To push, or not to push?!

A journey of resource loading in the browser

Fluent Conference, June 2018

Patrick Hamann
@patrickhamann

fastly®

The New York Times
reddit
airbnb
deliveroo
ATRESMEDIA
stripe
lonely planet
Pinterest
La Redoute
wallapop
wayfair
a zillion things home
ticketmaster®
the guardian
imgur
R Rakuten.com
Boots
A&E
GitHub
SHAZAM®
FAST COMPANY
BBC
CONDÉ NAST
BuzzFeed
RollingStone
BUSINESS INSIDER
KICKSTARTER
Hotel Tonight
FOURSQUARE
W
WENNER MEDIA
New Relic.
imgIX
GOV.UK
vimeo
shopify
marfeel
KAYAK
Etsy

Why?

> “HTTP/2 will solve this”

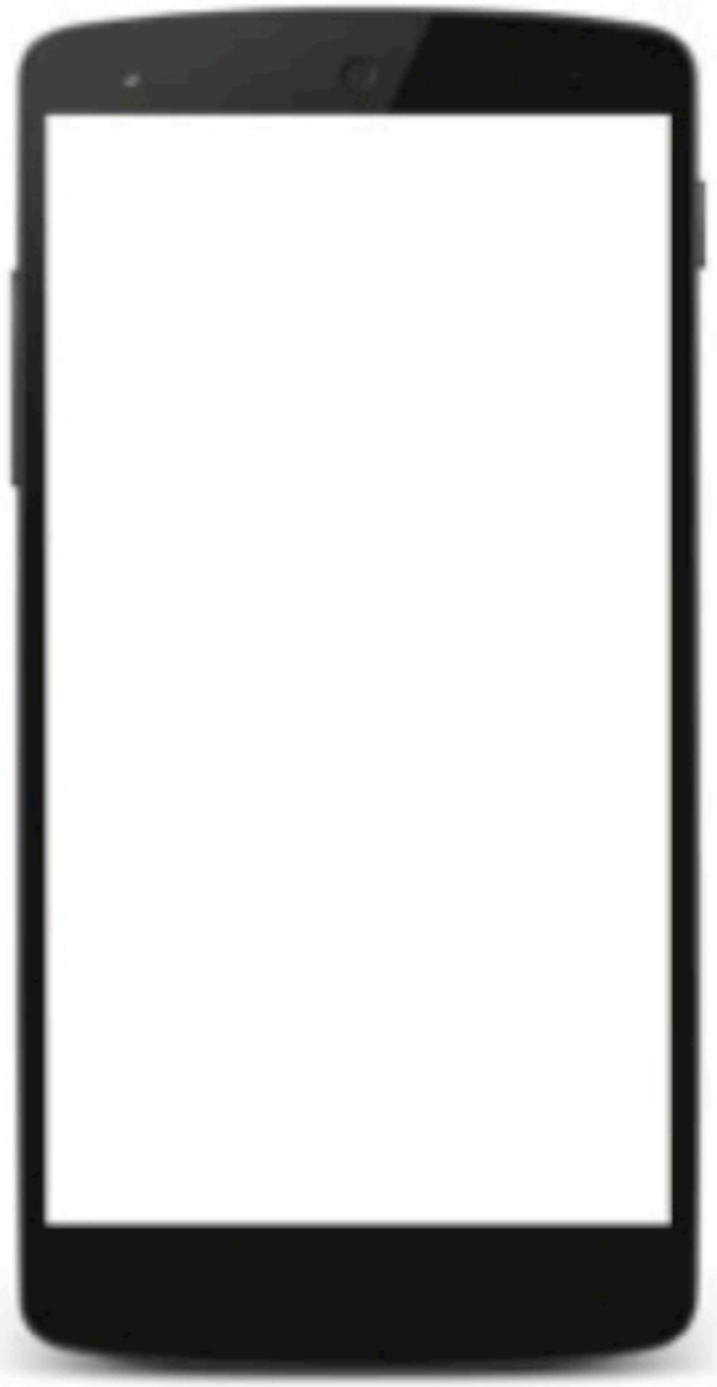
– Everybody

Resource loading in the browser is hard.

Resource loading is hard:

- ⏱️ Performance is tightly coupled to latency
- 🤝 Connection cost is high
- 📈 Congestion control is unavoidable
- 🙈 Critical resources can be hidden
- 💤 Bandwidth is often under-utilised
- ⚠️ Script execution is expensive

How can we load our resources most efficiently?



> A critical request is one that contains an asset that is essential to the content within the users viewport.

– Ben Schwarz, Calibre

What are my critical resources?

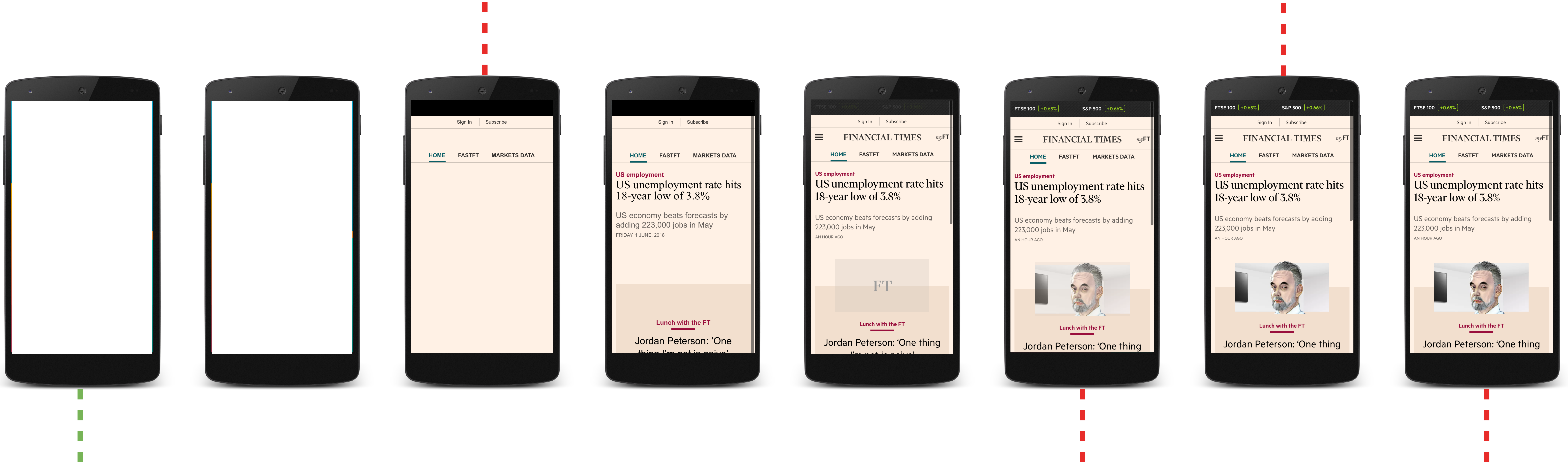
✅ Critical CSS for current route

✅ Fonts

✅ Hero images

✅ Initial application route

✅ Application bootstrap data



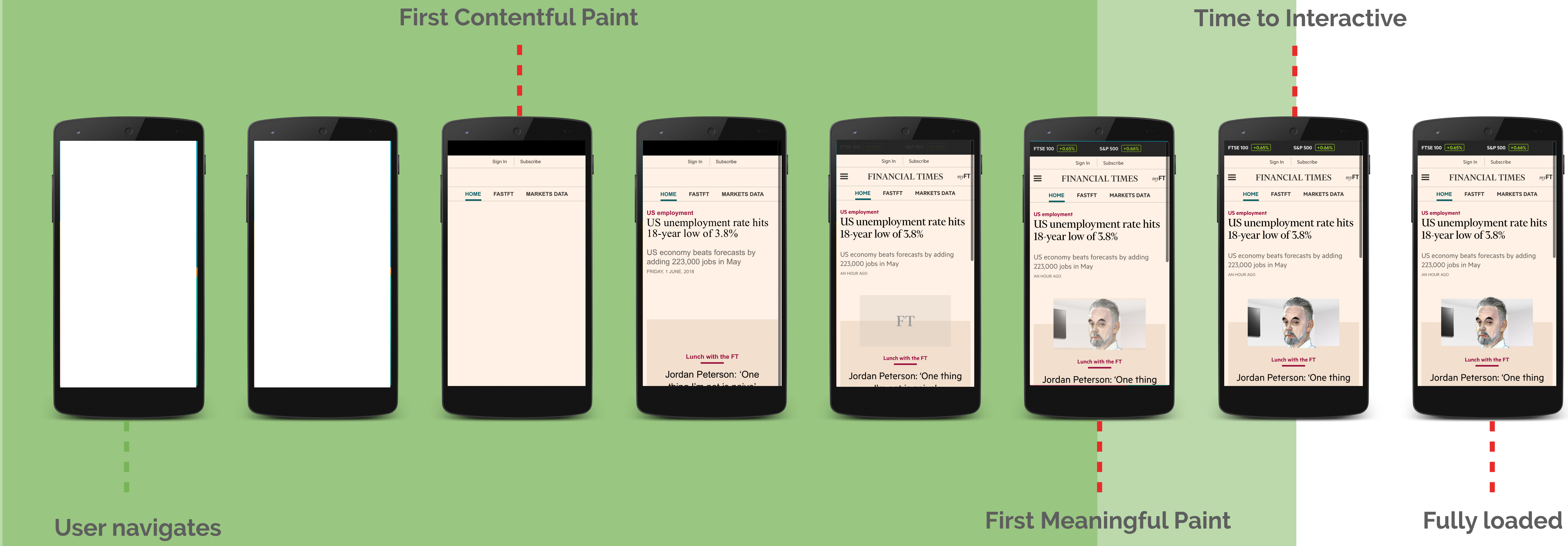
User navigates

First Contentful Paint

First Meaningful Paint

Time to Interactive

Fully loaded



User navigates

First Contentful Paint

First Meaningful Paint

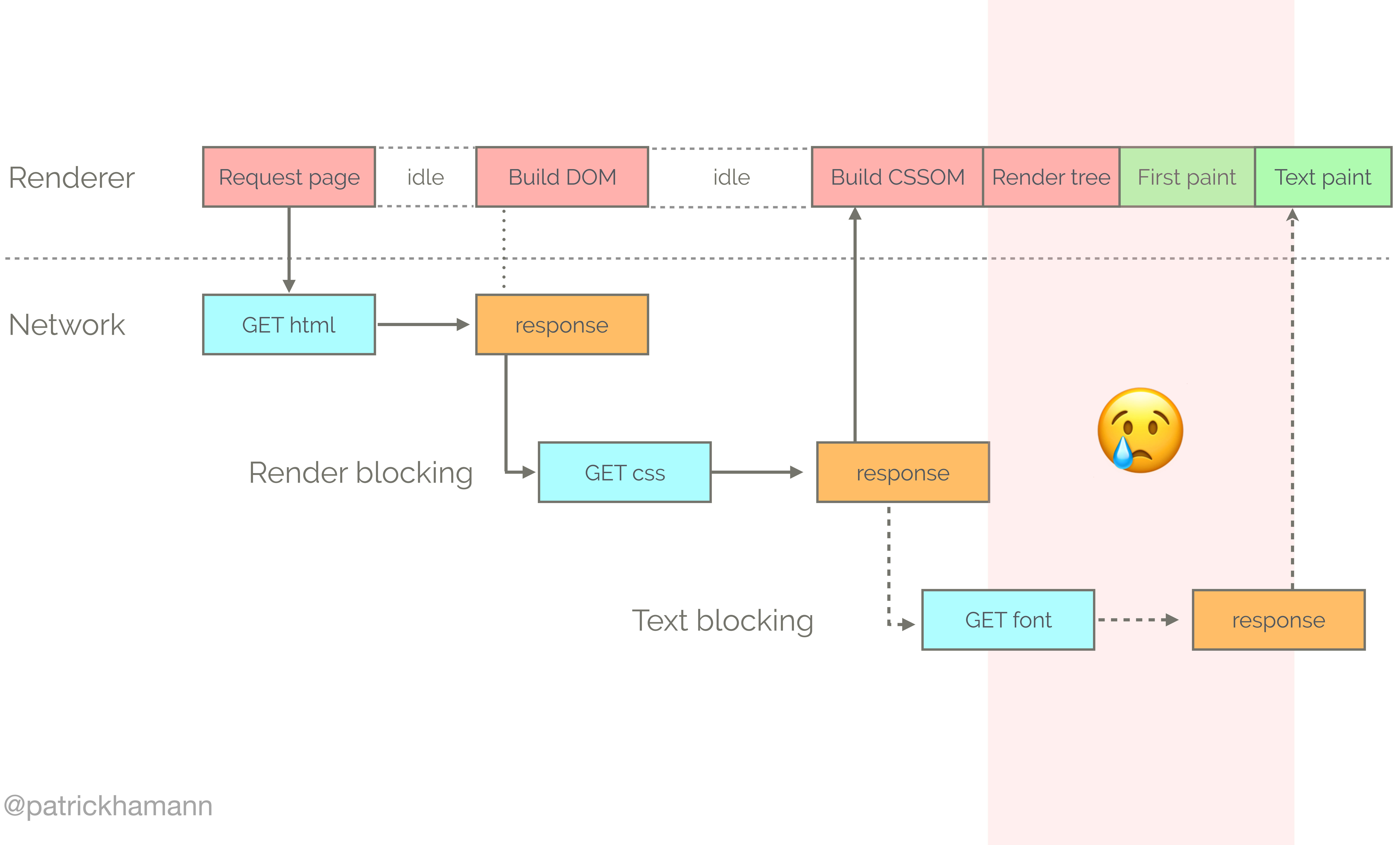
Time to Interactive

Fully loaded

A good loading strategy:

- ✅ Prioritises above-the-fold rendering
- ✅ Prioritises interactivity
- ✅ Is easy to use
- ✅ Is measurable

Preload



@patrickhamann

What are my hidden sub-resources?

✅ Fonts

✅ Application data

✅ Application routes

✅ Async third parties

Provides a declarative fetch primitive that initiates an early fetch and separates fetching from resource execution.

