# Stage -1

## HTTPS

#### **Primary goals**

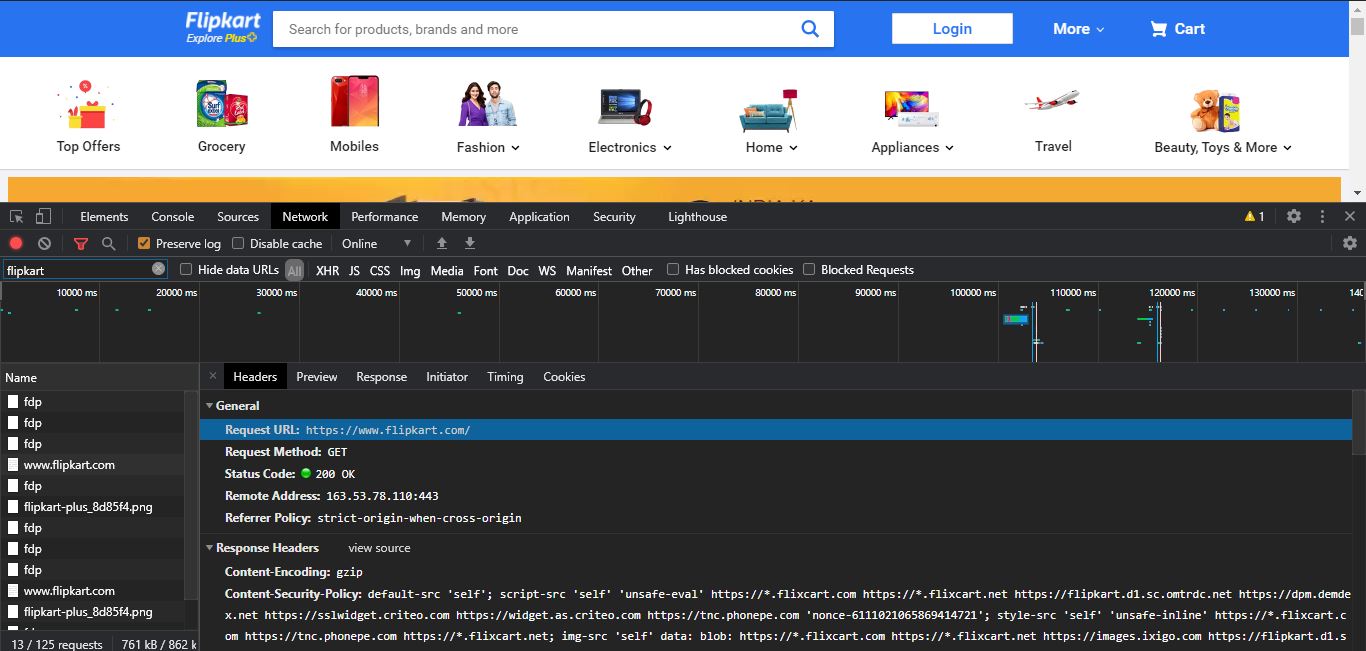
Get a clear understanding of how HTTP works.

Use tools like cURL & Postman to perform HTTP requests and analyse responses

#### **Understanding HTTP using Browsers**

HTTP stands for HyperText Transfer Protocol. It is, like the name suggests, a set of rules for querying the web. It works on a client-server model, where the client, in most cases, the browser, makes a request, and waits for the server to respond.

Browsers use HTTP Requests to fetch us web pages. When we enter a website URL, the browser creates a HTTP Request on our behalf and sends it to the server on which the website is hosted. The HTTP Response from the server is read by the browser and rendered for us beautifully as web pages instead of the raw HTML returned.



