

2021, Leon Etienne, Issam Charef

# Abhandlung zur Mehrsprachigkeit in der modernen Informatik

*Schwerpunkt: Webentwicklung*

# 1 Inhalt

1

<b>1 Inhalt .....</b>	<b>2</b>
<b>2 Vorwort .....</b>	<b>3</b>
<b>3 Problemstellung .....</b>	<b>3</b>
<b>4 Zielsetzung .....</b>	<b>3</b>
<b>5 Recherche .....</b>	<b>3</b>
5.1 Primärmethoden .....	3
5.1.1 Dynamische Übersetzung .....	3
5.1.2 Statische Übersetzung .....	3
5.2 Ausgeschlossene Problemstellungen .....	4
5.3 Vor- und Nachteile der versch. Methoden .....	5
5.3.1 Dynamische Übersetzung .....	5
5.3.2 Statische Übersetzung .....	5
<b>6 Implementation der Methoden .....</b>	<b>5</b>
6.1 Dynamische Übersetzung .....	5
6.2 Statische Übersetzung .....	6
6.2.1 PHP .....	6
6.2.2 Java oder Kotlin .....	7
6.2.3 JavaScript .....	10
<b>7 Handlungsempfehlungen .....</b>	<b>12</b>
7.1 Design .....	12
7.2 Kodierung .....	13
<b>8 Fazit .....</b>	<b>13</b>
<b>9 Verweise .....</b>	<b>14</b>

## 2 Vorwort

Diese Ausarbeitung entstand im Rahmen des Kursmodules 505 – Programmierung Grafischer Benutzeroberflächen<sup>1</sup> unter Kursführung durch Prof. Dr. Werner König, Hochschule Worms, University of Applied Sciences, in Form der zu erbringenden Prüfleistung. Ziel dieser Arbeit ist es, das Konzept, sowie Lösungsansätze der Mehrsprachigkeit in der Informatik, zu ermitteln, in der Praxis explorativ zu testen und diese Ergebnisse der Gesamtheit des Kursmodules zur Verfügung zu stellen.

## 3 Problemstellung

Die Informatik ist ein internationales Geschehen. Sie steht über jeglicher Landesgrenze, jedoch bestehen immer noch Sprachbarrieren. Informatik ist heutzutage so allgegenwärtig, dass sie fast schon ein Existenzkriterium für jedermann ist. Auch für diejenigen, die kein Englisch sprechen. So steht es an hoher Priorität niemanden zurückzulassen.

## 4 Zielsetzung

Es soll gegeben sein, dass sämtliche linguistikabhängigen Ressourcen sprachübergreifend zur Verfügung stehen. Dies schließt hauptsächlich Text, aber auch Grafiken, sowie Videoquellen ein. Ebenso betroffen sind Hyperlinks, Sprachwiedergaben wie z.B. Podcasts und herunterladbare Dokumente.

## 5 Recherche

In Anbetracht dessen, dass ein Teammitglied im Praxisverbund studiert und im Partnerunternehmen<sup>2</sup> in der Webentwicklungsbranche tätig ist, bietet sich dem Team ein großes Spektrum an state-of-the-art Beispielen<sup>3</sup>, wie professionelle Anwendungsentwickler an die Themenstellung „Mehrsprachigkeit“ herangehen.

### 5.1 Primärmethoden

Grundsätzlich unterteilen sich sämtliche Übersetzungsautomatismen in zwei Klassen: „Dynamische Übersetzung“, und „Statische Übersetzung“.

#### 5.1.1 Dynamische Übersetzung

Die dynamische Übersetzung zeichnet sich weitestgehend dadurch aus, dass sämtliche Inhalte maschinell übersetzt werden. Dies geschieht entweder über in der Applikation eingebettete Funktionsautomatismen, oder durch Dienste Dritter. Ein populärer Anbieter hierbei ist *Google*<sup>4</sup> mit *Google Cloud Translation*<sup>5</sup>.

Es ist nicht ausgeschlossen, dass auch Medien dynamisch übersetzt werden, jedoch ist das eine weitaus anspruchsvollere Problemstellung, die mit großen Parallelen zu maschinellern Lernen verbunden ist. Google bietet bereits diverse Anwendungen an, die Audioübersetzungen in Echtzeit liefern<sup>6</sup> und sogar visuelle Medien übersetzen können<sup>7</sup>.

Die dynamische Übersetzung von (externen) Verlinkungen hingegen erweist sich als große Hürde. Hierfür müsste der angewandte Automatismus nicht nur die Grundessenz der Themenstellung der Webseite verstehen, sondern auch wissen, welche n-sprachige Quelle nun der nativ-sprachigen entsprechen würde.

