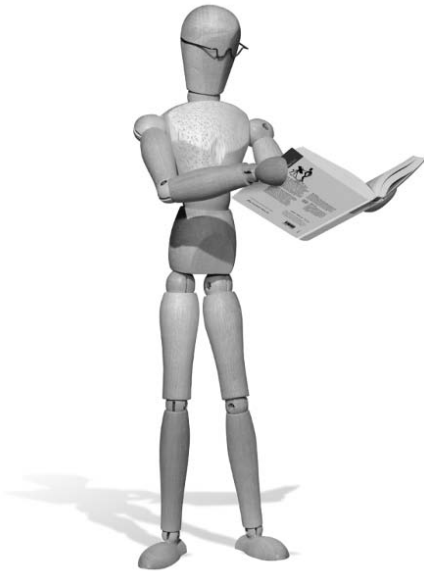


Dave Crane, Eric Pascarello, Darren James

Ajax in action

Das Entwicklerbuch für das Web 2.0



ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Inhaltsverzeichnis

Vorwort	19
Über dieses Buch	21
Zielgruppe	21
Der Inhalt im Überblick	22
Codeschreibweise	25
Codebeispiele zum Herunterladen	25
Das Author-Online-Forum	25
Teil I Ein neuer Blick auf Webanwendungen	27
1 Ein neues Design für das Web	29
1.1 Wozu Rich Clients mit Ajax?	30
1.1.1 Vergleich des Benutzereindrucks	30
1.1.2 Netzwerklatenz	35
1.1.3 Asynchrone Interaktionen	37
1.1.4 Muster der vorherrschenden und nebensächlichen Verwendung ...	40
1.1.5 Das Web mit anderen Augen sehen	41
1.2 Die vier Grundprinzipien von Ajax	42
1.2.1 Der Browser beherbergt eine Anwendung, keinen Inhalt	42
1.2.2 Der Server übermittelt Daten, keinen Inhalt	44
1.2.3 Die Interaktion von Benutzer und Anwendung kann flüssig ablaufen	46
1.2.4 Ajax erfordert Programmierarbeit und Disziplin	48
1.3 Rich Clients mit Ajax in der Praxis	49
1.3.1 Pioniere	49
1.3.2 Google Maps	50
1.4 Alternativen zu Ajax	53
1.4.1 Lösungen mit Macromedia Flash	53
1.4.2 Java Web Start und verwandte Technologien	53
1.5 Zusammenfassung	54
1.6 Quellen	55
2 Erste Schritte mit Ajax	57
2.1 Die Schlüsselemente von Ajax	57
2.2 Den Eindruck mit JavaScript steuern	60

2.3	Das Erscheinungsbild mit CSS festlegen	61
2.3.1	CSS-Selektoren	62
2.3.2	Eigenschaften von CSS-Stilen	64
2.4	Die Ansicht mit dem DOM gliedern	71
2.4.1	JavaScript für die Arbeit mit dem DOM	72
2.4.2	Einen DOM-Knoten finden	75
2.4.3	Einen DOM-Knoten erstellen	77
2.4.4	Stile zu einem Dokument hinzufügen	77
2.4.5	Kleine Abkürzung: Die Eigenschaft innerHTML	79
2.5	Daten mit XML asynchron laden	80
2.5.1	Iframes	81
2.5.2	Die Objekte XmlDocument und XMLHttpRequest	83
2.5.3	Eine Anfrage an den Server senden	85
2.5.4	Callback-Funktionen zur Überwachung von Anfragen	88
2.5.5	Der gesamte Lebenszyklus	89
2.6	Was Ajax auszeichnet	92
2.7	Zusammenfassung	94
2.8	Quellen	95
3	Ordnung schaffen	97
3.1	Ordnung in das Chaos	98
3.1.1	Muster: Ein gemeinsames Vokabular anlegen	99
3.1.2	Refactoring und Ajax	99
3.1.3	Maß halten	100
3.1.4	Refactoring in der Praxis	101
3.2	Fallstudien zum Refactoring	104
3.2.1	Browserinkonsistenzen: Die Muster Fassade und Adapter	104
3.2.2	Ereignishandler: Das Muster Beobachter	108
3.2.3	Handler von Benutzeraktionen wiederverwenden: Das Muster Befehl	111
3.2.4	Nur jeweils ein Verweis auf eine Ressource: Das Muster Singleton	114
3.3	Modell-Präsentation-Steuerung	119
3.4	Webserver-MPS	121
3.4.1	Die Ajax-Webservererschicht ohne Muster	121
3.4.2	Refactoring des Modells für den Geschäftsbereich	125
3.4.3	Inhalt und Darstellung trennen	128

