Video Remix Tool:
Implementing the Video Player of a YouTube Remix Editor

Jesper Häggström

Computer Game Programming, bachelor's level
2017

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
BACHELOR’S THESIS

Video Remix Tool:
Implementing the Video Player of a
YouTube Remix Editor

Jesper Häggström

Supervisor
Patrik Holmlund

June 9, 2017

Luleå University of Technology
Department of Computer Science, Electrical and Space
Engineering
Abstract

NUITEQ is a company based in skellefteå that develops multitouch software for touch screens with a focus on educational applications. They wanted to know if an application, that could aggregate small clips from YouTube videos to create summaries for school subjects could be created.

This project is made with the existing YouTube API to create an application that can be used to create summaries about different subjects and then be shared with students and teachers.

The end result is a web application that can be used to search for, play and clip YouTube videos. The result can then be shared with other people. This paper will go through how the YouTube API is used to search for videos and how to use it to create players for the videos.

Sammanfattning

NUITEQ är ett företag baserat i skellefteå som utvecklar multitouch program för pekskärmar. Dom ville veta om en application, som kan klippa ihop små snuttar av YouTube videor för att skapa en sammanfattning för olika skolämnen.

Projektet använder det existerande YouTube APIet för att skapa en applikation som kan användas till att skapa sammanfattningar om olika ämnen och kan sedan delas med studenter och lärare.

Acknowledgements

I’d like to thank everyone that has helped me on this project. I would like to thank Patrik Holmlund for giving feedback on this paper, Edward Tse for giving feedback on the application, Harry van der Veen for allowing me to work on this project on NUITEQ. Lastly, i would like to thank my colleagues Axel Lassinatti and Rebecka Delehag for their hard work and dedication during this project.
Abbreviations and Terms

- API - Application programming interface
- JavaScript - A dynamic prototype-based scripting language often used for programming websites.
- CSS - Cascading Style Sheets
- HTML5 - Hyper Text markup language version 5
- CORS - Cross-Origin Resource Sharing
- OS - Operating System
- Native App - Native application. This is an application that runs on a specific device.
- Web App - Web application. This is an application that runs on a browser.
- OECD - Organisation for Economic Co-operation and Development.
## Contents

1 Introduction ............................................. 1  
   1.1 Goal and purpose .................................. 1  
   1.2 Background ......................................... 2  
      1.2.1 Web application vs native application ..... 2  
      1.2.2 YouTube ...................................... 2  
   1.3 NUITEQ ............................................. 2  
   1.4 Limitations .......................................... 3  
   1.5 Method .............................................. 3  
   1.6 Social, Ethical and Environmental Considerations .... 4  
      1.6.1 Ethical considerations ......................... 4  
      1.6.2 Social considerations .......................... 4  
      1.6.3 Environmental considerations ................. 4  

2 Design and Implementation ............................. 5  
   2.1 Searching For Videos ............................... 5  
      2.1.1 Setting up the API ............................ 6  
      2.1.2 Quota ......................................... 7  
      2.1.3 Search ........................................ 7  
   2.2 The Player .......................................... 8  
      2.2.1 Player design .................................. 8  
      2.2.2 Player events .................................. 9  
      2.2.3 Player states .................................. 10  
      2.2.4 Loading the player ............................. 10  
      2.2.5 Starting a video ................................ 11  
   2.3 Buttons ............................................. 11  
   2.4 Saving a clip ....................................... 12  
      2.4.1 Showing the thumbnail ......................... 12  
   2.5 Playing videos in a row ............................ 12  

3 Results .................................................. 13  
   3.1 Searching for videos ............................... 13  
   3.2 Playing videos ...................................... 13  

4 Discussion ............................................. 15  
   4.1 Future work ......................................... 16  

5 Conclusion ............................................. 17  

6 References ............................................. 18
1 Introduction

The schools today are very different than they were 50 years ago. There is a lot more technology available that can be used to improve the education in today’s schools. Social studies teacher Mike Christiansen describes how he uses YouTube to help his students to learn better [1].

