インターネット概論 &サーバー入門

Webエンジニアになろう講習会 第5回

自己紹介

yasako

情報通信系に所属しているB2です．いつもフロントエンドを書いています。

前回のおさらい

きれいなコードを書くために

- インデントを適切に使おう
- 命名規則を守ろう
- コードを適切に分割しよう

きれいなコードを保つために

- Linter/Formatterを使おう
- コードレビューをしよう
- ペアプロ・モブプロをしよう

> 自分一人で維持するのは難しい.
> どんどん他人や道具の力を使っていこう！

JavaScriptの歴史

- ブラウザ黎明期から現在にかけて、沢山の人が思い思いに、より良いWeb体験を実現しようと努力していた
- その中でwebページに動きを持たせる技術がいくつか開発されてきた
  - Java Applet
  - Flash
  - JavaScript
  - Silverlight

現在のwebフロントエンド

- 皆が思い思いにより良いWeb体験・開発体験を探求している
- 流行の流れが早い
- **フレームワーク** - コンポーネントや状態管理などの概念の導入や，描画の最適化などを通じて開発者を助けるツール
- **モジュールバンドラー** - バラバラのコードを一つにまとめたり，サイズを小さくしたりするツール

Alt ○○

- 素のHTML, CSS, JavaScriptを書くのは嫌！
- もっと良い言語を作りたい
  - 静的型付け
  - 関数型
  - etc...

👉ならば改善しよう

目次 - 座学

- HTTP
  - HTTPとは
  - リクエストの構造
  - レスポンスの構造
- データ伝送の仕組み
  - TCP
  - IP

目次 - 実習

- Golang/Echoでサーバーを立てよう！
- エンドポイントを生やそう！

HTTPとは

HTTPって何だったっけ

（Web基礎講習会の復習）

- 「お願い（リクエスト）」と「お返事（レスポンス）」でやりとりをする仕組み
- リソースを取得・作成・編集・削除するやりとりができる
  - 「この動画投稿したいです」「201（投稿しました）」
  - 「このページのデータをください」「404（そんなページはありません）」

早速やってみよう！！

Webブラウザを使わずに http://example.com にリクエストを投げてみよう

早速やってみよう！！

- `nc example.com 80` をシェルで実行し、次のように入力してみよう

```
GET / HTTP/1.1
Host: example.com


```

- HTTPのリクエストは改行2つで終わります
- めっちゃ長い文字列がずらずらと出てきたら成功
- 成功したらCtrl+Cを押してコマンドを中断しよう

やってみた

サーバーからレスポンス(返事)が帰ってくる

```
leon@LeonnoMacBook-Pro ~ % nc example.com 80
GET / HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Content-Type: text/html
ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT
Cache-Control: max-age=1606
Date: Mon, 09 Jun 2025 15:22:02 GMT
Content-Length: 1256
Connection: keep-alive

<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
```

HTTPによる通信

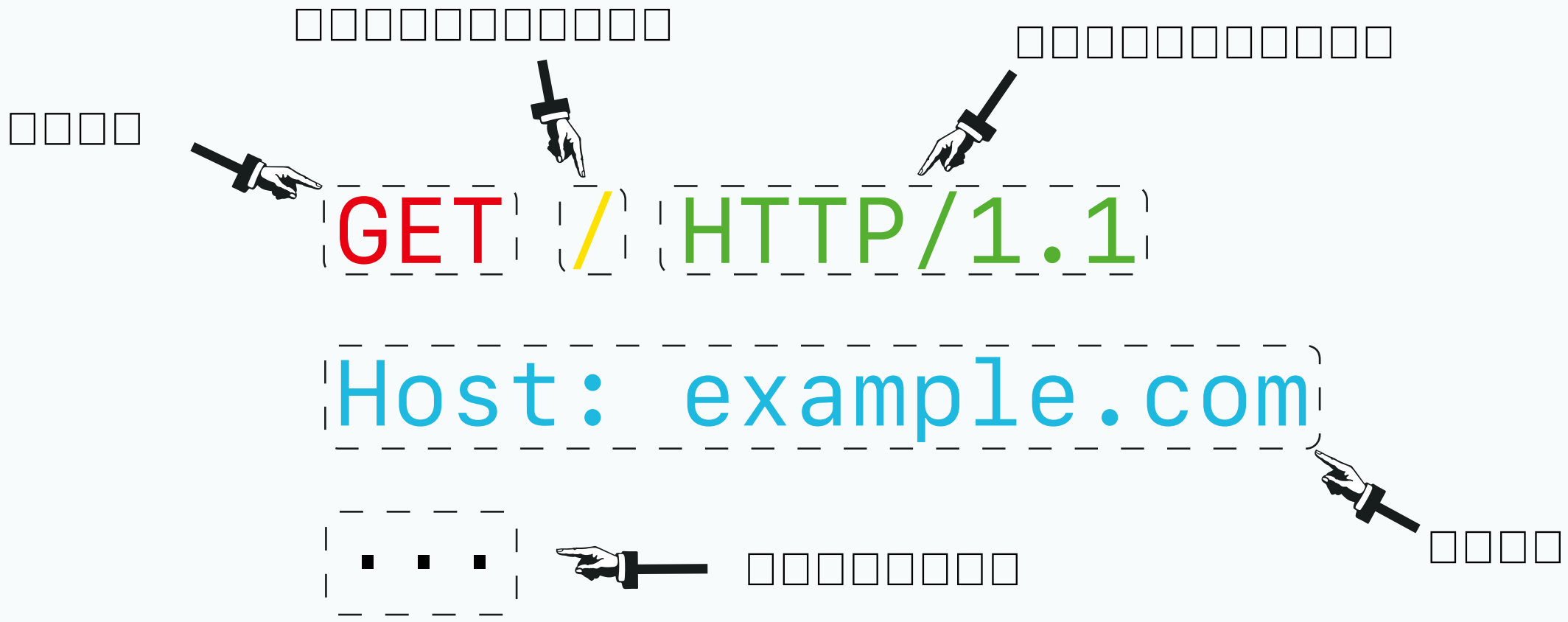
- 今日のブラウザによる通信も、基本的には全部さっきのような「これをください」「あげます」のやりとり
  - ページ本体以外にも画像とかを取得するので、何往復も通信するかもしれない
  - データが圧縮されていたり、暗号化されていたりするかもしれない
  - でも基本的には全部一緒

