package main

import "fmt"

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

package main

import (
    "fmt"
    "os"
)

func main() {
    argsAll := os.Args
    argsMinusExePath := os.Args[1:]
    arg3 := os.Args[3]
    fmt.Println(argsAll)
    fmt.Println(argsMinusExePath)
    fmt.Println(arg3)
}

Golang

a humble sales pitch to the holdouts

K. Heller

latest slides: https://github.com/pestophagous/works#golang-pitch
Backstory

sales pitch to the skeptics

sales pitch to the curmudgeons

sales pitch to the battle worn, battle weary, fad-resisting graybeards

(also plenty of content for enthusiastic polyglots)

```go
package main

import "fmt"

color "fmt"

color main() {
    fmt.Println("Hello, World!")
}
```
#if !defined(NDEBUG)
define BOOST_MULTI_INDEX_ENABLE_INVARIANT_CHECKING
#define BOOST_MULTI_INDEX_ENABLE_SAFE_MODE
#endif

#include <boost/multi_index_container.hpp>
#include <boost/multi_index/member.hpp>

using boost::multi_index_container;
using namespace boost::multi_index;
typedef multi_index_container<
car_model,
indexed_by<
 ordered_unique<
 tag<model>,
   BOOST_MULTI_INDEX_MEMBER(car_model,std::string,model)
 >
 ,
 ordered_non_unique<
   key_from_key<
     BOOST_MULTI_INDEX_MEMBER(car_manufacturer,const std::string,name),
     BOOST_MULTI_INDEX_MEMBER(car_model,const car_manufacturer *,manufacturer)
 >
 ,
 ordered_non_unique<
   tag<price>,BOOST_MULTI_INDEX_MEMBER(car_model,int,price)
 >
>
tag<model>,BOOST_MULTI_INDEX_MEMBER(car_manufacturer,const std::string,name),
BOOST_MULTI_INDEX_MEMBER(car_model,const car_manufacturer *,manufacturer)
>
car_table;

typedef car_table_manufacturer_view::iterator car_table_manufacturer_view::iterator;

int excerpted_code()
{
    const car_manufacturer * cadillac=
        *(cmt.insert(car_manufacturer("Cadillac")).first);
    const car_manufacturer * ford    =
        *(cmt.insert(car_manufacturer("Ford")).first);

car_table ct;
ct.insert(car_model("XLR",cadillac,76200));
car_table_manufacturer_view::iterator ictmv0,ictmv1;
std::cout<<"listing by method 2"<<std::endl;
while(ictmv0!=ictmv1){
    std::cout<<**ictmv0;
    ++ictmv0;
}
std::cout<<std::endl;
    return 0;
"}
What Do I Care About?

latest slides: https://github.com/pestophagous/works#golang-pitch
What Do I Care About?

- Multi-Paradigm
  - Procedural
  - Object Oriented
  - Functional Programming (closures, function composition)

- Type Safety (static types, compiler checked)

- Data Hiding
  - Hide subsets of methods within a class (private data)
  - Hide sets of helper classes within a module (export control)

(Bonus: immutability)
Applying The Tools

(a talk unto itself...)

(complete book information at end of slide deck)
What Do I Care About?

- Multi-Paradigm
  - Procedural
  - Object Oriented
  - Functional Programming

- Type Safety

- Data Hiding
  - Hide subsets of methods within a class (private data)
  - Hide sets of helper classes within a module (export control)

(Bonus: immutability)
Go giveth...

- Multi-Paradigm
  - Procedural
  - Object Oriented
  - Functional Programming

- Type Safety

- Data Hiding
  - Hide subsets of methods within a class (private data)
  - Hide sets of helper classes within a module (export control)

(Bonus: immutability)

gopher by Takuya Ueda
(https://twitter.com/tenntenn)
based on art by Renee French
Go taketh away...
Go taketh away...

- tabs versus spaces
- brace-indent style debate
- protected visibility
- compiler warning levels
- overloading
- implementation inheritance
- deep spaghetti inheritance
- composition versus inheritance
- exceptions versus return code
- telescoping constructors
- test harness contortions
- circular module dependencies

latest slides: https://github.com/pestophagous/works#golang-pitch
package main

import (  
    "fmt"  
    "strings"
)

// HasContent is true if there are any  
// non-whitespace characters in the input.  
func HasContent(text string) bool {   
    text = strings.TrimSpace(text)  
    isNotBlank := text != ""  
    return isNotBlank
}

func HasAnyContent(lines []string) bool {   
    for i := 0; i < len(lines); i++ {  
        if HasContent(lines[i]) {  
            return true
        }
    }

    return false
}

func main() {  
    var someBoolean bool = true  
    var someString1 string = "text"  
    var someInteger int = 32

    fmt.Println("Hello, playground")  
    fmt.Println(someBoolean, someString1, someInteger)

    fmt.Println("result of calling HasContent: ", HasContent(" - "))

    lines := []string{"", "", ""}  
    fmt.Println("calling HasAnyContent: ", HasAnyContent(lines))

    lines = append(lines, " x ")  
    fmt.Println("how about now: ", HasAnyContent(lines))
}

starting: https://play.golang.org/p/a-z_fg-7YK
finished: https://play.golang.org/p/SIKwc2xBwg
Go giveth...

