Clean SAPUI5

Lesbarer und wartbarer SAPUI5-Code

DAS INHALTS-VERZEICHNIS

Inhalt

Einleitung 17

1 Einführung 25

1.1 Was ist Clean SAPUI5? 26
1.1.1 Was ist Lesbarkeit? 26
1.1.2 Was ist die Geschichte von Clean SAPUI5? 27
1.2 Erste Schritte mit Clean SAPUI5 28
1.3 Umgang mit Legacy-Code 30
1.4 Code automatisch prüfen 32
1.5 Wie hängt Clean SAPUI5 mit anderen Leitfäden zusammen? 32
1.6 Zusammenfassung 34

2 JavaScript und SAPUI5 35

2.1 Funktionen von JavaScript ES6+ 36
2.1.1 Browserunterstützung 37
2.1.2 Schlüsselwort »const« 39
2.1.3 Hoisting und Function-Scoping mit »var« 43
2.1.4 Block Scoping mit »let« und »const« 44
2.1.5 Arrow-Funktionen 44
2.1.6 Template-Literale 47
2.1.7 Spread-Syntax 50
2.1.8 Destrukturierungszuordnung 64
2.1.9 Promises, »async« und »await« 70
2.1.10 »for await of«-Anweisungen 95
2.1.11 Standardparameter 99
2.1.12 Klassen 101
2.1.13 Module 115
2.2 TypeScript 119
2.2.1 Transpiling und Source-Mapping 121
2.2.2 Starke Typisierung 122

2.2.3 Features ... 123

2.3 Zusammenfassung ... 131

3 Projektstruktur 133

3.1 Komponenten in SAPUI5 ... 133

3.2 Wichtige Artefakte ... 136

3.2.1 Component Controller ... 137

3.2.2 Deskriptor ... 143

3.2.3 Root-View ... 144

3.2.4 Die Datei »index.html« ... 144

3.3 Freestyle-Anwendungen ... 145

3.4 SAP Fiori Elements ... 147

3.4.1 Floorplans ... 150

3.4.2 Anwendungsgenerierung ... 153

3.5 Bibliotheksprojekte ... 155

3.5.1 Die Datei »library.js« ... 155

3.5.2 SAPUI5-Reuse-Komponenten ... 156

3.5.3 Ordnerstruktur ... 157

3.6 Model-View-Controller-Assets ... 158

3.7 Zusammenfassung ... 161

4 Module und Klassen 163

4.1 Controller-Inflation ... 164

4.1.1 Model-View-Controller-Muster ... 165

4.1.2 Objekt-View ... 169

4.1.3 Dialoge ... 193

4.2 Modullebenszyklus ... 199

4.2.1 Ausführungskontext ... 200

4.2.2 Laden und Caching ... 201

4.2.3 Lebenszyklus einer Single-Page-Anwendung ... 204

4.3 Wiederverwendbarkeit und Testbarkeit ... 208

4.3.1 Schnittstelle definieren ... 208

4.3.2 Modulzustand ... 210

4.3.3 Dokumentieren ... 213
4.3.4 Modultests ... 214
4.3.5 Dependency Mocking ... 215

4.4 Servicemodule vs. Klassenmodule ... 219
4.4.1 Servicemodule ... 220
4.4.2 Klassenmodule ... 222

4.5 Zusammenfassung ... 226

5 Funktionen 229

5.1 Funktionsdefinition ... 229

5.2 Funktionsobjekt ... 231

5.3 Instanzmethoden ... 233

5.4 Event-Handler und Callbacks ... 236

5.5 Ausführungskontext der Callback-Funktion ... 236

5.6 Getter und Setter ... 238

5.7 Anonyme Funktionen ... 241

5.8 Funktionsparameter ... 243
5.8.1 Funktionsstelligkeit ... 243
5.8.2 Boolesche Parameter ... 248
5.8.3 REST-Parameter ... 252
5.8.4 Destrukturierungsparameter ... 253
5.8.5 Standardwerte ... 256

5.9 Promises ... 258

5.10 Generatoren ... 265

5.11 Funktionskörper ... 266
5.11.1 Do one Thing ... 266
5.11.2 Eine Abstraktionsebene absteigen ... 268
5.11.3 Funktionen klein halten ... 272

5.12 Funktionen aufrufen ... 274

5.13 Closures ... 277

5.14 Zusammenfassung ... 278

6 Namensgebung 281

6.1 Beschreibende Namen 282
6.2 Domänenbegriffe 283
6.3 Entwurfsmuster 285
6.4 Konsistenz 285
6.5 Namen kürzen 287
6.6 Füllwörter 288
6.7 Casing 289
6.8 Klassen und Enums 291
6.9 Funktionen und Methoden 292
6.9.1 Gängige Methoden für SAPUI5 293
6.9.2 Event-Handler 293
6.10 Variablen und Parameter 294
6.10.1 Boolesche Werte 294
6.10.2 Schleifenvariablen 295
6.10.3 Vergleichsfunktionen und -parameter 298
6.10.4 Ereignisparameter 299
6.10.5 Konstanten 299
6.11 Private Elemente 300
6.12 Namensräume 302
6.13 Control-IDs 304
6.14 Ungarische Notation 305
6.15 Alternative Regeln 307
6.16 Zusammenfassung 309

