

Backend Fundamental

Thinc - First Act

2023-10-10

Agenda

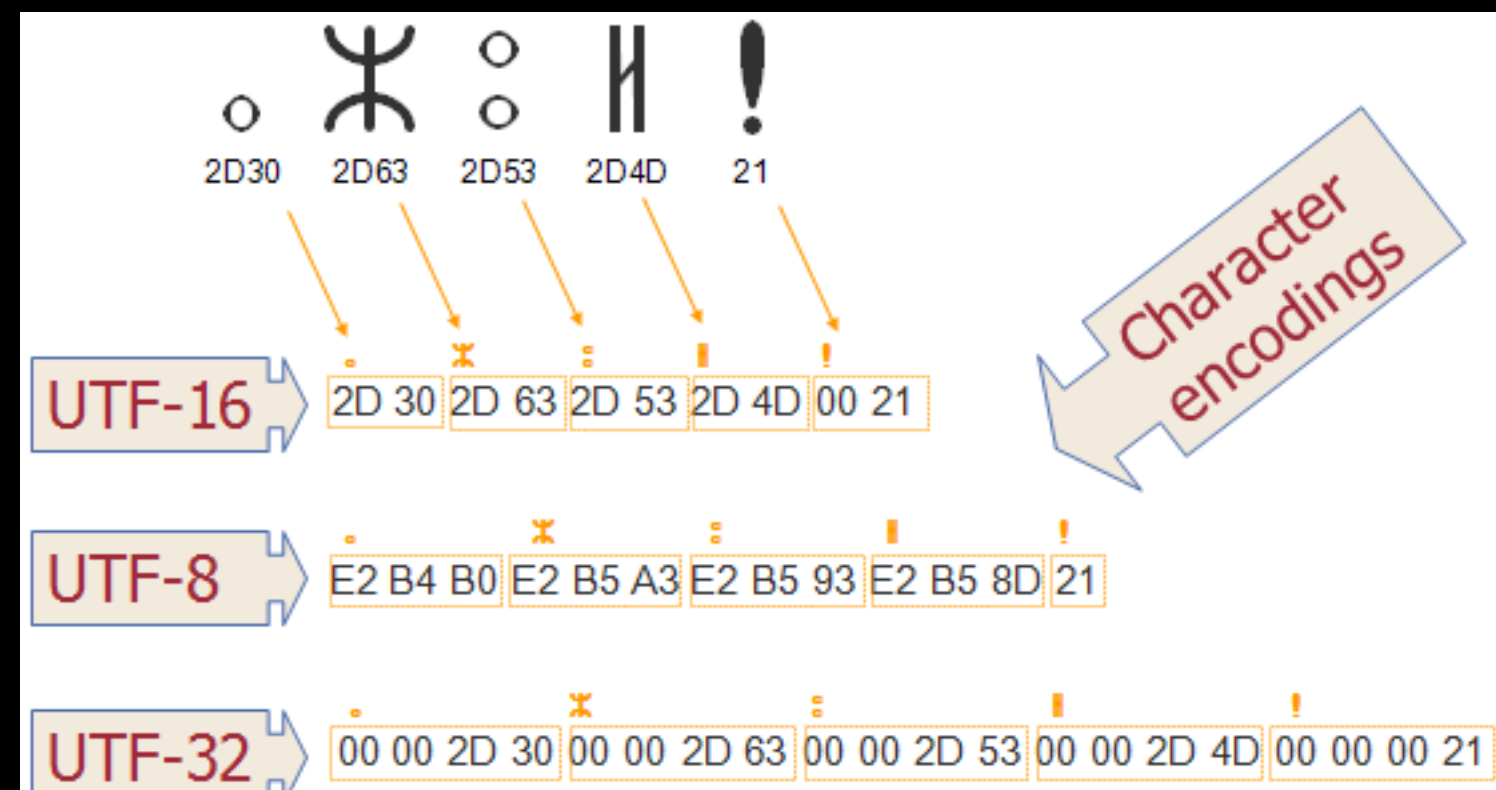
- Learn what is “HTTP Request”, and how does it really work.
- Implement basic backend server
 - Authorization, Content-Type header
 - Middleware

