

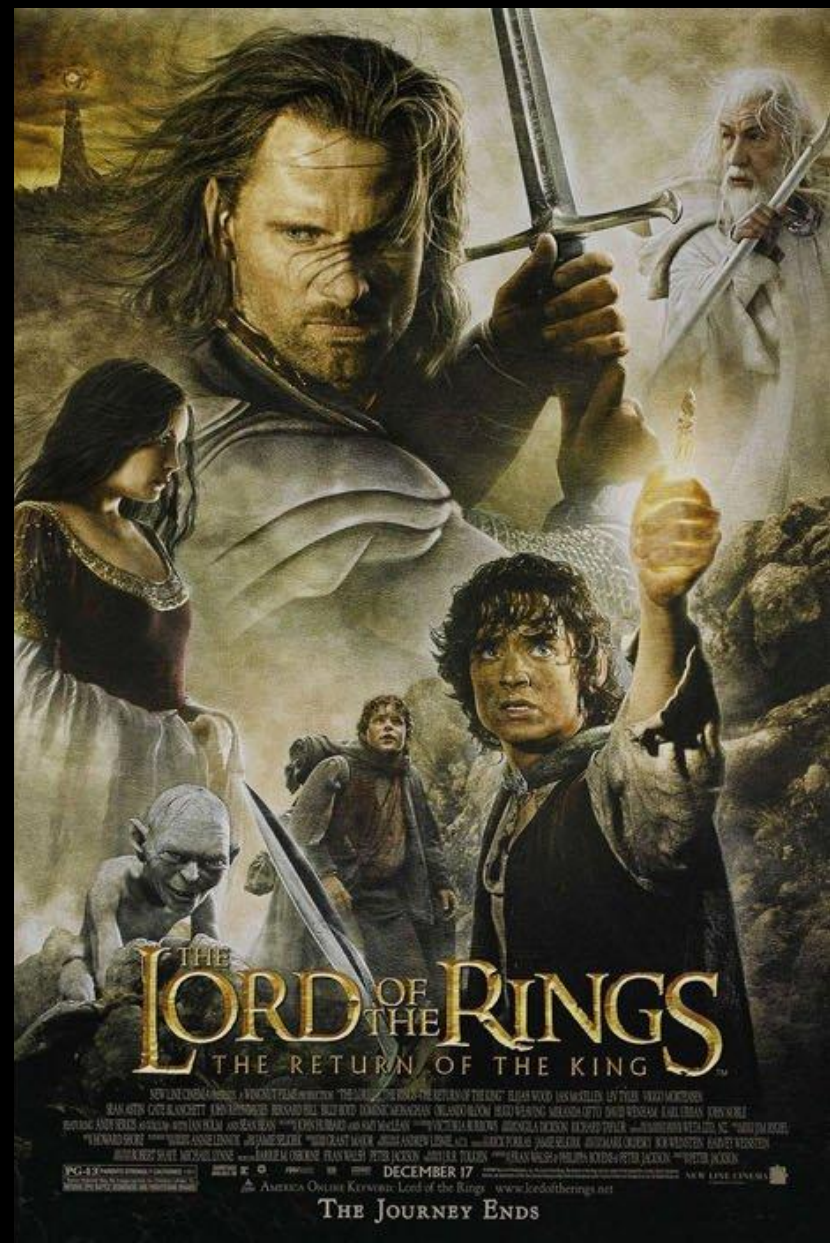
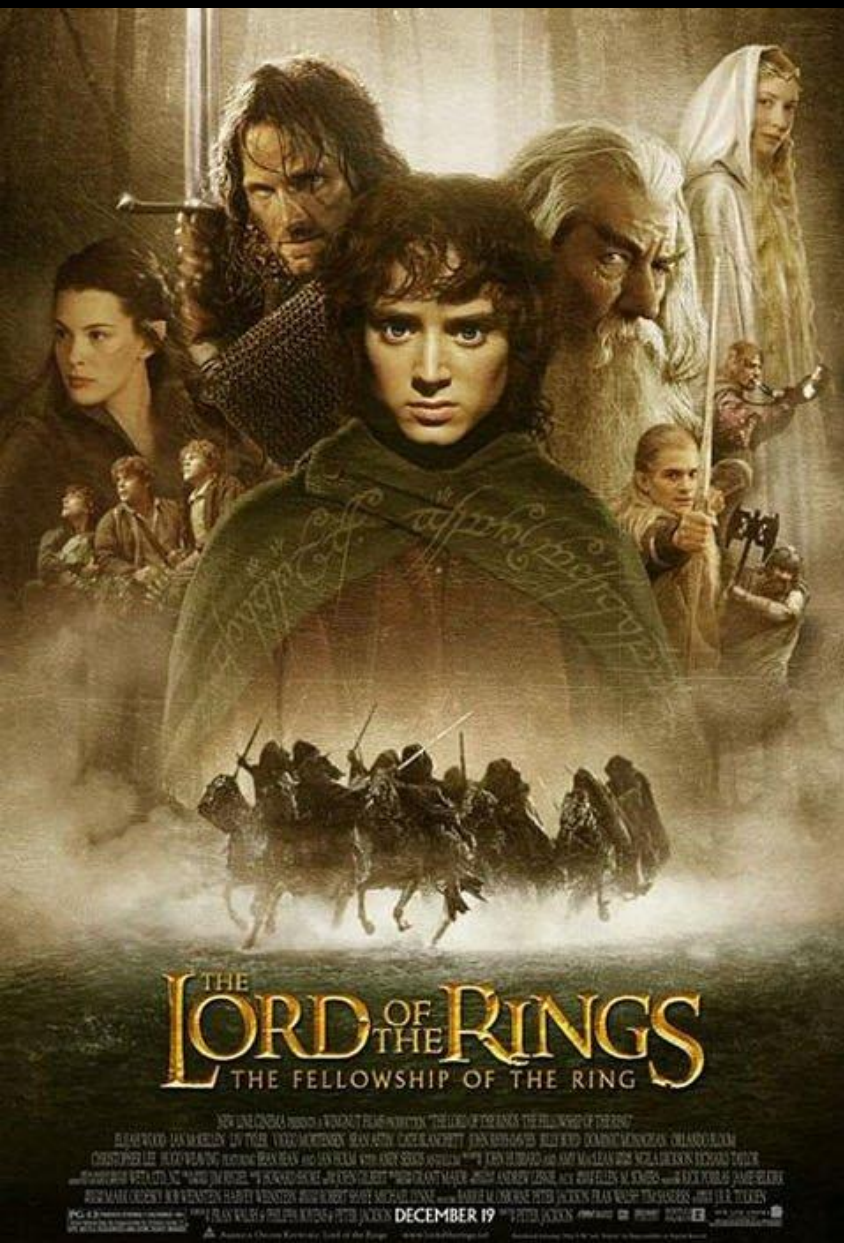
we speed

# Web Protocols For Frontend Developers

Robin Marx

@programmingart









Robin Marx

@programmingart



Today, after over 5 years of work, HTTP/3 was finally standardized as RFC 9114! [rfc-editor.org/rfc/rfc9114.ht...](https://rfc-editor.org/rfc/rfc9114.html)

Together with RFC 9204 (QPACK header compression) and RFC 9218 (Extensible Priorities) it ushers in an important new chapter for the Web!

Proud to have been part of this!



10:27 PM · Jun 6, 2022 · Twitter Web App



1,615 Retweets 156 Quote Tweets 4,903 Likes



Robin Marx

@programmingart



Today, after over 5 years of work, HTTP/3 was finally standardized as RFC 9114! [rfc-editor.org/rfc/rfc9114.ht...](https://rfc-editor.org/rfc/rfc9114.html)

Together with RFC 9204 (QPACK header compression) and RFC 9218 (Extensible Priorities) it ushers in an important new chapter for the Web!

Proud to have been part of this!



10:27 PM · Jun 6, 2022 · Twitter Web App

1,615 Retweets 156 Quote Tweets 4,903 Likes

## Lots of performance goodies:

- Connection Migration
- Head-of-Line blocking removal
- Better packet loss recovery
  - ACK ambiguity, SACKs, PTO/tail loss, fast handshake retransmit, ...
- **Better stream prioritization**
- **Faster handshake**
- Fine-grained flow control
- Congestion control flexibility
- ECN, RETRY, delayed ACKs, ...
- DATAGRAM and unreliable traffic

HTTP/3 Support

On



# How to tune your pages for HTTP/3?

Tune them for HTTP/2

## Fewer Domains

Consolidate on 1-3 connections in critical path

## Less Bundling

10 - 40 files should be fine (inlining CSS still ok)

## Help the Browser

Async / Defer JS  
Preload / Preconnect  
Fetch Priority  
Lazy loading

## No Server Push

Use 103 Early Hints instead

# How to tune your pages for HTTP/3?

Tune them for HTTP/2

## Fewer Domains

Consolidate on 1-3 connections in critical path

## Less Bundling

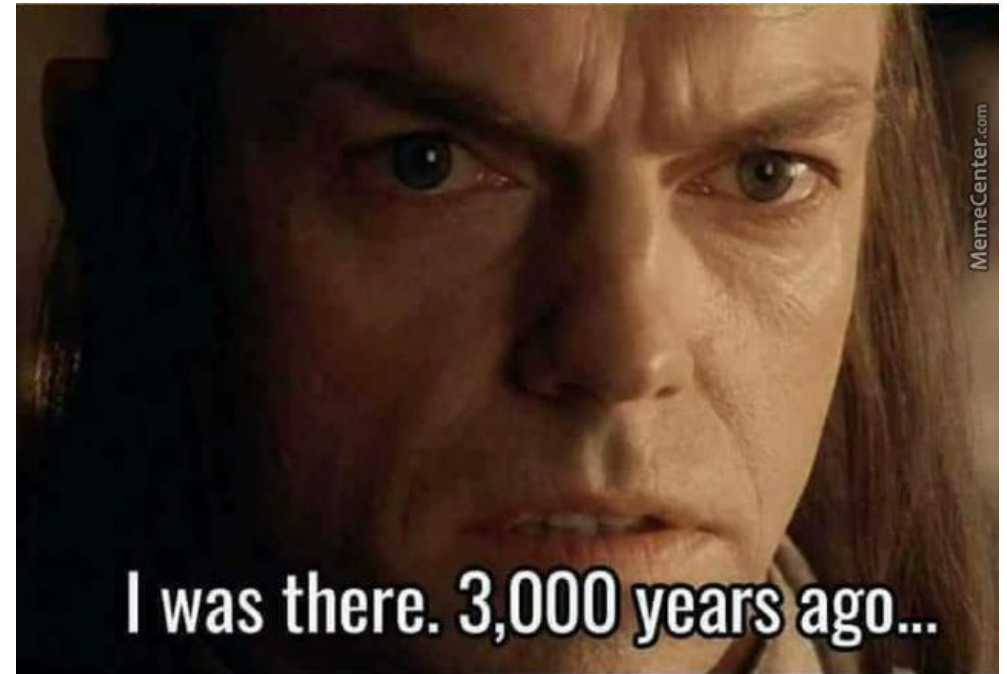
10 - 40 files should be fine (inlining CSS still ok)

## Help the Browser

Async / Defer JS  
Preload / Preconnect  
Fetch Priority  
Lazy loading

## No Server Push

Use 103 Early Hints instead



# The Black Box

# The Network Protocol



# Poking the Black Box



Server Push  
103 Early Hints



Resource Hints  
(preload, preconnect)



Fetch Priority



Lazy loading  
Async / Defer

# Poking the ~~Black-Box~~ Dangerous Artefact!

Server Push  
103 Early Hints



Resource Hints  
(preload, preconnect)



Fetch Priority



Lazy loading  
Async / Defer

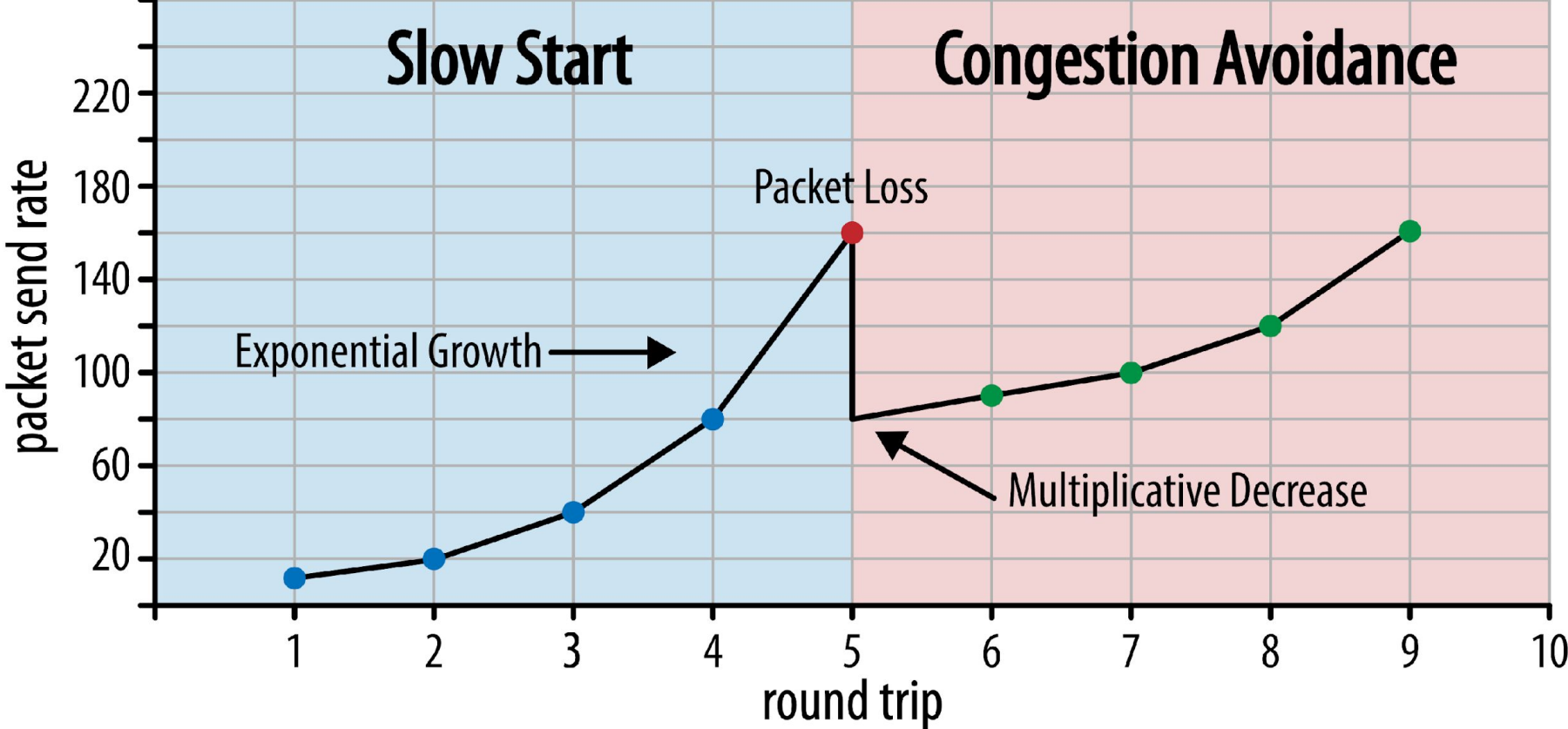
1. THE FELLOWSHIP OF THE PRIORITIES

2. THE TWO PRELOADS

3. THE RETURN OF SERVER PUSH

# CONCERNING CONGESTION CONTROL

# Congestion Control

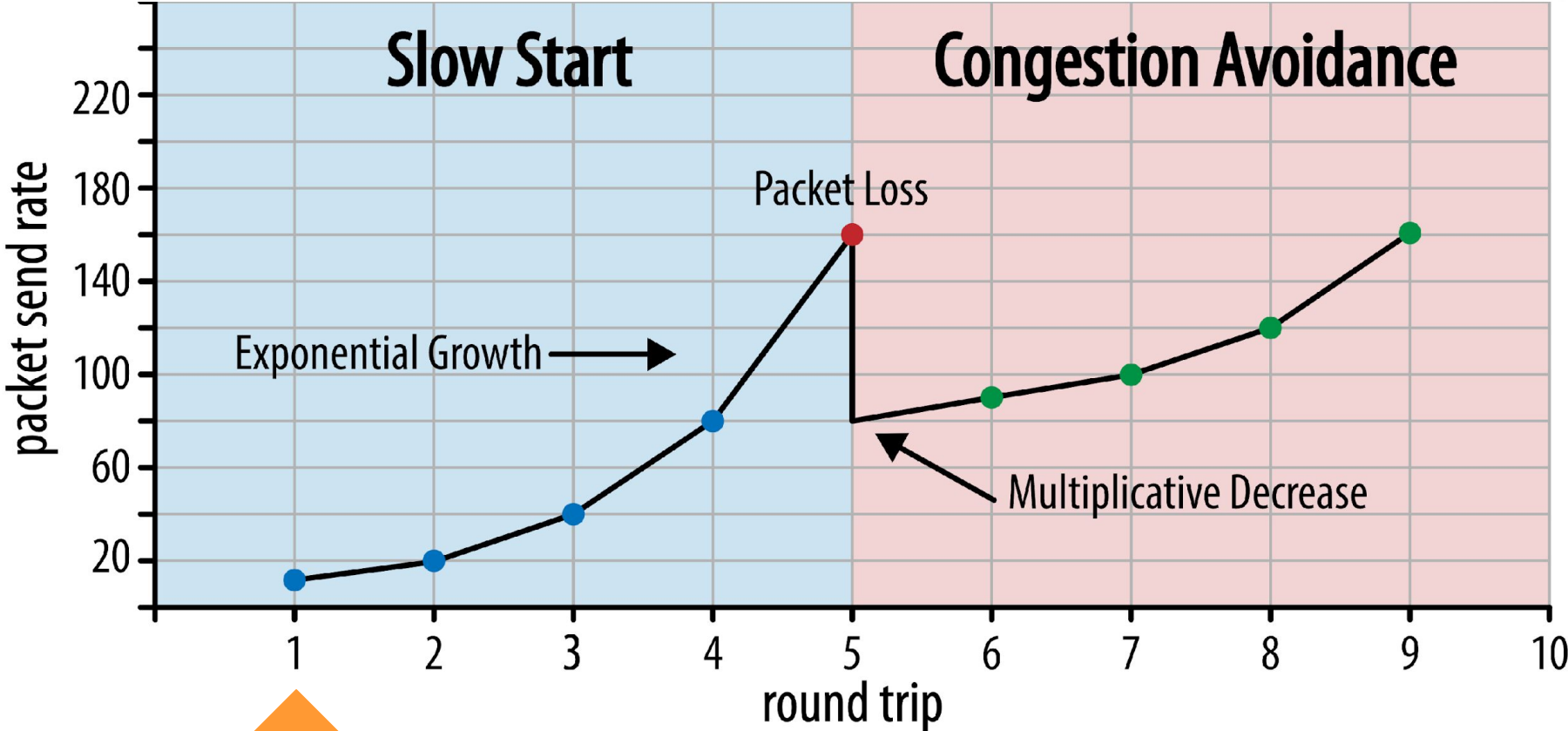


A meme featuring Aragorn from The Lord of the Rings. He is shown from the chest up, wearing his characteristic blue and brown attire. He has long, wavy brown hair and a goatee. He is looking slightly to the right with a smug, knowing expression, his right hand raised near his chin. The background is a blurred, warm-toned interior. Two lines of white, bold, sans-serif text are overlaid on the image: "ONE DOES NOT SIMPLY" at the top and "KNOW THE AVAILABLE BANDWIDTH" at the bottom.

