Go 1.27

@peterhellberg



Language

Generic methods

Method declarations may now declare their *own* type parameters, adding **generic functions** within the namespace of a particular type.

*Methods of **interfaces** may not declare type parameters, nor can interface methods be implemented by generic methods.*

Generic methods

```
type Tree[T any] struct {
    value       T
    left, right *Tree[T]
}

func (t *Tree[T]) Map[U any](f func(T) U) *Tree[U] {
    if t == nil {
        return nil
    }
    return &Tree[U]{f(t.value), t.left.Map(f), t.right.Map(f)}
}
```

Struct literal keys

A key in a struct literal may now be any **valid field selector** for the struct type.

*Previously, keys were limited to **top-level** field names only.*

Struct literal keys

```
type Base struct{ ID int }

type User struct {
    Base
    Name string
}

u := User{
    ID:   1, // promoted field – previously not allowed
    Name: "Go",
}
```

Function type inference

Function type inference has been **generalized** to apply in *all* contexts where a generic function is assigned to a variable of *(or converted to)* a matching function type.

Fewer explicit type arguments are needed when wiring up generic functions.

GO command



go fix

`go fix` gains new **modernizers**: `atomictypes`, `embedlit`, `slicesbackward`, and `unsafefuncs`.

The ***`fmtappendf`*** *analyzer was* ***removed*** (stylistic concerns), *and* ***`waitgroup`*** *was renamed to* ***`waitgroupgo`***.

go doc

`go doc` now supports the `package@version` syntax, e.g.

`go doc example.com/pkg@v1.2.3`.

*The new `-ex` option lists executable **examples** of a package or symbol; naming one prints its source code with comments.*

go mod tidy

For modules with `go 1.27` or later, `go mod tidy` now automatically **merges** duplicate require blocks.

*It ensures at most two blocks — one for **direct** and one for **indirect** dependencies — preserving existing comment blocks.*

go test

`go test` now runs the `stdversion` vet check by default, reporting use of standard library symbols that are *too new* for the Go version in force.

`go test -json` annotates `"Output"` lines with an optional `"OutputType"` field (“error”, “error-continue”, “frame”).

Tooling changes

A few smaller command and tool changes:

- *Response files (`@file`) are now supported by **`compile`**, **`link`**, **`asm`**, **`cgo`**, **`cover`**, and **`pack`***
- *The **`bzr`** version control system is no longer supported*
- ***Removed `GODEBUG`** settings in **`go.mod`** and **`//go:debug`** comments are accepted if set to their final default value* (otherwise it fails)
- ***`go tool trace -http=:6060`** now listens on **localhost** only*

Runtime



Faster memory allocation

The compiler generates **size-specialized** allocation routines, reducing the cost of small *(< 80 byte)* allocations by up to **30%**.

This yields a ~1% improvement in allocation-heavy programs, at the cost of ~60 KB of binary size.

Opt out with `GOEXPERIMENT=nosizespecializedmalloc` (expected to be removed in Go 1.28).

Goroutine leak profile

The profile reporting **leaked goroutines**, experimental in **Go 1.26**, is now *generally available*.

The **`goroutineleak`** profile is available in **`runtime/pprof`** and as the **`net/http/pprof`** endpoint **`/debug/pprof/goroutineleak`**.

*The **`goroutineleakprofile`** **`GOEXPERIMENT`** setting has been deleted.* (Credit: Vlad Saioc, Uber)

Goroutine leak profile

```
ch := make(chan result)
for _, w := range ws {
    go func() {
        res, err := processWorkItem(w)
        ch <- result{res, err}
    }()
}
for range len(ws) {
    r := <-ch
    if r.err != nil {
        return nil, r.err // This early return may leak goroutines.
    }
    results = append(results, r.res)
}
```

Tracebacks with labels

Modules with a `go 1.27` directive now include `runtime/pprof` goroutine **labels** in the traceback header line.

This can be disabled with the `tracebacklabels=0` ***GODEBUG*** *setting, an opt-out expected to be kept indefinitely.*

asynctimerchan removed

The `asynctimerchan` **GODEBUG** setting *(added in Go 1.23)* has been **permanently removed**.

Channels created by the ***`time`*** *package are now* ***always*** *unbuffered* (synchronous), *irrespective of* ***`GODEBUG`*** *settings.*

Also removed permanently: ***`tlsunsafeekm`***, ***`tlsrsakex`***, ***`tls3des`***, ***`tls10server`***, ***`x509keypairleaf`***, *and* ***`gotypesalias`***.

Compiler

Function literal names

The compiler generates **simpler** names for function literals *(closures)*, choosing the **same** name regardless of inlining.

Tests that check symbol names may need updates. Programs that incorrectly compare function code pointers may be more exposed.

Relative filenames in directives

`//line` and `/*line*/` directives now resolve relative filenames against the directory of the **containing file**.

