

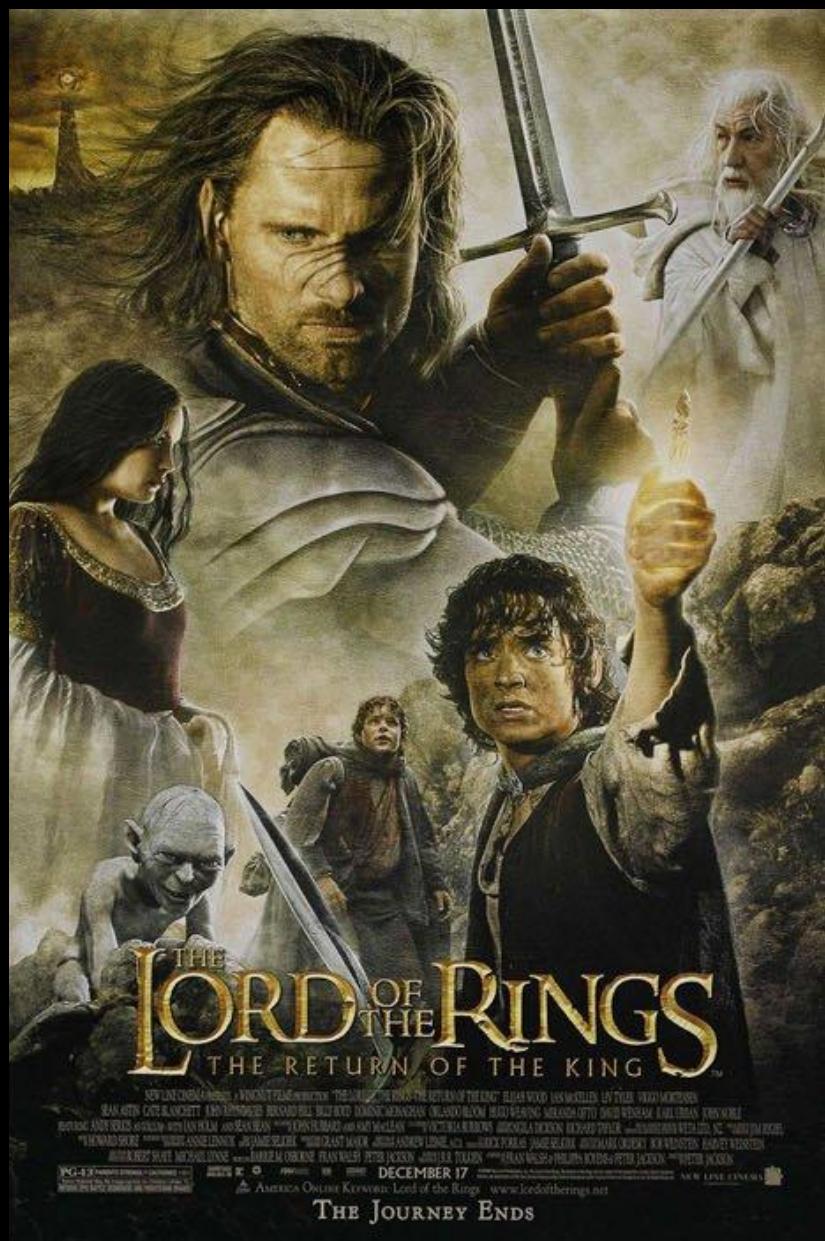
we ❤️ speed

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.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.html...

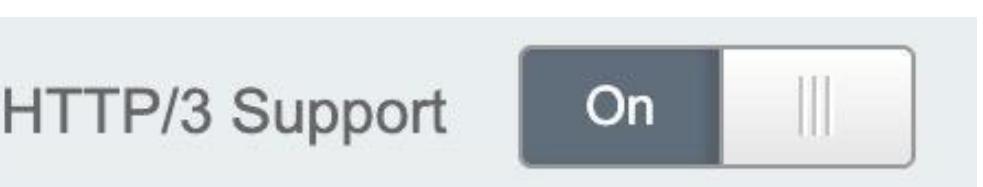
Together with RFC 9204 (QPACK header compression) and RFC 9218 (Extensible Priorities) it ushers in an important new chapter for the Web!

Proud to have been part of this!



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



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

The Network Protocol

Server Push
103 Early Hints



Resource Hints
(preload, preconnect)

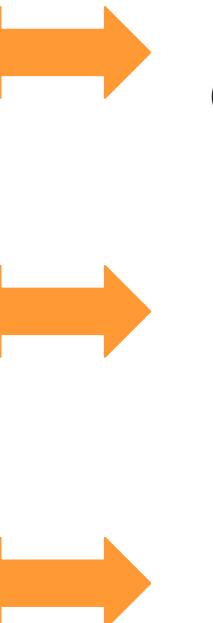
Fetch Priority

Lazy loading
Async / Defer

Poking the ~~Black Box~~ Dangerous Artefact!



~~Server Push~~
103 Early Hints



Resource Hints
(preload, preconnect)

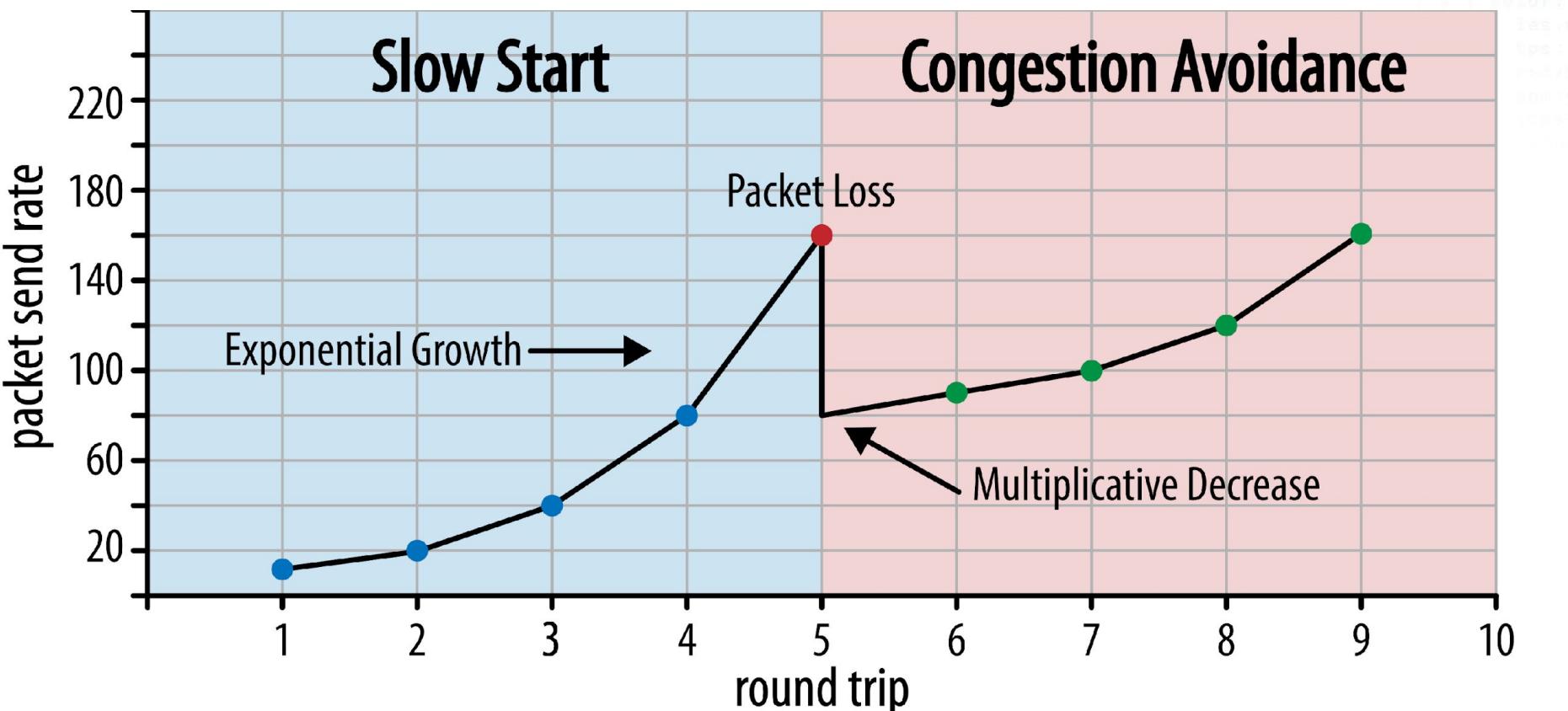
Fetch Priority

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

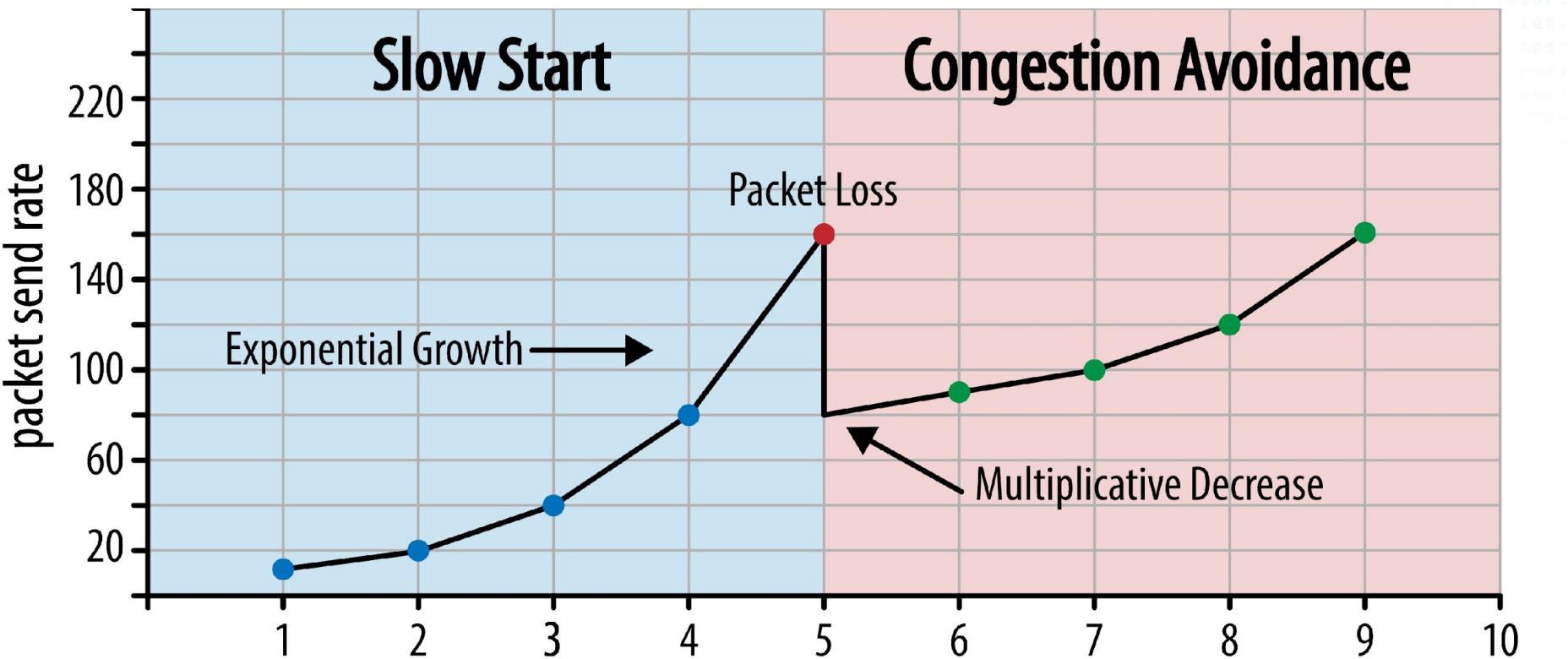


A close-up photograph of Aragorn's face from the Lord of the Rings movies. He has long brown hair and a beard, and is wearing ornate armor. He is looking directly at the camera with a slight smile and is pointing his index finger upwards towards the text.

ONE DOES NOT SIMPLY

KNOW THE AVAILABLE BANDWIDTH

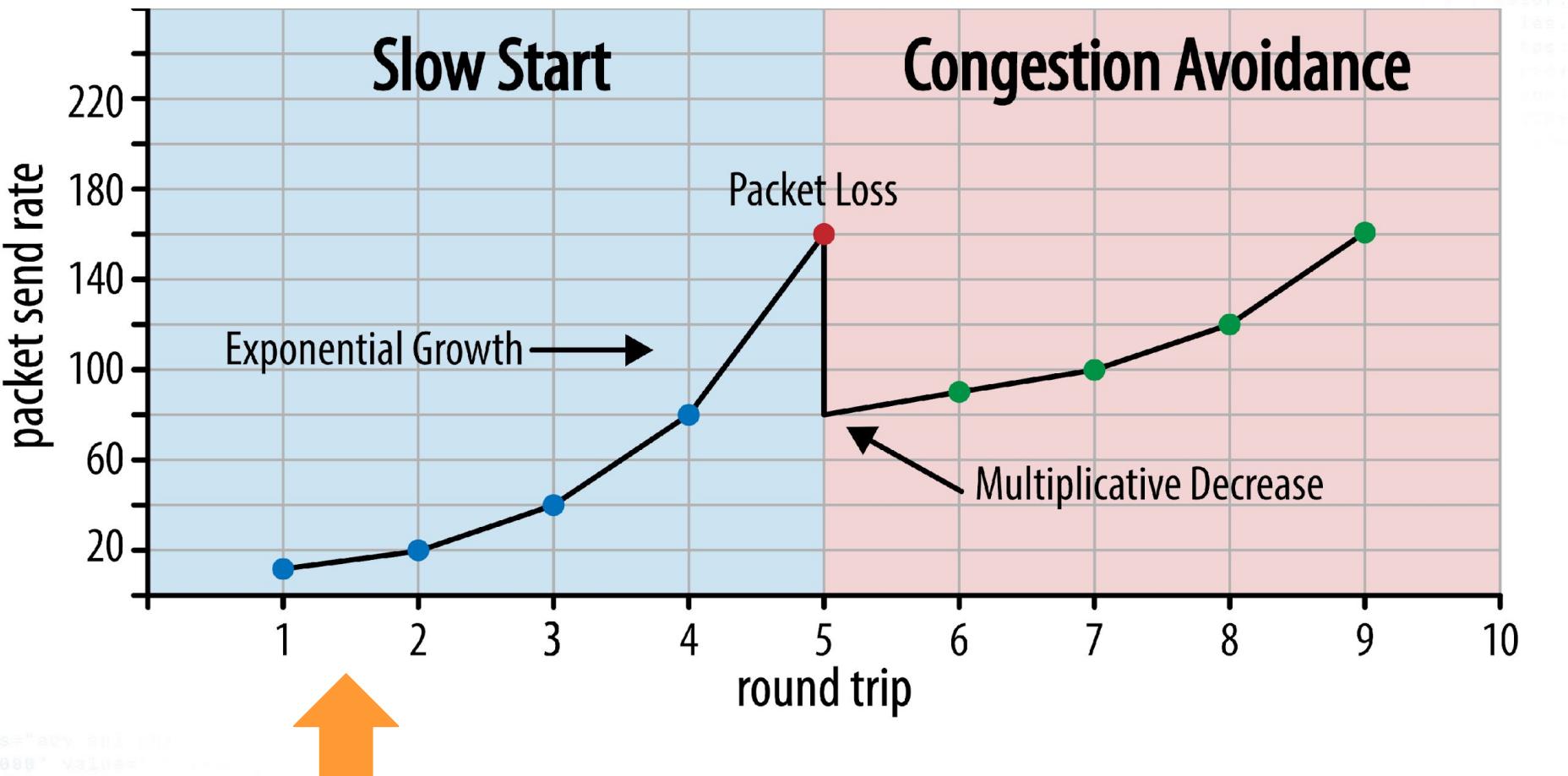
Congestion Control



10 packets
~14 KB

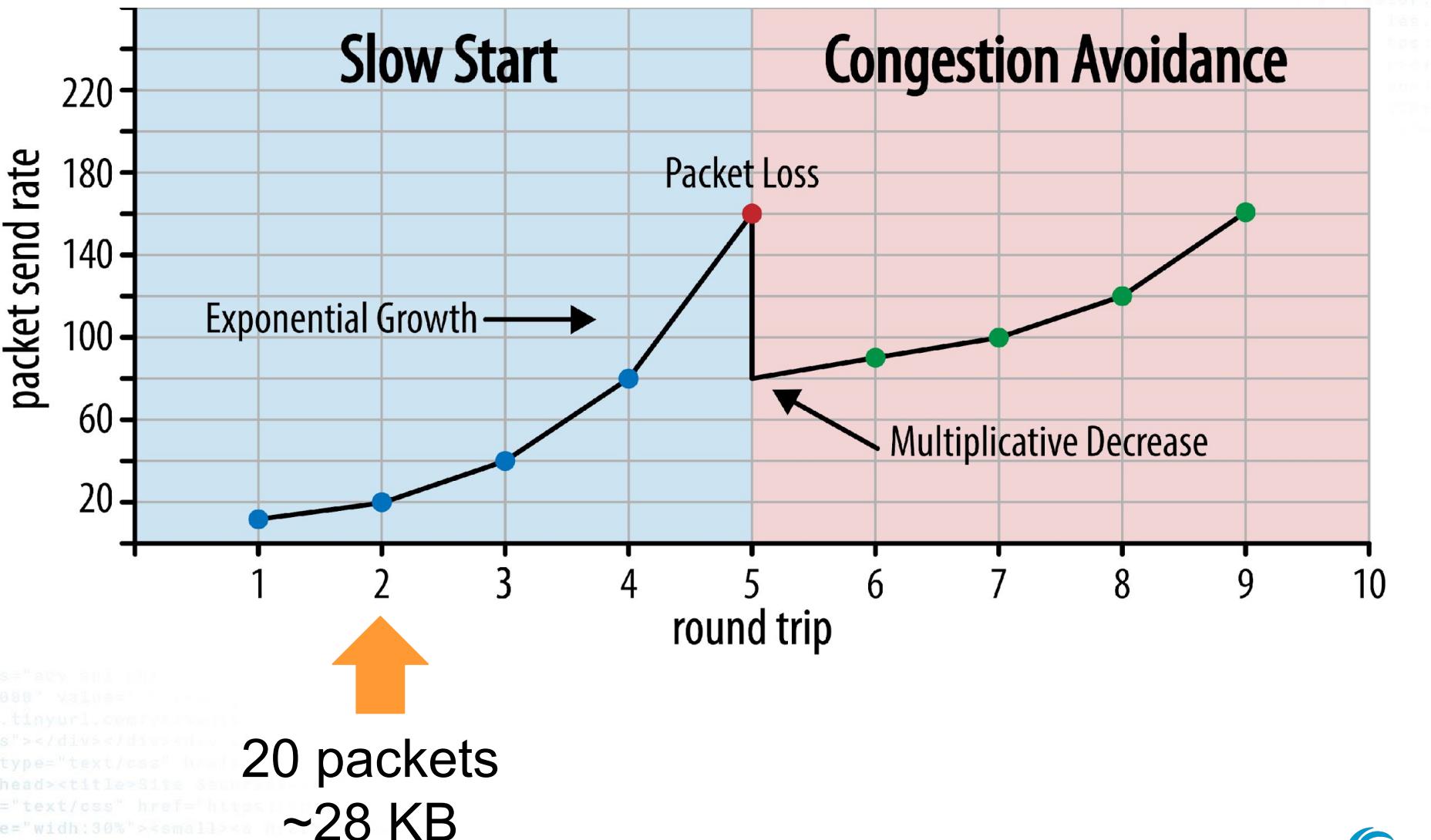
"Initial Congestion Window"

