Gleam Typed BEAM and JS language

CONCEPTS

Gleam compiles to Erlang (BEAM) or JavaScript. The target comes from gleam.toml, else Erlang, and --target overrides it.

src/*.gleam your modules
↓ `gleam build`
build/ BEAM files or JS
↓ `gleam run`
main() entry point
From source to running code

Tests live in test/ and gleam test runs main in <package>_test; dev code goes in dev/ with gleam dev.

Dependencies are in gleam.toml; manifest.toml is the resolved lock file, so commit it for applications.

PROJECTS

gleam new *<dir>*
Create a new package

gleam new *<dir>* **--skip-git**
Skip .git and .github

gleam run
Compile and run main

gleam run --target javascript
Run on the JS target

gleam run --runtime *<rt>*
JS runtime: nodejs, deno or bun

gleam shell
Erlang REPL with your code loaded

gleam clean
Delete the build directory

BUILD & CHECK

gleam build
Compile package and dependencies

gleam build --target *<t>*
Pick erlang or javascript

gleam build --warnings-as-errors
Fail on any warning

BUILD & CHECK (cont.)

gleam check
Type-check without code generation

gleam export erlang-shipment
Directory with bytecode and start script

TEST & FORMAT

gleam test
Run main of <package>_test

gleam test --target *<t>*
Test on one target

gleam format
Format the project

gleam format *<path>...*
Format given files or dirs

gleam format --check
Fail if a file is unformatted

DEPENDENCIES

gleam add *<pkg>*
Add latest compatible version

gleam add *<pkg>*@*<version>*
Pin a major or full version

gleam add --dev *<pkg>*
Add as a dev dependency

gleam remove *<pkg>*
Remove a dependency

gleam deps download
Fetch deps, write manifest.toml

gleam deps list
List all dependency versions

gleam deps outdated
Show dependencies behind latest

gleam deps update [*<pkg>***...]**
Upgrade within gleam.toml limits

gleam update
Alias for deps update

SYNTAX BASICS

```
let x = 1
```
Immutable binding

```
const limit = 10
```
Top-level constant

```
x |> f |> g
```
Pipe into the first argument

SYNTAX BASICS (cont.)

```
case v { 0 -> a  _ -> b }
```
Exhaustive pattern match

```
[first, ..rest]
```
List pattern or prepend

```
"pre" <> rest
```
String prefix pattern, concat

```
pub type T { A(n: Int)  B }
```
Custom type with variants

```
Person(..p, age: 30)
```
Record update

```
fn f(x: Int, add n: Int) -> Int
```
Labelled argument

```
use x <- result.try(f())
```
Flatten callbacks

```
Ok(v) / Error(e)
```
Result variants

```
Some(v) / None
```
Option, from gleam/option

```
import gleam/io.{println}
```
Import module and items

```
let assert Ok(v) = r
```
Crash if the pattern fails

```
todo as "msg"
```
Placeholder that crashes if run

```
assert a == b
```
Boolean check, used in tests

```
echo value
```
Print a debug value

PUBLISHING & DOCS

gleam docs build --open
Build HTML docs and open them

gleam publish
Publish the package to Hex

gleam hex authenticate
Log in to Hex

HELP

gleam help
List all commands

gleam help *<command>*
Help for one command

gleam --version
Print the compiler version