Node.js runtime | HTTP

In the backend, we mostly work on developing HTTP servers.

What are we learning today?

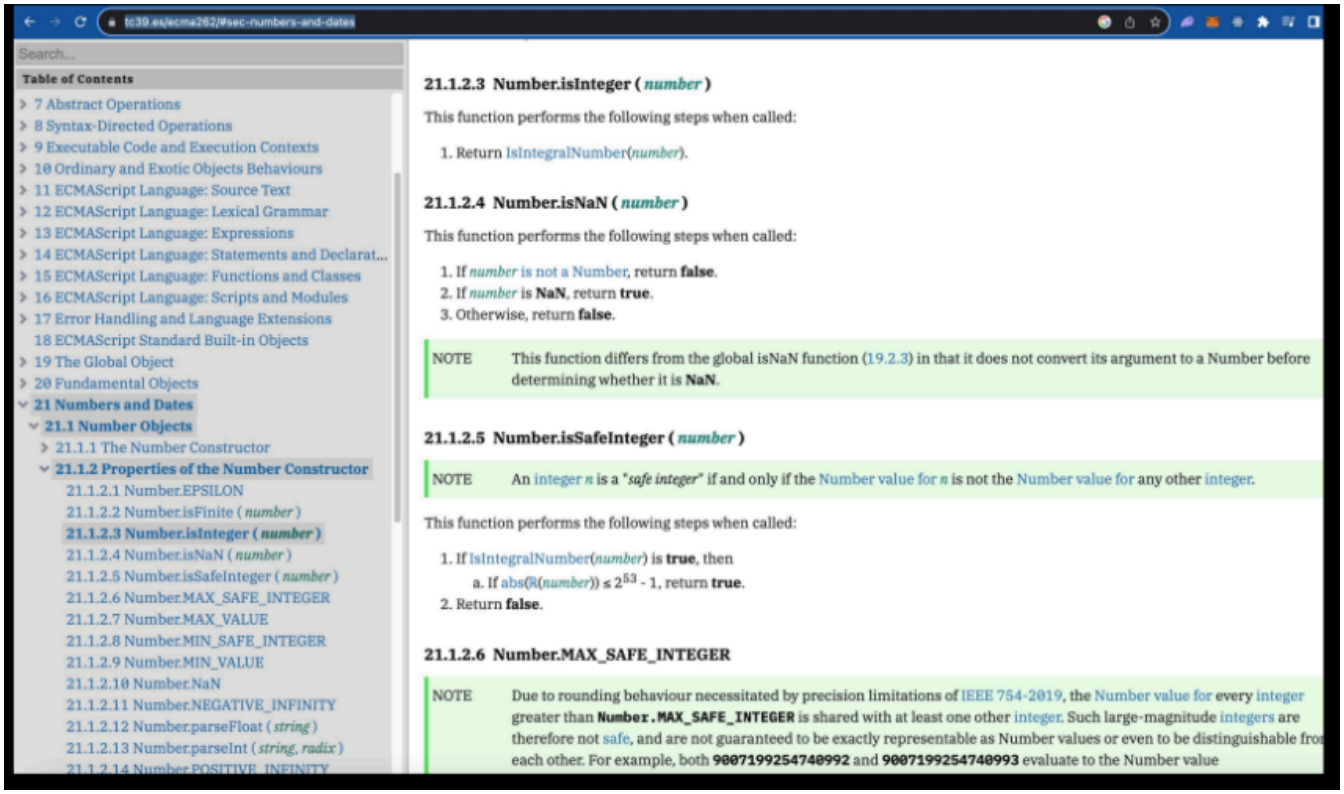
- Node.js and its runtime
- Backend communication protocol
- Express basics
- routes, status codes

What is ECMAScript?
What is Javascript?
What is Node.js?
What is Bun?

The difference between ECMAScript and Javascript and others are the same as the difference between Java and C++

ECMAScript

ECMAScript is a scripting language specification on which Javascript is based. This is just like docs whoever is designing the Javascript compiler/interpreter supports things like Numbers, String, etc.
This gets updated every year and the new version of Js should also support these new features

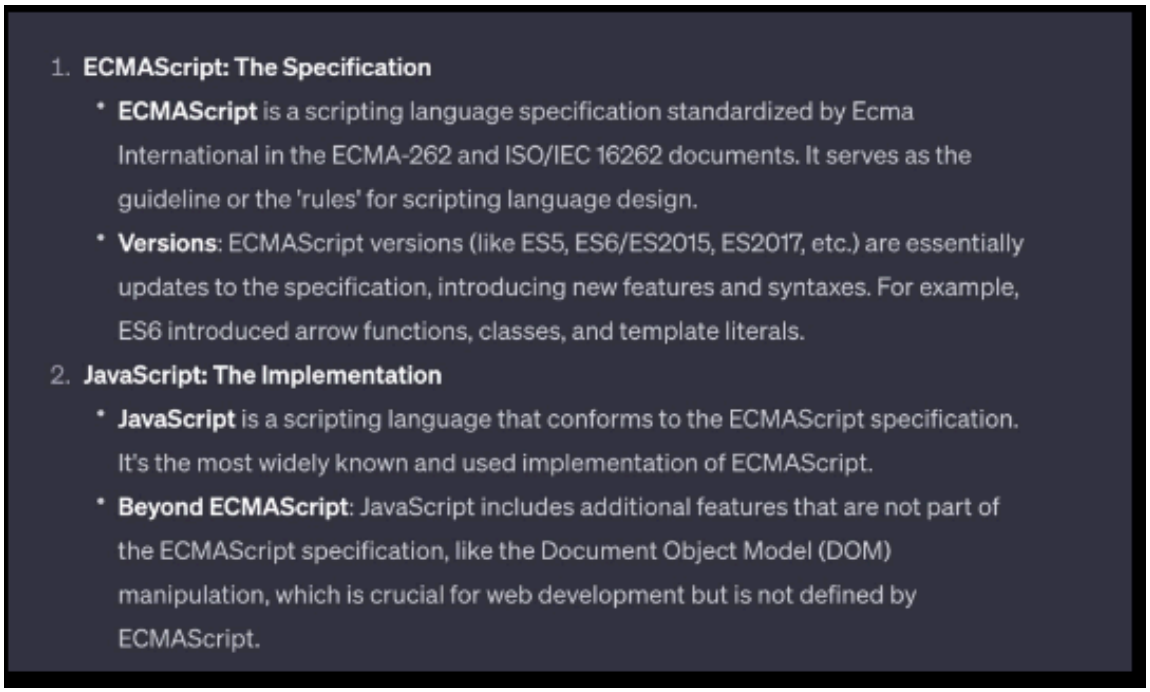


Javascript

Whoever is making the compiler/interpreter for JS follows ECMAScript
Eg Internet Explorer only supports ES5, if we try to write promise or arrow syntax then Internet Explorer will not be able to understand it.

Javascript is the implementation of ECMAScript
Example:
Date, var, const, let, function
It also includes additional features that are not part of the ECMAScript specification.
Examples:
setTimeout()

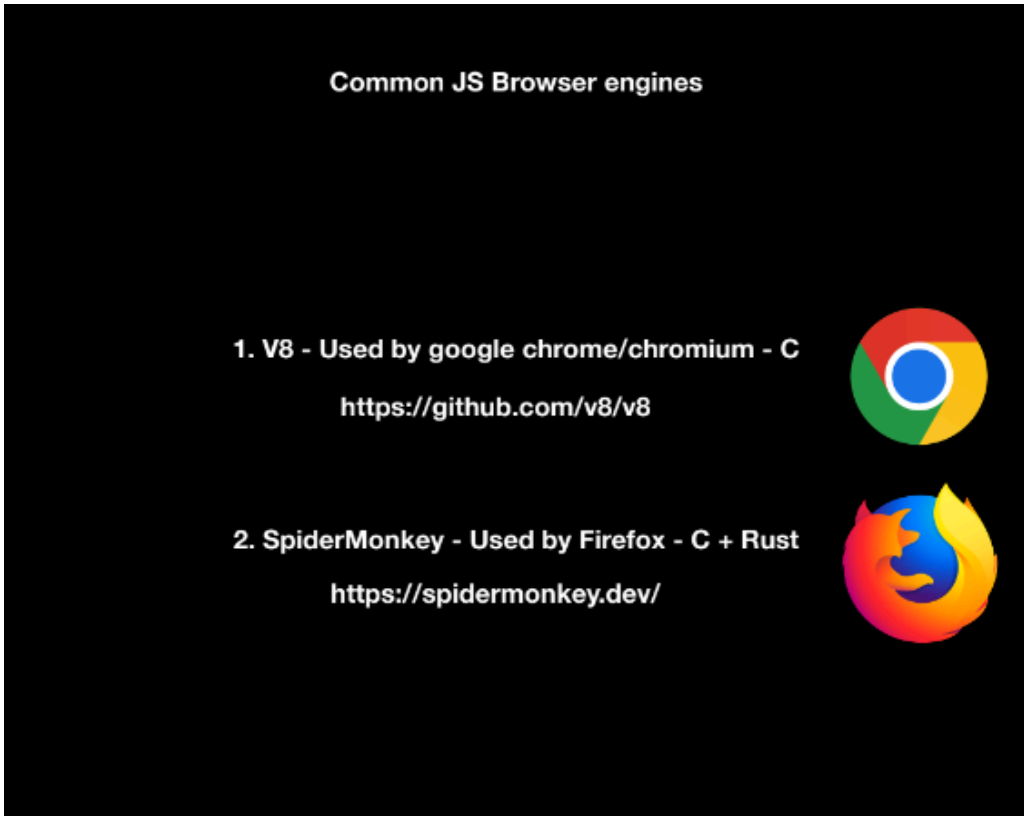
fs.readFile



1. **ECMAScript: The Specification**
   - **ECMAScript** is a scripting language specification standardized by Ecma International in the ECMA-262 and ISO/IEC 16262 documents. It serves as the guideline or the 'rules' for scripting language design.
   - **Versions**: ECMAScript versions (like ES5, ES6/ES2015, ES2017, etc.) are essentially updates to the specification, introducing new features and syntaxes. For example, ES6 introduced arrow functions, classes, and template literals.
2. **JavaScript: The Implementation**
   - **JavaScript** is a scripting language that conforms to the ECMAScript specification. It's the most widely known and used implementation of ECMAScript.
   - **Beyond ECMAScript**: JavaScript includes additional features that are not part of the ECMAScript specification, like the Document Object Model (DOM) manipulation, which is crucial for web development but is not defined by ECMAScript.

