

ThoughtWorks®

A Gentle Introduction

TDD IN GO

*Practice, libraries and tools for
Test-Driven Development in the Go language*

Luciano Ramalho | [@standupdev](https://twitter.com/standupdev) | [@ramalhoorg](https://twitter.com/ramalhoorg)

Repo with examples and slides: <https://tgo.li/tddgo>

AGENDA

Brief introduction

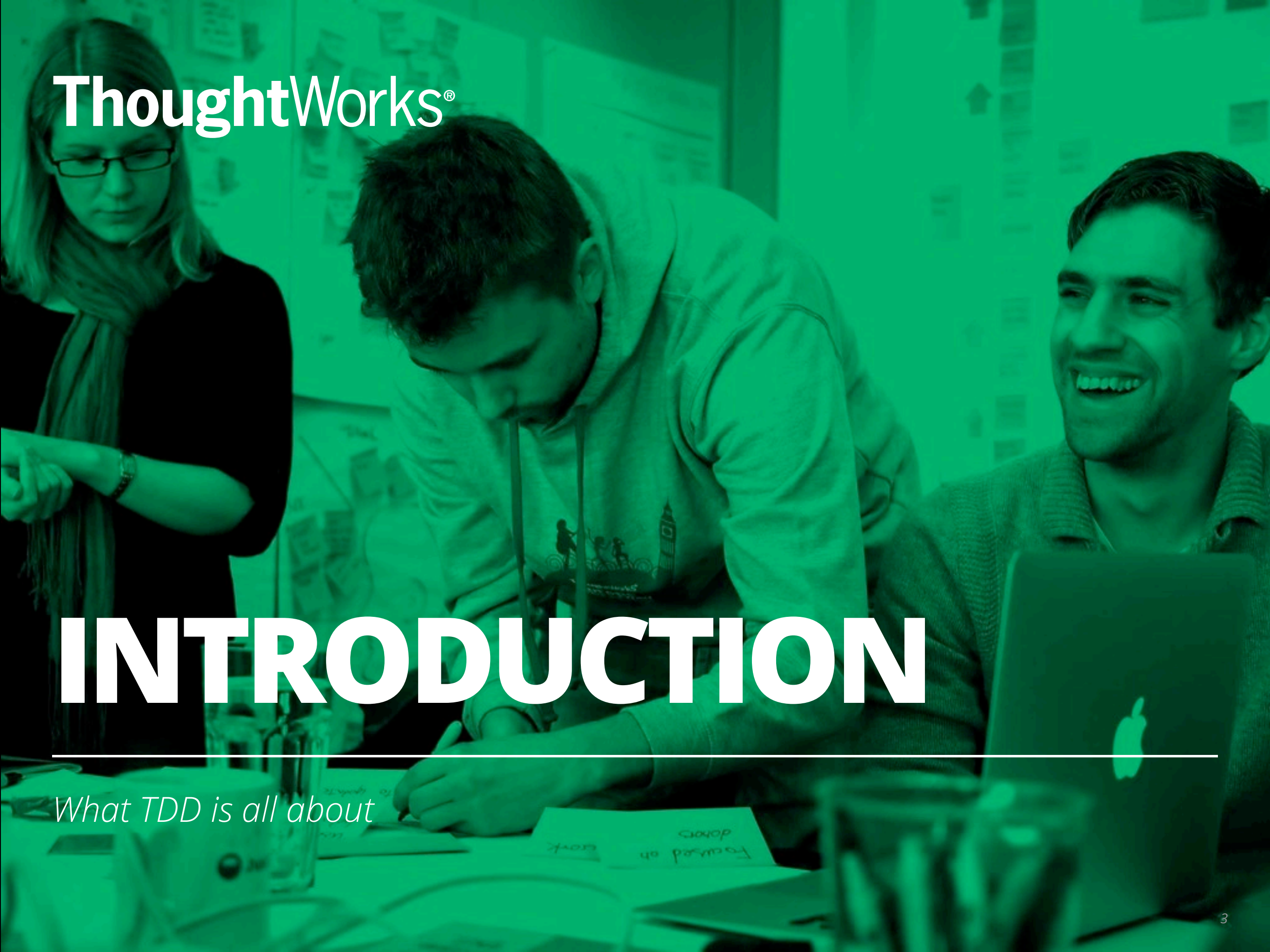
Coding Dojo (randori style): **runes** CLI app

