

Screen Sharing with Camera

Описание

Пример демонстрирует сценарий презентации: публикация потоков с веб-камеры и с экрана на одной странице с возможностью микширования этих двух потоков на сервере.

Параметры публикации экрана:

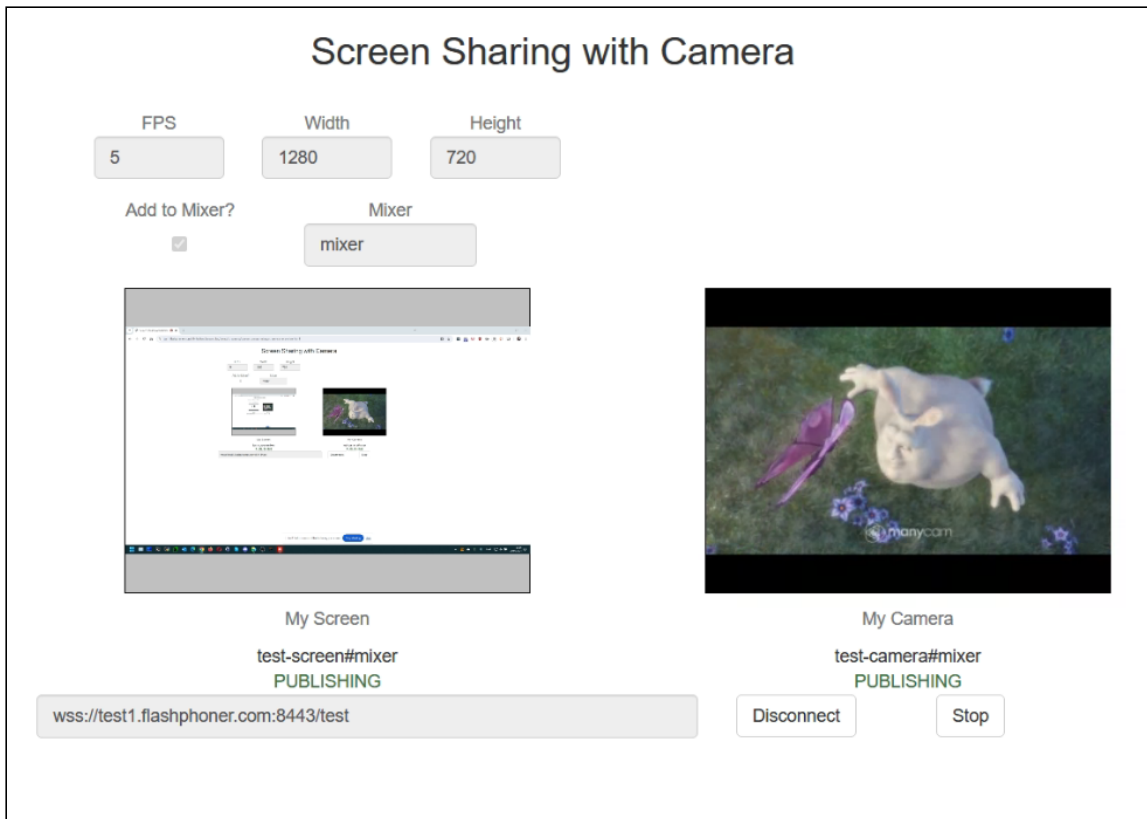
- `FPS` - частота кадров в секунду
- `Width` - ширина картинки
- `Height` - высота картинки

Параметры микширования:

- `Add to mixer` - добавлять ли потоки автоматически в микшер
- `Mixer` - имя микшера

Параметры соединения:

- Websocket URL WCS сервера



Код примера

Код данного примера находится на WCS-сервере по следующему пути:

```
/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/screen-camera-mixer
```

- `screen-camera-mixer.css` - файл стилей
- `screen-camera-mixer.html` - HTML страница примера
- `screen-camera-mixer.js` - скрипт, обеспечивающий работу примера

Тестировать данный пример можно по следующему адресу:

```
https://host:8888/client2/examples/demo/streaming/screen-camera-mixer/screen-camera-mixer.html
```

Здесь host - адрес WCS-сервера.

Работа с кодом примера

Для разбора кода возьмем версию файла `screen-camera-mixer.js` с хешем 32144d9, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке 2.0.243.

1. Инициализация API.

`Flashphoner.init()` [code](#)

```
Flashphoner.init();
```

2. Подключение к серверу

`Flashphoner.createSession()` [code](#)

```
const connect = function() {
  ...
  let url = field("url");
  ...
  console.log("Create new session with url " + url);
  Flashphoner.createSession({urlServer:
url}).on(SESSION_STATUS.ESTABLISHED, function(session){
  ...
  }).on(SESSION_STATUS.DISCONNECTED, function(){
  ...
  }).on(SESSION_STATUS.FAILED, function(){
  ...
  });
}
```

3. Получение от сервера события, подтверждающего успешное соединение

`STREAM_STATUS.ESTABLISHED` [code](#)

```
const connect = function() {
  ...
  let url = field("url");
  ...
  console.log("Create new session with url " + url);
  Flashphoner.createSession({urlServer:
url}).on(SESSION_STATUS.ESTABLISHED, function(session){
  //session connected, start streaming
  setStatus("screen", SESSION_STATUS.ESTABLISHED);
  setStatus("camera", SESSION_STATUS.ESTABLISHED);
  onConnected(session);
  }).on(SESSION_STATUS.DISCONNECTED, function(){
  ...
  }).on(SESSION_STATUS.FAILED, function(){
  ...
  });
}
```

