To push, or not to push?!

A journey of resource loading in the browser

JSConf EU, June 2018

Patrick Hamann
@patrickhamann


fastly

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.

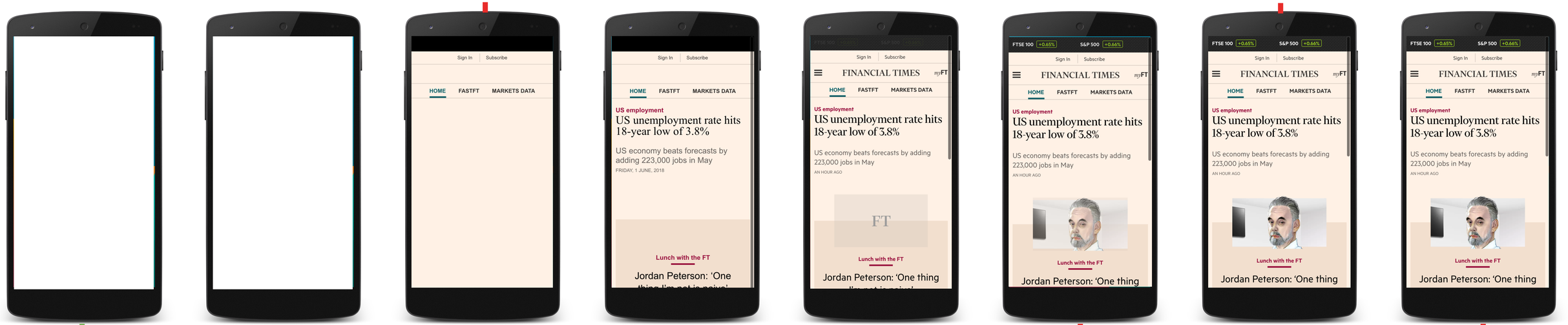
– Ben Schwarz, Calibre

What are my critical resources?

- ✅ Critical CSS for current route
- ✅ Fonts
- ✅ Hero images
- ✅ Initial application route
- ✅ Application bootstrap data

First Contentful Paint

Time to Interactive



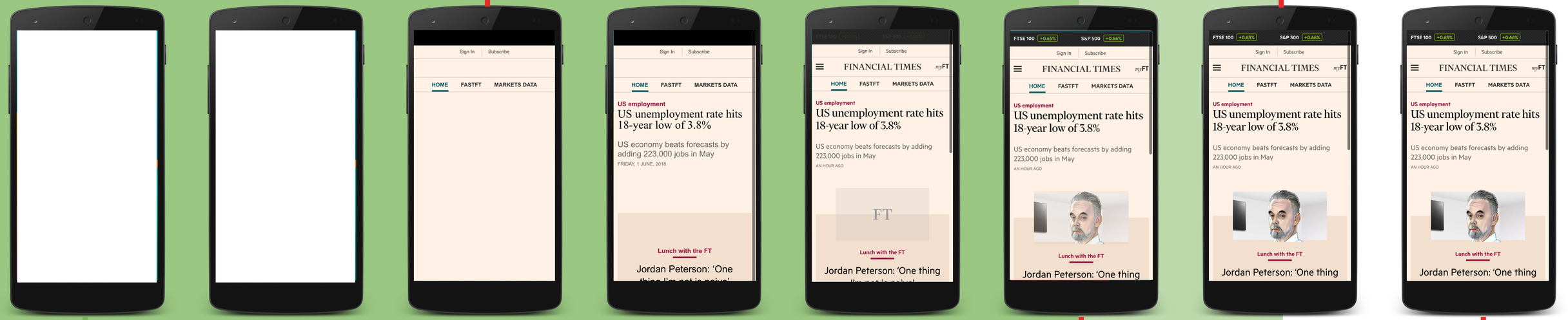
User navigates

First Meaningful Paint

Visually Complete

First Contentful Paint

Time to Interactive



User navigates

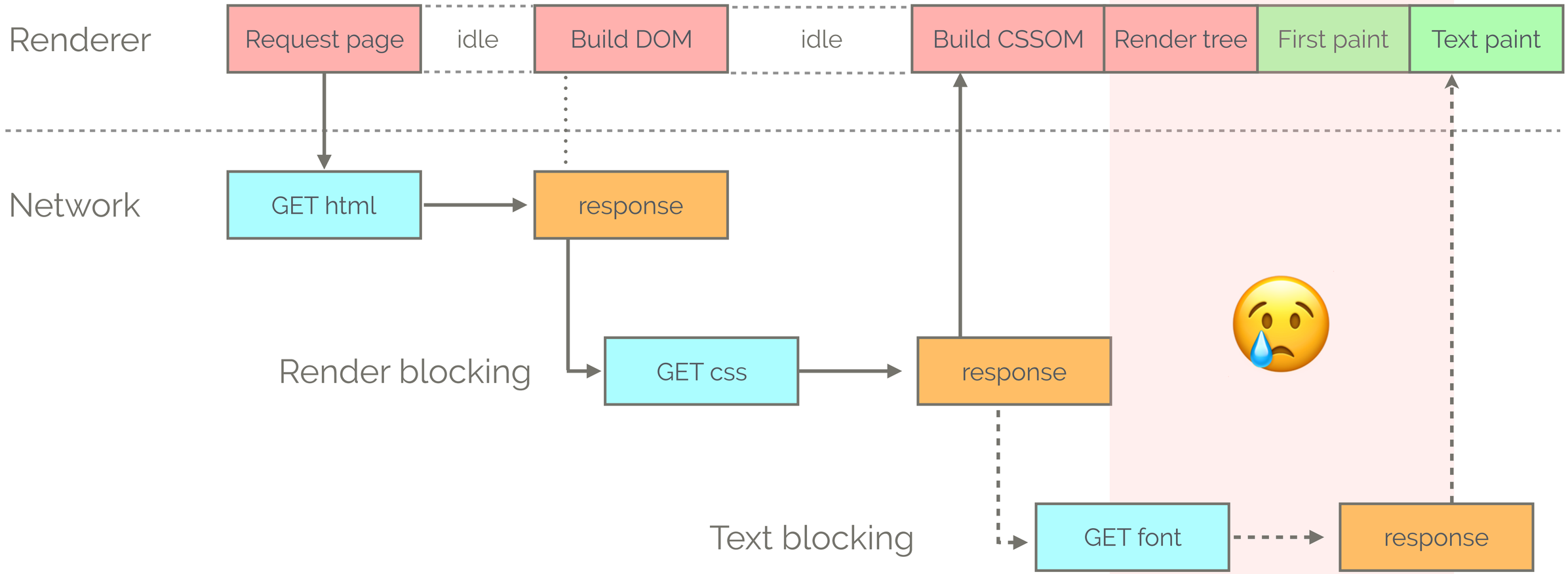
First Meaningful Paint

Visually Complete

A good loading strategy:

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

Preload


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

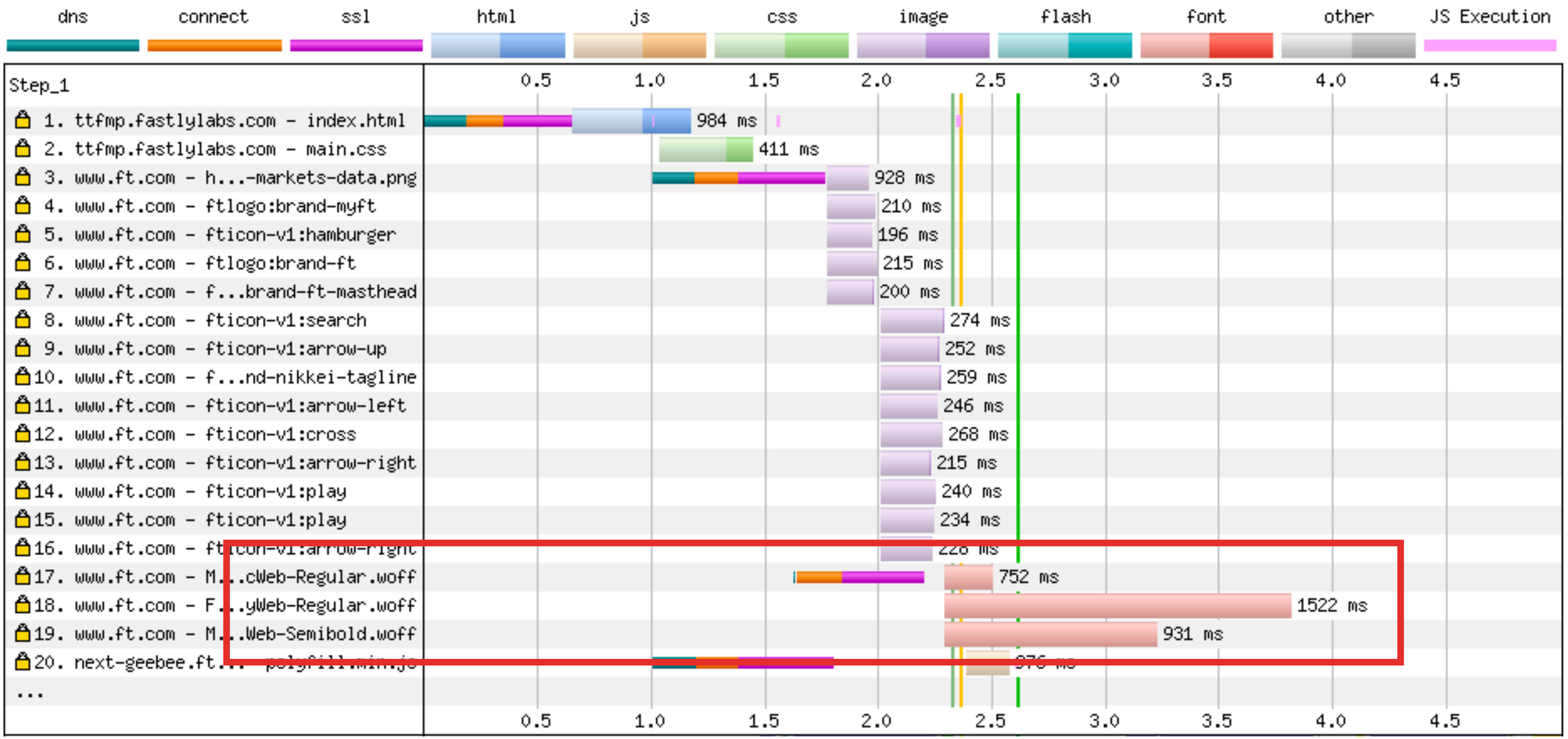
Provides a declarative fetch primitive that initiates an early fetch and separates fetching from resource execution.