Congestion Control

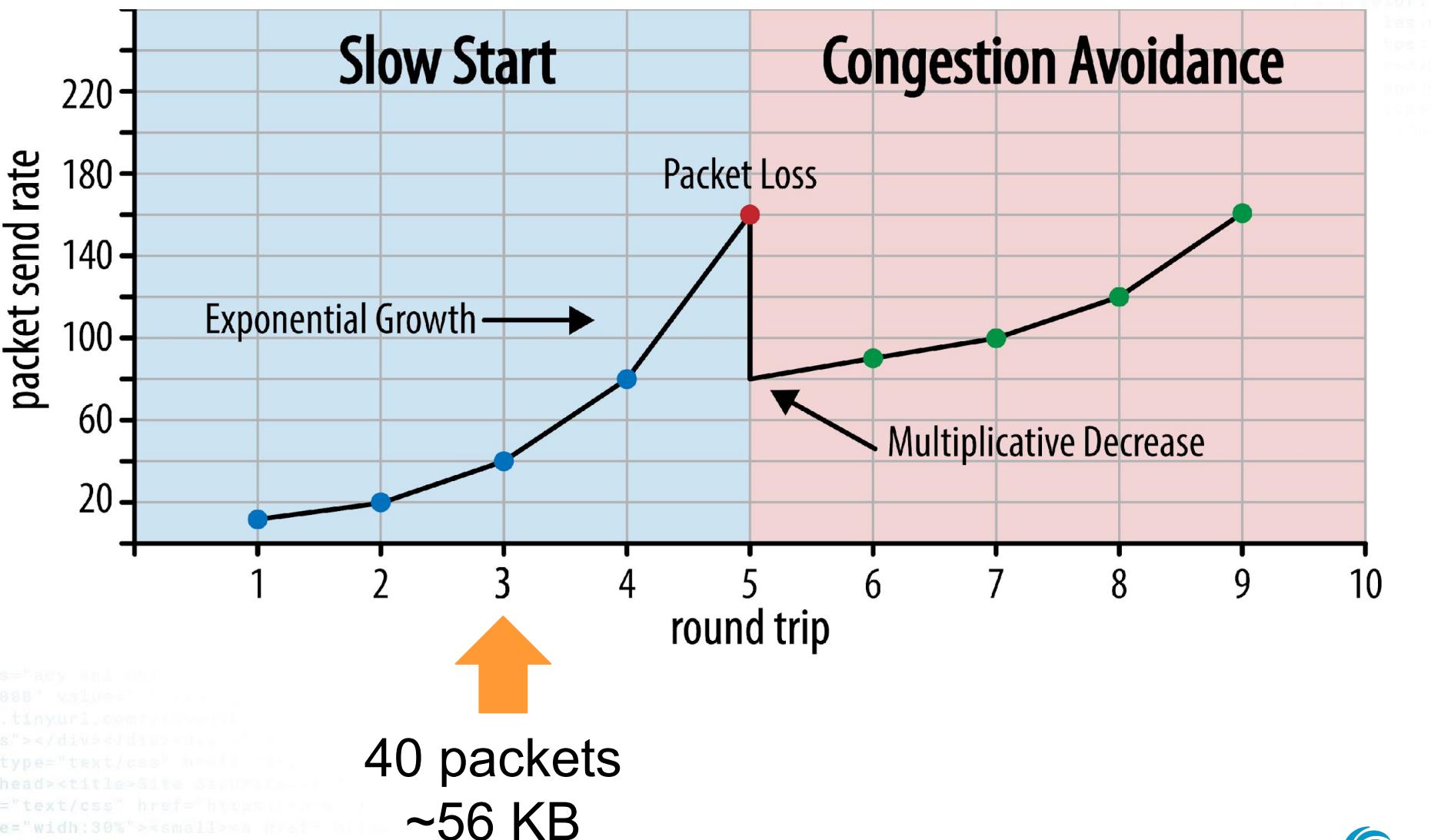


Wait full RTT for receiver to send
back acknowledgements (ACKs)

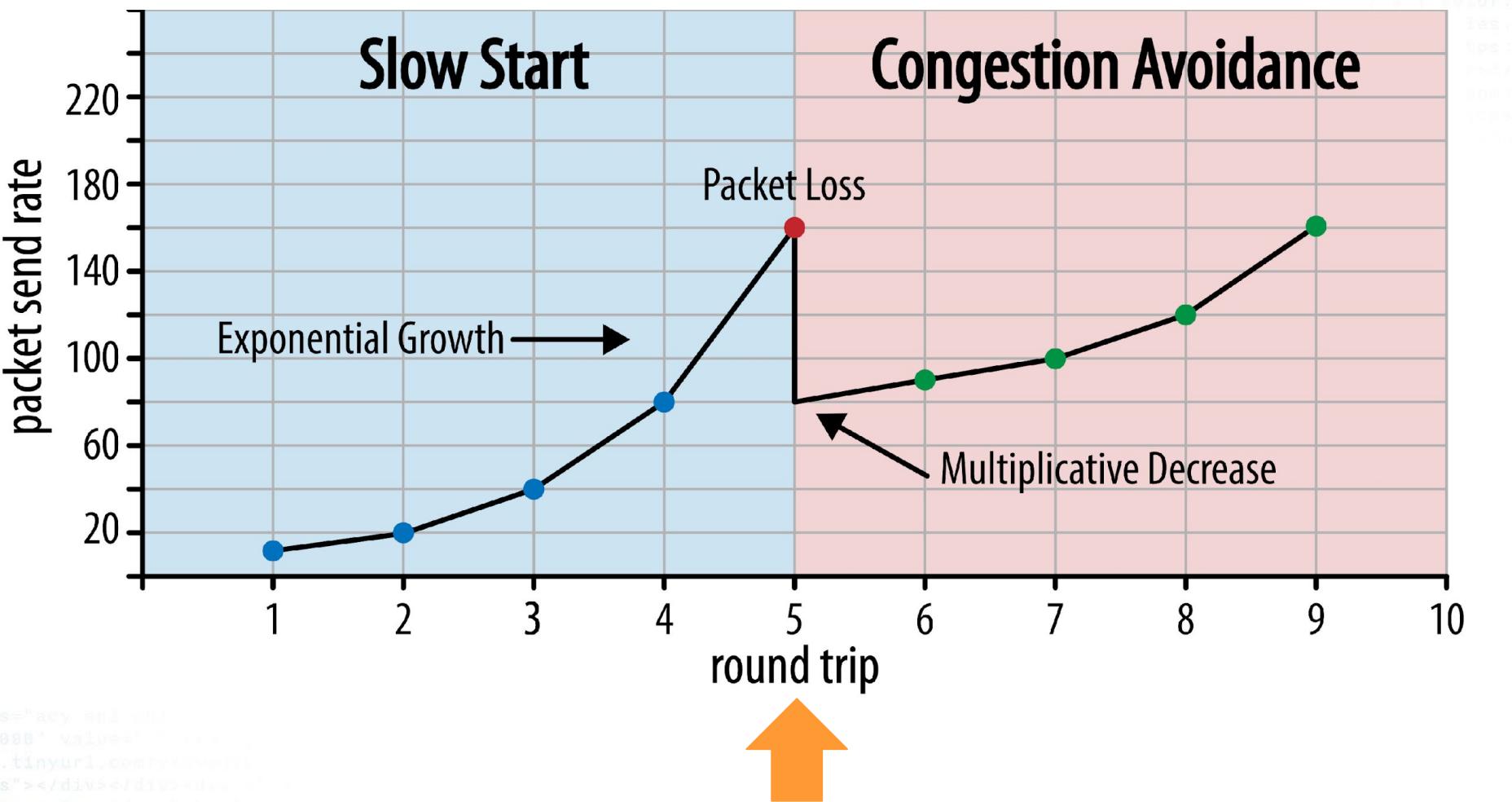
Congestion Control



Congestion Control



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++

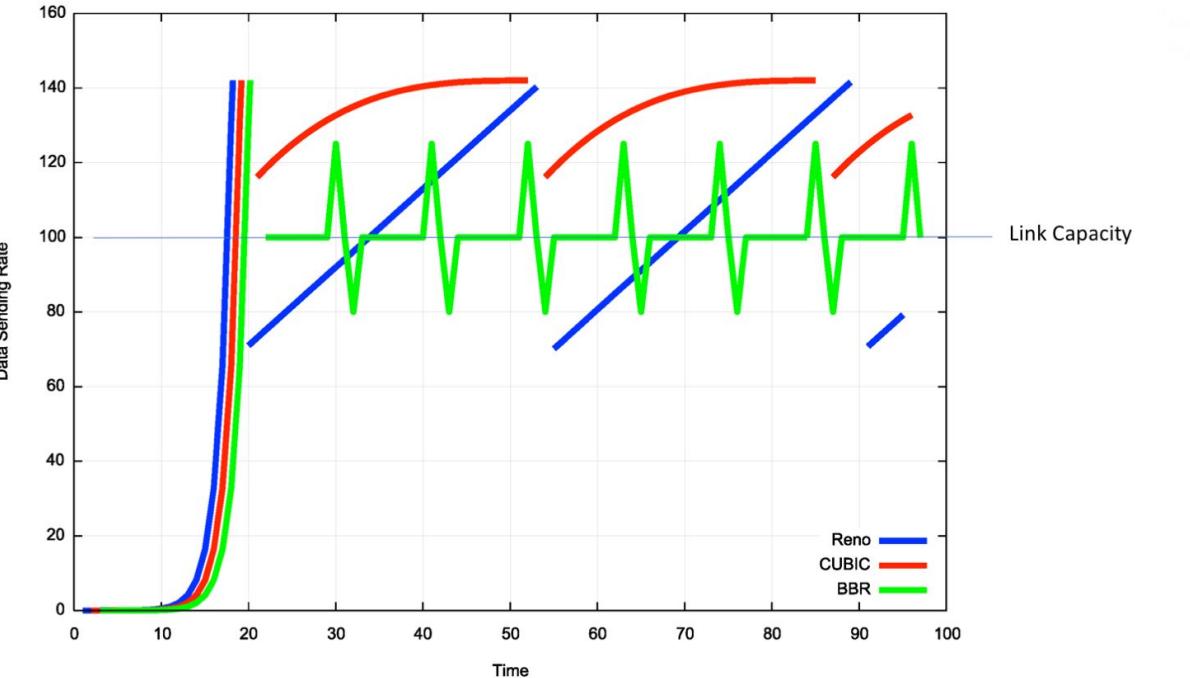
Fairness

Bufferbloat, AQM

ECN, L4s



Slow Start is common



Default	10 packets	14.6 KB
Most CDNs + hosting	+~ 30 packets	44 KB
Largest observed	100 packets	146 KB

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

Persistent Myth

WHY YOUR WEBSITE SHOULD BE UNDER 14KB IN SIZE

25 August 2022

Also available to read on [dev.to](#) (warning it is much larger than 14kB)

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

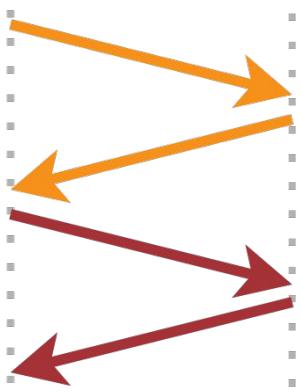
What is surprising is that a **14kB page can load much faster than a 15kB page** — maybe **612ms** faster — while the difference between a **15kB** and a **16kB** page is trivial.

