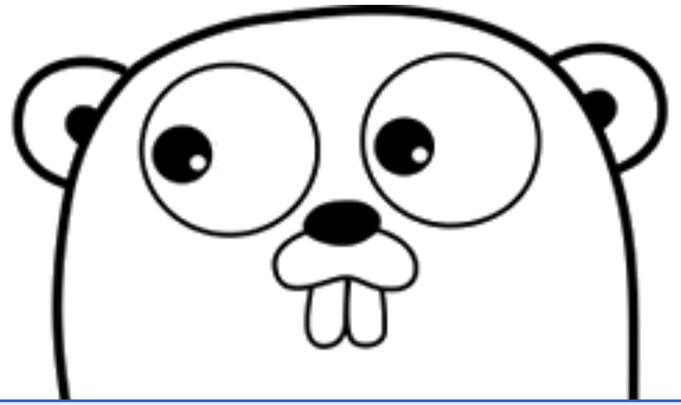
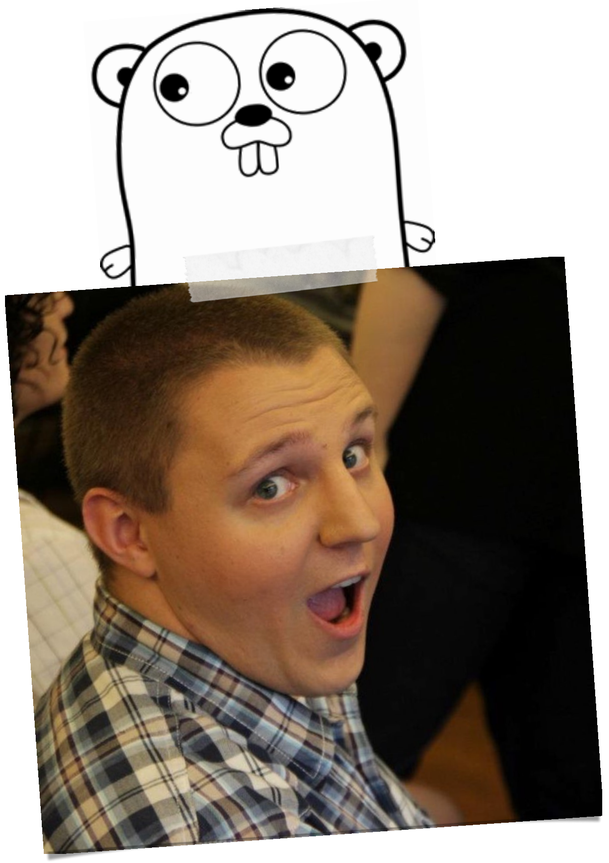
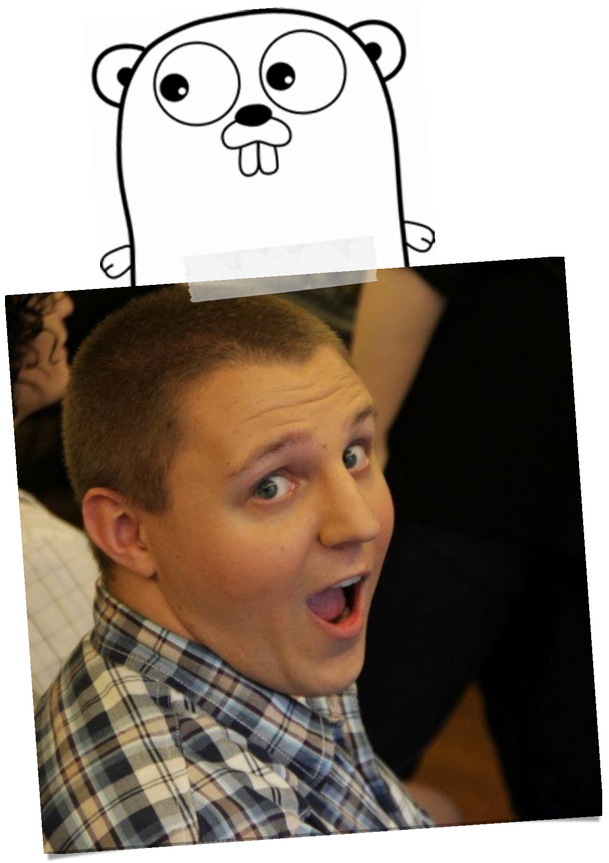


INTRO TO GO



SETH VARGO

@sethvargo



SETH VARGO

@sethvargo

HISTORY

2007

2007

"MODERN"

C and Java

Ruby and Python

NodeJS

Scala

JavaScript

Objective-C/Swift

2007

"MODERN"

C and Java

Ruby and Python

NodeJS

Scala

JavaScript

Objective-C/Swift

PROBLEMS

Pick one:

- efficient compilation
- efficient execution
- ease of development

Type safety

Modern

2007 2008

"MODERN"

C and Java

Ruby and Python

NodeJS

Scala

JavaScript

Objective-C/Swift

PROBLEMS

Pick one:

- efficient compilation
- efficient execution
- ease of development

Type safety

Modern

REQUIREMENTS

Compile FAST

Execute FAST

Concurrency

Built-in GC

Support for dependencies

Good enough for Google

2007 2008

"MODERN"
C and Java
Ruby and Python
NodeJS
Scala
JavaScript
Objective-C/Swift

PROBLEMS
Pick one:
- efficient compilation
- efficient execution
- ease of development
Type safety
Modern

REQUIREMENTS
Compile FAST
Execute FAST
Concurrency
Built-in GC
Support for dependencies
Good enough for Google

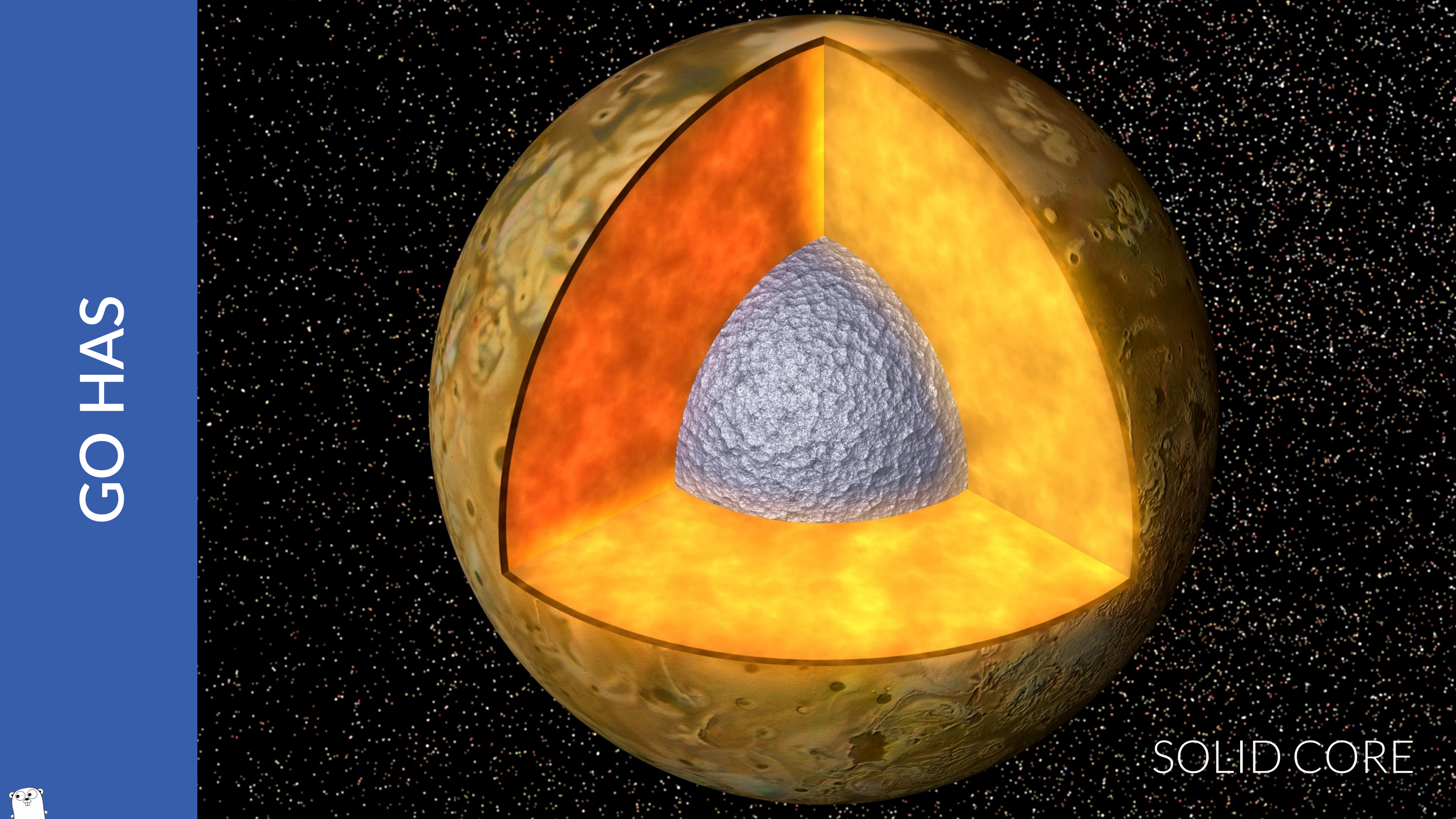
"MODERN"
C and Java
Ruby and Python
NodeJS
Scala
JavaScript
Objective-C/Swift

