API

What is an API

- Application Programming Interface
- It is not visible
- It's how one computer talks to another computer
- It doesn't matter what programming language you're using
    - Javascript
    - Python
    - PHP
    - Java
    - C
    - And every other modern language supports RESTFUL APIs

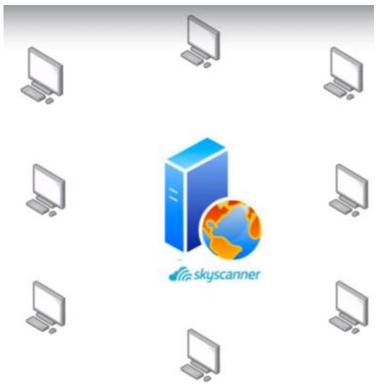
Metaphor

- Think of a restaurant
- You = the computer
- Waiter = fetch, post etc
- Chef = API

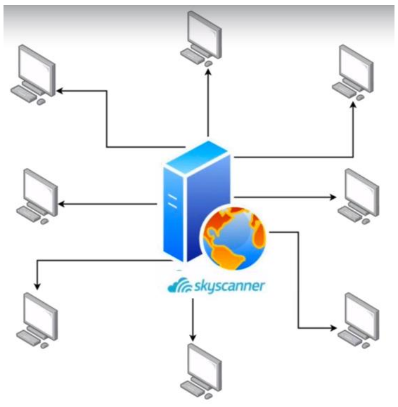
APIs in real life

- RESTful APIs are meant to be simple

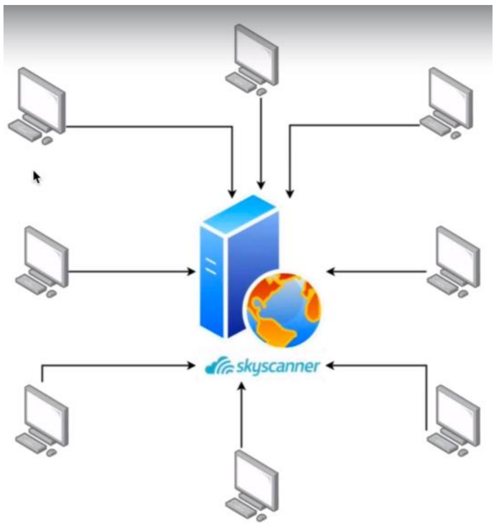
- This is a site that uses an API to collect flight prices from other websites



- These are airline services
- They hold all the data



- Skyscanner will ask each one for flight data
- Fetch



- Now you can see all the flight prices from other websites
- Data presented to you

What programming languages can we use

- Computers use APIs to talk to each other over the internet
- JS, PHP, Python, Ruby, C++, Java, C, C#
- We can use any modern language that you'd use for a website

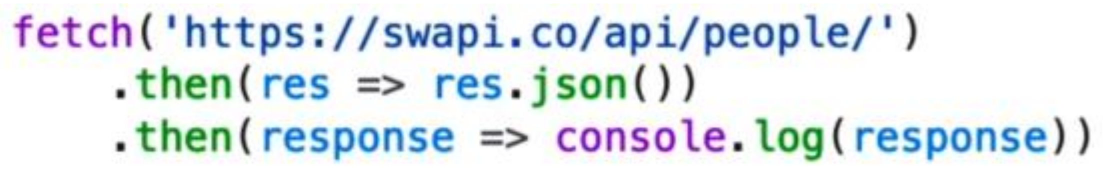
RESTful APIs

- Is a type of API
- Representational State Transfer
- Client computer (request) asks another computer for data, or to take an action (modify, delete, etc)

