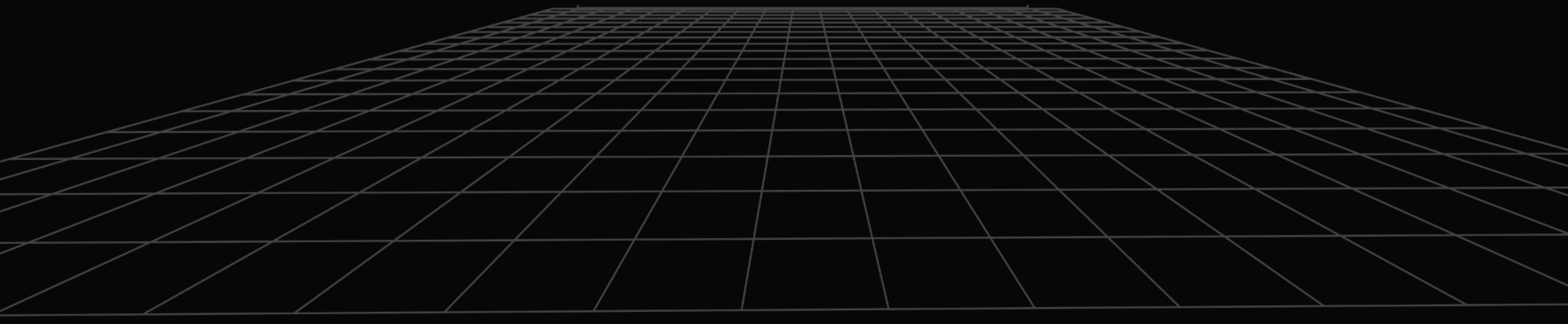


СТРУКТУРЫ

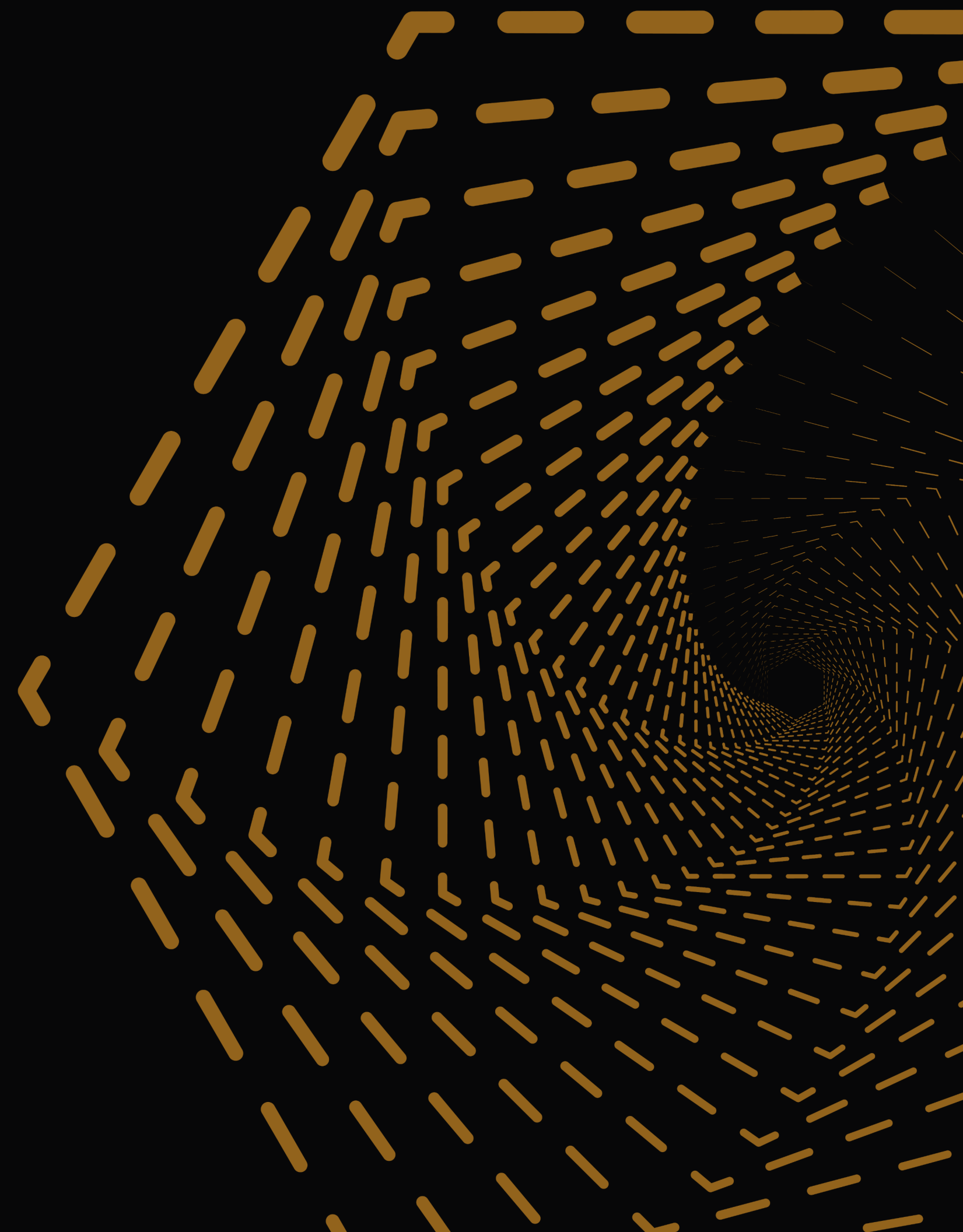


ПРОВЕРЬ ЗАПИСЬ



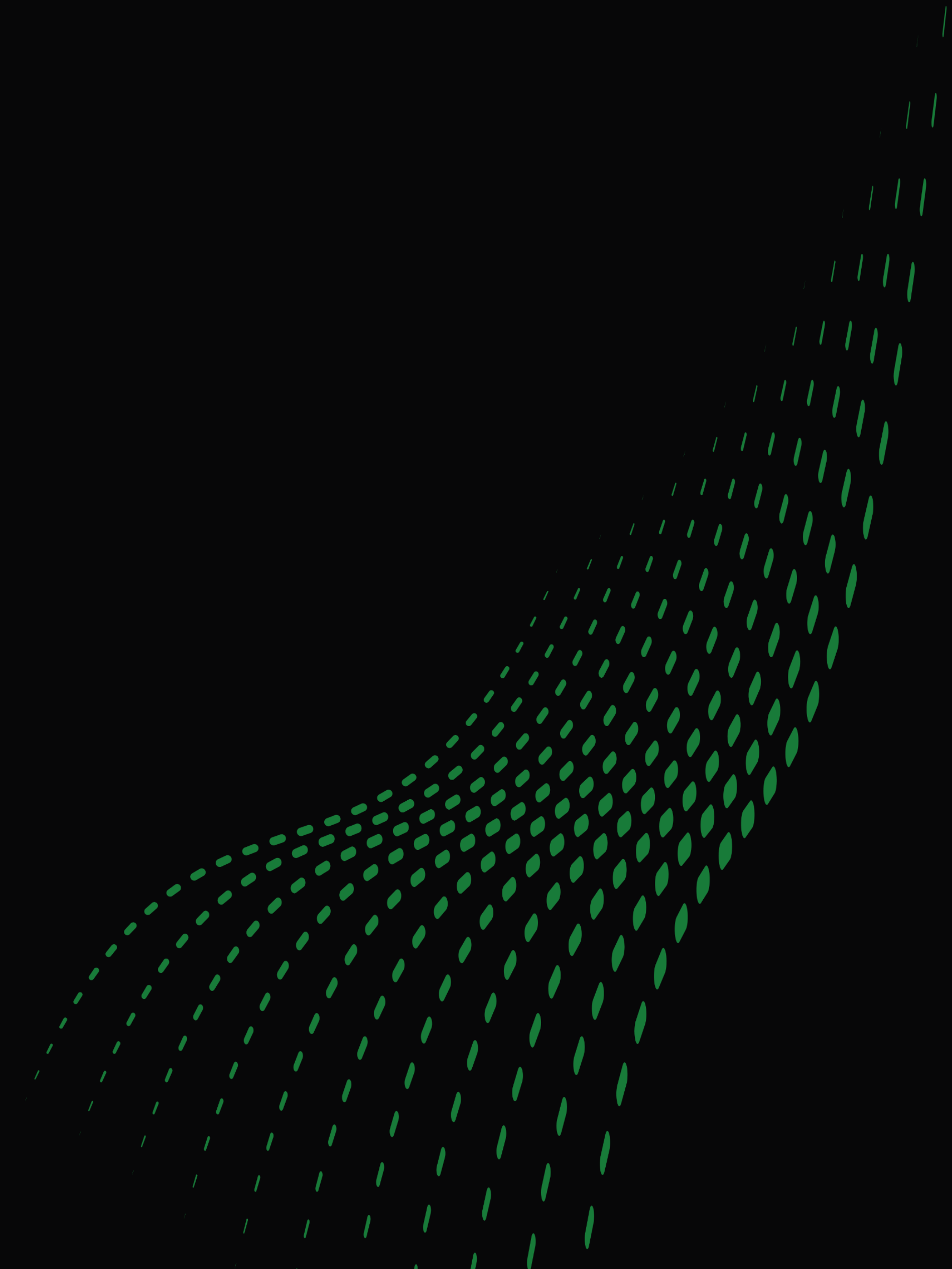
ПРАВИЛА ЗАНЯТИЯ

1. вопросы в чате можно задавать
в любое время
2. вопросы голосом задаем
по поднятой руке в Zoom
3. ответы на вопросы будут
в запланированных местах



ПЛАН ЛЕКЦИИ

1. Структуры
2. Встраивание типов
3. Выравнивание структур
4. Тонкости структур
5. Паттерны
6. Псевдонимы и определения типов



СТРУКТУРЫ

СТРУКТУРЫ ДАННЫХ

Программная единица, позволяющая хранить и обрабатывать однотипные и/или логически связанные данные



```
1 type Person struct {  
2     name    string  
3     surname string  
4     age     int  
5 }
```

EXAMPLE

Struct

EXAMPLE

Copy struct

Terminal: deep_go × + ∨



МЕТОД

В объектно-ориентированном программировании
– это **функция или процедура**, принадлежащая
какому-то классу или объекту

EXAMPLE

Struct sugar

EXAMPLE

Blank methods

Terminal: deep_go × + ∨



ПОЛУЧАТЕЛЬ

(Receiver) – представляет собой **указатель на текущий объект или объект структуры** (аналог `this` в других языках программирования)

EXAMPLE

Different receivers

Получатель **должен быть** указателем:

- Если метод должен изменить получателя
- Если в получателе есть поле, которое нельзя копировать (например тип из пакета `sync`)

Получатель **следует сделать** указателем:

- Если получатель – большой объект (чтобы его не копировать – размер объекта нужно бенчмаркать)

Получатель **должен быть** значением:

- Если нужно обеспечить неизменность получателя

Получатель **следует сделать** значением:

- Если получатель является базовым типом (int, float64, string, ...)
- Если изменяемые поля не являются частью получателя напрямую, а находятся внутри другой структуры

EXAMPLE

Recursive receiver

По умолчанию **мы можем выбирать тип получателя в виде значения**, если нет веских причин не делать этого, но если возникают сомнения, то использовать указатель

EXAMPLE

Bind receiver

**НЕ НУЖНО СМЕШИВАТЬ
ТИПЫ ПОЛУЧАТЕЛЕЙ!**

EXAMPLE

Nil receiver

Потому что получатель - это синтаксический сахар



```
1 type data struct {}  
2  
3 func (d *data) methodWithSugar() {  
4     // implementation...  
5 }  
6  
7 func methodWithoutSugar(d *data) {  
8     // implementation...  
9 }
```

