Entendiendo el Versionado semántico

Índice

1.

Introducción

Introducción

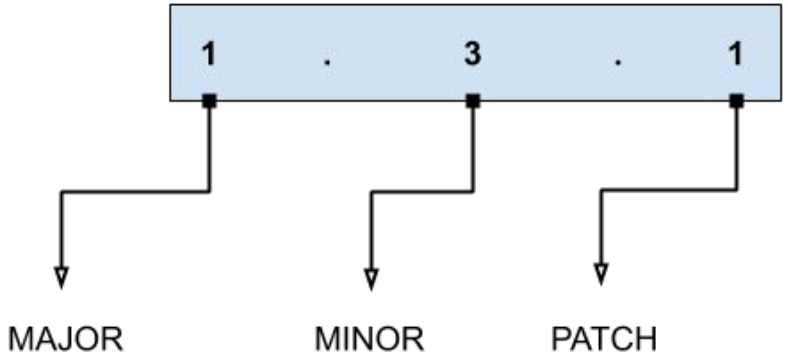
En el mundo del desarrollo del software existe un concepto denominado “dependency hell” que ocurre cuando eres incapaz de actualizar un paquete sin tener que lanzar nuevas versiones de todos los paquetes de los que depende.

Introducción

En el mundo del desarrollo del software existe un concepto denominado “dependency hell” que ocurre cuando eres incapaz de actualizar un paquete sin tener que lanzar nuevas versiones de todos los paquetes de los que depende.

Versionados como **SemVer** entran en juego para solucionar este problema mediante la introducción de **unas reglas que todo el mundo debería de seguir**.

Introducción



2.

Reglas

Reglas

Todas las reglas están recogidas en la especificación de **SemVer**. Las más importantes son:

Reglas

Todas las reglas están recogidas en la especificación de **SemVer**. Las más importantes son:

- **MAJOR**: Cambio es incompatible con el API actual.
- **MINOR**: Cambios incrementales. Agregar funcionalidad pero no romper nada.
- **PATCH**: Arreglos de bugs que no rompen el API.

Reglas

Todas las reglas están recogidas en la especificación de **SemVer**. Las más importantes son:

- **MAJOR**: Cambio es incompatible con el API actual.
- **MINOR**: Cambios incrementales. Agregar funcionalidad pero no romper nada.
- **PATCH**: Arreglos de bugs que no rompen el API.

Adicionalmente es posible lanzar versiones preliminares para testear. En este caso contamos también con:

- PREMAJOR
- PREMINOR
- PREPATCH
- PRERELEASE

Reglas

Un poco más técnico:

- **MAJOR**: Non-Backwards-compatible changes.
- **MINOR**: Backwards-compatible features are added
- **PATCH**: Backwards-compatible bugs are fixed