Variations on classic TDD

Overview of tools, techniques and libraries

References for further study

Repo with examples and slides: <https://tgo.li/tddgo>

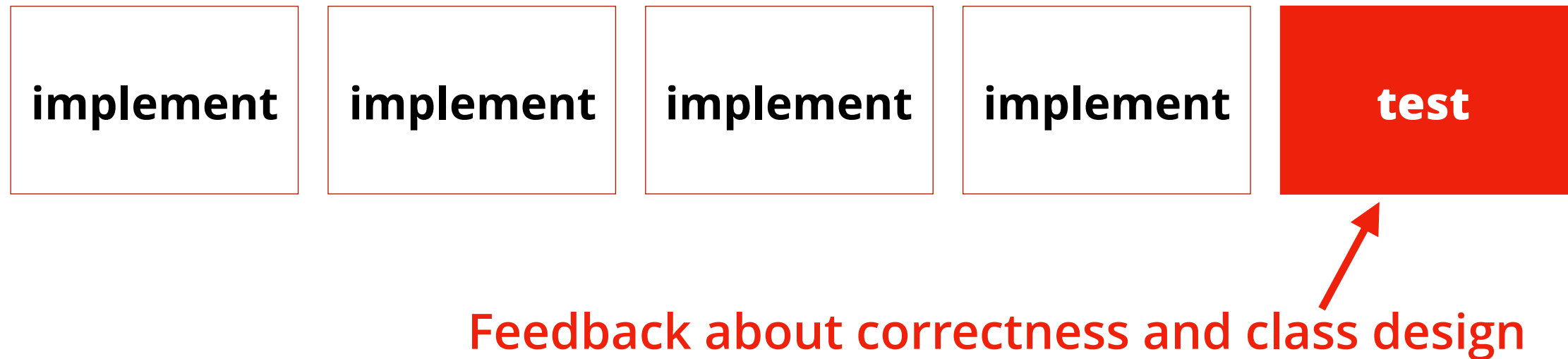


ThoughtWorks®

INTRODUCTION

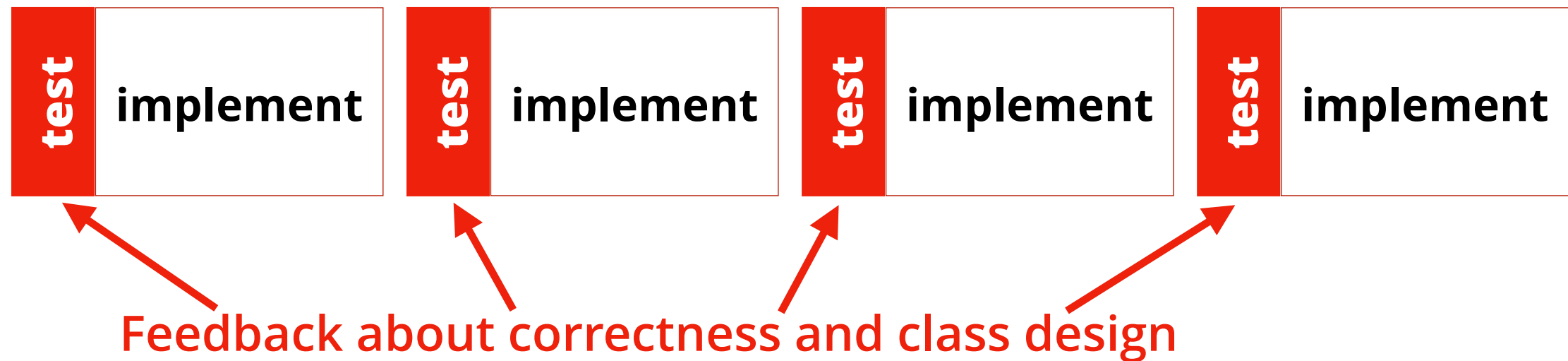
What TDD is all about

TRADITIONAL TESTING



Source: **Test-Driven Development**, by Hugo Corbucci and Mauricio Aniche

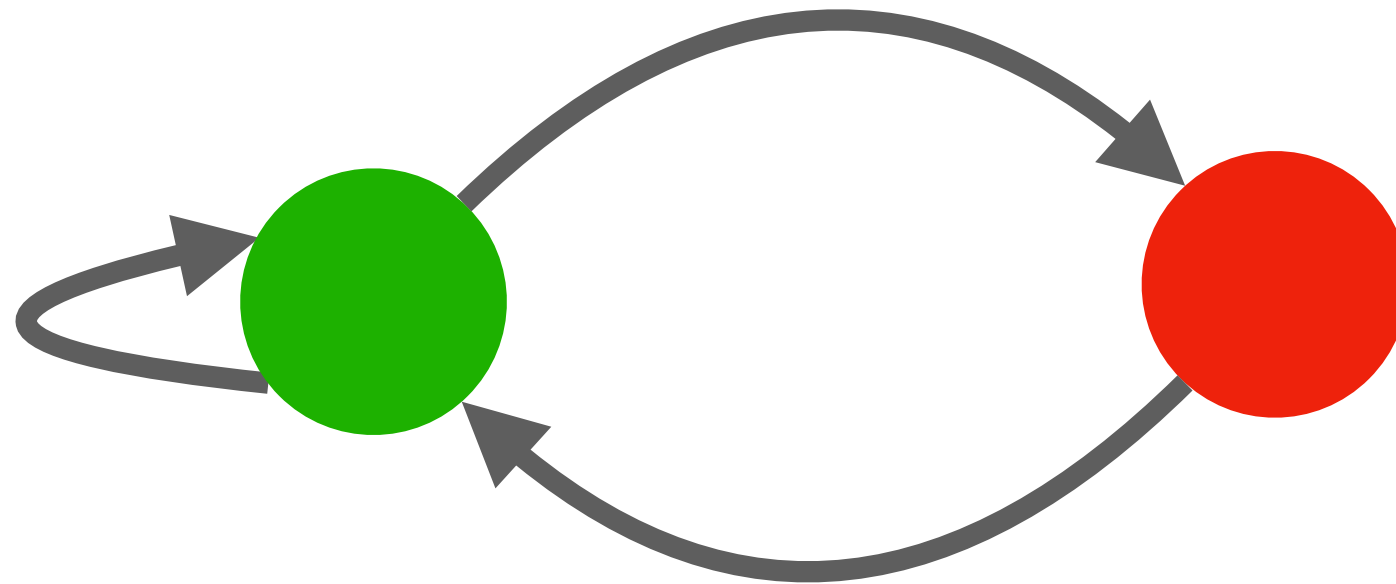
TEST-DRIVEN DEVELOPMENT



TDD CYCLE

1. Write the simplest test for next functionality

4. Refactor to
remove data and
code duplication



2. Watch it fail

3. Implement the simplest solution to satisfy the test

Source: **Test-Driven Development**, by Hugo Corbucci and Mauricio Aniche

TDD BEST PRACTICE: BABY STEPS

Work on small increments.

