Context

Today, we'll understand about HTTP Servers

More specifically

1. Intro to the HTTP Protocol, what does it solve
    1. Exploring the network tab in the chrome developer tools
    2. Request Response model
2. Diving into some HTTP Constructs
    1. Domain name/IP
    2. Port
    3. Methods
    4. Plaintext vs JSON vs HTML response
    5. Status codes
    6. Body, Headers
    7. Routes
3. Installing Postman and playing with it
4. Trying to code an in memory todo app
5. Assignment - Trying to code a filesystem based todo app

Why the HTTP Protocol?

Back in the day, HTTP was introduced so machines all around the world could talk to each other.

This would be useful for things like

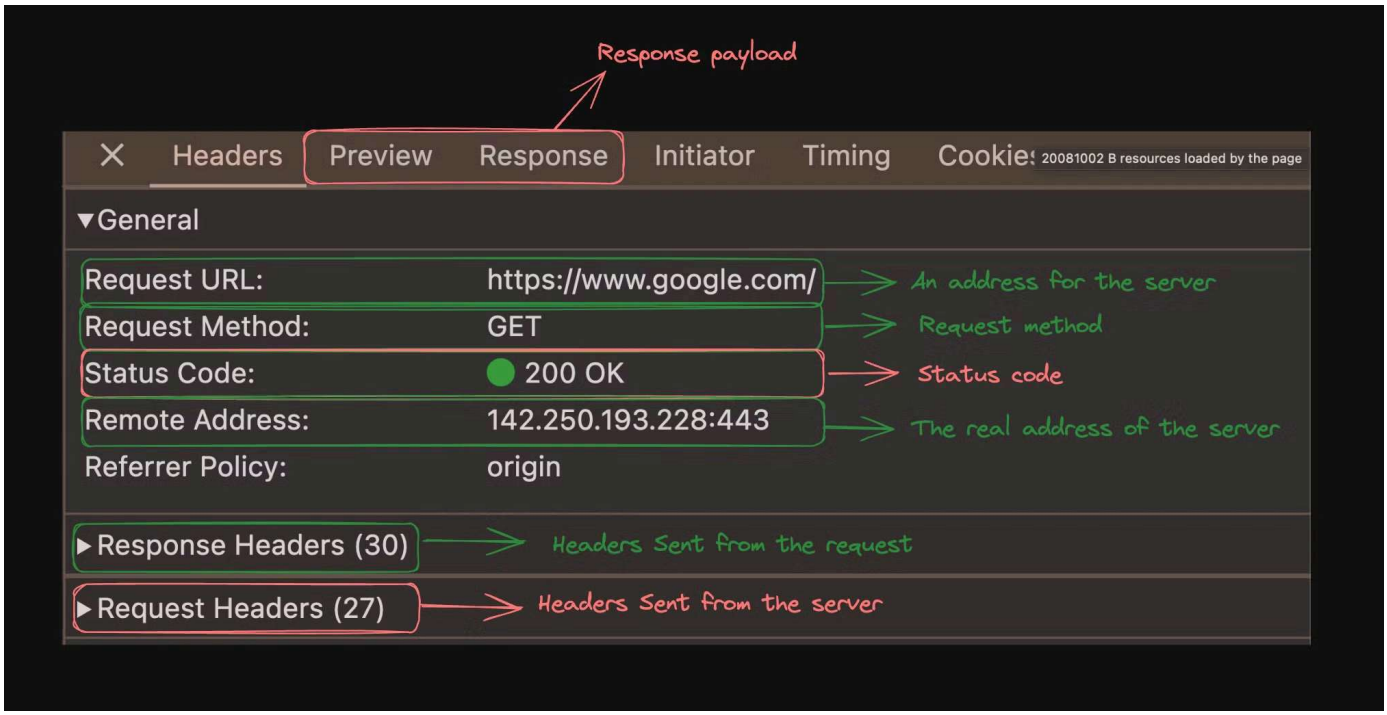
1. Talking via im (instant messenger)
2. Emails

available on a very big machine at

y ol was formalised and now spec'd out here - https://datatracker.ietf.org/doc/html/rfc2616

Mini assignments

Try exploring the network tab and seeing all the HTTP requests that go out when you visit https://google.com



Request Response model

Request response model

e model is a fundamental communication pattern.

It describes how data is exchanged between a client and a server or between two systems.

Are there other ways for you to communicate b/w machines?

Yes, there are various other protocols that exist that let machines communicate with each other.

1. Websockets
2. WebRTC
3. GRPC

...

Domain name/IP

Domain names

The way to reach a sever is through its Domain name . For example

1. google.com
2. app.100xdevs.com
3. x.com

IPs

Every domain that you see, actually has an underlying IP that it resolves to.

