

# **Medienprojekt I**

## **Thema: Videochop**

Michael Duve, Felix Maulwurf, Angelina Staech

29.11.2014

# Inhaltsverzeichnis

<b>1 Projektteilnehmer</b>	<b>3</b>
1.1 Michael Duve . . . . .	3
1.2 Felix Maulwurf . . . . .	3
1.3 Angelina Staeck . . . . .	3
<b>2 Projektbeschreibung</b>	<b>4</b>
2.1 Idee . . . . .	4
2.2 Zukunft . . . . .	4
2.3 Probleme . . . . .	4
2.4 Was läuft gut? . . . . .	4
2.5 Was läuft nicht so gut? . . . . .	5
2.6 Teamstruktur und Aufteilung? . . . . .	5
<b>3 Mockup</b>	<b>6</b>
<b>4 Testcases</b>	<b>7</b>
4.1 Canvas . . . . .	7
4.2 Drag-Drop . . . . .	8
4.3 Video Export . . . . .	8
4.4 Format Test . . . . .	8
4.5 Video Concat . . . . .	10
4.6 FFMPEG . . . . .	10
4.7 Video Controls . . . . .	11
<b>5 Module</b>	<b>12</b>
5.1 Video Item . . . . .	12
5.2 Video Item Loader . . . . .	13
5.3 Video List . . . . .	13
5.4 Video Preview . . . . .	14
5.5 Video Controls . . . . .	14
5.6 Video Timeline . . . . .	14
<b>6 Zeitaufwand</b>	<b>15</b>
<b>7 Externe Plugins</b>	<b>16</b>
7.1 jQuery Draggable . . . . .	16
7.2 jQuery Droppable . . . . .	16
7.3 jQuery Collision . . . . .	16
7.4 FFMPEG . . . . .	16
7.5 Popcorn JS Capture . . . . .	16
7.6 Filereader . . . . .	16
7.7 UserAgent Parser . . . . .	16
7.8 FileSaver . . . . .	16

videoChop  
slice.compose.online

# **1 Projektteilnehmer**

## **1.1 Michael Duve**



**Medieninformatik Fachsemester 4**

HTML5, CSS3, JS, PHP, MySQL, Webapps iOS,  
Grundkenntnisse Server, Java, min. Python Photoshop, Illustrator, InDesign  
Repositories (Git, Mercurial), JIRA, Documentation

## **1.2 Felix Maulwurf**



**Medieninformatik Fachsemester 4**

HTML5, CSS3, JS, PHP, MySQL, Java,  
Photoshop, Illustrator, After Effects, 3DSMax,  
Cinema4D, Office, Repositories (Git), LaTeX

## **1.3 Angelina Staech**



**Medieninformatik Fachsemester 4**

HTML5, CSS3, JS, PHP,  
Java, SQL, Illustrator, Photoshop

## 2 Projektbeschreibung

VideoChop ist eine Webanwendung basierend auf JavaScript, die dem Anwender ein einfaches und intuitives „Videoschnittstudio“ zur Verfügung stellt. VideoChop soll es ermöglichen Videos vom Desktop direkt in den Browser zu ziehen. Es steht dem User frei, ob er ein oder mehrere Videos schneiden oder zusammenfügen möchte. VideoChop wird eine übersichtliche und schlichte Oberfläche bieten, die auch Anfänger nicht überfordert. Die Videos werden in einer Zeitleiste organisiert und beschnitten. Wer mal eben schnell ein Video schneiden will, um Vor- oder Abspann oder etwaige Längen zu entfernen, braucht keine Speicher fressende und komplizierte bedienende Videobearbeitung. Der Nutzer kann sich sein Werk vor dem Abspeichern im Browser als Vorschau ansehen.

### 2.1 Idee

Die Idee ist es, dass VideoChop dem Nutzer eine Plattform zur Verfügung stellt, die keinerlei Daten serverbasiert speichert. Das Projekt basiert auf JavaScript und nutzt somit die Rechenressourcen des Nutzers. Die Belastung für den Server sind damit minimal. VideoChop soll einfach, intuitiv und schnell sein. JavaScript ist sehr mächtig, allerdings sind zu Projektbeginn die Möglichkeiten und Umsetzbarkeiten noch nicht geklärt.

