accu 2023

THE STORY OF THE CODE

DOM DAVIS

The Story Of The Code

An epic tale of... umm... stuff!

Dom Davis
@idomdavis
He/Him

Q&A

Q&A

A

Q&A

Q

Once upon a time...

Real programmers don't comment their code.

Real programmers don't comment their code.
If it was hard to write, it should be even harder to understand and modify.

Real programmers don't document.

Real programmers don't document.
Documentation is for simpletons who can't
read listing or the object code from the dump.

Real programmers don't eat quiche.

Real programmers don't eat quiche.
They like Twinkies, Coke, and
palate-scorching Szechwan food.

Real programmers don't comment their code.
If it was hard to write, it should be even harder to understand and modify.

Real programmers don't document.
Documentation is for simpletons who can't read listing or the object code from the dump.

Real programmers don't eat quiche.
They like Twinkies, Coke, and palate-scorching Szechwan food.

Real programmers don't comment their code.

Real programmers don't comment their code.

Some

~~Real~~ programmers don't comment their code.

Some

~~Real~~ programmers don't comment their code. at all

Some programmers don't comment their code some of the time.

Some

~~Real~~ programmers don't comment their code.

Some

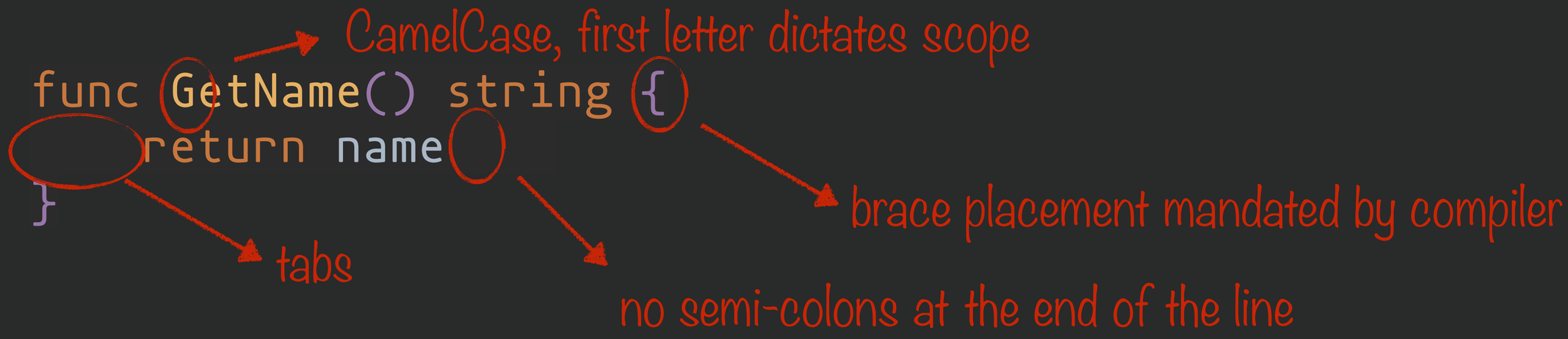
~~Real~~ programmers don't comment their code.

If it was hard to write,

Some

~~Real~~ programmers don't comment their code.
If it was hard to write, it should be even harder to understand and modify.

```
func GetName() string {
    return name
}
```

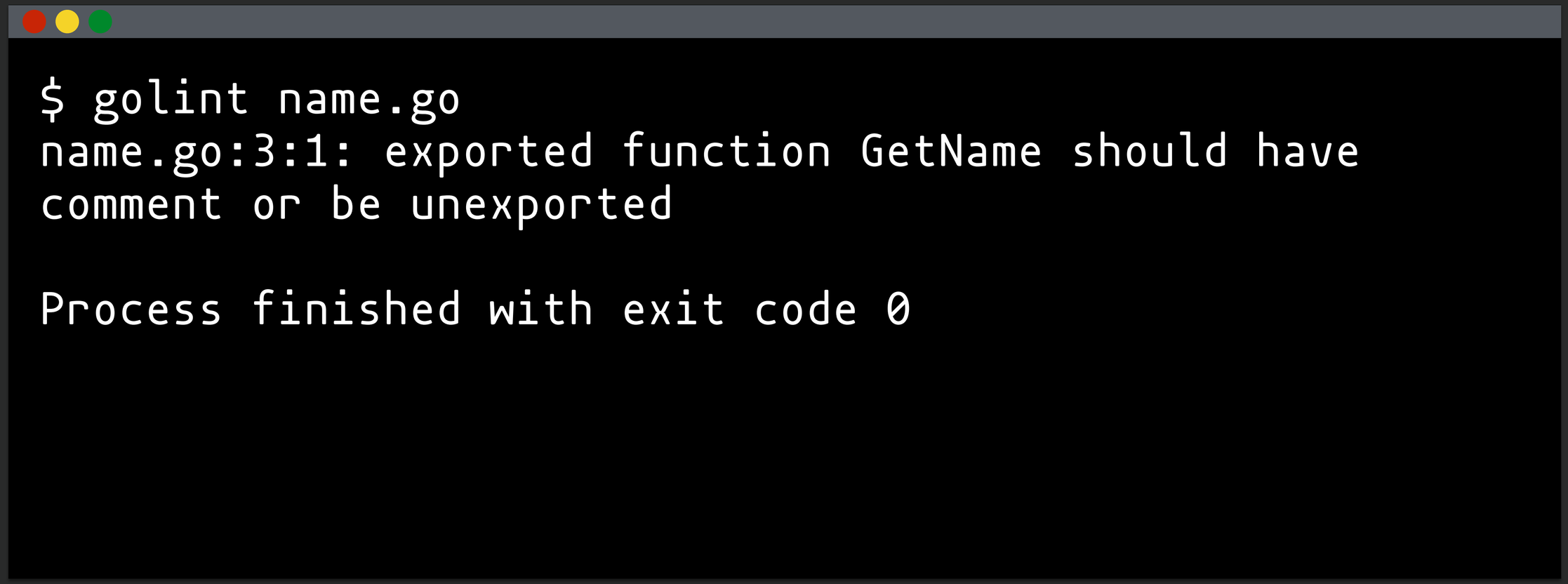


Code should be run through gofmt, or better still goimports

golint failures are errors, it either lints, or it doesnt

there is usually one idiomatic way of doing things

```go
func GetName() string {
    return name
}
```



```
// GetName comment
func GetName() string {
    return name
}
```

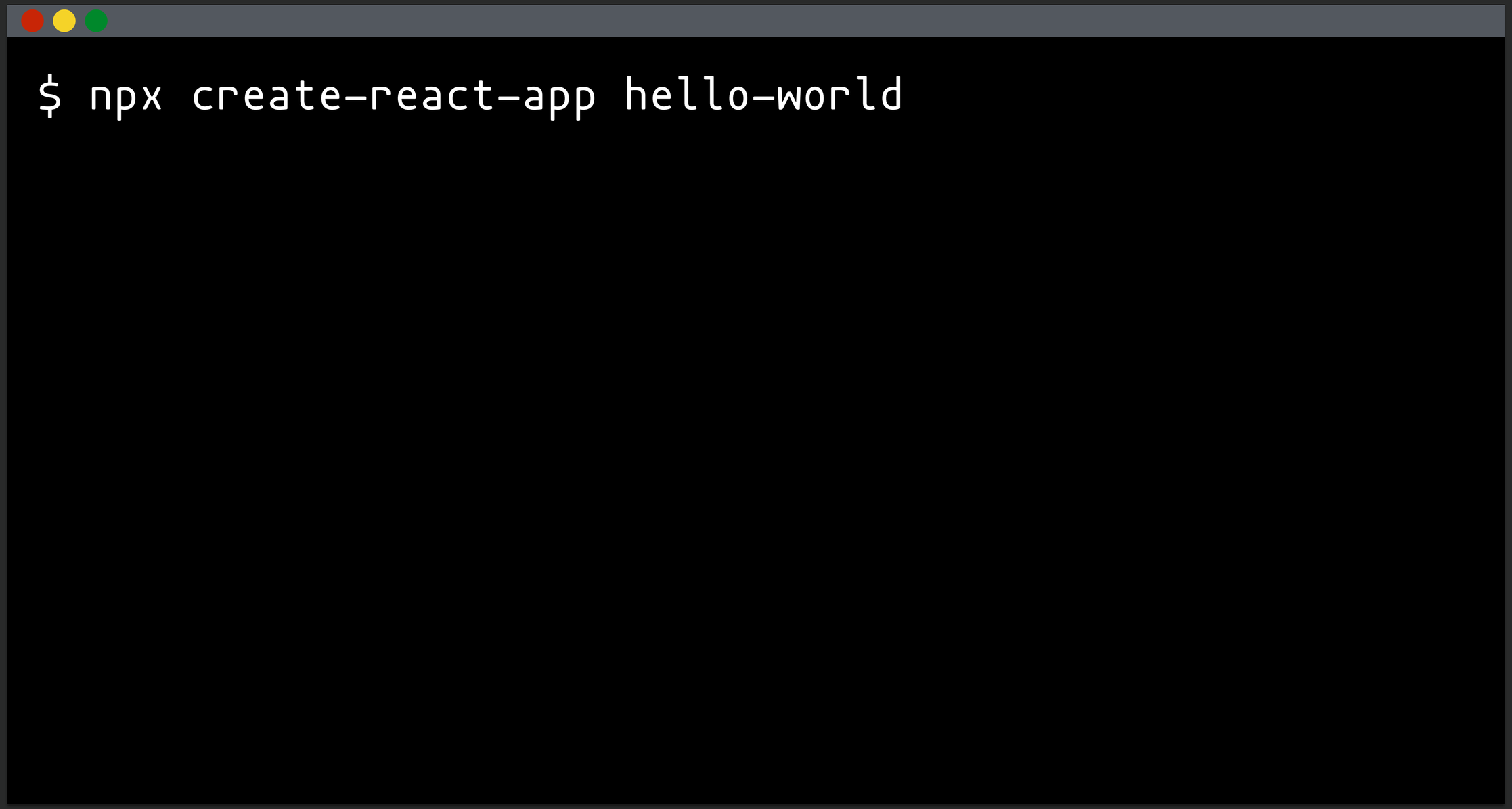
```go
// GetName returns the name.
func GetName() string {
    return name
}
```

