ЛЕКЦИЯ 1

Знакомство с GO

Дата релиза GoLang

GoLang был официально представлен корпорацией **Google** – **10 ноября 2009** года, спустя 2 года после начала его разработки (в ноябре 2007).

Последняя версия: **v1.13.7**

Официальный сайт: **golang.org**

Поддерживаемые ОС

- Linux
- MacOS
- Microsoft Windows
- iOS
- Android
- BSD
- DragonFly BSD
- FreeBSD
- NetBSD
- OpenBSD
- Plan 9
- Solaris

*Компиляция под iOS & Android возможна с помощью gomobile

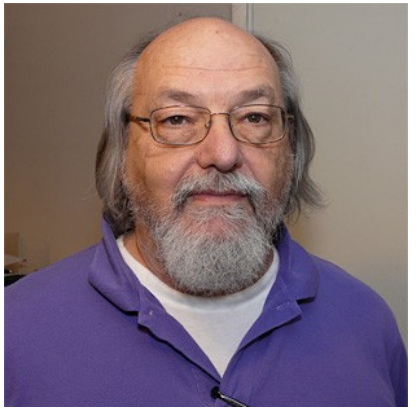
Авторы языка

Роб Пайк

Один из создателей кодировки UTF-8, а также ОС: Plan9 и Inferno

Роберт Гризмер

Один из разработчиков JavaScript движка Google V8



Кен Томпсон

Один из создателей языка B (который оказал влияние на C), а также ОС Unix

Бизнес ценность GO

Скорость разработки:
Syntax Sugar + абстрагируемся от низкоуровневых деталей,
не теряя эффективности
(управление памятью и потоками)

Удобство сопровождения:
Легкость применения DDD & TDD,
противоположность JavaScript
(node_modules)

Быстродействие и ресурсоэффективность:
например fasthttp
200,000 rps & 1.5M открытых
соединений на один физ. сервер

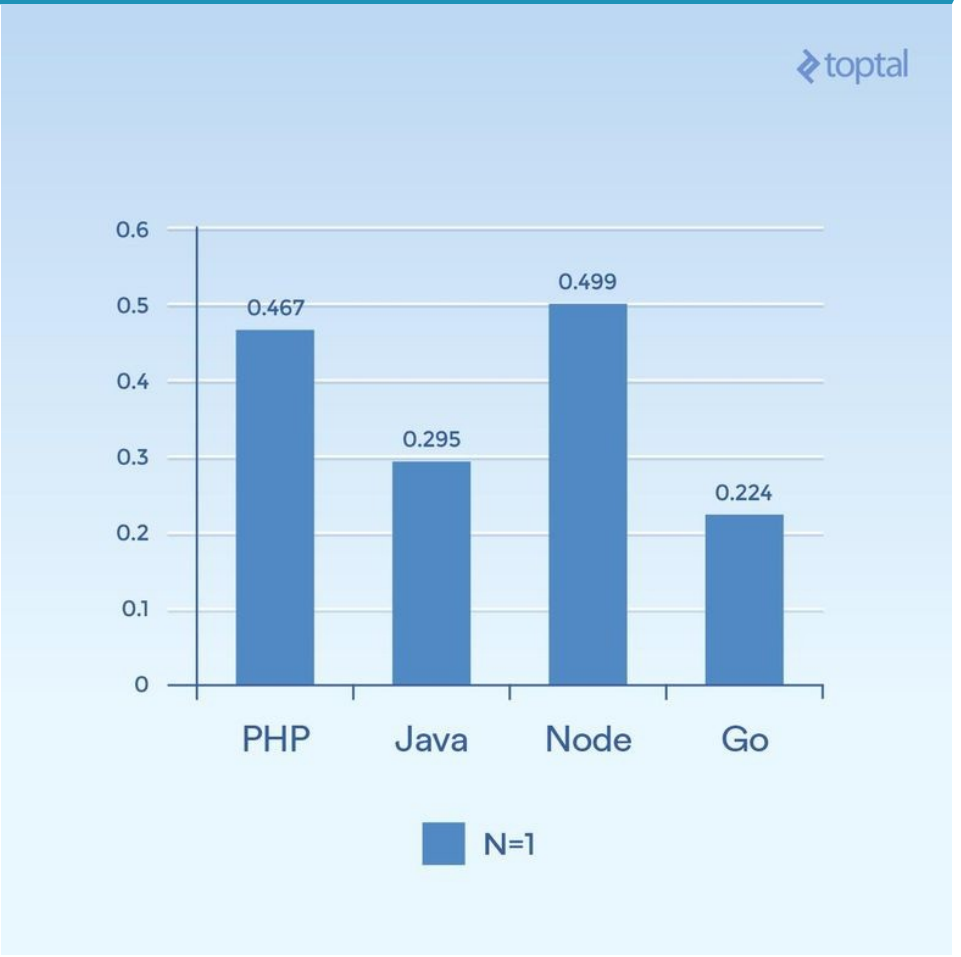
Бенчмарки

Описание теста:
2000 итераций
300 одновременных запросов

Задача:
Единократное хэширование SHA256, по каждому запросу (N = 1)

