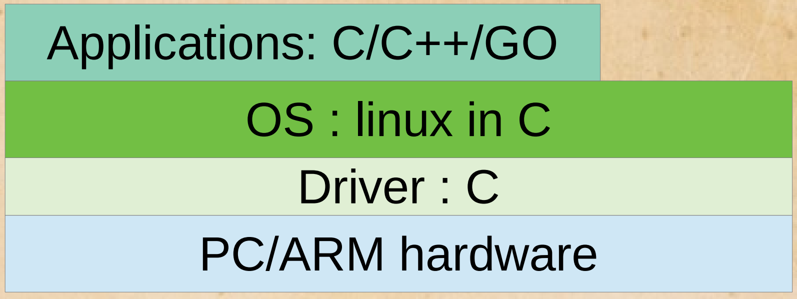
Web3 full stack design tutorial

GoLang module

伍成和 ChengHe Wu
www.deCensorMedia.org
wuchenghe@vk.com
https://github.com/brianwchh

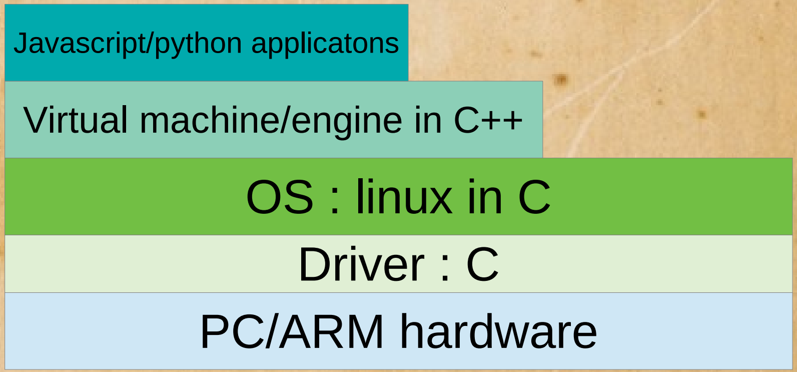
Brief intro to Go language

- Go vs C/C++



- Go invented based on C,bringing in ideas from other languages,for instance,package management,garbage collect for memory safety
- Go targets for web applications
- Go for blockchain tech.
- Go is better for deployment,has package management,easier to recreate environment in another device
- Go has garbage collect while c/c++ doesn’t

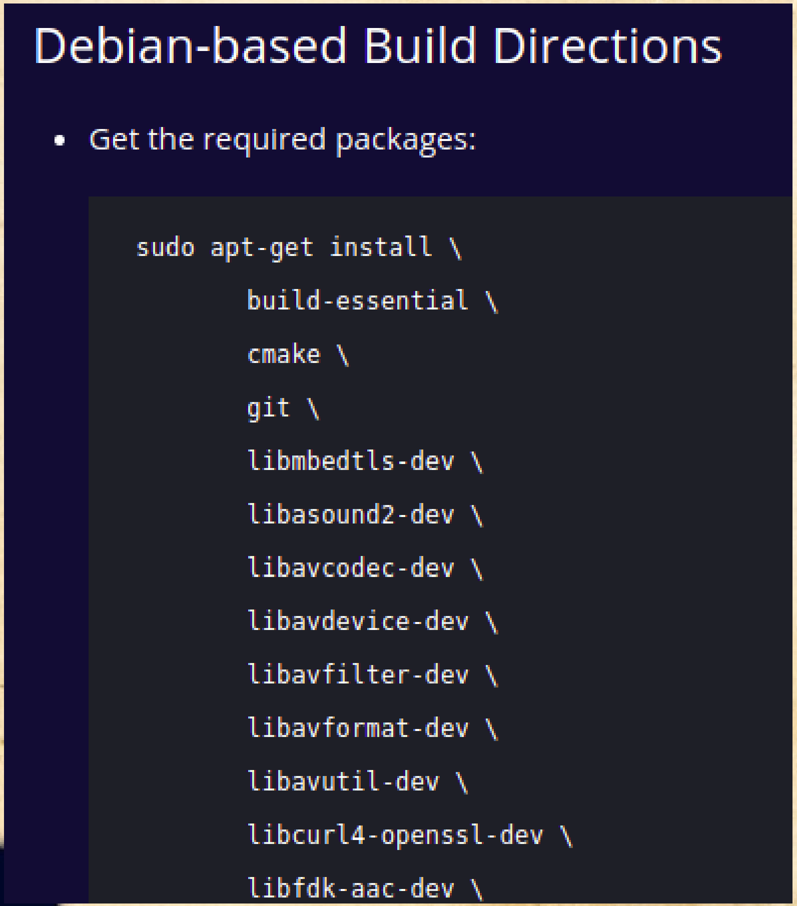
- Go vs javascript,python



- Go run natively on OS(fast) ,static language
- js/python on virtual machine(slow)
- All have package management,easy to recreate environment on another device
- Go is better for algorithm applications

Motivation of module in Golang

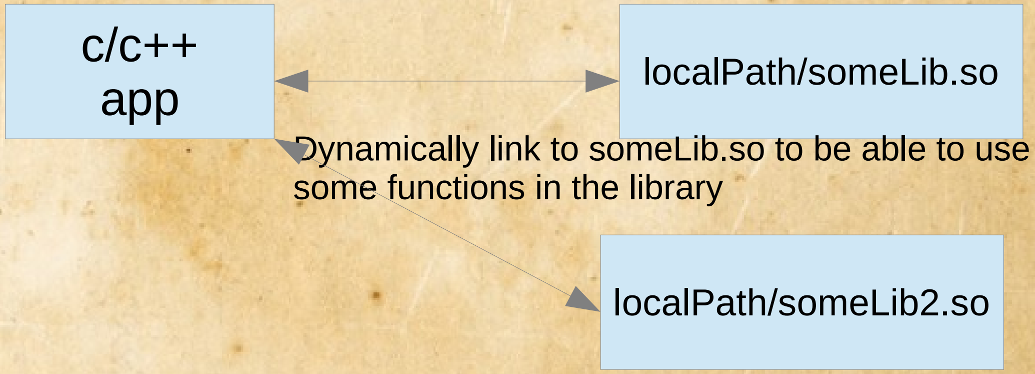
- Install libraries,writing makefile / cmake and recreate environment are headaches for c/c++,one has to manually download the dependent libraries from the official website and install,and then fill the local path in the makefile or cmake file,so that the project can build and run!



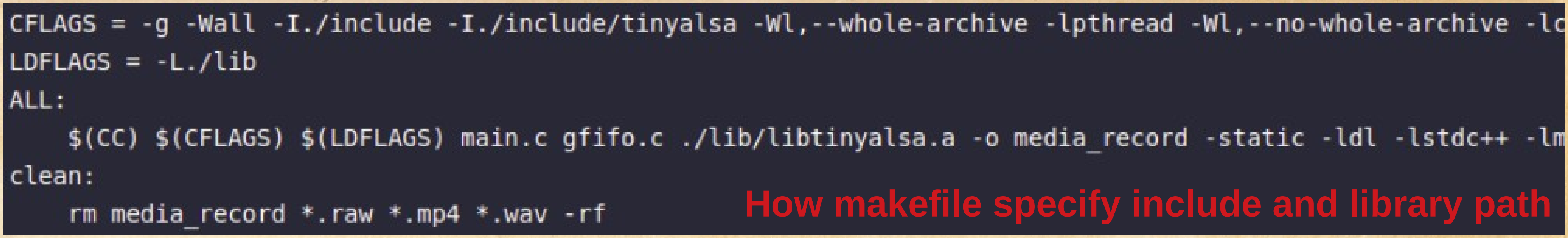
Debian-based Build Directions

- Get the required packages:

```
sudo apt-get install \
        build-essential \
        cmake \
        git \
        libmbedtls-dev \
        libasound2-dev \
        libavcodec-dev \
        libavdevice-dev \
        libavfilter-dev \
        libavformat-dev \
        libavutil-dev \
        libcurl4-openssl-dev \
        libfdk-aac-dev \
```

