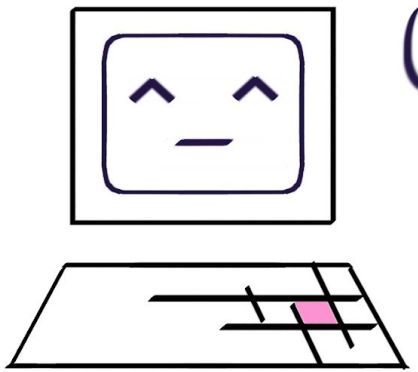
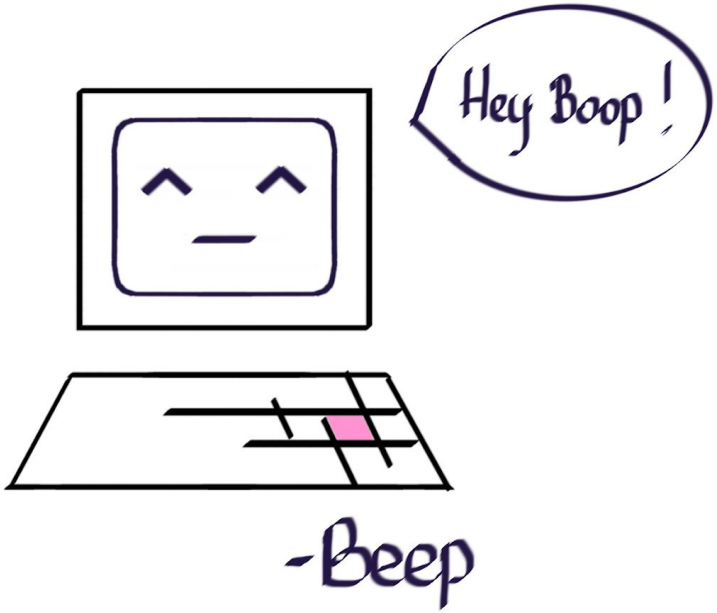
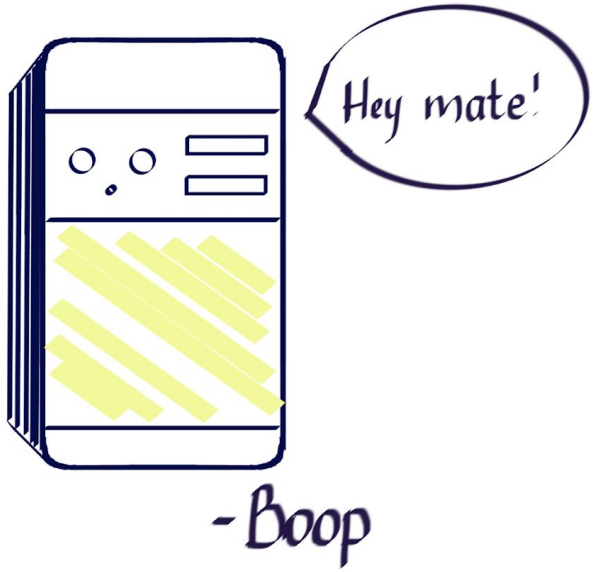


Ciao Verona!

my name is Irina;
i am an engineer at
mongodb.

🐦 - @_Irina || ✉ - irina@mongodb.com


Hey Boop!
-Beep


Hey mate!
-Boop

require('http2')

“a major protocol shift with the potential to accelerate the web.”

“How HTTP/2 Pushes the Web: an Empirical Study of HTTP/2 Server Push”
Zimmermann, Rüth, Wolters, Hohlfeld, 2017

“HTTP/2 will make our applications faster, simpler, and more robust …”

google folks over at WebFundamentals, 2018

“more efficient to parse, more compact ‘on the wire’, and most importantly, they are much less error-prone, compared to textual protocols like HTTP/1.x”

http2.github.io

Today's Agenda™:

Prologue: HTTP/1.1 vs HTTP/2

Act I: HTTP/2, the road less traveled by

Act II: some ⌨️🖱️ 💻 time

Act III: HTTP/2, why so fast

Act IV: HTTP/2 PUSH

PROLOGUE

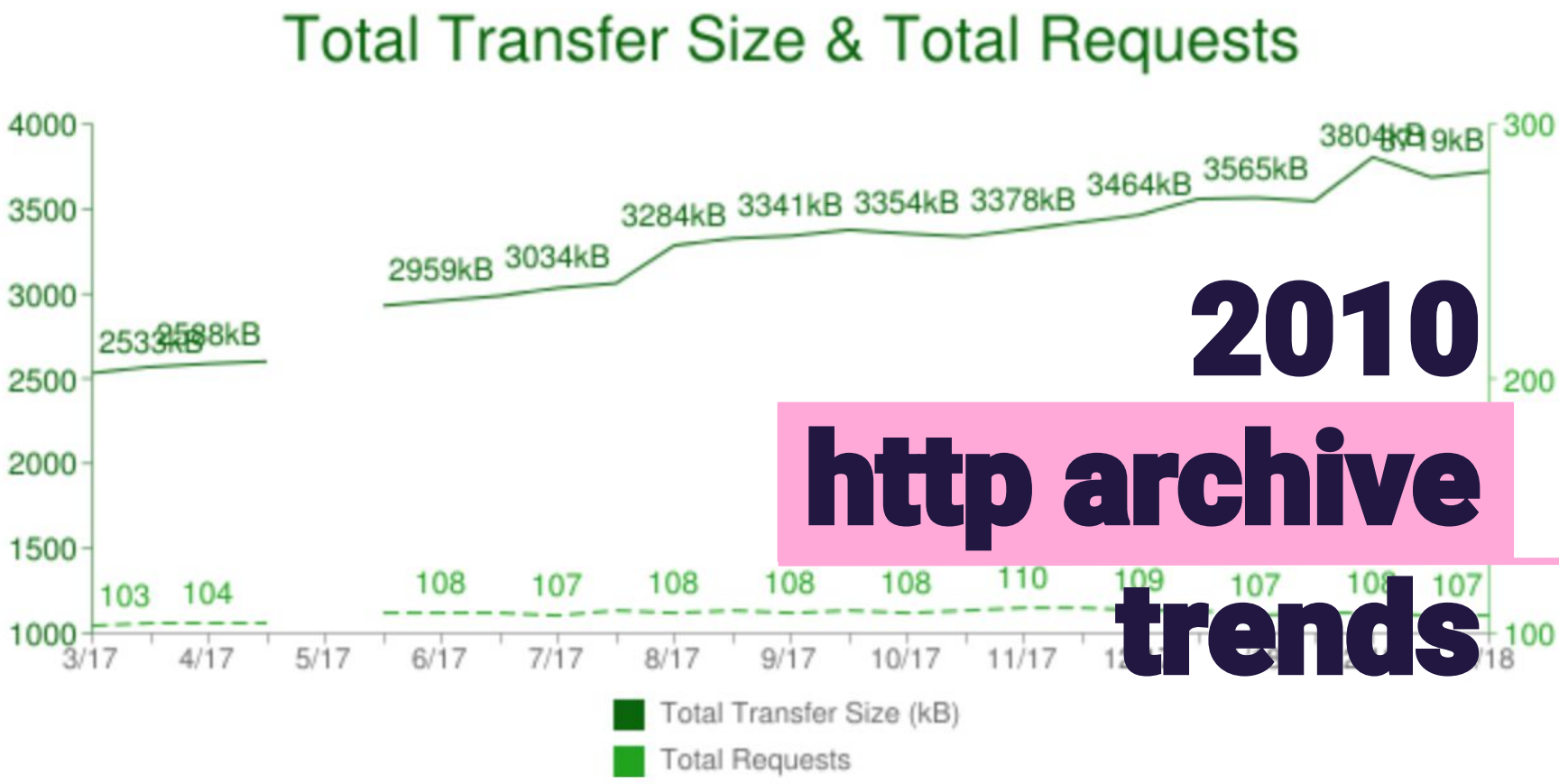
The coming of age story of your fav protocol.

(Previously on HTTP Files)

> “The focus of the protocol is on performance; specifically, end-user perceived latency, network and server resource usage.”

