Vitest Mocking Cheat Sheet by Artem Zakharchenko

EPICWEB.DEV

Functions

Mock functions

```
// Create a mock function that records its calls.
const listener = vi.fn()
emitter.on('hello', listener)
emitter.emit('hello', 'John')

expect(listener).toHaveBeenCalledWith('John')
```

Spies

```
// Spy on any method of the target object,
// recording its calls as if it's a mocked function.
vi.spyOn(target, 'method')

expect(target.method).toHaveBeenCalledTimes(3)
```

Mock implementation

```
// Replace the implementation of any mock function.
const mock = vi.spyOn(global, 'console')
              .mockImplementation(() => {})

// Reset this mock implementation to be an empty function.
// Note: this still KEEPS the mock active.
mock.resetMock()

// Restore the original implementation.
// Note: this REMOVES the mock altogether.
mock.mockRestore()

// Or restore ALL mock functions at once.
vi.restoreAllMocks()
```

Modules

```
// Hoist the values and methods you want to use in mocked modules.
const hoistedValue = vi.hoisted(() => 123)

// Mock any module by its specifier (name).
// Tip: use a dynamic `import` to keep this mock type-safe!
vi.mock(import('module-specifier'), async (importOriginal) => {
  // Get the original exports of the mocked module, if needed.
  const originalExports = await importOriginal()

  // Return a mocked exports object.
  return {
    // Access hoisted values.
    numbers: hoistedValue,
  }
})
```

Date and Time

Date / Timers

When testing code that depends on `Date` or timers (`setTimeout`/`setInterval`), you have to mock the date to make its value (and your tests) predictable.

```
// First, instruct Vitest that this test is mocking date.
vi.useFakeTimers()

// Set any Date as the current date (i.e. Date.now()).
// 💡 Include an explicit timezone to have consistent mocked dates.
// Below, I'm including "Z" to state this is a UTC date.
vi.setSystemTime(new Date('2024-12-31T01:21:33.000Z'))

// Advance the time in tests by 250ms.
vi.advanceTimersByTime(250)

// Run all the `setTimeout`/`setInterval` timers in tests.
// Useful when you have logic that executes later.
vi.runAllTimers()

// Once you're done, restore the date and time to its normal state.
vi.useRealTimers()
```

Queued Tasks

```
// Provide "queueMicrotask" to the "toFake" property
// to tell Vitest that this test is mocking ticks and tasks.
vi.useFakeTimers({ toFake: ['queueMicrotask'] })

// Run all the logic scheduled for later via APIs like
// `queueMicrotask` or `process.nextTick`.
vi.runAllTicks()
```

Network

Mocked responses

```
import { http } from 'msw'

// Intercept a "GET /user" request.
http.get('/user', () => {
  // Return any Fetch API Response to be used
  // as the mock response for the intercepted request.
  return Response.json({ id: 1, name: 'Kody' })
})
```

Network errors

```
import { http } from 'msw'

http.get('/user', () => {
  // Respond with a network error.
  // Note: This REJECTS the response promise.
  return Response.error()
})
```

Network

Response delay

```
import { http, delay } from 'msw'

http.post('/post/:postId', async () => {
  // Await a random realistic server response time.
  await delay()

  // Await 250ms sharp.
  await delay(250)

  // Keep this request pending forever.
  await delay('infinite')
})
```

Globals

Global methods

```
// Spy and/or mock the implementation of any global method.
const consoleMock = vi.spyOn(global.console, 'log')

// Restore the original value of this mocked method.
consoleMock.mockRestore()

// Or restore all mocked global methods at once.
vi.restoreAllMocks()
```

Global values

```
// Stub any global value by its property name.
vi.stubGlobal('location', new
URL('http://localhost/dashboard')

// Restore any mocked global values at once.
vi.unstubAllGlobals()
```

Environment variables

```
// Mock the value of any environment variable by its name.
vi.stubEnv('SHOW_CUSTOMER_DISCOUNT', '1')

// Restore any mocked environment variables at once.
vi.unstubAllEnvs()
```