Let's start with these facts

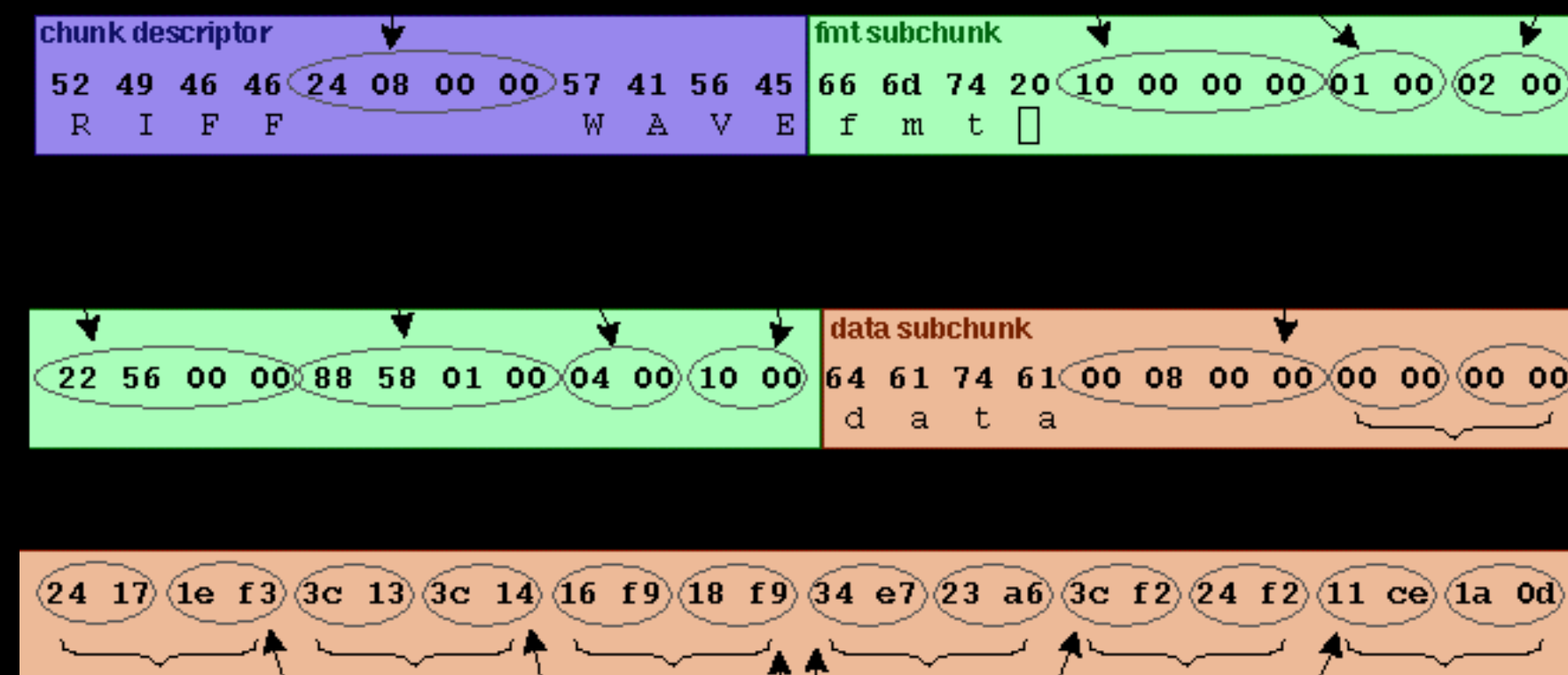
Once the connection between computers is established, it can *communicate* to the other computer by sending sequence of 0 and 1 (data).

What can sequence of 0 and 1 do?

