Gymnasium zum kurfürstlichen Schloss zu Mainz

Facharbeit

Webapps

das Ergebnis moderner Webtechnologien

von
Yannick Félix

24. April 2017

1 Kurzfassung

WebApps wurden von den meisten Benutzern, wenn auch unbewusst, benutzt. Oft fallen diese garnicht auf, da sie intuitiv zu benutzen sind.

Immer dann, wenn wir eine Webseite benutzten können, ohne dabei die Webseite an sich zu verlassen, oder neuzuladen, handelt es sich um eine WebApp. Einige Beispiele sind die Google Suche, Google's Email Dienst GMail oder auch Facebook.

Große Firmen stecken oft viel Geld und Ressourcen in die Entwicklung von nativen Apps für die beliebtesten Betriebssysteme, darunter Windows, Android und iOS. WebApps und auch deren Erweiterung progressive WebApps sind jedoch platformunabhängig und können auf jedem Betriebsystem benutzt werden. Somit genügt es eine Anwendung einmal als (progressive) WebApp zu entwickeln.

Eine progressive WebApp ermöglicht es sogar den Eindruck zu erwecken, dass sie eine native App sei, ohne dabei auf bekannte Features, wie Offline-Modi und Push-Benachrichtigungen zu verzichten.

Tut mir sehr leid, aber der "Benutzerprofildienst" ist heute leider nicht verfügbar. Selbst der "Gruppenrichtlinienclient" kann hier leider nichts mehr ausrichten... Versuche es in 42 Minuten erneut.

Inhaltsverzeichnis

3 Geschichte und heutige Technologien

Bevor auf die Geschichte und verschieden Technologien eingegangen wird, muss zwischen zwei Typen von Sprachen unterschieden werden: Clientseitige Sprachen, welche beim Benutzer im Browser ausgeführt werden, und serverseitige Sprachen, welche der Server ausführt, bevor oder während einer Anfrage.

3.1 Clientseitige Sprachen

HTML. Hypertext Markup Language. Von Beginn des "Internets" an ist sie die Sprache zum grundlegenden Aufbau einer Webseite. Mit der Urversion von 1992 hat heutiges HTML nicht mehr viel zu tun. Damals war Hauptbestandteil der Text und dessen Verlinkung.[9]

Über die Jahre hinweg hat sich HTML zu HTML5 respektive HTML5.1(seit Ende 2016 [4]) entwickelt. Letzteres beinhaltet vor allem Verbesserungen hinsichtlich Webapps, zum Beispiel die Einbindung von "responsive Images, Bildern, welche dem Browser in verschieden Auflösungen zur Verfügung gestellt werden. Der Browser entscheidet dann, welche Auflösung passend für die anzuzeigende Größe ist und veringert somit den Datenverkehr. [1]

JavaScript, kurz auch JS. Ursprünglich LiveScript genannt wurde JavaScript entwickelt um dynamisches HTML zu erlauben. Hierbei wird das HTML-Dokument nach dem Laden beim Benutzer verändert. Der Name JavaScript rührt daher, dass LiveScript in Kooperation von Netscape und Sun Mircosystems entwickelt wurden. Um die Bekanntheit von Sun's Sprache Java zu nutzen wurde LiveScript in JavaScript umbenannt, obwohl es eigentlich wenig mit Java syntaxtechnisch zu tun hat. [10]

Heutzutage ist JavaScript Kernbestandteil von Webapps und handelt jegliche clientseitige Aktivität.

Neben JS existieren bzw. existierten auch weitere Skriptsprachen, wie Flash und JavaApplets. Beide sind mittlerweile als obsolet markiert und stellen durch Sicherheitslücken ein hohes Sicherheitsrisiko dar.[7] Chrome hat diese, wie andere Browser auch, bereits standardmäßig deaktiviert.[2]

JSON, JavaScript Object Notation, ist ein Standard um Objekte zwischen verschiedenen Programmiersprachen zu serialisieren.

Vorteile von JSON gegenüber anderen Standards für einfachen Datenaustausch ist zum einen, dass es sowohl für Mensch, als auch für Maschine einfach zu lesen ist, zum anderen in praktisch jeder Programmiersprache ein Parser existiert um JSON-Objekte in respektive Objekte umzuwandeln. In JavaScript ist ein zusätzlicher Parser nicht von Nöten, da JSON eine valide JS Notation ist.[3]

Andere Standards zum Austausch von Daten sind zum Beispiel YAML und XML. Letzteres wird jedoch immer häufiger durch JSON ersetzt, da XML in der Kompilierung weit aus aufwendiger ist und für weniger Nutzdaten mehr Speicher benötigt.