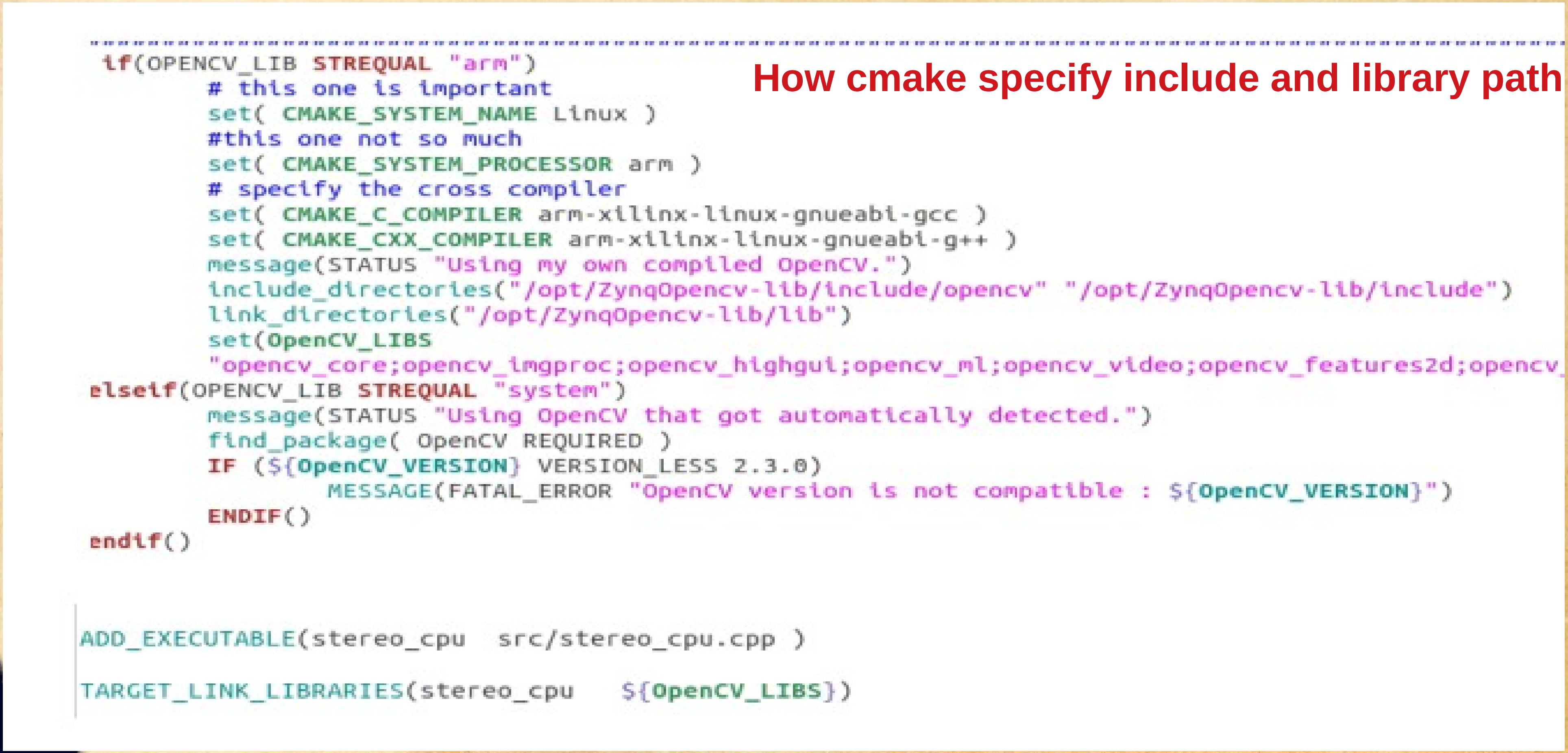

c/c++ app
localPath/someLib.so
Dynamically link to someLib.so to be able to use some functions in the library
localPath/someLib2.so

Motivation of module in Golang cont2



```
CFLAGS = -g -Wall -I./include -I./include/tinyalsa -Wl,--whole-archive -lpthread -Wl,--no-whole-archive -lc
LDFLAGS = -L./lib
ALL:
    $(CC) $(CFLAGS) $(LDFLAGS) main.c gfifo.c ./lib/libtinyalsa.a -o media_record -static -ldl -lstdc++ -lm
clean:
    rm media_record *.raw *.mp4 *.wav -rf
```

How makefile specify include and library path

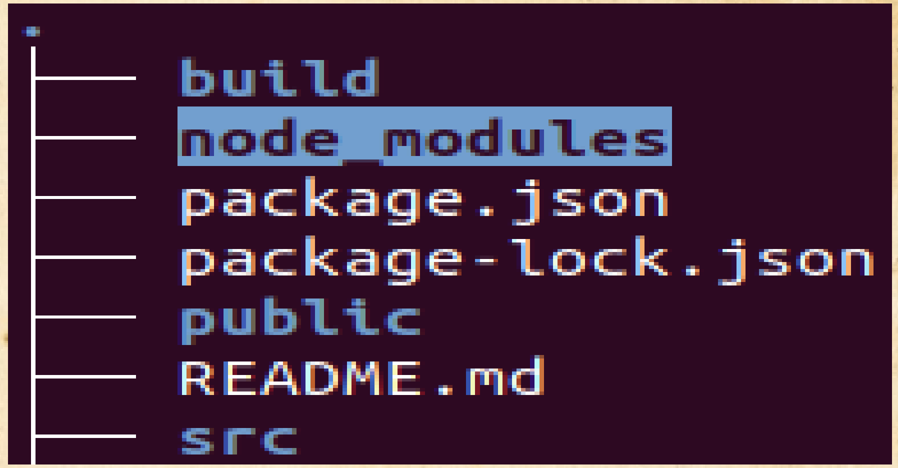


How cmake specify include and library path

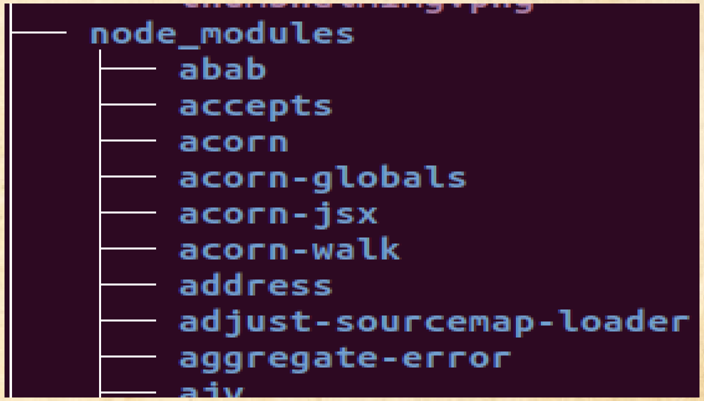
```
 if(OPENCV_LIB STREQUAL "arm")
        # this one is important
        set( CMAKE_SYSTEM_NAME Linux )
        #this one not so much
        set( CMAKE_SYSTEM_PROCESSOR arm )
        # specify the cross compiler
        set( CMAKE_C_COMPILER arm-xilinx-linux-gnueabi-gcc )
        set( CMAKE_CXX_COMPILER arm-xilinx-linux-gnueabi-g++ )
        message(STATUS "Using my own compiled OpenCV.")
        include_directories("/opt/ZynqOpencv-lib/include/opencv" "/opt/ZynqOpencv-lib/include")
        link_directories("/opt/ZynqOpencv-lib/lib")
        set(OpenCV_LIBS
        "opencv_core;opencv_imgproc;opencv_highgui;opencv_ml;opencv_video;opencv_features2d;opencv_
elseif(OPENCV_LIB STREQUAL "system")
        message(STATUS "Using OpenCV that got automatically detected.")
        find_package( OpenCV REQUIRED )
        IF (${OpenCV_VERSION} VERSION_LESS 2.3.0)
                MESSAGE(FATAL_ERROR "OpenCV version is not compatible : ${OpenCV_VERSION}")
        ENDIF()
endif()


ADD_EXECUTABLE(stereo_cpu  src/stereo_cpu.cpp )

TARGET_LINK_LIBRARIES(stereo_cpu   ${OpenCV_LIBS})
```

