Constants in Go

CONST PROPERTIES

- Cannot be redeclared/reassigned
- Can only hold scalar values
- Can hold large high precision numbers
- Can be created from expressions of other constants
- Can be untyped (kind)
- Constants exist only during compilation time

SCALAR DEF

A scalar variable, or scalar field, is a variable that holds one value at a time. It is a single component that assumes a range of number or string values. A scalar value is associated with every point in a space.

In computing, the term scalar is derived from the scalar processor, which processes one data item at a time

SYNTAX

```
// one line untyped
const identifier = 10

// one line typed
const identifier string = "Hello World!"

// multi line (group) mixed
const (
    // numbers
    num1 = 10
    num2 = 10.25
    num3 = 1 + 2i

    // strings
    str1 = "Hello"
    str2 = "World"

    // typed
    complexNum complex128 = 1 + 0i
)
```

IDENTIFIERS

An identifier is a sequence of one or more unicode letters and digits, the first character must be a unicode letter (including underscore)

```
const (
    identifier = 10
    _underscore = "underscore"
    hello_world = "hello_world"
    c1 = 1
    c2 = 2
    αβ = "Greek"
    汉字 = "Chinese"
)
```

RESERVED IDENTIFIERS

keywords

break	default	func	interface	select
case	defer	go	map	struct
chan	else	goto	package	switch
const	fallthrough	if	range	type
continue	for	import	return	var

operators & punctuation

+	&	+=	&=	&&	==	!=	(	)
-	|	-=	|=	||	<	<=	[	]
*	^	*=	^=	<-	>	>=	{	}
/	<<	/=	<<=	++	=	:=	,	;
%	>>	%=	>>=	--	!	...	.	:
			&^=					

BLANK IDENTIFIER

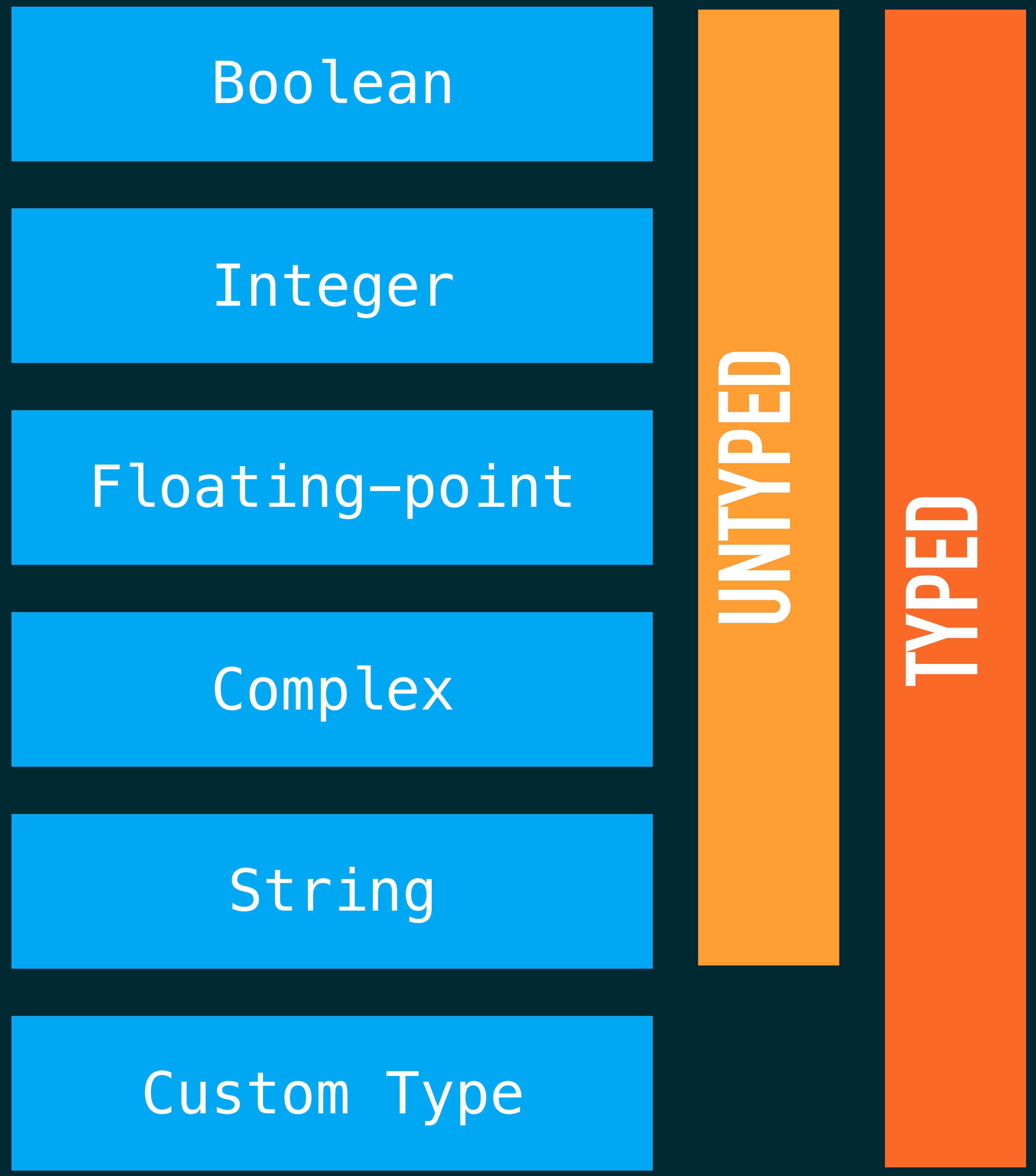
```go
import (
    // unused import
    _ "fmt"

    // side effects
    _ "github.com/go-sql-driver/mysql"
)


// unused identifier
const _ = iota


// unused (ignored) error
val, _ := SomeValAndErr()
```