- IETF HTTP Working Group

year is 1999; http/1.1



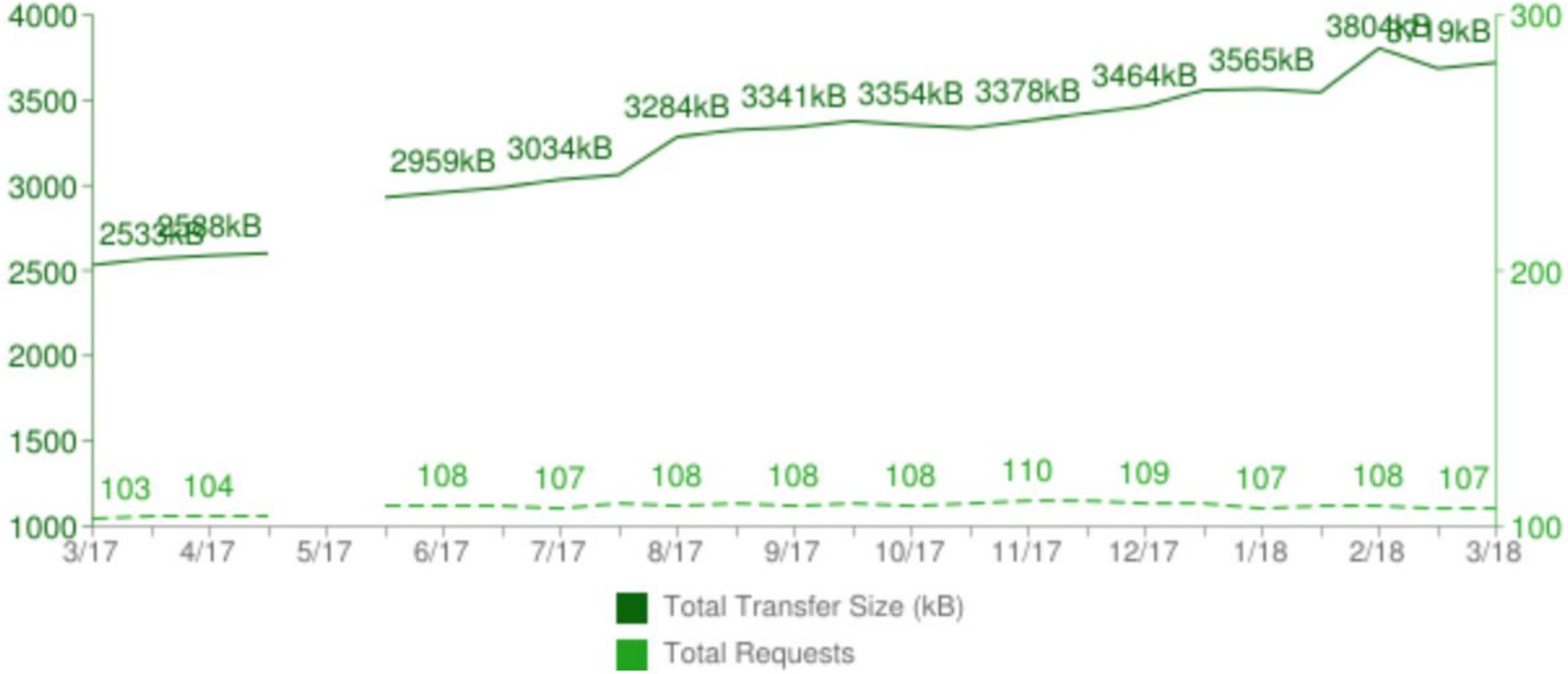
2010

http archive

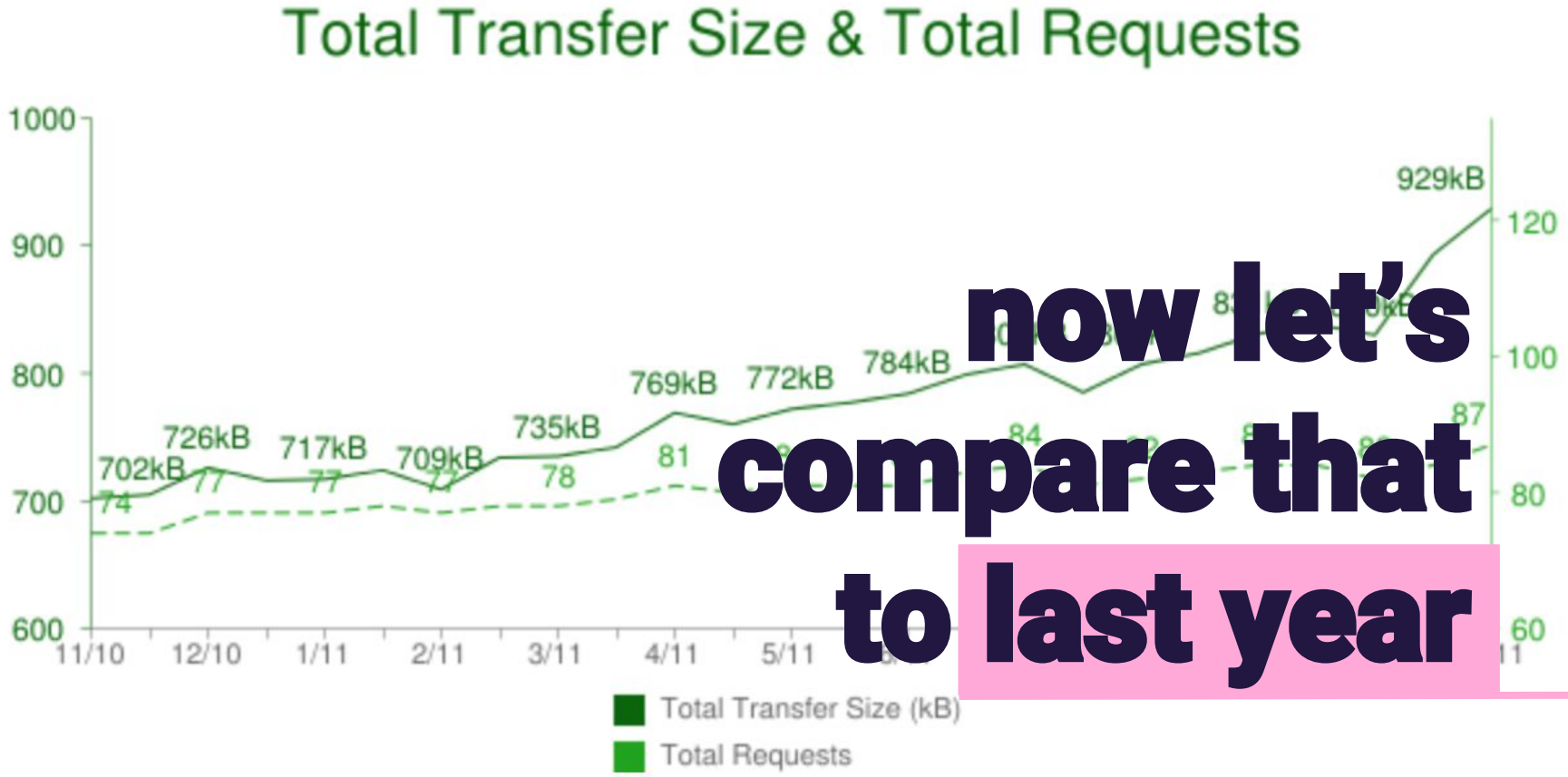
trends

{ query: "used" }

Total Transfer Size & Total Requests



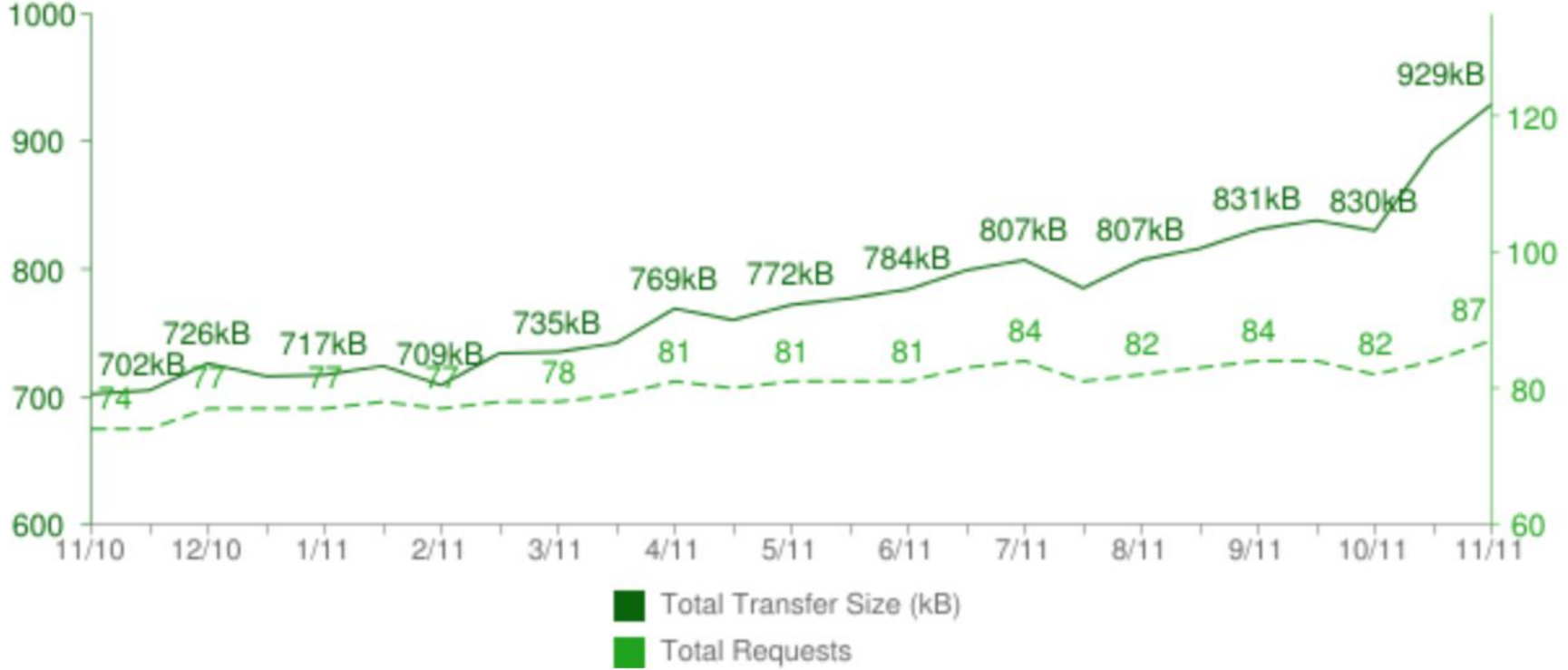
{ query: "used" }



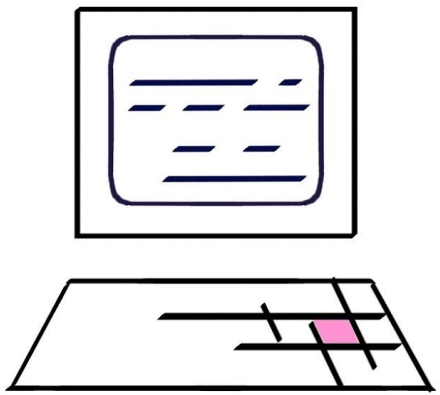
now let's compare that to last year

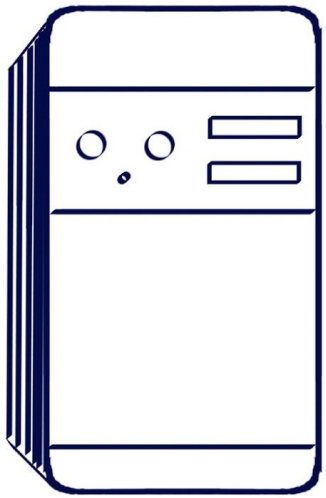
{ query: "used" }

Total Transfer Size & Total Requests



{ query: "used" }

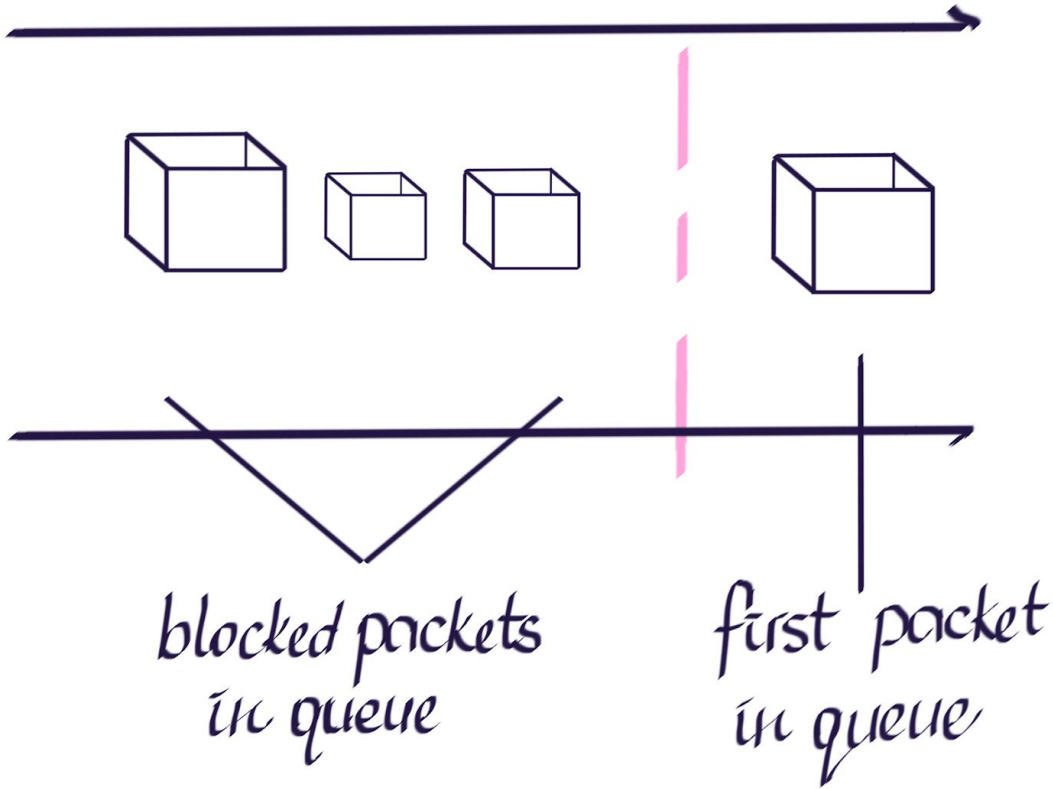




