Proseminar Webtechnologien WS2011/12
Ajax und Webentwicklung mit Prototype

Simon Könnecke, André Hacker

Meyerheimstr. 11
10439 Berlin
simon@koennecke-net.de
andrepacker@gmail.com

Abstract: Diese Arbeit diskutiert die Webtechnologie Ajax sowie die JavaScript Bibliothek Prototype. Dabei vergleichen wir Prototype mit der gegenwärtig am weitesten verbreiteten JavaScript Bibliothek jQuery und bewerten somit für welche Einsatzzwecke Prototype geeignet ist. Ergänzend wird eine prototypische Implementierung einer Webanwendung für den Verleih von Fahrrädern vorgestellt, die im Zuge dieser Arbeit unter Verwendung von Prototype und Ajax entwickelt wurde.

Inhaltsverzeichnis

1 Zusammenfassung

Die vorliegende Ausarbeitung im Rahmen des Proseminars Webtechnologien beschäftigt sich mit Ajax, einem Konzept für die asynchrone Datenübertragung zwischen Server und Client, und der JavaScript Bibliothek Prototype, einer plattformunabhängigen Implementierung von Ajax mit weiterer Funktionalität.

Diese Arbeit hat das Ziel, mögliche Einsatzfelder, aber auch die Grenzen und typischen Fallstricke von Ajax zu vermitteln. Darüber hinaus wird mit Prototype eine clientseitige JavaScript Bibliothek für Ajax vorgestellt, mit einer anderen JavaScript Bibliothek verglichen und bewertet. Prototypische Implementierung einer Fahrradverleih-Anwendung auf Basis Prototype mit AJAX.

2 Einleitung

Dieses Kapitel erklärt die Motivation für die Entstehung von Ajax und geht kurz auf die Entstehung von Prototype ein.

Es wird im Folgenden vorausgesetzt, dass die Leserschaft Grundwissen über die wichtigsten Technologien des World Wide Web hat, insbesondere über HTTP, HTML, CSS, XML, JavaScript und das DOM.

2.1 Ajax: Motivation

Mit der Erfindung des World Wide Web im Jahr 1991 wurde im Grunde auch die technologische Grundlage für eine neue Gattung von Anwendungen geschaffen, den sogenannten Webanwendungen. Dieser nicht eindeutig definierte Begriff bezeichnet Anwendungen, die auf einem Webserver und mittlerweile auch auf dem clientseitigem Browser ausgeführt werden, wobei die Interaktion mit dem Nutzer ausschließlich über den Browser erfolgt.[1]

Im vergangenen Jahrzehnt haben Webanwendungen einen unvergleichbaren Durchbruch erfahren. Prominenteste Beispiele hierfür sind soziale Netzwerke sowie Online-Textbearbeitungsprogramme. An dieser Stelle stellt sich die Frage, warum sich Webanwendungen sowohl bei Nutzern als auch bei Softwareentwicklern derart durchgesetzt haben und welche Rolle das Konzept Ajax dabei spielt.

[1] Tobias Kollmann, Web 2.0: Trends und Technologien im Kontext der Net Economy, GWV Fachverlag GmbH, Wiesbaden 2007, Seite 123

Aus Sicht des Entwicklers hat eine Webanwendung gegenüber einer klassischen Desktop-Anwendung den wesentlichen Vorteil, dass sie automatisch plattformunabhängig angeboten werden kann, da sie lediglich einen Browser voraussetzt. Außerdem können Wartungen an der Anwendung jederzeit und einfach auf dem Server durchgeführt werden und stehen sofort allen Nutzern zur Verfügung. Der Nutzer profitiert im gleichen Maße von der Aktualität und der Plattformunabhängigkeit der Anwendung. Er ist darüber hinaus sogar im Stande, die Anwendung auf fremden Computern auszuführen, ohne dass jemals eine Installation notwendig wird. Ebenso hat der Anwender Zugriff auf die enorme Rechenkapazität der Server.

Allerdings hatten traditionelle Webanwendungen lange den Nachteil, dass nach bestimmten Aktionen des Nutzers, z. B. dem Löschen eines Eintrages aus einer Liste, ein Aufruf auf dem Server und somit das erneute Laden und Anzeigen der gesamten Seite nötig war. Während dieser Zeit kann der Nutzer keine anderen Aktionen durchführen und die Anwendung ist nicht bedienbar. Da es die Anwender von Desktop-Anwendungen wie beispielsweise Microsoft Excel gewohnt sind mehrere Aktionen nacheinander ohne Wartezeit auszuführen, haben sie ein derartiges Verhalten bei einer Webanwendung als schlechte Usability empfunden.

Erst durch die schrittweise Einführung der Technologien die hinter dem Konzept Ajax stehen wurde diese Diskrepanz zwischen Webanwendungen und Desktop-Anwendungen weitestgehend aufgelöst. Ajax stellt hierbei eine Kombination dieser Technologien dar, so dass der Browser Anfragen an den Server mittels JavaScript im Hintergrund sendet und abhängig von der Antwort mittels JavaScript und DOM nur Teile der Seite neu lädt. Die Oberfläche im Browser bleibt auf diese Weise bedienbar. Zentrales Ziel von Ajax ist es die Diskrepanz zwischen Web- und Desktop-Anwendungen zu verringern und die Usability derselben zu verbessern, konkret durch die Unterstützung von Nebenläufigkeit und dynamisches Nachladen von Inhalten.[23] Die Vorteile der Webanwendungen kommen so dank Ajax erstmals voll zur Geltung, was wesentlich zur ungeahnten Verbreitung der Webanwendungen im vergangenen Jahrzehnt beigetragen hat.

