AJAX / Comet Chat Tutorial

== Server zu Client Kommunikation ==
Alles was über das HTTP Protokoll funktioniert geht immer vom Client aus. Der Client fragt etwas an, der Server antwortet. Anders funktioniert es nicht. Man kann nur versuchen einer Server zu Client Kommunikation möglichst nahe zu kommen.

== AJAX ==
Wer bereits etwas Erfahrung mit AJAX hat, der würde einen Chat vermutlich mit einem setInterval programmieren, der kontinuierlich in Abständen von 2-3 Sekunden beim Server nachfragt ob es Änderungen gibt.

Wenn der Client den Server kontinuierlich nach Informationen abfragt, dann nennt man das Pollen. Es ist eine Art Ping Pong und die Technik ist sehr einfach zu implementeren. Beispiele fürs Polling findet ihr hier: [wiki]Mehrere DIV Container mit AJAX aktualisieren[/wiki]

== Problem Polling ==
Das Problem am Polling ist, dass bei 100 gleichzeitigen Besuchern jede Sekunde Sekunde 100 Anfragen gemacht werden müssen, auch wenn gar nichts passiert.
Das erhöht die Last für den Server und auch beim Client vergeht unnötig Latenzzeit für die ganzen erzeugten Verbindungen.

== Comet ==
Für Comet gibt es keine Voraussetzungen, es ist keine Erweiterung oder eine spezielle Programmiersprache. Comet ist das Vorgehen, das es dem Server erlaubt Stück für Stück Informationen an den Client freizugeben währen eine einzige HTTP Verbindung offen gehalten wird.

Der Browser fragt also den Client: "gib mir alle Chat Nachrichten".
Und der Server antwortet: "Hier hast diese..." und nach gewisser Zeit "Hier hast du jene ...".
Er lässt sich mit der Antwort also Zeit und pumpt nach und nach mehr Nachrichten zurück an den Client, die sogar viel neuer sein können, als der Zeitpunkt zu dem der Client zum ersten mal angefragt hat.

=== Beispiel ===
Quellcode

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Comet example 1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
function chat(txt) {
document.getElementById("chat").innerHTML += txt;
}
</script>
</head>
<body>
<div id="chat"></div>
<?php
for($i=0; $i<10; $i++) {
echo '<script type="text/javascript">';
echo 'chat("<p>server is still alive at '.date('Y-m-d H:i:s').'</p>");';
echo '</script>';
// sende die Ausgabe zum Browser
flush();
// warte 0.2 Sekunden um den Server zu entlasten
usleep(200000);
```

```
28. }
29. ?>
30. <script type="text/javascript">
32. chat("<p>server is done</p>");
33. </script>
34. </body>
35. </html>
```

Alles anzeigen

Erweiterung mit Datenbankabfrage
In Zeile #18 könnt ihr nun eure Datenbankabfrage einbauen, die immer wieder prüft ob neue Nachrichten da sind. Wenn keine neuen Nachrichten vorhanden sind, müsst ihr natürlich auch keine Nachricht senden.

=== Demo ===
Ihr findet die Online Demo unter: demo.easy-coding.de/ajax/comet-chat-tutorial/example1.php

=== Einschränkungen #1 ===
Das Beispielssript beendet sich nach 10 Schleifendurchläufen. Ihr könntet es aber mit einer while Schleife unendlich laufen lassen, dann würdet ihr irgendwann einen Fehler bekommen, dass die "max execution time" - die maximale Ausführzeit für ein Script überschritten wurde.

Außerdem habt ihr potentiell immer noch 100 offene Verbindungen während einer Sekunde. Die Anzahl an Verbindungsaufbauten habt ihr aber schon deutlich reduziert.

== AJAX & Comet ==
Ihr könnt AJAX und Comet zusammen benutzen. Das wird so ablaufen, dass der Client nach neuen Nachrichten fragt und der Server immer antwortet, wenn eine neue Nachricht verfügbar ist. Danach schließt er die Verbindung und ein neuer AJAX Request wird gestartet.

Für die AJAX Funktionalität hängen wir die ajax.js ein, die wir hier schon in mehreren Beispielen verwendet haben.

Der Klasse die wir gleich erstellen müssen wir einen DIV Container und eine AJAX Url geben. Außerdem geben wir den Default Zeitstempel an. mit dem wir beginnen.

Quellcode

