JSON

Was ist JSON?

- Die JavaScript Object Notation (JSON) ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen.
- JSON wird zur Übertragung und zum Speichern von strukturierten Daten eingesetzt; es dient als Datenformat bei der Serialisierung.
- Insbesondere bei Webanwendungen und mobilen Apps wird es in Verbindung mit JavaScript, Ajax oder Web-Sockets zum Transfer von Daten zwischen dem Client und dem Server häufig genutzt.
- Jedes gültige JSON-Dokument soll ein gültiges JavaScript sein und per eval() interpretiert werden können.
- Davon abgesehen ist JSON aber unabhängig von der Programmiersprache.
- Parser existieren in praktisch allen verbreiteten Sprachen.

JSON-Datentypen

- Nullwert
  wird durch das Schlüsselwort **`null`** dargestellt.
- Boolescher Wert
  wird durch die Schlüsselwörter **`true`** und **`false`** dargestellt. Dies sind keine Zeichenketten.
- Zahl
  ist eine Folge der Ziffern 0–9.
  - Diese Folge kann durch ein negatives Vorzeichen - eingeleitet und einen Dezimalpunkt . unterbrochen sein.
  - Die Zahl kann durch die Angabe eines Exponenten e oder E ergänzt werden, dem ein Vorzeichen + oder - und eine Folge der Ziffern 0–9 folgt.

JSON-Datentypen

- Zeichenkette
  beginnt und endet mit doppelten geraden Anführungszeichen (").
  - Sie kann Unicode-Zeichen und Escape-Sequenzen enthalten.
  - Als Zeichenkodierung benutzt JSON standardmäßig UTF-8.
  - Auch UTF-16 und UTF-32 sind möglich.
- Array
  beginnt mit [und endet mit].
  - Es enthält eine durch Kommata geteilte, geordnete Liste von Werten gleichen oder verschiedenen Typs.
  - Leere Arrays sind zulässig.
  - Beispiele:

```
var monate = ["Januar", "Februar", "März", "April", "Juni"];
var prim = [1, 2, 3, 5, 7, 11, 13];
var bools = [true, false, false, true];
```

JSON-Datentypen

- Objekt
  beginnt mit { und endet mit }.
  - Es enthält eine durch Kommata geteilte, ungeordnete Liste von Eigenschaften.
  - Objekte ohne Eigenschaften, also leere Objekte, sind zulässig.
  - Die Daten können beliebig verschachtelt werden, beispielsweise ist ein Array von Objekten möglich.
  - Eine Eigenschaft
    besteht aus einem Schlüssel und einem Wert, getrennt durch einen Doppelpunkt (Schlüssel:Wert).
    - Die Schlüssel sollten eindeutig sein, da unterschiedliche Parser mit mehrfach vorkommenden Schlüsseln unterschiedlich umgehen.
    - Der Schlüssel ist eine Zeichenkette.
    - Der Wert ist einer der Datentypen.

JSON-Datentypen: Einschränkungen zu JavaScript

- JSON unterstützt nicht alle von JavaScript unterstützten Datentypen.
- Daher werden bei der Serialisierung
  - **`NaN`**, **`Infinity`** und **`-Infinity`** zu **`null`** serialisiert,
  - Date-Objekte als String in das ISO-8601-Format konvertiert und
  - Function-, RegExp- und Error-Objekte verworfen.

JSON und JavaScript-Objekte

- Ein JavaScript-Objekt im JSON-Format ist eine Liste von Schlüssel-Wert-Paaren in Klammern { }
- Syntax: { schlüssel1: wert1, schlüssel2: wert2 ... }

```
var student = {
    name: "Meier, Peter",
    alter: 25,
    note: 1.0,
    adresse: {
        strasse: "Hauptstrasse 42",
        plz: 68259,
        ort: "Mannheim"
    }
};
```

JSON und JavaScript-Objekte: Daten auslesen

```
var student = {
    name: "Meier, Peter",
    alter: 25,
    note: 1.0,
    adresse: {
        strasse: "Hauptstrasse 42",
        plz: 68259,
        ort: "Mannheim"
    }
};

log(student.name);
log(student.adresse.plz);
log(student.adresse.ort);
```

JSON und JavaScript-Objekte: Alternative Syntax

- JavaScript-Objekte können auch als assoziatives Array aufgefasst werden
- Syntax: { "schlüssel1": "wert1", "schlüssel2": "wert2" ... }

```
var student = {
    "name": "Meier, Peter",
    "alter": 25,
    "note": 1.0,
    "adresse": {
        "strasse": "Hauptstrasse 42",
        "plz": 68259,
        "ort": "Mannheim"
    }
};

log(student["name"]);
log(student["adresse"]["plz"]);
```

Funktionen in JSON

- Im Objekt-Literal können auch Funktionen zugewiesen werden
- Syntax: { name: function() { } ... }

```
var student = {
    name: "Meier, Peter",
    alter: 25,
    altern: function() { this.alter++; }
};

log(student.alter);
student.altern();
log(student.alter);
```

JSON vs. XML

- XML ist eine struktur-beschreibende Sprache.
- JSON im Gegensatz dazu eine Syntax-Konvention und überhaupt nicht deklarativ.
- Für den Datenaustausch an einer fest definierten Schnittstelle ist XML deutlich besser geeignet.
- JSON ist für flexible Schnittstellen besser geeignet.
- Eine Stärke von JSON liegt darin, dass es sich um valides JavaScript handelt.
  - Damit lässt sich eine JSON-Definition in JavaScript direkt mit der eval()-Funktion in ein JavaScript-Objekt umsetzen.
  - Bei Daten aus unsicheren Quellen sollte aber ein Parser verwendet werden, da eval auch ggf. schädliche Programmanweisungen ausführt.

JSON vs. XML

- XML ist eine Auszeichnungssprache und somit vielseitiger einsetzbar als JSON, das ein Datenaustauschformat ist.
- XML ist weiter verbreitet, wird jedoch von JSON aufgrund seiner Einfachheit dort zurückgedrängt, wo keine komplizierten Auszeichnungen notwendig sind.
- Beide Formate sind nicht gut zum Repräsentieren von Binärdatenmengen geeignet, da beide keinen Binärdatentyp unterstützen und zeichenweise interpretiert werden müssen.
- Die Syntax von JSON ist viel einfacher gestaltet und erscheint daher oft lesbarer und insbesondere leichter schreibbar.
- In der Regel produziert JSON auch geringeren Overhead im Vergleich zu XML...

JSON vs. XML

```
"person":{"firstname":"Fred",
          "lastname":"Flintstone",
          "age":38,
          "spouse":"Wilma" }
```

```
<person firstname='Fred'
        lastname='Flintstone'
        age='38'
        spouse='Wilma'/>
```

```
<person>
   <firstname>Fred<firstname>
   <lastname>Flintstone<lastname>
   <age>38</age>
   <spouse>Wilma</spouse>
</person>
```

- JSON-Objekte sind i.d.R. ca. 30% kleiner als XML-Dokumente, wenn man XML-Elemente statt Attribute verwendet.
- Im Gegensatz zu XML gibt es bei JSON keine Grammatiken wie DTD oder Schema.