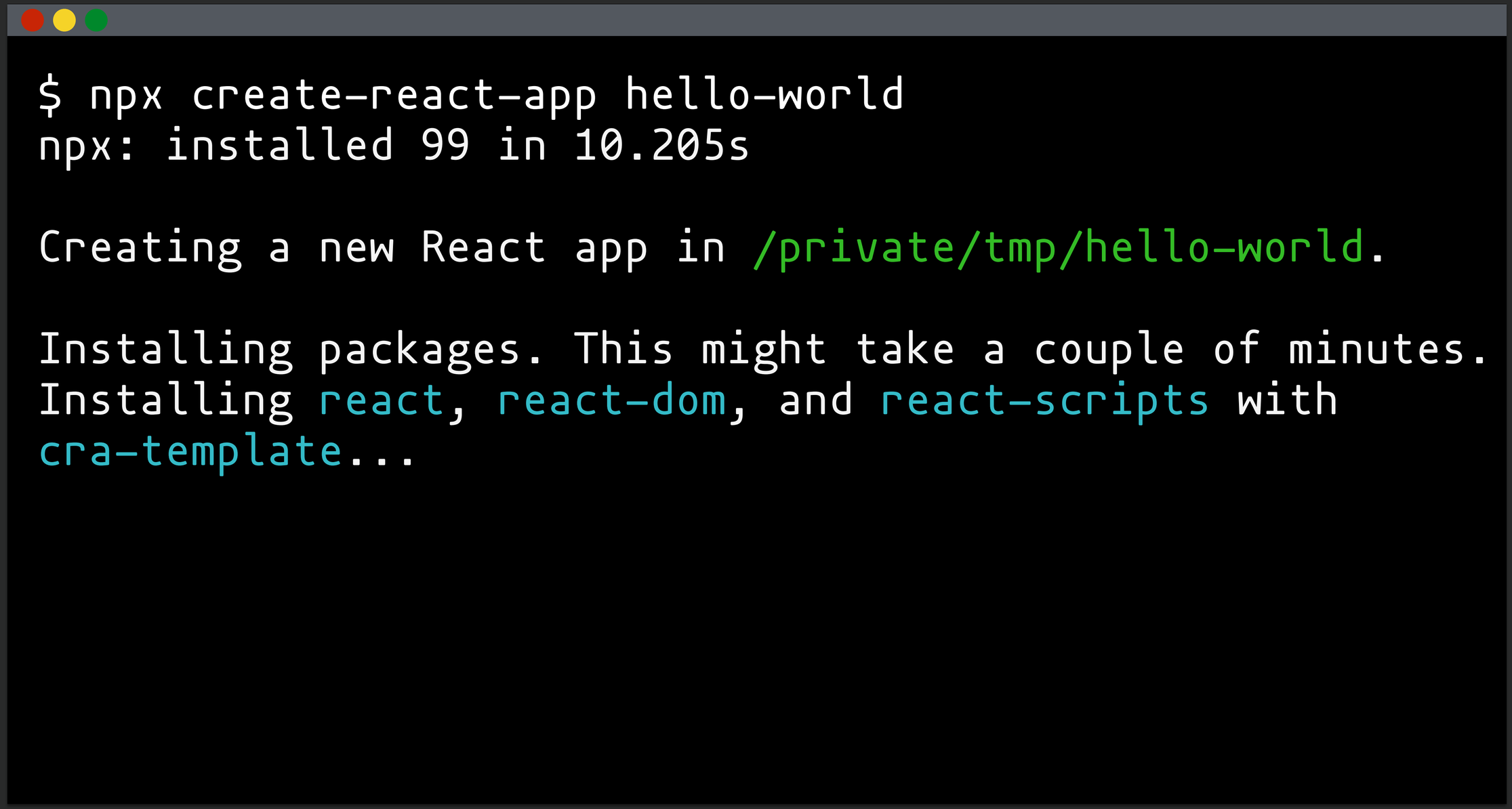
```go
// Name returns the name.
func Name() string {
    return name
}
```

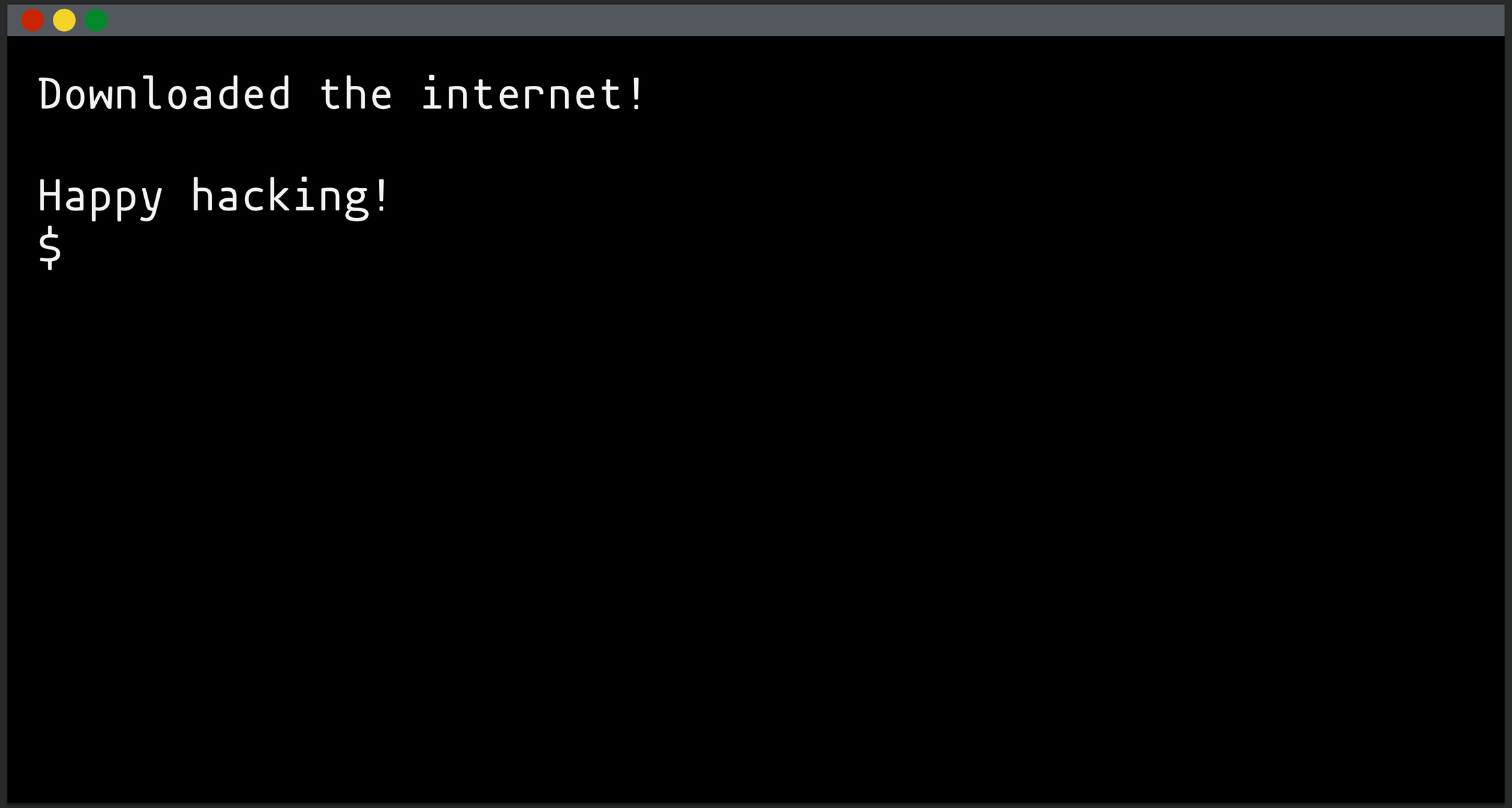
Real programmers don't document.
Documentation is for simpletons who can't
read listing or the object code from the dump.

Real programmers don't document.
Documentation is for simpletons who can't
read listing or the object code from the dump.

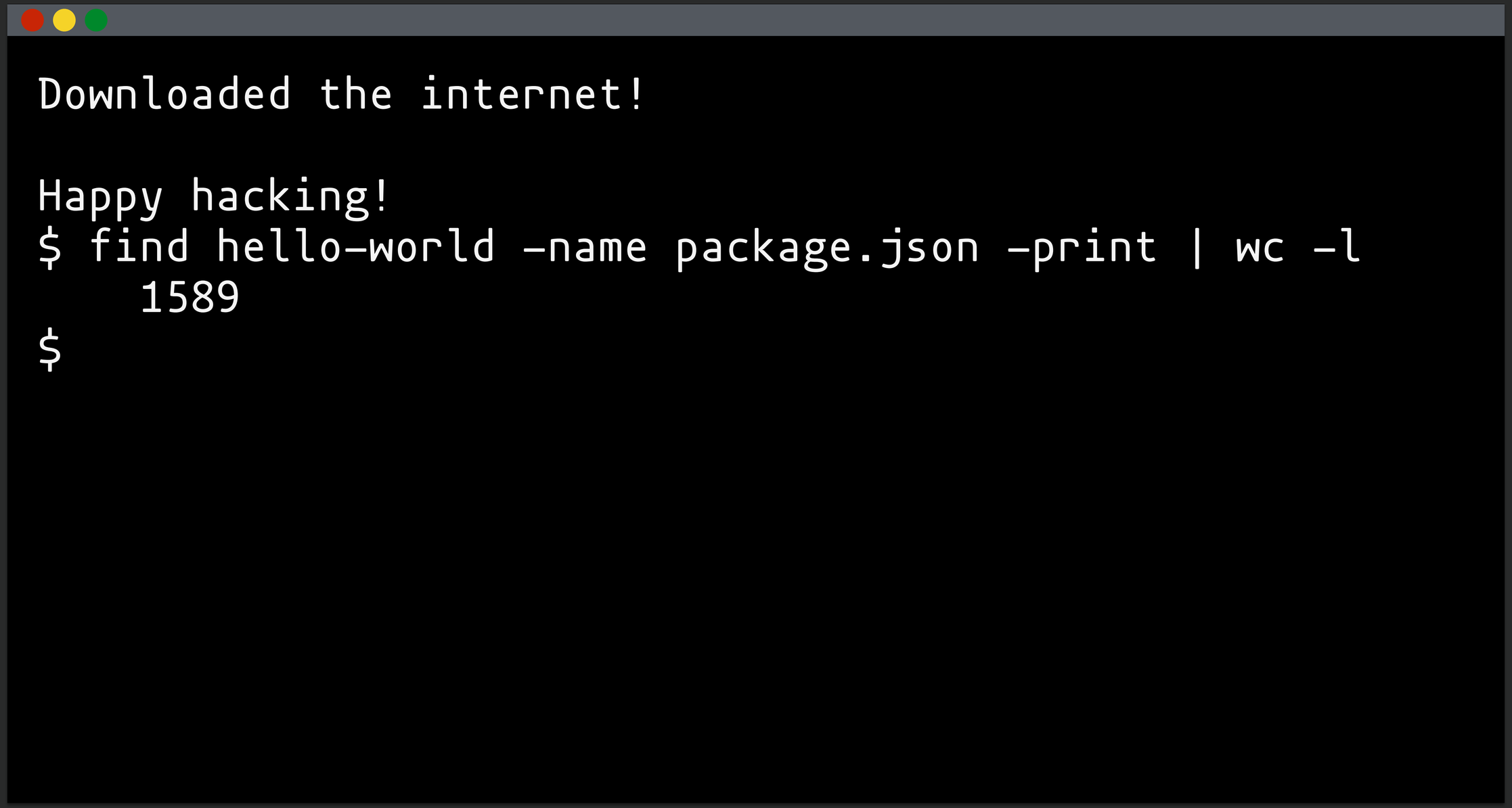
Real programmers don't document.
Documentation is for simpletons who can't
read listing or the object code from the dump.