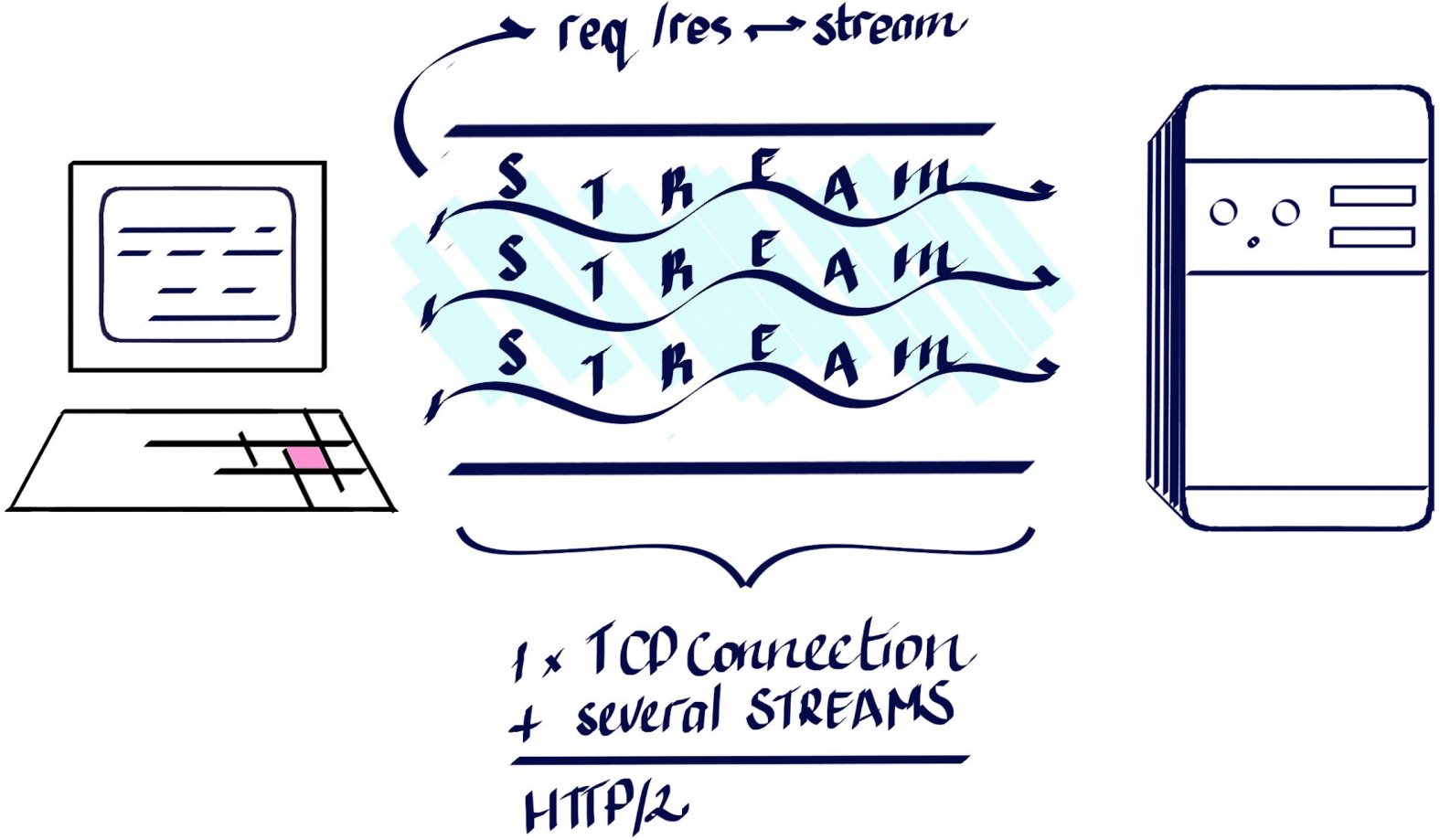
1 × TCP Connection
+ 1 × Request

HTTP/1.1

head-of-line blocking



year is 2015; http/2


req /res ⟷ stream
STREAM
STREAM
STREAM
1 x TCP Connection
+ several STREAMS
HTTP/2

HTTP/2 comes with a few lil wins

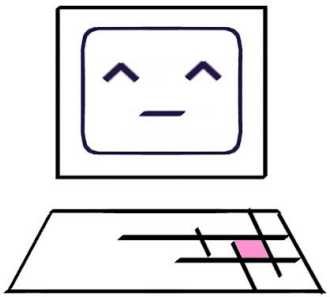
/me whispers: lil wins

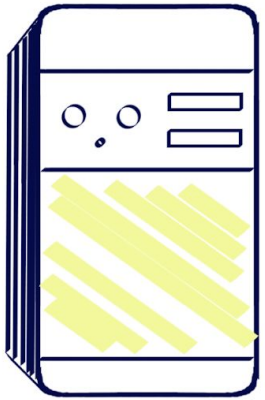
lil wins:

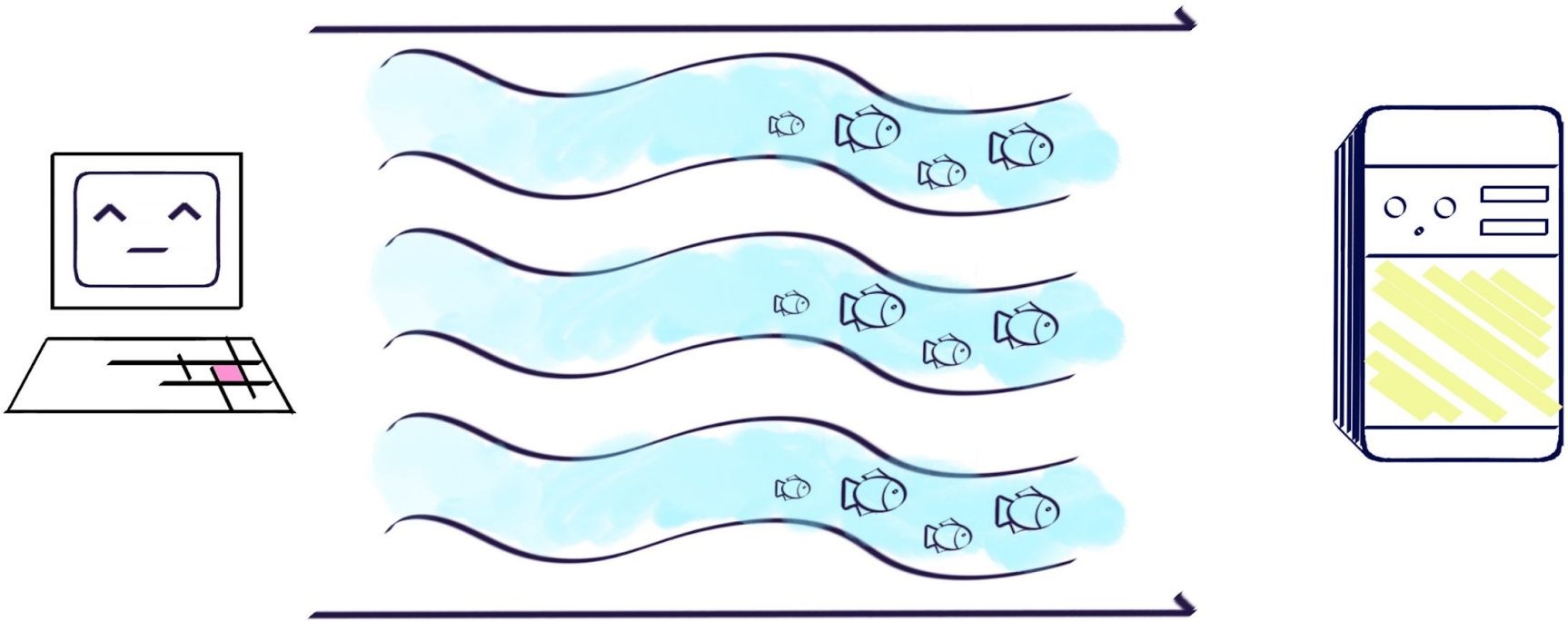
1. stream prioritization
2. server push
3. compressin' HTTP headers

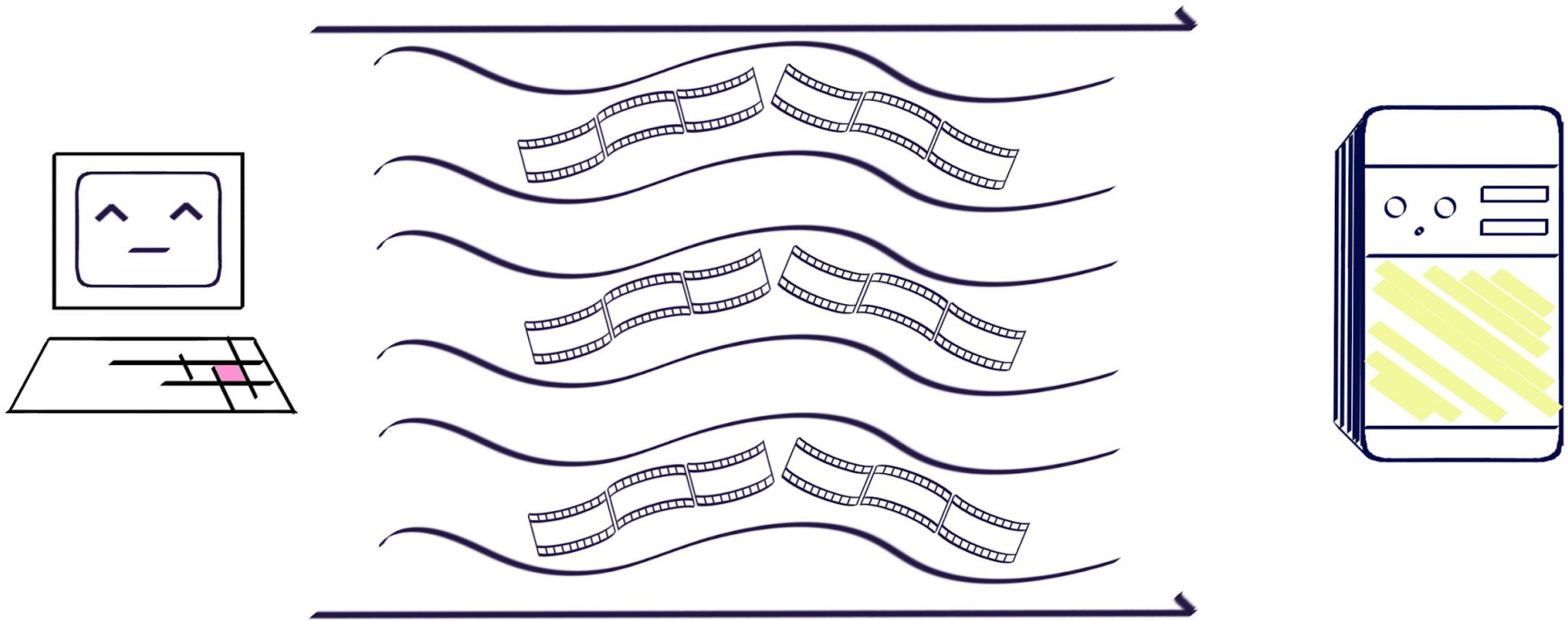
ACT 2. SCENE 1.

the connection saga.









