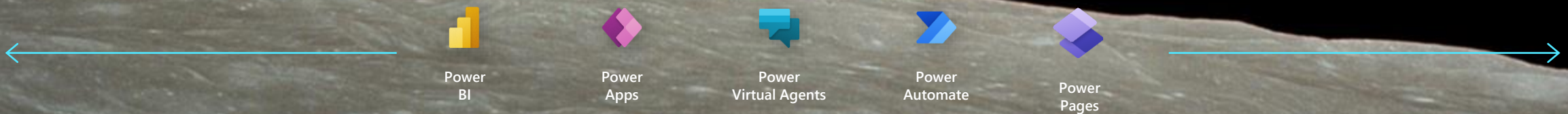


Das erste und einzige  
vollständig klimaneutrale  
Dynamics Community Event!

#DeutschlandPowerthon2022



# Aufzeichnung / Recording



**Keine Panik, meine Sessions wird  
nicht aufgezeichnet!**

**My session will not be recorded!**

# Sponsoren

## Platin



Dank unserer  
Sponsoren konnten  
wir **130** Tonnen Co2  
kompensieren

## Gold





Sven Sieverding

---

„Clean Code“ für Flows

Email: [sven@365knoten.de](mailto:sven@365knoten.de)

Twitter: [@365knoten](https://twitter.com/@365knoten)

Position/Rolle: [Consultant @ team-neusta](#)



# Agenda

- Kurz: Was ist Clean Code?
- Beispiel naiver Flow
- Besser lesbarer Flow
- Schnellerer Flow
- Subflows
- State Machines
- Dataflows



Code ist clean, wenn er...

lesbar

änderbar

erweiterbar

wartbar

ist

# Clean Code Prinzipien

DRY – Don't  
Repeat Yourself

KISS – Keep It  
Simple and Stupid

Vorsicht vor  
Optimierungen

FCoI – Favour  
Composition over  
Inheritance

SRP – Single  
Responsibility  
Principle

SLA – Single Level  
of Abstraction

Source Code  
Konventionen

SoC – Separation  
of Concerns

DIP – Dependency  
Inversion Principle

ISP – Interface  
Segregation  
Principle

POLA – Principle of  
Least  
Astonishment

LSP – Liskov  
Substitution  
Principle

OCP – Open Closed  
Principle

Information Hiding  
Principle

LoD – Law of  
Demeter

TDA – Tell, Don't  
Ask Principle

Entwurf und  
Implementation  
überlappen nicht

Implementation  
spiegelt Entwurf

YAGNI – You Ain't  
Gonna Need It

# Clean Code Prinzipien

DRY – Don't  
Repeat Yourself

KISS – Keep It  
Simple and Stupid

Vorsicht vor  
Optimierungen

FCoI – Favour  
Composition over  
Inheritance

SRP – Single  
Responsibility  
Principle

SLA – Single Level  
of Abstraction

Source Code  
Konventionen

SoC – Separation  
of Concerns

DIP – Dependency  
Inversion Principle

ISP – Interface  
Segregation  
Principle

POLA – Principle of  
Least  
Astonishment

LSP – Liskov  
Substitution  
Principle

OCP – Open Closed  
Principle

Information Hiding  
Principle

LoD – Law of  
Demeter

TDA – Tell, Don't  
Ask Principle

Entwurf und  
Implementation  
überlappen nicht

Implementation  
spiegelt Entwurf

YAGNI – You Ain't  
Gonna Need It





00 Ein Beispiel

# Herausforderung: Ausfuhrbeschränkungsflow

- Erstelle einen Flow,
  - der nächtlich einmal auf unsere Bestellschnittstelle zugreift und
  - für alle Bestellungen im Status „Vorbereitung“ mit wenigstens einem ausfuhrbeschränken Bauteil
  - eine Genehmigung vom Sachbearbeiter und Vorgesetzten einholt.
  - Wenn es keine Genehmigung gibt, soll eine Mail an die Zentrale geschickt werden, die dann die Bestellung händisch stoppt

Die Schnittstelle kann man nicht filtern, es kommt immer ein JSON mit den gesamten Tagesdaten zurück