Motivation of module in Golang cont3

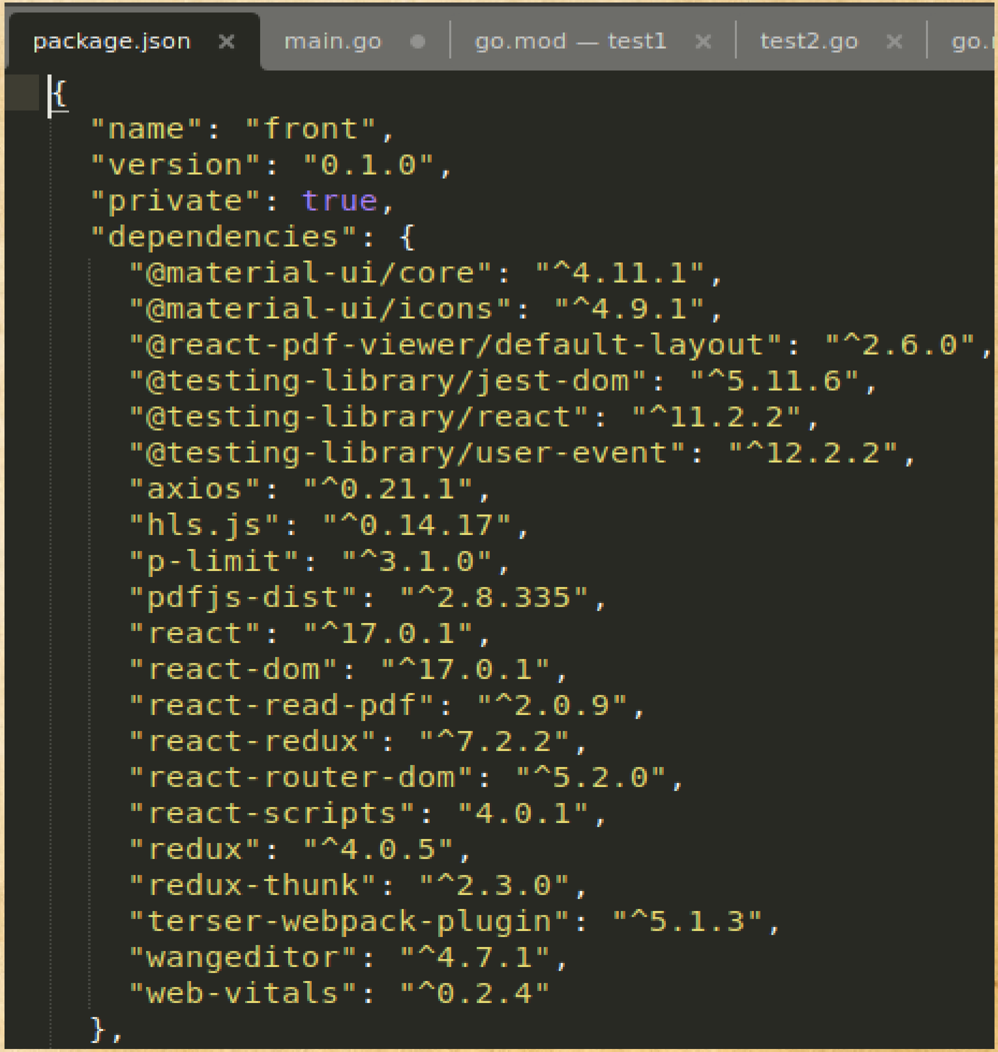
- How react javascript manage dependence?



```
.
├── build
├── node_modules
├── package.json
├── package-lock.json
├── public
├── README.md
├── src
```



```
├── node_modules
│   ├── abab
│   ├── accepts
│   ├── acorn
│   ├── acorn-globals
│   ├── acorn-jsx
│   ├── acorn-walk
│   ├── address
│   ├── adjust-sourcemap-loader
│   ├── aggregate-error
```



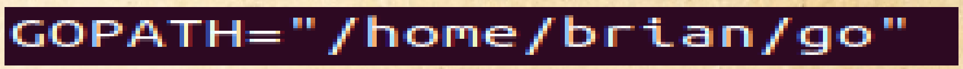
```
{
  "name": "front",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@material-ui/core": "^4.11.1",
    "@material-ui/icons": "^4.9.1",
    "@react-pdf-viewer/default-layout": "^2.6.0",
    "@testing-library/jest-dom": "^5.11.6",
    "@testing-library/react": "^11.2.2",
    "@testing-library/user-event": "^12.2.2",
    "axios": "^0.21.1",
    "hls.js": "^0.14.17",
    "p-limit": "^3.1.0",
    "pdfjs-dist": "^2.8.335",
    "react": "^17.0.1",
    "react-dom": "^17.0.1",
    "react-read-pdf": "^2.0.9",
    "react-redux": "^7.2.2",
    "react-router-dom": "^5.2.0",
    "react-scripts": "4.0.1",
    "redux": "^4.0.5",
    "redux-thunk": "^2.3.0",
    "terser-webpack-plugin": "^5.1.3",
    "wangeditor": "^4.7.1",
    "web-vitals": "^0.2.4"
  },
```

- npm install

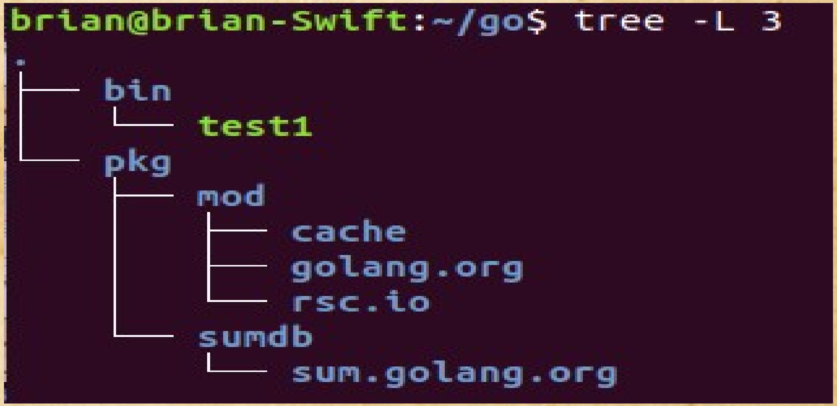
Inspiration from javascript and differences

- Bring idea from javascript and use module in Go

GOPATH is the directories for storing module packages. Defualt : *home*/username/go



```
GOPATH="/home/brian/go"
```

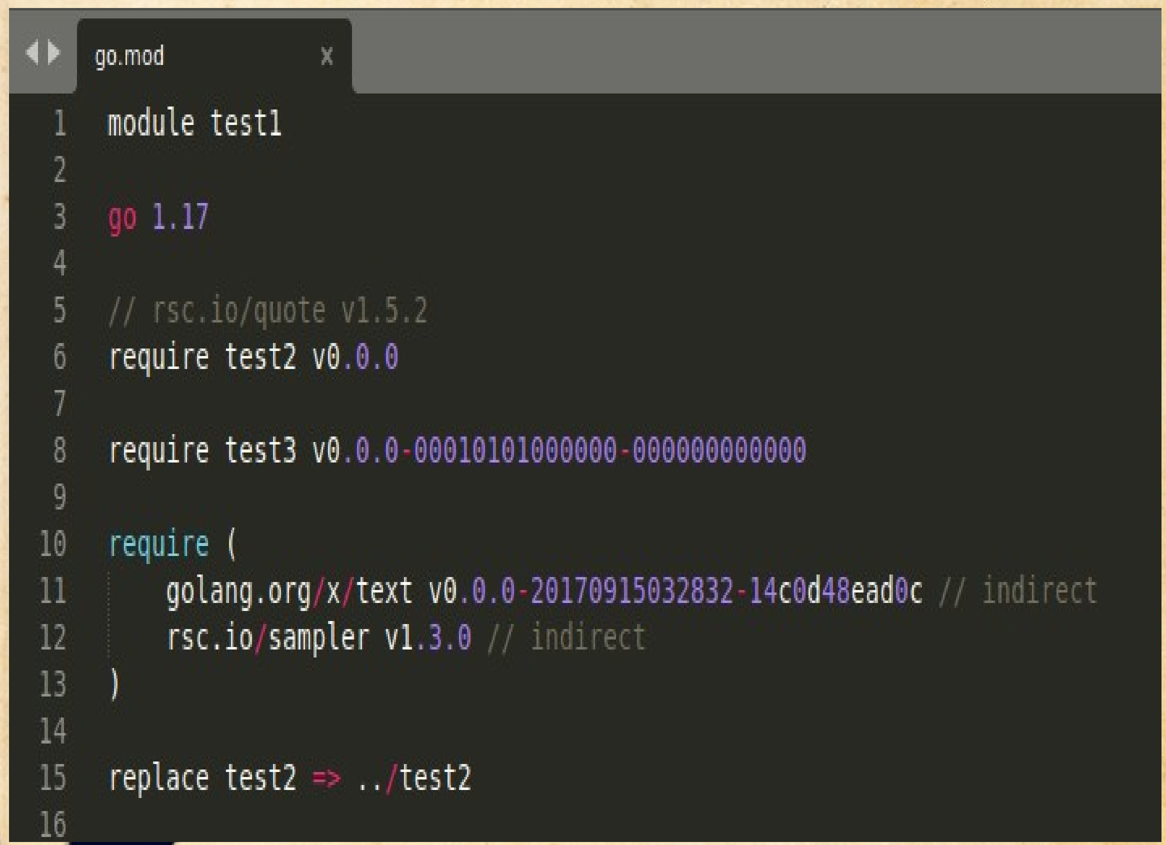


```
brian@brian-Swift:~/go$ tree -L 3
.
├── bin
│   └── test1
└── pkg
    ├── mod
    │   ├── cache
    │   ├── golang.org
    │   └── rsc.io
    └── sumdb
        └── sum.golang.org
```

