



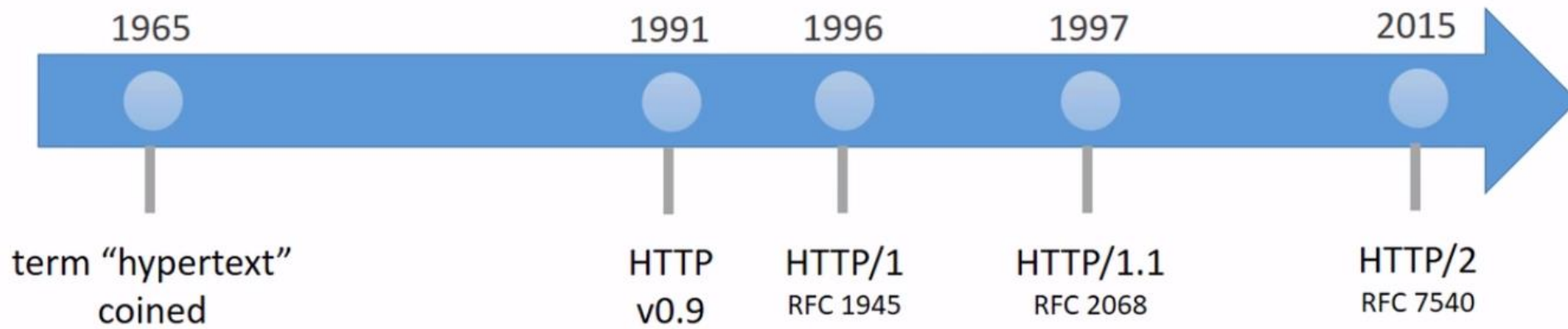
HTTP/2

The Evolution Continues



Agenda

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



Bird's eye view

The term "hypertext" was coined, based on Vannervar Bush's "memex", concept of shared memory.

1965

1991

HTTP v0.9 was released

1965 - 1991

TCP Based

ASCII
Protocol

Single-line
Request

Sequential
Request

HTTP v0.9



Diagram illustrating the components of HTTP/1:

- Headers
- Status Codes
- POST & HEAD

The diagram shows three horizontal yellow bars, each with a white text label. To the left of each bar is a vertical line that connects to a horizontal line extending to the right, forming a bracket-like structure. The background is white with a dark teal wavy shape at the bottom.

