Recipes for Success with Unit Testing

Guide

	PUBLIC	**PRIVATE**
To TEST it...	* 2	(rewire) __get__ 4
To STUB it...	(sinon) sandbox.stub() 5	(rewire) __set__

Administrative	
Create a Project	1
Check Coverage	(nyc) 6

Assertions	
Correct Output works	result - is 2
Check Coverage	error -is 6

Recipes

1. Create a project

** Prerequisite: node installed*

A. Create 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 ava settings to package.json

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

C. Add tsconfig.json

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

2. Test a public function

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

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

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

B. Create a test

```
t.is(error.message, 'foo');
```

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

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

4. Test a private function

A. (before) 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()
…
```

5. Stub function

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()
```

Alternatives to **callsFake:*

- ***.resolves(val)** resolves to that value*
- ***.reject(val)** rejects the input*
- ***.returns(val)** returns it*

6. 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**

(BONUS) Stub Private Functions
A. Include rewire and require the module const rewire = require**('rewire')** let app = rewire**('./app')**
B. Create a stub for the function
C. Typically in a beforeEach, assign the stub to the private function with __set__ **app.__set__('handleError', errorStub)**
D. Call and assert that: (1) the stub was called once and (2) returned what was expected **expect(sampleStub).to.have.been.calledOnce**
Optional ingredients ● **calledOnceWith(<params>)** ● **calledWithMatch(<partialResult>)**

Links

Main Tools

NodeJs	nodejs.org/en
TypeScript	typescriptlang.org
AVA	npmjs.com/package/ava

Other Packages

sinon	npmjs.com/package/sinon
rewire	npmjs.com/package/rewire
nyc	npmjs.com/package/nyc
inquirer	npmjs.com/package/inquirer

Join our Discord Server/Community https://discord.gg/KYk6jgmE