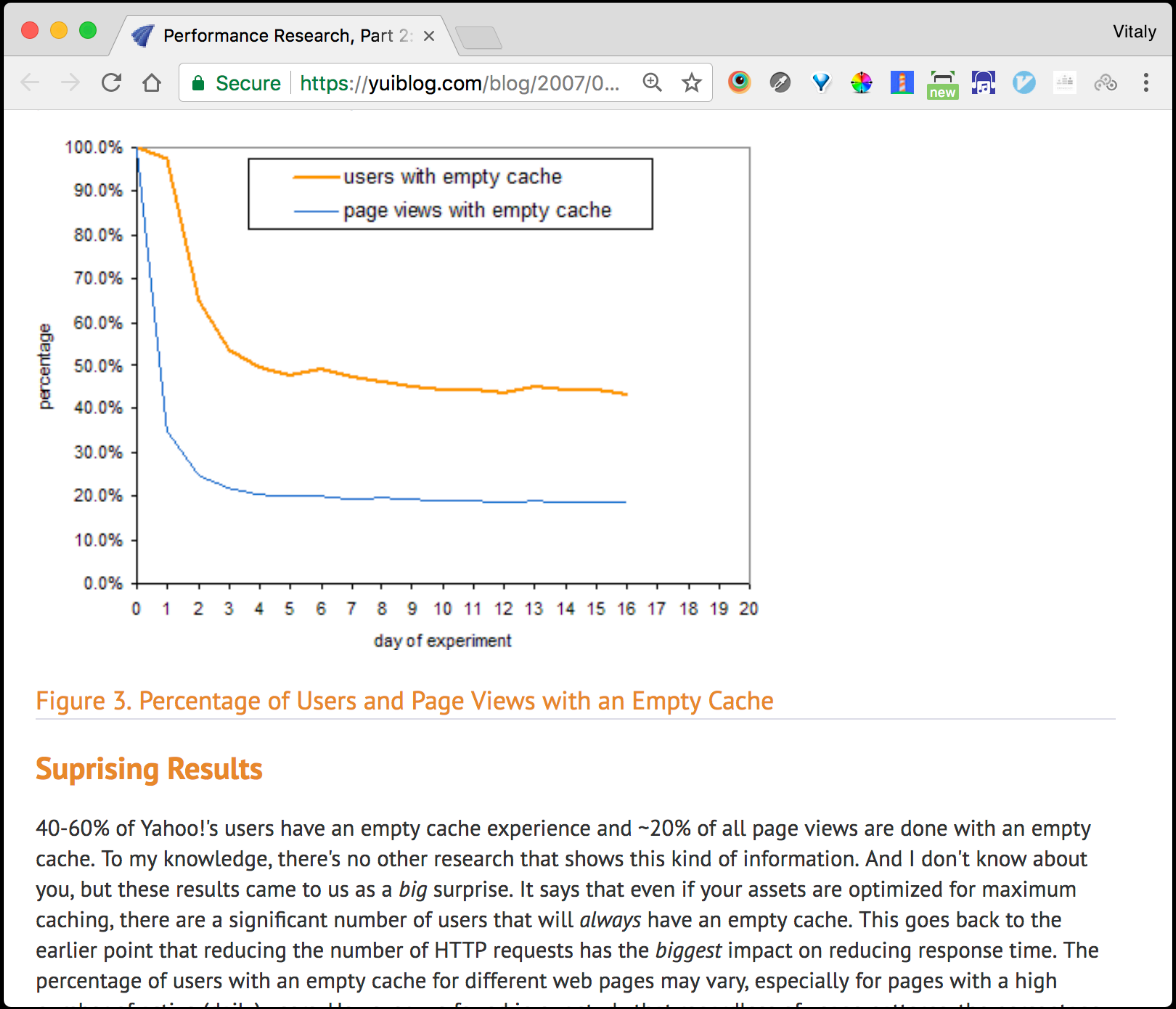
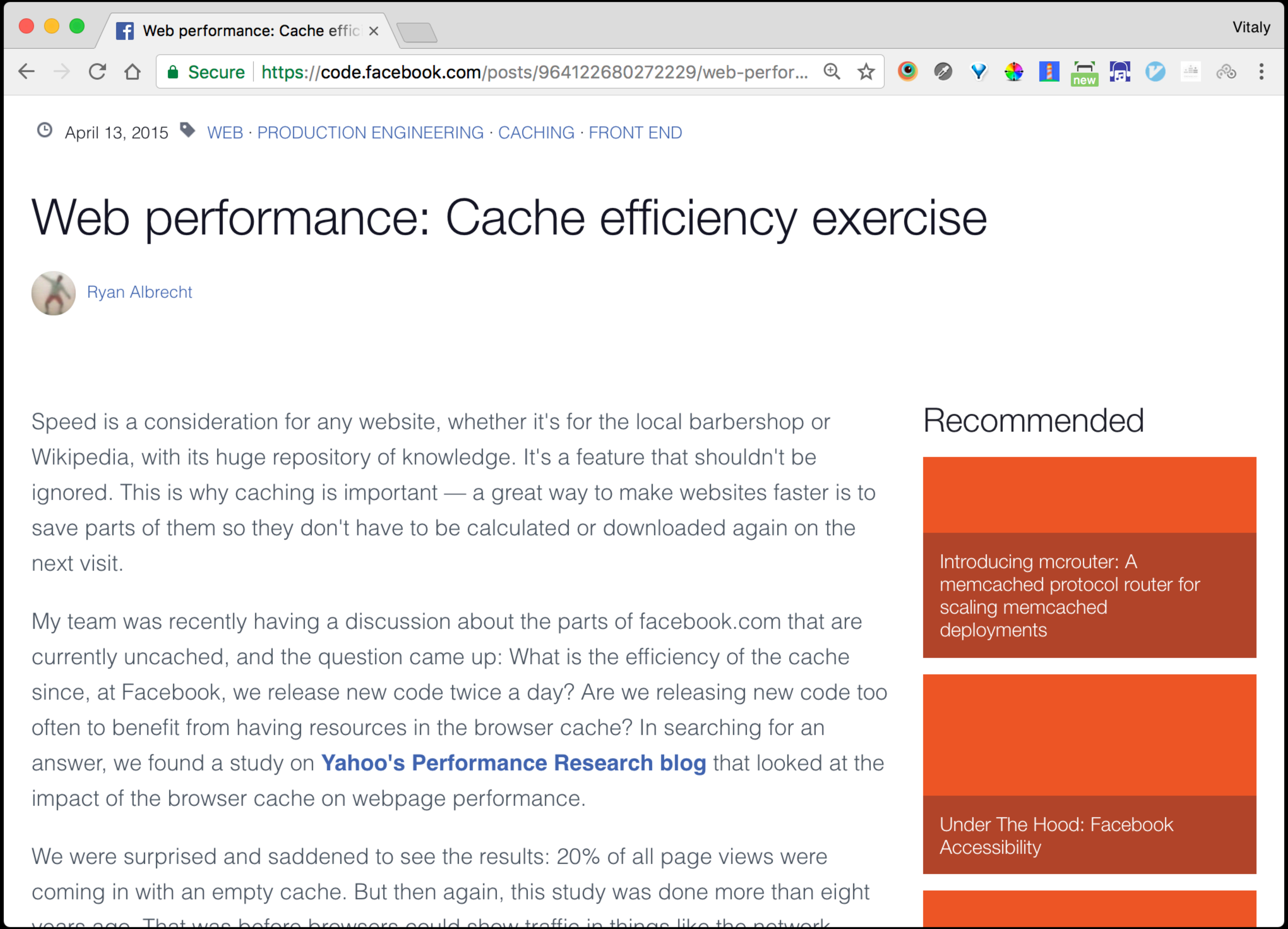


Figure 3. Percentage of Users and Page Views with an Empty Cache

Suprising Results

40-60% of Yahoo!'s users have an empty cache experience and ~20% of all page views are done with an empty cache. To my knowledge, there's no other research that shows this kind of information. And I don't know about you, but these results came to us as a *big* surprise. It says that even if your assets are optimized for maximum caching, there are a significant number of users that will *always* have an empty cache. This goes back to the earlier point that reducing the number of HTTP requests has the *biggest* impact on reducing response time. The percentage of users with an empty cache for different web pages may vary, especially for pages with a high


Web performance: Cache effic
Vitaly
Secure
https://code.facebook.com/posts/964122680272229/web-perfor...
new

April 13, 2015 WEB · PRODUCTION ENGINEERING · CACHING · FRONT END

Web performance: Cache efficiency exercise

Ryan Albrecht

Speed is a consideration for any website, whether it's for the local barbershop or Wikipedia, with its huge repository of knowledge. It's a feature that shouldn't be ignored. This is why caching is important — a great way to make websites faster is to save parts of them so they don't have to be calculated or downloaded again on the next visit.

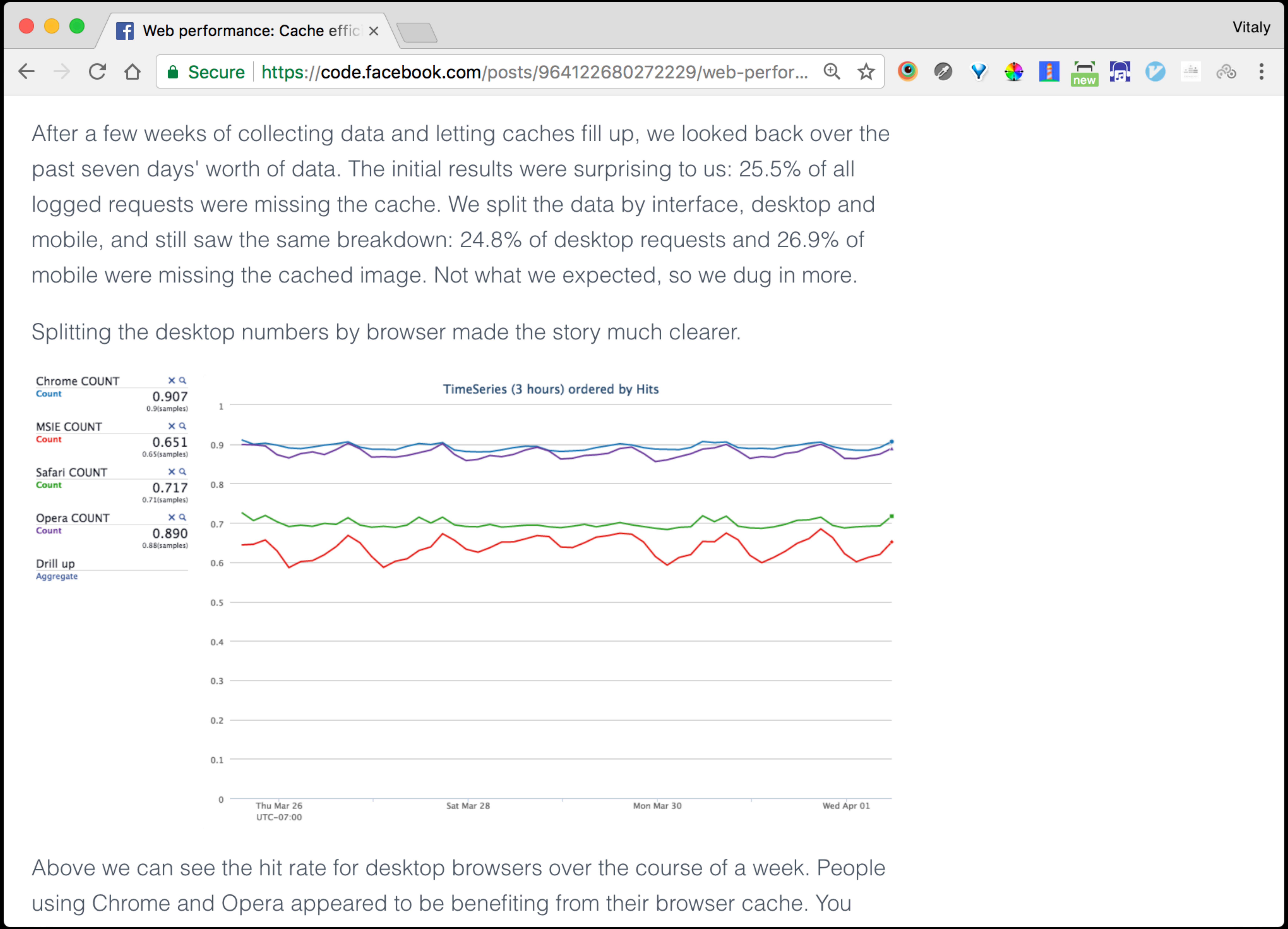
My team was recently having a discussion about the parts of facebook.com that are currently uncached, and the question came up: What is the efficiency of the cache since, at Facebook, we release new code twice a day? Are we releasing new code too often to benefit from having resources in the browser cache? In searching for an answer, we found a study on **Yahoo's Performance Research blog** that looked at the impact of the browser cache on webpage performance.

We were surprised and saddened to see the results: 20% of all page views were coming in with an empty cache. But then again, this study was done more than eight years ago. That was before browsers could show traffic in things like the network

Recommended

Introducing mcrouter: A memcached protocol router for scaling memcached deployments

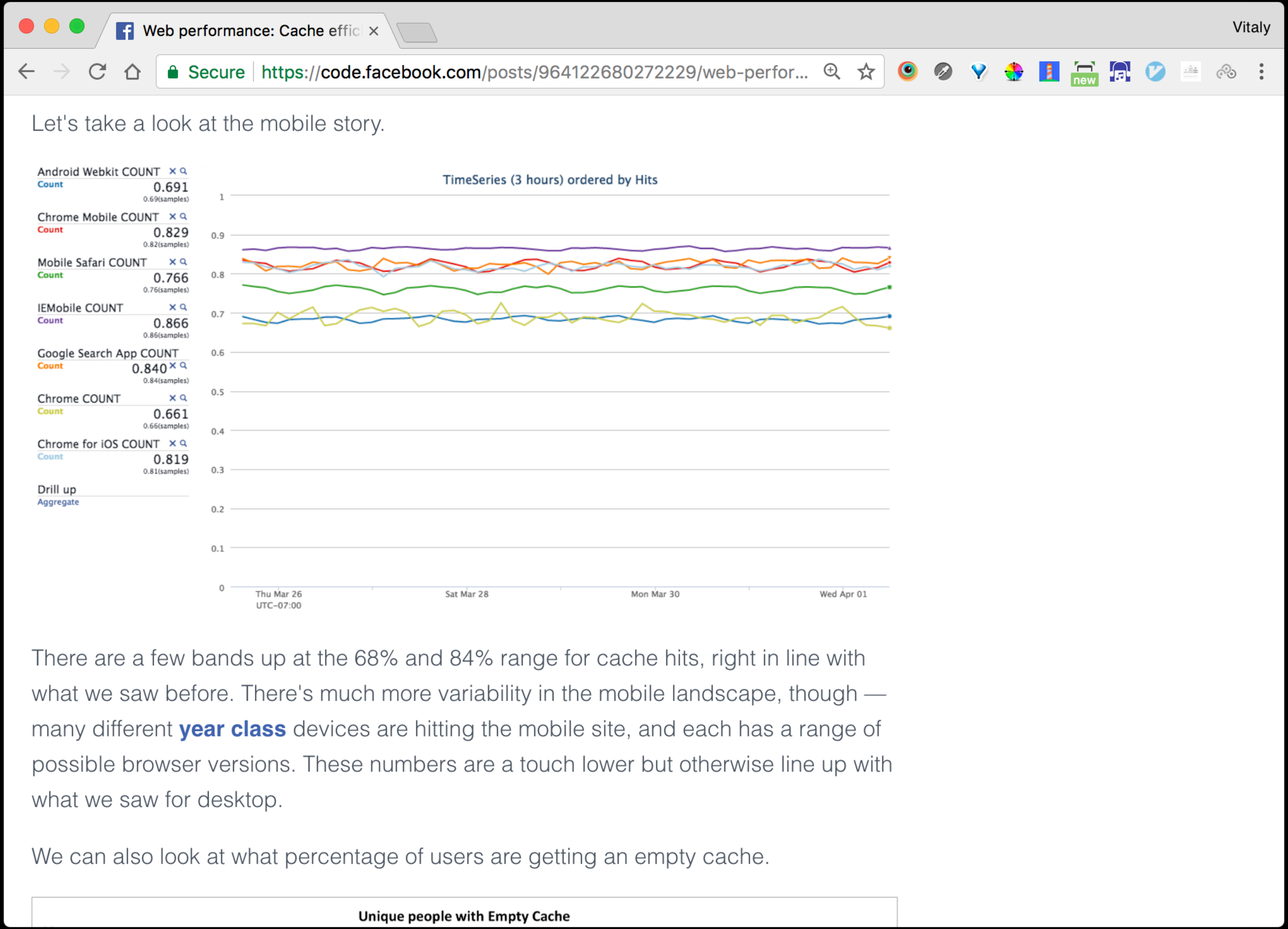
Under The Hood: Facebook Accessibility


Web performance: Cache effic
Vitaly
Secure
https://code.facebook.com/posts/964122680272229/web-perfor...

After a few weeks of collecting data and letting caches fill up, we looked back over the past seven days' worth of data. The initial results were surprising to us: 25.5% of all logged requests were missing the cache. We split the data by interface, desktop and mobile, and still saw the same breakdown: 24.8% of desktop requests and 26.9% of mobile were missing the cached image. Not what we expected, so we dug in more.

Splitting the desktop numbers by browser made the story much clearer.

Above we can see the hit rate for desktop browsers over the course of a week. People using Chrome and Opera appeared to be benefiting from their browser cache. You

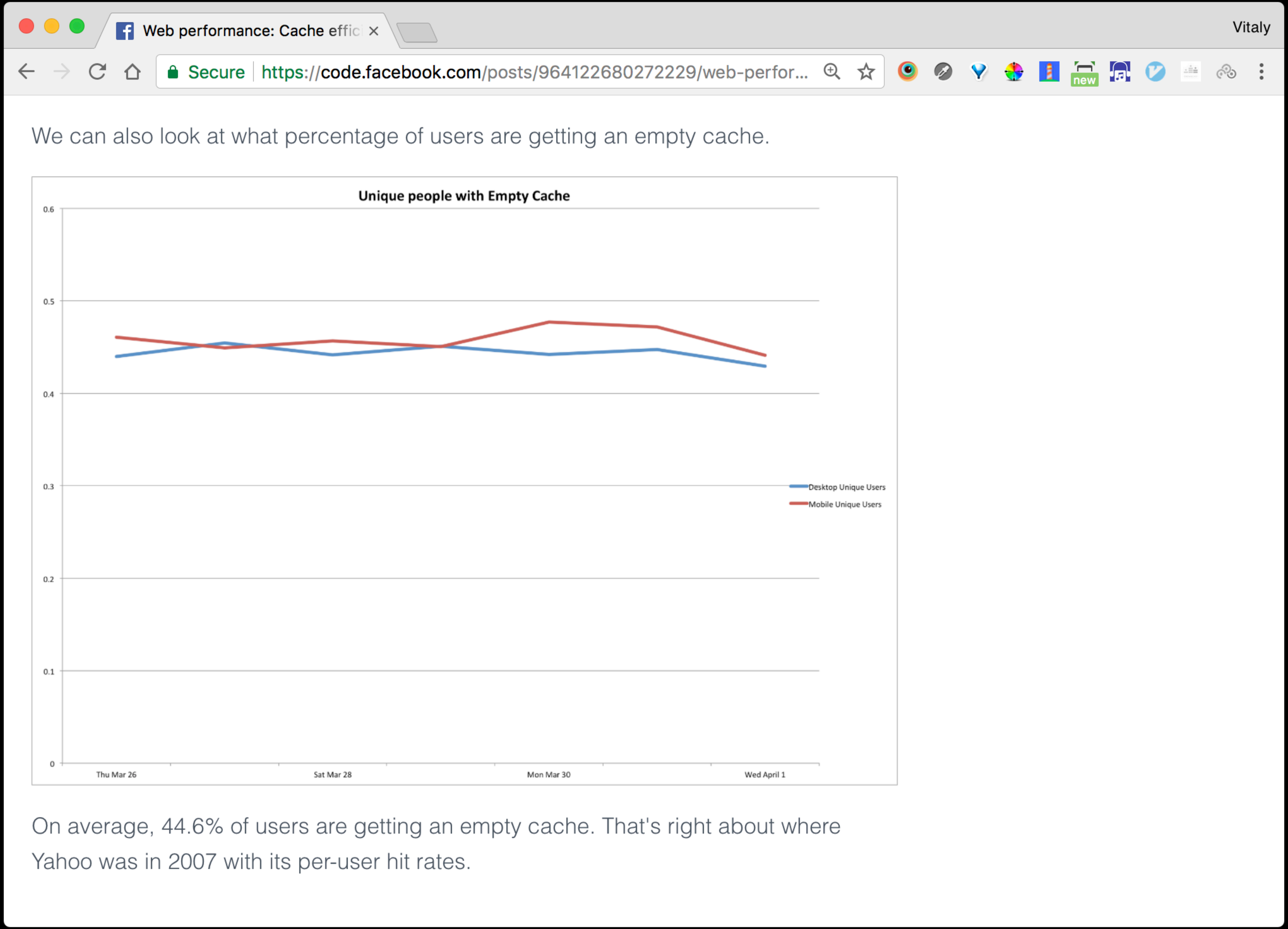


Let's take a look at the mobile story.

There are a few bands up at the 68% and 84% range for cache hits, right in line with what we saw before. There's much more variability in the mobile landscape, though — many different **year class** devices are hitting the mobile site, and each has a range of possible browser versions. These numbers are a touch lower but otherwise line up with what we saw for desktop.

We can also look at what percentage of users are getting an empty cache.

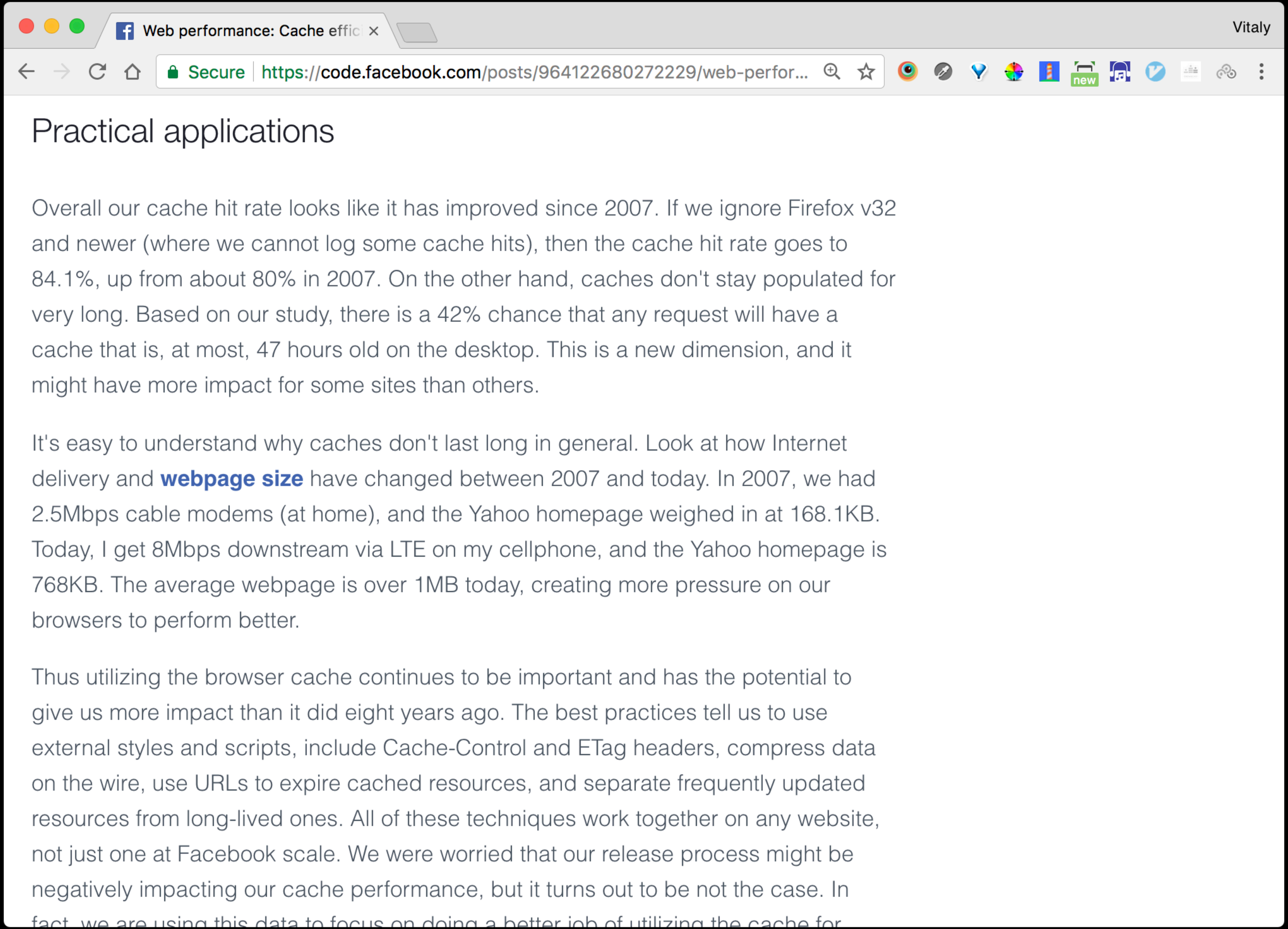
Unique people with Empty Cache


Web performance: Cache effic
Vitaly
Secure
https://code.facebook.com/posts/964122680272229/web-perfor...
new

We can also look at what percentage of users are getting an empty cache.

Unique people with Empty Cache
0.6
0.5
0.4
0.3
0.2
0.1
0
Thu Mar 26
Sat Mar 28
Mon Mar 30
Wed April 1
Desktop Unique Users
Mobile Unique Users

On average, 44.6% of users are getting an empty cache. That's right about where Yahoo was in 2007 with its per-user hit rates.


Web performance: Cache effic
Vitaly
Secure
https://code.facebook.com/posts/964122680272229/web-perfor...
new

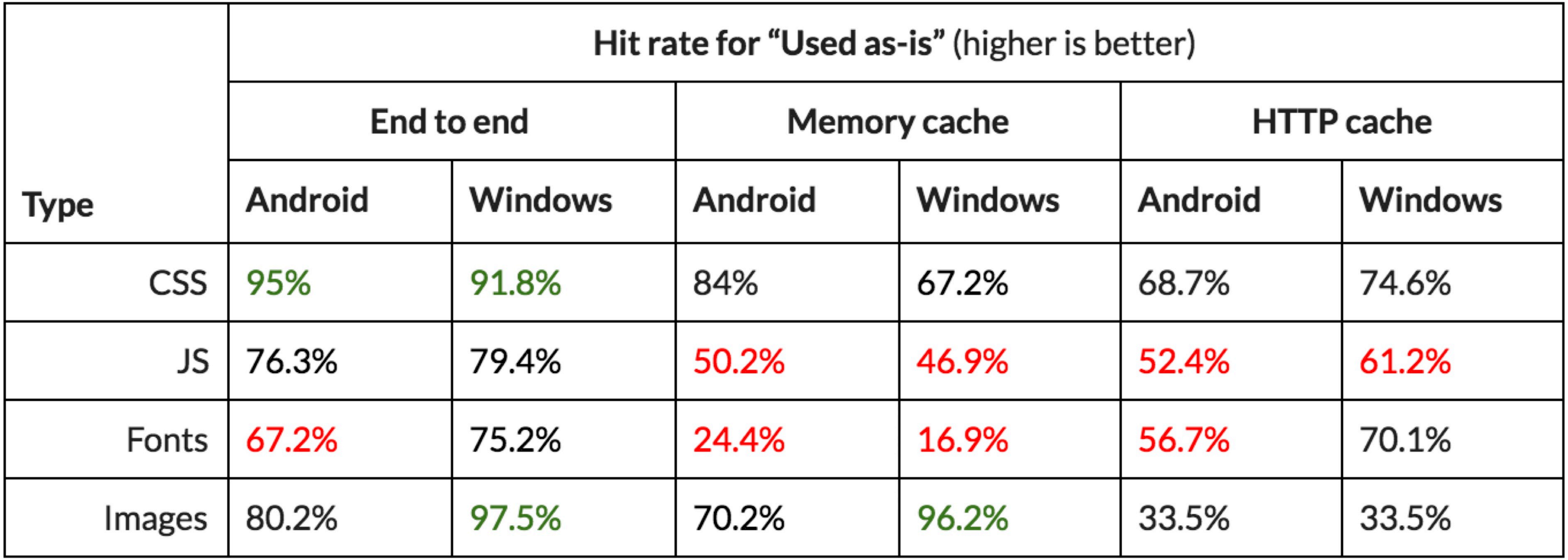
Practical applications

Overall our cache hit rate looks like it has improved since 2007. If we ignore Firefox v32 and newer (where we cannot log some cache hits), then the cache hit rate goes to 84.1%, up from about 80% in 2007. On the other hand, caches don't stay populated for very long. Based on our study, there is a 42% chance that any request will have a cache that is, at most, 47 hours old on the desktop. This is a new dimension, and it might have more impact for some sites than others.

It's easy to understand why caches don't last long in general. Look at how Internet delivery and **webpage size** have changed between 2007 and today. In 2007, we had 2.5Mbps cable modems (at home), and the Yahoo homepage weighed in at 168.1KB. Today, I get 8Mbps downstream via LTE on my cellphone, and the Yahoo homepage is 768KB. The average webpage is over 1MB today, creating more pressure on our browsers to perform better.

Thus utilizing the browser cache continues to be important and has the potential to give us more impact than it did eight years ago. The best practices tell us to use external styles and scripts, include Cache-Control and ETag headers, compress data on the wire, use URLs to expire cached resources, and separate frequently updated resources from long-lived ones. All of these techniques work together on any website, not just one at Facebook scale. We were worried that our release process might be negatively impacting our cache performance, but it turns out to be not the case. In fact, we are using this data to focus on doing a better job of utilizing the cache for

Chrome’s Cache Hit Rates



Type	Hit rate for “Used as-is” (higher is better)					
	End to end		Memory cache		HTTP cache	
	Android	Windows	Android	Windows	Android	Windows
CSS	95%	91.8%	84%	67.2%	68.7%	74.6%
JS	76.3%	79.4%	50.2%	46.9%	52.4%	61.2%
Fonts	67.2%	75.2%	24.4%	16.9%	56.7%	70.1%
Images	80.2%	97.5%	70.2%	96.2%	33.5%	33.5%

* Chrome has 4+ caches. The above reflects the main two - the HTTP and memory caches

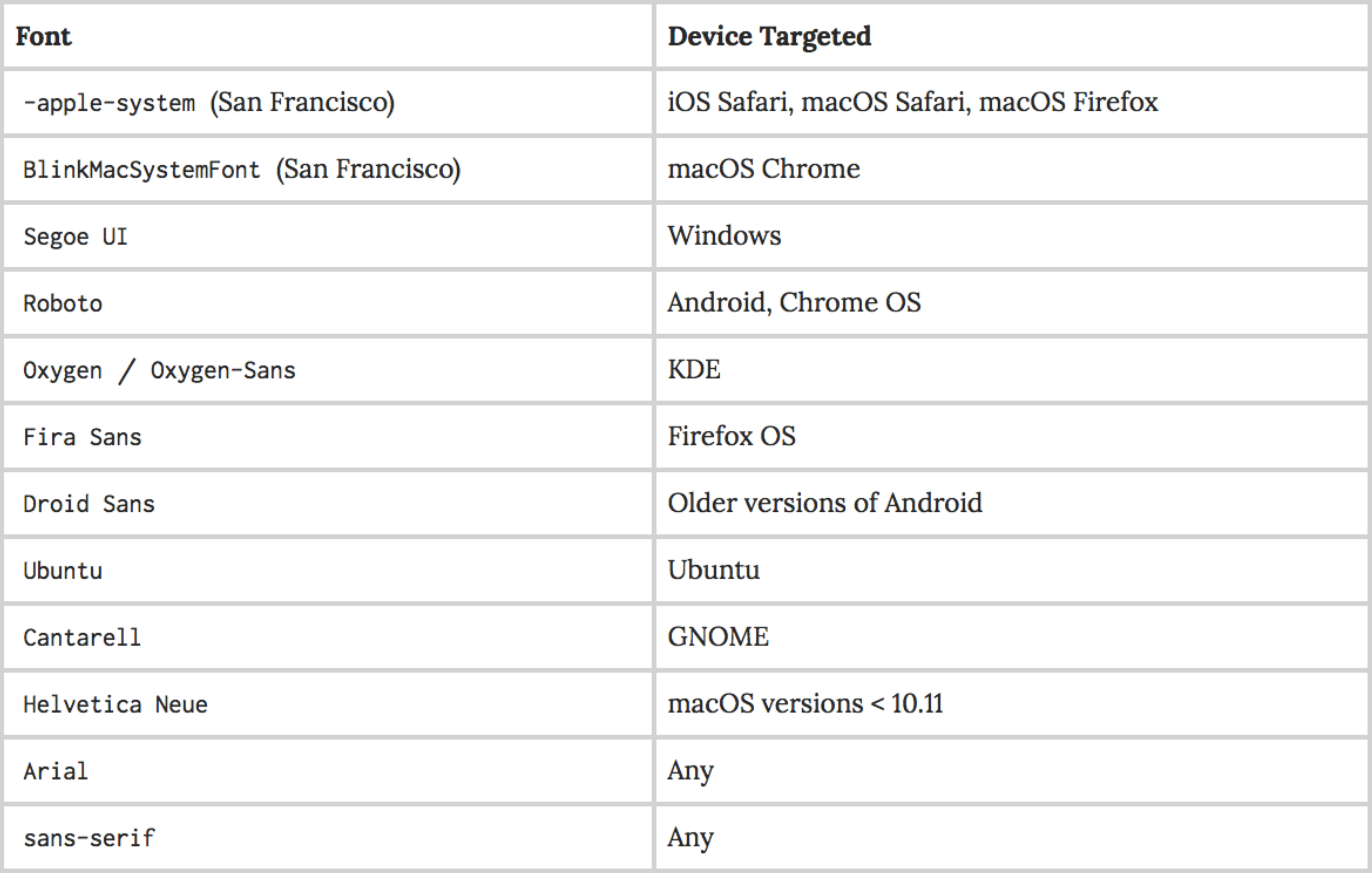
- When a font family name is used in CSS, browsers match it against all @font-face rules, download web fonts, display content.

- CSS:

```
body {
   font-family: 'Elena Regular',
   AvenirNext, Avenir,                 /* iOS */
  'Roboto Slab', 'Droid Serif',        /* Android */
  'Segoe UI',                          /* Microsoft  */
   Georgia, 'Times New Roman', serif;  /* Fallback  */
}
```

- CSS:

```
body {
   font-family: 'Elena Regular',
   AvenirNext, Avenir,                 /* iOS */
  'Roboto Slab', 'Droid Serif',        /* Android */
  'Segoe UI',                          /* Microsoft  */
   Georgia, 'Times New Roman', serif;  /* Fallback  */
}
```



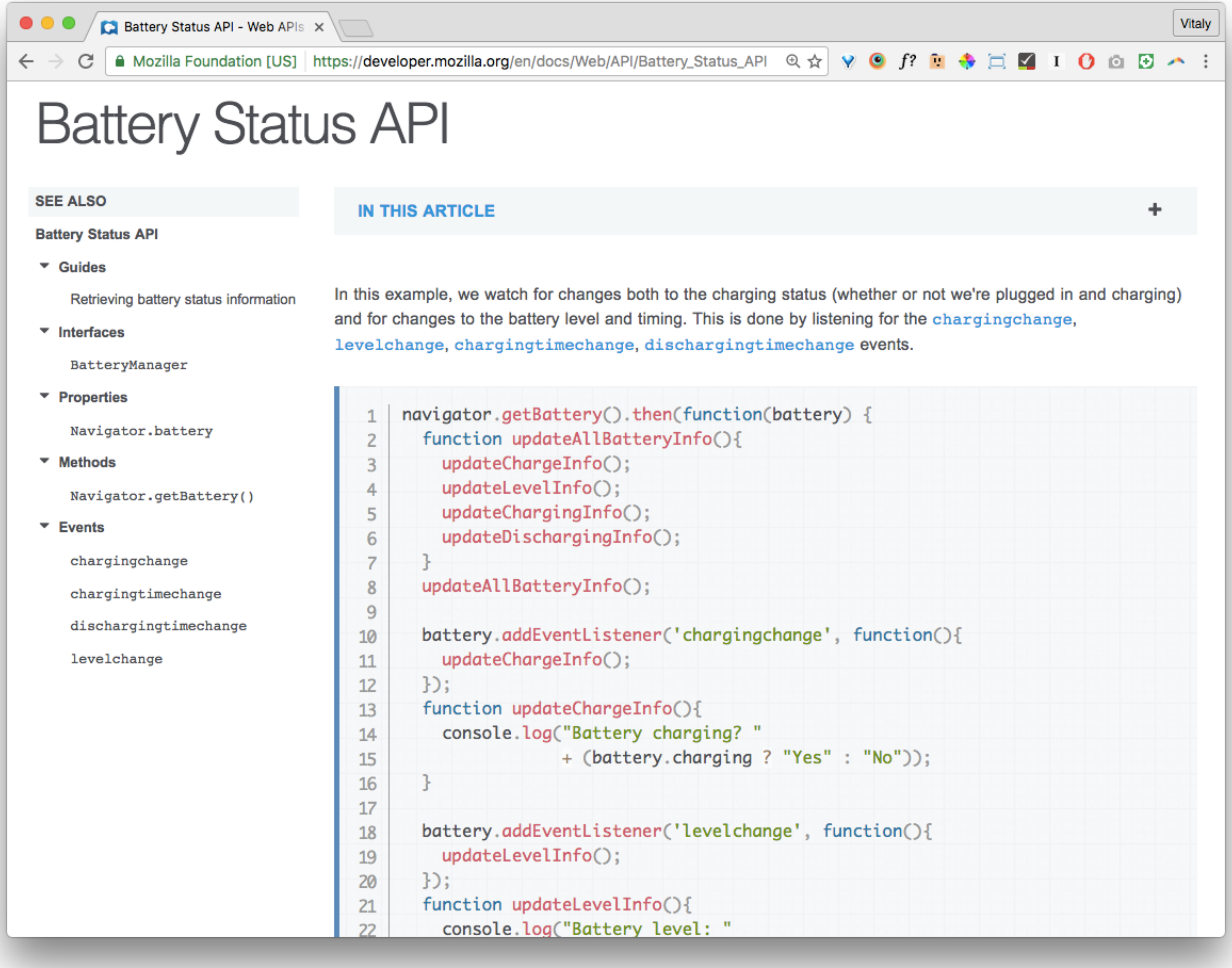
Font	Device Targeted
`-apple-system` (San Francisco)	iOS Safari, macOS Safari, macOS Firefox
`BlinkMacSystemFont` (San Francisco)	macOS Chrome
`Segoe UI`	Windows
`Roboto`	Android, Chrome OS
`Oxygen` / `Oxygen-Sans`	KDE
`Fira Sans`	Firefox OS
`Droid Sans`	Older versions of Android
`Ubuntu`	Ubuntu
`Cantarell`	GNOME
`Helvetica Neue`	macOS versions < 10.11
`Arial`	Any
`sans-serif`	Any

- CSS:

```
body {
   font-family: 'Elena Regular',     /* Web font */
   AvenirNext, Avenir,                /* iOS */
  -apple-system, BlinkMacSystemFont,  /* macOS San Francisco */
   Roboto Slab', 'Droid Serif',       /* Android */
  'Segoe UI',                         /* Microsoft  */
   Oxygen-Sans,                       /* KDE */
   Ubuntu,                            /* Ubuntu */
   Cantarell,                         /* GNOME */
   Georgia, 'Times New Roman', serif; /* Fallback  */
}
```

- CSS:

```
.lowBattery {
   font-family: /* 'Elena Regular' */ /* Web font */
   AvenirNext, Avenir,              /* iOS */
  -apple-system, BlinkMacSystemFont,   /* macOS San Francisco */
   Roboto Slab', 'Droid Serif',        /* Android */
  'Segoe UI',                          /* Microsoft  */
   Oxygen-Sans,                        /* KDE */
   Ubuntu,                             /* Ubuntu */
   Cantarell,                          /* GNOME */
   Georgia, 'Times New Roman', serif;  /* Fallback  */
}
```


Battery Status API - Web APIs
Vitaly
Mozilla Foundation [US]
https://developer.mozilla.org/en/docs/Web/API/Battery_Status_API

Battery Status API

SEE ALSO

Battery Status API

- Guides
  - Retrieving battery status information
- Interfaces
  - BatteryManager
- Properties
  - Navigator.battery
- Methods
  - Navigator.getBattery()
- Events
  - chargingchange
  - chargingtimechange
  - dischargingtimechange
  - levelchange

IN THIS ARTICLE +

In this example, we watch for changes both to the charging status (whether or not we're plugged in and charging) and for changes to the battery level and timing. This is done by listening for the `chargingchange`, `levelchange`, `chargingtimechange`, `dischargingtimechange` events.

```
navigator.getBattery().then(function(battery) {
  function updateAllBatteryInfo(){
    updateChargeInfo();
    updateLevelInfo();
    updateChargingInfo();
    updateDischargingInfo();
  }
  updateAllBatteryInfo();

  battery.addEventListener('chargingchange', function(){
    updateChargeInfo();
  });
  function updateChargeInfo(){
    console.log("Battery charging? "
                + (battery.charging ? "Yes" : "No"));
  }

  battery.addEventListener('levelchange', function(){
    updateLevelInfo();
  });
  function updateLevelInfo(){
    console.log("Battery level: "
```

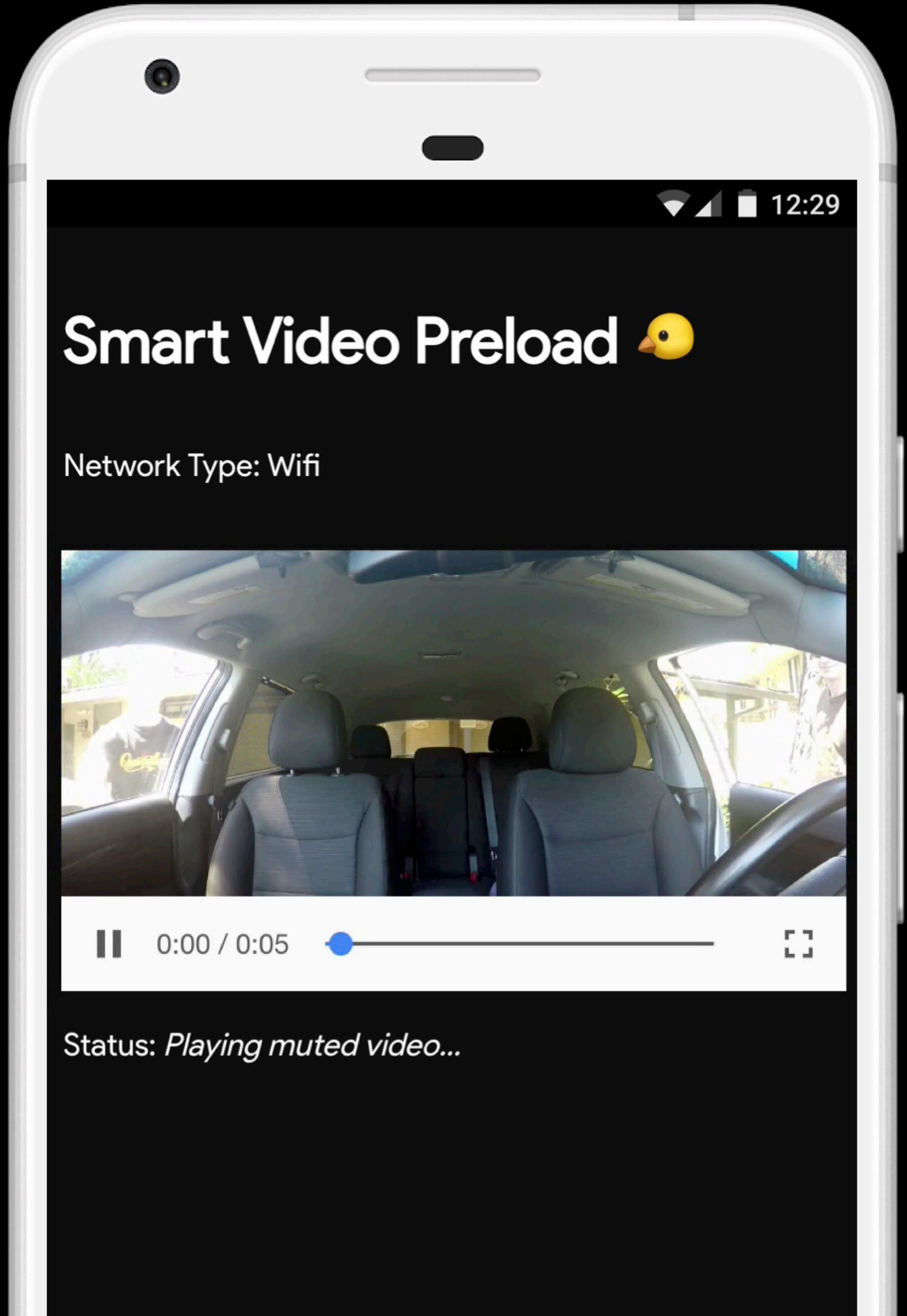
I want to adapt serving based on estimated network quality

```
// Network type that browser uses
navigator.connection.type
> 'wifi'

// New: Effective connection type
// using rtt and downlink values
navigator.connection.effectiveType
> '2G'
```

For more on navigator.connection.*
See 'Building a modern media experience'

Chrome 62

BEFORE

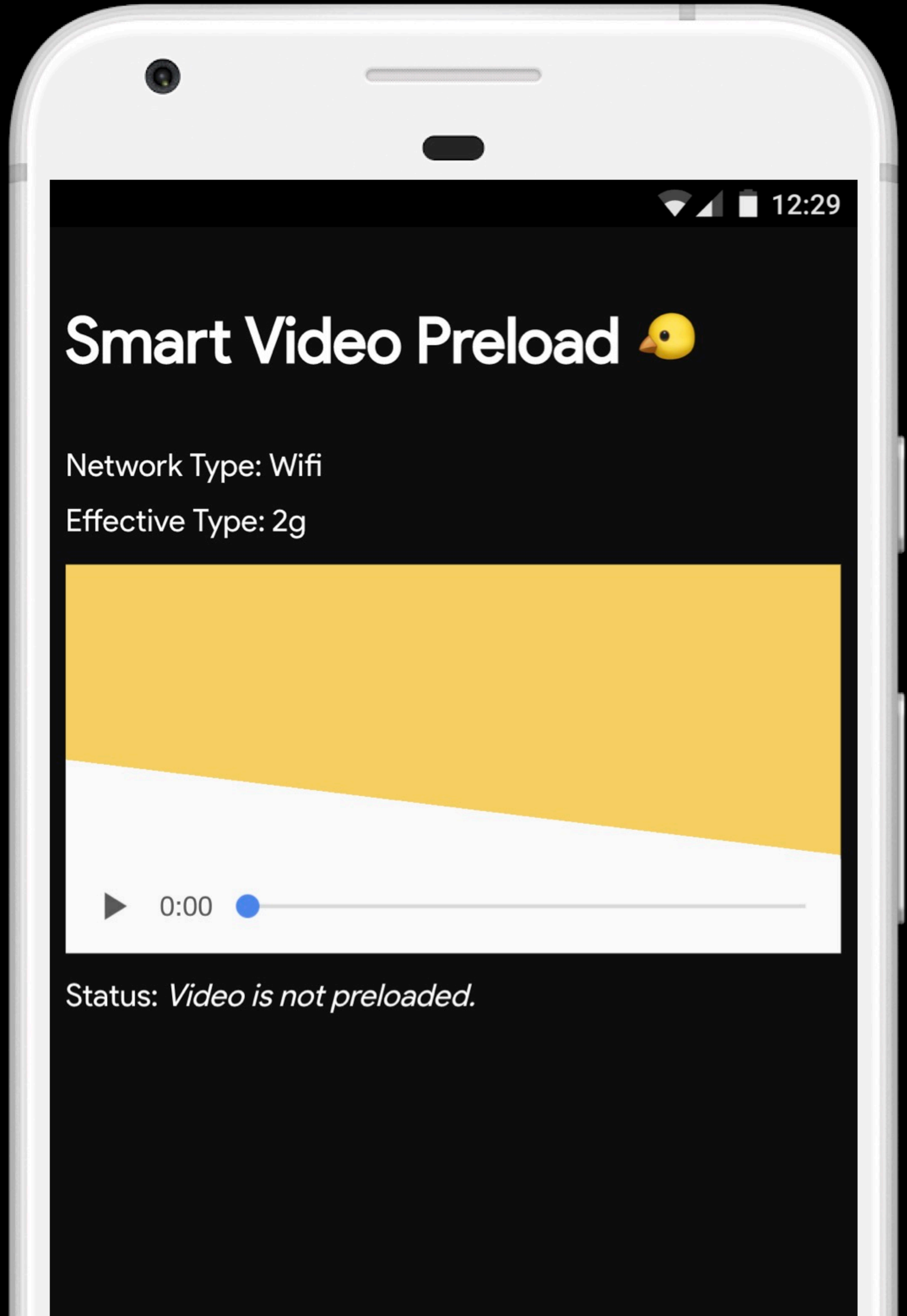


Smart Video Preload 🐥

Network Type: Wifi

Status: *Playing muted video...*

AFTER



Smart Video Preload 🐥

Network Type: Wifi
Effective Type: 2g

Status: *Video is not preloaded.*

MINIMIZE FOIT

DOWNLOAD FEWER THINGS

```
@font-face {
  font-family: 'Roboto';
  font-weight: 400;
  src: local('Roboto'), url(https://font.woff2) format(‘woff2');
}
```

MINIMIZE FOIT
DOWNLOAD EVEN FEWER THINGS

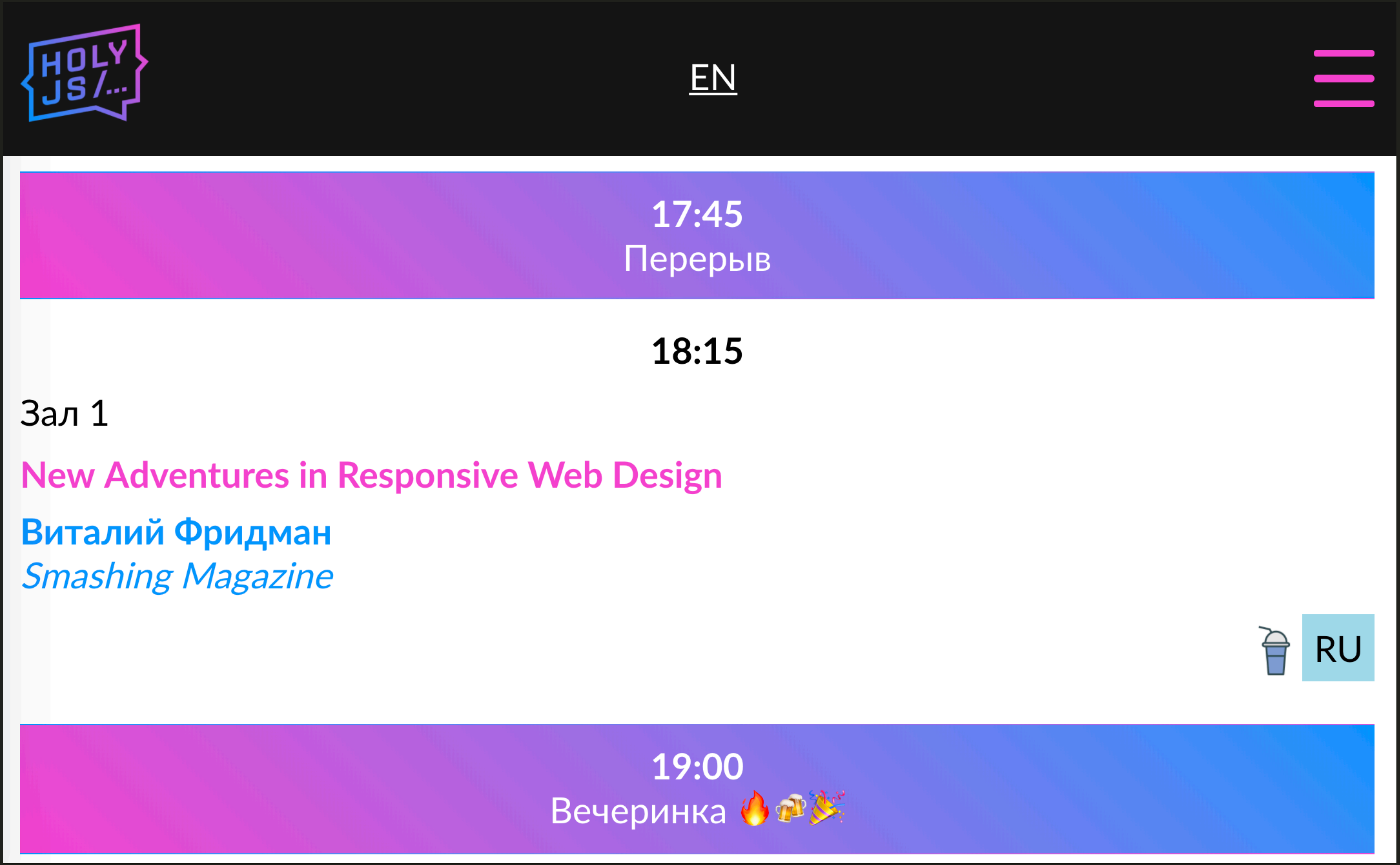
MINIMIZE FOIT

DOWNLOAD EVEN FEWER THINGS

```
@font-face {
  font-family: 'Roboto';
  font-weight: 400;
  src: local('Roboto'), url(https://font.woff2) format('woff2');
  unicode-range: U+0-A0;
}
```

- CSS:

```
@font-face {
  font-family: 'Open Sans', Arial, serif;
  src: local('Open Sans'),
       url(/fonts/open-sans-latin.woff2) format('woff2');
  unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02C6,
                 U+02DA, U+02DC, U+2000-206F, U+2074, U+20AC,
                 U+2212, U+2215, U+E0FF, U+EFFD, U+F000;
}

@font-face {
  font-family: 'Open Sans', Arial, serif;
  src: local('Open Sans'),
       url(/fonts/open-sans-cyrillic.woff2) format('woff2');
  unicode-range: U+0400–U+04FF, U+0500–U+052F;
}
```



EN

☰

17:45
Перерыв

18:15

Зал 1

New Adventures in Responsive Web Design

Виталий Фридман
Smashing Magazine

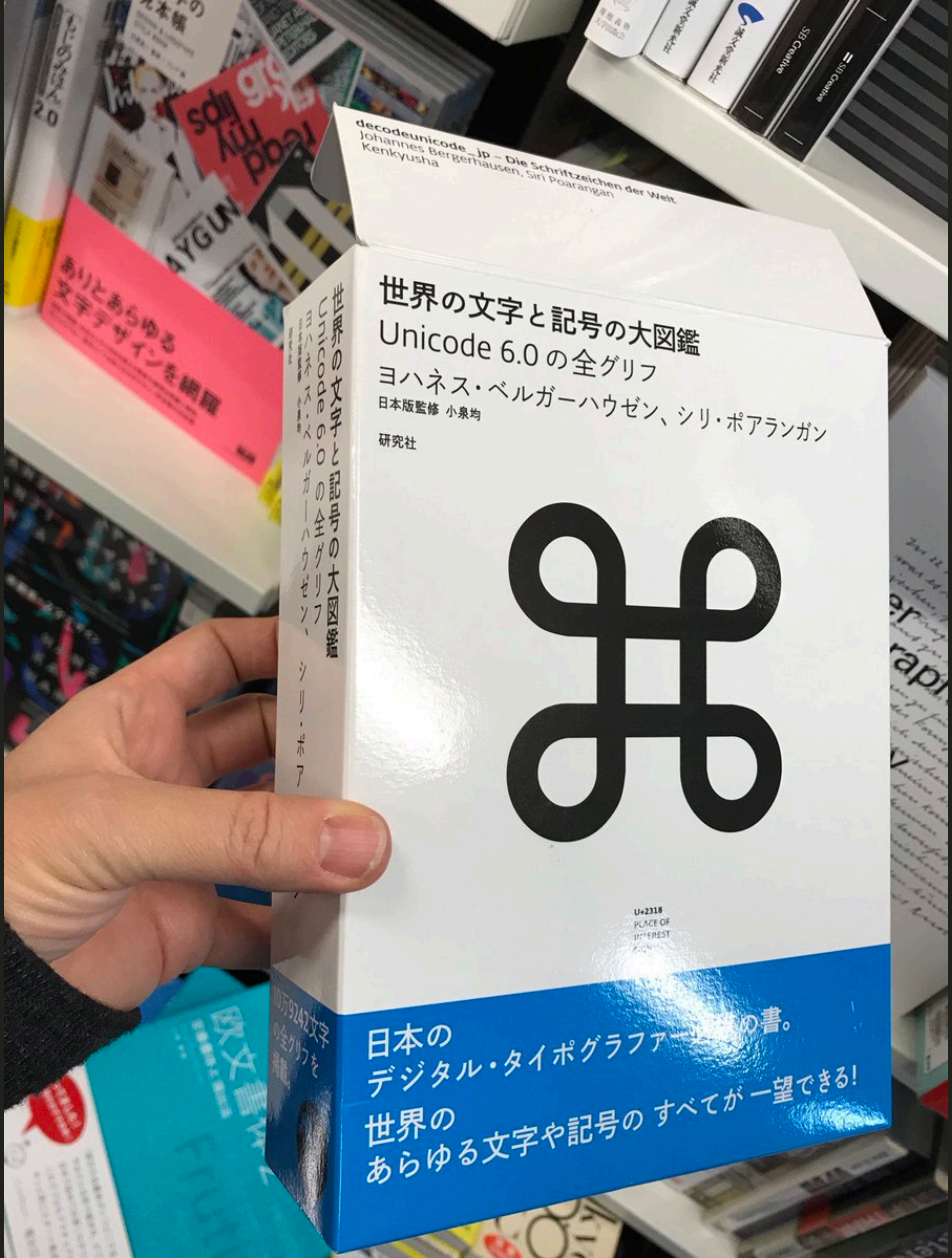
🥤 RU

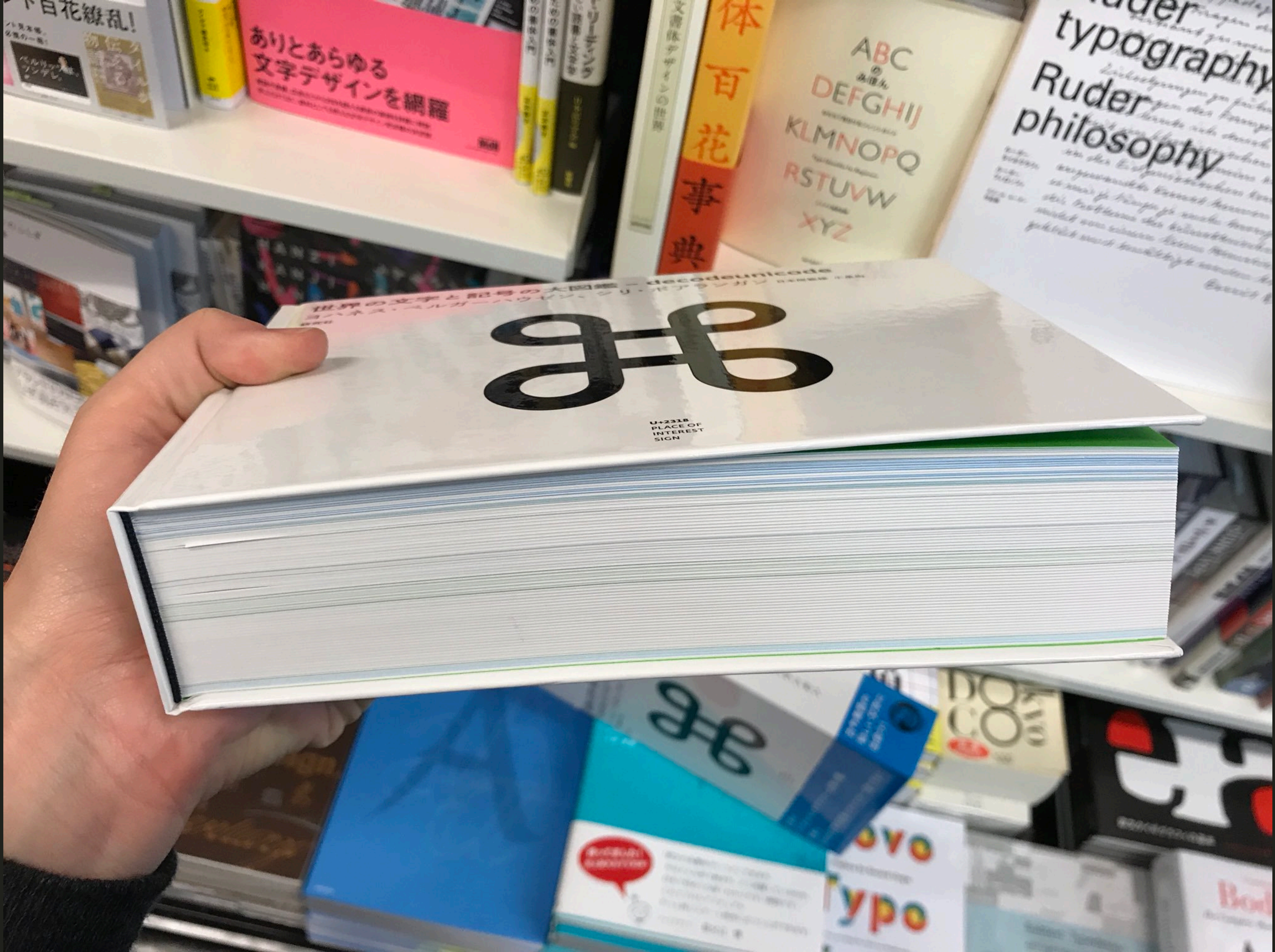
19:00
Вечеринка 🔥🍻🎉

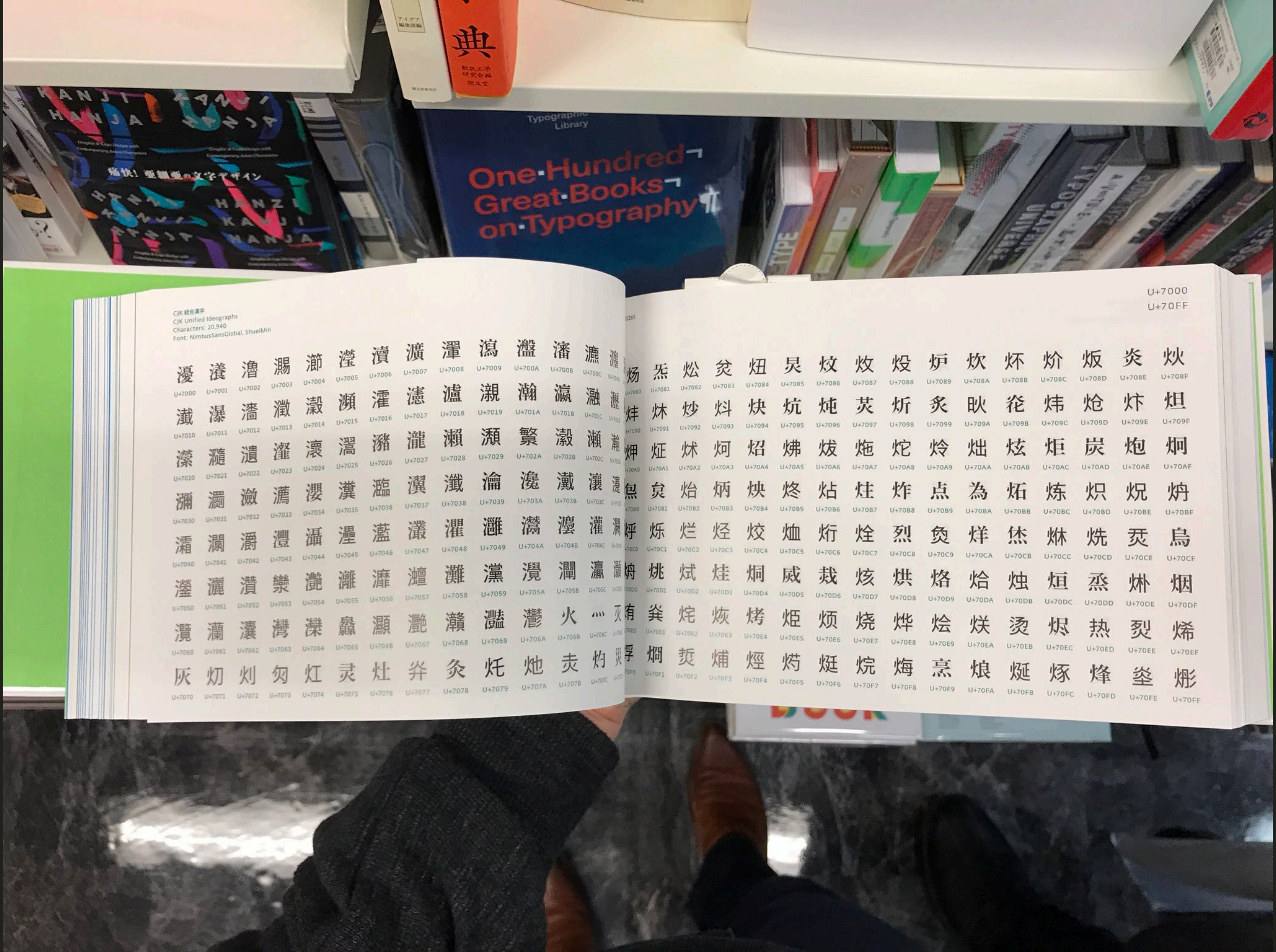
19:00

Вечеринка 🔥🍻🎉

🍻


decodeunicode_jp – Die Schriftzeichen der Welt
Johannes Bergerhausen, Siri Poarangan
Kenkyusha
世界の文字と記号の大図鑑
Unicode 6.0 の全グリフ
ヨハネス・ベルガーハウゼン、シリ・ポアランガン
日本版監修 小泉均
研究社
U+2318
PLACE OF
INTEREST
SIGN
日本の
デジタル・タイポグラファー
の書。
世界の
あらゆる文字や記号の すべてが 一望できる!
世界の文字と記号の大図鑑
Unicode 6.0 の全グリフ
ヨハネス・ベルガーハウゼン、シリ・ポア
文字
の全グリフを
掲載。

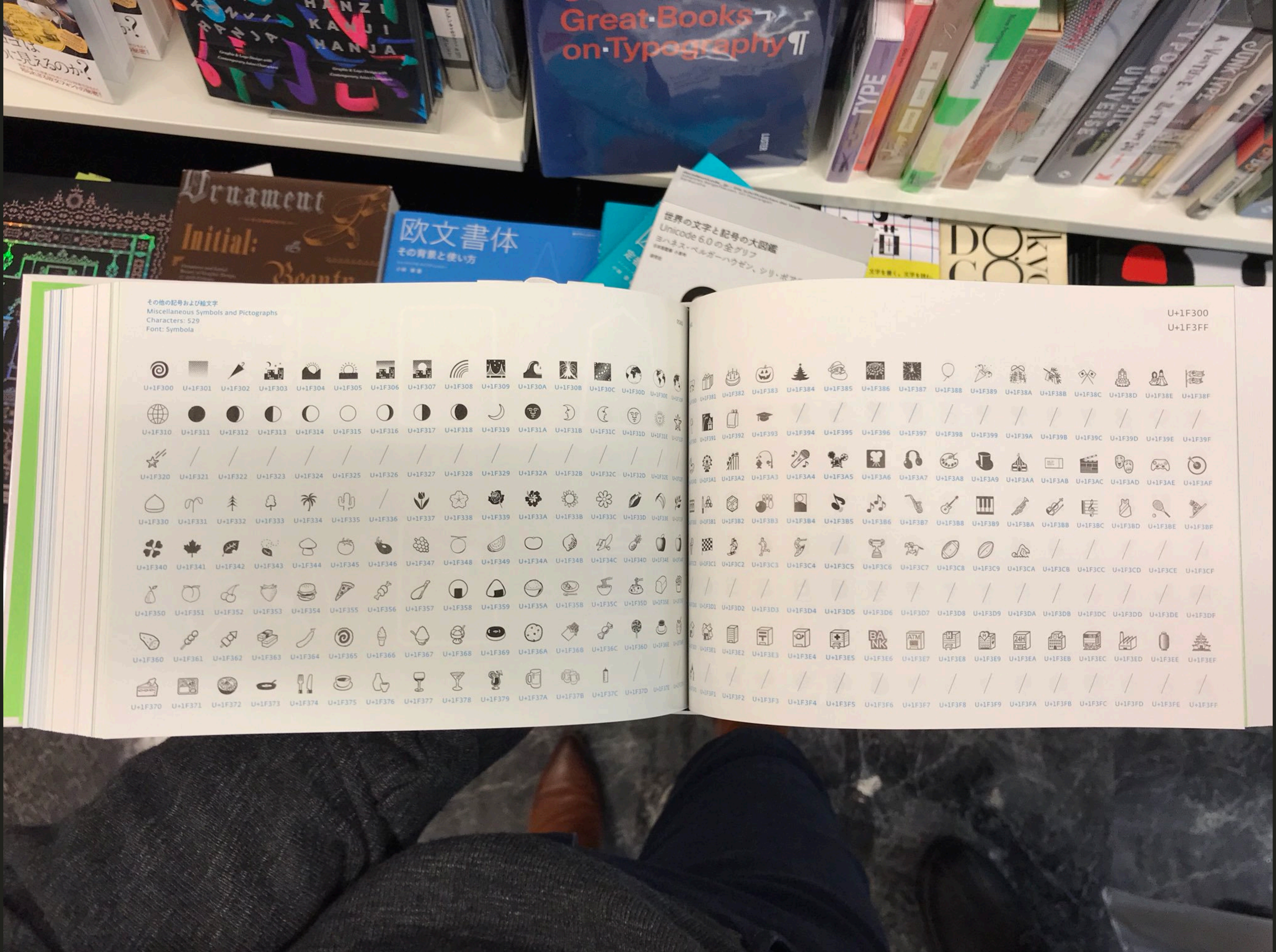

世界の文字と記号の大図鑑 — decodeunicode
U+2318
PLACE OF
INTEREST
SIGN
ありとあらゆる
文字デザインを網羅
Ruder
typography
Ruder
philosophy
ABC
のみほん
DEFGHIJ
KLMNOPQ
RSTUVW
XYZ
Typo


KANJI HANZI HANJA
痛快！亜細亜の文字デザイン
典
Typographic Library
One·Hundred Great·Books on·Typography
TYPE
BOOK

CJK 統合漢字
CJK Unified Ideographs
Characters: 20,940
Font: NimbusSansGlobal, ShueiMin

瀀 U+7000	瀁 U+7001	瀂 U+7002	瀃 U+7003	瀄 U+7004	瀅 U+7005	瀆 U+7006	瀇 U+7007	瀈 U+7008	瀉 U+7009	瀊 U+700A	瀋 U+700B	瀌 U+700C	瀍 U+700D
瀐 U+7010	瀑 U+7011	瀒 U+7012	瀓 U+7013	瀔 U+7014	瀕 U+7015	瀖 U+7016	瀗 U+7017	瀘 U+7018	瀙 U+7019	瀚 U+701A	瀛 U+701B	瀜 U+701C	瀝 U+701D
瀠 U+7020	瀡 U+7021	瀢 U+7022	瀣 U+7023	瀤 U+7024	瀥 U+7025	瀦 U+7026	瀧 U+7027	瀨 U+7028	瀩 U+7029	瀪 U+702A	瀫 U+702B	瀬 U+702C	瀭 U+702D
瀰 U+7030	瀱 U+7031	瀲 U+7032	瀳 U+7033	瀴 U+7034	瀵 U+7035	瀶 U+7036	瀷 U+7037	瀸 U+7038	瀹 U+7039	瀺 U+703A	瀻 U+703B	瀼 U+703C	瀽 U+703D
灀 U+7040	灁 U+7041	灂 U+7042	灃 U+7043	灄 U+7044	灅 U+7045	灆 U+7046	灇 U+7047	灈 U+7048	灉 U+7049	灊 U+704A	灋 U+704B	灌 U+704C	灍 U+704D
灐 U+7050	灑 U+7051	灒 U+7052	灓 U+7053	灔 U+7054	灕 U+7055	灖 U+7056	灗 U+7057	灘 U+7058	灙 U+7059	灚 U+705A	灛 U+705B	灜 U+705C	灝 U+705D
灠 U+7060	灡 U+7061	灢 U+7062	灣 U+7063	灤 U+7064	灥 U+7065	灦 U+7066	灧 U+7067	灨 U+7068	灩 U+7069	灪 U+706A	火 U+706B	灬 U+706C	灭 U+706D
灰 U+7070	灱 U+7071	灲 U+7072	灳 U+7073	灴 U+7074	灵 U+7075	灶 U+7076	灷 U+7077	灸 U+7078	灹 U+7079	灺 U+707A	灻 U+707B	灼 U+707C	災 U+707D

炀 U+7080	炁 U+7081	炂 U+7082	炃 U+7083	炄 U+7084	炅 U+7085	炆 U+7086	炇 U+7087	炈 U+7088	炉 U+7089	炊 U+708A	炋 U+708B	炌 U+708C	炍 U+708D	炎 U+708E	炏 U+708F
炐 U+7090	炑 U+7091	炒 U+7092	炓 U+7093	炔 U+7094	炕 U+7095	炖 U+7096	炗 U+7097	炘 U+7098	炙 U+7099	炚 U+709A	炛 U+709B	炜 U+709C	炝 U+709D	炞 U+709E	炟 U+709F
炠 U+70A0	炡 U+70A1	炢 U+70A2	炣 U+70A3	炤 U+70A4	炥 U+70A5	炦 U+70A6	炧 U+70A7	炨 U+70A8	炩 U+70A9	炪 U+70AA	炫 U+70AB	炬 U+70AC	炭 U+70AD	炮 U+70AE	炯 U+70AF
炰 U+70B0	炱 U+70B1	炲 U+70B2	炳 U+70B3	炴 U+70B4	炵 U+70B5	炶 U+70B6	炷 U+70B7	炸 U+70B8	点 U+70B9	為 U+70BA	炻 U+70BB	炼 U+70BC	炽 U+70BD	炾 U+70BE	炿 U+70BF
烀 U+70C0	烁 U+70C1	烂 U+70C2	烃 U+70C3	烄 U+70C4	烅 U+70C5	烆 U+70C6	烇 U+70C7	烈 U+70C8	烉 U+70C9	烊 U+70CA	烋 U+70CB	烌 U+70CC	烍 U+70CD	烎 U+70CE	烏 U+70CF
烐 U+70D0	烑 U+70D1	烒 U+70D2	烓 U+70D3	烔 U+70D4	烕 U+70D5	烖 U+70D6	烗 U+70D7	烘 U+70D8	烙 U+70D9	烚 U+70DA	烛 U+70DB	烜 U+70DC	烝 U+70DD	烞 U+70DE	烟 U+70DF
烠 U+70E0	烡 U+70E1	烢 U+70E2	烣 U+70E3	烤 U+70E4	烥 U+70E5	烦 U+70E6	烧 U+70E7	烨 U+70E8	烩 U+70E9	烪 U+70EA	烫 U+70EB	烬 U+70EC	热 U+70ED	烮 U+70EE	烯 U+70EF
烰 U+70F0	烱 U+70F1	烲 U+70F2	烳 U+70F3	烴 U+70F4	烵 U+70F5	烶 U+70F6	烷 U+70F7	烸 U+70F8	烹 U+70F9	烺 U+70FA	烻 U+70FB	烼 U+70FC	烽 U+70FD	烾 U+70FE	烿 U+70FF



その他の記号および絵文字
Miscellaneous Symbols and Pictographs
Characters: 529
Font: Symbola

U+1F300
U+1F3FF

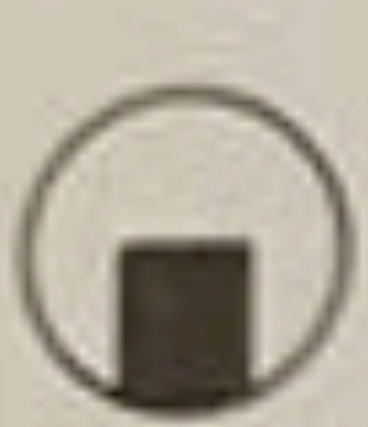
🌀 U+1F300	🌁 U+1F301	🌂 U+1F302	🌃 U+1F303	🌄 U+1F304	🌅 U+1F305	🌆 U+1F306	🌇 U+1F307	🌈 U+1F308	🌉 U+1F309	🌊 U+1F30A	🌋 U+1F30B	🌌 U+1F30C	🌍 U+1F30D	🌎 U+1F30E	🌏 U+1F30F
🌐 U+1F310	🌑 U+1F311	🌒 U+1F312	🌓 U+1F313	🌔 U+1F314	🌕 U+1F315	🌖 U+1F316	🌗 U+1F317	🌘 U+1F318	🌙 U+1F319	🌚 U+1F31A	🌛 U+1F31B	🌜 U+1F31C	🌝 U+1F31D	🌞 U+1F31E	🌟 U+1F31F
🌠 U+1F320	/ U+1F321	/ U+1F322	/ U+1F323	/ U+1F324	/ U+1F325	/ U+1F326	/ U+1F327	/ U+1F328	/ U+1F329	/ U+1F32A	/ U+1F32B	/ U+1F32C	/ U+1F32D	/ U+1F32E	/ U+1F32F
🌰 U+1F330	🌱 U+1F331	🌲 U+1F332	🌳 U+1F333	🌴 U+1F334	🌵 U+1F335	/ U+1F336	🌷 U+1F337	🌸 U+1F338	🌹 U+1F339	🌺 U+1F33A	🌻 U+1F33B	🌼 U+1F33C	🌽 U+1F33D	🌾 U+1F33E	🌿 U+1F33F
🍀 U+1F340	🍁 U+1F341	🍂 U+1F342	🍃 U+1F343	🍄 U+1F344	🍅 U+1F345	🍆 U+1F346	🍇 U+1F347	🍈 U+1F348	🍉 U+1F349	🍊 U+1F34A	🍋 U+1F34B	🍌 U+1F34C	🍍 U+1F34D	🍎 U+1F34E	🍏 U+1F34F
🍐 U+1F350	🍑 U+1F351	🍒 U+1F352	🍓 U+1F353	🍔 U+1F354	🍕 U+1F355	🍖 U+1F356	🍗 U+1F357	🍘 U+1F358	🍙 U+1F359	🍚 U+1F35A	🍛 U+1F35B	🍜 U+1F35C	🍝 U+1F35D	🍞 U+1F35E	🍟 U+1F35F
🍠 U+1F360	🍡 U+1F361	🍢 U+1F362	🍣 U+1F363	🍤 U+1F364	🍥 U+1F365	🍦 U+1F366	🍧 U+1F367	🍨 U+1F368	🍩 U+1F369	🍪 U+1F36A	🍫 U+1F36B	🍬 U+1F36C	🍭 U+1F36D	🍮 U+1F36E	🍯 U+1F36F
🍰 U+1F370	🍱 U+1F371	🍲 U+1F372	🍳 U+1F373	🍴 U+1F374	🍵 U+1F375	🍶 U+1F376	🍷 U+1F377	🍸 U+1F378	🍹 U+1F379	🍺 U+1F37A	🍻 U+1F37B	🍼 U+1F37C	/ U+1F37D	/ U+1F37E	/ U+1F37F
🎀 U+1F380	🎁 U+1F381	🎂 U+1F382	🎃 U+1F383	🎄 U+1F384	🎅 U+1F385	🎆 U+1F386	🎇 U+1F387	🎈 U+1F388	🎉 U+1F389	🎊 U+1F38A	🎋 U+1F38B	🎌 U+1F38C	🎍 U+1F38D	🎎 U+1F38E	🎏 U+1F38F
🎐 U+1F390	🎑 U+1F391	🎒 U+1F392	🎓 U+1F393	/ U+1F394	/ U+1F395	/ U+1F396	/ U+1F397	/ U+1F398	/ U+1F399	/ U+1F39A	/ U+1F39B	/ U+1F39C	/ U+1F39D	/ U+1F39E	/ U+1F39F
🎠 U+1F3A0	🎡 U+1F3A1	🎢 U+1F3A2	🎣 U+1F3A3	🎤 U+1F3A4	🎥 U+1F3A5	🎦 U+1F3A6	🎧 U+1F3A7	🎨 U+1F3A8	🎩 U+1F3A9	🎪 U+1F3AA	🎫 U+1F3AB	🎬 U+1F3AC	🎭 U+1F3AD	🎮 U+1F3AE	🎯 U+1F3AF
🎰 U+1F3B0	🎱 U+1F3B1	🎲 U+1F3B2	🎳 U+1F3B3	🎴 U+1F3B4	🎵 U+1F3B5	🎶 U+1F3B6	🎷 U+1F3B7	🎸 U+1F3B8	🎹 U+1F3B9	🎺 U+1F3BA	🎻 U+1F3BB	🎼 U+1F3BC	🎽 U+1F3BD	🎾 U+1F3BE	🎿 U+1F3BF
🏀 U+1F3C0	🏁 U+1F3C1	🏂 U+1F3C2	🏃 U+1F3C3	🏄 U+1F3C4	/ U+1F3C5	🏆 U+1F3C6	🏇 U+1F3C7	🏈 U+1F3C8	🏉 U+1F3C9	🏊 U+1F3CA	/ U+1F3CB	/ U+1F3CC	/ U+1F3CD	/ U+1F3CE	/ U+1F3CF
/ U+1F3D0	/ U+1F3D1	/ U+1F3D2	/ U+1F3D3	/ U+1F3D4	/ U+1F3D5	/ U+1F3D6	/ U+1F3D7	/ U+1F3D8	/ U+1F3D9	/ U+1F3DA	/ U+1F3DB	/ U+1F3DC	/ U+1F3DD	/ U+1F3DE	/ U+1F3DF
🏠 U+1F3E0	🏡 U+1F3E1	🏢 U+1F3E2	🏣 U+1F3E3	🏤 U+1F3E4	🏥 U+1F3E5	🏦 U+1F3E6	🏧 U+1F3E7	🏨 U+1F3E8	🏩 U+1F3E9	🏪 U+1F3EA	🏫 U+1F3EB	🏬 U+1F3EC	🏭 U+1F3ED	🏮 U+1F3EE	🏯 U+1F3EF
🏰 U+1F3F0	/ U+1F3F1	/ U+1F3F2	/ U+1F3F3	/ U+1F3F4	/ U+1F3F5	/ U+1F3F6	/ U+1F3F7	/ U+1F3F8	/ U+1F3F9	/ U+1F3FA	/ U+1F3FB	/ U+1F3FC	/ U+1F3FD	/ U+1F3FE	/ U+1F3FF

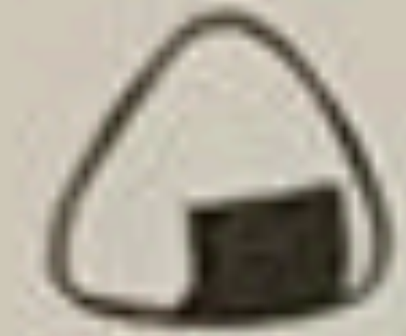
Font: Symbola

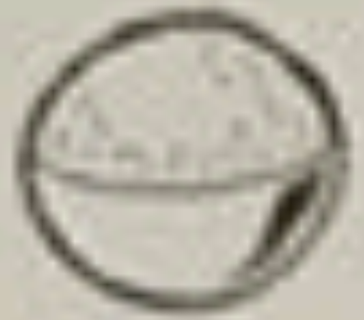
🌀 U+1F300	🌁 U+1F301	🌂 U+1F302	🌃 U+1F303	🌄 U+1F304	🌅 U+1F305	🌆 U+1F306	🌇 U+1F307	🌈 U+1F308	🌉 U+1F309	🌊 U+1F30A	🌋 U+1F30B	🌌 U+1F30C	🌍 U+1F30D	🌎 U+1F30E	🌏 U+1F30F
🌐 U+1F310	🌑 U+1F311	🌒 U+1F312	🌓 U+1F313	🌔 U+1F314	🌕 U+1F315	🌖 U+1F316	🌗 U+1F317	🌘 U+1F318	🌙 U+1F319	🌚 U+1F31A	🌛 U+1F31B	🌜 U+1F31C	🌝 U+1F31D	🌞 U+1F31E	🌟 U+1F31F
🌠 U+1F320	/ U+1F321	/ U+1F322	/ U+1F323	/ U+1F324	/ U+1F325	/ U+1F326	/ U+1F327	/ U+1F328	/ U+1F329	/ U+1F32A	/ U+1F32B	/ U+1F32C	/ U+1F32D	/ U+1F32E	/ U+1F32F
🌰 U+1F330	🌱 U+1F331	🌲 U+1F332	🌳 U+1F333	🌴 U+1F334	🌵 U+1F335	/ U+1F336	🌷 U+1F337	🌸 U+1F338	🌹 U+1F339	🌺 U+1F33A	🌻 U+1F33B	🌼 U+1F33C	🌽 U+1F33D	🌾 U+1F33E	🌿 U+1F33F
🍀 U+1F340	🍁 U+1F341	🍂 U+1F342	🍃 U+1F343	🍄 U+1F344	🍅 U+1F345	🍆 U+1F346	🍇 U+1F347	🍈 U+1F348	🍉 U+1F349	🍊 U+1F34A	🍋 U+1F34B	🍌 U+1F34C	🍍 U+1F34D	🍎 U+1F34E	🍏 U+1F34F
🍐 U+1F350	🍑 U+1F351	🍒 U+1F352	🍓 U+1F353	🍔 U+1F354	🍕 U+1F355	🍖 U+1F356	🍗 U+1F357	🍘 U+1F358	🍙 U+1F359	🍚 U+1F35A	🍛 U+1F35B	🍜 U+1F35C	🍝 U+1F35D	🍞 U+1F35E	🍟 U+1F35F
🍠 U+1F360	🍡 U+1F361	🍢 U+1F362	🍣 U+1F363	🍤 U+1F364	🍥 U+1F365	🍦 U+1F366	🍧 U+1F367	🍨 U+1F368	🍩 U+1F369	🍪 U+1F36A	🍫 U+1F36B	🍬 U+1F36C	🍭 U+1F36D	🍮 U+1F36E	🍯 U+1F36F
🍰 U+1F370	🍱 U+1F371	🍲 U+1F372	🍳 U+1F373	🍴 U+1F374	🍵 U+1F375	🍶 U+1F376	🍷 U+1F377	🍸 U+1F378	🍹 U+1F379	🍺 U+1F37A	🍻 U+1F37B	🍼 U+1F37C	/ U+1F37D	/ U+1F37E	/ U+1F37F

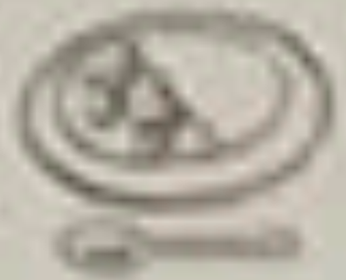
🎀 U+1F380	🎁 U+1F381	🎂 U+1F382	🎃 U+1F383
🎐 U+1F390	🎑 U+1F391	🎒 U+1F392	🎓 U+1F393
🎠 U+1F3A0	🎡 U+1F3A1	🎢 U+1F3A2	🎣 U+1F3A3
🎰 U+1F3B0	🎱 U+1F3B1	🎲 U+1F3B2	🎳 U+1F3B3
🏀 U+1F3C0	🏁 U+1F3C1	🏂 U+1F3C2	🏃 U+1F3C3
/ U+1F3D0	/ U+1F3D1	/ U+1F3D2	/ U+1F3D3
🏠 U+1F3E0	🏡 U+1F3E1	🏢 U+1F3E2	🏣 U+1F3E3
🏰 U+1F3F0	/ U+1F3F1	/ U+1F3F2	/ U+1F3F3

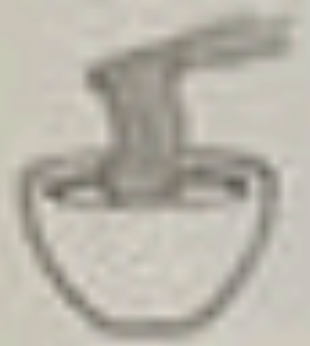
F357

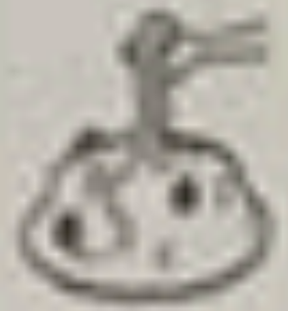

U+1F358

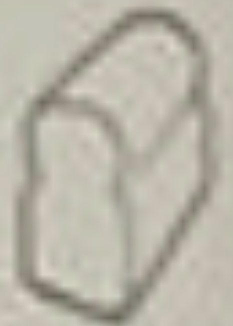

U+1F359


U+1F35A


U+1F35B

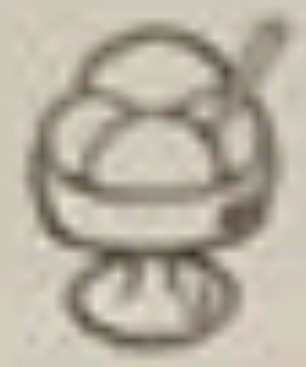

U+1F35C

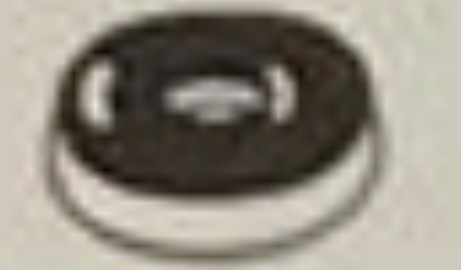

U+1F35D

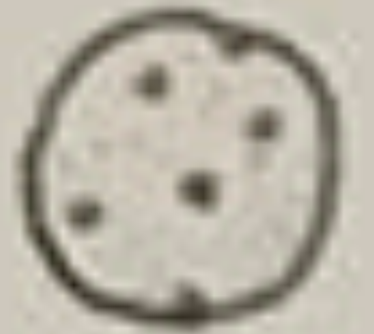

U+1F35E

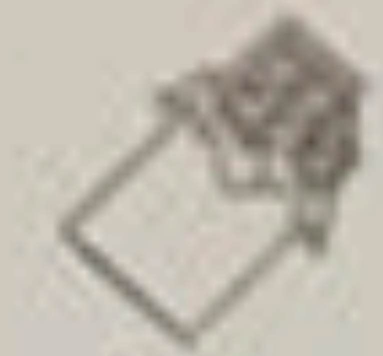
U+1F3

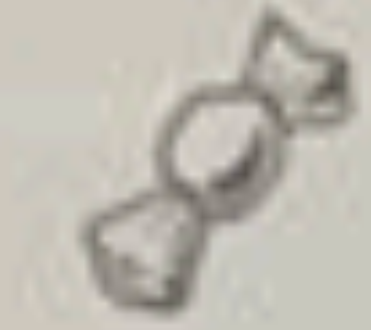
F367

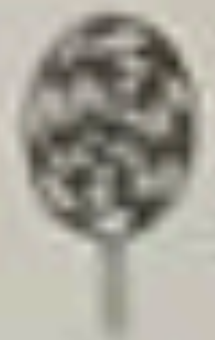

U+1F368

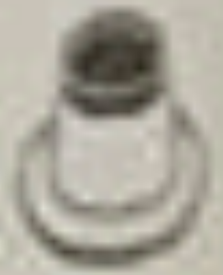

U+1F369


U+1F36A


U+1F36B

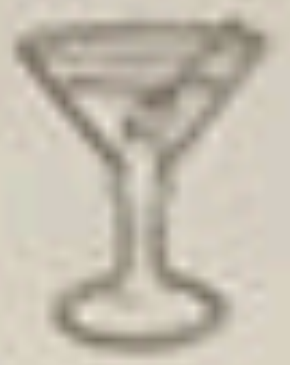

U+1F36C

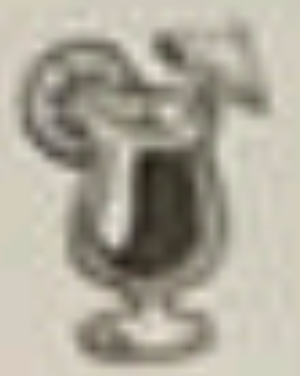

U+1F36D

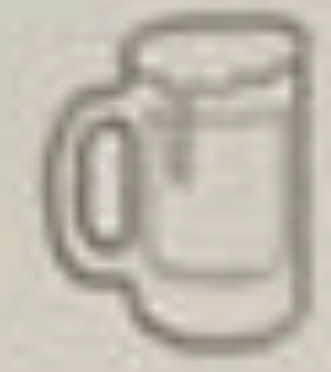

U+1F36E

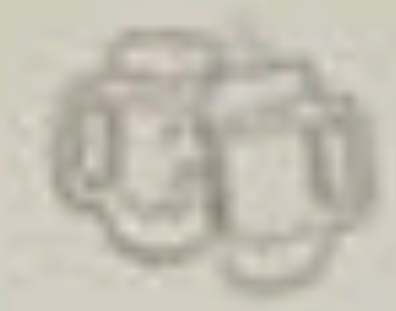
U+1F3

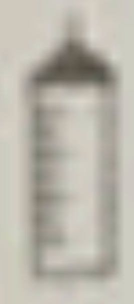
F377


U+1F378


U+1F379


U+1F37A


U+1F37B


U+1F37C

U+1F37D

U+1F37E

F3D0

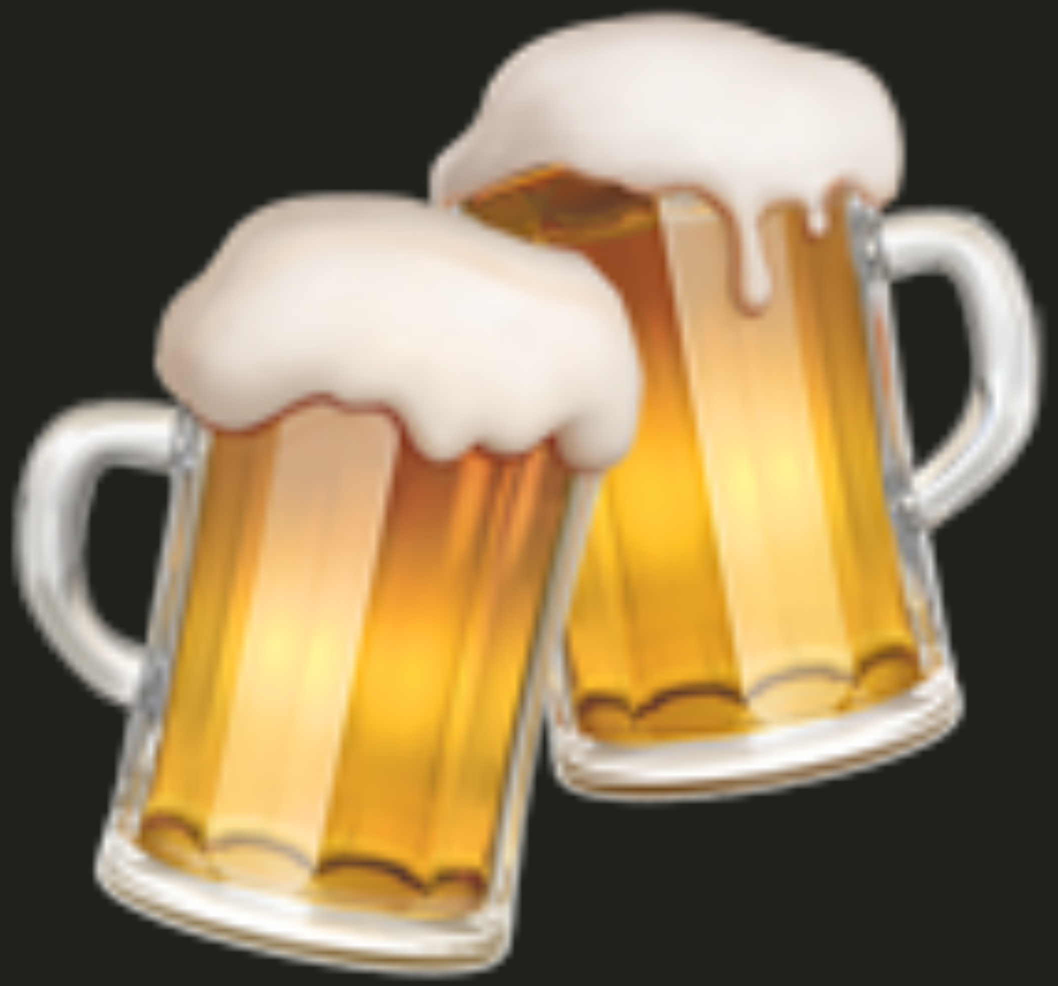
U+1F3D

F3E0

U+1F3E

F3F0

U+1F3F



U+1F37B

MINIMIZE FOIT

DOWNLOAD EVEN FEWER THINGS

```
@font-face {
  font-family: 'Roboto';
  font-weight: 400;
  src: local('Roboto'), url(https://font.woff2) format('woff2');
  unicode-range: U+0-A0;
}
```