Headers

Status Codes

POST & HEAD

HTTP/1

1997 - HTTP/1.1

Reuse TCP
Connection

GET, POST

PUT, DELETE

OPTIONS

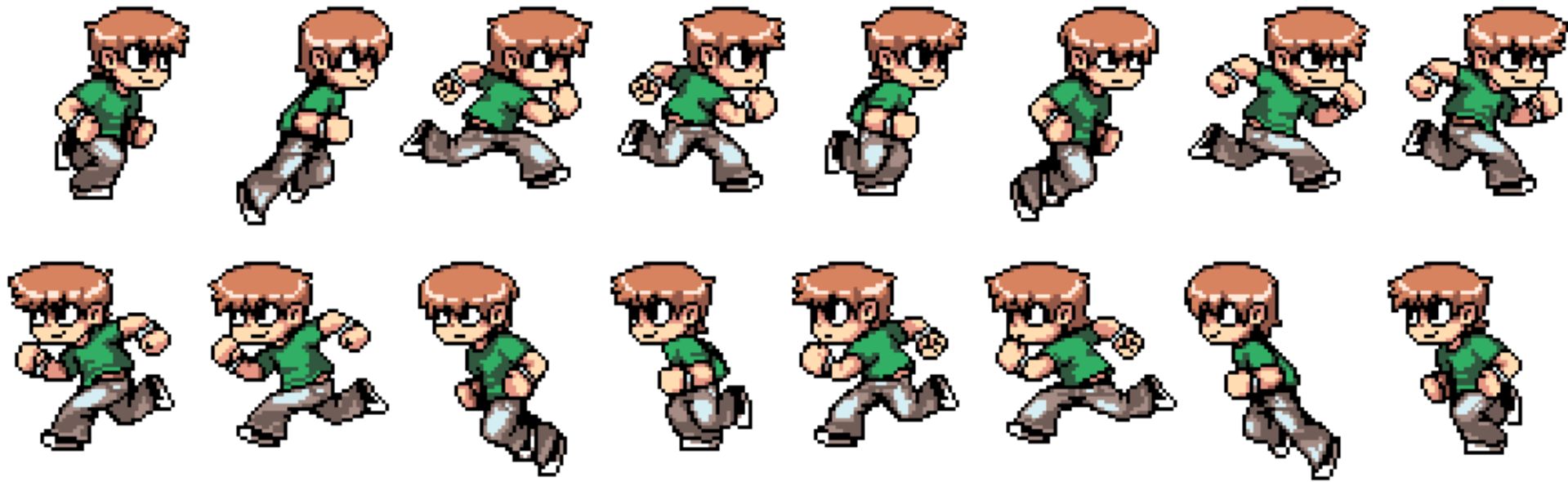
CONNECT

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

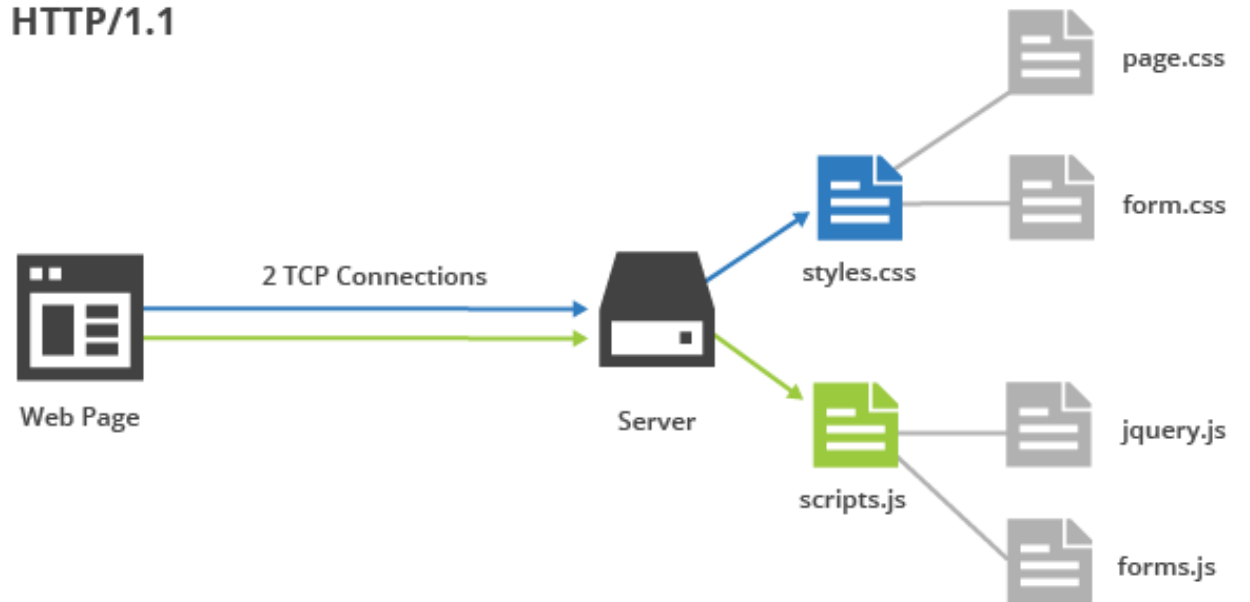
HTTP/1.X Workarounds

Inlining

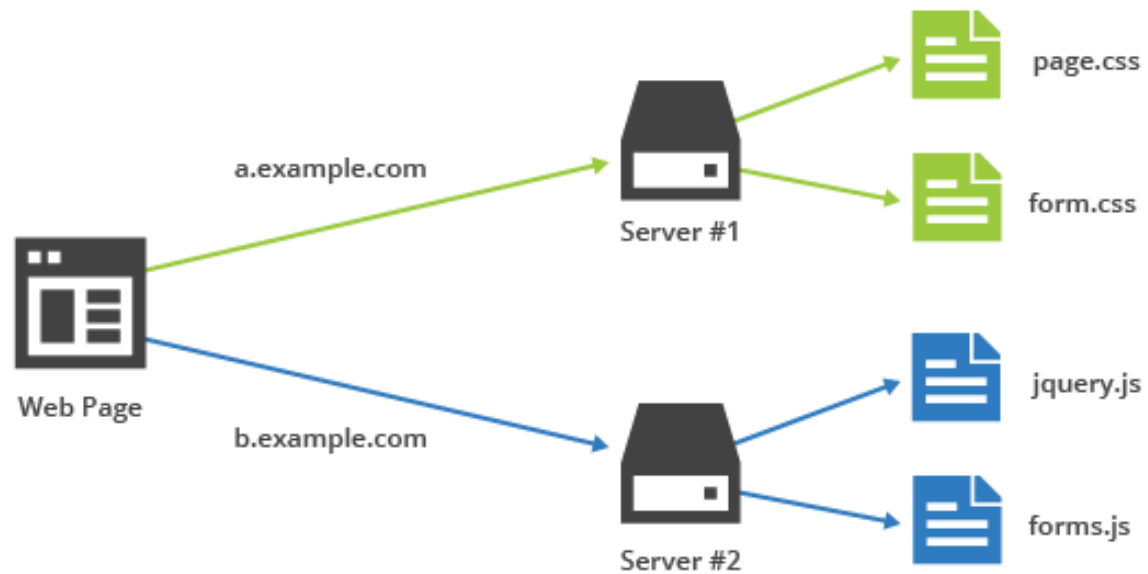