$ npx create-react-app hello-world

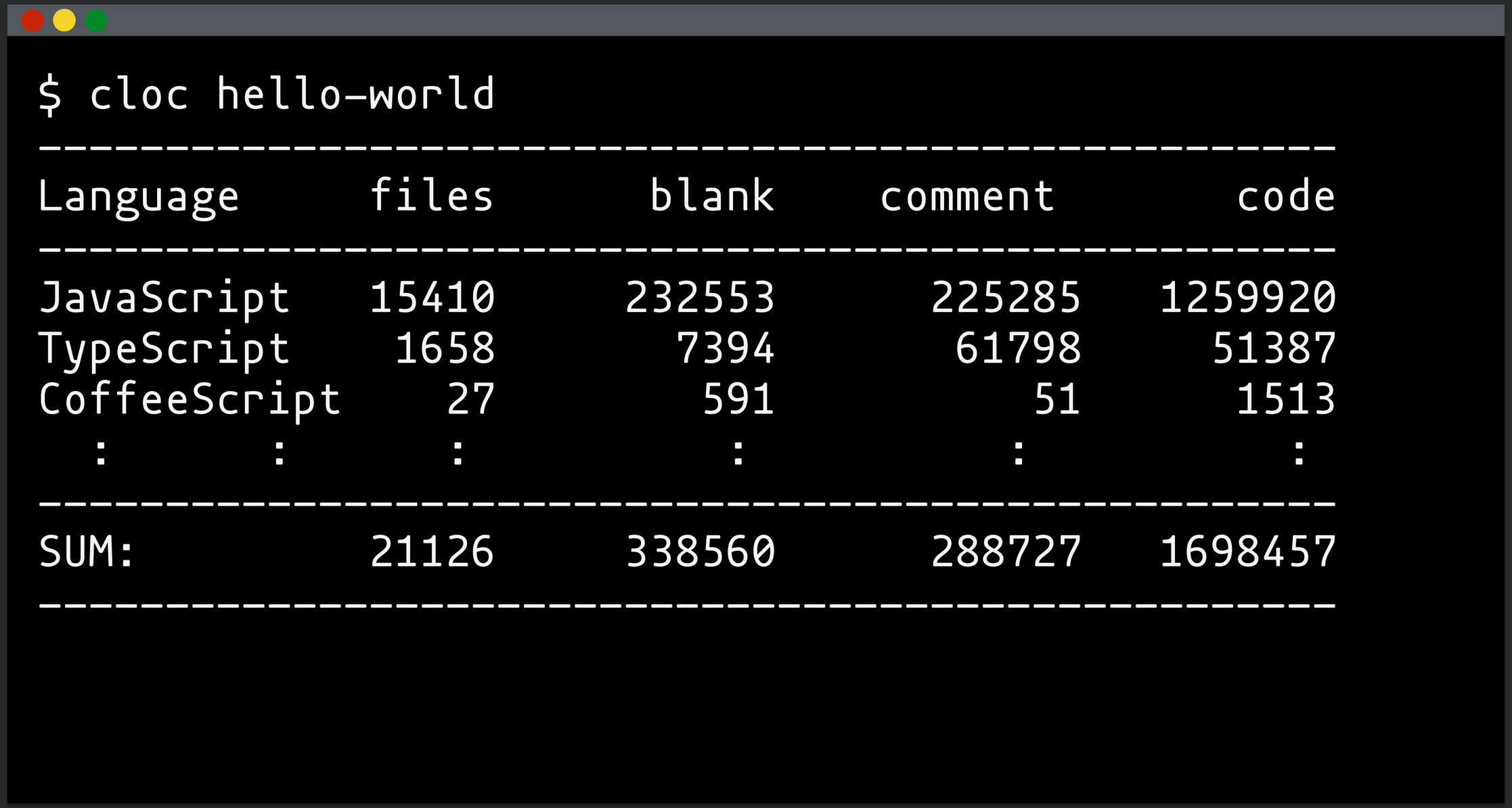

$ npx create-react-app hello-world
npx: installed 99 in 10.205s
Creating a new React app in /private/tmp/hello-world.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with
cra-template...


Downloaded the internet!
Happy hacking!
$

```
Downloaded the internet!

Happy hacking!
$ find hello-world -name package.json -print | wc -l
```


Downloaded the internet!
Happy hacking!
$ find hello-world -name package.json -print | wc -l
1589
$

```
Downloaded the internet!

Happy hacking!
$ find hello-world -name package.json -print | wc -l
    1589
$ cloc hello-world
```


$ cloc hello-world
Language files blank comment code
JavaScript 15410 232553 225285 1259920
TypeScript 1658 7394 61798 51387
CoffeeScript 27 591 51 1513
SUM: 21126 338560 288727 1698457

```
import "strings"

// Name returns the name.
func Name() string {
    return strings.ToTitle(name)

}
```

```go
import "strings"

// Name returns the name.
func Name() string {
    return strings.ToTitle(name)

}
```

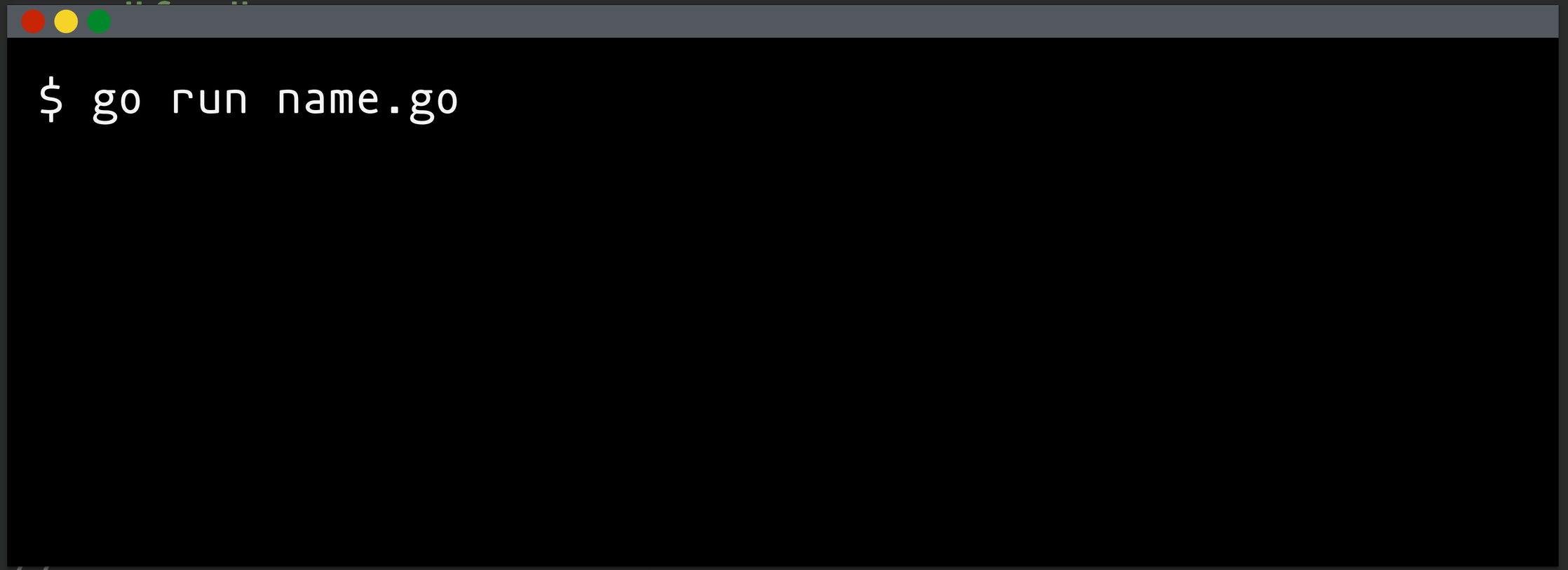
```
// ToTitle returns a copy of the string s with all Unicode
// letters mapped to their Unicode title case.
func ToTitle(s string) string {
    return Map(unicode.ToTitle, s)
}
```

```go
package main

import (
    "fmt"
    "strings"
)

const name = "Old McDonald"

func main() {
    fmt.Println(Name())
}

// Name returns the name
func Name() string {
    return strings.ToTitle(name)
}
```

```go
package main

import (
    "fmt"
    "strings"
)

const name = "Old McDonald"

func main() {
    fmt.Println(Name())
}

// Name returns the name
func Name() string {
    return strings.ToTitle(name)
}
```

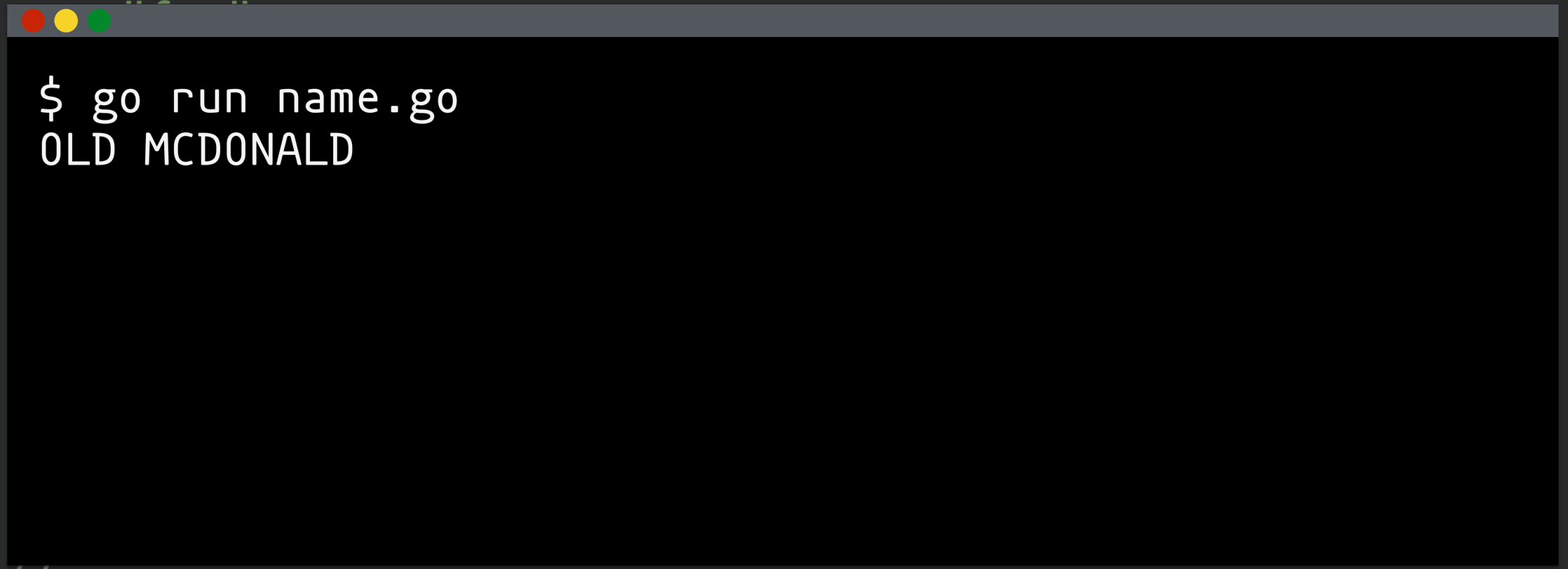
```go
package main

import (
    "fmt"
    "strings"
)

const name = "Old McDonald"

func main() {
    fmt.Println(Name())
}

// Name returns the name
func Name() string {
    return strings.ToTitle(name)
}
```

```
package main

import (
```