Unused identifiers/imports

Unused errors

Side effects

GENERAL CONST TYPES



SPECIFIC CONST TYPES

bool	uint	float32	untyped bool
int	uint8/byte	float64	untyped int/iota
int8	uint16	complex64	untyped rune
int16	uint32	complex128	untyped float
int32/rune	uint64	string	untyped complex
int64	uintptr		untyped string
			custom type

IOTA DEF

Greek name of the ninth letter of the Greek alphabet, they spelled it as either *iota* or *jota*(the letters *i* and *j* were simply variants of each other), and these spellings eventually passed into English as *iota* and *jot*. Since the Greek letter iota is the smallest letter of its alphabet, both words eventually came to be used in reference to very small things

IOTA EXAMPLES

```
const (
    // first value ignored
    _ = iota
    KB = 1 << (iota * 10)
    MB
    GB
)


const (
    Sunday = iota + 1
    Monday
    Tuesday
    Wednesday
    Thursday
    Friday
    Saturday
)
```

iota is an alias of untyped int

CONST OPERATION RESTRICTIONS

- Can't mix and match types
- Be explicit about the end result

C BACKGROUND

C

```
unsigned int u = 1e9;
long signed int i = 1;
... i + u ...
```

✓

Go

```
const (
  u uint = 1e9;
  i int = 1;
)
... i + u ...
```

✓

In a binary operation Go works only with values of the same type

CONST VISIBILITY (SCOPE)

Exported

- Starts with uppercase letter
- Recommended to have explanatory comments aka docs

Unexported

- Starts with lowercase letter

UNTYPED CONST(S) - KIND

Except bool and string types, every other const type are just numbers and they are just numbers

