**Why Are We Learning Node.js?**

Node.js lets us:

1. **Build Websites and Apps**: Just like PHP or Python, but using JavaScript.
2. **Handle Many Users Quickly**: Node.js is great for tasks like chat apps, online games, and live data systems because it’s fast and can handle many users at the same time.
3. **Use One Language Everywhere**: If you know JavaScript, you can use it on both the **frontend** (browser) and **backend** (server).

Now, let’s build a server to see how Node.js works behind the scenes.

**Step 1: What is a Server?**

A **server** is like a restaurant waiter:

1. It **listens** to requests from customers (e.g., you ordering food).
2. It **responds** with the right answer (e.g., serving your food).

In web terms:

* When you visit a website, your browser sends a request to a server.
* The server decides what to send back (HTML, data, etc.).

**Why build a server with Node.js?**

* Other languages like PHP have built-in tools to create servers (Apache or Nginx).
* In Node.js, we control everything ourselves using JavaScript. This gives us more power and flexibility.

**Step 2: Setting Up a Simple Server**

**1. Create a File for the Server**

Let’s create a file named server.js.

**Why?** This file will hold our code to handle requests and send responses.

**2. Add Basic Server Code**

Write this in server.js:

const http = require('http'); // Import Node.js' built-in HTTP module.

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

// 1. Set headers to tell the browser what type of response to expect.

res.writeHead(200, { 'Content-Type': 'text/plain' }); // Status 200 means OK.

// 2. Send a response message to the browser.

res.end('Hello, this is your Node.js server!'); // This ends the response.

});

// 3. Tell the server to listen for requests on port 3000.

const PORT = 3000;

server.listen(PORT, () => {

console.log(`Server is running at http://localhost:${PORT}`);

});

**Why Are We Doing These Steps?**

1. **const http = require('http');**
   * This is like hiring a waiter (the HTTP module).
   * We’re telling Node.js: "Hey, I want to create a server."
2. **http.createServer**
   * This is the main function that creates the waiter (server) who will handle requests.
3. **res.writeHead(200, { 'Content-Type': 'text/plain' });**
   * **What is this?**
     + 200: Status code that means "Everything is OK."
     + Content-Type: text/plain: Tells the browser, "I’m sending plain text, not HTML or images."
   * **Why is it important?**
     + The browser needs to know what kind of response it’s getting (e.g., text, HTML, JSON).
4. **res.end('Hello...')**
   * **What is this?**
     + It’s like the waiter delivering your food and saying, "Here’s your order."
   * **Why is it important?**
     + Every request needs a response, or the browser will keep waiting forever.
5. **server.listen(3000)**
   * **What is this?**
     + Tells the server to start listening for requests on port 3000.
   * **Why is it important?**
     + Think of a port like a door to your restaurant. If it’s closed, no one can enter.

**3. Run the Server**

* Run this command in your terminal:

node server.js

* Open your browser and go to http://localhost:3000.

**You should see:**

Hello, this is your Node.js server!

**Comparison to PHP**

In PHP, the server setup is hidden because it relies on tools like Apache.

* PHP automatically handles headers and responses.
* In Node.js, we handle this ourselves, giving us more flexibility.

**Step 3: Handling Different Requests**

Right now, our server only responds with "Hello...". Let’s make it smarter by responding differently based on the URL.

**1. Update the Code:**

javascript

Copy code

const http = require('http');

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

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

res.writeHead(200, { 'Content-Type': 'text/plain' });

res.end('Welcome to the Home Page!');

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

res.writeHead(200, { 'Content-Type': 'text/plain' });

res.end('Welcome to the About Page!');

} else {

res.writeHead(404, { 'Content-Type': 'text/plain' });

res.end('Page not found.');

}

});

const PORT = 3000;

server.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}`);

});

**Why Are We Doing This?**

1. **req.url**
   * This checks what the user is asking for (e.g., /, /about).
   * In PHP, routing is often handled automatically by frameworks like Laravel.
   * In Node.js, we control the routing manually, giving us more power.
2. **Responding Based on the URL**
   * If someone goes to /, we show the home page message.
   * If they go to /about, we show the about page message.
   * For any other page, we show a 404 error.

Use Node.js' fs module to read the HTML files:

* + const http = require('http');
  + const fs = require('fs'); // File System module.
  + const server = http.createServer((req, res) => {
  + if (req.url === '/') {
  + fs.readFile('./views/home.html', (err, data) => {
  + if (err) {
  + res.writeHead(500, { 'Content-Type': 'text/plain' });
  + res.end('Internal Server Error');
  + } else {
  + res.writeHead(200, { 'Content-Type': 'text/html' });
  + res.end(data); // Send the HTML content.
  + }
  + });
  + } else if (req.url === '/about') {
  + fs.readFile('./views/about.html', (err, data) => {
  + if (err) {
  + res.writeHead(500, { 'Content-Type': 'text/plain' });
  + res.end('Internal Server Error');
  + } else {
  + res.writeHead(200, { 'Content-Type': 'text/html' });
  + res.end(data);
  + }
  + });
  + } else {
  + res.writeHead(404, { 'Content-Type': 'text/html' });
  + res.end('<h1>404 Not Found</h1><p>The page you are looking for does not exist.</p>');
  + }
  + });
  + const PORT = 3000;
  + server.listen(PORT, () => {
  + console.log(`Server running at http://localhost:${PORT}`);
  + });