

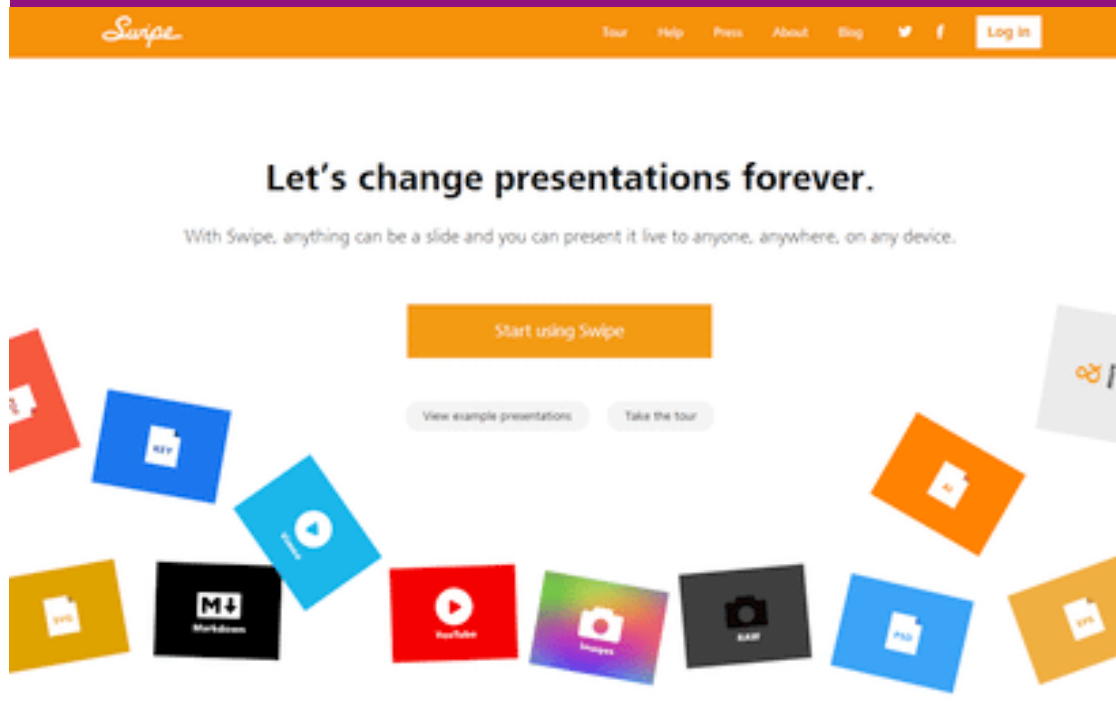


# go-swipe

[kaakaa/swipe-go](#)

