

# Der Inhalt (im Überblick)

	Einführung	xxi
1	Das interaktive Web: <i>Auf die virtuelle Welt reagieren</i>	1
2	Daten speichern: <i>Alles hat seinen Platz</i>	33
3	Den Client erforschen: <i>Browserforschung</i>	85
4	Entscheidungen treffen: <i>Wenn eine Abzweigung kommt, folgen Sie ihr</i>	135
5	Schleifen: <i>Auf die Gefahr, mich zu wiederholen</i>	189
6	Reduzieren, Wiederverwenden, Recyclen: <i>Funktionen</i>	243
7	Dem Benutzer alles entlocken: <i>Formulare und Validierung</i>	289
8	Die Seite zähmen: <i>HTML schneiden und würfeln mit dem DOM</i>	343
9	Objekte als Frankensteindaten: <i>Daten zum Leben erwecken</i>	393
10	Benutzerdefinierte Objekte erstellen: <i>Jedem das Seine mit benutzerdefinierten Objekten</i>	449
11	Bugs zur Strecke bringen: <i>Gute Skripten schief gewickelt</i>	485
12	Dynamische Daten: <i>Empathische Webapplikationen</i>	537

# 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 ordentlich 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 zu wissen?

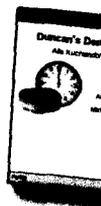
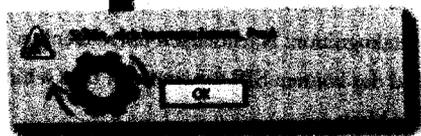
Für wen ist dieses Buch?	xxii
Wir wissen, was Sie gerade denken	xxiii
Und wir wissen, was Ihr Gehirn gerade denkt	xxiii
Metakognition: Nachdenken übers Denken	xxv
Und das können SIE tun, um sich Ihr Gehirn untertan zu machen	xxvii
Lies Mich	xxviii
Die Fachgutachter dieses Buchs	xxx
Danksagung	xxxi

## Auf die virtuelle Welt reagieren

### Die Nase voll von einem Web aus nichts als passiven Seiten?

Alles schon gehabt. Das nennt man Bücher. Und sie sind für eine Menge Dinge gut – Lesen, Lernen ... aber eben nicht **interaktiv**. Und genauso wenig interaktiv ist das Web ohne ein bisschen JavaScript. Klar können Sie ein Formular übermitteln und mit ausgefuchstem HTML- und CSS-Code tricksen, aber unterm Strich spielen Sie nur *Immer Ärger mit Bernie* auf einer leblosen Webseite. Echte, lebendige **Interaktivität** erfordert **ein bisschen mehr Grips** und **etwas mehr Arbeit** ... aber es lohnt sich!

Das (Online-)Volk hat Bedürfnisse	2
Als wenn man mit einer Wand redet ... nichts passiert	3
JavaScript gibt Kontra	4
Licht, Kamera, Inter-Ächtschn!	6
Sagen Sie dem Browser mit dem <script>-Tag, dass Sie JavaScript schreiben	11
Ihr Webbrowser kennt HTML, CSS <b>und</b> JavaScript	12
Des Menschen bester virtueller Freund ... braucht Ihre Hilfe	15
Der iRock wird interaktiv	16
Die iRock-Webseite erstellen	17
Testlauf	17
JavaScript-Events: Der iRock bekommt eine Stimme	18
Eine Funktion zum »Alarmieren« des Benutzers	19
Die iRock-Begrüßung	20
Testen Sie Ihren interaktiven Stein	20
Jetzt machen wir den iRock wirklich interaktiv	22
Interaktive Kommunikation hat ZWEI Richtungen	23
Nach dem Benutzernamen fragen	24
Bildwiederholung: Was ist passiert?	27
Testen Sie iRock 1.0	28