7 Variablen und Literale 311

7.1 Variablen 311
7.2 Literale 316
7.2.1 Zahlen 317
7.2.2 Zeichenfolge 322
7.2.3 Boolesche Werte 326
7.2.4 Reguläre Ausdrücke 327

7.2.5 Arrays ... 329
7.2.6 Objekte ... 331
7.2.7 »null« und »undefined« ... 333

7.3 Zusammenfassung ... 335

8 Kontrollfluss 337

8.1 Bedingungen ... 338
8.1.1 »if«, »else if« und »else« ... 338
8.1.2 »switch« ... 340

8.2 Schleifen ... 342
8.2.1 »while«-Schleifen ... 342
8.2.2 »do...while«-Schleifen ... 342
8.2.3 »for«-Schleifen ... 343
8.2.4 »for...of«-Schleifen ... 343
8.2.5 »for...in«-Schleifen ... 344

8.3 Bedingte Komplexität ... 345
8.3.1 Leere »if«-Zweige vermeiden ... 345
8.3.2 Positive Bedingungen aufbauen ... 346
8.3.3 Komplexe Bedingungen zerlegen ... 346
8.3.4 Bedingungen kapseln ... 347
8.3.5 Steuerparameter vermeiden ... 348
8.3.6 Verschachtelte Anweisungen vermeiden ... 349
8.3.7 Deklarativen Stil gegenüber imperativem Stil bevorzugen ... 352

8.4 Zusammenfassung ... 353

9 Fehlerbehandlung 355

9.1 »throw«- und »try/catch«-Anweisungen ... 355

9.2 Fehlerobjekte verwenden ... 357

9.3 Fehlerbehandlung über Meldungen ... 359

9.4 Fehlerbehandlung mit Controls ... 361

9.5 Best Practices für die Fehlerbehandlung ... 365

9.6 Zusammenfassung ... 369

10 Formatierung 371

10.1 Motivation ... 371

10.2 Vertikale und horizontale Formatierung ... 372
10.2.1 Vertikale Formatierung ... 373
10.2.2 Horizontale Formatierung ... 379

10.3 Textbereich ein- oder ausrücken ... 382

10.4 XML-Views ... 385
10.4.1 Vertikaler Abstand und Eigenschaften ... 385
10.4.2 Aggregationen ... 387
10.4.3 Bindings ... 389

10.5 Weitere Hinweise ... 391
10.5.1 Tabs vs. Leerzeichen ... 391
10.5.2 Zeilenenden ... 392
10.5.3 Dangling Commas ... 392
10.5.4 Ternäre Operatoren und Inlinelogik ... 393

10.6 Formatierung für TypeScript in SAPUI5 ... 395
10.6.1 Typen vs. Schnittstellen ... 396
10.6.2 Klassen ... 397
10.6.3 Definitionsdateien ... 400
10.6.4 Namensräume und Module ... 400
10.6.5 Enums ... 403
10.6.6 »import«-Anweisung ... 405
10.6.7 Vorgeschlagene Formatierungsstandards ... 406

10.7 Erstellen und Pflegen eines Codestil-Leitfadens ... 407

10.8 Formatierungswerkzeuge ... 409
10.8.1 Formatierung vs. Linting ... 409
10.8.2 EditorConfig ... 411
10.8.3 Prettier ... 412

10.9 Zusammenfassung ... 415

11 Kommentare 417

11.1 Drücken Sie Ihre Absicht im Code aus ... 418

11.2 Das Gute: Kommentarplatzierung und -nutzung ... 419
11.2.1 Absichtserklärende Kommentare ... 419

11.2.2 Zusammenfassungskommentare ... 422
11.2.3 Kommentare an Variablen- und Parameterdefinitionen ... 425
11.2.4 Parametername oder Argumentkommentare ... 427
11.2.5 JSDoc-Kommentare ... 428

11.3 Das Schlechte: Zu vermeidende oder umzustrukturierende Kommentare 431
11.3.1 Formatierung und Ausrichtung ... 432
11.3.2 Abschnitte und Trennzeichen ... 435
11.3.3 Auskommentierter Code ... 438
11.3.4 No-Code-Kommentare ... 440
11.3.5 Schließende-Klammer-Kommentare ... 441
11.3.6 Metakommentare ... 441

11.4 Das Hässliche: Sonderkommentare ... 442
11.4.1 Rechtliche Kommentare ... 442
11.4.2 To-do- und andere Code-Tags ... 443
11.4.3 Pragma-Kommentare ... 444

11.5 Zusammenfassung ... 445

12 Statische Codeprüfungen und Codemetriken 447

12.1 Linting ... 449
12.1.1 JavaScript-Linter ... 450
12.1.2 XML-Linting ... 466

12.2 Codemetriken ... 469
12.2.1 Zyklomatische Komplexität ... 470
12.2.2 Schachtelung ... 475
12.2.3 Funktionsparameter ... 481
12.2.4 Funktionslänge ... 483
12.2.5 Fan-in/Fan-out ... 485
12.2.6 Codeduplizierung ... 487
12.2.7 Wartbarkeit ... 489