- Different from javascript is that goLang store all projects dependence in $GOPATH location,while javascript store in individual project’s node_modules directory. Advantage is that this avoid downloading duplicated packages from the internet,especially some packages are very large, for instance,opencv library. For real time application like image processing,in most project we don’t use javascript,because javascript is not designed to handle computation intensive application.

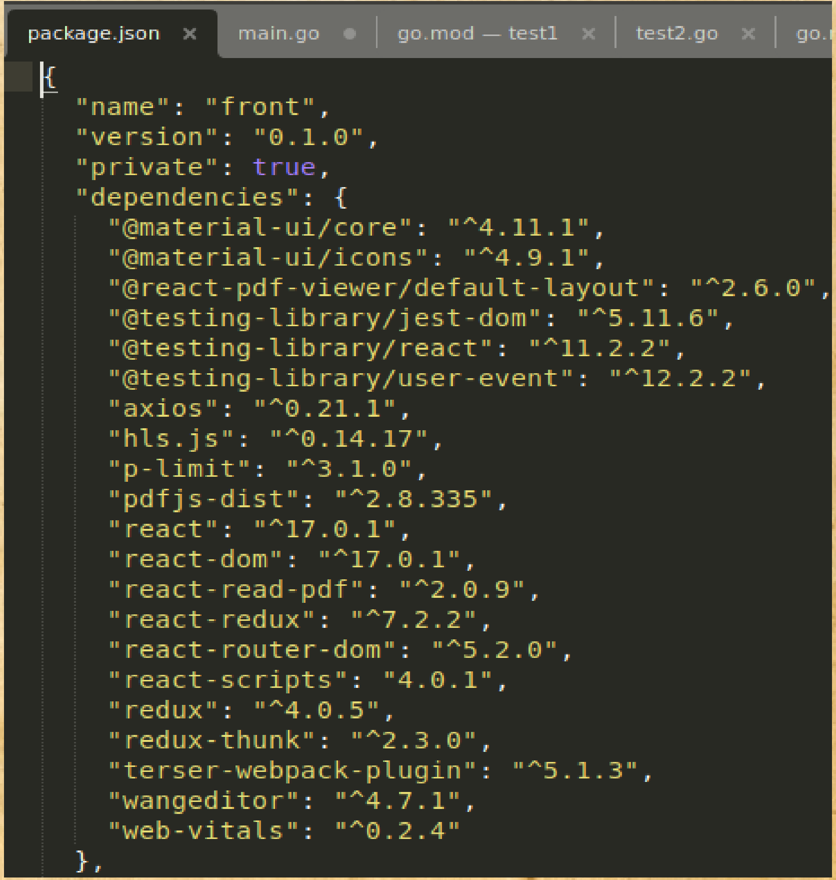
On go.mod file

- Install Go: https://go.dev/doc/install
- Create module (project): go mod init module_name



go.mod

```
module test1

go 1.17

// rsc.io/quote v1.5.2
require test2 v0.0.0

require test3 v0.0.0-00010101000000-000000000000

require (
	golang.org/x/text v0.0.0-20170915032832-14c0d48ead0c // indirect
	rsc.io/sampler v1.3.0 // indirect
)

replace test2 => ../test2

```



package.json · main.go · go.mod — test1 · test2.go · go.

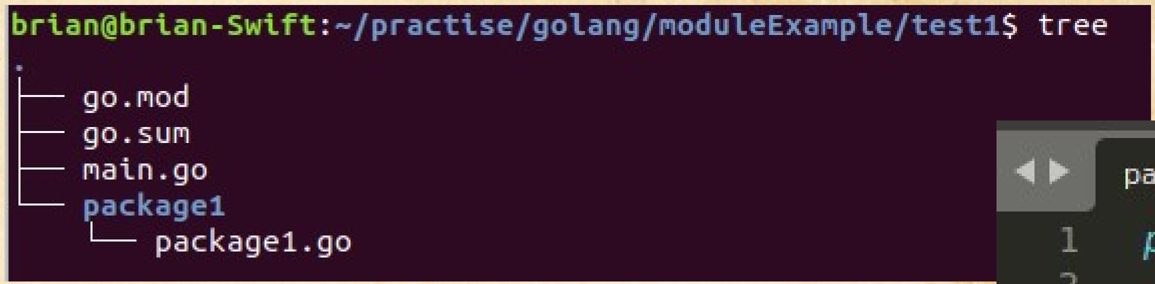
```
{
  "name": "front",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@material-ui/core": "^4.11.1",
    "@material-ui/icons": "^4.9.1",
    "@react-pdf-viewer/default-layout": "^2.6.0",
    "@testing-library/jest-dom": "^5.11.6",
    "@testing-library/react": "^11.2.2",
    "@testing-library/user-event": "^12.2.2",
    "axios": "^0.21.1",
    "hls.js": "^0.14.17",
    "p-limit": "^3.1.0",
    "pdfjs-dist": "^2.8.335",
    "react": "^17.0.1",
    "react-dom": "^17.0.1",
    "react-read-pdf": "^2.0.9",
    "react-redux": "^7.2.2",
    "react-router-dom": "^5.2.0",
    "react-scripts": "4.0.1",
    "redux": "^4.0.5",
    "redux-thunk": "^2.3.0",
    "terser-webpack-plugin": "^5.1.3",
    "wangeditor": "^4.7.1",
    "web-vitals": "^0.2.4"
  },
```

