

Ajax: Asynchronous JavaScript and XML



Traditionell übermitteln Webanwendungen Daten vom Client zum Server (z.B. Formulareingaben), die zuvor vom Benutzer ausgefüllt wurden. Der Webserver antwortet, indem er dem Browser des Nutzers eine entsprechend den zuvor übermittelten Formulardaten neu generierte Webseite schickt.

Weil der Webserver bei jeder Anfrage seitens des Nutzers eine neue Webseite erzeugen und übermitteln muss, erscheint die Anwendung dem Benutzer insgesamt als träge und wenig intuitiv, da eine Webanwendung aktualisiert bzw. neu geladen werden muss, bevor die zurückgeschickten Daten vom Server in der Benutzeroberfläche angezeigt werden.

Ajax ist also ein Konzept, das es Webanwendungen ermöglicht, neue Daten vom Server zu erhalten und/oder dem Server für die weitere Verarbeitung zu senden, ohne dass die Seite als Ganzes neu geladen wird.

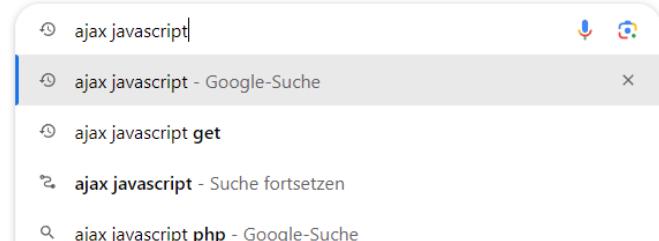
Ajax-Anwendungen sind in der Lage, Anfragen an den Server zu schicken, bei denen nur die Daten angefordert werden, die tatsächlich benötigt werden. Dies geschieht über den Aufruf eines Webservices (REST, SOAP). Der Aufruf erfolgt als asynchrone Kommunikation, d. h. während die Daten vom Server geladen werden, kann der User weiter mit der Oberfläche interagieren. Sind die Daten fertig geladen, dann wird eine zuvor benannte JavaScript-Funktion aufgerufen, die die Daten in die Webseite einbinden kann.

Ein konkretes Beispiel für Ajax:

weiterlesen

Stellen Sie sich vor, Sie lesen einen Artikel auf einer Webseite und am Ende gibt es einen "Mehr anzeigen" oder "weiterlesen"-Button. Ohne AJAX würde ein Klick auf diesen Button die ganze Seite neu laden, um mehr Inhalt zu zeigen. Mit AJAX kann die Webseite stattdessen nur den zusätzlichen Inhalt nachladen, ohne dass man warten muss, bis die ganze Seite sich erneuert. Das macht die User-Experience angenehmer.

Ein weiteres Beispiel für die Nutzung von AJAX ist die Suche bei Google. Wenn der User anfängt, eine Suchanfrage einzugeben, schlägt Google in Echtzeit Suchbegriffe vor, ohne dass die Seite neu geladen wird.



Beschreiben Sie zwei wesentliche Vorteile von Webanwendungen bei denen Ajax zum Einsatz kommt.

Wenn komplettes laden der Seite keinen Sinn macht

Asynchrones laden ermöglicht laden von Daten während der Nutzer weiter arbeiten kann

