Shaka Player Project Ideas

Google Summer of Code

Updated: Mar 23rd 2021

Update 03.21:

1. We've added a few projects to our list. Please hold to start working on the projects until the GSoC program officially starts.
2. Now we have 4 projects, and each project has one or more tasks, so that you have more flexibility to choose some of or all the tasks in one project for your proposal. We've grouped the UI tasks into one larger UI project.
3. We've also added a proposal template, which is optional to use.
4. We encourage you to focus on the listed projects now. If you have a work-in-progress PR, you can include it when applying for GSoC. However, please understand that we have limited capacity now and would not be able to provide close guidance until the GSoC officially starts in the summer. You are welcome to send us a PR once you have a complete, fully-tested, ready-for-review solution as a general contributor.
5. If you have questions, please send it to the Google group.
6. Please be mindful of posting on the Github page just to express your interest or just to sync up. It may spam other subscribed developers, and also confuse other students who are applying for GSoC at the same time.
7. Please be kind and only @us for urgent issues on Github regarding your questions or PRs, since that might spam the team members who are not GSoC mentors.

Slack channel:

The video-dev is a Slack workspace for general video developers https://video-dev.herokuapp.com/ , and you can join the **#shaka-player** channel (https://video-dev.slack.com/archives/C01QRAFHLQK) in the video-dev workspace. You can post

your questions at the #shaka-player channel. However, we don't use Slack as a primary tool for communication, so please expect a slower response.

Introduction

This is a collection of project ideas for Google Summer of Code 2021. These projects are intended to be good starting projects for new Shaka Player contributors, to complete relatively independently in a few weeks. Familiarity with Javascript, Typescript or Python will be helpful, but not required.

To start, checkout our demo and tutorial page, and the contributing guide.
If you have any questions, you can discuss with us via the Google Group, or with the GSoC mentor (michellezhuo@google.com).

Project Ideas list

- Generate Typescript Type Definitions
- Shaka Player HLS Enhancements
- Support Fallback with Stream Alternates
- Support HLS identity key format
- Support AES-128 on HLS
- Shaka Player UI Library Enhancement
- Add Thumbnail Support for UI
- Add Overflow Menu Buttons to the UI Control Panel
- Add Video Stats Panel in UI
- Shaka Streamer Enhancements
- Offer hermetic copies of Shaka Packager and FFmpeg
- Support Concatenation of Inputs

New Projects

Project [New]: Generate Typescript Type Definitions

Estimated complexity: hard
Languages: Javascript, Typescript, Python

Some developers would like to use shaka with a typescript project, but the generated type definitions are incomplete and don't work yet. We want to generate the type definitions for Typescript with an automated solution.

Phase 1: We need automated tests to verify that typescript definitions are working and complete.
Phase 2: We need the automated generated definitions.

Possible solution 1:

We already have automation for the Closure-equivalent. Please see build/generateExterns.js. We can extend this to generate typescript typings, or generate typescript typings from the output of this tool (dist/shaka-player.compiled.externs.js).

Possible solution 2:

Use the Typescript tool to generate .d.ts files from .js files using JSDoc syntax.
Tutorials:
Creating .d.ts Files from .js files
Generating TypeScript Definition Files from JavaScript

Github tracking: Github issue , Pull Request

Project [New]: Shaka Player HLS Enhancements

Shaka Player is an open-source JavaScript library for adaptive media. It plays adaptive media formats (such as DASH and HLS) in a browser, without using plugins or Flash. Instead, Shaka Player uses the open web standards MediaSource Extensions and Encrypted Media Extensions. Shaka Player also supports offline storage and playback of media using IndexedDB. Content can be stored on any browser. Storage of licenses depends on browser support. Our main goal is to make it as easy as possible to stream adaptive bitrate video and audio using modern browser

technologies. We try to keep the library light, simple, and free from third-party dependencies. Everything you need to build and deploy is in the sources.

HLS: HTTP Live Streaming (also known as HLS) is an HTTP-based adaptive bitrate streaming communications protocol developed by Apple Inc. and released in 2009. Support for the protocol is widespread in media players, web browsers, mobile devices, and streaming media servers. As of 2019, an annual video industry survey has consistently found it to be the most popular streaming format.

The HLS specification specifies the data format of the files. For example, you can see the "identity key format" here. You don't have to read the whole specification, searching for the keyword like "identity" can help you to find the useful section in the specification.

In our code base, hls_parser.js parses the HLS manifest.

You can choose **2 or 3** sub projects from the HLS Enhancements.

Estimated complexity: medium
Languages: Javascript

Support Fallback with Stream Alternates

According to HLS specification one should be able to specify multiple variant-streams, and if one of them is down the player should pick another variant-stream. This does not seem to be the case for Shaka Player.

