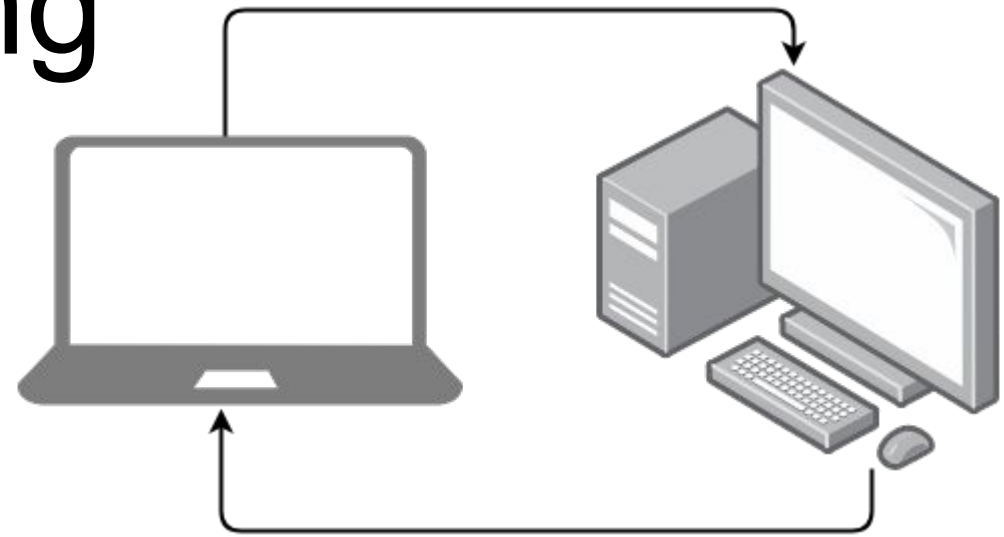
Understanding APIs

How APIs and RESTful APIs work behind the scenes



What is an API?

Application Programming Interface

But not a “visual” interface like what we see in this presentation

It’s how one computer can talk 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.

More on this in a bit...

There are many types of APIs.

But RESTful APIs are most commonly talked about these days.

RESTful APIs are what we're talking about today, too.

Think of an API as a waiter as a restaurant.

Art by iconicbestiary at https://www.freepik.com/iconicbestiary

Art by iconicbestiary at https://www.freepik.com/iconicbestiary

Art by iconicbestiary at https://www.freepik.com/iconicbestiary

The waiter relays the message to the kitchen.

Art by iconicbestiary at https://www.freepik.com/iconicbestiary

Where the chef makes the food.

Art by freepik at https://www.freepik.com/freepik

Then the waiter brings your food to you.

Art by iconicbestiary at https://www.freepik.com/iconicbestiary

It’s that simple!

Just always keep this metaphor in mind.
Remember the restaurant.

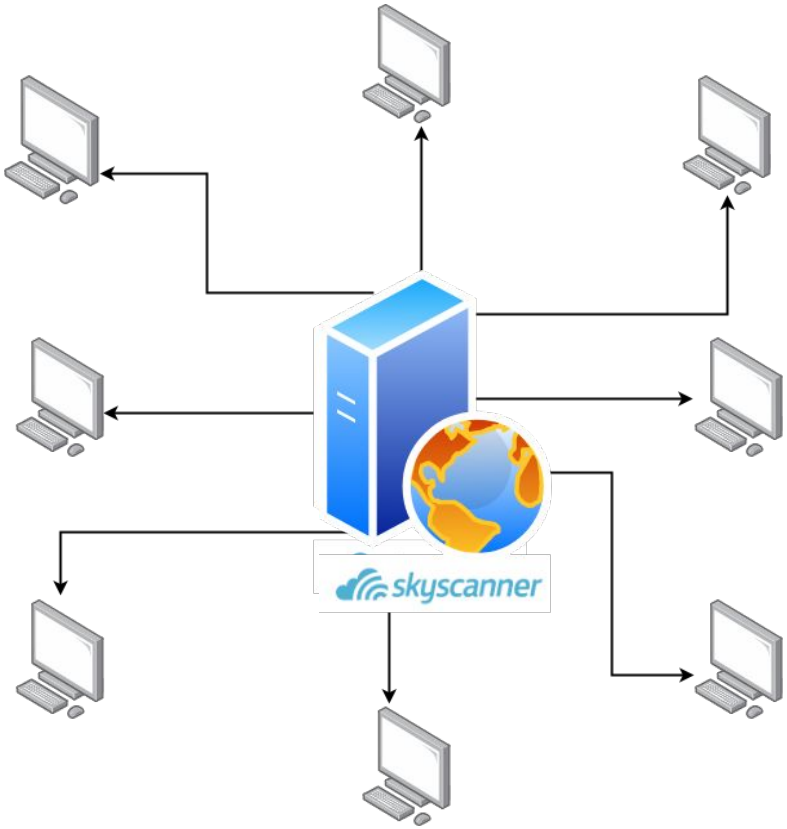
RESTful APIs are meant to be simple.

Let’s look at a real life example with actual computers.

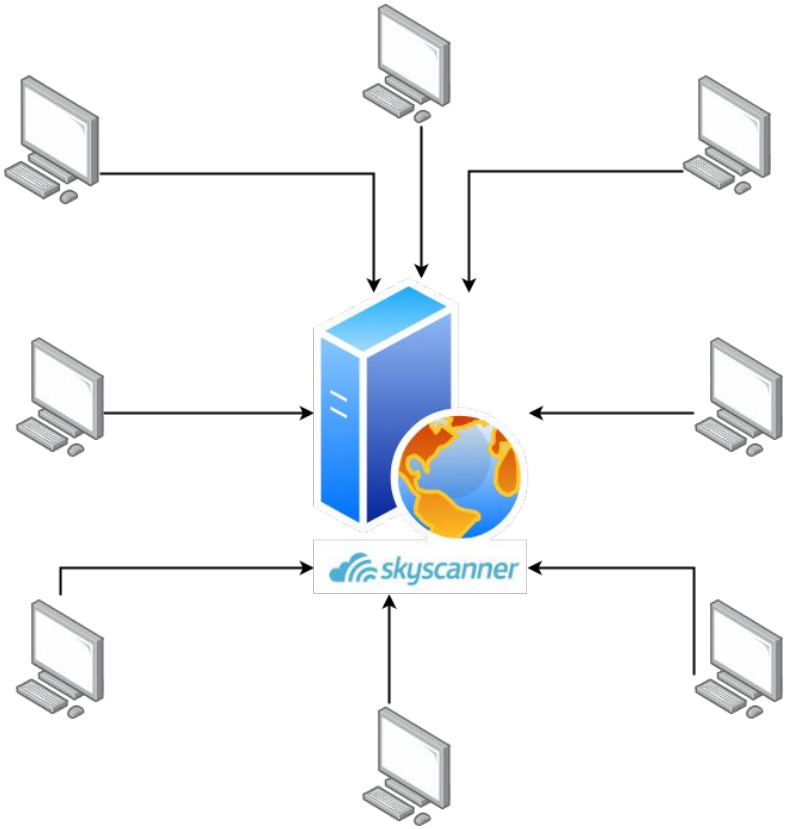
This is a site that uses an API to collect flight prices from other websites.

These are airline services. They hold all the data.

Skyscanner will ask each one for flight data.