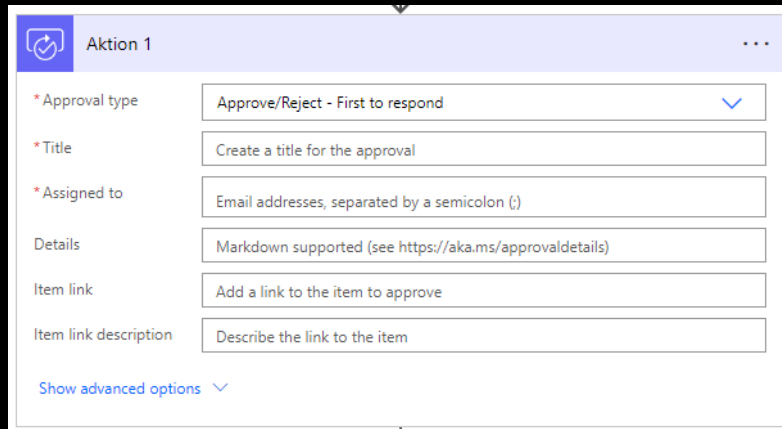
```
[
  {
    "Auftragsnummer": "DMDOLTW1",
    "Historie": [
      {
        "Status": "Bestellt",
        "Datum": "2022-08-24T20:39:35.964Z"
      },
      {
        "Status": "Vorbereitung",
        "Datum": "2022-09-11T16:13:25.099Z"
      }
    ],
    "Bauteile": [
      {
        "Anzahl": "12",
        "Name": "Fresh",
        "Bauteilnummer": "XFRIPNX1",
        "Hersteller": "Paucek - Pfeffer",
        "Ausfuhrbeschraenkt": true,
        "Einzelpreis": "181"
      },
      {
        ...
      }
    ]
  }
]
```



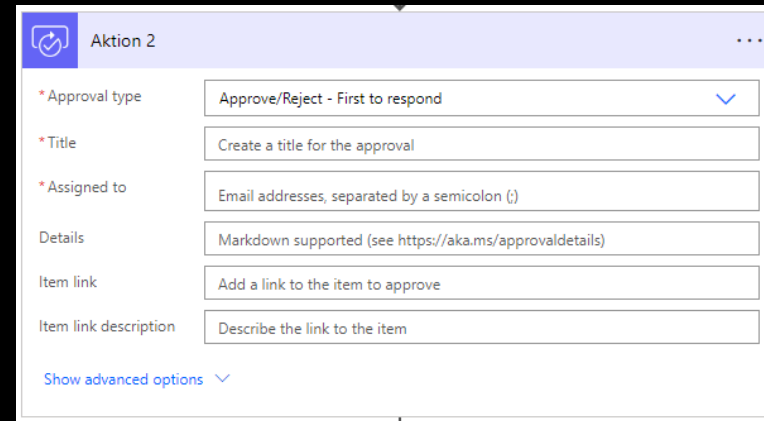
# 01 Navier Flow

# Lesbarkeit: Aktionen umbenennen

- Benenne Aktionen sprechend um
- Merke dir aber den Typ der Aktion
  - Entweder im Aktionsnamen
  - Oder in der Notiz



Create an Approval



Start and Wait for an  
Approval

# Lesbarkeit: Aktionen umbenennen

- Benenne Aktionen sprechend um
- Merke dir aber den Typ der Aktion
  - Entweder im Aktionsnamen
  - Oder in der Notiz

Aktion 1

\* Approval type: Approve/Reject - First to respond

\* Title: Create a title for the approval

\* Assigned to: Email addresses, separated by a semicolon (;)

Details: Markdown supported (see <https://aka.ms/approvaldetails>)

Item link: Add a link to the item to approve

Item link description: Describe the link to the item

Show advanced options

Start and Wait for an  
Approval

Aktion 2

\* Approval type: Approve/Reject - First to respond

\* Title: Create a title for the approval

\* Assigned to: Email addresses, separated by a semicolon (;)

Details: Markdown supported (see <https://aka.ms/approvaldetails>)

Item link: Add a link to the item to approve

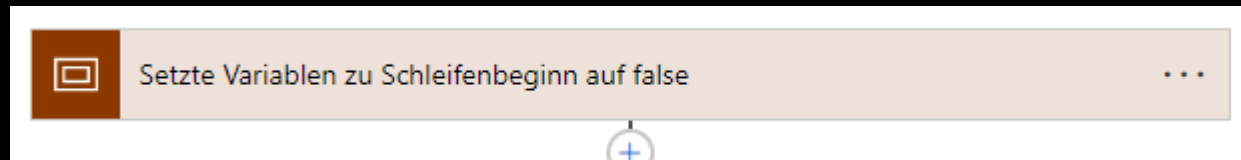
Item link description: Describe the link to the item