#### 5.1.2 Statische Übersetzung

Bei der statischen Übersetzung ist jede Sprachvariante einer Quelle bereits im Voraus definiert. Hierfür gibt es verschiedene Möglichkeiten. Vor zehn Jahren (rel. 2021) war es noch allgegenwärtig, dass dieselbe Webseite

---

<sup>1</sup> (Prof. Dr. Herbert Thielen 2021)

<sup>2</sup> (Büro Medienagenten 2020)

<sup>3</sup> (Büro Medienagenten 2021)

<sup>4</sup> (Google LLC 2021)

<sup>5</sup> (Google LLC 2021)

<sup>6</sup> (Google LLC 2021)

<sup>7</sup> (Google LLC 2021)

in statischer Form mehrfach in verschiedenen Sprachvarianten gehostet wird. Heutzutage werden hierzu entweder\* Sprachdateien verwendet, in denen, meist in Form von Key-Value-Pairs (KVP)<sup>8</sup>, lokalisierte Textbruchstücke einem internen Identifier zugeordnet werden. Dort findet ebenfalls die Pflege statt. Diese Textbruchstücke können beliebiges darstellen, da man sie bedingungslos im Seitenquelltext einsetzen darf. So ist es nicht von Relevanz, ob ein KVP nun ein Wort übersetzt, eine angepasste URL zu einer Ressource, oder gar länderspezifische Elemente, wie z. B. ein Cookiebanner, enthält.

\*Oftmals, insbesondere bei Verwendung eines Content Management Systems (CMS)<sup>9</sup>, werden Sprachvarianten in einem Datenbanksystem gespeichert. Das Hauptmerkmal eines CMS ist, dass es eine Benutzeroberfläche bietet, um alle Inhalte, nicht nur, aber auch, die Sprachvarianten zu pflegen. Der große Vorteil hierbei ist, dass es eventuell Fehlermeldungen erheben kann, sollte eine Sprachvariante z. B. noch leer sein. Ebenfalls ist es nicht unüblich, dass CMS die Werte hinter des statischen Übersetzungsautomaten dynamisch generieren kann. Unabhängig davon, ob statisch oder dynamisch übersetzt wird, muss es immer eine Fallback- Sprache oder Ressource geben. Sollte eine Sprachvariante nicht laden, oder das gefragte Element nicht gedeckt sein, sollte zumindest die Weltsprache angezeigt werden.

## 5.2 Ausgeschlossene Problemstellungen

Einige Problemstellungen stellen sich als besonders herausfordernd dar. Diese Ansätze werden in dieser Ausarbeitung nicht näher dargelegt, da sie den Rahmen dieser sprengen würden. Diese sind hauptsächlich, aber nicht ausschließlich:

1. Sprachartefakte, wie in z. B. Isländisch: Hier verändert sich die Schreibweise einzelner Wörter basierend auf dem Rest des Textes<sup>10</sup>
2. Dynamische Übersetzung von Medien
3. Zeit
4. Emoticons

### 5.2.1.1 Sprachartefakte wie in z.B. Isländisch

Manche Sprachen, unter anderem Isländisch, verändern die Schreibweise eines Wortes  $n$ , basierend auf der Bedeutung eines unabhängig platzierten Wortes  $k$ . Dies stellt keine Schwierigkeit bei statischen Übersetzungen festgelegter Texte dar. Sobald der Satz allerdings unbestimmte Inhalte enthält (z.B. Kundennamen), funktioniert eine solche Herangehensweise nicht mehr.

### 5.2.1.2 Dynamische Übersetzung von Medien

Das dynamische Übersetzen von Medien erfordert fortgeschrittene Technologien aus dem Bereich der künstlichen Intelligenz. Bei diesem Vorgang werden beispielsweise Texte auf Bildern durch neue Texte in einer anderen Sprache getauscht, oder Tonaufnahmen in eine Tonaufnahme in einer anderen Sprache übersetzt.

Es gibt existierende Technologien dieser Art, jedoch sind diese zumeist gebührenpflichtig.

### 5.2.1.3 Zeitzonen

Aufgrund dessen, dass verschiedene Orte auf der Welt zum selben absoluten Zeitpunkt verschiedene Daten auf ihren Uhren stehen haben, zeigt sich ein großes Problem für Informatiker. So muss für jede Region eine gegengeebene Uhrzeit angepasst werden.

Viel schwerer fällt ins Gewicht, dass es Schaltjahre gibt und diese schwierige Randfälle eröffnen.

Ebenso haben einige Länder eigene Regeln, die die Komplexität noch weiter erhöhen. Beispielsweise findet die Zeitumstellung in Großbritannien eine Woche früher statt als im Rest Europas<sup>11</sup>, Somalia lässt einen bestimmten Tag pro Jahr wegfallen<sup>12</sup> und das Territorium "West-Bank" hat Israelische, sowie Palästinensische Bürger, die jeweils die Zeitzone ihrer Staatsangehörigkeit verfolgen<sup>13</sup>.

Dieses Thema ist so komplex und tiefgehend, dass es weit über diese Ausarbeitung hinaus geht.