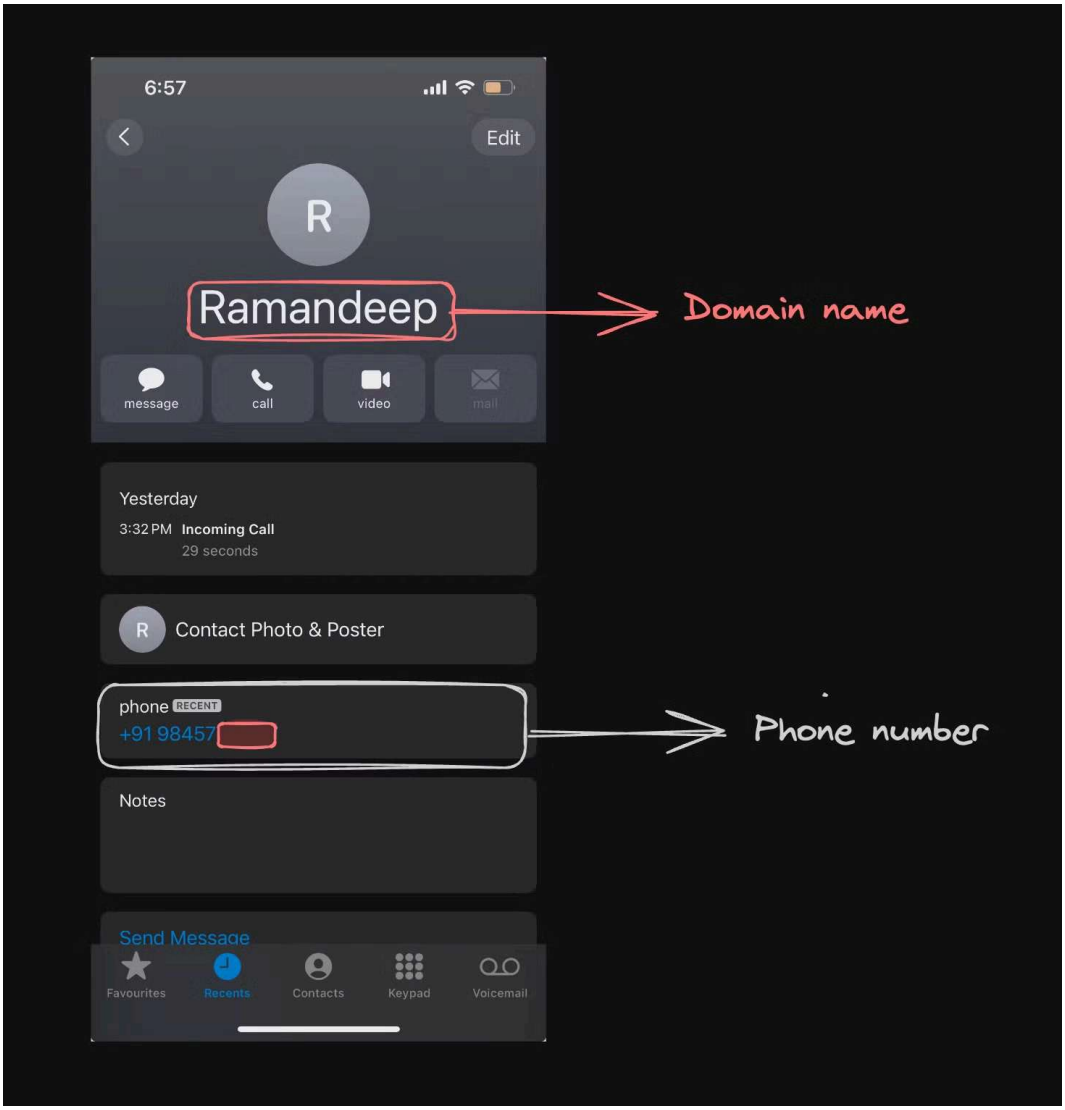
You can check the ip by running the ping command.

```
ping google.com
```

When you try to visit a website, you are actually visiting the underlying IP address .