HTTP/1.X Workarounds

- File Concatination

HTTP/1.1



Domain Sharding



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

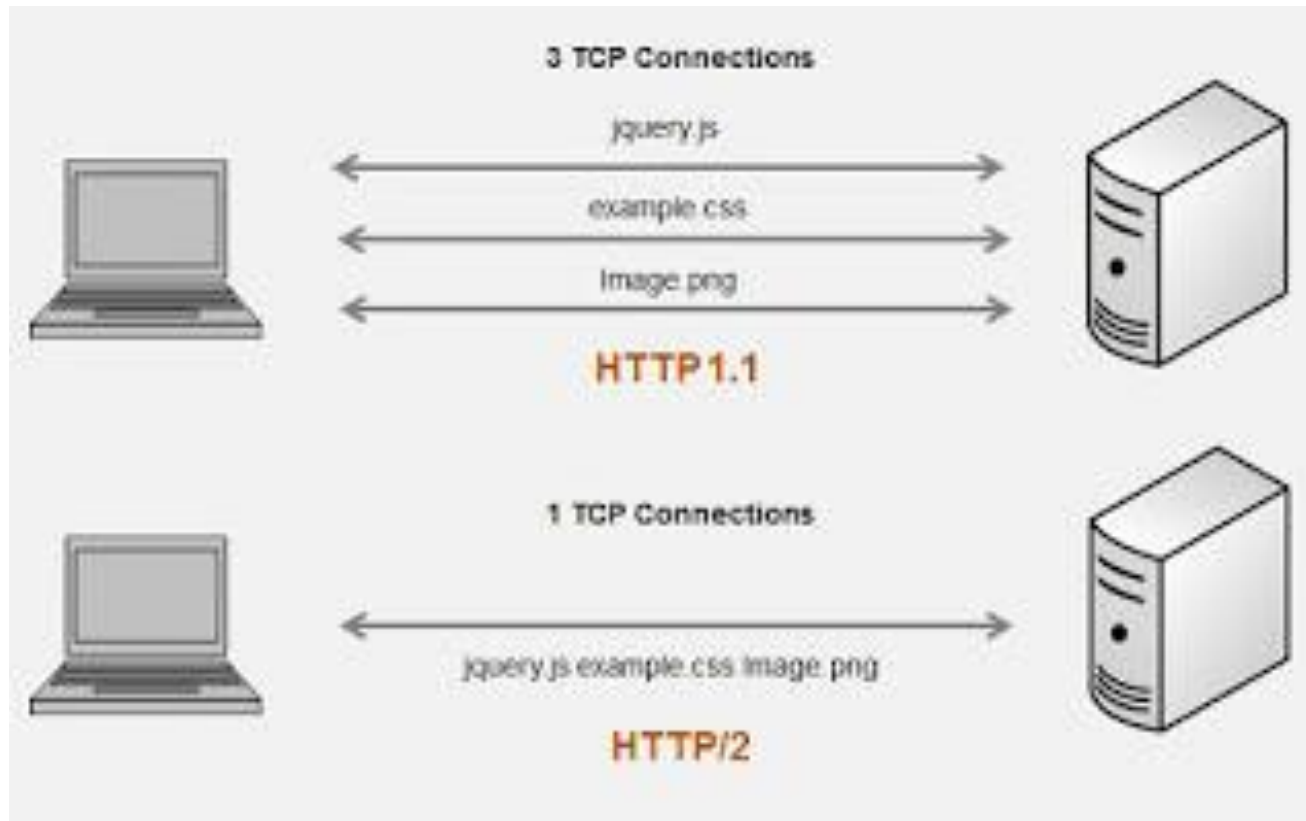
Header
Compression

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

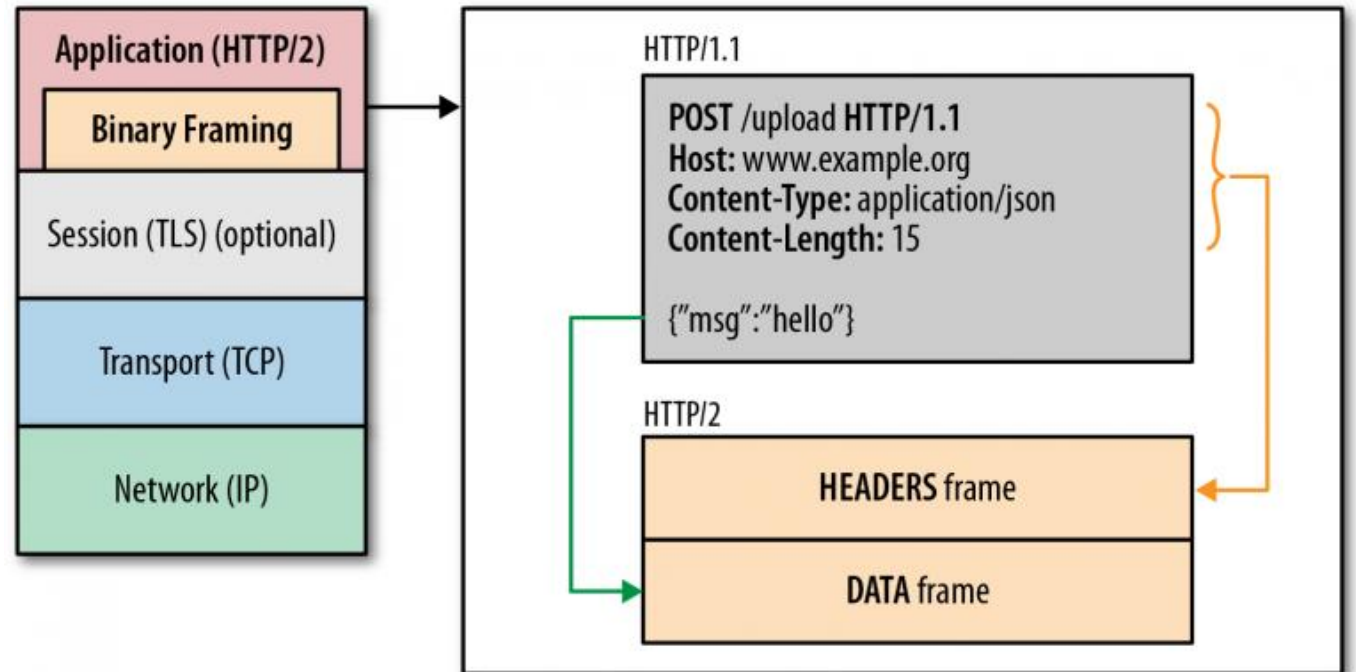


Reduces no. of three way handshakes

