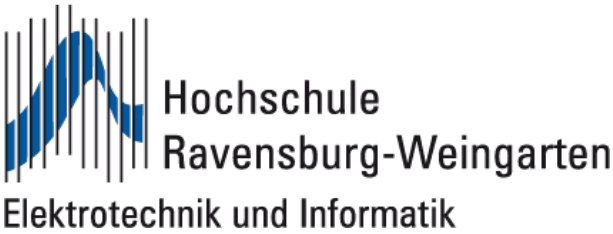

Bachelor-Thesis
Web Performance für den mobilen Endanwender

Zusammenfassung

9. März 2015

Columbus Interactive
Eywiesenstraße 6
88212 Ravensburg

Fakultät Elektrotechnik und Informatik
Studiengang Angewandte Informatik
Hochschule Ravensburg-Weingarten
Doggenriedstraße, 88250 Weingarten

Autor:
Andreas Lorer
andreas.lorer@hs-weingarten.de
88250 Weingarten
Wilhelmstraße 4

Bachelor-Thesis

Web Performance für den mobilen Endanwender

Inhaltsverzeichnis

1 Einleitung

2 Motivation

Larry Page, CEO und Mitgründer von Google, sagt:

> *„As a product manager you should know that speed is the number one feature.“*(Holzle 2010)

Niemand mag es zu warten, auch nicht auf eine Website. Die Studie „The Psychology of Web Performance“ kam bereits im Jahr 2008 schon auf folgende Ergebnisse:

> *„Slow web pages lower perceived credibility and quality. Keep your page load times below tolerable attention thresholds, and users will experience less frustration, lower blood pressure, deeper flow states, higher conversion rates, and lower bailout rates. Faster websites are actually perceived to be more interesting and attractive.“ (WebSiteOptimization.com 2008)*

Das haupt Vermarktungsargument für den Chrome Browser war damals: er sei schneller als die Konkurenz. Tatsächlich ist für Google Geschwindigkeit alles. Deshalb hat Google im Jahr 2010 angekündigt, dass Geschwindigkeit in die Berechnung des `Page Rankings` mit einfließt.

> *„Faster sites create happy users and we've seen in our internal studies that when a site responds slowly, visitors spend less time there. [...] Recent data shows that improving site speed also reduces operating costs. Like us, our users place a lot of value in speed — that's why we've decided to take site speed into account in our search rankings“(Google 2010)*

Aktuell (2015) geht Google sogar noch einen Schritt weiter und informiert tausende Webmaster per E-Mail über die schlechte Usability ihrer Websites für mobile Besucher und warnt ausdrücklich vor dementsprechend „angepassten Rankings“.(t3n 2015) Im Hinblick auf die Zukunft wird der Marktanteil an mobilen Internetnutzern noch weiter wachsen und die Optimierung der Ladezeiten gewinnt dadurch noch mehr an bedeutung. Zwischen 2011 und 2014 stieg die Anzahl der Smartphone nutzer von 18% auf 50% an. Dies ist ein Wachstum von 32% innerhalb von nur 3 Jahren.(TNS Infratest 2014)

Die Antwort auf diesen Trend läutete eine Ära ein, die wir heute unter dem Namen `Responsive Webdesign` kennen. „Responsive“ muss aber sehr viel mehr bedeuten, als nur eine angepasste Darstellung für eine bestimmte Art von Gerät. „Two out of three mobile shoppers expect pages to load in 4 seconds or less.“ (Radware 2013). Der Anwender erwartet also auf dem Smartphone änliche oder gleiche Ladezeiten wie er auch von der Nutzung eines Desktop-Pc's gewohnt ist. Diese Erwartungen werden von dem Großteil der im Internet besuchbaren Seiten nicht erfüllt.
Der Inhalt einer Seite muss darum so aufbereitet werden, dass dieser auch auf Geräten mit langsamer Internetverbindung, hoher Latenz und einem begrenzten Datentarif, in einer für den Anwender annehmbaren Geschwindigkeit, angezeigt werden kann.

2.1 Zielsetzung

Um gängige Methoden und Techniken der Ladezeit Optimierung anzuwenden wird das Projekt anhand der Website `http://andreaslorer.de` durchgeführt. Das Ziel ist es, die Ladezeit der

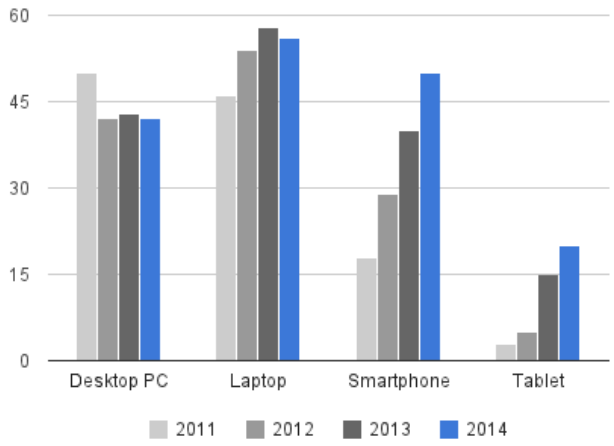

Abbildung 1: Gerätenutzung in der Gesamtbevölkerung (2011 – 2014)(TNS Infratest 2014)

Website auf dem Smartphone, als auch auf dem Desktop von 10 Sekunden auf unter 1 Sekunden zu verringern. Mit Ladezeit ist dabei nicht die Zeit gemeint, die benötigt wird um die Website komplett zu laden, sondern die Zeit bis ein erste visuelle Rückmeldung für den Anwender zu sehen ist. Diese vom Anwender wahrgenommene Rückmeldung nennt man auch „Perceived Performance“ und bedeutet, dass die Ladezeit als schneller empfunden wird, als es eigentlich laut Messwerten der Fall ist. Näheres dazu wird in Punkt 5.9 beschrieben.

3 Eigene Leistung

Meine Leistung besteht darin, einen Leitfaden zu erstellen, der einen Gesamtüberblick ermöglicht. Die Arbeit soll es dem Leser ermöglichen Fehler in der Struktur von Webanwendungen zu finden, die für die Geschwindigkeit hinderlich sind. Es soll herausgefunden werden, was die „Best Practices“ sind um die Ladezeit zu minimieren, wie ein moderner „workflow“ aussehen kann, damit eine Webanwendung schon bei seiner Entstehung schnell ladet und im Projektverlauf schnell bleibt. Des weiteren soll erklärt werden, was für herausforderungen es zu meistern gilt um eine schnelle Webanwendung zu erreichen, welche Tools es gibt und welche Vor- oder Nachteile diese mit sich bringen.
Diese Arbeit befasst sich nicht, mit der Geschwindigkeit von Datenbanken, SQL-Abfragen oder sonstigen problemen die durch einen Engpass ein schnelles Laden der Seite verhindern könnten.

4 Ist-Zustand

Die Webseite ist auf einem `shared Hosting` [1] aufgesetzt und antwortet auf ein Ping Kommando in rund 13ms. Dadurch, dass es keine Möglichkeit gibt `root Rechte` [2] auf einem shared hosting zu bekommen können so manche serverseitige Einstellungen nicht durchgeführt werden. Diese werden dann zwar Aufgezeigt, aber kommen für dieses Projekt nicht zum Einsatz.
Die Website hat als Ausgangsbasis einen gängigen Aufbau. Sie besteht aus einer Bilder Gallerie basierend auf PHP und dem Bootstrap Framework.