Some common JS Browser engines.

These are what compiles Javascript code into 0's and 1.



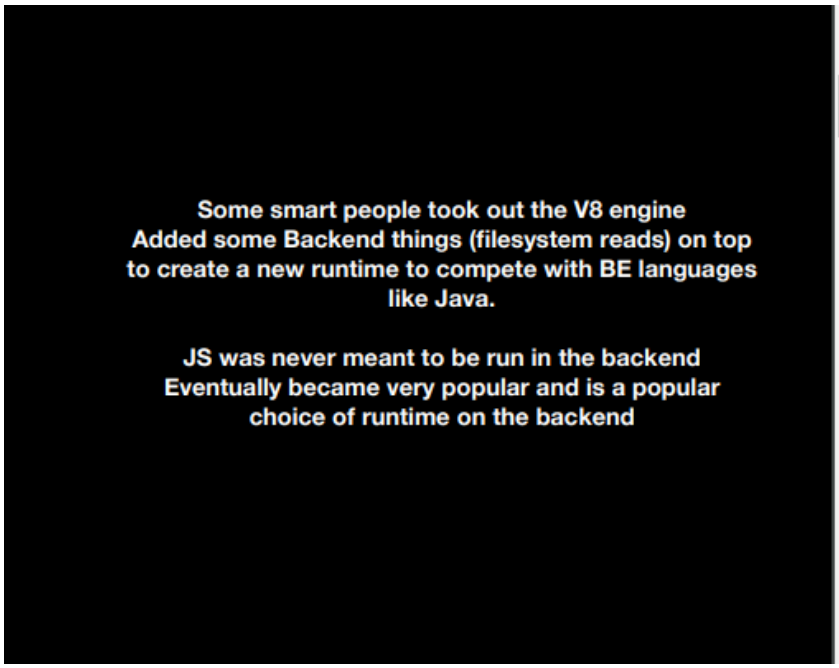
V8 Engine:

V8 is an open-source JavaScript engine developed by the Chromium project for Google Chrome and Chromium web browsers. It's written in C++ and compiles Javascript code into native machine code before executing it. Which greatly improves the performance.

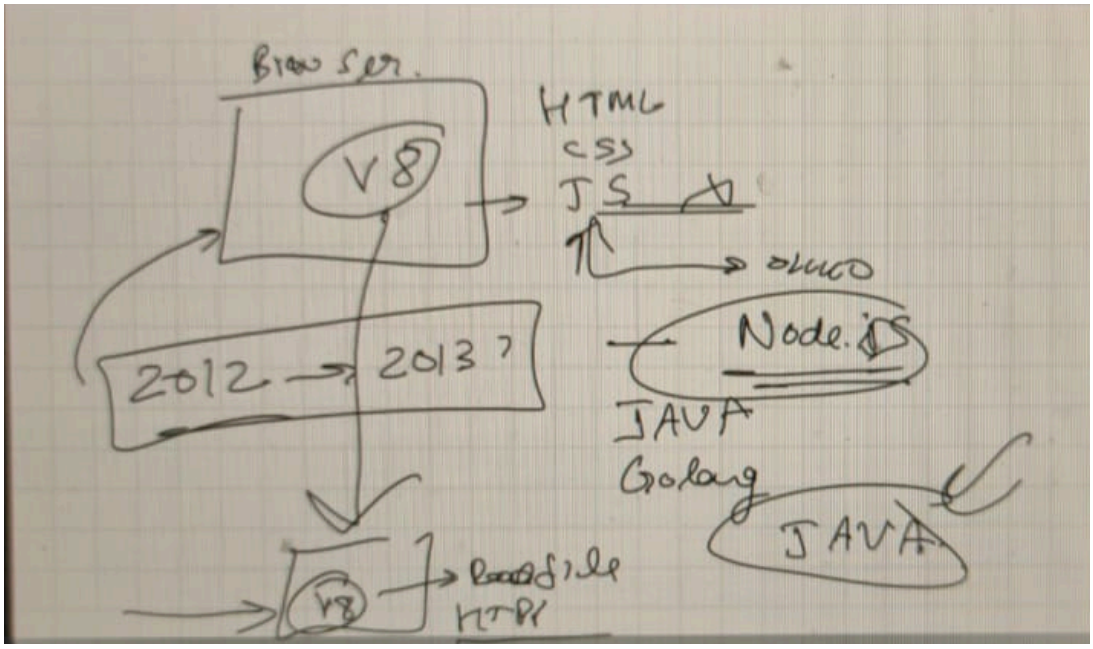
Whenever new versions of the ECMAScript come the developers of V8 and SpiderMonkey have to make sure to implement these new features in their Js Browser engines.

Node.js uses V8 Engine

Javascript primary goal was that whenever Browser opens, Javascript runs. It was never meant to work on Backend, languages like Golang and JAVA were used



Then in around 2012-2013, some smart people thought that we had a JS browser engine like V8 and JS was already used in Browsers, so they thought to pull out the V8 engine and add some functionality like readFile and HTTP to use Javascript for Backend also.



Now anyone can convert the JS code into 0,1s (basically run it) and also have some backend features.

This is how Node.js came into the picture

Node.js is written in C/C++

It became popular since

- Javascript is easy to understand
- We can hire engineers who know both frontend and backend.

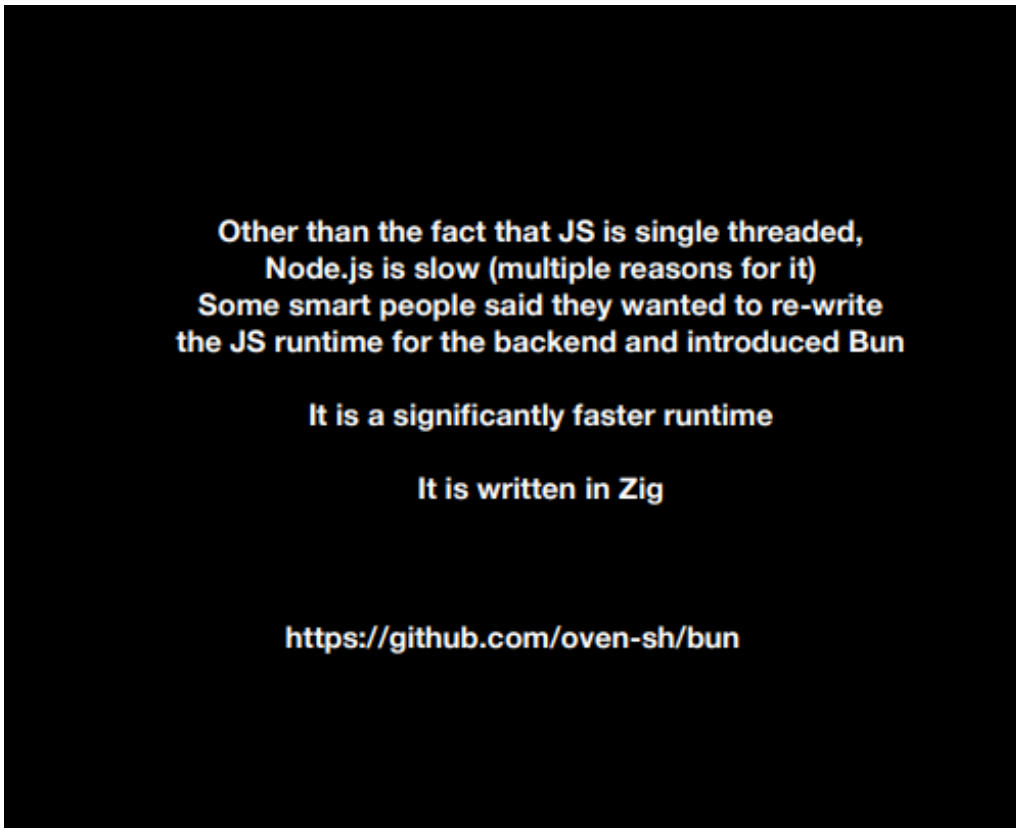
What is Node.js?

- Is it a programming language? NO!
- Is it a framework? NO!
- It is **runtime**, something which can compile Javascript and also give some additional functionality.

What is Bun?