Preload with HTTP header:

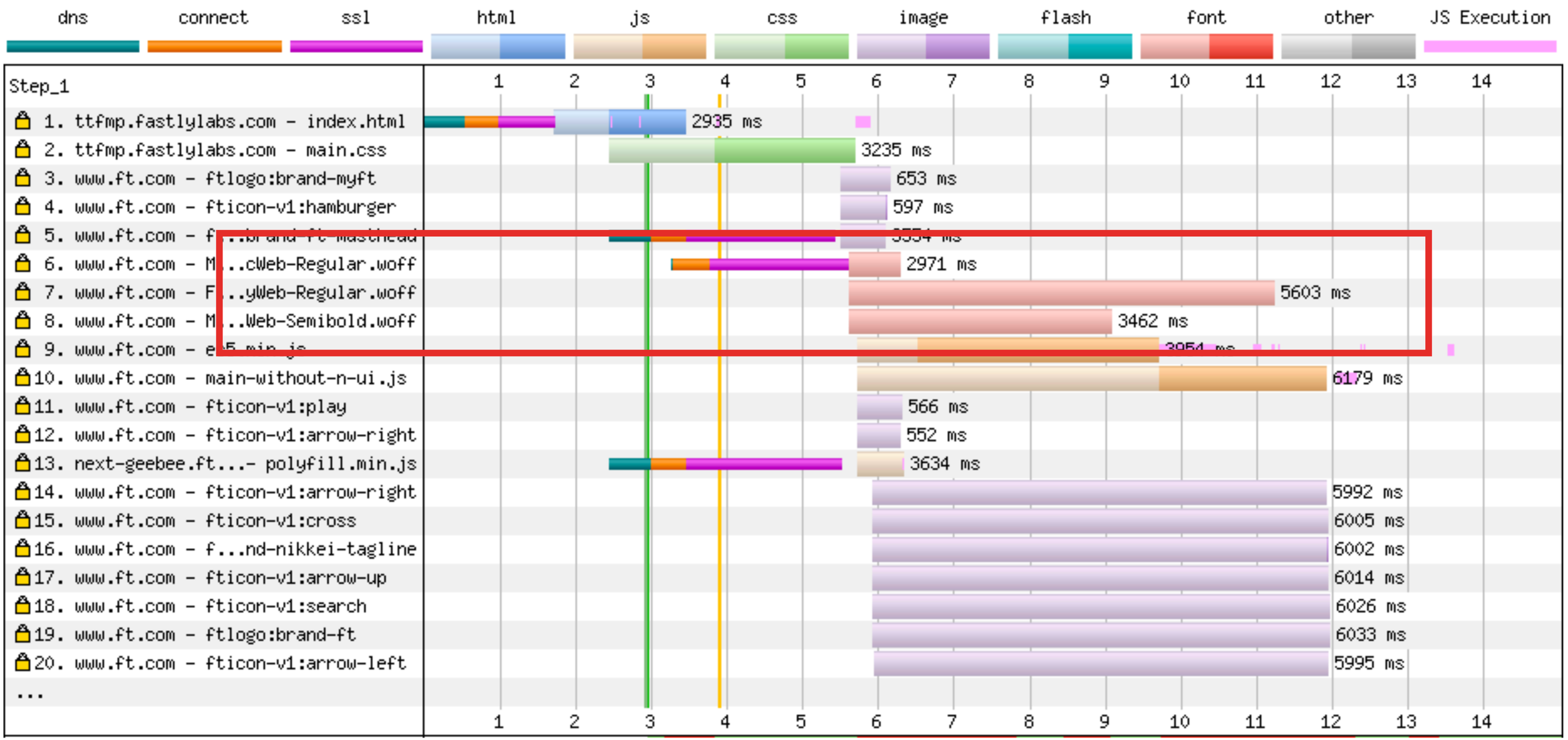
```
Link: <my-awesome-font.woff>; rel=preload; as=font; crossorigin
Link: <application-data.json>; rel=preload; as=fetch;
Link: <sub-module.mjs>; rel=modulepreload;
```

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>
```


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
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...
...

Before


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

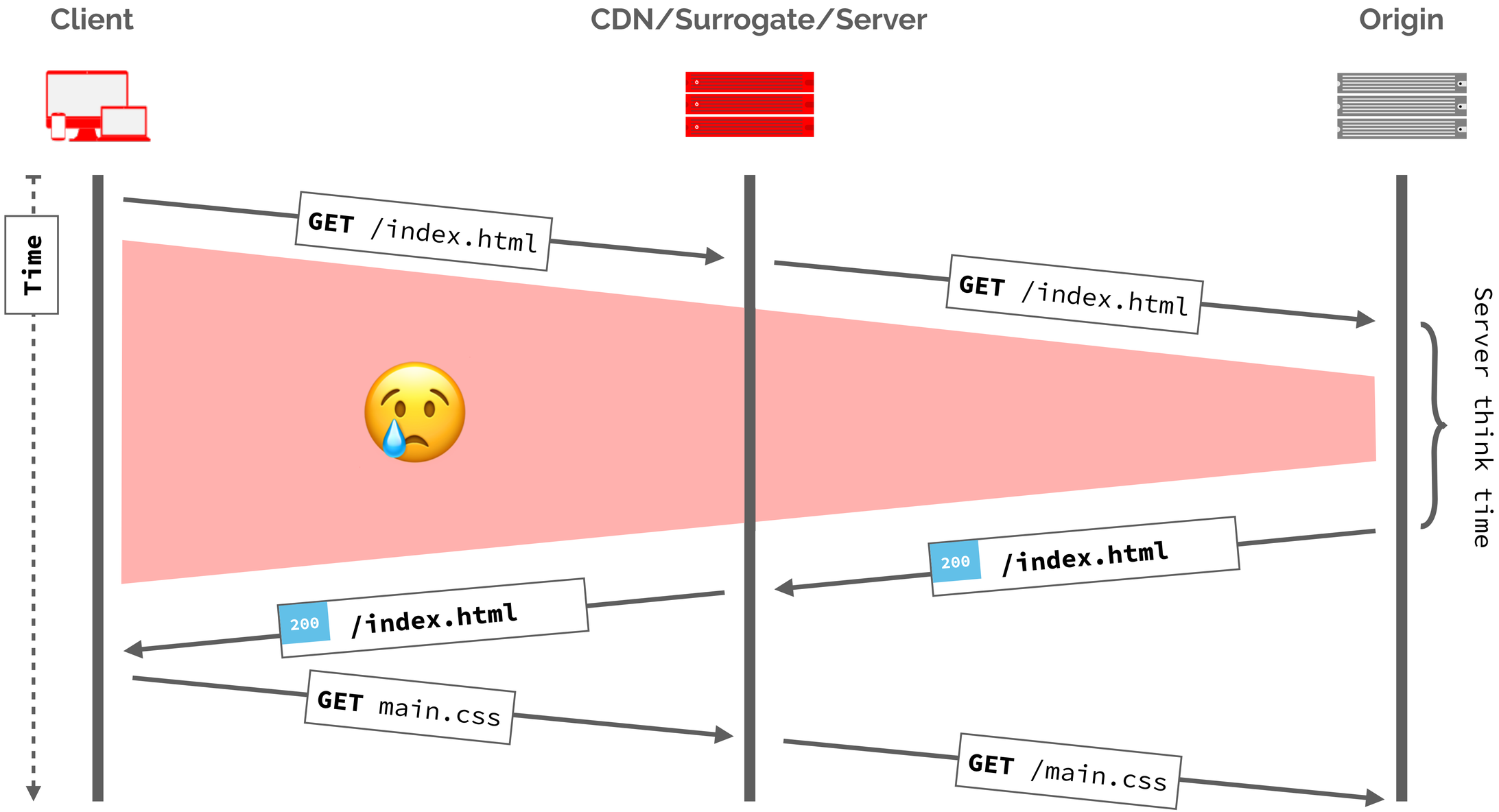
After

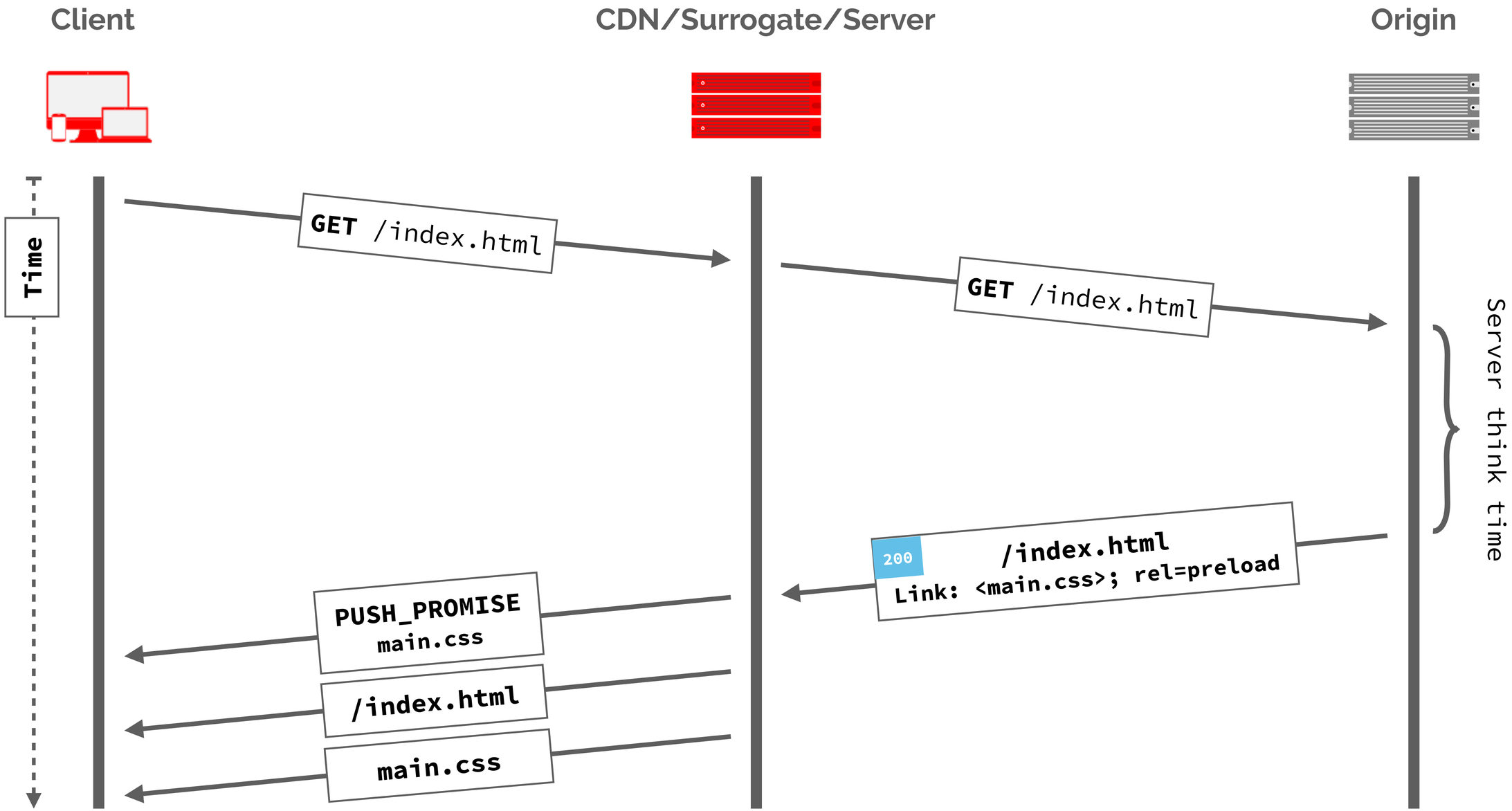
Shopify's switch to preloading fonts saw a 50% (1.2 second) improvement in time-to-text-paint. This removed their flash-of-invisible text completely.

– Shopify

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


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
```