3.5	Bibliotheken und Frameworks von Drittanbietern	132
3.5.1	Browserübergreifende Bibliotheken	132
3.5.2	Komponenten und Komponentensammlungen	137
3.5.3	Anwendungs-Frameworks	140
3.6	Zusammenfassung	143
3.7	Quellen	144
Teil II	Kern Techniken	147
4	Die Seite als Anwendung	149
4.1	Eine andere Art von MPS	149
4.1.1	Das Muster in verschiedenen Maßstäben wiederholen	150
4.1.2	MPS im Browser anwenden	151
4.2	Die Präsentation in einer Ajax-Anwendung	154
4.2.1	Die Logik von der Präsentation fern halten	154
4.2.2	Die Präsentation von der Logik fern halten	160
4.3	Die Steuerung in einer Ajax-Anwendung	164
4.3.1	Klassische JavaScript-Ereignishandler	164
4.3.2	Das Ereignismodell des W3C	168
4.3.3	Ein flexibles Ereignismodell in JavaScript umsetzen	169
4.4	Modelle in einer Ajax-Anwendung	174
4.4.1	Den Geschäftsbereich mit JavaScript modellieren	174
4.4.2	Serverinteraktion	176
4.5	Die Präsentation aus dem Modell gewinnen	178
4.5.1	Reflection für ein JavaScript-Objekt	178
4.5.2	Arrays und Objekte	182
4.5.3	Eine Steuerung hinzufügen	185
4.6	Zusammenfassung	188
4.7	Quellen	189
5	Die Rolle des Servers	191
5.1	Umgang mit der Serverseite	192
5.2	Serverseitige Programmierung	192
5.2.1	Verbreitete Implementierungssprachen	193
5.2.2	N-schichtige Architekturen	193
5.2.3	Client- und serverseitige Geschäftsbereichsmodelle	194

5.3	Der Überblick: Übliche serverseitige Designs	195
5.3.1	Einfache Webserverkodierung ohne Framework	196
5.3.2	Model2-Arbeitsfluss-Frameworks	197
5.3.3	Komponentengestützte Frameworks	198
5.3.4	Dienstorientierte Architekturen	201
5.4	Die Einzelheiten: Daten austauschen	206
5.4.1	Auf den Client beschränkte Interaktionen	206
5.4.2	Einführung in das Beispielprojekt des Planetenbrowsers	207
5.4.3	Die Webseite im Mittelpunkt: Inhaltsbezogene Interaktionen	210
5.4.4	Das Plug-In im Mittelpunkt: Skriptbezogene Interaktionen	213
5.4.5	Die Anwendung im Mittelpunkt: Datenbezogene Interaktionen	219
5.5	Auf den Server schreiben	224
5.5.1	HTML-Formulare	225
5.5.2	Das XMLHttpRequest-Objekt	227
5.5.3	Wirkungsvoller Umgang mit Benutzeraktualisierungen	228
5.6	Zusammenfassung	238
5.7	Quellen	239
Teil III Ajax für Profis		241
6	Der Eindruck auf den Benutzer	243
6.1	Hochwertige Anwendungen erstellen	244
6.1.1	Antwortbereitschaft	244
6.1.2	Stabilität	245
6.1.3	Konsistenz	246
6.1.4	Einfachheit	246
6.1.5	Umsetzung	247
6.2	Den Benutzer auf dem Laufenden halten	247
6.2.1	Antworten auf eigene Anfragen handhaben	248
6.2.2	Aktualisierungen von anderen Benutzern handhaben	250
6.3	Ein Benachrichtigungssystem für Ajax entwerfen	254
6.3.1	Benachrichtigungen modellieren	255
6.3.2	Die Anforderungen an die Benutzerschnittstelle festlegen	257
6.4	Ein Benachrichtigungs-Framework implementieren	259
6.4.1	Symbole in der Statusleiste darstellen	259
6.4.2	Ausführliche Benachrichtigungen ausgeben	261
6.4.3	Das Puzzle zusammensetzen	263
6.5	Das Framework für Anfragen im Netzwerk verwenden	270

