

Recipes for Success with Unit Testing

Prepared by Yisroel Yakovson of NoStack

This document (and a lot more) is at the [class repo](#).

Join our [Discord Server/Community](#) to ask any questions!

[Monitored for two weeks from the date of the class.]

Guide

Assertions	
For correct input	result-is 2
For faulty input	error-assert 3

	PUBLIC	PRIVATE
To TEST it...	test 2	test-private 4
To STUB it...	stub 5	stub-private 6

Administrative	
To create a project	1
To check coverage	(nyc) 7

Setup

1. Install [node](#), [npm](#)
2. Install editor (recommended: [Visual Studio Code](#))
3. Add [Snippets for the recipes](#)

Recipes

1. Create a project

** Prerequisite: node installed*

[A] Initiate project in directory

```
mkdir $NAME  
cd $NAME  
npm init -y  
npm init ava  
npm i -D typescript ts-node  
mkdir src  
mkdir test
```

[B] Add test runner config (AVA settings in package.json)

```
"ava": {  
  "files": [
```

```
"test/**/*.test.ts"  
],  
"extensions": [  
  "ts"  
],  
"require": [  
  "ts-node/register"  
]  
},
```

[C] Add language config (tsconfig.json)

```
{  
  "compilerOptions": {  
    "declaration": true,  
    "importHelpers": true,  
    "module": "commonjs",  
    "outDir": "lib",  
    "rootDirs": [". /src", ". /test"],  
    "strict": true,  
    "target": "es2017"  
  },  
  "include": [  
    "src/**/*.ts"  
]  
}
```

2. Test a public function (Snippets: test/result-is)

[A] Create a file **test/**/<name>.test.ts**

[B] Add the test runner and any other necessary libraries (such as an assertion library and plugins)

```
import test from 'ava'
```

[C] Create a test

```
test('description', t => {  
  })
```

[D] Add calls and assertions inside function e.g.

```
const result = func(a)  
t.is(result, x);
```

3. Test a thrown error (Snippet: throws-assert)

[A] Assign a variable to t.throws()*

```
const error = t.throws(() => {  
  fn();  
})
```

[B] Create a test

```
t.regex(error.message, /$regex$/);
```

*Note the *async* version of (a):

```
const error = await t	throwsAsync(  
  async () => {  
    await fn();  
  })
```

4. Test a private function (Snippet: test-private)

[A] Include rewire and use for module

```
const rewire = require('rewire')  
let $MODULE$ = rewire('$PATH$')
```

[B] Declare a stub for testing

```
const $FUNCTION$Stub =  
app.__get__('$FUNCTION$')
```

[C] Run the stub in a normal test, e.g.

```
const result = $FUNCTION$Stub()  
...
```

SAMPLE TEST FILE

```
// test of a()...
```

```
const sample = require('../src/sample')
const {a} = sample.a
const result = a(1)
t.is(result, 4)

// test of b()...
const rewire = require('rewire');
const sample = rewire('../src/sample') // [a]
const b = sample.__get__('b') // [b]
const result = b(1) // [c]
t.is(result, 2)
```

SAMPLE SRC FILE sample.ts

```
function b(x) {
  return x + 1
}

export function a(x) {
  return b(x) * 2
}
```

5. Stub function (Snippet: stub)

[A] Declare the sandbox, module and fake function

```
const sinon = require('sinon')
let sandbox = sinon.createSandbox()

const $MODULE$ = require('$PATH$')
const $FUNC$Fake = ()=> {
  // replacement code here...
}
```

[B] Declare the stub* [possibly in *beforeEach*]

```
const newStub = sandbox.stub(
  $MODULE$, '$FUNC$'
).callsFake($FUNC$Fake)
```

Run test(s) on a function that calls

[C] Reset sandbox [possibly in *afterEach*]

```
sandbox.restore()
```

SAMPLE TEST FILE

```
// stub slowFunction
```

```
const sinon = require('sinon');
```

```
let sandbox = sinon.createSandbox() [a]
```

```
[b]
```



```
const someModule = require('someModule');  
const fake = ()=> 'abc123' [c]
```

```
sandbox.stub(someModule, 'slowFunction')  
  .callsFake(slowFunctionFake) [d]
```

```
// test using the stub [e]
```

```
const result = a(1)  
t.is(result, expected)
```

```
sandbox.restore() [f]
```

SAMPLE SRC FILE sample.ts

```
const {slowFunction} = require('someModule')  
  
export async function a(x) {  
  x = await slowFunction()  
}
```

6. Stub private functions (Snippet: stub-private)

[A] Include rewire and require the module

```
const rewire = require('rewire')
```

```
let $MODULE$ = rewire('$PATH')
```

[B] Create a stub for the function

```
const $FUNC$Fake = ()=> {  
    // replacement code here...  
}
```

[C] Typically in a beforeEach, assign the stub to the private function with `__set__`

```
const revert = $MODULE$.__set__('$FUNC$',  
$FUNC$Fake )
```

[D] Run any tests

[E] Restore the module

```
revert()
```

SAMPLE TEST FILE

```
// stub b()  
const rewire = require('rewire');  
const sample = rewire('../src/sample') [a]  
const stub = () => 5 [b]  
sample.__set__('b', stub) [c]  
  
// test a() [d]  
const {a} = sample
```

```
const result = a(1)
t.is(result, 10)
```

SAMPLE SRC FILE sample.ts

```
function b(x) {
  return x + 1
}
```

```
export function a(x) {
  return b(x) * 2
}
```

7. Check for coverage

A. Install nyc

```
npm i -D nyc
```

B. Add `coverage` to scripts

```
"scripts": {  
  ...,  
  "coverage": "nyc npm run test"  
}
```

C. Run as needed

\$ npm run coverage

Links

[Class Repo](#)

Main Tools

- [npm](#)
- [VS Code](#)
- Follow [these steps](#) to use extension
- [ava-recipes](#)_____

Main Tools

- [NodeJs](#)
- [TypeScript](#)
- [AVA](#)

Other Packages

- [sinon](#)
- [rewire](#)

- [nyc](#)
- [inquirer](#)

Join our [Discord Server/Community](#)

Webinar [Recording](#)

