REST and JSON

JSON

JavaScript Object Notation

- Goal: Transfer data
  - Between computers / processes
  - Between programming languages
- Best for “data” object.
  - Key/Value pairs. Dictionaries, Arrays
- Not great for object relationships
  - Linked Lists, Graphs, OOP

JSON vs XML

fight!

```xml
<?xml version="1.0" encoding="UTF-8"?>
<cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
  <cas:authenticationSuccess>
    <cas:user>fischerm</cas:user>
      <cas:attributes>
        <cas:dbkey>1111111111111</cas:dbkey>
        <cas:emplid>22222222</cas:emplid>
        <cas:activestudent>0</cas:activestudent>
        <cas:activeemployee>1</cas:activeemployee>
      </cas:attributes>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

XML

```json
{
  "serviceResponse": {
    "authenticationSuccess": {
      "user": "fischerm",
      "attributes": {
        "dbkey": 1111111111111,
        "emplid": 22222222,
        "activestudent": 0,
        "activeemployee": 1
      }
    }
  }
}
```

JSON

JSON vs XML

fight!

```
<?xml version="1.0" encoding
<cas:serviceResponse xmlns:c
  <cas:authenticationSuccess
    <cas:user>fischerm</cas:
      <cas:attributes>
        <cas:dbkey>11111111
        <cas:emplid>2222222
        <cas:activestudent>0
        <cas:activeemployee>
      </cas:attributes>
  </cas:authenticationSucces
</cas:serviceResponse>
```

XML

- XML Strengths
  - Plain Text
  - Flexible
  - Can be defined with formal document definitions
  - Excellent validation tools
- XML Drawbacks
  - Verbose
  - “Automatic” ideals never fully realized
  - Flexibility leads to ambiguous mapping to internal data objects

JSON vs XML

- JSON Strengths
  - Plain Text
  - Literally JavaScript Object Notation
  - Lighter weight formatting. Simple Key/Value
  - Arrays explicitly supported. Allows for cleaner mapping to programming language objects
  - JSON Drawbacks
    - No Comments 😢
    - Validation was an afterthought
    - Can’t encode complex relationships

```
"serviceResponse": {
  "authenticationSuccess": {
    "user": "fischerm",
    "attributes": {
      "dbkey": 1111111111111,
      "emplid": 22222222,
      "activestudent": 0,
      "activeemployee": 1
    }
  }
}
```

JSON

JSON

Formal Grammar

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

- Parent construct is either an object or array
  - `{ … }` for Object
  - `[ … ]` for Array

```
{
  "name": "Mark",
  "netid": "fischerm"
}
```

```
[
  {"name": "Mark", "netid": "fischerm"},
  {"name": "Rhonda", "netid": "rroyse"},
  {"name": "Tim", "netid": "tdarby"},
]
```

JSON
Formal Grammar

- Whitespace is unimportant
  - These are all equivalent

```
{
  "name": "Mark",
  "netid": "fischerm"
}
```

```
{"name": "Mark", "netid": "fischerm"}
```

```
{"name":"Mark","netid":"fischerm"}
```

```
{
"name":
"Mark",
"netid":
"fischerm"
}
```

JSON

Objects

- Objects are defined with curly braces
- Key/Value pairs are separated by a colon
- Keys are strings
- Values can be strings, numbers, boolean, null, objects, or arrays
- `"key": "value"` pairs separated by commas
- Trailing commas are not allowed

```
{
   "name": "Mark",
   "netid": "fischerm"
}
```

```
{
   "name": "Mark",
   "netid": "fischerm",
}
```

JSON

Objects

- Keys must be strings
- Double Quotes are required

```
{
   "name": "Mark",
   "netid": "fischerm"
}
```

```
{
   'name': 'Mark',
   'netid': 'fischerm'
}
```

JSON

Arrays

- Arrays are defined by square brackets
- Comma separated list of values
- Values can be strings, numbers, boolean, null, objects, or arrays

