

---

# Inhalt

<b>Vorwort</b> .....	<b>XIII</b>
----------------------	-------------

---

## Teil 1: ~~Hallo Welt~~ Hallo Welt

<b>1 Bin ich hier richtig?</b> .....	<b>3</b>
<b>2 Zwischen Hybris und Demut</b> .....	<b>7</b>
Schwächen als Stärken .....	9
Richtiges muss nicht schwierig sein .....	12

---

## Teil 2: Programmieren als Verständigung

<b>3 Du bist wie die andern</b> .....	<b>17</b>
<b>4 Konventionen</b> .....	<b>19</b>
Englisch oder nicht? .....	20
Die Steinchen des Anstoßes .....	23
Konventionen im Team .....	26
<b>5 Namensgebung</b> .....	<b>29</b>
Namenskonventionen .....	29
Von Byzanz über Konstantinopel nach Istanbul .....	31
Was Namen können sollten .....	33
Der Stoff, aus dem die Namen sind .....	40
Boolesche Variablen .....	50

Objektorientierte Programmierung .....	52
Datenbanken .....	53
Falsche Freunde .....	55
Wie es weitergeht .....	58
<b>6 Kommentare .....</b>	<b>61</b>
Mehr ist manchmal mehr .....	63
Zur äußeren Form von Kommentaren .....	64
Dokumentationskommentare .....	66
Wann und was soll man kommentieren? .....	67
Anzeichen, dass ein Kommentar eine gute Idee wäre .....	69
Problematische Kommentare .....	74
<b>7 Code lesen .....</b>	<b>77</b>
Muss ich wirklich? .....	77
Zuerst die Dokumentation lesen .....	79
Sourcecode ausdrucken .....	80
Zeichnen Sie schematisch auf, was einzelne Programmteile tun .....	81
Von oben nach unten, von leicht nach schwer .....	82
Lernen Sie Spurenlesen .....	82
80/20 ist gut genug (meistens) .....	83
Vergessen Sie die Daten nicht .....	84
Der Beweis ist das Programm .....	84
Gemeinsames Code-Lesen .....	85
<b>8 Hilfe suchen .....</b>	<b>87</b>
Der richtige Zeitpunkt .....	88
An der richtigen Stelle fragen .....	91
Die Anfrage richtig strukturieren .....	91
An den Leser denken .....	94
Nicht zu viel erwarten .....	95
Keine unbewussten Fallen stellen .....	96
Höflich bleiben – egal, was passiert .....	96
<b>Lizenz zum Helfen .....</b>	<b>99</b>
Der falsche Anlass .....	99
Die eigennützige Motivation .....	101
Die fehlende Einfühlung .....	102
Zu viel auf einmal .....	103
Antworten auf konkrete Fragen .....	105
Wenn Sie selbst keine Antwort wissen .....	106

Wenn Sie mit schlechteren Programmierern zusammenarbeiten .....	107
Schlechten Code gefasst ertragen .....	108
<b>9 Überleben im Team .....</b>	<b>111</b>
Ich war's nicht! .....	113
Der Bus-Faktor .....	114
Zusammenarbeit mit Anwendern .....	116
Zusammenarbeit mit Freiwilligen .....	117
Aussprache von Begriffen .....	117

---

### Teil 3: Umgang mit Fehlern

<b>10 Unrecht haben für Anfänger .....</b>	<b>123</b>
Im Irrtum zu Hause .....	124
Fehlerforschung im Alltag .....	125
Der Hund hat die Datenbank gefressen! .....	126
Der gepolsterte Helm .....	127
<b>11 Debugging I: Fehlersuche als Wissenschaft .....</b>	<b>131</b>
Systematische Fehlersuche .....	133
Beobachtung .....	135
Was das Beobachten erschwert .....	136
Analyse und Hypothesenbildung .....	138
Was das Bilden von Hypothesen erschwert .....	138
Test der Hypothesen .....	139
Was das Testen von Hypothesen erschwert .....	140
<b>12 Debugging II: Finde den Fehler .....</b>	<b>143</b>
Fehlermeldungen sind unsere Freunde .....	143
Wer will da was von mir? .....	144
Diagnosewerkzeuge und -strategien .....	147
Wenn sonst nichts hilft .....	160
Wenn auch das nicht hilft .....	162
Die häufigsten Fehlerursachen schlechter Programmierer .....	163
<b>13 Schlechte Zeichen oder Braune M&amp;Ms .....</b>	<b>165</b>
Zu große Dateien .....	166
Sehr lange Funktionen .....	167
Zu breite Funktionen .....	167