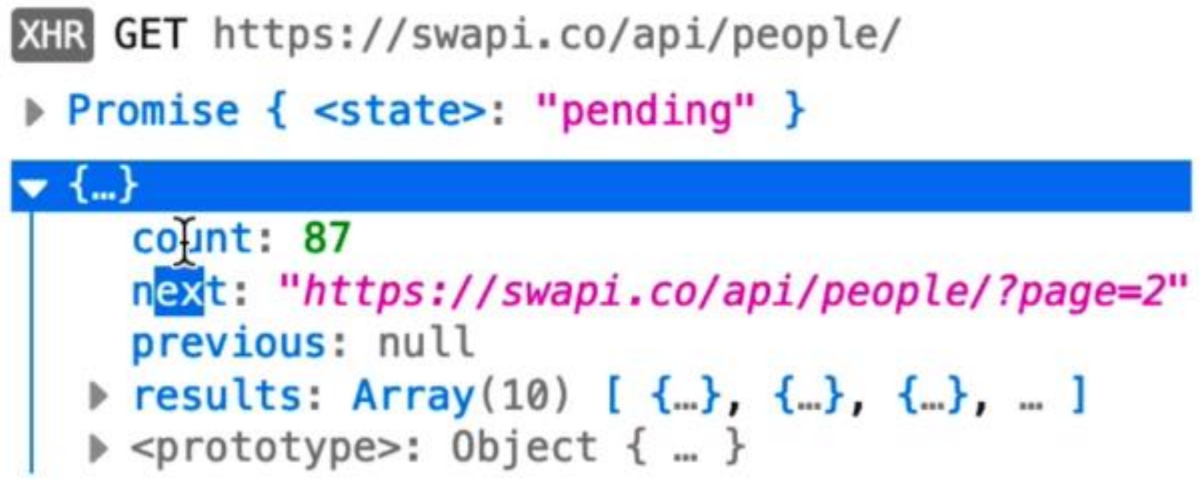
JSON

- Javascript Object Notation
- Swapi.co (Star wars api)
- APIs return data in JSON format
- Structured key value pair

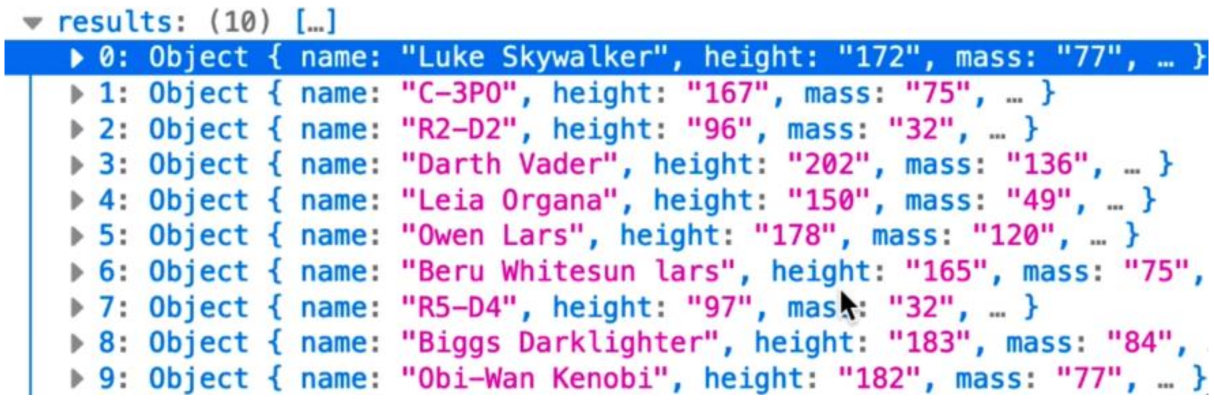
JSON in real life



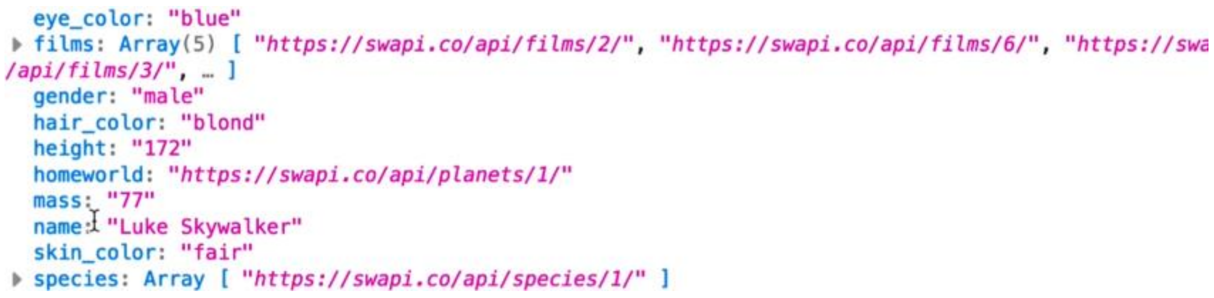
```
fetch('https://swapi.co/api/people/')
     .then(res => res.json())
     .then(response => console.log(response))
```



```
XHR GET https://swapi.co/api/people/
▶ Promise { <state>: "pending" }
▼ {…}
    count: 87
    next: "https://swapi.co/api/people/?page=2"
    previous: null
  ▶ results: Array(10) [ {…}, {…}, {…}, … ]
  ▶ <prototype>: Object { … }
```



```
▼ results: (10) […]
  ▶ 0: Object { name: "Luke Skywalker", height: "172", mass: "77", … }
  ▶ 1: Object { name: "C-3PO", height: "167", mass: "75", … }
  ▶ 2: Object { name: "R2-D2", height: "96", mass: "32", … }
  ▶ 3: Object { name: "Darth Vader", height: "202", mass: "136", … }
  ▶ 4: Object { name: "Leia Organa", height: "150", mass: "49", … }
  ▶ 5: Object { name: "Owen Lars", height: "178", mass: "120", … }
  ▶ 6: Object { name: "Beru Whitesun lars", height: "165", mass: "75",
  ▶ 7: Object { name: "R5-D4", height: "97", mass: "32", … }
  ▶ 8: Object { name: "Biggs Darklighter", height: "183", mass: "84",
  ▶ 9: Object { name: "Obi-Wan Kenobi", height: "182", mass: "77", … }
```