6.6	Die Aktualität der Daten anzeigen	274
6.6.1	Einen einfachen Hervorhebungsstil festlegen	275
6.6.2	Hervorhebungen mit der Bibliothek Scriptaculous Effects	277
6.7	Zusammenfassung	278
6.8	Quellen	278
7	Sicherheit	279
7.1	JavaScript und Browsersicherheit	280
7.1.1	Einführung in die Richtlinie des »Ursprungsservers«	280
7.1.2	Wichtige Gesichtspunkte für Ajax	281
7.1.3	Probleme bei Subdomänen	282
7.1.4	Browserübergreifende Sicherheit	283
7.2	Kommunikation mit Remotediensten	284
7.2.1	Proxys für Remotedienste	285
7.2.2	Webdienste	286
7.3	Vertrauliche Daten schützen	297
7.3.1	Man-in-the-middle-Angriffe	297
7.3.2	Secure HTTP verwenden	298
7.3.3	Daten in einfachen HTTP-Verbindungen mit JavaScript verschlüsseln	299
7.4	Ajax-Datenströme schützen	301
7.4.1	Eine sichere Webschicht entwerfen	302
7.4.2	Den Zugriff auf Webdaten einschränken	306
7.5	Zusammenfassung	311
7.6	Quellen	312
8	Leistung	313
8.1	Was ist Leistung?	313
8.2	Die Ausführungsgeschwindigkeit von JavaScript	315
8.2.1	Zeitmessung auf die harte Tour	315
8.2.2	Der Venkman-Profiler	321
8.2.3	Die Ausführungsgeschwindigkeit für Ajax optimieren	323
8.3	Speicherbedarf für JavaScript	336
8.3.1	Speicherlecks vermeiden	336
8.3.2	Besondere Gesichtspunkte für Ajax	340
8.4	Leistungsorientiertes Design	346
8.4.1	Den Speicherbedarf messen	346
8.4.2	Ein einfaches Beispiel	351
8.4.3	Ergebnisse: Speicherbedarf um das 15ofache verringern	356

8.5	Zusammenfassung	359
8.6	Quellen	359
Teil IV Ajax in Beispielen		361
9	Dynamische doppelte Auswahlliste	363
9.1	Ein Skript für eine doppelte Auswahlliste	363
9.1.1	Grenzen einer clientseitigen Lösung	364
9.1.2	Grenzen einer serverseitigen Lösung	365
9.1.3	Lösungen mit Ajax	365
9.2	Die clientseitige Architektur	366
9.2.1	Das Formular entwerfen	367
9.2.2	Die Client/Server-Interaktion entwerfen	369
9.3	Den Server implementieren: VB.NET	370
9.3.1	Das XML-Antwortformat definieren	371
9.3.2	Den serverseitigen Code schreiben	372
9.4	Die Ergebnisse darstellen	375
9.4.1	Das XML-Dokument durchsuchen	375
9.4.2	Stylesheets anwenden	378
9.5	Ergänzende Themen	379
9.5.1	Abfragen mit Mehrfachauswahl	379
9.5.2	Von der doppelten zur dreifachen Auswahlliste	381
9.6	Refactoring	382
9.6.1	Der neue und verbesserte net.ContentLoader	383
9.6.2	Eine Komponente für doppelte Auswahllisten erstellen	388
9.7	Zusammenfassung	396
10	Auto-Vervollständigen	397
10.1	Frameworks für Auto-Vervollständigen	398
10.1.1	Übliche Funktionen für Auto-Vervollständigen	398
10.1.2	Google Suggest	400
10.1.3	Auto-Vervollständigen bei Ajax in Action	401
10.2	Das serverseitige Framework: C#	402
10.2.1	Der Server und die Datenbank	402
10.2.2	Den serverseitigen Code testen	405
10.3	Das clientseitige Framework	406
10.3.1	Der HTML-Code	406
10.3.2	Der JavaScript-Code	407
10.3.3	Zugriff auf den Server	417

