Test Driven Development and Pair Programming: What You Need To Know

SOULCYCLE


EXIT
RIDE HARD
VE HARDER

What is Test Driven Development?

TDD is an innovative software development process where tests are written, BEFORE writing the bare minimum of code required for the test to be fulfilled. The code will then be refactored, as often as necessary, in order to pass the test.

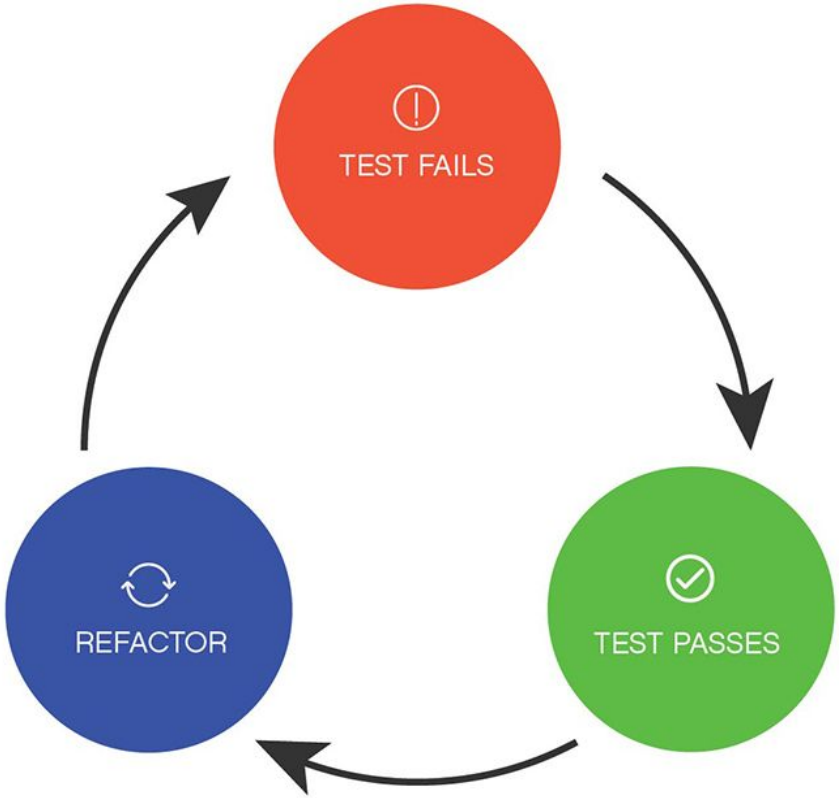
1. You are not allowed to write any production code, unless it’s to make a failing unit test pass.
2. You are not allowed to write any more of a unit test, than is sufficient to fail.
3. You are not allowed to write any more production code, than is sufficient to pass the one failing unit test.

You want me to do what?

Why use TDD?

- Increased productivity
- Considering interfaces from the beginning
- Cleaner code
- Safer refactoring
- Fewer bugs
- Increased return

