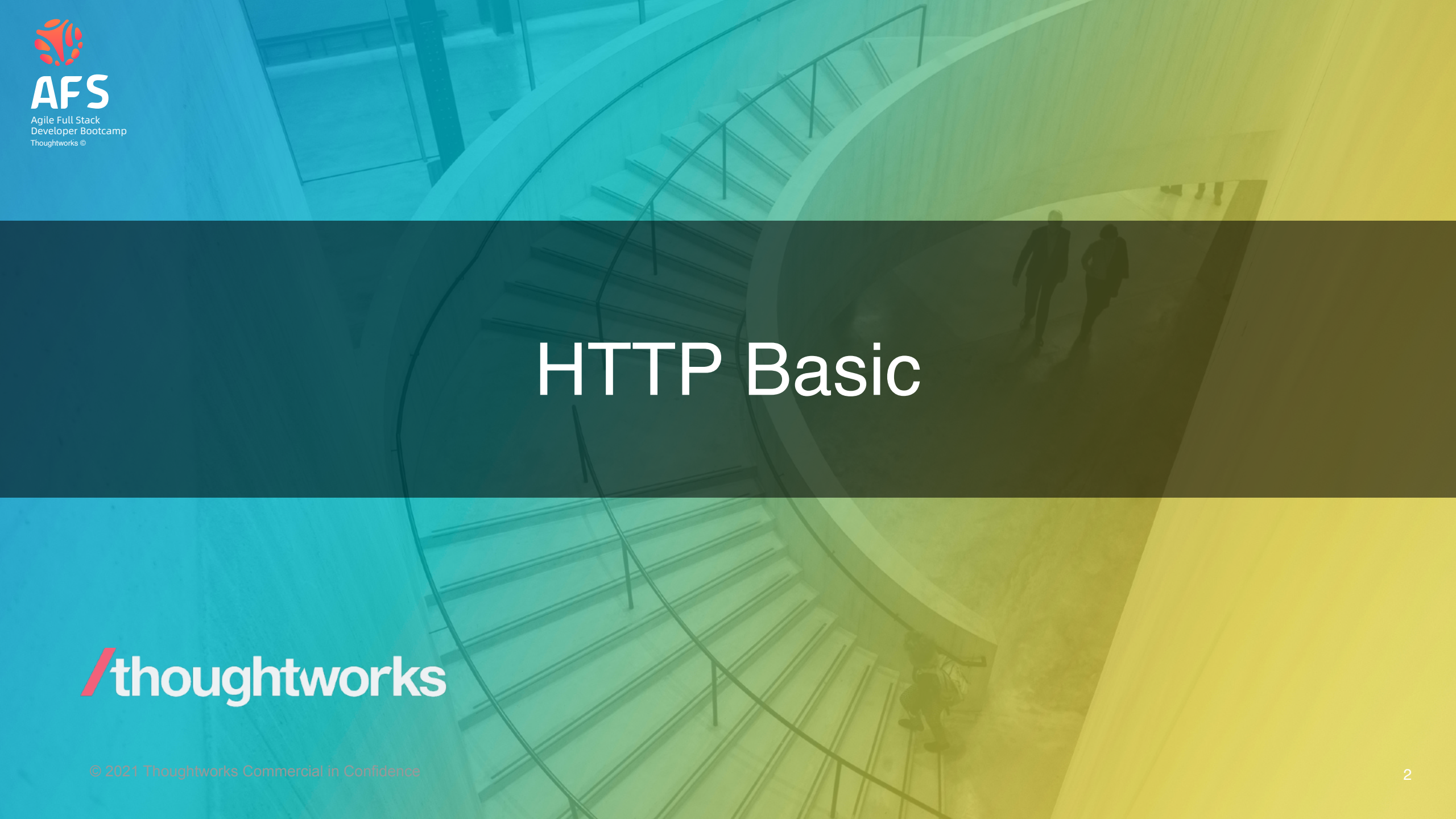
AFS
Agile Full Stack
Developer Bootcamp
Thoughtworks ©

