<WA1/>

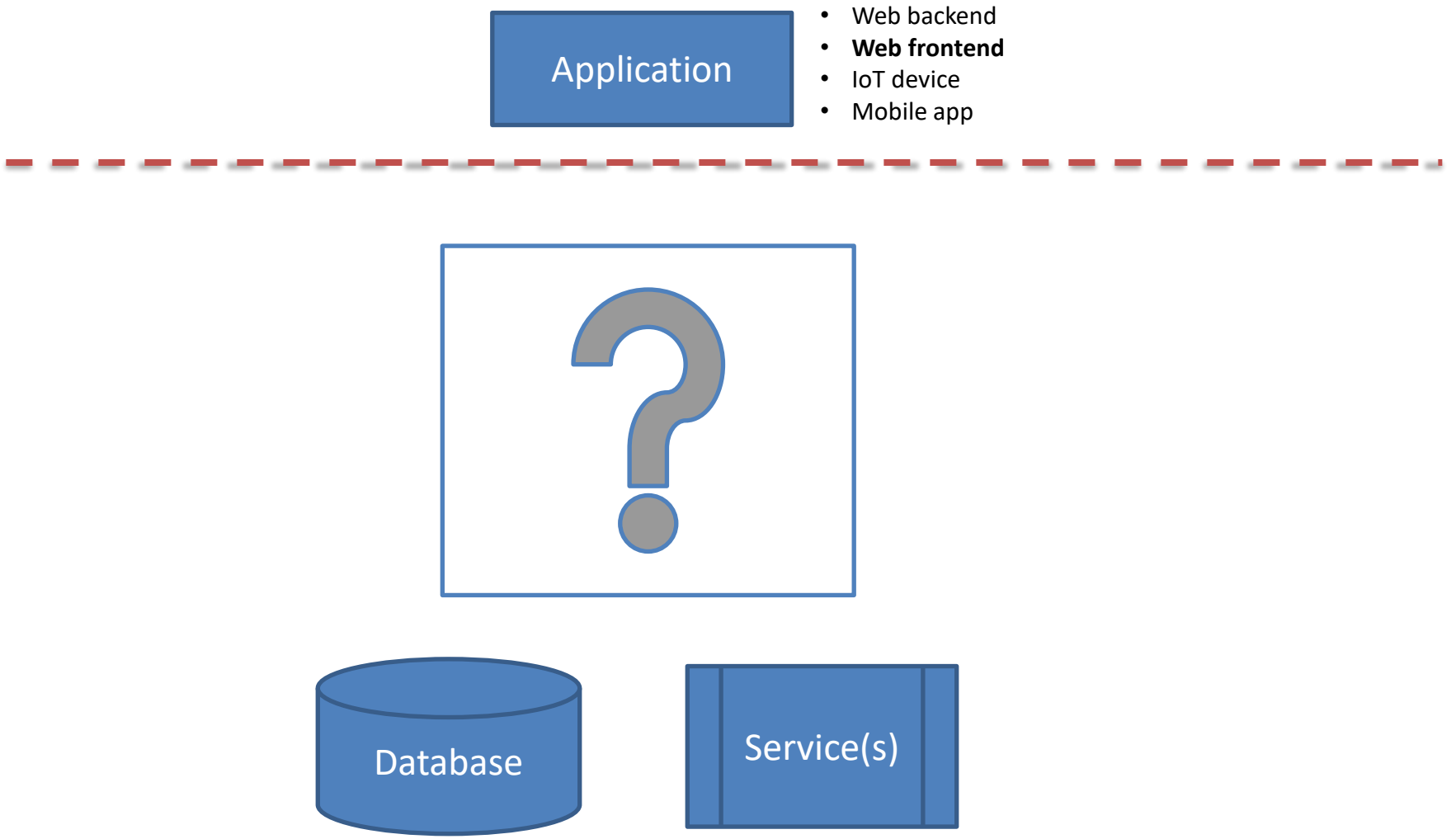
2020

REST API

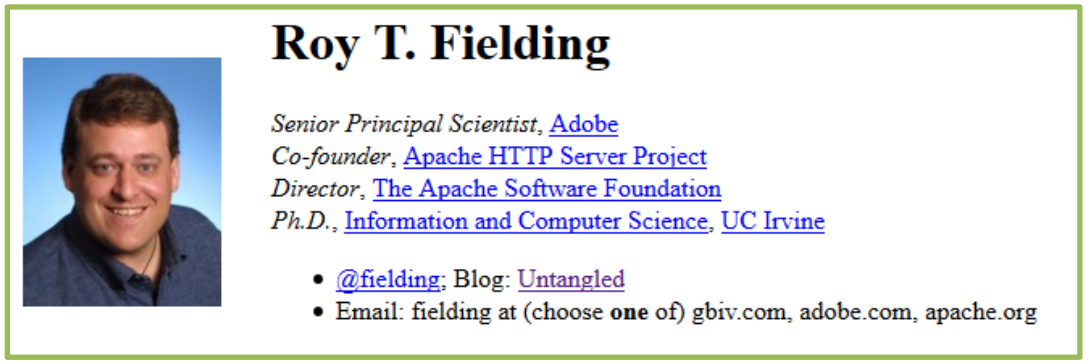
The glue between clients and servers

Enrico Masala
Fulvio Corno
Luigi De Russis

Goal


Application
Web backend
Web frontend
IoT device
Mobile app
Database
Service(s)

