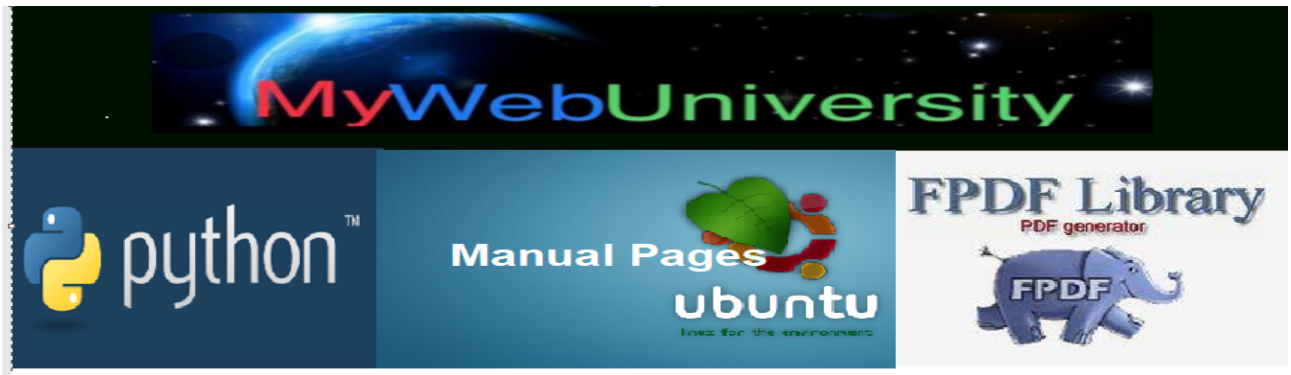

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man npm-update.1

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

NAME

npm-update

Synopsis

<!-- AUTOGENERATED USAGE DESCRIPTIONS -->

Description

This command will update all the packages listed to the latest version

(specified by the tag config), respecting the semver

constraints of both your package and its dependencies (if they also require the

same package).

It will also install missing packages.

If the -g flag is specified, this command will update globally installed

packages.

If no package name is specified, all packages in the specified location (global

or local) will be updated.

Note that by default npm update will not update the semver values of direct

dependencies in your project package.json, if you want to also update

values in package.json you can run: npm update --save (or add the

save=true option to a configuration file

to make that the default behavior).

Example

For the examples below, assume that the current package is app and it depends

on dependencies, dep1 (dep2, .. etc.). The published versions of dep1

are:

```
{
  "dist-tags": { "latest": "1.2.2" },
  "versions": [
    "1.2.2",
    "1.2.1",
    "1.2.0",
    "1.1.2",
    "1.1.1",
    "1.0.0",
    "0.4.1",
    "0.4.0",
    "0.2.0"
  ]
}
```

Caret Dependencies

If app's package.json contains:

```
"dependencies": {
  "dep1": "^1.1.1"
}
```

Then npm update will install dep1@1.2.2, because 1.2.2 is latest and 1.2.2 satisfies ^1.1.1.

Tilde Dependencies

However, if app's package.json contains:

```
"dependencies": {
  "dep1": "~1.1.1"
}
```

In this case, running npm update will install dep1@1.1.2. Even though the latest tag points to 1.2.2, this version do not satisfy ~1.1.1, which is equivalent to >=1.1.1 <1.2.0. So the highest-sorting version that satisfies ~1.1.1 is used, which is 1.1.2.

Caret Dependencies below 1.0.0

Suppose app has a caret dependency on a version below 1.0.0, for example:

```
  "dependencies": {
    "dep1": "^0.2.0"
  }
```

npm update will install dep1@0.2.0, because there are no other versions which satisfy ^0.2.0.

If the dependence were on ^0.4.0:

```
  "dependencies": {
    "dep1": "^0.4.0"
  }
```

Then npm update will install dep1@0.4.1, because that is the highest-sorting version that satisfies ^0.4.0 (>= 0.4.0 <0.5.0)

Subdependencies

Suppose your app now also has a dependency on dep2

```
  {
    "name": "my-app",
    "dependencies": {
        "dep1": "^1.0.0",
        "dep2": "1.0.0"
    }
  }
```

and dep2 itself depends on this limited range of dep1

```
  {
  "name": "dep2",
    "dependencies": {
      "dep1": "~1.1.1"
    }
  }
```

Then npm update will install dep1@1.1.2 because that is the highest version that dep2 allows. npm will prioritize having a single version of dep1 in your tree rather than two when that single version can satisfy the semver requirements of multiple dependencies in your tree.

In this case if you really did need your package to use a newer version you would need to use npm install.

Updating Globally-Installed Packages

npm update -g will apply the update action to each globally installed package that is outdated -- that is, has a version that is different from wanted.

Note: Globally installed packages are treated as if they are installed with a caret semver range specified. So if you require to update to latest you may need to run npm install -g [<pkg>...]

NOTE: If a package has been upgraded to a version newer than latest, it will be downgraded.

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->

See Also

? npm install

? npm outdated

? npm shrinkwrap

? npm registry

? npm folders

? npm ls