```
[ 1, 2, 3 ]
```

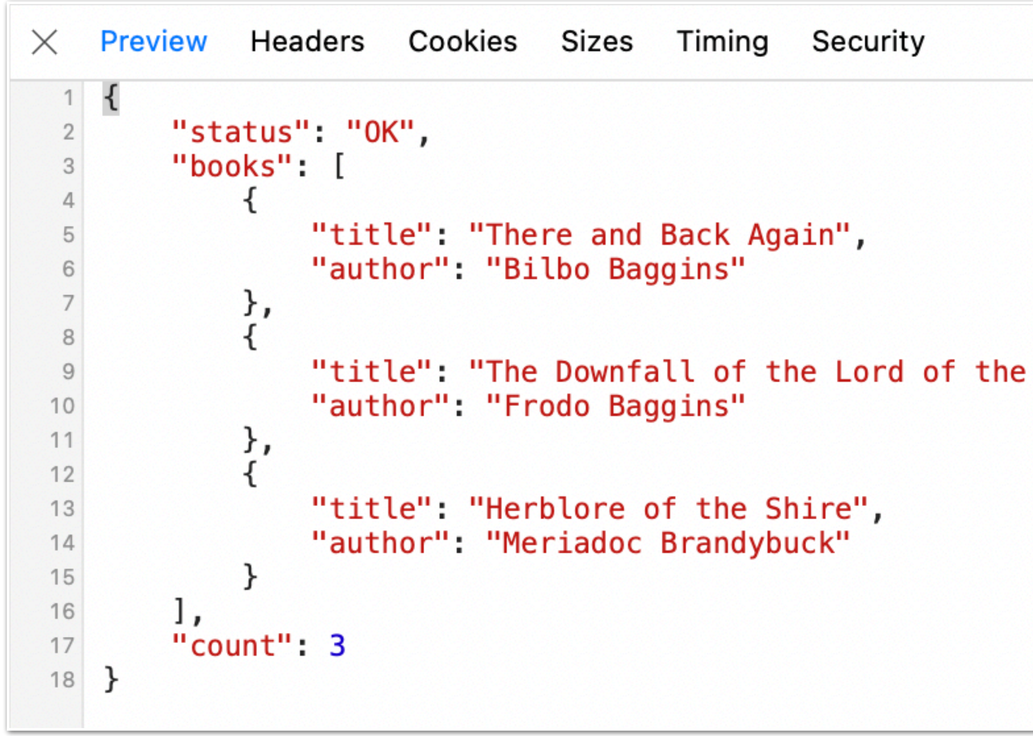
```
[
  "one",
  "two",
  "three"
]
```

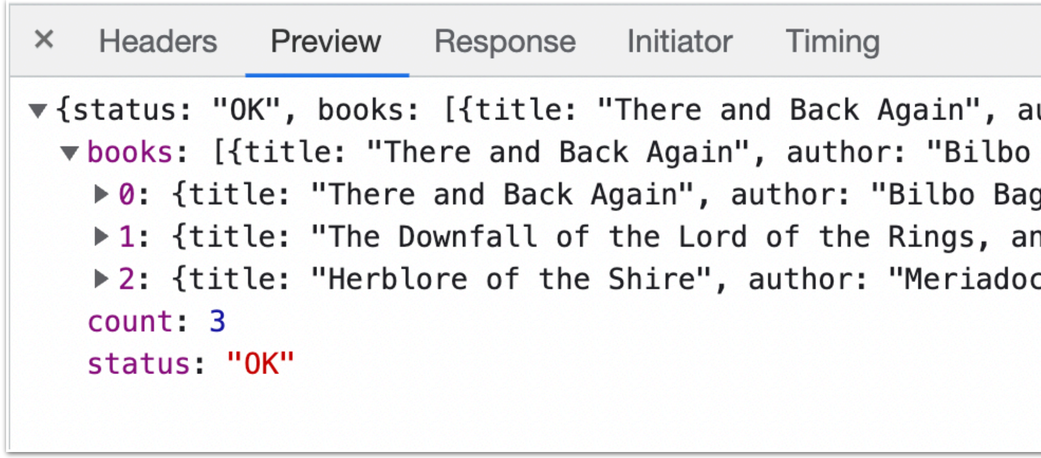
```
[
  {"numeral": 1, "name": "one", "odd": true},
  {"numeral": 2, "name": "two", "odd": false},
  {"numeral": 3, "name": "three", "odd": true}
]
```

JSON

For Humans

- Many code editors will auto-format JSON for you
- Postman will pretty-print JSON output or show raw
- Some browsers display pretty-print JSON, others render it as an expandable tree





JSON

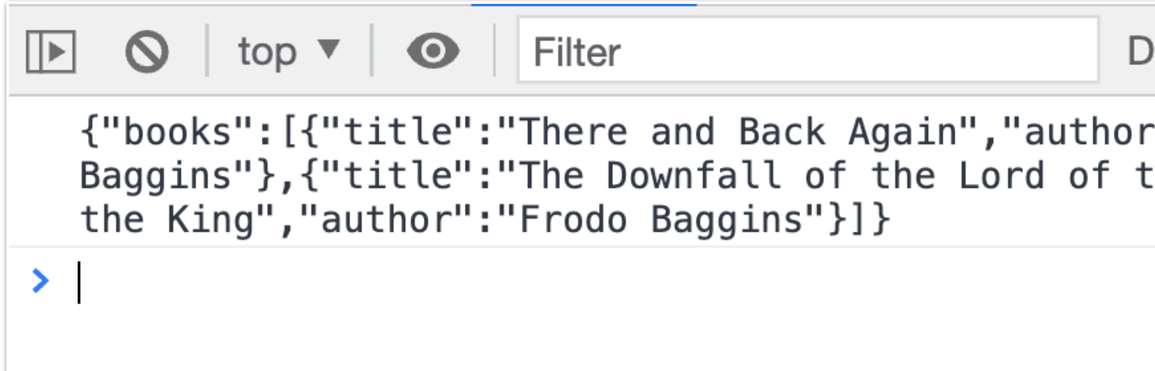
In JavaScript

- Language level JSON object
- Can’t create instances with `new`, static methods only
- JavaScript → JSON