HTTP Basic & RESTful

/thoughtworks

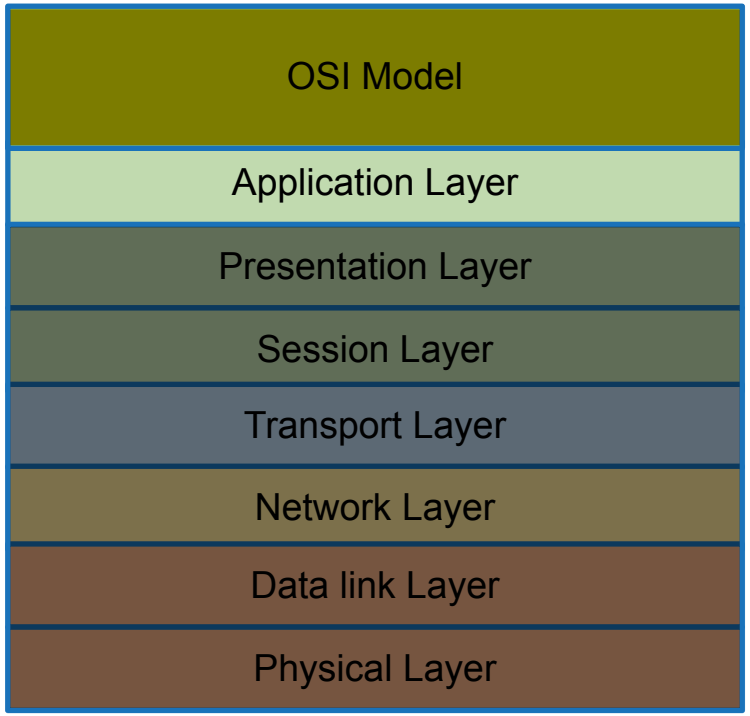
HTTP Basic



thoughtworks

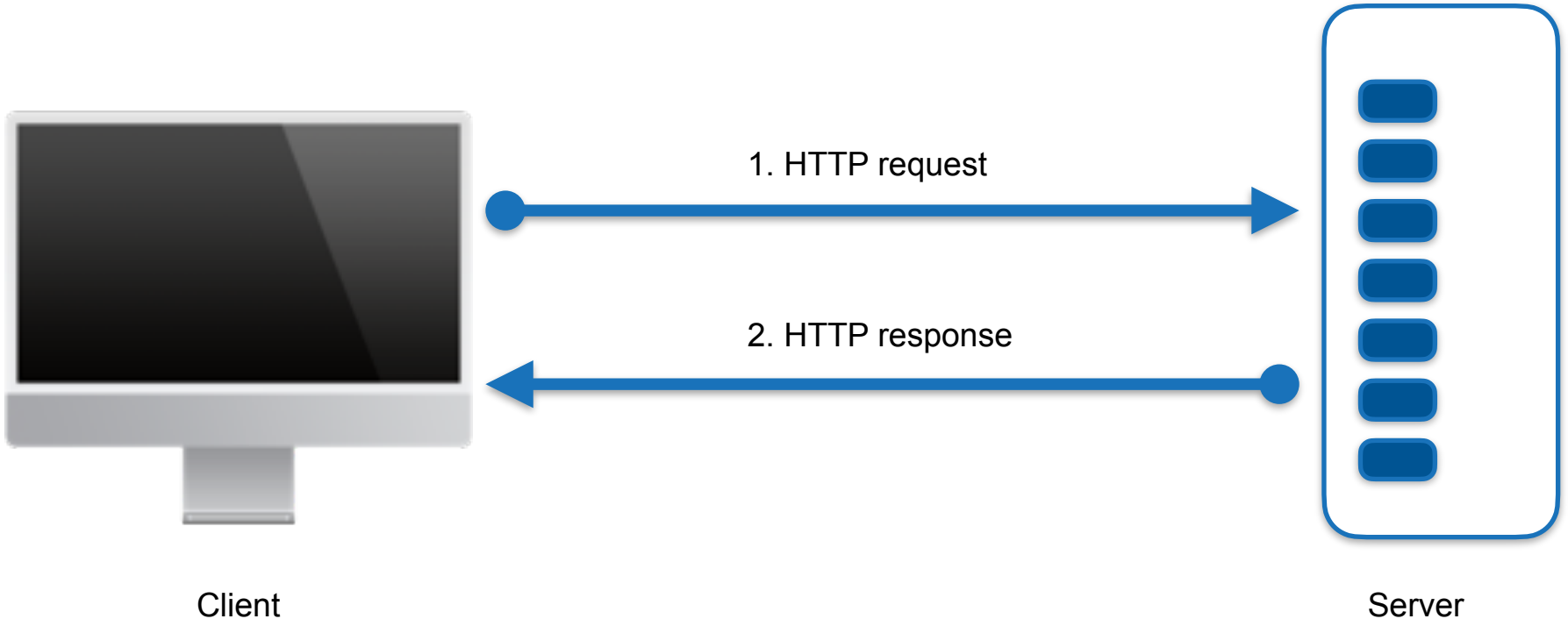
What is HTTP(Hypertext Transfer Protocol)