今日の目標: さっきの一連のやりとりの中身を理解すること

リクエスト

これください！

リクエストの構造：メソッド



メソッド

「どういう種類のリクエストか？」を明示する

- `GET` - このデータください
  - 例 - `https://google.com` のhtmlをください
  - 例 - オンラインゲームで、自分が持っているアイテムの一覧を取得したい
- `POST` - 新たにデータを作ってください
  - 例 - SNSで新規投稿をする
  - 例 - アンケートの回答内容を送信する

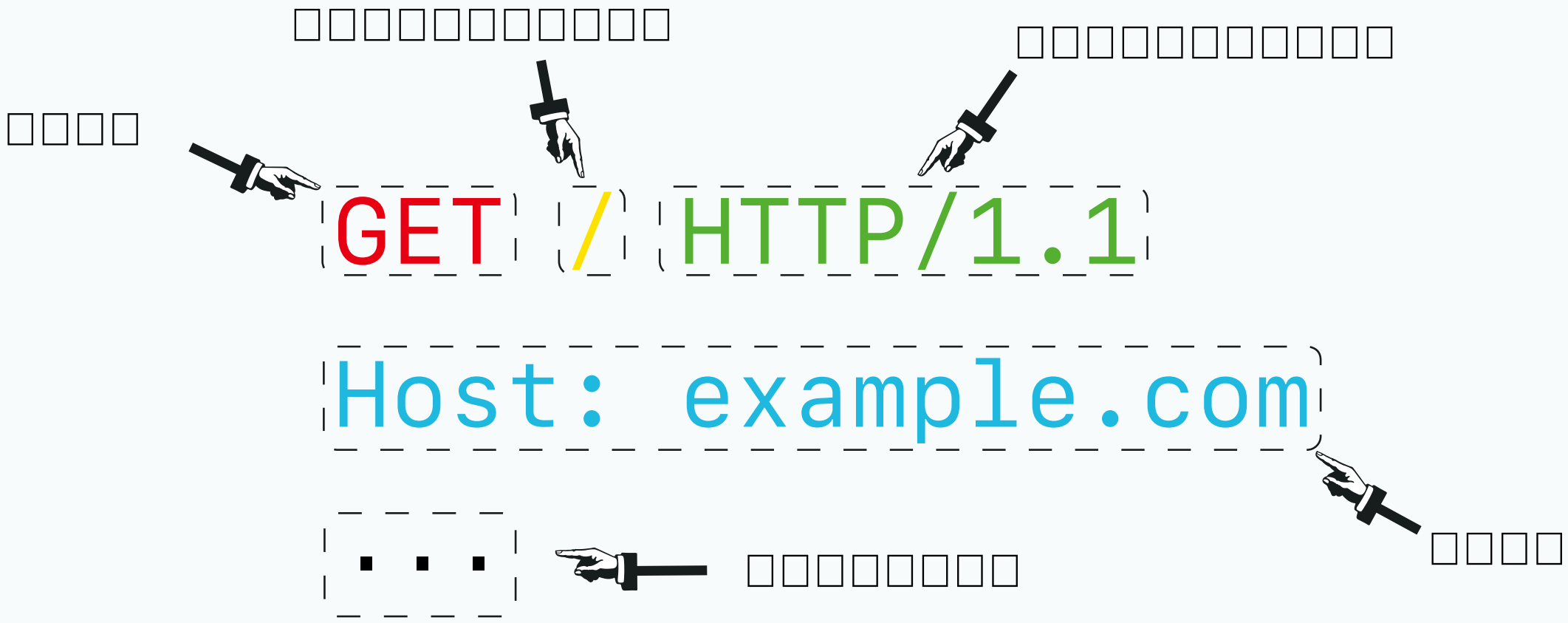
メソッド

- `PUT` - ここにデータを作ってください/ここのデータを上書きしてください
  - 例 - traQで，チャンネルトピックを書き換える
- `PATCH` - このデータのこの場所だけ書き換えてください
  - 例 - traQで、あるチャンネルの特定ユーザーだけの通知状態を書き換える
- `DELETE` - このデータを消してください
  - 例 - 昨日投稿したXのポストを見たら恥ずかしくなったため，削除