Type Safety (Compiler Type Checks)

```go
def Salutation(name string, dog bool) string {
    s := fmt.Sprintf("To: ", name)
    if dog {
        s += " and Dog"
    }
    return s
}

def main() {
    greeting := Salutation("Mary", true)
    // cannot use true (type bool) as type int in
    // argument to Salutation2
    // greeting = Salutation2("Mary", true)

    fmt.Println(greeting)
}
```

starting: https://play.golang.org/p/c5LruL7UaE
finished: https://play.golang.org/p/khUC-xYfcK
Go giveth...

Functional Programming (Closures. First-class Functions.)

```go
code
func MakeCounter() func() int {
    counterValue := 0
    return func() int {
        counterValue++
        return counterValue
    }
}

code
func main() {
    counter := MakeCounter()
    fmt.Println(counter())
    fmt.Println(counter())
    fmt.Println(counter())
    fmt.Println(counter())
}
```

https://play.golang.org/p/b2NDPPjFph

latest slides: https://github.com/pestophagous/works#golang-pitch
Go giveth...

Functional Programming (Closures. First-class Functions.)

```go
golang
func romanNumeralDict() func(int) string {
    // innerMap is captured in the closure below
    innerMap := map[int]string{
        1000: "M",
        900: "CM",
        500: "D",
        400: "CD",
        100: "C",
    }

    return func(key int) string {
        return innerMap[key]
    }
}

func main() {
    fmt.Println(romanNumeralDict()(1000))
    dict := romanNumeralDict()
    fmt.Println(dict(400))
}

// http://stackoverflow.com/a/27457144/10278
https://play.golang.org/p/B8I_mfPlue
```
Go giveth...

Object Oriented Programming

type Classroom struct {
    deskCount int
}

func (c Classroom) AddOneDesk() { // this needs refinement!
    c.deskCount++
}

func main() {
    room := &Classroom{deskCount: 2}
    fmt.Println(room)

    room.AddOneDesk() // probably doesn't do what you expect
    fmt.Println(room)
}
type Classroom struct { // Note: no declaration of implemented base interfaces.
    deskCount int
}

type Office struct {
    deskCount int
}

func (c *Classroom) AddOneDesk() {
    c.deskCount++
}

func (o *Office) AddOneDesk() {
    o.deskCount++
}

// DeskHolder interface is implemented // by Classroom and Office.
type DeskHolder interface {
    AddOneDesk() // AddDeskTo accepts any object that fulfills the DeskHolder interface.
    AddOneDesk() // AddDeskTo accepts any object that fulfills the DeskHolder interface.
}

func AddDeskTo(holder DeskHolder) {
    holder.AddOneDesk()
}

func main() {
    room := &Classroom{deskCount: 2}
    fmt.Println(room)

    room.AddOneDesk()
    fmt.Println(room)

    office := &Office{deskCount: 0}
    fmt.Println(office)

    office.AddOneDesk()
    fmt.Println(office)

    AddDeskTo(office)
    AddDeskTo(room)
}

starting: https://play.golang.org/p/6VDzSiz-JG
finished: https://play.golang.org/p/3a00EesJyA
Go giveth...

Automated Testing

// Running this test code relies on the following prereqs:
//   Create a directory that only contains two files.
//   One file is named classroom.go and contains the content
//   of https://play.golang.org/p/3a00EesJyA
//   The other is named classroom_test.go and contains this.
// Then navigate inside the directory and run:
//   go test -v -bench=.

package main

import (
    "fmt"
    "testing"
)

func TestClassroom(t *testing.T) {
    const startVal = 2
    room := &Classroom{deskCount: startVal}
    room.AddOneDesk()
    if startVal == room.deskCount {
        t.Error("AddOneDesk did not change desk count")
    }
}

func ExampleAddOneDesk() {
    room := &Classroom{deskCount: 20}
    room.AddOneDesk()
    fmt.Println(room.deskCount)
    // Output:
    // 21
}

func BenchmarkAddOneDesk(t *testing.B) {
    const startVal = 2
    room := &Classroom{deskCount: startVal}
    room.AddOneDesk()
}
Go giveth...