MINIMIZE FOIT

DOWNLOAD EVEN FEWER THINGS EARLIER

MINIMIZE FOIT

DOWNLOAD EVEN FEWER THINGS EARLIER

```
@font-face {
  font-family: 'Roboto';
  font-weight: 400;
  src: local('Roboto'), url(https://font.woff2) format(‘woff2');
}
```

MINIMIZE FOIT

DOWNLOAD EVEN FEWER THINGS EARLIER

```
@font-face {
  font-family: 'Roboto';
  font-weight: 400;
  src: local('Roboto'), url(https://font.woff2) format(‘woff2');
}

<link rel=“preload" href=“https://font.woff2”
      as="font" type="font/woff2" crossorigin>
```

MINIMIZE FOIT

FONT-DISPLAY: OPTIONAL

MINIMIZE FOIT

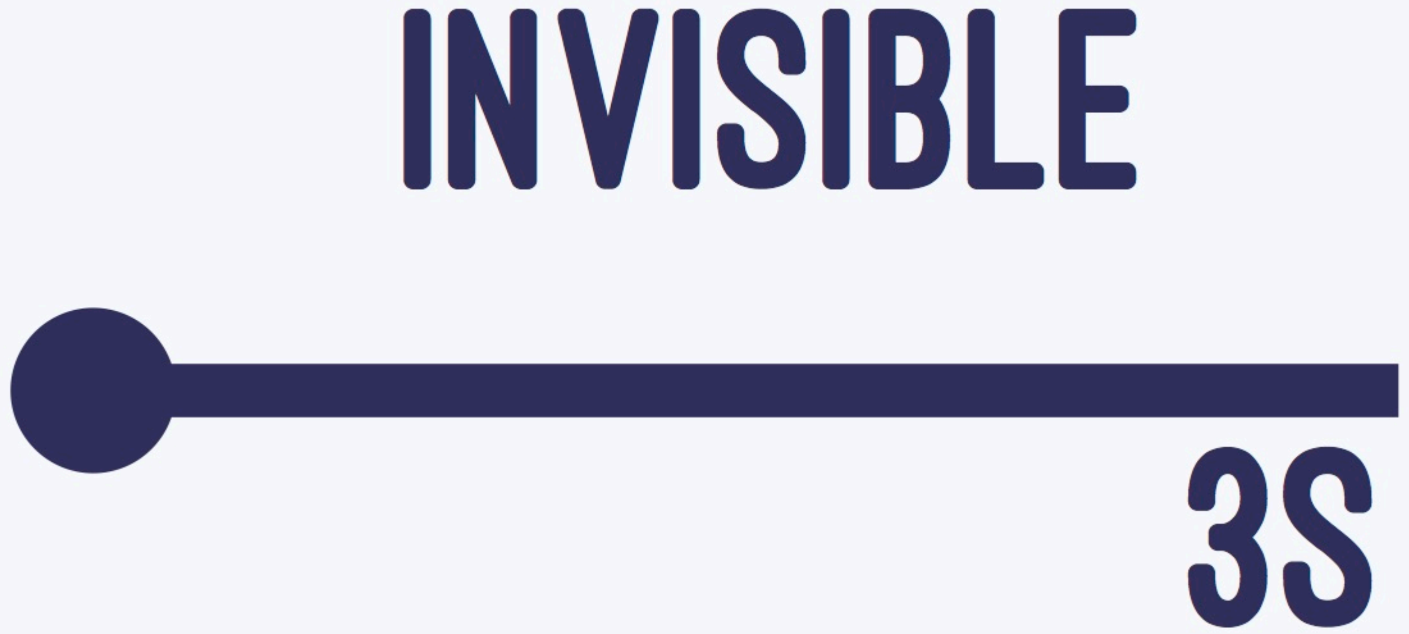
FONT-DISPLAY: OPTIONAL

```
@font-face {
  font-family: 'Roboto';
  font-weight: 400;
  src: local('Roboto'), url(https://font.woff2) format(‘woff2’);

}
```

MINIMIZE FOIT
FONT-DISPLAY: OPTIONAL

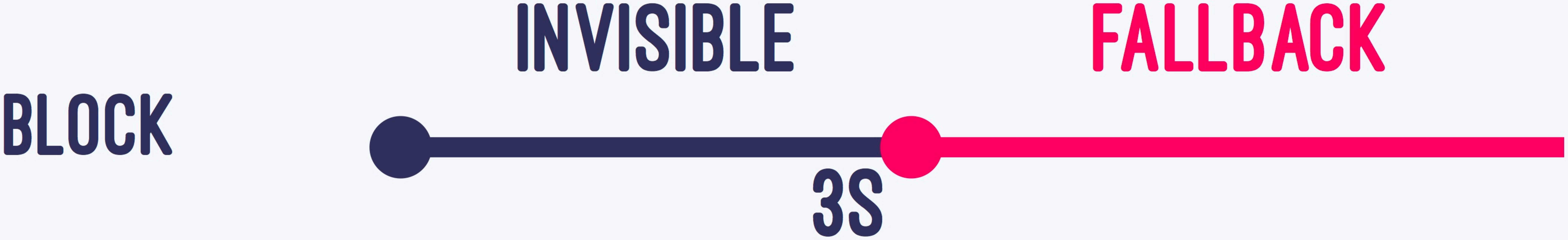
```
@font-face {
  font-family: 'Roboto';
  font-weight: 400;
  src: local('Roboto'), url(https://font.woff2) format(‘woff2’);
  font-display: optional;
}
```

BLOCK



INVISIBLE

3S



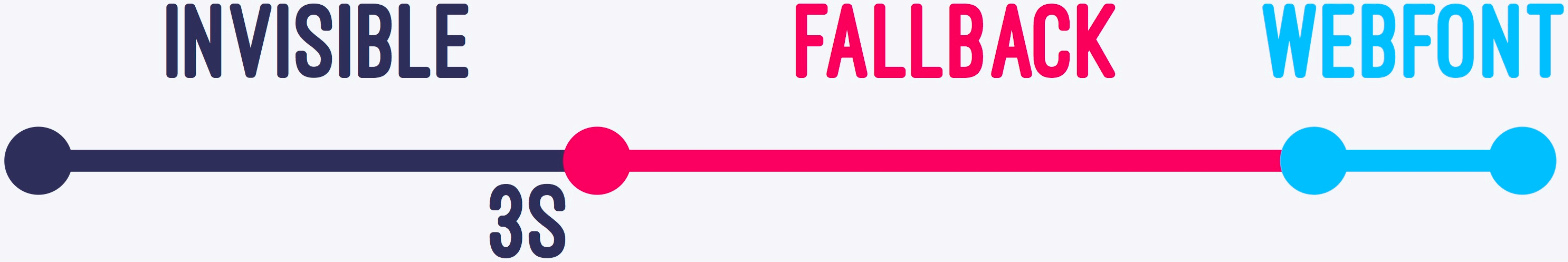
BLOCK

INVISIBLE

FALLBACK

3S

BLOCK

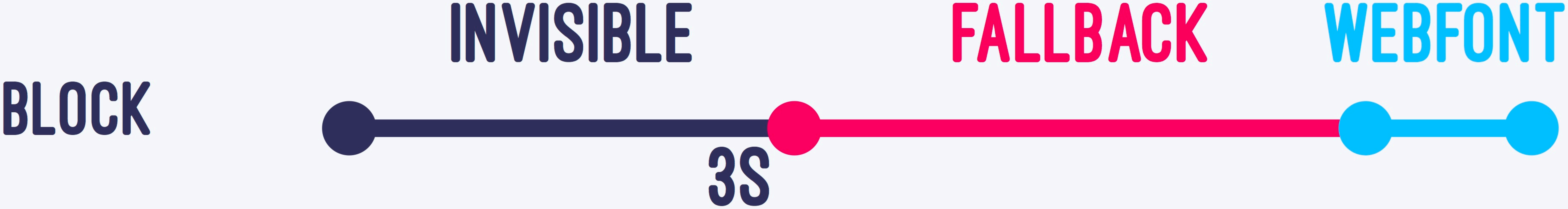


INVISIBLE

FALLBACK

WEBFONT

3S



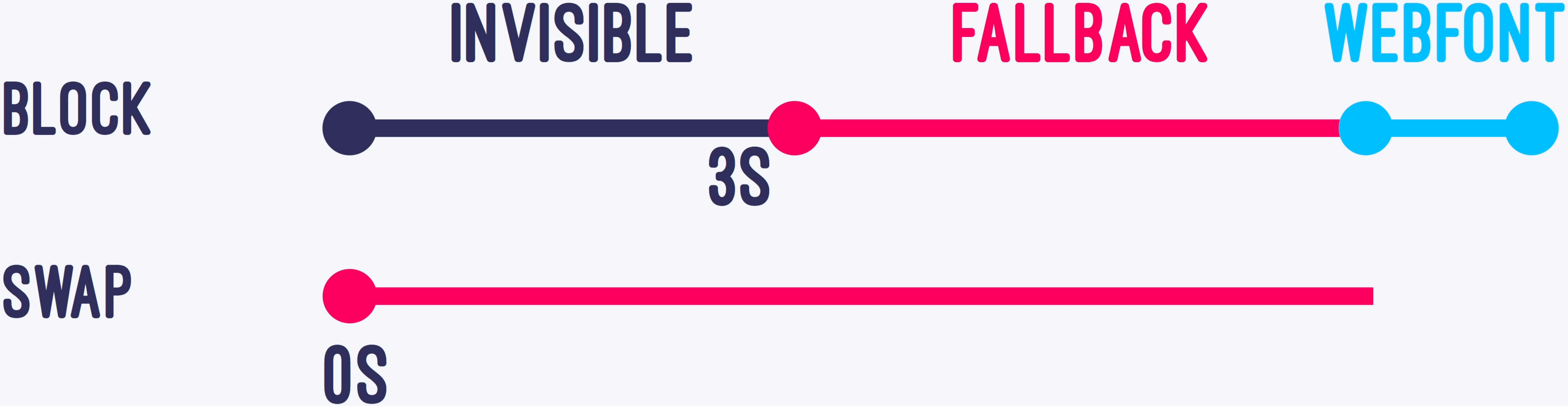
BLOCK

INVISIBLE

FALLBACK

WEBFONT

3S



INVISIBLE

FALLBACK

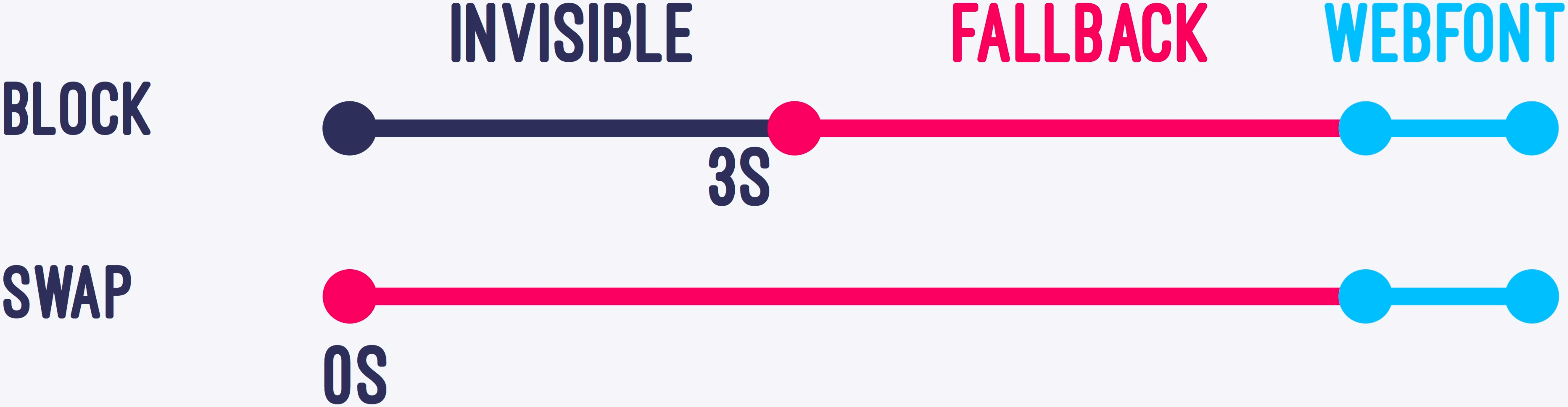
WEBFONT

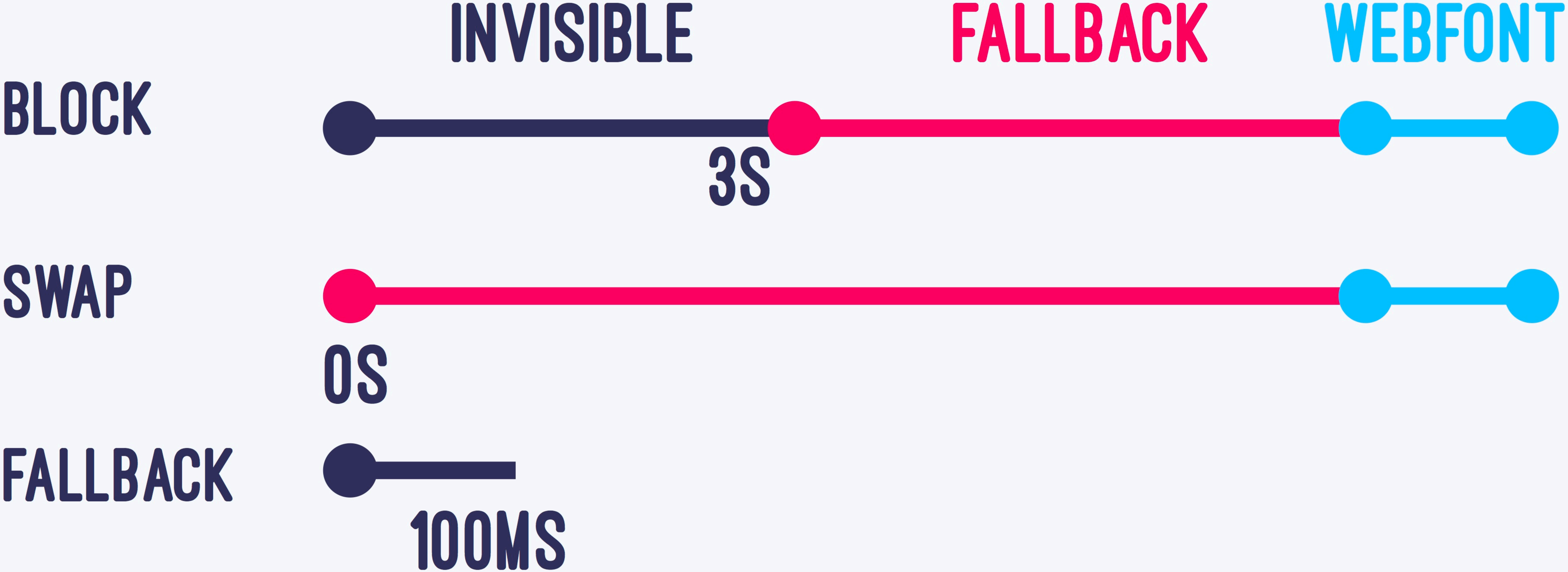
BLOCK

3S

SWAP

0S


INVISIBLE
FALLBACK
WEBFONT
BLOCK
3S
SWAP
0S



INVISIBLE

FALLBACK

WEBFONT

BLOCK

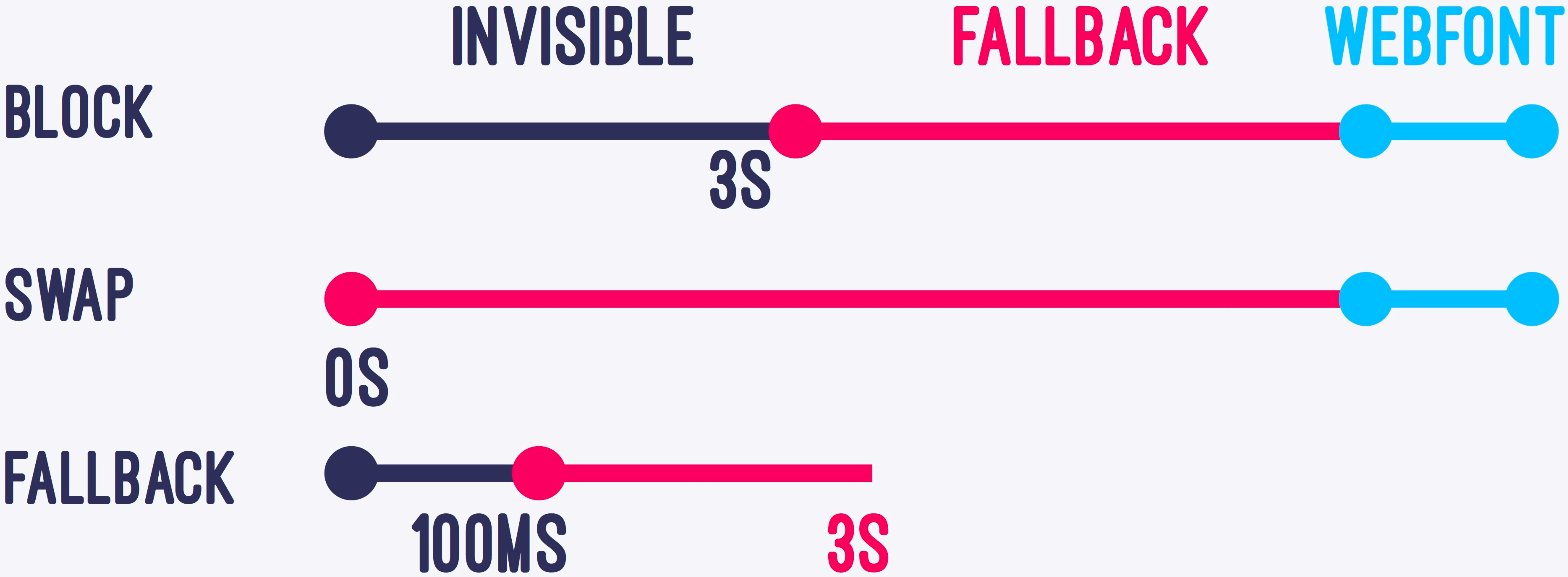
3S

SWAP

0S

FALLBACK

100MS



INVISIBLE

FALLBACK

WEBFONT

BLOCK

3S

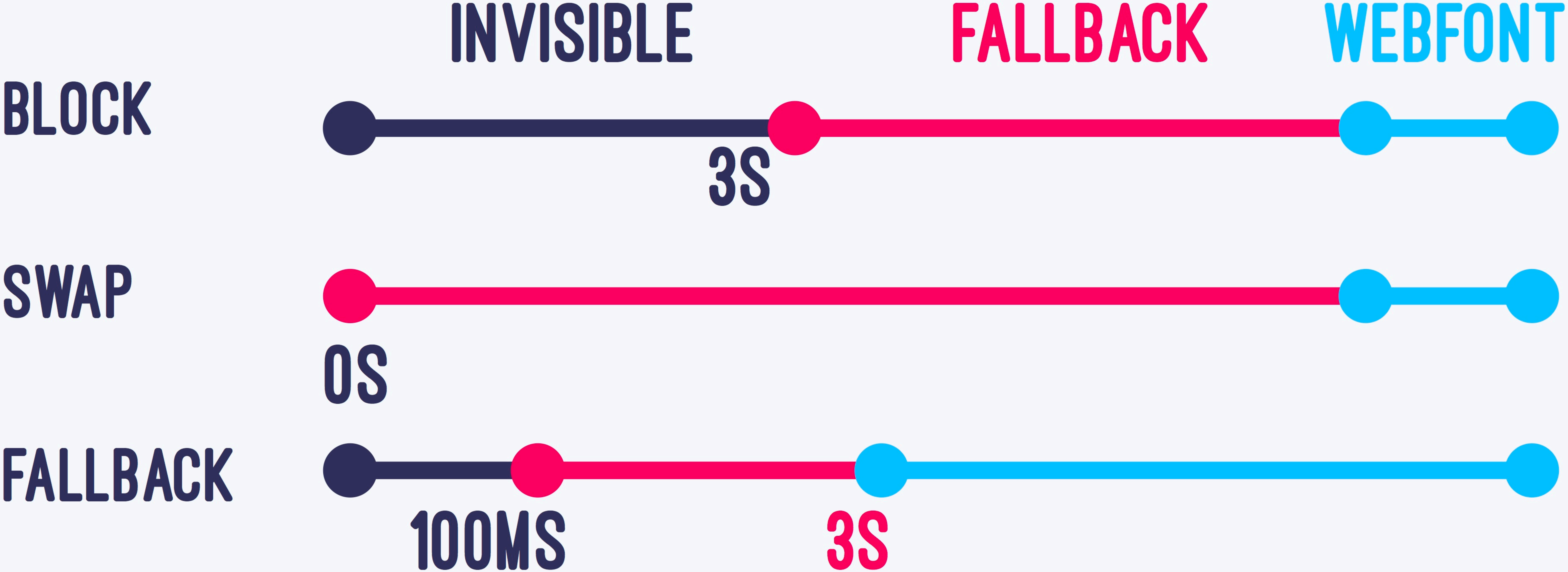
SWAP

0S

FALLBACK

100MS

3S



INVISIBLE

FALLBACK

WEBFONT

BLOCK

3S

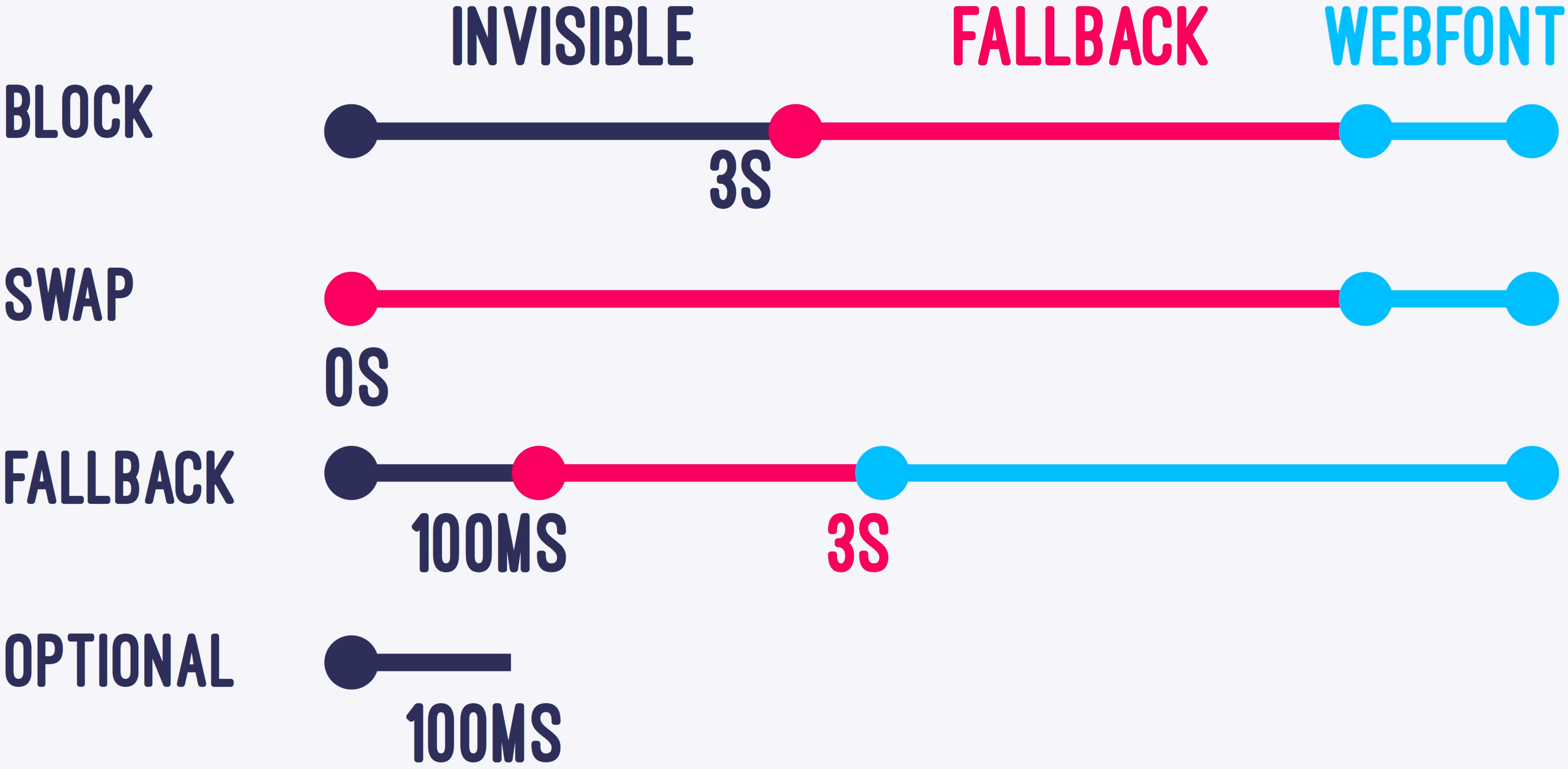
SWAP

0S

FALLBACK

100MS

3S



INVISIBLE

FALLBACK

WEBFONT

BLOCK

3S

SWAP

0S

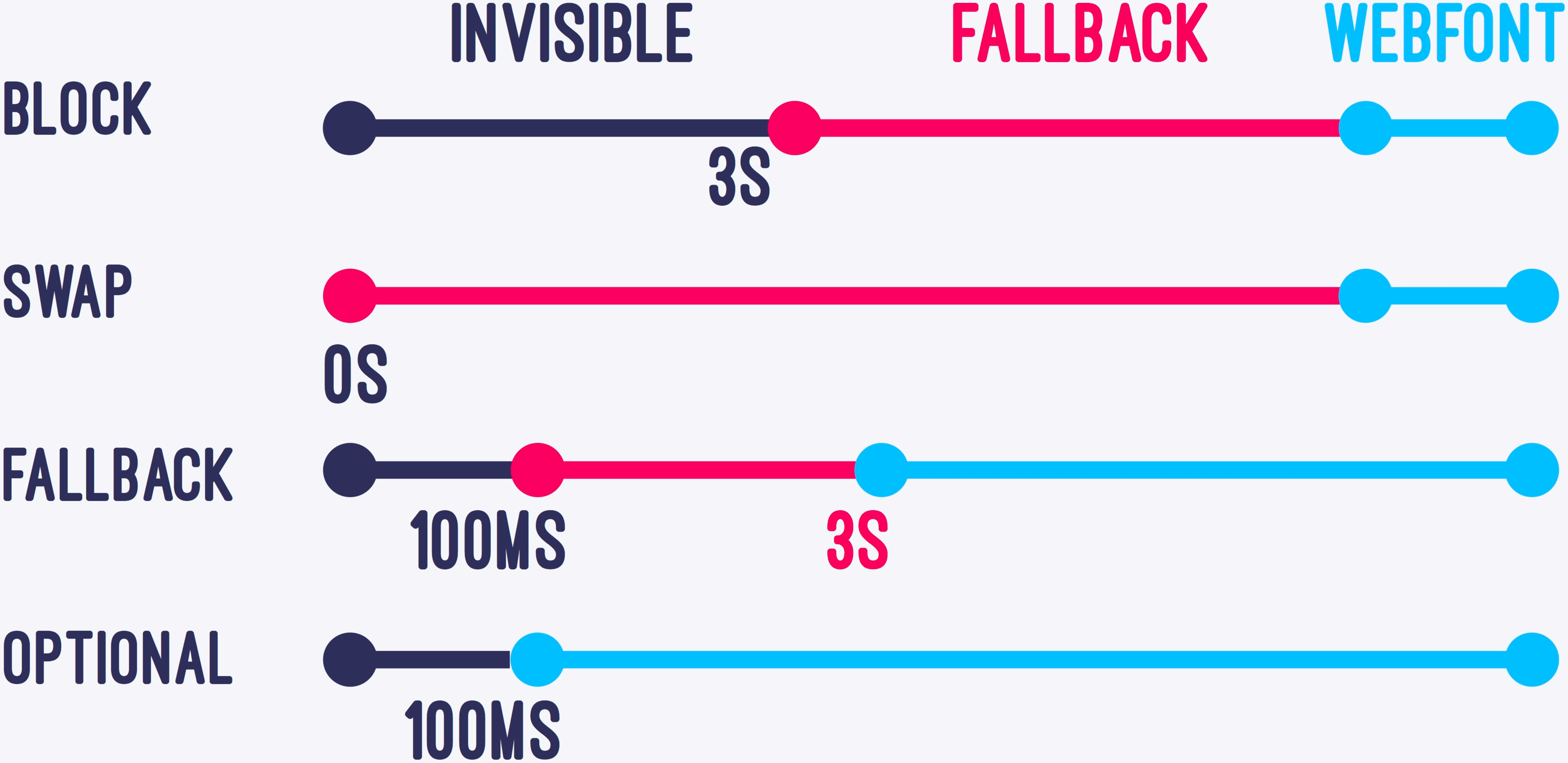
FALLBACK

100MS

3S

OPTIONAL

100MS



INVISIBLE

FALLBACK

WEBFONT

BLOCK

3S

SWAP

0S

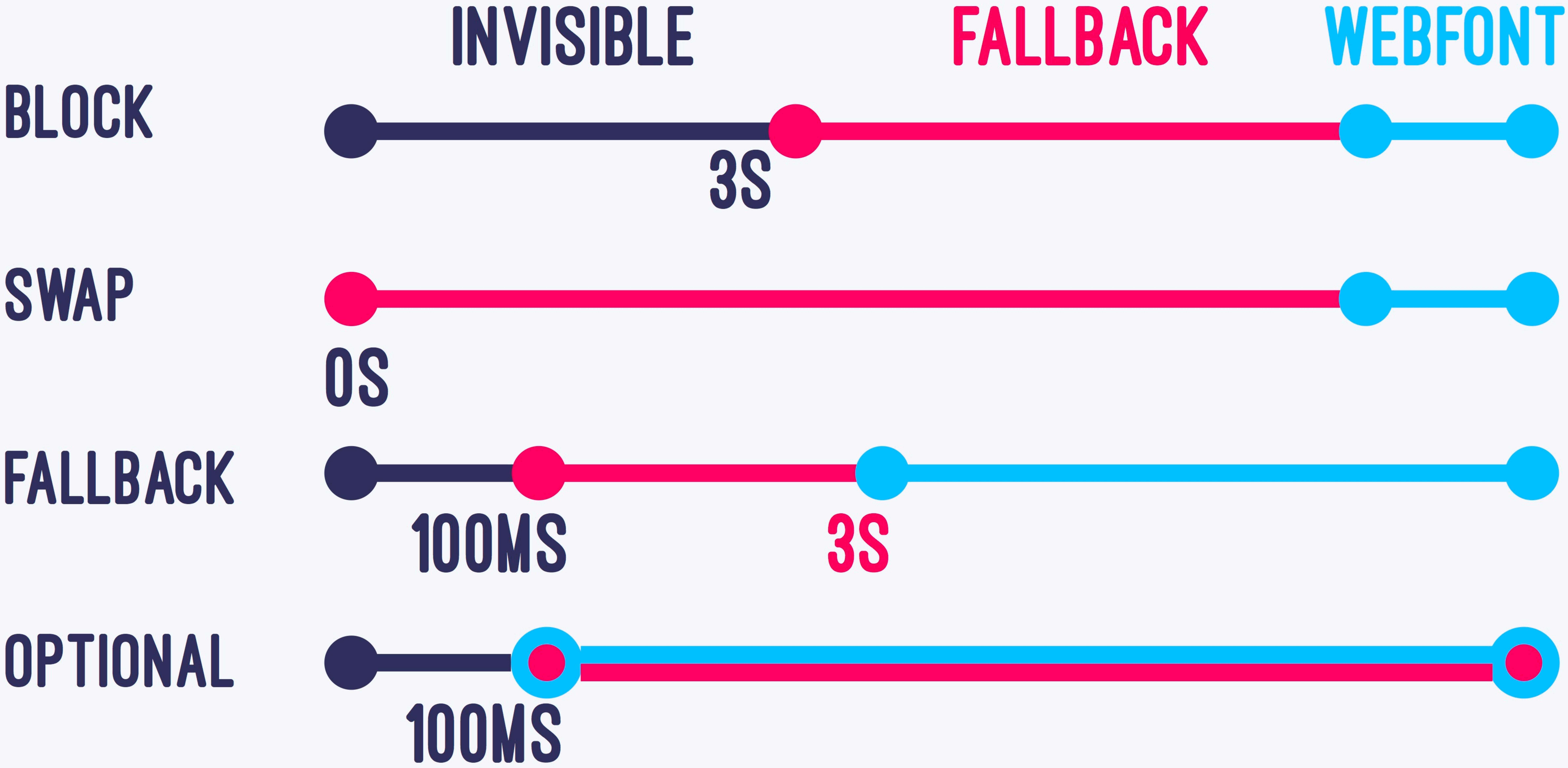
FALLBACK

100MS

3S

OPTIONAL

100MS



INVISIBLE

FALLBACK

WEBFONT

BLOCK

3S

SWAP

0S

FALLBACK

100MS

3S

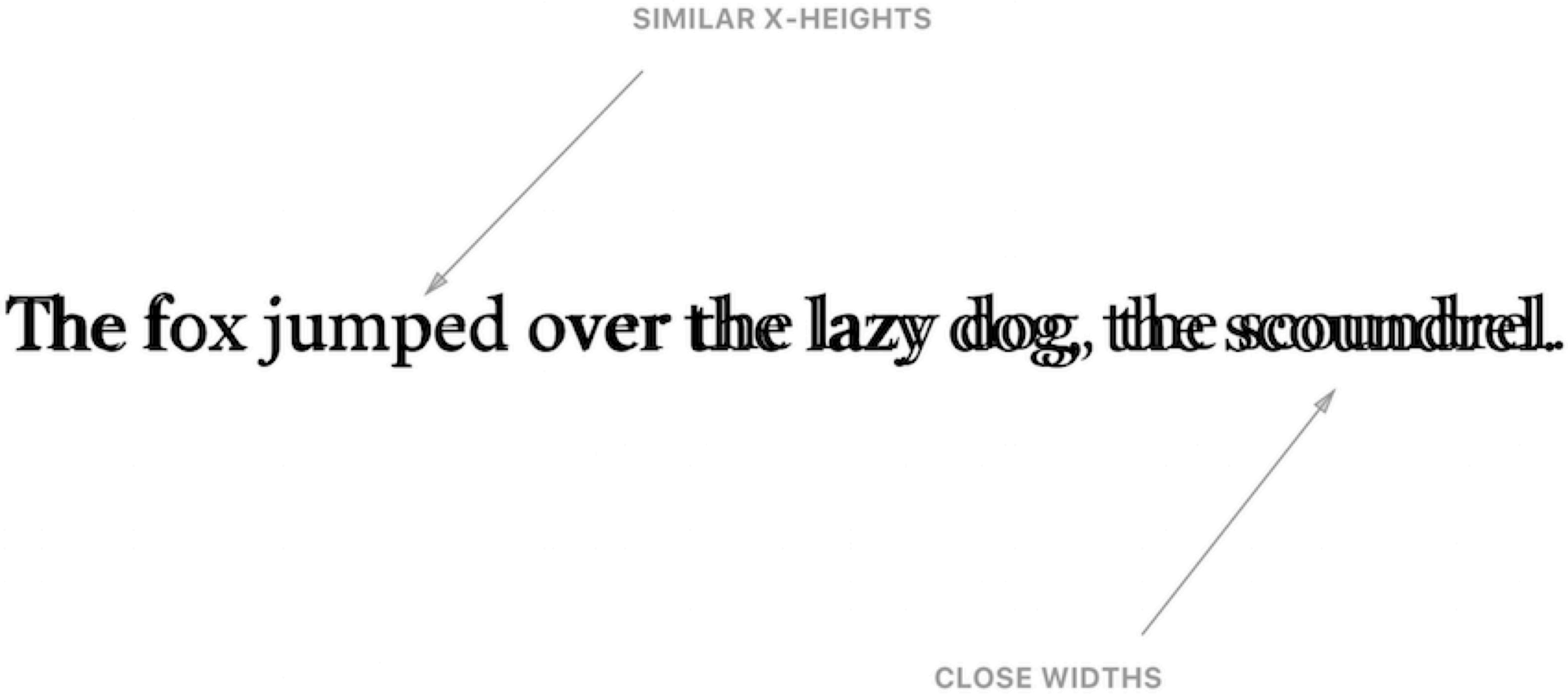
OPTIONAL

100MS

MINIMIZE FOUT

BE LESS JARRING

The fox jumped over the lazy dog, the scoundrel.

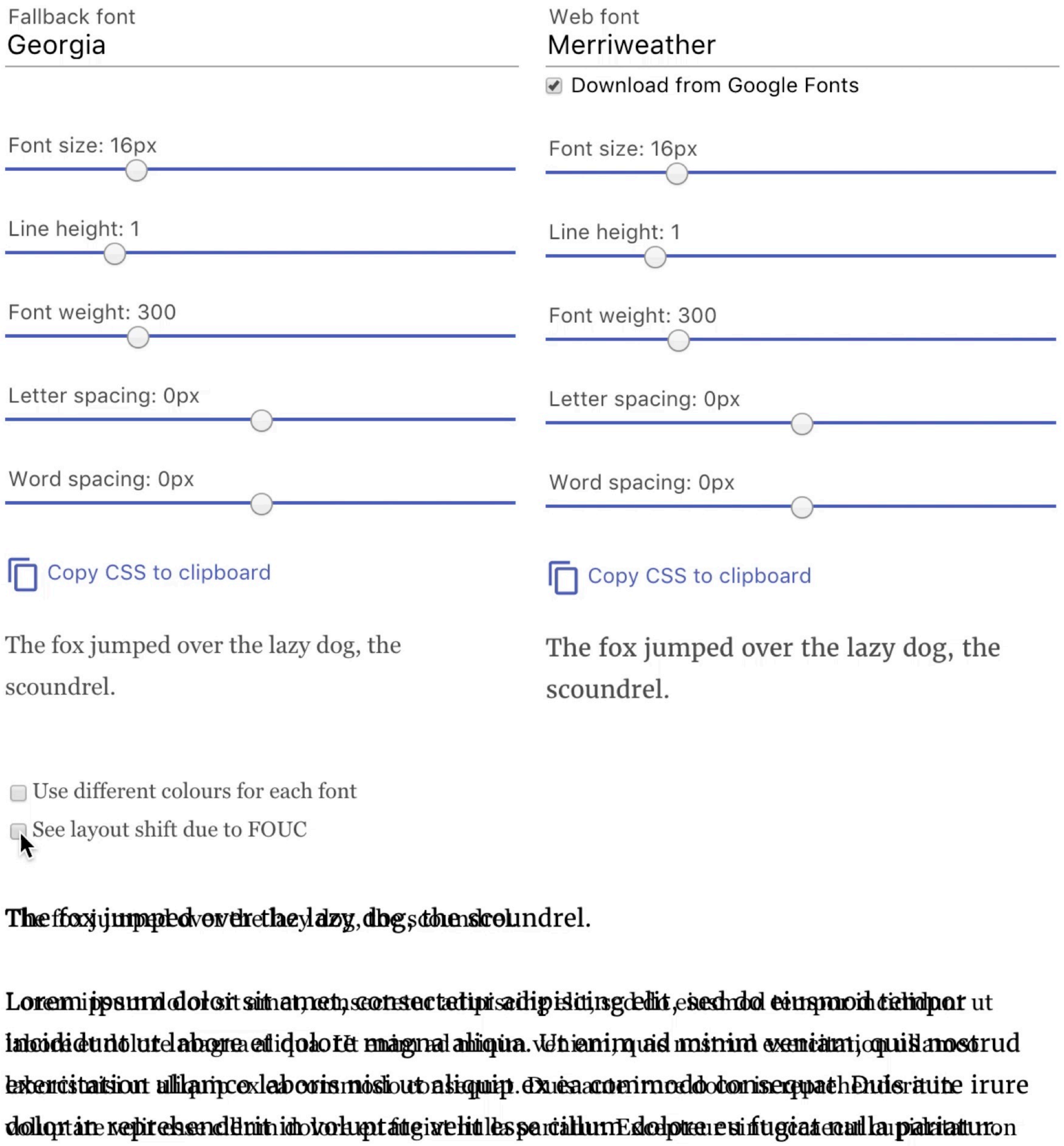


SIMILAR X-HEIGHTS

The fox jumped over the lazy dog, the scoundrel.

CLOSE WIDTHS

HTTPS://MEOWNI.CA/FONT-STYLE-MATCHER



Fallback font
Georgia

Font size: 16px

Line height: 1

Font weight: 300

Letter spacing: 0px

Word spacing: 0px

Copy CSS to clipboard

The fox jumped over the lazy dog, the scoundrel.

Web font
Merriweather

☑ Download from Google Fonts

Font size: 16px

Line height: 1

Font weight: 300

Letter spacing: 0px

Word spacing: 0px

Copy CSS to clipboard

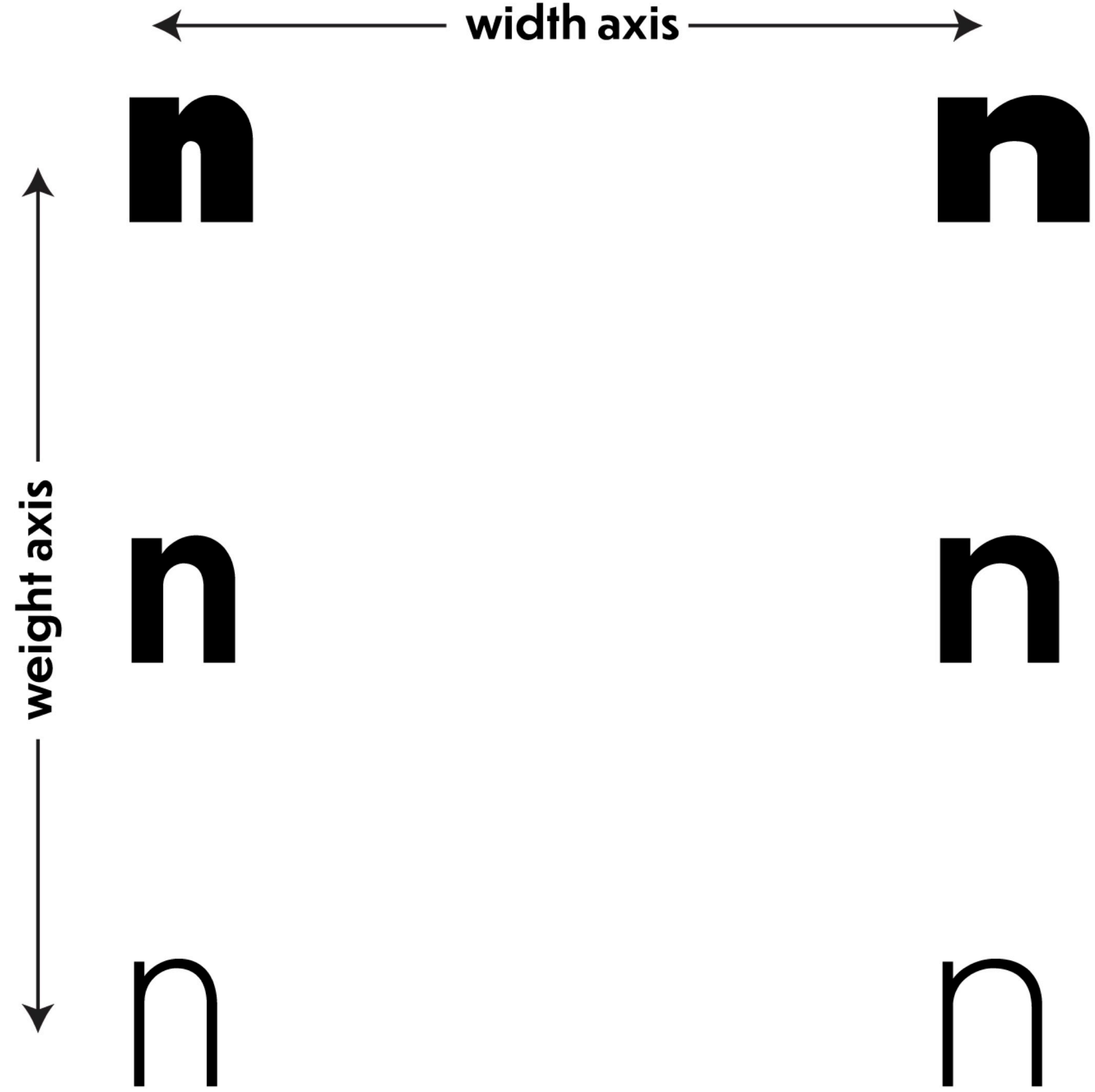
The fox jumped over the lazy dog, the scoundrel.

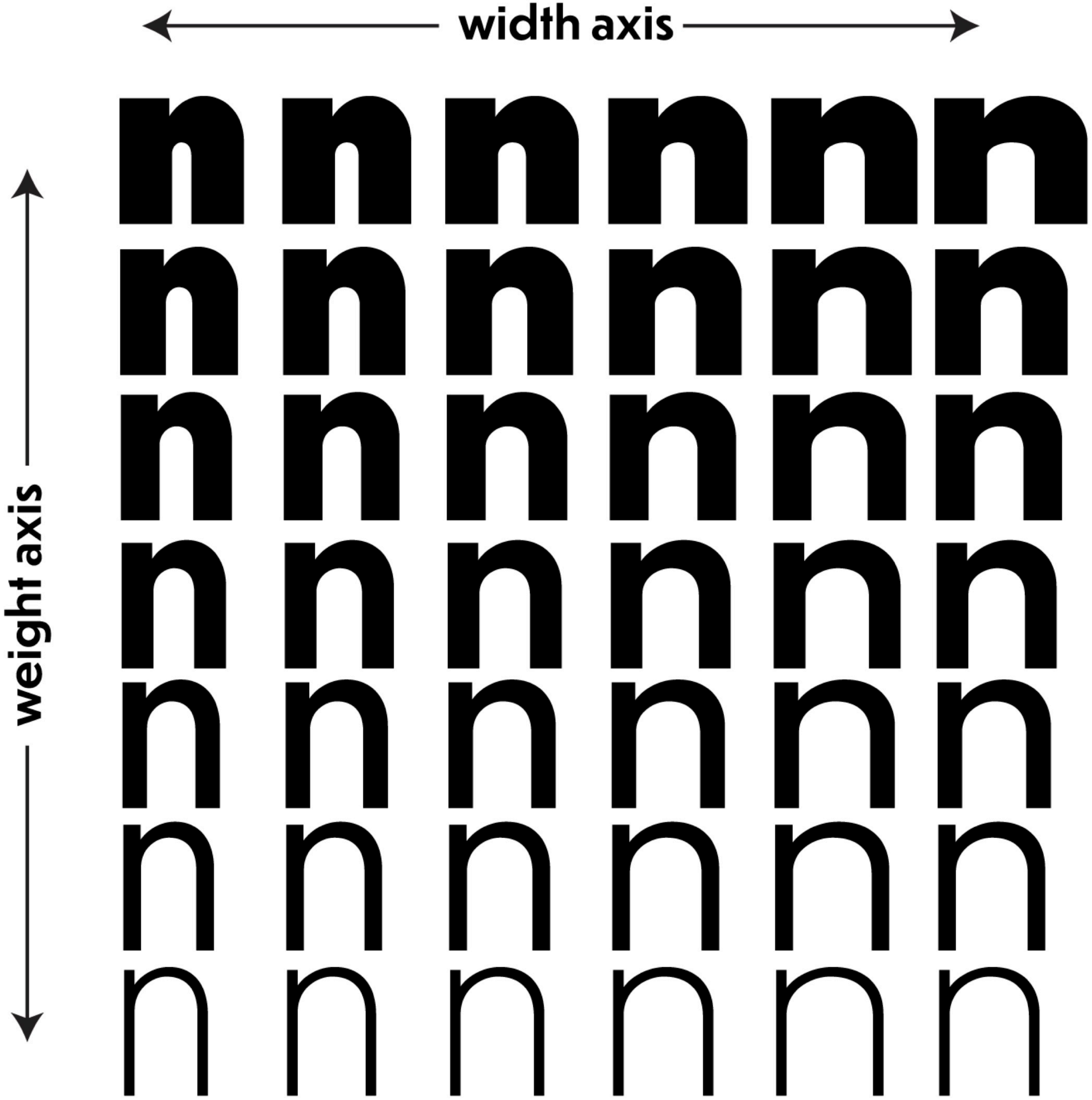
□ Use different colours for each font

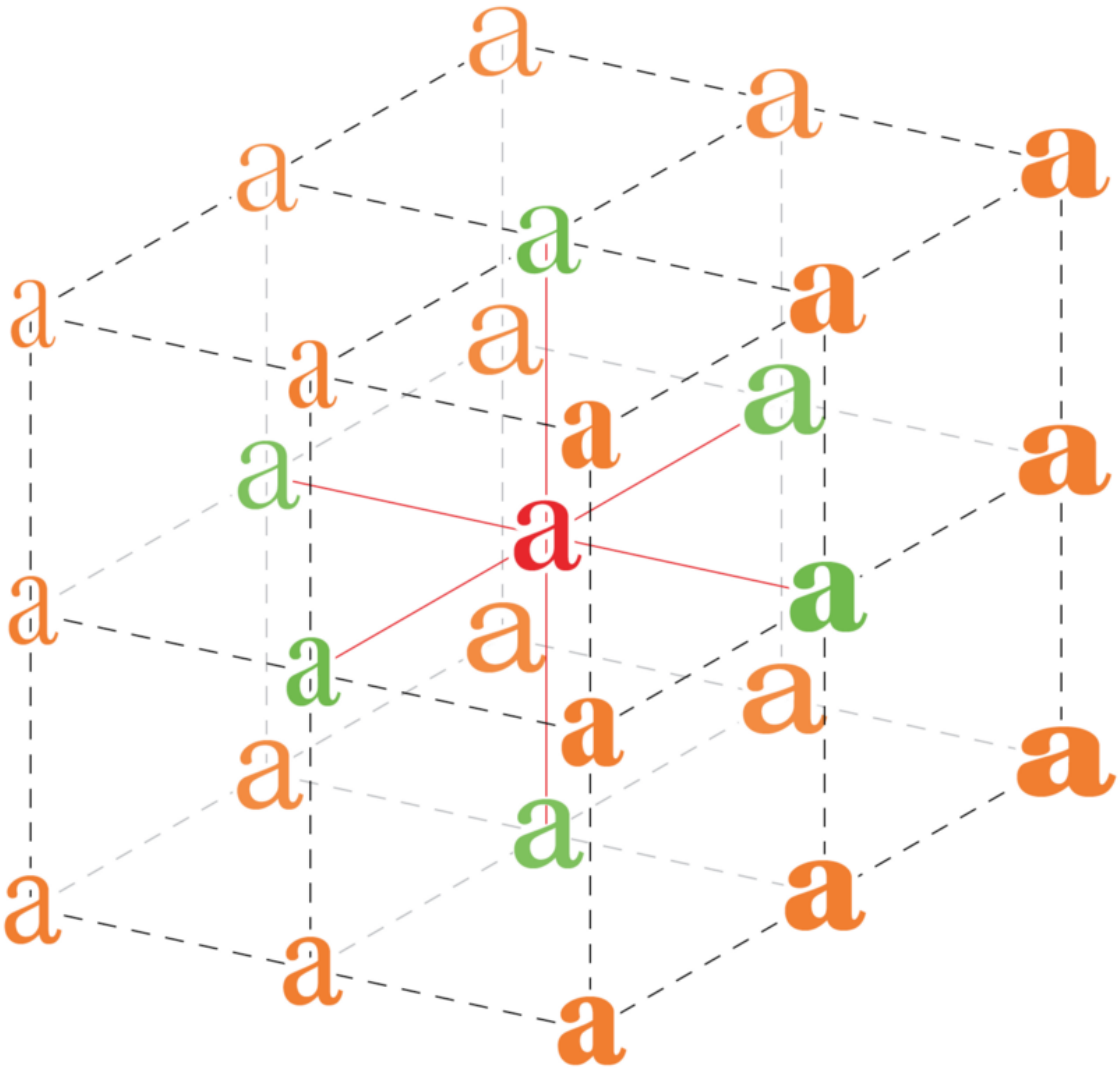
□ See layout shift due to FOUC

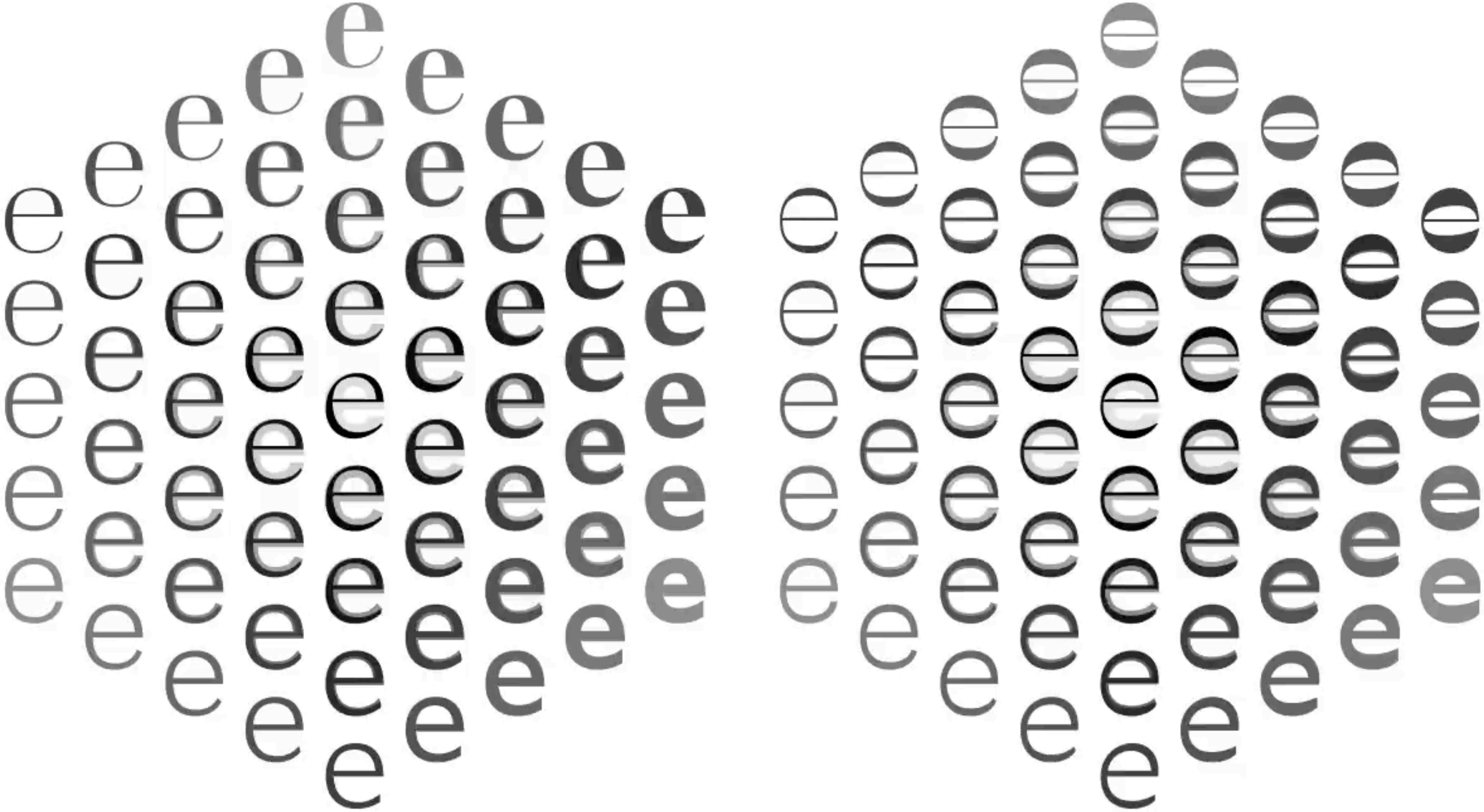
The fox jumped over the lazy dog, the scoundrel.

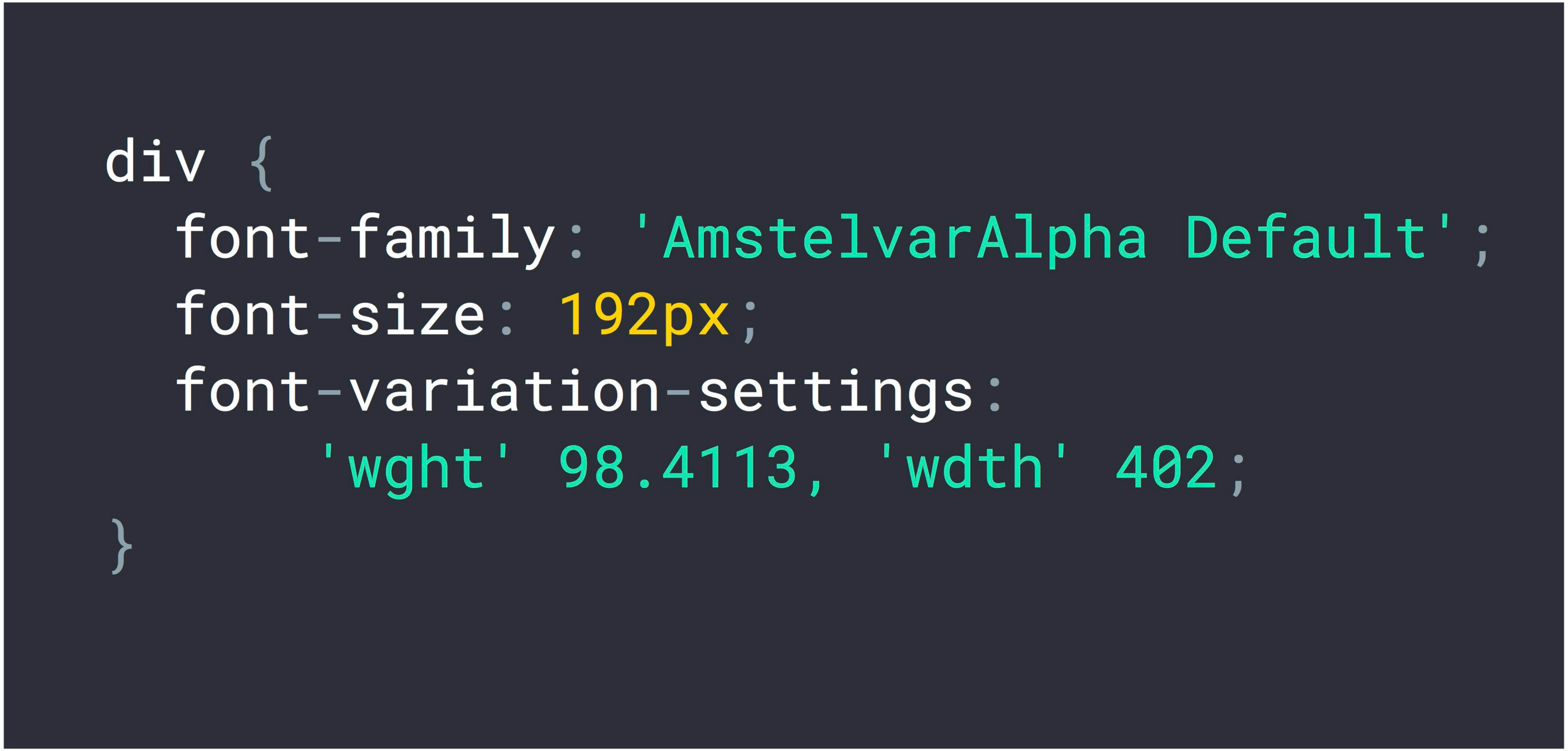
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.


width axis
weight axis
n
n
n
n
n
n


width axis
weight axis

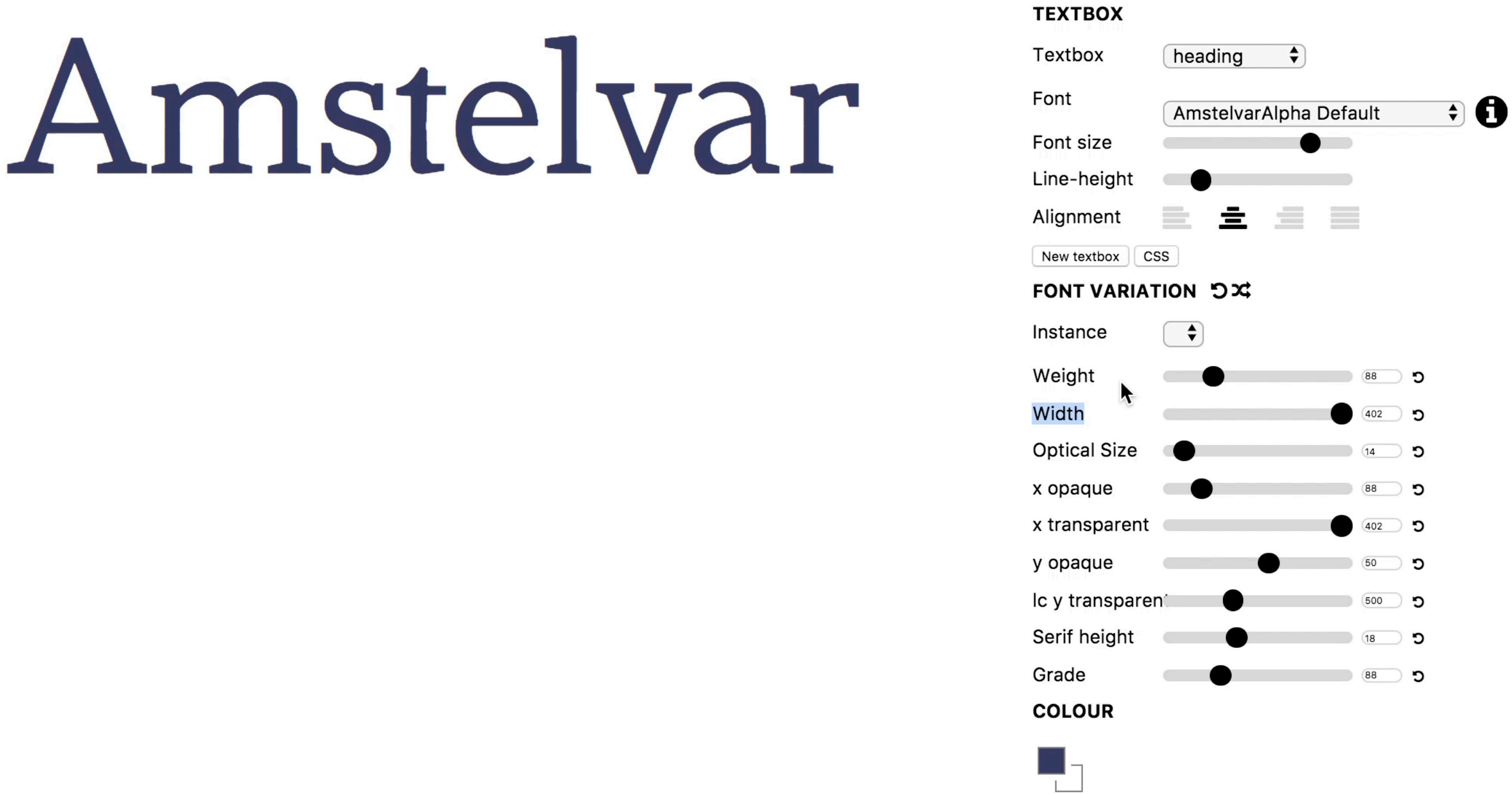






```css
div {
  font-family: 'AmstelvarAlpha Default';
  font-size: 192px;
  font-variation-settings:
      'wght' 98.4113, 'wdth' 402;
}
```

HTTPS://AXIS-PRAXIS.ORG



Amstelvar

TEXTBOX

Textbox: heading

Font: AmstelvarAlpha Default

Font size

Line-height

Alignment

FONT VARIATION

Instance

Weight 88

Width 402

Optical Size 14

x opaque 88

x transparent 402

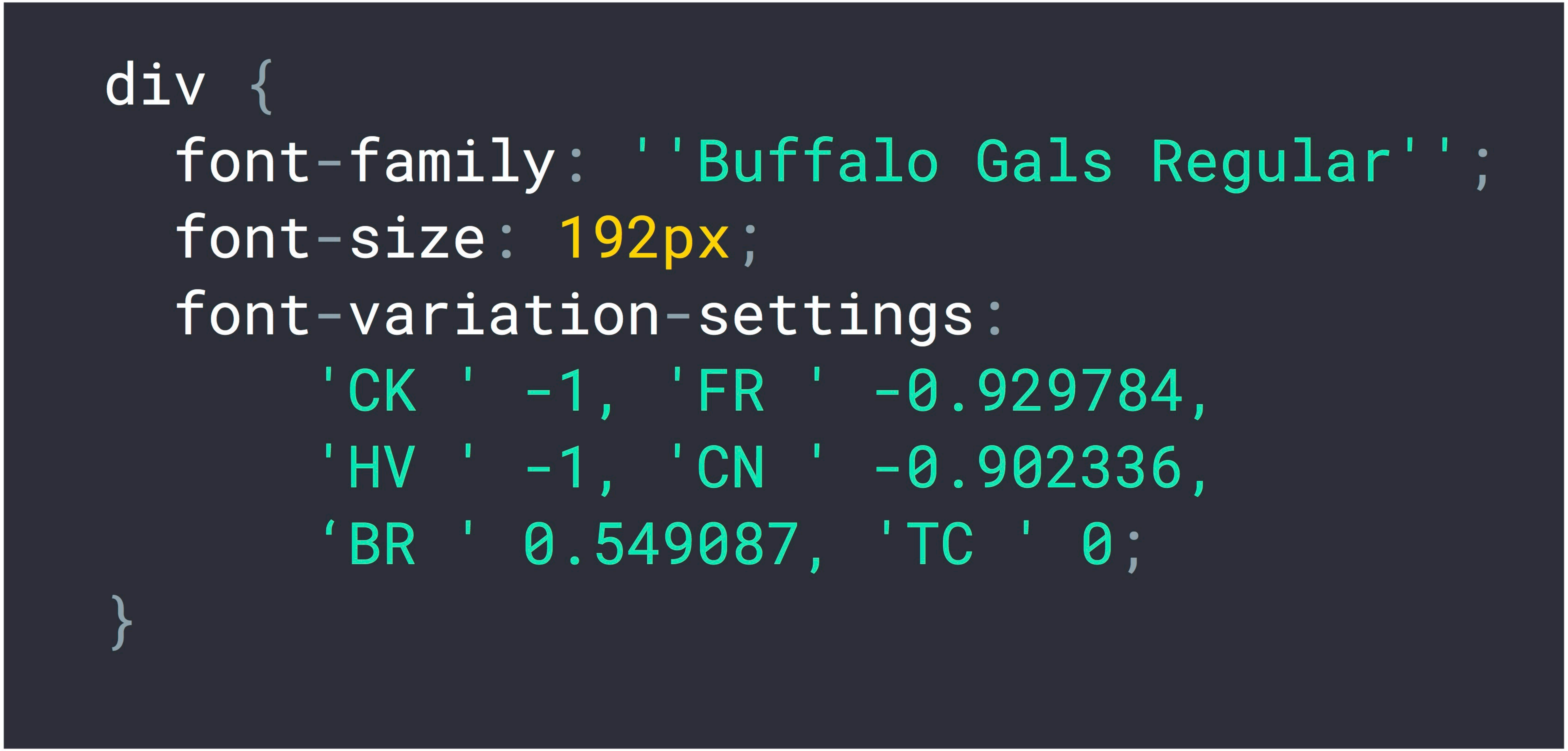
y opaque 50

lc y transparent 500

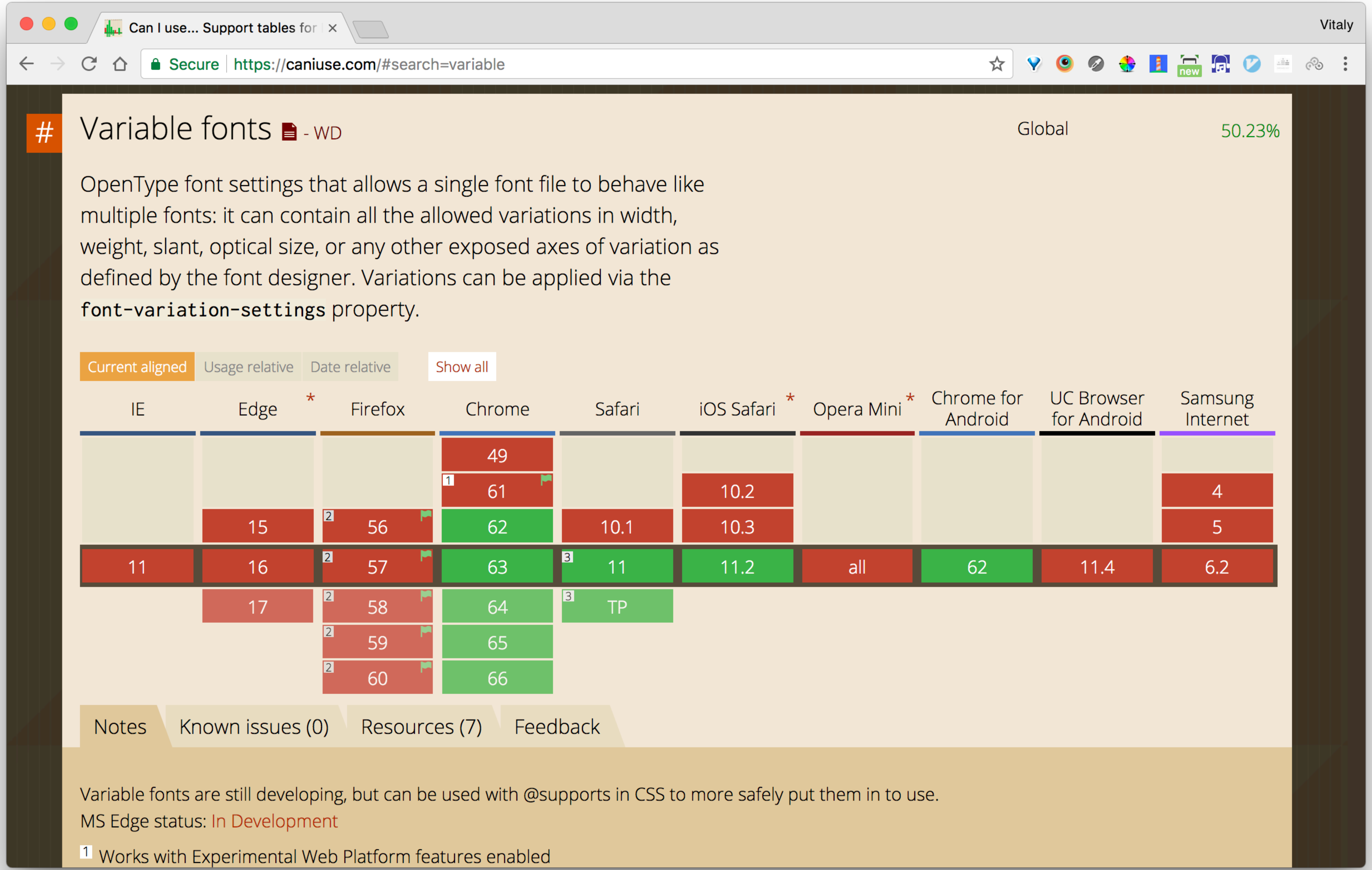
Serif height 18

Grade 88

COLOUR



```
div {
  font-family: ''Buffalo Gals Regular'';
  font-size: 192px;
  font-variation-settings:
      'CK ' -1, 'FR ' -0.929784,
      'HV ' -1, 'CN ' -0.902336,
      ‘BR ' 0.549087, 'TC ' 0;
}
```


Can I use... Support tables for
Vitaly
Secure
https://caniuse.com/#search=variable
new

Variable fonts - WD

Global 50.23%

OpenType font settings that allows a single font file to behave like multiple fonts: it can contain all the allowed variations in width, weight, slant, optical size, or any other exposed axes of variation as defined by the font designer. Variations can be applied via the `font-variation-settings` property.

Current aligned
Usage relative
Date relative
Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			[1] 61		10.2				4
	15	[2] 56	62	10.1	10.3				5
11	16	[2] 57	63	[3] 11	11.2	all	62	11.4	6.2
	17	[2] 58	64	[3] TP					
		[2] 59	65						
		[2] 60	66						

Variable fonts are still developing, but can be used with @supports in CSS to more safely put them in to use.

MS Edge status: In Development

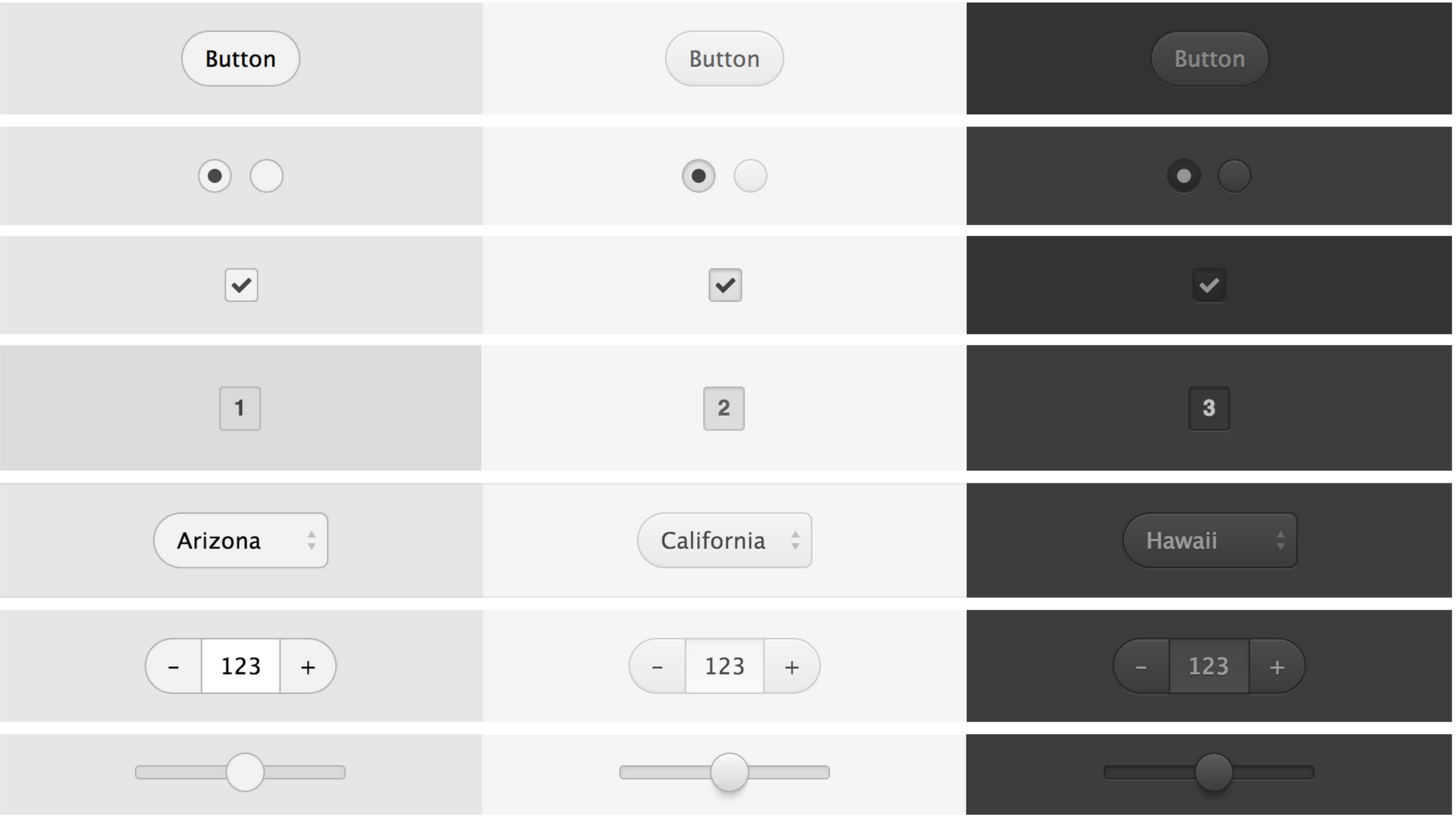
[1] Works with Experimental Web Platform features enabled

FONTASTIC

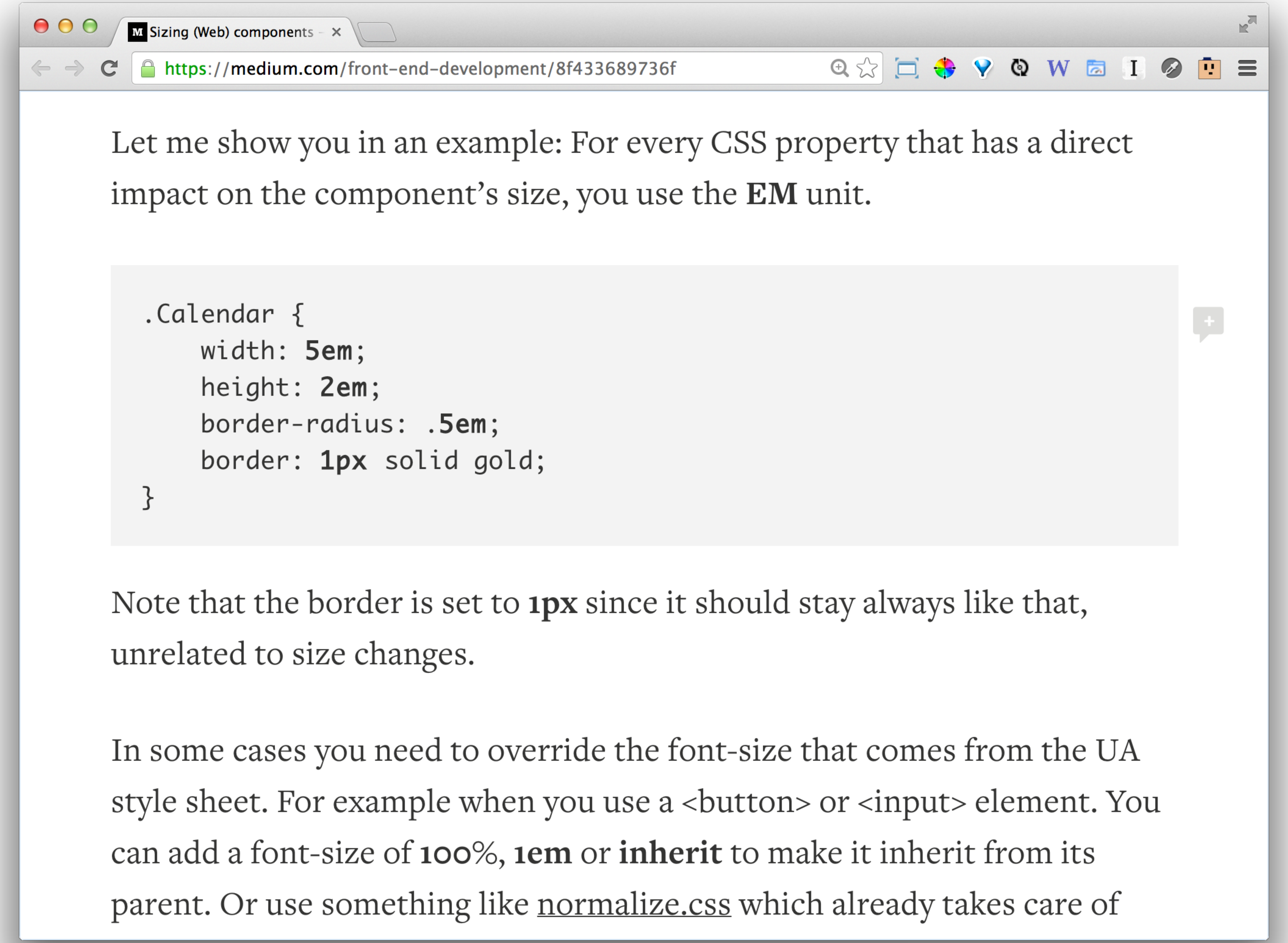
WEB PERFORMANCE

BONUS LEVEL

> “How do you *efficiently* scale up / down any UI component (e.g. a slider or calendar) and keep all the proportions intact—without fiddling with width, height or border-radius manually?
>
> — *@simurai*


Button
Button
Button
1
2
3
Arizona
California
Hawaii
–
123
+
–
123
+
–
123
+

> “By sneaking a *Trojan horse* into your components. We use *rem* for components “root” and *em* for sub-parts of the components. Then, by adjusting the *font-size of the root,* we adjust *all* size-related CSS properties of a component at once.
>
> — *@simurai*

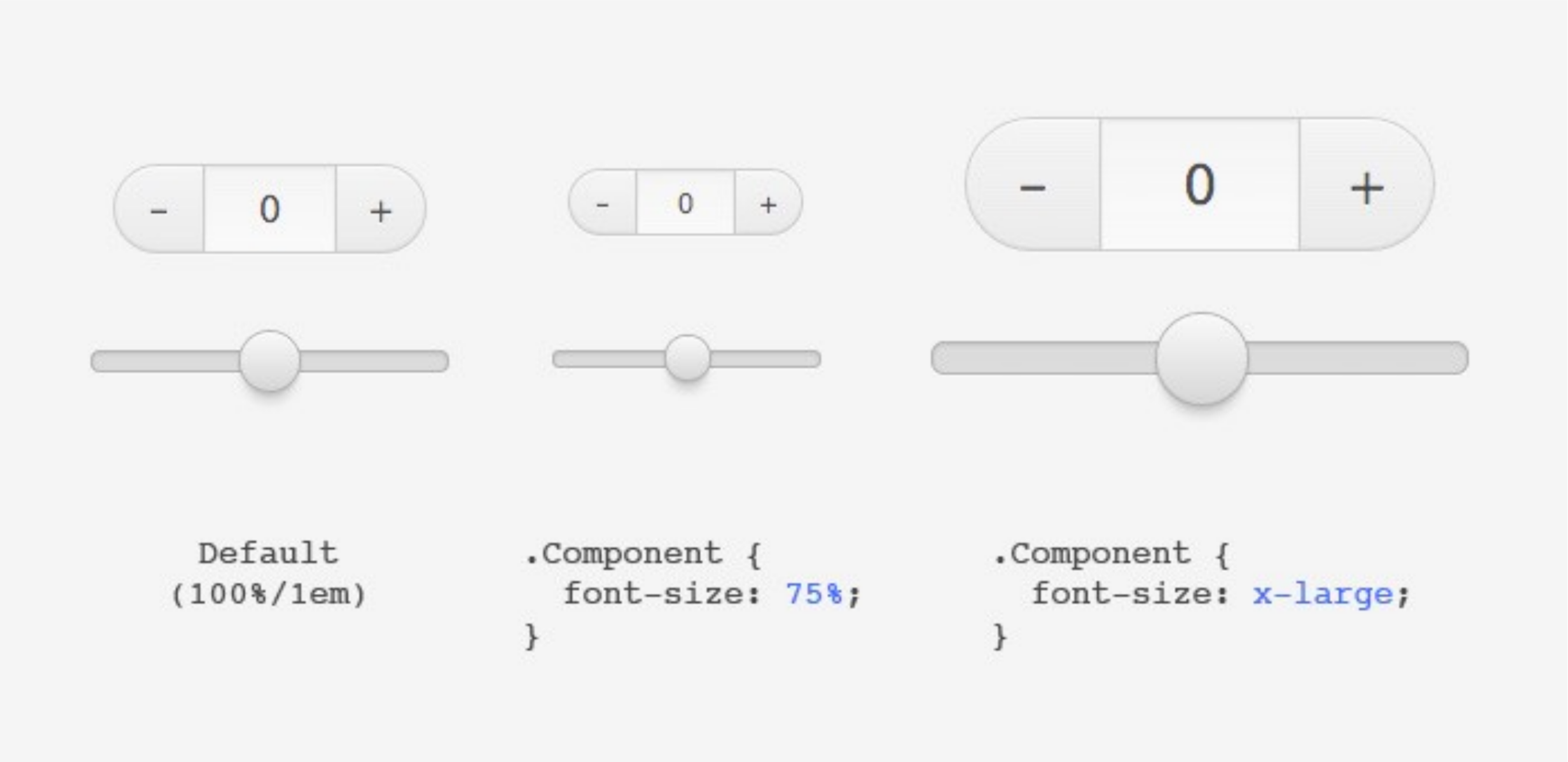

Sizing (Web) components -
https://medium.com/front-end-development/8f433689736f

Let me show you in an example: For every CSS property that has a direct impact on the component’s size, you use the **EM** unit.

```
.Calendar {
    width: 5em;
    height: 2em;
    border-radius: .5em;
    border: 1px solid gold;
}
```

Note that the border is set to **1px** since it should stay always like that, unrelated to size changes.