EXAMPLE

Implicit method call

**GO НЕ ПОДДЕРЖИВАЕТ ОБЪЕДИНЕНИЯ,
НО ИХ МОЖНО РЕАЛИЗОВАТЬ**

EXAMPLE

Union 1

EXAMPLE

Union 2

У структур в Go могут быть теги (набор пар ключей и значений). Теги опциональные – по умолчанию тег каждого поля равен пустой строке



```
1 type User struct {  
2     Name      string `json:"name" xml:"name"`  
3     Surname   string `json:"surname" xml:"surname"`  
4 }
```

EXAMPLE

Struct inside function

EXAMPLE

Anonymous struct

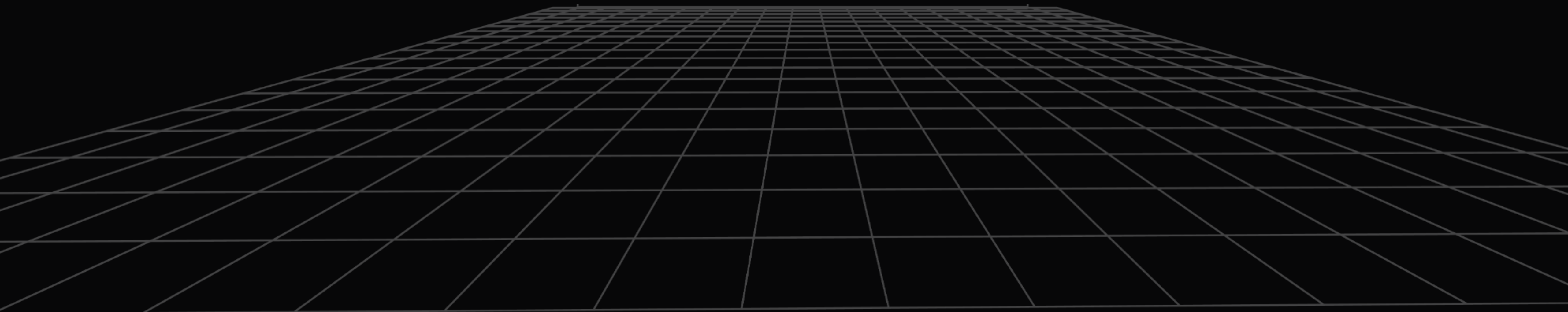
Теги полей и порядок объявлений полей в типе структуры имеют значение для идентификации типа структуры

Два безымянных типов структур идентичны,
только если они имеют одинаковую
последовательность объявлений полей

Поля идентичны, только если их
соответствующие имена, соответствующие
типы и соответствующие теги идентичны

FAQ

Структуры



ВСТРАИВАНИЕ ТИПОВ

ВСТРАИВАНИЕ ТИПОВ



```
1 type Foo struct {  
2     Bar  
3 }  
4  
5 type Bar struct {  
6     data int  
7 }
```

Встроенные поля также еще называют анонимными полями –
однако каждое встроенное поле имеет имя, указанное неявно
(имя типа встроенного поля действует как имя поля)



```
1 type Foo struct {  
2     Bar  
3 }  
4  
5 type Bar struct {  
6     data int  
7 }  
8  
9 func main() {  
10     var foo Foo  
11     foo.data = 100  
12     foo.Bar.data = 100  
13 }
```

Нельзя рекурсивно встраивать себя
(но можно использовать указатели)



```
1 // compilation error
2 type Data struct {
3     Data
4 }
5
6 // compilation error
7 type Data struct {
8     d Data
9 }
10
11 type Data struct {
12     d *Data
13 }
```

EXAMPLE

Skipping selectors

Поэтому их и называют еще **анонимными полями**
— из-за того, что их можно пропустить!

EXAMPLE

Ambiguous selectors

НУЖНО ЯВНО УКАЗЫВАТЬ

к какой встроенной структуре
происходит обращение

data

values	Derived1
values	Derived2

**КОГДА ИСПОЛЬЗОВАТЬ
ВСТРАИВАНИЕ?**

EXAMPLE

Type embedding incorrect

EXAMPLE

Type embedding correct

В GO НЕТ НАСЛЕДОВАНИЯ!

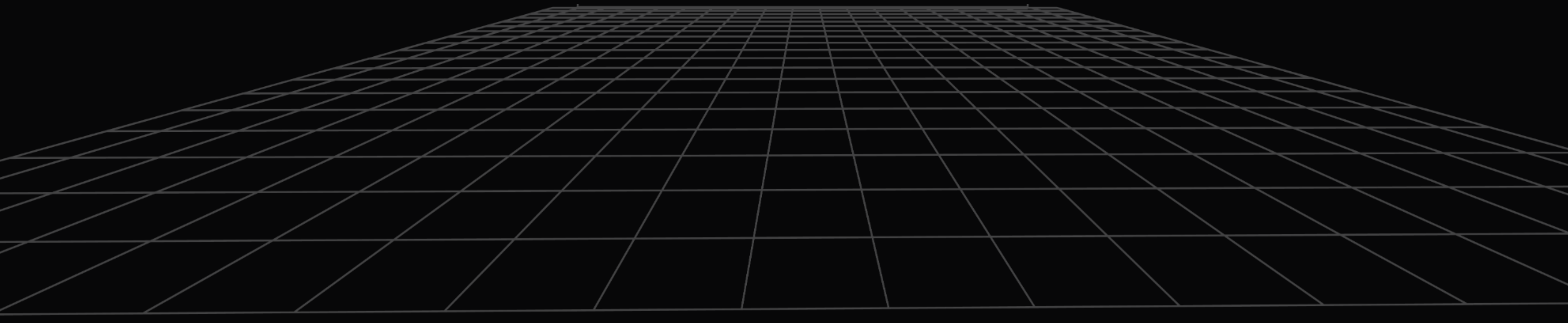
EXAMPLE

Inheritance

**Методы с одинаковой сигнатурой
«перекрываются» при встраивании**

FAQ

Встраивание типов



ВЫРАВНИВАНИЕ СТРУКТУР

**КАК УЗНАТЬ РАЗМЕР
ОБЪЕКТА СТРУКТУРЫ?**

**ВЛИЯЕТ ЛИ КОЛИЧЕСТВО МЕТОДОВ
НА РАЗМЕР ОБЪЕКТА СТРУКТУРЫ?**

EXAMPLE

Struct alignment 1

В GO ИСПОЛЬЗУЕТСЯ

«ТРЕБУЕМОЕ ВЫРАВНИВАНИЕ»

