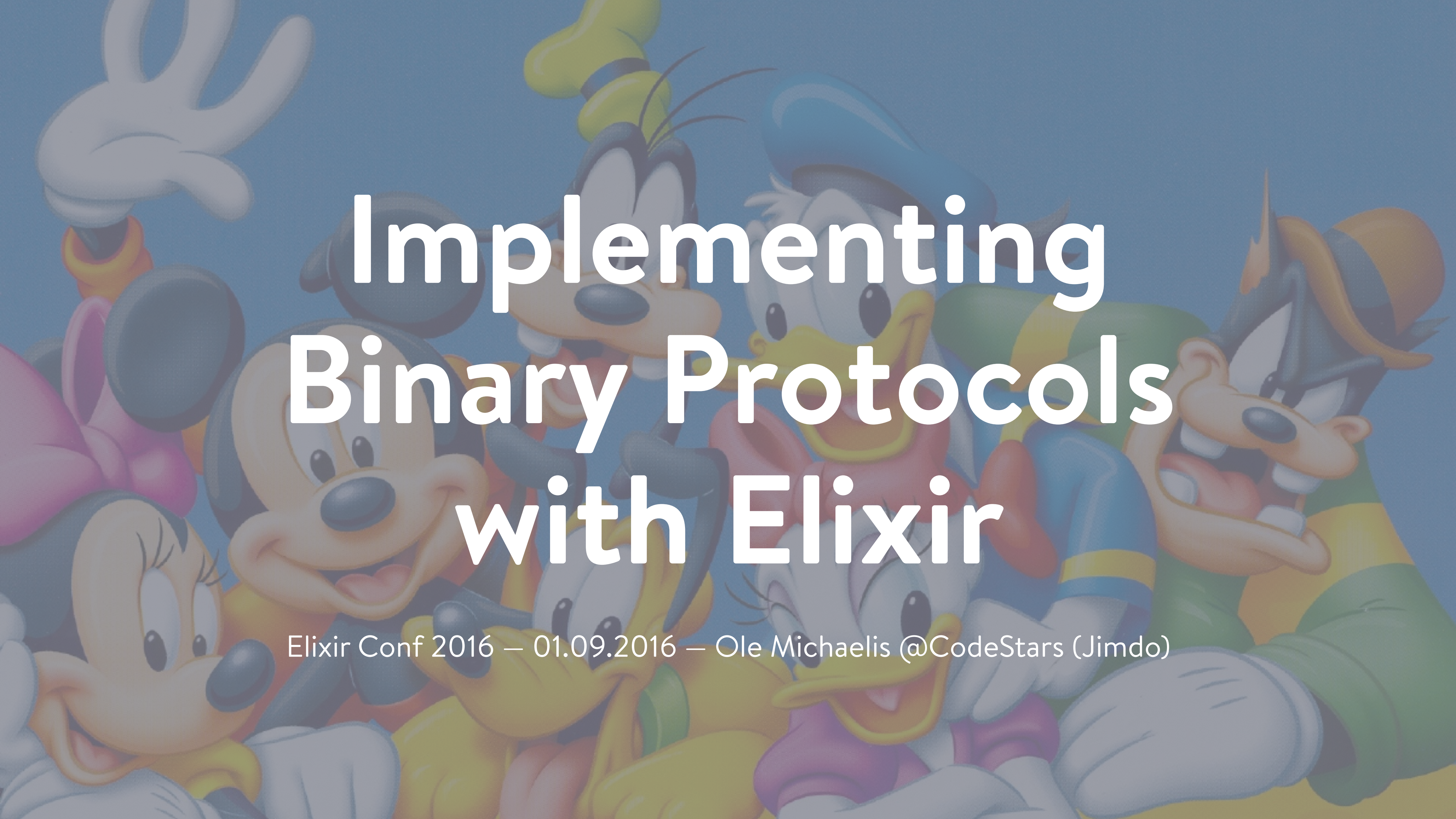




Implementing Binary Protocols with Elixir

Elixir Conf 2016 — 01.09.2016 — Ole Michaelis (@CodeStars (Jimdo))



The web is
SCREWED!