Automated Testing

*If you aren't looking at performance until late in the project cycle, you have lost an incredible amount of information as to when performance changed. If performance is going to be an important architectural and design criterion, then performance testing should begin as soon as possible.* ...

*... Instead of having to think about the entire architecture when you encounter performance problems, you can focus on the most recent changes.*

—Rebecca Parsons

---

97 Things Every Software Architect Should Know
Edited by Richard Monson-Haefel
ISBN-10: 059652269X
[https://books.google.com/books?id=HDknEjQJkbUC](https://books.google.com/books?id=HDknEjQJkbUC)
Go giveth...
Go taketh away...
Go taketh away...

Style Nitpicking

go fmt
Go taketh away...

Style Nitpicking

IndentationError: unindent does not match any outer indentation level
Go taketh away...

Style Nitpicking

IndentationError: unindent does not match any outer indentation level

IndentationError: expected an indented block
Go taketh away...

Style Nitpicking

IndentationError: unindent does not match any outer indentation level

IndentationError: expected an indented block

IndentationError: unexpected indent
Go taketh away...

Style Nitpicking

```go
package main

import {
    "fmt"
    "math"
    "log"
    "errors"
    "io"
}

func main() {
    flag := true
    if(flag){ // parentheses will be removed
        fmt.Println("true"); // semicolon
    } // indentation will be repaired

    fmt.Println("another thing") // indentation

    x := [...]int{1, 2, 3} // the 2 blank lines above here
                            // will be reduced to 1 by go fmt

    fmt.Println(x)
}

type Address struct {
    heading string
    street string // members of the struct will
                   // be column-aligned by go fmt
    apt string
    code int
    isUSA bool
}

starting: https://play.golang.org/p/CS8LIfLHPM
finished: https://play.golang.org/p/faSgHYhhdg
```
Go taketh away...

Style Nitpicking

```
type Point struct {
    x int
    y int
}
var points = [2]Point{
    Point{x: 2, y: 3},
    Point{x: 3, y: 4},
}
x := []int{1, 2, 3}
for _, _ = range x {
    fmt.Println("hello")
}
x := []int{1, 2, 3}
y := x[1:len(x)]
```

```
type Point struct {
    x int
    y int
}
var points = [2]Point{
    {x: 2, y: 3},
    {x: 3, y: 4},
}
x := []int{1, 2, 3}
for range x {
    fmt.Println("hello")
}
x := []int{1, 2, 3}
y := x[1:]
```
func compute() bool {
    result := true
    if 2 > 1 {
        result := false
    }
    return result
}

func main() {
    x := 0
    fmt.Println(compute())
    fmt.Println("done!")
}
Go taketh away...

Compiler Warnings

```go
cpackage main

import (  
    "fmt"  
    "math" // error: imported and not used: "math"
)

cconst b byte = 256 // error: constant 256 overflows byte

cfunc decider(i int, j int) bool {  
    if i < j {  
    }  
} // error: missing return at end of function

cfunc main() {

    numbers := [3]int{1, 2, 3}

    var idx bool = true
    x := numbers[idx] // error: non-integer slice index idx

    fmt.Println("Hello")
}
```

https://play.golang.org/p/w1KGwifTo6  latest slides: https://github.com/pestophagous/works#golang-pitch
Go taketh away...

Exceptions & Throw/Catch
"Errors are...an important part of a package's API or an application's user interface, and failure is just one of several expected behaviors. This is the approach Go takes to error handling."
"Errors are...an important part of a package's API or an application's user interface, and failure is just one of several expected behaviors. This is the approach Go takes to error handling."
package main

import (  
    "fmt"  
    "net/mail"  
    "os/user"  
    "time"
)

func ConvertToStringWeWant(group *user.Group) string {  
    return "TODO"
}

func GetGroupInformation(groupName string) (string, error) {  
    var grp *user.Group  
    var err error  
    if grp, err = user.LookupGroup(groupName); err != nil {  
        return "", err
    }  
    s := ConvertToStringWeWant(grp)  
    // Do other arbitrary logic here...  
    return s, nil
}

func main() {  
    var loc *time.Location  
    var addr *mail.Address  
    var err error  
    loc, err = time.LoadLocation("America/N_Yorkia")  
    fmt.Println(err)

    addr, err = mail.ParseAddress("xyz@lm@jk@rs")  
    fmt.Println(err)

    s, err := GetGroupInformation("defghijklmnop")  
    fmt.Println(err)

    if err == nil {  
        fmt.Println(loc, addr, s)
    }
}

starting: https://play.golang.org/p/bdAEISB1Nj
finished: https://play.golang.org/p/U8zKlrarek
Go taketh away...