# Swipe

- Swipe – simple, easy, interactive presentations.
- Markdownからスライドを作成出来るサービス





# go-swipe

- What?
  - Go言語製
  - Gists で書いたMarkdownをSwipeにアップロードするツール
- Why?
  - 勉強会用に集めた情報をGistsにまとめることが多かった
  - Keynote作るのが面倒になる時があるので、情報まとめたMarkdownからスライドが作れると良い
- How?
  - コマンドラインからインタラクティブに
- Another?
  - hakimel/reveal.js
  - Remark

```
C:\Users\kaakaa_hoe\Documents\github\swipe-go-test>ls
conf.json  main.go

C:\Users\kaakaa_hoe\Documents\github\swipe-go-test>
C:\Users\kaakaa_hoe\Documents\github\swipe-go-test>cat conf.json
{
  "Gist" : {
    "User": "kaakaa",
    "DocId": "29ceacc3a8fa7b86f6bd"
  },
  "Swipe" : {
    "Email": "stooneer.hoe@gmail.com",
    "Password": "12345678",
    "Coloring": true
  }
}
C:\Users\kaakaa_hoe\Documents\github\swipe-go-test>cat main.go
package main

import (
    "github.com/kaakaa/swipe-go"
)

func main() {
    swipe.SwipeUpload()
}

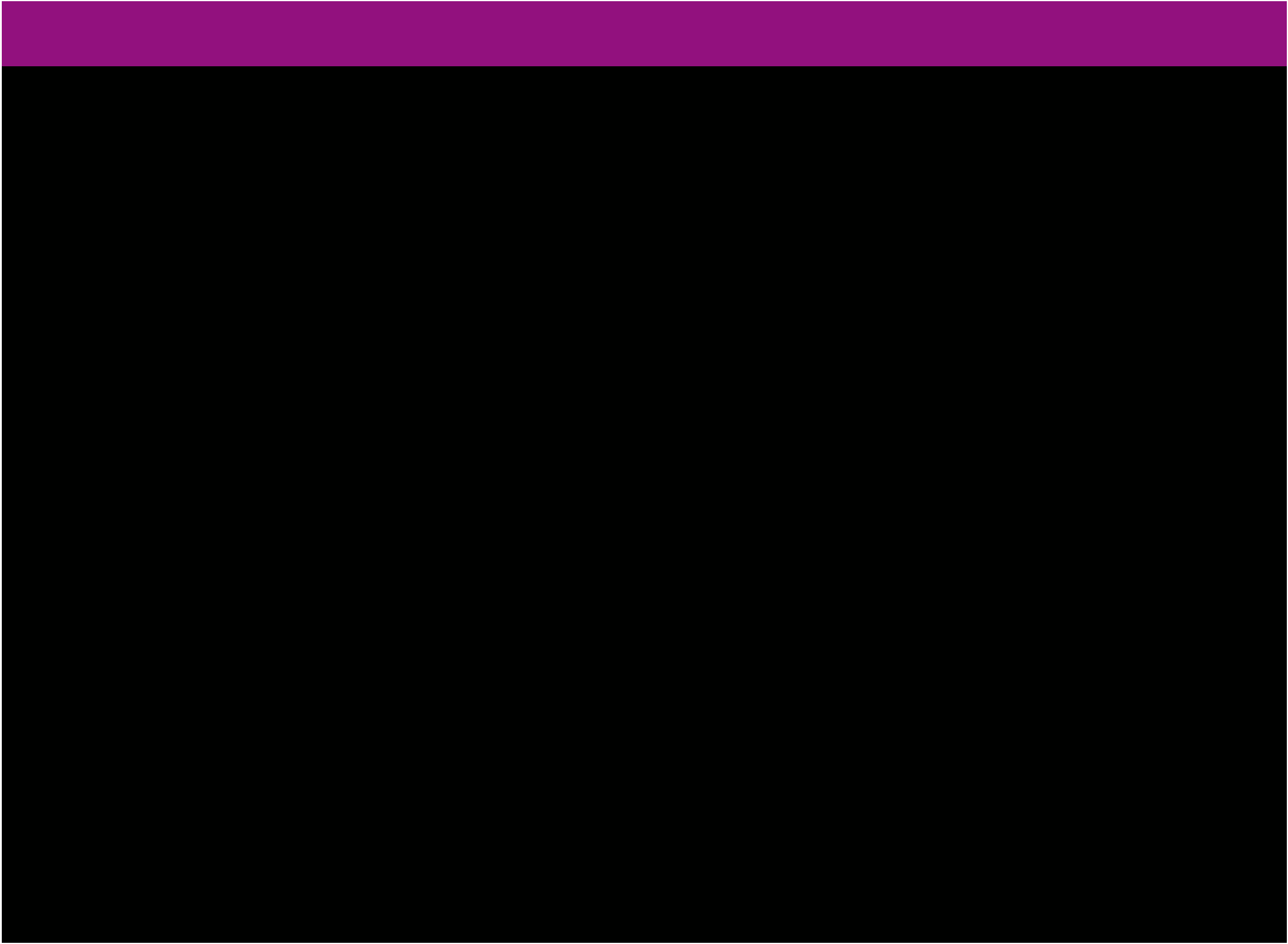
C:\Users\kaakaa_hoe\Documents\github\swipe-go-test>
C:\Users\kaakaa_hoe\Documents\github\swipe-go-test>go run main.go
Gist Document Infomation
  Gist User ID(default: kaakaa)? kaakaa
  Gist Document ID(default: 29ceacc3a8fa7b86f6bd)?

info: Downloading Gist File 'https://gist.githubusercontent.com/kaakaa/29ceacc3a8fa7b86f6bd/raw/slide.md'
Complete Downloading (7195 Bytes)

Input Swipe.to Account Info
  Swipe Email(default: stooneer.hoe@gmail.com)?
  Swipe Password?

Complete Uploading ==> https://www.swipe.to/edit/a5f7e50dddfc43d6ba00a986a06f04cd

C:\Users\kaakaa_hoe\Documents\github\swipe-go-test>|
```