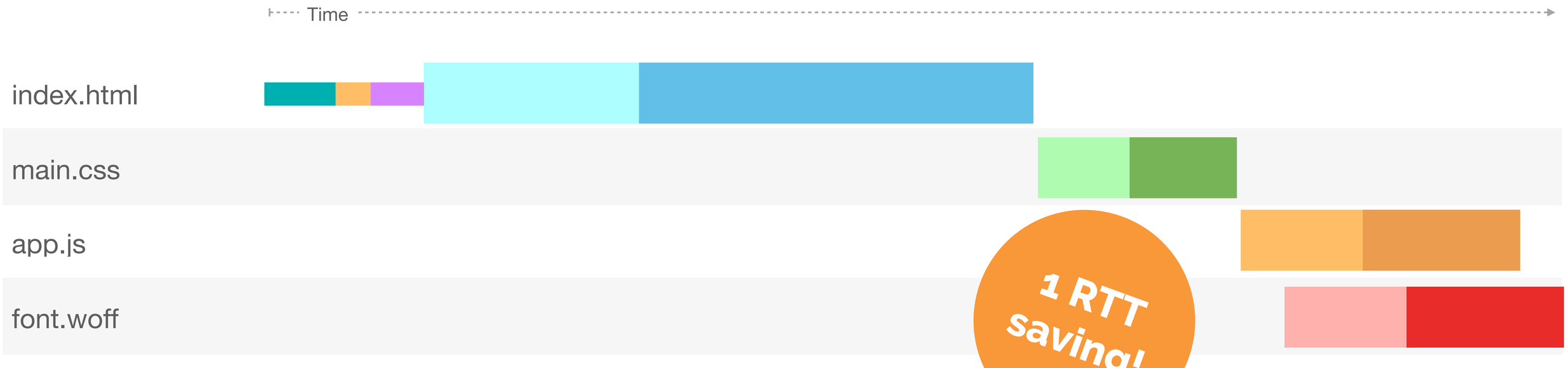
Use no push attribute to disable push semantics and only use preload.

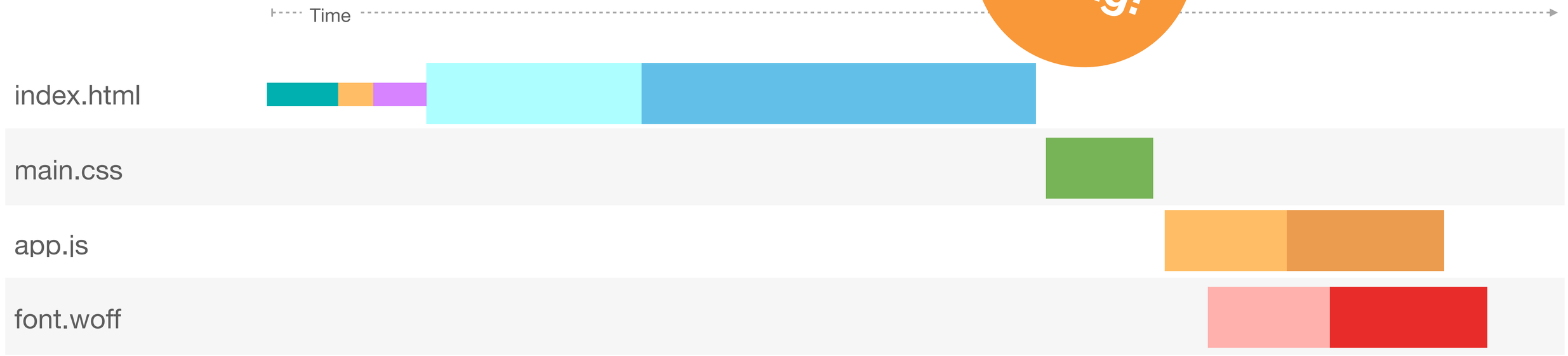
```
Link: <main.css>; rel=preload; as=style; nopush
```

