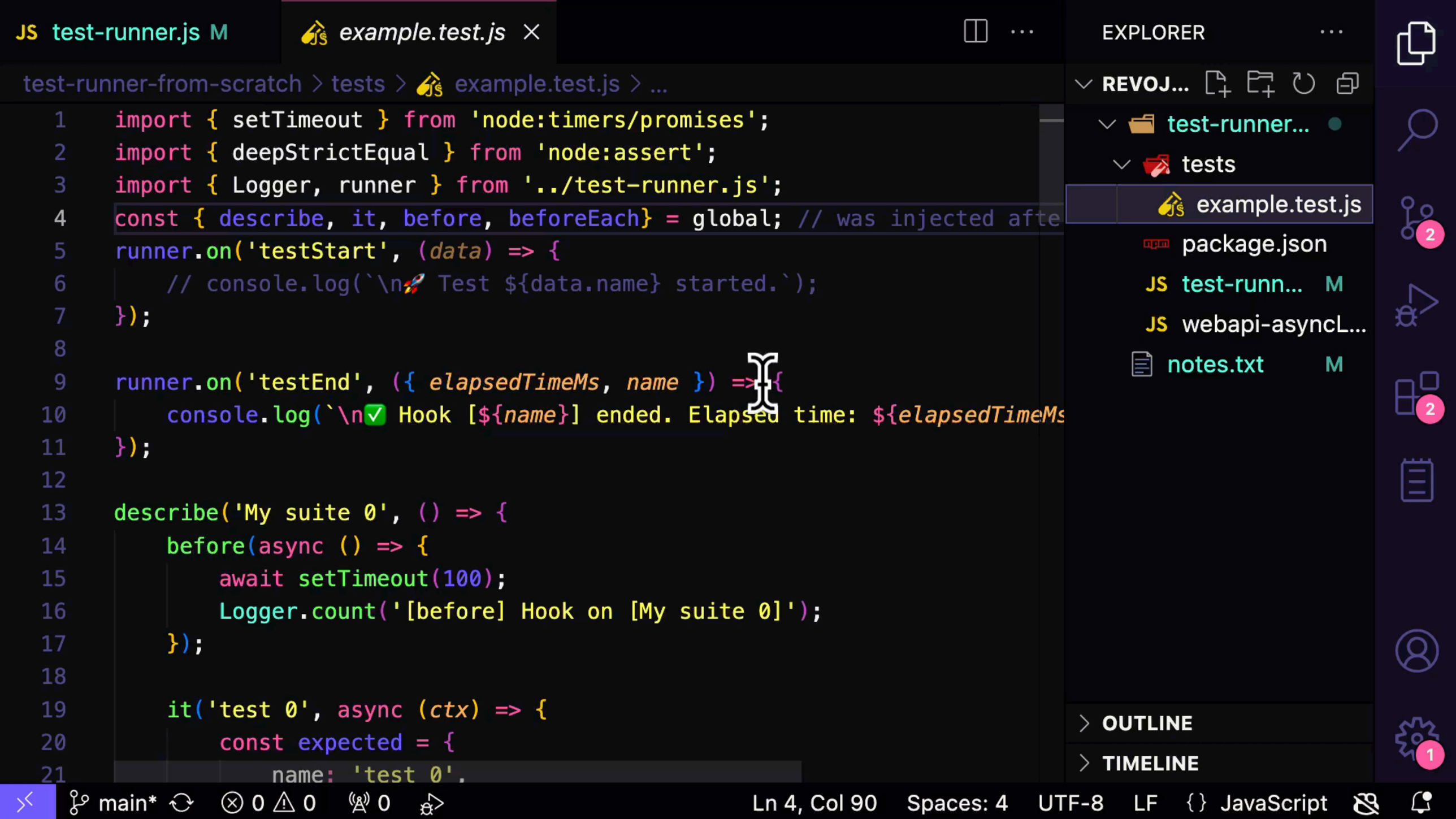
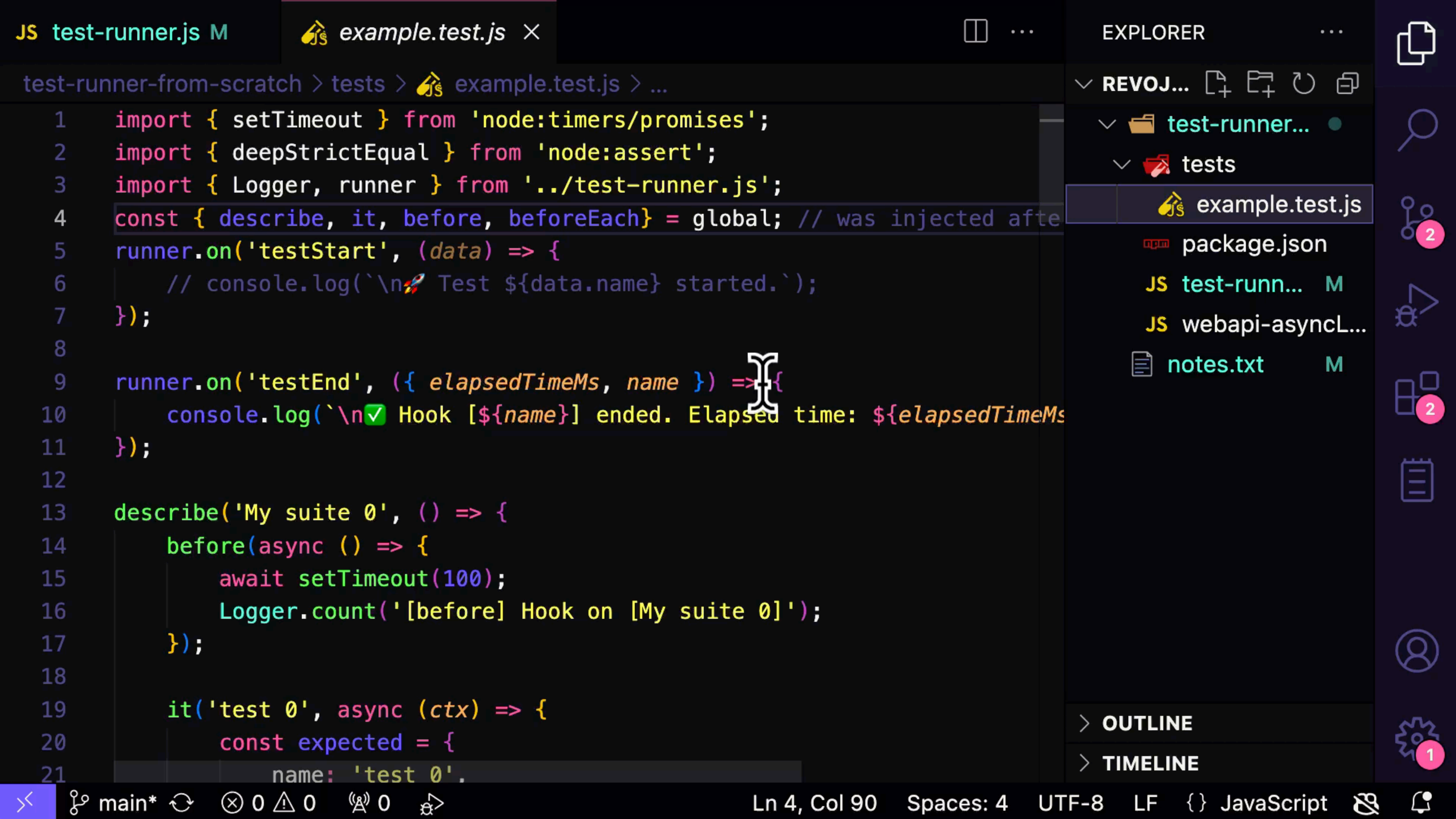