---

<sup>8</sup> (Autoren Wikipedias 2021)

<sup>9</sup> (Autoren Wikipedias 2021)

<sup>10</sup> (Scott, Internationalis(z)ing Code - Computerphile 2014)

<sup>11</sup> (Scott, The Problem with Time & Timezones - Computerphile 2013)

<sup>12</sup> (Scott, The Problem with Time & Timezones - Computerphile 2013)

<sup>13</sup> (Scott, The Problem with Time & Timezones - Computerphile 2013)

#### 5.2.1.4 Emoticons

Emoticons sollen international sein und jegliche Sprachbarriere brechen, da Emoticons auf nonverbaler Kommunikation basieren. Allerdings gibt es auch nonverbale Sprachbarrieren. So bedeutet beispielsweise das Emoticon „👌“ in den vereinigten Staaten und Europa „Alles Okay!“, in Belgien und Tunesien steht es für „Null“ und in Japan für Geld. In Frankreich legt man mit dieser Geste seine Begeisterung am Essen dar und in Italien wird gefragt, wovon der Gegenüber spricht. Häufig stellt diese Geste auch eine obszöne Beleidung dar.<sup>14</sup>

Während Emoticons durchaus in statischen Übersetzungen von Hand gewählt und platziert werden können, sind diese nur sehr schwer dynamisch zu übersetzen und sprengen den Rahmen dieser Ausarbeitung.

## 5.3 Vor- und Nachteile der versch. Methoden

### 5.3.1 Dynamische Übersetzung

Die dynamische Übersetzung ermöglicht es, insbesondere in großen Projekten, mit wenig Aufwand eine enorme Menge an Sprachen abzudecken.

Allerdings ist die Übersetzungsqualität oft mangelhaft und beschränkt sich darauf, dass Benutzer die grundlegenden Informationen erhalten.

### 5.3.2 Statische Übersetzung

Die Implementierung eines statischen Übersetzungsautomatismus ist weitaus unkomplizierter als ein dynamischer. Im Kontrast dazu ist die Pflege eines großen, statisch übersetzten Projektes mit beträchtlichem Aufwand verbunden. Jede sprachabhängige Ressource, auch Text, muss sofort in jeder bereitgestellten Sprache angelegt werden. Änderungen müssen demgemäß in sämtlichen Sprachen erfolgen.

Dem Aufwand entsprechend kann die Übersetzungsqualität tadellos ausfallen. Im Idealfall beauftragt der\*die Redakteur\*in eine\*n Übersetzungsdienstleister\*in und lässt somit professionelle Sprachvarianten anfertigen.

## 6 Implementation der Methoden

### 6.1 Dynamische Übersetzung

Die dynamische Übersetzung kann in verschiedenen Methoden implementiert werden.

Es gibt verschiedene Tools und Frameworks, die diese anbieten. Im Fall der Webentwicklung kann der Google Cloud Translation<sup>15</sup> sehr hilfreich sein, da es einfach implementierbar ist.

Die Implementierung kann durch das Einfügen eines JavaScript<sup>16</sup> Codesegmentes in der Webseite, wo die Mehrsprachigkeit umgesetzt werden soll.

```
<script type="text/javascript">
  function googleTranslateElementInit() {
    new google.translate.TranslateElement({pageLanguage: 'en'}, 'google_translate_element');
  }
</script>
```

---

<sup>14</sup> (Mai 2021)

<sup>15</sup> (Google LLC 2021)