Результат:

NodeJS – 0.499 мс
PHP – 0.467 мс
Java – 0.295 мс
Go – 0.224 мс



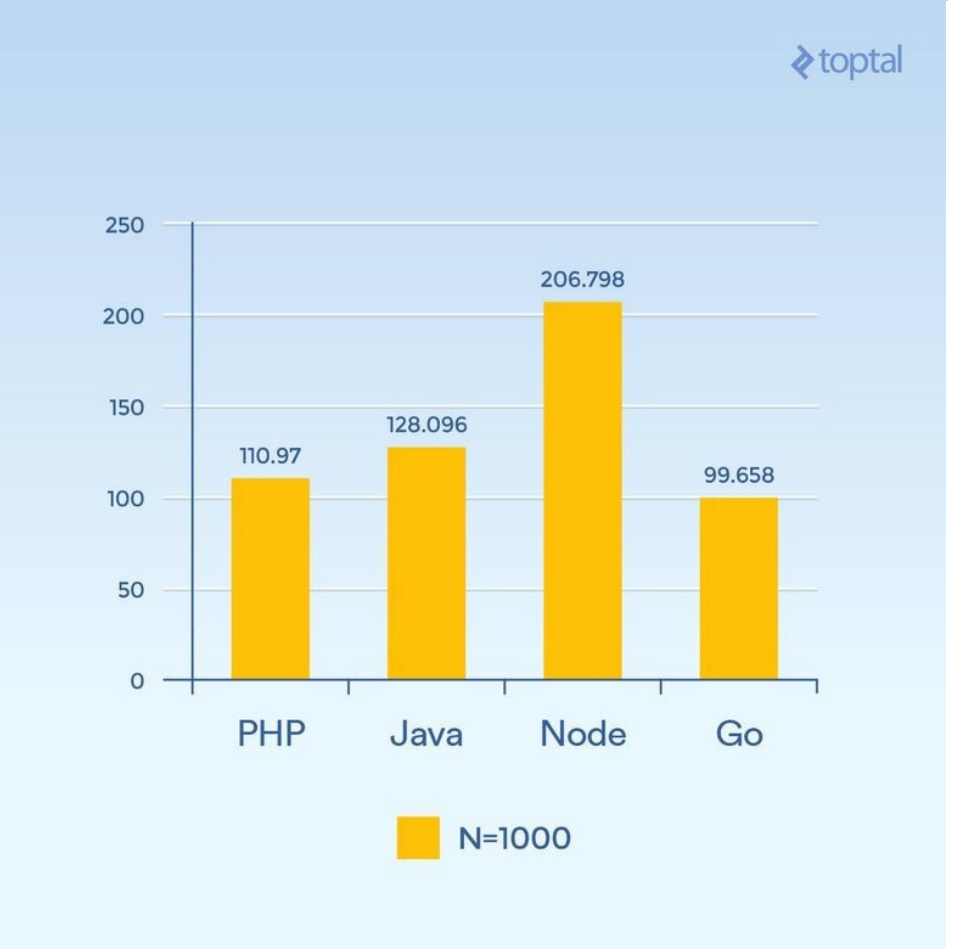
Бенчмарки

Описание теста:
2000 итераций
300 одновременных запросов

Задача:
Тысячекратное хэширование SHA256, по каждому запросу (N = 1000), CPU test

Результат:

