Hochschule der Medien Stuttgart  
Bibliotheks-und Informationsmanagement (Bachelor)  
Sommersemester 2016

**Dokumentation**

**„Studentenfutter“**

Einführung in das Programmieren

Prof. Magnus Pfeffer

**Vorgelegt von: Abgabedatum:**

Lara-Kristina Herbrich 29.06.2016

lh052@hdm-stuttgart.de

Matrikelnummer: 29841

Maike Weickart

mw181@hdm-stuttgart.de

Matrikelnummer: 29898

Inhaltsverzeichnis

[Abbildungsverzeichnis 2](#_Toc454888015)

[1. Idee 3](#_Toc454888016)

[2. Aufbau der App 4](#_Toc454888017)

[2.1 Startseite 4](#_Toc454888018)

[2.2 Rezepte 5](#_Toc454888019)

[2.3 Einkaufsliste 6](#_Toc454888020)

[2.4 Formular 6](#_Toc454888021)

[3. Umsetzung in JavaScript 7](#_Toc454888022)

[3.1 Datenstruktur des RezeptBuches 7](#_Toc454888023)

[3.2 Abspeichern der Einkaufsliste 9](#_Toc454888024)

[3.3 Löschen der Einkaufsliste und des lokalen Speichers 9](#_Toc454888025)

[3.4 Timer 10](#_Toc454888026)

[4. Aussicht 11](#_Toc454888027)

[Quellenverzeichnis 12](#_Toc454888028)

[Textverzeichnis 13](#_Toc454888029)

[Bilderverzeichnis 13](#_Toc454888030)

# Abbildungsverzeichnis

Erste Skizze 3

Screenshot Menü 4

Screenshot Rezepte 5

Screenshot Einkaufsliste 6

Screenshot Formular 6

Screenshot Datenstruktur 7

Screenshot der Pfeilfunktionen 8

Screenshot Zählerfunktion 10

Screenshot Einblendung des Zählerbuttons 10

# Idee

Im Folgenden wird das im Zuge des Wahlpflichtfaches „Einführung in das Programmieren“ entstandene Projekt dokumentiert. Angestrebt wurde dabei die Entwicklung einer Webapplikation, in der Rezepte verzeichnet sind, eine Einkaufsliste integriert ist und über eine Stoppuhrfunktion verfügt.

Die unten gezeigte Grafik ist ein erster Entwurf des Aussehens der App. Jede der Spalte stellt dabei eine Seite der App dar. Vorgesehen waren: eine Seite die bei Starten der App aufgerufen wird, eine Seite auf der die Rezepte abrufbar sind, eine Seite in der die Einkaufsliste abrufbar ist, sowie eine Seite auf der die Stoppuhr aufgerufen werden kann. Durch Wischen mit dem Finger über den Bildschirm soll der Nutzer zwischen den Seiten wechseln können

|  |  |  |  |
| --- | --- | --- | --- |
| START  Rezept  Einkaufsliste  Stoppuhr  🡪 Links | REZEPT  Bild  Zutaten (Person)  Beschreibung | EINKAUFEN  Zwiebel  300g Hackfleisch  1kg Tomaten | Stoppuhr  Zähler ist gestellt  Am Ende ertönt Klingel |

Sliden zur nächsten Seite

Abb. 1: Erste Skizze

Die im Rezept ersichtlichen Zutaten, sollen durch klicken eines Buttons in die Einkaufsliste hinzugefügt werden können. In dieser soll die Möglichkeit bestehen einzelne Listenpunkte wieder zu entfernen oder die ganze Liste zu löschen. Die Stoppuhrfunktion soll es dem Nutzer ermöglichen während des Kochens einen Timer zu stellen um Koch- und Backzeiten im Blick zu behalten. Diese Funktion wurde später in die Rezepte eingebunden und wird im späteren Verlauf der Dokumentation genauer erläutert. Ein Formular zum selbständigen Hinzufügen von Rezepten ersetzt die Seite des Timers. Auch die Rezepte sollen über eine Funktion besitzen, mit der Notizen und Anmerkungen abgespeichert werden können.

Um die Vorstellungen in die Tat umzusetzen, musste sich vorerst mit einigen Themen beschäftigt werden. Wichtig waren bei der Umsetzung des Projekts: die Objektorientierte Strukturierung der Daten und deren Abspeicherung. Aus diesem Grund wurde die Dokumentation von JSON vor Beginn des Programmierens gelesen. Bei der Umsetzung dieser Datenstruktur gab es jedoch zu Beginn Probleme, welche unter Hilfe von Herr Pfeffer gelöst werden konnten. Im Folgenden werden der Aufbau der App, die Umsetzung in JavaScript, sowie die Aussicht der App bei weiter Bearbeitung in dieser Dokumentation erläutert.

# Aufbau der App

Die Struktur der Anwendung besteht aus mehreren Funktionen, die für das Anzeigen der Rezepte, die Einkaufsliste, sowie des Formulars zuständig sind. Des Weiteren befinden sich im Quellcode der rezeptBuch.js Funktionen die die Nutzung des Timers ermöglichen. Im Folgenden werden die Bestandteile und Vorgehensweisen dieser Bestandteile erläutert und erklärt. Der Javascript-Code der dies beinhaltet, ist in einem HTML-Code integriert, dieser ermöglicht, dass die App im Browser abgerufen und genutzt werden kann.

Die im Folgenden abgebildeten Screenshots zeigen wie die App auf einem Apple iPhone 6s visualisiert werden würde.