メソッド

- 他にも色々あります
- https://developer.mozilla.org/ja/docs/Web/HTTP/Reference/Methods で色々なメソッドの説明が見れます！

リクエストの構造：パス・クエリパラメータ



パスとクエリパラメータ

- 読み書きしたいデータの所在地を表す
- https://google.com/search?q=traP のパスとクエリパラメータはどれか？

URLの構造

https://example.com:8080/path/to/dir?q=somequery#hoge

- `https://` - プロトコル※1
- `example.com` - ドメイン名
  - アクセスするサーバーを表す
- `:8080` - ポート番号
  - あとでやります．

※1 HTTPを暗号化したプロトコルであるHTTPSを利用していることを示します．暗号化せずにサイトにアクセスする場合はURLは`http://`から始まります．

URLの構造

https://example.com:8080/path/to/dir?q=somequery#hoge

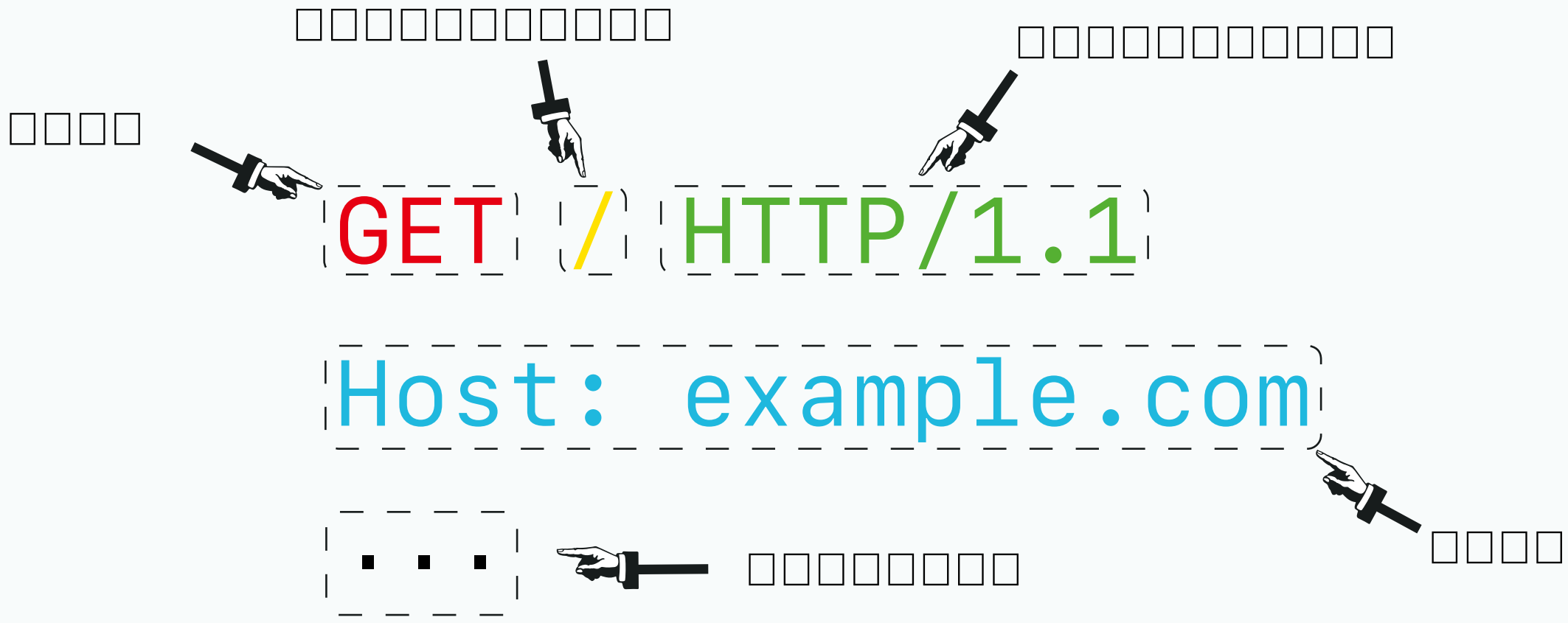
- `/path/to/dir` - パス
  - サーバー内のリソースの位置を指し示します.
- `?q=somequery` - クエリパラメータ
  - `?` で始まる
  - `key=value` の形式でパラメータを指定できる
  - `&` を使い複数のパラメータを指定できる
    - 例 - `?min=0&max=100`