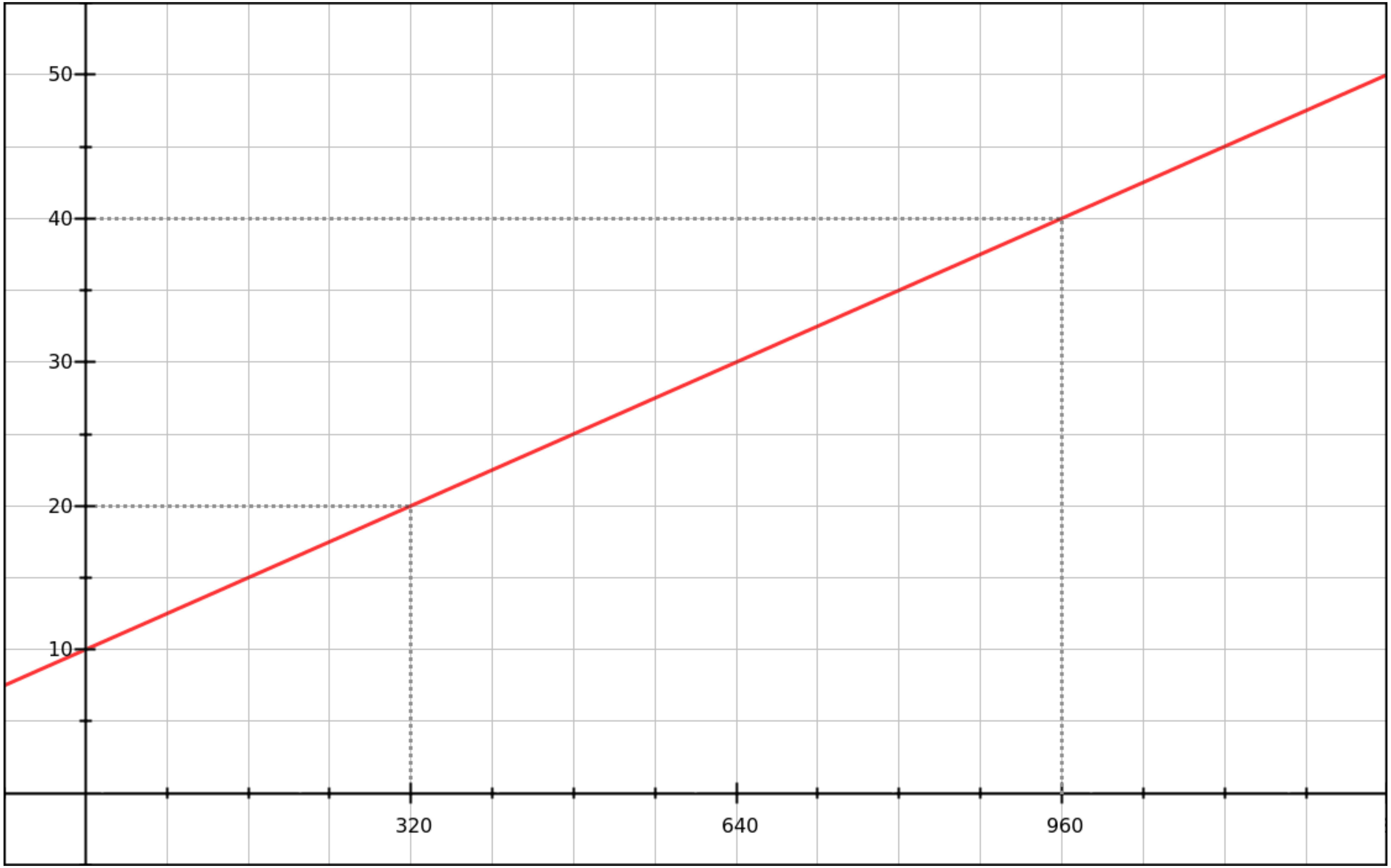
In some cases you need to override the font-size that comes from the UA style sheet. For example when you use a <button> or <input> element. You can add a font-size of **100%**, **1em** or **inherit** to make it inherit from its parent. Or use something like normalize.css which already takes care of

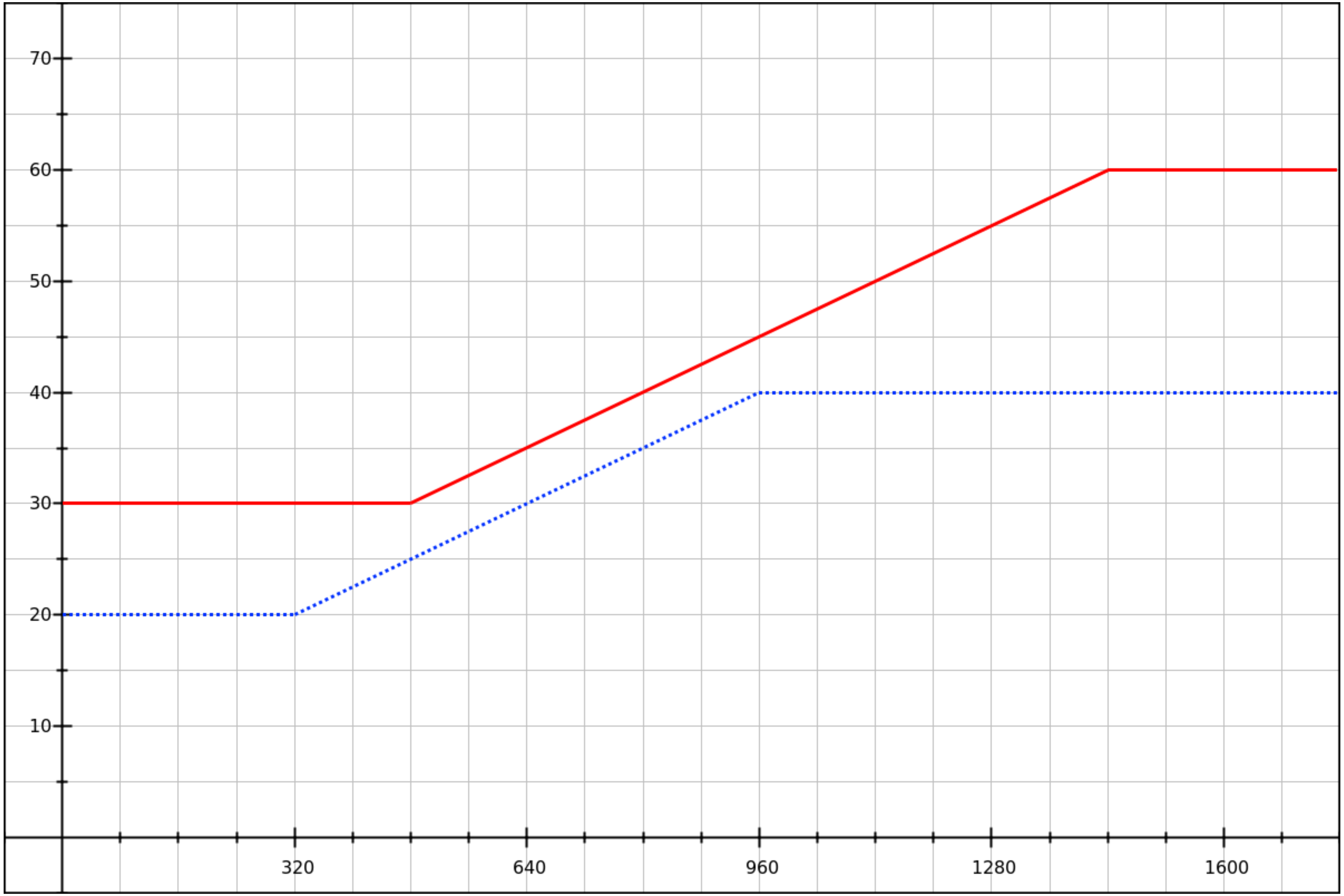


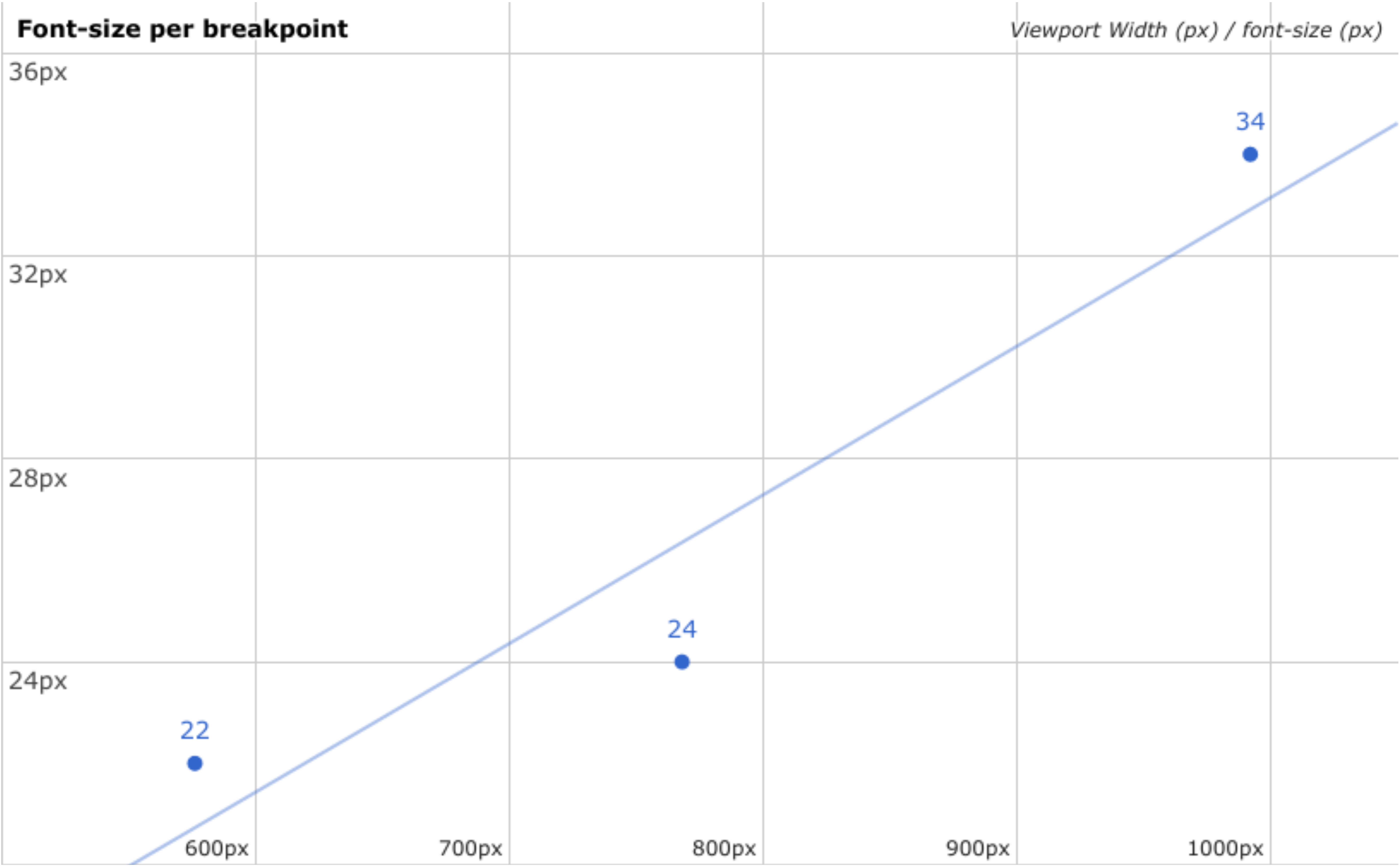
```
.Component {
  font-size: 75%;
}
```

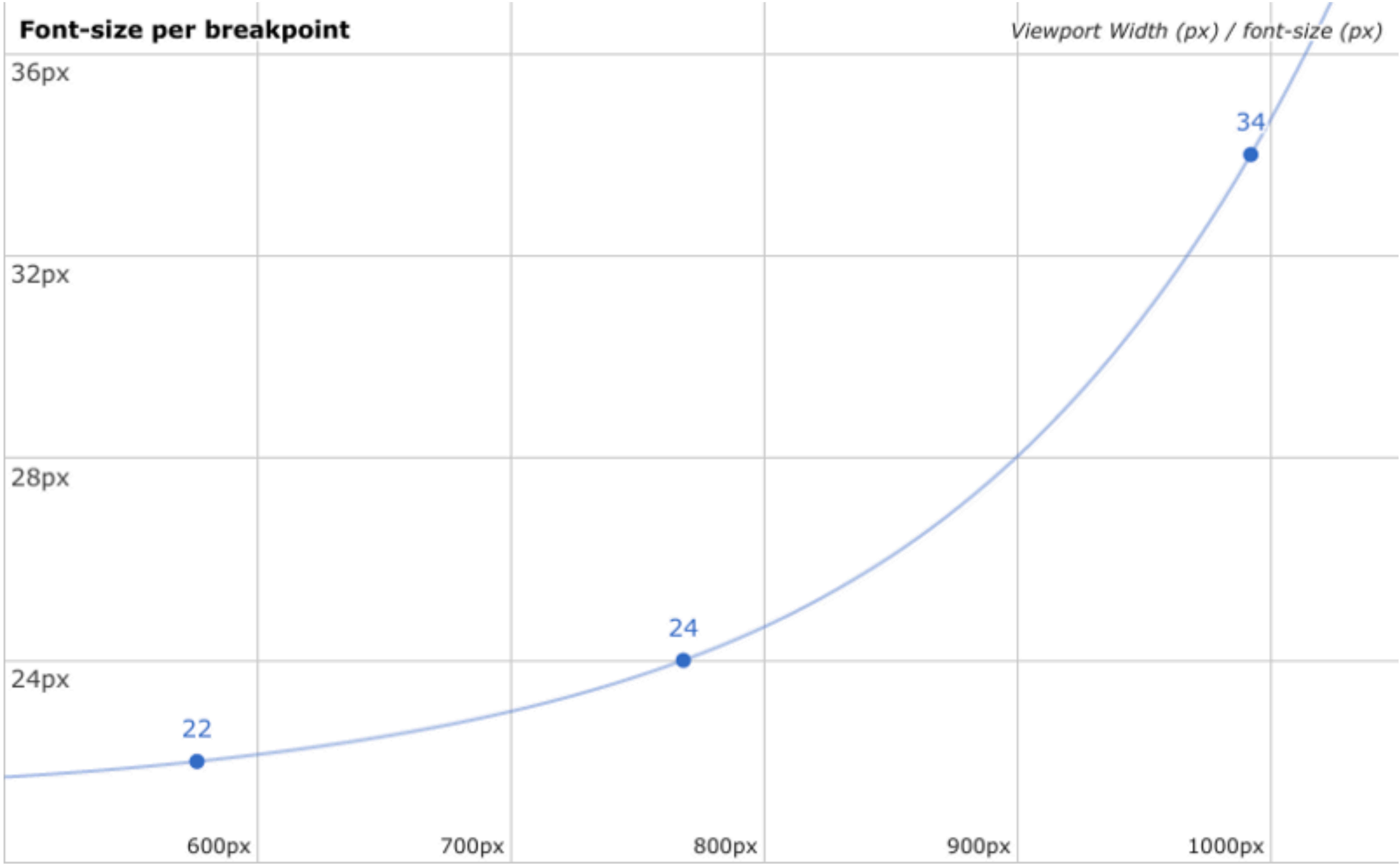
```
.Component {
  font-size: x-large;
}
```

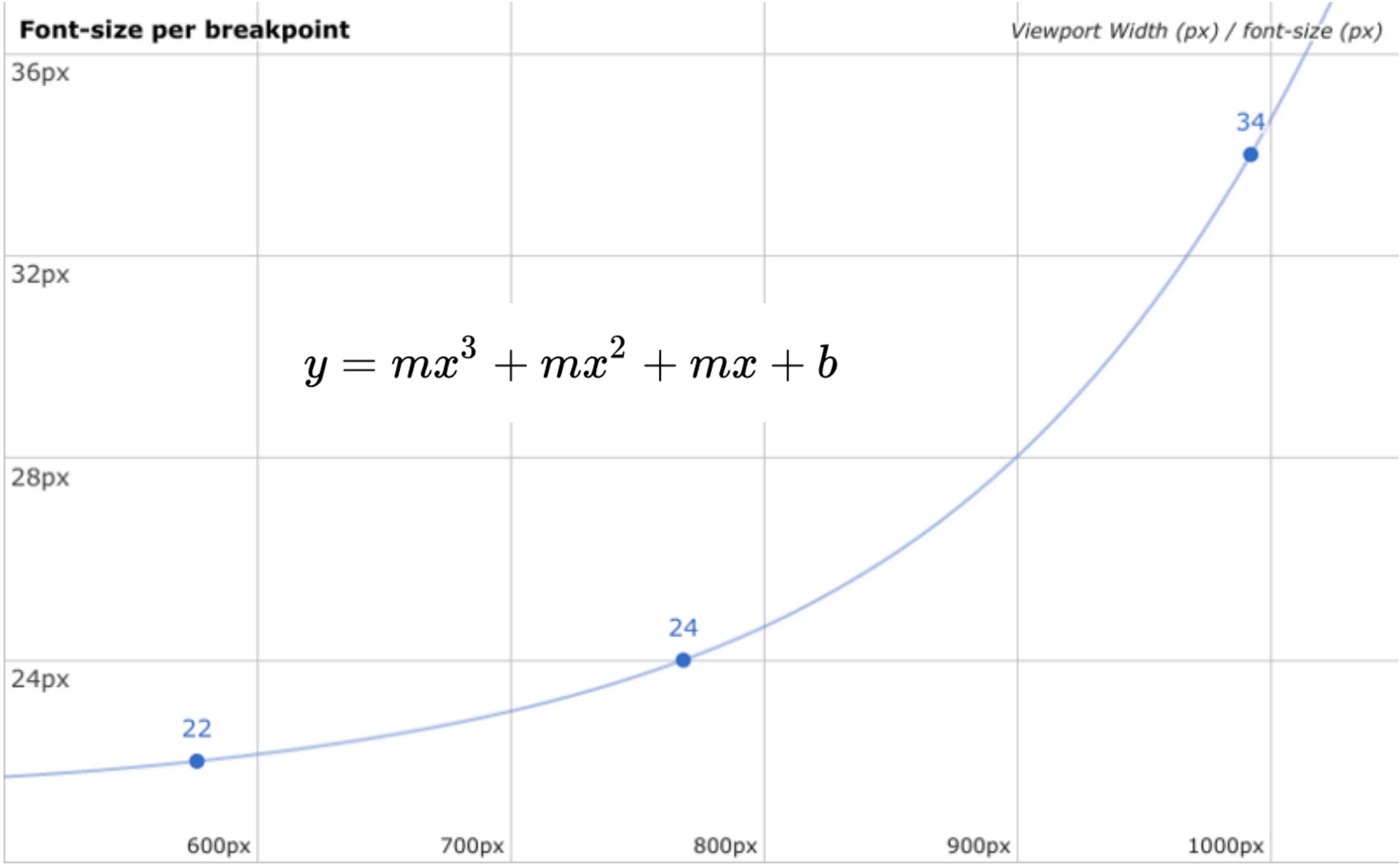
Example used: Digit components


50
40
30
20
10
320
640
960

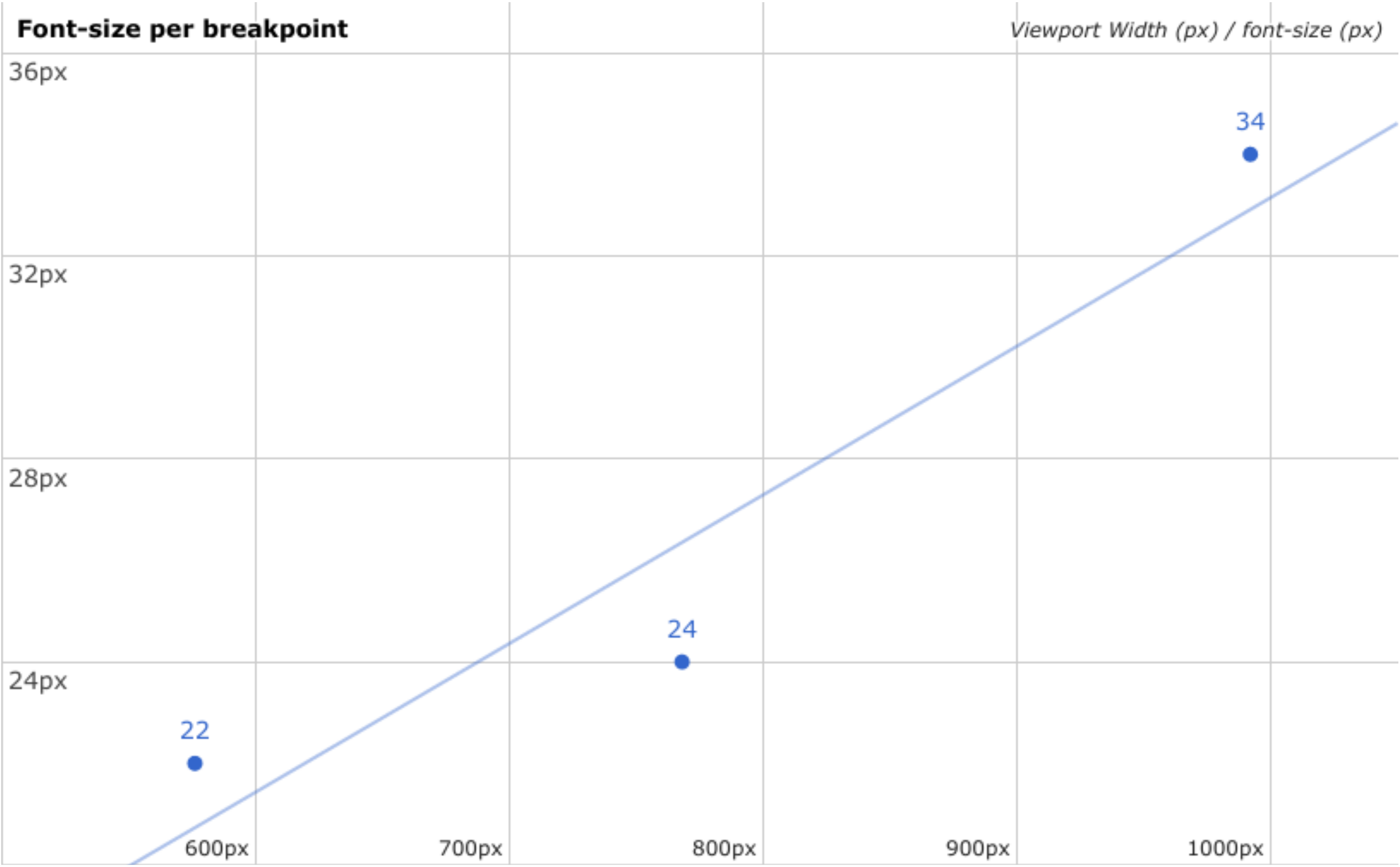

70
60
50
40
30
20
10
320
640
960
1280
1600


Font-size per breakpoint
Viewport Width (px) / font-size (px)
36px
32px
28px
24px
34
24
22
600px
700px
800px
900px
1000px

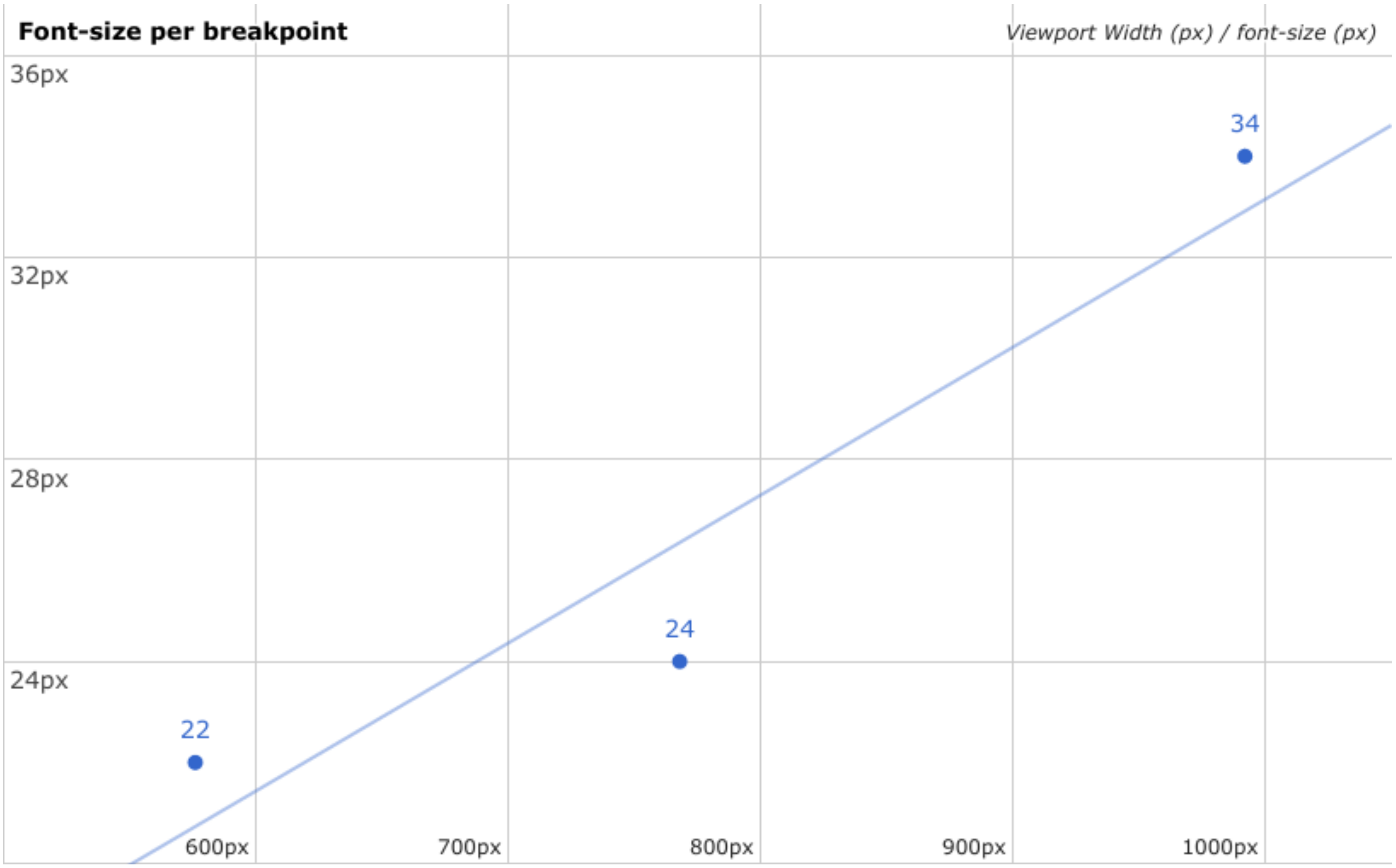

Font-size per breakpoint
Viewport Width (px) / font-size (px)
36px
32px
28px
24px
22
24
34
600px
700px
800px
900px
1000px


Font-size per breakpoint
Viewport Width (px) / font-size (px)
36px
32px
28px
24px
22
24
34
600px
700px
800px
900px
1000px

$$y = mx^3 + mx^2 + mx + b$$


Font-size per breakpoint
Viewport Width (px) / font-size (px)
36px
32px
28px
24px
34
24
22
600px
700px
800px
900px
1000px

$$y = mx + b$$



m = slope

b = the y-intercept

x = current **viewport width**

y = resulting **font size**

$$\overline{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\overline{Y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$m = \frac{\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})}{\sum_{i=1}^{n}(x_i - \overline{X})^2}$$

$$b = \overline{Y} - m\overline{X}$$

m = slope

x = current **viewport width**

b = the y-intercept

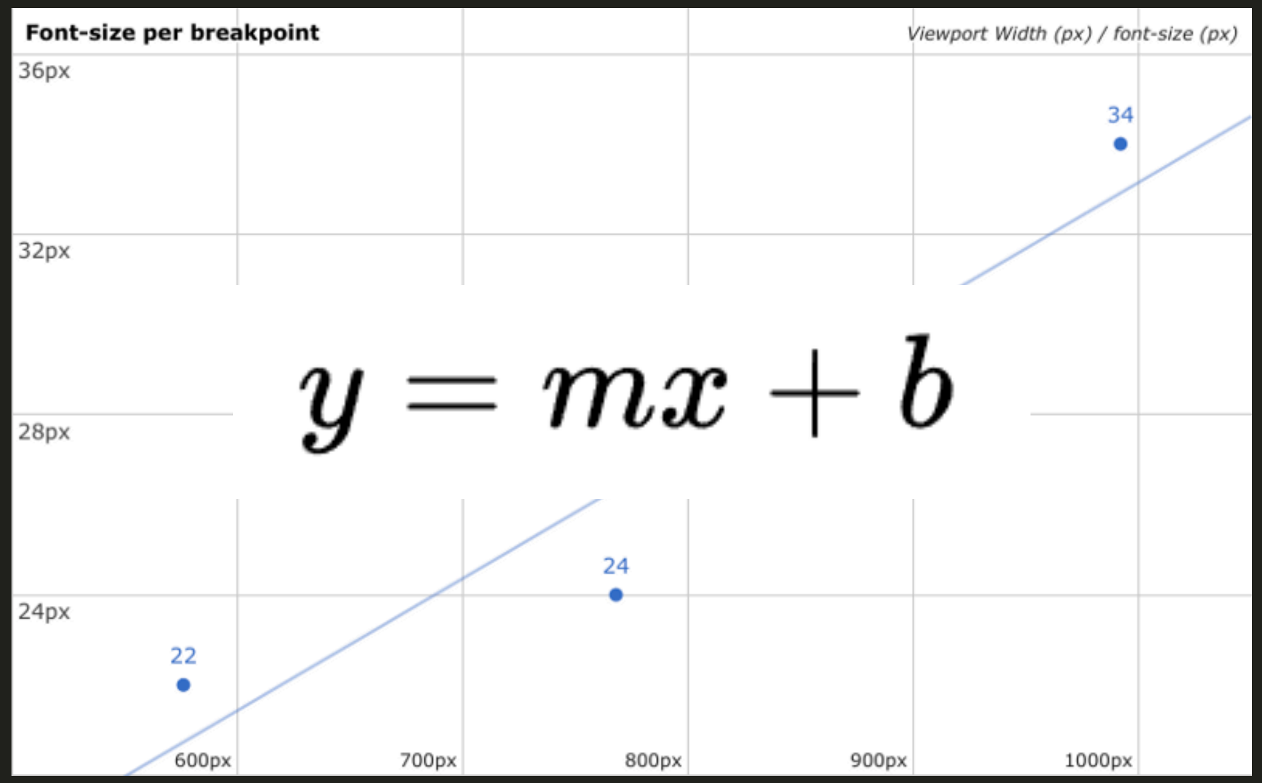
y = resulting **font size**

Fluid Typography, rem

```
/* CSS Reset of your choice */
body { font-size: 100%; line-height: 1.45em; }

/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
@media screen and (max-width: 650px) {
    h1 { font-size: 2.8125rem; } /* 2.8125 ÷ 1.875 = 1.5 /*
    h2 { font-size: 1.875rem; } /* 1.875 ÷ 0.75 = 2.25 /*
    h3 { font-size: 0.75rem; }
}

@media screen and (max-width: 1050px) {
    h1 { font-size: 3.375rem; } /* 3.375 ÷ 2.25 = 1.5 /*
    h2 { font-size: 2.25rem; } /* 2.25 ÷ 1 = 2.25 /*
    h3 { font-size: 1rem; }
}
```

Fluid Typography, calc

```
/* CSS Reset of your choice */
body { font-size: 100%; line-height: 1.45em; }

/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 { font-size: calc({slope-h1}*100vw + {y-intercept-h1}px); }
h2 { font-size: calc({slope-h2}*100vw + {y-intercept-h2}px); }
h3 { font-size: calc({slope-h3}*100vw + {y-intercept-h3}px); }
```



$$y = mx + b$$

m = slope

b = the y-intercept

x = current viewport width

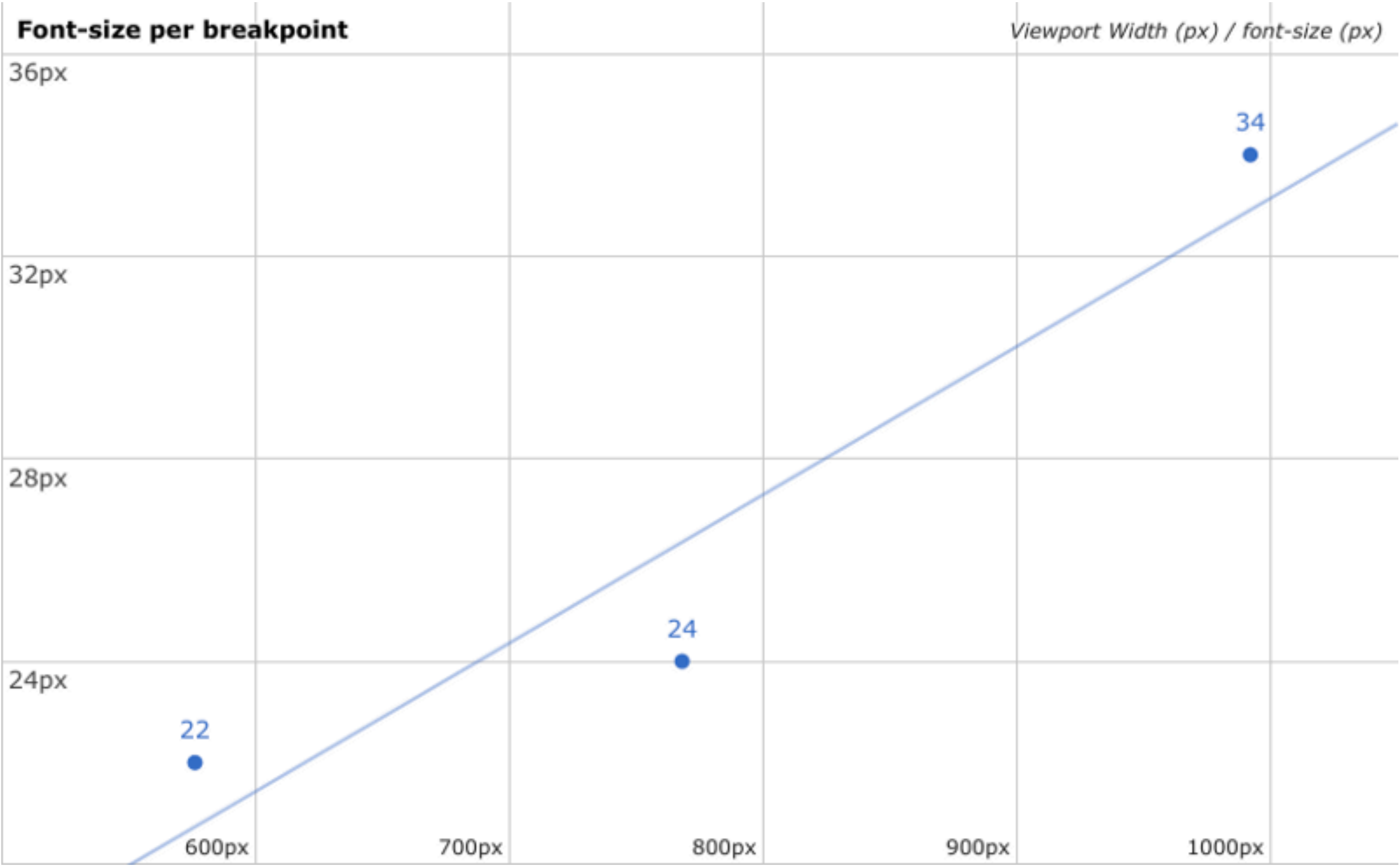
y = resulting font size

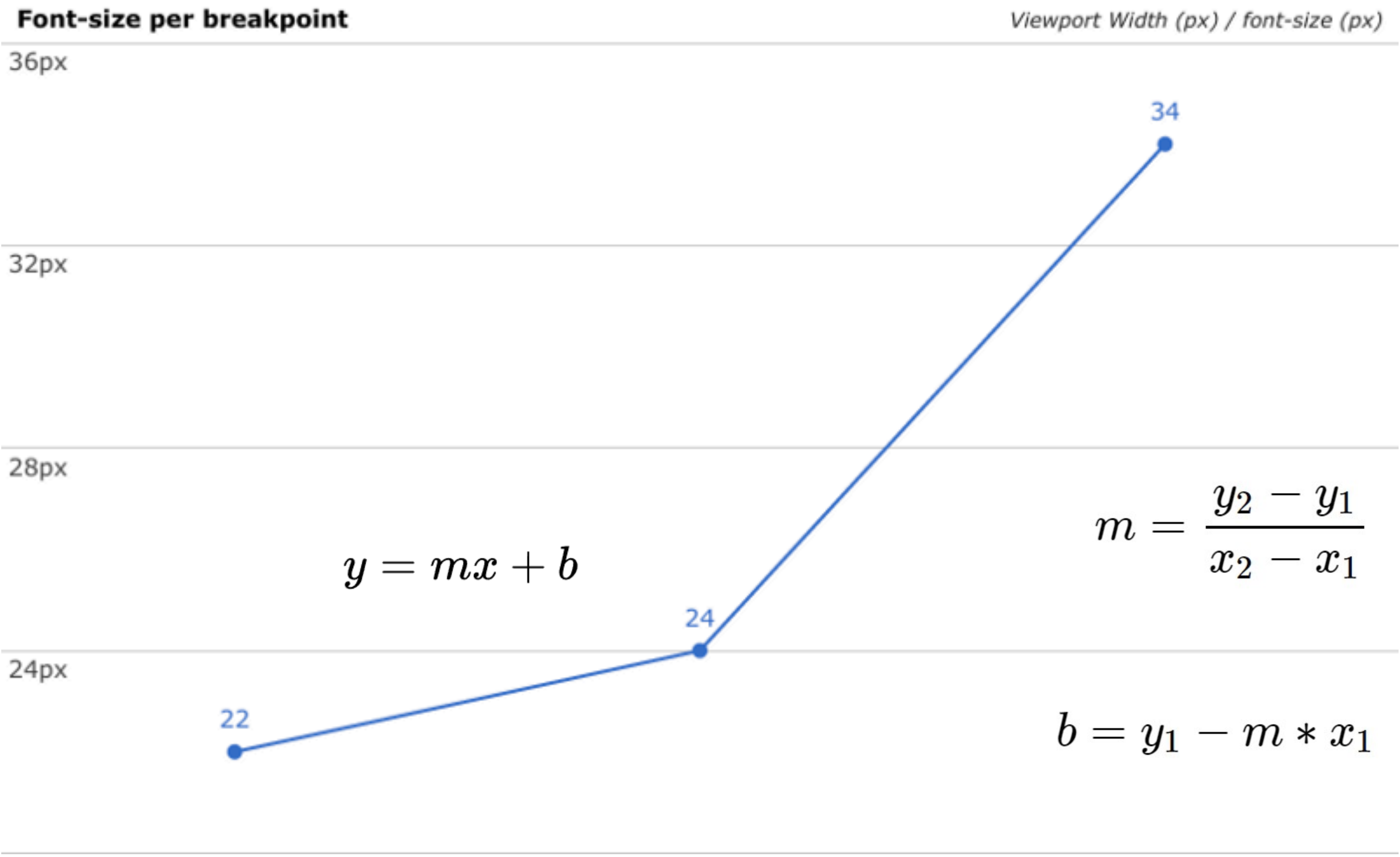
Fluid Typography, calc + px

```
/* CSS Reset of your choice */
body { font-size: 100%; line-height: 1.45em; }

/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 { font-size: calc({slope-h1}*100vw + {y-intercept-h1}px); }
h2 { font-size: calc({slope-h2}*100vw + {y-intercept-h2}px); }
h3 { font-size: calc({slope-h3}*100vw + {y-intercept-h3}px); }

nav { width: calc({slope-nav}*100vw + {y-intercept-nav}px); }
button { padding: calc({slope-btn}*100vw + {y-intercept-btn}px); }

.lightbox {
    font-size: calc({slope-lightbox}*100vw +
                    {y-intercept-lightbox}px);
}
```

Fluid Typography, calc + rem

```css
/* CSS Reset of your choice */
body { font-size: 100%; line-height: 1.45em; }

/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 { font-size: calc({slope-h1}*100vw + rem({y-intercept-h1})rem); }
h2 { font-size: calc({slope-h2}*100vw + rem({y-intercept-h2})rem); }
h3 { font-size: calc({slope-h3}*100vw + rem({y-intercept-h3})rem); }

nav { width: calc({slope-nav}*100vw + rem({y-intercept-nav})rem); }
button { padding: calc({slope-btn}*100vw + rem({y-intercept-btn})rem); }

.lightbox {
    font-size: calc({slope-lightbox}*100vw +
                    rem({y-intercept-lightbox})rem);
}
```

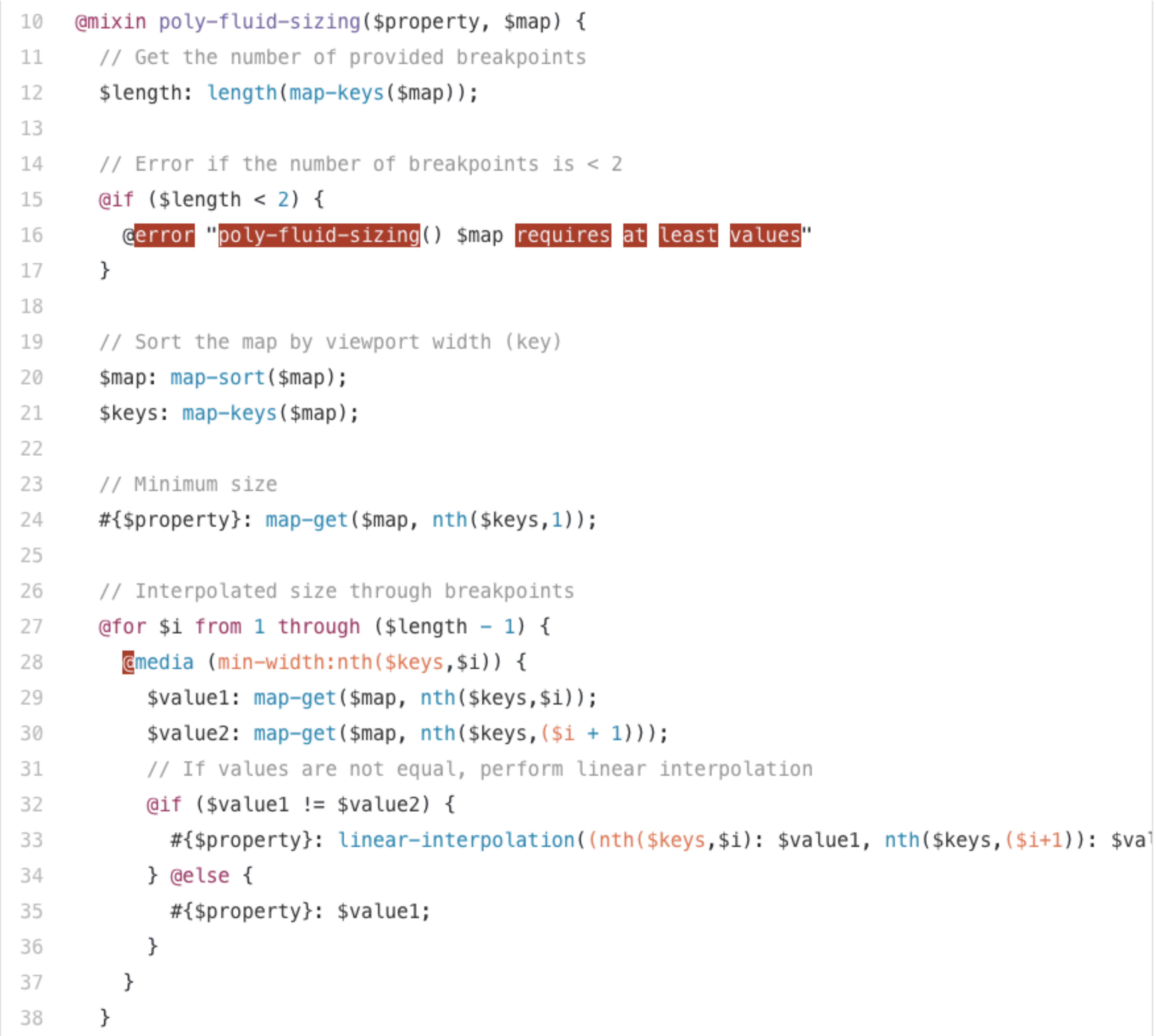

Font-size per breakpoint
Viewport Width (px) / font-size (px)
36px
32px
28px
24px
34
24
22
600px
700px
800px
900px
1000px



$$y = mx + b$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - m * x_1$$



```
@mixin poly-fluid-sizing($property, $map) {
  // Get the number of provided breakpoints
  $length: length(map-keys($map));

  // Error if the number of breakpoints is < 2
  @if ($length < 2) {
    @error "poly-fluid-sizing() $map requires at least values"
  }

  // Sort the map by viewport width (key)
  $map: map-sort($map);
  $keys: map-keys($map);

  // Minimum size
  #{$property}: map-get($map, nth($keys,1));

  // Interpolated size through breakpoints
  @for $i from 1 through ($length - 1) {
    @media (min-width:nth($keys,$i)) {
      $value1: map-get($map, nth($keys,$i));
      $value2: map-get($map, nth($keys,($i + 1)));
      // If values are not equal, perform linear interpolation
      @if ($value1 != $value2) {
        #{$property}: linear-interpolation((nth($keys,$i): $value1, nth($keys,($i+1)): $va
      } @else {
        #{$property}: $value1;
      }
    }
  }
```

Fluid Typography, SCSS → CSS

```scss
/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 {
    $map(576px: 22px, 320px: 18px, 992px: 34px, 768px: 24px);
    @include polyFluidSizing('font-size',$map);
}

h2 {
    $map(592px: 14px, 380px: 16px, 1017px: 22px, 694px: 18px);
    @include polyFluidSizing('font-size',$map);
}

nav {
    $map(500px: 8px, 396px: 6px, 990px: 12px, 605px: 10px);
    @include polyFluidSizing('padding',$map);
}
```

Fluid Typography, SCSS → CSS

```
/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 {
    $map(576px: 22px, 320px: 18px, 992px: 34px, 768px: 24px);
    @include polyFluidSizing('font-size',$map);
}
```

```
h1 { font-size: 18px; } /* Minimum font size: 18px; */

@media screen and (min-width: 320px) { /* Interpolation: 18px → 22px */
    h1 { font-size: calc(1.04166667vw + 14.2444px); }
}

@media screen and (min-width: 576px) { /* Interpolation: 22px → 24px */
    h1 { font-size: calc(2.1821vw + 9.4621px); }
}
```

Fluid Typography, SCSS → CSS

```
/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 {
    $map(576px: 22px, 320px: 18px, 992px: 34px, 768px: 24px);
    @include polyFluidSizing('font-size',$map);
}
```

```
@media screen and (min-width: 768px) { /* Interpolation: 24px → 34px */
    h1 { font-size: calc(4.7787vw + 21.2444px); }
}

@media screen and (min-width: 992px) { /* Maximum font size: 34px */
    h1 { font-size: 34px; }
}
```

Fluid Typography, SCSS → CSS

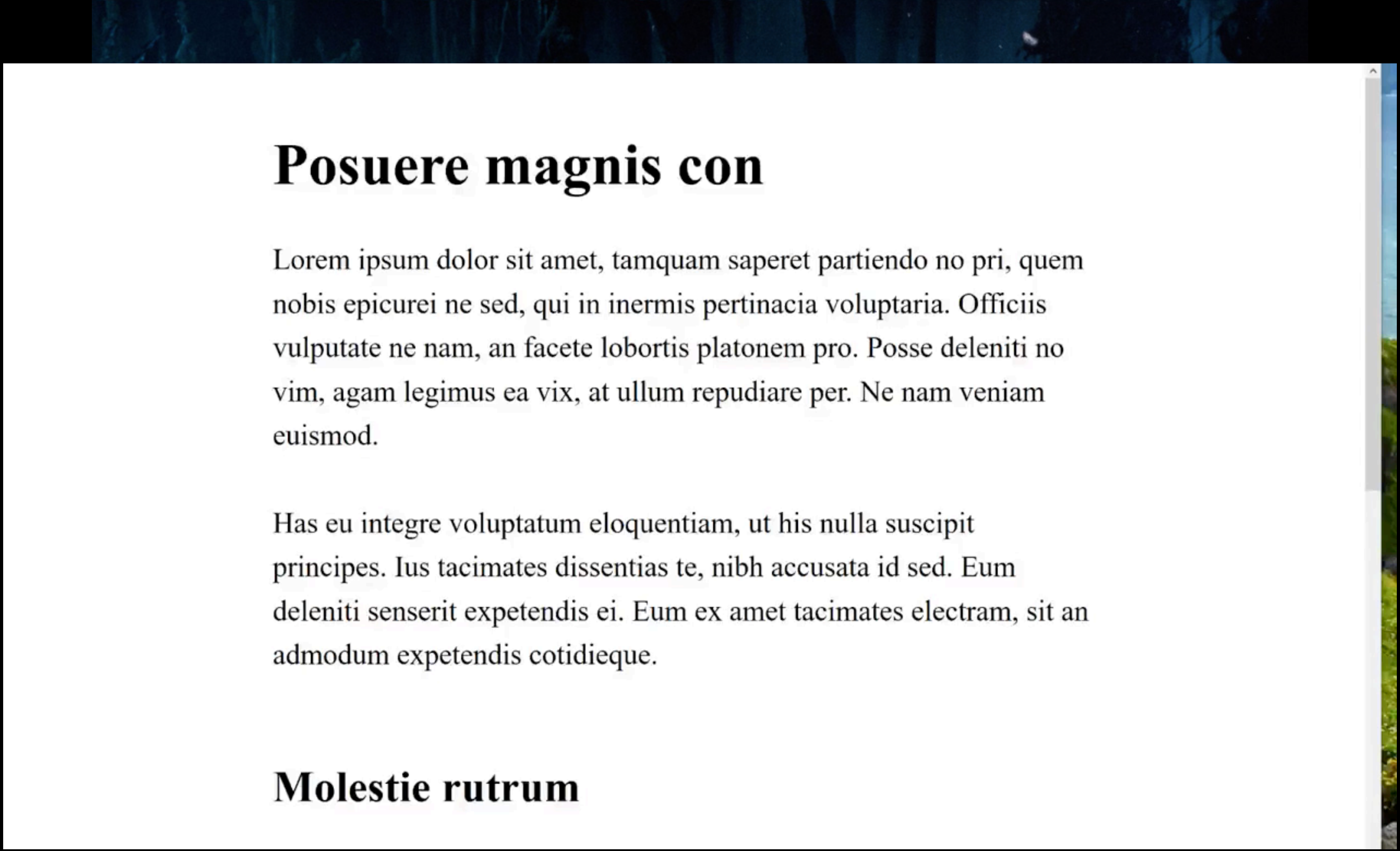
```
/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 {
    $map(576px: 1.5rem, 320px: 1rem, 992px: 3.375rem, 768px: 2.8125rem);
    @include polyFluidSizing('font-size',$map);
}
```

```
@media screen and (min-width: 768px) { /* Interpolation: 24px → 34px */
    h1 { font-size: calc(4.7787vw + 21.2444px); }
}

@media screen and (min-width: 992px) { /* Maximum font size: 34px */
    h1 { font-size: 34px; }
}
```

Fluid Typography, SCSS → CSS

```scss
/* 2:3 Perfect Fifth: 7.111, 10.667, 16 (i), 24, 36, 54  */
h1 {
    $map(320px: 1rem, 576px: 1.5rem, 768px: 2.8125rem, 992px: 3.375rem);
    @include polyFluidSizing('font-size',$map);
}
```

```css
@media screen and (min-width: 768px) { /* Interpolation: 24px → 34px */
    h1 { font-size: calc(4.7787vw + 21.2444px); }
}

@media screen and (min-width: 992px) { /* Maximum font size: 34px */
    h1 { font-size: 34px; }
}
```

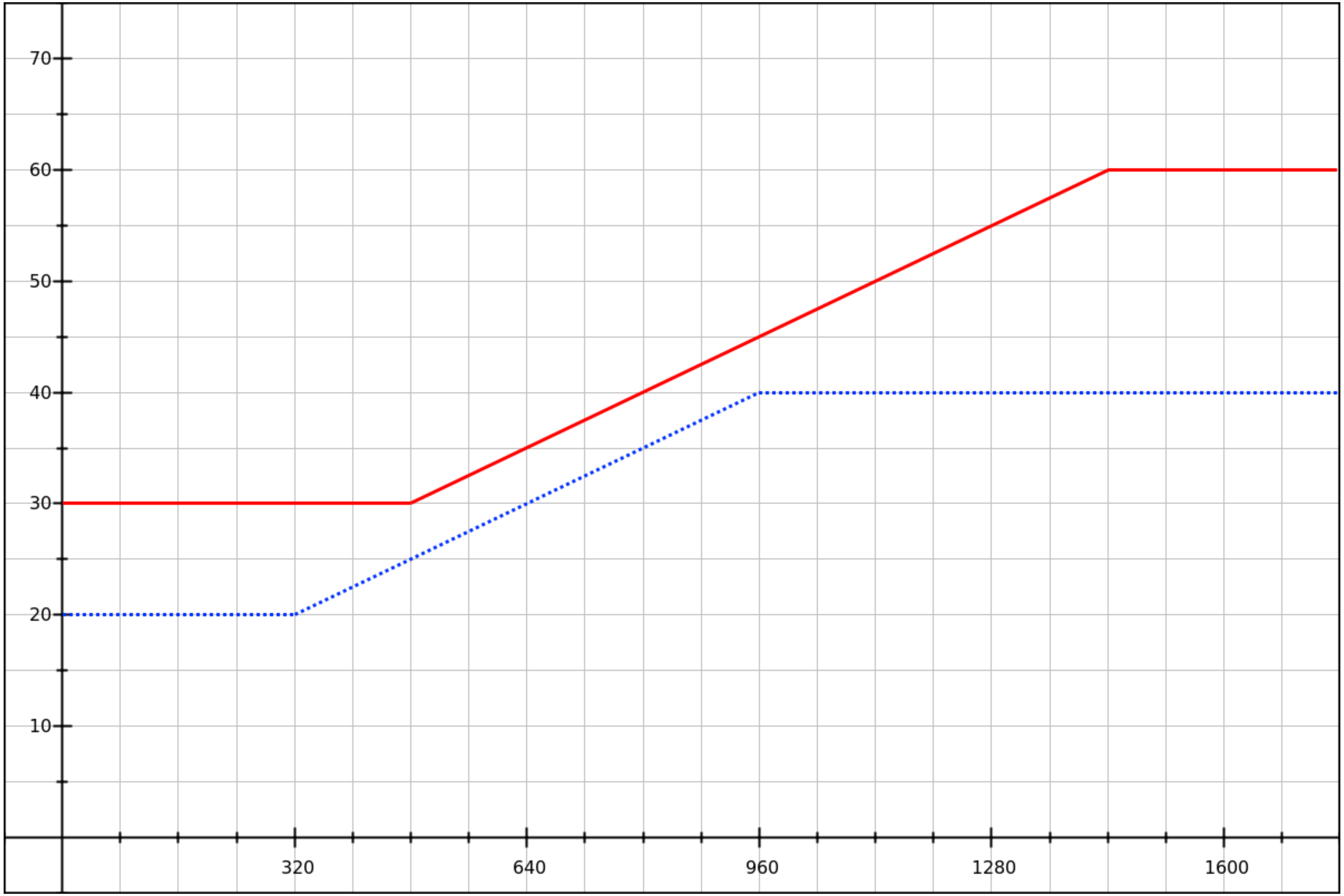


Posuere magnis con

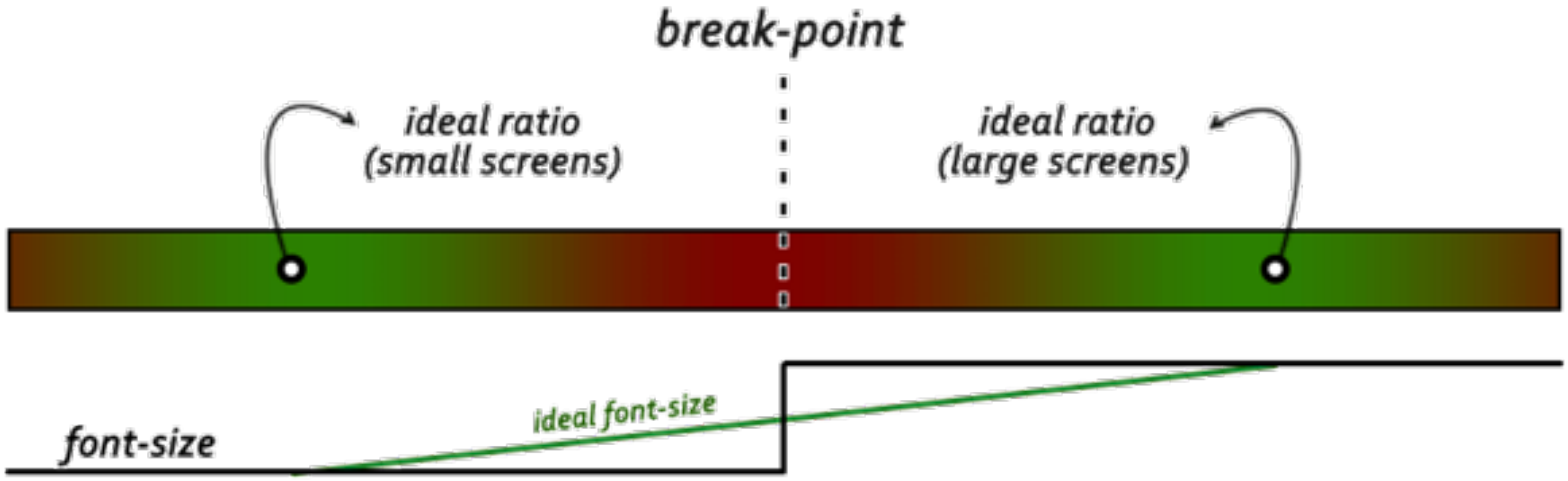
Lorem ipsum dolor sit amet, tamquam saperet partiendo no pri, quem nobis epicurei ne sed, qui in inermis pertinacia voluptaria. Officiis vulputate ne nam, an facete lobortis platonem pro. Posse deleniti no vim, agam legimus ea vix, at ullum repudiare per. Ne nam veniam euismod.

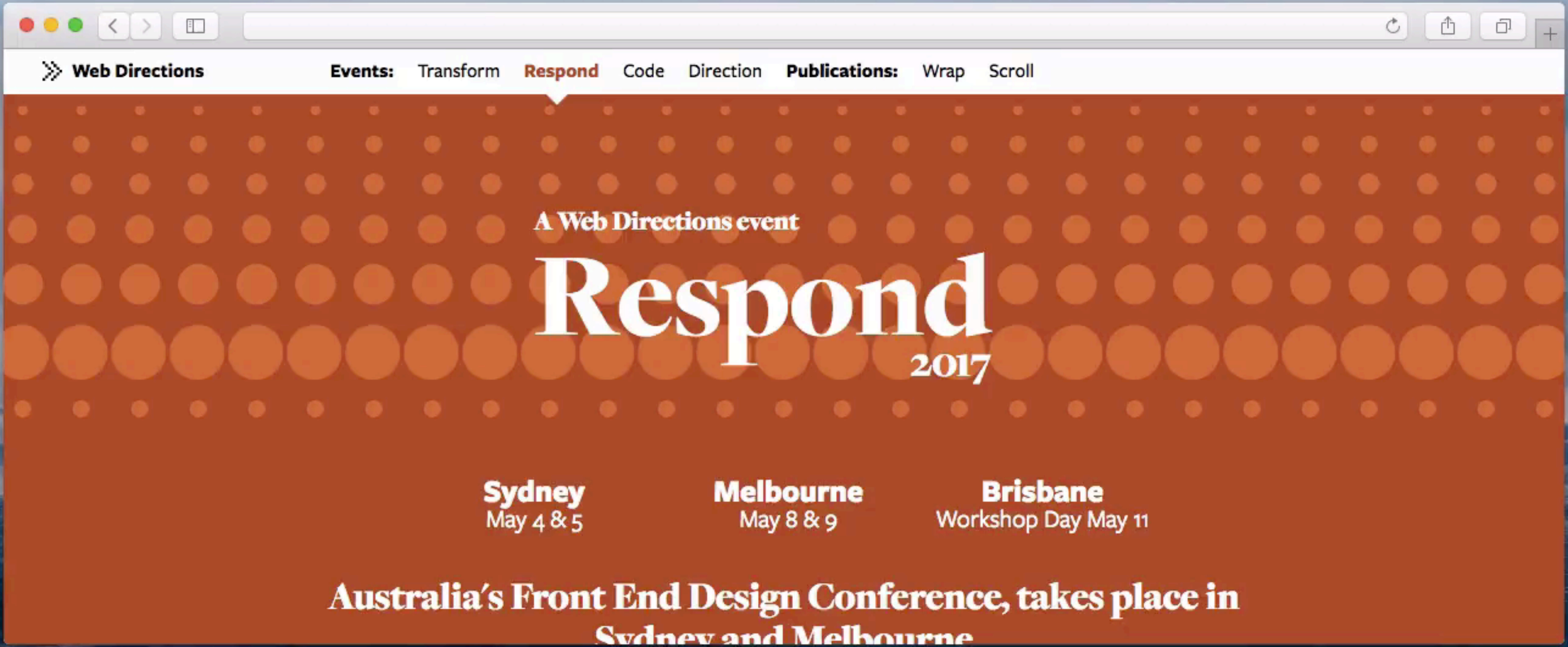
Has eu integre voluptatum eloquentiam, ut his nulla suscipit principes. Ius tacimates dissentias te, nibh accusata id sed. Eum deleniti senserit expetendis ei. Eum ex amet tacimates electram, sit an admodum expetendis cotidieque.

Molestie rutrum


70
60
50
40
30
20
10
320
640
960
1280
1600

~~Media queries.~~


break-point
ideal ratio
(small screens)
ideal ratio
(large screens)
ideal font-size
font-size



Web Directions

Events: Transform **Respond** Code Direction **Publications:** Wrap Scroll

A Web Directions event

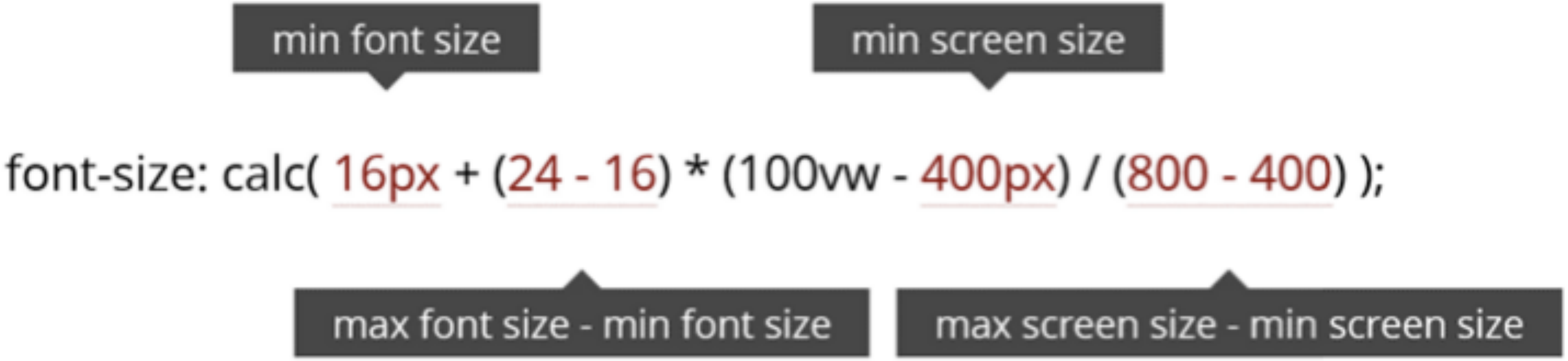
Respond
2017

Sydney
May 4 & 5

Melbourne
May 8 & 9

Brisbane
Workshop Day May 11

Australia's Front End Design Conference, takes place in Sydney and Melbourne



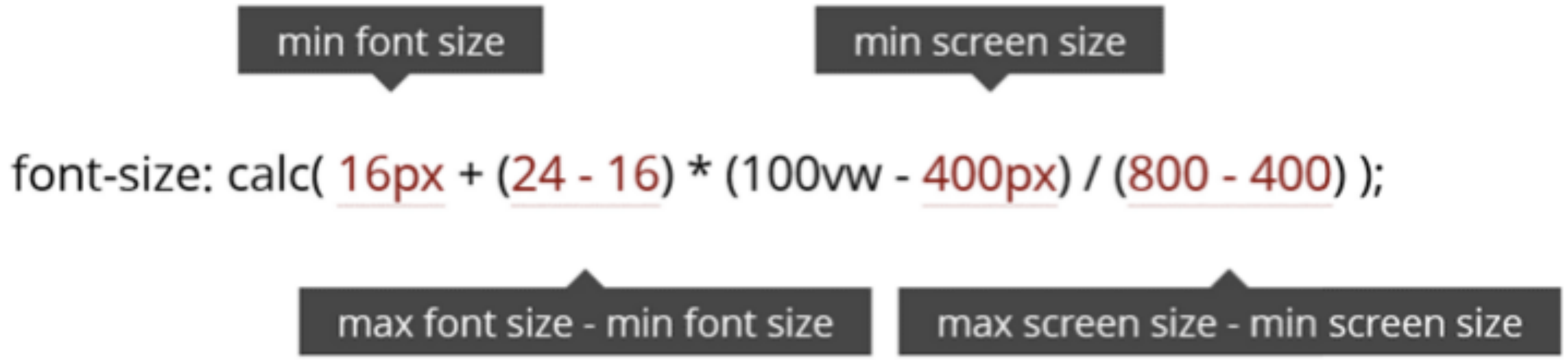
min font size

min screen size

```
font-size: calc( 16px + (24 - 16) * (100vw - 400px) / (800 - 400) );
```

max font size - min font size

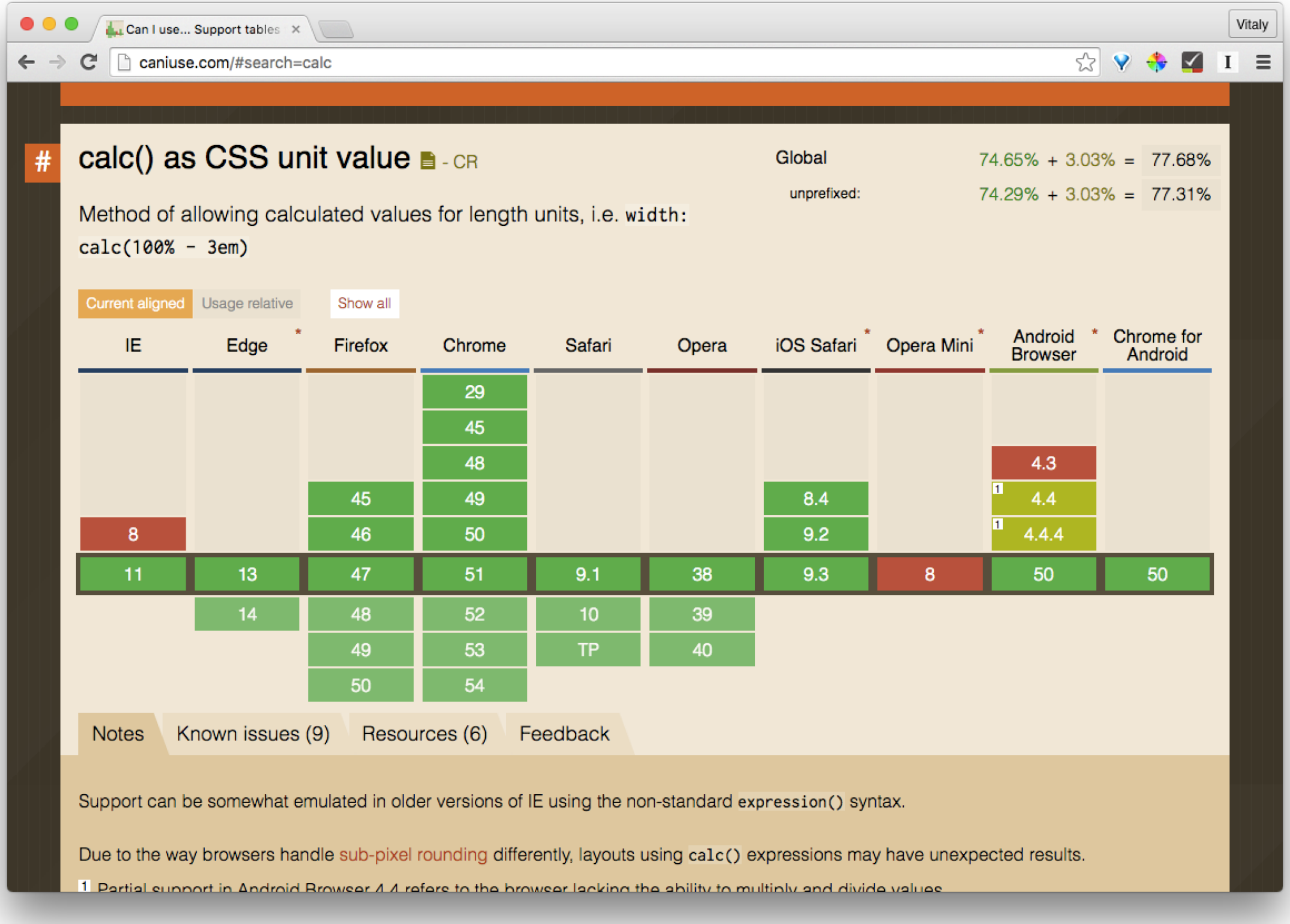
max screen size - min screen size



```
font-size: calc( 16px + (24 - 16) * (100vw - 400px) / (800 - 400) );
```

min font size
min screen size
max font size - min font size
max screen size - min screen size

> You choose the *min* and *max* font-size and the *screen sizes,* over which the font should scale and plug them into the equation. You can use any unit type including ems, rems or px.
>
> — *Mike Riethmuller*



calc() as CSS unit value - CR

Global 74.65% + 3.03% = 77.68%

unprefixed: 74.29% + 3.03% = 77.31%

Method of allowing calculated values for length units, i.e. `width: calc(100% - 3em)`

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			29						
			45						
			48					4.3	
		45	49			8.4		[1] 4.4	
8		46	50			9.2		[1] 4.4.4	
11	13	47	51	9.1	38	9.3	8	50	50
	14	48	52	10	39				
		49	53	TP	40				
		50	54						

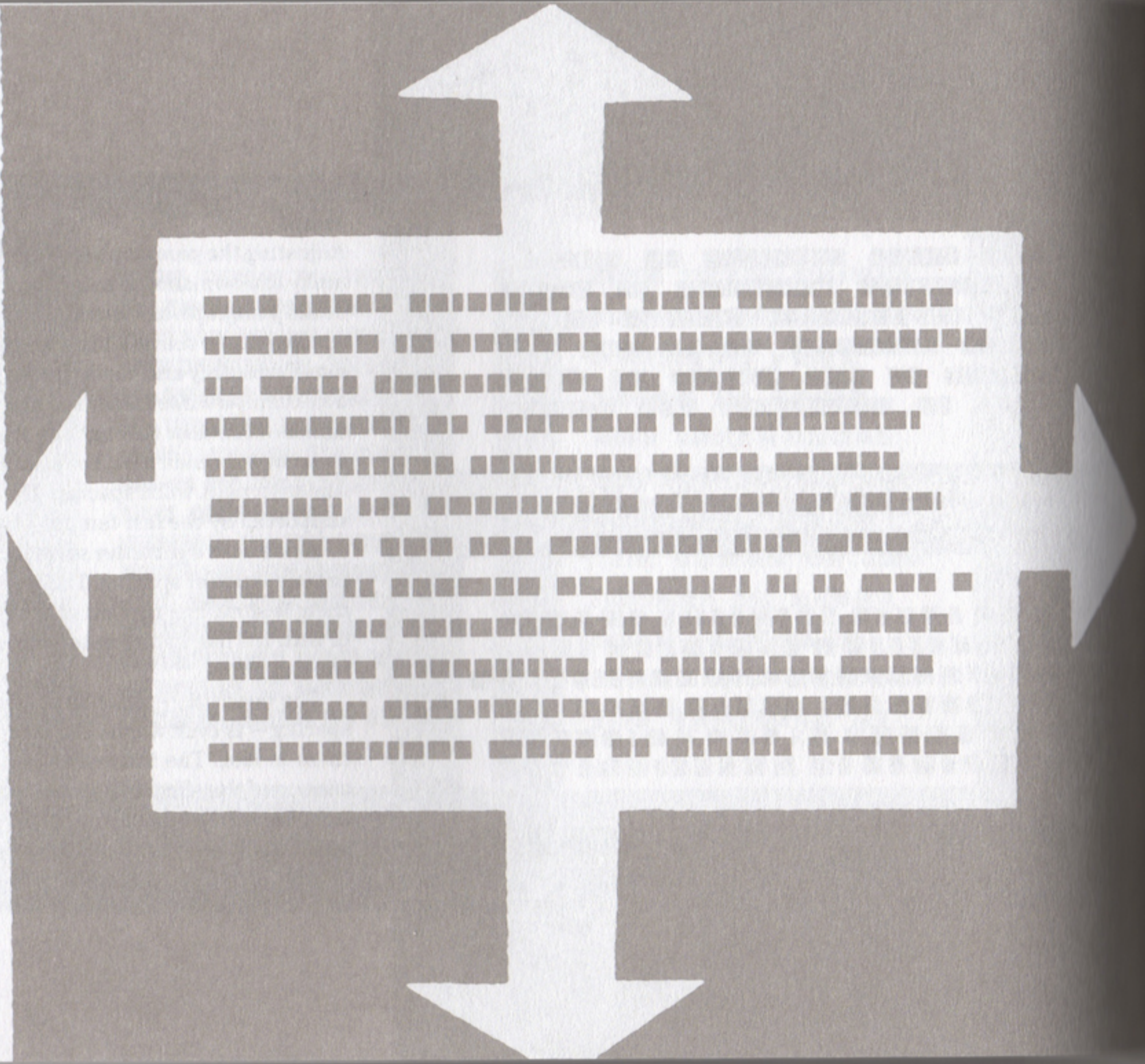
Notes | Known issues (9) | Resources (6) | Feedback

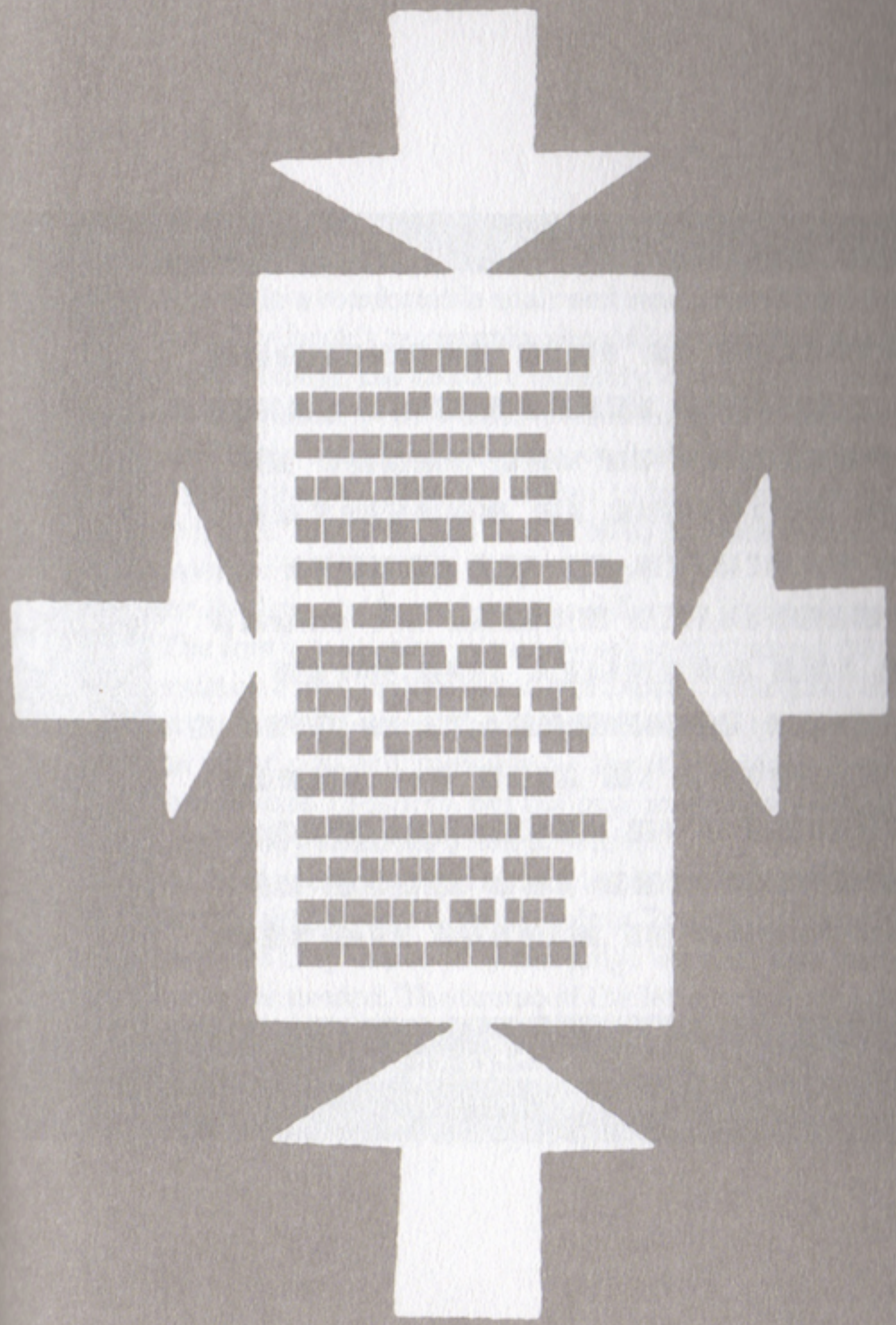
Support can be somewhat emulated in older versions of IE using the non-standard `expression()` syntax.

Due to the way browsers handle sub-pixel rounding differently, layouts using `calc()` expressions may have unexpected results.

[1] Partial support in Android Browser 4.4 refers to the browser lacking the ability to multiply and divide values.

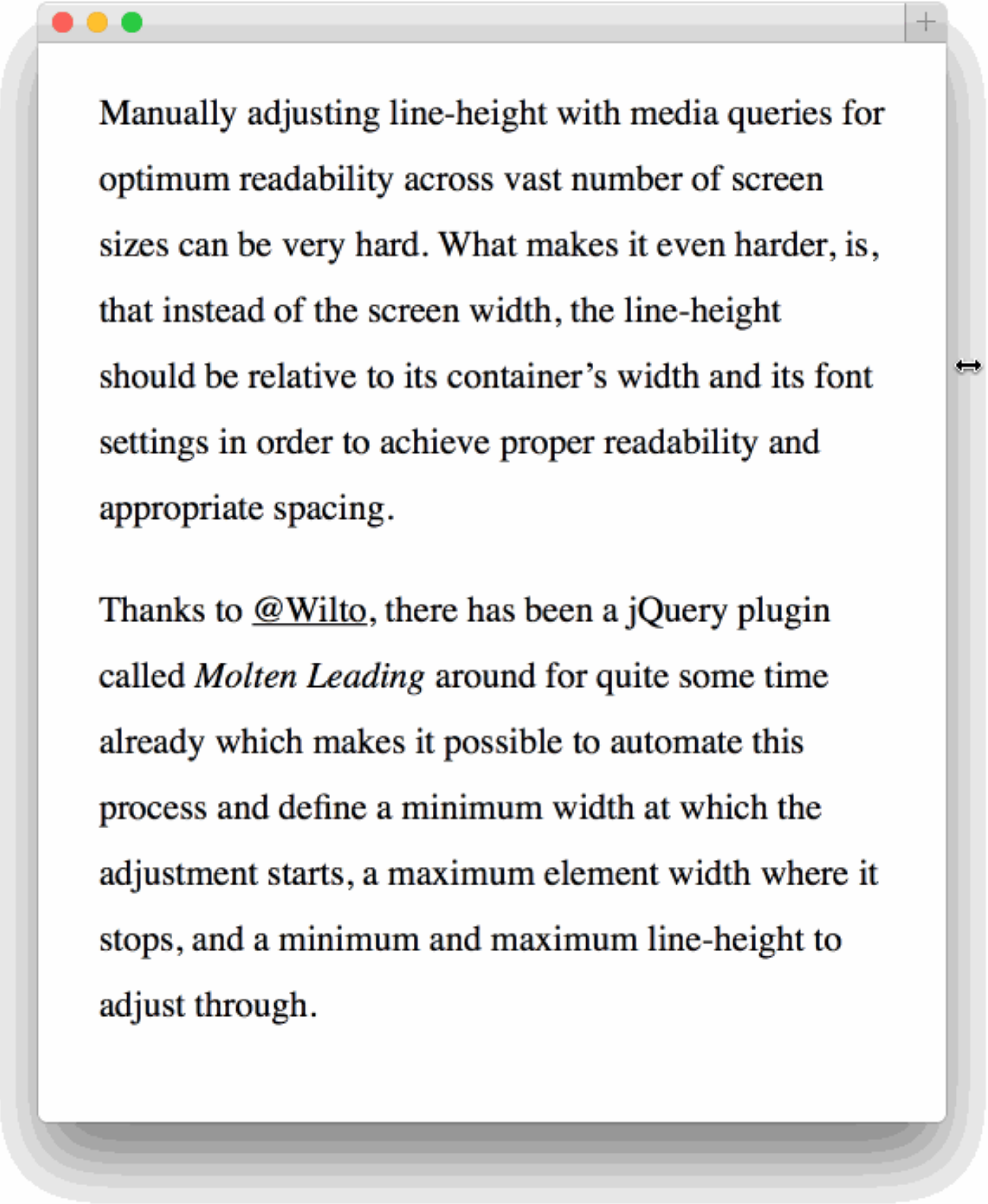
D
AND





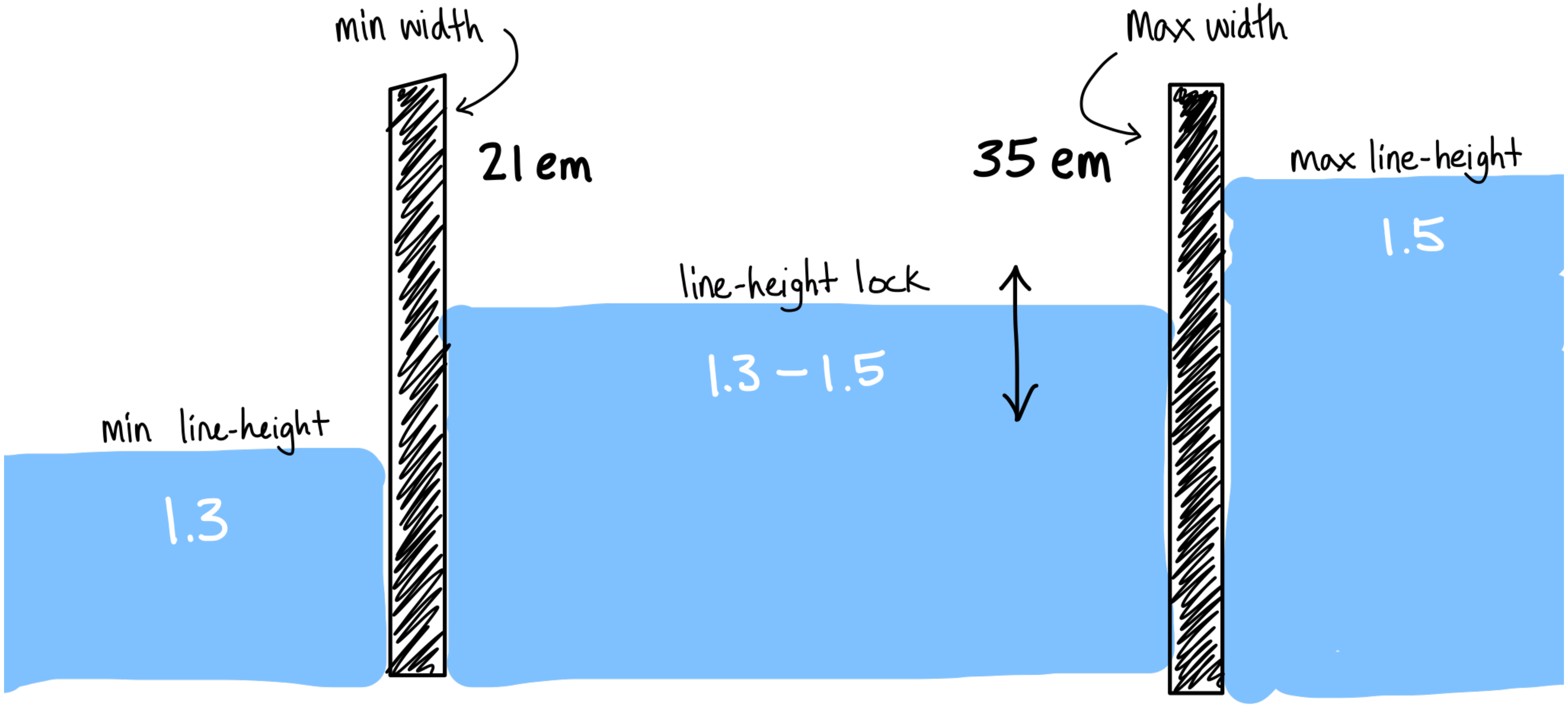
Colu
The t
grapl
to th
shoul
lines
the te

Bu
space
ing a
adjus
differ
expa
and t



Manually adjusting line-height with media queries for optimum readability across vast number of screen sizes can be very hard. What makes it even harder, is, that instead of the screen width, the line-height should be relative to its container's width and its font settings in order to achieve proper readability and appropriate spacing.

Thanks to @Wilto, there has been a jQuery plugin called *Molten Leading* around for quite some time already which makes it possible to automate this process and define a minimum width at which the adjustment starts, a maximum element width where it stops, and a minimum and maximum line-height to adjust through.


min width
21 em
Max width
35 em
max line-height
1.5
line-height lock
1.3 – 1.5
min line-height
1.3

And here's what a CSS lock looks like in code:

```
line-height: calc(1.3em + (1.5 - 1.3) * ((100vw - 21em)/(35 - 21)));
```

To understand how the formula works within **calc()**, we're going to work through it backwards.

1. See the very last part? **35–21**. That gives us the *full range of our paragraph's width*. It resolves to 14, because **14em** is the difference between our paragraph's width at its most narrow and most wide.

2. To the left of that, we've got **100vw–21em**. Because of the way CSS calc works, this resolves to an em-based value—and gives us a numerator to place above the 14em we already figured out. So, for example, let's say the viewport width (100vw) is equivalent to 34em. 34em–21em = 13em. *Note that the viewport unit in this step is our secret sauce. The fact that this value can change dynamically with browser window width is what makes a dynamic line-height value possible.*

3. So the whole expression to the right of the multiplication sign gets distilled down to this: 13em / 14em, or **0.928571429em**. Think of this as how close we are to the "upper gate" of our lock. If it's near zero, we're close to the lower gate. If it's near one, we're close to the upper gate.

4. Moving to the left of the multiplication sign, we compute the difference between our maximum and minimum line heights. 1.5–1.3 = **0.2**. This gives us *the full range of our fluid line height*.

5. Now we multiply the full range of our fluid line height (step 4) by how far along we are toward the upper gate of our lock (step 3):
0.2 * 0.928571429em = **0.185714286em**.

And here's what a CSS lock looks like in code:

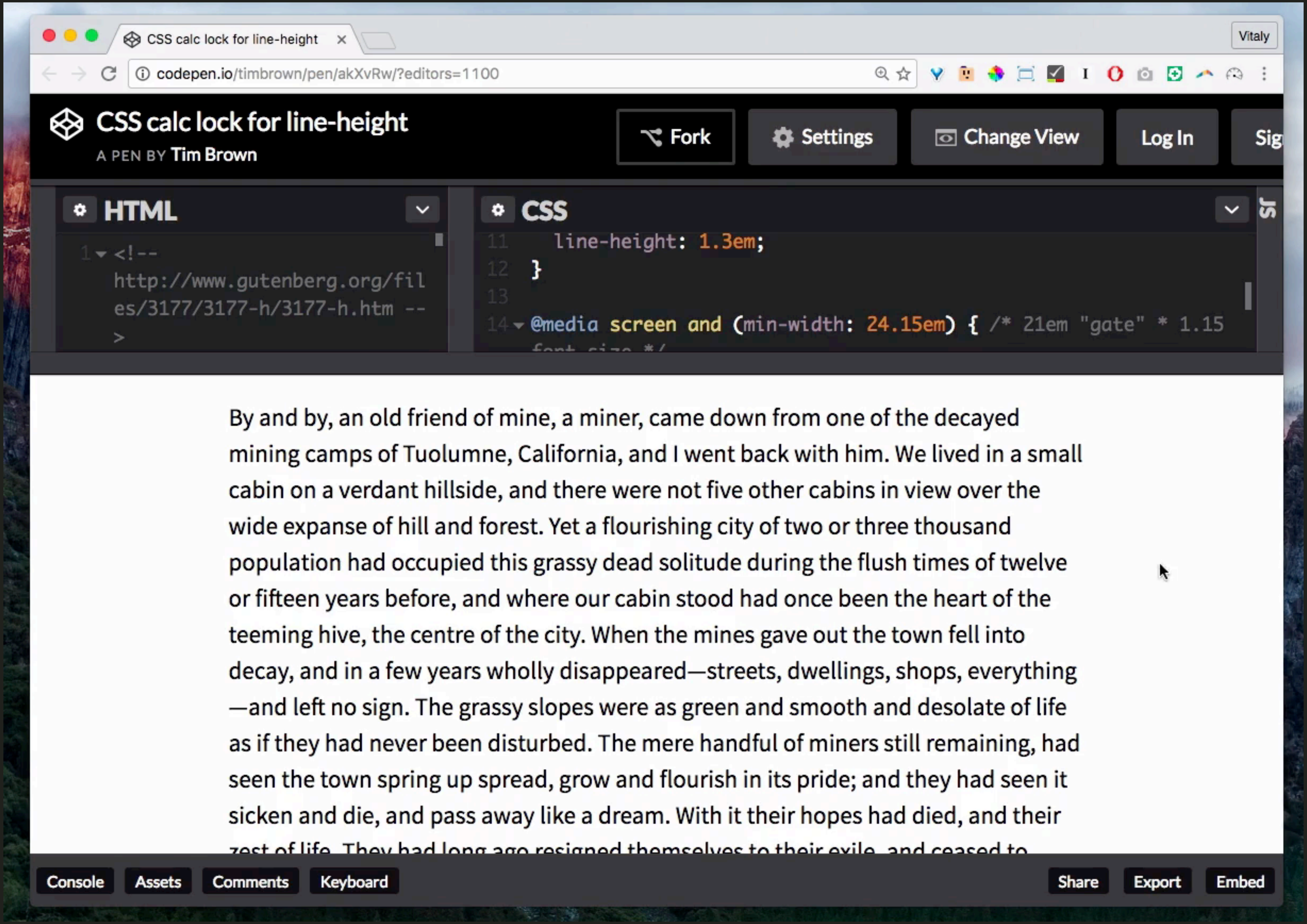
```
line-height: calc(1.3em + (1.5 - 1.3) * ((100vw - 21em)/(35 - 21)));
```

To understand how the formula works within **calc()**, we're going to work through it backwards.

1. See the very last part? **35–21**. That gives us the *full range of our paragraph's width*. It resolves to 14, because **14em** is the difference between our paragraph's width at its most narrow and most wide.

2. To the left of that, we've got **100vw–21em**. Because of the way CSS calc works, this resolves to an em-based value—and gives us a numerator to place above the 14em we already figured out. So, for example, let's say the viewport width (100vw) is equivalent to 34em. 34em–21em = 13em. *Note that the viewport unit in this step is our secret sauce. The fact that this value can change dynamically with browser window width is what makes a dynamic line-height value possible.*

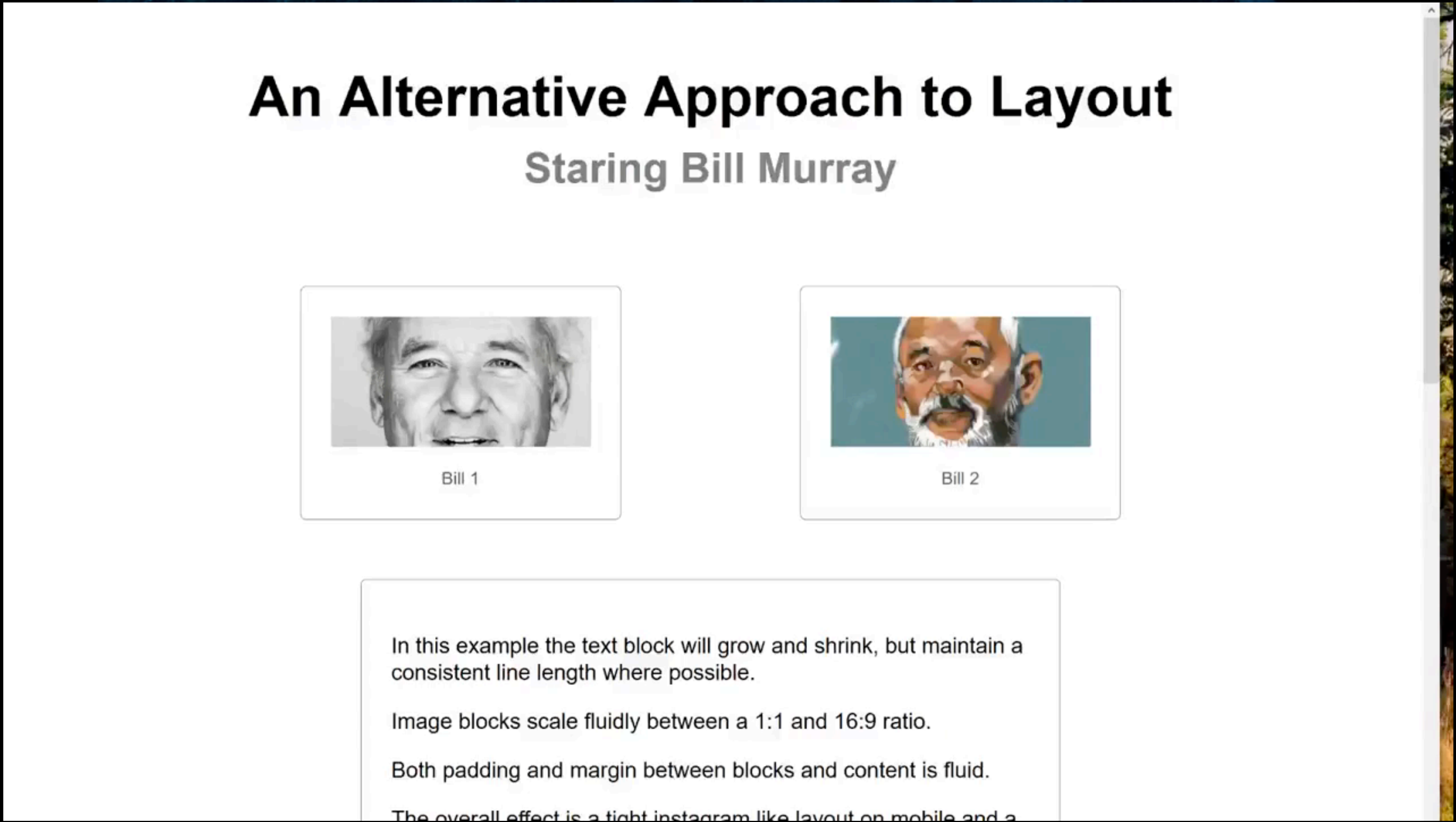
3. So the whole expression to the right of the multiplication sign gets distilled down to this: 13em / 14em, or **0.928571429em**. Think of this as how close we are to the "upper gate" of our lock. If it's near zero, we're close to the lower gate. If it's near one, we're close to the



```
<!--
http://www.gutenberg.org/files/3177/3177-h/3177-h.htm --
>
```

```
  line-height: 1.3em;
}

@media screen and (min-width: 24.15em) { /* 21em "gate" * 1.15
font-size */
```

By and by, an old friend of mine, a miner, came down from one of the decayed mining camps of Tuolumne, California, and I went back with him. We lived in a small cabin on a verdant hillside, and there were not five other cabins in view over the wide expanse of hill and forest. Yet a flourishing city of two or three thousand population had occupied this grassy dead solitude during the flush times of twelve or fifteen years before, and where our cabin stood had once been the heart of the teeming hive, the centre of the city. When the mines gave out the town fell into decay, and in a few years wholly disappeared—streets, dwellings, shops, everything—and left no sign. The grassy slopes were as green and smooth and desolate of life as if they had never been disturbed. The mere handful of miners still remaining, had seen the town spring up spread, grow and flourish in its pride; and they had seen it sicken and die, and pass away like a dream. With it their hopes had died, and their zest of life. They had long ago resigned themselves to their exile, and ceased to



An Alternative Approach to Layout

Staring Bill Murray

Bill 1

Bill 2

In this example the text block will grow and shrink, but maintain a consistent line length where possible.

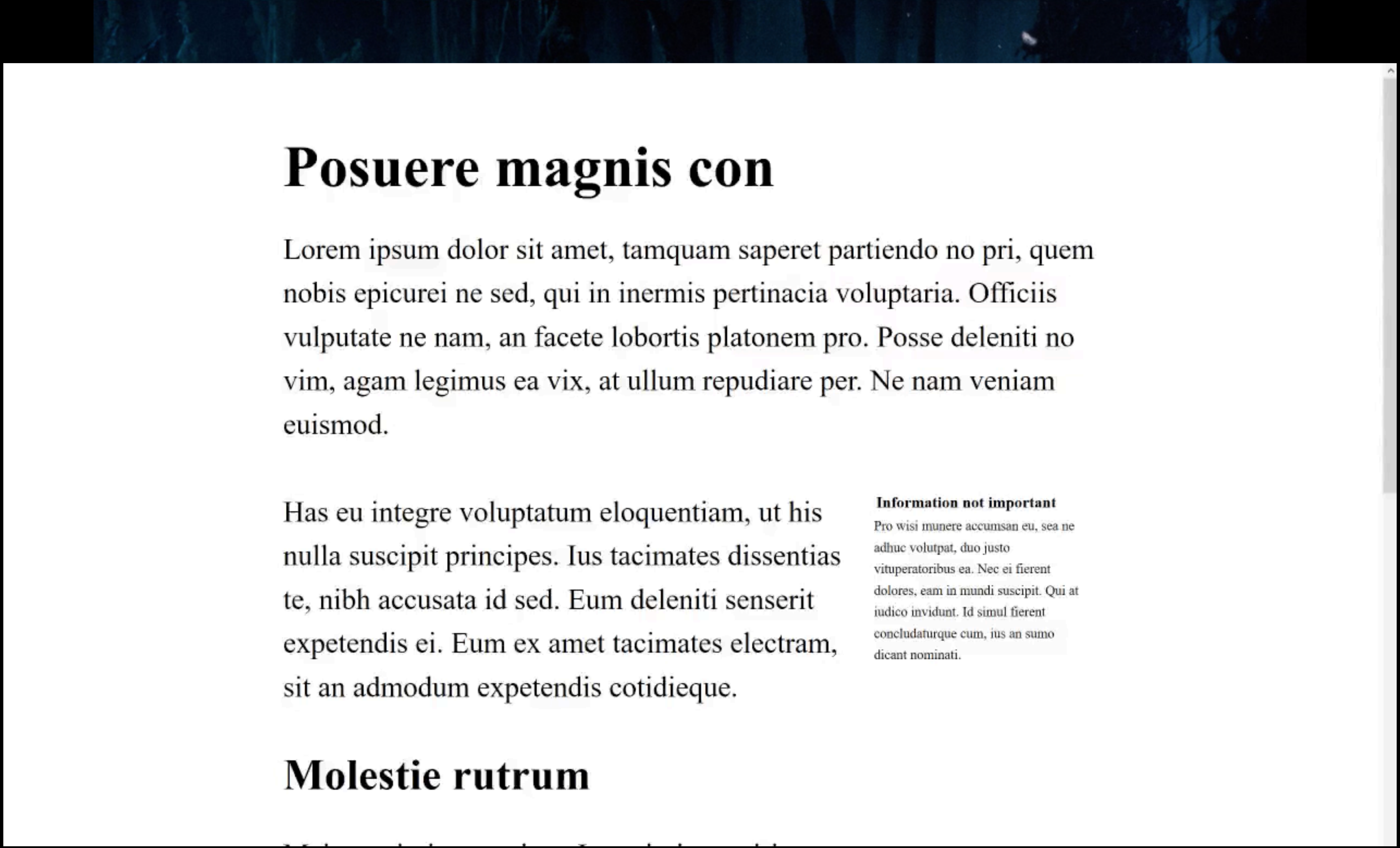
Image blocks scale fluidly between a 1:1 and 16:9 ratio.