In 2012 72% of 15 year-old-students from OECD (Organisation for Economic Co-operation and Development) countries reported that they are using a computer in school while 96% of them have one at home [2] and in 2015 according to Pew Research centre, 73% of teens have access to a smartphone [3]. While the use of technology has increased both at home and at schools a report from OECD says that students that use computers very often at school do much worse than students who use computers moderately or rarely while the students that use computers moderately at school do better than the ones that use it rarely. OECD director for Education and Skills Andreas Schleicher said School systems need to find more effective ways to integrate technology into teaching and learning to provide educators with learning environments that support 21st century pedagogies and provide children with the 21st century skills the need to succeed in tomorrow’s world.[2]

With the rise of mobile devices such as smartphones together with their versatility and internet capabilities educators have started to take steps to minimise the downsides of them while praising the upsides [4].

The Video Remix Tool is an application that would fit very well in a classroom that uses the capabilities of smartphones. NUITEQs idea for the application is that it will be used by students and teachers to create summaries for different subjects.

The application is able to search for videos from YouTube and then save short clips from them. The clips can after they have been saved either be edited to change how long they are or annotations such as text and figures can be added to them. The clips can be moved around so they are in a different order. When the user is done the application gives them a link that can then be shared with other people.

1.1 Goal and purpose

The big goal of this project was to create an application that could be used by teachers and students to take existing YouTube videos about a subject and create small clips from them. By creating a lot of small clips they would be able to create a playlist of clips that explains a subject and then share it with other people.

NUITEQ wanted the application to have a few features such as:

- Search for and play YouTube videos.
- To change the speed the video is playing at.
- Create small clips from the videos.
- Be able to change the clip after it has been created.
- Play all the clips one after the other.
1.2 Background

The idea for this application came from NUITEQ because they wanted to create an application that could be used by teachers and students to create short clips from longer YouTube videos about a subject and then play them after each other as a summary about the subject. The users would then be able to share their creations with other people. This section covers the background for the application.

1.2.1 Web application vs native application

Native applications are applications that are made for a specific platform and then downloaded to the device. They are most often downloaded from an app store. Web applications, on the other hand, are made for browsers and do not have to be downloaded.

Each type of application has pros and cons. Some of the pros and cons described by Priya Viswabathan [5] are described here. Some pros and cons for native applications are that they are built for that specific device which means that they perform faster on that device than a web application, but they tend to be more expensive and they are harder to have on multiple platforms. Web applications are easier to maintain and they work on all different devices, but they can’t use all the features of a device so it will not be as fast as a native application.

The choice to make a web application came because NUITEQ wanted the application to work on more than one type of mobile device and a web application does that with much less work than a native application.

1.2.2 YouTube

YouTube[6] is a video sharing site that have existed since 2005 and since then a lot of videos has been uploaded to the site, including a lot of educational videos. Because of the amount of educational videos that were uploaded, YouTube created the channel YouTube EDU in 2009.

YouTube have two different kinds of licenses, their standard license [7] which allows YouTube to use a video to spread, change and use it for marketing. This license also gives users of YouTube permission to watch the video without having to pay. Their other license is a creative commons license [8] that allows other people to use a video with the license. They can reuse the video or edit it however they want. The standard license is the license all videos get if they don’t change it to creative commons.

1.3 NUITEQ

NUITEQ [9] is a world leading company in human-computer interaction and they are focused on natural user interfaces and multitouch technology. They have created the touchscreen software Snowflake multiteach [10] that is used to improve the way teachers teach, Snowflake business [11] which can be used to create interactive presentations and Snowflake entertainment [12] that includes games. Businesses and schools in over 70 countries use them.
1.4 Limitations

Because the application is a web application and has to work on mobile devices a few limitations show up. The first limitation is that not all users use the same browser and browsers do not do things the same, which means that the application needs testing on each browser. A limitation with mobile browsers is that some of them restrict features such as autoplay, this means that the videos only play when the user gives an input, such as pressing a button.

A bad internet connection or a slow internet is also a limitation. Because the videos start to buffer when the connection is bad, this decreases the user experience and can lead to people not using the application. This is seen more on mobile devices because they use wireless internet.

1.5 Method

The implementation process of this project has consisted of a few different iterative phases.

The first phase was a research phase that answered the questions, what platform will the application be on? What languages should the application be programmed in? Does the application need to use an API? When the questions had been answered the phase consisted of researching the API’s that the application would use, how they were used, what functions they had and how they worked. The phase was revisited when NUITEQ asked a question if something was possible.

After this, the next phase was the implementation phase where the API’s were used to implement the video player and the search functions of the application. When that was done the implementation was tested and if bugs were found the phase shifted to the implementation phase again.

