# Experience test-driven development (TDD)

Nordic Testing Days
5 June 2025

Joep Schuurkes
Staff Test Engineer
Dutch Electoral Council

@joeposaurus@chaos.social
https://smallsheds.garden

# prep for "Experience TDD"

https://codeberg.org/joeposaurus/counterstring-codekata#setup-for-the-kata

https://tnyr.me/#JbRuq5JN3d

# agenda

instructions (15 mins)

the kata (50 mins)

10:35 | break (5 mins) | 10:40

reflect & share (15 mins)

how to apply (30 mins)

wrap-up (5 mins)

# instructions

# the goal

experience test-driven development

understand the basic principles of TDD

explore how these principles apply to your work

# **NOT** the goal

become an expert in TDD

be able to teach devs TDD

apply TDD in your daily work

# red - green - refactor

think of the next step

write a failing test

run the test(s) => red

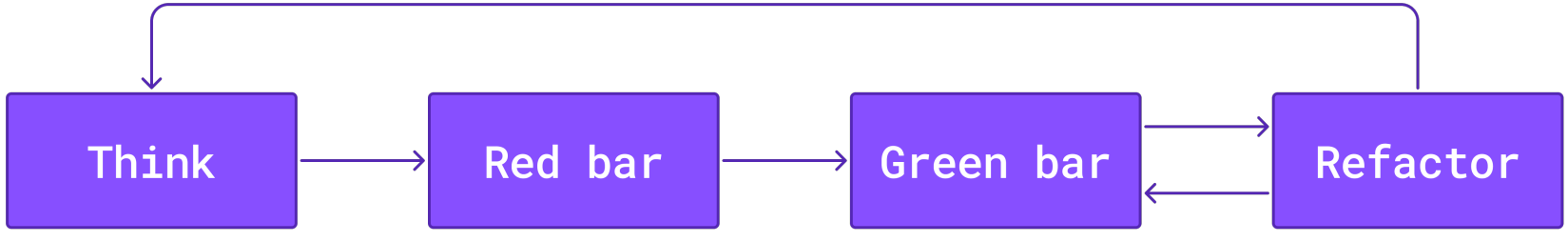
write just enough code to get the test to pass

run the test(s) => green

refactor => refactor

run the test(s)

# red - green - refactor




*The Art of Agile Development (2nd ed.) - James Shore*

# The Three Modes of TDD

1. Obvious Implementation
2. Fake It 'Til You Make It
3. **Triangulation**

# Fizz Buzz

*1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, ...*

# Triangulation - length 1

```
def test_fizzbuzz_1():
        assert fizzbuzz(1) == "1"

def fizzbuzz(length):
        return ""
```

RED - test fails

# Triangulation - length 1

```
def test_fizzbuzz_1():
        assert fizzbuzz(1) == "1"

def fizzbuzz(length):
        return "1"
```

GREEN - test passes

# Triangulation - length 1

```
def test_fizzbuzz_1():
        assert fizzbuzz(1) == "1"

def fizzbuzz(length):
        return "1"
```

Refactor? - No

# Triangulation - length 2

```
def test_fizzbuzz_1():
    assert fizzbuzz(1) == "1"


def test_fizzbuzz_2():
    assert fizzbuzz(2) == "1, 2"


def fizzbuzz(length):
    return "1"
```

RED - test fails

# Triangulation - length 2

```
def test_fizzbuzz_1():
    assert fizzbuzz(1) == "1"

def test_fizzbuzz_2():
    assert fizzbuzz(2) == "1, 2"

def fizzbuzz(length):
    if length == 1:
        return "1"
    else:
        return "1, 2"
```

GREEN - tests pass

# Triangulation - length 2

```
def test_fizzbuzz_1():
    assert fizzbuzz(1) == "1"

def test_fizzbuzz_2():
    assert fizzbuzz(2) == "1, 2"

def fizzbuzz(length):
    if length == 1:
        return "1"
    else:
        return "1, 2"
```

Refactor? - No

# Triangulation - length 3

```
# other tests ommitted from slide

def test_fizzbuzz_3():
    assert fizzbuzz(3) == "1, 2, Fizz"

def fizzbuzz(length):
    if length == 1:
        return "1"
    else:
        return "1, 2"
```

RED - test fails

# the kata

# code kata?

*an exercise in programming*
*which helps hone your skills*
*through practice and repetition*

invented by Dave Thomas

# a counterstring

*3*5*7*9*

# the setup

work in pairs or on your own

disable your coding assistant (if any)

starting point:
a function/method that returns an empty string
a (failing) test for a counterstring of length 1

reference implementation:
https://counterstring.smallsheds.garden

# the exercise

counterstring: *3*5*7*9*

think - red - green - refactor

triangulation

https://counterstring.smallsheds.garden

https://smallsheds.garden/my-talks/

hand signal 🙋

# break

# reflect & share

# share experiencies & code

share with another person/pair

10 minutes in total

# share experiencies & code

What was different?

What was the same?

# how to apply

# a question

What kinds of code do you write?

# reflect & discuss

three/six/nine/... groups

three rounds

rotate after 7 minutes

one person stays for a next round

# reflect & discuss

Where can you apply TDD in your work?

Where do you see challenges?

# wrap-up

# my two cents (1/2)

decide on the next small step
and how you can test it

take the step

test the step

tidy up (if needed)

# my two cents (2/2)

size of steps depends on length feedback loop

size of steps depends on level of confidence

amount of refactoring depends on level of experience

# thank you!

slides at smallsheds.garden/my-talks/