Velinkungsproblem von globalen Modulen

Bei **nvm-windows** liegt das Problem an einem bekannten Mechanismus: `nvm-windows` verwaltet Node-Versionen über einen **Symlink** (standardmäßig `C:\Program Files\nodejs`). Dieser Symlink zeigt dynamisch auf die gerade aktive Node-Version unter `%APPDATA%\nvm\v<version>`.
Wenn du `npm install -g <package>` ausführst, gibt es zwei typische Szenarien, warum das Binary nicht mehr im `PATH` landet:

1. **Ein abweichendes oder altes globales `prefix` in der `.npmrc`:** Wenn dort noch ein hartkodierter Pfad (wie `%APPDATA%\npm`) hinterlegt ist, legt npm die `.cmd`-Dateien dort ab, aber nvm-windows bindet diesen Pfad **nicht** in den `PATH` ein – nvm erwartet globale Pakete direkt im Symlink-Verzeichnis der jeweiligen Node-Version.
2. **Fehlende Berechtigungen im Node-Symlink:** Liegt Node unter `C:\Program Files\nodejs`, kann npm ohne Admin-Rechte dort keine `.cmd`-Dateien anlegen und weicht stillschweigend aus.

Schritt 1: Prüfen, wo npm globale Pakete aktuell ablegt

Führe in der PowerShell aus:

```
npm config get prefix
```

- **Fall A (Standard bei nvm-windows):** Der Pfad zeigt auf das aktive Symlink-Verzeichnis (z. B. `C:\Program Files\nodejs`). Hier müssten die `.cmd`-Dateien liegen.
- **Fall B (Häufige Ursache):** Der Pfad zeigt auf `C:\Users\<Name>\AppData\Roaming\npm`. In diesem Fall findet Windows die Befehle nicht, weil dieser Ordner bei nvm-windows **nicht** im `PATH` steht.

Lösungsweg 1: Den dedizierten npm-Ordner nutzen (Empfohlen & Best Practice)

Damit globale Pakete auch bei einem Node-Versionswechsel über `nvm use` erhalten bleiben und keine Admin-Rechte im Programmverzeichnis nötig sind, bindet man den globalen npm-Pfad fest ein.

1. **Globales Prefix in npm festlegen:**

```
npm config set prefix "$env:APPDATA\npm"
```

2. **Diesen Ordner dauerhaft in den Benutzer-`PATH` eintragen:**

```
[Environment]::SetEnvironmentVariable(
    "Path",
    [Environment]::GetEnvironmentVariable("Path", "User") + ";$env:APP
    "User"
)
```

3. **Paket nochmals global installieren:**

```
npm install -g gulp-cli
```

4. **Terminal neu starten** (Windows Terminal / PowerShell schließen und neu öffnen), danach testen:

```
gulp -v
```

Lösungsweg 2: Den nvm-Standard wiederherstellen

Falls du möchtest, dass nvm-windows globale Pakete strikt pro Node-Version im jeweiligen Versionsordner verwaltet:

1. **Globales Prefix zurücksetzen:**

```
npm config delete prefix
```

2. **Prüfen, ob der nvm-Symlink im `PATH` vorhanden ist:**

```
$env:PATH -split ';' | Select-String "nodejs"
```

Hier muss der nvm-Symlink (`C:\Program Files\nodejs` oder dein benutzerdefinierter Pfad) auftauchen. 3. **Installieren:**

```
npm install -g gulp-cli
```

(Hinweis: Falls eine Fehlermeldung bezüglich `EPERM` oder fehlender Schreibrechte auftaucht, die Konsole einmalig als Administrator ausführen oder Lösungsweg 1 wählen).