



Dokumentation

LTE MOBILE

Lycée Technique d'Esch-sur-Alzette

T3IF – PROJE – 04.03.2013

Gashi Dren

Dankeswort

Für die Unterstützung zu der Abschlussarbeit „Projet de fin d'études“ der 13ten Klasse möchte Ich mich an dieser Stelle bei folgenden Personen bedanken:

Roger Kries, Informatiklehrer und Betreuer, für die Unterstützung und Hilfe während der gesamten Projektzeit, Bereitstellung des Servers für die Testzwecke, Ideen für die Vorgehensweisen und für die motivierenden Feedbacks.

Maurizio Spagnuolo, Informatiklehrer und Betreuer, der immer zur Stelle war um bei Fehlerdiagnosen zu helfen, Verbesserungsmöglichkeiten vorgeschlagen hat und auch mit Roger Kries Vorgehensweisen empfohlen hat. Danke für die Unterstützung der gesamten Projektarbeit und für die Hilfe für das Aufstellen der AJAX-Routine.

Laurent Haan, Informatiklehrer, für die Zeit, die er sich für mich genommen hat als Ich nicht mehr weiter wusste um nach entscheidenden Fehlern zu suchen und diese zu beheben.

Serge Linckels, Informatiklehrer, der mit einen Lösungsvorschlag zum Layout von untergeordneten verschachtelten Listen gegeben hat, damit diese jeweils ordnungsgemäß angezeigt werden.

Sandrina Fernandes, meiner Freundin, für die Unterstützung zur der Aufstellung der Farben der Web-App und für das Korrekturlesen.

Inhaltsverzeichnis

Dankeswort.....	2
Inhaltsverzeichnis.....	3
Abbildungsverzeichnis	4
Glossar und Abkürzungen.....	5
1. Einführung	6
1.1 Aufgabenstellung	6
1.1.1 Auszug der gestellten Forderungen.....	8
1.2 Erläuterung der entwickelten Lösung.....	8
1.3 Voraussetzungen an den Leser.....	8
2. Beschreibung der Arbeit	9
2.1 Problemstellung.....	9
2.2 Analyse des Problems.....	9
2.3 Konzeption	10
2.3.1 Framework jQuery-Mobile	10
2.3.2 Grundstruktur der Web-App	10
2.3.3 CSS-Quellcode	11
2.3.3.1 Die CSS-Datei	11
2.3.3.2 Generierter CSS.....	12
2.3.4 Server-Side Script	12
2.3.5 Client-Side Script.....	13
2.3.6 Struktur der Konfiguration.....	14
2.3.7 Filteralgorithmus.....	14
2.3.7.1 Zusammenspiel der einzelnen Funktionen	14
2.3.7.2 Prinzip	17
2.3.7.2.1 Detaillierte Erklärung zu einem Teil des Flussdiagramms.....	19
2.4 Schwierigkeiten beim Umsetzen	20
2.4.1 JSON	20
2.4.2 Eigener Callback-Parameter	20
2.4.3 Event-Übergabe per Parameter.....	20
2.4.4 Löschen von Script-Tags.....	21
2.4.5 Abfangen des Back-Buttons	21
2.5 Ausbaumöglichkeiten.....	22
2.5.1 Zurück-Knopf des Browsers abfangen.....	22
2.5.2 Benutzerfreundliche Oberfläche für Konfigurationsdatei	22
2.5.3 Plug-In für Konfiguration.....	22
2.5.4 Erstellen der Seiten anhand des realen HASHs	22
2.5.5 Anpassen von iFrames-Objekten	22
3. Schlussfolgerung.....	24
3.1 Stärken und Schwächen des Produktes.....	24
3.1.1 Synchrones Verfahren bei Seitenanfragen	24
3.1.2 Dynamisches Erstellen / kein Refresh möglich	24
3.2 Meinungsäußerung.....	25
4. Referenzen	26

Abbildungsverzeichnis

Abbildung 1: Diagramm von Mobile Benutzer.....	6
Abbildung 2: Grundprinzip des Projekts.....	9
Abbildung 3: CSS-Style.....	12
Abbildung 4: CSS-generierter Code	12
Abbildung 5: PHP-Script.....	12
Abbildung 6: AJAX-POST-Aufruf.....	13
Abbildung 7: Beispiel eines Menü-Eintrags	14
Abbildung 8: Teil der global einstellbaren Variablen	14
Abbildung 9: Zusammenspiel der einzelnen Funktionen.....	16
Abbildung 10: Filter-Flussdiagramm	18
Abbildung 11: Inhaltsangaben in der Konfigurationsdatei	19
Abbildung 12 : JSON Editor Online	20
Abbildung 13: Verschachtelte Event-Übergabe	21
Abbildung 14: iframe auf einem Smartphone.....	23
Abbildung 15: iframe auf einem Desktop-PC	23
Abbildung 16: CSS-Code zum Anpassen von iframes.....	23
Abbildung 17: Browsen verschiedener Nutzer	24

Glossar und Abkürzungen

Terminologie	Beschreibung
AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
CMS	Content Management System
DOM	Document Object Model
PHP	PHP: Hypertext Preprocessor
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
URL	Uniform Resource Locator (Quellenanzeiger)
XML	Extensible Markup Language

1. Einführung

Ganz klar, das World Wide Web boomt mehr als nie zuvor. Statische, passive Webseiten sind überholt und werden immer seltener. Für eine reine private Webseite kann es zwar noch durchaus angemessen sein, dass sie in HTML und CSS erstellt wurde und keine Interaktivität und Dynamik bietet. Für professionelle Präsenzen und Webangebote mit hohen Besucherzahlen sind solche statische Seiten unangemessen.

Der Trend geht in Richtung Mobilität. Immer mehr Menschen steigen im Alltag, sei es beim Surfen, Email-checken oder Spielen, auf mobile Geräte (Smartphone, Tablet) um. Betrachten wir folgendes Diagramm:

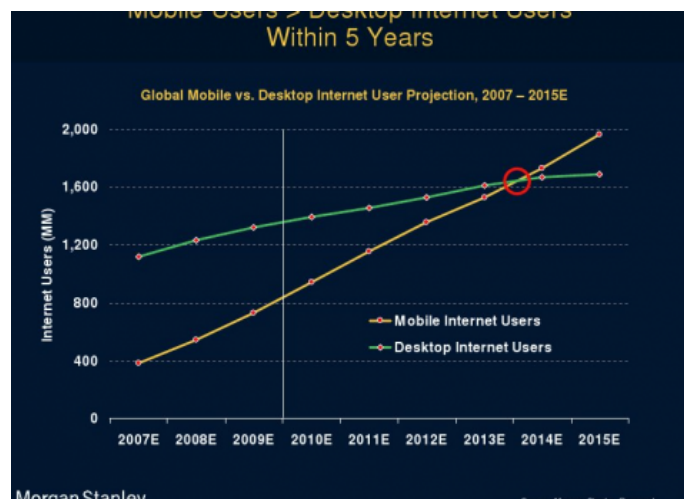


Abbildung 1: Diagramm von Mobile Benutzer

In diesem Diagramm ist klar zu sehen, dass nach Schätzungen zu Folge, im Jahre 2014, die Kennlinie der Anzahl an mobilen Benutzer sich mit deren der Anzahl der Desktop Nutzer überschneidet und diese auch überschreiten wird. Deshalb ist es wichtig, dass immer mehr Unternehmen mobile Applikationen entwerfen.

Die verschiedenen mobilen Betriebssysteme stellen dabei ein kleines Problem, denn Applikationen müssen in verschiedenen Programmiersprachen entwickelt werden. Eine Webseite, die von einem Browser erreichbar ist, jedoch nicht!