### 2.2 Zukunft

Zukünftig sollen weitere Funktionalitäten hinzukommen.

- Audiospuren manipulieren
- Audio isolieren und neues Material auf Audio legen
- Webcam Recorder zum erstellen eigener Videos
- Export in verschiedene Formate

### 2.3 Probleme

- Das Verbinden mehrerer Videos zu einem mit Hilfe von JavaScript ist sehr problematisch.
- FFMPEG ist kein Javascript, sondern lediglich umgeskriptet, sodass man nicht alle Befehle wie in der Konsole ausführen kann
- Chrome liebt sein Aw... Snap! und das passiert sehr oft, wenn der Speicher voll wird

### 2.4 Was läuft gut?

Alles! Wir sind im Zeitplan. Haben Spaß am Projekt und immer wieder noch mehr Ideen für die Umsetzung. Langsam aber sicher geht es jetzt an das Module schreiben.

## **2.5 Was läuft nicht so gut?**

Zusammensetzen. Meistens schaffen wir es nur alle 3 in Skype zusammen zu kommen.  
Das liegt aber an unseren Nebenjobs und der Zeitplanung.

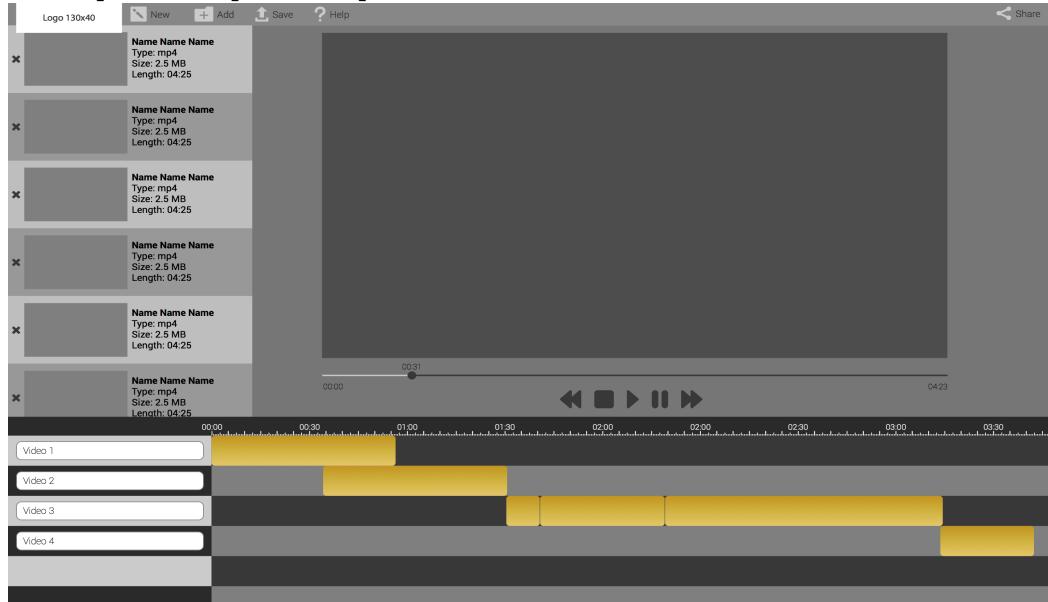
## **2.6 Teamstruktur und Aufteilung?**

Jeder macht alles. Jeder macht das, wozu er Lust hat. Michael teilt sein Wissen aus  
seinem Entwickler-Job.



