Der Inhalt (in der Übersicht)

	Einführung	xxiii
1	Ein Sprung ins kalte JavaScript-Wasser: *Zeit für nasse Füße*	1
2	Richtigen Code schreiben: *Die nächsten Schritte*	43
3	Einführung in Funktionen: *Jetzt funkt's*	79
4	Etwas Ordnung in die Daten bringen: *Arrays*	125
5	Objekte verstehen: *Ein Ausflug nach Objectville*	173
6	Interaktion mit der Webseite: *Das DOM kennenlernen*	229
7	Typen, Gleichheit, Umwandlung und der ganze Rest: *Seriöse Typen*	265
8	Die Einzelteile zusammenfügen: *Eine App schreiben*	317
9	Asynchron programmieren: *Events be-handeln*	381
10	Funktionen erster Klasse: *Befreite Funktionen*	429
11	Anonyme Funktionen, Geltungsbereiche und Closures: *Seriöse Funktionen*	475
12	Fortgeschrittene Objektkonstruktion: *Objekte erstellen*	521
13	Prototypen verwenden: *Extrastarke Objekte*	563
A	Die Top Ten der Themen, die wir nicht behandelt haben: *Was übrig bleibt*	623
	Index	641

Der Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und JavaScript. *Sie* versuchen, etwas zu lernen, und Ihr *Hirn* tut sein Bestes, damit das Gelernte nicht *hängen bleibt*. Es denkt nämlich: »Wir sollten lieber Platz für wichtigere Dinge lassen, z. B. für das Wissen darüber, welche Tiere einem gefährlich werden könnten oder dass es eine ganz schlechte Idee ist, nackt Snowboard zu fahren.« Tja, wie schaffen wir es nun, Ihr Gehirn davon zu überzeugen, dass Ihr Leben davon abhängt, etwas über JavaScript-Programmierung zu wissen?

Für wen ist dieses Buch?	xxiv
Wir wissen was Sie gerade denken.	xxv
Wir stellen uns unseren Leser als einen aktiv Lernenden vor	xxvi
Metakognition: Nachdenken übers Denken	xxvii
Das haben WIR getan	xxviii
Das können SIE tun, um sich Ihr Gehirn untertan zu machen	xxix
Lies mich!	xxx
Fachgutachter	xxxiii
Danksagungen	xxxiv

Ein Sprung ins kalte JavaScript-Wasser

1

Zeit für nasse Füße

JavaScript verleiht Superkräfte. Es ist **die Programmiersprache** des Webs. Mit JavaScript bekommen Ihre Webseiten nämlich **Verhalten**. Das heißt: keine trockenen, langweiligen statischen Seiten mehr, die einfach nur dasitzen und Sie anstarren. Mit JavaScript erreichen Sie Ihre Benutzer direkt. Sie können auf interessante Events reagieren, Daten aus dem Web einbinden, Diagramme direkt in Ihren Seiten erstellen und vieles mehr. Haben Sie JavaScript einmal verstanden, können Sie vollkommen neue Verhaltensweisen für Ihre Benutzer erstellen.

Wie JavaScript funktioniert	2
Wie Sie JavaScript schreiben werden	3
JavaScript in die Seite einbinden	4
JavaScript, du hast einen langen Weg hinter dir ...	6
Anweisungen definieren	10
Variablen und Werte	11
Finger weg von der Tastatur!	12
Der richtige Ausdruck	15
Dinge mehr als einmal tun	17
Wie eine while-Schleife funktioniert	18
Entscheidungen in JavaScript	22
Und wenn Sie richtig VIELE Entscheidungen treffen müssen	23
Sprechen Sie Ihre Benutzer direkt an	25
Ein näherer Blick auf console.log	27
Die Konsole öffnen	28
Die erste richtige JavaScript-Applikation	29
Wie bekomme ich den Code in meine Seite? (... mal die Möglichkeiten zählen ...)	32
Wir müssen euch trennen	33

2 Die nächsten Schritte

Mittlerweile kennen Sie schon Variablen, Typen, Ausdrücke und noch mehr. Sie wissen bereits ein paar Dinge über JavaScript. Das heißt, Sie können schon **richtigen Code** schreiben. Code, der etwas Interessantes tut – Code, den andere Leute benutzen wollen. Jetzt fehlt Ihnen nur noch die **richtige Erfahrung** beim Schreiben von Code. Und das wollen wir hier und jetzt ändern. Wie? Indem wir kopfüber ins kalte Wasser springen und ein komplettes JavaScript-Spiel programmieren. Klingt ehrgeizig, aber wir lassen Sie nicht hängen. Auf geht's, lassen Sie uns direkt loslegen. Und wenn Sie damit eine neue Firma gründen wollen – nur zu! Der Code gehört Ihnen.

