

Inhaltsverzeichnis

Teil 1 Node.js ...

1

1	Was ist Node.js?	3
1.1	Die Zeitalter des Webs	3
1.1.1	1990 bis 2000: Das Web 1.0	3
1.1.2	2000 bis 2010: Das Web 2.0	5
1.1.3	2010 bis heute: Das Web 3.0	6
1.2	JavaScript – Fluch oder Segen?	8
1.2.1	Die Nachteile von JavaScript	8
1.2.2	Die Vorteile von JavaScript	9
1.2.3	Die Ausführungsgeschwindigkeit	9
1.2.4	JavaScript = LISP im Webbrowser	10
1.3	Node.js im Überblick	11
1.3.1	Wie baut man moderne Webanwendungen?	11
1.3.2	Die Idee hinter Node.js	11
1.3.3	Wofür eignet sich Node.js?	14
1.4	Zusammenfassung	15
2	Installation und Konfiguration	17
2.1	Die Qual der Wahl: Welche Plattform?	17
2.1.1	Linux	17
2.1.2	Mac OS X	18
2.1.3	Windows	18
2.2	Installation unter Linux	19
2.2.1	Node.js per Hand übersetzen	19
2.2.2	Vorgefertigte Pakete verwenden	20
2.3	Installation unter Mac OS X	21
2.3.1	Den Macintosh Installer verwenden	21
2.4	Installation unter Windows	23
2.4.1	Den Windows Installer verwenden	23

2.5	Testen der Installation	25
2.5.1	Node.js starten	25
2.5.2	Node.js beenden	26
2.5.3	Die installierte Version ermitteln	26
2.6	Zusammenfassung	27
3	Erste Schritte: Hallo Node.js!	29
3.1	Hallo Node.js!	29
3.1.1	Die Anwendung starten	29
3.1.2	Asynchrone Ausführung	30
3.1.3	Die Anwendung beenden	30
3.2	Einen http-Server implementieren	31
3.2.1	Das http-Modul	31
3.2.2	Einen http-Server starten	32
3.2.3	Der Umgang mit Headern	34
3.2.4	Eine Anwendung skalieren	35
3.3	Einen TCP-Server implementieren	36
3.3.1	Das net-Modul	36
3.3.2	Einen TCP-Server starten	36
3.3.3	Daten verarbeiten	38
3.4	Events	40
3.4.1	Das EventEmitter-Objekt	40
3.4.2	Auf Ereignisse reagieren	40
3.4.3	Ereignisse deregistrieren	41
3.5	Das Beispielprojekt	41
3.5.1	Was ist silkveil.js?	41
3.5.2	Anforderungen und Einschränkungen	42
3.5.3	Implementierung	43
3.6	Zusammenfassung	47
4	Module verwenden: require & Co.	49
4.1	Die require-Funktion	49
4.1.1	Integrierte Module	49
4.1.2	Externe Module als Datei	50
4.1.3	Externe Module als Verzeichnis	50
4.1.4	Das node_modules-Verzeichnis	51
4.1.5	Caching	52
4.1.6	Das exports-Objekt	52
4.2	Die integrierten Module im Überblick	53
4.2.1	Kategorien	53
4.2.2	TCP und Web	53

4.2.3	Lokales System	54
4.2.4	Daten und Kommunikation	55
4.2.5	Werkzeuge	55
4.3	Das Beispielprojekt	56
4.3.1	Anforderungen und Einschränkungen	56
4.3.2	Implementierung	56
4.4	Zusammenfassung	60
5	Node.js erweitern: Die Paketverwaltung npm	61
5.1	Der Node.js Package Manager (npm)	61
5.1.1	Was ist npm?	61
5.1.2	npm installieren	61
5.1.3	Installation testen	62
5.2	Umgang mit Modulen	62
5.2.1	Module installieren	62
5.2.2	Module aktualisieren	65
5.2.3	Module entfernen	66
5.2.4	Module suchen	66
5.3	Abhängigkeiten verwalten	67
5.3.1	Die Datei package.json	67
5.3.2	Abhängigkeiten automatisch auflösen	69
5.4	Eigene Module entwickeln	69
5.4.1	Ein Konto für npm anlegen	69
5.4.2	Ein Modul veröffentlichen	70
5.5	Das Beispielprojekt	71
5.5.1	Anforderungen und Einschränkungen	71
5.5.2	Implementierung	72
5.6	Zusammenfassung	75
6	Fehlersuche und -behebung: Arbeiten mit dem Debugger	77
6.1	Der integrierte Debugger von Node.js	77
6.1.1	Den Debugger starten	77
6.1.2	Die Ausführung steuern	79
6.1.3	Umgang mit Haltepunkten	82
6.1.4	Objekte überwachen	84
6.2	node-inspector	85
6.2.1	Grafisches Debuggen	85
6.2.2	Den Debugger starten	86
6.2.3	Die Ausführung steuern	88
6.2.4	Umgang mit Haltepunkten	89
6.2.5	Objekte überwachen	90