Preload with HTTP header:

```
Link: <my-awesome-font.woff>; rel=preload; as=font; crossorigin
Link: <application-data.json>; rel=preload; as=fetch;
Link: <sub-module.mjs>; rel=modulepreload;
```

Preload with markup:

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

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

@patrickhamann

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
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
...
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

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

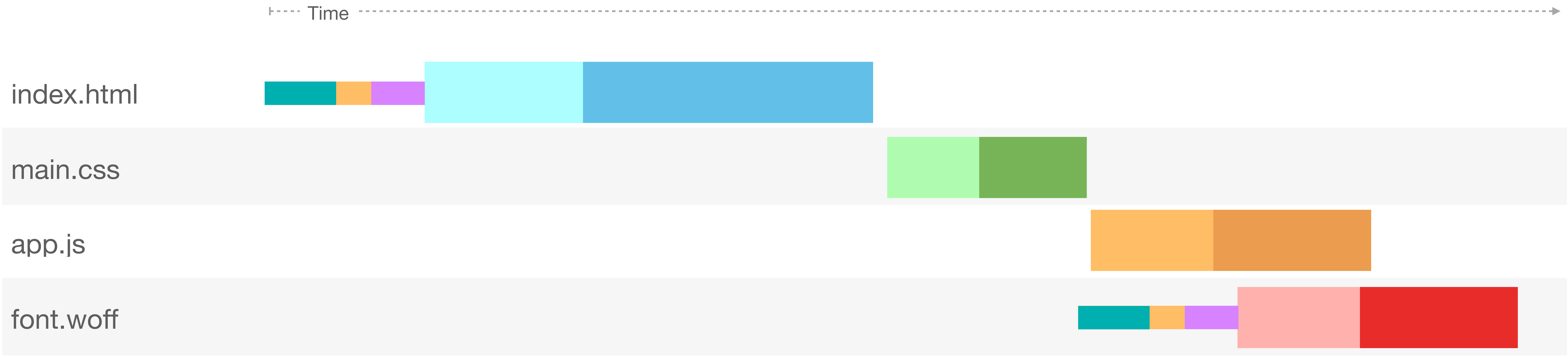
1. ttfmp.fastlylabs.com - index.html 2935 ms
2. ttfmp.fastlylabs.com - main.css 3235 ms
3. www.ft.com - ftlogo:brand-myft 653 ms
4. www.ft.com - fticon-v1:hamburger 597 ms
5. www.ft.com - f...brand-ft-masthead 3554 ms
6. www.ft.com - M...cWeb-Regular.woff 2971 ms
7. www.ft.com - F...yWeb-Regular.woff 5603 ms
8. www.ft.com - M...Web-Semibold.woff 3462 ms
9. www.ft.com - e...5.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
...

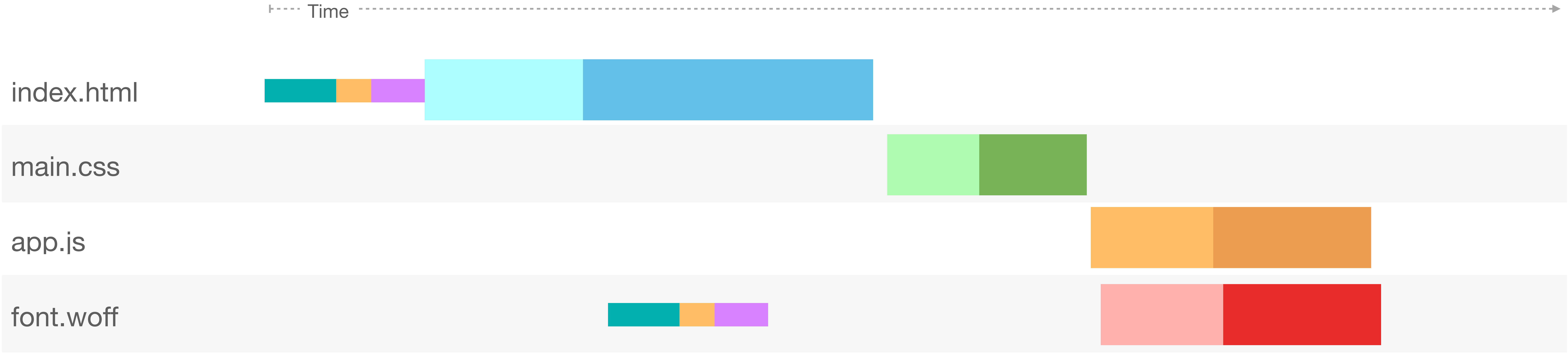
After

> Shopify's switch to preloading fonts saw a 50% (1.2 second) improvement in time-to-text-paint. This removed their flash-of-invisible text completely.
>
> – Shopify

Preconnect

No preconnect

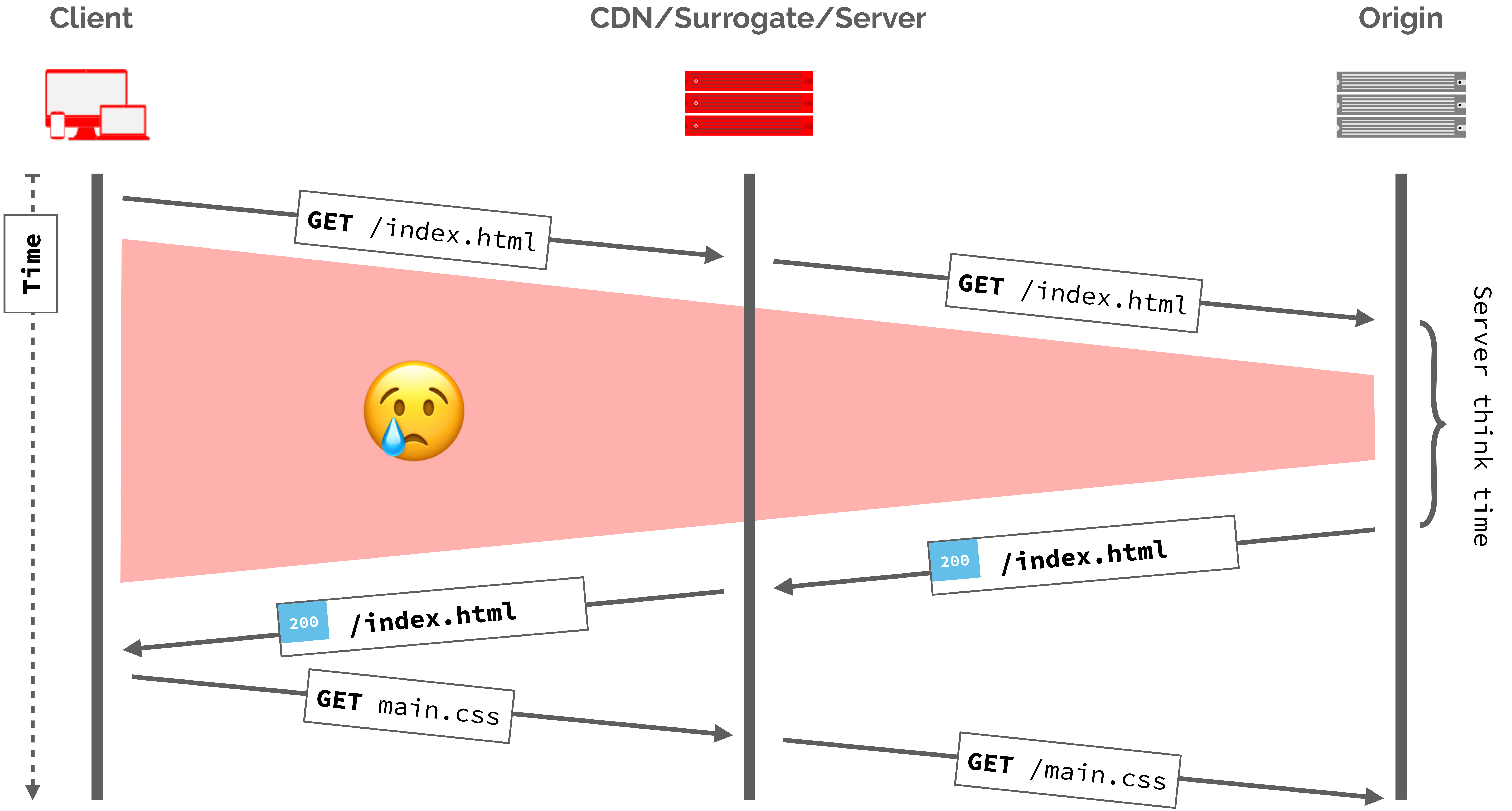


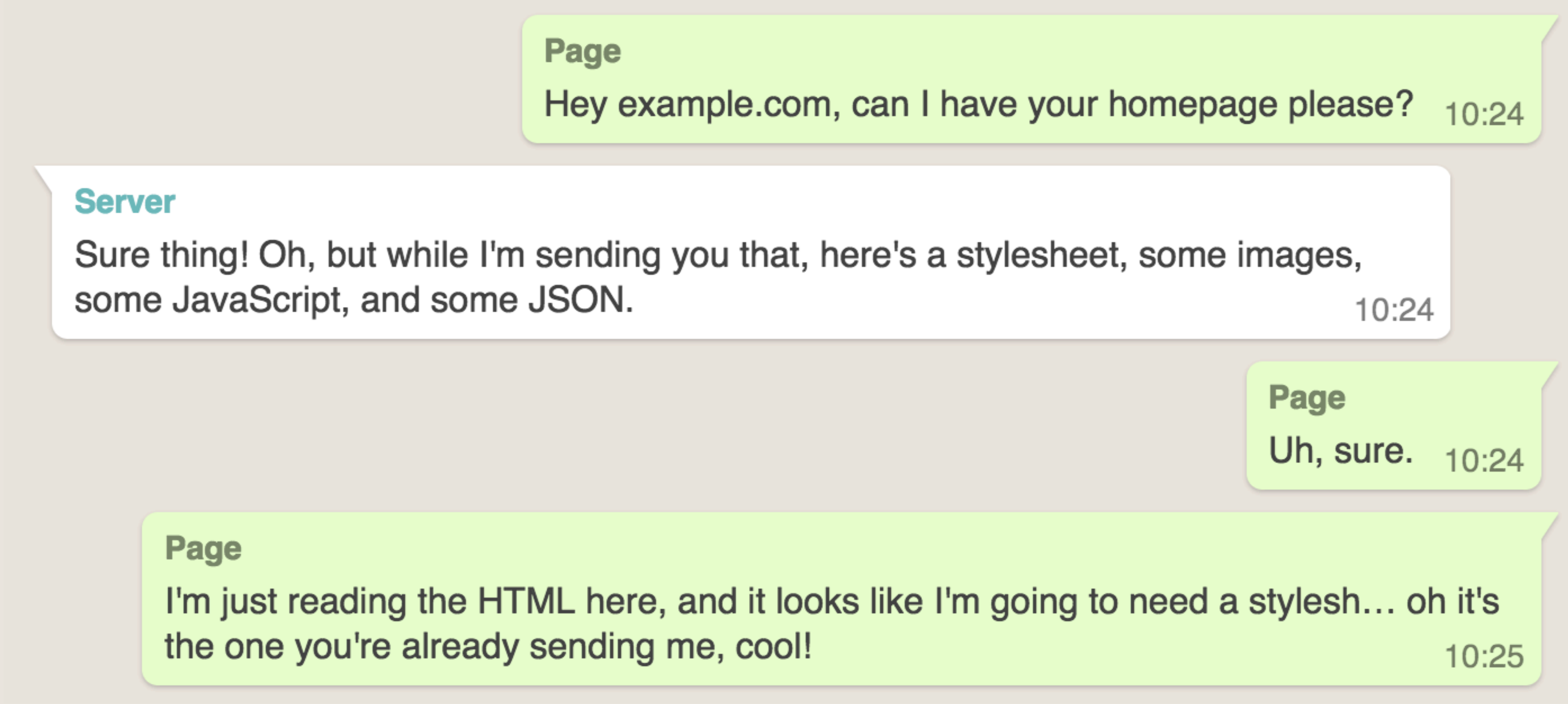


Preconnect

Are indicating resource hints via the HTML response too late?

Server push


Client
CDN/Surrogate/Server
Origin
Time
GET /index.html
GET /index.html
Server think time
200 /index.html
200 /index.html
GET main.css
GET /main.css



Page

Hey example.com, can I have your homepage please? 10:24

Server

Sure thing! Oh, but while I'm sending you that, here's a stylesheet, some images, some JavaScript, and some JSON. 10:24

Page

Uh, sure. 10:24

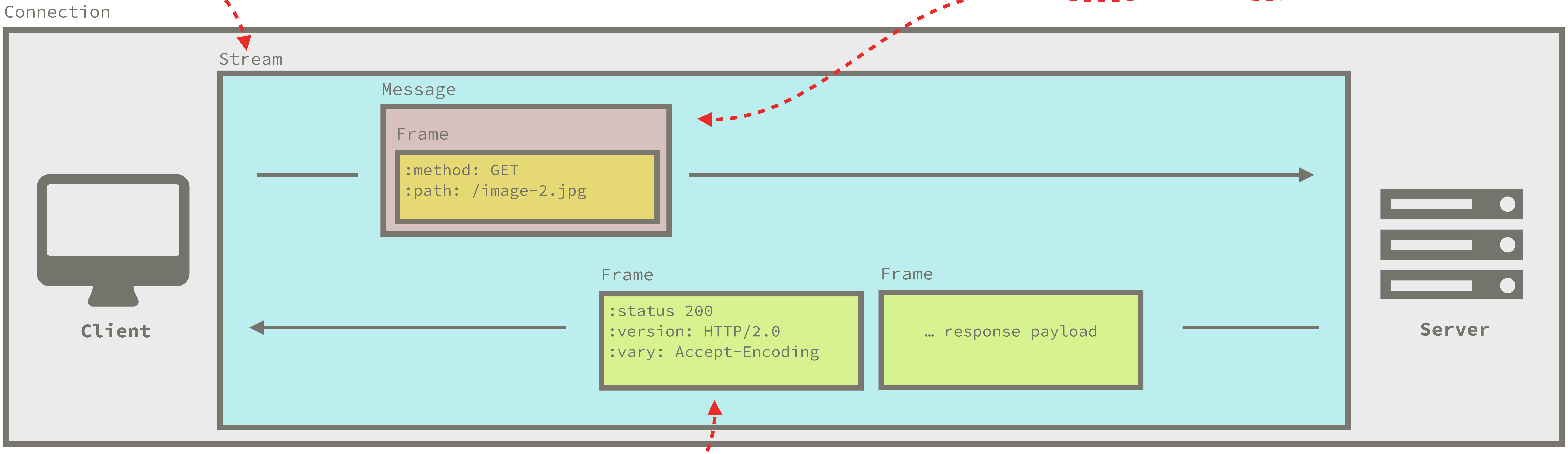
Page

I'm just reading the HTML here, and it looks like I'm going to need a stylesh… oh it's the one you're already sending me, cool! 10:25