Schiffe versenken: Die Programmierung 44
Unser erster Versuch... 44
Der allgemeine Spielverlauf 45
Arbeiten wir uns durch den Pseudocode 47
Bevor Sie weiter machen sollten Sie den HTML-Code nicht vergessen! 49
Der Code für die einfache Version von »Schiffe versenken« 50
Und damit zur Logik des Spiels 51
Schritt eins: Die Schleife einrichten und Benutzereingaben auslesen 52
Wie prompt funktioniert 53
Die Benutzereingabe überprüfen 54
Und? Haben wir getroffen? 56
Der Code zum Finden von Treffern 57
Etwas Spielanalyse für den Benutzer 58
Und damit ist die Logik vollständig 60
Ein wenig Qualitätssicherung 61
Lassen Sie uns über Wortreichtum reden... 65
Das Schiffe-versenken-Spiel fertigstellen 66
Eine zufällige Position zuweisen 67
Das weltberühmte Rezept zur Erzeugung von Zufallszahlen 67
Etwas mehr QA 69
Herzlichen Glückwunsch zu Ihrem ersten richtigen JavaScript-Programm. Aus diesem Anlass ein paar Worte zum Code-Recycling. 71

3

Jetzt funkt's

Hier kommt Ihre erste Superkraft. Inzwischen können Sie schon ein wenig programmieren. Daher wollen wir Ihnen jetzt das Konzept der **Funktionen** vorstellen. Funktionen geben Ihnen die Macht, Code zu schreiben, der unter verschiedensten Umständen **wiederverwendet** werden kann. Dieser Code ist deutlich besser **wartbar**, er kann **abstrahiert** und mit einem einfachen Namen versehen werden. Dadurch können Sie die Komplexität ignorieren und sich auf die wichtigen Dinge konzentrieren. Funktionen sind ein wichtiger Schritt auf Ihrem Weg vom Skripter zum Programmierer und der Schlüssel zum Programmierstil von JavaScript. In diesem Kapitel zeigen wir Ihnen die Grundlagen: die Mechanik, die ganzen kleinen Details, die zum Verständnis von Funktionen wichtig sind. Danach werden wir Ihre Kenntnisse im Laufe dieses Buchs immer mehr erweitern und vertiefen. *Jetzt* wollen wir aber erst einmal solides Fundament schaffen.

Was stimmt denn mit dem Code nicht?	81
Haben wir übrigens schon über FUNKTIONEN gesprochen?	83
Schön, aber wie funktioniert das überhaupt?	84
Was kann einer Funktion übergeben werden?	89
JavaScript verwendet Werteparameter	92
Seltsame Funktionen	94
Funktionen können auch Dinge zurück geben	95
Eine Funktion mit return-Anweisung nachvollziehen	96
Globale und lokale Variablen	99
Den Geltungsbereich globaler und lokaler Variablen verstehen	101
Das kurze Leben der Variablen	102
Vergessen Sie nicht, alle lokalen Variablen zu deklarieren!	103

Etwas Ordnung in die Daten bringen

4 Arrays

JavaScript besteht nicht nur aus Zahlen, Strings und Boolesche Werten. Bisher haben Sie in Ihrem JavaScript-Code nur primitive Datentypen – einfache Strings, Zahlen und Boolesche Werte wie »Fido«, 23 und true – benutzt. Damit lässt sich schon eine Menge anstellen, aber irgendwann müssen Sie mit **mehr Daten** arbeiten. Das könnten alle Artikel in einem Warenkorb sein oder alle Titel einer Playlist oder eine Liste von Sternen und ihre scheinbare Größe oder ein kompletter Produktkatalog. Dafür brauchen wir etwas mehr *Wumms*. Der Datentyp der Wahl für geordnete Daten dieser Art ist das JavaScript-**Array**. In diesem Kapitel zeigen wir Ihnen, wie Sie Ihre Daten in einem Array ablegen, es weitergeben und damit arbeiten können. Später in diesem Buch werden Sie weitere Möglichkeiten kennenlernen, Ihre **Daten zu strukturieren**; jetzt wollen wir uns aber erst mal mit Arrays befassen.