Both padding and margin between blocks and content is fluid.

The overall effect is a tight instagram like layout on mobile and a

Fluid Behavior, Turn OFF

```css
html {
    font-size: calc(100% + 8 * ((100vw - 400px) / 400);
}

.testimonial { /* Use rem to adjust font-sizes */
    width: 4.5rem;
    padding: 0.5rem;
    font-size: 1.75rem;
}

/* Turn OFF fluid behavior in a fixed container */
.fixed-container { font-size: 18px; }
.fixed-container .testimonial { font-size: 2em; }
```



Posuere magnis con

Lorem ipsum dolor sit amet, tamquam saperet partiendo no pri, quem nobis epicurei ne sed, qui in inermis pertinacia voluptaria. Officiis vulputate ne nam, an facete lobortis platonem pro. Posse deleniti no vim, agam legimus ea vix, at ullum repudiare per. Ne nam veniam euismod.

Has eu integre voluptatum eloquentiam, ut his nulla suscipit principes. Ius tacimates dissentias te, nibh accusata id sed. Eum deleniti senserit expetendis ei. Eum ex amet tacimates electram, sit an admodum expetendis cotidieque.

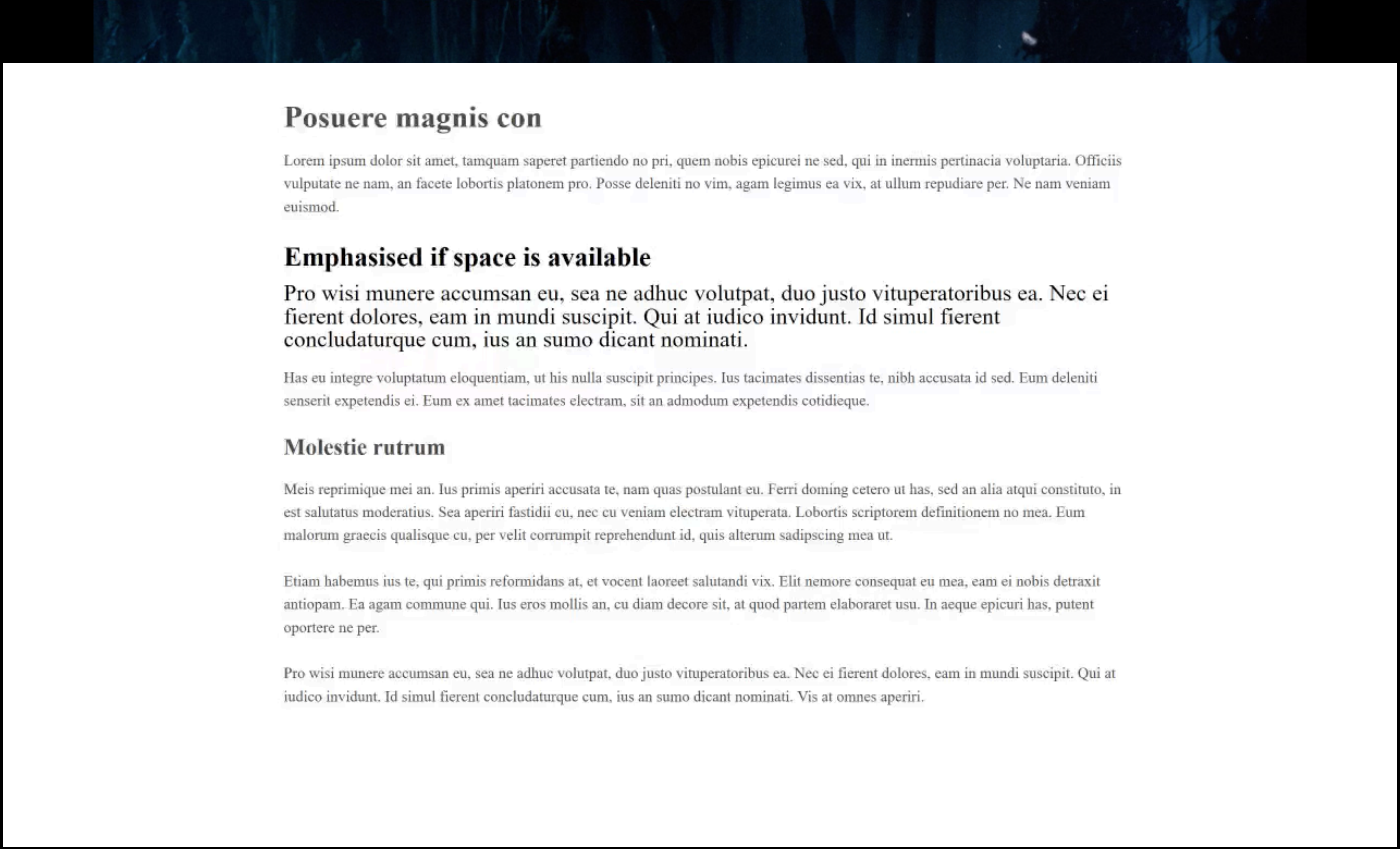
Information not important

Pro wisi munere accumsan eu, sea ne adhuc volutpat, duo justo vituperatoribus ea. Nec ei fierent dolores, eam in mundi suscipit. Qui at iudico invidunt. Id simul fierent concludaturque cum, ius an sumo dicant nominati.

Molestie rutrum

Fluid Behavior, Turn ON

```
html {
    font-size: 100%;
}

.testimonial { /* Use rem to adjust font-sizes */
    width: 4.5rem;
    padding: 0.5rem;
    font-size: 1.75rem;
}

/* Turn ON fluid behavior in a fluid container */
.fluid-container { font-size: calc(100% + 8 * ((100vw - 400px) / 400); }
.fluid-container .testimonial { font-size: 2em; }
```



Posuere magnis con

Lorem ipsum dolor sit amet, tamquam saperet partiendo no pri, quem nobis epicurei ne sed, qui in inermis pertinacia voluptaria. Officiis vulputate ne nam, an facete lobortis platonem pro. Posse deleniti no vim, agam legimus ea vix, at ullum repudiare per. Ne nam veniam euismod.

Emphasised if space is available

Pro wisi munere accumsan eu, sea ne adhuc volutpat, duo justo vituperatoribus ea. Nec ei fierent dolores, eam in mundi suscipit. Qui at iudico invidunt. Id simul fierent concludaturque cum, ius an sumo dicant nominati.

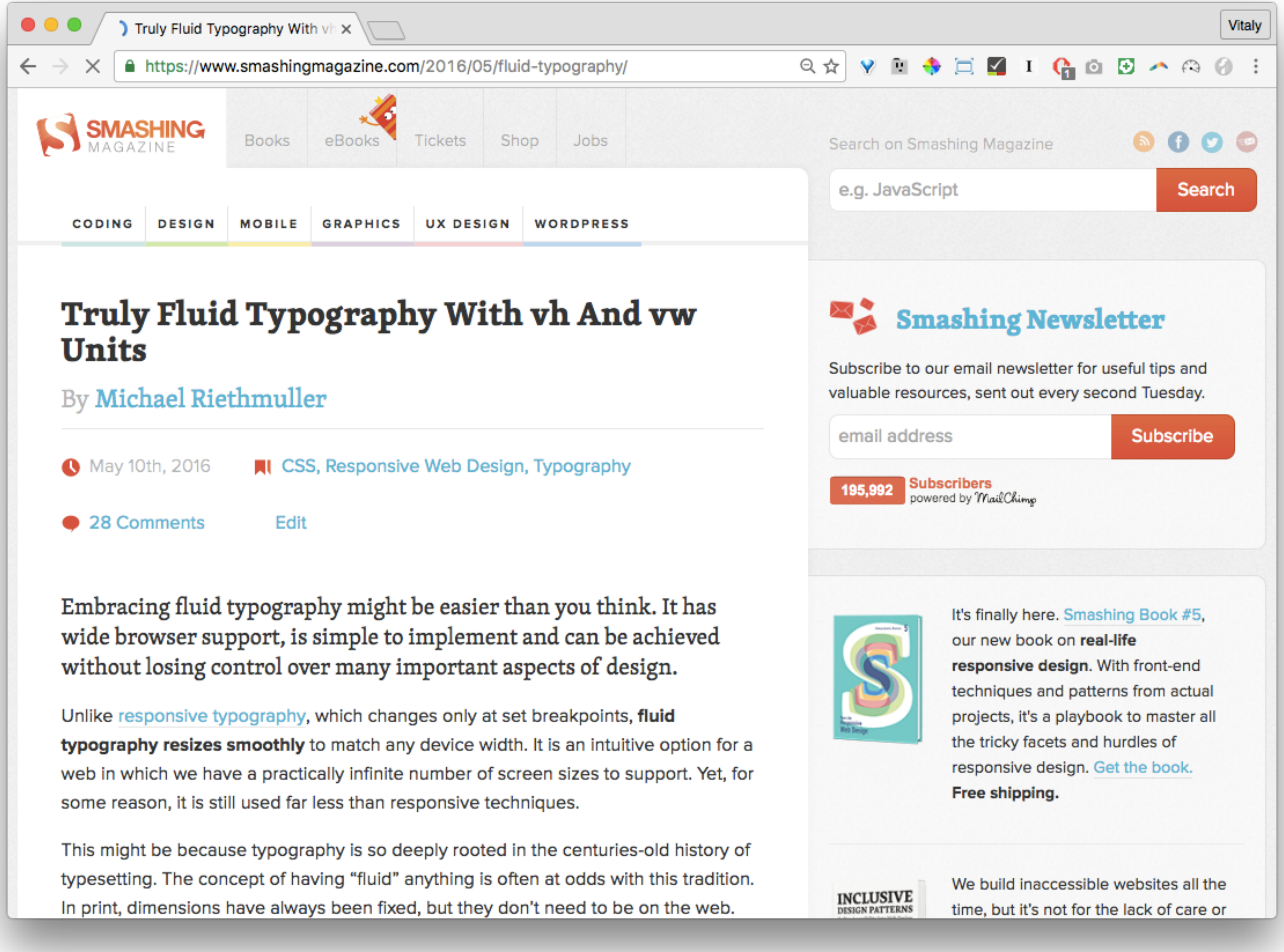
Has eu integre voluptatum eloquentiam, ut his nulla suscipit principes. Ius tacimates dissentias te, nibh accusata id sed. Eum deleniti senserit expetendis ei. Eum ex amet tacimates electram, sit an admodum expetendis cotidieque.

Molestie rutrum

Meis reprimique mei an. Ius primis aperiri accusata te, nam quas postulant eu. Ferri doming cetero ut has, sed an alia atqui constituto, in est salutatus moderatius. Sea aperiri fastidii cu, nec cu veniam electram vituperata. Lobortis scriptorem definitionem no mea. Eum malorum graecis qualisque cu, per velit corrumpit reprehendunt id, quis alterum sadipscing mea ut.

Etiam habemus ius te, qui primis reformidans at, et vocent laoreet salutandi vix. Elit nemore consequat eu mea, eam ei nobis detraxit antiopam. Ea agam commune qui. Ius eros mollis an, cu diam decore sit, at quod partem elaboraret usu. In aeque epicuri has, putent oportere ne per.

Pro wisi munere accumsan eu, sea ne adhuc volutpat, duo justo vituperatoribus ea. Nec ei fierent dolores, eam in mundi suscipit. Qui at iudico invidunt. Id simul fierent concludaturque cum, ius an sumo dicant nominati. Vis at omnes aperiri.



Truly Fluid Typography With vh And vw Units

By Michael Riethmuller

May 10th, 2016 CSS, Responsive Web Design, Typography

28 Comments Edit

Embracing fluid typography might be easier than you think. It has wide browser support, is simple to implement and can be achieved without losing control over many important aspects of design.

Unlike responsive typography, which changes only at set breakpoints, **fluid typography resizes smoothly** to match any device width. It is an intuitive option for a web in which we have a practically infinite number of screen sizes to support. Yet, for some reason, it is still used far less than responsive techniques.

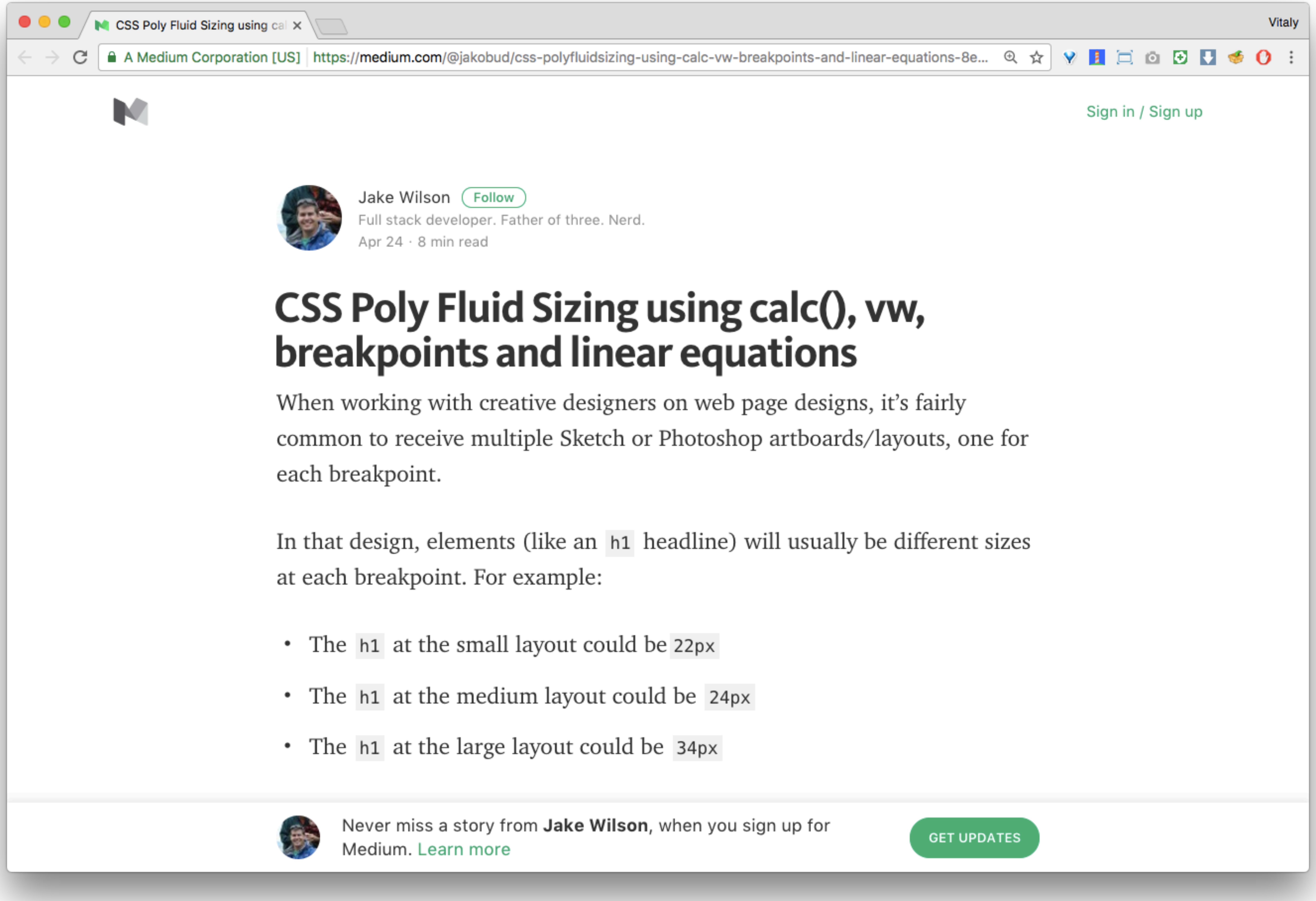
This might be because typography is so deeply rooted in the centuries-old history of typesetting. The concept of having “fluid” anything is often at odds with this tradition. In print, dimensions have always been fixed, but they don’t need to be on the web.

Smashing Newsletter

Subscribe to our email newsletter for useful tips and valuable resources, sent out every second Tuesday.

It's finally here. Smashing Book #5, our new book on **real-life responsive design**. With front-end techniques and patterns from actual projects, it's a playbook to master all the tricky facets and hurdles of responsive design. Get the book. **Free shipping.**

We build inaccessible websites all the time, but it's not for the lack of care or


CSS Poly Fluid Sizing using cal
Vitaly
A Medium Corporation [US]
https://medium.com/@jakobud/css-polyfluidsizing-using-calc-vw-breakpoints-and-linear-equations-8e...
Sign in / Sign up
Follow
GET UPDATES

Jake Wilson

Full stack developer. Father of three. Nerd.

Apr 24 · 8 min read

CSS Poly Fluid Sizing using calc(), vw, breakpoints and linear equations

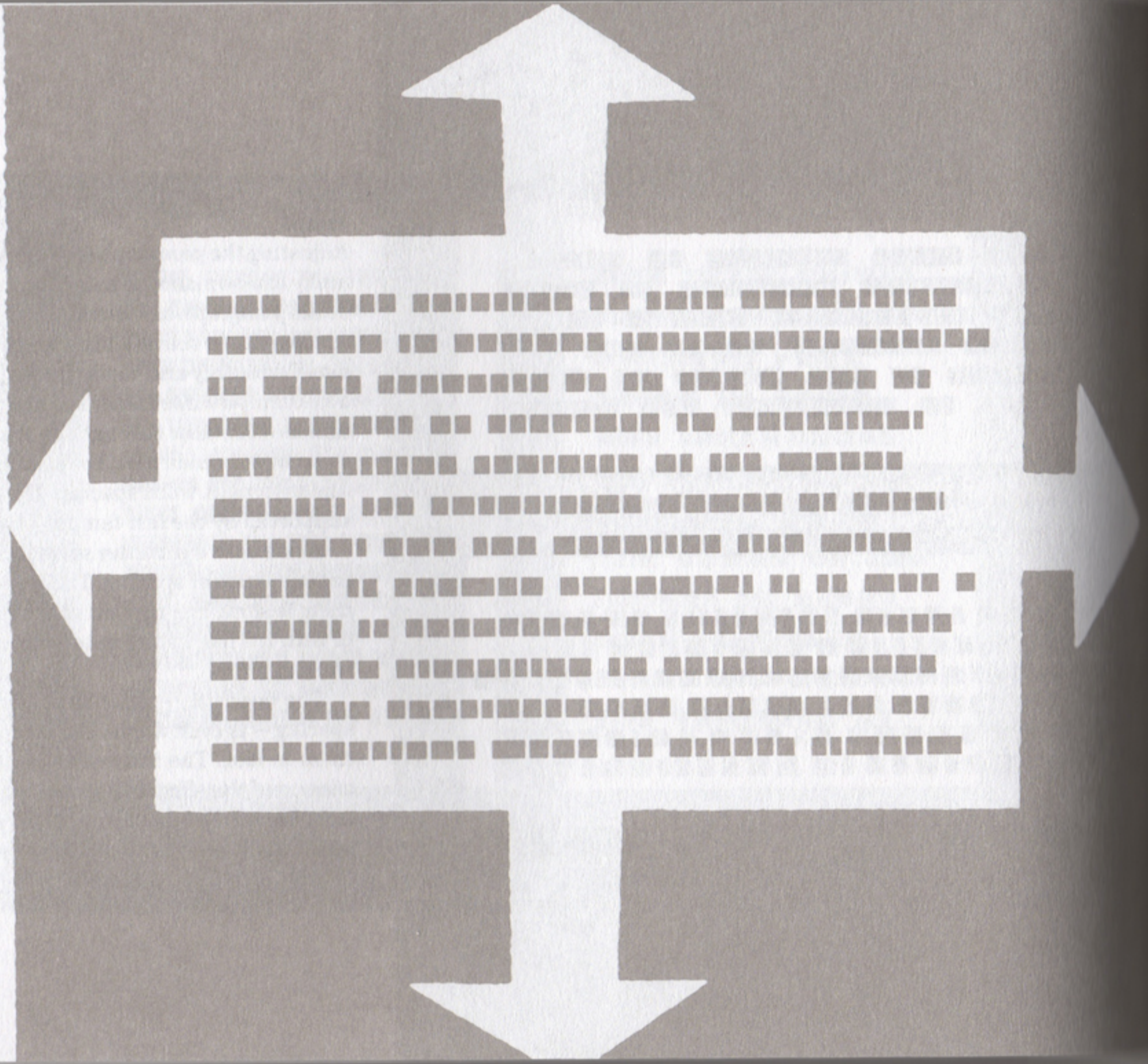
When working with creative designers on web page designs, it's fairly common to receive multiple Sketch or Photoshop artboards/layouts, one for each breakpoint.

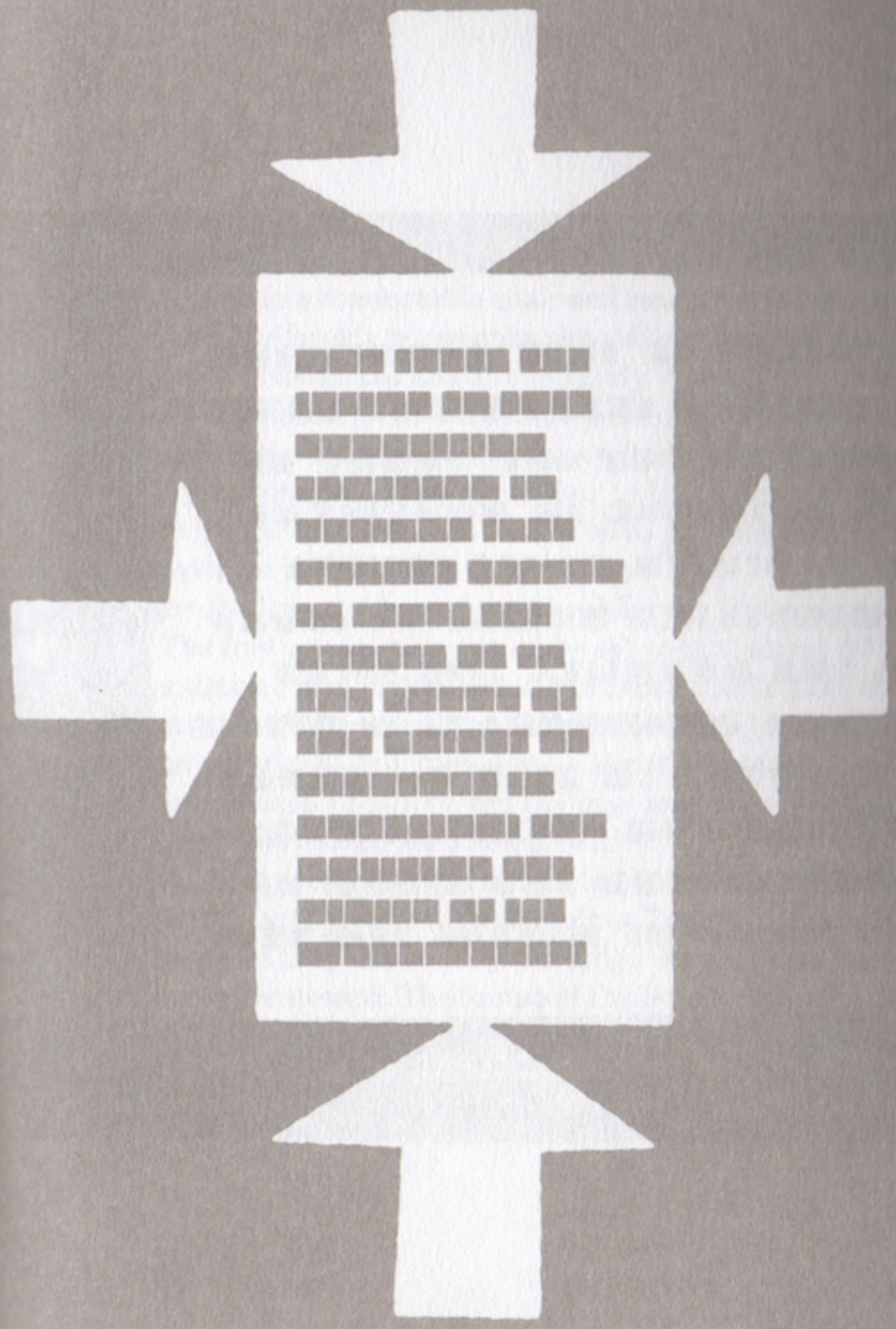
In that design, elements (like an `h1` headline) will usually be different sizes at each breakpoint. For example:

- The `h1` at the small layout could be `22px`
- The `h1` at the medium layout could be `24px`
- The `h1` at the large layout could be `34px`

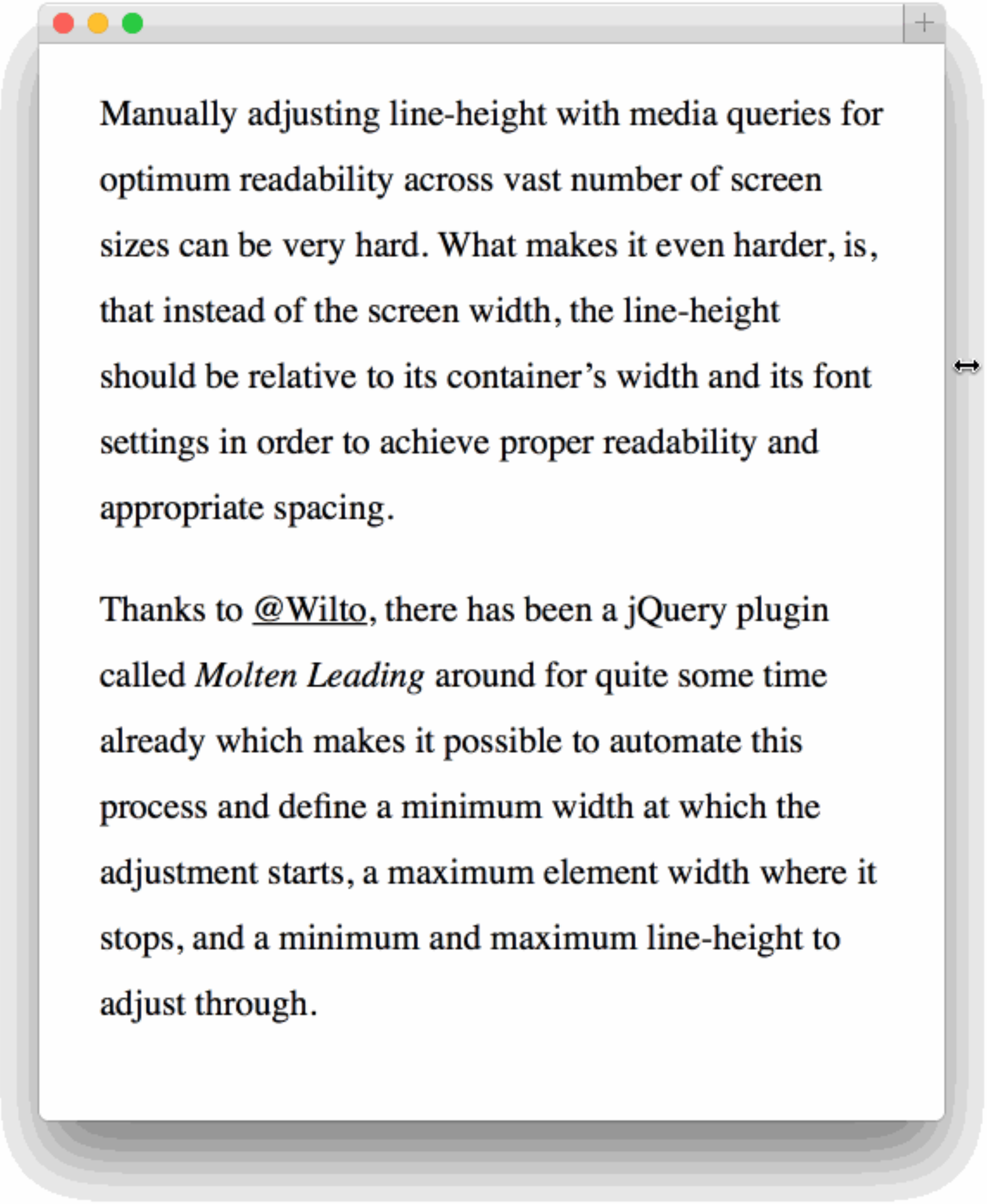
Never miss a story from **Jake Wilson**, when you sign up for Medium. Learn more

D
AND



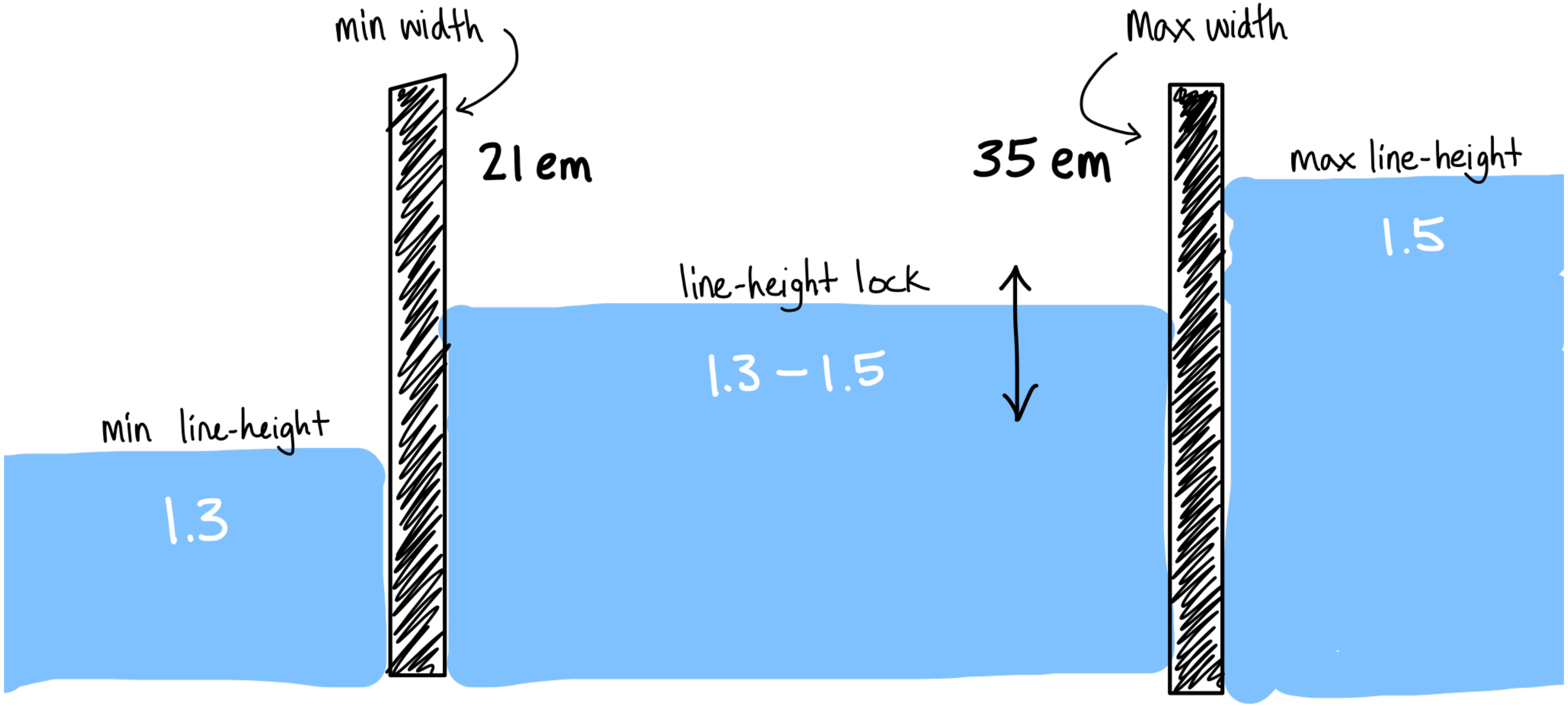


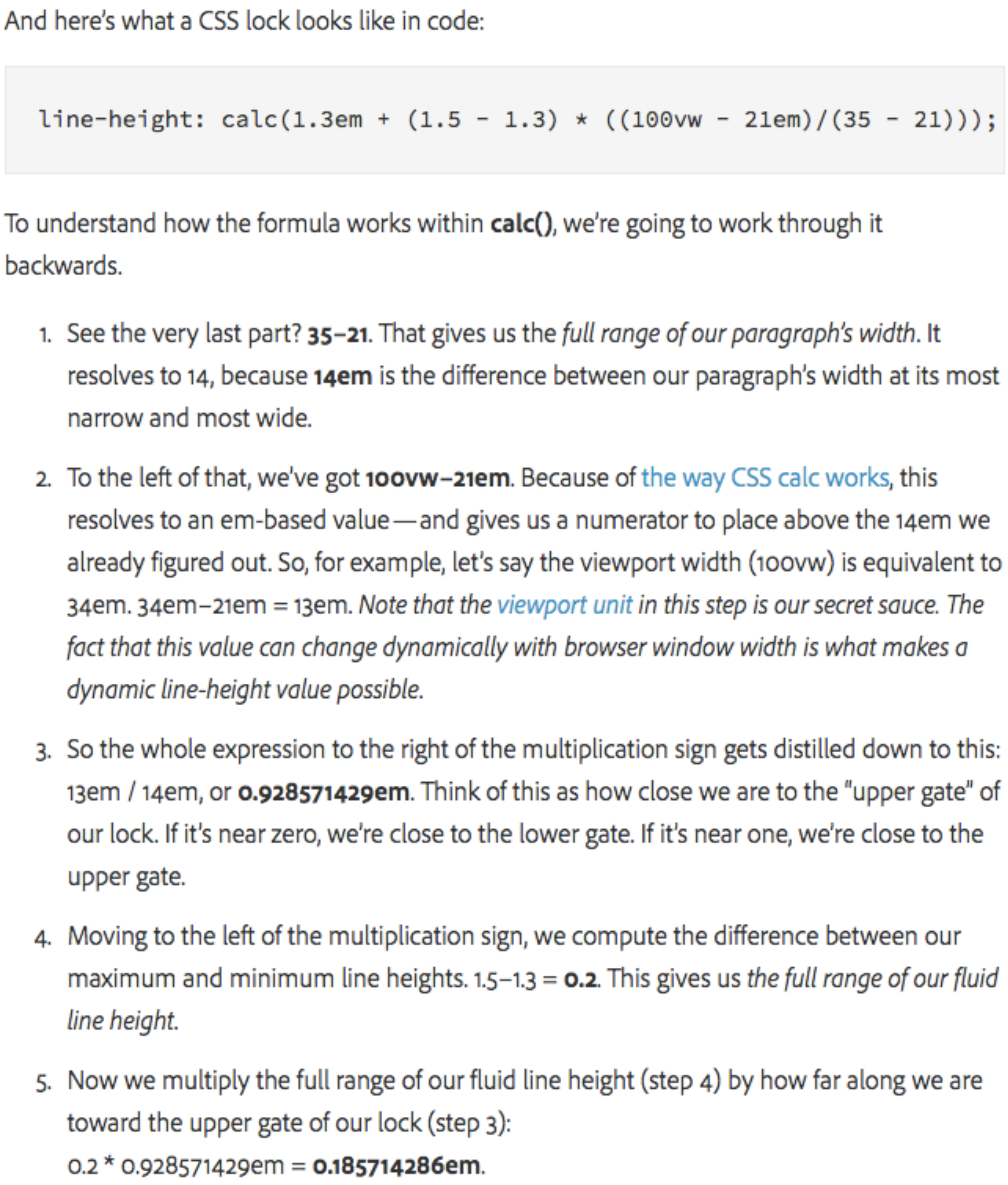
Colu
The t
graph
to th
shoul
lines
the t
Bu
space
ing a
adjus
differ
expa
and t



Manually adjusting line-height with media queries for optimum readability across vast number of screen sizes can be very hard. What makes it even harder, is, that instead of the screen width, the line-height should be relative to its container’s width and its font settings in order to achieve proper readability and appropriate spacing.

Thanks to @Wilto, there has been a jQuery plugin called *Molten Leading* around for quite some time already which makes it possible to automate this process and define a minimum width at which the adjustment starts, a maximum element width where it stops, and a minimum and maximum line-height to adjust through.


min width
21 em
Max width
35 em
max line-height
1.5
line-height lock
1.3 – 1.5
min line-height
1.3



And here's what a CSS lock looks like in code:

```
line-height: calc(1.3em + (1.5 - 1.3) * ((100vw - 21em)/(35 - 21)));
```

To understand how the formula works within **calc()**, we're going to work through it backwards.

1. See the very last part? **35–21**. That gives us the *full range of our paragraph's width*. It resolves to 14, because **14em** is the difference between our paragraph's width at its most narrow and most wide.
2. To the left of that, we've got **100vw–21em**. Because of the way CSS calc works, this resolves to an em-based value—and gives us a numerator to place above the 14em we already figured out. So, for example, let's say the viewport width (100vw) is equivalent to 34em. 34em–21em = 13em. *Note that the viewport unit in this step is our secret sauce. The fact that this value can change dynamically with browser window width is what makes a dynamic line-height value possible.*
3. So the whole expression to the right of the multiplication sign gets distilled down to this: 13em / 14em, or **0.928571429em**. Think of this as how close we are to the "upper gate" of our lock. If it's near zero, we're close to the lower gate. If it's near one, we're close to the upper gate.
4. Moving to the left of the multiplication sign, we compute the difference between our maximum and minimum line heights. 1.5–1.3 = **0.2**. This gives us *the full range of our fluid line height*.
5. Now we multiply the full range of our fluid line height (step 4) by how far along we are toward the upper gate of our lock (step 3):
   0.2 * 0.928571429em = **0.185714286em**.

And here's what a CSS lock looks like in code:

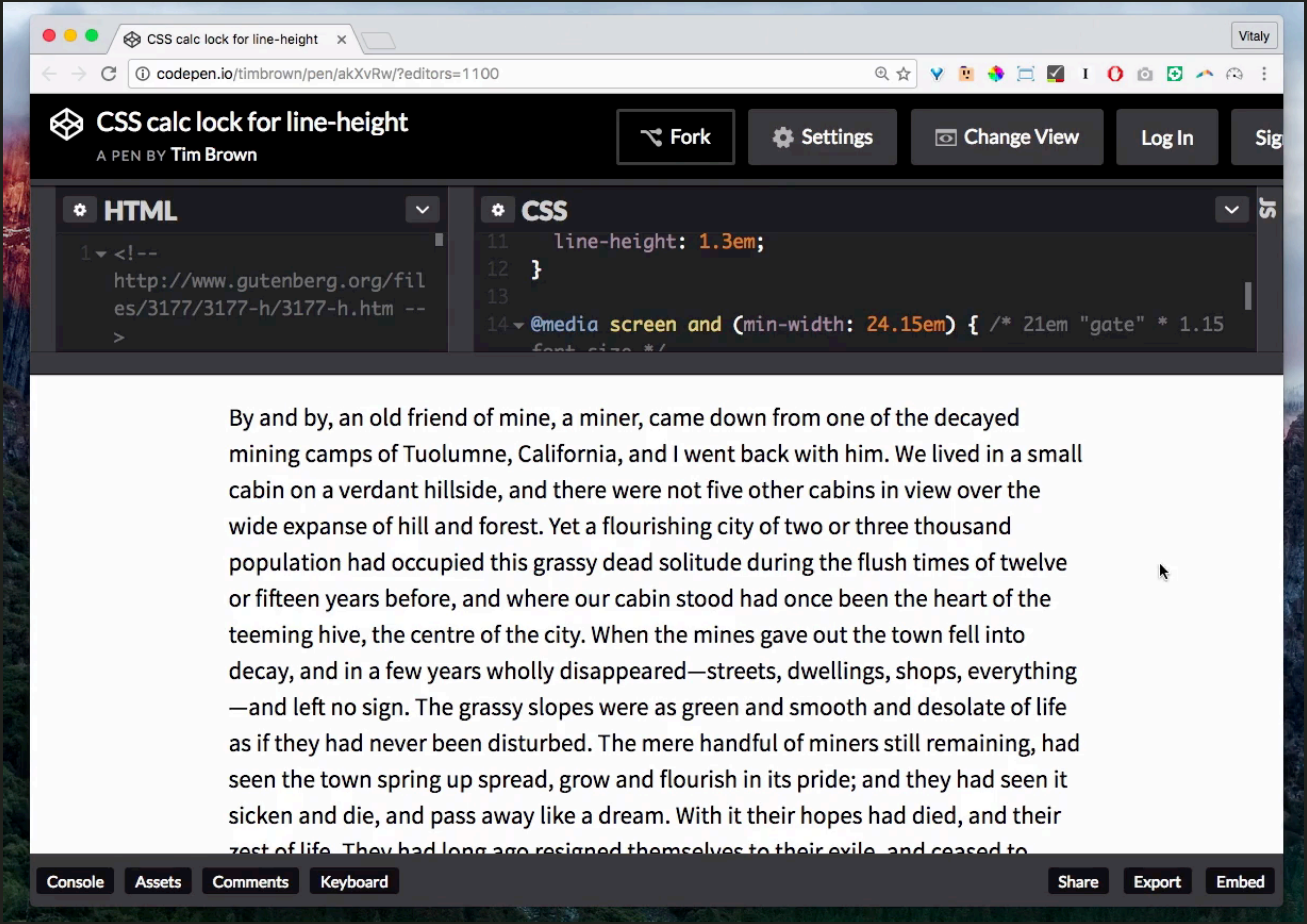
```
line-height: calc(1.3em + (1.5 - 1.3) * ((100vw - 21em)/(35 - 21)));
```

To understand how the formula works within **calc()**, we're going to work through it backwards.

1. See the very last part? **35–21**. That gives us the *full range of our paragraph's width*. It resolves to 14, because **14em** is the difference between our paragraph's width at its most narrow and most wide.

2. To the left of that, we've got **100vw–21em**. Because of the way CSS calc works, this resolves to an em-based value—and gives us a numerator to place above the 14em we already figured out. So, for example, let's say the viewport width (100vw) is equivalent to 34em. 34em–21em = 13em. *Note that the viewport unit in this step is our secret sauce. The fact that this value can change dynamically with browser window width is what makes a dynamic line-height value possible.*

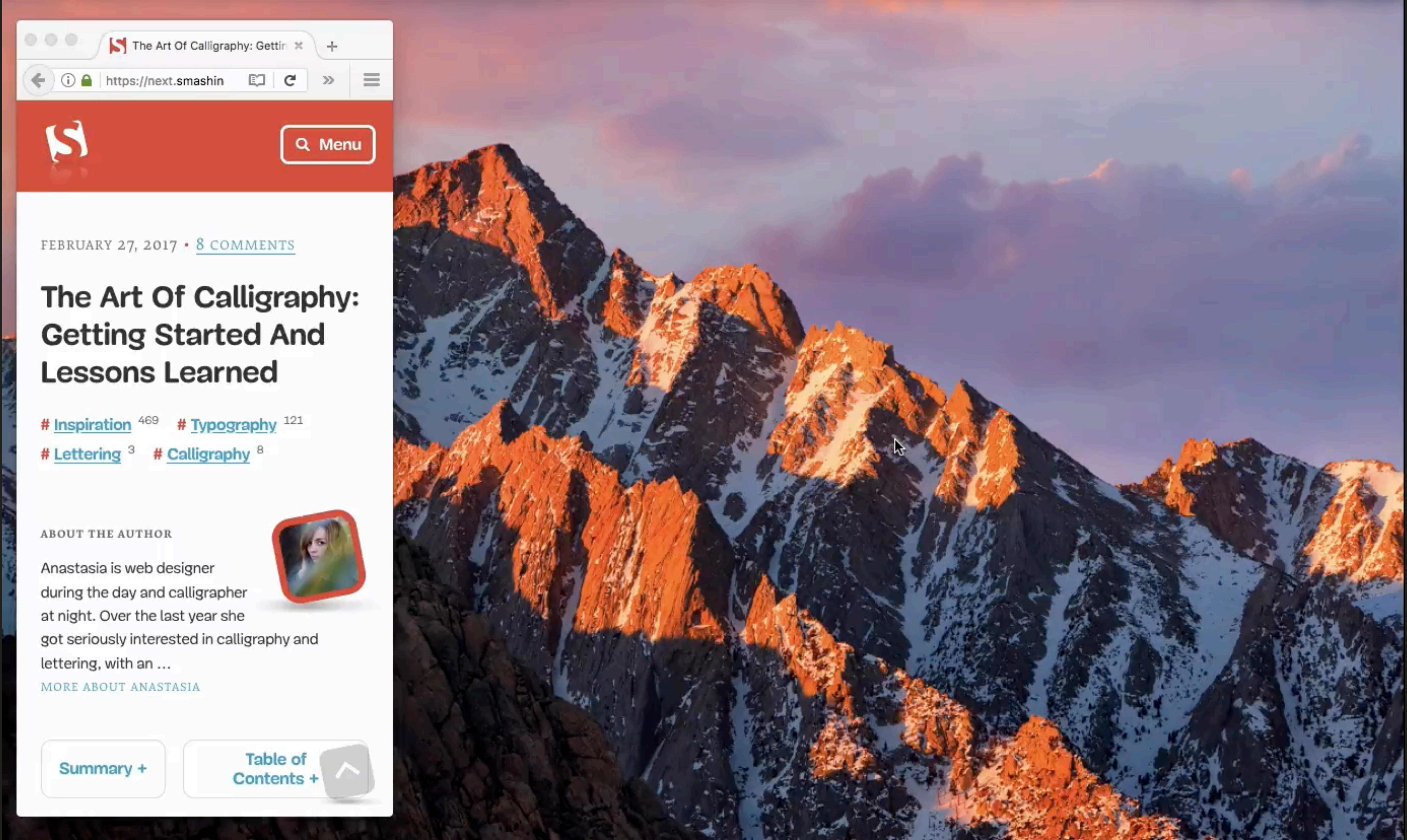
3. So the whole expression to the right of the multiplication sign gets distilled down to this: 13em / 14em, or **0.928571429em**. Think of this as how close we are to the "upper gate" of our lock. If it's near zero, we're close to the lower gate. If it's near one, we're close to the



```html
<!--
  http://www.gutenberg.org/fil
  es/3177/3177-h/3177-h.htm --
  >
```

```css
  line-height: 1.3em;
}

@media screen and (min-width: 24.15em) { /* 21em "gate" * 1.15
font-size */
```

By and by, an old friend of mine, a miner, came down from one of the decayed mining camps of Tuolumne, California, and I went back with him. We lived in a small cabin on a verdant hillside, and there were not five other cabins in view over the wide expanse of hill and forest. Yet a flourishing city of two or three thousand population had occupied this grassy dead solitude during the flush times of twelve or fifteen years before, and where our cabin stood had once been the heart of the teeming hive, the centre of the city. When the mines gave out the town fell into decay, and in a few years wholly disappeared—streets, dwellings, shops, everything —and left no sign. The grassy slopes were as green and smooth and desolate of life as if they had never been disturbed. The mere handful of miners still remaining, had seen the town spring up spread, grow and flourish in its pride; and they had seen it sicken and die, and pass away like a dream. With it their hopes had died, and their zest of life. They had long ago resigned themselves to their exile, and ceased to



FEBRUARY 27, 2017 • 8 COMMENTS

The Art Of Calligraphy: Getting Started And Lessons Learned

Inspiration 469 # Typography 121
Lettering 3 # Calligraphy 8

ABOUT THE AUTHOR

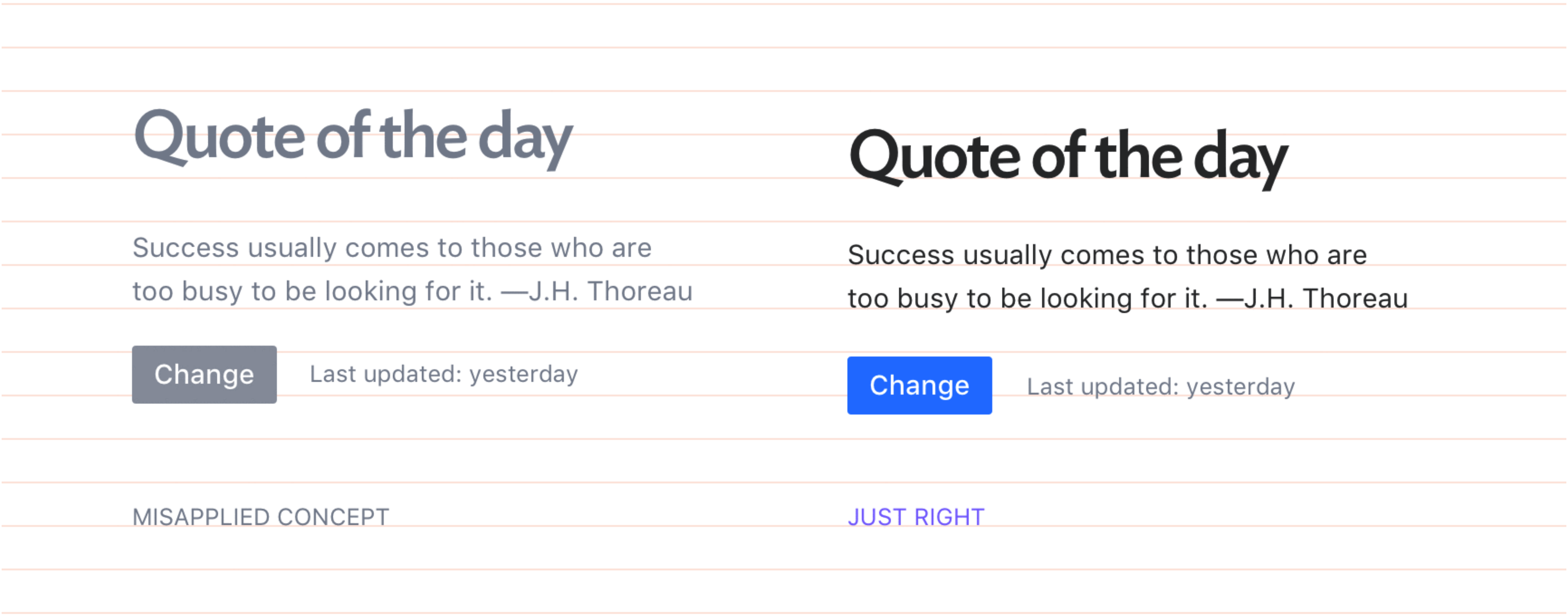
Anastasia is web designer during the day and calligrapher at night. Over the last year she got seriously interested in calligraphy and lettering, with an …

MORE ABOUT ANASTASIA

> “With *CSS Custom Properties*, we now can separate logic from design, effectively separating variable declarations from property declarations. Because logic lives above design then, I like to call this separation *logic fold*.
>
> — *Mike Riethmueller*
>
> *https://vimeo.com/235428198*

Live variables.

> “All *variable expressions* and calc statements that use CSS custom properties will be recalculated when the variable is redefined. Unlike preprocessors, they have knowledge of the DOM and can be *scoped* to DOM elements.
>
> — *Jonathan Harrell*
>
> *https://jonathan-harrell.com/unlocking-benefits-css-variables/*



Quote of the day

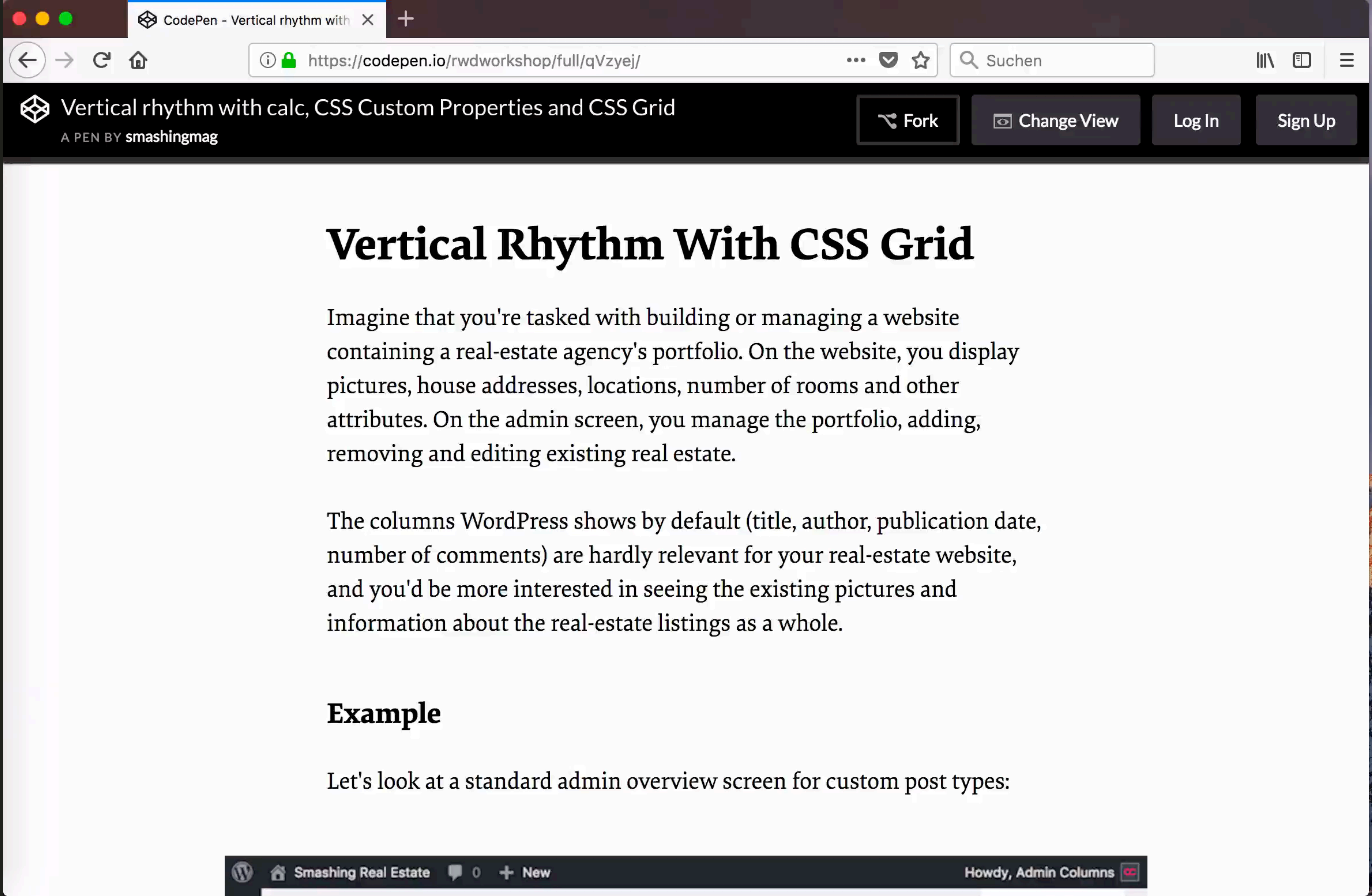
Success usually comes to those who are
too busy to be looking for it. —J.H. Thoreau

MISAPPLIED CONCEPT

Quote of the day

Success usually comes to those who are
too busy to be looking for it. —J.H. Thoreau

JUST RIGHT



CodePen - Vertical rhythm with
https://codepen.io/rwdworkshop/full/qVzyej/
Suchen
Vertical rhythm with calc, CSS Custom Properties and CSS Grid
A PEN BY smashingmag
Fork
Change View
Log In
Sign Up
Smashing Real Estate
0
New
Howdy, Admin Columns

Vertical Rhythm With CSS Grid

Imagine that you're tasked with building or managing a website containing a real-estate agency's portfolio. On the website, you display pictures, house addresses, locations, number of rooms and other attributes. On the admin screen, you manage the portfolio, adding, removing and editing existing real estate.

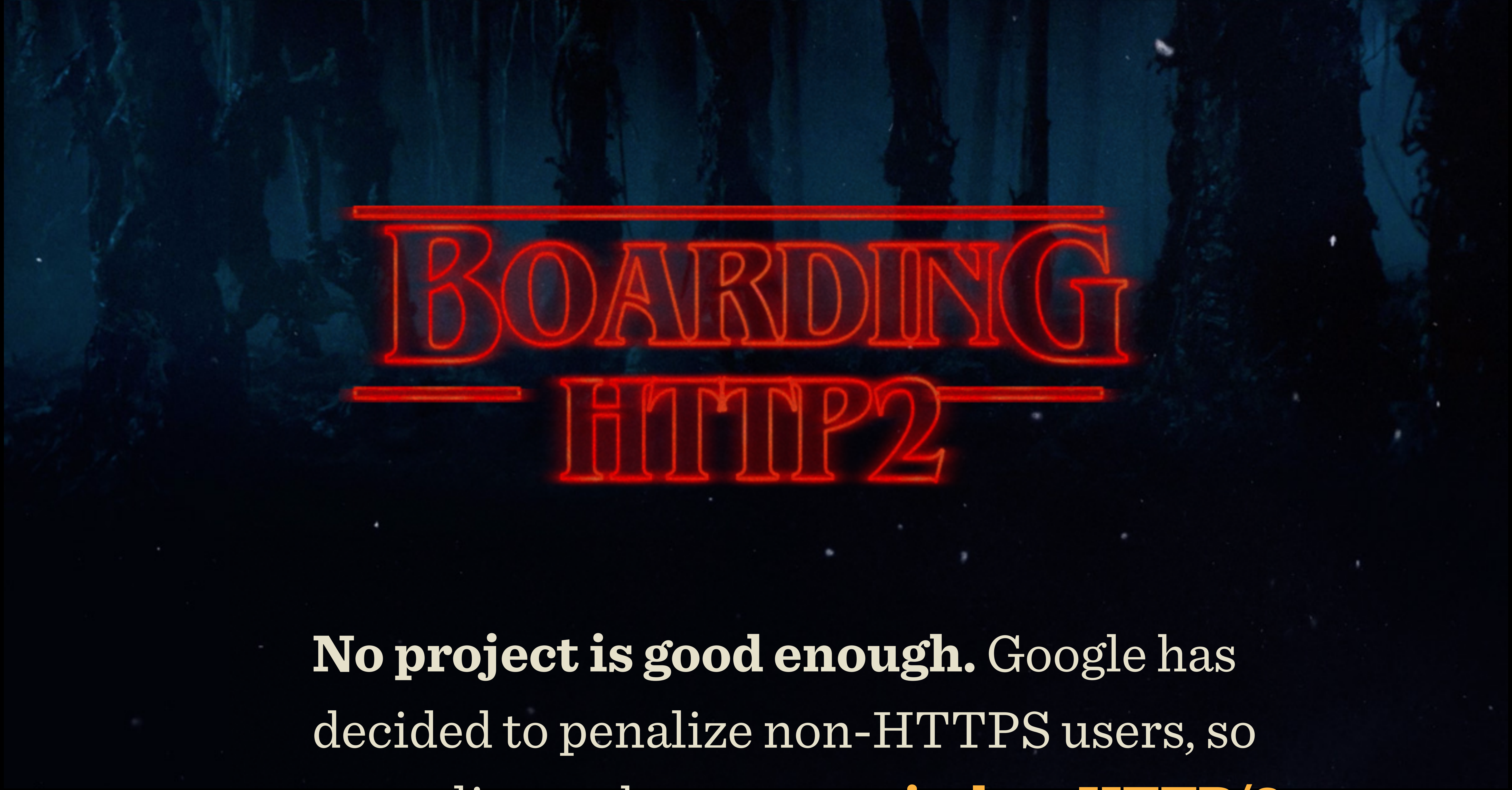
The columns WordPress shows by default (title, author, publication date, number of comments) are hardly relevant for your real-estate website, and you'd be more interested in seeing the existing pictures and information about the real-estate listings as a whole.

Example

Let's look at a standard admin overview screen for custom post types:

Level 4

HTTP2

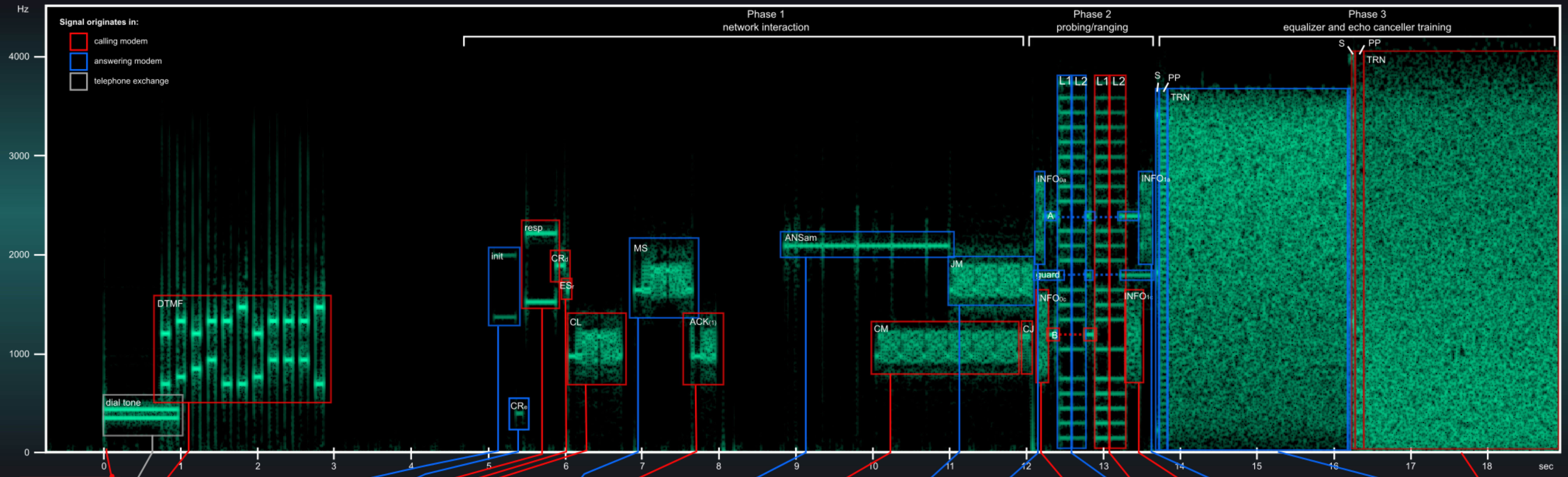


No project is good enough. Google has decided to penalize non-HTTPS users, so your client asks you to **switch to HTTP/2** to boost performance. What does it mean?

7000
6000
5000
4500
4000
3500
3000
2500
2000
1500
1200
1000
700
500
400
300
200
100
Hz
0.1 s
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

The Sound of the Dialup: an Example Handshake

© Oona Räisänen, windyoona@gmail.com



modem goes off hook

telephone exchange sends a dial tone

modem dials 1-(570)234-0003, a number in Pennsylvania

answering modem initiates a V.8 *bis* transaction

answering modem asks caller to list its capabilities

caller responds to V.8 *bis* initiation, agrees to list its capabilities

caller requests to escape from telephony into information transfer mode

FSK data @ 300 bps:

```
01111110 01111110 01111110
01001000 10010011 00000001
00000001 00000001 10010000
10101101 01000000 00000000
00101001 10000001 11000001
11000010 11100010 00100011
11100100 00110111 01111110
01111110
```

"I'm capable of full V.8.
I can transmit ACK.
By the way,
my country is the U.S.
I was manufactured by
Net2phone Inc."

FSK data @ 300 bps:

```
01111110 00111111 01111110
01111110 10001000 10000001
00000001 00000001 00000001
10000001 00101101 01000000
00000000 00101001 10000001
11000001 00000000 11100000
10000000 01111110 11011100
01111110 01111110 01111110
```

"Why don't we use
V.8 then."

FSK data @ 300 bps:

```
01111110 01111110
00101000 10111011
01100101 01111110
01111110
```

"Okay, mode
acknowledged.
Terminating
V.8 bis transaction."

answering modem disables echo suppressors and cancellers in PSTN

FSK data @ 300 bps:

```
1111111111 0000001111 0100000111
0101001101 0110010001 0001010011
0010101001 0101100001 0111001001
```

(repeated 6 times)

"Modulation modes available:
-PCM V.90/V.92 analogue
-V.34 duplex
-V.32/V.32 bis
-V.23 duplex
-V.22/V.22 bis
-V.21

V.42 LAPM is available.

PSTN connection is over a
regular landline (not cellular).
My network is analogue."

FSK data @ 300 bps:

```
1111111111 0000001111 0100000111
0101001101 0110010001 0000010011
0111000101 0101100001 0010101001
```

(repeated 3 times)

"Oh, I can do any of those
modes as well. Except my
V.90/V.92 is digital.

V.42 LAPM is available.

PSTN connection is over a
regular landline (not cellular).
My network is digital."

DPSK data @ 600 bps:

```
11110111001011111111111000
01000100011111110110011111
```

"I can transmit at either carrier frequency at whichever standard symbol rate except 3429. Tx and Rx symbol rates must be equal.

I can reduce power if needed.
I can support up to 1664-point signal constellations.
My clock source is internal."

DPSK data @ 600 bps:

```
11110111001011111111111000
10100010111011000111101 1
```

"I can transmit at either carrier frequency at whichever standard symbol rate except 3429. Tx and Rx symbol rates must be equal.

I can reduce power if needed.
I can support up to 1664-point signal constellations.
My clock source is external."

modems send a wide-spectrum probing signal in both directions to do measurements of the telephone line

DPSK data @ 600 bps:

```
0111101111001000011000000000110001
1000010111100001011100001011100001
0000101010100100000000001100101010
0010000111
```

"Please don't reduce your
power by more than 6 dB.

I'm not equipped to accurately
measure the frequency offset
of the tones you sent.

Using different symbol rates,
we could achieve the following:

symrate	ur pre-emph α	bps
2400	6 dB	14400
2743	8 dB	16800
2800	8 dB	16800
3000	8 dB	16800
3200	8 dB	19200
3429	10 dB	21600

I can use the higher frequency
carrier at any symbol rate."

DPSK data @ 600 bps:

```
111101110010000000000000001111001000
010110000000000010011011000110111111
```

"Please don't reduce power
any further.

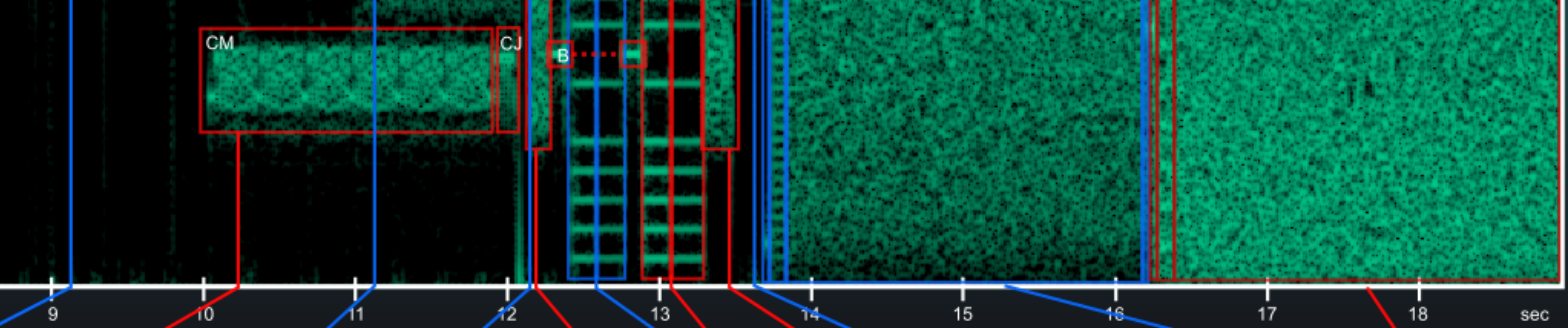
Your tones were offset by 0 Hz.

Let's use a symbol rate of 3200.
We can achieve a maximum of
4800 bps. For your pre-emphasis
filter, select the parameters
β = 1.0 dB and γ = 2.0 dB.

Please use carrier at 1920 Hz
for transmission, I'll use 1829."

Modems go to scrambled data and learn how the other modem sounds over the channel. The final bitrate and signal constellation are also decided on.

A final training phase called Phase 4 will follow, after which the modem speaker goes mute and data can be put through the connection.



FSK data @ 300 bps:

```
1111111111 0000001111 0100000111
0101001101 0110010001 0000010011
0111000101 0101100001 0010101001
```

(repeated 3 times)

"Oh, I can do any of those modes as well. Except my V.90/V.92 is digital.

V.42 LAPM is available.

PSTN connection is over a regular landline (not cellular). My network is digital."

DPSK data @ 600 bps:

```
111101110010111111111000
0100010001111110110011111
```

"I can transmit at either carrier frequency at whichever standard symbol rate except 3429. Tx and Rx symbol rates must be equal.

I can reduce power if needed.
I can support up to 1664-point signal constellations.
My clock source is internal."

DPSK data @ 600 bps:

```
111101110010111111111000
101000101110110001111011
```

"I can transmit at either carrier frequency at whichever standard symbol rate except 3429. Tx and Rx symbol rates must be equal.

I can reduce power if needed.
I can support up to 1664-point

modems send a wide-spectrum probing signal in both directions to do measurements of the telephone line

DPSK data @ 600 bps:

```
011110111001000011000000000110001
100001011100001011100001011100001
000010101010010000000001100101010
0010000111
```

"Please don't reduce your power by more than 6 dB.

I'm not equipped to accurately measure the frequency offset of the tones you sent.

Using different symbol rates, we could achieve the following:

symrate	ur pre-emph α	bps
2400	6 dB	14400
2743	8 dB	16800
2800	8 dB	16800

DPSK data @ 600 bps:

```
11110111001000000000000001111001000
01011000000000000100110110001101111
```

"Please don't reduce power any further.

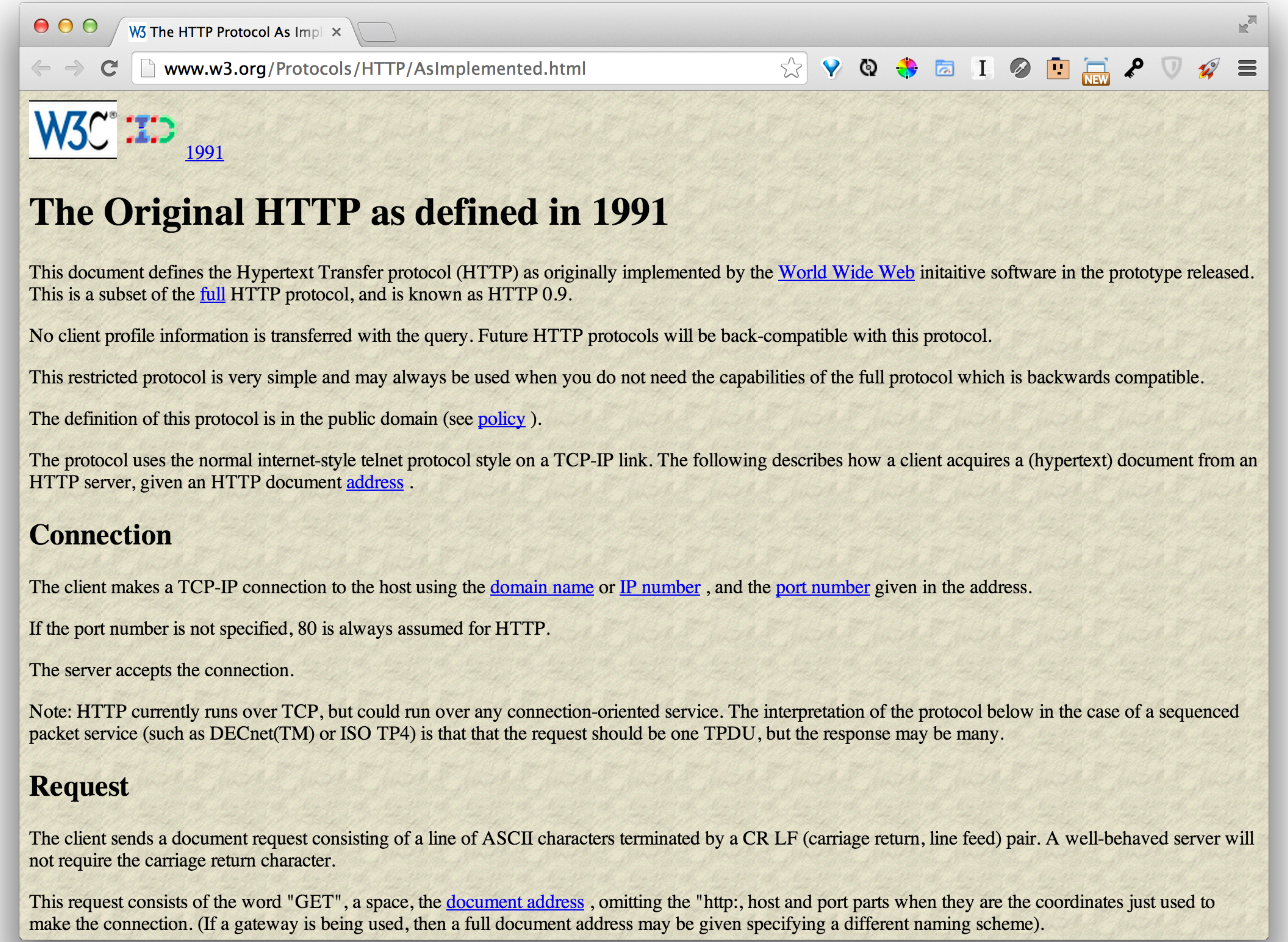
Your tones were offset by 0 Hz.

Let's use a symbol rate of 3200. We can achieve a maximum of 4800 bps. For your pre-emphasis filter, select the parameters β = 1.0 dB and γ = 2.0 dB.

Please use carrier at 1920 Hz for transmission, I'll use 1829."

Modems go to scrambled data and learn how the other modem sounds over the channel. The final bitrate and signal constellation are also decided on.

A final training phase called Phase 4 will follow, after which the modem speaker goes mute and data can be put through the connection.



W3 The HTTP Protocol As Impl ×
www.w3.org/Protocols/HTTP/AsImplemented.html

1991

The Original HTTP as defined in 1991

This document defines the Hypertext Transfer protocol (HTTP) as originally implemented by the World Wide Web initaitive software in the prototype released. This is a subset of the full HTTP protocol, and is known as HTTP 0.9.

No client profile information is transferred with the query. Future HTTP protocols will be back-compatible with this protocol.

This restricted protocol is very simple and may always be used when you do not need the capabilities of the full protocol which is backwards compatible.

The definition of this protocol is in the public domain (see policy).

The protocol uses the normal internet-style telnet protocol style on a TCP-IP link. The following describes how a client acquires a (hypertext) document from an HTTP server, given an HTTP document address .

Connection

The client makes a TCP-IP connection to the host using the domain name or IP number , and the port number given in the address.

If the port number is not specified, 80 is always assumed for HTTP.

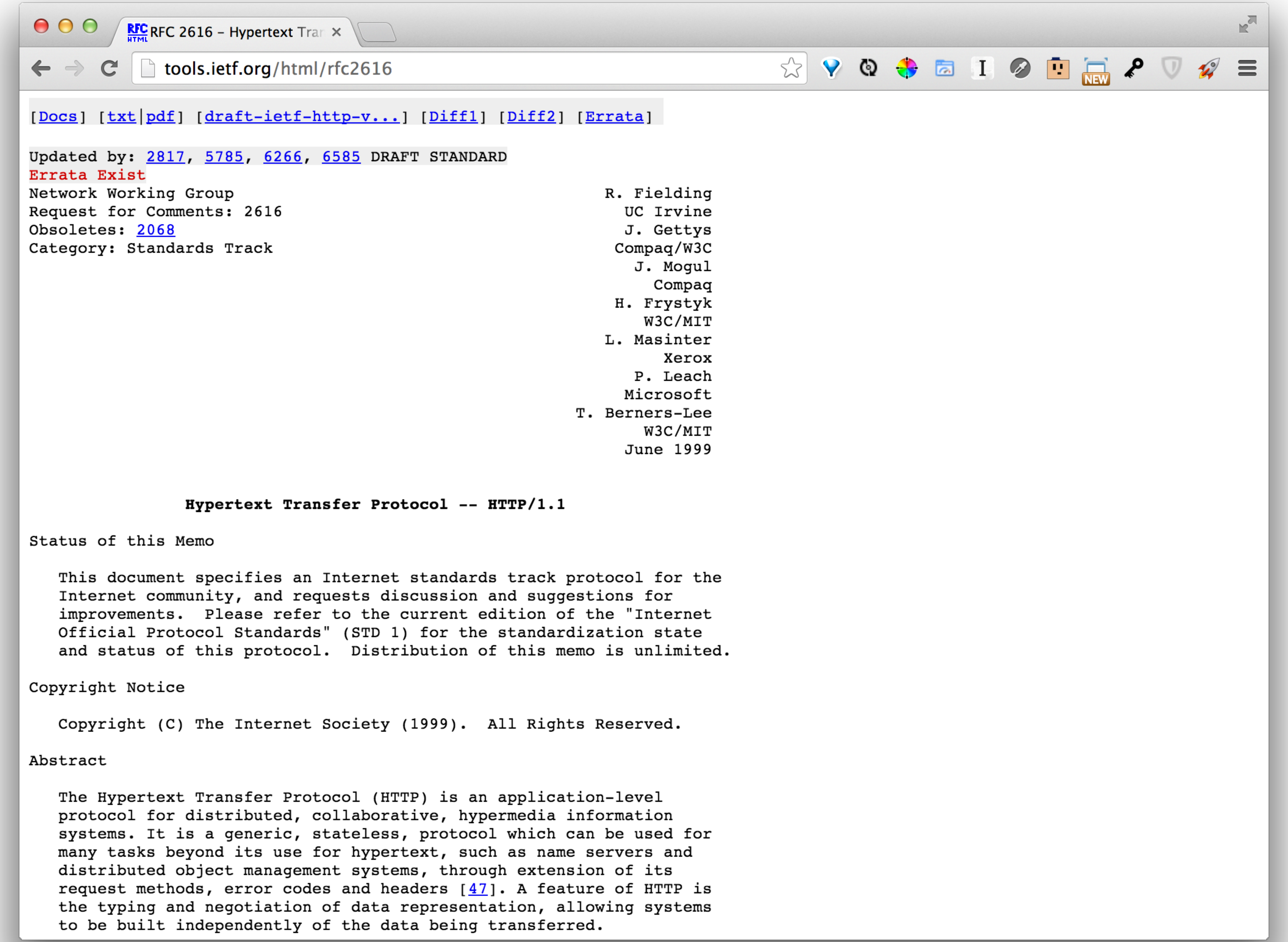
The server accepts the connection.

Note: HTTP currently runs over TCP, but could run over any connection-oriented service. The interpretation of the protocol below in the case of a sequenced packet service (such as DECnet(TM) or ISO TP4) is that that the request should be one TPDU, but the response may be many.

Request

The client sends a document request consisting of a line of ASCII characters terminated by a CR LF (carriage return, line feed) pair. A well-behaved server will not require the carriage return character.

This request consists of the word "GET", a space, the document address , omitting the "http:, host and port parts when they are the coordinates just used to make the connection. (If a gateway is being used, then a full document address may be given specifying a different naming scheme).



[Docs] [txt|pdf] [draft-ietf-http-v...] [Diff1] [Diff2] [Errata]

Updated by: 2817, 5785, 6266, 6585 DRAFT STANDARD
Errata Exist

```
Network Working Group                                      R. Fielding
Request for Comments: 2616                                   UC Irvine
Obsoletes: 2068                                              J. Gettys
Category: Standards Track                                   Compaq/W3C
                                                              J. Mogul
                                                                Compaq
                                                            H. Frystyk
                                                               W3C/MIT
                                                           L. Masinter
                                                                 Xerox
                                                              P. Leach
                                                             Microsoft
                                                        T. Berners-Lee
                                                               W3C/MIT
                                                             June 1999
```

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [47]. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP/1.1

- *HTTP/1.1* was designed for connections and bandwidth that are significantly different today.