Bun is a competitor of node.js it is written in **Zig** programming language.

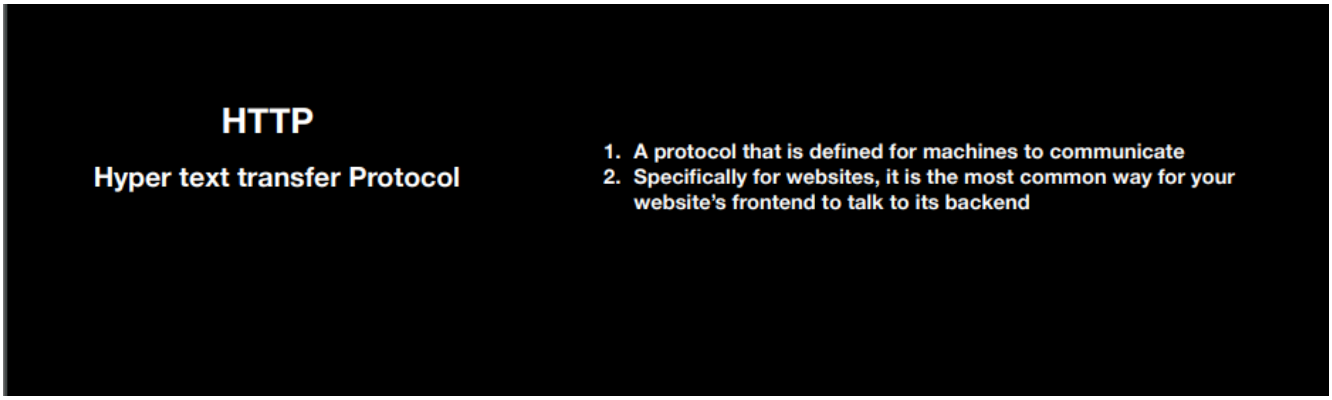
It has outperformed node.js in many ways.



What can you do with Node.js??

1. Create clis (command line interface)
2. Create a video player
3. Create a game
4. Create an **HTTP Server.**

What is an HTTP Server?



What is a protocol?
In real life eg is Shaking hand (one party extend hand other shakes)

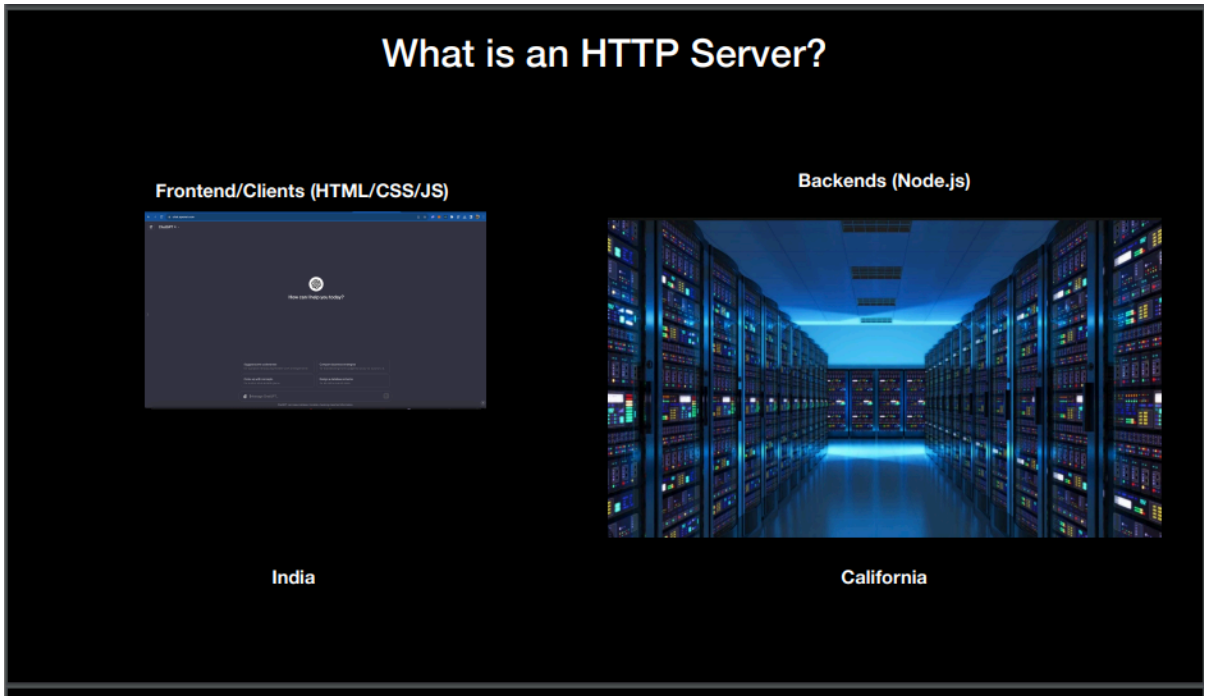
For example, live classes of Zoom is like suppose my machine is sending audio and video to Zoom which is then being sent to the viewers system.
The protocol being used here is WebRTC protocol , protocol used for video communication.

HTTP protocol is specifically for website, it is the most common way for your website's frontend to talk to its backend.

What are Frontends and Backends?

Let's see an example of chat.openai.com
Frontend:
Html/Css/Js
React/ReactNative /Flutter

Backend code we write in :
Java, rust, Golang



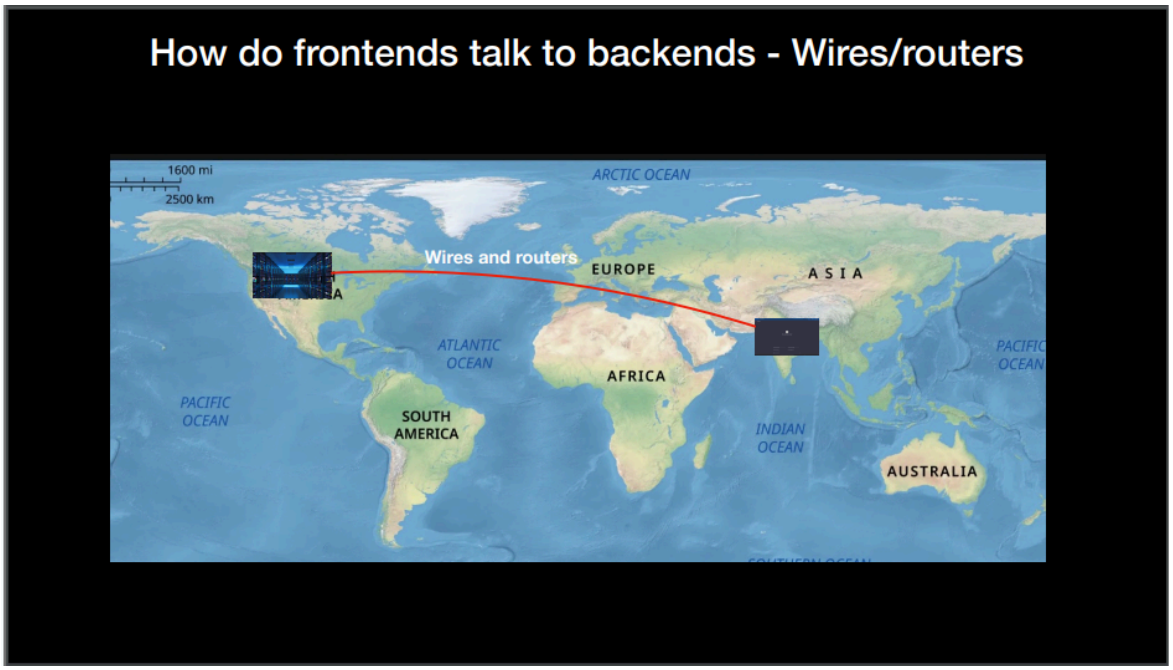
What is the backend?
When we write some prompts in chatgpt, conversion of the prompts to answer which we see coming from Chatgpt is done in Servers of Chatgpt this is the backend.
When we go to chatgpt and click on enter. The protocol that lets the frontend communicate with the backend and get some response is the HTTP protocol.
Up to 95% of the protocol used is HTTP.

E.g.
When we do any Google search we write something (Frontend) and then press the search button.
A place where some request is sent and some responses come back, a very big data sender is Backend server.

HTTP lets us communicate between the Backend and Frontend.

Our Goal for this lecture is to write an HTTP server in Node.js

How do frontends talk to backends?