12.3 Zusammenfassung ... 491

13 Testen 493

13.1 Prinzipien ... 494
13.1.1 Testbaren Code schreiben ... 494

13.1.2 Testpyramide ... 497
13.1.3 FIRST-Prinzipien der Testautomatisierung ... 500
13.1.4 Testgetriebene Entwicklung ... 501
13.1.5 Mocking nutzen ... 502
13.1.6 WDI5 vs. OPA5 vs. QUnit ... 503
13.1.7 Page Objects ... 504
13.1.8 Codeabdeckung realistisch bewerten ... 506
13.1.9 Regeln zur Lesbarkeit ... 507

13.2 Zu testender Code ... 508
13.2.1 Aussagekräftige Namen für zu testenden Code ... 509
13.2.2 Aufruf des zu testenden Codes in die eigene Testmethode extrahieren ... 509

13.3 Injektion ... 510
13.3.1 Konstruktorinjektion ... 511
13.3.2 Setter-Injektion ... 511
13.3.3 Sinon.JS ... 512
13.3.4 Prototype ... 513
13.3.5 OData-V2-Mock-Server ... 515

13.4 Testmethoden und Journeys ... 516
13.4.1 Namen von Testmethoden ... 516
13.4.2 Journeys: Der größere Kontext des Testens ... 517
13.4.3 Journey-Schritte: Was wird gemacht? ... 517
13.4.4 Given-When-Then verwenden ... 518

13.5 Testdaten ... 519
13.5.1 Die Bedeutung von Testdaten eindeutig machen ... 519
13.5.2 Das Erkennen von Unterschieden erleichtern ... 520
13.5.3 Verwendung von Konstanten zur Beschreibung des Zwecks und der Wichtigkeit von Testdaten ... 520
13.5.4 Testdaten für OData-V2-Mock-Server ... 521

13.6 Assertions ... 523
13.6.1 Sich auf wenige fokussierte Assertions beschränken ... 523
13.6.2 Richtigen Assert-Typ verwenden ... 524
13.6.3 Inhalte validieren und nicht die Quantität ... 525
13.6.4 Qualität validieren und nicht Inhalte ... 525
13.6.5 Auf erwartete Ausnahmen prüfen ... 526
13.6.6 Benutzerdefinierte Assertions schreiben, um Code zu kürzen und Duplizierung zu vermeiden ... 526
13.6.7 Aussagekräftige Fehlermeldungen bei Timeouts für Page Objects anzeigen ... 528

13.7 Zusammenfassung ... 529

14 TypeScript und verwandte Technologien 531

14.1 TypeScript 531
14.1.1 Aktueller Status 533
14.1.2 Verwendung von SAPUI5 mit TypeScript 534
14.1.3 TypeScript zu einer SAPUI5-App hinzufügen 534
14.1.4 Verwendungsbeispiele 535
14.1.5 So bleiben Sie auf dem Laufenden 547

14.2 UI5 Web Components 547
14.2.1 Aktueller Status 548
14.2.2 Clean Code mit Webkomponenten 550
14.2.3 Verwendungsbeispiele 551
14.2.4 So bleiben Sie auf dem neuesten Stand 556

14.3 Fundamental Library 556
14.3.1 Fundamental Library Styles 557
14.3.2 Fundamental Library für Angular 557
14.3.3 Aktueller Status 559
14.3.4 Wie Sie auf dem neuesten Stand bleiben 560

14.4 Zusammenfassung 560

15 Wie Sie Clean SAPUI5 umsetzen 561

15.1 Gemeinsames Verständnis der Teammitglieder 562

15.2 Kollektive Code Ownership 562

15.3 Clean Code Developer Initiative 564

15.4 Den Broken-Window-Effekt angehen 566
15.4.1 Statische Codeprüfung 569
15.4.2 Metriken 569
15.4.3 Codeabdeckung 570

15.5 Code-Reviews und Lernen 570
15.5.1 Code-Review-Präfix 570
15.5.2 Styleguide 570
15.5.3 Praktiken sichtbar machen 571
15.5.4 Feedback-Kultur 571

15.6 Clean Code Advisor 574

15.7 Lerntechniken ... 574
15.7.1 Kata ... 575
15.7.2 Dojo ... 576
15.7.3 Code-Retreat ... 576
15.7.4 Fellowship ... 577
15.7.5 Pair Programming ... 577
15.7.6 Mob Programming ... 578
15.7.7 Gewohnheiten ... 578
15.8 Continuous Learning in funktionsübergreifenden Teams ... 579
15.8.1 Profil eines Teammitglieds ... 580
15.8.2 Funktionsübergreifende Teams ... 581
15.8.3 Multiplikatoren im Team ... 581
15.8.4 Community of Practice ... 582
15.9 Zusammenfassung ... 582

Die Autoren ... 583
Index ... 585