[**https://www.codeproject.com/Articles/1202361/What-is-package-lock-json-file-in-Node-NPM**](https://www.codeproject.com/Articles/1202361/What-is-package-lock-json-file-in-Node-NPM)

When you are doing development in Angular, Node NPM is your tool for package management. In simple words, we have a “*package.json*” file and all dependencies are listed inside it. When you are doing NPM, you will always find “*package-lock.json*” file. So in this tutorial, we will unleash the importance of this lock file.

To understand the importance of lock, let's understand how software versioning works.

Most software versions follow semantic versioning. In semantic versioning, versions are divided into three distinct numbers as shown in the image below.

|  |  |
| --- | --- |
| The first number is termed as “major version”, second “minor version” and third “revision”.    **Major version**: Any increment in major version is an indication that there are breaking changes in the software functionality. It’s very much possible that the old code will not work with these changes and have to be tested properly.    **Minor version**: This version is incremented when we add new features, but the old code still works.    **Revision**: This version is incremented when we are just doing bug fixes. So there are no new functionalities added, no breaking changes and backward compatible with old code. | https://www.codeproject.com/KB/Nodejs/1202361/1.jpg |

|  |  |
| --- | --- |
| NPM follows semantic versioning, but it also has some more special characters like “^”, “~”, “>” and so on. They dictate how NPM get latest should behave for Major and Minor versions.    For these formats, 3 formats are very primary. Let’s understand each of them.    Exact (1.6.5), Major/Minor ( ^1.6.5) or Minor(~1.6.5). | https://www.codeproject.com/KB/Nodejs/1202361/2.jpg |

|  |  |
| --- | --- |
| **Exact (1.6.5)**: This will do a get latest of exact version 1.6.5 not more or not less. If that version is not available, it will throw up an exception.    **Major/Minor(^1.6.5)**: The carrot sign will get minimum 1.6.5 and if there are any higher MINOR / REVISION versions, it will get that. It WILL NEVER GET HIGHER MAJOR VERSIONS. So if 1.6.5 has 1.6.7 it will get that, if it has 1.7.7 it will that, but if it has 2.0 it will NOT get that.    **Minimum or lower (~1.6.5)**: The tilde sign will get HIGHER REVISIONS. For if 1.6.5 has 1.6.7 it will get that, but if it has 1.7.5 it will not be installed, if it has 2.0 it will not be installed. | https://www.codeproject.com/KB/Nodejs/1202361/3.jpg |

As discussed in the previous sections, *package.json* has “^” and “~” versioning mechanism. Now suppose in your *package.json*, you have mentioned "jquery": "^3.1.0"and Jquery has a new version “3.2.1”. So in actual, it will install or in other words, LOCK DOWN to “3.2.1”.

So in *package.json*, you will have “^3.1.0”, but actually you will be using “3.2.1”. This entry of actual version is present in “*package-lock.json*”. So package lock files have the EXACT versions which are used in your code.

Below is the image snapshot of both the files.