  — *Single request* per connection,

  — *Max. 10 connections per domain,*

  — Exclusively *client-initiated requests,*

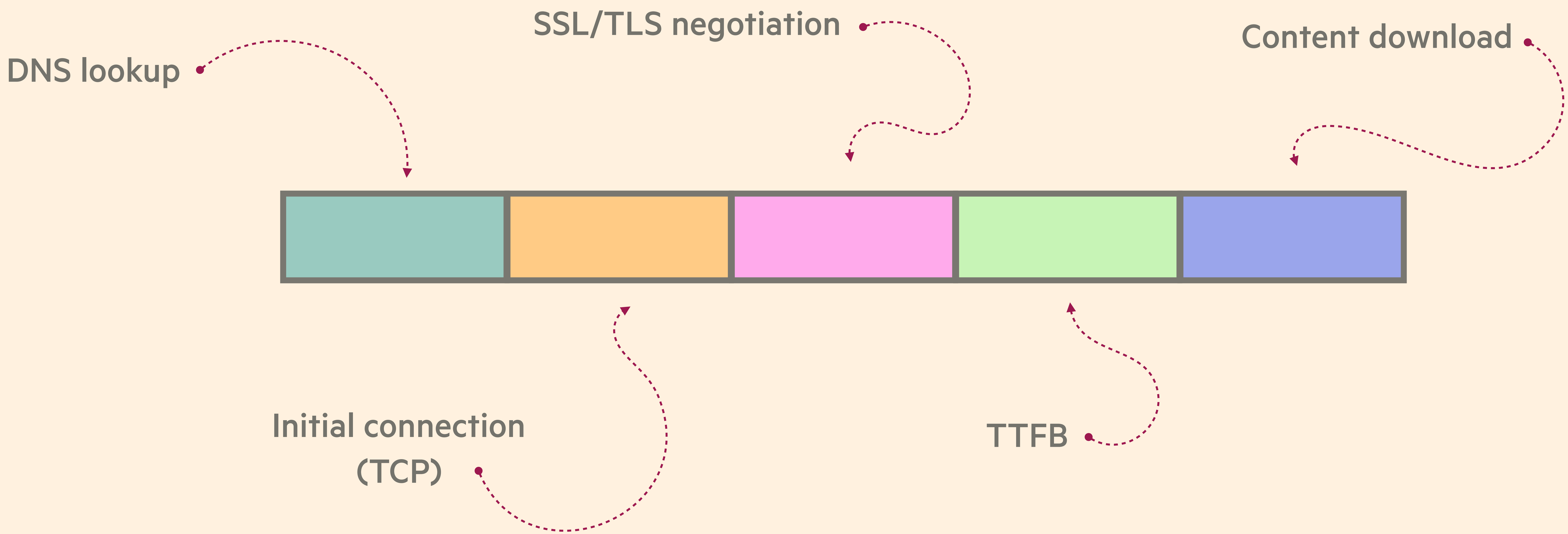
  — *Uncompressed* request and response headers,

  — *Redundant* headers,

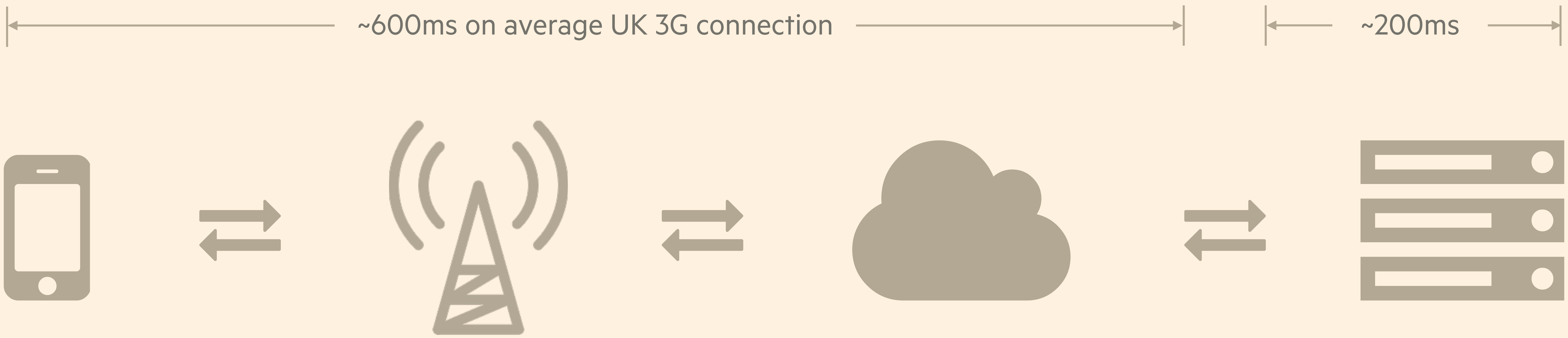
  — *Optional* data compression,

  — HTTP is slow, but *HTTPS is even slower.*

HTTP connection



Average round trip time on UK 3G connection



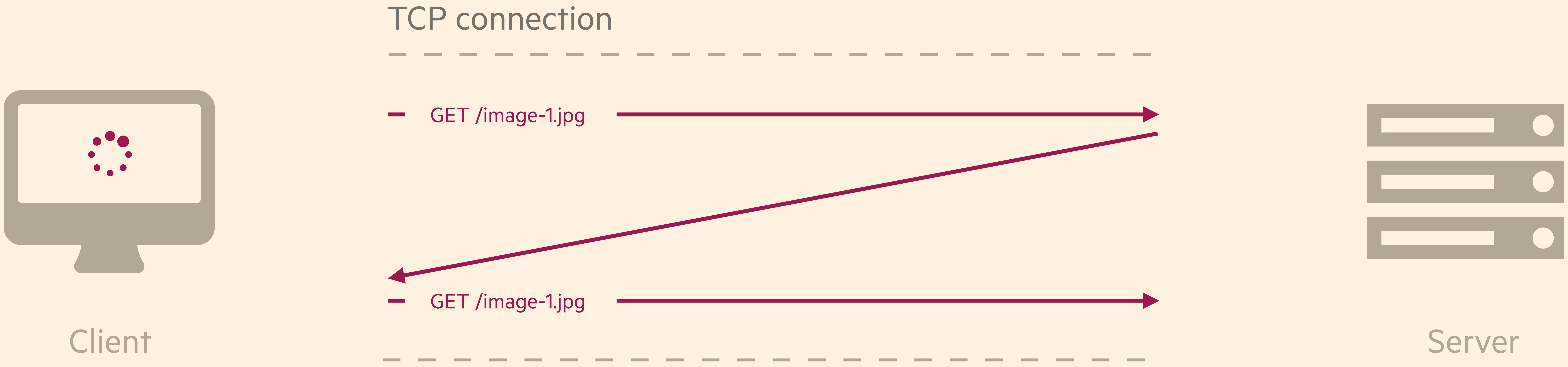
Control plane latency

Internet routing latency
CDNs, ISPs, Caches, Proxies

Internal latency
Firewalls, Load Balancers, Servers

Dmytrii Shchadei — http://www.slideshare.net/metrofun/reduce-mobile-latency

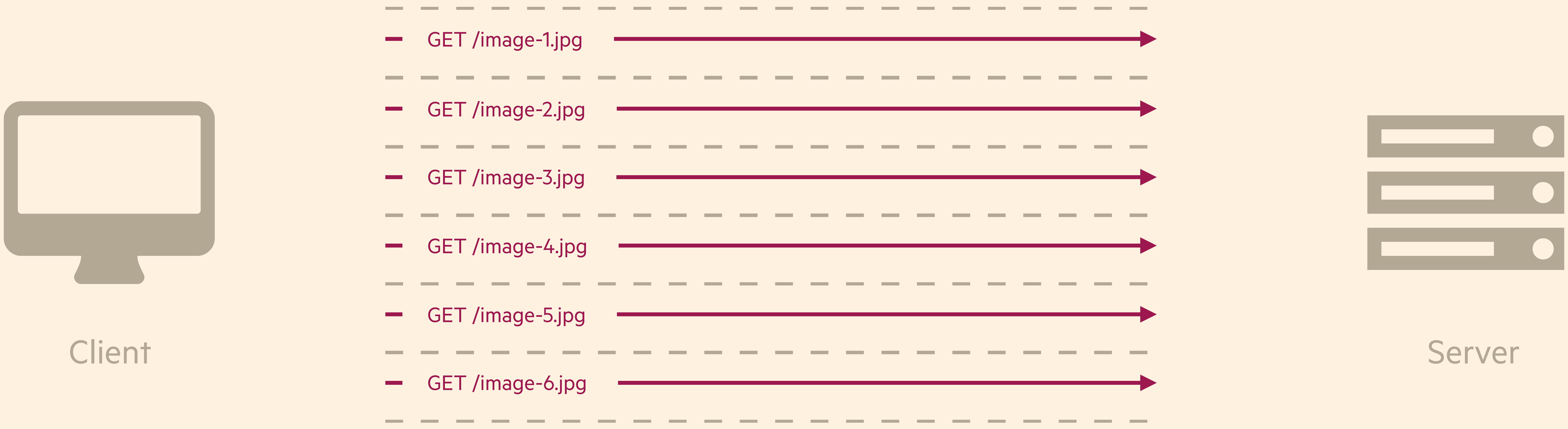
Head of line blocking



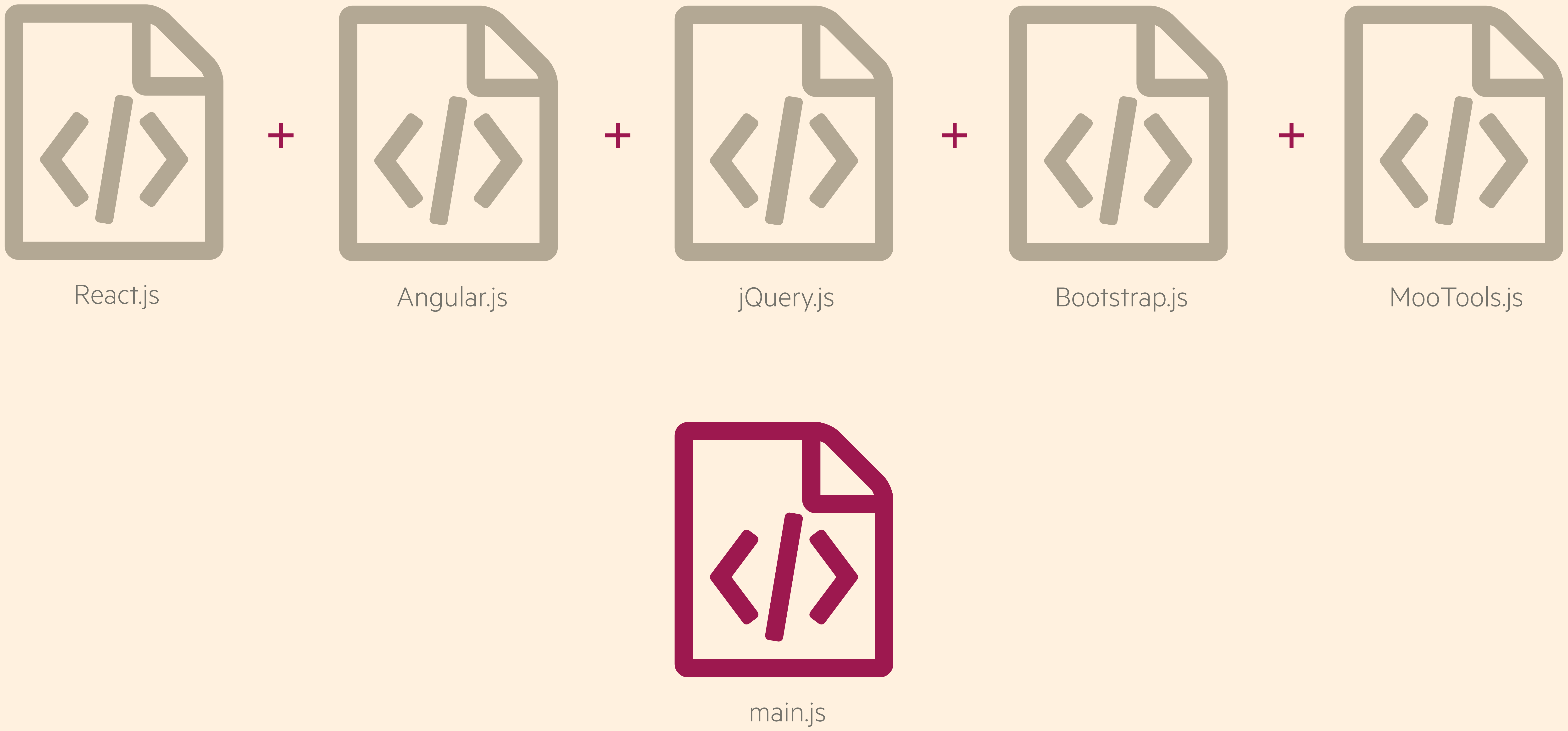
TCP connection

- GET /image-1.jpg
- GET /image-1.jpg

Head of line blocking

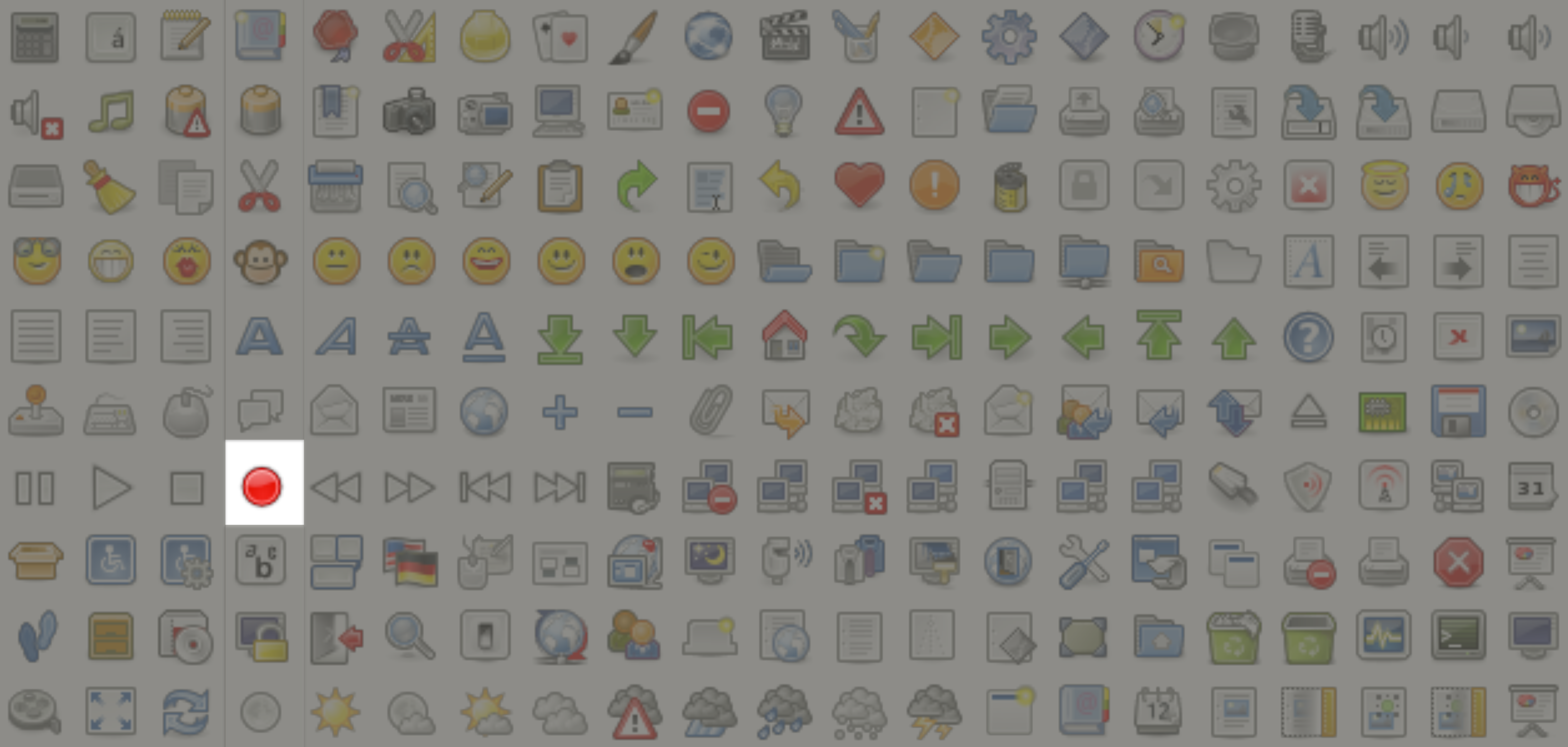


Hacks: concatenation



React.js + Angular.js + jQuery.js + Bootstrap.js + MooTools.js

main.js



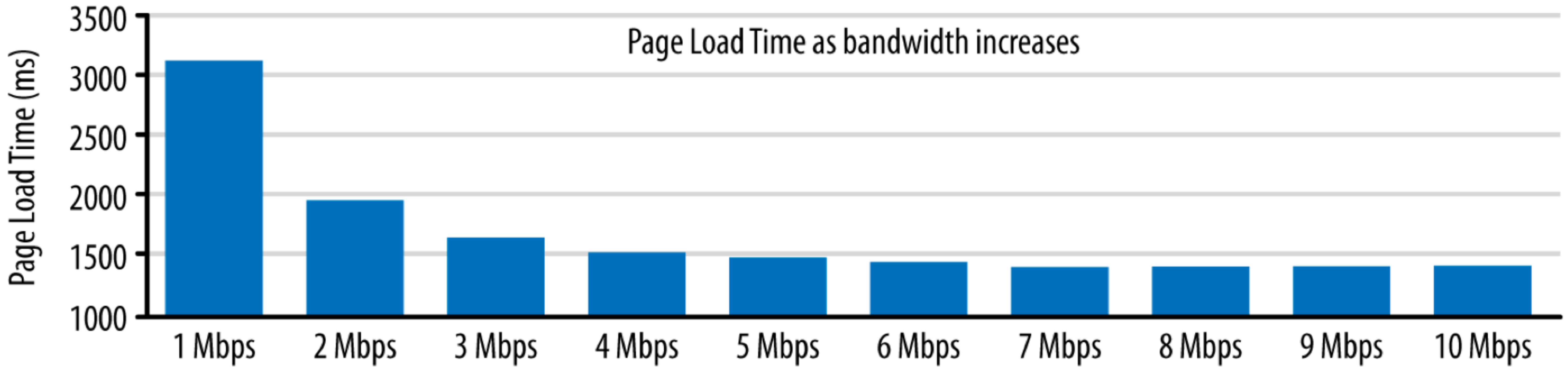
Hacks: domain sharding

```
https://www.flickr.com/search/?text=hacks
 1. www.flickr.com - search/                 2439 ms
 2. s.yimg.com - combo                        489 ms
 3. s.yimg.com - combo                        313 ms
 4. s.yimg.com - combo                        313 ms
 5. s.yimg.com - pxl.gif                      313 ms
 6. c4.staticflick...49_1aab6b9e51.jpg        662 ms
 7. c4.staticflick..._b4a2587134_t.jpg        872 ms
 8. c1.staticflick..._98b4ecadc2_m.jpg       1061 ms
 9. c1.staticflick...92_91fc5c5348.jpg       1010 ms
10. c1.staticflick...57_bd6a02232c.jpg       1022 ms
11. c1.staticflick...11_70d324abd4.jpg        407 ms
12. c3.staticflick..._f240d58e68_t.jpg        674 ms
13. c3.staticflick..._ff4c9185da_m.jpg        814 ms
14. c3.staticflick..._2846e95dc8_m.jpg        768 ms
15. c4.staticflick...43_5472d81483.jpg       1189 ms
16. c4.staticflick...88_df1230a345.jpg       1449 ms
17. c3.staticflick..._c3751a954b_m.jpg       1419 ms
18. c4.staticflick..._288b40caba_n.jpg       1447 ms
19. c4.staticflick..._3d98d06b6c_n.jpg       1542 ms
20. c4.staticflick..._7ee223f39c_n.jpg       1605 ms
21. s.yimg.com - combo                        463 ms
22. s.yimg.com - loader-hermes-min.js         470 ms
23. s.yimg.com - general-d6bcdcb0.png         461 ms
24. s.yimg.com - icons-36c2f8d6.png          1330 ms
25. c3.staticflick...4ae1d356d9_m.jpg        1499 ms
```

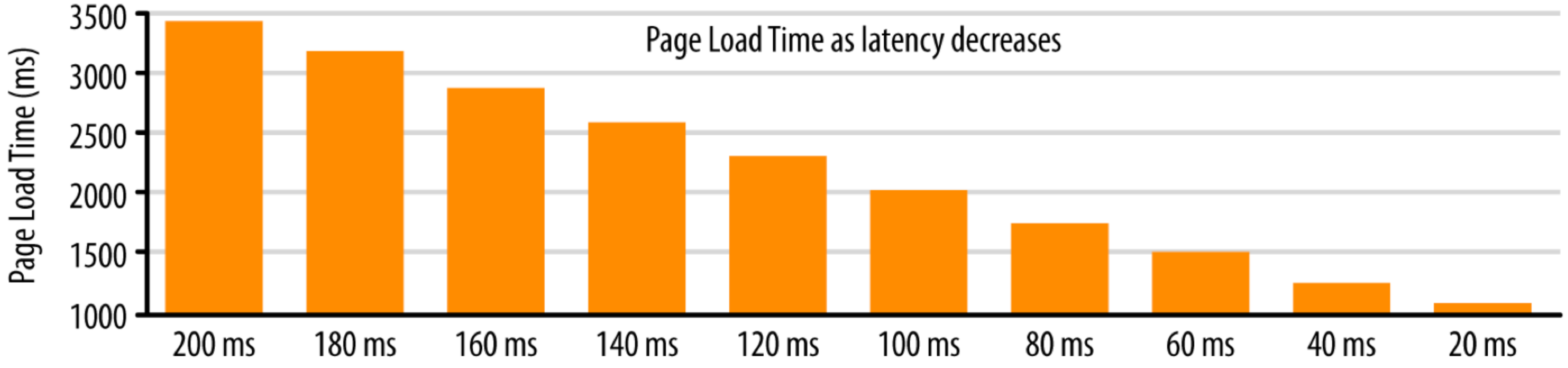
Hacks: inlining

```
{
    "css" : "@font-face{font-family:EgyptianText;src:url(data:application/x-font-woff;base64,
            d09GRgABAAAAAEopAAwAAAAAkbwAAAAAAABI+AAAATEAAAJiAAAAAAAAAABHUE9TAAA6JAAADjkAACw2TlE43Ed
            TVUIAAEhgAAAAlwAAAOJbQlwHT1MvMgAAAZQAAABYAAAAYG3Bc1FjbWFwAAAEGAAAArEAAAPwbLUopmdseWYAAAiQAA
            AspAAATqhnZM4UaGVhZAAAARwAAAA2AAAANvyN/WdoaGVhAAABVAAAACAAAAAkBzsEUmhtdHgAAAHsAAACLAAAA3Tkk
            SIRbG9jYQAABswAAAHCAAABwrHgn2ptYXhwAAABdAAAAB0AAAAgAO8AwG5hbWUAADU0AAADbAAACR36aaKkcG9zdAAA
            OKAAAAGCAAACBV5fDrgAAQAAAAEAg1gtvrBfDzz1AAsD6AAAAADMT0ckAAAAAMz2cdj/Pv8OBOsD9AAAAAkAAgAAAAA
            AAHjaY2BkYGDm/6/OwMAq8t/uvw3LawagCAq4BQByNwWVeNpjYGRgYHjAkMLAzhDFwMIA4iEAMwMjACoNAbAAAAB42mN
            gZrJgnMDAysDA1MUUwcDA0AOkuxgYGK8xGDE8BIqysTKzsjKzcTAD5dgZkICzv68vgwODwm8mZv7/6gwMzF8YPikwMEw
            GyTGJMV0EUgoMzABSPAzoeNptkk9IVFEUxr9zrkVpCaODRhNTk8g00zSXeAMNFNHTZEhIk4IkZYxpV0QLoUCypxARRJs
            QWrRt3R9qEyK4qEWLIPqzKKxdEAUVNJta9Pzu7SUW8fi9c7l/zv3Od480AX2DnI4gr3XkZAzbtAQrz7FDb2KDhmjVU1y
            fQJsOYJOuQ8HMImdG0Gu4Fz8wjo+AtKBf7qGmFVh94edrMoEK2SwvUZUqCjKNkHFYLiAlfThGyvKEeRZR1cdYr3O85wa
            KXk9EpsgzFE2KuhpkEd36lHPUaW4zxuQD959M4kNGater6NJ32KjzaDdfkfL5XC3UL+dxhPfnGa28R4/U41/SxJSmnQc
            cf6cPkyhplnsmsUWL6KCOjATokt54SQ/6cdo8QEYvEufbaeoaR0nmuEbfJEIn11J6CK1miP4NUo/zkd7JXRSdZ4xWFni
            nq9vh6o5W1eT0O03/wWvM/o3XF2CtBPFP8m1F2784XauQ19S7gIoGzBFhj84gq9dY9zTypgbbspM6m7hERs1b6qQ2+cQ
            83egx/b5v2vmGlu/fJ7fi+/IFu4g1Mxh2fbFSH8+ZV+ydR7xnCaFRhNrpKYllX1h67jziO+FzvF/KzFNmXqfP9Yxb516
            fK0p8i373ie+NBtKkzY29f39wPiZ4j5o4QQbJOdIg80l9d8hZcp2M6RV6MEtGcZhsJSHJkN3kDNlLDpDtZIBUyL4kBuS
            4uYzCmjQK7iy/o2SI//oyLHWhCnjahZP3T9ZQGEZPP0XRxI17lU/FreAWJ4gD994LBy7ArbgRERUXIiruPXFvcaPg3qI
            YBT//CWNiQn1aopH4g7fpvWl78+a+5zwFCpF/V8PQjMtPT4bzXNhVXGtnXV5am9KWjiznMKmc5zLXsYwAI9gYZIQZ0cZ
            yI8n1yJVj+phVzBqm2/Qz/c1AM8Q87et2l/3psixVMPEnkE4c4ChnVeEqaaoQZAw0xqvCMlXIdGWb5cxKZjXTdCq0/VPB
            cCrsAesbzrCynDkhL90KzTtqNcrbZDnn9yTkf/fM8pzwRHrCPNW/huc+9spQT531erTuCUxlntYFLKbg+M4PtNEobHjz
            7yhDWcrhw2M+Up4KVKQSlflEFaqKVnVqUJNccvjKDpL4wFOeUZpYEljDRpKJ5w4rieMk6WxiGyUpQSm2kCIap9nJeMJYr
            9OdZyJnOCdCF7jIJR1oEte4Il67mMxnNpAm+jcIpyjFWK1upjCNGUwngs1EMpMoZjGbucxRn/NxqdNoFrKIJer4C4ksY6
            lMrsCb4vJ5iwzuco/7POQ2DzjIIU6xX55SieEEqyhiPHX8ef7L4DkveMkrXvOGt7zjPVl/cTHxxU0talMHP+pSj/o0oCGN
            aEwTJcyfAJrRnBa0pBWtaaPMBdKO9nRQ9jrJYBDBdCGErnSjOz0IpSe96E0f+tKP/gxgIIMYzBCGMozhjGAko2R9DGMZp
            x5sxhPEd5JIhovbVJGbLnYR4hYlcja3OSJnc1sgZtGitljcloqZTSxGBm2zsaISJ/bxcruWddipWy8zv8ezAsnZKN/ITa
            JsJzlvtjpzslKwXWlJkf1d7Fa+97LPIW87+KIdhzmiv+UJxzhOJo9k46TcpJKtxJxRbs4pLfkpuewkBCXFzkYaN+X1thKX
```

Bandwidth *vs.* Latency impact on Page Load Time



Single digit % perf improvement after 5 Mbps



Linear improvement in page load time!

Delivering The Goods, Paul Irish, *https://www.youtube.com/watch?v=R8W_6xWphtw*

HTTP/2

- *HTTP/2.0* promises speed improvement, decreased network latency and better management of assets.

  — *64%* reduction in page load times *(23% on mobile)*,

  — *Unlimited number of parallel requests* per connection,

  — Quicker slow-start and better compression,

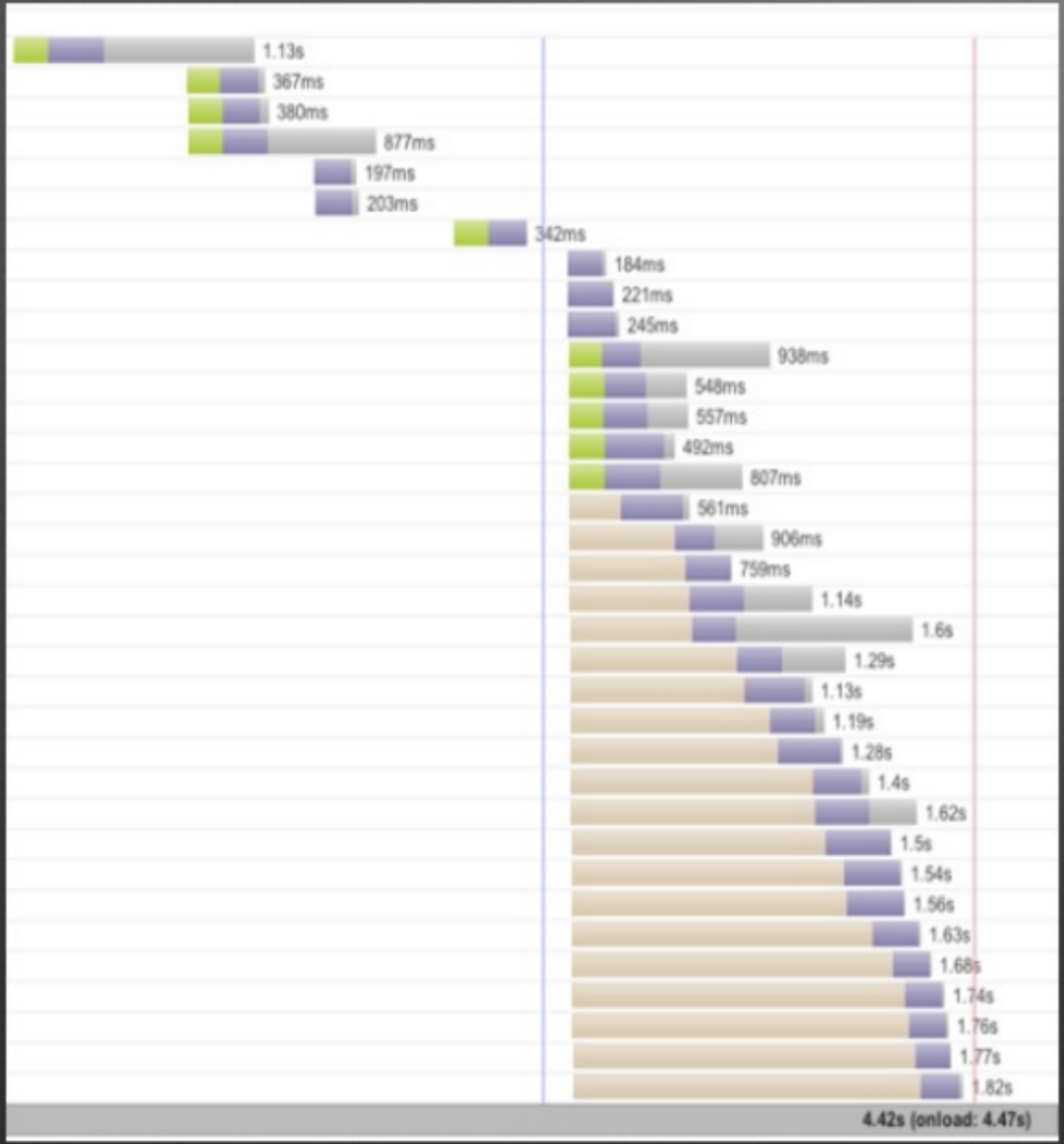
  — *One connection per host* handles all requests,

  — Developers can *prioritize* and *push* resources,

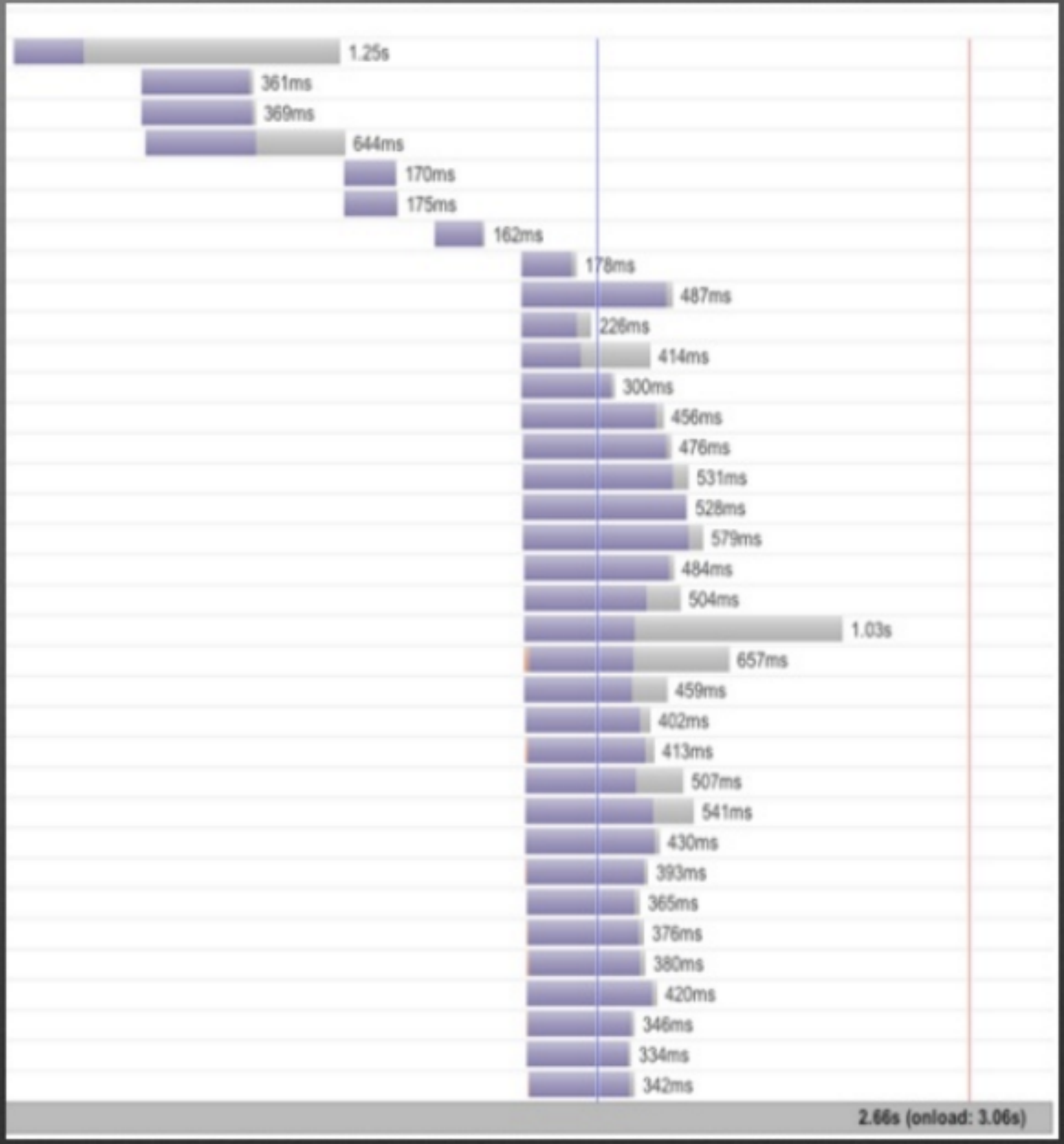
  — *Browsers* require the protocol to run over HTTPS,

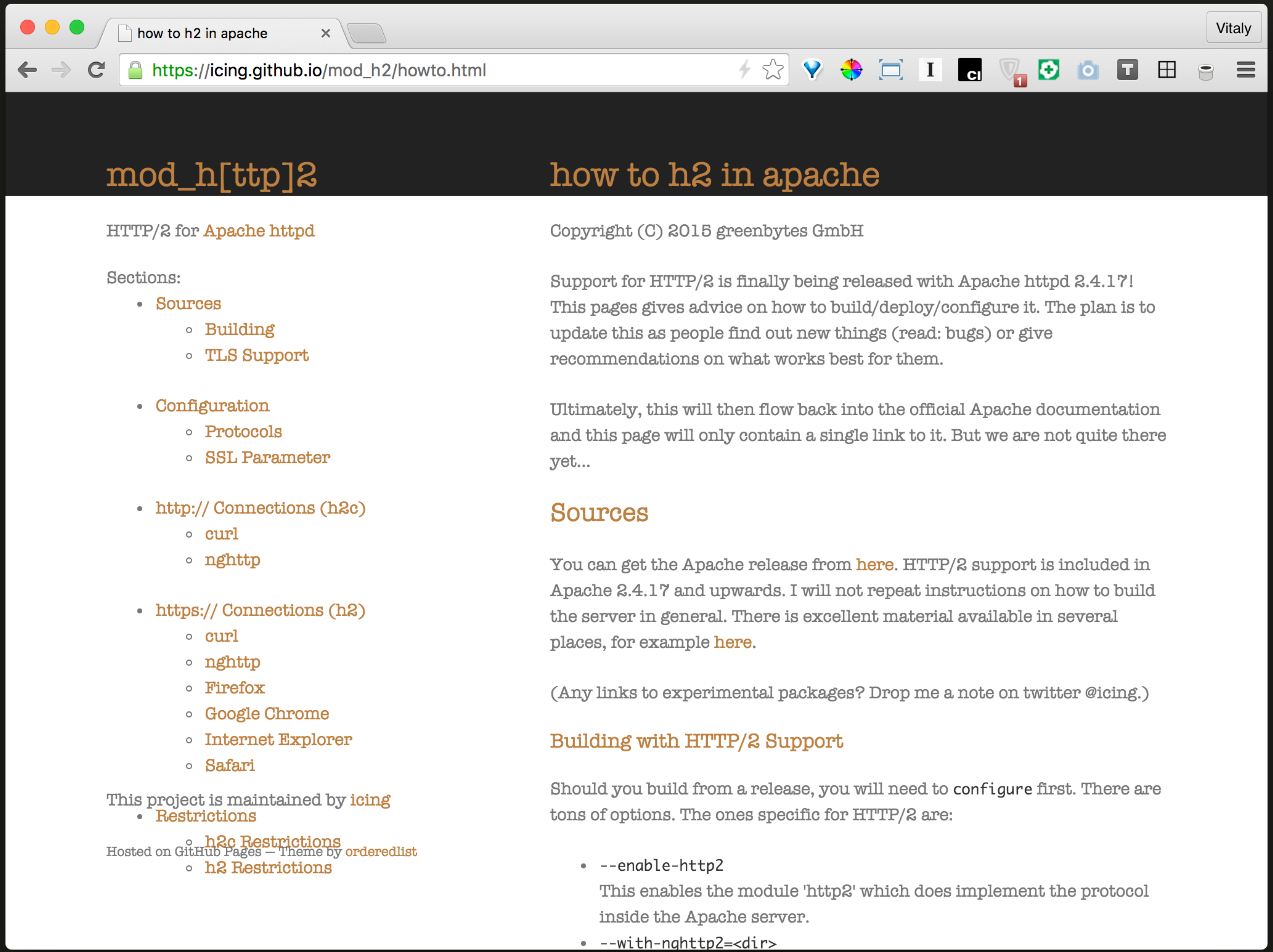
  — Extension of HTTP/1.1; as such, falls back to *HTTP/1.1*.

HTTP/1.1



HTTP/2





mod_h[ttp]2

HTTP/2 for Apache httpd

Sections:

- Sources
  - Building
  - TLS Support
- Configuration
  - Protocols
  - SSL Parameter
- http:// Connections (h2c)
  - curl
  - nghttp
- https:// Connections (h2)
  - curl
  - nghttp
  - Firefox
  - Google Chrome
  - Internet Explorer
  - Safari
- Restrictions
  - h2c Restrictions
  - h2 Restrictions

This project is maintained by icing

Hosted on GitHub Pages — Theme by orderedlist

how to h2 in apache

Copyright (C) 2015 greenbytes GmbH

Support for HTTP/2 is finally being released with Apache httpd 2.4.17! This pages gives advice on how to build/deploy/configure it. The plan is to update this as people find out new things (read: bugs) or give recommendations on what works best for them.

Ultimately, this will then flow back into the official Apache documentation and this page will only contain a single link to it. But we are not quite there yet...

Sources

You can get the Apache release from here. HTTP/2 support is included in Apache 2.4.17 and upwards. I will not repeat instructions on how to build the server in general. There is excellent material available in several places, for example here.

(Any links to experimental packages? Drop me a note on twitter @icing.)

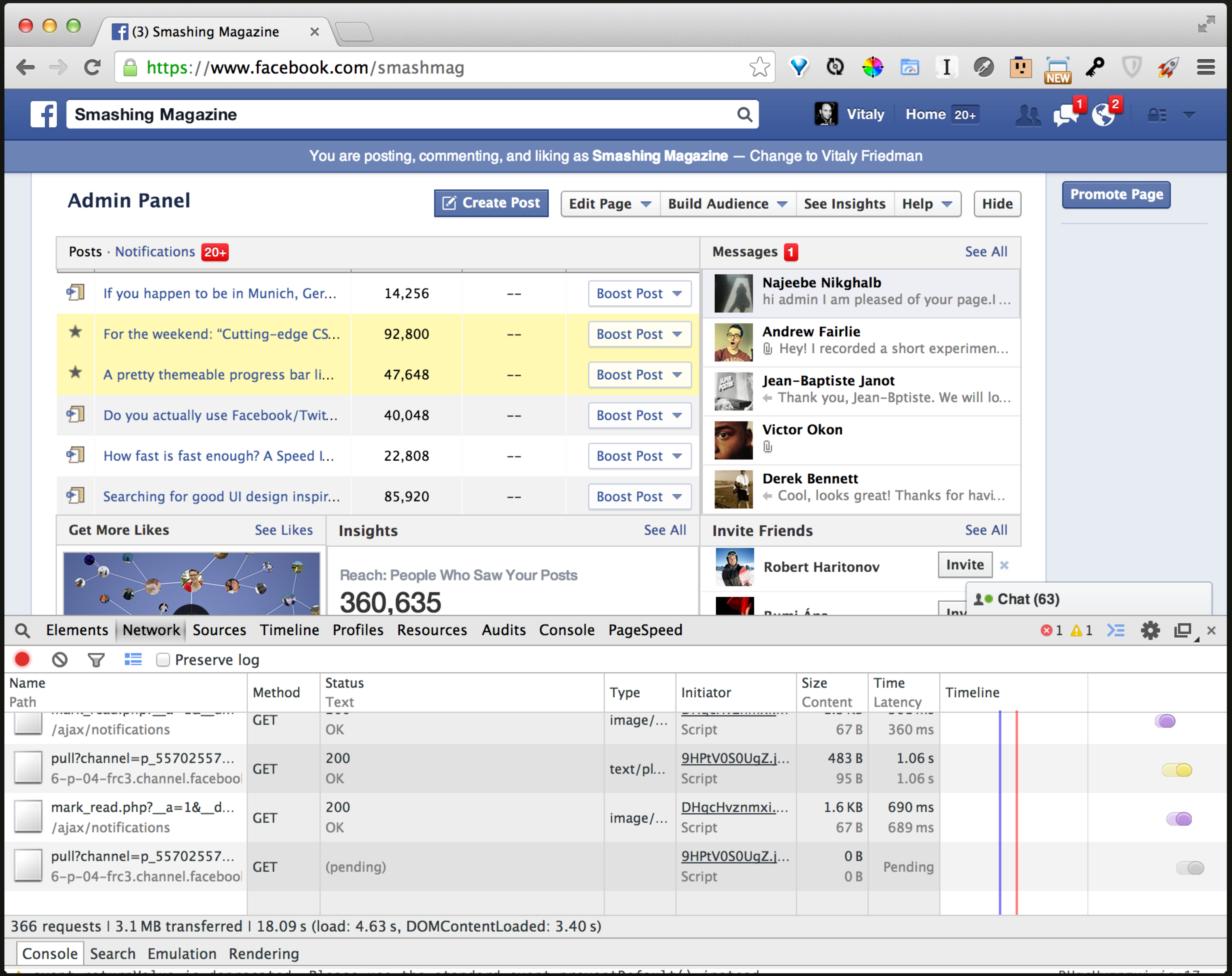
Building with HTTP/2 Support

Should you build from a release, you will need to `configure` first. There are tons of options. The ones specific for HTTP/2 are:

- `--enable-http2`
  This enables the module 'http2' which does implement the protocol inside the Apache server.
- `--with-nghttp2=<dir>`

— Requires *server-side* and *client-side* implementations.

— In Apache httpd 2.4.17, NGINX 1.9.5, NGINX Plus R7.



You are posting, commenting, and liking as Smashing Magazine — Change to Vitaly Friedman

Post	Reach		
If you happen to be in Munich, Ger...	14,256	--	Boost Post
For the weekend: "Cutting-edge CS...	92,800	--	Boost Post
A pretty themeable progress bar li...	47,648	--	Boost Post
Do you actually use Facebook/Twit...	40,048	--	Boost Post
How fast is fast enough? A Speed I...	22,808	--	Boost Post
Searching for good UI design inspir...	85,920	--	Boost Post

Najeebe Nikghalb
hi admin I am pleased of your page.I ...

Andrew Fairlie
Hey! I recorded a short experimen...

Jean-Baptiste Janot
Thank you, Jean-Bptiste. We will lo...

Victor Okon

Derek Bennett
Cool, looks great! Thanks for havi...

Robert Haritonov

Reach: People Who Saw Your Posts
360,635

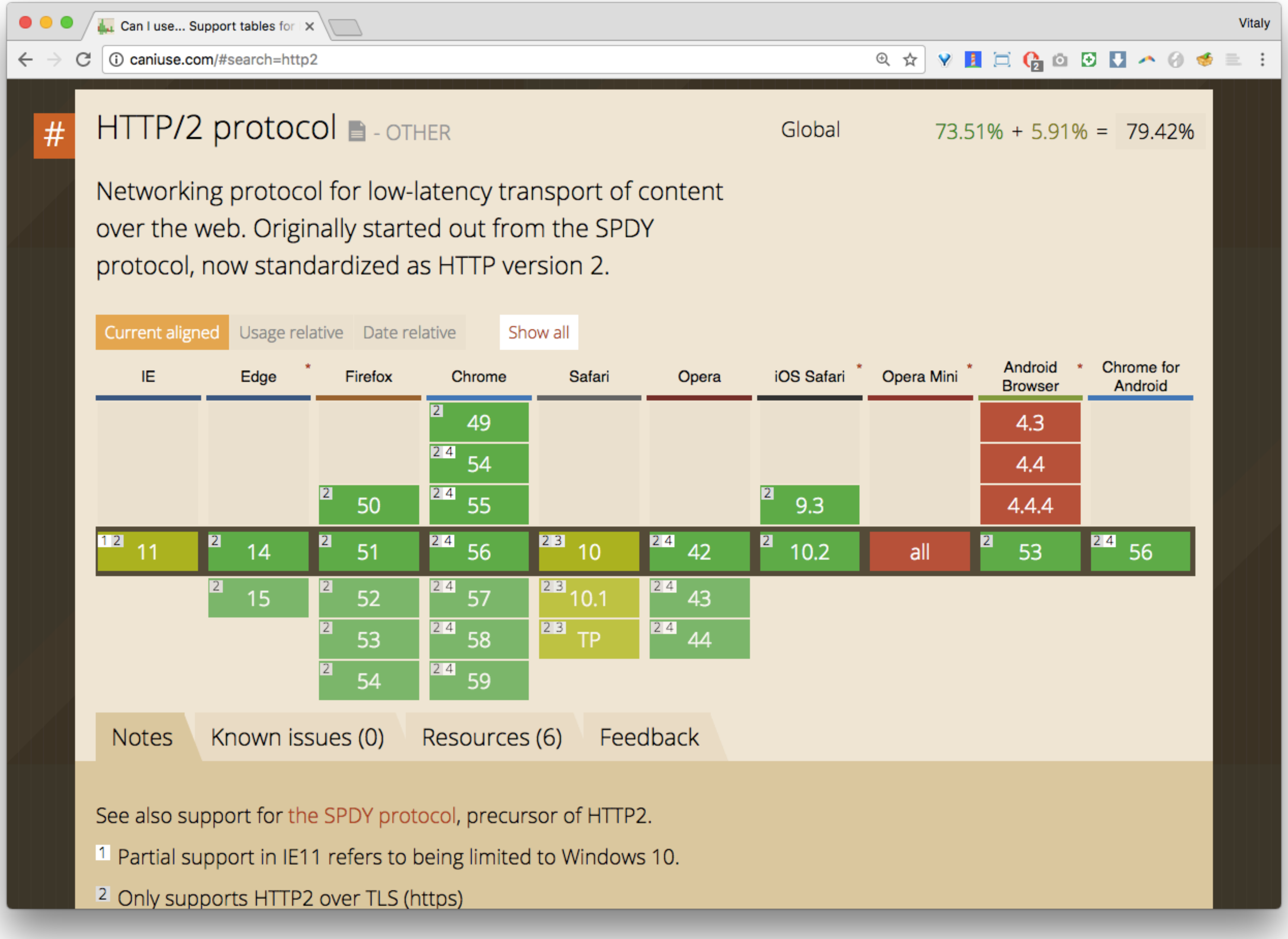
Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
/ajax/notifications	GET	OK	image/...	Script	67 B	360 ms	
pull?channel=p_55702557... 6-p-04-frc3.channel.faceboo	GET	200 OK	text/pl...	9HPtV0S0UqZ.j... Script	483 B 95 B	1.06 s 1.06 s	
mark_read.php?__a=1&__d... /ajax/notifications	GET	200 OK	image/...	DHqcHvznmxi... Script	1.6 KB 67 B	690 ms 689 ms	
pull?channel=p_55702557... 6-p-04-frc3.channel.faceboo	GET	(pending)		9HPtV0S0UqZ.j... Script	0 B 0 B	Pending	

366 requests | 3.1 MB transferred | 18.09 s (load: 4.63 s, DOMContentLoaded: 3.40 s)

— Requires *server-side* and *client-side* implementations.

— In Apache httpd 2.4.17, NGINX 1.9.5, NGINX Plus R7.

— Used by *Gmail, WordPress, Facebook, Twitter, Cloudflare.*



HTTP/2

- *HTTP/2.0* promises speed improvement, decreased network latency and better management of assets.

  — *64%* reduction in page load times *(23% on mobile)*,

  — *Unlimited number of parallel requests* per connection,

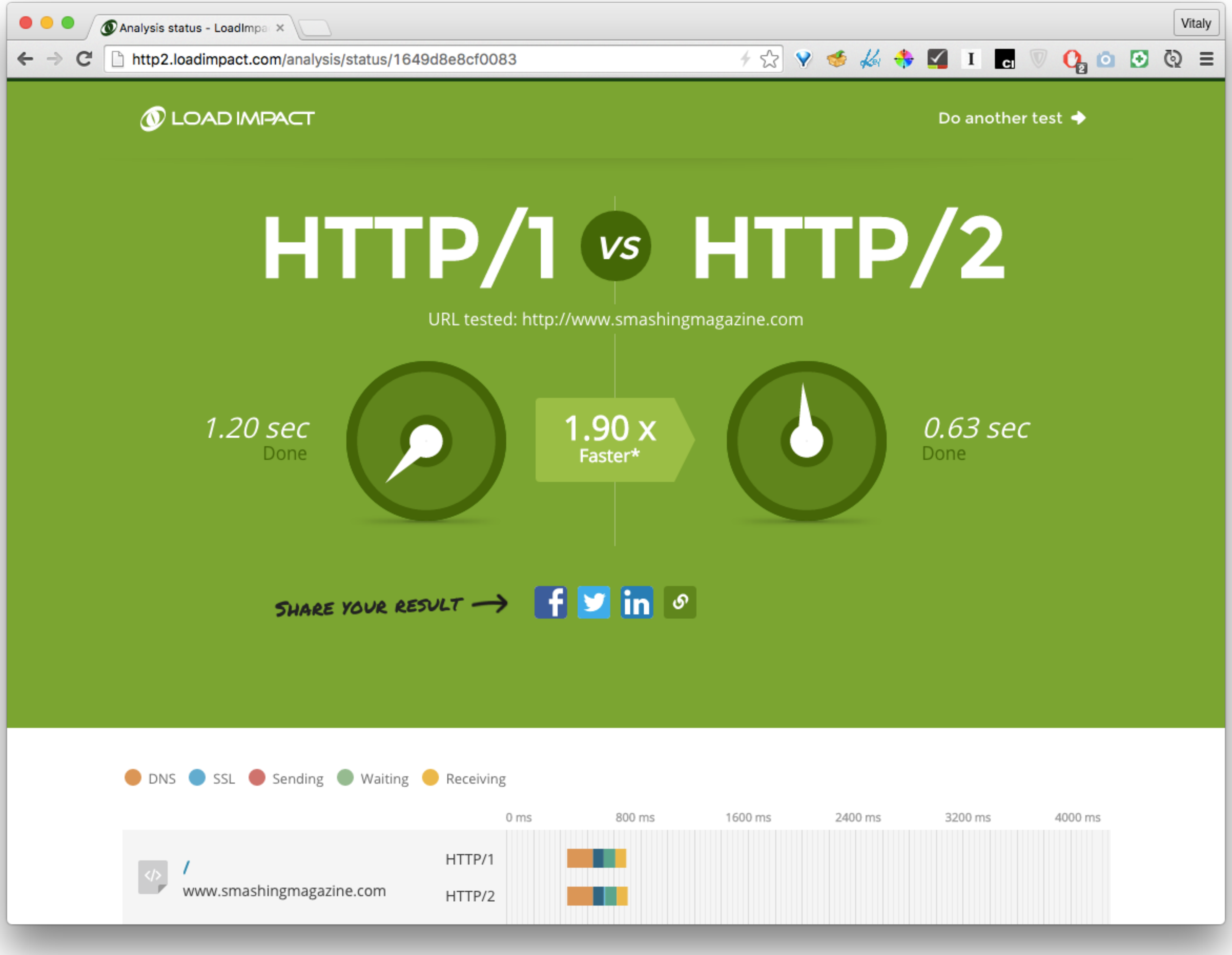
  — Quicker slow-start and better compression,

  — *One connection per host* handles all requests,

  — Developers can *prioritize* and *push* resources,

  — *Browsers* require the protocol to run over HTTPS,

  — Extension of HTTP/1.1; as such, falls back to *HTTP/1.1*.


Analysis status - LoadImpa
Vitaly
http2.loadimpact.com/analysis/status/1649d8e8cf0083
LOAD IMPACT
Do another test

HTTP/1 vs HTTP/2

URL tested: http://www.smashingmagazine.com

1.20 sec
Done

1.90 x
Faster*

0.63 sec
Done

● DNS ● SSL ● Sending ● Waiting ● Receiving

/
www.smashingmagazine.com

HTTP/1

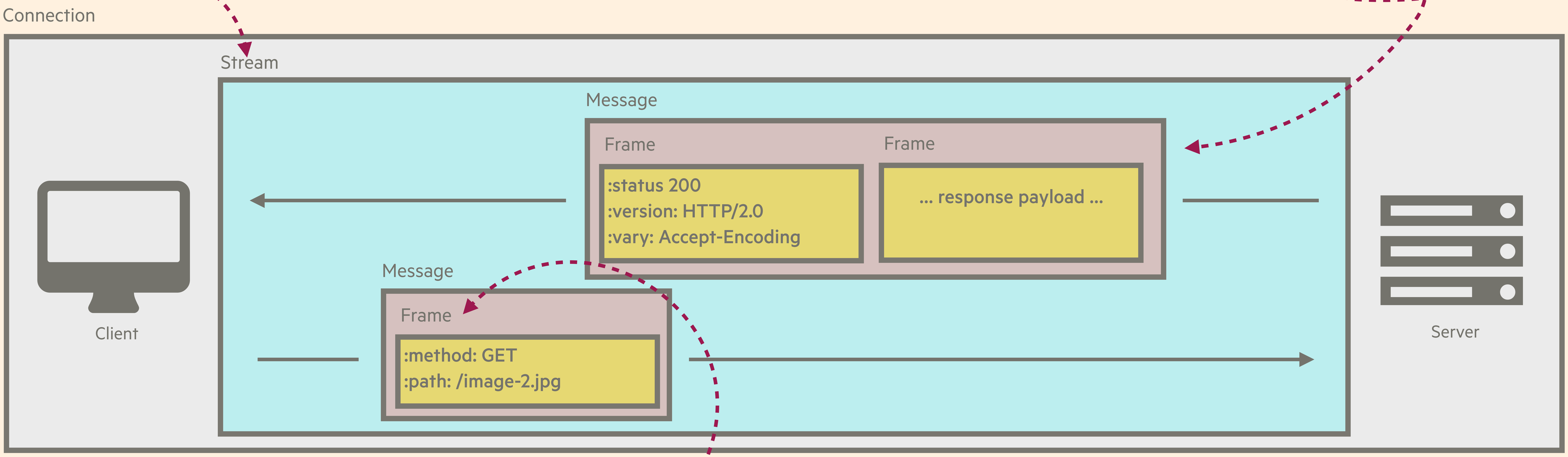
HTTP/2

HTTP/2 core features

1. **Multiplexing** (via streams, frames, and messages)
2. Binary data format
3. **Prioritisation**
4. Header compression
5. Flow control
6. **Server push**

Stream
A virtual channel within an established connection which carries bidirectional messages.

Message
A complete sequence of frames that map to a logical HTTP message, such as a request.



```
:status 200
:version: HTTP/2.0
:vary: Accept-Encoding
```

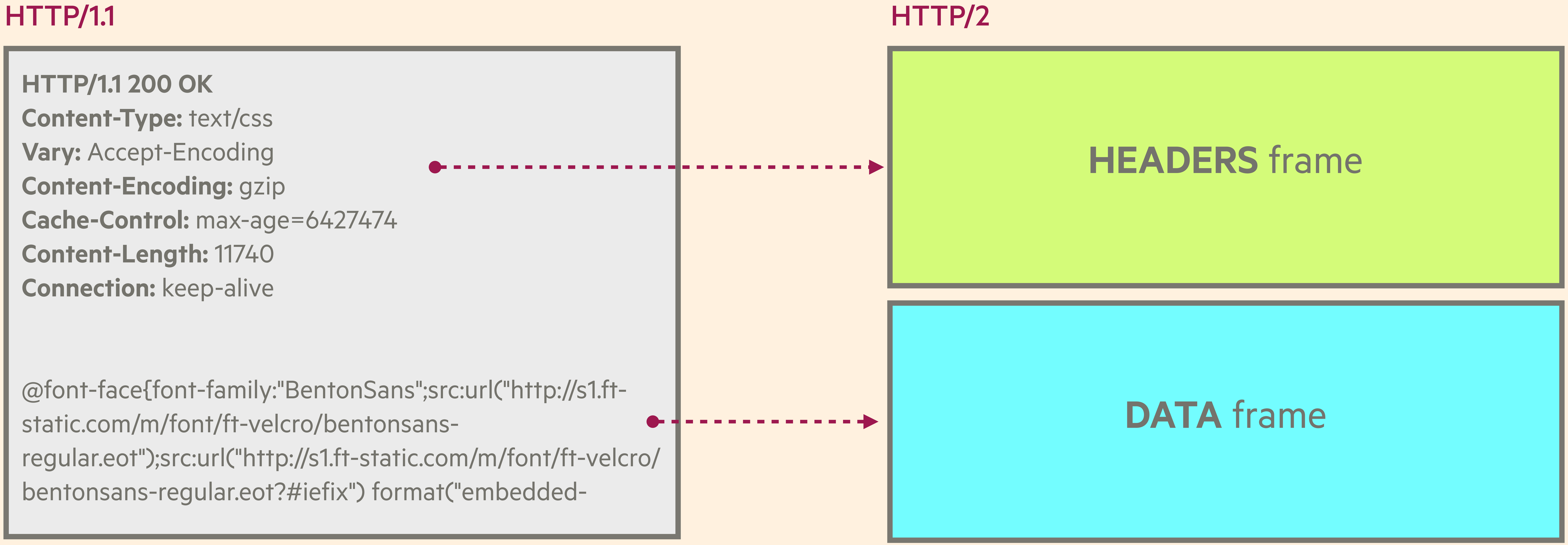
... response payload ...

```
:method: GET
:path: /image-2.jpg
```

Frame
The smallest unit of communication, which carries a specific type of data—e.g., HTTP headers, payload, commands e.t.c.

Multiplexing: terminology

Multiplexing: Frames



HTTP/1.1

HTTP/1.1 200 OK
Content-Type: text/css
Vary: Accept-Encoding
Content-Encoding: gzip
Cache-Control: max-age=6427474
Content-Length: 11740
Connection: keep-alive

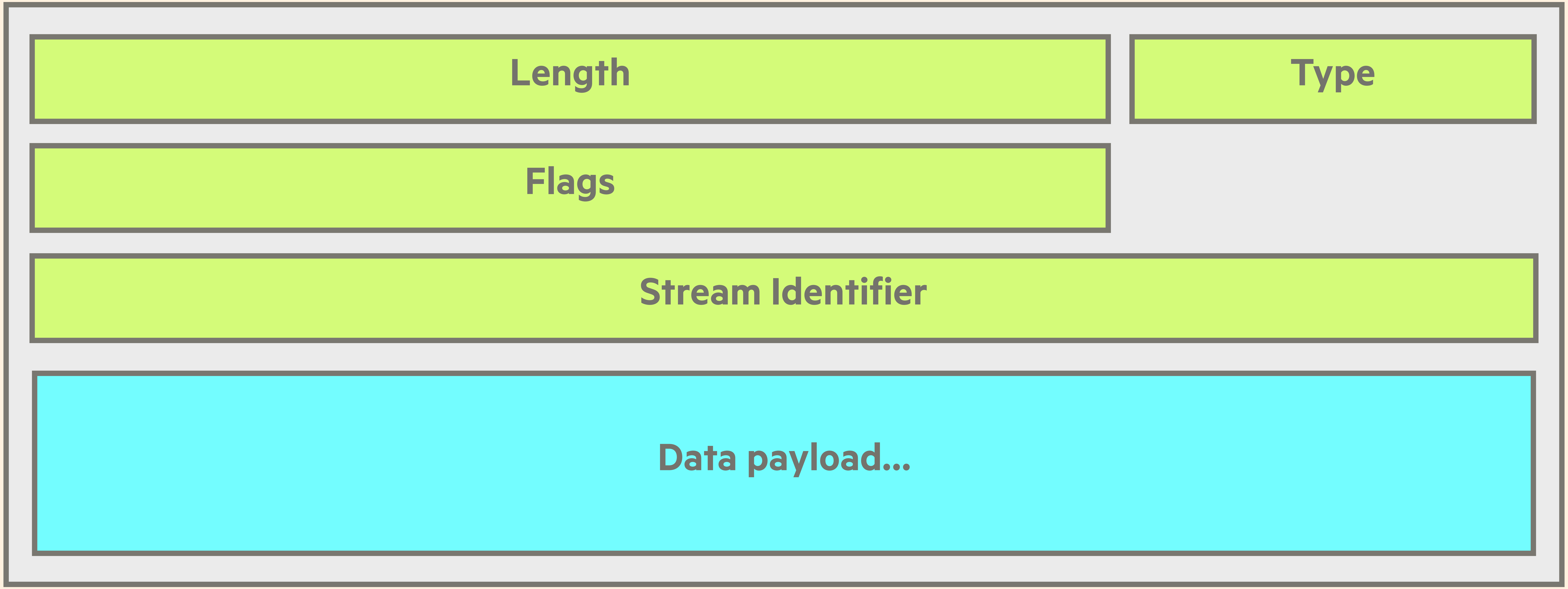
@font-face{font-family:"BentonSans";src:url("http://s1.ft-static.com/m/font/ft-velcro/bentonsans-regular.eot");src:url("http://s1.ft-static.com/m/font/ft-velcro/bentonsans-regular.eot?#iefix") format("embedded-

HTTP/2

HEADERS frame

DATA frame

Binary frames



Length

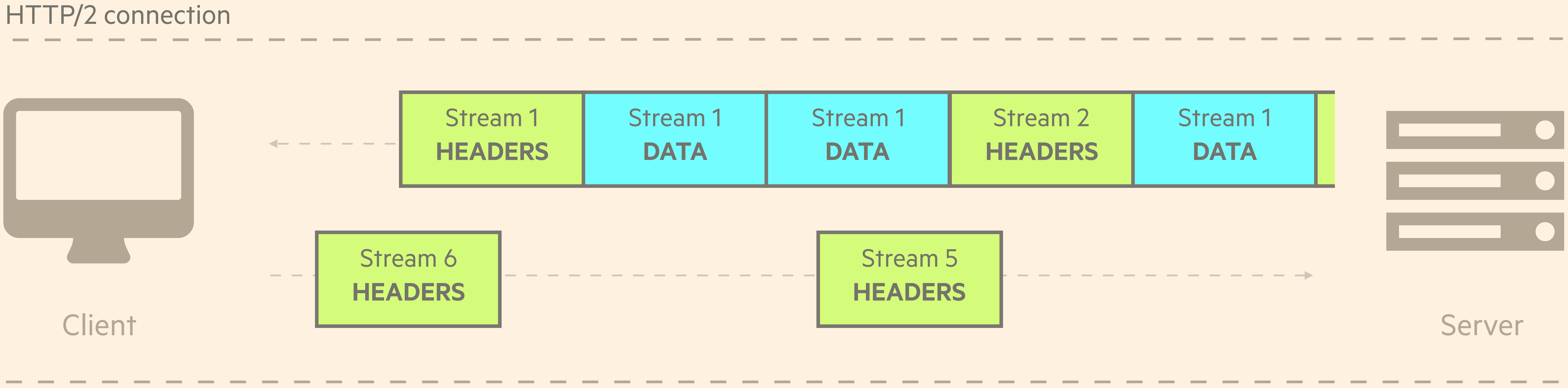
Type

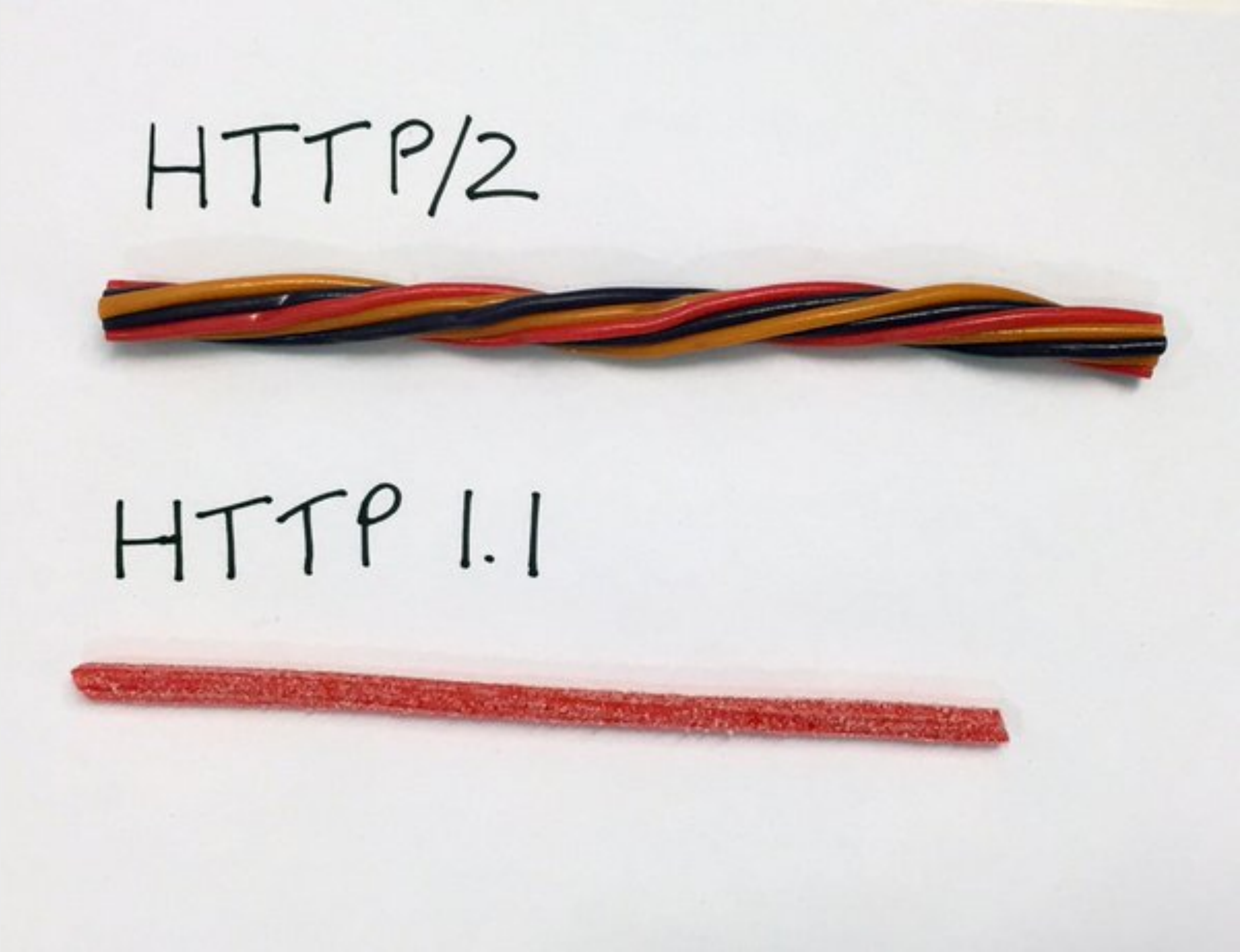
Flags

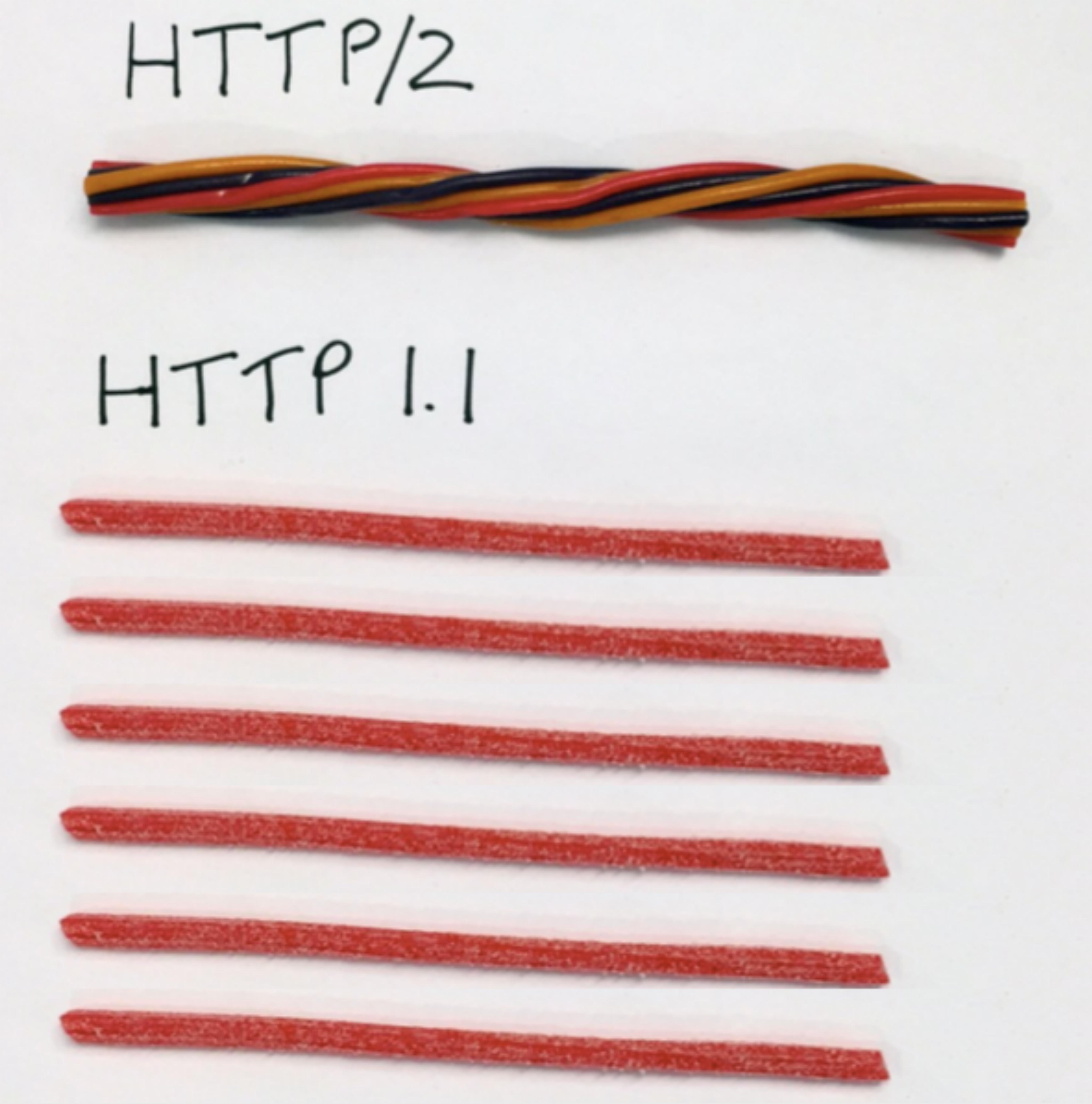
Stream Identifier

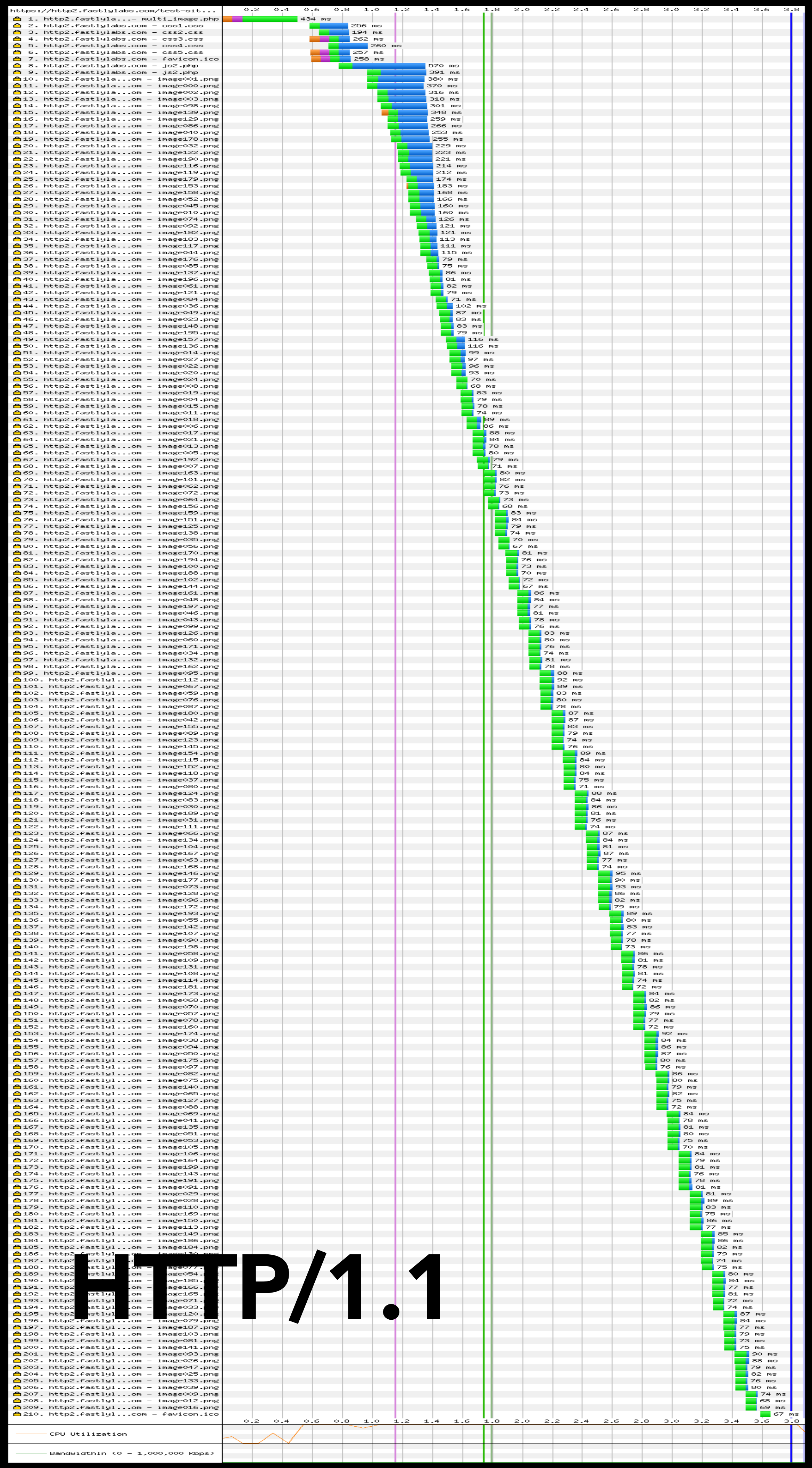
Data payload...

Multiplexing: Streams

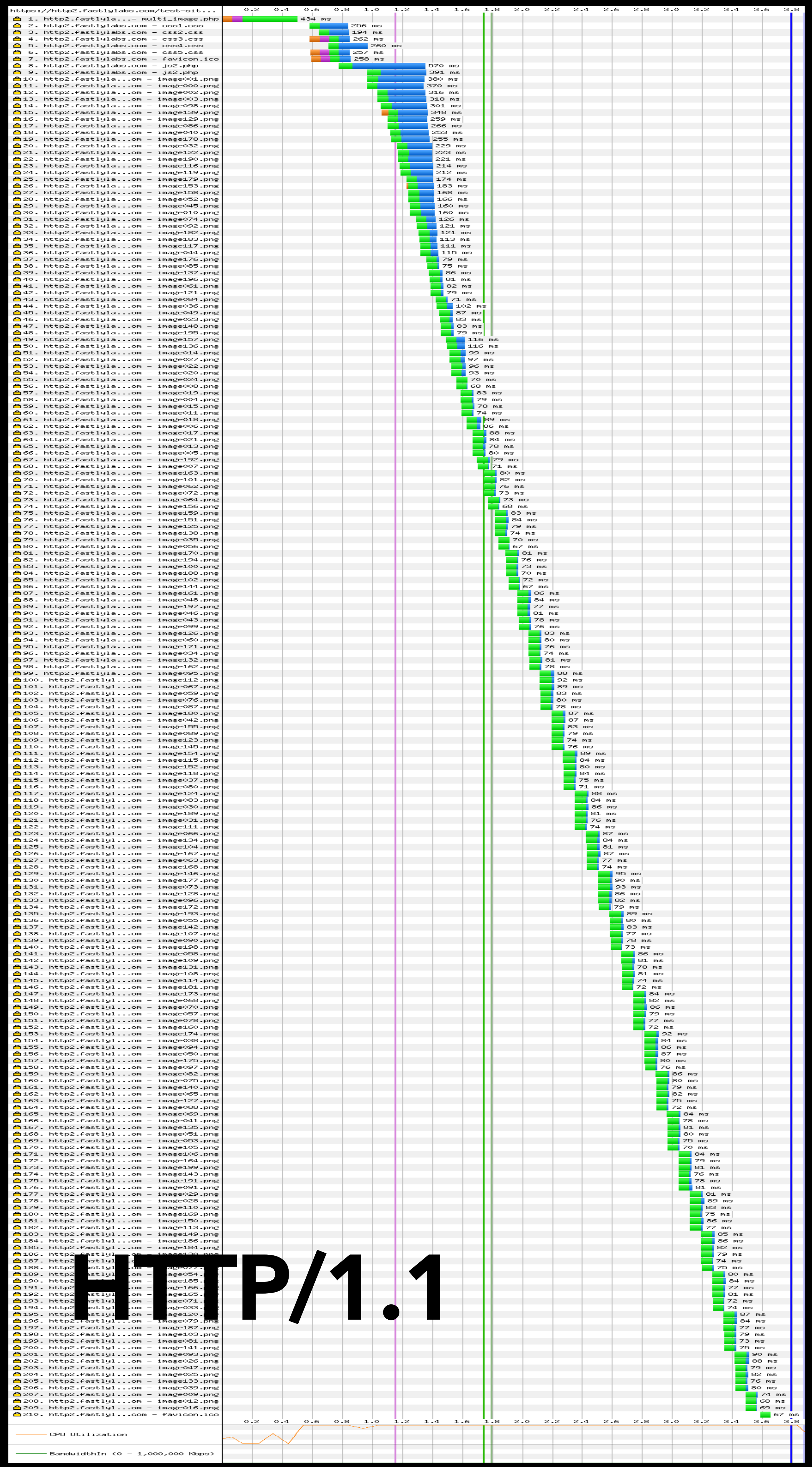



HTTP/2
HTTP 1.1

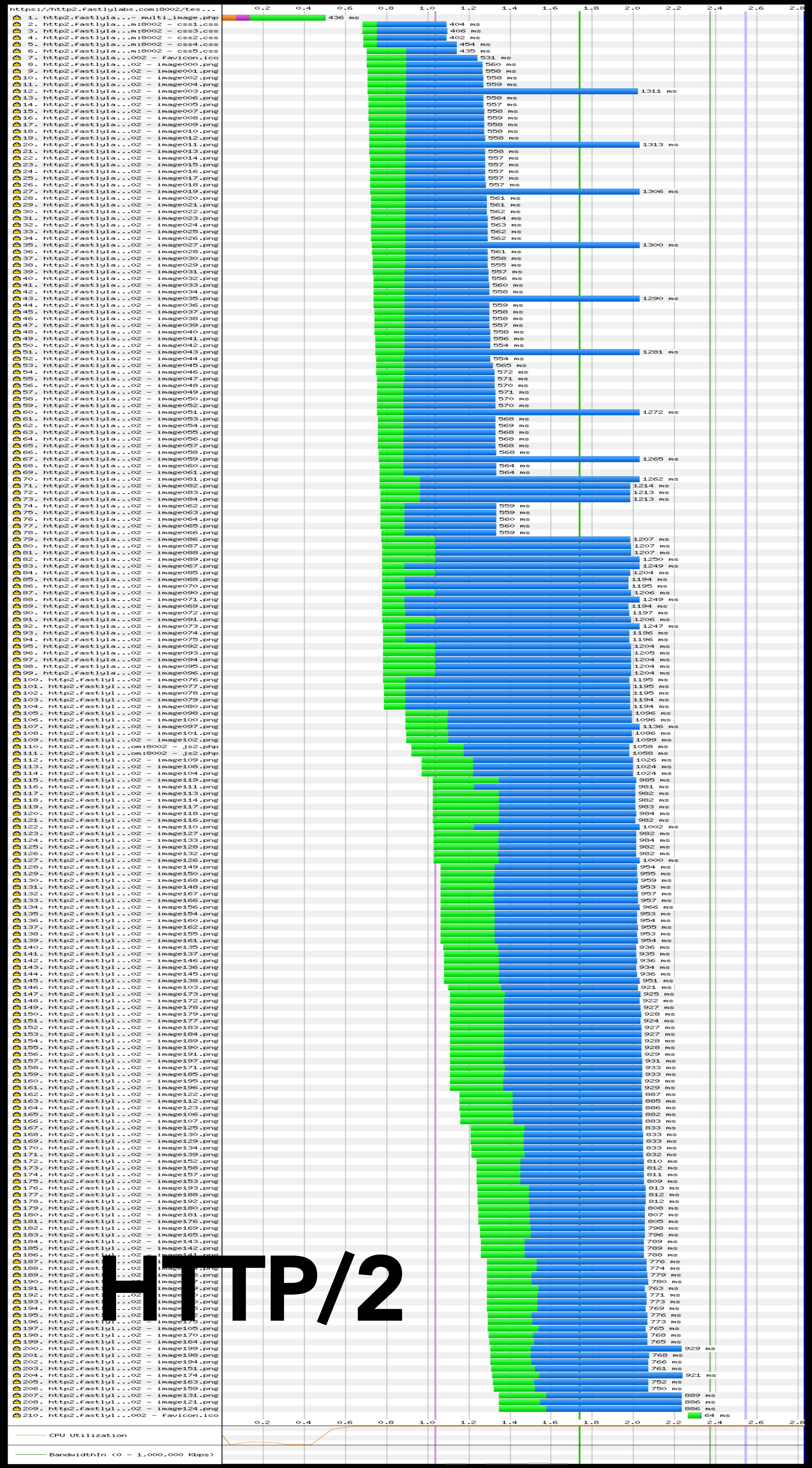

HTTP/2
HTTP 1.1



HTTP/1.1



HTTP/1.1



HTTP/2

HTTP/2

- *HTTP/2.0* promises speed improvement, decreased network latency and better management of assets.

  — *64%* reduction in page load times *(23% on mobile)*,

  — *Unlimited number of parallel requests* per connection,

  — Quicker slow-start and better compression,

  — *One connection per host* handles all requests,

  — Developers can *prioritize* and *push* resources,

  — *Browsers* require the protocol to run over HTTPS,

  — Extension of HTTP/1.1; as such, falls back to *HTTP/1.1*.

Keeping score…

	0% PLR				2% PLR			
	5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms		5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms	
	🦊	◉	🦊	◉	🦊	◉	🦊	◉
DocComplete	h2	h2	h2	h2	h1	h1	h1	h1
DCL Start	h1	h1	h2	h1	h1	h1	h2	h1
Speed Index	h2/h1	h2	h2	h2	h1	h1	h2	h2

		0% PLR				2% PLR			
		5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms		5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms	
		🦊	🌐	🦊	🌐	🦊	🌐	🦊	🌐
Site1a (Fastly)	DocComplete	h2	h2	h2	h1	h1	h1	h1	h1
	DCL Start	h2	h1	h2	h2	h2/h1	h1	h2	h2
	Speed Index	h1	h2	h2	h2	h1	h2/h1	h2/h1	h2
Site1b	DocComplete	h2/h1	h2	h2	h2	h1	h2	h1	h2/h1
	DCL Start	h1	h2	h1	h1	h1	h2/h1	h1	h1
	Speed Index	h1	h2	h2	h1	h1	h2/h1	h1	h1
Site1c	DocComplete	h1/h2	h2	h2	h2	h1	h1	h1	h1
	DCL Start	h1	h1/h2	h1	h1	h1	h2	h1	h1
	Speed Index	h2	h2	h1	h2	h1	h2	h1	h1
Site2a	DocComplete	h2	h2	h2	h2	h1	h2/h1	h1	h1
	DCL Start	h2	h2	h2	h2	h1	h1	h1	h1
	Speed Index	h1	h2	h1	h2	h1	h2	h1	h2
Site2b	DocComplete	h2	h2	h2	h2	h1	h1/h2	h1	h1
	DCL Start	h2	h2	h1	h2	h1	h2	h1	h2
	Speed Index	h2	h1/h2	h1	h1/h2	h2	h2	h1	h1
Site3a	DocComplete	h2	h2	h1	h2	h2	h2	h1	h1
	DCL Start	h2	h2	h2	h2	h2	h2	h2	h2
	Speed Index	h2	h2	h1	h1	h1/h2	h1/h2	h1	h1
Site3b	DocComplete	h2	h2	h2	h1/h2	h2	h2/h1	h2	h2
	DCL Start	h2	h2	h2	h2	h2	h2	h2	h2
	Speed Index	h1	h2	h1	h1	h1	h2	h1	h1
Site3c	DocComplete	h1	h2	h2	h2	h1	h2	h2	h2
	DCL Start	h1/h2	h2	h1	h1/h2	h2/h1	h2	h1	h2/h1
	Speed Index	h1	h2	h2	h2	h2	h2	h2	h2

		0% PLR				2% PLR			
		5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms		5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms	
		🦊	🌐	🦊	🌐	🦊	🌐	🦊	🌐
Site1a (Fastly)	DocComplete	h2	h2	h2	h1	h1	h1	h1	h1
	DCL Start	h2	h1	h2	h2	h2/h1	h1	h2	h2
	Speed Index	h1	h2	h2	h2	h1	h2/h1	h2/h1	h2
Site1b	DocComplete	h2/h1	h2	h2	h2	h1	h2	h1	h2/h1
	DCL Start	h1	h2	h1	h1	h1	h2/h1	h1	h1
	Speed Index	h1	h2	h2	h1	h1	h2/h1	h1	h1
Site1c	DocComplete	h1/h2	h2	h2	h2	h1	h1	h1	h1
	DCL Start	h1	h1/h2	h1	h1	h1	h2	h1	h1
	Speed Index	h2	h2	h1	h2	h1	h2	h1	h1
Site2a	DocComplete	h2	h2	h2	h2	h1	h2/h1	h1	h1
	DCL Start	h2	h2	h2	h2	h1	h1	h1	h1
	Speed Index	h1	h2	h1	h2	h1	h2	h1	h2
Site2b	DocComplete	h2	h2	h2	h2	h1	h1/h2	h1	h1
	DCL Start	h2	h2	h1	h2	h1	h2	h1	h2
	Speed Index	h2	h1/h2	h1	h1/h2	h2	h2	h1	h1
Site3a	DocComplete	h2	h2	h1	h2	h2	h2	h1	h1
	DCL Start	h2	h2	h2	h2	h2	h2	h2	h2
	Speed Index	h2	h2	h1	h1	h1/h2	h1/h2	h1	h1
Site3b	DocComplete	h2	h2	h2	h1/h2	h2	h2/h1	h2	h2
	DCL Start	h2	h2	h2	h2	h2	h2	h2	h2
	Speed Index	h1	h2	h1	h1	h1	h2	h1	h1
Site3c	DocComplete	h1	h2	h2	h2	h1	h2	h2	h2
	DCL Start	h1/h2	h2	h1	h1/h2	h2/h1	h2	h1	h2/h1
	Speed Index	h1	h2	h2	h2	h2	h2	h2	h2

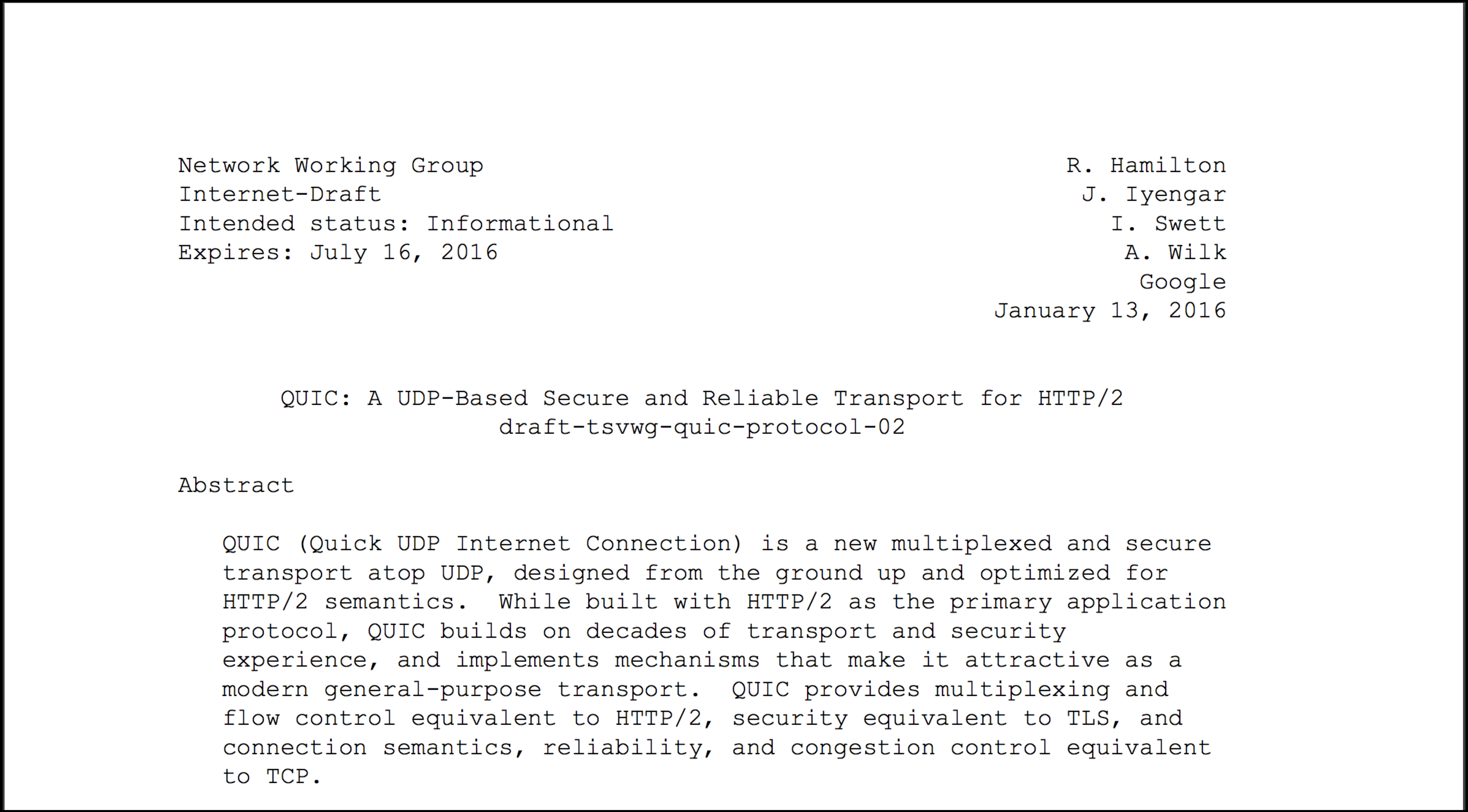
		0% PLR				2% PLR			
		5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms		5Mbps/1Mbps; 40ms		780Kbps/330Kbps; 200ms	
Site1a (Fastly)	DocComplete	h2	h2	h2	h1	h1	h1	h1	h1
	DCL Start	h2	h1	h2	h2	h2/h1	h1	h2	h2
	Speed Index	h1	h2	h2	h2	h1	h2/h1	h2/h1	h2
Site1b	DocComplete	h2/h1	h2	h2	h2	h1	h2	h1	h2/h1
	DCL Start	h1	h2	h1	h1	h1	h2/h1	h1	h1
	Speed Index	h1	h2	h2	h1	h1	h2/h1	h1	h1
Site1c	DocComplete	h1/h2	h2	h2	h2	h1	h1	h1	h1
	DCL Start	h1	h1/h2	h1	h1	h1	h2	h1	h1
	Speed Index	h2	h2	h1	h2	h1	h2	h1	h1
Site2a	DocComplete	h2	h2	h2	h2	h1	h2/h1	h1	h1
	DCL Start	h2	h2	h2	h2	h1	h1	h1	h1
	Speed Index	h1	h2	h1	h2	h1	h2	h1	h2
Site2b	DocComplete	h2	h2	h2	h2	h1	h1/h2	h1	h1
	DCL Start	h2	h2	h1	h2	h1	h2	h1	h2
	Speed Index	h2	h1/h2	h1	h1/h2	h2	h2	h1	h1
Site3a	DocComplete	h2	h2	h1	h2	h2	h2	h1	h1
	DCL Start	h2	h2	h2	h2	h2	h2	h2	h2
	Speed Index	h2	h2	h1	h1	h1/h2	h1/h2	h1	h1
Site3b	DocComplete	h2	h2	h2	h1/h2	h2	h2/h1	h2	h2
	DCL Start	h2	h2	h2	h2	h2	h2	h2	h2
	Speed Index	h1	h2	h1	h1	h1	h2	h1	h1
Site3c	DocComplete	h1	h2	h2	h2	h1	h2	h2	h2
	DCL Start	h1/h2	h2	h1	h1/h2	h2/h1	h2	h1	h2/h1
	Speed Index	h1	h2	h2	h2	h2	h2	h2	h2

PLR in the real world

UNITED STATES (MEASUREMENTS: 83744587)

0% | 0-1.5% | > 1.5%

percentage
100
90
80
70
60
50
40
30
20
10
0
10
12.5
15
17.5
20
22.5
25
27.5
30
32.5
35
37.5
40
42.5
45
47.5
50
52.5
55
57.5
60
latency (ms)
🟧 0
🟦 0-1.5
🟩 >1.5



Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 16, 2016

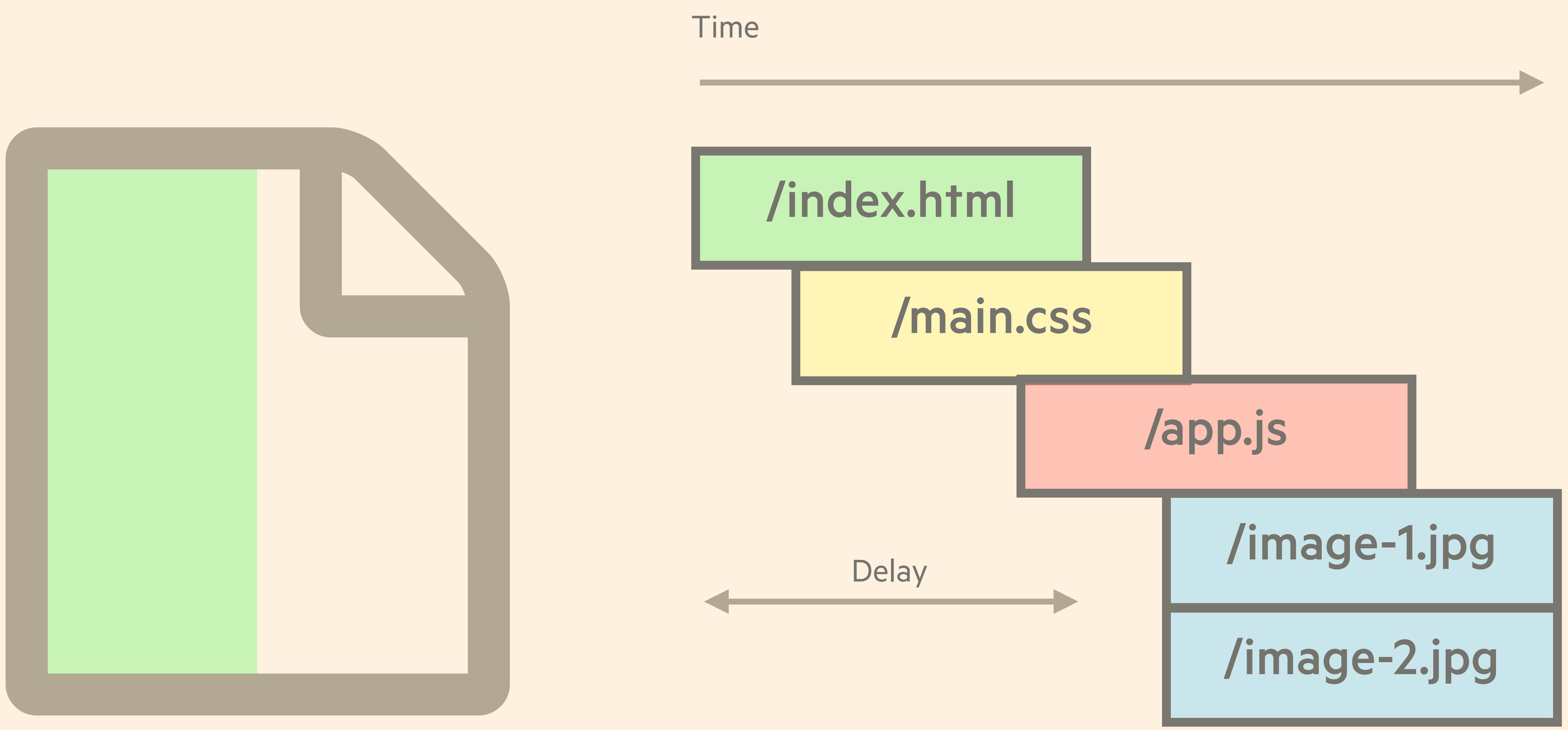
R. Hamilton
J. Iyengar
I. Swett
A. Wilk
Google
January 13, 2016

QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2
draft-tsvwg-quic-protocol-02

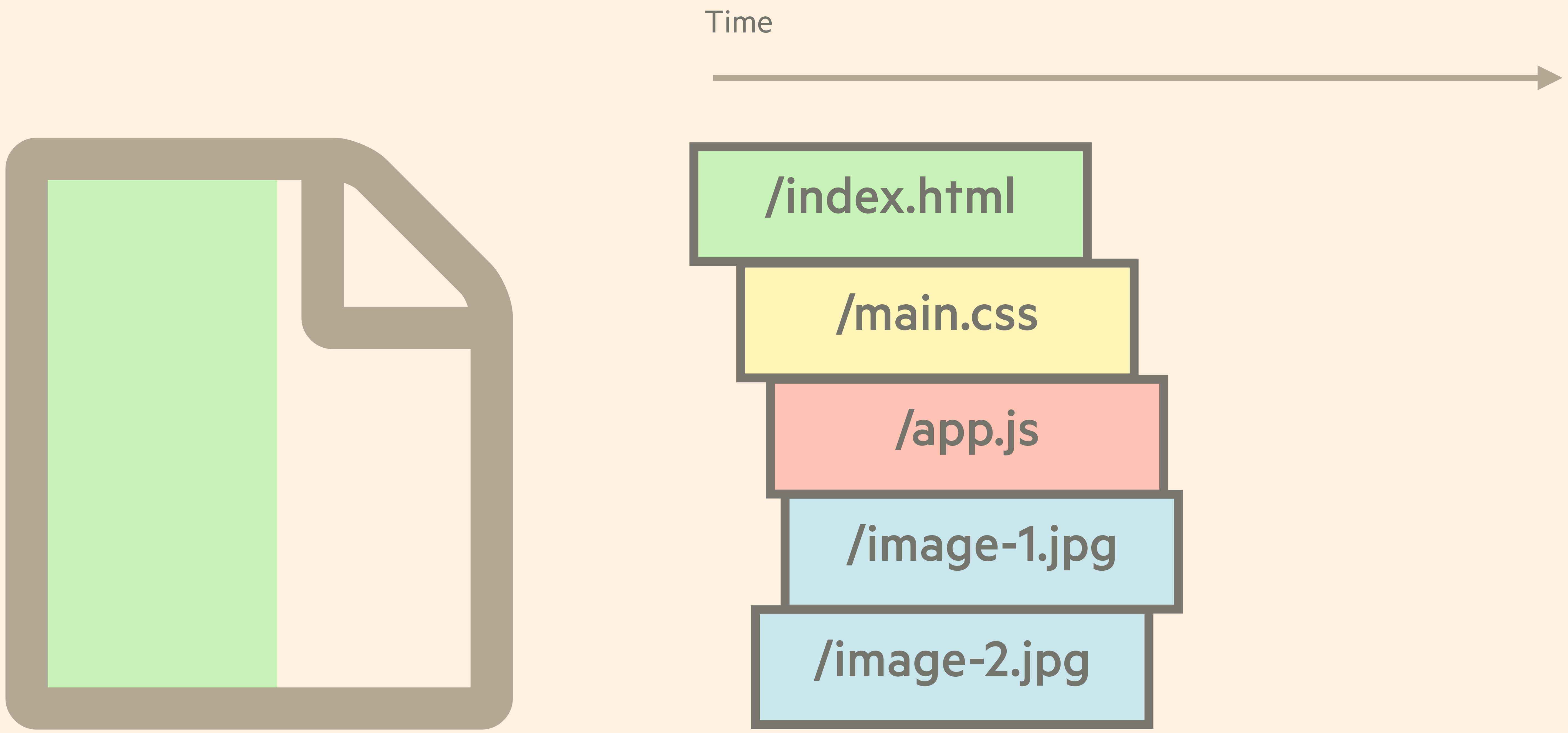
Abstract

QUIC (Quick UDP Internet Connection) is a new multiplexed and secure transport atop UDP, designed from the ground up and optimized for HTTP/2 semantics. While built with HTTP/2 as the primary application protocol, QUIC builds on decades of transport and security experience, and implements mechanisms that make it attractive as a modern general-purpose transport. QUIC provides multiplexing and flow control equivalent to HTTP/2, security equivalent to TLS, and connection semantics, reliability, and congestion control equivalent to TCP.