5 Begriffe

5.1 Pattern und Antipattern

„Entwurfsmuster (englisch design patterns) sind bewährte Lösungsschablonen für wiederkehrende Entwurfsprobleme sowohl in der Architektur als auch in der Softwarearchitektur und -entwicklung. Sie stellen damit eine wiederverwendbare Vorlage zur Problemlösung dar, die in einem bestimmten Zusammenhang einsetzbar ist.“(Wikpedia 2015)

„Während „Design Patterns“ in der Software-Entwicklung allgemein übliche und bekannte Ansätze sind, um Probleme zu lösen, sind Anti-Patterns Negativ-Beispiele – die zeigen, wie man es nicht macht - von bereits durchgeführten, gescheiterten Projekten, die dem erkennenden Mitarbeiter zielgerichtete Hinweise darauf geben, wie die Aufgabenstellung besser gelöst werden könnte. Als Synonym ist auch der Begriff Negativmuster im Gebrauch. Es ist tatsächlich möglich, daß das, was gestern noch als allgemein gangbarer Lösungsweg bezeichnet wurde, heute schon ein „Antipattern“ ist [...]“ (Stepken 2006)

5.2 TCP Three Way Handshake

TCP ist das meistgenutzte Verbindungsprotokoll im Internet. Auf diesem Protokoll wird der HTTP Request aufgebaut, der die eigentlichen Daten enthält. Bevor Daten zwischen Server und Browser ausgetauscht werden können, muss eine Verbindung aufgebaut werden. Abbildung 2 beschreibt den Prozess des Verbindungsaufbaus.[3]

5.3 Latenz

Latenz bezeichnet die Verzögerung, bis ein Paket von Sender A zu Empfänger B gelangt ist.

5.4 Round Trip Time (RTT)

„Round Trip Time“ wird im Deutschen Paketumlaufzeit genannt. Es bezeichnet die Zeit die ein Datenpaket braucht um in einem Netzwerk von Sender A zu Empfänger B und wieder zurück zu gelangen. Bei einer Latenz von 100ms würde die RTT folglich 200ms betragen (Annahme: Hin- und Rückweg haben die selbe Zeit).

[1] Bei shared Hosting werden mehrere Websites von verschiedenen Website-Betreibern von dem gleichen Webserver gehostet. Bei Shared Hosting teilen sich in der Regel Hunderte andere Websites einen Server (ItWissen.info 2015)

[2] Standardmäßig existiert unter Linux immer ein Konto für den Benutzer „root“ mit der User-ID 0. Dies ist ein Systemaccount mit vollem Zugriff auf das gesamte System, und damit auch auf alle Dateien und Einstellungen aller Benutzer (wiki.ubuntuusers 2014)

[3] Für ein tieferes Verständnis empfiehlt sich dieser Artikel: High Performance Browser Networking - Chapter 2: Building Blocks of TCP

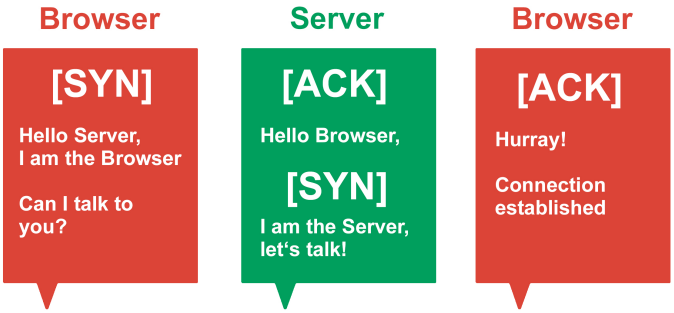

Abbildung 2: Three-Way-Handshake zum Aufbau einer TCP Verbindung zwischen Browser und Server (Eigene Abbildung nach: (Stefanov 2011))

5.5 TCP Slow Start

Ein Round Trip kann nicht beliebig viele Bytes transportieren sondern ist durch die sogenannte „Congestion Window Size“[4] limitiert. Der Überbegriff für dieses Verhalten nennt sich „Slow Start“

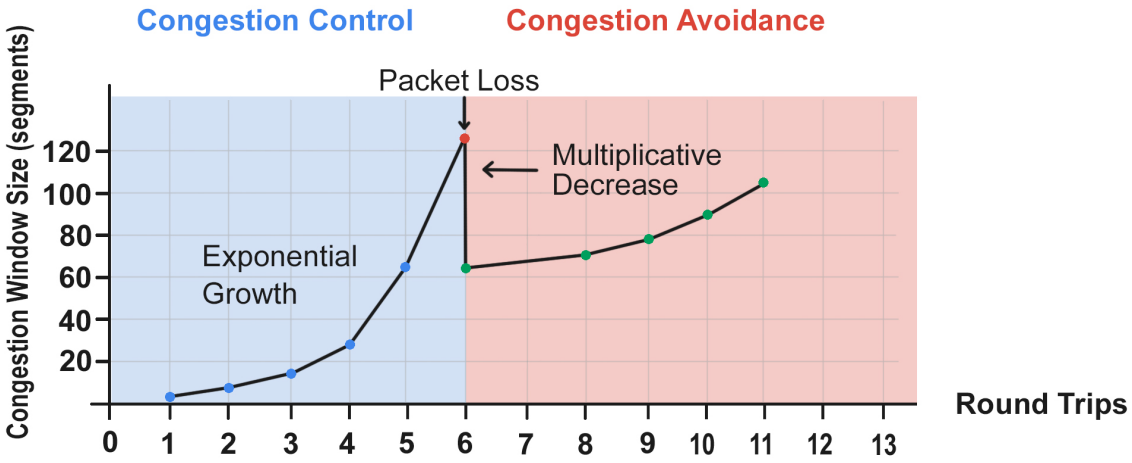

Abbildung 3: Congestion Control und Congestion Avoidance (Eigene Abbildung nach (Grigorik 2013c))

- Congestion Control: Nach dem eine neue Verbindung per TCP aufgebaut wurde, können weder Server noch Client wissen, wie schnell die Verfügbare Bandbreite ist, mit der Daten ausgetauscht werden können. Um das Netzwerk, vor einem Datenstau zu schützen, wird mit einem sehr niedrigen Wert begonnen, der dann ansteigt bis das Limit erreicht ist. Dieses Verhalten nennt sich auch „Congestion Control“ und verhindert das Aufstauen von Daten.
- Congestion Window Size: Diese Größe bestimmt, wieviel Bytes der pro Segmente geschickt werden darf, bis diese vom Empfänger per `ACK` (acknowledgement) bestätigt werden müssen. Die Größe der Segmente ist Standardmäßig 1460 bytes und die Rate bis zum ACK

[4] engl. congestion: Stauung, Überlastung, Anhäufung

ist im April 2013 von 4 auf 10 Segmente erhöht worden.(Grigorik 2013c). In der Grafik wird davon ausgegangen, das der erste Round Trip 4 Segmente senden darf. Die Datenrate Wächst exponentiell an, damit möglichst schnell die volle Bandbreite nutzbar ist.

- Congestion Avoidance bedeutet, dass sich die Datenrate wieder um ein Vielfaches verringert, falls es zu einem Paketverlust kommt. Da es besonders bei WLAN oder Mobilfunknetzen des öfteren zu Packetverlusten kommen kann ist dieser Aspekt besonders hervorzuheben, denn er verzögert das erreichen der maximal möglichen Datenrate.

Slow Start bedeutet also aus sicht der Performance, dass bei einer neuen TCP Verbindung nicht die maximale Bandbreite zu Verfügung steht. Bei größeren Dateien wird zwar durch das exponentielle Wachstum das Maximum schnell erreicht, gerade aber bei kleineren Dateien mit wenigen Kilobyte ist dies oft nicht der Fall.

5.6 Http/1.1

5.7 Content Delivery Network (CDN)

Ein Content Delivery Network (CDN), oder auch Content Distribution Network genannt, ist ein Netz lokal verteilter und über das Internet verbundener Server, mit dem Inhalte ausgeliefert werden.
CDN-Knoten sind auf viele Orte verteilt um Anfragen (Requests) von End-Nutzern nach Inhalten (Content) möglichst ökonomisch zu bedienen.
Große CDNs unterhalten tausende Knoten mit zehntausenden Servern.(wikipedia 2015)

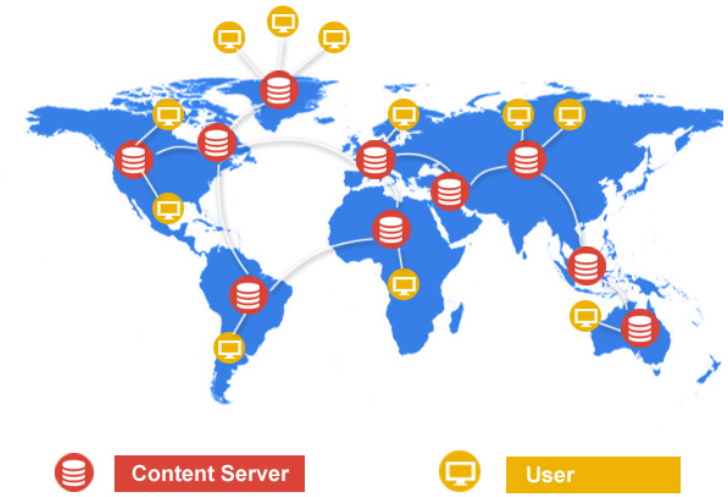

Abbildung 4: Schematische Darstellung eines CDN (Eigene Abbildung nach (Ritz 2014))

5.8 Above The Fold

Damit ist der auf einem Bildschirm sichtbare Bereich vor dem Scrollen gemeint. Diesem Bereich wird eine besondere Wichtigkeit zugesprochen.