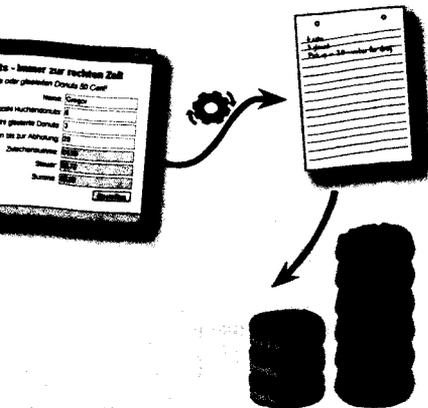


# 2

## Alles hat seinen Platz

In der realen Welt übersehen Menschen oft, wie wichtig es ist, für alles den rechten Platz zu haben. Aber nicht in JavaScript. Da gibt es den Luxus begehbarer Schränke und den Vorzug von Dreiergaragen einfach nicht. In JavaScript **hat alles seinen festen Platz**, und dafür sind Sie jetzt zuständig. Es geht um **Daten** – wie sie **dargestellt** werden, wie sie **gespeichert** werden und wie Sie sie **wiederfinden**, nachdem Sie sie abgelegt haben. Als Speicherspezialist für JavaScript können Sie einer Rumpelkammer von JavaScript-Daten mit einer Armee von Etiketten und Ablagen zu Leibe rücken.

Ihre Skripten können Daten speichern	34
Skripten denken in Datentypen	35
Konstanten bleiben GLEICH, Variablen können sich ÄNDERN	40
Variablen haben anfangs keinen Wert	44
Variablen mit "=" initialisieren	45
Konstanten sind beständig	46
Was macht einen Namen aus?	50
Legale und illegale Variablen- und Konstantennamen	51
CamelCase für Variablennamen	52
Der nächste große Schritt (mit den Donuts)	55
Die Webseite für Duncan's Donuts planen	56
Ein erster Versuch mit den Donut-Kalkulationen	58
Initialisieren Sie Ihre Daten ... sonst ...	61
NaN ist keine Zahl	62
In der Zwischenzeit bei Duncan's ...	63
Sie können mehr als nur Zahlen addieren	64
parseInt() und parseFloat() konvertieren Text in eine Zahl	65
Warum werden zu viele Donuts bestellt?	66
Sie haben das Problem gefunden	69
Donut-Sabotage	70
Mit getElementById() Formulardaten lesen	71
Die Daten des Webformulars validieren	72
Sie haben Duncan's Donuts gerettet ... wieder mal!	75
Intuitiv auf Benutzereingaben reagieren	77
Online-Donuts – ein durchschlagender Erfolg!	80



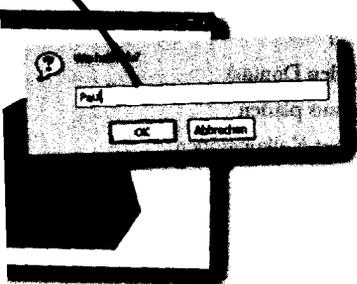
## Den Client erforschen

### Browserforschung

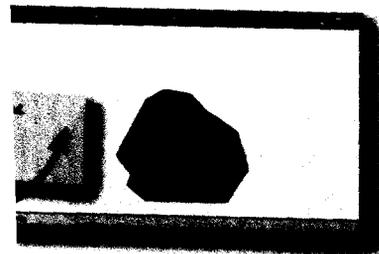
# 3

**Manchmal muss JavaScript wissen, was in der Welt um Sie herum vor sich geht.** Anfangs sind Ihre Skripten nur Code in einer Webseite, aber letztendlich leben sie in einer Welt, die der Browser oder der Client erschaffen hat. **Intelligente Skripten** müssen oft mehr über die Umgebung erfahren, in der sie laufen. Dazu müssen sie **mit dem Browser kommunizieren**. Ob es darum geht, die Bildschirmgröße herauszufinden oder auf die Stoptaste des Browsers zuzugreifen – Skripten können eine Menge davon profitieren, wenn sie ihre Beziehung zum Browser vertiefen.

Los geht's!



Geschafft!



