Heiko Spindler

Single-Page-Web-Apps

JavaScript im Einsatz: Webseiten erstellen mit AngularJS, Meteor und jQuery Mobile

FRANZIS

Heiko Spindler

Single-Page-Web-Apps

Heiko Spindler

Single-Page-Web-Apps

JavaScript im Einsatz: Webseiten erstellen mit AngularJS, Meteor und jQuery Mobile

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Programmleiter: Dr. Markus Stäuble
Lektorat: Christian Immler
Satz: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: CPI, Printed in Germany

ISBN 978-3-645-60310-2

Vorwort

Für wen ist das Buch gedacht?

Das Buch richtet sich an alle, die Interesse an einer neuen Art von Web-Entwicklung haben und den Nutzern mehr Dynamik bieten wollen. Um sinnvolle Single-Page-Web-Applikationen (SPWA) zu bauen, werden wir das Rad nicht neu erfinden. Die Zeiten, in denen man alles von Hand machen musste, sind für die Web-Entwicklung zum Glück vorbei. Der Hauptfokus des Buches liegt auf der Auswahl und dem Einsatz der richtigen Frameworks, für die Implementierung von SPWA. Dabei steht die Pragmatik und Einfachheit im Vordergrund. Gleichzeitig gibt das Buch mit vielen Beispielen Hilfestellung für die Entwicklung von unternehmenskritischen Applikationen mit JavaScript.

Erste Grundkenntnisse im Umgang mit HTML5, CSS 3 und einer Programmiersprache sind sehr sinnvoll und werden vorausgesetzt. Erfahrungen in der Entwicklung von JavaScript sind hilfreich, aber kein Muss.

Wie ist das Buch aufgebaut?

Das Buch gliedert sich in vier Hauptteile:

1. Einleitung in das Thema:
   Die Einleitung führt in das Thema ein und erklärt die grundlegenden Konzepte und Herausforderungen beim Erstellen von Web-Applikationen.

2. JavaScript für Entwickler:
   Das zweite Kapitel enthält eine Einführung in die professionelle Programmierung mit JavaScript. Dabei werden vor allem die Unterschiede zu verbreiteten Programmiersprachen, wie zum Beispiel Java, aufgezeigt. Hilfreich sind erste Erfahrungen in der Programmierung.

3. Single-Page-Web-Apps erstellen:
   Der Hauptteil zeigt Schritt für Schritt, wie dynamische Web-Applikationen mit AngularJS erstellt werden. Jedes Beispiel hebt dabei einzelne Aspekte hervor. Besonderes Augenmerk liegt auf dem effizienten Einsatz und Zusammenspiel von aktuellen Frameworks. Zusätzlich vermittelt der Abschnitt viele Tipps und Tricks für die tägliche Arbeit.

4. Von der DB bis ins Web mit Meteor:
   Die Programmiersprache JavaScript eignet sich auch für die Implementierung von Server-Komponenten. Meteor stellt ein durchdachtes Applikationsframework bereit, das mit herausragenden Eigenschaften überrascht: Datenänderungen werden automatisch an alle Clients verteilt. Weil viele Funktionen und Module vorgefertigt sind, kann man sich sich ganz auf das Wesentliche konzentrieren. Mehrere nachvollziehbare Beispiele erklären den Einstieg in die Entwicklung mit Meteor.

5 Mobile Single-Page-Web-Apps:
Mobile Geräte stellen einen immer größeren Markt im Web dar, wobei die teilweise großen Unterschiede, zum Beispiel bei den Displays oder der Bedienung, besondere Anforderungen stellen. Das fünfte Kapitel zeigt, wie sich spezielle Web-Applikationen für diese Geräteklasse mit jQuery-Mobile erstellen lassen.

Downloads zum Buch
Alle im Buch erwähnten Quellcodes finden Sie auf *www.buch.cd* zum Download.

Ich bedanke mich bei meiner Familie für die Unterstützung. Ein besonderer Dank geht an Niko Köbler.

Inhaltsverzeichnis

Einleitung

1.1 Das Web als Plattform

In den letzten Jahren hat sich das Internet als die umfassende Kommunikationsplattform etabliert und bildet ebenfalls die Basis für den Erfolg mobiler Geräte. Es entstehen ständig neue Technologien. Ganze Branchen müssen sich in dieser »schönen neuen Welt« umorientieren und tragfähige Geschäftsmodelle finden. Die Software-Entwicklung im Zentrum dieser Veränderung muss neue Trends erkennen und die eigenen Kompetenzen immer wieder an diese Entwicklungen anpassen. Die Dynamik, die sich manchmal aus kleinen Neuerungen ergibt, ist erstaunlich. Ein neuer Begriff am Horizont ist das Paradigma von Single-Page-Web-Applikationen (SPWA): eine neue Art, Web-Anwendungen zu bauen.

1.2 Was ist eine Single-Page-Web-Applikation?

Eine Single-Page-Web-Applikation ist eine Web-Anwendung, die keinen Seitenwechsel (Refresh) durchführt, sondern die Oberfläche über dynamischen Austausch der HTML-Elemente mit JavaScript ändert.

Bei Aktionen in der Anwendung entfällt das kurze Flackern, das sonst die Anfrage beim Webserver und beim Neuaufbau begleitet. Das scheint auf den ersten Blick nicht

besonders spektakulär. Einem Benutzer fällt diese kurze Wartezeit vom Drücken eines Bestell-Knopfes bis zum Aufbau des Warenkorbes auf einer Shopping-Seite aber eventuell unangenehm auf.

Schon eine Verzögerung von mehr als zwei Sekunden wird oft als störend empfunden. Bei herkömmlichen Webseiten wird die gesamte Seite (HTML-Code, Bilder, Stylesheet und Scriptdateien) übertragen. Natürlich strengen sich die modernen Browser an und merken sich Elemente, die sich nicht geändert haben. Zumindest der HTML-Code einer Seite muss neu geladen werden. Diese Übertragung erfordert eine gewisse Zeit: Verbindungsaufbau, Übertragung und Darstellung summieren sich schnell zu Sekunden.

