Node.js

- Node.js is an open source server environment.
- Node.js allows you to run JavaScript on the server.

Creating a simple HTTP server

1. Importing “http” module:

```
const http = require("http");
```

2. Create an HTTP server:

```
const server = http.createServer();
server.listen(3000);
```

3. Check if server is running:

```
server.listen(3000, () => {
    console.log('Server running at http://localhost:3000');
});
```

4. Send response by response.write() and finish response by response.end():

```
const server = http.createServer((request, response) => {
    response.write("Hello, World")
    response.end()
});
```

5. Send header:

```
const server = http.createServer((request, response) => {
    response.writeHead(200, {'Content-Type': 'text/plain'})
    response.write("Hello, World")
    response.end()
});
```

BEWARE OF THIS: text/html

INPUT	OUTPUT

`const server = http.createServer((request, response) => {`
`  response.writeHead(200, {'Content-Type': 'text/plain'})`
`  response.write("Hello, World")`
`  response.end()`
`});`
	Hello, World

`const server = http.createServer((request, response) => {`
`  response.writeHead(200, {'Content-Type': 'text/html'})`
`  response.write("Hello, <b>World</b>")`
`  response.end()`
`});`
	Hello, **World**

Request.method()

- We can implement routes for different HTTP methods:

```
const http = require("http");
const server = http.createServer((req, res) => {
    if (req.method === "GET") {
        res.writeHead(200, {"Content-Type": "text/plain"});
        res.end("Received a GET request.");
    } else if (req.method === "POST"){
        res.writeHead(200, {"Content-Type": "text/plain"});
        res.end("Received a POST request.");
    } else {
        res.writeHead(404, {"Content-Type": "text/plain"});
        res.end("404 Not Found");
    }
})
```

Parsing a URL

- It refers to the process of **breaking down a URL** into its component parts for easier access and manipulation. This can be particularly useful when you need to extract specific parts of a URL, such as the protocol, hostname, path, query parameters, or fragment identifiers.

- We can use “url” and “querystring” to extract the URL and its parameters:

```
const url = require("url");
const querystring = require("querystring");

const server = http.createServer((req, res) => {
  const parsedUrl = url.parse(req.url);
  const queryParams = querystring.parse(parsedUrl.query);
});
```

- HTML Codes:

200 Series	400 Series	500 Series
• 200 OK: Request was successful, and the server has returned the requested data. • 201 Created: Request was successful, and a new resource has been created as a result.	• 400 Bad Request: Server cannot process the request due to a client error. • 401 Unauthorized: Client must authenticate itself to get the requested response. • 403 Forbidden: Client does not have permission to access the requested resource. • 404 Not Found: Requested resource could not be found on the server.	• 500 Internal Server Error: Server encountered a situation it doesn’t know how to handle. • 502 Bad Gateway: Received an invalid response. • 503 Service Unavailable: Server is currently unable to handle the request due to maintenance or overload. • 503 Gateway Timeout.

Express.js

- Framework for Node.js to simplify the process of building web applications.
- Characteristics:
    - Minimalistic and flexible.
    - Easy routing: offers a straightforward way to define routes for handling various HTTP requests.
    - Middleware support: middleware functions can be added to the request-response cycle to perform various tasks.

- Import express:

```
const express = require("express");
const app = express();
```

- We can use middleware to parse form data:
    - **express.urlencoded():** Middleware for parsing form data with 'Content-Type: application/x-www-form-urlencoded'.
    - **express.json():** Middleware for parsing JSON data with 'Content-Type: application/json'.

```
app.use(express.urlencoded({extended:false}));
app.use(express.json());
```