OUTGROWN

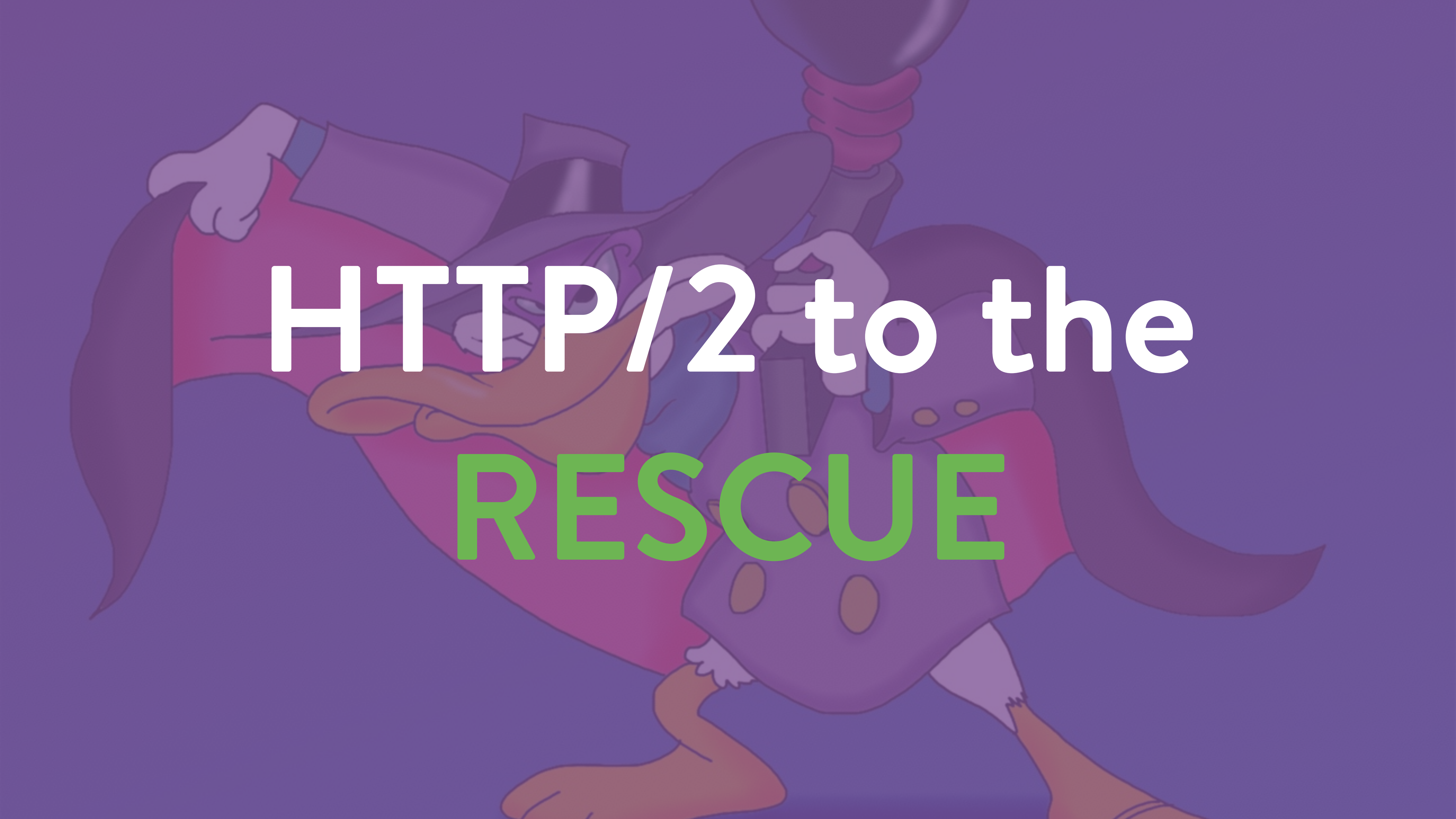
SLOW

AVG SITE 2014

101 Assets

2.60 MB

38 Connections

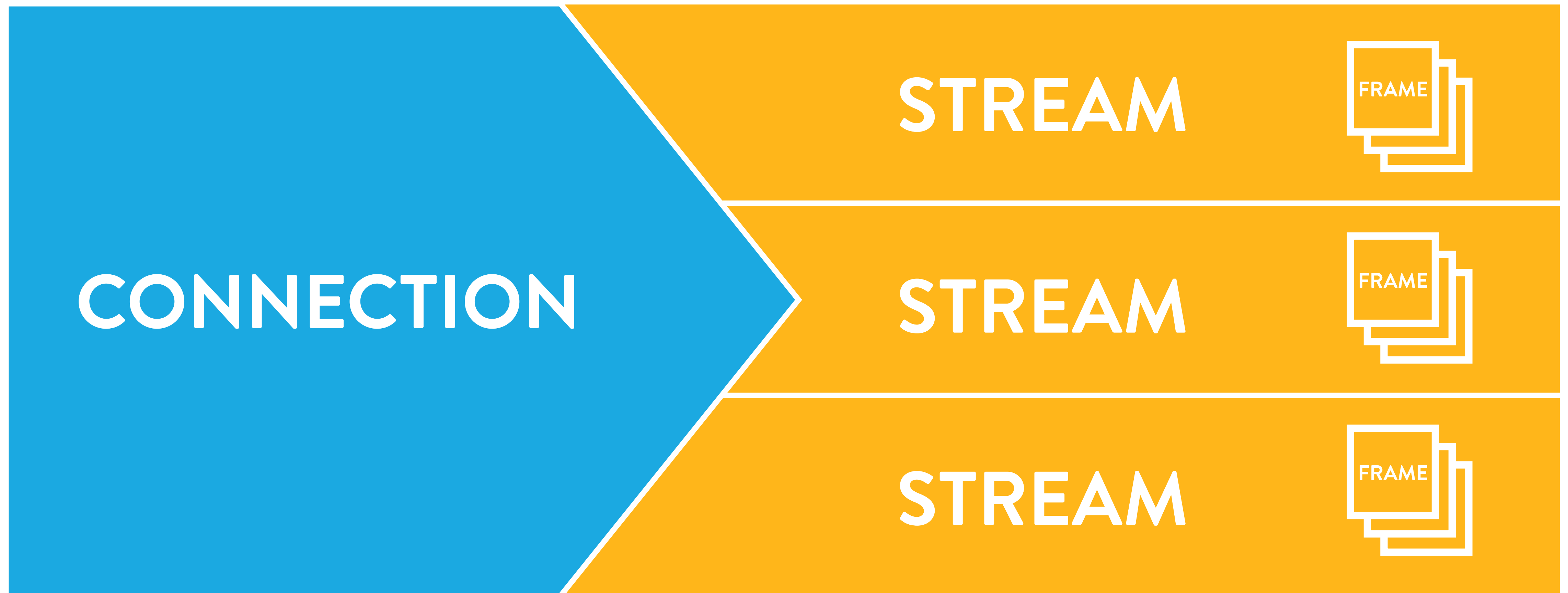


HTTP/2 to the RESCUE

COMPATIBLE

MULTIPLYED

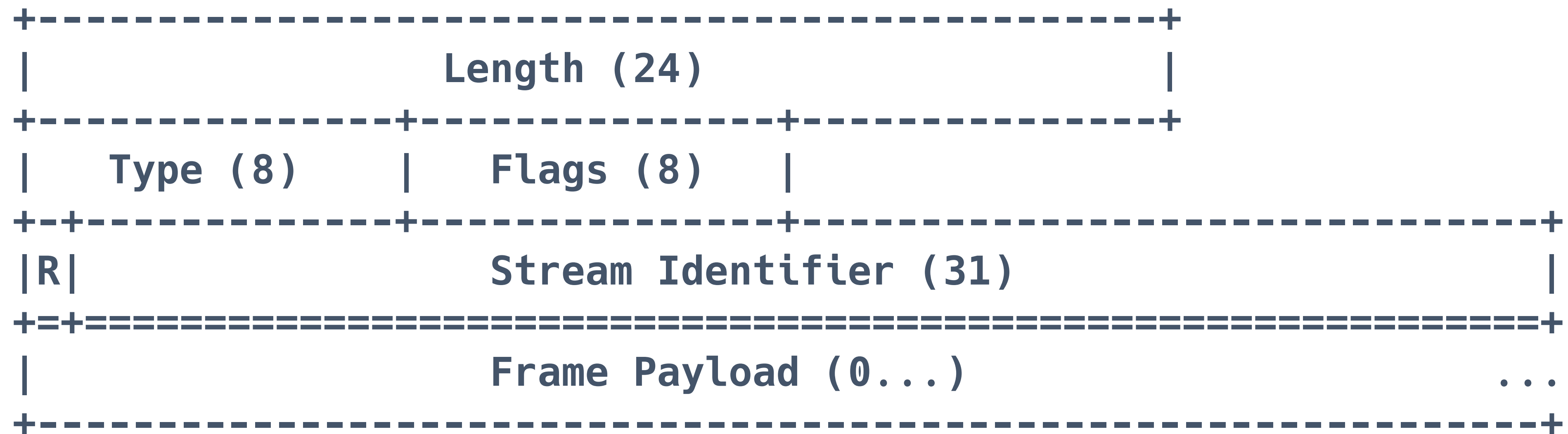
HTTP/2 CONNECTIONS



FRAMES

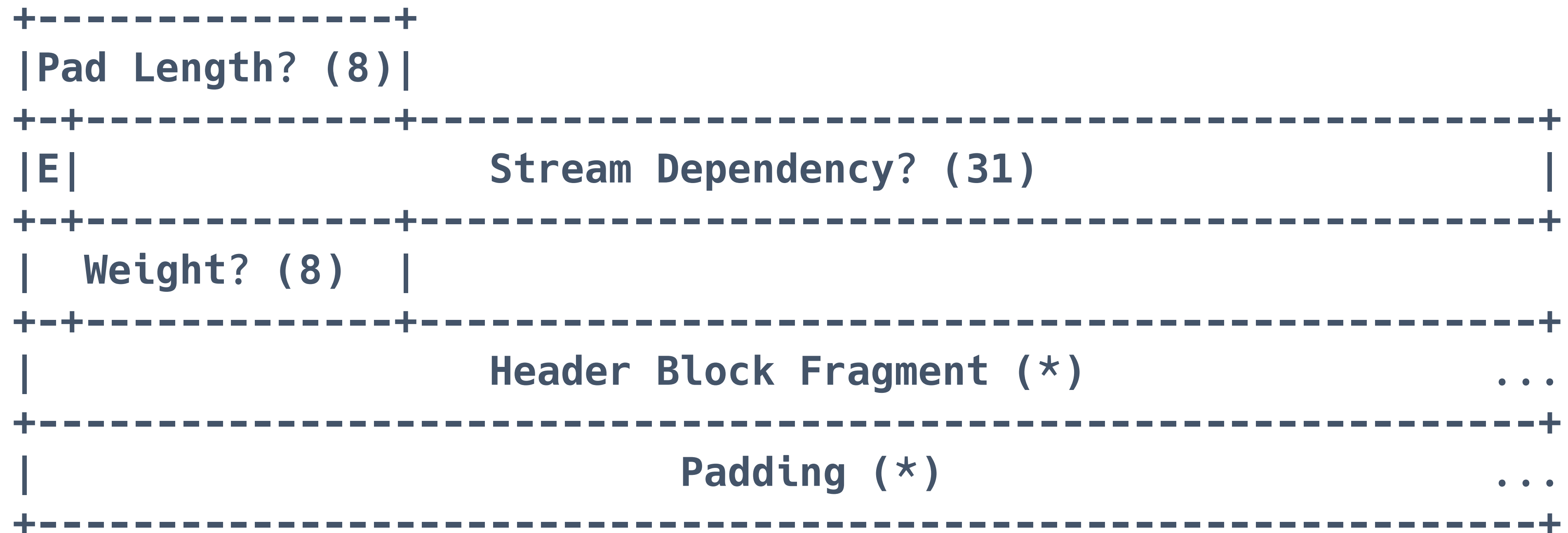
BINARY

FRAME LAYOUT

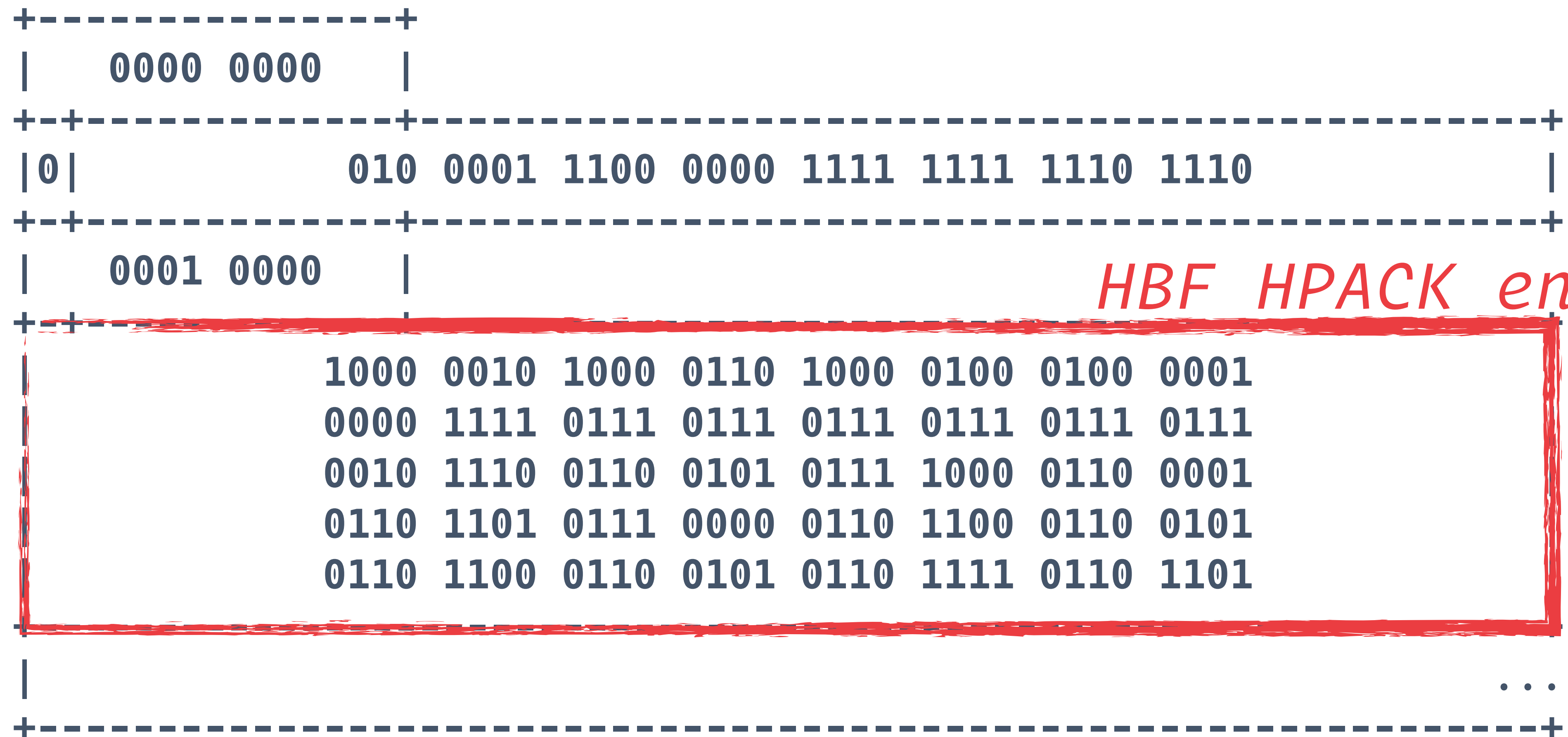


10 FRAME TYPES

HEADERS 0x1



HEADERS 0x1





RFC 7541

HPACK

HEADER COMPRESSION TABLE

request headers

:method	GET
:scheme	https
:host	jimdo.com
:path	/resource
user-agent	Mozilla/5.0
custom-hdr	some-value

static table

1	:authority	
2	:method	GET
...	...	...
51	referer	
...	...	...

dynamic table

62	user-agent	Mozilla/5.0
63	:host	jimdo.com
...	...	...

encoded headers

2	
7	
63	
19	huffman('/resource')
62	
	huffman('custom-hdr')
	huffman('some-value')

