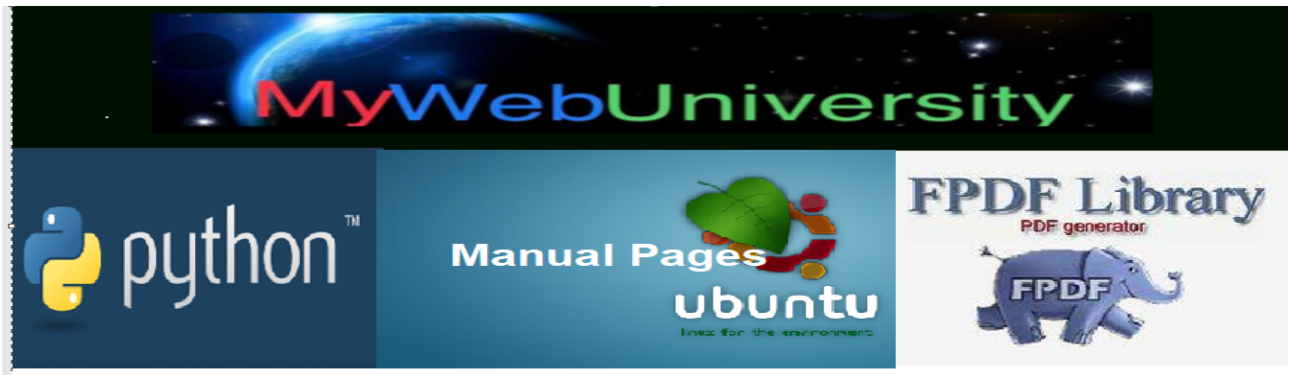

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'npm-run-script.1'

$ man npm-run-script.1

NPM-RUN-SCRIPT(1) General Commands Manual NPM-RUN-SCRIPT(1)

NAME

npm-run-script

Synopsis

<!-- AUTOGENERATED USAGE DESCRIPTIONS -->

Description

This runs an arbitrary command from a package's "scripts" object. If no

"command" is provided, it will list the available scripts.

run[-script] is used by the test, start, restart, and stop commands, but

can be called directly, as well. When the scripts in the package are

printed out, they're separated into lifecycle (test, start, restart) and

directly-run scripts.

Any positional arguments are passed to the specified script. Use -- to

pass --prefixed flags and options which would otherwise be parsed by npm.

For example:

npm run test -- --grep="pattern"

The arguments will only be passed to the script specified after npm run

and not to any pre or post script.

The env script is a special built-in command that can be used to list

environment variables that will be available to the script at runtime. If an

"env" command is defined in your package, it will take precedence over the

built-in.

In addition to the shell's pre-existing PATH, npm run adds

node_modules/.bin to the PATH provided to scripts. Any binaries
provided by locally-installed dependencies can be used without the
node_modules/.bin prefix. For example, if there is a devDependency on
tap in your package, you should write:

```
"scripts": {"test": "tap test/*.js"}
```

instead of

```
"scripts": {"test": "node_modules/.bin/tap test/*.js"}
```

The actual shell your script is run within is platform dependent. By default,
on Unix-like systems it is the /bin/sh command, on Windows it is
cmd.exe.

The actual shell referred to by /bin/sh also depends on the system.
You can customize the shell with the
script-shell config.

Scripts are run from the root of the package folder, regardless of what the
current working directory is when npm run is called. If you want your
script to use different behavior based on what subdirectory you're in, you
can use the INIT_CWD environment variable, which holds the full path you
were in when you ran npm run.

npm run sets the NODE environment variable to the node executable
with which npm is executed.

If you try to run a script without having a node_modules directory and it
fails, you will be given a warning to run npm install, just in case you've
forgotten.

Workspaces support

You may use the workspace or
workspaces configs in order to run an
arbitrary command from a package's "scripts" object in the context of the
specified workspaces. If no "command" is provided, it will list the available
scripts for each of these configured workspaces.

Given a project with configured workspaces, e.g:

```
.
+-- package.json
`-- packages
```

```
+-- a
|   `-- package.json
+-- b
|   `-- package.json
`-- c
    `-- package.json
```

Assuming the workspace configuration is properly set up at the root level package.json file. e.g:

```
{
  "workspaces": [ "./packages/*" ]
}
```

And that each of the configured workspaces has a configured test script, we can run tests in all of them using the workspaces config:

```
npm test --workspaces
```

Filtering workspaces

It's also possible to run a script in a single workspace using the workspace config along with a name or directory path:

```
npm test --workspace=a
```

The workspace config can also be specified multiple times in order to run a specific script in the context of multiple workspaces. When defining values for the workspace config in the command line, it also possible to use -w as a shorthand, e.g:

```
npm test -w a -w b
```

This last command will run test in both ./packages/a and ./packages/b packages.

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->

See Also

- ? npm scripts
- ? npm test
- ? npm start
- ? npm restart

? npm stop

? npm config

? npm workspaces