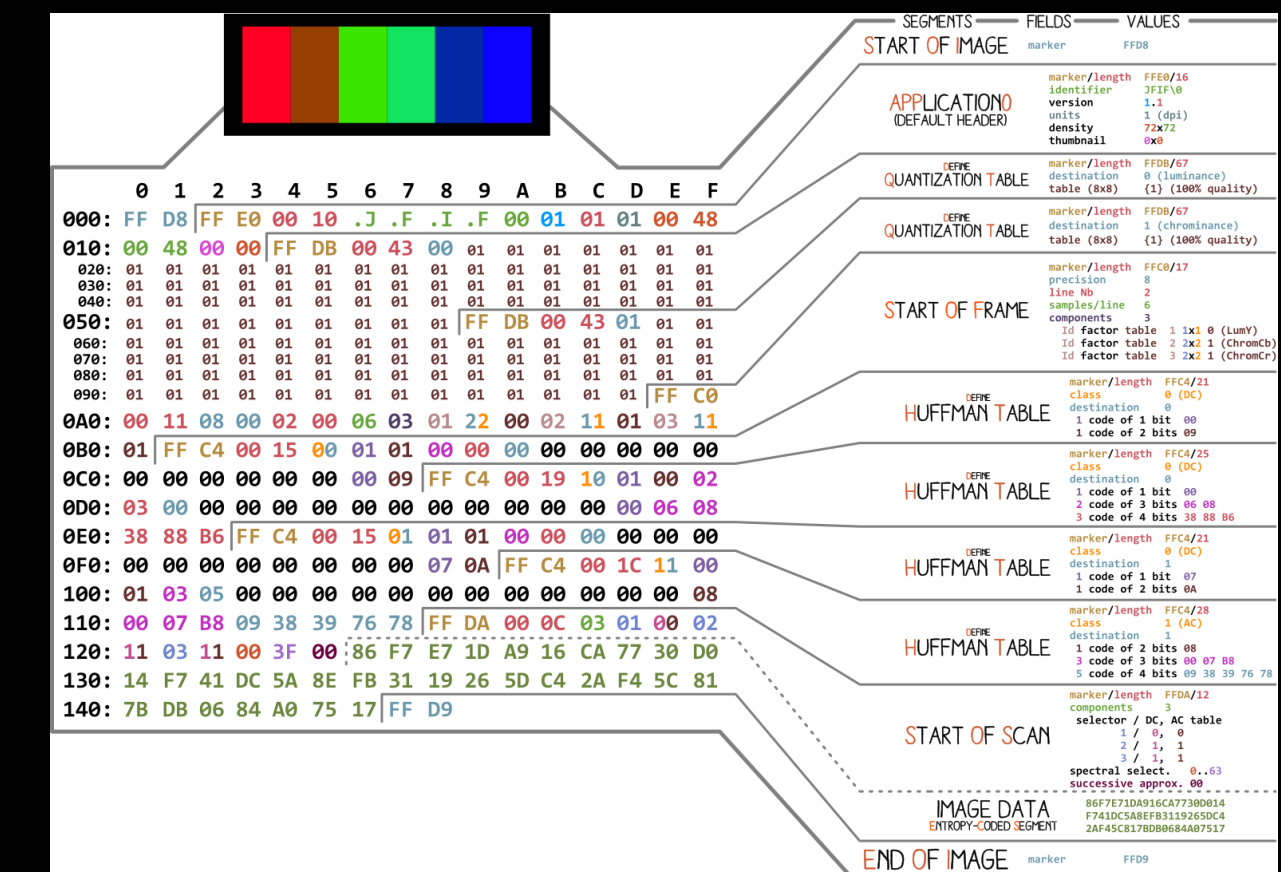
- These bits can mean anything, it is just a representation.
- There're endless possibility of how people defining what does each sequence of bit mean and there's no *correct* way, so there're methods of interpreting these data.



Character encoding



.wav file format



.jpg file format

In fact, this is really similar to what happen in Linguists.

Computer	Linguists	Description
Data	Phoneme (How to pronounce words)	Same (data/phoneme) has different meaning depends on (encoding/language).
Encoding	Language	People use (encoding/language) that previous generation have created.
Software	Book	If (encoding/language) changes, the (software/book) can be misinterpreted.