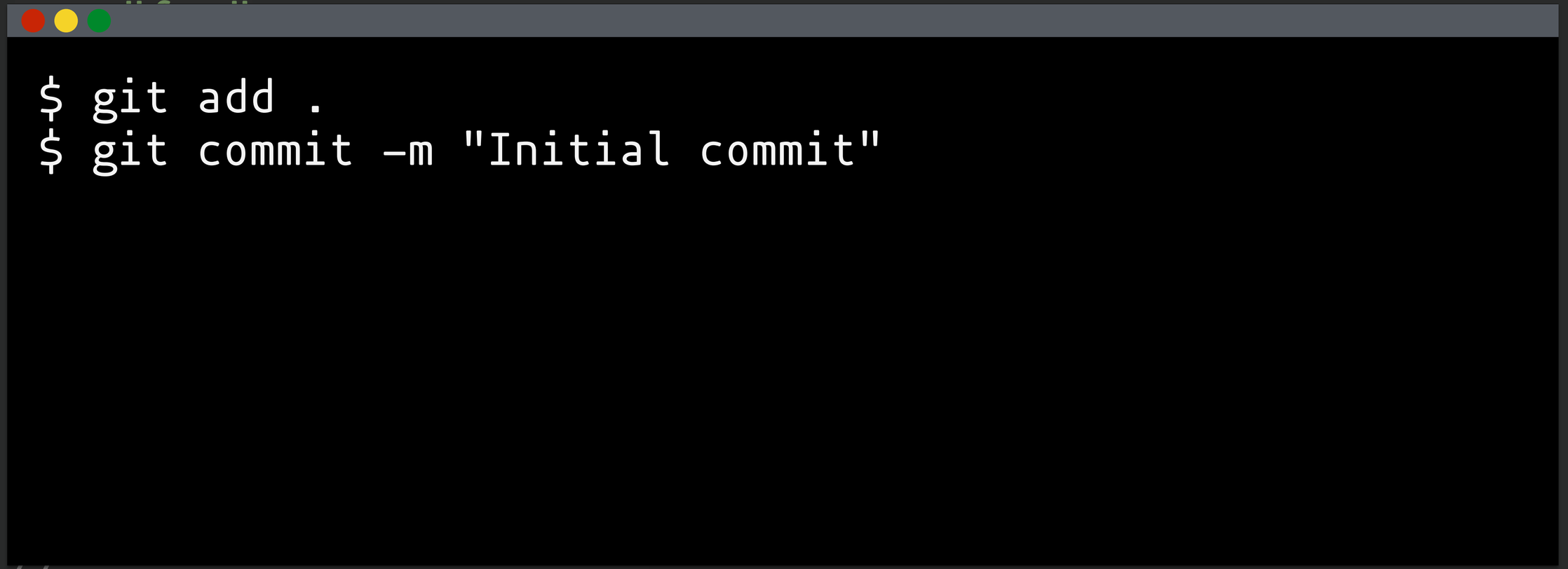

```
func Name() string {
    return strings.ToTitle(name)
}
```

```
package main

import (
```

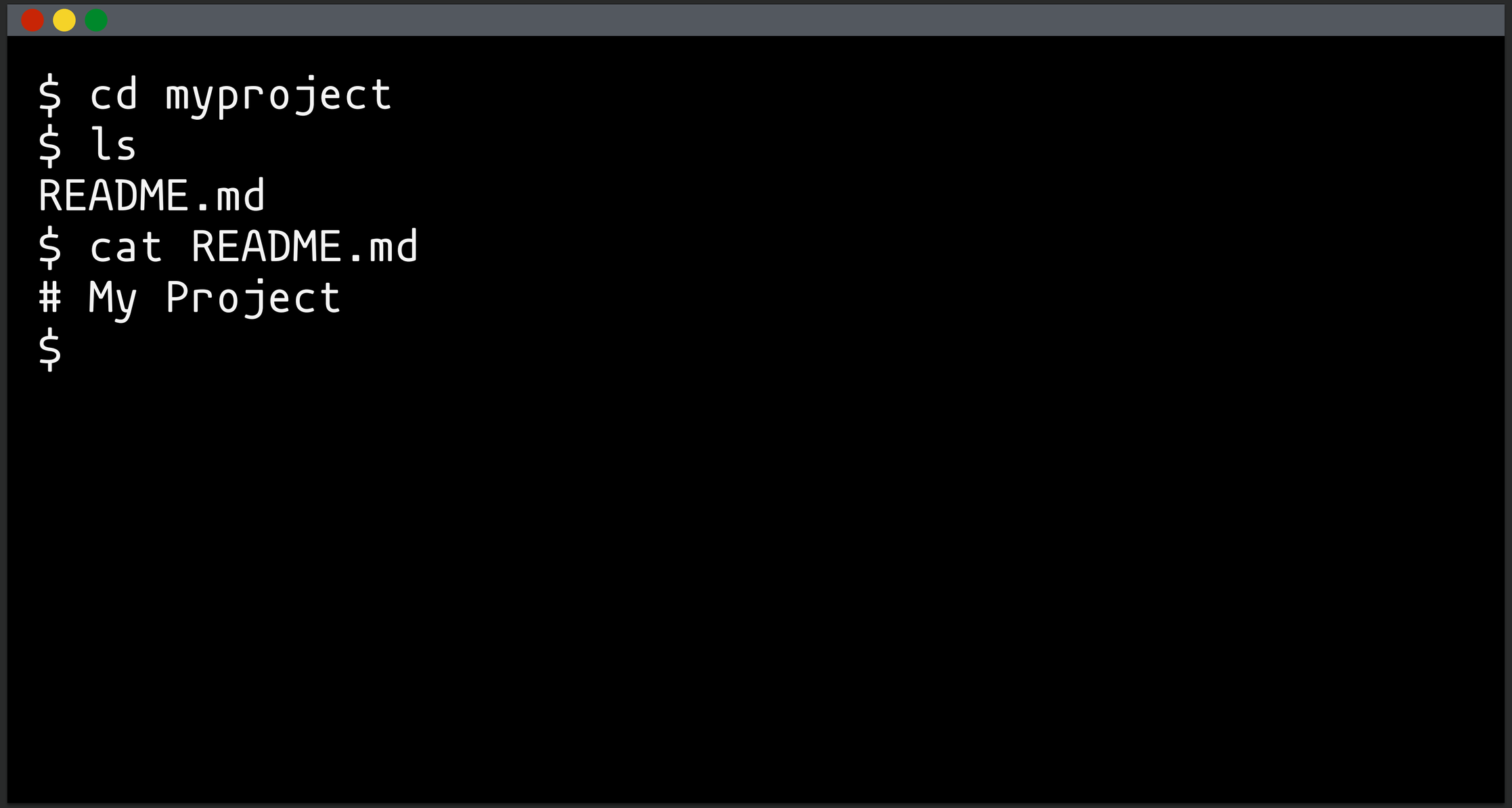



```
func Name() string {
    return strings.ToTitle(name)
}
```

```
package main

import (
```




```
func Name() string {
    return strings.ToTitle(name)
}
```


$ cd myproject
$ ls
README.md
$ cat README.md
# My Project
$

Once upon a time...

Chapter 3
Coding Begins

domdavis / gomake

Watch 1 | Star 3 | Fork 0

Code | Issues 1 | Pull requests 0 | Actions | Projects 0 | Security | Insights

Simple makefile for projects using Golang

9 commits | 1 branch | 0 packages | 1 release | 1 contributor

Branch: master | New pull request | Find file | Clone or download

domdavis Use `go list ./... | grep -v /vendor/` instead of `./...` ... Latest commit 893f051 on 5 May 2017

Makefile	Use `go list ./... | grep -v /vendor/` instead of `./...`	3 years ago
README.md	Move badges to top of README.md	3 years ago
install.sh	Fix raw path to Makefile and incorrect commands creating local version	3 years ago

gobase

Clone ...

Some base items for creating Go projects

master | Filter files

/

Name	Size	Last commit	Message
.gitignore	54 B	2019-10-18	Comment #1: Update .gitignore Don't rely on iml and DS_Store being in the global gitignore. Noticed creating ...
.golangci.yml	782 B	2019-11-04	Fix #5: Backport linter config changes
LICENSE	1.04 KB	2019-10-18	Initial commit
Makefile	412 B	2019-10-22	Fix #4: Remove setup target
README.md	884 B	2019-10-18	Fix #2: Replace dummy badge URLs with real ones.
bitbucket-pipelines.yml	96 B	2019-10-18	Initial commit
doc.go	57 B	2019-10-18	Issue #1: Add template doc.go file Easiest file to add, can be deleted if a main is going to be added instead. St...
go.mod	47 B	2019-10-18	Issue #1: Add go.mod Now we have a go file we can turn this into a proper go package and test the Makefile.
setup.sh	1.66 KB	2019-10-18	Fix #2: Replace dummy badge URLs with real ones.

README.md

Go Base

build passing | issues 0 | pull requests 0 | godoc reference | license MIT

Go Base is a base template for open source projects running from BitBucket.

ur-project

Invite Clone •••

Setup a skeleton project in Bitbucket and locally

main Files Filter files

/

Name	Size	Last commit	Message
license		2023-03-08	Issue #12: Make language agnostic
project		3 minutes ago	Issue #12: Remove deprecated golint command
.gitignore	16 B	2023-03-08	Issue #12: Make language agnostic
LICENSE	1.04 KB	2023-03-08	Issue #12: Make language agnostic
README.md	879 B	2023-03-08	Issue #12: Make language agnostic
setup.sh	1.97 KB	2023-03-08	Issue #12: Make language agnostic