Was eine SPWA auszeichnet, in Kürze:

- Die Implementierung erfolgt mit den Technologien HTML5, CSS 3 und JavaScript.
- Die HTML-Seiten werden zum größten Teil dynamisch im Browser erzeugt.
- Der Datenaustausch mit einem Backend erfolgt meist in Form von JSON oder XML.
- Die Webseite verhält sich mehr wie eine Applikation.
- Es gibt keinen Reload oder Seitenwechsel.
- Der aktuelle Zustand der Applikation wird im Browser gehalten.
- Neben den klassischen Eingabe-Elementen bieten diese Applikationen meist erweiterte, aktivere Benutzeroberflächen an.

Interessant ist, dass HTML, JavaScript und CSS nicht neu sind, sondern schon lange verfügbar. JavaScript erhält einen deutlich höheren Stellenwert als bisher.

1.2.1 Die Vorteile von Single-Page-Web-Applikationen

Welche Vorteile entstehen aus diesem neuen Ansatz? Eines der Hauptziele vieler Anbieter ist es, Applikationen mit mehr Logik als bisher zu erstellen:

- Integration verschiedener Informationsquellen im Client
- Automatische Berechnungen und frühzeitige Validierung von Benutzereingaben
- Flexible Anpassung auf individuelle Benutzerwünsche und Vorlieben
- Mehr Interaktivität und Dynamik bei der Bedienung
- Aktive und automatische Benachrichtigung beim Auftreten von Ereignissen oder Änderungen an Daten

Bessere Verteilung

Eine SPWA ist über eine URL im Browser universell erreichbar. Eine Installation ist nicht notwendig. Für den Privatgebrauch ist das nur ein nettes Feature. Im Unterneh-

mensumfeld reduziert diese Eigenschaft die Kosten für Installation und Verteilung. Damit einher geht auch die Möglichkeit, leicht mehrere Versionen parallel zu betreiben. Leicht können neue Funktionen während einer Testphase ausprobiert werden. Eine andere URL für die neue Version ist schnell per E-Mail kommuniziert.

Die einheitliche Plattform

Eine heutige Applikation soll möglichst universell verfügbar sein. Viele Benutzer wollen Dienste auf unterschiedlichen Geräten nutzen. Der Zugriff muss vom heimischen PC genauso gut funktionieren wie vom Tablet oder Smartphone aus. Eine separate Entwicklung für jede Zielplattform ist teuer. Das Web wird die übergreifende Plattform für alle Betriebssysteme und Gerätearten.

Laut einer Studie von Cisco Systems (*bit.ly/19jj1Sw*) waren schon 2010 über 12,5 Milliarden Geräte mit dem Internet verbunden. Der größte Teil ist mit Browsern und JavaScript ausgestattet. Wenn man als Anbieter eine große Zielgruppe erreichen möchte, muss man auf diese Technologien setzen.

Einige Smartphone-Hersteller entwickeln neue mobile Betriebssysteme, die auf eine reine HTML-Oberfläche setzen. Alle Apps sind damit formal Web-Anwendungen. Vertreter dieser Gattung sind Tizen von Samsung und Firefox OS, hinter dem die Mozilla-Organisation steht. Offensichtlich sind das nicht nur Experimente, erste Geräte sind auf dem Markt verfügbar.

Reduzierung der Technologien

Die oben beschriebene Reduzierung auf die am meisten verbreiteten Technologien hat weitere positive Aspekte auf Seiten der Software-Entwicklung: weniger Know-how muss für die Entwicklung und Wartung bereit gehalten werden. Sicherlich muss in die effiziente Entwicklung mit dem neuen Paradigma einmalig investiert werden, langfristig reduziert es erheblich die Kosten.

Integration über REST-Schnittstellen

SPWA bieten sich gut als Integrationspunkt an. Eine Backend-Komponente kann leicht per REST-Service angebunden werden. Als Datenformat haben sich JSON (JavaScript Object Notation) und XML durchgesetzt. Der Client kann unterschiedliche Datenquellen zusammenführen und in der Applikation integrieren und präsentieren.

Offline-Fähigkeiten

Mit den neuen Fähigkeiten von HTML5 wie dem LocalStorage gibt es zum ersten Mal eine wirkliche Möglichkeit, effiziente Cache-Strategien und Offline-Fähigkeiten zu etablieren.

Geringe Einstiegshürden: Start Easy

Es ist sehr leicht, mit den beschriebenen Technologien zu starten. Als Entwicklungsumgebung reicht ein guter Texteditor aus. Manche bieten sogar Syntax-Hervorhebung für HTML und JavaScript an. Zum Ausführen reicht ein Browser, der mit den entspre-

chenden Plug-ins sogar Debugging bereitstellt. Alle diese Komponenten sind für den Einsteiger kostenlos verfügbar. Kostenpflichtige Entwicklungsumgebungen bieten eventuell mehr Komfort und einige Arbeitserleichterungen, sind aber keine Voraussetzung.

Das Grundwissen über HTML und JavaScript ist bei vielen Entwicklern und Designern vorhanden. Man muss sicherlich tiefer in die Details eintauchen, wenn man komplexe Anwendungen entwickeln möchte.

1.3 Ein Blick zurück in die Historie

Die Ursprünge von HTML reichen zurück bis zum Ende des letzten Jahrhunderts. So gesehen handelt es sich um keine neue Technologie. Der ursprüngliche Fokus bei der Entstehung war, statische Inhalte in Form von wissenschaftlichen Texten und Bildern zu veröffentlichen. Über die Verlinkung sollten thematisch ähnliche Quellen leicht erreichbar sein.

