

Inhalt

Materialien zum Buch	17
Geleitwort	19
Vorwort	21

1 Im Web, als App: Geschichte und Einstieg 25

1.1 Wie Apps auf unser Handy kamen – eine kleine Zeitreise	27
1.1.1 Der PC für die Hosentasche	28
1.1.2 Der Browser als Anwendungsplattform	28
1.1.3 Es gibt eine App dafür: der Siegeszug des App Store	30
1.1.4 Das Verhältnis von Web und Apps	34
1.2 Progressive Web Apps:	
wohin die Reise der Anwendungsentwicklung geht	35
1.2.1 Die Plattformproblematik	37
1.2.2 Das Web ist die Plattform	38
1.2.3 Das Rückgrat der Progressive Web Apps	39
1.2.4 Progressive Web Apps als Über-Pattern	41
1.3 Voraussetzungen und Basics:	
welches Wissen Sie schon mitbringen sollten	42
1.3.1 HTML, CSS, JavaScript und TypeScript	42
1.3.2 Single-Page Web Applications	43
1.3.3 Promises: asynchrone Operationen im Griff	45
1.3.4 Mobile Anwendungsentwicklung	47
1.4 Tools installieren: das Handwerkszeug zur PWA-Entwicklung	48
1.4.1 Visual Studio Code	48
1.4.2 Git	50
1.4.3 Google Chrome	51
1.4.4 Node.js	51
1.4.5 Angular CLI	53
1.4.6 Workbox CLI	53
1.4.7 ngrok	54
1.5 Setup: Das erste PWA-Projekt	54
1.5.1 Dateien anlegen	55
1.5.2 »index.html« – der Einsprungspunkt für Ihre PWA	56
1.5.3 »sw.js« – das Service-Worker-Skript	58

1.5.4	Website mit lite-server ausprobieren	61
1.5.5	Website mit ngrok auf einem Mobilgerät testen	63
1.6	Zusammenfassung	65

2 Mächtiges modernes Web 67

2.1	Audio-/Videoelement: Multimediainhalte ohne Plug-in wiedergeben	70
2.2	Canvas-Element: ansprechende 2D- und 3D-Visualisierungen	72
2.2.1	Paper.js	72
2.2.2	THREE.js	74
2.3	Gamepad API: App mit dem Game-Controller steuern	78
2.4	WebAssembly: Binärcode für das Web mit nahezu nativer Performance	82
2.5	Web Share API: Teilen von Inhalten aus dem Browser heraus	84
2.6	Web Speech API: Text-to-Speech im Browser	87
2.7	Media Capture and Streams: auf Kamera und Mikrofon zugreifen	88
2.8	Generic Sensor API: Zugriff auf die Gerätesensoren	91
2.9	Pointer Events und Pressure.js: Force Touch im Web	93
2.10	Geolocation: Implementierung standortbezogener Dienste	94
2.11	Zusammenfassung	98

3 Zehn Eigenschaften, die PWA einzigartig machen 99

3.1	Voranschreiten mit Progressive Enhancement	100
3.2	App-ähnlich: Sieht aus wie eine App, fühlt sich an wie eine App	103
3.3	Verbindungsunabhängigkeit: Kein Funkloch hält Sie auf	106
3.4	Immer schön frisch bleiben: der Service-Worker-Updateprozess	109
3.5	Sicher: Mit großer Macht kommt große Verantwortung	110
3.5.1	Sicherheitsmaßnahmen im Webbrowser	110
3.5.2	Sichere Verbindungen mit HTTPS	112
3.6	Einer für alle: Responsive Webdesign	115

3.7	Auffindbarkeit: Websites und Apps unterscheiden	118
3.8	Installierbarkeit: So kommt Ihre PWA auf den Homebildschirm	120
3.9	Nutzer binden: Anwender mit Pushbenachrichtigungen zurückholen	123
3.10	Verlinkbar: auf Anwendungen und Zustände verweisen	126
3.10.1	Hash-basiertes Routing	127
3.10.2	Pfad-basiertes Routing	128
3.10.3	Deep Linking und Link Capturing	129
3.11	Zusammenfassung	129

