

Inhalt

Geleitwort	15
Vorwort	17
Materialien zum Buch	20

1 Einführung 21

1.1 Einleitung	21
1.2 Entstehung und Historie	22
1.3 Einsatzgebiete von JavaScript	24
1.3.1 Clientseitige JavaScript-Webanwendungen	24
1.3.2 Serverseitige JavaScript-Anwendungen	25
1.3.3 Desktop-JavaScript-Anwendungen	26
1.3.4 Mobile JavaScript-Anwendungen	26
1.3.5 Embedded-Anwendungen	26
1.3.6 Popularität von JavaScript	26
1.4 Laufzeitumgebungen	27
1.4.1 V8	27
1.4.2 SpiderMonkey/TraceMonkey/JägerMonkey/OdinMonkey	27
1.4.3 JavaScriptCore	28
1.4.4 Chakra	28
1.4.5 Rhino	28
1.4.6 Nashorn	29
1.4.7 Dyn.js	29
1.4.8 Auswahl der richtigen Laufzeitumgebung	29
1.4.9 Interpreter und Just-in-time-Compiler	30
1.5 Entwicklungsumgebungen	30
1.5.1 IntelliJ WebStorm	31
1.5.2 Visual Studio Code	31
1.5.3 Aptana Studio 3	32
1.5.4 Sublime Text 2	33
1.5.5 NetBeans	33
1.5.6 JSFiddle, JSBin und Codepen	34
1.5.7 Fazit	35
1.6 Debugging-Tools	35
1.6.1 Das »console«-Objekt	36

1.6.2	Browser	37
1.6.3	Node.js Inspector	39
1.6.4	IDEs und Editoren	39
1.7	Einführung in die Sprache	40
1.7.1	Statische Typisierung vs. dynamische Typisierung	40
1.7.2	Datentypen und Werte	40
1.7.3	Variablen und Konstanten	49
1.7.4	Funktionen	51
1.7.5	Operatoren	55
1.7.6	Kontrollstrukturen und Schleifen	59
1.7.7	Fehlerbehandlung	61
1.7.8	Sonstiges Wissenswertes	63
1.8	Zusammenfassung und Ausblick	65
2	Funktionen und funktionale Aspekte	67
<hr/>		
2.1	Die Besonderheiten von Funktionen in JavaScript	67
2.1.1	Funktionen als First-Class-Objekte	68
2.1.2	Funktionen haben einen Kontext	76
2.1.3	Funktionen definieren einen Sichtbarkeitsbereich	80
2.1.4	Alternativen zum Überladen von Methoden	83
2.1.5	Funktionen als Konstruktorfunktionen	87
2.2	Standardmethoden jeder Funktion	87
2.2.1	Objekte binden mit der Methode »bind()«	87
2.2.2	Funktionen aufrufen über die Methode »call()«	89
2.2.3	Funktionen aufrufen über die Methode »apply()«	91
2.3	Einführung in die funktionale Programmierung	92
2.3.1	Eigenschaften funktionaler Programmierung	92
2.3.2	Unterschied zur objektorientierten Programmierung	93
2.3.3	Unterschied zur imperativen Programmierung	94
2.3.4	Funktionale Programmiersprachen und JavaScript	94
2.4	Von der imperativen Programmierung zur funktionalen Programmierung	94
2.4.1	Iterieren mit der Methode »forEach()«	95
2.4.2	Werte abbilden mit der Methode »map()«	96
2.4.3	Werte filtern mit der Methode »filter()«	97
2.4.4	Einen Ergebniswert ermitteln mit der Methode »reduce()«	99
2.4.5	Kombination der verschiedenen Methoden	100

2.5	Funktionale Techniken und Entwurfsmuster	101
2.5.1	Komposition	102
2.5.2	Rekursion	104
2.5.3	Closures	105
2.5.4	Partielle Auswertung	107
2.5.5	Currying	115
2.5.6	Das IIFE-Entwurfsmuster	117
2.5.7	Das Callback-Entwurfsmuster	118
2.5.8	Self-defining Functions	125
2.6	Funktionale reaktive Programmierung	127
2.6.1	Einführung	127
2.6.2	ReactiveX und RxJS	129
2.6.3	Praxisbeispiel: Drag & Drop	132
2.6.4	Praxisbeispiel: Echtzeitdaten über Web-Sockets	134
2.7	Zusammenfassung und Ausblick	137

