Semantic Versioning

El control de versiones semantico

BY DAVID HERNANDEZ

Qué es_?

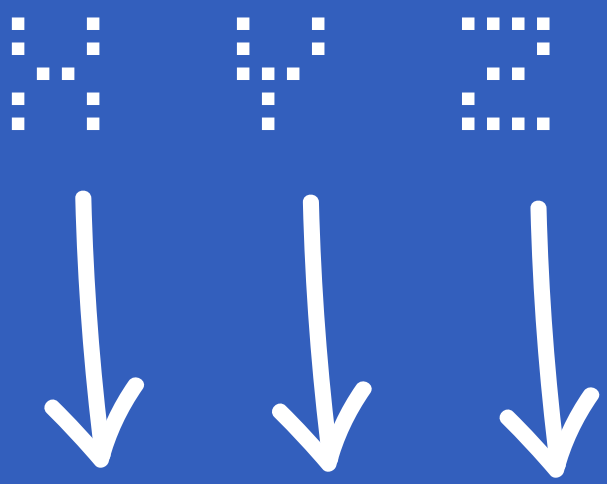
También conocido como SemVer, es un sistema de control de versiones que ha ido en aumento en los últimos años. Siempre ha sido un problema para los desarrolladores de software, los administradores de versiones y los consumidores. Tener una forma universal de versionar los proyectos de desarrollo de software, es la mejor manera de realizar un seguimiento de lo que sucede con el software, ya que casi todos los días se crean nuevos complementos, funcionalidades, bibliotecas y extensiones.

¿Por qué es importante el control de versiones del software?

El control de versiones del software nos ayuda a identificar el estado en el que se encuentra el software o el paquete con un nombre único y/o un número. Las versiones ayudan a los desarrolladores a realizar un seguimiento de los cambios en el software o los paquetes de terceros que están utilizando.

Existen diferentes tipos de esquemas de control de versiones, pero uno de los más populares es un esquema llamado Control de versiones semántico (SemVer) que fue propuesto por Tom Preston-Werner en 2013 para especificar cómo se debe controlar la numeración de versiones.

Formato



1.0.0

Major Version
Breaking change, e.g. a rebrand, feature set added

Minor Version
Non-breaking noteworthy change, e.g. new component, updated styles

Patch Version
Small request or bug fix, e.g. update or edit existing elements

Versionado semántico es un número de 3 componentes en formato XYZ, donde:

La especificación SemVer asume que los proyectos que utilizan este esquema deben tener una API pública.

X representa una versión **principal** debería aumentar cuando hemos introducido una nueva funcionalidad que rompe nuestra API, es decir, aumentar este valor cuando hemos agregado un cambio incompatible con versiones anteriores a nuestro proyecto. Cuando se aumenta este número, debemos restablecer los números Minor y Patch a 0 .

Y representa una versión **menor**. Se utiliza para el lanzamiento de nuevas funciones en el sistema. Cuando aumenta la versión secundaria debe restablecer la versión del parche a cero. Si la versión actual es la 2.6.9, la próxima actualización para una versión secundaria será la 2.7.0. Aumenta el valor de Y al implementar nuevas funciones de forma compatible con versiones anteriores

Z significa **Versiones de parches**: las versiones de los parches se utilizan para corregir errores. No hay cambios de funcionalidad en las actualizaciones de la versión del parche. Si la versión actual es 2.6.9, la próxima versión para una actualización de parche será 2.6.10. No hay límite para estos números. Aumente el valor de Z al corregir errores

Puntos a tener en cuenta:

- La primera versión comienza en 0.1.0 y no en 0.0.1, ya que no se han realizado correcciones de errores, sino que comenzamos con un conjunto de características como el primer borrador del proyecto.

- Antes de 1.0.0 es solo la fase de desarrollo, donde te enfocas en hacer las cosas. Esta etapa es para los desarrolladores en los que se estan creando el sistema.

- SemVer no cubre las bibliotecas con la etiqueta 0.*.* . **La primera versión estable es la 1.0.0.**