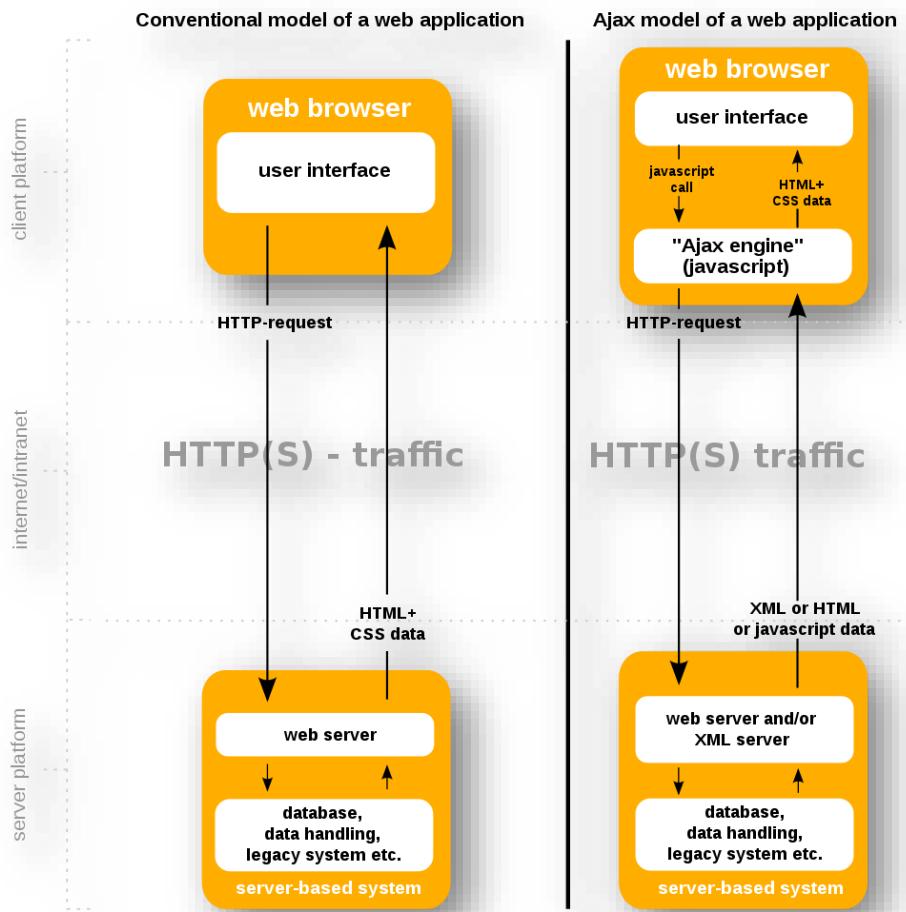


Hauptunterschiede zwischen JSON und XML:

Kriterium	JSON	XML
Format	Textbasiert	Textbasiert
Lesbarkeit	Einfacher für Menschen zu lesen und zu schreiben	Weniger leicht lesbar als JSON wegen der ausführlicheren Markierungen
Datenstruktur	Array und Objekt	Hierarchische Struktur (Baumstruktur) mit Elementen und Attributen
Metadaten	Bietet wenig Unterstützung für Metadaten	Unterstützt Metadaten umfangreich durch Attribute und Namensräume
Kommentare	Unterstützt keine Kommentare	Unterstützt Kommentare
Verarbeitung	Schneller und einfacher zu parsen	Benötigt aufwändigeres Parsen und Verarbeiten
APIs und Bibliotheken	Breite Unterstützung in vielen Programmiersprachen	Breite Unterstützung, aber manchmal komplexer in der Handhabung
Datenaustausch	Bevorzugt für Web APIs und Konfigurationen	Eingesetzt in Web Services (wie SOAP) und komplexen Dokumenten
Erweiterbarkeit	Weniger flexibel in der Erweiterung	Hoch flexibel durch eigene Tags und Schemata
Standardisierung	Standardisiert durch ECMA-404 und RFC 8259	Standardisiert durch das W3C (World Wide Web Consortium)