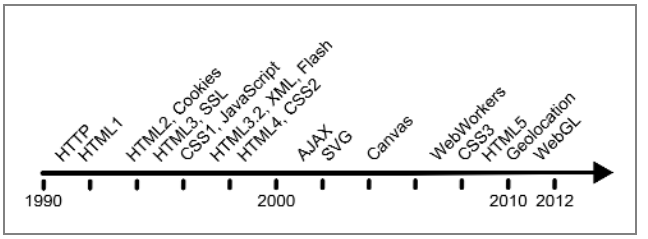

Bild 1.1: Zeitliche Einordnung der Entstehung einiger grundlegenden Technologien für Web-Applikationen.

Das Erstellen von Applikationen war nicht das Ziel. Dieser Umstand drückt sich auch in dem zustandslosen http-Protokoll aus. Jede Anfrage ist isoliert und eine Applikation muss sich selbst um den logischen Zusammenhang kümmern, meist mit Hilfe von Session-Kennzeichen oder Cookies.

Eine Lösungsstrategie für den Bau von Applikationen hieß: Halte alle Informationen auf dem Server und behandle den Browser wie eine Darstellungseinheit für statische HTML-Inhalte. Die eigentliche Applikation läuft auf dem Server. Er verwaltet den Zustand und reagiert auf Benutzeraktionen. Als Ergebnis erhält der Benutzer eine neu generierte HTML-Seite.

Tipp: Evolution of the Web
Eine ansprechende und interaktive Darstellung der Historie bietet die Seite »Evolution of the Web« unter der Adresse *www.evolutionoftheweb.com*. Die Seite dokumentiert ebenfalls alle gängigen Browser, die zeitliche Abfolge der Versionen und die implementierten Standards.

Diese Architektur funktioniert weiterhin sehr gut. Viele professionelle Angebote sind damit erfolgreich am Markt. Eigentlich wollen die Nutzer aber lieber mit einer Webapplikation arbeiten, die sich mehr wie eine native, installierte Anwendung anfühlt. Diese Programme zeichnen sich durch eine wesentlich aktivere und intuitivere Art der Benutzung aus. Eine Aktion führt schneller zur gewünschten Reaktion.

Aus diesen Forderungen entwickelten sich Plug-ins, die eine proprietäre Technologie im Browser bereitstellten. Die bekanntesten Vertreter sind Java Applets, Flash/Shockwave und Silverlight. Diese bieten mehr Freiheit und ermöglichen aktivere Applikationen. Insbesondere Spiele und Multimedia-Applikationen wurden mit diesen Softwaretechnologien implementiert.

Die Nachteile der heterogenen Welt traten in den letzten Jahren immer stärker in den Vordergrund: Die Plug-ins sind nicht auf allen Plattformen verfügbar. Ständig müssen neuen Versionen installiert werden. In manchen Fällen stellten die Erweiterungen sogar Sicherheitsrisiken dar.

Seit 2005 wirbelte ein neuer Begriff viel Staub auf: AJAX (Asynchronous JavaScript and XML). Mit Hilfe von dynamischen und asynchronen Anfragen bringt AJAX mehr Dynamik in den Browser. Die HTML-Seite wird weiterhin auf dem Server erzeugt und der größte Teil bleibt statisch. Kleine Bereiche werden je nach Benutzeraktionen nachgeladen oder »on-the-fly« ausgetauscht. Partial Page Rendering (PPR) ist eines der Schlagworte.

Typische Beispiele sind Vorschlagslisten, die sich abhängig von der Eingabe anpassen und Vorschläge automatisch vervollständigen. In einigen populären Frameworks sind AJAX-Komponenten heute häufig zu finden. Frameworks wie JSF (Java Server Faces) gelingt damit der Spagat: Dynamik im Client zu bieten, ohne das Programmierparadigma grundlegend zu ändern. Für manche Aufgabenstellungen ist das ein akzeptabler Weg.

Aus diesen ersten Anfängen haben sich die aktuellen mächtigen Frameworks, allen voran jQuery, entwickelt. Ohne diese Hilfe bei der Manipulation des DOM-Baums einer HTML-Seite verzweifelt man schnell an den Unterschieden gängiger Browser und ihren verschiedenen Versionen.

Vor diesem Hintergrund stellt die aktuelle Entwicklung zur Single-Page-Web-Applikation den logisch nächsten Schritt dar. Das Web erwächst vom einfachen Container für statische Inhalte zur Plattform und Umgebung für komplexe, integrierte Anwendungen, die sich vor herkömmlichen Applikationen nicht mehr verstecken müssen.

Eine der ersten Applikationen mit diesem Ansatz war GMail, gefolgt von den anderen Office-Produkten von Google, die jetzt unter der Bezeichnung »Google Apps for Business« angeboten werden. Von der Textverarbeitung über Tabellenkalkulation bis hin zu Präsentationssoftware ist fast alles dabei, was der typische Anwender braucht.

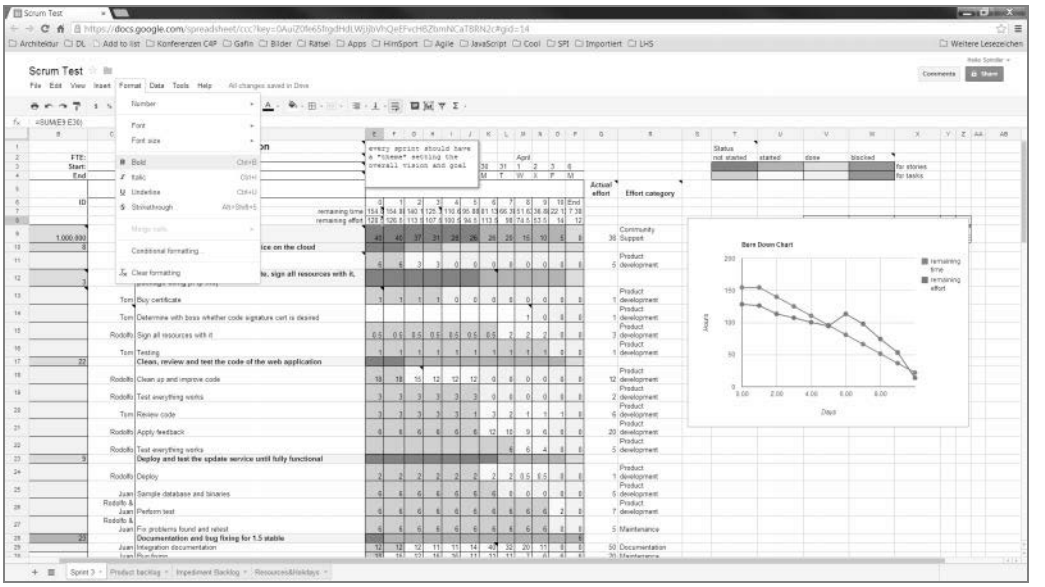

Bild 1.2: Tabellenkalkulation im Netz mit »Google Apps For Business«.