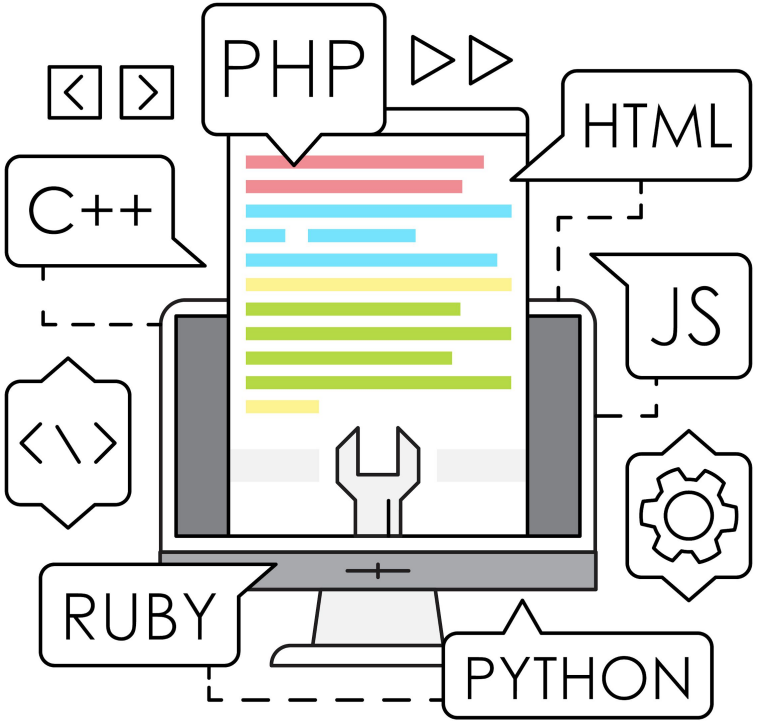


Now you can see all the flight prices from other websites



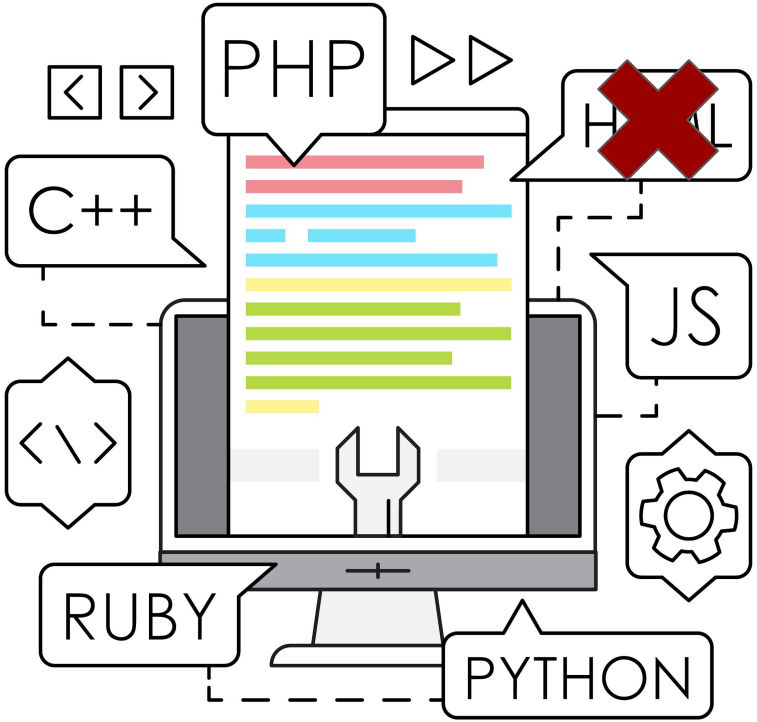
Computers use APIs to talk to each other over the internet.

What programming languages can we use?



Art by artmonkey at https://www.freepik.com/artmonkey

What programming languages can we use?



Art by artmonkey at https://www.freepik.com/artmonkey

We can use any modern language that you’d use for a website.

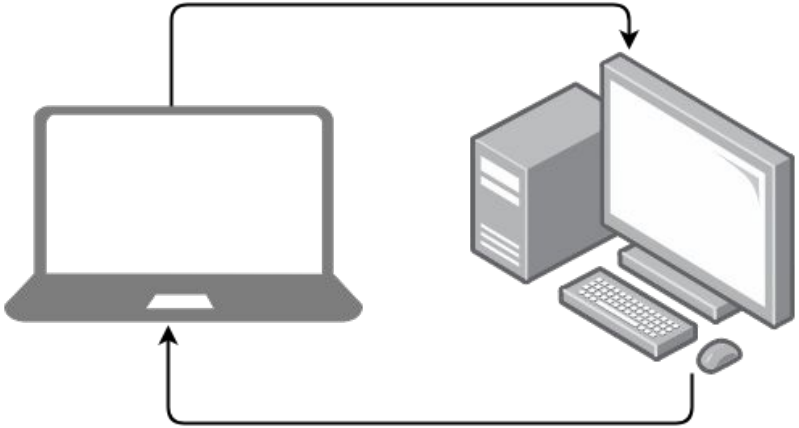
- Python
- JavaScript
- PHP
- Java
- C
- Ruby
- Etc

What are RESTful APIs?

REST is a type of API

Art by freepik at https://www.freepik.com/freepik

REpresentational
State
Transfer



Simply put:

- Client computer asks another computer for data, or to take an action

SWAPI

The Star Wars API

https://swapi.co/

JavaScript Demo

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

```
fetch('https://swapi.co/api/people/?limit=10&page=2')
    .then(res => res.json())
    .then(response => console.log(response))
```

```
▶ Promise {<pending>}
```

```
▼ {count: 87, next: "https://swapi.co/api/people/?limit=10&page=3", previous: "https://swapi.co/api/people/?limit=10&page=1", results: Array(10)}
    count: 87
    next: "https://swapi.co/api/people/?limit=10&page=3"
    previous: "https://swapi.co/api/people/?limit=10&page=1"
  ▼ results: Array(10)
    ▼ 0:
        birth_year: "41.9BBY"
        created: "2014-12-10T16:20:44.310000Z"
        edited: "2014-12-20T21:17:50.327000Z"
        eye_color: "blue"
      ▶ films: (3) ["https://swapi.co/api/films/5/", "https://swapi.co/api/films/4/", "https://swapi.co/api/films/6/"]
        gender: "male"
        hair_color: "blond"
        height: "188"
        homeworld: "https://swapi.co/api/planets/1/"
        mass: "84"
        name: "Anakin Skywalker"
        skin_color: "fair"
      ▶ species: ["https://swapi.co/api/species/1/"]
      ▶ starships: (3) ["https://swapi.co/api/starships/59/", "https://swapi.co/api/starships/65/", "https://swapi.co/api/starships/39/"]
        url: "https://swapi.co/api/people/11/"
      ▶ vehicles: (2) ["https://swapi.co/api/vehicles/44/", "https://swapi.co/api/vehicles/46/"]
      ▶ __proto__: Object
    ▶ 1: {name: "Wilhuff Tarkin", height: "180", mass: "unknown", hair_color: "auburn, grey", skin_color: "fair", …}
    ▶ 2: {name: "Chewbacca", height: "228", mass: "112", hair_color: "brown", skin_color: "unknown", …}
    ▶ 3: {name: "Han Solo", height: "180", mass: "80", hair_color: "brown", skin_color: "fair", …}
    ▶ 4: {name: "Greedo", height: "173", mass: "74", hair_color: "n/a", skin_color: "green", …}
    ▶ 5: {name: "Jabba Desilijic Tiure", height: "175", mass: "1,358", hair_color: "n/a", skin_color: "green-tan, brown", …}
    ▶ 6: {name: "Wedge Antilles", height: "170", mass: "77", hair_color: "brown", skin_color: "fair", …}
    ▶ 7: {name: "Jek Tono Porkins", height: "180", mass: "110", hair_color: "brown", skin_color: "fair", …}
    ▶ 8: {name: "Yoda", height: "66", mass: "17", hair_color: "white", skin_color: "green", …}
    ▶ 9: {name: "Palpatine", height: "170", mass: "75", hair_color: "grey", skin_color: "pale", …}
```

Hello, JSON

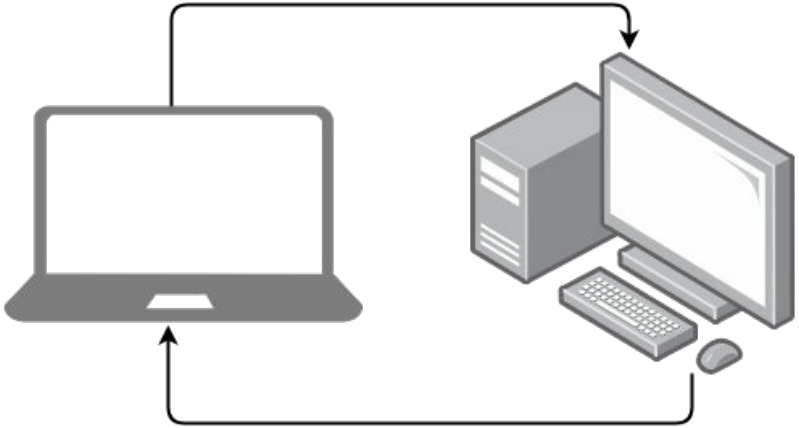
Hello, JSON

JavaScript Object Notation

Most languages have a data structure that looks like a JavaScript Object.

