Node.js runtime | HTTP

Week 2.2

What are we learning today

Foundation

Foundation Javascript, async nature of JS
Node.js and its runtime
Databases (NoSQL/SQL)
Mongo and Postgres deep dive
Typescript beginner to advance

Backend

Backend communication protocols
Express basic to advance
ORMs
Middlewares, routes, status codes, global catches
Zod
MonoRepos, turborepo
Serverless Backends
OpenAPI Spec
Autogenerated clients
Authentication using external libraries
Scaling Node.js, performance benchmarks
Deploying npm packages

Frontend

Reconcilers and Frontend frameworks
React beginner to advance
Internals of state, Context API
State management using recoil
CSS you need to know of, Flexbox, basic styling
Frontend UI frameworks, Deep dive into Tailwind
Containerization, Docker
Next.js
Custom hooks
In house auth using next auth

Basic Devops

Docker end to end
Deploying to AWS servers
Newer clouds like fly/Remix
Nginx and reverse proxies

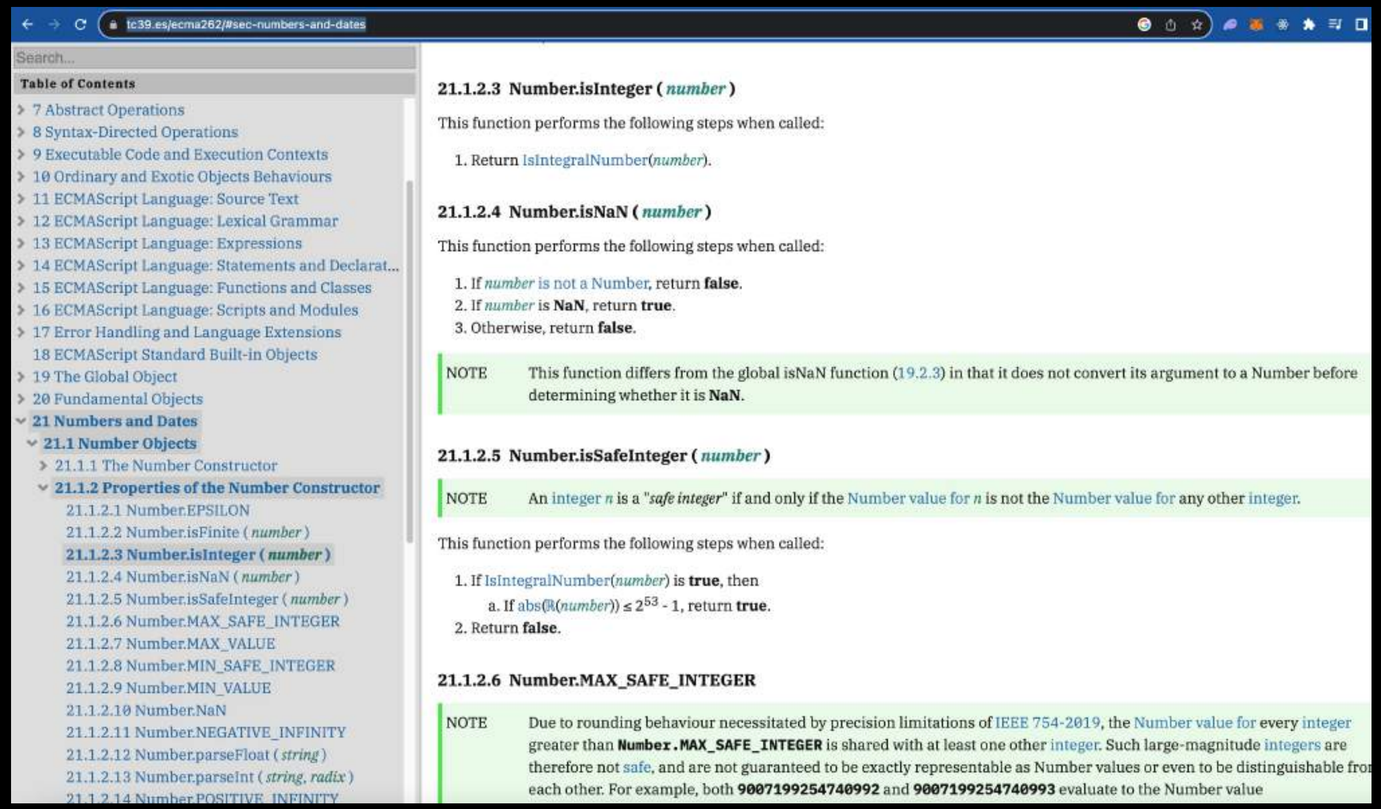
Projects

GSoC Project setting up and issue solving
Building Paytm/Wallet End to End

Node.js and its runtime

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

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



ECMAScript

ECMAScript is a scripting language specification on which JavaScript is based. Ecma International is in charge of standardizing ECMAScript.

https://tc39.es/ecma262/#sec-numbers-and-dates

What is ECMAScript

What is Javascript?