Time between red/green states measured in minutes, not hours.

At first: practice with the smallest increments you can think.

Like 4L on 4WD: be willing and ready to engage reduced gear when the going gets tough.

TDD BEST PRACTICE: CALL SHOT

Before running test, call out expected outcome.

When pairing, co-pilot should call outcome.

- Test will error out because there's no method named **parseLine**.
- Test will fail because function returns **1**, but the expected result is **42**.
- Test will pass.

TDD BEST PRACTICE: IMPROVE FAILING REPORTS

Figure 5.2 *Improving the diagnostics as part of the TDD cycle*



Source: **Growing Object-Oriented Software, Guided by Tests**
by Steve Freeman, Nat Pryce

A photograph of three people in a workshop or office environment, overlaid with a semi-transparent green filter. On the left, a woman with glasses and a scarf looks down. In the center, a man in a hoodie is writing on a piece of paper. On the right, a man is smiling while looking at a laptop. The background is filled with various papers and sticky notes.

ThoughtWorks®

CODING DOJO

Let's practice

CODING DOJO: RULES FOR RANDORI SESSION

Rotating pairs of pilot and co-pilot.

After 7 minutes, call volunteer for co-pilot.

When tests are green, audience can make suggestions for refactoring or next test.

When a test is red, audience should only offer suggestions when requested by pair.

OUR GOAL

\$ runes cat eyes

U+1F638 🐱 GRINNING CAT FACE WITH SMILING EYES

U+1F63B 🐱 SMILING CAT FACE WITH HEART-SHAPED EYES

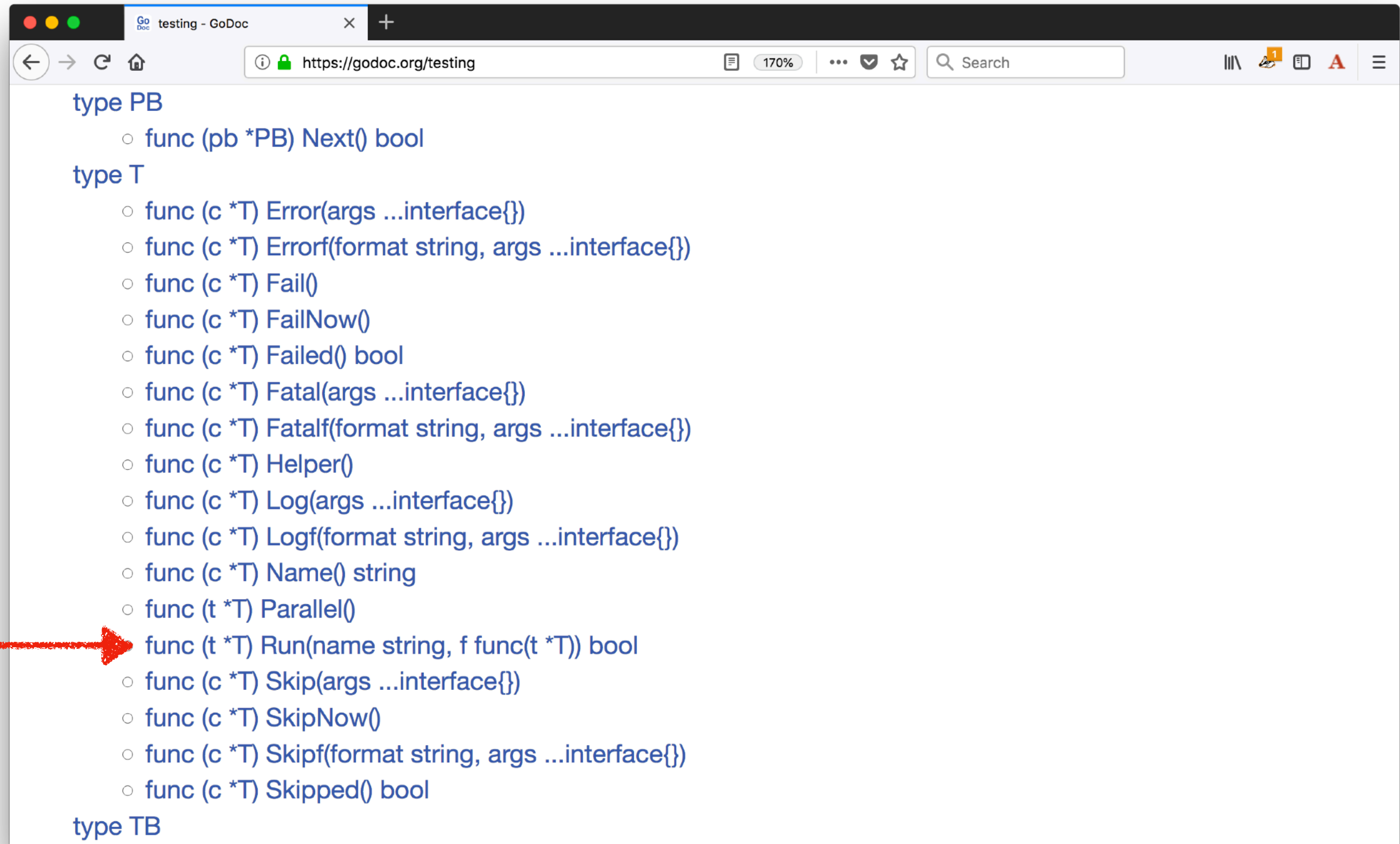
U+1F63D 🐱 KISSING CAT FACE WITH CLOSED EYES