Können Sie Bubbles-R-Us helfen?	126
Mehrere Werte in JavaScript darstellen	127
Wie Arrays funktionieren	128
Wie groß ist das Array eigentlich?	130
Der Phras-O-Mat	132
Inzwischen bei Bubbles-R-Us...	135
Über ein Array iterieren	138
Moment mal! Es gibt noch eine bessere Methode über ein Array zu iterieren	140
Wir müssen mal wieder über Ihren Wortschatz reden	146
Eine Neufassung der for-Schleife mit dem Postinkrement-Operator	147
Schnelle Probefahrt	147
Ein neues Array anlegen (und Elemente hinzufügen)	151
Und die Gewinner sind ...	155
Eine schnelle Codeüberprüfung ...	157
Eine printAndGetHighScore-Funktion schreiben	158
Den Code mit der printAndGetHighScore-Funktion refaktorieren	159
Alles zusammengenommen ...	161

5 Ein Ausflug nach Objectville

Bis jetzt haben Sie in Ihrem Code nur »primitive« Datentypen und Arrays benutzt. Die Programmierung sind Sie bisher eher **prozedural** angegangen um mit einfachen Anweisungen, Bedingungen, for-/while-Schleifen und Funktionen, die nicht gerade objektorientiert waren. Eigentlich waren sie *überhaupt nicht* objektorientiert. Wir haben zwar hier und da, ohne es zu wissen, ein paar Objekte benutzt, aber eigene Objekte haben Sie noch nicht geschrieben. Jetzt ist es Zeit, diese langweilige prozedurale Stadt zu verlassen und selbst ein paar **Objekte** zu erstellen. In diesem Kapitel werden Sie lernen, wie Sie sich das Leben deutlich erleichtern können, zumindest was die **Programmierung** angeht. (Im selben Buch auch noch Ihrem modischen Geschmack auf die Sprünge zu helfen, ist dann doch ein bisschen viel verlangt.) Eine kleine Warnung: Haben Sie die Objekte einmal kennengelernt, wollen Sie nicht wieder zurück. Schicken Sie uns eine Postkarte, wenn Sie angekommen sind.

Hat gerade jemand »Objekte« gesagt?! 174
Ein paar Gedanken zu Eigenschaften... 175
Objekte anlegen 177
Was bedeutet eigentlich »objektorientiert«? 180
Wie Eigenschaften funktionieren 181
Wie kann eine Variable ein Objekt enthalten? 186
Primitive Datentypen und Objekte im Vergleich 187
Noch mehr Dinge mit Objekten... 188
Die Vorauswahl Schritt für Schritt 190
Noch ein paar Worte zur Übergabe von Objekten an Funktionen 192
Objekte mit Verhalten versehen 198
Die drive-Methode verbessern 199
Warum weiß die drive-Methode nicht von der Eigenschaft started? 202
Wie `this` funktioniert 204
Wie Verhalten einen Zustand beeinflussen kann 210
Und jetzt soll der Zustand das Verhalten beeinflussen 211
Herzlichen Glückwunsch zu Ihren ersten Objekten! 213
Objekte sind überall (und sie machen Ihnen das Leben leichter)! 214

6

Das DOM kennenlernen

Ihre JavaScript-Fähigkeiten haben große Fortschritte gemacht. Mittlerweile haben Sie sich von einem Anfänger zu einem Skripter, ja sogar zu einem echten **Programmierer** entwickelt. Es fehlt aber noch etwas. Damit sich Ihre JavaScript-Kenntnisse wirklich entfalten können, brauchen Sie eine Möglichkeit, mit der Webseite, die Ihren Code enthält, zu interagieren. Nur so ist es möglich, **dynamische** Seiten zu schreiben, die reagieren, antworten und sich nach dem Laden automatisch aktualisieren. Wie funktioniert diese Interaktion? Über das **DOM**, das **Document Object Model**. Wir werden uns das DOM in diesem Kapitel Stück für Stück vornehmen und sehen, wie es zusammen mit JavaScript dazu bewegt werden kann, Ihrer Seite ein paar neue Tricks beizubringen.