```
JSON.stringify( var )
```

```
let obj = {
    'books': [
        {
            'title': "There and Back A
            'author': "Bilbo Baggins"
        },
        {
            'title': "The Downfall of
King",
            'author': "Frodo Baggins"
        },
    ]
}

console.log( JSON.stringify(obj) )
```



JSON

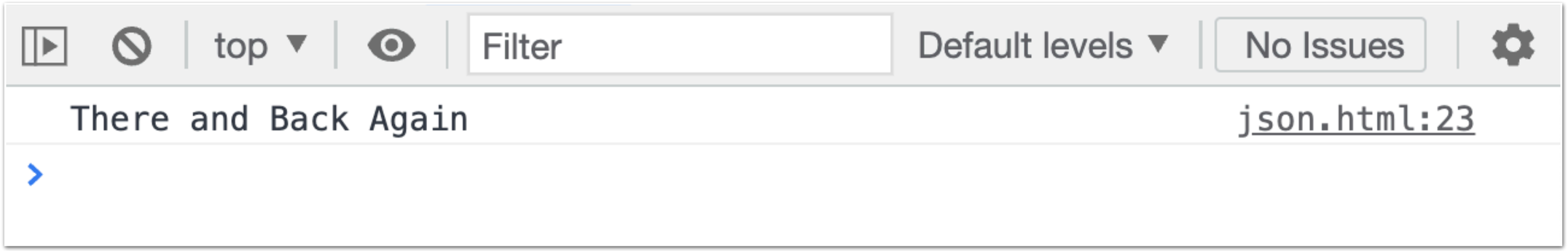
In JavaScript

- JSON → JavaScript

```
JSON.parse( string )
```

```
let jsonString = '{"title":"There and Back Again","author":"Bilbo Baggins"}'

book = JSON.parse( jsonString )
console.log( book.title )
```



JSON

In JavaScript

- JSON is always valid JavaScript
- JavaScript is *NOT* always valid JSON
- Example: Trailing commas are fine in JavaScript, but are invalid in JSON

```
let obj = {
    'books': [
        {
            'title': "There and Back A
            'author': "Bilbo Baggins"
        },
        {
            'title': "The Downfall of
King",
            'author': "Frodo Baggins"
        },
    ]
}