Once a week during meetings with NUITEQ feedback about the application was given together with small priority lists and what should be done during the week. During these meetings questions about if something could be done was asked, and after the meeting the questions got researched. Then they were either implemented or deemed not possible to do. Figure 1 shows how the phases is connected.

Figure 1: Shows how the difference phases in the method is connected
1.6 Social, Ethical and Environmental Considerations

1.6.1 Ethical considerations

The ethical considerations are many. Because the videos used in the clips are taken from YouTube, the consideration comes into is it legal to use all videos from YouTube or is it only okay to use some with a special license. Another consideration is that mobile browsers and OS limits the autoplay feature so that videos can not start playing automatically and as this application needs to be able to play videos after each other it is worked around. Which can lead to the application using more data and then people will have to pay more.

1.6.2 Social considerations

The social impact this application could have is helping to modernize teaching in schools by going away from a teacher standing in front of the class and talking about the subject, the teacher can create a video about the subject that all the students can watch at their own speed so that the faster students can finish quickly and the students that have some troubles with the subject can rewatch the video.

1.6.3 Environmental considerations

From an environmental perspective, this application may help with decreasing the usage of paper because the teacher can tell the students to watch a video instead of giving out handouts on paper.
2 Design and Implementation

In this section, the design of the player is covered, together with how to search for videos and how to play videos with the player.

The languages used in the application is HTML5 for the base of the application, CSS for the design of the application and JavaScript for the functionality of the application.

The application window has five different areas. They are the menu, where buttons to the search screen and add title card. Next is the module part, which is the part where the search results show up. The next part is the video area which is the area that the video plays at. Next part is the control bar where all the buttons to control the videos are and lastly the timeline which shows all saved clips. In figure 2 the different parts are shown.

![Figure 2: The different areas of the application.](image)

For more information about the design of the application see my colleague Rebecka Delehags paper[13] and for more information about the structure of the application see my colleague Axel Lassinatis paper[14]

2.1 Searching For Videos

To search for videos the YouTube Data API v3 [15] is used. This section describes how to set up the API for use, explains about the quota cost for
using the API and how to search for videos with the API.

2.1.1 Setting up the API

To search for videos the application uses the YouTube Data API V3 [15]. A few steps have to be taken to use the API [16]:

1. The creator of the application has to have a Google Account and access the Google Developers Console [17]. Where the creator creates a project.

2. Obtain an API key by clicking on credentials and create API key. Optionally the key can be restricted so it only works on certain websites or IP addresses.

3. Lastly, enable the API that the project should use.

Now the API can be used by the application by including the script shown in figure 3.

```html
<script src="https://apis.google.com/js/client.js?onload=googleApiClientReady"></script>
```

Figure 3: The script that needs to be included to use the YouTube Data API

After the application has included the script the API key is set by calling the API function `gapi.client.setApiKey` and the API that should be used needs to be loaded by calling the function `gapi.client.load` and the parameters are which API that the application should use. When the API has finished loading the function `HandleAPILoaded` is called to show that the API is ready to use. In figure 4 the code for this is shown.

```javascript
//Init.
this.Init = function()
{
  try
  {
    //Youtube api key
    gapi.client.setApiKey("The Api Key goes here");
    //Which api that should be loaded
    gapi.client.load('youtube', 'v3', function () {
      handleAPILoaded();
    });
  }
  catch (err)
  {
    console.log(err);
  }
}
```

Figure 4: Sets the API key and loads the API that should be used.
2.1.2 Quota

To make sure that developers use the YouTube Data API correctly, and not creating applications that reduce the quality or limits access for others. The API uses a quota and each project that uses the API has a quota of 1 million units per day [18]. The quota can be increased in the Google Developer Console for the project.

Each request that the application makes with the API costs a certain amount of quota. So using as little quota as possible for each request is necessary if a lot of people use the application.

2.1.3 Search

The search screen is the screen you search from. It has a search field that the user uses to write what they want to search for. A search button that the user presses when they want to search and a drop down menu where they can choose which order the videos should show up in. The search screen is shown in figure 5.