AJAX, Asynchronous JavaScript and XML, ist ein grundlegender Bestandteil moderner Webanwendungen. AJAX ermöglicht es, anders als der Name vermuten lässt, nicht nur XML sondern jegliche Daten vom Server asynchron, also nachdem die eigentliche HTML-Seite bereits geladen wurde, nachzuladen. Hierbei wird

von JavaScript eine HTTP- bzw. HTTPs-Anfrage initiiert, dessen Antwort daraufhin auch von JavaScript verarbeitet wird, um zum Beispiel Teile einer Seite zu aktualisieren, ohne die gesamte Seite neuladen zu müssen.[8]

WebSockets, werden, im Gegensatz zu AJAX-Verbindungen, aufrecht erhalten, um so schnell wie möglich Daten zwischen dem Server und dem Client austauschen zu können. Diese Technologie eignet sich vor allem für Echtzeitanwendungen, wie Livechats und Browserspiele[15]

3.2 Serverseitige Sprachen

PHP, "PHP: Hypertext Preprocessor", mittlerweile, mit über 80% der Websites, die am weitesten verbreitetste serverseitige Skriptsprache.[6]

PHP1, damals noch für *Personal Home Page Tools*, wurde als Ersatz zu Perlskripten von Rasmus Lerdorf entwickelt. Mit PHP3, welches von Andi Gutmans und Zeev Suraski entwickelt wurde(Lerdorf wurde als Entwickler auch eingestellt), wurde die Sprache von Grund auf neu entwickelt und unter dem rekursiven Akronym *PHP: Hypertext Preprocessor* veröffentlicht.

Mit PHP4 war es zudem möglich objektorientiert zu Programmieren, welches mit PHP5 weiter verbessert wurde.

Die aktuellste Version ist PHP7.1. Mit PHP7 kamen, 11 Jahre nach PHP5, vorallem eine verbesserte Performance und sowie einige neue Features dazu.[11]

3.3 Webapp

Die Funktion einer interaktiven Webseite setzt die Möglichkeit vorraus, dass der Webserver Daten empfangen kann, die der Benutzer eingegeben hat. Bereits in der Ur-Version von HTML war es möglich Werte an den Webserver zu schicken, ähnlich dem heutigen HTTP-GET Verfahren. Eine der ersten Seiten war "SPIRES-HEP", ein Webinterface für eine Datenbank der Stanford Universität aus dem Jahr 1991.

Die erste von der Öffentlichkeit wahrgenommene Webanwendung war Yahoo, welches ebenfalls an der Stanford Universität von zwei Studenten entwickelt wurde.

Seit AJAX in JavaScript Einzug erhalten hat, ist es möglich auch dynamisch Teile der angezeigten Webseite zu verändern, oder sogar die Präsentationsschicht komplett auf den Client auszulagern, hierbei sendet während der normalen Benutzung der Server nur noch benötigte Informationen.[14]

Aktuell spricht man auch von "Progressiven Web Apps", wenn diese nicht nur komplett clientseitig ausgeführt werden können, sondern auch Features wie ein Offline-Modus und Push-Benachritigungen enthalten. Außerdem können diese den Eindruck erwecken, eine native App zu sein, obwohl keine Installation benötigt wird. Leider werden diese bisher nur von Chrome und Firefox unterstützt, andere Browser haben zum Zeitpunkt der Recherche angekündigt progressive WebApps in Zukunft zu unterstützen.[12]

4 Entwicklung einer progressiven WebApp

4.1 Vorbereitung

Zur Vorbereitung gehören neben der Einrichtung der Entwicklungsumgebung und des Webservers auch die Sammlung benötigter Bibliotheken.

Bei der Einrichtung des Webservers wurde ein Ordner für dieses Projekt samt passender Zugriffsrechte und gültigen SSL-Zertifikat erstellt. Dieses Zertifikat ist nicht nur für die Funktionalität von Service Workern, bei welchen eine Verbindung via SSL Pflicht ist, sondern auch für den generellen Datenschutz des Nutzers wichtig.

Bei den benutzten Bibliotheken wurde sich auf zwei Hauptbibliotheken beschränkt:

Materialize CSS[1]**:** Dies ist eine CSS/JS-Bibliothek, welche jegliche Designelemente Googles Material Design Definition zur Verfügung stellt.

jQuery[2] ist eine der beliebtesten JavaScript-Bibliothek[5], welche viele häufig benötigten JavaScript-Funktionen in einfachere, schnell zu benutztende Funktionen vereint.

Einige kleinere JavaScript Funktionen wurde zusätzlich verwendet um zum Beispiel Hashfunktionen wie md5 auszuführen oder Graphen und Diagramme zu zeichnen.

