



# I have 100% Coverage

How could I have a regression?

# \$ whoami

Formerly DecSecOps @ FOCUS

- 10 Hosts / 50 Services
  - ~ 5 transactions/s
  - 1 environment
  - < 1 deployment/week
  - 5th dev
- 100 Hosts / 400 Services
  - 100 transactions/s
  - 10 environments
  - 200 deployments/day
  - 15 devs

The cornerstone of this transformation? TESTS!

# Tests Quality

Who has:

- Tests for their code?
- High Code Coverage?
- Seen these tests fail?
- Confidence in their tests?

# Table of contents

1. Coverage
2. Mutation Testing

# Coverage

## Coverage is Great!

- Shows which lines/branches of code are covered by tests
- Which lines aren't
  - Quite Useful for Quirky Exception Testing

## Code

```
export function divideBy(a: number, b: number): number {  
  if (b === 0) {  
    throw new Error("Division by 0 is Invalid!");  
  }  
  return a / b;  
}
```

## Tests

```
import "mocha";  
import { expect } from "chai";  
import { divideBy } from "../src/divide-by.mjs";  
  
describe("Error Handling Tests", () => {  
  it("Existence Check", () => expect(divideBy).to.exist);  
  it("Doesn't Throw on Non-0 Argument", () => expect(divideBy(1,1)).to.not.throw);  
  it("Naive Notation", () => expect(() => divideBy(0,0)).to.throw);  
});
```

```
$ npx tsc && npx c8 mocha
```

Error Handling Tests

- ✓ Existence Check
- ✓ Doesn't Throw on Non-0 Argument
- ✓ Naive Notation

3 passing (3ms)

Yay, Right?

## All files divide-by.mts

66.66% Statements 4/6

50% Branches 1/2

100% Functions 1/1

66.66% Lines 4/6

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
1 1x export function divideBy(x: number): number {
2 1x   if (x === 0) {
3     throw new Error("Division by 0 is invalid");
4   }
5 1x   return 0 / x;
6 1x }
7
```



```
1  it("Naive Notation", () =>
2    expect(() => divideBy(0)).to.throw();
3  it("Correct Notation", () =>
4    expect(() => divideBy(0)).to.throw());
5  it("Explicit Notation", () =>
6    expect(() => divideBy(0)).to.throw("Division by 0 is invalid"));
7
```

## All files divide-by.mts

100% Statements 6/6

100% Branches 3/3

100% Functions 1/1

100% Lines 6/6

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
1 1x export function divideBy(x: number): number {
2 3x   if (x === 0) {
3 2x     throw new Error("Division by 0 is invalid");
4 2x   }
5 1x   return 0 / x;
6 1x }
7
```

Tests validate your code works as expected

Coverage validates you're invoking your code

We still haven't assessed how good our tests are!

# Mutation Testing

# Mutation Testing: RIP

- *Reach* the code (Coverage)
- *Infect* the code
- *Propagate* & catch the error (Tests)

# Mutation Testing: Code Requirements

- Tests Green
- Acceptable Coverage

# Mutation Testing: Mutators

Original	Mutated
$a + b$	$a - b$
$a - b$	$a + b$
$a * b$	$a / b$
$a / b$	$a * b$
$a \% b$	$a * b$

Source: Stryker Supported Mutators



**STRYKER**  
**KILL THE MUTANTS**



# Installing Stryker

```
npm i -D stryker-cli  
npx stryker init
```

# Running Stryker

```
npx stryker run
```

## Edge Cases on \* <=> / mutation

0 and 1 have special identity behaviours.

$$\begin{array}{ll} 0 \times 1 = 0 \div 1 & 1 \times 1 = 1 \div 1 \\ 0 \times 2 = 0 \div 2 & 2 \times 1 = 2 \div 1 \\ 0 \times 3 = 0 \div 3 & 3 \times 1 = 3 \div 1 \end{array}$$

**You probably don't want to test using these values**

# Final Tests

```
import "mocha";
import { expect } from "chai";
import { divideBy } from "../src/divide-by.mjs";

describe("Error Handling Tests", () => {
  it("Existence Check", () => expect(divideBy).to.exist);
  it("Explicit Notation", () =>
    expect(() => divideBy(1, 0)).to.throw("Division by 0 is Invalid!"));
  it("Integer Division", () => expect(divideBy(4,2)).equals(2));
});
```

# And Another Thing...

All tests/coverage-quirks.tes.mjs

- ✗ Error Handling Tests Existence Check (covered 0)
- ✓ Error Handling Tests Explicit Notation (killed 5)
- ✓ Error Handling Tests Result Expression (killed 1)
- ✓ Error Handling Tests Integer Division (killed 1)

# Final Tests... For Real!

```
import "mocha";
import { expect } from "chai";
import { divideBy } from "../src/divide-by.mjs";

describe("Error Handling Tests", () => {
  it("Explicit Notation", () =>
    expect(() => divideBy(1, 0)).to.throw("Division by 0 is Invalid!"));
  it("Integer Division", () => expect(divideBy(4,2)).equals(2));
});
```

# Conclusion

- Code Coverage is *great* to know what's not tested
- Mutation Testing allows us to assert the validity of our tests

All those giving Code Coverage a hard time

Should pick up Mutation Testing

- Me. 2024



# Thank You!

<https://github.com/carboneater/confoo-2024-mutation-testing>