Semistrukturierte Daten

JSON

Stefan Woltran
Emanuel Sallinger

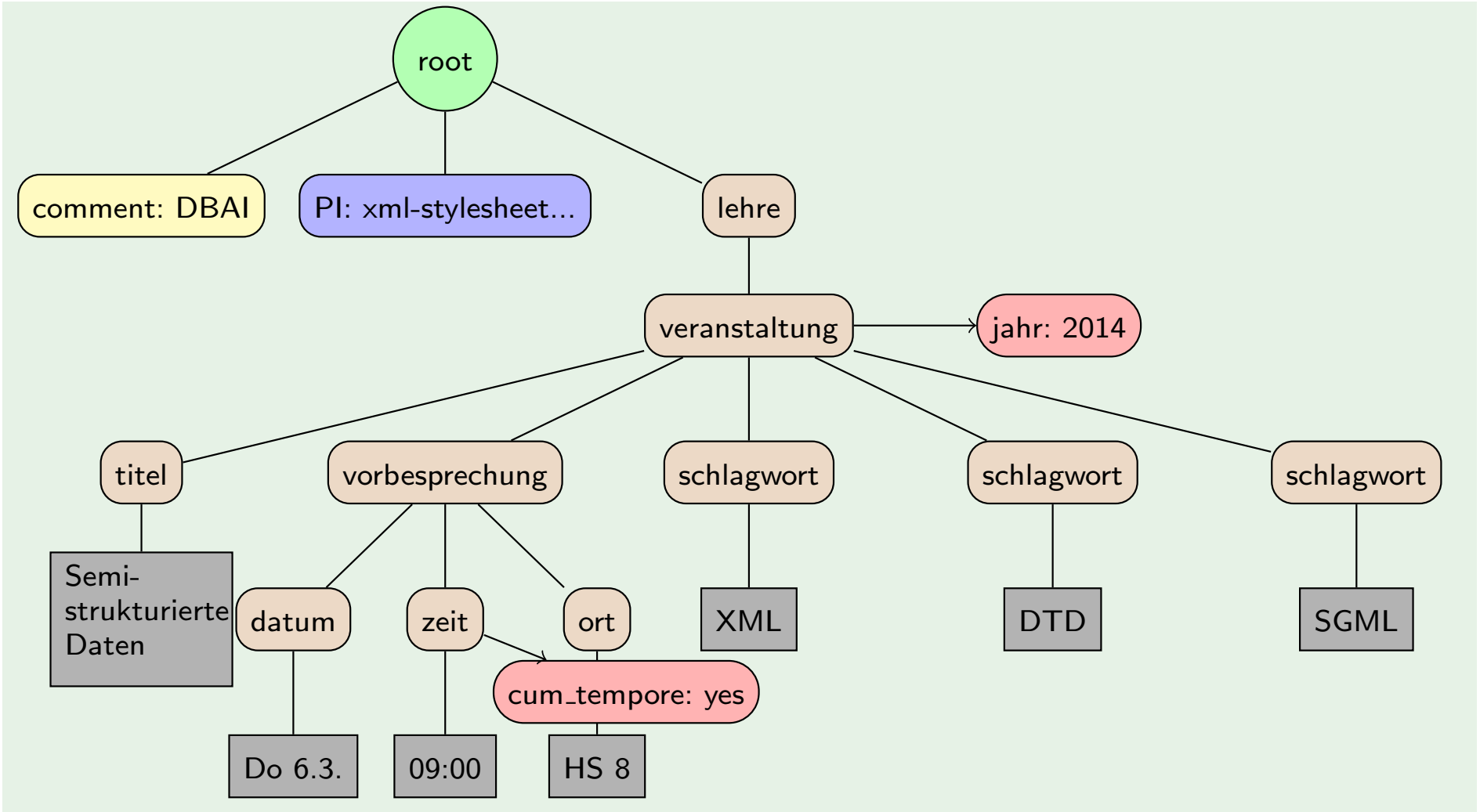
Institut für Informationssysteme
Technische Universität Wien

Sommersemester 2014

JSON

- JSON steht für *JavaScript Object Notation*
- Subset des ECMAScript (JavaScript) Standards
  - Nicht an JavaScript "gebunden" $\Rightarrow$ sprachenunabhängig
  - Parser existieren für fast alle Programmiersprachen (`json.org`)
- Human-readable, plain text Format
- Oft verwendet in Web Anwendungen und Web Services