```
1. var chat = new Chat('chat', 'example2-backend.php', new Date().getTime());
```

Die eigentlich update Funktion muss also, immer wenn sie eine Antwort bekommt (es also neue Nachrichten gibt), einen neuen Request senden. Immer wenn es zu Fehlern kommt, geben wir dem Script 5 Sekunden Aufschub und starten den Request dann erneut.

Die gesamte Klasse und Client Logik sieht damit wie folgt aus:

Quellcode

```
1. function Chat(div, url, lastupdate) {
2. this.div = div;
3. this.url = url;
4. this.lastupdate = lastupdate || null;
6. this.start = function(lastupdate) {
7. this.lastupdate = lastupdate || this.lastupdate;
8. ajaxPost(this.url, 'lastupdate='+ this.lastupdate, function(up) {
9. return function() {
10. if (this.readyState == 4) {
11. // success
```

```
12. if(this.status == 200) {
13. var data = eval('(' + this.responseText + ')');
14. // start next timer, if response is empty, reuse the last known update index
15. up.start(data.lastupdate || up.lastupdate);
16. // fill chat
17. if(data.html) {
18. document.getElementById(up.div).innerHTML += data.html;
19. }
20. }
21. // connection error
22. else {
23. // try again in 5 seconds
24. setTimeout(function(){
25. up.start(up.lastupdate);
26. }, 5000);
27. }
28. }
29. };
30. }(this));
31. };
32. }
```

Alles anzeigen

Serverseitig in der example2-backend.php werden wir nur Antworten wenn wir eine Antwort haben. Wir werden dann HTML Code und den neuen Zeitstempel zurückgeben:

Quellcode

```
1. <?php
2. function getNewMessagesSince($timestamp) {
3. if(rand(1,20) == 5) {
4. return array(
5. 'lastupdate' => time(),
6. 'html' => '<p>server is still alive at '.date('Y-m-d H:i:s').'</p>'
7. );
8. }
9. return false;
10. }
11. $timeout = time() + 10;
12. while(!($row = getNewMessagesSince($_POST['lastupdate'])) && time() < $timeout) {
13. // warte 0.2 Sekunden um den Server zu entlasten
14. usleep(200000);
15. }
16. // liefere den Inhalt
17. echo json_encode($row);
```

Alles anzeigen

=== Demo ===
Ihr findet die Online Demo unter: demo.easy-coding.de/ajax/comet-chat-tutorial/example2.php

== Fazit ==
Das zweite Script antwortet immer, wenn es eine neue Antwort gibt. Bei einem häufig genutzten Chat wäre aber Beispiel eins besser, weil man hier mehrere Nachrichten über eine Verbindung schicken kann.
Auch denkbar wäre den readyState 3 als Stream zu benutzen. Allerdings kann der Internet Explorer dies nicht. Mit einer JavaScript Bibliothek, die für die unterstützen Browser readyState 3 nutzt, und für die Internet Explorers dieser Welt auf

eine Iframe/Ajax Kombination zurück greift, wäre dem AJAX/Comet Chat nichts mehr im Wege

Die beste Lösung ist also die Kombination aller Verfahren. Ich gebe an dieser Stelle keine Referenzimplementierung vor. Ihr solltet mit den hier erworbenen Kenntnissen aber loslegen können um die perfekte Lösung zu implementieren.

Den Download aller Scripte findet ihr hier: demo.easy-coding.de/ajax/comet-chat-tutorial/download.zip

== Weiterführende Wikis ==

- [wiki]Comet Chat Beispiel mit PHP + MySQL[/wiki]