Domain name - Phone contact

h Intro to HTTP 1 of 14 umber

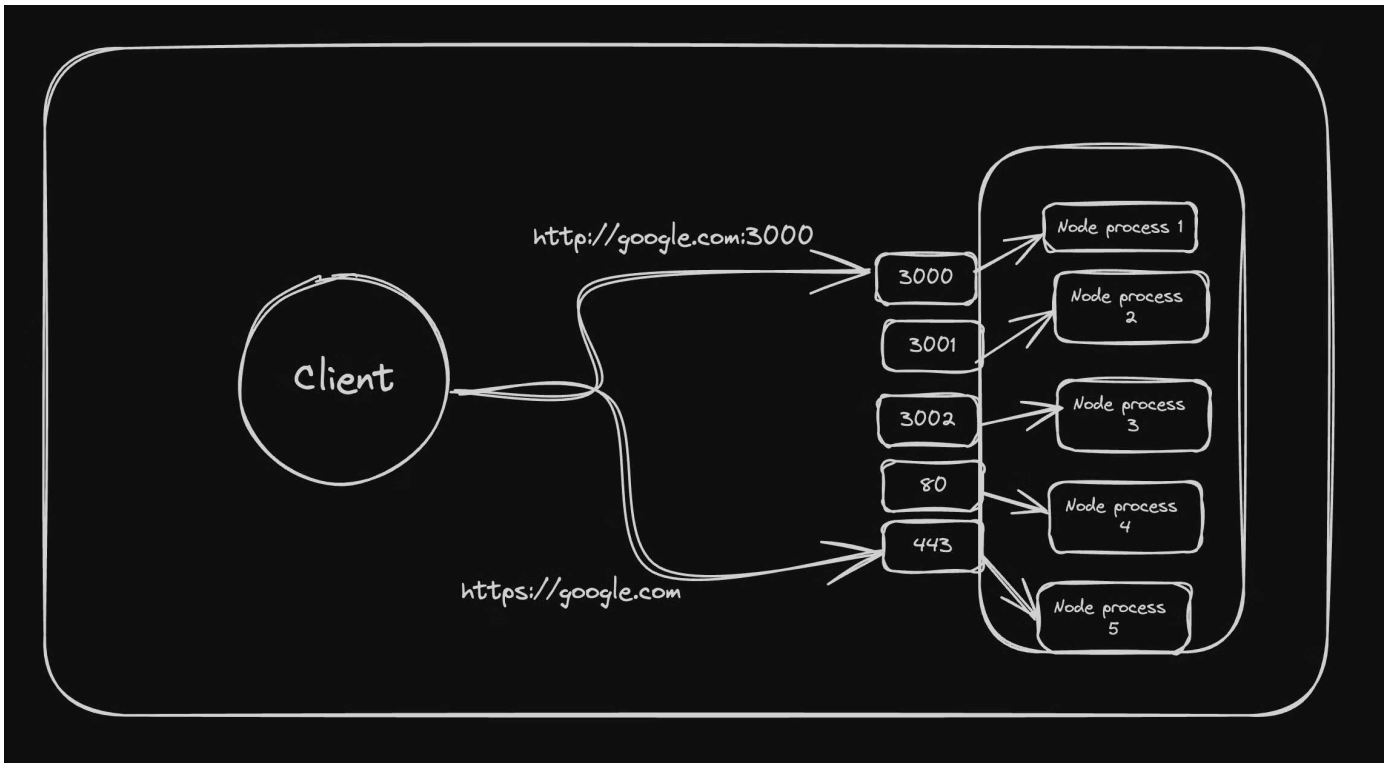


Ports

nts used by protocols to identify

specific processes running on a computer or server. They help direct

network traffic to the correct application or service on a system.