<sup>16</sup> (Pluralsight 2021)

```

<script type="text/javascript">
(function(){var gtConstEvalStartTime = new Date();/*

Copyright The Closure Library Authors.
SPDX-License-Identifier: Apache-2.0
*/

var h=this|self,l=/^[\w+.-]+(?:[0,2])$/m=null,function n(a){return(a=a.querySelector&&a.querySelector("script[nonce]"))&&(a=a.nonce||a.getAttribute("nonce"))&&l.test(a)?a:""}function p(a,b){function c(){}c.prototype=b.prototype;a.i=b.prototype;a.prototype=new c;a.prototype.constructor=a;a.h=function(g,f,k){for(var e=Array(arguments.length-2),d=2;d<arguments.length;d++)e[d-2]=arguments[d];return b.prototype[f].apply(g,e)}}function q(a){return a};function r(a){if(Error.captureStackTrace>Error.captureStackTrace(this,r);else var b=Error().stack&&(this.stack=b))&&(this.message=String(a));(r.prototype.name="CustomError",function u(a,b){a=a.split("%s");for(var c="",g=a.length-1,f=0;g>f;f++)c+=a[f]+(f<b.length?b[f]:"%s");r.call(this,c+e)})(u,r);u.prototype.name="AssertionError",function v(a,b){throw new u("Failure"+(a?" ":"")+a,"").Array.prototype.slice.call(arguments,1)};};var w;function x(a,b){this.g=b==y?a:""}x.prototype.toString=function(){return this.g+""};var y={};function z(a){var b=document.getElementsByTagName("head")[0];(b=document.body.parentNode).appendChild(document.createElement("head"));b.appendChild(a)}function loadJs(a){var b=document;var c="SCRIPT";"application/xhtml+xml"===b.contentType&&(c=c.toLowerCase());c=b.createElement(c);c.type="text/javascript";c.charset="UTF-8";if(void 0===w){w=null;var g=h.trustedTypes;if(g&&g.createPolicy){try{b=g.createPolicy("goog#html",{createHTML:q,createScript:q,createScriptURL:q})}catch(t){h.console&&h.console.error(t.message)}w=b}else w=b}/b.createScriptURL(a);a=new x(a,y);a.{try{var f=c&&c.ownerDocument,k=f&&(f.defaultView?f.parentWindow:k)}l(h);if(k.Element&&k.Location)(var e=k;break a)}catch(t){e=null;if(e&&"undefined"!=typeof e.HTMLScriptElement&&(c instanceof e.HTMLScriptElement)&&(c instanceof e.Location||c instanceof e.Element)))e=typeof c;if("object"===e&&null!=c){if("function"===e)try{var d=c.constructor.displayName||c.constructor.name||Object.prototype.toString.call(c)}catch(t){d="<object could not be stringified>"}else d=void 0===c?"undefined":"null"===c?"null":typeof c;v("Argument is not a %s (or a non-Element, non-Location mock); got: %s",HTMLScriptElement",d)}a instanceof x&&a.constructor===x?d=a.g:(d=typeof a.v("expected object of type TrustedResourceUrl, got '"+a+"' of type '"+("object"!=d?d:Array.isArray(a)?"array":d)+"");d="type_error:TrustedResourceUrl");c.src=d;(d=c.ownerDocument&&c.ownerDocument.defaultView)&&d!=h?d=h:(d=document):(null===m&&(m=(h=document)).d=m);d&&c.setAttribute("nonce",d);z(c)}function loadCss(a){var b=document.createElement("link");b.type="text/css";b.rel="stylesheet";b.charset="UTF-8";b.href=a;z(b)}function isNS(a){a=a.split(".");for(var b=window,c=0;c<a.length;c++){if(!b[b[a[c]])return 1;return 0}function setupNS(a){a=a.split(".");for(var b=window,c=0;c<a.length;c++){b[b[a[c]]]=b[b[a[c]]]={};b[b[a[c]]]={};return b}hasOwnProperty(b).hasOwnProperty(a[c])?b[b[a[c]]]=b[b[a[c]]]={};b[b[a[c]]]={};return b}window.addEventListener&&"undefined"!=typeof document.readyState&&window.addEventListener("DOMContentLoaded",function(){document.readyState="complete"},1);if(!isNS("google.translate.Element")){return}(function(){var c=setupNS("google.translate_const");c.cest=gtConstEvalStartTime;gtConstEvalStartTime=undefined;c.cl="de";c.cuc=googleTranslateElementInit;c.cac="";c.came="";c.ctkk="449881.943577500";var h=translate.googleleapis.com;var s=true?https://window.location.protocol+https://http://://;var b=s+h.c_pah+h.c_pas=c.pbi+b+/translate_static/img/te_bk.gif;c_pci=b+/translate_static/img/te_ctrl.gif;c_pci=b+/translate_static/css/translateelement.css;c_puh=translate.google.com;loadCss(c._ps);loadJs(b+/translate_static/js/element/main_de.js);})();})();

</script>

```

Abbildung 1: Google Translation Service, Einbettungscode<sup>17</sup>

Das Skript soll so aussehen, damit kann nicht viel angefangen werden aber es geht hier hauptsächlich um eine Query Anfrage, die die Sprachtexte von der aktuellen Seite nimmst und im Google Translate Datenbank suchst und die passenden Worte bzw. Sätze übersetzt und dann die Texte zusammenfügt und dann wird das ganze nochmal in unserer aktuellen Seite geschickt und dann wird es in unserer JavaScript Dokument angezeigt. Bevor das alles geschehen wird, der Skript holt von der Datenbank alle Sprachen, die die API anbietet.

Anzumerken ist, dass hierbei jQuery<sup>18</sup> als Abhängigkeit notwendig ist.

Das Anzeigen von allen Sprachen, die Google Cloud Translation anbietet, kann einfach über einen Button angezeigt werden, da wie schon erwähnt, das Skript alle möglichen Sprachen von Googles Datenbank holt. Dieser Button wird automatisch an der benötigten ID `google_translate_element_id` erkannt.

Nach der Implementierung dieses API sollte jetzt die Mehrsprachigkeit funktionieren.

## 6.2 Statische Übersetzung

### 6.2.1 PHP

Die Implementation via PHP ist lediglich eine 18-zeilige Codedatei, die zu Beginn der aufzurufenden PHP-Datei eingebunden wird.

```

1  <?php
2
3      if (isset($_GET["lang"]))
4      {
5          $requestedLang = $_GET["lang"];
6          $local = file_get_contents("./lang/loc/" . $requestedLang . ".json");
7      }
8      else
9      {
10         $local = NULL;
11     }
12
13     // Load fallback language
14     if (!$local)
15     {
16         $local = file_get_contents("./lang/loc/dede.json");
17     }
18     $local = json_decode($local, true);
19 }