1.4 Warum erst jetzt?

Grundsätzlich sind diese Technologien schon seit langem verfügbar. Bereits im Jahr 1992 spendierte Netscape seinem Browser in der Version 2.0 JavaScript. Warum wird diese Technik erst jetzt für die Entwicklung so interessant?

Dafür gibt es eine Reihe von Gründen, die sich in den letzten zwei bis drei Jahren herausgebildet haben:

JavaScript wird schneller

JavaScript wird interpretiert und nicht nativ auf der CPU des Rechners ausgeführt. Die JavaScript-Engines wurden erst in den letzten Jahren optimiert. Intensive Anwendungslogik und das Manipulieren von HTML-Seiten sind erst jetzt effizient möglich. Vor allem Google hat im eigenen Browser (Chrome) JavaScript deutlich beschleunigt, schon mit dem Ziel vor Augen, der Sprache den Stellenwert einer Plattform zu geben. Seit 2008 findet ein regelrechtes »Speed Race« zwischen den Entwicklern von Browsern und JavaScript-Umgebungen statt.

Stark im Team: HTML5, CSS 3 und JavaScript

Die Fähigkeiten der einzelnen Technologien ergänzen sich heute im Browser sehr gut. Aus der Erfahrung der letzten Jahre konnten mit den aktuellen Versionen gezielt Funktionslücken geschlossen werden. Mussten früher bestimmte Layout-Elemente von Hand zusammengestückelt werden, wie zum Beispiel der Schattenwurf oder

abgerundete Ecken, bieten heute CSS und HTML viele Möglichkeiten, um die Kreativität der Designer zu beflügeln.

Die folgende Liste ist nur eine kleine Auswahl der neuen Fähigkeiten, die HTML5 und die begleitenden Technologien bieten:

- Leichtes Integrieren und Abspielen von Multimedia-Inhalten
- Canvas-Objekt für das Zeichnen von Grafiken
- LocalStorage für Caching und Offline-Fähigkeiten
- Integrationen von SVG (Scalable Vector Graphic)
- Erweiterungen von CCS 3 (Transformationen)
- Erweiterungen für HTML-Formulare (Placeholder und Eingabevalidierung)

EcmaScript Version 5

Aber auch die Sprache JavaScript selbst wurde erwachsener: Mit dem Sprachstandard EcmaScript 5 kommen Sprachfeatures hinzu, die JavaScript auf eine professionellere Ebene heben: Der »Strict Mode« erlaubt es, Prüfungen einzuschalten, die fehlerträchtige Programmkonstrukte melden. Der Interpreter moniert diese Stellen und zwingt Entwickler zu mehr Disziplin. Ziel ist es, mehr Lesbarkeit und Nachvollziehbarkeit zu erreichen. Da dieses Thema für die professionelle Entwicklung in Teams besonders wichtig ist, liefert das Kapitel 2 weitere Informationen.

Produktive Frameworks

Eine Programmiersprache alleine reicht für die professionelle Entwicklung nicht aus. Notwendig sind auch Frameworks, die Strukturen vorgeben und Routineaufgaben abnehmen.

Frameworks wie jQuery, Prototype und »Script.aculo.us« haben den Zugriff auf den DOM-Baum vereinheitlicht und die problematischen Unterschiede zwischen den Browsern versteckt. Gerade jQuery hat daran einen großen Verdienst. Vielleicht sind diese neuen Frameworks sogar der Hauptgrund für den aktuellen Erfolg von JavaScript.

1.5 Für welche Applikationen sind SPWA geeignet?

Bisher werden meist einfache und kleine Applikationen in dieser Form erstellt, die in ihrem Funktionsumfang überschaubar sind. Komplexere Beispiele sind noch die Ausnahme. Grundsätzlich sind SPWA nicht für alle Arten von Anwendungen geeignet. Sie haben ihre Stärken bei dynamischen Webseiten, die sich wie eine Applikation verhalten sollen und zum Beispiel Berechnungen ausführen, Eingaben prüfen oder den Nutzern viele Möglichkeiten der Anpassung bieten.

SPWA bieten sich als Ersatz für die Fälle an, in denen bisher Technologien wie Java Applets, Flash oder Silverlight zum Einsatz kamen. Browser-Spiele können somit als eine Spezialform von Web-Anwendungen gesehen werden.

Durch die vermehrte Dynamik auf dem Client scheinen ebenfalls neue Applikationskonzepte realisierbar: kollaborative Applikationen, die durch Nachrichten (anderer Benutzer oder automatisch erzeugt) reagieren. Beispiele für diese Anwendung könnten Gruppen-Whiteboards oder das gemeinsame Arbeiten an Dokumenten sein.

Der Ansatz passt ebenfalls gut für mobile Anwendungen: Die Kommunikation zwischen Client und Server reduziert sich auf den Transport von Daten, nachdem die Applikation initial geladen wurde. Das kommt den meist geringen Bandbreiten im mobilen Umfeld entgegen. Sinnvoll scheint der Einsatz für datengetriebene Business-Applikationen, die ein Frontend für umfangreiche Logik auf dem Server darstellen.