[1]: One different is that encoding/protocol is clearly defined in their specification, so they do not change over time unlike language. (They evolve using versioning instead)

Where are we going?

- There's nothing special about utf-8, every people seem to agree that this encoding makes sense and build their stuff upon it.
- It does not have to be “the best” or “optimal”.

HTTP (HyperText Transfer Protocol)

- All of websites use this protocol to send you what they want to send.
- An engineer team (IETF) has developed specification for HTTP.



HTTP (HyperText Transfer Protocol)

What

Client (you) send HTTP Request to server.

When

Every time you want *any* data from the server.

Where

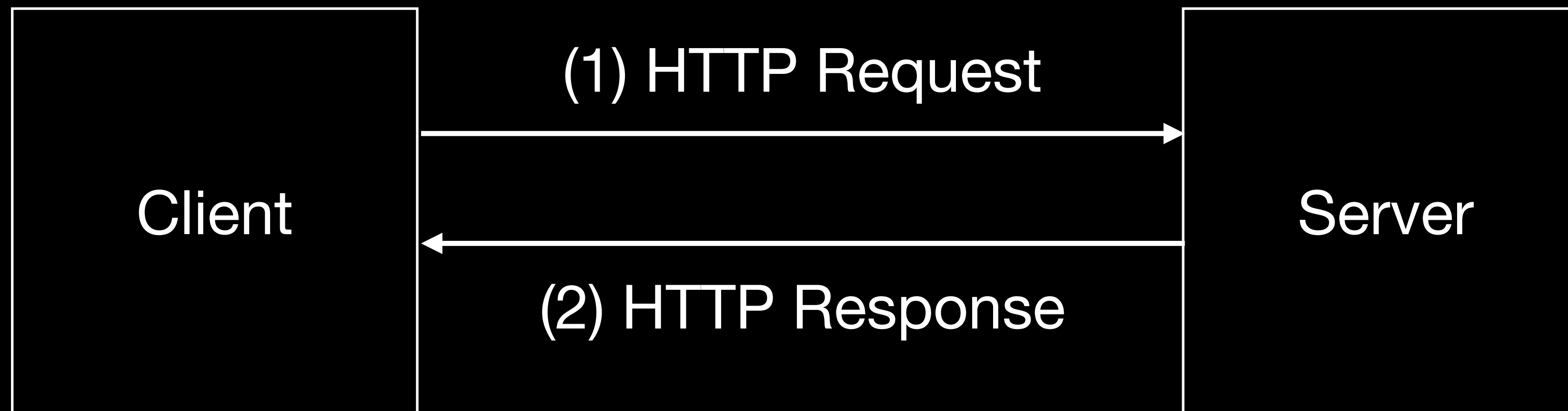
From your browser to their server.

Why

To get data you want.

HTTP (HyperText Transfer Protocol)

Flow



HTTP Request

There're a lot of information in the specification (by lot I mean 175 pages for HTTP/1.1 which is reference for the presentation).

So, I'm going to show only important parts.

- Request-Line
- Header
- Message Body

Ref

HTTP Request

Status Line: Method

- Use to tell action
 - GET → Retrieve
 - POST → Create
 - PUT → Edit, Patch → Partial Edit
 - DELETE → Delete
- POST, PUT usually have message body

HTTP Request

Status Line: URI

- Use to tell where to find resource
- Consist of path + query in format `{path}?{query1}&{query2}&...&{queryN}`
- Each query is pair of key and value
- Client can send zero or more query, each query separated by ‘&’
- This form dictionary

`/watch?v=dQw4w9WgXcQ&t=4`

[1] I am unable to find specification for query as dictionary, this is the best I can find.
“However, as query components are **often** used to carry identifying information in the form
"key=value" pairs and ...”. [\[ref\]](#)
Maybe, this dictionary is non-standard use of query. But people seem to widely accepted it.

[2] I think [1] make senses because I never find a *conventional*
insert array as query.

