

Inhalt

| | |
|---|-----------|
| Vorwort | 13 |
| 1 Einführung | 17 |
| 1.1 Einleitung | 17 |
| 1.2 Entstehung und Historie | 18 |
| 1.3 Einsatzgebiete von JavaScript | 19 |
| 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 |
| 1.4 Laufzeitumgebungen | 22 |
| 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 |
| 1.5 Entwicklungsumgebungen | 25 |
| 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 |
| 1.6 Debugging-Tools | 30 |
| 1.6.1 Das console-Objekt | 30 |
| 1.6.2 Browser | 32 |
| 1.6.3 node-inspector | 34 |

| | | |
|------------|--|----|
| 1.7 | Einführung in die Sprache | 34 |
| 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 |
| 1.8 | Zusammenfassung und Ausblick | 58 |

2 Funktionen und funktionale Aspekte

| | | |
|------------|---|----|
| 2.1 | Die Besonderheiten von Funktionen in JavaScript | 61 |
| 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 |
| 2.2 | Standardmethoden jeder Funktion | 80 |
| 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 |
| 2.3 | Einführung in die funktionale Programmierung | 85 |
| 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 |
| 2.4 | Von der imperativen Programmierung zur funktionalen Programmierung | 87 |
| 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 |
| 2.5 | Funktionale Techniken und Entwurfsmuster | 94 |
| 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 |
| 2.6 | Zusammenfassung und Ausblick | 119 |

3 Objektorientierte Programmierung mit JavaScript

| | | |
|------------|---|------------|
| 3.1 | Objekte | 121 |
| 3.1.1 | Arten von Objekten | 121 |
| 3.1.2 | Objekte erstellen | 122 |
| 3.2 | Prototypen | 133 |
| 3.3 | Vererbung | 136 |
| 3.3.1 | Prototypische Vererbung | 137 |
| 3.3.2 | Pseudoklassische Vererbung | 145 |
| 3.3.3 | Kopierende Vererbung | 150 |
| 3.4 | Datenkapselung | 152 |
| 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 |
| 3.5 | Emulieren von statischen Eigenschaften und statischen Methoden | 157 |
| 3.6 | Emulieren von Interfaces | 159 |
| 3.6.1 | Interfaces emulieren mit Attribute Checking | 160 |
| 3.6.2 | Interfaces emulieren mit Duck Typing | 161 |
| 3.7 | Emulieren von Namespaces | 162 |
| 3.8 | Emulieren von Modulen | 164 |
| 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 |
| 3.9 | Zusammenfassung und Ausblick | 171 |

4 ECMAScript 6

| | | |
|------------|--|------------|
| 4.1 | Einführung | 173 |
| 4.2 | Block-Scope und Konstanten | 175 |
| 4.2.1 | Block-Scope | 175 |
| 4.2.2 | Konstanten | 180 |
| 4.3 | Striktere Trennung zwischen Funktionen und Methoden | 183 |
| 4.3.1 | Arrow-Funktionen | 184 |
| 4.3.2 | Definition von Methoden | 186 |
| 4.4 | Flexiblerer Umgang mit Funktionsparametern | 188 |
| 4.4.1 | Beliebige Anzahl an Funktionsparametern | 188 |
| 4.4.2 | Abilden von Arrays auf Funktionsparameter | 190 |
| 4.4.3 | Standardwerte für Funktionsparameter | 191 |
| 4.4.4 | Benannte Parameter | 194 |
| 4.5 | Mehrfachzuweisungen über Destructuring | 196 |
| 4.5.1 | Array-Destructuring | 196 |
| 4.5.2 | Objekt-Destructuring | 200 |
| 4.6 | Iteratoren und Generatoren | 203 |
| 4.6.1 | Iteratoren | 204 |
| 4.6.2 | Generatorfunktionen und Generatoren | 206 |
| 4.7 | Promises | 209 |
| 4.8 | Proxies | 212 |
| 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 |
| 4.9 | Collections | 216 |
| 4.9.1 | Maps | 216 |
| 4.9.2 | Weak-Maps | 218 |
| 4.9.3 | Sets | 219 |
| 4.9.4 | Weak-Sets | 220 |

| | | |
|-------------|--|-----|
| 4.10 | Module | 220 |
| 4.10.1 | Module exportieren | 221 |
| 4.10.2 | Module importieren | 222 |
| 4.11 | Klassen | 223 |
| 4.11.1 | Definition von Klassen | 223 |
| 4.11.2 | Vererbung | 224 |
| 4.12 | Neue Methoden der Standardobjekte | 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 |
| 4.13 | Sonstiges neue Features | 235 |
| 4.13.1 | Template-Strings | 235 |
| 4.13.2 | Symbole | 238 |
| 4.13.3 | for-of-Schleife | 238 |
| 4.14 | Zusammenfassung und Ausblick | 239 |

| | | |
|------------|--|-----|
| 5 | Der Entwicklungsprozess | 241 |
| 5.1 | Einleitung | 241 |
| 5.2 | Styleguides und Code Conventions | 244 |
| 5.3 | Codequalität | 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 |
| 5.4 | Dokumentation | 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 |
| 5.5 | Konkatenation, Minification und Obfuscation | 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 |
| 5.6 | Package Management | 271 |
| 5.6.1 | Backend Package Management mit NPM | 272 |
| 5.6.2 | Frontend Package Management mit Bower | 279 |
| 5.6.3 | Fazit | 285 |
| 5.7 | Building | 285 |
| 5.7.1 | Grunt | 286 |
| 5.7.2 | Gulp JS | 289 |
| 5.7.3 | Fazit | 291 |
| 5.8 | Scaffolding | 291 |
| 5.8.1 | Yeoman | 292 |
| 5.8.2 | Lineman | 297 |
| 5.9 | Zusammenfassung und Ausblick | 299 |

6 **JavaScript-Anwendungen testen**

| | | |
|------------|--|------------|
| 6.1 | Testgetriebene Entwicklung | 301 |
| 6.1.1 | Grundlagen und Begriffsdefinition | 301 |
| 6.1.2 | Testgetriebene Entwicklung in JavaScript | 304 |
| 6.1.3 | QUnit | 305 |
| 6.1.4 | mocha | 311 |
| 6.1.5 | Integration in Build-Tools | 320 |
| 6.2 | Test-Doubles | 324 |
| 6.2.1 | Sinon.JS | 325 |
| 6.2.2 | Spies | 325 |
| 6.2.3 | Stubs | 331 |
| 6.2.4 | Mock-Objekte | 334 |
| 6.3 | Testabdeckung | 336 |
| 6.3.1 | Einführung | 337 |
| 6.3.2 | Blanket.js | 337 |
| 6.4 | DOM-Tests | 341 |
| 6.5 | Funktionstests | 344 |
| 6.5.1 | PhantomJS | 344 |
| 6.5.2 | CasperJS | 347 |
| 6.6 | Zusammenfassung und Ausblick | 350 |

| | | |
|------------|---|-----|
| 7.1 | Einführung | 353 |
| 7.2 | Erzeugungsmuster | 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 |
| 7.3 | Strukturmuster | 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 |
| 7.4 | Verhaltensmuster | 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 |
| 7.5 | Zusammenfassung und Ausblick | 418 |

8 Architekturmuster und Konzepte moderner JavaScript-Webframeworks

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