## Startseite

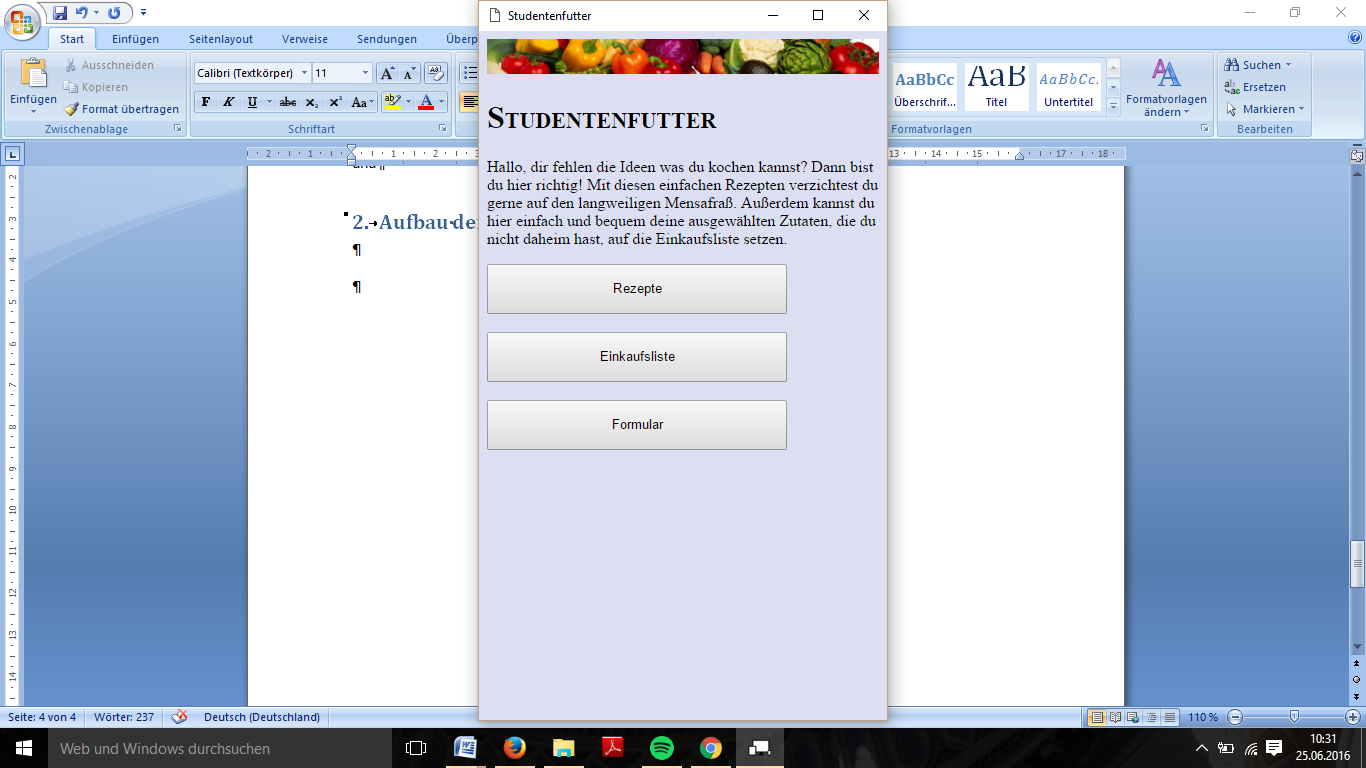
Die Startseite zeigt drei Buttons mit denen man zu den verschiedenen Bereichen der Anwendung gelangen kann. Durch das Hinzufügen von <body onload=“zeigeMenue()“> in den HTML-Code, wird gewährleistet, dass beim Öffnen der App immer das Menü angezeigt wird. Die function zeigeMenue() ist dabei dafür zuständig, dass das Menü gezeigt werden kann. Durch den Befehl „document.getElementById('display') .innerHTML = myString;“ werden die in der Variablen myString festgelegten Eigenschaften im <div> mit der ID „display“ ausgegeben. In myString sind das Bild am oberen Rand, der Text, sowie die Buttons festgelegt. Diese Buttons sind mit verschiedenen Funktionen versehen, welche ‚onclick‘, also beim Anklicken, weitere Funktionen aktivieren. Der Rezepte Button aktiviert die function zeigeRezept().

Abb. 2: Screenshot Menü zeigeShoppinglist() wird durch den Button mit der Aufschrift „Einkaufsliste“ in Gang gesetzt. Der letzte Button ruft die function zeigeFormular() auf. Damit alles visuell ausgegeben werden kann befinden sich in diesen Funktionen HTML Elemente.

## Rezepte

Abb. 3: Screenshot Rezepte

Der erste Screenshot zeigt den Kopf der Seite, in diesem bieten Buttons eine schnelle Navigation zu den anderen Bestandteilen der App. Diese funktionieren nach dem gleichen Prinzip wie die der Startseite.

Die function zeigeRezept() beinhaltet ebenso eine Variable myString, in der festgeschrieben ist, wie die Rezepte ausgegeben werden. Festgesetzt wurden darin die HTML-Elemente, die beiden Pfeile links und rechts sind ermöglichen das vor und zurück blättern der Rezepte. Durch Klicken der Buttons über dem Bild des Rezeptes werden die Zutaten auf die ausgewählte Portionenzahl herunter gerechnet. Die Rezeptliste ist in eine Tabelle mit drei Spalten eingefügt. In der ersten Spalte befindet sich die Zutat, die zweite beinhaltet die Mengenangabe mit Einheit. In der letzten Spalte befindet sich ein Icon, das mit dem Befehl ‚onclick‘ die function addToListe(eintrag) aktiviert. Durch diese werden die Zutaten auf die Einkaufsliste gesetzt. Unter der Liste der Rezepte befindet sich die Zubereitungsdauer, die Schwierigkeitsangabe und die Beschreibung. In dieser befinden sich weitere Buttons mit denen der Timer aktiviert werden kann. Abschließend gibt es ein Textfeld in dem eine Notiz eingetragen werden kann.

