Jonathan Chaffer · Karl Swedberg

jQuery lernen und einsetzen

Bessere Webanwendungen mit einfachen JavaScript-Techniken entwickeln

Learning jQuery
Deutsche Ausgabe der 3. engl. Aufl.

dpunkt.verlag

jQuery lernen und einsetzen

Jonathan Chaffer · Karl Swedberg

jQuery lernen und einsetzen

Bessere Webanwendungen mit einfachen JavaScript-Techniken entwickeln

Übersetzung der 3. engl. Auflage

dpunkt.verlag

Lektorat: Dr. Michael Barabas
Copy Editing: Ursula Zimpfer, Herrenberg
Übersetzung und Satz: G&U Language & Publishing Services GmbH, Flensburg, (www.GundU.com)
Herstellung: Nadine Thiele
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Buch 978-3-89864-786-1
PDF 978-3-86491-150-7
ePub 978-3-86491-151-4

1. Auflage 2012

Ringstraße 19B
69115 Heidelberg

First published in the English language under the title »Learning jQuery«, Third Edition

5 4 3 2 1 0

Geleitwort

Ich fühle mich geehrt, dass Karl Swedberg und Jonathan Chaffer die Mühe auf sich genommen haben, »Learning jQuery« zu schreiben. Als erstes jQuery-Buch setzte es den Standard, den andere jQuery-Bücher – und praktisch auch alle anderen JavaScript-Bücher – zu erreichen versuchen. Es ist eines der meistgekauften JavaScript-Bücher, was zu keinem geringen Teil seiner Genauigkeit und seinen vielen Details zu verdanken ist.

Er freut mich besonders, dass Karl und Jonathan dieses Buch geschrieben haben, weil ich sie bereits so gut kannte, dass ich wusste, sie wären die besten für diese Aufgabe. Als Teil des Kernteams von jQuery hatte ich Karl über mehrere Jahre intensiv kennen gelernt und natürlich besonders, während dieses Buch entstand. Wenn ich das Ergebnis betrachte, wird deutlich, dass seine Fähigkeiten als Entwickler und als ehemaliger Englischlehrer hier eine hervorragende Kombination abgegeben haben.

Beide Autoren konnte ich persönlich treffen, was in der Welt verteilter Open-Source-Projekte eher selten ist, und beide sind und bleiben herausragende Mitglieder der jQuery-Gemeinschaft.

Die jQuery-Bibliothek wird von den unterschiedlichsten Personen der jQuery-Community genutzt. Diese Gemeinschaft ist voll von Designern, Entwicklern, Personen mit Programmiererfahrung oder ohne. Auch im jQuery-Team finden sich Personen mit den unterschiedlichsten beruflichen Hintergründen, die alle ihren Beitrag zum Projekt leisten. Eine Sache ist jedoch allen jQuery-Anwendern gemein: Wir sind eine Gemeinschaft von Entwicklern und Designern, die sich wünscht, dass die JavaScript-Entwicklung einfach wird.

Es ist an dieser Stelle fast schon klischeehaft zu sagen, dass ein Open-Source-Projekt gemeinschaftsorientiert ist oder dass es sich darauf fokussiert, neuen Anwendern den Einstieg zu erleichtern. Bei jQuery ist das jedoch keine leere Floskel, sondern die treibende Kraft des Projekts. Es gibt momentan mehr Personen im jQuery-Team, die sich der Gemeinschaft, der Dokumentation und den Plugins widmen, als der Wartung der Codebasis. Obwohl eine korrekte Bibliothek

extrem wichtig ist, macht die Gemeinschaft den Unterschied zwischen einem stockenden, mittelmäßigen und einem Projekt aus, das allen Anforderungen gerecht wird oder sie noch übertrifft.

Wie wir das Projekt betreiben und wie Sie den Code einsetzen, unterscheidet sich deutlich von anderen Open-Source-Projekten und den meisten JavaScript-Bibliotheken. Das jQuery-Projekt und die Community verfügen über weitreichende Sachkunde, wir wissen, warum jQuery eine völlig andere Art zu Programmieren ist, und tun unser bestes, dieses Wissen weiterzugeben.

Die jQuery-Gemeinschaft können Sie nicht durch bloßes Lesen verstehen, Sie müssen daran teilhaben, um vollständig darin einzutauchen. Ich hoffe, Sie haben die Möglichkeit dazu. Nehmen Sie an den Foren, Mailinglisten und Blogs teil, und lassen Sie uns Ihnen dabei helfen, jQuery besser kennen zu lernen.

Für mich ist jQuery viel mehr als ein Haufen Quelltext. Es ist die Summe der über die Jahre angesammelten Erfahrungen, die eine Bibliothek bilden, die beträchtlichen Höhen und Tiefen, die schwierige Entwicklung und die aufregenden Momente, das Projekt wachsen und blühen zu sehen. Und es ist ein Zusammenwachsen der Anwender und der Team-Mitglieder, das Bemühen, sie zu verstehen, und der Versuch, sich anzupassen und zu wachsen.

Als ich dieses Buch erstmals gesehen und seinen Ansatz, jQuery als einheitliches Werkzeug zu erläutern, gelesen habe, war ich verblüfft und angeregt zugleich. Zu erkennen wie andere lernen, verstehen und jQuery so anpassen, wie es für sie richtig ist, macht das Besondere dieses Projekts aus.

Ich bin nicht der einzige, der jQuery von einer anderen Warte als der normalen Werkzeug-Anwender-Beziehung betrachtet. Ich bin nicht sicher, ob ich genau sagen kann, warum, aber ich habe es wieder und wieder beobachtet – der Augenblick, in dem der Anwender zu lächeln beginnt, weil er erkennt, wie viel jQuery ihm helfen kann.

Es gibt diesen besonderen Moment, bei dem es beim Anwender »Klick macht«, wenn er erkennt, dass dieses Werkzeug viel, viel mehr ist als ein einfaches Hilfsmittel. In diesem Augenblick ändert sich auch sein Verständnis völlig, wie dynamische Webanwendungen zu programmieren sind. Das ist eine faszinierende Sache und immer der schönste Augenblick in einem jQuery-Projekt.

Ich hoffe, dass Sie dieses Erlebnis ebenfalls haben werden.

John Resig
Erfinder von jQuery

Über die Autoren

Jonathan Chaffer ist Mitglied der Rapid Development Group, einer Web-Entwicklungsfirma in Grand Rapids, Michigan. Zu seiner Arbeit gehört die Projektüberwachung und Implementierung einer Vielzahl von Technologien mit einem Schwerpunkt auf PHP, MySQL und JavaScript. Er leitet außerdem jQuery-Trainingsseminare für Webentwickler.

In der Open-Source-Gemeinschaft ist Jonathan sehr aktiv beim Drupal-CMS-Projekt, das jQuery als JavaScript-Framework ausgewählt hat. Er ist der Schöpfer des Content Construction Kits, eines beliebten Moduls für die Verwaltung strukturierter Inhalte auf Drupal-Sites. Er ist für die wesentlichen Überarbeitungen im Drupal-Menüsystem und der Entwickler-API verantwortlich.

Jonathan lebt mit seiner Frau Jennifer in Grand Rapids.

Ich möchte Jenny für ihre unermüdliche Begeisterung und Unterstützung danken, Karl für die Motivation weiterzumachen, als es beim Skript langsamer voranging, und der Ars-Technica-Gemeinschaft für ihre fortwährende Inspiration bei diesem technischen Werk. Zusätzlich möchte ich Mike Henry und dem Twisted Pixel-Team für die netten Ablenkungen zwischen meinen Schreibphasen danken.

Karl Swedberg ist Webentwickler bei Fusionary Media in Grand Rapids, Michigan, wo er viel Zeit damit verbringt, tolle Sachen mit JavaScript anzustellen. Als Mitglied des jQuery-Teams ist Karl zuständig für die Pflege der jQuery-API-Site unter api.jquery.com. Er veröffentlicht außerdem Anleitungen in seinem Blog learningjquery.com und präsentiert auf Workshops und Konferenzen. Wenn er nicht gerade programmiert, verbringt Karl gern seine Zeit mit der Familie, röstet in seiner Garage Kaffee und trainiert im örtlichen Fitnessclub.