6.3	Das Beispielprojekt	92
6.3.1	Anforderungen und Einschränkungen	92
6.3.2	Implementierung	93
6.4	Zusammenfassung	96
7	Testen, testen, testen: Assert, Mocha & Co.	97
7.1	Arrange, Act, Assert	97
7.1.1	Wie funktioniert das »Assert«?	97
7.1.2	Das assert-Modul	98
7.1.3	should	101
7.1.4	node-assertthat	104
7.2	Mocha als Testumgebung	105
7.2.1	Was ist Mocha?	105
7.2.2	TDD oder BDD?	106
7.2.3	TDD mit Mocha	107
7.2.4	BDD mit Mocha	110
7.2.5	Testausgabe aufbereiten	111
7.3	Das Beispielprojekt	113
7.3.1	Anforderungen und Einschränkungen	113
7.3.2	Implementierung	114
7.4	Zusammenfassung	117
8	Deployment und Betrieb: Cluster, Nginx & Co.	119
8.1	Node.js neu starten	119
8.1.1	Das cluster-Modul	119
8.1.2	forever	121
8.2	Node.js als Dienst ausführen	124
8.2.1	Allgemeines	124
8.2.2	Upstart	124
8.3	Node.js intern hosten	126
8.3.1	Was ist Nginx?	126
8.3.2	Installation von Nginx	127
8.3.3	Konfiguration von Nginx	128
8.3.4	Integration von Node.js und Nginx	130
8.3.5	Node.js und IIS	132
8.4	Node.js extern hosten	135
8.4.1	Heroku	135
8.4.2	Windows Azure	138

8.5	Das Beispielprojekt	140
8.5.1	Anforderungen und Einschränkungen	140
8.5.2	Implementierung	140
8.6	Zusammenfassung	141
Teil 2	... & Co.	143
9	Eine Middleware für Node.js: Connect	145
9.1	Middleware entwickeln	145
9.1.1	Was ist Middleware?	145
9.1.2	Filter und Provider	146
9.1.3	Umgang mit Fehlern	147
9.2	Connect verwenden	148
9.2.1	Connect installieren	148
9.2.2	Zusammenfügen einer Anwendung	149
9.2.3	Setup-Funktionen für Module	151
9.2.4	Module an Routen binden	151
9.3	Die integrierten Module im Überblick	151
9.3.1	Kategorien	151
9.3.2	Anfragen und Antworten	152
9.3.3	Parser	153
9.3.4	Webserver und Sessions	153
9.3.5	Werkzeuge	154
9.4	Module von Drittanbietern	154
9.4.1	Datenquellen für Sessions	154
9.4.2	Sonstige Module	156
9.5	Das Beispielprojekt	156
9.5.1	Anforderungen und Einschränkungen	156
9.5.2	Implementierung	156
9.6	Zusammenfassung	158
10	Ein Grundgerüst für jede Webanwendung: Express	159
10.1	Express installieren	159
10.1.1	Installation	159
10.1.2	Eine Webanwendung erzeugen	160
10.1.3	Express global installieren	160
10.2	Webanwendungen konfigurieren	161
10.2.1	Konfigurationsblöcke verwenden	161
10.2.2	Konfigurationseinstellungen von Express	163

10.3	Routen verwenden	164
10.3.1	Routen definieren	164
10.3.2	Parameter verwenden	165
10.4	Middleware verwenden	166
10.4.1	Middleware für Routen	166
10.4.2	Middleware für Parameter	168
10.5	http-Methoden verarbeiten	168
10.5.1	POST	168
10.5.2	PUT und DELETE	169
10.5.3	Methodenunabhängiger Code	169
10.6	Webseiten dynamisch erzeugen	170
10.6.1	Vorlagen verwenden	170
10.6.2	Parser integrieren	171
10.7	Das Beispielprojekt	172
10.7.1	Anforderungen und Einschränkungen	172
10.7.2	Implementierung	172
10.8	Zusammenfassung	173
11	Vorlagen, HTML und CSS: Jade und Stylus	175
11.1	Jade	175
11.1.1	Installation und Konfiguration	175
11.1.2	Vorlagen definieren	176
11.1.3	Vorlagen und Daten verknüpfen	180
11.1.4	Vorlagen und Code verknüpfen	181
11.1.5	Vorlagen verschachteln	182
11.2	Stylus	184
11.2.1	Installation und Konfiguration	184
11.2.2	Vorlagen definieren	185
11.2.3	Vorlagen mit Code verknüpfen	187
11.3	Das Beispielprojekt	188
11.3.1	Anforderungen und Einschränkungen	188
11.3.2	Implementierung	188
11.4	Zusammenfassung	193
12	Client und Server verbinden: Socket.io und NowJS	195
12.1	Socket.io	196
12.1.1	Abstraktion von Websockets	196
12.1.2	Installation und Konfiguration	196
12.1.3	Client und Server verbinden	197
12.1.4	Vom Client zum Server	198