REST



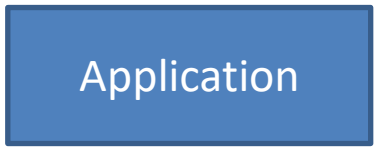
- **Representational State Transfer**
- A style of software architecture for distributed systems
- Platform-independent
  - you don't care if the server is Unix, the client is a Mac, or anything else
- Language-independent
  - C# can talk to Java, etc.
- Standards-based
  - runs on top of HTTP
- Can easily be used in the presence of firewalls

What is a Resource?

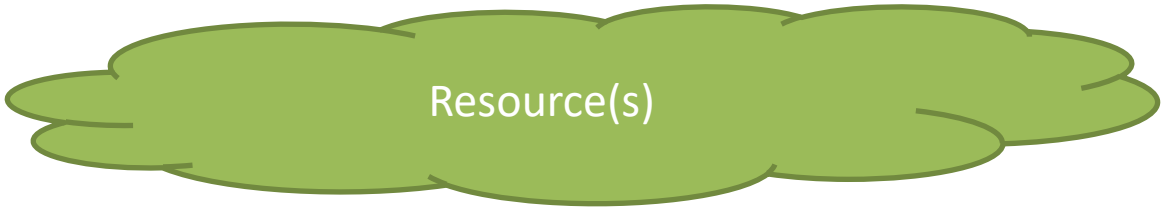
{REST}

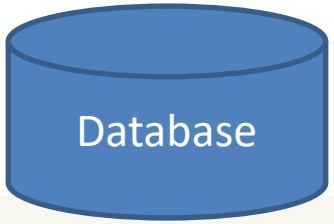
- A resource can be anything that has identity
  - a document or image
  - a service, e.g., "today's weather in New York"
  - a collection of other resources
  - non-networked objects (e.g., people)
- The resource is the **conceptual mapping** to an entity or set of entities, not necessarily the entity that corresponds to that mapping at any particular point in time!

REST Architecture



- Web backend
- **Web frontend**
- IoT device
- Mobile app





Main Principles

{REST}

- Resource: source of specific information
- Mapping: Resources ⇔ URIs
- Client and server exchange *representations* of the resource
  - the same resource *may* have different representations
  - e.g., XML, JSON, HTML, RDF, …

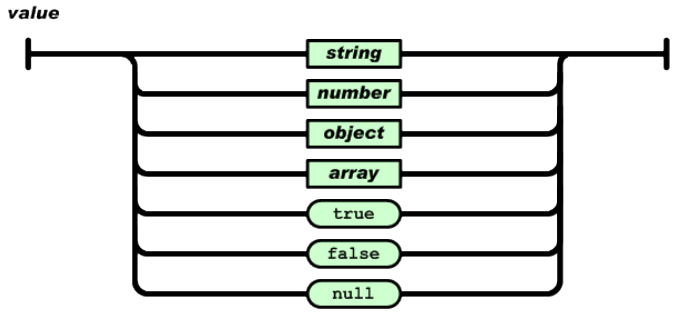
JSON - JavaScript Object Notation

/ˈdʒeɪ·sən/

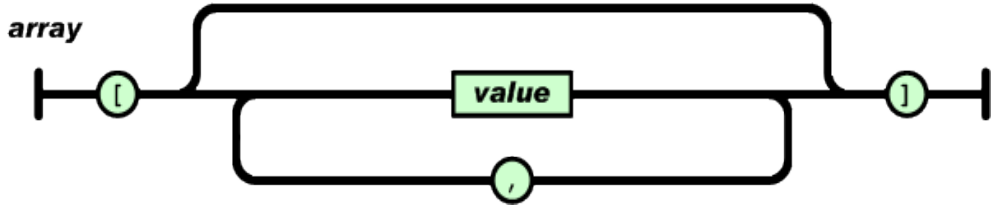
- Lightweight Data Interchange Format
  - Subset of JavaScript syntax for object literals
  - Easy for humans to read and write
  - Easy for machines to parse and generate
  - https://www.json.org/
  - ECMA 404 Standard: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
  - RFC 8259: https://tools.ietf.org/html/rfc8259
- Media type: `application/json`

JSON Logical Structure

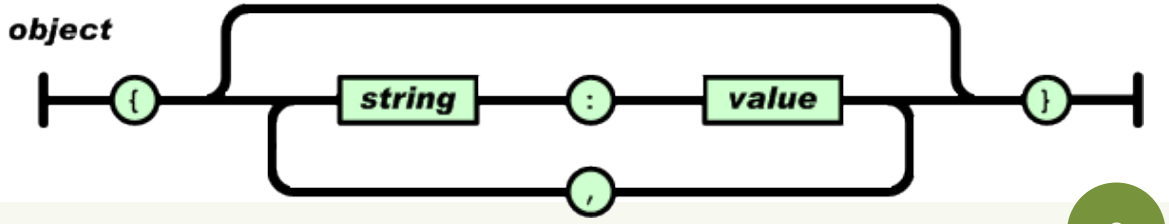
- Primitive types: string, number, true/false/null
  - Strings MUST use "double" quotes, not 'single'



