Datei

In ROBO Pro Coding werden Ordner als spezielle Arten von Dateien betrachtet, die zur Organisation und Strukturierung von Projektdaten dienen. Daher werden in diesem Abschnitt sowohl Dateien als auch Ordner behandelt.

Dateien

Datei öffnen

Der Block "Datei ... mit Modus ... öffnen" ermöglicht das Öffnen einer Datei in einem ausgewählten Modus. Dieser Block ist in zwei Varianten verfügbar:

- Dropdown-Auswahl: Diese Variante zeigt alle vorhandenen Dateien im data-Verzeichnis des Projektordners im Dropdown-Menü an.
- Freie Eingabe: Diese Option erlaubt, den Namen einer Datei direkt einzugeben. Diese Flexibilität ermöglicht das Erstellen einer neuen Datei, falls diese beim Schreiben nicht bereits existiert. Gleichzeitig ermöglicht es den Zugriff auf bereits bestehende Dateien. Dies ist besonders vorteilhaft für dynamisches Arbeiten mit verschiedenen Dateinamen, die nicht im Voraus bekannt sind.

Mögliche Modi zum Öffnen sind:

- Lesen: Öffnet die Datei zum Lesen von Inhalten.
- Schreiben: Öffnet die Datei, um Inhalte zu überschreiben.
- Anhängen: Öffnet die Datei, um Inhalte am Ende der Datei anzuhängen, ohne bestehende Inhalte zu überschreiben.

Bei Ausführung gibt der Block einen Dateideskriptor zurück, der in nachfolgenden Befehlen genutzt werden kann, um die Datei zu lesen, zu schreiben oder Daten anzuhängen.

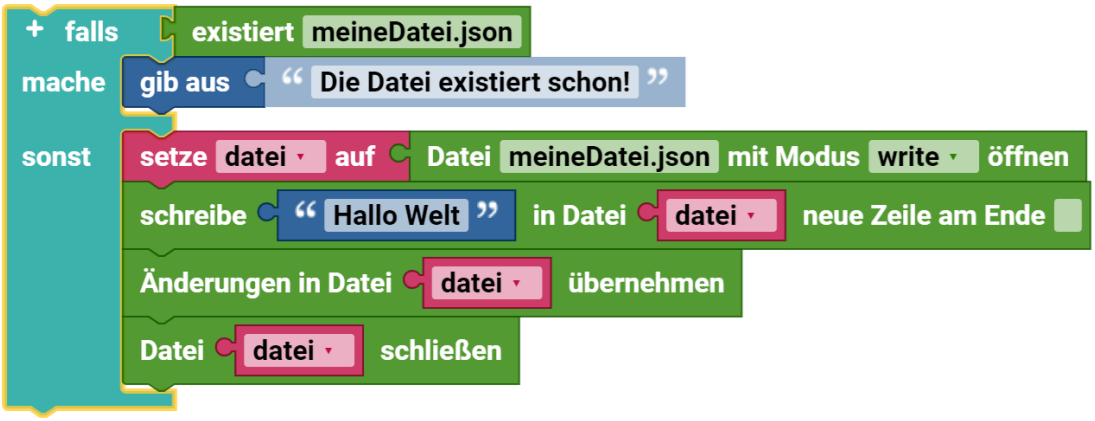
Das folgende Beispiel illustriert die Verwendung des Blocks, um die Datei meineDatei.json im Lesemodus zu öffnen, die sich im data-Verzeichnis befindet. Der Dateideskriptor wird in der Variable datei gespeichert:

Überprüfung der Existenz einer Datei

Der Block "existiert" ermöglicht es zu prüfen, ob eine spezifische Datei bereits existiert. Auf Basis dieser Prüfung können unterschiedliche Aktionen ausgeführt werden.