Jake Archibald – https://jakearchibald.com/2017/h2-push-tougher-than-i-thought/

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

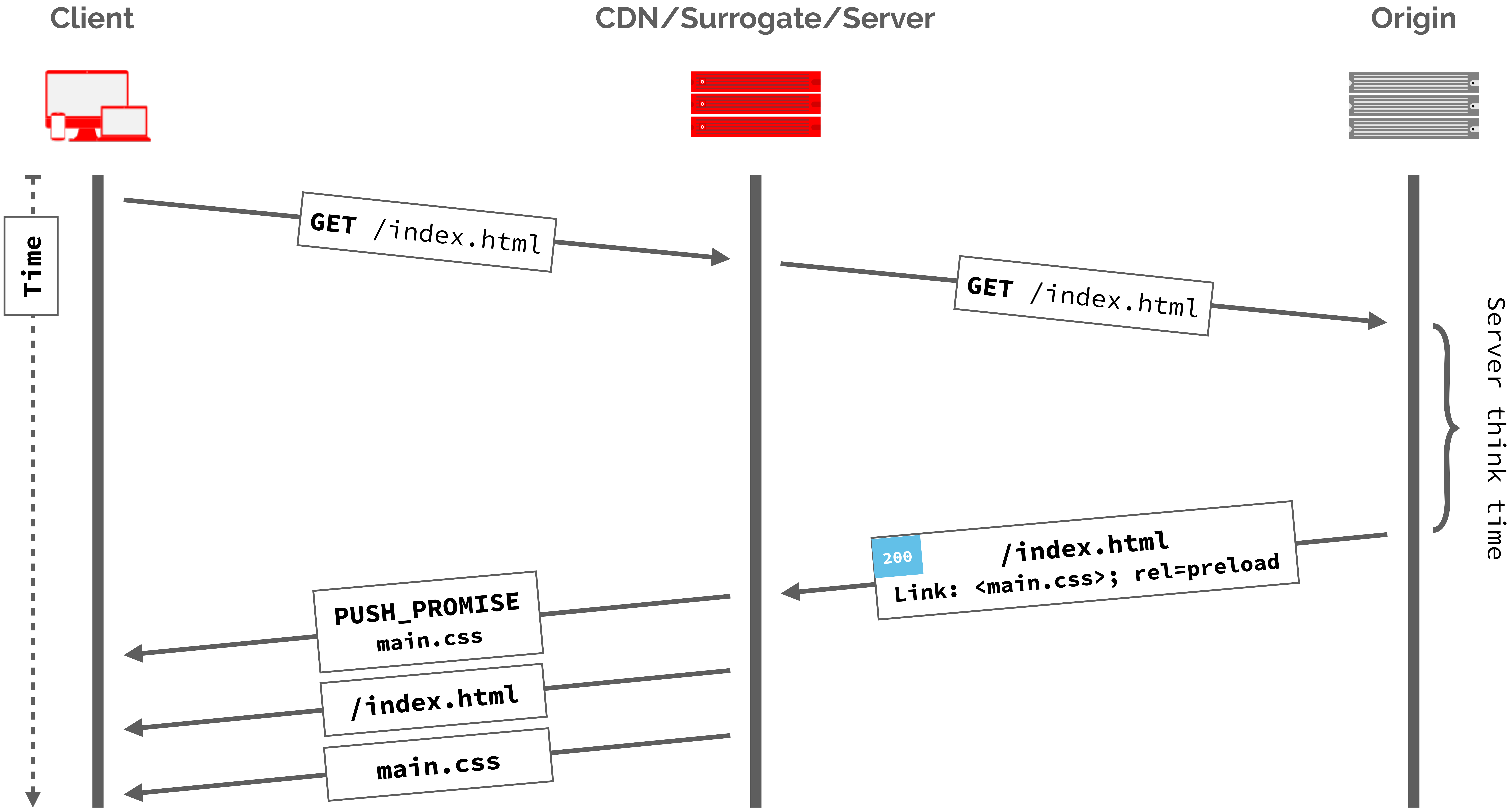
Message
A complete sequence of frames that map to a logical HTTP message, such as a request.



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

Multiplexing

@patrickhamann


Client
CDN/Surrogate/Server
Origin
Time
GET /index.html
GET /index.html
Server think time
200
/index.html
Link: <main.css>; rel=preload
PUSH_PROMISE
main.css
/index.html
main.css

So how can I push?

Indicate push via preload Link header.

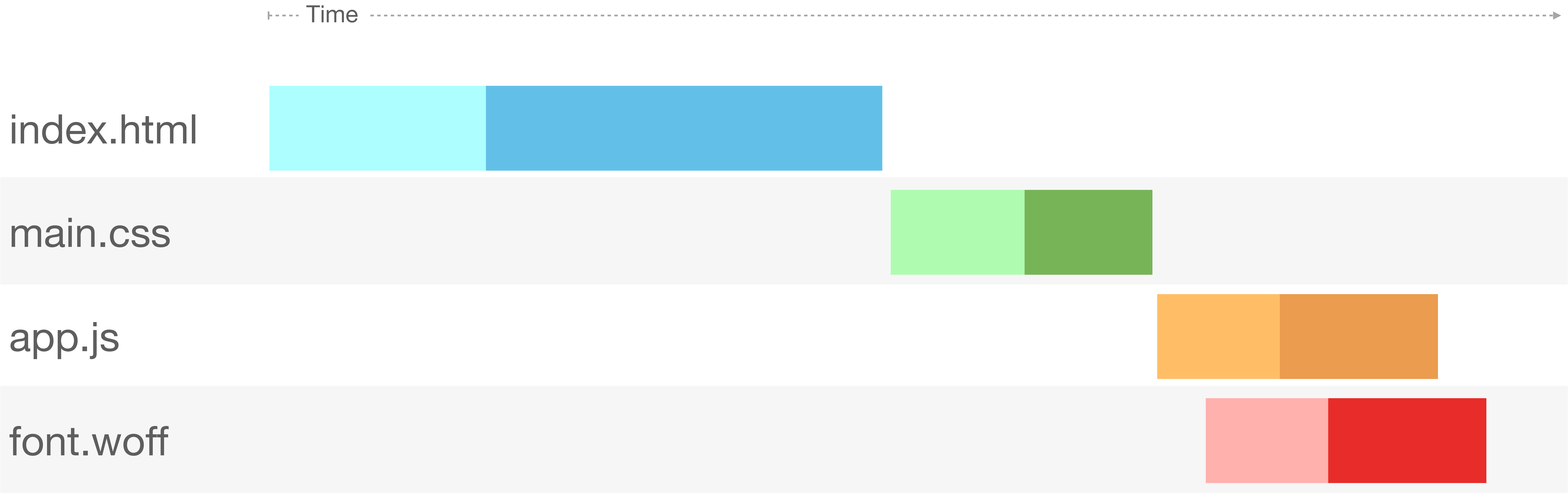
```
Link: <font.woff2>; rel=preload; as=font crossorigin
```

Use no push attribute to disable push semantics and only use preload.

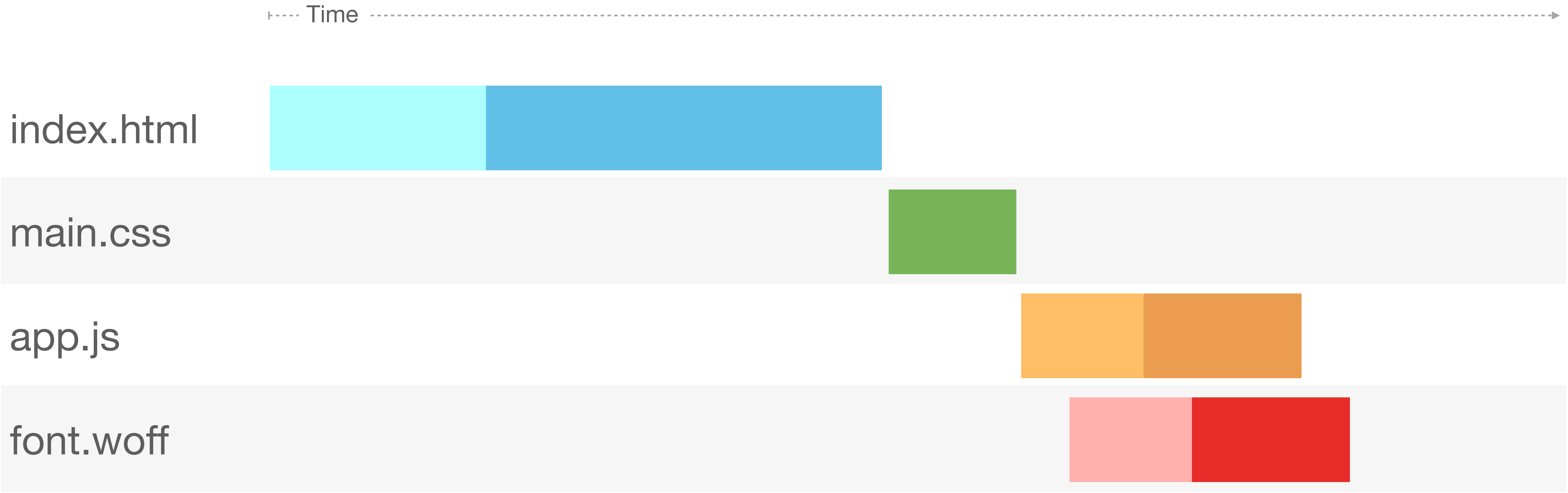
```
Link: <main.css>; rel=preload; as=style; nopush
```

Fastly uses x-http2-push-only attribute to disable preload semantics

```
Link: <application.js>; rel=preload; as=style; x-http2-push-only
```

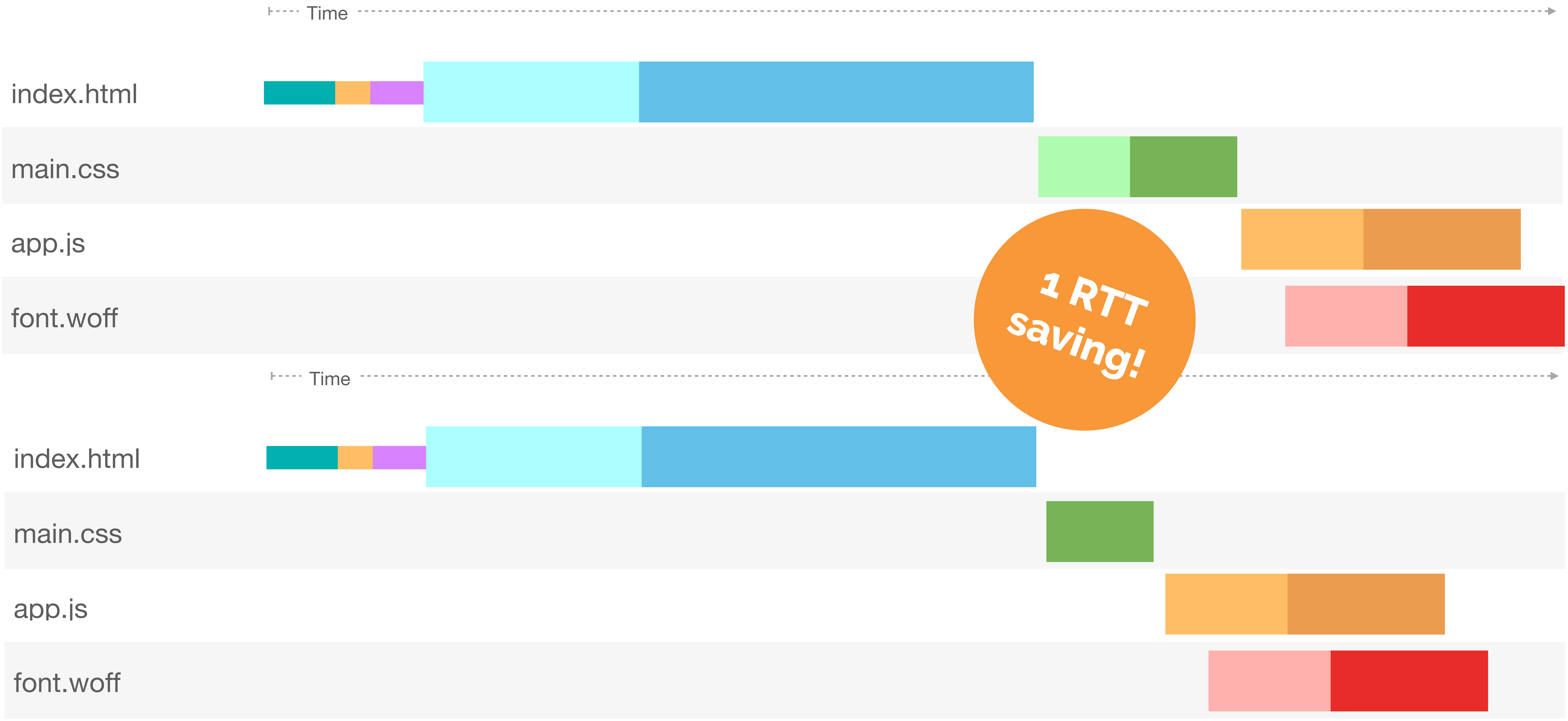


Before

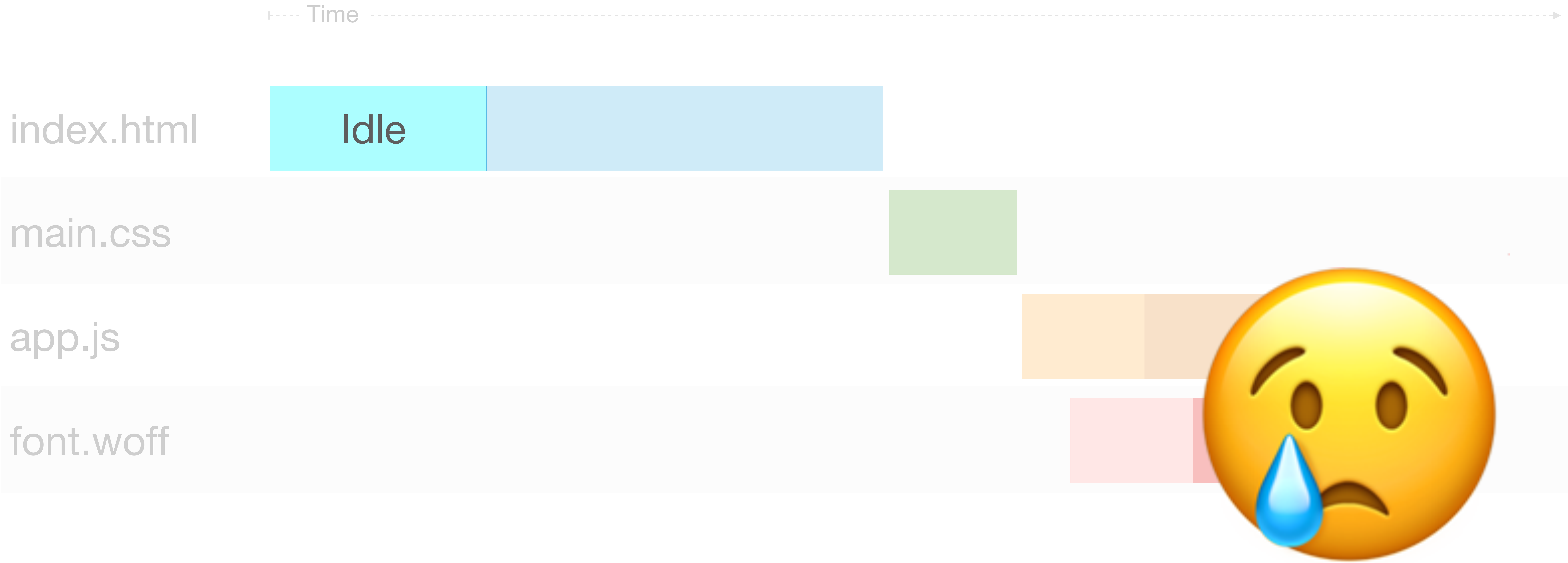


After

No Push


Time
index.html
main.css
app.js
font.woff
1 RTT saving!
Time
index.html
main.css
app.js
font.woff