12.1.5	Vom Server zum Client	199
12.1.6	Ereignisse bestätigen	200
12.1.7	Socket.io und Firewalls	200
12.1.8	Socket.io und Hosting	200
12.2	NowJS	201
12.2.1	Abstraktion von Socket.io	201
12.2.2	Installation und Konfiguration	202
12.2.3	Client und Server verbinden	202
12.2.4	Vom Client zum Server	203
12.2.5	Vom Server zum Client	204
12.2.6	Verbindungen verwalten	205
12.3	Das Beispielprojekt	205
12.3.1	Anforderungen und Einschränkungen	205
12.3.2	Implementierung	206
12.4	Zusammenfassung	217
13	Datenbanken ansprechen: Redis, MongoDB und PostgreSQL	219
13.1	Redis	219
13.1.1	Datenbanktypen	219
13.1.2	Was ist Redis?	221
13.1.3	Installation und Konfiguration	221
13.1.4	Verbinden	222
13.1.5	Anweisungen ausführen	224
13.1.6	Datentypen	224
13.1.7	Performance	226
13.1.8	Replikation	228
13.2	MongoDB	229
13.2.1	Was ist MongoDB?	229
13.2.2	Installation und Konfiguration	231
13.2.3	Verbinden	232
13.2.4	Zugriff auf Collections	233
13.2.5	Zugriff auf Dokumente	234
13.2.6	Indizierung von Dokumenten	236
13.2.7	Replikation und Sharding	237
13.2.8	GridFS	238
13.3	PostgreSQL	241
13.3.1	NoSQL versus SQL	241
13.3.2	Installation und Konfiguration	242
13.3.3	Verbinden	242
13.3.4	Abfragen ausführen	243

13.4	Das Beispielprojekt	243
13.4.1	Anforderungen und Einschränkungen	243
13.4.2	Implementierung	243
13.5	Zusammenfassung	246
14	Verteilte Webanwendungen: Kue	247
14.1	Installation und Konfiguration	247
14.1.1	Kue installieren	247
14.1.2	Redis konfigurieren	248
14.2	Aufträge vergeben	248
14.2.1	Queue erzeugen	248
14.2.2	Aufträge vergeben	249
14.2.3	Prioritäten setzen	249
14.2.4	Wiederholen im Fehlerfall	250
14.2.5	Verzögertes Ausführen	250
14.2.6	Aufträge überwachen	250
14.2.7	Aufträge grafisch verwalten	251
14.2.8	Aufträge per REST verwalten	252
14.3	Aufträge verarbeiten	253
14.3.1	Queue erzeugen	253
14.3.2	Aufträge verarbeiten	253
14.3.3	Aktivität anzeigen	254
14.3.4	Aufträge parallel verarbeiten	254
14.4	Das Beispielprojekt	255
14.4.1	Anforderungen und Einschränkungen	255
14.4.2	Implementierung	255
14.5	Zusammenfassung	261
15	Infrastruktur verwenden: Amanda, Lingua und Passport	263
15.1	Amanda	263
15.1.1	Was ist Amanda?	263
15.1.2	Installation und Konfiguration	264
15.1.3	Schema definieren	264
15.1.4	Schema validieren	268
15.1.5	Fehler behandeln	269

15.2	Lingua	270
15.2.1	Was ist Lingua?	270
15.2.2	Installation und Konfiguration	270
15.2.3	Ressourcendateien	271
15.2.4	Statische Texte ausgeben	271
15.2.5	Dynamische Texte ausgeben	272
15.2.6	Sprache auswählen	272
15.3	Passport	273
15.3.1	Was ist Passport?	273
15.3.2	Installation und Konfiguration	273
15.3.3	Ressourcen schützen	274
15.3.4	Benutzername und Kennwort	275
15.3.5	OpenID	277
15.3.6	Andere Authentifizierungsanbieter	279
15.3.7	Abmelden	279
15.4	Das Beispielprojekt	279
15.4.1	Anforderungen und Einschränkungen	279
15.4.2	Implementierung	280
15.5	Zusammenfassung	290
16	Code optimieren: Piler	291
16.1	Piler	291
16.1.1	Installation und Konfiguration	291
16.1.2	JavaScript-Dateien verarbeiten	292
16.1.3	CSS-Dateien verarbeiten	294
16.1.4	CSS-Dateien in Echtzeit aktualisieren	295
16.1.5	Namensräume verwenden	296
16.1.6	Code zwischen Client und Server teilen	297
16.1.7	Andere Compiler integrieren	298
16.2	Das Beispielprojekt	298
16.2.1	Anforderungen und Einschränkungen	298
16.2.2	Implementierung	299
16.3	Zusammenfassung	301
	Nachwort	303
	Index	305