Beispiel (Dokumentbaum)



Beispiel (XML)

```
<?xml version="1.0"?>
<!-- DBAI -->
<?xml-stylesheet type="text/css"href="lehre.css"?>
<lehre>
  <veranstaltung jahr="2014">
    <titel>Semistrukturierte Daten</titel>
    <vorbesprechung>
      <datum>Do 6.3.</datum>
      <zeit cum_tempore="yes">09:00</zeit>
      <ort>HS 8</ort>
    </vorbesprechung>
    <schlagwort>XML</schlagwort>
    <schlagwort>DTD</schlagwort>
    <schlagwort>SGML</schlagwort>
  </veranstaltung>
</lehre>
```

Beispiel (JSON)

```
{"lehre" :
    [
      {"veranstaltung" :
        {"titel" : "Semistrukturierte Daten",
         "jahr" : 2014,
         "vorbesprechung" :
           {"datum" : "Do 6.3.",
            "zeit" :
              {"cum_tempore" : "yes",
               "value" : "09:00",
               "ort" : "HS 8"}
            },
          "schlagwort" : ["XML", "DTD", "SGML"]
        }
      }
    ]
}
```

Aufbau

Datentypen

- Arrays
- Objekte
- Basistypen

- Arrays und Objekte können verschachtelt sein
- Auf oberster Ebene muss ein Objekt oder Array stehen
- Kommentare (zB in JavaScript Syntax) sind nicht erlaubt

Basistypen

- String
- Number
- Boolean (`true` oder `false`)
- `null`

Beispiele für Strings:

```
"string"    "unicode escape \u00FC"    "newline \n"
```

Beispiele für Numbers:

```
12.3    -10e13
```

Arrays

- Geordnete Liste von Werten
- Elemente der Liste können beliebige Typen aufnehmen
- Werte werden in eckige Klammern gesetzt und durch Beistriche getrennt

Einfaches Array:

```
["string", 12.3, true]
```

Verschachteltes Array:

```
["string", [1,2,3]]
```

Objekte

- Menge von Key-Value Paaren
- Key ist immer ein String (Duplikate sind *nicht* erlaubt)
- Value ist ein beliebiger Typ
- Key-Value Paare (Doppelpunkt zwischen Key und Value) werden in geschwungen Klammern gesetzt und durch Beistriche getrennt

Beispiel eines komplexeren Objekts:

```
{"lva" : "SSD",
 "name" : {"name" : "Max", "matrnr" : "0828112" },
 "grades" : [12.5, 15, "S1"]}
```

Verwendungskontext

- Im Webbereich oft als leichtgewichtige Alternative zu XML
  - Parsing von JSON ist weniger komplex als Parsing von XML
  - Stärkere Nähe zu den JavaScript Datenstrukturen
- Meistens als Datenaustauschformat verwendet
- In NoSQL Systemen auch als Datenspeicherungsformat (zB MongoDB, CouchDB,...)
- Eher selten eingesetzt als Dokumentformat bzw. Auszeichnungssprache

Rund um JSON

Noch keine etablierten Standards, aber eine Reihe von Kandidaten

- Schemasprachen
  - JSON Schema[1] derzeit als Entwurfsversion
- Abfragesprachen
  - Systeme verwenden oft eigene Abfragesprachen
  - JSONiq[2]
  - JAQL

[1] `http://json-schema.org/`
[2] `http://www.jsoniq.org/`

Relationales Modell und JSON im Vergleich[3]

	Relational	**JSON**
Struktur	Tabellen	Geschachtelte Mengen/Arrays
Schema	Im voraus fixiert	Selbstbeschreibend, flexibel
Abfragen	Ausdrucksstarke Sprachen	Kein Standard
Sortierung	Keine	Arrays
Implementierung	Native Systeme	Prog.Sprachen, NoSQL

[3] Widom, Jennifer. JSON Data, CS145 - Introduction to Databases, Stanford University, Fall 2012. Lecture Slides.

XML und JSON im Vergleich[4]

	XML	**JSON**
Verbosity	Mehr	Weniger
Komplexität	Mehr	Weniger
Validität	DTD, XSD weit verbreitet	JSON Schema nicht verbreitet
Prog. Interface	Keine	Direkter gekoppelt
Abfragen	XPath, XQuery, XSLT	JSON Path, JSON Query, JAQL

[4] Widom, Jennifer. JSON Data, CS145 - Introduction to Databases, Stanford University, Fall 2012. Lecture Slides.