\$ █

SIMPLE EXAMPLE-TEST

```
func Example() {  
    main()  
    // Output:  
    // Please provide one or more words to search.  
}
```

TESTING PACKAGE: T TYPE API



Go testing - GoDoc

https://godoc.org/testing

170%

Search

type PB

- func (pb *PB) Next() bool

type T

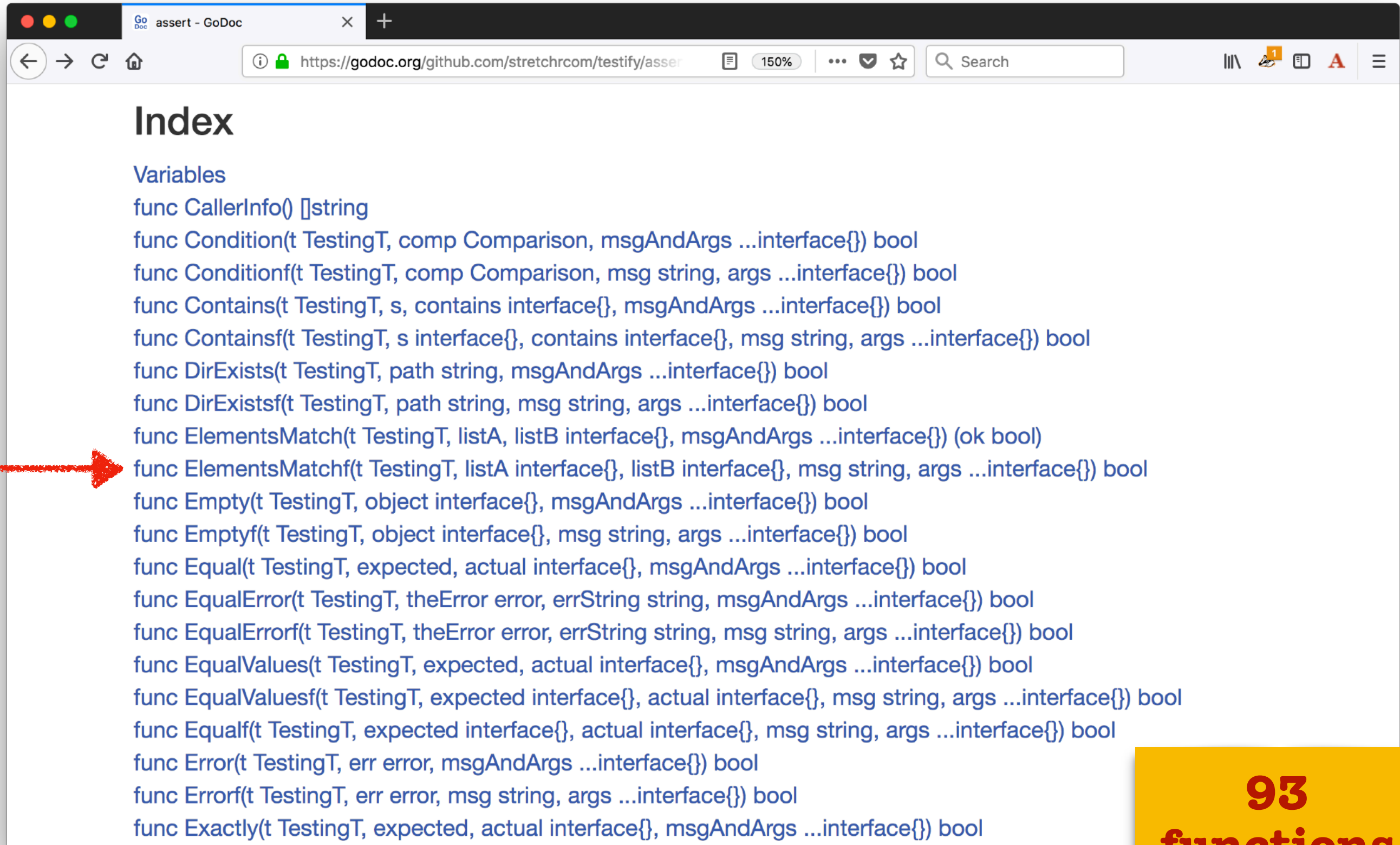
- func (c *T) Error(args ...interface{})
- func (c *T) Errorf(format string, args ...interface{})
- func (c *T) Fail()
- func (c *T) FailNow()
- func (c *T) Failed() bool
- func (c *T) Fatal(args ...interface{})
- func (c *T) Fatalf(format string, args ...interface{})
- func (c *T) Helper()
- func (c *T) Log(args ...interface{})
- func (c *T) Logf(format string, args ...interface{})
- func (c *T) Name() string
- func (t *T) Parallel()
- func (t *T) Run(name string, f func(t *T)) bool
- func (c *T) Skip(args ...interface{})
- func (c *T) SkipNow()
- func (c *T) Skipf(format string, args ...interface{})
- func (c *T) Skipped() bool

type TB

TABLE TEST WITH SUB-TESTS

```
func TestMake(t *testing.T) {
    testCases := []struct {
        elems    []string
        wantLen  int
    }{
        {[]string{}, 0},
        {[]string{"a"}, 1},
        {[]string{"a", "b"}, 2},
        {[]string{"a", "b", "a"}, 2},
    }
    for _, tc := range testCases {
        t.Run(fmt.Sprintf("%v gets %d", tc.elems, tc.wantLen), func(t *testing.T) {
            s := Make(tc.elems...)
            assert.Equal(t, tc.wantLen, s.Len())
        })
    }
}
```

PACKAGE TESTIFY: ASSERT SUB-PACKAGE API



The screenshot shows a web browser window displaying the GoDoc page for the `testify/assert` package. The browser's address bar shows the URL `https://godoc.org/github.com/stretchr/testify/assert`. The page title is "assert - GoDoc". The main content area is titled "Index" and lists various functions under the "Variables" section. A red arrow points to the `ElementsMatchf` function.