Clients, Server und JavaScript	86
Was kann der Browser für Sie tun?	88
Der iRock muss mehr reagieren	90
Timer knüpfen Aktionen an abgelaufene Zeit	92
Timer Schritt für Schritt	93
Einen Timer mit setTimeout() stellen	94
Genauer hingeguckt: die setTimeout()-Funktion	95
Viele Bildschirmgrößen, viele Beschwerden	99
Mit dem document-Objekt die Breite des Clientfensters ermitteln	100
Mit den Eigenschaften des document-Objekts die Breite des Clientfensters bestimmen	101
Höhe und Breite des iRock festlegen	102
Den iRock an die Seitengröße anpassen	103
onresize wird ausgelöst, wenn sich die Browsergröße ändert	107
Das onresize-Event passt den Stein an	108
Kennen wir uns? Den Benutzer erkennen	110
Die Lebensdauer eines Skripts	111
Cookies überleben Ihr Skript	112
Cookies haben einen Namen, einen Wert ... und ein Verfallsdatum	117
JavaScript kann AUSSERHALB von Webseiten leben	119
Benutzer mit einem Cookie begrüßen	120
begruessen() jetzt mit Cookies	121
Vergessen Sie nicht, das Cookie zu schreiben	122
Cookies beeinflussen die Browsersicherheit	124
Ein Leben ohne Cookies	126
Mit den Benutzern reden ... das ist besser als nichts	129

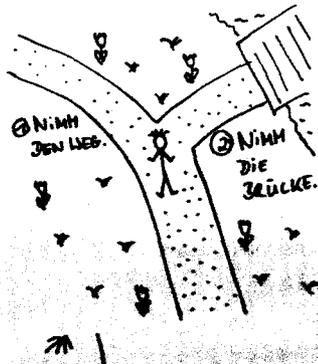
# Entscheidungen treffen

## 4

### Wenn eine Abzweigung kommt, folgen Sie ihr

Im Leben dreht sich alles um Entscheidungen. Stehen bleiben oder weitergehen, gerührt oder geschüttelt, Deal oder Gerichtsverfahren ... ohne die Fähigkeit, Entscheidungen zu treffen, würden wir nie etwas erreichen. In JavaScript ist es auch so – **durch Entscheidungen können Skripten zwischen verschiedenen Aktionen wählen. Entscheidungen erzählen die »Story« Ihrer Skripten weiter**, denn selbst die banalsten Skripten haben irgendeine Geschichte. Traue ich dem, was die Benutzerin eingegeben hat, und buche ihren Trip mit der Nessie-Expedition? Oder prüfe ich lieber nochmals, ob sie nicht in Wirklichkeit nur eine Busfahrt nach Loch Ness möchte? Sie haben die Wahl!

DIE ABENTEUER  
EINES  
STRICHMÄNNCHENS



Glückskandidat, kommen Sie zu mir!	136
Wenn dies zutrifft ... dann tu jenes	138
if-Anweisungen testen eine Bedingung – und schreiten zur Tat	139
Mit if zwischen zwei Dingen wählen	141
Sie haben mehrere Möglichkeiten mit if	142
Ein else für Ihre if-Anweisung	143
Ein Abenteuer epischen Ausmaßes	144
Variablen entwickeln die Story weiter	146
Verstärken Sie Ihre JavaScript-Bemühungen	148
Mehrstufige Entscheidungen mit if/else	154
Ein if darf in einem anderen if stehen	155
Ihre Funktionen steuern die Seiten	157
Mit Pseudocode das Abenteuer entwerfen	158
Strichmännchenungleichheit	162
!= Psst, ich habe nichts zu sagen	163
Entscheidungen mit Vergleichsoperatoren	164
Kommentare, Platzhalter und Dokumentation	166
JavaScript-Kommentare beginnen mit //	167
Geltungsbereich und Kontext: wo Daten lebendig sind	169
Überprüfen Sie die Abenteuervariablen	170
Wo existieren meine Daten?	171
Fünf an der Zahl	174
Verschachteltes if/else ist kompliziert	175
Switch-Anweisungen haben viele Fälle	177
Werden Sie Insider für switch-Anweisungen	178
Testlauf für eine switch/case-Anweisung	179
Testlauf für das Strichmännchenabenteuer mit switch	183

# Schleifen

## Auf die Gefahr, mich zu wiederholen

# 5

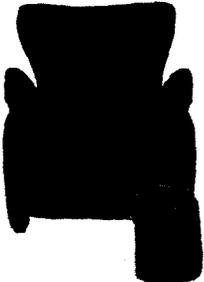
### Manche sagen, Wiederholungen seien die Würze des Lebens.