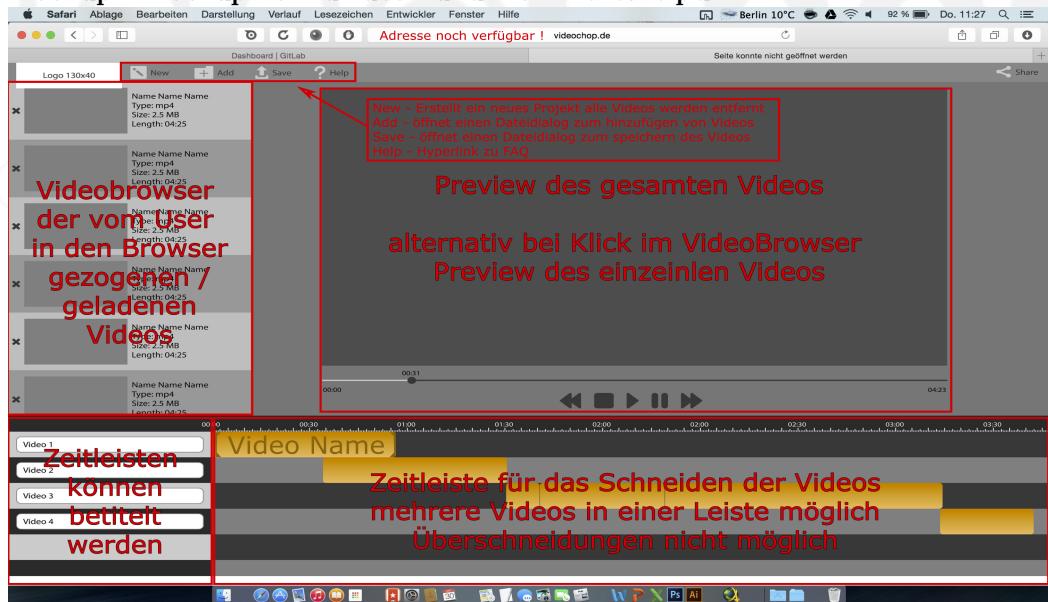
### 3 Mockup

mockup - mockup-vc-MD.pdf



Das Mockup ist nur eine Orientierung für die finale Website.

mockup - mockup-vc-md-Ideen-und-Kommentare.pdf



Es wurde in Gruppenarbeit erstellt.

## 4 Testcases

Für jede Funktion haben wir Testcases erstellt. Die Testcases werden jeweils nach Bedarf erweitert. Das bedeutet, dass im weiteren Verlauf der Entwicklung neue Testcases hinzukommen können. Die Testcases sind spezifisch und nicht modular programmiert. Aus den Testcases haben wir unsere Module abgeleitet.

### 4.1 Canvas

#### testcases - canvas - index.html

Allgemeiner Funktionstest des Canvas Objekts.

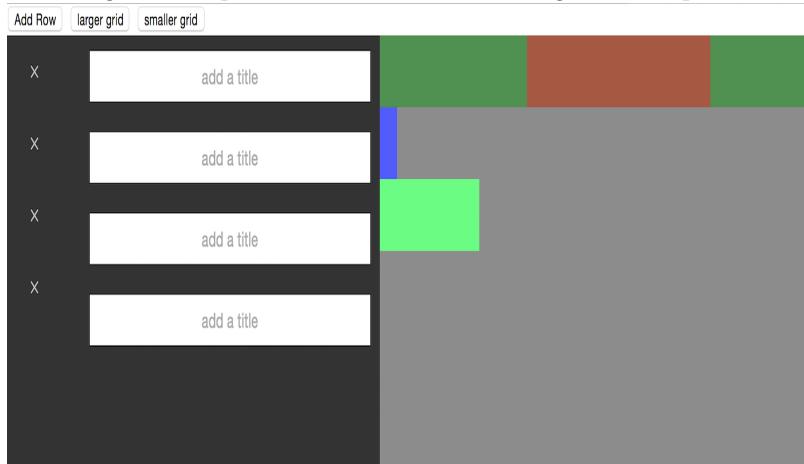


Abilden eines Videos auf einem Canvas Objekt. Wir wollten in diesem Test-Case versuchen ein Video auf einem Canvas abzuspielen. Wir hatten aber nicht im Blick, dass die Audiospur dann ja fehlt und das man jeden Frame des Videos nur als Bild aus dem Canvas speichern kann.

## 4.2 Drag-Drop

### testcases - drag-drop - index.html

Der Drag and Drop Testcase dient als Grundlage für die spätere Timeline.



Aus diesem Testcase wurde das Modul Video Timeline abgeleitet.

Es ist möglich neue Zeilen hinzuzufügen und bestehende Leisten zu löschen.