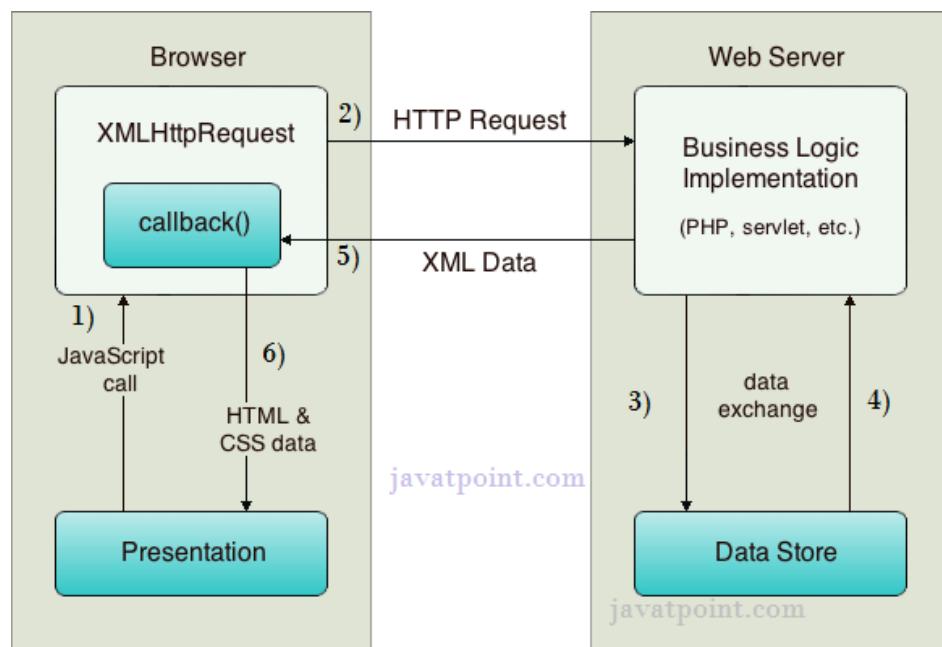
Darstellung 1:

Datenaustausch zwischen Client und Server bei Webanwendungen
(links: ohne AJaX, rechts: mit AJaX)



Darstellung 2:

Der Ablauf eines Datenaustausches mit AJaX



Wie lässt sich AJAX in Webanwendungen implementieren?

Schritt 1: Das XMLHttpRequest-Objekt (XR)

Das XMLHttpRequest-Objekt (kurz **XHR**) ist der Kern der gesamten Ajax-Technologie. Es sorgt für den Datenaustausch im Hintergrund.

Um die Funktionalität des XHR-Objekts nutzen zu können, müssen Sie zuerst über den Konstruktor XMLHttpRequest() eine Instanz erzeugen.

```
<script>
    let ajaxhttp = new XMLHttpRequest();
</script>
```

Damit haben Sie die Instanz eines XMLHttpRequest-Objekts erstellt und können über die Variable ajaxhttp auf die Eigenschaften und Methoden zum asynchronen Informationsaustausch zugreifen.

Schritt 2: Eine http-Anfrage erstellen

Nachdem Sie die Objektinstanz erstellt haben, können Sie eine Verbindung zu der Datei oder dem Skript auf dem Server aufbauen, woher die Daten geliefert werden sollen. Die Antwort des Servers können Sie im Client verarbeiten.

Die Verbindung zum Server definieren Sie über die Methode open(). Das ist aber nur eine Initialisierung und noch keine Datenanforderung

open(Methode, URL, [asynchron])

- Neben der aufzurufenden URL geben Sie als Methode POST oder GET an, mit der die Daten übertragen werden sollen.
- Wie die Kommunikation zwischen Browser und Webserver ablaufen soll, legen Sie mit einem Wert true oder false fest.
- Standardmäßig wird asynchron (true) kommuniziert. Dies hat den Vorteil, dass der Browser nicht blockiert wird, weil auf die Antwort gewartet wird.

Beispiel-Syntax zum Erstellen einer http-Anfrage:

```
ajaxhttp.open("POST", "datei.txt", true);
ajaxhttp.open("GET", "datei.php", false);
```



Ajax-Anfragen dürfen aus Sicherheitsgründen nur Daten von der gleichen Domain nachfordern, von der die anfordernde Webseite stammt (**Same Origin Policy**). Die Vorgehensweise wird Sandkastenprinzip (**Sandbox**) genannt. Andernfalls würde man einen sogenannten **Cross-Domain-Zugriff** durchführen und damit Missbrauch und Manipulation riskieren. Allerdings ist die Einschränkung erheblich, und es gibt einige Anstrengungen, diese Beschränkungen aufzuheben, ohne die Sicherheit zu gefährden.