```

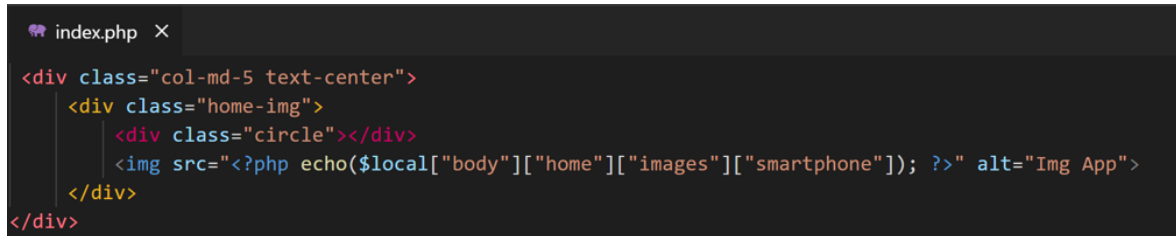
Abbildung 2: loadLang.php des Prüfleistungsprojektes

<sup>17</sup> (Charef 2021)

<sup>18</sup> (OpenJS Foundation 2021)

Hierbei wird vorausgesetzt, dass es Sprachdateien im Verzeichnis `./lang/loc/` gibt, die in JSON<sup>19</sup>-Notation aufgebaut sind. Ebenfalls wird vorausgesetzt, dass die gewünschte Sprache als GET-Parameter `,lang'` übergeben wird. Ist dieser GET-Parameter nicht vorhanden, oder verweist auf eine nicht-existierende Sprachdatei, so wird die in Zeile 15 festgelegte Fallback-Sprachdatei geladen.

Auf die Inhalte dieser Sprachdateien ist nun mit einer Dereferenzierung der globalen Variable `,local'`, (kurz=*locale*) zuzugreifen.



```
index.php X
<div class="col-md-5 text-center">
  <div class="home-img">
    <div class="circle"></div>
    " alt="Img App">
  </div>
</div>
```

Abbildung 3: Zugriff auf Inhalte der Sprachdateien

## 6.2.2 Java oder Kotlin

Die Umsetzung unter Java<sup>20</sup> und Kotlin<sup>21</sup> unterscheidet sich ausschließlich durch das Datenstrukturformat der Sprachdateien. Diese verwendet XML<sup>22</sup>-artige Dateien, die vom IDE (Android Studio beispielsweise) generiert werden können.

Vorgehensweise zu der Implementierung einer statischen Übersetzung mit dem Java/Kotlin:

- Nachdem ein neues Projekt angelegt ist, ist hier beispielsweise ein TextView<sup>23</sup> im Layout zu sehen.



```
<TextView
    android:id="@+id/wtext"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/msg"
    android:layout_marginTop="30sp"
    android:padding="25sp"
    android:layout_below="@+id/willkommen_text"
    android:gravity="center"
    android:textSize="15sp"
    android:textColor="#013d62"
/>
```

Abbildung 4: Android TextView-Element

- Den wichtigsten Elementen, die hier zu betrachten sind, sind:
  - *Android:id*: Die ID wird gebraucht, um den TextView wieder in der Klasse zu finden.
  - *Android:text*: ist zuständig um den Text in dem TextView zu schreiben. Hier ist sehr wichtig, dass der Text nicht direkt im TextView geschrieben wird, sonst kann keine Mehrsprachigkeit implementiert werden.

<sup>19</sup> (ECMA-404 The JSON Data Interchange Standard. kein Datum)

<sup>20</sup> (Oracle 2021)

<sup>21</sup> (Kotlin Foundation 2021)

<sup>22</sup> (Autoren Wikipedias 2021)

<sup>23</sup> (Google LLC 2021)



Abbildung 5: Android Button-Elemente zur Sprachauswahl

- Hier werden die 2 Buttons um zwischen die Sprachen zu wechseln angelegt.
- Hier ist das Attribut *Android:id* wichtig, um die Buttons wieder in der Klasse zu finden.

Bemerkung: *Android:text* ist hier direkt in den Buttons geschrieben, weil es bei allen Sprachen gleichbleiben muss.



```

deutsch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { changeLanguage("de"); }
});

marokkanisch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { changeLanguage("ar"); }
});

```

Abbildung 6: Android ClickListener für Sprachauswahl

- Die Buttons werden hier in unser Klasse aufgerufen und mit der override-Methode *setOnClickListener* implementiert, um das Verhalten nach dem Einklicken zu bestimmen. Anschließend ist eine Funktion *changeLanguage* aufgerufen.

```

public void changeLanguage(String language) {

    if (language == "ar") {
        Toast.makeText(getApplicationContext(), text: "ARABISCH", Toast.LENGTH_LONG).show();
        setLocale("ar");
    }

    if (language == "de") {
        Toast.makeText(getApplicationContext(), text: "DEUTSCH", Toast.LENGTH_LONG).show();
        setLocale("de");
    }

}

```

Abbildung 7: Android changeLanguage-Funktion

- Die Funktion *changeLanguage* prüft, welche Sprache gespeichert werden soll, um die Sprache zu laden. Diese wird durch eine andere Methode zum Laufen gebraucht.

```

public void setLocale(String lang) {
    Locale myLocale = new Locale(lang);
    Resources res = getResources();
    DisplayMetrics dm = res.getDisplayMetrics();
    Configuration conf = res.getConfiguration();
    conf.locale = myLocale;
    res.updateConfiguration(conf, dm);
    Intent refresh = new Intent( packageContext: this, MainActivity.class);
    finish();
    startActivity(refresh);
}