## Einkaufsliste

Auch auf dieser Seite befindet sich die Navigation am oberen Rand. Die Einkaufsliste wird ebenso wie die anderen Seiten durch eine Funktion ausgegeben. In diesem Fall ist es die function zeigeShoppinglist(). In dieser wird die globale Variable shoppinglist ausgegeben. Mit dem Button „Alles löschen“ wird die function allesLoeschen() aktiviert wird, diese leert die gesamte Einkaufsliste. Die „löschen“ Buttons hinter den Listenpunkten setzen ‚onclick‘ die function löschen(index) in Gang, wodurch nur der gewählte Punkt aus der Liste entfernt wird.

Abb. 4: Screenshot Einkaufsliste

## Formular

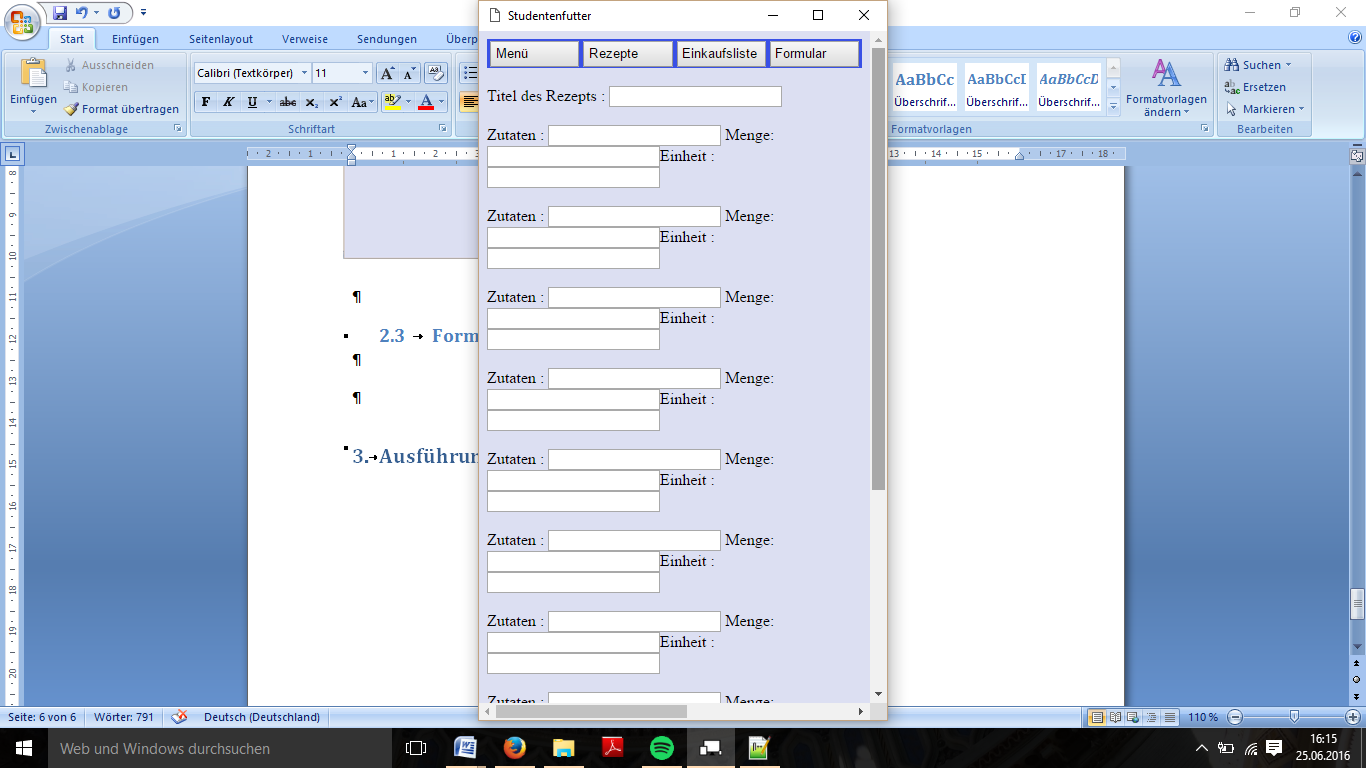
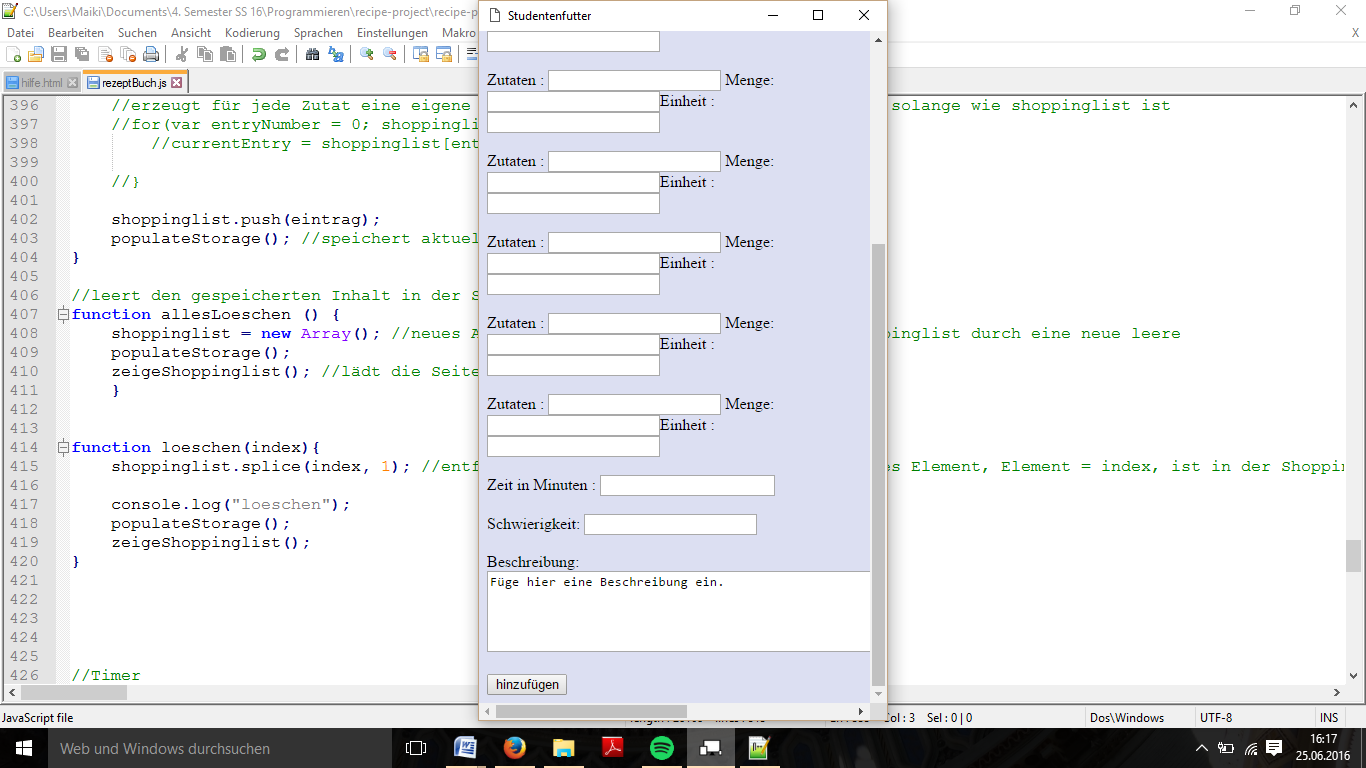
Das Formular beinhaltet mehrere <input>- Felder, die in der function zeigeFormular() eingesetzt sind. Durch die Anzeige im Modus des iPhone 6s entsteht das Problem, dass sich die Label verschieben. Am unteren Ende des Formulars befindet sich ein Textfeld, in dem die Zubereitungsbeschreibung eingefügt werden kann. Der Button „hinzufügen“ soll eine Funktion in Gang setzen, mit der das Rezept ab-