Ich danke meiner Frau Sara und meinen beiden Kindern Benjamin und Lucia für all die Freude, die sie meinem Leben geben. Dank auch an Jonathan Chaffer für seine Geduld und die gemeinsame Arbeit an diesem Buch.

Vielen Dank auch an John Resig für die beste JavaScript-Bibliothek der Welt und alle anderen, die ihren Code, ihre Zeit und Expertise in dieses Projekt investiert haben. Danke an die Mitarbeiter von Packt Publishing, die technischen Redakteure dieses Buchs, jQuery Cabal und den vielen anderen, die auf dem langen Weg Hilfe und Inspiration geleistet haben.

Die Fachgutachter

Kaiser Ahmed ist professioneller Webentwickler mit einem Bachelor-Grad von der Khula University of Engineering and Technology (KUET). Außerdem ist er Mitbegründer des vollständig auf Outsourcing gestützten Unternehmens CyberXpress.Net Inc. in Bangladesch.

Er verfügt über ein breites Spektrum an technischen Fertigkeiten, Kenntnissen über das Internet und Erfahrung über die ganze Palette der Onlineentwicklung hinweg, die er dazu einsetzt, Onlinepräsentationen für viele Kunden zu erstellen und zu verbessern. Er hat viel Freude an der Gestaltung der Architektur und Infrastruktur von Websites, an der Back-End-Entwicklung mit Open-Source-Werkzeugen (Linux, Apache, MySQL, PHP [LAMP] und andere) und an der Front-End-Entwicklung mit CSS und HTML/XHTML.

Er möchte seiner liebenden Frau Maria Akter für ihre Unterstützung danken.

Kevin Boudloche ist ein Webentwickler aus Mississippi. Webseiten erstellt er als Hobby seit mehr als acht Jahren und professionell seit drei Jahren. Sein Schwerpunkt liegt auf der Entwicklung von Front-Ends und Webanwendungen.

Carlos Estebes ist der Gründer von Ehxioz (*http://ehxioz.com/*), einer jungen Softwareentwicklungsfirma aus Los Angeles, die sich auf die Entwicklung moderner Webanwendungen spezialisiert hat und die neuesten Technologien und Methoden zur Webentwicklung nutzt. Er hat über zehn Jahre Erfahrung als Webentwickler und einen Bachelor-Grad in Informatik von der California State University in Los Angeles.

Inhaltsverzeichnis

Einleitung

Angeregt von Pionieren auf diesem Gebiet wie Dean Edwards und Simon Willison, stellte John Resig 2005 einen Satz von Funktionen zusammen, um den Vorgang zu vereinfachen, durch Programme Elemente auf einer Webseite zu finden und ihnen Verhalten zuzuweisen. Als er sein Projekt im Januar 2006 erstmals der Öffentlichkeit vorstellte, hatte er ihm DOM-Bearbeitungsmöglichkeiten und einfache Animationen hinzugefügt. Er nannte es jQuery, um die zentrale Rolle hervorzuheben, die das Abfragen (*to query*) von Webseiten und ihre Bearbeitung mit JavaScript spielten. In den wenigen Jahren, die seither vergangen sind, wurde der Funktionsumfang von jQuery erweitert und die Leistung verbessert. Viele der beliebtesten Websites im Internet greifen inzwischen auf jQuery zurück. Resig bleibt zwar der leitende Entwickler, doch jQuery ist in echter Open-Source-Manier zu einem Projekt gewachsen, das sich eines Kernteams von JavaScript-Spitzenentwicklern und einer lebendigen Community von Tausenden von Entwicklern rühmen kann.

Die JavaScript-Bibliothek jQuery kann auch Ihre Websites unabhängig von Ihrem Hintergrund aufwerten. Sie bietet eine breite Palette von Funktionen, eine leicht zu erlernende Syntax und eine solide plattformübergreifende Kompatibilität in einer einzigen, kompakten Datei. Überdies wurden Hunderte von Plug-ins entwickelt, um den Funktionsumfang von jQuery zu erweitern und es zu einem unverzichtbaren Werkzeug für praktisch jede clientseitige Skriptaufgabe zu machen.

Dieses Buch gibt eine behutsame Einführung in die Prinzipien von jQuery, damit Sie Ihren Seiten Interaktion und Animationen hinzufügen können – auch wenn frühere Versuche, JavaScript zu schreiben, Sie nur in Verwirrung gestürzt haben. Dieses Buch hilft Ihnen, die Klippen zu umschiffen, die bei Ajax, Ereignissen, Effekten und anspruchsvolleren Merkmalen der Sprache JavaScript lauern. Außerdem fungiert es als kurzes Nachschlagewerk zur Bibliothek jQuery, die sie immer wieder benutzen können.

Der Inhalt dieses Buches

In Kapitel 1, *Erste Schritte*, lernen Sie die JavaScript-Bibliothek jQuery kennen. Das Kapitel beginnt mit einer Beschreibung von jQuery und dem Nutzen für Sie. Anschließend erfahren Sie, wie Sie die Bibliothek herunterladen und einrichten und wie Sie Ihr erstes Skript schreiben.

In Kapitel 2, *Elemente auswählen*, lernen Sie, wie Sie die Selektorausdrücke und DOM-Durchquerungsmethoden von jQuery nutzen, um Elemente auf einer Seite zu finden, wo auch immer sie stecken mögen. Sie verwenden jQuery, um unterschiedliche Seitenelemente zu formatieren, teilweise sogar auf eine Weise, die mit reinem CSS nicht möglich ist.

In Kapitel 3, *Ereignisbehandlung*, nutzen Sie den Ereignisbehandlungsmechanismus von jQuery, um beim Auftreten bestimmter Browserereignisse Verhaltensweisen auszulösen. Sie erfahren, wie Sie mit jQuery auf unaufdringliche und einfache Weise Ereignisse an Elemente anhängen können, selbst wenn die Seite noch nicht vollständig geladen ist. Außerdem erhalten Sie einen Überblick über anspruchsvollere Themen wie Event Bubbling, Delegation und die Verwendung von Namensräumen.

In Kapitel 4, *Formatierung und Animation*, werden die Animationstechniken von jQuery eingeführt. Sie erfahren, wie Sie Seitenelemente mit sowohl nützlichen als auch ästhetischen Effekten ein- und ausblenden und verschieben können.

In Kapitel 5, *DOM-Bearbeitung*, lernen Sie, wie Sie Ihre Seite auf Befehl umgestalten können. Sie erfahren, wie Sie sowohl den Inhalt als auch die Struktur eines HTML-Dokuments im laufenden Betrieb ändern können.

In Kapitel 6, *Daten mit Ajax senden*, lernen Sie die verschiedenen Möglichkeiten kennen, mit denen jQuery den Zugriff auf serverseitige Funktionen ohne hinderliche Seitenaktualisierungen vereinfacht. Nachdem Sie nun die grundlegenden Bestandteile der Bibliothek beherrschen, können Sie sich genauer ansehen, wie Sie sie erweitern können, um sie an Ihre Bedürfnisse anzupassen.

Kaptitel 7, *Plug-ins verwenden*, zeigt Ihnen, wie Sie Plug-ins finden, installieren und verwenden, u.a. die leistungsfähige Plug-in-Bibliothek jQuery UI.

In Kapitel 8, *Plug-ins entwickeln*, lernen Sie, wie Sie die eindrucksvollen Erweiterungsfähigkeiten von jQuery nutzen können, um eigene Plug-ins von Grund auf zu erstellen. Sie legen hier eigene Hilfsfunktionen an, fügen jQuery-Objektmethoden hinzu und sehen sich die Widget-Factory von jQuery UI näher an. Danach beschäftigen wir uns erneut mit den Grundbausteinen von jQuery, um einige anspruchsvollere Techniken zu erlernen.

In Kapitel 9, *Komplexe Selektoren und Durchlaufen des DOM*, erweitern Sie Ihre Kenntnisse über Selektoren und das Durchlaufen des DOM. Hier erwerben Sie die Fähigkeit, die Leistung von Selektoren zu optimieren, den DOM-Elementstack zu bearbeiten und Plug-ins zu schreiben, die die Auswahl- und Durchlaufmöglichkeiten erweitern.

