JSON
JavaScript Object Notation

Tobias Maier[1]

[1]DHBW Stuttgart Campus Horb, 2012

Zusammenfassung. Diese Arbeit erläutert die Idee und den Nutzen von JSON.

1 Einleitung

„Der Begriff Web 2.0 prägt seit einigen Jahren den Sprachjargon rund um das World Wide Web. Entstanden ist er irgendwann um 2003. Das Web 2.0 steht für eine neue Ära im Umgang mit dem Internet. Der Gebrauch der Versionsnummer soll eine Abgrenzung zum bisherigen WWW, sozusagen der Version 1.0 darstellen.

Bestanden Websites bis dahin meist aus statischen HTML-Seiten, die von einem festen Autoren-Kreis bearbeitet wurden, rückte nun die dynamische Bearbeitung von Inhalten immer mehr in den Vordergrund. Benutzer einer Website konnten und sollten selbst Inhalte einstellen. Überall entstanden Gästebücher, Foren, Weblogs und Wikis. Das Content-Management wurde zu einem zentralen Begriff: Bearbeiter einer Website brauchten keine umfassenden Programmierkenntnisse mehr, sondern konnten mit webbasierten Oberflächen ihre Seiten bearbeiten und die Änderungen sofort sehen bzw. online stellen.

Heute werden die meisten Seiten beim Aufrufen dynamisch aus einer Datenbank zusammengebaut. Die Aktualität der Inhalte ist dadurch deutlich gestiegen, ebenso die Unabhängigkeit von Inhalt und Design. Selbst für die größten Internetauftritte ist es nur noch ein geringer Aufwand, das Layout komplett umzustellen. Hätte man dafür früher alle HTML-Seiten einzeln anpacken müssen, genügen in Zeiten des Web 2.0 meist wenige Änderungen in einigen zentralen Dateien.

Die klassische Server-Client-Aufteilung im WWW ist denkbar einfach: Der Server liefert Webseiten als HTML-Code aus, der Client, in Form eines Browsers, hat lediglich die Aufgabe, sie dem Benutzer entsprechend darzustellen. Interaktion des Benutzers mit der Website umfasst immer ein Senden der Daten an den Server, die dortige Verarbeitung und anschließende Auslieferung des Ergebnisses mittels einer neuen HTML-Seite in Richtung Browser. So war es zu Zeiten des Web 1.0.

Inzwischen hat sich die Aufgabenverteilung zwischen Server und Client stark verändert. Der Server liefert weiterhin Seiten aus, die er jedoch zumeist zum Aufrufzeitpunkt erst dynamisch generiert. Die Inhalte dazu bezieht er aus einer Datenbank, die als dritte Komponente hinzugekommen ist. Der Client (Browser) stellt die Inhalte, die er bekommt, nicht nur dar, sondern ist darüber hinaus auch für Animationen, Interaktionsmöglichkeiten und dynamisches Nachladen einzelner Seitenteile verantwortlich.“

Mit diesen Worten habe ich bereits meine letzte Arbeit zum Thema JavaScript bzw. JavaScript-Frameworks eingeleitet (s. Verweis im Quellenverzeichnis). In dieser Ar-

beit geht es jedoch nicht um die Programmiersprache, mit der im Browser die Inhalte aufbereitet und dargestellt werden, sondern um das Format, über das Client und Server heute in den meisten Fällen kommunizieren: JSON.

2 Aus der Historie

Eng verbunden mit Web 2.0 ist AJAX. Die Abkürzung steht für Asynchronous JavaScript and XML. Dahinter verbirgt sich ein Konzept der asynchronen Kommunikation zwischen Server und Client: Der Browser sendet (meist ohne sichtbaren Effekt für den Benutzer) eine HTTP-Anfrage an den Server, erhält dessen Antwort und wertet diese entsprechend aus, beispielsweise indem er eine Meldung ausgibt oder die Oberfläche verändert. Für weitere Informationen zu AJAX sei auch an dieser Stelle auf die bereits referenzierte Arbeit über JavaScript-Frameworks verwiesen.

Der ursprüngliche Gedanke hinter AJAX sieht – wie der Name bereits verrät – vor, dass die Antwort des Servers an den Client aus einem validen XML-Dokument besteht. In der Praxis ist dies meist ein HTML-Quellcode, der ja dem XML-Standard entspricht. Das zurückgelieferte HTML wird dann ohne weitere Bearbeitung in die aktuelle HTML-Seite eingebaut. Damit lassen sich Seiteninhalte dynamisch und ohne ein komplettes Aktualisieren der Seite verändern.

Für den eben beschriebenen Anwendungsfall eignet sich XML bzw. HTML hervorragend. Müssen die zurückgelieferten Daten allerdings noch bearbeitet werden oder dienen gar nicht der Anzeige für den Benutzer, wird es schon aufwendiger. Der Client muss das XML dann zunächst parsen, was eine gewisse Zeit und Rechenleistung kostet.