Abbildung 5: Darstellung des sichtbaren Bereichs vor dem Scrollen

> *„In an analysis of 57,453 eyetracking fixations, we found that there was a dramatic drop-off in user attention at the position of the page fold. Elements above the fold were seen more than elements below the fold: the 100 pixels just above the fold were viewed 102% more than the 100 pixels just below the fold.“* (Schade 2015)

Wichtige Informationen oder Navigationselemente sind meistens dort zu finden. Eine Webseite die nach dem Paradigma des Responsive-Webdesign aufgebaut ist kann dabei 3 oder mehrere Ansichten haben die alle einen unterschiedlichen „above the fold“ bereich haben. Eine Anwendung kann aber auch unterschiedliche Seiten haben, auf dem der Anwender beim Aufrufen der Seite landen kann. Zum Beispiel wenn dieser An- oder Abgemeldet ist. Paradebeispiel dafür sind Facebook oder Twitter.

5.9 Perceived Performance

Abbildung 6 zeigt die Seiten A und B, mit nahezu identischer Ladezeit. Der Unterschied besteht darin, dass Seite B bereits nach 1.5 Sekunden eine erste visuelles Rückmeldung für den Anwender zu sehen ist wohingegen Seite A erst nach 5.5 Sekunden dem Anwender zeigt, dass sie überhaupt ladet. „Perceived Performance“ steht also für die Zeit bis ein erste visuelle Rückmeldung für den Anwender zu sehen ist und bedeutet, dass die Ladezeit als schneller empfunden wird, als es eigentlich laut Messwerten der Fall ist. Warum diese „Perceived Performance“ für eine Webanwendung so wichtig ist zeigen mehrere Studien, deren Daten in folgender Infographik aufbereitet sind.
Bereits kleine Verbesser- oder Verschlechterungen der Ladezeit können einen großen Einfluss in auf den Anwender haben. Yahoo hat herausgefunden, dass wenn eine Seite um nur 400 Millisikunden schneller ist, sich der Traffic um 9% erhöhte.(Stefanov 2008) 57% der Online Konsumenten haben eine Seite, die länger als 3 Sekunden ladet bereits wieder verlassen. 78% der Anwender empfinden sogar Zorn oder Stress wenn eine Seite nicht Ladet oder dies nicht ersichtlich ist.

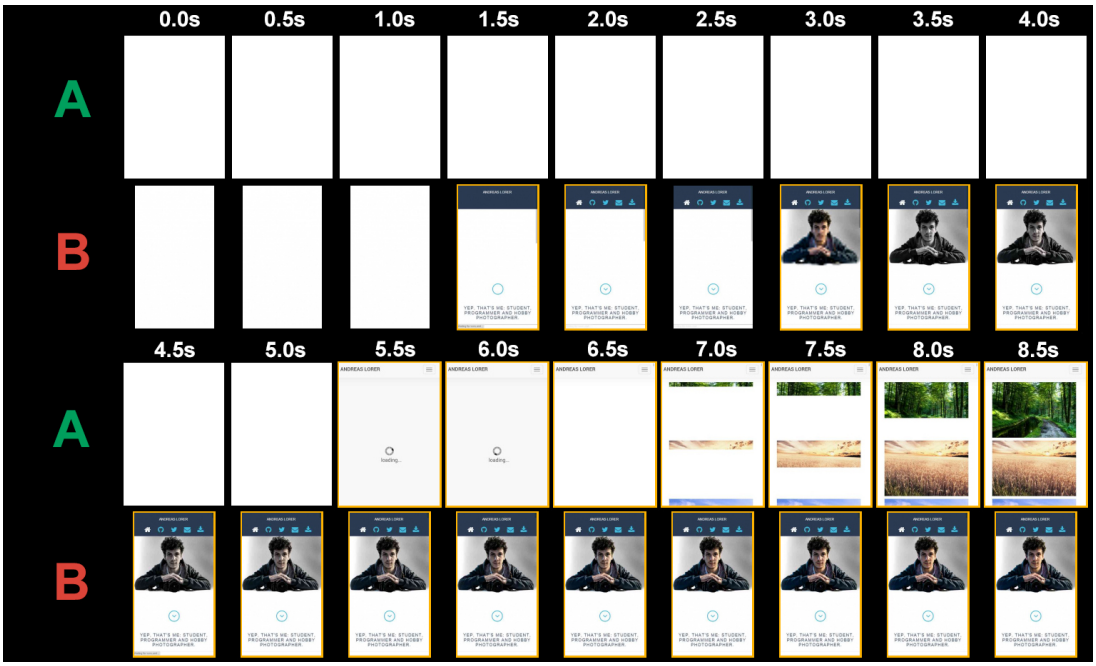

Abbildung 6: Zwei Seiten im Vergleich (Eigene Abbildung via webpagtest.org)

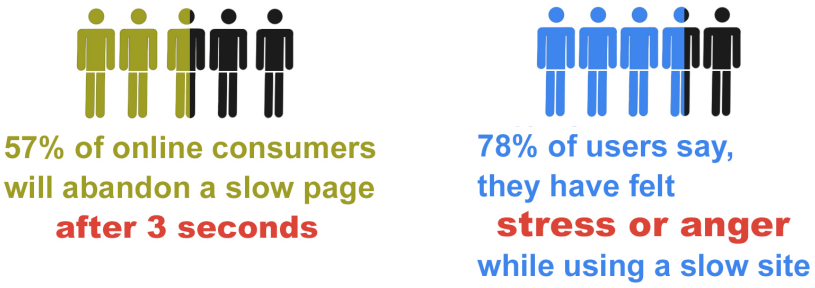

Abbildung 7: Einfluss und Effekt einer langsamen Seite auf den Anwender (Eigene Abbildung nach Daten von: (Radware 2014, p. 8))

6 Die 1000 ms Barriere

Das Ziel dieser Arbeit, die 1000 Millisekunden Barriere zu durchbrechen, wurde nicht durch einen Zufall gewählt. Der Anwender nimmt die Geschwindigkeit einer Seite subjektiv wahr. Sie wird in der folgenden Grafik interpretiert:

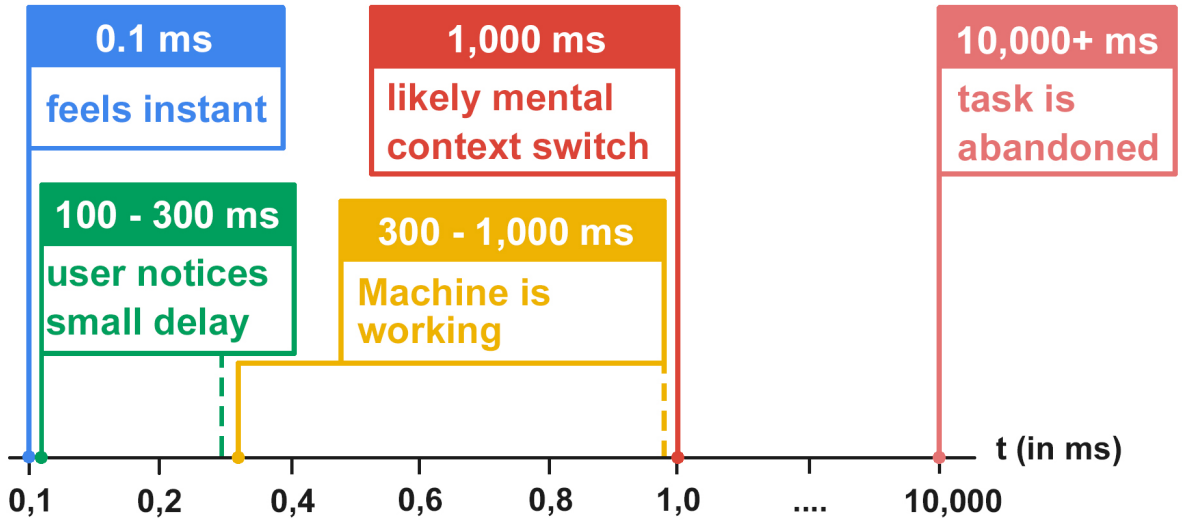

Abbildung 8: Zeit und Wahrnehmung durch den Anwender (Eigene Abbildung nach Daten von: (Grigorik 2013d))