console.log( JSON.stringify(obj) )
```

JSON

In Python

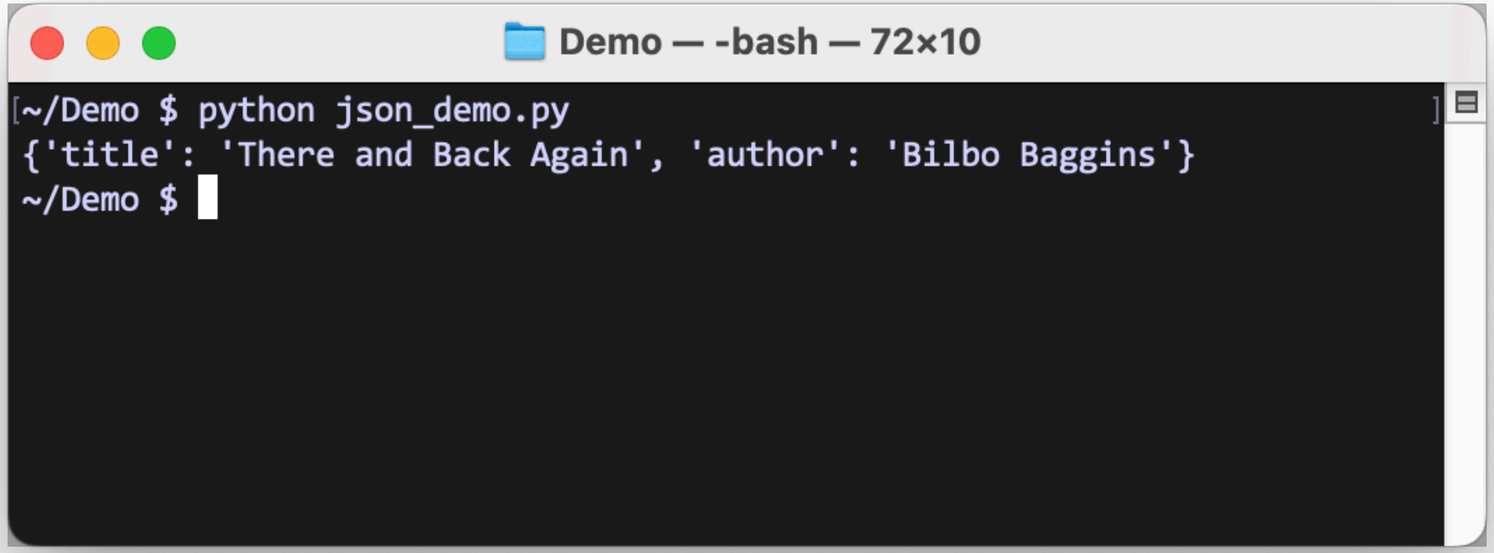
- json module is part of the Python standard library
- JSON → Python

```
json.loads( var )
```

```
import json

jsonString = '{"title":"There and Back Again","auth

print(json.loads(jsonString))
```



JSON
In Python

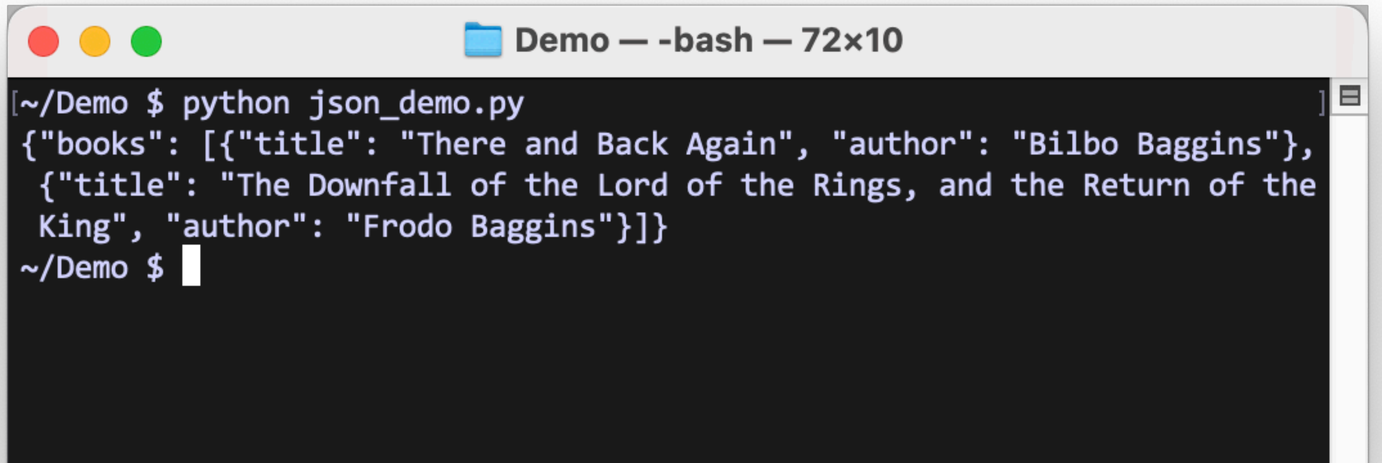
- Python → JSON