4.2 Struktur

4.2.1 JavaScript

...

4.2.2 Datenbank

...

4.2.3 Klassen in PHP

...

4.2.4 API

...

[1] Materialize Frontend-Framework: http://materializecss.com/
[2] jQuery JavaScript Library: https://jquery.com/

4.3 Entwicklungsprozess

Die Reihenfolge der einzelnen Schritte sind in den meisten Fällen persönliche Präferenz. In diesem Fall wird mit der Entwicklung des Clients begonnen, denn dieser soll, da er die eigentliche WebApp widerspiegelt, auch ohne Internetverbindung laufen. Die meiste serverseitige Entwicklung kann somit im Anschluss geschehen.

4.3.1 Client Grundlagen

Zunächst werden Grundlagen im JavaScript, also clientseitig gelegt. Ziel einer progressiven WebApp ist es, dass diese nicht nur aussieht, als sei sie eine native App, sondern auch ohne Datenverbindung funktioniert. Hierfür wird der Client an das Prinzip einer Single-Page-Webanwendung[13] angelehnt. Somit wird nur selten die gesamte Seite neugeladen, sondern das Grundgerüst wird zu Beginn einer Sitzung geladen und jeder Inhalt wird von JavaScript mittels DOM-Manipulation dynamisch eingefügt.

Der erste Schritt wird also mit dem Erstellen der Datei “index.html” getan. Diese enthält die sog. App-Shell, welche jenes Grundgerüst darstellt.

Im Beispiel des Schulplaners ist es jedoch erwünscht die Login- / Registrierungsoberfläche von der “App” ansich abzugrenzen. Deshalb erhält die Datei “index.html” hier nicht dei App-Shell, sondern die Login-Maske. Für das Grundgerüst wird abweichend “app.html” verwendet.

Die eigentliche Entwicklung dieses Grundgerüsts unterscheidet sich nicht stark von der einer herkömmlichen HTML- Webseite. Jedoch wird hier nur die Menüführung und Kopfzeile erstellt. Der eigentliche Inhalt wird später mittels JavaScript eingefügt.

4.3.2 Das Manifest

Das Manifest ist der Bestandteil einer progressiven WebApp, der es (unter anderem) möglich macht, diese zum Homescreen hinzuzufügen.

Es besteht aus einer JSON-Datei, welche in jedem HTML-Dokument einer WebApp im `<head>`-Bereich verlinkt sein muss. In dieser Datei werden neben dem Namen, Icons und der Start-URL auch Parameter zur Darstellung wie der Hauptfarbe und der Orientation festgelegt. Der Parameter `"display": "standalone"` macht aus einer normalen Webseite eine “echte” WebApp, indem es jegliche UI-Elemente des Browsers verbergt und die Webseite wie eine native App aussehen lässt.

4.3.3 Der Service-Worker

Grundlegend ist ein Service-Worker ein JavaScript-Skript, welches vom Browser “installiert” wird. Das heißt der Browser speichert das Skript und führt dieses, nicht nur solange eine Seite bzw. Tab offen ist, sondern auch darüber hinaus, aus.

In Google Chrome zum Beispiel funktionieren Service-Worker, genauso wie einige andere Funktionen, darunter Zugriff auf die Webcam oder den Standort, nur

wenn die aufgerufene Seite ein “secure Origin” ist, hierbei *muss* die Seite unter anderem vollständig über SSL ausgeliefert werden. Ausnahmen hierfür sind zu Entwicklungszwecken `localhost` und `127.0.0.1`. Außerdem darf pro Webseite nur ein Service-Worker existieren.

Eine Aufgabe, die dieser Service-Worker erledigt, ist das sogenannte “Caching” von Dateien. Hierbei werden alle zur Offline Benutzung benötigten Resourcen in einem “Cache” zwischengespeichert. Somit kann die WebApp größtenteils auch offline verwendet werden.

Da die Entwicklung dieses Teils des Service-Workers oft mit viel reproduktiver Arbeit verbunden ist, haben die Google Chrome Entwickler eine Bibliothek[3] zur Verfügung gestellt. Mit dieser lässt sich nach eingestellten Regeln ein Service-Worker erstellen, der alle benötigten Resourcen “cachet”. Somit muss auf dem Entwicklungsrechner lediglich ein lokales Skript ausgeführt werden, welches alle zu cachenden Dateien im Service-Worker verlinkt.

