DAY-10

Node.js Modules and NPM Package Management

- ***What Are Node.js Modules?***
  - Think of modules in Node.js as building blocks or tools that help you do specific tasks in your code. Some modules come with Node.js by default (called core modules), and others can be downloaded from the internet (called third-party modules).
    1. **Core Modules:** These are built into Node.js, so you don't need to install them. They let you do things like:
       - Create a web server (http module)
       - Work with files (fs module)
       - Work with paths (path module)

<For example, if you want to create a simple web server, you can use the http module that's already included in Node.js.>

2. **Third-Party Modules:** These are modules created by other developers. For example, if you want to make your life easier by using a web framework like Express (a popular tool for building web servers in Node.js), you can install it using NPM (which we'll explain shortly).

3. **Local Modules:** These are modules you create yourself. You can write your own code in separate files and reuse it in different parts of your project.

- ***How Do You Use Node.js Modules?***
  - To use a module in your code, you simply require it. Here's an example of using the built-in http module to create a basic web server:

- ***Java script File:***

```
// Load the built-in http module
const http = require('http');

// Create a simple server that sends a message when accessed
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello from Node.js!');
});
// Make the server listen on port 3000
server.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

DAY-10

- ***What is NPM?***
  - NPM stands for Node Package Manager, and it's like a giant store where you can find tools (called packages or modules) that help make your programming tasks easier.
  - You can install packages that do things like help you build websites faster, connect to databases, manage users, and much more.
  - When you install Node.js, you also get NPM installed automatically.

How to Use NPM?

1. **Start a Project:**
   - Before using NPM, you need to create a project. When you start a Node.js project, you create a file called package.json that keeps track of all the packages you install.
   - To create this file, open your terminal (a command window) and run this command:
     npm init -y

2. **Installing Packages:**
   - When you need a new tool or library, you can install it with NPM. For example, if you want to install Express (a framework that helps you create web servers quickly), you would run:
     npm install express

3. **Using Installed Packages:**
   - After installing a package like Express, you can use it in your code by requiring it. Here's an example:

- ***Java script File:***

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

app.get('/', (req, res) => {
  res.send('Hello, Express!');
});

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

DAY-10

- ***NPM Commands You Should Know:***

– **Here are some useful commands you'll use with NPM:**

1. **Install a Package:**
   `npm install <package-name>`

   Example: npm install express to install the Express package.

2. **Uninstall a Package:**
   `npm uninstall <package-name>`

   Example: npm uninstall express to remove the Express package.

3. **Run Your Project:** If you have a script (like a start-up script in package.json), you can run it with:
   `npm start`

4. **List Installed Packages:** If you want to see what packages you've installed, run:
   `npm list`

- ***Conclusion:***

  - **Node.js Modules** are like tools that help you do specific tasks in your code (like creating a web server or handling files).
  - **NPM (Node Package Manager)** is a tool that lets you easily install and manage packages (tools) created by other developers. It helps make your work easier by letting you add new features without having to build everything yourself.
  - With NPM, you can install packages, use them in your project, and manage dependencies all with a few simple commands.

DAY-10

Project: To-Do List Manager

***Step 1*: Set Up the Project**

1. ***Create a new folder:***

```
mkdir To-Do List Manager
cd To-Do List Manager
```

2. ***Initialize a new Node.js project:***

```
npm init -y
```

– This creates a package.json file.

3. ***Install Required Packages:***

```
npm install yargs chalk fs-extra
```

4. ***Use an Older Version of chalk:***

```
npm install chalk@4
```

***Step 2*: Set Up the File Structure**

```
todo-list/
├── package.json
├── index.js      # Entry point
├── tasks.js      # Helper functions
└── data/
    └── tasks.json  # Task storage file
```

Step 3*: *data/tasks.json: Create an empty file to store tasks.

```
[]
```

DAY-10

***Step 4*: Add the Code: (index.js)**

```
const yargs = require("yargs");
const chalk = require("chalk");
const { addTask, listTasks, deleteTask, markTaskComplete } = require("./tasks");

// Add Command
yargs.command({
 command: "add",
 describe: "Add a new task",
 builder: {
  title: { describe: "Task title", demandOption: true, type: "string" },
  description: { describe: "Task description", type: "string" },
 },
 handler(argv) {
  addTask(argv.title, argv.description);
 },
});

// List Command
yargs.command({
 command: "list",
 describe: "List all tasks",
 handler() {
  listTasks();
 },
});