4 Web App Manifest: Aussehen der App definieren 131

4.1	App-Aussehen auf dem Homebildschirm anpassen	135
4.1.1	Name	135
4.1.2	Icons	136
4.1.3	Farben	142
4.1.4	Splashscreen	143
4.1.5	Weitere Metadaten	146
4.2	App-Verhalten anpassen	146
4.2.1	Start-URL	146
4.2.2	Scope	147
4.2.3	Anzeigemodi	148
4.2.4	Bildschirmausrichtung	153
4.2.5	Verwandte Apps	154
4.2.6	Service Worker	155
4.2.7	Alterskennzeichnung	155
4.3	Web App Manifest referenzieren	156
4.4	Zur Installation auffordern	157
4.4.1	Auslaufmodell App-Install-Banner	159
4.4.2	Eigene Methode	160
4.4.3	Manuelle Methode	163
4.4.4	Deinstallation	165
4.5	Zukunftsmusik: Badging API	166
4.6	Microsoft Store Ingestion: Wie die App den Weg in den Microsoft Store findet	168

4.7	PWA Builder	170
4.8	Zusammenfassung	170
5	Service Worker: Einer muss ja arbeiten	171
5.1	Vom Web Worker zum Service Worker	171
5.1.1	Web Worker	172
5.1.2	Shared Worker	173
5.1.3	Service Worker	174
5.2	Kontrollzwang mit Vorteilen: Service Worker als zentraler Proxy	176
5.2.1	Wo lebt der Service Worker in meinem Browser?	178
5.2.2	Erst mal offline: das Offline-First-Paradigma	178
5.2.3	IndexedDB: Wer noch alles im Offlineteam mitspielt	180
5.3	Lebenszyklus	188
5.3.1	Registrieren	189
5.3.2	Installieren	194
5.3.3	Aktivieren	197
5.3.4	Überschreiben	199
5.4	Schnittstellen	199
5.4.1	Netzanfragen abfangen und manipulieren	200
5.4.2	Briefchen schreiben mit postMessage	202
5.4.3	Mit Clients interagieren	205
5.5	Wir alle machen Fehler: Service Worker debuggen	207
5.5.1	Google Chrome	207
5.5.2	Mozilla Firefox	213
5.5.3	Apple Safari	214
5.5.4	Microsoft Edge	216
5.6	Background Sync API	217
5.6.1	Progressive Enhancement berücksichtigen	218
5.6.2	Registrieren eines Synchronisationsereignisses	219
5.6.3	Hintergrundsynchronisation ausführen	220
5.7	Navigation Preload	221
5.8	Zusammenfassung	223

6	Cache API: So lädt die App auch ohne Netzverbindung	225
6.1	HTTP-Crashkurs: So war das noch mal mit Anfragen und Antworten	226
6.1.1	Auslaufmodell XMLHttpRequest	229
6.1.2	Fetch API	231
6.2	Auslaufmodell Application Cache	235
6.3	Caches verwalten	236
6.3.1	Caches benennen	236
6.3.2	Caches öffnen	237
6.3.3	Caches auflisten und löschen	237
6.4	Antworten zur Seite legen:	
	Man weiß nie, wann man sie wieder gebrauchen kann	238
6.4.1	Abfrage zwischenspeichern	238
6.4.2	Kombination mit dem Service Worker: install und addAll	241
6.4.3	Speicherkontingent abfragen	243
6.4.4	Websitespeicher persistent machen	245
6.5	It's a match! – Passende Antworten aus dem Cache holen	246
6.5.1	Einzelantwort beziehen	246
6.5.2	Kombination mit dem Service Worker: fetch und match	247
6.5.3	Mehrere Antworten beziehen	249
6.5.4	Alle Zwischenspeicher durchsuchen	250
6.6	Cacheeinträge löschen	250
6.7	Caches debuggen	251
6.8	Caching-Strategien	252
6.8.1	Cache Only	253
6.8.2	Network Only	254
6.8.3	Cache falling back to network	254
6.8.4	Generic Fallback	255
6.8.5	Network falling back to cache	256
6.8.6	Cache then network	257
6.8.7	Cache & network race	258
6.8.8	Service-Worker-Side Templating	259
6.9	Weitere Service-Worker-Use-Cases	259
6.10	Zusammenfassung	260