HTTP Request

Headers

- Tell metadata of the request
- Important header (for today)
 - Authorization
 - To tell which user is performing the action.
 - Example “Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==” [[ref](#)]

HTTP Request Example

```
$ curl -v 'www.youtube.com/watch?v=dQw4w9WgXcQ'
* Trying 2404:6800:4016:803::200e:80 ...
* Connected to www.youtube.com (2404:6800:4016:803::200e) port 80 (#0)
> GET /watch?v=dQw4w9WgXcQ HTTP/1.1
> Host: www.youtube.com
> User-Agent: curl/7.85.0
> Accept: */*
>
```

- My DNS resolver happens to resolve to IPv6 which does not matter.
- Port 80 is default for HTTP, HTTPS is beyond scope of the lecture :o
- Request header section ends with 'CRLF'

HTTP Response

- Status Line
- Header (Serve same purpose for request but for response)
- Message Body (Same format as request)

Ref

HTTP Response

Status Line: Status Code

- Use to tell status of the request with 3 digit
- [\[ref\]](#)

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. There are 5 values for the first digit:

- 1xx: Informational - Request received, continuing process
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

HTTP Response

Status Line: Status Code

- Common status code
 - 200 OK
 - 201 Created
 - 301 Moved Permanently → Redirection
 - 400 Bad Request → The request is malformed
 - 401 Unauthorized → User has not login
 - 403 Forbidden → User login, but does not have enough permission
 - 404 Not Found

```
$ curl -v google.com
-- (HTTP Request) --
< HTTP/1.1 301 Moved Permanently
< Location: http://www.google.com/
-- (Irrelevant) --
```

google.com tell client to send request to www.google.com instead.
301 status code must use along with 'Location' header.

HTTP Response

Headers





























































- Tell metadata of the request
- Important header (for today)
 - Content-Type
 - tell how to body should be interpreted/parsed
 - This are call MIME type
 - Common MIME type
 - text/html, text/css, application/json, application/xml
 - image/{png,jpeg,svg}

HTTP Response Example

```
$ curl -v jsonplaceholder.typicode.com/todos/1
- (HTTP Request) -
< HTTP/1.1 200 OK
< Date: Fri, 06 Oct 2023 20:46:21 GMT
< Content-Type: application/json; charset=utf-8
< Content-Length: 83
- (Irreverent headers) -
< alt-svc: h3=":443"; ma=86400
<
{
  "userId": 1,
  "id": 1,
  "title": "delectus aut autem",
  "completed": false
}
```

- ‘Content-Type’ header match the response body format
- Response header section ends with ‘CRLF CRLF’