**ONE DOES NOT SIMPLY**

**KNOW THE AVAILABLE BANDWIDTH**

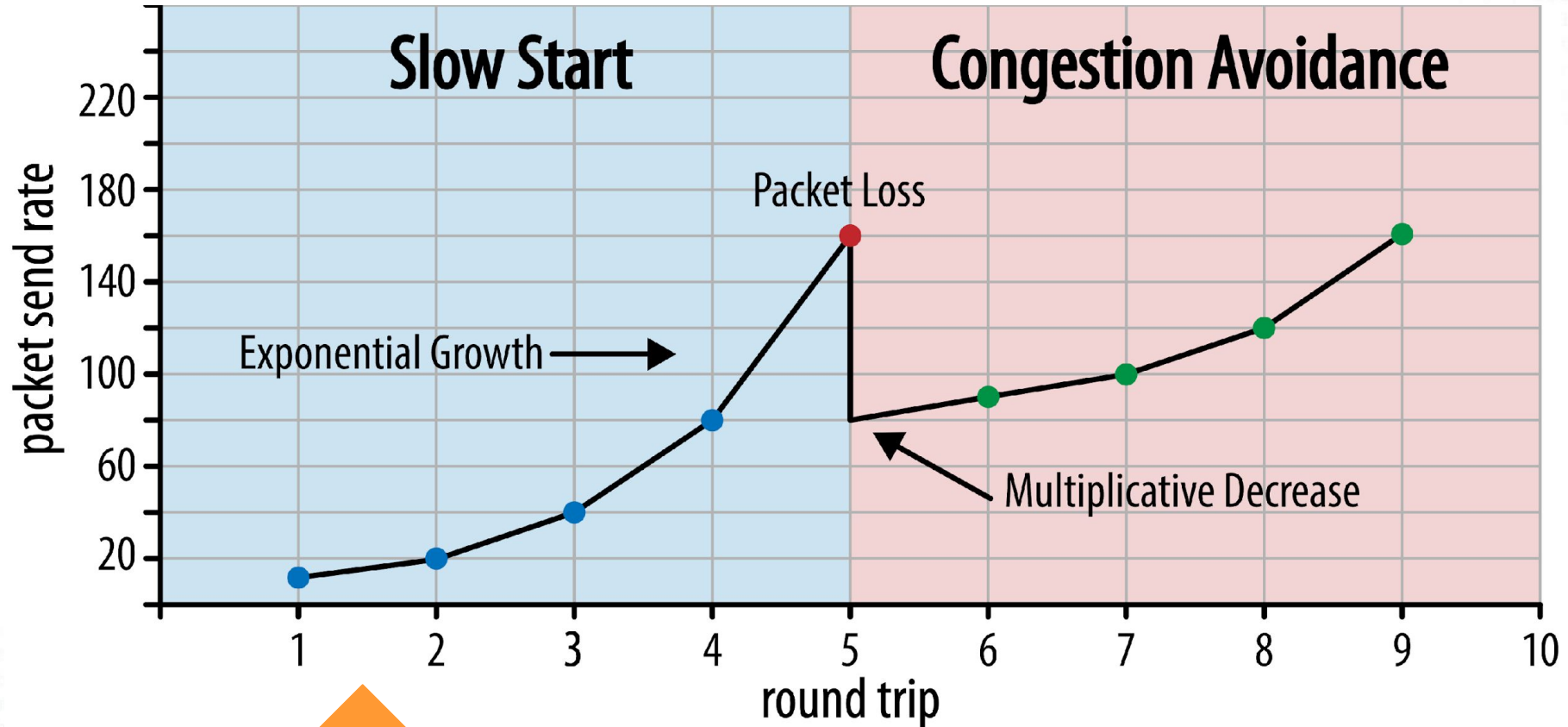
# Congestion Control



10 packets  
~14 KB

“Initial Congestion Window”

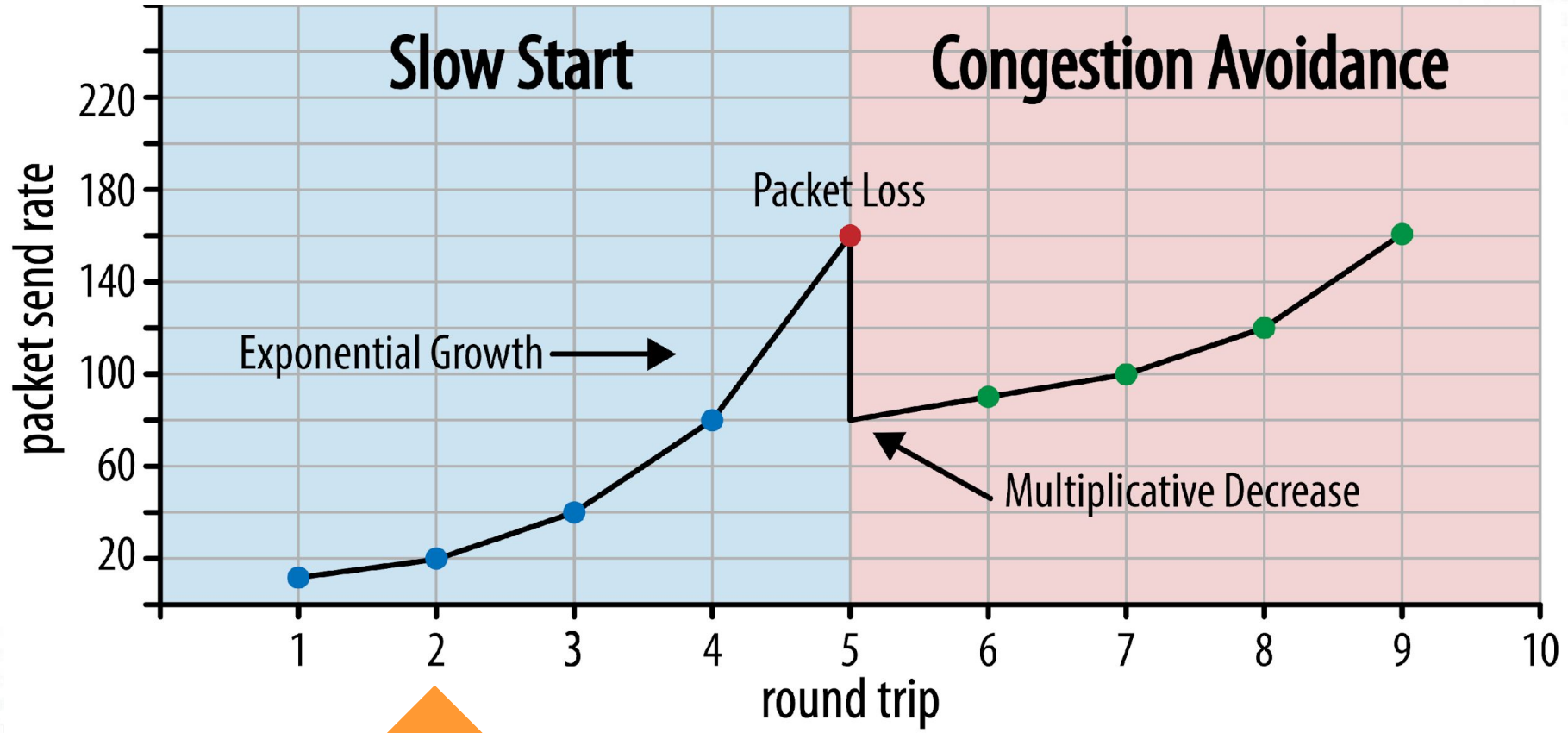
# Congestion Control



Wait full RTT for receiver to send back **acknowledgements (ACKs)**

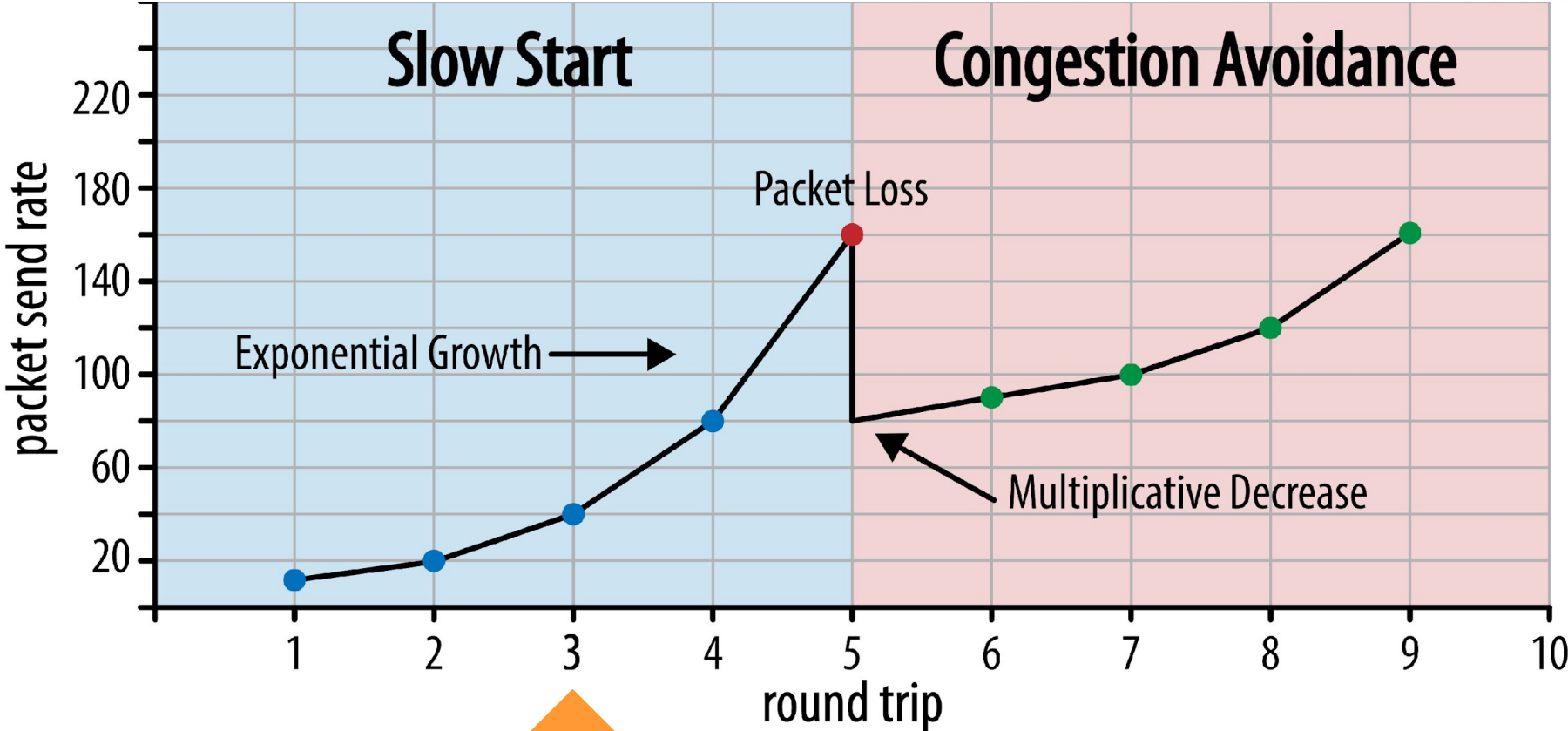


# Congestion Control



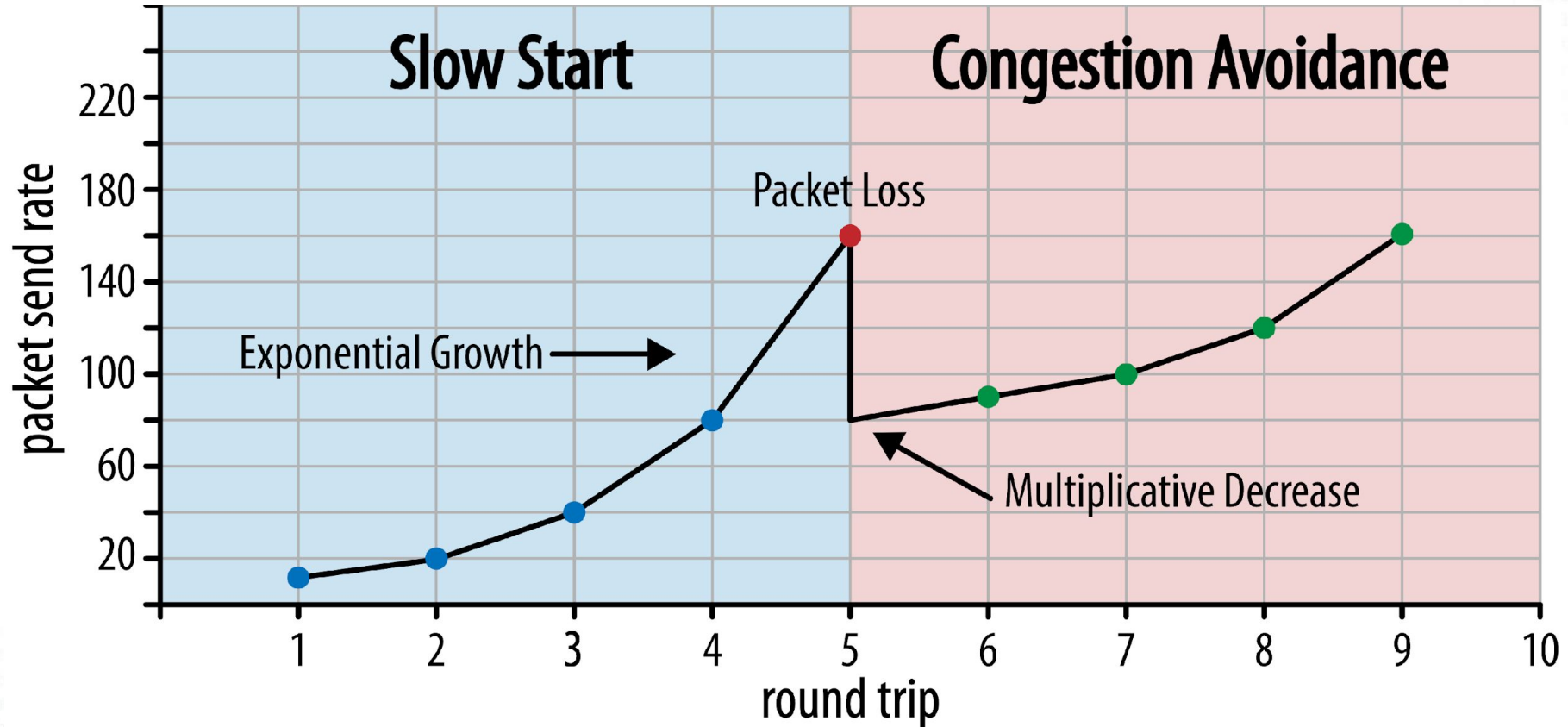
20 packets  
~28 KB

# Congestion Control



40 packets  
~56 KB

# Congestion Control



1 or more packets **didn't** get acknowledged

# You have no idea how deep the hobbit hole goes

**Cubic, BBR**

**Pacing, Chirping**

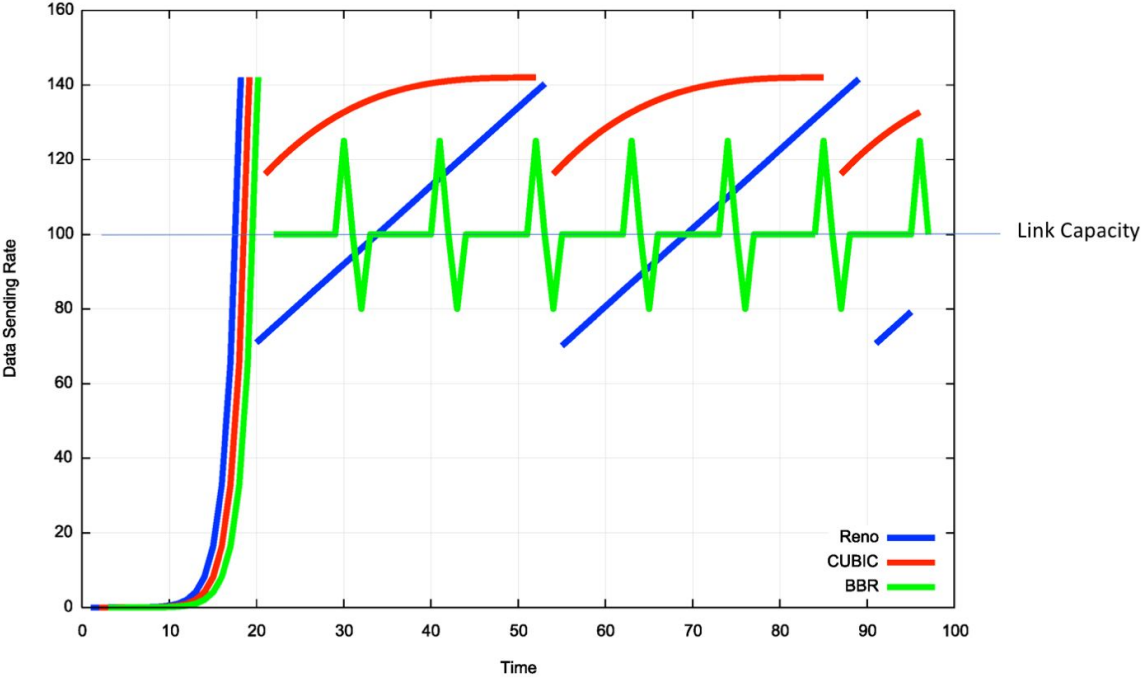
**HyStart++**

**Fairness**

**Bufferbloat, AQM**

**ECN, L4s**

# Slow Start is common



<b>Default</b>	10 packets	<b>14.6 KB</b>
<b>Most CDNs + hosting</b>	+ - 30 packets	44 KB
<b>Largest observed</b>	100 packets	146 KB

**1 TCP packet =  
+ - 1460 bytes**

<https://sirupsen.com/napkin/problem-15>  
<https://www.cdnplanet.com/blog/initcwnd-settings-major-cdn-providers>  
[https://developers.google.com/speed/pagespeed/service/tcp\\_initcwnd\\_paper.pdf](https://developers.google.com/speed/pagespeed/service/tcp_initcwnd_paper.pdf)  
<https://www.comsys.rwth-aachen.de/fileadmin/papers/2019/2019-rueth-iwtmsm.pdf>

# Persistent Myth

**WHY YOUR WEBSITE  
SHOULD BE UNDER 14KB  
IN SIZE**

25 August 2022

Also available to read on [dev.to](https://dev.to) (*warning it is much larger than 14kB*)

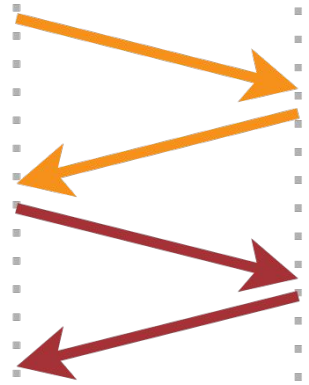
Having a smaller website makes it load faster — that's not surprising.