Im letzten Kapitel gab es eine »Knacken Sie den Geheimcode«-Aufgabe	230
Und was macht dieser Code genau?	231
Wie JavaScript tatsächlich mit Ihrer Seite interagiert	233
Ein Rezept für Ihr eigenes DOM	234
Ihr erster Eindruck vom DOM	235
Mit getElementById auf ein Element zugreifen	240
Was genau gibt mir das DOM zurück?	241
Finden Sie Ihr inneres HTML	242
Was passiert wenn Sie das DOM verändern?	244
Ein kleiner Probeflug	247
Denken Sie nicht mal daran, meinen Code vor dem vollständigen Laden der Seite auszuführen!	249
Sie sagen »Event-Handler«, ich sage »Callback«	250
Ein Attribut per setAttribute bearbeiten	255
Mehr Spaß mit Attributen	256
Vergessen Sie nicht, dass getElementById auch null zurückgeben kann!	256
Und wofür ist das DOM sonst noch gut?	258

7 Seriöse Typen

Jetzt wollen wir einmal ernsthaft über Typen reden. Eine der guten Seiten von JavaScript ist, dass Sie auch ohne viel Detailwissen schon sehr weit kommen können. Aber um **die Sprache wirklich zu meistern**, die Gehaltserhöhung zu bekommen und es im Leben richtig zu etwas bringen, müssen Sie sich extrem gut mit **Typen** auskennen. Wissen Sie noch, was wir am Anfang über JavaScript gesagt haben? Dass es keine auf dem Silbertablett servierte, durch akademische Kreuzgutachten überprüfte Sprache ist? Das stimmt wohl, aber das akademische Leben hat Steve Jobs und Bill Gates nicht von ihrem Erfolg abgehalten, und JavaScript hält es auch nicht ab. Das bedeutet, dass JavaScript ... nun ja ... kein besonders durchdachtes Typensystem besitzt und wir noch ein paar **Eigenheiten** bemerken werden. Aber machen Sie sich keine Sorgen! In diesem Kapitel werden wir die Sache in den Griff bekommen, und Sie werden schnell in der Lage sein, den peinlichen Momenten, die Ihnen mit Typen begegnen können, aus dem Weg zu gehen.

Die Wahrheit ist irgendwo da draußen ...	266
Achtung, Sie könnten undefined begegnen wenn Sie es nicht erwarten ...	268
Die Verwendung von null	271
Mit NaN arbeiten	273
Es wird noch merkwürdiger	273
Wir müssen Ihnen etwas gestehen	275
Den Gleichheitsoperator (auch bekannt als ==) verstehen	276
Wie der Gleichheitsoperator seinen Operanden konvertiert	277
Die Gleichheit strikter verstehen	280
Noch mehr Typumwandlungen ...	286
Testen, ob zwei Objekte gleich sind	289
Die truthy ist irgendwo da darußen	291
Was JavaScript als falsey ansieht	292
Das geheime Leben der Strings	294
Strings, die gleichzeitig primitive Datentypen und Objekte sind	295
Eine Fünf-Minuten-Tour der String-Methoden (und -Eigenschaften)	297
Krieg der Stühle	301

8 Eine App schreiben

Greifen Sie sich Ihren Werkzeugkasten. Damit meinen wir den Werkzeugkasten mit all Ihren neuen Programmierkenntnissen, dem Wissen über das DOM und sogar mit etwas HTML und CSS. In diesem Kapitel bringen Sie alles zusammen und erstellen Ihre erste richtige **Webapplikation** – keine **albernen Kinderspiele** mit nur einem Schiff, das auf einer Zeile versteckt ist. Jetzt bauen wir **das komplette Spiel**: ein schönes Spielfeld mit mehreren Schiffen und Benutzereingaben direkt auf der Webseite. Die nötige Seitenstruktur erstellen wir mit HTML, visuelle Stile schreiben wir in CSS, und mit JavaScript programmieren wir das Verhalten für das Spiel. Machen Sie sich bereit: Dies ist ein richtiges Entwicklungskapitel, in dem wir ernsthaften Code schreiben.

