

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 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>11111111111111</cas:dbkey>
      <cas:emplid>22222222</cas:emplid>
      <cas:activestudent>0</cas:activestudent>
      <cas:activeemployee>1</cas:activeemployee>
    </cas:attributes>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

XML

```
{
  "serviceResponse": {
    "authenticationSuccess": {
      "user": "fischerm",
      "attributes": {
        "dbkey": 11111111111111,
        "emplid": 22222222,
        "activestudent": 0,
        "activeemployee": 1
      }
    }
  }
}
```

JSON

JSON vs XML

fight!

```
<?xml version="1.0" encoding="UTF-8"?>
<cas:serviceResponse xmlns:cas="http://www.yesilcay.com/cas">
  <cas:authenticationSuccess>
    <cas:user>fischerm</cas:user>
    <cas:attributes>
      <cas:dbkey>1111111111</cas:dbkey>
      <cas:emplid>2222222222</cas:emplid>
      <cas:activestudent>0</cas:activestudent>
      <cas:activeemployee>1</cas:activeemployee>
    </cas:attributes>
  </cas:authenticationSuccess>
</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",  
}
```

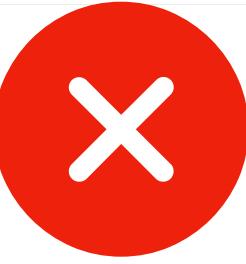
JSON Objects

- 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

A screenshot of a browser's developer tools Network tab. The "Preview" tab is selected, showing a JSON response. The JSON data is:

```
1 {  
2   "status": "OK",  
3   "books": [  
4     {  
5       "title": "There and Back Again",  
6       "author": "Bilbo Baggins"  
7     },  
8     {  
9       "title": "The Downfall of the Lord of the  
10      "author": "Frodo Baggins"  
11     },  
12     {  
13       "title": "Herblore of the Shire",  
14       "author": "Meriadoc Brandybuck"  
15     }  
16   ],  
17   "count": 3  
18 }
```

A screenshot of a browser's developer tools Network tab. The "Preview" tab is selected, showing a JSON response as an expandable tree. The JSON data is:

```
▼ {status: "OK", books: [{title: "There and Back Again", au...  
  ▼ books: [{title: "There and Back Again", author: "Bilbo Bag...  
    ► 0: {title: "There and Back Again", author: "Bilbo Bag...  
    ► 1: {title: "The Downfall of the Lord of the Rings, an...  
    ► 2: {title: "Herblore of the Shire", author: "Meriadoc...  
  count: 3  
  status: "OK"
```