What is Node.js?

What is Bun?

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.

What is ECMAScript

What is Javascript?

What is Node.js?

What is Bun?

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.

Date, var, const, let, function

setTimeout
fs.readFile

What is ECMAScript

What is Javascript?

What is Node.js?

What is Bun?

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 is responsible for compiling JavaScript code into native machine code before executing it, which greatly improves performance.

Common JS Browser engines

1. V8 - Used by google chrome/chromium - C

https://github.com/v8/v8

2. SpiderMonkey - Used by Firefox - C + Rust

https://spidermonkey.dev/

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

Some smart people took out the V8 engine
Added some Backend things (filesystem reads) on top to create a new runtime to compete with BE languages like Java.

JS was never meant to be run in the backend
Eventually became very popular and is a popular choice of runtime on the backend

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

Other than the fact that JS is single threaded,
Node.js is slow (multiple reasons for it)
Some smart people said they wanted to re-write
the JS runtime for the backend and introduced Bun

It is a significantly faster runtime

It is written in Zig

https://github.com/oven-sh/bun

We will be focusing on Node.js
Specifically, how to write Backend applications using Javascript

What can you do with Node.js?

1. Create clis
2. Create a video player
3. Create a game
4. Create an HTTP Server

What is an HTTP Server?

What is an HTTP Server?

HTTP

Hyper text transfer Protocol

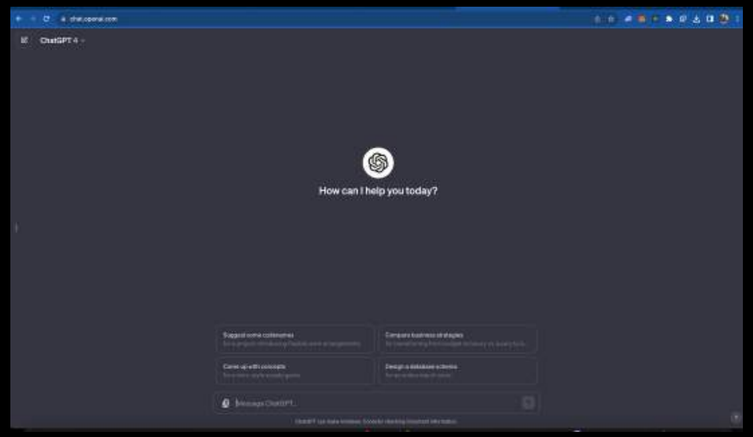
1. A protocol that is defined for machines to communicate
2. Specifically for websites, it is the most common way for your website's frontend to talk to its backend

What is an HTTP Server?

First lets understand what are Frontends and Backends

What is an HTTP Server?

Frontend/Clients (HTML/CSS/JS)