Diesmal wollen wir ein ECHTES »Schiffe-versenken«-Spiel erstellen	318
Ein Schritt zurück ... zu HTML und CSS	319
Die HTML-Seite erstellen: Ein Überblick	320
Etwas Stil hinzufügen	324
Die Klassen hit und miss verwenden	327
Das Spiel entwickeln	329
Den View implementieren	331
Die Funktionsweise von displayMessage	331
Die Funktionsweise von displayHit und displayMiss	333
Das Model	336
Wie die Schiffe intern dargestellt werden	338
Das Model-Objekt implementieren	341
Nachdenken über die fire-Methode	342
Den Controller implementieren	349
Den Rateversuch den Spielers verarbeiten	350
Den Code planen ...	351
Die parseGuess-Methode implementieren	352
Rateversuche zählen und feuern	355
Ein Event-Handler für den Feuer!-Button	359
Die Eingabe an den Controller übergeben	360
Die Schiffe richtig platzieren	364
Die generateShip-Methode erstellen	365
Die Startpostion für ein neues Schiff erzeugen	366
Die generateShip-Methode fertigstellen	367

9 Events be-handeln

Nach diesem Kapitel werden Sie merken, dass Sie nicht mehr in Kansas sind. Bis jetzt haben Sie Code geschrieben, der typischerweise von oben nach unten ausgeführt wird. Sicher, Ihr Code ist vermutlich etwas komplexer und enthält ein paar Funktionen, Objekte und Methoden, aber irgendwann wird auch dieser Code ausgeführt. Es tut uns wirklich leid, wenn wir Ihnen das erst so spät im Buch sagen, aber **so schreibt man eigentlich keinen JavaScript-Code**. Stattdessen wird JavaScript meistens geschrieben, um auf **Ereignisse (Events) zu reagieren**. Was für Events? Alle möglichen! Das kann ein Mausklick auf Ihrer Seite, ankommende Daten aus dem Netzwerk, ein im Browser ablaufender Timer oder eine Änderung im DOM sein, um nur ein paar zu nennen. Eigentlich passieren **ständig** irgendwelche Events im Hintergrund des Browsers. In diesem Kapitel werden wir unsere Art, in JavaScript zu programmieren, neu überdenken und lernen, warum und wie Sie Code schreiben können, der auf Events reagiert.

Was sind Events?	383
Was ist ein Event-Handler?	384
Den ersten Event-Handler erstellen	385
Probefahrt für Ihr Event	386
Events verstehen... indem wir ein Spiel programmieren	388
Das Spiel implementieren	389
Probefahrt	390
Noch mehr Bilder	394
Jetzt müssen wir den onclick-Eigenschaften jedes Bilds den gleichen Handler zuweisen	395
Den gleichen Handler für alle Bilder benutzen	396
Die Funktionsweise des Event-Objekts	399
Das Event-Objekt benutzen	401
Probefahrt für das Event-Objekt und die target-Eigenschaft	402
Events und ihre Warteschlange	404
Noch mehr Events	407
Wie setTimeout funktioniert	408
Das Bilderrätsel fertigstellen	412
Probefahrt für den Timer	413

10 Befreite Funktionen

Erst Funktionen machen Sie zum Rockstar. Jedes Handwerk, jede Kunst besitzt ein Schlüsselprinzip, das die mittelmäßigen Spieler von den großen Virtuosen unterscheidet. Bei JavaScript ist es das Verständnis von **Funktionen**. Funktionen sind ein wesentlicher Bestandteil von JavaScript und bieten viele Techniken für das **Design und die Organisation** von Code. Die Basis hierfür ist ein solides Wissen um die Verwendung von Funktionen. Funktionen auf diesem Niveau zu lernen, ist interessant und kann bewusstseinsverändernd wirken, also machen Sie sich bereit. Es ist ein bisschen, als würde Ihnen Willy Wonka eine Führung durch die Schokoladenfabrik geben. Beim Lernen von JavaScript-Funktionen werden Ihnen einige ziemlich wilde, verrückte, aber auch wundervolle Dinge begegnen.

