

# Inhalt

Geleitwort .....	15
Vorwort .....	17
Materialien zum Buch .....	20

---

## 1 Einführung 21

<b>1.1 Einleitung .....</b>	21
<b>1.2 Entstehung und Historie .....</b>	22
<b>1.3 Einsatzgebiete von JavaScript .....</b>	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
<b>1.4 Laufzeitumgebungen .....</b>	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
<b>1.5 Entwicklungsumgebungen .....</b>	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
<b>1.6 Debugging-Tools .....</b>	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
<b>1.7</b>	<b>Einführung in die Sprache .....</b>	<b>40</b>
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
<b>1.8</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>65</b>

## **2 Funktionen und funktionale Aspekte**

---

<b>2.1</b>	<b>Die Besonderheiten von Funktionen in JavaScript .....</b>	<b>67</b>
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
<b>2.2</b>	<b>Standardmethoden jeder Funktion .....</b>	<b>87</b>
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
<b>2.3</b>	<b>Einführung in die funktionale Programmierung .....</b>	<b>92</b>
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
<b>2.4</b>	<b>Von der imperativen Programmierung zur funktionalen Programmierung .....</b>	<b>94</b>
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

<b>2.5</b>	<b>Funktionale Techniken und Entwurfsmuster</b>	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
<b>2.6</b>	<b>Funktionale reaktive Programmierung</b>	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
<b>2.7</b>	<b>Zusammenfassung und Ausblick</b>	137

## **3 Objektorientierte Programmierung mit JavaScript**

---

<b>3.1</b>	<b>Objekte</b>	141
3.1.1	Arten von Objekten .....	141
3.1.2	Objekte erstellen .....	142
<b>3.2</b>	<b>Prototypen</b>	154
<b>3.3</b>	<b>Vererbung</b>	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
<b>3.4</b>	<b>Datenkapselung</b>	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
<b>3.5</b>	<b>Emulieren von statischen Eigenschaften und statischen Methoden</b>	196

<b>3.6 Emulieren von Interfaces</b> .....	199
3.6.1 Interfaces emulieren mit Attribute Checking .....	200
3.6.2 Interfaces emulieren mit Duck-Typing .....	201
<b>3.7 Emulieren von Namespaces</b> .....	202
<b>3.8 Emulieren von Modulen</b> .....	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
<b>3.9 Modulsyntax</b> .....	212
3.9.1 Module exportieren .....	212
3.9.2 Module importieren .....	213
<b>3.10 Zusammenfassung und Ausblick</b> .....	215

---

<b>4 ECMAScript 2015 und neuere Versionen</b>	219
<b>4.1 Einführung</b> .....	219
<b>4.2 Block-Scope und Konstanten</b> .....	221
4.2.1 Block-Scope .....	221
4.2.2 Konstanten .....	226
<b>4.3 Striktere Trennung zwischen Funktionen und Methoden</b> .....	229
4.3.1 Arrow-Funktionen .....	230
4.3.2 Definition von Methoden .....	232
<b>4.4 Flexiblerer Umgang mit Funktionsparametern</b> .....	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
<b>4.5 Mehrfachzuweisungen über Destructuring</b> .....	243
4.5.1 Array-Destructuring .....	243
4.5.2 Objekt-Destructuring .....	248
<b>4.6 Iteratoren und Generatoren</b> .....	252
4.6.1 Iteratoren .....	252
4.6.2 Generatorfunktionen und Generatoren .....	255

<b>4.7</b>	<b>Promises</b>	258
<b>4.8</b>	<b>Proxies</b>	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
<b>4.9</b>	<b>Collections</b>	265
4.9.1	Maps	265
4.9.2	Weak Maps	267
4.9.3	Sets	267
4.9.4	Weak Sets	269
<b>4.10</b>	<b>Neue Methoden der Standardobjekte</b>	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
<b>4.11</b>	<b>Sonstiges neue Features</b>	279
4.11.1	Template-Strings	279
4.11.2	Symbole	282
4.11.3	»for-of«-Schleife	282
<b>4.12</b>	<b>Zusammenfassung und Ausblick</b>	283

