Doing Web Apps

JavaScript und AJAX

Autor: Rüdiger Marwein
Letzte Änderung: 2009-05-16
Version: 0.6

Inhaltsverzeichnis

1.Einleitung

JavaScript ist eine Java ähnelnde, clientseitige Scriptsprache. Sie wurde 1995 von Netscape erstmals mit dem Navigator 2.0 veröffentlicht.

Sie wird vornehmlich verwendet um Attribute von Elementen einer Webseite zu ändern bzw die Elemente eine Webseite zu verwalten und Benutzerinteraktionen zu unterstützen.

Die Elemente einer HTML-Webseite sind in einer XML-Struktur abgelegt, deren Eltern-Kind-Beziehung in einem sog. Document Object Model abgebildet werden. Über die DOM-Zugriffsmethoden von JavaScript ist jedes Element einer Webseite programmatisch erfassbar. Der DOM-Baum lässt auch Schreiboperationen zu, wodurch der Inhalt und Aufbau einer Webseite auch nachträglich verändert werden kann.

Relativ neu (IE5 – 2000, Mozilla – 2002) ist das xmlHttpRequest[1] Objekt. Auf diesem Objekt basiert ein Verfahren, dass sich AJAX nennt, welches das xmlHttpRequest Objekt verwendet um einen nebenläufigen HTTP-Request an einen Server zu stellen, ohne die Webseite selbst dabei neu zu laden.

AJAX ist keine Technik sondern die Idee mit dem xmlHttpRequest Objekt XML-Daten im Hintergrund von einem Server zu holen und sie mit JavaScript zu verarbeiten. **A**synchronous **J**avaScript **A**nd **X**ML.

Aktuell geht der Trend weg von XML, hin zu JSON (JavaScript Object Notation) oder AHAH[2] (Asynchronus HTTP And HTML).

JSON ist darum eine gute Wahl, weil es einen sehr niedrigen Verarbeitungsaufwand auf Clientseite Bedarf und im Gegensatz zu XML mit wesentlich weniger Auszeichnungselementen auskommt, wodurch die Bandbreite geschont wird. Als weiteres Argument für JSON steht, dass die Services zumeist ohnehin von Browser-Clients per JavaScript verarbeitet werden müssen und so ein Zwischenschritt der Aufbereitung entfällt.

2.JSON

Die JavaScript Object Notation vereinfacht das Erstellen strukturierter Objekte und Arrays.

```
// bisher
var myObject = new Object();
myObject.myArray = new Array('hallo', 'welt');
myObject.myValue = 199;
// 101 Bytes mit Formatierung, 91 ohne

// oder mit JSON
var myObject = {
  myArray : ['hallo', 'welt'],
```

1 Vgl. http://en.wikipedia.org/wiki/XMLHttpRequest
2 Vgl. http://en.wikipedia.org/wiki/AHAH

```
  myValue : 199
};
// 68 Bytes mit Formatierung, 52 ohne
```

Objekte werden mit geschweiften Klammern deklariert, Arrays mit eckigen Klammern. Attribute werden mit Komma getrennt, das letzte Attribut hat keine Komma am Ende.

Um ein Gefühl für den XML-Overhead zu bekommen das obige beispielhaft als XML:

```
<object>
  <myArray>
    <string>hallo</string>
    <string>welt</string>
  </myArray>
  <myValue value="199" />
</object>
<!-- 129 Bytes mit Formatierung, 103 ohne →
```

Eine JSON Zeichenkette kann ohne große Mühe in ein JavaScript Objekt transformiert werden mittels eval().

```
eval('var myObject = '+jsonString);
alert(myObject.myArray[0] + myObject.myArray[1]);
```

Der vom Server übertragene JSON-String wäre

```
{ myArray : ['hallo', 'welt'], myValue : 199 }
```

3.JavaScript Crashkurs

Die Kontrollstrukturen und Schleifen von JavaScript sind für fortgeschrittene Programmierer trivial und bedürfen keiner weiteren Ausführung. Es sei an dieser Stelle auf SelfHTML verwiesen, dessen Online-Inhalte unter http://de.selfhtml.org/ abrufbar sind. Eine CHM-Hilfedatei zum offline nachschlagen ist unter http://aktuell.de.selfhtml.org/extras/selfchm.htm zu finden.

3.1.Die wichtigsten Eckdaten der JavaScript-Programmierung in Kürze

Bei Funktions- und Variablennamen muss die Groß-/Kleinschreibung beachtet werden.

```
var posx = 9, posX = 10, PosX = 11;
```

Zeichenketten werden mit + (Plus) verbunden, Zeichenkette können in Hochkomma oder Gänsefüßchen stehen.

```
document.write('Hallo' + name + ", wie geht es Dir?");
```

Eine Systemmeldung kann mit „alert“ erzeugt werden

```
alert("Wollen Sie den Eintrag wirklich löschen?");
```

Variablen werden mit `var` deklariert

```
var myVar = "Hallo Welt";
```

Der DOM-Baum ist durch das globale Objekt `document` von überall zugreifbar.

```
var links = document.getElementsByTagName("A");
```

Elemente mit einer `id` können punktuell isoliert und manipuliert werden.

```
// wenn <div id="myIdentifier">haha</div>
var myDiv = document.getElementById('myIdentifier');
myDiv.innerHTML = 'hihi';
// => <div id="myIdentifier">hihi</div>
```

Die Stylesheet-Eigenschaften eines Elementes sind über .style erreichbar. Im Gegensatz zum CSS werden die Parameternamen mit gemischter Groß-/Kleinschreibung notiert. Die CSS-Klasse des Elements kann über das Attribut className gelesen und geändert werden.

