Step8Up Academy

NODE PACKAGE MANAGER (NPM) AND TEST DRIVEN DEVELOPMENT (TDD)

FULL STACK SKILLS BOOTCAMP

AN INTRODUCTION TO NPM, AUTOMATED TESTING, AND BEST PRACTICES

- **Lesson Overview:**
- In this lesson, we will be introduced to:

1. Node Package Manager (NPM)
2. Test Driven Development (TDD)
3. Testing JavaScript with Jest
4. Mocking with Jest
5. Excluding content from GitHub (gitignore)

WHAT IS NPM?

- **Definition**: NPM is the default package manager for Node.js.
- **Key Features**:
  - Allows installation and management of third-party libraries and tools.
  - Includes tools for dependency management and running scripts.

START COMMANDS AND RUNNING A SIMPLE SCRIPT

- **Scripts in NPM**: Define custom commands for your application in the package.json file.
- **Example**:

```
"scripts": {
  "start": "node index.js"
}
```

How to Run:

```
npm start
```

INSTALLING PACKAGES FOR PRODUCTION AND DEVELOPMENT

- **Production vs Development**:
  - **Production dependencies**: Essential packages required for running the app.
    - Command: npm install <package>
  - **Development dependencies**: Tools needed for development and testing, not for production.
    - Command: npm install --save-dev <package>
- **Example**:
  - express for production.
  - jest for testing in development.

  Demo…

WHAT IS TDD (TEST DRIVEN DEVELOPMENT)?

- **Definition**:

  A software development process where tests are written before the actual code.

- **Workflow**:
  - Write a test.
  - Run the test (it should fail).
  - Write the minimal amount of code to make the test pass.
  - Refactor and repeat.

REAL WORLD APPLICATIONS FOR TDD

- **Key Benefits**:
  - Ensures code correctness before implementation.
  - Facilitates debugging and refactoring.
  - Enhances code reliability in applications with critical operations (e.g., payment gateways, APIs).
- **Common Use Cases**:

  API development, legacy system updates, user authentication.

INTRODUCING JEST FOR JAVASCRIPT

- **What is Jest?**:

  A JavaScript testing framework, commonly used for unit and integration testing.

- **Key Features**:

  Easy to set up.

  Built-in assertions.

  Supports mocking and spying.

TESTING EXCEPTIONS IN JEST

- Example of Testing an Error

```
function compileCode() {
  throw new Error('Compilation Error');
}

test('compiling code throws an error', () => {
  expect(() => compileCode()).toThrow('Compilation Error');
});
```

MOCKS AND SPYON IN JEST

- **What are Mocks?:**

  Simulate the behavior of real functions in tests.**Example with jest.fn()**

```
const mockCallback = jest.fn(x => x + 1);
expect(mockCallback(2)).toBe(3);
```

- **spyOn Example**: Track calls to a method

  demo...

```
const calculator = {
  add: (a, b) => a + b
};
const spy = jest.spyOn(calculator, 'add');
calculator.add(2, 3);
expect(spy).toHaveBeenCalledWith(2, 3);
```

GITIGNORE: HOW TO EXCLUDE FILES AND FOLDERS FROM A REPOSITORY

- **Purpose of .gitignore**:

  Prevent certain files or folders from being tracked by Git.

- **Typical Example**

```
node_modules/
.env
```

REAL WORLD EXAMPLES OF .GITIGNORE

- **Common Scenarios**:

  Node.js projects: Exclude node_modules/.

  Environment files: Exclude .env to protect sensitive information.

  Log files: Prevent log files from cluttering the repository.

CONCLUSION

- The importance of using NPM for dependency management.
- How TDD improves software quality and maintainability.
- The role of Jest in writing efficient, testable code.
- Proper usage of .gitignore to manage project files in Git.

Step8Up Academy

QUESTIONS?