Push



Time

index.html

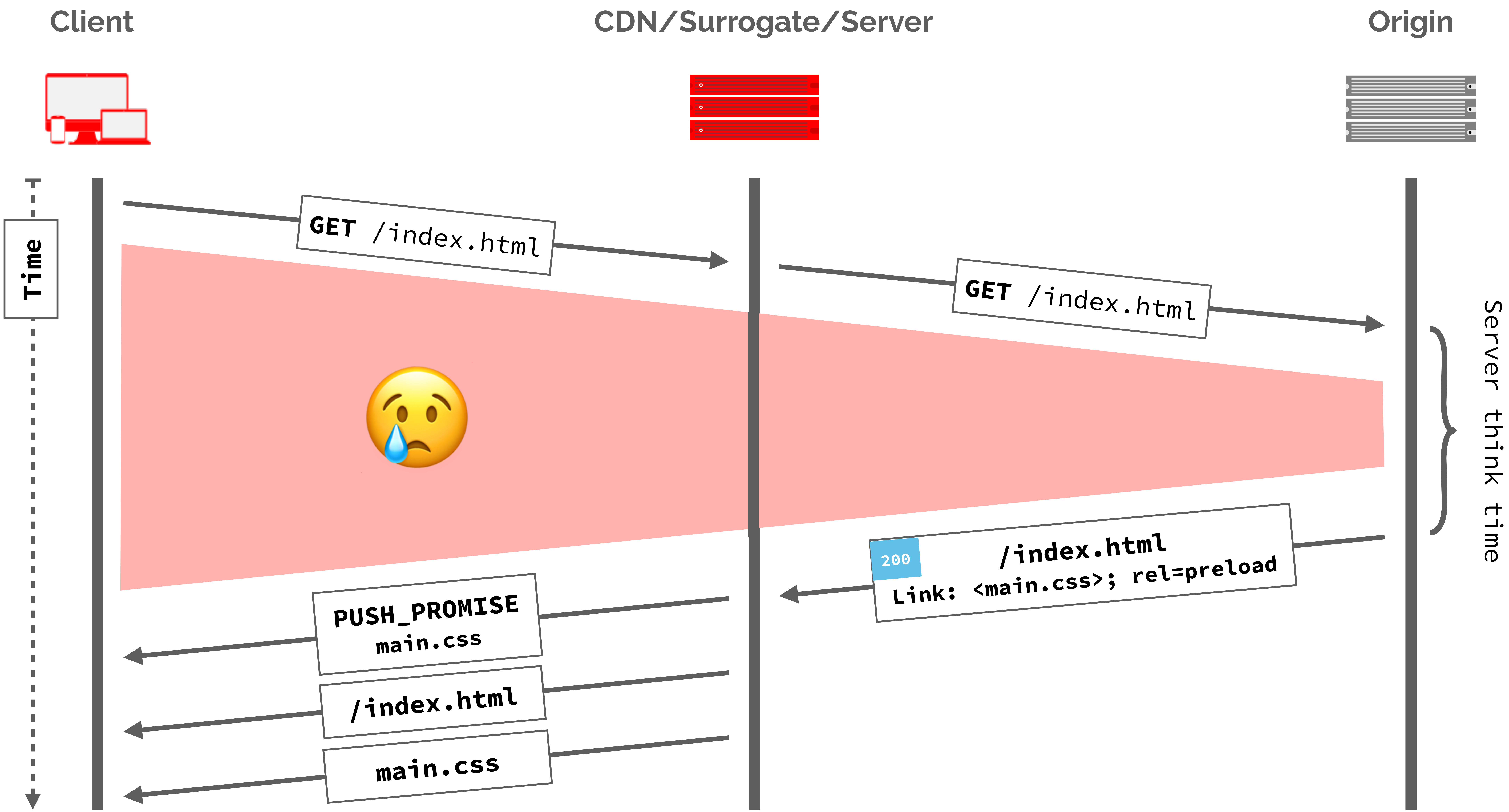
Idle

main.css

app.js

font.woff

After


Client
CDN/Surrogate/Server
Origin
Time
GET /index.html
GET /index.html
Server think time
200
/index.html
Link: <main.css>; rel=preload
PUSH_PROMISE
main.css
/index.html
main.css

Server push benefits:

✅ 1 RTT saving

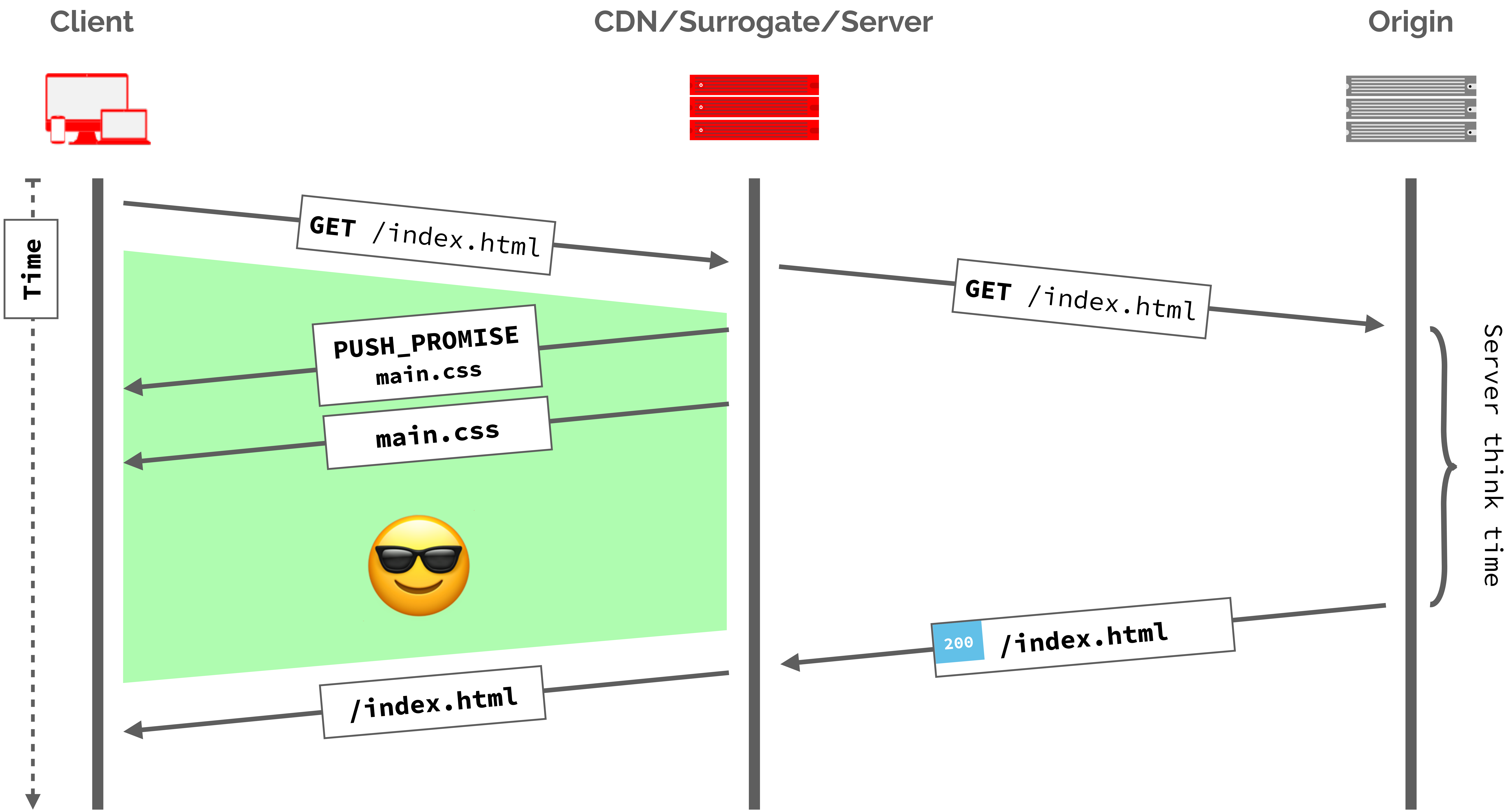
✅ Useful for long server think time

✅ Useful for long RTT times

⚠️ Link header indication is too late

Is indicating push via the HTML response too late?

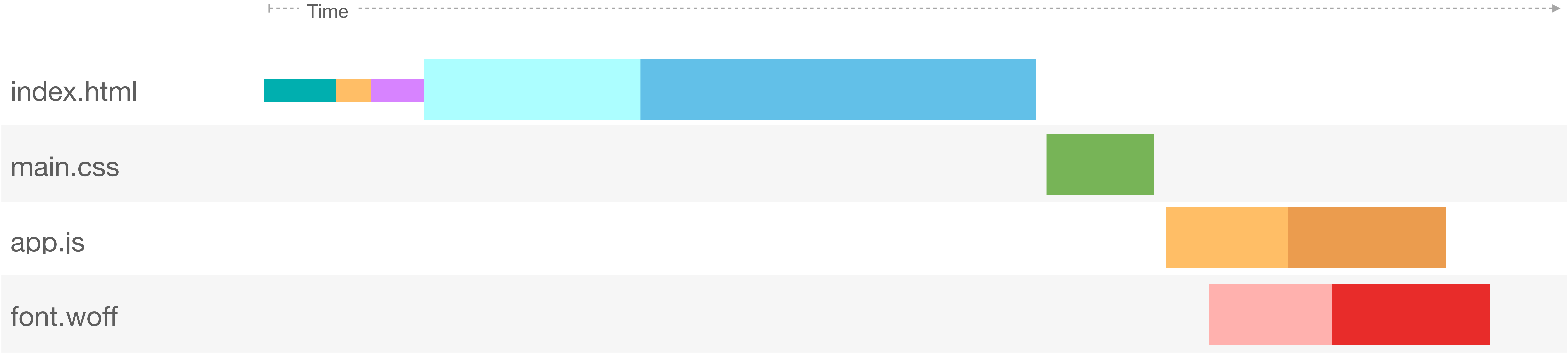
Async push

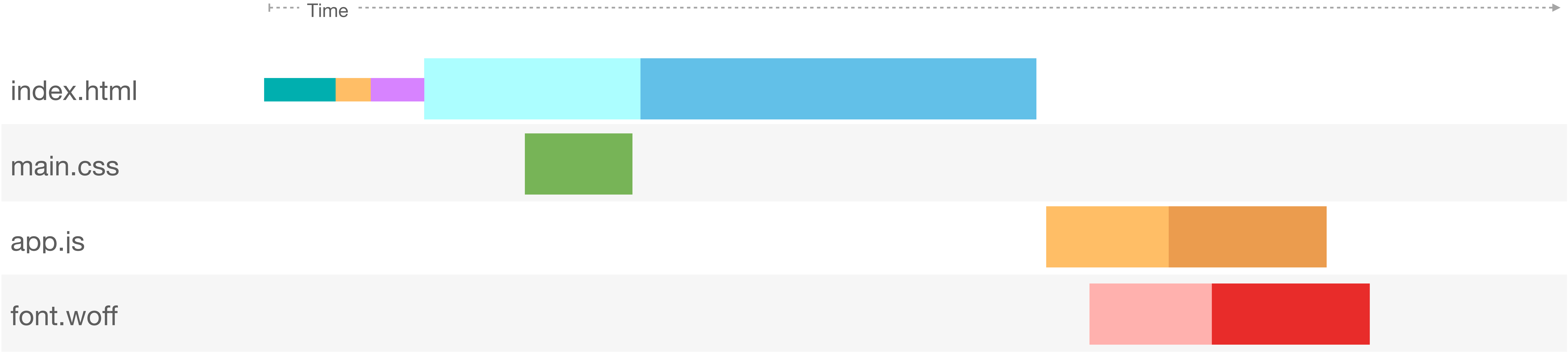

Client
CDN/Surrogate/Server
Origin
Time
GET /index.html
GET /index.html
PUSH_PROMISE
main.css
main.css
Server think time
200
/index.html
/index.html

```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);
```

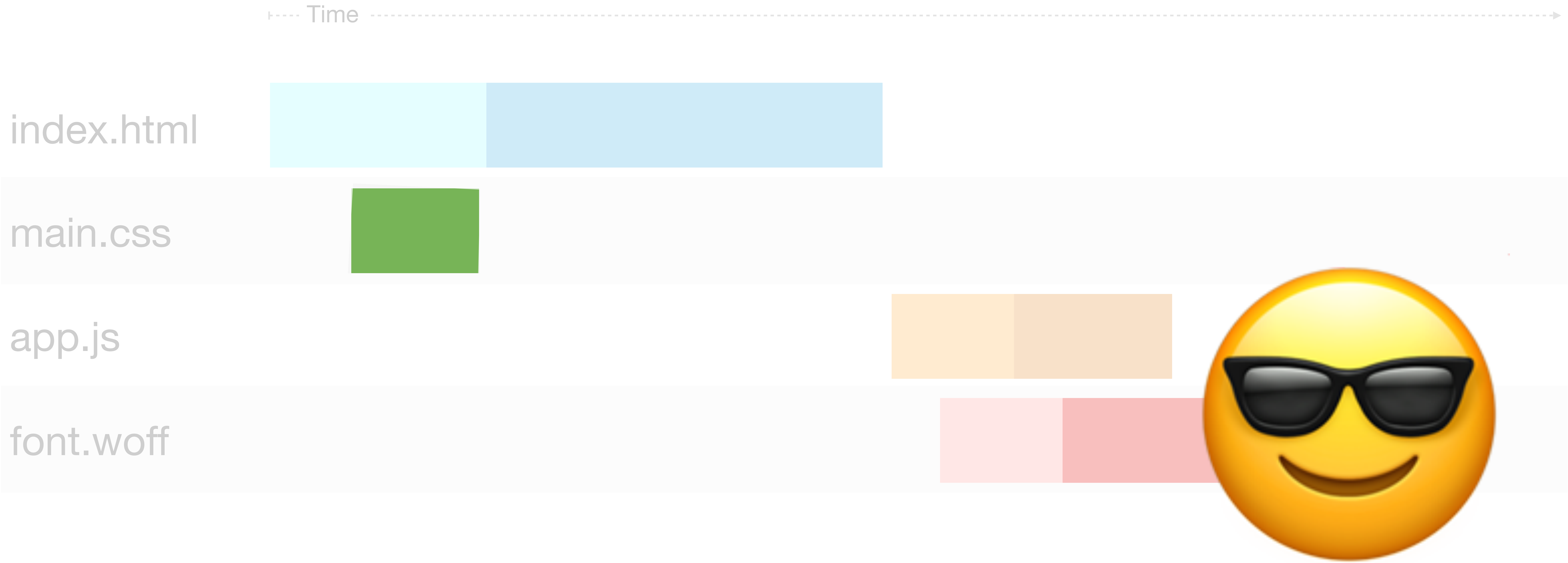
```
sub vcl_recv {
  if (fastly_info.is_h2 && req.url ~ "^/index.html") {
    h2.push('/critical.css');
  }

  // etc ...

}
```