<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



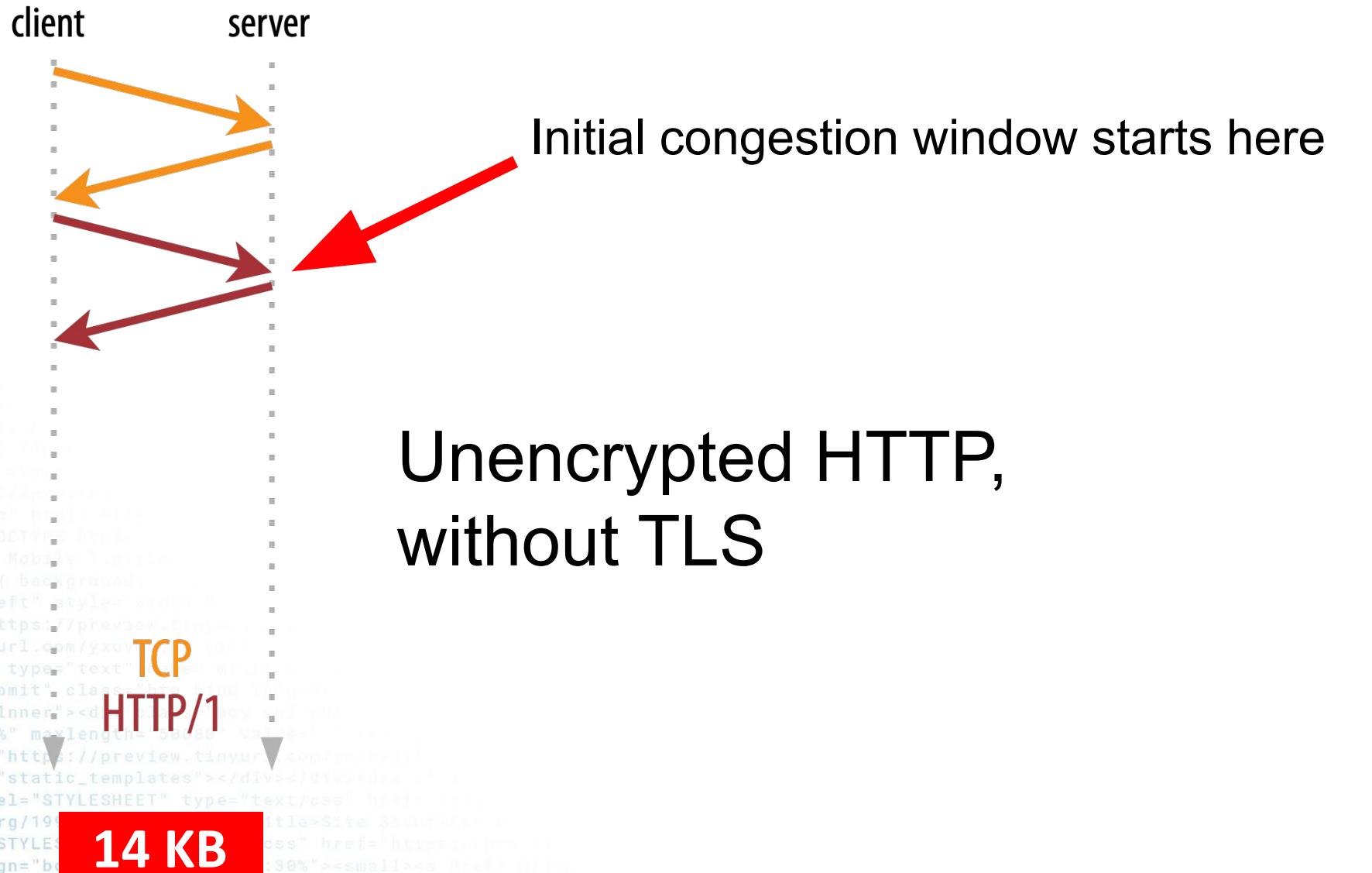
Unencrypted HTTP,
without TLS

TCP

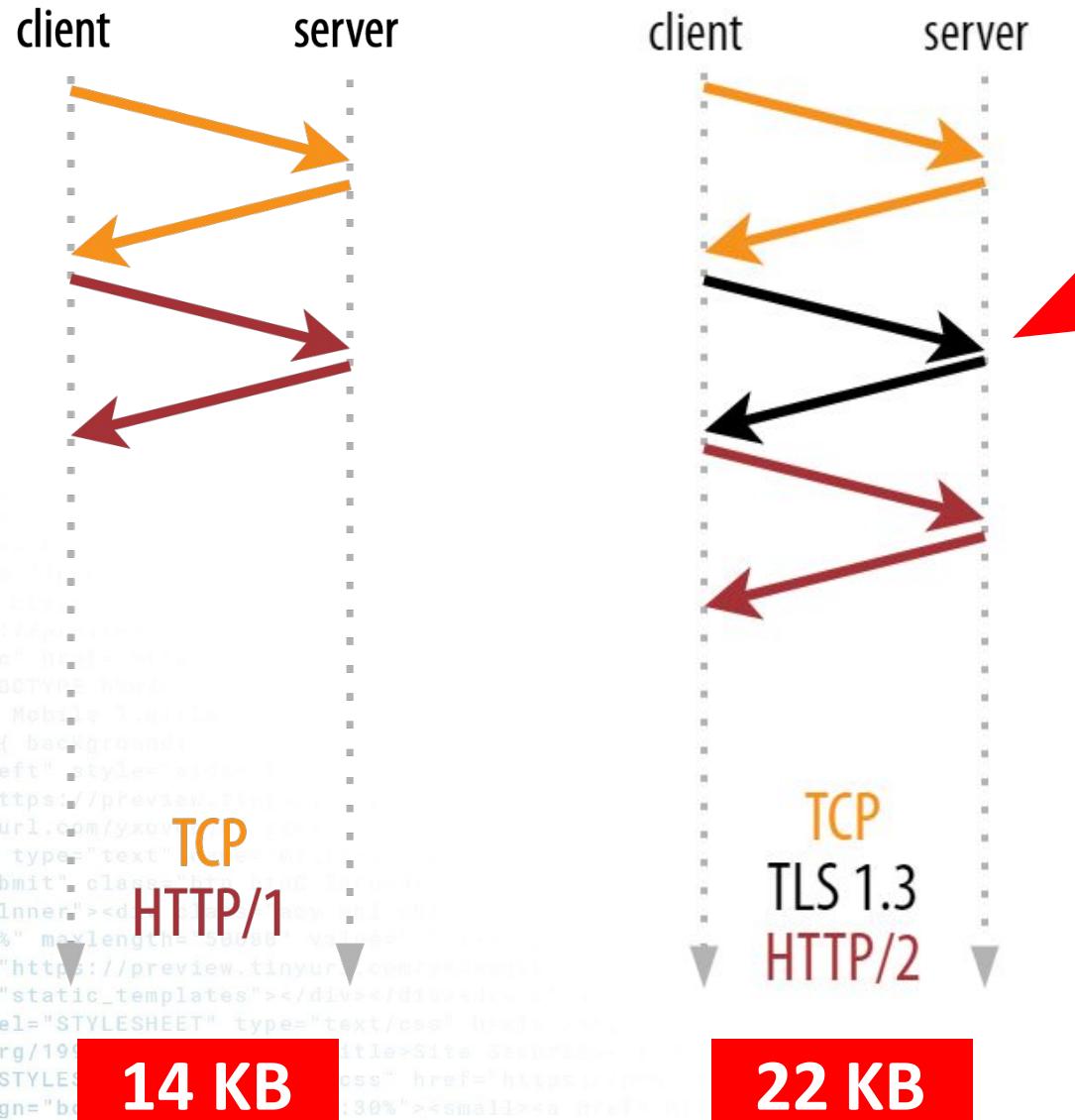
HTTP/1

14 KB

Data you can send in the first HTTP Round Trip



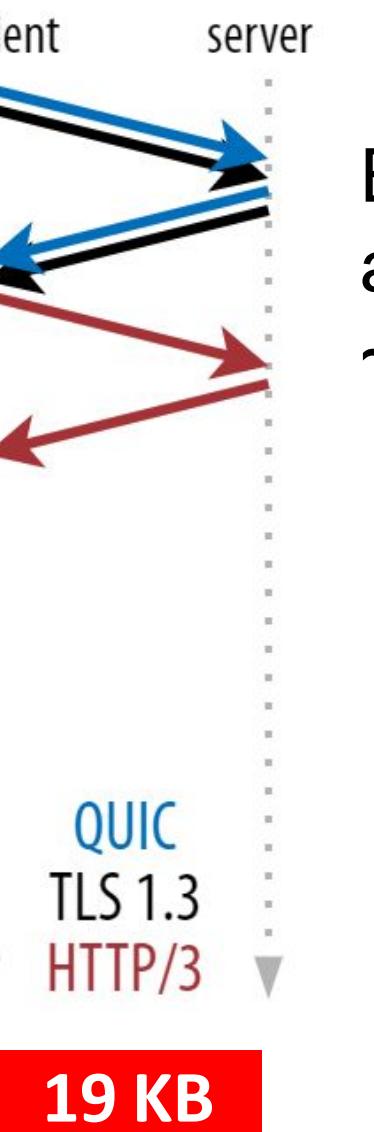
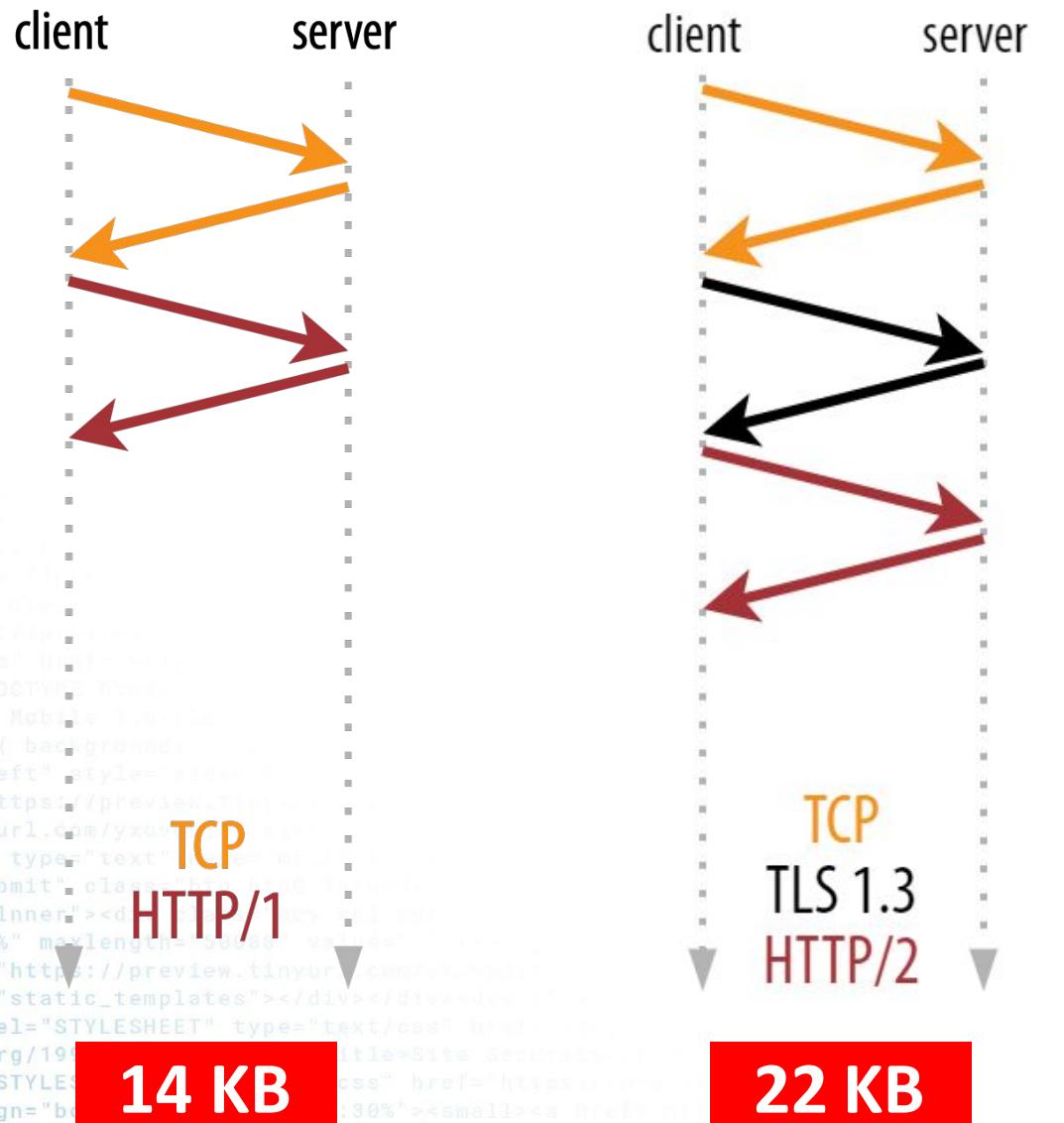
Data you can send in the first HTTP Round Trip



Initial congestion window starts here

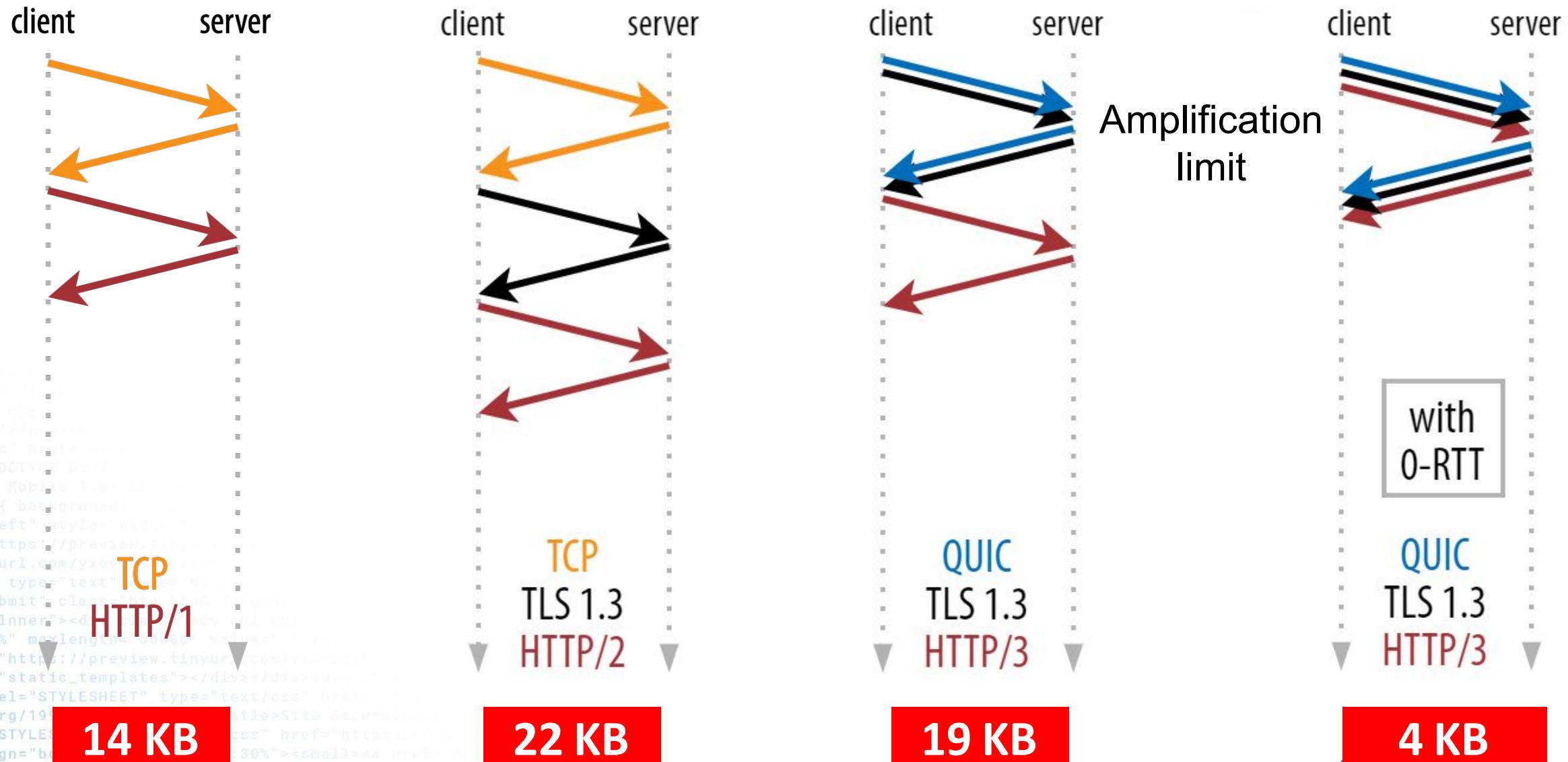
TLS handshake
uses *some* data,
but not the full 14KB

Data you can send in the first HTTP Round Trip



Early QUIC packets
are smaller:
~1280 bytes

Data you can send in the first HTTP Round Trip



Key Takeaways

Loading everything in 1st 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, ...





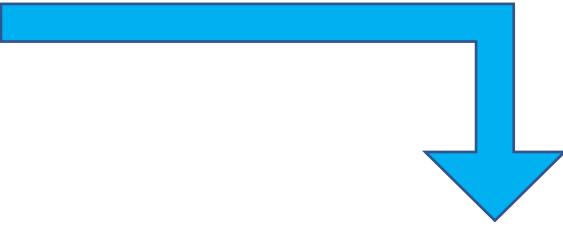




THE FELLOWSHIP
OF THE PRIORITIES

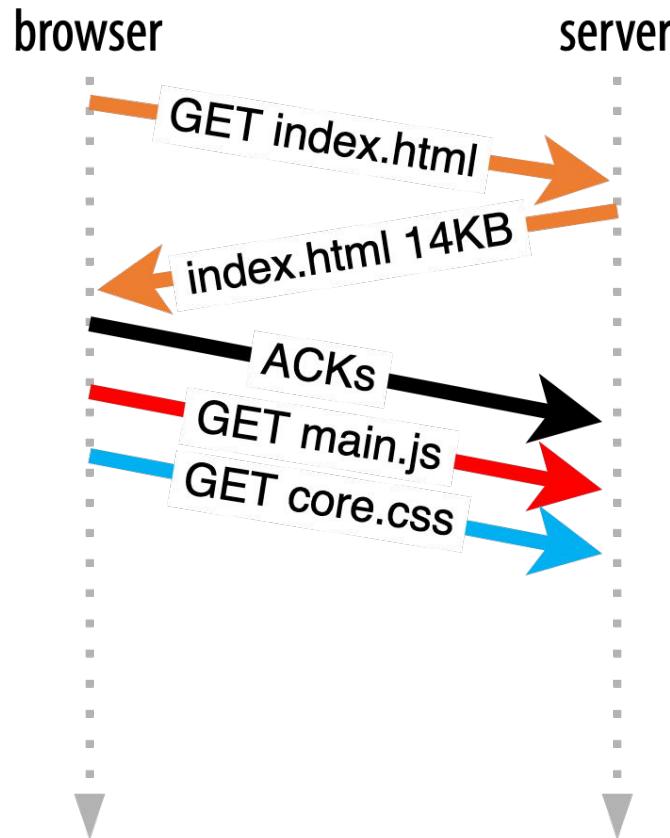
Simple page we'll use

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

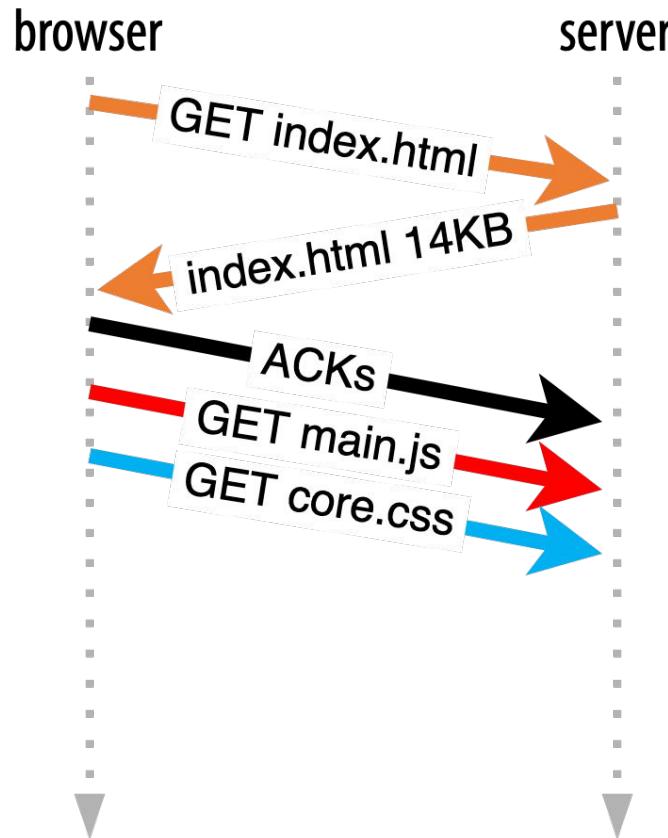


```
@font-face(font.woff2);
```

First RTT is always just HTML



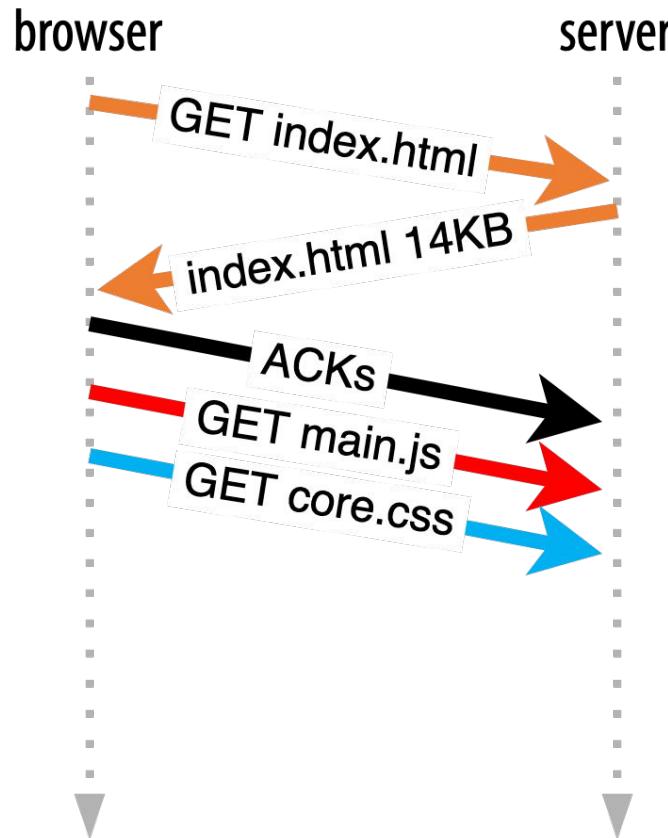
What to send first?