FORMAT

Indexed Header

0	1	2	3	4	5	6	7
+-----+-----+-----+-----+-----+-----+-----+							
1	Index (7+)						
+-----+-----+-----+-----+-----+-----+-----+							

Literal Header Fields /w Incremental Indexing

Indexed Name

0	1	2	3	4	5	6	7
+-----+							
0	1	Index (6+)					
+-----+							
H	Value Length (7+)						
+-----+							
Value String (Length octets)							
+-----+							

New Name

0	1	2	3	4	5	6	7
0	1	0					
H	Name Length (7+)						
Name String (Length octets)							
H	Value Length (7+)						
Value String (Length octets)							

Literal Header Fields w/o Indexing

Indexed Name

0	1	2	3	4	5	6	7
+-----+-----+-----+-----+-----+-----+-----+-----+							
0	0	0	0	Index (4+)			
+-----+-----+-----+-----+-----+-----+-----+-----+							
H	Value Length (7+)						
+-----+-----+-----+-----+-----+-----+-----+-----+							
Value String (Length octets)							
+-----+-----+-----+-----+-----+-----+-----+-----+							

New Name

0	1	2	3	4	5	6	7
+-----+-----+-----+-----+-----+-----+-----+-----+							
0	0	0	0			0	
+-----+-----+-----+-----+-----+-----+-----+-----+							
H	Name Length (7+)						
+-----+-----+-----+-----+-----+-----+-----+-----+							
Name String (Length octets)							
+-----+-----+-----+-----+-----+-----+-----+-----+							
H	Value Length (7+)						
+-----+-----+-----+-----+-----+-----+-----+-----+							
Value String (Length octets)							
+-----+-----+-----+-----+-----+-----+-----+-----+							

Literal Header Fields never Indexing

Indexed Name

0	1	2	3	4	5	6	7
+-----+-----+-----+-----+-----+-----+-----+-----+							
0	0	0	1	Index (4+)			
+-----+-----+-----+-----+-----+-----+-----+-----+							
H	Value Length (7+)						
+-----+-----+-----+-----+-----+-----+-----+-----+							
Value String (Length octets)							
+-----+-----+-----+-----+-----+-----+-----+-----+							

New Name

0	1	2	3	4	5	6	7
0	0	0	1	0			
Name Length (7+)							
Name String (Length octets)							
Value Length (7+)							
Value String (Length octets)							



Implementing HPACK

DECODE



DIGRESSION:

Binaries & Strings

STRINGS ARE BINARIES

👉 ~ iex

Erlang/OTP 19 [erts-8.0.2] [source] [64-bit] [smp:4:4]
[async-threads:10] [hipe] [kernel-poll:false] [dtrace]

Interactive Elixir (1.3.2) - press Ctrl+C to exit (type
h() ENTER for help)

iex(1)> ?e

101

iex(2)> << 108::utf8 >>

"1"

iex(3)> << 108 >>

"1"

iex(4)>

```
iex(4)> string = "José"  
"José"
```

```
iex(5)> string <> << 0 >>
```

```
<<74, 111, 115, 195, 169, 0>>
```

```
iex(6)> "🎉 elixir conf" <> << 0 >>
```

```
<<240, 159, 142, 137, 32, 101, 108, 105, 120, 105, 114,  
  32, 99, 111, 110, 102, 0>>
```

```
iex(7)>
```

```
defmodule HPack do

  @type header :: {String.t, String.t}

  @spec decode(String.t, pid) :: [header]
  def decode(hbf, table)
    ...
  end

  ...
end
```

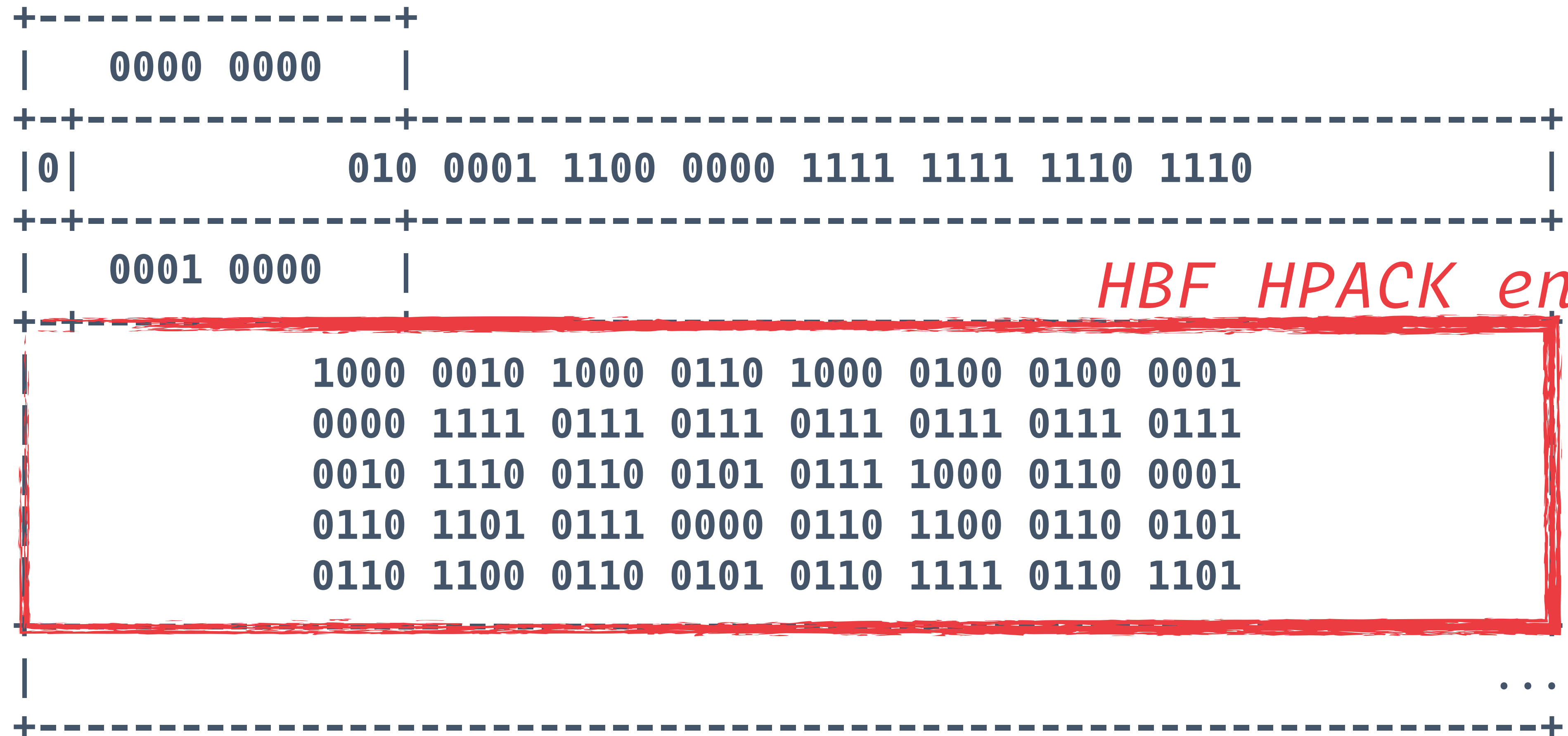
```
defmodule HPack do

  @type header :: {String.t, String.t}

  @spec decode(String.t, pid) :: [header]
  def decode(hbf, table)
    parse(hbf, [], table)
  end

  ...
end
```