What is surprising is that a **14kB page can load much faster than a 15kB page** — maybe **612ms** faster — while the difference between a **15kB** and a **16kB** page is trivial.

<https://twitter.com/tunetheweb/status/1563130446841450497>  
<https://endtimes.dev/why-your-website-should-be-under-14kb-in-size>

# Data you can send in the first **HTTP** Round Trip

client                  server

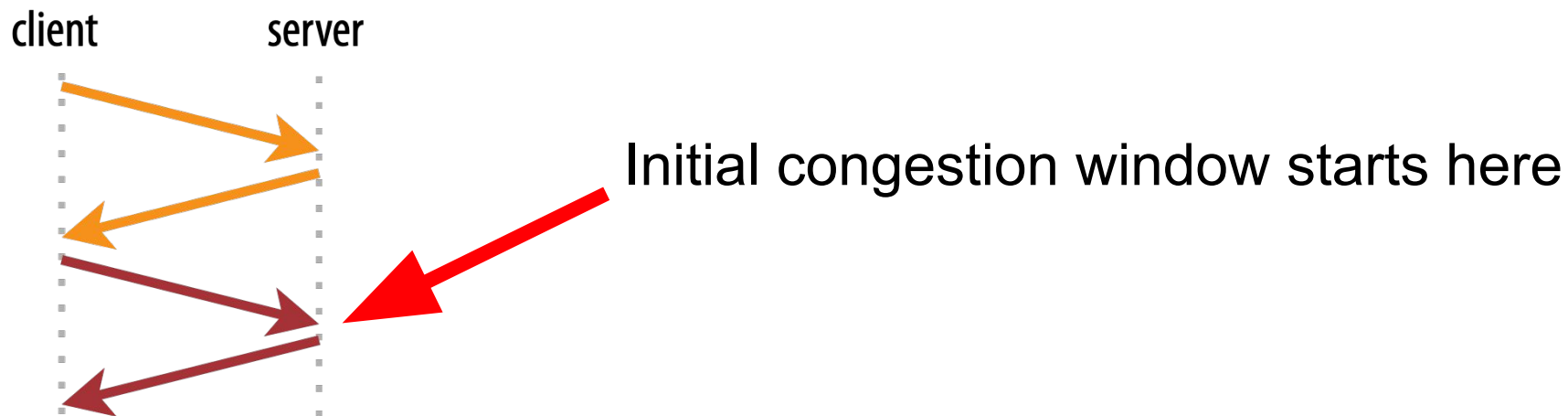


Unencrypted HTTP,  
without TLS

TCP  
HTTP/1

**14 KB**

# Data you can send in the first HTTP Round Trip



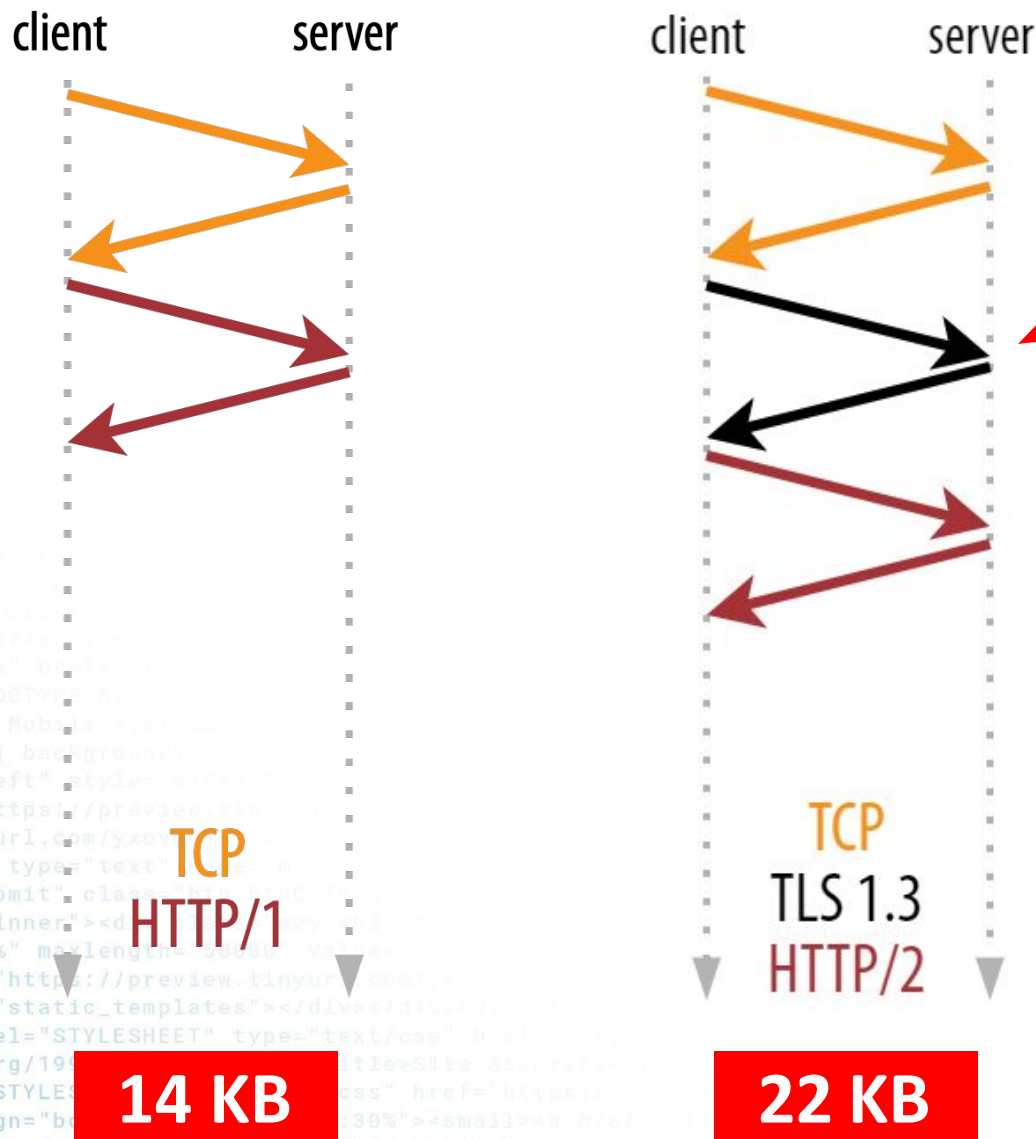
Unencrypted HTTP,  
without TLS

TCP  
HTTP/1

14 KB



# Data you can send in the first HTTP Round Trip

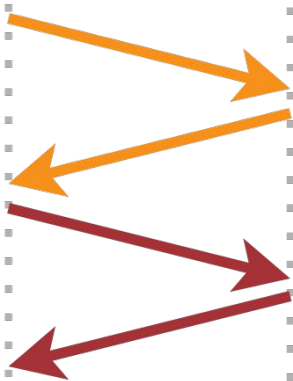


Initial congestion window starts here

TLS handshake uses *some* data, but not the full 14KB

# Data you can send in the first HTTP Round Trip

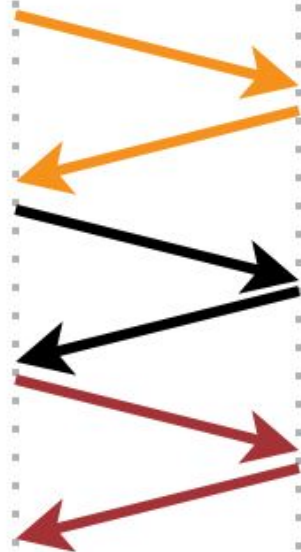
client server



TCP  
HTTP/1

14 KB

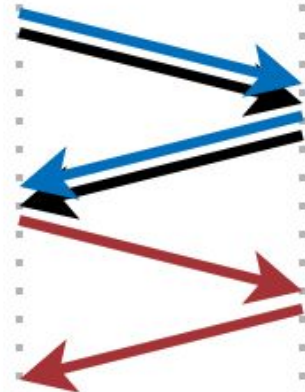
client server



TCP  
TLS 1.3  
HTTP/2

22 KB

client server

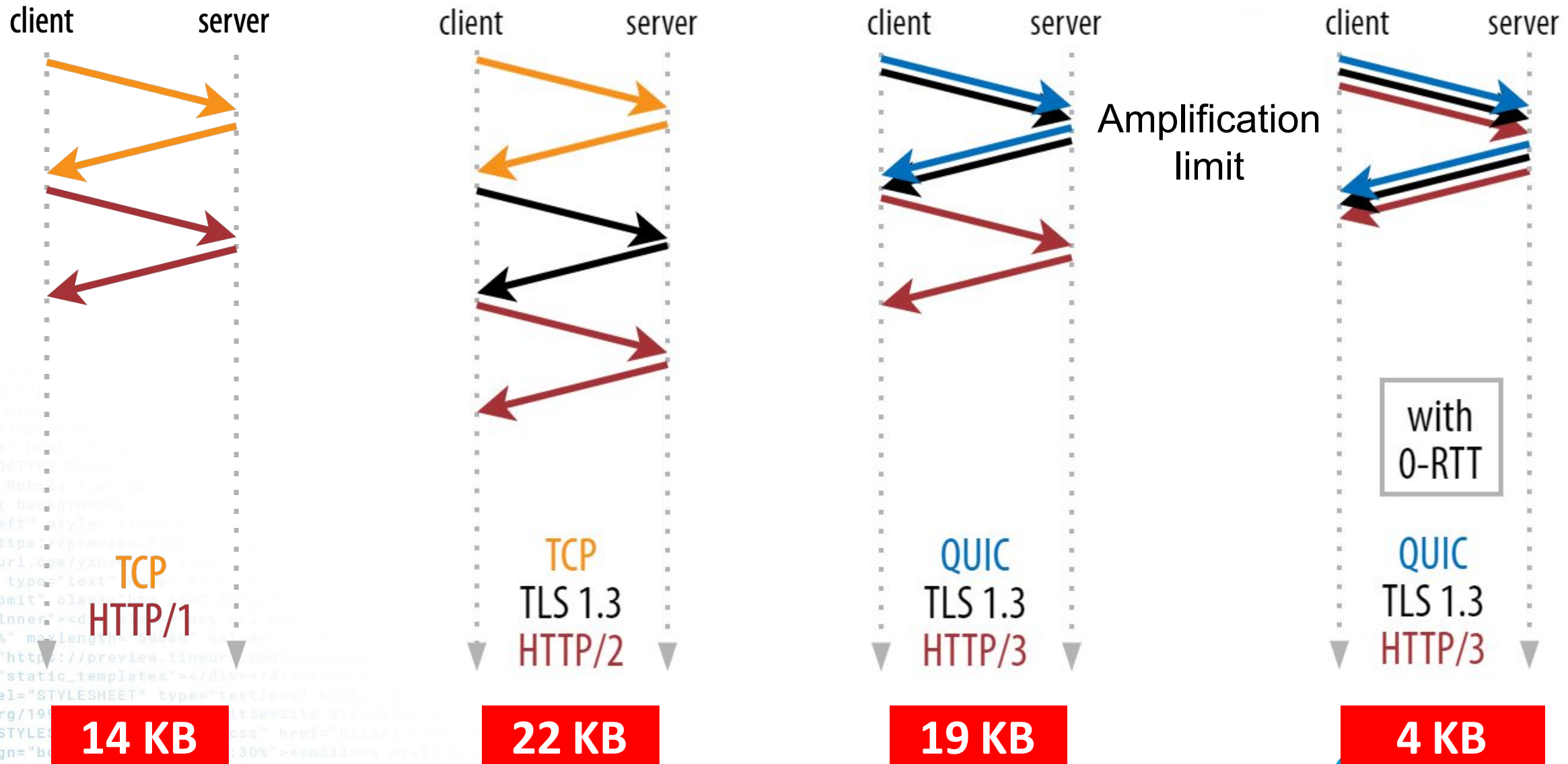


QUIC  
TLS 1.3  
HTTP/3

19 KB

Early QUIC packets  
are smaller:  
~1280 bytes

# Data you can send in the first HTTP Round Trip



# Key Takeaways

Loading everything in 1<sup>st</sup> RTT  
= **Utopia**

Loading critical resources in as few RTTs as possible  
= **Excellent idea!**



Minification, **compression**, dead code elimination, tree shaking, route splitting, image formats, font subsetting, ...







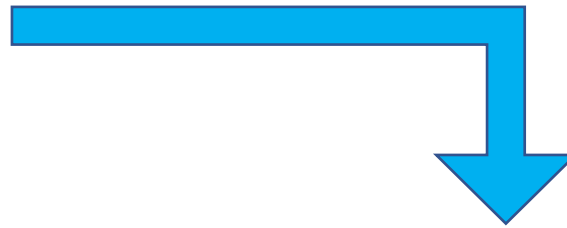


THE FELLOWSHIP  
OF THE PRIORITIES



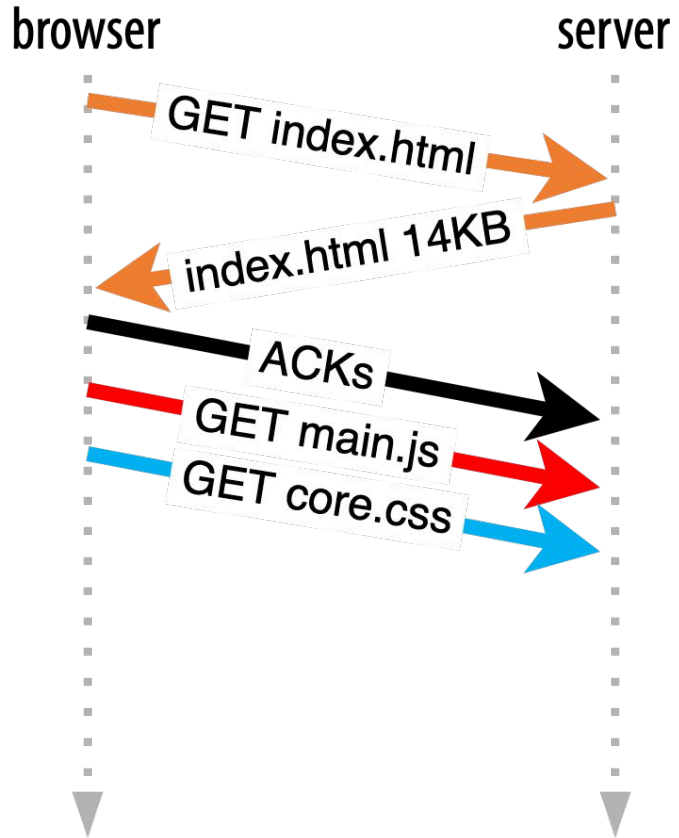
# Simple page we'll use

```
<head>  
  <script main.js>  
  <link core.css>  
</head>
```

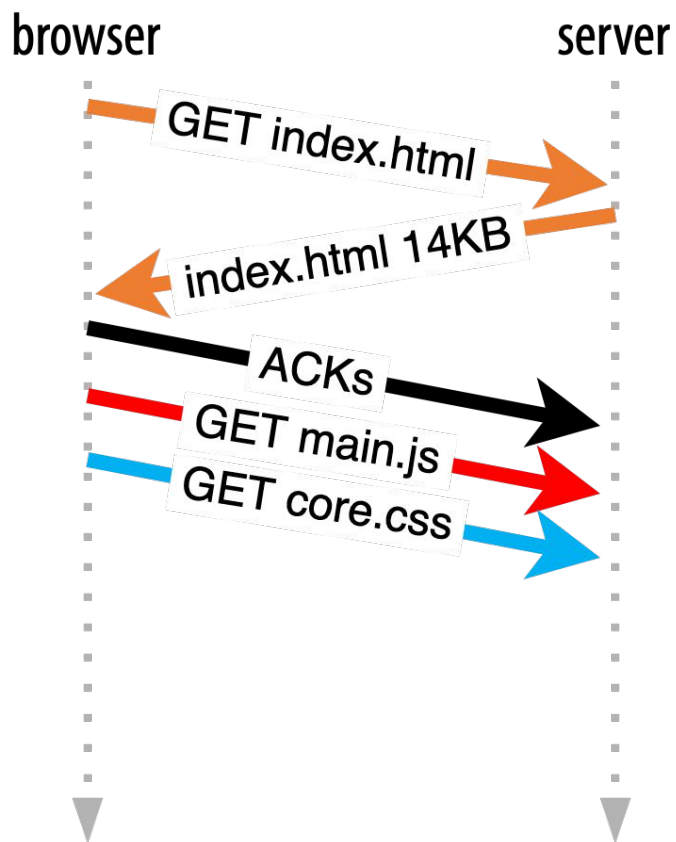


```
@font-face(font.woff2);
```

# First RTT is always just HTML



# What to send first?



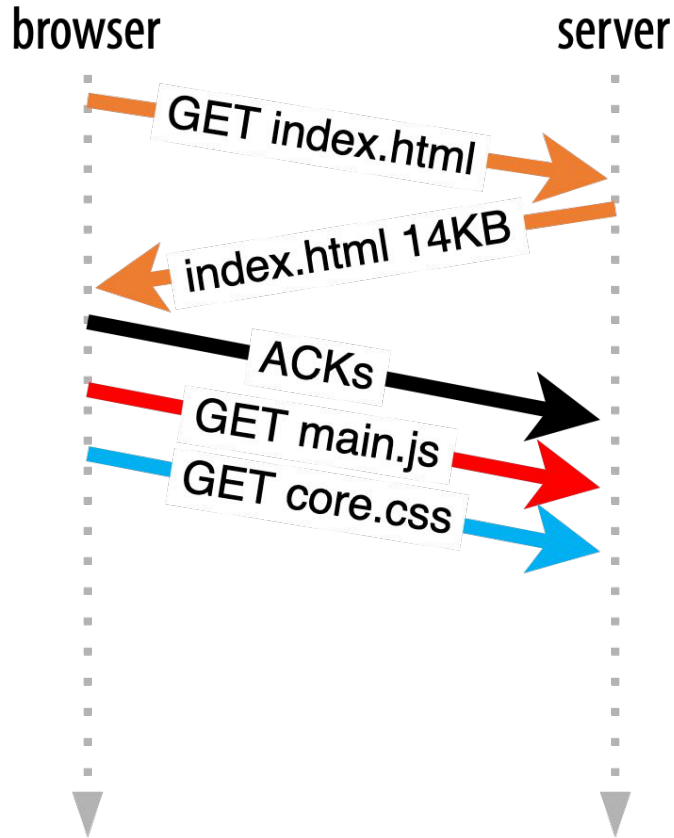
## How to fill 20 packets / 28 KB?