Sicher, etwas Neues und Interessantes ist bestimmt aufregend, aber es sind die kleinen, sich wiederholenden Dinge, die uns durch den Tag bringen. Zwanghafte Handhygiene, ein nervöser Tick, bei jeder verdammten Mail, die Sie bekommen, auf »Allen antworten« klicken ... Okay, vielleicht sind Wiederholungen in der realen Welt nicht ganz so großartig, aber in der JavaScript-Welt können sie extrem praktisch sein. Sie werden überrascht sein, wie oft Sie in einem Skript **Code-teile mehrmals ausführen** müssen. Und genau da entfaltet die Macht der Schleifen ihren Glanz. Ohne **Schleifen** würden Sie eine Menge Zeit damit vergeuden, unnötig viel Code zu kopieren und einzufügen.



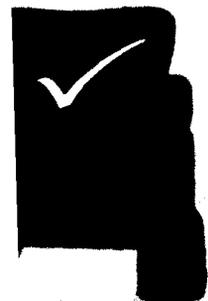
sitz\_frei.png

**Besetzt**



sitz\_besetzt.png

**Gewählt**



sitz\_gewaehlt.png

Ein X markiert die Stelle	190
Ein Déjà vu nach dem anderen – for-Schleifen	191
Schatzsuche mit einer for-Schleife	192
Die for-Schleife zerlegt	193
Mandango: Kinositzeuche für Machos	194
Verfügbarkeit der Sitze prüfen	195
Schleifen, HTML und Sitzverfügbarkeit	196
Kinositze als Variablen	197
Arrays nehmen mehrere Daten auf	198
Array-Werte werden mit Schlüsselwörtern gespeichert	199
Von JavaScript zu HTML	203
Die Mandango-Sitze visualisieren	204
Testlauf: die Solo-Sitzsuche	209
Zu viel des Guten: endlose Schleifen	210
Schleifen brauchen einen Ausgang (oder zwei!)	211
Ein »break« in Aktion	212
Eine logische, elegante und wohlgeformte Lösung mit &&	217
Boolesche Operatorlogik enthüllt	218
Für ein »while«-chen wiederholen ... bis eine Bedingung erfüllt ist	222
Die while-Schleife Schritt für Schritt	223
Die richtige Schleife finden	225
Kinositzdatenmodellierung	231
Ein Array eines Arrays: zweidimensionale Arrays	232
Zwei Schlüssel für die 2-D-Array-Daten	233
Mandango in 2-D	235
Ein Kino voller Männersitze	239

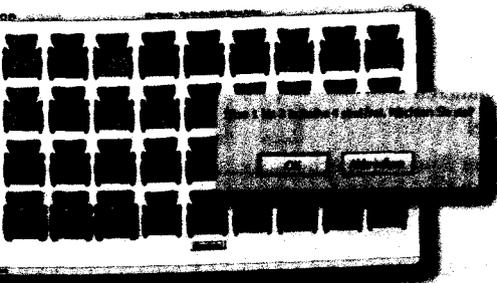
# 6

## Reduzieren, Wiederverwenden, Recyceln

Gäbe es in JavaScript eine Umweltbewegung, würde sie von Funktionen angeführt. Mit Funktionen können Sie JavaScript-Code effizienter machen. Und ja, wiederverwendbarer. Funktionen sind aufgabenorientiert, gut für die Codeorganisation und exzellente Problemlöser. Klingt nach allen Voraussetzungen für einen guten Lebenslauf! Abgesehen von ganz einfachen Exemplaren profitieren alle Skripten von einer funktionalen Reorganisation. Auch wenn es schwierig ist, die CO<sup>2</sup>-Bilanz der durchschnittlichen Funktion zu beziffern, so leisten sie dennoch ihren Beitrag, um Skripten so umweltfreundlich wie möglich zu machen.



Die Mutter aller Probleme	244
Funktionen als Problemlöser	246
Das A und O einer Funktion	247
Ein Funktion, die Sie bereits kennen	248
Ein besserer Thermostat mit mehr Daten	251
Informationen an Funktionen übergeben	252
Funktionsargumente als Daten	253
Funktionen eliminieren doppelten Code	254
Eine Sitzänderungsfunktion	257
sitzAendern() macht Mandango noch besser	259
Feedback ist wichtig	261
Daten aus Funktionen zurückgeben	262
Viele bunte Rückgabewerte	263
Den Sitzstatus ermitteln	267
Den Sitzstatus anzeigen	268
Funktionen mit Bildern verknüpfen	269
Codewiederholungen sind nicht gut	270
Trennen Sie die Funktionalität vom Inhalt	271
Funktionen sind nur Daten	272
Funktionen aufrufen oder referenzieren	273
Events, Callbacks und HTML-Attribute	277
Events mit Funktionsreferenzen verknüpfen	278
Funktionslitterale sind die Rettung	279
Wo ist die Verknüpfung?	280
Das Gerüst einer HTML-Seite	283