7 Workbox 261

7.1	Befehle der Workbox CLI	262
7.2	Workbox-Projekt aufsetzen	263
7.3	Precaching	265
7.4	Runtime Caching	270
7.4.1	Routen und Caching-Strategien	271
7.4.2	Konfiguration über runtimeCaching	271
7.4.3	Plug-ins	274
7.4.4	Navigations-Fallbacks	275
7.5	Service Worker erweitern	276
7.5.1	Precaching	276
7.5.2	Runtime-Caching	279
7.6	Workbox in den Buildprozess integrieren	279
7.6.1	Node.js und Gulp	280
7.6.2	Webpack	281
7.7	Navigation Preload aktivieren	282
7.8	Offlineanalysedaten erfassen	282
7.9	Zusammenfassung	283

8 Push API: Rufen Sie nicht uns an – wir rufen Sie an! 285

8.1	Das Push-Prinzip	286
8.2	Nur eine Chance: Pushregistrierung beantragen	287
8.2.1	Progressive Enhancement berücksichtigen	288
8.2.2	Pushregistrierung auslösen	288
8.2.3	Einwilligung des Anwenders einholen	291
8.3	Informationsaustausch	293
8.4	Den Server Pushnachrichten verschicken lassen	297
8.5	Pushereignisse behandeln	302
8.5.1	Pushereignisse entgegennehmen	303
8.5.2	Benachrichtigungsbanner anzeigen	303
8.5.3	Auf die Auswahl des Anwenders reagieren	307
8.6	Sonderfall Apple	310

8.7	Drittanbieterdienste: OneSignal & Co.	314
8.7.1	OneSignal	314
8.7.2	PushCrew	315
8.7.3	Pushpad	315
8.8	Pushnachrichten zur Laufzeit	316
8.8.1	socket.io	316
8.8.2	ASP.NET Core SignalR	317
8.8.3	Anzeige von Benachrichtigungen	318
8.9	Zusammenfassung	319

9 PWA und Angular: Single-Page-Application-Framework einsetzen 321

9.1	Projekt-Setup	323
9.1.1	Angular CLI	323
9.1.2	Neues Projekt anlegen	324
9.2	Responsive und App-like: Navigationsgrundgerüst mit Angular Material	325
9.2.1	Angular Material installieren	326
9.2.2	App Shell implementieren	327
9.3	Linkable: Routing implementieren	329
9.3.1	Komponenten generieren	329
9.3.2	Routen anpassen	330
9.3.3	Navigationsleiste anpassen	332
9.4	App-Like, die zweite: Moderne Web-APIs einsetzen	333
9.5	PWA-Unterstützung installieren	335
9.6	Discoverable: Web App Manifest anpassen	336
9.7	Connectivity Independent, die erste: Quelldateien der Anwendung offlinefähig machen	338
9.7.1	Service Worker registrieren	339
9.7.2	Caching des Angular Service Workers konfigurieren	339
9.7.3	Produktionsbuild ausführen	344
9.7.4	Debugging	346
9.7.5	Grenzen des Angular Service Workers	348
9.7.6	Service Worker entfernen	348

9.8	Connectivity Independent, die zweite: strukturierte Daten zwischenspeichern	348
9.8.1	Modellklasse generieren	349
9.8.2	Service anlegen	349
9.8.3	To-do-Einträge hinzufügen	350
9.8.4	To-do-Einträge abrufen	353
9.8.5	To-do-Einträge als erledigt markieren	354
9.8.6	To-do-Einträge synchronisieren	356
9.9	Reengageable: Pushereignisse mit SwPush	359
9.10	Fresh: Updateprozess mit SwUpdate	364
9.11	Installable: Installation anbieten	366
9.12	Angular Universal: mit Server-Side-Rendering zur App Shell	368
9.13	PRPL-Entwurfsmuster	369
9.14	Zusammenfassung	370