1.6 Nachteile von Single-Page-Applikationen

Wirklich verstanden hat man einen Ansatz erst, wenn man neben dessen Stärken auch seine Schwächen kennt. Weniger gut geeignet scheint der SPWA-Ansatz für folgende Arten von Applikationen zu sein:

- Applikationen, die sehr aufwendige Berechnungen durchführen oder direkt große und umfangreiche Datenmengen verarbeiten müssen.
- Web-Angebote, deren Schwerpunkt auf statischen Inhalten liegt. Beispiele hierfür sind Artikelseiten oder Blogs.
- Applikationen, die spezielle Hardware erfordern oder sehr betriebssystemnah sind. Hierzu zählen ebenfalls spezielle Bedienkonzepte oder Fähigkeiten der Geräte. Es gibt Werkzeuge, die dabei helfen, Web-Applikationen in native Apps zu verwandeln – ein Beispiel hierfür stellt das 5. Kapitel vor. Diese Frameworks erzielen gute Ergebnisse, erreichen aber nicht immer die volle Integration in das native System.

1.7 Was sind die Herausforderungen?

Single-Page-Web-Applikationen verhalten sich nicht wie typische Webseiten, woraus einige besondere Herausforderungen entstehen. Für die meisten dieser Herausforderungen gibt es aber recht einfache Lösungen. In den weiteren Kapiteln des Buches zeigen konkrete Beispiele, wie das funktioniert.

Initiales Laden

Das erstmalige Laden der Applikation könnte länger dauern als bei einer klassischen Webseite. Die umfangreicheren JavaScript-Dateien inklusive eingesetzter Bibliotheken müssen über die Leitung. Hinzu kommt vermutlich eine längere Initialisierung durch

den Browser beim Starten. Eine statische HTML-Seite kann wahrscheinlich schon angezeigt werden, bevor sie komplett geladen ist. Bei einer SPWA muss das JavaScript komplett geladen sein, um es starten zu können. Wenn dann erst noch Daten vom Server zu holen sind, geht viel Zeit ins Land. Damit ist der erste Aufruf einer SPWA wahrscheinlich ein kritischer Punkt, der schon während der Entwicklung zu prüfen ist.

Moderne Browser

Voraussetzung für den Einsatz von SPWA ist ein moderner Browser in einer aktuellen Version. Der Internet Explorer von Microsoft bietet erst ab Version 9 die volle Unterstützung des JavaScript-Standards EcmaSript Version 5. Bei vielen Nutzern und Unternehmen sind deutlich ältere Browser im Einsatz. Zum Glück hat sich dieses Problem aber in den letzten Jahren abgeschwächt. Für eine geplante Applikation empfiehlt es sich, die Zielgruppe und die exakten Anforderungen aufeinander abzustimmen.

Number Crunching

Applikationen, die sehr umfangreiche Berechnungen durchführen, sollten nicht in einer interpretierten Sprache wie JavaScript erstellt werden. Die Anbieter, insbesondere Google, haben ihre Browser schon deutlich optimiert, aber der Unterschied zu einer kompilierten Sprache besteht weiter. Es gibt bessere Möglichkeiten für die Implementierung von rechenintensiven Algorithmen (wie Number Crunching). Eine Web-Anwendung kann natürlich eine wunderbare Oberfläche für eine solche Anwendung bieten.

Es ist trotzdem erstaunlich, was mittlerweile mit JavaScript möglich ist. So bieten die modernen Browser eine gute Integration mit WebGL und können (mit Unterstützung einer modernen Grafik-Karte) 3-D-Modelle flüssig anzeigen. Erste Spiele zeigen, dass dieses Vorgehen selbst auf vielen Tablets und Smartphones gut funktioniert.

Browser History und Deep Links

Klickt man sich durch eine Web-Applikation, sieht man meist, wie sich die URL verändert und eventuell mit Parametern anreichert. Die gerade sichtbare Seite kann über diese URL direkt angesprungen werden (Deep Links).

Das ist eine nützliche Funktion, von der Benutzer bei Bookmarklisten regen Gebrauch machen. Eine SPWA durchbricht dieses Prinzip leider, da sich die Seite dynamisch ändert. Aus Sicht des Browsers gibt es keinen Grund, die URL zu ändern, obwohl sich die Applikation in einem neuen Zustand befindet.

Mittlerweile gibt es mit der History-API die Möglichkeit, das Verhalten des Browsers für die Anzeige der URL zu steuern. Damit können Applikationen den Nutzern den Status einer Applikation in der Eingabezeile für die URL zeigen und Bookmarks können wie gewohnt gespeichert werden.

SEO: Die Kunst, gefunden zu werden

Was nützt die schönste Web-Applikation, wenn sie nicht gefunden wird? SEO (Search Engine Optimization) vereint alle Maßnahmen, mit denen eine Webseite

optimiert werden kann, damit diese für ausgewählte Begriffe bei Suchmaschinen einen hohen Stellenwert erhält. Ziel ist es, bei den gewünschten Begriffen möglichst weit vorne in der Ergebnisliste zu landen. Für statische Webseiten gibt es viele Tipps und Tricks für die erfolgreiche Optimierung.

Viele Maßnahmen, die das Gestalten der Seite betreffen (On-Site-Optimierung), widersprechen dem Konzept, eine Anwendung mit dynamischen Inhalten zu bauen. Diese Maßnahmen funktionieren bei SPWA nicht automatisch. Der Crawler hat eine andere Sicht auf die Seite als ein menschlicher Benutzer, weil er das JavaScript nicht interpretiert und nicht ausführt, um die dynamisch angepassten Seiten zu indexieren. Er findet also kaum Inhalt für die Indexierung.

Dieses Problem ist nicht leicht zu lösen. Eine Lösungsidee könnte sein, dem Crawler eine zusätzliche statische Version der Seite anzubieten. Dieses Vorgehen widerspricht leider den Vorgaben von Google, weil damit viel Missbrauch getrieben wurde (Cloaking). Im Extremfall kann das Vorgehen zu einer negativen Bewertung oder dem kompletten Ausschluss führen. Als Lösung bietet sich an, zusätzlich statische Inhalte bereitzustellen, die immer sichtbar sind.

