JSON Schema

Daniel Flasch

Verteilte Anwendungssysteme, WS 17/18

Agenda

- Einführung
- Grundlagen
- Fortgeschrittene Techniken
- Übung

Einführung

Was ist JSON?

- Java Script Object Notation
- Kompaktes Datenformat
- JSON-Dokument = gültiges JavaScript
- Unabhängig von Programmiersprache
- Codierung in UTF-8

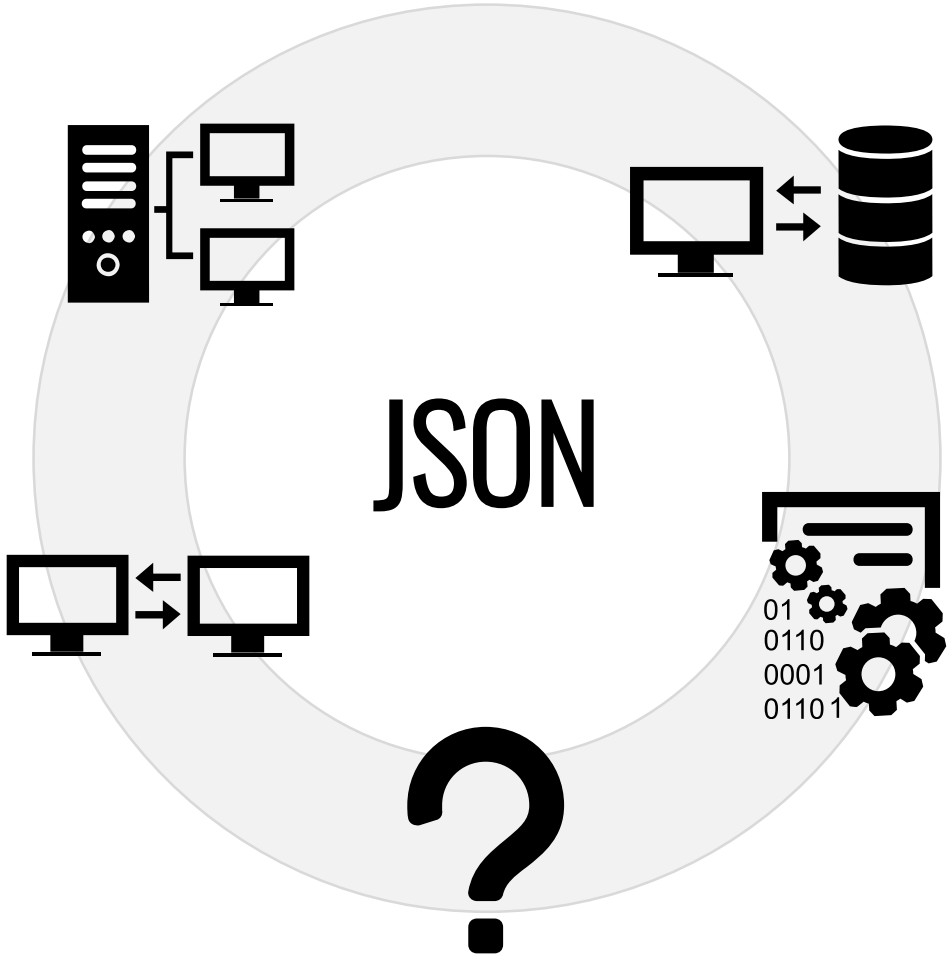
```
{
  "topic":"JSON Schema",
  "date":"2017-12-12",
  "speaker":"Daniel Flasch"
}
```

Die Syntax

- Objektaufbau mit Key/Value
- Komma als Trennzeichen
- Datentypen
  - Objekte { }
  - Arrays []
  - Zahlen: 42/4.2
  - Strings “ “
  - Boolean true/false
  - null

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ]
}
```

Wofür wird es verwendet?



Vergleich XML und JSON

Gemeinsamkeiten

- Selbstbeschreibend
- Hierarchischer Aufbau

Vorteile JSON { … }

- Geringerer Overhead
- Einfaches Parsen
- Native Verwendung im JS-Umfeld

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Was ist JSON Schema?

- Äquivalent zu XML-Schema für JSON
- Ist selbst JSON-basiert
- Definition von JSON-Datenstrukturen
- Für Validierung und Dokumentation
- Bisher nur im Entwurf-Status

```
{
  "$schema": "http://json-schema.org/schema#",
  "title": "Product",
  "type": "object",
  "required": ["id", "name"],
  "properties": {
    "id": {
      "type": "number",
      "description": "Product identifier"
    },
    "name": {
      "type": "string",
      "description": "Name of the product"
    },
    "price": {
      "type": "number",
      "minimum": 0
    }
  }
}
```

Grundlagen

Einfachstes Schema

```
{ }
```

```
42
```

```
"I'm a string"
```

```
{
  "an": [
    "arbitrarily",
    "nested" ],
  "data": "structure"
}
```

Deklarationen und Metadaten

- Sind **optional**
- **$schema** beschreibt die Schema-Version
- **$id** dient als Identifizierung
- **title** gibt Schema einen Titel
- **description** beschreibt das Schema genauer

```
{
  "$schema": "http://json-schema.org/schema#",
  "$id": "http://yourdomain.com/schemas/myschema.json",
  "title": "some title for your schema",
  "description": "place to describe schema"
}
```

Type-Schlüsselwort

- Typ definieren
- Einschränkung des Wertebereichs
- Mehrere Datentypen möglich