go.mod is used for

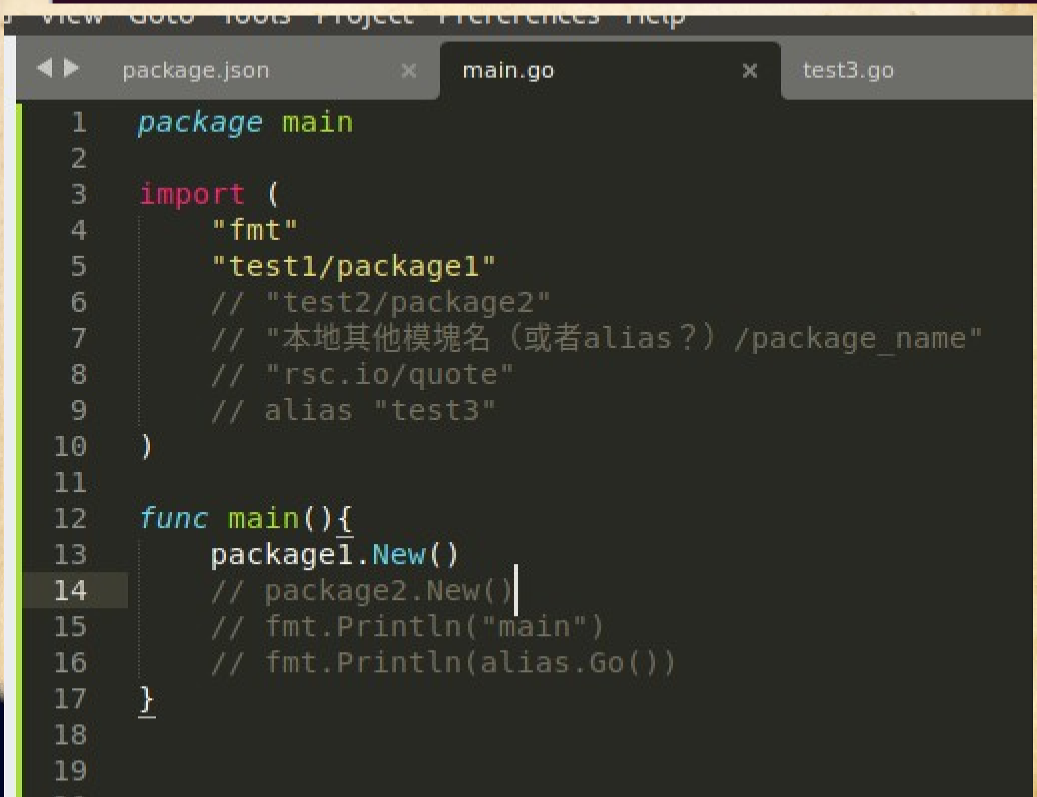
- Similar to package.json in js managing package dependencies,but does more than that, it replace tedious makefile/cmake in c/c++ project,and **automatically** download dependent packages,pointing to the include header files and linking to library. The key word **automatically** is a great improvement compared to c/c++. So Golang is invented for **engineering** and **deployment** orientation
- Accessing package within module,relative path import.
- Accessing package between local modules
- Accessing packages from internet
- Redirect url to local path
- Add alias to unstable online package
- Will explain in detail in following slices

Accesing package within module

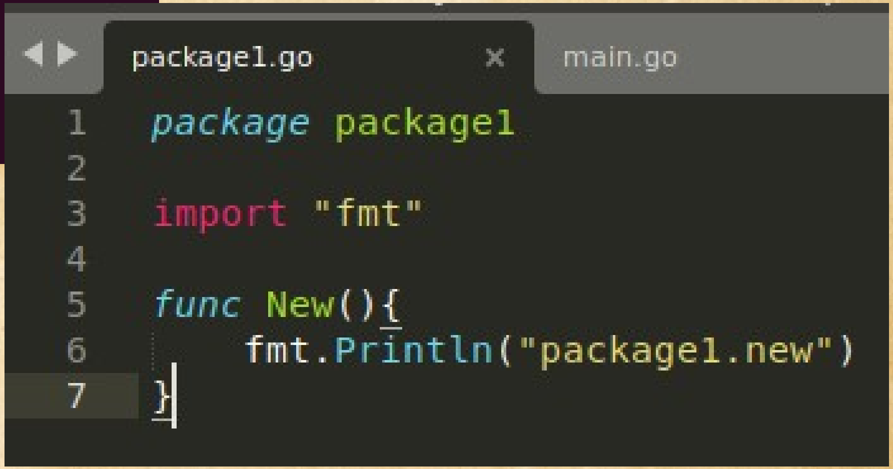
- Application scenario:accessing package1 from main.go in module test1



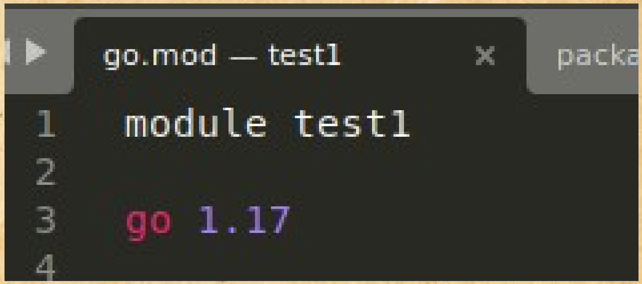
```
brian@brian-Swift:~/practise/golang/moduleExample/test1$ tree
.
├── go.mod
├── go.sum
├── main.go
└── package1
    └── package1.go
```



```go
package main

import (
	"fmt"
	"test1/package1"
	// "test2/package2"
	// "本地其他模塊名（或者alias？）/package_name"
	// "rsc.io/quote"
	// alias "test3"
)

func main(){
	package1.New()
	// package2.New()
	// fmt.Println("main")
	// fmt.Println(alias.Go())
}
```



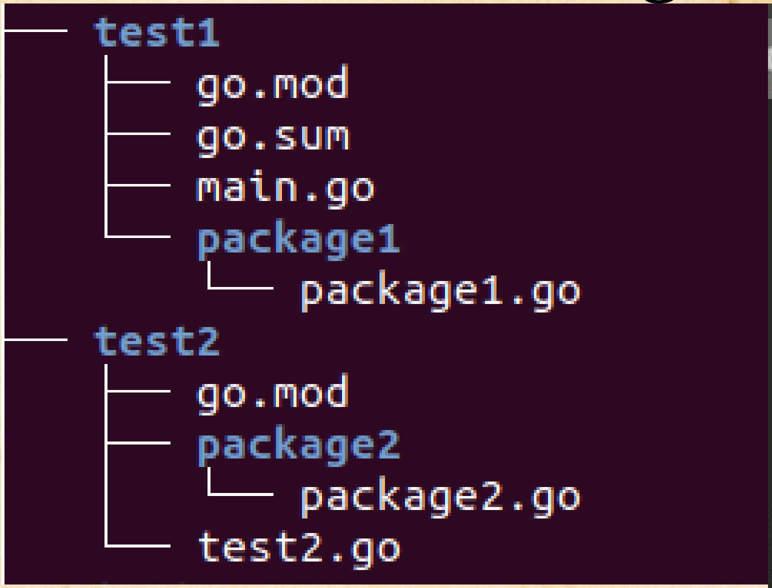
```go
package package1

import "fmt"

func New(){
	fmt.Println("package1.new")
}
```



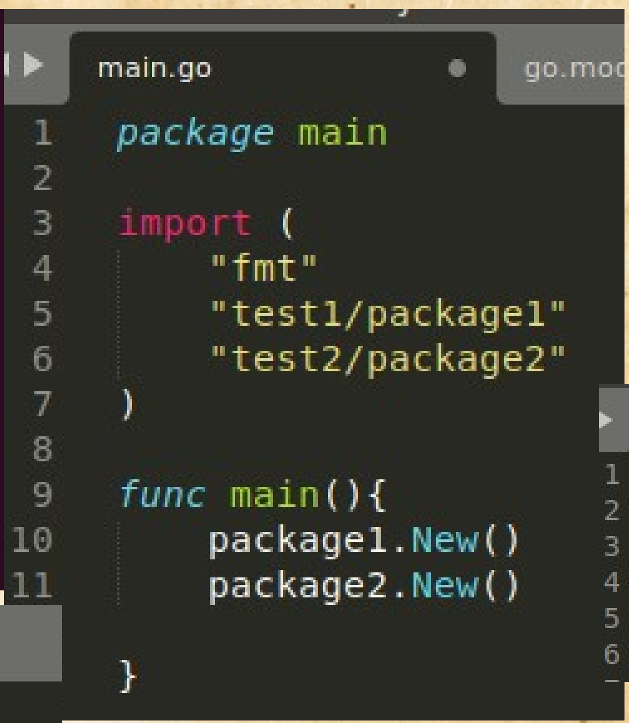
```
module test1

go 1.17
```

Accessing package between local modules

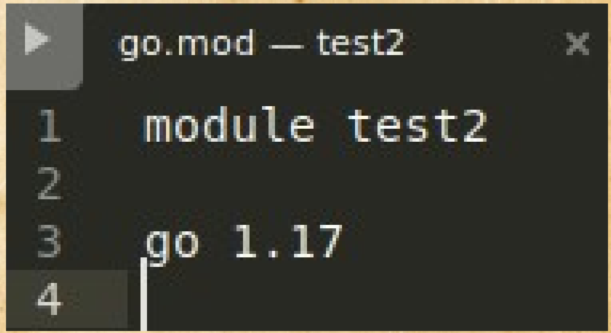
- Application scenario:accessing package2 of module test2 from main.go in module test1



