Semantische Versionierung anwenden

Disclaimer

Die **semantische Versionierung** ist ein gängiger Standard in der Softwareentwicklung, um Versionsnummern zu vergeben. Sie macht Änderungen im Code nachvollziehbar, erleichtert den Umgang mit Abhängigkeiten und verbessert die Kommunikation über den Entwicklungsstand.

Weitere Informationen: semver.org

Was ist die semantische Versionierung?

Die semantische Versionierung beschreibt ein dreiteiliges Versionsschema:

```
MAJOR.MINOR.PATCH
```

- **MAJOR**: Inkompatible Änderungen der öffentlichen API
- **MINOR**: Neue, abwärtskompatible Funktionalitäten
- **PATCH**: Abwärtskompatible Fehlerbehebungen

Beispiele aus der Praxis

- `1.2.3`: Erste stabile Version, zwei kleinere Feature-Erweiterungen, drei Bugfixes
- `2.0.0`: Führt zu inkompatiblen API-Änderungen, z. B. durch Entfernen oder Umbenennen von Funktionen
- `1.3.0`: Fügt neue Funktionalitäten hinzu, ohne vorhandene zu verändern
- `1.3.1`: Behebt einen Fehler aus der Version 1.3.0

Regeln der semantischen Versionierung

1. Erhöhe die **MAJOR**-Version bei inkompatiblen API-Änderungen
2. Erhöhe die **MINOR**-Version bei neuen, abwärtskompatiblen Funktionen
3. Erhöhe die **PATCH**-Version bei abwärtskompatiblen Bugfixes

Vorabversionen und Build-Metadaten

Zusätzlich können Vorabversionen und Build-Metadaten angegeben werden:

```
1.0.0-alpha
1.0.0-beta+exp.sha.5114f85
```

- **Vorabversionen:** `-alpha`, `-beta`, `-rc` usw. – für nicht stabile Entwicklungsstände
- **Build-Metadaten:** `+build` – zusätzliche Informationen wie Git-Hashes oder Build-Zeitpunkte

---

Zusammenfassung

- Die semantische Versionierung gibt eine klare Struktur für Versionen vor
- Sie erleichtert das Management von Abhängigkeiten in Projekten
- Sie wird häufig in Kombination mit Conventional Commits verwendet

---

Version #2
Erstellt: 18 Mai 2025 16:15:30 von Justin Fiedler
Zuletzt aktualisiert: 18 Mai 2025 16:33:18 von Justin Fiedler