Show advanced options

Create an Approval

# Lesbarkeit: Scopes

- Benutze Scopes, um zusammengehörende Funktionen zu gruppieren
  - Um mehr Kontextinformationen bereitzustellen
  - Um viele gleiche Aktionen logisch zusammenzufassen



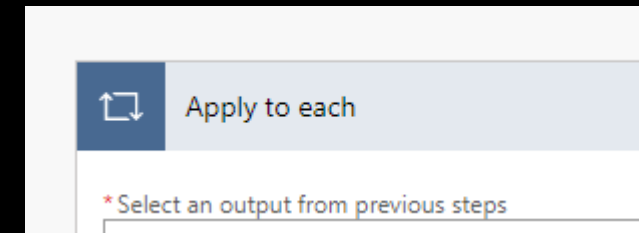




02 Besser lesbar

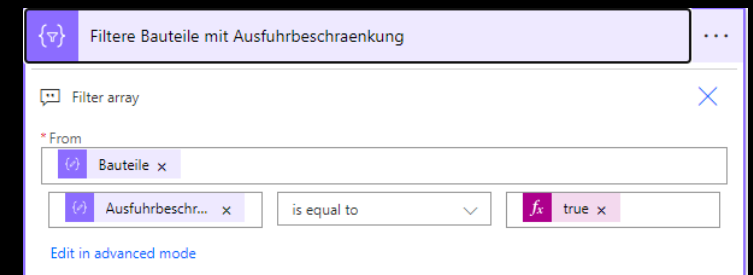
# Lesbar/wartbar: Filtern statt Schleifen

- Schleifen sind langsam und sind strukturell schwierig zu ändern
- Schleifen fügen eine „Ebene“ hinzu und machen den Flow visuell „unordentlich“
- Parallelität in Schleifen kann zu Fehlern führen



Vermeide Schleifen, wenn möglich

- Benutze z.B. Filter & Formeln, um mit Listen/Arrays umzugehen



```
not(equals(length(body('Filtere_Bauteile_mit_Ausfuhrbeschränkung')),0))
```

# Lesbar/wartbar: Filtern statt Schleifen

- Schleifen sind langsam

Flow „02 Besser lesbar“

Laufzeit ca. 4min 30 sec

Flow „03 Filtered“

Laufzeit ca. 14 sec


# Generelle Hinweise zur Struktur


- So wenig Verschachtelung wie möglich
  - Muss das immer eine Schleife/Condition/Switch sein?
    - => Führt häufig zu unübersichtlicher/später schwer änderbarer Struktur
    - => Führt häufig zu Duplizierung von anderen Aktionen. (Mailaktion im Ja- und Nein Fall)
  - Geschickte Formeln/alternative Strukturen haben häufig denselben Effekt
- So wenig Aktionen wie möglich und nötig
  - Nicht übertreiben: Ziel ist eine einfachere Lesbarkeit
  - Anzahl der Aktionsausführungen in der Lizenz ist limitiert
  - Murphy's Law: Alle Flows haben Fehler, aber große Flows haben ggf. große Fehler


# Lesbar/erweiterbar: Variablen


- Benutze so wenig Variablen wie nötig
  - Variablen müssen in einer Aktion definiert werden und mit wenigstens einer Aktion gesetzt werden. Ist das immer nötig?
  - Variablen sind global. Welche Seiteneffekte erkaufe ich mir durch den Einsatz?
  - Variablen und Schleifen passen nicht gut zusammen.
    - Parallelität der Schleifenabarbeitung muss ggf. auf 1 gesetzt werden.

# Lesbar/erweiterbar: JSON als Config

 Konfiguration initialisieren

 ...

 Parse JSON



\* Content

```
{
  "SharePointUrl": "https://365knoten.sharepoint.com/sites/MyAppPage",
  "SharePointListName": "StateMachine",
  "SharePointItemLimit": 10
}
```


\* Schema


```
{
  "type": "object",
  "properties": {
    "SharePointUrl": {
      "type": "string"
    },
    "SharePointListName": {
      "type": "string"
    },
    "SharePointItemLimit": {


```


Generate from sample

Konfiguration initialisieren

 Body

 SharePointUrl

 SharePointListName

 SharePointItemLimit



03 Filtered



# Lesbar/wartbar: Guard Clauses

```
function myfunction(myvar) {  
  var result = null;  
  
  if (myvar == null) {  
    result = "leer";  
  } else if (myvar == "Wert") {  
    result = "Wert";  
  } else {  
    result = "kein Wert";  
  }  
  
  return result;  
}
```