JSON

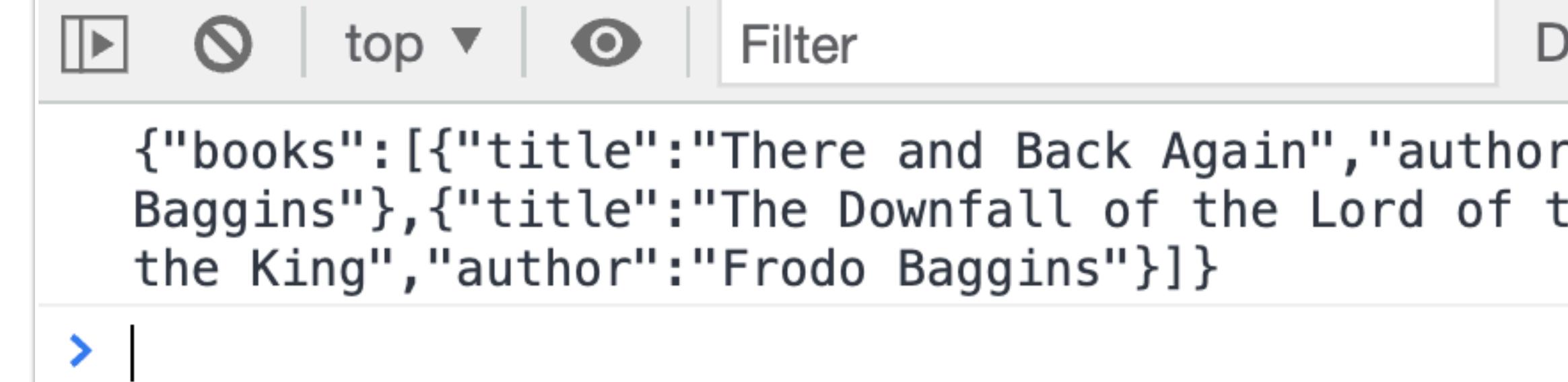
In JavaScript

- 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 Again",
      'author': "Bilbo Baggins"
    },
    {
      'title': "The Downfall of the Lord of the King",
      'author': "Frodo Baggins"
    }
  ]
}

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



A screenshot of a browser's developer tools console. At the top, there are standard navigation buttons: a play button, a stop button, a 'top' dropdown, a refresh button, and a 'Filter' input field. Below the toolbar, the console displays the JSON string produced by the code above. The string is: `{"books": [{"title": "There and Back Again", "author": "Bilbo Baggins"}, {"title": "The Downfall of the Lord of the King", "author": "Frodo Baggins"}]}`. The text is displayed in a monospaced font, with the JSON structure clearly visible.