10.4	Erweiterte Funktionalität: Mehrere Elemente mit unterschiedlichen Abfragen	428
10.5	Refactoring	429
10.5.1	1. Tag: Den Plan für die Komponente TextSuggest entwickeln	430
10.5.2	2. Tag: TextSuggest erstellen – sauber und konfigurierbar	434
10.5.3	3. Tag: Ajax im Einsatz	438
10.5.4	4. Tag: Ereignisse handhaben	443
10.5.5	5. Tag: Das Popup-Menü für Vorschläge	450
10.5.6	Nachbereitung	458
10.6	Zusammenfassung	458
11	Das verbesserte Ajax-Webportal	461
11.1	Die Evolution der Portale	462
11.1.1	Das klassische Portal	462
11.1.2	Das Portal mit »intelligenter« Oberfläche	463
11.2	Ajax-Portalarchitektur mit Java	465
11.3	Die Ajax-Anmeldung	466
11.3.1	Die Benutzertabelle	466
11.3.2	Der serverseitige Anmeldecode: Java	468
11.3.3	Das clientseitige Anmelde-Framework	471
11.4	DHTML-Fenster implementieren	476
11.4.1	Die Datenbank der Portalfenster	477
11.4.2	Der serverseitige Code des Portalfensters	478
11.4.3	Die externe JS-Bibliothek hinzufügen	483
11.5	Ajax-Funktionen für die automatische Speicherung	485
11.5.1	Die Bibliothek anpassen	485
11.5.2	Informationen automatisch in der Datenbank speichern	488
11.6	Refactoring	491
11.6.1	Den Konstruktor definieren	493
11.6.2	Die Bibliothek AjaxWindows.js anpassen	494
11.6.3	Die Portalbefehle festlegen	496
11.6.4	Die Ajax-Verarbeitung durchführen	500
11.6.5	Nachbereitung	502
11.7	Zusammenfassung	502
12	Live-Suche mit XSLT	505
12.1	Grundlagen zu Suchtechniken	505
12.1.1	Die klassische Suche	506
12.1.2	Die Nachteile der Frame- und Popup-Methoden	507

