Moderne Webtechnologien I (WT 1)
Übung: Ajax

WiSe 23/24

Prof. Dr. C. Köhn
Daniela Böing

2. Januar 2024

Website:
https://www.hochschule-bochum.de/fbe/fachgebiete/institut-fuer-informatik/labor-fuer-medienkommunikation-internet-und-robotik/

Hochschule Bochum
Bochum University
of Applied Sciences

MIRO LAB

Inhaltsverzeichnis

1 JSON und Ajax

1.1 Ajax

1.1.1 Das Ajax-Prinzip

Ajax steht für „**A**synchronous **J**avaScript **A**nd **X**ML“. Dabei werden Daten zwischen dem Browser und dem Server „asynchron“ ausgetauscht. Das heißt, eine HTML Seite muss nicht komplett neu geladen werden, um neue Daten anzuzeigen. Bestimmte Teile der Seite bzw. Nutzdaten können gezielt aktualisiert werden.

Das Grundprinzip einer Ajax-Webapplikation ist also, dass bei einer Anfrage nicht eine neue Seite zurückgeliefert wird, sondern die Daten aktualisiert werden, wobei diese Abfrage immer vom Client aus erfolgt. Der Server kann selbstständig keine Daten liefern.

Eine typische Anwendung von Ajax sind Suggestions (Suchvorschläge).

1.1.2 Vorbereitung

Für diese Übung wird ebenfalls ein Webserver benötigt, auf dem PHP läuft. Daher wird auch hier die Vagrantbox eingesetzt.

Übung 1:

Starten Sie die Vagrantbox wie in den letzten Übungen mittels `vagrant up`.

1.2 JSON

1.2.1 Über JSON

JSON (**J**ava**S**cript **O**bject **N**otation, gesprochen wie der englische Name „Jason“) ist ein Datenaustauschformat, das in Webanwendungen und mobilen Apps weit verbreitet ist. In diesem Bereich stellt es auch eine oft genutzte Alternative zu XML dar.

JSON-Dokumente sind gültige JavaScript-Dokumente und können daher auch einfach mittels JavaScript verarbeitet werden.

Die Grundstruktur eines JSON-Dokuments (Endung `.json`) ist folgende:

```
{
  "schluessel1": wert1,
  "schluessel2": wert2
}
```

Die **Schlüssel** stehen dabei immer in doppelten Anführungszeichen. Der Name ist frei wählbar, wobei es natürlich trotzdem sinnvoll ist, dieselben Vorgaben zu beachten, wie bei der Vergabe von anderen Variablennamen.

Für die **Werte** sind folgende Datentypen möglich:

- `string`: eine Zeichenkette, die in Anführungszeichen stehen muss
- `number`: eine Zahl
- `boolean`: der boolesche Wert `true` oder `false`
- `array`: ein Array aus Werten oder aus Objekten, das wie folgt aufgebaut ist:

```
{
  "schluessel": [
    {
      "attribut1": wert1,
      "attribut2": wert2
    },
    {
      "attribut1": wert3,
      "attribut2": wert4
    }
  ]
}
```

- `object`: ein JavaScript-Objekt, das auch Kindobjekte und Arrays enthalten kann
- `null`: für fehlende oder leere Werte

Ein Schlüssel-Wert-Paar wird immer durch ein **Komma** getrennt, wenn es sich nicht um das letzte Paar in dem Dokument handelt.

Übung 2:

Schreiben Sie ein JSON-Dokument, das ein Array an Scores (`highscores`) widerspiegelt. Die Scores haben entsprechend der Datenbanktabelle folgende Eigenschaften:

- eine `ID` (Zahl),
- einen `Username` (String),
- einen Punktestand `points` (Zahl).

Speichern Sie dieses für die folgenden Übungen als eigenes JSON-Dokument ab.

1.2.2 Verarbeitung von JSON-Daten

JSON-Daten können in JavaScript mit den Methoden `JSON.stringify()` und `JSON.parse()` verarbeitet werden. Dabei wandelt `JSON.stringify()` die JavaScript-Objekte in einen JSON-String um, während `JSON.parse()` JSON-Dokumente in JavaScript-Objekte umwandelt.

Um JSON-Daten zu erzeugen oder zu lesen, können in PHP die beiden Methoden `json_encode()` sowie `json_decode()` verwendet werden.

Ein Beispiel: Ein JSON-Dokument der Form:

```
var jsonExample = '{
  "studenten": [
    {
      "matrnr": 1,
      "name": "John Smith"
    }
  ]
}'
```

kann wie folgt abgerufen werden:

```
var studenten = JSON.parse(jsonExample).studenten;
```

Anschließend kann, da es sich um Objekte als Array-Inhalte handelt, auch in Objektsyntax (also mit dem Punktoperator) auf die Eigenschaften zugegriffen werden.