- Composite type – Array: ordered lists of values



- Composite type – Objects: list of key-value pairs
  - Keys are strings (not identifiers)
  - MUST be "quoted"



JSON Example


```
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 555-1234",
        "646 555-4567"
    ]
}
```

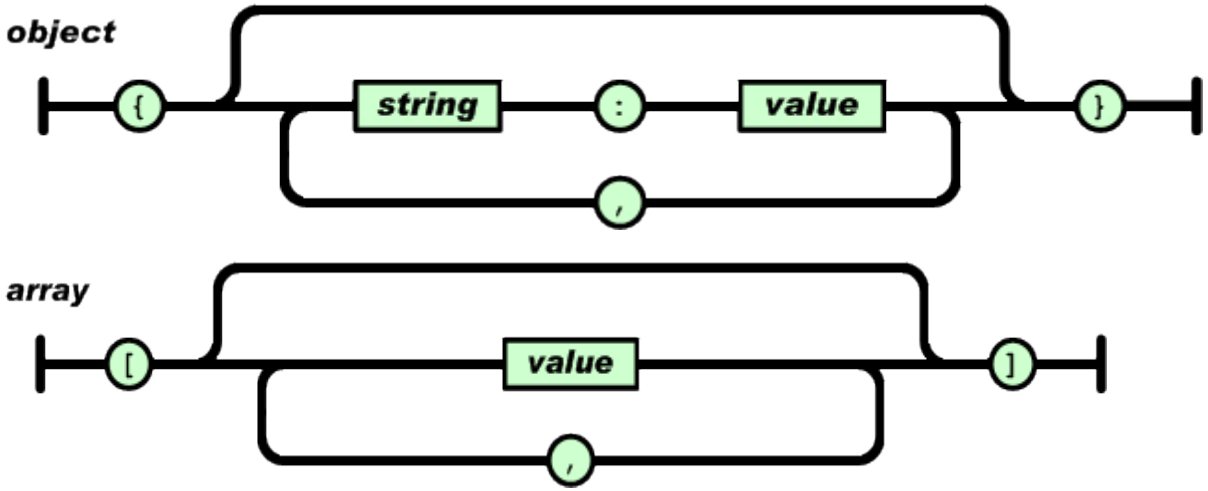
Name/Value Pairs

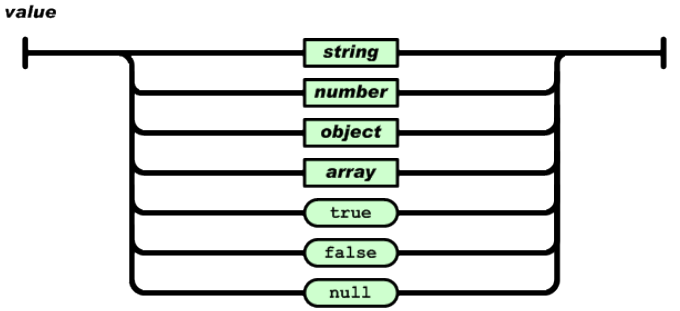
Child properties

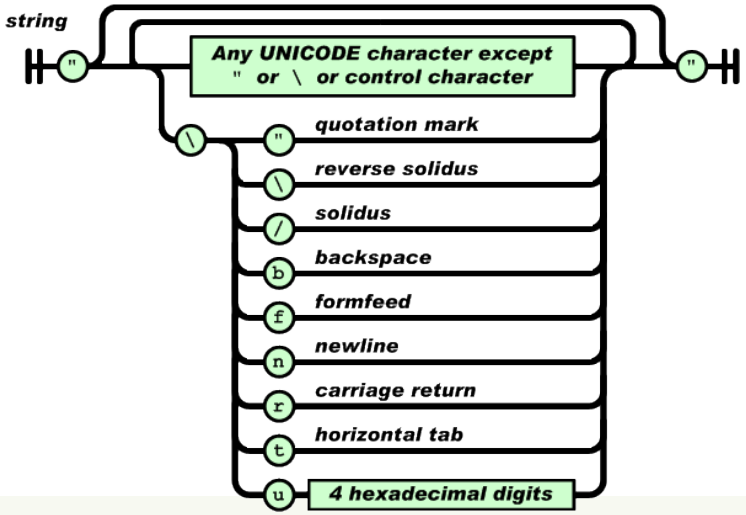
String Array

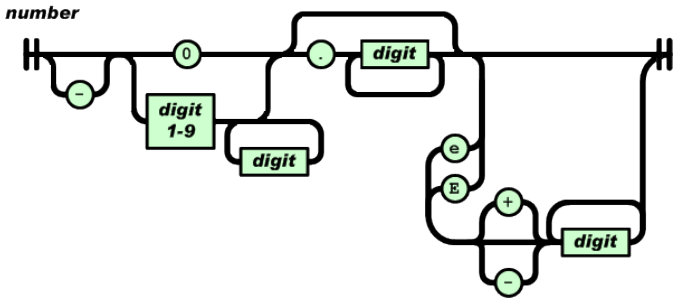
Number data type

JSON Full Syntax







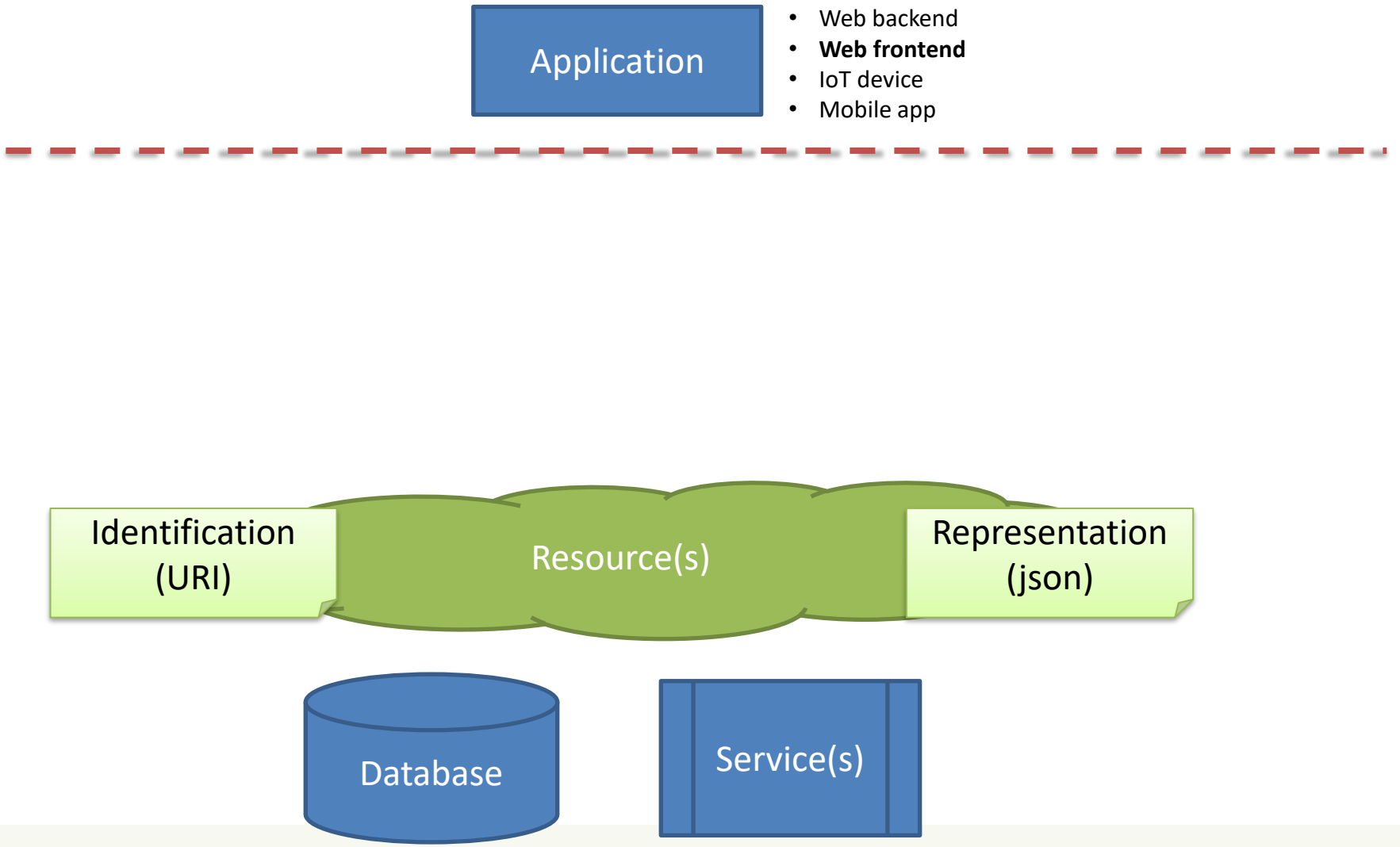


Using JSON in JavaScript

- `JSON.stringify` to convert objects into JSON
  - `const aString = JSON.stringify(myObj)`
  - Works recursively also on nested objects/arrays
  - Excludes function properties (methods) and undefined-valued properties
- `JSON.parse` to convert JSON back into an object
  - `const myObj = JSON.parse(aString)`
  - All created objects have the default `{}` Object prototype
    - Can fix with a *reviver* callback

https://javascript.info/json

REST Architecture



Main Types of Resources

{REST}

- **Collection** resource
  - Represents a set (or list) of resources of the same type
  - Format: /resource
    - `http://api.polito.it/students`
    - `http://api.polito.it/courses`

- **Element** (Item, Simple) resource
  - Represents a single item, and its properties
  - Has some state and zero or more sub-resources
    - Sub-resources can be simple resources or collection resources
  - Format: /resource/identifier
    - `http://api.polito.it/students/s123456`
    - `http://api.polito.it/courses/01zqp`

