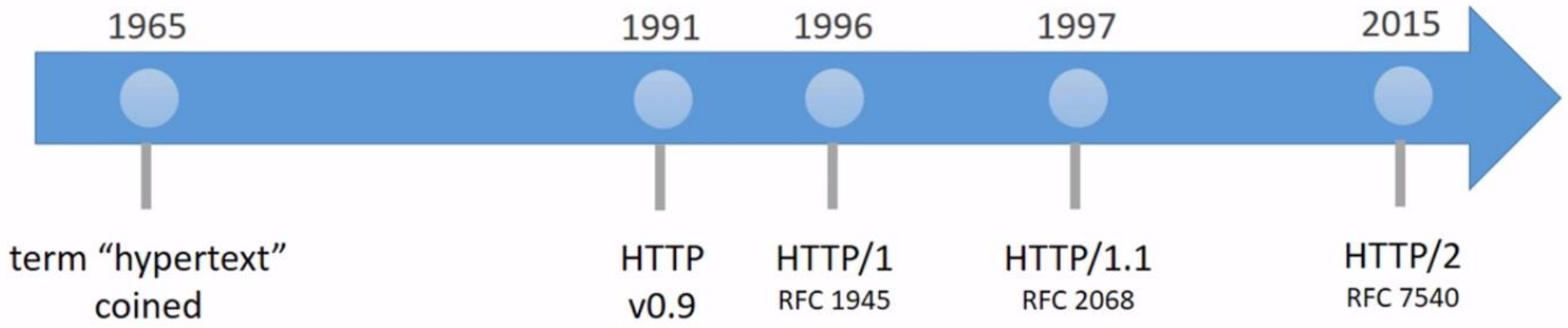
HTTP/2

The Evolution Continues

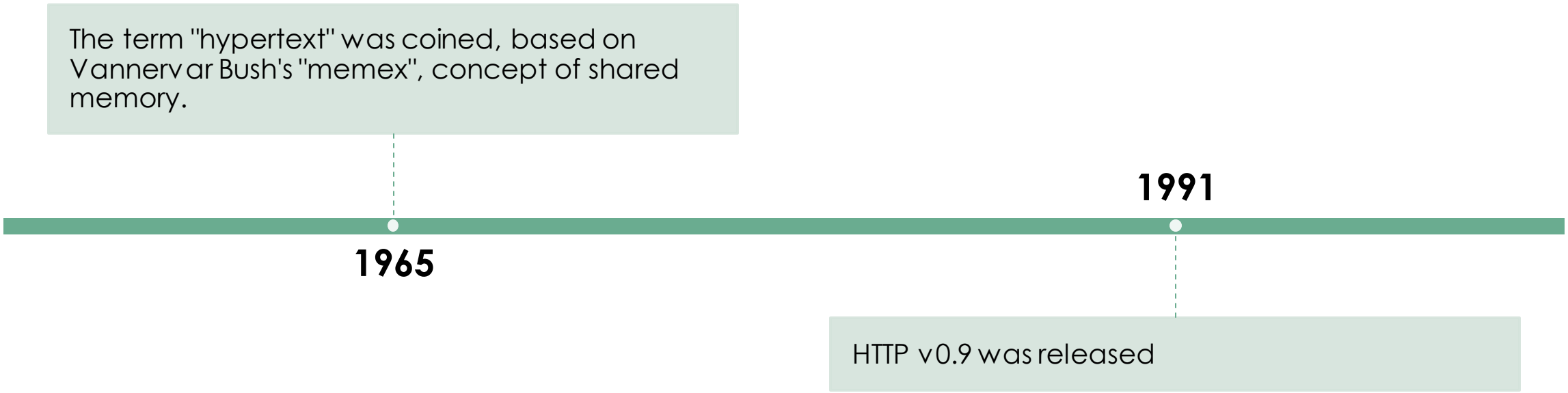
Agenda

1. HISTORY LESSON
2. HTTP 1.X ISSUES
3. WORKAROUNDS
4. HTTP2



Bird's eye view

1965 - 1991


The term "hypertext" was coined, based on Vannervar Bush's "memex", concept of shared memory.
1965
1991
HTTP v0.9 was released