Alles sichtbar: Die gläserne Applikation

Bei einer SPWA liegt der JavaScript-Programmcode im Klartext auf dem Client-Rechner vor. Ein Blick in das Cache-Verzeichnis oder in die Browser-Konsole entblößt die komplette Applikation mit ihrer gesamten Logik. Ungewollt wird damit vielleicht eine Preisstaffel oder die Endscheidungs- beziehungsweise Validierungslogik offen gelegt. In einer serverseitigen Applikation hat man hingegen alles unter Kontrolle und unter Verschluss. Gesendet wird nur der generierte HTML-Code.

Es gibt Maßnahmen, mit denen das Problem abgeschwächt werden kann: Der JavaScript-Code kann »obfuscated« werden: Alle sinnvollen Namen für Variablen und Funktionen werden in kryptische, wenig aussagefähige, eventuell zufällige Bezeichner transferiert.

Dadurch verschwindet ebenfalls jede sinnvolle Formatierung. Das Lesen und Verstehen des Algorithmus wird deutlich aufwendiger, aber nicht vollständig verhindert. Alternativ können sensible Daten und Berechnungen auf den Server ausgelagert und per Web-Service vom Client aufgerufen werden. Mit allen Maßnahmen erkauft man sich zusätzlichen Aufwand für die Implementierung.

1.7.1 JavaScript: Gehasst und geliebt?

An der Sprache JavaScript selbst scheiden sich die Geister vieler Entwickler. Wenn man sich der Sprache als Neuling nähert, erscheint sie auf den ersten Blick sehr einfach. Schnell kommt man zu ersten kleinen Programmen. Die Sprache bietet viele Möglichkeiten, sehr kompakt zu programmieren. Das kann leider auch leicht zu unverständlichen Codes führen. Außerdem bietet JavaScript einige außergewöhnliche

Sprachkonstrukte, die leicht missverstanden werden. Das zweite Kapitel stellt einige Beispiele im Detail vor.

Bei kleinen Projekten sind diese Fallstricke meist unproblematisch. In größeren Teams und umfangreichen Projekten hingegen erwächst daraus ein Risiko: Der Programmcode wird eventuell anfälliger für Fehler, der Wartungsaufwand könnte steigen. Ein verständlicher und wartbarer Code muss einen hohen Stellenwert haben. Ohne sinnvolle Konventionen und Disziplin bei allen Beteiligten im Team ist das nur schwer zu erreichen.

Werkzeuge und Tools

Neben der eigentlichen Programmiersprache gehören zum professionellen Umfeld zusätzliche Werkzeuge. Für den Start in die Web-Entwicklung reicht ein besserer Texteditor aus; wenn es komplexer wird, sollte man Wert auf mehr Unterstützung legen. Von anderen Programmiersprachen sind Features wie Syntax-Hervorhebung, Auto-Vervollständigung oder Refactoring bekannt. Diese Funktionen zahlen sich schnell aus. Für eine dynamische Sprache scheint es aufwändiger zu sein, diese mächtigen Funktionen anzubieten. Insgesamt bleibt der Funktionsumfang meist hinter anderen Sprachen zurück.

Eine recht neue Entwicklung sind Programmierumgebungen, die direkt im Browser funktionieren. Gerade für die Web-Entwicklung können so Code und die Ergebnisse auf einen Blick bearbeitet werden. Vertreter dieser Gattung sind zum Beispiel jsFiddle (*jsfiddle.net*) und jsBin (*jsbin.com/.net*).

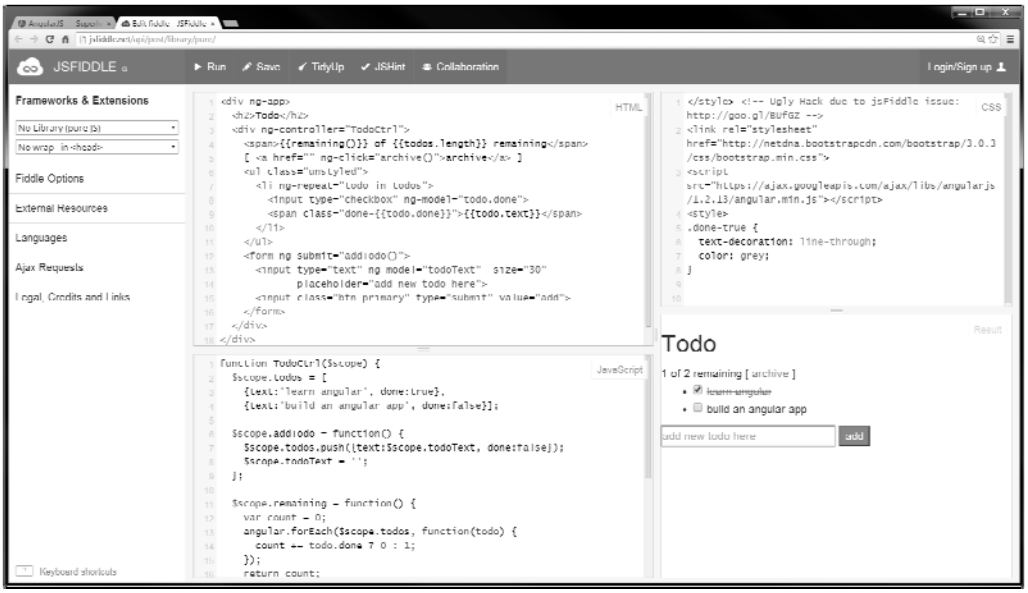

Bild 1.3: Ein geöffnetes Beispiel in jsFiddle mit HTML-, JavaScript- und CSS-Code und dem ausgeführten Ergebnis unten rechts.

Werkzeuge zur Code-Analyse und für die Automatisierung entstehen gerade erst oder sind noch nicht völlig ausgereift. Hier gibt es im Umfeld der Web-Entwicklung Nachholbedarf.