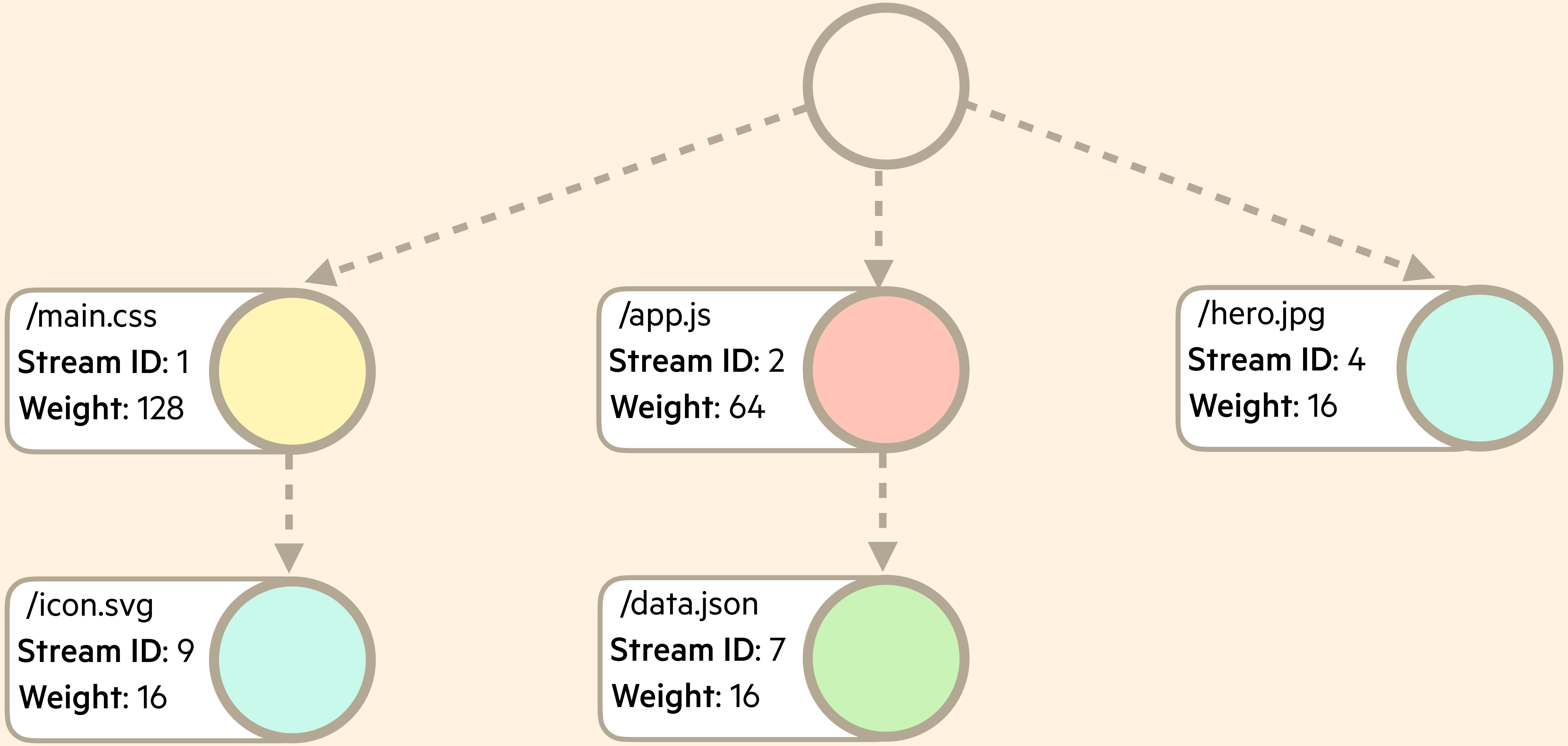
HTTP/1.1 Prioritisation



HTTP/2 Prioritisation



HTTP/2 Prioritisation



/main.css
Stream ID: 1
Weight: 128

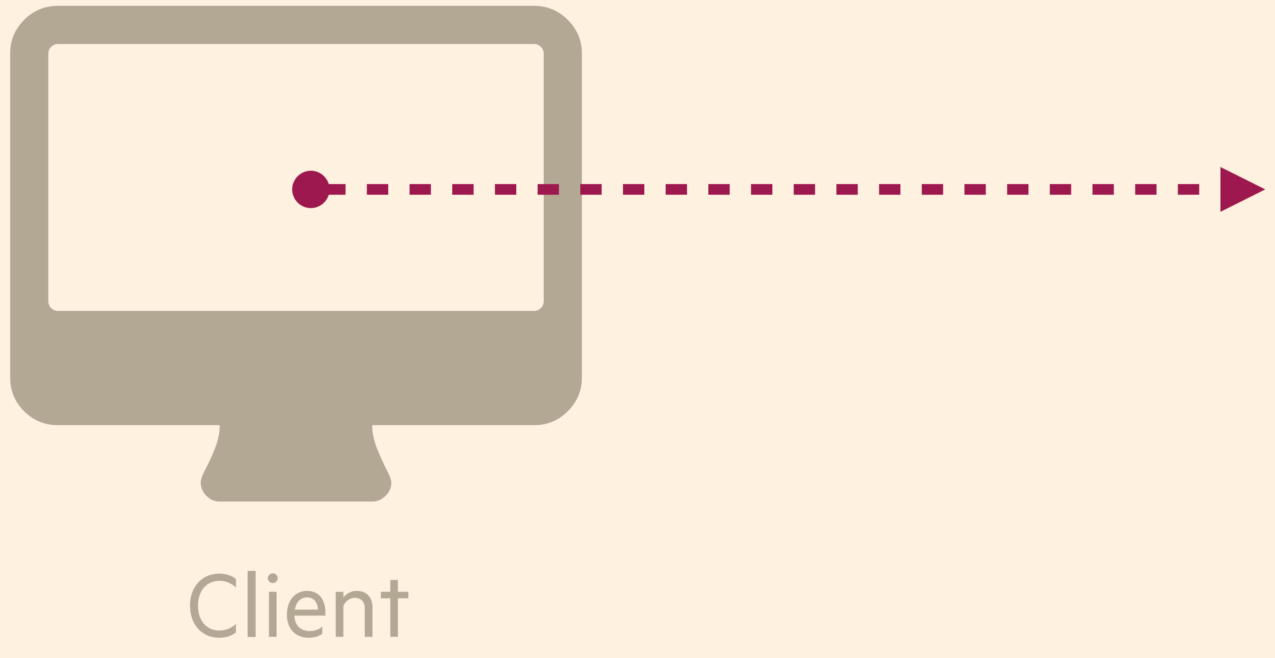
/app.js
Stream ID: 2
Weight: 64

/hero.jpg
Stream ID: 4
Weight: 16

/icon.svg
Stream ID: 9
Weight: 16

/data.json
Stream ID: 7
Weight: 16

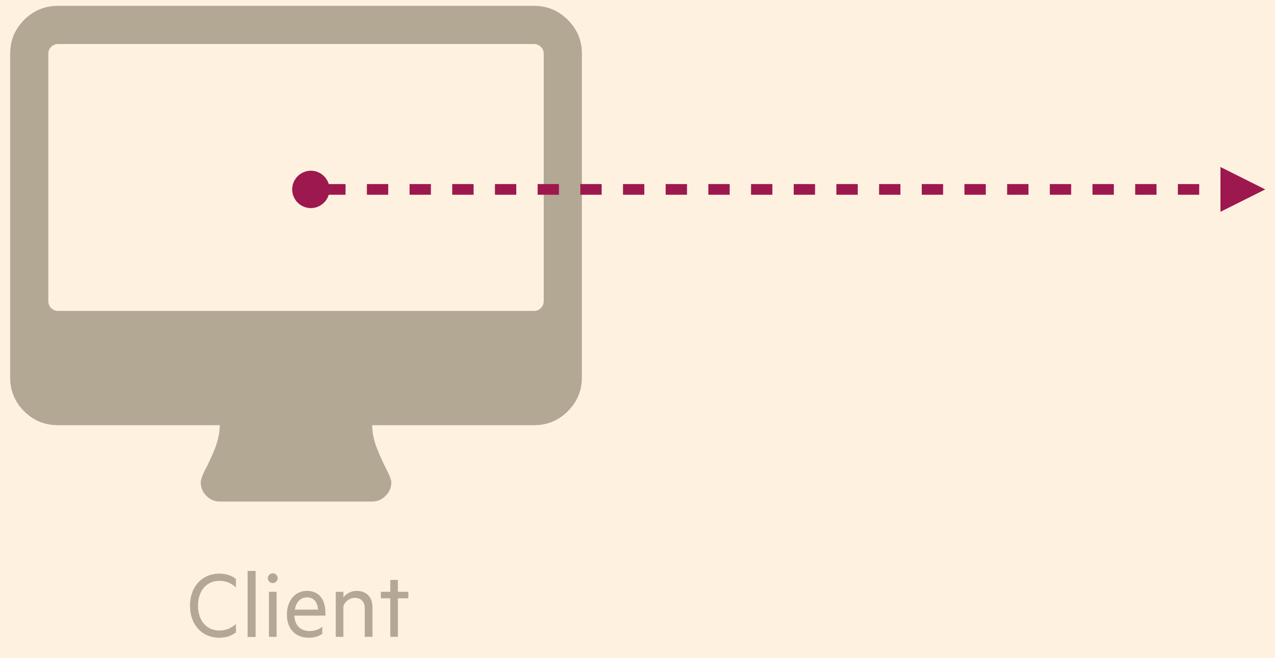
Header compression: HPACK



:method:	GET
:scheme:	https
:host:	next.ft.com
:path:	/main.css

Cookie	FTUserTrack=213.216.148.1.1432658066641353;
	SIVISITOR=Mi45NzIuMjIzMDc2NzM2NTU0NS4xNDMyNzI0MTE2NDc4LjZmM2U4YmFj*;
	__gads=ID=0d68ab230d47cf5f:T=1432724114:S=ALNI_MZykGfLfvkhayKSL3LXYT9YQNtRjg;
	cookieconsent=accepted;

Header compression: HPACK

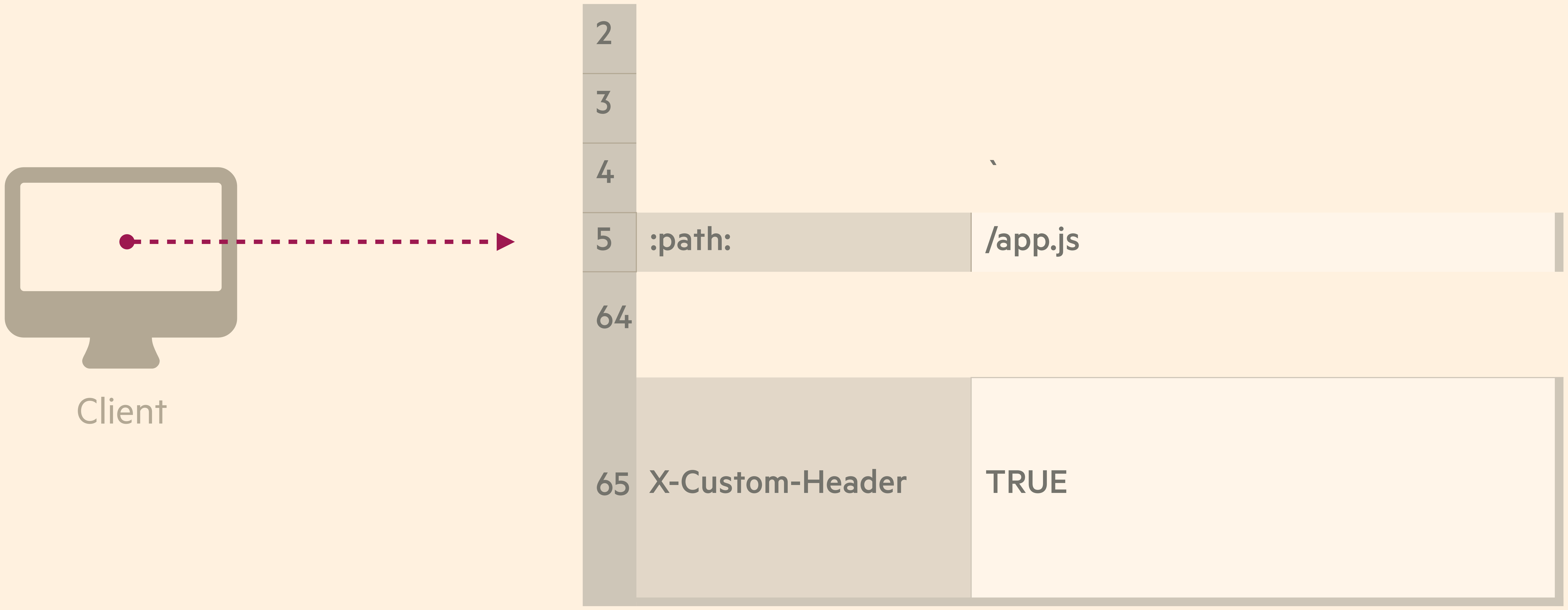


2	:method:	GET
3	:scheme:	https
4	:host:	next.ft.com
5	:path:	/main.css
64	Cookie	FTUserTrack=213.216.148.1.1432658066641353; SIVISITOR=Mi45NzIuMjIzMDc2NzM2NTU0NS4xNDMyNzI0MTE2NDc4LjZmM2U4YmFj*; __gads=ID=0d68ab230d47cf5f:T=1432

Static table

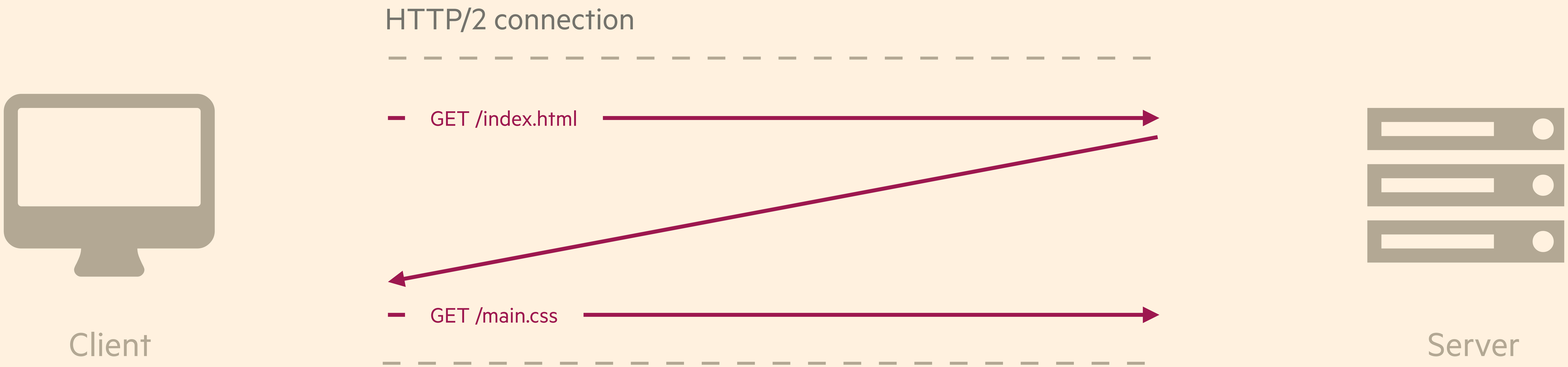
Dynamic table

Header compression: HPACK

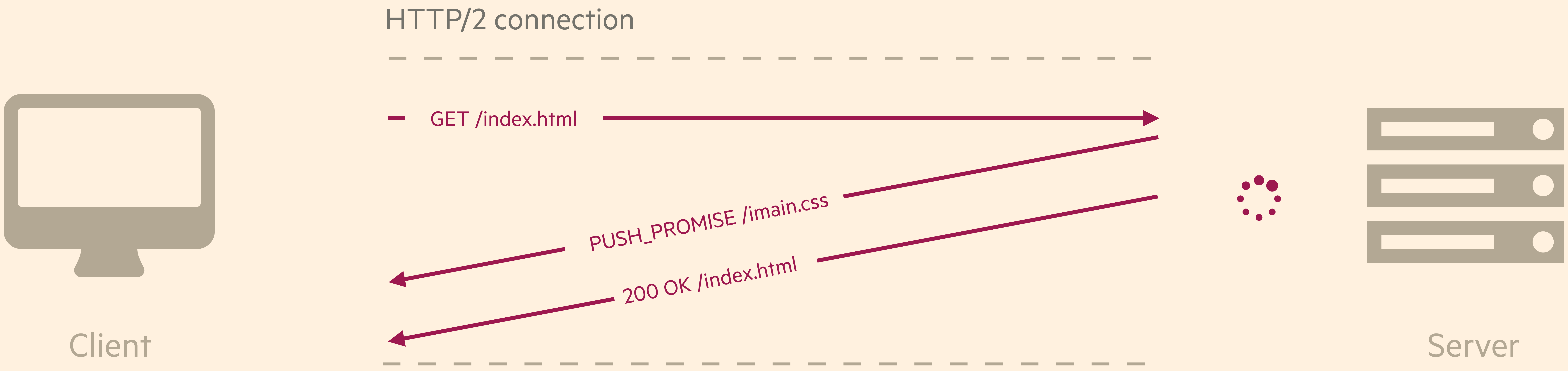


2		
3		
4		`
5	:path:	/app.js
64		
65	X-Custom-Header	TRUE

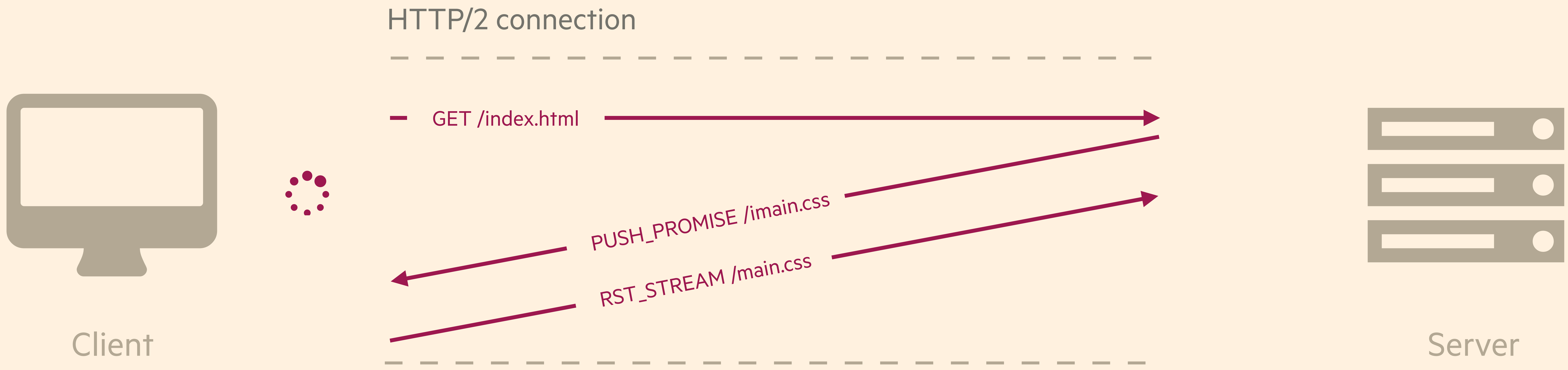
Push



Push


HTTP/2 connection
GET /index.html
PUSH_PROMISE /imain.css
200 OK /index.html
Client
Server

Push



HTTP/2 connection

GET /index.html

PUSH_PROMISE /imain.css

RST_STREAM /main.css

No push – first view

abs.com/push/cs...
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
.om – css_push.php
170 ms
s.com – css1.css
42 ms
s.com – css2.css
45 ms
s.com – css3.css
47 ms
s.com – css4.css
102 ms
.com – image00.png
118 ms
.com – image01.png
135 ms
.com – image02.png
151 ms
.com – image03.png
170 ms
.com – image04.png
188 ms
.com – image05.png
206 ms
.com – image06.png
240 ms
.com – image07.png
240 ms
.com – image08.png
268 ms
.com – image09.png
274 ms
.com – favicon.ico
51 ms

No push – repeat view

labs.com/push/cs...
..om - css_push.php
152 ms
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0.20
0.22
0.24
0.26
0.28
0.30
0.32
0.34
0.36

Push – first view

labs.com/push/cs...
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
...om - css_push.php
164 ms
abs.com - css1.css
3 ms
abs.com - css2.css
47 ms
abs.com - css3.css
50 ms
abs.com - css4.css
50 ms
Pushed
...com - image00.png
54 ms
...com - image01.png
70 ms
...com - image02.png
89 ms
...com - image03.png
107 ms
...com - image04.png
125 ms
...com - image05.png
144 ms
...com - image06.png
160 ms
...com - image07.png
194 ms
...com - image08.png
195 ms
...com - image09.png
211 ms
...com - favicon.ico
48 ms

Push – repeat view

.com/push/cs...
0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22 0.24 0.26 0.28 0.30 0.32 0.34 0.36 0.38 0.40 0.42 0.44 0.46
m - css_push.php
145 ms
com - css1.css
50 ms
com - css2.css
50 ms
com - css3.css
51 ms
com - css4.css
44 ms
Pushed
0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22 0.24 0.26 0.28 0.30 0.32 0.34 0.36 0.38 0.40 0.42 0.44 0.46

Severs

Server	Version	Push
NGINX	1.9.5+	No
Apache	2.4.17+	Yes
IIS	10	Yes
Jetty	9.3+	Yes

https://github.com/http2/http2-spec/wiki/Implementations

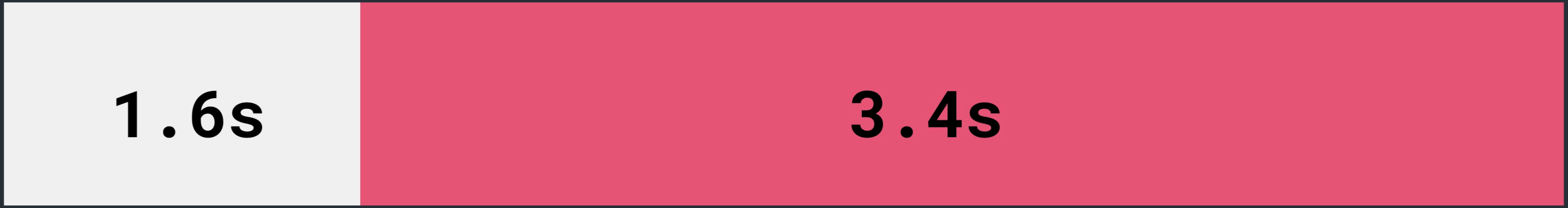
CDNs

Provider	Version	Push
Akamai	Yes	No
CloudFlare	Yes	No
KeyCDN	Yes	No
MaxCDN	No	No
Fastly	No	No
AWS CloudFront	No	No

https://istlsfastyet.com/

Time to Interactive Budget

5s



1.6s

3.4s

DNS LOOKUP
TCP HANDSHAKE
HTTPS HANDSHAKE

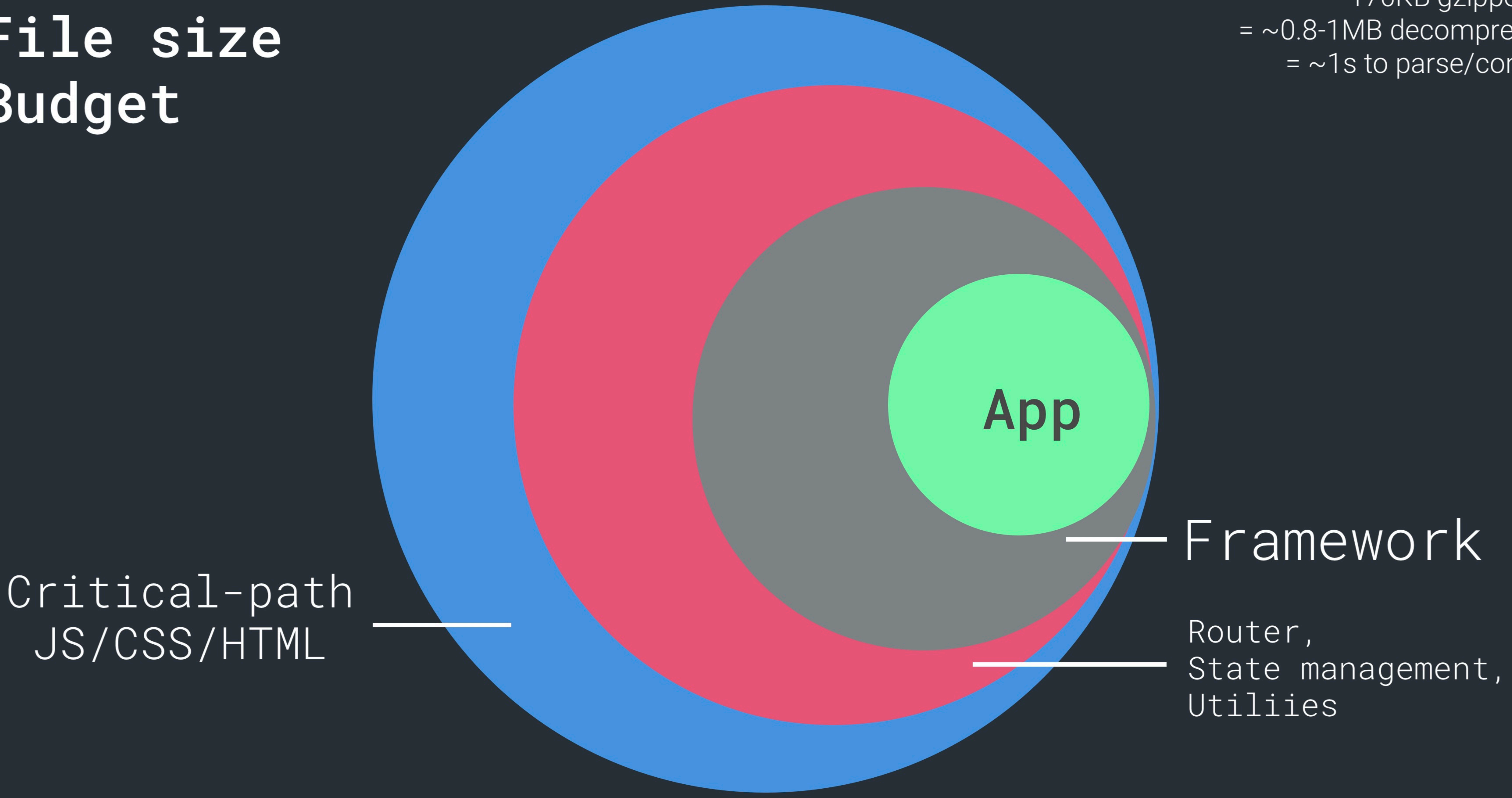
At 400Kbps we can send 3.4 x 50KB = 170KB

Baseline is ~$200 Android phone
On a slow 3G network, emulated at:

400ms RTT, 400Kbps transfer

File size Budget

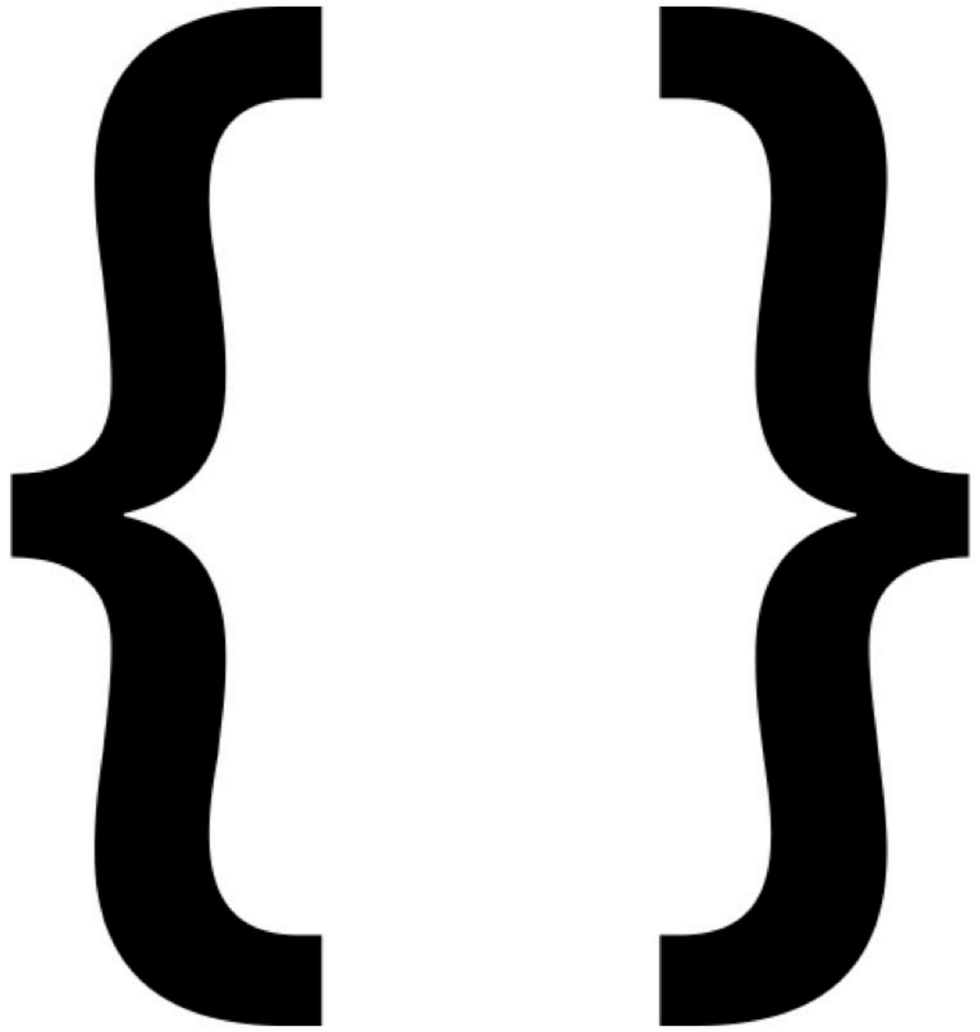
170KB gzipped JS
= ~0.8-1MB decompressed
= ~1s to parse/compile



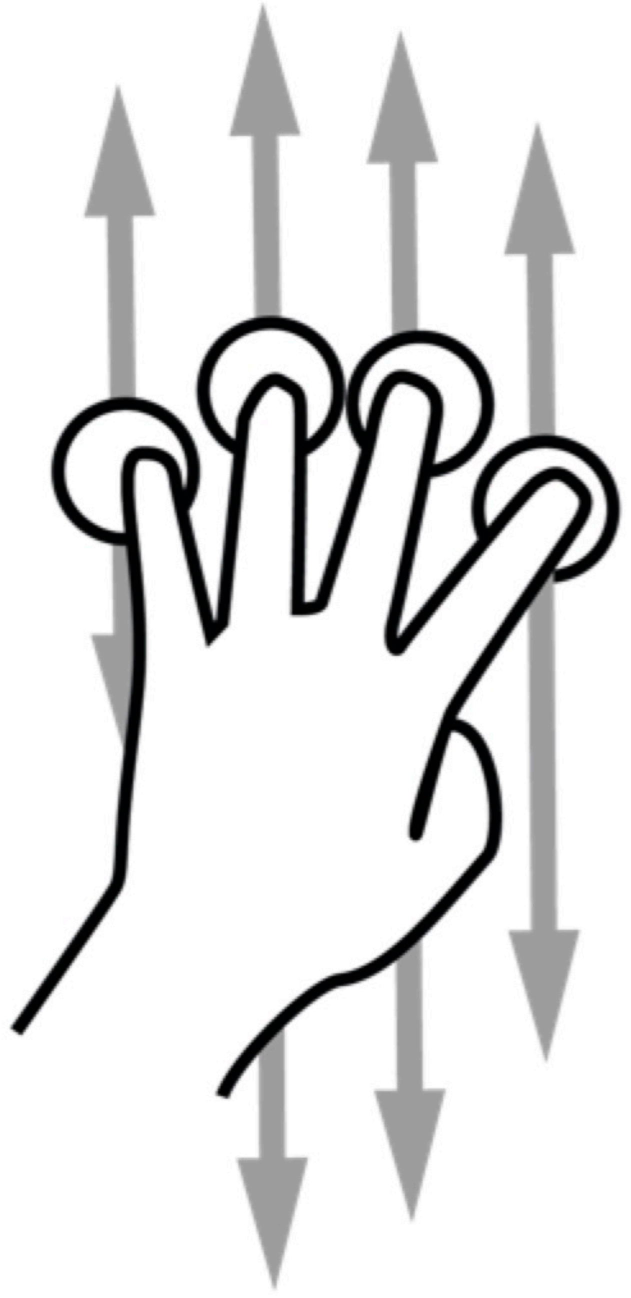
170KB

Evaluating the performance of Web Frameworks

NETWORK TRANSFER



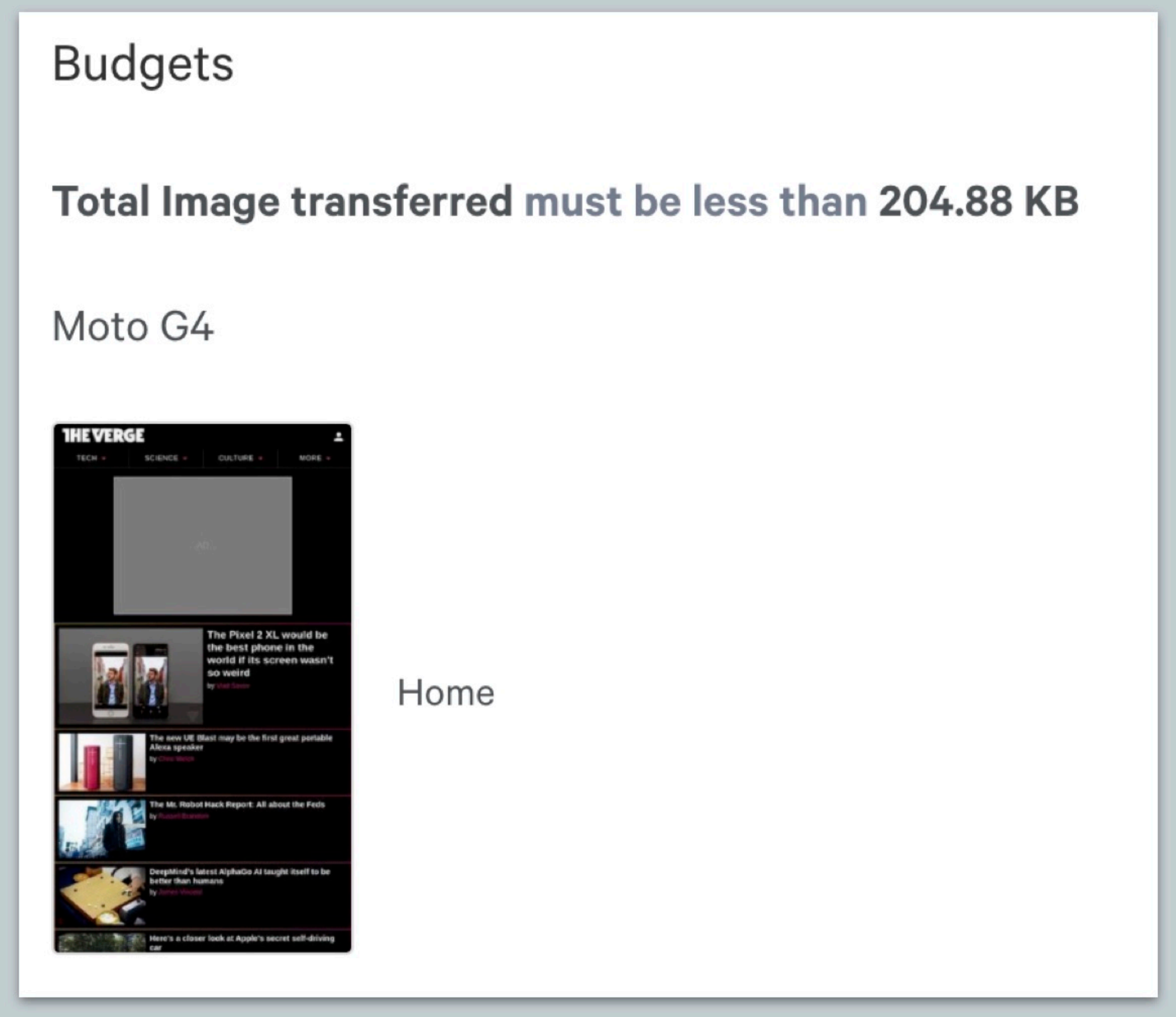
PARSE/COMPILE



RUNTIME COST

https://twitter.com/kristoferbaxter/status/908144931125858304

Performance Budget Tools

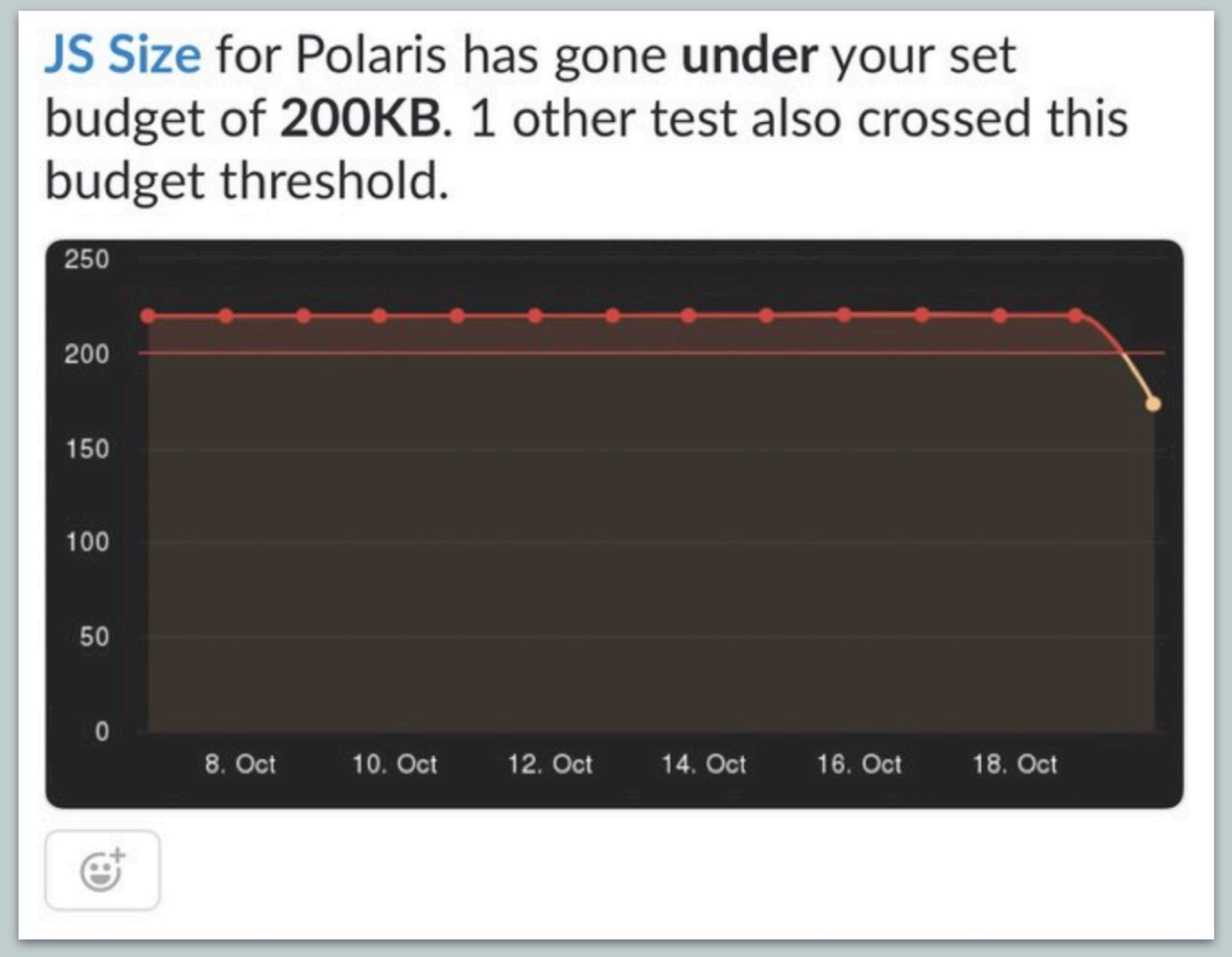


Budgets

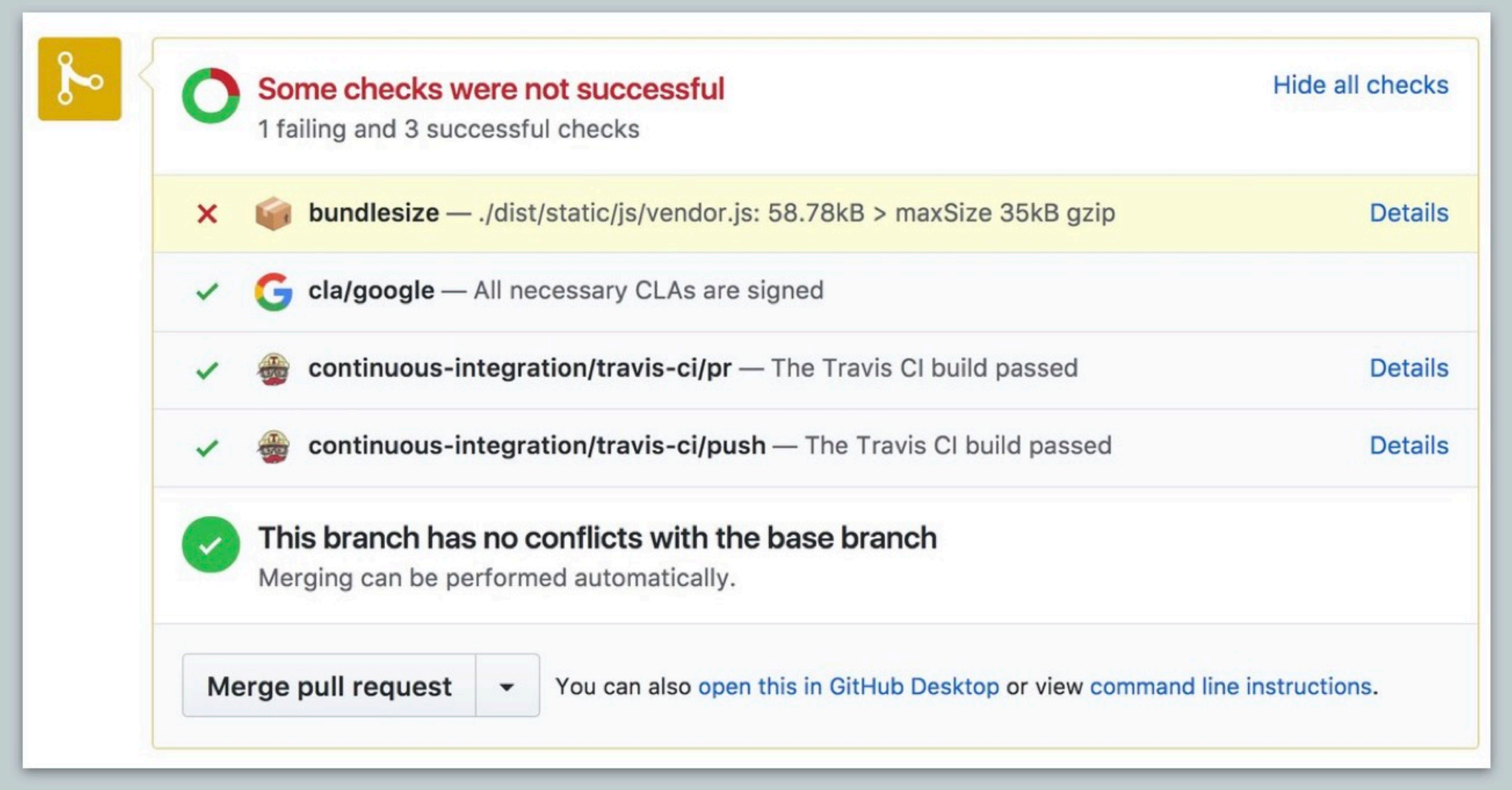
Total Image transferred must be less than **204.88 KB**

Moto G4

Home



JS Size for Polaris has gone **under** your set budget of **200KB**. 1 other test also crossed this budget threshold.



Some checks were not successful

1 failing and 3 successful checks

Hide all checks

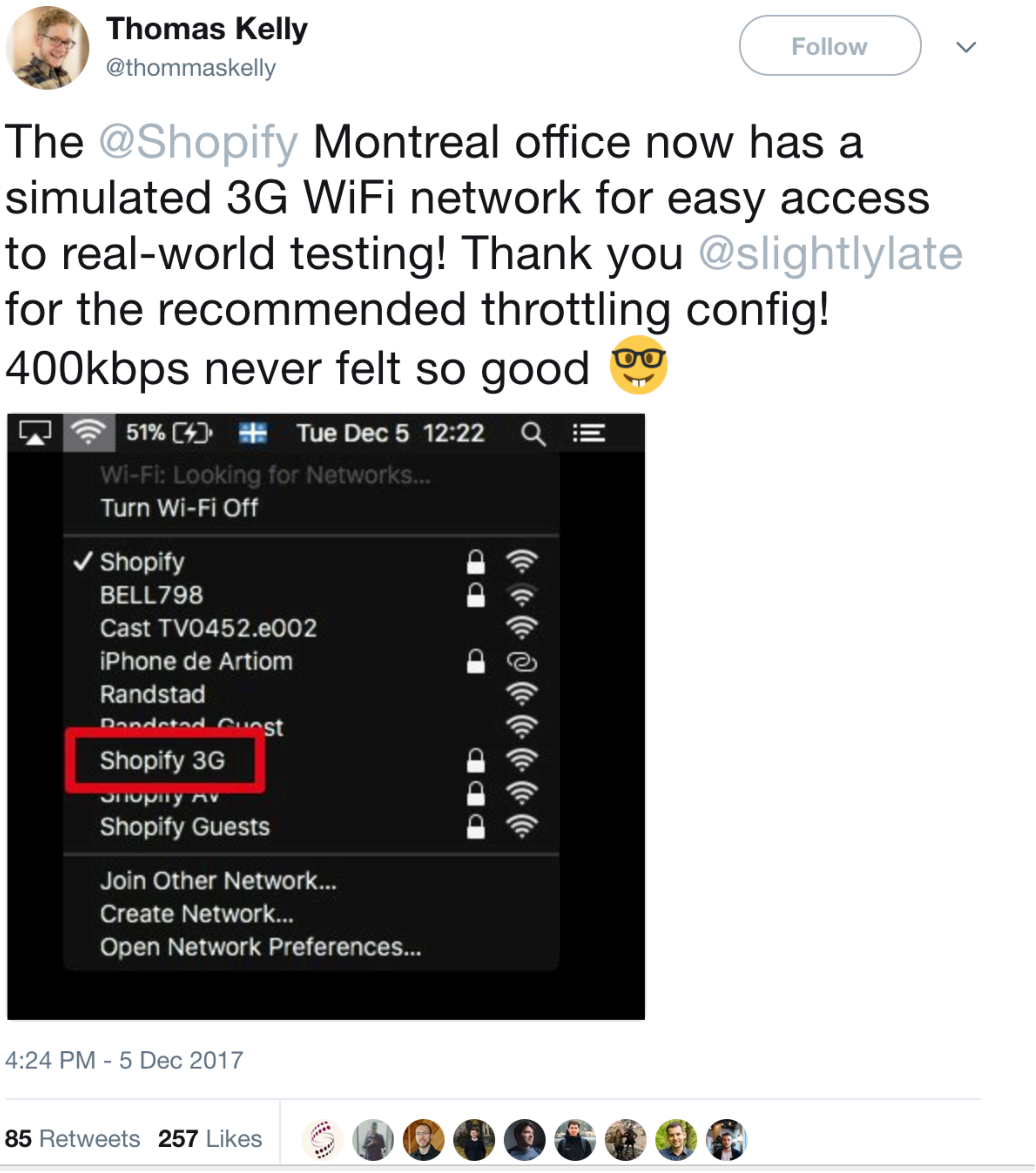
- ✕ **bundlesize** — ./dist/static/js/vendor.js: 58.78kB > maxSize 35kB gzip — Details
- ✓ **cla/google** — All necessary CLAs are signed
- ✓ **continuous-integration/travis-ci/pr** — The Travis CI build passed — Details
- ✓ **continuous-integration/travis-ci/push** — The Travis CI build passed — Details

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request ▾ You can also open this in GitHub Desktop or view command line instructions.

CALIBRE **SPEEDCURVE** **BUNDLESIZE**



Thomas Kelly
@thommaskelly

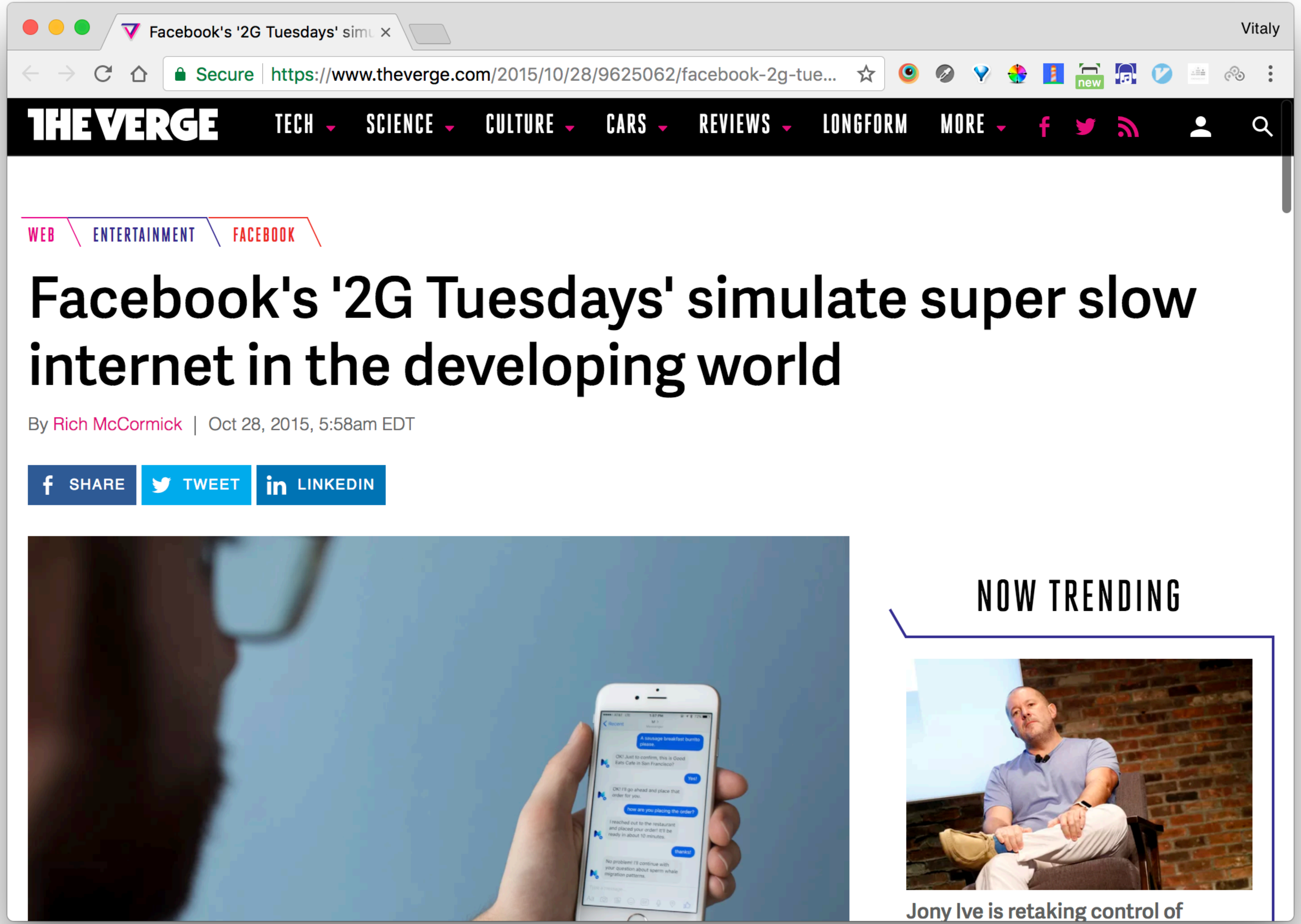
Follow

The @Shopify Montreal office now has a simulated 3G WiFi network for easy access to real-world testing! Thank you @slightlylate for the recommended throttling config! 400kbps never felt so good 🤓

4:24 PM - 5 Dec 2017

85 Retweets 257 Likes

TUES
2G



Facebook's '2G Tuesdays' simu ×
Vitaly
Secure
https://www.theverge.com/2015/10/28/9625062/facebook-2g-tue...
THE VERGE
TECH
SCIENCE
CULTURE
CARS
REVIEWS
LONGFORM
MORE

WEB ENTERTAINMENT FACEBOOK

Facebook's '2G Tuesdays' simulate super slow internet in the developing world

By Rich McCormick | Oct 28, 2015, 5:58am EDT

SHARE
TWEET
LINKEDIN
NOW TRENDING

Jony Ive is retaking control of

HTTP/1.1 Deployment Strategy

- Ultimately, we want to *deliver content fast* — and we don’t want users to re-download assets too frequently.

- A common deployment strategy is quite *simple:*

  — Deploy changes with *unique* file names to *invalidate cache,*

  — *Inline the scripts* directly in HTML to avoid HTTP requests,

  — *Load versioned scripts* using `<script>` tags in HTML,

  — Alternatively, serve *all assets bundled into one versioned file*.

- A better way is to use a *“scout” approach.*

HTTP/1.1 Deployment Strategy

- A better way is to use a *“scout” approach.*
  - JavaScript `(scout.js)`:

```
module.exports = {
    baseUrl : 'https://mysite.com/static/',
    resources : {
        // 1–2 referenced files, have long cache times
        vendor : ‘vendor-d41d8cd98f.js’,
        application : 'application-a32e3ec23d.js'
    }
};
```

- JavaScript `(scout.js)`:

```
module.exports = {
    baseUrl : 'https://mysite.com/static/',
    resources : {
        // 1–2 referenced files, have long cache times
        vendor : ‘vendor-d41d8cd98f.js’,
        application : 'application-a32e3ec23d.js'
    }
};
```

- *scout.js* exists to keep assets highly *cacheable* and prompt changes to these assets to *take effect quickly*.
- Hence, the *scout file* (or HTML) needs a *short cache time.* Assets updated → scout eventually triggers an update.

> Why a *“scout”* approach instead of loading the versioned files using `<script>` tags directly in HTML?
> You can deploy changes in CSS and JavaScript without requiring a re-deploy of all HTML pages, so there is *no need to re-download HTML!*
>
> — *Rebecca Murphy*

HTTP/1.1 Deployment Strategy

- *"Scout" approach works well* and is widely spread, but it raises issues regarding *first visit* vs. repeat visits:

  — Should the scout load *many* small files or *few* large files?

  — *First visitors* have a slow experience due to HTTP requests,

  — *Repeat users* request large files with every minor change,

  — Even if *nothing* changes, repeat users request the scout.

- With *HTTP/2,* HTTP-requests are *cheap.* Use the "scout" approach with *many small files!*

> “*Packaging (still) matters* because there are issues with sending many small JavaScript files to the browser. First, the *compression* of a large package could benefit from dictionary reuse, whereas small separate packages will not.
>
> — *Yoav Weiss*
>
> *https://aerotwist.com/blog/when-everything-is-important-nothing-is/*

> “Secondly, browsers *have not yet been optimized* for such workflows. For example, Chrome will trigger inter-process communications (IPCs) linear to the number of resources, so including hundreds of resources will have *browser runtime costs.*
>
> — *Yoav Weiss*
>
> *https://aerotwist.com/blog/when-everything-is-important-nothing-is/*

HTTP/1.1 Deployment Strategy

- *“Scout” approach works well* and is widely spread, but it raises issues regarding *first visit* vs. repeat visits:
  - — Should the scout load *many* small files or *few* large files?
  - — *First visitors* have a slow experience due to HTTP requests,
  - — *Repeat users* request large files with every minor change,
  - — Even if *nothing* changes, repeat users request the scout.
- With *HTTP/2,* HTTP-requests are *cheap.* Use the “scout” approach with *many small files!*

HTTP/1.1 Deployment Strategy

- *“Scout” approach works well* and is widely spread, but it raises issues regarding *first visit* vs. repeat visits:

  — Should the scout load *many* small files or *few* large files?

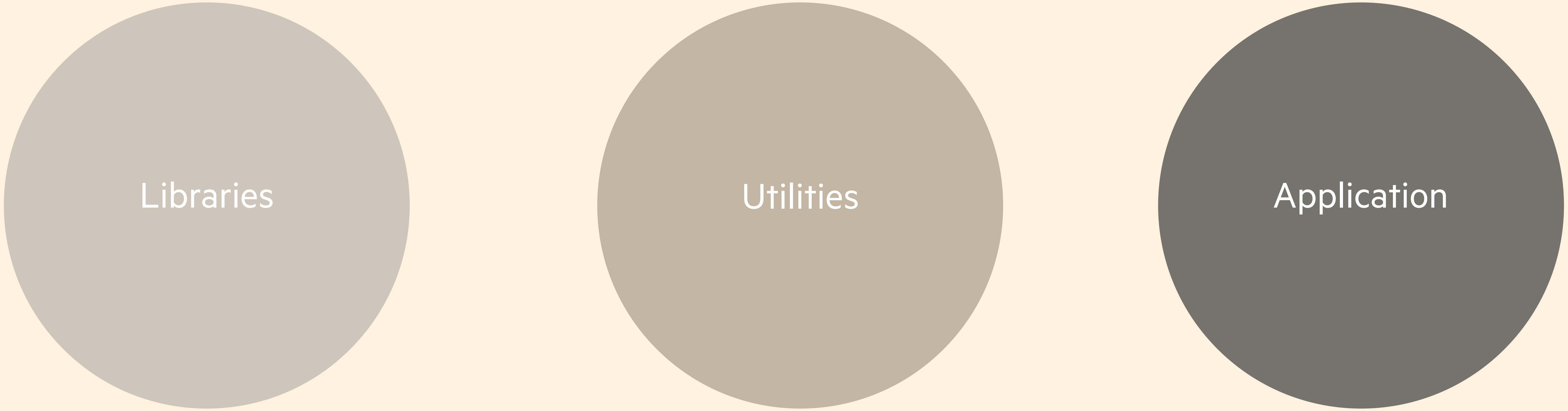
  — *First visitors* have a slow experience due to HTTP requests,

  — *Repeat users* request large files with every minor change,

  — Even if *nothing* changes, repeat users request the scout.

- With *HTTP/2,* HTTP-requests are *cheap.* Use the “scout” approach with *max. 10 packages.*

Bundle by frequency of change!

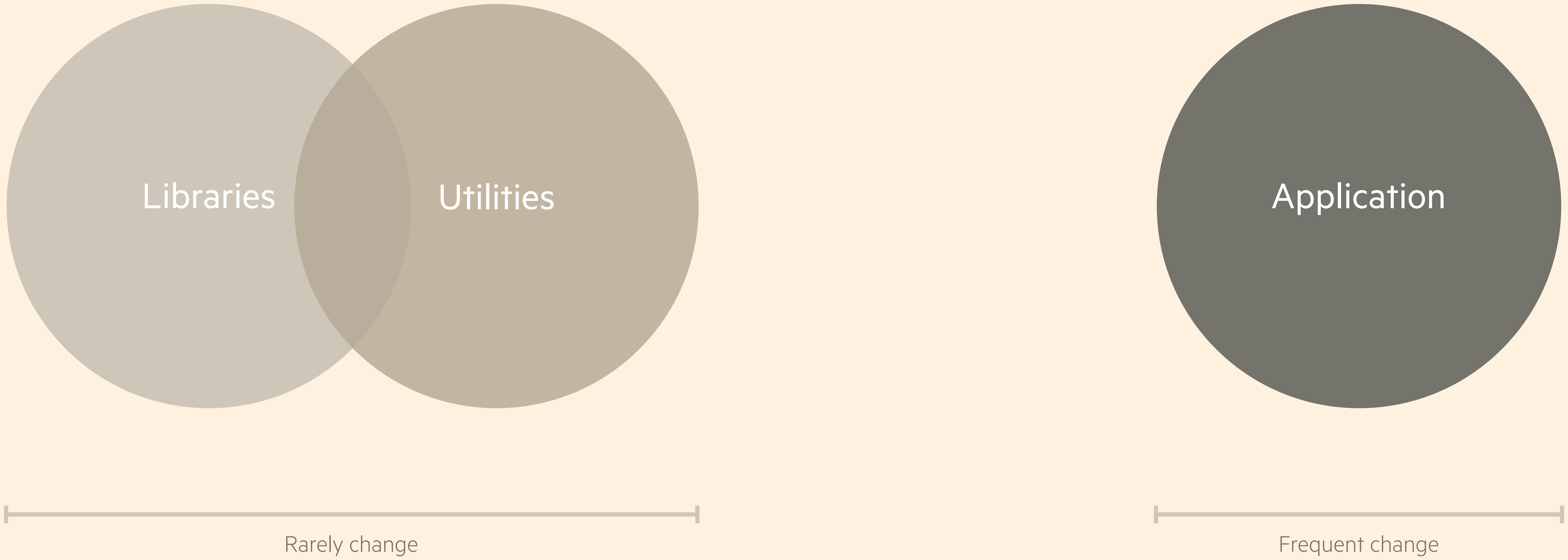


Libraries

Utilities

Application

Bundle by frequency of change!



Bundle by frequency of change!

core.js

app.js

Rarely change

Frequent change

HTTP/2 Deployment Strategy

- With *HTTP/2,* HTTP-requests are *cheap.* Use the “scout” approach with *many small files!*

- *Inlining critical CSS* is an overhead in HTTP/2 world. *Server push* is helpful but slow. *Load CSS in series.*

HTTP/2 Deployment Strategy

- *Inlining critical CSS* is an overhead in HTTP/2 world. *Server push* is helpful but slow. *Load CSS in series.*
- We’ve moved away from…

```
<head>
    <link rel="stylesheet" href="combined.min.css">
</head>
<body>
    …content…
</body>
```

HTTP/2 Deployment Strategy

- *Inlining critical CSS* is an overhead in HTTP/2 world. *Server push* is helpful but slow. *Load CSS in series.*

- ...towards:

```
<head>
   <style> /* Critical CSS styles */ </style>
    <link rel="preload" href=“full.css" as="style"
          onload=“this.rel=‘stylesheet'">
    <noscript><link rel="stylesheet" href="full.css"></noscript>
    <script>
        /*! loadCSS. [c] 2017 Filament Group, Inc. MIT License */
        (function(){ … }());
    </script>
</head>
```

HTTP/2 Deployment Strategy

- ...or alternatively:

```
<head>
<!-- #if expr="$HTTP_COOKIE=/fullcss\=true/" -->
    <link rel="stylesheet" href="full.css">
<!-- #else -->
    <style>
        /* Critical CSS styles, plus: */
        article, .comments, aside, footer { display: none; }
    </style>
    <script>
        loadCSS("full.css"); /* or rest.css + critical.css */
    </script>
    <noscript><link rel="stylesheet" href="full.css"></noscript>
<!-- #endif -->
</head>
```

HTTP/2 Deployment Strategy

- A simple, “recommended” “HTTP/2” way:

```
<head>
    <link rel="stylesheet" href="site-header.css">
    <link rel="stylesheet" href="article.css">
    <link rel="stylesheet" href="comments.css">
    <link rel="stylesheet" href="sidebar.css">
    <link rel="stylesheet" href="site-footer.css">
</head>
<body>
    …content…
</body>
```

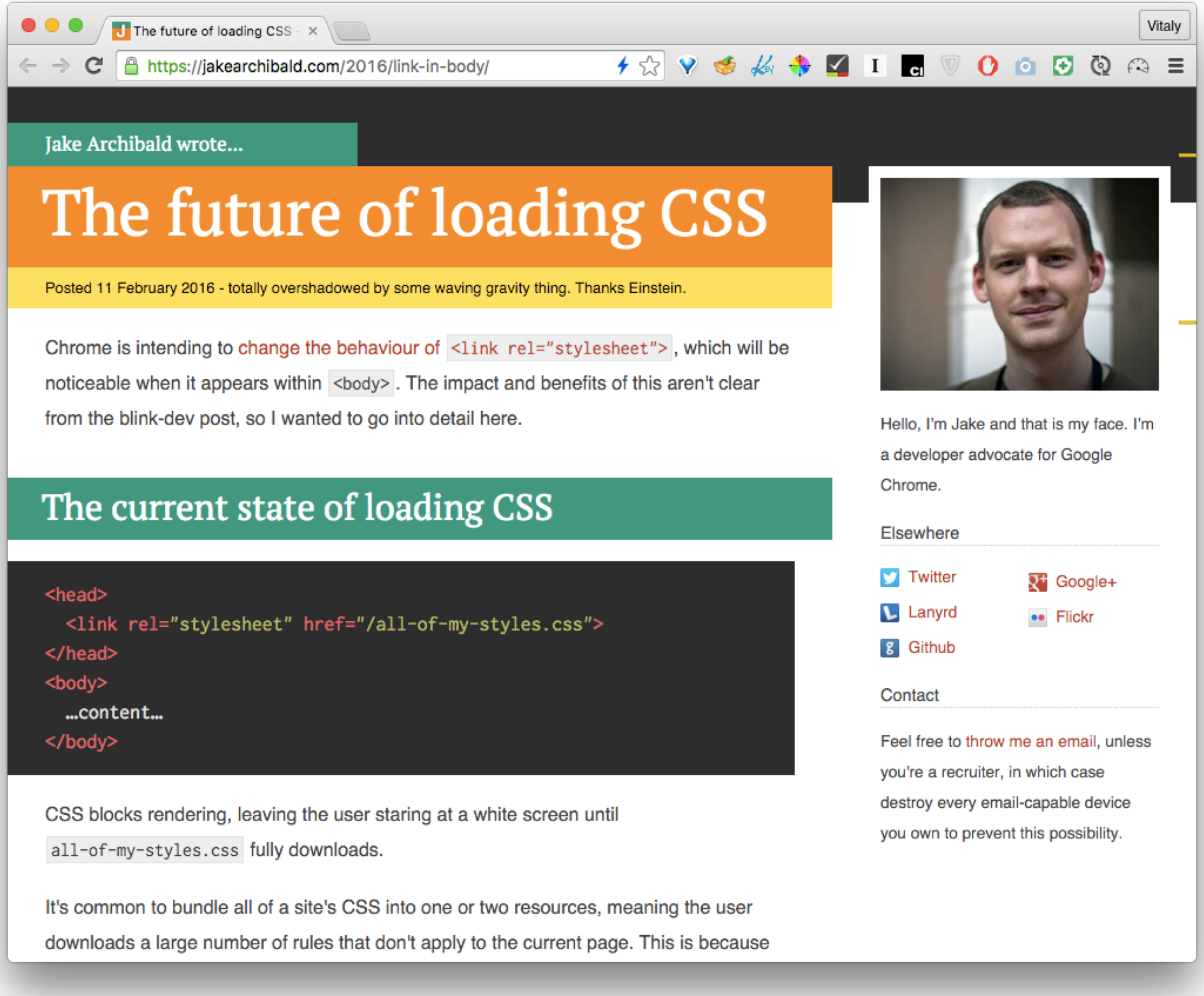
HTTP/2 Deployment Strategy

- But “progressive CSS” way is even better:

```
<head>…</head>
<body>
    <!-- HTTP/2 push critical or inline; whatever faster -->
    <link rel="stylesheet" href="site-header.css">
    <header>…</header>

    <link rel="stylesheet" href="article.css">
    <main>…</main>

    <link rel="stylesheet" href="comments.css">
    <section class="comments">…</section>
    …content…
</body>
```

HTTP/2 Deployment Strategy

- *Multi-Stage CSS loading* removes the need for critical CSS, and provides *sequential* rendering.
- *Browser behavior* supports the technique; browsers block rendering when necessary.
- *Render dependency tree* with CSS custom properties can control how <link>'s are injected.


The future of loading CSS
Vitaly
https://jakearchibald.com/2016/link-in-body/

Jake Archibald wrote...

The future of loading CSS

Posted 11 February 2016 - totally overshadowed by some waving gravity thing. Thanks Einstein.

Chrome is intending to change the behaviour of `<link rel="stylesheet">`, which will be noticeable when it appears within `<body>`. The impact and benefits of this aren't clear from the blink-dev post, so I wanted to go into detail here.

The current state of loading CSS

```
<head>
  <link rel="stylesheet" href="/all-of-my-styles.css">
</head>
<body>
  …content…
</body>
```

CSS blocks rendering, leaving the user staring at a white screen until `all-of-my-styles.css` fully downloads.

It's common to bundle all of a site's CSS into one or two resources, meaning the user downloads a large number of rules that don't apply to the current page. This is because

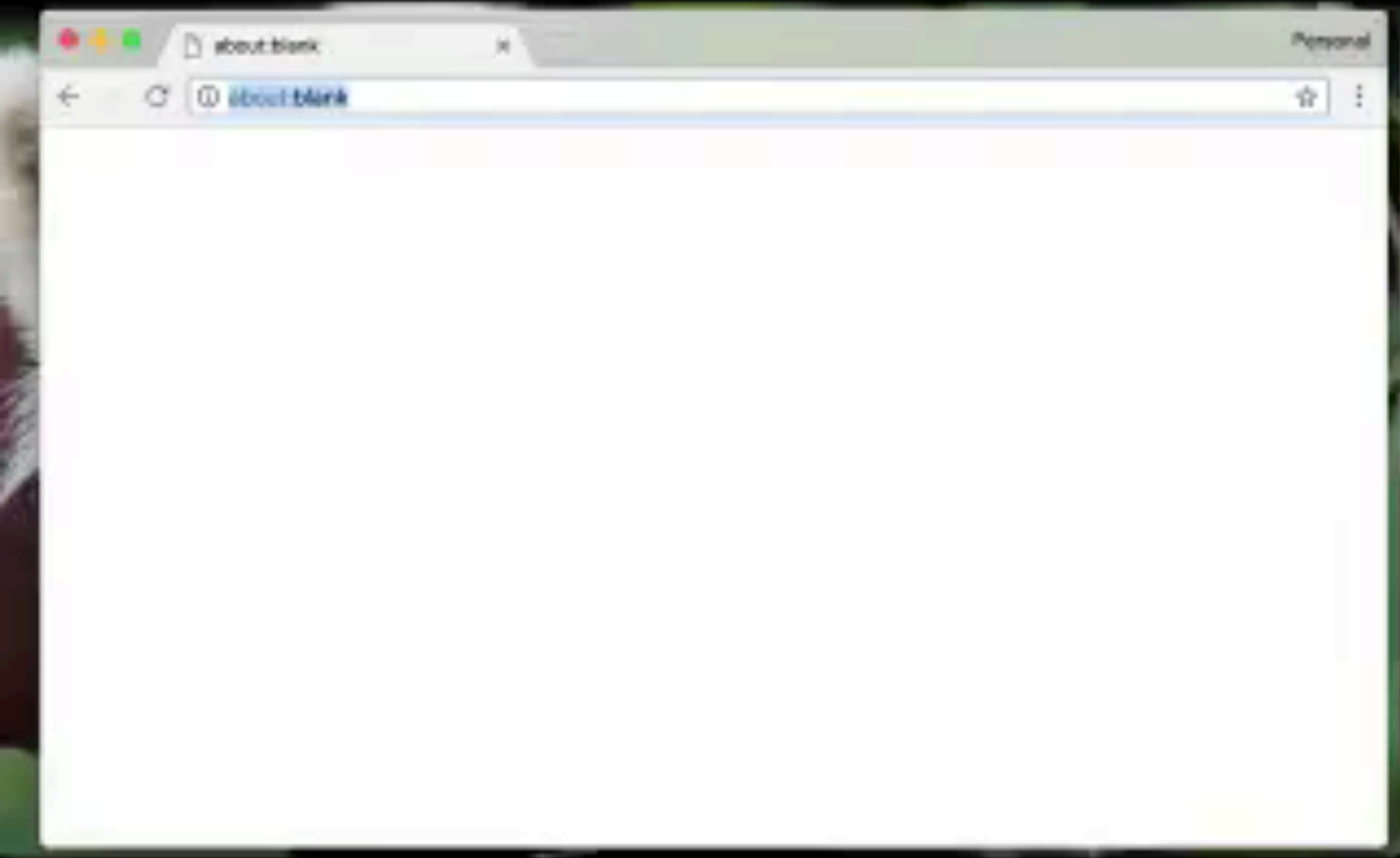
Hello, I'm Jake and that is my face. I'm a developer advocate for Google Chrome.

Elsewhere

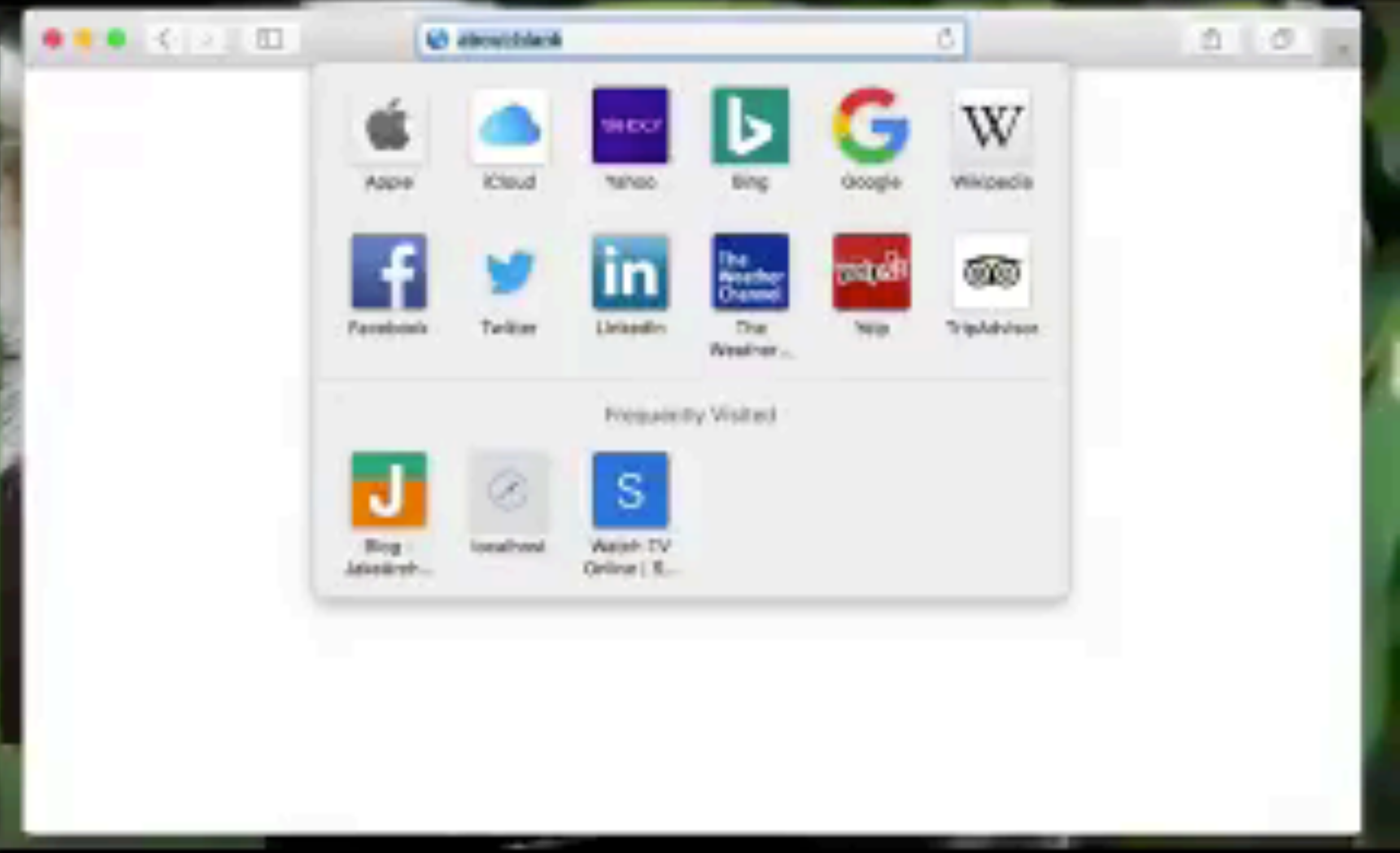
- Twitter
- Google+
- Lanyrd
- Flickr
- Github

Contact

Feel free to throw me an email, unless you're a recruiter, in which case destroy every email-capable device you own to prevent this possibility.



With progressive CSS



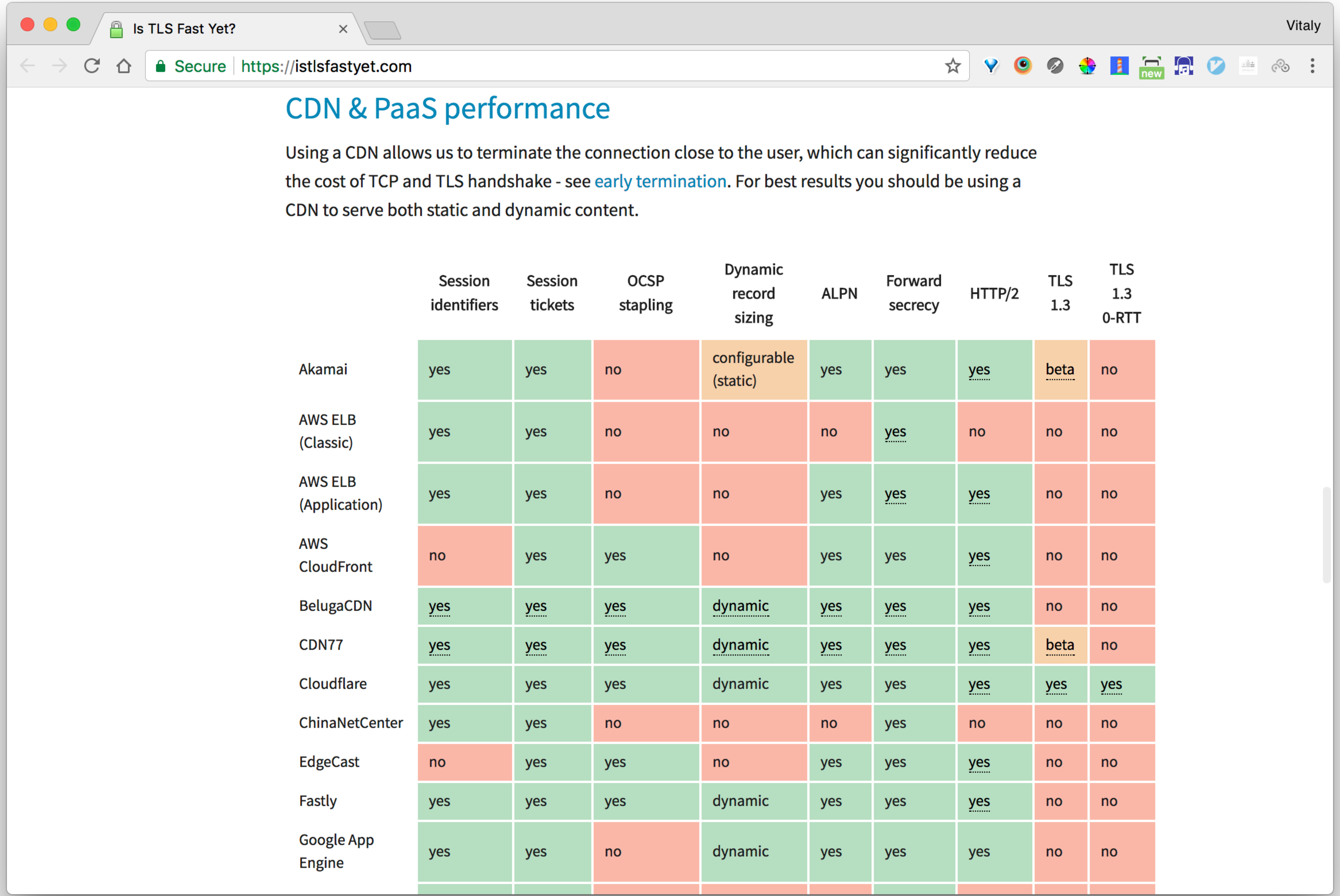
Without

Getting Infrastructure Right

- *Offload static and dynamic* content to CDNs, cache with Service Workers, then enhance with minimal JS.

  — *Articles/comments* are initially stored in Markdown,
  — Markup is generated *once* (table of contents, summary etc.),
  — *Placeholders/skeleton screens* populated into the markup,
  — *All static assets* are served from a service worker / CDN,
  — *Minimal JavaScript modifies* placeholders in the background.

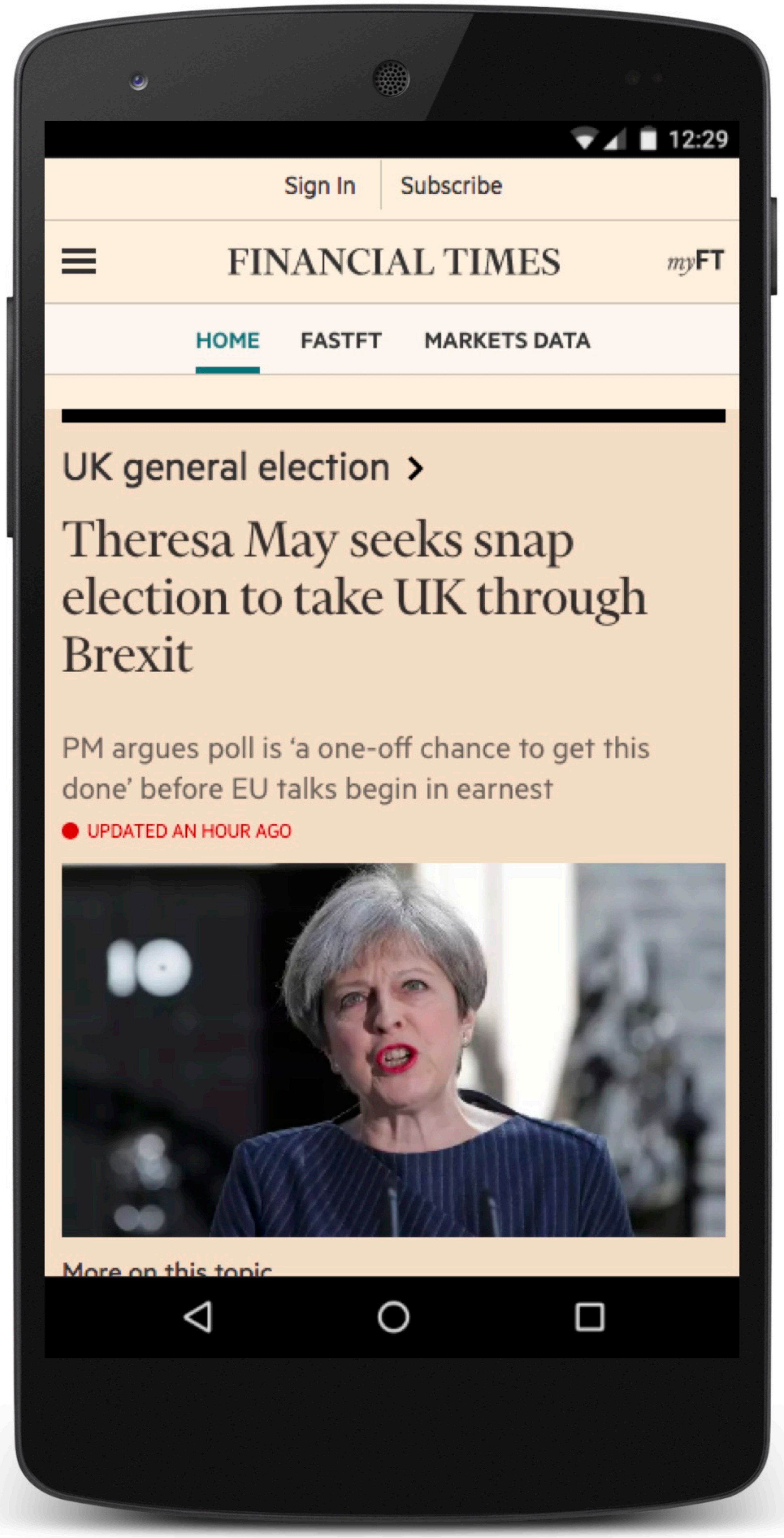
- *Caching:* assets cacheable either for a *very short* time (if they're likely to change) or *indefinitely* (if they're static).



CDN & PaaS performance

Using a CDN allows us to terminate the connection close to the user, which can significantly reduce the cost of TCP and TLS handshake - see early termination. For best results you should be using a CDN to serve both static and dynamic content.

	Session identifiers	Session tickets	OCSP stapling	Dynamic record sizing	ALPN	Forward secrecy	HTTP/2	TLS 1.3	TLS 1.3 0-RTT
Akamai	yes	yes	no	configurable (static)	yes	yes	yes	beta	no
AWS ELB (Classic)	yes	yes	no	no	no	yes	no	no	no
AWS ELB (Application)	yes	yes	no	no	yes	yes	yes	no	no
AWS CloudFront	no	yes	yes	no	yes	yes	yes	no	no
BelugaCDN	yes	yes	yes	dynamic	yes	yes	yes	no	no
CDN77	yes	yes	yes	dynamic	yes	yes	yes	beta	no
Cloudflare	yes	yes	yes	dynamic	yes	yes	yes	yes	yes
ChinaNetCenter	yes	yes	no	no	no	yes	no	no	no
EdgeCast	no	yes	yes	no	yes	yes	yes	no	no
Fastly	yes	yes	yes	dynamic	yes	yes	yes	no	no
Google App Engine	yes	yes	no	dynamic	yes	yes	yes	no	no



UK general election ›

Theresa May seeks snap election to take UK through Brexit

PM argues poll is ‘a one-off chance to get this done’ before EU talks begin in earnest

● UPDATED AN HOUR AGO

More on this topic

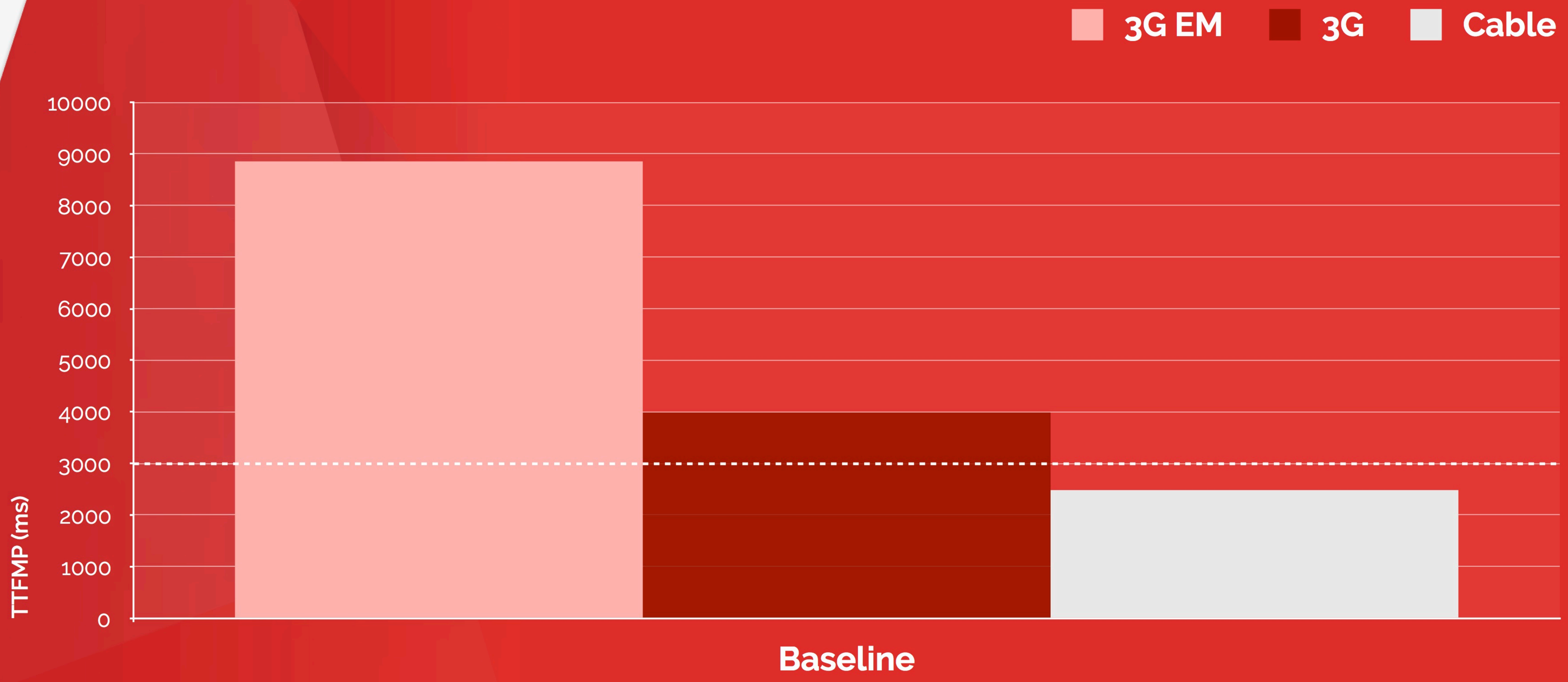
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>FT.com</title>

        <link rel="stylesheet" href="main.css" />

        Other head elements...

    </head>
    <body>

        Content ...