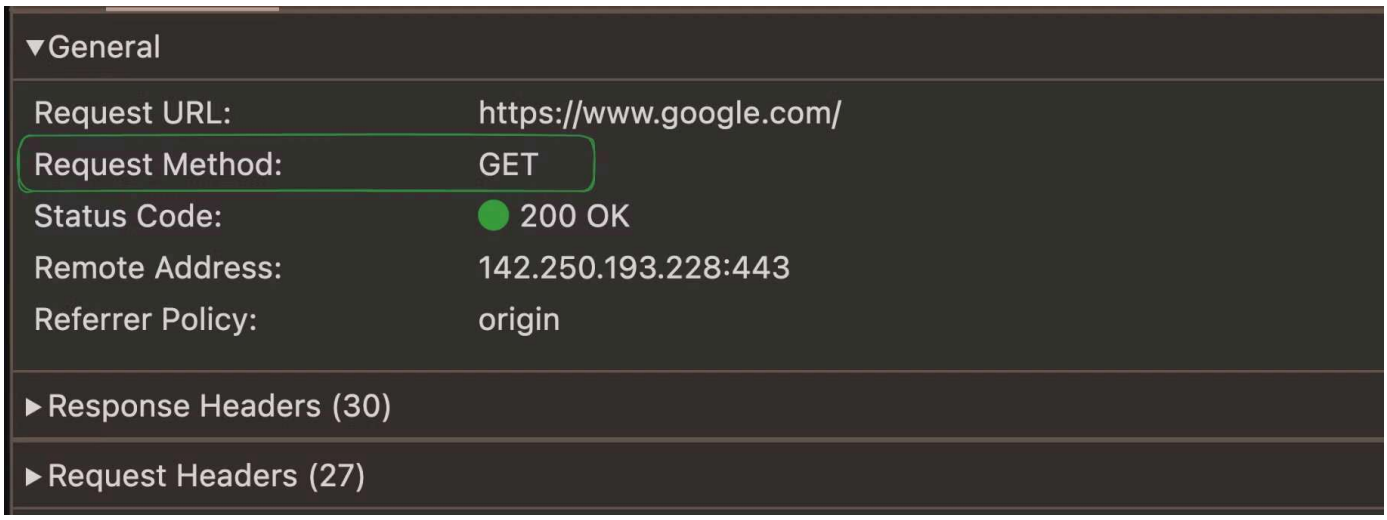
Methods

HTTP methods are used to specify the type of action that the client wants
r Intro to HTTP 1 of 14 ce on the server.

You done NEED to use all the methods, but you always should. You can do everything you want with a GET or POST method, but it is usually advisable to use them right.

Common methods

1. GET - Retrieve data from a server. (Get my TODOS)
2. POST - Submit data to be processed by a server. (Create a TODO)
3. PUT - Update or create a resource on the server (Update my todo)
4. DELETE - Remove a resource from the server. (Delete my todo)



Response

The response represents what the server returns you in response to the request.

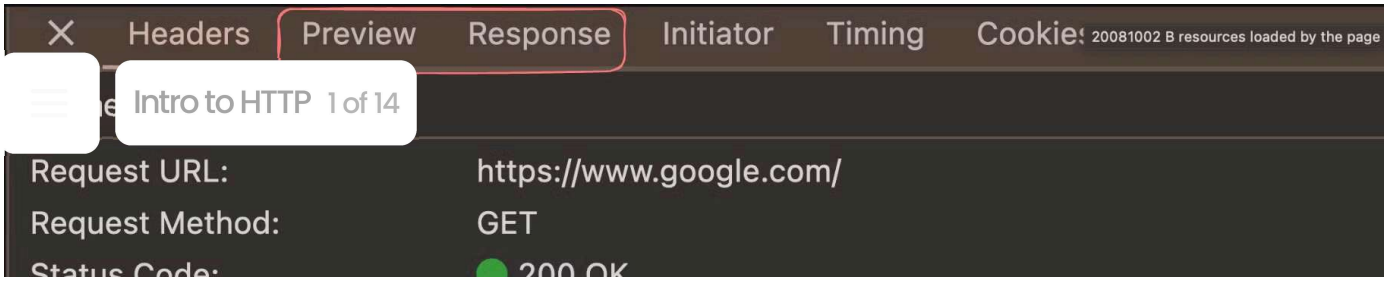
It could be

1. Plaintext data - Not used as often
2. HTML - If it is a website
3. JSON Data - If you want to fetch some data (user details, list of todos...)

JSON

JSON stands for **JavaScript Object Notation**. It is a lightweight, text-based format used for data interchange

```
{
  "name": "John Doe",
  "age": 30,
  "isEmployed": true,
  "address": {
    "street": "123 Main St",
    "city": "Anytown"
  },
  "phoneNumbers": ["123-456-7890", "987-654-3210"]
}
```



Status codes

HTTP status codes are three-digit numbers returned by a server to indicate the outcome of a client's request. They provide information about the status of the request and the server's response.

200 series (Success)

- **200 OK**: The request was successful, and the server returned the requested resource.
- **204 No Content**: The request was successful, but there is no content to send in the response

300 series (Redirection)

- **301 Moved Permanently**: The requested resource has been moved to a new URL permanently. The client should use the new URL provided in the response.
- **304 Not Modified**: The resource has not been modified since the last request. The client can use the cached version.

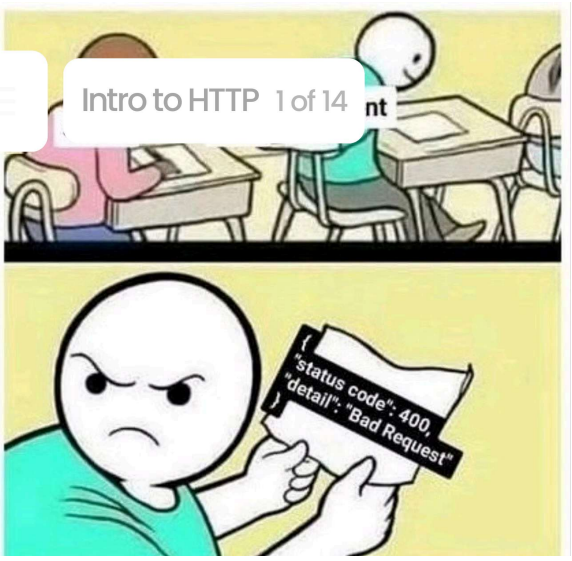
400 series (Client Error)

