Was ist Pi.Dev?

Einleitung und Kontext

Das Video stellt **Pi** vor (auch bekannt als "Shitty Coding Agent" oder offiziell unter pi.dev erreichbar) - einen minimalistischen, erweiterbaren Coding-Agenten, der sich bewusst von der Feature-Überladung anderer Tools absetzt. Der Ersteller des Videos präsentiert Pi als eine erfrischende Alternative zu etablierten Agenten wie Claude Code, Codex oder Open Code.

https://youtu.be/OMFIPv8a4qA

Der Schöpfer: Mario Zechner (BadLogic)

Pi wurde von **Mario Zechner** entwickelt, bekannt unter seinem Online-Pseudonym "BadLogic". Er ist ein erfahrener Open-Source-Enthusiast, der in der Vergangenheit **libGDX** entwickelt hat - eines der populärsten Game-Development-Frameworks für Java. Mario war frustriert von allen existierenden Coding-Agenten und entschied sich, einen eigenen zu bauen, der seiner Philosophie entspricht.

Zitat von Mario:

> "Ich hasse alle existierenden Coding-Agenten. Pi's Philosophie ist: Passe den Coding-Agenten an deine Bedürfnisse an, anstatt umgekehrt."

Die Kernphilosophie: Was Pi NICHT hat

Das Besondere an Pi ist ironischerweise das, was es **nicht** mitbringt. Der Video-Ersteller betont wiederholt, dass die Stärke in der Abwesenheit von Features liegt:

Bewusst weggelassene Features:

- **Keine MCPs** (Model Context Protocol Server)
- **Keine Sub-Agenten**
- **Kein Plan-Modus**
- **Kein Background-Bash**
- **Keine Permission-Pop-ups**
- **Keine übermäßig komplexe UI**

Kritik an Claude Code:

Mario beschreibt Claude Code als "Raumschiff" mit einer exzessiven Menge an Features:

> "Füge dieses Feature hinzu und jenes Feature... irgendwann ist Claude Code zu einem Raumschiff geworden. Es macht so viele Dinge, dass du wahrscheinlich nur 5% davon jemals nutzt. Du kennst vielleicht 10% insgesamt, und der Rest - die übrigen 90% - das ist wie die dunkle Materie der KI."

Kritik an Open Code:

- Häufige Session-Kompaktierung macht den Agenten "dümmer" während Gesprächen
- Stilles Löschen von Tool-Outputs nach einer bestimmten Token-Menge
- LSP-Unterstützung (Language Server Protocol) kann problematisch sein, da der Agent während der Implementierung sofortiges (falsches) Feedback erhält, bevor die Änderungen fertig sind

---

Technische Details und Installation

Installation:

```
npm install
```

Pi ist in **TypeScript** geschrieben - was der Video-Ersteller humorvoll als "die Sprache für KI" bezeichnet.

Konfiguration:

- Einstellungen unter: `~/.pi/agent_settings.json`
- Sessions werden als JSON-Dateien gespeichert (im Gegensatz zu Open Code, das kürzlich auf SQLite umgestellt hat)
- System-Prompt ist minimal und vollständig anpassbar

Provider-Unterstützung:

Pi ist **modell-agnostisch** und unterstützt:

- Anthropic (Claude)
- OpenAI
- Ollama (lokale Modelle)
- Andere lokale Provider
- "Big Pickle" (kostenloser Anbieter für einfache Anfragen)

Wichtige Funktionen und Shortcuts

Session-Management:

Befehl/Shortcut	Funktion
`pi`	Agent starten
`pi -r`	Session fortsetzen
`--no-session`	Ohne Session-Speicherung ausführen
`Ctrl+D`	Session beenden, zurück zur CLI
`Ctrl+G`	Prompt zum Editor senden
`/tree`	Hierarchie der Session-Branches anzeigen
`!`	Ad-hoc Bash-Befehle ausführen

Session-Branching:

Eine besonders nützliche Funktion ist die Möglichkeit, **Sessions zu forken** und von jedem Punkt in der Konversation abzuzweigen. Dies hilft dabei:

- "Side Quests" zu vermeiden, die das Context Window verschmutzen
- Halluzinationen zu reduzieren
- Die Fehlerrate zu senken

Gist-Export:

Sessions können exportiert werden als:

- Durchsuchbare HTML-Dateien
- GitHub Private Gists

File-Tagging:

Ähnlich wie bei Cursor können Dateien getaggt werden, um das Context-Management zu verbessern und präzisere Antworten zu erhalten.

Bild-Unterstützung:

Bilder können durch Einfügen des Pfads verwendet werden (die Drag-and-Drop-Funktionalität ist laut Video-Ersteller etwas unklar dokumentiert).

Erweiterbarkeit: Das Pi-Ökosystem

Obwohl Pi minimalistisch ist, bietet es umfangreiche Erweiterungsmöglichkeiten:

Erweiterungstypen:

- **Tools**: Zusätzliche Werkzeuge
- **Commands**: Benutzerdefinierte Befehle
- **Shortcuts**: Tastenkürzel
- **Themes**: Visuelle Anpassungen
- **Prompts/Templates**: Slash-Befehle für häufige Aktionen

Prompt-Templates:

Unter dem Ordner `prompts` können eigene Slash-Befehle erstellt werden, z.B.:

- `/review` - Code auf Bugs und Sicherheitsprobleme prüfen

Packages installieren:

```
pi install <package-path>
```

Bekannte Erweiterungen:

- **Babysitter**: Reduziert Halluzinationen (verlangsamt aber den Prozess)
- **Sub-Agent-Erweiterungen**: Für diejenigen, die Sub-Agenten möchten
- Themes und Prompts auf der offiziellen Package-Website

Skills-Integration:

Pi erkennt automatisch bereits installierte Skills von Claude Code oder Open Code und macht diese verfügbar.

