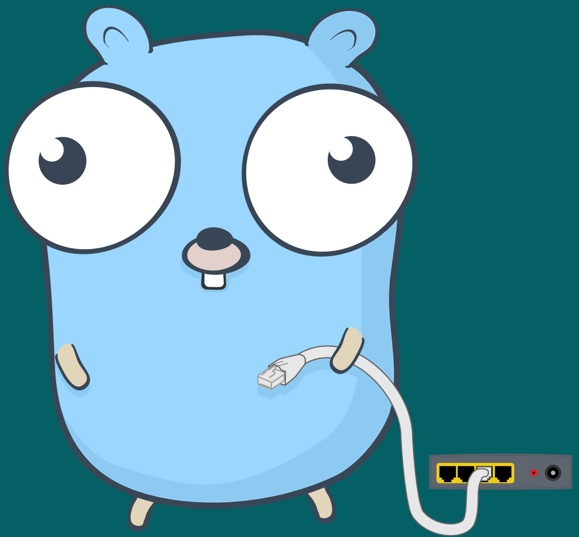
PARK PARK

GO GO!



Topics covered

- Why Go?
- Hello World
- Web services
- Web services with JSON
- SQL
- Directory structure
- Useful links

Topics *not* covered

- Language features
- Testing

Why Go?

My favourite things

1. Opinionated as #"!
2. Typed
3. Best standard-library
4. Interface
5. Great community

Other things

- Concurrency
- Tooling
- OOP, but not Java OOP

Before we start

Godoc

- https://godoc.org

```
$ go doc
```

hello.go

```go
package main

import "fmt"

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

webservice.go

```go
package main

import (
	"fmt"
	"net/http"
)

func main() {
	http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
		fmt.Fprintf(w, "Welcome to PARKPARK!")
	})

	http.ListenAndServe(":3022", nil)
}
```

jsonservice.go

```go
package main

import (
	"encoding/json"
	"fmt"
	"net/http"
)

type greetInput struct {
	Name string `json:"name"`
}

type greetOutput struct {
	Text string `json:"text"`
}

func HandleGreet(w http.ResponseWriter, r *http.Request) {
	decoder := json.NewDecoder(r.Body)
	var input greetInput
	var output greetOutput
	err := decoder.Decode(&input)
	if err != nil {
		http.Error(w, "Invalid input", http.StatusBadRequest)
```

jsonservice.go

```
$ go run code/jsonservice/jsonservice.go
```

```
$ curl http://localhost:3022
Welcome to PARKPARK!
$ curl http://localhost:3022/greet
Invalid input
$ curl -X POST -d '{"name" : "PARKPARK"}' http://localhost:3022/greet
{"text":"Hello PARKPARK"}
$ curl -X POST -d '{"name" : "PARKPARK"}' http://localhost:3022/greet
{"text":"Hello PARKPARK"}
```

sql.go

```go
// https://github.com/golang/go/wiki/SQLInterface

package main

import (
	"database/sql"
	"fmt"
	"log"

	_ "github.com/mattn/go-sqlite3"
)

func main() {
	db, err := sql.Open("sqlite3", "./sql.db")
	if err != nil {
		log.Fatal(err) // bad practice, dont do this
```

SQL interface from stdlib

From development to production

- Just a single binary (and assets)
- You don't need a web-server
- You don't need rewrite rules

Development

- Start locally.
- Use ENV vars to configure
- You probably don't need Docker
- go run cmd/myawesomeproject.go
- go build ./...
- go install
- go test ./...

Production

- A very slim Docker container
- Deployed to our Kubernetes cluster
- Using Gitlab-CI to automatically build

Standard Code Layout

- `/git/awesomesauce/`
- `/git/awesomesauce/cmd`
- `/git/awesomesauce/pkg`
- `/git/awesomesauce/internal`

https://github.com/golang-standards/project-layout

Starting a new project

Pre Go 1.11

`$GOPATH` and what not

Now

```
cd ~/git/awesomesauce
go mod init awesomesauce
go get gitlab.com/fancypants/fancypantslib
```

Your dependencies are now stored in `go.mod` and version locked by `go.sum`

You don't need to store your code in `~/go/src/` anymore

Resources

- Go Doc (again): https://godoc.org
- Go Web: https://gowebexamples.com/
- Go By Example: https://gobyexample.com/
- JustForFunc YT Channel: https://www.youtube.com/channel/UC_BzFbxG2za3bp5NRRRXJSw
- Awesome Go: https://awesome-go.com/
- Level Up? Gophercises! https://gophercises.com/