// Delete Command
yargs.command({
 command: "delete",
 describe: "Delete a task by ID",
 builder: {
  id: { describe: "Task ID", demandOption: true, type: "number" },
 },
 handler(argv) {
  deleteTask(argv.id);
 },
});

// Complete Command
yargs.command({
```

DAY-10

```
  command: "complete",
  describe: "Mark a task as completed",
  builder: {
    id: { describe: "Task ID", demandOption: true, type: "number" },
  },
  handler(argv) {
    markTaskComplete(argv.id);
  },
});

// Parse Commands
yargs.parse();
```

***Step 5*: Add the Code: (tasks.js)**

```
const fs = require("fs-extra");
const chalk = require("chalk");

const dataPath = "./data/tasks.json";

// Load tasks from JSON file
const loadTasks = () => {
  try {
    return fs.readJSONSync(dataPath);
  } catch {
    return [];
  }
};

// Save tasks to JSON file
const saveTasks = (tasks) => {
  fs.writeJSONSync(dataPath, tasks);
};

// Add a new task
const addTask = (title, description = "") => {
  const tasks = loadTasks();
  tasks.push({ id: tasks.length + 1, title, description, completed: false });
  saveTasks(tasks);
  console.log(chalk.green("Task added successfully!"));
};
```

DAY-10

```
// List all tasks
const listTasks = () => {
  const tasks = loadTasks();
  console.log(chalk.blue("Your Tasks:"));
  tasks.forEach((task) =>
    console.log(
      `${chalk.yellow(`ID: ${task.id}`)} - ${chalk.bold(task.title)} [${
        task.completed ? chalk.green("Completed") : chalk.red("Pending")
      }]`
    )
  );
};

// Delete a task by ID
const deleteTask = (id) => {
  const tasks = loadTasks();
  const filteredTasks = tasks.filter((task) => task.id !== id);
  if (filteredTasks.length === tasks.length) {
    console.log(chalk.red("Task not found!"));
    return;
  }
  saveTasks(filteredTasks);
  console.log(chalk.red("Task deleted successfully!"));
};

// Mark a task as completed
const markTaskComplete = (id) => {
  const tasks = loadTasks();
  const task = tasks.find((t) => t.id === id);
  if (task) {
    task.completed = true;
    saveTasks(tasks);
    console.log(chalk.green("Task marked as completed!"));
  } else {
    console.log(chalk.red("Task not found!"));
  }
};

module.exports = { addTask, listTasks, deleteTask, markTaskComplete };
```

DAY-10

***Step 6*: Run and Test:**

1. ***Add a Task***

`node index.js add --title="Learn Node.js" --description="Complete basics"`

- ***Output:***

```
Admin@BLACK-DELL MINGW64 /e/file/jira/DAY-10 Node.js Modules and NPM Package Management/To-Do List Manager
$ node index.js add --title="Learn Node.js" --description="Complete basics"
Task added successfully!
```

2. ***List Tasks***

`node index.js list`

- ***Output:***

```
Admin@BLACK-DELL MINGW64 /e/file/jira/DAY-10 Node.js Modules and NPM Package Management/To-Do List Manager
$ node index.js list
Your Tasks:
ID: 1 - Learn Node.js [Pending]
ID: 2 - Learn Node.js [Pending]
```

3. ***Add Another Task***

`node index.js add --title="Practice JavaScript" --description="Array methods"`

- ***Output:***

```
Admin@BLACK-DELL MINGW64 /e/file/jira/DAY-10 Node.js Modules and NPM Package Management/To-Do List Manager
$ node index.js add --title="Learn Node.js" --description="Complete basics"
Task added successfully!

Admin@BLACK-DELL MINGW64 /e/file/jira/DAY-10 Node.js Modules and NPM Package Management/To-Do List Manager
$ node index.js list
Your Tasks:
ID: 2 - Learn Node.js [Pending]
ID: 2 - Practice JavaScript [Pending]
ID: 3 - Learn Node.js [Pending]
```

4. ***Mark a Task as Completed:***

`node index.js complete --id=2`

- ***Output:***

```
Admin@BLACK-DELL MINGW64 /e/file/jira/DAY-10 Node.js Modules and NPM Package Management/To-Do List Manager
$ node index.js complete --id=1
Task marked as completed!
```

DAY-10

5. ***Delete a Task:***

```
node index.js delete --id=2
```

- ***Output:***

```
Admin@BLACK-DELL MINGW64 /e/file/jira/DAY-10 Node.js Modules and NPM Package Management/To-Do List Manager
$ node index.js delete --id=2
Task deleted successfully!
```