Index

Variables

- `func CallerInfo() []string`
- `func Condition(t TestingT, comp Comparison, msgAndArgs ...interface{}) bool`
- `func Conditionf(t TestingT, comp Comparison, msg string, args ...interface{}) bool`
- `func Contains(t TestingT, s, contains interface{}, msgAndArgs ...interface{}) bool`
- `func Containsf(t TestingT, s interface{}, contains interface{}, msg string, args ...interface{}) bool`
- `func DirExists(t TestingT, path string, msgAndArgs ...interface{}) bool`
- `func DirExistsf(t TestingT, path string, msg string, args ...interface{}) bool`
- `func ElementsMatch(t TestingT, listA, listB interface{}, msgAndArgs ...interface{}) (ok bool)`
- `func ElementsMatchf(t TestingT, listA interface{}, listB interface{}, msg string, args ...interface{}) bool`
- `func Empty(t TestingT, object interface{}, msgAndArgs ...interface{}) bool`
- `func Emptyf(t TestingT, object interface{}, msg string, args ...interface{}) bool`
- `func Equal(t TestingT, expected, actual interface{}, msgAndArgs ...interface{}) bool`
- `func EqualError(t TestingT, theError error, errString string, msgAndArgs ...interface{}) bool`
- `func EqualErrorf(t TestingT, theError error, errString string, msg string, args ...interface{}) bool`
- `func EqualValues(t TestingT, expected, actual interface{}, msgAndArgs ...interface{}) bool`
- `func EqualValuesf(t TestingT, expected interface{}, actual interface{}, msg string, args ...interface{}) bool`
- `func Equalf(t TestingT, expected interface{}, actual interface{}, msg string, args ...interface{}) bool`
- `func Error(t TestingT, err error, msgAndArgs ...interface{}) bool`
- `func Errorf(t TestingT, err error, msg string, args ...interface{}) bool`
- `func Exactly(t TestingT, expected, actual interface{}, msgAndArgs ...interface{}) bool`

**93
functions**

EXAMPLE-TEST WITH FAKE COMMAND-LINE ARGUMENTS

```
func Example_2WordQuery() { // ①
    oldArgs := os.Args // ②
    defer func() { os.Args = oldArgs }()
    os.Args = []string{"", "cat", "smiling"}
    main() // ③
    // Output:
    // U+1F638      🐱😊      GRINNING CAT FACE WITH SMILING EYES
    // U+1F63A      🐱😄      SMILING CAT FACE WITH OPEN MOUTH
    // U+1F63B      🐱😻      SMILING CAT FACE WITH HEART-SHAPED EYES
}
```



ThoughtWorks®

TOOLS

Libraries and utilities for TDD in Go

LIBRARIES FOR TESTING

Go testing libraries with most Github stars*

5251 ★	stretchr/testify
3685 ★	smartystreets/goconvey
2166 ★	onsi/ginkgo
1452 ★	golang/mock
902 ★	DATA-DOG/go-sqlmock
884 ★	gavv/httpexpect
709 ★	onsi/gomega
575 ★	google/go-cmp
512 ★	franela/goblin
502 ★	h2non/baloo
496 ★	h2non/gock
404 ★	DATA-DOG/godog
387 ★	go-check/check

*Data collected 2018-07-05

A photograph of three people in a collaborative workspace, overlaid with a semi-transparent green filter. On the left, a woman with glasses and a scarf looks down. In the center, a man in a hoodie is writing on a piece of paper. On the right, a man is smiling while looking at a laptop. The background shows a wall with various papers and notes.

ThoughtWorks®

TECHNIQUES

Coding tips

SIMPLE EXAMPLE-TEST USING STRINGER INTERFACE

```
func ExampleMake() {  
    w := []string{"beta", "alpha", "gamma", "beta"}  
    s := Make(w...)  
    fmt.Println(s)  
    // Output: Set{alpha beta gamma}  
}
```

EXAMPLE-TEST WITH UNORDERED OUTPUT

```
func ExampleSet_Channel() {  
    set := MakeFromText("beta alpha delta gamma")  
    // iteration order over underlying map is undefined  
    for elem := range set.Channel() {  
        fmt.Println(elem)  
    }  
    // Unordered output:  
    // alpha  
    // beta  
    // delta  
    // gamma  
}
```

TEST WITH FAKE ENVIRONMENT VARIABLE

```
func restore(nameVar, value string, existed bool) {
    if existed {
        os.Setenv(nameVar, value)
    } else {
        os.Unsetenv(nameVar)
    }
}

func TestGetUCDPATH_isSet(t *testing.T) {
    pathBefore, existed := os.LookupEnv("UCD_PATH")
    defer restore("UCD_PATH", pathBefore, existed)
    ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())
    os.Setenv("UCD_PATH", ucdPath)
    got := getUCDPATH()
    if got != ucdPath {
        t.Errorf("getUCDPATH() [set]\nwant: %q; got: %q", ucdPath, got)
    }
}
```

TEST WITH HTTP SERVER DOUBLE

```
func TestFetchUCD(t *testing.T) {
    srv := httptest.NewServer(http.HandlerFunc(
        func(w http.ResponseWriter, r *http.Request) {
            w.Write([]byte(lines3Dto43))
        })
    defer srv.Close()

    ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())
    done := make(chan bool) // ❶
    go fetchUCD(srv.URL, ucdPath, done) // ❷
    _ = <-done // ❸
    ucd, err := os.Open(ucdPath)
    if os.IsNotExist(err) {
        t.Errorf("fetchUCD did not save:%v\n%v", ucdPath, err)
    }
    ucd.Close()
    os.Remove(ucdPath)
}
```