HEADERS 0x1



```

#      0      1      2      3      4      5      6      7
#  +---+---+---+---+---+---+---+---+
#  | 1 |           Index (7+)           |
#  +---+---+---+---+---+---+---+---+

```

Figure 5: Indexed Header Field

```

defp parse(<< 1::size(1), rest::bitstring >>, hdrs, tbl) do
  # rest => 000 0010 1000 0110... | [] | #PID<0.89.0>
  { idx, rest } = parse_int7(rest) # {2, << 0b100001... >>}
  hdr = Table.lookup(idx, tbl) # {":method", "GET"}
  parse(rest, [ hdr | hdrs ], tbl)
end

```

```

#   0   1   2   3   4   5   6   7
# +---+---+---+---+---+---+---+
# | 1 |           Index (7+)           |
# +---+---+---+---+---+---+---+

```

Figure 5: Indexed Header Field

```

defp parse(<< 1::size(1), rest::bitstring >>, hdrs, tbl) do
  # rest => 000 0110 1000 0100 0100 0001 0000 1111 0111...
  # hdrs => [{"method", "GET"}]
  { idx, rest } = parse_int7(rest) # {6, << 0b100001... >>}
  hdr = Table.lookup(idx, tbl) # {":scheme", "http"}
  parse(rest, [ hdr | hdrs ], tbl)
end

```

```

#      0      1      2      3      4      5      6      7
# +---+---+---+---+---+---+---+---+
# | 1 |           Index (7+)           |
# +---+---+---+---+---+---+---+

```

Figure 5: Indexed Header Field

```

defp parse(<< 1::size(1), rest::bitstring >>, hdrs, tbl) do
  # rest => 000 0100 0100 0001 0000 1111 0111 0111 0111...
  # hdrs => [{":scheme", "http"}, {":method", "GET"}]
  {idx, rest} = parse_int7(rest) # {4, << 0b01000001... >>}
  hdr = Table.lookup(idx, tbl) # {":path", "/" }
  parse(rest, [hdr | hdrs], tbl)
end

```

```

#   0   1   2   3   4   5   6   7
# +---+---+---+---+---+---+---+
# | 0 | 1 |           Index (6+)       |
# +---+---+---+---+---+---+---+
# | H |           Value Length (7+)    |
# +---+---+---+---+---+---+---+
# | Value String (Length octets)      |
# +---+---+---+---+---+---+---+

```

Figure 6: Literal Header Field /w Incremental Indexing – Indexed Name

```

defp parse(<< 0::1, 1::1, rest::bitstring >>, hdrs, tbl) do
  # rest => 00 0001 0000 1111... | [{"path", "/"}, {"scheme",
  { idx, rest } = parse_int6(rest) # {1, << 0b100001... >>}
  { value, rest } = parse_string(rest) # {"www.example.com",
  { header, _ } = Table.lookup(idx, tbl) # {"authority", ""} >>
  Table.add({header, value}, tbl)
  parse(rest, [ {header, value} | hdrs], tbl)
end

```

```
defmodule HPack do
```

```
  defp parse(<< >>, hdrs, _),  
    do: Enum.reverse(hdrs)  
  # => [{":method", "GET"},  
  #      {":scheme", "http"},  
  #      {":path", "/"},  
  #      {":authority", "www.example.com"}]
```

```
end
```

HEADERS 0x1

```
+-----+
| 0000 0000 |
+-----+
|0|          010 0001 1100 0000 1111 1111 1110 1110 |
+-----+
| 0001 0000 |
+-----+
|          :method => GET
|          1000 0010 1000 0110 1000 0100 0100 0001 |
|          0000 1111 0111 0111 0111 0111 0111 0111 |
|          0010 1110 0110 0101 0111 1000 0110 0001 |
|          0110 1101 0111 0000 0110 1100 0110 0101 |
|          0110 1100 0110 0101 0110 1111 0110 1101 |
+-----+
|          ...
+-----+
```

HEADERS 0x1

```
+-----+
| 0000 0000 |
+-----+
|0|          010 0001 1100 0000 1111 1111 1110 1110 |
+-----+
| 0001 0000 |
+-----+
|          :scheme => http
|          1000 0010 1000 0110 1000 0100 0100 0001 |
|          0000 1111 0111 0111 0111 0111 0111 0111 |
|          0010 1110 0110 0101 0111 1000 0110 0001 |
|          0110 1101 0111 0000 0110 1100 0110 0101 |
|          0110 1100 0110 0101 0110 1111 0110 1101 |
+-----+
|                                     ...
+-----+
```

HEADERS 0x1

```
+-----+
| 0000 0000 |
+-----+
|0|          010 0001 1100 0000 1111 1111 1110 1110|
+-----+
| 0001 0000 |
+-----+
|          1000 0010 1000 0110 1000 0100 0100 0001|
|          0000 1111 0111 0111 0111 0111 0111 0111|
|          0010 1110 0110 0101 0111 1000 0110 0001|
|          0110 1101 0111 0000 0110 1100 0110 0101|
|          0110 1100 0110 0101 0110 1111 0110 1101|
+-----+
|                                     ...|
+-----+
```

:path => /

HEADERS 0x1

```
+-----+
| 0000 0000 |
+-----+
|0|          010 0001 1100 0000 1111 1111 1110 1110 |
+-----+
| 0001 0000 |
+-----+
|          1000 0010 1000 0110 1000 0100 0100 0001 |
|          0000 1111 0111 0111 0111 0111 0111 0111 |
|          0010 1110 0110 0101 0111 1000 0110 0001 |
|          0110 1101 0111 0000 0110 1100 0110 0101 |
|          0110 1100 0110 0101 0110 1111 0110 1101 |
+-----+
|                                                                ...
+-----+
```

:authority =>

HEADERS 0x1

```
+-----+
| 0000 0000 |
+-----+
|0|          010 0001 1100 0000 1111 1111 1110 1110 |
+-----+
| 0001 0000 |
+-----+
|          :authority => www.example.com
+-----+
|          1000 0010 1000 0110 1000 0100 0100 0001
|          0000 1111 0111 0111 0111 0111 0111 0111
|          0010 1110 0110 0101 0111 1000 0110 0001
|          0110 1101 0111 0000 0110 1100 0110 0101
|          0110 1100 0110 0101 0110 1111 0110 1101
+-----+
|          ...
+-----+
```