- It's the application-layer protocol for transferring data like: HTML page, Images, Video…

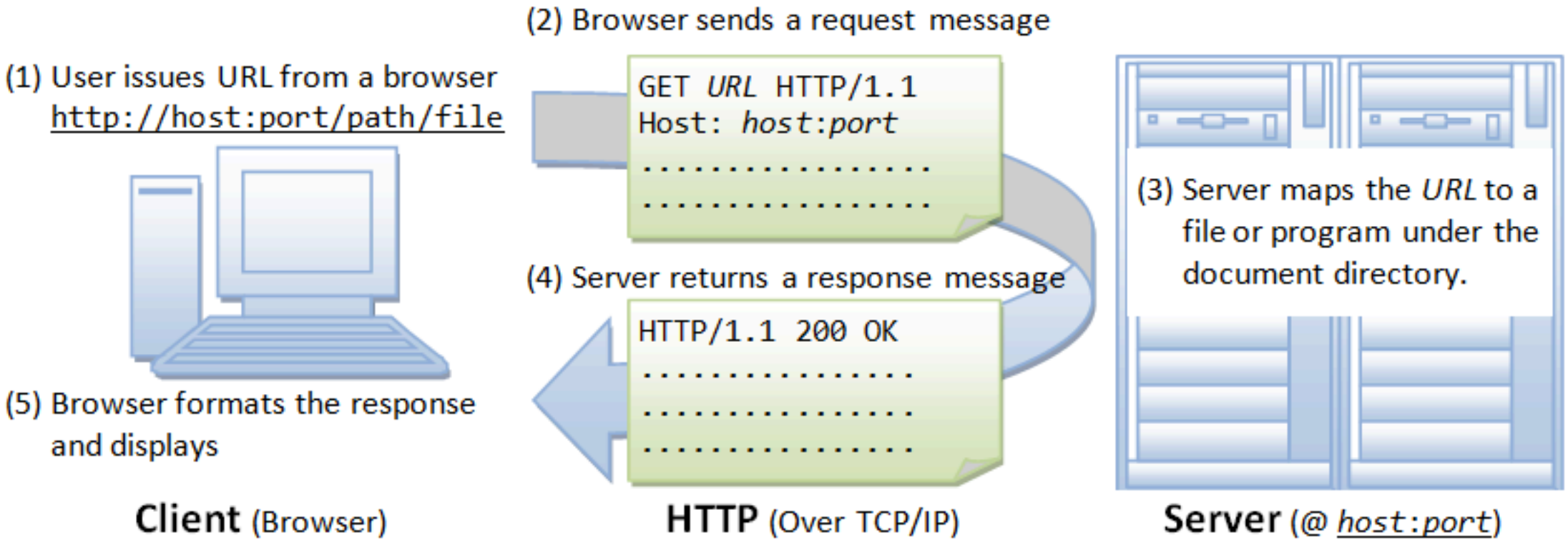


What is HTTP(Hypertext Transfer Protocol)