Abb. 5: Screenshot Formular gespeichert wird. Des Weiteren soll-

te noch ein Bild-Upload-Bereich eingerichtet werden, damit alle Angaben, die ein vorhandenes Rezept beinhaltet, eingegeben werden können.

# Umsetzung in JavaScript

In diesem Abschnitt der Dokumentation werden die Ausführung der einzelnen Funktionen und der Aufbau der Datenstruktur genauer erläutert.

## 3.1 Datenstruktur des RezeptBuches

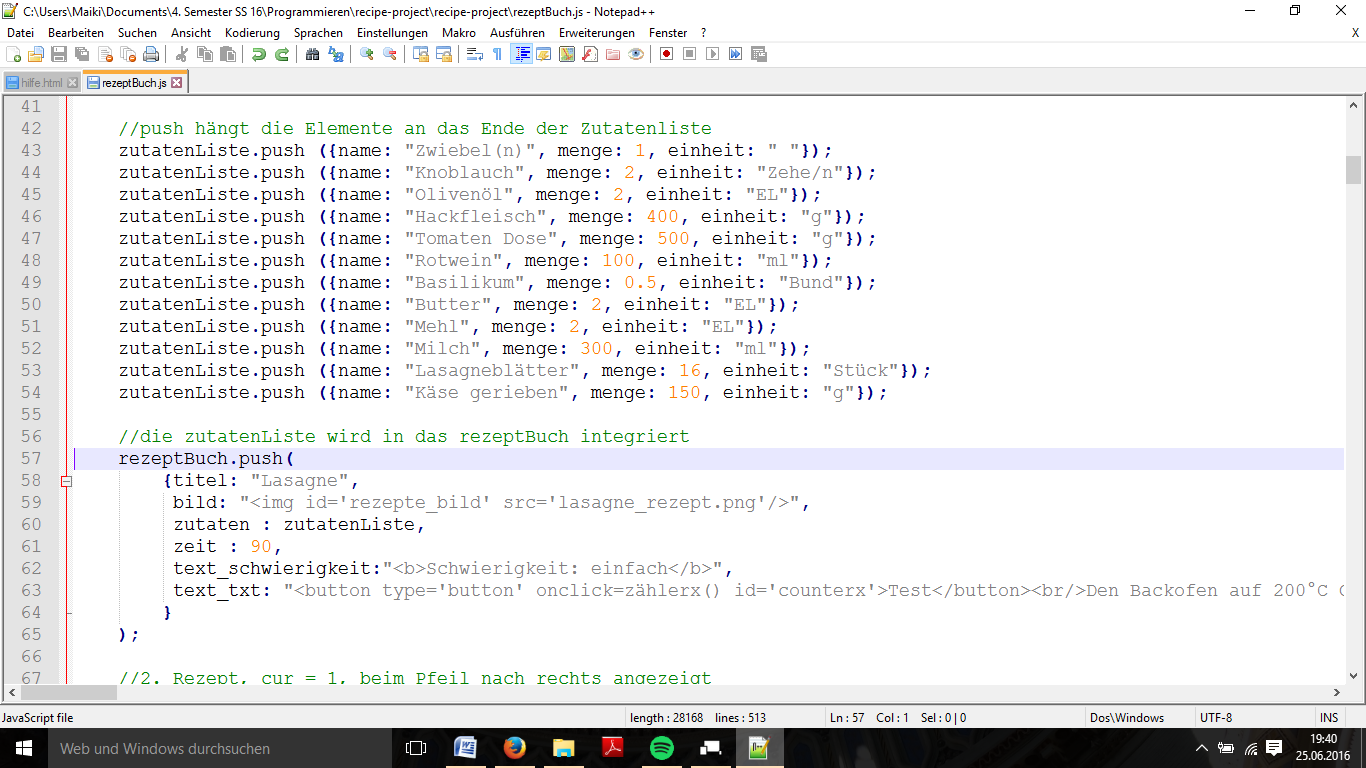
Um die Rezepte mit ihren Zutaten in Javascript abspeichern zu können, sollten sie in einer objektorientierten Struktur abgespeichert werden. Diese wurde mit Hilfe von Prof. Pfeffer erstellt. Im Falle dieses Projektes beinhaltet die function loadData() eine solche Struktur. In der function wird die Variable rezeptBuch als leerer Array erstellt, sowie die Variable zutatenListe.