Его значение равно размеру памяти,
требуемому самому большому полю
в структуре

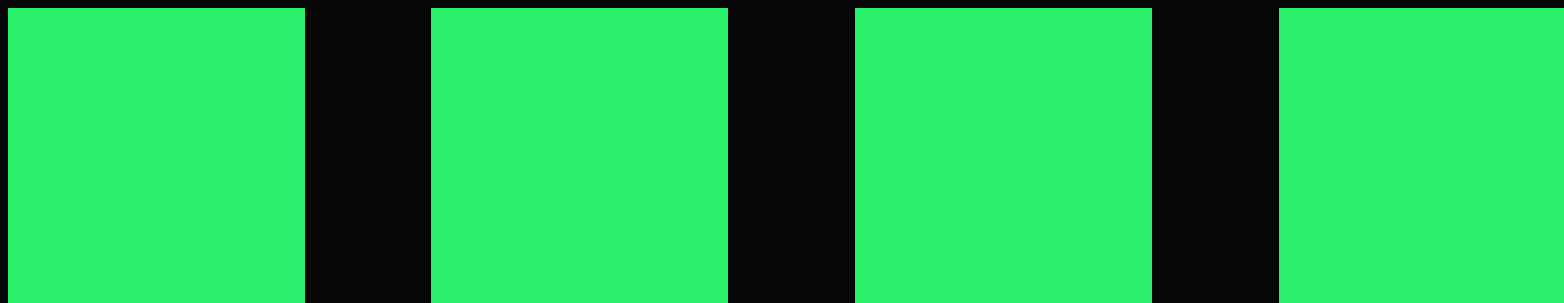


То есть, если в структуре есть только
поля `int32`, то выравнивание составит 4 байта,
а если есть и `int32`, и `int64` – то 8 байтов.