While inspecting we have req url, req method, status code: 200 ==> ALL OK ,

**Remote Address:**

1. Dig command (Domain Information Groper): for querying Domain Name System (DNS) name servers , ====> linux : dig [www.crio.do](http://www.crio.do)

*As soon as u type the domain name. Browser will ask the OS I want to read this crio.do , who should I talk to. OS contacts DNS servers(ISP) and send a query or request.*

*DNS server sends a list of possible IP’s where u can contact crio.do. And u’ll get the IP of crio server from the cloud.*

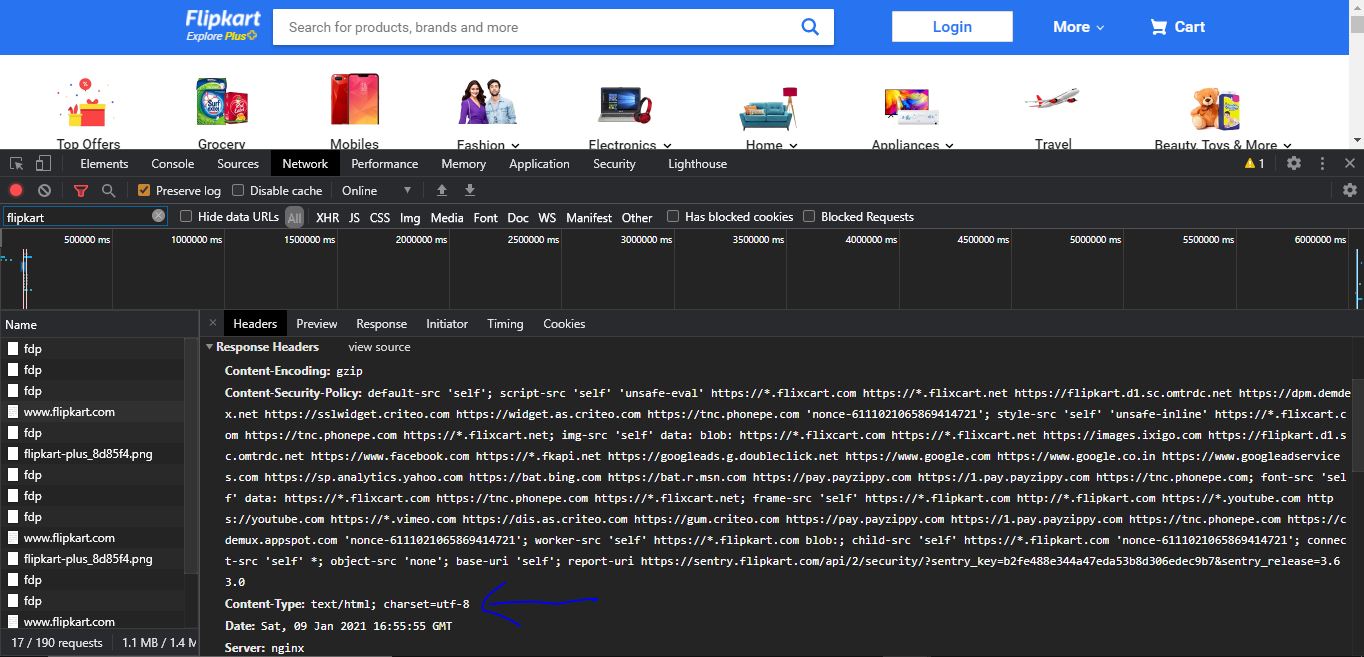
*When a user wants to access a webpage, they enter the address (or DNS name) into their browser's address bar. This information is then sent in the form of a 'DNS query' to their Internet Service Provider's (ISP) DNS servers. Every ISP has a database of DNS names and their corresponding IP addresses.*

REMOTE ADDRESS HAS **443** IN THE END WHICH BASICALLY DEPENDS ON THE PROTOCOL USED.