1.7.2 Interne Strukturen von Applikationen

Nichttriviale Anwendungen kommen ohne eine gute interne Struktur nicht aus. Die Aufteilung in separate Module mit abgegrenzten Aufgabenbereichen ist ein kritischer Erfolgsfaktor. Erste Ansätze und Frameworks sind in den letzten Jahren entstanden, haben sich aber noch nicht vollständig durchgesetzt. Im Bereich der UI- und MVC-Frameworks (Model-View-Controller) hingegen schießen die Frameworks wie Pilze aus dem Boden und man verliert als Entwickler den Überblick.

Ein Hauptanliegen der folgenden Kapitel ist, in diesem Umfeld mehr Überblick zu liefern. Welche Frameworks gibt es und wie lassen sie sich zu einer sinnvollen und tragfähigen Anwendungsarchitektur kombinieren?

Test-getriebene Entwicklung (TDD)

Sehr zu empfehlen ist die test-getriebene Entwicklung, die sich für andere Plattformen als sehr sinnvoll erwiesen hat. Für jede nichttriviale Funktion erstellt der Entwickler Unit-Tests, die immer wieder leicht ausgeführt werden können. Die Tests prüfen jede Funktion in Isolation und stellen sicher, dass eine Änderung keinen anderen Teil der Applikation negativ beeinflusst. Ein plötzlich fehlschlagender Test legt eine technische Abhängigkeit schnell offen. Das Team erhält die Chance, schnell zu reagieren. Das Vorgehen bringt dem Team mehr Sicherheit, auch bei wachsender Komplexität in größeren Projekten mit vielen technischen Abhängigkeiten.

1.8 Architektur-Modell

Wie sieht das grobe Architektur-Modell einer Web-Anwendung aus? Um diese Frage zu beantworten, betrachten wir die wesentlichen Komponenten mit ihren Aufgaben und Beziehungen untereinander. Einfach dargestellt, sind die wesentlichen Teile folgende: Client (in Form des Browsers), Server und Datenbank.

Die folgende Abbildung zeigt die Veränderung einer typischen Anwendungsarchitektur über die letzten Jahre. Dabei ist die wesentliche Logik immer weiter vom Server hin zum Client gewandert.

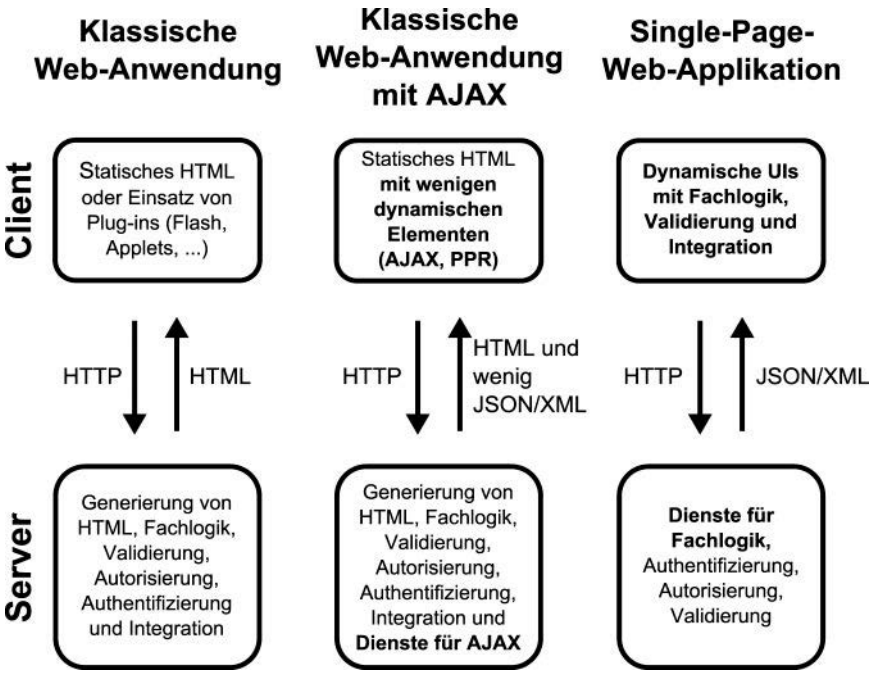

Bild 1.4: Grober Architekturvergleich von klassischer Web-Applikation und SPWA.

Die Datenbank dient hauptsächlich als Container für Daten. Trotzdem können hier wesentliche Aufgaben einer Anwendung realisiert sein, zum Beispiel Historisieren, Prüfung der Konsistenz oder das Aufbereiten von Daten bei großem Volumen.

Die zeitliche Abfolge (von links nach rechts) zeigt, wie sich der Server mehr zum Anbieter von Diensten verändert und den Client unterstützt. Typische Basisdienste sind Authentifizierung und Autorisierung. Es bietet sich an, die anwendungsspezifische Logik ebenfalls in Form von Diensten oder Services zu realisieren. Das könnten Berechnungen und Validierungen sein. Sinnvolle und generische Services können ebenfalls leicht in anderen Szenarien erneut Verwendung finden.

Der Client verwandelt sich vom Viewer für statisches HTML in eine Ausführungsumgebung und Plattform. Er führt die wesentliche Logik direkt aus, erzeugt die Oberfläche dynamisch und spricht Dienste auf dem Server an. Der Client integriert unter Umständen andere externe Dienste und führt die Daten zusammen.

Zusätzlich zeigt die Abbildung, wie sich das Datenformat für die Kommunikation zwischen Server und Client von HTML zu Rohdaten in Form von JSON oder XML gewandelt hat. Meist ändert sich im Zug dessen auch das Kommunikationsverhalten: von seltenen umfangreichen Übertragungen kompletter HTML-Seiten hin zu kleineren und häufigen Transfers.

1.9 Ein erstes Beispiel

An dieser Stelle soll ein erstes kleines Beispiel zeigen, wie leicht die Entwicklung mit JavaScript und modernen Frameworks sein kann. Dazu werfen wir einen Blick auf ein einfaches Beispiel mit AngularJS, das im dritten Kapitel tiefer mit mehren praxisrelevanten Applikationen vorgestellt wird.