In dem Beispiel wird der "existiert"-Block genutzt, um zu prüfen, ob die Datei bereits vorhanden ist:

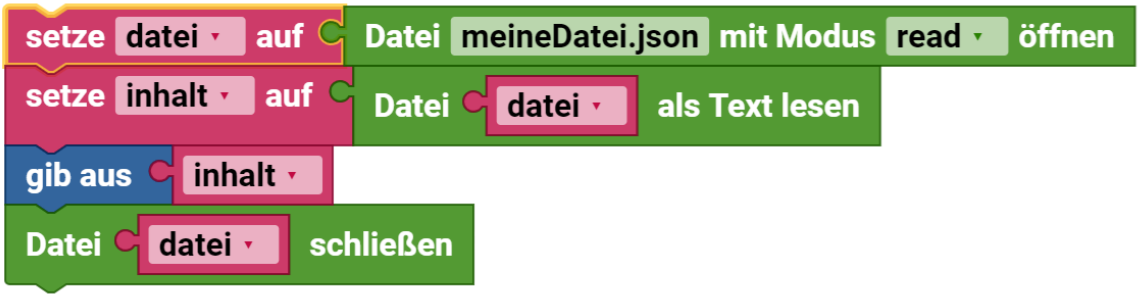
- Wenn die Datei existiert: Der Controller gibt die Nachricht „Die Datei existiert schon!“ aus.
- Wenn die Datei nicht existiert: Die Datei meineDatei.json wird erstellt und geöffnet. Zunächst wird ein Dateideskriptor erstellt, der in den folgenden Schritten verwendet wird, um auf die Datei zuzugreifen. Anschließend wird der Text „Hallo Welt“ in die Datei geschrieben und die vorgenommenen Änderungen werden gespeichert. Schließlich wird der Dateideskriptor geschlossen, um die Datei ordnungsgemäß zu schließen und Ressourcen freizugeben.



Datei als Text einlesen

Der Block "Datei ... als Text lesen" ermöglicht es, den Inhalt einer Datei einzulesen. Dieser Prozess ist besonders nützlich, um Daten aus einer Datei zu extrahieren und sie weiterzuverarbeiten.

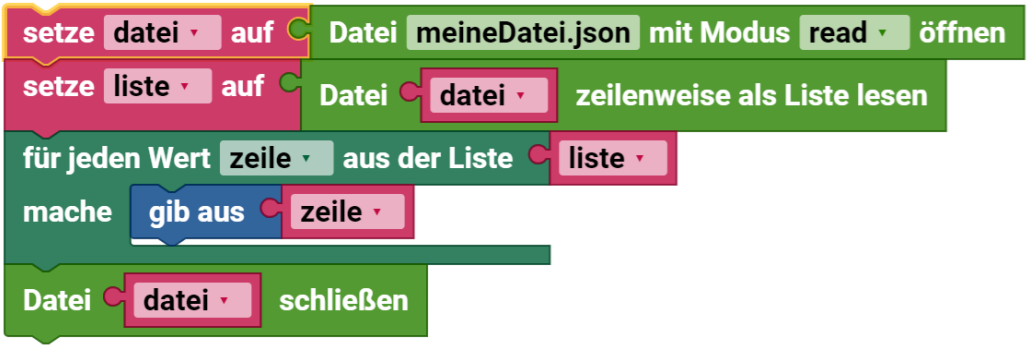
Im Beispiel wird die Datei zunächst im Lesemodus geöffnet, dann wird der Inhalt der Datei in eine Variable namens inhalt geladen. Anschließend wird dieser Inhalt ausgegeben. Schließlich wird die Datei wieder geschlossen, um die Ressourcen ordnungsgemäß freizugeben.



Datei zeilenweise als Liste einlesen

Der Block "Datei ... zeilenweise als Liste lesen" ermöglicht es, den Inhalt einer Datei zeilenweise in Form einer Liste einzulesen, wobei jedes Listenelement eine Zeile repräsentiert. Dieser Prozess ist besonders nützlich, um die Zeilen aus einer Datei zu extrahieren und sie weiterzuverarbeiten.