```

Abbildung 8: Android setLocale-Funktion

- Die Funktion *setLocale* bekommt als Parameter die gewünschte Sprache, dann wird ein neues Objekt von der Klasse *Locale* erzeugt.  
In Java/Kotlin repräsentieren Locale-Objekte geografische, politische oder kulturelle Regionen. Die Sprache und die Region müssen getrennt werden, denn nicht immer gibt eine Region oder ein Land die Sprache eindeutig vor. Danach wird die Konfiguration unseres Projektes mit der neuen Sprachdatei aktualisiert.  
*New Intent* ist die Klasse, um nochmal die *main* Aktivität zu aktualisieren.

## 6.2.3 JavaScript

### 6.2.3.1 Konzept

Die Implementation durch JavaScript ist ebenso trivial wie elegant. Das Konzept basiert darauf strukturierte Daten in folgendem Format anzulegen:

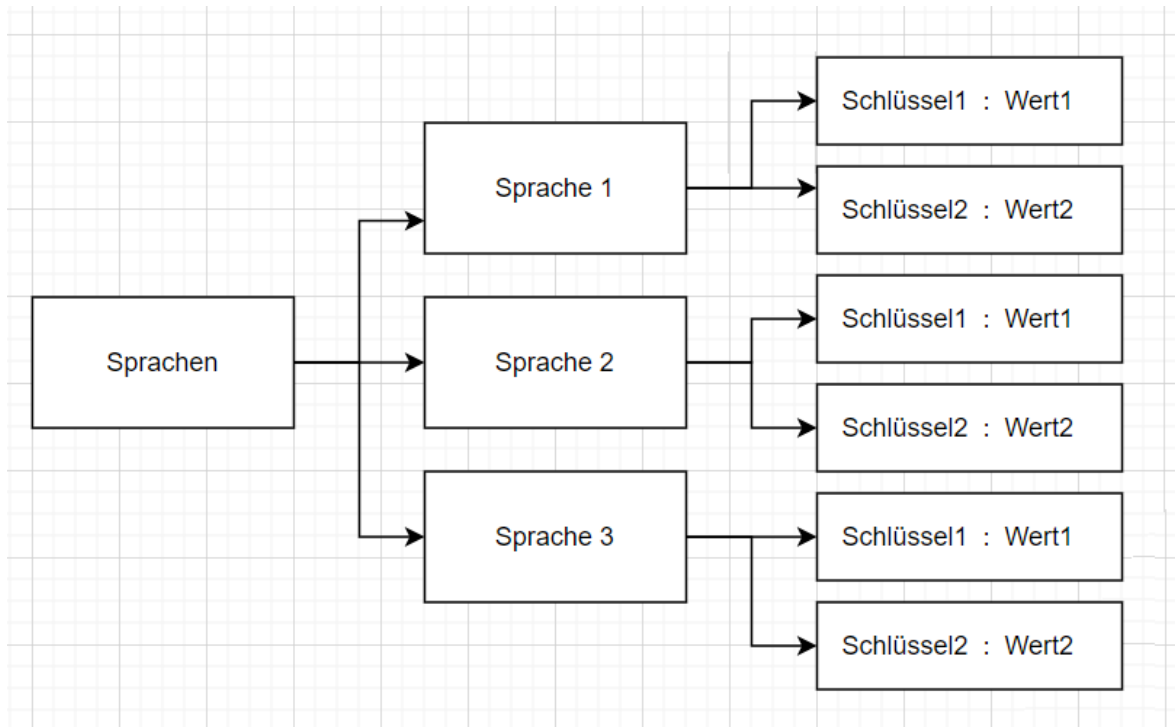


Abbildung 9: Diagramm der strukturierten Daten der JavaScript-Implementation

Hierfür bietet sich JSON-Notation an. Anschließend wird im Quelltext der Webseite lediglich das im Rahmen dieses Frameworks implementierte Attribut `langref` (language reference) definiert, welches ein DOM-Element mit einem Spracheintrag verknüpft. Diese Elemente bekommen somit den in den Sprachdateien definierten `innerHTML`-Wert zugewiesen.

```
<h2 langref="headline"></h2>
<p langref="bodytext"></p>
```

Abbildung 10: Das `langref`-Attribut

### 6.2.3.2 Implementierung

Ähnlich wie in PHP werden die Sprachdateien in JSON-Notation in einem Sprachverzeichnis erstellt. Jedoch geschieht das nicht in json-Dateien, sondern in JavaScript-Dateien. Grund dafür ist, da somit die Sprachdaten im Header der Webseite geladen werden können und somit bereits zur Verfügung stehen, bevor der Nutzer die Webseite sieht. Somit wird vermieden, dass die Webseite kurzzeitig ohne Sprachinhalte gezeigt wird. Das setzt voraus, dass es bereits ein Hauptobjekt (Abbildung 9, 'Sprachen') gibt, in dem die Sprachen definiert werden können.