[2] Tobias Kollmann, Web 2.0: Trends und Technologien im Kontext der Net Economy, GWV Fachverlag GmbH, Wiesbaden 2007, Seite 122
[3] Kai Jäger, Ajax in der Praxis: Grundlagen, Konzepte, Lösungen, Springer-Verlag Berlin Heidelberg, 2008, Seite 9

2.2 Prototype: Geschichte und Motivation

Die JavaScript-Bibliothek Prototype entstand im Rahmen der Entwicklung von Ajax-Funktionalitäten für das Web Application Frameworks Ruby on Rails. Sie wurde alleinig für den clientseitigen Gebrauch entwickelt und 2005 als eigenständige JavaScript Bibliothek veröffentlicht. In den letzten Jahren ist die Nutzung von Prototype stark zurückgegangen, insbesondere da sich jQuery als marktbeherrschende JavaScript Bibliothek etabliert hat.[4] Aus diesem Grund wurde Prototype im April 2011 von jQuery als Standardbibliothek für Ruby on Rails abgelöst.[5]

An dieser Stelle wird deutlich, dass Prototype nicht als Web Application Framework anzusehen ist, wie beispielsweise Ruby on Rails oder GWT, da es mit eigenen Bordmitteln nicht die Realisation von kompletten Webanwendungen ermöglicht. So wird insb. hinsichtlich der Ajax Funktionalität nur der clientseitige Teil abgedeckt, nicht jedoch der serverseitige Teil, der für einen Ajax-Datenaustausch obligatorisch ist.

2.3 Aufbau des Hauptteils

Der folgende Hauptteil beginnt mit einer Erklärung der Bedeutung und Funktionsweise von Ajax. Im Anschluss wird Prototype als Bibliothek vorgestellt, mit jQuery verglichen und bewertet für welche Einsatzgebiete Prototype geeignet ist. Im letzten Teil wird eine eigens mit Prototype realisierte Webanwendung zum Fahrradverleih vorgestellt, wobei der Fokus auf den subjektiv erlebten Schwierigkeiten aber auch positiven Erfahrungen bei der Nutzung von Ajax und Prototype liegt.

[4] http://trends.builtwith.com/javascript, BuildWith.com, JavaScript Libraries and Functions Distribution, Stand 20.12.2011 16:02
[5] http://weblog.rubyonrails.org/2011/4/21/jquery-new-default, jQuery: New Default, Stand 20.12.2011 16:02

3 Hauptteil

3.1 Konzept Ajax

Der Begriff Ajax steht für „Asynchronous JavaScript and XML“ und beschreibt ein Konzept für die Funktionsweise von Webanwendungen. Ajax wurde erstmals 2005 von Jesse James Garrett genannt.[6] Er gab einem Konzept einem Namen, dessen zugrundeliegenden Technologien bereits einige Jahre zuvor existierten und das seinerzeit bereits von vielen Webanwendungen realisiert wurde. So bezieht er sich explizit auf Google Maps und Autovervollständigung während der Google Suche um die Vorzüge von Ajax zu erklären. Wie in unserer Einleitung hergeleitet hat Ajax das Ziel, die Nachteile von traditionellen Webanwendungen zu überwinden, um für Webanwendungen die Bedienerfreundlichkeit zu erreichen, die der Nutzer von Desktop-Anwendungen gewohnt ist:

1. Die Datenübertragung soll asynchron und nebenläufig (im Hintergrund) erfolgen, so dass der Nutzer während der Übertragung weiterhin die Oberfläche bedienen kann (z. B. automatische Speicherung im Hintergrund).

2. Die Reaktionszeit und somit die gefühlte Bedienerfreundlichkeit soll verbessert werden, indem bei einer Aktion des Nutzers nicht die komplette Seite neu übertragen und angezeigt wird, sondern nur der Teil übertragen und neu angezeigt wird, der sich geändert hat.

Ajax nutzt zur Erreichung dieser Ziele eine Kombination aus bestehenden Technologien. Wie in dem Begriff „Asynchronous JavaScript and XML“ deutlich wird, gibt es eine Reihe von Kriterien, was Ajax ausmacht und welche Technologien involviert sind.[7]

1. Der Datenaustausch zwischen dem Client (Browser) und dem Server muss asynchron erfolgen.

2. Clientseitig muss JavaScript ausgeführt werden, insb. für die asynchrone Datenübertragung und die Interaktion mit dem Nutzer sowie für das dynamische Anzeigen von neuen Inhalten.

3. Die zwischen Browser und Server ausgetauschten Daten müssen im XML Format übertragen werden.

Heute existiert keine einheitliche Definition von Ajax. Es lässt sich allerdings sagen, dass die ersten zwei Kriterien maßgeblich sind, wohingegen das dritte Kriterium im Allgemeinen nicht mehr gefordert wird, da die Daten heute häufig in JSON formatiert sind. Hierauf wird später eingegangen.

---

[6] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications, Jesse James Garrett, Ajax: A New Approach to Web Applications, Stand 20.12.2011 16:10

[7] Kai Jäger, Ajax in der Praxis: Grundlagen, Konzepte, Lösungen, Springer-Verlag Berlin Heidelberg, 2008, Seite 7

Abbildung 1 erklärt die technische Funktionsweise von Ajax gemäß der ursprünglichen Definition. Auf die einzelnen Bestandteile wird im Folgenden eingegangen, beginnend bei dem Client.

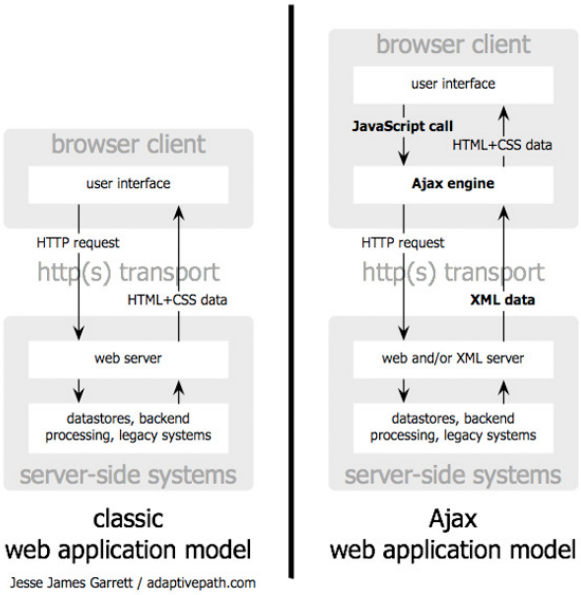

Abbildung 1 Synchrones Verhalten gegenüber asynchronem Verhalten von Webapplikationen[8]

Client und XMLHttpRequest

Eine wesentliche Technologie von Ajax ist die sogenannte XMLHttpRequest API von JavaScript, die es ermöglicht http-Anfragen direkt aus JavaScript an einen Webserver zu senden und die Antworten (z.B. XML, JSON, HTML oder Text) ebenso in JavaScript zu empfangen. Der Browser stellt sicher, dass die http-Anfrage nebenläufig im Hintergrund durchgeführt wird, so dass die Oberfläche weiter bedient werden kann. Ebenso wird die Abfrage in JavaScript asynchron ausgeführt, so dass auch das Skript nicht warten muss bis die Anfrage beantwortet ist, sondern weiter ausgeführt wird und somit insb. auf neue Nutzereingaben reagieren kann. JavaScript und DOM ist die zweite wesentliche Technologie von Ajax, die in diesem Dokument als bekannt vorausgesetzt wird.