TCP Based

ASCII Protocol

Single-line Request

Sequential Request

HTTP v0.9

- Headers
- Status Codes
- POST & HEAD

HTTP/1

1997 - HTTP/1.1

Reuse TCP Connection

GET, POST

PUT, DELETE

OPTIONS

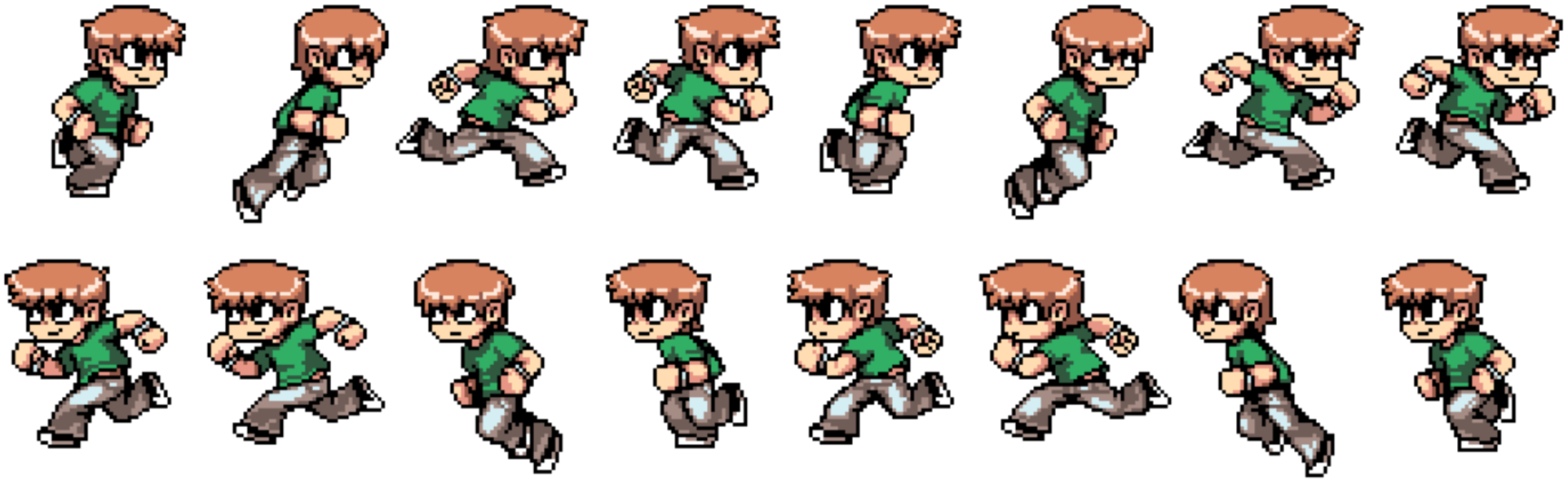
CONNECT

HEAD

TRACE

Issues with HTTP/1.X

- Head of line blocking
- Single Request/Response at a time
- Text based protocol(Uses ASCII encoding)
- Round-trip Bonanza
- Increased Latency