In Kapitel 10, *Komplexe Ereignisse*, beschäftigen Sie sich eingehender mit Techniken wie Delegation und Drosselung, mit denen sich die Leistung der Ereignisbehandlung erheblich verbessern lässt. Außerdem erstellen Sie eigene und besondere Ereignisse, die die Möglichkeiten von jQuery noch erweitern.

In Kapitel 11, *Anspruchsvolle Effekte*, optimieren Sie die grafischen Effekte, die jQuery bietet, indem Sie benutzerdefinierte Easing-Funktionen erstellen und auf jeden Schritt einer Animation reagieren. Hier erhalten Sie die Möglichkeit, Animationen zu bearbeiten, während sie auftreten, und mit benutzerdefinierten Warteschleifen Aktionen nach Zeitplan ablaufen zu lassen.

In Kapitel 12, *DOM-Manipulation für Fortgeschrittene*, vertiefen Sie Ihre praktischen Kenntnisse in der Bearbeitung des DOM mit Techniken wie dem Anhängen beliebiger Daten an Elemente. Aueßrdem lernen Sie, wie Sie die Verarbeitung der CSS-Eigenschaften von Elementen durch jQuery erweitern.

Kapitel 13, *Ajax für Fortgeschrittene*, gibt Ihnen tiefere Einblicke in Ajax-Transaktionen, unter anderem in das jQuery-System für verzögerte Objekte, mit dem Daten gehandhabt werden, die erst später verfügbar werden.

In Anhang A, *JavaScript-Closures*, erhalten Sie solide Grundkenntnisse von Closures in JavaScript. Sie erfahren, worum es sich dabei handelt und wie Sie sie zu Ihrem Vorteil einsetzen können.

In Anhang B, *JavaScript mit QUnit testen*, lernen Sie die Bibliothek QUnit kennen, die für Unit-Tests von JavaScript-Programmen da ist. Diese Bibliothek ist eine wichtige Ergänzung Ihres Werkzeugkastens zur Entwicklung und Wartung anspruchsvoller Webanwendungen.

Anhang C, *Kurzreferenz*, gibt einen Überblick über die gesamte Bibliothek jQuery und führt dabei sämtliche Methoden und Selektorausdrücke auf. Die übersichtliche Gestaltung ist praktisch, wenn Sie wissen, was Sie tun wollen, aber nicht auf den richtigen Namen der gewünschten Methode bzw. des Selektors kommen.

Voraussetzungen für dieses Buch

Um den Beispielcode auszuführen, der in diesem Buch vorgestellt wird, benötigen Sie einen modernen Webbrowser wie Mozilla Firefox, Apple Safari, Google Chrome oder Microsoft Internet Explorer.

Um mit den Beispiel herumzuspielen und die am Kapitelende aufgeführten Übungsaufgaben zu bearbeiten, benötigen Sie außerdem Folgendes:

- Einen einfachen Texteditor
- Webentwicklungstools für Ihren Browser wie z.B. Firebug (siehe Kapitel 1 im Abschnitt »Entwicklungswerkzeuge«)
- Das komplette Codepaket für jedes Kapitel. Darin ist auch eine Kopie der Bibliothek jQuery enthalten (siehe den Abschnitt »Den Beispielcode herunterladen« weiter hinten).

Um einige der Ajax-Beispiele auszuführen, die ab Kapitel 6 vorgestellt werden, brauchen Sie einen Webserver mit aktiviertem PHP.

Zielgruppe dieses Buches

Dieses Buch ist für Webdesigner gedacht, die interaktive Elemente erstellen möchten, und für Entwickler, die die bestmöglichen Benutzerschnittstellen für ihre Webanwendungen gestalten wollen. Grundkenntnisse in JavaScript-Programmierung werden vorausgesetzt. Sie müssen die Grundlagen von HTML und CSS kennen und sollten mit der Syntax von JavaScript vertraut sein. Vorkenntnisse in jQuery oder mit anderen JavaScript-Bibliotheken werden nicht benötigt.

Dieses Buch macht Sie mit dem Funktionsumfang und der Syntax von jQuery 1.6.x vertraut, der zurzeit neuesten Version.

Die Geschichte des Projekts jQuery

Dieses Buch deckt den Funktionsumfang und die Syntax von jQuery 1.6.x ab, der zurzeit neuesten Version. Der Tenor der Bibliothek – eine einfache Möglichkeit zu bieten, um Elemente auf einer Webseite zu finden und sie zu bearbeiten – hat sich im Verlauf der Entwicklung nicht geändert, aber die Einzelheiten der Syntax und der Funktionsmerkmale. Diese kurze Übersicht über die Geschichte des Projekts beschreibt die wichtigsten Änderungen von einer Version zur nächsten, was für Leser, mit älteren Versionen der Bibliothek arbeiten, hilfreich sein mag.

- **Phase der öffentlichen Entwicklung:** Im August 2005 erwähnte John Resig erstmals eine Verbesserung der Prototype-Bibliothek Behavior. Dieses neue Framework wurde am 14. Januar 2006 formal als jQuery veröffentlicht.
- **jQuery 1.0** (August 2006): Dieses erste stabile Release der Bibliothek bot bereits eine solide Unterstützung für CSS-Selektoren, Ereignisbehandlung und AJAX-Interaktion.
- **jQuery 1.1** (Januar 2007): Mit diesem Release wurde die API erheblich verschlankt. Viele selten genutzte Methoden wurden kombiniert, sodass weniger Methoden zu lernen und zu dokumentieren waren.
- **jQuery 1.1.3** (Juli 2007): Dieses untergeordnete Release bot erhebliche Geschwindigkeitssteigerungen der Selektor-Engine von jQuery. Seit dieser Option übertraf die Leistung von jQuery die anderer JavaScript-Bibliotheken wie Prototype, Mootools und Dojo.
- **jQuery 1.2** (September 2007): In diesem Release wurde die XPath-Syntax für die Elementauswahl entfernt, da sie zur CSS-Syntax redundant geworden

war. Die Gestaltung von Effekten wurde in diesem Release flexibler, und die Entwicklung Plug-ins wurde durch die Ergänzung von Ereignis-Namensräumen vereinfacht.

- **jQuery UI** (September 2007): Die neue Plug-in-Suite wurde als Ersatz für das beliebte, aber veraltete Plug-in Interface angekündigt. Es enthielt eine reichhaltige Sammlung vorgefertigter Widgets sowie einen Satz von Werkzeugen zur Gestaltung anspruchsvoller Elemente wie Drag-&-Drop-Oberflächen.
- **jQuery 1.2.6** (Mai 2008): Die Funktionsmerkmale des beliebten Plug-ins Dimensions von Brandon Aaron wurden in die Hauptbibliothek aufgenommen.
- **jQuery 1.3** (Januar 2009): Eine Generalüberholung der Selektor-Engine (Sizzle) bot einen gewaltigen Schub für die Leistung der Bibliothek. Die Ereignisdelegierung wurde jetzt formal unterstützt.
- **jQuery 1.4** (Januar 2010): Diese Version, die wahrscheinlich ehrgeizigste Aktualisierung seit 1.0, brachte viele Leistungsverbesserungen für die DOM-Bearbeitung sowie eine große Menge neuer oder verbesserter Methoden für fast jeden Aspekt der Bibliothek. Das Erscheinen Version 1.4 wurde vierzehn Tage lang mit Ankündigungen und Videos auf einer eigens dazu angelegten Website (*http://jquery14.com/*) begleitet.
- **jQuery 1.4.2** (Februar 2010): Zwei neue Methoden zur Ereignisdelegierung wurden hinzugefügt (`.delegate()` und `.undelegate()`). Das gesamte Ereignissystem von jQuery wurde generalüberholt, um eine flexiblere Benutzung und eine größere browserübergreifende Konsistenz zu erreichen.
- **jQuery Mobile** (August 2010): Das jQuery-Projekt skizzierte öffentlich seine Strategie, seine Forschung und seine Benutzerschnittstellenentwürfe für die mobile Webelemente mit jQuery und einem neuen Mobilframework unter *http://jquerymobile.com/*.
- **jQuery 1.5** (Januar 2011): Die Ajax-Komponenten wurden erheblich umgeschrieben, um die Erweiterbarkeit und Leistung zu verbessern. Außerdem enthielt jQuery 1.5 eine Implementierung des Promise-Musters zur Handhabung von Abfragen von sowohl synchronen als auch asynchronen Funktionen.
- **jQuery 1.6** (Mai 2011): Die Komponente Attribute wurde umgeschrieben, um die Unterscheidung zwischen HTML-Attributen und DOM-Eigenschaften genauer widerzuspiegeln. Außerdem erhielt das in jQuery 1.5 eingeführte Objekt Deferred die beiden neuen Methoden `.always()` und `.pipe()`.