Das mysteriöse Doppelleben des Schlüsselworts function 430
Funktionsdeklarationen im Vergleich mit Funktionsausdrücken 431
Die Funktionsdeklaration parsen 432
Was kommt jetzt? Der Browser führt den Code aus 433
Weiter geht's ... Die Bedingung 434
Wie Funktionen auch Werte sein können 439
Haben wir schon gesagt, dass Funktionen in JavaScript als erstklassig gelten? 442
Erster Klasse fliegen 443
Den Code schreiben, um Passagiere zu verarbeiten und zu überprüfen 444
Über die Passagiere iterieren 446
Eine Funktion an eine andere übergeben 447
Funktionen aus Funktionen zurückgeben 450
Den Code für die Getränkebestellung schreiben 451
Der Flugbegleiter-Code für die Getränkebestellung: ein neuer Ansatz 452
Bestellungen mit Funktionen erster Klasse 454
Webville Cola 457
Wie die sort-Methode für Array funktioniert 459
Die Einzelteile zusammensetzten 460
Probefahrt für die Sortierung 462

11 Seriöse Funktionen

Sie haben sich umfassend mit Funktionen befasst, aber es gibt noch mehr zu lernen. In diesem Kapitel gehen wir sogar noch weiter, jetzt kommt echter Hardcore. Wir zeigen Ihnen, wie man **wirklich** mit Funktionen **umgeht**. Dies wird kein superlanges Kapitel, aber es wird intensiv, und am Ende wird Ihr JavaScript ausdrucksstärker sein als je zuvor. Sie werden auch in der Lage sein, den Code Ihrer Mitarbeiter zu verstehen und eine Open Source-JavaScript-Bibliothek zu durchschauen. In diesem Kapitel behandeln wir nämlich einige häufige Code-Idiome und Konventionen, die bei Funktionen immer wieder vorkommen. Und wenn Sie noch nie von **anonymen Funktionen** oder **Closures** gehört haben, dann sind Sie hier mehr als richtig.

Sehen Sie Funktionen mal andersherum	476
Anonyme Funktionen benutzen	477
Wir müssen nochmal über Ihren Wortschatz reden	479
Wann wird eine Funktion definiert? Kommt drauf an ...	483
Was ist da gerade passiert? Warum war fly nicht definiert?	484
Funktionen verschachteln	485
Wie sich die Verschachtelung auf den Geltungsbereich auswirkt	486
Ein kleiner Rückblick zum lexikalischen Geltungsbereich	488
Wo der lexikalische Geltungsbereich richtig interessant wird	489
Wiedersehen mit Funktionen	491
Wiedersehen mit Funktionsaufrufen	492
Was zum Henker ist eine Closure?	495
Eine Funktion »schließen«	496
Mit Closures einen magischen Zähler implementieren	498
Ein Blick hinter die Kulissen	499
Closures durch Übergabe einer Funktionsreferenz als Argument erstellen	501
Die Closure enthält keine Kopie, sondern die tatsächliche Umgebung	502
Eine Closure per Event-Handler erstellen	503
Wie die Klick mich!-Closure funktioniert	506

12 Objekte erstellen

Bisher haben wir unsere Objekte von Hand erstellt. Für jedes Objekt haben wir ein **Objektliteral** verwendet, um die einzelnen Eigenschaften anzugeben. Das ist in kleinem Rahmen auch kein Problem. Für ernsthaften Code brauchen wir aber etwas Besseres. Und da kommen **Objektkonstruktoren** ins Spiel. Mit Konstruktoren können wir die Objekte wesentlich einfacher erstellen, die außerdem alle nach der gleichen **Designschablone** aufgebaut sind. Das heißt, durch die Verwendung von Konstruktoren können wir sicherstellen, dass jedes Objekt die gleichen Eigenschaften besitzt und die gleichen Methoden enthält. Außerdem ist der mit Konstruktoren erstellte Objektcode wesentlich **kürzer** und deutlich weniger fehleranfällig, besonders wenn viele Objekte erstellt werden müssen. Legen wir also einfach mal los! In kürzester Zeit werden Sie so gut »konstruktorisch« sprechen, als seien Sie in Objectville groß geworden.