HTTP/1.X Workarounds

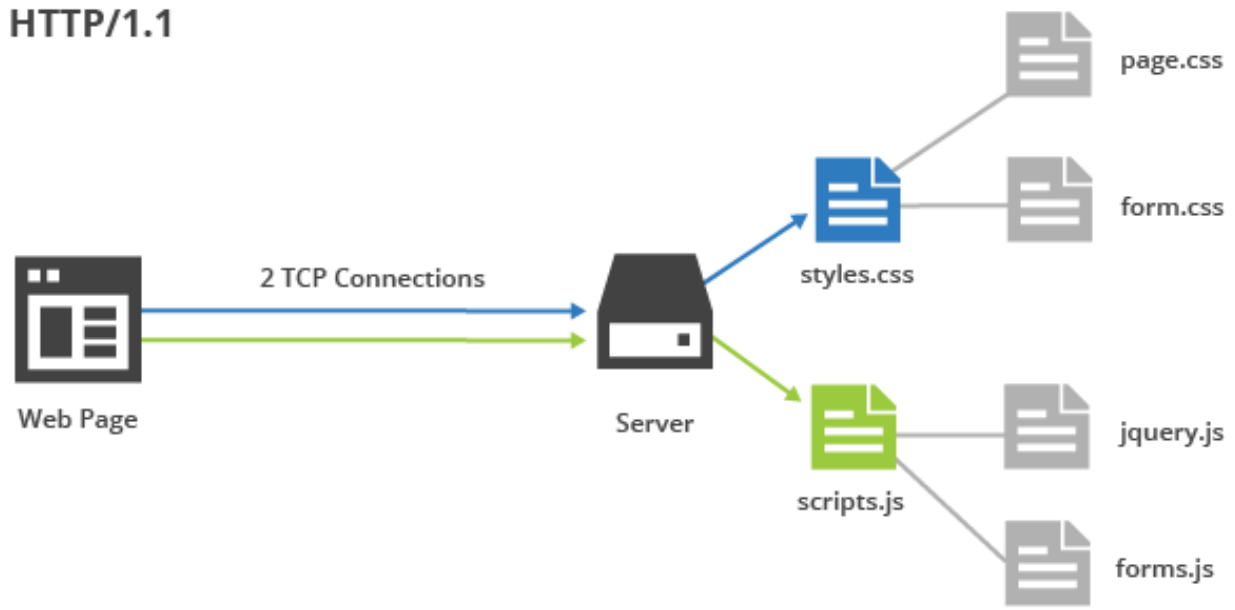
IMAGE SPRITING

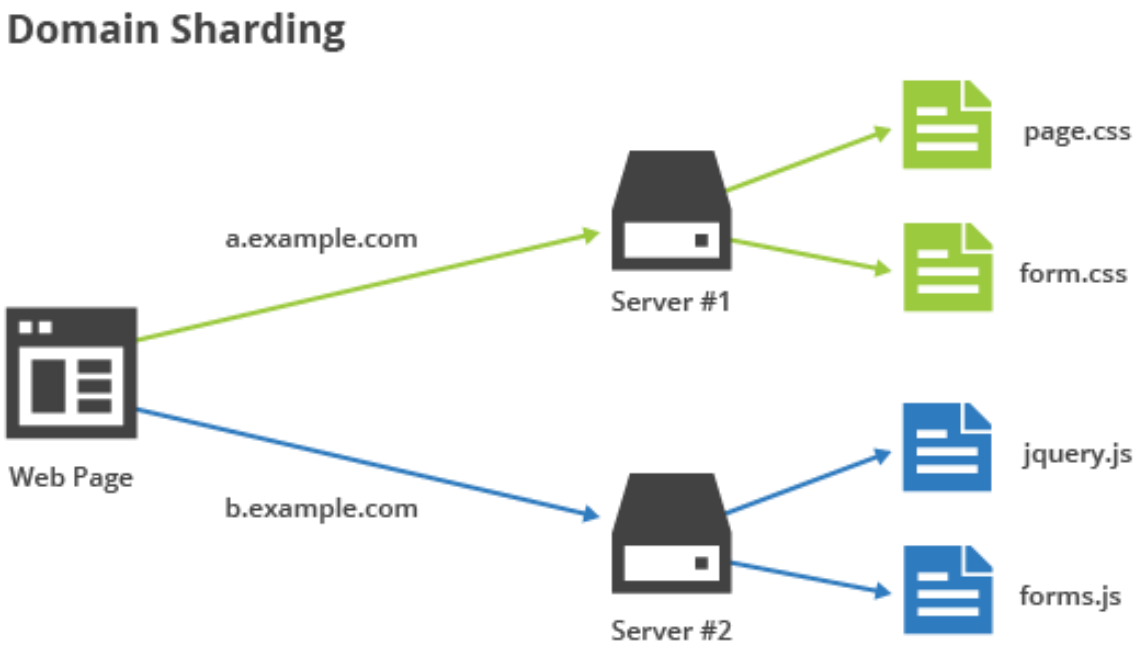
HTTP/1.X Workarounds

Inlining

HTTP/1.X Workarounds