STRINGS

```

# +---+---+-----+
# | H |      Value Length (7+)      |
# +---+-----+
# | Value String (Length octets) |
# +-----+

```

```

defp parse_string(<< 0::1, rest::bitstring >>) do
  # rest => 011 0000 01100101 01101100 01101001 01111000 01101001
  { len, rest } = parse_int7(rest) # {6, << 0b01100101... >>}
  << val::binary-size(len), rest::binary >> = rest # "elixir"
  { val, rest }
end

```

```

# +---+---+-----+
# | H |      Value Length (7+)      |
# +---+-----+
# | Value String (Length octets) |
# +-----+

```

```

defp parse_string(<< 1::1, rest::bitstring >>) do
  # rest => 011 0000 01100101 01101100 01101001 01111000 01101001
  { len, rest } = parse_int7(rest) # {6, << 0b01100101... >>}
  << val::binary-size(len), rest::binary >> = rest # "elixir"
  { Huffman.decode(val), rest }
end

```

INTEGERS

```

#      0      1      2      3      4      5      6      7
#  +---+---+---+---+---+---+---+---+
#  | 1 |           Index (7+)           |
#  +---+---+---+---+---+---+---+---+
#  Figure 5: Indexed Header Field

```


1337

0	1	2	3	4	5	6	7
+---+---+---+---+---+---+---+							
X	X	X	1	1	1	1	1
1	0	0	1	1	0	1	0
0	0	0	0	1	0	1	0
+---+---+---+---+---+---+---+							

- $Prefix = 31, I = 1306$
- $1306 \geq 128, \text{ encode}(154), I = 1306 / 128$
- $10 < 128, \text{ encode}(10), \text{ done}$

```
defp parse_int4(<< 0b1111::4, rest::binary >>),  
  do: parse_big_int(rest, 15, 0)
```

```
defp parse_int4(<< int::4, rest::binary >>),  
  do: { int, rest }
```

```
defp parse_big_int(<< 0::1, val::7, rest::binary >>, i, m),  
  do: { i + (val <<< m), rest }
```

```
defp parse_big_int(<< 1::1, val::7, rest::binary >>, i, m),  
  do: parse_big_int(rest, i + (val <<< m), m + 7)
```

ENCODE

```
defmodule HPack do
```

```
  def encode(hdrs, tbl) when is_list(hdrs),  
    do: encode(hdrs, << >>, tbl)
```

```
  defp encode([ {k, v} | hdrs ], hbf, tbl) do  
    partial = case Table.find(k, v, tbl) do  
      { :full, idx } -> encode_indexed(idx)  
      { :key, idx } -> encode_literal_indexed(idx, v, tbl)  
      { :none } -> encode_literal_not_indexed(k, v, tbl)  
    end
```

```
    encode(hdr, hbf <> partial, tbl)
```

```
  end
```

```
end
```

```

#      0      1      2      3      4      5      6      7
# +---+---+---+---+---+---+---+---+
# | 1 |           Index (7+)           |
# +---+---+---+---+---+---+---+---+

```

Figure 5: Indexed Header Field

```

defp encode_indexed(index),
  do: << 1::1, encode_int7(index)::bitstring >>

```

```

#  0    1    2    3    4    5    6    7
#  +---+---+---+---+---+---+---+
#  | 0 | 1 |           Index (6+)       |
#  +---+---+---+---+---+---+---+
#  | H |           Value Length (7+)    |
#  +---+---+---+---+---+---+---+
#  | Value String (Length octets)      |
#  +---+---+---+---+---+---+---+

```

Figure 6: Literal Header Field /w Incremental Indexing – Indexed Name

```

defp encode_literal_indexed(idx, val, tbl) do
  { name, _ } = Table.lookup(idx, tbl)
  Table.add({name, val}, tbl)
  << 0::1, 1::1, encode_int6(idx)::bitstring,
      encode_string(val)::binary >>
end

```

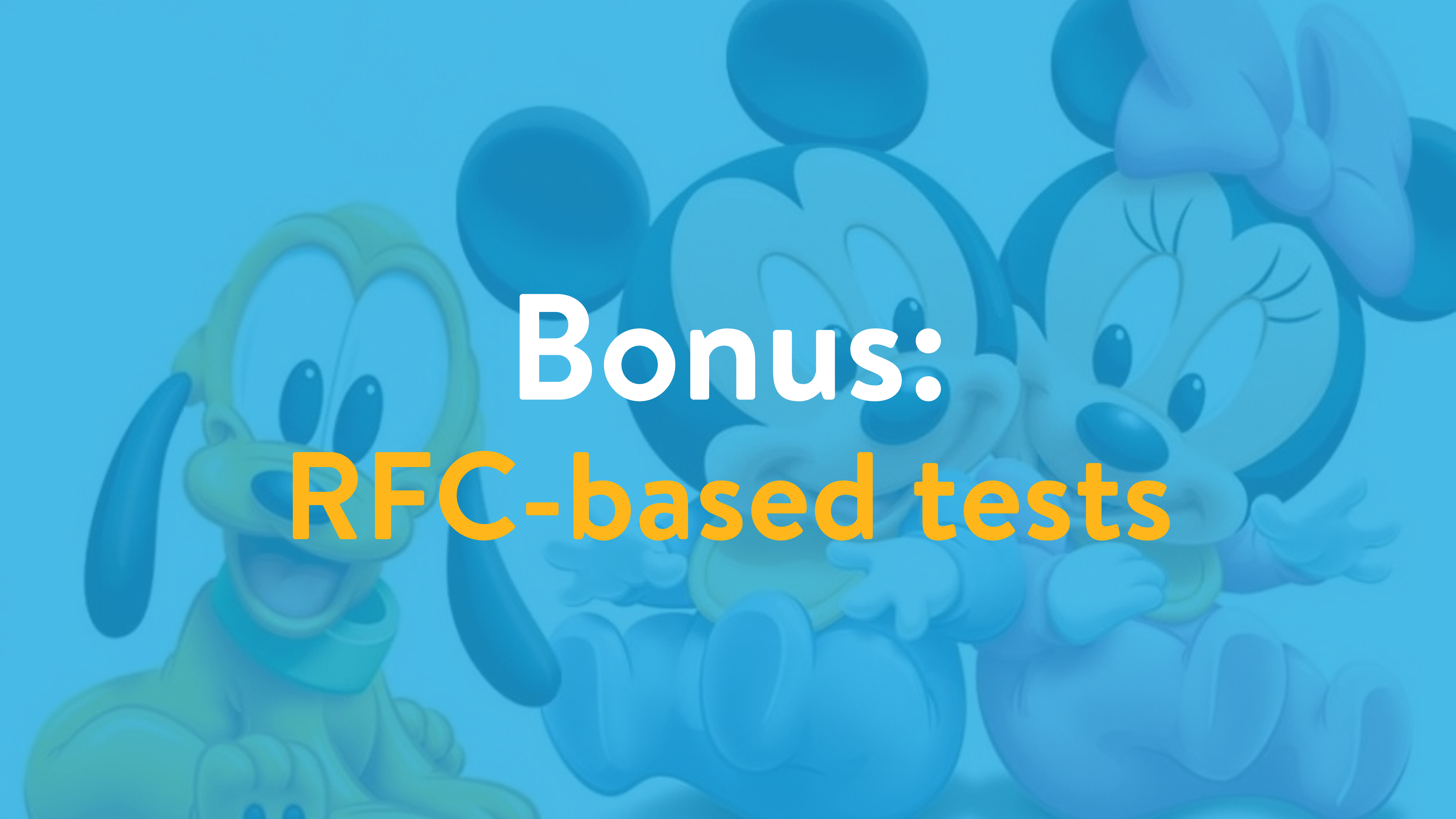
STRINGS

```
# +---+---+-----+
# | H |      Value Length (7+)      |
# +---+-----+
# | Value String (Length octets) |
# +-----+
```

```
defp encode_string(string) do
  huffman = Huffman.encode(string)
  len = byte_size(huffman)
  << 1::1, encode_int7(len)::bitstring, huffman::binary >>
end
```