Zudem kann man die hier farbigen Objekte, die spätere Videos darstellen sollen, innerhalb der Leisten verschieben. Die Schritte in denen verschoben werden, sprich die Größe des Grids ist einstellbar, damit wir später stufenlos skalieren können.

Die Videos können in neue Zeilen gezogen werden, allerdings ist es nicht möglich, dass sich Videos überschneiden.

## 4.3 Video Export

### testcases - export\_videos - index.html

Dieser Test befasst sich mit der Möglichkeit Videos aus dem Browser zu exportieren. Wir haben versucht mit der Funktion `toBlob()` das aktuelle Bild, welches auf das Canvas gezeichnet wird, in ein Blob mit den Informationen in einen Array zu speichern. Dies hat funktioniert und wir konnten auch anschließend eine MP4-Datei exportieren. Jedoch konnte es nicht als Video abgespielt werden, da die gespeicherten Informationen nicht MP4-konform waren. Der FileSaver hat demnach nicht zum gewünschten Ergebnis geführt in diesem Fall. Es konnte lediglich mit der Funktion `toDataURL()` das aktuelle Bild auf dem Canvas als URL gespeichert und angezeigt werden.

Durch das erfolgreiche Testen des Aneinanderfügens von zwei Videos und das zur Verfügung stellen als URL, benötigen wir das Canvas wahrscheinlich nicht mehr.

## 4.4 Format Test

### testcases - format-test - index.html

Wir haben nur versucht ein Video per Drag & Drop im Browser fallen zu lassen und

es dann als HTML5-Video anzeigen zu lassen. Der Test war erfolgreich. WebM, MP4 und OGG können auf der Fläche platziert werden. Der Browser lädt das Video in den Cache und kann mittels `createObjectUrl()` das Video wieder in ein Source-Objekt verwandeln.



## 4.5 Video Concat

### **testcases - concat-videos - index.html**

Dieser Test sollte einzig dazu dienen zwei MPG-Videos mittels binärer Konkatenation zu einem Video zusammenzufügen. Der Test war erfolgreich. Das Projekt ist dadurch umsetzbar. Jetzt können wir 2 Videos aneinanderfügen.

## 4.6 FFMPEG

### **testcases - ffmpeg - index.html**

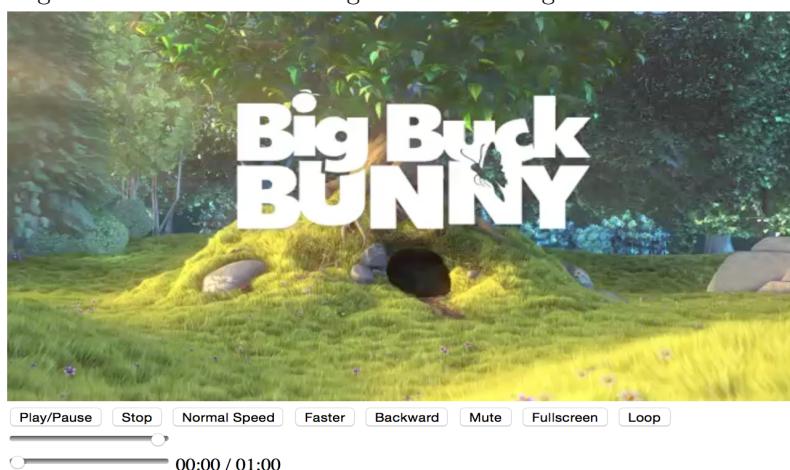
Das liebe FFMPEG bereitete am meisten Probleme. Dadurch, dass es eine Konsolen-Applikation ist und nicht natives Javascript mussten wir erstmal den Umgang mit Web-worker lernen und verstehen, wie man die Parameter richtig übergibt. Wir können Audio und Video separat rausrendern, sowie ein Dateiformat in ein anderes verwandeln. Die unterstützte Codec-Liste ist: <http://bgrins.github.io/videoconverter.js/demo/> hier zu finden. Einfach auf List Codecs klicken und den Command ausführen.

## 4.7 Video Controls

### testcases - video-controls - index.html

Der Testcase Video Controls sollte die später benötigten Funktionen zum kontrollieren des Videos über JavaScript simulieren.