Visuelle Effekte werden hauptsächlich über das ändern von Stylesheet-Eigenschaften realisiert, wie bspw Animationen, Farbeffekte, Transparenz etc...

```
// wenn <div id="target" style="border-bottom:1px solid red"
//           class="help">haha</div>
var element = document.getElementById("target");
element.style.borderBottom = "1px solid green";
element.className = "solved";
```

Das aktuelle Fenster ist im globalen Objekt `window` verfügbar.

```
alert(window.location.href);
```

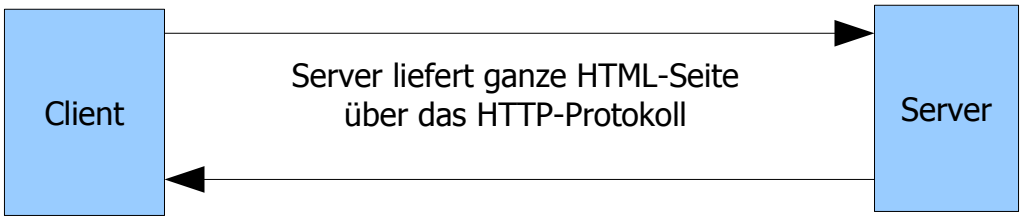
Um eine Reihe von Objekten - bspw des DOM-Baums - zu durchlaufen gibt es eine

spezielle Form der for-Schleife.

```
var links = document.getElementsByTagName("A");
for(i in links) {
  var link = links[i];
  link.target = "_blank";
}
```

4.xmlHttpRequest - AJAX entzaubert

Ein Webbrowser holt vom Webserver i.d.R. eine komplette Webseite mit allen Inhalten (Bilder, Stylesheets etc). Hierbei wird das Dokument in seiner Gesamtheit neu initialisiert.



Mit dem xmlHttpRequest Objekt ist es möglich, dass eine JavaScript-Komponente eine HTTP-Anfrage an den Webserver stellt; im Hintergund. Die HTTP-Antwort des Webservers wird ebenfalls mit JavaScript entgegengenommen und verarbeitet.

Was genau mit den Daten passiert ist offen. Es könnte beispielsweise eine Meldung ausgegeben werden, der Inhalt eines Elementes aktualisiert werden, eine Farbe geändert werden, eine Animation durchgeführt werden...

Einen xmlHttpRequest von Hand zu erstellen ist recht aufwändig und schon beim ersten Betrachten erkennt man, dass dieser leicht in eine Funktion kapselbar wäre.

```
var xmlHttp = null;
// Mozilla, Opera, Safari sowie Internet Explorer (ab v7)
if (typeof XMLHttpRequest != 'undefined') {
    xmlHttp = new XMLHttpRequest();
```

```
}
if (!xmlHttp) {
    // Internet Explorer 6 und älter
    try {
        xmlHttp  = new ActiveXObject("Msxml2.XMLHTTP");
    } catch(e) {
        try {
            xmlHttp  = new ActiveXObject("Microsoft.XMLHTTP");
        } catch(e) {
            xmlHttp  = null;
        }
    }
}
if (xmlHttp) {
    xmlHttp.open('GET', '/dienste/wetter', true);
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState == 4) {
            alert(eval(xmlHttp.responseText));
        }
    };
    xmlHttp.send(null);
}
```

Quelle: http://de.wikipedia.org/wiki/XMLHttpRequest

4.1.xmlHttpRequest in JavaScript Frameworks

Wie eingangs erwähnt lädt das xmlHttpRequest Objekt zur Kapselung ein. Darum ist es in den gängigen JavaScript-Frameworks bereits vereinfacht integriert.

JavaScript Frameworks sorgen für eine Vereinheitlichung der verschiedenen browserspezifischen Sprachkonstrukte und Eigenheiten. Wer mit JavaScript browserübergreifend arbeiten möchte, sollte sich mit den Frameworks auseinandersetzen.

jQuery

```
var data = null;
$.getJSON('/dienste/wetter', function(data) {
   alert(data);
});
```

Prototype

```
new Ajax.Request('/dienste/wetter', {
  method: 'get',
  onSuccess: function(transport) {
    alert(eval(xmlHttp.responseText));
```

```
  }
});
```

Mootools

```
var jsonRequest = new Request.JSON({url: "/dienste/wetter",
onSuccess: function(result,text) {
    alert(result);
}});
jsonRequest.get();
```

4.2.Beispielanwendung mit jQuery

HTML:

```
...
<div id="quoteBox">
  <div id="quote"></div>
  <div id="author"></div>
  <p><a href="#" onclick="nextQuote();return false;">Next</a></p>
</div>
<script type="text/javascript">
  nextQuote(); // load first quote immediately
</script>
...
```

JavaScript:

```
function nextQuote() {
  $.getJSON('/quotes/getRandom.php', function(data) {
   $('#quote').html(data.quote);
   $('#author').html(data.author);
  });
}
```

PHP-Script **getRandom.php** Beispiel 1:

```
$row = $DATABASE->getRandomQuote();
echo '{ quote:"'.$row['quote'].'", author:"'.$row['author'].'" }';
exit;
```

PHP-Script **getRandom.php** Beispiel 2:

```
$row = $DATABASE->getRandomQuote();
echo json_encode($row);
exit;
```