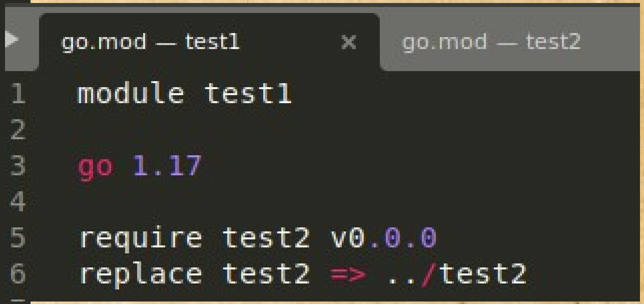
```
── test1
   ├── go.mod
   ├── go.sum
   ├── main.go
   └── package1
       └── package1.go
── test2
   ├── go.mod
   ├── package2
   │   └── package2.go
   └── test2.go
```



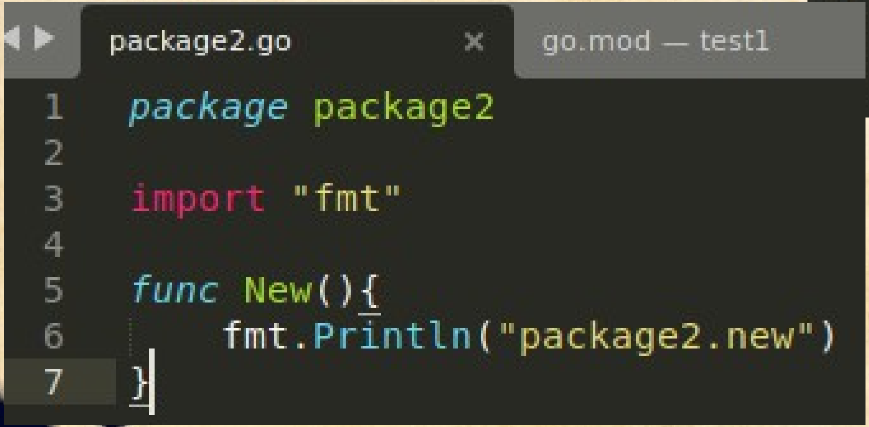
```go
package main

import (
	"fmt"
	"test1/package1"
	"test2/package2"
)

func main(){
	package1.New()
	package2.New()

}
```



```
module test2

go 1.17

```



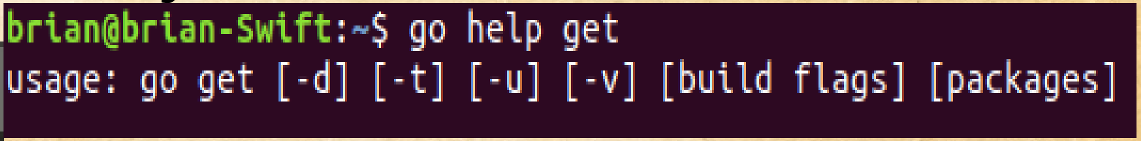
```
module test1

go 1.17

require test2 v0.0.0
replace test2 => ../test2
```



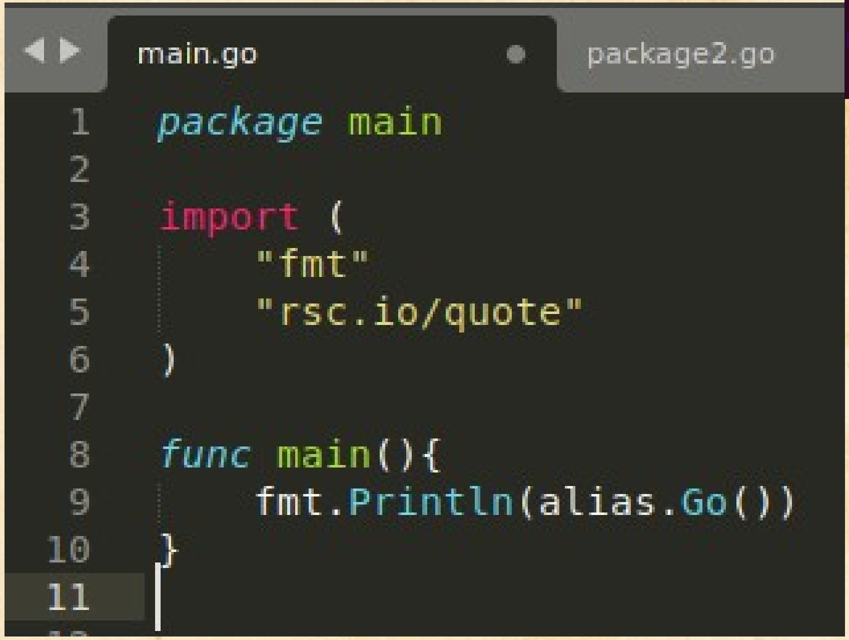
```go
package package2

import "fmt"

func New(){
	fmt.Println("package2.new")
}
```

Accessing packages from internet

- Similar to : npm install library_name

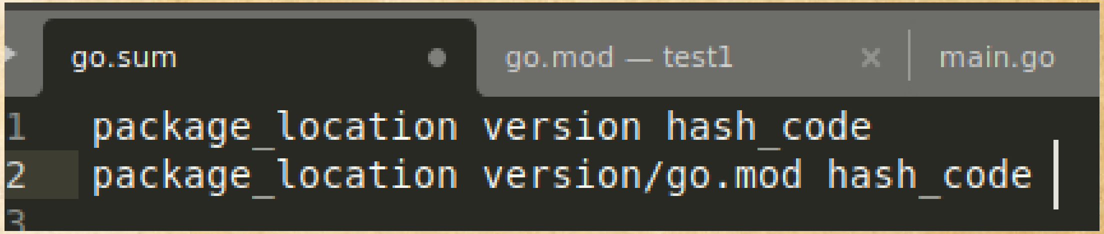


```
brian@brian-Swift:~$ go help get
usage: go get [-d] [-t] [-u] [-v] [build flags] [packages]
```

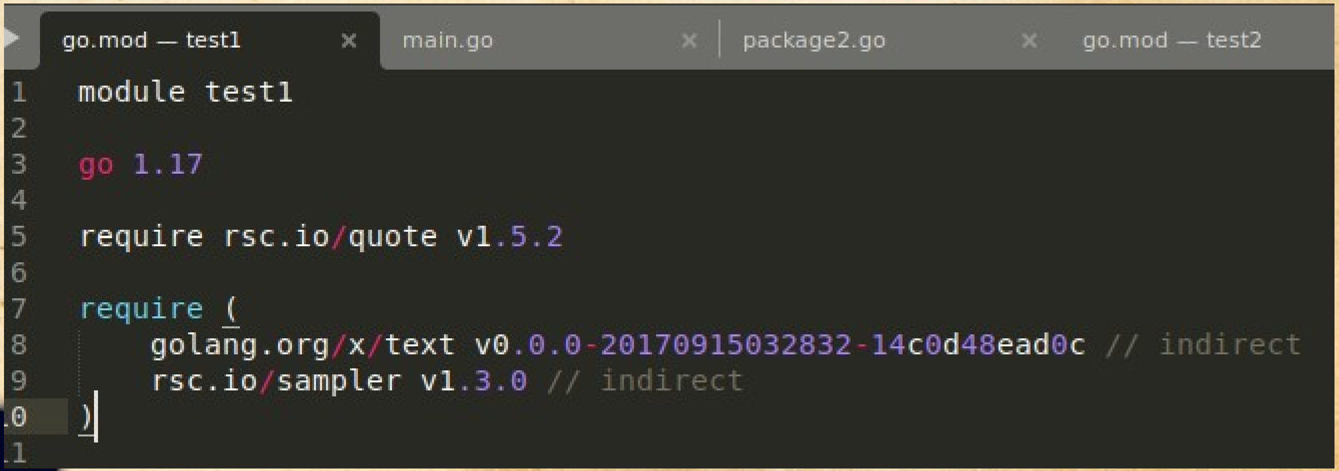


```go
package main

import (
	"fmt"
	"rsc.io/quote"
)

func main(){
	fmt.Println(alias.Go())
}
```