[8] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications, Jesse James Garrett, Ajax: A New Approach to Web Applications, Stand 20.12.2011 16:11

Die XMLHttpRequest API wurde von Microsoft für die Nutzung in der Webanwendung Outlook Web Access[9] erfunden und im Jahr 1999 mit dem Internet Explorer 5.0 erstmals als eigenes JavaScript Objekt veröffentlicht. Nachdem sich die API großer Beliebtheit erfreute wurde sie kurz darauf in allen gängigen Browsern implementiert, jedoch mit unterschiedlicher Schnittstelle. Daher setzt sich das W3C für eine Standardisierung ein und hat bereits einen Standardisierungsvorschlag abgegeben.

Übertragungsformat zwischen Browser und Server

XML als Übertragungsformat hat den Nachteil, dass es einen großen Overhead besitzt. Gerade weil bei Ajax-Webanwendungen häufig kleine Nachrichten ausgetauscht werden müssen ist dies ein großer Nachteil. Aus diesem Grund wird heute meist der offene Standard JSON „JavaScript Object Notation“ verwendet, der wesentlich weniger Overhead mit sich bringt. JSON ist eine Untermenge der JavaScript Objekt-Literal Notation, die es ermöglicht JavaScript Objekte zu definieren. Somit ist JSON formatierter Text zugleich gültiger JavaScript Code und kann theoretisch direkt über eval() zu einem Objekt ausgewertet werden, wovon jedoch aus Sicherheitsgründen abzuraten ist.

Serverseitige Logik

Im heutigen Sprachgebrauch trifft Ajax keinerlei Aussagen über den Aufbau der Server-Komponente. Voraussetzung ist lediglich, dass der Server das http-Protokoll und das verwendete Übertragungsformat unterstützt. In den meisten Fällen werden Ajax-Anfragen mittels einer serverseitigen Skriptsprache verarbeitet. Die Datenhaltung und Session-Verwaltung erfolgt über eine serverseitige Datenbank.

3.2 Ajax Nebenwirkungen und Lösungsansätze

Mit der Verwendung von Ajax gehen einige wesentliche Nachteile einher. Auf diese gehen wir im Einzelnen ein, da jeder der Ajax nutzt, diese Fallstricke sowie mögliche Lösungen kennen sollte. Lösungsansätze für alle Probleme sind im Internet dokumentiert und werden daher nur kurz angesprochen.

Nachteil Ajax: Zurück Button

Klickt der Nutzer auf den Zurück Button des Browsers, ruft dieser die Seite die an letzter Stelle in seiner Historie steht auf. Die Browser-Historie enthält jedoch nur Aufrufe von kompletten Webseiten und nicht die im Hintergrund ausgeführten Ajax-Anfragen, die häufig zur Anzeige von neuen Seiten (aus Nutzersicht) führen. Somit wird mit dem Zurück Button häufig weiter zurückgegangen, als es der Nutzer erwartet.

Ein Lösungsansatz ist die Manipulation der Browser-Historie mittels JavaScript und sogenannten Inline Frames (Iframes).

[9] http://www.alexhopmann.com/xmlhttp.htm, Alex Hopmann, The story of XMLHTTP, Stand 21.11.2011 15:37

Nachteil Ajax: Lesezeichen

Nutzer wollen die Möglichkeit haben, ein Lesezeichen zu speichern, d. h. anhand einer URL zu einem speziellen Zustand zurückzukehren. Da sich jedoch die Adresse durch Anfragen im Hintergrund nicht verändert, repräsentiert eine URL in einer Ajax-Anwendung häufig mehrere Zustände.

Lösungsansätze sind die dynamische Aktualisierung des Ankers (eng. fragment identifier) in der URL-Adresse. Dies ist der Teil der hinter der Raute (#) steht.

Nachteil Ajax: Polling Problem

Die Serverkomponente hat mit den zugrundeliegenden Technologien http und TCP/IP keine Möglichkeit, eine Nachricht an den Client zu schicken (Server Push), wenn dieser nicht zuvor eine Anfrage gestellt hat. Somit muss z. B. im Falle einer Ajax basierten Chat-Anwendung der Client regelmäßig den Server anfragen, ob neue Chat-Nachrichten verfügbar sind, was man Polling nennt und generell zu vergleichsweise vielen Nachrichten führt. So ergibt sich ein Trade-off zwischen der Schnelligkeit der Anzeige (latency) und der Serverauslastung (bzw. traffic).

Der häufigste Lösungsansatz ist es, serverseitig nach einer Anfrage des Clients die Verbindung nicht zu schließen und die Antwort nicht vollständig zurückzusenden, so dass der Browser „denkt“, es würden noch Daten übertragen werden. Dieser und weitere Ansätze werden unter dem Begriff Comet Programming in Web diskutiert.