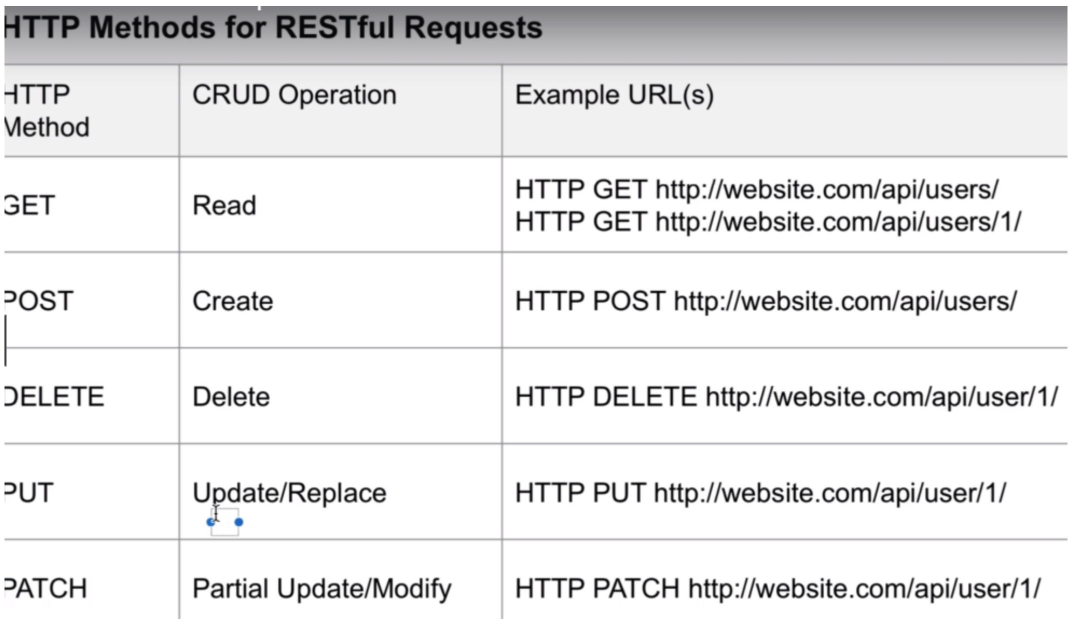


```
  eye_color: "blue"
▶ films: Array(5) [ "https://swapi.co/api/films/2/", "https://swapi.co/api/films/6/", "https://swa
/api/films/3/", … ]
  gender: "male"
  hair_color: "blond"
  height: "172"
  homeworld: "https://swapi.co/api/planets/1/"
  mass: "77"
  name: "Luke Skywalker"
  skin_color: "fair"
▶ species: Array [ "https://swapi.co/api/species/1/" ]
```

Request Methods

- CRUD Operations
    - Create
    - Read
    - Update
    - Delete

- HTTP GET
  - When you load a website
  - It's a request to get data from another computer
  - You're simply asking for data and you're not asking to perform a task
  - You're not creating, updating or deleting data
  - Most common request type
- HTTP POST
  - Does not go through the URL, but uses a URL as the endpoint
  - Asks another computer to create a new resource
  - Returns data about the newly created resource
  - Make a brand new resource
  - Create a new user on facebook
- HTTP DELETE
  - Does not go through the standard URL, but uses a URL as the endpoint
  - Asks another computer to delete a single resource or a list of resources
  - Use with caution
- HTTP PATCH
  - Does not go through the standard URL, but uses a URL as the endpoint
  - Asks another computer to update a piece of a resource
  - Are not fully supported by all browsers or frameworks, so we typically fall back on PUT requests
- HTTP PUT
  - Does not go through the standard URL, but uses a URL as the endpoint
  - Asks another computer to update an entire resource
  - If the resource does not exist, the API might decide to CREATE (CRUD) the resource