4. Публикация видеопотока с экрана

`Session.createStream()`, `Stream.publish()` code

Методу `createStream()` передаются следующие параметры:

- `streamName` - имя публикуемого потока
- `constraints.video.width` - ширина картинки
- `constraints.video.height` - высота картинки
- `constraints.video.frameRate` - FPS публикации
- `constraints.video.type: "screen"` - тип потока: публикация экрана
- `constraints.video.withoutExtension: true` - публикация экрана из браузера без использования расширения
- `constraints.video.mediaSource: "screen"` - только для публикации из Firefox
- `localVideoScreen` - элемент для локального отображения потока
- `disableConstraintsNormalization = true` - отключение нормализации параметров (только для MacOS Safari)

```
const startStreamingScreen = function(session) {
  let streamName = getStreamName("screen", field("url"));
  let constraints = {
    video: {
      width: parseInt($('#width').val()),
      height: parseInt($('#height').val()),
      frameRate: parseInt($('#fps').val()),
      type: "screen",
      withoutExtension: true
    }
  };
  if (Browser.isFirefox()) {
    constraints.video.mediaSource = "screen";
  }
  let options = {
    name: streamName,
    display: localVideoScreen,
    constraints: constraints
  }
  if (isSafariMacOS()) {
    options.disableConstraintsNormalization = true;
  }
  session.createStream(options).on(STREAM_STATUS.PUBLISHING,
function(screenStream) {
  ...
}).on(STREAM_STATUS.UNPUBLISHED, function() {
  ...
}).on(STREAM_STATUS.FAILED, function(stream) {
  ...
}).publish();
}
```

5. Получение от сервера события, подтверждающего успешную публикацию экрана

`STREAM_STATUS.PUBLISHING` [code](#)

По этому событию начинается публикация потока с камеры

```
const startStreamingScreen = function(session) {
  ...
  session.createStream(options).on(STREAM_STATUS.PUBLISHING,
function(screenStream) {
  /*
   * User can stop sharing screen capture using Chrome "stop" button.
   * Catch onended video track event and stop publishing.
   */
  document.getElementById(screenStream.id()).srcObject.getVideoTracks()
[0].onended = function (e) {
    screenStream.stop();
  };
  document.getElementById(screenStream.id()).addEventListener('resize',
function(event){
    resizeVideo(event.target);
  });
  setStatus("screen", STREAM_STATUS.PUBLISHING, screenStream);
  startStreamingCamera(session, screenStream);
}).on(STREAM_STATUS.UNPUBLISHED, function() {
  ...
}).on(STREAM_STATUS.FAILED, function(stream) {
  ...
}).publish();
}
```

6. Публикация видеопотока с камеры

`Session.createStream()`, `Stream.publish()` [code](#)

Методу `createStream()` передаются следующие параметры:

- `streamName` - имя публикуемого потока
- `localVideoCamera` - элемент для локального отображения потока

```
const startStreamingCamera = function(session, screenStream) {
  let streamName = getStreamName("camera", field("url"));
  let options = {
    name: streamName,
    display: localVideoCamera
  }
  session.createStream(options).on(STREAM_STATUS.PUBLISHING,
function(cameraStream) {
  ...
}).on(STREAM_STATUS.UNPUBLISHED, function() {
  ...
}).on(STREAM_STATUS.FAILED, function(stream) {
```

```
...
}).publish();
}
```

7. Получение от сервера события, подтверждающего успешную публикацию камеры

`STREAM_STATUS.PUBLISHING` [code](#)

```
const startStreamingCamera = function(session, screenStream) {
  ...
  session.createStream(options).on(STREAM_STATUS.PUBLISHING,
function(cameraStream) {
  document.getElementById(cameraStream.id()).addEventListener('resize',
function(event){
  resizeVideo(event.target);
  });
  setStatus("camera", STREAM_STATUS.PUBLISHING, cameraStream);
  onStart(cameraStream);
}).on(STREAM_STATUS.UNPUBLISHED, function() {
  ...
}).on(STREAM_STATUS.FAILED, function(stream) {
  ...
}).publish();
}
```

8. Остановка публикации камеры

`Stream.stop()` [code](#)

```
const setPublishButton = function(action, session, cameraStream) {
  $("#publishBtn").text(action).off('click').click(function(){
    if (action == "Start") {
      ...
    } else if (action === "Stop") {
      $(this).prop('disabled', true);
      cameraStream.stop();
    }
  }).prop('disabled', false);
}
```

9. Остановка публикации экрана

`Stream.stop()` [code](#)

```
const startStreamingCamera = function(session, screenStream) {
  ...
  session.createStream(options).on(STREAM_STATUS.PUBLISHING,
function(cameraStream) {
  ...
}
```

```
}).on(STREAM_STATUS.UNPUBLISHED, function() {
    setStatus("camera", STREAM_STATUS.UNPUBLISHED);
    screenStream.stop();
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).publish();
}
```