The Ultimate Guide to HTTP Resource Prioritization

Robin Marx  @programmingart
A healthy, well-balanced meal
A healthy, well-balanced meal

me

girlfriend
A healthy, well-balanced meal

me

girlfriend

sister
A healthy, well-balanced meal

me

girlfriend

sister

dad
(aka: the lord of chaos)
HTTP/1.1 (TCP)
- index.html
- style.css
- script.js
- image.jpg

HTTP/2 (TCP)
- parallel

HTTP/3 (QUIC)
- multiplexed
Problem 1:

What is the best multiplexing approach?
Sequential

Fair Round-Robin

Unfair Round-Robin

Combinations

HTML
JavaScript
CSS
Images
<head>
    <script src="script.js" /></script>
    <link rel="stylesheet" href="style.css" /></link>
</head>
<body>
    <img src="progressive1.jpg" />
    <img src="progressive2.jpg" />
</body>
Progressive jpeg example

normal (scanline)

progressive

https://tobias.is/blogging/even-faster-images-using-http2-and-progressive-jpegs/
<html><head>
  <script src="script.js" />
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <img src="progressive1.jpg" />
  <img src="progressive2.jpg" />
</body></html>

Best if progressive (~25%)
Best if not (~75%)

(~70 million jpegs in HTTP archive) https://twitter.com/programmingart/status/1222545046651785216?s=20
```html
<html>
<head>
  <script src="script.js" />
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <img src="progressive1.jpg" />
  <img src="progressive2.jpg" />
</body>
</html>
```
<head>
  <script src="script.js" />
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <img src="progressive1.jpg" />
  <img src="progressive2.jpg" />
  <script src="later.js" async />
</body>
<head>
  <script src="script.js" />
  <link rel="stylesheet" href="style.css" />
</head>

<body>
  <img src="progressive1.jpg" />
  <img src="progressive2.jpg" />
  <script src="later.js" async />
</body>

fetch("data.json")

Render blocking

RE-PRIORITIZATION
Browser doesn’t know

1. Size of resource
2. If the resource can be used progressively
3. What the resource will actually do
4. If the resource references other resources
5. The “critical path”
Browser doesn’t know

1. Size of resource
2. If the resource can be used progressively
3. What the resource will actually do
4. If the resource references other resources
5. The “critical path”

So it has to guess

1. Mime-type
2. Position in the document
3. How it hopes developers use things like async, defer, preload, ...
**Browser heuristics**

<table>
<thead>
<tr>
<th>highest</th>
<th>lowest</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML, CSS, <strong>fonts</strong></td>
<td>HTML</td>
</tr>
<tr>
<td>JS before img 1, <strong>fetch</strong></td>
<td>JS, CSS</td>
</tr>
<tr>
<td>visible img</td>
<td><strong>fonts, fetch</strong></td>
</tr>
<tr>
<td>JS after img 1</td>
<td>img</td>
</tr>
<tr>
<td>invisible img, async, defer</td>
<td><strong>fetch</strong>, &lt;body&gt; JS</td>
</tr>
</tbody>
</table>

| HTML, CSS, **fonts**         | HTML                          |
| JS before img 1, **fetch**   | JS, CSS                       |
| visible img                  | **fonts, fetch**              |
| JS after img 1               | img                           |
| invisible img, async, defer  | **fetch**, <body> JS          |
Which one is best?

<table>
<thead>
<tr>
<th>Edge</th>
<th>Safari</th>
<th>Firefox</th>
<th>Chrome</th>
</tr>
</thead>
</table>

waiting...

https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web
Browser heuristics

<table>
<thead>
<tr>
<th>highest</th>
<th>lowest</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML, CSS, <strong>fonts</strong></td>
<td>invisible img, async, defer</td>
</tr>
<tr>
<td>JS before img 1, <strong>fetch</strong></td>
<td>JS after img 1</td>
</tr>
<tr>
<td>visible img</td>
<td>img</td>
</tr>
<tr>
<td>JS, CSS</td>
<td><strong>fonts</strong>, <strong>fetch</strong></td>
</tr>
<tr>
<td>CSS, &lt;head&gt; JS</td>
<td><strong>fetch</strong>, &lt;body&gt; JS</td>
</tr>
</tbody>
</table>

sequential

naïve unfair RR

complex unfair RR

fair RR (H2 default)
Round-Robin is bad!
Round-Robin is bad!? 

https://v8.dev/blog/v8-release-78
https://medium.com/reloading/javascript-start-up-performance-69200f43b201
Is Round-Robin bad?
Heuristics = on average

N = 40

FASTER

SLOWER

Fair Round-Robin

Everything else

https://h3.edm.uhasselt.be/
https://speeder.edm.uhasselt.be/www18
Heuristics = on average

