PWAs mit Angular

Jens Binfet

#1 Herausforderung im Engineering von Benutzeroberflächen:

In welchem Kontext wird die Applikation ausgeführt werden?

Web all the things?

Das Web als Plattform verspricht, diese Probleme zu lösen.

- Eine einzige Plattformabstraktion
- One code to rule them all
- Lauffähig auf so gut wie jeder Hardware
- Keine Softwareverteilungs / Updateproblematik
- ...

Aber - Funktioniert das auch in der Praxis?

Wellenbewegung des Web-Enthusiasmus

- Web 2.0
- Ajax-Ära
- Responsive Design
- Single-Page Applikationen mit JavaScript
- Progressive Web Apps

➔ Jede Wellenbewegung hat in ihrem teilweisen “Scheitern” das Web als Plattform enorm vorangebracht

Zusätzliche Anforderungen aus aktueller Sicht

- Internationalisierung / Lokalisierung
- Echtzeitkommunikation
- Offline-Fähigkeit
- Time to Interaction
- Notifications (Push)
- ...

Ein aktueller Lösungsansatz: PWA

Progressive Web-Apps sind Webapplikationen (HTML/CSS/JS), die zahlreiche Charakteristika einer *nativen App* mit den Vorteilen einer *Webapplikation* kombinieren.

➔ “Das Beste aus beiden Welten”

Charakteristika

- Single Codebase (Plattform & Deviceunabhängigkeit)
- Progressive Feature Enhancement
- Offline Funktionalität (Offline-First)
- Keine Installation
- Add-To-Homescreen (scheinbare Installation)
- Push-Notification
- URL
- Niedrige Time-to-Interaction

Wie hängen Angular und PWA zusammen?

PWA

Was ist Angular?

"Angular is a **Platform** that makes it easy to build applications with the **web** (PWA)"

Stephen Fluin - Developer Advocate Google

Was macht Angular so hilfreich?

Angular ist nicht nur ein Framework, sondern eine Plattform um das Framework!

Das bedeutet:

Angular Plattform

=

User-Experience + Developer Experience + Community

(Features) + (Entwicklungsprozess) + (Ökosystem)

Ist Angular das beste Framework?

Angular ist sicher nicht für jeden Anwendungsfall das beste Framework.

Es gibt da draußen jede Menge technisch exzellente Frameworks.

Angular hat aus meiner Sicht das Gesamtkonzept (Plattform) mit dem größten allgemeinen Mehrwert!

User Experience / Angular Kern Features

Ein selektiver Überblick

Single Page Pattern

- Keine Pagereloads für den Benutzer
- “App-Feeling”
- Nach dem initialen load, nur noch reiner Datenaustausch
- Funktioniert exzellent mit REST Datenquellen

Databinding

- Die Darstellung der Daten reagiert umgehend auf Änderung der Daten
- Angulars Observable Pattern (RxJs) gibt viel Gestaltungsfreiraum für Datenflüsse und Transformationen ohne Performanceeinbußen

Routing

- Das Routing geschieht im Client (Browser)
- Dadurch:
  - Bookmarkbare “Seiten” / Deeplinks
  - Lazy Loading ganzer Module
  - Daten-Preloading
  - “Guards”

Webcomponents

- W3C Standard
- UI-Elemente können als selbstdefinierte “HTML”-Elemente wiederverwendet werden
  - gekapselte Styles (Keine Seiteneffekte)
  - Verhalten des Elements (JS)
  - Struktur/Markup
- Diverse CSS-Frameworks bieten auf diese Weise Integrationen an (Bootstrap, Material etc.)