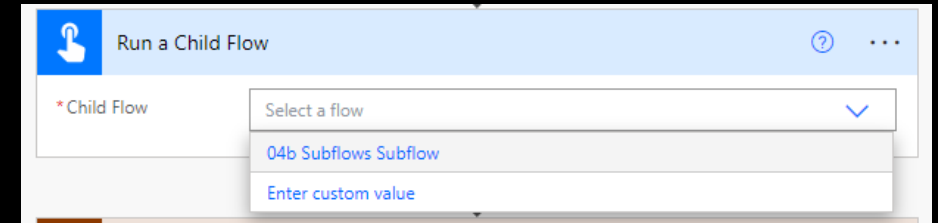


```
function myGuardedFunction(myvar) {  
  
  if (myvar == null) return "leer";  
  if (myvar == "Wert") return "Wert";  
  
  return "kein Wert"  
}
```

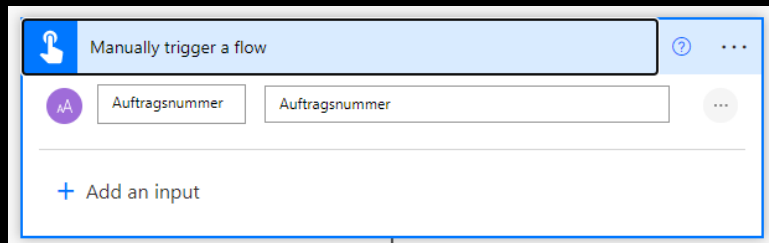
# Sub Flows - Seperation of Concerns

- Flow in Solutions haben die Aktion „Run a Child Flow“
- Können wie Funktionen benutzt werden

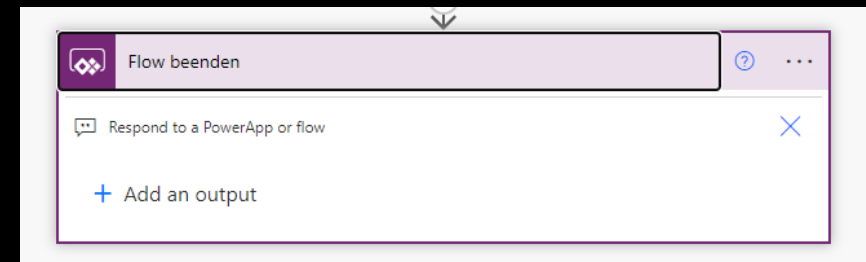
Parent: Run a Child Flow



Child: Trigger mit Parametern



Child: Parameter an Parent zurückgeben

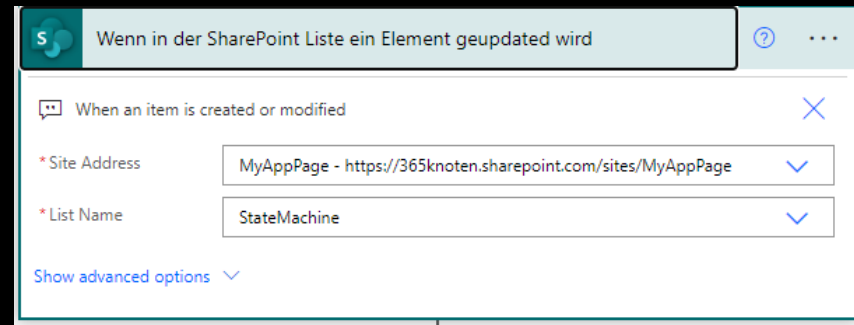
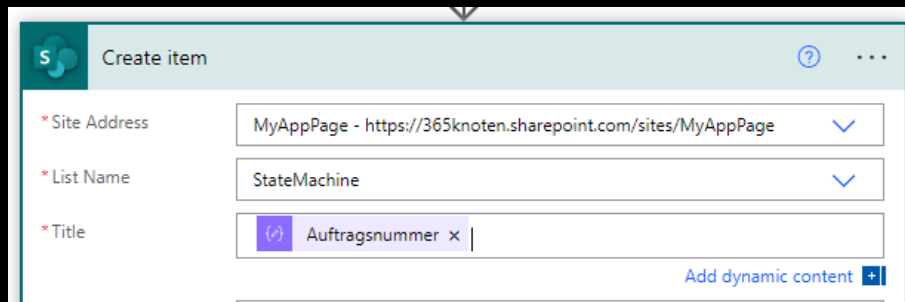
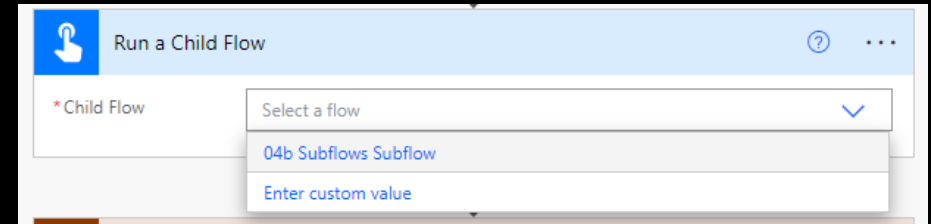