Das vorliegende Dokument beschreibt meine Projektarbeit „LTE MOBILE“, einer Web-App für unsere Schulseite, die konservativ in JavaScript, HTML5 und CSS3 erstellt wurde.

1.1 Aufgabenstellung

In Luxemburg ist das Abschlussjahr im technischen Sekundarunterricht die 13te Klasse. In diesem Jahr müssen neben Prüfungen auch Abschlussexamens am Ende des Schuljahres geschrieben werden. In der Technikerausbildung muss davor ein Abschlussprojekt „Projet de fin d'études“ absolviert und bestanden werden damit man zu dem Abschlussexamen zugelassen wird. Das Projekt muss auch schriftlich anhand einer Dokumentation festgehalten werden, welche auch zu einem Teil bewertet wird.

Am Anfang des Schuljahres (Mitte September) präsentieren die Lehrer/Betreuer der Abschlussklasse eine Vielzahl an Projekten, die sich die Schüler auswählen können. Es ist auch

möglich sich ein eigenes Projekt auszudenken, das jedoch einem gewissen Schwierigkeitsgrad entsprechen muss.

Mein Projekt habe Ich aus den Vorschlägen der Lehrer ausgewählt. Es handelt sich dabei um eine mobile Version der bestehenden Schulseite. Die mobile Seite soll eine Webseite sein, die jedoch wie eine „App“¹ wirkt. Alles wird gemäß einem Bildschirm eines Smartphones angepasst. Der Vorteil dieser Lösung gegenüber einer App ist, dass die mobile Webseite betriebssystemunabhängig ist. D.h. sie ist von Smartphones mit verschiedenen Betriebssystemen erreichbar. Nur ein Browser wird benötigt. Da die Daten von der Schulseite abgefragt werden, kann im Endeffekt die App nicht ohne die Schulseite funktionieren. Mit anderen Worten, falls die Schulseite aus irgendeinem Grund nicht erreichbar ist, so kann die mobile Version zwar ansprechbar sein (je nach Lokalität²), aber keine Daten können angezeigt werden.

Die Schwierigkeit dabei ist, dass die mobile Variante dynamisch funktionieren soll. Es wird nicht „hartcodiert“, denn sobald auf der realen Webseite eine Änderung vorgenommen wird, müsste man die Änderung auch auf der mobilen Applikation anwenden. Des Weiteren werde Ich die mobile Seite als App bzw. Web-App bezeichnen, auch wenn es nicht hundertprozentig dasselbe ist, weil diese Bezeichnung kompakter ist.

¹ Applikation für Mobilgeräte

² Falls die App nicht im selben Server liegt wie die Schulwebseite, dann gilt die Problematik nicht

1.1.1 Auszug der gestellten Forderungen

In folgender Tabelle sind die Forderungen der praktischen Arbeit herauskopiert und aufgelistet.

- Einarbeitung in die Entwicklungsumgebung “jQuery Mobile” + “jQuery”
- Filter-Funktion erstellen die mit Hilfe von Konfigurationsdateien an die Webseite (www.lte.lu) angepasst werden kann (keine Daten direkt im Code verwenden)
- Darstellung der Inhalte an mobile Geräte anpassen
 - Android
 - iPhone
 - Optional
 - Tablets
 - iPad

1.2 Erläuterung der entwickelten Lösung

Das Resultat besteht darin, dass die Web-App den Inhalt der Schulseite anfragt, speichert und diesen dann so umgestaltet, damit es angemessen dem Bildschirm des mobilen Gerätes angepasst wird. Bei jedem erneutem Klick, wird der entsprechende Inhalt für die Modifizierung neu geladen und gespeichert. Somit ist sichergestellt, dass die App synchron funktioniert.

Um den Forderungen nachzukommen, wird mithilfe einer Konfigurationsdatei Angaben von Seiten mit Menüs festgehalten, damit nach diesen im angefragten Inhalt der gewünschten Seite, gesucht werden kann. Deshalb ist es theoretisch möglich, eine komplett andere Webseite als die unserer Schule zu nehmen, und die App würde immer noch funktionieren, ohne im Quellcode etwas zu ändern.

1.3 Voraussetzungen an den Leser

Das Projekt ist schlussendlich eine Webseite und deshalb sind Grundkenntnisse über die Grundlagen des World Wide Web wie HTML, CSS, JavaScript und PHP notwendig, um das folgende Kapitel zu verstehen.

2. Beschreibung der Arbeit

In diesem Kapitel werden die Vorgehensweisen, Problemdarstellungen sowie Ausbaumöglichkeiten erläutert. Teil von Quellcodes samt Erklärungen oder Illustrationen sind ebenfalls enthalten.

2.1 Problemstellung

Wie bereits erwähnt soll die App den Inhalt (Daten) der Schulseite abfragen und speichern. Daraufhin muss eine Art Filter erfolgen, damit nur das Einstellbare in der Konfigurationsdatei angezeigt wird.

Probleme dabei bereiten die immer wechselnde Formatierung der „News“-Seite und die Bildergalerien der Schulseite. Deshalb müssen die Filteralgorithmen weitestgehend flexibel sein, denn jeder Inhalt kann anders sein. Eine weitere Hürde ist die Anpassung der Auflösung auf mobile Endgeräte.

2.2 Analyse des Problems

Für das Projekt wird zum größten Teil in JavaScript programmiert. Um das Programmieren zu erleichtern, wird mit einer gratis JavaScript-Bibliothek namens jQuery¹ gearbeitet. Da der Inhalt der Schulseite notwendig ist um eine Manipulation erst möglich zu machen, muss ein ganz kleiner Teil in PHP programmiert werden um diesen auch anzufordern und dann in JavaScript als Objekt abzuspeichern. Dann erst kann man Daten filtern. Die folgende Vorschau erklärt den Weg vom Eingeben der URL in der App bis zur Endgestaltung auf dem mobilen Bildschirm.

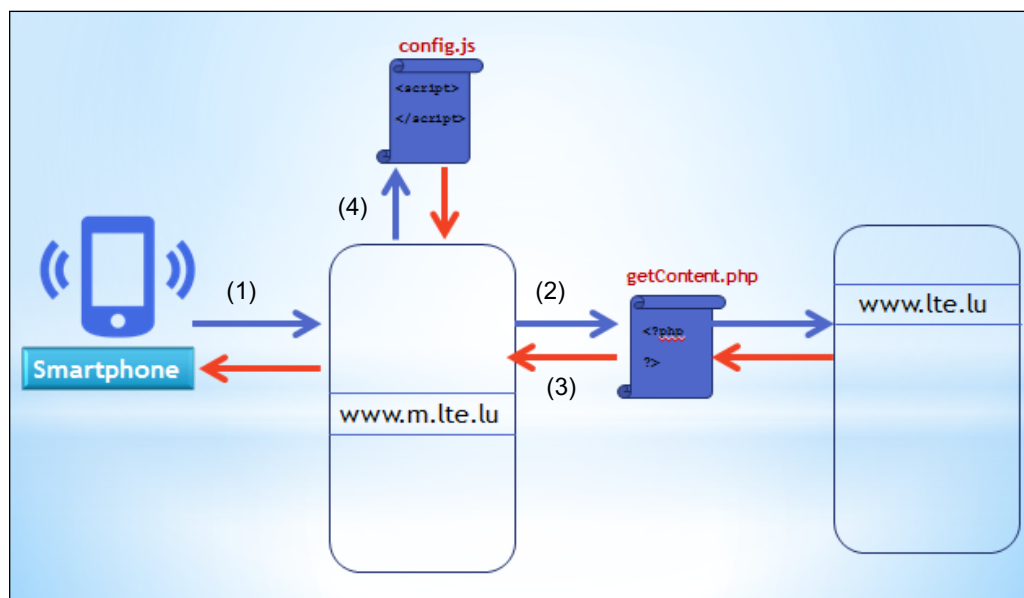


Abbildung 2: Grundprinzip des Projekts

¹ <http://www.jquery.com>

- 1) Der Nutzer eines mobilen Gerätes gelangt auf die App.
- 2) Daraufhin wird ein PHP-Script aufgerufen, die den Inhalt einer als Parameter angegebenen Webseite (String¹) abspeichert. In diesem Beispiel die URL www.lte.lu.
- 3) Sobald dies vollzogen ist, wird der String für das Filtern vorbereitet. Der Inhalt in Form von Text wird in ein DOM geparkt². Dieser Schritt ist notwendig, damit einfacher gefiltert werden kann.
- 4) Die Filterinformationen sind in einer Konfigurationsdatei gespeichert. Nachdem das Notwendige zusammengestellt wurde, wird das Resultat für den mobilen Nutzer sichtbar. Ab diesem Punkt, wird dieser Vorgang jedes Mal wiederholt, wenn der Nutzer auf irgendeinen internen Link klickt. Externe Seiten werden auch extern verwiesen.