JSON

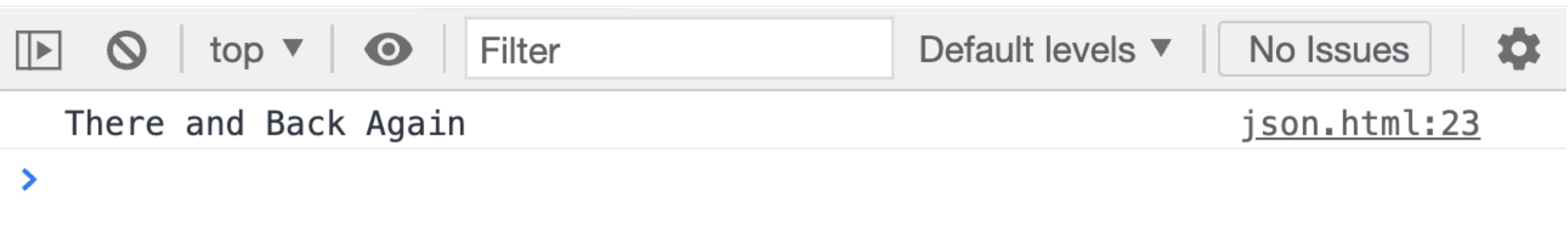
In JavaScript

- 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

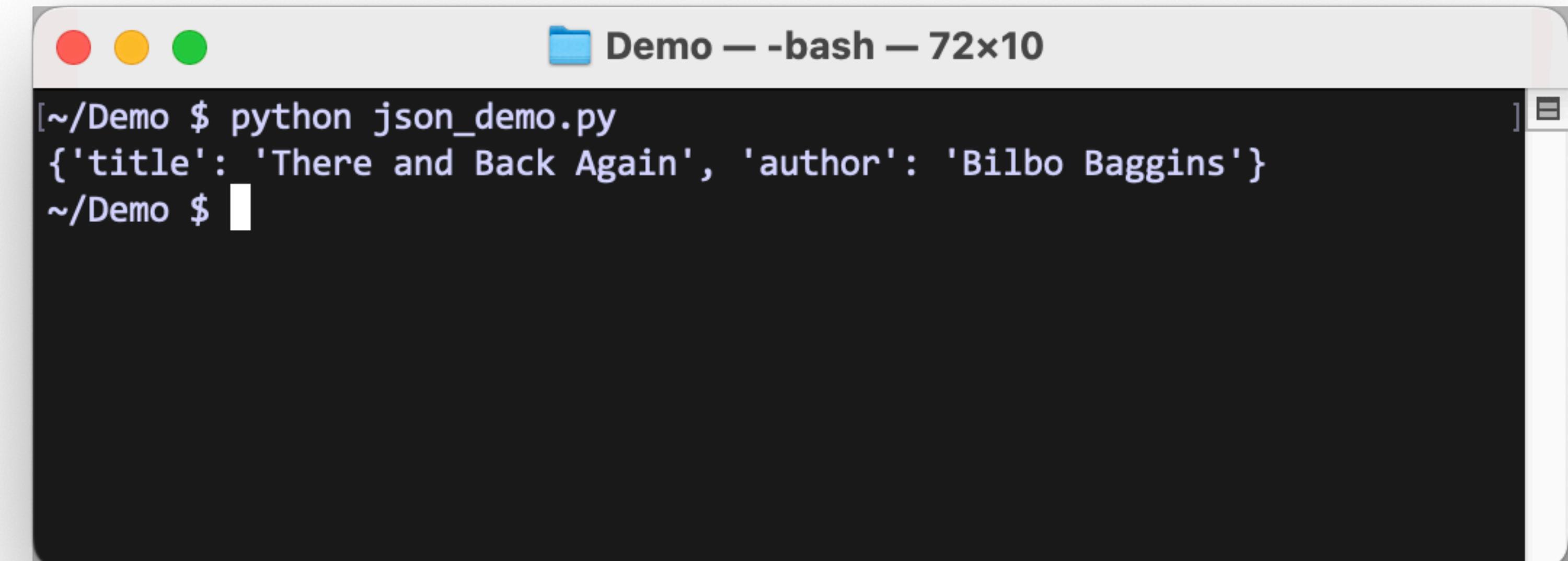
- json module is part of the Python standard library
- JSON → Python

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

```
import json

jsonString = '{"title": "There and Back Again", "author": "J.R.R. Tolkien"}'

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



