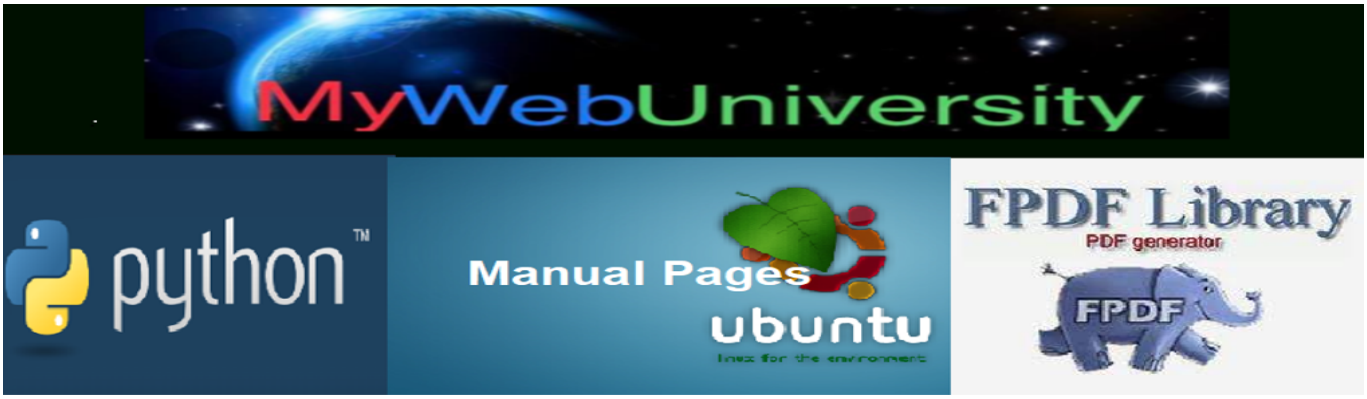

Rocky Enterprise Linux 9.2 Manual Pages on command 'npm-diff.1'

$ man npm-diff.1

NPM-DIFF(1) NPM-DIFF(1)

NAME

npm-diff - The registry diff command

Synopsis

npm diff [...<paths>]

npm diff --diff=<pkg-name> [...<paths>]

npm diff --diff=<version-a> [--diff=<version-b>] [...<paths>]

npm diff --diff=<spec-a> [--diff=<spec-b>] [...<paths>]

npm diff [--diff-ignore-all-space] [--diff-name-only] [...<paths>]

Description

Similar to its git diff counterpart, this command will print diff patches of files for packages published to the npm registry.

? npm diff --diff=<spec-a> --diff=<spec-b>

Compares two package versions using their registry specifiers, e.g: npm diff --diff=pkg@1.0.0 --diff=pkg@^2.0.0. It's also possible to compare across forks of any package, e.g: npm diff --diff=pkg@1.0.0 --diff=pkg-fork@1.0.0. Any valid spec can be used, so that it's also possible to compare directories or git repositories, e.g: npm diff --diff=pkg@latest --diff=./packages/pkg Here's an example comparing two different versions of a package named abbrev from the registry: npm diff --diff=abbrev@1.1.0 --diff=abbrev@1.1.1

On success, output looks like:

```
diff --git a/package.json b/package.json
index v1.1.0..v1.1.1 100644
--- a/package.json
+++ b/package.json
@@ -1,6 +1,6 @@
 {
   "name": "abbrev",
-  "version": "1.1.0",
+  "version": "1.1.1",
   "description": "Like ruby's abbrev module, but in js",
   "author": "Isaac Z. Schlueter <i@izs.me>",
   "main": "abbrev.js",
```

Given the flexible nature of npm specs, you can also target local directories or git repos just like when using npm install:

```
npm diff --diff=https://github.com/npm/libnpmdiff --diff=./local-path
```

In the example above we can compare the contents from the package installed from the git repo at github.com/npm/libnpmdiff with the contents of the ./local-path that contains a valid package, such as a modified copy of the original.

? npm diff (in a package directory, no arguments):

If the package is published to the registry, npm diff will fetch the tarball version tagged as latest (this value can be configured using the tag option) and proceed to compare the contents of files present in that tarball, with the current files in your local file system.

This workflow provides a handy way for package authors to see what package-tracked files have been changed in comparison with the latest published version of that package.

? npm diff --diff=<pkg-name> (in a package directory):