TDD Cycle



Best Practices

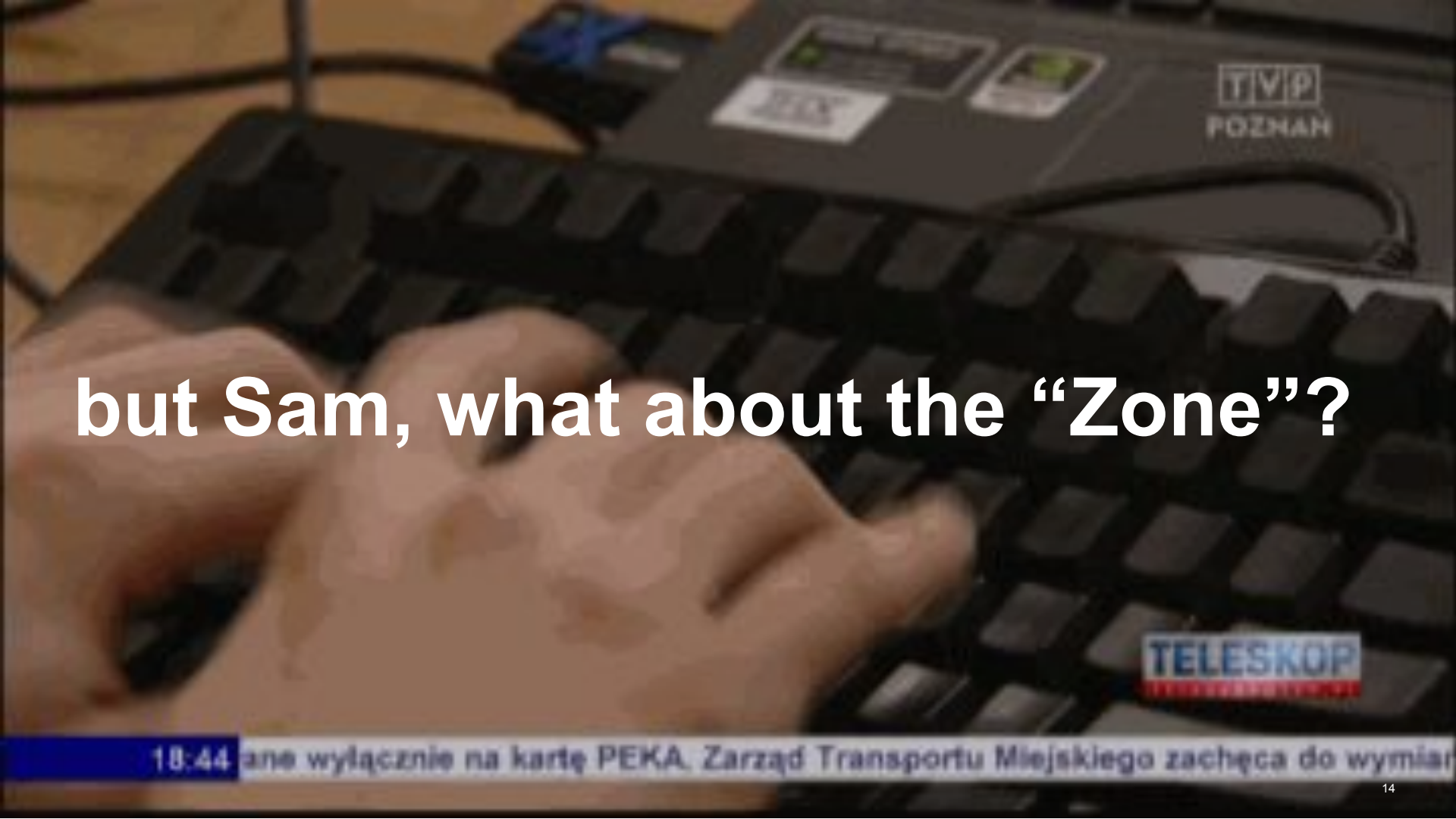
- Avoid functionality complexity
- Timeboxing
- Test repeatedly
- Great for unit testing

Before writing this off...

Let’s talk about Pair Programming

What is Pair Programming?

It is an agile software development technique in which two programmers work together at one workstation. One person drives, writes code while the other navigates by each line of code as it is typed in.



but Sam, what about the “Zone”?

Why pair program?

- Learn faster
- Transfer domain knowledge
- Highly focused
- Make friends!

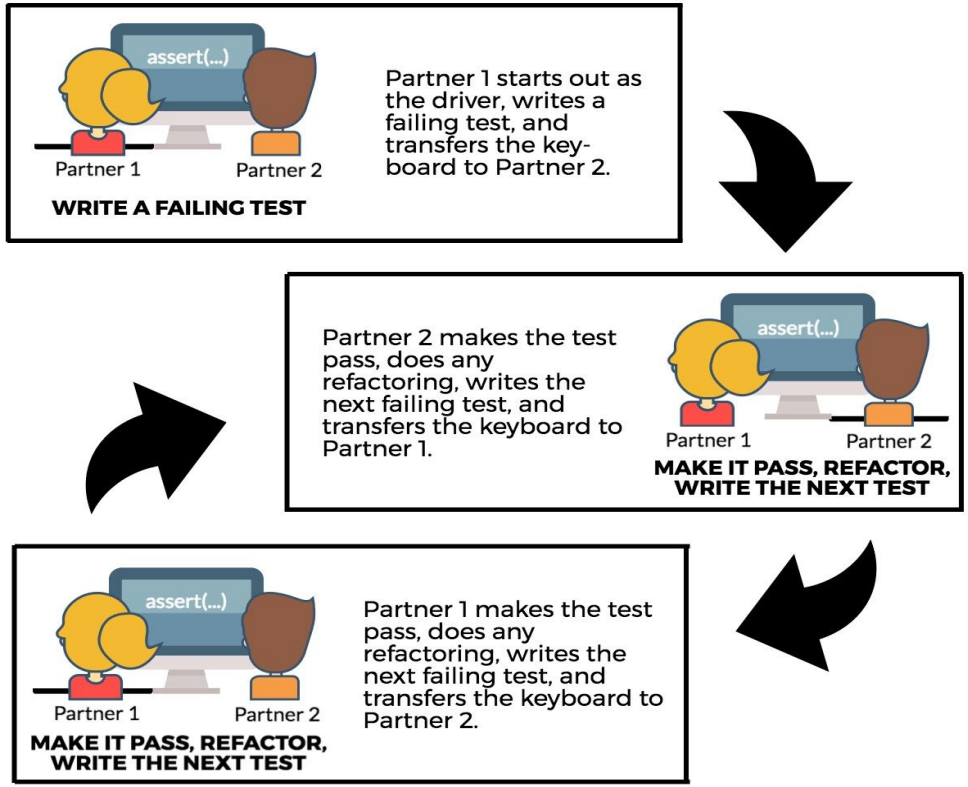
Two keyboards and two mice
Two big monitors
Two engineers
***One* computer**