Wie zu sehen ist bleiben gerade einmal eine Sekunde, bevor das Gehirn uns sagt, man solle doch einer anderen Aufgabe nachgehen bis der Ladevorgang abgeschlossen ist. Der Anwender verlangt visuelle Rückmeldung um „am Ball zu bleiben“, dies wurde bereits in Punkt 5.9 „Perceived Performance“ angesprochen. Auf vielen Webseiten sieht man deshalb, Ladebalken oder sogenannte `Spinner`, die dem Anwender sagen, dass der Ladevorgang in gange, aber noch nicht abgeschlossen ist.
Um das Ziel von einer Sekunde Ladezeit bis zum ersten Render zu erreichen, ist es nötig zu verstehen womit die meiste Zeit beim Aufrufen einer Webanwendung verbracht wird. Bevor eine Seite mittels Smartphone vom Browser dargestellt wird läuft eine ganze Reihe von Prozessen ab.

6.1 Touch Event

Der Aufruf einer Seite über das Smartphone erfolgt über ein Touch Event auf einen Link, Button oder die Seite wird per URL aufgerufen. Hierbei können je nach Gerät zwischen 50 (IPhone 5) und 123 Millisekunden (Moto X - Android) zwischen der Berührung des Touch Screen und dem Registrieren des Events vergehen.(Takahashi 2013) Der Browser wartet allerdings nochmals bis zu 300 Millisekunden, denn er muss abwarten ob vielleicht noch ein zweiter Finger aufgelegt wird, oder ob der Anwender Scrollen oder Zoomen möchte.(Google 2011)
Dieses Verhalten lässt sich bei vielen Browsern per `Meta Tag` abstellen:

```
1                    <meta name="viewport" content="user-scalable=no">
```

Dies setzt natürlich voraus, dass die Webanwendung kein Zoomen benötigt um sie zu bedienen! Gerade bei älteren Webseiten trifft das oftmals nicht zu, da sie keine für das Smartphone

angepasste Ansicht haben (responsive view). Eine vollständige Liste mit Meta Tags für die verschiedenen Browser ist der Fußnote zu entnehmen.[5]

6.2 Netzwerke

Warum gerade das nutzen des Internets per Smartphone so langsam sein kann (und oftmals ist) liegt zu einem Großteil am Netzwerk. Eine Studie untersuchte die Top eine Millionen Webseiten des Internets auf ihre Ladezeiten. Dabei wurde eine Verbindung von 5 Mbit/s und 28ms RTT benutzt. Eine RTT von 28 ms ist sehr schnell, vergleicht man sie zum Beispiel mit der Latenz des 3G Netzes (Abbildung 10). Diese Studie kam zu dem Ergebnis, dass fast 70% der Ladezeit nur durch warten auf das Netzwerk verbracht wird:

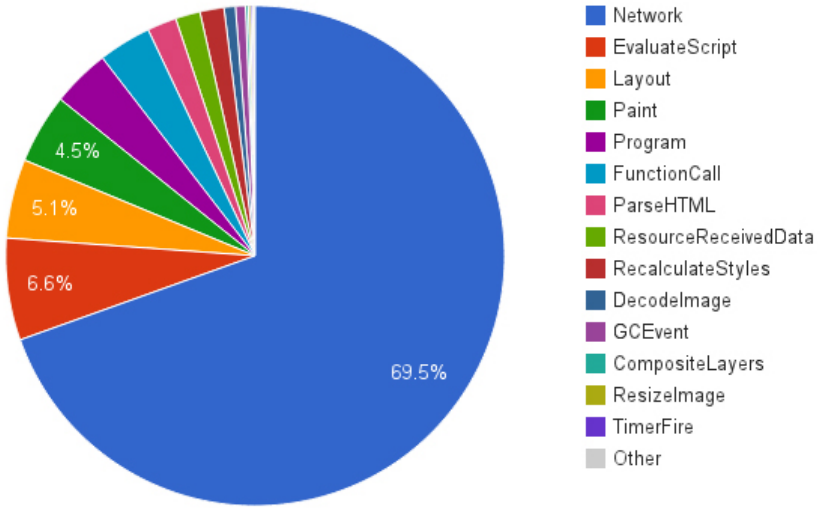

Abbildung 9: Untersuchung der top 1 Millionen Alexa Seiten (Abbildung von: (Tonyg 2013))

Es macht also durchaus Sinn, sich diesen Bereich näher anzusehen, um zu verstehen worauf Einfluss genommen werden kann und wo nicht.

6.2.1 Mobilfunknetz

Es gibt unterschiedliche Mobilfunktsandards mit denen Anwender Anbindung an das Internet erlangen. Aber selbst wenn einem Anwender 4G vom Mobilfunkanbieter versprochen wird, so ist die Netzabdeckung mit 4G noch nicht vollständig deckend. Das bedeutet, dass der Anwender auf ein niedrigeres Netz wie zum Beispiel 3G ausweichen muss. Die verschiedenen Netzwerke unterscheiden sich entscheidend in ihrer Datenrate und vor allem in der Latenz. Folgende Tabelle gibt eine Übersicht:
Unser Smartphone ist nicht ständig mit dem „wireless service provider“ verbunden. Ist eine erste Verbindung nötig, so muss das Smartphone dem Sendeturm mitteilen, dass es Kommunizieren möchte. Der Anbieter muss die Anfrage Authentifizieren, die Verbindung herstellen und dann die Anfrage in das Internet weiter leiten. Die Zeit bis eine Authentifizierung erfolgt ist, kann je nach Anbieter und Mobilfunkstandard zwischen $<$100ms (LTE) und 2,5 Sekunden (3G) liegen (Grigorik 2013a)! Bereits hier ist zu sehen, dass es „worst case“ Szenarien gibt, durch die es nicht möglich sein kann, dass eine Webanwendung unter einer Sekunde eine Rückmeldung gibt. Gerade

[5]Suppressing 300ms delay for touchscreen interactions: `http://tinyurl.com/psj5nxz`

Gernation	Data rate	Latency
2G	100 - 400 Kbit/s	300 - 1000 ms
3G	0,5 - 5 Mbit/s	100 - 500 ms
4G	1-50 Mbit/s	< 100 ms

Abbildung 10: Datenrate und Latenz für eine aktive mobile Verbindung (Eigene Abbildung nach Tabelle (Grigorik 2013e))

Mobilfunknetze unterliegen Stoßzeiten, die Funksignale von Smartphones können sich gegenseitig stören oder das Signal kann in gewissen Gegenden stärker oder schwächer sein.

6.3 Der HTTP-Request

Nachdem uns unser Mobilfunkanbieter mit dem Internet verbunden hat, kann die eigentliche Anfrage an den Server gestellt werden.