How to fill 20 packets / 28 KB?

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

What to send first?

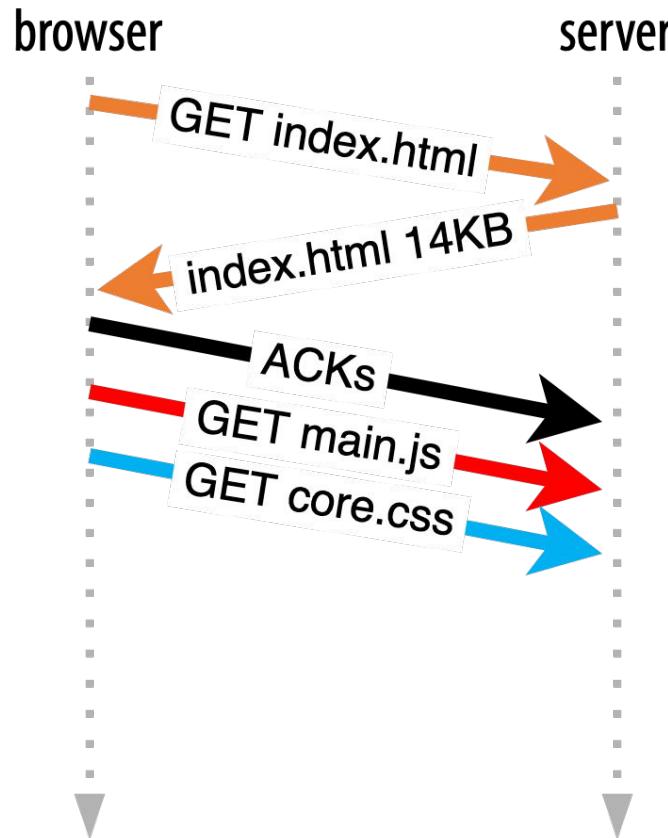


How to fill 20 packets / 28 KB?

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

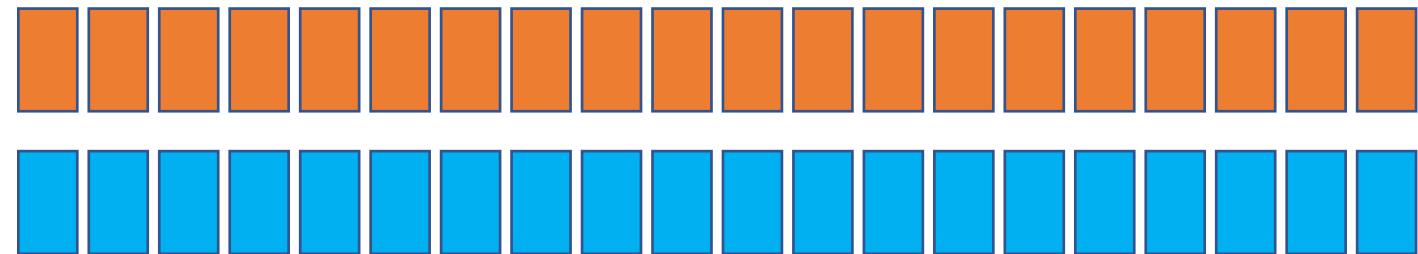


What to send first?

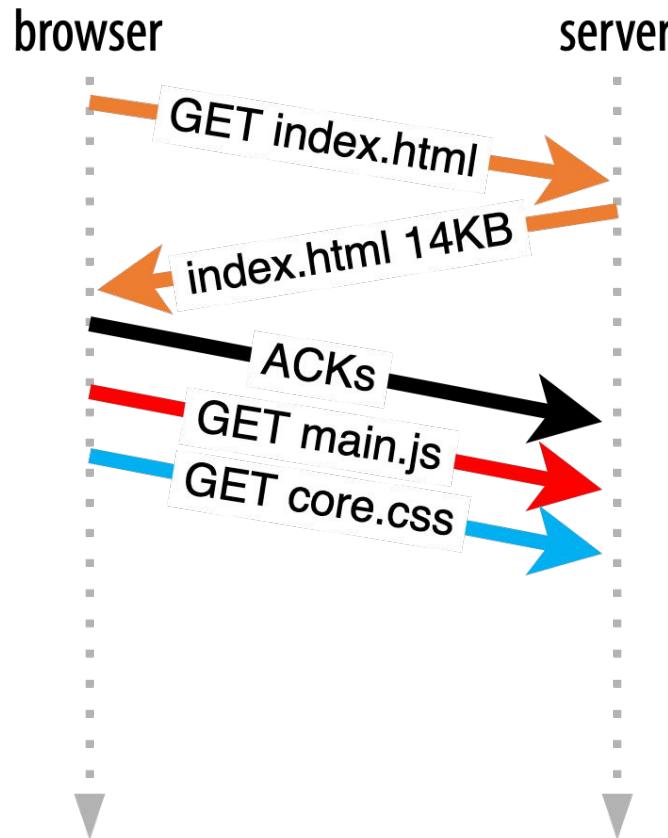


How to fill 20 packets / 28 KB?

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

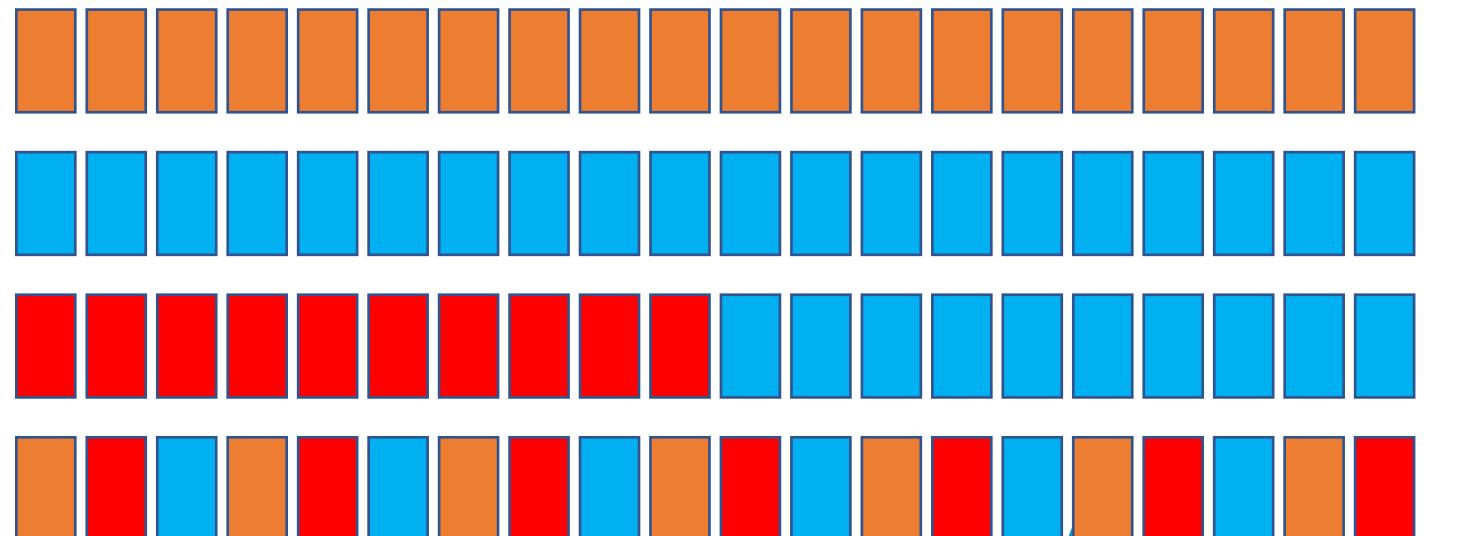


What to send first?



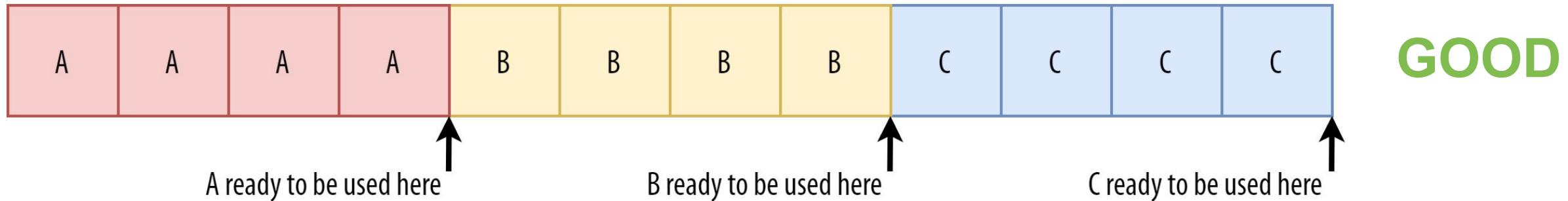
How to fill 20 packets / 28 KB?

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

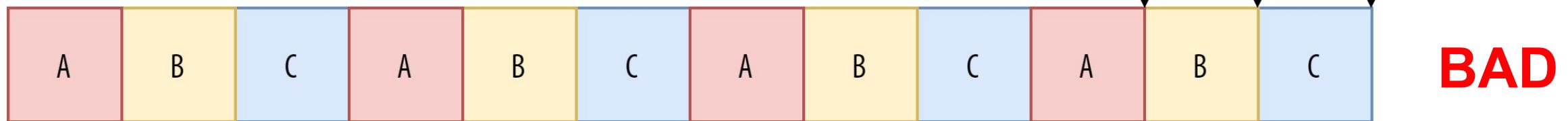


JS and CSS need to be 100% loaded to be used

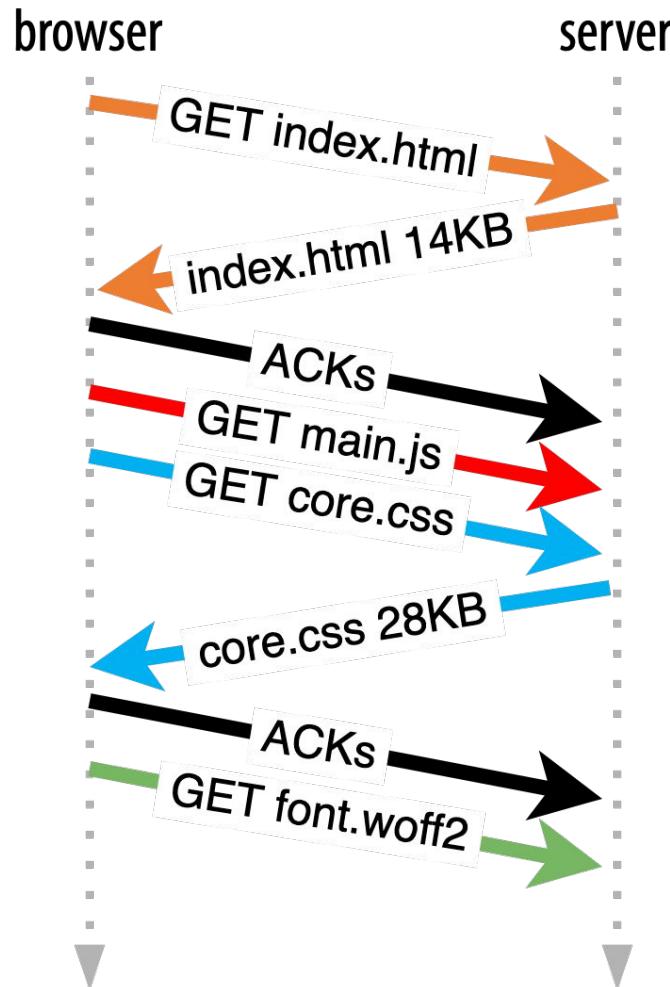
Sequential multiplexer



Round-robin multiplexer



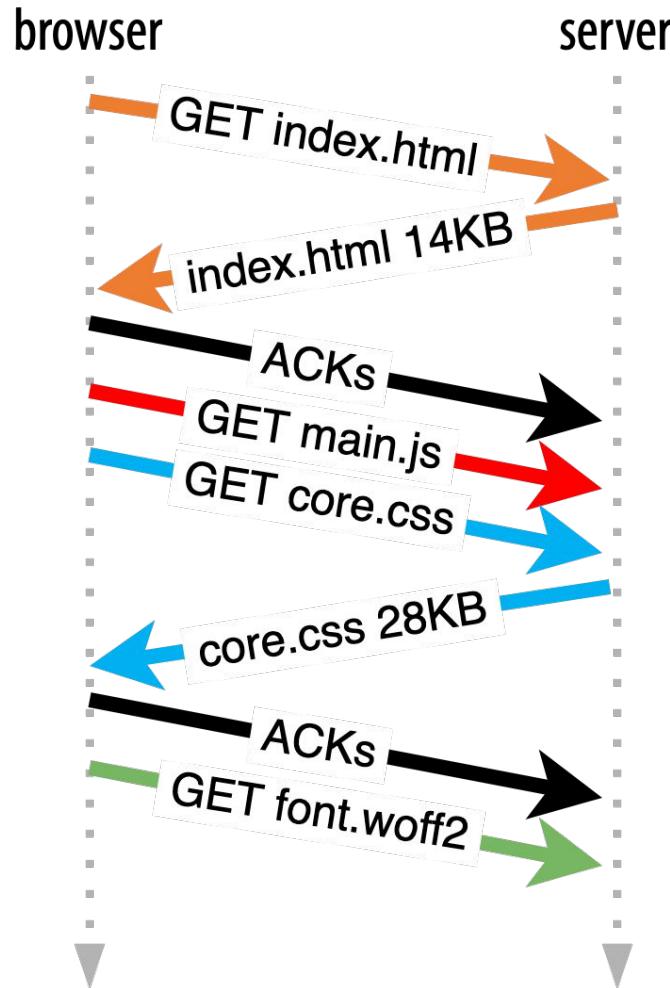
What to send first? Round two!



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!

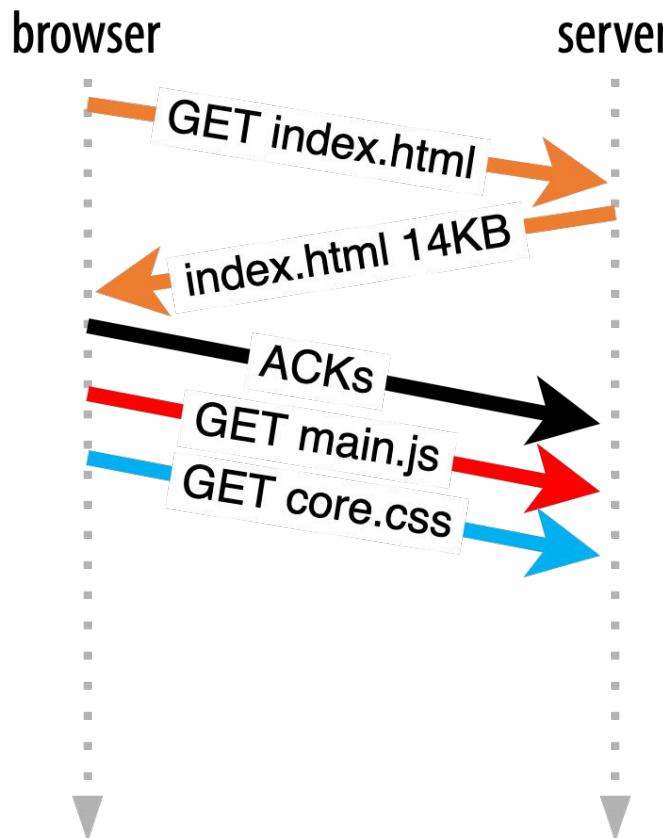


How to fill 40 packets / 56 KB?

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

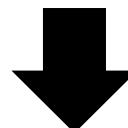
Send order has a big impact on
Web Performance Metrics!

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:



(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

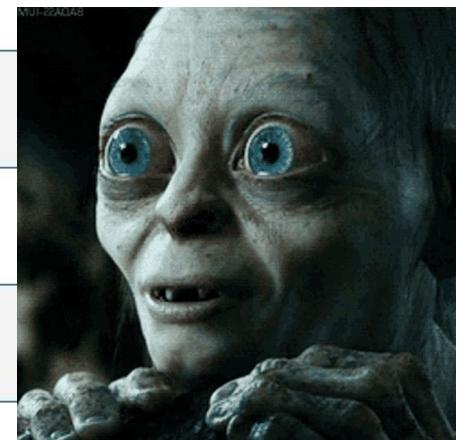
↓ Type / Priority →	Highest	High	Medium	Low	Lowest
Main resource (HTML)	  				
Font (@font-face)					
CSS (head)		 			

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!