12.1.3	Live-Suche mit Ajax und XSLT	509
12.1.4	Ergebnisse an den Client senden	510
12.2	Der clientseitige Code	511
12.2.1	Den Client einrichten	511
12.2.2	Den Vorgang auslösen	513
12.3	Der serverseitige Code: PHP	514
12.3.1	Das XML-Dokument erstellen	514
12.3.2	Das XSLT-Dokument erstellen	517
12.4	Das XSLT- und das XML-Dokument kombinieren	520
12.4.1	Microsoft Internet Explorer	522
12.4.2	Mozilla	522
12.5	Die Suchfunktion vervollständigen	524
12.5.1	Ein Stylesheet anwenden	524
12.5.2	Die Suchfunktion verbessern	526
12.5.3	Warum XSLT?	527
12.5.4	Das Lesezeichen-Problem von Ajax umgehen	529
12.6	Refactoring	530
12.6.1	Das XSLHelper-Objekt	530
12.6.2	Eine Komponente für die Live-Suche	535
12.6.3	Nachbereitung	539
12.7	Zusammenfassung	540
13	Eigenständige Anwendungen mit Ajax erstellen	541
13.1	Informationen von außen lesen	542
13.1.1	XML-Feeds im Einsatz	542
13.1.2	Die RSS-Struktur	543
13.2	Die intelligente Oberfläche erstellen	547
13.2.1	Der Ablauf	547
13.2.2	Das HTML-Framework ohne Tabellen	548
13.2.3	Konforme CSS-Formatierung	551
13.3	Die RSS-Feeds laden	556
13.3.1	Globaler Gültigkeitsbereich	557
13.3.2	Die Vorausladefunktion von Ajax	559
13.4	Besondere Übergangseffekte hinzufügen	562
13.4.1	Browserübergreifende Deckkraftregeln	562
13.4.2	Überblendeffekte	563
13.4.3	JavaScript-Timer integrieren	565

13.5	Erweiterte Funktionalität.....	567
13.5.1	Zusätzliche Feeds einfügen	567
13.5.2	Überspringen und Anhalten	570
13.6	Die Einschränkungen überwinden	572
13.6.1	Die Sicherheitseinschränkungen von Mozilla umgehen	573
13.6.2	Den Anwendungsbereich erweitern	576
13.7	Refactoring.....	576
13.7.1	Das Modell des RSS-Readers	576
13.7.2	Die Präsentation des RSS-Readers	579
13.7.3	Die Steuerung des RSS-Readers	584
13.7.4	Nachbereitung	596
13.8	Zusammenfassung	597
Anhang		599
A	Der Werkzeugkasten des Ajax-Profis	601
A.1	Leichter arbeiten mit den richtigen Werkzeugen	601
A.1.1	Die richtigen Werkzeuge zusammenstellen	602
A.1.2	Eigene Werkzeuge erstellen	603
A.1.3	Den Werkzeugkasten pflegen	604
A.2	Editoren und Entwicklungsumgebungen.....	604
A.2.1	Was ein Code-Editor können muss	604
A.2.2	Das derzeitige Angebot	606
A.3	Debugger	611
A.3.1	Wozu Debugger?	611
A.3.2	JavaScript-Debugger	611
A.3.3	HTTP-Debugger	617
A.3.4	Eine eigene browserübergreifende Ausgabekonsole erstellen	619
A.4	DOM-Inspektoren	622
A.4.1	Der Mozilla DOM Inspector	623
A.4.2	DOM-Inspektoren für Internet Explorer	624
A.4.3	Der Safari-DOM-Inspektor für Mac OS X	625
A.5	Firefox-Erweiterungen installieren.....	626
A.6	Quellen.....	629

B	JavaScript für OO-Programmierer	631
	B.1 JavaScript ist nicht Java	631
	B.2 Objekte in JavaScript	633
	B.2.1 Ad-hoc-Objekte erstellen	633
	B.2.2 Konstruktorfunktionen, Klassen und Prototypen	638
	B.2.3 Integrierte Klassen erweitern	641
	B.2.4 Vererbung bei Prototypen	642
	B.2.5 Reflection von JavaScript-Objekten	643
	B.2.6 Interfaces und »Duck Typing«	645
	B.3 Methoden und Funktionen	648
	B.3.1 Funktionen als Bürger erster Klasse	648
	B.3.2 Funktionen zu Objekten hinzufügen	650
	B.3.3 Funktionen von anderen Objekten leihen	650
	B.3.4 Die Ereignisbehandlung in Ajax und der Funktionskontext	652
	B.3.5 Closures in JavaScript	657
	B.3.6 Abschließende Gedanken	660
	B.4 Quellen	661
C	Ajax-Frameworks und -Bibliotheken	663
	Danksagungen	677
	Stichwortverzeichnis	679