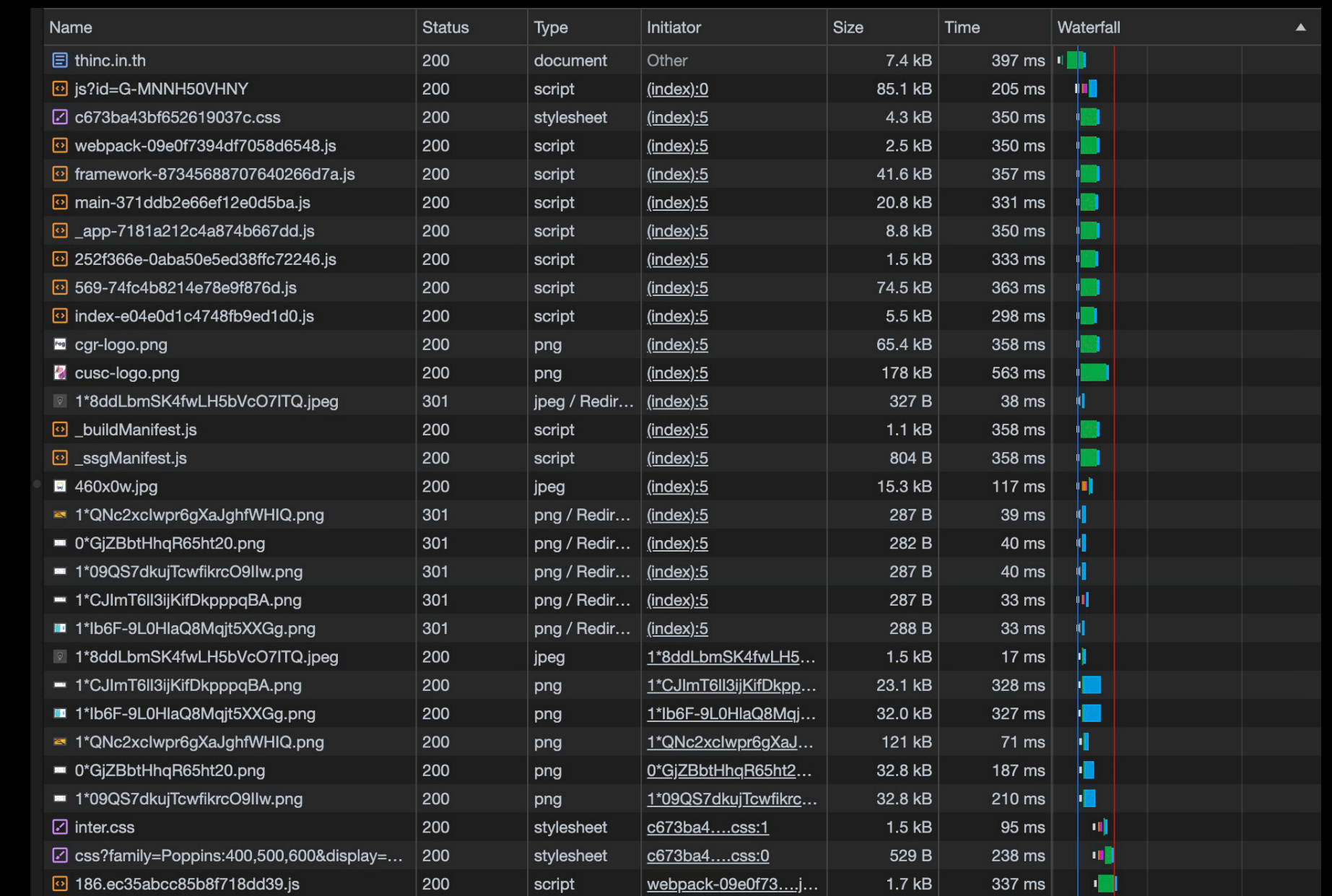
Why are there are a lot of requests?