3 Objektorientierte Programmierung mit JavaScript 141

3.1	Objekte	141
3.1.1	Arten von Objekten	141
3.1.2	Objekte erstellen	142
3.2	Prototypen	154
3.3	Vererbung	158
3.3.1	Prototypische Vererbung	159
3.3.2	Pseudoklassische Vererbung	166
3.3.3	Vererbung mit Klassensyntax	171
3.3.4	Kopierende Vererbung	173
3.3.5	Mehrfachvererbung mit Mixins	175
3.4	Datenkapselung	189
3.4.1	Öffentliche Eigenschaften	189
3.4.2	Private Eigenschaften	190
3.4.3	Privilegierte öffentliche Methoden	191
3.4.4	Nichtprivilegierte öffentliche Methoden	192
3.4.5	Private Methoden	195
3.5	Emulieren von statischen Eigenschaften und statischen Methoden	196

3.6	Emulieren von Interfaces	199
3.6.1	Interfaces emulieren mit Attribute Checking	200
3.6.2	Interfaces emulieren mit Duck-Typing	201
3.7	Emulieren von Namespaces	202
3.8	Emulieren von Modulen	204
3.8.1	Das klassische Module-Entwurfsmuster	204
3.8.2	Das Revealing-Module-Entwurfsmuster	205
3.8.3	Importieren von Modulen	206
3.8.4	Module Augmentation	208
3.8.5	Asynchronous Module Definition (AMD) und CommonJS	209
3.8.6	Universal Module Definition (UMD)	211
3.9	Modulsyntax	212
3.9.1	Module exportieren	212
3.9.2	Module importieren	213
3.10	Zusammenfassung und Ausblick	215
4	ECMAScript 2015 und neuere Versionen	219
4.1	Einführung	219
4.2	Block-Scope und Konstanten	221
4.2.1	Block-Scope	221
4.2.2	Konstanten	226
4.3	Striktere Trennung zwischen Funktionen und Methoden	229
4.3.1	Arrow-Funktionen	230
4.3.2	Definition von Methoden	232
4.4	Flexiblerer Umgang mit Funktionsparametern	234
4.4.1	Beliebige Anzahl an Funktionsparametern	234
4.4.2	Abbilden von Arrays auf Funktionsparameter	236
4.4.3	Standardwerte für Funktionsparameter	238
4.4.4	Benannte Parameter	240
4.5	Mehrfachzuweisungen über Destructuring	243
4.5.1	Array-Destructuring	243
4.5.2	Objekt-Destructuring	248
4.6	Iteratoren und Generatoren	252
4.6.1	Iteratoren	252
4.6.2	Generatorfunktionen und Generatoren	255

4.7	Promises	258
4.8	Proxies	260
4.8.1	Proxies seit ES2015	261
4.8.2	Emulieren von Proxies in ES5	262
4.8.3	Anwendungsbeispiel: Proxy als Profiler	263
4.8.4	Anwendungsbeispiel: Proxy zur Validierung	264
4.9	Collections	265
4.9.1	Maps	265
4.9.2	Weak Maps	267
4.9.3	Sets	267
4.9.4	Weak Sets	269
4.10	Neue Methoden der Standardobjekte	269
4.10.1	Neue Methoden in »Object«	269
4.10.2	Neue Methoden in »String«	271
4.10.3	Neue Methoden in »Array«	273
4.10.4	Neue Methoden in »RegExp«, »Number« und »Math«	277
4.11	Sonstiges neue Features	279
4.11.1	Template-Strings	279
4.11.2	Symbole	282
4.11.3	»for-of«-Schleife	282
4.12	Zusammenfassung und Ausblick	283
5	Der Entwicklungsprozess	285
<hr/>		
5.1	Einleitung	285
5.2	Node.js und NPM	287
5.2.1	NPM installieren	288
5.2.2	Node.js-Anwendungen installieren	288
5.3	Styleguides und Code Conventions	289
5.3.1	Einrückungen	291
5.3.2	Semikolons	292
5.3.3	Anführungszeichen bei Strings	293
5.3.4	Variablendeklaration	293
5.3.5	Namenskonventionen	294
5.3.6	Klammern	295
5.4	Codequalität	295
5.4.1	JSLint	296