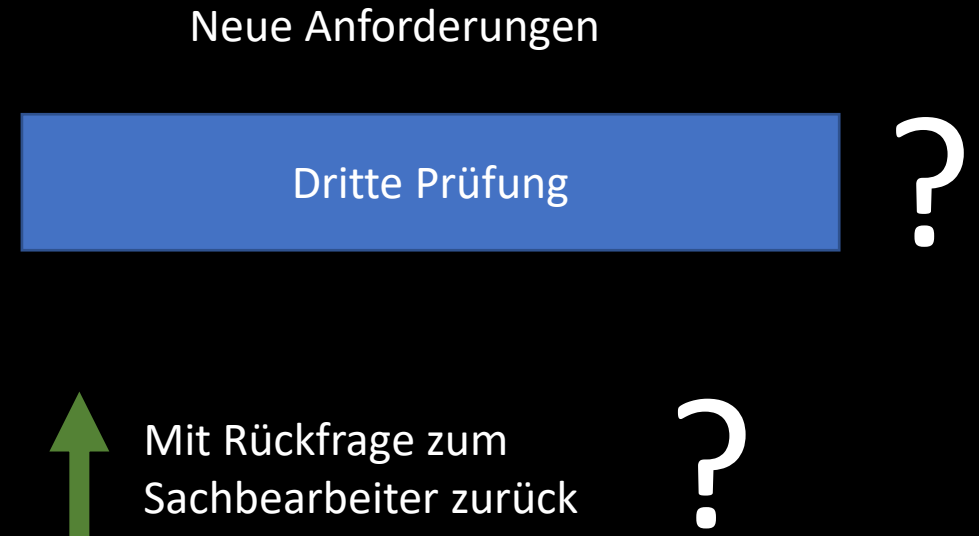
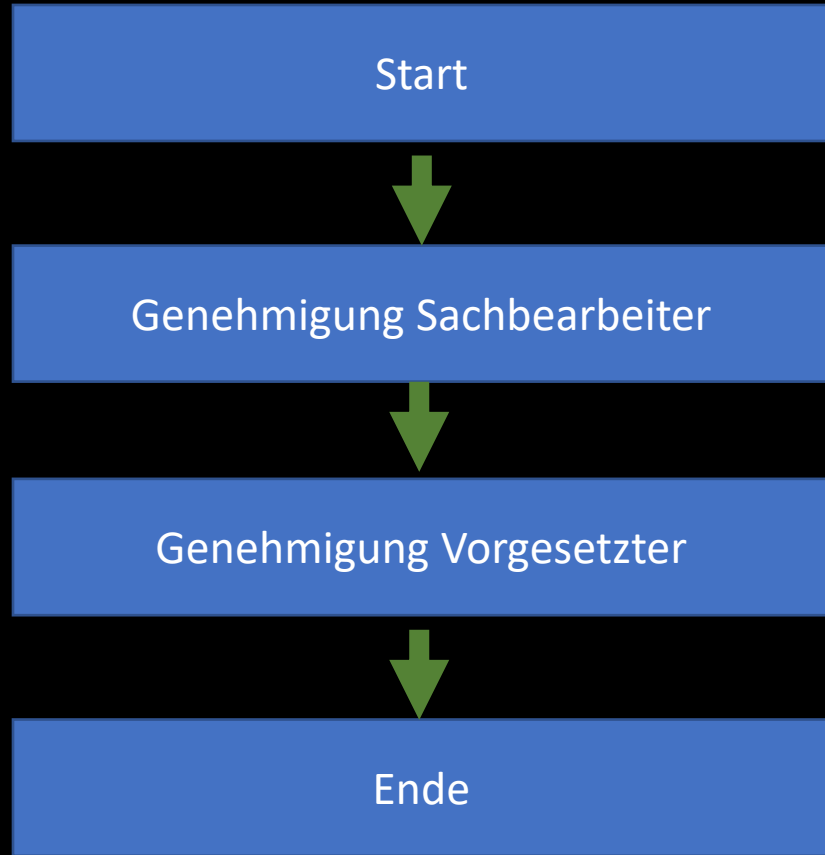
## 04 Subflows

