Учитывая номер версии МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ, следует увеличивать:

1. МАЖОРНУЮ версию, когда сделаны обратно несовместимые изменения API.
2. МИНОРНУЮ версию, когда вы добавляете новую функциональность, не нарушая обратной совместимости.
3. ПАТЧ-версию, когда вы делаете обратно совместимые исправления.

Дополнительные обозначения для предрелизных и билд-метаданных возможны как дополнения к МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ формату.

**Спецификация Семантического Версионирования**

Рассмотрим формат версий **X.Y.Z** (мажорная, минорная, патч). Баг-фиксы, не влияющие на API, увеличивают патч-версию, обратно совместимые добавления/изменения API увеличивают минорную версию и обратно несовместимые изменения API увеличивают мажорную версию.

Слова «ДОЛЖЕН» (MUST), «НЕ ДОЛЖЕН» (MUST NOT), «ОБЯЗАТЕЛЬНО» (REQUIRED), «СЛЕДУЕТ» (SHOULD), «НЕ СЛЕДУЕТ» (SHOULD NOT), «РЕКОМЕНДОВАННЫЙ» (RECOMMENDED), «МОЖЕТ» (MAY) и «НЕОБЯЗАТЕЛЬНЫЙ» (OPTIONAL) в этом документе должны быть интерпретированы в соответствии с [RFC 2119](http://tools.ietf.org/html/rfc2119).

1. ПО, использующее Семантическое Версионирование, должно объявить публичный API. Этот API может быть объявлен самим кодом или существовать строго в документации. Как бы ни было это сделано, он должен быть точным и исчерпывающим.
2. Обычный номер версии ДОЛЖЕН иметь формат X.Y.Z, где X, Y и Z — неотрицательные целые числа и НЕ ДОЛЖНЫ начинаться с нуля. X — мажорная версия, Y — минорная версия и Z — патч-версия. Каждый элемент ДОЛЖЕН увеличиваться численно. Например: 1.9.0 ->1.10.0 -> 1.11.0.
3. После релиза новой версии пакета содержание этой версии НЕ ДОЛЖНО быть модифицировано. Любые изменения ДОЛЖНЫ быть выпущены как новая версия.
4. Мажорная версия ноль (0.y.z) предназначена для начальной разработки. Всё может измениться в любой момент. Публичный API не должен рассматриваться как стабильный.
5. Версия 1.0.0 определяет публичный API. После этого релиза номера версий увеличиваются в зависимости от того, как изменяется публичный API.
6. Патч-версия Z (x.y.Z | x > 0) ДОЛЖНА быть увеличена только если содержит обратно совместимые баг-фиксы. Определение баг-фикс означает внутренние изменения, которые исправляют некорректное поведение.
7. Минорная версия (x.Y.z | x > 0) ДОЛЖНА быть увеличена, если в публичном API представлена новая обратно совместимая функциональность. Версия ДОЛЖНА быть увеличена, если какая-либо функциональность публичного API помечена как устаревшая (deprecated). Версия МОЖЕТ быть увеличена в случае реализации новой функциональности или существенного усовершенствования в приватном коде. Версия МОЖЕТ включать в себя изменения, характерные для патчей. Патч-версия ДОЛЖНА быть обнулена, когда увеличивается минорная версия.
8. Мажорная версия X (X.y.z | X > 0) ДОЛЖНА быть увеличена, если в публичном API представлены какие-либо обратно несовместимые изменения. Она МОЖЕТ включать в себя изменения, характерные для уровня минорных версий и патчей. Когда увеличивается мажорная версия, минорная и патч-версия ДОЛЖНЫ быть обнулены.
9. Предрелизная версия МОЖЕТ быть обозначена добавлением дефиса и серией разделённых точкой идентификаторов, следующих сразу за патч-версией. Идентификаторы ДОЛЖНЫ содержать только ASCII буквенно-цифровые символы и дефис [0-9A-Za-z-]. Идентификаторы НЕ ДОЛЖНЫ быть пустыми. Числовые идентификаторы НЕ ДОЛЖНЫ начинаться с нуля. Предрелизные версии имеют более низкий приоритет, чем соответствующая релизная версия. Предрелизная версия указывает на то, что эта версия не стабильна и может не удовлетворять требованиям совместимости, обозначенными соответствующей нормальной версией. Примеры: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92.
10. Сборочные метаданные МОГУТ быть обозначены добавлением знака плюс и ряда разделённых точкой идентификаторов, следующих сразу за патчем или предрелизной версией. Идентификаторы ДОЛЖНЫ содержать только ASCII буквенно-цифровые символы и дефис [0-9A-Za-z-]. Идентификаторы НЕ ДОЛЖНЫ быть пустыми. Сборочные метаданные СЛЕДУЕТ игнорировать, когда определяется старшинство версий. Поэтому два пакета с одинаковой версией, но разными сборочными метаданными, рассматриваются как одна и та же версия. Примеры: 1.0.0-alpha+001, 1.0.0+20130313144700, 1.0.0-beta+exp.sha.5114f85.
11. Приоритет определяет, как версии соотносятся друг с другом, когда упорядочиваются. Приоритет версий ДОЛЖЕН рассчитываться путём разделения номеров версий на мажорную, минорную, патч и предрелизные идентификаторы. Именно в такой последовательности (сборочные метаданные не фигурируют в расчёте). Приоритет определяется по первому отличию при сравнении каждого из этих идентификаторов слева направо: Мажорная, минорная и патч-версия всегда сравниваются численно. Пример: 1.0.0 < 2.0.0 < 2.1.0 < 2.1.1. Когда мажорная, минорная и патч-версия равны, предрелизная версия имеет более низкий приоритет, чем нормальная версия. Пример: 1.0.0-alpha < 1.0.0. Приоритет двух предрелизных версий с одинаковыми мажорной, минорной и патч-версией ДОЛЖНЫ быть определены сравнением каждого разделённого точкой идентификатора слева направо до тех пор, пока различие не будет найдено следующим образом: идентификаторы, состоящие только из цифр, сравниваются численно; буквенные идентификаторы или дефисы сравниваются лексически в ASCII-порядке. Численные идентификаторы всегда имеют низший приоритет, чем символьные. Больший набор предрелизных символов имеет больший приоритет, чем меньший набор, если сравниваемые идентификаторы равны. Пример: 1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-alpha.beta < 1.0.0-beta < 1.0.0-beta.2 < 1.0.0-beta.11 < 1.0.0-rc.1 < 1.0.0.