What is hooks?

In Mocha, hooks are functions that are executed before or after tests run. They allow you to set up preconditions (like initializing variables, setting up test data, or starting services) and clean up resources (like closing connections, resetting data, or stopping services) during your testing process. The hooks in Mocha follow a **Behavior-Driven Development (BDD)** style, which is designed to make the tests more readable and intuitive.

Note: Mocha provides the hooks before(), after(), beforeEach(), and afterEach(). These should be used to set up preconditions and clean up after your tests.

### before()

**Purpose**: The before() hook is used to run a piece of code once before all the tests in a test suite.

**Use case**: If you need to set up something globally for your tests (like initializing a database connection, setting environment variables, or loading configurations), this is the place to do it.

### after()

**Purpose**: The after() hook runs once after all the tests in a test suite have completed. This is used for clean-up tasks like closing connections, clearing data, or stopping services.

**Use case**: If you need to perform any cleanup, such as disconnecting from a database or deleting files after the tests are done, you use the after() hook.

### beforeEach()

**Purpose**: The beforeEach() hook runs before each individual test. This is useful for setting up conditions that need to be fresh for every test case, ensuring that each test starts with the same state.

**Use case**: You might want to reset or initialize specific data, or perform actions that should occur before each test runs, such as clearing a cache or setting up mocks.

### afterEach()

**urpose**: The afterEach() hook runs after each individual test. It's used to clean up anything that was set up in the beforeEach() or during the test itself, ensuring a clean slate for the next test.

**Use case**: If you need to clear out data, stop timers, or undo changes made during each test, you would use this hook.

Node: Tests can appear before, after, or interspersed with your hooks. Hooks will run in the order they are defined, as appropriate; all before() hooks run (once), then any beforeEach() hooks, tests, any afterEach() hooks, and finally after() hooks (once).

### Describing Hooks

Any hook can be invoked with an optional description, making it easier to pinpoint errors in your tests. If a hook is given a named function, that name will be used if no description is supplied.