# De-Coupling: Externe Tabelle

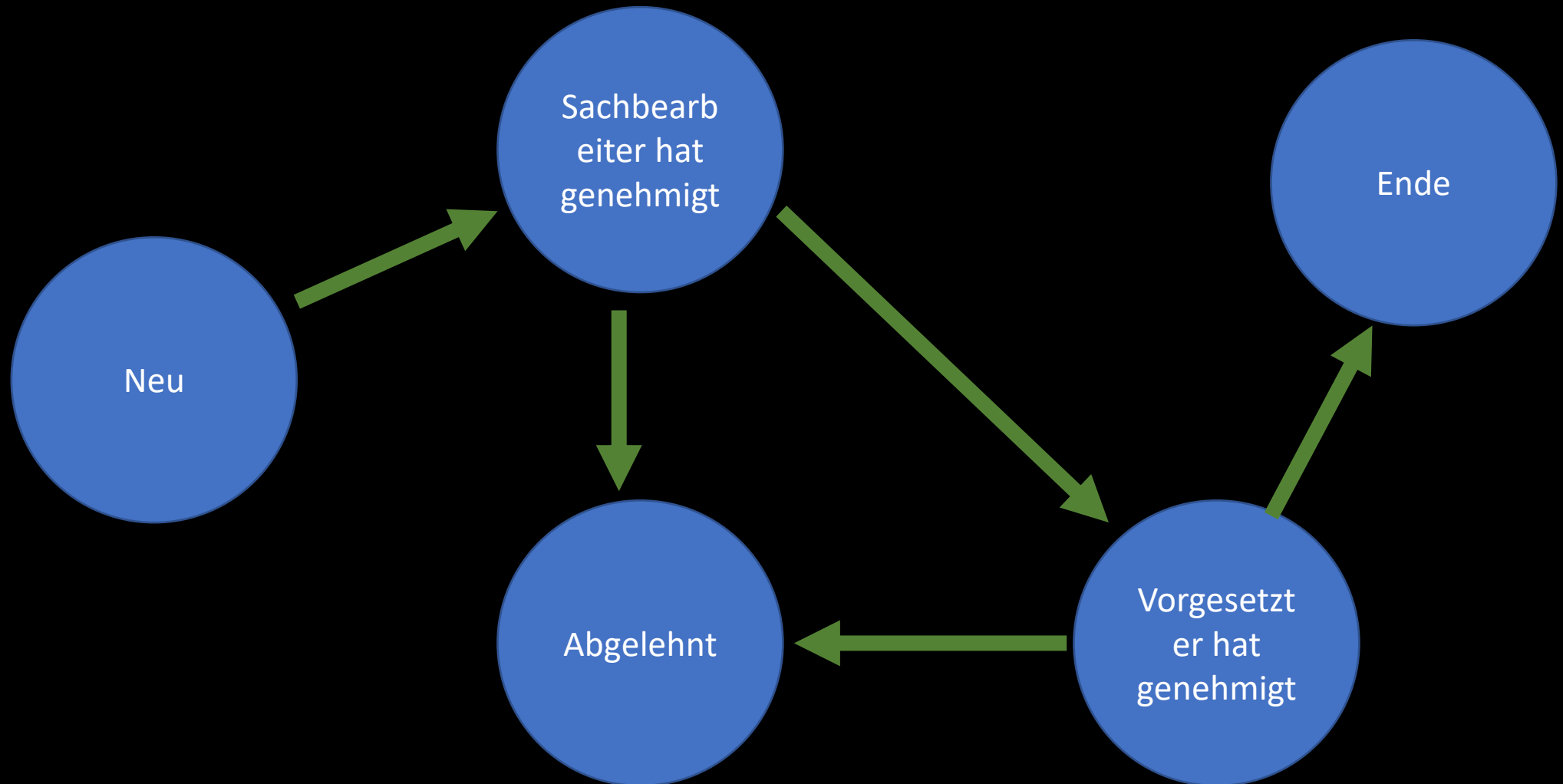
- „Run a Child Flow“ ist synchron:  
Parent Flow wartet auf Child Flow
- Asynchrones Fire&Forget z.B. mit SharePoint Liste/Dataverse Table