The screenshot shows a terminal window with the title bar 'Demo — -bash — 72x10'. The command entered is '[~/Demo \$ python json_demo.py]' followed by the resulting JSON object: '{'title': 'There and Back Again', 'author': 'Bilbo Baggins'}'. The terminal window has a dark theme.

JSON In Python

- Python → JSON

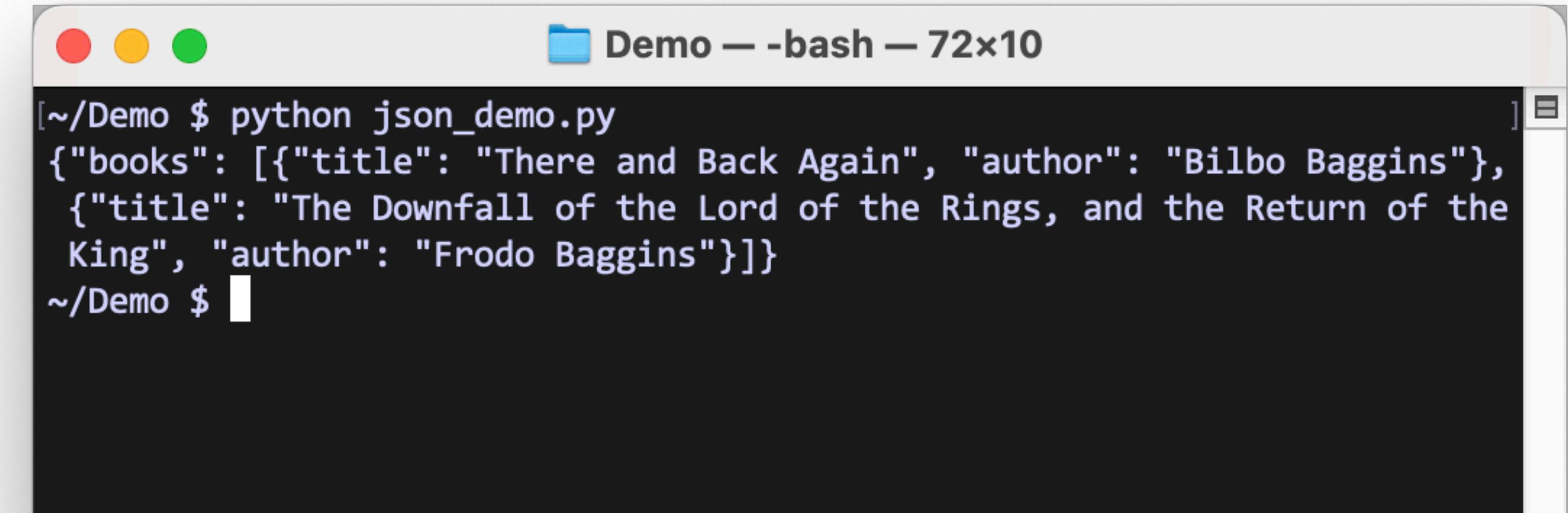
```
json.dumps( var )
```

