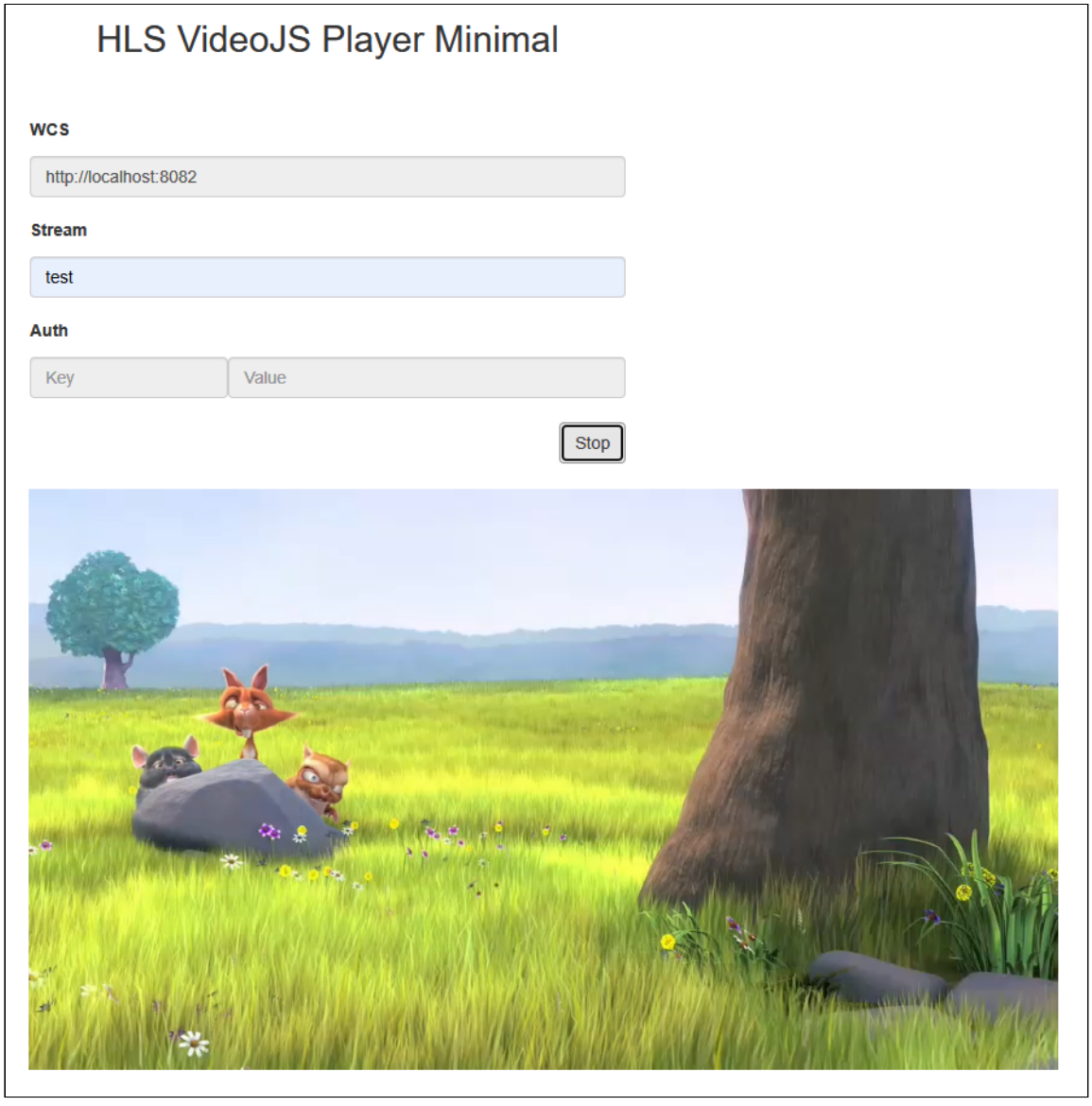
HLS VideoJS Player

Пример конвертации видеопотока в HLS и отображения его в браузере при помощи VideoJS

Данный плеер демонстрирует возможности WCS по преобразованию опубликованного на сервере потока в HLS и воспроизведению его в браузере. Нарезка потока в HLS запускается автоматически, при обращении к потоку, опубликованному на сервере, по HLS URL, например, для потока на рисунке ниже `http://localhost:8082/test/test.m3u8`



Код примера

Код данного примера находится на сервере по следующему пути:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/hls-player

- hls-player.css - файл стилей страницы с плеером
- video-js.css - файл стилей HLS-плеера
- hls-player.html - страница с плеером
- hls-player.js - скрипт, обеспечивающий запуск плеера
- player-page.html - общие элементы страницы плеера для трех примеров воспроизведения HLS
- video.js - скрипт, обеспечивающий работу плеера (http://videojs.com/, Apache License Version 2.0)
- videojs-hls.min.js - скрипт, обеспечивающий работу плеера (минимизированная версия)

Тестировать данный пример можно по следующему адресу:

https://host:8888/client2/examples/demo/streaming/hls-player/hls-player.html

Здесь host - адрес вашего WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла `hls-player.js` с хешем `ecbadc3`, которая находится здесь и доступна для скачивания в соответствующей сборке 2.0.212.

1. Определение HLS URL сервера

`getHLSUrl()` code

```
function initPage() {
    $("#header").text("HLS VideoJS Player Minimal");
    $("#urlServer").val(getHLSUrl());
    ...
}
```

2. Инициализация плеера

`videojs()` code

Плееру передается имя div-элемента, в котором должен быть проигран поток.

```
function initPage() {
    ...
    var remoteVideo = document.getElementById('remoteVideo');
    remoteVideo.className = "video-js vjs-default-skin";
    player = videojs(remoteVideo);
}
```

3. Определение имени потока (должен быть опубликован на сервере)

`encodeURIComponent()` code

```
function playBtnClick() {
    if (validateForm()) {
        var streamName = $('#playStream').val();
        streamName = encodeURIComponent(streamName);
        ...
    }
}
```

4. Формирование URL HLS-потока и запуск плеера

`player.play()` code

Если указаны ключ и токен авторизации, они будут включены в URL потока

```
function playBtnClick() {
    if (validateForm()) {
        ...
        var videoSrc = $("#urlServer").val() + '/' + streamName + '/' +
streamName + '.m3u8';
        var key = $('#key').val();
        var token = $("#token").val();
        if (key.length > 0 && token.length > 0) {
            videoSrc += "?" + key + "=" + token;
        }
        player.src({
            src: videoSrc,
            type: "application/vnd.apple.mpegurl"
        });
        console.log("Play with VideoJs");
        player.play();
        onStarted();
    }
}
```

5. Остановка воспроизведения

`player.dispose()` code

Этот метод удаляет со страницы div элемент, в котором был ранее инициализирован плеер

```
function stopBtnClick() {
    if (player != null) {
        console.log("Stop VideoJS player");
        //player.pause();
        player.dispose();
    }
    onStopped();
}
```

6. Создание нового `div` элемента и плеера в нем после остановки предыдущего плеера

code

```
function createRemoteVideo(parent) {
    remoteVideo = document.createElement("video");
    remoteVideo.id = "remoteVideo";
    remoteVideo.width=852;
    remoteVideo.height=480;
    remoteVideo.controls="controls";
    remoteVideo.autoplay="autoplay";
    remoteVideo.type="application/vnd.apple.mpegurl";
    remoteVideo.className = "video-js vjs-default-skin";
    parent.appendChild(remoteVideo);
    player = videojs(remoteVideo);
}
```