# Go言語

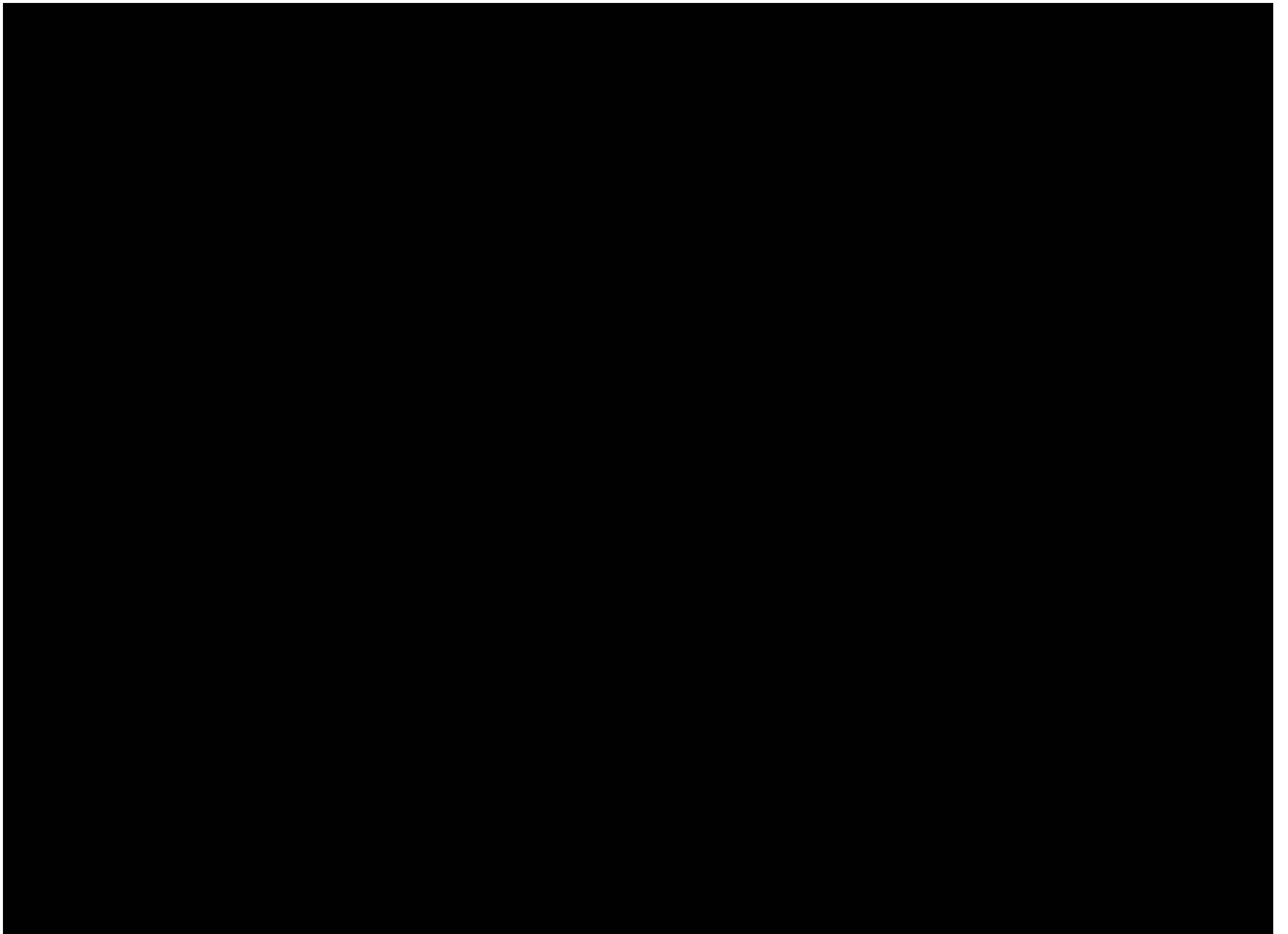
- Docker界限とかで使われている印象
- 

Googleは2009年11月10日(米国時間)、オープンソースのプログラミング言語「Go」を発表しました。  
Go言語は、Linux、Mac、Native Clientで動作する開発言語で、Android携帯上でも動作します。  
まだ発表されたばかりなのでこれからの動向が注目されています。