Historische Einzelheiten

Release Notes für ältere jQuery-Versionen sind auf der Projektwebsite unter *http:// jquery.org/history* zu finden.

Schreibweisen

In diesem Buch werden verschiedene Arten von Informationen durch unterschiedliche Formatierung des Textes gekennzeichnet. Im Folgenden finden Sie einige Beispiele für diese Formate und eine Erklärung ihrer Bedeutung.

Einzelne Codebegriffe werden im Text wie folgt dargestellt: »Dieser Code zeigt, dass wir der Methode `console.log()` jede Art von Ausdruck übergeben können.«

Ein ganzer Block von Code sieht folgendermaßen aus:

```
$('button.show-details').click(function() {
   $('div.details').show();
});
```

Um die Aufmerksamkeit auf einen bestimmten Teil eines Codeblocks zu lenken, sind die entsprechenden Zeilen oder Elemente fett hervorgehoben:

```
$('#switcher-narrow').bind('click', function() {
   $('body').removeClass().addClass('narrow');
});
```

Neue Begriffe und *wichtige Wörter* werden kursiv hervorgehoben. Begriffe, die Sie auf dem Bildschirm sehen, also z.B. in Menüs oder Dialogfeldern, werden folgendermaßen gekennzeichnet: »Die Registerkarte CONSOLE, die Sie im folgenden Screenshot sehen, werden wir am häufigsten verwenden, während wir den Umgang mit jQuery erlernen.«

Warnungen, wichtige Hinweise sowie Tipps und Tricks erscheinen in einem Kasten wie diesem.

Herunterladen des Beispielcodes

Die Dateien mit dem Beispielcode können Sie für dieses Buch unter *http://www.dpunkt.de/jquery* herunterladen.

1 Erste Schritte

Das World Wide Web von heute stellt ein dynamisches Umfeld dar und seine Benutzer haben hohe Ansprüche sowohl an die Gestaltung als auch an die Funktion von Websites. Entwickler, die bemerkenswerte, interaktive Sites erstellen wollen, nutzen JavaScript-Bibliotheken wie jQuery, um häufig vorkommende Aufgaben zu automatisieren und komplizierte zu vereinfachen. Einer der Gründe für die Beliebtheit von jQuery besteht darin, dass diese Bibliothek bei einer breiten Palette von Aufgaben Hilfestellung geben kann.

Da jQuery so viele verschiedene Funktionen bietet, erscheint es schwierig, einen Anfang zu finden. Die Bibliothek ist jedoch logisch und konsistent aufgebaut, und viele ihrer Konzepte entstammen den Strukturen von HTML und CSS (Cascading Style Sheets). Die Gestaltung der Bibliothek ermöglicht auch Designern mit geringen Programmierkenntnissen einen schnellen Einstieg, da viele von ihnen mit diesen Technologien mehr Erfahrung haben als mit JavaScript. In diesem ersten Kapitel werden wir tatsächlich ein funktionierendes jQuery-Programm schreiben, das aus lediglich drei Zeilen Code besteht. Auch erfahrenen Programmierern hilft die konzeptuelle Klarheit, wie wir in späteren, anspruchsvolleren Kapiteln noch sehen werden.

Schauen wir uns also an, was jQuery für uns tun kann.

1.1 Was bietet jQuery?

Die Bibliothek jQuery bietet eine Allzweck-Abstraktionsschicht für die gängige Skripterstellung im Web und eignet sich daher für fast jede Situation, in der Sie Skripte einsetzen müssen. Da sie erweiterbar ist und ständig Plug-ins entwickelt werden, um neue Fähigkeiten hinzuzufügen, können wir niemals alle möglichen Verwendungsmöglichkeiten und Funktionen in einem Buch beschreiben. Die Kernfunktionen jedoch helfen bei der Erledigung der folgenden Aufgaben:

- Zugriff auf Elemente in einem Dokument: Ohne JavaScript-Bibliothek müssen Entwickler häufig viele Zeilen Code schreiben, um den *DOM*-Baum (*Document Object Model*) zu durchlaufen und einzelne Teile der Struktur

eines HTML-Dokuments zu finden. Mit jQuery steht den Entwicklern ein robuster und effizienter Selektionsmechanismus zur Verfügung, der es einfach macht, genau den Teil eines Dokuments abzurufen, den Sie untersuchen oder bearbeiten wollen.

```
$('div.content').find('p');
```

- Ändern des Erscheinungsbilds einer Webseite: CSS bietet sich als leistungsfähige Methode an, die Darstellung eines Dokuments zu beeinflussen, leidet jedoch darunter, dass Webbrowser die Standards unterschiedlich oder nur unvollständig unterstützen. Mit jQuery können Entwickler diese Lücke schließen und auf einer browserunabhängigen Abstraktion aufsetzen. Außerdem kann jQuery Klassen und einzelne Formateigenschaften, die einem Teil des Dokuments zugewiesen sind, auch dann noch ändern, wenn dieses bereits vom Browser dargestellt wird.

```
$('ul > li:first').addClass('active');
```

- Ändern des Dokumentinhalts: jQuery ist nicht auf die Modifikation der reinen Darstellung beschränkt, sondern kann auch den Inhalt eines Dokuments ändern, wozu nur wenige Tastendrücke erforderlich sind. So ist es möglich, Text zu ändern, Bilder einzufügen oder auszutauschen, Listen umzusortieren oder die gesamte HTML-Struktur umzustellen und zu erweitern, und das alles mit einer einzigen, einfach zu verwendenden *Anwendungsprogrammierschnittstelle* (*Application Programming Interface*, *API*).

```
$('#container').append('<a href="more.html">more</a>');
```

- Reagieren auf die Aktionen des Benutzers: Selbst die raffiniertesten und leistungsfähigsten Verhaltensweisen nützen nichts, wenn wir nicht steuern können, wann sie stattfinden. Die Bibliothek jQuery bietet eine elegante Möglichkeit, ein breites Spektrum an Ereignissen, z.B. den Klick auf einen Link, abzufangen, ohne den HTML-Code dabei übermäßig mit deren Behandlung zu durchsetzen. Gleichzeitig überbrückt die API zur Ereignisbehandlung Browserunterschiede, die Webentwicklern häufig das Leben schwer machen.

```
$('button.show-details').click(function() {
   $('div.details').show();
});
```

- Animierte Darstellung von Änderungen an einem Dokument: Um solche interaktiven Änderungen wirkungsvoll umzusetzen, muss ein Designer dem Benutzer auch eine optische Rückmeldung geben. Die Bibliothek jQuery ermöglicht dies durch einen reichen Fundus an Effekten wie Fade (Ausblenden) und Wipe (Wegwischen) sowie durch einen Werkzeugkasten zum Gestalten eigener grafischer Effekte.

```
$('div.details').slideDown();
```

- Abrufen von Informationen von einem Server, ohne die ganze Seite aktualisieren zu müssen: Dieses Codemuster ist als Ajax bekannt geworden, was ursprünglich für »asynchronous JavaScript and XML« stand, inzwischen aber eine viel breitere Palette von Technologien für die Kommunikation zwischen Client und Server umfasst. Die Bibliothek jQuery nimmt diesem aufwendigen und sehr vom darstellenden Endgerät abhängigen Vorgang die Komplexität browserspezifischer Aspekte, , sodass sich Entwickler auf die eigentliche Serverfunktionalität konzentrieren können.

```
$('div.details').load('more.html #content');
```

- Vereinfachen häufig vorkommender JavaScript-Aufgaben: Neben all den dokumentspezifischen Merkmalen bietet jQuery auch Erweiterungen zu grundlegenden JavaScript-Konstrukten wie Iteration und Array-Manipulation.

```
$.each(obj, function(key, value) {
   total += value;
});
```

Den Beispielcode herunterladen

Die Dateien mit dem Beispielcode können Sie unter *http://www.dpunkt.de/jquery* herunterladen.

