Mix Elixir build tool

START HERE

`mix new <name>`
Create a project

`iex -S mix`
Shell with project loaded

`mix deps.get`
Fetch dependencies

`mix run <file.exs>`
Run a script in the app

`mix compile`
Build the project

`mix format`
Format the code

`mix test`
Run the tests

`mix help`
List all tasks

mix.exs deps, aliases → `deps.get` → **mix.lock** pinned versions → `compile` → **_build/<env>** beams per MIX_ENV → `release` → **Release** self-contained

CONCEPTS

Mix is task-driven: mix <task> [args]. Run mix help to see every task, including those from deps.

MIX_ENV picks the environment: dev (default), test (used by mix test) or prod. Each gets its own _build directory.

Commit mix.lock for applications so every machine builds the same dependency versions.

PROJECTS

`mix new <name>`
Create a new project

`mix new <name> --sup`
Project with a supervision tree

`mix new <name> --umbrella`
Umbrella with several apps

`mix new <path> --app <app> --module <Module>`
Choose the app and module names

`mix phx.new <name>`
Phoenix project (needs the phx_new archive)
Install first: mix archive.install hex phx_new. Phoenix has its own mix phx. tasks; see mix help | grep phx.*

COMPILE & RUN

`mix compile`
Compile the project

`mix compile --force`
Recompile everything

`mix compile --warnings-as-errors`
Fail when there are warnings

`mix run <file.exs> [args]`
Run a script inside the app
Extra arguments are available through System.argv/0.

`mix run -e "<expr>"`
Evaluate an expression

`mix run --no-halt`
Keep the app running

`mix eval "<expr>"`
Evaluate without starting the app

`iex -S mix`
Interactive shell with the project

COMPILE & RUN (cont.)

`mix clean`
Remove compiled files

`mix do <task> + <task>`
Run several tasks in one call
The + separator needs spaces on both sides (Elixir 1.14+); older versions used a comma.

TEST

`mix test`
Run the whole suite

`mix test <file>`
Run one test file

`mix test <file>:<line>`
Run the test at a line
A line that starts a describe block runs everything in it.

`mix test --failed`
Re-run last failures

`mix test --stale`
Only tests for changed modules

`mix test --only <tag>`
Run tests with a tag

`mix test --exclude <tag>`
Skip tests with a tag

`mix test --trace`
One test at a time, verbose

`mix test --max-failures <n>`
Stop after n failures

`mix test --seed 0`
Run in definition order

`mix test --slowest <n>`
Show the n slowest tests

`mix test --cover`
Report code coverage

`mix test --repeat-until-failure <n>`
Repeat to find flaky tests

DEPENDENCIES

`mix deps.get`
Fetch and lock dependencies

`mix deps.update <dep>`
Update one dependency
Also updates the deps it depends on.

`mix deps.update --all`
Update every dependency

`mix deps`
List deps and their status

`mix deps.tree`
Show the dependency tree

DEPENDENCIES (cont.)

`mix deps.compile`
Compile dependencies only

`mix deps.clean <dep>`
Remove a dependency's build

`mix deps.clean --unused --unlock`
Drop deps no longer in mix.exs

`mix deps.unlock <dep>`
Remove a lock entry

Dependency forms in mix.exs

```
{:plug, "~> 1.16"}
{:ex_doc, "~> 0.34", only: :dev, runtime: false}
{:lib, github: "<user>/<repo>", tag: "<tag>"}
{:local, path: "../local"}
```

FORMAT & QUALITY

`mix format`
Format files per .formatter.exs

`mix format --check-formatted`
Fail if anything is unformatted

`mix format <file>`
Format specific files

`mix format --dry-run --check-formatted`
Check without writing

`mix hex.outdated`
Show deps with newer versions

`mix hex.audit`
Check deps for retired packages

RELEASES

`MIX_ENV=prod mix release`
Build a production release

`MIX_ENV=prod mix release <name>`
Build a named release

`mix release --overwrite`
Replace existing release files

`mix release.init`
Copy release config templates

`_build/prod/rel/<name>/bin/<name> start`
Run the release in the foreground

`bin/<name> daemon`
Run in the background (Unix)

`bin/<name> remote`
Open a shell on the running node

RELEASES (cont.)

`bin/<name> stop`
Stop the running release

ENVIRONMENTS & TOOLING

`MIX_ENV=<env> mix <task>`
Run a task in another env

`MIX_DEBUG=1 mix <task>`
Print task debug output

`mix local.hex`
Install or update the Hex client

`mix local.rebar`
Install rebar for Erlang deps

`mix archive.install hex <package>`
Install a global archive

`mix hex.info <package>`
Show package details

`mix hex.search <term>`
Search packages on Hex

`mix hex.publish`
Publish a package

Aliases in mix.exs

```
defp aliases do
  [setup: ["deps.get", "compile"],
   check: ["format --check-formatted", "test"]]
end
```

RECIPES

Fresh clone to green tests

```
mix deps.get
mix compile --warnings-as-errors
mix test
```

CI gate

```
mix format --check-formatted
mix compile --warnings-as-errors
mix test --cover
```

Fix a failing test fast

```
mix test --failed --trace
# once it passes, run all again
mix test
```

Ship a release

```
mix deps.get --only prod
MIX_ENV=prod mix release
_build/prod/rel/<name>/bin/<name> start
```

HELP

`mix help`
List available tasks

`mix help <task>`
Docs for one task

`mix --version`
Show Elixir and Mix versions