An Introduction to APIs

Application Programming Interface

What is an API & How do they work?

- Code written by someone else that you're allowed to use
- Allows your website/ webapp to communicate with third-party providers and obtain/utilise their information in some way

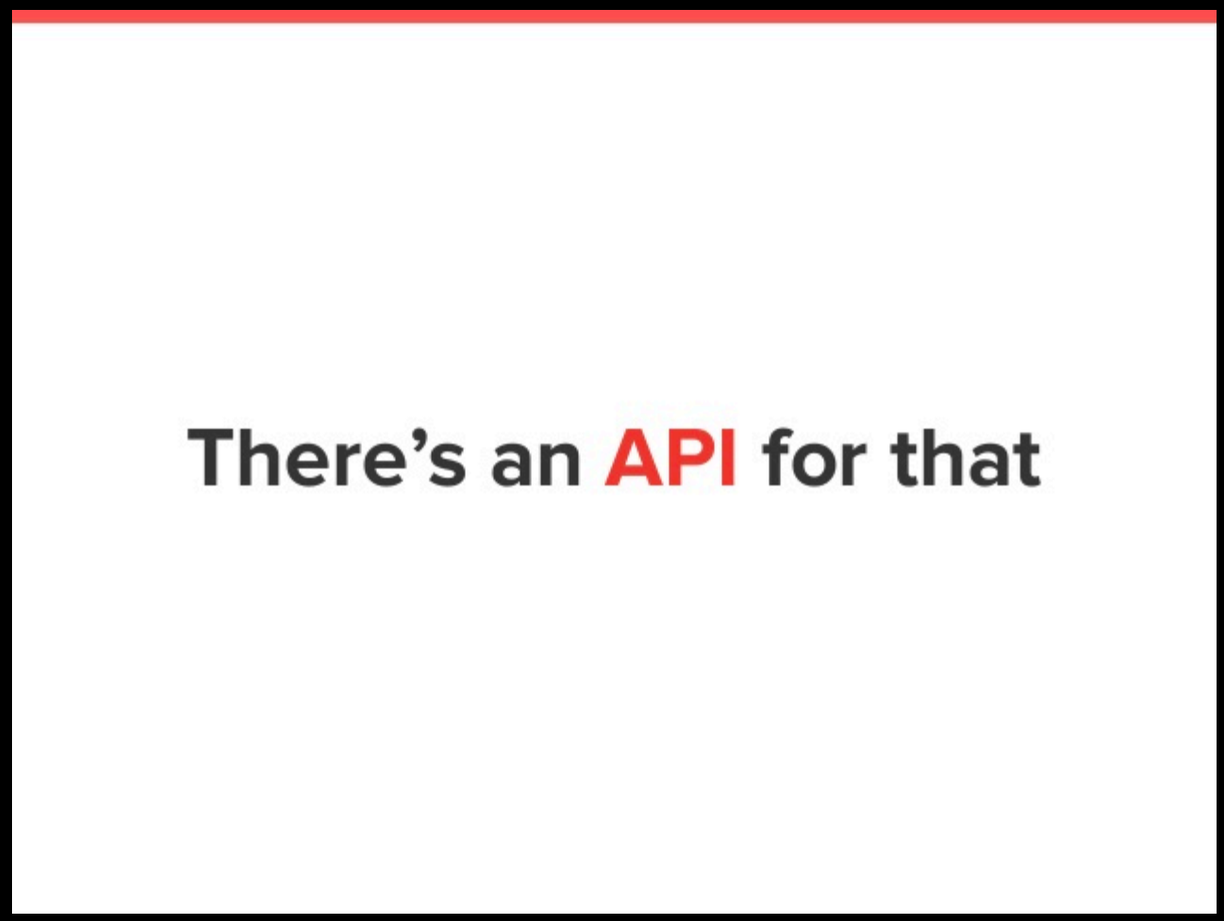
- API is like a messenger: you make a request to a server or a third party, the API carries your request to the server and then returns to you a response
- If it makes it easier, think of it as a waiter who takes your order, delivers it to the kitchen, then comes back with your food

