



# 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 Intro to HTTP 1 of 14 oI 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>

The screenshot shows the Network tab of a browser's developer tools. The 'Response' tab is selected. The response details are as follows:

- Request URL: https://www.google.com/ → An address for the server
- Request Method: GET → Request method
- Status Code: 200 OK → Status code
- Remote Address: 142.250.193.228:443 → The real address of the server
- Referrer Policy: origin

Below the main details, there are sections for 'Response Headers (30)' and 'Request Headers (27)', each with a green arrow pointing to their respective descriptions.

## Request Response model

## Request response model

☰ ∈ Intro to HTTP 1 of 14 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

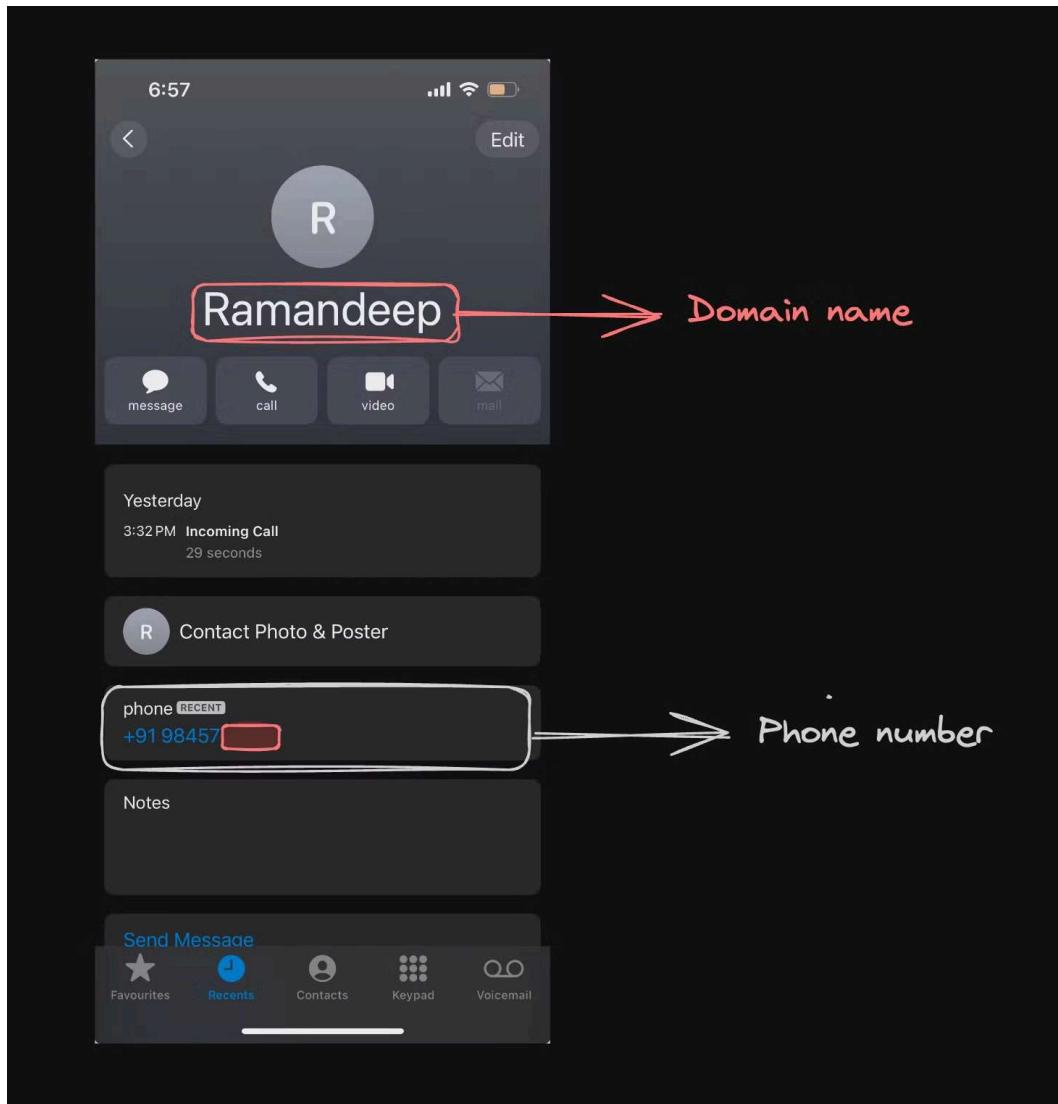


```
→ cms git:(main) ✘ ping google.com
PING google.com (142.250.182.46): 56 data bytes
64 bytes from 142.250.182.46: icmp_seq=0 ttl=59 time=10.008 ms
^C
```

