

EXPRESS AND BACKEND

What is an HTTP Server?

realdevelopertillplacement@gmail.com
8920597118

HTTP

Hyper text transfer Protocol

1. A protocol that is defined for machines to communicate
2. Specifically for websites, it is the most common way for your website's frontend to talk to its backend

What is an HTTP Server?

Frontend/Clients (HTML/CSS/JS)



India

Backends (Node.js)



California

How do frontends talk to backends - Wires/routers



What is an HTTP Server?



realdevelopertillplacement@gmail.com

8920597118

Some code that follows the HTTP Protocol
And is able to communicate with clients (browsers/mobile apps...)

Think of it to be similar to the call app in your phone
Which lets you communicate with your friends



What is an HTTP protocol?

HTTP stands for Hypertext Transfer Protocol. It's the underlying protocol used by the World Wide Web that defines how messages are formatted and transmitted, and how web servers and browsers should respond to various commands.

HTTP operates as a request-response protocol in the client-server computing model. This means that a client (typically a web browser) sends a request to a server, and the server responds with the requested resource, such as a web page, image, or other content.

Key features of HTTP include:

1. **Statelessness:** Each request from a client to a server must contain all the information necessary for the server to understand and fulfill the request. The server does not maintain any information about the state of the client.
2. **Connectionless:** Each request-response cycle operates independently of previous cycles. After the server sends a response to the client, the connection can be closed, and the client and server do not maintain a continuous connection.
3. **Text-based protocol:** HTTP messages are typically text-based, making them human-readable and easy to understand. Requests and responses consist of headers that provide metadata about the message, followed by an optional message body containing data.

HTTP has undergone several revisions over the years, with HTTP/1.1 being the most widely used version for many years. Recently, HTTP/2 and HTTP/3 have been

developed to address performance and security issues and to support modern web applications.

HTTP Protocol

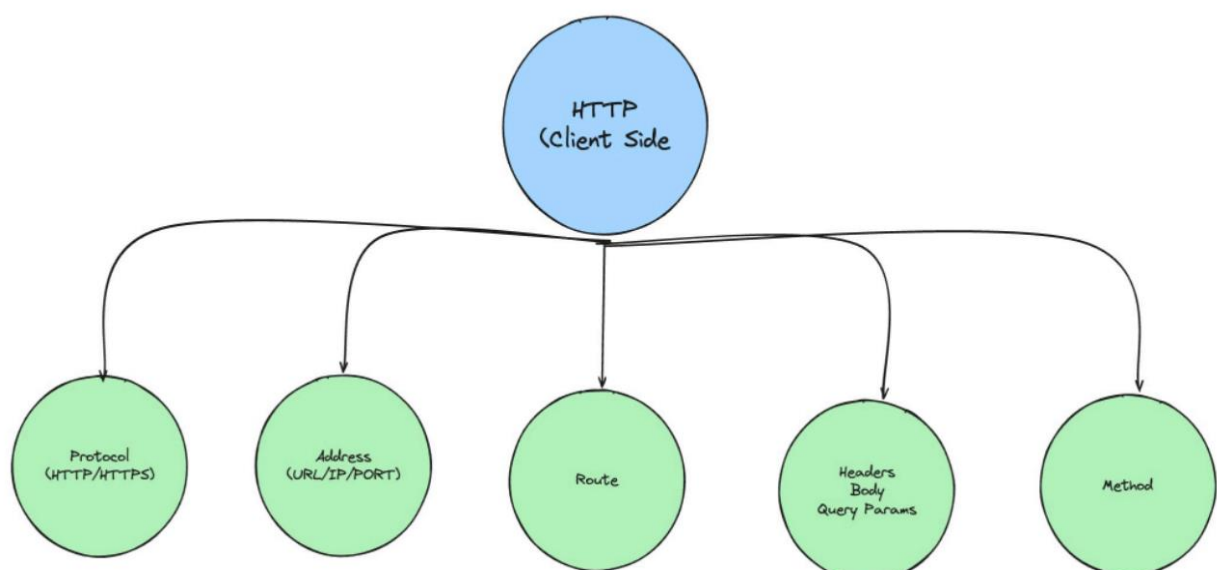
In the end, its the client throwing some information at a server
Server doing something with that information
Server responding back with the final result