PROBLEMS
Pick one:
- efficient compilation
- efficient execution
- ease of development
Type safety
Modern

REQUIREMENTS
Compile FAST
Execute FAST
Concurrency
Built-in GC
Support for dependencies
Good enough for Google

OVERVIEW

GO HAS



SOLID CORE

GO HAS

A ROBUST COMMUNITY

GO HAS

INCREDIBLE PERFORMANCE

GO HAS

```
package main

import (
"bytes"
"fmt"
"io/ioutil"
"path/filepath"
"sync"
"text/template"
dep "github.com/hashicorp/consul-template/dependency"
)

type Template struct {
sync.Mutex
// Path is the path to this template on disk.
Path string
// dependencies is the internal list of dependencies.
dependencies []dep.Dependency
// contents is string contents for this file when read from di
contents string
}

// NewTemplate creates and parses a new Consul Template templa
// path. If the template does not exist, an error is returned.
// initialization, the template is read and is parsed for depe
// errors that occur are returned.
func NewTemplate(path string) (*Template, error) {
template := &Template{Path: path}
if err := template.init(); err != nil {
return nil, err
}

return template, nil
}

// Execute evaluates this template in the context of the given
// return value is a slice of missing dependencies that were e
// during evaluation. If there are any missing dependencies fo
// considered unsafe to write the resulting contents to disk.
// value is the contents of the template as-rendered. This val
// final resulting template if there are any missing dependenc
// return value is any error that occurred while evaluating t
func (t *Template) Execute(brain *Brain) ([]dep.Dependency, []
usedMap := make(map[string]dep.Dependency)
```

```
1  package main
2
3  import (
4    "bytes"
5    "fmt"
6    "io/ioutil"
7    "path/filepath"
8    "sync"
9    "text/template"
10
11   dep "github.com/hashicorp/consul-template/dependency"
12 )
13
14 type Template struct {
15   sync.Mutex
16   // Path is the path to this template on disk.
17   Path string
18   // dependencies is the internal list of dependencies.
19   dependencies []dep.Dependency
20   // contents is string contents for this file when read from disk.
21   contents string
22 }
23
24 // NewTemplate creates and parses a new Consul Template template at the gi
25 // path. If the template does not exist, an error is returned. During
26 // initialization, the template is read and is parsed for dependencies. An
27 // errors that occur are returned.
28 func NewTemplate(path string) (*Template, error) {
29   template := &Template{Path: path}
30   if err := template.init(); err != nil {
31     return nil, err
32   }
33
34   return template, nil
35 }
36
37 // Execute evaluates this template in the context of the given brain. The
38 // return value is a slice of missing dependencies that were encountered
39 // during evaluation. If there are any missing dependencies found, it shou
40 // considered unsafe to write the resulting contents to disk. The second r
41 // value is the contents of the template as-rendered. This value may not b
42 // final resulting template if there are any missing dependencies. The las
43 // return value is any error that occurred while evaluating the template.
44 func (t *Template) Execute(brain *Brain) ([]dep.Dependency, []byte, error)
```

BUILT-IN LINTING AND FORMATTING

GO HAS

EASY CROSS-PLATFORM BINARIES

GO HAS

INTERNAL GC

GO DOES NOT HAVE

golang.org

The Go Programming Language

Documents | Packages | The Project | Help | Blog | Search

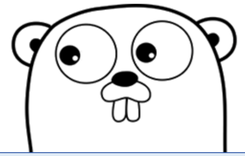
Try Go

Pop-out

```
// You can edit this code!
// Click here and start typing.
package main

import "fmt"

func main() {
        fmt.Println("Hello, 世界")
}
```

Hello, World! | Run | Share | Tour

Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.



Download Go

Binary distributions available for Linux, Mac OS X, Windows, and more.

Featured video

Featured articles

Package names

Go code is organized into packages. Within a package, code can refer to any identifier (name) defined within, while clients of the package may only reference the package's exported types, functions, constants, and variables. Such references always include the package name as a prefix: `foo.Bar` refers to the exported name `Bar` in the imported package named `foo`.

Published 4 February 2015

Errors are values

A common point of discussion among Go programmers, especially those new to the language, is how to handle errors. The conversation often turns into a lament at the number of times the sequence