Letztlich definieren wir die zwei Funktionen `GetLocalizedContent(language, langref)` und `UpdateLocalization(language)`.

```
function GetLocalizedContent(language, langref) {
    return lang[language][langref];
}
```

Abbildung 11: Implementation *GetLocalizedContent*

```
function UpdateLocalization(language) {
    $('*[langref]').each(function(i, v) {
        $(v).html(
            GetLocalizedContent(language, $(v).attr("langref"))
        );
    });
}
```

Abbildung 12: Implementation *UpdateLocalization*

*GetLocalizedContent* bekommt einen Sprachidentifizier und einen Schlüssel. Zurückgegeben wird der dem Schlüssel zugehörige Wert der jeweiligen Sprache.

*UpdateLocalization* erhält lediglich einen Sprachidentifizier, geht über jedes Element mit einem Attribut *langref* und setzt dessen inneres HTML zu dem Wert zugehörig zu der jeweiligen Sprache und dem Schlüssel angegeben als Wert von *langref*.

Sobald das DOM aufgebaut ist, wird direkt *UpdateLocalization* mit der Standardsprache aufgerufen, um die mit *langref* markierten Elemente zu befüllen. Hierfür bietet sich das *defer*-Attribut<sup>24</sup> an. Wird die Sprache geändert, so muss lediglich die globale Funktion *UpdateLocalization* mit dem gewollten Sprachidentifizier als Parameter erneut aufgerufen werden.

Abschließend noch Einblicke in relevante Code-Dateien:

```
<> index.html X
<> index.html > html > body > button#langctrl_de
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title langref="title"></title>
8      <script src="jquery.js"></script>
9
10     <script src="lang/master.js"></script>
11     <script src="lang/dede.js"></script>
12     <script src="lang/enus.js"></script>
13     <script src="lang/localization.js" defer></script>
14 </head>
```

Abbildung 13: Scriptladereihenfolge im Seitenheader

```
JS master.js X
lang > JS master.js
1      lang = {};
2
```

Abbildung 14: Erstellung des Hauptobjektes

<sup>24</sup> (w3schools kein Datum)

```

JS dede.js  X
lang > JS dede.js > ...
1  lang["dede"] = {
2      title: "hallo",
3      headline: "Ich bin eine Überschrift",
4      bodytext: "Und ich bin der Text",
5  }
6

```

Abbildung 15: Beispiels-Sprachdatei

## 7 Handlungsempfehlungen

### 7.1 Design

Nach der Implementierung der Mehrsprachigkeit werden sich einige Probleme bezüglich des Designs der Applikation herausstellen.

Aus Erfahrung wird folgendes nahegelegt:

- CSS-Frameworks, wie z.B. Bootstrap<sup>25</sup>, Foundation<sup>26</sup> und Semantic<sup>27</sup>, da die Entwickler an diese Probleme gedacht haben.
- *Height* muss am besten *auto* im Element verwenden, da die Länge von Worten/Sätzen in anderen Sprachen verschiedene Länge hat.
- Schriftfamilien spielen eine große Rolle. Man sollte sicherstellen, dass sämtliche verwendeten Schriftfamilien alle verwendeten Zeichensätze unterstützen.  
Google Fonts<sup>28</sup> ist beliebt und Zeichensatzabdeckung ist trivial zu überprüfen.
- Medien, die nur für eine einzige Lokalisierung relevant sind, gelten generell als zu vermeiden.
- Für eine Applikation, die Sprachen von rechts nach links (z.B. Arabisch) oder von oben nach unten unterstützen soll, sollte man die Styles in sprachspezialisierte Stylesheet-Dateien auslagern. Das kann auch bei der Weiterentwicklung sehr hilfreich sein.

---

<sup>25</sup> (Bootstrap Team 2021)

<sup>26</sup> (ZURB Inc 2021)

<sup>27</sup> (Semantic UI LLC 2021)

<sup>28</sup> (Google LLC 2021)

## 7.2 Kodierung

- Die Trennung von Struktur und Logik mittels MVC Pattern<sup>29</sup> wird bei der Mehrsprachigkeit generell nahegelegt.  
empfohlen.

## 8 Fazit

Es gibt viele verschiedene Möglichkeiten Mehrsprachigkeit in der Informatik anzuwenden. Jede dieser hat ihre eigenen Vor- und Nachteile. Somit ist es Situationsabhängig, welche dieser man nun anwenden möchte. Es ist nicht einmal generalisierbar, ob eine Anwendung überhaupt mehrsprachig sein sollte. Es lässt sich lediglich sagen, dass die meisten mit Kunden interagierenden Applikationen mehrsprachig sein sollten und den meisten dieser eine statische Übersetzung nahezulegen ist.