xkcd

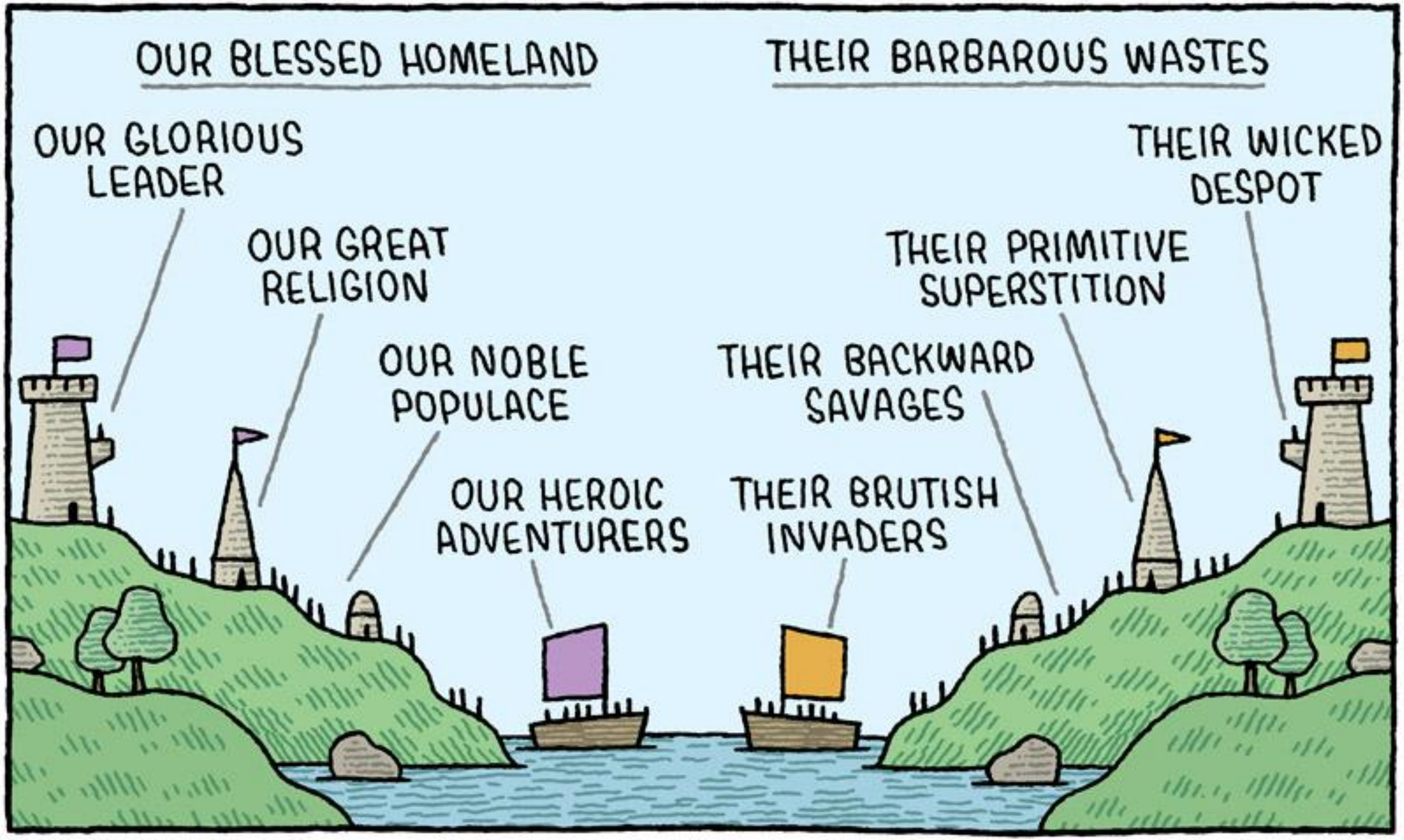

xkcd

BANG!

BANG!

Once upon a time...


Poor old Fox
Has lost his socks.


OUR BLESSED HOMELAND
THEIR BARBAROUS WASTES
OUR GLORIOUS LEADER
OUR GREAT RELIGION
OUR NOBLE POPULACE
OUR HEROIC ADVENTURERS
THEIR BRUTISH INVADERS
THEIR BACKWARD SAVAGES
THEIR PRIMITIVE SUPERSTITION
THEIR WICKED DESPOT

TOM GAULD

INITIAL COMMIT

Once upon a time,
happily ever after.

Add initial README.md

 **domdavis** committed on 31 Dec 2019

Aaand dump in a bunch of stuff in a single hit

 **domdavis** committed on 31 Dec 2019

Add initial README.md

 **domdavis** committed on 31 Dec 2019

?

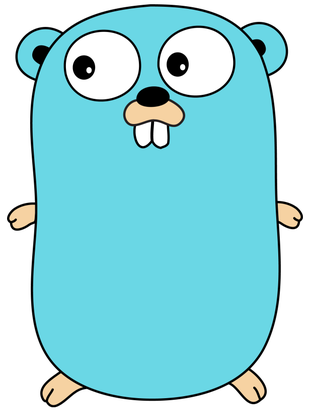
!

THE STORY OF THE CODE

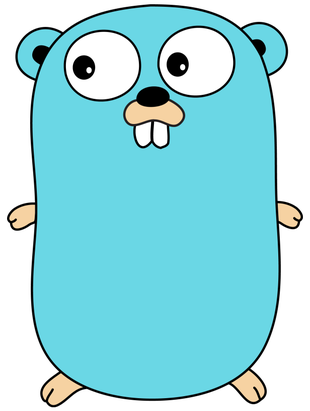
Once upon a time

Once upon a time

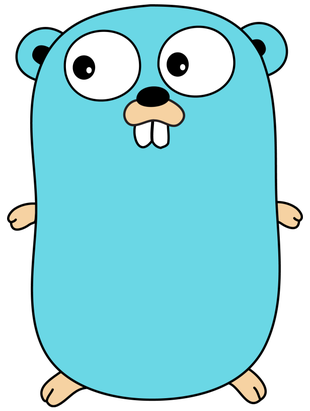
Once upon a time



Once upon a time



Once upon a time




DD8
RMA

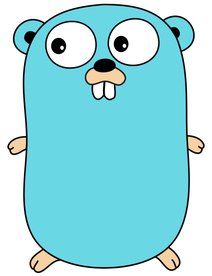
Bitbucket

Bitbucket

GitLab

Bitbucket

Bitbucket




xkcd

domdavis / gomake

Watch 1 | Star 3 | Fork 0

Code | Issues 1 | Pull requests 0 | Actions | Projects 0 | Security | Insights

Simple makefile for projects using Golang

9 commits | 1 branch | 0 packages | 1 release | 1 contributor

Branch: master | New pull request | Find file | Clone or download

domdavis Use `go list ./... | grep -v /vendor/` instead of `./...` ... Latest commit 893f051 on 5 May 2017

Makefile	Use `go list ./... | grep -v /vendor/` instead of `./...`	3 years ago
README.md	Move badges to top of README.md	3 years ago
install.sh	Fix raw path to Makefile and incorrect commands creating local version	3 years ago

gobase

Clone ...

Some base items for creating Go projects

master | Filter files

/

Name	Size	Last commit	Message
.gitignore	54 B	2019-10-18	Comment #1: Update .gitignore Don't rely on iml and DS_Store being in the global gitignore. Noticed creating ...
.golangci.yml	782 B	2019-11-04	Fix #5: Backport linter config changes
LICENSE	1.04 KB	2019-10-18	Initial commit
Makefile	412 B	2019-10-22	Fix #4: Remove setup target
README.md	884 B	2019-10-18	Fix #2: Replace dummy badge URLs with real ones.
bitbucket-pipelines.yml	96 B	2019-10-18	Initial commit
doc.go	57 B	2019-10-18	Issue #1: Add template doc.go file Easiest file to add, can be deleted if a main is going to be added instead. St...
go.mod	47 B	2019-10-18	Issue #1: Add go.mod Now we have a go file we can turn this into a proper go package and test the Makefile.
setup.sh	1.66 KB	2019-10-18	Fix #2: Replace dummy badge URLs with real ones.

README.md

Go Base

build passing | issues 0 | pull requests 0 | godoc reference | license MIT

Go Base is a base template for open source projects running from BitBucket.

Commit

Dom Davis committed **29b70c7**
2019-10-18

Initial commit

View source

master

No tags +

Pull requests

View raw commit

Stop watching

Run pipeline

Comments (0)

What would you like to say?

Files changed (7)

+4	−0	A	.gitignore
+46	−0	A	.golangci.yml
+21	−0	A	LICENSE
+46	−0	A	Makefile
+14	−0	A	README.md
+8	−0	A	bitbucket-pipelines.yml
+79	−0	A	setup.sh

https://cdna.artstation.com/p/assets/images/images/012/144/468/large/alex-kuster-tearsinrainstudy.jpg?1533248783

gobase

Clone ...

Some base items for creating Go projects

master | Filter files

/

Name	Size	Last commit	Message
.gitignore	54 B	2019-10-18	Comment #1: Update .gitignore Don't rely on iml and DS_Store being in the global gitignore. Noticed creating ...
.golangci.yml	782 B	2019-11-04	Fix #5: Backport linter config changes
LICENSE	1.04 KB	2019-10-18	Initial commit
Makefile	412 B	2019-10-22	Fix #4: Remove setup target
README.md	884 B	2019-10-18	Fix #2: Replace dummy badge URLs with real ones.
bitbucket-pipelines.yml	96 B	2019-10-18	Initial commit
doc.go	57 B	2019-10-18	Issue #1: Add template doc.go file Easiest file to add, can be deleted if a main is going to be added instead. St...
go.mod	47 B	2019-10-18	Issue #1: Add go.mod Now we have a go file we can turn this into a proper go package and test the Makefile.
setup.sh	1.66 KB	2019-10-18	Fix #2: Replace dummy badge URLs with real ones.

README.md

Go Base

build passing | issues 0 | pull requests 0 | godoc reference | license MIT

Go Base is a base template for open source projects running from BitBucket.

ur-project

Invite Clone •••

Setup a skeleton project in Bitbucket and locally

main Files Filter files

/

Name	Size	Last commit	Message
license		2023-03-08	Issue #12: Make language agnostic
project		3 minutes ago	Issue #12: Remove deprecated golint command
.gitignore	16 B	2023-03-08	Issue #12: Make language agnostic
LICENSE	1.04 KB	2023-03-08	Issue #12: Make language agnostic
README.md	879 B	2023-03-08	Issue #12: Make language agnostic
setup.sh	1.97 KB	2023-03-08	Issue #12: Make language agnostic

```
enum Bool
{
    True,
    False,
    FileNotFound
};
```

```
enum Bool
{
    True,
    False,
    FileNotFound
};
```

Who the hell wrote this⁉

```
enum Bool
{
    True,
    False,
    FileNotFound
};
```

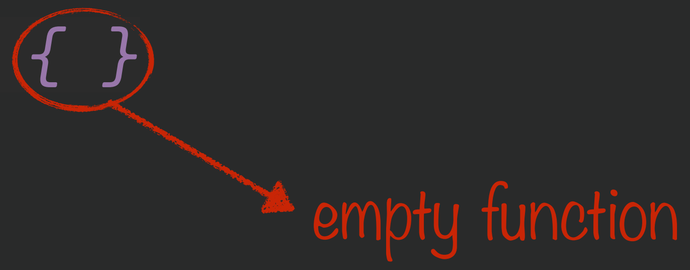
What don’t I understand about the context of this code?

```
enum Bool
{
    True,
    False,
    FileNotFound
};
```

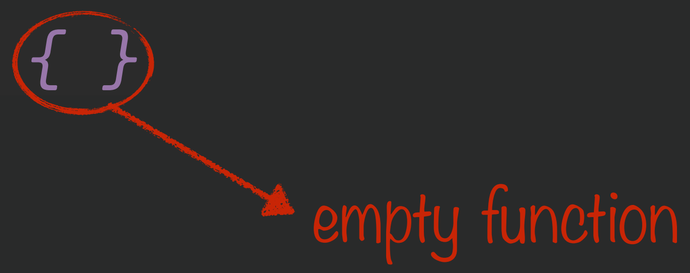
What don’t I know about the story of this code?

```
//nolint:
func Debug(session Session) { }
```

```
//nolint:
func Debug(session Session) { }
```



```
//nolint: no documentation
func Debug(session Session) { }
```



```
//nolint:
func Debug(session Session) { }
```

```
func Debug(session Session) {
    /*
     * Output the contents of session as a comment in
     * the error page we're currently building.
     */
}
```

```
//nolint:
func Debug(session Session) { }
```

```
//nolint: how can I document something I don't understand?
func Debug(session Session) { }
```

can still cause problems, but fewer

```
//nolint: how can I document something I don't understand?
func Debug(session Session) { }
```

can still cause problems, but fewer

Which idiot did this?

```
// Debug no longer does anything as it was leaking
// sensitive data.
func Debug(session Session) { }
```

```
// Debug places certain, non-sensitive data in a comment
// of the 500 error page. Items to be dumped need to be
// explicitly excluded.
//
// deprecated:use the application logger instead
func Debug(Session) {
    /*
     * Output select elements of the session as a comment
     * in the error page we're currently building.
     */
}
```

```
func Debug(session Session) {
    /*
     * Output the contents of session as a comment in
     * the error page we're currently building.
     */
}
```