Eine weitere Aufgabe des Service-Workers besteht darin, Push-Benachrichtigungen zu verwalten. Hierbei muss der Benutzer erst bestätigen, dass dieser Nachrichten erhalten darf. Durch eine Bestätigung hat dieser den Push-Dienst abonniert, und erhält ab dann Push-Benachrichtigungen. Diese Bestätigung kann jederzeit wieder zurückgenommen werden, falls zum Beispiel jene den Benutzer stören.

4.3.4 View-Klasse: Account Einstellungen

Durch ECMA-Script 2015 und 2016, kurz ECMA-Script 6 oder ES6, ist es möglich “echte” Klassen in JS zu schreiben. Diese werden ähnlich zu anderen Programmiersprachen so definiert:

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }
}
```

Als ersten Teil wird sich nun die “Account Einstellungen”-Seite vorgenommen, hier kann später neben Änderungen am Passwort und der Email-Adresse und anderen Optionen auch die Erlaubnis zum Senden bzw. Empfangen von Push-Benachrichtigungen gegeben, oder auch zurückgezogen werden.

Um eine gute Struktur im JS zu behalten, wird sich an den zuvor erklärten ES6-Klassen bedient. Für jeden “View”, also jede Unterseite, wird eine eigene “View-Klasse” erstellt. Diese beinhaltet, im Gegensatz zu Standart-Klassen, keine Daten, sondern beherbergt Methoden und Eigenschaften der Darstellung dieser Unterseite

[3] `sw-precache` - GitHub: `https://github.com/GoogleChrome/sw-precache`

5 Anhang

5.1 Literaturverzeichnis

[1] Philippe le Hegaret. *HTML 5.1 IS THE GOLD STANDARD*. Englisch. 17. Nov. 2016. URL: https://www.w3.org/blog/2016/11/html-5-1-is-the-gold-standard/ (besucht am 03. 04. 2017).

[2] Sascha Koesch. *Chrome 56 deaktiviert Flash*. 27. Jan. 2017. URL: http://de.engadget.com/2017/01/27/chrome-56-deaktiviert-flash/ (besucht am 03. 04. 2017).

[3] N/a. *Introducing JSON*. Englisch. URL: http://json.org/ (besucht am 03. 04. 2017).

[4] W3C. *HTML 5.1*. Englisch. 1. Nov. 2016. URL: https://www.w3.org/TR/2016/REC-html51-20161101/ (besucht am 02. 04. 2017).

[5] W3Tech. *Usage of JavaScript libraries for websites*. Englisch. 5. Apr. 2017. URL: https://w3techs.com/technologies/overview/javascript_library/all (besucht am 05. 04. 2017).

[6] W3Tech. *Usage of server-side programming languages for websites*. Englisch. 3. Apr. 2017. URL: https://w3techs.com/technologies/overview/programming_language/all (besucht am 03. 04. 2017).

[7] Markus Werner. *Weg mit Flash: Der Flash Player ist ein Sicherheitsrisiko*. 6. Apr. 2016. URL: https://www.basicthinking.de/blog/2016/04/06/flash-player-sicherheit/ (besucht am 03. 04. 2017).

[8] Wikipedia. *Ajax (Programmierung)*. 1. März 2017. URL: https://de.wikipedia.org/wiki/Ajax_(Programmierung) (besucht am 03. 04. 2017).

[9] Wikipedia. *Hypertext Markup Language*. 1. Apr. 2017. URL: https://de.wikipedia.org/wiki/Hypertext_Markup_Language (besucht am 02. 04. 2017).

[10] Wikipedia. *JavaScript*. 22. März 2017. URL: https://de.wikipedia.org/wiki/JavaScript (besucht am 03. 04. 2017).

[11] Wikipedia. *PHP*. 20. März 2017. URL: https://de.wikipedia.org/wiki/PHP (besucht am 03. 04. 2017).

[12] Wikipedia. *Progressive Web App*. 2. März 2017. URL: https://de.wikipedia.org/wiki/Progressive_Web_App (besucht am 04. 04. 2017).

[13] Wikipedia. *Single-Page-Webanwendung*. 19. März 2017. URL: https://de.wikipedia.org/wiki/Single-Page-Webanwendung (besucht am 08. 04. 2017).

[14] Wikipedia. *Webanwendung*. 8. März 2017. URL: https://de.wikipedia.org/wiki/Webanwendung (besucht am 04. 04. 2017).

[15] Wikipedia. *WebSocket*. 23. März 2017. URL: https://de.wikipedia.org/wiki/WebSocket (besucht am 03. 04. 2017).

5.2 Quellcode

To follow

6 Erklärung über die selbstständige Anfertigung der Arbeit

Hiermit erkläre ich, dass ich die vorliegende Hausarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Hausarbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Mainz, den 24. April 2017

Yannick Félix