```
json.dumps( var )
```

```
import json

obj = {
    "books": [
        {"title": "There and Back Again", "auth
        {
            "title": "The Downfall of the Lord
King",
            "author": "Frodo Baggins",
        },
    ]
}

print(json.dumps(obj))
```



JSON

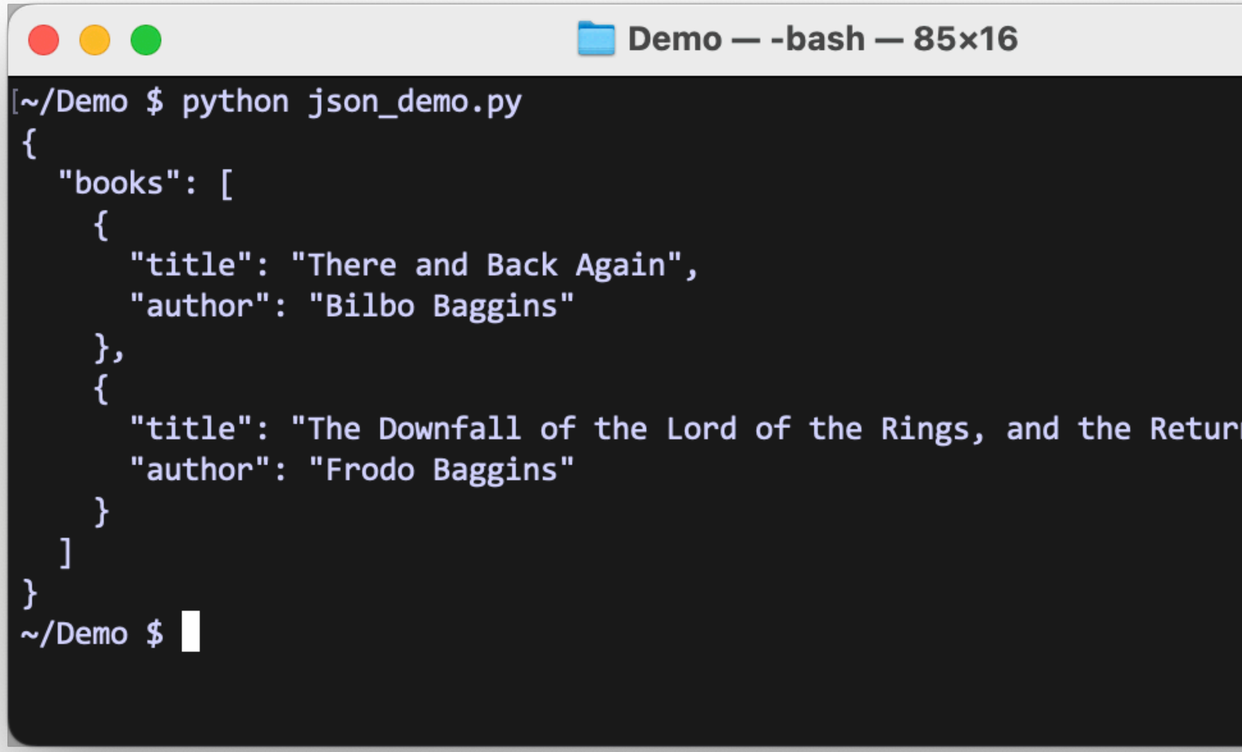
In Python

- Python → JSON
- Optional `indent` argument to `dumps` will pretty-print your JSON strings from Python

```
import json

. . .

print(json.dumps(obj, indent=2))
```



REST

Representational **S**tate **T**ransfer

REST

Representational **S**tate **T**ransfer

- JSON objects = DB records
- Send & Receive over HTTP
- URLs = object IDs

REST

Fundamentals

- REST is not a protocol, like HTTP, or SOAP
- REST is an architectural style, defined by a few key principles