1.2 Warum jQuery so gut funktioniert

Mit dem Wiederaufleben des Interesses an dynamischem HTML kam es zu einer starken Vermehrung der JavaScript-Frameworks. Manche sind spezialisiert und konzentrieren sich nur auf ein oder zwei Aufgaben, während andere versuchen, alle möglichen Verhaltensweisen und Animationen zu katalogisieren und sie vorgefertigt anzubieten. Um das oben angedeutete breite Spektrum an Funktionen zu bieten und gleichzeitig relativ kompakt zu bleiben, werden bei jQuery die folgenden Strategien eingesetzt:

- Nutzen von CSS-Kenntnissen: Indem der Mechanismus zum Auffinden von Seitenelementen auf CSS-Selektoren basiert, erbt jQuery eine knappe und doch lesbare Art, um die Struktur eines Dokuments auszudrücken. Die Bibliothek jQuery ist ein Einstiegspunkt für Designer, die ihren Seiten Verhaltensweisen hinzufügen möchten, da Kenntnisse der CSS-Syntax eine Voraussetzung für die professionelle Webentwicklung sind.
- Unterstützung von Erweiterungen: Um einen schleichenden Funktionszuwachs zu vermeiden, werden bei jQuery Sonderfälle in *Plug-ins* verbannt. Die Methode zum Erstellen neuer Plug-ins ist einfach und gut dokumentiert, was die Entwicklung einer großen Vielfalt origineller und nützlicher Module gefördert hat. Die meisten Funktionen im grundlegenden jQuery-Download sind intern durch die Plug-in-Architektur realisiert und können falls gewünscht entfernt werden, was zu einer noch kleineren Bibliothek führt.

- Wegabstrahieren von Eigenheiten der Browser: Es ist eine bedauernswerte Tatsache in der Webentwicklung, dass jeder Browser seine eigenen Bereiche hat, in denen er von den öffentlichen Standards abweicht. Ein erheblicher Anteil jeder Webanwendung kann ausgelagert werden, um sich um die Merkmale zu kümmern, die auf jeder Plattform anders sind. Während die im ständigen Wandel begriffene Browserlandschaft es unmöglich macht, für bestimmte anspruchsvolle Funktionen vollständig browserneutralen Code zu schreiben, fügt jQuery eine *Abstraktionsschicht* hinzu, die häufig vorkommende Aufgaben normalisiert, den Code erheblich vereinfacht und seinen Umfang verringert.
- Konsequente Verwendung von Mengen: Wenn wir jQuery anweisen: »Finde alle Elemente der Klasse `collapsible` und blende sie aus (»hide them«)«, besteht keine Notwendigkeit, jedes zurückgegebene Element in einer Schleife durchzugehen. Stattdessen sind Methoden wie `.hide()` dazu da, automatisch Mengen von Objekten zu verarbeiten statt Einzelobjekte. Diese Technik, die *implizite Iteration*, bedeutet, dass viele Schleifenkonstrukte unnötig werden, was den Code erheblich verkürzt.
- Zulassen mehrerer Aktionen in einer Zeile: Um die übermäßige Verwendung temporärer Variablen und verschwenderische Wiederholungen zu vermeiden, nutzt jQuery für die meisten seiner Methoden das Programmiermuster der *Verkettung*. Das bedeutet, dass das Ergebnis der meisten Operationen an einem Objekt das Objekt selbst ist, das gleich wieder für die nächste Aktion bereitsteht.

Diese Strategien haben dafür gesorgt, dass das jQuery-Paket schlank bleibt – komprimiert ungefähr 30 KB –, während es gleichzeitig Techniken bereitstellt, die unseren selbst geschriebenen Code, der die Bibliothek nutzt, kompakt halten.

Die Eleganz dieser Bibliothek gründet sich zum Teil auf das Design, zum Teil auf den evolutionären Prozess, den die rund um das Projekt aufgekommene, lebendige Community fördert. Benutzer von jQuery diskutieren nicht nur die Entwicklung von Plug-ins, sondern auch Verbesserungen an der Kernbibliothek. Außerdem tragen sowohl Benutzer als auch Entwickler zu der ständigen Verbesserung der offiziellen Projektdokumentation bei, die Sie unter *http://api.jquery.com* finden können.

Obwohl zur Konstruktion eines so flexiblen und stabilen Systems viel Arbeit erforderlich ist, kann das Endprodukt kostenlos genutzt werden. Dieses Open-Source-Projekt steht unter den beiden Lizenzen *MIT Licence* (zur freien Verwendung von jQuery auf jeder Site und zu seiner Nutzung in proprietärer Software) und *GNU Public Licence* (für die Einbindung in andere Open-Source-Projekte mit GNU-Lizenz) zur Verfügung.

1.3 Unsere erste Webseite mit jQuery

Nachdem wir uns damit befasst haben, welch ein Umfang an Funktionalität in jQuery zur Verfügung steht, sehen wir uns nun an, wie wir die Bibliothek tatsächlich nutzen. Dazu benötigen wir zunächst eine Kopie von jQuery.

1.3.1 jQuery herunterladen

Eine Installation ist nicht erforderlich. Um jQuery verwenden zu können, brauchen wir lediglich eine öffentlich verfügbare Kopie der Datei, die sich auf einer externen Site oder auf unserer eigenen befinden kann. Da JavaScript eine interpretierte Sprache ist, gibt es keine Kompilierung und keine Build-Phase, über die Sie sich Sorgen machen müssten. Wann immer wir auf einer Seite jQuery zur Verfügung haben müssen, verweisen wir einfach in einem `<script>`-Element des HTML-Dokuments auf den Speicherort der Datei.

Die offizielle jQuery-Website (*http://jquery.com/*) beinhaltet stets die aktuellste stabile Version der Bibliothek, die Sie gleich von der Homepage herunterladen können. Es kann sein, dass gleichzeitig mehrere Versionen von jQuery verfügbar sind. Für uns als Websiteentwickler ist die neueste unkomprimierte Version geeignet. In Produktionsumgebungen kann sie durch eine komprimierte Version ersetzt werden.

Angesichts der steigenden Beliebtheit von jQuery haben manche Unternehmen die Datei in ihren *CDNs* (*Content Delivery Networks*) kostenlos zur Verfügung gestellt. Vor allem Google (*http://code.google.com/apis/ajaxlibs/documentation/*) und Microsoft (*http://www.asp.net/ajax/cdn*) bieten die Datei auf leistungsfähigen, verzögerungsarmen Servern überall auf der Welt an, um einen schnellen Download unabhängig vom Standort des Benutzers sicherzustellen. Die CDN-Kopien von jQuery bieten aufgrund der Serververteilung und Zwischenspeicherung zwar Geschwindigkeitsvorteile, doch kann während der Entwicklung die Verwendung einer lokalen Kopie praktischer sein. In den Beispielen in diesem Buch nutzen wir eine Kopie der Datei in unserem eigenen System, sodass wir den Code ausführen können, unabhängig davon, ob wir mit dem Internet verbunden sind oder nicht.