URLの構造

https://example.com:8080/path/to/dir?q=somequery#hoge

- `#hoge` - フラグメント部
  - ブラウザが処理するデータ
  - (主に)文書のどの部分を指しているかを示す
  - リクエストには含まれない

リクエストの構造：プロトコルのバージョン

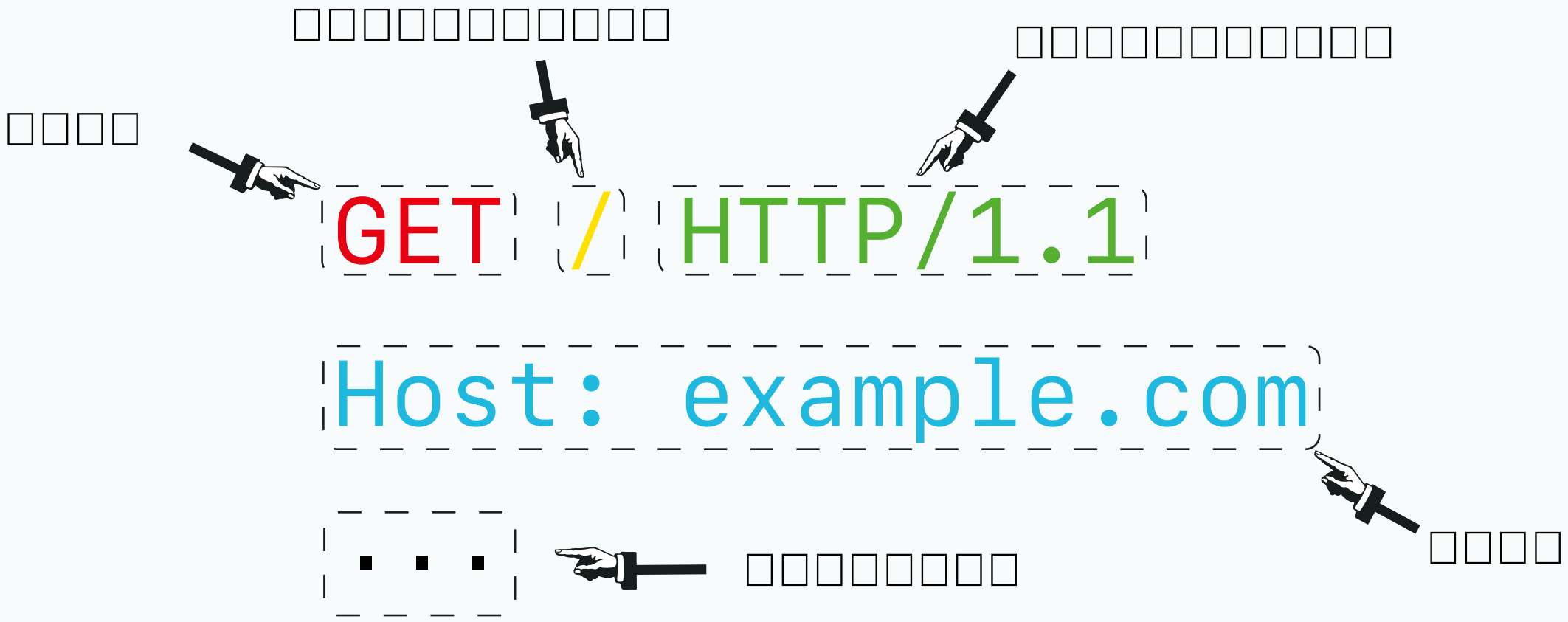


プロトコルのバージョン

- HTTPにはいくつかのバージョンがある
  - 例 - `HTTP/0.9` ※1, `HTTP/1.0`, `HTTP/1.1`, `HTTP/2`, `HTTP/3`
- バージョンが違うとプロトコル(通信形式)が変わるため，バージョンを区別する必要がある

※1 HTTP/0.9の時代はバージョンを区別する必要が無かったので、空欄だった

リクエストの構造：ヘッダー



ヘッダー

通信につけることのできる追加情報

- `ヘッダー名: 値` の形で構成される
- 様々な種類のヘッダーがある※1
  - `Host`
  - `Cookie`
  - `Content-Type`
  - etc...

※1 https://www.iana.org/assignments/http-fields/http-fields.xhtml にヘッダーの種類が収録されています．独自のヘッダーを利用することも可能です．

