RESTful API

representational state transfer “full” application programming interface

WHAT IS AN API?

“**an application programming interface**” the keyword is **interface**.

An API is the interface that a software program presents to other programs, to humans, and, in the case of web APIs, to the world via the internet. Although APIs are designed to work with other programs, they’re mostly intended to be understood and used by humans writing those other programs.

- It is a programming interface not a user interface
- It is intended to be limited in what it offers
- It is both Human and Machine readable
- It must be documented by its creators to be useful

AN API WORKS KIND OF LIKE A RESTAURANT

If you go to a restaurant as a customer, you are not allowed to enter the kitchen.

You need to know what is available. For that, you have the menu.

After looking at the menu, you make an order to a waiter, who passes it to the kitchen and who will then deliver what you have asked for.

The waiter can only deliver what the kitchen can provide.

AN API WORKS KIND OF LIKE A RESTAURANT

If you go to a restaurant as a customer, you are not allowed to enter the kitchen.
The waiter is the API. You are someone who is asking for service. You are an API customer or consumer. The API is providing consumables from the server.
You need to know what is available. For that, you have the menu.
The menu is the documentation which explains what you can ask for from the API.
After looking at the menu, you make an order to a waiter, who passes it to the kitchen and who will then deliver what you have asked for.
The kitchen is, for example, a server; a database or something that holds only a certain type of data
The waiter can only deliver what the kitchen can provide.

HOW DOES IT WORK?

An api uses URL's (web addresses) to communicate.

We send a url as a **REQUEST** and the data returned is called a **RESPONSE** .

A REQUEST has 4 parts:

1. The Endpoint
2. The Method
3. The Headers
4. The Data or body

HOW DOES IT WORK?

The endpoint or the root endpoint is really the beginning.
It is the base URL used by the api.

We will be using Github's API as our example:
https://api.github.com
We use paths to define our requested data
https://api.github.com/users/mrMARK-SUNY/repos

We add a query to the end of our url to sort our results.
https://api.github.com/users/mrMARK-SUNY/repos?sort=pushed

WHAT DO WE GET?

The Results are in JSON

JSON Javascript Object Notation

```
{
  "property1": "value1",
  "property2": "value2",
  "property3": "value3"
}
```

This is a data format not a script

Yes the quotes are required...

HOW DOES IT WORK?

The method is the type of request you send to the server.

We have 2 concepts working together here.

The protocols used by web server (HTTP) and the concept of CRUD - create, Read, Update & Delete

Webserver	CRUD
GET	READ
POST	CREATE
PUT	UPDATE
PATCH	UPDATE
DELETE	DELETE

HOW DOES IT WORK?

The headers are just more information about the api

```
curl -H "Content-Type: application/
json" https://api.github.com -v
```

The body is the data that is sent to the server. We use -d or —- data

So lets demo this all together by making a new repo in Github using the api…

WHAT ABOUT REST?

So we haven’t said anything about REST or restfulness or states or transfers.

REST is an approach to designing server systems. It does not remember the ‘state’ of any previous requests. It does not remember changes, or updates or keep track of anything. It sends a fresh copy of what’s there. This allows it to serve quickly with very little overhead.

RESTful API

representational state transfer “full” application programming interface

A lightweight interface to exchange limited data types and control programmatically between systems both local and remote. It is both machine and human readable.