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

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

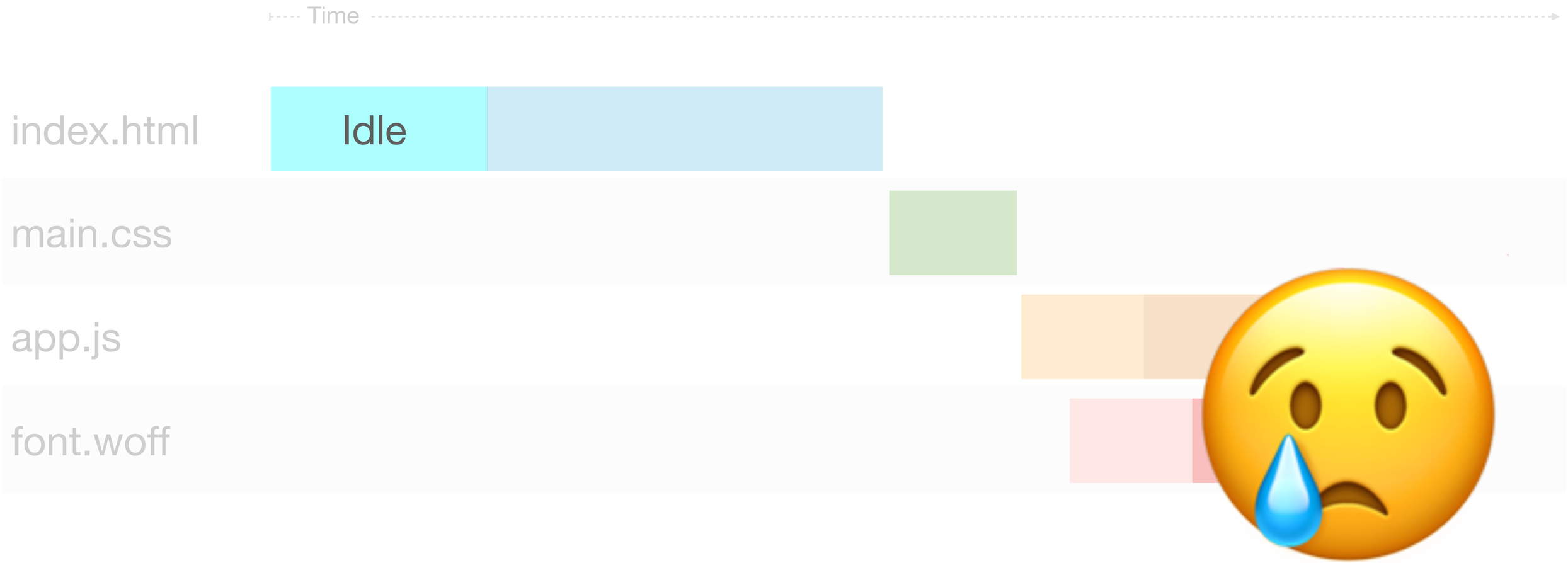
No Push



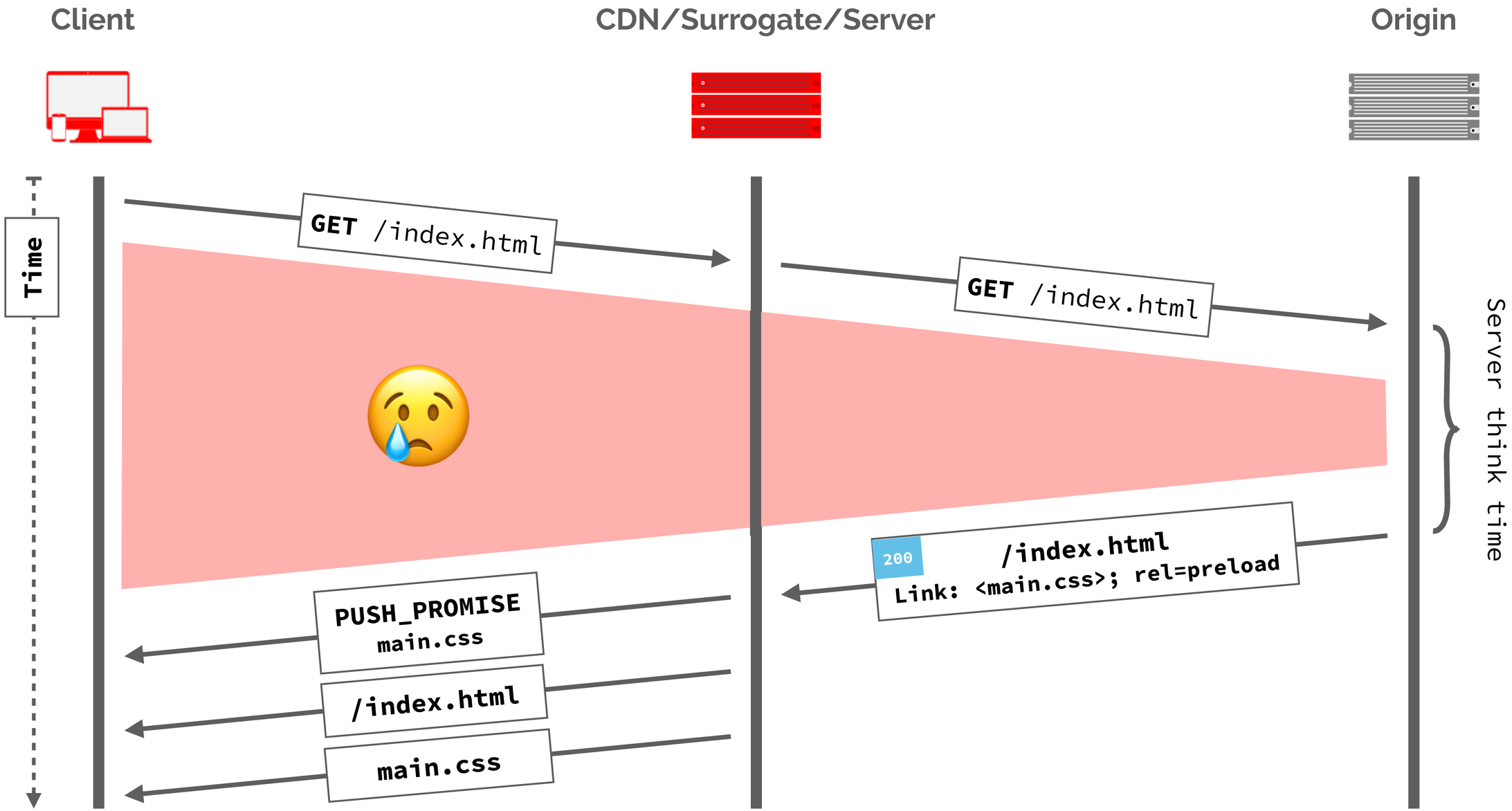


1 RTT saving!

Push



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

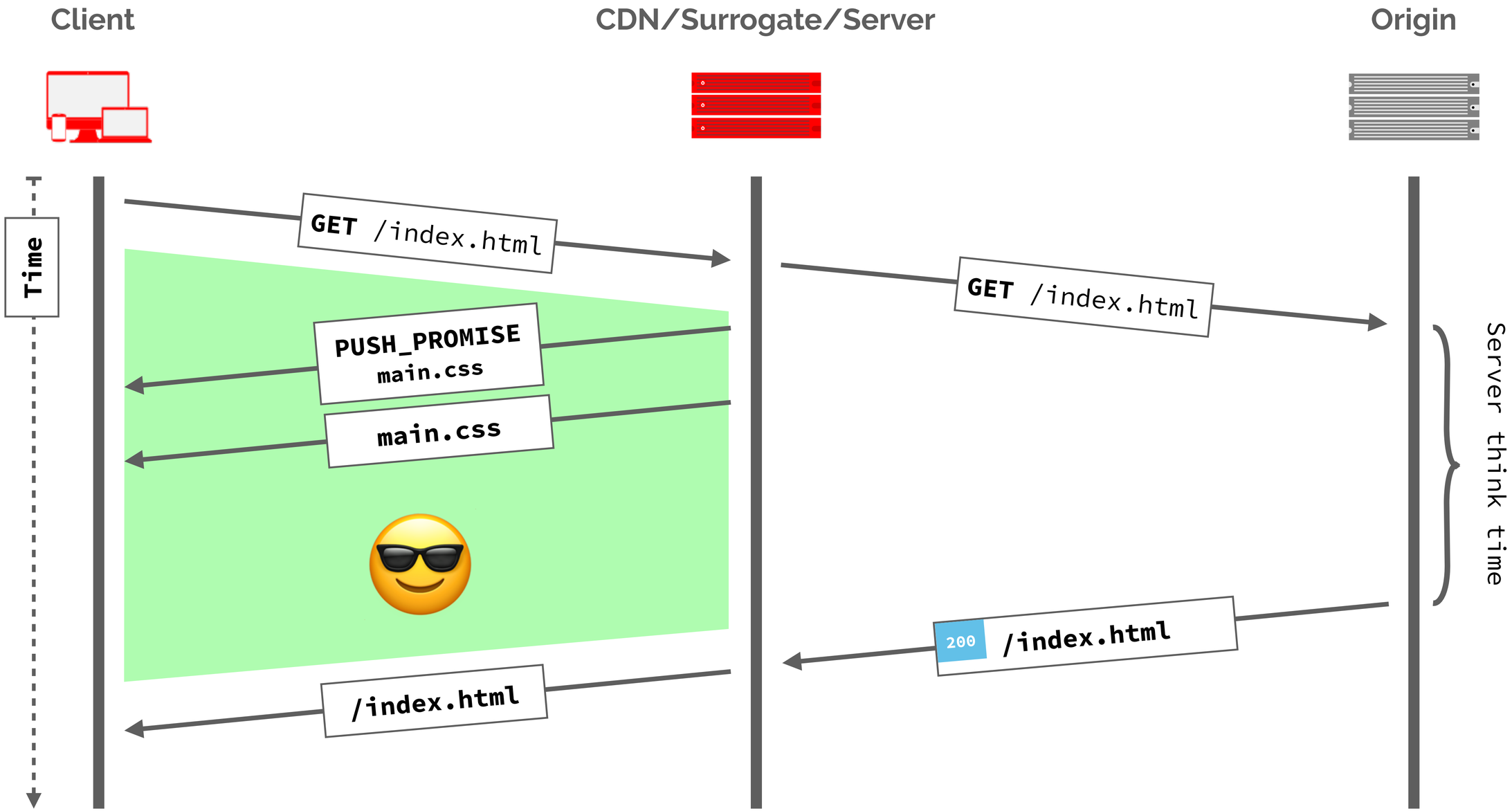
✅ Useful for long server think time

✅ Useful for long RTT times

⚠️ Link header indication is too late

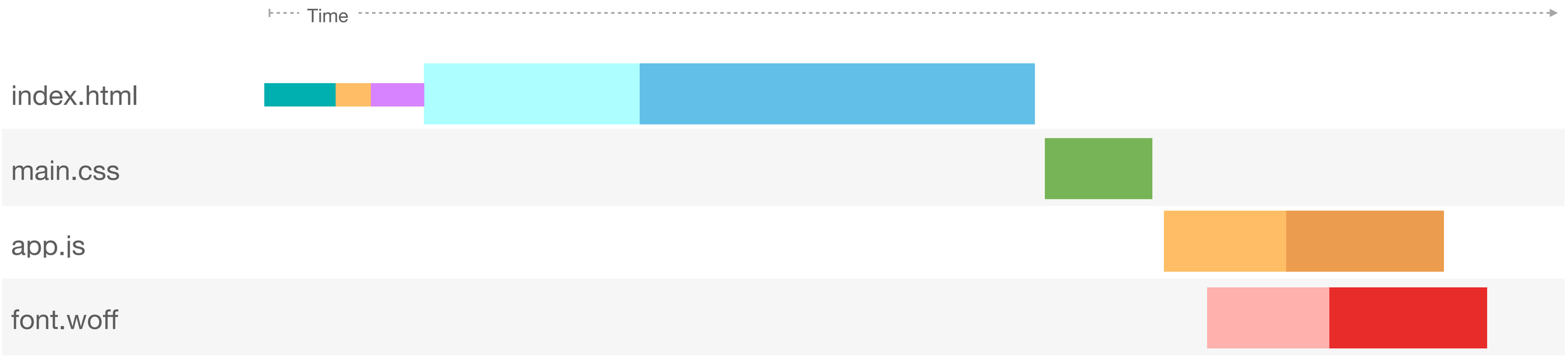
Is indicating push via the HTML response too late?