FASTER
N = 40

SLOWER

Average speedup factor for Above-the-fold resources

BETTER

https://h3.edm.uhasselt.be/
https://speeder.edm.uhasselt.be/www18
Heuristics = on average

Sequential for CSS/Script
+ Complex Weighted RR for rest

Priority (0-63)

50 (critical)
- Concurrency: 0
  - critical CSS *
  - critical script *
  - other critical *

30 (normal)
- Concurrency: 0
  - CSS
  - script
- Concurrency: 1
  - HTML
  - preload font *
  - async script *
- Concurrency: n
  - image
  - video
  - other

10 (idle)
- Concurrency: 0
  - prefetch *
- Concurrency: 1
  -
- Concurrency: n

* If Detectable

https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web/
Heuristics = on average

50% faster by default, particularly for Edge and Safari is not unusual

https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web/
The prioritizing scheduler beat the random scheduler on only 31% of pages tested.

Maximum benefit was 3.1%, even compared to LIFO.

Chrome’s approach is better than fair RR, but only up to 2.69%.

2016

2019
NOBODY KNOWS
How can you find out if you have a problem?

Test your pages
webpagetest.org
Some (imperfect) client-side options

1. Async, Defer
2. Preload

But: 1. Bugs! (too aggressive)
2. Browser support 🌐Mozilla Firefox

https://wicg.github.io/priority-hints/
https://web.dev/native-lazy-loading/
https://bugzilla.mozilla.org/show_bug.cgi?id=1405761
https://twitter.com/domfarolino/status/1221803122638508032?s=20
https://andydavies.me/blog/2019/02/12/preloading-fonts-and-the-puzzle-of-priorities/
Some (imperfect) client-side options

1. Async, Defer
2. Preload

But:
1. Bugs! (too aggressive)
2. Browser support [Google, Firefox]

3. Priority hints

But:
1. Possibly not fine-grained enough
2. Browser support [Google]
Server-side overrides

https://www.shimmercat.com/blog/coordinated-image-loading.html
https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web/
https://h2o.example.net/configure/http2_directives.html#http2-reprioritize-blocking-assets
Problem 2:

How to communicate this to the server?
Please serve resources in this order
Please serve resources in this order:

8 PRIORITY LEVELS

<table>
<thead>
<tr>
<th>Highest</th>
<th>Possible mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HTML</td>
</tr>
<tr>
<td>1</td>
<td>CSS</td>
</tr>
<tr>
<td>2</td>
<td>JavaScript</td>
</tr>
<tr>
<td>3</td>
<td>fonts</td>
</tr>
<tr>
<td>4</td>
<td>fetch</td>
</tr>
<tr>
<td>5</td>
<td>images</td>
</tr>
<tr>
<td>6</td>
<td>async and defer JS</td>
</tr>
<tr>
<td>7</td>
<td>video</td>
</tr>
</tbody>
</table>

Origin
fair

CDN

origin

https://lists.w3.org/Archives/Public/ietf-http-wg/2013JanMar/0554.html
https://lists.w3.org/Archives/Public/ietf-http-wg/2013JanMar/0560.html
https://lists.w3.org/Archives/Public/ietf-http-wg/2019AprJun/0113.html
“everything is of highest priority”

https://lists.w3.org/Archives/Public/ietf-http-wg/2013JanMar/0554.html
https://lists.w3.org/Archives/Public/ietf-http-wg/2013JanMar/0560.html
https://lists.w3.org/Archives/Public/ietf-http-wg/2019AprJun/0113.html
“everything is of highest priority”

CDN

fair but useless

0 0 0 0 0 0 0

“everything is of highest priority”

origin

https://lists.w3.org/Archives/Public/ietf-http-wg/2013JanMar/0554.html
https://lists.w3.org/Archives/Public/ietf-http-wg/2013JanMar/0560.html
https://lists.w3.org/Archives/Public/ietf-http-wg/2019AprJun/0113.html
HTTP/2: The Dependency Tree Awakens

HTML

CSS

200

100

Hero Image

JavaScript

2:1 RR ratio

Please serve resources in this order

GET this file and add as child of CSS, with weight 100

JavaScript

HTML

CSS

Hero Image

200

origin
BUT: isn’t actually used that way

Well yes, but actually no
BUT: only firefox uses a real tree

No siblings

Only siblings

siblings with "placeholders"

https://speeder.edm.uhasselt.be/www18
BUT: servers implement badly or not at all

https://www.youtube.com/watch?v=ct5MvtmL1NM
https://github.com/andydavies/http2-prioritization-issues
https://www.slideshare.net/patrickmeenan/http2-in-practice
https://discourse.haproxy.org/t/http-2-prioritization/4578/3
https://twitter.com/bagder/status/1222143040577589248?s=20