```
https://en.wikipedia.org/wiki/Representational_state_transfer
```

REST

Client-Server Architecture

- Separation of concerns
- Decouples user interface from data access and persistence
- Allows for many different architectures for client and server

REST

Uniform Interface

- Requests should identify resources
  - They do so by using a uniform resource identifier (URI)
- Resource manipulation through representations
  - When a client holds a representation of a resource, including any metadata attached, it has enough information to modify or delete the resource's state
- Self-descriptive messages contain metadata about how the client can best use them
- A REST client should then be able to use server-provided links dynamically to discover all the available resources it needs

REST

Statelessness

- Clients can request resources in any order, and every request is stateless or isolated from other requests
- Statelessness refers to a communication method in which the server completes every client request independently of all previous requests
- Implies that the server can completely understand and fulfill the request every time

REST

Layered System

- A client can connect to other authorized intermediaries between the client and server, and it will still receive responses from the server
- Design your RESTful web service to run on several servers with multiple layers such as security, application, and business logic, working together to fulfill client requests
- These layers remain invisible to the client

REST

Cacheability

- As on the World Wide Web, clients and intermediaries can cache responses
- Well-managed caching partially or completely eliminates some client–server interactions, further improving scalability and performance
- The cache can be performed at the client machine in memory or browser cache storage
- Additionally cache can be stored in a Content Delivery Network (CDN)

REST

Semantic HTTP Methods

Method	Description
GET	Get a representation of the target resource's state
POST	Let the host process a resource state sent in the request
PUT	Create or replace the state of a target resource with the state defined in the request
PATCH	Partially update a resource's state
DELETE	Delete the target resource's state
OPTIONS	Describe the available methods

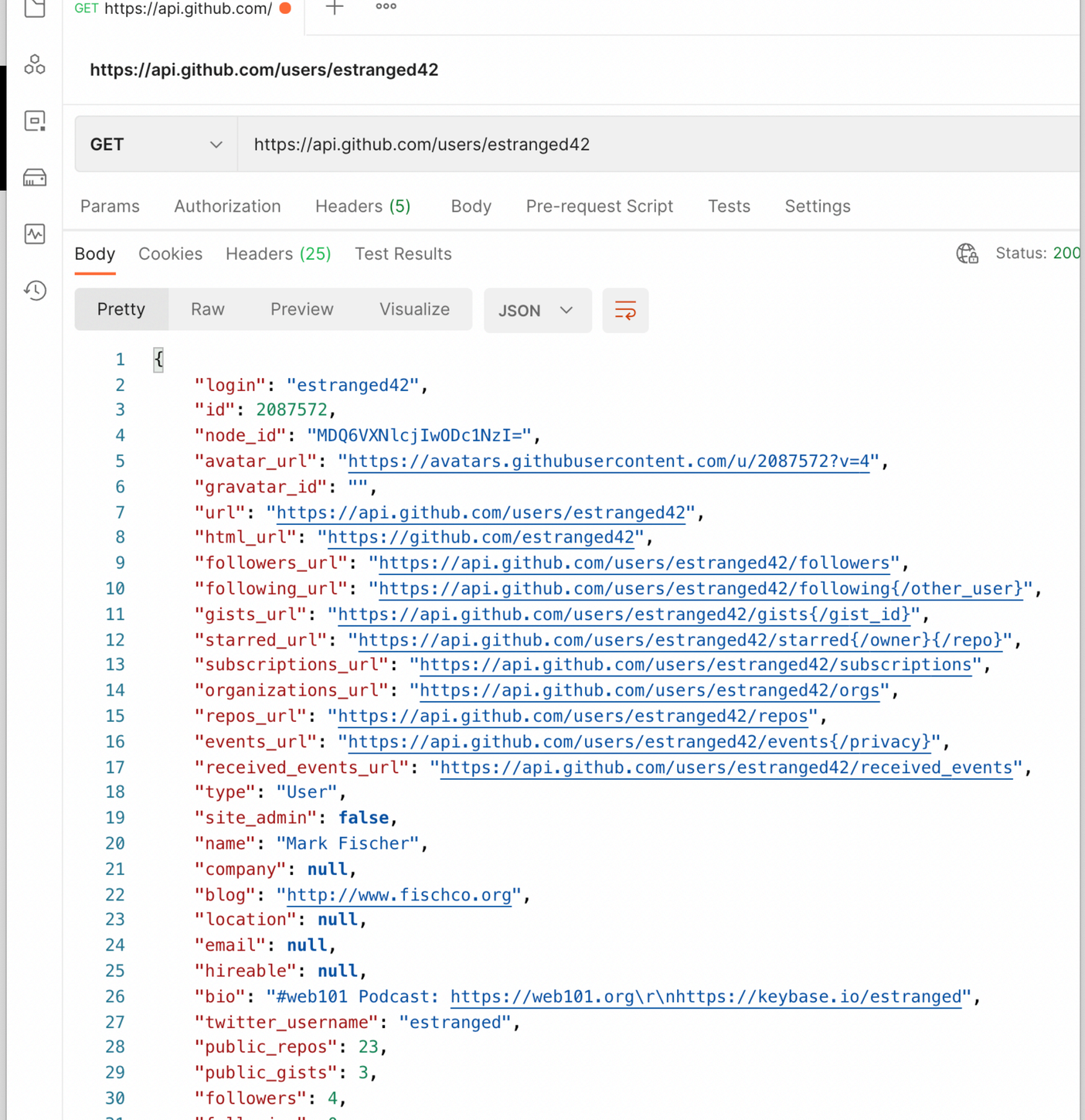
REST

GitHub API

- For example here is the GitHub API call to list basic info about my personal GitHub account