ヘッダー：Host

クライアントがアクセス先のホスト名を指定する．

- 同じIPアドレスのサーバーで，複数の異なるホスト名のサービスを運用できる．

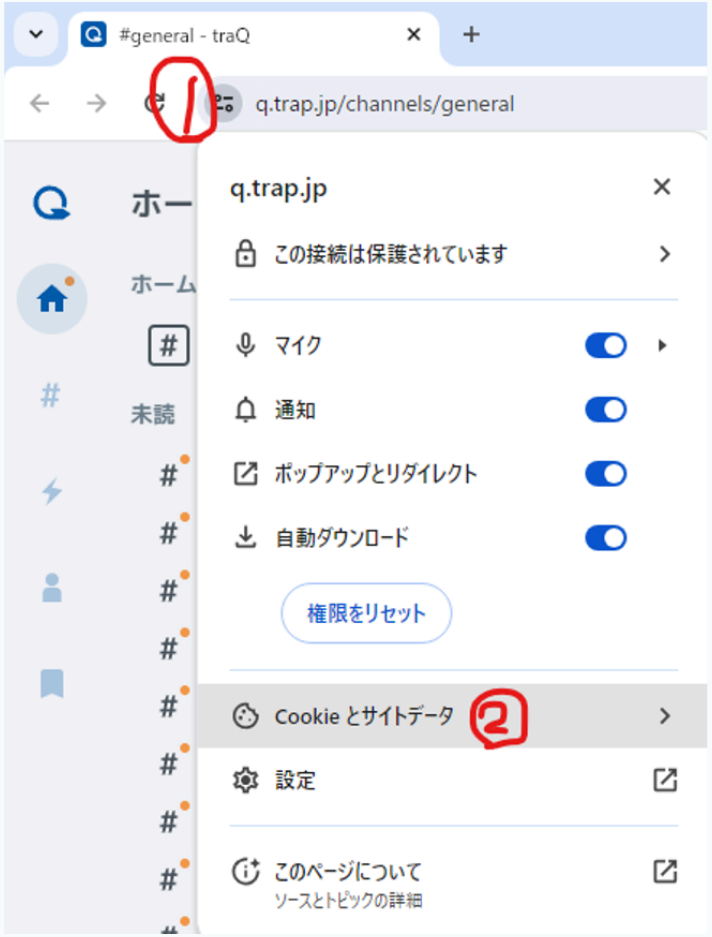
ヘッダー：Cookie

- HTTPはステートレス．※
  - サーバーは前のリクエストの内容を覚えていない
- ブラウザに状態を記録し，リクエスト毎に送ることによって他のリクエストと関連付けることができる
  i. 「あなたのユーザーIDは◯◯なので、これを覚えておいてくださいね」と伝えておく
  ii. 次回以降のリクエストで、ブラウザに毎回ユーザーIDを名乗ってもらう

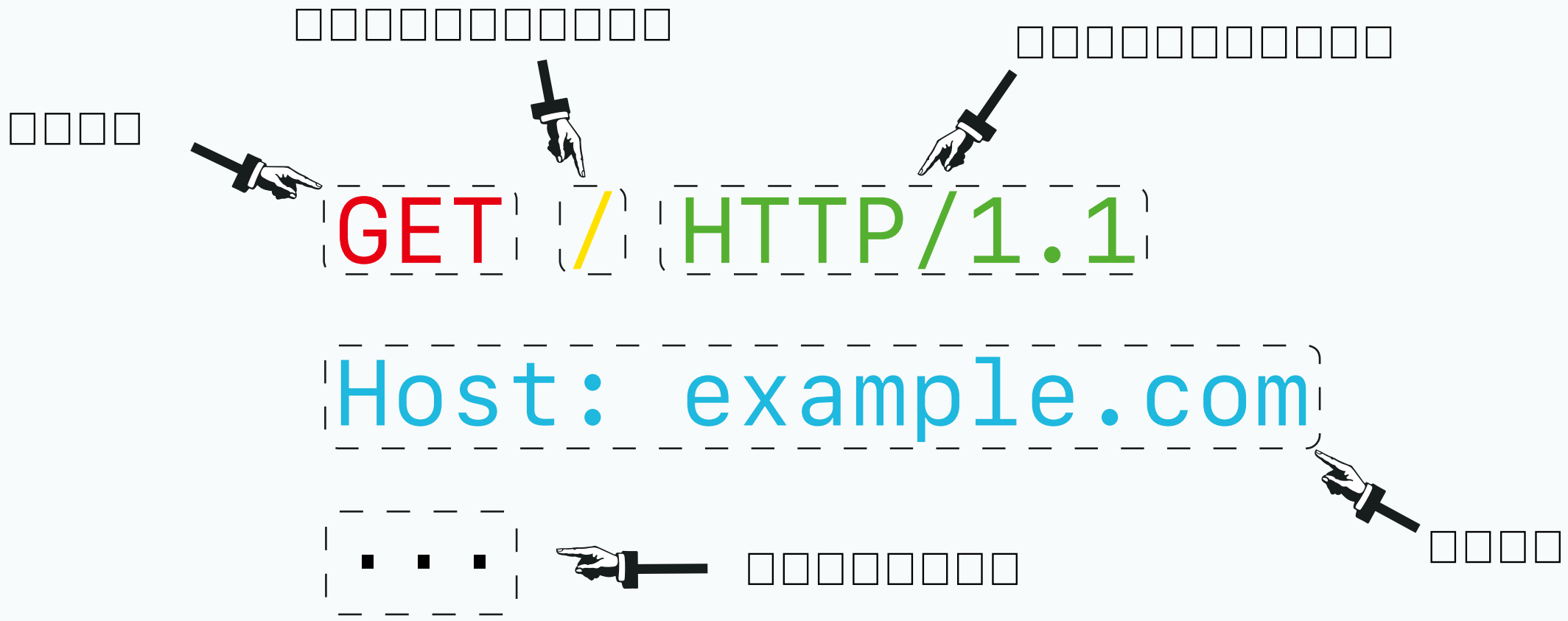
※ 例：麺屋こころにポイントカード忘れていったら特典受けられないよね

やってみよう：Cookieを削除してみよう

ブラウザの設定でq.trap.jp以下のCookieを全部消してみよう



リクエストの構造：リクエストボディ



リクエストボディ

送信するデータの本体

- `POST`，`PUT`，`PATCH` などのメソッドを使用する場合，リクエストにデータの本体を含めることができる．
- リクエストヘッダーの直後に入れることができる．