Abb. 6: Screenshot Datenstruktur

Der oben abgebildete Quellcode zeigt, wie die leeren Arrays „gefüllt“ werden. In die Variable zutatenListe kommen Name der Zutat, die Menge sowie die Einheit in der die Menge angegeben werden soll. Durch das anhängen von „.push“ wird an das Ende des Arrays eine neue Zutat angehängt. In die Variable rezeptBuch werden nun der Titel des Rezeptes, das Bild, die Zeit, der Schwierigkeitsgrad, sowie eine Zubereitungsbeschreibung hinzugefügt. Auch der Array zutatenListe wir nun in das rezeptBuch gepusht. Für jedes Rezept wurde eine solche Struktur erstellt.

Nun ist das nächste Ziel die Ausgabe der Rezepte im Browser. Die Funktion zeigeRezept() bewerkstelligt dies. In ihr wird zuerst die die function loadData() integriert, indem sie der Variablen rezeptListe gleich gesetzt wird. In der Variablen myString werden alle Daten festgesetzt die ausgegeben werden sollen. Dabei werden HTML-Elemente eingesetzt um eine strukturierte Ausgabe zu ermöglichen.

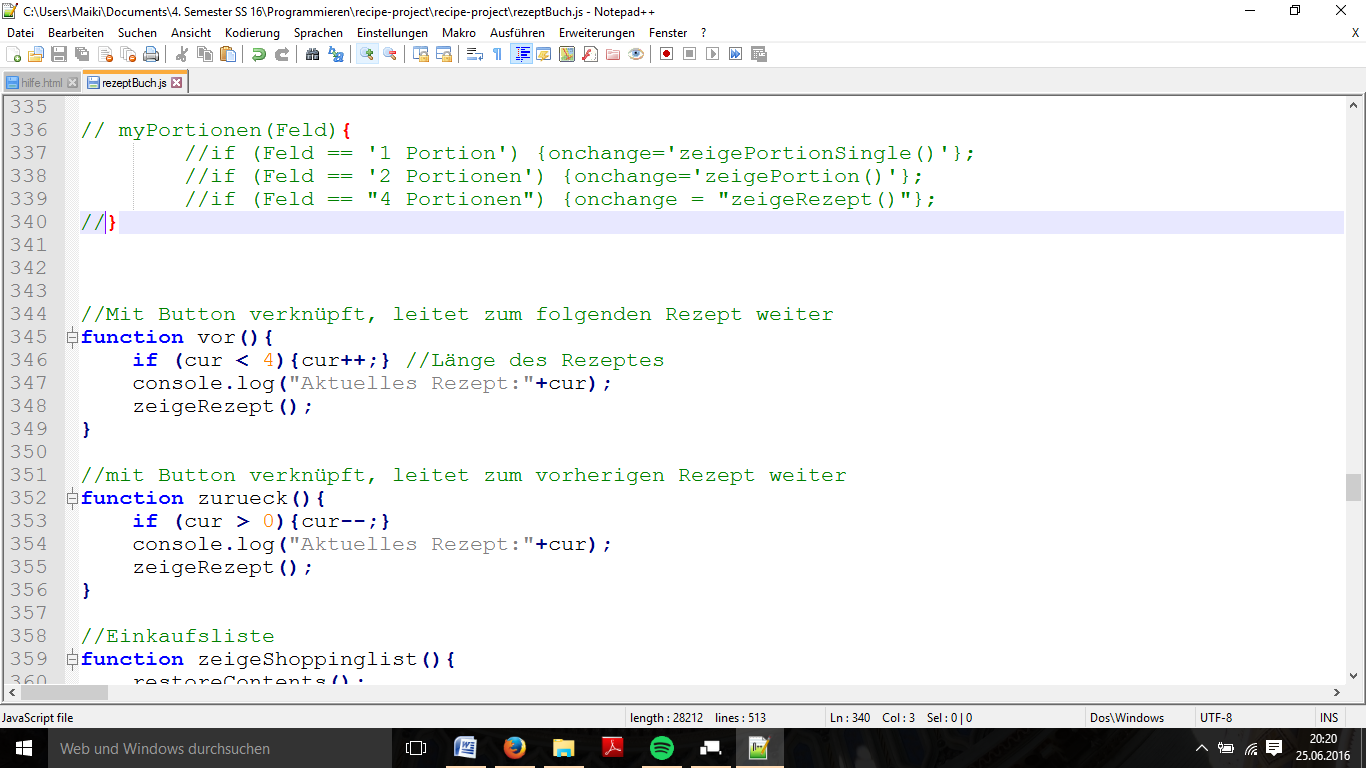
myString enthält auch die Icons, mit denen vor und zurückgeblättert werden kann. Sie sind verknüpft mit Funktionen, durch deren Befehl ‚onclick‘, werden diese bei Anklicken der Icons ausgelöst.