1.3 Aufbau einer Ajax-Applikation

Um eine Ajax-Applikation aufzubauen, wird JavaScript-Code benötigt. Hier wird das bereits angesprochene `XMLHttpRequest`-Objekt instanziiert, das für den (asynchronen) Datenaustausch zwischen Client und Server benötigt wird. Eine allgemeingültige Instanziierung, die zum Beispiel in einer Funktion ausgelagert sein kann, sieht wie folgt aus:

```
var httpReq = null;
if (window.XMLHttpRequest) {
  httpReq= new XMLHttpRequest();
} else if (typeof ActiveXObject != "undefined") {
  httpReq= new ActiveXObject("Microsoft.XMLHTTP");
}
```

Übung 3:

Laden Sie sich, falls noch nicht geschehen, die Vorgabedateien herunter.

Erzeugen Sie eine JavaScript-Datei *load_scores.js* , in der in einer Funktion (zum Beispiel `getXMLHttpRequest()`) ein `XMLHttpRequest`-Objekt wie oben gezeigt erzeugt und zurückgegeben wird.

Binden Sie anschließend die neue Datei in der Index-Datei ein.

Nach Erzeugung des Objekts kann dieses genutzt werden, um eine Verbindung zum Server aufzubauen. Dazu wird zunächst im `onreadystatechange`-Attribut des Objekts festgelegt, welche Funktion ausgeführt werden soll, wenn sich die Antwort des Servers „ändert“. Wie genau diese aussieht, wird gleich noch erklärt.

```
1 httpReq.onreadystatechange = function () {
2   console.log("Die Funktion wird bei einer Statusänderung des
      httpReq-Objekts aufgerufen.");
3 };
```

Anschließend kann eine Verbindung mittels `GET` oder `POST` geöffnet werden. Dazu wird die Methode `open(method, file, isAsynchron)` von `httpReq` verwendet. Zuletzt kann die Anfrage mittels `httpReq.send()` gesendet werden.

```
1 httpReq.open("GET", "Scores.json", true);
2 httpReq.send();
```

Im Falle von `GET` wird der Methode `send()` nur `NULL` übergeben, das Vorgehen bei `POST` wird später erläutert.

Übung 4:

Erzeugen Sie eine neue Methode, zum Beispiel `loadAllScores`. Rufen Sie dort die soeben geschriebene Methode zum Erzeugen eines `XMLHttpRequest`-Objekts auf. Rufen Sie dann vom Server die von Ihnen erzeugte JSON-Datei *Scores.json* ab wie oben beschrieben.

Sobald der Server die Anfrage verarbeitet hat, kann die Antwort empfangen werden. Im Zuge dessen finden Statusänderungen statt. Um sicherzustellen, dass die Anfrage vollständig und korrekt bearbeitet wurde, muss zunächst auf eine Statusänderung gewartet werden (also im Folgenden wird innerhalb der `onreadystatechange` zugewiesenen Funktion gearbeitet) und dann sichergestellt werden, dass die Antwort des Servers bereitliegt:

```
1 req.onreadystatechange = function() {
2   if (httpReq.readyState == 4 && httpReq.status === 200){
3     ...
```

```
4     }
5 }
```

Sobald dies der Fall ist, kann mit der Antwort gearbeitet werden. Diese liegt dann im Attribut `httpReq.responseText` vor:

```
1 console.log(httpReq.responseText);
```

Übung 5:

Ergänzen Sie Ihre Methode `loadAllScores()` nun so, dass sie auf die Antwort des Servers wartet und diese dann in der Konsole ausgibt. Die Methode soll beim Laden der Seite aufgerufen werden.

Übung 6:

Nutzen Sie nun die oben beschriebenen Methoden zu Parsen von JSON-Dokumenten. Gehen Sie wie folgt vor:

1. Vergeben Sie dem `tbody`-Element der Highscore-Tabelle eine ID und löschen Sie die bisherigen Inhalte sowie den Datenbankabruf weiter oben auf der Seite.
2. Rufen Sie das Element mit der neuen ID ab, sobald eine Antwort zur Verfügung steht.
3. Geben Sie die Inhalte in Tabellenform aus mittels einer for-Schleife. Beachten Sie, dass SIe zunächst das Array abrufen, dann über die Array-Inhalte iterieren und innerhalb der Schleife auf die einzelnen Objekteigenschaften zugreifen können.

Übung 7:

Erzeugen Sie nun zusätzlich im Highscore-Abschnitt einen einfachen Button, dem Sie die Methode ebenfalls zuweisen. Diese wird später noch benötigt.

1.4 Datenbankdaten in JSON-Format zurücksenden

Aktuell wird ein JSON-Dokument vom Webserver gelesen. Um tatsächlich Daten der Datenbank abzufragen und zurückzuliefern, ist es notwendig, dass eine PHP-Datei abgerufen wird, die die Daten zurückliefert.