1.3.2 Einrichten von jQuery in einem HTML-Dokument

Die meisten Beispiele für die Nutzung von jQuery bestehen aus drei Teilen: dem HTML-Dokument, der CSS-Datei zu seiner Formatierung und den JavaScript-Dateien, die die Interaktionen definieren. Als erstes Beispiel verwenden wir eine Seite mit dem Auszug aus einem Buch, dessen einzelnen Teilen eine Reihe von Klassen zugewiesen sind. Diese Seite enthält einen Verweis auf die neueste Version der Bibliothek jQuery, die wir heruntergeladen, in `jquery.js` umbenannt und in unserem lokalen Projektverzeichnis platziert haben. Das HTML-Dokument sieht wie folgt aus:

```
<!DOCTYPE html>

<html lang="en">
   <head>
      <meta charset="utf-8">
      <title>Through the Looking-Glass</title>

      <link rel="stylesheet" href="01.css">

      <script src="jquery.js"></script>
      <script src="01.js"></script>
   </head>

   <body>
      <h1>Through the Looking-Glass</h1>
      <div class="author">by Lewis Carroll</div>

      <div class="chapter" id="chapter-1">
         <h2 class="chapter-title">1. Looking-Glass House</h2>
         <p>There was a book lying near Alice on the table,
            and while she sat watching the White King (for she
            was still a little anxious about him, and had the
            ink all ready to throw over him, in case he fainted
            again), she turned over the leaves, to find some
            part that she could read,
            <span class="spoken">
            "&mdash;for it's all in some language I don't know,"
            </span>
            she said to herself.
         </p>
         <p>It was like this.</p>
         <div class="poem">
            <h3 class="poem-title">YKCOWREBBAJ</h3>
            <div class="poem-stanza">
               <div>sevot yhtils eht dna ,gillirb sawT'</div>
               <div>;ebaw eht ni elbmig dna eryg diD</div>
               <div>,sevogorob eht erew ysmim llA</div>
               <div>.ebargtuo shtar emom eht dnA</div>
            </div>
         </div>
         <p>She puzzled over this for some time, but at last
            a bright thought struck her.
            <span class="spoken">
            "Why, it's a Looking-glass book, of course! And if
            I hold it up to a glass, the words will all go the
            right way again."
            </span>
         </p>
         <p>This was the poem that Alice read.</p>
         <div class="poem">
            <h3 class="poem-title">JABBERWOCKY</h3>
            <div class="poem-stanza">
               <div>'Twas brillig, and the slithy toves</div>
```

```
                <div>Did gyre and gimble in the wabe;</div>
                <div>All mimsy were the borogoves,</div>
                <div>And the mome raths outgrabe.</div>
            </div>
        </div>
    </div>
  </body>
</html>
```

Dateipfade

Das tatsächliche Layout der Dateien auf dem Server spielt keine Rolle. Verweise von einer Datei auf eine andere müssen nur auf die gewählte Struktur angepasst werden. In den meisten Beispielen in diesem Buch verwenden wir für Verweise auf Dateien relative Pfade (`../images/foo.png`) und keine absoluten (`/images/foo.png`). Dadurch kann der Code lokal ausgeführt werden, ohne dass ein Webserver nötig ist.

Unmittelbar nach der normalen HTML-Präambel wird das Stylesheet geladen. Für dieses Beispiel verwenden wir das folgende:

```
body {
   background-color: #fff;
   color: #000;
   font-family: Helvetica, Arial, sans-serif;
}
h1, h2, h3 {
   margin-bottom: .2em;
}
.poem {
   margin: 0 2em;
}
.highlight {
   background-color: #ccc;
   border: 1px solid #888;
   font-style: italic;
   margin: 0.5em 0;
   padding: 0.5em;
}
```

Nach dem Verweis auf das Stylesheet werden die JavaScript-Dateien eingeschlossen. Es ist wichtig, dass sich das `‹script›`-Tag für die Bibliothek jQuery vor den Tags für unsere eigenen Skripte befindet, da das jQuery-Framework sonst nicht verfügbar wäre, wenn unser Code versucht, darauf zu verweisen.

Im weiteren Verlauf dieses Buches werden immer nur die jeweils relevanten Teile der HTML- und CSS-Dateien gezeigt. Die vollständigen Dateien stehen auf der Begleitwebsite zu diesem Buch unter *http://www.dpunkt.de/jquery* zur Verfügung.

Die entsprechende Seite ist in folgender Abbildung dargestellt:

Through the Looking-Glass

by Lewis Carroll

1. Looking-Glass House

There was a book lying near Alice on the table, and while she sat watching the White King (for she was still a little anxious about him, and had the ink all ready to throw over him, in case he fainted again), she turned over the leaves, to find some part that she could read, "—for it's all in some language I don't know," she said to herself.

It was like this.

YKCOWREBBAJ
sevot yhtils eht dna ,gillirb sawT'
;ebaw eht ni elbmig dna eryg diD
,sevogorob eht erew ysmim llA
.ebargtuo shtar emom eht dnA

She puzzled over this for some time, but at last a bright thought struck her. "Why, it's a Looking-glass book, of course! And if I hold it up to a glass, the words will all go the right way again."

This was the poem that Alice read.

JABBERWOCKY
'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Als Nächstes verwenden wir jQuery, um dem Gedicht ein neues Layoutformat zuzuweisen.

> Dieses Beispiel soll zeigen, wie einfach jQuery eingesetzt werden kann. In der Praxis führen Sie eine solche Formatierung ausschließlich mithilfe von CSS durch.

1.3.3 jQuery-Code hinzufügen

Unseren eigenen Code platzieren wir in der zweiten, zurzeit leeren JavaScript-Datei, die wir mit `<script src="01.js"></script>` in das HTML-Dokument eingefügt haben. In diesem Beispiel brauchen wir nur drei Zeilen Code:

```
$(document).ready(function() {
   $('div.poem-stanza').addClass('highlight');
});
```

Das Gedicht finden

Die grundlegende Operation in jQuery besteht darin, einen Teil des Dokuments auszuwählen. Dies geschieht mit der Funktion `$()`. Gewöhnlich erhält sie als Parameter einen String, der einen beliebigen CSS-Selektorausdruck enthalten

kann. In unserem Beispiel möchten wir alle `<div>`-Elemente in dem Dokument finden, denen die Klasse `poem-stanza` zugewiesen ist, weshalb der Selektor sehr einfach ist. Im weiteren Verlauf dieses Buches werden wir uns jedoch noch weit anspruchsvollere Optionen ansehen. In Kapitel 2, »Elemente auswählen«, werden wir viele Möglichkeiten kennenlernen, um einzelne Teile eines Dokuments ausfindig zu machen.

Wenn die Funktion `$()` aufgerufen wird, gibt sie eine neue *jQuery-Objektinstanz* zurück. Das ist der Grundbaustein, mit dem wir von nun an arbeiten werden. Dieses Objekt kann null oder mehrere DOM-Elemente kapseln und erlaubt uns, mit ihnen auf verschiedene Weise umzugehen. In unserem Fall wollen wir das Erscheinungsbild dieser Abschnitte auf der Seite ändern. Das erreichen wir, indem wir dem Gedichttext andere Klassen zuweisen.

Eine neue Klasse einfügen

Wie die meisten jQuery-Methoden trägt auch `.addClass()` einen Namen, der genau ihren Zweck angibt: Sie wendet eine CSS-Klasse auf den Teil der Seite an, den wir ausgewählt haben. Ihr einziger Parameter ist der Name der hinzuzufügenden Klasse. Mithilfe dieser Methode und ihres Gegenstücks `.removeClass()` können wir jQuery in Aktion beobachten, während wir die verschiedenen Selektorausdrücke ausprobieren. Vorerst fügen wir in unserem Beispiel einfach die Klasse `highlight` hinzu, die in unserem Stylesheet als kursiver Text mit grauem Hintergrund und Rand definiert ist.

Beachten Sie, dass keine Iteration notwendig ist, um die Klasse allen Gedichtstrophen hinzuzufügen. Wie wir bereits gesagt haben, führt jQuery in Methoden wie `.addClass()` eine *implizite Iteration* durch, sodass ein einziger Funktionsaufruf ausreicht, um alle ausgewählten Teile des Dokuments zu ändern.

Den Code ausführen

Die Kombination von `$()` und `.addClass()` reicht aus, um unser Ziel zu erreichen, das Erscheinungsbild des Gedichttextes zu ändern. Wenn wir diese Codezeile jedoch einfach in den Dokument-Header einfügen, wird sie keine Auswirkung zeigen. JavaScript-Code wird nämlich im Allgemeinen ausgeführt, sobald der Browser darauf stößt, und während der Verarbeitung des Headers ist noch gar kein HTML-Code vorhanden, der formatiert werden könnte. Wir müssen die Ausführung des Codes daher verzögern, bis das DOM für uns bereitsteht.