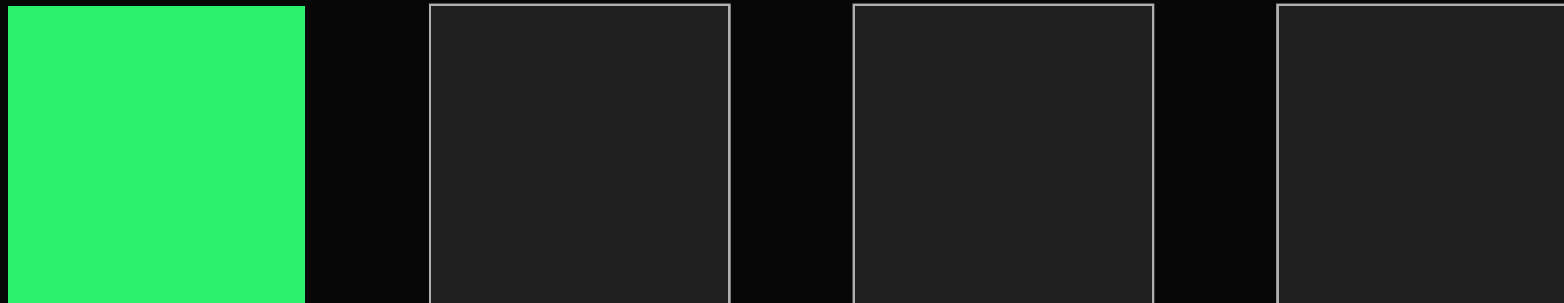
4 bytes



4 bytes

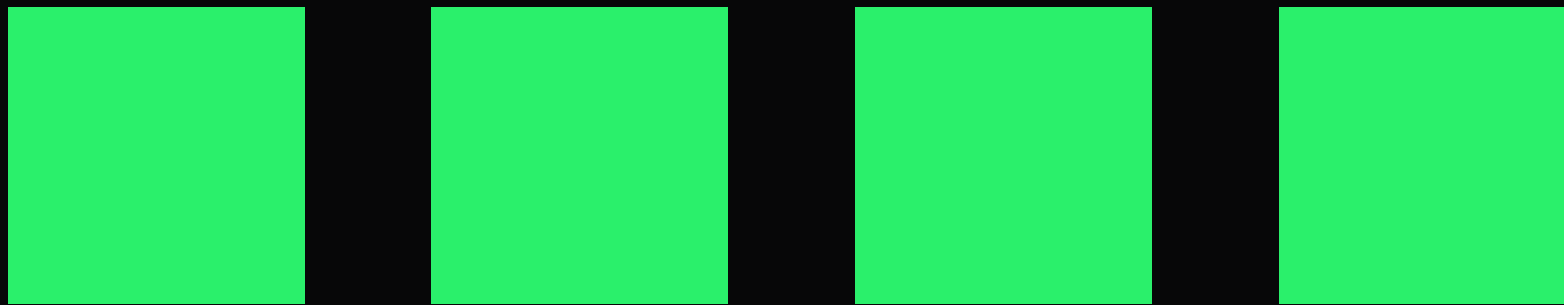


4 bytes

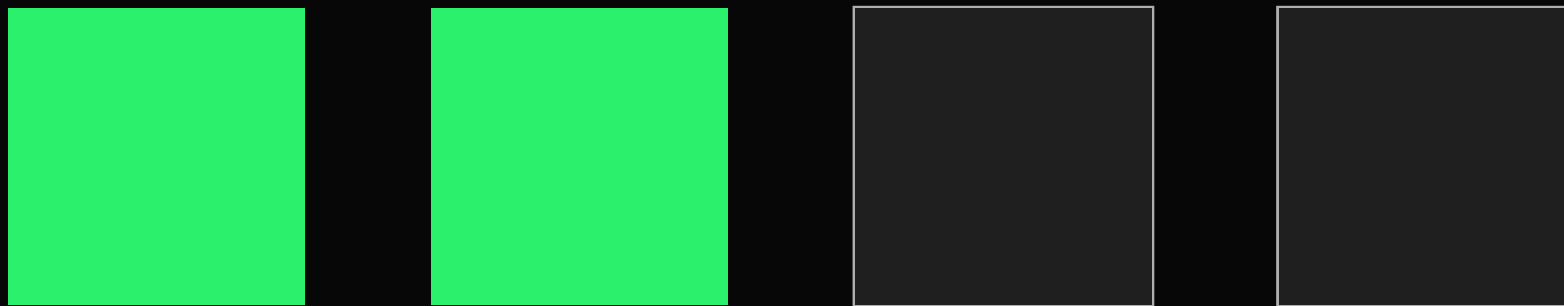


```
1 type data1 struct {  
2     aaa bool  
3     bbb int32  
4     ccc bool  
5 }
```

4 bytes



4 bytes



```
1 type data2 struct {  
2     aaa int32  
3     bbb bool  
4     ccc bool  
5 }
```

Компилятор не меняет порядок полей в структуре
и не может оптимизировать такие случае – **НО МЫ МОЖЕМ**

**НЕИСПОЛЬЗУЕМАЯ ПАМЯТЬ
ЗАПОЛНЯЕТСЯ НУЛЯМИ**

EXAMPLE

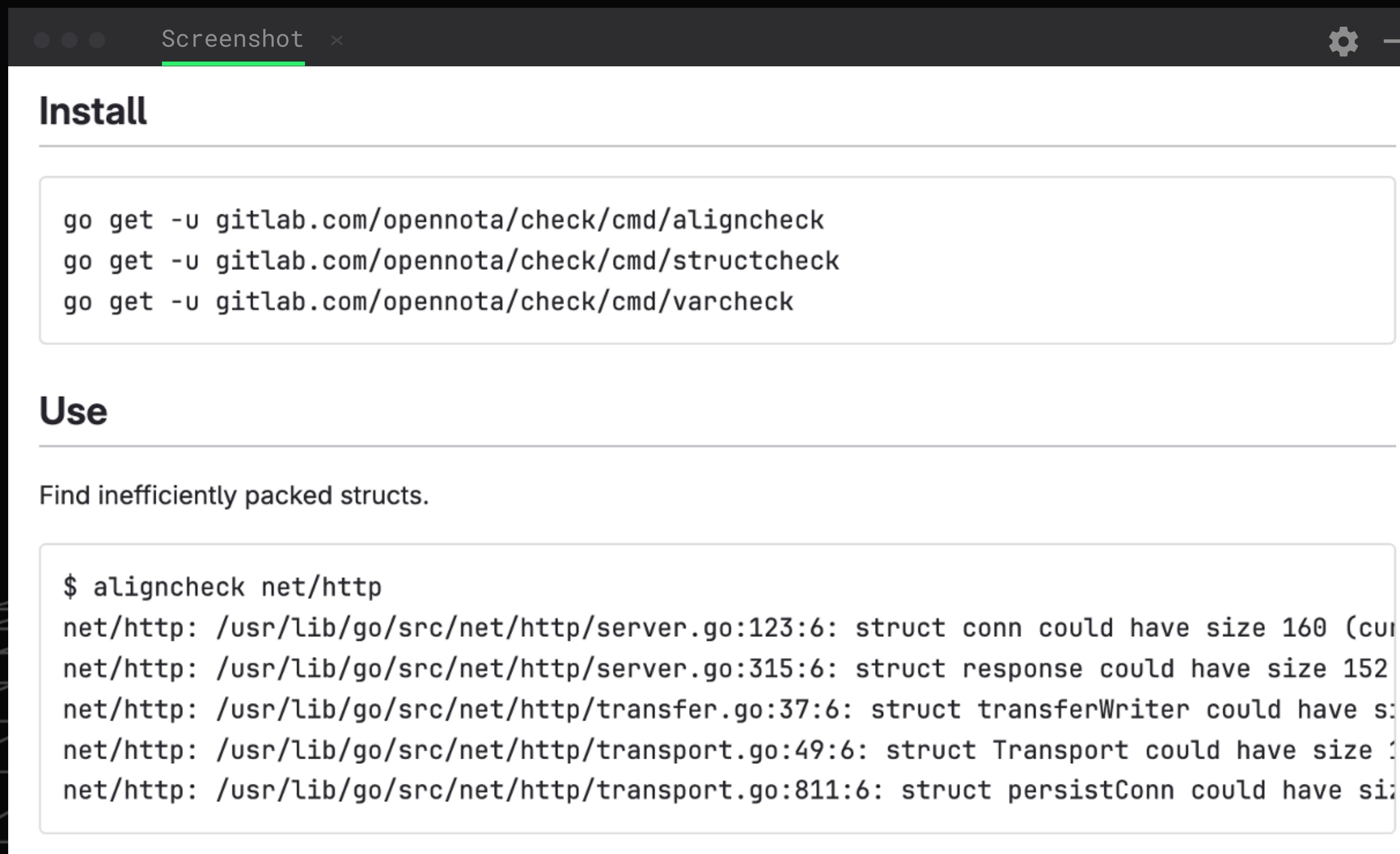
Struct alignment 2

Для массивов выравнивание идентично
выравниванию типа элемента массива

EXAMPLE

Alignment functions

ALIGNCHECK



Install

```
go get -u gitlab.com/opennota/check/cmd/aligncheck
go get -u gitlab.com/opennota/check/cmd/structcheck
go get -u gitlab.com/opennota/check/cmd/varcheck
```

Use

Find inefficiently packed structs.

```
$ aligncheck net/http
net/http: /usr/lib/go/src/net/http/server.go:123:6: struct conn could have size 160 (current 176)
net/http: /usr/lib/go/src/net/http/server.go:315:6: struct response could have size 152 (current 168)
net/http: /usr/lib/go/src/net/http/transfer.go:37:6: struct transferWriter could have size 128 (current 144)
net/http: /usr/lib/go/src/net/http/transport.go:49:6: struct Transport could have size 128 (current 144)
net/http: /usr/lib/go/src/net/http/transport.go:811:6: struct persistConn could have size 128 (current 144)
```

ВНИМАНИЕ!