```
import json

obj = {
    "books": [
        {"title": "There and Back Again", "author": "Bilbo Baggins"},  

        {"title": "The Downfall of the Lord of the Rings, and the Return of the King", "author": "Frodo Baggins"}]
}
```

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



A screenshot of a Mac OS X terminal window titled "Demo — -bash — 72x10". The window shows the command "python json_demo.py" being run, followed by its output: a JSON string representing a list of books. The books have titles like "There and Back Again" and "The Downfall of the Lord of the Rings, and the Return of the King", and authors like "Bilbo Baggins" and "Frodo Baggins".

```
[~/Demo $ python json_demo.py
{"books": [{"title": "There and Back Again", "author": "Bilbo Baggins"}, {"title": "The Downfall of the Lord of the Rings, and the Return of the King", "author": "Frodo Baggins"}]}
~/Demo $ ]
```

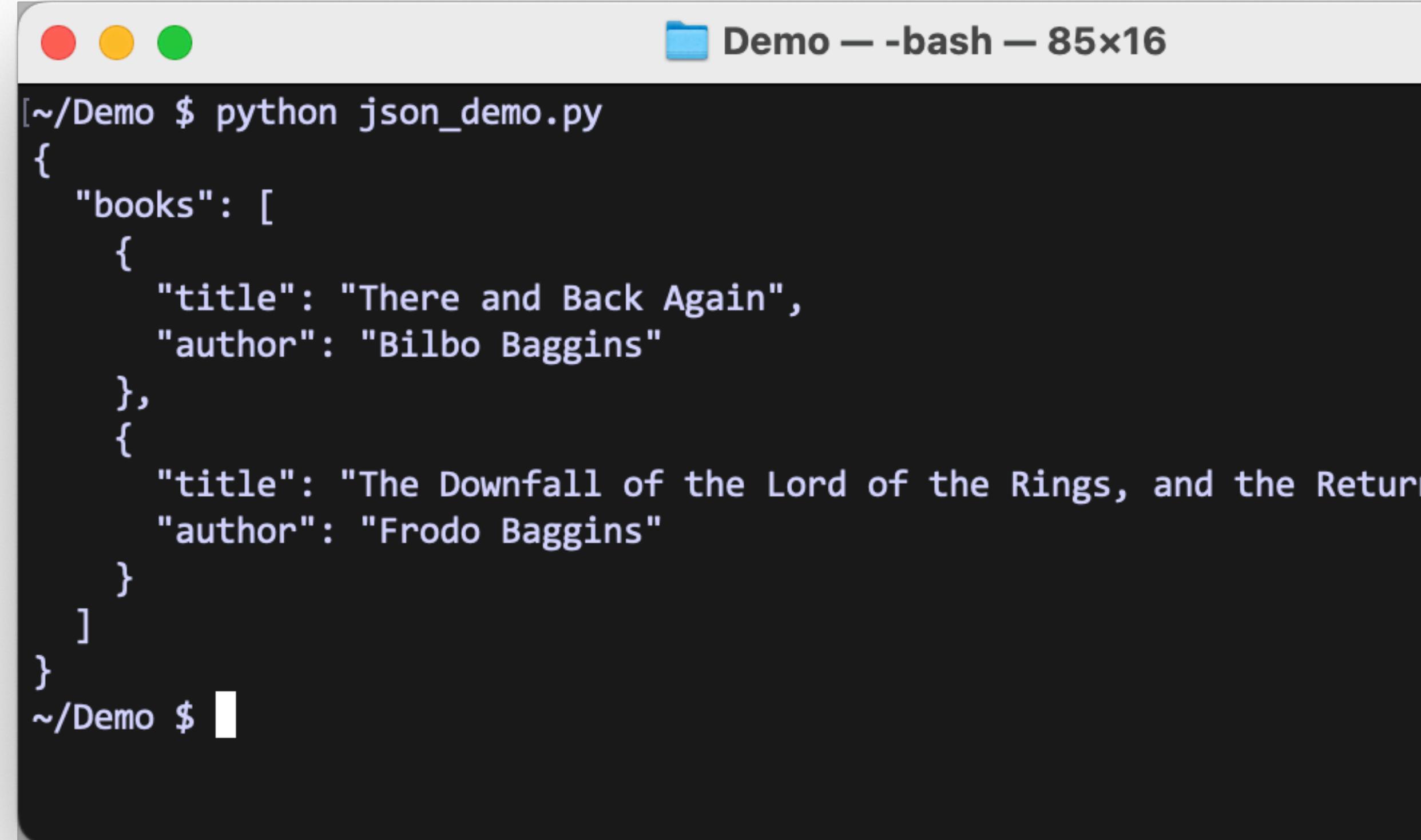
JSON In Python

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