Weitere Features

- Internationalisierung und Lokalisierung
- Accessibility
- native Browser Animationen
- Formularunterstützung (Validierung, Multistep etc.)
- Http-Library
- Modulsystem (einfach konsumierbares Ökosystem)

Developer Experience

TypeScript

- Vorteile typisierter Sprachen, ohne die Flexibilität von JavaScript zu verlieren
- Implizite Lösung des Feature-Gap Problems (Implementation ist unabhängig vom Unterstützungslevel in den anvisierten Browsern)
- Einfachere Zusammenarbeit auch mit unerfahrenen Kollegen
- Einfacherer Umstieg aus einer streng typisierten Sprache

IDE Unterstützung (Languageserver)

- Durch den Language Server von Angular kann die IDE Angular Code “verstehen”
- Falsche Bindings werden vor der Laufzeit erkannt
- Code Completion selbst in Templates

CLI - Commandline Tool

- Das Commandline Tool ist das Herz der DX-Strategie
- Es bündelt Tools und Best-Practices für den gesamten Dev-Lifecycle

Scaffold & Generate

- Setup - Generierung und Konfiguration einer Initialen App
  - Wir starten mit einer lauffähigen, präkonfigurierten App
  - Durch flags können Features direkt integriert werden
    z.B. git Initialisierung, Style Präprozessor (SASS)
- Generate - Entitäten (Module, Komponenten etc.) inklusive Tests generieren und integrieren

Serve & Develop

- Serve
  - Devbuild & Server (inkl. Livereload etc.)
  - CodeStyle (Codelyzer/Linting)
- i18n
- Direkter Doku Zugriff/Suche via CLI
- ...

Build & Distribute

- Build
  - Vorkonfigurierter Dev & Prod Buildtask & Profil
  - Eigene Build Konfigurationen
  - ENV-Variablen etc.

Test

- Test
  - Setup für Unitests (Karma)
  - End-to-End Tests (Protractor)
- Testabdeckungs Bericht
- Unittests werden bei der Generierung mit generiert (TDD / BDD ready)

Erweitertes Tooling

- Augury (Chrome Ext) - Deep App Inspection
- NG Universal Tools für PWA Optimierung:
  - Service Worker
  - App Shell
  - Push Notifications
- ...

Und das Backend? - Fullstack Generierung

- jHipster
- NinjaCodeGen
- Celerio Angular Quickstart (Spring Boot + rel. DB)
- ...