SLOW TEST THAT CAN BE SKIPPED

```
func TestOpenUCD_remote(t *testing.T) {  
    if testing.Short() { // ❶  
        t.Skip("skipped test [-test.short option]") // ❷  
    }  
    ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())  
    ucd, err := openUCD(ucdPath)  
    if err != nil {  
        t.Errorf("openUCD(%q):\n%v", ucdPath, err)  
    }  
    ucd.Close()  
    os.Remove(ucdPath)  
}
```

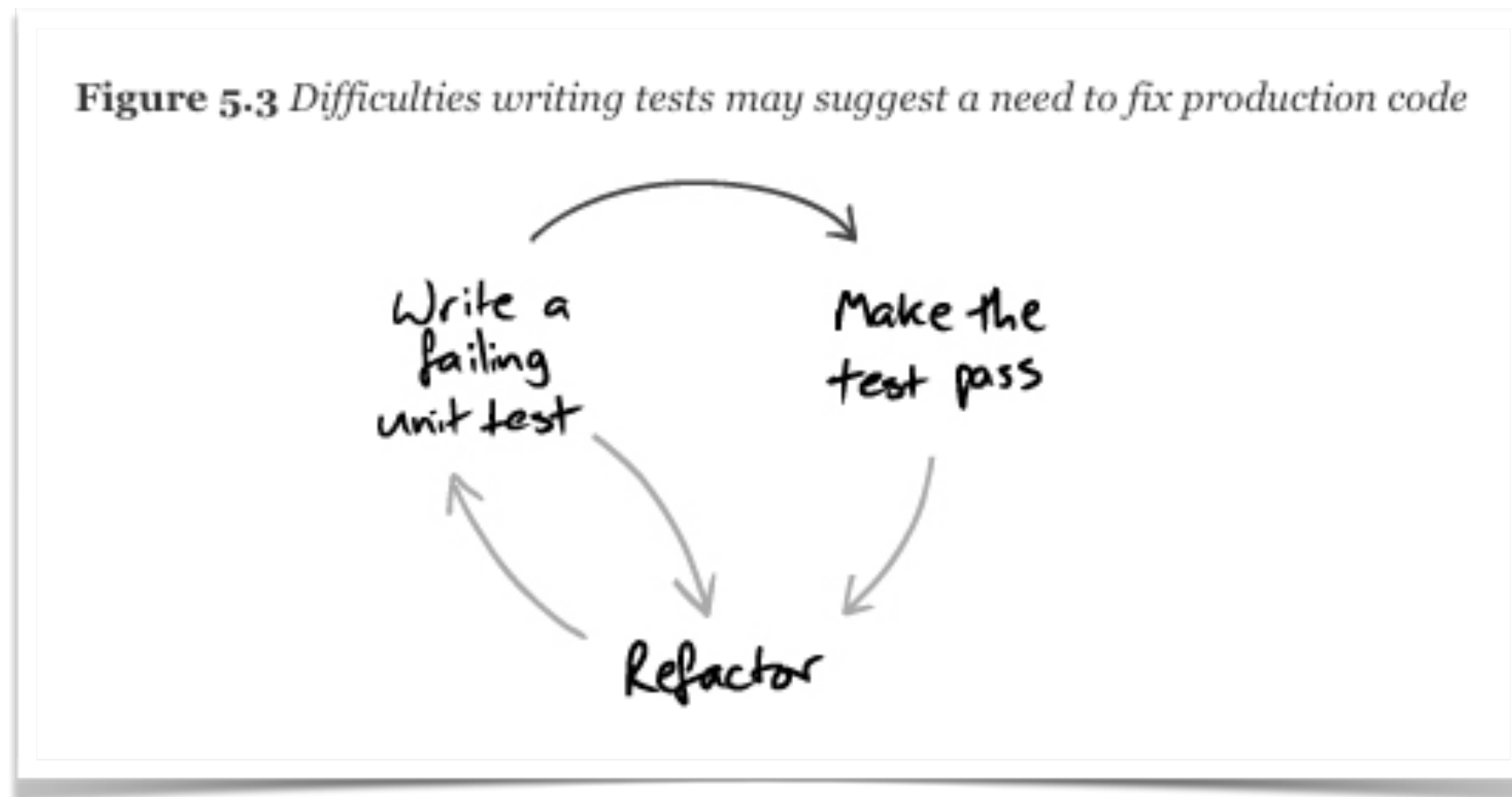
A photograph of three people in a collaborative office environment, overlaid with a semi-transparent green filter. On the left, a woman with glasses and a scarf looks down. In the center, a man in a hoodie is writing on a piece of paper. On the right, a man is smiling while looking at a laptop. The background shows a wall with various papers and notes.