## Dem Benutzer alles entlocken

# 7

Sie müssen nicht höflich oder hinterhältig sein, um mit JavaScript Daten über den Benutzer zu erhalten. Aber Sie müssen vorsichtig sein. Menschen haben diese eigenartige Tendenz, Fehler zu machen. Deshalb **können Sie sich nicht** immer darauf verlassen, dass die Daten in Onlineformularen **korrekt** oder gültig sind. Da kommt JavaScript ins Spiel. Indem Sie **Formulardaten** während der Eingabe **durch den entsprechenden JavaScript-Code schleusen**, können Sie **Webapplikationen zuverlässiger** machen und **Last vom Server nehmen**. Wir müssen **die kostbare Bandbreite für wichtigere Dinge sparen** wie Stuntvideos und hübsche Playmates.

Bannerocity: himmlische Botschaften	290
Wenn HTML nicht reicht	292
Auf Formulardaten zugreifen	293
Formularfelder folgen einer Event-Kette	295
Den Fokus verlieren mit onblur	296
Ein alert-Fenster für Validierungsmeldungen	297
Felder validieren, damit Sie »nicht nichts« haben	301
Validierung ohne lästige alert-Fenster	302
Ein subtilerer Nicht-leer-Validierer	303
Es kommt auf die Größe an	305
Die Datenlänge validieren	306
Postleitzahlen validieren	311
Ein Datum validieren	316
Reguläre Ausdrücke sind nicht »regulär«	318
Reguläre Ausdrücke definieren Vergleichsmuster	319
Metazeichen repräsentieren mehr als literale Zeichen	321
Reguläre Ausdrücke eingebläut: Quantifizierer	322
Datenvalidierung mit regulären Ausdrücken	326
Minimum und Maximum matchen	329
Dreistellige Jahresangaben beseitigen (dies ... oder das)	331
Können Sie mich hören? Telefonnummernvalidierung	333
Sie haben Post: E-Mail validieren	334
Die Ausnahme ist die Regel	335
Optionale Zeichen aus einer Auswahl matchen	336
Einen E-Mail-Validierer bauen	337



**Bannerocity ... himmlische Werbung!**

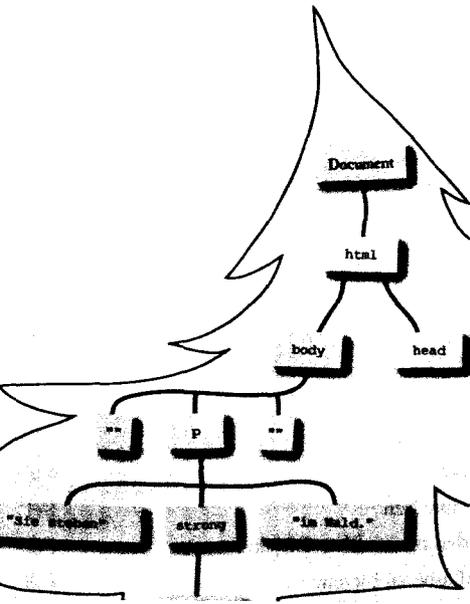


# 8

## HTML fein würfeln mit dem DOM

Den Webseiteninhalt mit JavaScript zu steuern ist ziemlich **ähnlich wie Backen**. Natürlich ohne das Chaos danach ... und leider auch ohne die essbare Belohnung. Aber: Sie erhalten den **vollen Zugriff auf die HTML-Zutaten** einer Webseite und, noch wichtiger, die Fähigkeit, das Rezept der Seite zu **ändern**. Mit **JavaScript können Sie den HTML-Code einer Webseite** nach Herzenslust manipulieren. Das eröffnet viele interessante Möglichkeiten, und alles dank einer **Sammlung von Standardobjekten** namens **DOM** (Document Object Model, Dokument-Objekt-Modell).