From the official HLS specification:

Not only do alternate media playlists in your master playlist operate as bandwidth or device alternatives, the alternates also serve as failure fallbacks. If the player cannot reload the media playlist file—due to issues such as 404 errors, server crashing, or a content distributor node problem—the player attempts to switch to another compatible media playlist provided on a different server. Offering multiple media playlists with the same bandwidth, the player switches to an identical playlist, providing consistent stream performance.

https://github.com/google/shaka-player/issues/2541

Support HLS identity key format

Add support for the clear-key format of HLS:

```
#EXT-X-KEY:METHOD=SAMPLE-AES-CTR,URI="data:text/plain;base64,bXbyXLF/Xha46u9rv1gtjg==",KEYFORMAT="identity"
```

Starting instructions:

https://github.com/google/shaka-player/issues/2146#issuecomment-635663638

https://github.com/google/shaka-player/issues/2146

Support AES-128 on HLS

With the increase of piracy, protecting media content is one of the key concerns of many publishers. The most popular method for content protection with the HTTP Live Streaming (HLS) protocol is AES-128 content encryption.

https://www.theoplayer.com/blog/content-protection-for-hls-with-aes-128-encryption

https://github.com/google/shaka-player/issues/850

Existing Projects

Shaka Player UI Library Enhancement

Update:

Our UI library is not an end-user application UI. Other video application developers use the Shaka Player UI library, and build customized UI and UX on top of it. We need to be very careful about making UI and style changes, not to break any existing applications.

* You can choose 2 or 3 tasks in the UI Library Enhancement project for your proposal.

Original:

Shaka Player UI library provides a high-quality accessible localized UI layer for the applications. It is an alternate bundle from the base Shaka Player library, that adds additional UI-specific classes and a streamlined declarative style of setup.

[NEW] Add Thumbnail Support for UI

Thumbnail track support is implemented in the player, and we want to show it in UI.

Reference: https://github.com/google/shaka-player/issues/559

Add Overflow Menu Buttons to the UI Control Panel

Estimated complexity: medium
Languages: Javascript

Currently the Shaka UI library has two types of buttons:

- Control panel buttons
- Overflow menu buttons

The overflow menu buttons are designed to display in the overflow menu, such as the Language button, the Playback Speed button, etc. You can see them by clicking the dot button on bottom right. The control panel buttons are designed to display on the main control panel, such as Volumes bar, the Fullscreen button, etc. The overflow menu buttons have dropdown menus, and are implemented with a different structure than the control panel buttons.

Currently, the application developers cannot add an overflow menu button to the control panel easily. This project will enable adding the overflow menu buttons to the control panel.

A possible solution would be having two versions of each button, to support placing them wherever the application developers want.

[Github Issue]

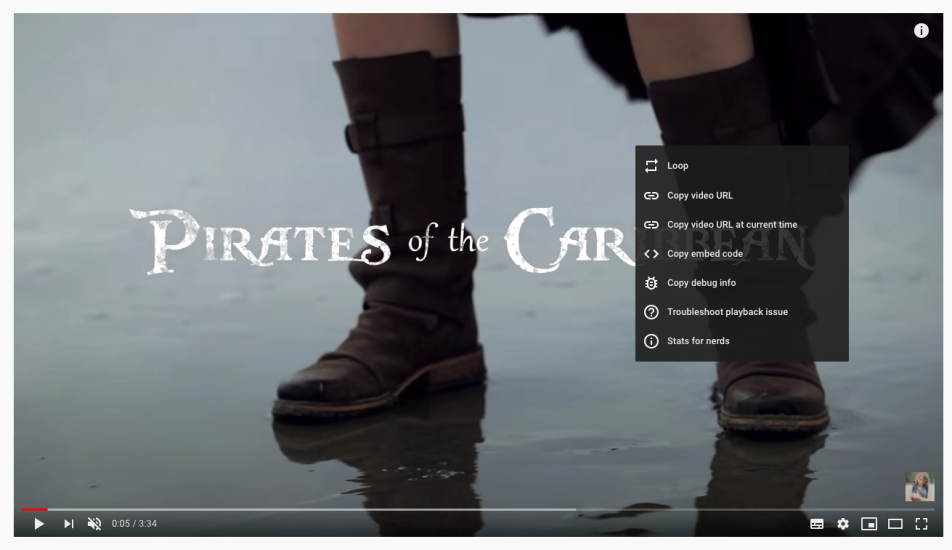
Add Video Stats Panel in UI

Estimated complexity: medium

Languages: Javascript

Similar to Youtube “Stats for nerds”, this project will implement a panel to display the video stats as part of the UI library. The users can see a menu on right click, and choose to display the Stats panel. This project involves:

1. implementing an overflow menu on right click on the video component
2. displaying an overflow menu with stats data on the video component



[Github Issue]