Nachteil Ajax: Indizierung für Suchmaschinen und Barrierefreiheit

Die Indizierung der Inhalte von Ajax-Webanwendungen ist erschwert, da Suchmaschinen nicht einfach den Links folgen können, sondern JavaScript simulieren müssen um an sämtliche dynamisch geladenen Inhalte zu gelangen. Damit geht einher, dass auch die Barrierefreiheit von Ajax-Webanwendungen verschlechtert ist, da beispielsweise Screen Reader für Sehbehinderte häufig kein JavaScript verstehen und somit nicht an alle Inhalte gelangen.[10]

Ein Lösungsansatz ist es, die Inhalte in einer statischen Version zu senden sofern der Client (d. h. Browser, Suchmaschine oder Screen Reader) kein JavaScript unterstützt. Ein weiterer ist es, auf separaten Seiten die dynamischen Inhalte zu beschreiben und auf die dynamische Seite zu verweisen

[10] http://www.sitepoint.com/ajax-screenreaders-work/, James Edwards, Ajax and Screenreaders: When Can it Work?, Stand 21.12.2011 16:30

Nachteil Ajax: Höhere Anforderungen an den Client

Ajax Webanwendungen setzen sowohl JavaScript als auch die XMLHttpRequest API auf dem Client voraus. Insbesondere alte Browser und Browser auf Geräten mit eingeschränkter Geschwindigkeit (z. B. Smartphones) oder ohne JavaScript haben daher teilweise keinen Zugriff auf Ajax-intensive Webseiten. [11]

Die Lösungsansätze für „Indizierung für Suchmaschinen und Barrierefreiheit“ können hier ebenso verwendet werden. Für Mobiltelefone gibt es spezielle Browser, z. B. Opera Mini, die JavaScript auf dem Server ausführen und somit eine eingeschränkte JavaScript Unterstützung für nicht JavaScript fähige Browser bieten. Da die Inhalte vorher entsprechend komprimiert werden, reduziert dies darüber hinaus den Datenverkehr.

Nachteil Ajax: Transparenz

Da Anfragen an den Server völlig im Hintergrund ausgeführt werden und abhängig von der Internetverbindung einige Sekunden in Anspruch nehmen können, ist es für den Nutzer temporär nicht ersichtlich, ob die gewünschte Aktion ausgeführt wird. Daher sollte direkt nach der Aktion des Nutzers für die Dauer der Anfrage eine entsprechende Rückmeldung an den Nutzer gegeben werden, wie beispielsweise ein sich bewegendes Wartesymbol.

3.3 MVC mit Ajax

Als weiterer Nachteil von Ajax könnte angeführt werden, dass dem Entwickler überlassen bleibt, wie er seine Ajax-Webanwendung strukturiert. Das Konzept sieht beispielsweise keine Trennung von Oberfläche und Logik vor und der Entwickler hat die Freiheit, aber auch die Last, selber für eine Trennung zu sorgen. Ebenso muss er überlegen, wie viel Logik auf dem Server, und wie viel Logik auf dem Client nötig ist. Aus diesem Grund macht es Sinn, zu überlegen wie man das bewährte Model-View-Controller Entwurfsmuster für Ajax anwenden kann.

Aus den Erfahrungen mit der Entwicklung der eigenen Webanwendung schlagen wir folgende Aufteilung vor.

[11] Lucas Ostermaier, Evaluation and Comparison of Ajax Frameworks Regarding Applicability, GRIN Verlag, Norderstedt 2008, Seite 9

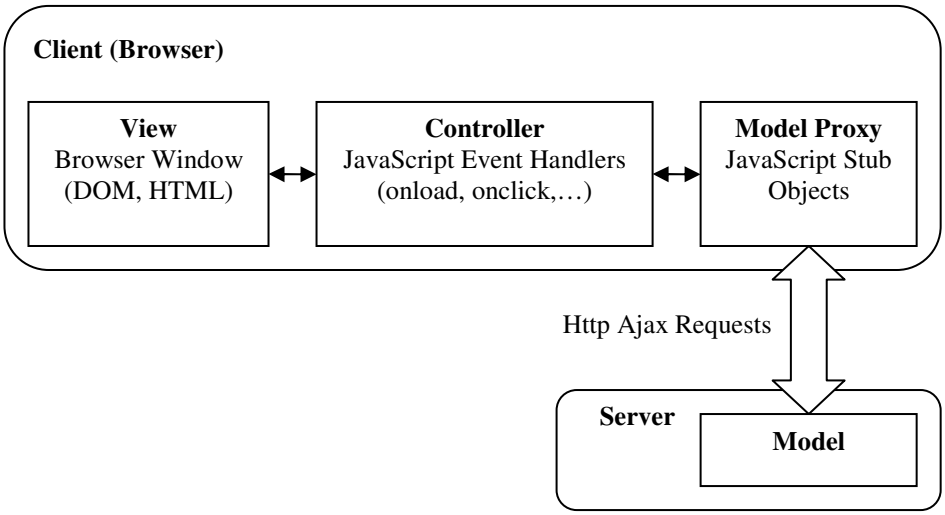

Abbildung 2 Vorschlag für die Realisierung von MVC für Ajax-Webanwendungen

Die **View** wird durch den Inhalt des Browserfensters dargestellt, das heißt insbesondere das DOM.

Die Funktion des **Controllers** übernimmt JavaScript Code, der auf die Interaktion des Nutzers mit der View, wie beispielsweise bei onclick oder onmouseover Events, reagiert. Falls nötig gibt er Veränderungen an das Model weiter, kann aber auch Veränderungen am Model an die View weitergeben.

Das **Model** ist eigentlich auf dem Server anzusiedeln, wo auch die persistente Datenhaltung realisiert ist. Um jedoch auf dem Client eine konsequente MVC-Struktur zu pflegen, kann man ein Proxy Objekt auf dem Client anlegen, das als Stellvertreter für das Model auf dem Server fungiert. Dieser Proxy kommuniziert über Ajax-Requests mit dem tatsächlichen Model auf dem Server.

Noch sinnvoller als diese Diskussion zu führen ist es jedoch, ein bestehendes Web-Framework zu nutzen, das bereits eine feste Struktur vorgibt. Die dadurch entstehende Einheitlichkeit macht es wesentlich einfacher, Projekte arbeitsteilig durchzuführen und sich in fremde Projekte einzuarbeiten.

3.4 Prototype

Wie oben eingeführt, ist Prototype eine Prototype clientseitige JavaScript Bibliothek, die 2005 von den Entwicklern von Ruby on Rails veröffentlicht wurde. Über die Namensgebung ist nichts bekannt, es fällt jedoch auf, dass in JavaScript eine gleichnamige Eigenschaft prototype existiert, die jedes Objekt besitzt. JavaScript ist eine objektorientierte Sprache obwohl JavaScript das Konzept der Klassen nicht kennt. Im modernen Verständnis von objektorientierter Programmierung (OOP) sind aber gerade Klassen maßgeblich für Objektorientiertheit.[12] Gerade über die Eigenschaft prototype wird in JavaScript nun Vererbung und vollständige Objektorientierung realisiert, weshalb wir vermuten, dass die Namensgebung aus dieser Eigenschaft erfolgt ist.