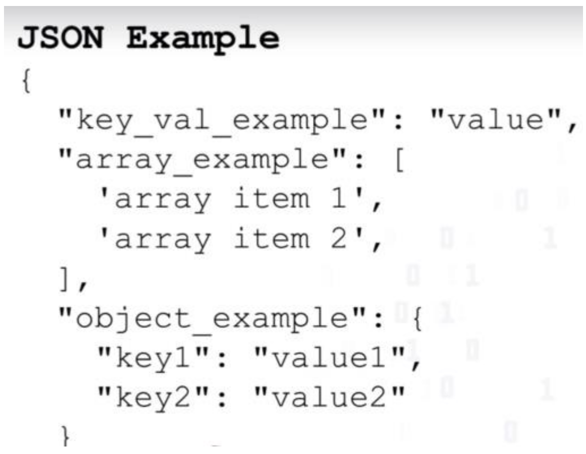
HTTP Methods for RESTful Requests

HTTP Method	CRUD Operation	Example URL(s)
GET	Read	HTTP GET http://website.com/api/users/ HTTP GET http://website.com/api/users/1/
POST	Create	HTTP POST http://website.com/api/users/
DELETE	Delete	HTTP DELETE http://website.com/api/user/1/
PUT	Update/Replace	HTTP PUT http://website.com/api/user/1/
PATCH	Partial Update/Modify	HTTP PATCH http://website.com/api/user/1/

Path has to have /firstname as an example appended at the end

Consuming APIs

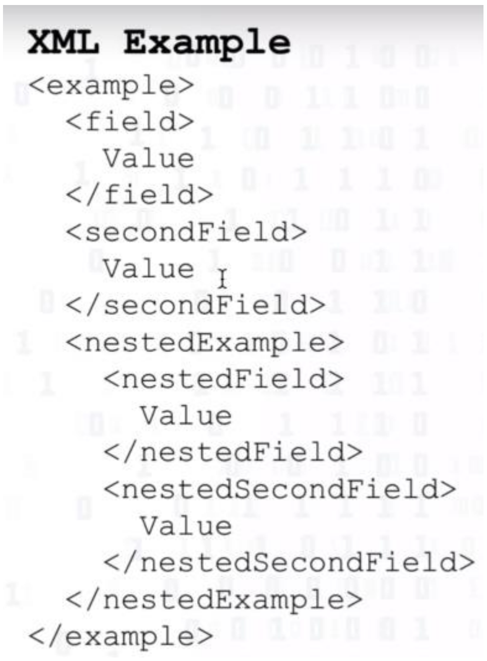
- APIs can be written in almost any server-side language
- APIs will generally return one of two types of data structures
    - JSON



JSON Example

```
{
  "key_val_example": "value",
  "array_example": [
    'array item 1',
    'array item 2',
  ],
  "object_example": {
    "key1": "value1",
    "key2": "value2"
  }
```

    - XML



XML Example

```
<example>
  <field>
    Value
  </field>
  <secondField>
    Value
  </secondField>
  <nestedExample>
    <nestedField>
      Value
    </nestedField>
    <nestedSecondField>
      Value
    </nestedSecondField>
  </nestedExample>
</example>
```

- APIs can be consumed in almost any language
- Browsers use JS for their API requests
- Servers use any language that runs on that computer

Requests and responses

- Request
    - When you request data from a server using GET, POST, PUT, PATCH or DELETE
- Response
    - When the server returns your data
    - Will always come with an HTTP Status Code

        - These “status codes” tell you what’s wrong (or right) without needing to give you text back to read

Common HTTP Status codes

- Healthy responses (2--)
    - 200 – OK
        - Request accepted
    - 201 – Created
        - Post request often return 201s when a resource is created\
    - 202 – Accepted
        - When a request is accepted but its not done processing
        - Task maybe goes into a queue
- Redirect Responses (3--)
    - 301 – Moved Permanently
        - When the endpoint has permanently changed
        - Update your endpoint
    - 302 – Found
        - The endpoint you are accessing is temporarily moved to somewhere else
- Client Responses (4--)
    - 400 – Bad Request
        - Server cannot or will not process your request
        - Often this is due to malformed API keys or an invalid payload
    - 401 – Unauthorized
        - You’re not allowed here
        - Usually this is because you’re missing authentication credentials (API keys)
    - 403 – Forbidden
        - The server understands your request but won’t execute it
        - Your API keys might not have the right permissions or your trying to use an endpoint that you don’t have access to
    - 404 – Not found
        - There’s nothing here
        - Move along, move along
    - 405 – Method not allowed
        - You’re using the wrong HTTP method
        - The endpoint might only accept GET requests and you might be POSTing to it, for example
    - 418
        - The server refuses to brew coffee because it is, permanently, a teapot
- Server responses (5--)
    - 500 – Internal Server Error
        - The server had a problem and couldn’t process the request
        - This is the only time you are out of control

API Security

- API Keys
  - Passwords to access an API
  - These are your authentication credentials
  - Almost every website requires API keys to perform some action
  - Facebook’s Graph API is a good example
    - Access token is generated with an API key