**1. process.env & process.env.PORT**

* process.env → an object in Node.js that stores environment variables.
* process.env.PORT → a specific environment variable usually used to define which port your app should run on (useful when deploying to platforms like Heroku, where the port is auto-assigned).

Example:

const express = require("express");

const app = express();

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

👉 If process.env.PORT = 5000, server runs on **5000**. If not set, it defaults to **3000**.

**2. Difference Between GET and POST**

* **GET**
  + Used to **retrieve data** from the server.
  + Parameters are sent in the **URL** (query string).
  + Example: /users?name=bhanu.
* **POST**
  + Used to **send data** to the server.
  + Data is sent in the **body** of the request (more secure).
  + Example: Sending login details (username, password).

**Example in Express.js:**

const express = require("express");

const app = express();

app.use(express.urlencoded({ extended: true }));

app.use(express.json());

// GET example

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

res.send("This is a GET request. Example: Fetching all users.");

});

// POST example

app.post("/users", (req, res) => {

const { name, email } = req.body;

res.send(`User created: Name = ${name}, Email = ${email}`);

});

app.listen(3000, () => console.log("Server running on port 3000"));

**Output:**

* GET http://localhost:3000/users →  
  👉 Response: "This is a GET request. Example: Fetching all users."
* POST http://localhost:3000/users with body { "name": "Bhanu", "email": "test@gmail.com" } →  
  👉 Response: "User created: Name = Bhanu, Email = test@gmail.com"

**3. Purpose of Routing in Express.js**

* Routing means **defining how an application responds** to client requests (GET, POST, PUT, DELETE) at specific endpoints.
* Helps in organizing APIs.

**Example route:**

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

res.send("Here is the list of users");

});

👉 Visiting http://localhost:3000/users → "Here is the list of users"

**4. Handling Dynamic Data in URL**

* :id → called a **route parameter**.
* Example: /users/101 → here 101 is dynamic.

**Example:**

app.get("/users/:id", (req, res) => {

const userId = req.params.id; // from URL

res.send(`User ID is: ${userId}`);

});

👉 Visiting http://localhost:3000/users/101 → "User ID is: 101"

**Difference between req.params and req.query:**

* **req.params**
  + Used for **URL parameters** (defined in route).
  + Example: /users/101 → req.params.id = 101.
* **req.query**
  + Used for **query string parameters**.
  + Example: /search?name=bhanu&age=21 →  
    req.query.name = "bhanu"  
    req.query.age = "21"

👉 Use **params** for identifying a specific resource (like userId).  
👉 Use **query** for filtering or searching.

**6. Keywords**

* **public** → usually a folder in Express apps for serving static files (CSS, JS, images).
* app.use(express.static("public"));

👉 Then you can access public/style.css as http://localhost:3000/style.css.

* **name and email** → fields in an HTML form.
* <form action="/users" method="POST">
* <input type="text" name="name" />
* <input type="email" name="email" />
* <button type="submit">Submit</button>
* </form>
* **express.urlencoded()**
  + Middleware to parse form data (application/x-www-form-urlencoded).
* app.use(express.urlencoded({ extended: true }));
* **fetch()**
  + JavaScript function used in the frontend to call backend APIs.
* fetch("/users", {
* method: "POST",
* headers: { "Content-Type": "application/json" },
* body: JSON.stringify({ name: "Bhanu", email: "test@gmail.com" })
* })
* .then(res => res.text())
* .then(data => console.log(data));