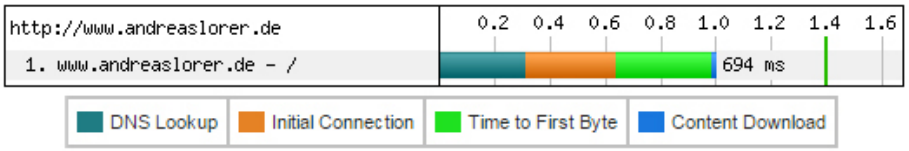

Abbildung 11: Anfrage der HTML-Datei von Irland mittels 3G Netz (Abbildung nach `http://webpagetest.org`)

- DNS Lookup: Um eine Verbindung mit dem Server herzustellen benötigt das HTTP Protokoll die IP Adresse des Ziels. Das heißt der DNS Server wird für den Namen „http://andreaslorer.de“ die zu diesem Namen zugehörige IP Adresse zurückgegeben.
- Initial Connection bezeichnet die Zeit die vergeht, bis eine neue Verbindung zum Server hergestellt wurde damit eine Kommunikation zwischen Browser und Server stattfinden kann. Hierbei findet der sogenannte TCP „Three-Way-Handshake“ statt, der dafür einen Round Trip benötigt.
- TTFB: Ist die Abkürzung für „Time to first byte“. Dieser Begriff beschreibt die Zeit die vergeht, bis das erste Byte vom Server beim Browser ankommt. Der Server muss den Request erst zusammenstellen bevor er ihn versenden kann. Dafür werden unter umständen Daten aus der Datenbank abgefragt oder es müssen berechnungen stattfinden. Diese Faktoren beeinflussen die TTFB und kann optimiert werden (schnellerer Server, bessere Datenbankanbindung, Caching).
- Content Download: Die Zeit die benötigt wird bis die Datei vom Server Heruntergeladen wurde.
- Nachdem das HTML Dokument heruntergeladen wurde, muss es vom Browser noch gelesen und interpretiert werden. Diese Zeit taucht im Diagramm nicht auf.

6.4 Das Herunterladen einer 40 KB Datei

Abbildung 12 zeigt Schematisch wie eine 40 KB Datei mittels einer neuen TCP Verbindung heruntergeladen wird. Sie soll verdeutlichen, wie vor allem die RTT eine entscheidende Rolle spielt und warum die Latenz die Geschwindigkeit einer Seite viel höher beeinflusst als die Bandbreite.

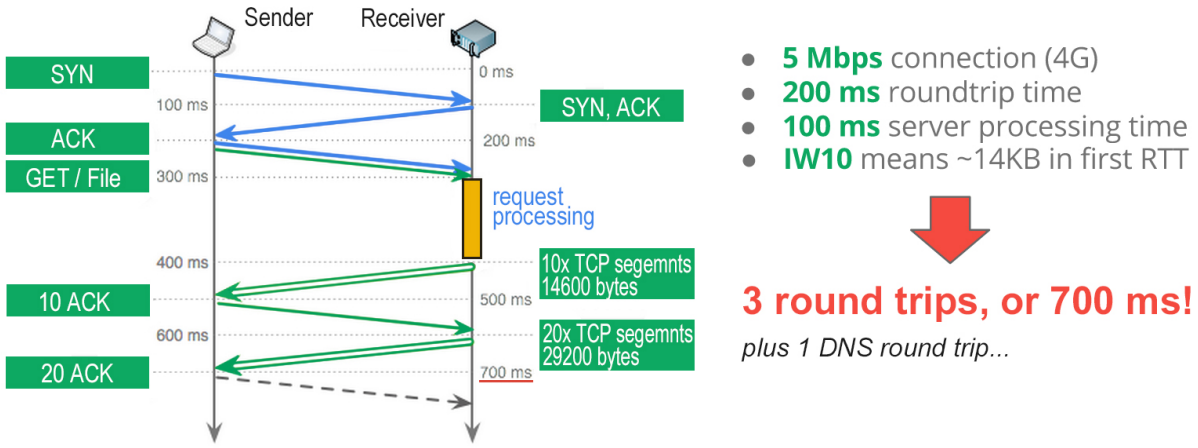

Abbildung 12: Herunterladen von 40KB mittels TCP (Abbildung nach (Grigorik 2013b))

Zuerst erfolgt der DNS Lookup, dann muss TCP per `three way handshake` eine Verbindung aufbauen. Dies kostet bereits 2 round trips, was in diesem Beispiel 400 ms entspricht. Durch den TCP slow start (siehe Punkt: 5.5 TCP slow start) steht bei einer neuen TCP Verbindung nicht die volle Bandbreite zur Verfügung. Deshalb kann die volle Datenmenge nicht auf einmal, sondern nur durch zusätzliche round trips heruntergeladen werden. Wenn die Performance einer Webanwendung verbessert werden soll, macht es also Sinn in round trips zu denken. Wieviel round trips sind nötig, bis ich dem Browser informationen übermittelt habe, so dass dieser etwas anzeigen kann?

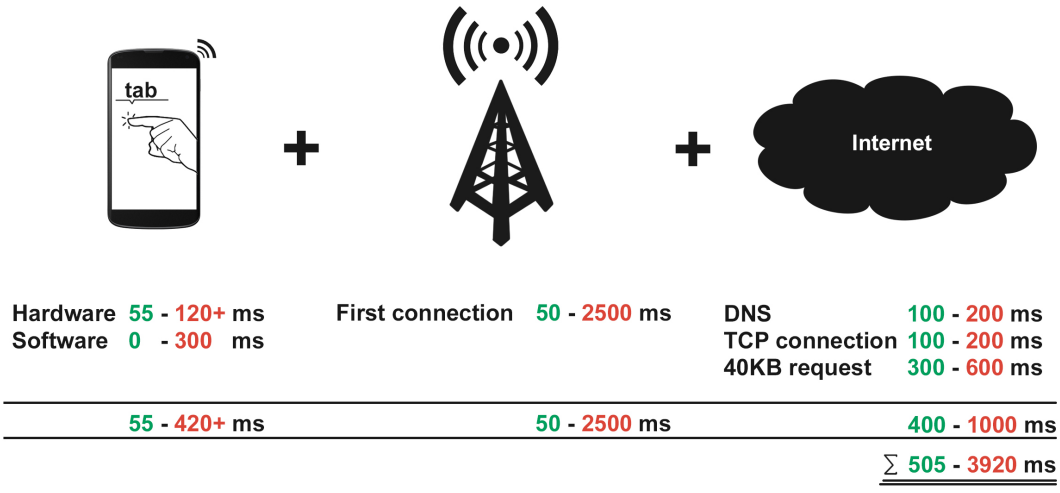

Abbildung 13: 1000 ms Budget (Eigene Abbildung)

Wie in Abbildung 13 zu sehen ist bleibt von dem 1000 Millisekunden Budged nicht mehr viel übrig, wenn man alleine die Netzwerkzeiten abzieht. Im „best case“ Szenario bleiben knapp 500 ms. Das bedeutet es müssen in den ersten Kilobytes, soviel nützliche Informationen vorhanden sein, damit der Browser bereits anfangen kann mit dem rendering zu beginnen, obwohl noch nicht alle Daten heruntergeladen sind. Noch spitzer formuliert: Der Browser sollte mit den ersten 14 KB (das ist die Menge an Daten die der erste round trip transportieren kann, siehe Abbildung 12) bereits den `above the fold` bereich rendern können. Um das zu ermöglichen ist es nötig, den Kritischen Rendering-Pfad zu optimieren.

6.5 Zusammengefasst

Folgende Optimierungsmaßnahmen ergeben sich auf der Ebene des Netzwerks:

- Näheres Platzieren der Bits: Die Latenz bestimmt vorrangig die Geschwindigkeit des Ladevorgangs. Durch die benutzung eines CDN's lassen sich Bits und Bytes näher am Endanwender platzieren.
- Sage das Nutzerverhalten voraus: Wenn der Anwender in einer Einkaufs-App 3 Schritte zum vollenden des Kaufvorgangs benötigt, dann lässt sich bei Schritt 1 bereits vorhersagen was er für weitere Ressourcen im nächsten Schritt benötigt. Diese Ressourcen könnten bereits geladen werden.
- Wahl eines guten Hostings: Die RTT als auch die `Server Response Time` sind je nach Anbieter unterschiedlich. Ein gutes Hosting kann hier unter umständen bereits eine enorme Verbesserung bedeuten. Zur not sollte gewechselt werden!