Less TCP/IP connection streams are more efficient

Increases overall throughput

Binary Protocol

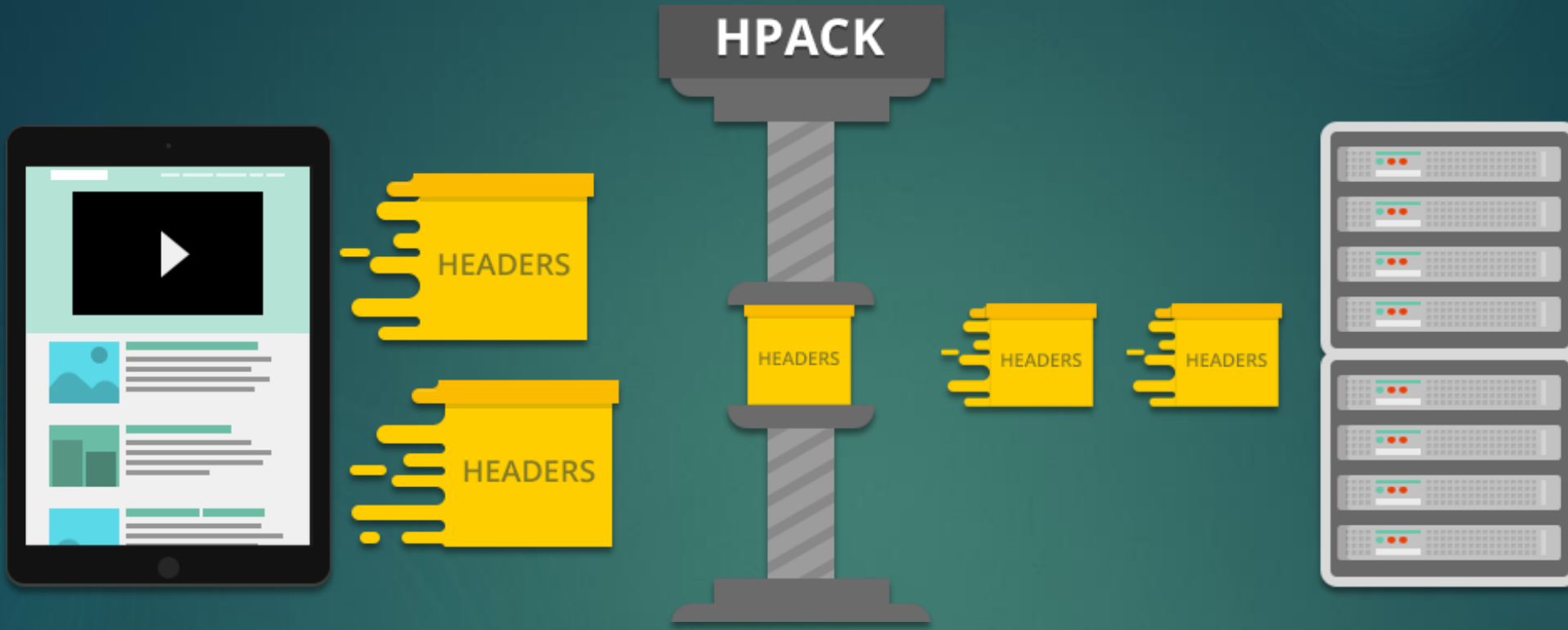


Binary Protocol

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

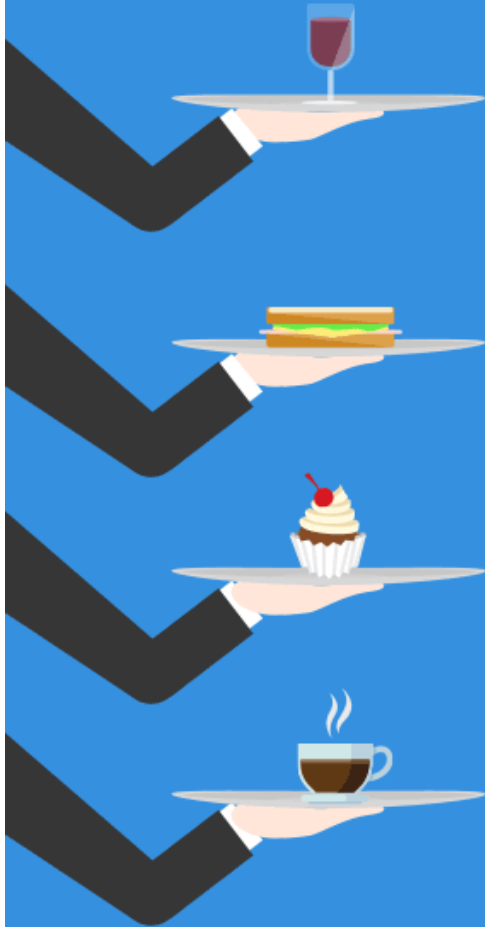


Specialized Algorithm for compressing Headers

Works like gzip

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

