# Zsh bietet eine leistungsstarke Funktion namens

Nachdem ich “Let Zsh Display Proxy Settings Before Running Network Commands” erkundet hatte, bin ich mit ChatGPT tiefer in Zsh-Hooks eingetaucht. Hier ist eine prägnante Übersicht für zukünftige Referenz.

---

In Zsh ermöglichen es Hooks, benutzerdefinierte Funktionen an bestimmten Punkten während der Shell-Operationen auszuführen. Neben `preexec` bietet Zsh mehrere Hooks, um Ihre Umgebung zu erweitern:

**1.** `precmd`

- Wann: Bevor die Eingabeaufforderung angezeigt wird.
- Verwendung: Aktualisieren der Eingabeaufforderung oder Durchführen von Aufräumarbeiten.
- Beispiel:

```
precmd() {
  echo "Bereit für den nächsten Befehl!"
}
```

**2.** `chpwd`

- Wann: Wenn sich das aktuelle Verzeichnis ändert.
- Verwendung: Umgebungsvariablen aktualisieren oder Aktionen basierend auf dem Verzeichnis auslösen.
- Beispiel:

```
chpwd() {
  echo "Gewechselt zu: $PWD"
}
```

**3.** `preexec_functions` **&** `precmd_functions`

- Wann: Ähnlich wie `preexec` und `precmd`, unterstützen jedoch mehrere Funktionen.
- Verwendung: Mehrere Aktionen anhängen, ohne bestehende Hooks zu überschreiben.
- Beispiel:

```
precmd_functions+=(additional_precmd)

additional_precmd() {
  echo "Zusätzliche precmd-Aufgabe."
}
```

**4.** `TRAPDEBUG`

- Wann: Nach jedem Befehl, bevor die Ergebnisse angezeigt werden.
- Verwendung: Debugging, Protokollierung von Befehlen.
- Beispiel:

```
TRAPDEBUG() {
  echo "Ausgeführt: $1"
}
```

**5.** `TRAPEXIT`

- Wann: Wenn die Shell beendet wird.
- Verwendung: Aufräumarbeiten oder Anzeigen von Abschiedsnachrichten.
- Beispiel:

```
TRAPEXIT() {
  echo "Auf Wiedersehen!"
}
```

### 6. `zle` Hooks

- Wann: Während der Zeilenbearbeitung.
- Verwendung: Anpassung des Verhaltens der Befehlszeile.
- Beispiel:

```
zle-line-init() {
  echo "Bearbeite einen neuen Befehl."
}
zle -N zle-line-init
```

### 7. History Hooks (`zshaddhistory`, `zshremovehistory`)

- Wann: Beim Hinzufügen oder Entfernen von Verlaufseinträgen.
- Verwendung: Verlauf filtern oder verwalten.
- Beispiel:

```
zshaddhistory() {
  [[ $1 == *"secret"* ]] && return 1
  return 0
}
```

### 8. `periodic`

- Wann: In Intervallen, die durch `period` festgelegt sind.
- Verwendung: Routinemäßige Überprüfungen oder Aktualisierungen.
- Beispiel:

```
periodic() {
  echo "Periodische Aufgabe wird ausgeführt..."
}
```

**9.** `add-zsh-hook`

- Zweck: Funktionen sicher zu Hooks hinzufügen.
- Verwendung: Mehrere Funktionen anhängen, ohne vorhandene zu überschreiben.
- Beispiel:

```
add-zsh-hook precmd another_precmd

another_precmd() {
  echo "Eine weitere precmd-Funktion."
}
```

**Zusammenfassung**

Das Hook-System von Zsh ist vielseitig und ermöglicht Automatisierung und Anpassung:

- `preexec`: Vor der Befehlsausführung.
- `precmd`: Vor der Anzeige der Eingabeaufforderung.
- `chpwd`: Bei Verzeichniswechsel.
- `TRAPDEBUG`: Nach dem Befehl für Debugging-Zwecke.
- `TRAPEXIT`: Beim Beenden der Shell.
- `zle`-Hooks: Während der Zeilenbearbeitung.
- History-Hooks: Verwaltung der Befehlsgeschichte.
- `periodic`: In festgelegten Intervallen.
- `add-zsh-hook`: Hinzufügen mehrerer Hook-Funktionen.

Die Nutzung dieser Hooks kann Ihr Zsh-Erlebnis erheblich verbessern, indem es Ihre Shell effizienter und besser an Ihren Arbeitsablauf angepasst macht.