Best Practice

{REST}

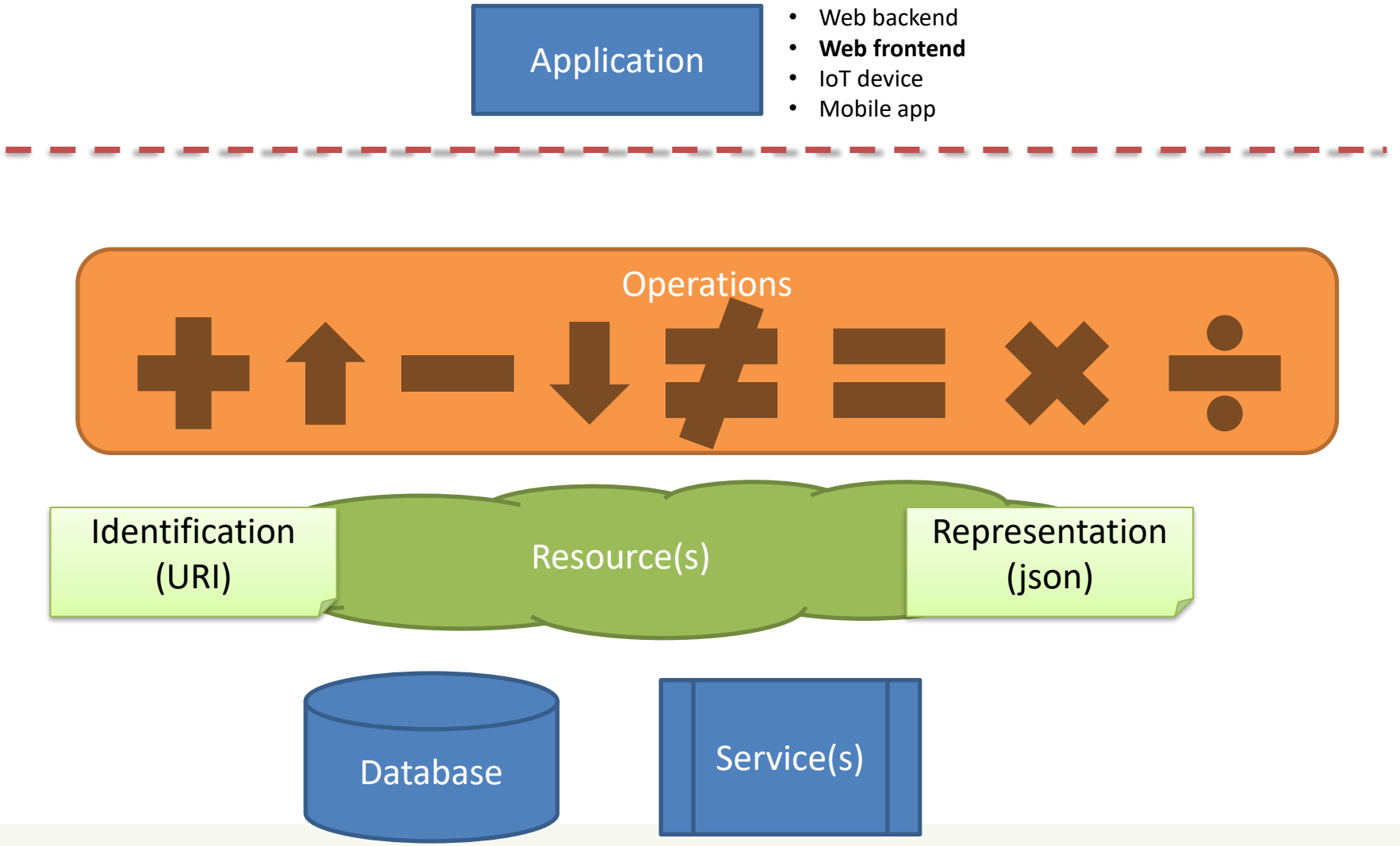
- Nouns (not verbs)
- Plural nouns
- Concrete names (not abstract)
  - /courses, not /items

Main Principles

{REST}

- **Resources** support **Operations** (Actions)
  - Add
  - Delete
  - Update
  - Find
  - Search
  - …