Boolean space

String space

Number space

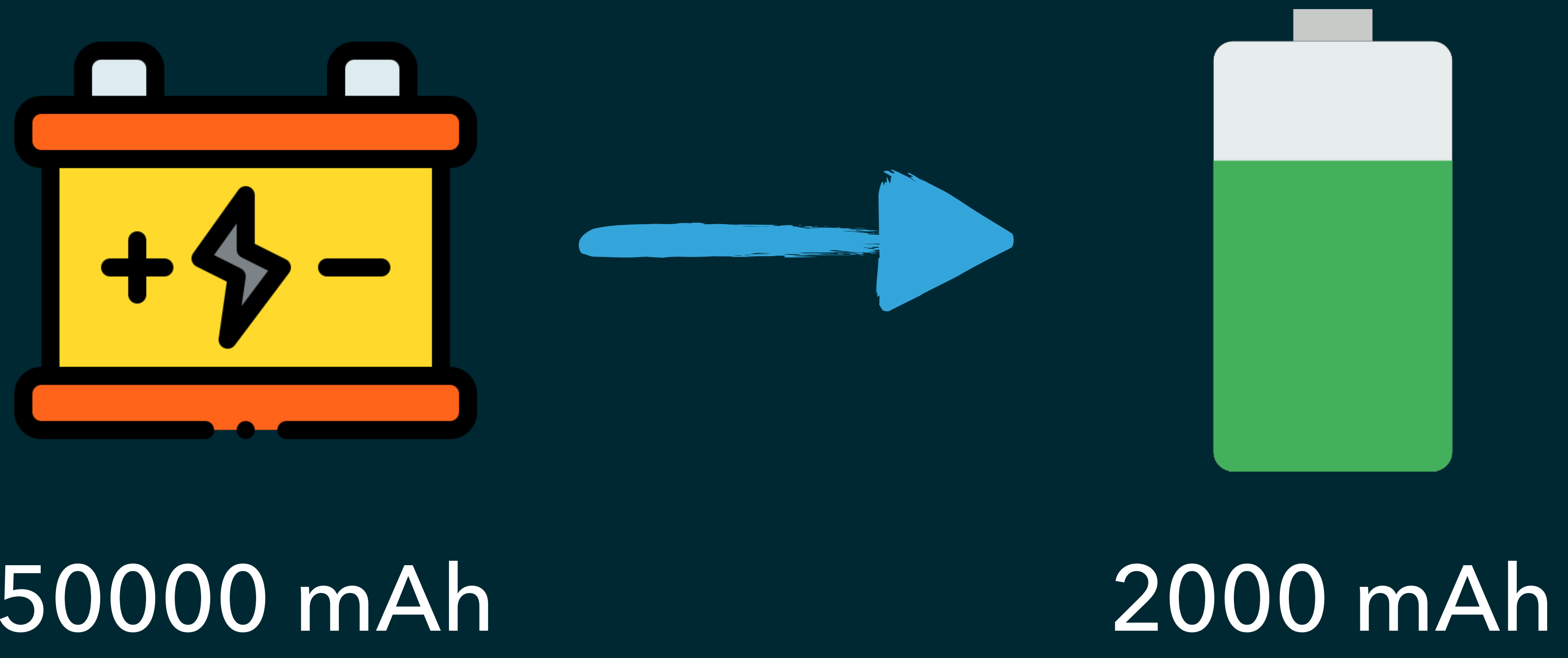
KIND DEFAULT TYPES

untyped bool	bool
untyped int	int
iota	int
untyped rune	int32
untyped float	float64
untyped complex	complex128
untyped string	string

NUMERIC VALUES

- Constants are mathematically exact values
- Range of values
- Signed / Unsigned
- Size

CONST OVERFLOW



Huge = 1.2676506002282294e+30 = 2^100

i64 = 2.147483648e+09 = 2^63

UNTYPED CONST HIGH PRECISION

Mathematically exact values

IMPLICIT TYPE CONVERSION

By syntax	Untyped constants
By type	Typed constants / variables

KIND PROMOTION

5)	Integer	K1	comparison	K2	untyped bool
4)	Floating-point	uint	shift	uint	untyped int
3)	Complex	K	operation (no shift)	K	untyped K
2)	String	K1	operation	K2	untyped const
1)	Custom type	K	operation	T	T

EXERCISE

```go
type T complex128

const (
   t T = 1;
   n = 2 + t * 'c' * 2.0 + 35i
   shift = 'a' << 4
)

func main() {
    fmt.Println("%T\n", n)
    fmt.Println("%T", shift)
}
```

```
main.T

int32
```

EXPRESSION CONVERSION

```
type T complex128

const (
  t T = 1;
  n = 2 + t * 'c' * 2.0 + 35i
)
```

1. Select the type with the highest priority
2. Convert all other kinds to the selected type
3. Apply the expression operations

Integer

Floating-point

Complex

String

Custom type

int ● T ● int32 ● float64 ● complex128