

Medienprojekt I

Thema: Videochop

Michael Duve, Felix Maulwurf, Angelina Staech

13.01.2014

Inhaltsverzeichnis

1 Projektteilnehmer	3
1.1 Michael Duve	3
1.2 Felix Maulwurf	3
1.3 Angelina Staeck	3
2 Projektbeschreibung	4
2.1 Idee	4
2.2 Zukunft	4
2.3 Probleme	4
2.4 Was läuft gut?	5
2.5 Was läuft nicht so gut?	5
2.6 Teamstruktur und Aufteilung?	5
3 Layout	5
4 Testcases	6
4.1 Drag-Drop	6
4.2 FFMPEG	6
5 Module	7
5.1 Video Item	7
5.2 Video Item Loader	7
5.3 Video List	8
5.4 Video Preview	8
5.5 Video Controls	8
5.6 Video Timeline	8
6 Zeitaufwand	8
7 Externe Plugins	9
7.1 jQuery UI	9
7.2 jQuery Collision	9
7.3 FFMPEG	9
7.4 Popcorn JS Capture	9
7.5 Filereader	9
7.6 UserAgent Parser	9
7.7 FileSaver	9
7.8 jQuery, Modernizr & Require	9

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.

- Handlebars benutzen
- Ordner/Gruppierungen der Videos
- Frames pro Sekunde anzeigen
- Mute-Button für einzelne Videos
- Eigene Audiospur einbaubar
- Text auf Video ablegen
- Import unterschiedlicher Formate
- Export in verschiedene Formate

2.3 Probleme

- Keine! Lediglich die Zeit fehlt ;)

2.4 Was läuft gut?

Das zusammensetzen klappt mittlerweile gut. Wir haben große Fortschritte gemacht.
Das neue Design gefällt uns Dreien.

2.5 Was läuft nicht so gut?

Das Fertigstellen einer Version 1.0 zögert sich immer wieder hinaus, weil Fehler gefunden werden. Mit 3 Personen ist das Projekt echt sehr umfangreich.

2.6 Teamstruktur und Aufteilung?

Es macht immer noch jeder alles. Meist teilen wir uns die Tickets selber zu

3 Layout

Komplett neues Design: Das kostete uns am meisten Zeit



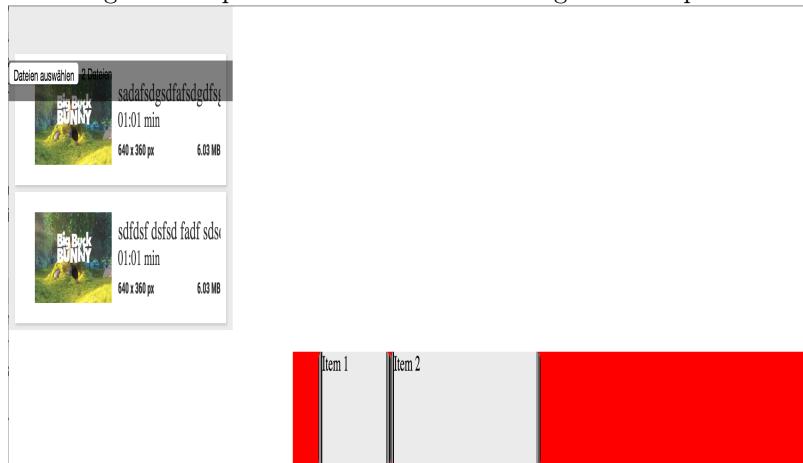
4 Testcases

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 Drag-Drop

<http://localhost/videochop/testcases/drag-drop-updated/index.html>

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



Der Testcase wurde angepasst. Der vorherige Test-Case verursachte zu viel Inkonsistenz beim Hinzufügen und Löschen.

4.2 FFmpeg

[testcases - ffmpeg - index.html](#)

FFmpeg funktioniert nun einwandfrei und wird im nächsten Milestone als Modul implementiert. Der Name des Moduls ist VideoExporter

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-Modul brauchte noch weitere Settings und Anpassungen. Einmal muss es das geladene Video halten. Die URL war als base64-String zu groß für das DOM, sobald zu viele Videos auf der Seite verbaut waren. Wir haben das Problem durch Veränderung der VideoURL und der Thumbnail-URL mittels Object-URL umgangen

```
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>';
},
```

5.2 Video Item Loader

```
testcases - module_video_item_loader - index.html
videochop - js - modules - videoItemLoader.js
```

Hier mussten wir auch ein paar kleinere Anpassungen machen. Dadurch, dass die dura-

tion eines Videos erst nach dem Event "metadataLoaded" zurückgegeben wird, mussten wir auf das asynchrone Event warten und dann erst das Video zur List hinzufügen.

5.3 Video List

testcases - module_video_list - index.html
videochop - js - modules - videoList.js

Hier waren auch Anpassungen notwendig. Wir wollten das man sich die Items in der VideoList anordnen kann wie man möchte. Das funktioniert nun. Zusätzlich wurde VideoList nun auch in unseren Controller eingebaut und funktioniert super mit dem neuen Layout.

5.4 Video Preview

Das Modul wurde angelegt. Es iteriert über eine Liste von VideoItems und spielt das gerade aktive Video von videoItem.settings.start bis zum vorgegebenen Ende ab. Dann Wechselt das Video zum nächsten. Momentan ist unser Gedanke die HTML-Videos aus und einzublenden, da das Wechseln der URL zu lange brauchen würde.

5.5 Video Controls

Dieses Modul brauchen wir so nicht mehr. VideoPreview übernimmt die Aufgaben.

5.6 Video Timeline

Dadurch, dass wir den Testcase Drag and Drop noch mal geupdatet haben, konnten wir dieses Modul nicht anfangen. Zum nächsten MS wird es aber angefangen sein.

6 Zeitaufwand

Aufgabe	Zeitaufwand
Design-/Layoutkonzept	5,5 Stunden
responsive Layoutumsetzung	16 Stunden
Testcase Drag and Drop	5 Stunden mit Recherche
PreviewVideo	7 Stunden
Anpassungen alter Module	5 Stunden
Reparatur Logo	1 Stunde
Sammlung FAQ-Fragen	0,5 Stunden
GIT-Comments und Aufräumen	0,5 Stunden
Bericht	1,5 Stunden
Debugging	3 Stunden
SCSS-Refactoring	1 Stunden
Insgesamt	56 Stunden

7 Externe Plugins

7.1 jQuery UI

URL: <http://jqueryui.com>
DOM-Elemente sind interaktiv nutzbar

7.2 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

7.3 FFMPEG

URL: <https://github.com/bgrins/videoconverter.js>
Wird benutzt um Videos umzuwandeln zu können.

7.4 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

7.5 Filereader

URL: <https://github.com/bgrins/filereader.js/>
Ein Plugin, die den Umgang mit Drag & Drop von Dateien STARK vereinfacht

7.6 UserAgent Parser

URL: <https://github.com/faisalman/ua-parser-js>
Wird benutzt um festzustellen mit welchem Browser der User unterwegs ist

7.7 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

7.8 jQuery, Modernizr & Require

Dazu brauchen wir keine Beschreibung.