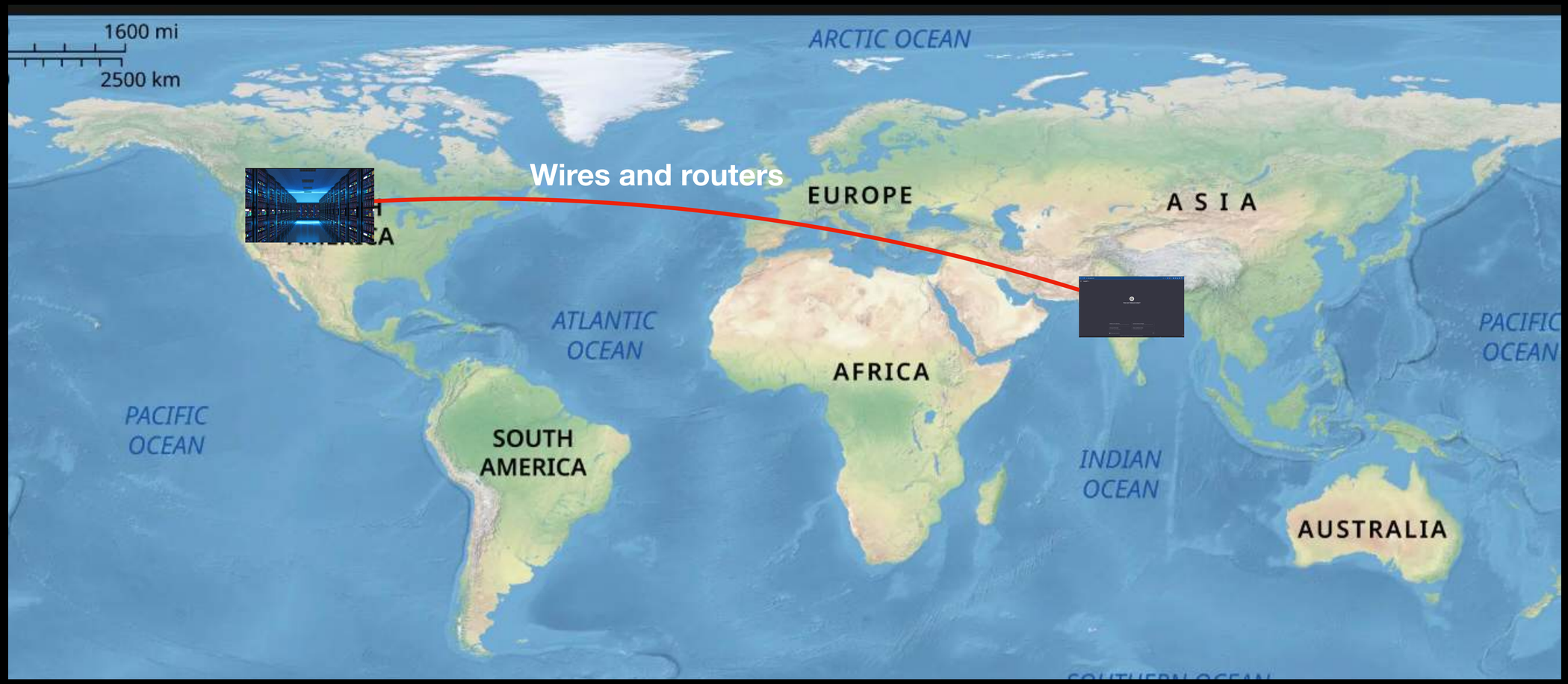


India

Backends (Node.js)

California

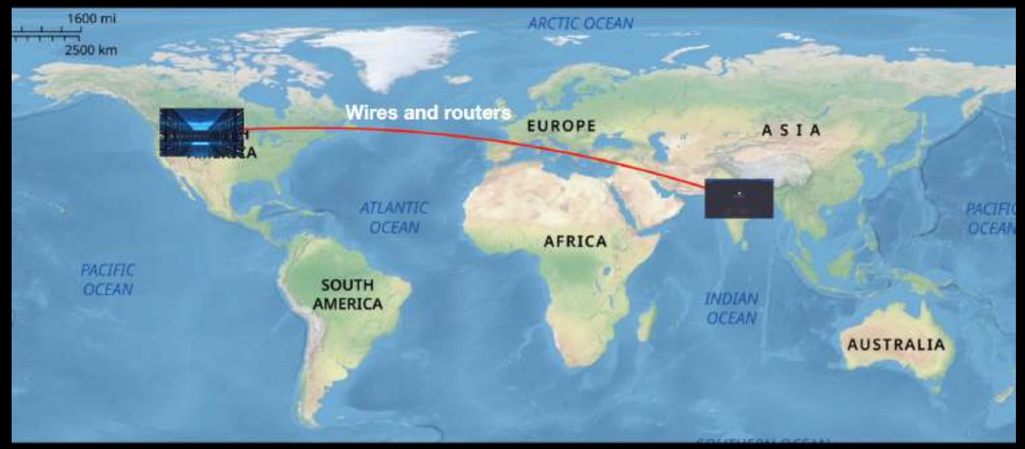
How do frontends talk to backends - Wires/routers



What is an HTTP Server?