Dies ist die Erklärung zur Realisierung des Projektes. Jede Art und Weise kann von dieser Lösung nicht massig abweichen. Grundsätzlich unterscheiden sich Varianten im Teil des PHP-Scripts. Anstatt jedes Mal den ganzen Inhalt der URL (Weblink) anzufordern, wird nur das Notwendige angefordert und angezeigt. Dies würde den JavaScript Code reduzieren, im Gegensatz aber den PHP-Code verlängern. Die Konfigurationsdatei müsste dann auch in ein PHP-verständiges Format abgespeichert werden.

2.3 Konzeption

Die oben beschriebene erste allgemeine Lösung wurde von mir ausgewählt. Der Inhalt wird einfach immer zu 100% abgespeichert, dann erst modifiziert. Somit muss die Konfigurationsdatei mit JavaScript gut auswertbar sein. Dies tut es auch, denn das gewählte Datenformat ist *JSON*. Darin besteht ein großer Vorteil, weil die Schreibweise immer noch dieselbe wie die von JavaScript ist und deshalb nicht geparkt werden muss. Der Nachteil dieser Lösung ist natürlich der entstehende Traffic, aber in Zeiten wie diesen (3G, LTE, N-WLAN) sehe ich dies als nicht besonders problematisch.

2.3.1 Framework jQuery-Mobile

Wie bereits erwähnt, wurde das Projekt mit Hilfe vom JavaScript-Framework³ jQuery Mobile realisiert. Somit wird das Programmieren erleichtert. Nicht umsonst lautet das Motto von jQuery: „Write less, do more!“, was so viel heißt wie „Schreib weniger, bezwecke mehr!“. JQuery Mobile setzt auf jQuery und ist speziell für mobile Webseiten gedacht. Sehr vorteilhaft ist die Unterstützung zahlreicher Systeme und Browser. In manchen älteren Betriebssystemen kann man wegen des Browsers leider nicht alle Eigenschaften von jQuery nutzen, deshalb aber bekommen diese automatisch eine funktionierende Basisversion.

2.3.2 Grundstruktur der Web-App

Eine mobile Seite setzt sich aus einem einspaltigen Layout. Hierbei muss erwähnt werden, dass die Struktur für mobile Geräte mit einem HTML5 „Doctype“ anfangen muss, damit man alle Features vom Framework nutzen kann. Im Head-Bereich wird ein meta-Tag mit folgenden Attributen gesetzt:

¹ Zeichenkette

² Aus Wikipedia, der freien Enzyklopädie:

In der Informatik: Die Zerlegung und Umwandlung einer beliebigen Eingabe in ein für die Weiterverarbeitung brauchbares Format.

³ Rahmenstruktur, Programmiergerüst

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

Dies fordert den Browser auf, die Breite der Seite in die Bildschirmbreite einzupassen, also nicht mit virtuellen Seiten zu arbeiten, sowie die anfängliche Zoom-Stufe auf 1 zu setzen. Die Seiten bestehen jeweils aus <div>-Tags mit der Attribute: data-role="page". HTML5 erlaubt den Nutzern, eigene Attribute zu definieren, die mit dem „data“-Vorsatz gekennzeichnet werden müssen. Dieses Attribut zum Beispiel sorgt dafür, dass der Inhalt innerhalb eines solchen <div>-Tags als eine andere Seite interpretiert wird. Das „data“ im Namen des Attributs hat nichts mit Datenverwaltung oder Ähnlichem zu tun. Es ist die offizielle Empfehlung von HTML5, dass alle Nicht-Standard-Attribute mit „data“ beginnen sollen, damit man sie von anderen unterscheiden kann. Durch einen Verweis auf dem DIV-Container gelangt man zur gewünschten Seite. Binnen einem <div> mit data-role="page" wird in weiter unterteilt mit jeweils anderen data-role-Attributen.

Beispiel:

```
<div data-role="page">
    <div data-role="header">...</div>
    <div data-role="navbar">...</div>
    <div data-role="content">...</div>
    <div data-role="footer">...</div>
</div>
```

Im Header-Bereich sitzen das Logo und der Back-Button. Im Content-Container kommt der Inhalt und im „Navbar“-Bereich die Menüs. Mit dem Attribut „data-position“ bleibt die Leiste immer am oberen/unteren Rand, auch wenn der Anwender den sich darunter befindenden Bereich scrollt.

Für weitere Informationen über die Struktur eines mobilen Gerätes, finden Sie unter folgendem Link:

```
http://jquerymobile.com/demos/1.0a4.1/docs/pages/docs-pages.html
```

2.3.3 CSS-Quellcode

Die Web-App kann auch vom einem PC oder Laptop abgerufen werden. Damit die Ansicht für jedes Gerät optimiert wird, wird der ganze Inhalt mit CSS, prozentual verarbeitet. Feste Werte sind keineswegs gut für die App, denn jeder Bildschirm hat eine andere Auflösung.

In diesem Kapitel wird nur das wichtigste an CSS-Code gezeigt, damit z.B. jeder x-beliebige Inhalt ordnungsgemäß zentriert wird.

2.3.3.1 Die CSS-Datei

Neben den fertigen, extra für mobile Geräte entwickelten CSS-Styles von jQuery, muss ein kleiner Teil, für das Zentrieren, auch selber geschrieben werden. Und zwar der Header- und Inhaltsblock.

```

.content{
  width: 85%;
  margin-right: auto;
  margin-left: auto;
}
#header{
  width: 100%;
  height: 80px;
  line-height: 80px;
  text-align: center;
}

```

Abbildung 3: CSS-Style

Die Breite des DIV-Containers des Inhaltes wird auf 85% gesetzt. Somit entspricht dessen Breite 85% des jeweiligen Bildschirms. Die Abstände (margin) werden auf auto gesetzt, damit diese sich des prozentualen Werts anpassen.

2.3.3.2 Generierter CSS

Ein Teil von CSS, wird vor dem Anzeigen des Inhaltes, generiert. Beispiele hierfür wären die Kopfzeilen eines Inhalts oder das Anpassen der Breite von iFrames oder Tabellen.

Beispiel für Kopfzeilen:

```

function contentHeaderFormatting(content,SymbolANDClassOrID){ // hei gëtt en CSS
  $(content).find(SymbolANDClassOrID).css(
    {
      'font-size' : '22px',
      'color' : '#FF0000',
      'text-decoration': 'underline'
    }
  );
}

```

Abbildung 4: CSS-generierter Code

Die Kopfzeile eines angegeben Inhaltes wird aufgesucht und mit der jQuery Methode .css() formatiert. Dieser CSS-Code kann vom Anwender jederzeit nach eigenen Wünschen gestaltet werden.

