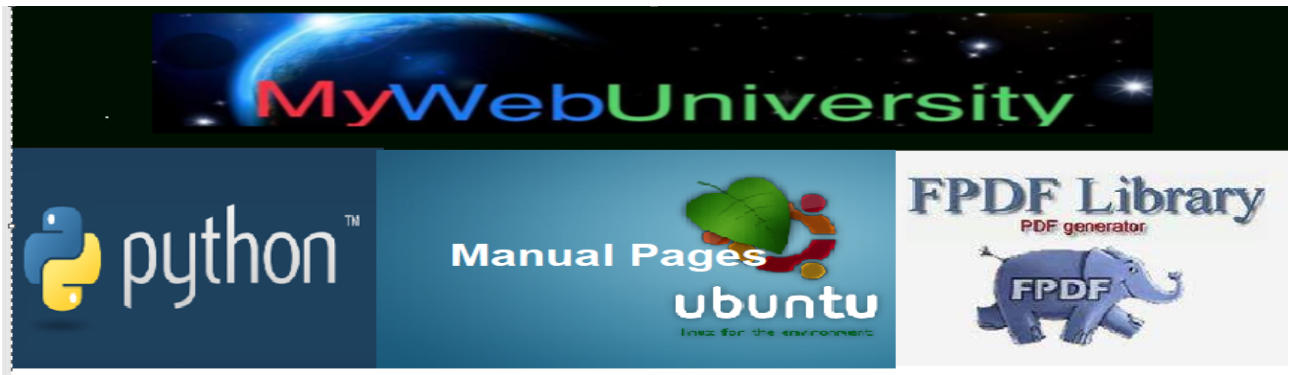

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man npm-pkg.1

NPM-PKG(1) General Commands Manual NPM-PKG(1)

NAME

npm-pkg

Synopsis

<!-- AUTOGENERATED USAGE DESCRIPTIONS -->

Description

A command that automates the management of package.json files.

npm pkg provide 3 different sub commands that allow you to modify or retrieve values for given object keys in your package.json.

The syntax to retrieve and set fields is a dot separated representation of the nested object properties to be found within your package.json, it's the same notation used in npm view to retrieve information from the registry manifest, below you can find more examples on how to use it.

Returned values are always in json format.

? npm pkg get <field>

Retrieves a value key, defined in your package.json file.

For example, in order to retrieve the name of the current package, you can run:

npm pkg get name

It's also possible to retrieve multiple values at once:

npm pkg get name version

You can view child fields by separating them with a period. To retrieve the value of a test script value, you would run the following command:

```
npm pkg get scripts.test
```

For fields that are arrays, requesting a non-numeric field will return all of the values from the objects in the list. For example, to get all the contributor emails for a package, you would run:

```
npm pkg get contributors.email
```

You may also use numeric indices in square braces to specifically select an item in an array field. To just get the email address of the first contributor in the list, you can run:

```
npm pkg get contributors[0].email
```

For complex fields you can also name a property in square brackets to specifically select a child field. This is especially helpful with the exports object:

```
npm pkg get "exports[.].require"
```

? npm pkg set <field>=<value>

Sets a value in your package.json based on the field value. When saving to your package.json file the same set of rules used during npm install and other cli commands that touches the package.json file are used, making sure to respect the existing indentation and possibly applying some validation prior to saving values to the file.

The same syntax used to retrieve values from your package can also be used to define new properties or overriding existing ones, below are some examples of how the dot separated syntax can be used to edit your package.json file.

Defining a new bin named mynewcommand in your package.json that points to a file cli.js:

```
npm pkg set bin.mynewcommand=cli.js
```

Setting multiple fields at once is also possible:

```
npm pkg set description='Awesome package' engines.node='>=10'
```

It's also possible to add to array values, for example to add a new contributor entry:

```
npm pkg set contributors[0].name='Foo' contributors[0].email='foo@bar.ca'
```

You may also append items to the end of an array using the special empty bracket notation:

```
  npm pkg set contributors[].name='Foo' contributors[].name='Bar'
```

It's also possible to parse values as json prior to saving them to your package.json file, for example in order to set a "private": true property:

```
  npm pkg set private=true --json
```

It also enables saving values as numbers:

```
  npm pkg set tap.timeout=60 --json
```

? npm pkg delete <key>

Deletes a key from your package.json

The same syntax used to set values from your package can also be used to remove existing ones. For example, in order to remove a script named build:

```
  npm pkg delete scripts.build
```

Workspaces support

You can set/get/delete items across your configured workspaces by using the workspace or workspaces config options.

For example, setting a funding value across all configured workspaces of a project:

```
  npm pkg set funding=https://example.com --ws
```

When using npm pkg get to retrieve info from your configured workspaces, the returned result will be in a json format in which top level keys are the names of each workspace, the values of these keys will be the result values returned from each of the configured workspaces, e.g:

```
  npm pkg get name version --ws
  {
    "a": {
      "name": "a",
      "version": "1.0.0"
    },
    "b": {
      "name": "b",
      "version": "1.0.0"
```

```
  }
}
```

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->

See Also

- ? npm install
- ? npm init
- ? npm config
- ? npm set-script
- ? workspaces