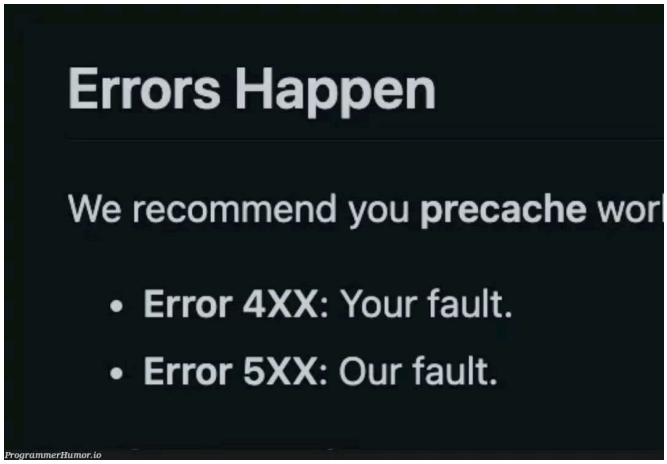
- **400 Bad Request**: The server could not understand the request due to

- **401 Unauthorized**: The request requires user authentication. The client must provide credentials.
- **403 Forbidden**: The server understood the request but refuses to authorize it.
- **404 Not Found**: The requested resource could not be found on the server.

500 series (Server Error)

- **500 Internal Server Error**: The server encountered an unexpected condition that prevented it from fulfilling the request.
- **502 Bad Gateway**: The server received an invalid response from an upstream server while acting as a gateway or proxy.





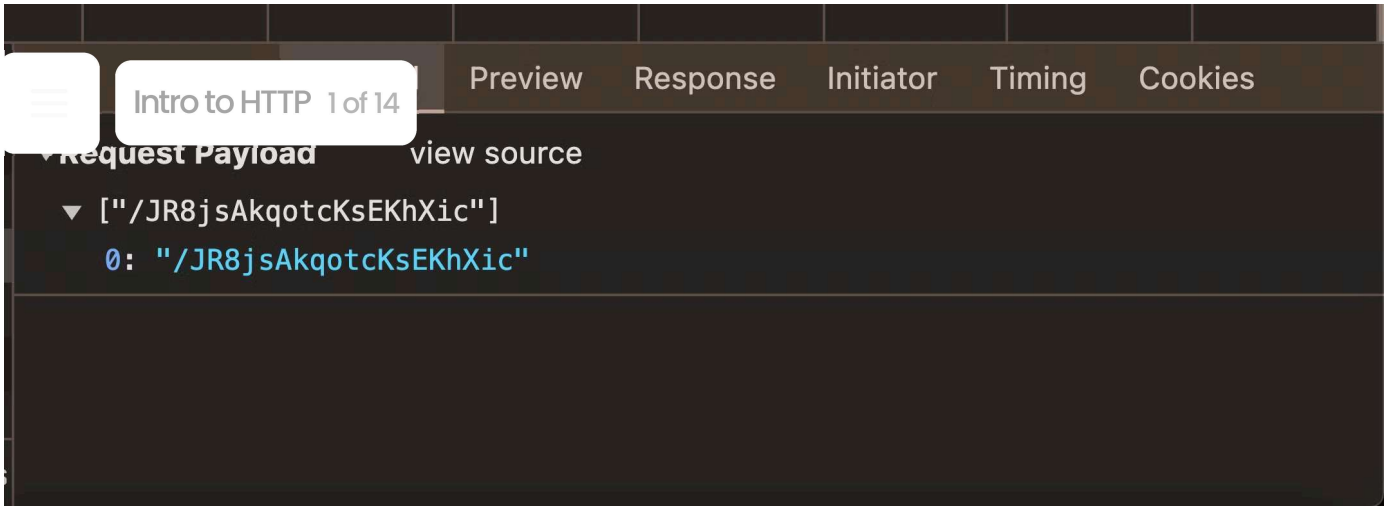
Body

In HTTP communications, the **body** (or **payload**) refers to the part of an HTTP message that contains the actual data being sent to the server.

It is usually JSON data that is transferred to the server.

