**Understanding Semver**

Let's look at the dependencies in the **package.json** file:

**"dependencies": {  
    "pino": "^6.2.1"  
  },  
  "devDependencies": {  
    "standard": "^14.3.3"  
  }**

We've installed two dependencies, **pino** at a Semver range of ^6.2.1 and **standard** at a Semver range of ^14.3.3. Our package version number is the Semver version 1.0.0. There is a distinction between the Semver format and a Semver range.

Understanding the Semver format is crucial to managing dependencies. A Semver is fundamentally three numbers separated by dots. The reason a version number is updated is because a change was made to the package. The three numbers separated by dots represent different types of change.

*Click on each box to learn more about these different types of changes.*

**Understanding Semver**

* MAJOR

MAJOR is the left-most number. It means that the change breaks an API or a behavior.

* MINOR

MINOR is the middle number. It means that the package has been extended in some way, for instance a new method, but it's fully backwards compatible. Upgrading to a minor should not break the package.

* PATCH

PATCH is the right-most number. It means that there has been a bug fix.

This is the core of the Semver format, but there are extensions which won't be covered here, for more information on Semver see [Semver's website](https://semver.org/" \t "_blank).

A Semver range allows for a flexible versioning strategy. There are many ways to define a Semver range.

One way is to use the character "x" in any of the MAJOR.MINOR.PATCH positions, for example 1.2.x will match all PATCH numbers. 1.x.x will match all MINOR and PATCH numbers.

By default **npm install** prefixes the version number of a package with caret (^) when installing a new dependency and saving it to the **package.json** file.

Our specified **pino** version in the **package.json** file is ^6.2.1. This is another way to specify a Semver range: by prefixing the version with a caret (^). Using a caret on version numbers is basically the same as using an x in the MINOR and PATCH positions, so ^6.2.1 is the same as 5.x.x. However there are exceptions when using 0, ^0.0.0 is not the the same as 0.x.x, see the [*"Caret Ranges ^1.2.3 ^0.2.5 ^0.0.4"*](https://docs.npmjs.com/misc/semver#caret-ranges-123-025-004) section of npmjs Documentation. However for non-zero MAJOR numbers, ^MAJOR.MINOR.PATCH is interpreted as MAJOR.x.x.

The complete syntax for defining ranges is verbose, see the [*"npm-semver: The semantic versioner for npm"*](https://docs.npmjs.com/misc/semver) section of npmjs Documentation.