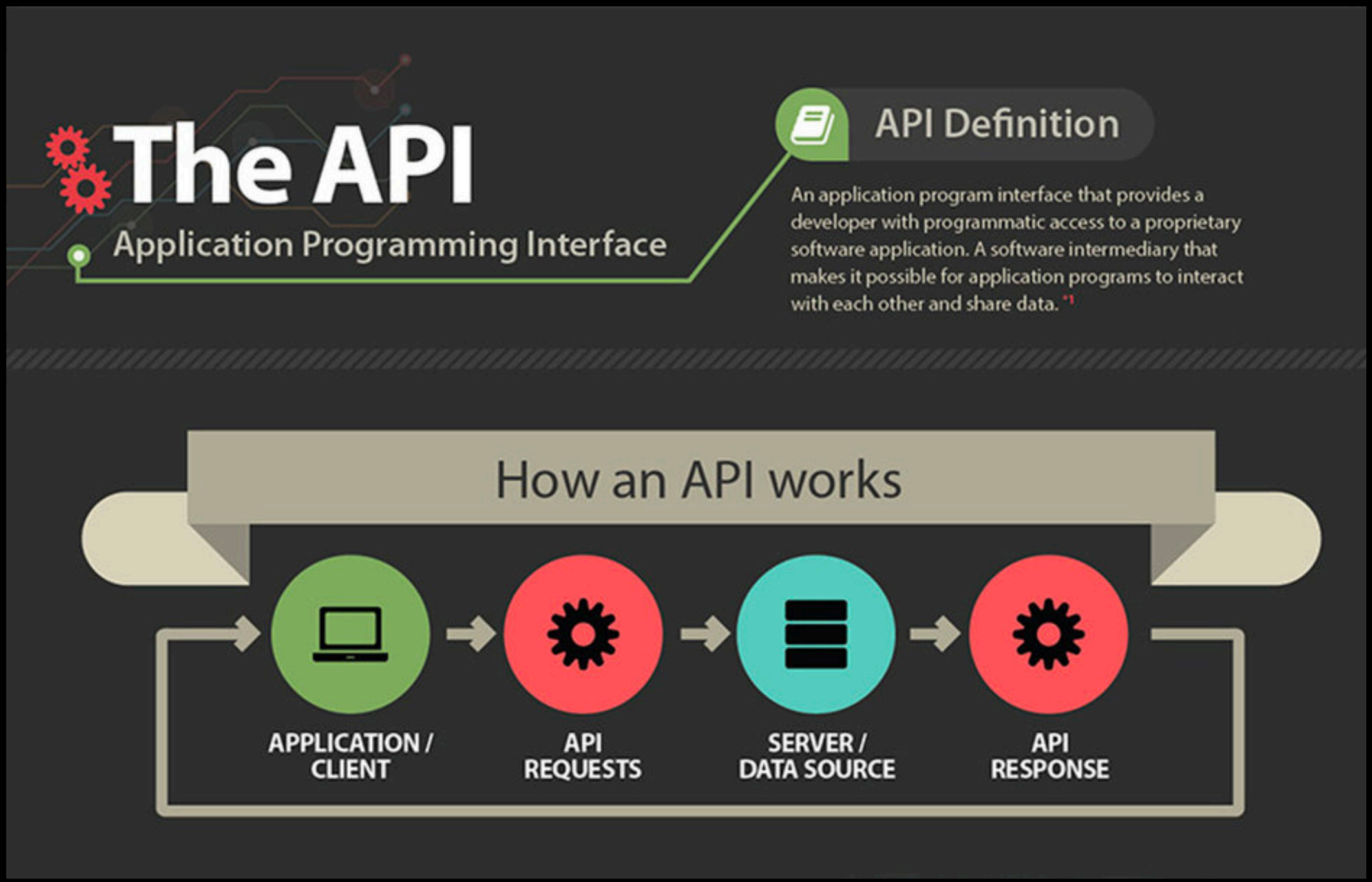
Some examples of APIs you've already interacted with

- Google login
- Google Maps
- Facebook login
- Facebook comments
- Flight and hotel booking websites

Why do we use APIs?

- Because it makes our lives easier
- Why re-invent the wheel - if it's there available for you, you might as well use it
- Access and use raw data (e.g. weather, financial etc.)