Im Folgenden einige ausgewählte Funktionen[13]:

- **Ajax**
  Die Prototype-Klasse bietet die Möglichkeit Information wie XML, JSON und andere beliege Formate über eine http-Anfrage nebenläufig zu laden.

- **DOM Selektoren**
  Sie werden als Convenience-Funktion zum schnellen Auswählen von HTML-Elementen verwendet. Dabei kann z.B. ID oder CSS-Klasse als Zugriffspunkt gewählt werden.

- **Event Handling**
  Eine Reihe von Funktionen für die eventbasierte Entwicklung werden zur Verfügung gestellt.

- **Class**
  Dieser Bestandteil von Prototype führt das Klassenkonzept in JavaScript ein, basierend auf der JavaScript Eigenschaft prototype.

Prototype-Klassen wie Ajax oder Event bieten für den Entwickler Browserunabhängigkeit beim Erstellen von Webanwendungen.

Analyse und Bewertung

Im folgenden Abschnitt wollen wir Prototype mit jQuery vergleichen. Die einzelnen Kriterien werden in den jeweiligen Spiegelstrichen erläutert.

[12] Kai Jäger, Ajax in der Praxis: Grundlagen, Konzepte, Lösungen, Springer-Verlag Berlin Heidelberg, 2008, Seite 59
[13] http://api.prototypejs.org/, Stand: 28.11.2011 um 16 Uhr

- **Verbreitungsgrad**
  Bertrachtet man die Abbildung 3 wird deutlich, warum wir jQuery als vergleichsbare Bibliothek genommen haben. Prototype hat ein Anteil von ca. drei Prozent und jQuery mit jQuery UI von ca. 50 Prozent.[14] Die Zahlen wurden über die wichtigsten 100.000 Webseiten ermittelt.[15]

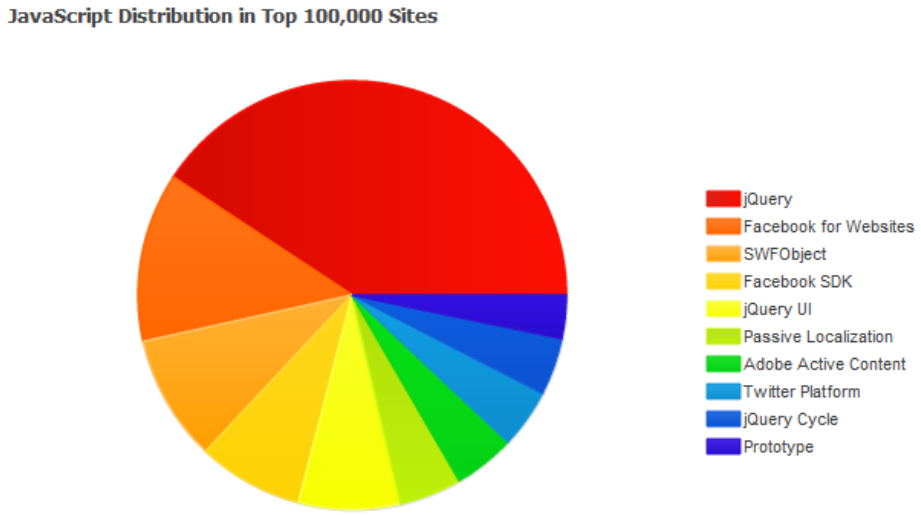

Abbildung 3 Verwendete JavaScript Bibliotheken auf den Top 100.000 Webseiten

- **Einarbeitungsaufwand**
  Durch sehr ähnliche Codesyntax ist der Einarbeitungsaufwand gleichartig. Hierzu ein paar Codebeispiele von Prototype und jQuery.

  Bei Prototype:

```
$("idName").hide();
```

  Bei jQuery:

```
$("#idName").hide();
```

  Bei den Ajax Aufruf werden etwas verschiedene Wege gegangen.
  Bei Prototype:

```
new Ajax.Request('/your/url', {
 onSuccess: function(response) {
   $(“main_content”).addClass("done");
```

[14] http://trends.builtwith.com/javascript, JavaScript Usage Statistics, Stand: 29.11.2011 um 10:23 Uhr
[15] http://trends.builtwith.com/faq.aspx, Stand: 29.11.2011 um 12:11 Uhr

```
  }
});
```

Bei jQuery:

```
$.ajax({
  url: "test.html",
  context: document.body,
  success: function(){
    $("#main_content").addClass("done");
  }
});
```

- **Dokumentation**
  Die Dokumentation von beiden Bibliotheken sind gleichermaßen ausführlich. Die einzelnen Methoden werden kurz erläutert und meist wird ein Beispiel angeführt. Bei jQuery gibt es zusätzlich kleine Demos.[16]

- **Funktionsumfang**
  Der Funktionsumfang der beiden Bibliotheken ist ähnlich. Es kann durch unterschiedliche Lösungsansätze zu Abweichungen kommen.

- **Erweiterungen**
  Bei Prototype gibt es zwei nennenswerte Erweiterungen: Scripty2 (http://scripty2.com) und Scriptaculous (http://script.aculo.us/). Im Gegensatz dazu hat jQuery eine sehr viel länger Liste, siehe dazu das Portal von jQuery Plugins: http://plugins.jquery.com.