Mit dem Konstrukt `$(document).ready()` erlaubt uns jQuery, das Auslösen von Funktionsaufrufen für den Zeitpunkt einzuplanen, an dem das DOM geladen ist, wobei nicht unbedingt darauf gewartet wird, dass alle Bilder vollständig dargestellt werden. Eine solche zeitliche Planung von Ereignissen ist zwar auch ohne jQuery möglich, doch `$(document).ready()` bildet eine besonders elegante, browserübergreifende Lösung mit diversen Vorteilen:

- Sie nutzt die browsereigene Implementierung des Ereignisses `DOM ready`, wenn sie verfügbar ist, und fügt als Sicherheitsnetz noch den Ereignishandler `window.onload` hinzu.
- Sie ermöglicht mehrere Aufrufe von `$(document).ready()` und führt sie in der Reihenfolge der Aufrufe durch.
- Sie führt an `$(document).ready()` übergebene Funktionen selbst dann aus, wenn sie hinzugefügt werden, nachdem das Browserereignis bereits aufgetreten ist.
- Sie führt die Ereignisplanung asynchron durch, damit Skripte bei Bedarf für eine Verzögerung sorgen können.
- Sie simuliert in einigen Browsern ein `DOM-ready`-Ereignis, indem sie wiederholt nach dem Vorhandensein einer DOM-Methode sucht, die normalerweise zur selben Zeit verfügbar wird wie das DOM.

Die Parameter der Methode `.ready()` können einen Verweis auf eine bereits definierte Funktion übernehmen, wie der folgende Codeausschnitt zeigt:

```
function addHighlightClass() {
   $('div.poem-stanza').addClass('highlight');
}

$(document).ready(addHighlightClass);
```

Listing 1–1

Wie die ursprüngliche Version des Skripts und das folgende Listing 1–2 zeigen, akzeptiert die Methode auch eine *anonyme Funktion* (manchmal auch *Lambda-Funktion* genannt):

```
$(document).ready(function() {
   $('div.poem-stanza').addClass('highlight');
});
```

Listing 1–2

Diese Ausdrucksweise mit anonymer Funktion ist in jQuery sehr praktisch für Methoden, die eine nicht wiederverwendbare Funktion als Argument übernehmen. Außerdem kann der dadurch erzeugte *Funktionseinschluss* (auch *Hülle* oder *Closure* genannt) ein anspruchsvolles und leistungsfähiges Werkzeug sein. Allerdings können sich daraus auch unerwünschte Folgen für die Speichernutzung ergeben, wenn man nicht vorsichtig ist. Funktionseinschlüsse werden ausführlich in Anhang A, »***JavaScript-Closures***«, behandelt.

1.3.4 Das fertige Produkt

Mit unserem JavaScript-Code sieht unsere Seite jetzt wie im folgenden Screenshot aus:

Through the Looking-Glass

by Lewis Carroll

1. Looking-Glass House

There was a book lying near Alice on the table, and while she sat watching the White King (for she was still a little anxious about him, and had the ink all ready to throw over him, in case he fainted again), she turned over the leaves, to find some part that she could read, "—for it's all in some language I don't know," she said to herself.

It was like this.

YKCOWREBBAJ

sevot yhtils eht dna ,gillirb sawT'
;ebaw eht ni elbmig dna eryg diD
,sevogorob eht erew ysmim llA
.ebargtuo shtar emom eht dnA

She puzzled over this for some time, but at last a bright thought struck her. "Why, it's a Looking-glass book, of course! And if I hold it up to a glass, the words will all go the right way again."

This was the poem that Alice read.

JABBERWOCKY

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Da der JavaScript-Code die Klasse `highlight` eingefügt hat, erscheinen die Gedichtstrophen jetzt wie im Stylesheet `01.css` definiert, also kursiv und in Kästen mit grauem Hintergrund.

1.4 Einfaches JavaScript und jQuery im Vergleich

Ohne jQuery kann selbst eine so einfache Aufgabe wie diese kompliziert sein. In einfachem JavaScript hätten wir die Klasse `highlight` wie im folgenden Codeausschnitt hinzufügen können:

```
window.onload = function() {
   var divs = document.getElementsByTagName('div');
   for (var i = 0; i < divs.length; i++) {
      if (hasClass(divs[i], 'poem-stanza')
         && !hasClass(divs[i], 'highlight')) {
         divs[i].className += ' highlight';
      }
   }

   function hasClass( elem, cls ) {
      var reClass = new RegExp(' ' + cls + ' ');
      return reClass.test(' ' + elem.className + ' ');
   }
};
```

Listing 1–3

Trotz ihrer Länge kann diese Lösung viele der Aufgaben, um die sich jQuery in Listing 1–2 für uns kümmert, nicht erledigen, wie z.B. die folgenden:

- Korrekte Berücksichtigung anderer `window.onload`-Ereignishandler
- Sofortiges Handeln nach der Bereitstellung des DOM
- Optimieren des Elementabrufs und anderer Aufgaben mit modernen DOM-Methoden

Wie Sie sehen, lässt sich jQuery-Code einfacher schreiben, einfacher lesen und schneller ausführen als das Gegenstück in einfachem JavaScript.

1.5 Entwicklungswerkzeuge

Wie dieser Codevergleich zeigt, ist jQuery-Code gewöhnlich kürzer und klarer als das Gegenstück in einfachem JavaScript. Das heißt jedoch nicht, dass wir stets fehlerfreien Code schreiben oder jederzeit intuitiv verstehen können, was auf unseren Seiten geschieht. Das Schreiben von jQuery-Code wird uns viel leichter von der Hand gehen, wenn wir gängige Entwicklungswerkzeuge zu Hilfe nehmen.

In allen modernen Browsern stehen hochwertige Entwicklungswerkzeuge zur Verfügung, weshalb wir einfach die Umgebung wählen können, in der wir uns am meisten zu Hause fühlen. Unter anderem bieten sich uns folgende Möglichkeiten:

- **Internet Explorer-Entwicklertools:**
  http://msdn.microsoft.com/de-de/library/dd565628.aspx
- **Safari Web Inspector:**
  http://developer.apple.com/technologies/safari/developer-tools.html
- **Chrome Developer Tools:**
  http://code.google.com/intl/de-DE/chrome/devtools/
- **Firebug für Firefox:**
  http://getfirebug.com/

Jedes dieser Instrumente bietet ähnliche Entwicklungsfunktionen:

- Die Möglichkeit, Aspekte des DOM zu untersuchen und zu ändern
- Die Möglichkeit, die Beziehungen zwischen dem CSS-Code und seinen Auswirkungen auf die Seitendarstellung zu untersuchen
- Bequeme Verfolgung der Skriptausführung durch spezielle Methoden
- Anhalten der Ausführung laufender Skripte und Untersuchen von Variablenwerten

In den Einzelheiten unterscheiden sich diese Funktionen zwar von Browser zu Browser, aber das Grundprinzip bleibt das gleiche. In diesem Buch erfordern

einige Beispiele die Verwendung eines dieser Werkzeuge. Zur Veranschaulichung setzen wir hier Firebug ein, aber die Entwicklungswerkzeuge für die anderen Browser sind ebenfalls gute Alternativen.

1.5.1 Firebug

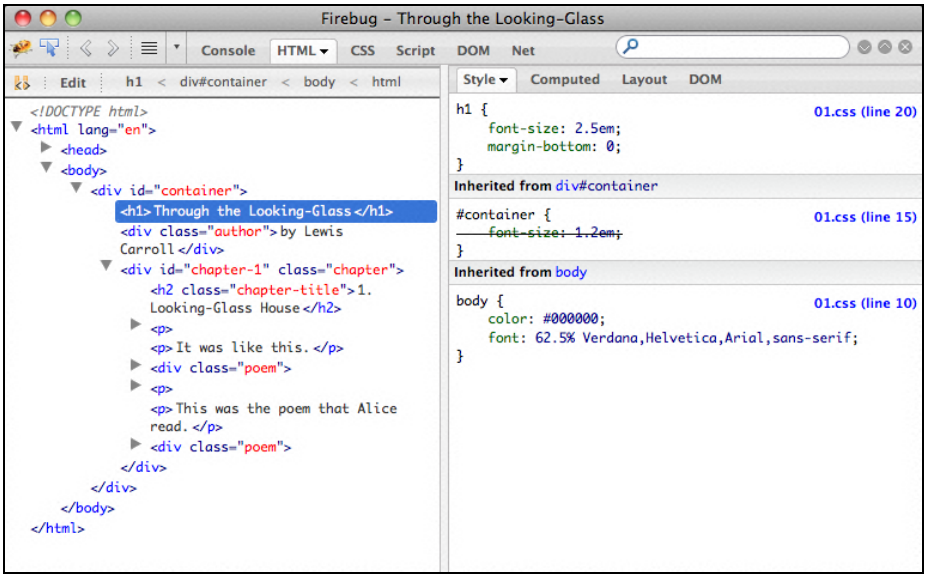
Aktuelle Anleitungen zur Installation und Verwendung von Firebug finden Sie auf der Projekthomepage unter *http://getfirebug.com/*. Dieses Werkzeug ist zu vielschichtig, um es hier ausführlich zu beschreiben. Stattdessen geben wir nur einen Überblick über einige der wichtigsten Funktionen.