- File Concatination





HTTP/1.X Workarounds

DOMAIN SHARDING

drum roll
please...

HTTP/2

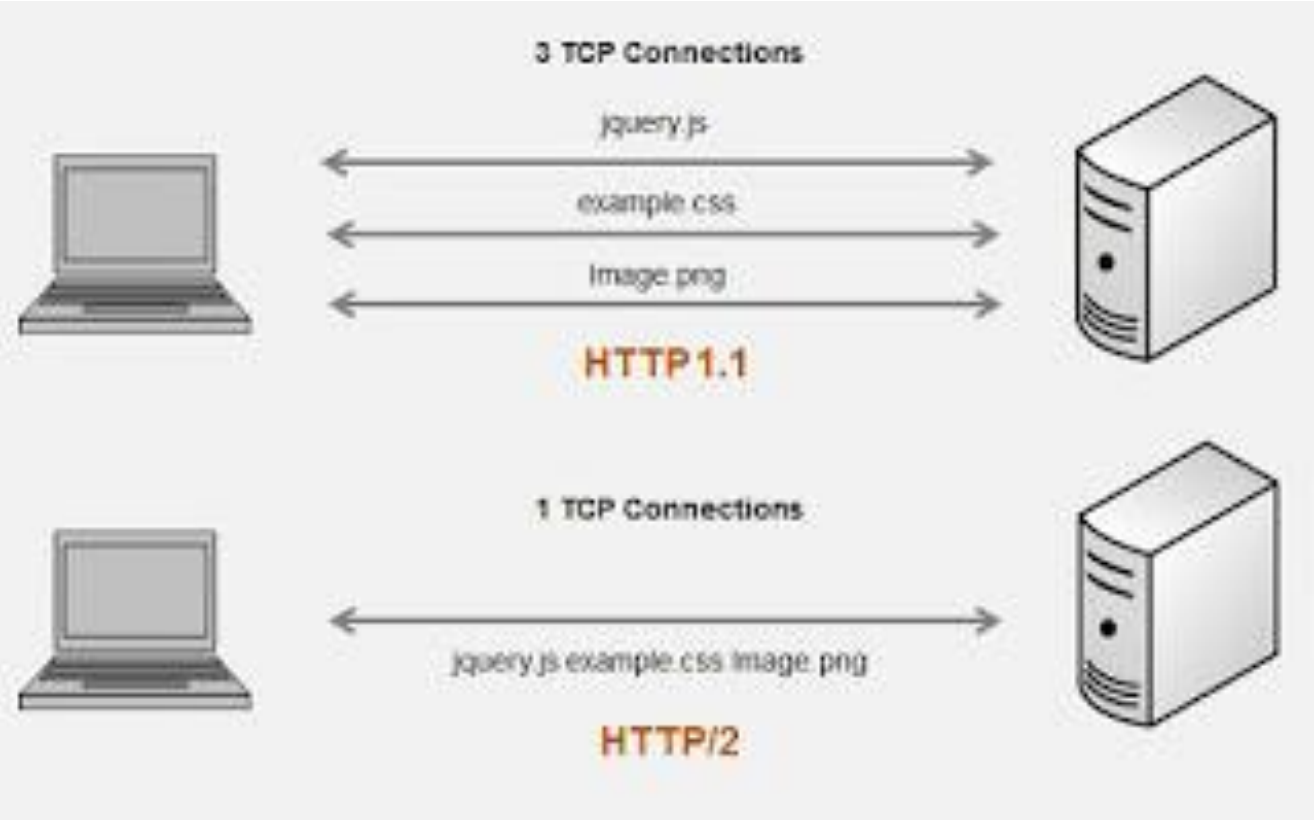
For Faster and Safer Internet

How does it help?