ThoughtWorks®

VARIATIONS

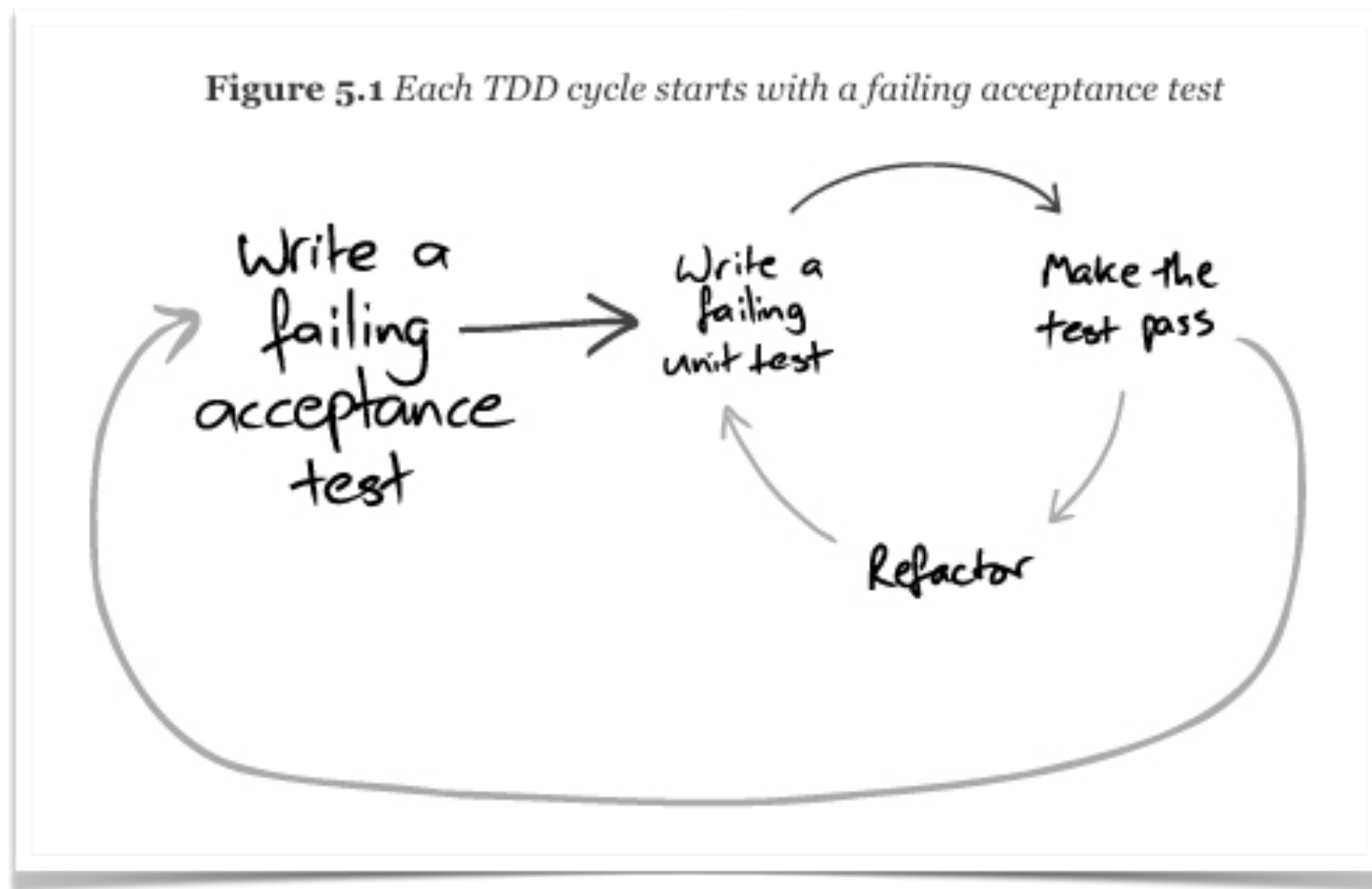
Beyond the classic TDD cycle

TDD CYCLE: REFACTOR AFTER TEST



Source: **Growing Object-Oriented Software, Guided by Tests**
by Steve Freeman, Nat Pryce

TDD CYCLES: MOCKIST STYLE



Source: **Growing Object-Oriented Software, Guided by Tests**
by Steve Freeman, Nat Pryce

TDD STYLES

Chicago style, a.k.a. “classic”

Mostly inside-out: from unit tests to acceptance tests

London style, a.k.a. “mockist”

Mostly outside-in: from acceptance tests to unit tests

ANDREW GERRAND ON FAKES

“Go eschews mocks and fakes in favour of writing code that takes broad interfaces.”

“That’s generally how we get around dependency injection frameworks and large mocking frameworks: just by writing code that uses small interfaces. Then we have small fakes like the ResponseRecorder — small fakes that allow us to inspect how they were used. There are frameworks that generate those kinds of fakes — one of them is called Go Mock [...]. They're fine, but I find that on balance the hand-written fakes tend to be easier to reason about, and clearer to see what is going on. That's my personal experience. But I am not an "enterprise" Go programmer so maybe people need that, I don't know. That's my advice.”

— Andrew Gerrand in *Testing Techniques* (I/O 2014) <https://tgo.li/2upCkek>

MARTIN FOWLER ON TDD STYLES

Mock's Aren't Stubs

https://martinfowler.com/a


Search

MARTINFOWLER.COM

Mocks Aren't Stubs

The term 'Mock Objects' has become a popular one to describe special case objects that mimic real objects for testing. Most language environments now have frameworks that make it easy to create mock objects. What's often not realized, however, is that mock objects are but one form of special case test object, one that enables a different style of testing. In this article I'll explain how mock objects work, how they encourage testing based on behavior verification, and how the community around them uses them to develop a different style of testing.

02 January 2007



Martin Fowler

Translations: [French](#) · [Italian](#) · [Spanish](#) · [Portuguese](#) · [Korean](#)
Find **similar articles** to this by looking at these tags: [popular](#) · [testing](#)

Contents

- Regular Tests
- Tests with Mock Objects
 - Using EasyMock
- The Difference Between Mocks and Stubs
- Classical and Mockist Testing
- Choosing Between the Differences
 - Driving TDD
 - Fixture Setup
 - Test Isolation
 - Coupling Tests to Implementations
 - Design Style
- So should I be a classicist or a mockist?