Anmerkung zu den Screenshots

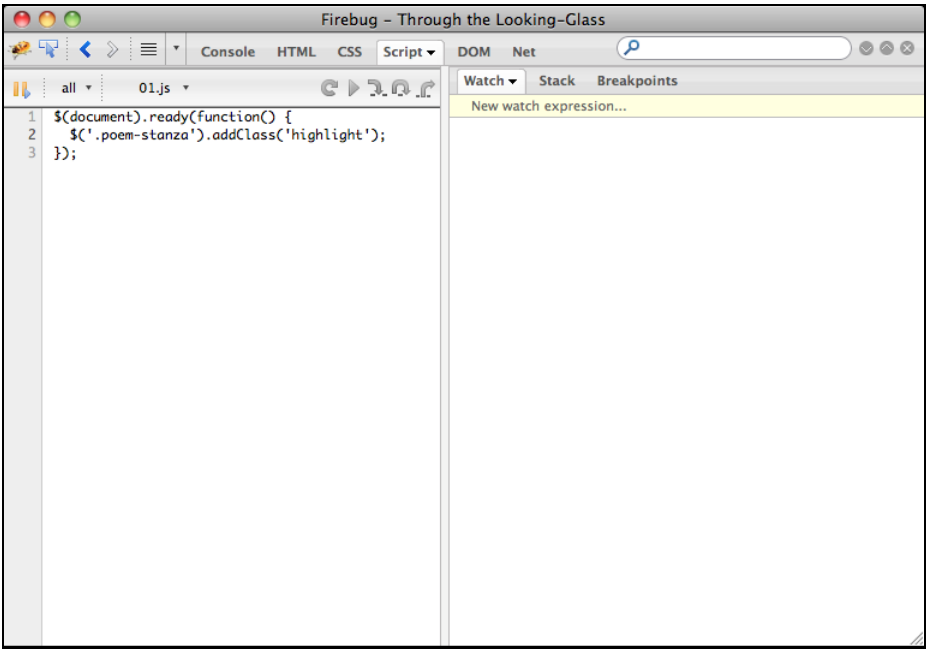
Firebug wird rasch weiterentwickelt, weshalb die folgenden Screenshots von dem abweichen können, was Sie in Ihrer Umgebung zu sehen bekommen. Einige der Beschriftungen und Schaltflächen kommen durch das optionale Add-on *FireQuery* zustande, das Sie unter *http://firequery.binaryage.com/* erhalten.

Wenn Firebug aktiviert ist, wird ein neues Bedienfeld mit Informationen über die aktuelle Seite angezeigt.

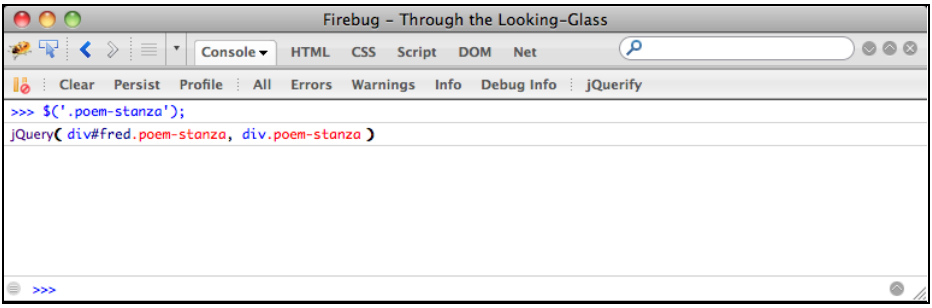
Auf der Standardregisterkarte HTML dieses Bedienfelds ist auf der linken Seite eine Darstellung der Seitenstruktur zu sehen und auf der rechten Seite Angaben über das jeweils markierte Element (z.B. die CSS-Regeln, die darauf angewandt werden). Diese Registerkarte eignet sich vor allem zur Untersuchung der Seitenstruktur und zur Behebung von CSS-Fehlern, wie Sie im folgenden Screenshot erkennen können:



Auf der Registerkarte SCRIPT können wir uns den Inhalt aller geladenen Skripte auf der Seite ansehen, wie der folgende Screenshot zeigt. Durch einen Klick auf eine Zeilennummer setzen wir einen Haltepunkt (breakpoint). Wenn das Skript eine Zeile mit einem solchen Haltepunkt erreicht, wird seine Ausführung angehalten, bis wir sie mit einem Klick auf eine Schaltfläche wieder aufnehmen. Auf der rechten Seite können wir Variablen und Ausdrücke eingeben, deren Wert wir jederzeit ablesen möchten.



Beim Erlernen von jQuery werden wir am häufigsten die Registerkarte CONSOLE verwenden, die Sie im folgenden Screenshot sehen. In einem Feld am unteren Rand können wir eine JavaScript-Anweisung eingeben, deren Ergebnis dann in dem Bedienfeld angezeigt wird:



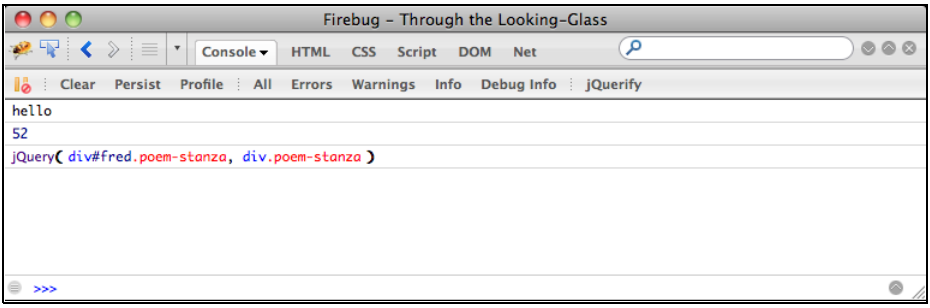
In diesem Beispiel haben wir denselben jQuery-Selektor verwendet wie in Listing 1–2, aber keine Aktion an den ausgewählten Elementen durchgeführt. Doch selbst in dieser Form gibt uns die Anweisung interessante Informationen. Wir können erkennen, dass das Ergebnis des Selektors ein jQuery-Objekt ist, das auf zwei `.poem-stanza`-Elemente auf der Seite verweist. Mit dieser Konsolenfunktion können wir jederzeit jQuery-Code schnell ausprobieren, und zwar direkt im Browser.

Außerdem können wir auch direkt in unserem Code mit der Konsole in Verbindung treten, indem wir wie folgt die Methode `console.log()` verwenden:

```
$(document).ready(function() {
   console.log('hello');
   console.log(52);
   console.log($('div.poem-stanza'));
});
```

Listing 1–4

Dieser Code zeigt, dass wir der Methode `console.log()` jede Art von Ausdruck übergeben können. Einfache Werte wie Strings und Zahlen werden unmittelbar ausgegeben, kompliziertere Werte wie jQuery-Objekte dagegen werden, wie der folgende Screenshot zeigt, zur Inspektion übersichtlich dargestellt:



Die Funktion `console.log()` (die in jedem der erwähnten Browser-Entwicklungswerkzeuge funktioniert) ist eine zweckmäßige Alternative zur JavaScript-Funktion `alert()`. Sie wird sich vor allem beim Testen unseres jQuery-Codes als sehr praktisch erweisen.