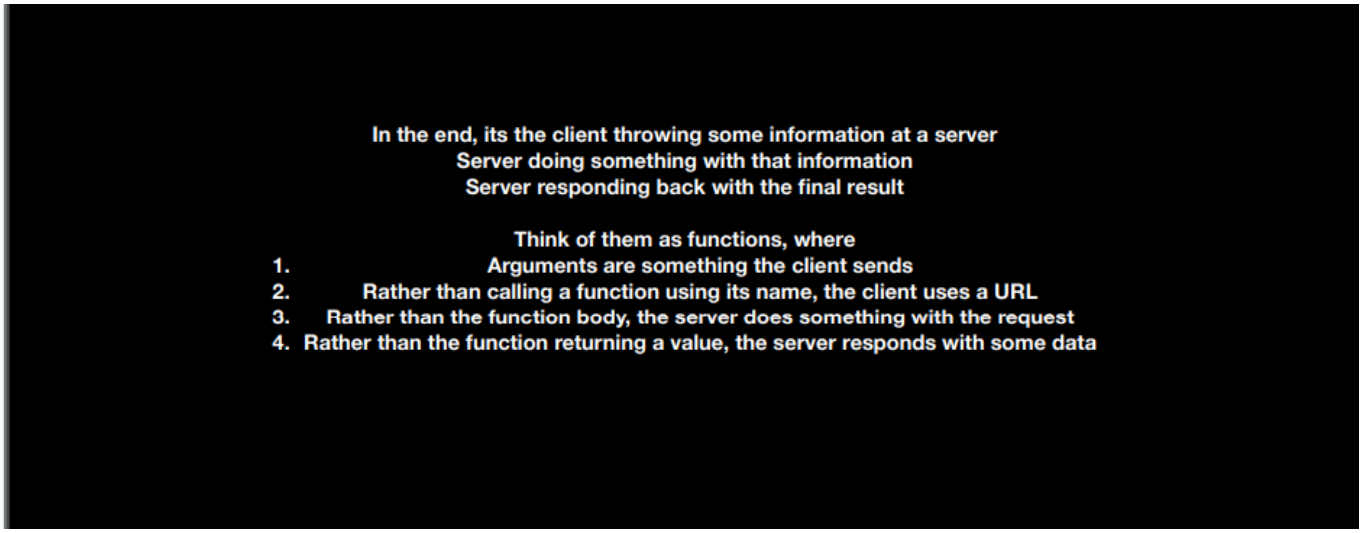


There are a large bunch of wires inside the ocean.

What is an HTTP Server?

Some code that follows the HTTP Protocol and can communicate with clients (browsers/mobile apps…)
HTTP server does a lot of work including authentication.
Think of it to be similar to the call app on your phone Which lets you communicate with your friends.

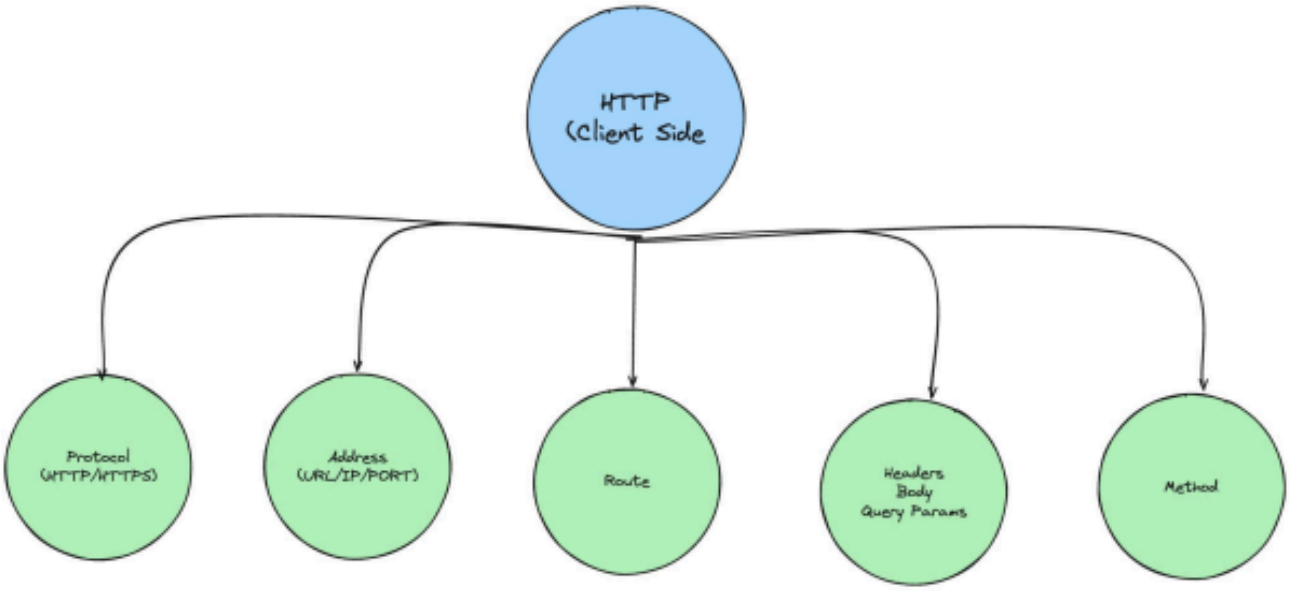
HTTP Protocol



The function is something that takes input, does something with it, and outputs it.

HTTP server is also called remote procedure calls, they are similar in function like they take some input and return some response all this is done web.
Calling function in some other system(Server)

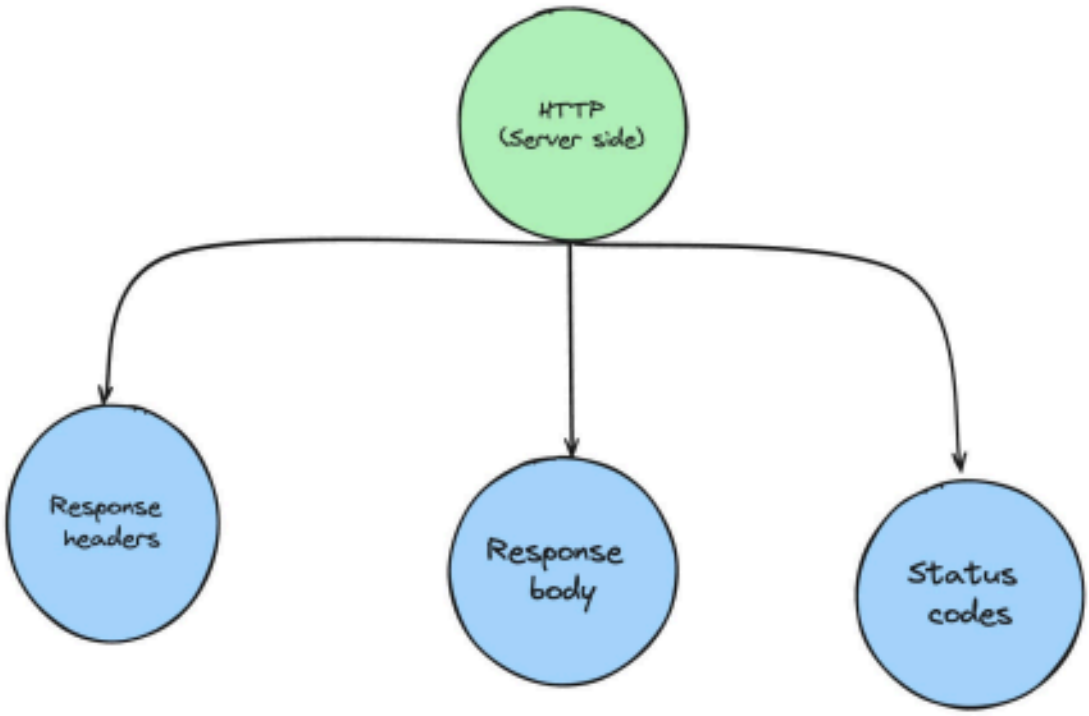
Things the client needs to worry about



Things the client needs to worry about, basically things need to know while sending the request.

Protocol(HTTP/HTTPS(Secure)): what protocol are we exposed to
Address(URL/IP/PORT): address of the backend server.
Route:
Header, Body, and Query Parameters:
Method:

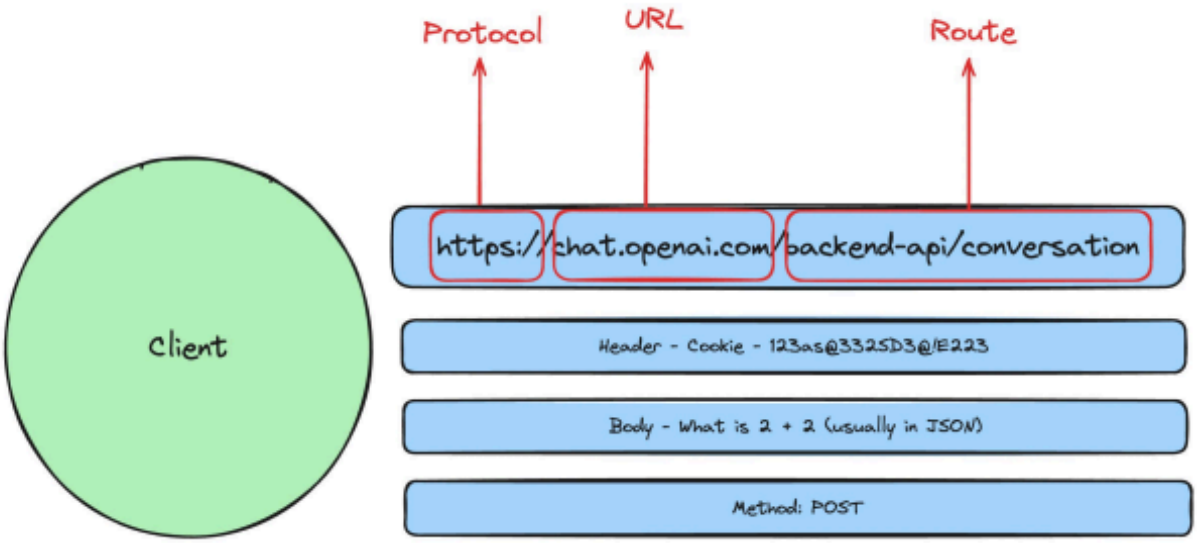
Things server need to worry about:



Response body: here the response of Google search comes from.