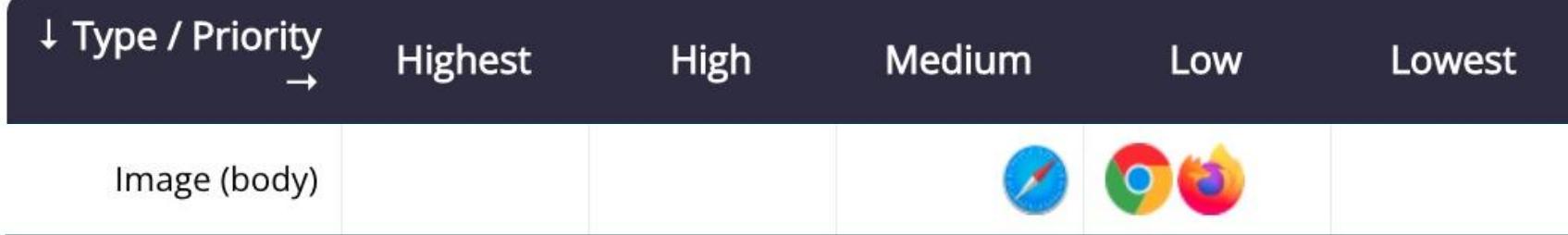
```
<link rel="preload" href="lcp-image.jpg" as="image">
```