- **Bibliotheksgrößen**
  Die Dateigröße ist ein wichtiges Kriterium für die gesamte Ladezeit einer Webseite. Dabei kann das Laden in zwei Phasen eingeteilt werden. Die erste ist der Datentransfer vom Webserver zum Browser und die zweite das Laden im Browser. Da die Browser sehr unterschiedlich sind, betrachten wir nur den Faktor der Dateigröße. Bei Prototype 1.7 liegt die Dateigröße bei 160 KB und bei jQuery 1.5.1 bei 242 KB, wenn es minified ist liegt es sogar nur bei 92 KB. Minified bedeutet, dass der JavaScript Code komprimiert ist. Dies wird dadurch erreicht, dass alle nicht wichtigen Zeichenfolgen weggelassen und Namen gekürzt werden, ohne die Funktionalität zu beeinflussen.[17]

[16] http://matthiasschuetz.com/javascript-framework-matrix/de/, Matthias Schütz, JavaScript Framework Matrix, Stand 30.11.2011 um 9:21 Uhr
[17] Steve Souders, High performance web sites: essential knowledge for frontend engineers, O'Reilly Media, 2007 Sebastopal USA, Seite 69

- **Effizienz**
  Die Effizienz kann man mittels eines Benchmarks feststellen. Bei JavaScript Bibliotheken ist ein elementarer Faktor die clientseitige Hardware, daher können die Ergebnisse von Client zu Client unterschiedlich ausfallen. Hier benutzen wir den Benchmark Slickspeed (http://mootools.net/slickspeed/, Stand: 29.11.2011 um 17:00 Uhr). Die JavaScript Bibliotheken werden auf die Zugriffsgeschwindigkeit der Selektoren getestet. Bei unserem Durchlauf wurden folgende Werte ermittelt:

Selectors	jQuery 1.5.1	Prototype 1.7
…	…	…
final time (less is better)	40	40

Abbildung 4 JavaScript Benchmark von jQuery und Prototype ohne Test-Kriterien.

  Dabei wird deutlich, dass die beiden betrachteten Bibliotheken in diesem Benchmark keine Unterschiede aufweisen, wohingegen andere Bibliotheken deutlich langsamer waren. Dies ist aber erst seit der neuen Prototype-Version der Fall.[18]

Es gibt eine Reihe von weiteren Kriterien wie Ressourcenfreundlichkeit, Browser Feature Detection und vieles mehr. Diese werden in diesem Dokument nicht mehr betrachtet.

Aktuelle Themen – Problem „DOM-Erweiterung“

Die Prototype Bibliothek setzt auf den Ansatz der DOM-Erweiterung, so dass Standardobjekte neue Eigenschaften erhalten. Dies hat große Nachteile gegenüber dem DOM-Wrapper, wird jedoch von vielen Entwicklern als der „objektorientiertere“ Weg gesehen. Mit Prototype 2.0 ist daher eine umfassende Architekturänderung geplant. Aus Sicht vieler Entwickler, die Prototype nutzen, geht damit eines der letzten Alleinstellungsmerkmale gegenüber anderen Bibliotheken verloren.[19]

Zur Verdeutlichung hier der Zugriff auf ein DOM Element mit DOM-Erweiterung:

```
anyDomElement.hide();
```

Und ohne DOM-Erweiterung:

```
Element.hide(anyDomElement);
```

Bewertungsfazit

[18] http://blog.creonfx.com/javascript/mootools-vs-jquery-vs-prototype-vs-yui-vs-dojo-comparison-revised, Peter Velichkov, MooTools vs jQuery vs Prototype vs YUI vs Dojo Comparison Revised, Stand: 29.11.2011 um 18 Uhr
[19] http://perfectionkills.com/whats-wrong-with-extending-the-dom/, What’s wrong with extending the DOM, Stand 21.11.2011 16:21

Die Vergleichskriterien sind im Überblick in Abbildung 5 dargestellt. Für die Bewertung gilt dabei folgende Skala: ++ Sehr gut, + Gut, 0 fast kein Unterschied, - schlecht, -- sehr schlecht.

	jQuery 1.5.1	Prototype 1.7
Dokumentation	++	+
Effizienz von den Selektoren	++	++
Verbreitungsgrad	++	--
Einarbeitungsaufwand	0	0
Funktionsumfang	0	0
Erweiterungen	++	-
Größe der Basis Bibliothek	92 KByte (minified), 242 KByte (uncompressed)	160 Kbyte (uncompressed)

Abbildung 5 Aggregation der Vergleichskriterien mit Wertung

Das Fazit gilt nur für die aktuelle Betrachtung, da beide Bibliotheken weiter entwickelt werden. Somit kann unser Urteil nur sehr eingeschränkt gelten. Sollte jetzt ein Fazit gezogen werden, so ist momentan Prototype 1.7 mit jQuery 1.5.1 hinsichtlich dem oben diskutierten Benchmark, dem Umfang der grundlegenden Funktionen sowie dem Umfang der Dokumentation gleich auf. Bei der Erweiterungen und dem Verbreitungsgrad macht jedoch eindeutig jQuery das Rennen.

Der Ursprung von Prototype ist bei dem Projekt Ruby on Rails anzusiedeln, wobei Ruby on Rails in der neuen Version auf jQuery umgestiegen ist. Des Weiteren wird durch die Umstellung bei Prototype von DOM-Erweiterung auf DOM-Wrapper ein wichtiges Argument weggenommen.

Prototype ist unter dem Strich eine zeitgemäße Bibliothek, deren Community ehemals stark, mittlerweile aber nur noch schwach ausgeprägt ist. Die Zukunft bei clientseitigen JavaScript Bibliotheken würden wir somit bei jQuery sehen und Prototype eher als ein Auslaufmodell bezeichnen.

3.5 Anwendung Fahrradverleih

Anforderung

Unser Ziel war es, eine Webanwendung für einen Fahrradverleih zu erstellen. Es sollen folgende Anforderungen an die Funktionalität erfüllt werden:

- Fahrradsuche über Google Maps, sowie relative Entfernungsangabe zum nächsten Fahrrad
- Fahrrad ausleihen, zurückgeben und reservieren
- Rechnungen und Mahnungen erstellen

- Fahrräder kaufen und verkaufen
- Kategorisierung von Fahrrädern
- einfache Statistiken erstellen

Realisierte Funktionalität

Dieser Abschnitt klärt kurz welche Funktionen implementiert wurden.

- **Navigationsmenüs bezogen auf Rollen des Besuchers:**
  Unterschieden werden die Rollen Besucher, registriere Kunden und Administratoren. Die jeweilige Ansicht wird auf die Rolle angepasst. Die Anpassung der Ansichten ist notwendig für eine benutzergerechte Funktionsfreischaltung.

- **Kaufen und Verkaufen von Fahrrädern:**
  Als Administrator kann man neue Fahrräder aufnehmen und verkaufen.