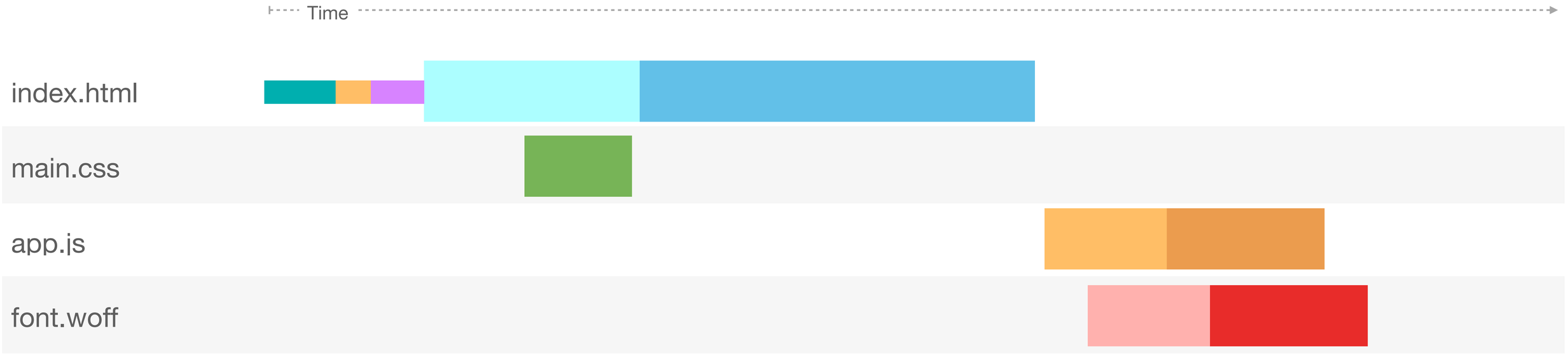
Async push


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

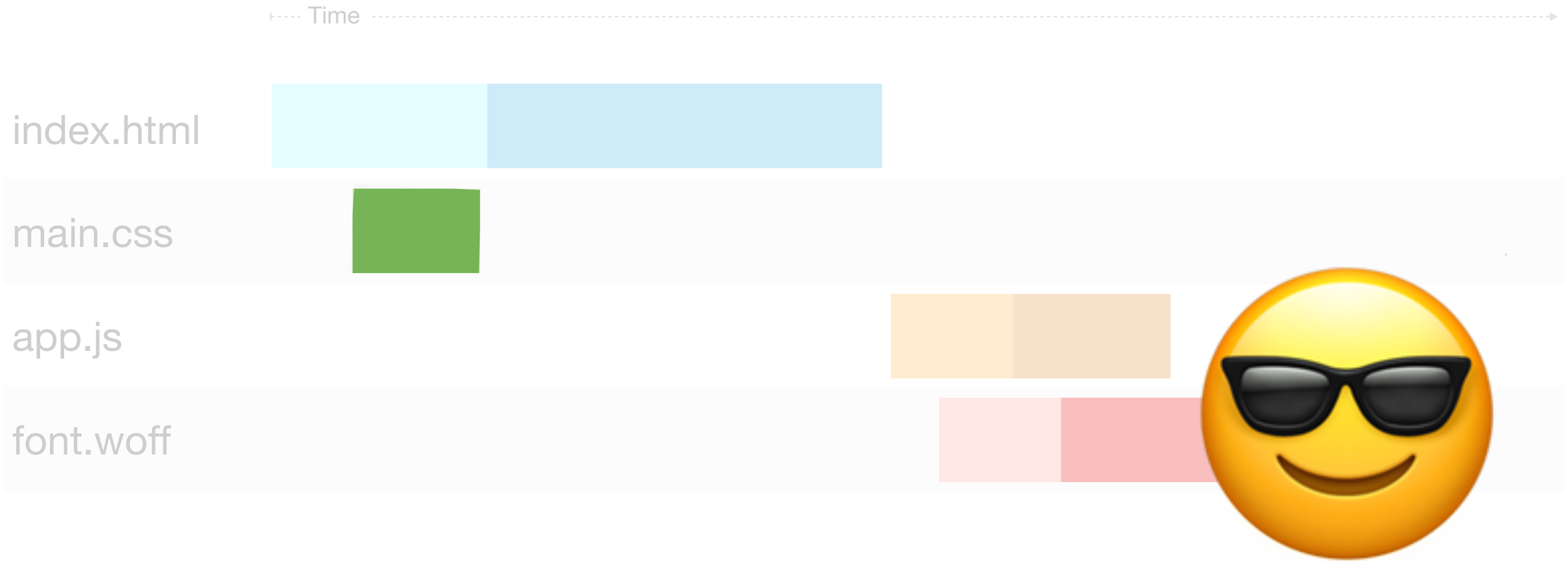
```
const http2 = require('http2');

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);
```

Push





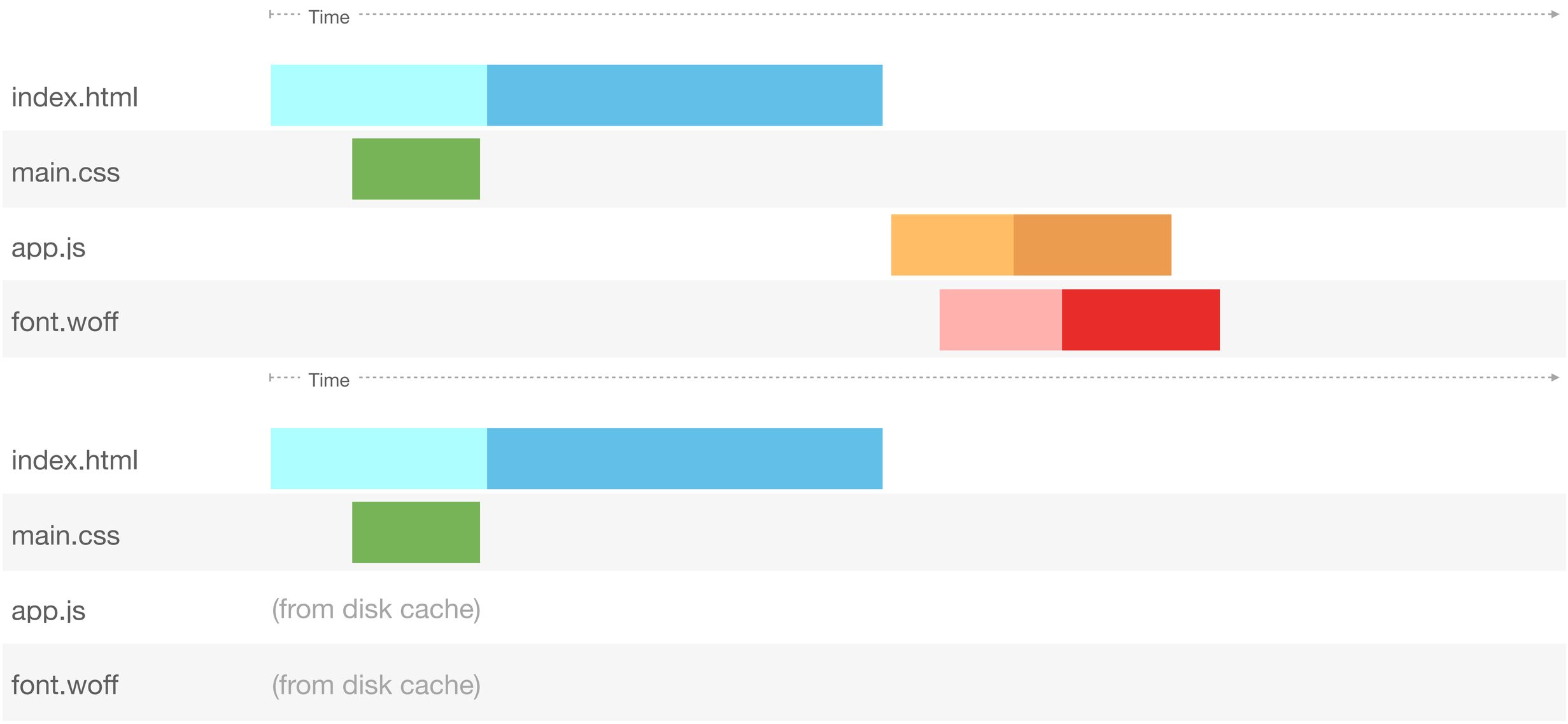
Async push



Utilising idle network server think time == win!

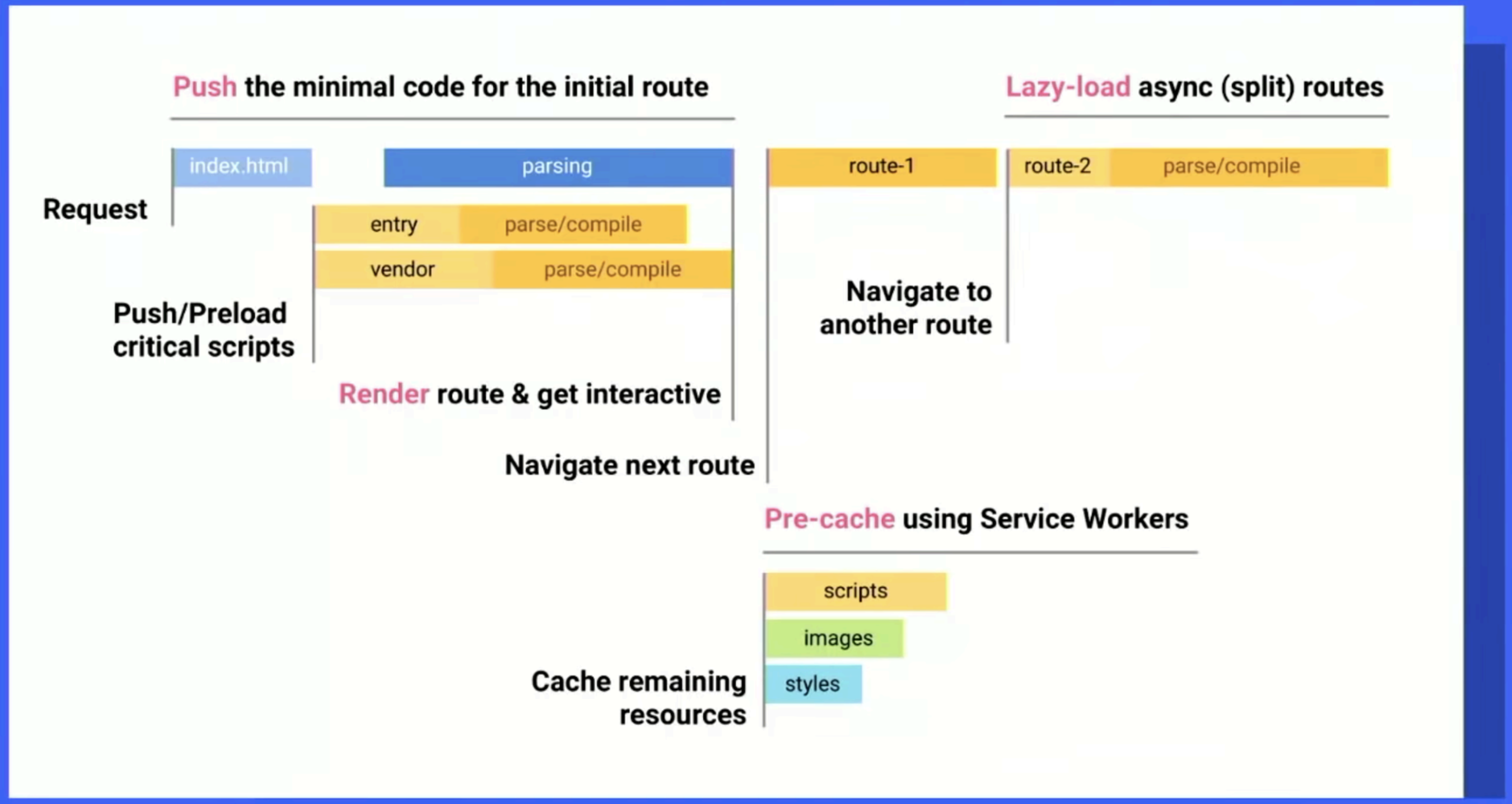
What about the repeat view?