HTTP/1.1



HTTP/2

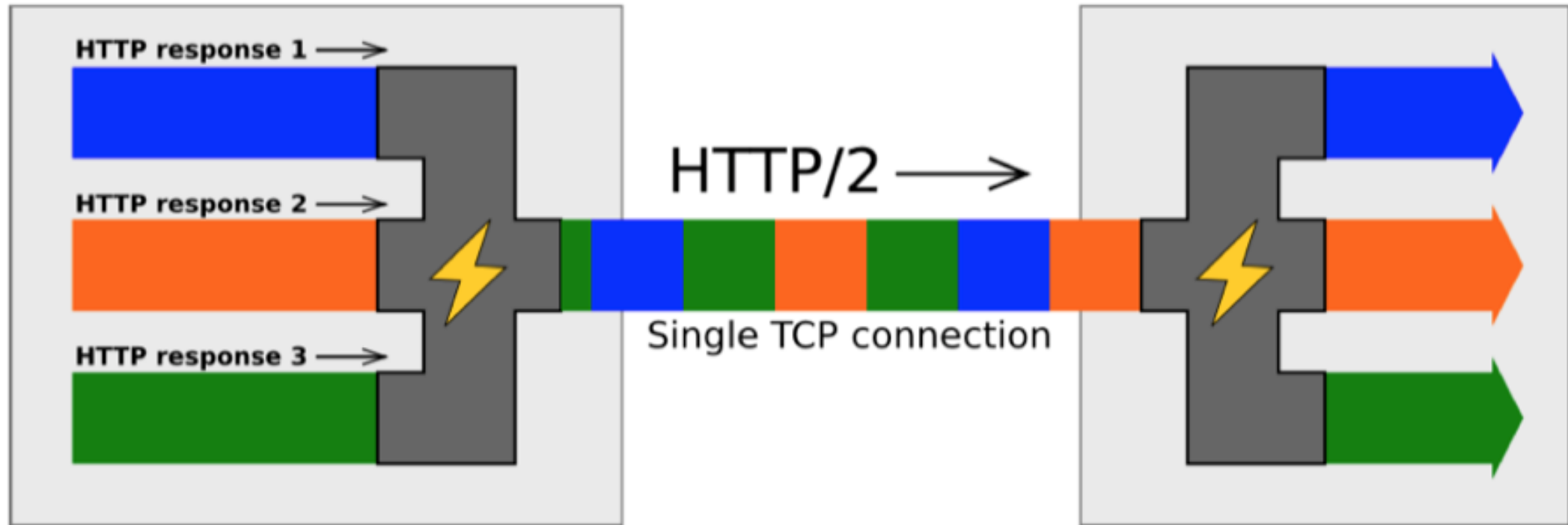


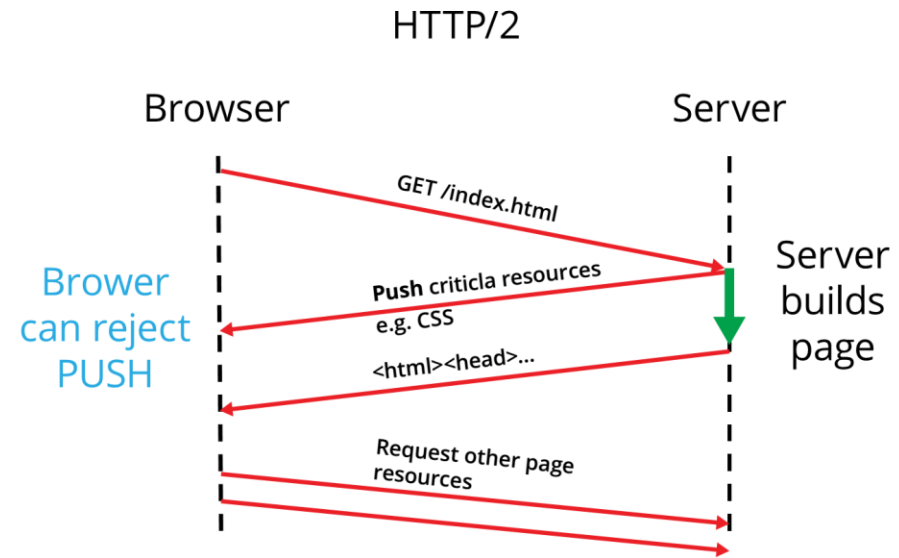
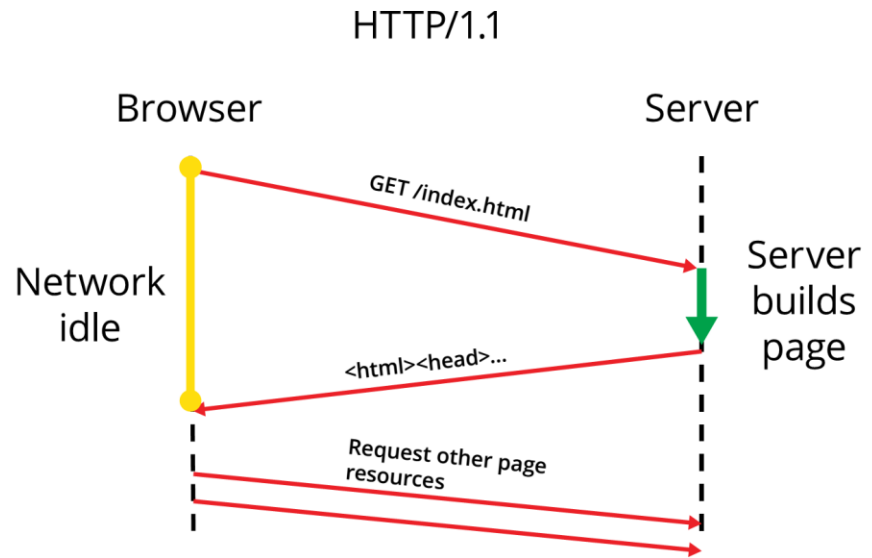
Request
multi-
plexing

HTTP/2 Inside: multiplexing

Server

Client





HTTP/2 Server Push



Soo.... how does
that help us?

The background features a collage of mathematical elements: a blue-tinted image of two faces, the volume formula $V = \frac{1}{3} \pi r^2 h$, a diagram of a cone with height h , a table of trigonometric values for 30°, 45°, and 60°, a quadratic formula $y = ax^2 + bx + c$, the quadratic formula for roots $(x_1, x_2) = \frac{-b \pm \Delta}{2a}$, the discriminant $\Delta = \sqrt{b^2 - 4ac}$, and diagrams of right-angled triangles with angles 30°, 45°, and 60° and sides labeled with $\sqrt{3}$ and $\sqrt{2}$.

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

SO CURIOUS

WOW

SUCH INQUISITIVE

MUCH WORK TO FIND OUT

MUST BE SO ANNOYED N

Curious about HTTP/3?

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



THE END