Think of them as functions, where

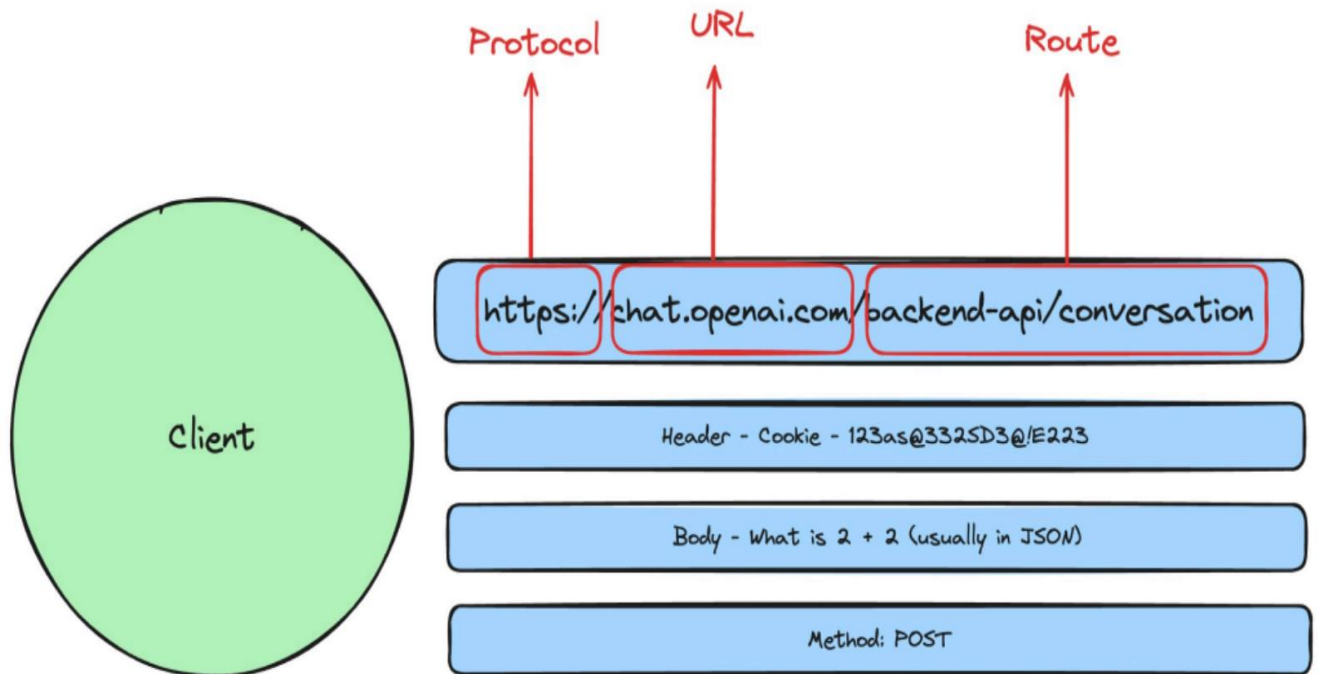
1. Arguments are something the client sends
2. Rather than calling a function using its name, the client uses a URL
3. Rather than the function body, the server does something with the request
4. Rather than the function returning a value, the server responds with some data

When we are making an HTTP Request, then the following things we need to provide in our request