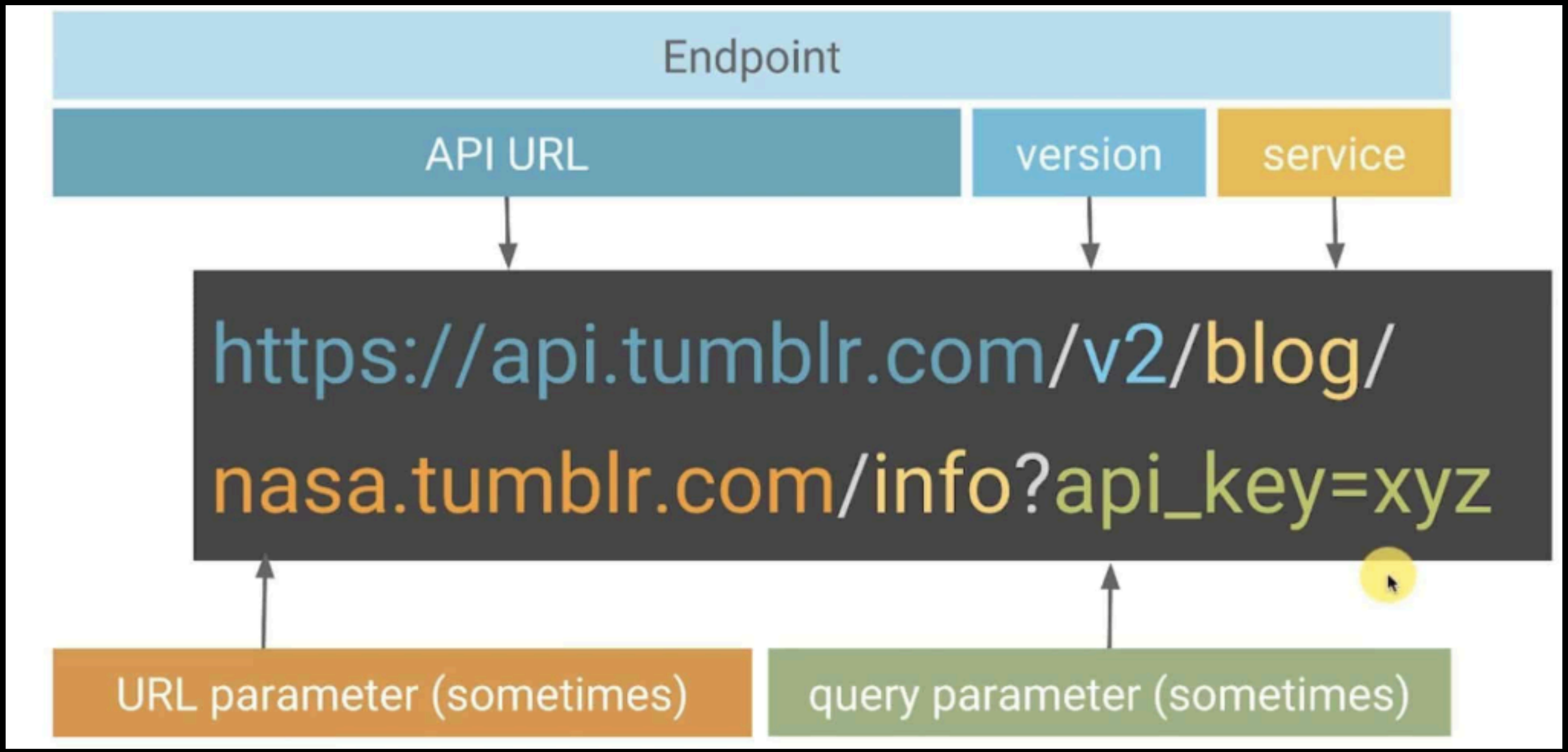



The API
Application Programming Interface
API Definition
An application program interface that provides a developer with programmatic access to a proprietary software application. A software intermediary that makes it possible for application programs to interact with each other and share data. *1
How an API works
APPLICATION / CLIENT
API REQUESTS
SERVER / DATA SOURCE
API RESPONSE

API flow

Anatomy of an API URL/Endpoint

This is what you make your requests to:-



What is an API Key?

- Not all APIs require an API key, but many do
- Especially for APIs that require you to pay
- Unique for everybody and is linked to the user’s identity
- Used to verify whether this user is authorised to use the API
- Used to track and control how an API is being used

DO. NOT. GIVE. YOUR.
API. KEY. TO. ANYONE.

Types of Requests: GET & POST

- GET request - makes a request to GET information from the server (e.g. generate random photos of dogs, cats, desserts, or any other information)
- POST request - makes a request to POST information to the server (e.g. login - you send username & password to check if you are an existing user)

JSON - JavaScript Object Notation

Your response will be in JSON format


```
{
  "meta": {
      "status": 200,
      "msg": "OK"
  },
  "response": [
    {
        "type": "photo",
        "blog_name": "dessertgallery",
        "blog": {
            "name": "dessertgallery",
            "title": "THE DESSERT GALLERY",
            "description": "There's always, always, ALWAYS room for Dessert!",
            "url": "https://dessertgallery.tumblr.com/",
            "uuid": "t:291zBtw8HymcOcYI7OsPZA",
            "updated": 1570347019
        },
        "id": 188164493084,
        "post_url": "https://dessertgallery.tumblr.com/post/188164493084/healt
        "slug": "healthy-carrot-cake-bars-your-source-of-sweet",
        "date": "2019-10-06 07:30:19 GMT",
        "timestamp": 1570347019,
        "state": "published",
        "format": "html",
        "reblog_key": "PtFr1DWS",
        "tags": [
            "cakes",
            "desserts",
            "recipes",
            "foodporn",
            "food photography",
            "pastries"
```

fetch()

```
fetch("https://api.chucknorris.io/jokes/random")
  .then(function(response) {
    return response.json();
  })
.then(function(data) {
  console.log(data.value);
});
```

1. fetch() function takes in the endpoint & makes a GET request
2. 1st .then => we convert the HTTP response into an object that we can use i.e. into JSON format
3. 2nd .then => returns the JSON data so we can read/use it
4. Do what you want with the data! Display it, append it somewhere etc. etc.

pass in API URL

call fetch

then

get json

then

use json

```
fetch('https://api.tumblr.com/v2/tagged?tag=gif')
  .then(function(response){
    return response.json(); //get json object
  }).then(function(json){
    console.log(json) //use json object here
  });
```

AJAX call

Asynchronous JavaScript & XML

```
$.ajax({
  url: "https://api.chucknorris.io/jokes/random",
  method: "GET",
  success: function(result) {
    console.log(result.value);
  },
  error: function(error) {
    console.log(`Error: ${error}`);
  }
});
```

1. Calls the URL endpoint

2. Indicates that it’s a GET request

3. If successful, do something with the response

4. If unsuccessful, console.log the error