- 30KB: Remainder of **HTML**
- 200KB: **main.js**
- 60KB: **core.css**

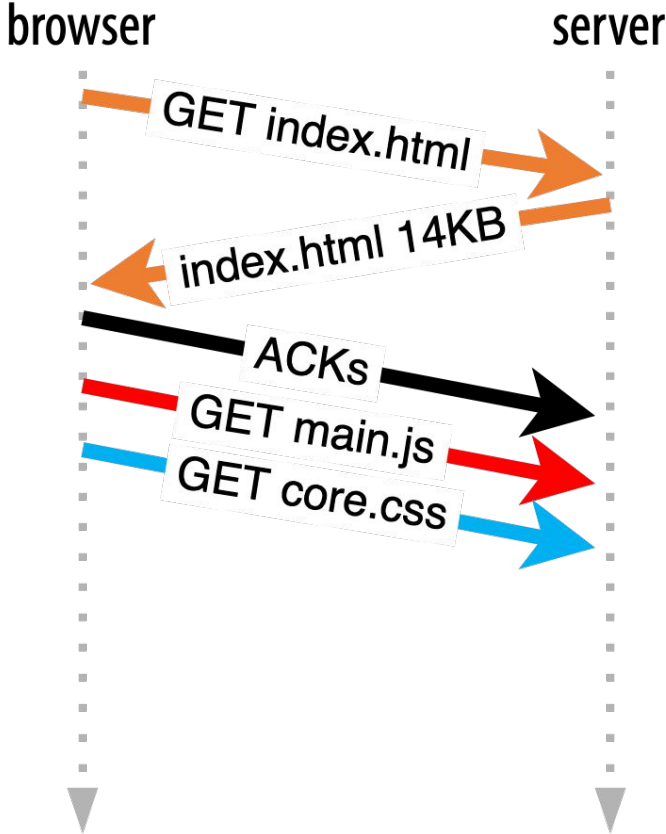
# What to send first?

## How to fill 20 packets / 28 KB?

- 30KB: Remainder of **HTML**
- 200KB: **main.js**
- 60KB: **core.css**

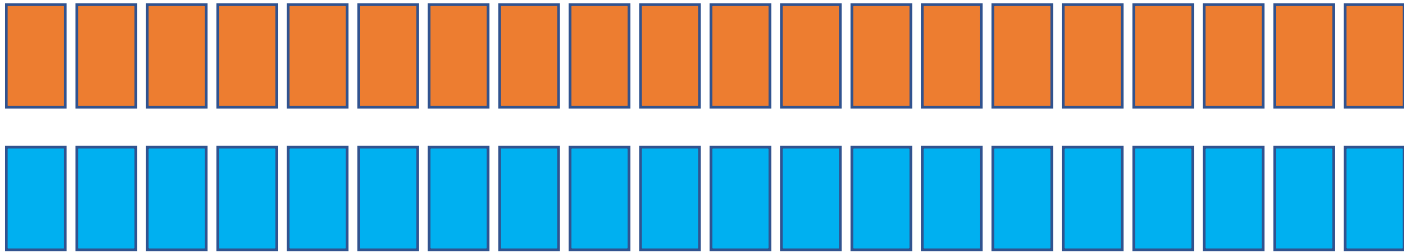


# What to send first?

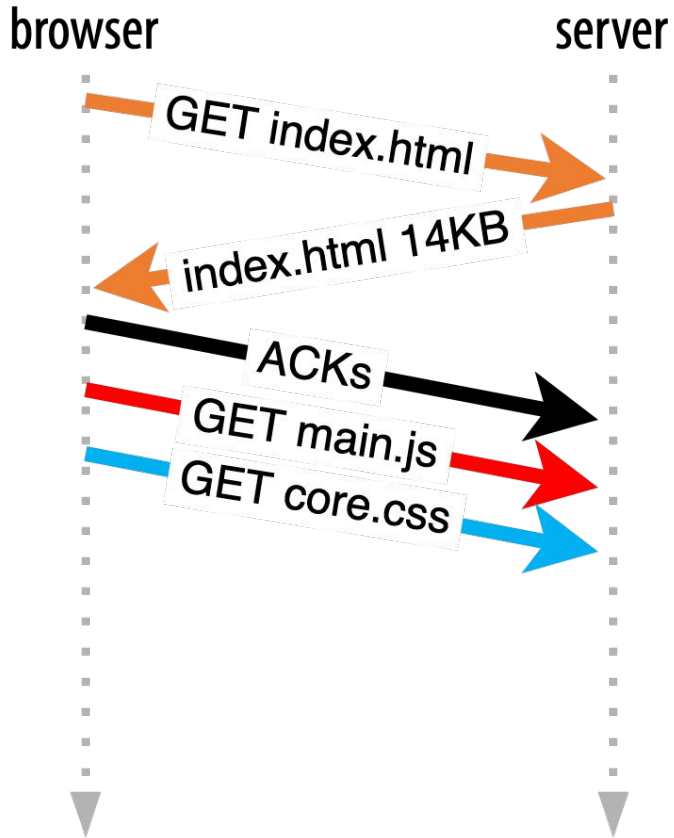


## How to fill 20 packets / 28 KB?

- 30KB: Remainder of **HTML**
- 200KB: **main.js**
- 60KB: **core.css**

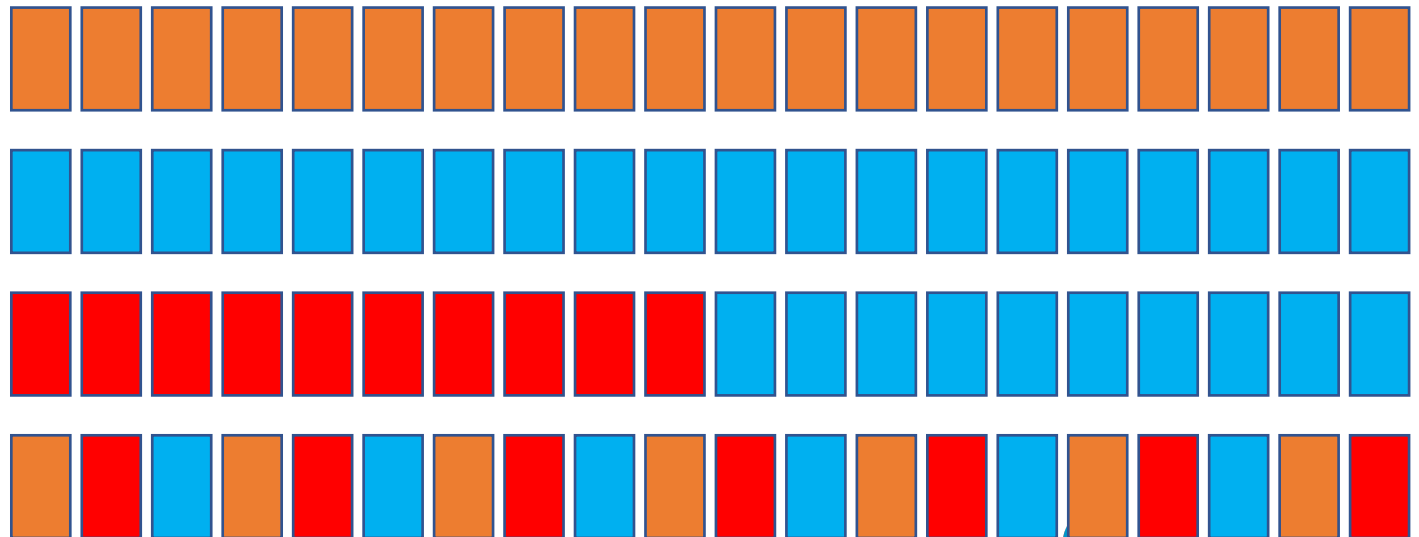


# What to send first?



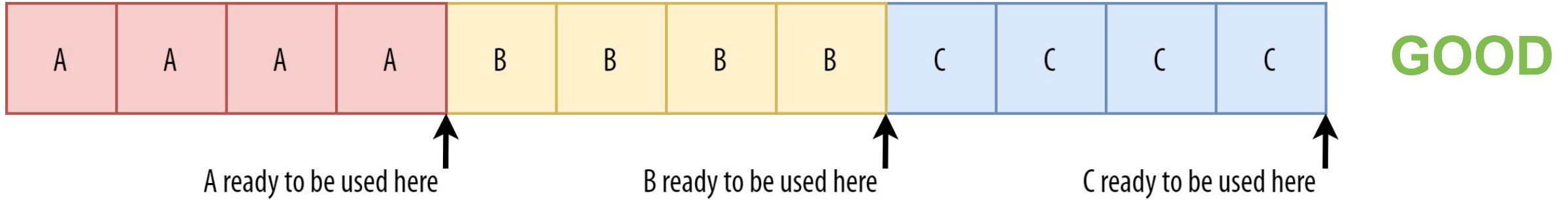
## How to fill 20 packets / 28 KB?

- 30KB: Remainder of **HTML**
- 200KB: **main.js**
- 60KB: **core.css**

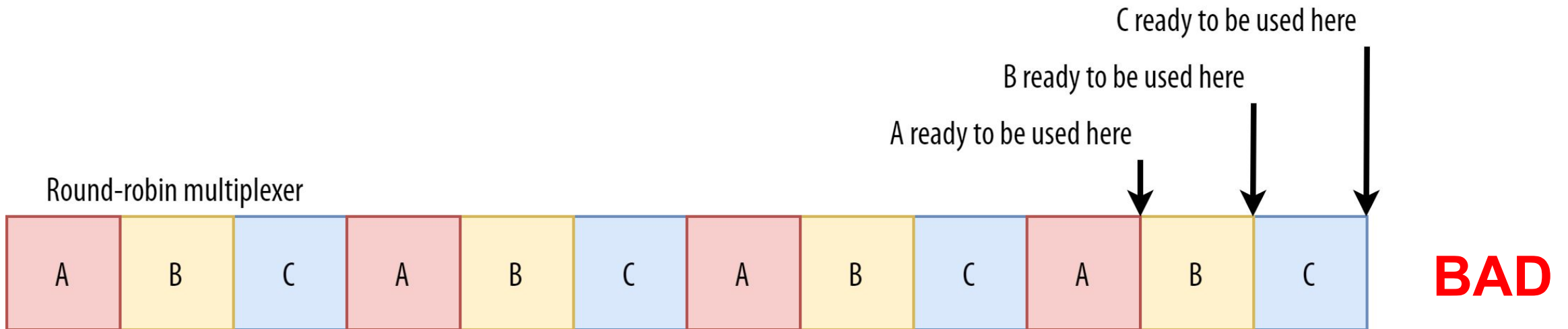


# JS and CSS need to be 100% loaded to be used

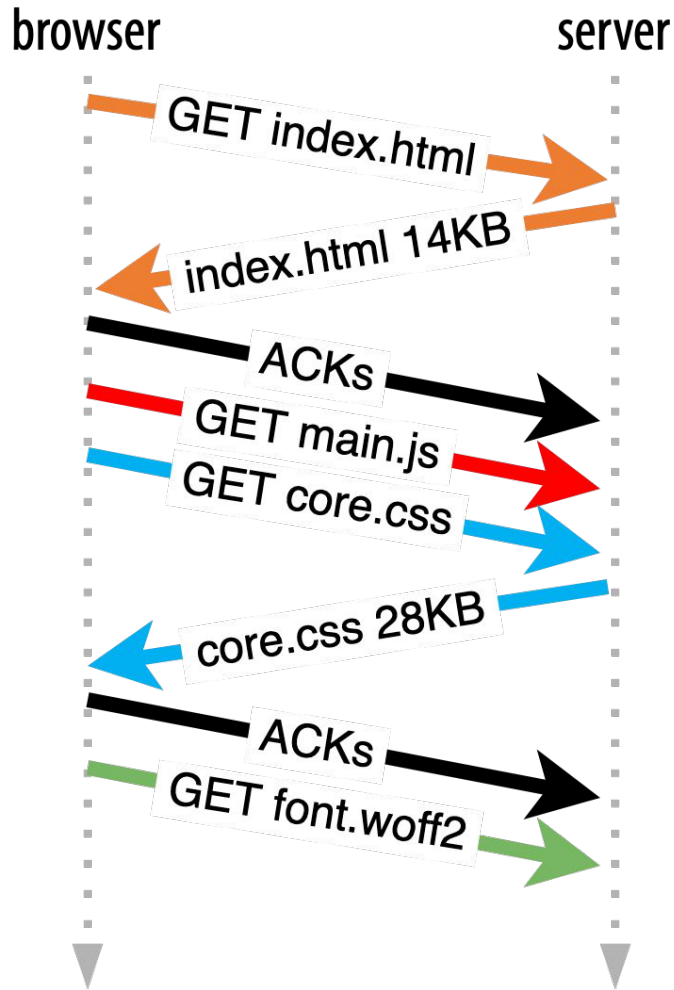
Sequential multiplexer



Round-robin multiplexer



# What to send first? Round two!

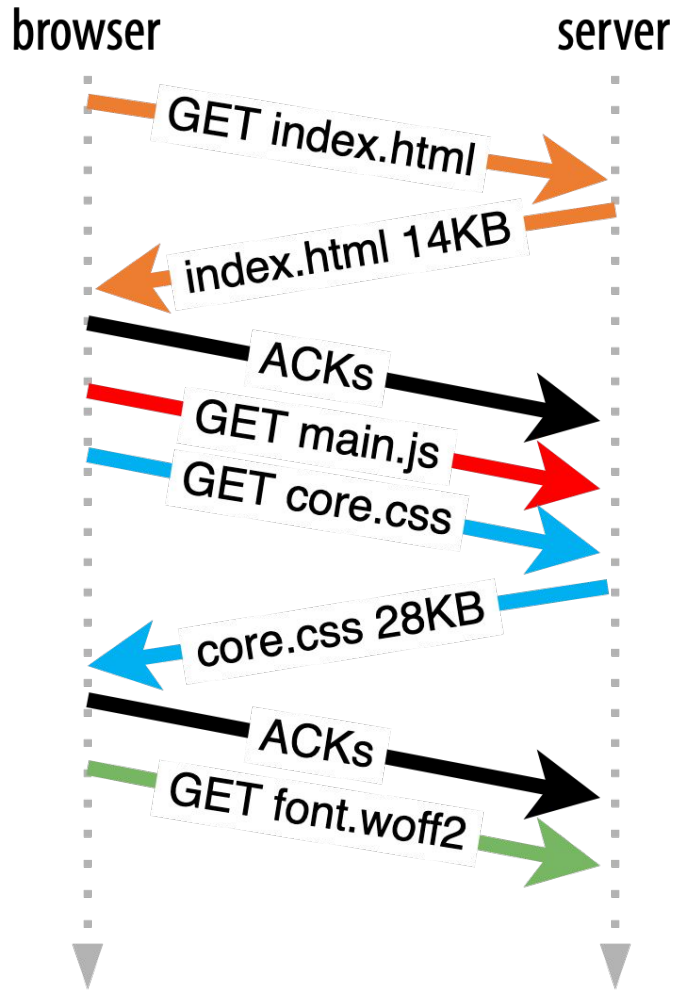


## How to fill 40 packets / 56 KB?

- 30KB: Remainder of **HTML**
- 200KB: **main.js**
- 32KB: **core.css**
- 50KB: **font.woff2**



# What to send first? Round two!

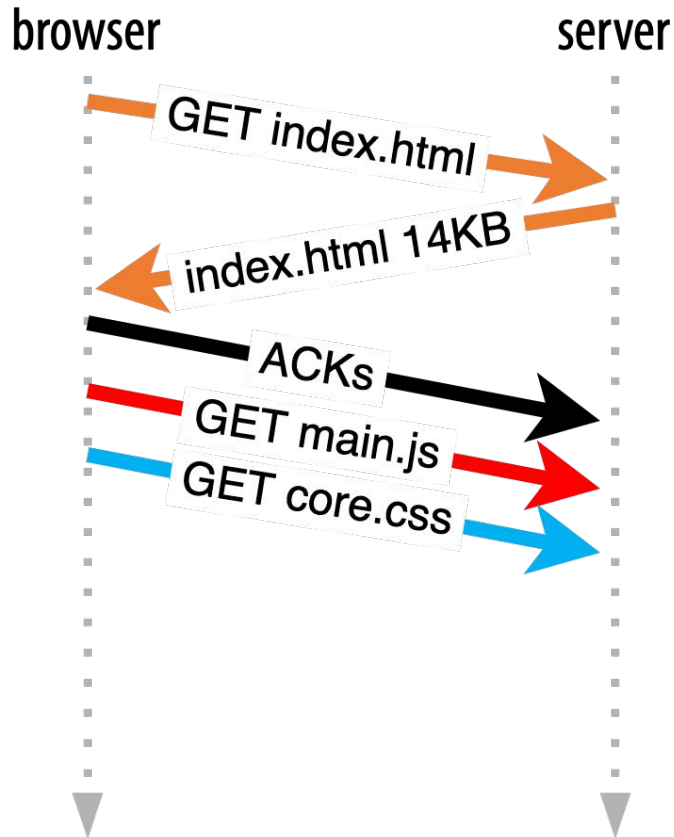


## How to fill 40 packets / 56 KB?

- 30KB: Remainder of **HTML**
- 200KB: **main.js**
- 32KB: **core.css**
- 50KB: **font.woff2**

Send order has a big impact on  
**Web Performance Metrics!**

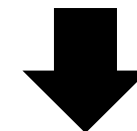
# Browsers send priorities with each request



- **HTML:** Most important
- **main.js:** Third most important
- **core.css:** Second most important

**How to fill 20 packets / 28 KB?**

Send most important resource first, *simple!*



# (HTTP/2) Servers often don't listen to browsers...

## Browser instructions:



Apache



nginx



NodeJS



# (HTTP/2) Servers often don't listen to browsers...

## Browser instructions:



Apache



nginx



NodeJS



**Worst case for web performance**







# Browsers decide resource priorities



## 2 Problems:

1. They don't agree on what is most important
2. It often works differently than what you'd expect

# Browsers don't agree

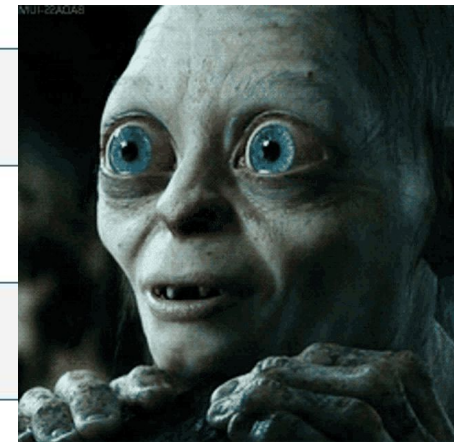
↓ Type / Priority →	Highest	High	Medium	Low	Lowest
Main resource (HTML)					
Font (@font-face)					
CSS (head)					

# Browsers don't agree

↓ Type / Priority →	Highest	High	Medium	Low	Lowest
Main resource (HTML)					
Font (@font-face)					
CSS (head)					
JS (head)					
JS (async)					
JS (defer)					
JS (body)					
JS (bottom)					

# Browsers don't agree

↓ Type / Priority →	Highest	High	Medium	Low	Lowest
Main resource (HTML)					
Font (@font-face)					
CSS (head)					
JS (head)					
JS (async)					
JS (defer)					
JS (body)					
JS (bottom)					





# Guess what this does!




```
<link rel="preload" href="lcp-image.jpg" as="image">
```

(How) does this change the image priority?