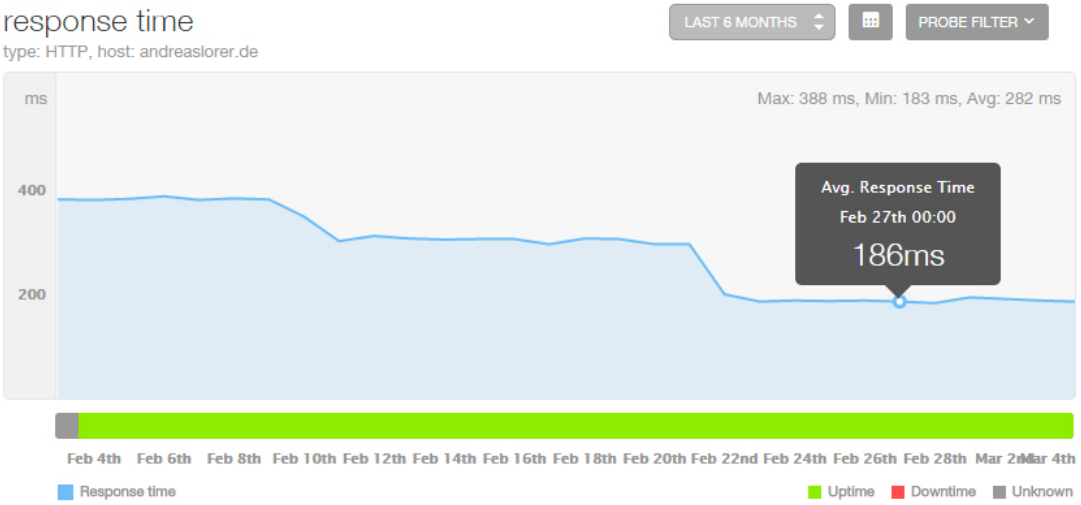

Abbildung 14: Verringerung der response time für http://andreaslorer.de (Abbildung nach pingdom.com)

Wie in der Grafik zu sehen ist, sank die Response Zeit meines Hosting Providers von durchschnittlichen fast 400 Millisekunden auf 183 ms. Dies kann mehrere Gründe haben, so kann sich das Routing zum Server geändert haben, der Server kann ein Update erhalten haben oder die Maschine kann gewechselt worden sein. Was letzten endes dazu geführt hat kann nicht gesagt werden.

- Senden von weniger Daten: Das schnellste Bit ist das, dass nicht gesendet wird. Das zusammenfügen und verkleinern von Javascript und CSS Dateien verringert die Dategröße. Zudem lassen sich die Daten per GZIP zwischen Browser und Server komprimieren. Wie dies Möglich ist wird später noch Konkretisiert.
- todo: serve useful bits in the first 14kb of your html
- Vermeiden von Weiterleitungen: Abbildung 15 zeigt den Seitenaufruf von hs-weingarten.de. Wie zu sehen ist, bekommen wir zwei mal einen HTTP 301 (Wert in Klammer) Response zurück:

  > 301 - Moved Permanently: *„Die angeforderte Ressource steht ab sofort unter der im „Location"-Header-Feld angegebenen Adresse bereit (auch Redirect genannt). Die alte Adresse ist nicht länger gültig."* (wikipedia 2014)

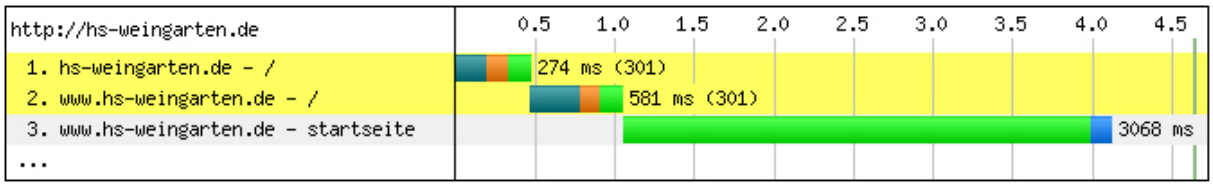

Abbildung 15: Redirects verursachen große Verzögerungen (Abbildung nach webpagtest.org)

Wenn ein Anwender also hs-weingarten.de eingibt erfolgt der DNS Lookup und die TCP Verbindung wird aufgebaut nur um dann zu erfahren, dass die Ressource unter dieser Adresse nicht zur Verfügung steht. Danach erfolgt die DNS auflösung für www.hs-weingarten.de bei dem der gleiche Vorgang vonstatten geht. Es erfolgt die Weiterleitung auf www.hs-weingarten.de/web/willkommen/startseite.html mit wieder dem selben Vorgang. Nach 855ms konnte die erste Anfrage an die richtige Zieladresse aufgegeben werden! Ruft man sich die RTT von einem 3G Netz ins Gedächtnis dürfte klar werden, was Weiterleitungen für den Smartphonenutzer bedeuten können.

Die Weiterleitung von hs-weingarten.de auf www.hs-weingarten.de ist für die Suchmaschinenoptimierung [6] allerdings Sinnvoll, denn sonst würde unter zwei Namen der gleiche Inhalt zu finden sein. Dies ist aus sicht von Google „Duplicated Content" und kann zu einer Abstrafung im Ranking führen.[7]

[6] engl. SEO - search engine optimization

[7] Mehr zu diesem Thema gibt es unter `http://www.sem-deutschland.de/seo-tipps/duplicate-content-definition/`

6.6 Kritischer Rendering-Pfad

Auf Englisch „critical render path“ genannt, ist der wohl wichtigste Begriff, wenn es um schnelle Ladezeiten geht. Durch die Optimierung des Rendering-Pfads kann die benötigte Zeit für das erste Rendern der Seite erheblich verkürzt werden. Das Verständnis des Rendering-Pfads ist zudem eine wesentliche Voraussetzung für die Erstellung von schnellen Webanwendungen und soll in diesem Abschnitt ausführlich erklärt werden. Dabei wird der Begriff in seine Teile Zerlegt: Kritischer, Rendering und Rendering-Pfad.

6.6.1 Rendering

Rendering: Der Browser liest das HTML Dokument und übersetzt dieses. Diesen Vorgang nennt man auch parsing. Stellt der Browser fest, dass die für das Rendern nötigen Ressourcen vorhanden sind, beginnt er mit dem Zeichnen. Bei der Optimierung von Webanwendungen ist besonderst das Auftreten des ersten Zeichnens interessant (engl. „First Paint Event“). Je früher das sogenannte `First Paint Event` auftritt umso höher ist die `Perceived Performance` der Webanwendung.

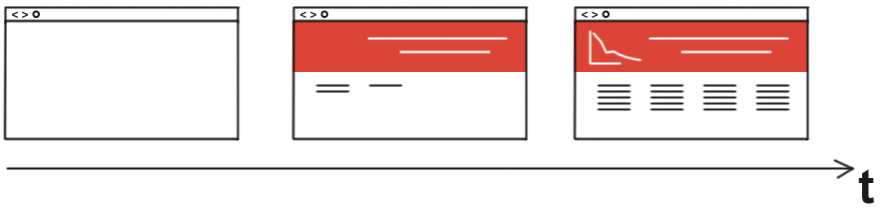

Abbildung 16: Der Render Prozess (Eigene Abbildung)

6.6.2 Rendering-Pfad

Der `Rendering-Pfad` setzt sich aus den für die Anwendung nötige Ressourcen zusammen. Webanwendungen bestehen schließlich nicht nur aus einer HTML Datei, sondern aus mehreren Javascript und CSS Dateien.