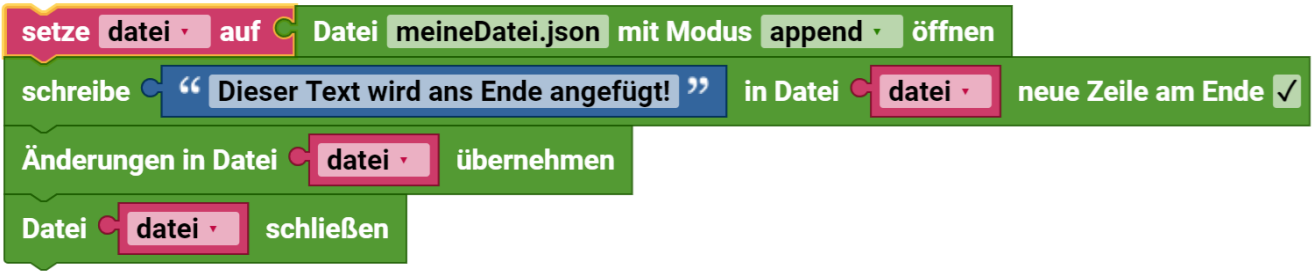
Im dargestellten Beispiel wird der Prozess zur Verwendung dieses Blocks demonstriert. Zunächst wird die Datei meineDatei.json im Lesemodus (read) geöffnet. Der Inhalt der Datei wird dann zeilenweise gelesen und in einer Liste gespeichert. Jede Zeile der Datei wird als ein separates Element in der Liste liste gespeichert. Es folgt eine Schleife, in der jedes Element der Liste (jede Zeile) durchlaufen und mittels "gib aus" ausgegeben wird. Nachdem alle Zeilen verarbeitet wurden, wird die Datei geschlossen, um die Ressourcen freizugeben.



Schreiben in einer Datei

Der Block "Schreibe ... in Datei ... neue Zeile am Ende" ermöglicht das Schreiben von Text in eine Datei. Optional kann ein Zeilenumbruch (\n) am Ende des Textes eingefügt werden, um die Lesbarkeit bei mehreren Schreibvorgängen zu verbessern.

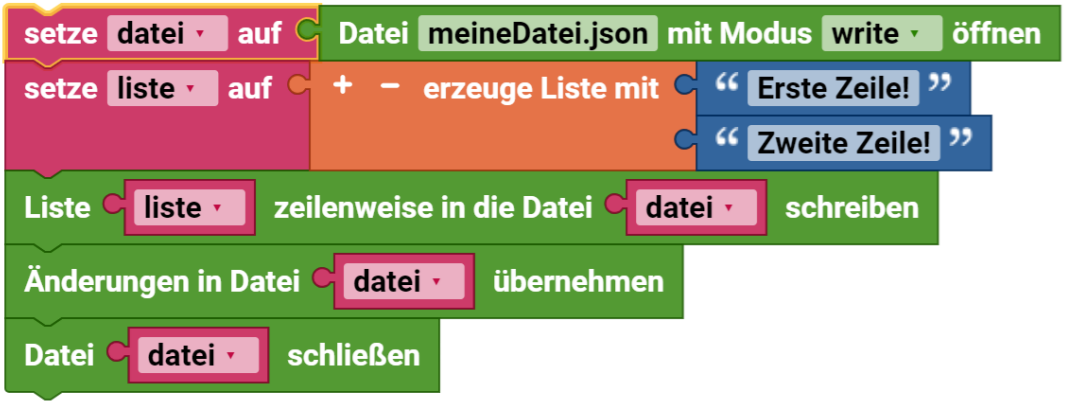
Im Beispiel wird die Datei meineDatei.json im Anhäng-Modus (append) geöffnet. Anschließend wird der Text „Hallo Welt“ in die Datei geschrieben. Dies führt dazu, dass der Inhalt an das Ende der bestehenden Daten in der Datei angehängt wird, ohne die vorhandenen Inhalte zu überschreiben. Nach dem Schreiben werden die Änderungen gespeichert und die Datei geschlossen.