НЕ СПЕШИТЕ!

Существует большой соблазн броситься и сразу начать править все свои структуры, чтобы они занимали как можно меньше места, но это темный путь, и не стоит становиться на эту дорожку раньше срока

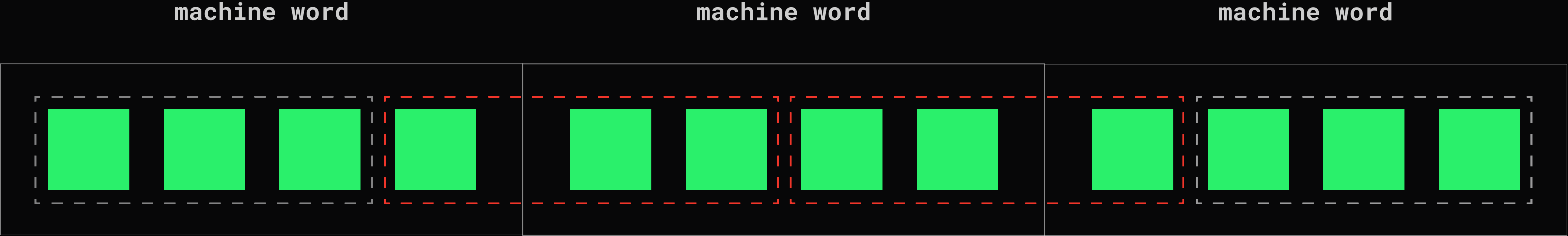
Почти всегда, **гораздо важней читаемость кода**, чем такие оптимизации. Важно понимать, по какой причине у значения типа именно такой размер и что вообще происходит, а уже в случае необходимости заниматься оптимизацией

ЗАЧЕМ НУЖНО ВЫРАВНИВАНИЕ?

Большинство современных процессоров читают данные из памяти не отдельными байтами, а блоками по 2-4-8 байт – соответственно, если у вас данные не выравнены в памяти и начало попадает в один блок, а конец в другой, то для чтения надо будет получить два таких блока

Представим, что нет
выравнивания и размер
структуры занимает 3 байта

```
1 type data struct {
2     aaa int16
3     bbb bool
4 }
5
6 func main() {
7     var values []data
8     for value := range values {
9         _ = value
10    }
11 }
```

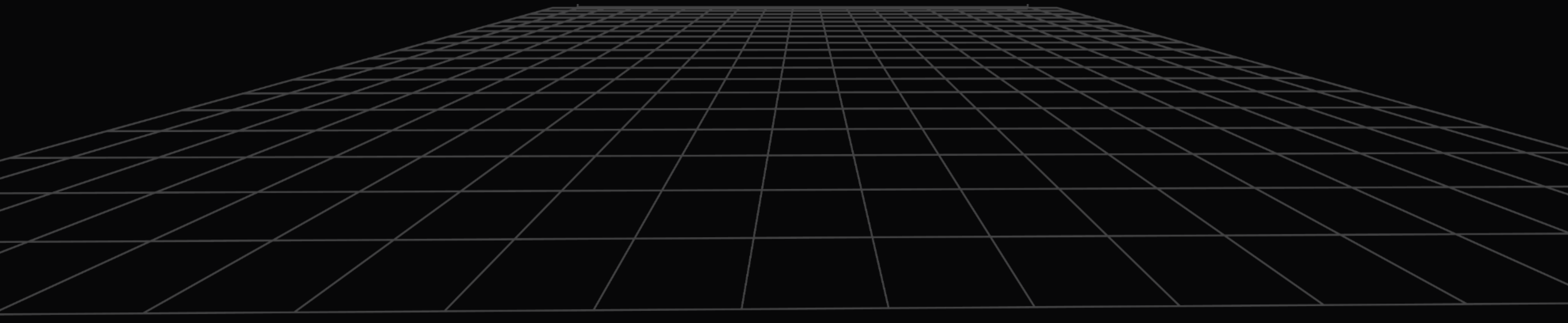


**КАКОЕ БУДЕТ ВЫРАВНИВАНИЕ У
СТРУКТУРЫ, СОСТОЯЩИХ ИЗ INT64
ПОЛЕЙ НА 32 БИТНОЙ МАШИНЕ?**

type	alignment guarantee
-----	-----
bool, uint8, int8	1
uint16, int16	2
uint32, int32	4
float32, complex64	4
arrays	depend on element types
structs	depend on field types
other types	size of a native word

FAQ

Выравнивание структур



ПЕРЕРЫВ 5 МИНУТ

ТОНКОСТИ СТРУКТУР

EXAMPLE

Empty struct size

**РАЗМЕР ПУСТОЙ
СТРУКТУРЫ РАВЕН НУЛЮ!**

EXAMPLE

Empty structs

EXAMPLE

Final zero field

ALL THE FIELDS OF AN ADDRESSABLE STRUCT VALUE CAN BE TAKEN ADDRESSES.

If the size of the final field in a non-zero-sized struct value is zero, then taking the address of the final field in the struct value will return an address which is beyond the allocated memory block for the struct value. The returned address may point to another allocated memory block which closely follows the one allocated for the non-zero-sized struct value. As long as the returned address is stored in an active pointer value, the other allocated memory block will not get garbage collected, which may cause memory leaking.

