

# Inhalt

Vorwort .....	13
<b>1 Einführung</b> .....	<b>17</b>
<b>1.1 Einleitung</b> .....	<b>17</b>
<b>1.2 Entstehung und Historie</b> .....	<b>18</b>
<b>1.3 Einsatzgebiete von JavaScript</b> .....	<b>19</b>
1.3.1 Clientseitige JavaScript-Webanwendungen .....	19
1.3.2 Serverseitige JavaScript-Anwendungen .....	20
1.3.3 Desktop-JavaScript-Anwendungen .....	21
1.3.4 Mobile JavaScript-Anwendungen .....	21
1.3.5 Embedded-Anwendungen .....	21
1.3.6 Popularität von JavaScript .....	21
<b>1.4 Laufzeitumgebungen</b> .....	<b>22</b>
1.4.1 V8 .....	22
1.4.2 SpiderMonkey/TraceMonkey/JägerMonkey/OdinMonkey .....	22
1.4.3 JavaScriptCore .....	23
1.4.4 Rhino .....	23
1.4.5 Nashorn .....	23
1.4.6 Dyn.js .....	23
1.4.7 Auswahl der richtigen Laufzeitumgebung .....	24
1.4.8 Interpreter und Just-in-time-Compiler .....	24
<b>1.5 Entwicklungsumgebungen</b> .....	<b>25</b>
1.5.1 Cloud9 .....	25
1.5.2 Aptana Studio 3 .....	26
1.5.3 Sublime Text 2 .....	27
1.5.4 NetBeans .....	27
1.5.5 IntelliJ WebStorm .....	28
1.5.6 JSFiddle, JSBin und Codepen .....	29
1.5.7 Fazit .....	30
<b>1.6 Debugging-Tools</b> .....	<b>30</b>
1.6.1 Das console-Objekt .....	30
1.6.2 Browser .....	32
1.6.3 node-inspector .....	34

<b>1.7</b>	<b>Einführung in die Sprache .....</b>	<b>34</b>
1.7.1	Statische Typisierung vs. dynamische Typisierung .....	34
1.7.2	Datentypen und Werte .....	35
1.7.3	Variablen und Konstanten .....	44
1.7.4	Funktionen .....	45
1.7.5	Operatoren .....	49
1.7.6	Kontrollstrukturen und Schleifen .....	53
1.7.7	Fehlerbehandlung .....	55
1.7.8	Sonstiges Wissenswertes .....	57
<b>1.8</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>58</b>

---

## 2 Funktionen und funktionale Aspekte 61

---

<b>2.1</b>	<b>Die Besonderheiten von Funktionen in JavaScript .....</b>	<b>61</b>
2.1.1	Funktionen sind First-Class-Objekte .....	62
2.1.2	Funktionen haben einen Kontext .....	70
2.1.3	Funktionen definieren einen Sichtbarkeitsbereich .....	73
2.1.4	Alternativen zum Überladen von Methoden .....	77
2.1.5	Funktionen als Konstruktorfunktionen .....	80
<b>2.2</b>	<b>Standardmethoden jeder Funktion .....</b>	<b>80</b>
2.2.1	Objekte binden mit der Methode »bind()« .....	81
2.2.2	Funktionen aufrufen über die Methode »call()« .....	83
2.2.3	Funktionen aufrufen über die Methode »apply()« .....	84
<b>2.3</b>	<b>Einführung in die funktionale Programmierung .....</b>	<b>85</b>
2.3.1	Eigenschaften funktionaler Programmierung .....	85
2.3.2	Unterschied zur objektorientierten Programmierung .....	86
2.3.3	Unterschied zur imperativen Programmierung .....	87
2.3.4	Funktionale Programmiersprachen und JavaScript .....	87
<b>2.4</b>	<b>Von der imperativen Programmierung zur funktionalen Programmierung .....</b>	<b>87</b>
2.4.1	Iterieren mit der Methode »forEach()« .....	88
2.4.2	Werte abbilden mit der Methode »map()« .....	89
2.4.3	Werte filtern mit der Methode »filter()« .....	90
2.4.4	Einen Ergebniswert ermitteln mit der Methode »reduce()« .....	91
2.4.5	Kombination der verschiedenen Methoden .....	93
<b>2.5</b>	<b>Funktionale Techniken und Entwurfsmuster .....</b>	<b>94</b>
2.5.1	Komposition .....	94