2.3.4 Server-Side Script

Der folgende Code besteht darin, dass anhand einer PHP-Funktion, der gesamte HTML(!)-Inhalt einer Webseite angefragt und gespeichert wird.

```

<?php
    $www = $_REQUEST['www'];
    $content = file_get_contents($www);
    $content = preg_replace('#<script(.*)>(.*?)</script>#is', '', $content);
    // http://stackoverflow.com/questions/7130867/remove-script-tag-from-html-content
    echo $content;
?>

```

Abbildung 5: PHP-Script

Erklärung:

- \$www entspricht der URL, die von JavaScript dem PHP-Script übergeben wird
- Anhand der vordefinierten PHP-Funktion `file_get_content()` wird der Inhalt von der als Parameter angegebenen URL abgefragt und in der Variabel `$content` abgespeichert. Der Type der Variabel ist ein String.
- Um Probleme zu verhindern, wird jeglicher JavaScript-Code (erkennbar durch `script-Tags`) aus `$content` anhand Regulären Ausdrücken¹ herausgelöscht.
- Schlussendlich wird das Resultat angezeigt, dies sieht man jedoch nicht, ist aber notwendig, damit JavaScript ein Resultat zurückbekommt. Die Variabel wird so über einen Umweg übergeben.

2.3.5 Client-Side Script

Der PHP-Script wird, wie im vorherigen Kapitel angeführt, mit JavaScript (AJAX), aufgerufen.

```
function getContent(url, callback){
    $.post(
        phpGetContentPath,
        { www: url },
        function(data){
            copyData = data;
            domOBJ = $(copyData);
            initIMG(domOBJ);
            initHREF(domOBJ);
            // 1x optional callback function
            if (callback && typeof(callback) === "function") callback();
            // source : http://www.impressivewebs.com/callback-functions-javascript/
        }
    );
}
```

Abbildung 6: AJAX-POST-Aufruf

Die Abbildung zeigt den ganzen Code der `getContent()`-Prozedur. Die Prozedur ruft die Methode POST auf, und sendet dem PHP-Script die gewünschte URL. Nachdem dieser das Resultat mit `echo` übergeben hat, steht dies in der Variabel `data`. Daraufhin wird der String geparkt und in die globale Variabel `domOBJ` für die erleichterte Manipulation abgespeichert. Zwei Prozeduren folgen, die die globale Variabel nach Links und Bilder (a- und img-Tags) durchsuchen und deren Attribute (`href/src`) in absolute Pfade² umwandeln, d.h. die `defaultURL` aus der Konfigurationsdatei (siehe Kapitel [2.3.6 Struktur der Konfiguration](#)) vor dem Pfad dazu kopiert. Dies ist notwendig, weil die Pfadangaben von internen Links, in der abgefragten Webseite als relative Pfade³ gespeichert sind.

Die Prozedur hat als zweiten Parameter eine optionale `callback`⁴-Funktion. Diese wird erst dann ausgeführt, wenn der vorherige Code vollständig durchgenommen worden ist. Dies ist wichtig, da je nach Traffic, das Anfragen der Webseite zur Zeitverzögerungen führen kann. Ohne diesen

¹ http://www.html-world.de/program/php_art_8.php

² vollständiger Pfad zu einem bestimmten Verzeichnis/Datei

³ Pfad zu einem Verzeichnis/Datei vom momentanen HTML-Dokument aus

⁴ Aus Wikipedia der freien Enzyklopädie:

Eine Rückruffunktion bezeichnet in der Informatik eine Funktion, die einer anderen Funktion als Parameter übergeben wird, und von dieser unter gewissen Bedingungen aufgerufen wird.

Parameter würde in manchen Fällen, die globale Variabel des DOMs nicht richtig aktualisiert werden.

2.3.6 Struktur der Konfiguration

Hauptbestandteil der Konfigurationsdatei ist das JSON-Objekt. Es besteht aus drei Unterobjekten: Menüs, Inhalt und Kopfzeilen des Inhalts.

Jedes dieser Unterobjekte bildet ein Array aus verschiedenen Daten. Die Attribute sind fast immer die gleichen: eine id, class und ein Tag. Diese Informationen genügen um nach spezifischen Inhalten/DOM-Objekten zu suchen.

```
var JSONObj =
{
  "menus":
  [
    //***** 0) navigation *****
    {
      "id": "phoca-topmenu",
      "class": null,
      "tag": "ul li a",
      "icon": "home",
      "text": "Nav",
      "titlePath": "Navigation" // no search result
    },
  ],
}
```

Abbildung 7: Beispiel eines Menü-Eintrags

Weitere Objekt, die jeweils in einer globalen Variabel abgespeichert sind, nennen sich Exclude und AppPrepend. In Exclude können Bilder oder Textpassagen angegeben werden, die nicht angezeigt werden sollen (z.B.: ein Druckerbild). In AppPrepend kann HTML-Code angegeben werden um diesen vor oder nach einem Objekt auf der Hauptseite (DOM) anzuhängen.

Schlussendlich können hier noch Einstellungen, wie URL der Schulseite und dessen Logos und vieles mehr, eingetragen werden. Siehe folgendes Abbild.

```
var defaultURL = 'http://www.lte.lu/';
var logoPath = 'images/logo4.png';
var bgImage = 'images/lte4-2.png';
```

Abbildung 8: Teil der global einstellbaren Variablen

Um genaueres über die Konfigurationsdatei zu erfahren, so schauen Sie im Handbuch „[Handbuch ProjektT3IF 2012-2013 GASHI DREN.doc](#)“ der Web-App nach.

2.3.7 Filteralgorithmus

Die Problematik des Filteralgorithmus ist in kleinere Teile zerlegt, sprich mehrere Funktionen die einen kleinen Teil zum Endprodukt beitragen.

2.3.7.1 Zusammenspiel der einzelnen Funktionen

In folgender Tabelle finden Sie die wichtigsten Funktionen/Prozeduren mit ihren jeweiligen Parameterangaben samt Erklärung:

Funktion	Parameter	Erklärung
getContent	1) url → Type: String 2) callback → Type: Function	<p>Ruft PHP-Script mit AJAX auf und sendet diesem die eingegebene URL. Der PHP-Script gibt den Inhalt zurück und die JavaScript-Prozedur verwandelt das Resultat in einen DOM. Der Callback-Parameter ist eine optionale Funktion, die erst am Schluss ausgeführt wird. Callback wartet bis jeglicher vor ihm stehende Code ausgeführt wurde.</p> <p>Dies ist eine Prozedur und hat keinen Rückgabewert.</p>
createNewPage	Erwartet keine Parameter	<p>Erstellt eine neue Seite anhand der Aktuellen. Die aktuelle Seite ist in einer globalen Variablen gespeichert.</p> <p>Dies ist eine Prozedur und hat keinen Rückgabewert.</p>
loadNextPage	1) evt → Type: Event des Objektes 2) url → Type: String 3) txt → Type: String	<p>Unterbindet den Event¹-Handler² (Bsp.: href-Attribut eines Linkes) und ruft getContent()-Prozedur mit dem href-Attribute (url) des Event-Handlers und die Prozedur createNewPage() auf. Der txt Parameter dient als <i>Caption</i>³ des geklickten Event-Handlers um diesen mit der momentanen Menüposition zu vergleichen und eventuelle Multiklicks zu vermeiden. Schlussendlich wird auf der neue erstellten Seite verwiesen.</p> <p>Dies ist eine Prozedur und hat keinen Rückgabewert.</p>
getMenus	Erwartet keine Parameter	<p>Durchläuft anhand einer Schleife das JSON-Objekt und sucht nach Menüs. Diese werden in einer Liste gespeichert. Jeder Menüeintrag bekommt ein onclick-Attribut indem die Funktion grabMenus mit jeweiligem Parameter aufgerufen wird. Diese Funktion gibt als Resultat</p>