(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">
```



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 window. A red arrow points from the top-left towards the 'Develop' menu item in the menu bar. Another red arrow points from the bottom-left towards the 'Priority' column in a table of network resources.

Safari Technology Preview | File Edit View History Bookmarks **Develop** Window Help

Open Page With
User Agent
barrypollard-macbookpro
Service Workers
Web Extension Background Content
Experimental Features
Enter Responsive Design Mode
Show Snippet Editor
Hide Web Inspector
Show JavaScript Console
Show Page Source
Show Page Resources
Start Timeline Recording
Start Element Selection
Empty Caches
Enable Intelligent Tracking Prevention Debug Mode
Disable Images
Disable Styles
Disable JavaScript
Disable Extensions
Disable Site-specific Hacks
Disable Local File Restrictions
Disable Cross-Origin Restrictions
WebRTC

✓ Declarative Shadow DOM
✓ Default ARIA for Custom Elements
✓ Defer async scripts until DOMContentLoaded
✓ Deprecate RSAES-PKCS1-v1_5 Web Crypt...
Deprecation Reporting
Disable Full 3rd-Party Cookie Blocking (ITP)
Disable Removal of Non-Cookie Data After...
✓ Disallow sync XHR during page dismissal
Enable background-fetch API
✓ Experimental MediaSession coordinator API
✓ Experimental MediaSession playlist API
FTP support enabled
✓ Fetch Metadata
✓ **Fetch Priority**
✓ File System Access API
✓ Form requestSubmit
✓ Form-associated custom elements
✓ GPU Process: Canvas Rendering
GPU Process: DOM Rendering
✓ GPU Process: Screen and Window capture
Gamepad trigger vibration support
✓ Gamepad.vibrationActuator support
✓ HTML <dialog> element
HTML <model> element
HTML <model> elements for stand-alone c...
✓ HTML inert attribute
HTML popover attribute
Highlight API
ITP Debug Mode

Brought to you by the Chrome DevRel team

By Chrome DevRel

Elements Console Sources Network

Name	Domain	Type	Priority
zrBPJq27O4Hs8haszVnK.svg	web-dev.imgur.n...	svg	High
9WSNd3mdbXACF19ELKJ1.png	web-dev.imgur.n...	webp	High
CZo4R87iOBYiRpIq6NcP.jpg	web-dev.imgur.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://twitter.com/tunetheweb/status/1653879920026693633>

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

Watch out! Browsers also Artificially Delay Resources



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

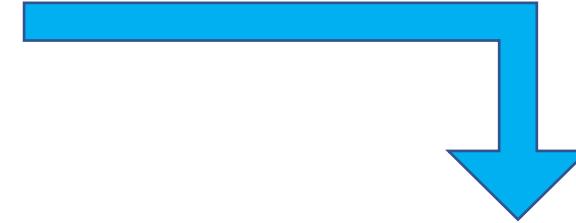
*Just looking at Chrome / CWV is **not enough***

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

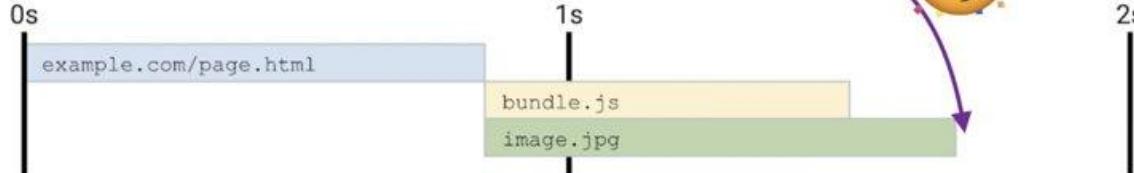


Image request is initiated alongside render-blocking requests

Without preload



Image request is deferred until after render-blocking requests

~~With preload~~

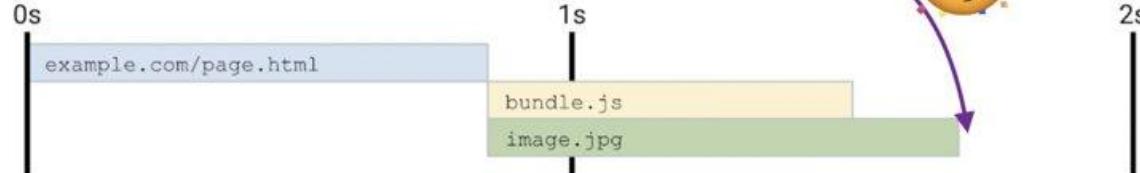


Image request is initiated alongside render-blocking requests

40 packets
~56 KB

Where will you fit
the image data?

Without preload



Image request is deferred until after render-blocking requests

With preload

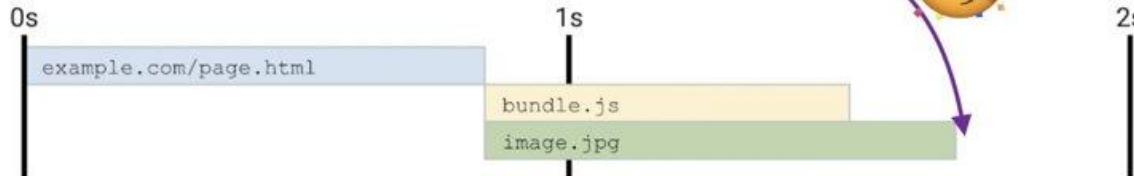
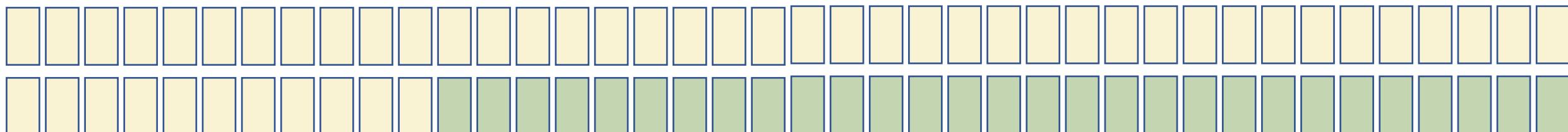


Image request is initiated alongside render-blocking requests

40 packets
~56 KB

Where will you fit
the image data?

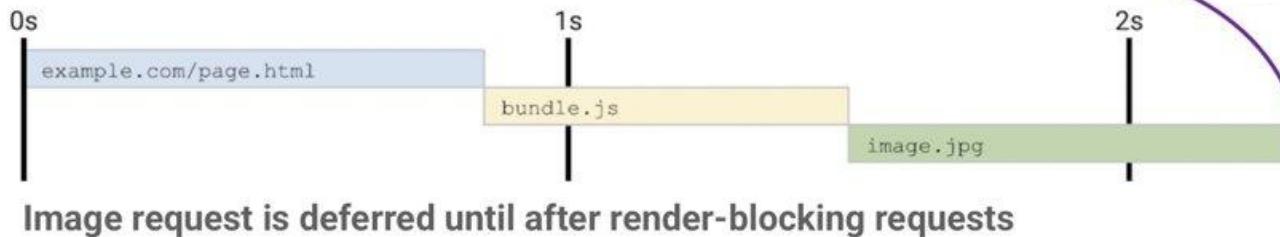


Bad prioritization
TERRIBLE
prioritization

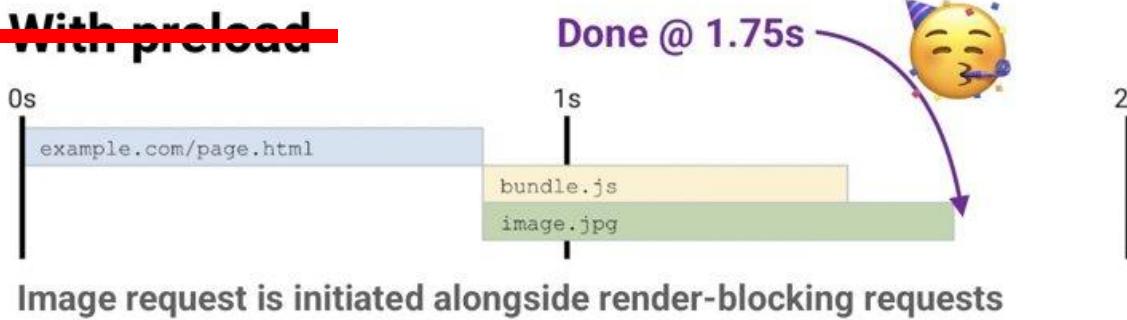


GOOD
prioritization

~~Without preload~~

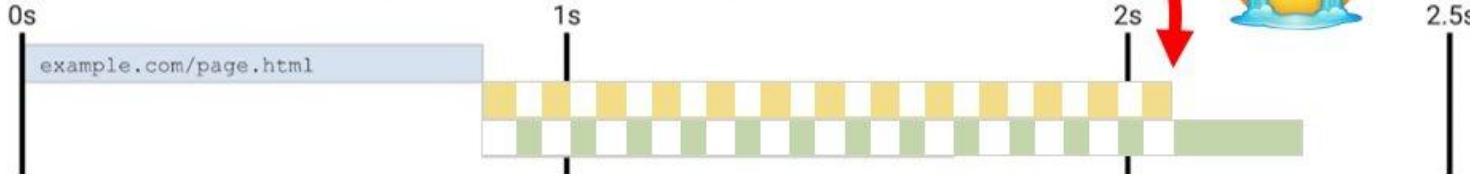


~~With preload~~

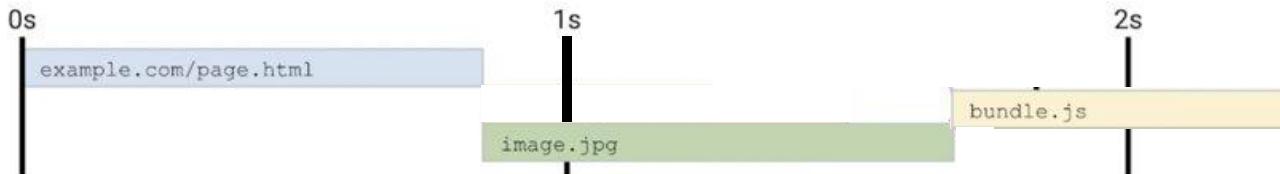


Bad
prioritization

Bundle.js was delayed!!!



TERRIBLE
prioritization



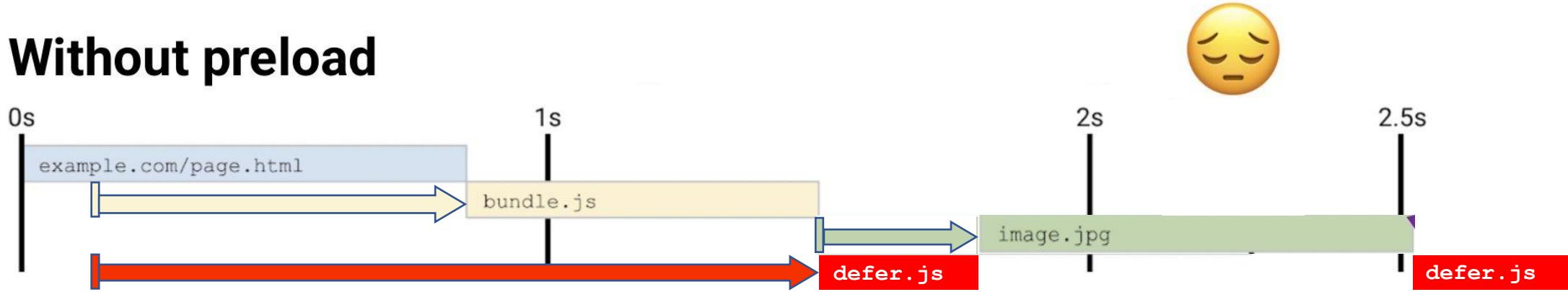
Preload only helps in certain situations

Without preload



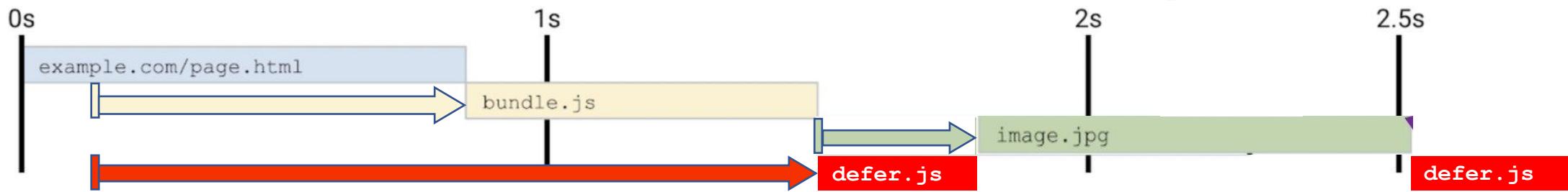
Preload only helps in certain situations

Without preload

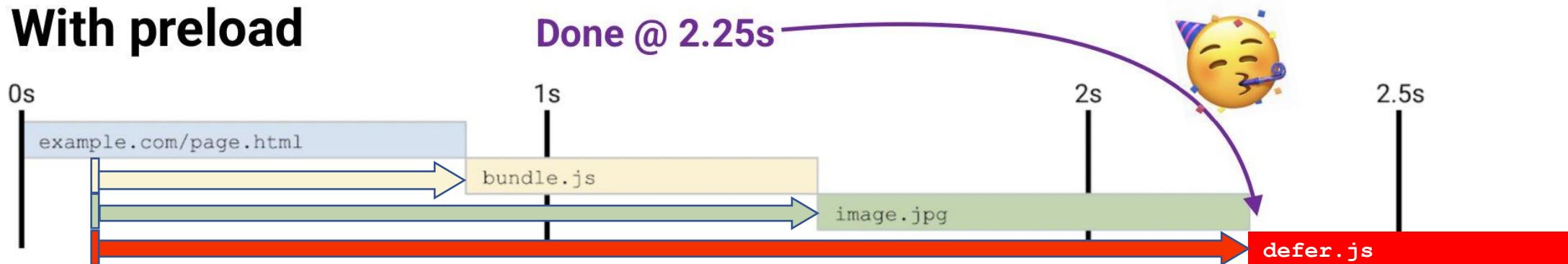


Preload only helps in certain situations

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 pipeline. If these resources are never actually used by the page, that means potentially critical requests will be delayed, slowing down the overall page load.

- /css/chunk-0112032d.cc09ddcf.css
- /css/chunk-0150f84c.6cbbfa5a.css
- /js/chunk-22accc54.21481e62.js
- /css/chunk-0222f9ab.69690fd5.css

[Expand All](#)





#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

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

WOMEN WITH BEARDS

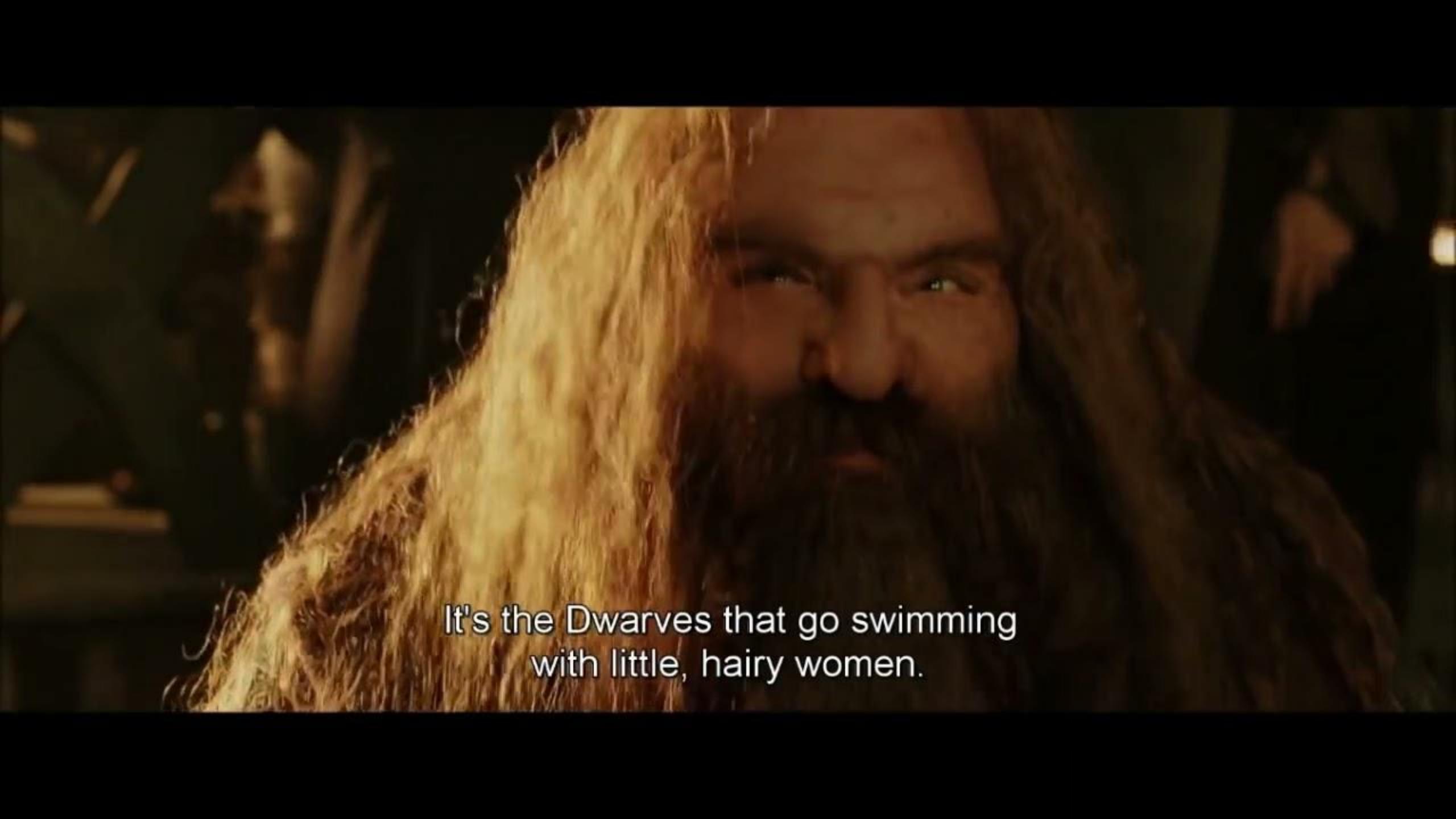


It's true you don't see many dwarf women.



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

It's the beards.

A close-up shot of a dwarf's face. He has long, light-colored hair that is slightly messy, a full dark beard, and a mustache. His eyes are a pale green color. He is looking directly at the camera with a neutral expression. The lighting is warm and focused on his face, while the background is dark and out of focus.

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



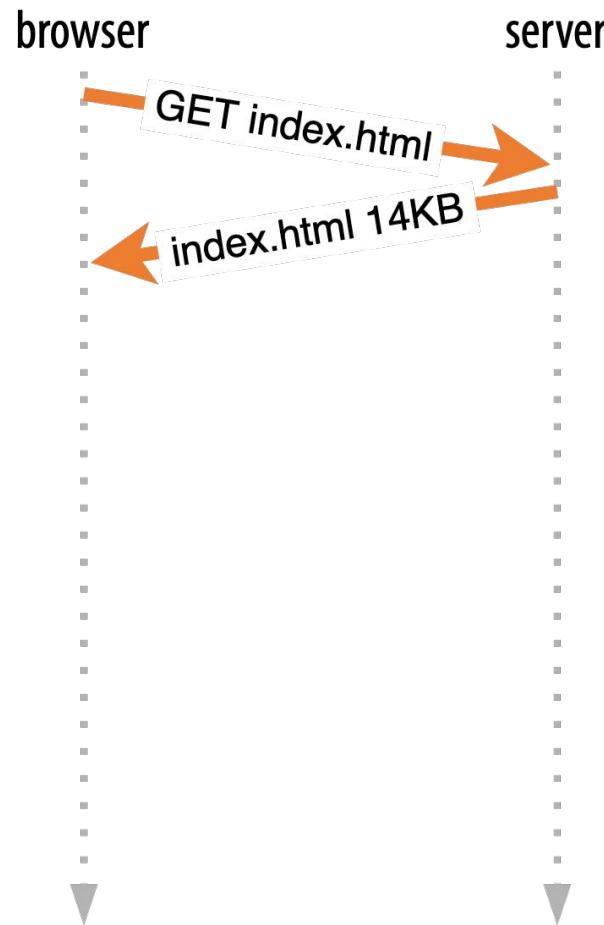




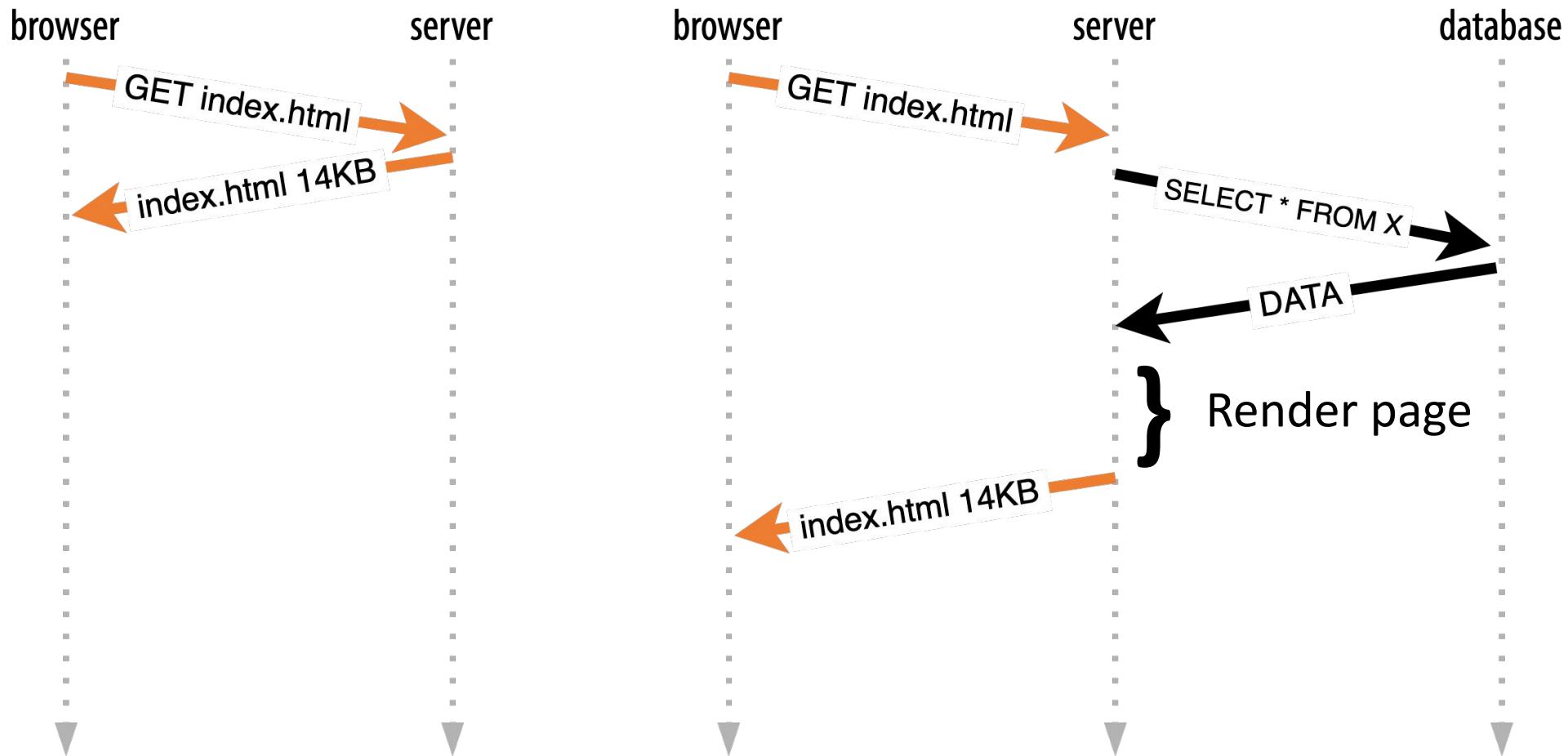


THE RETURN OF SERVER PUSH

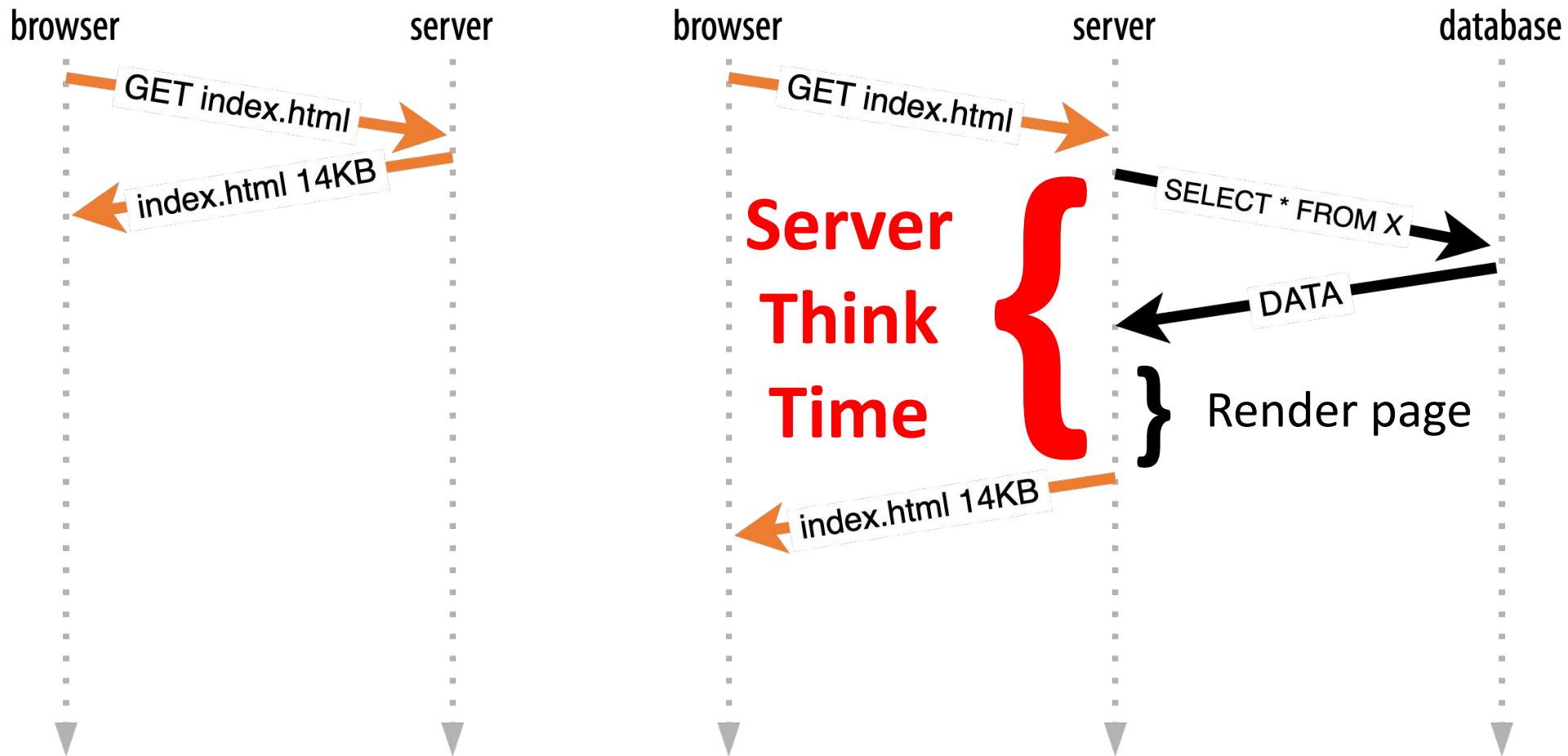
Server Think Time



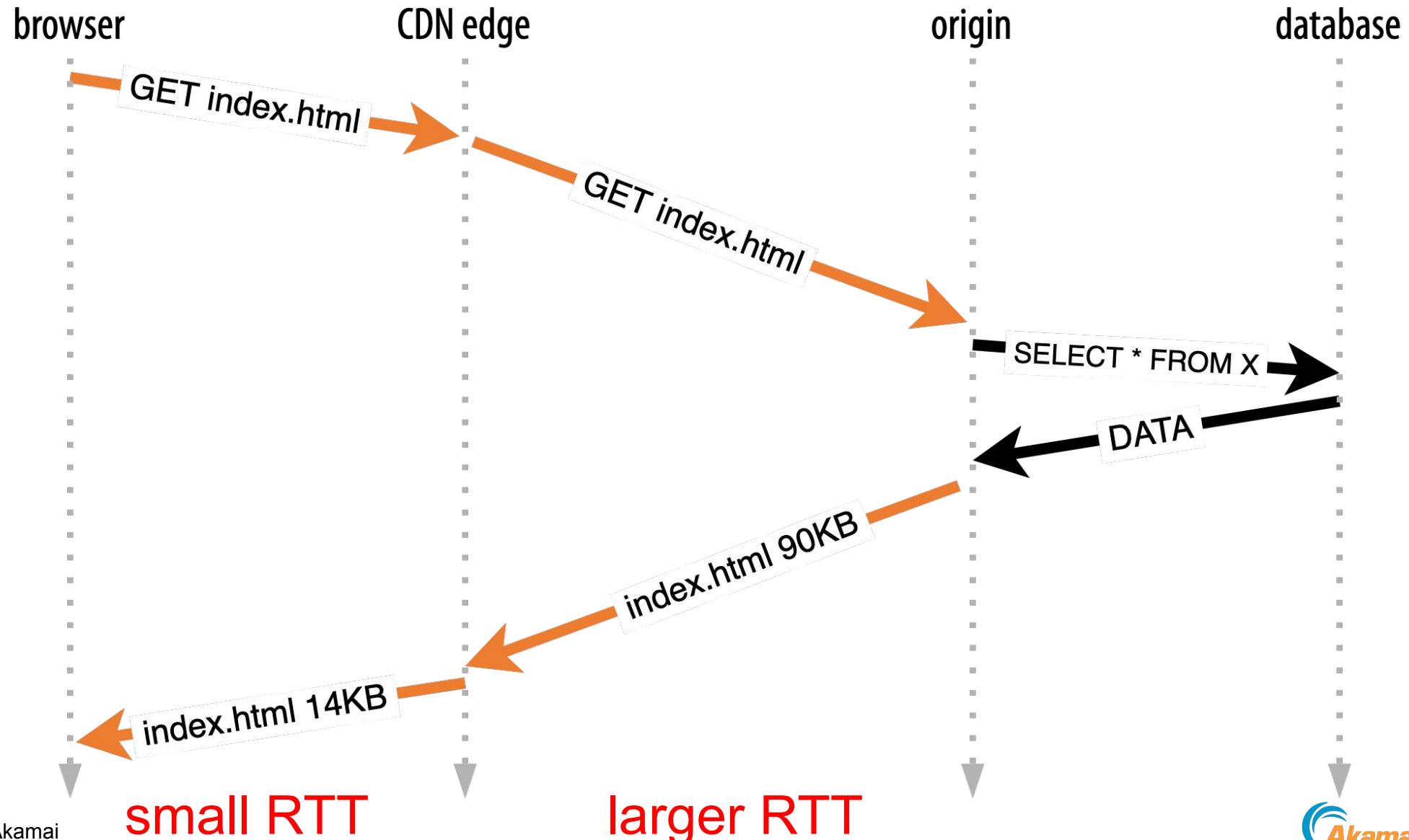
Server Think Time



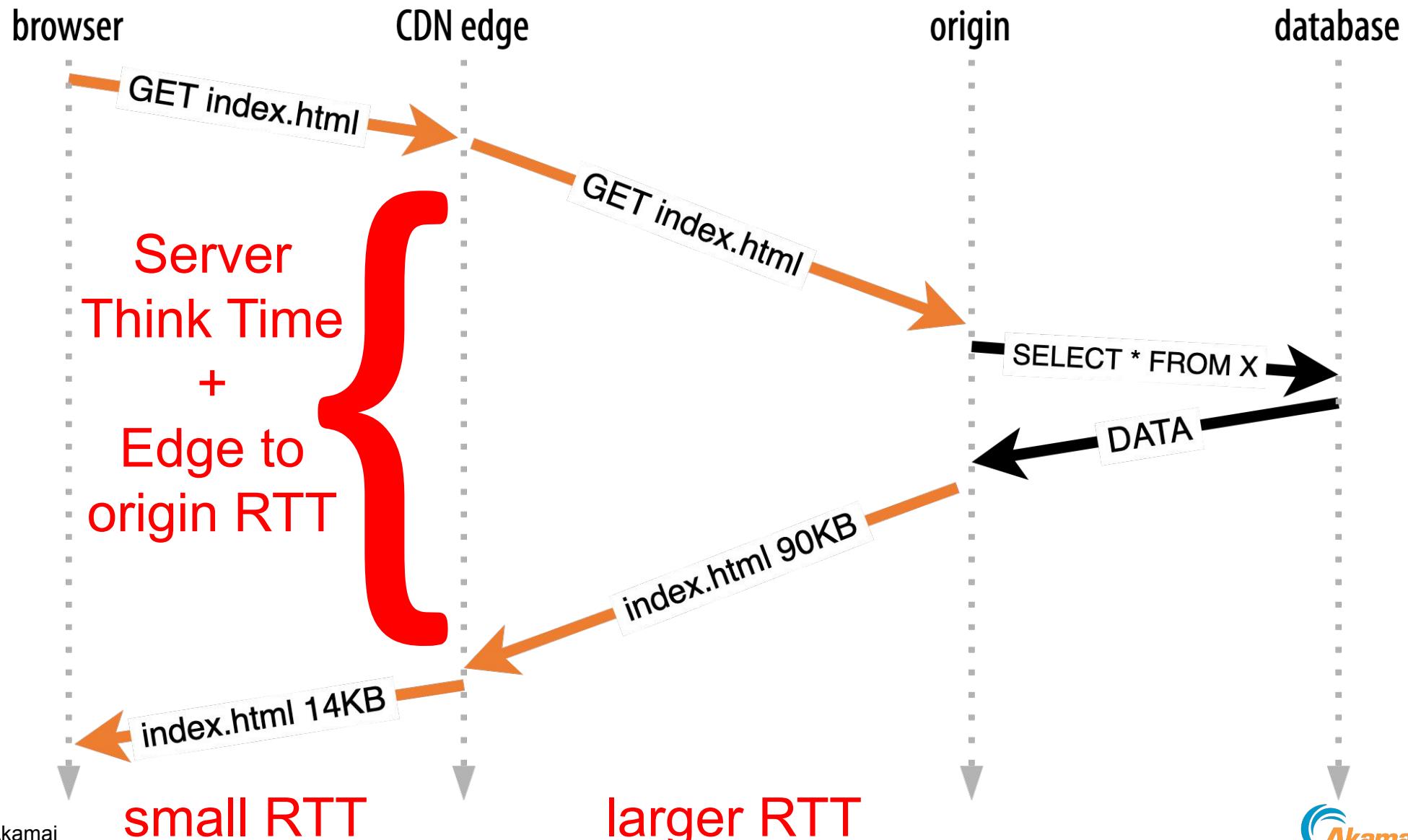
Server Think Time



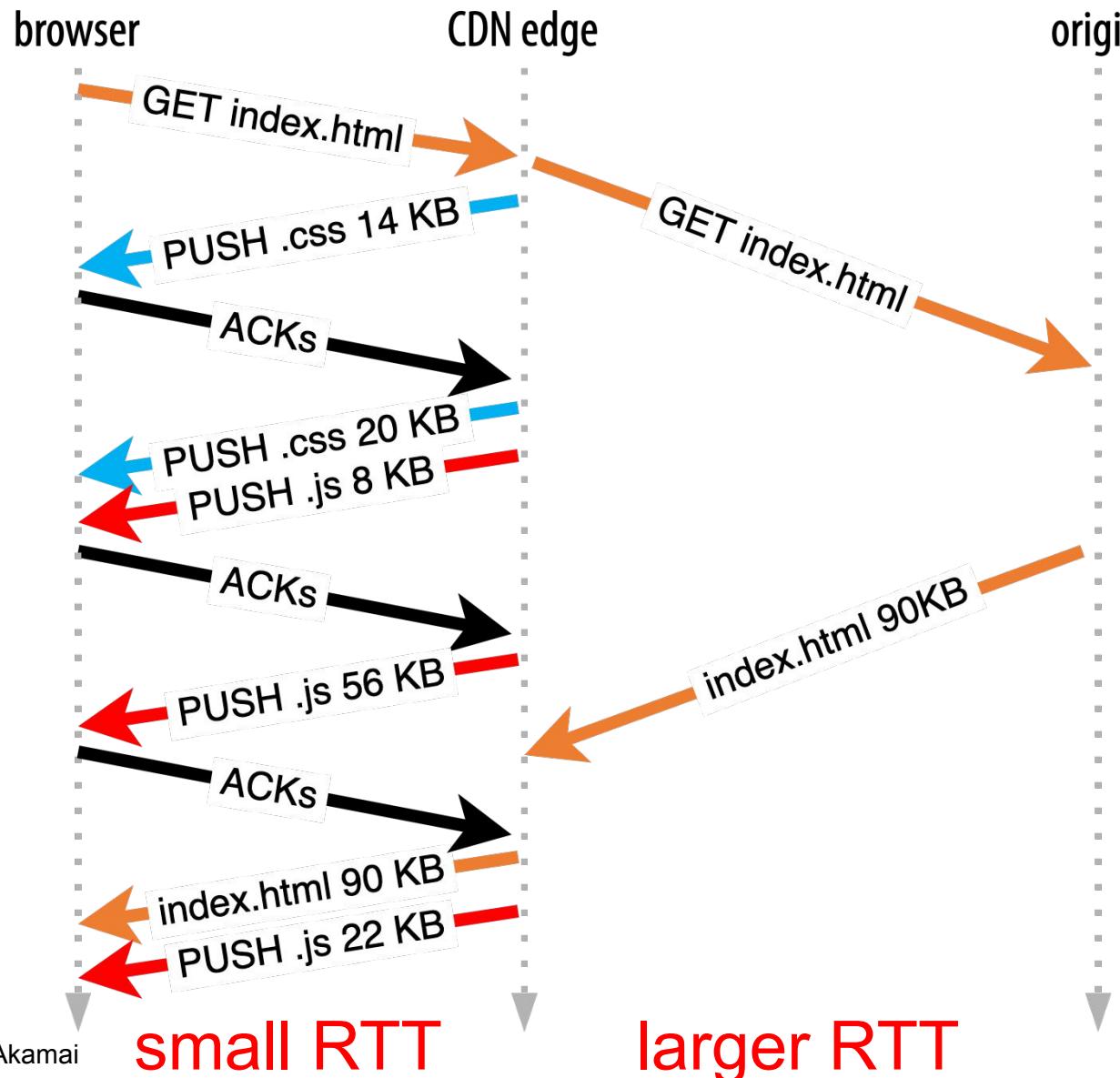
CDN: Origin Fetch Delay



CDN: Origin Fetch Delay



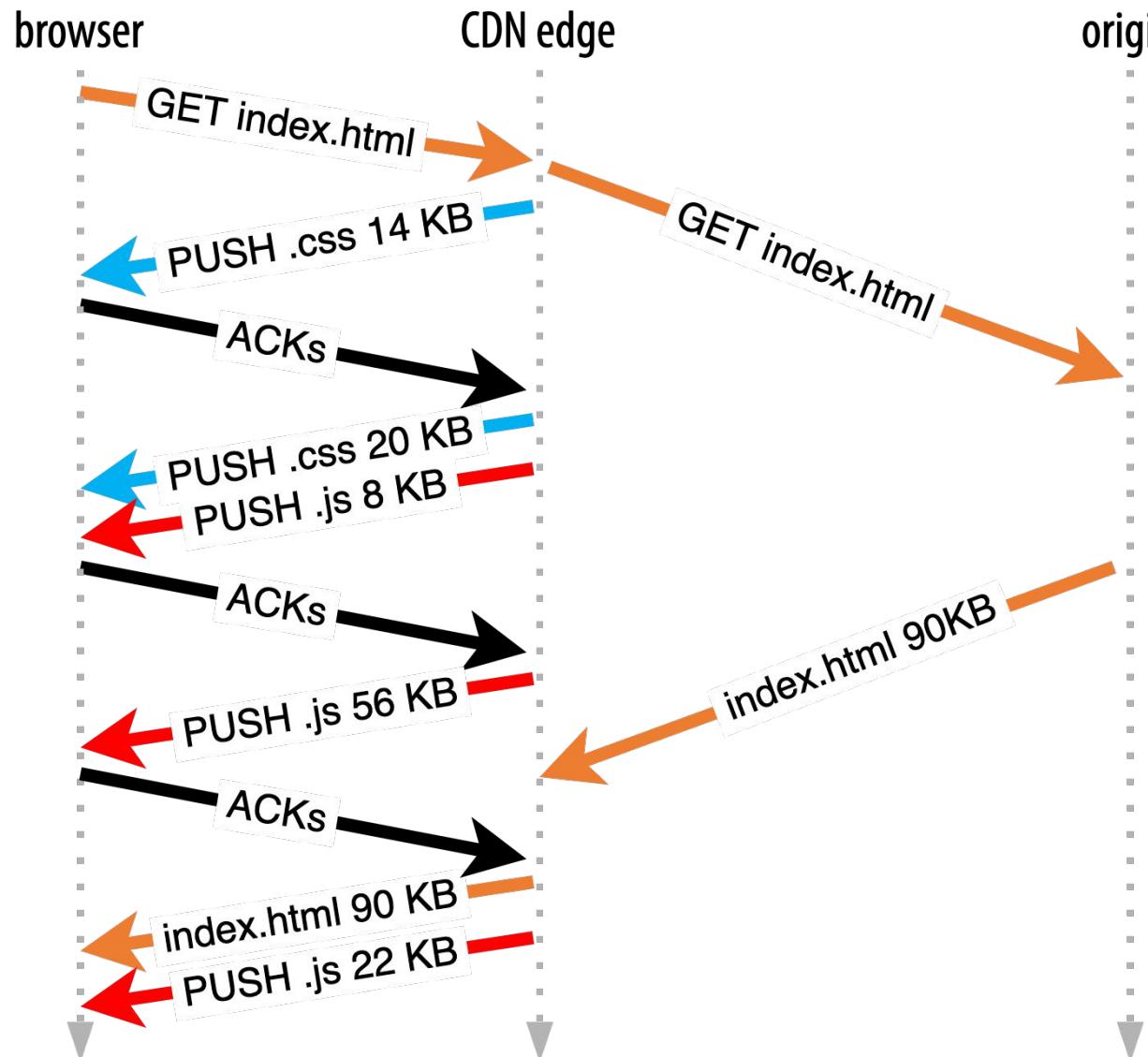
The Promise of Push



Server PUSHes data during wait

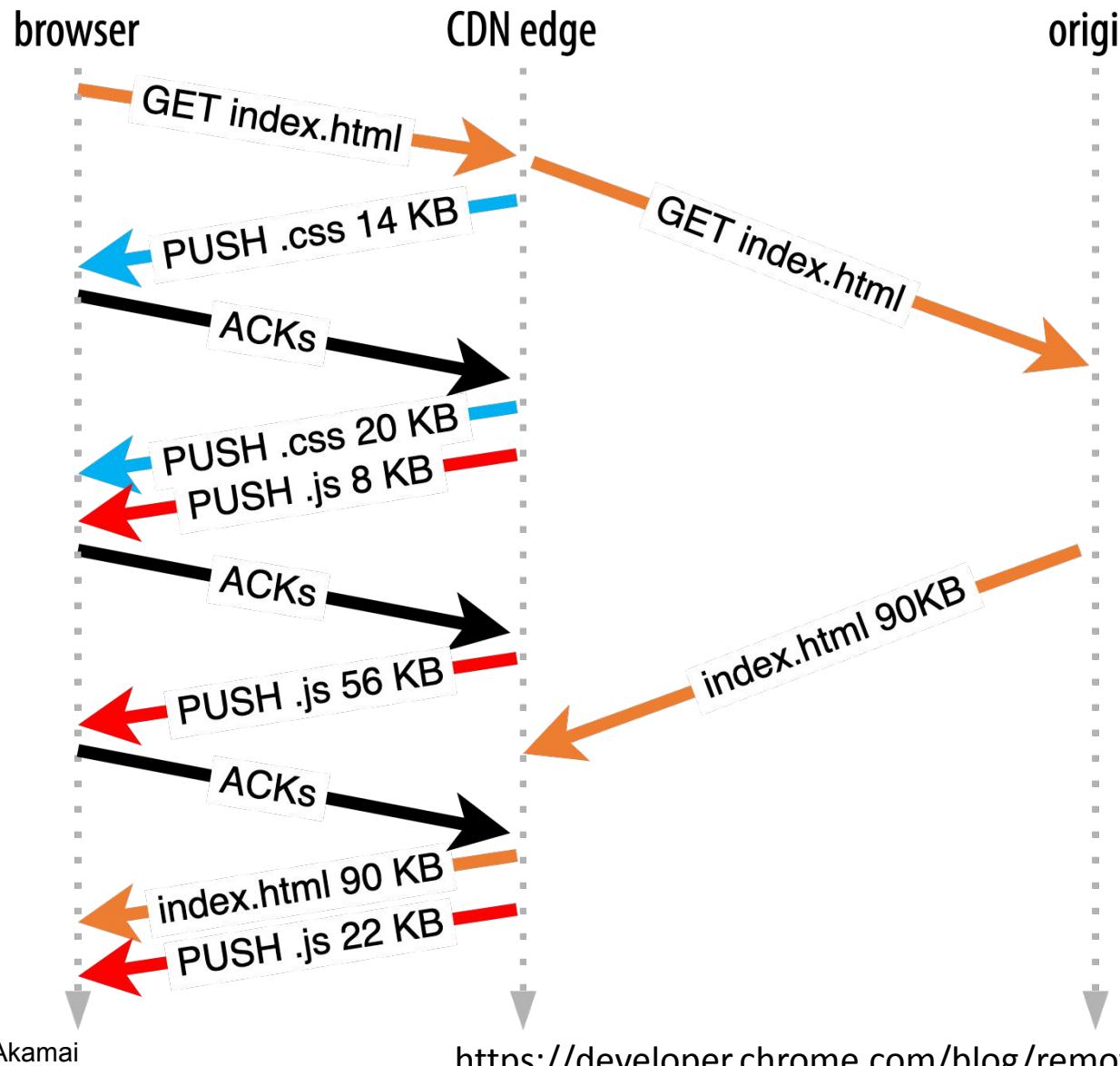
critical .CSS and .JS

The Problems of Push



1. PUSH cached data
2. **PUSH only 1st party data**
3. Many implementation bugs/inconsistencies

The Decay of Push



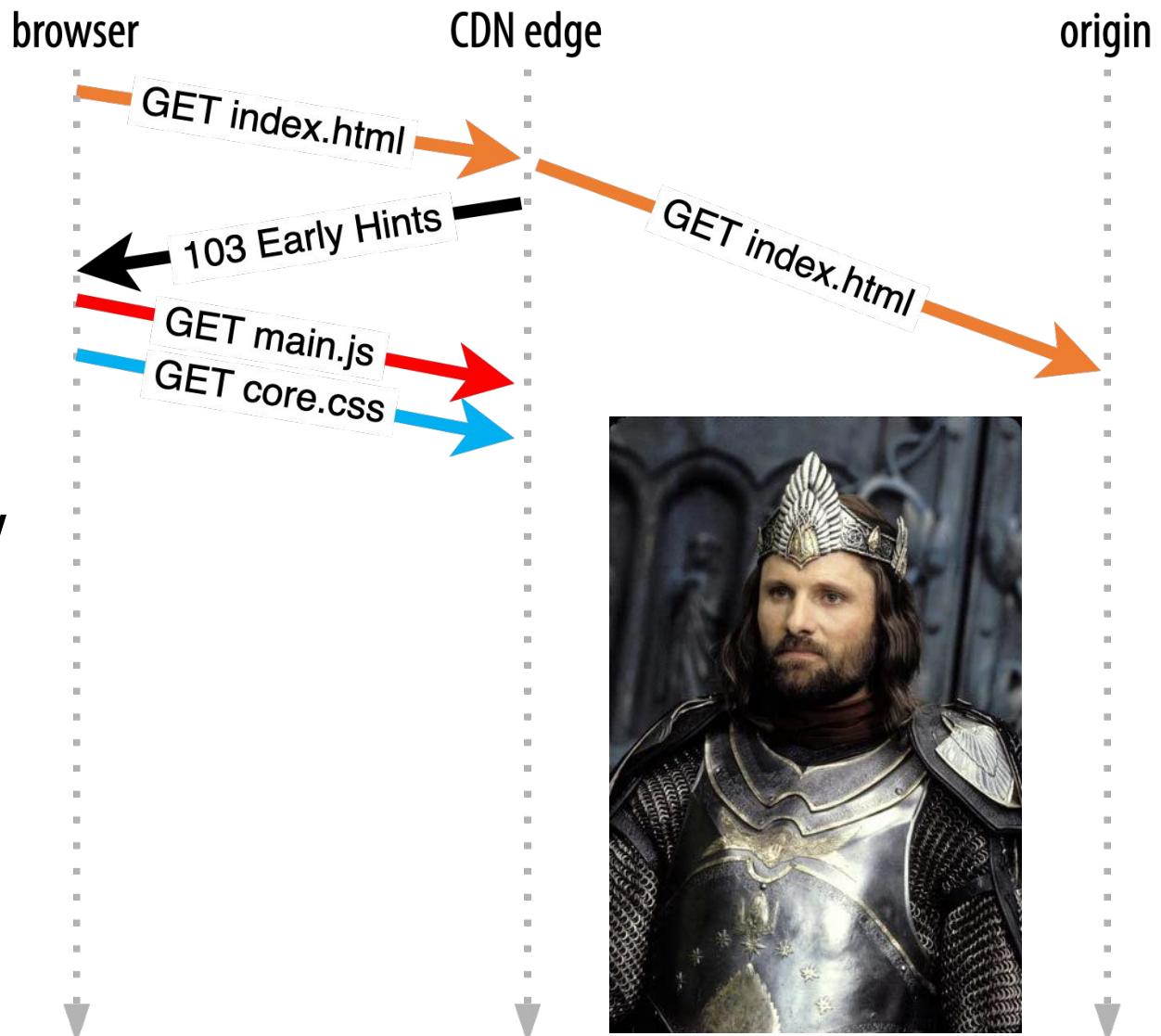
1. Deprecated in Chrome for HTTP/2
2. Never implemented for HTTP/3



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">
```