- Request-response mode/Client-Server network protocol

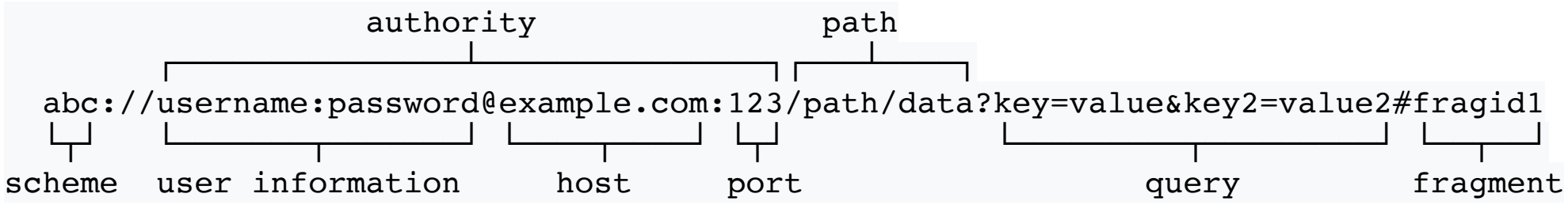


What is HTTP(Hypertext Transfer Protocol)



https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/images/HTTP_Steps.png

URI Syntax



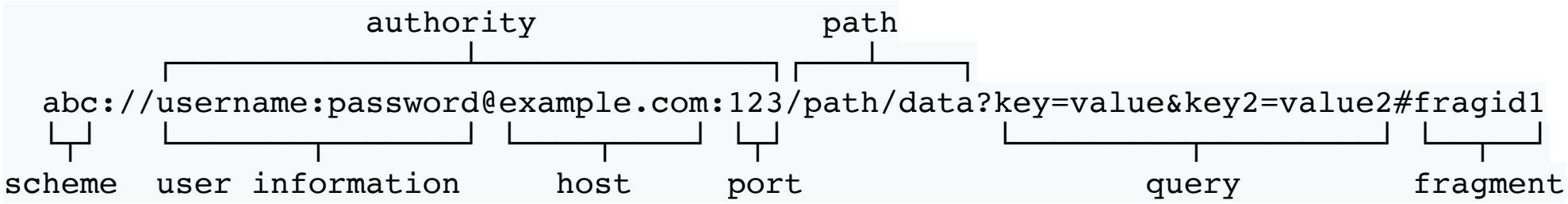
- Scheme - The protocol you are using (non-empty)
- Host - Host name or ip number
- Port - TCP port number that protocol server is using (optional)
- Path - Path reference of object on server
- Query - Query key-value pair (optional)
- Fragment - Used often in sub section title (optional)

