Episode 11: Hex

Summary

- Install a Hex package
- Compile/use a Hex package
- Publish to hex

Hex

Hex is a package manager for the Erlang ecosystem.

Install: `mix local.hex`

Useful commands

```
mix deps.get
mix compile
```

Get dependencies

- Downloads packages from Hex/Git
- Caches Hex packages on your file system
- Places dependencies in your project's `deps/` folder

Remove a dependency Remove from `deps`, then run `mix deps.clean package_name`.

Update a dependency Change version in `mix.exs`, then run `mix deps.update package_name`.

Configuring a dependency Recall that we can configure our own apps in `mix.exs`. We can do the same for dependencies.

```
config :package_name, setting_a: "value",
                      setting_b: "value"
```

Searching for a package

- @h4cc's Awesome Elixir list
- Hex Web Search
- `mix hex.search PACKAGE_NAME`

Adding a package

Add `{:package_name, opts}` to the list returned by `deps()` in `mix.exs`.

You can also add packages from git repositories. Read up on this in the mix deps documentation. You will also find all of the options for adding packages, including `:only`, `:override`, and more.

A useful dependency option: `:only`

```
{:package_name, "~> 1.0.0", only: :dev}
{:package_name, "~> 1.0.0", only: [:dev, :test]}
```

The `:only` keyword indicates that a dependency should only be used in a particular environment.

Specifying a dependency

> Version parsing and requirements follow SemVer 2.0 schema.

A version on its own (e.g., "1.0.0") means to only install that version.

There are several operators that precede a version, including >, >=, <, <=, == and `!=`. Requirements also support `and` and `or` for complex conditions.

There is one more operator, ~>, with a special meaning.

Here are some more examples from the Version docs:

~>	Translation
`~> 2.0.0`	`>= 2.0.0 and < 2.1.0`
`~> 2.1.2`	`>= 2.1.2 and < 2.2.0`
`~> 2.1.3-dev`	`>= 2.1.3-dev and < 2.2.0`

~>	Translation
`~> 2.0`	`>= 2.0.0 and < 3.0.0`
`~> 2.1`	`>= 2.1.0 and < 3.0.0`

Publishing to Hex

> *Quoted from Episode 11: Hex Shownotes.*

1. Ensure project has a unique name
2. Add `package/0` to your Mixfile
3. Include all dependency app names in the `application/0` list.
4. Update `project/0` with `:source_url`, `:description` and `:package`.

Sample `package/0` function:

```
defp package do
  [
    files: ["lib", "mix.exs", "README.md"],
    maintainers: ["Your Name"],
    licenses: ["MIT"], # or whatever license you want
    links: %{
      "Github" => "https://github.com/username/repo"
    }
  ]
end
```

Sample `application/0` function:

```
def application do
  [applications: [:dependency_a, :dependency_b]]
end
```

Sample `project/0` function: (with lines you need to add highlighted)

```
 def project do
   [app: :my_project_name,
    version: "0.0.1",
    elixir: "~> 1.0",
```

```
    build_embedded: Mix.env == :prod,
    start_permanent: Mix.env == :prod,
+   source_url: "https://github.com/username/repo",
+   description: "Short description of project here",
+   package: package,
    deps: deps]
 end
```

To publish, run `mix hex.publish`.