→ for quick ESLint
  → use @antfu/eslint-config
  → use the formatter

→ @hono/zod-openapi
  → wrapper around hono

→ create a not-found handler
  → app.notFound handler

→ [] created a stoker npm package with all those defaults

→ error handler → app.onError
  → consistent error handler

→ using pino as a logger
  - hono-pino
  - use pretty-print to view the log better
    - pino-pretty

→ pino stuff has alot of depreation
  → have alot of debugging to do

→ using types for Bindings
  → available as variables for Hono app

→ log level
  → using the appropriate log-level for the environment with ProcessEnv.LOG_LEVEL
  → info, warn, error etc.

→ dotenv and dotenv-expand
  → make sure to use type safety when using environment variables
  → limit where you use Process.env
  → look at env.ts
    → the app will not start until you have a valid environment variable file
  → look at t3.env

→ serve favicon
  - in stoker package

→ Create a function that creates the app
  → useful for testing separately
  → see lib/createApp

→ disable strict mode
  - /home > not the same thing
  - /home/

→ linking API version with package.json version field
  → import version from package.json

→ interactive documentation
  → Swagger UI - for interactivity
  → it sits on OpenAPI specification
  → scalar is another for interactivity
    → use @scalar/hono-api-reference
  → take OpenAPI course on linux academy

→ using stoker
  → to define default hook
  → to abstract jsonContent response of router

→ http-status-codes
  → but doesn't work well with hono

→ createMessageObjectSchema
  → helper for a route that only returns
    { message: " " }

→ using taos
  → group a bunch of routes together when showing in interaction

→ drizzle-orm

→ drizzle-zod
  → go from db schema to zod schema

→ use CROSS-ENV when working with windows

→ when writing test
  → you will Probably use @ts-expect-error because you will be testing bad paths

→ setup isolated db for test environment and env variables

→