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

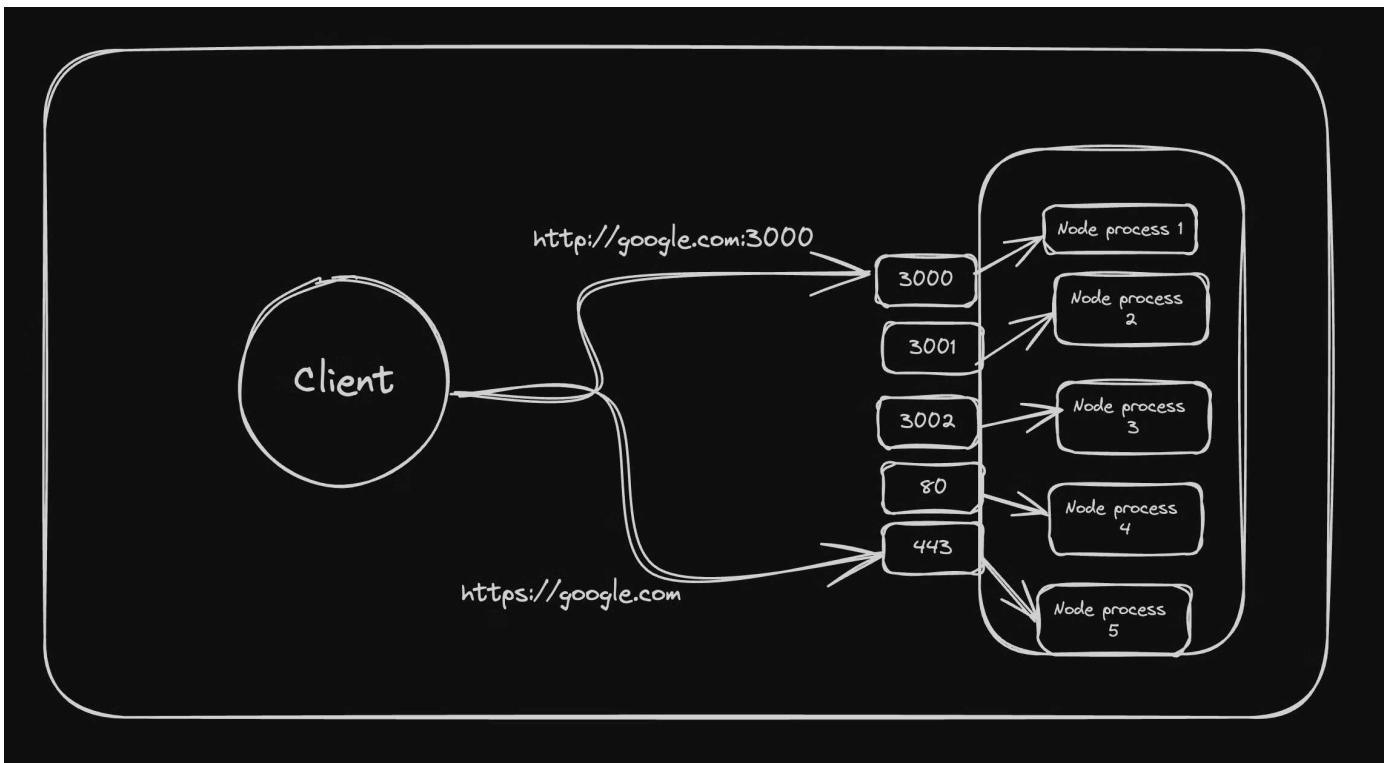


# Ports

Ports are numbers 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.