- Secure By Default
- Single TCP Connection
- Binary Protocol
- Fully Multiplexed
- Header Compression
- Server Push

Secure By Default

- TLS specification is optional
- Browser vendors **only** support http/2 **with** TLS implementation



Single TCP Connection

- One TCP connection per server.
- Reduces number of requests made.

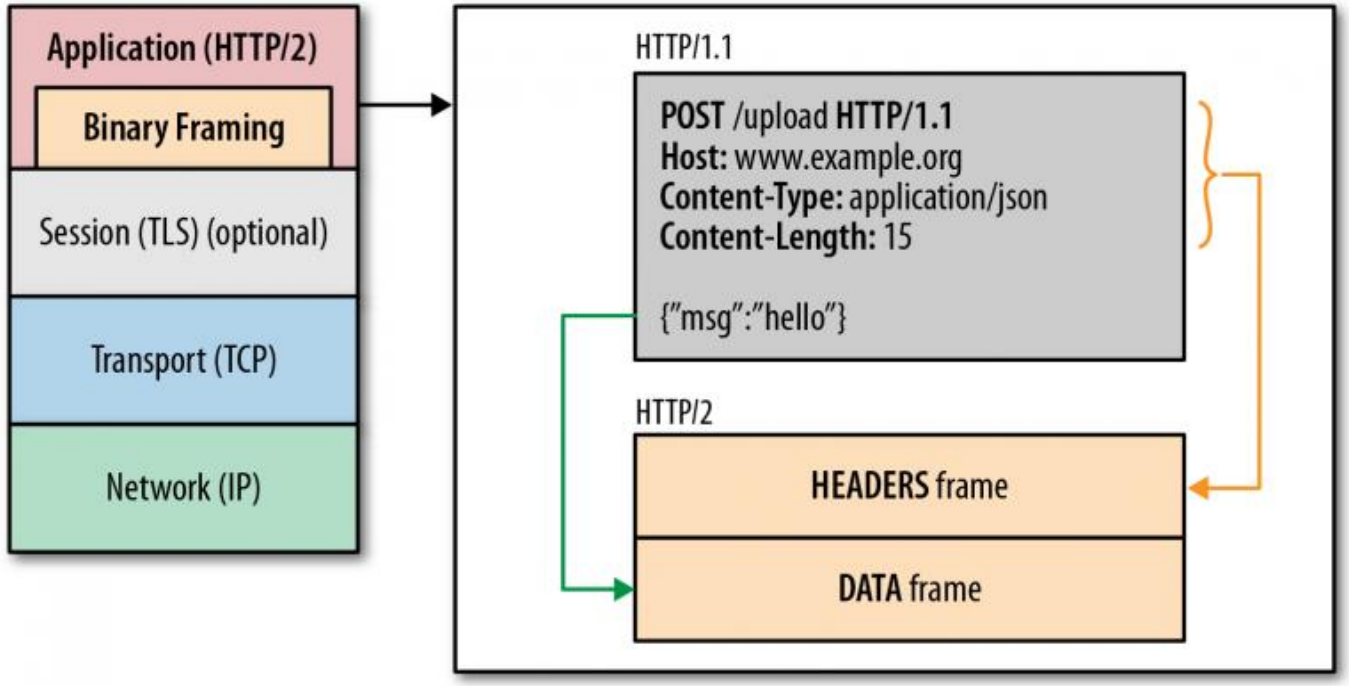
Single TCP Connection

Reduces no. of three way handshakes

Less TCP/IP connection streams are more efficient

Increases overall throughput

Binary Protocol

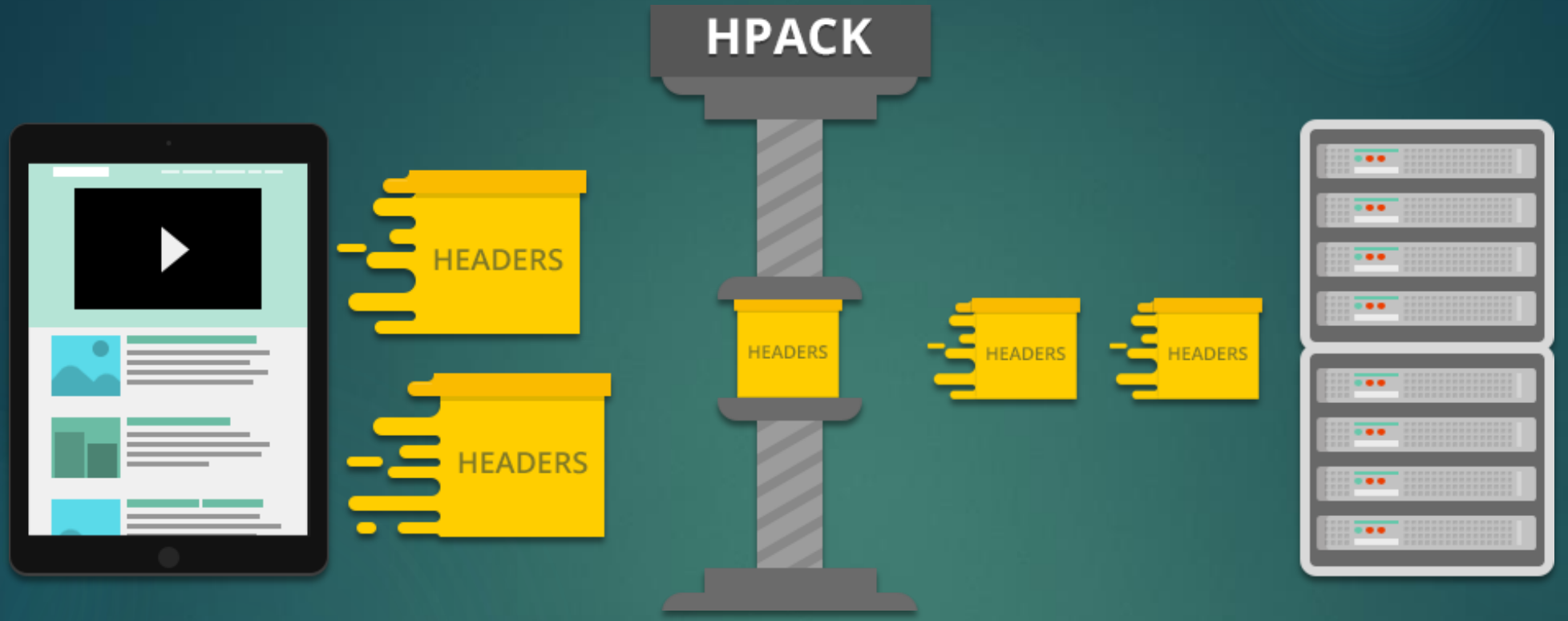


Binary Protocol

A request/response in HTTP1.1 is a single enclosed unit, in HTTP/2 messages are split up in *frames*