Which idiot did this⁉

```
func Debug(session Session) {
    Page.Add(session.InnocuousData())
}
```

```
func Debug(session Session) {
    Page.Add(session.UsefulData())
}
```

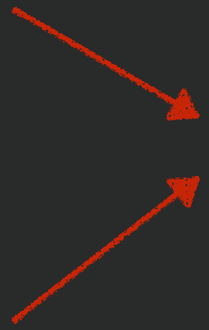
```
func Debug(session Session) {
    Page.Add(session.MOARData())
}
```

```
func Debug(session Session) {
    Page.Add(session.AllTheData())
}
```

```
func Debug(session Session) {
    Page.Add(session.AllTheData())
}
```

```
type Session struct {
    DebugInfo string
}
```

```
func Debug(session Session) {
    Page.Add(session.AllTheData())
}
```



two different areas in the code base

```
type Session struct {
    DebugInfo     string
    SensitiveInfo string
}
```

Chapter 17
Watching The World Burn

```go
// Name returns the name.
func Name() string {
    return name
}
```

```
type Person struct {
    Name string
}
```

```
// Person type.
type Person struct {

    // Name of the Person.
    Name string
}
```

```
// Person type.
type Person struct {

    // Name contains an arbitrary formatted string representing the moniker for
    // the Person.
    Name string
}
```

```
// A Person describes a character in a book.
type Person struct {

    // Name contains an arbitrary formatted string representing the moniker for
    // the Person.
    Name string
}
```

```
// A Character in a book.
type Character struct {

    // Name contains an arbitrary formatted string representing the moniker for
    // the Person.
    Name string
}
```

```
// A Character in a book.
type Character struct {

    // Name contains an arbitrary formatted string representing the moniker for
    // the Person. Name cannot be empty.
    Name string
}
```

```go
package main

import "fmt"

// A Character in a book.
type Character struct {

    // Name contains an arbitrary formatted string representing the moniker for
    // the Person. Name cannot be empty.
    Name string
}

func main() {
    c := Character{}

    fmt.Println(c.Name)
}
```

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// Name returns an arbitrary formatted string representing the moniker for
// the Person. Name will never be empty.
func (c Character) Name() string {
    if c.name == "" {
        c.name = DefaultName
    }

    return c.name
}

func main() {
    c := Character{}

    fmt.Println(c.Name())
}
```

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// Name returns an arbitrary formatted string representing the moniker for
// the Person. Name will never be empty.
func (c Character) Name() string {
    if c.name == "" {
        c.name = DefaultName
    }

    return c.name
}

func main() {
    c := Character{}

    fmt.Println(c.Name())
```

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// NewCharacter will create a new Character with the given name. If the name is
// empty then DefaultName will be used.
func NewCharacter(name string) Character {
    if name == "" {
        name = DefaultName
    }

    return Character{name: name}
}

// Name returns an arbitrary formatted string representing the moniker for
// the Person. Name will never be empty.
func (c Character) Name() string {
    return c.name
}

func main() {
    c := NewCharacter("Captain Obvious")

    fmt.Println(c.Name())
}
```

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// NewCharacter will create a new Character with the given name. If the name is
// empty then DefaultName will be used.
func NewCharacter(name string) Character {
    if name == "" {
        name = DefaultName
    }

    return Character{name: name}
}

// Name returns an arbitrary formatted string representing the moniker for
// the Person. Name will never be empty.
func (c Character) Name() string {
    return c.name
}

func main() {
    c := Character{}

    fmt.Println(c.Name())
}
```

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// NewCharacter will create a new Character with the given name. If the name is
// empty then DefaultName will be used.
func NewCharacter(name string) Character {
    if name == "" {
        name = DefaultName
    }

    return Character{name: name}
}

// Name returns an arbitrary formatted string representing the moniker for
// the Person. Name will never be empty.
func (c Character) Name() string {
    if c.name == "" {
        c.name = DefaultName
    }

    return c.name
}
```

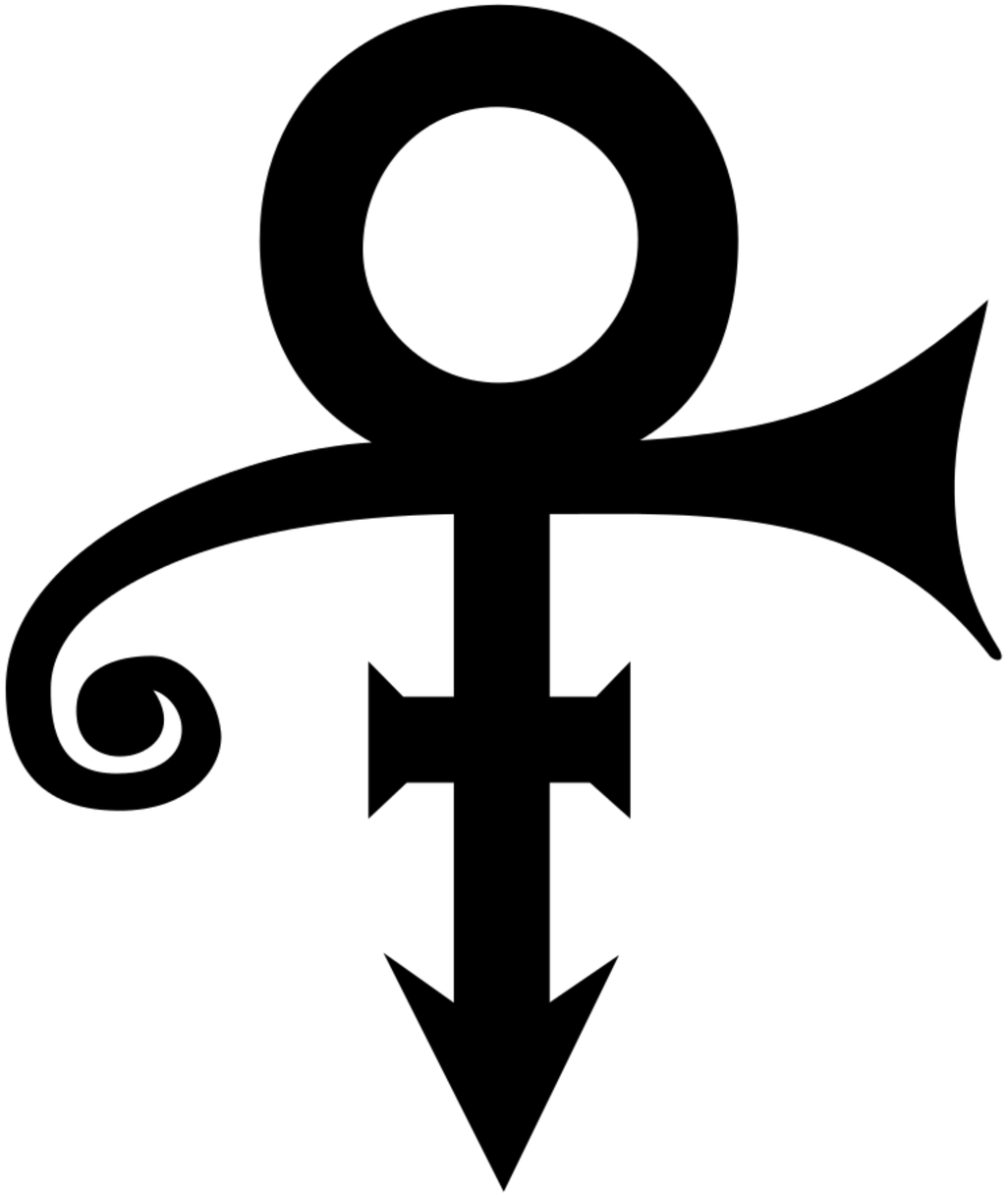
```
type Name struct {
    Order

    Title       string
    Forename    string
    MiddleNames []string
    Surname     string
    Suffixes    []string
}

type Order int

var (
    ForenameSurname = Order(1)
    SurnameForename = Order(2)
)
```

```
type Name string

var name = Name("Madonna")
```

```
type Name string

var name = Name("Jennifer Null")
```

```
type Name string

var name = Name("Q")
```

```
type Name string

var name = Name("Q 🦄")
```



MacDonald

MacDonald
Macdonald

```go
// Name returns an arbitrary formatted string containing the moniker for the
// Character. Name will always be populated, but may be the DefaultName if no
// name was provided for the Character. No assumptions can be made about the
// format regarding case, title, number of names, or order of the names. It can
// be assumed that the Name can be displayed as is.
func (c Character) Name() string {
    if c.name == "" {
        c.name = DefaultName
    }

    return c.name
}
```

```go
func TestQuery_Execute(t *testing.T) {
    t.Run("A query can execute itself", func(t *testing.T) {
        var results struct {
            N int
        }

        driver := store.Connection
        defer func() { store.Connection = driver }()

        store.Connection = mock.Driver{
            Queryer: store.BoltQueryer(mock.NewBolt(mock.NewSession(
                mock.NewResult([]string{"n"}, [][]interface{}{{1}})))),
        }

        cypher := "MATCH (n) return n"
        q := store.Query{Statement: cypher, Results: &results}
        err := q.Execute()

        switch {
        case err != nil:
            t.Errorf("unexpected error running query: %v", err)
        case results.N != 1:
            t.Errorf("unexpected value running query: %d", results.N)
        }
    })
}
```

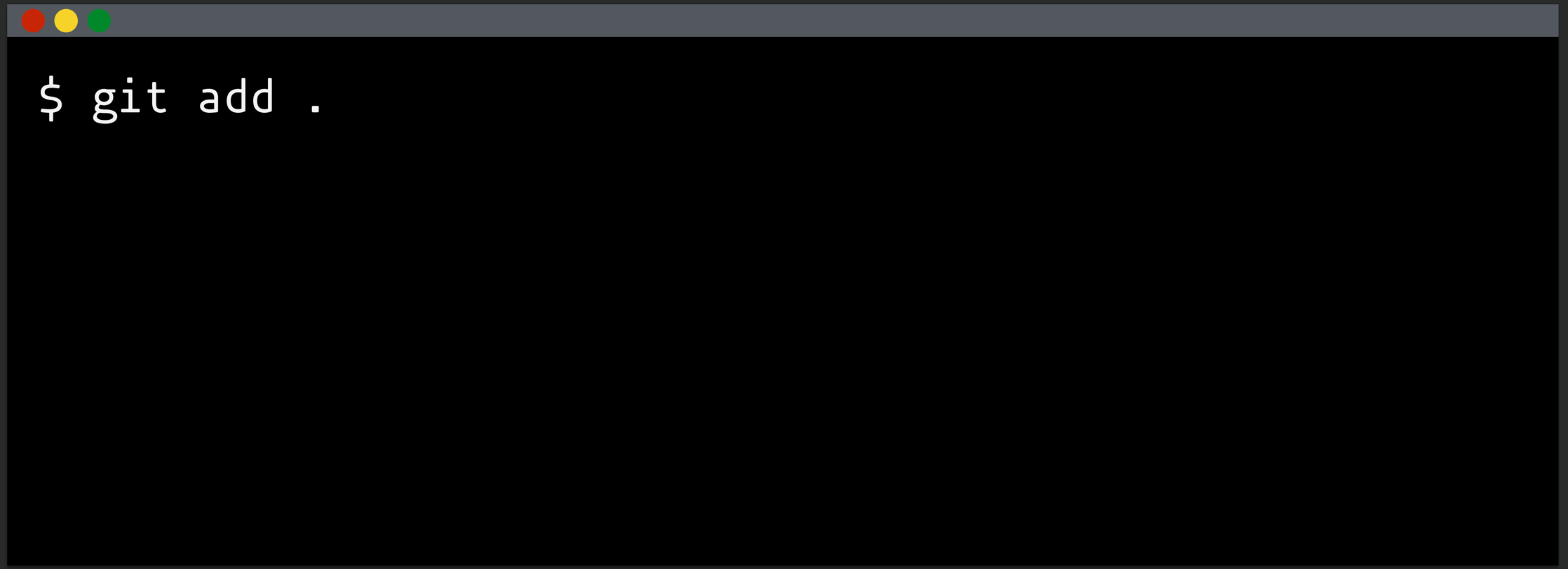
```go
func TestQuery_Execute(t *testing.T) {
    t.Run("A query can execute itself", func(t *testing.T) {
        var results struct {
            N int
        }

