npm-package-lock.json

A manifestation of the manifest

DESCRIPTION

`package-lock.json` is automatically generated for any operations where npm modifies either the `node_modules` tree, or `package.json`. It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.

This file is intended to be committed into source repositories, and serves various purposes:

- Describe a single representation of a dependency tree such that teammates, deployments, and continuous integration are guaranteed to install exactly the same dependencies.
- Provide a facility for users to "time-travel" to previous states of `node_modules` without having to commit the directory itself.
- To facilitate greater visibility of tree changes through readable source control diffs.
- And optimize the installation process by allowing npm to skip repeated metadata resolutions for previously-installed packages.

One key detail about `package-lock.json` is that it cannot be published, and it will be ignored if found in any place other than the toplevel package. It shares a format with npm-shrinkwrap.json, which is essentially the same file, but allows publication. This is not recommended unless deploying a CLI tool or otherwise using the publication process for producing production packages.

If both `package-lock.json` and `npm-shrinkwrap.json` are present in the root of a package, `package-lock.json` will be completely ignored.

FILE FORMAT

name

The name of the package this is a package-lock for. This must match what's in `package.json`.

version

The version of the package this is a package-lock for. This must match what's in `package.json`.

lockfileVersion

An integer version, starting at `1` with the version number of this document whose semantics were used when generating this `package-lock.json`.

packageIntegrity

This is a subresource integrity value created from the `package.json`. No preprocessing of the `package.json` should be done. Subresource integrity strings can be produced by modules like `ssri`.

preserveSymlinks

Indicates that the install was done with the environment variable `NODE_PRESERVE_SYMLINKS` enabled. The installer should insist that the value of this property match that environment variable.

dependencies

A mapping of package name to dependency object. Dependency objects have the following properties:

version

This is a specifier that uniquely identifies this package and should be usable in fetching a new copy of it.

- bundled dependencies: Regardless of source, this is a version number that is purely for informational purposes.
- registry sources: This is a version number. (eg, `1.2.3`)
- git sources: This is a git specifier with resolved committish. (eg, `git+https://example.com/foo/bar#115311855adb0789a0466714ed48a1499ffea97e`)
- http tarball sources: This is the URL of the tarball. (eg, `https://example.com/example-1.3.0.tgz`)
- local tarball sources: This is the file URL of the tarball. (eg `file:///opt/storage/example-1.3.0.tgz`)
- local link sources: This is the file URL of the link. (eg `file:libs/our-module`)

integrity

This is a Standard Subresource Integrity for this resource.

- For bundled dependencies this is not included, regardless of source.
- For registry sources, this is the `integrity` that the registry provided, or if one wasn't provided the SHA1 in `shasum`.
- For git sources this is the specific commit hash we cloned from.

- For remote tarball sources this is an integrity based on a SHA512 of the file.
- For local tarball sources: This is an integrity field based on the SHA512 of the file.

resolved

- For bundled dependencies this is not included, regardless of source.
- For registry sources this is path of the tarball relative to the registry URL. If the tarball URL isn't on the same server as the registry URL then this is a complete URL.

bundled

If true, this is the bundled dependency and will be installed by the parent module. When installing, this module will be extracted from the parent module during the extract phase, not installed as a separate dependency.

dev

If true then this dependency is either a development dependency ONLY of the top level module or a transitive dependency of one. This is false for dependencies that are both a development dependency of the top level and a transitive dependency of a non-development dependency of the top level.

optional

If true then this dependency is either an optional dependency ONLY of the top level module or a transitive dependency of one. This is false for dependencies that are both an optional dependency of the top level and a transitive dependency of a non-optional dependency of the top level.

All optional dependencies should be included even if they're uninstallable on the current platform.

requires

This is a mapping of module name to version. This is a list of everything this module requires, regardless of where it will be installed. The version should match via normal matching rules a dependency either in our `dependencies` or in a level higher than us.

dependencies

The dependencies of this dependency, exactly as at the top level.