Push





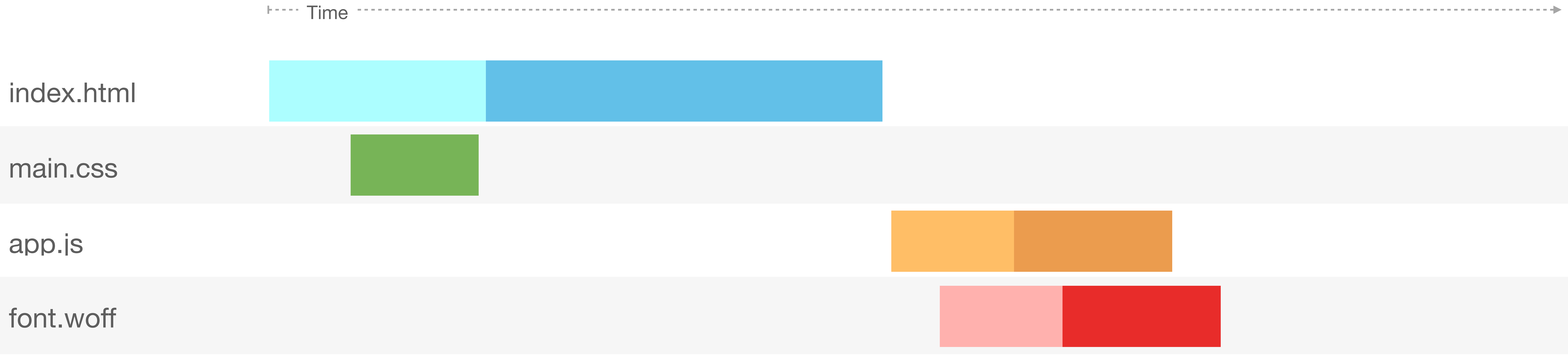
Async push

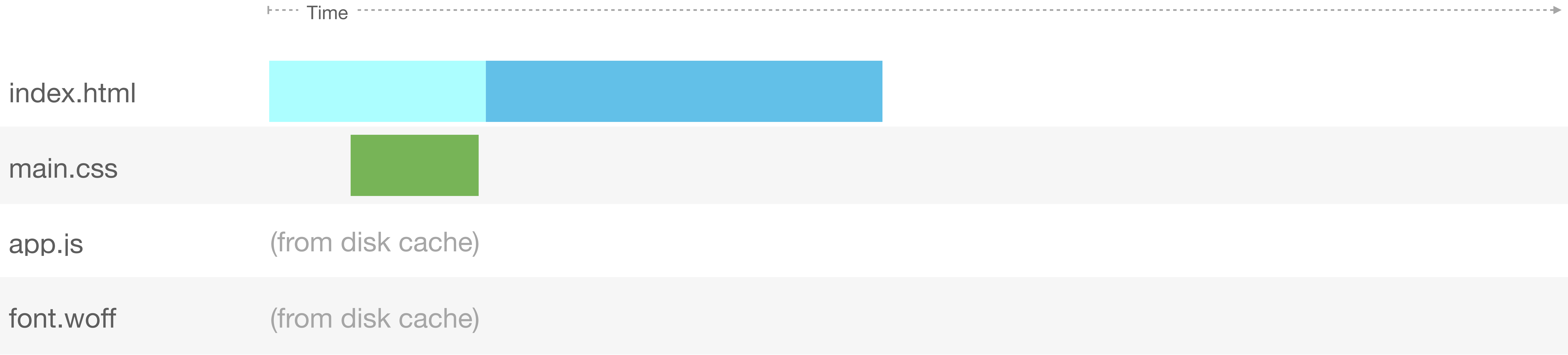


Utilising idle network server think time == win!

What about the repeat view?

First view





Repeat view

The server has no knowledge of client cache state.

In the wild

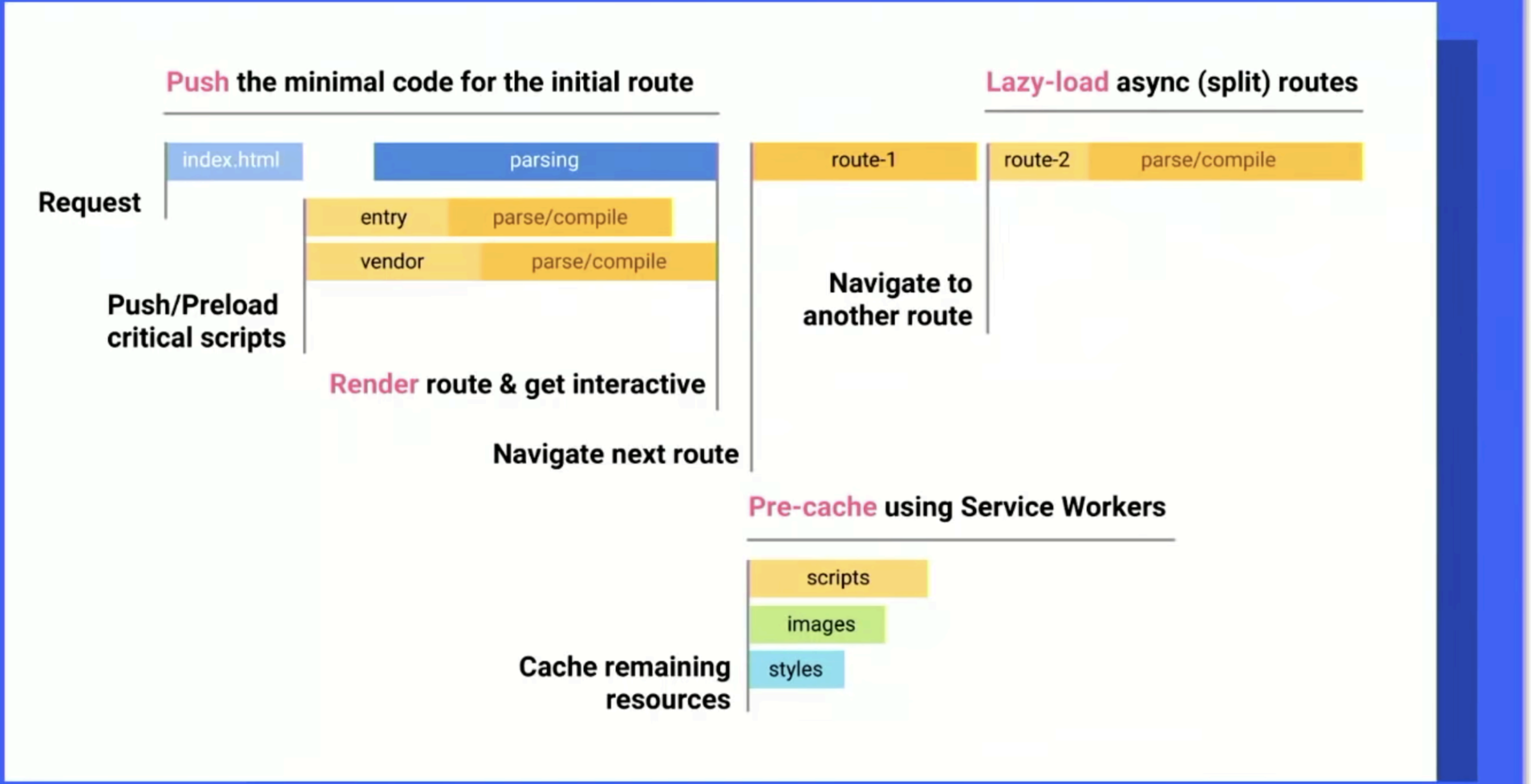
> Faster image loads times,
> 15% reduction in time to first byte
>
> – Facebook

Client cache

EDGE/CDN
WEB SERVER
345 ms
110 ms
Feed request
Image request
Image cache
Cache HIT
PUSH_PROMISE
Lookup Image
SCALE

HTTP2 server push - Facebook https://atscaleconference.com/videos/http2-server-push-lower-latencies-around-the-world/

