Kommunikationstechnik Teil 2 Datenformate

DI (FH) Andreas Pötscher

HTL Litec

Figure 1: JavaScript Logo

Einfachen Daten werden in IT Systemen als primitive Datentypen dargestellt. Diese können je nach Datentyp verschiedene Werte enthalten. In dieser Unterlage wird dafür die Programmiersprache JavaScript verwendet. Auf dieser basiert auch Node-Red, das für die Übungsbeispiele verwendet wird.

In JS (JavaScript) gibt es folgende primitive Datentypen:

- **Number**: Für numerische Werte, sowohl Ganzzahlen als auch Gleitkommazahlen.
- **String**: Für Zeichenketten, also Text.
- **Boolean**: Für Wahrheitswerte `true` oder `false`.
- **Null**: Wert, der bewusst als Abwesenheit von einem Wert verwendet wird.

Die Syntax von JS basiert auf der C Syntax.

Dynamische Typisierung

Im großen Unterschied zu C verwendet JavaScript keine strenge Typisierung sondern eine dynamische Typisierung, das sogenannte Duck Typing. Duck Typing ist ein Konzept in JavaScript, bei dem der Datentyp eines Objekts oder einer Variable nicht aufgrund der Initialisierung, sondern aufgrund seines Verhaltens oder seiner Eigenschaften bestimmt wird.

Link zum Beispiel

Kontrollstrukturen in JS

Bei den Kontrollstrukturen gibt es im weitesten Sinne die gleichen wie in C/C++.

- Verzweigungen (`if`,`switch`)
- Schleifen (`for`, `while`)

Die Vergleichsoperatoren wie `&&`, `||`, `!`, usw funktionieren wie in C. Aus der dynamischen Typisierung ergibt sich dennoch ein wichtiger Unterschied.

Kontrollstrukturen in JS

Es gibt Vergleichsoperatoren die nur den Inhalt der Variable vergleichen und Vergleichsoperatoren die auch den Datentyp vergleichen.

Operator	Was wird verglichen	Beispiel
==	Wert	`10 == "10"  //true`
===	Wert und Datentyp	`10 === "10"  //false`
!=	Wert	`10 != "10"  //false`
!==	Wert und Datentyp	`10 !== "10"  //true`

Link zum Beispiel

Arrays in JS

Arrays funktionieren JS sehr ähnlich wie in C. Sie bieten aber zusätzliche Funktionen zu C Arrays.

- Mit `push()` kann ein Element am Ende hinzugefügt werden.
- Mit `pop()` kann ein Element am Ende entfernt werden.

Link zum Beispiel

Objekte in JS

Objekte in JavaScript sind komplexe Datentypen, die verwendet werden, um Daten in Form von Schlüssel-Wert-Paaren zu organisieren. Ein Schlüssel ist eine Zeichenkette (String), die den Zugriff auf den zugehörigen Wert ermöglicht. Der Wert kann dabei jeder beliebige Datentyp (auch wieder ein Objekt) sein.

Link zum Beispiel

Daten lassen sich zum Austausch mittels Datei, Webservice, serieller Schnittstelle oder anderen Möglichkeiten, strukturiert darstellen. Dazu gibt es grundsätzlich zwei Möglichkeiten.

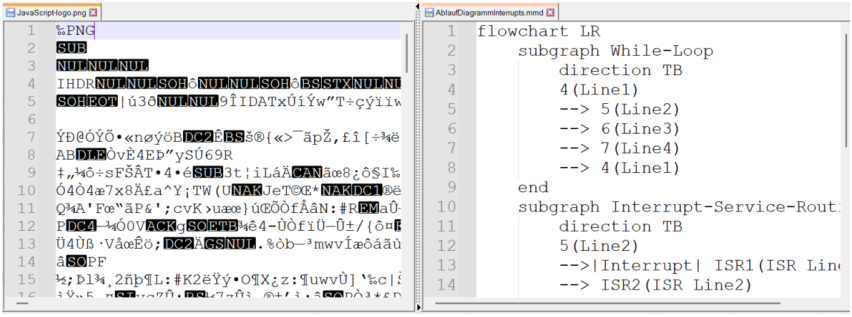

Figure 2: Binärformat vs. Textformat

Binäre Datenformate:

Dabei wird die Information nicht als lesbarer Text gespeichert, sondern als Abfolge von Bytes die dann jeweils eine spezielle Bedeutung haben. Der Vorteil dieser Formate ist, dass sie oft speicher-effizienter sind als Textformate.

Beispiele dazu sind:

- doc
- jpeg
- avi
- exe

Textbasierte Datenformate:

Dabei werden die Daten als menschlich lesbarer Text gespeichert. In der Software werden diese zumeist als String geladen und dann in eine Datenstruktur umgewandelt. Der Vorteil ist, dass die Daten klar lesbar und auch veränderbar sind.

