Testing Your Functions

Nathan Taylor
SOFTWARE ENGINEER

@taylonr taylonr.com

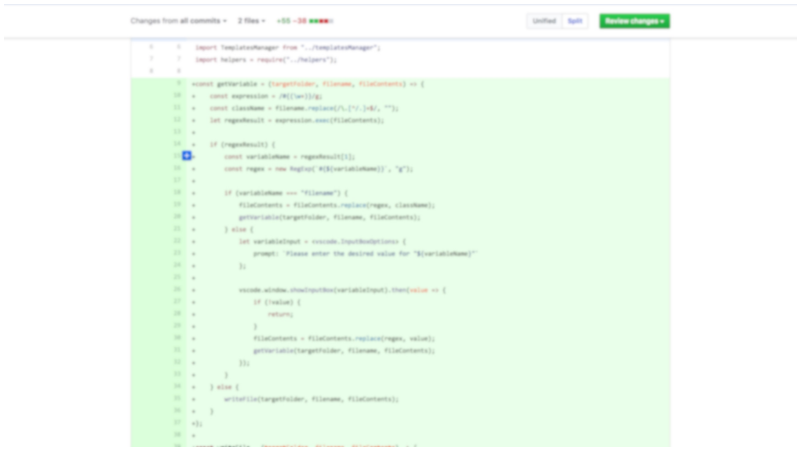

STAR
STAR

You still have an open spot?

Yep, and it’s yours if you want it.

Absolutely!

Growth and progress



```
import TemplatesManager from "../templatesManager";
import helpers = require("../helpers");

+const getVariable = (targetFolder, filename, fileContents) => {
+    const expression = /#{(\w+)}/g;
+    const className = filename.replace(/\.[^/.]+$/, "");
+    let regexResult = expression.exec(fileContents);
+
+    if (regexResult) {
+        const variableName = regexResult[1];
+        const regex = new RegExp('#{${variableName}}', 'g');
+
+        if (variableName === "filename") {
+            fileContents = fileContents.replace(regex, className);
+            getVariable(targetFolder, filename, fileContents);
+        } else {
+            let variableInput = vscode.InputBoxOptions= {
+                prompt: `Please enter the desired value for "${variableName}"`
+            };
+
+            vscode.window.showInputBox(variableInput).then(value => {
+                if (!value) {
+                    return;
+                }
+                fileContents = fileContents.replace(regex, value);
+                getVariable(targetFolder, filename, fileContents);
+            });
+        }
+    } else {
+        writeFile(targetFolder, filename, fileContents);
+    }
+};
+
```

PRs with test coverage

Common Test Problems

Testing is like a good diet

Why don’t more people write tests for their code?

Setup is time consuming

```
const addAddress(id, contact) => {
    const user = R.find(byId(id), users);
    user.location.contacts.push(contact);
}
```

```
const addAddress(id, contact) => {
    const user = R.find(byId(id), users);
    user.location.contacts.push(contact);
}
```

```
const addAddress(id, contact) => {
    const user = R.find(byId(id), users);
    user.location.contacts.push(contact);
}
```

Create widget with users

```
describe('Widget Component', () => {
    let widget;
    beforeEach(() => {
        widget = new Widget(<users>);
    });

    <other tests>

    it('should add a contact', () => {
        widget.addContact(1, <contact>);
        ...
    });
});
```

```
describe('Widget Component', () => {
    let widget;
    beforeEach(() => {
        widget = new Widget(<users>);
    });

    <other tests>

    it('should add a contact', () => {
        widget.addContact(1, <contact>);
        ...
    });
});
```

```
describe('Widget Component', () => {
    let widget;
    beforeEach(() => {
        widget = new Widget(<users>);
    });

    <other tests>

    it('should add a contact', () => {
        widget.addContact(1, <contact>);
        ...
    });
});
```

```
const addAddress(id, contact) => {
    const user = db.users.byId(id);
    const location = db.locations.byId(user.locId);
    location.contacts.push(contact);
    db.save(location);
}
```

Using a database

Setting up Mocks

```
it('should save to the DB', () => {
    sinon.stub(db, 'save');
    sinon.stub(db.users, 'byId').returns(<userObject>);

    widget.addContact(1, <contact>);
    assert db.save.called === true;
});
```

Testing with Functional Programming

Testing tutorials use math functions

```
it(‘should return 2 for 1+1’, () => {
    assert add(1,1) === 2;
});

it(‘should return 0 for -1 + 1’, () => {
    assert add(-1, 1) === 0;
});
```

```
const addContact(user, contact) => {
    user.location.contacts.push(contact);
    return user;
}
```

Pure function

```
it(‘should add the contact’, () => {
    const user = widget.addContact(<user>, <contact>);
    assert user.location.contact === contact;
});
```

Testing a pure function

```
const addContact(id, contact) => {
    const user = R.find(byId(id), users);
    user.location.contacts.push(contact);
}
const addContact(user, contact) => {
    user.location.contacts.push(contact);
    return location;
}
```

Passing in dependencies simplifies code

```
it('should add the contact', () => {
    const user = widget.addContact(<user>, <contact>);
    assert user.location.contact === contact;
});
```

Check the return value

```
if(user.password !==
    saltedPassword(password)){

    this.invalidLogins++;

}
```

```
it('Should increase the invalid
count', () => {
    user.login('nate','pwd');

    assert user.invalidLogins === 1
});
```

Summary


EASY


STAR
STAR

You've got this

Recap

Declare What You Mean

```
for(i = 0; i < users.length; i++){
}
```

```
users.map(() => {})
```

The Power of Functions

```
products.filter(isActive);
users.filter(isActive);
metadata.filter(isActive);
locations.filter(isActive);
```

◀ Reusing isActive

Pipe

```
const isEmptyString = R.pipe(
    R.defaultTo(''),
    R.trim,
    R.isEmpty
);

isEmptyString(‘abc’);
```

Side Effects May be Harmful

The goal is not to eliminate all side effects

What about performance?

```
{
    name: ‘Charles Spurgeon’,
    email:
    ‘charles@mailinator.com’,
    account: { … },
    address: { … },
    jobs: [{...}. {...}]
}
```

```
{
    name: ‘Charles Spurgeon’,
    email:
    ‘charles@mailinator.com’,
    account: 0x12345678,
    address: 0x234567BB,
    jobs: 0x3456789A
}
```

Testing Functional Programs

React

Redux

Ramda

Thank You