Published 12 January 2015

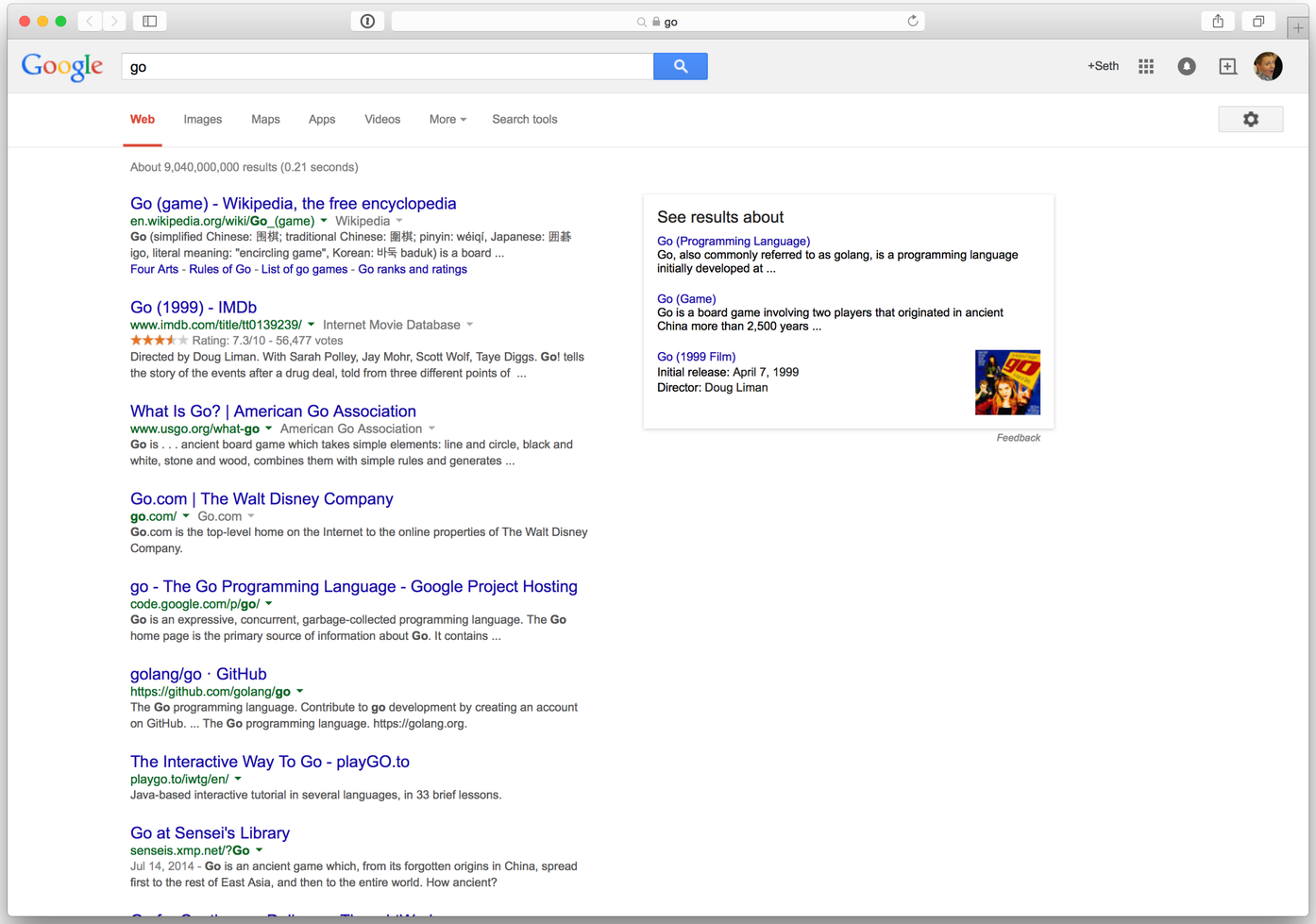
Read more

A PRETTY WEBSITE

GO DOES NOT HAVE

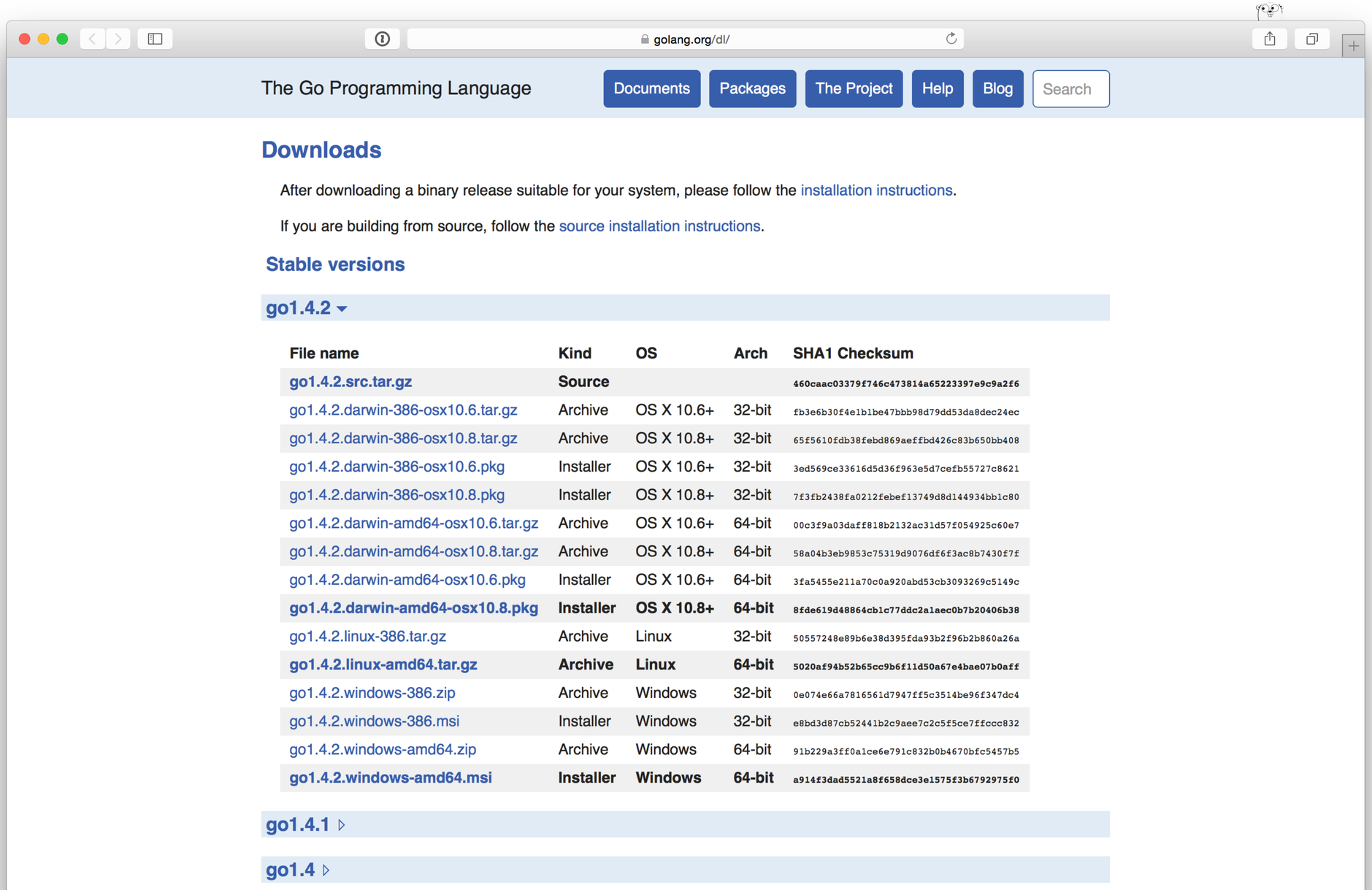
GENERICS

GO DOES NOT HAVE



SEO

INSTALL



Downloads

After downloading a binary release suitable for your system, please follow the installation instructions.

If you are building from source, follow the source installation instructions.