¹ Ereignis in event-gesteuerte Programmiersprachen

² Objekt, das ein bestimmtes Ereignis ausgelöst hat

³ In der Informatik: Beschriftung eines Steuerelements

		einen String aus, der im Inhaltbereich angehängt wird.
grabMenus	1) nr → Type: integer	<p>Der Parameter nr wird anhand der Reihenfolge in der Konfigurationsdatei vergeben. Jetzt wird nach diesem nr-Eintrag gesucht und das Resultat wird anhand einer untergeordneten Liste angezeigt. Deren onclick-Event lautet: loadNextPage.</p> <p>getContent wird hier zum ersten Mal aufgerufen damit der DOM initialisiert wird.</p> <p>Diese Funktion gibt als Resultat einen String aus, der im Inhaltbereich angehängt wird.</p>

Folgende Abbildung stellt noch einmal das Zusammenspiel der einzelnen Funktionen/Prozeduren schematisch dar:

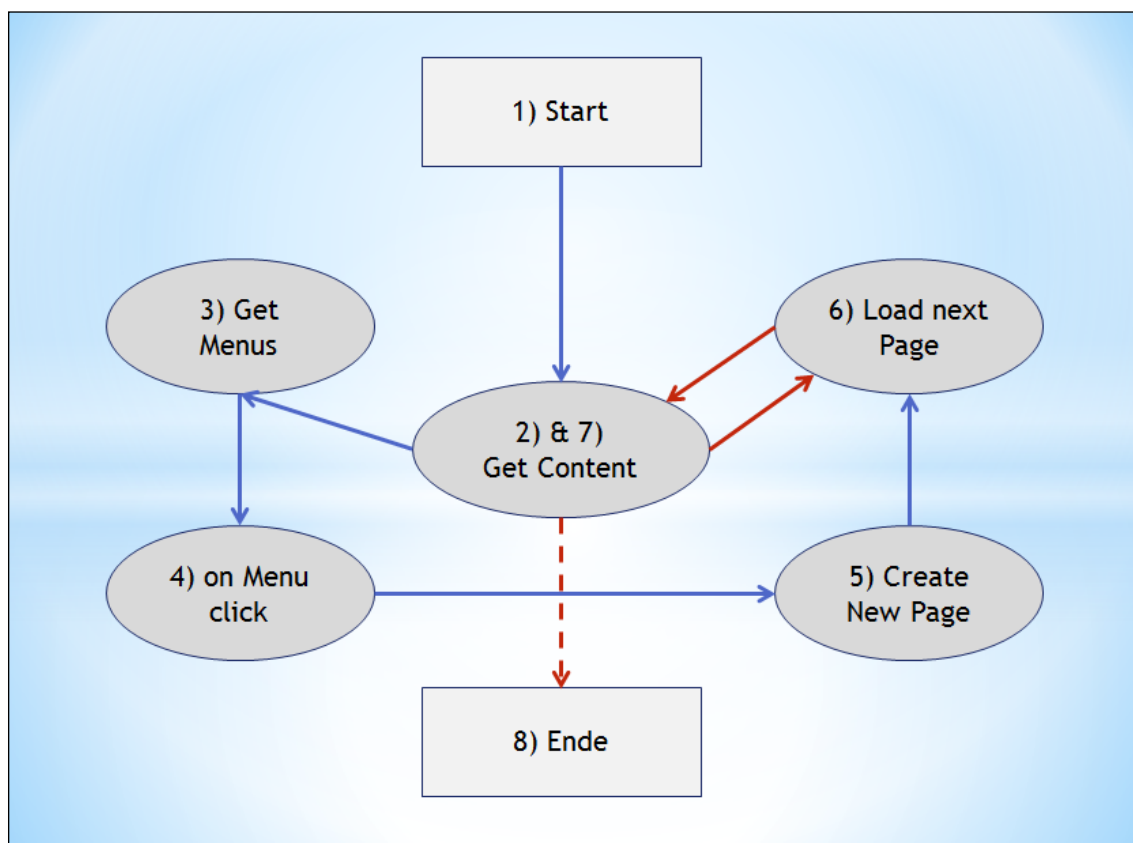


Abbildung 9: Zusammenspiel der einzelnen Funktionen

2.3.7.2 Prinzip

Die oben beschriebenen Funktionen/Prozeduren haben jeweils die gleiche Logik um nach Inhalten zu suchen. Je nach Einsatz werden nur minimale Abweichungen vorgenommen, jedoch ist das Prinzip immer das Gleiche. Erst werden die Suchkriterien anhand der ID oder der Class aus der Konfigurationsdatei zusammengesetzt und dann wird mithilfe der `.find()` Methode aus der jQuery Bibliothek nach diesem Muster gesucht. Während der Filterprozedur wird natürlich schon eine nächste Seite generiert und am Schluss darauf verwiesen.

Das folgende Flussdiagramm stellt das Wichtigste der Filterprozedur dar:

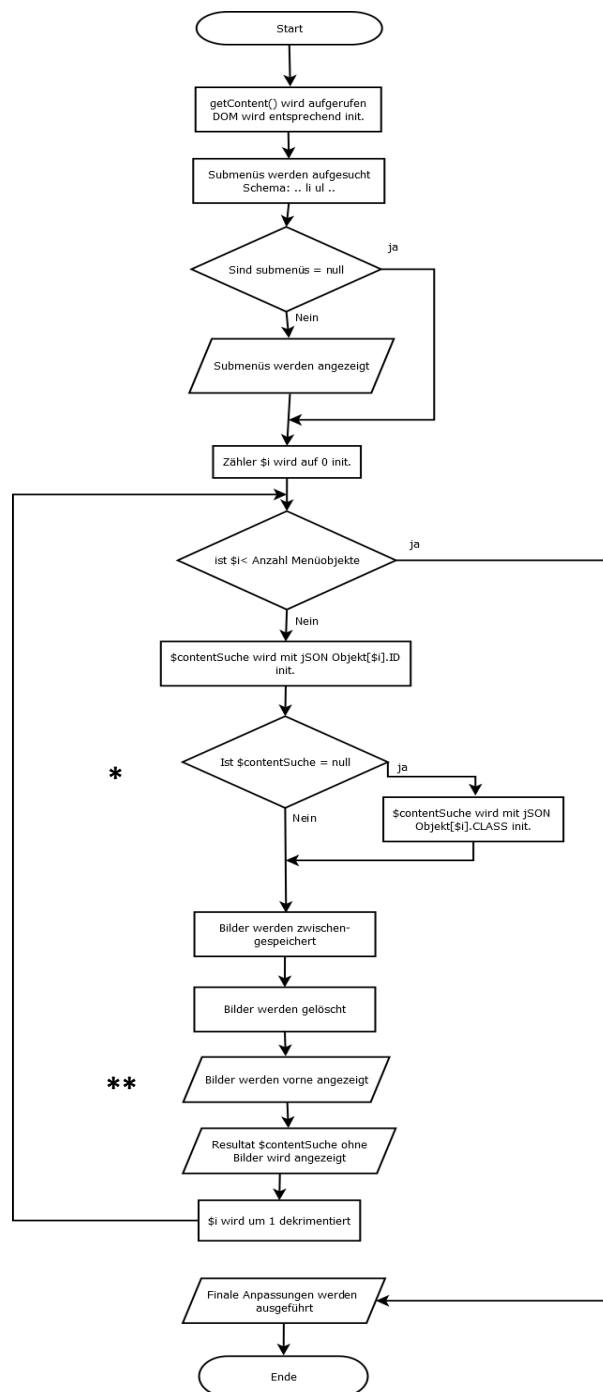


Abbildung 10: Filter-Flussdiagramm

Die Schleife kommt zustande, weil viele Webseiten mit einem CMS ihren Inhalt in mehreren DIV-Containern mit verschiedenen Klassen oder Identifikationsnamen speichern.