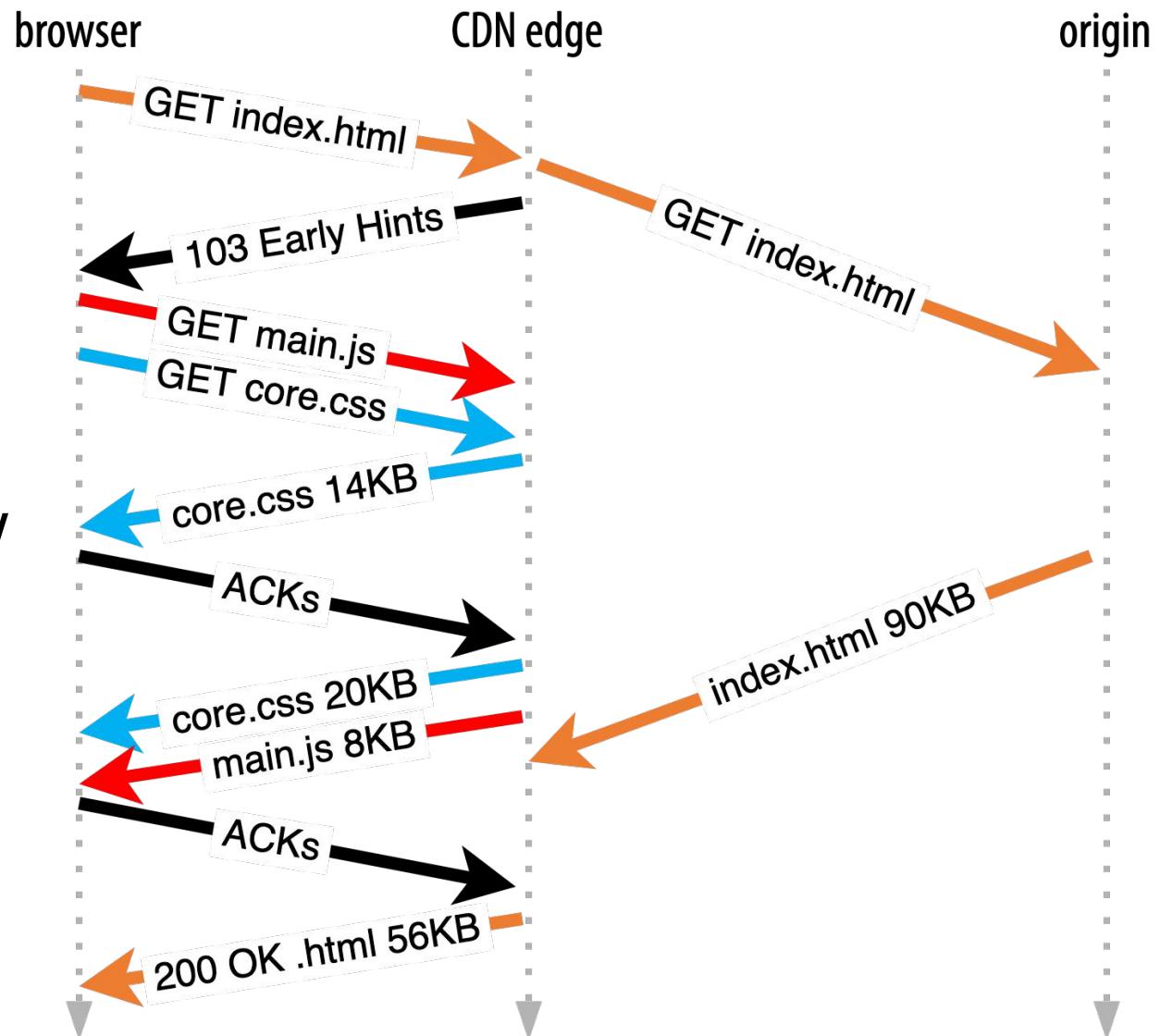
In 103 Early Hints **HTTP response headers**:

```
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

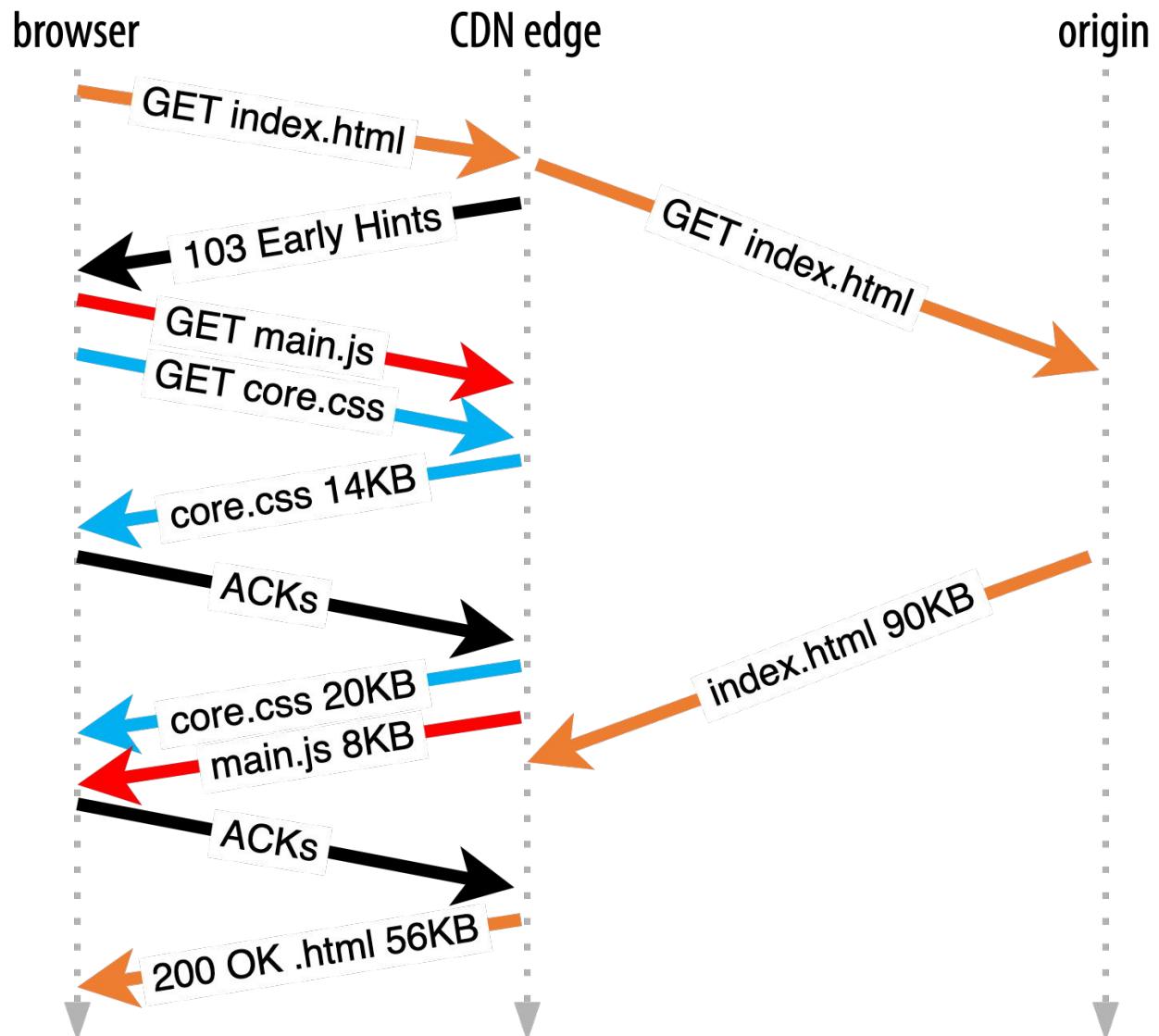
- Don't PUSH data, but send *links to resources*
- **Browser** determines itself what to request and when/how
 - Caching!
 - Prioritization!



The Killer Feature

103 Early Hints

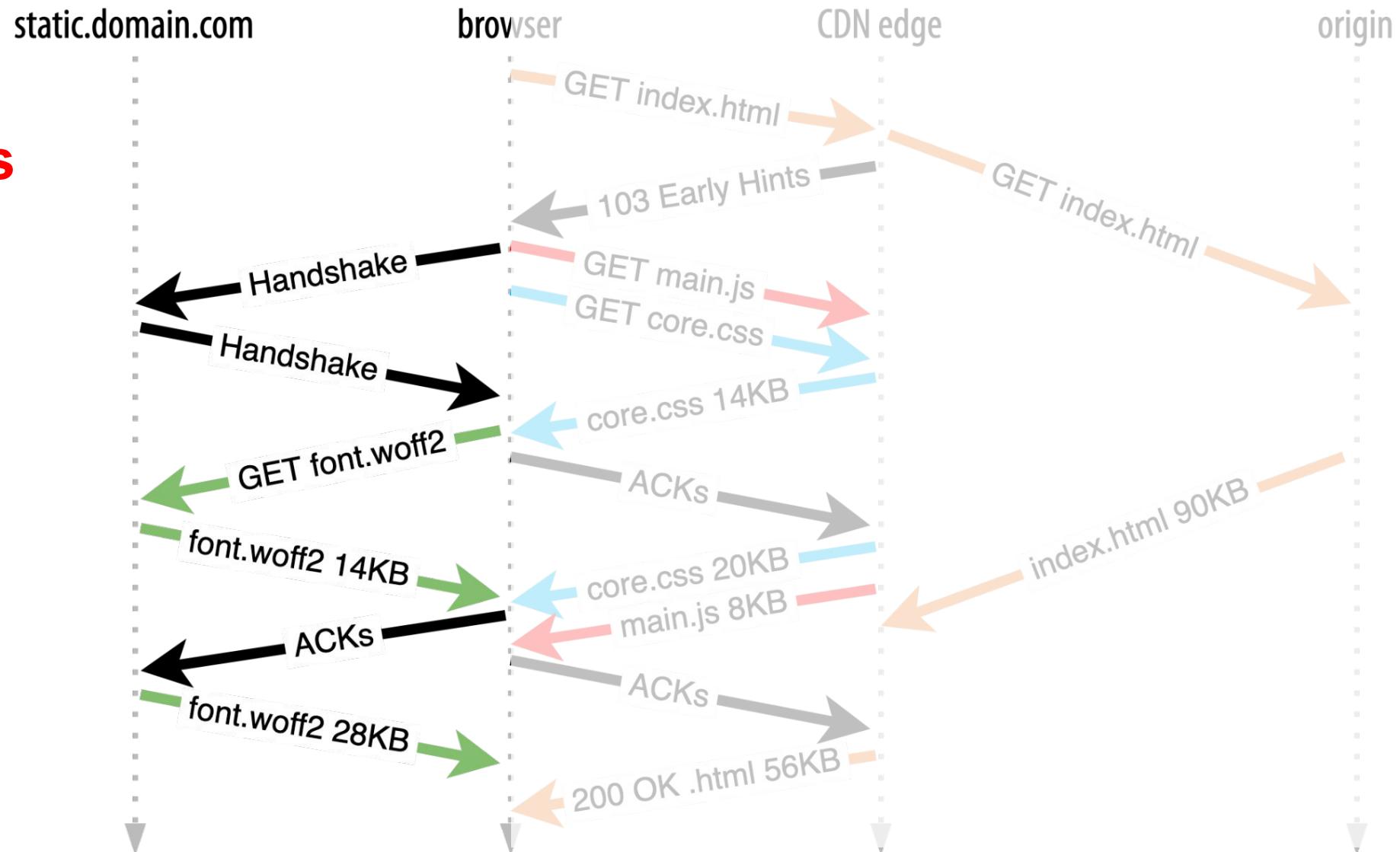
- Supports secondary domains and 3rd party resources!



All that Glitters is Gold

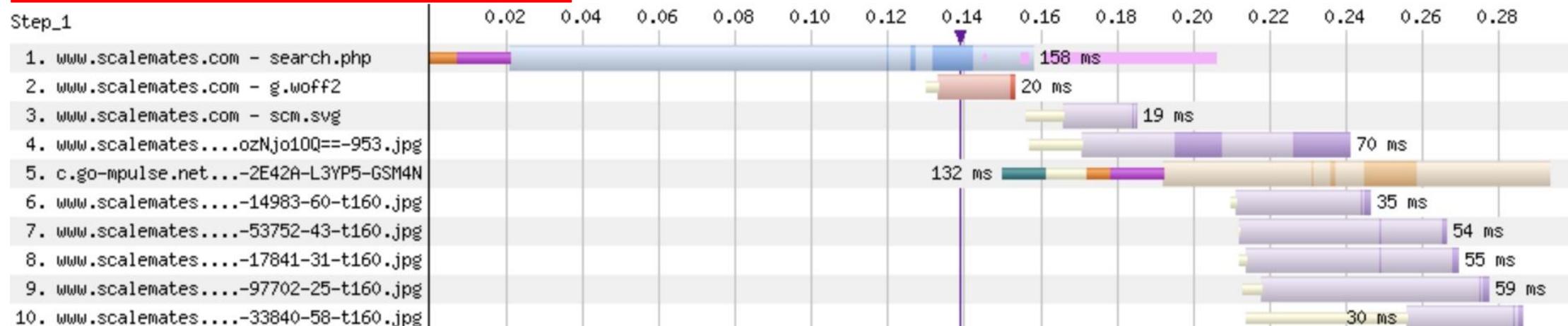
103 Early Hints

- Supports secondary domains and 3rd party resources!



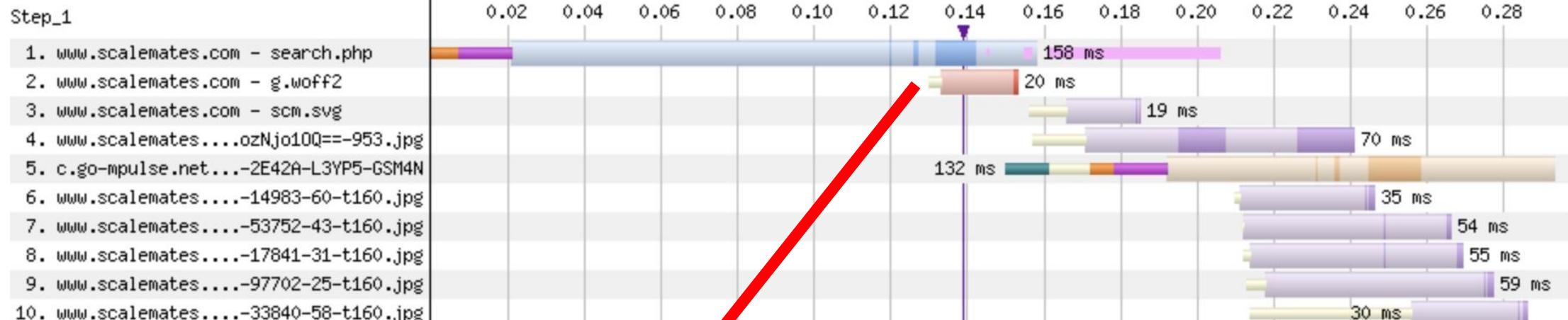
103 Early Hints impact

103 Off

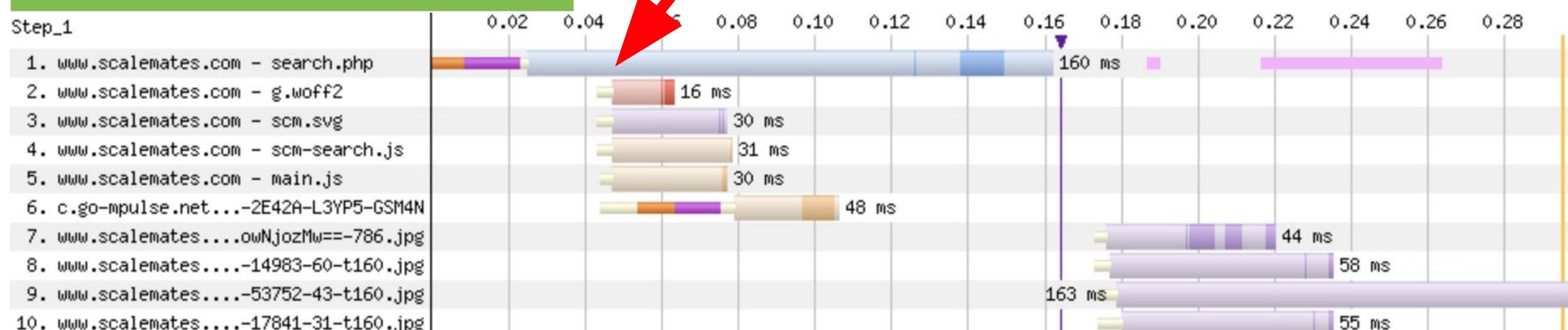


103 Early Hints impact

103 Off



103 On



Preload vs Preload

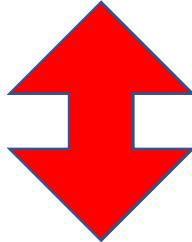
In HTML: preload **late-discovered resources**