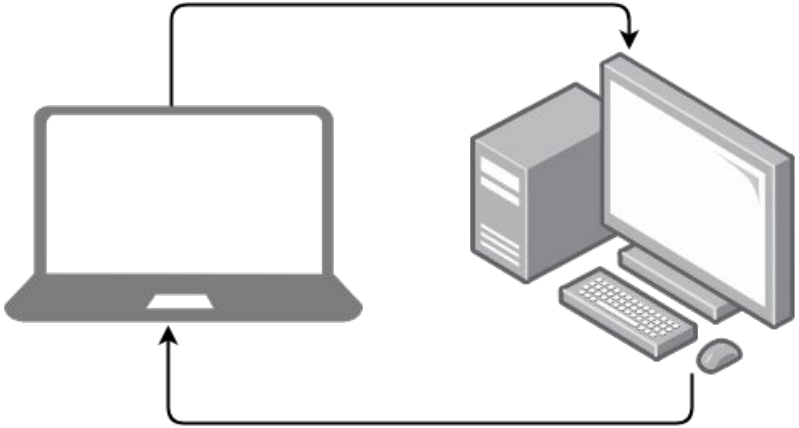
One day, someone decided it should be a standard.

Then it became the standard.



Request Methods:

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE
- HTTP PATCH

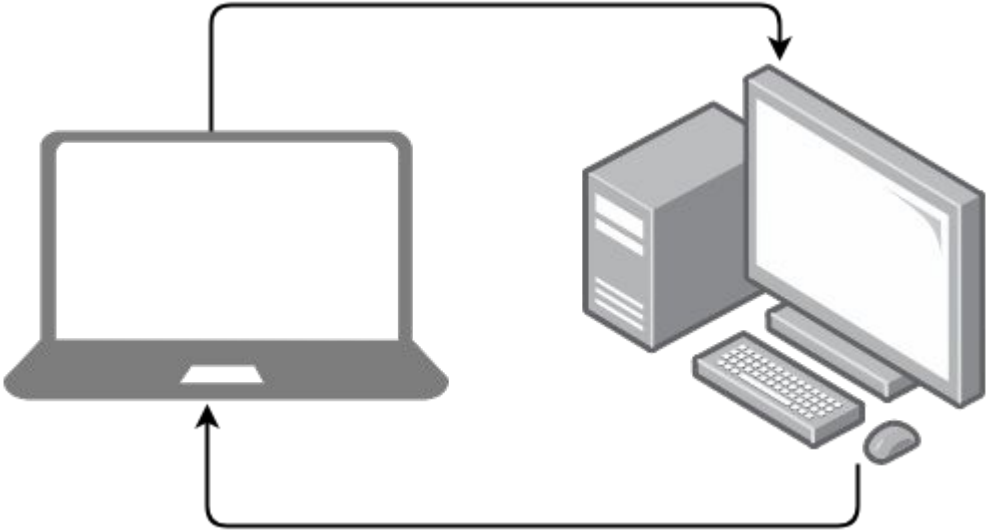


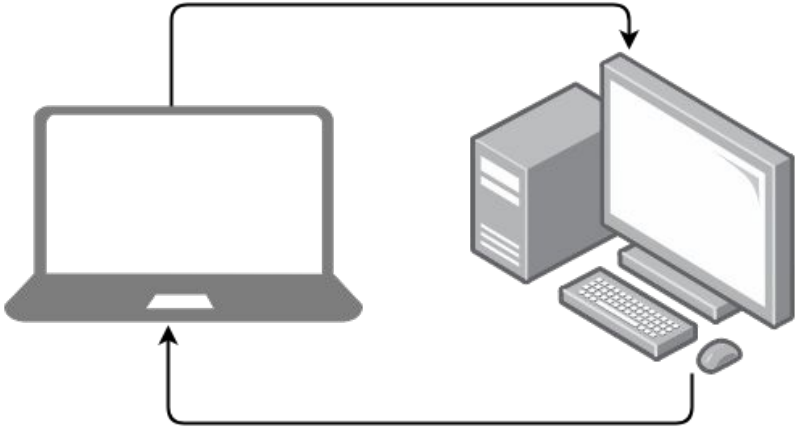
The first method:

- **HTTP GET**
- HTTP POST
- HTTP PUT
- HTTP DELETE
- HTTP PATCH

GET Requests

How do the work?





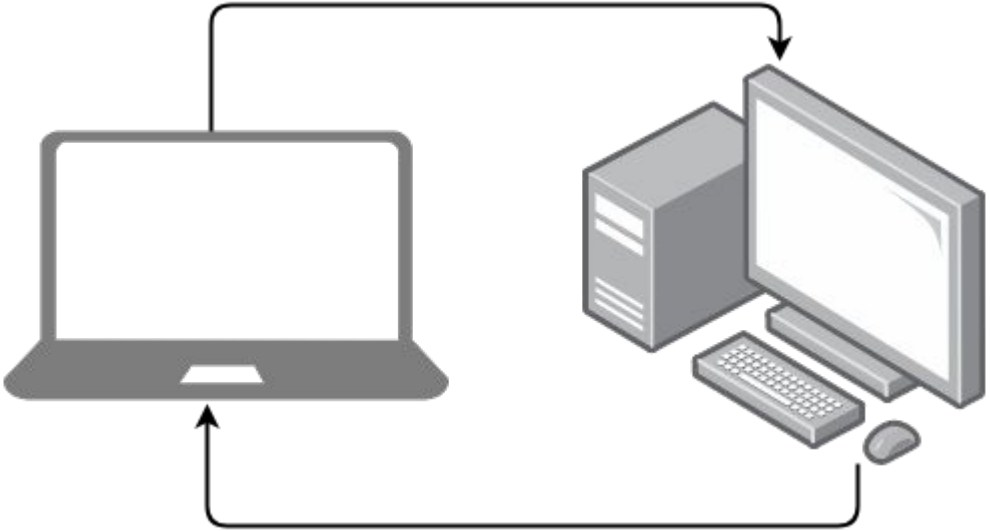
GET Requests

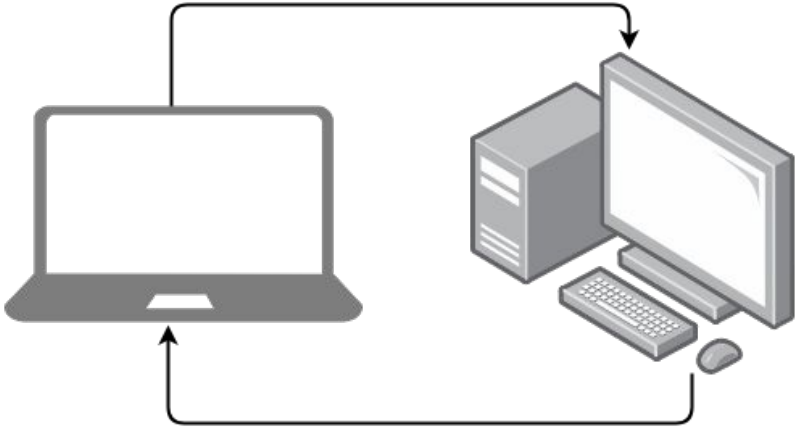
- When you load a website. That’s a GET request
- It’s a request to get data from another computer
- You’re simply asking for data and you’re not asking to perform a task
- You’re not creating, updating or deleting data
- Most common request type

HTTP 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 Requests

How do the work?