- **Fahrräder in Kategorien verwalten:**
  Die Kategorien sind zugleich die Preiskategorien der Fahrräder. Beim Anlegen eines Fahrrads kann entschieden werden, welcher Kategorie es zugehört.

- **Kunden**
  Neue Kunden können sich über ein Formular registrieren und bestehende Kunden können im Administrationsbereich verwaltet werden.

- **Fahrrad leihen und zurückgeben:**
  Dem Benutzer werden alle Fahrräder in seiner Nähe angezeigt. Für genauere Information kann der Benutzer auf der Karte die Markierung auswählen. Sollte es das Fahrrad der Wahl sein, kann unmittelbar das Fahrrad geliehen werden. Der Benutzer kann zu einer Zeit nur ein Fahrrad leihen. Für die Rückgabe steht ein eigener Menüpunkt bereit. Nach der Rückgabe steht es den Benutzer wieder frei ein neues Fahrrad zu leihen.

Nicht realisierte Funktionalität:

Bei unserem System wurden folgende Eigenschaften nicht implementiert:

- Relative Entfernungsangabe zum nächsten Fahrrad, da es eine Browserapplikation und keine Mobileapplikation ist. Dadurch ist eine genaue Standortbestimmung nicht möglich, somit ist der Nutzen dieser Funktionalität nicht ausreichend gegeben.

- Reservieren von Fahrrädern, da das Konzept First Come First Served angewendet wurde. Bei einer Reservierung müsste auch ein Ort für die Übergabe festgelegt werden, was in unnötig viel Komplexität resultieren würde.

Technische Implementierung

Die technische Beschreibung geht auf die eingesetzten Frameworks und Technologien ein, die zum Erreichen der Aufgabe verwendet wurden. Der Fahrradverlieh wurde als Webanwendung implementiert, wobei es immer zwei Seiten zu betrachten gibt. Die eine Seite ist der Client, in unserem Fall ein Endbenutzer bzw. Administrator und die andere Seite ist der Server, der die Datenbank und Geschäftslogik bereitstellen soll.

Die **Clientanwendung** ist eine HTML-Seite, die von unterschiedlichen Browsern angesteuert werden kann. Die Gewährleistung, dass die Seite überall gleich erscheint übernimmt ein CSS Design-Toolkit, in unserem Fall Bootstrap von Twitter.[20] Das Toolkit stellt eine Reihe von Grundformen wie Tabellen, Navigation und vieles mehr bereit. Das Bootstrap Toolkit benötigt, um interaktive Komponenten zu ermöglichen, die JavaScript Bibliothek jQuery. Bootstrap wäre ohne jQuery sonst nicht im vollen Umfang nutzbar. Die Serveranfragen werden über Prototype aus der Anwendung heraus gestellt und verarbeitet. Des Weiteren wird die Google Maps API für die Suche von freien Fahrädern verwendet.

Die **Serveranwendung** wurde mit dem JavaScript-Framework nodeJS[21] und der SQL Datenbank MySql geschrieben. Die Verwendung von nodeJS wurde wegen der client- und serverseitigen Nutzung von JavaScript bevorzugt. Der einheitliche Datenaustausch über die Serialisierung in JSON stellt hier einen großen Vorteil dar. Die Server-Anwendung benötigt für die Anbindung an die MySql Datenbank und für den Websever folgende Module:

- Mysql[22]:
  Die Module übernehmen die Verbindung zum MySql-Server und die Implementierung vom Protokoll. Dadurch können wie gewohnt MySql SQL-Anweisungen übermittelt werden.

- Sequelize[23]:
  Sequelize ist eine einfache in JavaScript geschriebene objektrelationale Abbildung (in Englisch: object-relational mapping, ORM). Sie kann einfache Tabellen und rudimentäre Assoziationen darstellen.

- Express[24]:
  Express ist ein HTTP-Server in JavaScript verfasst. Dabei kann man Dateisystemverzeichnisse direkt freigeben und dynamische Seiten einfach einbetten.

[20] http://twitter.github.com/bootstrap/, Stand 20.12.2011 19:33
[21] http://nodejs.org, Stand 20.12.2011 19:33
[22] https://github.com/joyent/node/wiki/#wiki-db-mysql, Stand 20.12.2011 19:33
[23] http://sequelizejs.com, Stand 20.12.2011 19:33
[24] http://expressjs.com, Stand 20.12.2011 19:33

Erfahrungen

Die Erfahrung unserseits mit den einzelnen Bibliotheken und Technologien sind im Groben der Wunsch nach einer umfassenderen und harmonierenden Lösung. Die vielen unterschiedlichen Frameworks und Bibliotheken, die die Arbeit abnehmen sollen, schaffen im Zusammenspiel leider häufiger Mehraufwand. Des Weiteren möchte man sich als Entwickler nicht mit HTML Code beschäftigen da ein Großteil der clientseitigen Funktionen leicht abstrahiert werden könnte. Da jedoch kein Webframework eingesetzt wurde, musste häufig „Copy und Paste“ angewendet werden, da jede Tabelle doch geringfügig anders aussieht. Hier eine Auswahl von konkreten Erfahrungen und Problemen, die bei der Entwicklung aufgetreten sind:

- Prototype hat weniger Erweiterungen als jQuery (jQuery UI, Modal View, Rich Text Editor, etc.).

- Häufig entstand die Notwendigkeit, jQuery einzubinden (Bootstrap, etc.). Dadurch entstand in unser Projekt ein Namenskonflikt mit Prototype bei der Selektor Funktion $().

- Die oben genannten Ajax-Nebenwirkungen wurden aus Zeitgründen so gut wie gar nicht berücksichtigt (Zurück-Button, Lesezeichen, etc.). Sie erfordern expliziten Aufwand, weshalb sie in der Praxis vermutlich oft vernachlässigt werden. Dies ist ein großes Problem von Ajax, weil durch Anwendung des Konzeptes Ajax auf einen Schlag wesentliche Nachteile gegenüber traditionellen Webanwendungen entstehen und das Konzept nicht vorsieht, diese Probleme zu berücksichtigen. Hier machen sich echte Webframeworks bezahlt, die Lösungen für die gängigsten Probleme anbieten.

