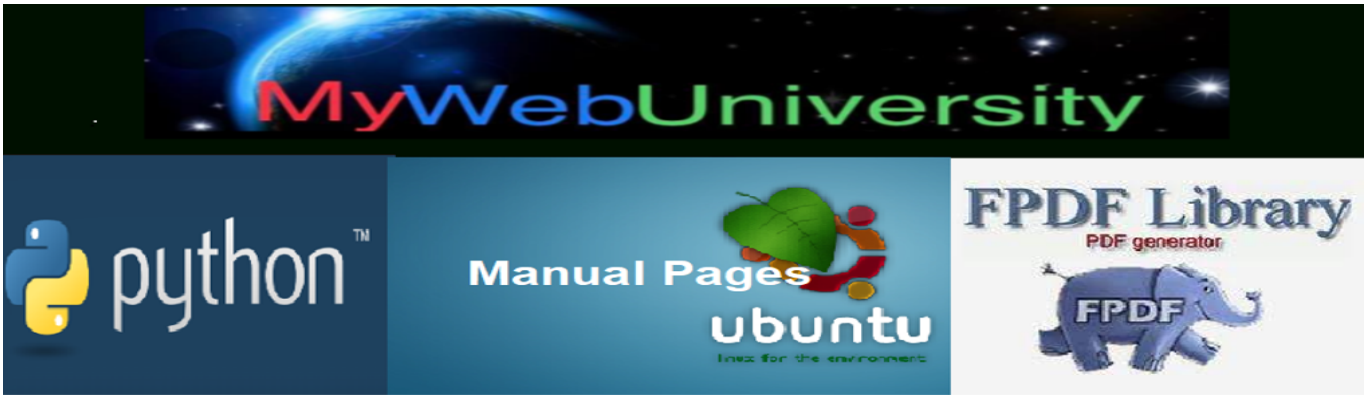

Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'npx.1'

$ man npx.1

NPX(1) NPX(1)

NAME

npx - Run a command from a local or remote npm package

Synopsis

npm exec -- <pkg>[@<version>] [args...]

npm exec --package=<pkg>[@<version>] -- <cmd> [args...]

npm exec -c '<cmd> [args...]'

npm exec --package=foo -c '<cmd> [args...]'

npx <pkg>[@<specifier>] [args...]

npx -p <pkg>[@<specifier>] <cmd> [args...]

npx -c '<cmd> [args...]'

npx -p <pkg>[@<specifier>] -c '<cmd> [args...]'

alias: npm x, npx

--package=<pkg> (may be specified multiple times)

-p is a shorthand for --package only when using npx executable

-c <cmd> --call=<cmd> (may not be mixed with positional arguments)

Description

This command allows you to run an arbitrary command from an npm package (either one in? stalled locally, or fetched remotely), in a similar context as running it via npm run.

Whatever packages are specified by the --package option will be provided in the PATH of the executed command, along with any locally installed package executables. The --package option may be specified multiple times, to execute the supplied command in an environment where all specified packages are available.

If any requested packages are not present in the local project dependencies, then they are installed to a folder in the npm cache, which is added to the PATH environment variable in the executed process. A prompt is printed (which can be suppressed by providing either --yes or --no).

Package names provided without a specifier will be matched with whatever version exists in the local project. Package names with a specifier will only be considered a match if they have the exact same name and version as the local dependency.

If no -c or --call option is provided, then the positional arguments are used to generate the command string. If no --package options are provided, then npm will attempt to deter? mine the executable name from the package specifier provided as the first positional argu? ment according to the following heuristic:

? If the package has a single entry in its bin field in package.json, or if all entries are aliases of the same command, then that command will be used.

? If the package has multiple bin entries, and one of them matches the unscoped portion of the name field, then that command will be used.

? If this does not result in exactly one option (either because there are no bin entries, or none of them match the name of the package), then npm exec exits with an error.

To run a binary other than the named binary, specify one or more --package options, which will prevent npm from inferring the package from the first command argument.

npx vs npm exec

When run via the npx binary, all flags and options must be set prior to any positional ar? guments. When run via npm exec, a double-hyphen -- flag can be used to suppress npm's parsing of switches and options that should be sent to the executed command.

For example:

```
$ npx foo@latest bar --package=@npmcli/foo
```

In this case, npm will resolve the foo package name, and run the following command:

```
$ foo bar --package=@npmcli/foo
```

Since the --package option comes after the positional arguments, it is treated as an argu? ment to the executed command.

In contrast, due to npm's argument parsing logic, running this command is different:

```
$ npm exec foo@latest bar --package=@npmcli/foo
```

In this case, npm will parse the --package option first, resolving the @npmcli/foo pack? age. Then, it will execute the following command in that context:

$ foo@latest bar

The double-hyphen character is recommended to explicitly tell npm to stop parsing command line options and switches. The following command would thus be equivalent to the npx com? mand above:

$ npm exec -- foo@latest bar --package=@npmcli/foo

Examples

Run the version of tap in the local dependencies, with the provided arguments:

$ npm exec -- tap --bail test/foo.js

$ npx tap --bail test/foo.js

Run a command other than the command whose name matches the package name by specifying a --package option:

$ npm exec --package=foo -- bar --bar-argument

~ or ~

$ npx --package=foo bar --bar-argument

Run an arbitrary shell script, in the context of the current project:

$ npm x -c 'eslint && say "hooray, lint passed"'

$ npx -c 'eslint && say "hooray, lint passed"'

Compatibility with Older npx Versions

The npx binary was rewritten in npm v7.0.0, and the standalone npx package deprecated at that time. npx uses the npm exec command instead of a separate argument parser and in? stall process, with some affordances to maintain backwards compatibility with the argu? ments it accepted in previous versions.

This resulted in some shifts in its functionality:

? Any npm config value may be provided.

? To prevent security and user-experience problems from mistyping package names, npx prompts before installing anything. Suppress this prompt with the -y or --yes option.

? The --no-install option is deprecated, and will be converted to --no.

? Shell fallback functionality is removed, as it is not advisable.

? The -p argument is a shorthand for --parseable in npm, but shorthand for --package in npx. This is maintained, but only for the npx executable.

? The --ignore-existing option is removed. Locally installed bins are always present in the executed process PATH.

? The --npm option is removed. npx will always use the npm it ships with.

? The --node-arg and -n options are removed.

? The --always-spawn option is redundant, and thus removed.

? The --shell option is replaced with --script-shell, but maintained in the npx executable for backwards compatibility.

See Also

? npm help run-script

? npm help scripts

? npm help test

? npm help start

? npm help restart

? npm help stop

? npm help config

? npm help exec

undefined NaN NPX(1)