        driver := store.Connection
        defer func() { store.Connection = driver }()

        store.Connection = mock.Driver{
            Queryer: store.BoltQueryer(mock.NewBolt(mock.NewSession(
                mock.NewResult([]string{"n"}, [][]interface{}{{1}})))),
        }

        cypher := "MATCH (n) return n"
        q := store.Query{Statement: cypher, Results: &results}
        err := q.Execute()

        switch {
        case err != nil:
            t.Errorf("unexpected error running query: %v", err)
        case results.N != 1:
            t.Errorf("unexpected value running query: %d", results.N)
        }
    })
}
```

```
store.Connection = mock.Driver{
    Queryer: store.BoltQueryer(mock.NewBolt(mock.NewSession(
      mock.NewResult([]string{"n"}, [][]interface{}{{1}})))),
}
```

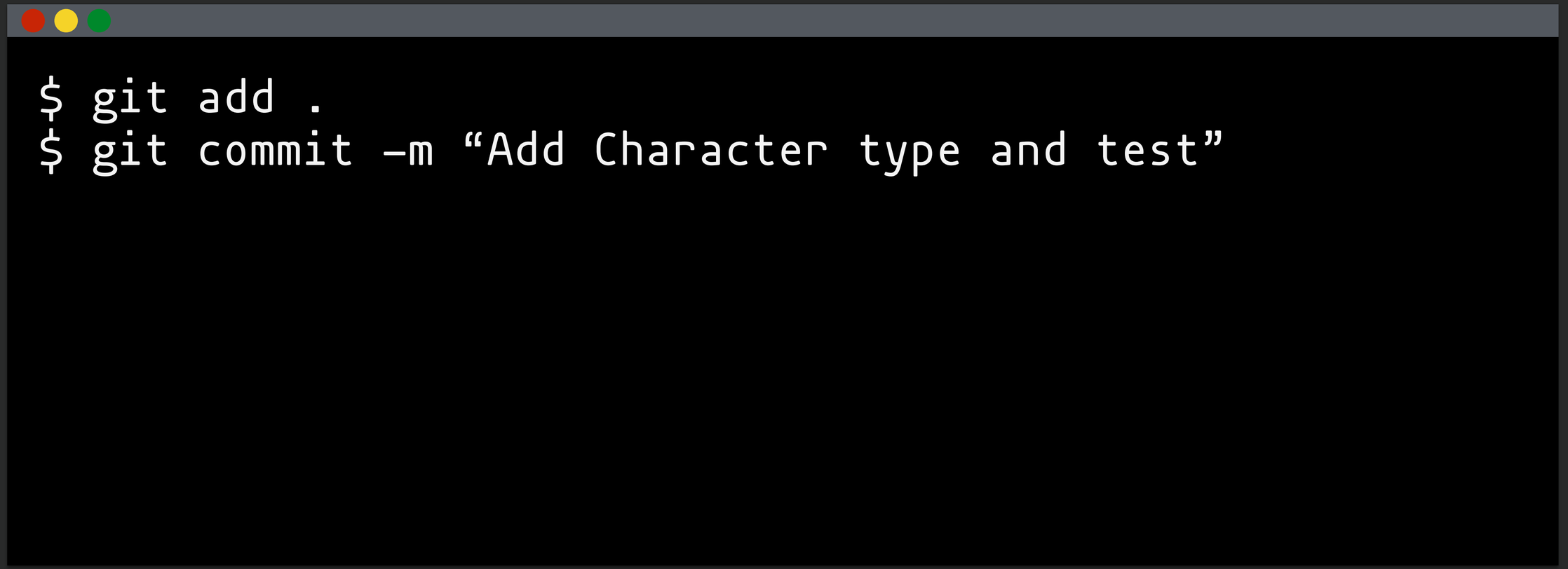
```
store.Connection = mock.Driver{
    QueryFn: store.MockQueryFn(mock.NewResult([][]interface{}{ })),
}
```

```go
func ExampleCharacter_Name() {
    c := NewCharacter("Unfortunate Conflict Of Evidence")

    fmt.Println(c.Name())

    // Output:
    // Unfortunate Conflict Of Evidence
}
```

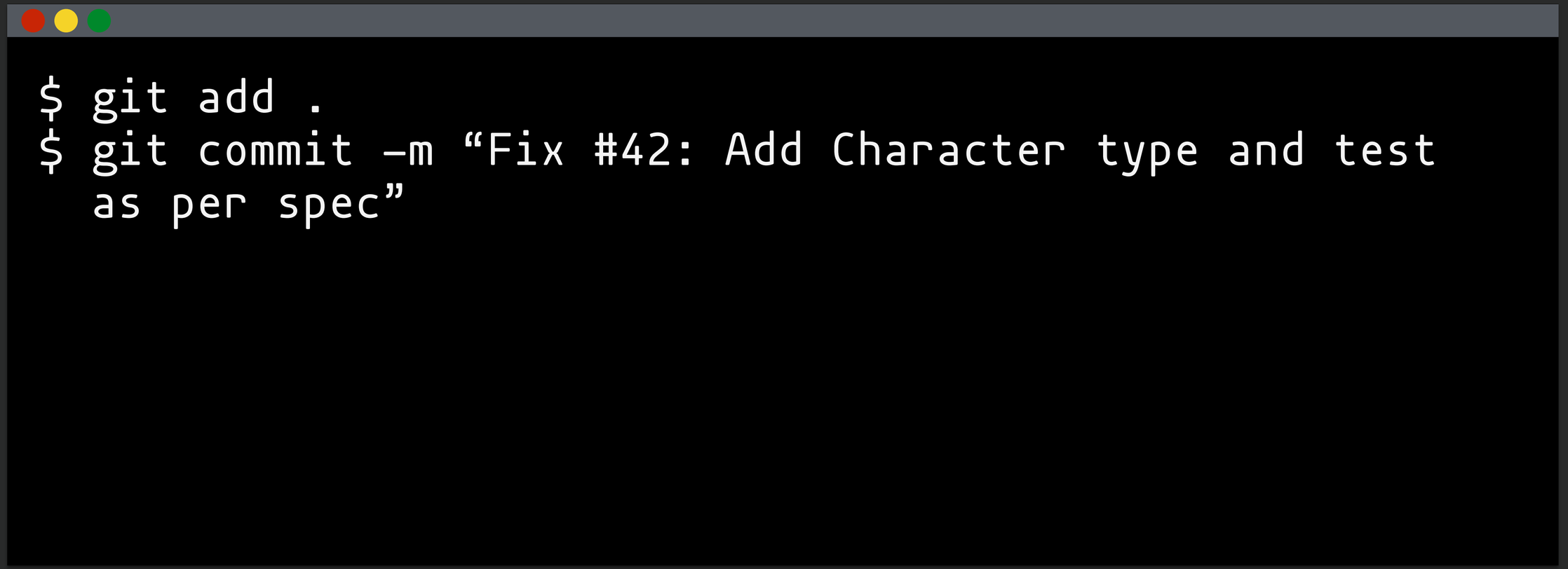
```
func ExampleCharacter_Name() {
    c := NewCharacter("Unfortunate Conflict Of Evidence")
```



```
func ExampleCharacter_Name() {
    c := NewCharacter("Unfortunate Conflict Of Evidence")
```



```
func ExampleCharacter_Name() {
    c := NewCharacter("Unfortunate Conflict Of Evidence")
```



```
// Person is a logical representation of a physical
// person.
type Person interface {
```

```
$ git add .
$ git commit -m “Fix #42 Implement person interface as
per spec”
```

Setup section of makefile no longer needed

Issue #4 RESOLVED

Dom Davis REPO OWNER created an issue 2019-10-22

It's all covered in the build image, and we don't want to make assumptions about local dev and how some tools are installed.

Comments (2)

Dom Davis REPORTER

- changed status to resolved

Fix #4: Remove setup target

→ <<cset 9adcdd0c4d40>>

Edit • Pin to top • Mark as spam • Delete • 2019-10-22

Dom Davis REPORTER

Merged in ISSUE-4 (pull request #4)

Fix #4: Remove setup target

Approved-by: Dom Davis dom@domdavis.com

→ <<cset 27895917be9d>>

Edit • Pin to top • Mark as spam • Delete • 2019-10-22

What would you like to say?

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// NewCharacter will create a new Character with the given name. If the name is
// empty then DefaultName will be used.
func NewCharacter(name string) Character {
    if name == "" {
        name = DefaultName
    }

    return Character{name: name}
}

// Name returns an arbitrary formatted string containing the moniker for the
// Character. Name will always be populated, but may be the DefaultName if no
// name was provided for the Character. No assumptions can be made about the
// format regarding case, title, number of names, or order of the names. It can
// be assumed that the Name can be displayed as is.
func (c Character) Name() string {
    if c.name == "" {
        c.name = DefaultName
    }

    return c.name
}
```

Code	Documentation

Code	Documentation
Changed	Changed

Code	Documentation
Changed	Not Changed

Code	Documentation
Not Changed	Changed

Code	Documentation
Not Changed	Not Changed

Code	Documentation
Changed	Broken Elsewhere

Dom Davis / gobase / Pull requests

Fix #4: Remove setup target

ISSUE-4 → master MERGED

Created 2019-10-22 · Last updated 2019-10-22

No reviewers Unapprove •••

Merged pull request

Merged in ISSUE-4 (pull request #4)

2789591 · Dom Davis · 2019-10-22

0 comments

Add a comment

2 commits

Makefile •••

```
@@ -1,29 +1,15 @@
- LINTER_VERSION=1.21.0
-
  all: clean build test vet lint

  ci: all report
```

Setup section of makefile no longer needed

Issue #4 RESOLVED

Dom Davis REPO OWNER created an issue 2019-10-22

It's all covered in the build image, and we don't want to make assumptions about local dev and how some tools are installed.

