<https://www.youtube.com/watch?v=QMUSkra7Blk>

<https://blog.greenkeeper.io/introduction-to-semver-d272990c44f2>

Basics:

* It’s a three number system for versioning release of a software.
* It requires a public API to see how we are breaking the API or how the number is changing the API-
* The Numbers 
  8.2.6 
  Major version 
  Minor version 
  Patch 

Major Changes:

* Breaks the API
* There is no backward compatibility.

Minor Changes:

* New Feature Added.
* Does not break the API.

Patch

* For a bug fix
* Never breaks the API or backwards compatibility (Software should be able to work with the older version).

The Numbers 
Major version 
Major changes 
Breaks the API 
82.6 
Minor version 
New features 
Does not break API 
Patch 
Bug fixes 

8.2.6:

This tells us that:

* There have been 7 breaking changes that we made.
* 2 new features have been added
* 6 bugs were fixed after the 2nd minor version were added.
* If a new bug is fixed in 8.2.6, then a new version would be 8.2.7
* If a new feature is added to 8.2.7 version, then it would be changed to 8.3.0. Here minor version is changed from 2 to 3 and patch is reset to 0.
* If we are breaking the backwards compatibility with the Api, then the version would change to 9.0.0

Pre-release versioning:

* We want to track the version even before the release.
* So we can use following options:

Pre-release Versioning 
Initial development uses Major version O (e.g. 0.1.0) 
You may use pre-release strings (e.g. alphal, rc3, etc.) 
• These are appended to the end (e.g. 1.0.O.a/pha1) 
A publicly-released API starts at 1.0.0 

Benefits of Semantic Versioning

1. Clearer compatibility /dependencies:

* If some is using my API and dependent on it, then proper version will allow him to see how he/she can use my changes.

1. Encourages well defined APIs in terms of documentation etc.

1. Makes upgrade decisions clearer.