Beispiel aus der Schulseite:

¹ Blöcke mit einem „*“ werden jeweils detaillierter erklärt

```

{
  "id": null,
  "class": "contentpaneopen",
  "tag": "table"
},
{
  "id": null,
  "class": "contentpane",
  "tag": "table"
},
{
  "id": null,
  "class": "xsg2",
  "tag": "div"
}

```

Abbildung 11: Inhaltsangaben in der Konfigurationsdatei

In drei verschiedenen Plätzen stehen Inhalte mit einem anderen Identifikationsattribut als normalerweise. Deshalb kann man in der Konfigurationsdatei mehrere Inhaltsangaben speichern. In diesem Beispiel gibt es drei verschiedene Plätze wo die Inhalte stehen, wobei das Letzte für die Galerie ist. Der Algorithmus geht alle Möglichkeiten durch, und speichert diese, falls vorhanden, nacheinander ab.

2.3.7.2.1 Detaillierte Erklärung zu einem Teil des Flussdiagramms

| *

\$ContentSuche ist eine Variabel die mit den Eigenschaften des JSON-Objektes initialisiert wird. Dazu wird ein Muster erstellt, welches bei der Suche nach dem korrekten Inhalt-Block, hilft. Dieses Muster setzt sich aus einer id bzw. Class und einem Tag.

Beispiel:

```
#phoca-topmenu ul li a
```

Bei diesem Beispiel handelt es sich um eine id, erkennbar an der Raute „#“. Falls in der Konfigurationsdatei keine id vorhanden ist, sprich die id den Wert null zugewiesen bekommen hat, wird das Symbol „.“ für die Zusammensetzung der Variabel genommen und es wird das entsprechende Klassenattribut (Class) für das Suchmuster genommen. Die Tags `ul li a` weisen auf einen Link der sich in einer untergeordneten Liste befindet. Je genauer das Suchmuster desto sicherer ist es, dass man das gewünschte Resultat bekommt. Der `a`-Tag befindet sich in einem DIV-Container, dessen id „phoca-topmenu“ ist. In diesem DIV-Tag sind aber untergeordnete Listen, indem der `a`-Tag das letzte „child“ (engl. Kind) bildet, also das letzte untergeordnete Objekt. Das Beispiel kann man mit einem absoluten Pfad vergleichen.

| **

Die Struktur des Inhalts soll immer gleich aussehen. Erst werden Bilder angezeigt, dann erst der Inhalt (Text, Links, Tabellen, etc.). Somit ist es notwendig alle Bilder¹ vom DOM zwischen zu speichern, diese dann zu löschen und den Inhalt, ohne Bilder, anzuzeigen. Als Resultat haben wir nun den reinen Inhalt. Die gespeicherten Bilder werden schlussendlich davor gesetzt.

¹ Mit Bilder sind in HTML die `img`-Tags samt Attributen, gemeint

2.4 Schwierigkeiten beim Umsetzen

2.4.1 JSON

Da Ich zuvor nie etwas über JSON gehört, geschweige denn gesehen/gelesen habe, war dies ein fremdes Wort für mich. Meine Betreuer haben mir dies für die Konfigurationsdatei vorgeschlagen und Ich habe mich sofort darüber im Internet¹ erkundigt und Übungen dazu gemacht. Die JSON-Variabel ist eine Art record, eine Ansammlung von mehreren Werten aus verschiedenen Typen. Viele Versuche scheiterten bis Ich ohne Fehler, das Grundgerüst des JSON-Objektes erstellt hatte. Das Problem war, dass das JSON-Objekt, verschiedene Unterobjekte (z.B. Menüs), die Arrays beinhalten mit wiederum verschiedene Werten. Die Problematik für die Realisation einer effizienten Konfiguration wurde mit dieser Seite <http://www.jsoneditoronline.org/> vereinfacht. Hier kann man rechts in einer Spalte seine Daten eingeben, und links davon wird der JSON-Code erstellt. Daher habe Ich, durch Inspizieren des Codes, das Prinzip schlussendlich verstanden und die gewünschte Struktur hinbekommen.

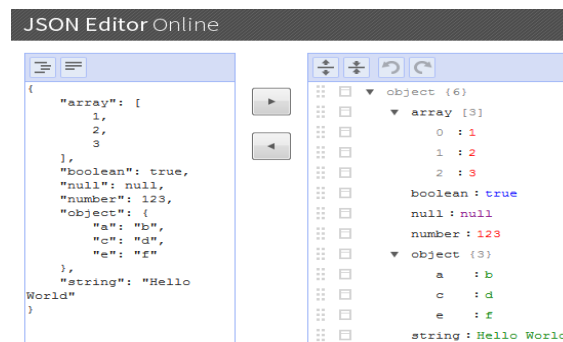


Abbildung 12 : JSON Editor Online

2.4.2 Eigener Callback-Parameter

Es war wichtig, die Funktion `getContent()` zu erweitern um weitere Funktionen nach Ende des Ausführen von `getContent()` aufrufen zu können, damit das DOM richtig initialisiert wird. Siehe Kapitel [2.3.5 Client-Side Script](#) und Abbildung. 6: AJAX-POST Aufruf.

2.4.3 Event-Übergabe per Parameter

In Teil des JavaScript Codes entstanden Probleme bei verschachtelten Funktionen, indem eine (mehrere) Funktion(en) als Parameter einen Event zugeteilt bekommen.

Beispiel (Theorie):

Falls eine Funktion in ihrem Code eine andere Funktion aufruft, die als Parameter einen Event erwartet, dann ist es wichtig diesen Event schon als Parameter in der Funktion davor weiterzugeben, damit die untergeordnete Funktion diese weiter-vergeben oder verarbeiten kann. Sonst wird in JavaScript eine sogenannte „Anonyme JavaScript Funktion“ deklariert. Dies bedeutet, dass die verkettete Funktion zwar einen Event übergeben bekommt, aber diese nicht kennt.

¹ Quelle: <http://www.w3schools.com>

Beispiel (Praxis):

```

$('#content'+nextPageNr.toString())
    .find('a')
    .bind("click",
        function(event){
            if(isInternalLink($(this).attr('href'))){
                loadNextPage(event,$(this).attr('href'),$(this).text()); // Rückkopplung
            }
        });

```

Abbildung 13: Verschachtelte Event-Übergabe

Die Funktion `loadNextPage()` in der oberen Abbildung bekommt den Event des Handlers, in diesem Fall alle `<a>`-Links, die von der Methode `.find()` aufgesucht werden, als Parameter übergeben. Die Funktion ist jedoch verschachtelt im Callback Parameter der Methode `.bind()`. Deshalb muss man schon dort den Event weitergeben, damit `loadNextPage()` diese kennt.

2.4.4 Löschen von Script-Tags

Die Schulseite wird von einem eigenen CMS verwaltet. Dieser generiert JavaScript-Code beim Laden von E-Mail Adressen. Das Problem dabei war, dass der Script ein `document.write()` ausgelöst hat und das Dokument überschrieben hat. Somit wurden Seiten mit E-Mail Adressen (erkennbar an Attribute `href=mailto:[email]`) nicht angezeigt und die Funktion `loadNextPage()` generierte eine fast leere Seite.