Client-side

Usually, the communication would happen like this.



Protocol: https

URL: chat.openai.com

Route: backend-api/conversation

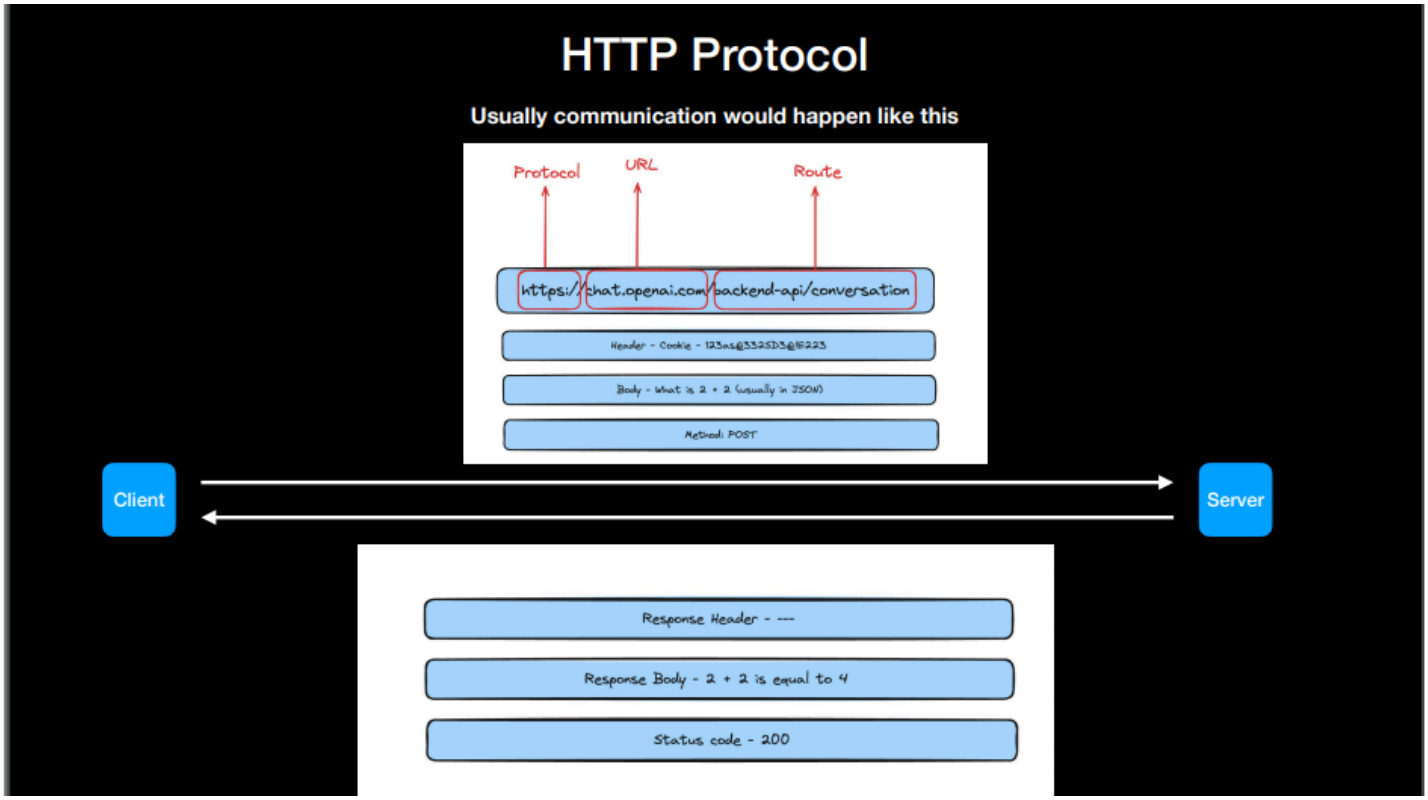
Whatever you see after a website is written, we need to specify the route to know what exactly we want to do, here conversation means that we want to get a response to our prompt from the chatgpt.

Header: Through the Cookie: Backend knows that this person is logged in

Body: Main argument we send, example if we ask prompt in chatgpt, What is 2+2 (usually in JSON it is sent)

Method: Post

Network Tab in Chrome inspect shows us, network calls.



Response header: used mostly when we are signed in, cookie sent by the server which is stored

Response Body: 2 + 2 is equal to 4

Status Code:

HTTP Protocol

What will happen when we enter the search button in the Google search?

Things that happen in your browser after you fire this request

1. **The browser parses the URL**

   Getting out where are we sending the request
2. **Does a DNS Lookup** (converts google.com to an IP)

We go to google.com but underhood it goes to the IP, the way to find the server on the internet is to specify an IP.

IP => 237.1.22.55

ChatGpt and Google servers also have an IP

DNS(Domain Name Service) Resolution, It is very hard to remember IP so they will choose a domain name and then map the IP with the domain name.

Eg It is similar to our contact list when I call Nishu, it will automatically call no. 1234567890

3. **Establishes a connection to the IP** (does handshake …)(Computer network stuff)

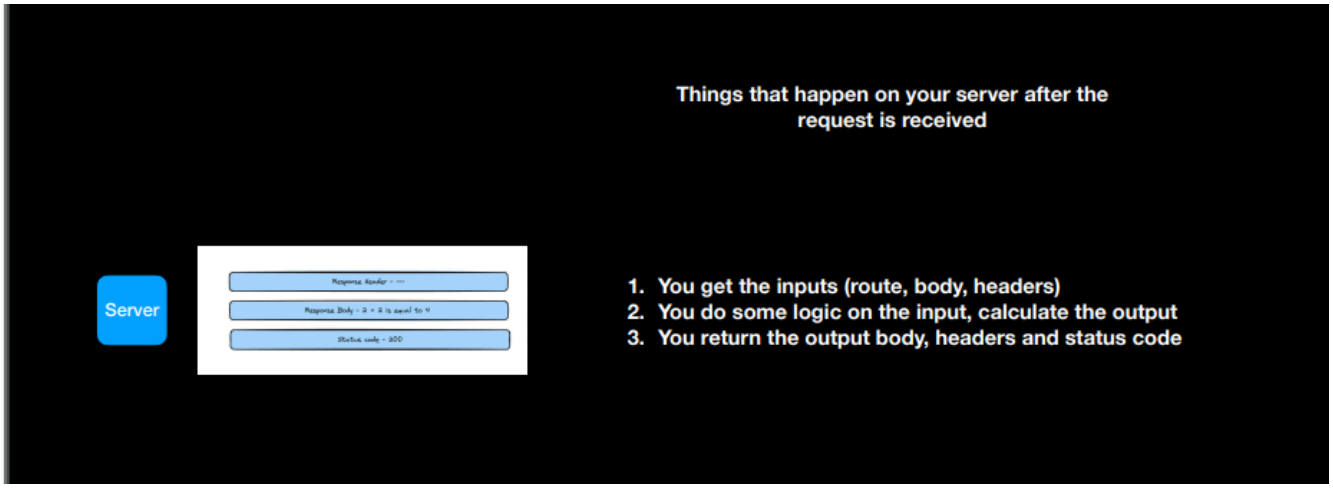
What is DNS resolution?

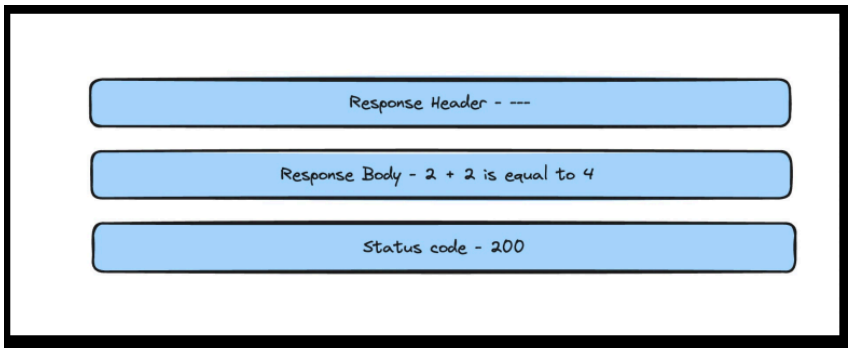
URLs are just like contacts in your phone.

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

Things that happen on your server after the request is received.