---

<sup>29</sup> (Tutorials Point 2021)

## 9 Verweise

- Autoren Wikipedias. *Wikipedia - Attribute-value pair*. 23. März 2021. [https://en.wikipedia.org/wiki/Attribute%E2%80%93value\\_pair](https://en.wikipedia.org/wiki/Attribute%E2%80%93value_pair) (Zugriff am 27. Mai 2021).
- . *Wikipedia - Content management system*. 19. Mai 2021. [https://en.wikipedia.org/wiki/Content\\_management\\_system](https://en.wikipedia.org/wiki/Content_management_system) (Zugriff am 27. Mai 2021).
- . *XML - Wikipedia*. 2021. [https://en.wikipedia.org/wiki/XML#:~:text=Extensible%20Markup%20Language%20\(XML\)%20is,free%20open%20standards%E2%80%94define%20XML](https://en.wikipedia.org/wiki/XML#:~:text=Extensible%20Markup%20Language%20(XML)%20is,free%20open%20standards%E2%80%94define%20XML). (Zugriff am 16. Juni 2021).
- Bootstrap Team. *Bootstrap, the most popular HTML, CSS and JS library in the world*. 2021. <https://getbootstrap.com/> (Zugriff am 16. Juni 2021).
- Büro Medienagenten. *Büro Medienagenten - Startseite*. 2020. <https://www.medienagenten.de/> (Zugriff am 26. Mai 2021).
- . *Referenzen der Medienagenten*. 2021. <https://www.medienagenten.de/referenzen> (Zugriff am 26. Mai 2020).
- Charef, Issam. „Screenshot Google Cloud Translation Embedding Code Snippet.“ 10. Juni 2021. *ECMA-404 The JSON Data Interchange Standard*. kein Datum. <https://www.json.org/json-en.html> (Zugriff am 16. 07 2021).
- Google LLC. 2021. <https://fonts.google.com/> (Zugriff am 16. Juni 2021).
- . *Google*. 2021. [www.google.de](http://www.google.de).
- . *Google Cloud Translation*. 2021. <https://cloud.google.com/translate> (Zugriff am 26. Mai 2021).
- . *Google Translate Help - Translate Images*. 2021. <https://support.google.com/translate/answer/6142483?hl=en> (Zugriff am 26. Mai 2021).
- . *TextView | Android Developers*. 2021. <https://developer.android.com/reference/kotlin/android/widget/TextView> (Zugriff am 16. Juni 2021).
- Kotlin Foundation. *Kotlin Programming Language*. 2021. <https://kotlinlang.org/> (Zugriff am 16. Juni 2021).
- Mai, Jochen. *Karrierebibel*. 04. Juni 2021. <https://karrierebibel.de/handzeichen-gesten-ausland/> (Zugriff am 11. Juli 2021).
- OpenJS Foundation. *jQuery*. 2021. <https://jquery.com/> (Zugriff am 16. Juni 2021).
- Oracle. *Java | Oracle*. 2021. <https://www.java.com/de/> (Zugriff am 16. Juni 2021).
- Pluralsight. *JavaScript.com*. 2021. <https://www.javascript.com/> (Zugriff am 16. Juni 2021).
- Prof. Dr. Herbert Thielen, Prof. Dr. Zdravko Bozakov. „Modulhandbuch-AnInf-2021.pdf.“ *Hochschule Worms*. 24. Mai 2021. <https://www.hs-worms.de/fileadmin/media/fachbereiche/informatik/AInf/Modulhandbuch/Modulhandbuch-AnInf-2021s.pdf>.
- Scott, Thomas. *Internationalis(z)ing Code - Computerphile*. Webvideo, University of Nottingham: Computerphile, 2014.
- Scott, Thomas. *The Problem with Time & Timezones - Computerphile*. Webvideo, University of Nottingham: Computerphile, 2013.
- Semantic UI LLC. *Semantic UI*. 2021. <https://semantic-ui.com/> (Zugriff am 16. Juni 2021).
- Tutorials Point. *Design Patterns - MVC Pattern*. 2021. [https://www.tutorialspoint.com/design\\_pattern/mvc\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm) (Zugriff am 16. Juni 2021).
- w3schools. kein Datum. [https://www.w3schools.com/tags/att\\_script\\_defer.asp](https://www.w3schools.com/tags/att_script_defer.asp) (Zugriff am 18. 07 2021).
- ZURB Inc. *The most advanced responsive front-end framework in the world*. 2021. <https://get.foundation/> (Zugriff am 16. Juni 2021).

Abbildung 1: Google Translation Service, Einbettungscode .....	6
Abbildung 2: loadLang.php des Prüfleistungsprojektes.....	6
Abbildung 3: Zugriff auf Inhalte der Sprachdateien.....	7
Abbildung 4: Android TextView-Element.....	7
Abbildung 5: Android Button-Elemente zur Sprachauswahl .....	8
Abbildung 6: Android ClickListener für Sprachauswahl .....	9
Abbildung 7: Android changeLanguage-Funktion .....	9
Abbildung 8: Android setLocale-Funktion .....	9
Abbildung 9: Diagramm der strukturierten Daten der JavaScript-Implementation .....	10
Abbildung 10: Das langref-Attribut.....	10
Abbildung 11: Implementation GetLocalizedContent .....	11
Abbildung 12: Implementation UpdateLocalization.....	11
Abbildung 13: Scriptladereihenfolge im Seitenheader .....	11
Abbildung 14: Erstellung des Hauptobjektes .....	11
Abbildung 15: Beispiels-Sprachdatei .....	12