When a library is installed,it will automatically add information to go.mod and go.sum



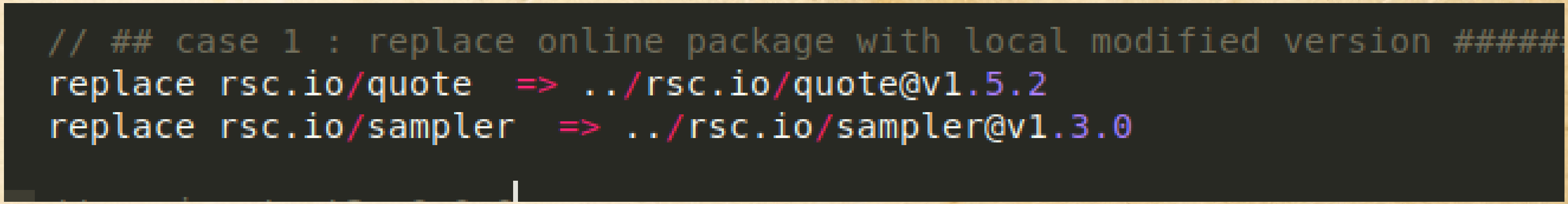
```
package_location version hash_code
package_location version/go.mod hash_code
```



```
module test1

go 1.17

require rsc.io/quote v1.5.2

require (
	golang.org/x/text v0.0.0-20170915032832-14c0d48ead0c // indirect
	rsc.io/sampler v1.3.0 // indirect
)
```

Redirect url to local path

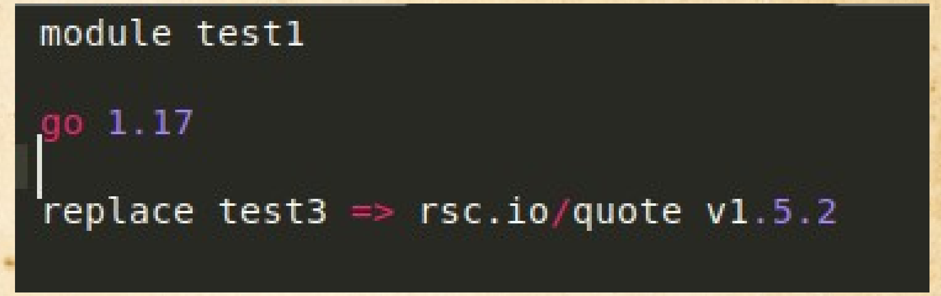
- Application scenario: we have customized a online module library,and we want to use local one instead of the online version.



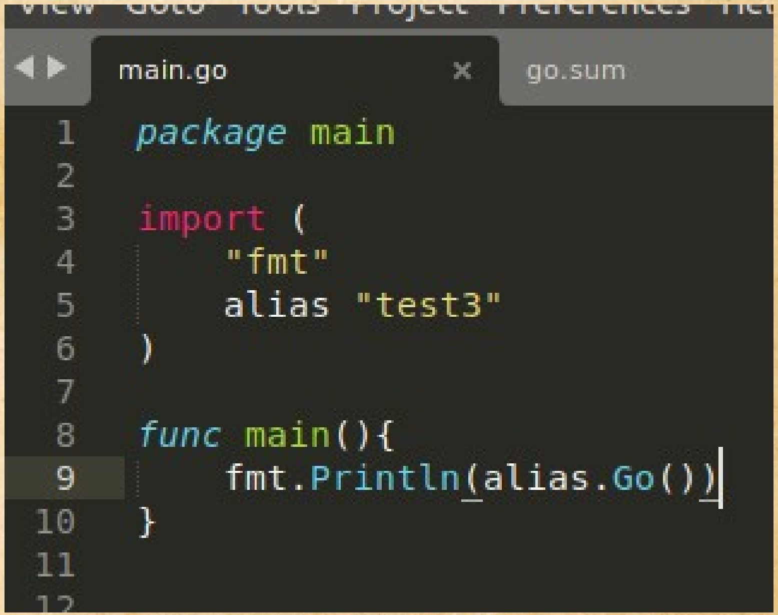
```
// ## case 1 : replace online package with local modified version #####
replace rsc.io/quote  => ../rsc.io/quote@v1.5.2
replace rsc.io/sampler  => ../rsc.io/sampler@v1.3.0
```

Add alias to unstable online package

- Application scenario: when some package is out of maintenance,has variant version,we only want to change one line of code in go.mod instead of changing import in everywhere



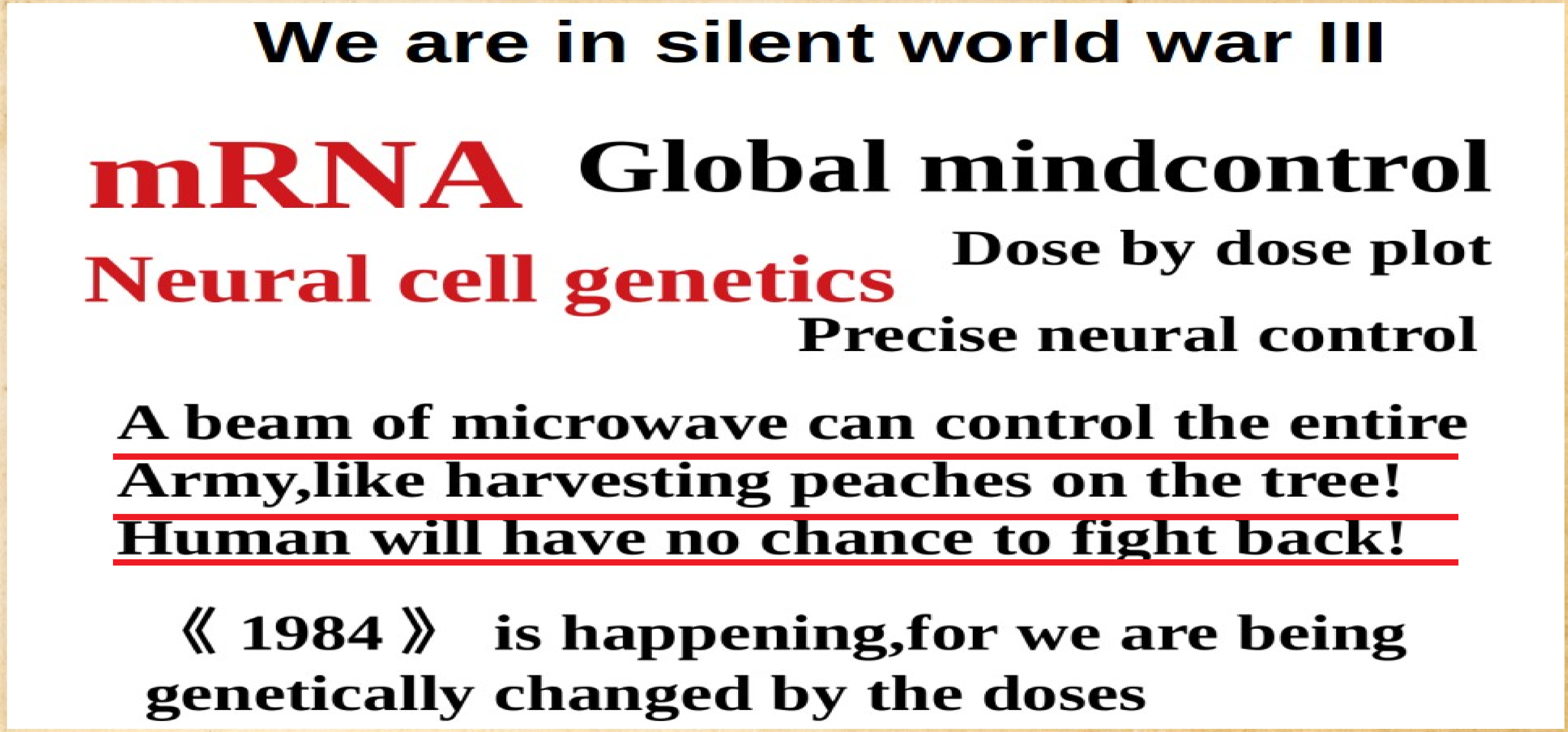
```
module test1

go 1.17

replace test3 => rsc.io/quote v1.5.2
```



```go
package main

import (
	"fmt"
	alias "test3"
)

func main(){
	fmt.Println(alias.Go())
}
```

Module name test3 is just kind of like a pointer connection alias and rsc.io/quote v1.5.2, the online package which might be changed later,actually for all online packages,I recommend using such import method so that it is convenient to switch between modules

Global mRNA genetics,global mindcontrol



We are in silent world war III

mRNA Global mindcontrol

Neural cell genetics

Dose by dose plot

Precise neural control

A beam of microwave can control the entire Army,like harvesting peaches on the tree! Human will have no chance to fight back!