Folgende Funktionen wurden geschrieben und getestet :



- Button
  - Play/Pause
  - Stop
  - Faster / Normal Speed
  - Backward
  - Mute
  - Fullscreen
  - Loop
- Slides
  - Volume
  - Current Time
- Timeupdate
  - Display Current Time
  - Display Duration Time

Das Crossbrowser Testing hat ergeben, dass einige Funktionen noch einmal überarbeitet werden müssen.

Aus diesem Testcase wurde das Modul Video Controls abgeleitet.

## 5 Module

### 5.1 Video Item

testcases - module\_video\_item - index.html

videochop - js - modules - videoItem.js

Jedes vom Nutzer in den Browser gezogene Video wird durch ein VideoItem repräsentiert. Es speichert alle relevanten Daten. Die Daten werden beim in den Browser ziehen vom Modul VideoItemLoader ausgelesen und in ein neues VideoItem geschrieben.

```
function VideoItem(settings) {
    this.settings = {
        video: null,
        name: "TEST",
        length: 100,
        start: 0,
        end: 100,
        size: 5500,
        resolution: {
            width: 3840,
            height: 2160
        },
        thumbnail: null
    };

    // if settings were not set by initializing, fill with default settings
    $.extend(this.settings, settings || {});
}

this.initialize();
```

Das Videoitem bekommt die Eigenschaften Video-URL, Name, Länge, Startpunkt, Endpunkt, Größe, Auflösung und Thumbnail übergeben oder es werden die Default-Settings gesetzt. In der Funktion getMarkUp() werden die HTML-Felder in der index.html dann mit den Werten des VideoItems befüllt, um es in der VideoItemList entsprechend anzuzeigen.

```
getMarkUp: function () {
    return '<li class="file" id="video-item-' + this.id + '">' +
        '<div class="file-delete icon_close"></div>' +
        '' +
        '<div class="file-info">' +
        '<p>Name: <span class="file-name">' + this.settings.name + '</span></p>' +
        '<p>Duration: <span class="file-duration">' + this.timeFormat() + '</span></p>' +
        '<p>Size: <span class="file-size">' + this.sizeFormat() + '</span></p>' +
        '</div></li>';
},
```

Die Größe und die Länge des Videos werden formatiert in den Funktionen timeFormat() und sizeFormat().

## 5.2 Video Item Loader

**testcases - module\_video\_item\_loader - index.html**  
**videochop - js - modules - videoItemLoader.js**

Mit diesem Modul wird nach dem Drag and Drop eines Videos vom Desktop in den Browser hinein ein neues VideoItem erstellt und zurückgegeben.

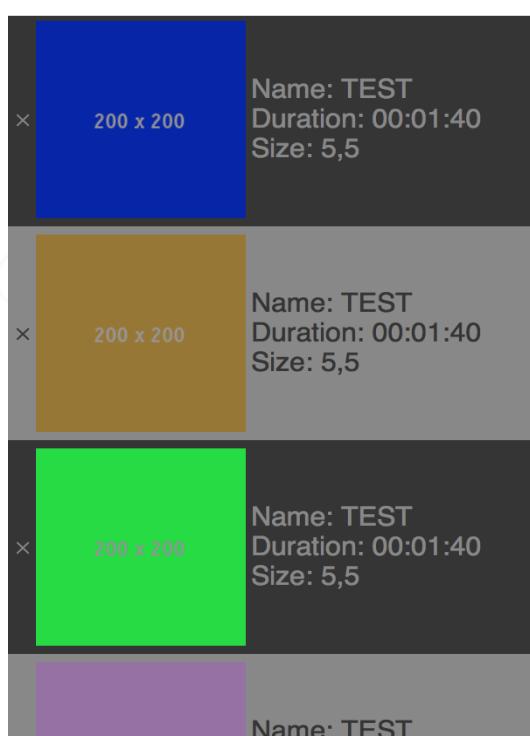
Es gibt die Attribute data, extension, name, prettySize, size und type welche zunächst auf Default-Werte gesetzt sind. Wird ein neuer VideoItemLoader instanziert, muss die Funktion add() aufgerufen werden, welche aus dem übergebenen Video eine Video-URL erzeugt. In der Funktion wird dann die Funktion loadMetaData aufgerufen und bekommt diese URL übergeben. In der Funktion wird ein Video-HTML-Element erzeugt und die URL wird diesem hinzugefügt. Anschließend wird ein neues VideoItem erzeugt und zurückgegeben, welchem die Eigenschaften des aktuellen Videos übergeben werden.