Anfangs versuchte Ich die Attribute mit einem `mailto` aufzuspüren und zu löschen. Dies war nicht möglich, da der Script der Schulseite schon alles überschrieben hat. Also musste das Problem im Bereich vom Anfragen des HTML-Codes der Schulseite sein. Sobald ein DOM erstellt wird, ist es schon zu spät, da der Script ausgeführt wurde. Deswegen mussten die Script-Tags im PHP-Script gelöscht werden. Da es zu diesem Zeitpunkt der HTML-Code in Form von String ist, musste eine String-Manipulation her. Dies erreicht man einfach mit Regulären Expressionen (Siehe [2.3.4 Server-Side Script](#)).

2.4.5 Abfangen des Back-Buttons

Die aktuelle Seite wird anhand einer globalen Variabel festgehalten, die je nach generieren einer neuen Seite oder zurückspringen auf einer vorherige Seite, in- oder dekrementiert wird. Um zurück zu browsen, habe Ich einen eigenen Back-Knopf im Header Bereich der Seite implementiert, die nach verweisen auf einer vorherigen Seite, die Variabel mit der aktuellen Seite dekrementiert. Das Problem jedoch ist der Back-Knopf des Browsers (Firefox, Safari, Opera, etc.). Auch nach vielen Versuchen konnte Ich diesen nicht richtig abfangen (reagieren auf dessen Event), um die Variabel zu aktualisieren. Somit bleibt dies ein Manko, denn falls man diesen anstatt den Back-Knopf der mobilen Seite benutzt, wird die aktuelle Seite nicht neu initialisiert und man gelangt nicht mehr zu der Seite, von der man zurück gesprungen ist. Dies ist wegen einem Multi-Klick Algorithmus bedingt, der die momentane Seite mit der neu geklickten Seite vergleicht. Beim Drücken des Back-Knopfes des Browsers, wird neben der Variabel für die aktuelle Seite auch die Variabel für das aktuelle Menü nicht verändert.

Beispiel:

Nehmen wir an man befinde sich gerade auf der dritten Seite (Hauptseite/ Département/ Informatique). Falls man nun wieder auf „Informatique“ klicken möchte, vergleicht der Algorithmus dessen Text mit jenen vom momentanen Menü. Somit ist „Informatique“ nicht neu klick-bar und man verhindert ein unnötiges Generieren einer gleichen Seite. Falls man nun den Zurück-Knopf

der mobilen Version anklickt, wird die globale Variabel für die aktuelle Lage zurück gesetzt. Beim Drücken des Back-Knopfs des Browsers, geschieht dies jedoch nicht. Man befinde sich also in „Département“, die aktuelle Lage ist jedoch „Informatique“ und man kann nicht auf „Informatique“ wechseln. Man müsste einen Zwischenweg nehmen, indem man den Zurück-Knopf der App. benutzt und dann wieder auf den gewünschten Link.

2.5 Ausbaumöglichkeiten

In diesem Kapitel werden verschiedene Ausbaumöglichkeiten sowie Verbesserungen, die zu einem späteren Zeitpunkt implementiert werden können, vorgeschlagen.

2.5.1 Zurück-Knopf des Browsers abfangen

Für Erklärung des Problems, siehe [2.4.5 Abfangen des Back-Buttons](#). Diese Verbesserung müsste noch durchgeführt werden um den Fehler zu beseitigen.

2.5.2 Benutzerfreundliche Oberfläche für Konfigurationsdatei

Die Konfigurationsdatei sollte nur eine in JavaScript erfahrene Person verwalten. Sobald ein Komma oder Ähnliches nicht an der richtigen Stelle ist, funktioniert nichts mehr. Deshalb könnte man eine Oberfläche für nicht-Informatiker entwerfen, inder man seine Daten in Form von TextBoxen angibt und die Konfigurationsdatei im Hintergrund generiert wird. Gute Kenntnisse in JSON sind von Nöten.

2.5.3 Plug-In für Konfiguration

Um die Konfiguration maximal zu erleichtern, könnte man ein Plug-In¹ entwerfen, indem man alle gewünschten Objekte einer fertigen Seite anklickt, und die Konfigurationsdatei würde dann im Hintergrund selber generiert werden mit den Informationen der geklickten Tags.

2.5.4 Erstellen der Seiten anhand des realen HASHs

Für Problematik siehe Kapitel [3.1.2 Dynamisches Erstellen / Kein Refresh möglich](#).

Die Seiten der App werden nach diesem Muster „page/Zahl/“ dynamisch erstellt. Ein Refresh einer Seite ist ohne weiteres nicht möglich. Um dies aber zu ermöglichen, muss der Code zum Generieren einer Seite umgeändert werden. Die App darf nicht die Seiten mit einem string („page“) und einem inkrementierendem Zähler erstellen sondern der HASH² sollte genau derselben Zieladresse des Linkes entsprechen.

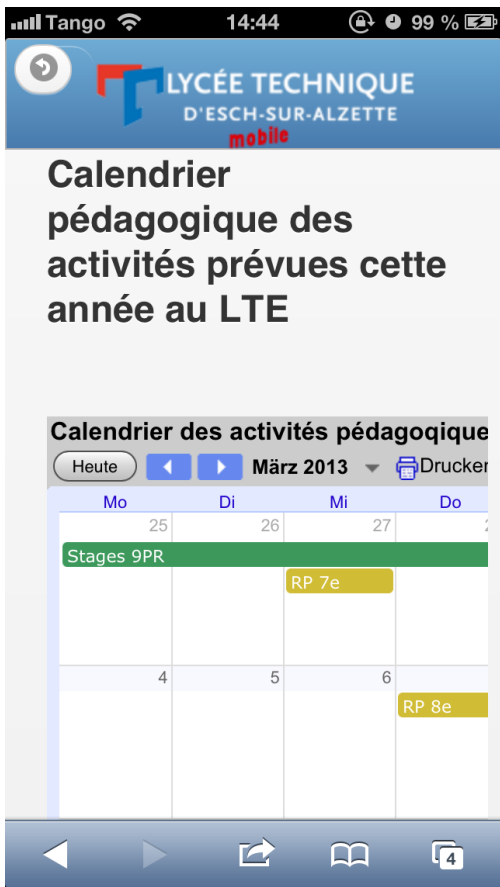
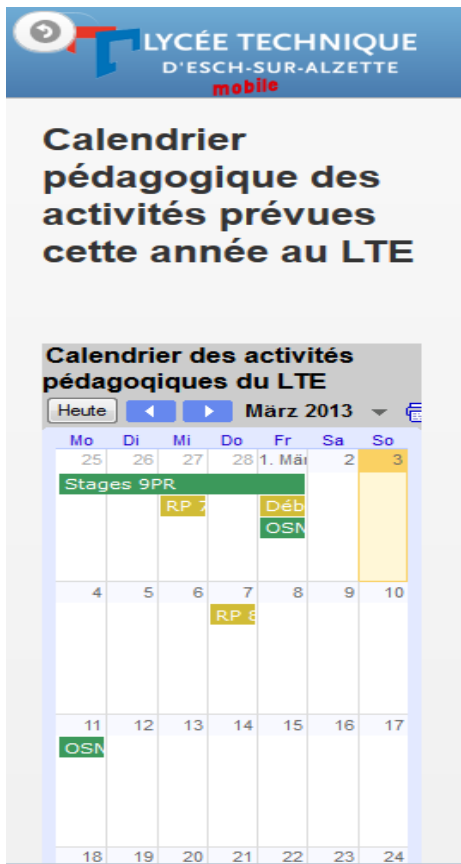
Bei einem Refresh müsste dann nur noch die Funktion loadNextPage(), mit der aktuellen Adresse als Parameter, aufgerufen werden.