When the user presses the search button the application takes the values from the search field and the selected value from the drop down menu. Then a request is made to the YouTube Data API by calling the function `gapi.client.youtube.search.list`. The function has a field parameter which limits the quota and bandwidth usage by decreasing the information that is sent to the user. The videoEmbeddable and videoSyndicated limits the videos that the API returns to the application to those that can be embeddable and played on another site than YouTube. The values from the search field and drop down menu is assigned to the parameters q and order respectively. The maxResults parameter decides the maximum number of results per page while the pageToken parameter is which page the user wants to search for. The type parameter tells the API to only return videos and not channels or playlists.

![Search Screen](image)

Figure 5: This is the search screen for the application.
When the request is set the application calls a function to see if there are any previous thumbnails on the search page and if there are it cleans them up. After this, the application requests the information and receives it and then goes through the information received and saves the information for each video.

The information received includes the ID, thumbnail, the published date and the title of each video.

After this, it concatenates all the video id’s it saved into a single string and then uses another API function called `gapi.client.youtube.video.list` to get more information such as duration and description for each video. Then the information is requested and for each video, a figure is created and added to an HTML5 div. Each figure takes the thumbnail that was saved for each video and also how long each video is. After this, the application creates the next and previous page buttons. More on them in section 2.3.

Now the application shows the thumbnails to the user and when a user presses a thumbnail the user is moved to another screen where the video that was chosen will play. In figure 6 the screen after a user has searched for something shown.

### 2.2 The Player

The video player used is created with help of the IFrame player API [19]. This section describes the design of the player, how to load a player and how to play a video in the player.

#### 2.2.1 Player design

The player has five buttons in order from left to right, step back 5 seconds, set start time, play and pause, set end time and step forward 5 seconds. Above the buttons is a progress bar that shows how much of the video that has

![Figure 6: This is the search screen after a user has searched for something.](image)
played and above that is the player itself. Above the video is a back button which takes the user back to the same page on the search screen as it was on before.

The player’s looks are decided when the player is created (see section 2.2.4) and is after that manipulated with CSS. In figure 7 the video is shown with black bars above and below the video. To avoid this the video is scaled up using CSS and then the sides is cut off as shown in figure 8.

![Figure 7: Player with black bars above and below video.](image)

![Figure 8: Player after it has been scaled up.](image)

### 2.2.2 Player events

The API fires events that the application subscribes to by adding event listeners that listen for those events. The event listeners are added when the
player is created (see section 2.2.4). The events have two properties, the first is target which tells the application which player sent the event and data that is a value relevant to the specific event. The events the application used are onReady that is sent when the player has finished loading, onError that is sent when there is an error with the player. The last one used is onState-Change which is sent when the player changes state. More on the states used in section 2.2.3.

2.2.3 Player states

The player has a number of states that it can be in. The states that exist for the player are:

- Unstarted, which is the state it is in when it first loads. The data value it has is -1.
- Ended, the state the player is in when the video has ended. It’s data value is 0.
- Playing, the player is in this state when it is playing. It’s data value is 1.
- Paused, the state of the player when the video is paused. It’s data value is 2.
- Buffering, the state of the player when the video is buffering. It’s data value is 3.
- Video cued, the state of the player when a video has been cued and is ready to start playing. It’s data value is 5.

2.2.4 Loading the player

To load the player the application has to include the script in figure 9.

![Figure 9: Script for Iframe API.](https://www.youtube.com/iframe_api)

When the script has finished loading it will call the API function `onYouTubeIframeAPIReady` and it is only then the application is allowed to start creating players.

The application creates a player by assigning a variable to the function `YT.Player`. The function takes an HTML5 div tag and replaces it with an iframe tag. Other parameters it takes is a video id, some player variables and events the player should listen for.

The player variables change things about the player. The player variables used are:

- Controls that hides the normal YouTube controls.
• Rel that stops the player from showing related videos when a video has finished playing.

• Showinfo hides information such as video name and who uploaded the video.

• iv_load_policy makes sure no annotation is shown in the video.

• Modestbranding takes away a few YouTube icons.

• Disablekb disables the keyboard so no shortcuts can be pressed.

2.2.5 Starting a video

The application can start a video in four different ways. The first is by calling the API function `playVideo` when the player sends the onReady event. The second is to call the API function `loadVideoById` and then the video will start playing when the video has finished loading. The third way is to use the API function `seekTo` which starts the video at the time specified in the function. The last way is to set a player variable autoplay to 1, then the video will start playing when the player has finished loading.