If we are using a library like express they hide a lot of these stuff, complexity of creating an http server

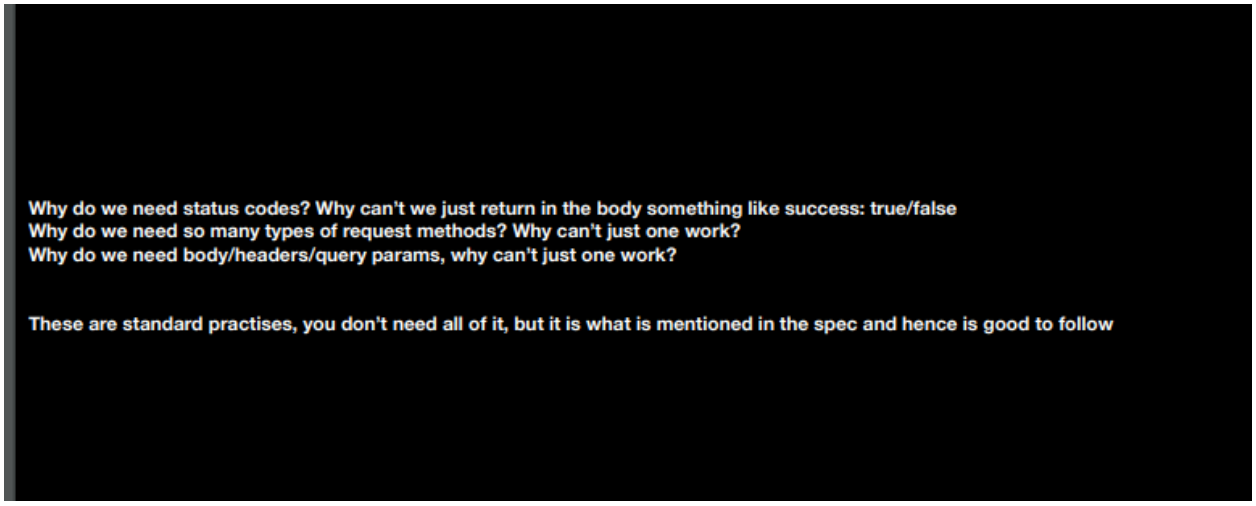
What are the common methods you can send to your backend server?

1. GET(get history of GPT)
2. PUT(updating something)
3. POST(adding a conversation)
4. DELETE

    2+2 doesn't need to be a POST request it depends on GPT backend server developers

What are the common status codes the backend responds with?

1. 200 - Everything is ok
2. 404 - Page/route not found
3. 403 - Authentication issues (access denied)
4. 500 - Internal server error (some bug in the backend)(if your backend crashes etc)



How do I create an HTTP server of my own?
How do I expose it over the internet like chatgpt.com

Q/Na

1. HTTP and https (https is more secure)
   People can snipe on the data if it is on http since it is not secure, and then pull out the password and then get into the people's FB and Google, etc.
2. Parsing URL means figuring out ip or we can say we got url https://chat.openai.com then getting chat.openai.com is parsing.

3. ipconfig will give you the local IP of the machine
4. HTTP servers can be written in many languages C, C++, boot, node.js, java
   We will use node.js using express library

Express

Run this command in the terminal, the file path must be where you are working.

npm init -y

This will create a file called package.json

Next to it create a file called **index.js** in the same directory(C:\Users\NTC\Desktop\Dev\Week2\Http_Server>)

Write these commands in vs code terminal.

npm install express

Example:

Basic HTTP server

Code

```
//The goal is to create an HTTP server

// fs comes from node.js we do not have to import it
// importing library called express
const express = require('express');
const app = express();
// express function returns an app object
const port = 3000

// fs.readFile("a.txt", 'utf-8',()=>{}) This callback function will run at
the end of the file being read
```

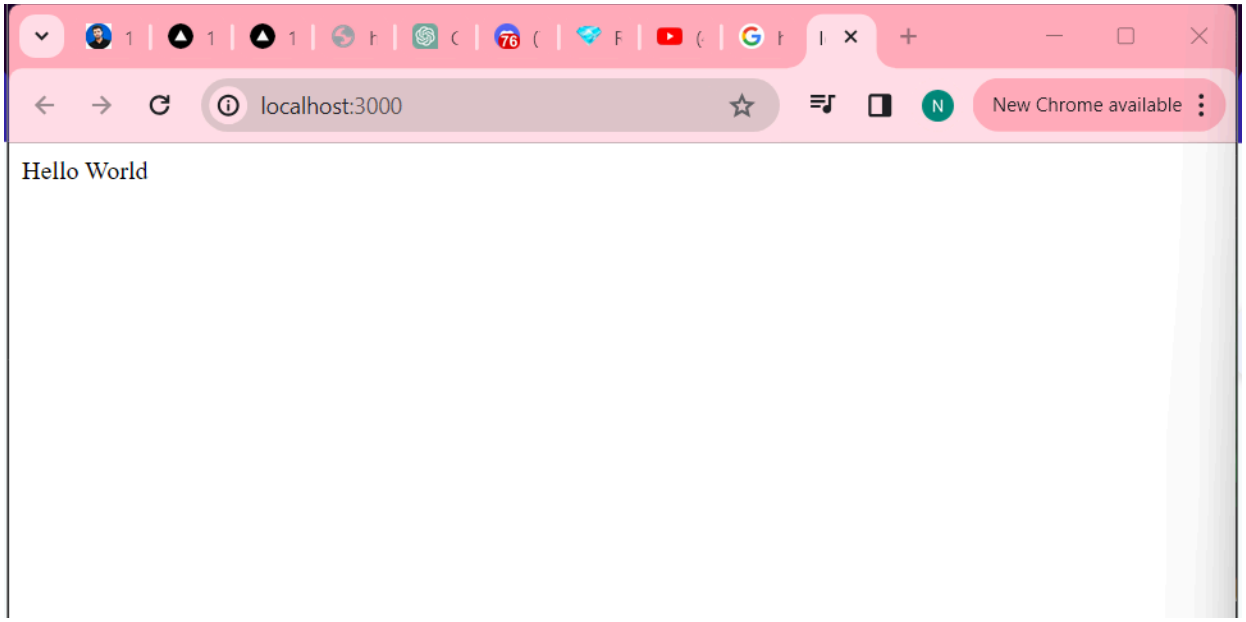
```
app.get('/', function(req, res){
// headers, body, query parameter
//This callback function will run whenever somebody tries to hit our
backend server, control reaches here(how to bind port logic has been
written by guys in express)
    res.send("Hello World");
})

// chatgpt developer would have responded to our previous example like
this
// app.post('/backend-api/conversation',function(req,res){
//     //Run a machine-learning model
//     res.send("The response")
// })
//We don't need this endpoint this is just an example

// even if we don't send a callback here it will be fine
// app.listen(port);
app.listen(port, function(){
    console.log(`Example app listening at http://localhost:${port}`);
})
```

It will host a website locally at http://localhost:3000/

It will look like this



In the console it will output:

Under the hood, this HTTP server is written in C/C++ by Express, this is just like the JS api provided by node.js(it uses an HTTP module that may use C/C++)
Express is written at the top of the http library (const http = require(“http”))

After this, we need to application-based logic. Like social media and need to learn more about databases.

Task: Create a todo app that lets users store todos on the server.

Challenging Task:

- Create an HTTP server from scratch in C.
- Create an HTTP server in Rust using actix-web
- Create an HTTP server in Golang using the guerrilla framework

Q. Javascript is single threaded then how can Node.js several requests at a time?

Node.js handles a request at a time if there is some operation like a database call inside a request it is asynchronous. Hence it can go to other requests too.

```
const express = require('express');
const app = express();
const port = 3000

app.get('/', function(req, res){
    // db call suppose which takes around 1s, db call is asynchronous
    let a = 0;
    console.log("request has been reached");
    for(i=1;i<1000000000;i++){
        a=a+i;
    }
    res.send("Hello World");
})
```

In the console, we can see that the output

request has been reached

but, the localhost:3000/ isn't displaying **Hello World** because the thread is stuck in the for loop

Q. What is the difference between express and fs library?

fs -> filesystem - read file on system, write to files on system

express -> HTTP server

Q. What is the difference between public and private IP?

There can be only 222.123.234.654

There will be a total: 255*255*255*255 IP

Private IP is repeated across.

Public IPs are unique.

Another example of a basic express http server

```
const express = require('express');
const app = express();
const port = 3000

app.get('/route-handler', function(req, res){
    // headers, body, query parameters
    //Do machine learning model in case of GPT
    res.json({
        name: "Thapa",
        age: 22
    })
})
app.listen(port)
```

From the backend, we can return HTML also, for example,

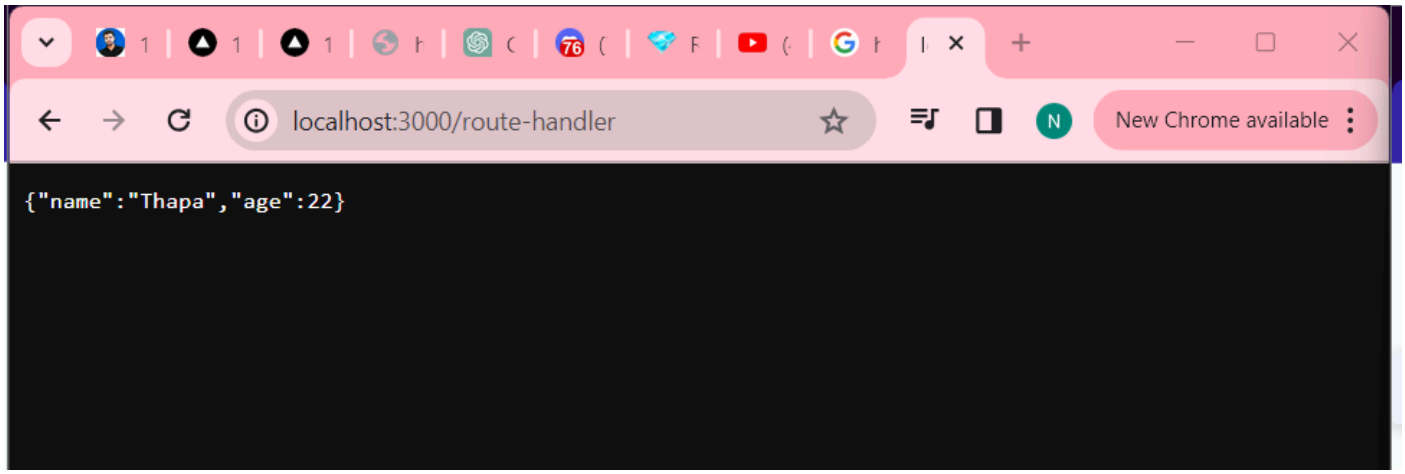
app.get('/',(req, res)=>{

res.send('Hey Bro')