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



HTTP Protocol

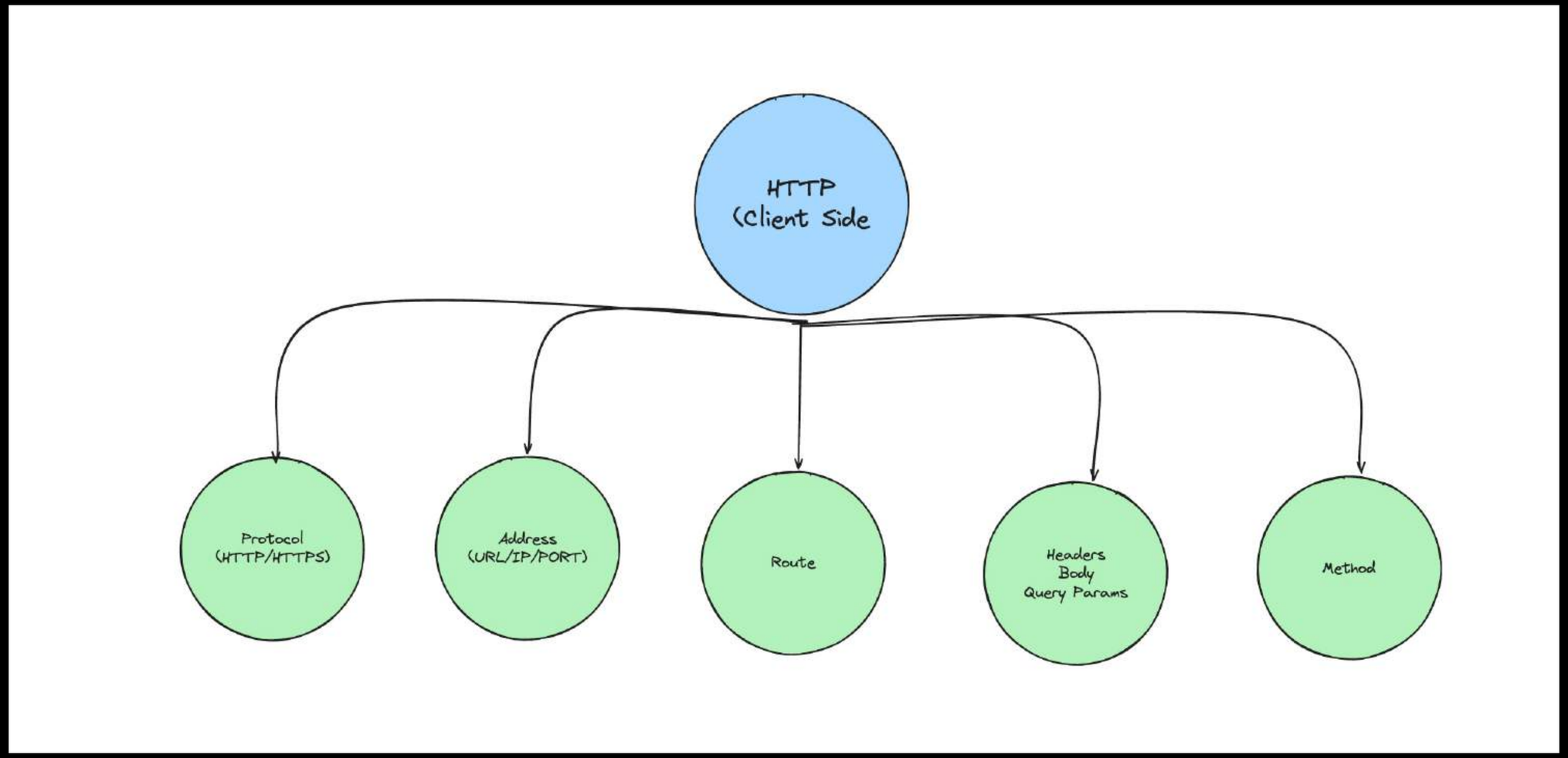
In the end, its the client throwing some information at a server
Server doing something with that information
Server responding back with the final result