To avoid these kinds of memory leak problems, the standard Go compiler will ensure that taking the address of the final field in a non-zero-sized struct will never return an address which is beyond the allocated memory block for the struct.

The standard Go compiler implements this by padding some bytes after the final zero-sized field when needed. If the types of all fields in a struct type are zero-sized (so the struct is also a zero-sized type), then there is no need to pad bytes in the struct, for the standard Go compiler treats zero-sized memory blocks specially.

EXAMPLE

Empty structs internal

EXAMPLE

Defer calculating 1

EXAMPLE

Defer calculating 2

EXAMPLE

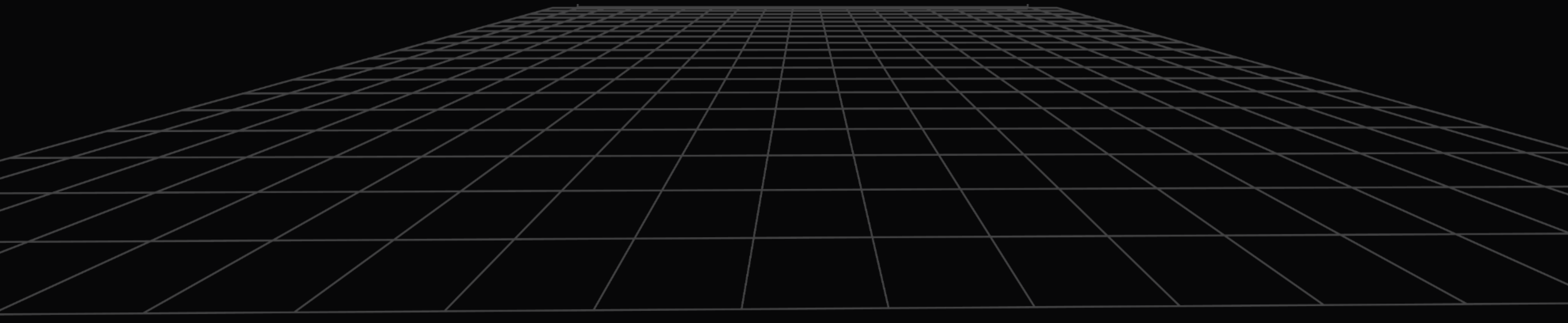
Comparison benchmark

EXAMPLE

Check struct size

FAQ

Тонкости структур



ПАТТЕРНЫ

Terminal: deep_golang × + ∨



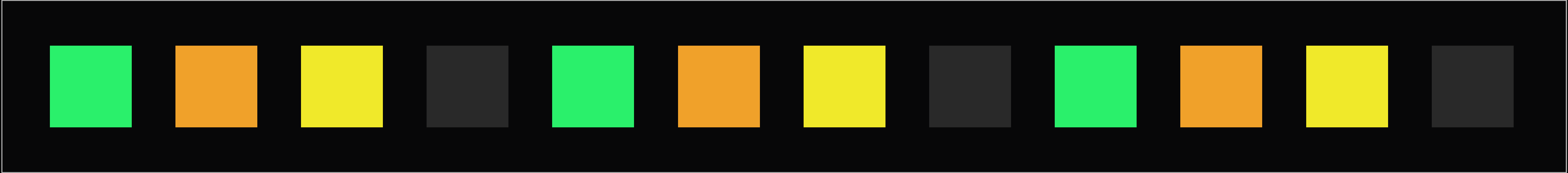
DOD (DATA ORIENTED DESIGN)

это способ оперировать данными
в cache friendly манере

EXAMPLE

DOD

Object Oriented Design



Data Oriented Design



CLOSER

**Очищать все ресурсы в одном месте программы
не всегда удобно или вообще возможно
сделать, не ухудшив архитектуру приложения**

Для решения этой проблемы существует отдельный паттерн. Пусть каждый компонент, заинтересованный в том, чтобы ему сообщили, когда необходимо очистить ресурсы и прекратить работу, сообщит об этом отдельному компоненту, зарегистрировав у него обработчик

EXAMPLE

Closer

Можно не писать свое,
а использовать готовую реализацию

The screenshot shows the Go Package Catalog interface for the 'closer' package. The top navigation bar includes the Go logo, a search bar, and links for 'Why Go', 'Learn', 'Docs', 'Packages' (active), and 'Community'. The breadcrumb trail is 'Discover Packages > github.com/xlab/closer'. The package name 'closer' is displayed with 'package' and 'module' tabs. Below this, the version 'v1.1.0' is marked as 'Latest', and other metadata like 'Published: Oct 12, 2022', 'License: MIT', 'Imports: 10', and 'Imported by: 209' are shown. A 'Details' section contains status checks: 'Valid go.mod file', 'Redistributable license', 'Tagged version', and 'Stable version', along with a link to 'Learn more about best practices'. The 'Repository' is 'github.com/xlab/closer' and 'Links' include 'Open Source Insights'. On the left, a sidebar has a 'Jump to ...' search and a list of sections: 'README' (selected), 'Usage', 'Table of exit codes', 'License', 'Documentation', and 'Source Files'. The main content area shows the 'README' for 'Closer', which includes a 'PASSED' badge, a 'reference' badge, and a description of the package's purpose. A code block at the bottom shows a terminal snippet: 'xlab ~/Documents/dev/go/src/github.com/xlab/closer \$ go run cmd/example/main.go 10 seconds to go... ^C^C^C^C Done'.