2.5.2	Rekursion .....	97
2.5.3	Closures .....	97
2.5.4	Partielle Auswertung .....	100
2.5.5	Currying .....	107
2.5.6	Das IIFE-Entwurfsmuster .....	109
2.5.7	Das Callback-Entwurfsmuster .....	110
2.5.8	Self-Defining Functions .....	117
<b>2.6</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>119</b>

## 3 Objektorientierte Programmierung mit JavaScript 121

---

<b>3.1</b>	<b>Objekte .....</b>	<b>121</b>
3.1.1	Arten von Objekten .....	121
3.1.2	Objekte erstellen .....	122
<b>3.2</b>	<b>Prototypen .....</b>	<b>133</b>
<b>3.3</b>	<b>Vererbung .....</b>	<b>136</b>
3.3.1	Prototypische Vererbung .....	137
3.3.2	Pseudoklassische Vererbung .....	145
3.3.3	Kopierende Vererbung .....	150
<b>3.4</b>	<b>Datenkapselung .....</b>	<b>152</b>
3.4.1	Öffentliche Eigenschaften .....	152
3.4.2	Private Eigenschaften .....	153
3.4.3	Privilegierte öffentliche Methoden .....	153
3.4.4	Nichtprivilegierte öffentliche Methoden .....	154
3.4.5	Private Methoden .....	156
<b>3.5</b>	<b>Emulieren von statischen Eigenschaften und statischen Methoden .....</b>	<b>157</b>
<b>3.6</b>	<b>Emulieren von Interfaces .....</b>	<b>159</b>
3.6.1	Interfaces emulieren mit Attribute Checking .....	160
3.6.2	Interfaces emulieren mit Duck Typing .....	161
<b>3.7</b>	<b>Emulieren von Namespaces .....</b>	<b>162</b>
<b>3.8</b>	<b>Emulieren von Modulen .....</b>	<b>164</b>
3.8.1	Das klassische Module-Entwurfsmuster .....	164
3.8.2	Das Revealing-Module-Entwurfsmuster .....	165
3.8.3	Importieren von Modulen .....	166

3.8.4	Module Augmentation .....	168
3.8.5	AMD, CommonJS und ECMAScript-6-Module .....	169
<b>3.9</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>171</b>
<b>4</b>	<b>ECMAScript 6 .....</b>	<b>173</b>
<hr/>		
<b>4.1</b>	<b>Einführung .....</b>	<b>173</b>
<b>4.2</b>	<b>Block-Scope und Konstanten .....</b>	<b>175</b>
4.2.1	Block-Scope .....	175
4.2.2	Konstanten .....	180
<b>4.3</b>	<b>Striktere Trennung zwischen Funktionen und Methoden .....</b>	<b>183</b>
4.3.1	Arrow-Funktionen .....	184
4.3.2	Definition von Methoden .....	186
<b>4.4</b>	<b>Flexiblerer Umgang mit Funktionsparametern .....</b>	<b>188</b>
4.4.1	Beliebige Anzahl an Funktionsparametern .....	188
4.4.2	Abbilden von Arrays auf Funktionsparameter .....	190
4.4.3	Standardwerte für Funktionsparameter .....	191
4.4.4	Benannte Parameter .....	194
<b>4.5</b>	<b>Mehrfachzuweisungen über Destructuring .....</b>	<b>196</b>
4.5.1	Array-Destructuring .....	196
4.5.2	Objekt-Destructuring .....	200
<b>4.6</b>	<b>Iteratoren und Generatoren .....</b>	<b>203</b>
4.6.1	Iteratoren .....	204
4.6.2	Generatorfunktionen und Generatoren .....	206
<b>4.7</b>	<b>Promises .....</b>	<b>209</b>
<b>4.8</b>	<b>Proxies .....</b>	<b>212</b>
4.8.1	Proxies in ES6 .....	212
4.8.2	Emulieren von Proxies in ES5 .....	214
4.8.3	Anwendungsbeispiel: Proxy als Profiler .....	215
4.8.4	Anwendungsbeispiel: Proxy zur Validierung .....	215
<b>4.9</b>	<b>Collections .....</b>	<b>216</b>
4.9.1	Maps .....	216
4.9.2	Weak-Maps .....	218
4.9.3	Sets .....	219
4.9.4	Weak-Sets .....	220