1. Higher priority / more important
2. No change / default importance
3. Lower priority / less important







# Guess what this does!

```
<link rel="preload" href="lcp-image.jpg" as="image">
```

↓ Type / Priority →	Highest	High	Medium	Low	Lowest
Image (body)				 	

# Guess what this does!

```
<link rel="preload" href="lcp-image.jpg" as="image">
```

↓ Type / Priority →	Highest	High	Medium	Low	Lowest
Image (body)				 	
Image (preload)				  	



# Guess what this does!

```
<link rel="preload" href="lcp-image.jpg" as="image">
```

↓ Type / Priority →	Highest	High	Medium	Low	Lowest
Image (body)				 	
Image (preload)				  	
Image (preload + fetchpriority)				 	

# Here Be Dragons!



```
<link rel="preload" href="lcp-image.jpg" as="image" fetchpriority="high">
```



Preload by itself doesn't bump priority → only sends request earlier

⇒ **Combine with fetchpriority!**

# Fetch Priority


```

```

Tell browser which image will contribute to  
**Largest Contentful Paint**

# Fetch Priority

Deprioritize (initially) invisible carousel images,  
but make the visible one load faster!



```
<ul class="carousel">
  
  
  
  
</ul>
```

# Coming to a Browser near you!

The screenshot shows the Safari Technology Preview browser interface. The 'Develop' menu is open, and 'Fetch Priority' is checked under 'Experimental Features'. A red arrow points to the 'Fetch Priority' option. Another red arrow points to the 'Priority' column in the table below.

Name	Domain	Type	Priority
zrBPJq27O4Hs8haszVnK.svg	web-dev.imgix.n...	svg	High
9WSNd3mdbXACF19ELKJ1.png	web-dev.imgix.n...	webp	High
CZo4R87iOBYiRplq6NcP.jpg	web-dev.imgix.n...	webp	Low
data:image/svg+xml,%3C...%3E	—	svg	—
data:image/svg+xml,%3C...%0A	—	svg	—
ga-audiences	www.google.com	gif	Low

[https://bugzilla.mozilla.org/show\\_bug.cgi?id=1797715](https://bugzilla.mozilla.org/show_bug.cgi?id=1797715)

<https://twitter.com/tunetheweb/status/1653879920026693633>

<https://github.com/WebKit/WebKit/commit/69a182bca140bcc4d604dd6eba4ea48a16e9d00a>



# Watch out! Browsers also Artificially Delay Resources



Name	Priority	Time	1000.0ms	2.00s	3.00s
test_all.html	High	288ms			
font1.woff2	Medium	282ms			
font2.woff2	Medium	344ms			
script.js	High	344ms			
style.css	High	344ms			
img1.png	Low	543ms			
img1.png	Low	674ms			
script.js	Low	719ms			
style.css	High	341ms			
fonts.css	High	344ms			
style.css	Low	716ms			
script.js	High	344ms			
script.js	Medium	343ms			
script.js	High	349ms			
script.js	Medium	349ms			
script.js	High	349ms			
script-fetch.js	High	349ms			
style.css	High	348ms			
img1.png	Medium	1.05s			
img2.svg	Medium	1.10s			
script.js	High	348ms			
img1.png	Low	2.87s			
style.css	High	330ms			
script.js	High	330ms			

In **tight mode**, low priority resources are only loaded if there are less than 2 in-flight requests

**tight mode = until all blocking scripts are executed**

# Key Takeaway

Resource Loading Behaviour  
is **inconsistent and unpredictable** across

Browsers  
Servers

Network/loading conditions

*Just looking at Chrome / CWV is **not enough***

# THE TWO PRELOADS

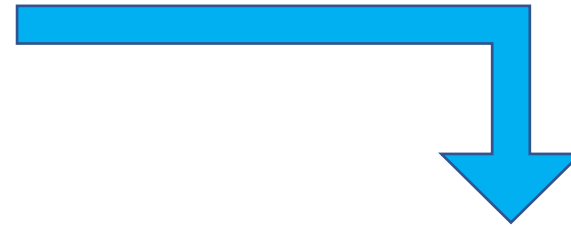
# Preload is weird enough on its own

<head>

<preload image.jpg>

<script bundle.js>

</head>



```
img.setAttribute("src", image.jpg);
```

# Without preload



Image request is deferred until after render-blocking requests

# With preload

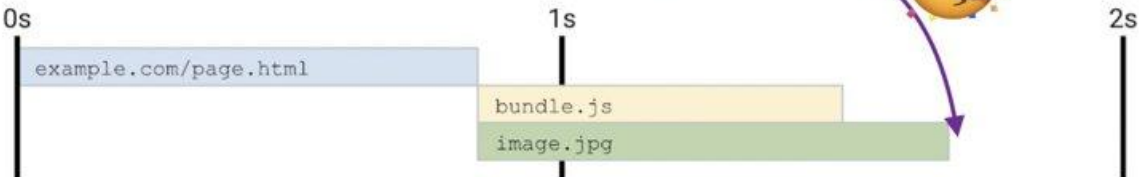


Image request is initiated alongside render-blocking requests

## Without preload



Image request is deferred until after render-blocking requests

## ~~With preload~~

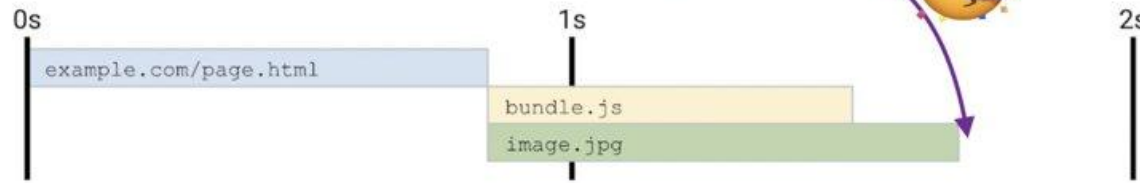


Image request is initiated alongside render-blocking requests

**40 packets**  
**~56 KB**

**Where will you fit  
the image data?**

## Without preload



Image request is deferred until after render-blocking requests

## ~~With preload~~

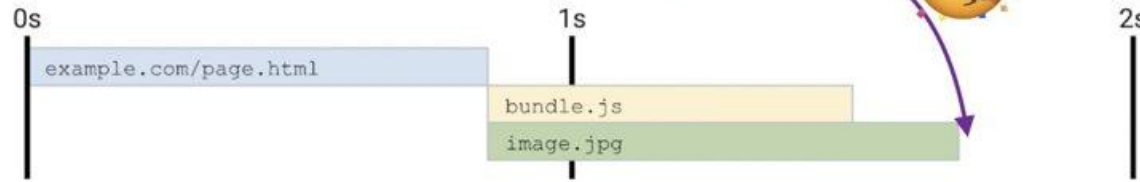
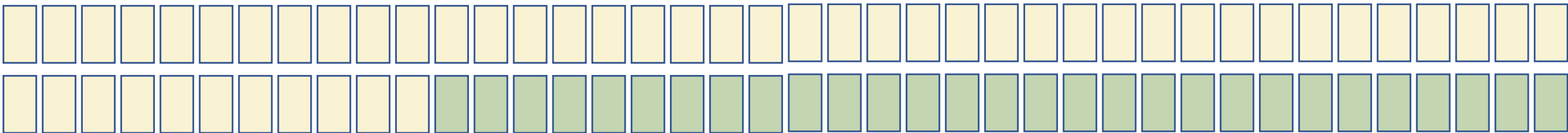


Image request is initiated alongside render-blocking requests

**40 packets**  
**~56 KB**

**Where will you fit  
the image data?**



## Without preload



Image request is deferred until after render-blocking requests

## ~~With preload~~

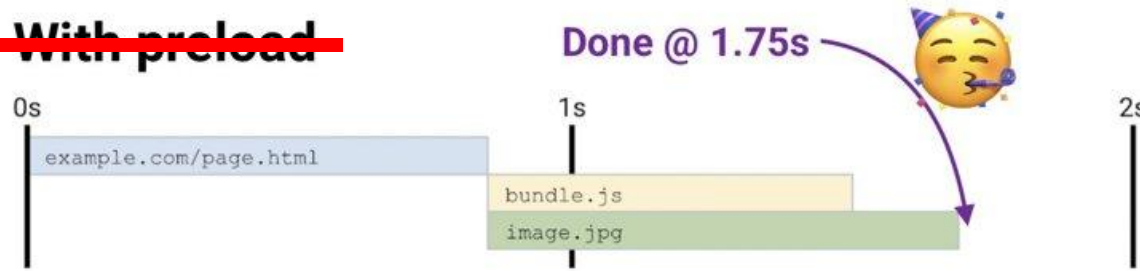
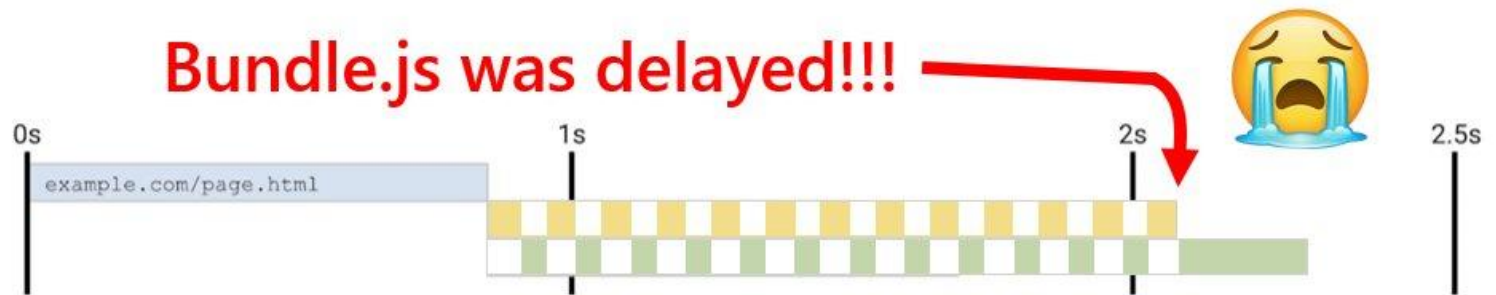


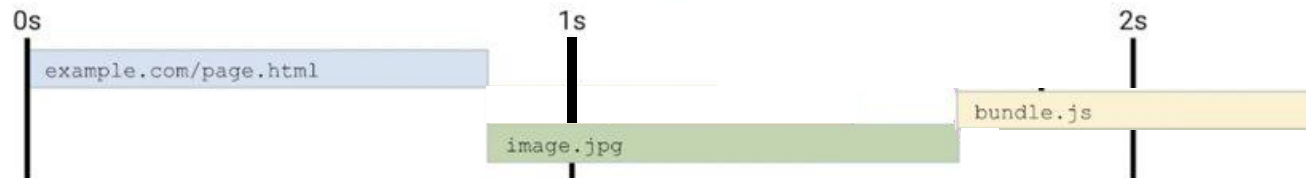
Image request is initiated alongside render-blocking requests

**Bundle.js was delayed!!!**



Bad prioritization

**TERRIBLE**  
prioritization





**GOOD**  
prioritization

~~Without preload~~

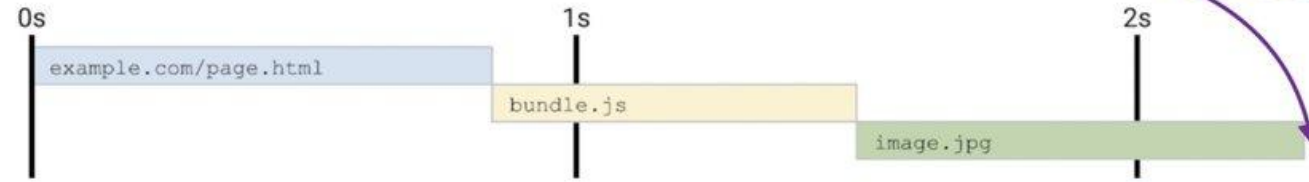


Image request is deferred until after render-blocking requests

~~With preload~~

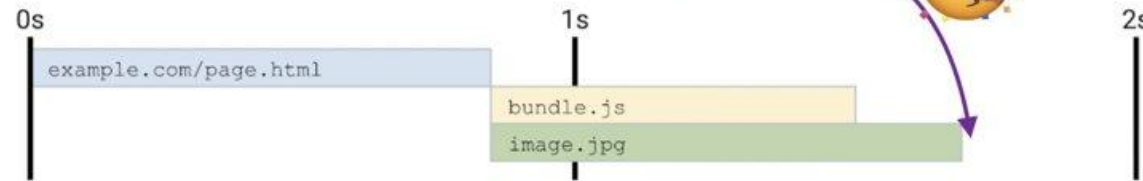
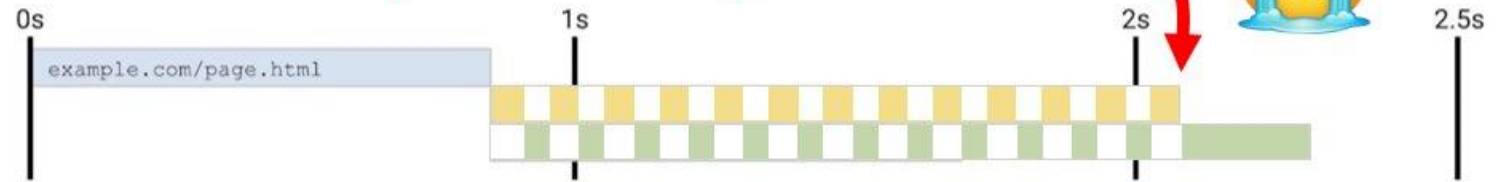


Image request is initiated alongside render-blocking requests

Bad  
prioritization

**Bundle.js was delayed!!!**



**TERRIBLE**  
prioritization



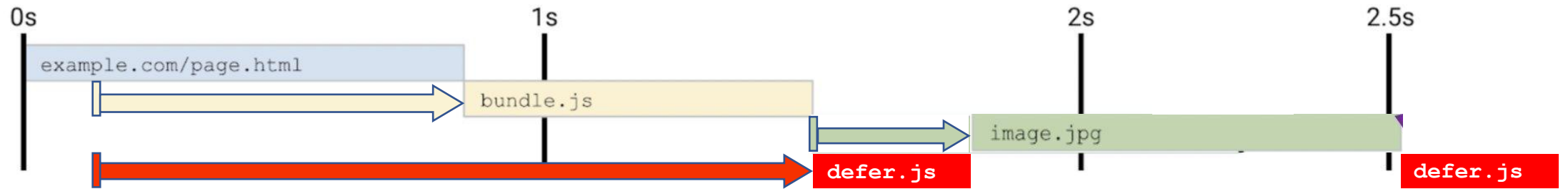
# Preload only helps in certain situations

## Without preload



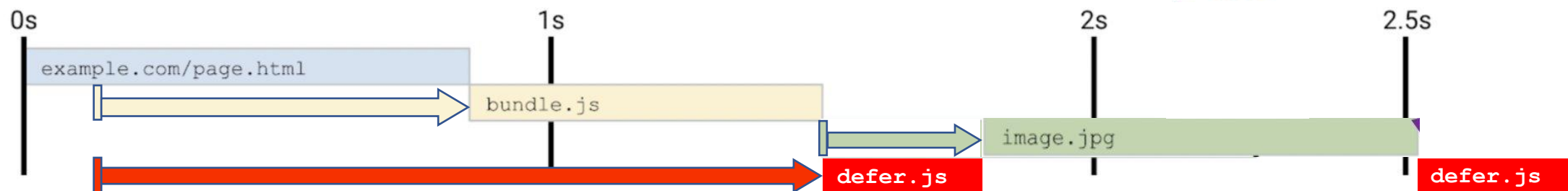
# Preload only helps in certain situations

## Without preload

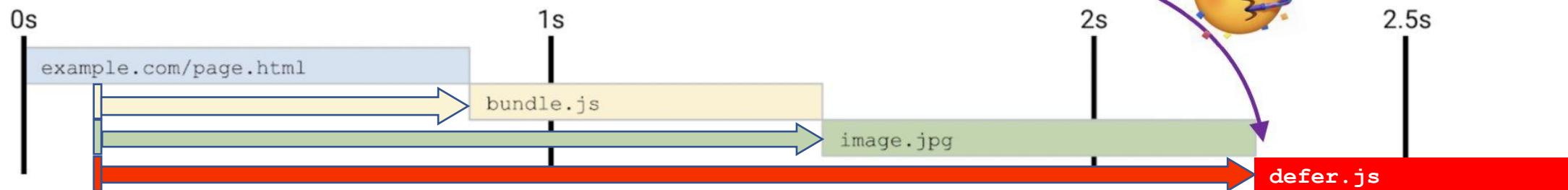


# Preload only helps in certain situations

## Without preload



## With preload



# Preload is often misunderstood

381 resources are being preloaded, but are not used during page load.

Preloaded resources are fetched at a high priority, delaying the arrival of other resources in the page. Resources that are never actually used by the page, that means potentially critical requests will be delayed, slowing down the page.

- /css/chunk-0112032d.cc09ddcf.css
- /css/chunk-0150f84c.6cbbfa5a.css
- /js/chunk-22acc54.21481e62.js
- /css/chunk-0222f9ab.69690fd5.css
- /js/chunk-22acc54.21481e62.js

Expand All

Relevant Experiments

Remove Unused Resources

This experiment identifies critical resources that are not used during page load.

Assets included

Timeline key: Visual change, Visual change + Layout Shift, Largest Contentful Paint, Largest Contentful Paint + Layout Shift

Adjust Priority Settings

| Time      | 3.0s | 3.5s | 4.0s | 4.5s | 5.0s | 5.5s | 6.0s | 6.5s | 7.0s |
|-----------|------|------|------|------|------|------|------|------|------|
| 1: Before |      |      |      |      |      |      |      |      |      |
| 2: After  |      |      |      |      |      |      |      |      |      |



Harry Roberts ✓  
@csswizardry

#lazyweb Can you trivially disable rel=preload in Next.js? I'd say preload is one of the most misunderstood and overused recent performance 'improvements', and, when employed naively, usually does more harm than good.

12:36 PM · Jan 14, 2021 · Twitter Web App

## Brinteaser:

What happens if you preload an async/defer JS that's on the bottom of the HTML?

WOMEN  
WITH BEARDS



It's true you don't see many dwarf women.

