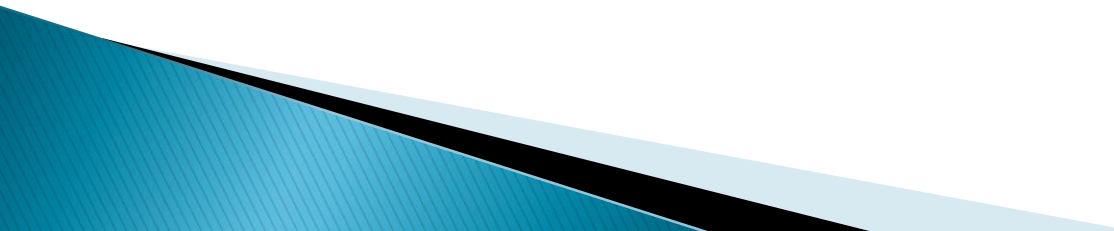


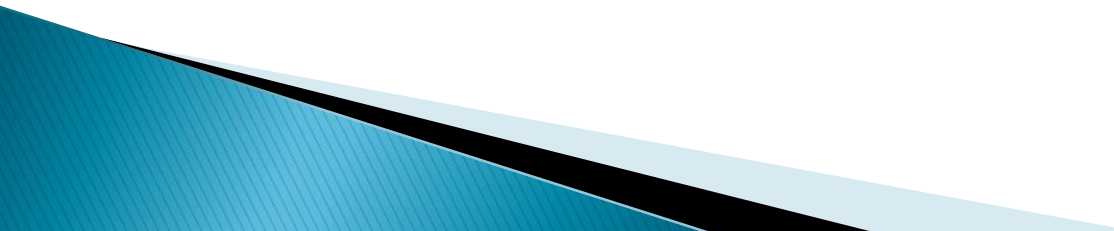
JSON

JavaScript Object Notation

Übersicht

- ▶ Web 2.0 // AJAX
 - ▶ Datenübertragung mit XML
 - ▶ Schwachstellen von XML
 - ▶ Idee hinter JSON
 - ▶ Aufbau von JSON
 - ▶ Vorteile JSON
- 

Web 2.0 // AJAX

- ▶ Websites sollen Inhalte dynamisch nachladen
 - ▶ Realisierung mittels AJAX:
Asynchronous JavaScript and XML
 - ▶ JavaScript ruft Seiten im Hintergrund auf
 - ▶ Dynamische Manipulation der Oberflächen
- 

Datenübertragung mit XML

- ▶ XML als klassische Auszeichnungssprache
- ▶ Sehr mächtiges Format

- ▶ Anwendungsbeispiel:
Liste von aktiven Benutzern eines Portals soll dynamisch nachgeladen werden.

Datenübertragung mit XML

```
<onlineusers>
  <user>
    <uid>617</name>
    <name>Tobias Maier</name>
    <email>a09005@hb.dhbw-stuttgart.de</name>
    <online>
      <since>1106728364</since>
      <status>busy</status>
    </online>
  </user>
  <user>
    ...
  </user>
</onlineusers>
```

Schwachstellen von XML

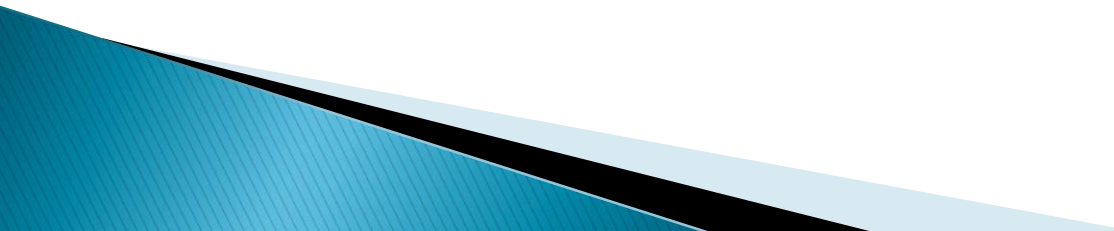
- ▶ Großer Datenoverhead durch XML-Struktur
- ▶ Client-Seite (meist JavaScript) muss XML parsen, um brauchbare Daten zu erhalten
- ▶ Allerdings: Einsatz von „Direkt-HTML“ möglich

Idee hinter JSON

- ▶ Daten werden als „serialisierte“ JavaScript-Objekte übertragen
- ▶ En- und Decoding-Funktionen in allen modernen Sprachen

Idee hinter JSON

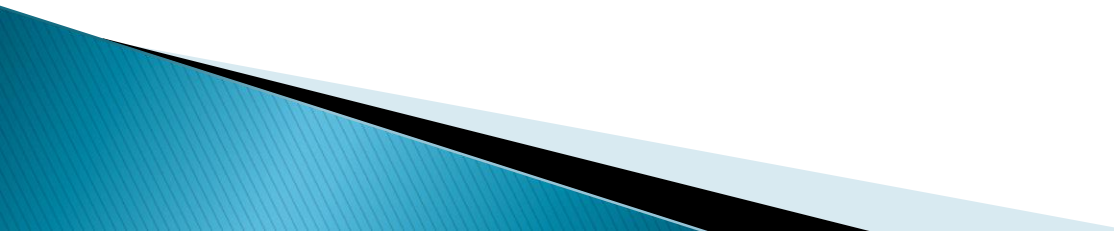
```
{
  {
    uid: 617,
    name: "Tobias Maier",
    email: a09005@hb.dhbw-stuttgart.de,
    online: {
      since: 1106728364,
      status: "busy"
    }
  },
  {
    ...
  }
}
```



Aufbau von JSON

- ▶ Verschachtelte JavaScript-Objekte möglich
- ▶ Verfügbare Datentypen:
 - Boolescher Wert
 - Zahl
 - Zeichenkette
 - Nullwert
 - Array
 - Objekt

Vorteile JSON

- ▶ Deutlich schlankeres Format
 - Overhead im Beispiel von 101 auf 43 Zeichen reduziert
 - ▶ Besser lesbar für Menschen
 - ▶ Objekt-orientiert
 - ▶ Direkte Verarbeitung der JSON-Objekte in JavaScript
- 

Vielen Dank!

Tobias Maier

DHBW Stuttgart Campus Horb // SPIRIT/21

tmaier@spirit21.de