First view



Repeat view

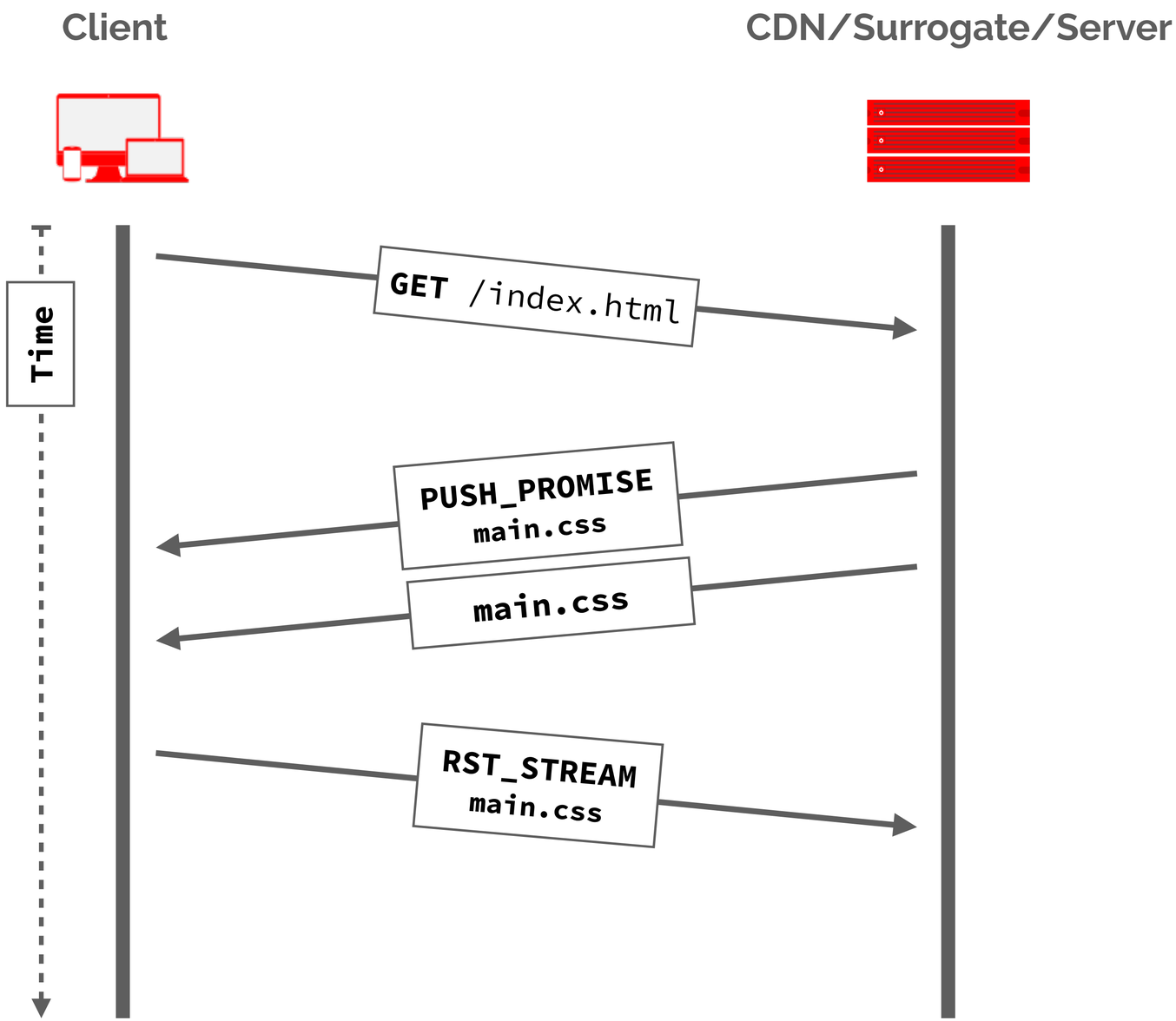
PRPL Pattern

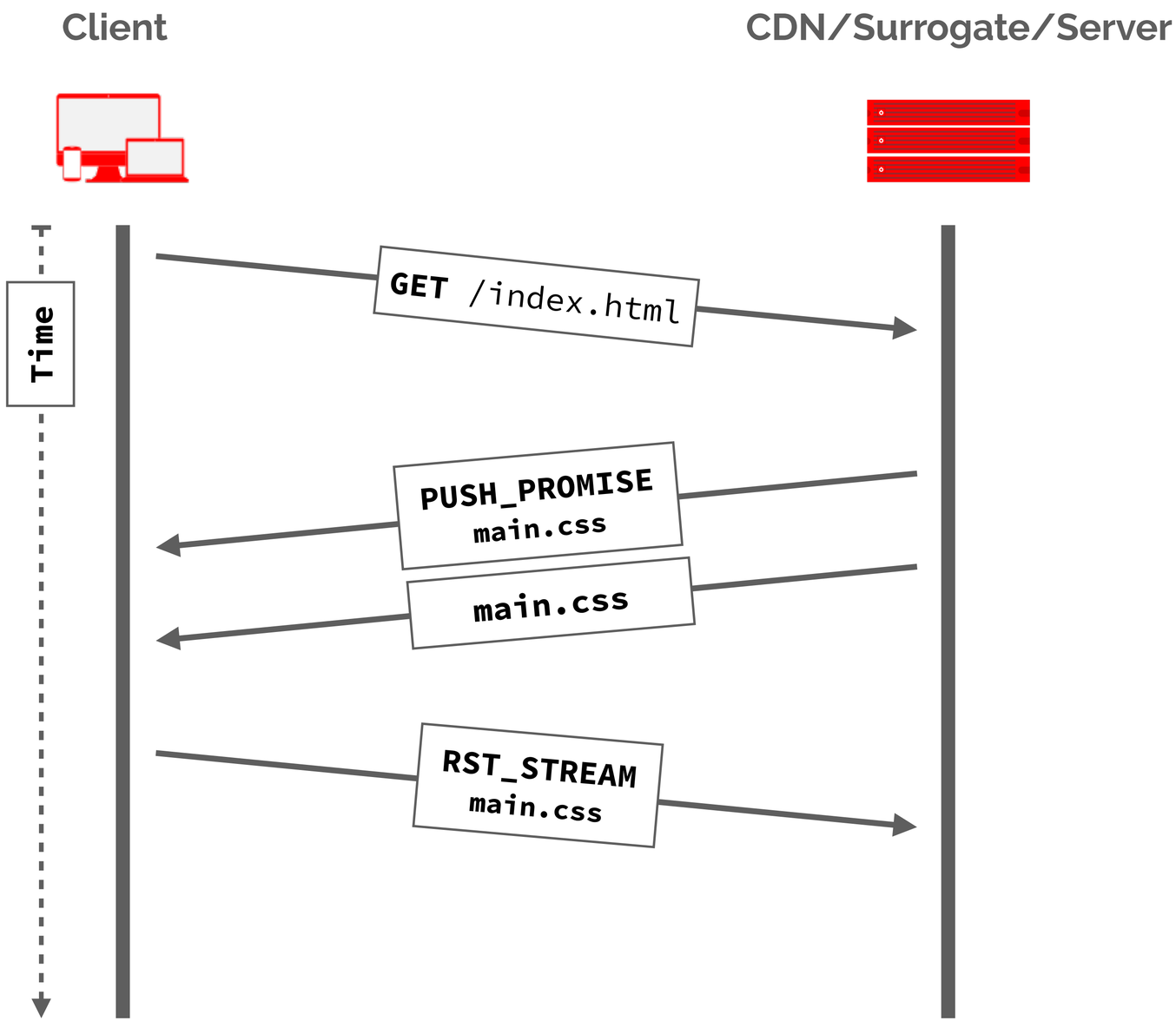


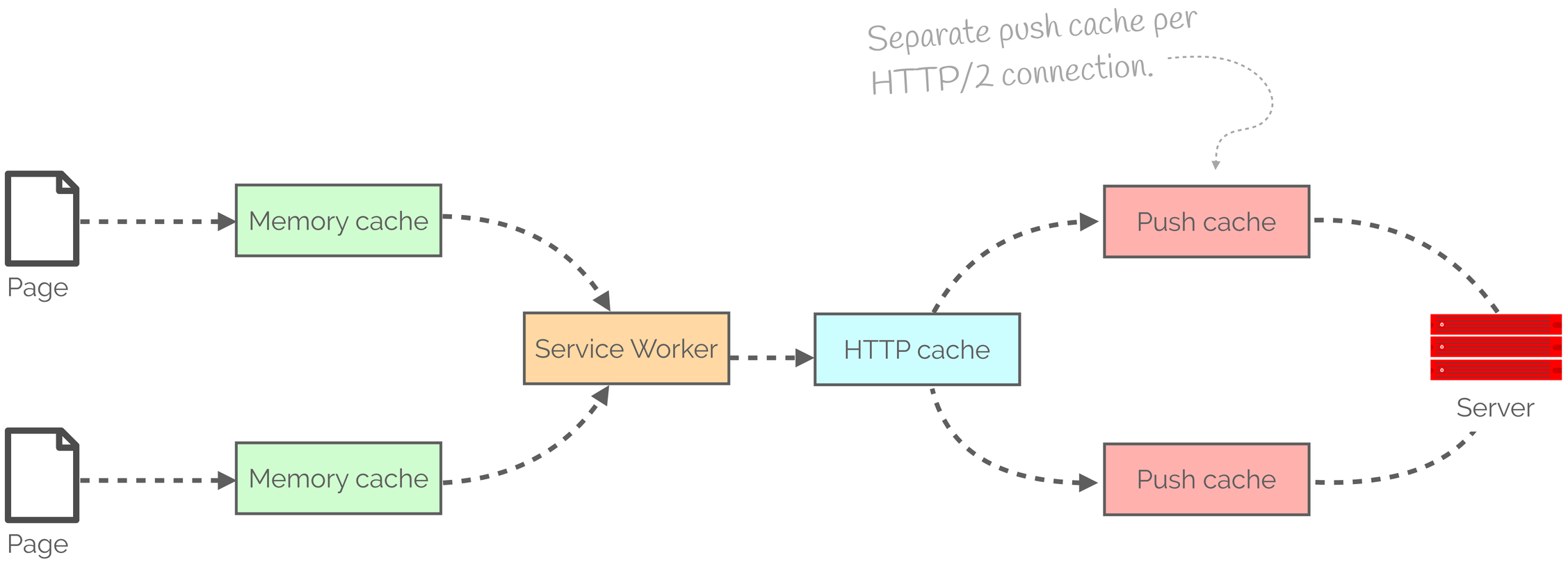
Nikkei.com - Web performance made easy: Addy Osmani, Ewa Gasperowicz https://youtu.be/Mv-l3-tJgGk

The server has no knowledge of client cache state.