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

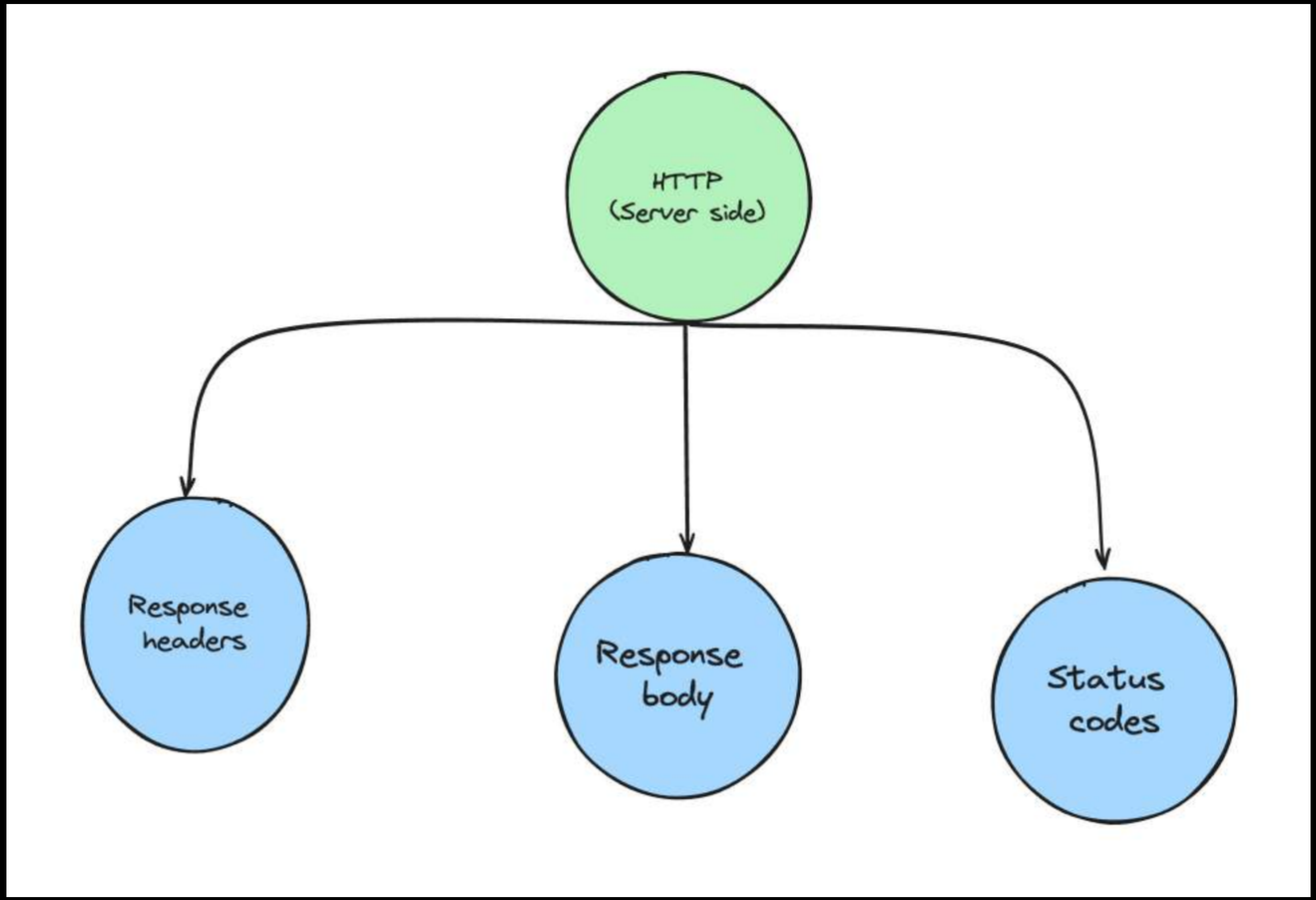
HTTP Protocol

Things client needs to worry about



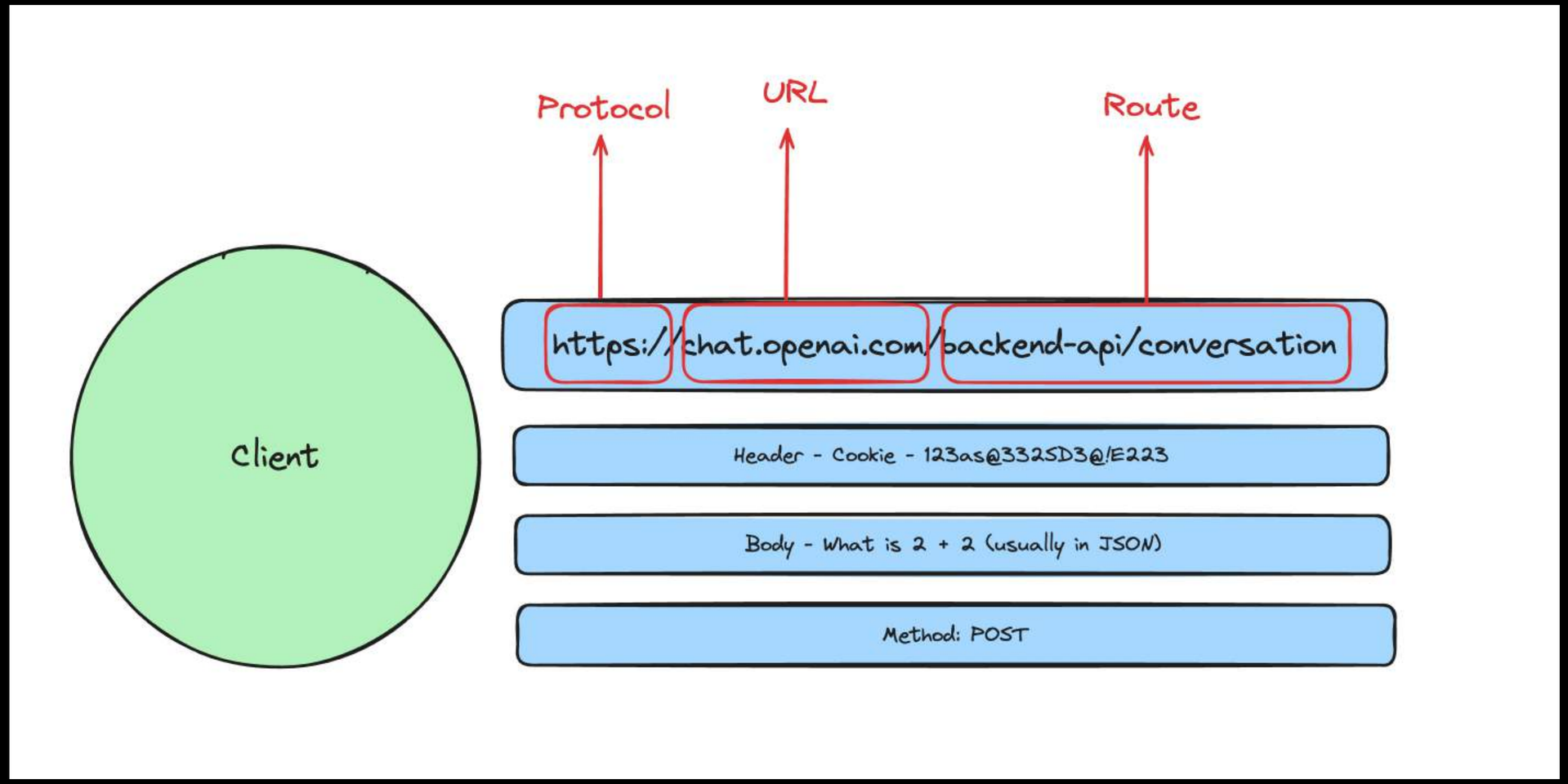
HTTP Protocol

Things server needs to worry about



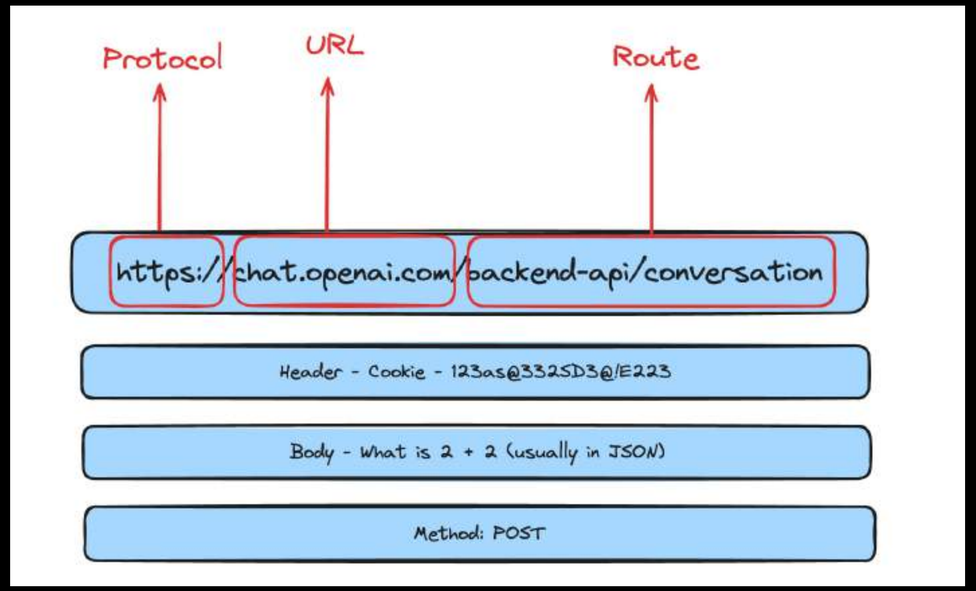
HTTP Protocol

Usually communication would happen like this



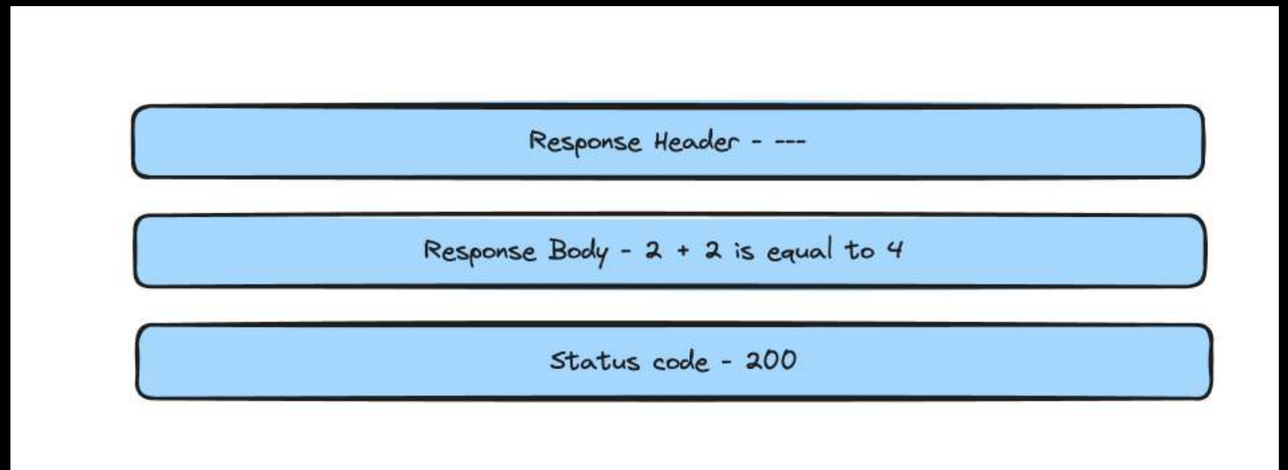
HTTP Protocol

Usually communication would happen like this



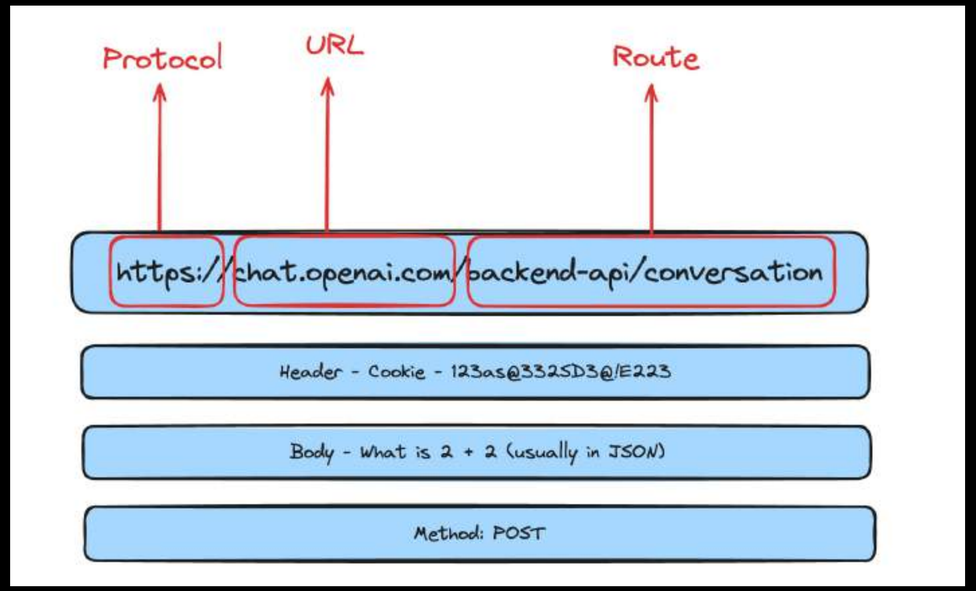
Client → Server

Server → Client



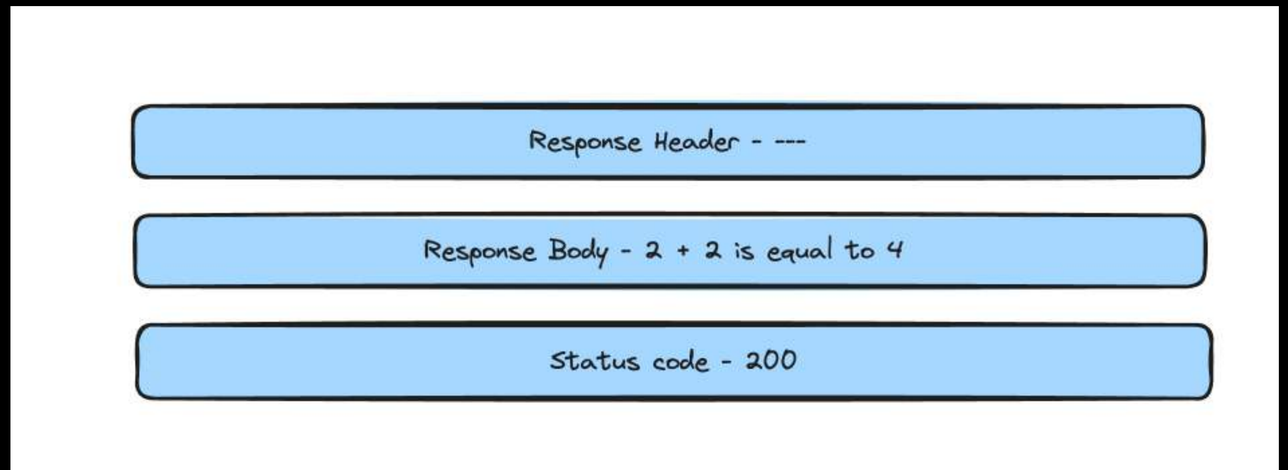
HTTP Protocol

Usually communication would happen like this



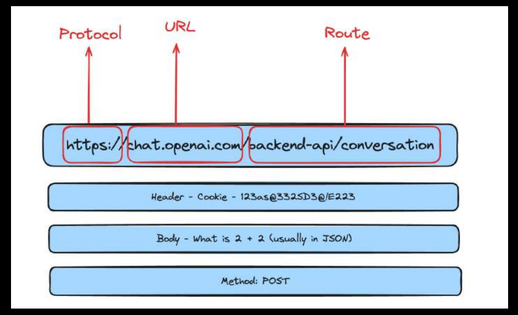
Client → Server

Server → Client