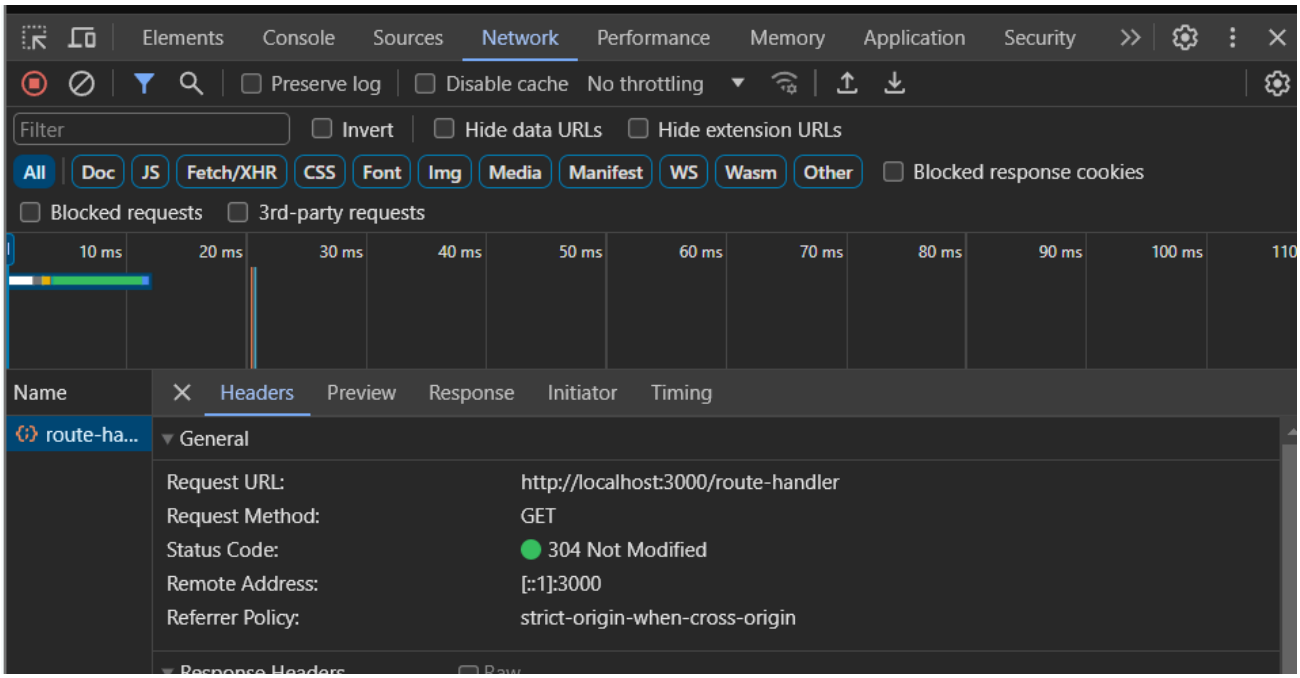
})

Browser here interpret and think that what came back it looks like HTML and render it like an html

Output:



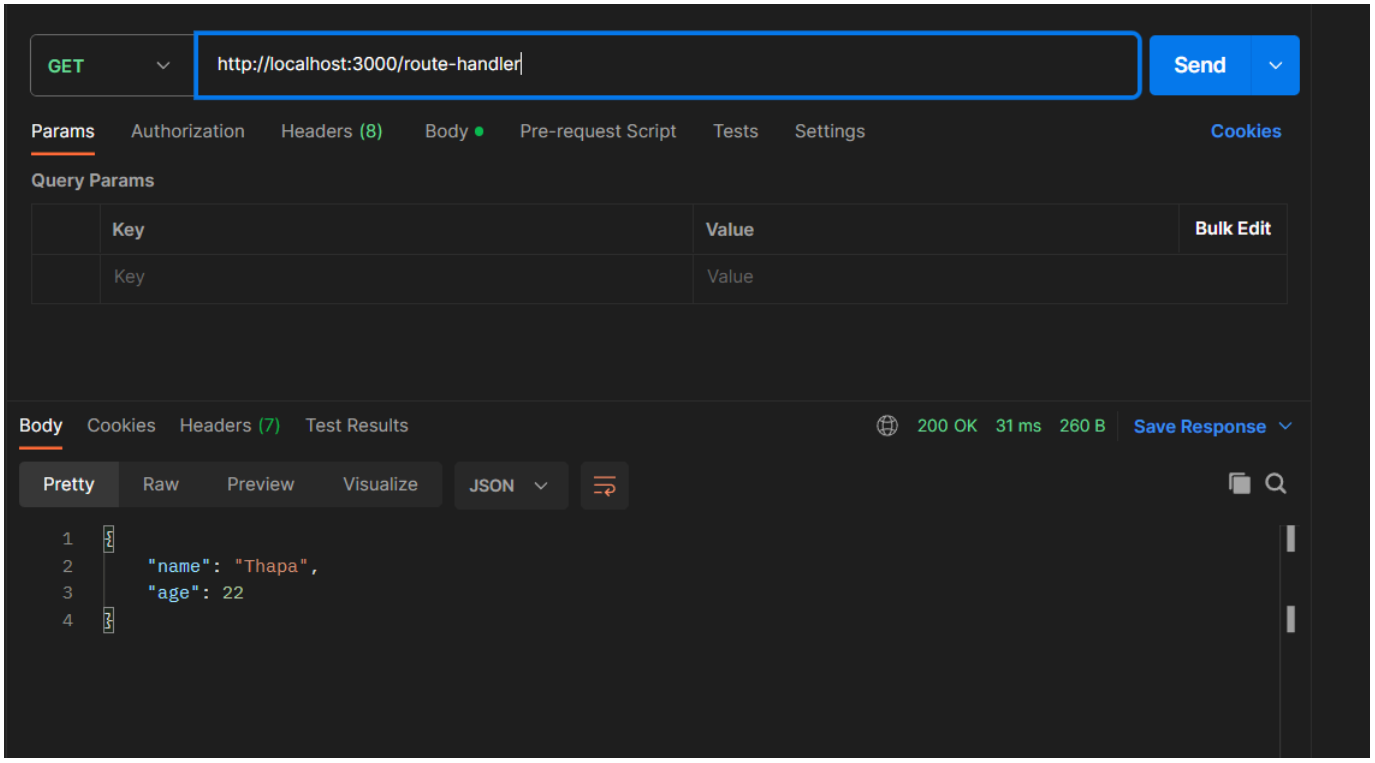
Chrome inspect tool



What is REST API?

What is POSTMAN?

POSTMAN allows us to send requests similar to what the browser does.



When we use app.post() request we can also send some data to the server

Eg

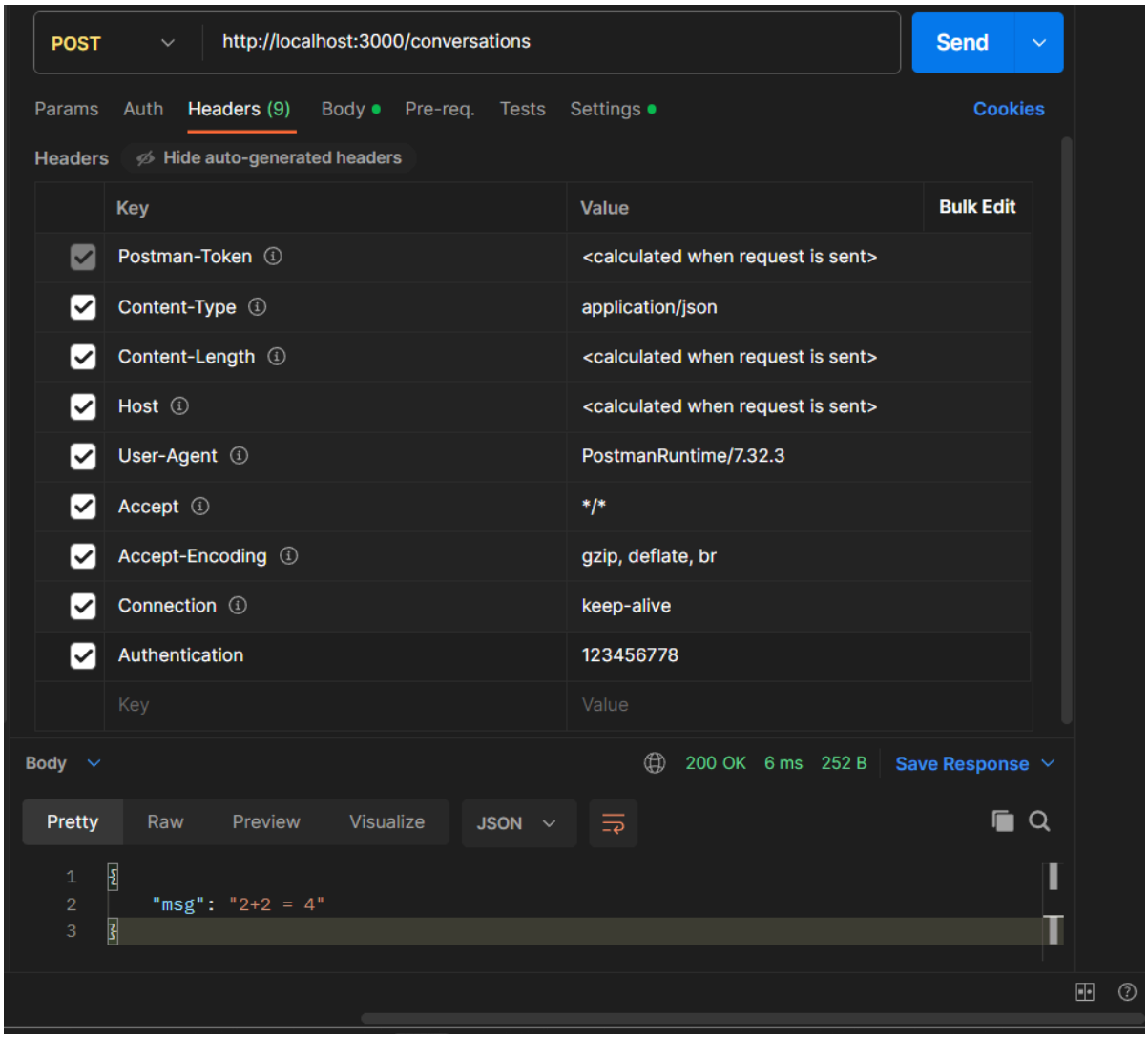
```
const express = require('express');
const app = express();
const port = 3000

app.post('/conversations',(req,res) => {
    console.log(req.headers)
    res.send({
        msg:"2+2 = 4"
    })
})

app.listen(port)
```