Comments (2)

Dom Davis REPORTER

- changed status to resolved

Fix #4: Remove setup target

→ <<cset 9adcdd0c4d40>>

Edit • Pin to top • Mark as spam • Delete • 2019-10-22

Dom Davis REPORTER

Merged in ISSUE-4 (pull request #4)

Fix #4: Remove setup target

Approved-by: Dom Davis dom@domdavis.com

→ <<cset 27895917be9d>>

Edit • Pin to top • Mark as spam • Delete • 2019-10-22

What would you like to say?

Fix #5: Backport linter config changes ISSUE-5 → master

Dom Davis - #5, created 2019-11-04, updated 2019-11-04

Fix #4: Remove setup target ISSUE-4 → master

Dom Davis - #4, created 2019-10-22, updated 2019-10-22

Fix #3: Remove redundant space ISSUE-3 → master

Dom Davis - #3, created 2019-10-18, updated 2019-10-18

Fix #2: Replace dummy badge URLs with real ones. ISSUE-2 → master

Dom Davis - #2, created 2019-10-18, updated 2019-10-18

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// NewCharacter will create a new Character with the given name. If the name is
// empty then DefaultName will be used.
func NewCharacter(name string) Character {
    if name == "" {
        name = DefaultName
    }

    return Character{name: name}
}

// Name returns an arbitrary formatted string containing the moniker for the
// Character. Name will always be populated, but may be the DefaultName if no
// name was provided for the Character. No assumptions can be made about the
// format regarding case, title, number of names, or order of the names. It can
// be assumed that the Name can be displayed as is.
func (c Character) Name() string {
    if c.name == "" {
        c.name = DefaultName
    }

    return c.name
}
```

```go
func ExampleCharacter_Name() {
    c := NewCharacter("  Youthful Indiscretion  ")

    fmt.Println(c.Name())

    // Output:
    // ?
}
```

```go
func ExampleCharacter_Name() {
    c := NewCharacter("  ")

    fmt.Println(c.Name())

    // Output:
    // ?
}
```

```
type Person struct {
    Name string
}
```

```go
package main

import "fmt"

// A Character in a book.
type Character struct {
    name string
}

// The DefaultName of a character, used when no name is set.
const DefaultName = "Unnamed character"

// NewCharacter will create a new Character with the given name. If the name is
// empty then DefaultName will be used.
func NewCharacter(name string) Character {
    if name == "" {
        name = DefaultName
    }

    return Character{name: name}
}

// Name returns an arbitrary formatted string containing the moniker for the
// Character. Name will always be populated, but may be the DefaultName if no
// name was provided for the Character. No assumptions can be made about the
// format regarding case, title, number of names, or order of the names. It can
// be assumed that the Name can be displayed as is.
func (c Character) Name() string {
    if c.name == "" {
        c.name = DefaultName
    }

    return c.name
}
```

The Three Hard Problems

The Three Hard Problems

1) Cache invalidation

The Three Hard Problems

1) Cache invalidation
2) Naming things

The Three Hard Problems

1) Cache invalidation
2) Naming things
3) Exactly once delivery

The Three Hard Problems

1) Cache invalidation
2) Naming things
3) Exactly once delivery
2) Naming things

The Three Hard Problems

1) Cache invalidation
2) Naming things
3) Exactly once delivery
2) Naming things
4) Off by one errors

```
config.Config.Config()
```

```
config.Config.Config()
```

```
config.Global.Config()
```

```
config.Global.Config()
```

```
settings.Global.Config()
```

```
settings.Global.Config()
```

```
let validate = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\
\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
```

What have we learned?

What have we learned?

We suck at comments

What have we learned?

We suck at comments
We suck at documentation

What have we learned?

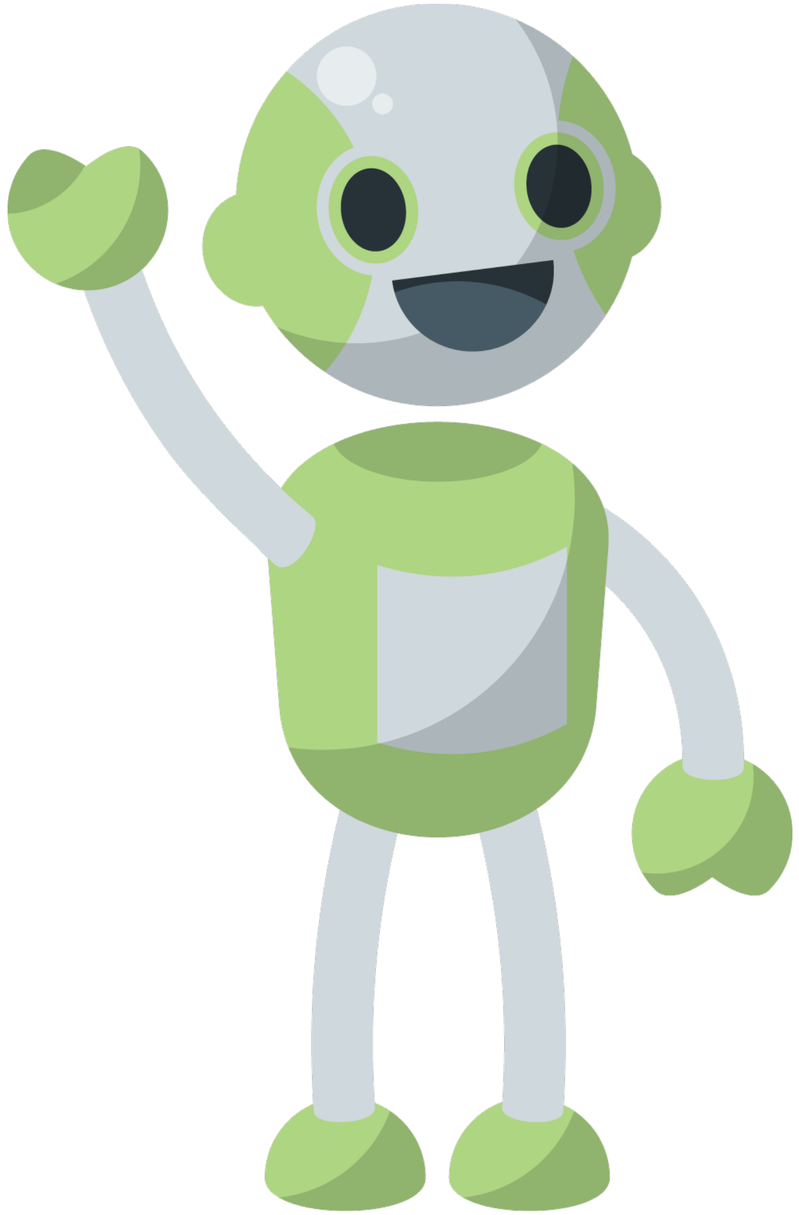
We suck at comments
We suck at documentation
We suck at git commits

What have we learned?

We suck at comments
We suck at documentation
We suck at git commits
We suck at naming things

What have we learned?

We suck at comments
We suck at documentation
We suck at git commits
We suck at naming things
We suck at software development



```
// Thing is what?
type Thing interface{
    // Action does what?
    Action() int
}
```

```
// Operation is a single operation on the system.
type Operation interface {
    // Action does what?
    ID() int
}
```

```go
// Operation is used to provide the context around an
// request made to the system.
type Operation interface {
    // ID of the operation.
    ID() int
}
```

```
// Context of a request. Each request is given a context
// with a unique ID. Data surrounding the request is
// stored in the context.
type Context interface {
    // ID of the context.
    ID() int
}
```

```
// Context of a request. Each request is given a context
// with a unique ID. Data surrounding the request is
// stored in the context.
type Context interface {
    // ID of the context. The ID will be unique and is
    // static.
    ID() int
}
```

```
// Context of a request. Each request is given a context
// with a unique ID. Data surrounding the request is
// stored in the context.
type Context interface {
    // UUID of the context. The UUID is set when the
    // context is created.
    UUID() int
}
```

```
func ExampleUUID() {
    var context Context

    fmt.Println(context.UUID() != "")

    // Output:
    // true

}
```

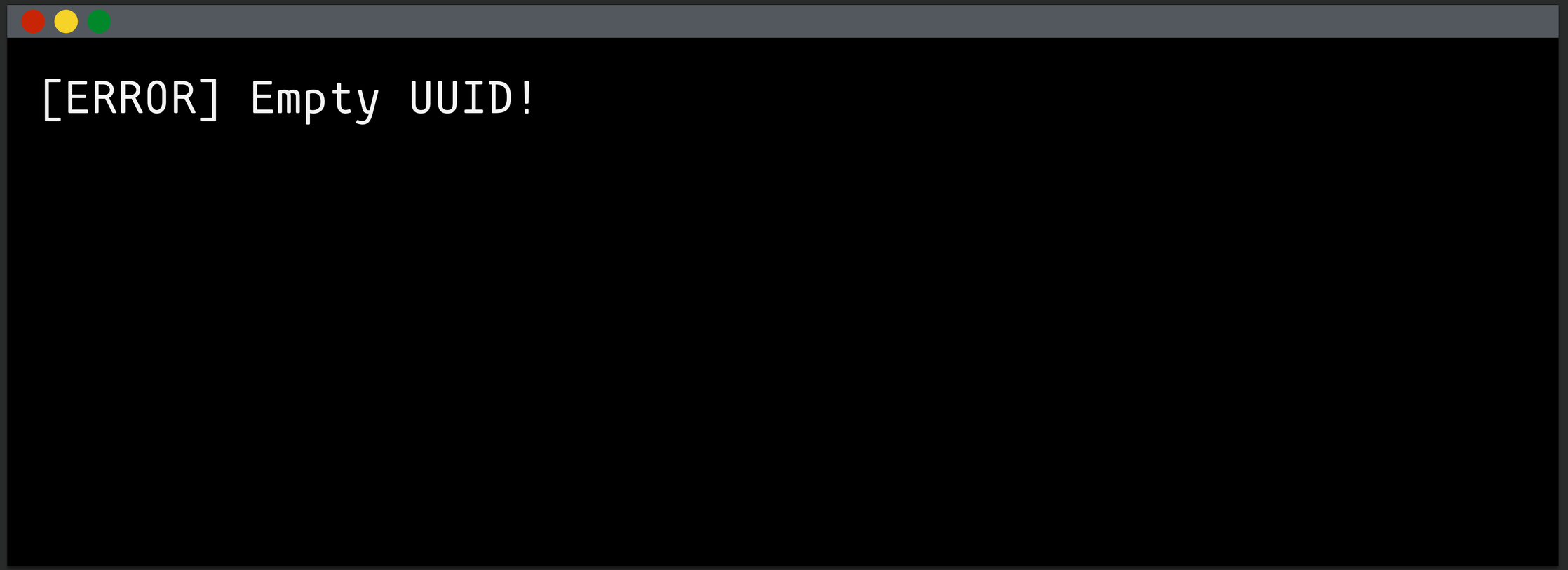
```
// Context of a request. Each request is given a context
// with a unique ID. Data surrounding the request is
// stored in the context
```

```
$ git add .
$ git commit -m “ISSUE-1: Define base Context Interface”
```

Base Architecture

Server

Simple HTTP server, with a single login endpoint which will set a context so we can store some details. The context will get passed about in a JWT token and popped into a type that will be given to each request.


[ERROR] Empty UUID!

…happily ever after.

THE END