1.9.1 »Hello World« mit AngularJS

Eine der Hauptaufgaben ist die Verbindung der Daten (in Form eines Modells) und der Darstellung in der View. Der View-Teil wird, wie bisher üblich, als HTML umgesetzt. Das Programm soll unsere Eingabe in einem HTML-Eingabefeld an einer anderen Stelle in der Seite wieder ausgeben:

```
<!doctype html>
<html ng-app>
<head>
 <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.4/angular.min.js"><
/script>
</head>
<body>
  <div>
    <label>Name:</label>
    <input type="text" ng-model="theName" placeholder="Enter a name here">
    <hr>
    <h1>Hello {{theName.toUpperCase()}}!</h1>
  </div>
</body>
</html>
```

Bild 1.5: Ausgabe des ersten AngularJS-Beispiels.

Das Script-Tag bindet das Framework ein. Ein weiterer Code in Form von Bibliotheken oder handgeschriebenem JavaScript ist in diesem Fall nicht notwendig.

Content Delivery Network
Ein Content Delivery Network (CDN) speichert global oft angefragte Dateien (wie zum Beispiel populäre Frameworks) zwischen und liefert diese sehr optimiert aus. Eine Bibliothek muss nicht lokal im Projekt liegen, sondern kann über ein CDN bezogen werden. Die Adresse für AngularJS in der Version 1.2.4 lautet:
ajax.googleapis.com/ajax/libs/angularjs/1.2.4/angular.min.js

An dem äußeren umschließenden HTML-Tag befindet sich die Markierung `ng-app`. Damit erkennt AngularJS, dass es die Kontrolle für diesen Teil der Seite übernehmen soll. Im Beispiel ist das die komplette Seite. In einer umfangreichen Webseite könnte man so nur genau den Bereich auswählen, der durch AngularJS dynamisch verändert werden soll. Dadurch reduziert sich der Aufwand bei der Initialisierung und weniger HTML-Code ist durch das Framework zu analysieren.

Eine ähnliche Kennzeichnung findet sich am Eingabefeld `ng-model="theName"`. Sie definiert den Namen für die Modell-Variable, die mit dem Eingabefeld verbunden ist. Das reicht für unser einfaches Beispiel aus. Wir müssen keine zusätzliche Modellvariable von Hand erstellen. AngularJS überwacht alle Änderungen in diesem Eingabefeld und überträgt diese in das Modell. Wie die Modellvariable im JavaScript angesprochen wird, untersuchen wir später.

Im Beispiel soll die Eingabe an einer anderen Stelle ausgegeben werden: Hierzu bietet AngularJS einen Template-Mechanismus an: AngularJS sucht im HTML-Code nach Elementen in doppelten geschweiften Klammern der Form: `{{ Ausdruck }}`. Der Ausdruck in den Klammern wird ausgewertet und durch sein Ergebnis ersetzt. Hier könnte jegliche Art von Logik angesprochen werden. Im Beispiel wandeln wir lediglich die Eingabe in Großbuchstaben um.

Stichwortverzeichnis

Heiko Spindler

Single-Page-Web-Apps

Single-Page-Web-Apps sind eine neue Art, Web-Anwendungen zu bauen. Im Gegensatz zu klassischen Webseiten führen Single-Page-Web-Apps keinen Seitenwechsel mehr durch – die Oberfläche wird über dynamischen Austausch der HTML-Elemente auf einer einzigen Seite mit JavaScript verändert.

Die Implementierung erfolgt mit den Technologien HTML5, CSS3 und JavaScript. Die HTML-Seiten werden zum größten Teil dynamisch im Browser erzeugt. Die Daten werden meist über JSON oder XML mit einem Backend ausgetauscht.

Bessere Verteilung

Eine Single-Page-Web-App ist über eine URL im Browser universell erreichbar. Eine Installation ist nicht notwendig. Im Unternehmensumfeld reduziert diese Eigenschaft enorm die Kosten für Installation und Verteilung.

Einheitliche Plattform

Viele Benutzer wollen Dienste auf unterschiedlichen Geräten nutzen. Der Zugriff muss vom heimischen PC genauso gut funktionieren wie vom Tablet oder Smartphone aus. Eine separate Entwicklung für jede Zielplattform ist teuer. Das Web wird die übergreifende Plattform für alle Betriebssysteme und Gerätearten. Dank HTML und JavaScript laufen Single-Page-Web-Apps auf allen wichtigen mobilen Betriebssystemen wie Android, Windows Phone und iOS.

Offline-Fähigkeiten

Mit den neuen Fähigkeiten von HTML5, wie dem LocalStorage, gibt es zum ersten Mal eine Möglichkeit, effiziente Cache-Strategien und Offline-Fähigkeiten zu etablieren.

Geringe Einstiegshürden

Es ist sehr leicht, mit den verwendeten Technologien zu starten – als Entwicklungsumgebung reicht ein guter Texteditor aus. Zum Ausführen reicht ein Browser, der mit den entsprechenden Plug-ins sogar Debugging bereitstellt. Alle diese Komponenten sind für den Einsteiger kostenlos verfügbar.

Aus dem Inhalt:

Über den Autor:

Heiko Spindler ist Softwarearchitekt und Autor für Themen rund um die Softwareentwicklung. Er spricht regelmäßig auf Konferenzen und schreibt Fachartikel. Als Dozent unterrichtet er seit 2008 an der Fachhochschule Gießen-Friedberg im Studiengang Wirtschaftsinformatik.

Zusätzlich beschäftigt er sich seit vielen Jahren mit Kreativitätstechniken, Gehirnjogging, Mind Mapping und Gedächtnistraining. Heiko Spindler betreibt die Websites HirnSport.de und DenkTipps.de und ist Buchautor zum Thema Gehirnjogging.

9 783645 603102

30,– EUR [D] **/ 30,90 EUR** [A]

ISBN 978-3-645-60310-2

FRANZIS