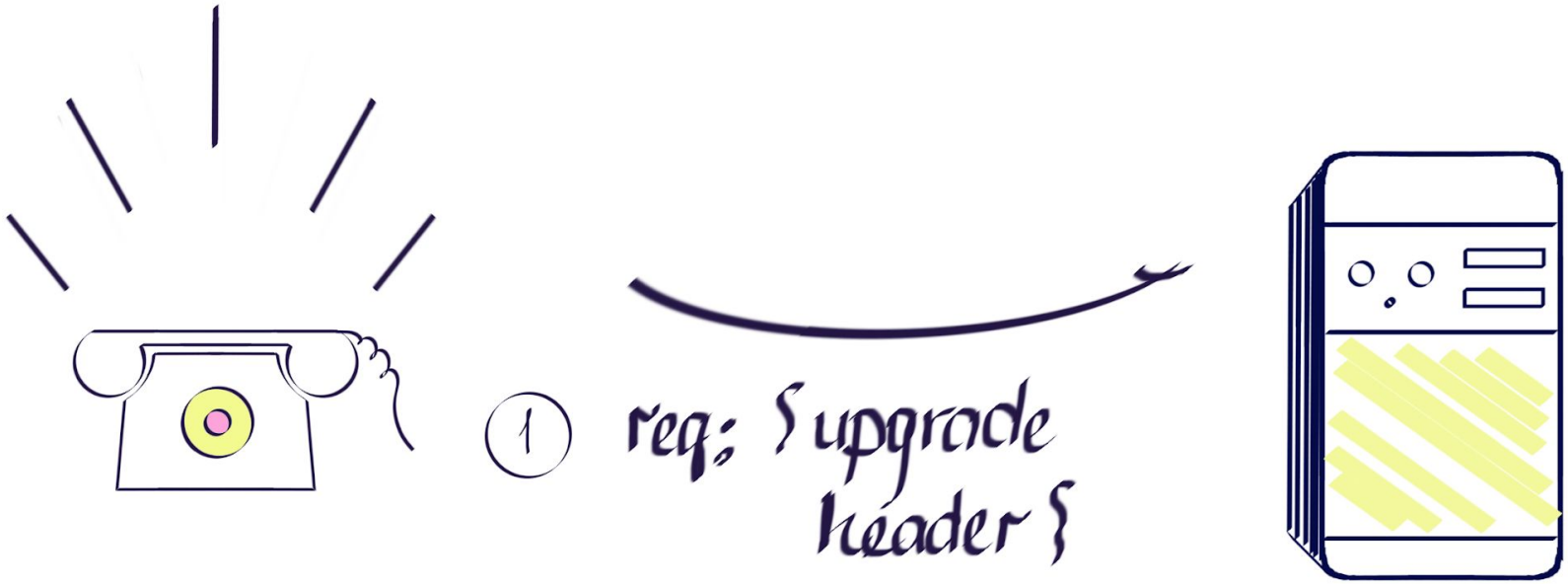
Clients won't always know ahead of time if server can accept HTTP/2

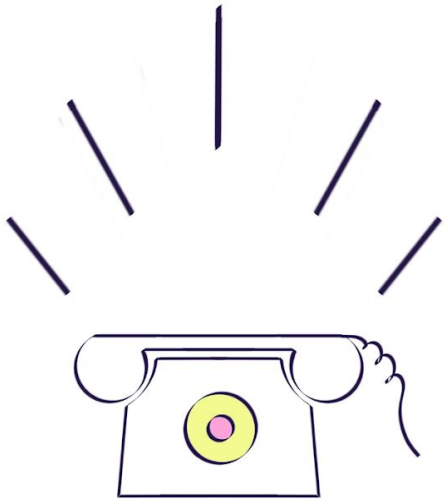
send an ‘Upgrade’ header

```
GET / HTTP/1.1
Host: 2018.jsday.it
Connection: Upgrade, HTTP2-Settings
HTTP2-Settings: <settings payload>
```

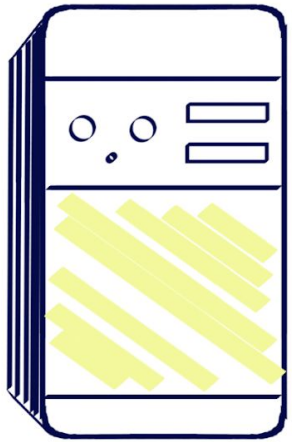
ALPN

Negotiations

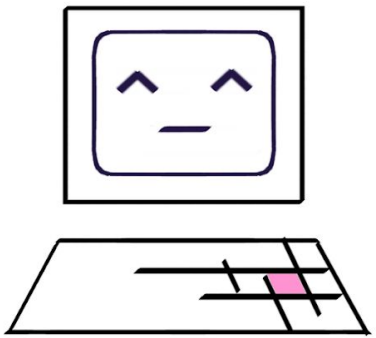

1
req: { upgrade
header }



① req: { upgrade header }

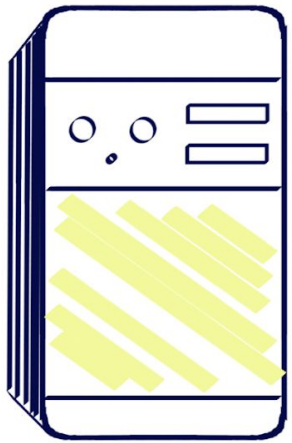


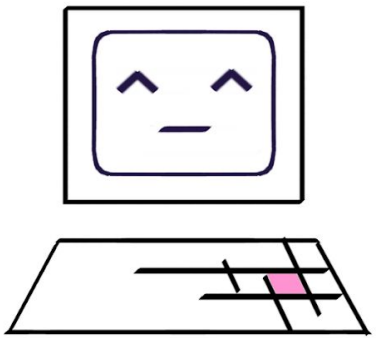
② √ can upgrade (101 response)



② can upgrade (101 response)

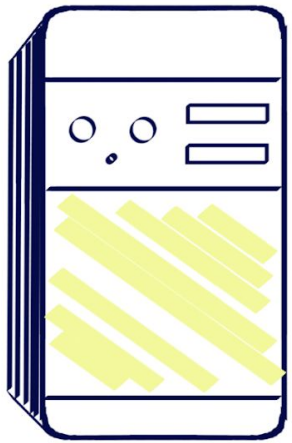
③ acknowledge upgrade

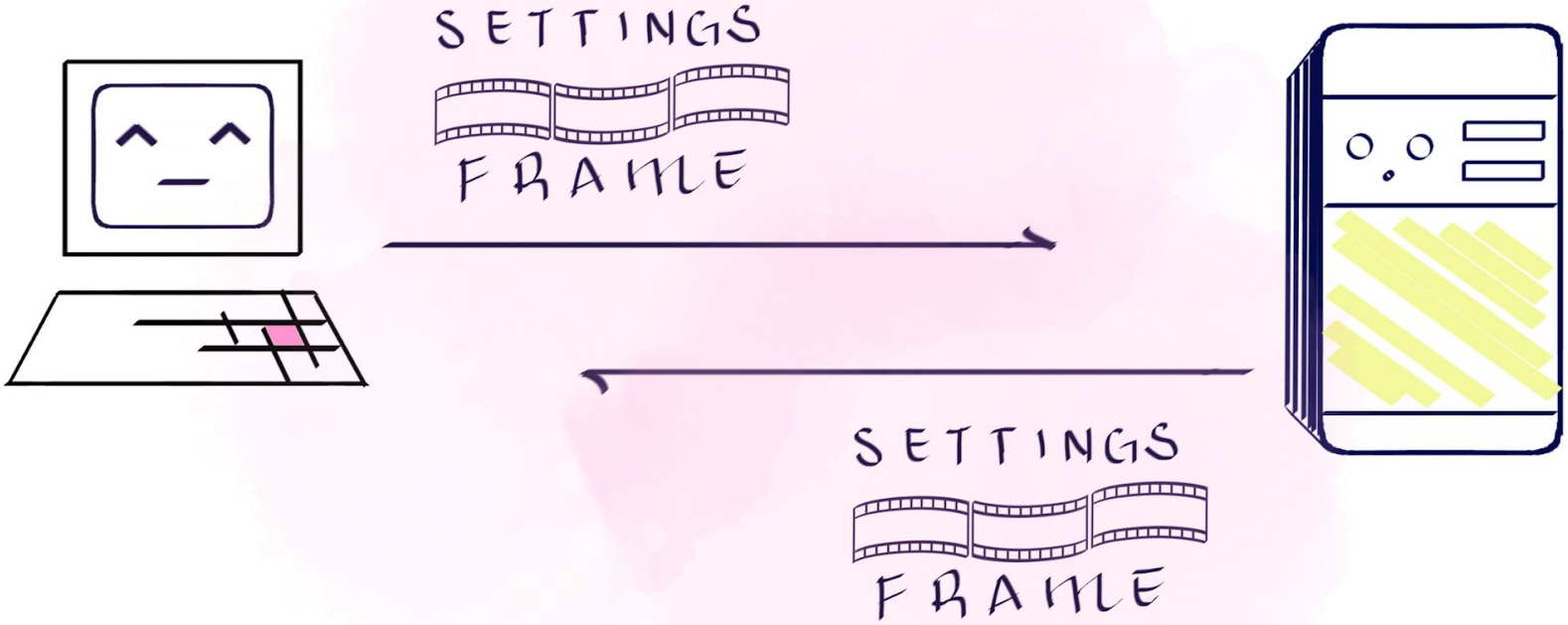




② can upgrade (101 response)

③ acknowledge upgrade




SETTINGS
FRAME
SETTINGS
FRAME

ok

ok, let's build a server

P.S. package used for local development:

tls-keygen

ACT 2. SCENE 2.

frames

frames

frames

frames

frames

ACT 2. SCENE 2.

frames

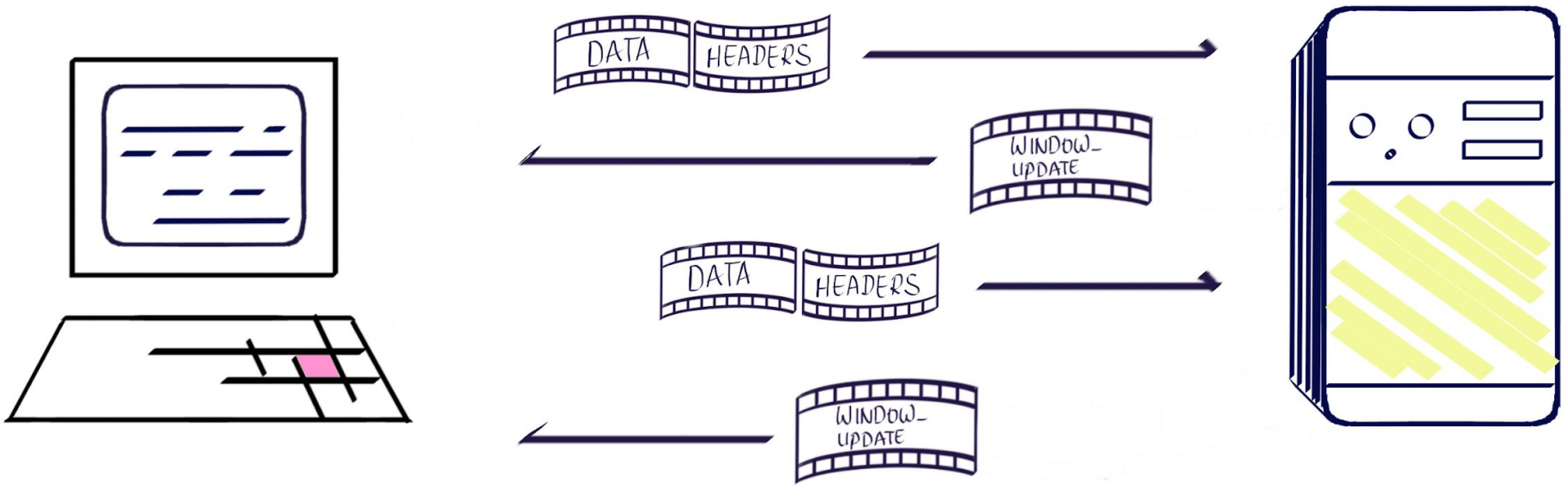
frames

frames frames

(did i mention frames?)

frames

HTTP/2 uses Frames to communicate between endpoints.


DATA
HEADERS
WINDOW_
UPDATE
DATA
HEADERS
WINDOW_
UPDATE

some frames come with

(like HEADERS, PUSH_PROMISE, and CONTINUATION)

header blocks.

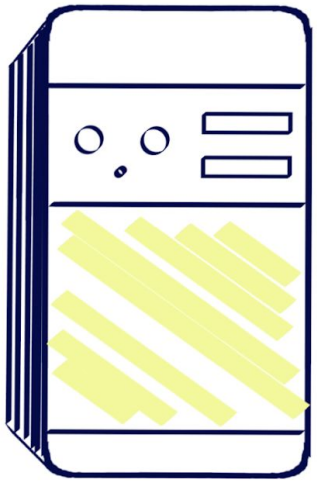
lil wins:

1. stream prioritization
2. server push
3. compressin' HTTP headers

Part of optimization of HTTP/2 comes from compressing headers.