《 1984 》 is happening,for we are being genetically changed by the doses

Pdf : http://www.decensormedia.org/mindcontrol

Global mRNA genetics,global mindcontrol

Brain wave! AI brain control! 두뇌 제어

그들은 우리의 두뇌를 읽고 있습니다

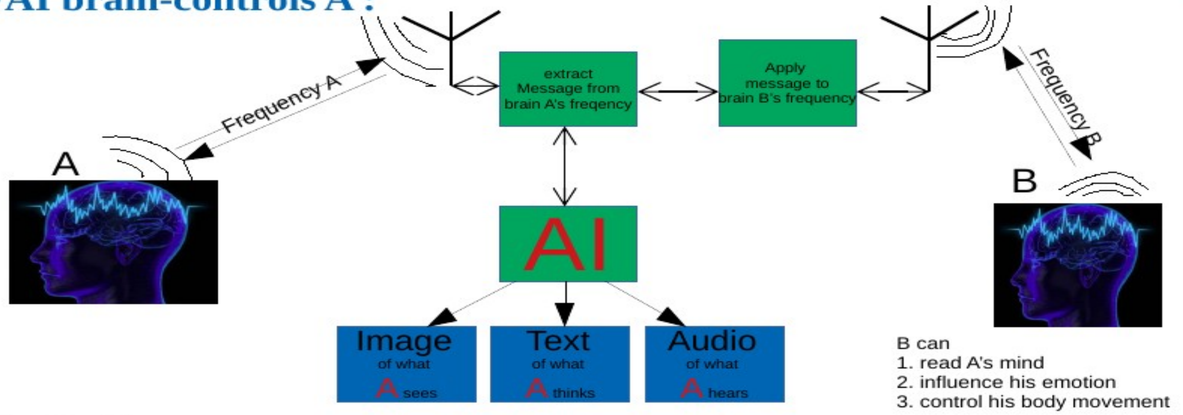
They **read our mind**, Influence our **emotion**

Even **control body movement** when kidnap

We need decentralization:www.decensormedia.org

A picture explains how B/AI brain-controls A :

Military군사 수준 VS Commercial상업 급료



B can
1. read A's mind
2. influence his emotion
3. control his body movement

SOS: **hackers destroying my online job,isolating/corner me,stalking me!**

China Operation Fox Hunt, Kidnap

Pdf : http://www.decensormedia.org/mindcontrol

Global mRNA genetics,global mindcontrol

you don’t hear this from the mainstream media, because big figures are mindcontrolled! They will be facing lawsuit and in big trouble, even life danger if they go against the dark shadow!

Mindcontrol is far beyond just reading your brain knowing your secret and set a trap for you, so they have some evidence to make you end up in jail for thousand years long!

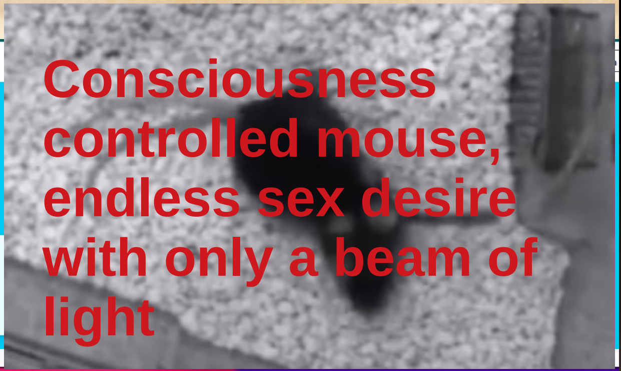
To be honest I still feel like in a dream to realize that they are so mad to inject uniform-slavery-control gene into our blood and our offspring’s blood!

We see many many cases of people dying from heart problem, these are live feedback test and warning signals as well ! Because our information is severely censored, many accounts are shut down for talking about the dose! So you should smell something familiar ! NO body has the right to silence anybody in the name of love or their justice standard! Only they are afraid of something or try to hide something ugly!

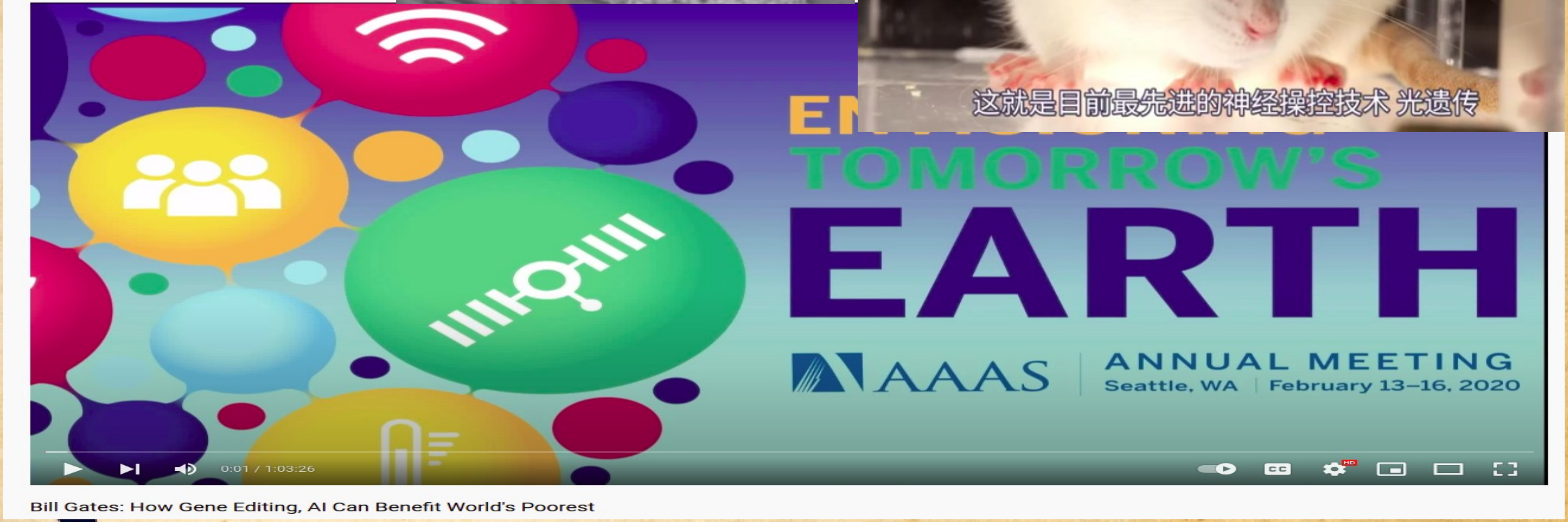
Remember one very basic physics: light is electromagnetic wave! If they can control mouse’s sex intercourse desire with a beam of light, they can also control our heart and many other organ with a beam of electromagnetic wave!

Pdf : http://www.decensormedia.org/mindcontrol

Bill Gates: How Gene Editing, AI can Benefit World's Poorest


Consciousness controlled mouse, endless sex desire with only a beam of light

Optogenetics gene editing engineering
这就是目前最先进的神经操控技术 光遗传



Bill Gates: How Gene Editing, AI Can Benefit World's Poorest

Pdf : http://www.decensormedia.org/mindcontrol

Wake up & Thank you

伍成和 ChengHe Wu
www.deCensorMedia.org
wuchenghe@vk.com
https://github.com/brianwchh