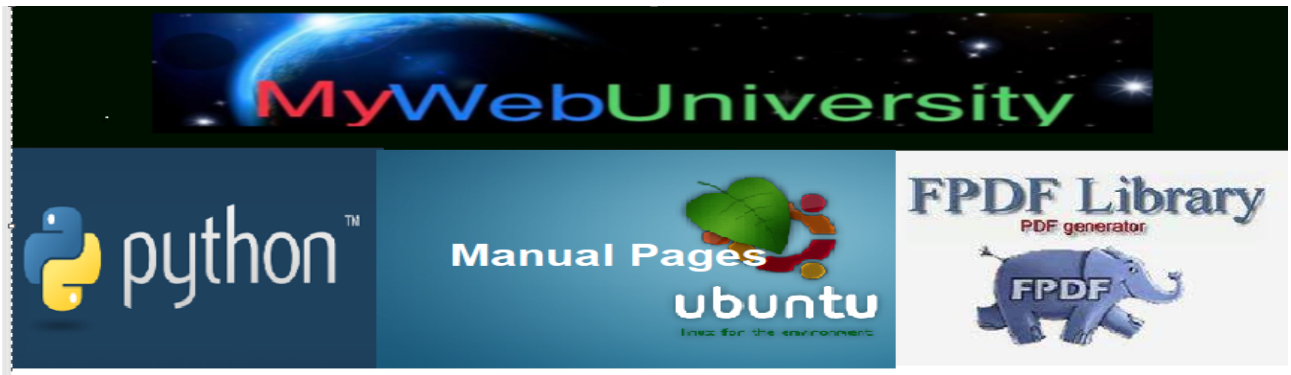

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man npm-init.1

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

NAME

npm-init

Synopsis

<!-- AUTOGENERATED USAGE DESCRIPTIONS -->

Description

npm init <initializer> can be used to set up a new or existing npm

package.

initializer in this case is an npm package named create-<initializer>,

which will be installed by npm-exec, and then have its

main bin executed -- presumably creating or updating package.json and

running any other initialization-related operations.

The init command is transformed to a corresponding npm exec operation as

follows:

? npm init foo -> npm exec create-foo

? npm init @usr/foo -> npm exec @usr/create-foo

? npm init @usr -> npm exec @usr/create

? npm init @usr@2.0.0 -> npm exec @usr/create@2.0.0

? npm init @usr/foo@2.0.0 -> npm exec @usr/create-foo@2.0.0

If the initializer is omitted (by just calling npm init), init will fall

back to legacy init behavior. It will ask you a bunch of questions, and

then write a package.json for you. It will attempt to make reasonable

guesses based on existing fields, dependencies, and options selected. It is

strictly additive, so it will keep any fields and values that were already set. You can also use -y/--yes to skip the questionnaire altogether. If you pass --scope, it will create a scoped package.

Note: if a user already has the create-<initializer> package globally installed, that will be what npm init uses. If you want npm to use the latest version, or another specific version you must specify it:

? npm init foo@latest # fetches and runs the latest create-foo from the registry

? npm init foo@1.2.3 # runs create-foo@1.2.3 specifically

Forwarding additional options

Any additional options will be passed directly to the command, so npm init foo -- --hello will map to npm exec -- create-foo --hello.

To better illustrate how options are forwarded, here's a more evolved example showing options passed to both the npm cli and a create package, both following commands are equivalent:

? npm init foo -y --registry=<url> -- --hello -a

? npm exec -y --registry=<url> -- create-foo --hello -a

Examples

Create a new React-based project using create-react-app:

$ npm init react-app ./my-react-app

Create a new esm-compatible package using create-esm:

$ mkdir my-esm-lib && cd my-esm-lib

$ npm init esm --yes

Generate a plain old package.json using legacy init:

$ mkdir my-npm-pkg && cd my-npm-pkg

$ git init

$ npm init

Generate it without having it ask any questions:

$ npm init -y

Workspaces support

It's possible to create a new workspace within your project by using the
workspace config option. When using npm init -w <dir> the cli will
create the folders and boilerplate expected while also adding a reference
to your project package.json "workspaces": [] property in order to make
sure that new generated workspace is properly set up as such.
Given a project with no workspaces, e.g:

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

You may generate a new workspace using the legacy init:

```
$ npm init -w packages/a
```

That will generate a new folder and package.json file, while also updating
your top-level package.json to add the reference to this new workspace:

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

The workspaces init also supports the npm init <initializer> -w <dir>
syntax, following the same set of rules explained earlier in the initial
Description section of this page. Similar to the previous example of
creating a new React-based project using
create-react-app, the following syntax
will make sure to create the new react app as a nested workspace within your
project and configure your package.json to recognize it as such:

```
npm init -w packages/my-react-app react-app .
```

This will make sure to generate your react app as expected, one important
consideration to have in mind is that npm exec is going to be run in the
context of the newly created folder for that workspace, and that's the reason
why in this example the initializer uses the initializer name followed with a
dot to represent the current directory in that context, e.g: react-app .:

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

```
+-- a
|   `-- package.json
`-- my-react-app
    +-- README
    +-- package.json
    `-- ...
```

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->

See Also

- ? package spec
- ? init-package-json module
- ? package.json
- ? npm version
- ? npm scope
- ? npm exec
- ? npm workspaces