Test Harness Contortions

```go
import {
    "errors"
    "fmt"
    "os"
}

type FakeTesterFileInfo struct {
    os.FileInfo
}

func (f FakeTesterFileInfo) Size() int64 {
    return 70000
}

func ProcessFile(fileInfo os.FileInfo) error {
    if fileInfo.Size() > 65535 {
        return errors.New("file too big")
    }

    // do some kind of processing here
    return nil
}

func main() {
    err := ProcessFile(FakeTesterFileInfo{})
    fmt.Println(err)
    fmt.Println("done")
}

/*
https://golang.org/pkg/os/#FileInfo

type FileInfo interface {
    Name() string       // base name of the file
    Size() int64        // length in bytes for regular files
    Mode() FileMode     // file mode bits
    ModTime() time.Time // modification time
    IsDir() bool        // abbreviation for Mode().IsDir()
    Sys() interface{}   // underlying data source (or nil)
}
*/
```

starting: https://play.golang.org/p/xhpImxN5HV
finished: https://play.golang.org/p/m-2Ne2WE-t
Parting Words

The cleanest code is the code not written.

Go gopher by Renee French

latest slides: https://github.com/pestophagous/works#golang-pitch
The cleanest code is the code not written.
For everything else, there's Go.
Your next dose of Go:

**The Go Programming Language**
by Alan A. A. Donovan, Brian W. Kernighan
ISBN-10: 0134190440
https://books.google.com/books?id=SJHvCgAAQBAJ

Comprehensive. Authoritative. A joy to read.

**Go in Action**
by William Kennedy, Brian Ketelsen, Erik St. Martin
ISBN-10: 1617291781
https://books.google.com/books?id=HD MmrgEACAAJ

Excellent quick-start on goroutines. Clear, deep treatment of slices.

Surprisingly gritty, creepy art from the Go artist:
https://twitter.com/reneefrench

- Go for C++ devs: https://talks.golang.org/2015/go4cpp.slide
- Go for Javaneros: https://talks.golang.org/2014/go4java.slide
- Go for Pythonistas: https://talks.golang.org/2013/go4python.slide

Interview with Go co-creator Robert Griesemer:
https://www.youtube.com/watch?v=on5DeUyWDqI
- probing questions on exceptions & generics
- interesting language design & comparative language topics
<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
<th>ISBN-10</th>
<th>Google Books Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applying UML and Patterns</td>
<td>Craig Larman</td>
<td>0131489062</td>
<td><a href="https://books.google.com/books?id=tuxQAAGAAAMAAJ">https://books.google.com/books?id=tuxQAAGAAAMAAJ</a></td>
</tr>
<tr>
<td>Refactoring</td>
<td>Martin Fowler</td>
<td>0201485672</td>
<td><a href="https://books.google.com/books?id=UTgFCAAQBAJ">https://books.google.com/books?id=UTgFCAAQBAJ</a></td>
</tr>
<tr>
<td>Effective C++</td>
<td>Scott Meyers</td>
<td>0321334876</td>
<td><a href="https://books.google.com/books?id=eQq9AQAAQBAJ">https://books.google.com/books?id=eQq9AQAAQBAJ</a></td>
</tr>
<tr>
<td>Head First Design Patterns</td>
<td>Eric Freeman &amp; Elisabeth Robson</td>
<td>0596007124</td>
<td><a href="https://books.google.com/books?id=NbCNAQAAQBAJ">https://books.google.com/books?id=NbCNAQAAQBAJ</a></td>
</tr>
<tr>
<td>Working Effectively with Legacy Code</td>
<td>Michael Feathers</td>
<td>0131177052</td>
<td><a href="https://books.google.com/books?id=vlo_nWopSYC">https://books.google.com/books?id=vlo_nWopSYC</a></td>
</tr>
<tr>
<td>Structure and Interpretation of Computer Programs</td>
<td>Harold Abelson, Gerald Jay Sussman, Julie Sussman</td>
<td>0262510871</td>
<td><a href="https://books.google.com/books?id=6QOXQgAACAAJ">https://books.google.com/books?id=6QOXQgAACAAJ</a></td>
</tr>
<tr>
<td>Refactoring in Large Software Projects</td>
<td>Martin Lippert, Stephen Roock</td>
<td>0470858923</td>
<td><a href="https://books.google.com/books?id=erBQAAMAAJ">https://books.google.com/books?id=erBQAAMAAJ</a></td>
</tr>
<tr>
<td>Large-Scale C++ Software Design</td>
<td>John Lakos</td>
<td>0201633620</td>
<td><a href="https://books.google.com/books?id=AuMpqAACAAAMAAJ">https://books.google.com/books?id=AuMpqAACAAAMAAJ</a></td>
</tr>
</tbody>
</table>

latest slides: [https://github.com/pestophagous/works#golang-pitch](https://github.com/pestophagous/works#golang-pitch)