Inline-Ausführung und Aliase

One-Shot-Queries:

Mit `-p` kann Pi für einzelne Anfragen verwendet werden:

```
pi -p "Was belegt Port 8080?"
```

Modell-Liste anzeigen:

```
pi list models
```

Eigene Aliase erstellen:

```
alias q='pi -p --model <günstiges-modell>'
```

Dann einfach:

```
q "Wie spät ist es in Tokyo?"
```

Context ist König

Der Video-Ersteller betont wiederholt die Bedeutung des **Context Windows**:

> "Wenn es eine Sache gibt, auf die wir uns einigen können, dann ist es, dass Context König ist. Oder genauer gesagt: Context, sowohl seine Qualität als auch seine Größe, wird bestimmen, wie glücklich oder frustriert du sein wirst."

Probleme mit überladenem Context:

- Halluzinationen häufen sich
- Fehlerfrequenz steigt
- Agent wird "dümmer"

Pi's Lösungsansätze:

- Session-Branching und Forking
- Minimaler System-Prompt
- Keine automatische Session-Kompaktierung wie bei anderen Tools

---

Vergleich mit anderen Tools

Feature	Pi	Claude Code	Open Code
MCPs	(erweiterbar)		
Sub-Agenten	(erweiterbar)		
Plan-Modus			
Session-Kompaktierung			
Erweiterbarkeit	Hoch	Gering	⚠ Mittel
Footprint	Sehr klein	Groß	Mittel
Session-Branching		⚠	⚠

Praktische Anwendung des Video-Erstellers

Der Ersteller beschreibt seinen eigenen Workflow:

1. **Home-Server-Setup**: Pi läuft auf einem neuen Home-Server

2. **Kontextdatei**: Eine gut gestaltete MD-Datei mit Instruktionen und Context
3. **Soft Guardrails**: Eine Datei mit grundlegenden Richtlinien, um sich nicht ständig wiederholen zu müssen
4. **Tmux-Integration**: Statt Sub-Agenten werden Tmux-Sessions und Fork-Sessions verwendet

---

Zusätzliches Wissen über Pi

Aus der Community und Dokumentation:

1. **Awesome-Pi**: Es gibt eine community-gepflegte "Awesome Pi"-Seite mit Ressourcen und Erweiterungen
2. **Thorsten Ball's Artikel**: Der Ersteller referenziert einen Artikel, der zeigt, dass man mit nur ~400 Zeilen Code einen funktionierenden Agenten bauen kann - beschrieben als "Der Kaiser hat keine Kleider"
3. **Anerkennung von Experten**: Pi wird gelobt von:
   - Peter Steinberg (Ersteller von Open Claw)
   - Einem der Unity-Gründer
   - Dem Ersteller von Flask und Jinja (Armin Ronacher), der kürzlich den Entwickler hinter Pi eingestellt hat
4. **GitHub-Erfolg**: Pi wird als Agent beschrieben, der eines der am schnellsten wachsenden Projekte in der GitHub-Geschichte antreibt
5. **Philosophie der Einfachheit**:

   > "Alle diese Agenten sind im Grunde nur Schleifen über den Modellen."

Fazit des Video-Erstellers

Positiv:

- Schlank und schnell
- Einfach zu installieren
- Geringe Einstiegshürde
- Hohe Erweiterbarkeit bei Bedarf
- Session-Branching sehr nützlich
- Transparente Funktionsweise

Neutral/Abwägend:

- Noch keine SQLite-basierte Session-Verwaltung (JSON-Dateien)
- Bild-Unterstützung könnte klarer dokumentiert sein
- Für "Vibe Coding" und LinkedIn-Posts über Agent-Schwärme nicht geeignet

Endurteil:

> "Es ist noch zu früh, um es sicher zu sagen, aber Pi ist ein Keeper."

Der Ersteller betont, dass er trotz seiner Begeisterung für Open Code (mit all seinen Features) die Schlankheit von Pi sehr schätzt. Die Abwesenheit von Features ist nicht als Mangel zu verstehen, sondern als bewusste Designentscheidung, die Entwicklern ermöglicht, genau das hinzuzufügen, was sie brauchen - nicht mehr und nicht weniger.

Empfehlung

Pi eignet sich besonders für:

- Entwickler, die Kontrolle über ihre Tools schätzen
- Minimalisten, die Feature-Bloat vermeiden wollen
- Diejenigen, die ihre Agenten selbst erweitern möchten
- Terminal-Power-User
- Entwickler, die mit verschiedenen LLM-Providern arbeiten

Pi eignet sich weniger für:

- Entwickler, die "out of the box" alles haben wollen
- Diejenigen, die komplexe Multi-Agenten-Setups ohne eigene Konfiguration erwarten
- "Vibe Coders", die maximale Automatisierung suchen

Revision #1
Created 2026-05-31 15:56:23 UTC by art10m
Updated 2026-05-31 15:59:10 UTC by art10m