```
GET https://api.github.com/users/estranged42
```

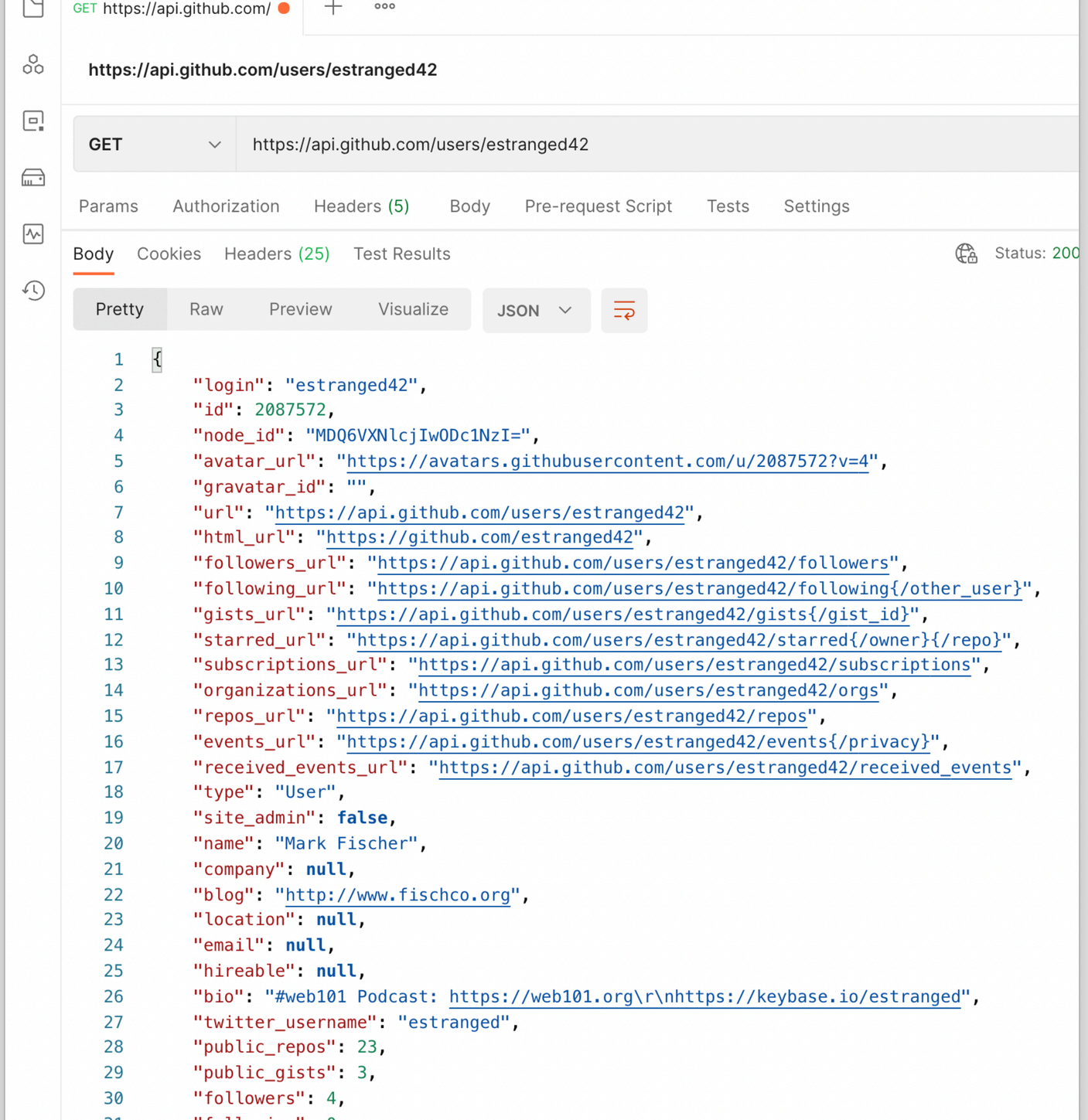
GET https://api.github



REST

GitHub API

- Since I requested a single thing, I received a dictionary in response



REST

GitHub API

- If I request all of my repositories, I'll receive an array response

```
GET https://api.github.com/users/estranged42/repos
```

REST

GitHub API

- Typically all the records in a list will have the same fields, although JSON does not enforce this.


