1. **Understanding Node.js Environment Steps Performed**

1. Installed Node.js.
2. Verified installation using:

```
node -v
npm -v
```

3. Created folder `practice-node`.
4. Initialized npm using:

```
npm init -y
```

Explanation

1. package.json

- Stores project metadata.
- Contains project name, version, dependencies, scripts.
- Automatically created after npm initialization.

2. node_modules

- Contains installed external packages.
- Created after running `npm install`.
- Should not be shared in version control.

3. dependencies

- Lists external libraries required for the project.
- Stored inside package.json.
- Installed using `npm install package-name`.

2: Create Basic Node Server (Without Express)

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

const server = http.createServer((req, res) => {
    res.write("This is a Node server");
    res.end();
});

server.listen(3000, () => {
    console.log("Server running on port 3000");
});
```

Output:

```
Server running on port 3000

Time limit exceeded
```

3: Install and Explore Express

Steps

1. Created folder express-practice.
2. Initialized npm.
3. Installed Express:

npm install express

Observations

- package.json updated with express dependency.
- node_modules folder created automatically.

What is Express?

Express is a web framework for Node.js used to simplify:

- Server creation
- Routing
- Middleware handling
- API development

Why is Express Used?

- Reduces code complexity
- Simplifies routing
- Supports middleware
- Ideal for building REST APIs

4: Create Simple Routes

Code

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

app.get('/', (req, res) => {
   res.send("Home Page");
});

app.get('/contact', (req, res) => {
   res.send("Contact Page");
});

app.get('/help', (req, res) => {
   res.send("Help Page");
});

app.listen(3000);
```

Observations

- Undefined routes show: Cannot GET /route
- Express differentiates routes using:
    - HTTP method (GET, POST)
    - URL path

```
Output

ERROR!
/tmp/xdtSbgTc3T/main.js:20
Undefined routes show: Cannot GET /route
          ^^^^^^
```

5: Return JSON Data

Code

```
app.get('/api/user', (req, res) => {
   res.json({
      name: "John",
      age: 20,
      course: "Computer Science"
```

```
    });
});
```

JSON Format

Example:

```
{
  "name": "John",
  "age": 20
}
```

JSON vs Plain Text

JSON	Plain Text
Structured	Unstructured
Machine-readable	Mainly human-readable
Used in APIs	Used for display

6: Middleware Practice

Code

```
app.use((req, res, next) => {
    console.log("Method:", req.method);
    console.log("URL:", req.url);
    next();
});
```

Observations

- Middleware runs for every request.
- Executes before route handler.
- Flow:

Client → Middleware → Route → Response

7: nodemon Practice

Installation

npm install --save-dev nodemon

Run Command

npx nodemon server.js

Difference Between node and nodemon

node	nodemon
Manual restart	Auto restart
Used in production	Used in development

Importance

- Saves development time.
- Automatically reloads server after changes.

8: 404 Handling

Code

```
app.use((req, res) => {
   res.status(404).send("Page Not Found");
});
```

Importance of 404 Handling

- Improves user experience.
- Displays proper error messages.
- Prevents confusion.
- Essential in backend development.

9: Request–Response Analysis

What Happens When User Types:

localhost:3000/

Step-by-Step Process

1. Browser sends HTTP request.
2. Server receives request.
3. Middleware executes.

4. Express checks matching route.
5. Route handler runs.
6. Server sends response.
7. Browser displays result.

Definitions

Request

A message sent from client to server containing:

- HTTP method
- URL
- Headers
- Body (optional)

Response

A message sent from server to client containing:

- Status code
- Headers
- Data (HTML/JSON/Text)

Middleware Execution

Middleware executes:

- After request is received
- Before response is sent
- In the order defined in the code