PRPL Pattern



Push the minimal code for the initial route

Lazy-load async (split) routes

Request

index.html

parsing

entry parse/compile

vendor parse/compile

Push/Preload critical scripts

Render route & get interactive

Navigate next route

route-1

route-2 parse/compile

Navigate to another route

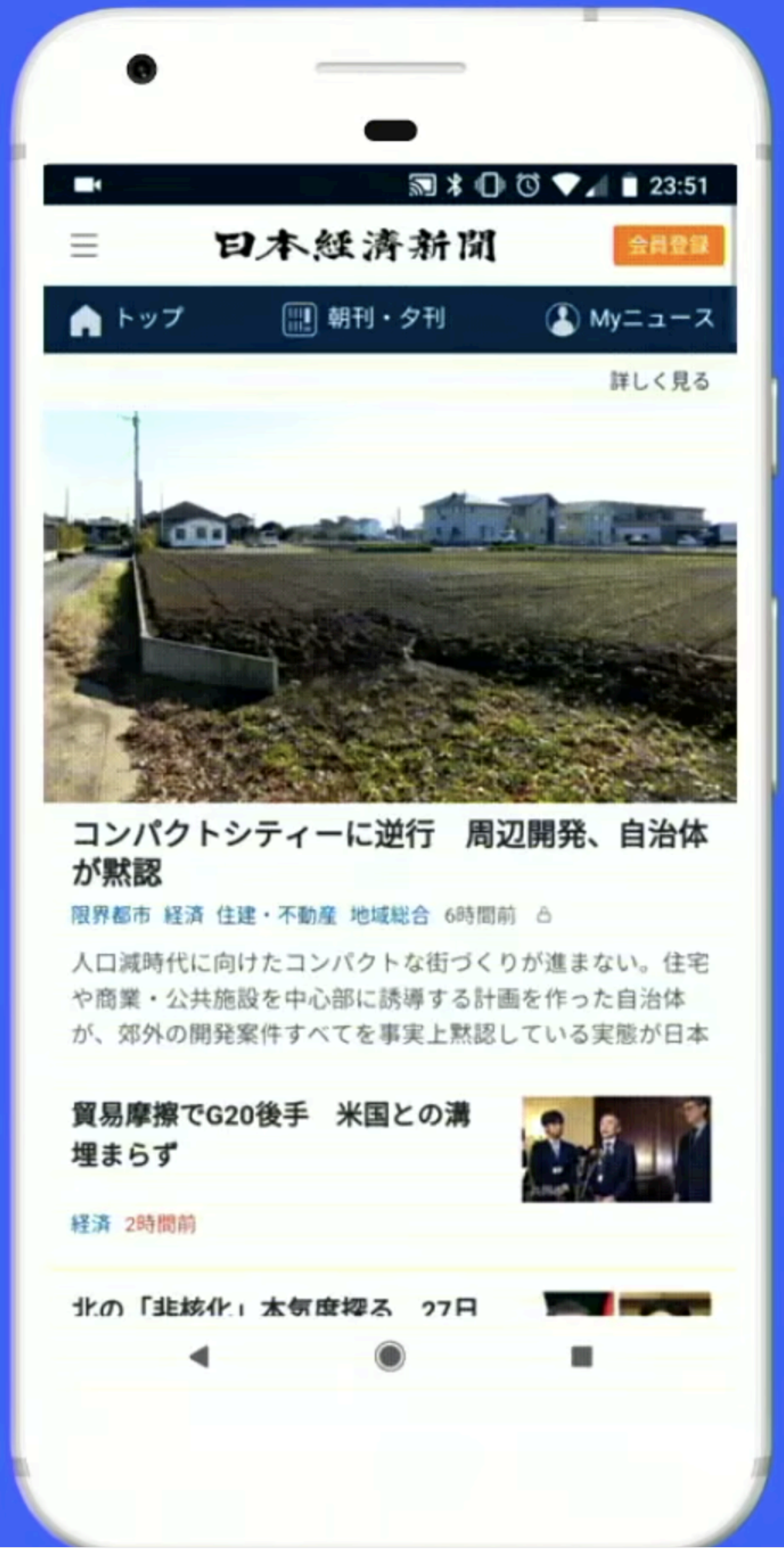
Pre-cache using Service Workers

scripts

images

styles

Cache remaining resources



23:51

日本経済新聞

コンパクトシティーに逆行　周辺開発、自治体が黙認

限界都市　経済　住建・不動産　地域総合　6時間前

人口減時代に向けたコンパクトな街づくりが進まない。住宅や商業・公共施設を中心部に誘導する計画を作った自治体が、郊外の開発案件すべてを事実上黙認している実態が日本

貿易摩擦でG20後手　米国との溝埋まらず

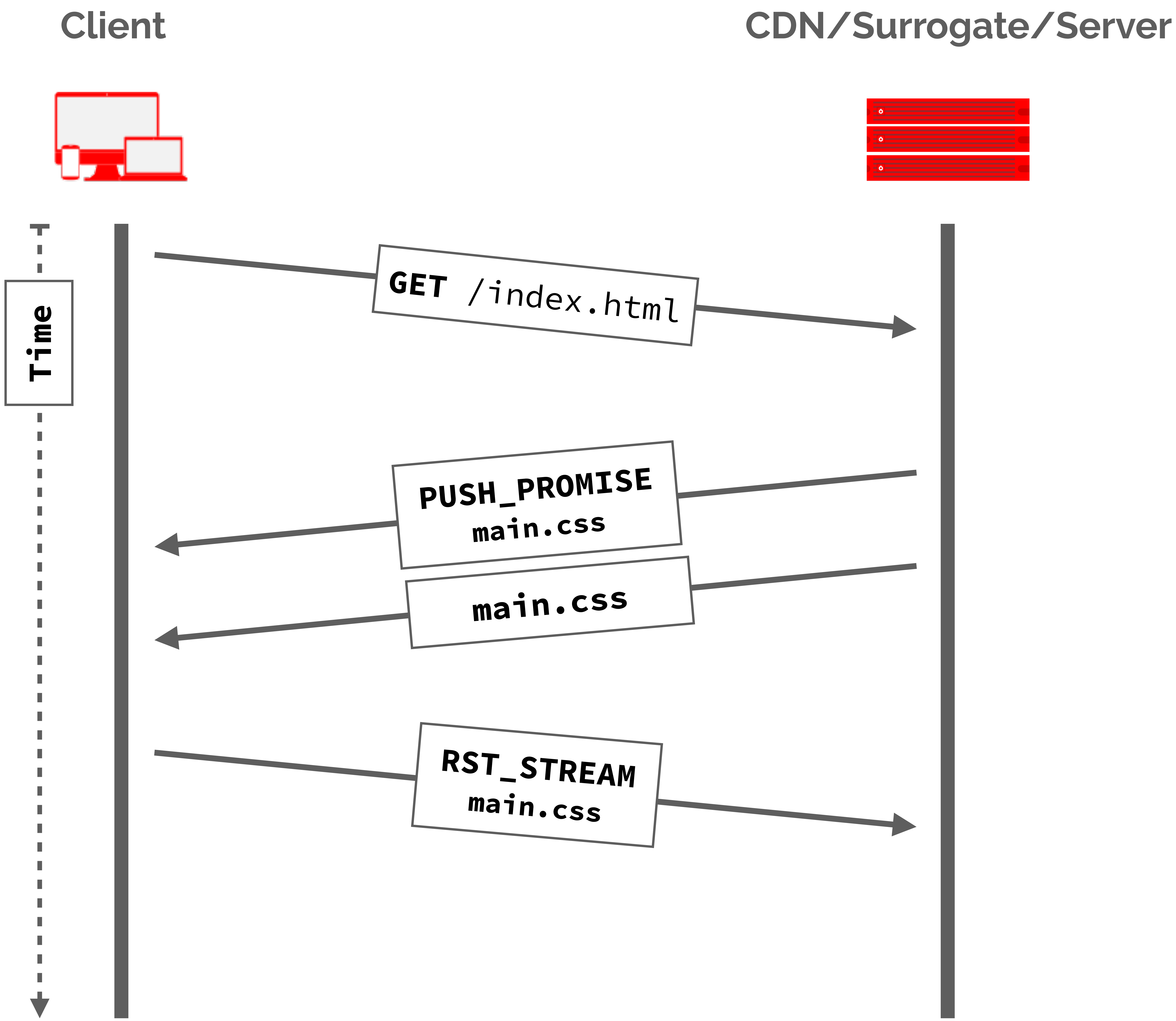
経済　2時間前

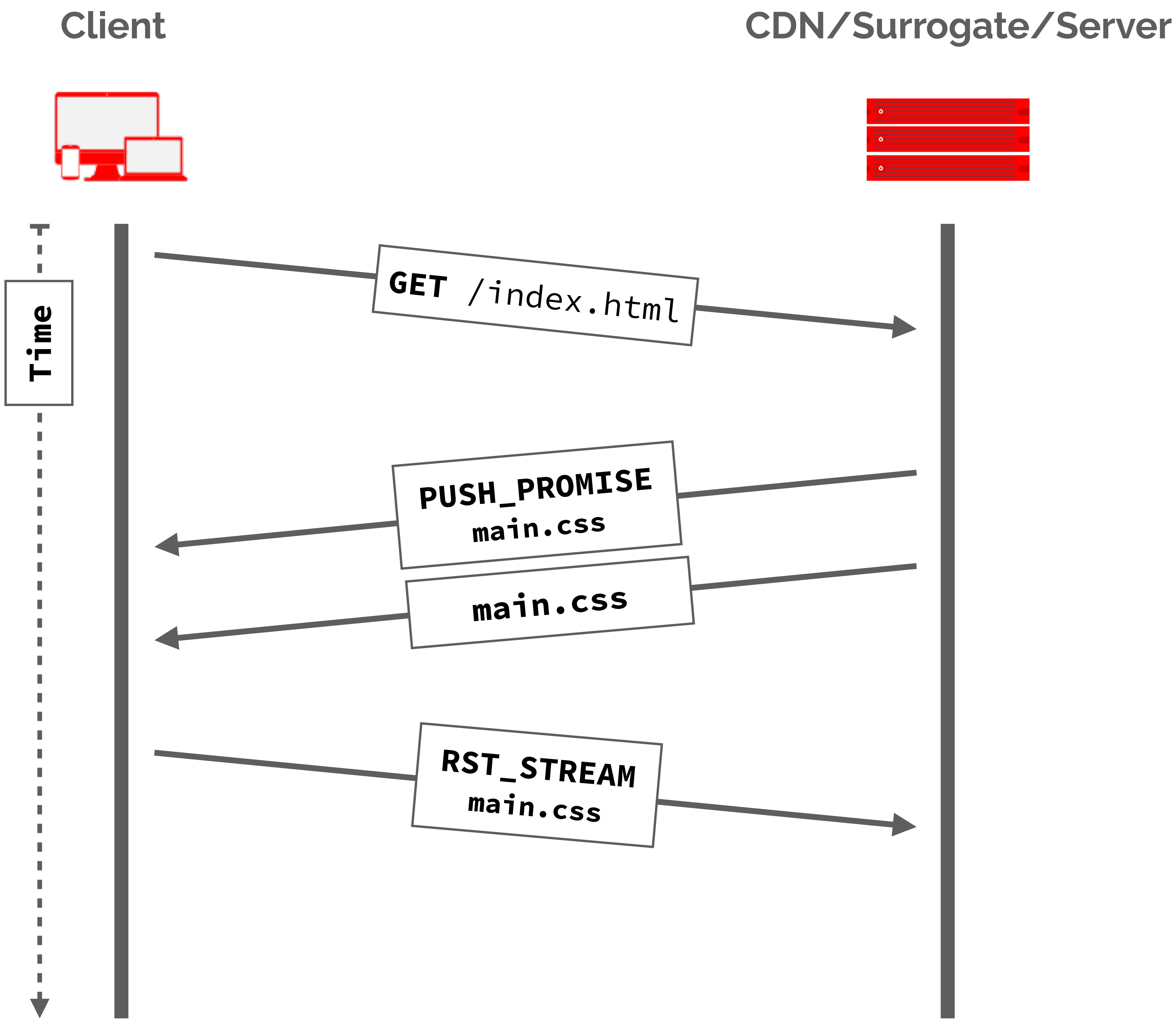
北の「非核化」本気度探る　27日

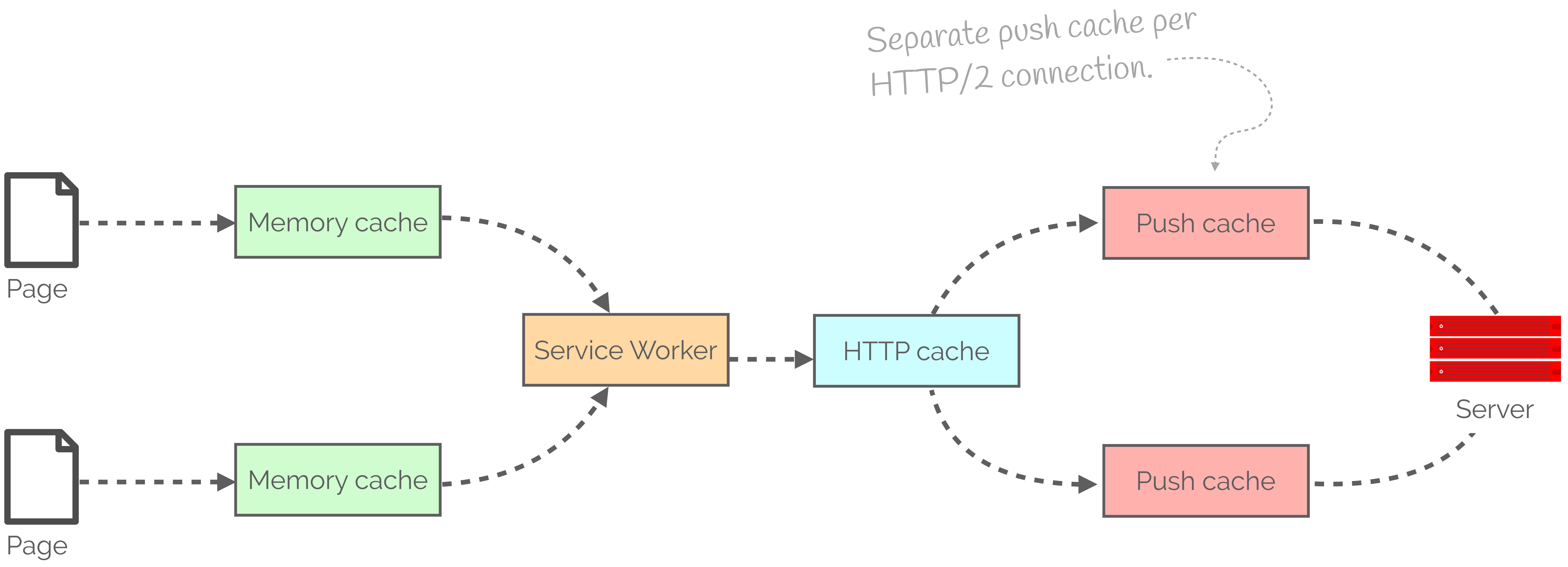
Nikkei.com - Web performance made easy: Addy Osmani, Ewa Gasperowicz https://youtu.be/Mv-l3-tJgGk

Poll?

So what’s the problem?


Client
CDN/Surrogate/Server
Time
GET /index.html
PUSH_PROMISE
main.css
main.css
RST_STREAM
main.css

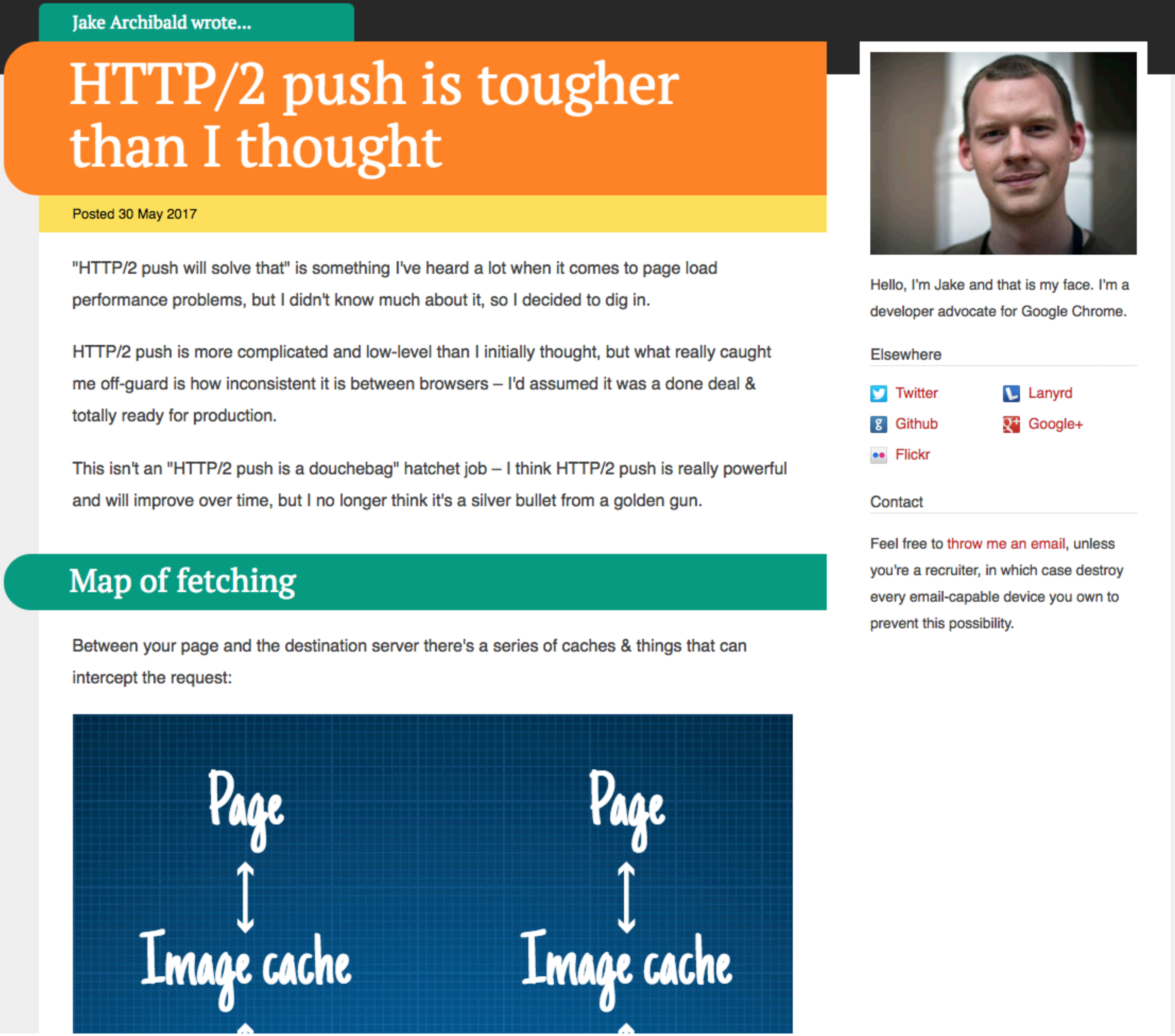

Client
CDN/Surrogate/Server
Time
GET /index.html
PUSH_PROMISE
main.css
main.css
RST_STREAM
main.css



@patrickhamann

Push cache semantics

- ⚠️ Connection must be authoritative
- ⚠️ Cache per HTTP/2 connection
- ⚠️ Items can only be claimed once
- ⚠️ It’s the last cache
- ⚠️ It’s not spec’d



Jake Archibald wrote...

HTTP/2 push is tougher than I thought

Posted 30 May 2017

"HTTP/2 push will solve that" is something I've heard a lot when it comes to page load performance problems, but I didn't know much about it, so I decided to dig in.

HTTP/2 push is more complicated and low-level than I initially thought, but what really caught me off-guard is how inconsistent it is between browsers – I'd assumed it was a done deal & totally ready for production.

This isn't an "HTTP/2 push is a douchebag" hatchet job – I think HTTP/2 push is really powerful and will improve over time, but I no longer think it's a silver bullet from a golden gun.

Map of fetching

Between your page and the destination server there's a series of caches & things that can intercept the request:

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

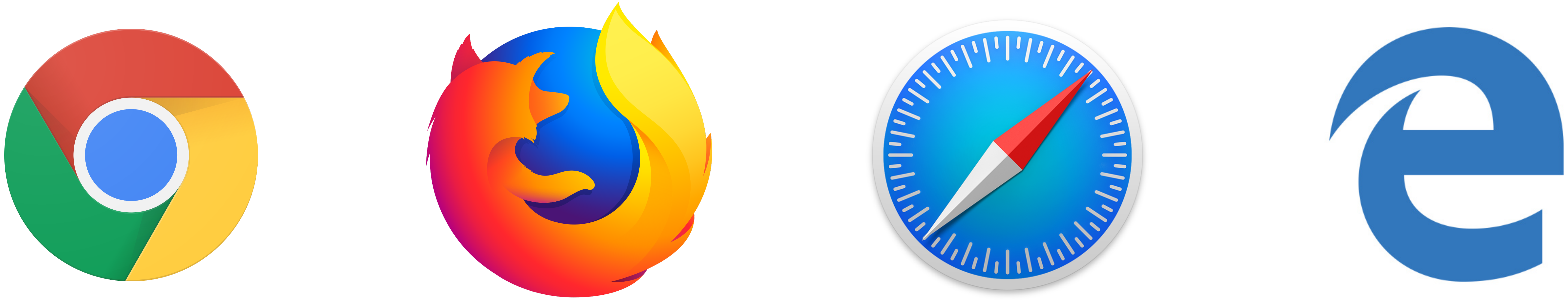
Elsewhere

- Twitter
- Lanyrd
- Github
- Google+
- Flickr

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.

HTTP/2 push is tougher than I thought – Jake Archibald https://jakearchibald.com/2017/h2-push-tougher-than-i-thought/



✅ ✅ ⚠️ ⚠️

HTTP/2 Server Push - Browser inconsistencies

0.008% of requests on the Fastly network are push initiated.

When should I push?