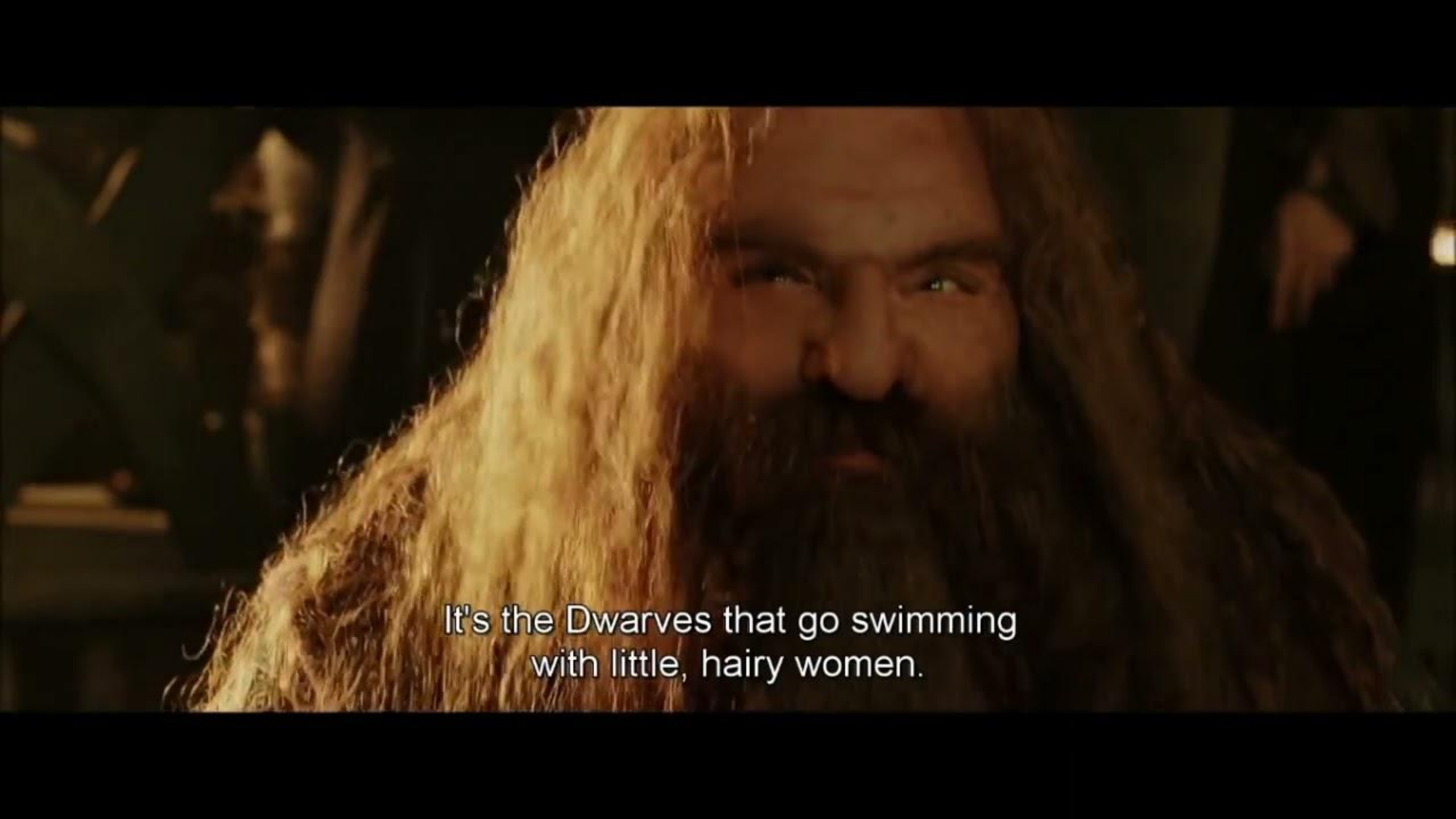


And, in fact, they are so alike in voice and appearance, that they're often mistaken for dwarf men.



It's the beards.





It's the Dwarves that go swimming  
with little, hairy women.