## 5.3 Video List

**testcases - module\_video\_list - index.html**  
**videochop - js - modules - videoList.js**

Das VideoList Modul stellt den Container für die VideoItems zur Verfügung.

click to add a video -> simulates a new drop



## **5.4 Video Preview**

Dieses Modul wurde noch nicht programmiert.

Stellt den Container für die Video Preview zur Verfügung.

## **5.5 Video Controls**

Dieses Modul wurde noch nicht programmiert.

Stellt alle Funktionen zum kontrollieren der Videos zur Verfügung.

## **5.6 Video Timeline**

Dieses Modul wurde noch nicht programmiert.

Das Video Timeline Modul stellt sämtliche Funktionen für die spätere Video Timeline zur Verfügung.



## 6 Zeitaufwand

Aufgabe	Zeitaufwand
Projektkonzept	3 Stunden
Einrichten Entwicklungsumgebung	5 Stunden
Installation & Konfiguration IDE	4 Stunden
HTML-Struktur	2 Stunden
JS-Module Template	1 Stunde
Coding Conventions	30 Minuten
Test-Case: Canvas	2 Stunden
Mockup	7 Stunden
Test-Case: Drag&Drop	6 Stunden
Test-Case: Video-Formate	2 Stunden
Test-Case: Video Controls	2 Stunden
Wiki: Lizenz prüfen	1 Stunde
Logo	2 Stunden
Test-Case: Video Controls Part II	3 Stunden
Content: Impressum	1.5 Stunden
HTML-Gerüst bauen	6 Stunden
Test-Case: Drag&Drop Part II	7 Stunden
Video Controls: Cross Browser Test	1 Stunde
Dokumentation: IDE	0.5 Stunden
Test-Case: Drag&Drop Part III	3 Stunden
Module: VideoItem	3 Stunden
Module: VideoList	4 Stunden
Module: VideoItemLoader	3 Stunden
Projektbeschreibung	4 Stunden
HTML-Gerüst: Timeline	2 Stunden
Test-Case: Video Part III	offen
Test-Case: Canvas Part II	offen
Module: Video Preview	offen
Module: Video Controls	offen
Module: Video Timeline	offen
FFMPEG	24 Stunden

## **7 Externe Plugins**

### **7.1 jQuery Draggable**

URL: <http://jqueryui.com/draggable/>

Wird benutzt um DOM-Elemente browserübergreifend dragbar zu machen

### **7.2 jQuery Droppable**

URL: <http://jqueryui.com/droppable/>

Wird benutzt um DOM-Elemente browserübergreifend auf bestimmten Elementen bestimmte Aktionen ausführen zu lassen.

### **7.3 jQuery Collision**

URL: <http://eruciform.com/static/jquidragcollide/jquery-ui-draggable-collision.js>

Wird benutzt, damit zwei VideoItems in der Timeline nicht aufeinander liegen dürfen

### **7.4 FFMPEG**

URL: <https://github.com/bgrins/videoconverter.js>

Wird benutzt um Videos umwandeln zu können.

### **7.5 Popcorn JS Capture**

URL: <https://github.com/rwaldron/popcorn.capture>

Wird benutzt, um ein Poster von einem Video an einer beliebigen Position als Thumbnail zu extrahieren

### **7.6 Filereader**

URL: <https://github.com/bgrins/filereader.js/>

Ein Plugin, die den Umgang mit Drag & Drop von Dateien STARK vereinfacht

### **7.7 UserAgent Parser**

URL: <https://github.com/faisalman/ua-parser-js>

Wird benutzt um festzustellen mit welchem Browser der User unterwegs ist

### **7.8 FileSaver**

URL: <https://github.com/eligrey/FileSaver.js>

Hiermit können wir Dateien direkt auf dem Desktop des Users speichern, weil dieses Plugin die HTML5-API ausbessert

## **7.9 jQuery, Modernizr & Require**

Dazu brauchen wir keine Beschreibung.