Liste zeilenweise in die Datei schreiben

Der Block "Liste ... zeilenweise in die Datei ... schreiben" ermöglicht das Schreiben einer Liste in eine Datei, wobei jedes Element der Liste in eine eigene Zeile geschrieben wird. Dies ist besonders nützlich für ein strukturiertes Speichern von Daten, bei denen jedes Listenelement klar von den anderen getrennt ist.

Im dargestellten Beispiel wird die Datei meineDatei.json zunächst im Schreibmodus geöffnet. Dies bedeutet, dass, falls die Datei bereits vorhanden ist, alle existierenden Inhalte überschrieben werden. Anschließend wird eine Liste mit den Elementen "Erste Zeile!" und "Zweite Zeile!" erstellt. Diese Elemente werden dann zeilenweise in die Datei geschrieben, sodass jede Zeile der Datei einem Element der Liste entspricht. Nachdem die Liste in die Datei geschrieben wurde, werden die Änderungen übernommen, um die neu geschriebenen Daten zu sichern. Zum Abschluss des Prozesses wird die Datei geschlossen, um den Zugriff ordnungsgemäß zu beenden.



Änderungen in die Datei schreiben

Dieser Block "Änderungen in Datei ... übernehmen" stellt sicher, dass alle Änderungen, die an der geöffneten Datei vorgenommen wurden, auch tatsächlich gespeichert werden. Er führt im Wesentlichen einen Flush-Vorgang aus, bei dem alle bislang gepufferten Daten in den permanenten Speicher der Datei geschrieben werden. Nachdem Inhalte in die Datei geschrieben oder modifiziert wurden, sollte dieser Block verwendet werden, um die Änderungen festzuschreiben.

Datei schließen

Der Block "Datei ... schließen" wird verwendet, um eine geöffnete Datei ordnungsgemäß zu schließen.

Ordner

Ordner erstellen

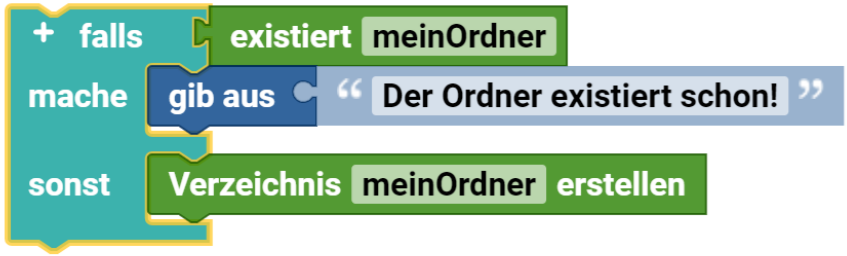
Der Block "Verzeichnis ... erstellen" ermöglicht es, ein neues Verzeichnis zu erstellen, falls es bisher nicht existiert.

Überprüfung der Existenz eines Ordners

Der Block "existiert" ermöglicht es zu prüfen, ob ein spezifischer Ordner bereits existiert. Auf Basis dieser Prüfung können unterschiedliche Aktionen ausgeführt werden.

In dem Beispiel wird der "existiert"-Block genutzt, um zu prüfen, ob der Ordner bereits vorhanden ist:

- Wenn der Ordner existiert: Der Controller gibt die Nachricht „Der Ordner existiert schon!“ aus.
- Wenn der Ordner nicht existiert: Der Block "Verzeichnis meinOrdner erstellen" wird ausgeführt, um den Ordner zu erstellen.



Revision #24
Created 8 December 2022 07:05:08 by Alexander Steiger
Updated 15 July 2024 13:21:13 by phuesing