Tief verschachtelte if/then-Bedingungen	168
Mitten im Code auftauchende Zahlen	170
Komplexe arithmetische Ausdrücke im Code	170
Globale Variablen	171
Reparaturcode	172
Eigene Implementierung vorhandener Funktionen	173
Sonderfälle	174
Inkonsistente Schreibweisen	174
Funktionen mit mehr als fünf Parametern	174
Code-Duplikation	175
Zweifelhafte Dateinamen	176
Leselabyrinth	176
Ratlose Kommentare	176
Sehr viele Basisklassen oder Interfaces	177
Sehr viele Methoden oder Member-Variablen	177
Auskommentierte Codeblöcke und Funktionen	178
Browservorschriften	178
Verdächtige Tastaturgeräusche	179
<b>14 Refactoring</b>	<b>181</b>
Neu schreiben oder nicht?	182
Wann sollte man refakturieren?	183
Eins nach dem anderen	186
Code auf mehrere Dateien verteilen	191
Ein Codemodul in kleinere aufspalten	191
Nebenwirkungen entfernen	194
Code zusammenfassen	195
Bedingungen verständlicher gestalten	198
Die richtige Schleife für den richtigen Zweck	201
Schleifen verständlicher gestalten	201
Variablen kritisch betrachten	203
Refactoring von Datenbanken	204
Was man nebenbei erledigen kann	206
Ist das jetzt wirklich besser?	208
Wann man auf Refactoring besser verzichtet	208
Ein Problem und seine Lösung	211
<b>15 Testing</b>	<b>213</b>
Warum testen?	213
Testverfahren	214
Datenvalidierungen	220

Performancetests	222
Richtig testen	225
<b>16 Warnhinweise</b>	<b>227</b>
GET und POST	228
Zeichenkodierung	229
Zeitangaben	230
Kommazahlen als String, Integer oder Decimal speichern	232
Variablen als Werte oder Referenzen übergeben	233
Der schwierige Umgang mit dem Nichts	236
Rekursion	237
Usability	238
<b>17 Kompromisse</b>	<b>241</b>
Trügerische Tugenden	243
Absolution: Wann Bad Practice okay ist	247

---

## Teil 4: Wahl der Mittel

<b>18 Mach es nicht selbst</b>	<b>255</b>
Der Weg zur Lösung	257
Bibliotheken	258
Umgang mit Fremdcode	261
Was man nicht selbst zu machen braucht	262
<b>19 Werkzeugkasten</b>	<b>273</b>
Editoren	274
Welche Programmiersprache ist die richtige?	275
REPL	279
Diff und Patch	282
Paketmanager	284
Frameworks	286
Entwicklungsumgebungen	289
<b>20 Versionskontrolle</b>	<b>297</b>
Alternativen	299
Arbeiten mit einem VCS	300
Konflikte auflösen	302
Welches Versionskontrollsystem?	303
Gute Ideen beim Arbeiten mit Versionskontrolle	305

Schlechte Ideen beim Arbeiten mit Versionskontrolle .....	306
Versionskontrollsysteme als Softwarebausteine .....	307
<b>21 Command and Conquer – vom Überleben auf der Kommandozeile .....</b>	<b>309</b>
Mehr Effizienz durch Automatisierung .....	310
Unsere langbärtigen Vorfahren .....	312
Windows .....	313
Was jeder Programmierer wissen sollte .....	313
Navigation .....	318
Dateien .....	318
Betrachten .....	321
Suchen und Finden .....	322
Ressourcen schonen .....	325
Zusammenarbeit .....	326
Zeitsteuerung .....	326
Editieren auf dem Server .....	328
Internet .....	328
Muss ich mir das alles merken? .....	330
Not the whole Shebang! .....	330
<b>22 Objektorientierte Programmierung .....</b>	<b>333</b>
Vorteile der objektorientierten Programmierung .....	335
Die Prinzipien objektorientierter Programmierung .....	337
Sinnvoller Einsatz von OOP .....	344
Nachteile und Probleme .....	347
Unterschiedliche Objektmodelle, je nach Sprache .....	348
Objektorientierte Programmierung und Weltherrschaftspläne .....	348
<b>23 Aufbewahrung von Daten .....</b>	<b>351</b>
Dateien .....	352
Versionskontrollsysteme .....	357
Datenbanken .....	357
<b>24 Sicherheit .....</b>	<b>365</b>
Wichtige Konzepte .....	366
Vor- und Nachteile der Offenheit .....	368
Vom Umgang mit Passwörtern .....	370
Authentifizierungsverfahren .....	371
SQL Injection und XSS – die Gefahren in User-Content .....	375
Weißer Listen sind besser als schwarze .....	380
Alle Regler nach links .....	381

Auch die Hintertür abschließen .....	383
Penetration Testing .....	384
Die Fehler der anderen .....	385
Sicherheit ist ein Prozess .....	386
<b>25 Nützliche Konzepte .....</b>	<b>389</b>
Exceptions .....	389
Error Handling .....	392
State und Statelessness .....	396
IDs, GUIDs, UUIDs .....	397
Sprachfamilien .....	399
Variablentypen .....	401
Trennung von Inhalt und Präsentation .....	404
Trennung von Entwicklungs- und Produktivserver .....	405
Selektoren .....	406
Namespaces .....	408
Scope von Variablen .....	410
Assertions .....	411
Transaktionen und Rollbacks .....	414
Hashes, Digests, Fingerprints .....	415
CRUD und REST .....	417
<b>26 Wie geht es weiter? .....</b>	<b>419</b>
Was ist ein guter Programmierer? .....	420
Zum Weiterlesen .....	421
Danksagungen .....	422
<b>Index .....</b>	<b>423</b>