Things client needs to worry about



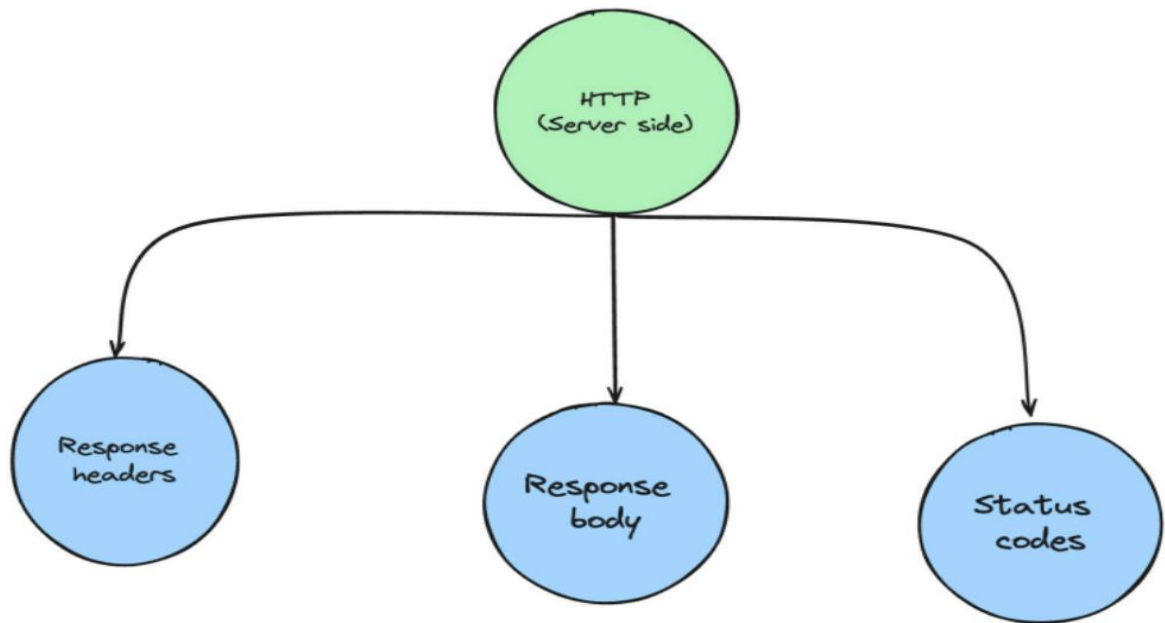
Basically, this happens when you make the request at chat gpt:-



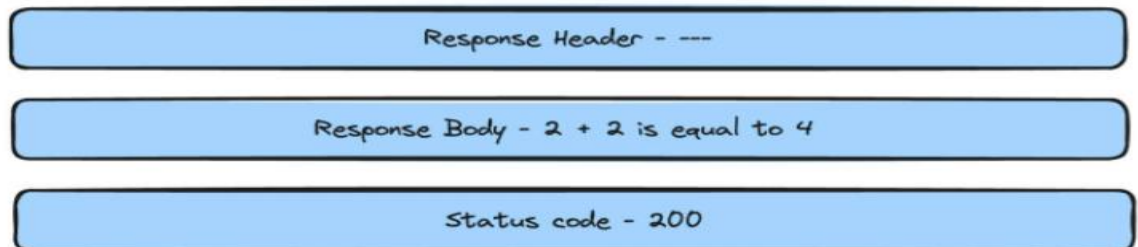
URL tells where we are making the request,
Route determines what exactly we want.

Others like Method, Headers, and body
contain other data.