9 / 34 deployments pass
BUT: difficult to do a server-side override

Find best parent

As sibling with high weight

As sibling under correct placeholder with some weight
Server-overrides ~ = redo the whole thing

Priority (0-63)

50 (critical)
- Concurrency: 0
  - critical CSS *
  - critical script *
  - other critical *

30 (normal)
- Concurrency: 0
  - CSS
  - script
- Concurrency: 1
  - HTML
  - preload font *
  - async script *
- Concurrency: n
  - image
  - video
  - other

10 (idle)
- Concurrency: 0
  - prefetch *
- Concurrency: 1
- Concurrency: n

* If Detectable

not on e, revert to mime-type

Look at browser’s PRIORITY messages
• Guess which browser
• Put resources into fully new server-side scheme

https://twitter.com/programmingart/status/1222550479168733186?s=20
This is fine.
Current HTTP/3 proposal under consideration: Back to SPDY basics

<table>
<thead>
<tr>
<th>urgency</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisite</td>
<td>0</td>
</tr>
<tr>
<td>Default</td>
<td>1</td>
</tr>
<tr>
<td>Supplementary SERVER</td>
<td>2</td>
</tr>
<tr>
<td>Supplementary CLIENT</td>
<td>3</td>
</tr>
<tr>
<td>Supplementary CLIENT</td>
<td>4</td>
</tr>
<tr>
<td>Supplementary CLIENT</td>
<td>5</td>
</tr>
<tr>
<td>Supplementary SERVER</td>
<td>6</td>
</tr>
<tr>
<td>Background</td>
<td>7</td>
</tr>
</tbody>
</table>

Current HTTP/3 proposal under consideration: Back to SPDY basics

- Prerequisite
- Default
- Supplementary SERVER
- Supplementary CLIENT
- Supplementary CLIENT
- Supplementary CLIENT
- Supplementary SERVER
- Background

Easy server-side overrides

Current HTTP/3 proposal under consideration: Back to SPDY basics

**urgency**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Prerequisite</td>
</tr>
<tr>
<td>1</td>
<td>Default</td>
</tr>
<tr>
<td>2</td>
<td>Supplementary SERVER</td>
</tr>
<tr>
<td>3</td>
<td>Supplementary CLIENT</td>
</tr>
<tr>
<td>4</td>
<td>Supplementary CLIENT</td>
</tr>
<tr>
<td>5</td>
<td>Supplementary CLIENT</td>
</tr>
<tr>
<td>6</td>
<td>Supplementary SERVER</td>
</tr>
<tr>
<td>7</td>
<td>Background</td>
</tr>
</tbody>
</table>

**incremental**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sequential</td>
</tr>
<tr>
<td>1</td>
<td>Can be Round-Robined</td>
</tr>
</tbody>
</table>

Easy server-side overrides

Current HTTP/3 proposal under consideration: Back to SPDY basics

Using an HTTP header:
- Easy to debug
- Easy to track
- Easy to reason about

```
fetch("image.jpg", {
  headers: {
    "priority": "u=4, i=?1"
  }
});
```

Many open questions...

How to handle round-robin in practice?

urgency = 3

Incremental = 1 1 0 1 1 1 1

https://github.com/httpwg/http-exensions/issues?q=is%3Aissue+is%3Aopen+label%3Apriorities
Many open questions...

How to handle round-robin in practice?

urgency = 3

Incremental = 1 1 0 1 1 1 1

Note: unfair Round-Robin no longer really possible

https://github.com/httpwg/http/extensions/issues?q=is%3Aissue+is%3Aopen+label%3Apriorities
Many open questions...

• Many people don’t like using HTTP headers for this
• Headers also cannot be used for changing priorities
• Exposing this to JavaScript had lots of push-back
• Are 8 levels enough? Do they really need semantics?
• What about the fairness issue?
Problem 3:

There is more than one protocol layer
Bufferbloat

Expected:

Actual:

https://www.youtube.com/watch?v=pg_xk_Pecu4
https://www.youtube.com/watch?v=ct5MvtmL1NM
TCP Head-of-Line blocking

HTTP/2

HTTP
1 2 3 4

TRANSPORT
1 2 3 4

TCP
TCP Head-of-Line blocking

Waiting for retransmit of packet 2

https://h3.edm.uhasselt.be/
QUIC has no more Head-of-Line blocking!

HTTP/2

HTTP/3

HTTP

TRANSPORT

TCP

QUIC

https://h3.edm.uhasselt.be/
QUIC has no more Head-of-Line blocking!

HTTP/2

HTTP/3

TCP

QUIC

Waiting for retransmit of packet 2

3 and 4 can be delivered to browser before 2
Has QUIC really solved Head-of-Line blocking?