    </body>
</html>
```

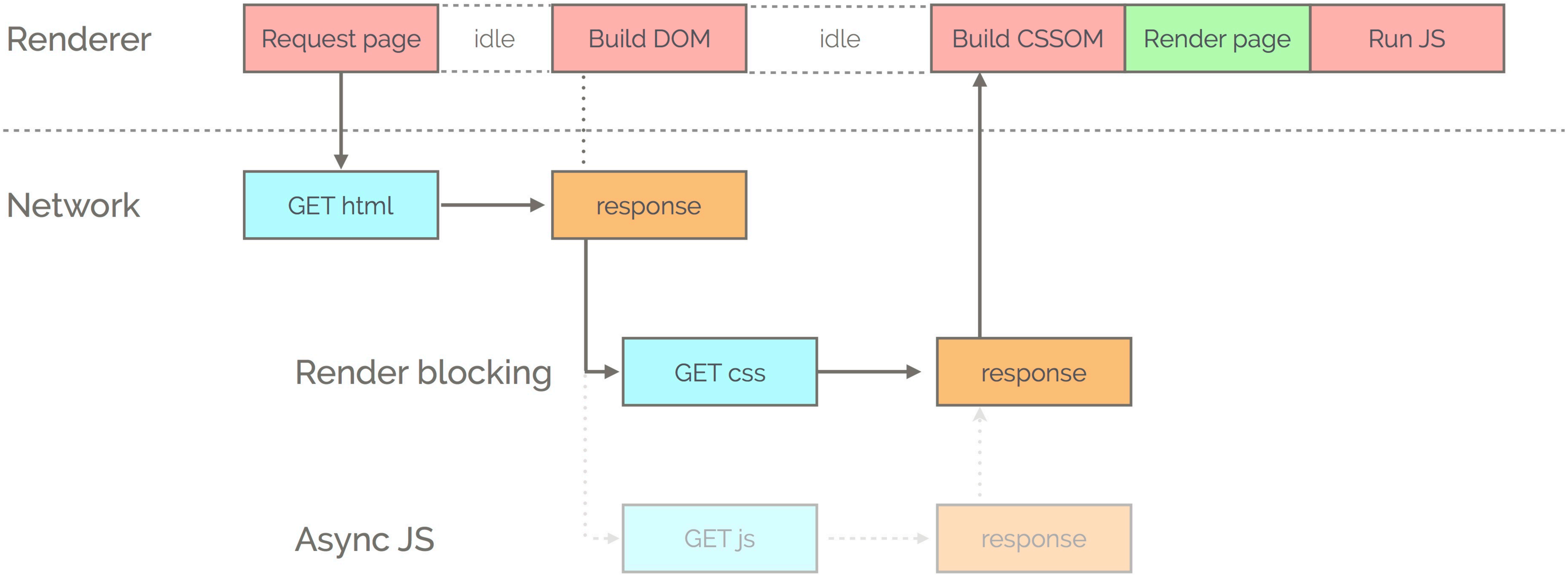
dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
Step_1
1. ttfmp.fastlylabs.com - index.html
2. ttfmp.fastlylabs.com - main.css
3. www.ft.com - h...-markets-data.png
4. www.ft.com - ftlogo:brand-myft
5. www.ft.com - fticon-v1:hamburger
6. www.ft.com - f...brand-ft-masthead
7. www.ft.com - ftlogo:brand-ft
8. www.ft.com - fticon-v1:arrow-right
9. www.ft.com - fticon-v1:arrow-right
10. www.ft.com - fticon-v1:play
11. www.ft.com - fticon-v1:play
12. www.ft.com - fticon-v1:arrow-left
13. www.ft.com - fticon-v1:arrow-up
14. www.ft.com - f...nd-nikkei-tagline
15. www.ft.com - fticon-v1:cross
16. www.ft.com - fticon-v1:search
17. www.ft.com - main-without-n-ui.js
18. next-geebee.ft...- polyfill.min.js
19. www.ft.com - es5.min.js
20. www.ft.com - M...cWeb-Regular.woff
1
2
3
4
5
6
7
8
9
10
11
12
13
14
2658 ms
2255 ms
2851 ms
562 ms
462 ms
499 ms
535 ms
683 ms
897 ms
921 ms
901 ms
664 ms
864 ms
863 ms
790 ms
780 ms
2336 ms
5471 ms
5455 ms
1848 ms

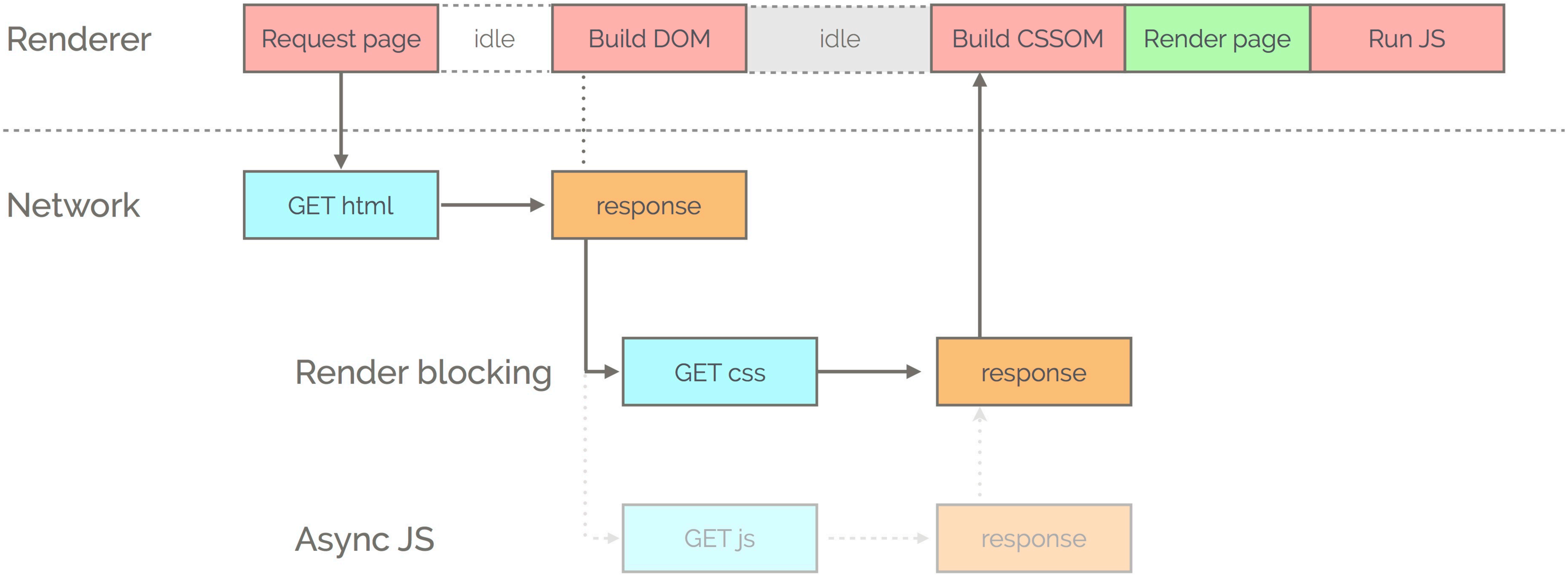
fastly CSS and the first meaningful paint

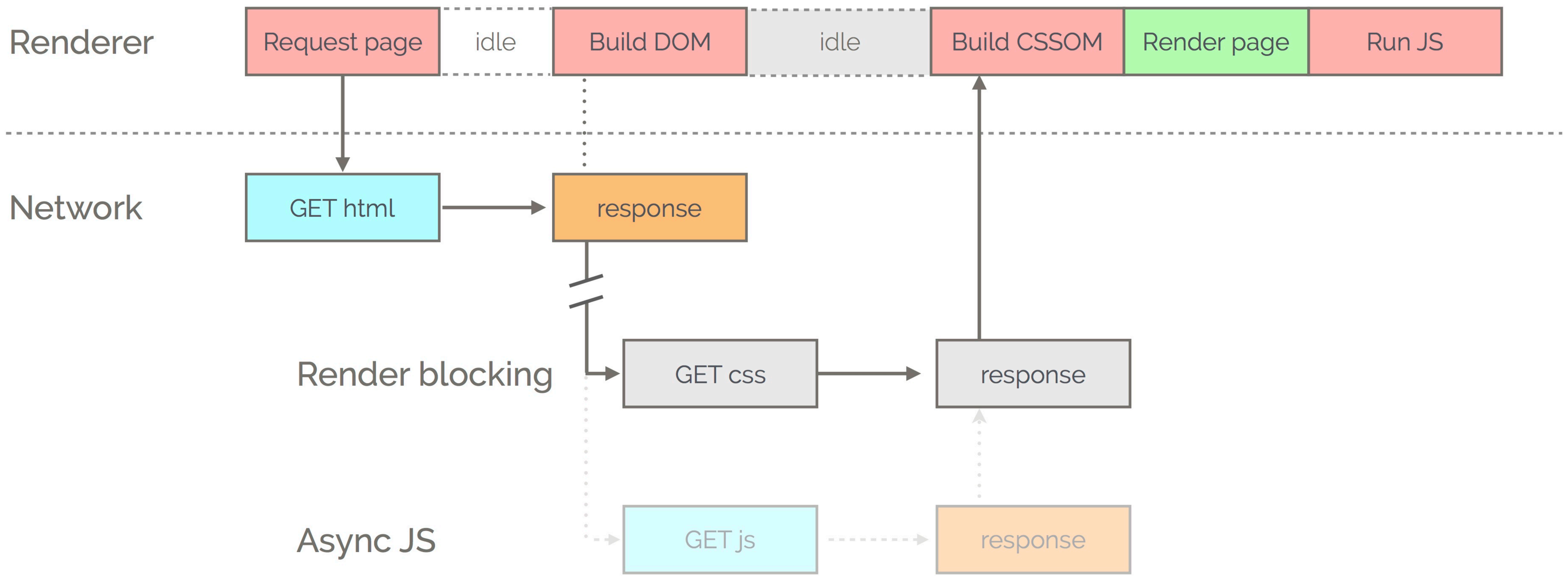
@patrickhamann

Baseline results

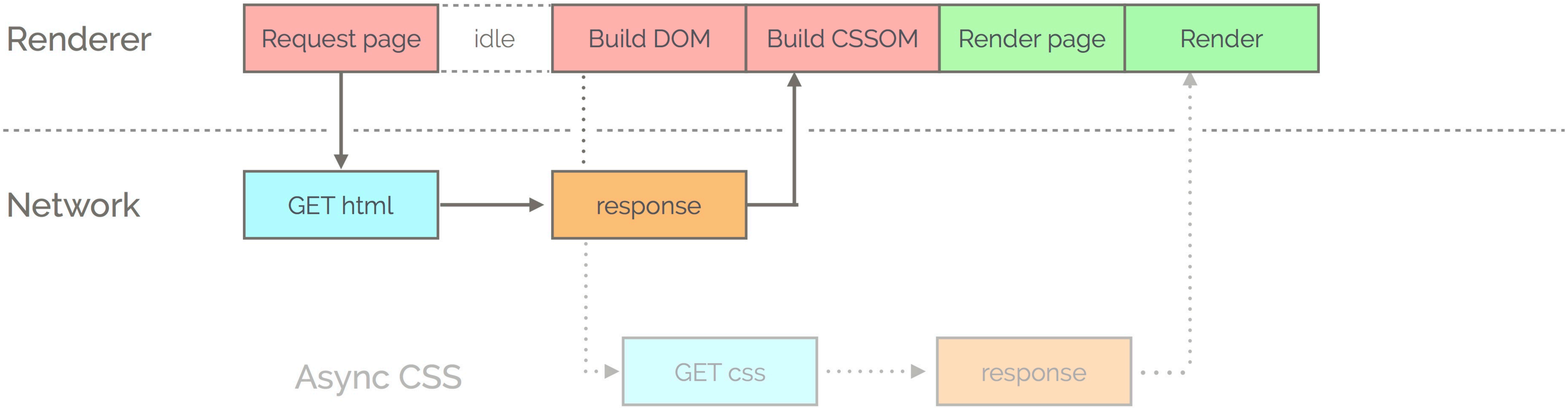



Renderer
Request page
idle
Build DOM
idle
Build CSSOM
Render page
Run JS
Network
GET html
response
Render blocking
GET css
response
Async JS
GET js
response


Renderer
Request page
idle
Build DOM
idle
Build CSSOM
Render page
Run JS
Network
GET html
response
Render blocking
GET css
response
Async JS
GET js
response



Renderer
Request page
idle
Build DOM
idle
Build CSSOM
Render page
Run JS
Network
GET html
response
Render blocking
GET css
response
Async JS
GET js
response


Renderer
Request page
idle
Build DOM
Build CSSOM
Render page
Render
Network
GET html
response
Async CSS
GET css
response

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>FT.com</title>

        <style>
            html{-ms-text-size-adjust:100%;-webkit-text-size-adjust:100%}a{background-color:tran

            Critical styles ...
        </style>

        <link rel="preload" href="main.css" as="style" onload="this.rel='stylesheet'">
        <noscript><link rel="stylesheet" href="main.css"></noscript>
        <script>
            /*! loadCSS. [c]2017 Filament Group, Inc. MIT License */
            (function(){ ... }());
        </script>

        Other head elements...

    </head>
    <body>

```

https://github.com/filamentgroup/loadCSS

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
Step_1
1 2 3 4 5 6 7 8 9 10 11 12 13 14
1. ttfmp.fastlylabs.com - index.html 2658 ms
2. ttfmp.fastlylabs.com - main.css 2255 ms
3. www.ft.com - h...-markets-data.png 2851 ms
4. www.ft.com - ftlogo:brand-myft 562 ms
5. www.ft.com - fticon-v1:hamburger 462 ms
6. www.ft.com - f...brand-ft-masthead 499 ms
7. www.ft.com - ftlogo:brand-ft 535 ms
8. www.ft.com - fticon-v1:arrow-right 683 ms
9. www.ft.com - fticon-v1:arrow-right 897 ms
10. www.ft.com - fticon-v1:play 921 ms
11. www.ft.com - fticon-v1:play 901 ms
12. www.ft.com - fticon-v1:arrow-left 664 ms
13. www.ft.com - fticon-v1:arrow-up 864 ms
14. www.ft.com - f...nd-nikkei-tagline 863 ms
15. www.ft.com - fticon-v1:cross 790 ms
16. www.ft.com - fticon-v1:search 780 ms
17. www.ft.com - main-without-n-ui.js 2336 ms
18. next-geebee.ft...- polyfill.min.js 5471 ms
19. www.ft.com - es5.min.js 5455 ms
20. www.ft.com - M...cWeb-Regular.woff 1848 ms

Before

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
Step_1
1 2 3 4 5 6 7 8 9 10 11 12 13 14
1. ttfmp.fastlylabs.com - index.html 2658 ms
2. ttfmp.fastlylabs.com - main.css 2255 ms
3. www.ft.com - h...-markets-data.png 2851 ms
4. www.ft.com - ftlogo:brand-myft 562 ms
5. www.ft.com - fticon-v1:hamburger 462 ms
6. www.ft.com - f...brand-ft-masthead 499 ms
7. www.ft.com - ftlogo:brand-ft 535 ms
8. www.ft.com - fticon-v1:arrow-right 683 ms
9. www.ft.com - fticon-v1:arrow-right 897 ms
10. www.ft.com - fticon-v1:play 921 ms
11. www.ft.com - fticon-v1:play 901 ms
12. www.ft.com - fticon-v1:arrow-left 664 ms
13. www.ft.com - fticon-v1:arrow-up 864 ms
14. www.ft.com - f...nd-nikkei-tagline 863 ms
15. www.ft.com - fticon-v1:cross 790 ms
16. www.ft.com - fticon-v1:search 780 ms
17. www.ft.com - main-without-n-ui.js 2336 ms
18. next-geebee.ft...- polyfill.min.js 5471 ms
19. www.ft.com - es5.min.js 5455 ms
20. www.ft.com - M...cWeb-Regular.woff 1848 ms

Before

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Step_1

1. ttfmp.fastlylabs.com - index.html 2969 ms
2. ttfmp.fastlylabs.com - main.css 2592 ms
3. www.ft.com - ftlogo:brand-myft 645 ms
4. www.ft.com - f...brand-ft-masthead 600 ms
5. www.ft.com - fticon-v1:hamburger 3294 ms
6. next-geebee.ft...- polyfill.min.js 6034 ms
7. www.ft.com - fticon-v1:arrow-right 501 ms
8. www.ft.com - fticon-v1:play 507 ms
9. www.ft.com - es5.min.js 4095 ms
10. www.ft.com - main-without-n-ui.js 5985 ms
11. www.ft.com - F...yWeb-Regular.woff 5565 ms
12. www.ft.com - M...cWeb-Regular.woff 2777 ms
13. www.ft.com - M...Web-Semibold.woff 3392 ms
14. www.ft.com - fticon-v1:arrow-right 3979 ms
15. www.ft.com - f...nd-nikkei-tagline 3979 ms
16. www.ft.com - fticon-v1:cross 3987 ms
17. www.ft.com - fticon-v1:search 3993 ms
18. www.ft.com - ftlogo:brand-ft 3998 ms
19. www.ft.com - fticon-v1:arrow-up 5860 ms
20. www.ft.com - fticon-v1:arrow-left 5852 ms

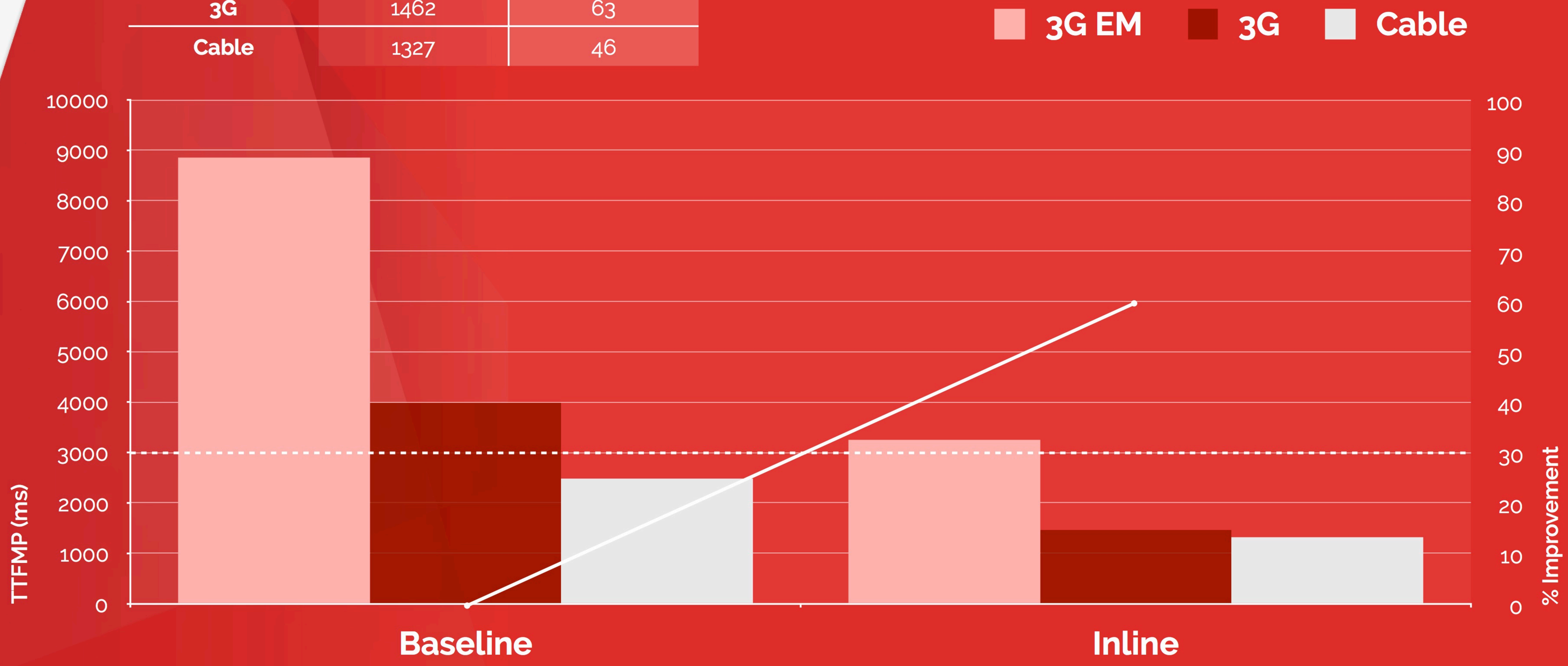
After

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
Step_1
1 2 3 4 5 6 7 8 9 10 11 12 13 14
1. ttfmp.fastlylabs.com - index.html
2969 ms
2. ttfmp.fastlylabs.com - main.css
2692 ms
3. www.ft.com - ftlogo:brand-myft
645 ms
4. www.ft.com - f...brand-ft-masthead
600 ms
5. www.ft.com - fticon-v1:hamburger
3294 ms
6. next-geebee.ft...- polyfill.min.js
6034 ms
7. www.ft.com - fticon-v1:arrow-right
501 ms
8. www.ft.com - fticon-v1:play
507 ms
9. www.ft.com - es5.min.js
4095 ms
10. www.ft.com - main-without-n-ui.js
5985 ms
11. www.ft.com - F...yWeb-Regular.woff
5565 ms
12. www.ft.com - M...cWeb-Regular.woff
2777 ms
13. www.ft.com - M...Web-Semibold.woff
3392 ms
14. www.ft.com - fticon-v1:arrow-right
3979 ms
15. www.ft.com - f...nd-nikkei-tagline
3979 ms
16. www.ft.com - fticon-v1:cross
3987 ms
17. www.ft.com - fticon-v1:search
3993 ms
18. www.ft.com - ftlogo:brand-ft
3998 ms
19. www.ft.com - fticon-v1:arrow-up
5860 ms
20. www.ft.com - fticon-v1:arrow-left
5852 ms
1 2 3 4 5 6 7 8 9 10 11 12 13 14

After

Inline critical CSS results

	TTFMP	% impovement
3G EM	3259	63
3G	1462	63
Cable	1327	46



Pros

- No blocking resources
- No SPOF on CSS
- Eliminates critical request
- Instant painting

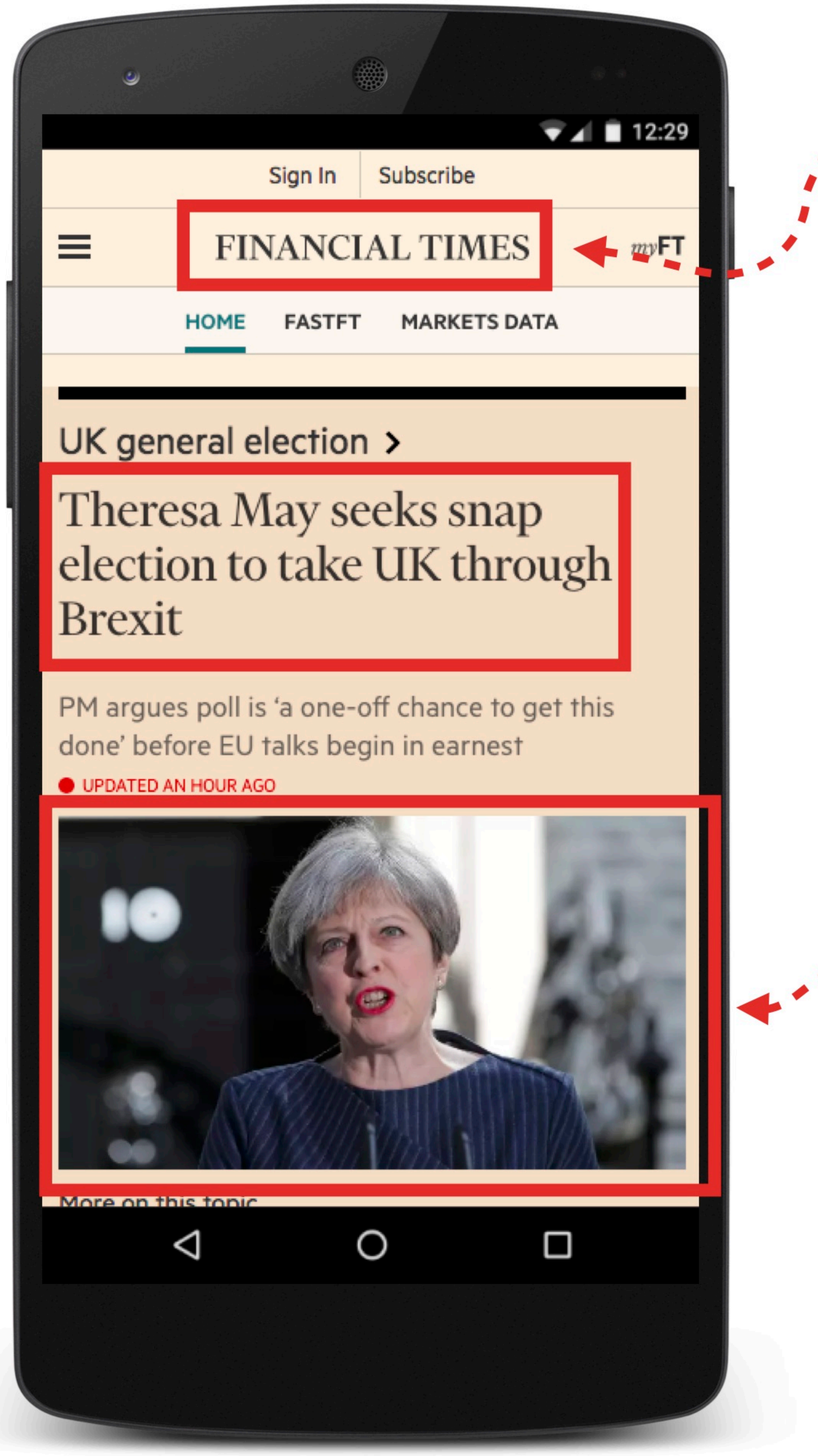
Cons

- Causes reflow
- **Not cachable**
- Hard to maintain
- Hard to automate

Logo?

Fonts?

Hero image?



UK general election ›

Theresa May seeks snap election to take UK through Brexit

PM argues poll is 'a one-off chance to get this done' before EU talks begin in earnest

● UPDATED AN HOUR AGO

More on this topic

Lighthouse
Version: 1.6.0
BETA
Progressive Web App
Best Practices
Performance
Fancier stuff
Export...

Results for: https://www.ft.com/
Generated on: 4/20/2017, 10:20:19 PM GMT+1

Oversized Images ?

Critical Request Chains: **5** ?

Longest chain: **6,181.9ms** over **1** requests, totalling **3.94KB**

▼ View critical network waterfall

Initial navigation

```
/ (www.ft.com)
    …raw/fticon-v1:hamburger (www.ft.com) - 2,790.7ms, 43.12KB
    …raw/ftlogo:brand-myft (www.ft.com) - 2,903.7ms, 44.98KB
    …o-fonts-assets@1.3.0/MetricWeb-Regular.woff (www.ft.com) - 3,969.8ms, 92.08KB
    …o-fonts-assets@1.3.0/MetricWeb-Semibold.woff (www.ft.com) - 4,004.5ms, 93.18KB
    …o-fonts-assets@1.3.0/FinancierDisplayWeb-Regular.woff (www.ft.com) - 4,141.7ms, 107.31KB
…52cc644a/main.css (www.ft.com) - 2,603.6ms, 23.97KB
…v2/polyfill.min.js (next-geebee.ft.com) - 2,982.3ms, 1.04KB
…raw/ftlogo:brand-ft (www.ft.com) - 3,011.2ms, 0.76KB
…backgrounds/header-markets-data.png (www.ft.com) - 3,018.5ms, 0.54KB
…42dac1ba/main-without-n-ui.js (www.ft.com) - 4,012.7ms, 48.24KB
…v5.5.2/es5.min.js (www.ft.com)
    …raw/fticon-v1:arrow-down (www.ft.com) - 4,820.5ms, 71.23KB
…raw/http%3A%2F%2Fpr….com%2Ff3d6f1be-252b-11e7-a34a-538b4cb30025 (www.ft.com) - 5,071.2ms, 5.79KB
…v2/favicon-194x194.png (www.ft.com) - 6,181.9ms, 3.94KB
```

Render-blocking Stylesheets ?

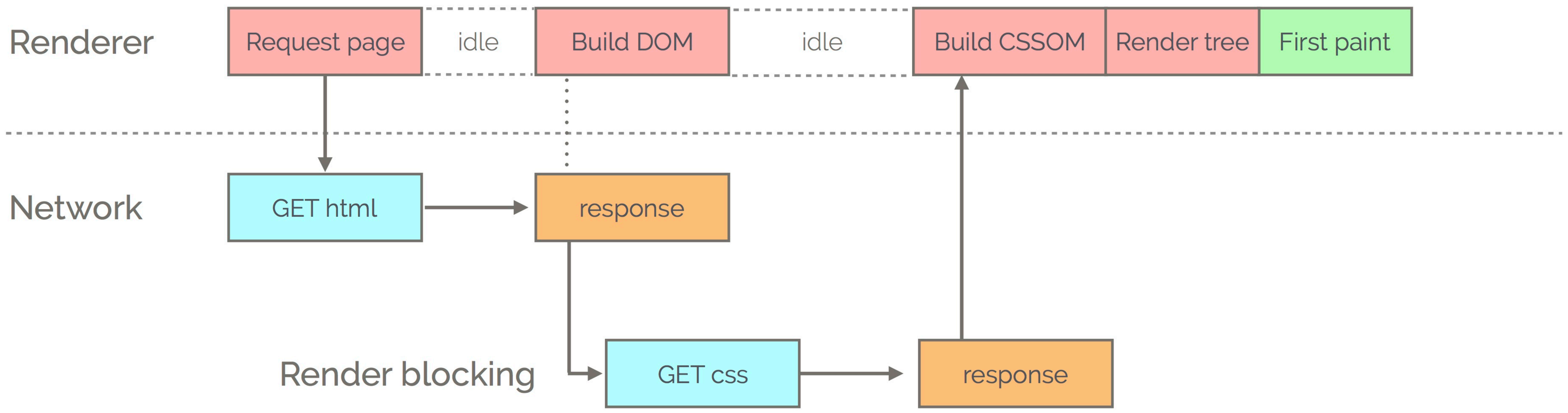
Lighthouse https://developers.google.com/web/tools/lighthouse/

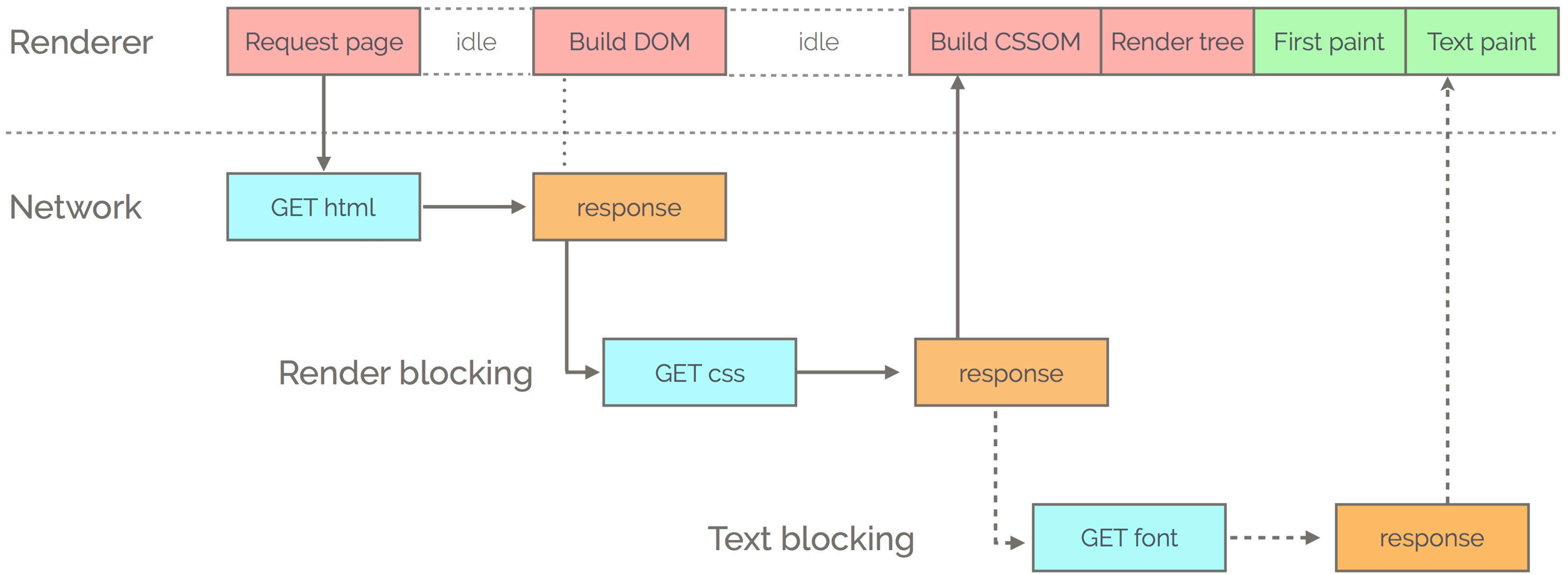
dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

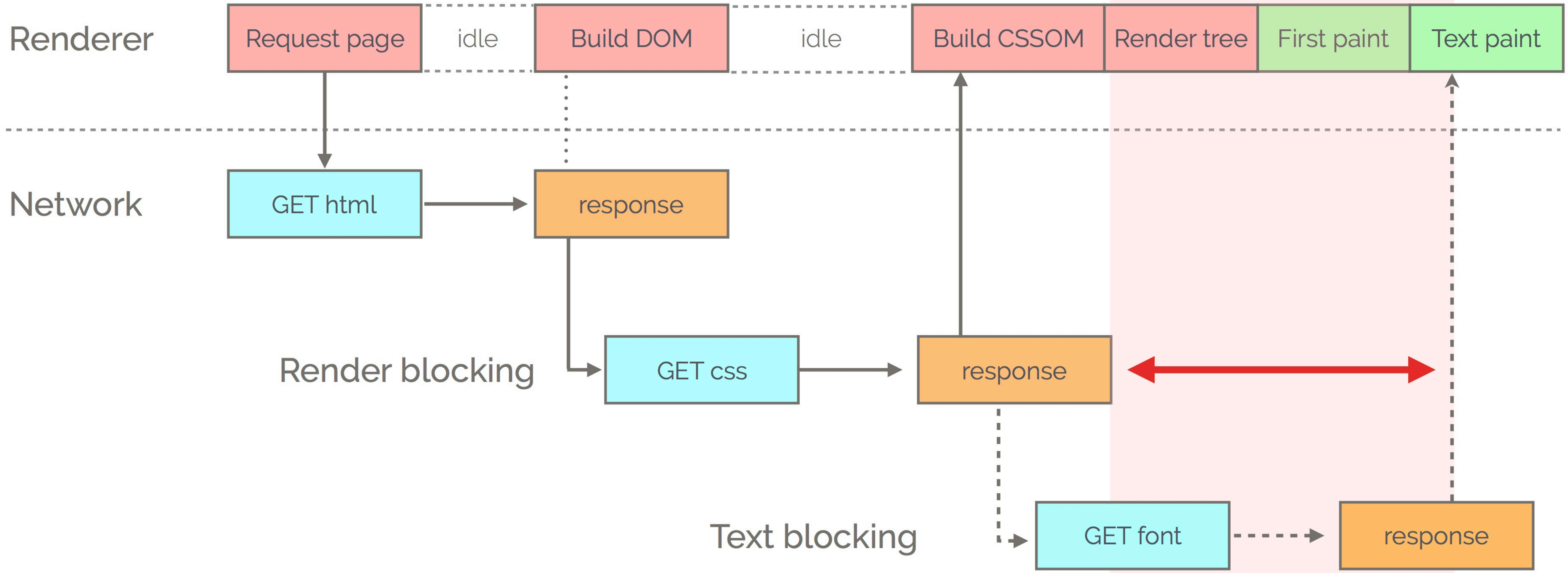
Step_1

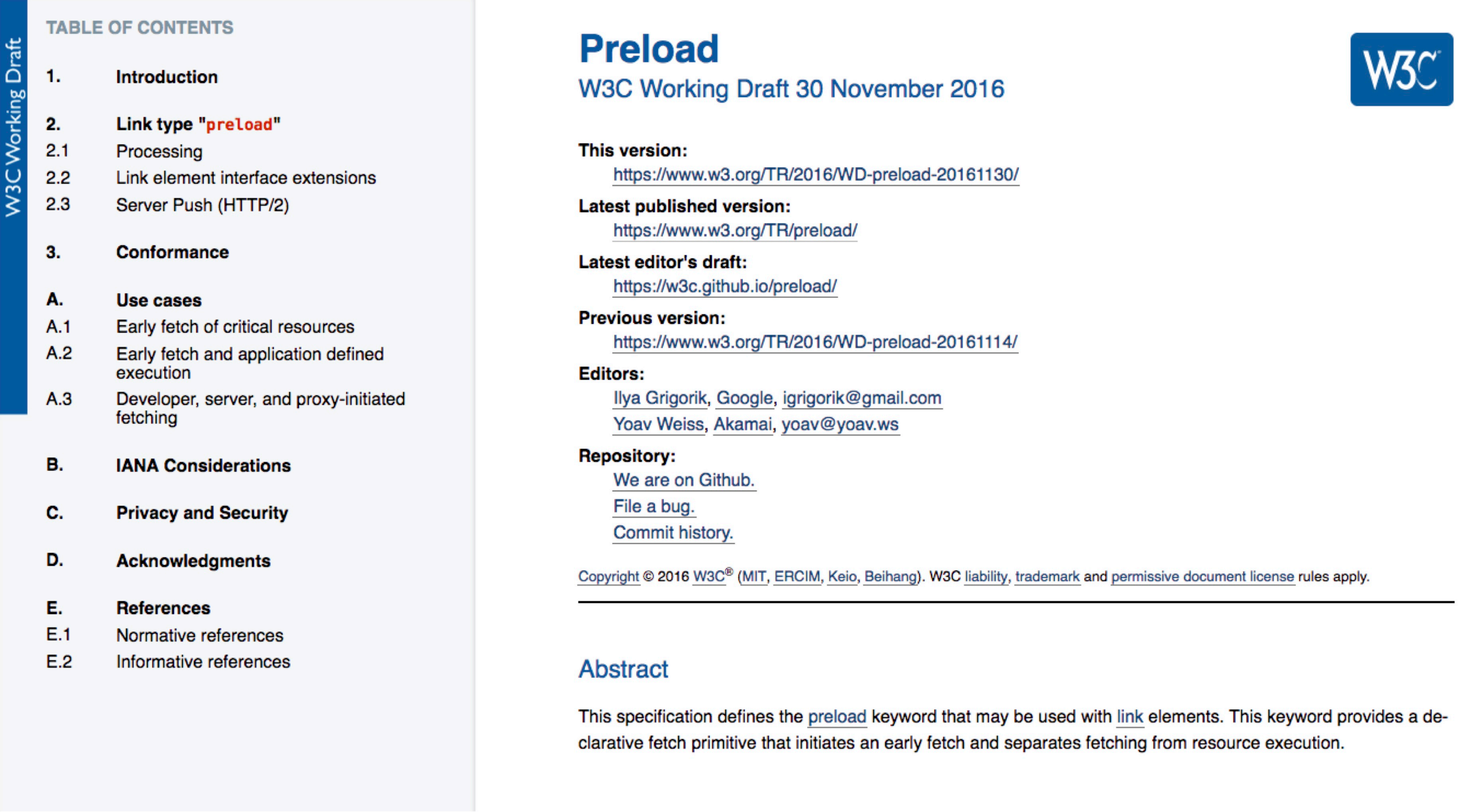
```
 1. ttfmp.fastlylabs.com - index.html            984 ms
 2. ttfmp.fastlylabs.com - main.css              411 ms
 3. www.ft.com - h...-markets-data.png           928 ms
 4. www.ft.com - ftlogo:brand-myft               210 ms
 5. www.ft.com - fticon-v1:hamburger             196 ms
 6. www.ft.com - ftlogo:brand-ft                 215 ms
 7. www.ft.com - f...brand-ft-masthead           200 ms
 8. www.ft.com - fticon-v1:search                274 ms
 9. www.ft.com - fticon-v1:arrow-up              252 ms
10. www.ft.com - f...nd-nikkei-tagline           259 ms
11. www.ft.com - fticon-v1:arrow-left            246 ms
12. www.ft.com - fticon-v1:cross                 268 ms
13. www.ft.com - fticon-v1:arrow-right           215 ms
14. www.ft.com - fticon-v1:play                  240 ms
15. www.ft.com - fticon-v1:play                  234 ms
16. www.ft.com - fticon-v1:arrow-right           228 ms
17. www.ft.com - M...cWeb-Regular.woff           752 ms
18. www.ft.com - F...yWeb-Regular.woff          1522 ms
19. www.ft.com - M...Web-Semibold.woff           931 ms
20. next-geebee.ft... - polyfill.min.js          376 ms
...
```

Before


Renderer
Request page
idle
Build DOM
idle
Build CSSOM
Render tree
First paint
Network
GET html
response
Render blocking
GET css
response


Renderer
Request page
idle
Build DOM
idle
Build CSSOM
Render tree
First paint
Text paint
Network
GET html
response
Render blocking
GET css
response
Text blocking
GET font
response


Renderer
Request page
idle
Build DOM
idle
Build CSSOM
Render tree
First paint
Text paint
Network
GET html
response
Render blocking
GET css
response
Text blocking
GET font
response



Preload

W3C Working Draft 30 November 2016

This version:
https://www.w3.org/TR/2016/WD-preload-20161130/

Latest published version:
https://www.w3.org/TR/preload/

Latest editor's draft:
https://w3c.github.io/preload/

Previous version:
https://www.w3.org/TR/2016/WD-preload-20161114/

Editors:
Ilya Grigorik, Google, igrigorik@gmail.com
Yoav Weiss, Akamai, yoav@yoav.ws

Repository:
We are on Github.
File a bug.
Commit history.

Abstract

This specification defines the preload keyword that may be used with link elements. This keyword provides a declarative fetch primitive that initiates an early fetch and separates fetching from resource execution.

Preload, W3C working draft https://www.w3.org/TR/preload/

Preload with markup:

```
<!-- preload stylesheet resource via declarative markup -->
<link rel="preload" href="/styles/other.css" as="style">

<!-- or, preload stylesheet resource via JavaScript -->
<script>
    var res = document.createElement("link");
    res.rel = "preload";
    res.as = "style";
    res.href = "styles/other.css";
    document.head.appendChild(res);
</script>
```

Preload with HTTP header:

```
Link: </styles/other.css> rel=“preload”; as=“style” nopush
```

```
< GET /index.html

< HTTP/1.1 200 OK
< Cache-Control: private, max-age=60, must-revalidate
< Expires: Thu, 20 Apr 2017 21:39:52 GMT
< Last-Modified: Thu, 20 Apr 2017 13:11:33 GMT
< ETag: "966eca3815d88d8848933d33c68ab2bd"
< Content-Type: text/html
< Content-Encoding: gzip
< Content-Length: 43177
< Date: Thu, 20 Apr 2017 21:39:52 GMT
< Connection: keep-alive
< Link: <main.css>; as=style; rel=preload; nopush,
  <MetricWeb-Regular.woff>; rel=preload as=font crossorigin nopush,
  <MetricWeb-Semibold.woff>; rel=preload as=font crossorigin nopush,
  <ftlogo.svg>; as=image; rel=preload; nopush
```

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

Step_1

1. ttfmp.fastlylabs.com - index.html 984 ms
2. ttfmp.fastlylabs.com - main.css 411 ms
3. www.ft.com - h...-markets-data.png 928 ms
4. www.ft.com - ftlogo:brand-myft 210 ms
5. www.ft.com - fticon-v1:hamburger 196 ms
6. www.ft.com - ftlogo:brand-ft 215 ms
7. www.ft.com - f...brand-ft-masthead 200 ms
8. www.ft.com - fticon-v1:search 274 ms
9. www.ft.com - fticon-v1:arrow-up 252 ms
10. www.ft.com - f...nd-nikkei-tagline 259 ms
11. www.ft.com - fticon-v1:arrow-left 246 ms
12. www.ft.com - fticon-v1:cross 268 ms
13. www.ft.com - fticon-v1:arrow-right 215 ms
14. www.ft.com - fticon-v1:play 240 ms
15. www.ft.com - fticon-v1:play 234 ms
16. www.ft.com - fticon-v1:arrow-right 228 ms
17. www.ft.com - M...cWeb-Regular.woff 752 ms
18. www.ft.com - F...yWeb-Regular.woff 1522 ms
19. www.ft.com - M...Web-Semibold.woff 931 ms
20. next-geebee.ft... - polyfill.min.js 376 ms
...

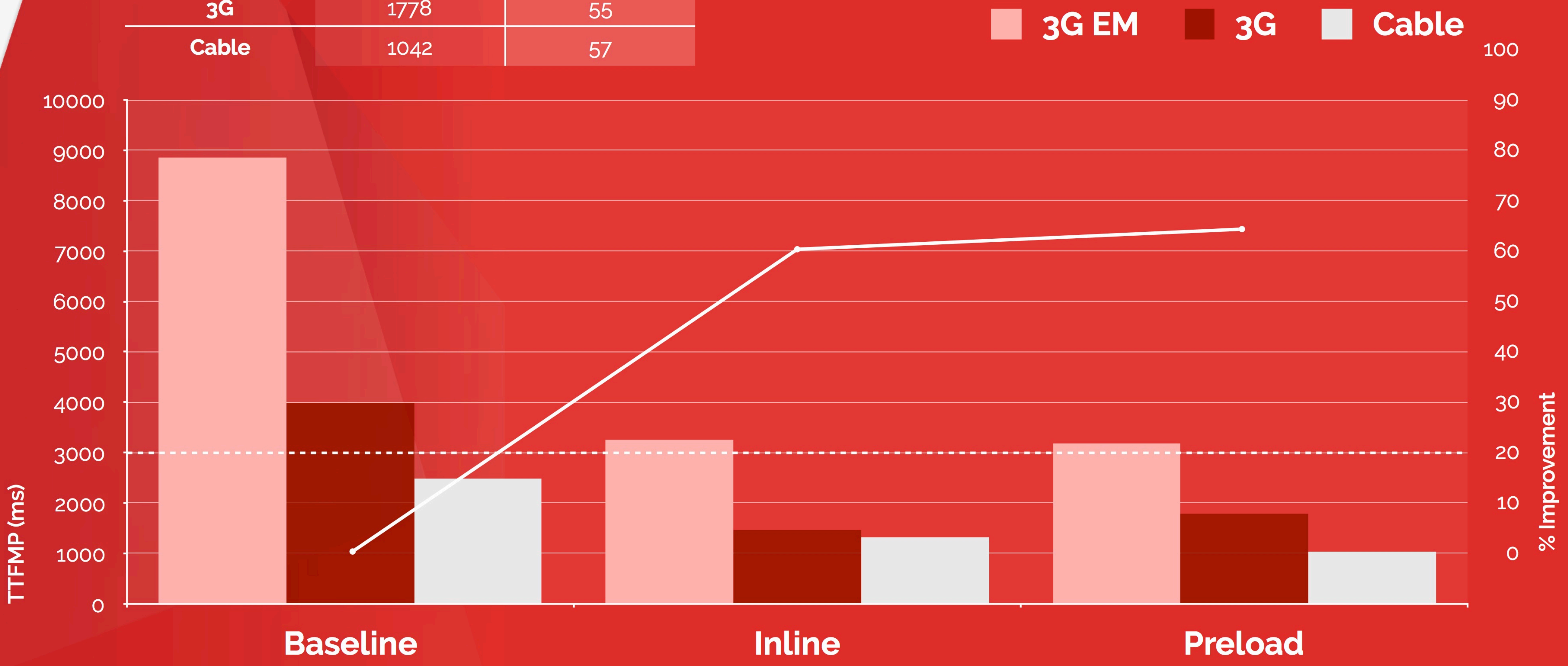
Before

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
Step_1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
1. ttfmp.fastlylabs.com - index.html
2935 ms
2. ttfmp.fastlylabs.com - main.css
3235 ms
3. www.ft.com - ftlogo:brand-myft
653 ms
4. www.ft.com - fticon-v1:hamburger
597 ms
5. www.ft.com - f...brand-ft-masthead
3554 ms
6. www.ft.com - M...cWeb-Regular.woff
2971 ms
7. www.ft.com - F...yWeb-Regular.woff
5603 ms
8. www.ft.com - M...Web-Semibold.woff
3462 ms
9. www.ft.com - es5.min.js
3954 ms
10. www.ft.com - main-without-n-ui.js
6179 ms
11. www.ft.com - fticon-v1:play
566 ms
12. www.ft.com - fticon-v1:arrow-right
552 ms
13. next-geebee.ft...- polyfill.min.js
3634 ms
14. www.ft.com - fticon-v1:arrow-right
5992 ms
15. www.ft.com - fticon-v1:cross
6005 ms
16. www.ft.com - f...nd-nikkei-tagline
6002 ms
17. www.ft.com - fticon-v1:arrow-up
6014 ms
18. www.ft.com - fticon-v1:search
6026 ms
19. www.ft.com - ftlogo:brand-ft
6033 ms
20. www.ft.com - fticon-v1:arrow-left
5995 ms
...
1
2
3
4
5
6
7
8
9
10
11
12
13
14

After

Preload results

	TTFMP	% impovement
3G EM	3176	64
3G	1778	55
Cable	1042	57

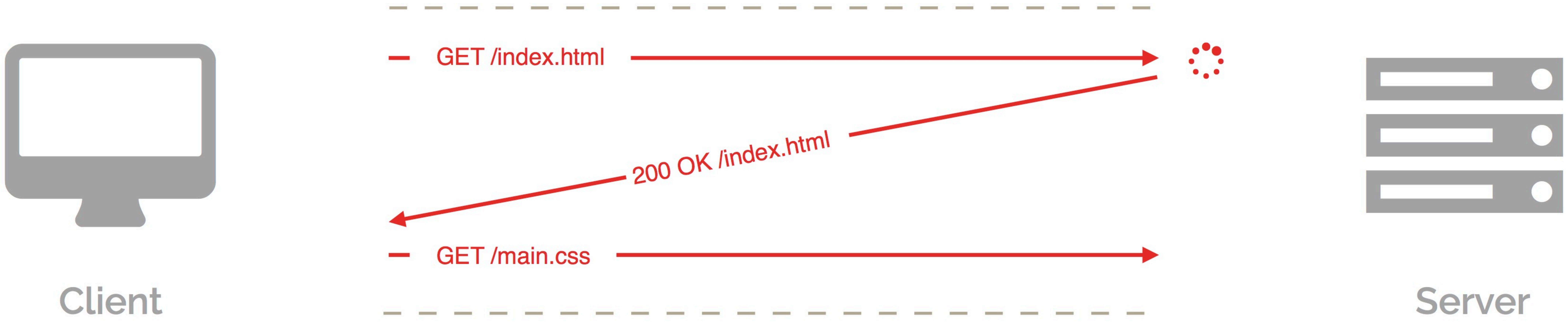


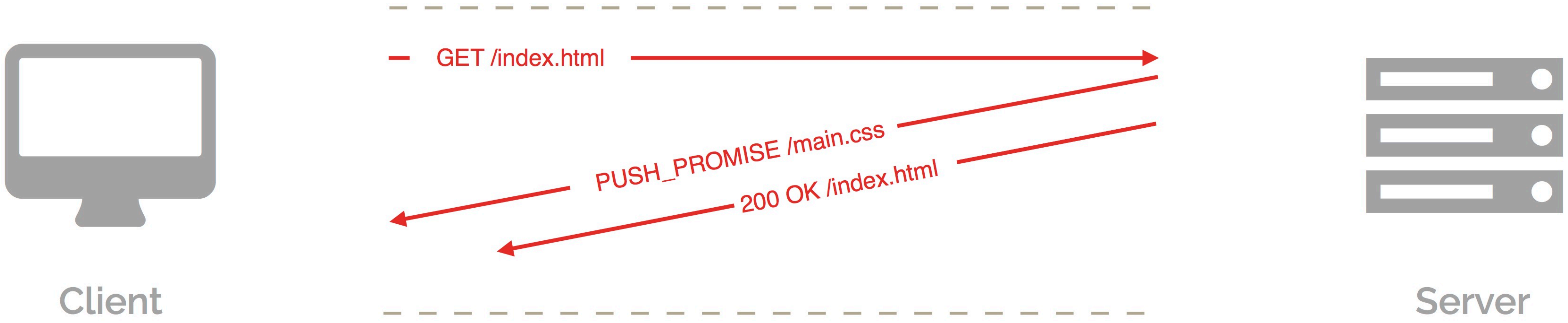
Pros

- Indicate hidden resources
- Dictate priority and order
- Separates fetch from exec

Cons

- Easy to create contention
- Requires server logic


GET /index.html
200 OK /index.html
GET /main.css
Client
Server


GET /index.html
PUSH_PROMISE /main.css
200 OK /index.html
Client
Server

Indicate push of critical styles with Link preload header:

```
Link: <critical.css>; rel=“preload”; as=“style”
```

Convert inline styles into normal link rel=“stylesheet” declaration and async the main styles

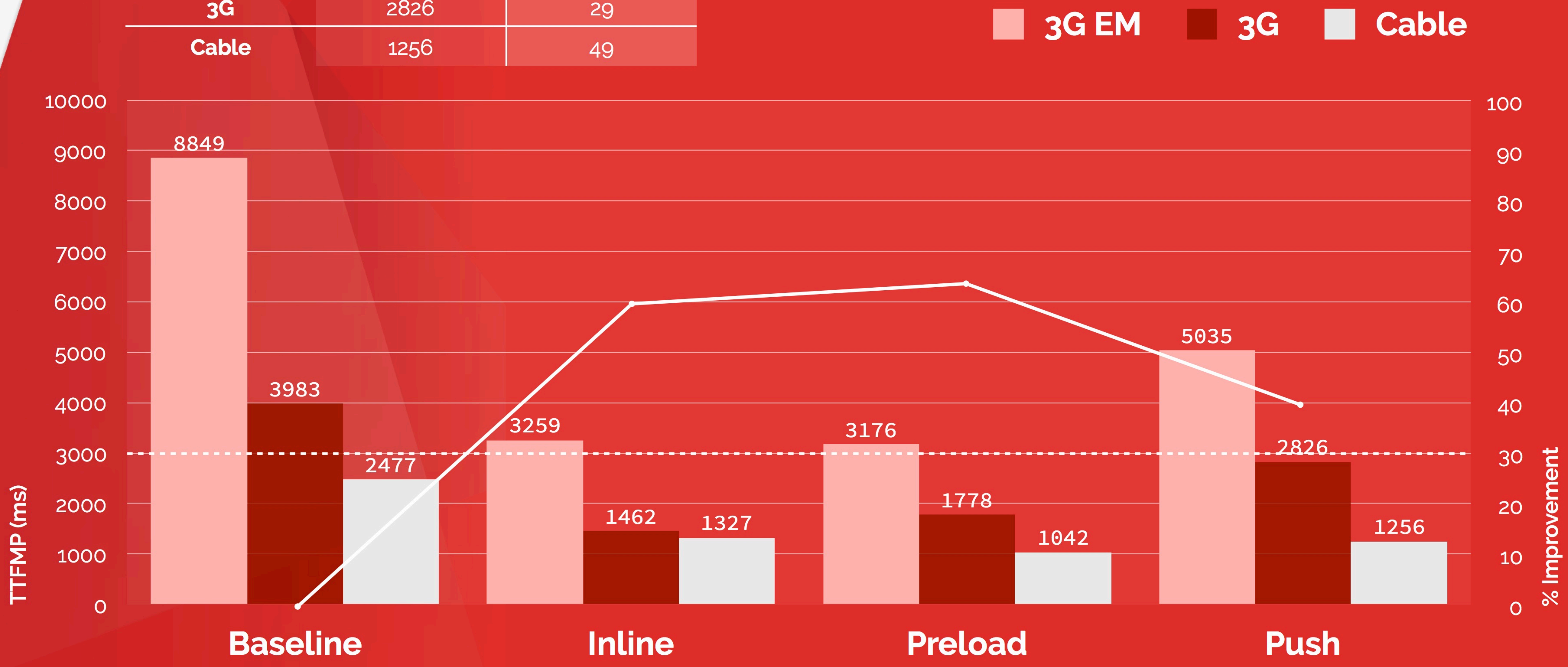
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>FT.com</title>

        <link rel="stylesheet" href="critical.css" />
        <link rel="preload" href="main.css" as="style" onload=“this.rel='stylesheet'" />
       ...

```

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
Step_1
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
9.5
1. ttfmp.fastlylabs.com - index.html
2547 ms
2. ttfmp.fastlylabs.com - main.css
2460 ms
3. ttfmp.fastlyla...om - critical.css
274 ms
4. next-geebee.ft...- polyfill.min.js
2973 ms
5. www.ft.com - ftlogo:brand-ft
496 ms
6. www.ft.com - h...-markets-data.png
512 ms
7. www.ft.com - M...cWeb-Regular.woff
2802 ms
8. www.ft.com - F...yWeb-Regular.woff
5523 ms
9. www.ft.com - M...Web-Semibold.woff
3374 ms
10. www.ft.com - fticon-v1:hamburger
2224 ms
11. www.ft.com - ftlogo:brand-myft
2316 ms
12. www.ft.com - f...brand-ft-masthead
2218 ms
13. www.ft.com - fticon-v1:cross
6192 ms
14. www.ft.com - fticon-v1:arrow-left
6213 ms
15. www.ft.com - fticon-v1:arrow-up
6211 ms
16. www.ft.com - fticon-v1:search
6208 ms
17. www.ft.com - fticon-v1:play
6198 ms
18. www.ft.com - f...nd-nikkei-tagline
6186 ms
19. www.ft.com - es5.min.js
3621 ms
20. www.ft.com - fticon-v1:arrow-right
6176 ms

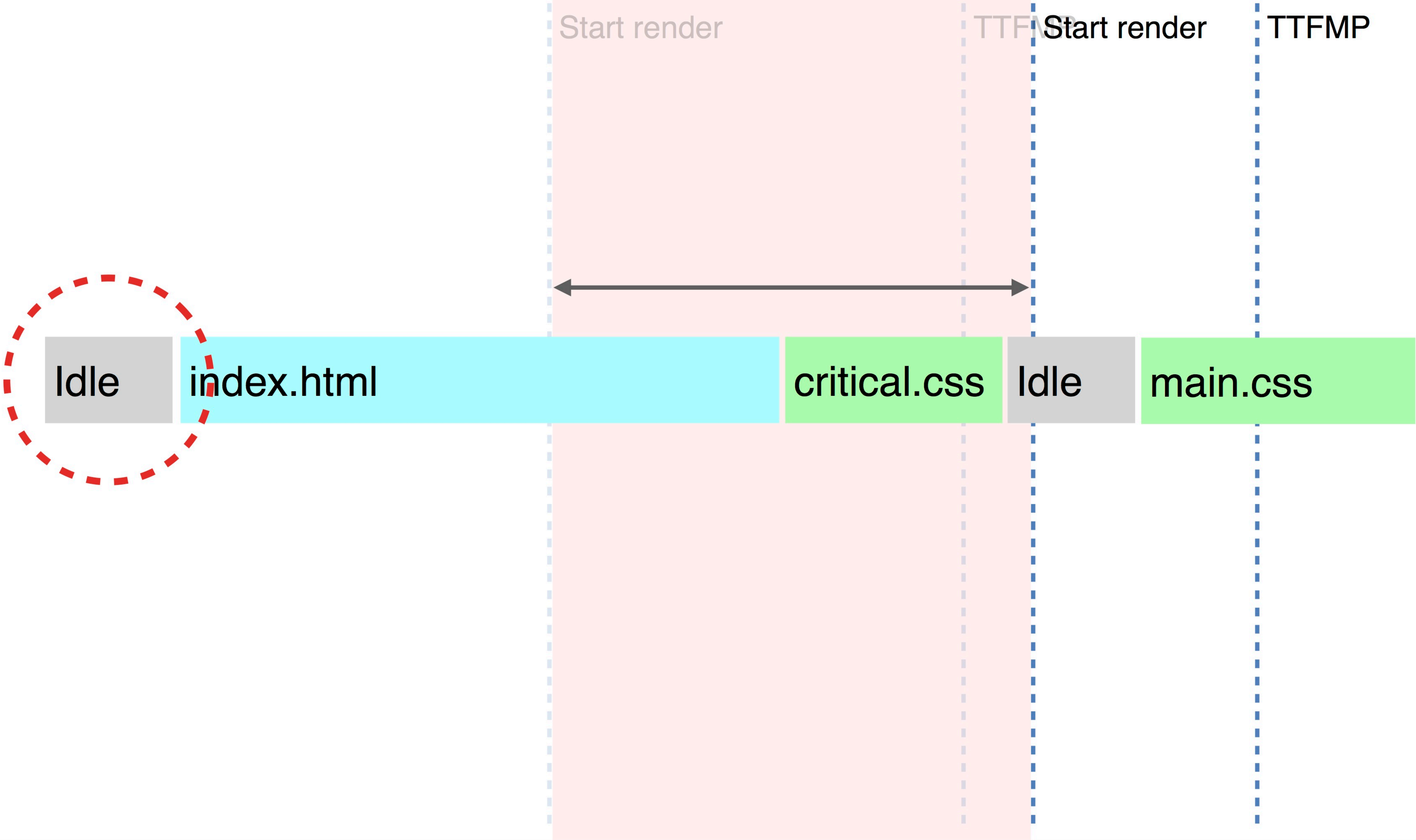
Result

Push results

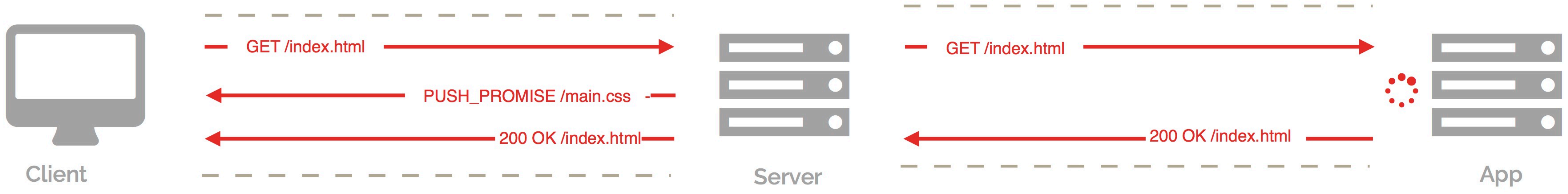
	TTFMP	% impovement
3G EM	5035	43
3G	2826	29
Cable	1256	49



Async push


Start render
TTFMP
Start render
TTFMP
Idle
index.html
critical.css
Idle
main.css

After


GET /index.html
PUSH_PROMISE /main.css
200 OK /index.html
Client
Server
GET /index.html
200 OK /index.html
App

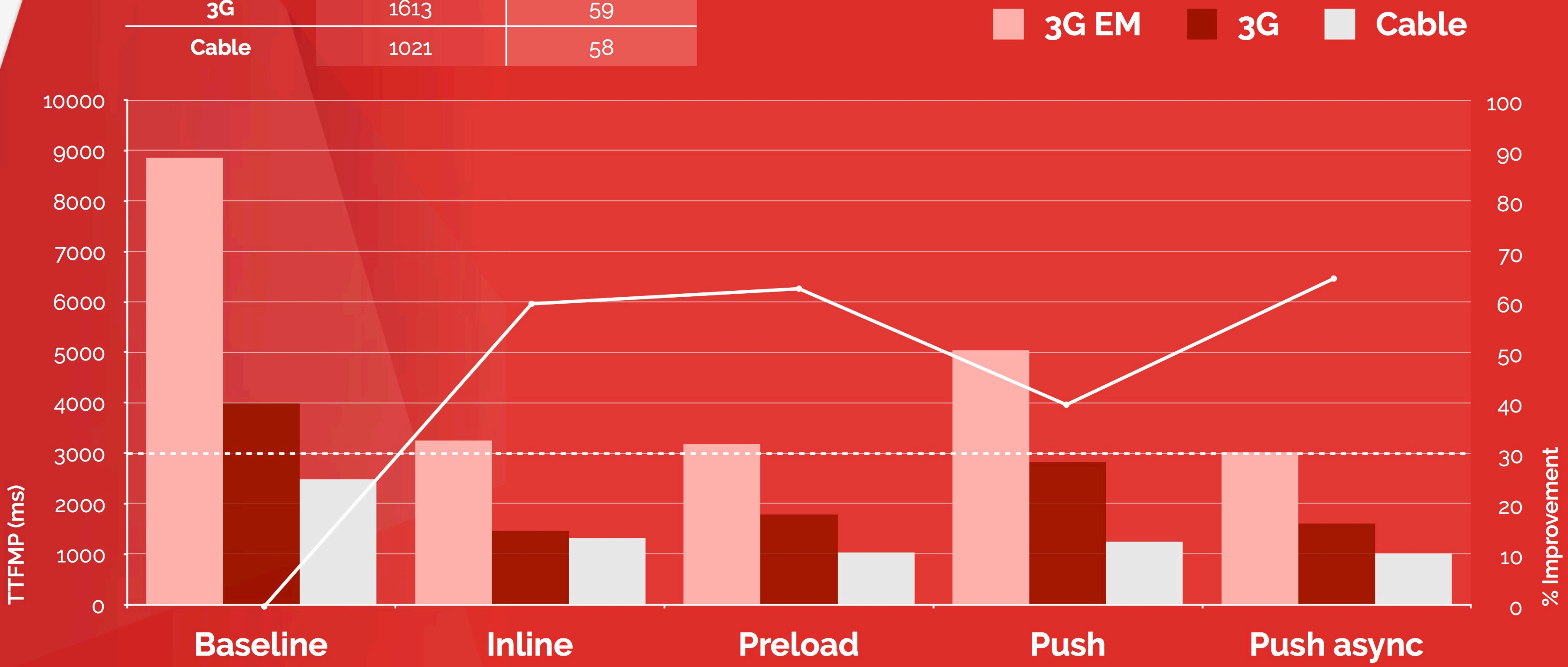
```javascript
const http2 = require('http2');

function handler(request, response) {
  if (request.url === "/index.html") {
    const push = response.push('/critical.css');
    push.writeHead(200);
    fs.createReadStream('/critical.css').pipe(push);
  }

  // Generate index response:
  // - Fetch data from  DB
  // - Render template
  // etc ...

  response.end(data);
}

const server = http2.createServer(opts, handler);
server.listen(80);
```

dns
connect
ssl
html
js
css
image
flash
font
other
JS Execution
Step_1
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5
1. ttfmp.fastlylabs.com - index.html
2972 ms
2. ttfmp.fastlyla...om - critical.css
407 ms
...
4. ttfmp.fastlylabs.com - main.css
1209 ms
5. www.ft.com - fticon-v1:hamburger
3027 ms
6. www.ft.com - ftlogo:brand-ft
821 ms
7. www.ft.com - f...brand-ft-masthead
791 ms
8. www.ft.com - h...-markets-data.png
610 ms
9. www.ft.com - ftlogo:brand-myft
725 ms
10. www.ft.com - M...Web-Semibold.woff
6483 ms
11. www.ft.com - F...yWeb-Regular.woff
3241 ms
12. www.ft.com - M...cWeb-Regular.woff
3016 ms
13. www.ft.com - fticon-v1:arrow-right
490 ms
14. www.ft.com - fticon-v1:play
497 ms
15. www.ft.com - main-without-n-ui.js
6327 ms
16. www.ft.com - fticon-v1:search
6377 ms
17. www.ft.com - es5.min.js
3638 ms
18. www.ft.com - fticon-v1:cross
6356 ms
19. www.ft.com - f...nd-nikkei-tagline
6349 ms
20. www.ft.com - fticon-v1:arrow-right
6335 ms
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5

After

Push async results

	TTFMP	% impovement
3G EM	3062	65
3G	1613	59
Cable	1021	58



Pros

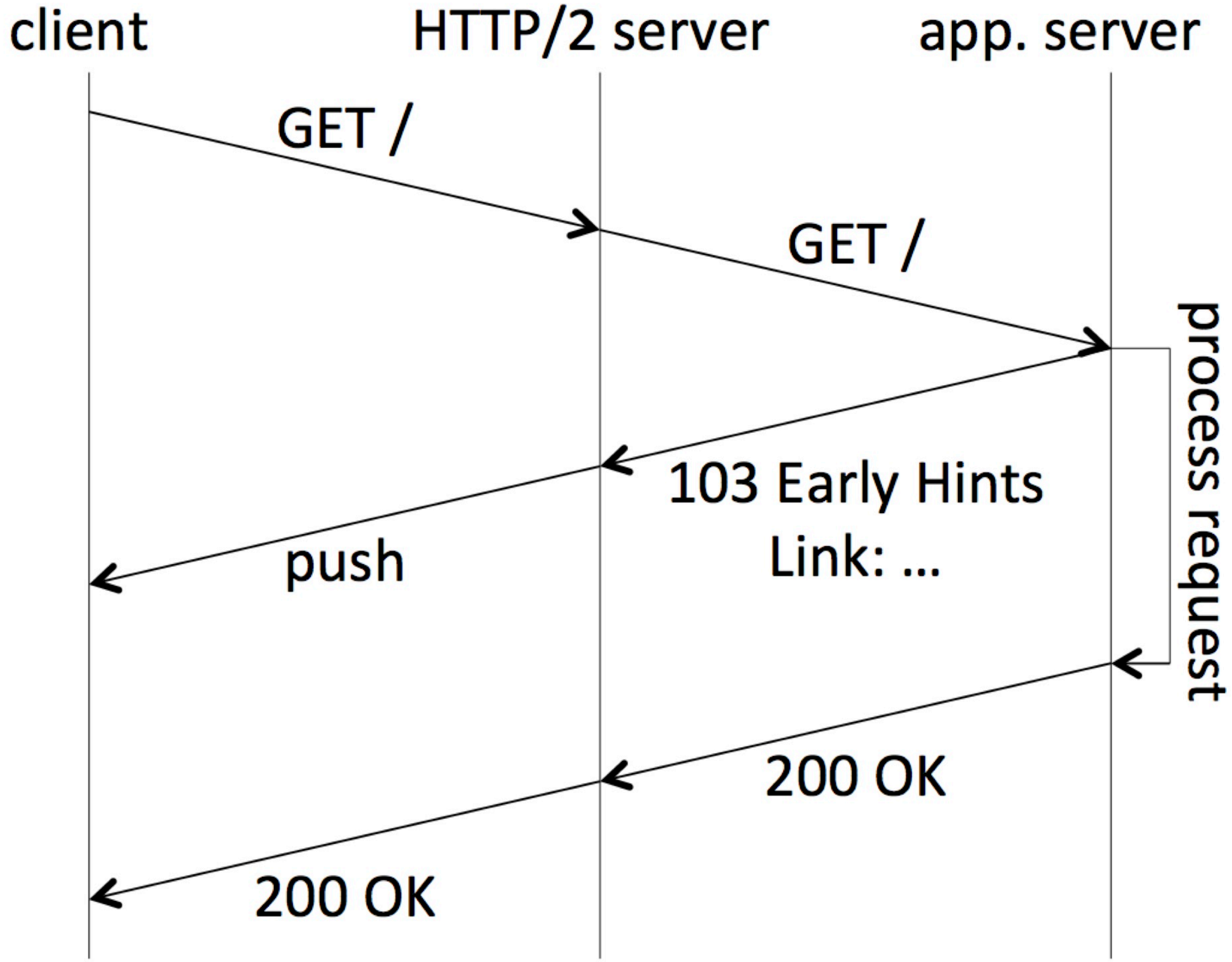
- Uses idle time
- Ensures delivery before preload

Cons

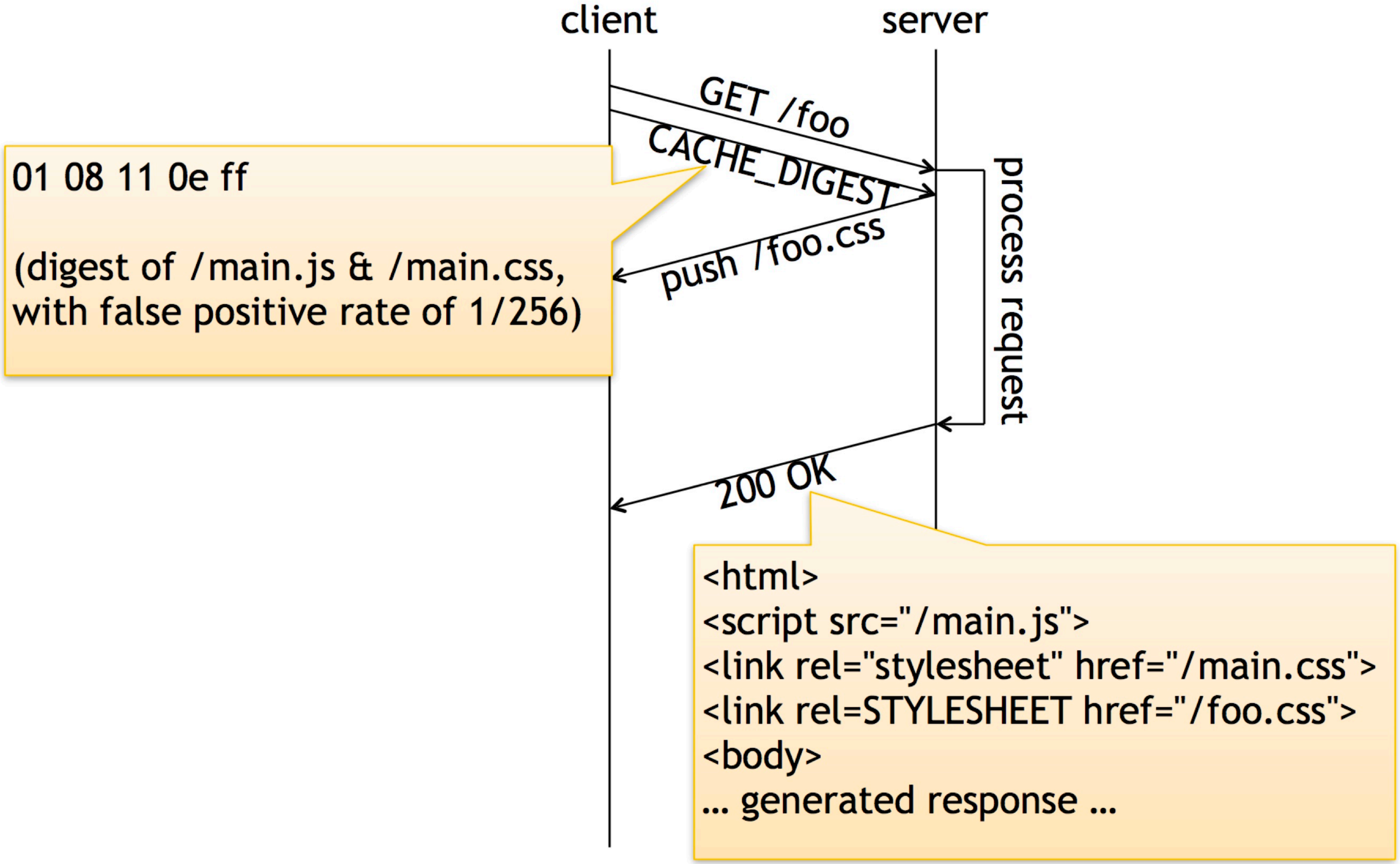
- Easy to create contention
- Limited avaliablity
- Custom server logic
- Hard to debug

The future

```
GET / HTTP/1.1
Host: example.com

HTTP/1.1 103 Early Hints
Link: </style.css>; rel=preload

HTTP/1.1 200 OK
Content-Type: text/html
Link: </style.css>; rel=preload

<!DOCTYPE HTML>
...
```



103 Early hints status code for HTTP, K. Oku https://tools.ietf.org/html/draft-ietf-httpbis-early-hints-01



01 08 11 0e ff

(digest of /main.js & /main.css, with false positive rate of 1/256)

```
<html>
<script src="/main.js">
<link rel="stylesheet" href="/main.css">
<link rel=STYLESHEET href="/foo.css">
<body>
… generated response …
```

Cache digests for HTTP/2, K. Oku https://tools.ietf.org/html/draft-ietf-httpbis-cache-digest-01

HTTP/2 Strategy

- A switch to HTTP/2 *isn’t just a switch* — both existing content and deployment strategy have to be revised:

  — Prepare the infrastructure: HTTPS, servers and CDNs,
  — In front-end, make use of *multiplexing* and parallelism,
  — Load CSS *progressively*, grouping CSS in separate files,
  — *Load versioned scripts* via scout with logically grouped files,
  — *Prevent and monitor* mixed content issues and warnings,
  — Keep in mind *server push* to serve critical content faster,
  — Combine HTTP/2 with *service workers* to maximize perf.

LEVEL5

RESOURCEHINTS

Resource hints allow developers to provide some *hints* to the browser to prompt the download of assets, or rendering, *silently* in the background.

Resource hints allow developers to provide some *hints* to the browser to prompt the download of assets, or rendering, *silently* in the background.

- `<link rel="prefetch" href="(url)">` tells browsers to fetch a resource that will *probably* be needed for the next navigation *(low priority)*.

- `<link rel="prefetch" href="(url)">`

  tells browsers to fetch a resource that will *probably* be needed for the next navigation *(low priority)*.

- `<link rel="prerender" href="(url)">`

  tells browsers to render the specified page in the background *(low priority)*.

- `<link rel="prerender" href="(url)">`

  tells browsers to render the specified page in the background *(low priority)*.

- `<link rel="dns-prefetch" href="(url)">`

  gives a hint to the browser to perform a DNS lookup in the background *(low priority)*.

- `<link rel="dns-prefetch" href="(url)">`

  gives a hint to the browser to perform a DNS lookup in the background *(low priority)*.

- `<link rel= "preconnect" href="(url)">`

  gives a hint to the browser to initiate the connection handshake (DNS, TCP, TLS) in the background *(low priority)*.

- `<link rel="preconnect" href="(url)">`

  gives a hint to the browser to initiate the connection handshake (DNS, TCP, TLS) in the background *(low priority)*.

- `<link rel="preload" href="(url)" as="(type)">`

  gives a hint to the browser to prefetch resources and set the right resource priority for loading assets.

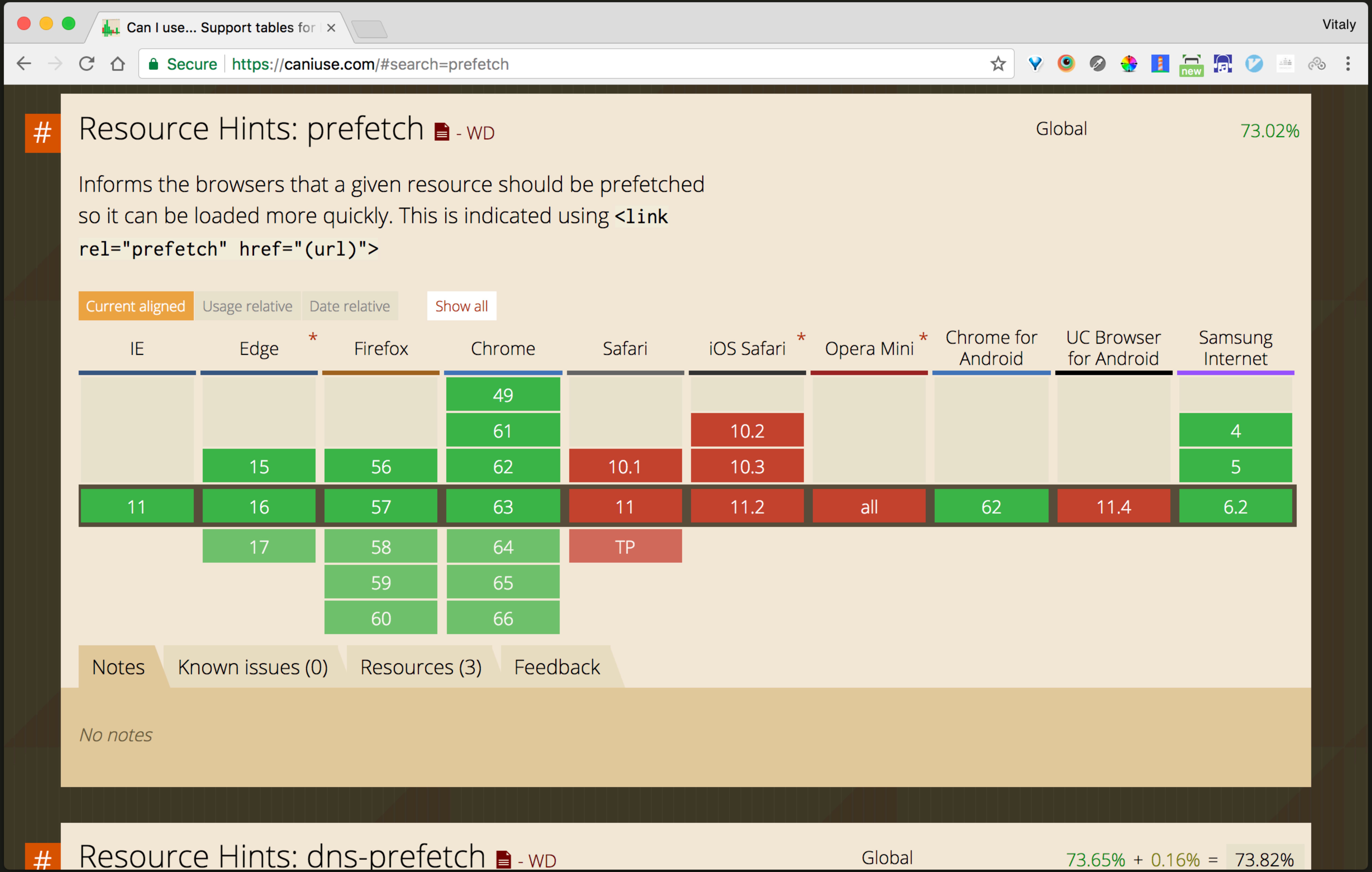
The basic use case for `preload` is loading of *late-discovered critical resources*. If we omit the `as` attribute, it’s just an XHR request, fetching with a fairly low priority.

```html
<link rel="preload"
      href="late-discovered.js"
      as="script">
```

The `as` attribute tells the browser what it is downloading. E.g. `audio`, `font`, `image`, `script`, `style`, `track`, `video`, `document`.

```
<link rel="preload"
      href="font.woff2”
      type="font/woff2”
      crossorigin
      as="font">
```

E.g. you could include `preload` directives for web fonts that you know you’ll need for rendering of the page.


Can I use... Support tables for
Vitaly
Secure
https://caniuse.com/#search=prefetch

Resource Hints: prefetch - WD

Global 73.02%

Informs the browsers that a given resource should be prefetched so it can be loaded more quickly. This is indicated using `<link rel="prefetch" href="(url)">`

Current aligned | Usage relative | Date relative | Show all

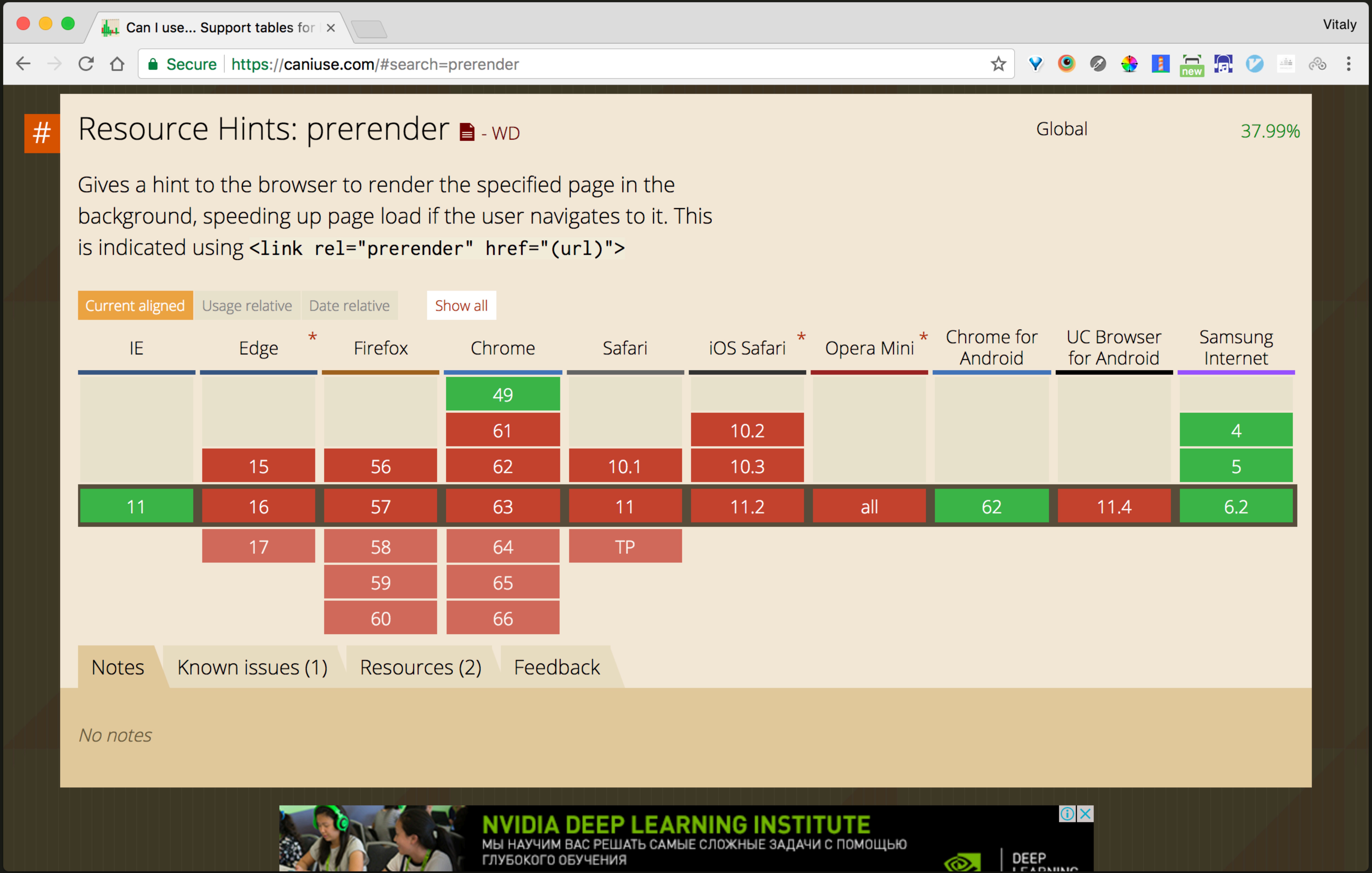
IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			61		10.2				4
	15	56	62	10.1	10.3				5
11	16	57	63	11	11.2	all	62	11.4	6.2
	17	58	64	TP					
		59	65						
		60	66						

Notes | Known issues (0) | Resources (3) | Feedback

No notes

Resource Hints: dns-prefetch - WD

Global 73.65% + 0.16% = 73.82%



Can I use... Support tables for ×
Vitaly
Secure https://caniuse.com/#search=prerender
new

Resource Hints: prerender - WD

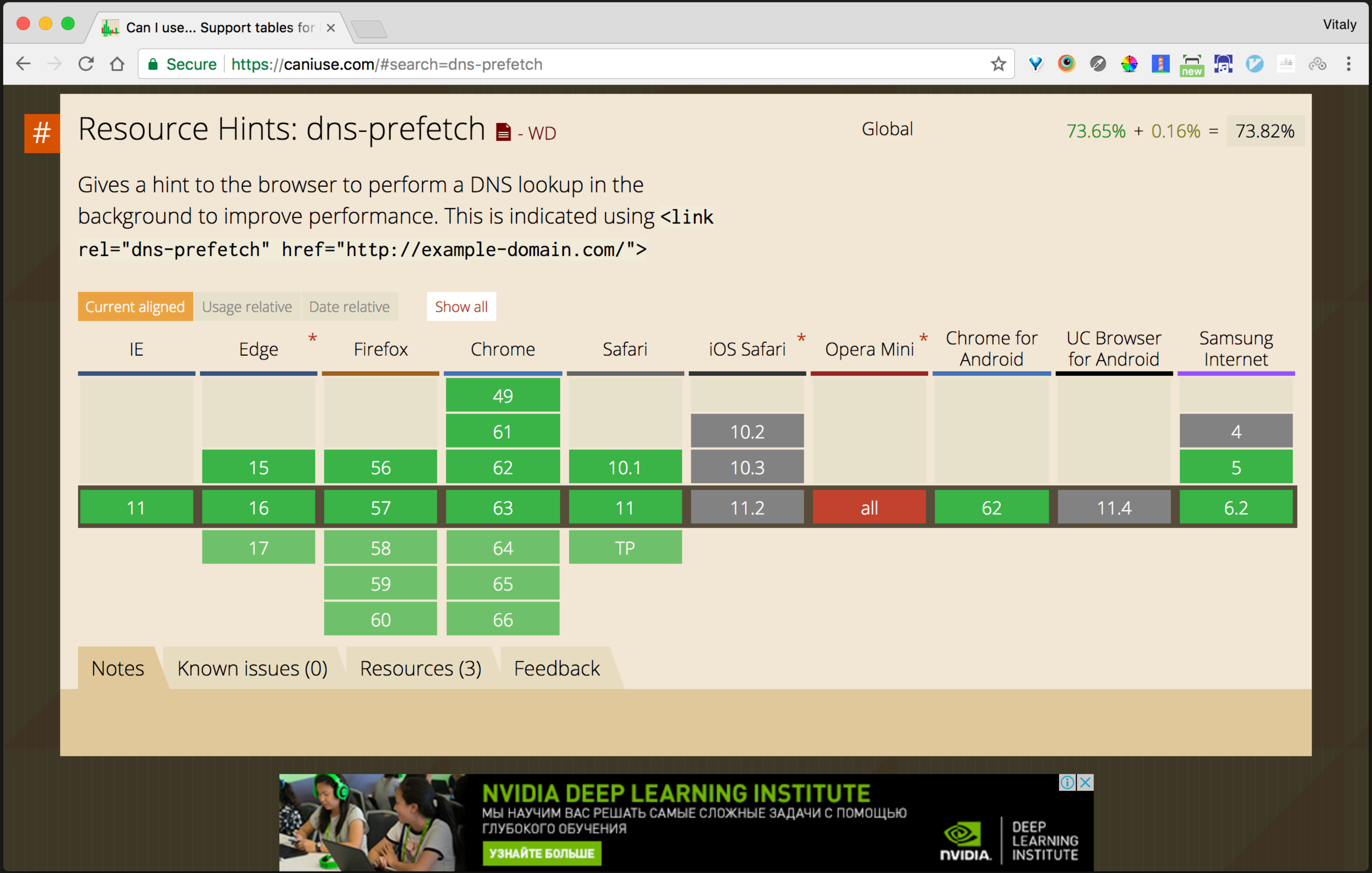
Global 37.99%

Gives a hint to the browser to render the specified page in the background, speeding up page load if the user navigates to it. This is indicated using `<link rel="prerender" href="(url)">`

Current aligned
Usage relative
Date relative
Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			🟩 49						
			🟥 61		🟥 10.2				🟩 4
	🟥 15	🟥 56	🟥 62	🟥 10.1	🟥 10.3				🟩 5
🟩 11	🟥 16	🟥 57	🟥 63	🟥 11	🟥 11.2	🟥 all	🟩 62	🟥 11.4	🟩 6.2
	🟥 17	🟥 58	🟥 64	🟥 TP					
		🟥 59	🟥 65						
		🟥 60	🟥 66						

No notes

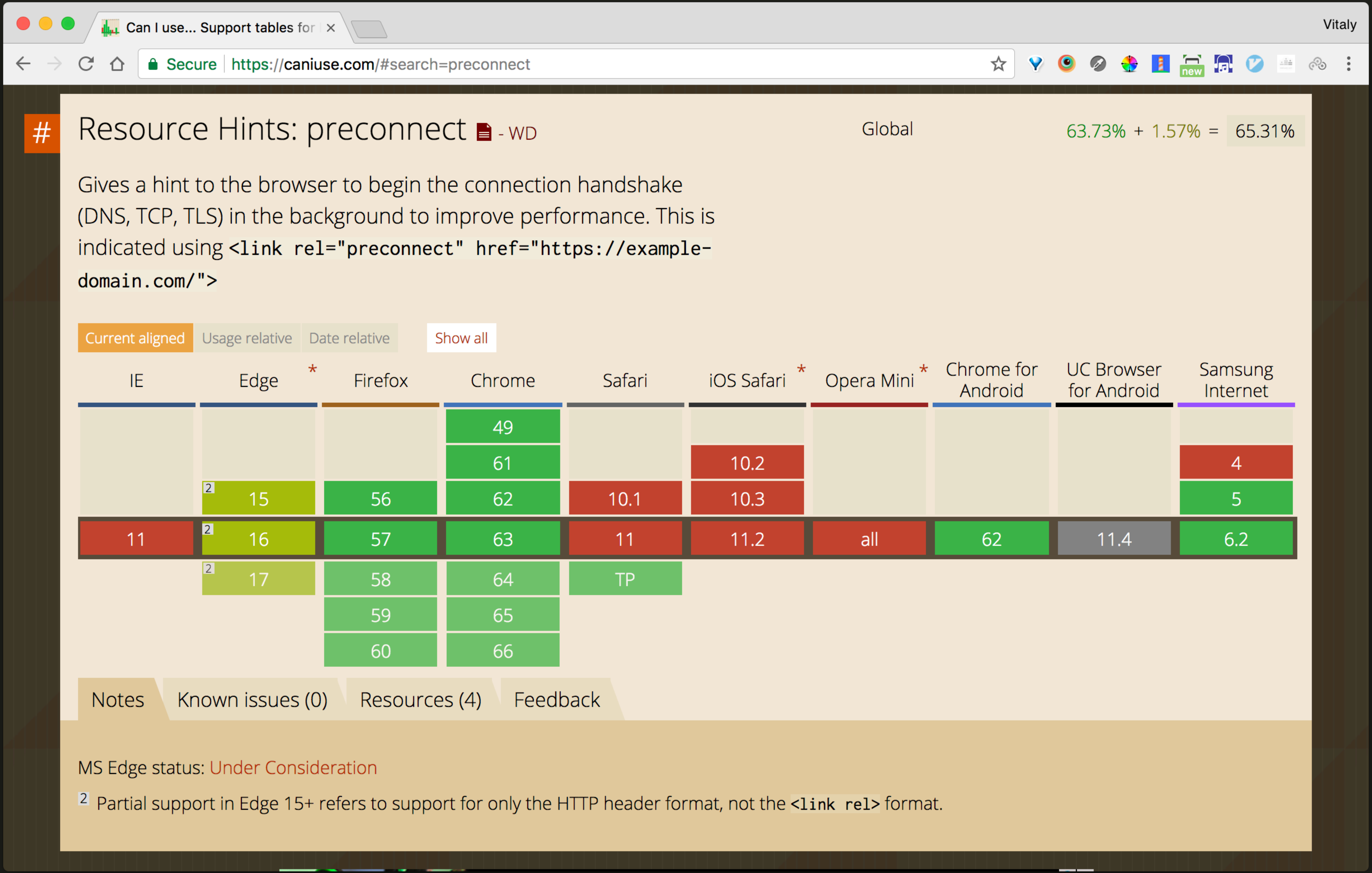

Can I use... Support tables for
Vitaly
Secure
https://caniuse.com/#search=dns-prefetch

Resource Hints: dns-prefetch - WD

Global 73.65% + 0.16% = 73.82%

Gives a hint to the browser to perform a DNS lookup in the background to improve performance. This is indicated using `<link rel="dns-prefetch" href="http://example-domain.com/">`

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			61		10.2				4
	15	56	62	10.1	10.3				5
11	16	57	63	11	11.2	all	62	11.4	6.2
	17	58	64	TP					
		59	65						
		60	66						



Can I use... Support tables for ×
Vitaly
Secure https://caniuse.com/#search=preconnect
new

Resource Hints: preconnect - WD

Global 63.73% + 1.57% = 65.31%

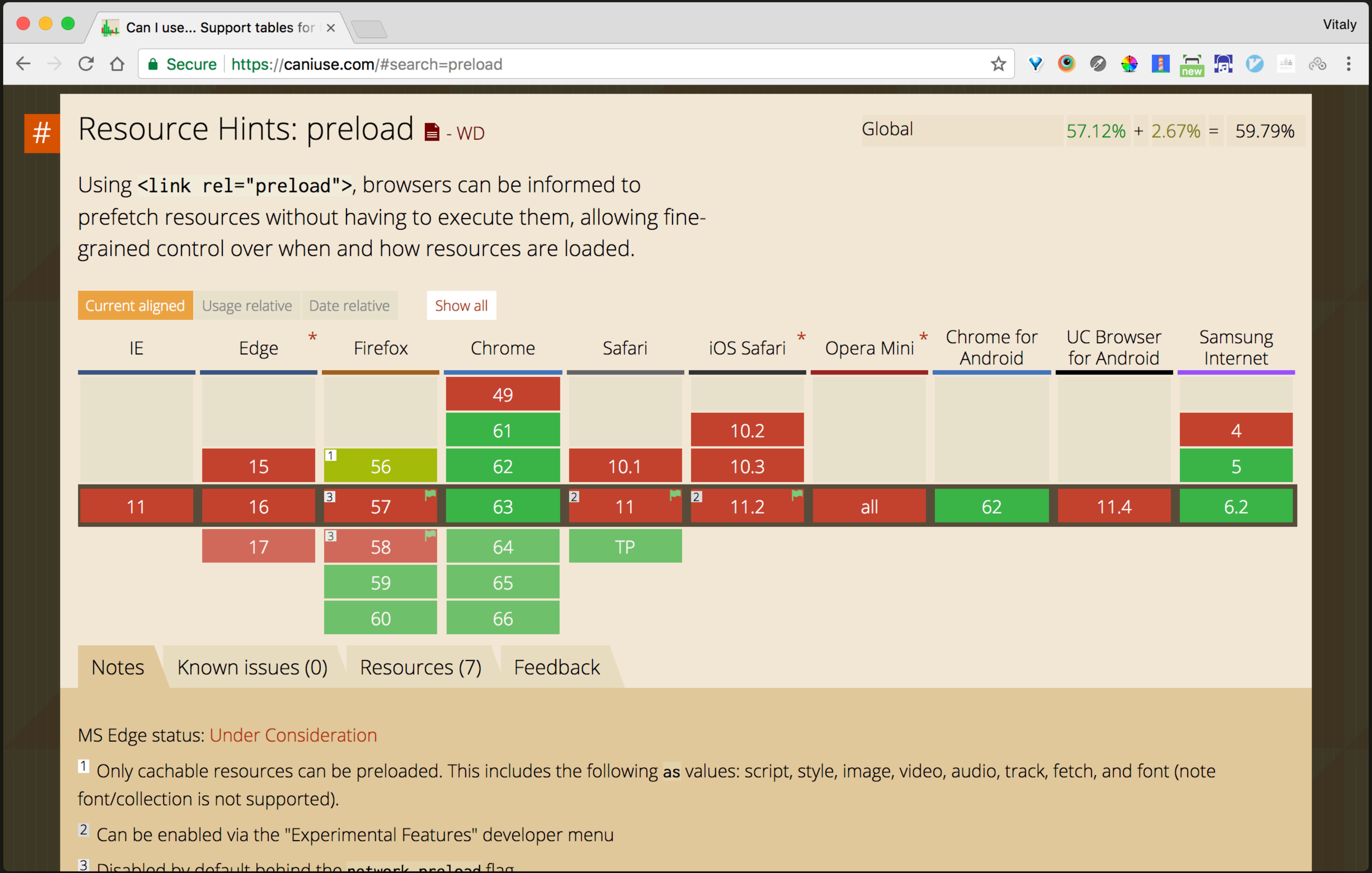
Gives a hint to the browser to begin the connection handshake (DNS, TCP, TLS) in the background to improve performance. This is indicated using `<link rel="preconnect" href="https://example-domain.com/">`

Current aligned
Usage relative
Date relative
Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			61		10.2				4
	[2] 15	56	62	10.1	10.3				5
11	[2] 16	57	63	11	11.2	all	62	11.4	6.2
	[2] 17	58	64	TP					
		59	65						
		60	66						

MS Edge status: Under Consideration

[2] Partial support in Edge 15+ refers to support for only the HTTP header format, not the `<link rel>` format.



Can I use... Support tables for
Vitaly
Secure
https://caniuse.com/#search=preload

Resource Hints: preload - WD

Global 57.12% + 2.67% = 59.79%

Using `<link rel="preload">`, browsers can be informed to prefetch resources without having to execute them, allowing fine-grained control over when and how resources are loaded.

Current aligned
Usage relative
Date relative
Show all

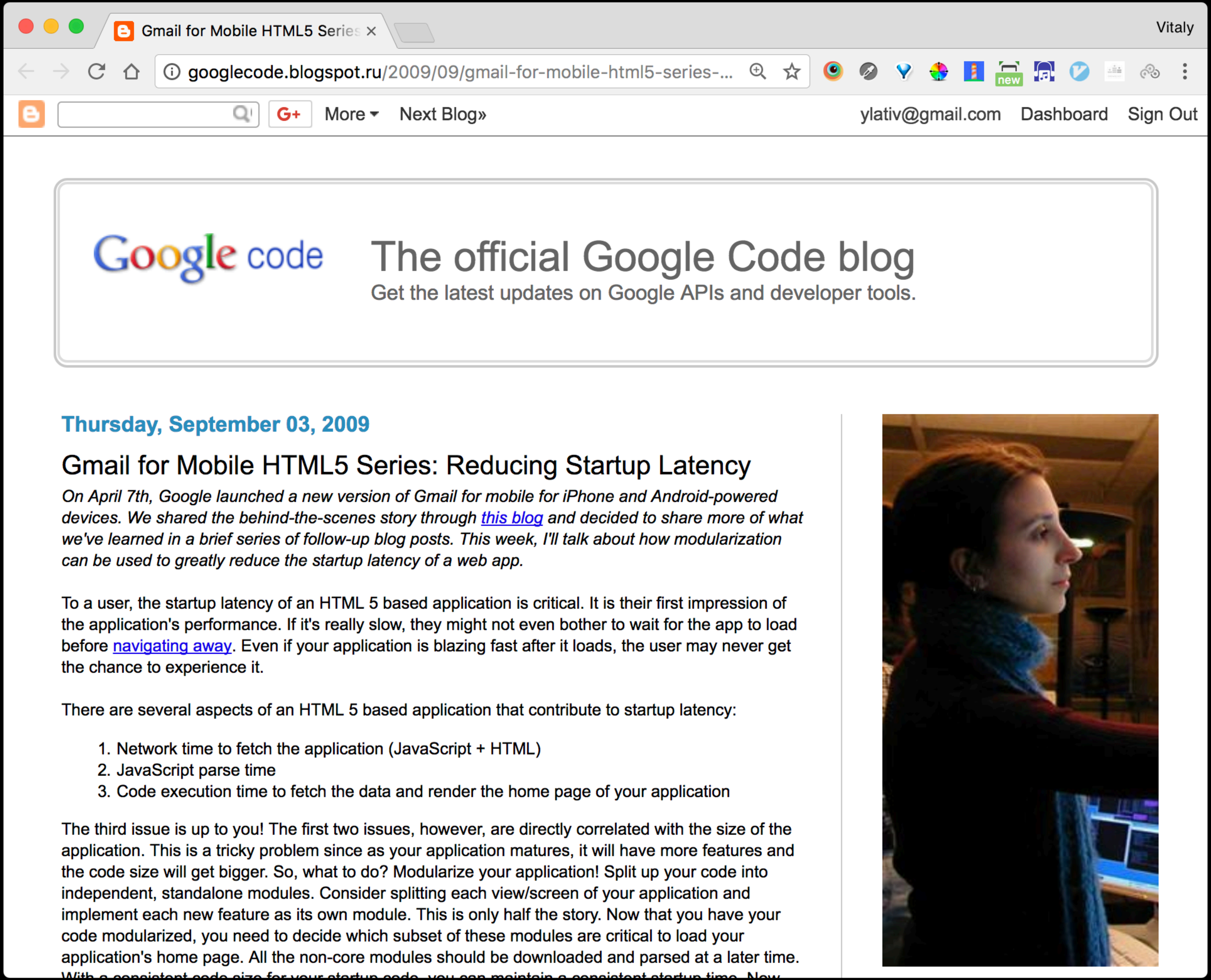
IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			61		10.2				4
	15	[1] 56	62	10.1	10.3				5
11	16	[3] 57	63	[2] 11	[2] 11.2	all	62	11.4	6.2
	17	[3] 58	64	TP					
		59	65						
		60	66						

MS Edge status: Under Consideration

[1] Only cachable resources can be preloaded. This includes the following `as` values: script, style, image, video, audio, track, fetch, and font (note font/collection is not supported).

[2] Can be enabled via the "Experimental Features" developer menu

[3] Disabled by default behind the `network.preload` flag


Gmail for Mobile HTML5 Series
Vitaly
googlecode.blogspot.ru/2009/09/gmail-for-mobile-html5-series-...
G+
More ▾
Next Blog»
ylativ@gmail.com
Dashboard
Sign Out
Google code

The official Google Code blog

Get the latest updates on Google APIs and developer tools.

Thursday, September 03, 2009

Gmail for Mobile HTML5 Series: Reducing Startup Latency

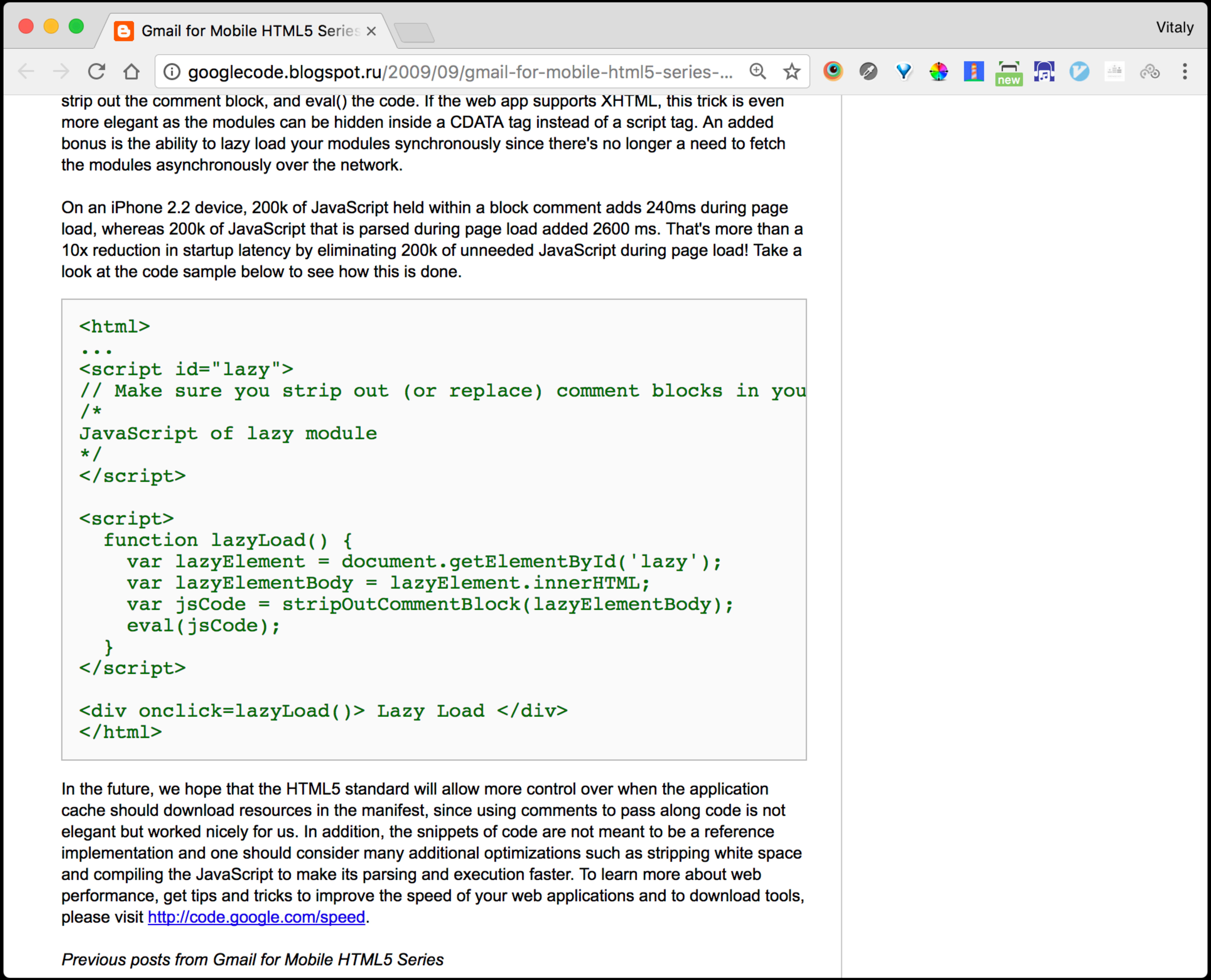
On April 7th, Google launched a new version of Gmail for mobile for iPhone and Android-powered devices. We shared the behind-the-scenes story through this blog and decided to share more of what we've learned in a brief series of follow-up blog posts. This week, I'll talk about how modularization can be used to greatly reduce the startup latency of a web app.

To a user, the startup latency of an HTML 5 based application is critical. It is their first impression of the application's performance. If it's really slow, they might not even bother to wait for the app to load before navigating away. Even if your application is blazing fast after it loads, the user may never get the chance to experience it.

There are several aspects of an HTML 5 based application that contribute to startup latency:

1. Network time to fetch the application (JavaScript + HTML)
2. JavaScript parse time
3. Code execution time to fetch the data and render the home page of your application

The third issue is up to you! The first two issues, however, are directly correlated with the size of the application. This is a tricky problem since as your application matures, it will have more features and the code size will get bigger. So, what to do? Modularize your application! Split up your code into independent, standalone modules. Consider splitting each view/screen of your application and implement each new feature as its own module. This is only half the story. Now that you have your code modularized, you need to decide which subset of these modules are critical to load your application's home page. All the non-core modules should be downloaded and parsed at a later time. With a consistent code size for your startup code, you can maintain a consistent startup time. Now


Gmail for Mobile HTML5 Series
googlecode.blogspot.ru/2009/09/gmail-for-mobile-html5-series-...
Vitaly
new

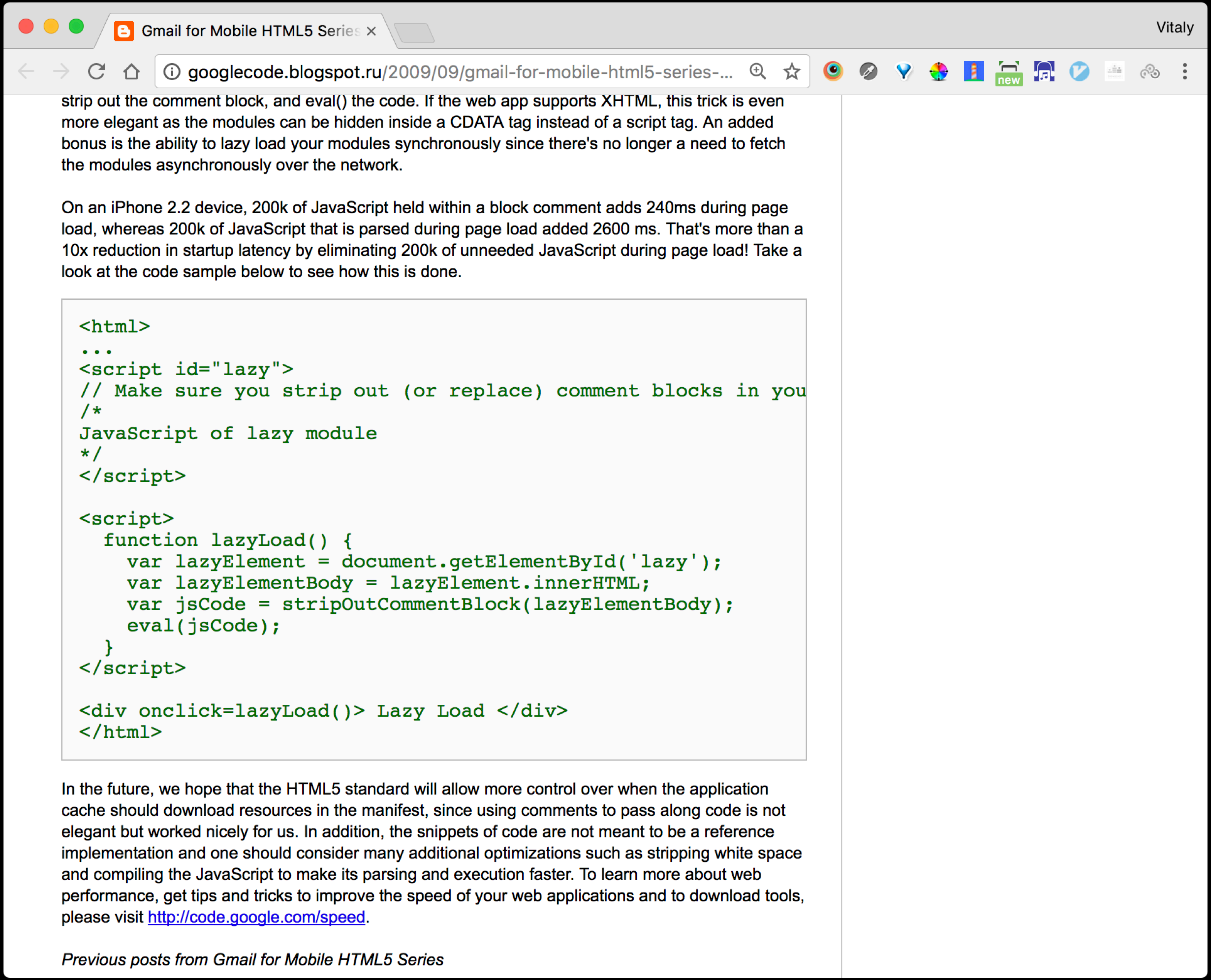
strip out the comment block, and eval() the code. If the web app supports XHTML, this trick is even more elegant as the modules can be hidden inside a CDATA tag instead of a script tag. An added bonus is the ability to lazy load your modules synchronously since there's no longer a need to fetch the modules asynchronously over the network.

On an iPhone 2.2 device, 200k of JavaScript held within a block comment adds 240ms during page load, whereas 200k of JavaScript that is parsed during page load added 2600 ms. That's more than a 10x reduction in startup latency by eliminating 200k of unneeded JavaScript during page load! Take a look at the code sample below to see how this is done.