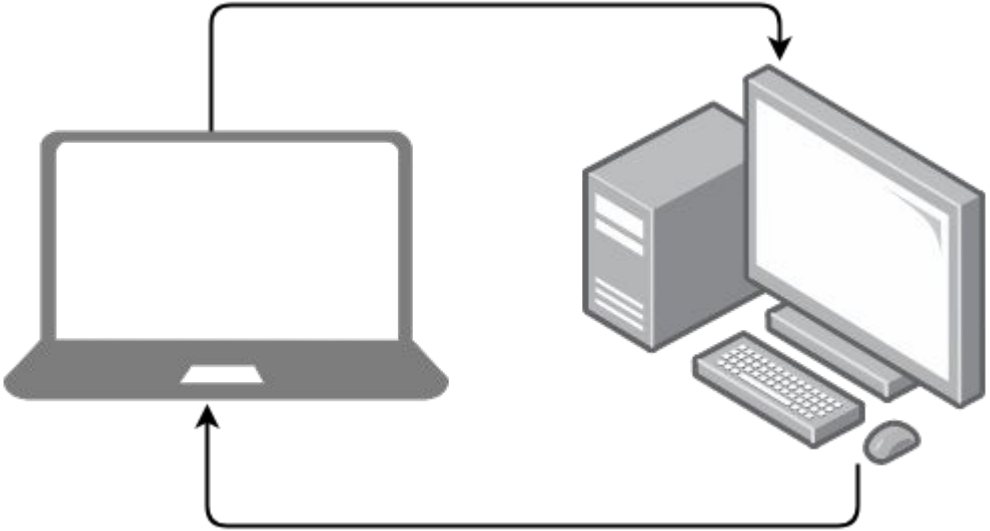
POST Requests

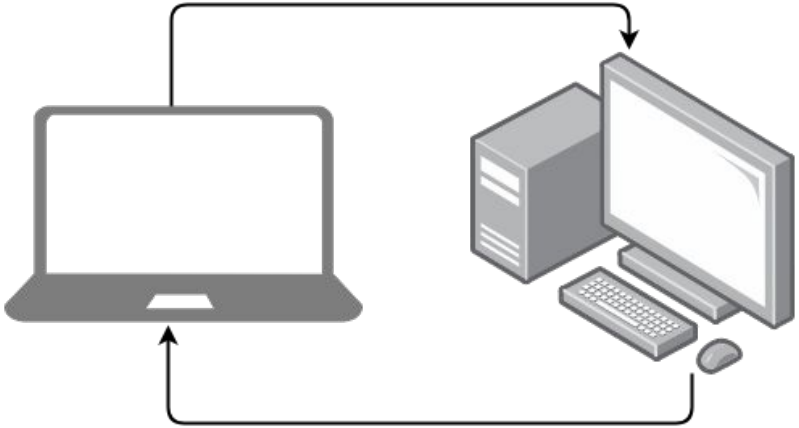
- Do not go through the standard URL, but use a URL as the endpoint
- Ask another computer to create a new resource
- Returns data about the newly created resource

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 Requests

How do the work?





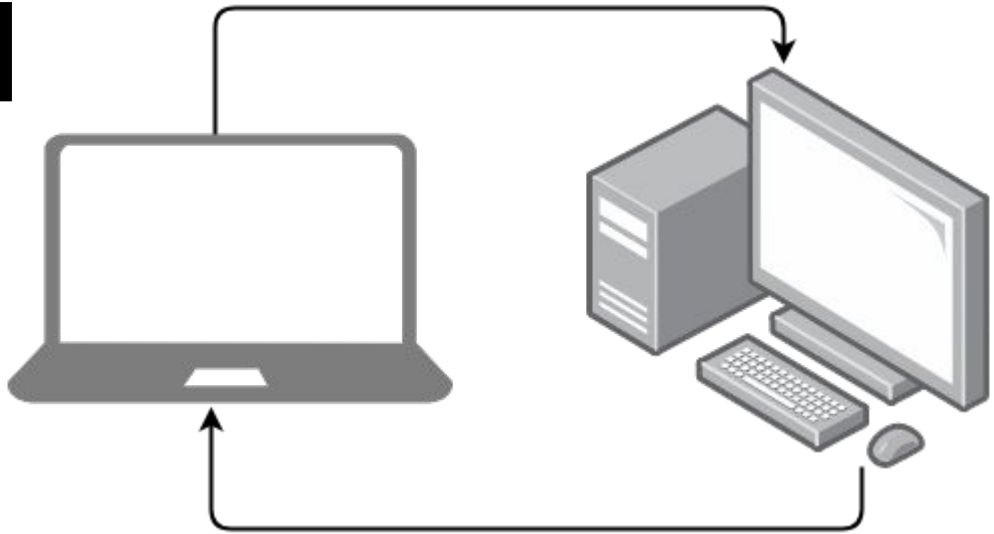
DELETE Requests

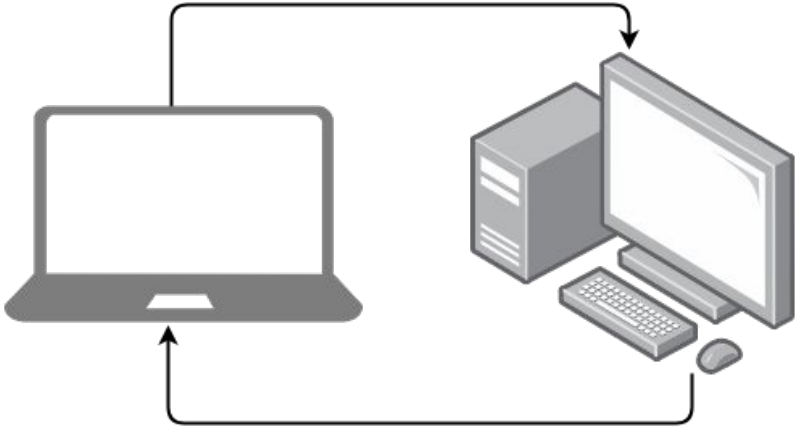
- Do not go through the standard URL, but use a URL as the endpoint
- Ask another computer to delete a single resource or a list of resources
- **Use with caution**

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/PATCH Requests

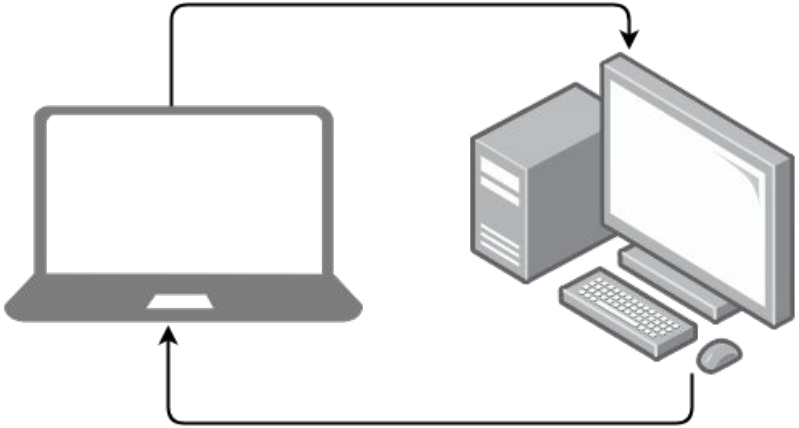
How do the work?





PATCH Requests

- Do not go through the standard URL, but use a URL as the endpoint
- Ask 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
- Example: Updating a users first name



PUT Requests

- Do not go through the standard URL, but use a URL as the endpoint
- Ask another computer to **update an entire** resource
- If the resource doesn’t 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/

More details at https://restfulapi.net/http-methods/

Consuming APIs

Art by iconicbestiary at https://www.freepik.com/iconicbestiary

APIs can be **written** in almost any server-side language.

APIs will generally return one of two types data structures:

JSON or **XML**

JSON Example

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

XML Example

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

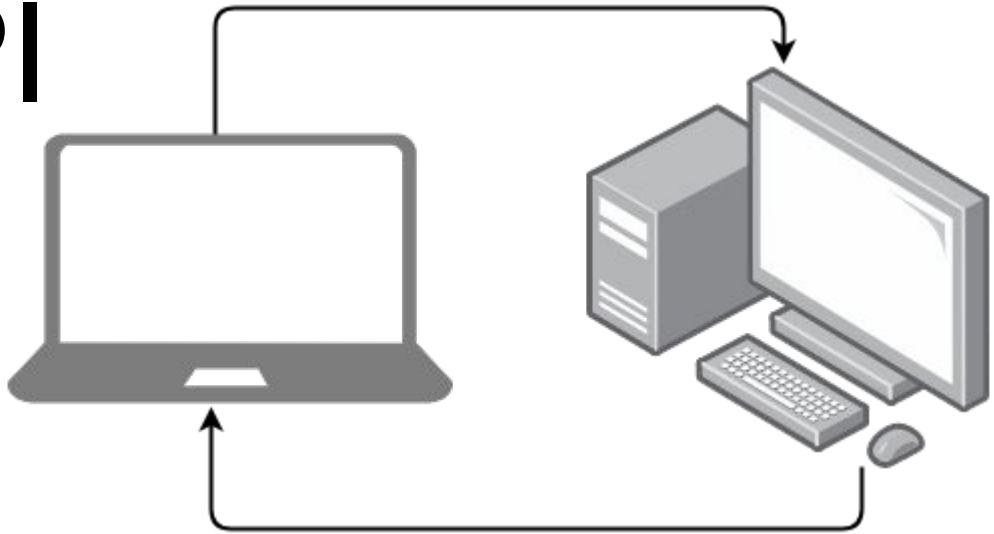
APIs can be **consumed** in almost any language.

