## Page:1

# HTTP

## 📚 Server Basics and HTTP Protocol

### **What is a Server?**

* A server is a system that serves requests from clients, responding with the necessary data or resources.
* The client is the system making the request to the server.

### **HTTP and HTTPS**

* **HTTP** (HyperText Transfer Protocol): A set of rules for communication between a client and a server.
* **HTTPS** (HTTP Secure): HTTP with an added layer of security using SSL (Secure Socket Layer).

### **HTTP Methods/Verbs**

* **GET**: Read data from the server.
* **PUT**: Replace the entire resource on the server.
* **PATCH**: Modify a specific part of a resource.
* **DELETE**: Remove a resource from the server.
* **POST**: Add or send data to the server.

These methods enable CRUD operations (Create, Read, Update, Delete).

## **🛠️ Creating a Basic Server with Node.js**

1. **Setup a Node Project**
   * Initialize a project: npm init -y
   * Create a file named index.js.
2. **Create a Server Using HTTP Module**

javascriptCopy code

const http = require("http");

const server = http.createServer((request, response) => {

if (request.url === "/") {

response.end("Hello");

} else if (request.url === "/reports") {

response.end("Here are the reports");

} else if (request.url === "/data") {

response.end("Data....");

## Page:2

} else {

response.end("Invalid End Point");

}

});

server.listen(4500, () => {

console.log("Listening on port 4500");

});

* This server listens on port 4500 and responds based on the request URL.

1. **.end vs .write**
   * .write: Sends data to the client but does not end the response.
   * .end: Signals the end of the response, letting the client know all data has been sent.

## **📂 Sending Data from a File**

* You can serve data from a file as a response:

javascriptCopy code

const http = require("http");

const fs = require("fs");

const server = http.createServer((req, res) => {

if (req.url === "/data") {

fs.readFile("./text.txt", { encoding: "utf-8" }, (err, data) => {

if (err) {

res.write("No data\n");

res.end(err);

} else {

res.end(data);

} }); }});

server.listen(4500, () => {

console.log("Listening on port 4500");});

## Page:3

## **📑 Headers in HTTP Responses**

* **Headers** provide additional information about the request or response. For example, specifying the type of content:

javascriptCopy code

res.setHeader("Content-type", "text/html");

res.end("<h1>Hello Guys!!</h1>");

## **📋 Summary**

* A server handles requests from clients, serving data or resources.
* HTTP/HTTPS are protocols defining how this communication happens, with methods like GET, POST, PUT, PATCH, and DELETE for different operations.
* In Node.js, the http module allows creating servers to handle different endpoints and serve responses, including files and headers.

This overview covers the essentials of setting up a basic server, understanding HTTP, and handling different types of requests and responses.

The server.listen(4500, () => { console.log("Listening on port 4500"); }); statement is used to start a server and make it listen for incoming connections on a specified port—in this case, port 4500.

# **Assinmnet is Here**

**Zip File is Here**

[httpassinment.zip](https://prod-files-secure.s3.us-west-2.amazonaws.com/573b8a4c-45ba-456a-af81-7153ceb80491/ec77004b-c776-4e67-b5d8-ec34466a52b5/httpassinment.zip)

## **instructions for Project**

1. **Setup the Project**:
   * Clone the repository or download the project files.
   * Run npm install to install the dependencies.
2. **Run the Server**:
   * Use the command npm run serverto start the server.
3. **Access the Endpoints**:
   * Open a web browser or use tools like Thunder Clinet to access the following routes:

## Page:4

* + - /home: Returns a welcome message.
    - /about: Returns an about message.
    - /getproductdata: Returns the product data from db.json.

1. /user: Returns the user data from db.json.

This setup will provide a basic introduction to handling HTTP routes, reading data from a JSON file, and serving it as a response using Express.js in Node.js.

# http functions

{

\_connectionListener: [Function: connectionListener],

METHODS: [

'ACL', 'BIND', 'CHECKOUT',

'CONNECT', 'COPY', 'DELETE',

'GET', 'HEAD', 'LINK',

'LOCK', 'M-SEARCH', 'MERGE',

'MKACTIVITY', 'MKCALENDAR', 'MKCOL',

'MOVE', 'NOTIFY', 'OPTIONS',

'PATCH', 'POST', 'PROPFIND',

'PROPPATCH', 'PURGE', 'PUT',

'QUERY', 'REBIND', 'REPORT',

'SEARCH', 'SOURCE', 'SUBSCRIBE',

'TRACE', 'UNBIND', 'UNLINK',

'UNLOCK', 'UNSUBSCRIBE'