JS test-runner.js M
example.test.js
test-runner-from-scratch > tests > example.test.js > ...
1 import { setTimeout } from 'node:timers/promises';
2 import { deepStrictEqual } from 'node:assert';
3 import { Logger, runner } from '../test-runner.js';
4 const { describe, it, before, beforeEach} = global; // was injected afte
5 runner.on('testStart', (data) => {
6 // console.log(`\n🚀 Test ${data.name} started.`);
7 });
8
9 runner.on('testEnd', ({ elapsedTimeMs, name }) => {
10 console.log(`\n✅ Hook [${name}] ended. Elapsed time: ${elapsedTimeMs
11 });
12
13 describe('My suite 0', () => {
14 before(async () => {
15 await setTimeout(100);
16 Logger.count('[before] Hook on [My suite 0]');
17 });
18
19 it('test 0', async (ctx) => {
20 const expected = {
21 name: 'test 0',
EXPLORER
REVOJ...
test-runner...
tests
example.test.js
package.json
JS test-runn... M
JS webapi-asyncL...
notes.txt M
OUTLINE
TIMELINE
main*
0 0
0
Ln 4, Col 90
Spaces: 4
UTF-8
LF
JavaScript


JS test-runner.js M
example.test.js
test-runner-from-scratch > tests > example.test.js > ...
1 import { setTimeout } from 'node:timers/promises';
2 import { deepStrictEqual } from 'node:assert';
3 import { Logger, runner } from '../test-runner.js';
4 const { describe, it, before, beforeEach} = global; // was injected afte
5 runner.on('testStart', (data) => {
6 // console.log(`\n🚀 Test ${data.name} started.`);
7 });
8
9 runner.on('testEnd', ({ elapsedTimeMs, name }) => {
10 console.log(`\n✅ Hook [${name}] ended. Elapsed time: ${elapsedTimeMs
11 });
12
13 describe('My suite 0', () => {
14 before(async () => {
15 await setTimeout(100);
16 Logger.count('[before] Hook on [My suite 0]');
17 });
18
19 it('test 0', async (ctx) => {
20 const expected = {
21 name: 'test 0',
EXPLORER
REVOJ...
test-runner...
tests
example.test.js
package.json
JS test-runn... M
JS webapi-asyncL...
notes.txt M
OUTLINE
TIMELINE
main*
0 0
0
Ln 4, Col 90
Spaces: 4
UTF-8
LF
JavaScript

@erickwendel_

Home
Testing with TAP
Producers
Consumers
Specification

Test Anything Protocol

TAP, the Test Anything Protocol, is a simple text-based interface between testing modules in a test harness. It decouples the reporting of errors from the presentation of the reports.

https://testanything.org/