Symcon Module Validator

Fehlende $-Abhängigkeit verhindert jede Validierung
Reproduktion, Ursachenanalyse und robuster Korrekturvorschlag

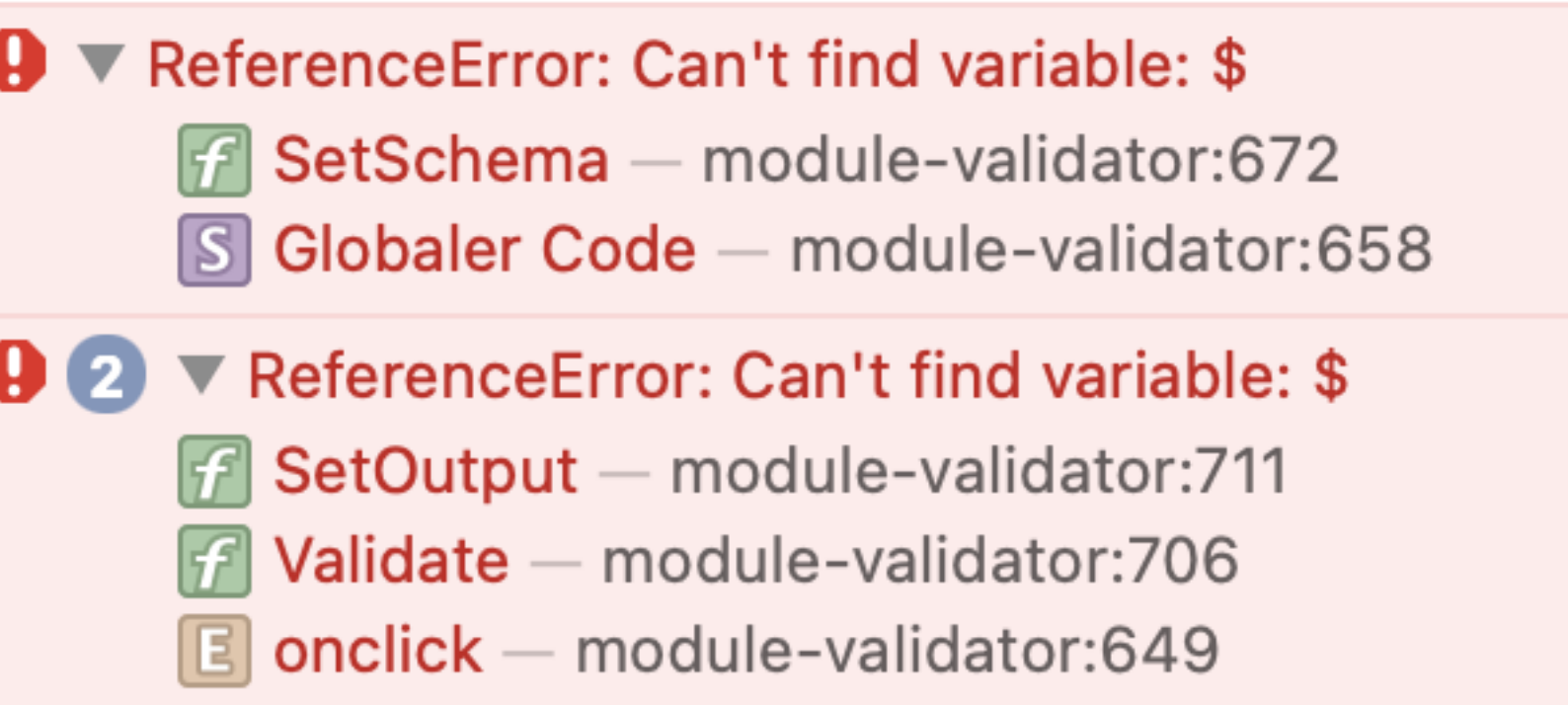
Status	Reproduzierbarer Laufzeitfehler
Betroffene Seite	Offizieller Symcon Module Validator
Stand	29.07.2026
Abgrenzung	Die geprüften Metadaten bestehen die offiziellen Schemas mit AJV 6.10.2; der Defekt liegt vor der fachlichen Prüfung.

1. Kurzfassung

Der ausgelieferte Validator-Code verwendet in allen vier Validatorfunktionen jQuery-Notation ($). Im aktuellen Dokument wird AJV 6.10.2 geladen, jedoch keine jQuery-Bibliothek. Deshalb bricht die Seite ab, bevor eine Eingabe gegen ein Schema geprüft werden kann.

2. Reproduktion

- Validator-Seite öffnen.
- Dateityp auswählen, zum Beispiel `library.json`.
- Gültiges JSON in das Feld „Inhalt“ einsetzen.
- „Validieren“ anklicken.



Vom Benutzer unabhängig reproduzierter Konsolenfehler. SetSchema: Zeile 672; SetOutput: Zeile 711.

Ist- und Soll-Ergebnis

Ist	Safari: `Can't find variable: $`; Chromium: `$ is not defined`. Kein fachliches Ergebnis.
Soll	Gültigkeitsmeldung oder konkrete Schemafehler für die ausgewählte JSON-Datei.

3. Technische Ursache

Die sichtbaren Stacktraces zeigen nur zwei Stellen. Tatsächlich verwenden alle vier Funktionen die fehlende Abhängigkeit:

Funktion	Aktuelle Zugriffe
LoadData	`$('#sourcefile'), $('#inputbox')`
SetSchema	`$('#schematype')`
Validate	`$('#schematype'), $('#inputbox')`
SetOutput	`$('#validationresulttitle'), $('#validationresult')`

Das bloße Reparieren von `SetSchema` und `SetOutput` wäre unvollständig. Dass eine frühere jQuery-Abhängigkeit bei einer Website-Überarbeitung entfallen ist, ist eine plausible Schlussfolgerung aus dem aktuellen HTML, keine Aussage zur Änderungshistorie.