```
import json

. . .

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



A screenshot of a terminal window titled "Demo — -bash — 85x16". The window shows the command `python json_demo.py` being run, followed by a JSON object representing a list of books. The JSON is formatted with two levels of indentation, demonstrating the use of the `indent` parameter in the `dumps` function.

```
[~/Demo $ python json_demo.py
{
  "books": [
    {
      "title": "There and Back Again",
      "author": "Bilbo Baggins"
    },
    {
      "title": "The Downfall of the Lord of the Rings, and the Return of the King",
      "author": "Frodo Baggins"
    }
  ]
}
~/Demo $
```

REST

Representational State Transfer

REST

Representational State Transfer

- 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

GET https://api.github.com/ + ⚡

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

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

Params Authorization Headers (5) Body Pre-request Script Tests Settings

Body Cookies Headers (25) Test Results Status: 200

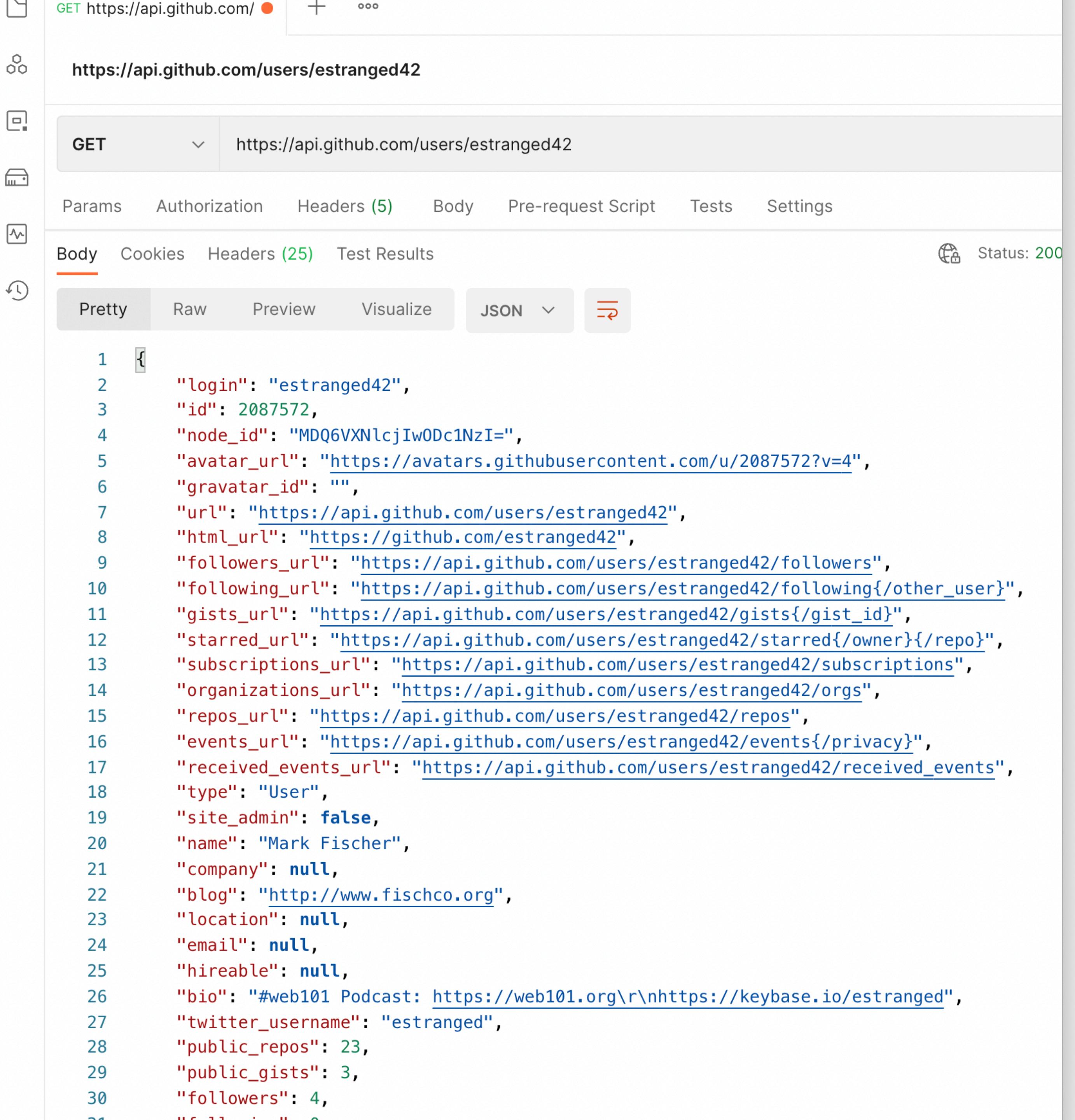
Pretty Raw Preview Visualize JSON ↻

```
1 {  
2   "login": "estranged42",  
3   "id": 2087572,  
4   "node_id": "MDQ6VXNlcjIwODc1NzI=",  
5   "avatar_url": "https://avatars.githubusercontent.com/u/2087572?v=4",  
6   "gravatar_id": "",  
7   "url": "https://api.github.com/users/estranged42",  
8   "html_url": "https://github.com/estranged42",  
9   "followers_url": "https://api.github.com/users/estranged42/followers",  
10  "following_url": "https://api.github.com/users/estranged42/following{/other_user}",  
11  "gists_url": "https://api.github.com/users/estranged42/gists{/gist_id}",  
12  "starred_url": "https://api.github.com/users/estranged42/starred{/owner}{/repo}",  
13  "subscriptions_url": "https://api.github.com/users/estranged42/subscriptions",  
14  "organizations_url": "https://api.github.com/users/estranged42/orgs",  
15  "repos_url": "https://api.github.com/users/estranged42/repos",  
16  "events_url": "https://api.github.com/users/estranged42/events{/privacy}",  
17  "received_events_url": "https://api.github.com/users/estranged42/received_events",  
18  "type": "User",  
19  "site_admin": false,  
20  "name": "Mark Fischer",  
21  "company": null,  
22  "blog": "http://www.fischco.org",  
23  "location": null,  
24  "email": null,  
25  "hireable": null,  
26  "bio": "#web101 Podcast: https://web101.org\r\nhttps://keybase.io/estranged",  
27  "twitter_username": "estranged",  
28  "public_repos": 23,  
29  "public_gists": 3,  
30  "followers": 4,  
31  "following": 0
```