URI Syntax

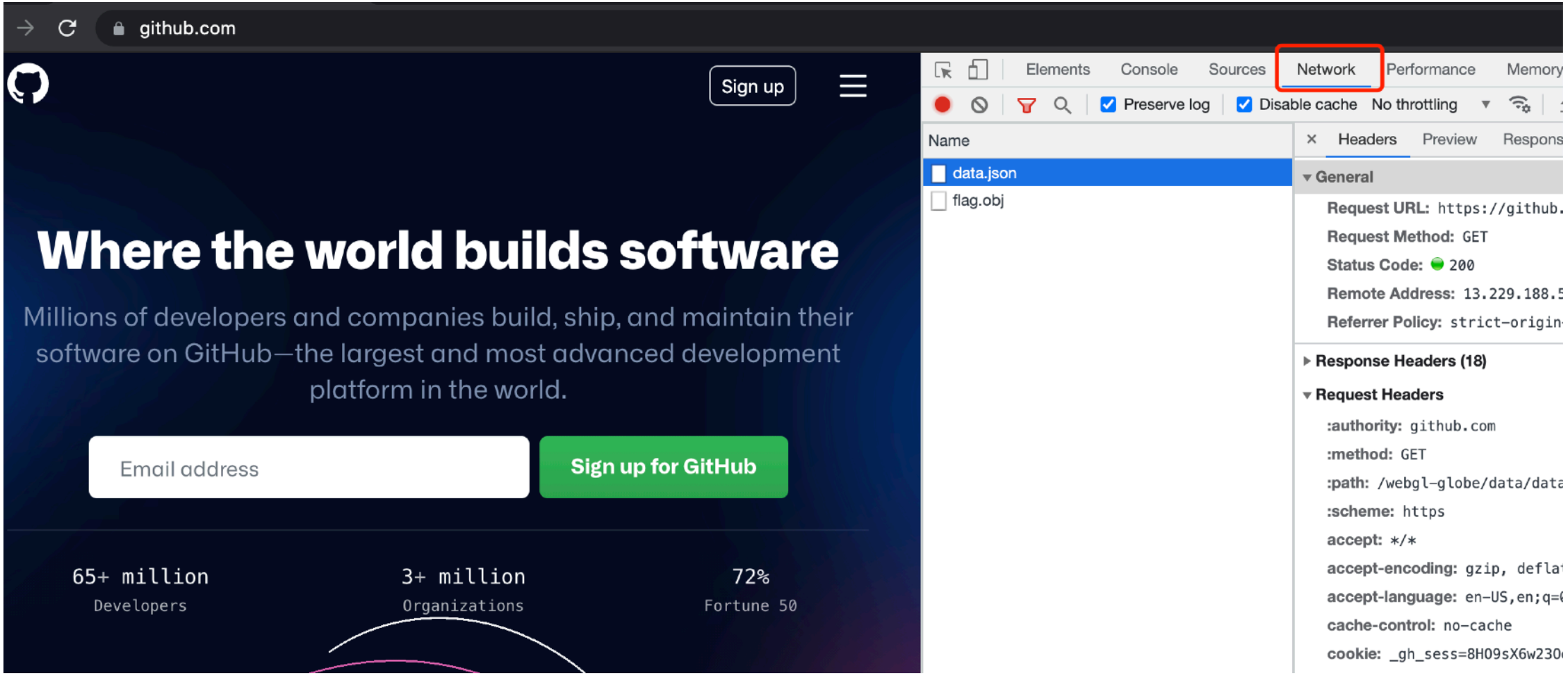
https://play.google.com/log?format=json&hasfast=true&authuser=0

URL

- Special type of URI
- When talking about web addresses, we usually use “URL”



HTTP Request & Response in Chrome



https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#http_messages

HTTP Method

- **GET** Can use the GET request to get a web resource from the server.
- **POST** Used to create a new resource on the server.
- **PUT** Used to update existing resource identified by the URI.
- **DELETE** Used to delete the data on the server.
- …

HTTP Response code

Code	Code Description
1xx: Infomational	It means the request has been received and the process is continuing.
2xx: Success	It means the action was successfully received, understood, and accepted.
3xx: Redirection	It means further action must be taken in order to complete the request.
4xx: Client Error	It means the request contains incorrect syntax or cannot be fulfilled.
5xx: Server Error	It means the server failed to fulfill an apparently valid request.

RESTful

thoughtworks

What is REST?

https://restfulapi.net/

https://restfulapi.net/resource-naming/

What is REST

REST is acronym for **RE**presentational **S**tate **T**ransfer.

It is **architectural style** and was first presented by Roy Thomas Fielding in 2000 in his famous dissertation.

- Every URI represents the **resource**
- Transfer the **representational** of the resource between client and server
- To make the “representational state transfer”, client use **HTTP verbs**(GET, POST, PUT, DELETE…) to let the server process the resource

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_1

Resource

- Any information that can be named can be called a resource
  - A document or image, a temporal service, an object and so on.
  - REST uses **resource identifier** to identify the particular resource involved in an interaction between components.
- The name of the resource is **noun** not a verb

Resource

A collection of users (use plural)