REST Architecture

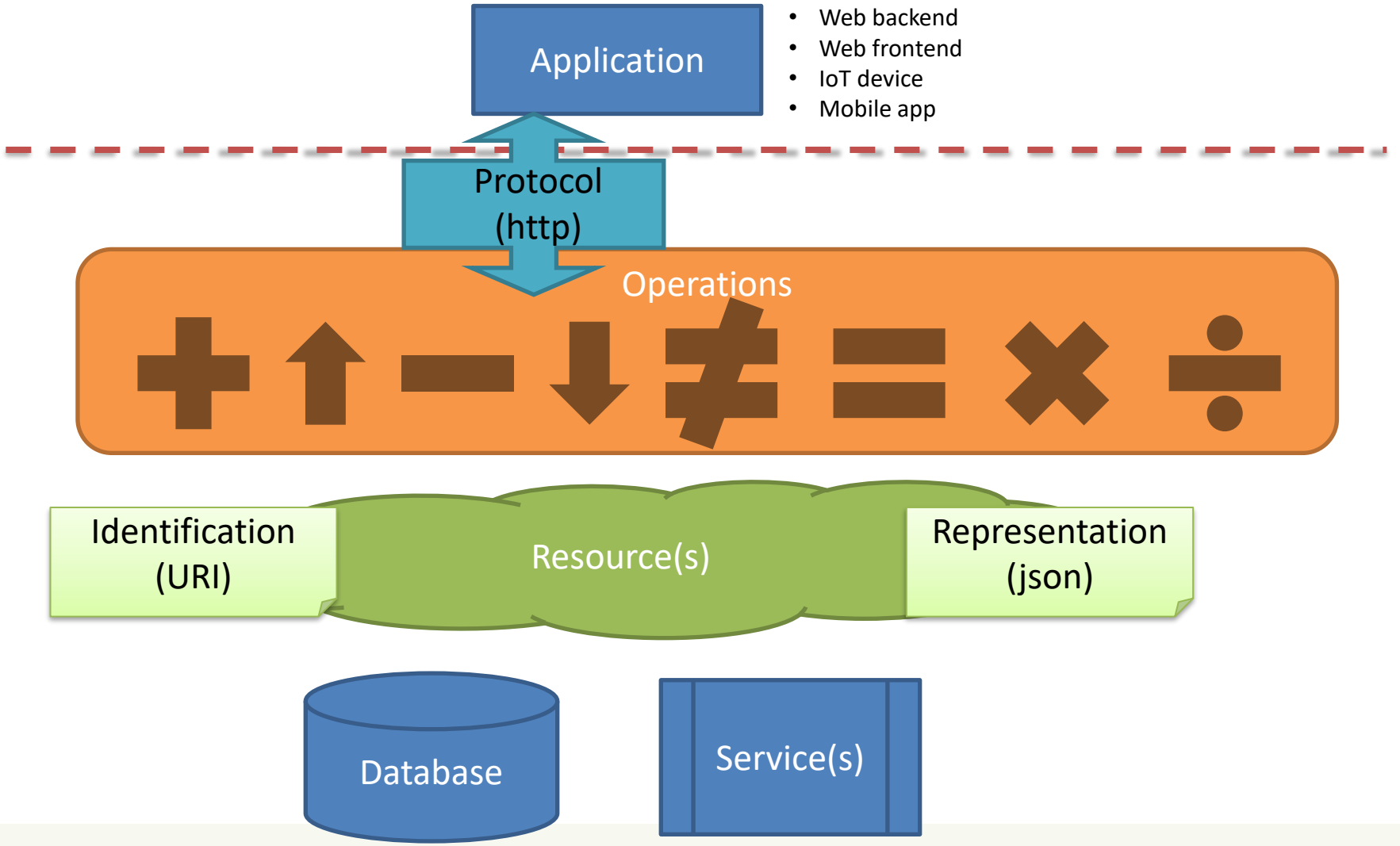


Actions use HTTP Methods

{REST}

- GET
  - Retrieve the representation of the resource (in the HTTP response body)
  - Collection: the list of items
  - Element: the properties of the element
- POST
  - Create a new resource (data in the HTTP request body)
  - Use a URI for a Collection
- PUT
  - Update an existing element (data in the HTTP request body)
  - Mainly for elements' properties
- DELETE

REST Architecture



Actions on Resources: Example

Resource	GET	POST	PUT	DELETE
/dogs	List dogs	Create a new dog	Bulk update dogs (avoid)	Delete all dogs (avoid)
/dogs/1234	Show info about the dog with id 1234	ERROR	If exists, update the info about dog #1234	Delete the dog #1234

Relationships

{REST}

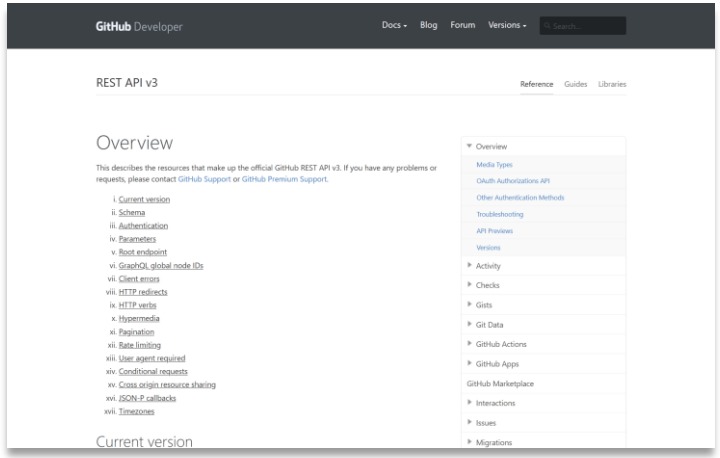
- A given Element may have a (1:1 or 1:N) relationship with other Element(s)
- Represent with: /resource/identifier/resource
- `http://api.polito.it/students/s123456`/courses (list of courses followed by student `s123456`)
- `http://api.polito.it/courses/01qzp/students` (list of students enrolled in course `01qzp`)