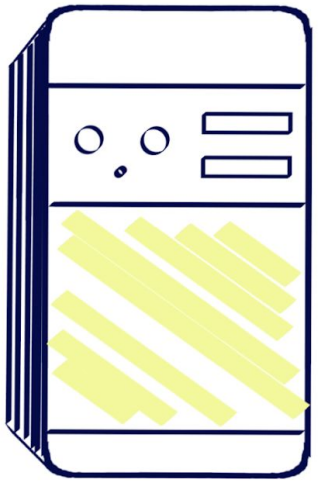
> “plain text [...] adds anywhere from 500–800 bytes of overhead per transfer, and sometimes kilobytes more if HTTP cookies are being used”

Oh, hey, it’s the google folks from WebFundamentals again ✧



- :authority: nodejs.org
- :method: GET
- :path: /api/http2.html
- :scheme: https
- accept-encoding: gzip, deflate, br

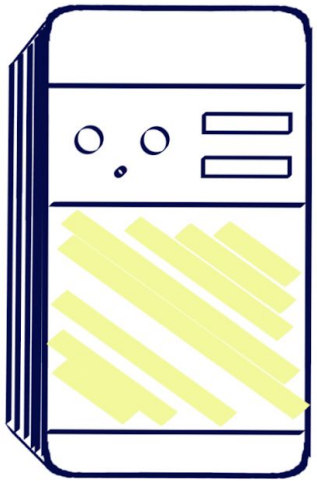
1 sent as a list



- :authority: nodejs.org
- :method: GET
- :path: /api/http2.html
- :scheme: https
- accept-encoding: gzip, deflate, br

```
:authority: nodejs.org
:method: GET
:path: /api/http2.html
:scheme: https
accept-encoding: gzip, deflate, br
```

2 Serialized header block



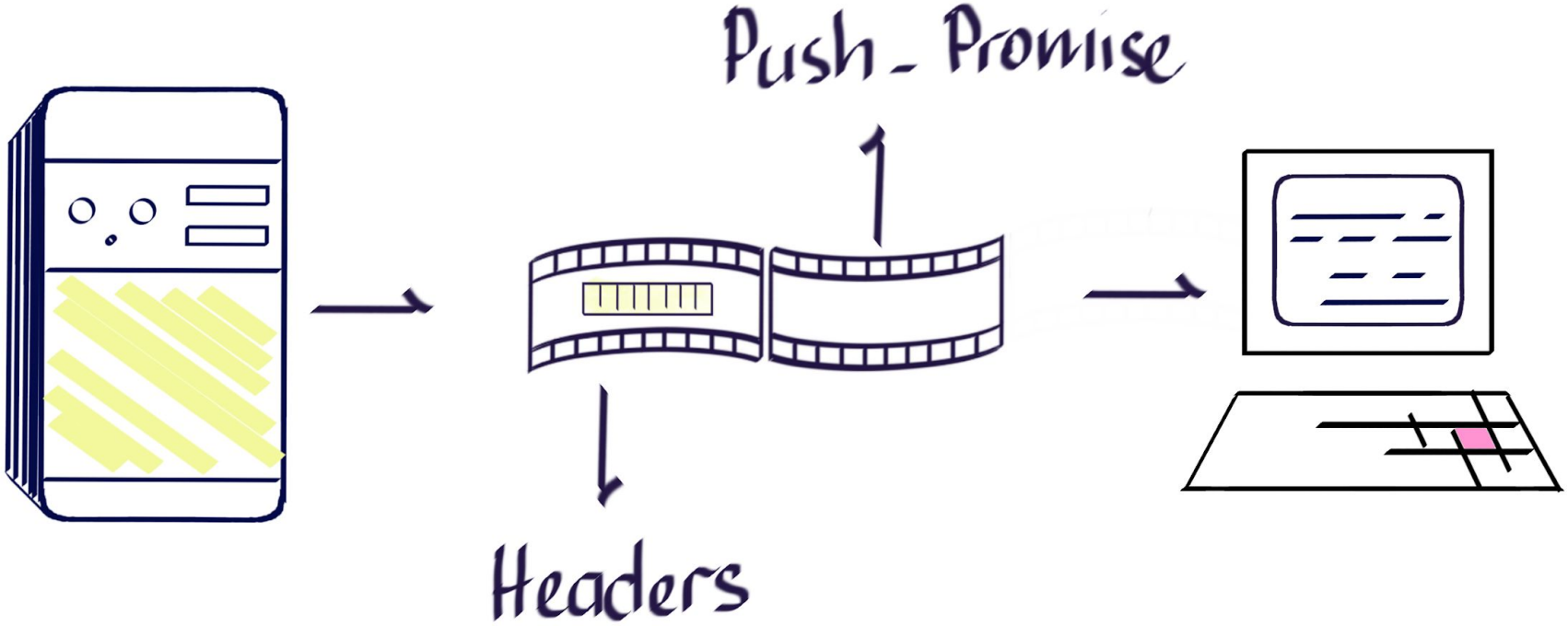
- :authority: nodejs.org
- :method: GET
- :path: /api/http2.html
- :scheme: https
- accept-encoding: gzip, deflate, br

:authority: nodejs.org
:method: GET
:path: /api/http2.html
:scheme: https
accept-encoding: gzip, deflate, br

3

octet block

huffman encoded


Push - Promise
Headers

sent with HEADERS, PUSH_PROMISE, or CONTINUATION frames.

ok, let's build a ~~server~~ client

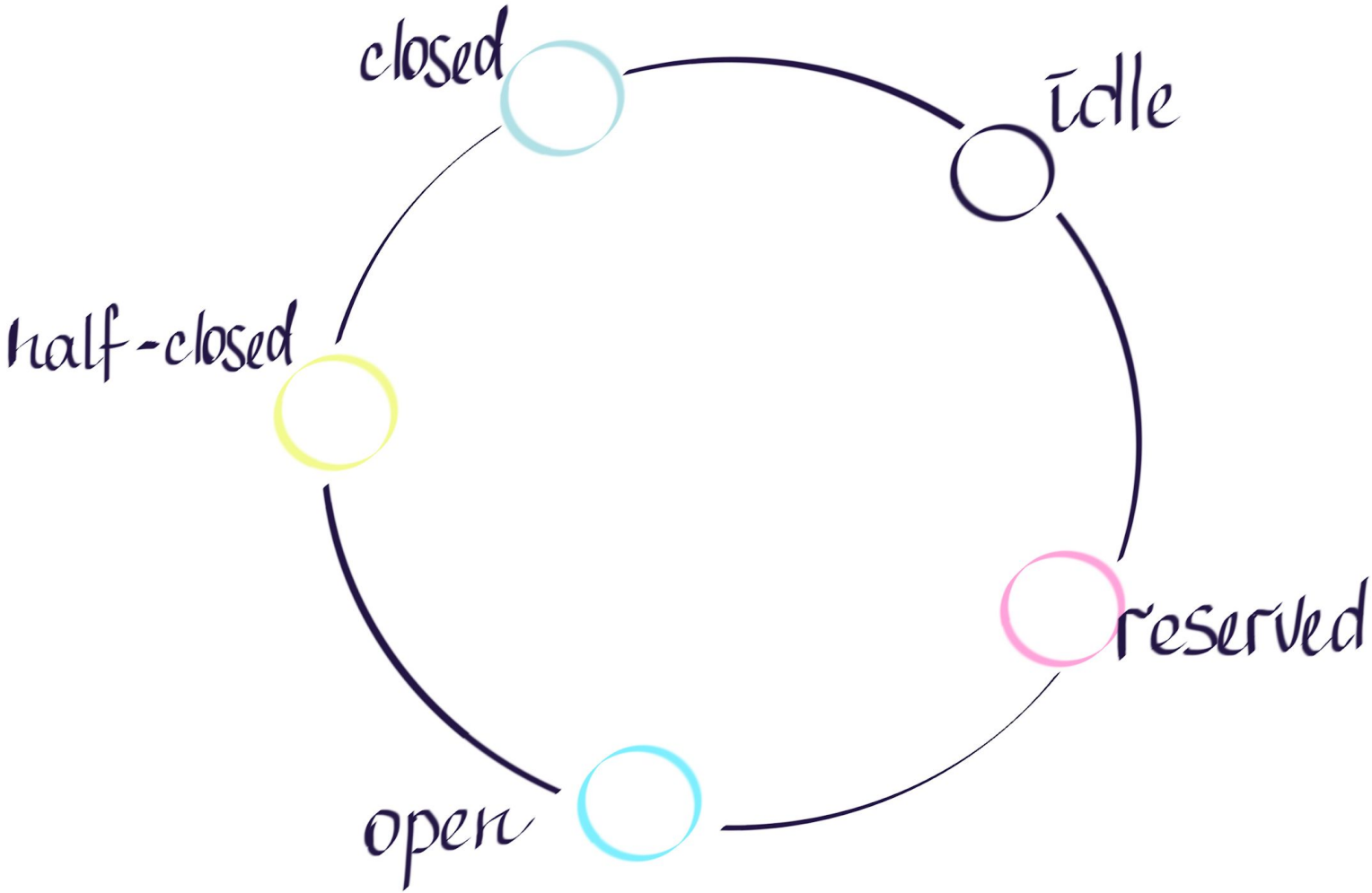
P.S. for more generic http2-requests, use http2-request

ACT 2. SCENE 3.

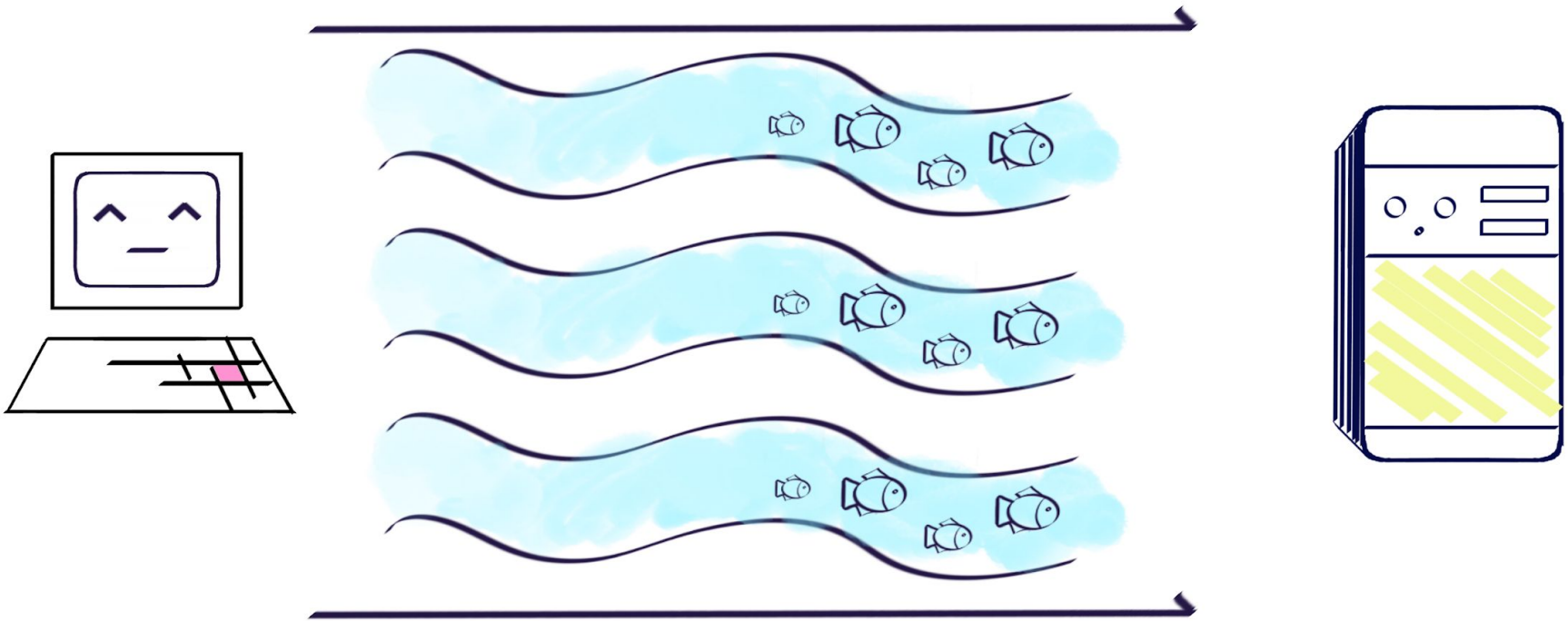
What is an HTTP/2 Stream even?