Abb. 7: Screenshot der Pfeilfunktionen

Der oben abgebildete Ausschnitt des Codes zeigt die Funktionen. Dort wurde festgelegt, dass solange die globale Variable cur <4 ist, die Funktion ein cur, also ein Rezept, weiterspringt. Der Befehl zeigeRezept() ruft diese Funktion auf. Die function zurueck() arbeitet nach demselben Prinzip, nur dass ihre Bedingung ist, dass die cur >0 sein sollte.

Bei Erstellung von Buttons, zur Verlinkung auf functions muss die richtige Zeichensetzung beachtet werden. Da sich alles in einem Javascript-Script befindet, muss man HTML-Code in Anführungszeichen setzen. Damit Funktionen funktionsfähig sind, und als Javascript erkannt werden, müssen sie mit \“ ..... \“ gekennzeichnet werden.

Durch eine dort eingefügte for-Schleife, werden alle Rezepte aufgerufen und in Tabellen ausgegeben. Die for Schleife nutzt die globale Variable var cur=0, um die Rezepte zu nummerieren und eine Ausgabe zu ermöglichen. Die Bedingung der Schleife sieht dabei folgendes vor:   
for (var i = 0; i < rezeptListe[cur].zutaten.length; i++).

Das bedeutet, dass für jedes Rezept alle Werte von rezeptListe, bzw. loadData ausgegeben werden sollen. Die Schleife läuft, solange i < der Länge der Zutaten ist. Sie erstellt mithilfe der Variablen myString mehrere Tabellen, in denen die Bestandteile der loadData() ausgegeben werden. Diese for-Schleife erstellt auch die Buttons, mit denen einzelne Zutaten zur Einkaufsliste hinzugefügt werden können. Die function addToListe(eintrag) wird durch Anklicken aufgerufen.

Um alles im HTML-Element ‚display‘ auszugeben wird folgender Befehl ausgeführt:  
document.getElementById('display').innerHTML = myString;

Alle Funktionen, die etwas visuell ausgeben sollen, arbeiten nach diesem Prinzip. Sie haben immer das Wort „zeige“ in der Funktionsbezeichnung.

## 3.2 Abspeichern der Einkaufsliste

Damit die Einkaufliste nach dem Schließen der App, beim Wiederöffnen, noch immer gefüllt ist, muss ein Speicher eingerichtet werden. Hier wurde sich für einen localStorage entschieden. Dieser ermöglicht das clientseitige Speichern von Daten im Browser. Im Gegensatz zu Cookies ist diese Variante sicherer, da sie nicht mit einem Server kommunizieren muss.

Mit der Funktion populateStorage () werden die Inhalte der Shppinglist im Key shoppingInhalt gespeichert. Dafür ist der Befehl localStorage.setItem maßgebend. Bei der Verwendung von localStorage muss darauf geachtet werden, dass abzuspeichernde Daten in das JSON Format umgeschrieben werden müssen. Der Befehl JSON.stringify führt dies auf. Um an gewünschter Stelle den Inhalt des localStorage mit dem key shoppingInhalt wieder abzurufen wurde die function restoreContents() erstellt. Diese nutzt shoppinglist = JSON.parse(localStorage.getItem('shoppingInhalt')) um die Daten aus dem Speicher wieder zurück in Javascript zu übersetzen und den Inhalt des shoppingInhalt auslesen zu können.

## 3.3 Löschen der Einkaufsliste und des lokalen Speichers

Die Einkaufsliste kann entweder ganz gelöscht werden oder nur einzelne Listenpunkte von ihr. Zuständig dafür sind die Funktionen allesLöschen() und löschen(index). Erstere leert die Variable shoppinglist, in dem sie einem neuen Array gleichgesetzt wird. Sie wird also überschrieben. Um dies zu speichern, wird die function populateStorage() integriert. Durch aufrufen zeigeShoppinglist() wird die Einkaufsliste aktualisiert.

Die Funktion löschen(index) entfernt nur den ausgewählten Punkt von der Liste. Index steht dabei für die Nummer, die die Zutat im Array hat. Durch den Befehl shoppinglist.splice(index,1) wird festgelegt, dass index gelöscht werden und nur einen Listenpunkt dabei entfernt werden soll. Auch in dieser Funktion wird das Ergebnis erneut im Speicher abgelegt und die Einkaufsliste aktualisiert.

## 3.4 Timer

Der Timer stellte einige Herausforderungen dar, da die Zeit nicht manuell eingegeben werden sollte, sondern die Zeit aus den Zubereitungsbeschreibungen entnommen werden musste. Eine Lösung fand sich mit Hilfe dieses Tutorials:

<https://wiki.selfhtml.org/wiki/JavaScript/Anwendung_und_Praxis/komfortable_Timer-Funktion>

Der kommentierte Quellcode erklärt Schritt für Schritt den Aufbau des Codes. Dieser ermöglicht das Einbeziehen von festgesetzten Zeiten. Durch Funktionen konnten diese individuellen Zeiten in das jeweilige Rezept eingefügt werden. Die folgenden Ausschnitte des Quellcodes verdeutlichen die