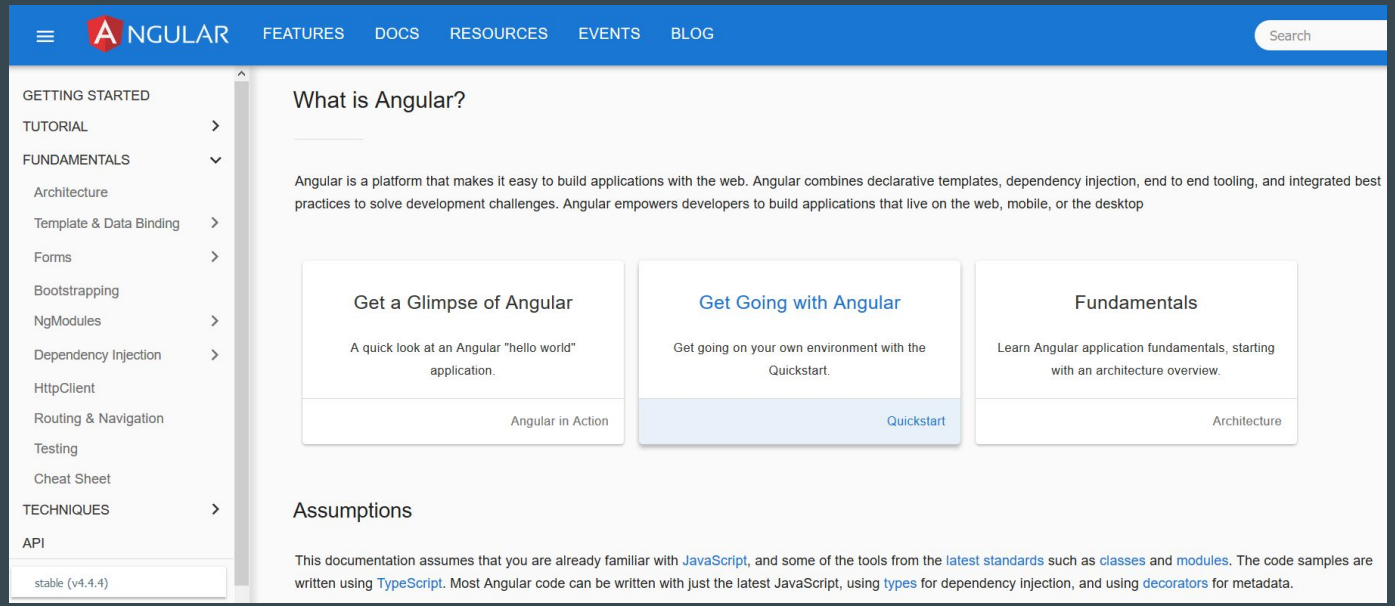
Community

Kern Projekt - Ein Community Projekt?

- 12.000+ OpenSource Contributors im Projekt (2016)
- Offene Entwicklung auf Github
- Google veröffenlicht massiv Videos aber auch ein ständig aktualisiertes Tutorial und Themenartikel z.B. zu Testing, Security oder Internationalisierung
- Google investiert zwar am meisten Entwickler Ressourcen und hat auch die Projektleitung, pflegt aber einen sehr integrativen Stil (z.B. dedizierte Communitymanager)

Dokumentation

Angular hat eine herausragende Dokumentation (genauso wie übrigens TypeScript)



Ökosystem

- Durch die Community und die Offenheit des Projektes gibt es eine Menge an hochqualitativen Dritt Modulen
  - in i18n kein xliff sondern json nutzen?
  - Statt Material Design Twitter Bootstrap nutzen?
  - Ich brauche eine komplexe Datatable (Filter, Sort, Aggregate)?
  - Integration in Firebase, GraphQL etc.?
  - JWT, Basic oder Session Authentication?

➔ Kein Problem, es gibt bereits eine Komponente/Modul dafür

Und was wenn der Browser nicht genug ist?

- Angular Universal
- Electron
- NativeScript
- Ionic
- React Native
- Windows UWP
- ...

Fazit

Größere Projekte / Enterprise mit Angular?

- Gerade die Plattform Angular bietet ein exzellentes Umfeld
- Skalierbarkeit ist durch TypeScript und Tooling gegeben
- Gerade heterogene Teams profitieren von der Doku und den Best Practice “Trampelpfaden”
- Angular ist ein etabliertes Projekt, maintained von einer Firma, die selbst massiv auf die Technologie setzt
- Strategische Technologieentscheidung: Sehr vielseitig einsetzbar und damit eine gute Aufwand-Nutzen Relation

Ist Angular für jedes Projekt die richtige Wahl?

Natürlich nicht!

- Hochspezialisierte Projekte
- Starker Bezug zu (Spezial) Hardware (z.B. Messgeräte)
- Grafikintensive Applikationen (Spiele, Konstruktionssimulationen ...)
- ...

Für welche Projekte ist Angular gut geeignet?

- Formularbasierte Anwendungen
- Anwendungen in einem Microservice Kontext
- Anwendungen mit Multiplattform Anspruch
- Visualisierungen & Datenauswertung
- Dashboards, Cockpits etc. (aggregative Applikationen)
- ...

Welche Mehrwerte habe ich?

Gegenüber klassischer nativer oder klassischer Webentwicklung:

- Das Beste aus beiden Welten
- Einen extrem agilen Entwicklungsprozess (Scrum, TDD ...)
- Klar definierte Trennung zwischen FE & Server/DB -> REST
- Ein standardkonformes “mitwachsendes” Framework
- Sehr starke Annäherung zu “One Code to rule them all”
- Starker Fokus auf UX & Accessibility (B2C / Behörden)
- Starke Unterstützung für Code Quality & Testabdeckung

Demo / Fragen & Diskussion