Stable versions

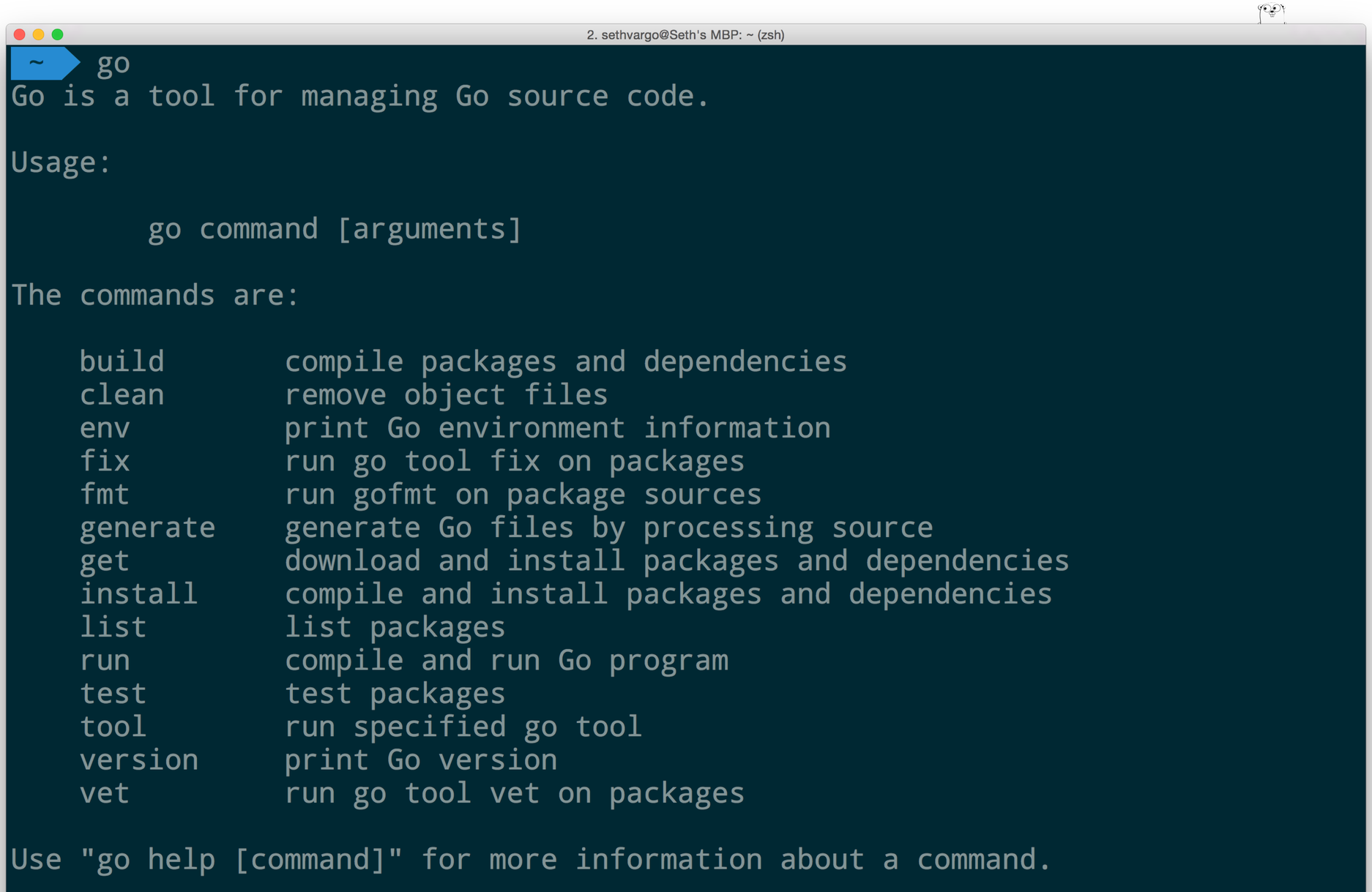
go1.4.2

File name	Kind	OS	Arch	SHA1 Checksum
go1.4.2.src.tar.gz	**Source**			**460caac03379f746c473814a65223397e9c9a2f6**
go1.4.2.darwin-386-osx10.6.tar.gz	Archive	OS X 10.6+	32-bit	fb3e6b30f4e1b1be47bbb98d79dd53da8dec24ec
go1.4.2.darwin-386-osx10.8.tar.gz	Archive	OS X 10.8+	32-bit	65f5610fdb38febd869aeffbd426c83b650bb408
go1.4.2.darwin-386-osx10.6.pkg	Installer	OS X 10.6+	32-bit	3ed569ce33616d5d36f963e5d7cefb55727c8621
go1.4.2.darwin-386-osx10.8.pkg	Installer	OS X 10.8+	32-bit	7f3fb2438fa0212febef13749d8d144934bb1c80
go1.4.2.darwin-amd64-osx10.6.tar.gz	Archive	OS X 10.6+	64-bit	00c3f9a03daff818b2132ac31d57f054925c60e7
go1.4.2.darwin-amd64-osx10.8.tar.gz	Archive	OS X 10.8+	64-bit	58a04b3eb9853c75319d9076df6f3ac8b7430f7f
go1.4.2.darwin-amd64-osx10.6.pkg	Installer	OS X 10.6+	64-bit	3fa5455e211a70c0a920abd53cb3093269c5149c
go1.4.2.darwin-amd64-osx10.8.pkg	**Installer**	**OS X 10.8+**	**64-bit**	**8fde619d48864cb1c77ddc2a1aec0b7b20406b38**
go1.4.2.linux-386.tar.gz	Archive	Linux	32-bit	50557248e89b6e38d395fda93b2f96b2b860a26a
go1.4.2.linux-amd64.tar.gz	**Archive**	**Linux**	**64-bit**	**5020af94b52b65cc9b6f11d50a67e4bae07b0aff**
go1.4.2.windows-386.zip	Archive	Windows	32-bit	0e074e66a7816561d7947ff5c3514be96f347dc4
go1.4.2.windows-386.msi	Installer	Windows	32-bit	e8bd3d87cb52441b2c9aee7c2c5f5ce7ffccc832
go1.4.2.windows-amd64.zip	Archive	Windows	64-bit	91b229a3ff0a1ce6e791c832b0b4670bfc5457b5
go1.4.2.windows-amd64.msi	**Installer**	**Windows**	**64-bit**	**a914f3dad5521a8f658dce3e1575f3b6792975f0**

go1.4.1

go1.4

$GOPATH


2. sethvargo@Seth's MBP: ~ (zsh)
~ go
Go is a tool for managing Go source code.
Usage:
go command [arguments]
The commands are:
build compile packages and dependencies
clean remove object files
env print Go environment information
fix run go tool fix on packages
fmt run gofmt on package sources
generate generate Go files by processing source
get download and install packages and dependencies
install compile and install packages and dependencies
list list packages
run compile and run Go program
test test packages
tool run specified go tool
version print Go version
vet run go tool vet on packages
Use "go help [command]" for more information about a command.

```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go

```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go

```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go