Beispiele dazu sind:

- json
- xml
- csv
- yaml

JSON

JSON steht für “JavaScript Object Notation” und ist ein einfaches Datenformat, das menschenlesbar ist und außerdem sehr einfach von Software interpretiert werden kann.

Ein json Objekt besteht immer aus einem oder mehreren Schlüssel-Werte-Paaren (Key-Value Pairs). Im einfachsten Fall:

```
{
   "Name":"John Doe"
}
```

Der Value kann auch ein anderer Datentyp sein. Z.B.

```
{
    "Name":"John Doe",
    "Alter": 30,
    "Stadt": "New York",
    "Verifiziert": true
}
```

JSON

Auch ein Array oder eine weiteres json Objekt ist als Value möglich.

```
{
  "Name": "John Doe",
  "Alter": 30,
  "Stadt": "New York",
  "Verifiziert": true,
  "Freunde": ["Jane", "Bob", "Alice"],
  "Profil": {
    "Hobbys": ["Lesen", "Joggen", "Fotografie"],
    "MitgliedSeit": 2020,
    "PremiumStatus": false
  }
}
```

- Für die meisten Programmiersprachen und Anwendung gibt es Bibliotheken die json Strings in Datenstrukturen umwandeln können.
- In Node Red können json Daten mit dem json von einem String in eine Objekt umgewandelt werden.

Figure 3: JSON Node in Node-Red

XML steht für “eXtensible Markup Language” und ist eine Auszeichnungssprache, die zur Darstellung hierarchisch strukturierter Daten in Textform verwendet wird.

XML verwendet Tags, um Daten zu kennzeichnen und zu strukturieren. Ein Tag besteht aus einem Elementnamen, der von spitzen Klammern umschlossen ist, z.B., `<ElementName>`. Es gibt zwei Arten von Tags: Start-Tags und End-Tags. Zum Beispiel:

```
<Name>John Doe</Name>
```

Attribute von John Doe können bei XML auf zwei verschiedenen Arten hinzugefügt werden. Entweder als Einrückung mit einem eigen Tag oder als Attribut beim Start-Tag.

```
<Person Alter="30" Verifiziert="true">
  <Name>John Doe</Name>
  <Stadt>New York</Stadt>
</Person>
```

Ein Array kann in XML nicht direkt dargestellt werden. Stattdessen gibt es einen Container Knoten (<Freunde>) mit mehreren gleichen Einträgen (<Freund>).

```
<Person Alter="30" Verifiziert="true">
  <Name>John Doe</Name>
  <Stadt>New York</Stadt>

  <Freunde>
    <Freund>Jane</Freund>
    <Freund>Bob</Freund>
    <Freund>Alice</Freund>
  </Freunde>
</Person>
```

- Für die meisten Programmiersprachen und Anwendung gibt es Bibliotheken die xml Strings in Datenstrukturen umwandeln können.
- In Node Red können xml Daten mit dem xml von einem String in eine Objekt umgewandelt werden.

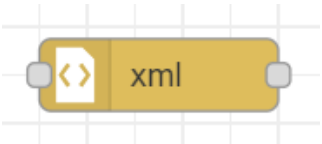

Figure 4: XML Node in Node-Red

CSV

CSV steht für “Comma-Separated Values” oder auf Deutsch “Kommagetrennte Werte”. Es handelt sich um ein einfaches Dateiformat zur Darstellung von tabellarischen Daten in Textform. In CSV-Dateien werden Daten in Zeilen und Spalten organisiert, wobei die Werte durch ein bestimmtes Trennzeichen, typischerweise ein Komma (`,`) oder ein Strichpunkt (`;`), getrennt sind. Die erste Zeile kann optional als Kopfzeile verwendet werden.

```
Name;Alter;Stadt
John;25;New York
Jane;30;San Francisco
Bob;22;Los Angeles
```

CSV

Im Gegensatz zu json oder xml ist csv keine Verschachtelung in Klassen oder Arrays direkt möglich. Csv kann nur flache Datenstrukturen darstellen. Die Datenstruktur aus dem JSON Beispiel kann nur sehr umständlich und unflexibel dargestellt werden. Soll z.B. ein weiterer Freund hinzugefügt werden muss eine neue Spalte für alle Einträge hinzugefügt werden.

```
Name;Alter;Stadt;Verifiziert;Freund1;Freund2;Freund3
John Doe;30;New York;true;Jane;Bob;Alice
```

- Für die meisten Programmiersprachen und Anwendung gibt es Bibliotheken die csv Strings in Datenstrukturen umwandeln können.
- In Node Red können csv Daten mit dem csv von einem String in eine Objekt umgewandelt werden.

Figure 5: CSV Node in Node-Red