<b>4.10</b>	<b>Module</b>	220
4.10.1	Module exportieren	221
4.10.2	Module importieren	222
<b>4.11</b>	<b>Klassen</b>	223
4.11.1	Definition von Klassen	223
4.11.2	Vererbung	224
<b>4.12</b>	<b>Neue Methoden der Standardobjekte</b>	226
4.12.1	Neue Methoden in »Object«	226
4.12.2	Neue Methoden in »String«	227
4.12.3	Neue Methoden in »Array«	230
4.12.4	Neue Methoden in »RegExp«, »Number« und »Math«	233
<b>4.13</b>	<b>Sonstiges neue Features</b>	235
4.13.1	Template-Strings	235
4.13.2	Symbole	238
4.13.3	for-of-Schleife	238
<b>4.14</b>	<b>Zusammenfassung und Ausblick</b>	239

## 5 Der Entwicklungsprozess 241

---

<b>5.1</b>	<b>Einleitung</b>	241
<b>5.2</b>	<b>Styleguides und Code Conventions</b>	244
<b>5.3</b>	<b>Codequalität</b>	251
5.3.1	JSLint	251
5.3.2	JSHint	252
5.3.3	ESLint	253
5.3.4	JSBeautifier	255
5.3.5	Google Closure Linter	256
5.3.6	Fazit	257
<b>5.4</b>	<b>Dokumentation</b>	257
5.4.1	JSDoc 3	257
5.4.2	YUIDoc	259
5.4.3	JSFuck 5	260
5.4.4	Unterstützte Tags	261
5.4.5	Fazit	263
<b>5.5</b>	<b>Konkatenation, Minification und Obfuscation</b>	263
5.5.1	YUI Compressor	265

5.5.2	Google Closure Compiler .....	266
5.5.3	UglifyJS 2 .....	267
5.5.4	Fazit .....	270
<b>5.6</b>	<b>Package Management .....</b>	<b>271</b>
5.6.1	Backend Package Management mit NPM .....	272
5.6.2	Frontend Package Management mit Bower .....	279
5.6.3	Fazit .....	285
<b>5.7</b>	<b>Building .....</b>	<b>285</b>
5.7.1	Grunt .....	286
5.7.2	Gulp JS .....	289
5.7.3	Fazit .....	291
<b>5.8</b>	<b>Scaffolding .....</b>	<b>291</b>
5.8.1	Yeoman .....	292
5.8.2	Lineman .....	297
<b>5.9</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>299</b>
<b>6</b>	<b>JavaScript-Anwendungen testen .....</b>	<b>301</b>
<b>6.1</b>	<b>Testgetriebene Entwicklung .....</b>	<b>301</b>
6.1.1	Grundlagen und Begriffsdefinition .....	301
6.1.2	Testgetriebene Entwicklung in JavaScript .....	304
6.1.3	JUnit .....	305
6.1.4	mocha .....	311
6.1.5	Integration in Build-Tools .....	320
<b>6.2</b>	<b>Test-Doubles .....</b>	<b>324</b>
6.2.1	Sinon.JS .....	325
6.2.2	Spies .....	325
6.2.3	Stubs .....	331
6.2.4	Mock-Objekte .....	334
<b>6.3</b>	<b>Testabdeckung .....</b>	<b>336</b>
6.3.1	Einführung .....	337
6.3.2	Blanket.js .....	337
<b>6.4</b>	<b>DOM-Tests .....</b>	<b>341</b>
<b>6.5</b>	<b>Funktionstests .....</b>	<b>344</b>
6.5.1	PhantomJS .....	344
6.5.2	CasperJS .....	347
<b>6.6</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>350</b>