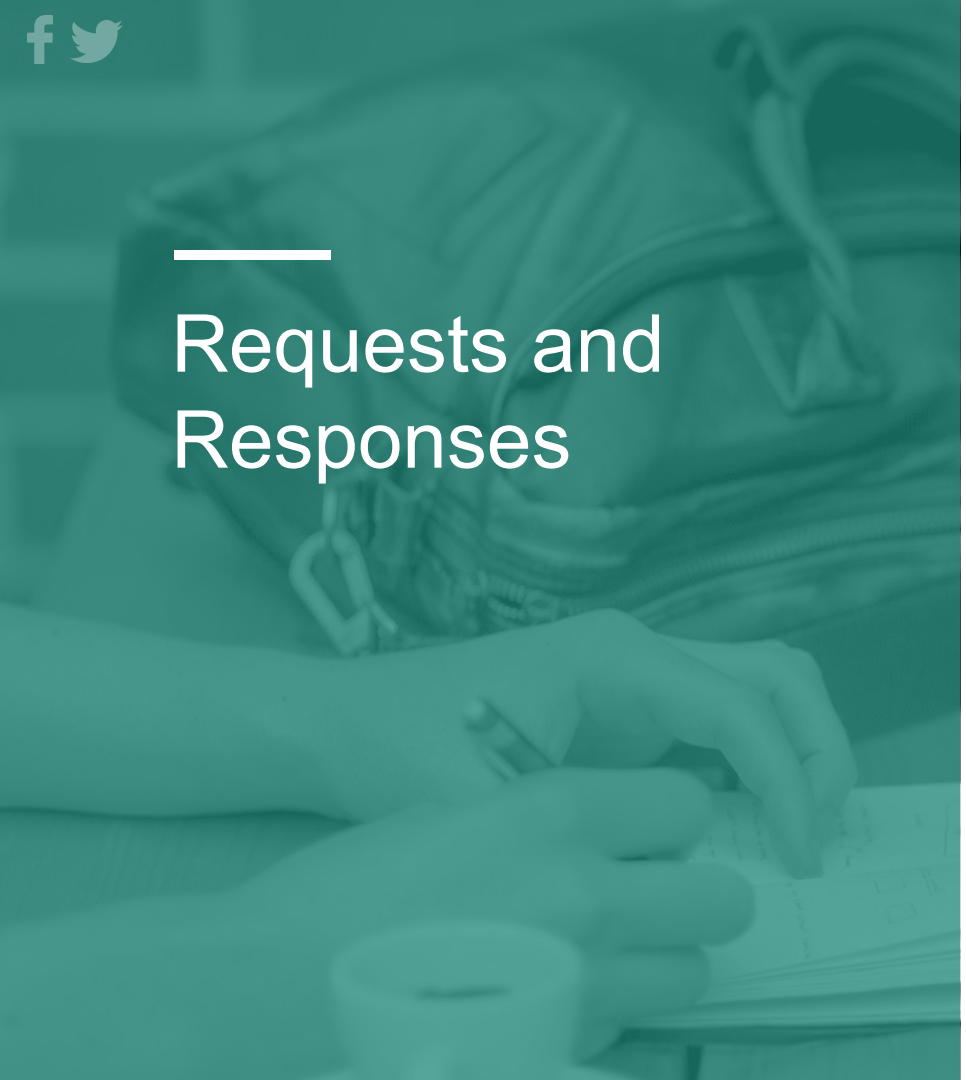
Browsers use JavaScript for their API requests.

Servers use any language that runs on that computer.

Common API Responses

What are they?

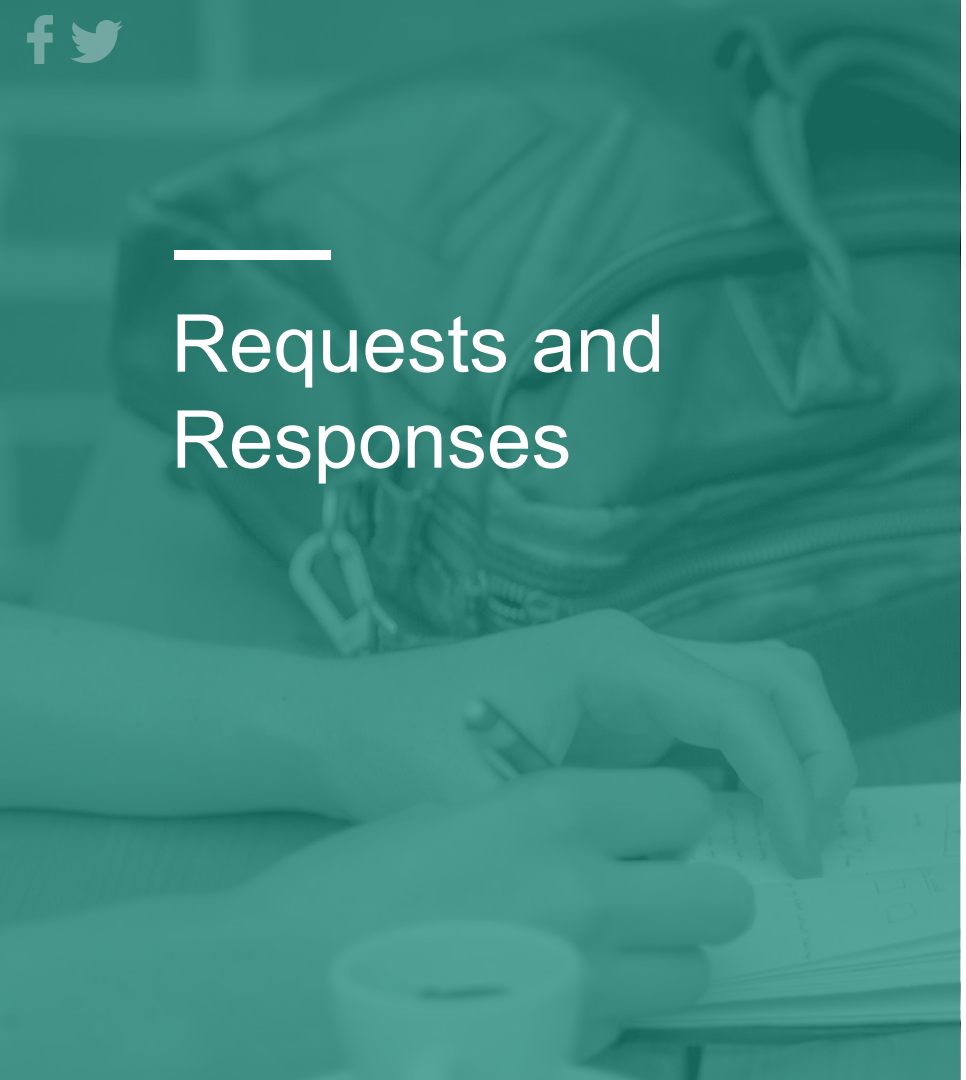




Requests and Responses

When you request data from a server using GET, POST, PUT, PATCH or DELETE… that's a request.