Objekte mit Objektliteralen erstellen	522
Konventionen für Objekte verwenden	523
Einführung in Objektkonstruktoren	525
Einen Konstruktor erstellen	526
Einen Konstruktor benutzen	527
Wie Konstruktoren funktionieren	528
Konstruktoren können auch Methoden enthalten	530
Zeit für die Produktion!	536
Jetzt wollen wir ein paar neue Autos Probe fahren	538
Verlassen Sie sich nicht einfach auf Objektliterale	539
Die Argumente als Objektliteral neu verdrahten	540
Den Car-Konstruktor umbauen	541
Objektinstanzen verstehen	543
Auch per Konstruktor erstellte Objekte können eigene unabhängige Eigenschaften haben	546
Konstruktoren im währen Leben	548
Der Array-Objektkonstruktor	549
Noch mehr Spaß mit eingebauten Objekten	551

13

Extrastarke Objekte

Zu lernen, wie man Objekte erstellt, war erst der Anfang. Jetzt ist es Zeit, Ihren Objekten ein paar Muskeln wachsen zu lassen. Wir brauchen neue Möglichkeiten, **Beziehungen** zwischen den Objekten herzustellen und **Code** zwischen ihnen **auszutauschen**. Und wir benötigen die Fähigkeit, bestehende Objekte zu erweitern und aufzuwerten. Anders gesagt: Wir brauchen mehr Werkzeuge. In diesem Kapitel werden Sie sehen, dass JavaScript ein sehr mächtiges **Objektmodell** besitzt, aber auch, dass es sich von den üblichen objektorientierten Programmiersprachen unterscheidet. Anstelle des typischen klassenbasierten Objektmodells verwendet JavaScript ein mächtigeres **Prototypenmodell**, bei dem die Objekte voneinander erben oder das Verhalten anderer Objekte erweitern können. Wofür das gut sein soll, werden Sie gleich sehen. Auf geht's ...

Ach so, bevor wir anfangen, haben wir noch eine bessere Möglichkeit unsere Objekte in einem Diagramm darzustellen 565
Wiedersehen mit Objektkonstruktoren 566
Ist die Duplizierung der Methode wirklich ein Problem? 568
Was sind Prototypen? 569
Von einem Prototyp erben 570
Wie die Vererbung funktioniert 571
Prototypen überschreiben 573
Den Prototyp einrichten 576
Prototypen sind dynamisch 582
Eine interessantere Implemetierung der sit-Methode 584
Noch einmal: die Funktionsweise der sitting-Eigenschaft 585
Die Herangehensweise für Ausstelungshunde 589
Eine Prototypenkette einrichten 591
Die Funktionsweise der Vererbung in der Prototypenkette 592
Den Prototyp für Ausstellungshunde erstellen 594
Eine Ausstellungshunde-Instanz erzeugen 598
Eine letzte Reinigung der Ausstellungshunde 602
Dog.call Schritt für Schritt 604
Die Kette endet nicht bei Dog 607
Vererbung zu Ihrem Vorteil nutzen ... durch Überschreiben von eingebautem Verhalten 608
Vererbung zum eigenen Vorteil nutzen ... durch Erweiterung einen JavaScript-eigenen Objekts 610
Die große Einheitstheorie von Allem 612
Besser leben mit Objekten 612
Die Einzelteile zusammenfügen 613
Was kommt als Nächstes? 613

14 Die Top Ten der Themen, die wir nicht behandelt haben

Wir sind ganz schön weit gekommen, und Sie haben das Buch fast durch. Wir werden Sie vermissen, aber bevor wir Sie gehen lassen, wollen wir Sie nicht ohne etwas zusätzliche Vorbereitung losschicken. Leider können wir nicht alles, was Sie noch wissen müssen, in diesem relativ kurzen Kapitel unterbringen. Ursprünglich hatten wir tatsächlich sämtliches notwendiges Wissen über JavaScript-Programmierung (das nicht bereits in den anderen Kapiteln behandelt wurde) hier untergebracht, indem wir die Schriftgröße ungefähr auf 0,00004 Punkt reduziert haben. Es passte alles rein, aber niemand konnte es lesen. Also haben wir das meiste wieder rausgeschmissen und die besten Sachen für den Top-Ten-Anhang behalten.

Das hier ist tatsächlich das Ende des Buchs. Das heißt, bis auf den Index natürlich (den Sie unbedingt lesen müssen!).

1. jQuery 624
2. Mehr mit dem DOM anstellen 626
3. Das window-Objekt 627
4. arguments 628
5. Mit Ausnahmen umgehen 629
6. Event-Handler mit addEventListener hinzufügen 630
7. Reguläre Ausdrücke 632
8. Rekursion 634
9. JSON 636
10. Serverseitiges JavaScript 637

Index 641