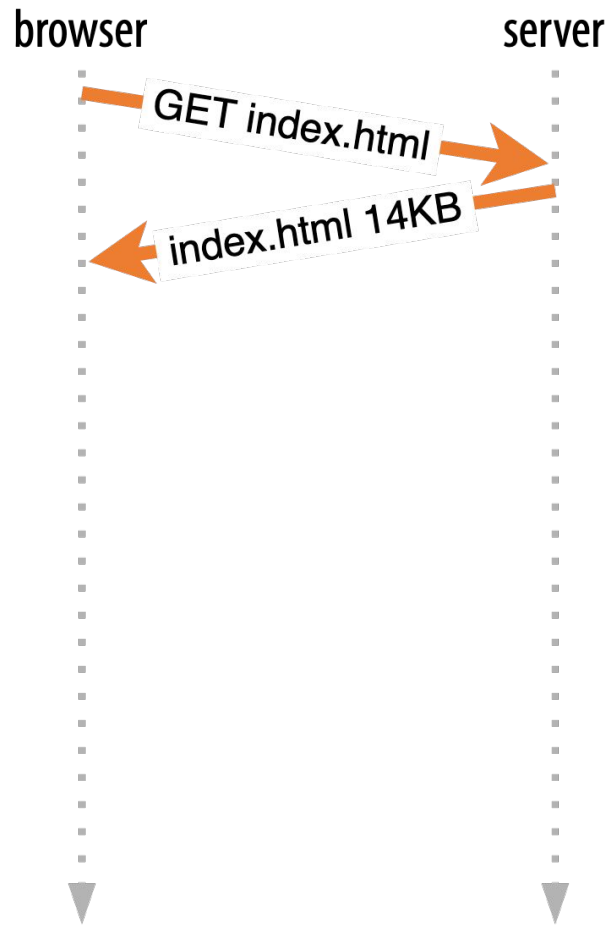




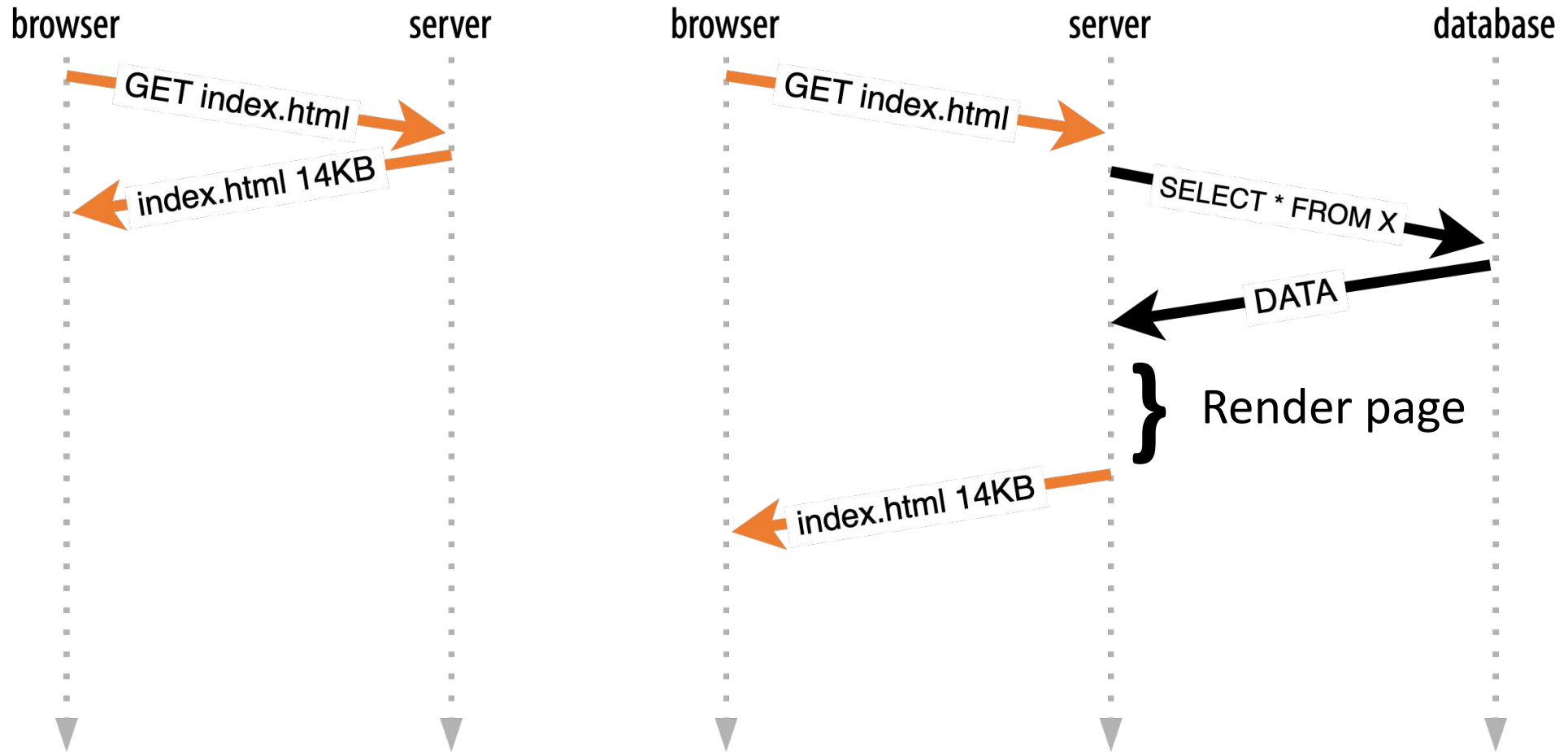


THE RETURN  
OF SERVER PUSH

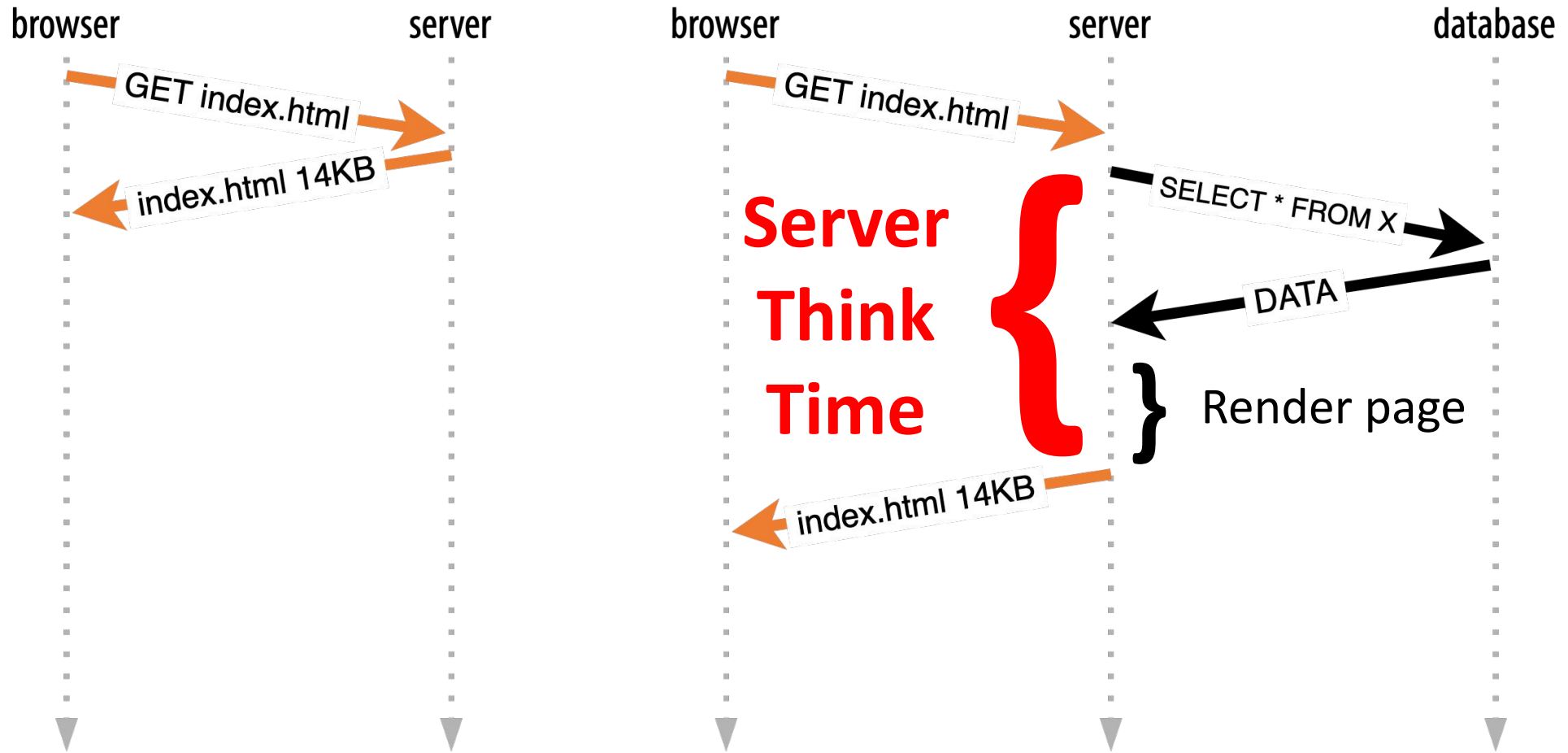
# Server Think Time



# Server Think Time

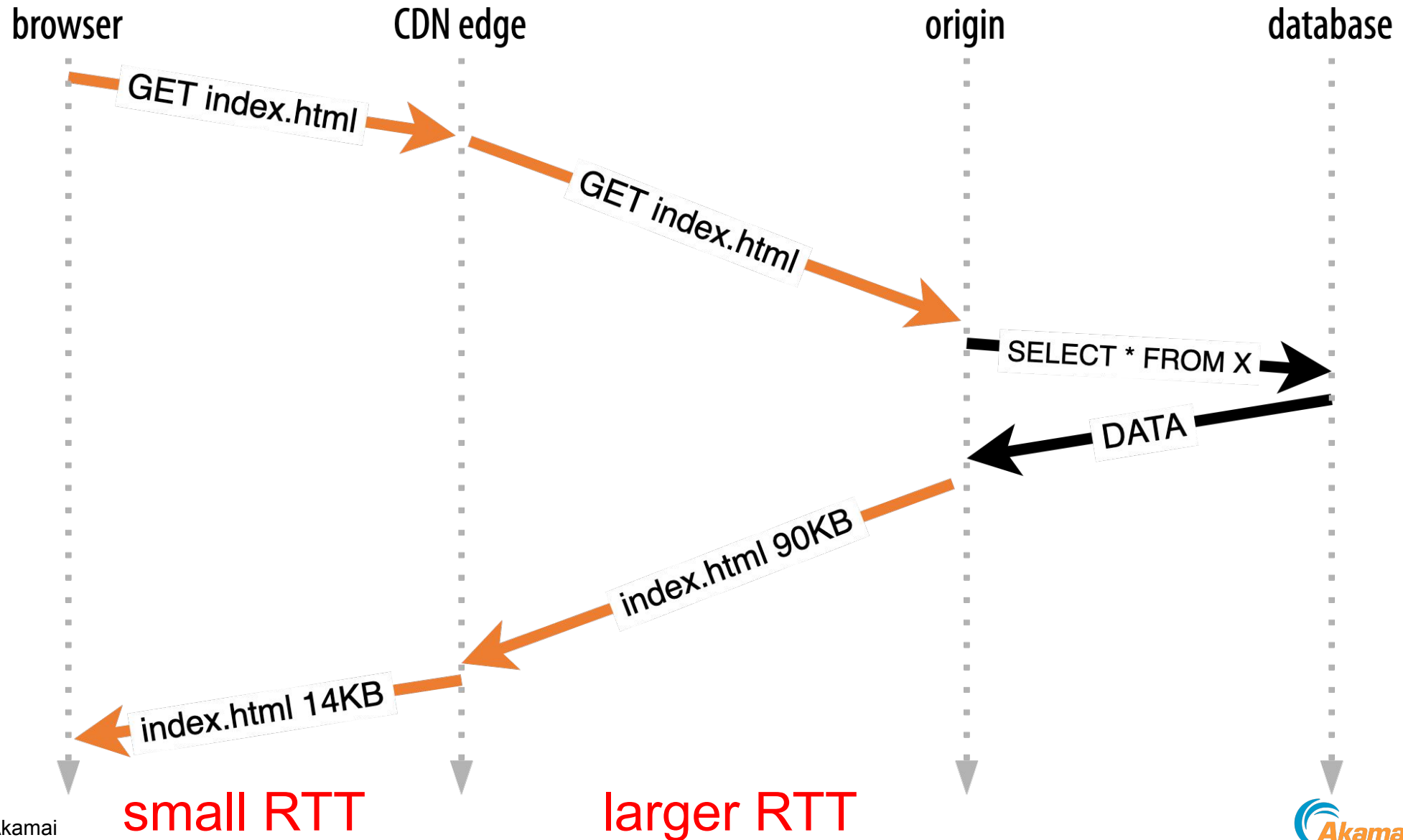


# Server Think Time

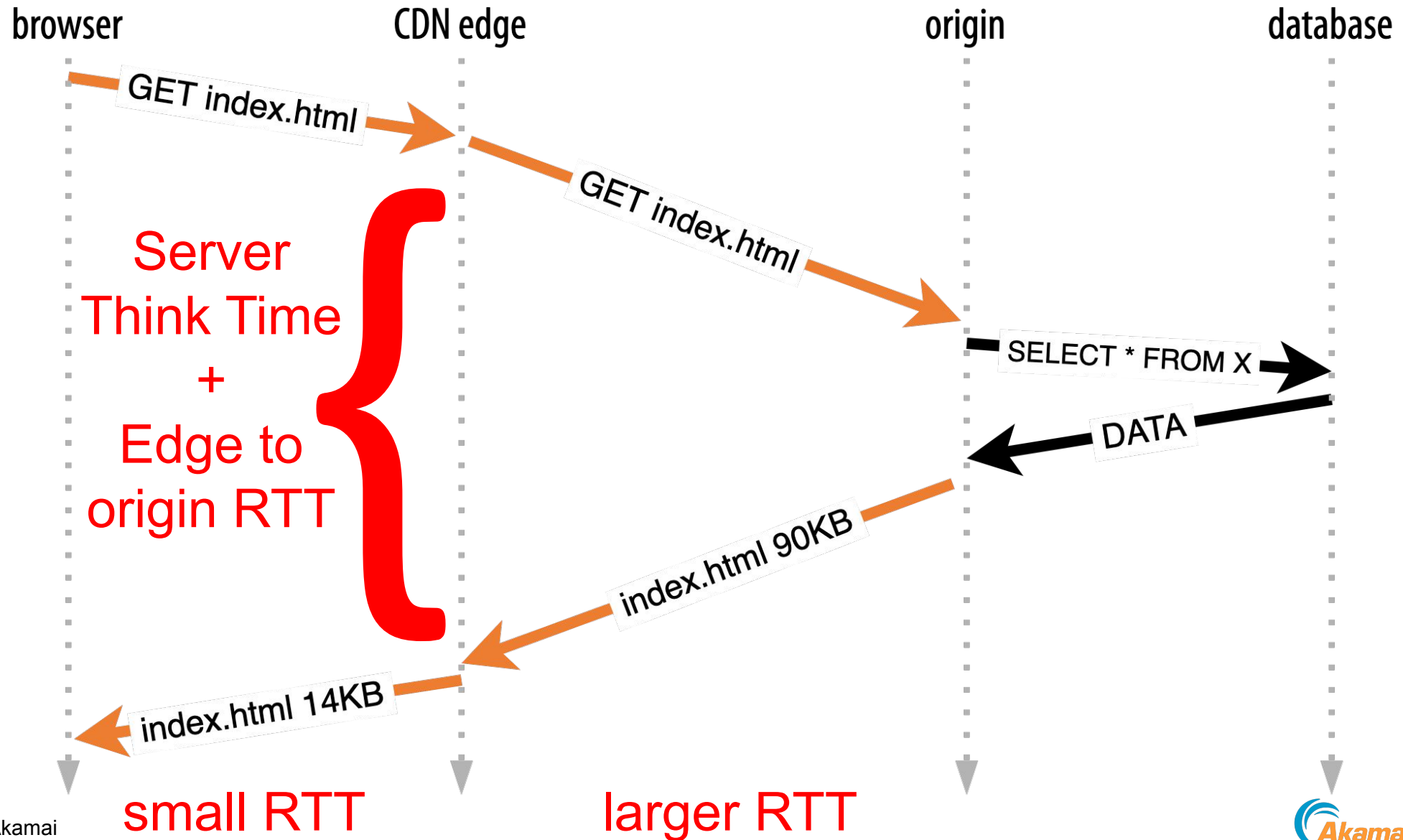




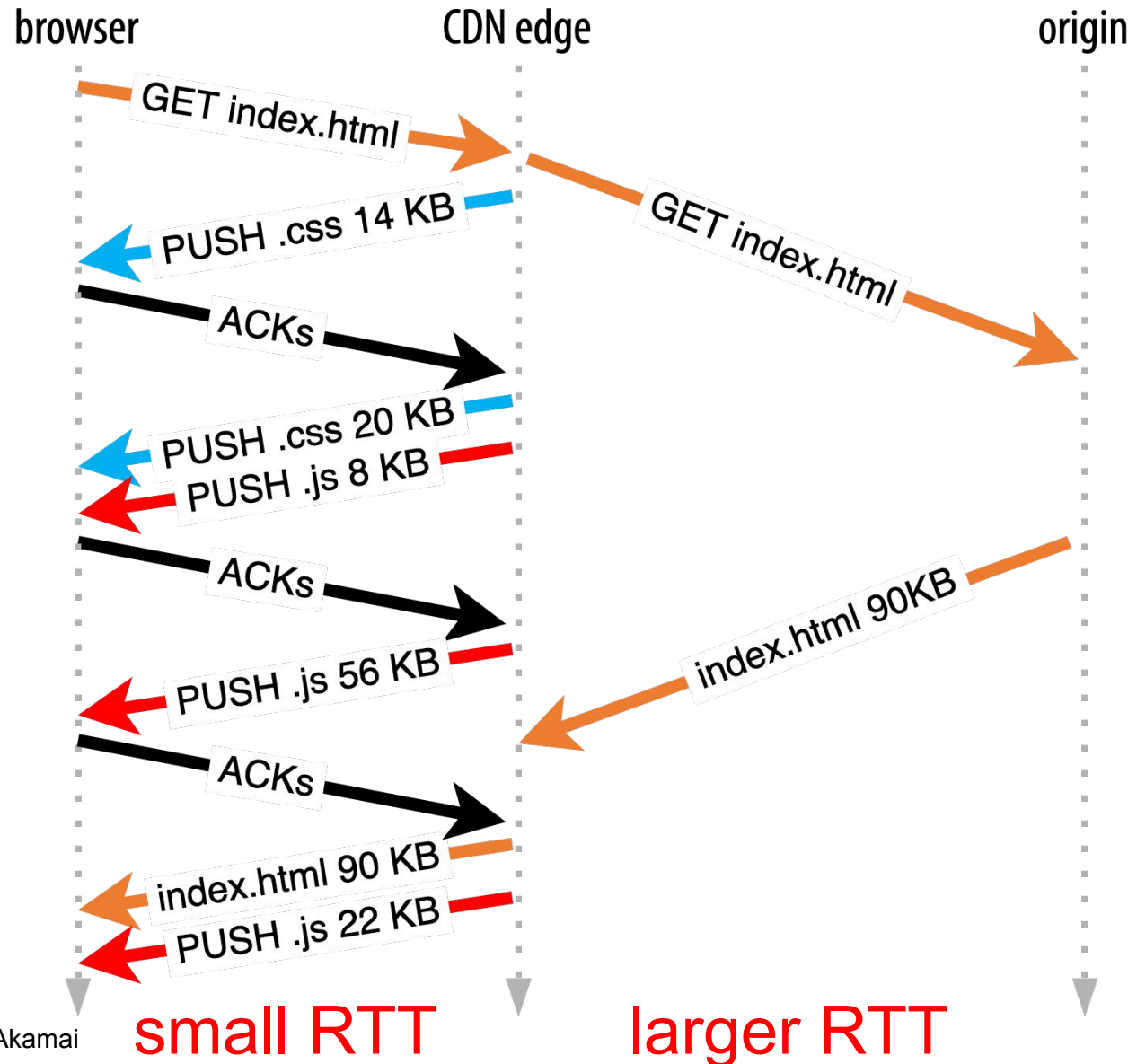
# CDN: Origin Fetch Delay



# CDN: Origin Fetch Delay



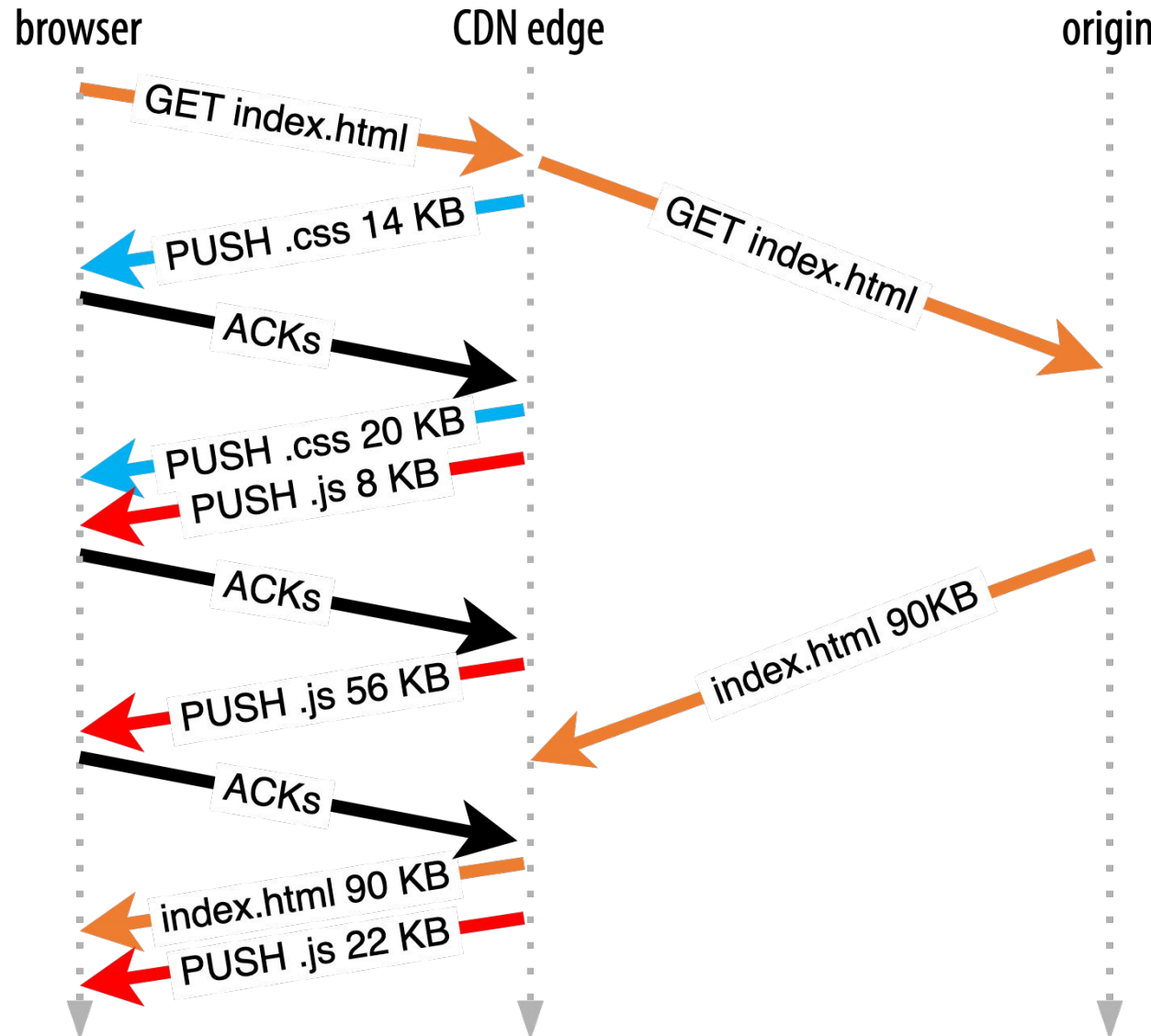
# The Promise of Push



Server PUSHes data during wait

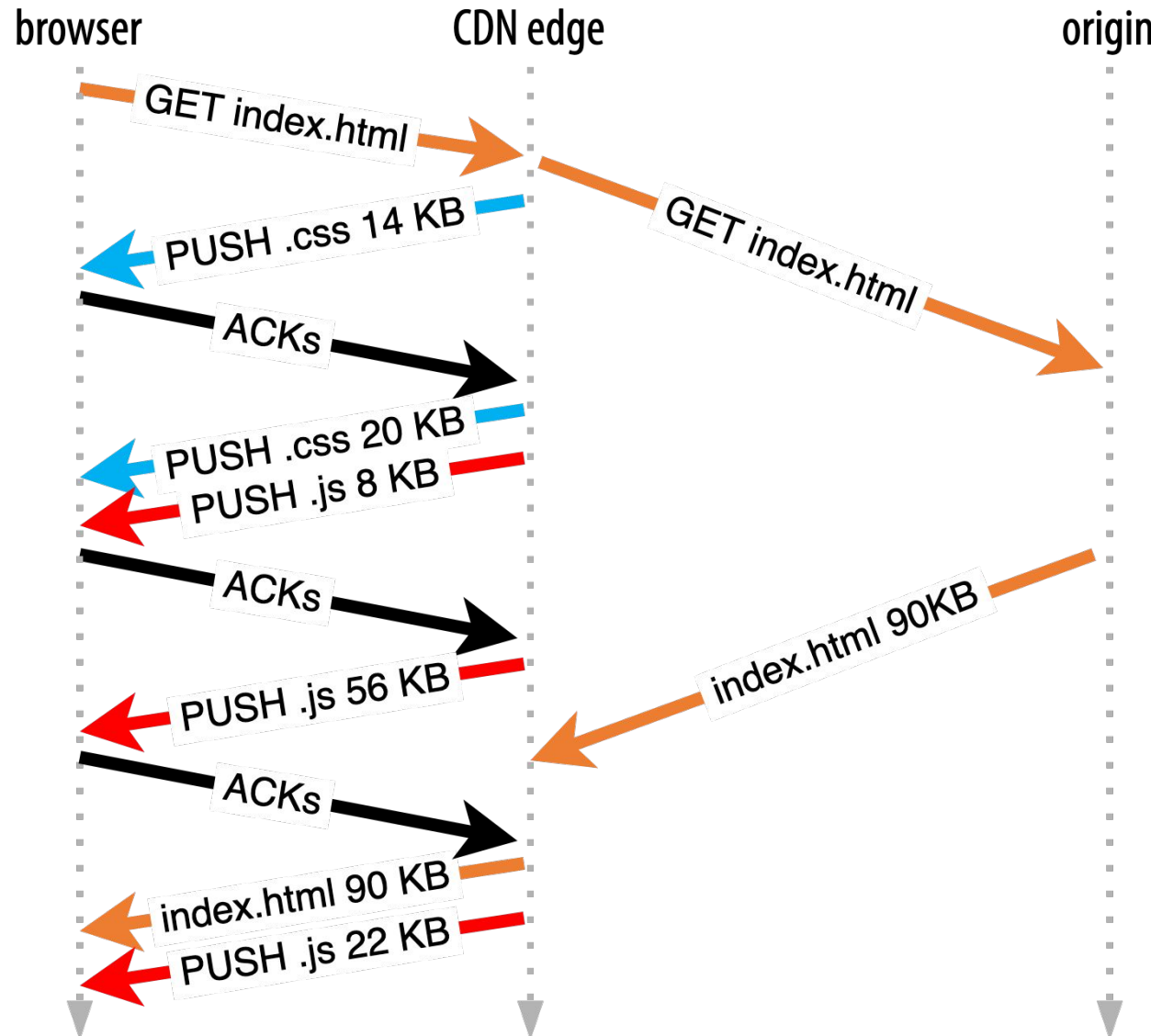
critical **.CSS** and **.JS**

# The Problems of Push



1. PUSH cached data
2. PUSH only 1<sup>st</sup> party data
3. Many implementation bugs/inconsistencies

# The Decay of Push



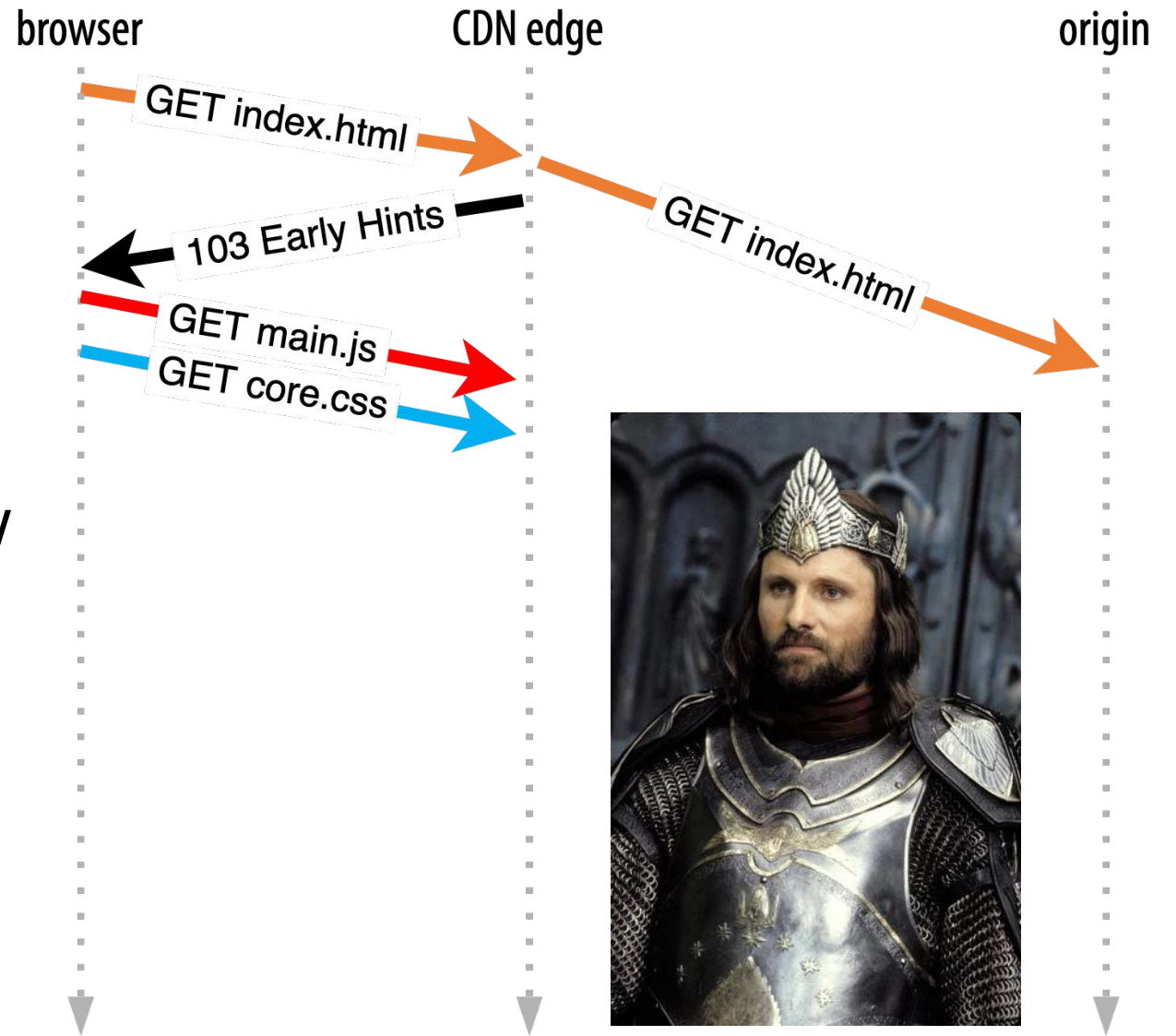
1. Deprecated in Chrome for HTTP/2
2. Never implemented for HTTP/3



# A New King Arises

## 103 Early Hints

- Don't PUSH data, but send *links to resources*
- **Browser** determines itself what to request and when/how
  - Caching!
  - Prioritization!



# How it works

In HTML:

```
<link rel="preload" href="/core.css" as="style">  
<link rel="preload" href="https://static.domain.com/font.woff2" as="font" crossorigin>  
<link rel="preconnect" href="https://soonneeded.org">
```

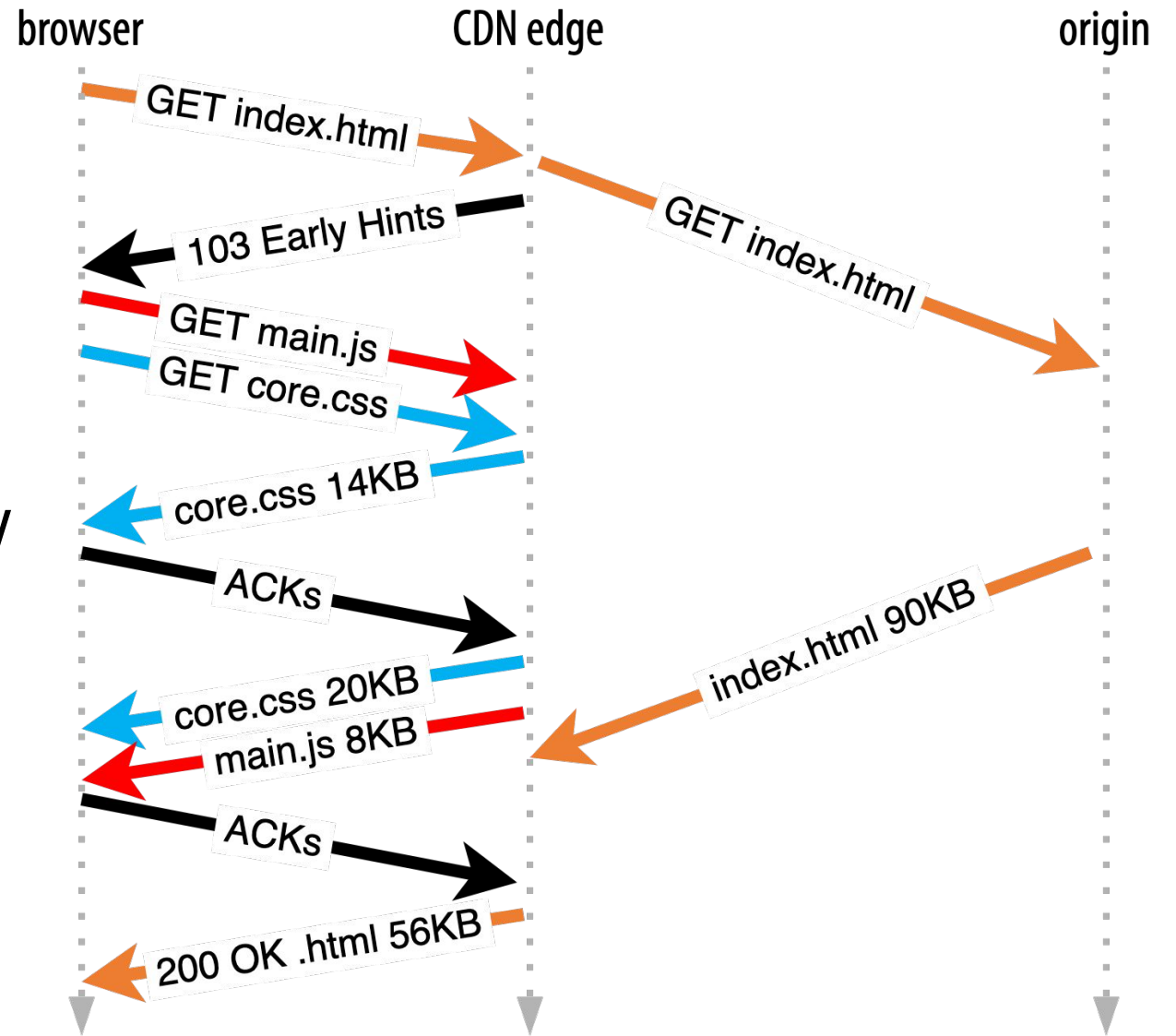
In 103 Early Hints **HTTP response headers:**

```
Link: </core.css>; rel=preload; as=style;  
Link: <https://static.domain.com/font.woff2>; rel=preload; as=font; crossorigin  
Link: <https://soonneeded.org>; rel=preconnect
```

# A New King Arises

## 103 Early Hints

- Don't PUSH data, but send *links to resources*
- **Browser** determines itself what to request and when/how
  - Caching!
  - Prioritization!