```
<html>
...
<script id="lazy">
// Make sure you strip out (or replace) comment blocks in you
/*
JavaScript of lazy module
*/
</script>

<script>
  function lazyLoad() {
    var lazyElement = document.getElementById('lazy');
    var lazyElementBody = lazyElement.innerHTML;
    var jsCode = stripOutCommentBlock(lazyElementBody);
    eval(jsCode);
  }
</script>

<div onclick=lazyLoad()> Lazy Load </div>
</html>
```

In the future, we hope that the HTML5 standard will allow more control over when the application cache should download resources in the manifest, since using comments to pass along code is not elegant but worked nicely for us. In addition, the snippets of code are not meant to be a reference implementation and one should consider many additional optimizations such as stripping white space and compiling the JavaScript to make its parsing and execution faster. To learn more about web performance, get tips and tricks to improve the speed of your web applications and to download tools, please visit http://code.google.com/speed.

Previous posts from Gmail for Mobile HTML5 Series


Gmail for Mobile HTML5 Series
Vitaly
googlecode.blogspot.ru/2009/09/gmail-for-mobile-html5-series-...
new

strip out the comment block, and eval() the code. If the web app supports XHTML, this trick is even more elegant as the modules can be hidden inside a CDATA tag instead of a script tag. An added bonus is the ability to lazy load your modules synchronously since there's no longer a need to fetch the modules asynchronously over the network.

On an iPhone 2.2 device, 200k of JavaScript held within a block comment adds 240ms during page load, whereas 200k of JavaScript that is parsed during page load added 2600 ms. That's more than a 10x reduction in startup latency by eliminating 200k of unneeded JavaScript during page load! Take a look at the code sample below to see how this is done.

```
<html>
...
<script id="lazy">
// Make sure you strip out (or replace) comment blocks in you
/*
JavaScript of lazy module
*/
</script>

<script>
  function lazyLoad() {
    var lazyElement = document.getElementById('lazy');
    var lazyElementBody = lazyElement.innerHTML;
    var jsCode = stripOutCommentBlock(lazyElementBody);
    eval(jsCode);
  }
</script>

<div onclick=lazyLoad()> Lazy Load </div>
</html>
```

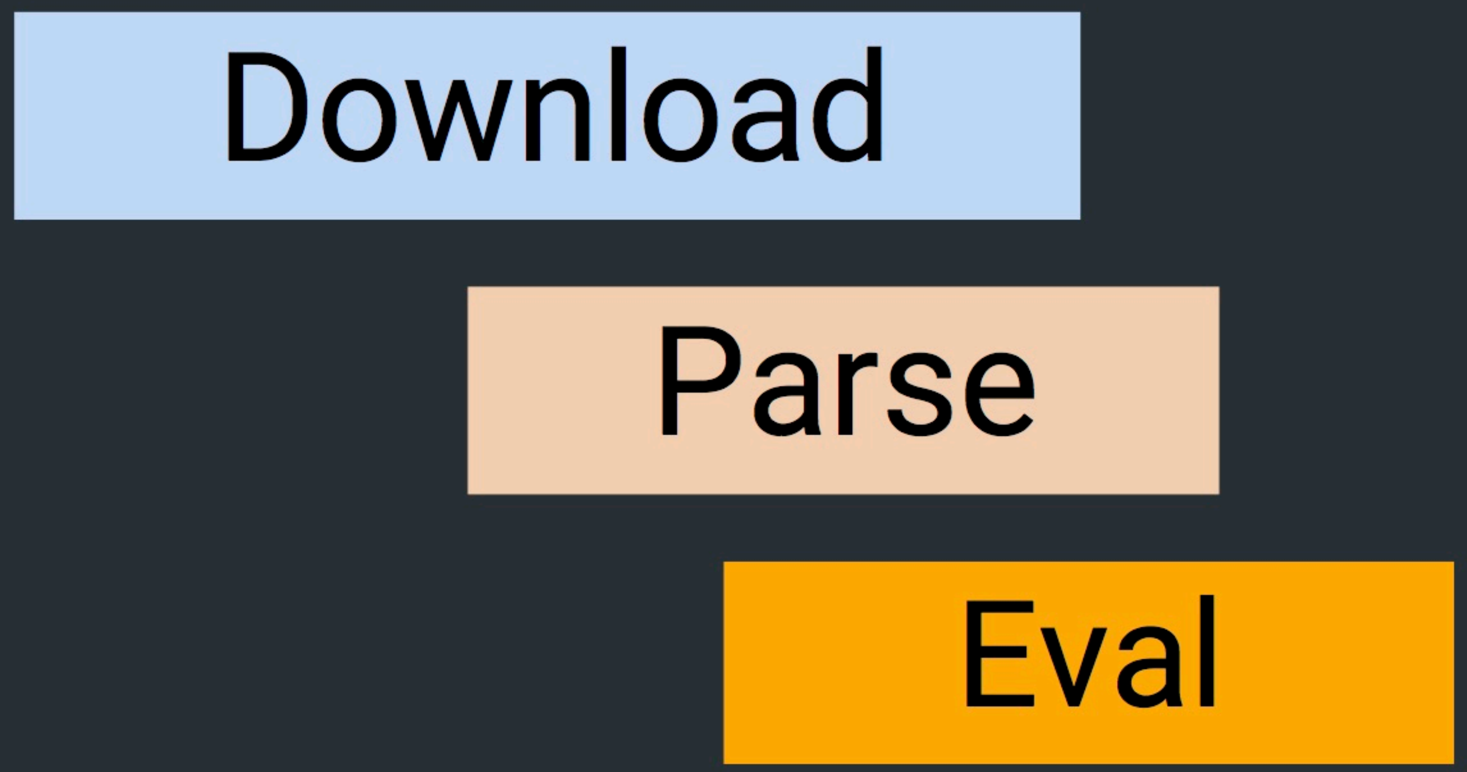
In the future, we hope that the HTML5 standard will allow more control over when the application cache should download resources in the manifest, since using comments to pass along code is not elegant but worked nicely for us. In addition, the snippets of code are not meant to be a reference implementation and one should consider many additional optimizations such as stripping white space and compiling the JavaScript to make its parsing and execution faster. To learn more about web performance, get tips and tricks to improve the speed of your web applications and to download tools, please visit http://code.google.com/speed.

Previous posts from Gmail for Mobile HTML5 Series

JavaScript has a cost.

Fast =

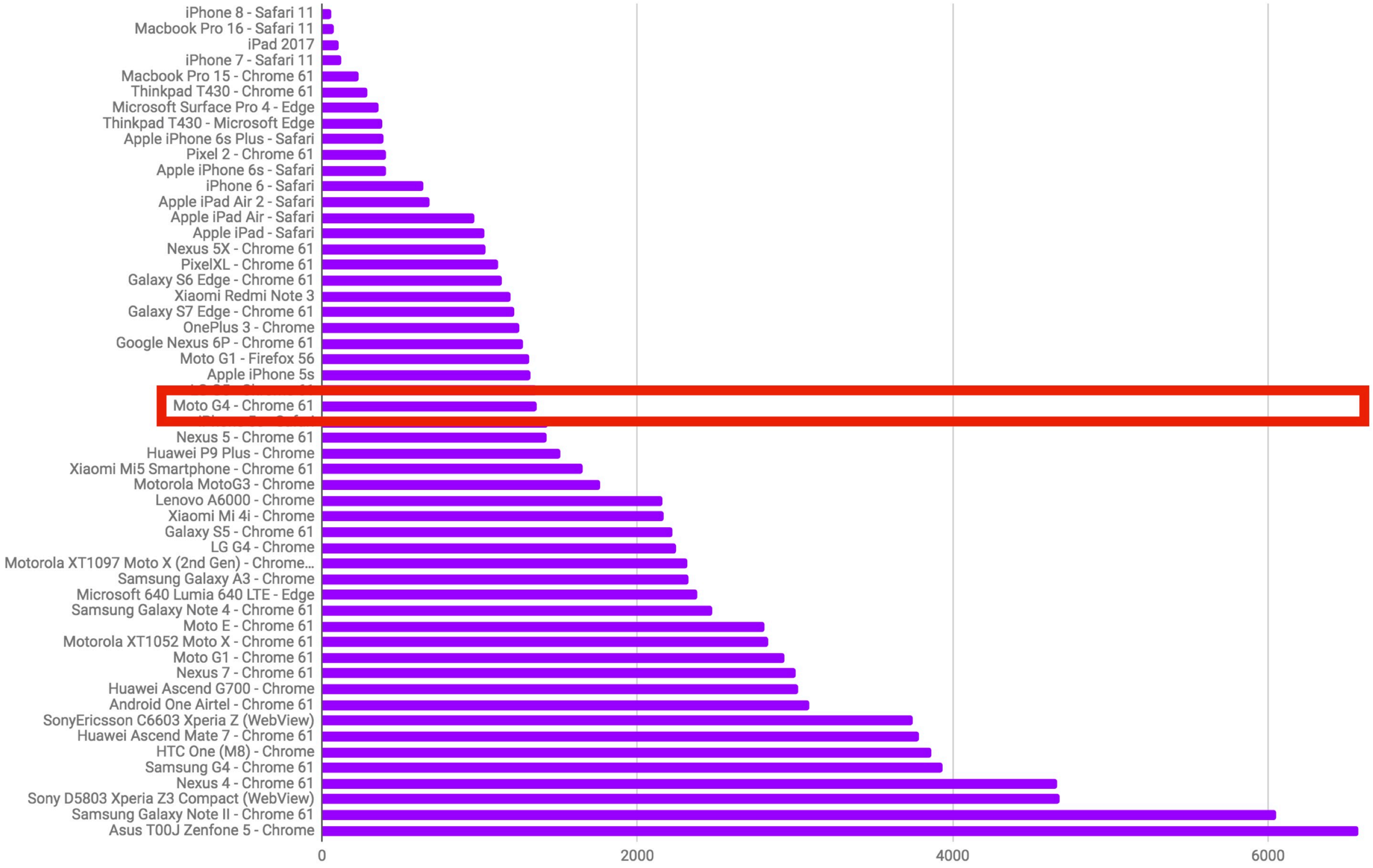
Fast at



On mobile devices

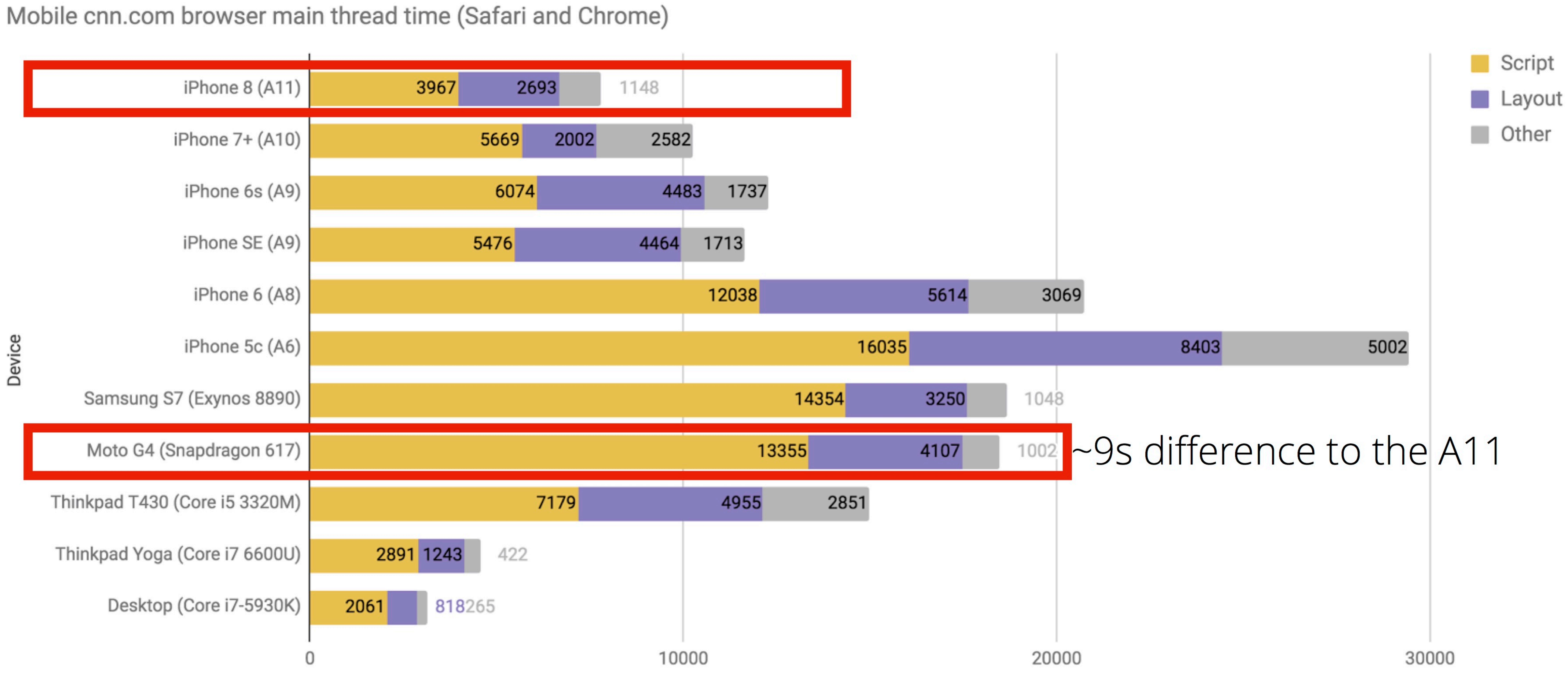
~1MB JavaScript (uncompressed)

2017 JavaScript Parse Costs



Average Phone

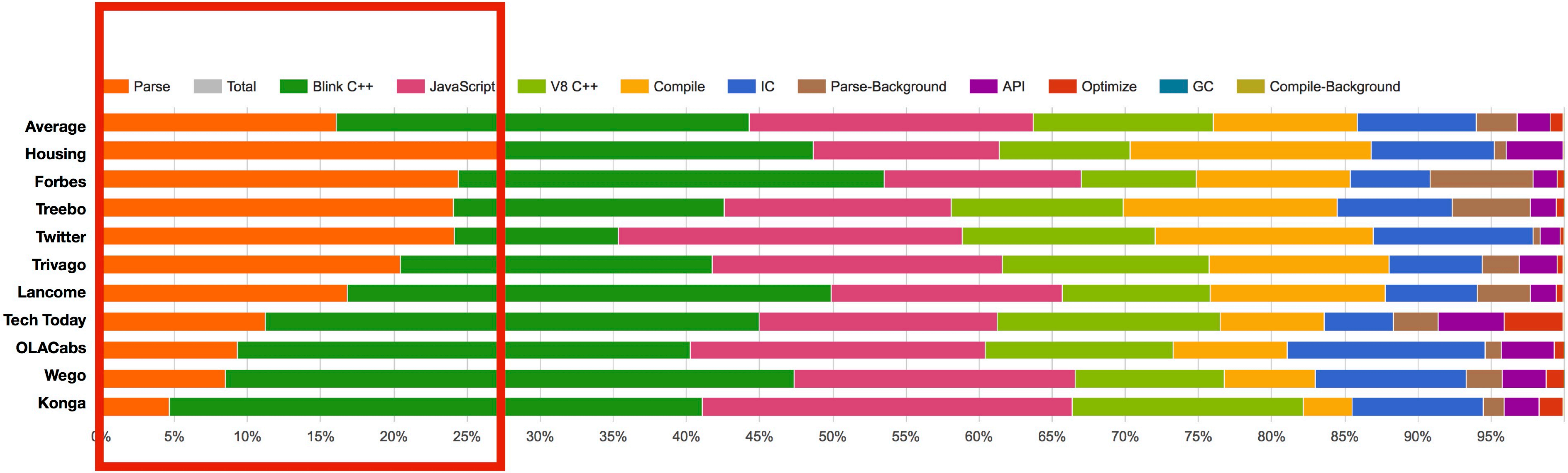
JavaScript Parse Cost On Mobile - CNN



~9s difference to the A11

With thanks to Pat Meenan

Where do mobile sites spend their time loading?



With thanks to Camillo and Mathias @ V8

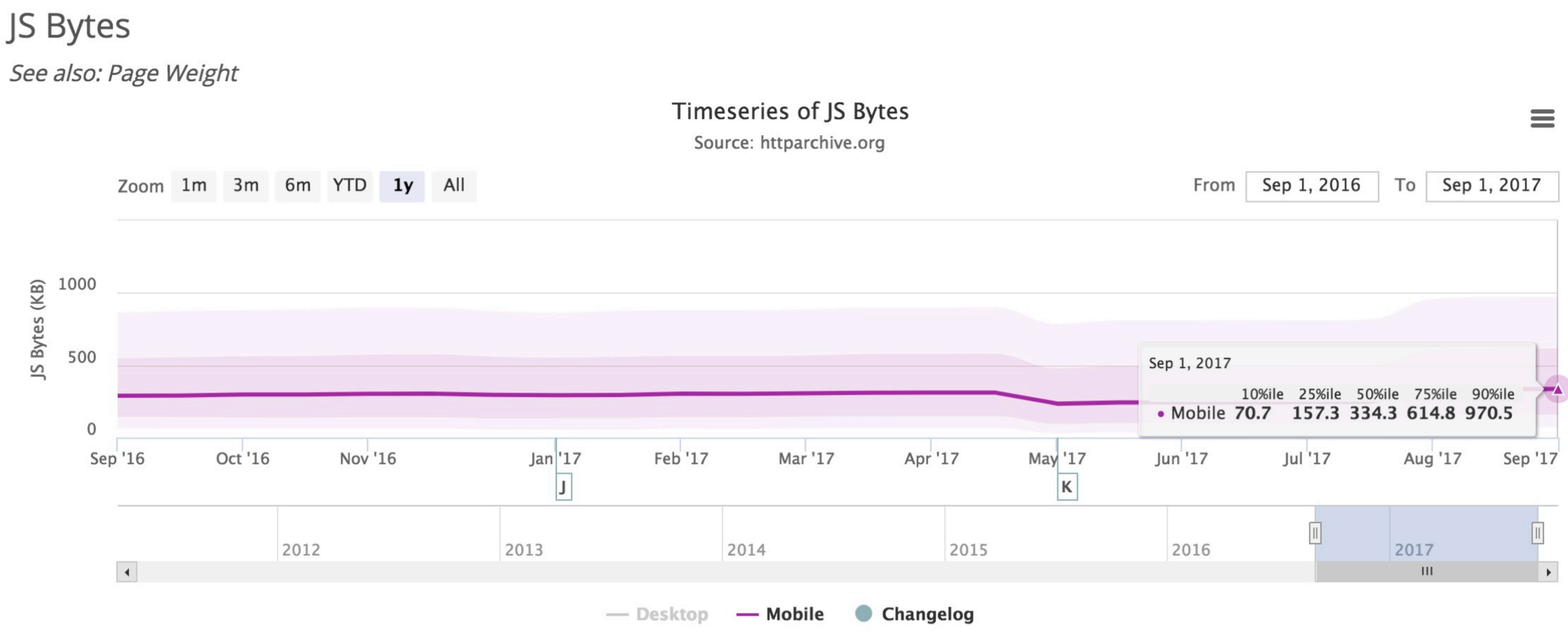
STATE OF JAVASCRIPT ON MOBILE

SITES ARE SENDING USERS...

P90: ~1MB

P75: 0.6MB

OF JS SPENDING ~4s ON PARSE/COMPILE



JS Bytes

See also: Page Weight

Timeseries of JS Bytes

Source: httparchive.org

Zoom 1m 3m 6m YTD **1y** All

From Sep 1, 2016 To Sep 1, 2017

Sep 1, 2017

	10%ile	25%ile	50%ile	75%ile	90%ile
• Mobile	70.7	157.3	334.3	614.8	970.5

http archive

Using Dev Tools mobile emulation, Moto G4 calibrated CPU, Cable (5/1mbps, 28ms)

http://beta.httparchive.org

WEB SPEED METRICS ON MOBILE

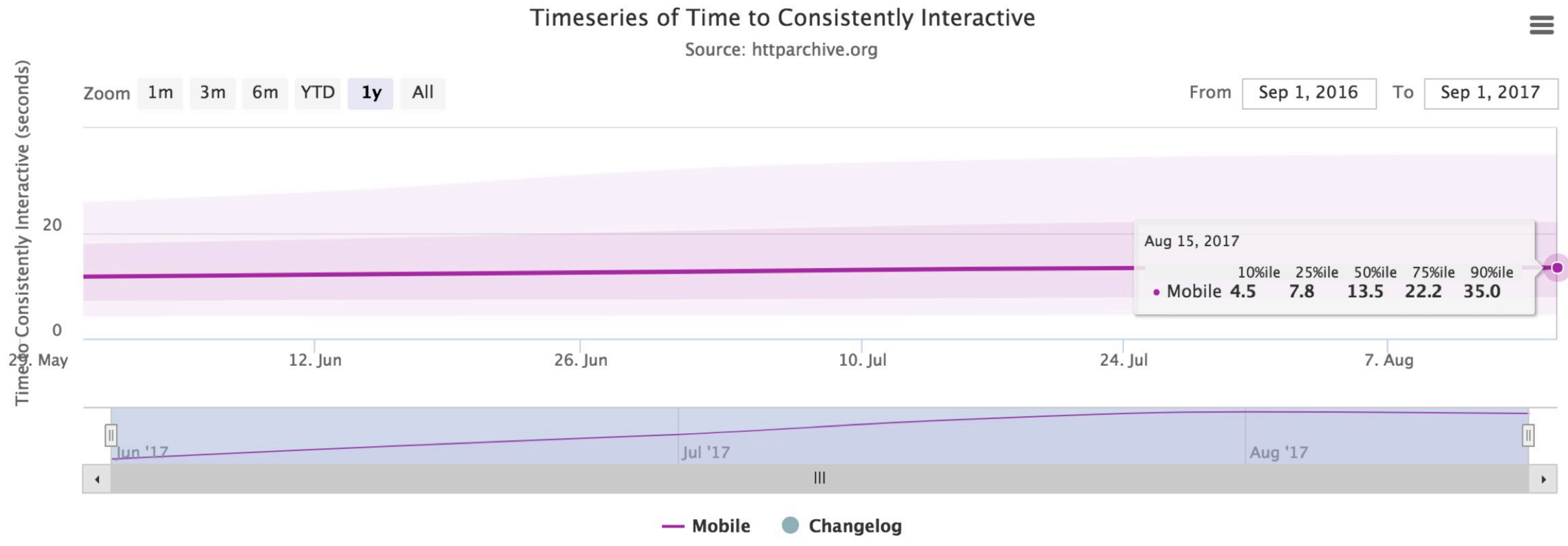
SITES ARE INTERACTIVE IN..

P90: 35s

P75: 22s

P90: 11s before First Meaningful Paint

Time to Consistently Interactive



Timeseries of Time to Consistently Interactive

Source: httparchive.org

Aug 15, 2017

	10%ile	25%ile	50%ile	75%ile	90%ile
● Mobile	4.5	7.8	13.5	22.2	35.0

http archive

Using Dev Tools mobile emulation, Moto G4 calibrated CPU, Cable (5/1mbps, 28ms)

http://beta.httparchive.org

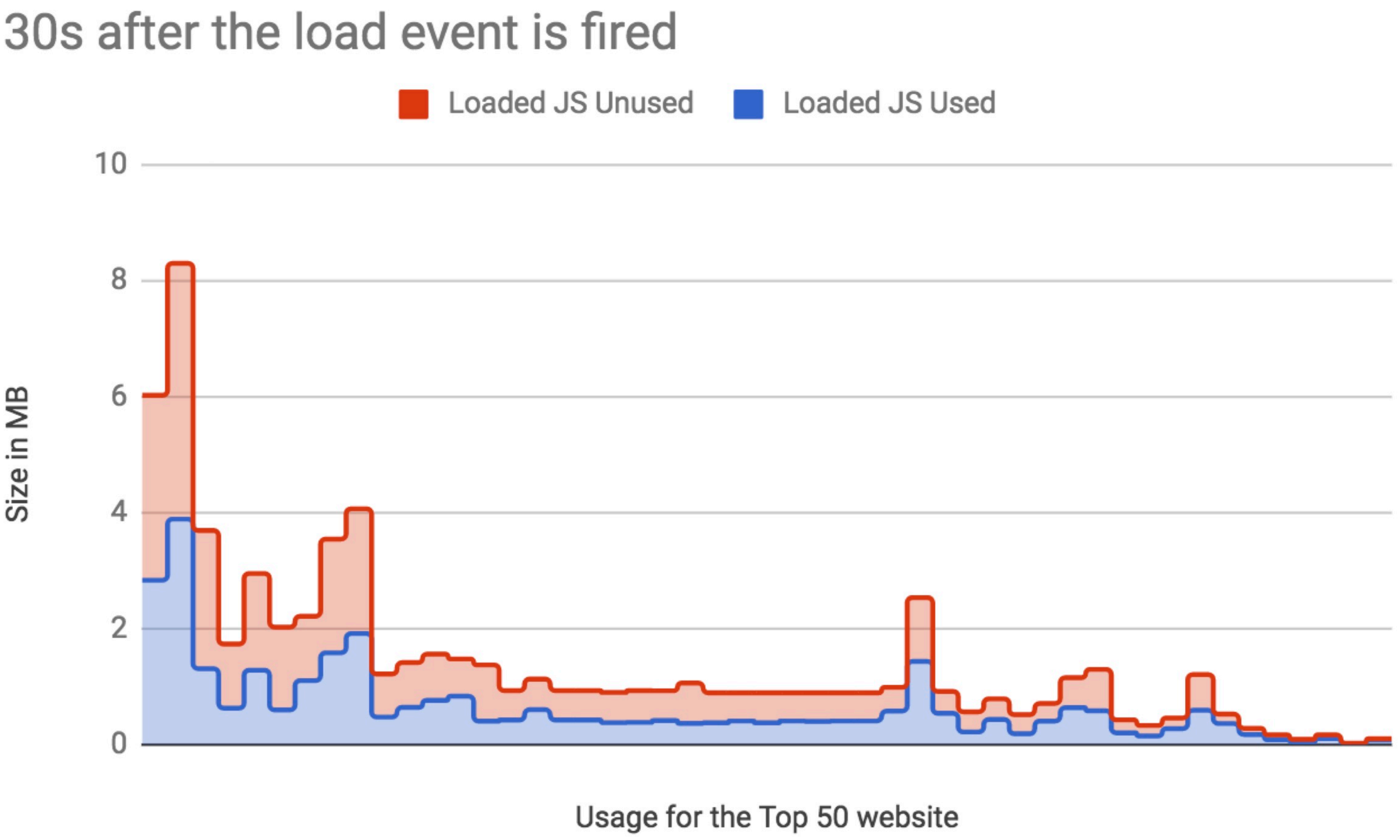
Is all of that **~1MB** used upfront?

JS CODE COVERAGE OF TOP 50 SITES

SITES MAY USE ONLY

40%

OF THE JAVASCRIPT THEY LOAD UPFRONT.



With thanks to fmeawad@chromium.org

STATE OF THE WEB ON MOBILE

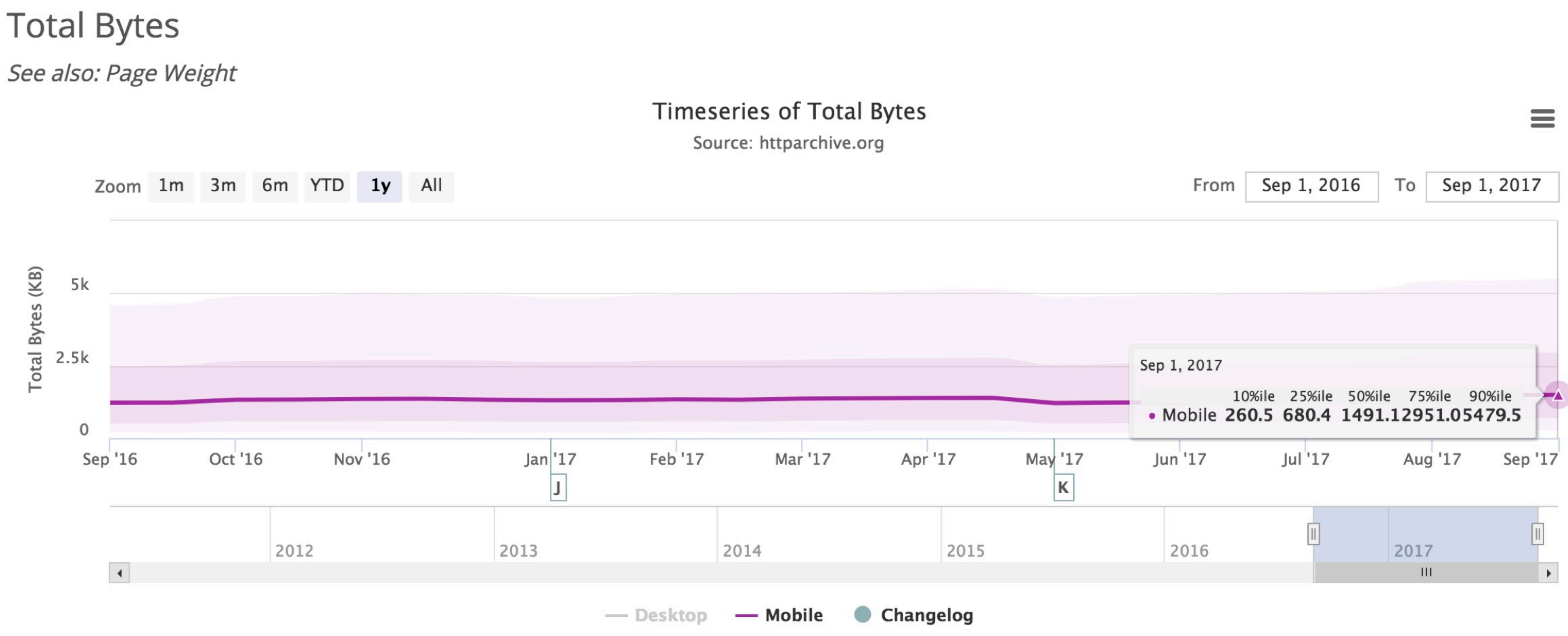
SITES ARE SENDING USERS...

P90: 5.4MB

P75: 2.9MB

P90 3.8MB (70%) of this is images

P90 1MB (18%) of this is JS



Total Bytes

See also: Page Weight

Timeseries of Total Bytes

Source: httparchive.org

Zoom 1m 3m 6m YTD **1y** All

From Sep 1, 2016 To Sep 1, 2017

Sep 1, 2017

	10%ile	25%ile	50%ile	75%ile	90%ile
• Mobile	260.5	680.4	1491.1	2951.0	5479.5

Using Dev Tools mobile emulation, Moto G4 calibrated CPU, Cable (5/1mbps, 28ms)

http://beta.httparchive.org

WEB SPEED METRICS ON MOBILE

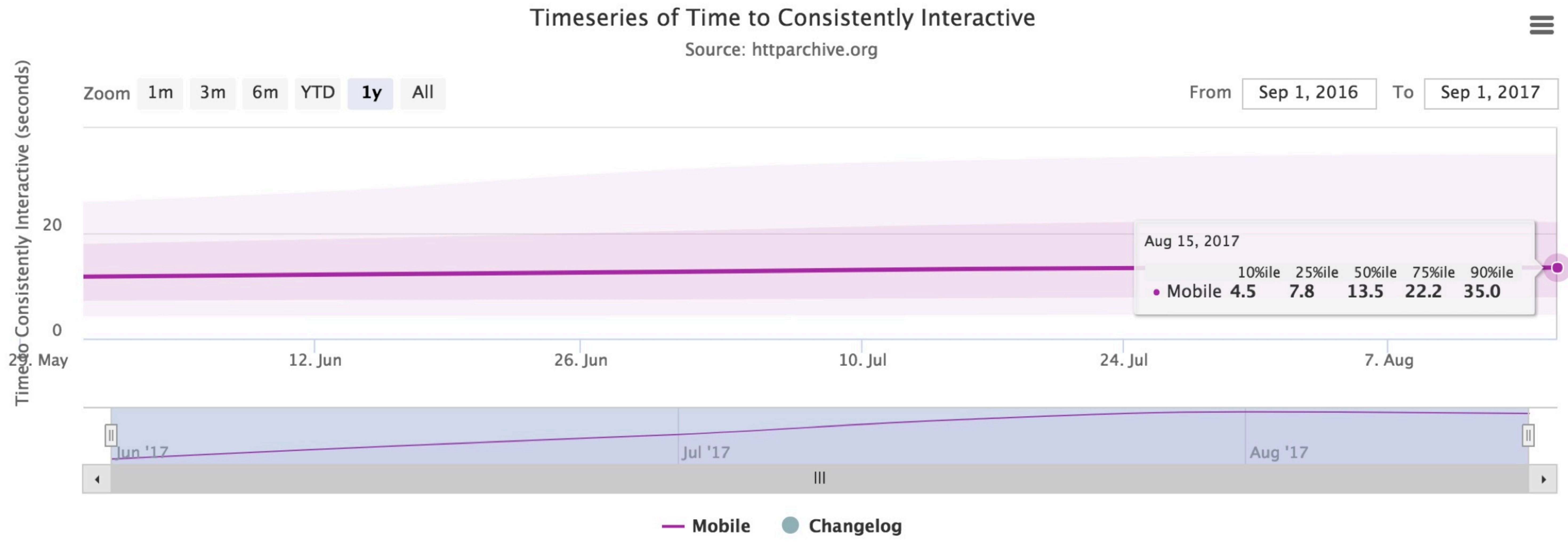
SITES ARE INTERACTIVE IN..

P90: 35s

P75: 22s

P90: 11s before First Meaningful Paint

Time to Consistently Interactive



http archive

Using Dev Tools mobile emulation, Moto G4 calibrated CPU, Cable (5/1mbps, 28ms)

http://beta.httparchive.org

```
var preload = document.createElement("link");
link.href= "myscript.js"
link.rel= "preload";
link.as= "script";
document.head.appendChild(link);
```

E.g. you could *request* the fetching of a resource because you know you’ll need it, but you don’t want to execute it yet.

```
var script = document.createElement("script");
script.src= "myscript.js"
document.body.appendChild(script);
```

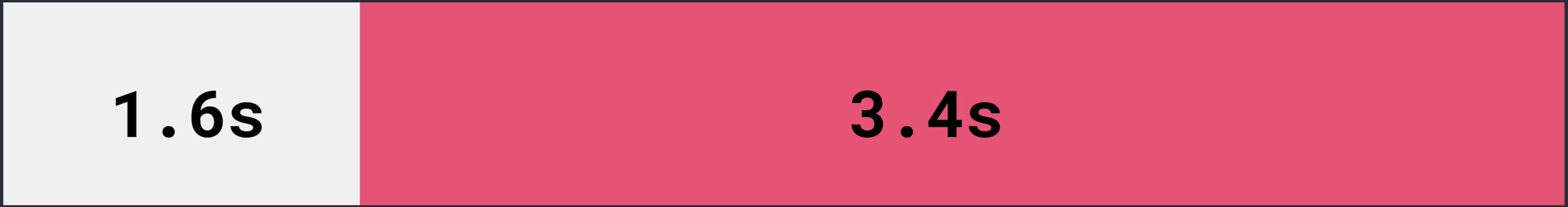
E.g. you could *request* the fetching of a resource because you know you’ll need it, but you don’t want to execute it yet.

```
<link rel="preload" as="image"
      href="map.png" media="(max-width: 600px)">
<link rel="preload" as="script"
      href="map.js" media="(min-width: 601px)">
```

E.g. you could load assets *conditionally* (e.g. a static map on smaller screens, and an interactive map on large screens).

Time to Interactive Budget

5s


1.6s

3.4s

DNS LOOKUP
TCP HANDSHAKE
HTTPS HANDSHAKE

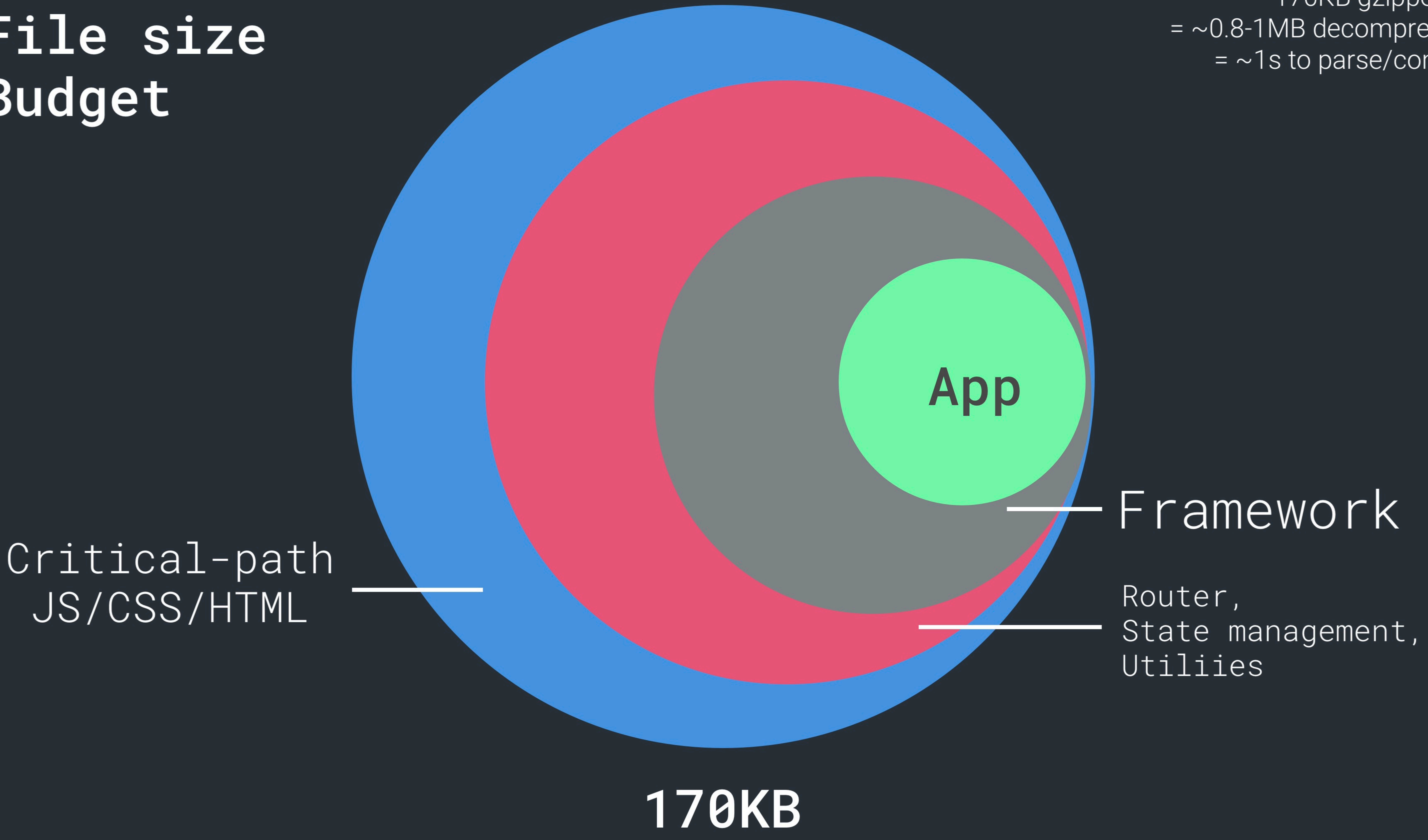
At 400Kbps we can send 3.4 x 50KB = 170KB

Baseline is ~$200 Android phone
On a slow 3G network, emulated at:

400ms RTT, 400Kbps transfer

File size Budget

170KB gzipped JS
= ~0.8-1MB decompressed
= ~1s to parse/compile



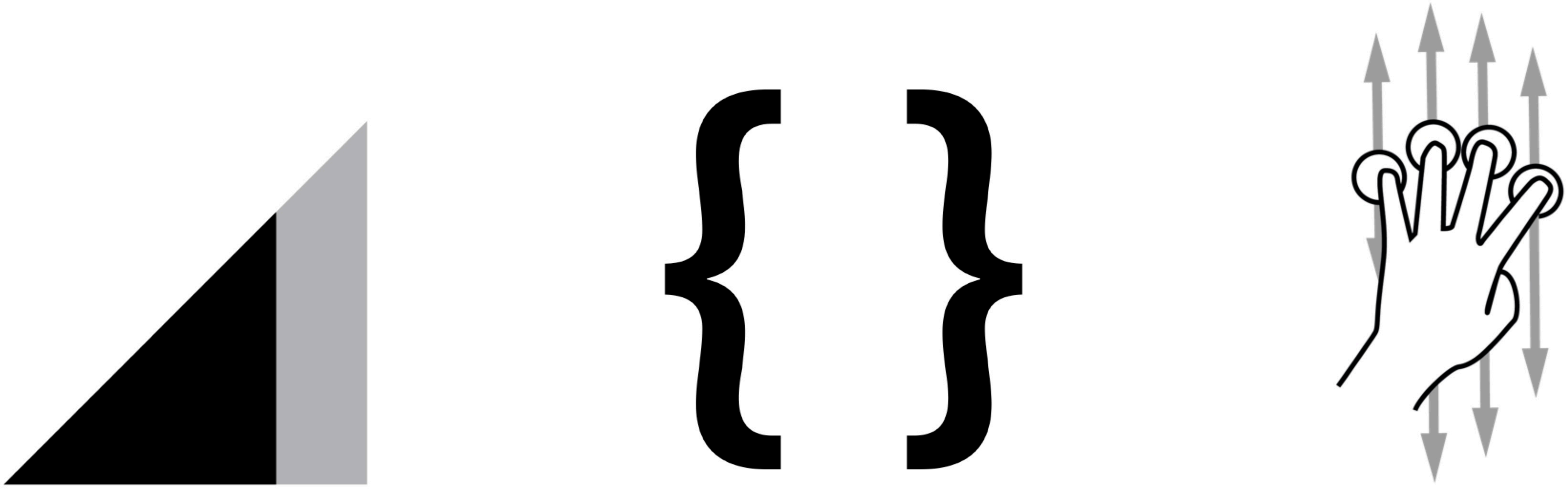
All bytes are not equal

JS != JPG

170KB JS

170KB JPEG

Evaluating the performance of Web Frameworks



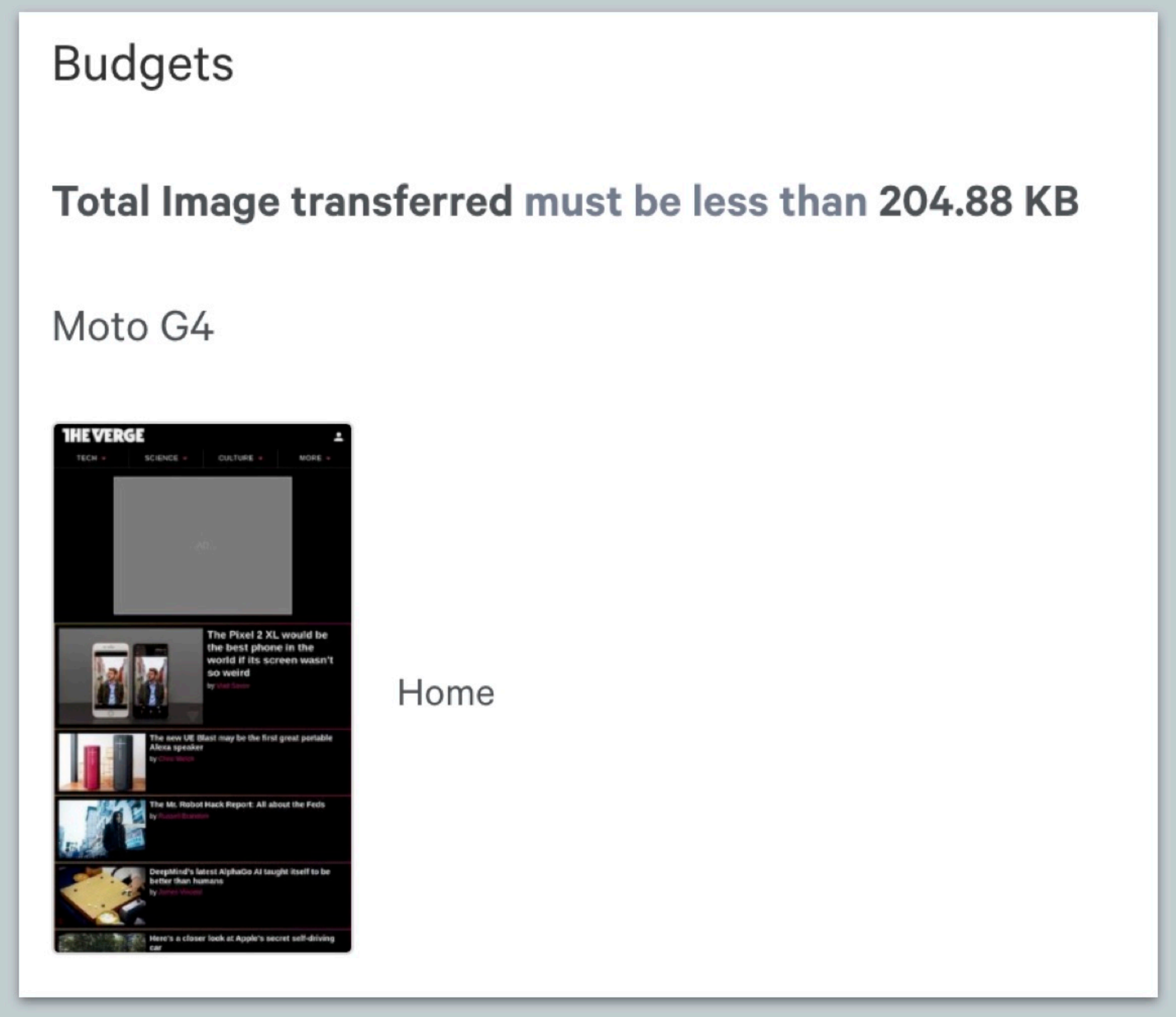
NETWORK TRANSFER

PARSE/COMPILE

RUNTIME COST

https://twitter.com/kristoferbaxter/status/908144931125858304

Performance Budget Tools



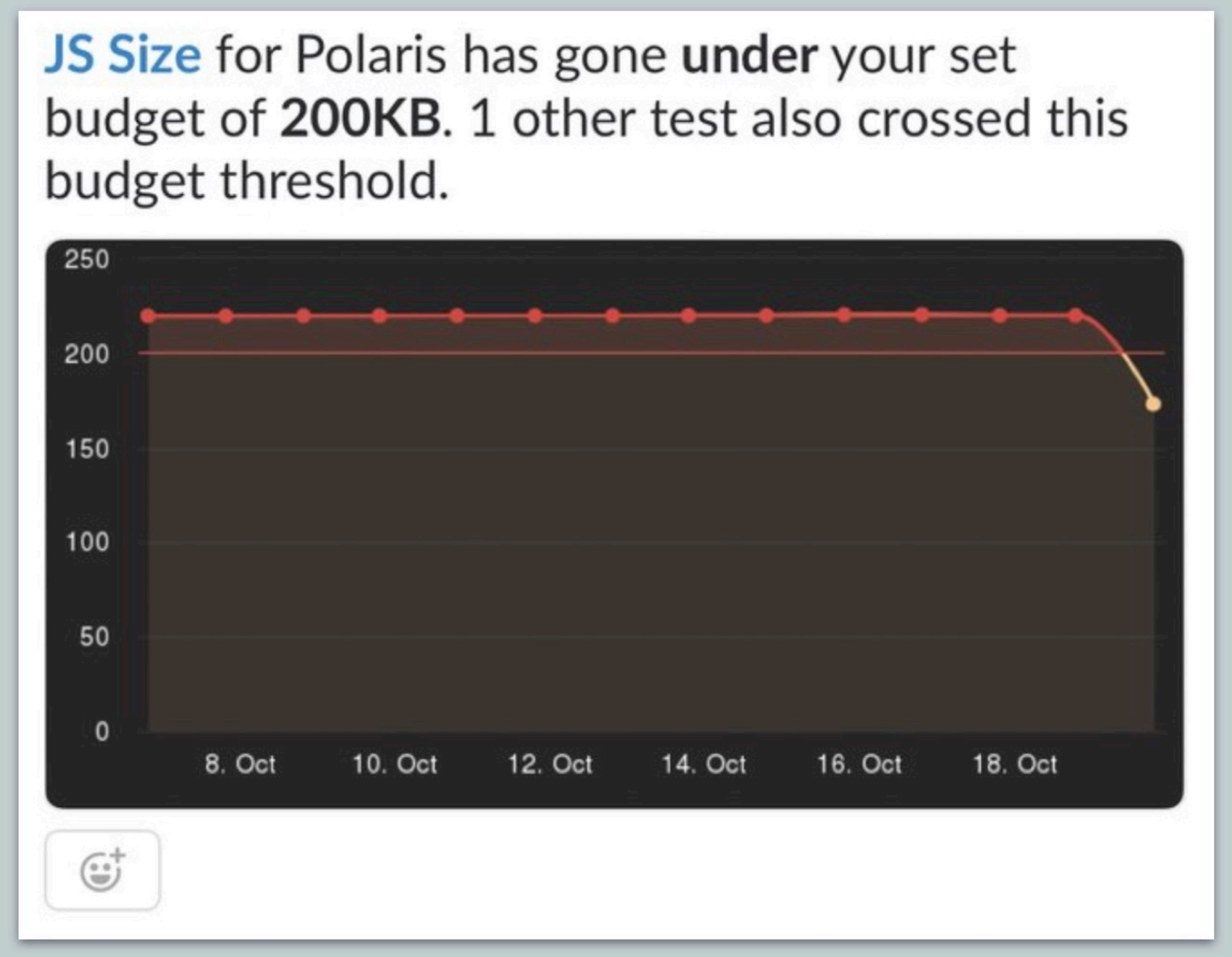
Budgets

Total Image transferred must be less than **204.88 KB**

Moto G4

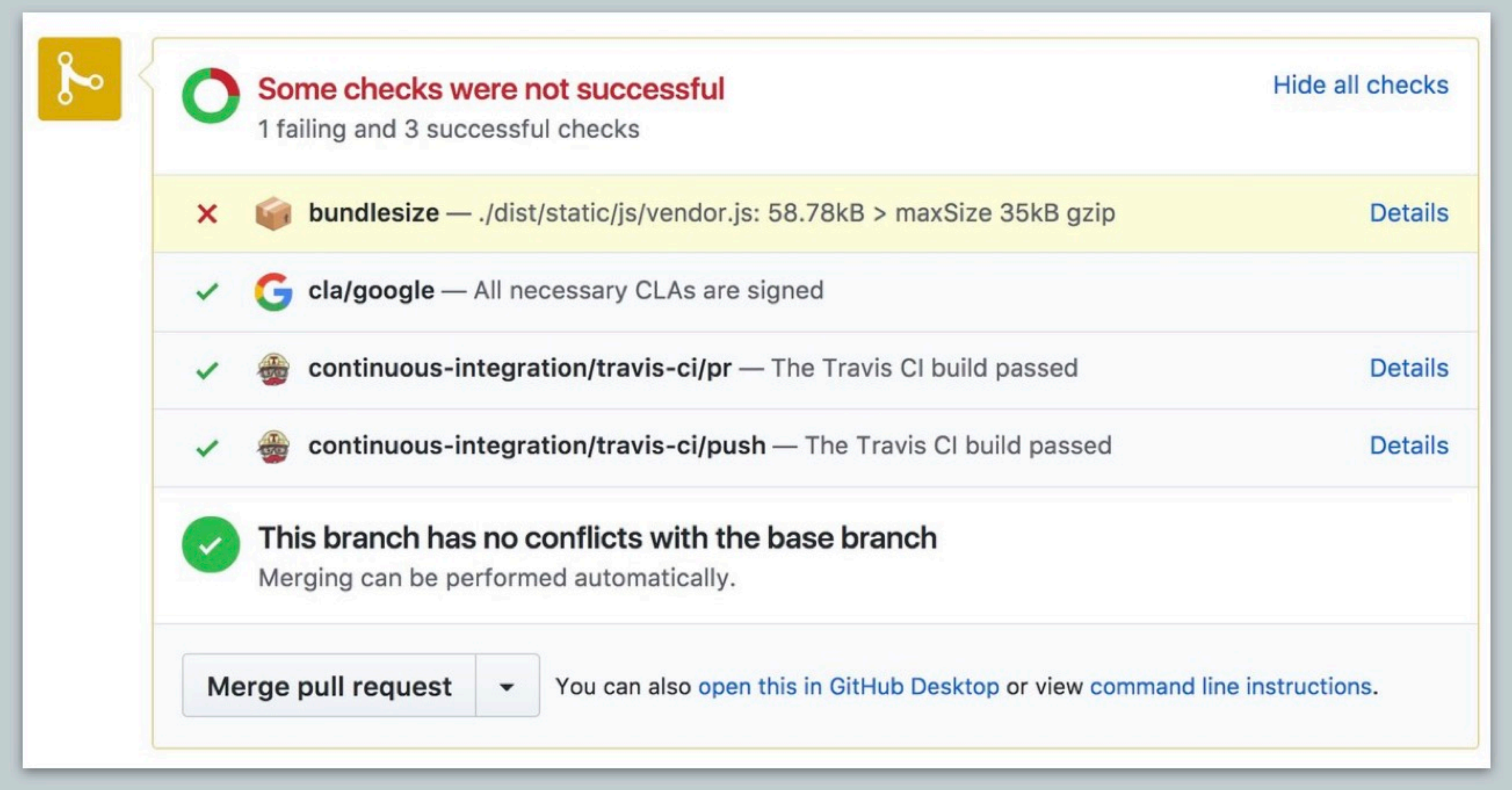
Home

CALIBRE



JS Size for Polaris has gone **under** your set budget of **200KB**. 1 other test also crossed this budget threshold.

SPEEDCURVE



Some checks were not successful

1 failing and 3 successful checks

Hide all checks

✗ **bundlesize** — ./dist/static/js/vendor.js: 58.78kB > maxSize 35kB gzip Details

✓ **cla/google** — All necessary CLAs are signed

✓ **continuous-integration/travis-ci/pr** — The Travis CI build passed Details

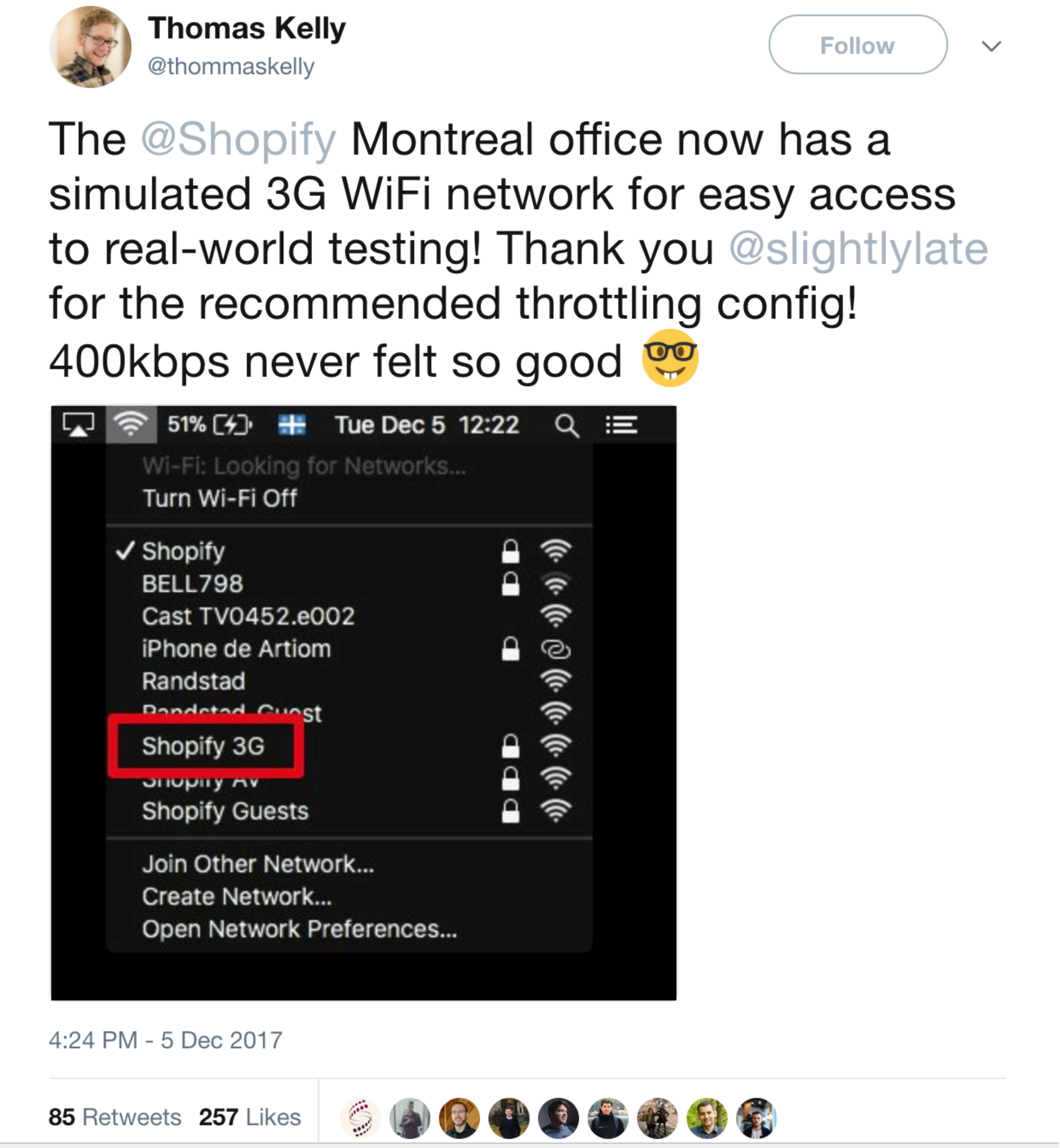
✓ **continuous-integration/travis-ci/push** — The Travis CI build passed Details

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request ▾ You can also open this in GitHub Desktop or view command line instructions.

BUNDLESIZE



Thomas Kelly
@thommaskelly

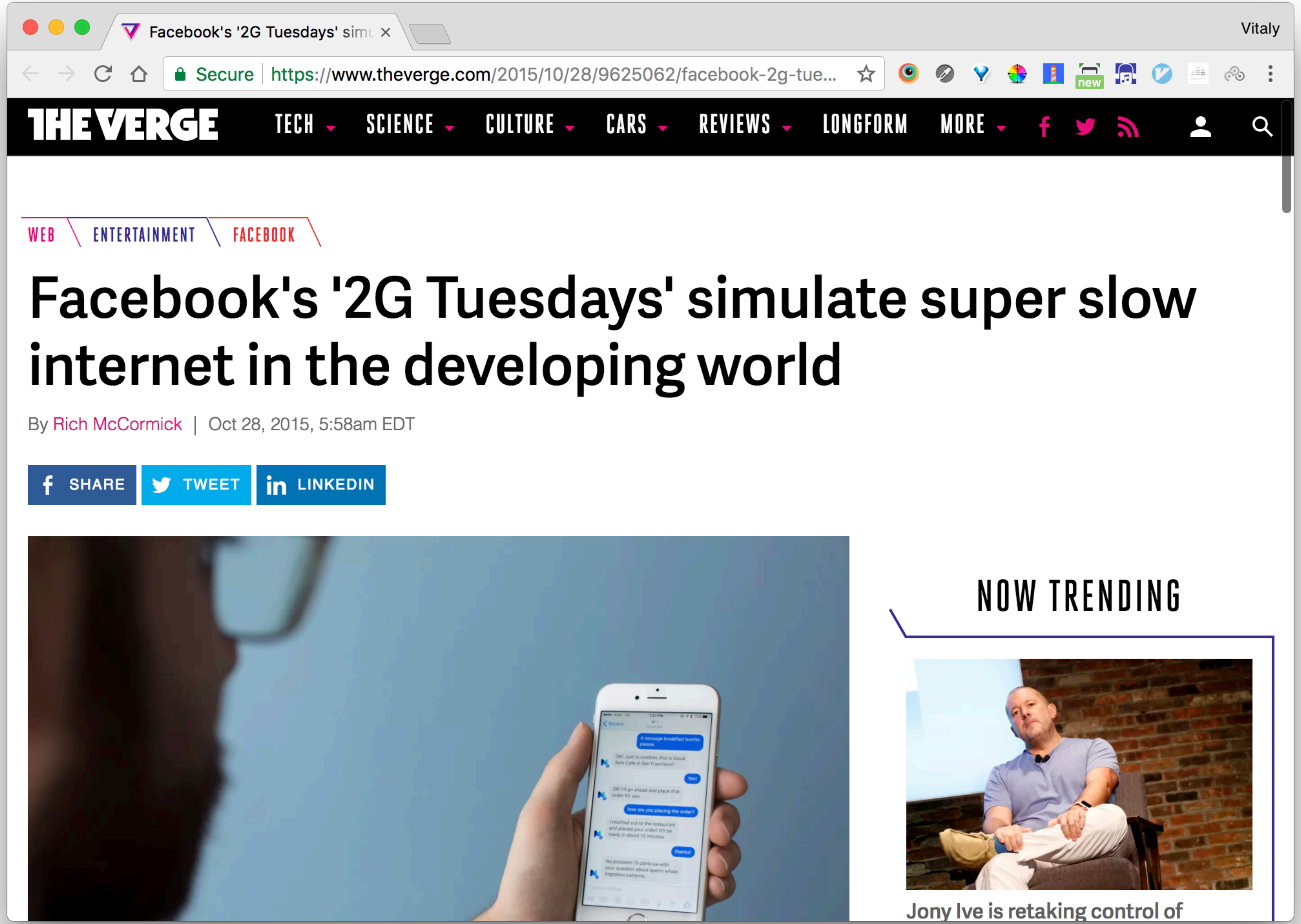
Follow

The @Shopify Montreal office now has a simulated 3G WiFi network for easy access to real-world testing! Thank you @slightlylate for the recommended throttling config! 400kbps never felt so good 🤓

4:24 PM - 5 Dec 2017

85 Retweets 257 Likes

TUES
2G



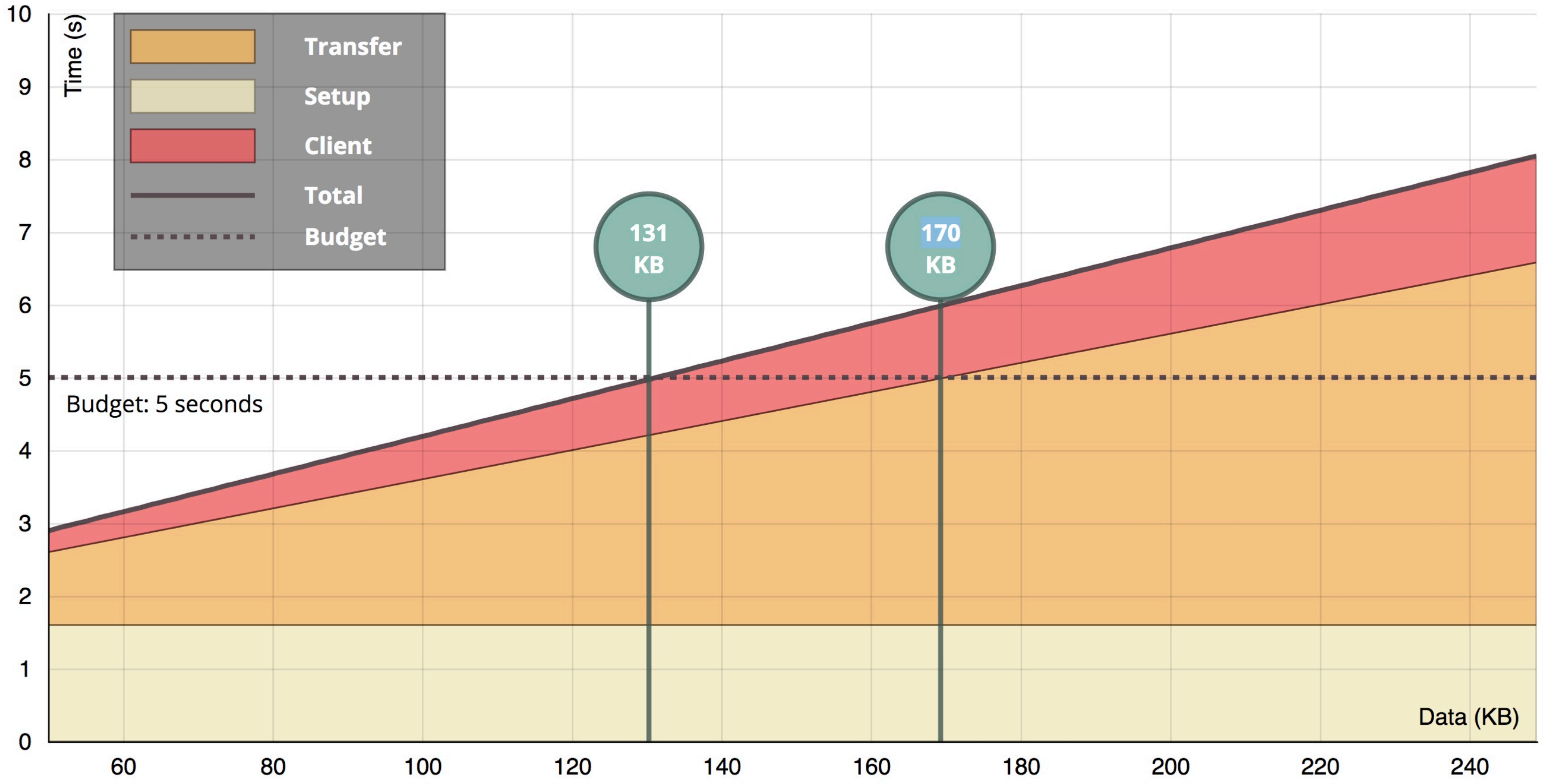
Facebook's '2G Tuesdays' simulate super slow internet in the developing world

By Rich McCormick | Oct 28, 2015, 5:58am EDT

NOW TRENDING

Jony Ive is retaking control of

REAL-WORLD WEB PERF BUDGETS



bit.ly/perf-budgets

Modern

LOADING BEST PRACTICES

Pinterest tinder™

@addyosmani

WRAPPING UP

Front-End Performance Checklist 2017

Below you'll find an overview of the ***front-end performance issues*** *you might need to consider to ensure that your response times are fast and smooth.*

Get ready and set goals

□ **Be 20% faster than your fastest competitor.**
Measure "start rendering" (WebPageTest) and "first meaningful paint" times (Lighthouse) on a Moto G, a mid-range Samsung device and a good middle-of-the-road device like the Nexus 4, preferably in an open device lab — on regular 3G, 4G and Wi-Fi connections. Collect data, set up a spreadsheet, shave off 20%, and set up your goals (performance budgets).

□ **Share the checklist with your colleagues.**
Make sure that the checklist is familiar to every member of your team. Every decision has performance implications, and your project would hugely benefit from front-end developers being actively involved. Map design decisions against the performance budget.

□ **100-millisecond response time, 60 frames per second.**
Each frame of animation should complete in less than 16 milliseconds — ideally 10 milliseconds, thereby achieving 60 frames per second (1 second ÷ 60 = 16.6 milliseconds). Be optimistic and use the idle time wisely. For high pressure points like animation, it's best to do nothing else where you can and the absolute minimum where you can't.

Define the environment

□ **Choose and set up your build tools.**
Don’t pay much attention to what’s supposedly cool. As long as you are getting results fast and you have no issues maintaining your build process, you’re doing just fine.

□ **Progressive enhancement.**
Design and build the core experience first, and then enhance the experience with advanced features for capable browsers, creating resilient experiences. If your website runs fast on a slow machine with a poor screen in a poor browser on a suboptimal network, then it will only run faster on a fast machine with a good browser on a decent network.

□ **Pick your battles wisely: Angular, React, Ember and co.**
Favor a framework that enables server-side rendering. Be sure to measure boot times in server- and client-rendered modes on mobile devices before settling on a framework. Understand the nuts and bolts of the framework you’ll be relying on. When building web apps, look into the PRPL pattern and application shell architecture.

□ **Google’s AMP or Facebook’s Instant Articles?**
You can achieve good performance without them, but AMP does provide a solid performance framework, with a free CDN, while Instant Articles will boost your performance on Facebook. You could build progressive web AMPs, too.

□ **Choose your CDN wisely.**
Depending on how much dynamic data you have, you might be able to “outsource” some part of the content to a static site generator, push it to a CDN and serve a static version from it, thus avoiding database requests (JAMStack). Double-check that your CDN performs content compression and conversion, smart HTTP/2 delivery and edge-side includes for you.

□ **Use the “cutting-the-mustard” technique.**
Send the core experience to legacy browsers and an enhanced experience to modern browsers. Be strict in the loading of assets: load the core immediately, enhancements on *DomContentLoaded* and extras on the *Load* event.

□ **Consider micro-optimizations and progressive booting.**
You might need some time to initialize the app before you can render the page. Your goal: Use server-side rendering to get a quick first meaningful paint, but also include some minimal JavaScript to keep the time-to-interactive close to the first meaningful paint. Then, either on demand or as time allows, boot non-essential parts of the app. Display skeleton screens instead of loading indicators. Use tree-shaking, code-splitting and an ahead-of-time compiler to offload some of the client-side rendering to the server.

□ **Are HTTP cache headers set properly?**
Double-check that expires, cache-control, max-age and other HTTP cache headers are set properly. In general, resources should be cacheable either for a very short time (if they are likely to change) or indefinitely (if they are static). Use *cache-control: immutable*, designed for fingerprinted static resources, to avoid revalidation.

□ **Limit third-party libraries, and load JavaScript asynchronously.**
As developers, we have to explicitly tell the browser not to wait and to start rendering the page with the *defer* and *async* attributes in HTML. If you don’t have to worry much about IE 9 and below, then prefer *defer* to *async;* otherwise, use *async*. Use static social-sharing buttons and static links to interactive maps, instead of relying on third-party libraries.

□ **Are images properly optimized?**
Optimize images. As far as possible, use responsive images with *srcset*, *sizes* and the <*picture*> element. Make use of the WebP format, by serving WebP images with <*picture*> and a JPEG fallback or by using content negotiation (using *Accept* headers). For critical images, use

□ **Push critical CSS quickly.**
Collect all of the CSS required to start rendering the first visible portion of the page ("critical CSS" or "above-the-fold" CSS), and add it inline in the *<head>* of the page. Consider the conditional inlining approach. Alternatively, use HTTP/2 server push, but then you might need to create a cache-aware HTTP/2 server-push mechanism.

□ **Use tree-shaking and code-splitting to reduce payloads.**
Tree-shaking is a way to clean up your build process by only including code that is actually used in production. Code-splitting splits your code base into "chunks" that are loaded on demand. Make use of both via WebPack. Also, use Rollup as a JavaScript module bundler.

□ **Improve rendering performance.**
Isolate expensive components with CSS containment. Make sure that there is no lag when scrolling the page or when an element is animated, and that you're consistently hitting 60 frames per second. If that's not possible, then making the frames per second consistent is at least preferable to a mixed range of 60 to 15. Use CSS *will-change* to inform the browser about which elements will change.

□ **Warm up the connection to speed up delivery.**
Use skeleton screens, and lazy-load all expensive components, such as fonts, JavaScript, carousels, videos and iframes. Use resource hints to save time on *dns-prefetch*, *preconnect*, *prefetch*, *pretender* and *preload*.

HTTP/2

□ **Get ready for HTTP/2.**
HTTP/2 is supported very well and offers a performance boost. It isn't going anywhere, and in most cases, you're better off with the latter. The downsides are that you'll have to migrate to HTTPS, and depending on how large your HTTP/1.1 user base is (users on legacy OS' or with

and check your certificate. Make sure that all external plugins and tracking scripts are loaded via HTTPS, that cross-site scripting isn't possible and that both HTTP Strict Transport Security headers and Content Security Policy headers are properly set.

□ **Do your servers and CDNs support HTTP/2?**
Different servers and CDNs are probably going to support HTTP/2 differently. Use *Is TLS Fast Yet?* to check your options, or quickly look up how your servers are performing and which features you can expect to be supported.

□ **Is Brotli or Zopfli compression in use?**
Brotli, a new lossless data format, is widely supported in Chrome, Firefox and Opera. It's more effective than Gzip and Deflate (HTTPS only). The catch: Brotli doesn't come preinstalled on most servers today, and it's not easy to set up without self-compiling NGINX or Ubuntu. Alternatively, you can look into using Zopfli on resources that don't change much — it encodes data to Deflate, Gzip and Zlib formats and is designed to be compressed once and downloaded many times.

□ **Is OCSP stapling enabled?**
By enabling OCSP stapling on your server, you can speed up TLS handshakes. The OCSP protocol does not require the browser to spend time downloading and then searching a list for certificate information, hence reducing the time required for a handshake.

□ **Have you adopted IPv6 yet?**
Studies show that IPv6 makes websites 10 to 15% faster due to neighbor discovery (NDP) and route optimization. Update the DNS for IPv6 to stay bulletproof for the future. Just make sure that dual-stack support is provided across the network — it allows IPv6 and IPv4 to run simultaneously alongside each other. After all, IPv6 is not backwards-compatible.

☐ **Have you tested in proxy browsers and legacy browsers?**
Testing in Chrome and Firefox is not enough. Look into how your website works in proxy browsers and legacy browsers (including UC Browser and Opera Mini). Measure average Internet speed in your countries of interest to avoid big surprises. Test with network throttling, and emulate a high-DPI device. BrowserStack is fantastic, but test on real devices as well.

☐ **Is continuous monitoring set up?**
Having a private instance of WebPagetest is always beneficial for quick and unlimited tests. Set up continuous monitoring of performance budgets with automatic alerts. Set your own user-timing marks to measure and monitor business-specific metrics. Look into SpeedTracker, Lighthouse and Calibre.

Quick wins

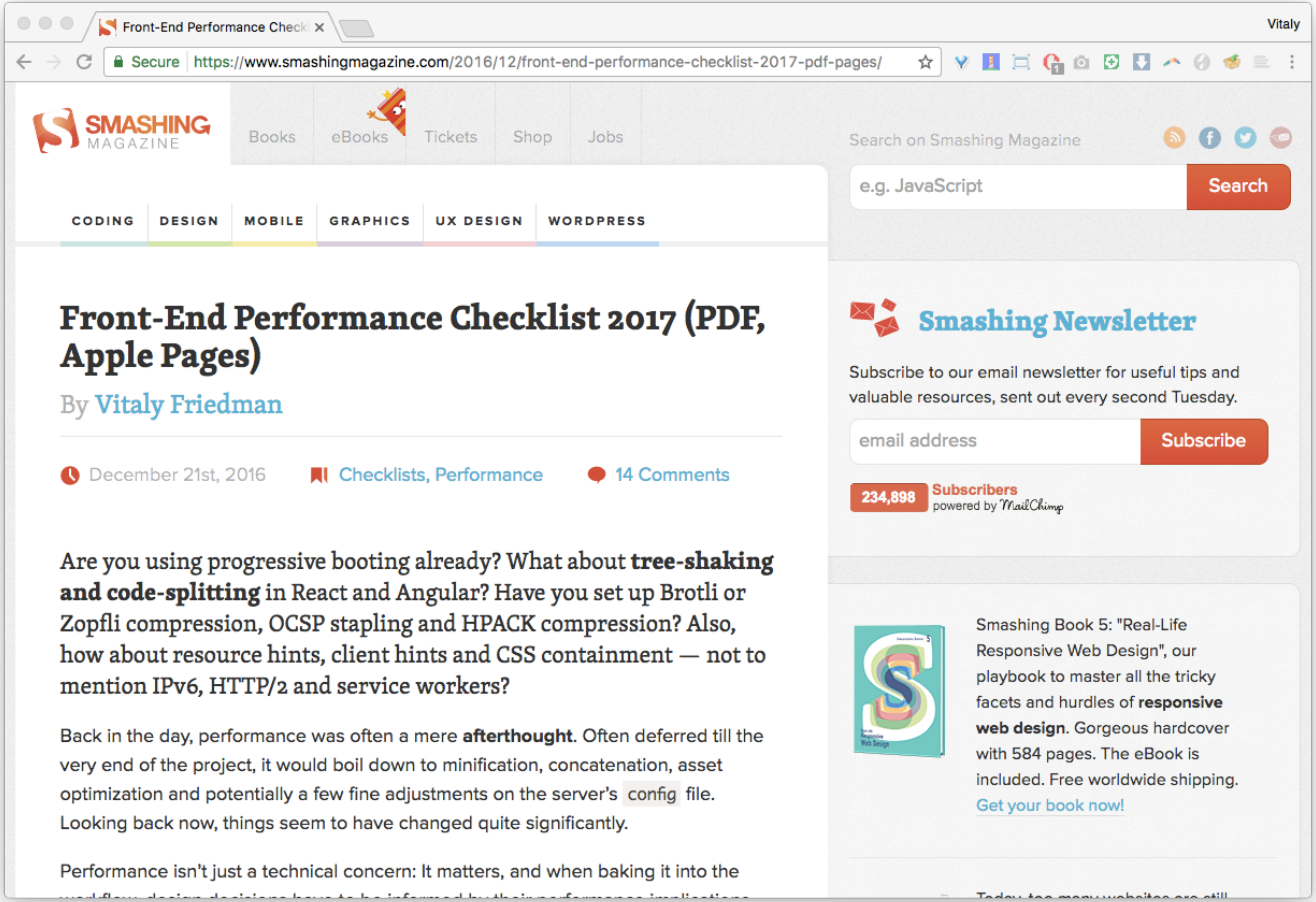
This list is quite comprehensive, and completing all of the optimizations might take quite a while. So if you had just 1 hour to get significant improvements, what would you do? Let's boil it all down to 10 low-hanging fruits. Obviously, before you start and once you finish, measure results, including start rendering time and SpeedIndex on 3G and cable connections.

1. Your goal is a start rendering time under 1 second on cable and 3 seconds on 3G, and a SpeedIndex value under 1000. Optimize for start rendering time and time-to-interactive.
2. Prepare critical CSS for your main templates, and include it in the *<head>* of the page. (Your budget is 14 KB.)
3. Defer and lazy-load as many scripts as possible, both your own and third-party scripts — especially social media buttons, video players and expensive JavaScript.
4. Add resource hints to speed up delivery with faster *dns-lookup*, *preconnect*, *prefetch*, *preload* and

prerender.

5. Subset web fonts, and load them asynchronously (or just switch to system fonts instead).
6. Optimize images, and consider using WebP for critical pages (such as landing pages).
7. Check that HTTP cache headers and security headers are set properly.
8. Enable Brotli or Zopfli compression on the server. (If that's not possible, don't forget to enable Gzip compression.)
9. If HTTP/2 is available, enable HPACK compression, and start monitoring mixed-content warnings. If you're running over LTS, also enable OCSP stapling.
10. If possible, cache assets such as fonts, styles, JavaScript and images — actually, as much as possible! — in a service worker cache.

Huge thanks to Anselm Hannemann, Patrick Hamann, Addy Osmani, Andy Davies, Tim Kadlec, Yoav Weiss, Rey Bango, Mariana Peralta, Jacob Groß, Tim Swalling, Bob Visser, Kev Adamson and Rodney Rehm for reviewing, as well as our fantastic community for sharing insights for everybody to use. You are truly smashing.


Front-End Performance Check
Vitaly
Secure
https://www.smashingmagazine.com/2016/12/front-end-performance-checklist-2017-pdf-pages/
SMASHING MAGAZINE
Books
eBooks
Tickets
Shop
Jobs
Search on Smashing Magazine
e.g. JavaScript
Search
CODING
DESIGN
MOBILE
GRAPHICS
UX DESIGN
WORDPRESS

Front-End Performance Checklist 2017 (PDF, Apple Pages)

By Vitaly Friedman

December 21st, 2016 Checklists, Performance 14 Comments

Are you using progressive booting already? What about **tree-shaking and code-splitting** in React and Angular? Have you set up Brotli or Zopfli compression, OCSP stapling and HPACK compression? Also, how about resource hints, client hints and CSS containment — not to mention IPv6, HTTP/2 and service workers?

Back in the day, performance was often a mere **afterthought**. Often deferred till the very end of the project, it would boil down to minification, concatenation, asset optimization and potentially a few fine adjustments on the server's `config` file. Looking back now, things seem to have changed quite significantly.

Performance isn't just a technical concern: It matters, and when baking it into the workflow, design decisions have to be informed by their performance implications.

THE END

CALCULATE YOUR SCORE.

THANKS FOR PLAYING. _