“Independent, bidirectional sequence of frames exchanged between the client and the server within an HTTP/2 connection”

Belse, et al. RFC 7540


closed
idle
half-closed
reserved
open

multiple streams exist per single connection.



each endpoint can combine frames received from multiple streams.

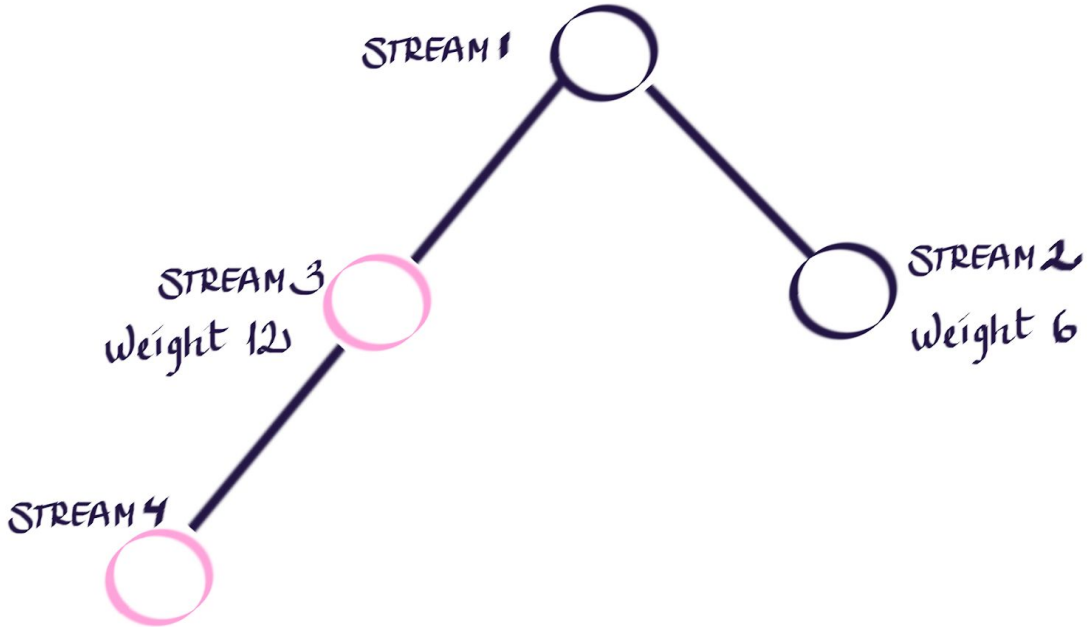
lil wins:

1. stream prioritization
2. server push
3. compressin' HTTP headers

why so ~ f ~ a ~ s ~ t ~ ?

With many streams per connection, endpoints want to prioritize resource allocation

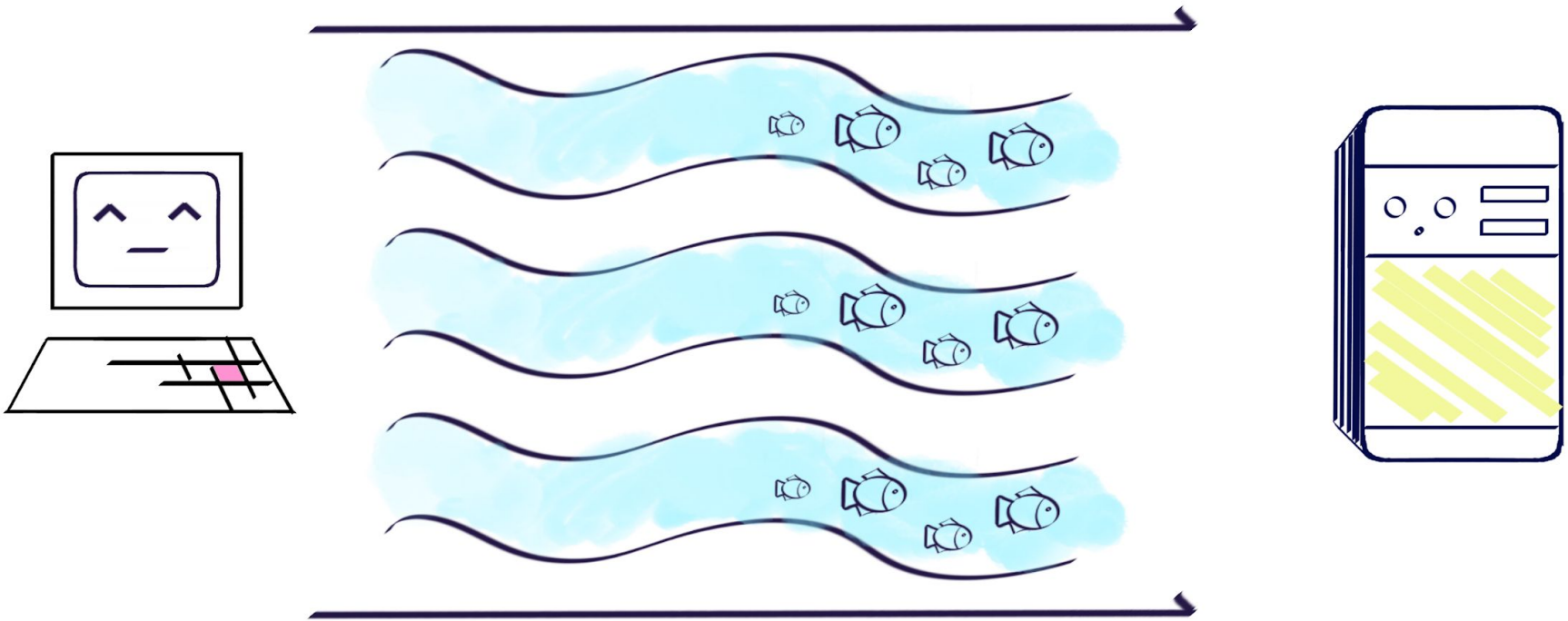
an endpoint can assign stream dependencies and stream weights


STREAM 1
STREAM 3
Weight 12
STREAM 2
Weight 6
STREAM 4

ACT 3. SCENE 1.

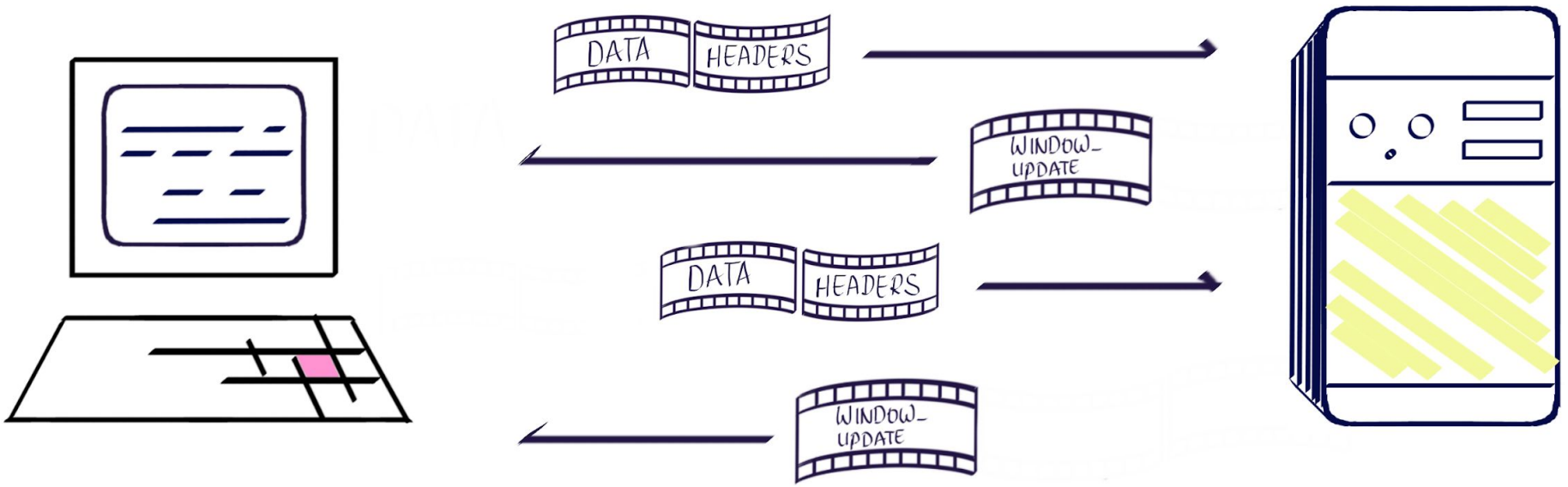
~f~l~o~w~

~c~o~n~t~r~o~l



Multiplexing streams can cause congestion in a TCP Connection

flow-control allows for the receiver to communicate how much load it can handle

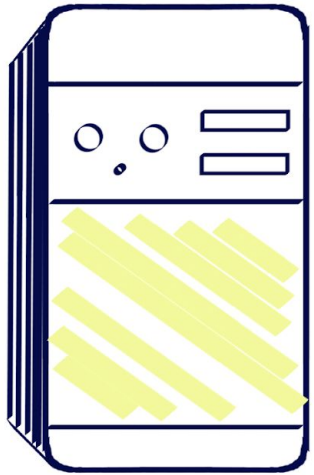

DATA
HEADERS
WINDOW_
UPDATE
DATA
HEADERS
WINDOW_
UPDATE

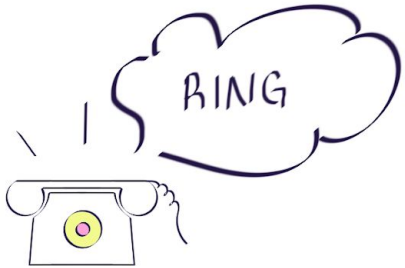
Works like a credit based system

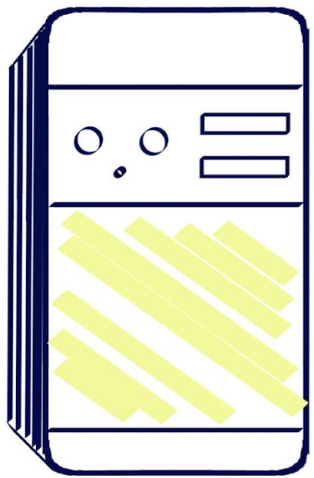
Flow-control helps prioritize streams so they don’t block

ACT 4. SCENE 1.