Funktional, aber plump ... Oberflächliches	344
Die Szenen ohne alert-Fenster beschreiben	345
Auf HTML-Elemente zugreifen	347
Kommen Sie mit dem inneren HTML in Berührung	348
Wald und Bäume sehen: das Document Object Model (DOM)	353
Ihre Seite ist eine Sammlung von DOM-Knoten	354
Mit Eigenschaften auf den DOM-Baum klettern	357
Knotentext mit dem DOM ändern	360
Standardkonforme Abenteuer	365
Auf der Suche nach besseren Optionen	366
Knotentextersetzung neu überdacht	368
Knotentext mit einer Funktion ersetzen	369
Dynamische Optionen sind eine tolle Sache	370
Interaktive Optionen sind noch besser	371
Stilfrage: CSS und DOM	372
Stilklassen wechseln	373
Klasse Optionen	374
Testen Sie die gestylten Abenteueroptionen	375
Missratene Optionen: der leere Button	376
Stil-Styling à la carte	377
Mehr Optionen, mehr Komplexität	380
Den Entscheidungsbaum mitverfolgen	382
Die Entscheidungschronik in HTML	383
HTML-Code fabrizieren	384
Die Abenteuergeschichte mitverfolgen	387



## Daten zum Leben erwecken

# Objekte als Frankensteindaten

# 9

JavaScript-Objekte sind nicht halb so grausig, wie der liebe Doktor Sie glauben macht. Aber sie sind insofern interessant, als sie verschiedene Teile der JavaScript-Sprache kombinieren, sodass sie zusammen mächtiger sind als die Einzelteile. **Objekte kombinieren Daten mit Aktionen** und schaffen so einen neuen **Datentyp**, der sehr viel »lebendiger« ist als die Daten, die Sie bisher gesehen haben. Dabei entstehen **Arrays, die sich selbst sortieren können, Strings, die sich selbst durchsuchen**, und Skripten, die haarig werden und den Mond anheulen! Okay, Letzteres vielleicht nicht, aber Sie verstehen, worauf wir hinauswollen ...

### Aktionen

```
function anzeigen (was, wann, wo) {  
    ...  
}  
function versenden (wer) {  
    ...  
}
```

### Objekt

```
var wer;  
var was;  
var wann;  
var wo;  
function anzeigen (  
    ...  
)  
function versenden (  
    ...  
)
```

Party mit JavaScript-Unterstützung	394
Daten + Aktionen = Objekt	395
Ein Objekt ist Eigentümer seiner Daten	396
Objektmitglieder mit einem Punkt referenzieren	397
Ein Blog für Würfeltüftler	399
Benutzerdefinierte Objekte erweitern JavaScript	401
Benutzerdefinierte Objekte konstruieren	402
Was gehört in einen Konstruktor?	403
Blog-Objekte zum Leben erwecken	404
Ein unordentlicher Blog	408
Ein JavaScript-Objekt für das Datum	410
Zeit berechnen	411
Das Blog-Datum überdenken	412
Ein Objekt in einem Objekt	413
Objekte zu Text konvertieren	416
Auf die Einzelteile eines Datums zugreifen	417
Arrays als Objekte	420
Arrays benutzerdefiniert sortieren	421
Sortieren leicht gemacht mit Funktionsliteralen	422
Das Blog-Array durchsuchen	425
In Strings suchen: indexOf()	427
Das Blog-Array durchsuchen	428
Math ist ein organisatorisches Objekt	434
Zufallszahlen mit Math.random	436
Eine Funktion zur Methode machen	441
Die Enthüllung des funkelnden neuen Blog-Objekts	442
Was haben Objekte für YouTube zu bieten?	443