Intro to HTTP 1 of 14



# Methods

HTTP methods are used to specify the type of action that the client wants

☰ ▶ Intro to HTTP 1 of 14 See on the server.



You don't 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)

▼ General

Request URL:	<a href="https://www.google.com/">https://www.google.com/</a>
Request Method:	GET
Status Code:	200 OK
Remote Address:	142.250.193.228:443
Referrer Policy:	origin

► Response Headers (30)

► Request Headers (27)

# Response

Intro to HTTP 1 of 14

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"]  
}
```

The screenshot shows a network request for 'Intro to HTTP' (1 of 14). The request URL is <https://www.google.com/>, the method is GET, and the status code is 200 OK. A cookie for '20081002' is listed.

# 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.

100 series (Client Error)

- **400 Bad Request:** The server could not understand the request due to
  - v Intro to HTTP 1 of 14
- **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.

The screenshot shows a browser developer tools Network tab interface. At the top, there are tabs: X, Headers, Preview, Response, Initiator, Timing, and Cookies. The Headers tab is selected. Below the tabs, there's a section titled 'General' with the following details:

- Request URL: <https://www.google.com/>
- Request Method: GET
- Status Code: 200 OK (highlighted with a red border)
- Remote Address: 142.250.193.228:443
- Referrer Policy: origin

Below this, there are sections for 'Response Headers (30)' and 'Request Headers (27)'. The entire interface has a dark theme.



## Errors Happen

We recommend you **precache** wor

- **Error 4XX:** Your fault.
- **Error 5XX:** Our fault.

ProgrammerHumor.io

# 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"  
}
```

The screenshot shows a browser developer tools Network tab. The title bar includes tabs for Preview, Response, Initiator, Timing, and Cookies. A sub-menu for 'Request Payload' is open, showing the URL '/JR8jsAkqotcKsEKhXic'.

# 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.

The screenshot shows a browser developer tools Network tab. The title bar includes tabs for Headers, Payload, Preview, Response, Initiator, Timing, and Cookies. The Headers tab is selected. The Request URL is highlighted with a red box and labeled 'route' with a pink arrow pointing to it.

Request URL:	https://waa-pa.clients6.google.com/\$rpc/google.internal.waa.v1.Waa/Create
Request Method:	POST
Status Code:	200 OK
Remote Address:	142.250.207.234:443
Referrer Policy:	origin



# 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: barkingt

Intro to HTTP 1 of 14



# Do we understand the following now?

The screenshot shows a browser developer tools Network tab for a request to `https://www.google.com/`. The Response tab is active. The response details are as follows:

- Request URL:** `https://www.google.com/` → An address for the server
- Request Method:** GET → Request method
- Status Code:** 200 OK → Status code
- Remote Address:** 142.250.193.228:443 → The real address of the server
- Referrer Policy:** origin

**Response Headers (30)** → Headers Sent from the request

**Request Headers (27)** → Headers Sent from the server

A callout points to the "Response payload" section.