Every frame can be assigned to a *stream* by its *stream-id*

These frames can be sent/received asynchronously



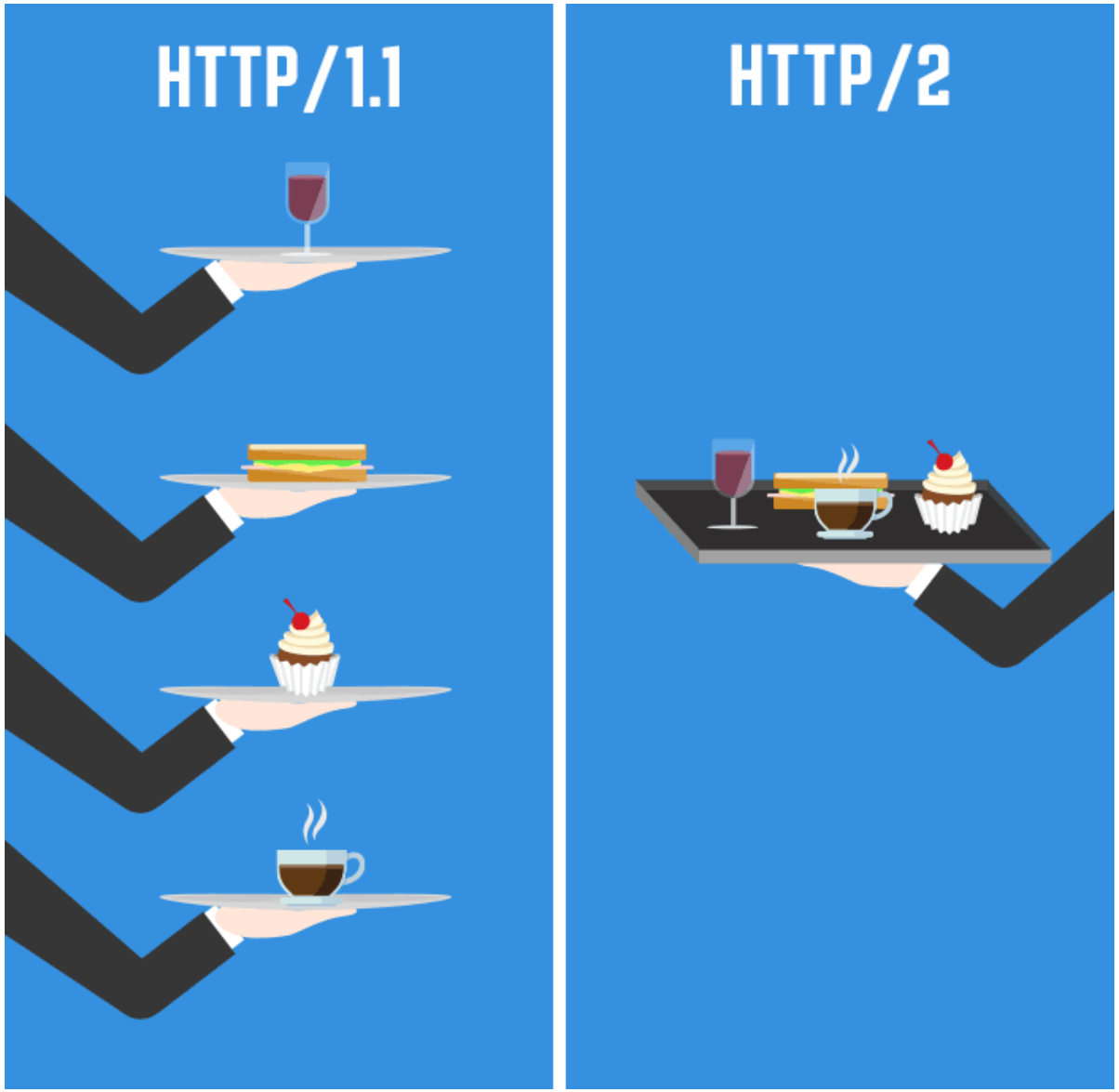
Header Compression

HPACK