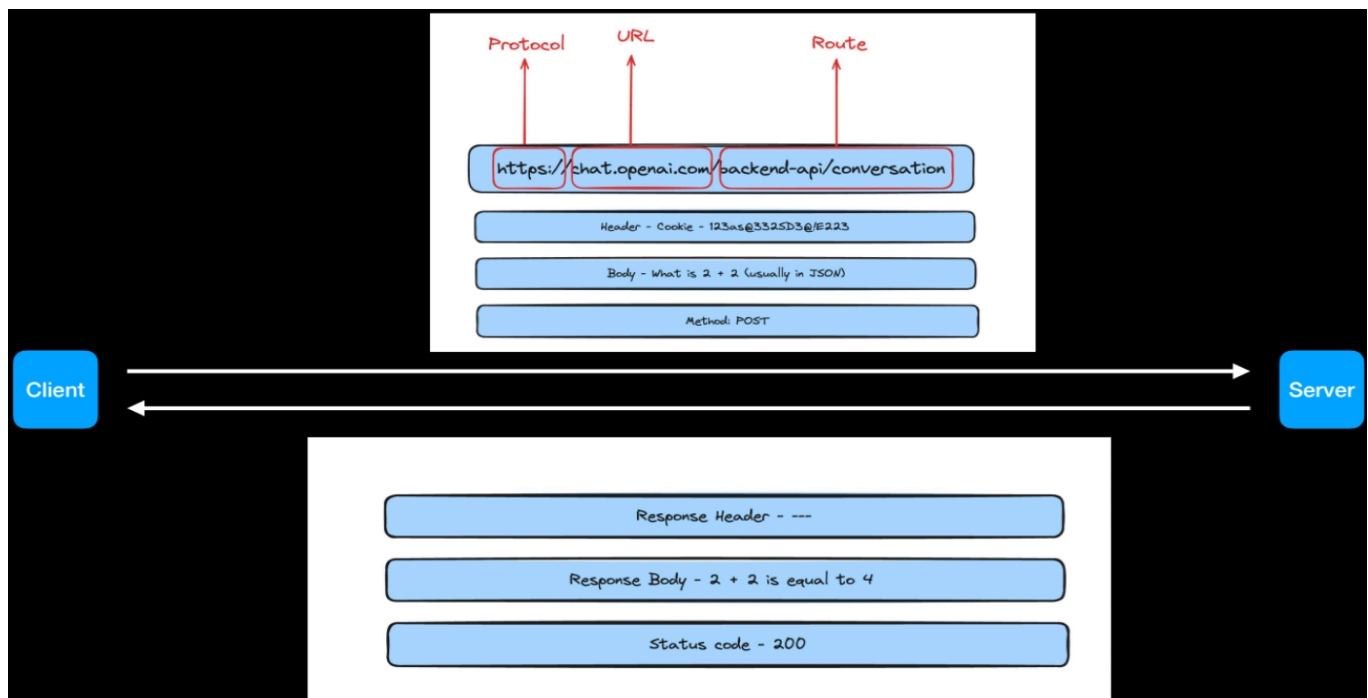
Things server needs to worry about



This is what responded by the server.



But overall this is going on :-



Now we will study what happens when we fire a request:-

Things that happen in your browser after you fire this request (we will get to how to fire request to a backend server later)

1. Browser parses the URL
2. Does a DNS Lookup (converts google.com to an IP)
3. Establishes a connection to the IP (does handshake...)

What is DNS resolution
URLs are just like contacts in your phone
In the end, they map to an IP
If you ever buy a URL of your own, you will need to point it to the IP of your server

Client

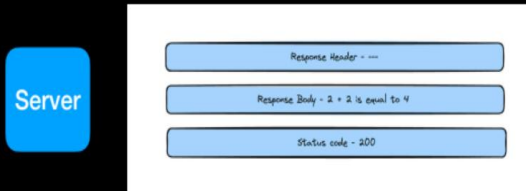
The diagram shows the structure of an HTTP request:

- URL:** `https://chat.openai.com/backend-api/conversation` (labeled with Protocol, URL, and Route).
- Header:** `Cookie - 123as@332SD3@E223`
- Body:** `What is 2 + 2 (usually in JSON)`
- Method:** `POST`

Browser parses the URL means it analyses the URL and determines where the Request is

being made, and on what Route. DNS converts the URL into an IP address, An IP address is a way to find a server on the Internet.

Things that happen on your server after the request is received



The diagram shows a blue box labeled 'Server' on the left. To its right is a white box containing three stacked blue rounded rectangles. The top rectangle is labeled 'Response Header - ...'. The middle rectangle is labeled 'Response Body - 2 * 2 is equal to 4'. The bottom rectangle is labeled 'Status code - 200'.

1. You get the inputs (route, body, headers)
2. You do some logic on the input, calculate the output
3. You return the output body, headers and status code

A simple HTTP server:-

Lets create a simple HTTP Server

```
1  const express = require('express' 4.18.2 )
2  const app = express()
3  const port = 3000
4
5  app.get('/', (req, res) => {
6    res.send('Hello World!')
7  })
8
9  app.listen(port, () => {
10    console.log(`Example app listening on port ${port}`)
11  })
```

Whenever you start to create a server , write a command `npm init` , and this will create a `package.json` file which will store all your node modules related to server.