When using a single package name (with no version or tag specifier) as an argument, npm diff will work in a similar way to npm-outdated npm-outdated and reach for the registry to figure out what current published version of the package named <pkg-name>

will satisfy its dependent declared semver-range. Once that specific
version is known npm diff will print diff patches comparing the
current version of <pkg-name> found in the local file system with
that specific version returned by the registry.
Given a package named abbrev that is currently installed:
npm diff --diff=abbrev
That will request from the registry its most up to date version and
will print a diff output comparing the currently installed version to this
newer one if the version numbers are not the same.
? npm diff --diff=<spec-a> (in a package directory):
Similar to using only a single package name, it's also possible to declare
a full registry specifier version if you wish to compare the local version
of an installed package with the specific version/tag/semver-range provided
in <spec-a>.
An example: assuming pkg@1.0.0 is installed in the current node_modules
folder, running:
npm diff --diff=pkg@2.0.0
It will effectively be an alias to
npm diff --diff=pkg@1.0.0 --diff=pkg@2.0.0.
? npm diff --diff=<semver-a> [--diff=<semver-b>] (in a package directory):
Using npm diff along with semver-valid version numbers is a shorthand
to compare different versions of the current package.
It needs to be run from a package directory, such that for a package named
pkg running npm diff --diff=1.0.0 --diff=1.0.1 is the same as running
npm diff --diff=pkg@1.0.0 --diff=pkg@1.0.1.
If only a single argument <version-a> is provided, then the current local
file system is going to be compared against that version.
Here's an example comparing two specific versions (published to the
configured registry) of the current project directory:
npm diff --diff=1.0.0 --diff=1.1.0
Note that tag names are not valid --diff argument values, if you wish to compare to a pub?
lished tag, you must use the pkg@tagname syntax.

It's possible to also specify positional arguments using file names or globs pattern matching in order to limit the result of diff patches to only a subset of files for a given package, e.g:

```
npm diff --diff=pkg@2 ./lib/ CHANGELOG.md
```

In the example above the diff output is only going to print contents of files located within the folder ./lib/ and changed lines of code within the CHANGELOG.md file.

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS START --> <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

diff

? Default:

? Type: String (can be set multiple times)

Define arguments to compare in npm diff. <!-- automatically generated, do not edit manu? ally --> <!-- see lib/utils/config/definitions.js -->

diff-name-only

? Default: false

? Type: Boolean

Prints only filenames when using npm diff. <!-- automatically generated, do not edit man? ually --> <!-- see lib/utils/config/definitions.js -->

diff-unified

? Default: 3

? Type: Number

The number of lines of context to print in npm diff. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

diff-ignore-all-space

? Default: false

? Type: Boolean

Ignore whitespace when comparing lines in npm diff. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

diff-no-prefix

? Default: false

? Type: Boolean

Do not show any source or destination prefix in npm diff output.

Note: this causes npm diff to ignore the --diff-src-prefix and --diff-dst-prefix configs.

<!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/defini?

tions.js -->

diff-src-prefix

? Default: "a/"

? Type: String

Source prefix to be used in npm diff output. <!-- automatically generated, do not edit

manually --> <!-- see lib/utils/config/definitions.js -->

diff-dst-prefix

? Default: "b/"

? Type: String

Destination prefix to be used in npm diff output. <!-- automatically generated, do not

edit manually --> <!-- see lib/utils/config/definitions.js -->

diff-text

? Default: false

? Type: Boolean

Treat all files as text in npm diff. <!-- automatically generated, do not edit manually

--> <!-- see lib/utils/config/definitions.js -->

global

? Default: false

? Type: Boolean

Operates in "global" mode, so that packages are installed into the prefix folder instead

of the current working directory. See npm help folders for more on the differences in be?

havior.

? packages are installed into the {prefix}/lib/node_modules folder, instead of the current

working directory.

? bin files are linked to {prefix}/bin

? man pages are linked to {prefix}/share/man

<!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/defini?

tions.js -->

tag

? Default: "latest"

? Type: String

If you ask npm to install a package and don't tell it a specific version, then it will in?

stall the specified tag.

Also the tag that is added to the package@version specified by the npm tag command, if no explicit tag is given.

When used by the npm diff command, this is the tag used to fetch the tarball that will be compared with the local files by default. <!-- automatically generated, do not edit manu?

ally --> <!-- see lib/utils/config/definitions.js -->

workspace

? Default:

? Type: String (can be set multiple times)

Enable running a command in the context of the configured workspaces of the current project while filtering by running only the workspaces defined by this configuration op?

tion.

Valid values for the workspace config are either:

? Workspace names

? Path to a workspace directory

? Path to a parent workspace directory (will result in selecting all workspaces within that folder)

When set for the npm init command, this may be set to the folder of a workspace which does not yet exist, to create the folder and set it up as a brand new workspace within the project.

This value is not exported to the environment for child processes. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

workspaces

? Default: null

? Type: null or Boolean

Set to true to run the command in the context of all configured workspaces.

Explicitly setting this to false will cause commands like install to ignore workspaces al?

together. When not set explicitly:

? Commands that operate on the node_modules tree (install, update, etc.) will link workspaces into the node_modules folder. - Commands that do other things (test, exec, publish, etc.) will operate on the root project, unless one or more workspaces are spec?

ified in the workspace config.

This value is not exported to the environment for child processes. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

include-workspace-root

? Default: false

? Type: Boolean

Include the workspace root when workspaces are enabled for a command.

When false, specifying individual workspaces via the workspace config, or all workspaces via the workspaces flag, will cause npm to operate only on the specified workspaces, and not on the root project. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

<!-- AUTOGENERATED CONFIG DESCRIPTIONS END -->

See Also

? npm help outdated

? npm help install

? npm help config

? npm help registry