Go 1.26

@peterhellberg



Language

Enhanced new function

The built-in `new` function now allows its operand to be an **expression**, specifying the initial value of the variable.

This is especially useful for serialization packages such as ***`encoding/json`*** *or protocol buffers.*

Enhanced new function

```
type Person struct {
    Name string `json:"name"`
    Age  *int   `json:"age"` // age if known; nil otherwise
}

func personJSON(name string, born time.Time) ([]byte, error)
{
    return json.Marshal(Person{
        Name: name,
        Age:  new(yearsSince(born)),
    })
}
```

Self-referential generic types

The restriction preventing generic types from referring to themselves in their type parameter list has been **lifted**.

Generic types can now specify constraints that refer to themselves.

Self-referential generic types

```
type Adder[A Adder[A]] interface {
    Add(A) A
}

func algo[A Adder[A]](x, y A) A {
    return x.Add(y)
}
```

Previously, the self-reference to ***`Adder`*** *on the first line was not allowed.*

GO command



Revamped go fix

The `go fix` command has been **completely revamped** as the home of Go's modernizers.

*It provides a dependable way to update codebases to the latest idioms and APIs, with dozens of fixers built on the same analysis framework as `go vet`. Fixers should **not** change program behavior.*

Revamped go fix

A new **source-level inliner** allows users to automate API migrations using `//go:fix inline` directives.

The historical, obsolete fixers have been removed.

go mod init

`go mod init` now defaults to a lower **`go`** version in new **`go.mod`** files.

*Running it with toolchain version **`1.N.X`** creates a **`go.mod`** specifying **`go 1.(N-1).0`**. The **Go 1.26** release creates **`go.mod`** files with **`go 1.25.0`**.*

This encourages creating modules compatible with currently supported Go versions.

Removed tools

The `cmd/doc` tool and `go tool doc` have been **deleted**.

`go doc` serves as a replacement with the same flags, arguments, and behavior.

pprof

The `pprof` tool web UI *(enabled with the `-http` flag)* now defaults to the **flame graph** view.

The previous graph view is still available in the "View → Graph" menu or via `/ui/graph`.

Runtime



Green Tea garbage collector

The **Green Tea** garbage collector, previously experimental in **Go 1.25**, is now enabled by default.

It offers better locality and CPU scalability for marking and scanning small objects, with an expected **10–40%** reduction in garbage collection overhead in real-world programs.

It can be disabled by setting `GOEXPERIMENT=nogreenteagc` *at build time* (expected to be removed in Go 1.27).

Faster cgo calls

The baseline runtime overhead of **cgo** calls has been reduced by 30%.

Heap base address randomization

On 64-bit platforms, the runtime now **randomizes** the heap base address at startup for security.

This makes it harder for attackers to predict memory addresses and exploit vulnerabilities when using cgo.

Can be disabled by setting `GOEXPERIMENT=norandomizedheapbase64` *at build time.*

Goroutine leak profile

A new profile type reporting **leaked goroutines** is now available as an experiment.

A *leaked* goroutine is one blocked on a concurrency primitive *(channels, `sync.Mutex`, `sync.Cond`, ...)* that can **never** become unblocked. The runtime detects this using the garbage collector.

Enable the `goroutineleak` profile with `GOEXPERIMENT=goroutineleakprofile` at build time (goal: enabled by default in Go 1.27).