<b>7.1</b>	<b>Einführung</b> .....	353
<b>7.2</b>	<b>Erzeugungsmuster</b> .....	354
7.2.1	Objekte an einer zentralen Stelle erzeugen (Abstract Factory/Factory Method) .....	355
7.2.2	Nur ein Objekt von einem Typ erstellen (Singleton) .....	358
7.2.3	Erstellen von komplexen Objekten (Builder) .....	360
7.2.4	Ahnliche Objekte erstellen (Prototype) .....	363
<b>7.3</b>	<b>Strukturmuster</b> .....	366
7.3.1	Die Schnittstelle anpassen (Adapter) ..	366
7.3.2	Abstraktion und Implementierung entkoppeln (Bridge) .....	370
7.3.3	Objekte in Baumstrukturen anordnen (Composite) .....	371
7.3.4	Eigenschaften unter Objekten teilen (Flyweight) .....	375
7.3.5	Objekte mit zusätzlichen Funktionalitäten ausstatten (Decorator) .....	379
7.3.6	Einheitliche Schnittstelle für mehrere Schnittstellen (Facade) .....	381
7.3.7	Den Zugriff auf Objekte abfangen (Proxy) .....	383
<b>7.4</b>	<b>Verhaltensmuster</b> .....	385
7.4.1	Über Datenstrukturen iterieren (Iterator) .....	385
7.4.2	Den Zugriff auf Objekte beobachten (Observer) .....	388
7.4.3	Eine Vorlage für einen Algorithmus definieren (Template Method) .....	393
7.4.4	Funktionen als Parameter übergeben (Command) .....	396
7.4.5	Algorithmen als Funktionen beschreiben (Strategy) .....	400
7.4.6	Das Zusammenspiel mehrerer Objekte koordinieren (Mediator) .....	403
7.4.7	Den Zustand eines Objekts speichern (Memento) .....	404
7.4.8	Operationen auf Objekten von Objekten entkoppeln (Visitor) .....	406
7.4.9	Das Verhalten eines Objekts abhängig vom Zustand ändern (State) .....	411
7.4.10	Eine Repräsentation für die Grammatik einer Sprache definieren (Interpreter) .....	414
7.4.11	Anfragen nach Zuständigkeit bearbeiten (Chain of Responsibility) ...	415
<b>7.5</b>	<b>Zusammenfassung und Ausblick</b> .....	418

<b>8</b>	<b>Architekturmuster und Konzepte moderner JavaScript-Webframeworks</b>	<b>423</b>
<b>8.1</b>	<b>Model View Controller</b>	<b>423</b>
<b>8.2</b>	<b>Model View Presenter</b>	<b>424</b>
<b>8.3</b>	<b>MVC und MVP in Webanwendungen</b>	<b>425</b>
8.3.1	Klassische Webanwendungen	425
8.3.2	Moderne Webanwendungen	427
<b>8.4</b>	<b>Model View ViewModel</b>	<b>432</b>
8.4.1	MVVM am Beispiel von Knockout.js	434
8.4.2	Kombination von MVC und MVVM am Beispiel von AngularJS	437
<b>8.5</b>	<b>Routing</b>	<b>440</b>
8.5.1	Routing am Beispiel von AngularJS	441
8.5.2	Routing am Beispiel von Backbone.js	442
<b>8.6</b>	<b>Zusammenfassung und Ausblick</b>	<b>443</b>
	<b>Index</b>	<b>445</b>