ScreenShot

GO

Search packages or symbols

Why Go ▾ Learn Docs ▾ Packages Community ▾

Discover Packages > github.com/xlab/closer

closer package module

Version: v1.1.0 Latest | Published: Oct 12, 2022 | License: MIT | Imports: 10 | Imported by: 209

Details Valid go.mod file Redistributable license Tagged version Stable version [Learn more about best practices](#)

Repository github.com/xlab/closer

Links [Open Source Insights](#)

Jump to ... f

README

Closer

The aim of this package is to provide an universal way to catch the event of application's exit and perform some actions before it's too late. `closer` doesn't care about the way application tries to exit, i.e. was that a panic or just a signal from the OS, it calls the provided methods for cleanup and that's the whole point.

```
xlab ~/Documents/dev/go/src/github.com/xlab/closer $ go run cmd/example/main.go
10 seconds to go...
^C^C^C^C Done
```

**КАК В GO ПРИНЯТО
КОНСТРУИРОВАТЬ ОБЪЕКТЫ
С ОПЦИОНАЛЬНЫМИ
АРГУМЕНТАМИ?**

EXAMPLE

Optional parameters

EXAMPLE

Functional options

FUNCTIONAL OPTIONS

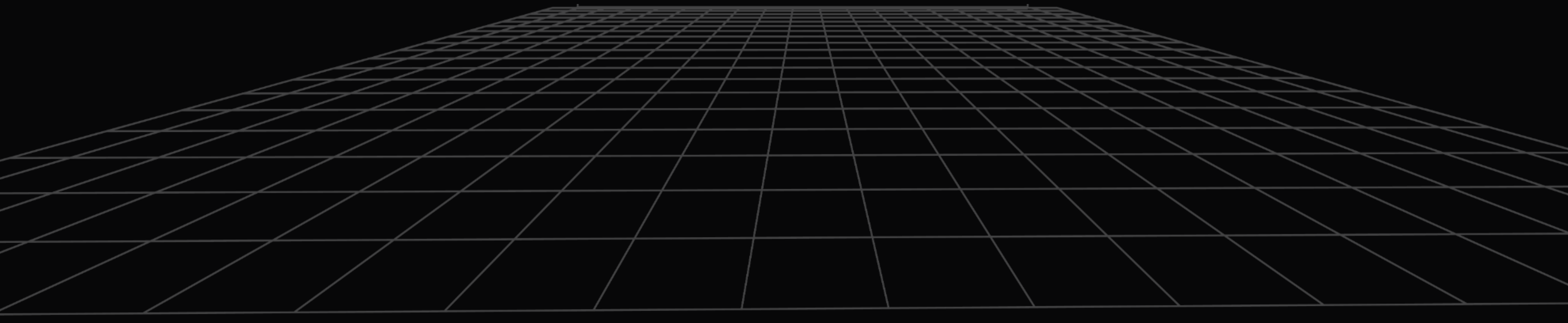
- Не надо рефакторить имеющиеся вызовы в случае изменений, они останутся без изменений
- Вызовы функций менее громоздкие, когда весь набор опций требуется очень редко, либо никогда
- Параметры становятся именованными и их становится проще читать

EXAMPLE

Configurable object

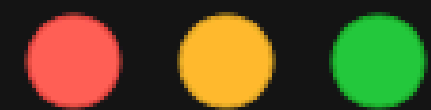
FAQ

Паттерны



ПСЕВДОНИМЫ И ОПРЕДЕЛЕНИЯ ТИПОВ

TYPE DEFENITION



```
1 type NewTypeName int
2
3 type (
4     NewTypeName1 string
5     NewTypeName2 NewTypeName
6 )
```

Новый определенный тип и соответствующий ему исходный тип в определениях типов – это два разных типа. Новый определенный тип и исходный тип будут иметь один и тот же базовый тип (underlying type)

TYPE ALIAS



```
1 type (  
2     Name = string  
3     Age  = int  
4 )  
5  
6 type Table = map[Name]Age
```

- **Type alias** – не создает новый тип, просто псевдоним типа
- **Type definition** – создает новый тип

EXAMPLE

Type alias and definition

ПРИ ПРИСВАИВАНИИ:

- **type alias** – ничего не нужно делать
- **type definition** – нужно кастить, потому что это разные типы данных

EXAMPLE

Type alias and definition with method

МОЖНО ОБЪЯВИТЬ

МЕТОД ДЛЯ ТИПА T, ЕСЛИ:

- тип T не является базовым типом данных и его алиасом
- тип T находится в том же пакете, где и метод
- тип T не должен быть указателем
- тип T не должен быть интерфейсом

EXAMPLE

Methods with types

EXAMPLE

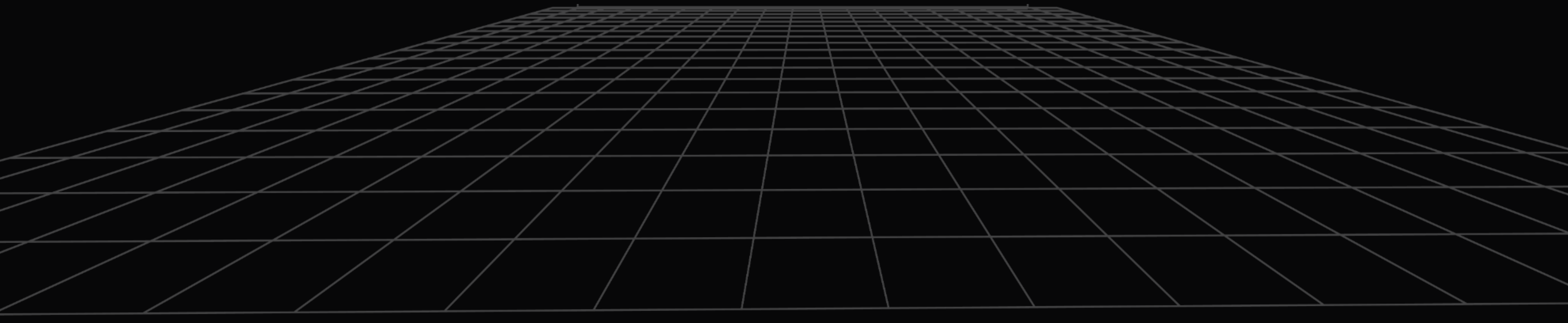
Methods for basic types

EXAMPLE

Unsafe cast

FAQ

Псевдонимы и определения типов



ПОЖАЛУЙСТА, ЗАПОЛНИ ОПРОС О ЗАНЯТИИ

Ссылка в чате и в группе участников