Waiting for retransmit of packet 2

QUIC 1 3 4 2

https://h3.edm.uhasselt.be/
Has QUIC really solved Head-of-Line blocking?

**QUIC (Round Robin)**

1 3 4 2

Waiting for retransmit of packet 2

**QUIC (Sequential)**

1 2 3 4

Waiting for retransmit of packet 2

**TCP**

1 2 3 4

Waiting for retransmit of packet 2

[https://h3.edm.uhasselt.be/](https://h3.edm.uhasselt.be/)
Multiple other challenges in QUIC

Retransmits, multipath, flow control, scheduling APIs,...

Please read our paper:


(no one else will)
Problem 4: There is no problem
Incompatible results

Edge 50% slower
9/34 deployments broken

Max 3.1% difference
Haven’t seen large scale complaints...
Incompatible results

Edge 50% slower
9/34 deployments broken

Max 3.1% difference
Haven’t seen large scale complaints...

Web Performance isn’t important
Robin has wasted his life
Personal conclusion

- Most important for complex pages on slow networks
- Network often isn’t the bottleneck anymore... (single thread JS)
- If things break, they break hard, and are fixed
- People might not identify problems as being priority-related
- Uneven browser usage shares across platforms

https://www.w3counter.com/globalstats.php
Personal conclusion

• QUIC will highlight some long-standing and new issues

• Proposed HTTP/3 scheme isn’t going to solve all problems

• But it will be easier to debug, understand, reason about

• It can also be retrofitted onto HTTP/2!!!
Problem 5:

Too many slides, not enough time
Browser doesn’t know

1. Size of resource
2. If the resource can be used progressively
3. What the resource will actually do
4. If the resource references other resources
5. The “critical path”

So it has to guess

1. Mime-type
2. Position in the document
3. How it hopes developers use things like async, defer, preload, ...

More than the server typically knows
Server could know...

(a) The HTML for a simple page.

(b) The dependency graph generated by Scout.
Another aspect: pipelined executions on main thread

https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web/
Is Round-Robin bad?

HTTP/1.1 (TCP)

HTTP/2 (TCP)

HTTP/3 (QUIC)

Bandwidth
Which resources can/should be Round-Robined?

HTML: can: if not too big
JS: partly: not blocking, rest could be ok
CSS: no: not blocking, rest could be ok

Video: no: please don’t

Fonts: no: please don’t

Images: some: useful for size metrics (but then: use CSS / <img> attributes please) but generally serious doubts about usefulness progressive JPEGs (see next slide)
On the need for progressive JPEGs

⇒ Main downside in chrome is not taking advantage of progressive jpegs... personal opinion: don’t matter that much
   https://tobias.is/blogging/even-faster-images-using-http2-and-progressive-jpegs/
   http://blog.patrickmeenan.com/2013/06/progressive-jpegs-ftw.html
   https://blog.radware.com/applicationdelivery/wpo/2014/09/progressive-image-rendering-good-evil/ : progressive JPEG is worse than normal...
   https://twitter.com/kornelski/status/1222618103873441793?s=20 : limited proof it works

⇒ https://www.davidtnaylor.com/eyeorg.pdf : speedindex is badly correlated with user opinions (See also
   https://www.mediawiki.org/wiki/Wikimedia_Performance_Team/Perceived_Performance: “Existing RUM metrics like onload, TTFP as well as SpeedIndex correlate very poorly with user-perceived page load time”)

⇒ Maybe will change in future with AV1, but...

⇒ Also: webp is smaller, so what matters more: file size or blocky previews?

Preloadscanner: doesn’t always know an image will be off-screen at that point --> will have to change priorities down the line (Chrome does do this)
2 PRIORITY Messages Between Client and Server

Add CSS as child of HTML

HTML

Hero Image

JavaScript

https://h3.edm.uhasselt.be
2 PRIORITY Messages Between Client and Server

Add CSS as child of HTML

Non-exclusively

https://h3.edm.uhasselt.be
2 PRIORITY Messages Between Client and Server

Add CSS as child of HTML

OR

Non-exclusively

Exclusively

https://h3.edm.uhasselt.be
HTTP/3: PRIORITY messages should be ordered...

Add Image as child of HTML

Add CSS as child of HTML

OR

HTML

Hero Image

CSS

HTML

Hero Image

OR

CSS

HTML

Hero Image

QUIC: introduces new problems: retransmission!
Welcome to the jungle
Image sources

- https://i.pinimg.com/originals/99/e9/68/99e9689aa6a7a68d846ae942b401eacd.jpg
- https://2.bp.blogspot.com/-7ED7iessIXU/WmLu04G6HWI/AAAAAAAAKec/ysLVyakN-cL2D0CKD23IZUEUSdIFA13QCLcBGAs/s1600/Chrome_win_browser_wars.png
A free PowerPoint Template made by Slidor.