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


Separate push cache per HTTP/2 connection.
Page
Memory cache
Service Worker
HTTP cache
Push cache
Server
Page
Memory cache
Push cache

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

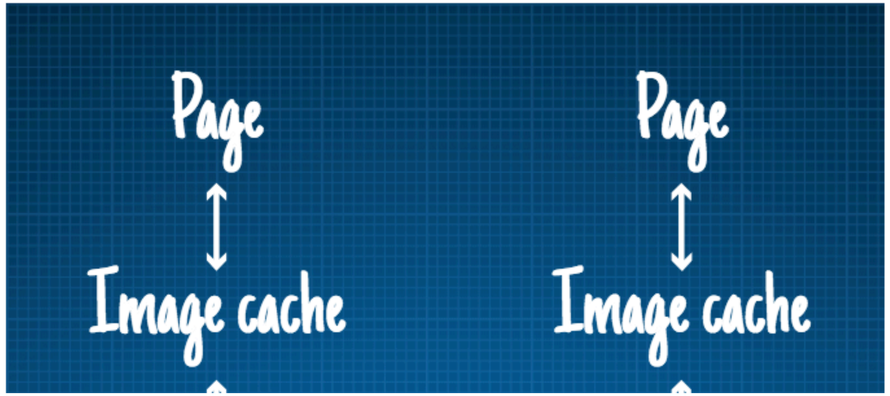
"HTTP/2 push will solve that" is something I've heard a lot when it comes to page load performance problems, but I didn't know much about it, so I decided to dig in.

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 (native, Electron etc)

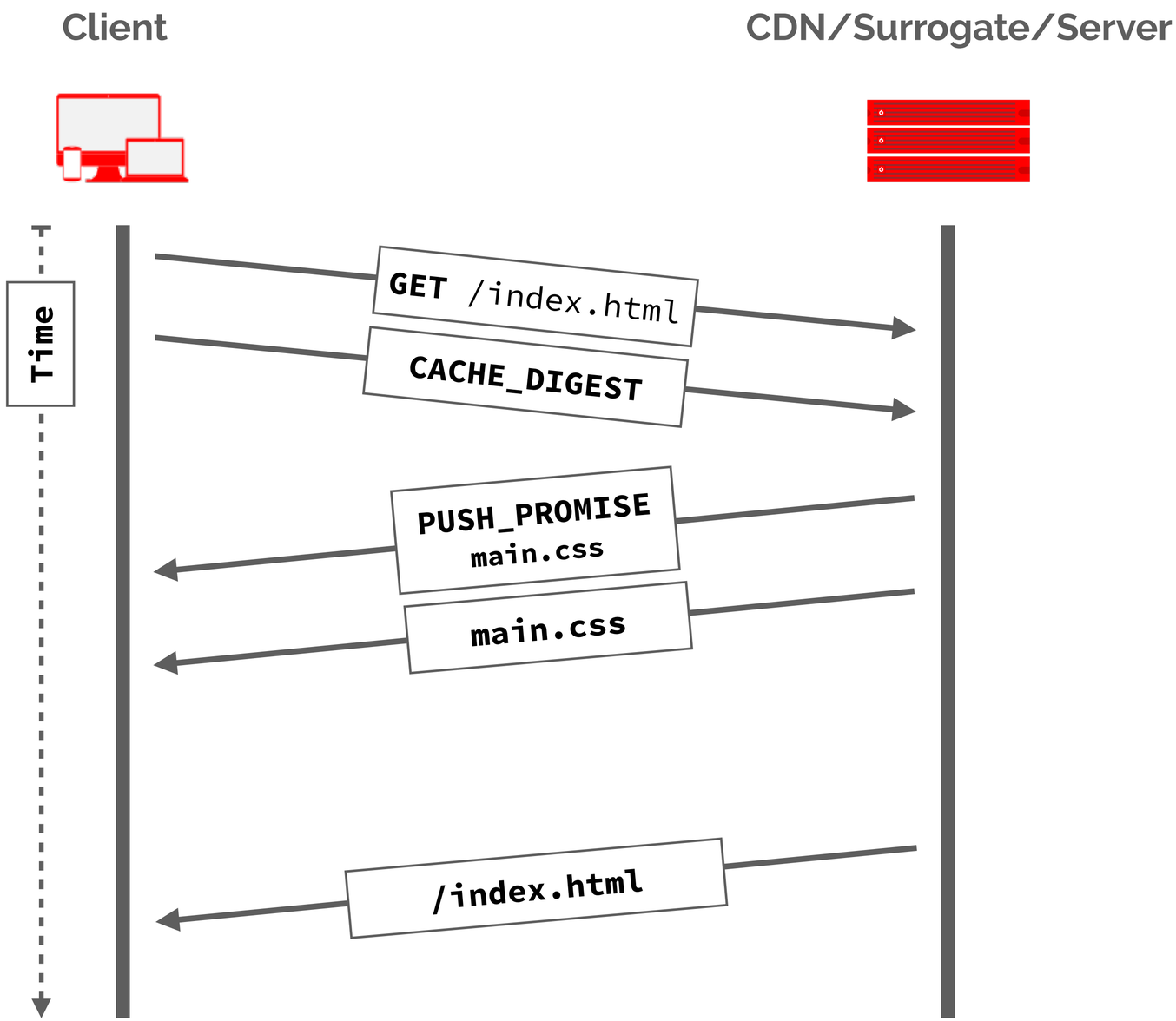
Is the 1 RTT saving worth the complexity?

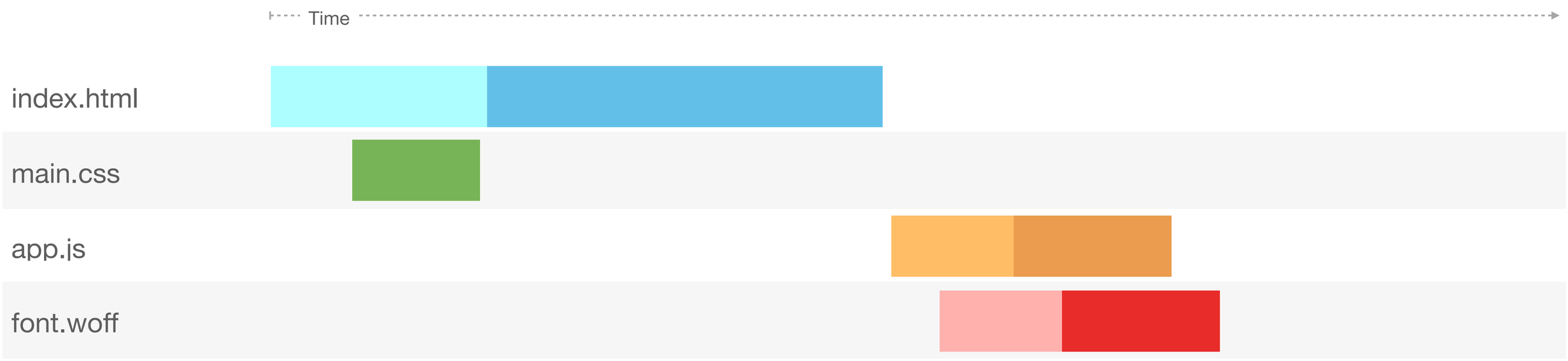
Are there other solutions?

The future

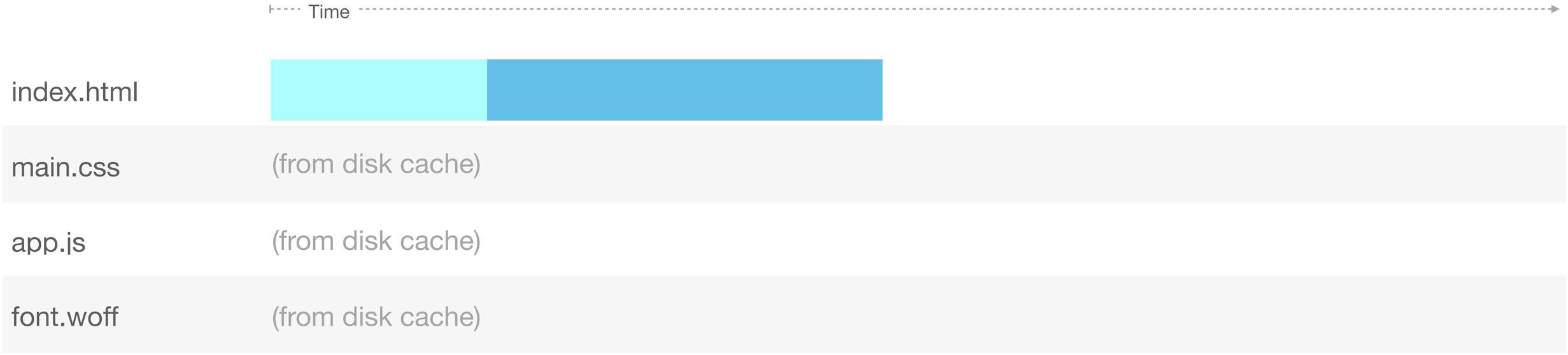
Can we fix the problems with push?

Cache digests

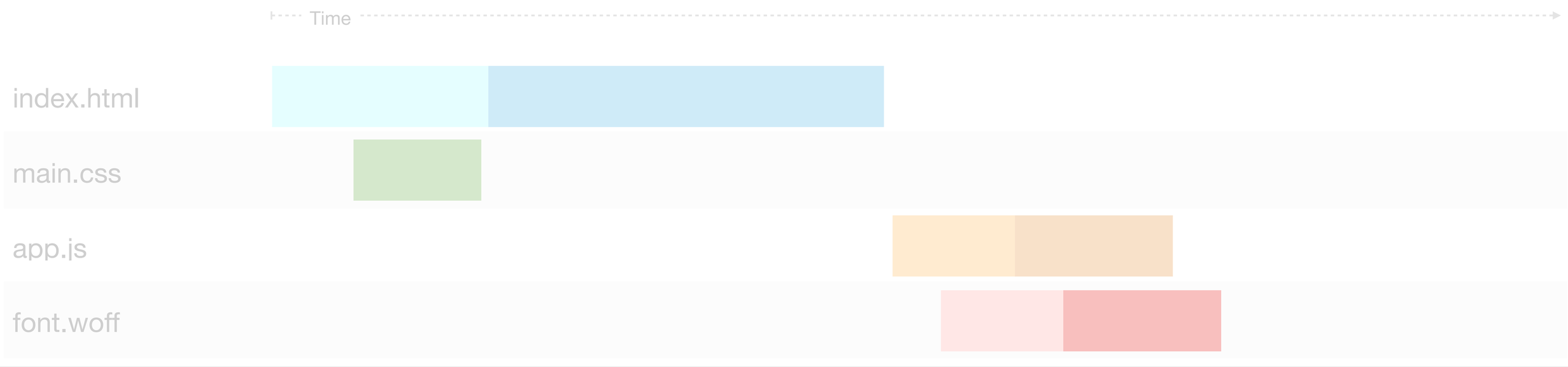

Client
CDN/Surrogate/Server
Time
GET /index.html
CACHE_DIGEST
PUSH_PROMISE
main.css
main.css
/index.html