Specialized Algorithm for compressing Headers

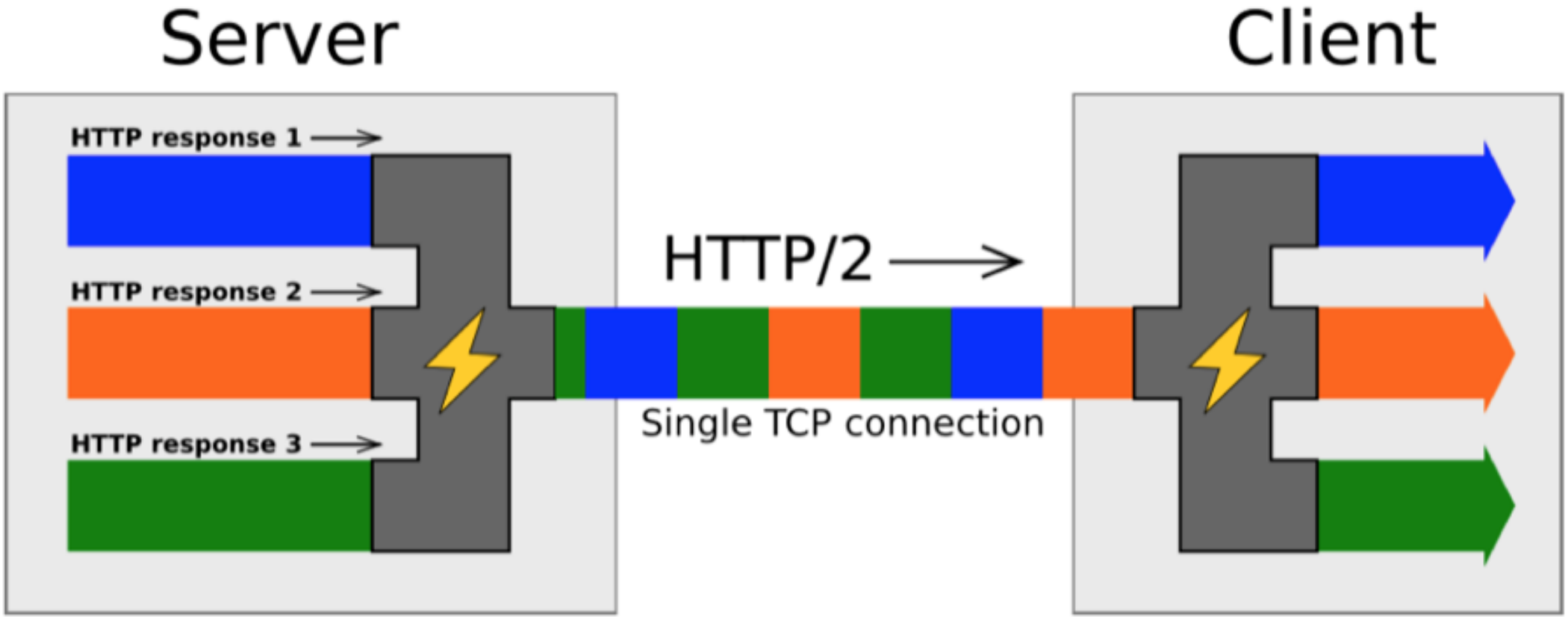
Works like gzip

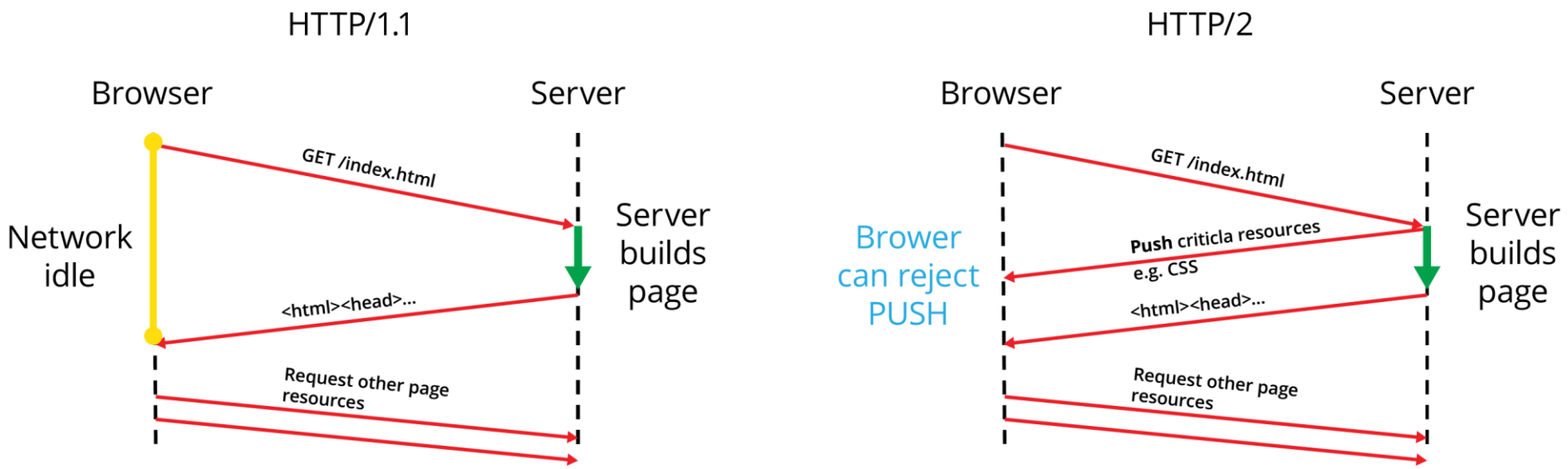
Has a look-up table of ~62 entries from most popular websites