5.4.2	JSHint	297
5.4.3	ESLint	298
5.4.4	JSBeautifier	300
5.4.5	Google Closure Linter	301
5.4.6	Fazit	301
5.5	Dokumentation	302
5.5.1	JSDoc 3	302
5.5.2	YUIDoc	304
5.5.3	ESDoc	305
5.5.4	Unterstützte Tags	307
5.5.5	Fazit	310
5.6	Konkatenation, Minification und Obfuscation	310
5.6.1	YUI Compressor	313
5.6.2	Google Closure Compiler	314
5.6.3	UglifyJS2	316
5.6.4	Fazit	317
5.7	Package Management und Module Bundling	318
5.7.1	Package Management mit NPM	319
5.7.2	Module Bundling mit Webpack	326
5.7.3	Fazit	331
5.8	Building	332
5.8.1	Grunt	332
5.8.2	Gulp JS	336
5.8.3	NPM Scripts nutzen	337
5.8.4	Fazit	340
5.9	Scaffolding	341
5.9.1	Yeoman	342
5.9.2	Starterkits	347
5.10	Zusammenfassung und Ausblick	348

6 JavaScript-Anwendungen testen 351

6.1	Testgetriebene Entwicklung	351
6.1.1	Grundlagen und Begriffsdefinition	351
6.1.2	Testgetriebene Entwicklung in JavaScript	354
6.1.3	QUnit	355
6.1.4	mocha	361
6.1.5	Jest	370

6.1.6	Weitere Frameworks	372
6.1.7	Integration in Build-Tools	372
6.2	Test-Doubles	375
6.2.1	Sinon.JS	377
6.2.2	Spies	377
6.2.3	Stubs	383
6.2.4	Mock-Objekte	387
6.3	Testabdeckung	390
6.3.1	Einführung	390
6.3.2	Blanket.js	391
6.3.3	Istanbul	395
6.4	DOM-Tests	396
6.5	Funktionstests	399
6.5.1	PhantomJS	400
6.5.2	CasperJS	402
6.6	Zusammenfassung und Ausblick	406

7 Fortgeschrittene Konzepte der objektorientierten Programmierung 409

7.1	SOLID	409
7.1.1	Single-Responsibility-Prinzip (SRP)	410
7.1.2	Open-Closed-Prinzip (OCP)	413
7.1.3	Liskovsches Substitutionsprinzip (LSP)	419
7.1.4	Interface-Segregation-Prinzip (ISP)	423
7.1.5	Dependency-Inversion-Prinzip (DIP)	424
7.2	Fluent APIs	426
7.2.1	Einführung	426
7.2.2	Synchrone Fluent APIs	427
7.2.3	Asynchrone Fluent APIs mit Callbacks	430
7.2.4	Asynchrone Fluent APIs mit Promises	437
7.2.5	Zugriff auf Original-API	440
7.2.6	Generische Fluent API Factory	442
7.3	Aspektorientierte Programmierung in JavaScript	443
7.3.1	Einführung	443
7.3.2	Begrifflichkeiten	445
7.3.3	AOP durch Methodenneudefinition	446

7.3.4	Die JavaScript-Bibliothek meld	449
7.3.5	AOP über Decorators	453
7.3.6	Die Bibliothek aspect.js	455
7.4	Zusammenfassung und Ausblick	458