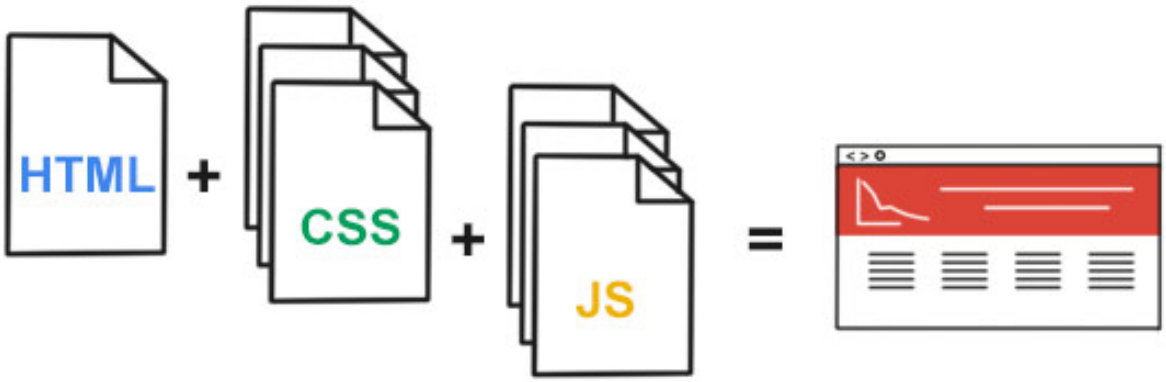

Abbildung 17: Ressourcen die für das Rendern von nöten sind (Eigene Abbildung)

Gegeben ist das folgende Beispiel einer simplen HTML Datei.

```
1                    <!DOCTYPE html>
2                    <meta charset="utf-8">
3                    <title>Web Performance fuer den mobilen
                         Endanwender</title>
4
5                    <link href="assets/styles.css" rel="stylesheet" />
6                    <script src="assets/script.js"></script>
7
8                    <p> Hello world! </p>
```

Listing 1: Beispiel Code

Nachdem diese Datei heruntergeladen wurde, beginnt der Browser sie von oben nach unten zu Parsen. Dabei stößt er in Zeile 5 auf einen `Link-Tag` der ihn anweißt diese Datei herunterzuladen. In Zeile 6 findet der Browser einen `Script-Tag`. Auch diese Datei muss herunterladen, interpretiert und ausgeführt werden, denn jede Javascript Datei kann den DOM-Baum[8] oder das CSS manipulieren. So können per Javascript sowohl Elemente dem DOM-Baum hinzugefügt, als auch weggenommen werden oder Elemente können eine Änderung ihrer CSS Attribute erhalten. Dieser Umstand verbietet es dem Browser mit dem Rendering zu beginnen, da bis zur Ausführung der Javascript Dateien noch Manipulationen erfolgen können. Bevor also das „Hello world“ in Zeile 8 angezeigt werden kann, ist das rendern blockiert. Dieses Verhalten nennt sich auch „Render Blocking“ und wird sowohl von Javascript als auch von CSS Dateien ausgelöst. Folglich spricht man hierbei auch von „Render Blocking Javascript“ und „Render Blocking CSS“. Erst wenn diese Blockierenden Ressourcen geladen und interpretiert wurden, kann der Browser mit dem Rendern beginnen.

6.6.3 Critical render path

`Critical render path` sind genau die Javascript und CSS Dateien, die für den für das Rendern des `above the fold` (Punkt: 5.8) von nöten sind. Um dies umzusetzen ist es nötig, die Ressourcen in zwei Teile zu zerlegen: Für das Rendering **absolut** notwendig und nicht notwendig. Alle Dateien die nicht notwendig für das erste Rendern sind sollten so lange mit dem Laden verzögert werden, bis die Anwendung geladen ist. Wie genau so eine Umsetzung aussieht, wird in Punkt: **??** ausführlich gezeigt.

6.6.4 Zusammengefasst

Folgende Pattern lassen sich, bedingt durch den `Kritischen Rendering-Pfad`, für die Erstellung von Webanwendungen ableiten.

- CSS Dateien möglichst weit oben im „<head>“ Bereich platzieren und Javascript vor dem schließen des „</body> tags“. Da Javascript und CSS so lange Blockieren, bis sie heruntergeladen wurden, kann mit dem Parsen des gesamten Dokumentes nicht fortgefahren werden. (Bart 2014)
  Das Platzieren von Javascript am Ende des Dokuments hat allerdings den Nachteil, dass sich der Zeitpunkt des Herunterladens verzögert. Deshalb ist es ratsam genau die Javascript Dateien in den „<head>“ zu verlagern, die `Kritisch` für das Rendern des `above the fold` Bereichs sind und den rest der Scripte vor den „</body> tag“.

- ... todo concatenating wegen anzahl an blockierenden elementen und http 1.1 und anzahl an roundtrips. Nachteil caching

[8] Der DOM-Baum: `http://wiki.selfhtml.org/wiki/JavaScript/Objekte/DOM`

- aufteilen von ressourcen in 2 gruppen, defering
- inlining von css (datei muss nicht requestet / heruntergeladen werden sondern kann gleich geparsed werden)
- herausforderung: critical ressourcen erkennen
- detect render blocking css / js via pagespeed insight

6.7 Analyse des Wasserfalls

Für ein besseres Verständnis des Kritischen Rendering-Pfads soll der stand meiner Webseite vor und nach den Performance Optimierungsmaßnahmen analyisiert werden. In Abbildung 18 ist ein Ausschnitt des Wasserfallmodell der nicht Optimisierten Webseite dargestellt. Das volle Diagramm ist der Fußnote [9] zu entnehmen.

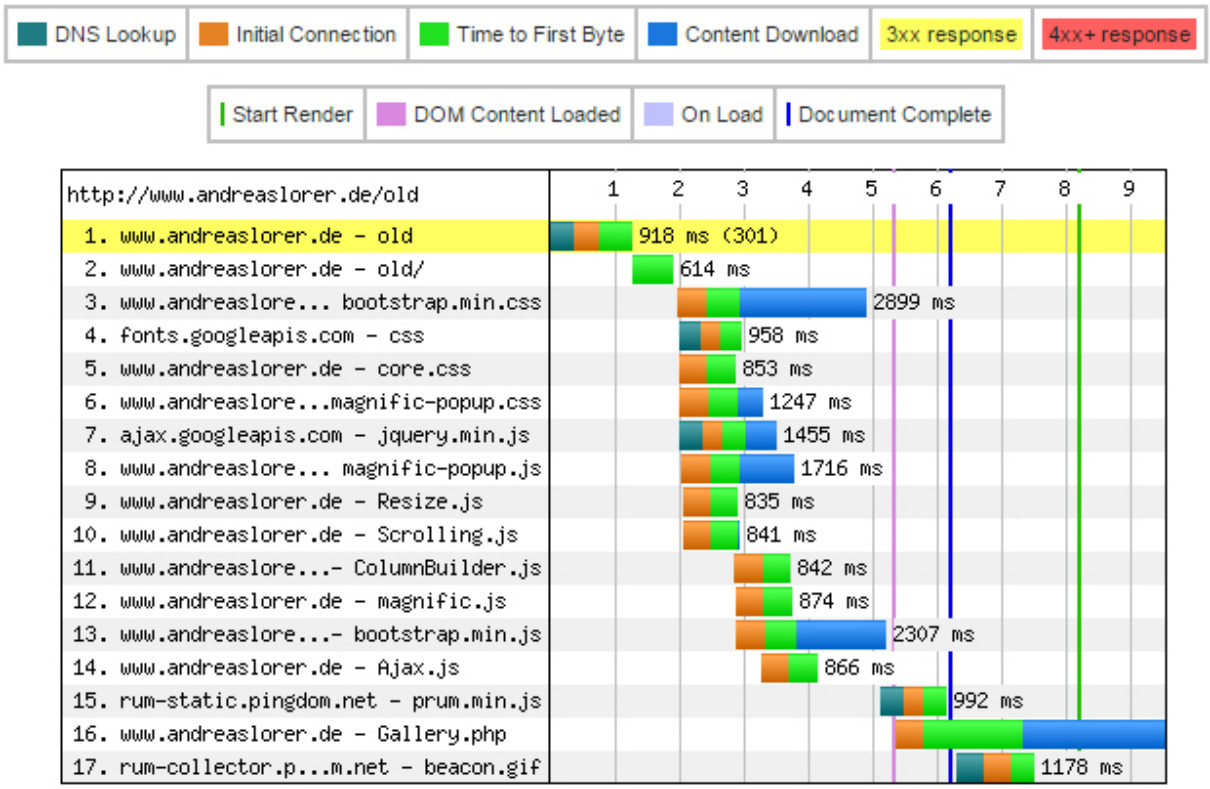

Abbildung 18: Ausschnitt des Wasserfallmodells der alten Seite (Eigene Abbildung nach webpagetest.org)

Die Abbildung zeigt, dass der Browser zuerst alle CSS und Javascript Dateien herunterläd. Hierbei fällt auf, dass er nicht mit allen gleichzeitig beginnen kann, sondern wie in Punkt 5.6: HTTP/1.1 nur 6 TCP Verbindungen pro Hostname aufbauen darf.