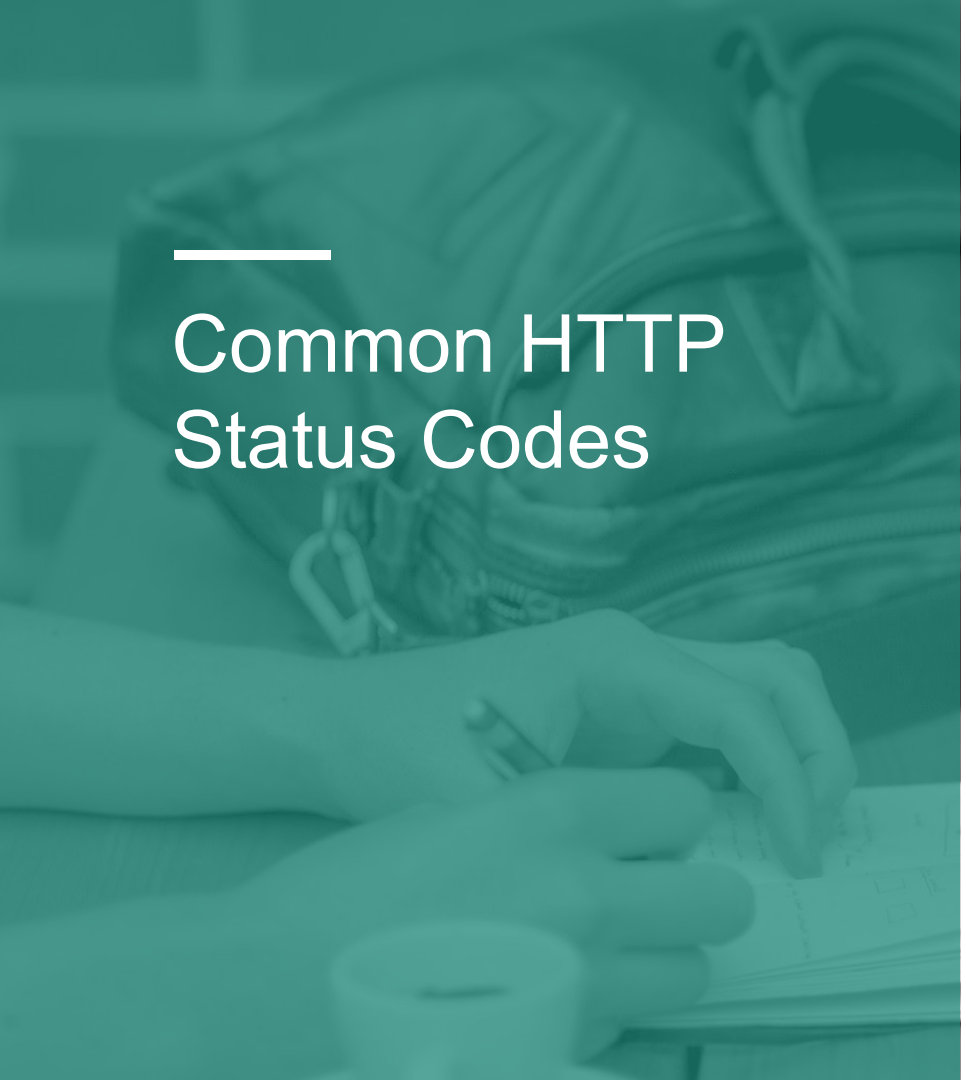
When the server returns your data… that's a response



Requests and Responses

Responses will always come with an HTTP Status Code.

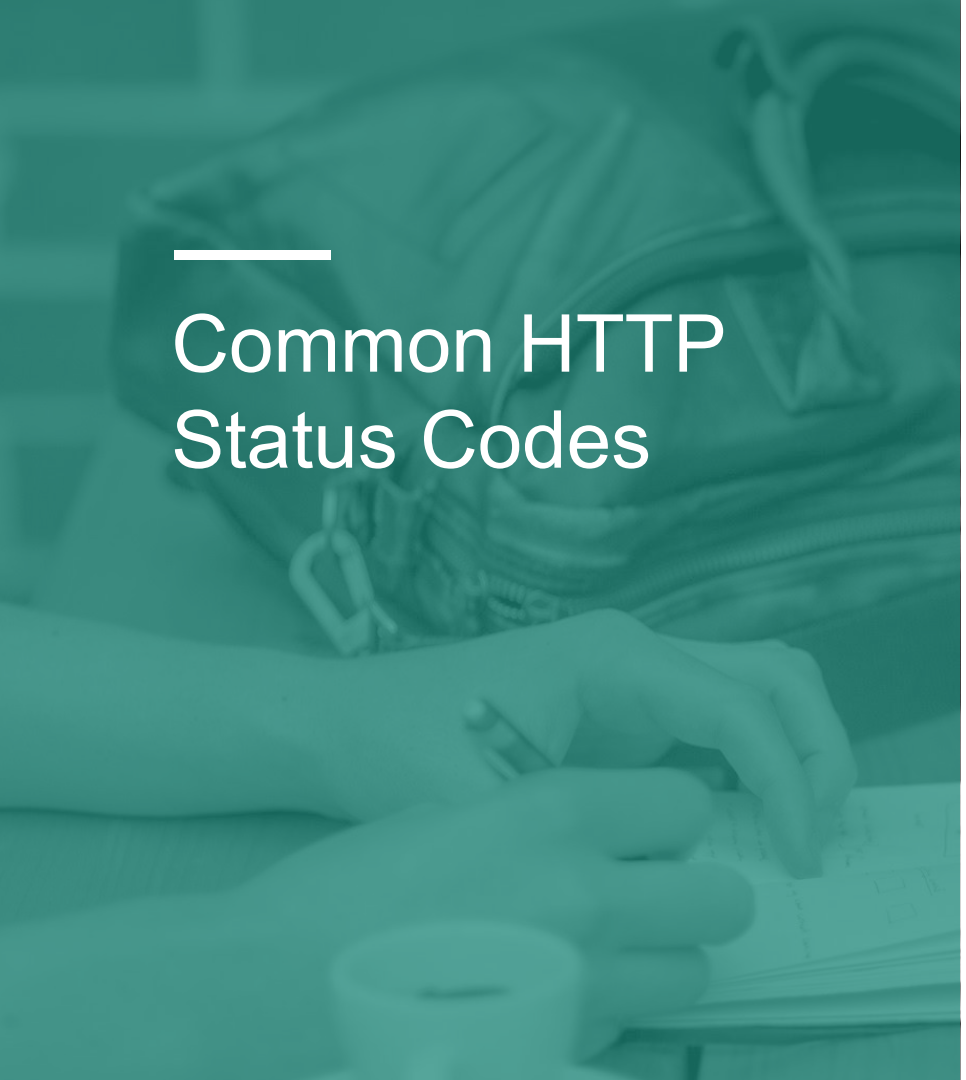
And these “status codes” tell you what’s wrong (or right) without needing to give you text back to read.



Common HTTP Status Codes

Healthy Responses (2--)

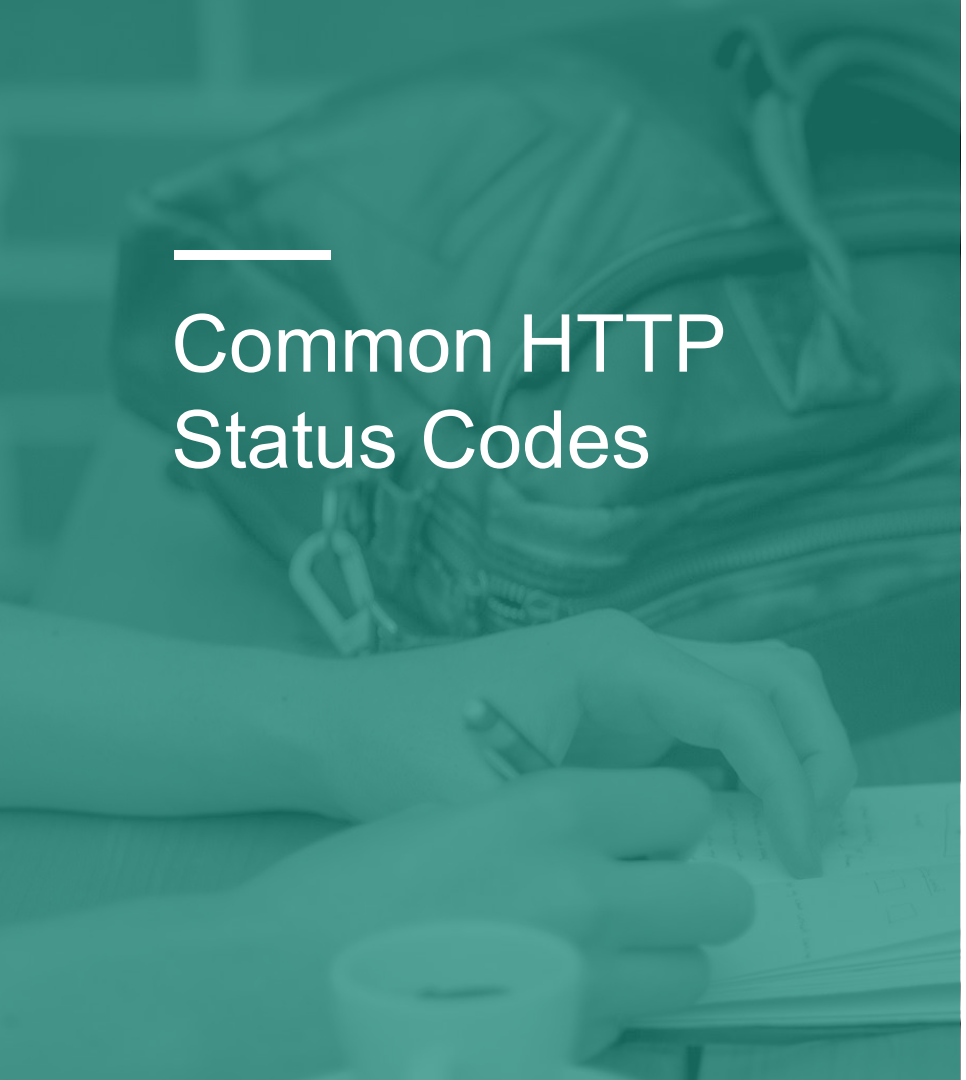
- 200 — OK.
  Request accepted.
- 201 — Created.
  POST requests often return 201s when a resource is created.
- 202 — Accepted.
  When a request is accepted but its not done processing.
  Maybe the task goes into a queue.