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)
}
```

Compiler

Stack allocation for slices

The compiler can now allocate the backing store for slices on the stack in more situations, which **improves** performance.

The bisect tool can be used with the `-compile=variablemake` flag to find problematic allocations, or all new stack allocations can be disabled with `-gcflags=all=-d=variablemakehash=n`.

Linker

Windows ARM64

On 64-bit ARM-based Windows *(the `windows/arm64` port)*, the linker now supports **internal linking** mode for cgo programs.

Enabled via `-ldflags=-linkmode=internal`.

Minor executable changes

Several changes affect programs analyzing executables, but not running programs:

- ***`moduledata`*** *is now in its own section, named* ***`.go.module`***
- *The* ***`.gopclntab`*** *section no longer contains relocations, and moves from relro to the rodata segment*
- *The* ***`.gosymtab`*** *section has been* ***removed***
- *ELF sections are now sorted by address in the section header list*

Standard library

New crypto/hpke package

The new `crypto/hpke` package implements **Hybrid Public Key Encryption** *(HPKE)* as specified in RFC 9180.

This includes support for post-quantum hybrid KEMs.

New experimental simd/archsimd package

The new, **experimental `simd/archsimd`** package provides architecture-specific SIMD operations, enabled by setting **`GOEXPERIMENT=simd`** at build time.

Currently available on amd64, it supports 128-, 256-, and 512-bit vector types *(**`Int8x16`**, **`Float64x8`**, ...)*. The API is **not** yet stable, and a high-level portable SIMD package is planned.

New experimental runtime/secret package

The new, **experimental `runtime/secret`** package helps securely erase temporaries in code manipulating secret information, typically cryptographic.

It securely erases registers, stack, and new heap allocations, easing forward secrecy. Enable it with `GOEXPERIMENT=runtimesecret` at build time *(amd64 and arm64 on Linux)*.

errors.AsType

The new `errors.AsType` function is a **generic** version of `errors.As`.

It is type-safe, faster, and easier to use in most cases.

reflect iterators

The `reflect` package gains new methods returning iterators:

- ***`Type.Fields`***, ***`Type.Methods`***, ***`Type.Ins`***, ***`Type.Outs`***
- ***`Value.Fields`***, ***`Value.Methods`***

Each iteration yields the type information (**`StructField`** or **`Method`**) *and the corresponding field or method* ***`Value`***.

log/slog MultiHandler

The new **`slog.NewMultiHandler`** function creates a **`MultiHandler`** that invokes all of the given Handlers.

`Enabled` *reports whether any handler is enabled;* ***`Handle`****,* ***`WithAttrs`****, and* ***`WithGroup`*** *call the corresponding method on each enabled handler.*

Faster io.ReadAll

`io.ReadAll` now allocates less intermediate memory and returns a minimally sized final slice.

*It is often about **2x faster** while allocating around half the total memory, with more benefit for larger inputs.*

Deterministic crypto randomness

Across the `crypto/...` packages, the `random` parameter to key and signature functions is now **ignored** — a secure cryptographic random source is always used.

Use the new ***`testing/cryptotest.SetGlobalRandom`*** *for deterministic testing. The* ***`GODEBUG=cryptocustomrand=1`*** *setting temporarily restores the old behavior.*

Minor changes to the library

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

bytes crypto crypto/dsa crypto/ecdh crypto/ecdsa
crypto/ed25519 crypto/fips140 crypto/mlkem
crypto/mlkem/mlkemtest crypto/rand crypto/rsa
crypto/sha3 crypto/subtle crypto/tls crypto/x509
debug/elf errors fmt go/ast go/token go/types
image/jpeg io log/slog net net/http net/http/httptest
net/http/httputil net/netip net/url os os/signal reflect
runtime/metrics testing testing/cryptotest time

Ports

Ports

- **Darwin**: Go 1.26 is the last release that runs on macOS 12 Monterey. Go 1.27 will require macOS 13 Ventura or later.
- **FreeBSD**: The freebsd/riscv64 port (`GOOS=freebsd GOARCH=riscv64`) is now marked broken.
- **Windows**: The broken 32-bit windows/arm port (`GOOS=windows GOARCH=arm`) has been removed.
- **PowerPC**: Go 1.26 is the last release supporting the ELFv1 ABI on big-endian 64-bit PowerPC Linux (`GOARCH=ppc64`).
- **RISC-V**: The `linux/riscv64` port now supports the race detector.
- **S390X**: Now supports passing function arguments and results using registers.
- **WebAssembly**: The runtime manages heap memory in smaller increments, reducing memory usage for heaps less than 16 MiB.

Learn more

Please read the Go 1.26 Release Notes

https://go.dev/doc/go1.26