NodeJS – 206.798 мс
Java – 128.096 мс
PHP – 110.97 мс
Go – 99.658 мс



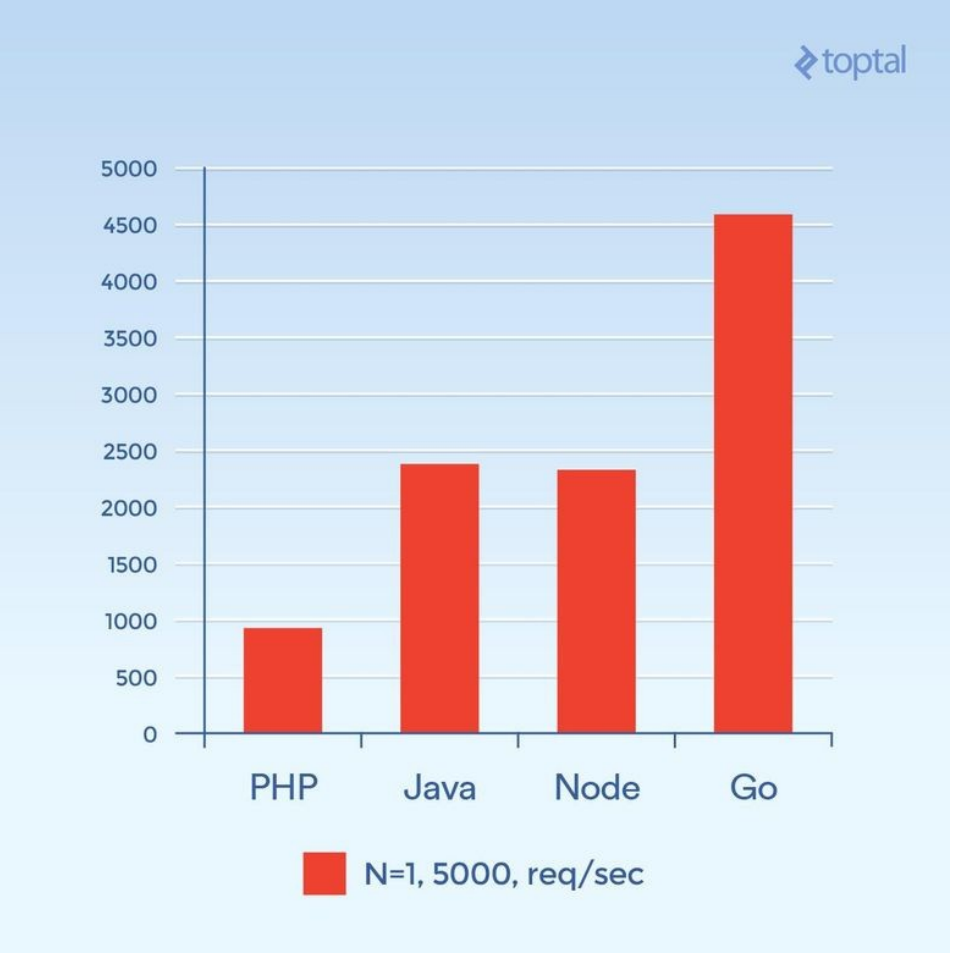
Бенчмарки

Описание теста:
2000 итераций
5000 одновременных запросов

Задача:
Единократное хэширование SHA256, по каждому запросу (N = 1)

Результат:

PHP – <1000 rps
NodeJS – <2500 rps
Java – <2500 rps
Go – >4500 rps

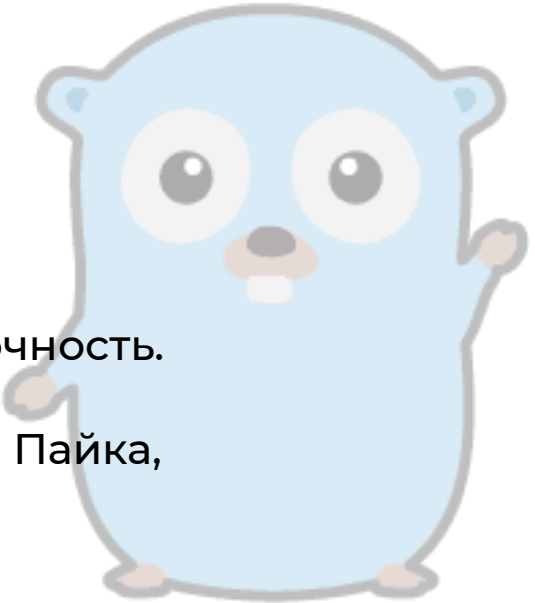


Причины успеха GO

Что действительно сделало Go успешным?

Часто называют ряд причин:

- Скорость компиляции
- Скорость исполнения
- Простой деплой
- Утилиты
- Библиотеки



Иногда называют реализацию интерфейсов или многопоточность.

Все перечисленное - важные моменты, но по мнению Роба Пайка,
на самом деле не являются ключевой причиной успеха Go.

Что делают другие языки?

Выходят новые версии языков Java, JavaScript (ECMAScript), Typescript, C#, C++, PHP и др.

Наблюдается тенденция активного заимствования разных фич друг у друга.