REST

GitHub API

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



The screenshot shows a Postman interface with a GET request to `https://api.github.com/users/estranged42`. The response status is 200. The response body is a JSON object representing a GitHub user, with lines numbered 1 through 31. The JSON object includes fields like login, id, node_id, avatar_url, gravatar_id, url, html_url, followers_url, following_url, gists_url, starred_url, subscriptions_url, organizations_url, repos_url, events_url, received_events_url, type, site_admin, name, company, blog, location, email, hireable, bio, twitter_username, public_repos, public_gists, and followers.

```
1  {
2      "login": "estranged42",
3      "id": 2087572,
4      "node_id": "MDQ6VXNlcjIwODc1NzI=",
5      "avatar_url": "https://avatars.githubusercontent.com/u/2087572?v=4",
6      "gravatar_id": "",
7      "url": "https://api.github.com/users/estranged42",
8      "html_url": "https://github.com/estranged42",
9      "followers_url": "https://api.github.com/users/estranged42/followers",
10     "following_url": "https://api.github.com/users/estranged42/following{/other_user}",
11     "gists_url": "https://api.github.com/users/estranged42/gists{/gist_id}",
12     "starred_url": "https://api.github.com/users/estranged42/starred{/owner}{/repo}",
13     "subscriptions_url": "https://api.github.com/users/estranged42/subscriptions",
14     "organizations_url": "https://api.github.com/users/estranged42/orgs",
15     "repos_url": "https://api.github.com/users/estranged42/repos",
16     "events_url": "https://api.github.com/users/estranged42/events{/privacy}",
17     "received_events_url": "https://api.github.com/users/estranged42/received_events",
18     "type": "User",
19     "site_admin": false,
20     "name": "Mark Fischer",
21     "company": null,
22     "blog": "http://www.fischco.org",
23     "location": null,
24     "email": null,
25     "hireable": null,
26     "bio": "#web101 Podcast: https://web101.org\r\nhttps://keybase.io/estranged",
27     "twitter_username": "estranged",
28     "public_repos": 23,
29     "public_gists": 3,
30     "followers": 4,
31     "following": 0
```