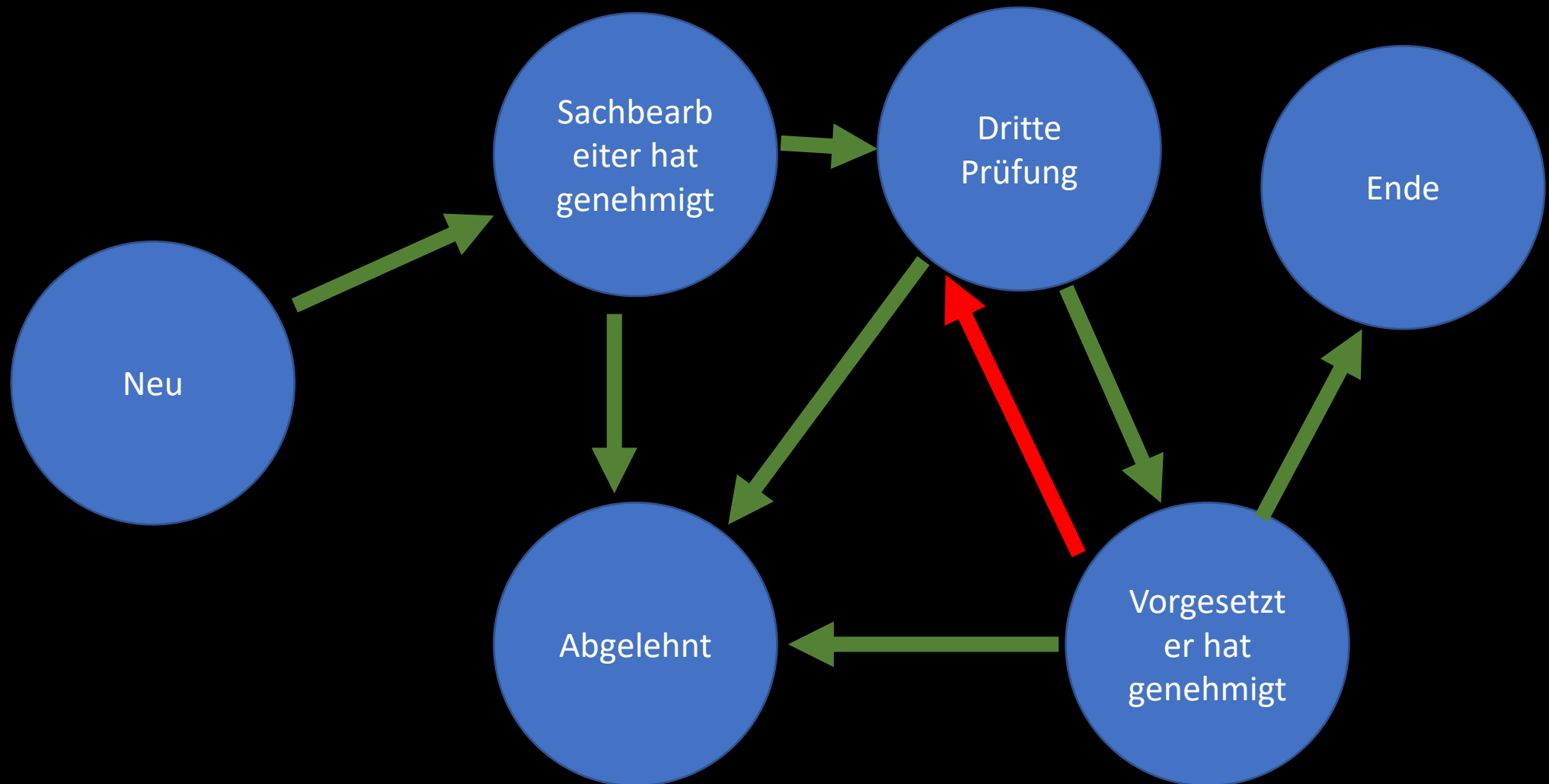
# Sequentielle Flows



# State Machine



# State Machine: New State



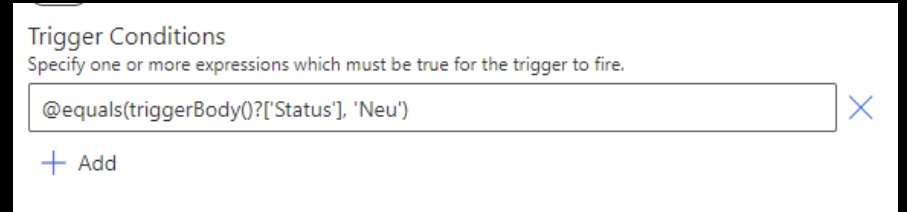




# 05 State Machines

# Seperation of Concern: Trigger Conditions

- Besser: Statt eines Flows mehrere Flows in der State Machine  
=> Ein Flow pro State



Trigger Conditions

Specify one or more expressions which must be true for the trigger to fire.

@equals(triggerBody()?['Status'], 'Neu')

+ Add

- Trigger Condition am Auslöser
  - Starte diesen Flow nur, wenn eine bestimmte Bedingung gegeben ist



# 06 State Machines Single States

# Erweiterbar: Dataflows

- Ist ein Powerautomate-Flow immer die beste Wahl?
  - Gibt es ggf. bessere Alternativen (im PowerPlattform-Umfeld?)
- Dataflows
  - Importiere Daten ins Dataverse mittels PowerQuery
  - Sehr, sehr effizient
  - Nur unter <https://make.powerapps.com/>, nicht unter <https://make.powerautomate.com/>

# Dataflow: Formular

- `let`
- `Source = Json.Document(Web.Contents("https://365knoten-my.sharepoint.com/personal/sven_365knoten_onmicrosoft.com/Documents/Clean%20Code%20f%C3%BCr%20Flows/data.json"))`,
- `"Converted to table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error)`,
- `"Expanded Column1" = Table.ExpandRecordColumn("Converted to table", "Column1", {"Auftragsnummer", "Erfassungsdatum", "Aenderungsdatum", "Historie", "Bauteile"}, {"Auftragsnummer", "Erfassungsdatum", "Aenderungsdatum", "Historie", "Bauteile"})`,
- `"Changed column type" = Table.TransformColumnTypes("Expanded Column1", {"Auftragsnummer", type text}, {"Erfassungsdatum", type datetime}, {"Aenderungsdatum", type datetime}, {"Historie", type any}, {"Bauteile", type any}))`,