INTEGERS

```
defp encode_int6(i) when i < 0b111111,  
  do: << i::6 >>  
defp encode_int6(i),  
  do: << 0b111111::6, encode_big_int(i-0b111111)::bits >>  
  
defp encode_big_int(i) when i < 0b10000000,  
  do: << 0::1, i::7 >>  
defp encode_big_int(i),  
  do: << 1::1, i::7, encode_big_int(i >>> 7)::binary >>
```



Bonus:

RFC-based tests

```
defmodule HPack.RFCSpec.DecodeTest do
  use ExUnit.Case

  setup do
    {:ok, tbl} = HPack.Table.start_link 10000
    {:ok, table: tbl}
  end

  ...

end
```



The value 42 is to be encoded starting at an octet boundary. This implies that a 8-bit prefix is used.

- 42 is less than 255 ($2^8 - 1$) and is represented using the 8-bit prefix.

0	1	2	3	4	5	6	7
0	0	1	0	1	0	1	0

42 stored on 8 bits

C.2 Header Field Representation Examples

This section shows several independent representation examples.

C.2.1 Literal Header Field with Indexing

The header field representation uses a literal name and a literal value. The header field is added to the dynamic table.

Header list to encode:

custom-key: custom-header

Hex dump of encoded data:

400a 6375 7374 6f6d 2d6b 6579 0d63 7573	@.custom-key.cus
746f 6d2d 6865 6164 6572	tom-header

Decoding process:

40	== Literal indexed ==
0a	Literal name (len = 10)
6375 7374 6f6d 2d6b 6579	custom-key
0d	Literal value (len = 13)
746f 6d2d 6865 6164 6572	tom-header

C.2.1 Literal Header Field with Indexing

```
@tag :rfc
```

```
test "Lit Hd
```

```
hbf = <<
```

Hex dump of encoded data:

400a 6375 7374 6f6d 2d6b 6579 0d63 7573	@.custom-key.cus
746f 6d2d 6865 6164 6572	tom-header

```
0x40, 0x0a, 0x63, 0x75, 0x73, 0x74, 0x6f, 0x6d,  
0x2d, 0x6b, 0x65, 0x79, 0x0d, 0x63, 0x75, 0x73,  
0x74, 0x6f, 0x6d, 0x2d, 0x68, 0x65, 0x61, 0x64,  
0x65, 0x72
```

```
>> # @.custom-key.custom-header
```

```
[ dec_header | _ ] = HPack.decode(hbf, tbl)
```

```
assert dec_header == { "custom-key", "custom-header" }
```

```
assert HPack.Table.size(tbl) == 55
```

```
end
```

```
# C.2.1 Literal Header Field with Indexing
```

```
@tag :rfc
```

```
test "Lit Hdr Field with Idx-ing", %{table: tbl} do
```

```
  hbf = <<
```

```
    400a 6375 7374 6f6d 2d6b 6579 0d63 7573 | @.custom-key
```

```
    746f 6d2d 6865 6164 6572                | tom-header
```

```
>>
```

```
[ dec_header | _ ] = HPack.decode(hbf, tbl)
```

```
assert dec_header == { "custom-key", "custom-header" }
```

```
assert HPack.Table.size(tbl) == 55
```

```
end
```

```
# C.2.1 Literal Header Field with Indexing
```

```
@tag :rfc
```

```
test "Lit Hdr Field with Idx-ing", %{table: tbl} do
```

```
  hbf = ~b(
```

```
    400a 6375 7374 6f6d 2d6b 6579 0d63 7573 | @.custom-key
```

```
    746f 6d2d 6865 6164 6572                | tom-header
```

```
)
```

```
[ dec_header | _ ] = HPack.decode(hbf, tbl)
```

```
assert dec_header == { "custom-key", "custom-header" }
```

```
assert HPack.Table.size(tbl) == 55
```

```
end
```

```
defmodule RFCBinaries do
  def sigil_b(string, []) do
    String.split(string, "\n")
    |> Enum.map(fn(part) -> String.split(part, "|") |> List.first end)
    |> Enum.map(fn(part) -> String.split(part, " ") end)
    |> List.flatten |> Enum.filter(&(byte_size(&1) > 0))
    |> Enum.map(fn(part) -> String.to_integer(part, 16) end)
    |> Enum.map(fn(i) ->
      case i do
        x when x in 0..255 -> << x::size(8) >>
        x when x in 256..65535 -> << x::size(16) >>
      end
    end)
    |> Enum.reduce(fn(b, acc) -> acc <> b end)
  end
end
```

👌 `hpack git:(master) mix test --only rfc`

Including tags: `[::rfc]`

Excluding tags: `[::test]`

.....

Finished in 0.1 seconds

33 tests, 0 failures, 22 skipped

Randomized with seed 121899

👌 `hpack git:(master)`



github.com/nestQuick/elixir-hpack

nesQuick / elixir-hpack

Unwatch 2

Unstar 14

Fork 2

Code

Issues 1

Pull requests 1

Wiki

Pulse

Graphs

Settings

A Elixir implementation of the HPack protocol for HTTP/2 header compression — Edit