Representations

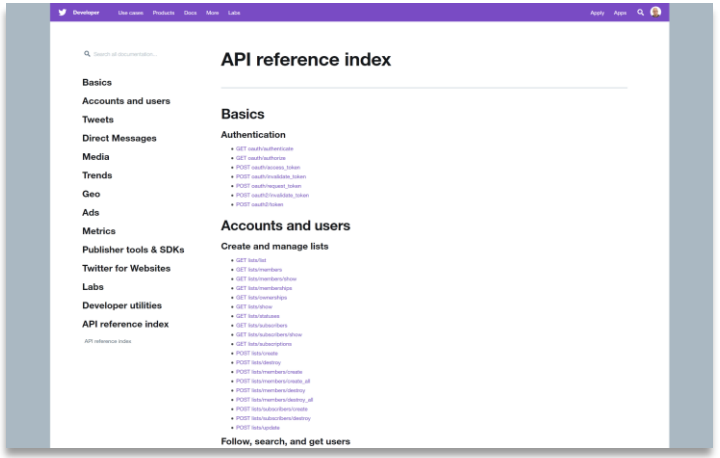
{REST}

- Returned in GET, sent in PUT/POST
- Different formats are possible
- Mainly: XML, JSON
  - But also: SVG, JPEG, TXT, …
  - In POST: URL-encoding
- Format may be specified in
  - Request headers
    - **`Accept: application/json`**
  - URI extension
    - `http://api.polito.it/students/s123456.json`
  - Request parameter
    - `http://api.polito.it/students/s123456?format=json`

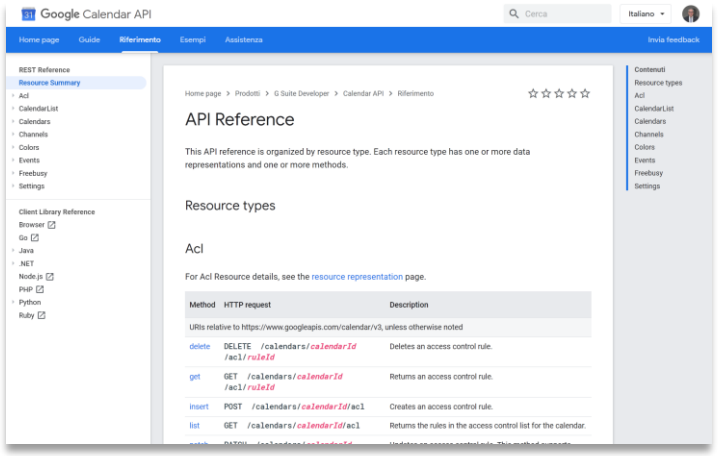
Real World Examples



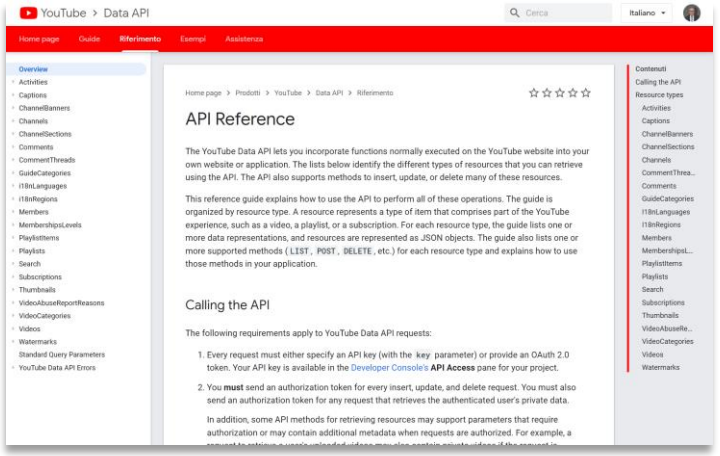
https://developer.github.com/v3/



https://developer.twitter.com/en/docs/api-reference-index



https://developers.google.com/calendar/v3/reference/



https://developers.google.com/youtube/v3/docs

Complex resource search

{REST}

- Use `?`*`parameter`*`=`*`value`* for more advanced resource filtering (or search)
  - E.g., `https://api.twitter.com/1.1/statuses/user_timeline.json`**`?screen_name=twitterapi&count=2`**

Errors

{REST}

- When errors or exceptions are encountered, use meaningful HTTP Status Codes
  - The Response Body may contain additional information (e.g., informational error messages)

```
{
   "developerMessage" : "Verbose, plain language description of
the problem for the app developer with hints about how to fix
it.",
   "userMessage":"Pass this message on to the app user if
needed.",
   "errorCode" : 12345,
   "more info": "http://dev.teachdogrest.com/errors/12345"
}
```

Authentication

Twitter Streaming API

```
Authorization: OAuth
oauth_consumer_key="xvz1evFS4wEEPTGEFPHBog", …
```

Amazon Web Services API

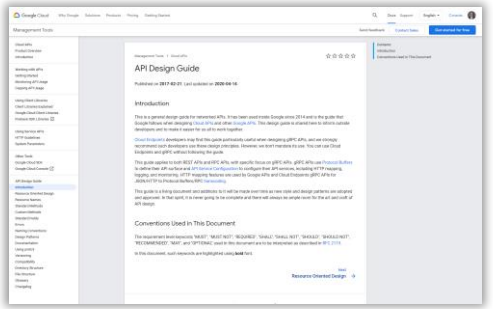
```
Authorization: AWS
AKIAIOSFODNN7EXAMPLE:frJIUNo//yllqDzg=
```

