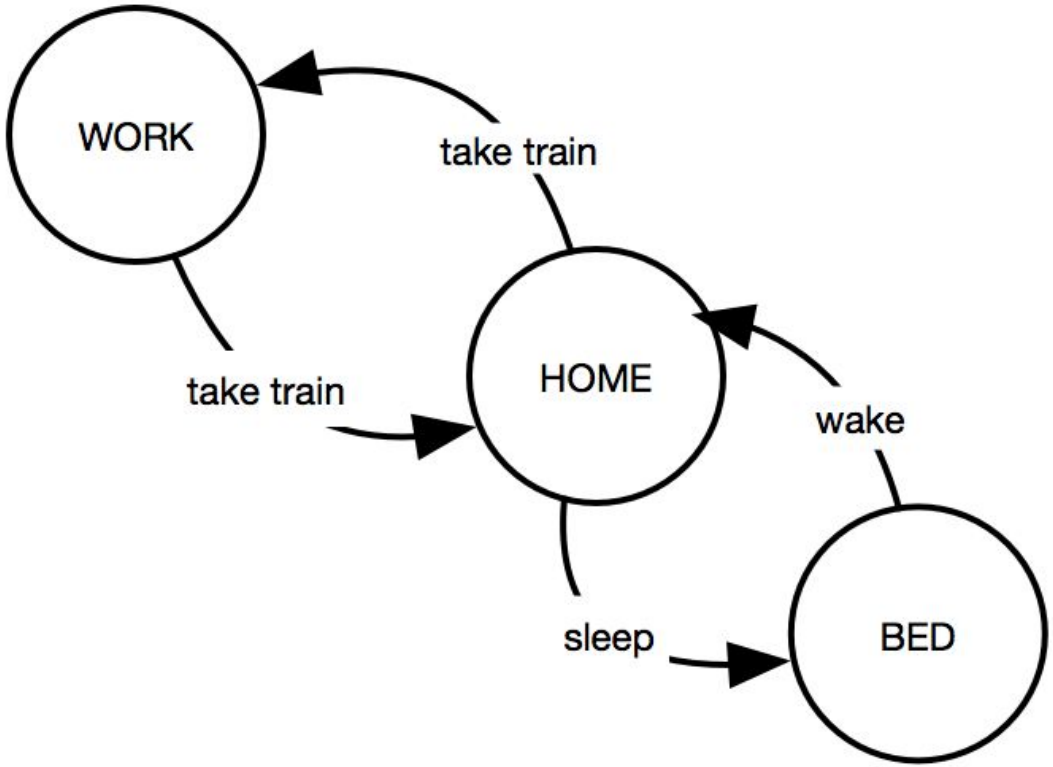
Finite State Machine



END

Шо то и зачем ?

Знакомьтесь, это Вася!

Ему нужно сделать анимированную менюшку!

Начинает он с одного выпадающего окошка, тестирует анимацию, выставляет ease out 100% и наслаждается полученным результатом. Но вскоре он понимает, что для того, чтобы управлять менюшкой, хорошо бы знать закрыто оно сейчас или нет. Мы-то с вами тут программисты опытные, все понимаем, что нужно добавить флаг. Не вопрос, флаг есть.

```
var opened = false
```

Вроде, работает. Но, если быстро кликать по кнопке, меню начинает моргать, открываясь и закрываясь не успев доанимироваться в конечное состояние. Вася добавляет флаг *animating*. Теперь код у нас такой:

```
var opened = false
var animating = false

func onClick(){
  if animating { return }
  if opened {
    close()
  } else {
    open()
  }
}
```

Через какое-то время Васе говорят, что меню может быть полностью выключено и неактивно. Не вопрос! Мы-то с вами тут программисты опытные, все понимаем, что… нужно добавить ЕЩЕ ОДИН ФЛАГ! И, всего-то через пару дней разработки, код меню уже пестрит двустрочными IF-ами типа вот такого:

```
if enabled && opened && !animating && !selected &&
finishedTransition && !endOfTheWorld && ... { ... }
```

Вася начинает задаваться вопросами:
как вообще может быть, что animating == true и enabled == false
почему у него время от времени все глючит
как тут вообще поймешь в каком состоянии находится меню.
Ага! Состояния...
О них дальше и пойдет речь.

На цьому місці вже всі точно перестали сомніватись шо слайди не моє...

// TODO: OZ Show some shit code!

The easiest State Machine in Swift

// TODO: Open E1.playground and say something smart!

TOO EASY ?

Simple -> Complicated!

// TODO: Open E2.playground wait some time and say something funny!

Have no idea what to add more!

TODO: \\ Make it looks like NO Internet connection!