```
{ "type" : "string" }
```

```
42
"I'm a string"
```

```
{ "type": ["number", "string"] }
```

```
true
42
"I'm a string"
```

Elementare Datentypen

```
{ "type" : "string" }
```

```
42
"I'm a string"
```

```
{ "type" : "boolean" }
```

```
42
true
false
```

```
{ "type" : "number" }
```

```
"I'm a string"
4.2
42
```

Komplexe Datentypen

```
{ "type" : "array" }
```

```
42
```

```
["one","two","three"]
```

```
{ "type" : "object" }
```

```
42
```

```
{ }
```

Aufzählungen

```
{
  "type": "string",
  "enum": ["red", "yellow", "green"]
}
```

```
42
"blue"
"yellow"
```

```
{
  "type": "integer",
  "enum": [1,2]
}
```

```
1
2
3
```

Objekt-Eigenschaften

```
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "number"
    }
  }
}
```

```
{
  "name":"John Doe",
  "age":"28"
}
```

```
{
  "name":"John Doe",
  "age":28
}
```

```
{
  "name":"John Doe",
  "age":28,
  "phone":"012-2345-67"
}
```

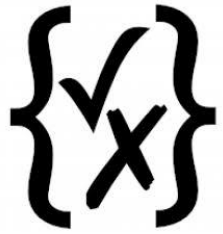
Objekt-Pflicht-Eigenschaften

```
{
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "number"
    }
  },
 „required":["name","age"]
}
```

```
{
  "name":"John Doe",
}
```

```
{
  "name":"John Doe",
  "age":28
}
```

```
{
  "name":"John Doe",
  "age":28,
  "phone":"012-2345-67"
}
```



Fortgeschrittene Techniken

Arrays (1/3)

- Verwendung des Schlüsselworts **item**
- Festlegung eines Datentyps für alle Elemente

```
{
  "type": "array",
  "items": {
    "type": "number"
  }
}
```

```
["one","two"]
[1,"two",3]
[1,2,3]
```

Arrays (2/3)

- Prüfung von Tupeln

```
{
  "type": "array",
  "items": [
    {
      "type": "number"
    },
    {
      "type": "string"
    },
    {
      "type": "string",
      "enum": ["Street", "Avenue"]
    }
  ]
}
```

```
[24, "Sussex", "Drive"]
[1600, "Pennsylvania", "Avenue"]
["Street"]
[10, "Street"]
```

Arrays (3/3)

- Prüfung auf Länge mit minItems/maxItems

```
{
  "type": "array",
  "minItems": 2,
  "maxItems": 3
}
```

```
["Street"]
```

```
[10, "Street"]
```

```
[24, "Sussex", "Drive"]
```

- Prüfung auf Einzigartigkeit

```
{
  "type": "array",
  "uniqueItems": true
}
```

```
[1,1]
```

```
[1,2]
```

Komplexe Wertebereiche

- allOf/anyOf/oneOf/not

```
{
  "allOf": [
    { "type": "string" },
    { "maxLength": 5 }
  ]
}
```

```
5
"string to long"
"short"
```

```
{ "not": { "type": "string" } }
```

```
10
"strings not allowed"
```

Wiederverwendung von Definitionen (1/2)

- Definitionen in gleicher/anderer Datei möglich
- Referenzieren mit **$ref** Schlüsselwort
- Nach **$ref** folgt der **Pfad** zur Definition
- Beispiel verweist auf externe Datei
  - **definitions.json** ist die Datei
  - **#** steht für das Wurzelelement
  - **address** für den **Schlüssel** address, hinter der die Definition liegt

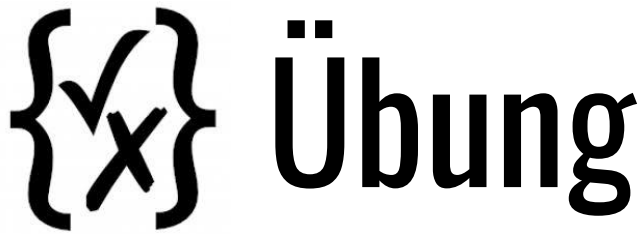
```
{ "$ref": "definitions.json#/address" }
```

Wiederverwendung von Definitionen (2/2)

```
{
  "definitions": {
    "address": {
      "type": "object",
      "properties": {
        "street_address": { "type": "string" },
        "city":           { "type": "string" },
        "state":          { "type": "string" }
      },
      "required": ["street_address", "city", "state"]
    }
  },

  "type": "object",
  "properties": {
    "billing_address": {
      "$ref": "#/definitions/address"
    },
    "shipping_address": {
      "$ref": "#/definitions/address"
    }
  }
}
```

```
{
  "shipping_address": {
    "street_address": "1600 Pennsylvania Avenue NW",
    "city": "Washington",
    "state": "DC"
  },
  "billing_address": {
    "street_address": "1st Street SE",
    "city": "Washington",
    "state": "DC"
  }
}
```


Übung

Links

- Präsentation
  - https://git.io/vbBcw (PowerPoint)
  - https://git.io/vbBcA (PDF)
- Übungsseite
  - 194.95.221.248:8080
- Quellen
  - http://json-schema.org/
  - https://spacetelescope.github.io/understanding-json-schema/