How to Pair Program

- Decide who will be driver and navigator.
- Discuss the task at hand and direction before any code is written.
- Settle disagreement with honesty
- Switch roles every 30 minutes
- Celebrate accomplishments!

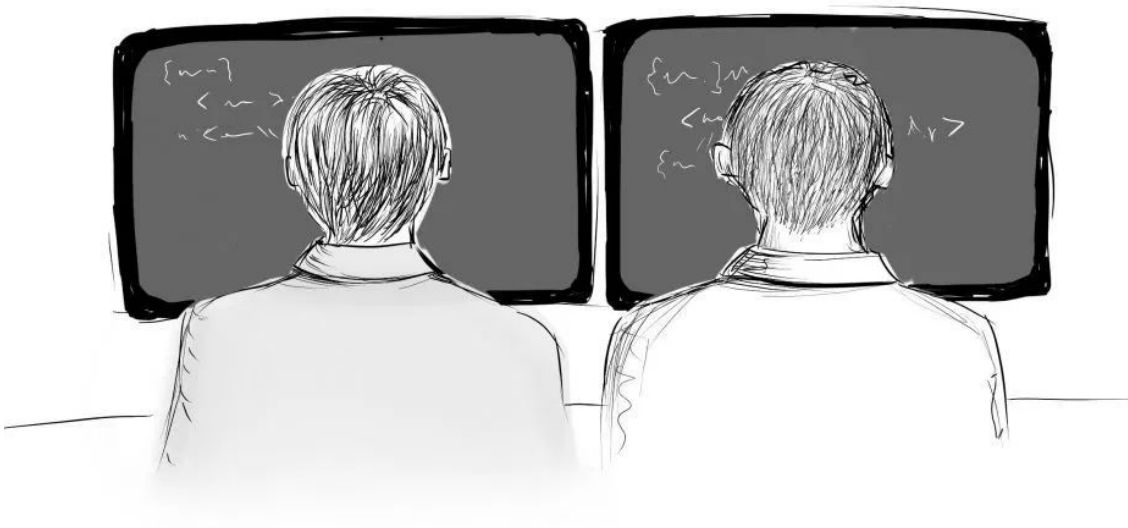
“Ping Pong” Pairing

In “ping pong” pairing, “write a failing test”, “make it pass”, “refractor” loop of Test-Driven Development is used



Best Practices

- Honesty and trust
- Clear communication and feedback in the moment
- Timebox with regular breaks in between
- Support your pair
- Pop a mint for your pairs sake



Code snippets!

```
test('should match image of pug', () => {
    expect.assertions(1);
    return functions.fetchPugImg()
    .then(data =>{
        expect(data.message).toMatchImageSnapshot();
    })
});
```

```
● should match image of pug

  expect.assertions(1)

  Expected one assertion to be called but received zero assertion calls.

    5 |
    6 | test('should match image of pug', () => {
  >  7 |     expect.assertions(1);
      |            ^
    8 |     return functions.fetchPugImg()
    9 |     .then(data =>{
   10 |         expect(data.message).toMatchImageSnapshot();

    at Object.assertions (function.test.js:7:12)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        1.677s, estimated 2s
Ran all test suites.
npm ERR! Test failed.  See above for more details.
Sams-MacBook-Pro:ttd_1 sammossallam$
```

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

const functions = {
    fetchPugImg: () =>
        axios
        .get('https://dog.ceo/api/breed/pug/images/random')
        .then(res =>
            res.data)
        .catch(err => 'error')

};

module.exports = functions;
```

```
> ttd_1@1.0.0 test /Users/sammossallam/coding/repos/code_not_on_github/ttd_1
> jest

 PASS  ./function.test.js
  ✓ should match image of pug (529ms)

 › 1 snapshot written.
Snapshot Summary
 › 1 snapshot written from 1 test suite.

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   1 written, 1 total
Time:        2.551s
Ran all test suites.
Sams-MacBook-Pro:ttd_1 sammossallam$
```

Start simple.
Print “Hello World” to the DOM.

Namaste!

@samswagbot
sam.mossallam@soul-cycle.com