特徴はGoogleによると・・・

- シンプルな言語である。
- コンパイル・実行速度が早い。
- 安全性が高い。
- 同期処理が容易に行える。
- なにより楽しい。
- オープンソースである。





# A Tour of Go

- Go言語の特徴を学べるREPL

```
1 package main
2 import "fmt"
3
4 func main() {
5     fmt.Println("Hello, 世界")
6 }
7
```

RUN ▼

Go 1

## A Tour of Go

### Hello, 世界

プログラミング言語Goツアー (Go言語基礎文法超速マスター) へようこそ！

このツアーには3つのセクション (基本コンセプト、メソッドとインターフェース、並行性) があります。

ツアー中には演習課題 (Exercise) もありますので、修了するために挑戦してみてください。

このツアーはインタラクティブです。コンパイルするために、RUNボタンをクリック (またはShift+Enter) し、プログラムをリモートのサーバ上で実行してみてください！ 実行結果はコードの下に表示されます。

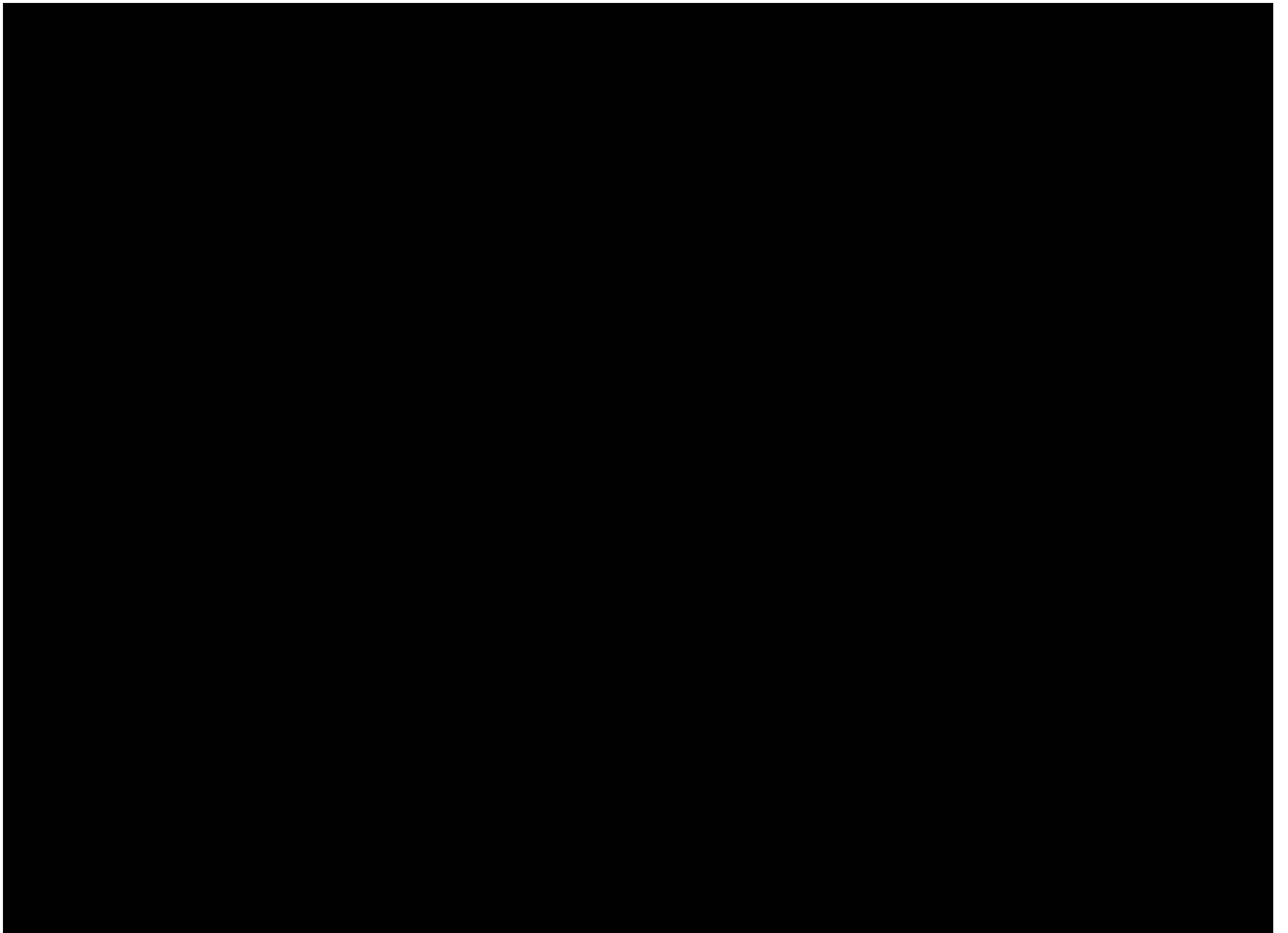
ツアーにあるサンプルプログラムは、Goの特徴を示しています。ここにあるプログラムは、Goを学習するための出発点となるでしょう。