While using https:// -----> 443 (port no)

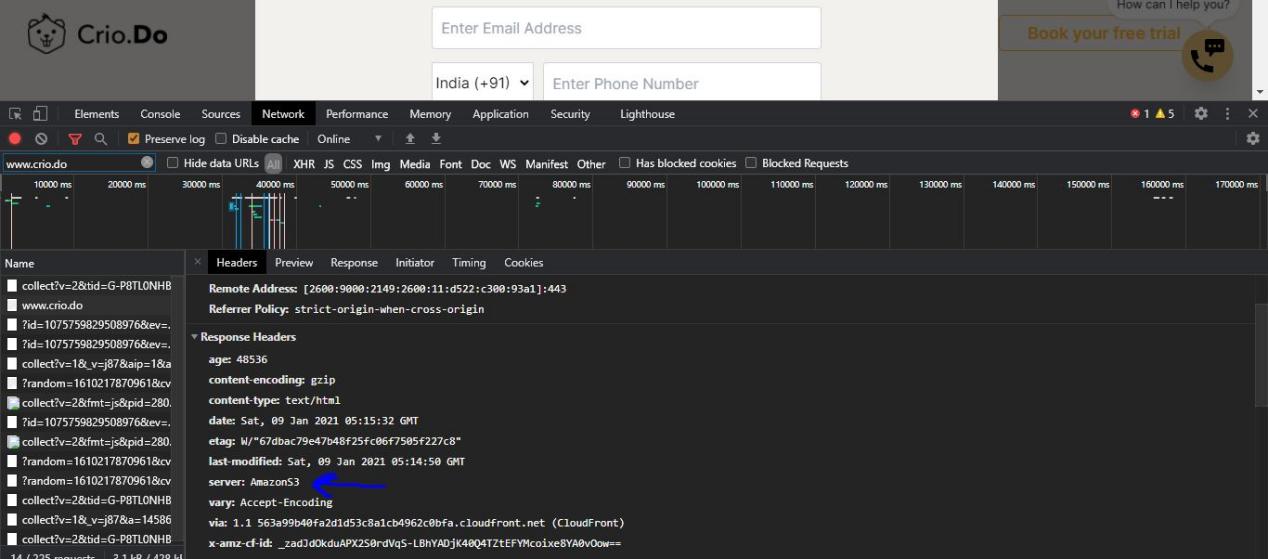
While using http://-------> 80(port no)