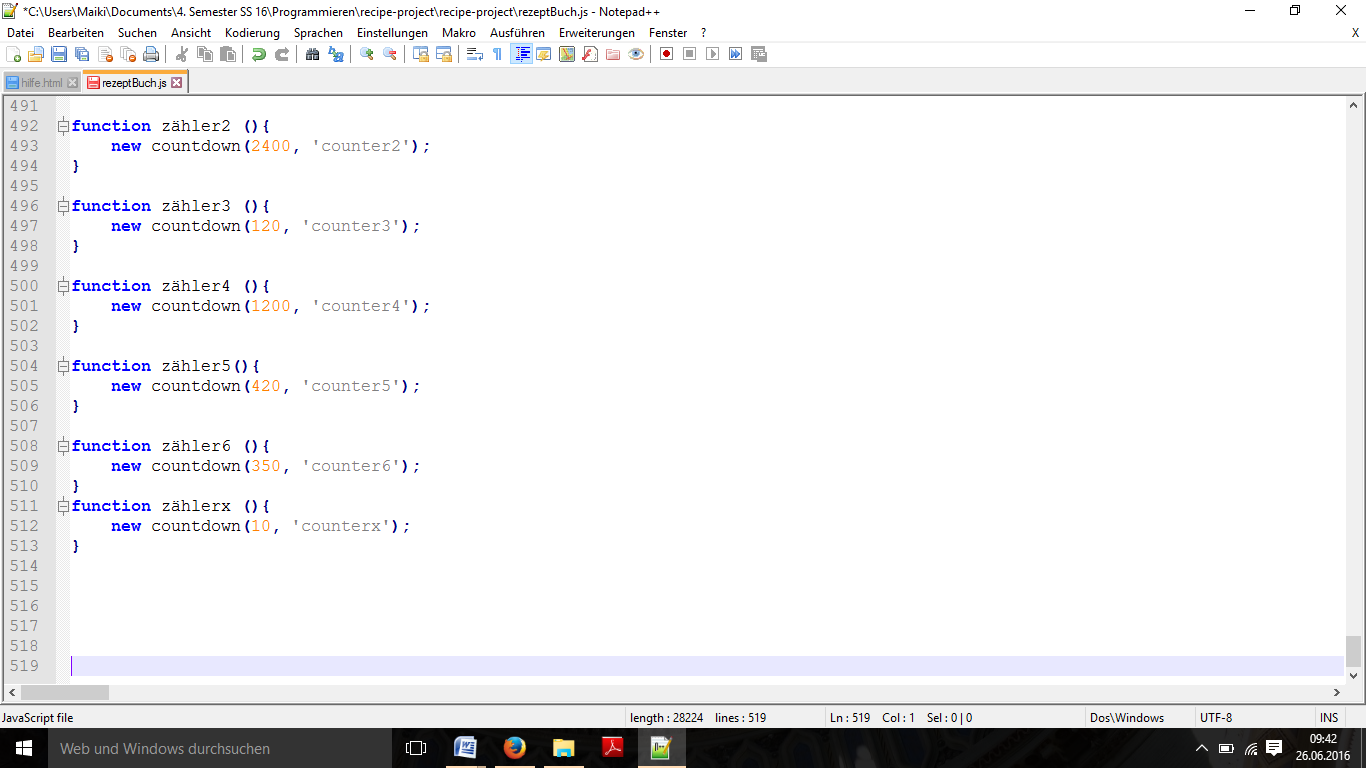


Abb. 8: Screenshot Zählerfunktion

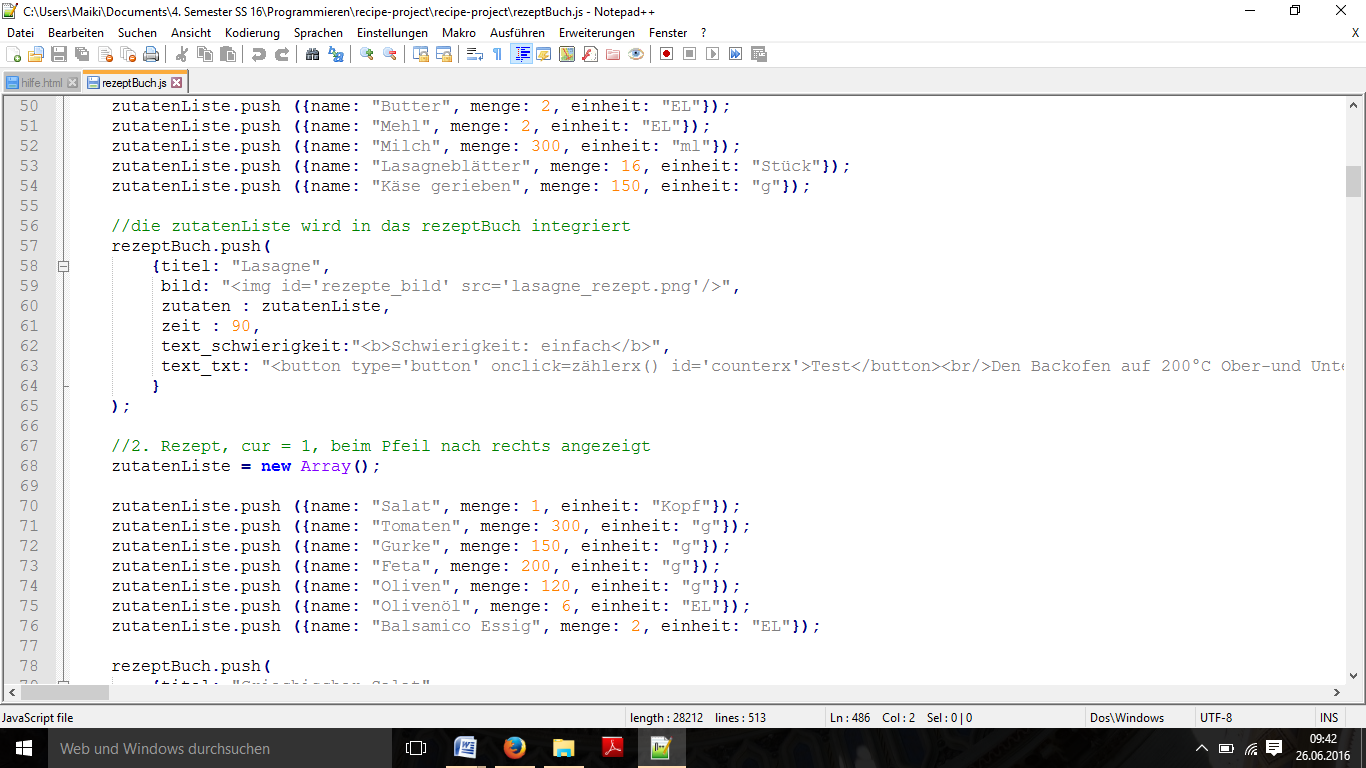


Abb. 9: Screenshot Einbindung des Zählerbuttons

Nach Ablaufen der Zeit erscheint ein Pop-up begleitet von einem Ton, dass dem Nutzer signalisiert, dass der Timer abgelaufen ist.