---

<b>5</b>	<b>Der Entwicklungsprozess</b>	285
<b>5.1</b>	<b>Einleitung</b>	285
<b>5.2</b>	<b>Node.js und NPM</b>	287
5.2.1	NPM installieren	288
5.2.2	Node.js-Anwendungen installieren	288
<b>5.3</b>	<b>Styleguides und Code Conventions</b>	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
<b>5.4</b>	<b>Codequalität</b>	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
<b>5.5</b>	<b>Dokumentation .....</b>	<b>302</b>
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
<b>5.6</b>	<b>Konkatenation, Minification und Obfuscation .....</b>	<b>310</b>
5.6.1	YUI Compressor .....	313
5.6.2	Google Closure Compiler .....	314
5.6.3	UglifyJS2 .....	316
5.6.4	Fazit .....	317
<b>5.7</b>	<b>Package Management und Module Bundling .....</b>	<b>318</b>
5.7.1	Package Management mit NPM .....	319
5.7.2	Module Bundling mit Webpack .....	326
5.7.3	Fazit .....	331
<b>5.8</b>	<b>Building .....</b>	<b>332</b>
5.8.1	Grunt .....	332
5.8.2	Gulp JS .....	336
5.8.3	NPM Scripts nutzen .....	337
5.8.4	Fazit .....	340
<b>5.9</b>	<b>Scaffolding .....</b>	<b>341</b>
5.9.1	Yeoman .....	342
5.9.2	Starterkits .....	347
<b>5.10</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>348</b>

---

<b>6</b>	<b>JavaScript-Anwendungen testen .....</b>	<b>351</b>
<b>6.1</b>	<b>Testgetriebene Entwicklung .....</b>	<b>351</b>
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
<b>6.2</b>	<b>Test-Doubles .....</b>	<b>375</b>
6.2.1	SinonJS .....	377
6.2.2	Spies .....	377
6.2.3	Stubs .....	383
6.2.4	Mock-Objekte .....	387
<b>6.3</b>	<b>Testabdeckung .....</b>	<b>390</b>
6.3.1	Einführung .....	390
6.3.2	Blanket.js .....	391
6.3.3	Istanbul .....	395
<b>6.4</b>	<b>DOM-Tests .....</b>	<b>396</b>
<b>6.5</b>	<b>Funktionstests .....</b>	<b>399</b>
6.5.1	PhantomJS .....	400
6.5.2	CasperJS .....	402
<b>6.6</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>406</b>

## **7 Fortgeschrittene Konzepte der objektorientierten Programmierung**

---

<b>7.1</b>	<b>SOLID .....</b>	<b>409</b>
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
<b>7.2</b>	<b>Fluent APIs .....</b>	<b>426</b>
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
<b>7.3</b>	<b>Aspektorientierte Programmierung in JavaScript .....</b>	<b>443</b>
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
<b>7.4</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>458</b>
<b>8</b>	<b>Die Entwurfsmuster der Gang of Four</b>	<b>459</b>
<b>8.1</b>	<b>Einführung .....</b>	<b>459</b>
<b>8.2</b>	<b>Erzeugungsmuster .....</b>	<b>460</b>
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
<b>8.3</b>	<b>Strukturmuster .....</b>	<b>476</b>
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
<b>8.4</b>	<b>Verhaltensmuster .....</b>	<b>499</b>
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
<b>8.5</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>544</b>

## **9 Architekturmuster und Konzepte moderner JavaScript-Webframeworks**

---

549

<b>9.1 Model View Controller .....</b>	550
<b>9.2 Model View Presenter .....</b>	551
<b>9.3 MVC und MVP in Webanwendungen .....</b>	551
9.3.1 Klassische Webanwendungen .....	552
9.3.2 Moderne Webanwendungen .....	553
<b>9.4 Model View ViewModel .....</b>	558
9.4.1 MVVM am Beispiel von Knockout.js .....	560
9.4.2 Kombination von MVC und MVVM am Beispiel von AngularJS .....	563
<b>9.5 Komponentenbasierte Architektur .....</b>	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
<b>9.6 Routing .....</b>	576
<b>9.7 Zusammenfassung und Ausblick .....</b>	578

## **10 Messaging**

---

581

<b>10.1 Einführung .....</b>	581
<b>10.2 AMQP .....</b>	583
10.2.1 Producer und Consumer .....	584
10.2.2 Exchanges .....	585
<b>10.3 AMQP unter JavaScript .....</b>	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
<b>10.4 MQTT .....</b>	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
<b>10.5</b>	<b>MQTT unter JavaScript .....</b>	<b>602</b>
10.5.1	Installation eines Message-Brokers für MQTT .....	602
10.5.2	MQTT-Clients für JavaScript .....	604
<b>10.6</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>607</b>

## **11 Continuous Integration**

---

611

<b>11.1</b>	<b>Vorbereitungen .....</b>	<b>612</b>
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
<b>11.2</b>	<b>Jenkins .....</b>	<b>619</b>
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
<b>11.3</b>	<b>Alternativen: Travis CI und CircleCI .....</b>	<b>638</b>
<b>11.4</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>640</b>
<b>Index</b>	<b>.....</b>	<b>641</b>