],

STATUS\_CODES: {

## Page:5

'100': 'Continue',

'101': 'Switching Protocols',

'102': 'Processing',

'103': 'Early Hints',

'200': 'OK',

'201': 'Created',

'202': 'Accepted',

'203': 'Non-Authoritative Information',

'204': 'No Content',

'205': 'Reset Content',

'206': 'Partial Content',

'207': 'Multi-Status',

'208': 'Already Reported',

'226': 'IM Used',

'300': 'Multiple Choices',

'301': 'Moved Permanently',

'302': 'Found',

'303': 'See Other',

'304': 'Not Modified',

'305': 'Use Proxy',

'307': 'Temporary Redirect',

'308': 'Permanent Redirect',

'400': 'Bad Request',

## Page:6

'401': 'Unauthorized',

'402': 'Payment Required',

'403': 'Forbidden',

'404': 'Not Found',

'405': 'Method Not Allowed',

'406': 'Not Acceptable',

'407': 'Proxy Authentication Required',

'408': 'Request Timeout',

'409': 'Conflict',

'410': 'Gone',

'411': 'Length Required',

'412': 'Precondition Failed',

'413': 'Payload Too Large',

'414': 'URI Too Long',

'415': 'Unsupported Media Type',

'416': 'Range Not Satisfiable',

'417': 'Expectation Failed',

'418': "I'm a Teapot",

'421': 'Misdirected Request',

'422': 'Unprocessable Entity',

'423': 'Locked',

'424': 'Failed Dependency',

'425': 'Too Early',

## Page:7

'426': 'Upgrade Required',

'428': 'Precondition Required',

'429': 'Too Many Requests',

'431': 'Request Header Fields Too Large',

'451': 'Unavailable For Legal Reasons',

'500': 'Internal Server Error',

'501': 'Not Implemented',

'502': 'Bad Gateway',

'503': 'Service Unavailable',

'504': 'Gateway Timeout',

'505': 'HTTP Version Not Supported',

'506': 'Variant Also Negotiates',

'507': 'Insufficient Storage',

'508': 'Loop Detected',

'509': 'Bandwidth Limit Exceeded',

'510': 'Not Extended',

'511': 'Network Authentication Required'

},

Agent: [Function: Agent] { defaultMaxSockets: Infinity },

ClientRequest: [Function: ClientRequest],

IncomingMessage: [Function: IncomingMessage],

OutgoingMessage: [Function: OutgoingMessage],

Server: [Function: Server],

## Page:8

ServerResponse: [Function: ServerResponse],

createServer: [Function: createServer],

validateHeaderName: [Function: wrappedFn] { withoutStackTrace: [Function (anonymous)] },

validateHeaderValue: [Function: wrappedFn] { withoutStackTrace: [Function (anonymous)] },

get: [Function: get],

request: [Function: request],

setMaxIdleHTTPParsers: [Function: setMaxIdleHTTPParsers],

maxHeaderSize: [Getter],

globalAgent: [Getter/Setter]

}

# **Automatice server run**

## **Install : npm i nodemon**

### **And add script in package.json**

### **"server":"nodemon filename" that required auto run**

--------------------------------------------------------------------------------------------------------------------------------------

## Page:9

# **Program**

## **Index.js**

const http = require("http");

// console.log(http);

const server = http.createServer((req, res) => {

// console.log(req); //buffering

// console.log(req.url);

const fs = require("fs");

if (req.url == "/") {

// res.end("home page");

res.write("home page");

res.end();

} else if (req.url == "/about") {

res.end("about page");

} else if (req.url == "/getdata") {

fs.readFile("./db.json", "utf8", (err, data) => {

if (err) {

res.end("Data Not Found");

} else {

res.end(data);

}

});

} else {

res.end("page not found ERROR 404");

}

});

server.listen(8080, () => {

console.log("server is running");

## Page:10

});

//http://localhost:8080

--------------------------------------------------------------------------------------------------------------------------------------

## **db.json**

[

{

"id": 1,

"title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",

"price": 109.95,

"description": "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",

"category": "men's clothing",

"image": "https://fakestoreapi.com/img/81fPKd-2AYL.\_AC\_SL1500\_.jpg",

"rating": {

"rate": 3.9,

"count": 120

}

},

{

"id": 2,

"title": "Mens Casual Premium Slim Fit T-Shirts ",

"price": 22.3,

"description": "Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing. And Solid stitched shirts with round neck made for durability and a great fit for casual fashion wear and diehard baseball fans. The Henley style round neckline includes a three-button placket.",

"category": "men's clothing",

"image": "https://fakestoreapi.com/img/71-3HjGNDUL.\_AC\_SY879.\_SX.\_UX.\_SY.\_UY\_.jpg",

"rating": {

"rate": 4.1,

## Page:11

"count": 259

}

}]

--------------------------------------------------------------------------------------------------------------------------------------

## **Package.json**

{

  "name": "http",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "dependencies": {

    "anymatch": "^3.1.3",

    "balanced-match": "^1.0.2",

    "binary-extensions": "^2.3.0",

    "brace-expansion": "^1.1.11",

    "braces": "^3.0.3",

    "chokidar": "^3.6.0",

    "concat-map": "^0.0.1",

    "debug": "^4.3.6",

    "fill-range": "^7.1.1",

    "glob-parent": "^5.1.2",

    "has-flag": "^3.0.0",

    "ignore-by-default": "^1.0.1",

    "is-binary-path": "^2.1.0",

    "is-extglob": "^2.1.1",

    "is-glob": "^4.0.3",

    "is-number": "^7.0.0",

## Page:12

"minimatch": "^3.1.2",

    "ms": "^2.1.2",

    "nodemon": "^3.1.4",

    "normalize-path": "^3.0.0",

    "picomatch": "^2.3.1",

    "pstree.remy": "^1.1.8",

    "readdirp": "^3.6.0",

    "semver": "^7.6.3",

    "simple-update-notifier": "^2.0.0",

    "supports-color": "^5.5.0",

    "to-regex-range": "^5.0.1",

    "touch": "^3.1.1",

    "undefsafe": "^2.0.5"

  },

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "server":"nodemon index.js"

  },

  "keywords": [],

  "author": "",

  "license": "ISC"

}