Schritt 3: Der AjaX-Eventhandler „onreadystatechange“

Da bei einer asynchronen Kommunikation nicht blockierend auf die Antwort gewartet wird, gibt es die Eigenschaft `onreadystatechange`. Dies ist ein Eventhandler des XHR-Objekts. An diesen binden Sie, wie beim Eventhandling üblich, eine beliebige Callback-Funktion.



Funktionen, die einer anderen Funktion als Parameter übergeben werden, nennt man *callback function* – Rückruffunktion. callback-Funktionen werden nach der Ausführung einer anderen Funktion ausgeführt – daher der Name *callback*.

```
/* Variante 1 */
ajaxhttp.onreadystatechange = function() { }

/* oder Variante 2 */
ajaxhttp.onreadystatechange = meineFunktion;
function meineFunktion {
}
```

Die Zustände und die Eigenschaft readyState



Die Callback-Funktion wird immer dann aufgerufen, wenn sich der Zustand des XHR-Objekts ändert. Folgende Zustände gibt es:

Zustand	Beschreibung
0	Das Objekt ist nicht initialisiert.
1	Das Objekt baut eine Verbindung auf.
2	Die Anfrage wurde erfolgreich gesendet.
3	Das Objekt wartet auf die Rückantwort.
4	Das Ergebnis wurde zurückgeliefert.

Eine HTTP-Anfrage durchläuft hierbei immer alle Zustände von 0 bis 4. Im Normalfall müssen Sie nur den Zustand 4 beachten, da erst zu dem Zeitpunkt die zurückgelieferten Daten über verschiedene Eigenschaften abgefragt werden können. Die Zustände stehen in der Eigenschaft `readyState` bereit.

Die Antworteigenschaften

Für die Verwertung der Antwort des Servers gibt es die folgenden Eigenschaften:

Eigenschaft	Beschreibung
<code>responseText</code>	Die Antwort des Servers wird als Zeichenkette bereitgestellt.
<code>responseXML</code>	Die Antwort des Servers wird als XML-Objekt mit DOM-Struktur zurückgeliefert.
<code>status</code>	Der HTTP-Statuscode
<code>statusText</code>	Der Beschreibungstext des Statuscodes

Schritt 4: Die eigentliche http-Anfrage senden

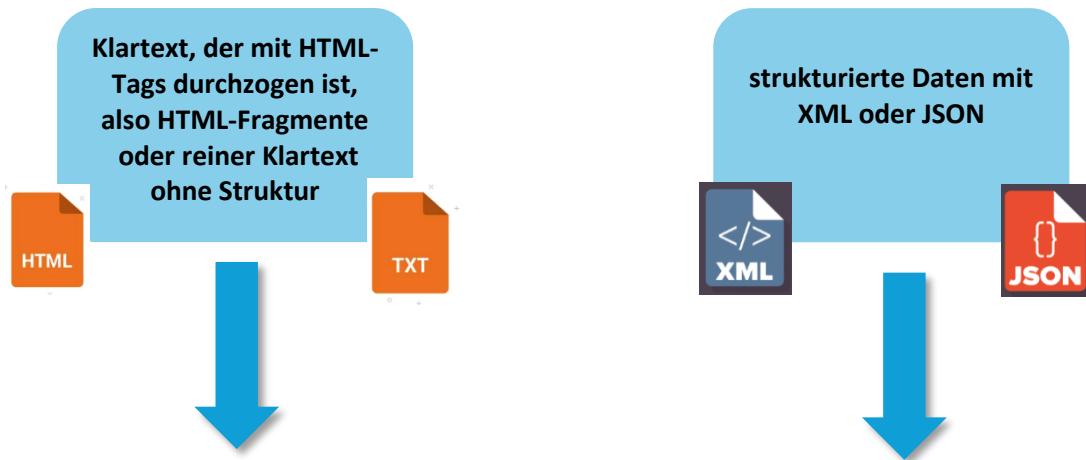
Mit der Methode `send()` senden Sie die definierte Anfrage an den Webserver und starten damit die Anforderung der Daten.

```
ajaxhttp.send(null);
```