Name	Status	Type	Initiator	Size	Time	Waterfall	
 thinc.in.th	200	document	Other	7.4 kB	397 ms		
 js?id=G-MNNH50VHNY	200	script	(index):0	85.1 kB	205 ms		
 c673ba43bf652619037c.css	200	stylesheet	(index):5	4.3 kB	350 ms		
 webpack-09e0f7394df7058d6548.js	200	script	(index):5	2.5 kB	350 ms		
 framework-87345688707640266d7a.js	200	script	(index):5	41.6 kB	357 ms		
 main-371ddb2e66ef12e0d5ba.js	200	script	(index):5	20.8 kB	331 ms		
 _app-7181a212c4a874b667dd.js	200	script	(index):5	8.8 kB	350 ms		
 252f366e-0aba50e5ed38ffc72246.js	200	script	(index):5	1.5 kB	333 ms		
 569-74fc4b8214e78e9f876d.js	200	script	(index):5	74.5 kB	363 ms		
 index-e04e0d1c4748fb9ed1d0.js	200	script	(index):5	5.5 kB	298 ms		
 cgr-logo.png	200	png	(index):5	65.4 kB	358 ms		
 cusc-logo.png	200	png	(index):5	178 kB	563 ms		
 1*8ddLbmSK4fwLH5bVcO7ITQ.jpeg	301	jpeg / Redir...	(index):5	327 B	38 ms		
 _buildManifest.js	200	script	(index):5	1.1 kB	358 ms		
 _ssgManifest.js	200	script	(index):5	804 B	358 ms		
 460x0w.jpg	200	jpeg	(index):5	15.3 kB	117 ms		
 1*QNc2xclwpr6gXaJghfWHIQ.png	301	png / Redir...	(index):5	287 B	39 ms		
 0*GjZBbtHhqR65ht20.png	301	png / Redir...	(index):5	282 B	40 ms		
 1*09QS7dkujTcwfikrcO9llw.png	301	png / Redir...	(index):5	287 B	40 ms		
 1*CJlmT6ll3ijKifDkpppqBA.png	301	png / Redir...	(index):5	287 B	33 ms		
 1*lb6F-9L0HlaQ8Mqjt5XXGg.png	301	png / Redir...	(index):5	288 B	33 ms		
 1*8ddLbmSK4fwLH5bVcO7ITQ.jpeg	200	jpeg	1*8ddLbmSK4fwLH5...	1.5 kB	17 ms		
 1*CJlmT6ll3ijKifDkpppqBA.png	200	png	1*CJlmT6ll3ijKifDkpp...	23.1 kB	328 ms		
 1*lb6F-9L0HlaQ8Mqjt5XXGg.png	200	png	1*lb6F-9L0HlaQ8Mqj...	32.0 kB	327 ms		
 1*QNc2xclwpr6gXaJghfWHIQ.png	200	png	1*QNc2xclwpr6gXaJ...	121 kB	71 ms		
 0*GjZBbtHhqR65ht20.png	200	png	0*GjZBbtHhqR65ht2...	32.8 kB	187 ms		
 1*09QS7dkujTcwfikrcO9llw.png	200	png	1*09QS7dkujTcwfikrc...	32.8 kB	210 ms		
 inter.css	200	stylesheet	c673ba4....css:1	1.5 kB	95 ms		
 css?family=Poppins:400,500,600&display=...	200	stylesheet	c673ba4....css:0	529 B	238 ms		
 186.ec35abcc85b8f718dd39.js	200	script	webpack-09e0f73....j...	1.7 kB	337 ms		

Why are there are a lot of requests?

What really happens is

1. When you go to thinc.in.th, your browser send *HTTP Request* to thinc.in.th
2. It detects that the response of the request is HTML (using `Content-Type` Header)
3. It renders the page as HTML
4. The rendered HTML need to make more request to the server. e.g., it need to fetch font, picture, or even more script to run on for the website.