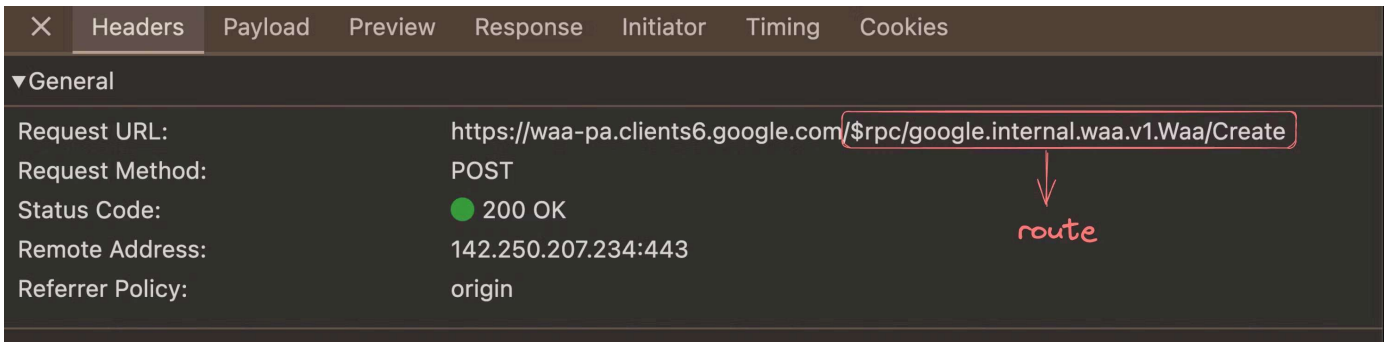
For example -

```
{
  todo: "Go to the gym"
}
```



Routes

In the context of HTTP, **routes** are paths or endpoints that define how incoming requests are handled by a server. Routing is a mechanism used to direct incoming HTTP requests to the appropriate handler functions or resources based on the URL path.



Headers

HTTP headers are key-value pairs included in HTTP requests and responses that provide metadata about the message.

Why not use body?

Even though you can use body for everything, it is a good idea to use headers for sending data that isn't directly related with the application logic .

For example, if you want to create a new TODO, you will send the TODO payload in the body

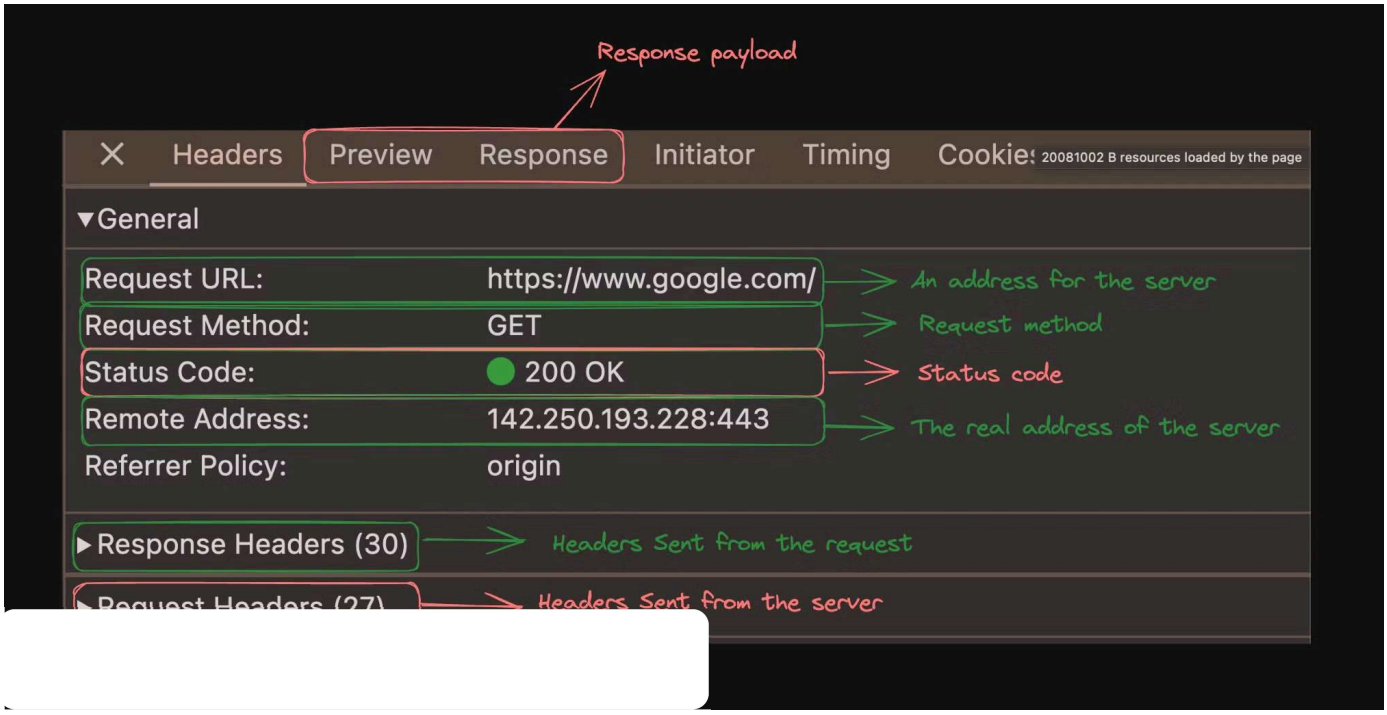
```
{
  description: "Go to gym"
}
```

But the Authorization information in the headers

Authorization: barkirqt

Intro to HTTP 1 of 14

Do we understand the following now?