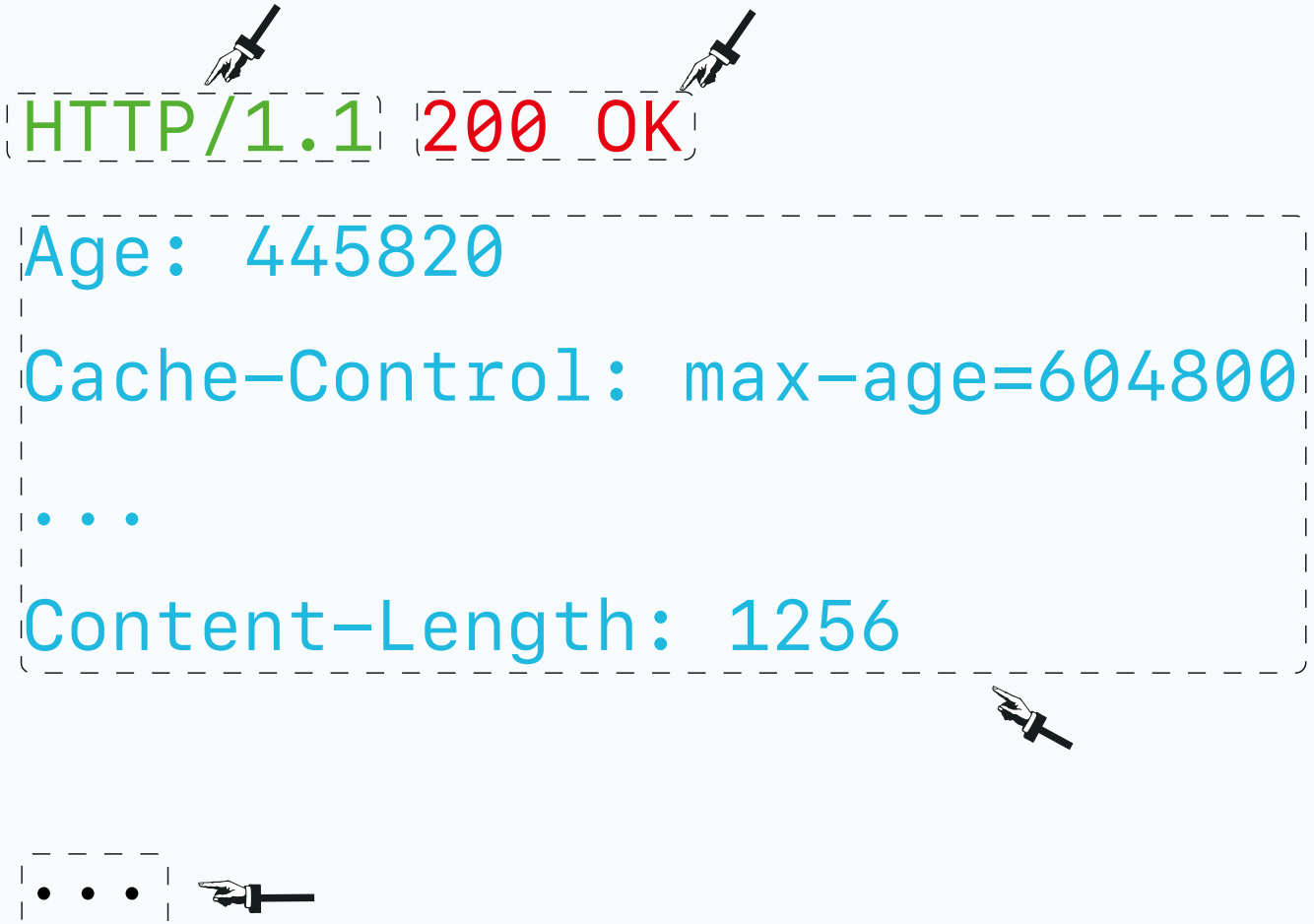
```
POST /api/v3/channels/64b3239d-ac51-4a40-817b-e14db8753128/messages HTTP/2
Host: q.trap.jp
Cookie: r_session=***censored***
Content-Length: 18
Accept: application/json, text/plain, */*
Content-Type: application/json

{"content":"test"}
```

レスポンス

リクエストに対する返事

レスポンスの構造


HTTP/1.1 200 OK
Age: 445820
Cache-Control: max-age=604800
...
Content-Length: 1256
...

やってみた（再掲）

サーバーからレスポンス(返事)が帰ってくる

```
leon@LeonnoMacBook-Pro ~ % nc example.com 80
GET / HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Content-Type: text/html
ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT
Cache-Control: max-age=1606
Date: Mon, 09 Jun 2025 15:22:02 GMT
Content-Length: 1256
Connection: keep-alive

<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
```

レスポンスステータスコード

うまくいったかどうかなど、リクエストの処理結果をカテゴリ別に返す

- 200番台は成功を表す
  - `200 OK` - リクエストが成功したことを示す.
  - `201 Created` - リクエストが成功し，新たにリソースが作成されたことを示す.
  - `204 No Content` - リクエストが成功したが，返すべきものがないことを示す.
  - etc...

レスポンスステータスコード

- 300番台はリダイレクトを表す
  - `302 Found` - 一時的な移動
  - `308 Permanent Redirect` - 永久的な移動

レスポンスステータスコード

- 400番台はクライアントが送ったリクエストに不備があることを表す
  - `403 Forbidden` - 権限がない
  - `404 Not Found` - リソースを発見できない
  - `429 Too Many Requests` - リクエストが多すぎる

レスポンスステータスコード

- 500番台はサーバー側で不具合が生じたことを表す
  - `500 Internal Server Error` - サーバー側でエラーが生じた
  - `503 Service Unavailable` - サーバーがリクエストを処理する準備ができていない

データの伝送

HTTPが依存するプロトコル

TCP

HTTPが依存する通信プロトコル.