10 App-like aussehen 373

10.1	Native Schriftarten einsetzen	373
10.2	Textauswahl und Link-Highlighting verhindern	375
10.2.1	Standardcursor setzen	375
10.2.2	Textauswahl verhindern	376
10.2.3	Link-Highlighting verhindern	377
10.2.4	Keine Callouts	378
10.3	App-like Anwendungsframeworks	379
10.3.1	ngx-admin	379
10.3.2	Angular Material	380
10.3.3	Framework7	380
10.4	Notches unterstützen	381
10.5	Zusammenfassung	386

11 Plattformverhalten 387

11.1	macOS	387
11.2	iOS	389

11.3	Android	391
11.4	Windows	395
11.5	Linux	397
11.6	Zusammenfassung	399

12 Alles richtig gemacht? – PWAs validieren mit Lighthouse & Co. 401

12.1	Barrierefreiheit testen mit aXe	402
12.2	Lighthouse: der Leuchtturm der Websitevalidierung	405
12.2.1	Performance	407
12.2.2	Progressive Web App	409
12.2.3	Best Practices	410
12.2.4	Accessibility	410
12.2.5	Search-Engine Optimization	410
12.3	webhint: ein Linter für das Web	411
12.3.1	Accessibility	414
12.3.2	Interoperability	414
12.3.3	Performance	415
12.3.4	Progressive Web App	415
12.3.5	Security	416
12.4	Zusammenfassung	416

13 Migrationsstrategien mit Apache Cordova und GitHub Electron 417

13.1	Apache Cordova	419
13.1.1	Projekt anlegen und konfigurieren	420
13.1.2	Plattformen	423
13.1.3	Ereignisse	427
13.1.4	Plug-ins	429
13.1.5	Anwendung bauen	437
13.1.6	Ionic	438
13.2	GitHub Electron	440
13.2.1	Projektkonfiguration	441

13.2.2	Ereignisse	443
13.2.3	Mit nativen Schnittstellen interagieren	448
13.2.4	Electron-Packager	453
13.3	Plattformunterschiede elegant verbergen	455
13.4	Chromium Embedded Framework: Desktopanwendungen schrittweise entkernen	459
13.5	Zusammenfassung und Fazit	460

14 Payment Request API: Wie Sie trotz fehlendem App Store an Ihr Geld kommen 463

14.1	Warum Check-out-Formulare nicht die Lösung sind	464
14.2	Einfaches Check-out mit der Payment Request API	466
14.2.1	Zahlungsmethode	468
14.2.2	Zahlungsdetails	468
14.2.3	Optionen	470
14.3	Ablauf einer Zahlungsanforderung	470
14.3.1	Prüfung und Anzeige der Zahlungsanforderung	471
14.3.2	Auf Änderungen reagieren	472
14.3.3	Abschluss des Verkaufs	473
14.4	Payment Method: Basic Card	474
14.5	Google Pay	478
14.6	Apple Pay	482
14.6.1	Konfiguration des Zahlungsmethodenobjekts	483
14.6.2	Extraschritt: Validierung des Händlers	485
14.6.3	Antwortobjekt auswerten	486
14.7	Fazit: Viele Wege führen zum Fallback	489
14.8	Ausblick: Payment Handler API	491
14.9	Zusammenfassung	492

15 BrandheiÙe Progressive Web Apps 495

15.1	Twitter Lite	496
15.2	Financial Times	497

15.3	Telegram	498
15.4	Pokédex	499
15.5	QR Scanner	500
15.6	Zusammenfassung	502

16 Fazit: Eine Codebasis, alle Plattformen 503

16.1	Ideales technologisches Umfeld	503
16.2	Interessen der Plattformhersteller	504
16.3	Wer heute schon PWAs baut	506
16.4	Limitationen	506
16.5	Chancen	507
16.6	Ausblick	508

Über den Autor	511
Index	513