```
[
    {
        "id": 126917848,
        "node_id": "MDEwOlJlcG9zaXRvcnkxMjY5MTc4NDg=",
        "name": "Adafruit-GFX-Library",
        "full_name": "estranged42/Adafruit-GFX-Library",
        "private": false,
        "html_url": "https://github.com/estranged42/Adafruit-GFX-Lib
        "description": "Adafruit GFX graphics core library, this is
        "fork": true,
        "url": "https://api.github.com/repos/estranged42/Adafruit-GF
        "forks_url": "https://api.github.com/repos/estranged42/Adafr
    },
    {
        "id": 121828156,
        "node_id": "MDEwOlJlcG9zaXRvcnkxMjE4Mjg3NTY=",
        "name": "Adafruit_TinyMPR121",
        "full_name": "estranged42/Adafruit_TinyMPR121",
        "private": false,
        "html_url": "https://github.com/estranged42/Adafruit_TinyMPR
        "description": null,
        "fork": false,
        "url": "https://api.github.com/repos/estranged42/Adafruit_Ti
        "forks_url": "https://api.github.com/repos/estranged42/Adafr
    },
    {
        "id": 124449121,
        "node_id": "MDEwOlJlcG9zaXRvcnkxMjQ0NDkxMjE=",
        "name": "arduino-status-screen",
        "full_name": "estranged42/arduino-status-screen",
        "private": false,
        "html_url": "https://github.com/estranged42/arduino-status-s
        "description": "LCD Status Screen with an Adafruit HUZZAH32
        "fork": false,
        "url": "https://api.github.com/repos/estranged42/arduino-sta
        "forks_url": "https://api.github.com/repos/estranged42/ardui
    },
```

REST

GitHub API

- Typically records will have some sort of unique identifier


```
[
    {
        "id": 12691784,
        "node_id": "MDEwOlJlcG9zaXRvcnkxMjY5MTc4NDg=",
        "name": "Adafruit-GFX-Library",
        "full_name": "estranged42/Adafruit-GFX-Library",
        "private": false,
        "html_url": "https://github.com/estranged42/Adafruit-GFX-Lib
        "description": "Adafruit GFX graphics core library, this is
        "fork": true,
        "url": "https://api.github.com/repos/estranged42/Adafruit-GF
        "forks_url": "https://api.github.com/repos/estranged42/Adafr
    },
    {
        "id": 12182875,
        "node_id": "MDEwOlJlcG9zaXRvcnkxMjE4Mjg3NTY=",
        "name": "Adafruit_TinyMPR121",
        "full_name": "estranged42/Adafruit_TinyMPR121",
        "private": false,
        "html_url": "https://github.com/estranged42/Adafruit_TinyMPR
        "description": null,
        "fork": false,
        "url": "https://api.github.com/repos/estranged42/Adafruit_Tr
        "forks_url": "https://api.github.com/repos/estranged42/Adafr
    },
    {
        "id": 12444912,
        "node_id": "MDEwOlJlcG9zaXRvcnkxMjQ0NDkxMjE=",
        "name": "arduino-status-screen",
        "full_name": "estranged42/arduino-status-screen",
        "private": false,
        "html_url": "https://github.com/estranged42/arduino-status-s
        "description": "LCD Status Screen with an Adafruit HUZZAH32
        "fork": false,
        "url": "https://api.github.com/repos/estranged42/arduino-sta
        "forks_url": "https://api.github.com/repos/estranged42/ardui
```

REST

GitHub API

- There are specific URLs for each individual repository

```
GET https://api.github.com/repos/estranged42/ArduinoCore-samd
```


```
{
  "id": 151516009,
  "node_id": "MDEwOlJlcG9zaXRvcnkxNTE1MTYwMDk=",
  "name": "ArduinoCore-samd",
  "full_name": "estranged42/ArduinoCore-samd",
  "private": false,
  "html_url": "https://github.com/estranged42/ArduinoCore-samd",
  "description": "Arduino Core for SAMD21 CPU",
  "fork": true,
  "url": "https://api.github.com/repos/estranged42/ArduinoCore-samd
  "forks_url": "https://api.github.com/repos/estranged42/ArduinoCor
  "parent": {
    "id": 37462949,
    "node_id": "MDEwOlJlcG9zaXRvcnkzNzQ2Mjk0OQ==",
    "name": "ArduinoCore-samd",
    "full_name": "arduino/ArduinoCore-samd",
    "private": false,
    "owner": {
      "login": "arduino",
      "id": 379109,
      "node_id": "MDEyOk9yZ2FuaXphdGlvbjM3OTEwOQ=="
    },
    "html_url": "https://github.com/arduino/ArduinoCore-samd",
    "description": "Arduino Core for SAMD21 CPU",
    "fork": false,
                                                              d",
                                                              re-
    "open_issues": 181,
    "watchers": 406
  },
}
```