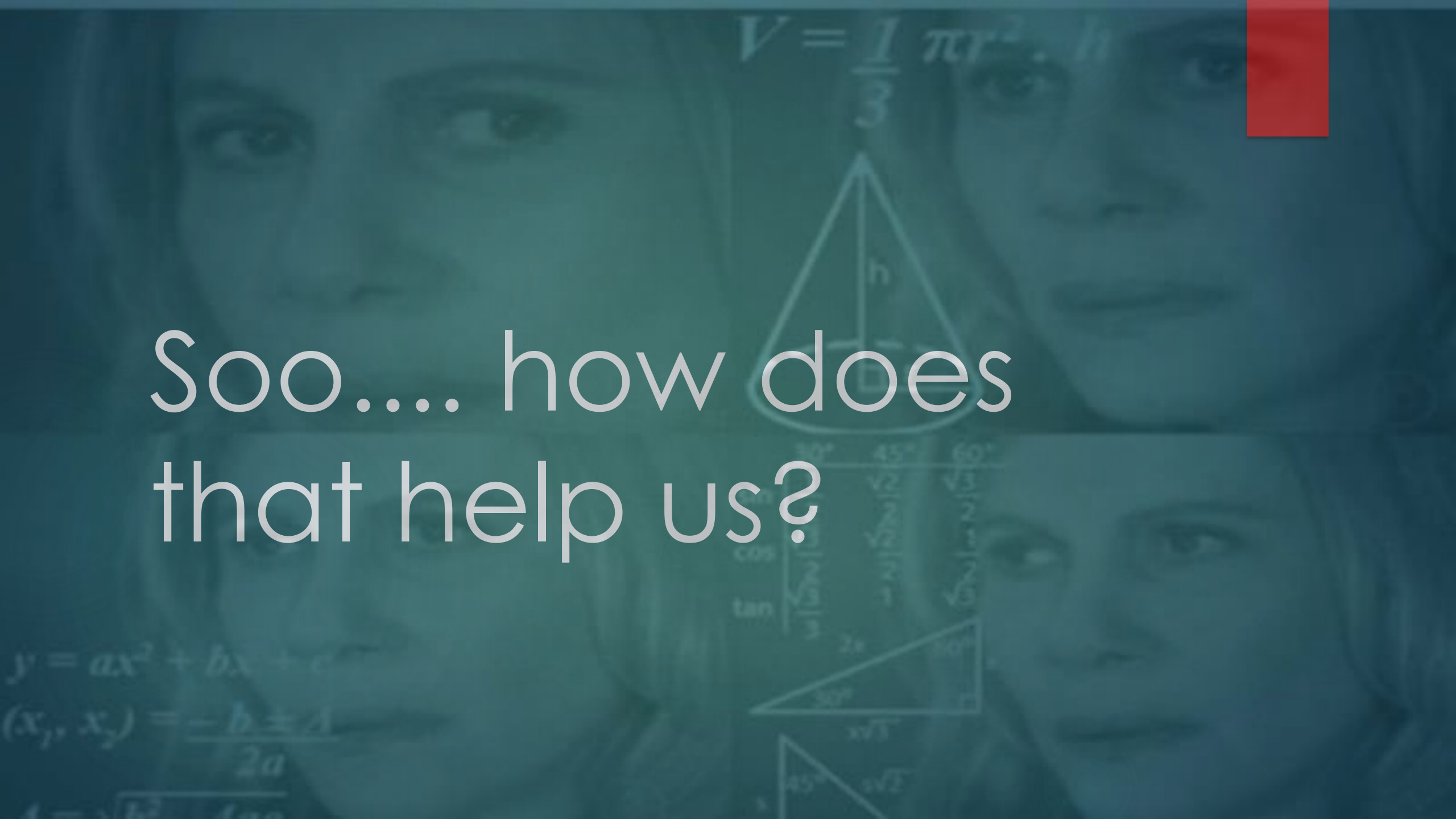
Request multi-plexing

HTTP/2 Inside: multiplexing





HTTP/2 Server Push



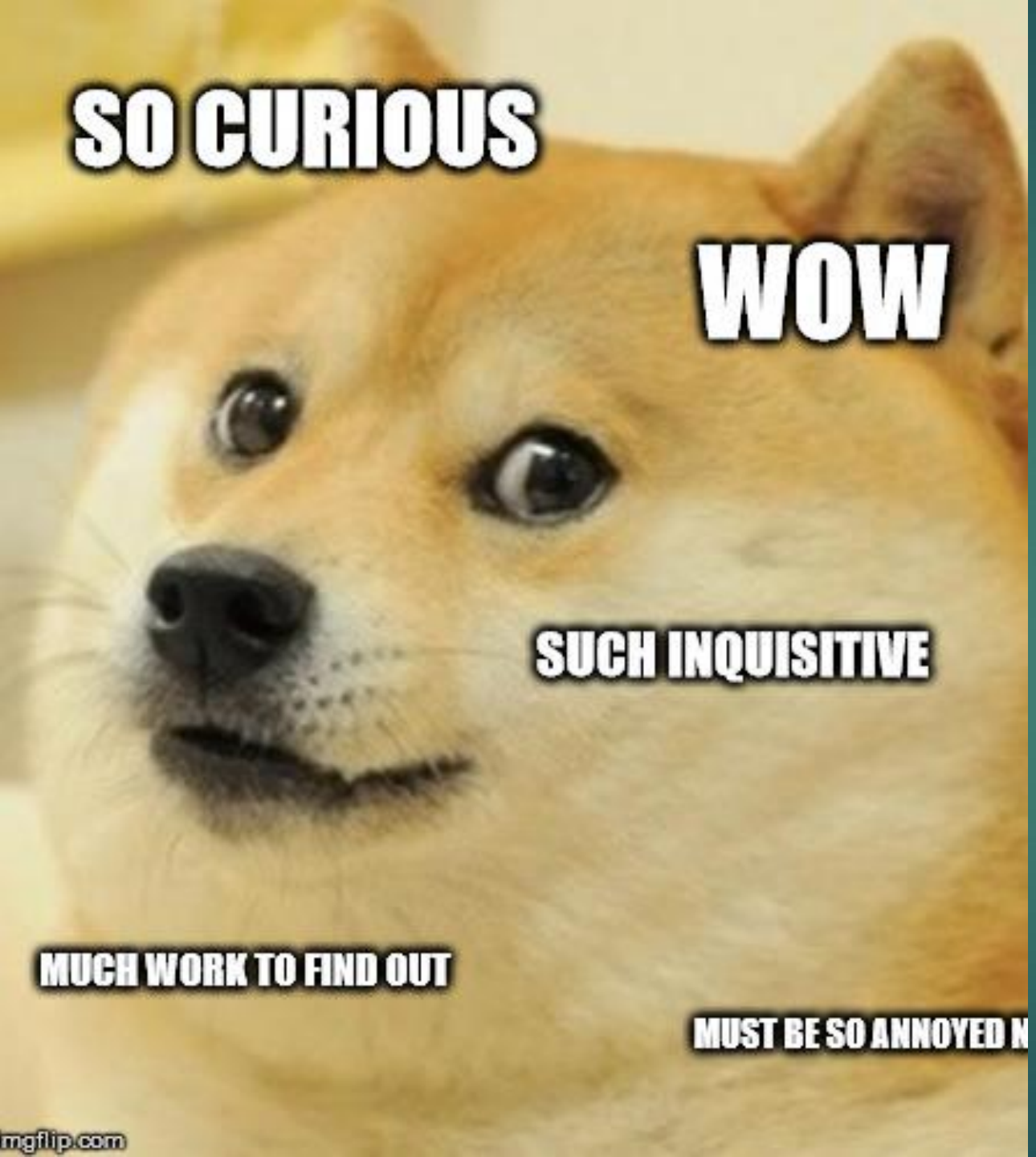
Soo…. how does that help us?

HTTP/2 on user-end

- Faster page loads
- More responsive loading
- Decreased bandwidth usage

HTTP/2 on developer's end

- No need for HTTP/1.X work-arounds
- Decreases CPU & Bandwidth usage on server end



Curious about HTTP/3?

IT MIGHT JUST HAPPEN SOONER THAN IT TOOK US TO MOVE FROM HTTP/1.1 TO HTTP/2

THE END