- ✅ You have long RTTs or server processing
- ✅ You can use async push
- ✅ You have a client-rendered app shell (PRPL)
- ✅ You control the client cache (SW, native, Electron etc)

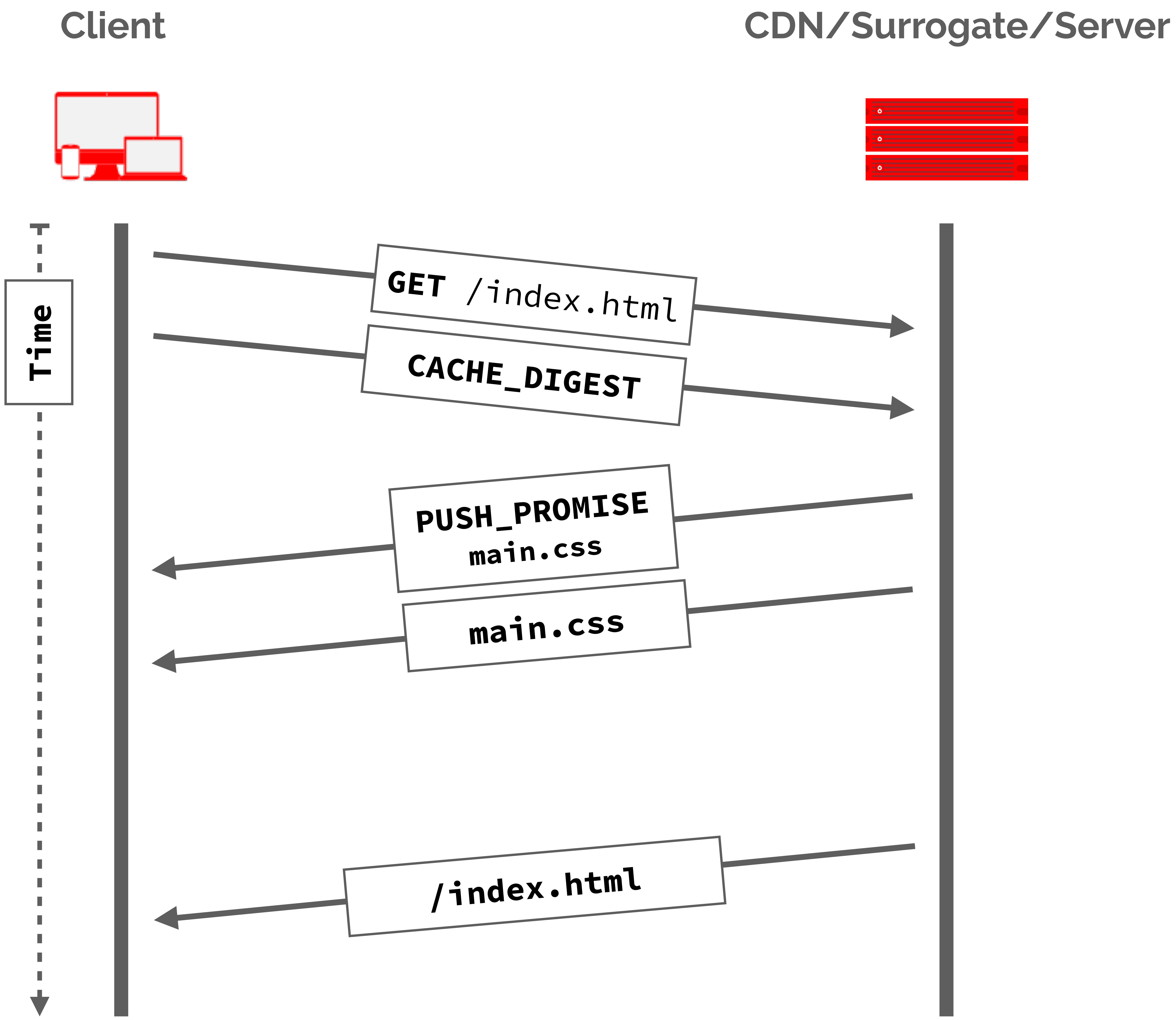
Is the 1 RTT saving worth the complexity?

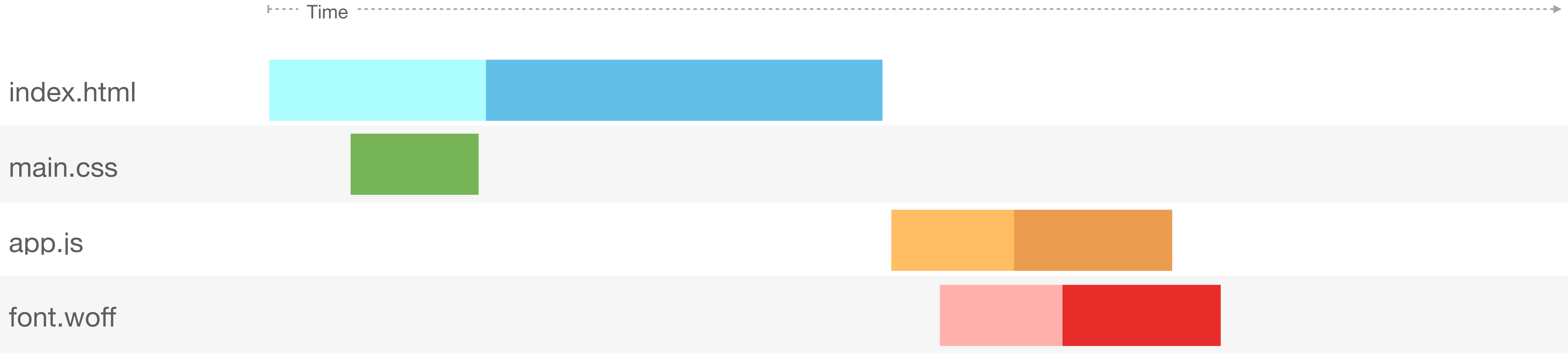
Are there other solutions?

The future

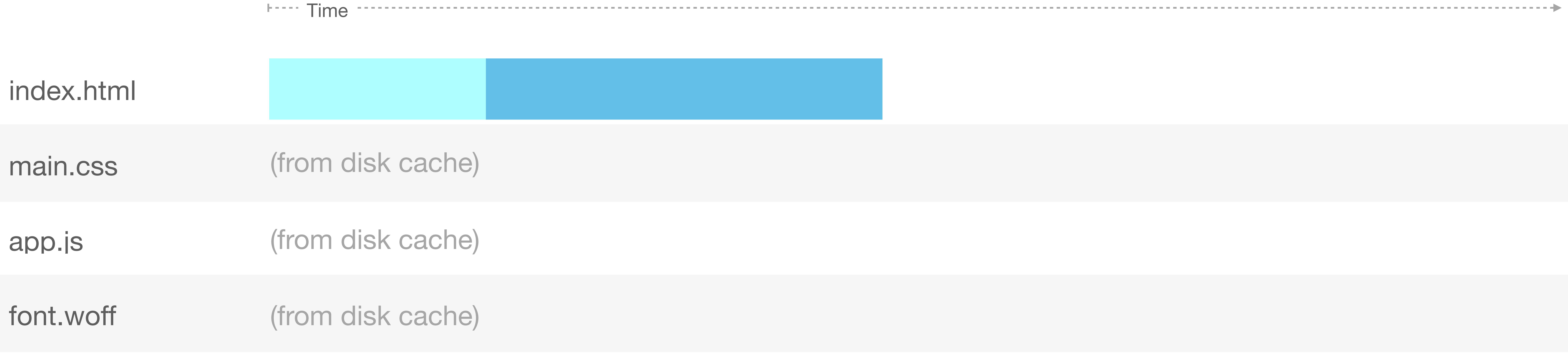
Can we fix the problems with push?

Cache digests

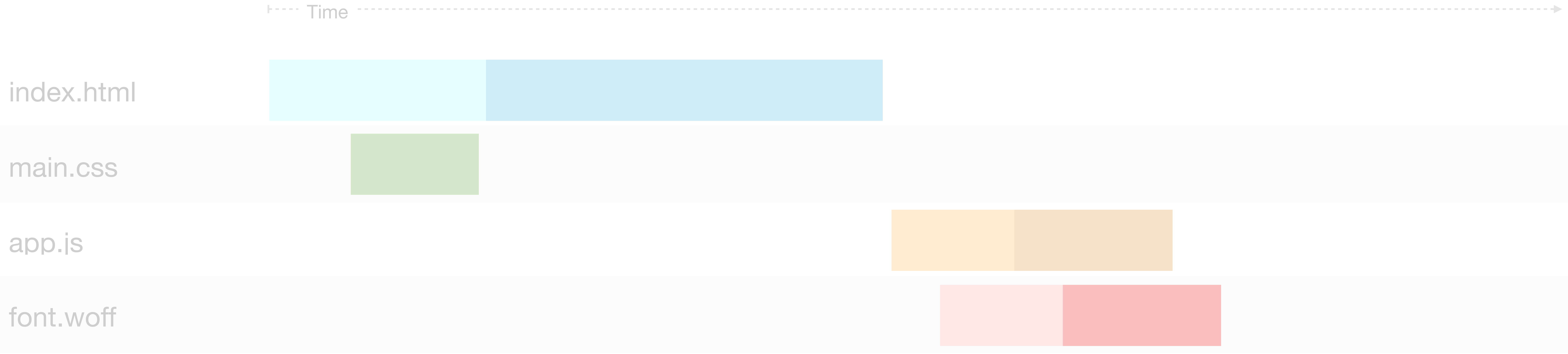

Client
CDN/Surrogate/Server
Time
GET /index.html
CACHE_DIGEST
PUSH_PROMISE main.css
main.css
/index.html



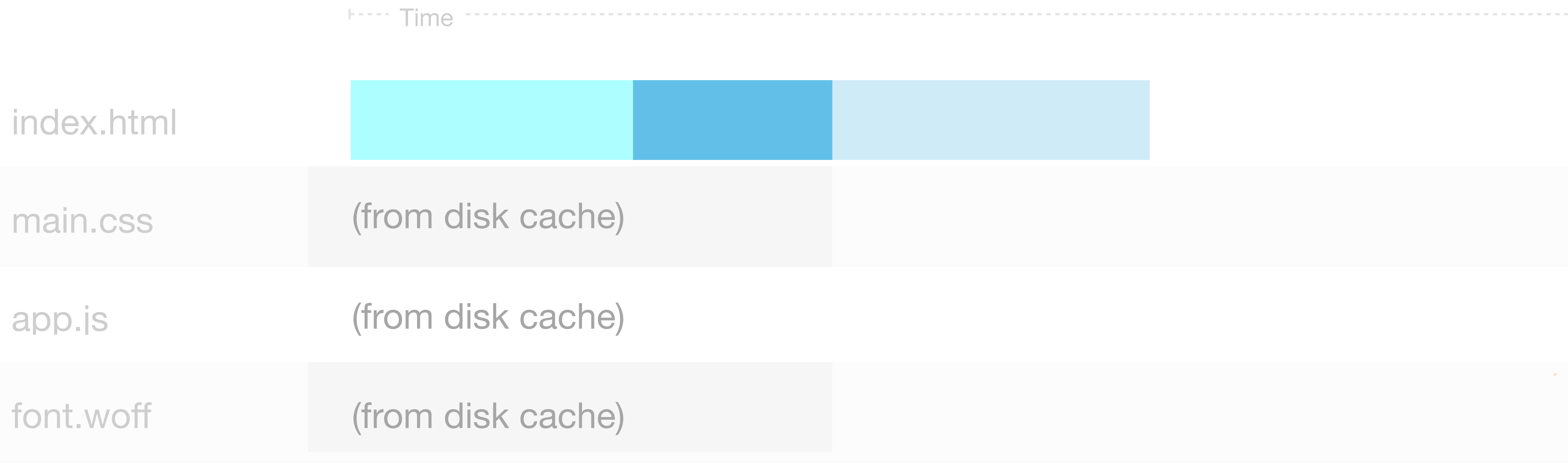
First view



Repeat view



First view

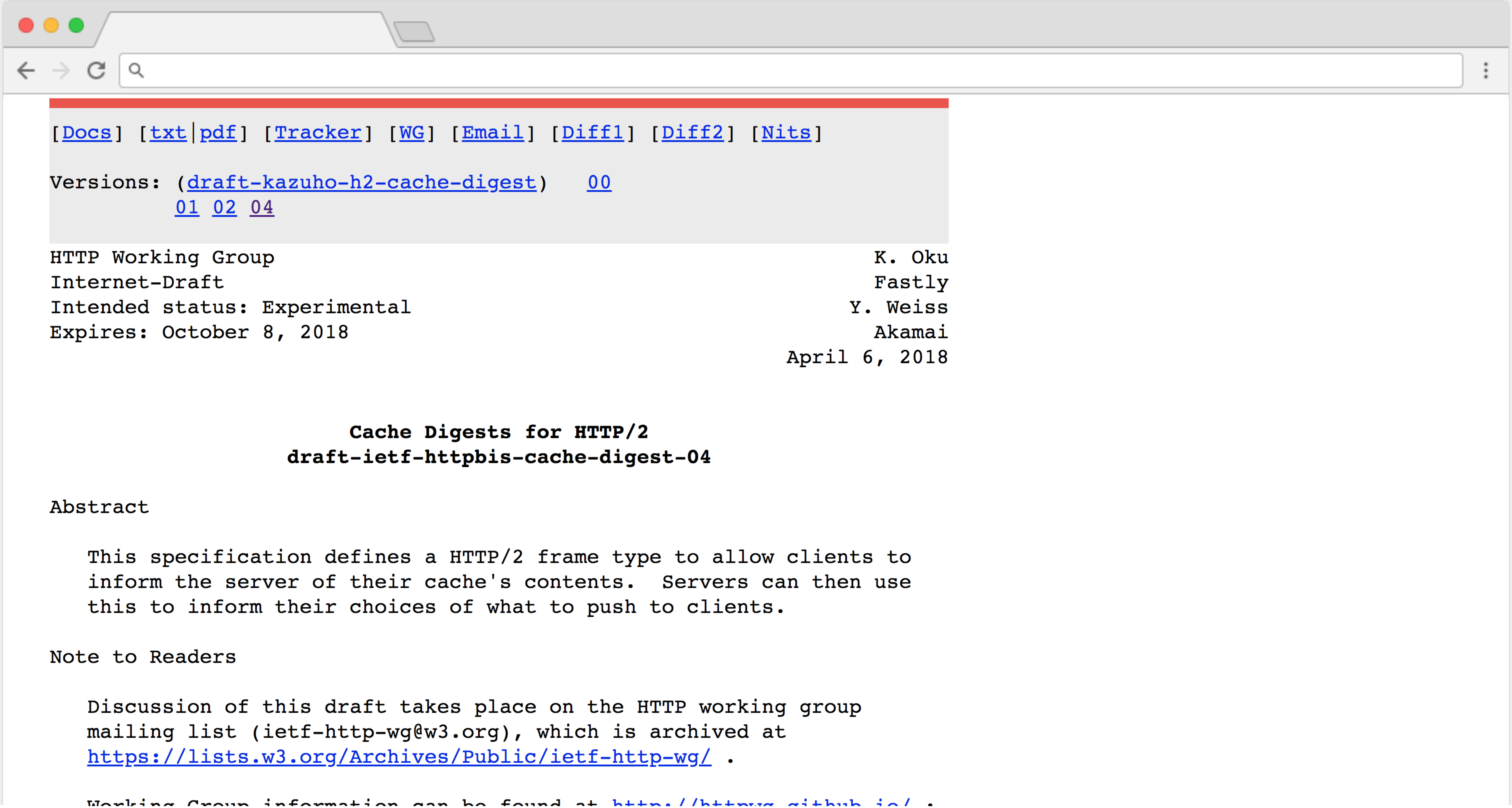


main.css (from disk cache)

app.js (from disk cache)

font.woff (from disk cache)

Repeat view



[Docs] [txt|pdf] [Tracker] [WG] [Email] [Diff1] [Diff2] [Nits]

Versions: (draft-kazuho-h2-cache-digest) 00
01 02 04

```
HTTP Working Group                                                K. Oku
Internet-Draft                                                    Fastly
Intended status: Experimental                                   Y. Weiss
Expires: October 8, 2018                                          Akamai
                                                           April 6, 2018
```

Cache Digests for HTTP/2
draft-ietf-httpbis-cache-digest-04

Abstract

This specification defines a HTTP/2 frame type to allow clients to inform the server of their cache's contents. Servers can then use this to inform their choices of what to push to clients.