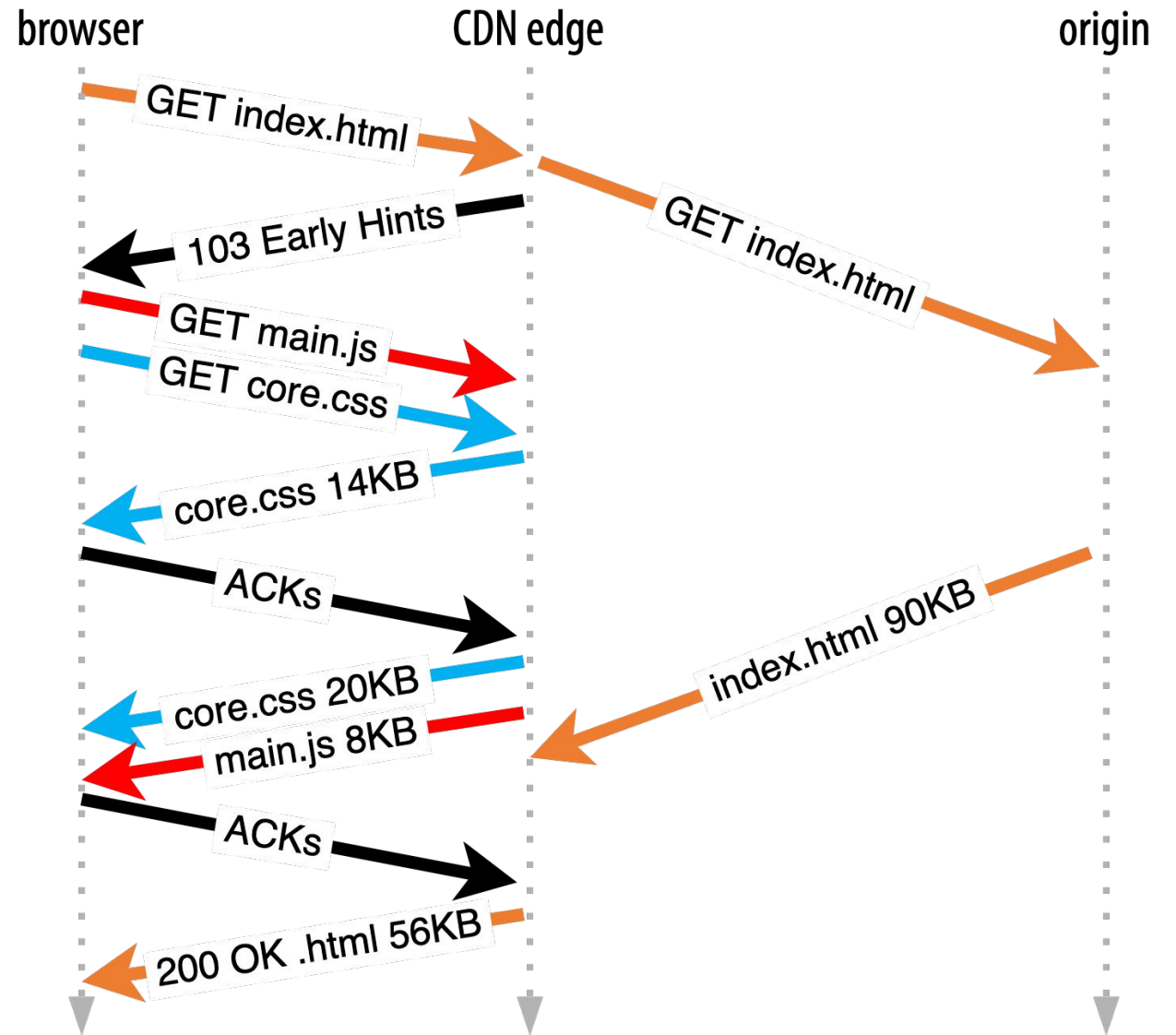




# The Killer Feature

## 103 Early Hints

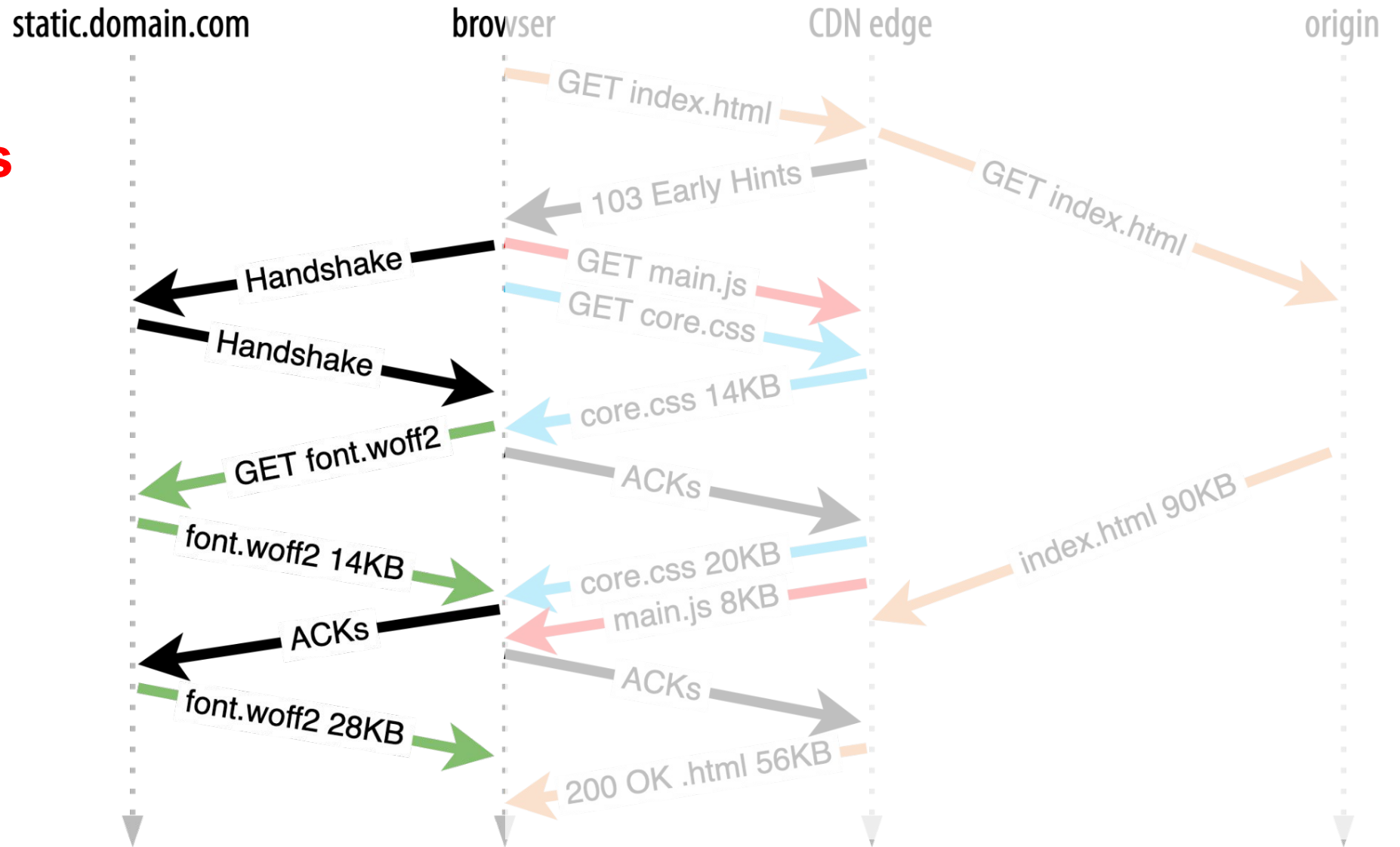
- Supports secondary domains and 3<sup>rd</sup> party resources!



# All that Glitters is Gold

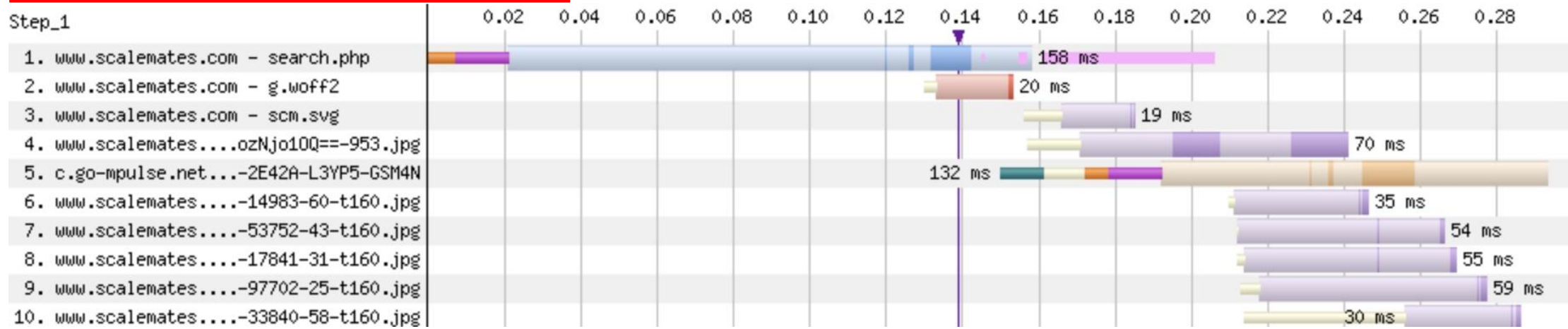
## 103 Early Hints

- Supports secondary domains and 3<sup>rd</sup> party resources!



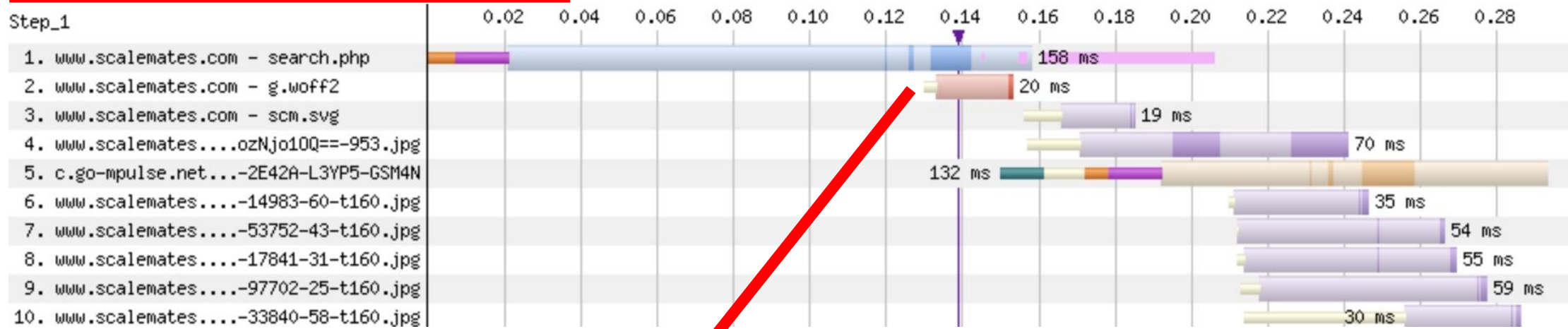
# 103 Early Hints impact

103 Off

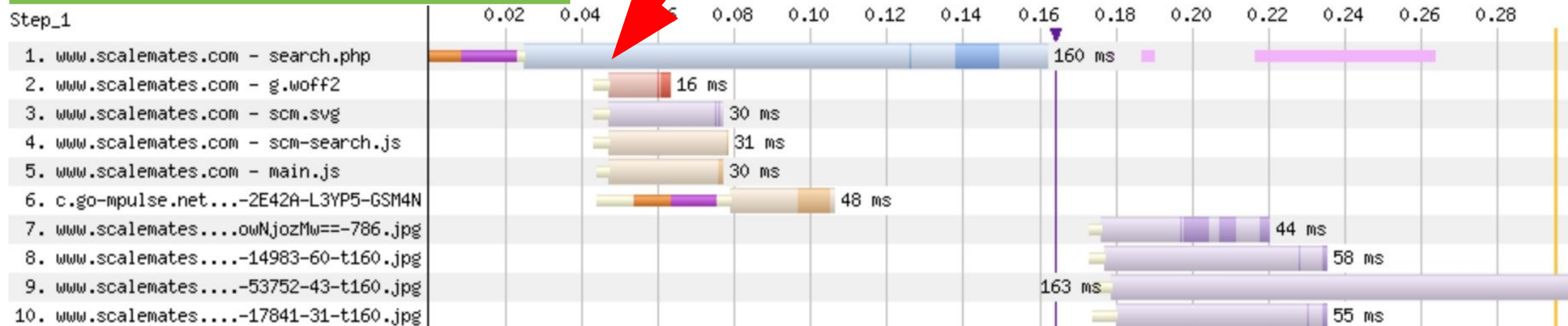


# 103 Early Hints impact

103 Off



103 On



# Preload vs Preload

In HTML: preload **late-discovered resources**

```
<preload font.woff2>
```

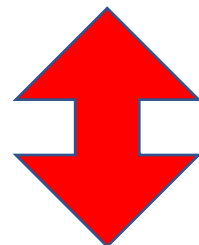
```
<preload hero.jpg>
```

# Preload vs Preload

In HTML: preload **late-discovered resources**

```
<preload font.woff2>
```

```
<preload hero.jpg>
```



In 103 Early Hints: preload **critical resources!**

```
Link: <core.css>; rel=preload;
```

```
Link: <main.js>; rel=preload;
```



Preload HTML

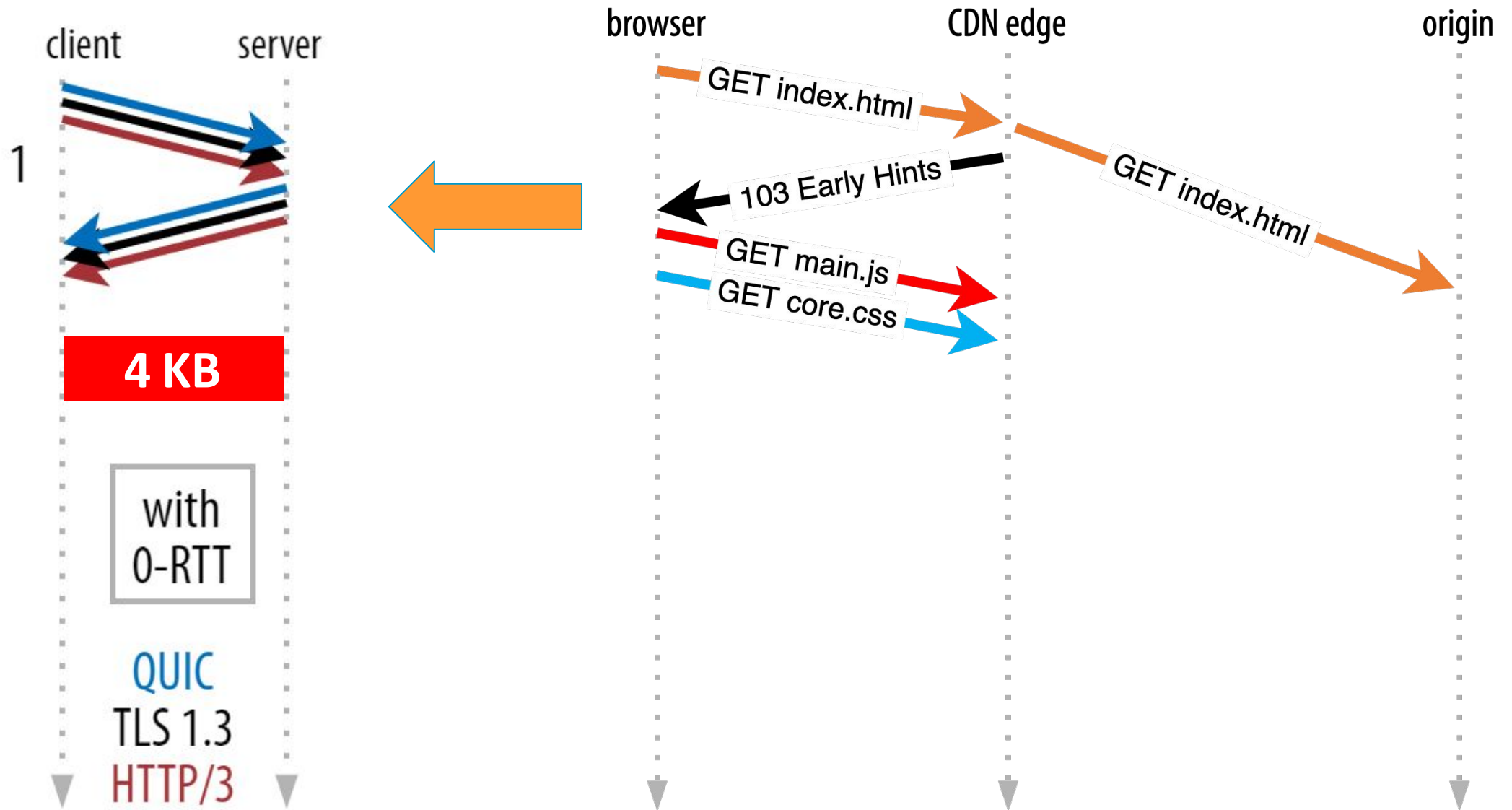
Preload HTTP Headers

Preload nopush

Preload 103 Early Hints

**Preload Scanner**

# 0-RTT + 103 Early Hints : a Match made in Heaven!





INTO THE WEST

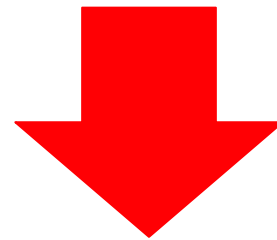
Be careful with protocol-related features



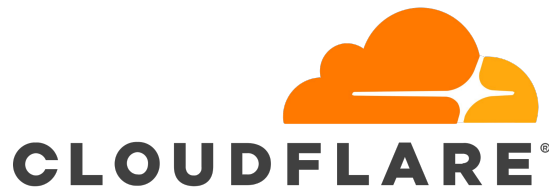
# Network (protocol) configuration is important!

Congestion control  
Initial congestion window size  
Prioritization  
HTTP/3 + 0-RTT support

...



Use a CDN





**LOOKS LIKE MEAT'S BACK ON THE MENU!**

# Would You Like To Know More?



@programmingart

rmarx@akamai.com

# Take Home Messages

HTTP/2 best practices still apply

Limited data during start of connection

Prioritization is a **dark art**

Preload makes your **head hurt**

103 Early Hints are the coolest thing since Server Push

**Lord of the Rings is the best movie trilogy**