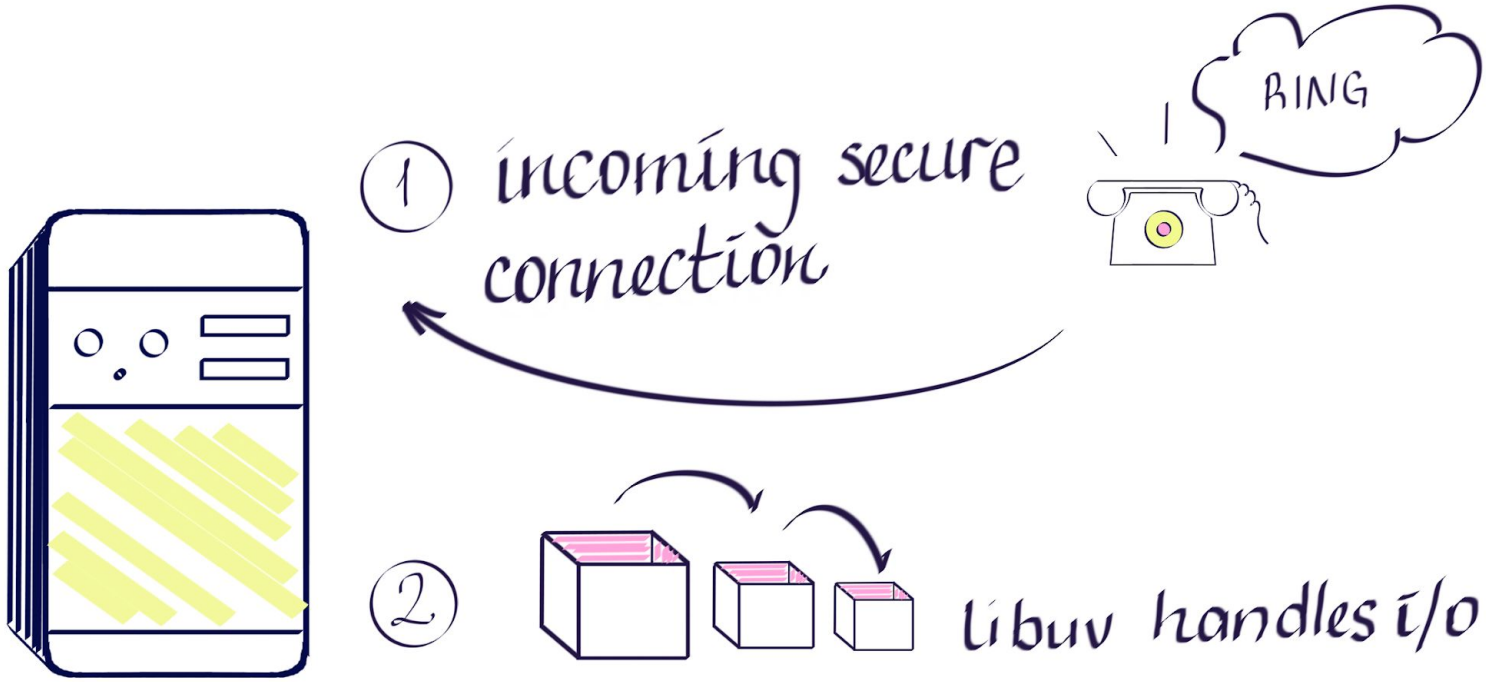
establishing a connection in node.js

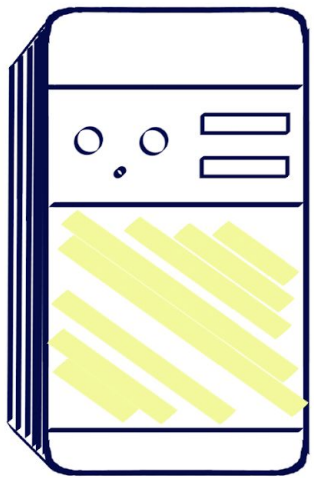



RING

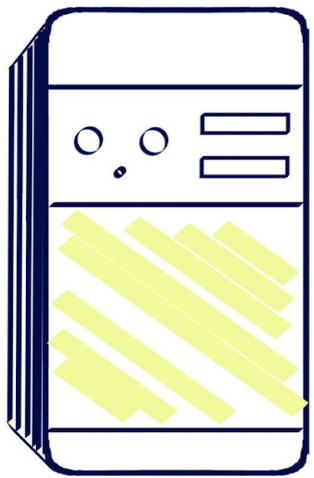


1 incoming secure connection


RING
1 incoming secure connection
2 libuv handles i/o

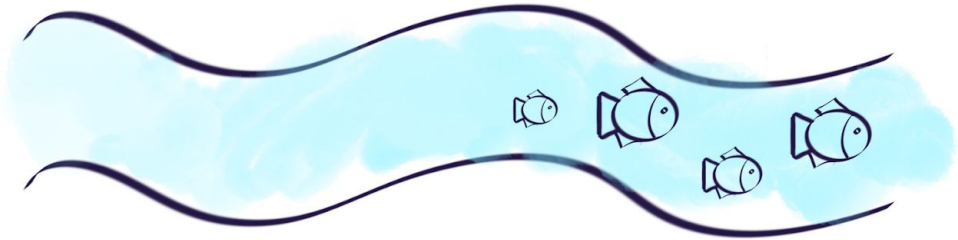


3 libssl decrypts connection

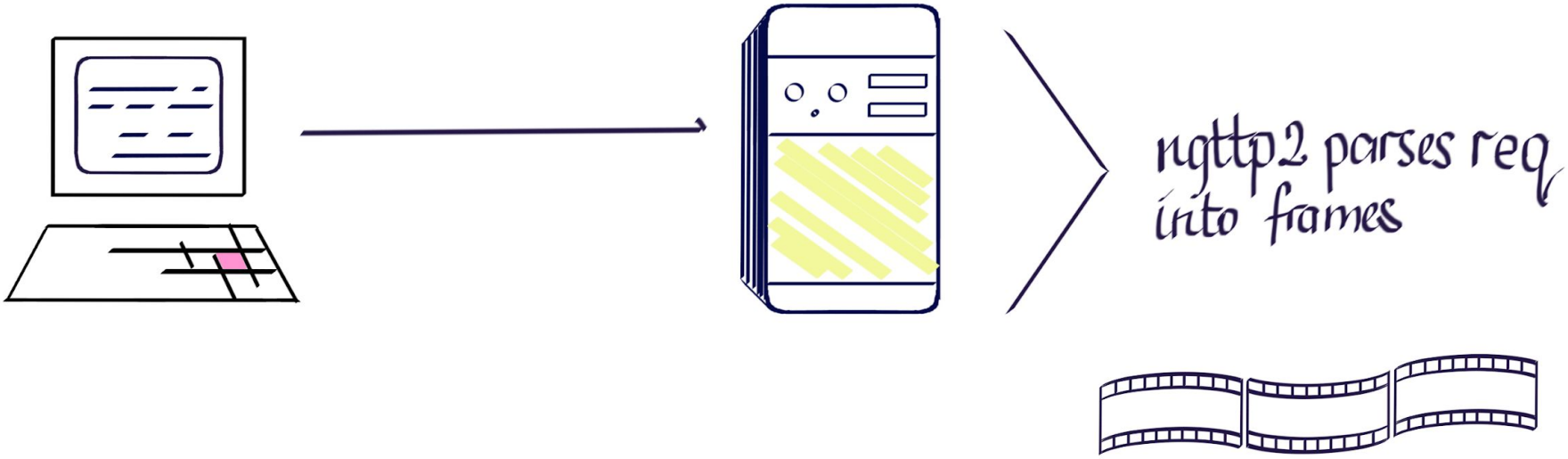


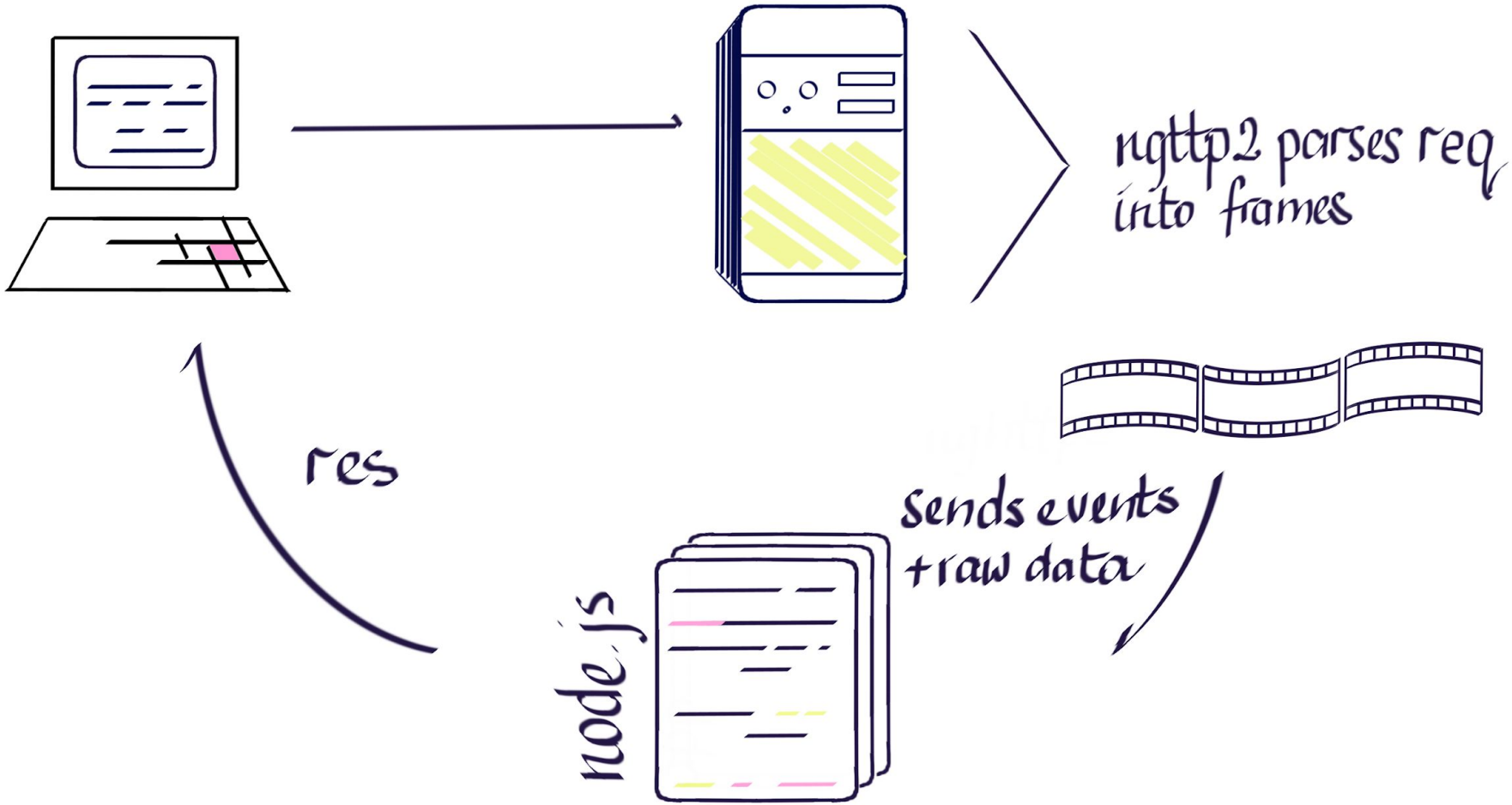
3 libssl decrypts connection

4 open a stream + session



ready for frframesframes


nghttp2 parses req
into frames


nghttp2 parses req
into frames
Sends events
+raw data
node.js
res

ACT 5. SCENE 1.

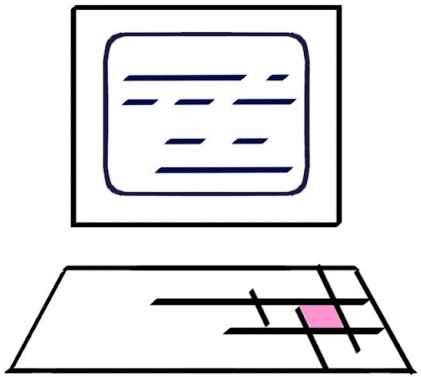
server push push.

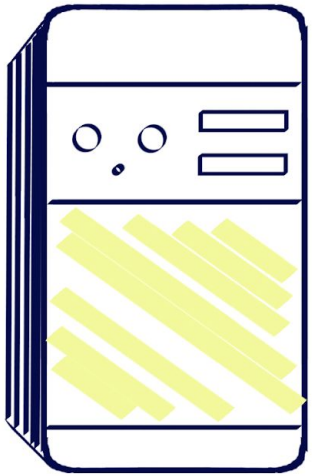
lil wins:

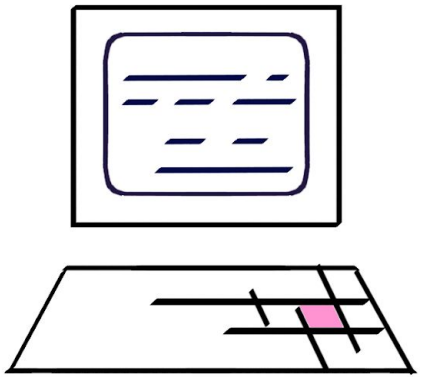
1. stream prioritization
2. server push
3. compressin' HTTP headers

need a /profile

/profile html!