Start the server

Then on Postman type the URL of the local host and then select the request type (post) and press send While selecting which Headers to send



We will use the following data (since we are logging req.headers)

To access some key-value pairs from this header JSON object
Inside the post request we will write
console.log(req.headers[“authentication”])

Inside this, we can do token checks, whether the user is valid or not.

Note: If we want to see these local hosts on my other device like a mobile phone, Then run ifconfig command in the terminal and use chatgpt to find the local IP of the device and type it in Chrome for this pc and the phone must be in same wifi and you are not in restrictive.

The benefit of domain name
Suppose we use command ping 100xdev.com and then we see an IP but when we enter this IP we are not able to see the 100xdev website why is it?

Backend has a header by which it knows from where the request is being sent. For example, Google Search and 100xdev.com can have the same IP, without the header Backend is confused about where the request is sent from.

The server can host multiple applications and multiple domains can point same server and whenever the request comes server can check from where the request is being sent.

Express says that I am going to completely ignore the body hence we can't use req.body() if we log it, will show undefined. We will other libraries like body-parser

npm install body-parser

And then import it, const bodyParser = require("body-parser")

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 3000

app.use(bodyParser.json());
// middleware
app.post('/conversations',(req,res) => {
// req has request data
// res will give us some live level function which we can use to respond to the request.
// now suppose we are using chatgpt then if we want to get the main prompt through the body we will do something like this.
    console.log(req.body.message)
    console.log(req.body)
    // it will display undefined unless we import and install the body-parser library
    res.send({
        msg:"2+2 = 4"
    })
})

app.listen(port)
```

npx nodemon index.js will automatically refresh the changes and save it

Q. How can we fetch data from somewhere?

fetch("http://localhost:3000/", {method: "POST"}).then(func)

port: assume every machine has a single IP, but a single machine can run multiple processes, if we want to run multiple HTTP servers then we have to bind it to a port.

If one process is occupying the port then other processes can't occupy that port.

LAZYVIM is useful.

```
const port2 = process.env.PORT || 3000
```

Way to access the **environment variable**. Environment variables are sort of independent from the programming language we are using.

Exporting environment variables in terminal

Eg

export PORT=3005 (in macos AND Linux)

set PORT=3005 (in windows)

Now if we start the process it will run on port http://localhost:3005/

Q. What is Query parameter?

Another way to send information to backend is query

Const message = req.query.message;

There is another way to send information to backend that is throught the url

Example : http://localhost:3005/route-handler?message=123&b=1&c=1

Query parameter are bunch of parameters on query itself.

Example:

We are using an Environment variable and using the query parameter also

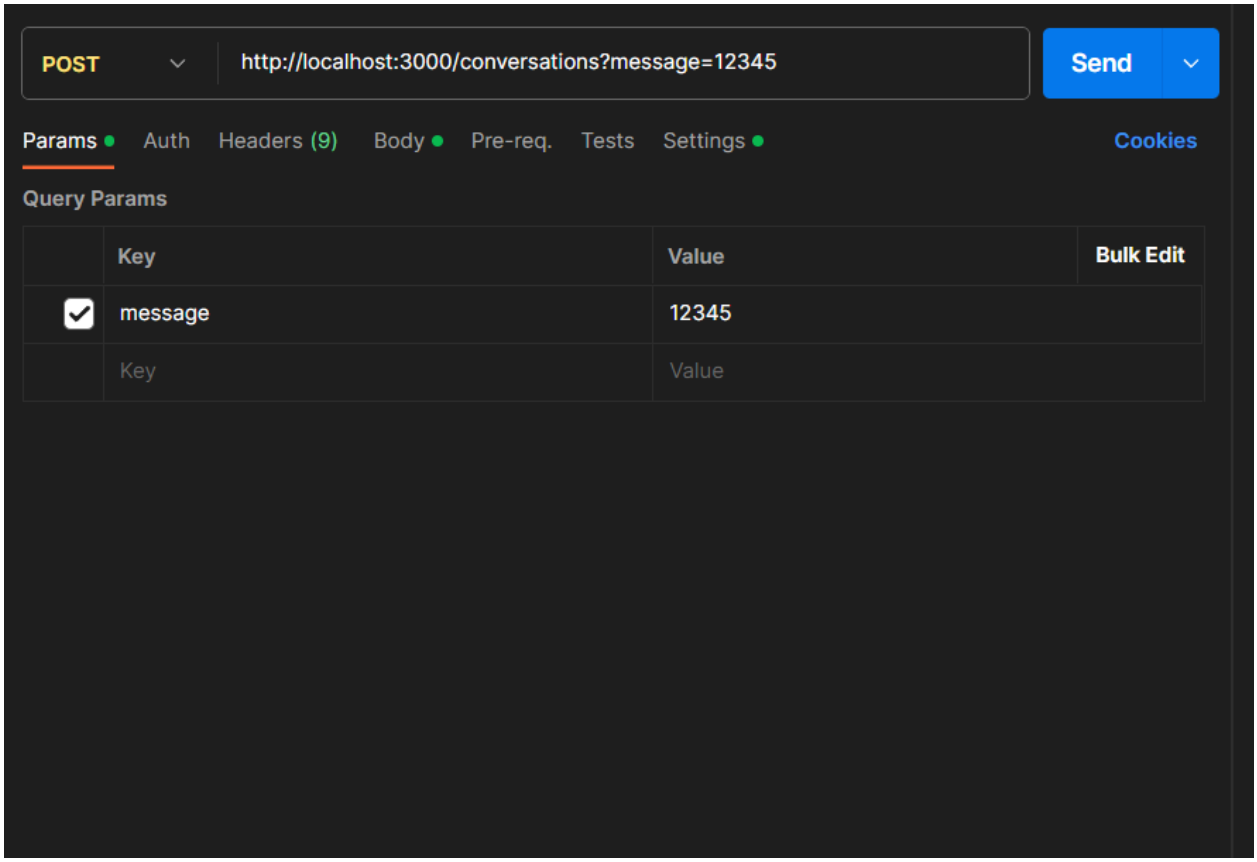
```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
// const port = 3000
const port = process.env.PORT || 3000
app.use(express.json());

app.post('/conversations',(req,res) => {
    const message = req.query.message;
    // it will display undefined unless we import and install the
body-parser library
    console.log(message);
    res.send({
        msg:"2+2 = 4"
    })
})

app.listen(port)
```

Terminal

Postman:



If it is a get request then most of the time we will send information through the query parameters

npm: is the node package manager it lets us bring those packages to out machine.

npx: is particularly useful for running packages without installing them globally.

Examples

npx nodemon index.js

Here we can see that nodemon is not present in our **package.json** file.

Q. Why doesn't express allow or read body?

We can use body-parser . The body-parser module enables us to parse incoming request bodies in a middleware. Express.js server needs to know what type of data you're sending over the network, so it knows how to parse it.

- Javascript and node.js are different, Javascript can be run in a browser and by the V8 engine. Js is used in frontend and node.js is used in backend.
- app.listen() is a functionality of express where it asks the user port to listen and express developers will write the logic of the HTTP server which will listen to this port. app.listen() starts to listen to all the requests to the port.
- Node.js compiles your code(Js to Binary) and give access to low level APIs(reading a file, opening a socket, listening to a port)
- Http server is started when we do app.listen(port)
- Sending status code

  Example

```
app.get('/',(req,res) => {
        console.log(req.headers.authorization
        setTimeout(function(){
                res.status(401).send("Hello world")
        },2000)
})
```