

Inhalt

Vorwort	XIII
1 Einführung	1
Muster	1
JavaScript: Konzepte	3
Objektorientiert	3
Keine Klassen	4
Prototypen	4
Umgebung	5
ECMAScript 5	5
JSLint	6
Die Konsole	6
2 Grundlagen	9
Wartbaren Code schreiben	9
Wenige globale Variablen verwenden	10
Das Problem mit globalen Variablen	11
Nebenwirkungen eines vergessenen var	12
Zugriff auf das globale Objekt	13
Single-var-Muster	14
Hoisting: Ein Problem mit verstreuten vars	15
for-Schleifen	16
for-in-Schleifen	18
Eingebaute Prototypen (nicht) erweitern	20
switch-Muster	21
Implizite Typecasts vermeiden	21
eval() vermeiden	22

Zahlen umwandeln mit <code>parseInt()</code>	23
Code-Richtlinien.	24
Einrücken	24
Geschweifte Klammern	25
Position der öffnenden Klammer	26
Leerraum	27
Namenskonventionen	28
Konstruktoren mit Großbuchstaben beginnen	29
Wörter trennen.	29
Andere Namens-Muster	29
Kommentare schreiben	31
Eine API-Dokumentation schreiben	31
YUIDoc-Beispiel.	32
Schreiben, um gelesen zu werden	35
Peer Reviews.	36
Minifizieren ... in der Produktionsumgebung.	37
JSLint nutzen	38
Zusammenfassung	39
3 Literale und Konstruktoren.	41
Objekt-Literale	41
Die Objekt-Literal-Syntax	42
Objekte aus einem Konstruktor	43
Object-Konstruktor-Überraschung.	43
Eigene Konstruktor-Funktionen.	44
Rückgabewerte von Konstruktoren	46
Muster zum Erzwingen von <code>new</code>	46
Namenskonventionen	47
<code>that</code> verwenden.	47
Selbst aufrufender Konstruktor	48
Array-Literal.	49
Die Array-Literal-Syntax	49
Kuriositäten im Array-Konstruktor.	49
Auf »Arrayhaftigkeit« prüfen	50
JSON.	51
Mit JSON arbeiten	51
Regexp-Literal	52
Regexp-Literal-Syntax	53

Wrapper für Primitive	54
Fehler-Objekte	56
Zusammenfassung	56
4 Funktionen	59
Hintergrund	59
Mehrdeutige Terminologie	60
Deklarationen versus Ausdrücke: Namen und Hoisting	61
name-Eigenschaft von Function	62
Funktions-Hoisting	63
Callback-Muster	65
Ein Callback-Beispiel	65
Callbacks und Scope	67
Asynchrone Event Listener	68
Timeouts	69
Callbacks in Bibliotheken	69
Funktionen zurückgeben	70
Selbst-definierende Funktionen	70
Immediate Function	72
Parameter einer Immediate Function	73
Rückgabewerte aus Immediate Functions	74
Vorteile und Anwendungsfälle	75
Immediate-Object-Initialisierung	76
Verzweigungen beim Initialisieren (Init-Time Branching)	77
Funktions-Eigenschaften – Ein Memoisierungs-Muster	79
Konfigurations-Objekte	80
Curry	82
Anwenden einer Funktion	82
Partielle Anwendung	83
Currying	84
Wann man Currying nutzt	86
Zusammenfassung	87
5 Objekt-Erzeugungs-Muster	89
Namensraum-Muster	89
Namensraum-Funktion	91
Abhängigkeiten deklarieren	93
Private Eigenschaften und Methoden	94
Private Member	94

Bevorzugte Methoden	95
Privacy-Lücken	95
Objekt-Literale und Privacy	97
Eigenschaften und Privacy	97
Private Funktionen als öffentliche Methoden bereitstellen (Revelation-Muster)	98
Modul-Muster	99
Bereitstellungs-Modul-Muster	101
Module, die Konstruktoren erzeugen	102
Globale Variablen in ein Modul importieren	103
Sandbox-Muster	103
Ein globaler Konstruktor	104
Module hinzufügen	105
Den Konstruktor implementieren	106
Statische Member	108
Öffentliche statische Member	108
Private statische Member	110
Objekt-Konstanten	111
Verkettungs-Muster	113
Vor- und Nachteile des Verkettungs-Musters	114
method()-Methode	114
Zusammenfassung	116
6 Muster zum Wiederverwenden von Code	117
Klassische versus moderne Vererbungs-Muster	117
Erwartete Ergebnisse bei klassischer Vererbung	118
Klassisches Muster Nr. 1 – Das Standard-Muster	119
Der Prototypen-Kette folgen	119
Nachteile bei Muster Nr. 1	122
Klassisches Muster Nr. 2 – Rent-a-Constructor	122
Die Prototypen-Kette	123
Mehrfachvererbung bei geliehenen Konstruktoren	124
Vor- und Nachteile des Geliehener-Konstruktor-Musters	125
Klassisches Muster Nr. 3 – Rent-and-Set-Prototyp	125
Klassisches Muster Nr. 4 – Gemeinsamer Prototyp	127
Klassisches Muster Nr. 5 – Ein temporärer Konstruktor	128
Die Superklasse speichern	129
Den Konstruktor-Zeiger zurücksetzen	130
Klass	131

Prototypische Vererbung	133
Diskussion	135
Ergänzung in ECMAScript 5	135
Vererbung durch das Kopieren von Eigenschaften	136
Mix-Ins	138
Methoden ausleihen	139
Beispiel: Von Array ausleihen	140
Ausleihen und Binden	140
Function.prototype.bind()	141
Zusammenfassung	142
7 Entwurfsmuster	143
Singleton	143
new verwenden	144
Instanz in einer statischen Eigenschaft	145
Instanz in einem Closure	146
Fabrik/Factory	148
Eingebaute Object-Fabrik	150
Iterator	151
Dekorierer/Decorator	153
Anwendung	153
Implementierung	154
Implementierung mit einer Liste	156
Strategie/Strategy	158
Beispiel: Datenväldierung	158
Fassade/Façade	161
Stellvertreter/Proxy	162
Ein Beispiel	163
Stellvertreter als Cache	170
Vermittler/Mediator	170
Vermittler-Beispiel	171
Beobachter/Observer	174
Beispiel Nr. 1: Zeitungs-Subskription	174
Beispiel Nr. 2: Das Tastatur-Spiel	178
Zusammenfassung	181
8 DOM- und Browser-Muster	183
Separation of Concerns	183

DOM Scripting	185
DOM-Zugriff	185
Veränderungen am DOM	186
Events	188
Event Handling	188
Event Delegation	190
Langlaufende Skripten.	191
setTimeout()	192
Web Workers.	192
Remote Scripting	193
XMLHttpRequest	193
JSONP.	195
Frames und Image Beacons	198
JavaScript-Code ausliefern	198
Skripten kombinieren	198
Minifizieren und Komprimieren.	199
Expires-Header.	200
Ein CDN verwenden.	200
Strategien zum Laden	200
Die Position des <script>-Elements	201
HTTP Chunking.	202
Dynamisches <script>-Element für nicht-blockierende Downloads.	204
Lazy-Loading	205
Loading on Demand.	206
JavaScript im Voraus laden	208
Zusammenfassung	209
Index.	211