Sometimes YouTube saves the time that a user was on in a video. When this happens the video would start playing from that time with no regard of the settings of the player. To fix this a variable was introduced that checked if the player was newly loaded and then when the video starts the time the video is at is checked against the time it should be at and if is not at that time the function `seekTo` is used to go to the correct time.

2.3 Buttons

The application includes buttons that the user can press and that uses functions from both the Iframe and YouTube data API’s.

- The play button causes the video to start playing by calling the function `playVideo`.

- The pause button causes the video to pause by calling the function `pauseVideo`.

- The next and previous page buttons calls the search function again with their respective page token to get the correct page of the search.

- The set end and start time buttons calls the Iframe API function `getCurrentTime` to get the current time the video is at and then saves it as a variable.

- The jump back and forward buttons causes the video to jump back or forward five seconds by calling the Iframe functions `seekTo` and `getCurrentTime`.
The back button takes the user back to the search screen they were on before.

2.4 Saving a clip

To save a clip the user has to first press the set start time button and then the set end time button. After that a clip is created with the specific start and end time and a thumbnail is shown on the timeline to represent the clip.

2.4.1 Showing the thumbnail

To allow the thumbnail to be shown the application enables CORS[20] by setting `Header set Access-Control-Allow-Origin ***` in the server file "httpd.conf" and in the code set `textureloader.crossOrigin = '';`. After this "https://cors-anywhere.herokuapp.com/"[21] is added in front of each thumbnails source location. This API enables cross-origin requests to anywhere.

2.5 Playing videos in a row

To play multiple clips in a row the application creates one player for each clip.

Because some of the mobile phone browsers restrict autoplay functionality functions such as `loadVideoById`, setting the autoplay variable and calling the function `playVideo` when the onReady event fires will not start the video automatically. They will only work if they are called by the user doing something such as clicking on a button.

To make the clips be able to be played after each other the application uses the function `cueVideoById` which causes the video to load and then send a video cued event to the application, on all the clips except the first one which uses `loadVideoById` to start playing. When the application receives the video cued event the video is started and then pauses instantly which results in that the clips can play after each other without needing to press the play button between each clip.
3 Results

The application has almost all the features NUITEQ wanted. The features that have been implemented and work fully are:

- To be able to search for YouTube videos.
- To be able to watch the YouTube videos and create small clips from them.
- To be able to change the clip after it has been created.

The feature to be able to play clips after each other works correctly on a computer, but on a mobile phone, it does not. The last feature that is to be able to change the speed of the video clip was not implemented. This will be discussed in section 4. Except for these two features, the application works on all the devices it has been tested on.

3.1 Searching for videos

A user can search for YouTube videos by pressing the button in the upper left corner of the screen and when the user does that they are moved to the search screen. Once there the user can search for a keyword by writing in the search field and then press the search button to initiate the search. When the user presses the search button the application uses the YouTube Data API to request videos from YouTube and then presents thumbnails of the videos together with the name and how long the video is to the user. The user can then click on a thumbnail to watch that video or scroll down to the bottom of the screen and press the next or previous page buttons to see more videos for the keyword. If the user wants they can also change the order the videos is shown on the screen by choosing either relevance to the keyword, view count, rating or when the video was uploaded from the drop-down menu in the upper right corner.

3.2 Playing videos

When a user clicks on a thumbnail on the search screen they are moved to the import screen where the application creates a player for the video with the help of the IFrame API. When the player has finished loading the video the user can start playing it by either clicking on the video itself or by pressing the play button. The user can jump back and forward 5 seconds in the video by pressing the buttons to the farthest left and right in the controls area. They can jump further ahead or backwards by clicking on the progress bar below the video. When they want to save a clip they press the set start time button where they want the clip to start and the set end time button where they want the clip to end. Then a thumbnail is shown on the timeline to represent the clip. The user can then either create more clips from the same video, search for more videos by either clicking the search screen button in the upper left corner to search for a new keyword or they can press the back button in the upper right corner and that will take them back to the search page they were
on before. They can also if they want to click on the thumbnail representing
the clip and a player will show up where they can watch that clip. If the
user wants to they can add annotations such as text or figures that are shown
over the video, whenever a new annotation is added the clip the user is on
will restart playing. The user can also edit the clip and change the start and
end time of it. When they user presses the play button to the left of the
timeline they will be taken to a player where all the clips on the timeline will
play after each other and they can see how it looks. When pressing the done
button to the right of the timeline a link will be created and can be shared.
When someone clicks on the link they are brought to a final player that plays
all the clips after each other.
4 Discussion

This section will discuss the results, the implementation of the application and any problems with it and any future work for the application.