```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

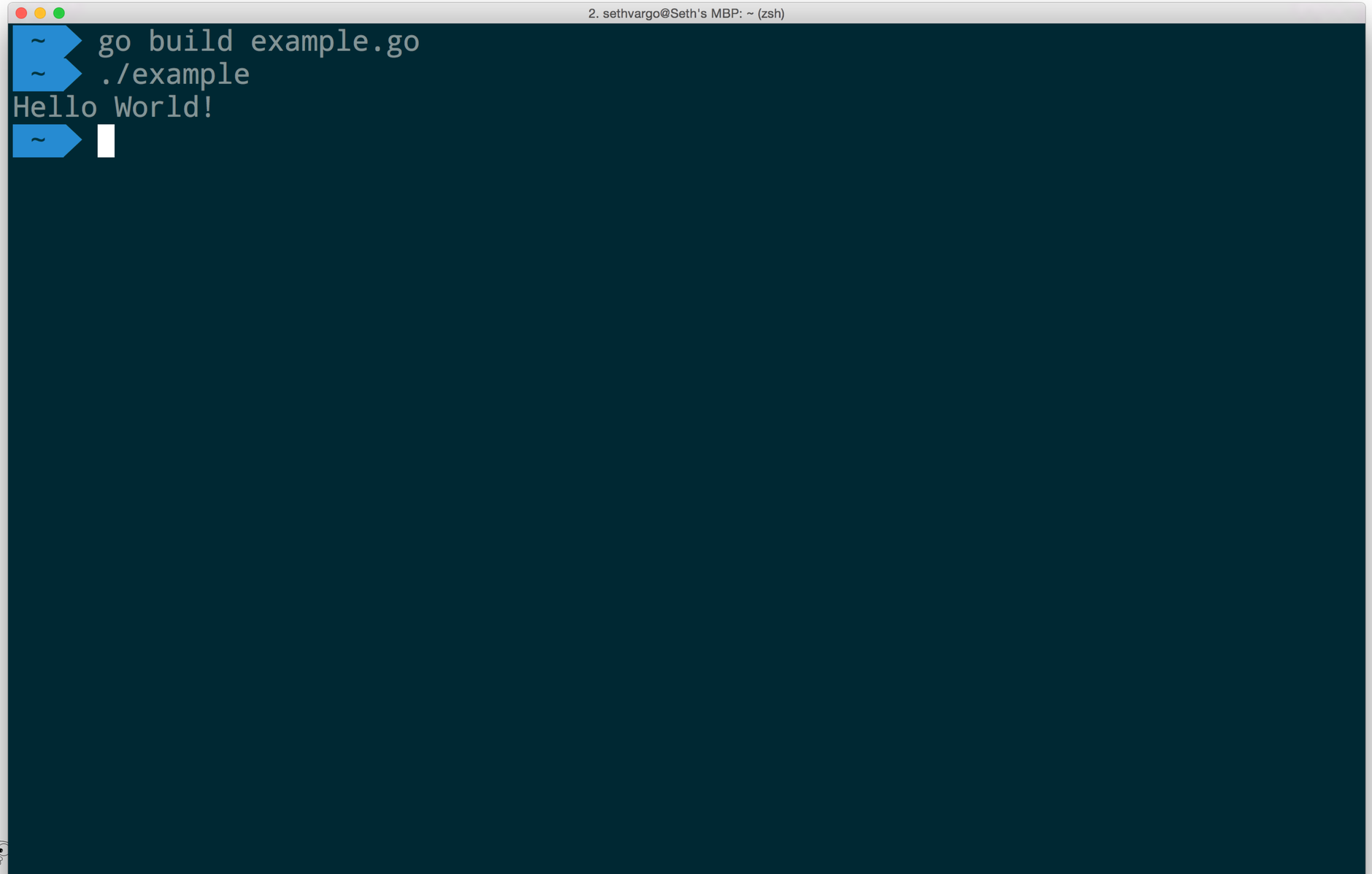
example.go

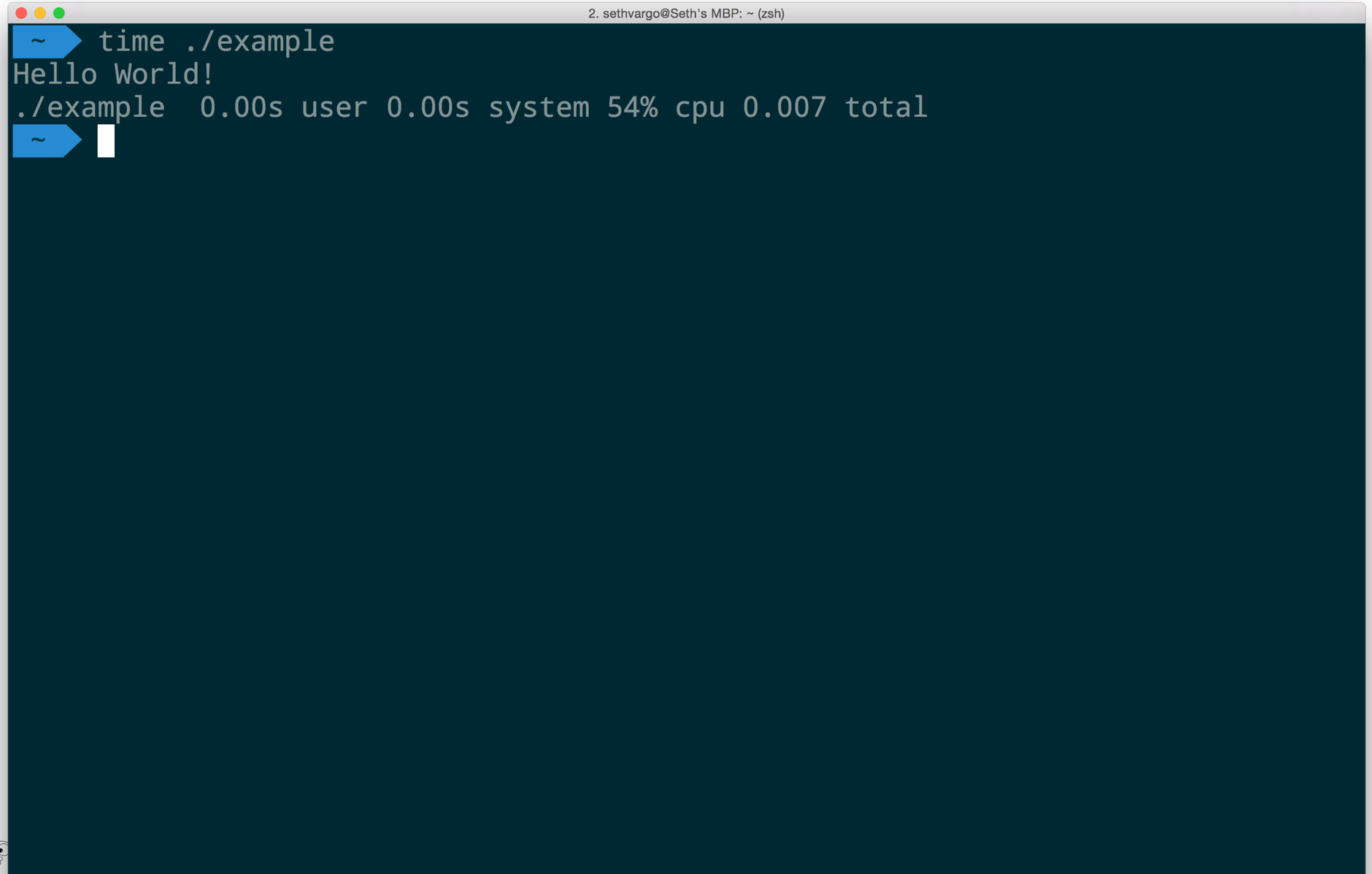
```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go

```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go


2. sethvargo@Seth's MBP: ~ (zsh)
~ go build example.go
~ ./example
Hello World!
~


2. sethvargo@Seth's MBP: ~ (zsh)
~ time ./example
Hello World!
./example 0.00s user 0.00s system 54% cpu 0.007 total
~

```go
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```go
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```go
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```go
package main

func main() {
   reader := bufio.NewReader(os.Stdin)

   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

func main() {
   reader := bufio.NewReader(os.Stdin)

   text, err := reader.ReadString('\n')

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main




func main() {

   text, err := reader.ReadString('\n')


   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main




func main() {

   text,     := "Hi\n"


   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main




func main() {

   text,     := "Hi\n"


   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace("Hi\n")))
}
```

example.go

```
package main




func main() {

   text,     := "Hi\n"


   fmt.Println(fmt.Sprintf("Your name is %q!", "Hi"))
}
```

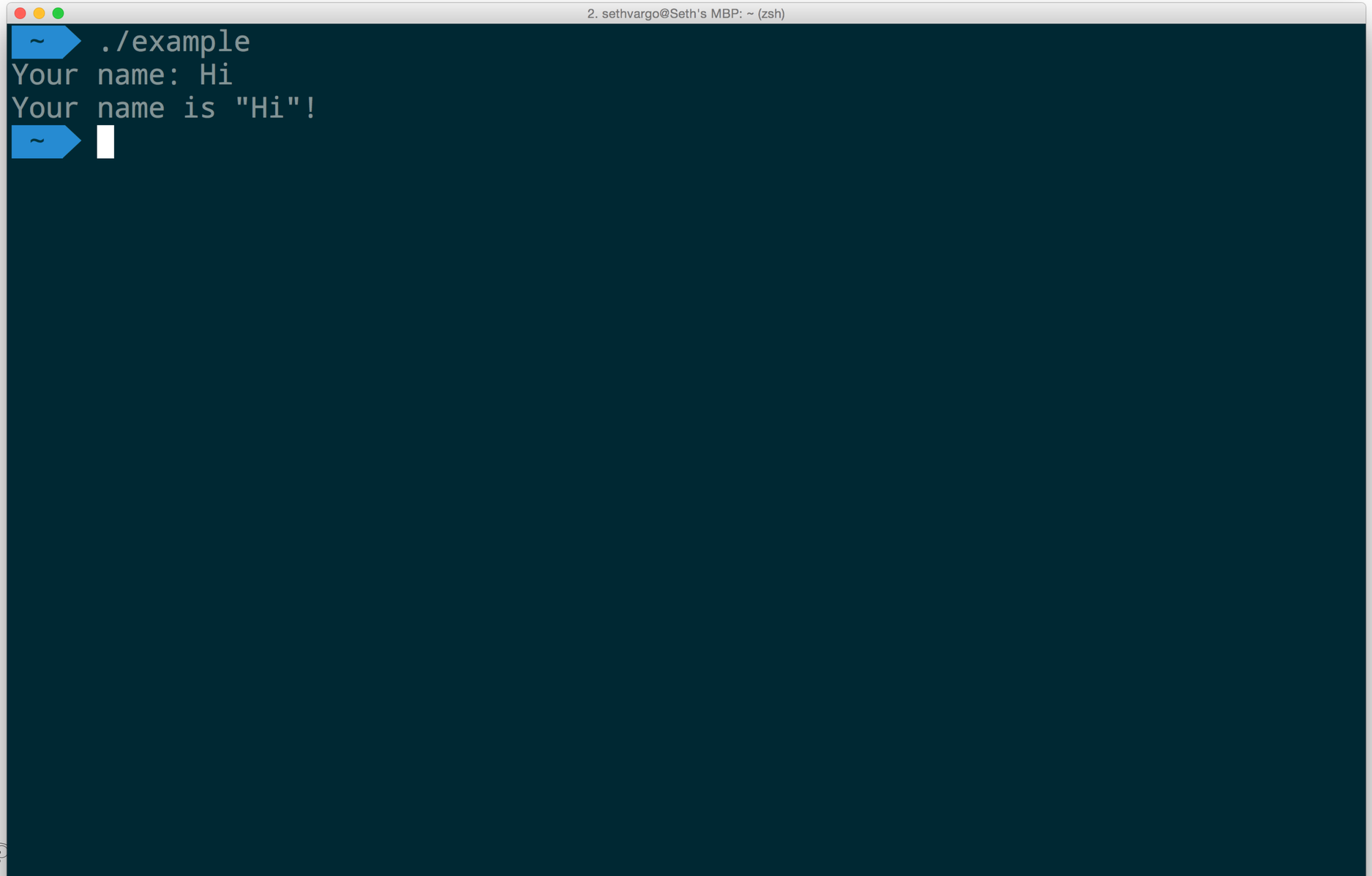
example.go

```
package main




func main() {

   text,     := "Hi\n"


   fmt.Println(fmt.Sprintf("Your name is \"Hi\"!"))
}
```

