Mocks

Terms

Mock functions
Partial mocking
Spies

Summary

- *Mock functions* simulate the behavior of real functions, allowing you to control their output and behavior during testing. They are useful for isolating specific code paths and testing functions in isolation.

- Mock functions can be created using **vi.fn().**

- You can mock an entire module and replace all exported functions with mocks using **vi.mock().**

- *Partial mocking* is valuable for mocking parts of a module while retaining some of its original behavior.

- *Function spying* allows you to monitor and record calls to functions during testing. They're useful for tracking function invocations and arguments without modifying their behavior.

- Mocks should be cleared before or after each test to ensure a clean slate for subsequent tests.

- While mocks are valuable for isolating and testing specific units, they can result in tests becoming tightly coupled to implementation details. Such tests may become fragile and break as the implementation evolves. Therefore, use mocks primarily to replace external dependencies that may be unavailable or slow during test execution, such as databases, the file system, APIs, etc.

- Tests should not be dependent on the current date and time as this can lead to different results during different test runs. Mocking dates and times is useful when testing time-sensitive logic to maintain result consistency.

Creating Mock Functions

```
const greet = vi.fn();
greet.mockReturnValue(‘hello’);
greet.mockResolvedValue(‘hello’);
greet.mockRejectedValue(‘error’);
greet.mockImplementation(name => ‘hello ‘ + name);
```

Assertions

```
expect(greet).toHaveBeenCalled();
expect(greet).toHaveBeenCalledOnce();
expect(greet).toHaveBeenCalledWith(‘mosh’);
```

Mocking Modules

```
vi.mock(‘../src/currency’);
vi.mocked(mockedFunc).mockReturnValue(0);
```

Clearing Mocks

```
vi.mocked(mockedFunc).mockClear();
vi.clearAllMocks();
```

Mocking Dates

```
vi.setSystemTime(‘2024-01-01 10:00’);
```