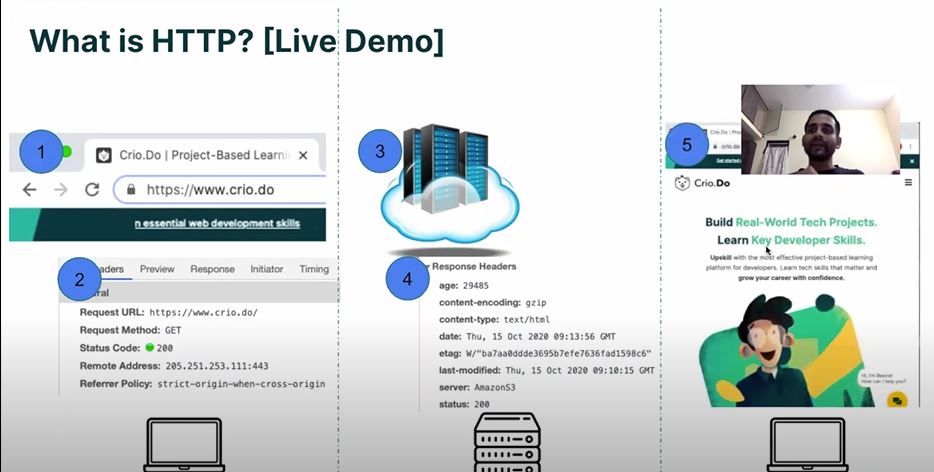
**CONTENT TYPE**



Inspecting crio.do

It is hosted on amazons3 amazon web services --->CLOUD SERVER



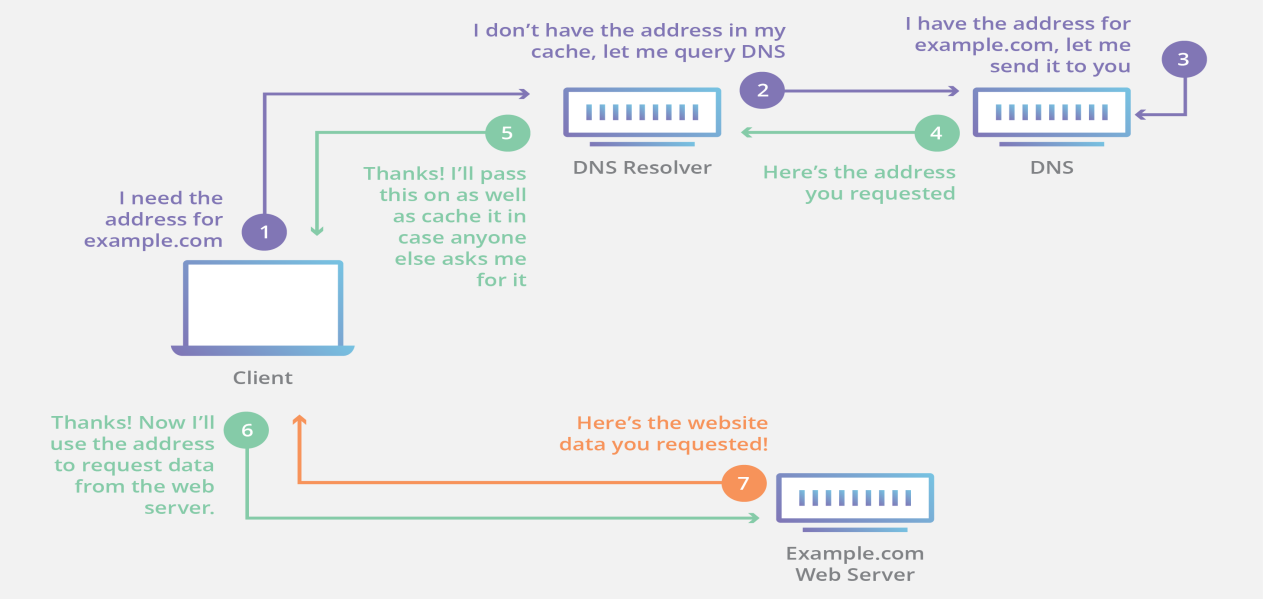


**DEFINITION (BASIC)**

IP stands for Internet Protocol, which is the set of rules that makes it possible for devices to communicate over the Internet.

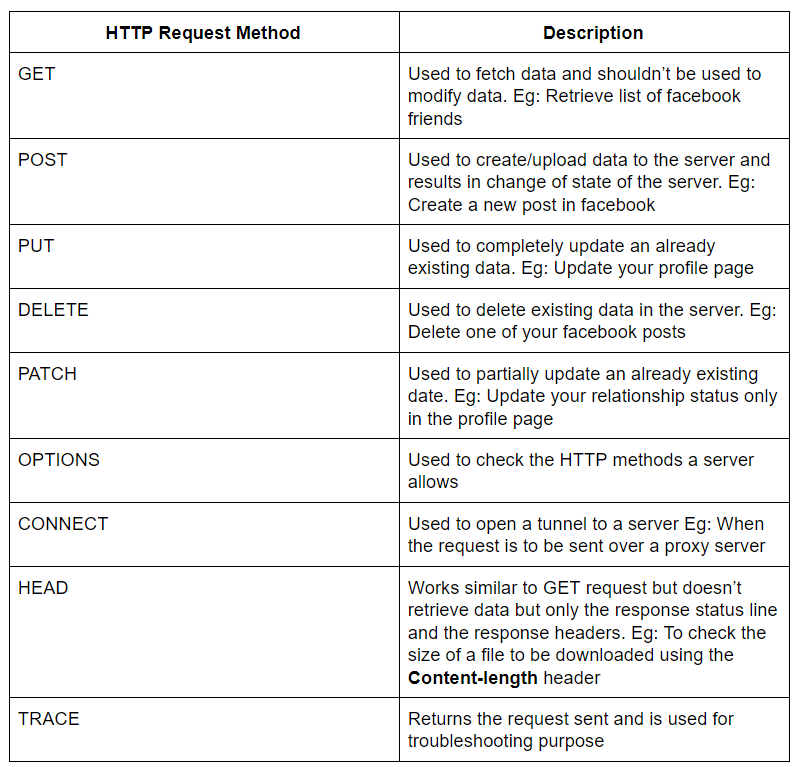
When a user types a domain name, like google.com, into a web browser, this will initiate a request to Google’s web server asking for content (the Google homepage). Once Google receives the request, it needs to know where to send the website content. For this reason, the request will contain the asker’s IP address. Using the provided IP address, Google can send a response back to the user’s device, which will then display that content in the user’s web browser.