/users

Locate user id 123 of users

/users/123

Filter user with age 23 and gender is male

/users?age=23&gender=male

Resource

Locate all orders of user 123

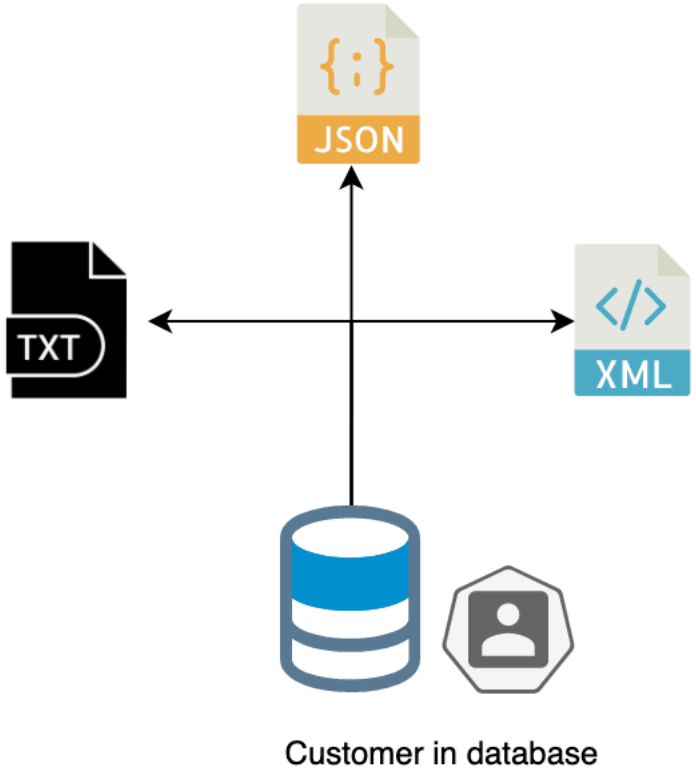
/users/123/orders

Locate the order id 345 of user 123

/users/123/orders/345

Representation

- Resource can be represented by different mode: JSON, XML, TEXT…
- Use Content-Type to refer the type of resource in HTTP header
- The client can use representation to access or manipulate the resource



JSON

- JSON (JavaScript Object Notation) is a lightweight data-interchange format
- JSON is built on two structures:
  - A collection of name/value pairs.
  - An ordered list of values.

JSON

- JSON (JavaScript Object Notation) is a lightweight data-interchange format
- JSON is built on two structures:
  - A collection of name/value pairs.
  - An ordered list of values.


```
{
  "userId": 1,
  "id": 1,
  "title": "sunt a",
  "body": "quia etm eveniet architecto"
}
```

JSON

- JSON (JavaScript Object Notation) is a lightweight data-interchange format
- JSON is built on two structures:
  - A collection of name/value pairs.
  - An ordered list of values.

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt a",
  "body": "quia etm eveniet architecto"
}
```

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "delectus aut autem",
    "completed": false
  },
  {
    "userId": 1,
    "id": 2,
    "title": "quis ut nam facilis et officia qui",
    "completed": false
  }
]
```

Use HTTP Method

- HTTP Method for CRUD
  - **POST** /users
  - **GET** /users/{id}
  - **PUT** /users/{id}
  - **DELETE** /users/{id}

API Design

- Search Resource: get customers of age 23
  - HTTP Method:
  - URL:

API Design

- Search Resource: get customers of age 23
  - GET
  - /customers?age=23

API Design

- Delete Resource: delete customer whose id is 123
  - HTTP Method:
  - URL:

API Design

- Delete Resource: delete customer whose id is 123
  - DELETE /customers/123

API Design

- Manipulate Resource: create a transaction with information (from: account124, to: account456, amount:100)
  - HTTP Method:
  - URL:

API Design

- Manipulate Resource: create a transaction with information (from: account124, to: account456, amount:100)

  - POST
  - /transactions

  Request Body:

```
{
  "from": "account124",
  "to": "account456",
  "amount": 100
}
```

Practice

Please design RESTful API

As a manager, I want to create and list all parking boys. So that I can find someone to park cars for the customer.

AC1. I should be able to create parking boy to the system. The parking boy contains the following information:

- `id` : The *ID* is a non-empty `String` representing the unique ID for a parking boy. This field will be created in backend.
- `name` : The *name* is parking boy's name.

Design the API:

- HTTP Method:
- URL:
- Request Body:
- Response Code:
- Response Body:

Thank You ！

thoughtworks