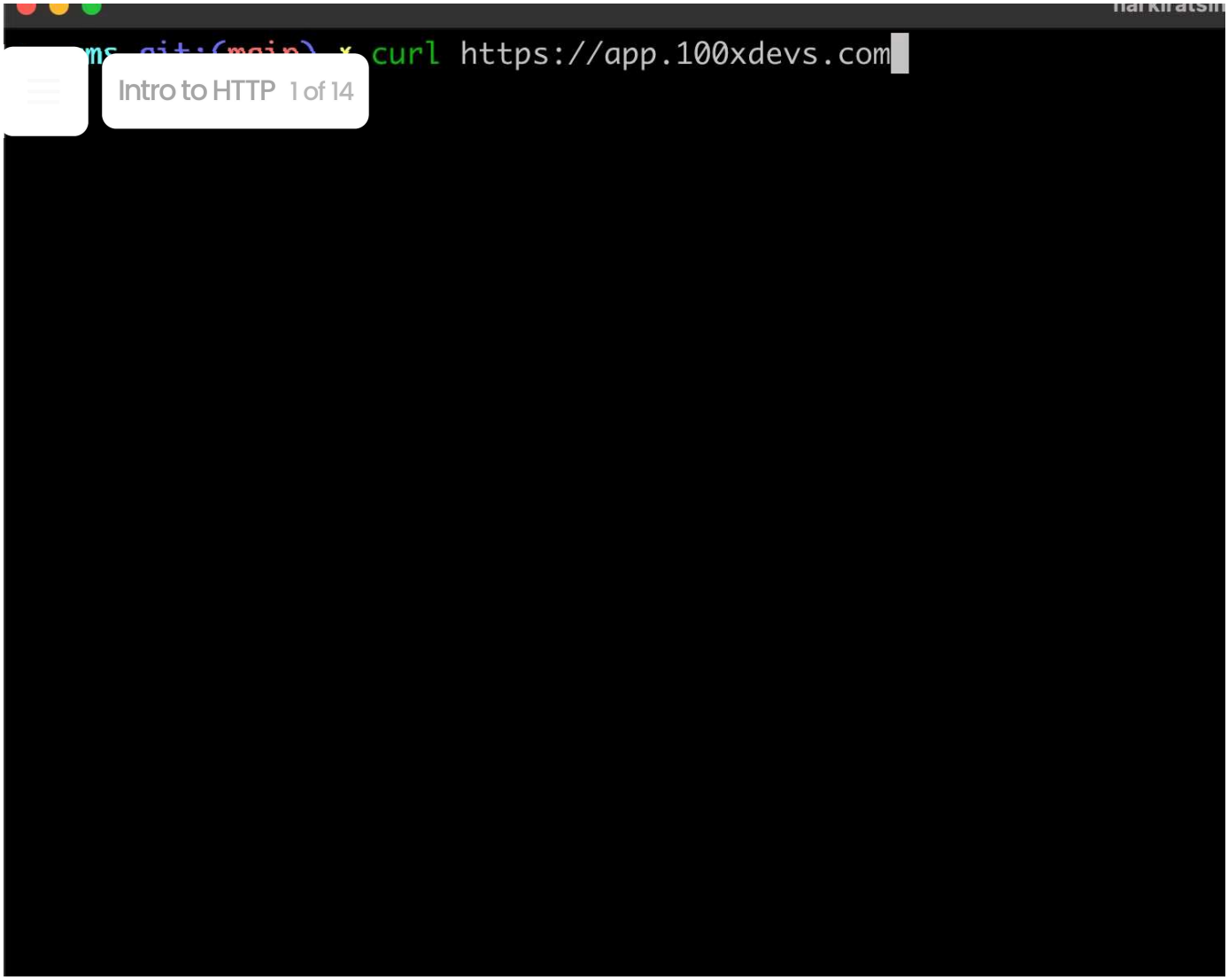
Clients (Postman/curl/browser)

Postman lets you send HTTP requests to a server, just like your browser. It gives you a prettier interface to send requests and play with them.