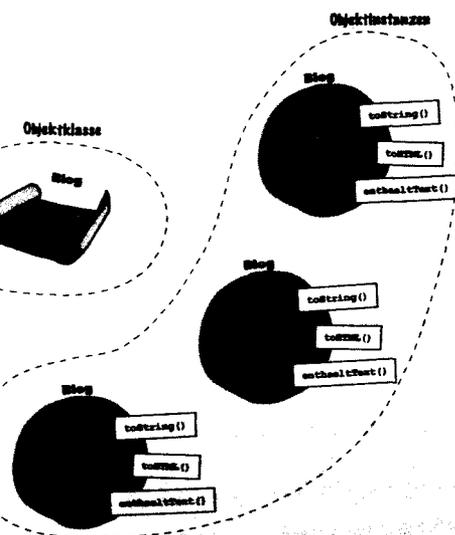
# 10

## Benutzerdefinierte Objekte erstellen

### Jedem das Seine mit benutzerdefinierten Objekten

JavaScript gewährt keine Geld-zurück-Garantie, aber Sie können definitiv machen, was Sie wollen. Benutzerdefinierte Objekte sind das JavaScript-Äquivalent eines entkoffinierten, großen, dreifachen, extra heißen, extra feinen, marmorierten Mocca macchiato ohne Sahne und ohne Schaum. Das nennen wir einen benutzerdefinierten Kaffee! Und mit benutzerdefinierten JavaScript-Objekten können Sie Code brühen, der genau das macht, was Sie wollen, während Sie die Vorteile von Eigenschaften und Methoden genießen. Das Resultat ist wiederverwendbarer, objektorientierter Code, der die JavaScript-Sprache effektiv erweitert ...extra für Sie!

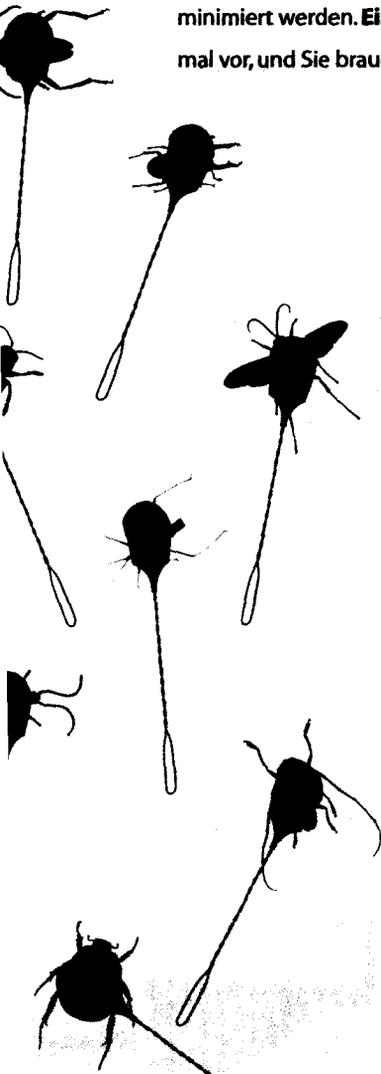
Die YouCube-Blog-Methoden überarbeiten	450
Zu viele Methoden	451
Klassen und Instanzen	452
Instanzen werden aus Klassen erstellt	453
Auf Instanzeigenschaften mit »this« zugreifen	454
Ein Eigentümer, mehrfache Ausführung: klasseneigene Methoden	455
Arbeiten Sie mit Prototypen auf Klassenebene	456
Klassen, Prototypen und YouCube	457
Klasseneigenschaften werden auch geteilt	462
Klasseneigenschaften erstellen mit prototype	463
Unterschrieben und abgeliefert	465
Doppelter Code ist tabu	467
Eine Methode zur Datumsformatierung	468
Standardobjekte erweitern	469
Benutzerdefiniertes Date-Objekt = besseres YouCube	470
Klassen können ihre eigenen Methoden haben	471
Die Sortiervergleichsfunktion untersuchen	473
Klassenmethoden aufrufen	474
Ein Bild sagt mehr als tausend Worte	475
Blog-Bilder in YouCube einbauen	476
Bildersprache in YouCube	478
Ein objektgetriebener YouCube	480



## Gute Skripten schief gewickelt

**Selbst die besten JavaScript-Pläne gehen manchmal schief.**

Wenn das passiert – und das wird es –, dürfen Sie keine Panik bekommen. Die besten JavaScript-Entwickler sind nicht die, die keine Fehler machen ... das sind die Lügner. Nein, die besten JavaScript-Entwickler sind diejenigen, die die von ihnen gemachten **Bugs erfolgreich aufspüren und ausmerzen**. Noch wichtiger ist es, dass erstklassige JavaScript-Bug-Bekämpfer **gute Programmiergewohnheiten entwickeln**, wodurch selbst die hinterhältigsten und miesesten Bugs minimiert werden. **Ein bisschen Prävention kann Sie weit bringen**. Aber Bugs kommen nun mal vor, und Sie brauchen ein ganzes Arsenal von Waffen, um sie zu bekämpfen!