example.go

```
package main




func main() {

   text,     := "Hi\n"


   fmt.Println("Your name is \"Hi\"!")
}
```

example.go


2. sethvargo@Seth's MBP: ~ (zsh)
~ ./example
Your name: Hi
Your name is "Hi"!
~

```go
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```go
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

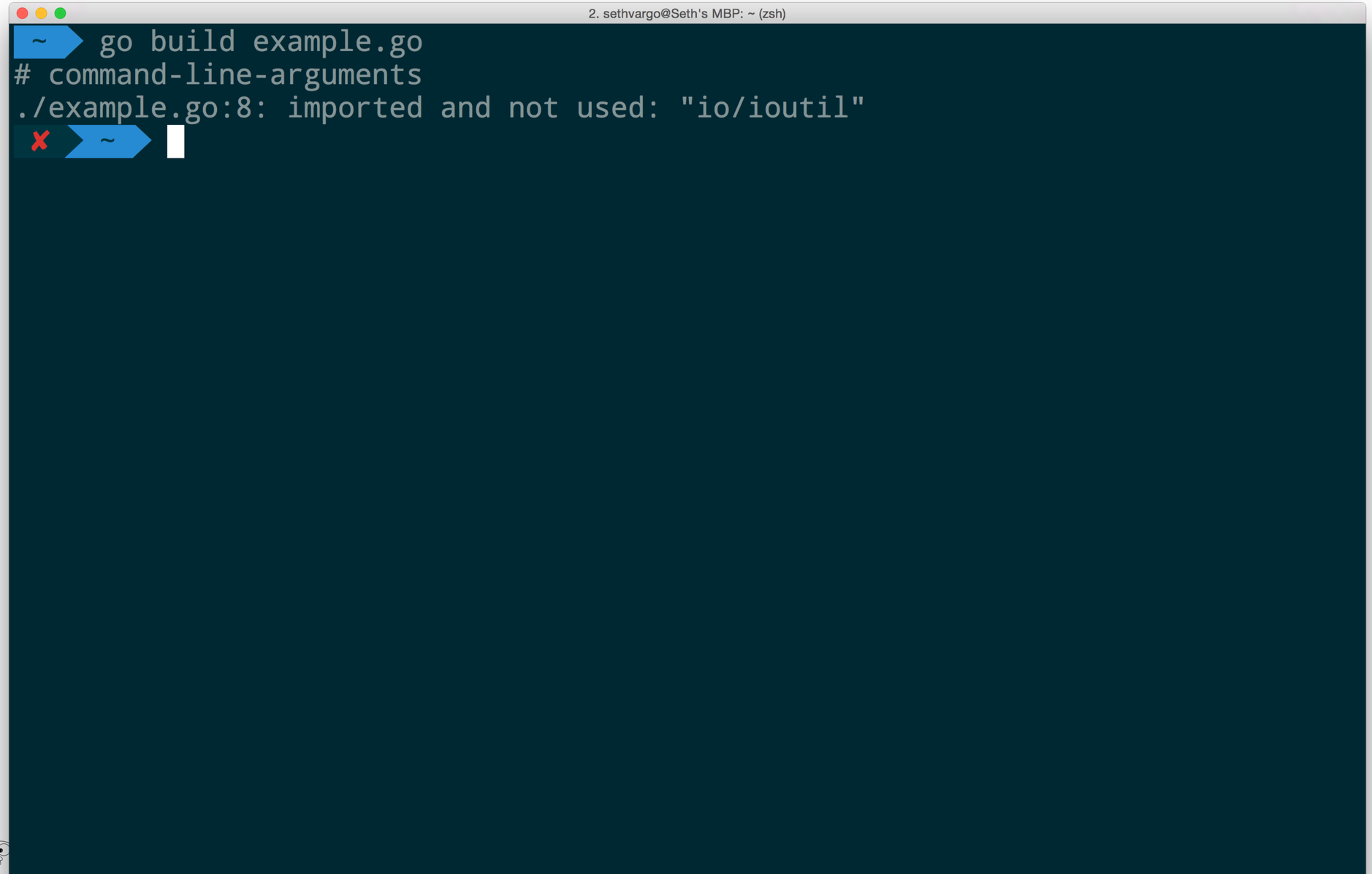
example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go

```
package main

import (
   "bufio"
   "fmt"
   "os"
   "strings"
   "io/ioutil"
)

func main() {
   reader := bufio.NewReader(os.Stdin)

   fmt.Print("Your name: ")
   text, err := reader.ReadString('\n')
   if err != nil {
      fmt.Println("error:", err.Error())
      return
   }

   fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go


2. sethvargo@Seth's MBP: ~ (zsh)
~ go build example.go
# command-line-arguments
./example.go:8: imported and not used: "io/ioutil"
✘ ~

FIZZBUZZ

```
def fizzbuzz(max)
  1.upto(max) do |i|
    if i % 5 == 0 and i % 3 == 0
      puts "FizzBuzz"
    elsif i % 3== 0
      puts "Fizz"
    elsif i % 5 == 0
      puts "Buzz"
    else
      puts i
    end
  end
end
```

fizzbuzz.rb

```
func fizzbuzz(max int) {
  for i := 1; i <= max; i++ {
    if i %3 == 0 && i %5 == 0 {
      fmt.Println("Fizzbuzz")
    } else if i%3 == 0 {
      fmt.Println("Fizz")
    } else if i%5 == 0 {
      fmt.Println("Buzz")
    } else {
      fmt.Println(i)
    }
  }
}
```

fizzbuzz.go

```go
func fizzbuzz(max int) {
  for i := 1; i <= max; i++ {
    if i %3 == 0 && i %5 == 0 {
      fmt.Println("Fizzbuzz")
    } else if i%3 == 0 {
      fmt.Println("Fizz")
    } else if i%5 == 0 {
      fmt.Println("Buzz")
    } else {
      fmt.Println(i)
    }
  }
}
```

fizzbuzz.go

```ruby
def fizzbuzz(max)
  1.upto(max) do |i|
    if i % 5 == 0 and i % 3 == 0
      puts "FizzBuzz"
    elsif i % 3== 0
      puts "Fizz"
    elsif i % 5 == 0
      puts "Buzz"
    else
      puts i
    end
  end
end
```

fizzbuzz.rb

```go
func fizzbuzz(max int) {
  for i := 1; i <= max; i++ {
    if i %3 == 0 && i %5 == 0 {
      fmt.Println("Fizzbuzz")
      continue
    }

    if i%3 == 0 {
      fmt.Println("Fizz")
      continue
    }

    if i%5 == 0 {
      fmt.Println("Buzz")
      continue
    }

    fmt.Println(i)
  }
}
```

fizzbuzz.go

```ruby
def fizzbuzz(max)
  1.upto(max) do |i|
    if i % 5 == 0 and i % 3 == 0
      puts "FizzBuzz"
      next
    end

    if i % 3== 0
      puts "Fizz"
      next
    end

    if i % 5 == 0
      puts "Buzz"
      next
    end

    puts i
  end
end
```

fizzbuzz.rb

FIBONACCI

```
def fib(max)
  if max < 2
    return max
  else
    return fib(max-1) + fib(max-2)
  end
end
```

fib.rb

```
func fib(max int) int {
  if max < 2 {
    return max
  } else {
    return fib(max-1) + fib(max-2)
  }
}
```