```
<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!**

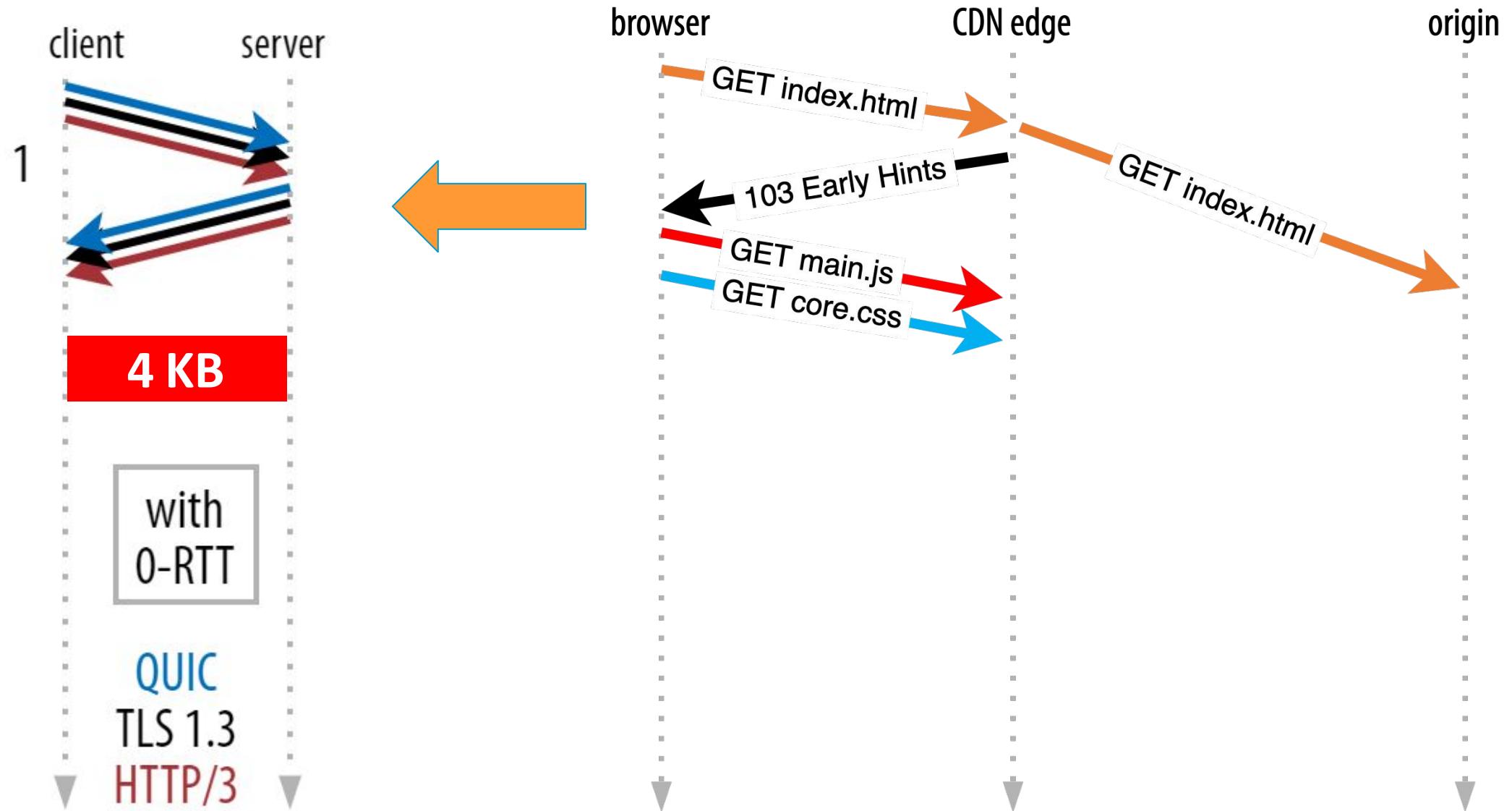
```
Link: <core.css>; rel=preload;  
Link: <main.js>; rel=preload;
```

A close-up, high-contrast photograph of a white dog's face. The dog has a shocked or screaming expression, with its mouth wide open and tongue slightly out. Its eyes are large and have a distinct blue tint. The lighting is dramatic, with strong highlights on the dog's white fur and deep shadows in the background, which appears to be a dark, possibly wooded area.

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

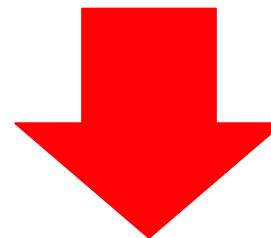
Be careful with protocol-related features



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