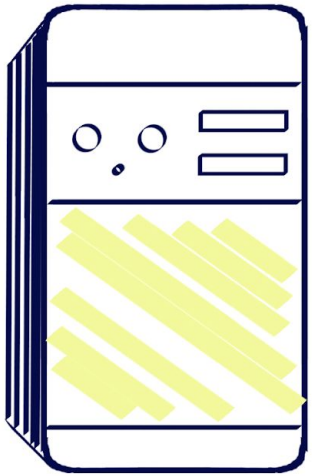


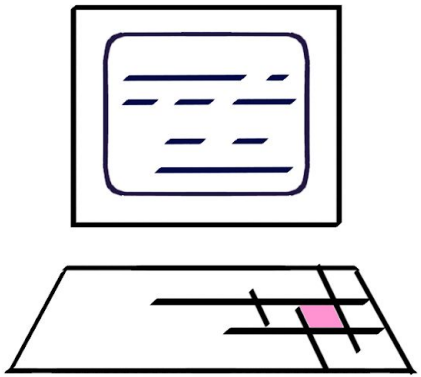
need a /profile

/profile html!

stylesheets pls!

bundle.css





need a /profile

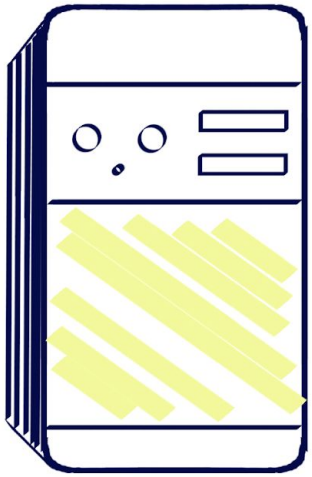
/profile html!

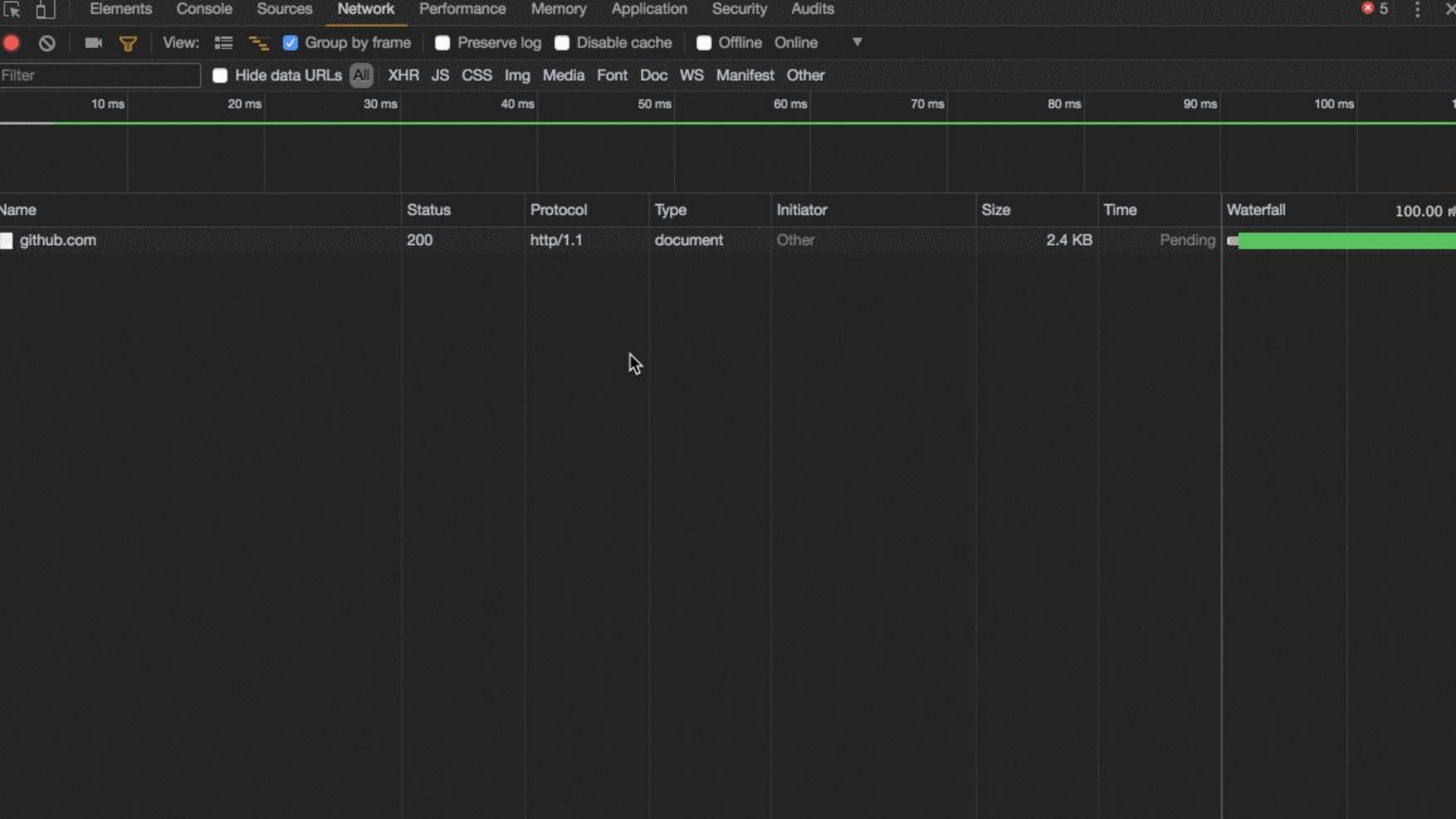
stylesheets pls!

bundle.css

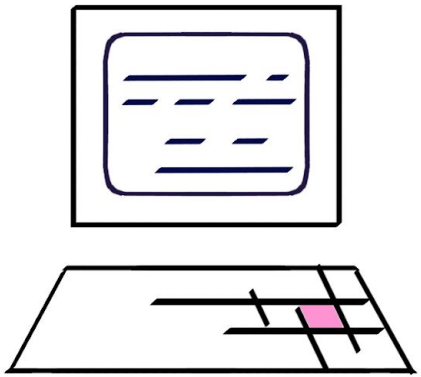
scripts also!

bundle.js




Elements Console Sources Network Performance Memory Application Security Audits
5
View:
Group by frame
Preserve log
Disable cache
Offline
Online
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms
Name Status Protocol Type Initiator Size Time Waterfall 100.00
github.com 200 http/1.1 document Other 2.4 KB Pending

server push allows the server to preemptively send data to the client before



need a /profile

/profile html!

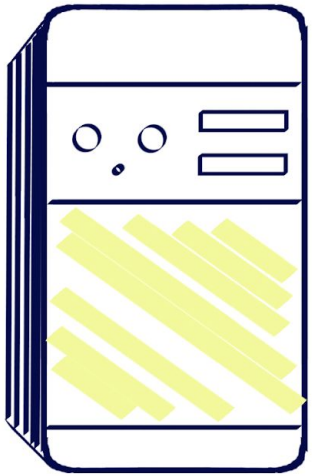
And you'll also need:

PUSH PROMISE

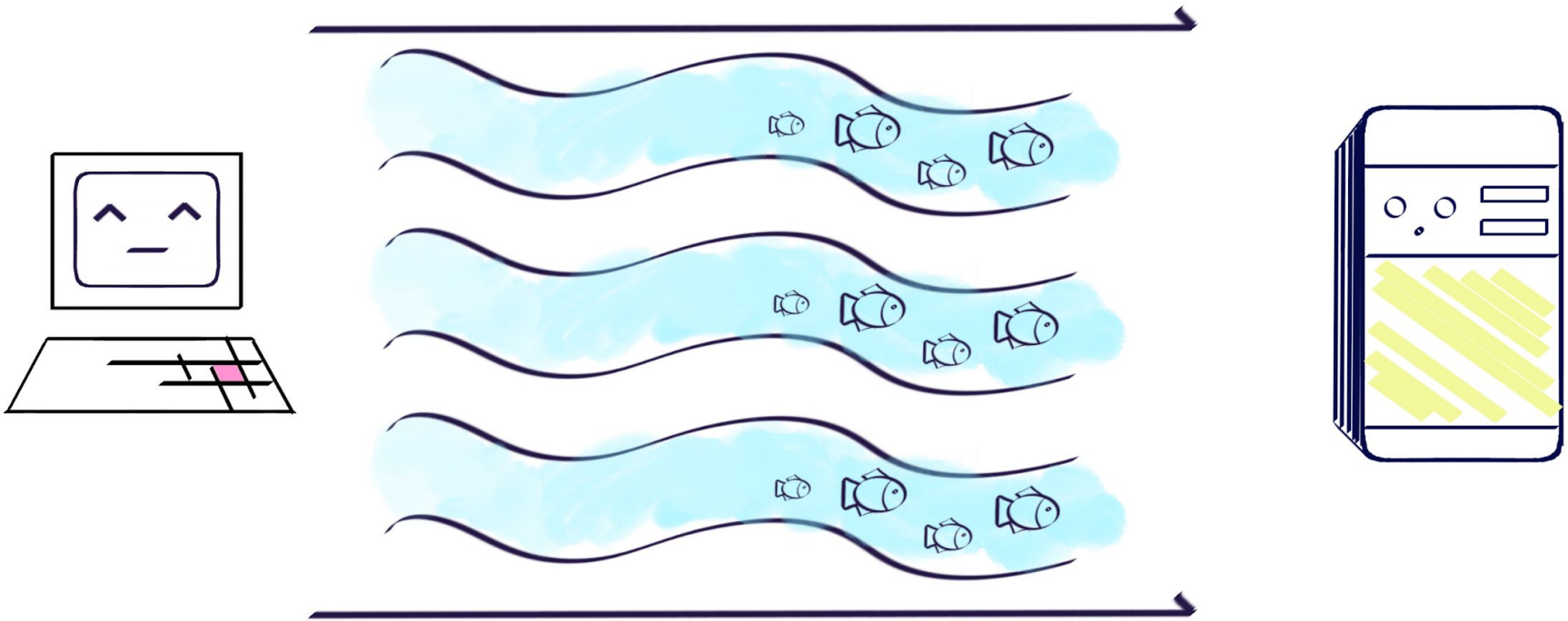
bundle.css

PUSH PROMISE

bundle.js



the tl;dr



Part of optimization of HTTP/2 comes from compressing headers.

Flow-control helps prioritize streams so they don’t block

‘push’ allows to preemtively load data to client’s cache

compat mode makes it easy to switch to HTTP/2.

compat mode

```
http2.createSecureServer({ key, cert, allowHTTP1: true  }, (req, res) => {
  res.end(`

      <body> ciao verona </body>

  `)

}).listen(8000, () => console.log(`listening on 8000`))
```

HTTP2

Stability: 1 - Experimental

The `http2` module provides an implementation of the HTTP/2 protocol. It can be accessed using:

```
const http2 = re
```

super close on not being under experimental tag

Core API

The Core API provides a low-le ... ally a ... ot ... ed fo ... with the existing HTTP/1 module API. However, the Compatibility API is.

The `http2` Core API is much more symmetric between client and server than the `http` API. For instance, most events, like `error`, `connect` and `stream`, can be emitted either by client-side code or server-side code.

le fin

say hi: 🐦 - @_Irlna || ✉ - irina@mongodb.com