Most of the functionality for the application has been implemented and is working as it should. The feature that allows the user to play multiple clips after each other works very well on a computer, but on a mobile, it will not play each clip. Instead, the user has to press the play button to start each clip. This is because some mobile browsers and OS developers restrict the functionality that is auto play and only allows videos to start playing when a user interacts with something such as pressing a button. They do this to protect users data because many data providers charge money based on the amount of data a person uses. That it is restricted causes great problems for the application and disturbs the experience for the user on mobile devices.

The functionality that the user should be able to change the speed the video is playing at was at first implemented and worked correctly on computers, but when testing was done on mobiles it did not work. This is because all the videos have specific speeds they can play at and while on computers the videos were allowed to play at different speeds, on mobile YouTube restricts the speeds the videos can play at to 1. The IFrame API was no help trying to solve this because the functions it has that change the speed of the video is not guaranteed to change the speed even if it is called. The only solutions found that would be able to make the videos play at higher speeds on mobile was to download an external player and play the video through that or to change the browser window to desktop mode. Because of this, the functionality was removed.

The search functionality works very well and returns videos for the search word as it should and it filters away videos that are not playable, but even though the application tells the YouTube Data API to only return videos that can be watched on another site than YouTube, some videos is not playable and when the user tries to start them gets a message that this video can only be played on YouTube. This causes an annoyance for the user that might have wanted to make a clip out of that video.

Another problem that exists is that the video can sometimes buffer or take a long time to start on mobile devices. This is most likely because mobiles use wireless network which can be slower and less reliable than an ethernet connection.

A smaller problem that exists is that because the player is scaled to avoid black bars above and below, or to the sides of the video, some information from the videos is lost, such as the top of a head. This is fixed by not scaling the video and have the black bars or by making sure that the video plays in their correct aspect ratio such as 16/9 or 4/3 instead of as a square. This was not done because NUITEQ wanted the video to be scaled up.

The application has mostly been tested with the developer tools on Google Chrome, Firefox and Microsoft Edge. The tools have modes that mimic some things about different mobiles. The testing on real mobiles was mostly done on SONY phones because they were the only ones on hand to test with. To be able to release the application it needs to be tested on more mobiles and
browsers to see that it actually works on them.

4.1 Future work

There exist some work that needs to be done in the future before the application is ready to be released, such as fixing the problem with having to click the play button to start playing each clip on mobile devices, fix the scaling so it either scales correctly everywhere or change it so it does not scale at all and instead keeps the same aspect ratio, optimize the application as much as possible to decrease loading times and maybe decrease the time the video is spent buffering and make it possible for the user to record their voice over the video.

The order the future work should be implemented in is:

1. The clips can play after each other.
2. Fix the scaling.
3. Optimize the application.
4. Add voice recording.

They should be implemented in this order because if number 1 is not implemented the application will not have the ability to play clips after each other, and if the application can’t play clips after each other, why should a user use the application at all. The scaling should be implemented second because it looks bad if the videos are not scaled correctly. The optimization should be done after that because, while the application can be used without being optimized, the user experience will be increased with shorter loading and buffering times. To allow the user to record their voice should be implemented last because the application can be used without it, while 1 and 2 need to be fixed before the application can be released, and 3 is more important to fix than adding voice recording.
5 Conclusion

The project was mostly successful, most of the functionality that NUITEQ wanted has been implemented and is working as it should. If NUITEQ wants to use this application they have to do some additional work to be able to release it.

An application like this has a lot of applications. It can be used by teachers to make short videos that the students can then watch at their own speed, this also frees up the teacher to walk around and help more than if they are standing in front of the class. It can be used by students, to help them learn a subject by making them look up information about it at then create a summary about the subject. The application can also be used by people not in school by allowing them to create short videos of clips that they like, such as cool tricks or nice goals in some sport and then share it with other people.

The limits that are placed on some phones and mobile browsers hinders some type of applications so that they can’t reach their full potential. The limitation of autoplay features on mobile devices hinders this application from reaching it’s full potential, and even if there exists workarounds for it, those themself are limiting in what is possible to work around.
6 References