Google API

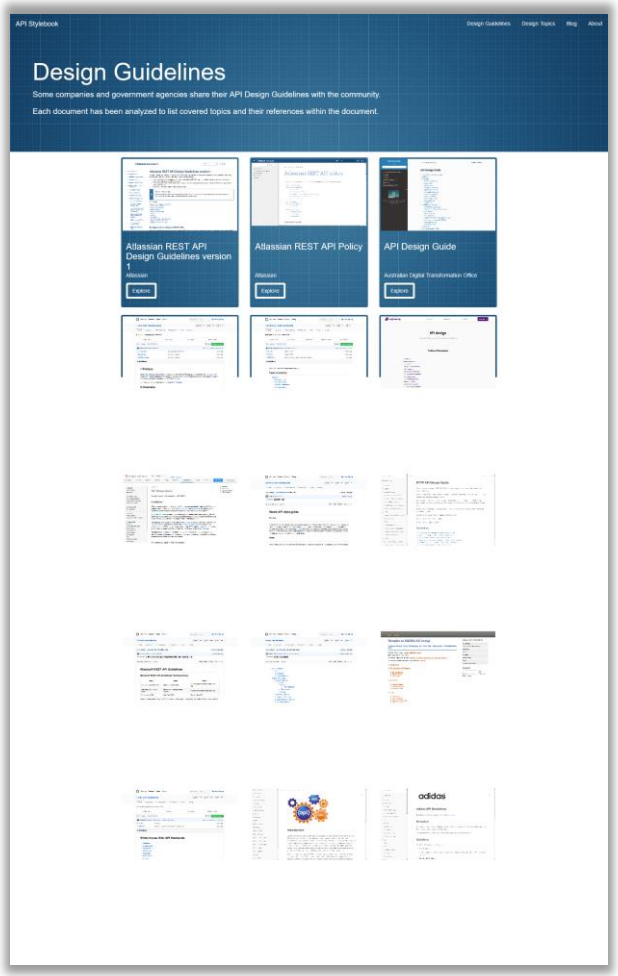
```
Authorization: Bearer 1/fFBGRNJru1FQd44AzqT3Zg
```

API Design

- How to design a set of APIs for your application?
- Practical guidelines, with applied standard practices
- Suggestion: Google API Design Guide
  - https://cloud.google.com/apis/design/



http://apistylebook.com/design/guidelines/



API Design Flow

1. Determine what types of **resources** an API provides.
2. Determine the **relationships** between resources.
3. Decide the resource **name schemes** based on types and relationships.
4. Decide the **resource schemas**.
5. Attach minimum set of **methods** to resources.

API Service Name	Collection ID	Resource ID	Resource ID	Resource ID
//mail.googleapis.com	/users	/name@example.com	/settings	/customFrom

Example (Gmail API)

- API service: `gmail.googleapis.com`
- A collection of users: `users/*`. Each user has the following resources.
  - A collection of messages: `users/*/messages/*`.
  - A collection of threads: `users/*/threads/*`.
  - A collection of labels: `users/*/labels/*`.
  - A collection of change history: `users/*/history/*`.
  - A resource representing the user profile: `users/*/profile`.
  - A resource representing user settings: `users/*/settings`.

Standard Methods

Standard Method	HTTP Mapping	HTTP Request Body	HTTP Response Body
List	GET <collection URL>	N/A	Resource* list
Get	GET <resource URL>	N/A	Resource*
Create	POST <collection URL>	Resource	Resource*
Update	PUT or PATCH <resource URL>	Resource	Resource*
Delete	DELETE <resource URL>	N/A	google.protobuf.Empty**

Let's read:
https://cloud.google.com/apis/design/standard_methods

Guidelines (1/2)

URL Design	
Plural nouns for collections	/dogs
ID for entity	/dogs/1234
Associations	/owners/5678/dogs
HTTP Methods	POST GET PUT DELETE
Bias toward concrete names	/dogs (not animals)
Multiple formats in URL	/dogs.json /dogs.xml
Paginate with limit and offset	?limit=10&offset=0
Query params	?color=red&state=running
Partial selection	?fields=name,state
Use medial capitalization	"createdAt": 1320296464 myObject.createdAt;
Use verbs for non-resource requests	/convert?from=EUR&to=CNY&amount=100
Search	/search?q=happy%2Blabrador
DNS	api.foo.com developers.foo.com

Guidelines (2/2)

Versioning	
Include version in URL	/v1/dogs
Keep one previous version long enough for developers to migrate	/v1/dogs /v2/dogs

Errors	
Status Codes	200 201 304 400 401 403 404 500
Verbose messages	{"msg": "verbose, plain language hints"}

Client Considerations	
Client does not support HTTP status codes	?suppress_response_codes=true
Client does not support HTTP methods	GET /dogs?method=post GET /dogs GET /dogs?method=put GET /dogs?method=delete
Complement API with SDK and code libraries	1. JavaScript 2. … 3. …

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/