2.5.5 Anpassen von iFrames-Objekten

Die Anpassung (Breiteneinstellung) von iFrames-DOM-Objekten, wurde von mir zwar erledigt, aber auf mobile Geräte scheint dies nicht zu funktionieren. Auf einem Browser eines Desktop-PCs schon. Siehe folgende Tabelle zum Vergleich:

¹ Erweiterungsmodul / Softwaremodul

² Teil der URL der mit einer Raute (#) von der restlichen Adresse getrennt ist (interne Seitenverweise).

Im Browser eines Smartphone	Im Browser eines Desktop-PCs
 <p>Abbildung 14: iframe auf einem Smartphone</p>	 <p>Abbildung 15: iframe auf einem Desktop-PC</p>

Code zum Anpassen der Breite:

```
$(contentDIV)
    .find('iframe')
    .removeAttr('width')
    .css('width','100%');
```

Abbildung 16: CSS-Code zum Anpassen von iframes

Der Code sucht nach iframe-Objekten und löscht deren Breiten-Attribut (width). Daraufhin wird den iframe-Objekten ein CSS-Style verpasst, der die Breite auf 100% setzt.

Obwohl es der gleiche Code ist, funktioniert dieser nur beim Desktop-PC.

3. Schlussfolgerung

3.1 Stärken und Schwächen des Produktes

3.1.1 Synchrones Verfahren bei Seitenanfragen

Das Hauptmerkmal der Web-App ist ganz klar das synchrone Verfahren. Bei jedem neuen Klick auf einen internen Link, wird jedes Mal der Inhalt komplett neu angefordert. Wird so zum Beispiel etwas auf der realen Schulseite geändert, so erfolgt die Änderung auch auf der App, sobald man die momentane Seite wechselt. Diese Methode kann man aber auch als Nachteil sehen, denn dies sorgt für hohen Traffic, wobei hoch als relativ anzusehen ist. Bis auf die Bilder, wird der Inhalt schnell geladen. Das Verweisen auf eine neue generierte Seite erfolgt bevor dem Laden der Bilder. Deshalb gelangt man auf eine neue Seite die, falls Bilder enthalten sind, zuerst leer ist und erst nach ein paar Sekunden sieht man den Inhalt. Dieses Problem wurde mit einem Ladezustand behoben. Der Nutzer sieht nun einen Ladekreis bis der Inhalt komplett geladen wurde.

Falls das mobile Gerät in Verbindung mit einem Wireless-Access Point ist, ist die Zeit vom Anfordern der Schulseite bis zur Endgestaltung auf dem Display recht kurz. Bei einer Verbindung mit einem Mobilfunknetz (Bsp.: 3G) kann es zu Verzögerungen von ein paar Sekunden kommen. Wegen den Übertragungsraten ist dies aber je nach Inhalt unvermeidbar.

Deshalb sehe ich das Problem vom Traffic als nebensächlich.

3.1.2 Dynamisches Erstellen / kein Refresh möglich

Die Seiten werden dynamisch, je nach geklickter Link-Adresse, generiert. Deshalb ist es nicht möglich, die aktuelle Seite neu zu laden. Die folgende Illustration soll dies veranschaulichen.

Beispiel:

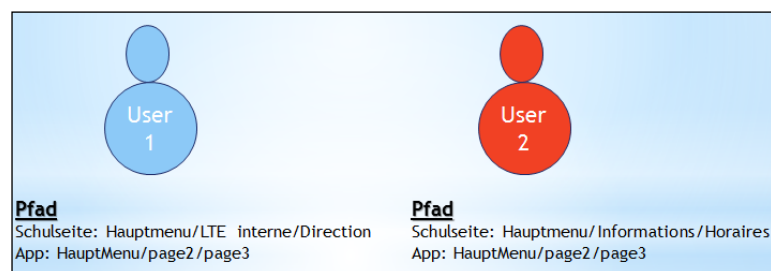


Abbildung 17: Browsen verschiedener Nutzer

User1 und User2 surfen gerade auf der App. User1 befindet sich momentan auf „page3“, diese entspricht „Direction“. User2 befindet sich ebenfalls auf „page3“, jedoch entspricht diese Seite eine andere als jene von User1, weil User2 andere Links geklickt hat. Die Web-App erstellt zwar immer die gleichen Seiten (Bezeichnung: page1, page2,...) aber je href-Attribut eines Links einen komplett anderen Inhalt. Deshalb ist es nicht möglich die aktuelle Seite neu zu laden. Falls man dies versucht, gelangt man zur Startseite. Dies wurde mit Absicht so gehandhabt, um eine leere Seite zu vermeiden.

Da die App nur eine Art Spiegelbild der Schulseite ist, wird das Neuladen einer Seite schwer zu implementieren sein. Das Erstellen einer Seite ist zwar dynamisch, aber nichts desto trotz, sehe ich das nicht mögliche Neuladen als Schwäche des Produktes.

3.2 Meinungsäußerung

Das Projekt habe Ich aus den Vorschlägen der Betreuer ausgewählt, weil das Thema App mich schon seit wenigen Jahren sehr interessiert hat. Leider hatte Ich nie den Willen und Motivation, um eine solche Applikation zu realisieren, denn die Revolution der App hat meiner Meinung nach erst seit einigen Jahren richtig angefangen. Außerdem war Ich auch noch nie im Besitz eines leistungsfähigen Smartphones.

Der Vorteil meines Abschlussprojektes ist die Betriebssystemunabhängigkeit. Ein Vorteil für mich wiederum war die konservative Programmierumgebung. Im Rahmen der vorliegenden Arbeit, wurde meines Erachtens bewiesen, dass die Grundlagen des WWW nicht in Vergessenheit geraten dürfen in puncto „mobile“. Das Zusammenspiel zwischen HTML5, CSS3 und JavaScript kann sehr leistungsstark sein und gerecht mit betriebssystemabhängigen Apps mithalten.

Hinsichtlich der Fragenstellung nach wurde alles erledigt. Es bleiben natürlich Möglichkeiten offen, um das eine oder andere zu verbessern, aber das Ziel der Arbeit ist erreicht.

Im Großen und Ganzen bin Ich stolz auf die Web-App, die wirklich eine Herausforderung war. Während den 5 Monaten Fristzeit vergingen oft Tage, in denen Ich zielstrebig programmiert habe. Während der Projektzeit habe Ich einiges Neues dazugelernt. Unter anderem die neuen HTML5 Attributen, AJAX-Anforderungen und das DOM-Manipulieren.

Neben den Programmierkenntnissen, habe Ich auch dazugelernt, dass man nicht immer versuchen soll, alles alleine zu lösen, so wie Ich es mir angewöhnt hatte. Ohne die Hilfe und Ratschläge der Betreuer und Lehrer, würde das Endprodukt nicht diesem heutigen gleichen.

Insofern steht zu hoffen, dass die erbrachten Kenntnisse mich für die Zukunft begleiten werden.

Gashi Dren, Belvaux, den 4. März 2013

4. Referenzen

a) **Abbildung 1: Diagramm von Mobilen Nutzern**

URL :

<http://blog.mobileroadie.com/2012/03/mobile-versus-pcs-which-smart-device-will-reign-supreme/>

b) **1) Einführung**

Aus dem Vorwort von:

- Ralph Steyer
- „[jQuery, Das JavaScript Framework für interaktives Design](#)“
- Verlag: Addison-Wesley
- ISBN: 978-3-8273-3072-7

c) **2.3.2 Grundstruktur der Web-App**

- Gerhard Völkl
- „[Web-Apps für mobile Geräte](#)“
- Ausgabe: c't kompakt Programmieren 03/2012

d) **2.3.1 Framework jQuery-Mobile**

URL :

<http://www.pc-magazin.de/ratgeber/mobile-webseiten-erstellen-mit-jquery-mobile-1277236.html>

- Dr. Florence Maurice
- 30.04.2012