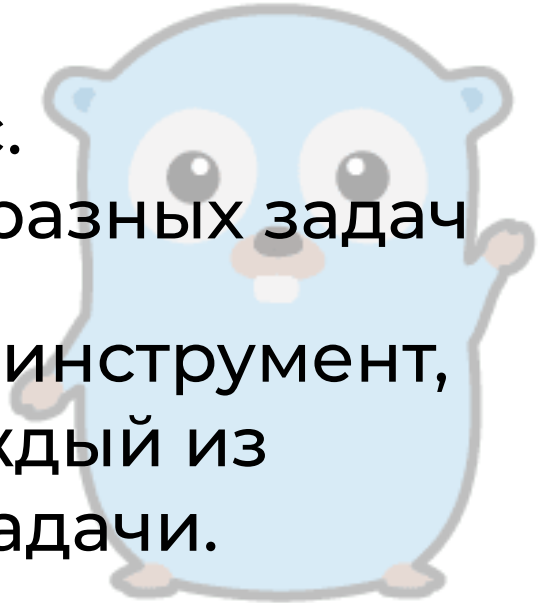
Это ведет к конвергенции, все эти языки по сути превращаются в один большой и тяжелый язык.

Задача - инструмент

Если языки сконвергировались, при разработке на них мы будем думать одинаково.
Ортогональность теряется (in/ext).

Но разные пути мышления это плюс.
Нужны разные языки для решения разных задач

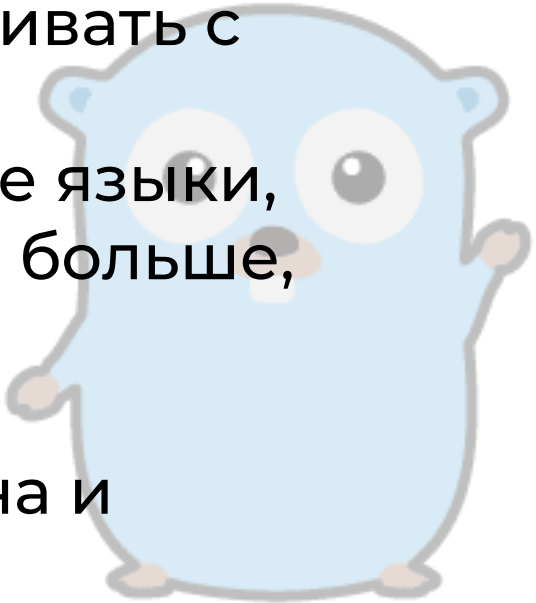
Нам не нужен один универсальный инструмент, нам нужен набор инструментов, каждый из которых лучший в решении своей задачи.



Простота залог успеха

Go пошел совсем другим путем.
Настоящая причина его успеха: простота

Go – очень прост и понятен, если сравнивать с другими устоявшимися языками.
Он не пытается быть похожим на другие языки, нет погони за фичами (они сделают его больше, но не лучше).



Простота Go на самом деле многогранна и сложна.

Читабельность

Если у языка много фич, Вы будете тратить время выбирая среди них. А когда выберете и реализуете, возможно придется переосмыслить и переделать.

Спустя время возникнет вопрос у Вас, или у разработчика после Вас:
"Почему код работает именно так?"

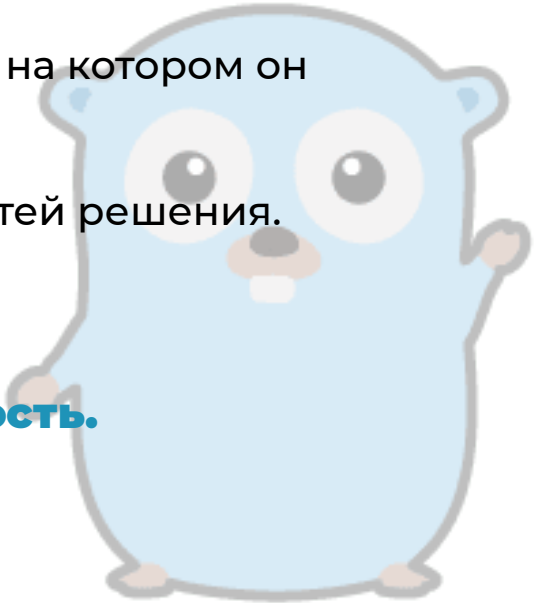
Код тем более сложен для понимания, чем более сложен язык на котором он написан.

Предпочтителен один путь или как можно меньше простых путей решения.

Фичи увеличивают сложность. Нам нужна простота.