8 Die Entwurfsmuster der Gang of Four 459

8.1	Einführung	459
8.2	Erzeugungsmuster	460
8.2.1	Objekte an einer zentralen Stelle erzeugen (Abstract Factory/Factory Method)	461
8.2.2	Nur ein Objekt von einem Typ erstellen (Singleton)	466
8.2.3	Erstellen von komplexen Objekten (Builder)	469
8.2.4	Ähnliche Objekte erstellen (Prototype)	474
8.3	Strukturmuster	476
8.3.1	Die Schnittstelle anpassen (Adapter)	476
8.3.2	Abstraktion und Implementierung entkoppeln (Bridge)	480
8.3.3	Objekte in Baumstrukturen anordnen (Composite)	482
8.3.4	Eigenschaften unter Objekten teilen (Flyweight)	487
8.3.5	Objekte mit zusätzlichen Funktionalitäten ausstatten (Decorator)	492
8.3.6	Einheitliche Schnittstelle für mehrere Schnittstellen (Facade)	495
8.3.7	Den Zugriff auf Objekte abfangen (Proxy)	497
8.4	Verhaltensmuster	499
8.4.1	Über Datenstrukturen iterieren (Iterator)	500
8.4.2	Den Zugriff auf Objekte beobachten (Observer)	503
8.4.3	Eine Vorlage für einen Algorithmus definieren (Template Method)	508
8.4.4	Funktionen als Parameter übergeben (Command)	511
8.4.5	Algorithmen als Funktionen beschreiben (Strategy)	518
8.4.6	Das Zusammenspiel mehrerer Objekte koordinieren (Mediator)	522
8.4.7	Den Zustand eines Objekts speichern (Memento)	524
8.4.8	Operationen auf Objekten von Objekten entkoppeln (Visitor)	528
8.4.9	Das Verhalten eines Objekts abhängig vom Zustand ändern (State)	534
8.4.10	Eine Repräsentation für die Grammatik einer Sprache definieren (Interpreter)	539
8.4.11	Anfragen nach Zuständigkeit bearbeiten (Chain of Responsibility) ...	540
8.5	Zusammenfassung und Ausblick	544

9	Architekturmuster und Konzepte moderner JavaScript-Webframeworks	549
9.1	Model View Controller	550
9.2	Model View Presenter	551
9.3	MVC und MVP in Webanwendungen	551
9.3.1	Klassische Webanwendungen	552
9.3.2	Moderne Webanwendungen	553
9.4	Model View ViewModel	558
9.4.1	MVVM am Beispiel von Knockout.js	560
9.4.2	Kombination von MVC und MVVM am Beispiel von AngularJS	563
9.5	Komponentenbasierte Architektur	566
9.5.1	Komponentenbasierte Architektur am Beispiel von Angular	567
9.5.2	Komponentenbasierte Architektur am Beispiel von React	570
9.5.3	Komponentenbasierte Architektur am Beispiel von Vue.js	573
9.6	Routing	576
9.7	Zusammenfassung und Ausblick	578
10	Messaging	581
10.1	Einführung	581
10.2	AMQP	583
10.2.1	Producer und Consumer	584
10.2.2	Exchanges	585
10.3	AMQP unter JavaScript	587
10.3.1	Installation eines Message-Brokers für AMQP	587
10.3.2	AMQP-Clients für JavaScript	587
10.3.3	Senden und Empfangen von Nachrichten	588
10.3.4	Verwenden von Exchanges	592
10.3.5	STOMP	595
10.4	MQTT	598
10.4.1	Publish-Subscribe und Topics	599
10.4.2	Wildcards	600
10.4.3	Quality of Service	600
10.4.4	Last Will and Testament	601

- 10.4.5 Retained Messages 601
 - 10.4.6 Persistent Sessions 602
- 10.5 MQTT unter JavaScript 602
 - 10.5.1 Installation eines Message-Brokers für MQTT 602
 - 10.5.2 MQTT-Clients für JavaScript 604
- 10.6 Zusammenfassung und Ausblick 607

11 Continuous Integration611

- 11.1 Vorbereitungen 612
 - 11.1.1 Installation von Docker 612
 - 11.1.2 Installation des Git-Servers Gogs 614
 - 11.1.3 Anlegen eines Git-Repositorys 617
 - 11.1.4 Hinzufügen eines SSH-Schlüssels 617
 - 11.1.5 Anlegen des Beispielprojekts 618
- 11.2 Jenkins 619
 - 11.2.1 Installation 620
 - 11.2.2 Installieren von Plugins 624
 - 11.2.3 Anlegen von Jobs 626
 - 11.2.4 Jenkins mit Node.js steuern 636
- 11.3 Alternativen: Travis CI und CircleCI 638
- 11.4 Zusammenfassung und Ausblick 640

Index 641