Der Parameter der Methode ist bei der Methode GET immer `null` oder leer. Bei POST werden darin eventuell an den Server zu übertragende Daten (etwa Formulareingaben) notiert.

Schritt 4: Das gewünschte Datenformat auswählen

Bei der Datenanforderung mit Ajax fordert man in der Praxis verschiedene Arten an Klartextdaten an:



HTML anfordern (oder reinen Klartext ohne Struktur)

Im Fall von Ajax-Datenanforderungen werden sehr oft HTML-Fragmente vom Webserver angefordert und diese in der Regel im Client einfach per HTML in der Webseite eingebaut. Hierbei werden einfach unter Verzicht auf eine vollständige Seitenstruktur **Texte mit HTML-Tags** gemischt. Meistens erfolgt der Einbau dieser Texte dann ohne eine weitere Verarbeitung der HTML-Fragmente im Client durch JavaScript oder eine andere clientseitige Programmiertechnik. Die Antwort des Webservers wird also vorwiegend unverändert angezeigt und muss vorher auf dem Server bereits in die endgültige Form gebracht werden, die im Client dann verwendet werden soll. **Wenn Sie auf alle HTML-Tags in den Daten verzichten, fordern Sie reinen Klartext an.** Das ist ebenfalls möglich.

Klartext mit Struktur (XML oder JSON)

Wenn Sie anspruchsvollere Applikationen erstellen wollen, können Sie die Antwort des Servers um ein strukturiertes Format erweitern, das im Client gezielt verwertet werden kann. Die Strukturierung der übertragenen Information lässt Ihnen weit mehr Möglichkeiten, um Geschäftsprozesse zu optimieren.

Als Strukturierungsformat kommen JSON oder XML infrage. Schickt der Webserver also XML oder JSON, können Sie im Client auf der Antwort operieren und damit Geschäftslogik zum Client verlagern. Allerdings ist das hauptsächlich bei Ajax eine sehr interessante Möglichkeit. Das direkte Laden dieser Datenstrukturen ist selten sinnvoll.



Beantworten Sie folgende Fragen mit Hilfe der oben dargestellten Informationen.

- 1. Beschreiben Sie den Begriff asynchrone Programmierung im Kontext der Webentwicklung.**

Senden/Empfangen zu unterschiedlichen Zeitpunkten, muss nicht auf laden warten

- 2. JSON und XML sind unterschiedliche Formate zum Austausch von Daten zwischen Server und Client. Nennen Sie je zwei Vorteile für den Einsatz von JSON und XML.**

JSON weniger Overhead, einfacher lesbar

XML unterstützt Kommentare, gut erweiterbar

- 3. Auf welcher Seite einer Client-Server-Anwendung (z.B. Webapplikation) wird die „AJAX-Engine“ implementiert?**

Client

- 4. Welches „Kern-Element“ sorgt für den Datenaustausch im Hintergrund bei AJAX?**

XML-HttpRequest-Object

- 5. Sie fordern eine JSON-Datei per AJAX von einem Server an. Welchen Zustand muss die http-Anfrage haben, damit Sie die gelieferten Daten weiterverarbeiten können?**

Ergebnis wurde zurückgeliefert

- 6. Beschreiben Sie den Unterschied zwischen der Methode open() und send() bei einem AJAX-Serveraufruf.**