4. Empfohlene Korrektur

Bevorzugt: die wenigen jQuery-Aufrufe durch native DOM-APIs ersetzen. Ein kurzfristiger Hotfix könnte jQuery wieder einbinden, würde aber eine unnötige Abhängigkeit konservieren.

```
$('#schematype').val()        -> element('schematype').value
$('#inputbox').val()          -> element('inputbox').value
$('#sourcefile')[0].files[0]  -> element('sourcefile').files[0]
.html(text)                   -> .textContent = text
.css('color', color)          -> .style.color = color
.attr('hidden', value)        -> .hidden = Boolean(value)
```

Robustheitskorrekturen

- Schema-Download vor der Validierung zuverlässig abwarten.
- HTTP-, Schema-, AJV- und Eingabe-JSON-Fehler getrennt melden.
- Ergebnisse mit `textContent` statt `innerHTML` ausgeben.
- `errorsText(..., { separator: '\n' })` und `/data\./g` verwenden.
- Bei fehlender Upload-Datei kontrolliert abbrechen.

Kompakter robuster Kern

```
let schemaPromise;
const element = id => document.getElementById(id);

function SetSchema() {
  const type = element('schematype').value;
  const url = `/assets/files/validation/${type}Schema.json`;
  schemaPromise = fetch(url, { cache: 'no-cache' })
    .then(response => {
      if (!response.ok) throw new Error(`HTTP ${response.status}`);
      return response.text();
    })
    .catch(error => {
      SetOutput('Das Validator-Schema konnte nicht geladen werden.',
        'red', error.message, false, false);
      return null;
    });
  return schemaPromise;
}

async function Validate() {
  const type = element('schematype').value;
  const inputText = element('inputbox').value;
  if (inputText.trim() === '') {
    SetOutput('', '', '', true, true);
    return;
  }

  let inputData;
  try {
    inputData = JSON.parse(inputText);
  } catch {
    SetOutput(`Die ${type}.json beinhaltet kein gueltiges JSON.`,
      'red', '', false, true);
    return;
  }

  const schemaText = await (schemaPromise || SetSchema());
  if (schemaText === null) return;

  try {
    if (typeof Ajv !== 'function') {
      throw new Error('AJV wurde nicht geladen');
    }
    const ajv = new Ajv({ allErrors: true });
    const validate = ajv.compile(JSON.parse(schemaText));
    if (validate(inputData)) {
      SetOutput(`Die ${type}.json ist gueltig.`,
        'white', '', false, true);
      return;
    }
    const errors = ajv
      .errorsText(validate.errors, { separator: '\n' })
      .replace(/data\./g, `${type} `);
    SetOutput('Die Datei ist fehlerhaft:',
      'red', errors, false, false);
  } catch (error) {
    SetOutput('Der Validator konnte die Pruefung nicht ausfuehren.',
      'red', error.message, false, false);
  }
}
```

Ausgabe ohne jQuery

```
function SetOutput(
  titleText, titleColor, resultText, titleHidden, resultHidden
) {
  const title = element('validationresulttitle');
  const result = element('validationresult');
  title.textContent = titleText;
  title.style.color = titleColor;
  title.hidden = Boolean(titleHidden);
  result.textContent = resultText;
  result.style.whiteSpace = 'pre-line';
  result.hidden = Boolean(resultHidden);
}
```

Dateiauswahl ohne jQuery

```
function LoadData() {
  const file = element('sourcefile').files[0];
  if (!file) return;

  const reader = new FileReader();
  reader.onload = () => {
    element('inputbox').value = reader.result;
  };
  reader.onerror = () => {
    SetOutput('Die Datei konnte nicht gelesen werden.',
      'red', '', false, true);
  };
  reader.readAsText(file, 'UTF-8');
}

// Nach den Funktionsdefinitionen zum Vorladen:
SetSchema();
```

5. Akzeptanzkriterien

- Initialer Seitenaufruf und Dateitypwechsel erzeugen keinen Konsolenfehler.
- library.json, module.json, locale.json und form.json laden ihr Schema.
- Texteingabe und Dateiauswahl funktionieren in Safari, Chromium und Firefox.
- Gültige, syntaktisch ungültige und schemawidrige Eingaben werden unterschieden.
- Schema-, Netzwerk- und AJV-Fehler werden als Validatorfehler ausgewiesen.
- Keine `$ is not defined`- oder `Can't find variable: $`-Fehler.

6. Gegenprobe

Die im konkreten Test verwendete `library.json` und vier `module.json` bestanden separat gegen die offiziellen Symcon-Schemas mit derselben AJV-Version 6.10.2. Das belegt nicht die Funktionsfähigkeit der Weboberfläche, grenzt den Defekt aber auf deren JavaScript- und Abhängigkeitsschicht ein.

7. Quellen

Bestehender Forumbeitrag:
https://community.symcon.de/t/php-module-json-dateien-mit-dem-module-validator-ueberpruefen/51071

Betroffener Module Validator:
https://www.symcon.de/de/service/dokumentation/entwicklerbereich/sdk-tools/tools/module-validator/

Offizielle Tool-Übersicht:
https://www.symcon.de/de/service/dokumentation/entwicklerbereich/sdk-tools/tools/

> Hinweis: Dieser Korrekturvorschlag wurde aus dem aktuell öffentlich ausgelieferten Seitencode und reproduzierbaren Browserfehlern abgeleitet. Es wurden keine privaten Installationsdaten verwendet.