fib.go

```go
func fib(max int) int {
  if max < 2 {
    return max
  } else {
    return fib(max-1) + fib(max-2)
  }
}
```

fib.go

```ruby
def fib(max)
  if max < 2
    return max
  else
    return fib(max-1) + fib(max-2)
  end
end
```

fib.rb

```go
func fib(max int) int {
  if max < 2 {
    return max
  } else {
    return fib(max-1) + fib(max-2)
  }
}

fib(40)
```

fib.go

```ruby
def fib(max)
  if max < 2
    return max
  else
    return fib(max-1) + fib(max-2)
  end
end

fib(40)
```

fib.rb

```go
func fib(max int) int {
  if max < 2 {
    return max
  } else {
    return fib(max-1) + fib(max-2)
  }
}


fib(40)
  1.02s user
  0.01s system
  99% cpu
  1.027 total
```

fib.go

```ruby
def fib(max)
  if max < 2
    return max
  else
    return fib(max-1) + fib(max-2)
  end
end


fib(40)
```

fib.rb

```go
func fib(max int) int {
  if max < 2 {
    return max
  } else {
    return fib(max-1) + fib(max-2)
  }
}
```

```
fib(40)
  1.02s user
  0.01s system
  99% cpu
  1.027 total
```

fib.go

```ruby
def fib(max)
  if max < 2
    return max
  else
    return fib(max-1) + fib(max-2)
  end
end
```

```
fib(40)
  17.37s user
  0.04s system
  99% cpu
  17.467 total
```

fib.rb

```
func fib(max int) int {
  i, j := 0, 1
  for c := 0; c < max; c++ {
    i, j = i+j, i
  }
  return i
}


fib(40)
```

fib.go

```
func fib(max int) int {
  i, j := 0, 1
  for c := 0; c < max; c++ {
    i, j = i+j, i
  }
  return i
}


fib(40)
  0.00s user
  0.00s system
  74% cpu
  0.004 total
```

fib.go

```
func fib(max int) int {
  i, j := 0, 1
  for c := 0; c < max; c++ {
    i, j = i+j, i
  }
  return i
}


fib(1000)
```

fib.go

```
func fib(max int) int {
  i, j := 0, 1
  for c := 0; c < max; c++ {
    i, j = i+j, i
  }
  return i
}


fib(1000)
  0.00s user
  0.00s system
  72% cpu
  0.004 total
```

fib.go

```
func fib(max int) int {
  i, j := 0, 1
  for c := 0; c < max; c++ {
    i, j = i+j, i
  }
  return i
}


fib(1000)
  0.00s user
  0.00s system
  72% cpu
  0.004 total

43466557686937456435688527675040625802564660517371780402481729089536555417
94905189040387984007925516929592259308032263477520968962323987332247116164
29964409065331879382989696499285160037044761377951668492288 75
```

fib.go

```
func fibGenerator()      ?     {




  return ?
}
```

fib.go

```
func fibGenerator()      ?     {


      (some closureyish thing)



  return ?
}
```

fib.go

```
func fibGenerator()      ?     {



    for i, j := 0, 1; ; i, j = i+j, i {
      // do something with i
    }


  return ?
}
```

fib.go

```
func fibGenerator()      ?     {


     func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      // do something with i
    }
  }()

  return ?
}
```

fib.go

```
func fibGenerator()      ?     {


  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      // do something with i
    }
  }()

  return ?
}
```

fib.go

```
func fibGenerator()      ?     {
  c := someMagicalGenerator()

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      // do something with i
    }
  }()

  return ?
}
```

fib.go

```
func fibGenerator()      ?     {
  c := someMagicalGenerator()

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c.send(i)
    }
  }()

  return ?
}
```

fib.go

```
func fibGenerator()      ?     {
  c := someMagicalGenerator()

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c.send(i)
    }
  }()

  return c
}
```

fib.go

```
someMagicalGenerator() ???
```

fib.go


CH 03

```
func fibGenerator()      ?     {
  c := someMagicalGenerator()

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c.send(i)
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator()      ?     {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c.send(i)
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c.send(i)
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c.send(i)
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}

func main() {
  fib := fibGenerator()
  for i := 0; i < 25; i++ {
    fmt.Println(  ?  )
  }
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}

func main() {
  fib := fibGenerator()
  for i := 0; i < 25; i++ {
    fmt.Println(<-fib)
  }
}
```

fib.go

```
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}

func main() {
  fib := fibGenerator()
  for i := 0; i < 25; i++ {
    fmt.Println(<-fib)
  }
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}

func main() {
  fib1 := fibGenerator()
  fib2 := fibGenerator()

  for i := 0; i < 5; i++ {
    fmt.Println(<-fib1)
  }

  for i := 0; i < 5; i++ {
    fmt.Println(<-fib2)
  }
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}

func main() {
  fib1 := fibGenerator()
  fib2 := fibGenerator()

  for i := 0; i < 5; i++ {
    fmt.Println(<-fib1)
  }

  for i := 0; i < 5; i++ {
    fmt.Println(<-fib2)
  }
}
```

fib.go

```
go func() {
  for i, j := 0, 1; ; i, j = i+j, i {
    c <- i
  }
}()
```

fib.go

```
for i, j := 0, 1; ; i, j = i+j, i {
  c <- i
}
```

fib.go

```
for i, j := 0, 1; ; i, j = i+j, i {
  c <- i
}
```

fib.go

```
for {
  c <- i
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int, 50)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int, 50)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}
```

fib.go

```
func fibGenerator() <-chan int {
  c := make(chan int, 50)

  go func() {
    for i, j := 0, 1; ; i, j = i+j, i {
      c <- i
    }
  }()

  return c
}
```

fib.go

STRUCTS

```
type fib struct {
  // buffer is the size of the channel buffer.
  buffer int

  // ch is the internal channel where numbers are written.
  ch chan int
}
```

structs.go

```
type Fib struct {
  // Buffer is the size of the channel buffer.
  Buffer int

  // Ch is the internal channel where numbers are written.
  Ch chan int
}
```

structs.go

```
type Fib struct {
  // Buffer is the size of the channel buffer.
  Buffer int

  // Ch is the internal channel where numbers are written.
  Ch chan int
}

func PublicFunc() {}

func privateFunc() {}
```

structs.go

```
type fib struct {
  // buffer is the size of the channel buffer.
  buffer int

  // ch is the internal channel where numbers are written.
  ch chan int
}
```

structs.go

```
type fib struct {
  // buffer is the size of the channel buffer.
  buffer int

  // ch is the internal channel where numbers are written.
  ch chan int
}

func NewFib(b int) (*fib, error) {
  f := new(fib)
  f.buffer = b
  f.ch = make(chan int, b)

  go func(f *Fib) {
    for i, j := 0, 1; ; i, j = i+j, i {
      f.ch <- i
    }
  }(f)

  return f, nil
}
```

structs.go

```
func main() {
  f, err := NewFib(10)
  if err != nil {
    panic(err)
  }

  for i := 0; i < 10; i++ {
    fmt.Println(<-f.ch)
  }
}
```

structs.go

```
func main() {
  f, err := NewFib(10)
  if err != nil {
    panic(err)
  }

  for i := 0; i < 10; i++ {
    fmt.Println(<-f.ch)
  }
}
```

```
type fib struct {
  // buffer is the size of the channel buffer.
  buffer int

  // ch is the internal channel where numbers are written.
  ch chan int
}
```

structs.go

```
type fib struct {
  // buffer is the size of the channel buffer.
  buffer int

  // ch is the internal channel where numbers are written.
  ch chan int
}

// Next gets the next number in the fib sequence.
func (f *fib) Next() int {

}
```

structs.go

```
type Fib struct {
  // buffer is the size of the channel buffer.
  buffer int

  // ch is the internal channel where numbers are written.
  ch chan int
}

// Next gets the next number in the fib sequence.
func (f *fib) Next() int {
  return <-f.ch
}
```

structs.go

```
func main() {
  f, err := NewFib(10)
  if err != nil {
    panic(err)
  }

  for i := 0; i < 10; i++ {
    fmt.Println(<-f.ch)
  }
}
```