- 通信が確実に/正しい順序で届くことを保証する.
- あらゆるデジタルなデータの伝送をすることができる.
  - HTTPのリクエストやメールの送信，コンピュータの遠隔操作(ssh)まで
- 通信をパケット(小包)に分けて送信する

👉TCPの重要な機能について紹介します

TCP: パケットの構造

- パケットは，ヘッダとデータ部分から構成される

- TCPヘッダの構造

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	送信元ポート																送信先ポート															
32	シーケンス番号																															
64	確認応答番号																															
96	ヘッダ長				予約済み				フラグ								ウィンドウサイズ															
128	チェックサム																緊急ポインタ															
160	オプション																															

TCP: アプリケーション間通信

アプリケーションごとにポート番号を振ることでアプリケーション間の通信を実現する．IPアドレスがコンピューターの住所なら，ポート番号は部屋番号．

- `0` - `1023` はWell Known Portsと呼ばれ，基本的なシステムサービスなどに利用される
- ポート番号の例
  - `53` - DNS
  - `80` - HTTP
  - `443` - HTTPS

TCP: 順序制御

- 大きなデータを複数のパケットに分割して送信できる．
- 「シーケンス番号」を振ることによってデータの順番が混ざらないようにしている．

TCP: 到達保証

TCPには、パケットが確実に届くことを保証する仕組みがある

- 受信側がパケットを受け取ったら「受け取りましたよ」という返事(ACKパケット)を返す
- 返事が届かなかったらパケットを送り返す

TCPが向かない用途

- データが届いた場合にいちいち返事を送る必要がある．
  - 常に必要なわけではない
    - 例 - 音声通話で，音声が届かない場合送り直してもしょうがない
- そのため、TCPはリアルタイム性が求められる用途には向かない
- そのような用途ではUDPなどの別のプロトコルを使う
  - UDPはTCPのような到達保証や順序制御がない
  - その代わり、パケットの送信が速い

IPv4

インターネットの基礎となるプロトコル(Internet Protocol).
改良版のIPv6がある.

- 様々な機械を経由して別のコンピューターにデータを届ける仕組み
- ネットワークに接続されているコンピューターは、IPアドレスと呼ばれる番号を持つ(インターネット上の住所のようなもの)
  - 例 - `203.0.113.1`, `198.51.100.1`

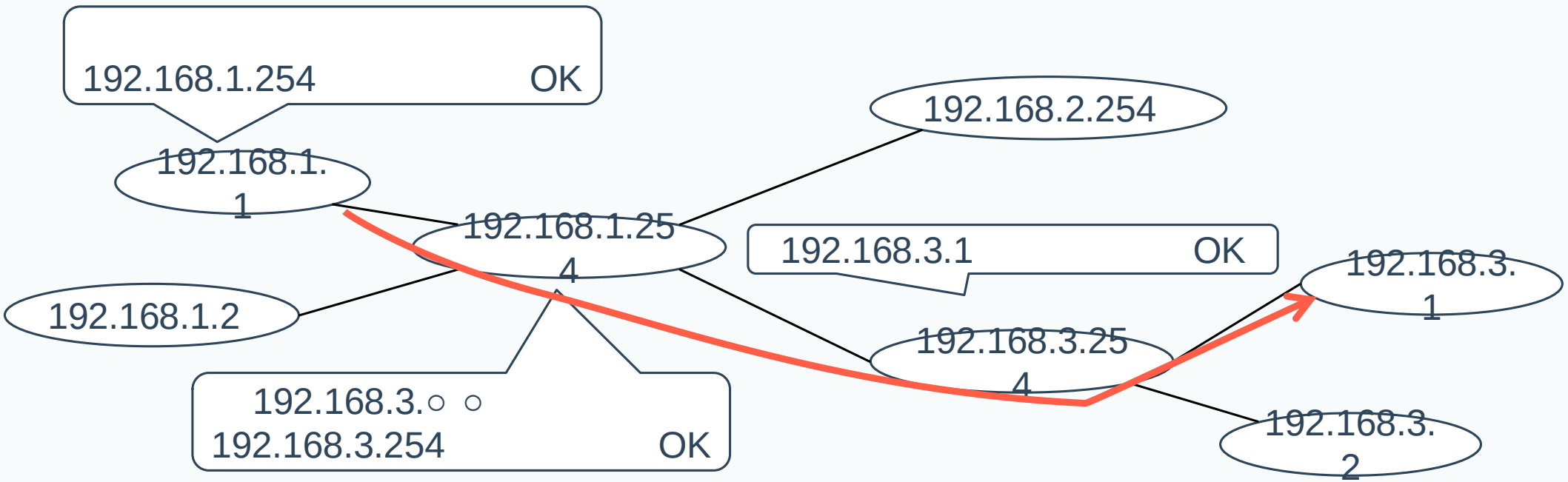
IPv4のヘッダー

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td colspan="4">バージョン</td><td colspan="4">ヘッダ長</td><td colspan="8">サービス種別</td><td colspan="16">全長</td></tr>
<tr><td>32</td><td colspan="16">識別子</td><td colspan="3">フラグ</td><td colspan="13">断片位置</td></tr>
<tr><td>64</td><td colspan="8">生存時間</td><td colspan="8">プロトコル</td><td colspan="16">チェックサム</td></tr>
<tr><td>96</td><td colspan="32">送信元アドレス</td></tr>
<tr><td>128</td><td colspan="32">宛先アドレス</td></tr>
</table>