Nachfolgendes Beispiel soll die Situation verdeutlichen: Ausgegangen werden soll von einer beliebigen Web 2.0-Anwendung, die dem Benutzer neben vielen anderen Inhalten auch eine Liste anderer aktiver Benutzer (etwa neben einem Chat-Fenster) präsentiert. Diese Liste muss regelmäßig dynamisch aktualisiert werden. Per AJAX sollen die relevanten Daten vom Server geholt werden. Dabei soll der Server kein fertiges HTML-Dokument zurückliefern, das die alte Benutzerliste dann ersetzt, sondern lediglich die reinen Daten, beispielsweise weil die Liste vor der Anzeige noch grafisch aufbereitet werden soll. Die Antwort des Servers könnte z.B. so aussehen:

```
<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>
```

Wie bereits angedeutet, muss der Client diese Antwort nun zunächst parsen, um aus dem XML brauchbare JavaScript-Objekte zu generieren und diese dann wiederum in HTML-Code zu gießen. Warum nicht direkt das aufbereitete HTML übertragen wird, könnte wie ebenfalls weiter oben schon vorgeschlagen daran liegen, dass der Browser bestimmte Informationen hat, um die Liste aufzubereiten, die er nicht an den Server übertragen möchte. Um eine saubere Trennung von Logik (Server) und Darstellung (Client) zu erreichen, ist so ein Vorgehen durchaus sinnvoll. Wer sich davon noch nicht überzeugen lässt, der möge einen Schritt weiter denken und die Tatsache in Betracht ziehen, dass eventuell noch eine oder mehrere Smartphone-Apps existieren, die über denselben Weg auf die Daten zugreifen müssen, diese allerdings komplett unterschiedlich darstellen.

Neben dem Aufwand, die XML-Daten zunächst zu parsen und in JavaScript umzuwandeln, hat die klassische XML-Struktur noch einen zweiten bedeutenden Nachteil: Ein gewaltiger Overhead an Daten muss jedes Mal mit übertragen werden. Besonders in Zeiten des mobilen Internets sollte ein Schwerpunkt der Webanwendungsentwicklung auf der Optimierung der Übertragungsgeschwindigkeit und damit einer Minimierung des Übertragungsumfangs liegen.

3 Schlankere Antworten dank JSON

Genau an dieser Stelle setzt JSON an. Die JavaScript Object Notation beschreibt eigentlich nichts anderes, als dass Daten in Form von JavaScript-Objekten übertragen werden. Das vorherige Beispiel sieht dann so aus:

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

Die Antwort ist valides JavaScript und kann mit der JavaScript-Funktion eval() direkt als Objekt interpretiert werden. Der Rechenaufwand ist hier viel geringer. Ebenso ist die Menge der zu übertragenden Daten viel geringer als mit XML. Um diese Aussage mit Zahlen zu belegen: Im XML-Beispiel waren für einen Benutzer 101 Zeichen nötig, nur um die Struktur zu beschreiben (von <user> bis </user>, ohne die eigentlichen Daten). Das JSON-Beispiel kommt für dieselbe Aufgabe mit lediglich 43 Zeichen aus (von { bis }, ebenfalls ohne Daten). Damit kann der Overhead schon bei diesem winzigen Beispiel um fast 60% reduziert werden.

4 Datenstruktur von JSON

Wie im Beispiel bereits gezeigt besteht eine JSON-Nachricht aus einem JavaScript-Objekt (definiert durch das einleitende { und das abschließende }) sowie beliebig viele Attribute dieses Objekts, die ihrerseits wiederum Objekte oder Arrays von Objekten sein können. Dadurch lassen sich die Daten beliebig tief verschachteln.

Darüber hinaus sind die Attribute des JSON-Objekts typisiert. Diesen Typ behalten sie auch nach der Interpretation beispielsweise durch JavaScript oder auch eine beliebige andere Programmiersprache (alle aktuellen Sprachen können JSON in eigene typisierte Objekte umwandeln). Folgende Datentypen unterstützt JSON:

- Boolscher Wert: true oder false
- Zahl: 0 bis 9, inklusive Vorzeichen (-) oder Exponent (e oder E)
- Zeichenkette: beliebige Folge von Zeichen in Anführungszeichen („string“)
- Nullwert: Schlüsselwort null
- Array: Liste von Werten: [value1, value2]
- Objekt: Liste von Eigenschaften: { key1: value1, key2: value2 }

5 Fazit

XML ist und bleibt sicherlich die mächtigste Auszeichnungssprache. Es ist vielseitig einsetzbar, nicht nur in der Übertragung von Daten. Gerade im webbasierten Datenaustausch ist JSON als objektbasiertes Austauschformat jedoch eine schlanke und sehr effiziente Alternative, die XML auf lange Sicht sicherlich aus dieser Domäne verdrängen wird.

Inspiriert durch JSON gibt es bereits etliche weiterführende oder vergleichbare Konzepte wie JSONP (JSON mit Padding), womit die Same-Origin-Policy von AJAX ausgetrickst werden kann, oder YAML (Yet Another Markup Language), eine Auszeichnungssprache, die Obermenge von JSON ist. Eine detaillierte Untersuchung dieser Konzepte sprengt jedoch den Rahmen dieser Ausarbeitung und soll deshalb nur als Ausblick angeführt werden.

6 Quellen

- jQuery & Co – JavaScript-Frameworks im Vergleich; Tobias Maier, 2011, veröffentlicht unter http://jwillmer.de/blog/2011/12/07/javascript-frameworks
- www.json.org
- www.ietf.org/rfc/rfc4627