This matches the behavior of ***`go/scanner`****. Absolute filenames are unaffected.*

Linker

macOS target support

The linker now accepts `-macos` and `-macsdk` options to specify the OS and SDK versions in the `LC_BUILD_VERSION` load command.

They default to the oldest supported macOS version (currently 13.0.0) *and a recent SDK version* (currently 26.2.0).

Standard library

New encoding/json/v2 package

`encoding/json/v2` is a **major revision** of `encoding/json`, with options-driven `Marshal` and `Unmarshal` families of functions.

It has *stricter* defaults — rejecting invalid UTF-8 and duplicate object names — and **`encoding/json`** is now backed by it.

*Unmarshaling is now **significantly faster**. The v1 API stays supported; opt out with `GOEXPERIMENT=nojsonv2` at build time.*

New encoding/json/jsontext package

`encoding/json/jsontext` provides **lower-level** syntactic processing of JSON.

*Its **`Encoder`** and **`Decoder`** operate on JSON as a sequence of **`Token`** and **`Value`**, maintaining a state machine for **valid** JSON text.*

New crypto/mldsa package

The new `crypto/mldsa` package implements the **post-quantum** ML-DSA signature scheme specified in *FIPS 204*.

`crypto/x509` *supports ML-DSA keys and signatures, and* ***`crypto/tls`*** *supports them in TLS 1.3* (`MLDSA44`, `MLDSA65`, `MLDSA87`).

New uuid package

The new **`uuid`** package generates and parses **UUIDs**.

New experimental simd package

The new, **experimental `simd`** package provides *portable*, vector-size-agnostic SIMD support, using hardware instructions if available.

It is available on all architectures with vector types such as **`Int8s`** and **`Float32s`**, enabled with **`GOEXPERIMENT=simd`**.

Updated experimental simd/archsimd

The architecture-specific `simd/archsimd` package, introduced in **Go 1.26**, revises its amd64 API and adds support for **arm64** *(Neon, 128-bit)* and **WebAssembly** *(128-bit)*.

The API is not yet stable and is non-portable by design. Enable it with ***`GOEXPERIMENT=simd`*** *at build time.*

bytes & strings: CutLast

The new `CutLast` functions in `bytes` and `strings` slice around the **last** occurrence of a separator.

```
before, after, found := strings.CutLast("a/b/c", "/")
// "a/b", "c", true
```

They can replace and simplify some uses of ***`LastIndex`****.*

net/http priorities

The HTTP/2 server now accepts client **priority signals** *(RFC 9218)*, serving higher-priority streams first.

HTTP/1 response bodies are now auto-drained on close *(up to a limit)*, improving connection reuse for most programs.

The old round-robin behavior returns via `Server.DisableClientPriority`*. The new* `Server.MaxHeaderValueCount` *caps accepted header values.*

hash/maphash & go/types hashing

`hash/maphash` gains a **`Hasher`** interface for hash-based structures, plus a **`ComparableHasher`** for **comparable** types.

`go/types` *gains a* ***`Hasher`*** *implementing it for* ***`Type`****s, respecting the* ***`Identical`*** *relation, so types can be used as map keys.* ***`HasherIgnoreTags`*** *is the analogue for* ***`IdenticalIgnoreTags`****.*

More minor additions

A selection of other handy additions:

- ***`math/big`***: ***`Int.Divide`*** *returns quotient and remainder with rounding modes*
- ***`math/rand/v2`***: ***`Rand`*** *gains the generic method* ***`N`***
- ***`net/url`***: *new* ***`URL.Clone`*** *and* ***`Values.Clone`*** *deep-copy methods*
- ***`net/http/httptest`***: ***`NewTestServer`*** *uses an in-memory fake network*
- ***`compress/flate`***: *faster compression — the encoded output may differ from Go 1.26*
- ***`crypto/x509`***: ***`SystemCertPool`*** *respects* ***`SSL_CERT_FILE`*** *and* ***`SSL_CERT_DIR`*** *on Windows and Darwin*
- ***`testing/synctest`***: *new* ***`Sleep`*** *helper;* ***`unicode`*** *upgraded to* ***Unicode 17***

As usual, there has been **minor** changes to a *number* of standard library packages, those packages are:

bytes compress/flate crypto crypto/ecdsa

crypto/tls crypto/x509 crypto/x509/pkix

database/sql database/sql/driver go/constant

go/scanner go/token go/types hash/maphash

math/big math/rand/v2 net net/http

net/http/httptest net/url runtime/secret strings

syscall testing/synctest unicode

Ports

Ports

- **Darwin**: Go 1.27 now requires *macOS 13 Ventura* or later, as announced in the Go 1.26 release notes.
- **PowerPC**: On big-endian 64-bit Linux (`GOARCH=ppc64`), the toolchain now generates binaries using the **ELFv2** ABI, gaining support for cgo, PIE, and external linking *(requires Linux kernel 3.13 or later)*.

Learn more

Please read the Go 1.27 Release Notes

https://go.dev/doc/go1.27