- Die Trennung zwischen Oberfläche und Logik war gut realisierbar (HTML Templates und JavaScript Code).

- Der HTTP-Server in nodeJS hat andere objektorientierte Ansätze als Prototype, weshalb Code-Wiederverwendung quasi nicht möglich war. Einfach ausgedrückt: JavaScript ist nicht gleich JavaScript.

- Die Übersichtlichkeit des eigenen JavaScript Codes war zunehmend schlecht. Eine Lösung wäre die Aufteilung auf mehrere Dateien.

- Der serverseitig eingesetzte ORMapper Sequelize konnte keine JOINS Realisieren und war somit für Abfragen über mehrere Tabellen (insbesondere Statistik) sehr ungeeignet.

- Es herrscht in JavaScript ein Zwang, wichtige Funktionen (Ajax-Aufrufe oder Datenbank-Abfragen) asynchron zu gestalten, obwohl diese von der Logik wesentlich einfacher sequentiell entwickelt werden könnten.

4 Zusammenfassung

Die Webtechnologie Ajax ist eine Schlüsseltechnologie für moderne Webanwendung und hat denselben zu ungeahntem Durchbruch verholfen. Dabei ist das Konzept denkbar einfach, durch eine JavaScript-Methode Informationen nebenläufig nachzuladen und den Aufruf asynchron zu verarbeiten. Dies ermöglicht das Weiterarbeiten an der Weboberfläche, wie es der Nutzer von Desktop-Anwendungen gewohnt ist.

Wie gezeigt, bedeutet der Einsatz von Ajax jedoch nicht nur, dass man die Vorzüge einer Webanwendung hat. Viele Entwickler vergessen häufig die diskutierten Nebenwirkungen. Das Problem ist also, dass Ajax ein Konzept ist, dass sich nicht um seine „Nebenwirkungen" kümmert und den Entwickler damit alleine lässt. Ebenso ist die heterogene Implementierung bei den Browsern ein Problem von Ajax. Nicht zuletzt aus diesen Gründen sind zahlreiche Webframeworks und JavaScript Bibliotheken entstanden.

Diese Ausarbeitung behandelt die JavaScript Bibliothek Prototype im Detail, die auch in der Testanwendung verwendet wurde. Als Grundlage für die Bewertung wurde ein Vergleich mit jQuery vorgenommen. Wir haben gezeigt, dass die Bedeutung von Prototype stark abgenommen hat, zugunsten der Bibliothek jQuery, die heute aufgrund Ihrer Erweiterbarkeit und ihrem Verbreitungsgrad dominiert. Unsere Prognose ist, dass dieser Trend anhält. Wir bewerten Prototype somit als Auslaufmodell.

Die Webanwendung Fahrradverleih erforderte die Auswahl von zusätzlichen Technologien, insbesondere für das Design der Seite (Bootstrap) und die komplette serverseitige Logik (statischer http-Server, Server für Ajax-Anfragen, Datenbank, Session Handling). Jede der Technologien erforderte Einarbeitungsaufwand, der vermutlich noch höher gewesen wäre, wenn wir serverseitig auch eine neue Programmiersprache gewählt hätten. Im Resultat wurde dennoch in kurzer Zeit eine sehr gut funktionierende Webanwendung geschrieben, die einer Desktopanwendung auf den ersten Blick in nichts nachsteht. Wir haben jedoch gesehen, dass die behandelten Probleme und Nebenwirkungen expliziten Aufwand erfordern, und daher häufig vernachlässigt werden. Zusammenfassend würden wir nur für sehr spezielle Anforderungen ein Vorgehen wie hier empfehlen und in allen anderen Fällen die Nutzung eines Webframeworks empfehlen.

5 Quellenverzeichnis

Literaturverzeichnis:

- Kai Jäger, Ajax in der Praxis: Grundlagen, Konzepte, Lösungen, Springer-Verlag, Berlin Heidelberg 2008

- Lucas Ostermaier, Evaluation and Comparison of Ajax Frameworks Regarding Applicability, GRIN Verlag, Norderstedt 2008

- Steve Souders, High performance web sites: essential knowledge for frontend engineers, O'Reilly Media, 2007 Sebastopal USA

- Tobias Kollmann, Web 2.0: Trends und Technologien im Kontext der Net Economy, GWV Fachverlag GmbH, Wiesbaden 2007

Onlineverzeichnis:

- http://trends.builtwith.com/javascript, BuildWith.com, JavaScript Libraries and Functions Distribution, Stand 20.12.2011 16:02

- http://trends.builtwith.com/faq.aspx, Stand: 29.11.2011 um 12:11 Uhr

- http://weblog.rubyonrails.org/2011/4/21/jquery-new-default, jQuery: New Default, Stand 20.12.2011 16:02

- http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications, Jesse James Garrett, Ajax: A New Approach to Web Applications, Stand 20.12.2011 16:10

- http://www.alexhopmann.com/xmlhttp.htm, Alex Hopmann, The story of XMLHTTP, Stand 21.11.2011 15:37

- http://www.sitepoint.com/ajax-screenreaders-work/, James Edwards, Ajax and Screenreaders: When Can it Work?, Stand 21.12.2011 16:30

- http://api.prototypejs.org/, Stand: 28.11.2011 um 16 Uhr

- http://matthiasschuetz.com/javascript-framework-matrix/de/, Matthias Schütz, JavaScript Framework Matrix, Stand 30.11.2011 um 9:21 Uhr

- http://blog.creonfx.com/javascript/mootools-vs-jquery-vs-prototype-vs-yui-vs-dojo-comparison-revised, Peter Velichkov, MooTools vs jQuery vs Prototype vs YUI vs Dojo Comparison Revised, Stand: 29.11.2011 um 18 Uhr

- http://perfectionkills.com/whats-wrong-with-extending-the-dom/, What's wrong with extending the DOM, Stand 21.11.2011 16:21

- http://twitter.github.com/bootstrap/, Stand 20.12.2011 19:33

- http://nodejs.org, Stand 20.12.2011 19:33

- https://github.com/joyent/node/wiki/#wiki-db-mysql, Stand 20.12.2011 19:33

- http://sequelizejs.com, Stand 20.12.2011 19:33

- http://expressjs.com, Stand 20.12.2011 19:33