24 commits

2 branches

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

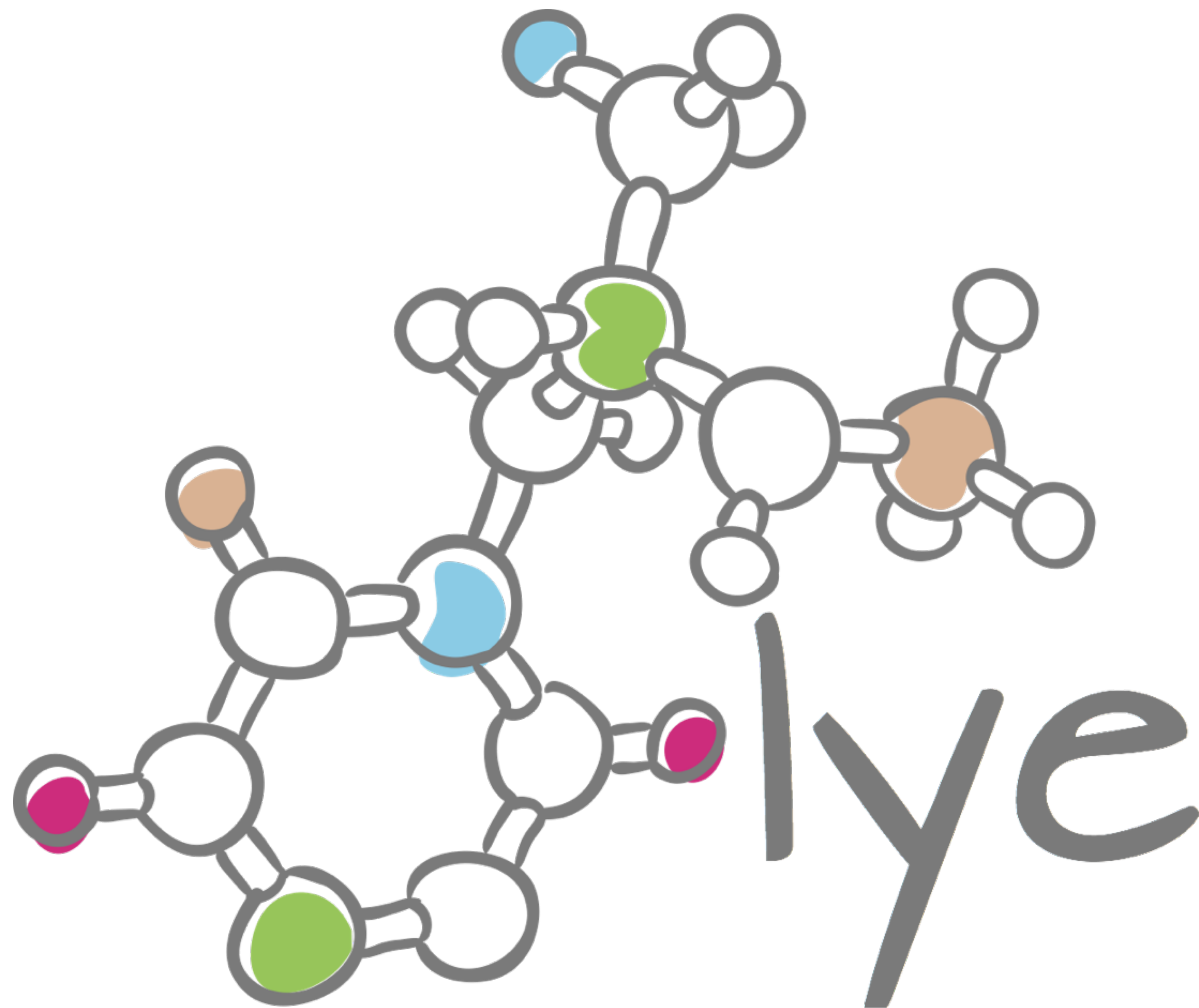
Clone or download



nesQuick doc deps

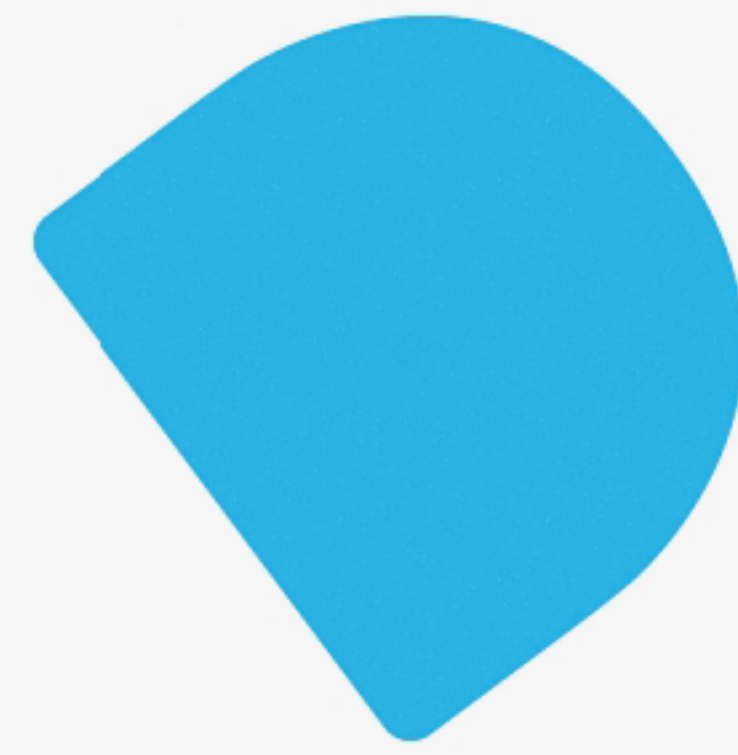
Latest commit f8006c0 on Mar 10

config	first draft including dynamic table	6 months ago
lib	encode big integers correctly	5 months ago
test	consistant writing of HPack	5 months ago
.gitignore	doc deps	5 months ago
.travis.yml	add travis ci	5 months ago
README.md	prepare version 1.0.0 release	5 months ago
mix.exs	doc deps	5 months ago
mix.lock	doc deps	5 months ago





Final Thoughts



OLE MICHAELIS

 CodeStars

 nesQuick

 codestars.eu





THANKYOU

 CodeStars

RESSOURCES

🐦 @lhoguin @bagder @igrigorik @tatsuhiro_t @mnot

<https://github.com/nesQuick/elixir-hpack>

<https://http2.github.io/http2-spec/compression.html>

<https://http2.github.io/http2-spec/>

<http://daniel.haxx.se/http2/>

https://github.com/ninenines/cowlib/blob/master/src/cow_hpack.erl

<http://elixir-lang.org/docs/stable/elixir/Bitwise.html#%3C%3C%3C/2>

<http://www.slideshare.net/peychevi/http2-and-quick-protocols-optimizing-the-web-stack>

<https://ma.ttias.be/architecting-websites-http2-era/>

<https://benramsey.com/talks/2015/05/phptek-http2/>

<https://speakerdeck.com/summerwind/2-prioritization>

http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#HTTP2_HEADER_COMPRESSION

<http://tools.ietf.org/html/rfc6585>

<https://http2.golang.org/>

<https://aprescott.com/posts/spdy-colon-headers>

<https://tools.ietf.org/html/draft-ietf-httpbis-header-compression-12>

<http://news.netcraft.com/archives/2015/06/25/june-2015-web-server-survey.html>

Thanks to all the
people who helped
me with this deck!