Note to Readers

Discussion of this draft takes place on the HTTP working group mailing list (ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/ .

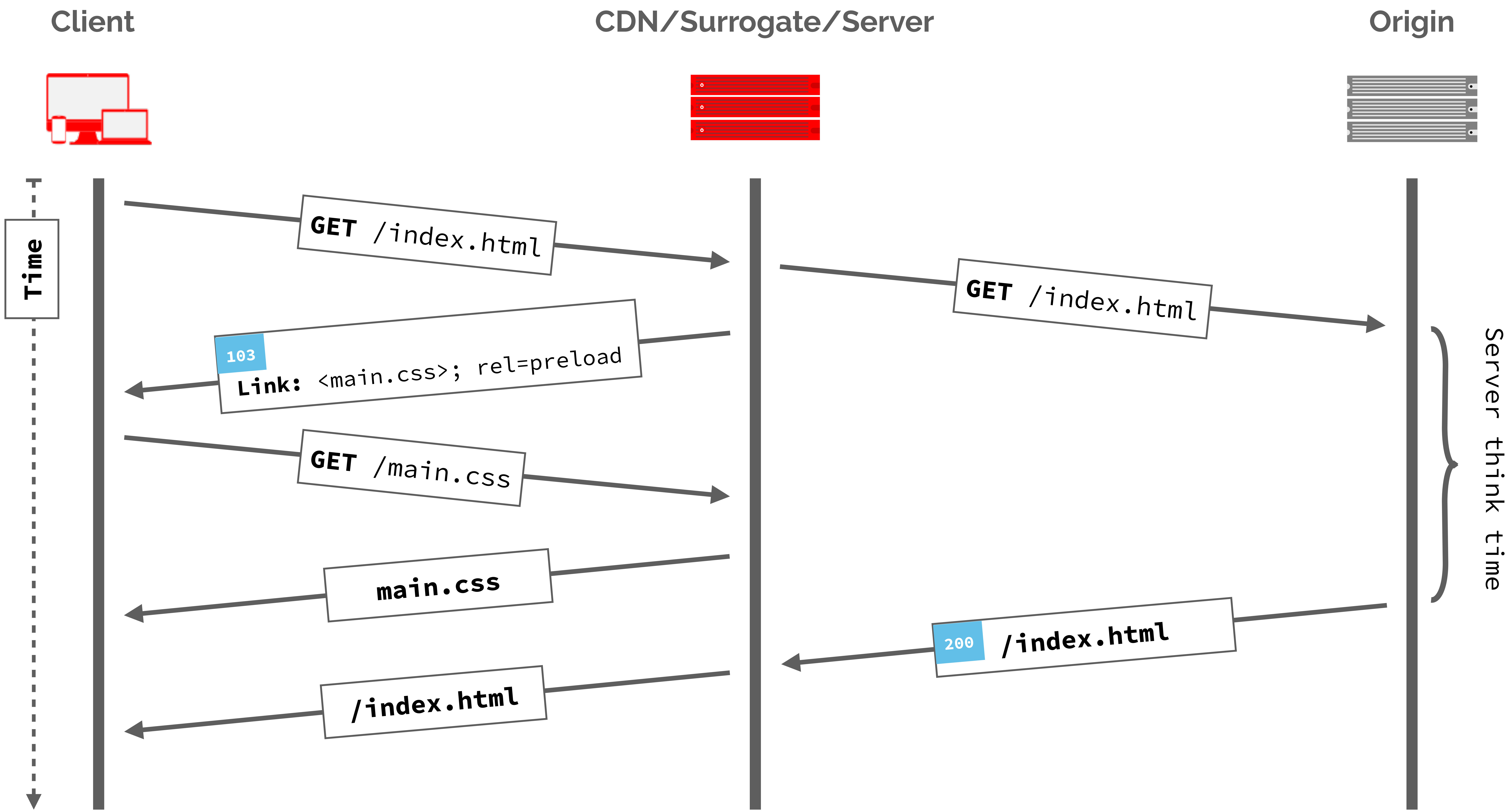
Working Group information can be found at http://httpwg.github.io/ ;

IETF Draft Cache Digests for HTTP/2 - K. Oku, Y. Weiss https://tools.ietf.org/html/draft-ietf-httpbis-cache-digest-04

This still seems too complicated…

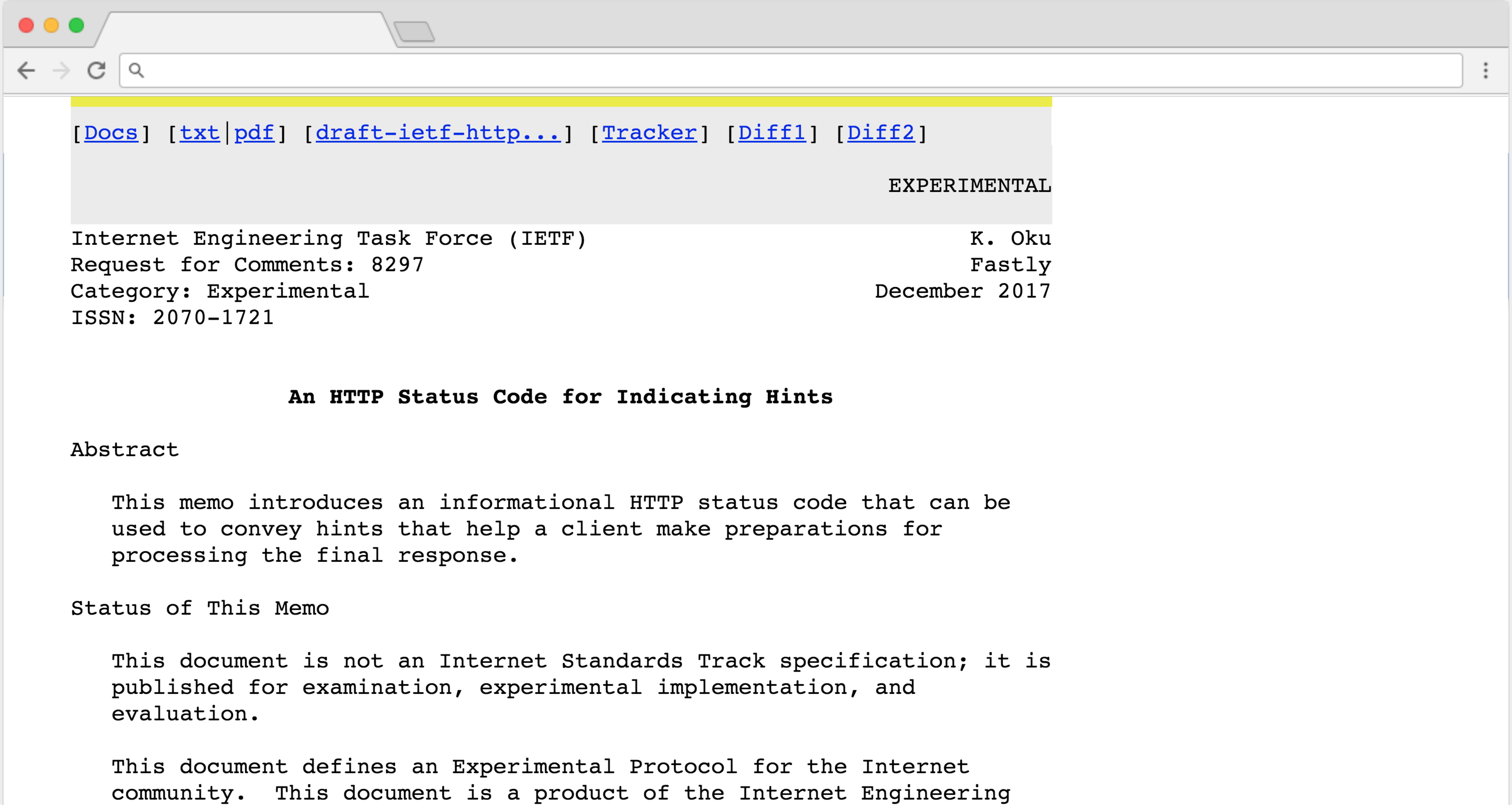
103 Early hints

The 103 (Early Hints) informational status code indicates to the client that the server is likely to send a final response with the header fields included in the informational response.



Client
CDN/Surrogate/Server
Origin
Time
GET /index.html
GET /index.html
103
Link: <main.css>; rel=preload
GET /main.css
main.css
200
/index.html
/index.html
Server think time

```
HTTP/1.1 103 Early Hints
Link: </style.css>; rel=preload; as=style
Link: </main.js>; rel=preload; as=script
Link: </application-data.json>; rel=preload; as=fetch

HTTP/1.1 200 OK
Date: Thu, 14 June 2018 16:10:00 PDT
Content-length: 1234
Content-type: text/html; charset=utf-8
Link: </main.css>; rel=preload; as=style
Link: </newstyle.css>; rel=preload; as=style
Link: </main.js>; rel=preload; as=script
```



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

EXPERIMENTAL

Internet Engineering Task Force (IETF)　　K. Oku
Request for Comments: 8297　　Fastly
Category: Experimental　　December 2017
ISSN: 2070-1721

An HTTP Status Code for Indicating Hints

Abstract

This memo introduces an informational HTTP status code that can be used to convey hints that help a client make preparations for processing the final response.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering

IETF RFC8297 Early Hints - K. Oku https://tools.ietf.org/html/rfc8297

103 Early hints:

✅ Same benefits as push

✅ Much simpler

✅ Leverages the browser caches

✅ Allows client to initiate fetches

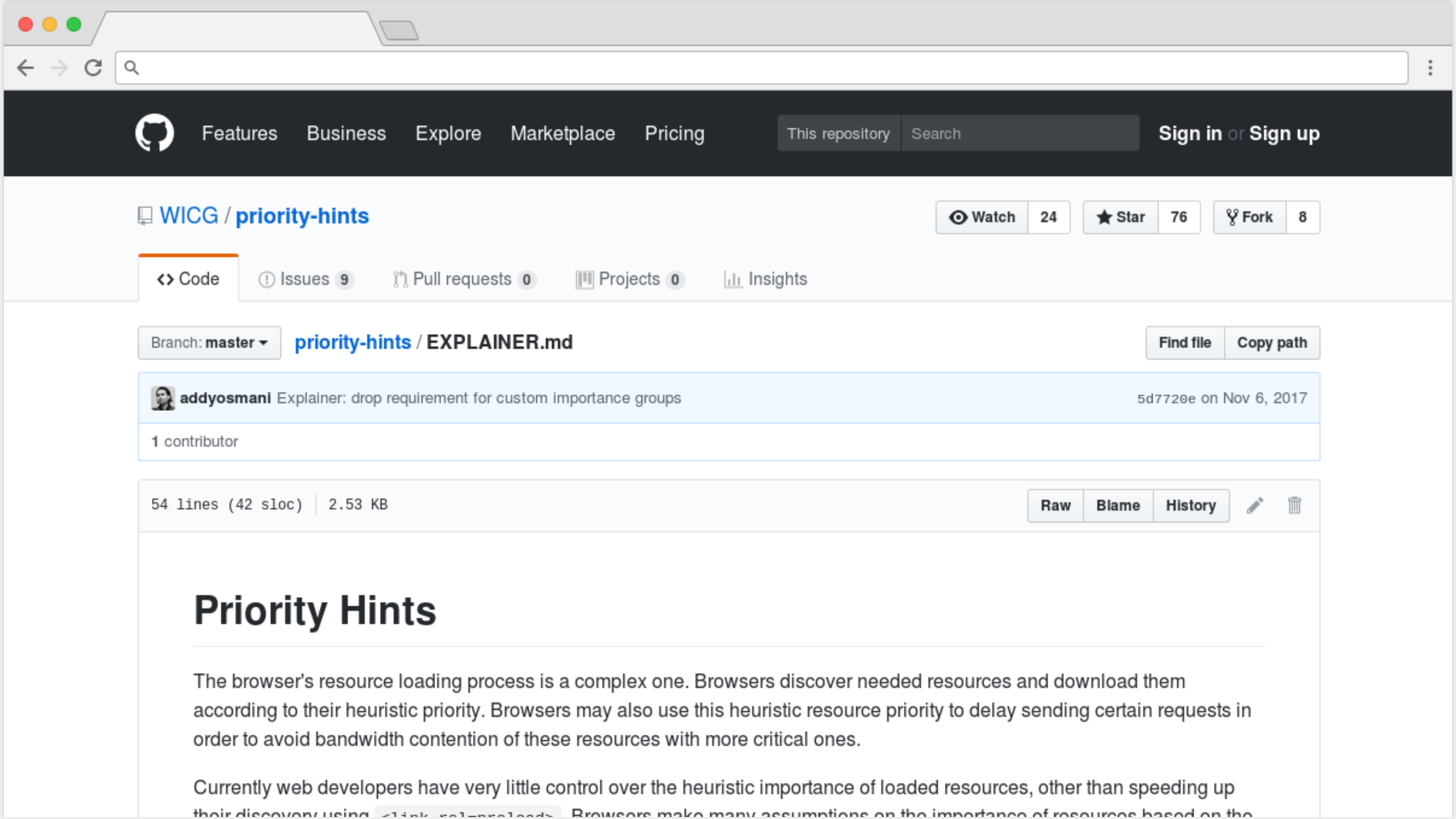
⚠️ No 1 RTT saving

Priority hints

```
<link rel="preload" as="script" href="critical-script.js">
<link rel="preload" as="style" href="theme.css" importance="low" onload="this.rel=stylesheet">

<style>/* critical-path styles */</style>
<img src="hero.jpg" importance="high">
<img src="meme.gif" importance="low">

<!-- superfluous fetch requests -->
<script>
  fetch('/api/related.json', { importance: 'low' });
</script>

<!-- scripts at the end of the document -->
<script src="critical-script.js"></script>
```

github.com/WICG/priority-hints

Web performance made easy: Addy Osmani, Ewa Gasperowicz https://youtu.be/Mv-l3-tJgGk



addyosmani Explainer: drop requirement for custom importance groups 5d7720e on Nov 6, 2017

1 contributor

54 lines (42 sloc) | 2.53 KB

Priority Hints

The browser's resource loading process is a complex one. Browsers discover needed resources and download them according to their heuristic priority. Browsers may also use this heuristic resource priority to delay sending certain requests in order to avoid bandwidth contention of these resources with more critical ones.

Currently web developers have very little control over the heuristic importance of loaded resources, other than speeding up their discovery using `<link rel=preload>`. Browsers make many assumptions on the importance of resources based on the

Priority Hints draft - https://github.com/WICG/priority-hints

Closing

HTTP/2 doesn’t solve everything.

Resource loading is hard.

Performance is for humans.
Optimise for user experiences.

The future is bright!

Resource loading checklist:

- ✅ Identify your critical resources
- ✅ Preload hidden sub-resources
- ✅ Preconnect critical third-parties
- ❌ Avoid pushing with preload
- ⚠️ Use async push with care
- 🚀 Decorate HTML with priority hints
- 🚀 Use Early Hints when available

Thanks!

Patrick Hamann
speakerdeck.com/patrickhamann

patrick@fastly.com
@patrickhamann