HTTP Protocol

Client



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

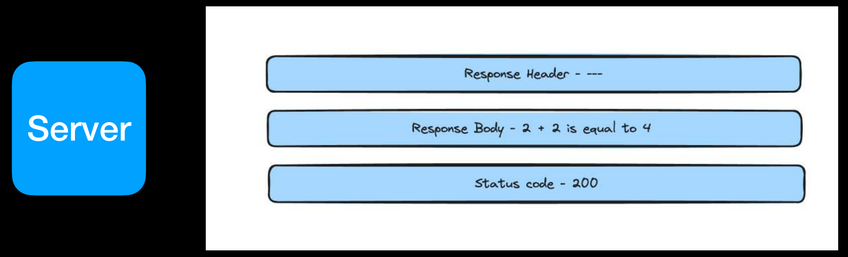
1. **Browser parses the URL**
2. **Does a DNS Lookup (converts google.com to an IP)**
3. **Establishes a connection to the IP (does handshake…)**

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

HTTP Protocol

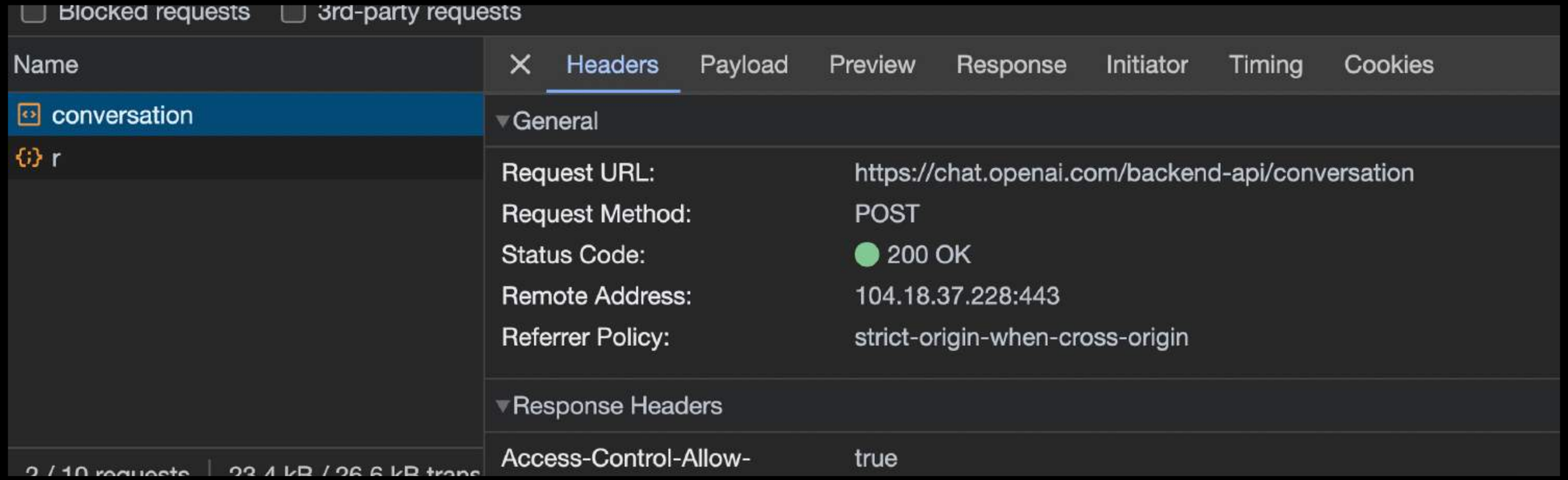
Things that happen on your server after the request is received



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

HTTP Protocol

Lets see it in action



HTTP Protocol

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

1. **GET**
2. **POST**
3. **PUT**
4. **DELETE**

HTTP Protocol

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**
4. **500 - Internal server error**

HTTP Protocol

Why do we need status codes? Why can’t we just return in the body something like success: true/false
Why do we need so many types of request methods? Why can’t just one work?
Why do we need body/headers/query params, why can’t just one work?

These are standard practises, you don’t need all of it, but it is what is mentioned in the spec and hence is good to follow

HTTP Protocol

Question at this point -
How do I create a HTTP server of my own?
How to I expose it over the internet like chatgpt.com

HTTP Protocol

Do you remember the fs library?
We used to to read from a file

index.js > ...

```js
let a = 1;
console.log(a);

fs.readFile("a.txt", "utf-8", (err, data) => {
  console.log("data read from the file is ");
  console.log(data);
})

let ans = 0;
for (let i = 0; i<100; i++) {
  ans = ans + i;
}
console.log(ans);
```

HTTP Protocol

Similarly, there are many libraries that let you create HTTP Servers
The most famous one is express

A great exercise to do is to create an HTTP server from scratch in C/C++
It is out of scope for this course, but if you’re looking for a challenge

HTTP Protocol

Library that we are using - Express

Lets create a simple HTTP Server

```
const express = require('express' 4.18.2 )
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

https://expressjs.com/en/starter/hello-world.html