The system that orchestrates all this is called DNS. It works like a phone book for IP addresses so that users can access web services using human-friendly domain names. When a user types a domain name like ‘facebook.com’ into their browser window, this begins a DNS query which ultimately leads to a DNS server translating the domain name into an IP address.



1. <https://learn.onemonth.com/understanding-http-basics/>
2. <https://personal.ntu.edu.sg/ehchua/programming/webprogramming/HTTP_Basics.html>
3. <https://developers.google.com/web/tools/chrome-devtools/network>

### CHAP-2



**HTTP GET**

GET APIs should be idempotent, which means that making multiple identical requests must produce the same result every time until another API (POST or PUT) has changed the state of the resource on the server.

For any given HTTP GET API, if the resource is found on the server, then it must return HTTP response code 200 (OK) – along with the response body, which is usually either XML or JSON content (due to their platform-independent nature).

In case resource is NOT found on server then it must return HTTP response code 404 (NOT FOUND). Similarly, if it is determined that GET request itself is not correctly formed then server will return HTTP response code 400 (BAD REQUEST).

## **HTTP POST**

if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a [Location](https://en.wikipedia.org/wiki/HTTP_location" \t "https://restfulapi.net/http-methods/_blank) header.

Many times, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status.

POST is neither safe nor idempotent, and invoking two identical POST requests will result in two different resources containing the same information (except resource ids)

[Home](https://restfulapi.net/" \o "REST API Tutorial) / [Resources](https://restfulapi.net/tutorial/resources/) / HTTP Methods

# **HTTP Methods**

REST APIs enable you to develop any kind of web application having all possible CRUD (create, retrieve, update, delete) operations. [REST guidelines](https://restfulapi.net/rest-architectural-constraints/) suggest using a specific HTTP method on a particular type of call made to the server (though technically it is possible to violate this guideline, yet it is highly discouraged).

Use below-given information to find a suitable HTTP method for the action performed by API.

Table of Contents

[HTTP GET](https://restfulapi.net/http-methods/" \l "get)[HTTP POST](https://restfulapi.net/http-methods/" \l "post)[HTTP PUT](https://restfulapi.net/http-methods/" \l "put)[HTTP DELETE](https://restfulapi.net/http-methods/" \l "delete)[HTTP PATCH](https://restfulapi.net/http-methods/" \l "patch)[Summary](https://restfulapi.net/http-methods/" \l "summary)[Glossary](https://restfulapi.net/http-methods/" \l "glossary)

## **HTTP GET**

Use GET requests ****to retrieve resource representation/information only**** – and not to modify it in any way. As GET requests do not change the state of the resource, these are said to be ****safe methods****. Additionally, GET APIs should be ****idempotent****, which means that making multiple identical requests must produce the same result every time until another API (POST or PUT) has changed the state of the resource on the server.

If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

For any given HTTP GET API, if the resource is found on the server, then it must return HTTP response code 200 (OK) – along with the response body, which is usually either XML or JSON content (due to their platform-independent nature).