Add Close Button to the UI Library

Status: Closed

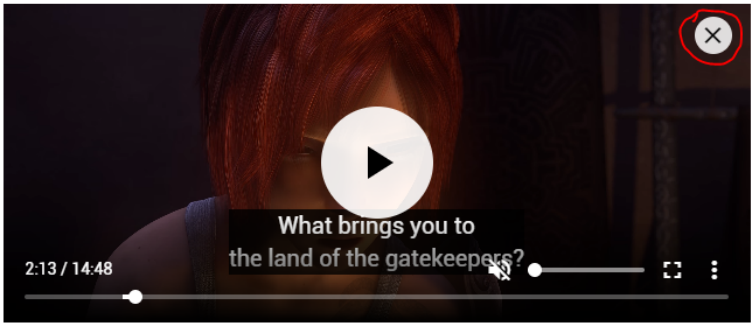
(Please be aware that this task is closed now. Thank you for your understanding!)

Estimated complexity: easy

Languages: Javascript

The Shaka UI library provides a few buttons for the application developer to add onto their UI, such as the Play/Pause button, Fullscreen button, etc. This project will add the Close button as a new component of the UI library, similar to the Close button on the Shaka demo page. The Close button will allow users to stop playing the video and close the video element.

The implementation of the demo page Close button is a good starting point for reference.



Shaka Streamer Enhancement

Shaka Streamer offers a simple config-file based approach to preparing streaming media. It greatly simplifies the process of using FFmpeg and Shaka Packager for both VOD and live content.

Shaka Streamer connects FFmpeg and Shaka Packager in a pipeline, such that output from FFmpeg is piped directly into the packager, and packaging and transcoding of all resolutions, bitrates, and languages occur in parallel.

Live documentation can be found here.

You can choose 1 or 2 tasks in the UI Library Enhancement project for your proposal.

[NEW] Offer hermetic copies of Shaka Packager and FFmpeg

Estimated complexity: hard

Languages: Python

Currently, users must install Shaka Packager and FFmpeg separately and system-wide in order to use Shaka Streamer.

If permitted by their licenses, we should try to bundle binaries of Shaka Packager and FFmpeg for convenience, and offer a command-line option to choose between hermetic versions and installed system versions.

Github issue: https://github.com/google/shaka-streamer/issues/60

Support Concatenation of Inputs

Estimated complexity: hard

Languages: Python

We'll allow Shaka Streamer to take a new input type with a list of media files, stitch the multiple media files together, and convert them to a m3u8 or mpd playlist as a continuous output.

For example, the input config can be:

inputs:

```
    # The type of input.
  - input_type: concat
    # The media type is required at this level only.
    media_type: video
    list:
      # These only need to have "name" attributes.
      -name: foo1.mp4
      -name: foo2.mp4
       is_interlaced: True  # If you have an interlaced source in the list

    # The type of input.
  - input_type: concat
    # The media type is required at this level only.
    media_type: audio
    language: "de"
    list:
      # These only need to have "name" attributes.
      -name: foo1.mp4
      -name: foo2.mp4
       track_num: 2  # If the 0th audio track is not the one you want here...
```

At the output, we will get an hls / dash playlist, with the three media sources stitched together.

There are several ways to concatenate things in ffmpeg, with various limitations.

1. Concat demuxer (same codecs, same time base, "etc" (ffmpeg doc is vague))
2. Concat protocol (same file format, only concatenate the formats like TS supported, analogous to "cat" command)
3. Concat filter (same resolution required, otherwise no restrictions)
4. External ffmpeg process (pre-encoding everything to match parameters first, then streaming it as one stream)

Github Issue

Add Support for More Audio Codecs

Status: Closed. Thank you for your understanding!

Estimated complexity: medium

Languages: Python

Currently Shaka Streamer supports a few video and audio codecs, and we'll expand our support for the audio codecs of ac-3 and ec-3.

Proposal Template

In the project proposal section, please include:

- Your understanding of the project
- Break-down of the steps
- How to test the results

In the self-introduction section, you are welcome to answer the following **optional** questions.

1. If you're selected as a GSoC student, would you like to work on other tasks besides the projects of your choice?
2. If you're not selected as a GSoC student, would you like to work on the projects as a general contributor?
3. What's your plan for the one year or two after GSoC (for example, Sep 2021 - Sep 2022) ?
4. Would you like to contribute to Shaka Player in the long term, after the GSoC ends?
5. What motivated you the most towards applying for GSoC?
    - Get recognized as a GSoC participant
    - Get the stipend
    - Work on an open source project
    - Work on a video streaming project
    - Other (Please let us know).
6. Please include your Github PR or Issue links on Shaka Player / Shaka Streamer (posted **before March 28th**) if you have any. (This is completely optional, please don't create a PR/Issue just to "show up" on our Github).