- `"Added custom" = Table.AddColumn("Changed column type", "InStateVorbereitung", each Table.Contains(Table.FromRecords([Historie]), [Status = "Vorbereitung"]) and not Table.Contains(Table.FromRecords([Historie]), [Status = "Verpackt"]) and not Table.Contains(Table.FromRecords([Historie]), [Status = "Versandt"])))`,
- `"Added custom 1" = Table.AddColumn("Added custom", "AusfuhrbeschraenktesBauteil", each Table.Contains(Table.FromRecords([Bauteile]), [Ausfuhrbeschraenkt = true]))`,
- `"Added custom 2" = Table.AddColumn("Added custom 1", "RelevanterDatensatz", each [InStateVorbereitung] and [AusfuhrbeschraenktesBauteil])`,
- `"Inserted conditional column" = Table.AddColumn("Added custom 2", "State", each if [RelevanterDatensatz] = true then "Neu" else "")`,
- `"Filtered rows" = Table.SelectRows("Inserted conditional column", each ([State] = "Neu"))`,
- `"Removed columns" = Table.RemoveColumns("Filtered rows", {"Erfassungsdatum", "Aenderungsdatum", "Historie", "Bauteile", "InStateVorbereitung", "AusfuhrbeschraenktesBauteil", "RelevanterDatensatz"})`
- `in`
- `"Removed columns"`



## 07 Dataflows

# Zusammenfassung

- Am Anfang:
  - Schwer anpassbarer monolithischer Flow (4 Bildschirmseiten)
  - Schwer zu erfassen, was der Flow eigentlich macht
  - Laufzeit Datenladen: 4min 30sec
- Am Ende
  - Ein LadeFlow und 4 einfache State Flow
  - Laufzeit Datenladen: 14 sec
  - Jeder Flow nicht länger als eine Bildschirmseite
  - Jeder Flow ist strukturell klar und es ist einfach zu erfassen, was er tut
  - Einfach um neue Funktionalitäten erweiterbar



Vielen Dank  
für die  
Aufmerksamkeit