IPの概要

- 世界中に何億台も機械があるが、そのうち届いてほしい機械にデータを届けたい
- 機械に番号を振る
- ものすごく良い感じに機械に番号を振ることで、機械同士が直接つながっていなくてもデータが届くようになっている

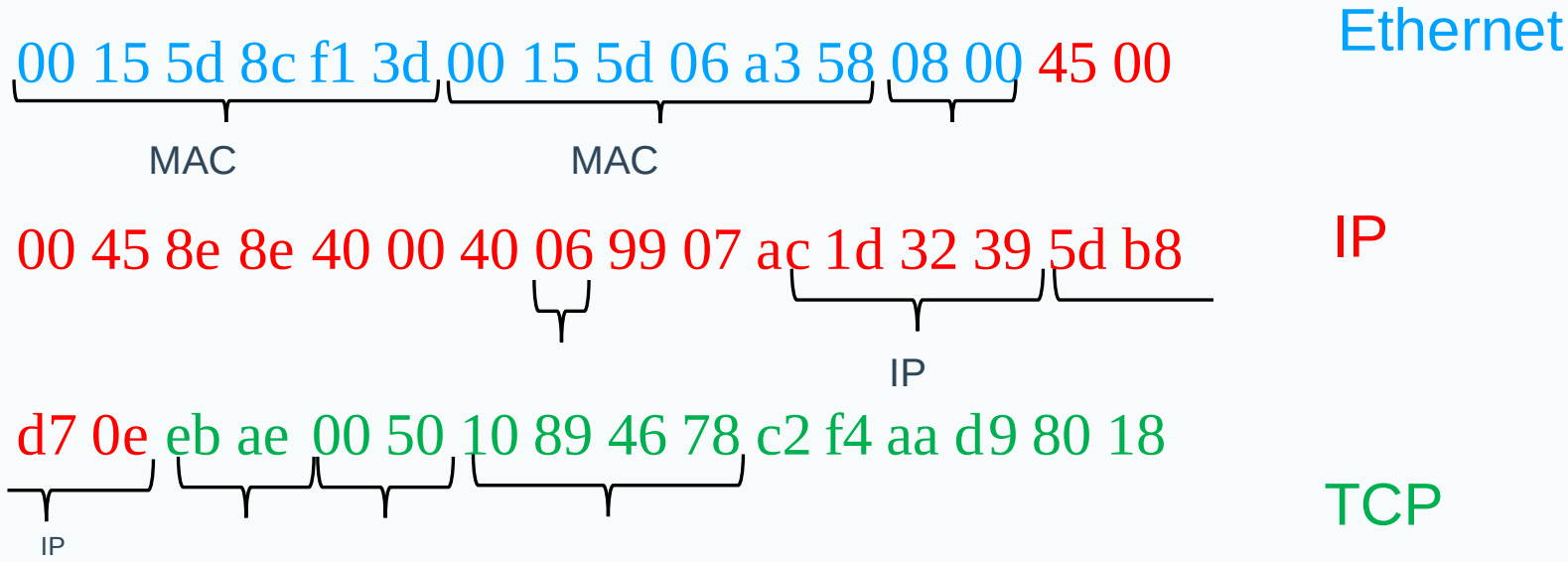
IP


192.168.1.254 OK
192.168.1.1
192.168.1.254
192.168.2.254
192.168.3.1 OK
192.168.3.1
192.168.1.2
192.168.3.254
192.168.3.254
192.168.3.2
192.168.3.○ ○
192.168.3.254 OK

IP

- IPは、物理的に隣接している相手と通信できることを前提にしている
- 通信手段は何でも良い
  - PPPoEかもしれないし、Wi-Fiかもしれない
  - その下のプロトコルがなんでも使える

パケット構造


00 15 5d 8c f1 3d 00 15 5d 06 a3 58 08 00 45 00
MAC
MAC
00 45 8e 8e 40 00 40 06 99 07 ac 1d 32 39 5d b8
IP
d7 0e eb ae 00 50 10 89 46 78 c2 f4 aa d9 80 18
IP
01 f6 9b d0 00 00 01 01 08 0a 09 2d ec 32 82 39
ad 07 48 6f 73 74 3a 65 78 61 6d 70 6c 65 2e 63
H o s t : e x a mp l e . c
6f 6d 0a
o m
Ethernet
IP
TCP
HTTP

まとめ

- HTTPリクエストは、メソッド、パス、クエリパラメータ、ヘッダー、ボディから構成される
- HTTPレスポンスは、ステータスコード、ヘッダー、ボディから構成される
- HTTPはTCP/IPの上で動作する※
- TCPはデータの順序制御と到達保証を提供するプロトコル
- IPアドレスはインターネット上の機械の住所のようなもの

※ HTTP/3はUDP/IP上のプロトコルであるQUICの上で動作します.

参考文献

オーム社

マスタリングTCP/IP 入門編

TCP/IPの基礎を学ぶための入門書です.

演習タイム