[9] `http://www.webpagetest.org/result/150206_11_M2M/1/details/`

7 Entwicklung

7.1 Tools

7.1.1 Google Chrome Developer Tool

Stalled / Waiting DNS Lookup TTFB Start render Document Complete Document Loaded Audits Timeline

7.1.2 Webpagetest

Was ist Webpagetest Speedindex Features

7.1.3 Google Spreadsheet

7.2 Ausgangspunkt

7.3 Prozess der Suche

7.4 Prozess der Validierung

8 Best-Practices

8.1 Serverseitig

8.1.1 Verringern von DNS Lookups

8.1.2 HTTP Requests

pro / contra von sprites und großen concatenated files

8.1.3 Caching

pro / contra / problem bei updates

8.1.4 Pagespeed Mod

8.1.5 Images

5. Most sites fail to leverage best practices for optimizing images. Despite the fact that images represent one of the single greatest performance challenges (and opportunities), 34% of pages failed to properly implement image compression, and 76% failed to take advantage of progressive image rendering. (http://www.radware.com/assets/0/314/6442478110/c810eee1-e86f-438a-b82f-3ad002bf1c75.pdf)

Literatur

[Bar14] Bart. *Where is the best place to put <script> tags in HTML markup?* http://stackoverflow.com/questions/436411/where-is-the-best-place-to-put-script-tags-in-html-markup [Aufgerufen am 08.03.2015]. 2014 (siehe S. 17).

[Goo10] Google. *Using site speed in web search ranking.* Website. 2010 (siehe S. 3).

[Goo11] Google. *Creating Fast Buttons for Mobile Web Applications.* https://developers.google.com/mobile/articles/fast_buttons [Aufgerufen am 03.03.2015]. 2011 (siehe S. 10).

[Gri13a] Ilya Grigorik. *Breaking the 1000ms Mobile Barriere.* https://docs.google.com/presentation/d/1wAxB5DPN-rcelwbGO6lCOus_S1rP24LMqA8m1eXEDRo/present?slide=id.g11c1373c5_3_0 [Aufgerufen am 04.03.2015]. Slides. 2013 (siehe S. 11).

[Gri13b] Ilya Grigorik. *Breaking the 1000ms Mobile Barriere.* https://docs.google.com/presentation/d/1wAxB5DPN-rcelwbGO6lCOus_S1rP24LMqA8m1eXEDRo/present?slide=id.g11c1373c5_5_35 [Aufgerufen am 04.03.2015]. Slides. 2013 (siehe S. 13).

[Gri13c] Ilya Grigorik. *High Performance Browser Networking.* http://tinyurl.com/p5dds9p [Aufgerufem am 02.03.2015. Chapter 2 Slow-Start. 2013 (siehe S. 6, 7).

[Gri13d] Ilya Grigorik. *High Performance Browser Networking.* http://tinyurl.com/lz8t3mh [Aufgerufen am 02.03.2015]. Chapter 10 Speed, Performance, and Human Perception. 2013 (siehe S. 10).

[Gri13e] Ilya Grigorik. *High Performance Browser Networking.* http://tinyurl.com/nojaxxa [Aufgerufen am 02.03.2015]. Chapter 7 Table 7.1. 2013 (siehe S. 12).

[Hol10] Urs Holzle. *Velocity 2010: Urs Holzle.* http://tinyurl.com/px7m64m [Aufgerufen am 27.02.2015]. Video from Velocity Conference. 2010 (siehe S. 3).

[ItW15] ItWissen.info. *Shared Hosting.* http://www.itwissen.info/definition/lexikon/Shared-Hosting-shared-hosting.html [Aufgerufen am 26.02.2015]. 2015 (siehe S. 5).

[Rad13] Radware. *Radware Mobile Infographic.* Website. http://blog.radware.com/wp-content/uploads/2013/11/Radware_SOTU_Fall_2013_Mobile_Infographic_Final1.jpg [Aufgerufen am 15.01.2015]. 2013 (siehe S. 3).

[Rad14] Radware. *STATE OF THE UNION - Ecommerce Page Speed and Web Performance.* http://www.radware.com/assets/0/314/6442478110/c810eee1-e86f-438a-b82f-3ad002bf1c75.pdf [Aufgerufen am 27.02.2015]. Seite 8. 2014 (siehe S. 9).

[Rit14] Michael Ritz. *Weltkarte in Schwarz.* http://www.landkartenindex.de/kostenlos/?p=31 [Aufgerufen am 25.02.2015]. 2014 (siehe S. 7).

[Sch15] Amy Schade. *The Fold Manifesto: Why the Page Fold Still Matters.* Techn. Ber. http://www.nngroup.com/articles/page-fold-manifesto/ [Aufgerufen am 26.02.2015]. Nielsen Normand Group, 2015 (siehe S. 8).

[Ste06] Guido Stepken. *Anti-Pattern in der Softwareentwicklung.* http://www.little-idiot.de/teambuilding/AntiPatternSoftwareentwicklung.pdf [Aufgerufen am 26.02.2015]. 2006 (siehe S. 5).

[Ste08] Stoyan Stefanov. *Exceptional Website Performance with YSlow 2.0.* http://de.slideshare.net/stoyan/yslow-20-presentation [Aufgerufen am 27.02.2015]. Slide Nummer 4. 2008 (siehe S. 8).

[Ste11] Stoyan Stefanov. *Book of Speed*. http://www.bookofspeed.com/chapter3.html [Aufgerufen am 02.03.2015]. siehe Abbildung 3.4 - The-three-way handshake. 2011 (siehe S. 6).

[t3n15] t3n. *Ist deine Website „mobile-friendly"? – Google steigert Druck auf Webmaster*. http://t3n.de/news/google-mobile-friendly-589402/ [Aufgerufen am 25.02.2015]. 2015 (siehe S. 3).

[Tak13] Dean Takahashi. *Apple's iPhone 5 touchscreen is 2.5 times faster than Android devices*. http://venturebeat.com/2013/09/19/apples-iphone-5-touchscreen-is-2-5-times-faster-than-android-devices/ [Aufgerufen am 03.03.2015]. Grafik. 2013 (siehe S. 10).

[TNS14] Google TNS Infratest BVDW. *Global Connected Consumer Study*. Website. http://www.netzproduzenten.de/wp-content/uploads/2014/08/global-connected-consumer-studie-deutschland.pdf [Aufgerufen am 14.12.2014]. 2014 (siehe S. 3, 4).

[Ton13] Rmistry und Tonyg. *Loading measurement: alexa top million netsim*. https://docs.google.com/document/d/1cpLSSYpqi4SprkJcVxbS7af6avKM0qc-imxvkexmCZs/edit [Aufgerufen am 04.03.2015]. 2013 (siehe S. 11).

[Web08] WebSiteOptimization.com. *The Psychology of Web Performance*. http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/ [Aufgerufen am 25.02.2015]. 2008 (siehe S. 3).

[wik14a] wikipedia. *HTTP-Statuscode*. http://de.wikipedia.org/wiki/HTTP-Statuscode [Aufgerufen am 04.03.2015]. 2014 (siehe S. 15).

[wik14b] wiki.ubuntuusers. *Der Benutzer root*. http://wiki.ubuntuusers.de/sudo [Aufgerufen am 26.02.2015]. 2014 (siehe S. 5).

[Wik15] Wikpedia. *Entwurfsmuster*. http://de.wikipedia.org/wiki/Entwurfsmuster [Aufgerufen am 26.02.2015]. 2015 (siehe S. 5).

[wik15] wikipedia. *Content Delivery Network*. http://de.wikipedia.org/wiki/Content_Delivery_Network [Aufgerufen am 04.03.2015]. 2015 (siehe S. 7).