structs.go

```
func main() {
  f, err := NewFib(10)
  if err != nil {
    panic(err)
  }

  for i := 0; i < 10; i++ {
    fmt.Println(f.Next())
  }
}
```

structs.go

```
func main() {
  f, err := NewFib(10)
  if err != nil {
    panic(err)
  }

  for {
    fmt.Println(f.Next())
    time.Sleep(100 * time.Millisecond)
  }
}
```


```
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
```

structs.go

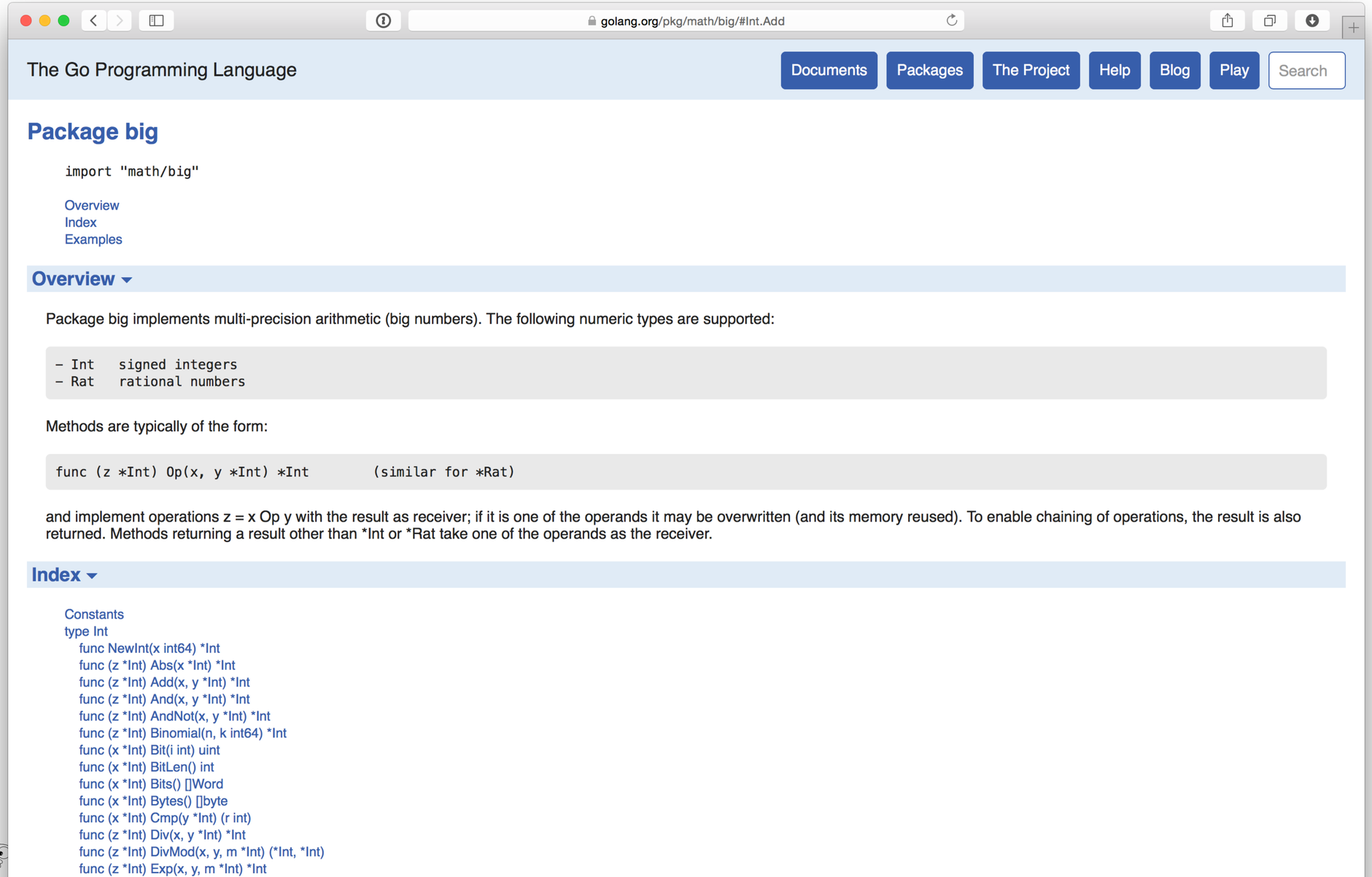
```
17799794160047141 89
28800671943708161 20
46600466103755303 09
75401138047463464 29
-62465836585876748 78
12935301461586715 51
-49530535124290033 27
-36595233662703317 76
-86125768786993351 03
61746438287398847 37
-24379330499594503 66
37367107787804343 71
12987777288209840 05
50354885076014183 76
63342662364224023 81
-70769893296857308 59
-7427230932633284 78
-78197124229490593 37
-85624355162123878 15
20645961345481044 64
-64978393816642833 51
-44332432471161788 87
75156614449290893 78
```

structs.go

What The Fuck Is This Shit?

By Dr. Seuss

```
uint64: range: 0 through 18446744073709551615.
```



Package big

```
import "math/big"
```

Overview
Index
Examples

Overview

Package big implements multi-precision arithmetic (big numbers). The following numeric types are supported:

```
- Int   signed integers
- Rat   rational numbers
```

Methods are typically of the form:

```
func (z *Int) Op(x, y *Int) *Int        (similar for *Rat)
```

and implement operations z = x Op y with the result as receiver; if it is one of the operands it may be overwritten (and its memory reused). To enable chaining of operations, the result is also returned. Methods returning a result other than *Int or *Rat take one of the operands as the receiver.

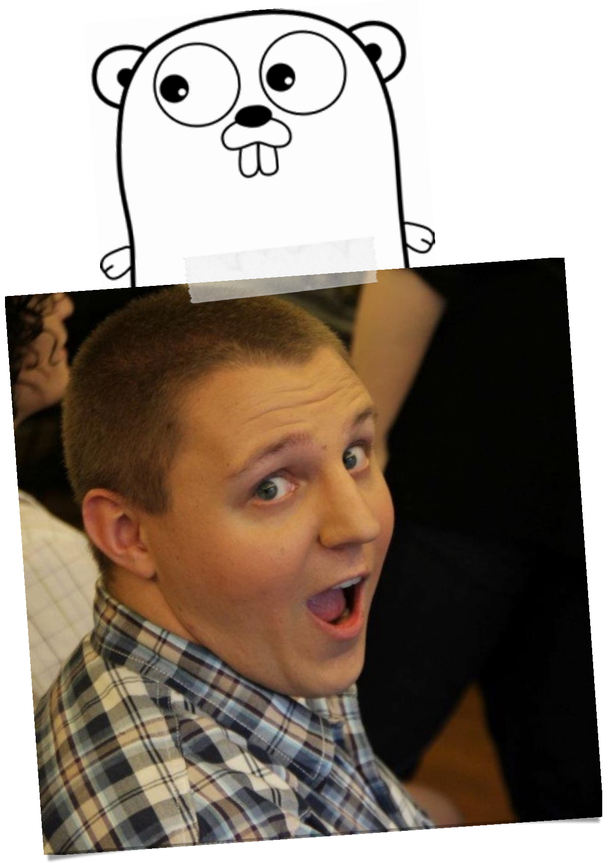
Index

Constants
type Int
  func NewInt(x int64) *Int
  func (z *Int) Abs(x *Int) *Int
  func (z *Int) Add(x, y *Int) *Int
  func (z *Int) And(x, y *Int) *Int
  func (z *Int) AndNot(x, y *Int) *Int
  func (z *Int) Binomial(n, k int64) *Int
  func (x *Int) Bit(i int) uint
  func (x *Int) BitLen() int
  func (x *Int) Bits() []Word
  func (x *Int) Bytes() []byte
  func (x *Int) Cmp(y *Int) (r int)
  func (z *Int) Div(x, y *Int) *Int
  func (z *Int) DivMod(x, y, m *Int) (*Int, *Int)
  func (z *Int) Exp(x, y, m *Int) *Int

MORE?

MORE?

tour.golang.org



SETH VARGO

@sethvargo

THANK YOU!


ScaleAbility