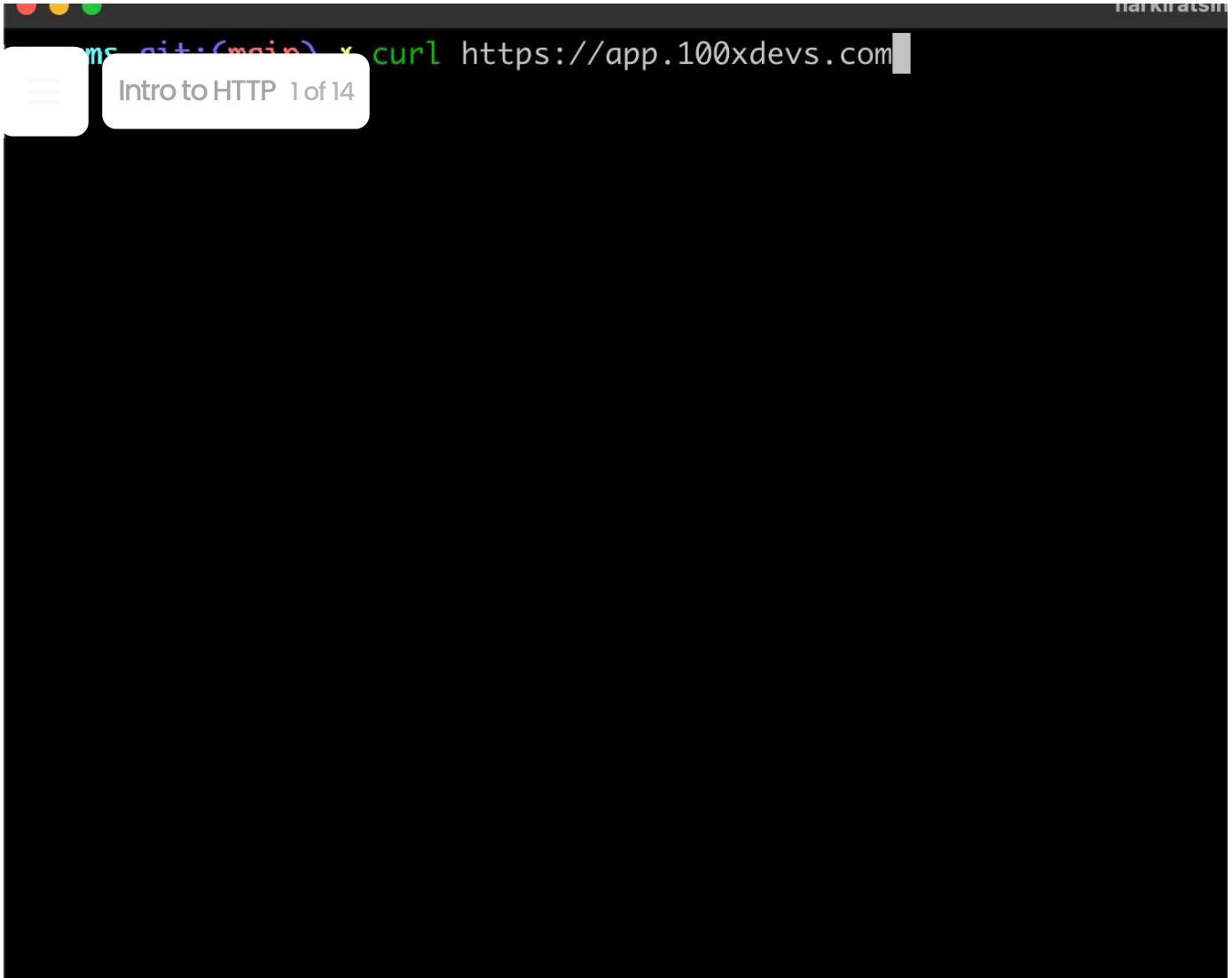
# Clients (Postman/curl/browser)

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



A screenshot of a terminal window titled "curl https://app.100xdevs.com". The window shows the command "curl https://app.100xdevs.com" in green text at the top. Below it, a large white text area displays the message "Intro to HTTP 1 of 14". The background of the terminal is black.

## Browser

The screenshot shows a grid of course cards on the 100xDevs website. The cards include:

- 0-100**: 11% Completed. Includes a 'View Content' button and a 'Certificate' button.
- ADHOC**: 24% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.
- Harnoor's Android cohort**: 0% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.
- Machine Learning + OPS**: 0% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.
- DATA STRUCTURES AND ALGORITHMS**: 0% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.
- COMPLETE WEB DEVELOPMENT COHORT**: 0% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.
- Cohort 3.0 | Web Dev**: 0% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.
- Cohort 3.0 | Devops**: 0% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.
- COMPLETE Web3 COHORT**: 34% Completed. Includes a 'View Content' button and a 'Join Discord Community' button.

On the left sidebar, there are links for Bookmarks, Questions, and Watch History. At the bottom left is a 'Logout' button.

## Postman

The screenshot shows the Postman interface with the following details:

- HTTP Method: POST
- URL: <https://app.100xdevs.com>
- Headers: (9)
- Body: (Text)
- Pre-request Script: (None)
- Tests: (None)
- Settings: (None)
- Cookies: (None)

# Writing HTTP code in js

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