REST GitHub API

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

```
[  
  {  
    "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-Library",  
    "description": "Adafruit GFX graphics core library, this is  
    "fork": true,  
    "url": "https://api.github.com/repos/estranged42/Adafruit-GFX-Library",  
    "forks_url": "https://api.github.com/repos/estranged42/Adafruit-GFX-Library"  
  },  
  {  
    "id": 121828756,  
    "node_id": "MDEwOlJlcG9zaXRvcnkxMjE4Mjg3NTY=",  
    "name": "Adafruit_TinyMPR121",  
    "full_name": "estranged42/Adafruit_TinyMPR121",  
    "private": false,  
    "html_url": "https://github.com/estranged42/Adafruit_TinyMPR121",  
    "description": null,  
    "fork": false,  
    "url": "https://api.github.com/repos/estranged42/Adafruit_TinyMPR121",  
    "forks_url": "https://api.github.com/repos/estranged42/Adafruit_TinyMPR121"  
  },  
  {  
    "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-screen",  
    "description": "A simple status screen for the Arduino Uno using the Adafruit GFX library.",  
    "fork": false,  
    "url": "https://api.github.com/repos/estranged42/arduino-status-screen",  
    "forks_url": "https://api.github.com/repos/estranged42/arduino-status-screen"  
  }]
```

```
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": "MDExOjJlcG9zaXRvcnkxMjY5MTc4NDg=",  
    "name": "Adafruit-GFX-Library",  
    "full_name": "estranged42/Adafruit-GFX-Library",  
    "private": false,  
    "html_url": "https://github.com/estranged42/Adafruit-GFX-Library",  
    "description": "Adafruit GFX graphics core library, this is  
    "fork": true,  
    "url": "https://api.github.com/repos/estranged42/Adafruit-GFX-Library",  
    "forks_url": "https://api.github.com/repos/estranged42/Adafruit-GFX-Library/forks"  
  },  
  {  
    "id": 121828156,  
    "node_id": "MDExOjJlcG9zaXRvcnkxMjE4Mjg3NTY=",  
    "name": "Adafruit_TinyMPR121",  
    "full_name": "estranged42/Adafruit_TinyMPR121",  
    "private": false,  
    "html_url": "https://github.com/estranged42/Adafruit_TinyMPR121",  
    "description": null,  
    "fork": false,  
    "url": "https://api.github.com/repos/estranged42/Adafruit_TinyMPR121",  
    "forks_url": "https://api.github.com/repos/estranged42/Adafruit_TinyMPR121/forks"  
  },  
  {  
    "id": 124449121,  
    "node_id": "MDExOjJlcG9zaXRvcnkxMjQ0NDkxMjE=",  
    "name": "arduino-status-screen",  
    "full_name": "estranged42/arduino-status-screen",  
    "private": false,  
    "html_url": "https://github.com/estranged42/arduino-status-screen",  
    "description": "LCD Status Screen with an Adafruit HUZZAH32",  
    "fork": false,  
    "url": "https://api.github.com/repos/estranged42/arduino-status-screen",  
    "forks_url": "https://api.github.com/repos/estranged42/arduino-status-screen/forks"  
  }]
```

REST GitHub API

- Typically records will have some sort of unique identifier

```
[  
  {  
    "id": 126917843,  
    "node_id": "MDEwOlJlcG9zaXRvcnkxMjY5MTc4NDg=",  
    "name": "Adafruit-GFX-Library",  
    "full_name": "estranged42/Adafruit-GFX-Library",  
    "private": false,  
    "html_url": "https://github.com/estranged42/Adafruit-GFX-Library",  
    "description": "Adafruit GFX graphics core library, this is  
    "fork": true,  
    "url": "https://api.github.com/repos/estranged42/Adafruit-GFX-Library",  
    "forks_url": "https://api.github.com/repos/estranged42/Adafruit-GFX-Library/forks"  
  },  
  {  
    "id": 121828753,  
    "node_id": "MDEwOlJlcG9zaXRvcnkxMjE4Mjg3NTY=",  
    "name": "Adafruit_TinyMPR121",  
    "full_name": "estranged42/Adafruit_TinyMPR121",  
    "private": false,  
    "html_url": "https://github.com/estranged42/Adafruit_TinyMPR121",  
    "description": null,  
    "fork": false,  
    "url": "https://api.github.com/repos/estranged42/Adafruit_TinyMPR121",  
    "forks_url": "https://api.github.com/repos/estranged42/Adafruit_TinyMPR121/forks"  
  },  
  {  
    "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-screen",  
    "description": "LCD Status Screen with an Adafruit HUZZAH32",  
    "fork": false,  
    "url": "https://api.github.com/repos/estranged42/arduino-status-screen",  
    "forks_url": "https://api.github.com/repos/estranged42/arduino-status-screen/forks"  
  }]
```

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/ArduinoCore-samd",  
  "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,  
    "open_issues": 181,  
    "watchers": 406  
  },  
}
```