In case resource is NOT found on server then it must return HTTP response code 404 (NOT FOUND). Similarly, if it is determined that GET request itself is not correctly formed then server will return HTTP response code 400 (BAD REQUEST).

#### **Example request URIs**

* HTTP GET http://www.appdomain.com/users
* HTTP GET http://www.appdomain.com/users?size=20&page=5
* HTTP GET http://www.appdomain.com/users/123
* HTTP GET http://www.appdomain.com/users/123/address

## **HTTP POST**

Use POST APIs ****to create new subordinate resources****, e.g., a file is subordinate to a directory containing it or a row is subordinate to a database table. When talking strictly in terms of REST, POST methods are used to create a new resource into the collection of resources.

Ideally, if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a [Location](https://en.wikipedia.org/wiki/HTTP_location" \t "https://restfulapi.net/http-methods/_blank) header.

Many times, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status.

Responses to this method are ****not cacheable****, unless the response includes appropriate [Cache-Control](https://en.wikipedia.org/wiki/Web_cache" \l "Cache_control" \t "https://restfulapi.net/http-methods/_blank) or [Expires](https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html" \t "https://restfulapi.net/http-methods/_blank) header fields.

Please note that POST is ****neither safe nor idempotent****, and invoking two identical POST requests will result in two different resources containing the same information (except resource ids).

#### **Example request URIs**

* HTTP POST <http://www.appdomain.com/users>
* HTTP POST <http://www.appdomain.com/users/123/accounts>

**HTTP PUT**

Use PUT APIs primarily to update existing resource (if the resource does not exist, then API may decide to create a new resource or not). If a new resource has been created by the PUT API, the origin server MUST inform the user agent via the HTTP response code 201 (Created) response and if an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are not cacheable.

The difference between the POST and PUT APIs can be observed in request URIs. POST requests are made on resource collections, whereas PUT requests are made on a single resource.

**HTTP DELETE**

As the name applies, DELETE APIs are used to delete resources (identified by the Request-URI).

A successful response of DELETE requests SHOULD be HTTP response code 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has been queued, or 204 (No Content) if the action has been performed but the response does not include an entity.

DELETE operations are idempotent. If you DELETE a resource, it’s removed from the collection of resources. Repeatedly calling DELETE API on that resource will not change the outcome – however, calling DELETE on a resource a second time will return a 404 (NOT FOUND) since it was already removed. Some may argue that it makes the DELETE method non-idempotent. It’s a matter of discussion and personal opinion.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are not cacheable.

|  |  |  |  |
| --- | --- | --- | --- |
| PATCH | Partial Update/Modify | 405 (Method not allowed), unless you want to modify the collection itself. | 200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid. |
| DELETE | Delete | 405 (Method not allowed), unless you want to delete the whole collection — use with caution. | 200 (OK). 404 (Not Found), if ID not found or invalid. |

<https://restfulapi.net/http-methods/>

When a browser loads a webpage, it has to make an HTTP request for each image, which slows down the overall time it takes to completely load the page.

## CHAPTER 3 : STATUS CODES

HTTP Status codes are part of the HTTP Response. It helps the client understand what happened to the request. Status codes are 3 digit numbers (201, 304 etc) and are categorised to 5 different families based on their starting digit. Along with the status code, a Reason-Phrase is also present (OK, Moved Permanently etc) which gives a short description of the status code. The Status Code is intended for machines whereas Reason-Phrase is for humans.

#### **Status codes - 2xx**

The 2xx family of status codes or status codes 200-299 signifies the HTTP request was successfully received & understood by the server. We’ve been seeing the 200 status codes all the way until now. That’s what we get when the server returns some resource for our request.

#### **Status codes - 3xx**

3xx family of status codes denotes that further action must be taken to complete the HTTP request made.

(MOVED PERMANENTLY)

#### **Status codes - 4xx**

Getting a 4xx status code tells us that there was an error in the HTTP request sent by the client - that would be the browser if we are visiting web pages.