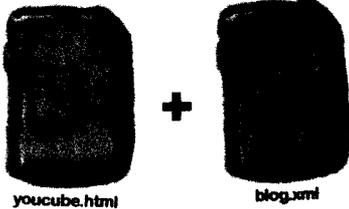
Debugging in der realen Welt	486
Der fehlerhafte IQ-Rechner	487
Versuchen Sie verschiedene Browser	488
Gut gelaunt debuggen	491
Wild gewordene undefinierte Variablen	495
Mit Intelligenzzahlen zaubern	497
Der Fall mit den Radioanrufer-Bugs	498
Eine Frage der Syntaxvalidierung (Bug Nr. 1)	500
Vorsicht mit den Strings	501
Anführungszeichen, Apostrophe und Konsistenz	502
Nicht nur Variablen können undefiniert sein (Bug Nr. 2)	504
Die üblichen Verdächtigen: die Checkliste	505
Jeder ist ein Gewinner (Bug Nr. 3)	506
Debuggen mit alert-Fenstern	507
Variablen beobachten mit alert	508
Alles Verlierer! (Bug Nr. 4)	514
Von lästigen alerts überflutet	515
Eine benutzerdefinierte Debug-Konsole bauen	517
Debuggen Sie Ihren Debugger	521
Die allerlästigsten Fehler: Laufzeitfehler	524
Die Bug-Dreierwette in JavaScript	525
Wenn Überwachung nicht reicht	527
Kommentare als vorübergehende Codeaktivierung	528
Das Risiko einer »Schattenvariablen«	530

# 12

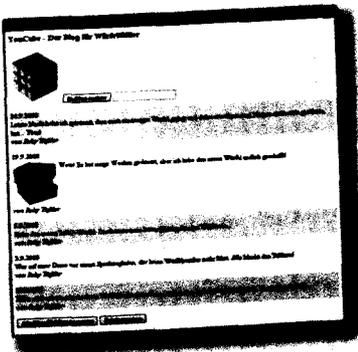
## Dynamische Daten

### Empathische Webapplikationen

Das moderne Web ist ein sehr interaktiver Ort, an dem Seiten nach Lust und Laune des Benutzers reagieren sollen. Oder zumindest ist das der Traum vieler Webuser und Entwickler. JavaScript spielt dabei eine entscheidende Rolle mit einer Programmier Technik namens **Ajax**, die einen Mechanismus bietet, mit dem der »Touch« von Webseiten dramatisch geändert werden kann. Mit Ajax verhalten sich Webseiten eher wie vollwertige Applikationen, da sie **Daten schnell und dynamisch laden und speichern** und damit **in Echtzeit auf den Benutzer eingehen können**. Und das ohne jede Seitenaktualisierung oder Browsertrickserei.



=



Sehnsucht nach dynamischen Daten	538
Ein datengetriebener YouCube	539
Ajax ist Kommunikation pur	541
Mit XML können Sie IHRE Daten auf IHRE Art taggen	543
XML + HTML = XHTML	545
XML und die YouCube-Blog-Daten	547
YouCube mit Ajax koppeln	550
JavaScript zur Rettung von Ajax: XMLHttpRequest	552
XMLHttpRequest ist ziemlich komplex	553
GET oder POST? Requests mit XMLHttpRequest	555
Sinnvolle Ajax-Requests	559
Interaktive Seiten beginnen mit einem Request-Objekt	563
Ruf mich, wenn du fertig bist	564
Antworten behandeln ... nahtlos	565
Das DOM hilft	566
YouCube wird datengetrieben	571
Gestörte Buttons	573
Die Buttons brauchen Daten	574
Zeitsparende, webbasierte Blog-Einträge	577
Blog-Daten schreiben	578
PHP hilft ... zumindest diesmal	579
Das PHP-Skript mit Daten füttern	582
Hochladen: Blog-Daten an den Server schicken	585
YouCube ... äh, benutzerfreundlicher machen	590
Felder für Benutzer autoausfüllen	591
Wiederkehrende Aufgaben? Wie wär's mit Funktionen?	592
Wie geht es jetzt weiter?	598