Фичи ухудшают читабельность. Нам нужна читабельность.

Читабельность первостепенна.



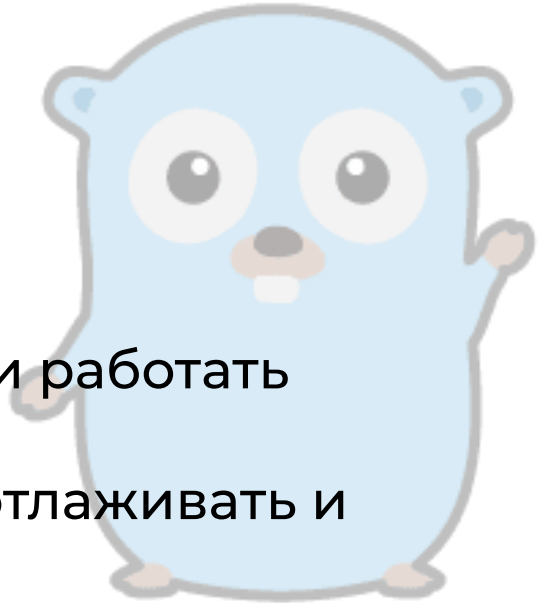
Читабельно = надежно

Читабельный код – это надежный код.

Он проще для понимания.

Он проще для работы над ним.

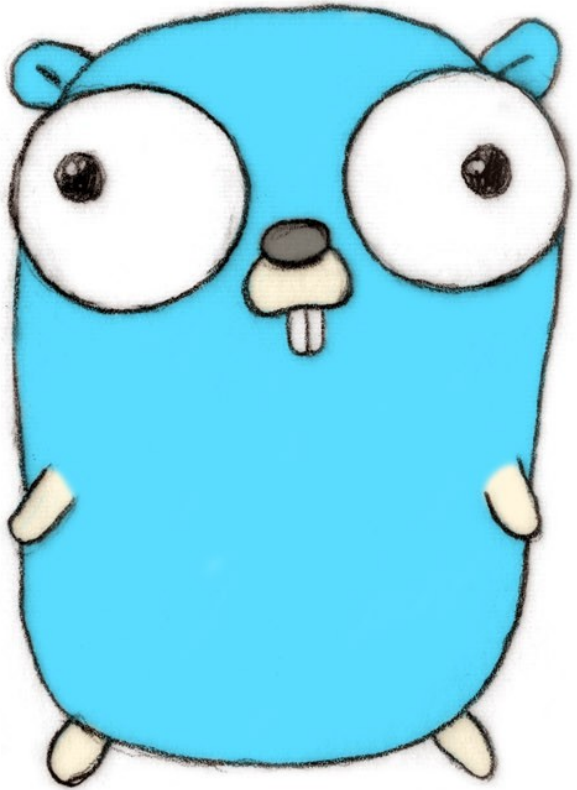
Если сломается, его просто починить.



Если язык сложный:

- Вы должны знать много вещей чтобы читать и работать над кодом.
- Вы должны понимать много нюансов чтобы отлаживать и чинить его.

Простота выразительна

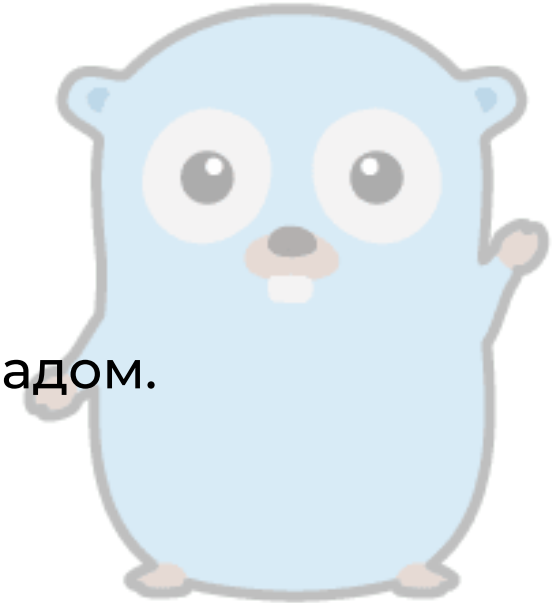


Простые вещи...

- **Сборщик мусора**
- **Горутины**
- **Константы**
- **Интерфейсы**
- **Пакеты**

Очень просты в использовании,
но на самом деле сложно устроены.

Каждый скрывает сложность за простым фасадом.



Пример

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", HelloServer)
    http.ListenAndServe(":8080", nil)
}

func HelloServer(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello world!")
}
```