プログラムを編集してもう一度実行してみてください！

それでは下にある右矢印のボタンをクリックするか、Page Downキーを押して次に進みましょう。なお、ページの上にある"Go"の旗でツアー全体の目次を見ることができます。

注：本サイトは、A Tour of Go を日本語訳したものです。日本語翻訳プロジェクトはこちらです。





# Hello World

```
package main

import (
    "fmt"
)

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

- main/パッケージのmainメソッドがエントリとなる
- 最初が大文字なら、その名前はエクスポートされる(publicとなる)

# Goの実行

```
go run main.go
```

- 依存ライブラリをダウンロード

Asciinema?

# Multiple Results

## Interface

```
func Open(name string) (file *File, err error) {  
    ...  
}
```

## Usage

```
fi, err := os.Open(path)  
if err != nil {  
    panic(err)  
}
```

# goroutine / channel

- 並行処理
  - Shared-memory communication
  - Message-passing communication
- Message-passing communication でも2種類
  - Actor Model (おそらくErlangとかScalaのAkka)
  - CSP (Goroutine)
- 下記が参考になった
  - 
  -

# goroutine sample

```
func main() {  
    ch := make(chan string)  
    end := make(chan int)  
  
    go func() {                // goroutineはブロックしない  
        fmt.Println(<-ch)      // chにinputがあるまでブロックする  
        end <- 0  
    }()  
  
    ch <- "start"  
    <-end                      // endにinputがあるまでブロックする  
    fmt.Println("finish")  
}
```



# go-swipeについて



# 処理内容

- Gistsからファイルをダウンロード
- Swipe
  - ログイン(Cookie取得)
  - 新規ドキュメント作成
  - ファイルアップロード

# Gistからファイルをダウンロード

`https://gist.githubusercontent.com/${userid}/${gistid}/raw/${filename}`

- 必要な情報をコマンドラインから入力
  - `conf.json`を用意しておけば `Enter` 押してるだけでいい

# Swipe (ログイン)

- ログインページのソースからログイン処理に必要な情報をHTMLソースから調べる

```
var str = []byte("email=" + url.QueryEscape(email) + "&" + "password=" + pass)
req, _ := http.NewRequest("POST", "https://www.swipe.to/login",
bytes.NewBuffer(str))
req.Header.Set("Referer", "https://www.swipe.to/home")
req.Header.Set("Content-Type", "application/x-www-form-urlencoded")
```

# Swipe (新規ドキュメント作成)

- ログインと同じくHTMLソースから、新規ドキュメント作成に必要な情報を調べる
  - Reactが動いてるらしいので、jsファイルも見る必要があった

```
req, _ := http.NewRequest("POST", "https://www.swipe.to/edit/create", nil)
req.Header.Set("Referer", "https://www.swipe.to/home")
```

- responseとしてドキュメントIDが返るので、保持しておく
  - 次のファイルアップロードに必要

# Swipe (ファイルアップロード)

- GistからダウンロードしたMarkdownをアップロードする
- 同じくHTMLソースを調べる
  - ファイルアップロードなのでmultipartパッケージ使ってPOST

```
req, _ := http.NewRequest("POST", "https://www.swipe.to/edit/upload", b)
req.Header.Set("Content-Type", contenttype)
req.Header.Set("Referer", "https://www.swipe.to/edit/" + id)
```

まとめ

# まとめ

- go-swipe
  - APIの変更に対して脆弱
  - tempファイルが削除されない...
- Go言語
  - Cっぽいのであまり得意でない
  - 例外処理の考え方が変わっているらしく、慣れない
- HTTPの簡単なGET/POSTあたりの考え方はわかってきた
  - ログインが必要なサイトでもcurlで試し打ちしながら try&error すれば何とかなることが分かった
  - イントラクローラに活着ている
  - でも、結構地味な作業で辛い