Common HTTP Status Codes

Redirect Responses (3--)

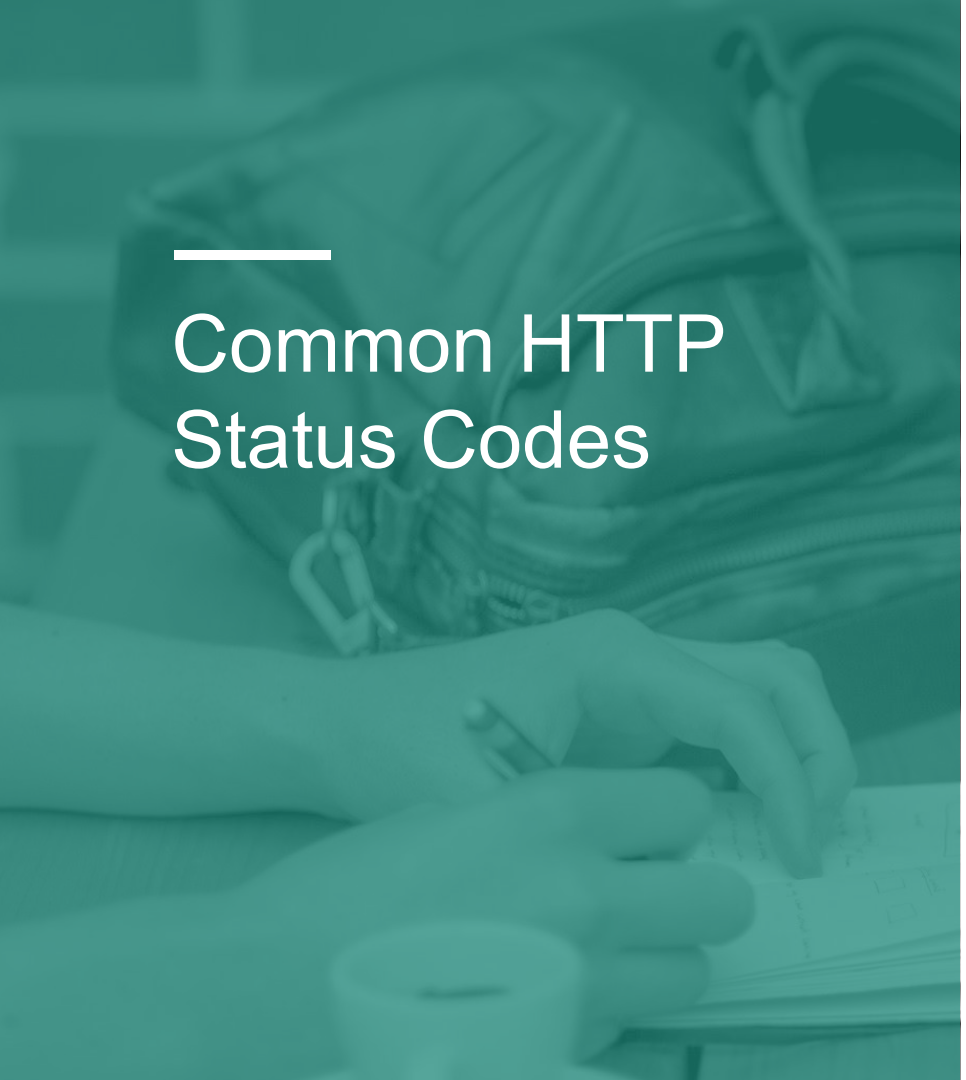
- 301 — Moved Permanently.
  When the endpoint has permanently changed. Update your endpoint.
- 302 — Found.
  The endpoint you’re accessing is temporarily moved to somewhere else.



Common HTTP Status Codes

Client Responses (4--)

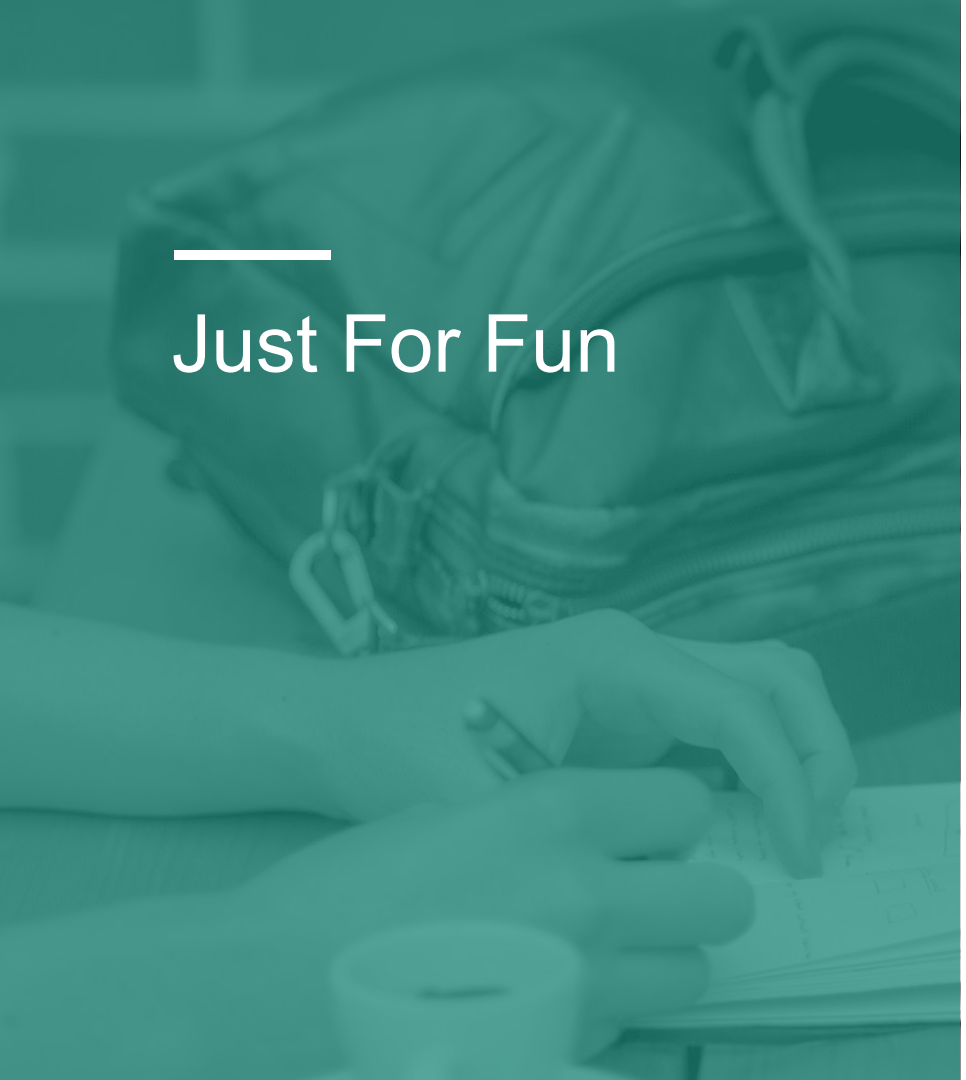
- 400 — Bad Request.
  Server cannot or will not process your request. Often this is due to malformed API keys or an invalid payload.
- 401 — Unauthorized.
  You're not allowed here. Usually this is because you're missing authentication credentials (API keys)
- 403 — Forbidden.
  The servers understands 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.



Common HTTP Status Codes

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**.