# Aussicht

Bei weiterer Bearbeitung der App, wäre es nötig einige weitere Funktionen hinzuzufügen. So ist es noch nicht möglich die Notiz am Ende der Rezepte abzuspeichern. Um dies zu bewerkstelligen wäre ein zweiter lokaler Speicher möglich. Durch das Klicken des Buttons „speichern“ kann der Inhalt des Textfeldes in einer Variablen abgespeichert werden, welche dann im localStorage gespeichert wird. Ähnlich muss man bei dem Abspeichern des Formulars vorgehen, damit die Rezepte bei erneutem öffnen der App noch immer vorhanden sind.

Auch die Einkaufsliste kann noch weiter verbessert werden. Wünschenswert und sinnvoll wäre hier eine Funktion, mit der bei hinzufügen von neuen Zutaten, die Liste durchsucht wird. Befindet sich die neu hinzugefügte Zutat bereits in der Einkaufsliste, sollen sie miteinander addiert werden. Ein Nachteil dieser Funktion wäre jedoch, dass falsch hinzugefügte Zutaten nicht so leicht wieder entfernt werden könnten, da sie bereits mit anderen addiert worden sind.

Eine elegantere Lösung der Portionenauswahl durch einen Dropdown Button ist bisher wegen des Mangels an Zeit fehlgeschlagen, das Grundgerüst dafür ist jedoch bereits vorhanden und müsste nur noch angepasst werden. Auch die Funktion zum Berechnen der Portionen könnte eventuell eleganter gestaltet sein, als die gegenwärtige Lösung.

Ebenso das Umwandeln der derzeitigen nur im Browser abrufbare Anwendung zu einer Webapp wäre sinnvoll, damit eine einfache Bedienung am Handy möglich ist.

# Quellenverzeichnis

mediaevent (2014): Javascript Arrays: Elemente einfügen und entfernen. URL: <http://www.mediaevent.de/javascript/Javascript-Array-Methoden.html> (31.05.2016)

selfHTML (2016): JavaScript/Anwendung und Praxis/komfortable Timer-Funktion. URL: <https://wiki.selfhtml.org/wiki/JavaScript/Anwendung_und_Praxis/komfortable_Timer-Funktion> (17.06.2016)

w3schools (o.J.): HTML5 LocalStorage. URL: <http://www.w3schools.com/html/html5_webstorage.asp> (09.06.2016)

selfHTML (2016): JavaScript/ WebStorage. URL: <https://wiki.selfhtml.org/wiki/JavaScript/Web_Storage#localStorage> (09.06.2016)

soundbible (o. J.): Japanese Temple Bell Small Sound. URL: <http://soundbible.com/1496-Japanese-Temple-Bell-Small.html> (17.06.2016)

stack overflow (2013): Playing audio with Javascript?. URL: <http://stackoverflow.com/questions/9419263/playing-audio-with-javascript> (17.06.2016)

datenschwamm (o. J.): Formular mit JavaScript. URL: <http://www.datenschwamm.de/4/javascript_formular.php> (21.06.2016)

selfHTML (2016): JavaScript/Anwendung mit Praxis/Verkettete Auswahllisten. URL: <https://wiki.selfhtml.org/wiki/JavaScript/Anwendung_und_Praxis/Verkettete_Auswahllisten> (20.06.2016)

## 

## Textverzeichnis

Mexikanische Tomatensuppe

eatsmarter (o.J.): Lasagne. URL: <http://eatsmarter.de/rezepte/lasagne> (08.06.2016)

Chefkoch (o.J.): Griechischer Salat. URL: <http://www.chefkoch.de/rezepte/541451151429388/Griechischer-Salat.html> (01.06.2016)

Eatsmarter (o.J.): Paprika-Zucchini-Nudeln. <URL:http://eatsmarter.de/rezepte/paprika-zucchini-nudeln> (01.06.2016)

## Bilderverzeichnis

Thieme (o.J.): Veggie. URL:  
<https://www.thieme.de/statics/bilder/thieme/final/de/bilder/tw_ernaehrung/Veggie_(2).jpg> (16.06.2016)

Lasagne Rezept  
eatsmarter (o.J.): Lasagne. URL: <http://eatsmarter.de/rezepte/lasagne> (08.06.2016)

Chefkoch (o.J.): Griechischer Salat. URL: <http://www.chefkoch.de/rezepte/541451151429388/Griechischer-Salat.html> (01.06.2016)

Blogspot (o.J.): Mexikanische Tomatensuppe. URL:<http://4.bp.blogspot.com/-RMm-xwKdnaI/Ud-spJ2Z5VI/AAAAAAAAGRM/e6iioqkQiZI/s1600/Spanische+Paprika-Tomatensuppe+2.jpg> (01.06.2016)

Eatsmarter (o.J.): Paprika-Zucchini-Nudeln. <URL:http://eatsmarter.de/rezepte/paprika-zucchini-nudeln> (01.06.2016)

Icon: Flaticon (o.J.): Magnifier tool free icon. URL: <http://www.iconsdb.com/custom-color/arrow-icons.html> (28.06.2016)