Dazu muss zunächst eine neue PHP-Datei erstellt werden mit folgendem Grundgerüst:

```
1 <?php
```

```
2   header("Content-Type: application/json; charset=utf-8");
3
4   echo "Response";
5 ?>
```

Das, was mittels `echo` „ausgegeben“ wird, ist dann die Antwort, die JavaScript mittels `req.responseText` abrufen kann. Der Header muss entsprechend abhängig vom Ergebnistyp gesetzt werden. Da es sich hier um ein JSON-Dokument handeln wird, kann dieser Header übernommen werden.

Übung 8:

Schreiben Sie eine PHP-Datei `get_scores.php` mit dem vorgenannten Grundgerüst. Passen Sie dann die Datei wie folgt an:

1. Bauen Sie hier die Datenbankverbindung auf und fragen alle Score-Einträge ab.
2. Senden Sie die Score-Einträge als JSON-Dokument zurück (Methoden zur Hilfe: `json_encode()` und `get_object_vars()`).
3. Beachten Sie, dass das letzte Element des Arrays nicht mit einem Komma abgeschlossen werden darf.
4. Testen Sie dies. Sie können die PHP-Datei auch direkt über die Vagrantbox-IP abrufen und sich das JSON-Dokument anschauen.
5. Rufen Sie nun die neu erzeugte PHP-Datei statt der JSON-Datei in Ihrer *load_scores.js* auf und testen dies.

1.5 Ajax-Anfragen mit POST

Im vorgenannten Beispiel wurde bereits dargestellt, wie eine Ajax-Anfrage über `GET` geschieht.

Soll die Anfrage über `POST` geschehen, damit der Anfrage Werte übergeben werden können, sieht dies folgendermaßen aus:

```
1 httpReq.open("POST", "auswertung.php", true);
2 httpReq.setRequestHeader("Content-type", "application/
      x-www-form-urlencoded");
3 httpReq.send("firstKey=firstValue&secondKey=secondValue");
```

Die Übergabe erfolgt also als String. Dort werden Key-Value-Paare durch das `&`-Zeichen getrennt.

Im PHP-Code, dem obigen Beispiel folgend in der Datei *auswertung.php*, kann dann auf diese Werte wie bekannt mittels des assoziativen Arrays `$_POST` und dem Namen des

Schlüssels zugegriffen werden. Beispielhaft könnte die PHP-Datei wie folgt aussehen:

```
1 <?php
2   header("Content-Type: text/html; charset=utf-8");
3   echo "<p>Erster Wert: " . $_POST["firstKey"] .
4     "<br />Zweiter Wert: " . $_POST["secondKey"] .
5     "</p>";
6 ?>
```

Hier ist der Header entsprechend der Antwort als Nicht-JSON-Dokument abgeändert.

Übung 9:

Erzeugen Sie die neuen Dateien *save_score.js* und *write_score.php*. und gehen dann wie folgt vor:

1. Übernehmen Sie die `getXMLHttpRequest`-Methode in der JS-Datei.
2. Ergänzen Sie außerdem eine Methode `addScore`, die beim Klicken des „Speichern“-Buttons aufgerufen werden soll (Sie können dieser hierzu eine neue ID vergeben).
3. Kommentieren Sie die beiden `form`-Tags auf der *game.php*-Seite aus, damit die Auswertung nachher nicht mehr über PHP, sondern entsprechend über AJAX geschieht.
4. Die `addScore`-Methode soll nun beim Aufruf die Inhalte der beiden Eingabefelder auslesen und an *write_score.php* mittels POST übertragen.
5. Die *write_score.php*-Datei ruft nun erneut eine Datenbankverbindung auf und fügt einen neuen Datensatz mittels der `add_highscore`-Methode und den beiden übergebenen POST-Parametern hinzu (s. `result.php`).
6. Den Rückgabewert der Methode sendet die PHP-Datei an das JS-Skript zurück.
7. `addScore` sorgt nun dafür, dass, sobald eine Antwort vorliegt und die Antwort gleich „1“ ist (also das Einfügen funktioniert hat), eine Erfolgsmeldung mittels `alert` ausgegeben wird.
8. Außerdem soll die Seite neu geladen werden, damit das Spiel zurückgesetzt wird (`location.reload(true)`).

Dann können Sie die Funktionalität testen. Hinweis: Wenn im Anschluss eine Fehlermeldung der Art `InvalidStateError` angezeigt wird, können Sie diese hier ignorieren. Dies zeigt zwar ein unsauberes Programmieren (in Bezug auf das Neuladen der Seite), sollte die Funktionalität jedoch nicht beeinflussen.