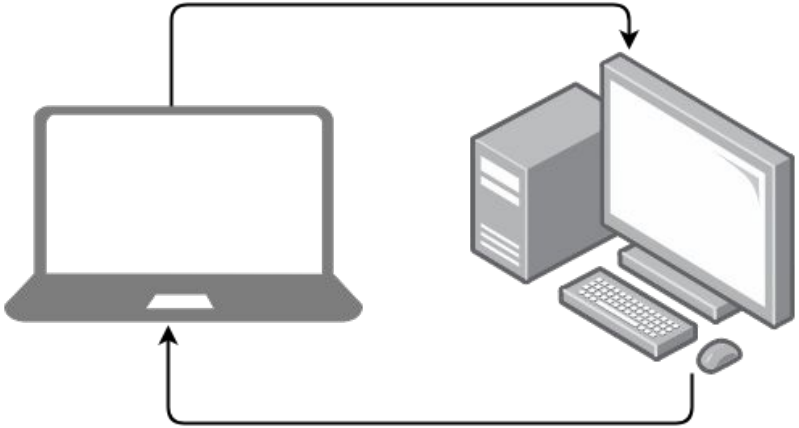


Just For Fun

Find out what a 418 response is

API Security

Art by freepik at https://www.freepik.com/freepik



API Keys

- API keys are “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
  graph.facebook.com/codingforeverybody

https://graph.facebook.com/codingforeverybody/


```
{
  - error: {
        message: "An access token is required to request this resource.",
        type: "OAuthException",
        code: 104,
        fbtrace_id: "Csj1sjSs3GF"
    }
}
```


Access token required.

Access tokens are usually generated with an API key.

Elements | Console | Sources | Network | Performance | Memory | Application | Audits | Security

View: | Group by frame | Preserve log | Disable cache | Offline | No throttling

Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

20 ms | 40 ms | 60 ms | 80 ms | 100 ms | 120 ms | 140 ms | 160 ms | 180 ms | 200 ms | 220 ms

Name	Status	Type
codingforeverybody/	400	document
icon16.png	200	png
FN-close.png	200	png
tab-logo.png	200	png
jsonview-core.css	200	stylesheet
options.png	200	png
favicon.ico	200	x-icon

Matching 400 status code.

“Bad Request”.

Something is missing.

https://graph.facebook.com/codingforeverybody/


```
{
  - error: {
        message: "An access token is required to request this resource.",
        type: "OAuthException",
        code: 104,
        fbtrace_id: "Csj1sjSs3GF"
    }
}
```


Access token required.

Access tokens are usually generated with an API key.

Elements | Console | Sources | Network | Performance | Memory | Application | Audits | Security

View: | Group by frame | Preserve log | Disable cache | Offline | No throttling

Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

20 ms | 40 ms | 60 ms | 80 ms | 100 ms | 120 ms | 140 ms | 160 ms | 180 ms | 200 ms | 220 ms

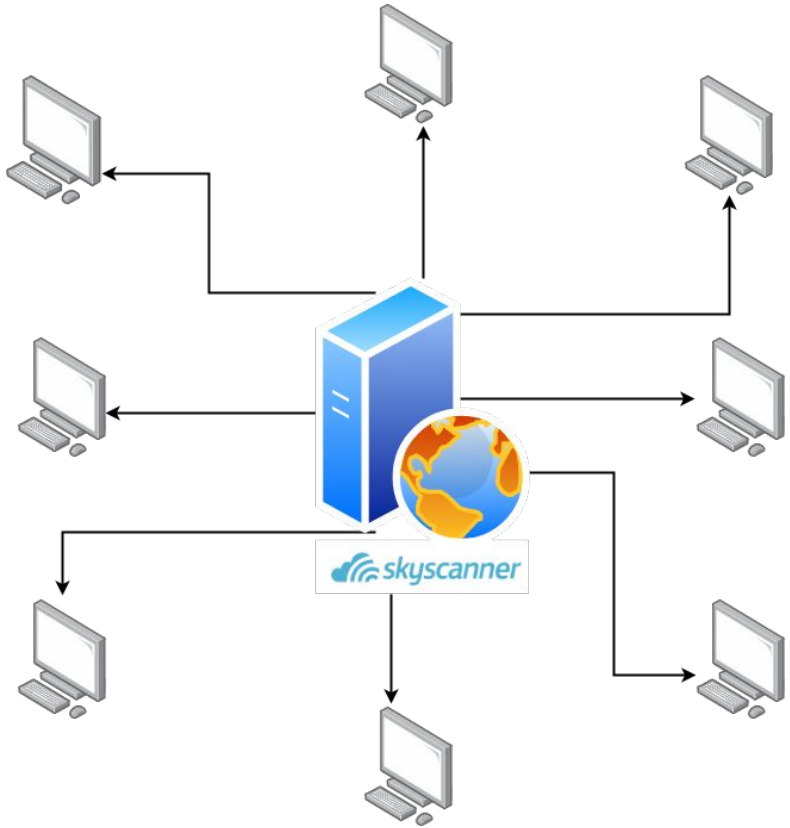
Name	Status	Type
codingforeverybody/	400	document
icon16.png	200	png
FN-close.png	200	png
tab-logo.png	200	png
jsonview-core.css	200	stylesheet
options.png	200	png
favicon.ico	200	x-icon

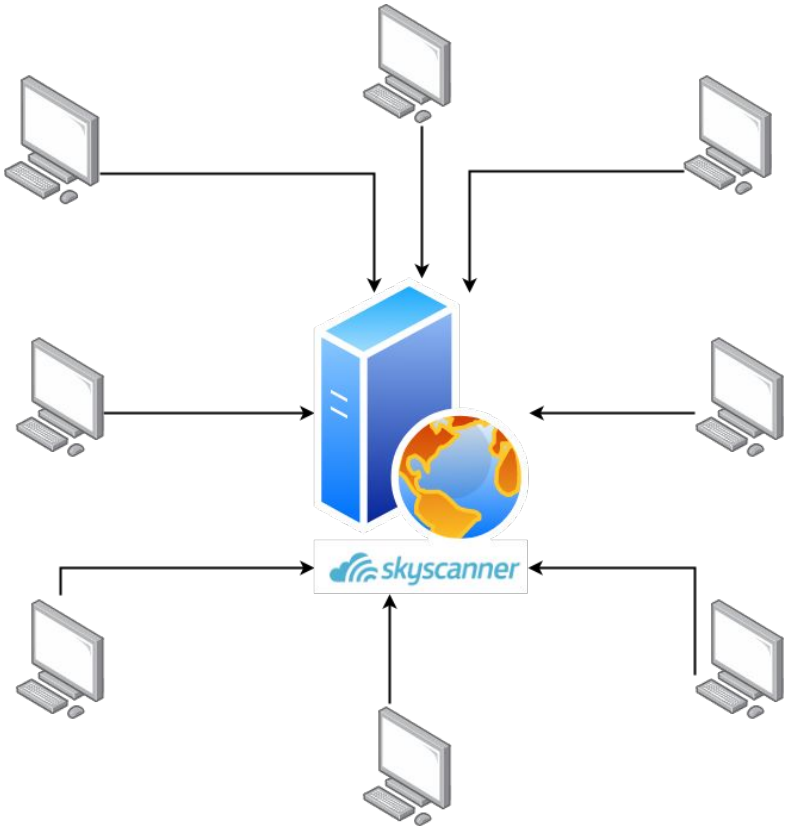
Other services might throw you a 403 or 405 status.

Art by Sapann Design at https://www.freepik.com/sapann-design

Think of an API as a waiter as a restaurant.

Art by iconicbestiary at https://www.freepik.com/iconicbestiary


skyscanner


skyscanner

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/

Most APIs are secured with API keys

Art by freepik at https://www.freepik.com/freepik

Free Resources

- https://restfulapi.net/http-methods/
- https://httpstatuses.com/
- https://swapi.co/

Art by rawpixel.com at https://www.freepik.com/rawpixel-com

RESTful API Cheat Sheet

REpresentational State Transfer

Request Methods

GET
- Used to get data only and does not modify the data at all
- Should return 200, 400 or 404 responses

POST
- Create a new resource (ie. creating a new user)
- Should return 200, 201 or 204 responses

DELETE
- Delete a resource (ie. delete a user)
- Should return 200, 202 or 204 responses

PUT
- Update a resource. If the resource doesn't exist, the api might decide to create it and return a 201 response
- Should return 200, 201 or 204 responses

Common Status Codes

2-- – Success Codes

200	OK
201	Created
202	Accepted
204	No Content

3-- – Redirection Codes

301	Moved Permanently
302	Found

4-- – Client Error Codes

400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed

5-- – Server Error Codes

501	Internal Server Error

Authentication

API keys are used as authentication credentials

CRUD Operations

Create, Read, Update, Delete

@KalebTaulien

Questions?