Name	Status	Type	Initiator	Size	Time	Waterfall
thinc.in.th	200	document	Other	7.4 kB	397 ms	
js?id=G-MNNH50VHNY	200	script	(index):0	85.1 kB	205 ms	
c673ba43bf652619037c.css	200	stylesheet	(index):5	4.3 kB	350 ms	
webpack-09e0f7394df7058d6548.js	200	script	(index):5	2.5 kB	350 ms	
framework-87345688707640266d7a.js	200	script	(index):5	41.6 kB	357 ms	
main-371ddb2e66ef12e0d5ba.js	200	script	(index):5	20.8 kB	331 ms	
_app-7181a212c4a874b667dd.js	200	script	(index):5	8.8 kB	350 ms	
252f366e-0aba50e5ed38ffc72246.js	200	script	(index):5	1.5 kB	333 ms	
569-74fc4b8214e78e9f876d.js	200	script	(index):5	74.5 kB	363 ms	
index-e04e0d1c4748fb9ed1d0.js	200	script	(index):5	5.5 kB	298 ms	
cgr-logo.png	200	png	(index):5	65.4 kB	358 ms	
cusc-logo.png	200	png	(index):5	178 kB	563 ms	
1*8ddLbmSK4fwLH5bVcO7ITQ.jpeg	301	jpeg / Redir...	(index):5	327 B	38 ms	
_buildManifest.js	200	script	(index):5	1.1 kB	358 ms	
_sbgManifest.js	200	script	(index):5	804 B	358 ms	
460x0w.jpg	200	jpeg	(index):5	15.3 kB	117 ms	
1*QNc2xclwpr6gXaJghfWHlQ.png	301	png / Redir...	(index):5	287 B	39 ms	
0*GjZBbtHhqR65ht20.png	301	png / Redir...	(index):5	282 B	40 ms	
1*09QS7dkujTcwfikrcO9llw.png	301	png / Redir...	(index):5	287 B	40 ms	
1*CJlmT6lI3ijKifDkpppqBA.png	301	png / Redir...	(index):5	287 B	33 ms	
1*lb6F-9L0HlaQ8Mqjt5XXGg.png	301	png / Redir...	(index):5	288 B	33 ms	
1*8ddLbmSK4fwLH5bVcO7ITQ.jpeg	200	jpeg	1*8ddLbmSK4fwLH5...	1.5 kB	17 ms	
1*CJlmT6lI3ijKifDkpppqBA.png	200	png	1*CJlmT6lI3ijKifDkpp...	23.1 kB	328 ms	
1*lb6F-9L0HlaQ8Mqjt5XXGg.png	200	png	1*lb6F-9L0HlaQ8Mqj...	32.0 kB	327 ms	
1*QNc2xclwpr6gXaJghfWHlQ.png	200	png	1*QNc2xclwpr6gXaJ...	121 kB	71 ms	
0*GjZBbtHhqR65ht20.png	200	png	0*GjZBbtHhqR65ht2...	32.8 kB	187 ms	
1*09QS7dkujTcwfikrcO9llw.png	200	png	1*09QS7dkujTcwfikrc...	32.8 kB	210 ms	
inter.css	200	stylesheet	c673ba4....css:1	1.5 kB	95 ms	
css?family=Poppins:400,500,600&display=...	200	stylesheet	c673ba4....css:0	529 B	238 ms	
186.ec35abcc85b8f718dd39.js	200	script	webpack-09e0f73....j...	1.7 kB	337 ms	

ExpressJS

Now we know that computer send bunch of 0 and 1 to other server in order to render website, retrieve data, mutate data.

ExpressJS will facilitate us by parsing these request into easy-to-use format and providing way to build HTTP response with your javascript data.

It also route this request respective handler.

Getting started

Install Node

Linux / OSX

This is also HTTP Request



```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.4/install.sh | bash
```

Windows

Download from <https://github.com/coreybutler/nvm-windows>

After nvm is installed, run

```
$ nvm install --lts  
$ nvm use --lts
```

Getting started

Init NodeJS package

```
$ mkdir thinc-first-act-backend  
$ cd thinc-first-act-backend  
$ npm init -y  
$ npm i express
```

This is also HTTP Request



Time for implementation!!!

Vote app

Specification

- App should have rate limit of 10 requests per ip per 10 seconds
- Status code and message is not strict but should return as appropriate.
- Persistence is not required.

Time for implementation!!!

Vote app

Specification

- GET /vote
 - Return object of vote score
 - Example: {"A": 5, "B": 6, "C": 7}
- GET /health
 - For health checking (check that app is up and running)

Time for implementation!!!

Vote app

Specification

- POST /vote
 - Use basic authorization with user = student id, no need to check password.
 - Example: "Authorization: MTIzNDU2Nzg5MDo=" # user: '1234567890'
 - Body is specify in this format
 - {"vote": "choiceA"}
 - Don't forget to add 'Content-Type' header
 - User can only vote once, if there's more attempt to vote return 409 Conflict

Next step 1

Configuration

Your app should be configurable using any method that is not hard-coded.

Next step??

- Router
- Serve static file
- Caching (header)
- Authentication

API Design

Yeah yeah, I can create these endpoints (or can you?), But how do I “design” it

Follow these guidelines

- RESTful web API design
- WebAPI Checklist by Mathieu Fenniak

Architecture

- Controller-Service-Repository Pattern
- Example by hagopj13

What next??

Choose 1 topic 🧛

- Deployment + Docker → How to deploy your app
- Database + ORM → How to persist your data
- TLS + DNS (https) → How to make your website https://____.com
- Git + CI/CD → How to work with other people