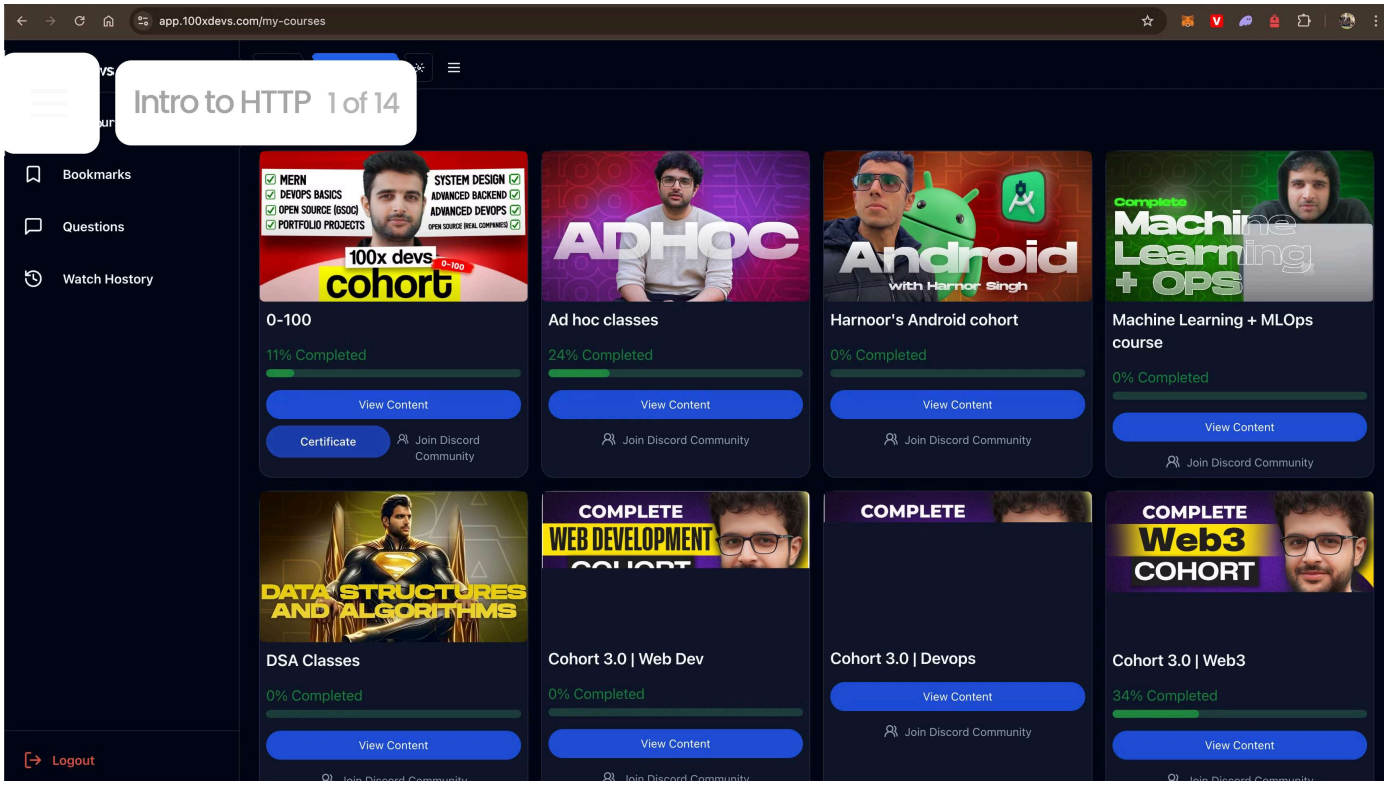
You can send a request from various clients , Postman being one of them.

Installing postman - https://www.postman.com/downloads/

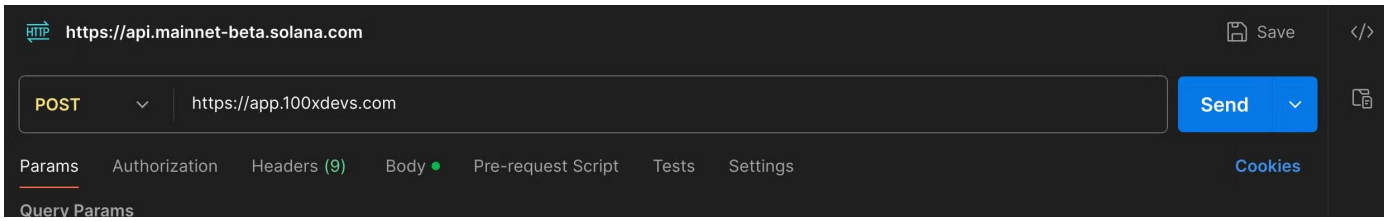
curl



Browser



Postman



Writing HTTP code in js

https://www.npmjs.com/package/express

Intro to HTTP 1 of 14