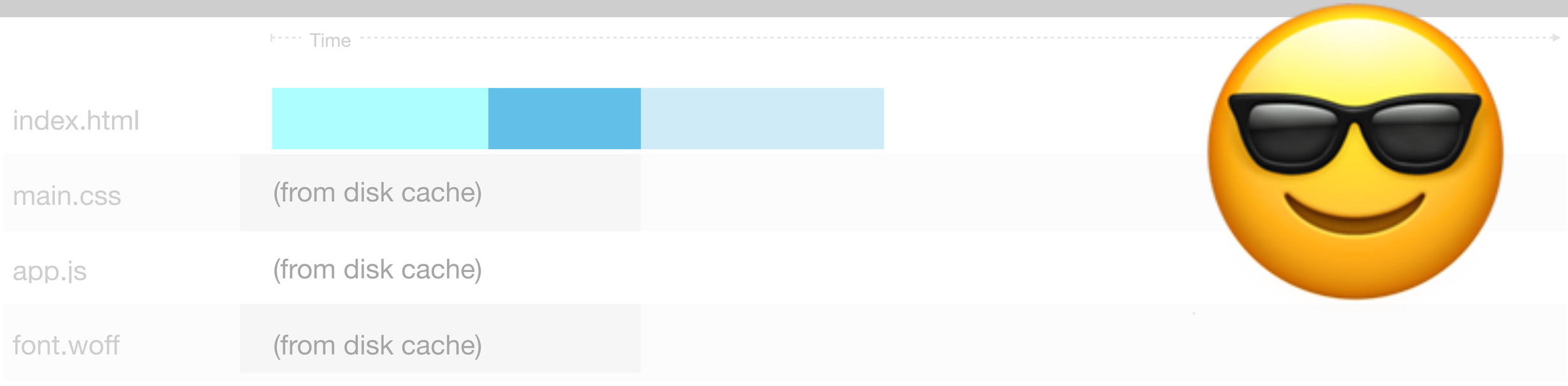
First view



Repeat view



First view



Repeat view

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

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

HTTP Working Group
Internet-Draft
Intended status: Experimental
Expires: October 8, 2018

K. Oku
Fastly
Y. Weiss
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/ .

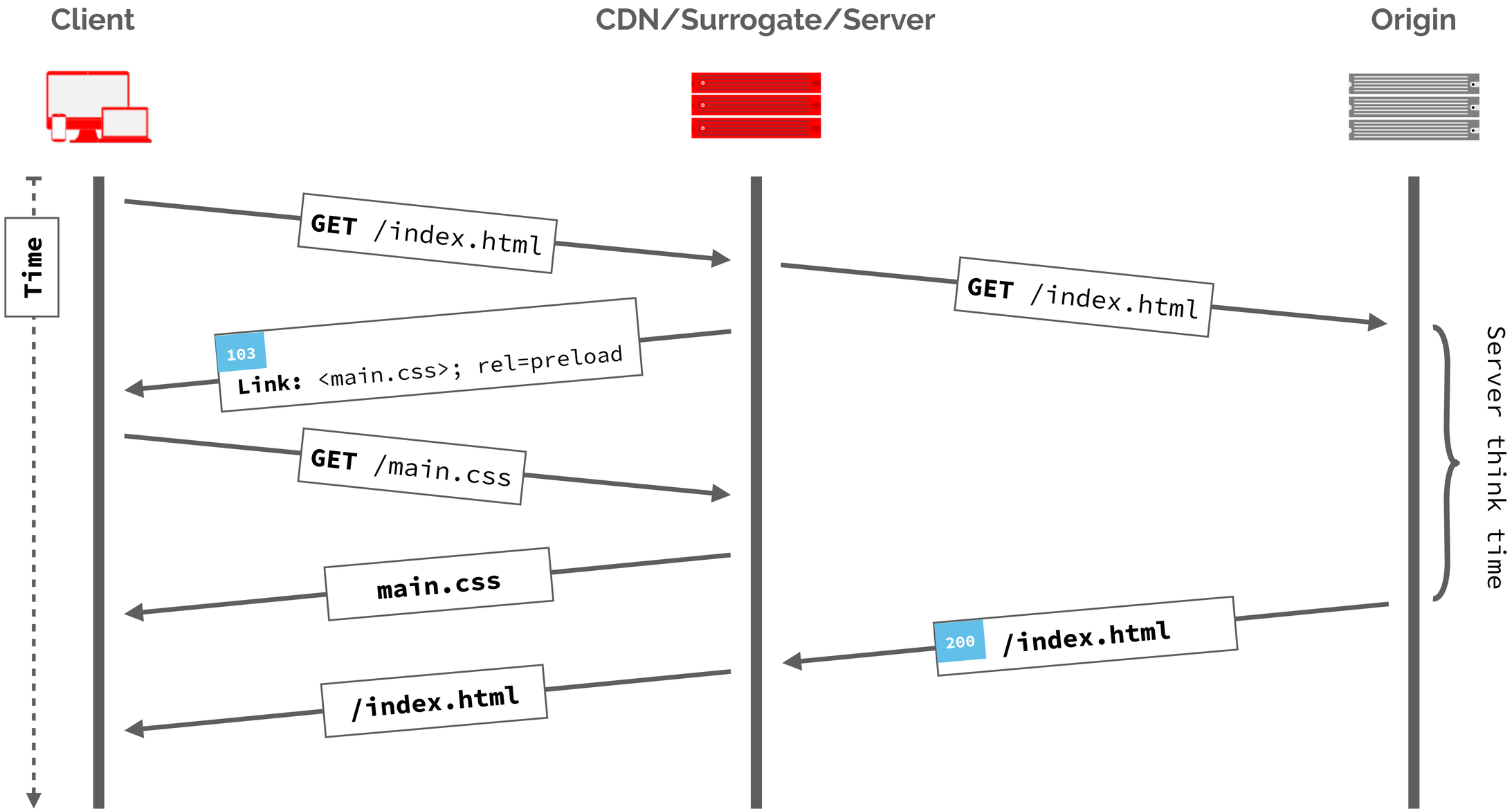
Working Group information can be found at http://httpwg.github.io/ .

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: Sun, 03 June 2018 14:40:00 CET
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

103 Early hints:

✅ Same benefits as push

✅ Much simpler

✅ Leverages the browser caches

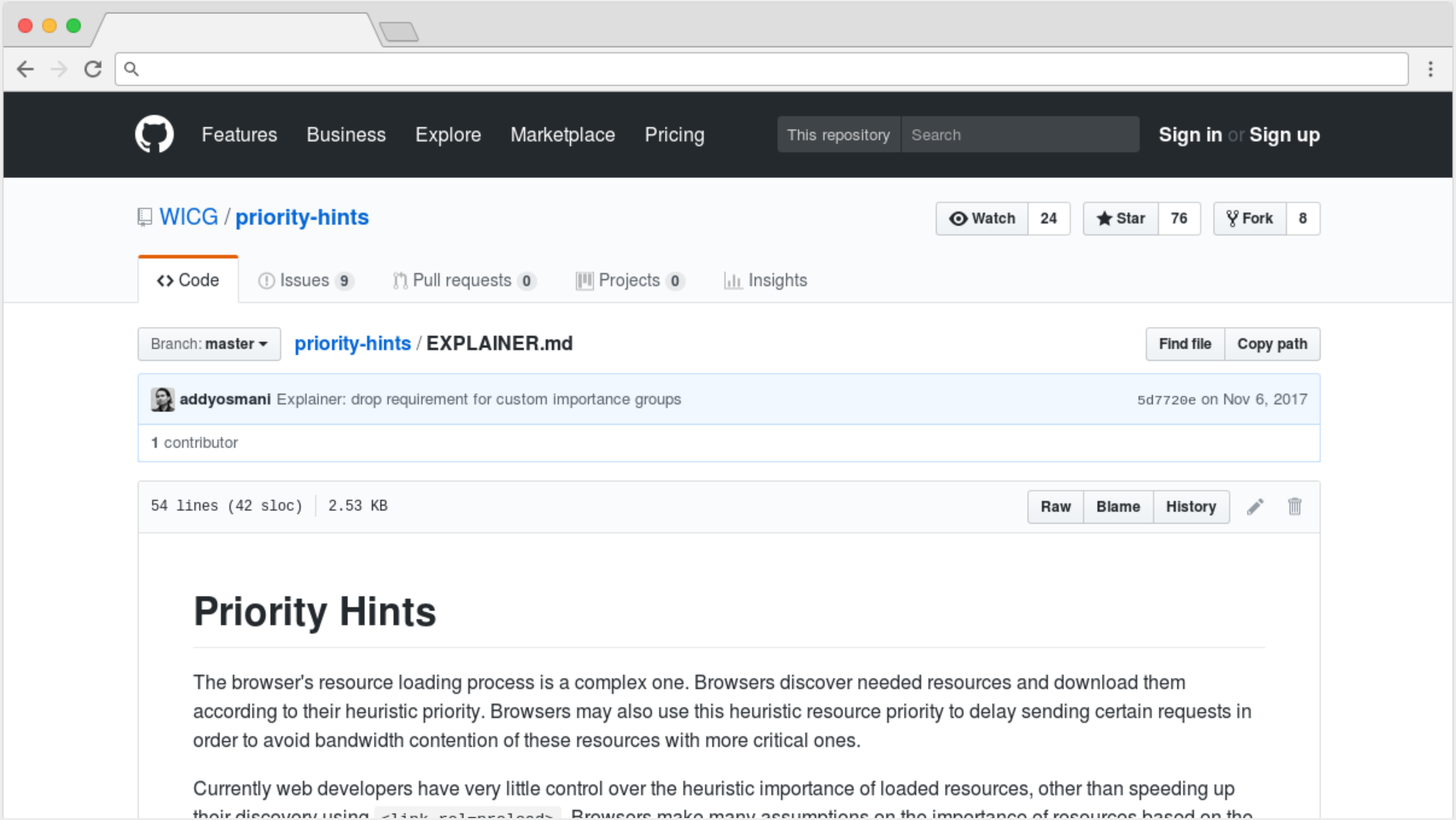
✅ Allows client to initiate fetches

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-I3-tJgGk



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.

@patrickhamann

Resource priorities 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

Danke

Patrick Hamann
speakerdeck.com/patrickhamann

patrick@fastly.com
@patrickhamann