There are a couple more HTTP status code families - 1xx & 5xx. 1xx is for information purposes while 5xx signifies there was a server error.

1. Informational responses (100–199)
2. Successful responses (200–299)
3. Redirects (300–399)
4. Client errors (400–499)
5. Server errors (500–599)

**USING cURL AND POSTMAN**

cURL is like a web-browser, but for the command line.You can make HTTP requests using cURL just like in a web-browser. The Response can be seen on the command line or redirected to a file.

<https://www.freecodecamp.org/news/how-to-start-using-curl-and-why-a-hands-on-introduction-ea1c913caaaa/>

## COMPOSING HTTP REQUEST

 The telnet client helps us connect to other computers on the internet.

<http://telnet.browseas.com/> : online telnet

1. Client initiates a TCP connection request to the server (Line 1) - this is performed when we execute the telnet command

*gitpod /workspace/me\_http $* ***telnet www.google.fr 80***

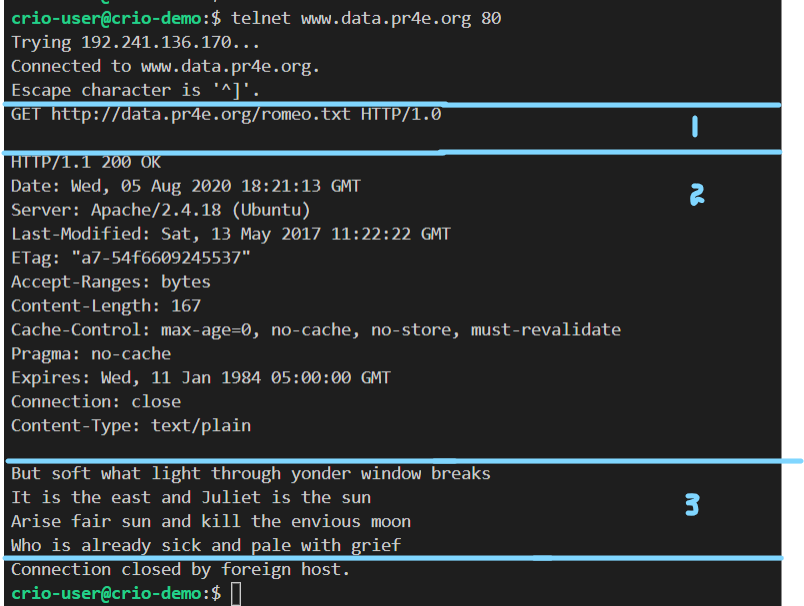
Trying 216.58.201.227...

Connected to www.google.fr.

Escape character is '^]'.

1. HTTP communication happens using this established TCP connection

*GET / HTTP/1.1*



**QUESTIONS**

1. Open a browser tab in Incognito. Visit https://crio.do/ after opening the Networks tab in DevTools.

Observe the size of data transferred. Open a new tab and do the same. Is there a difference in the size of data transferred now? Inspect the request & response headers in both situations to find out what’s happening.

How 'Last-Modified' and 'If-Modified-Since' headers work?

'Last-Modified' response header is sent by the server to specify the last modified date of the requested resource.

'If-Modified-Since' request header is sent by the browser to specify the last modified date of the cached copy of the resource (saved in the browser). The value of this header is same as the value of the previously received 'Last-Modified' header.

**Milestone 1**

Q 2 . Suppose Chrome versions below 80.0 don't support GIF images. We need our server to return a corresponding PNG image if any unsupported browser asks for the GIF image. How would the server know the Chrome version from which the request was made?

Browsers sends a User-Agent request header along with HTTP requests to denote the software it’s using. If you check the request headers sent to the Flipkar server on visiting <https://www.flipkart.com/>, you’ll see something like this

|  |
| --- |
| User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36 |

So the Chrome version here is 84.0 and the server can parse this to check if the browser supports GIF.