Source: <https://tgo.li/2IUqTXv>

A photograph of three people in a meeting, overlaid with a green tint. A woman on the left is pointing at a document. A man in the center is leaning over a table, writing on a piece of paper. A man on the right is sitting at a desk with a laptop, smiling. The background shows a wall with many sticky notes.

ThoughtWorks®

REFERENCES

Where to learn more

REFERENCES: BOOKS

Kent Beck: **Test Driven Development: By Example** <https://tgo.li/2NvBfcX>

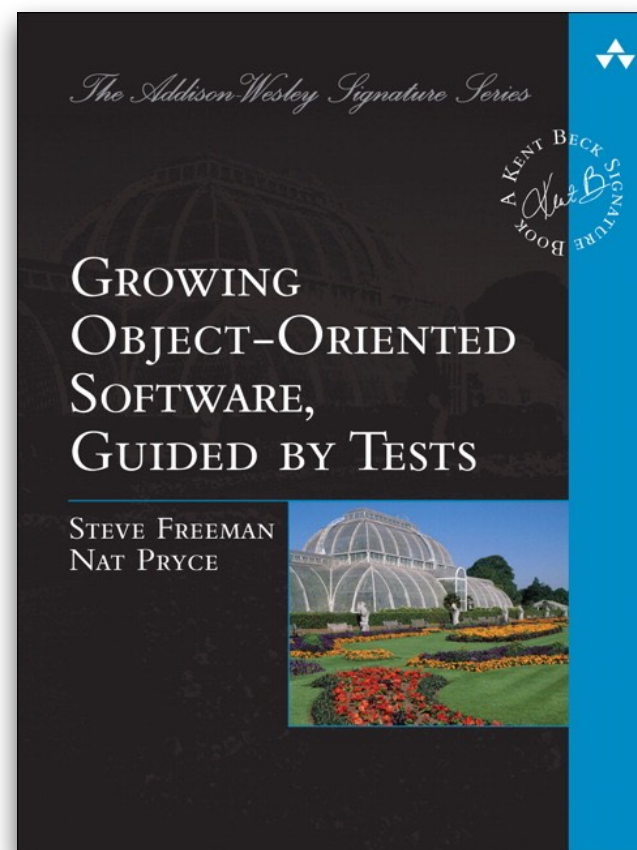
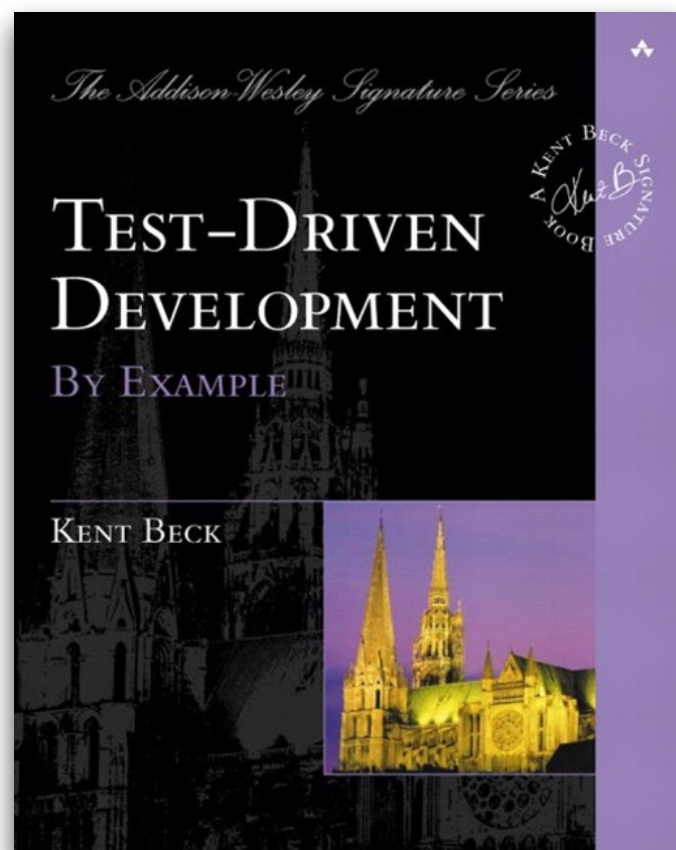
Steve Freeman, Nat Pryce:

Growing Object-Oriented Software, Guided by Tests <https://tgo.li/2tV8QoK>

Hugo Corbucci and Mauricio Aniche (book in Portuguese):

Test-Driven Development: Teste e design no mundo real com Ruby

<https://tgo.li/2zGSI4N>



REFERENCES: POSTS, VIDEOS

Andrew Gerrand [video]

Go Testing Techniques (Google I/O 2014) <https://tgo.li/2upCkek>

Francesc Campoy [video]

Unit Testing HTTP Servers (justforfunc #16) <https://tgo.li/2NSEGdZ>

Martin Angers [post]

Lesser-known Features of Go-Test <https://tgo.li/2m7ta1E>

Martin Fowler [post]

Mocks Aren't Stubs <https://tgo.li/2IUqTXv>

Martin Fowler, Kent Beck, David Heinemeier Hansson [post + videos]

Is TDD Dead? <https://tgo.li/2IWOAYn>

Michael Feathers , Steve Freeman [video]

Test Driven Development: Ten Years Later <https://tgo.li/2KD2Gnm>

THANK YOU

Let's connect!

*Luciano Ramalho
@standupdev | @ramalhoorg
luciano.ramalho@thoughtworks.com*

ThoughtWorks®