Q 2. We looked at how requesting for a HTML file inturn creates a new HTTP request to fetch resources like scripts & images within it. Visit a couple of websites & inspect the resources loaded. Is there any order in which the resources are loaded? Does HTTP mandate this?

Though it might seem CSS & JavaScript files are preferred over images, HTTP doesn’t favor any particular type of files to be loaded first. Further HTTP requests to fetch required resources for a page are made asynchronously meaning that each request is made independently without waiting for other requests to complete. As images are mostly of larger size than other resources, these requests get completed the last.Order in which resources are requested can also depend on their relative ordering in the HTML file.

Q 3. HTTP is a ‘stateless protocol’, meaning two corresponding requests do not share data - your prior request is not ‘remembered’ in any way by the following one - this obviously has some flip sides - you might need to keep resending data that you want to persist through requests - why is it still designed this way?

The reason for statelessness is load-balancing : modern day web isn’t just about one server or one client always - your one request could go to a server in North Virginia, the next to someplace in California - to manage traffic. And this means that you can’t really have one server holding on to data that it might never be using.

## MILESTONE 2.

Q1. Is it possible to send form data using a GET request? Why or why not?

Yes, it’s possible though not recommended. Usually, form data contains fields that are sensitive like passwords and using GET requests for submitting these means your password will be out in the open along with the request URL.

<https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending_and_retrieving_form_data#The_GET_method>

Data in a GET request is sent as part of the URL and this has a limit of 2048 characters.

[What is the URL character limit for get requests?](https://helpx.adobe.com/in/experience-manager/scene7/kb/base/is_protocol-_-forming_is/url-character-limit-get-requests.html)

[Can HTTP POST be limitless?](https://stackoverflow.com/questions/2880722/can-http-post-be-limitless)

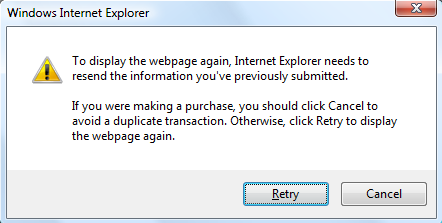
## MILESTONE 3.

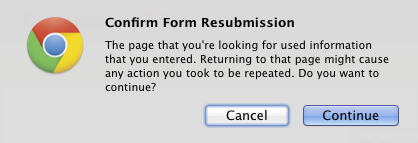
Q1. When you try to access a resource that requires logging in, like LinkedIn feed, <https://www.linkedin.com/feed>, you get redirected to the login screen. That should be a 301, right? Can you verify.

You’ll be able to see, that was a 302 instead of 301. While 301 denotes a permanent redirect, 302 says the requested resource is temporarily unavailable. Here, 302 is used as the requested resource was found, there just is another page to go through (Login page) before it can be accessed. More [here](https://stackoverflow.com/questions/2839585/what-is-correct-http-status-code-when-redirecting-to-a-login-page)

Q2 . One day or another, you’d have come across the below pop-up when trying to reload a web page containing a form. Why does this happen? Is there any way to avoid this happening?

The Post/Redirect/Get (PRG) Pattern





To avoid this usability issue, you want to try to keep POST events out of the browser history. Conveniently, there there is a mechanism for this that all the browsers respect. If a HTTP POST returns a HTTP 302 redirect, only the location of the redirect will be stored in the browser history. Hitting the back button will skip over the POST, and the user can bounce freely between the first and second forms.

Find out example situations that result in a 4xx or 5xx response code.

Refer - [4xx](https://www.websitepulse.com/kb/4xx_http_status_codes) - [5xx](https://www.websitepulse.com/kb/5xx_http_status_codes)

Interview Corner

What is HTTP? Why was it introduced?

What happens when you type google.com in a browser?

What are the main HTTP verbs/methods?

What are the different HTTP response codes?

What is the difference between HTTP and HTTPS?

What is cURL?

What is Postman used for?