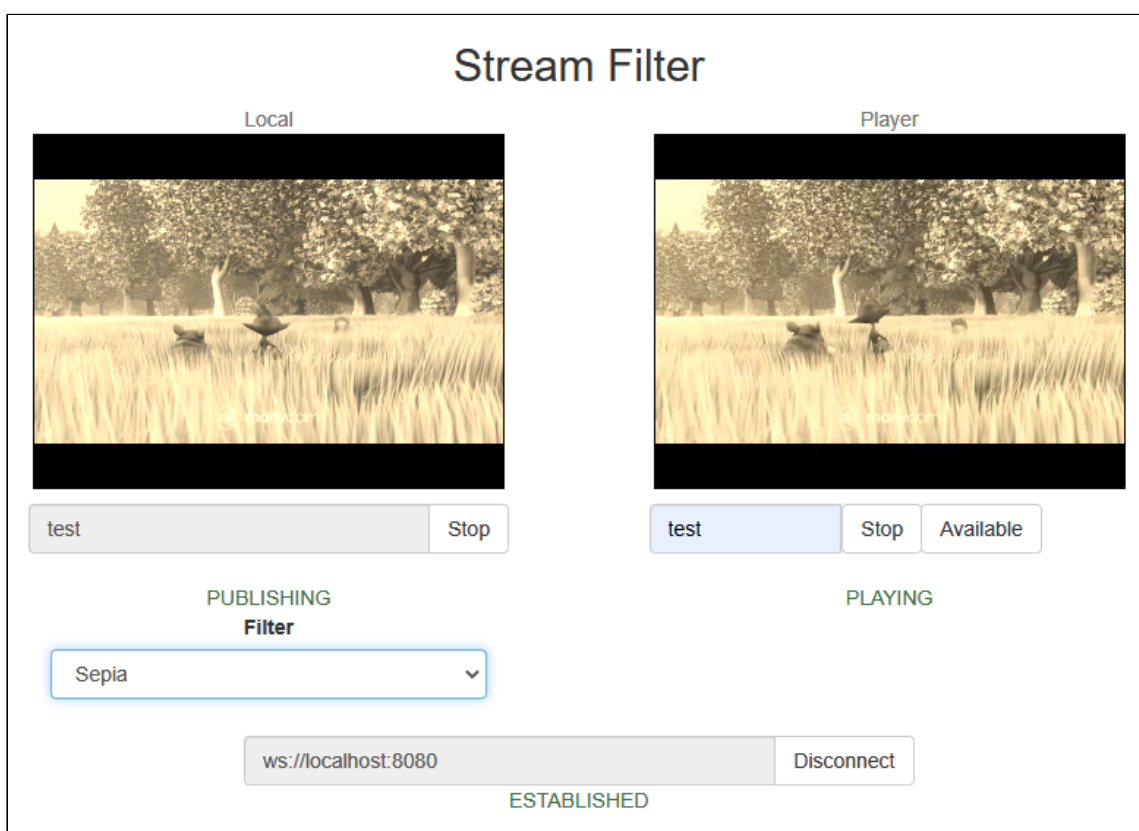


# Stream Filter

## Пример стримера с применением фильтров к изображению

Данный пример показывает, как применить фильтр или любое другое изменение к изображению при публикации потока с использованием элемента canvas



Данная возможность работает во всех основных браузерах, за исключением iOS Safari 12

## Код примера

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

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/stream\_filter

- stream\_filter.css - файл стилей
- stream\_filter.html - страница клиента

- `stream_filter.js` - скрипт, обеспечивающий работу примера

Тестировать данный пример можно по следующему адресу:

[https://host:8888/client2/examples/demo/streaming/stream\\_filter/stream\\_filter.html](https://host:8888/client2/examples/demo/streaming/stream_filter/stream_filter.html)

Здесь `host` - адрес WCS-сервера.

## Работа с кодом примера

Для разбора кода возьмем версию файла `stream_filter.js` с хешем `ecbadc3`, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке `2.0.212`.

### 1. Инициализация API

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

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

### 2. Подключение к серверу

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

```
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

`ConnectionStatusEvent ESTABLISHED` [code](#)

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function(session){
    setStatus("#connectStatus", session.status());
    onConnected(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
}).on(SESSION_STATUS.FAILED, function(){
    ...
});
```

## 4. Публикация видеопотока

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

При создании передаются:

- `streamName` - имя видеопотока
- `localVideo` - `div` элемент, в котором будет отображаться видео с камеры.

Для применения фильтров видео, захваченное с камеры, отрисовывается на канвасе при помощи опции `useCanvasMediaStream: true`

```
session.createStream({
  name: streamName,
  display: localVideo,
  cacheLocalResources: true
  receiveVideo: false,
  receiveAudio: false,
  useCanvasMediaStream: true
}).publish();
```

## 5. Получение от сервера события, подтверждающего успешную публикацию потока

`StreamStatusEvent PUBLISHING` [code](#)

По данному событию запускается отрисовка на канвасе с FPS 30

```
session.createStream({
  ...
}).on(STREAM_STATUS.PUBLISHING, function(stream){
  setStatus("#publishStatus", STREAM_STATUS.PUBLISHING);
  onPublishing(stream);
  intervalId = setInterval(draw, 1000.0 / 30);
}).on(STREAM_STATUS.UNPUBLISHED, function(){
  ...
}).on(STREAM_STATUS.FAILED, function(){
  ...
}).publish();
```

## 6. Воспроизведение видеопотока

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

При создании передаются:

- `streamName` - имя видеопотока (в том числе, это может быть имя потока, опубликованного выше)
- `remoteVideo` - `div` элемент, в котором будет отображаться видео

```
session.createStream({
  name: streamName,
  display: remoteVideo
  ...
}).play();
```

## 7. Получение от сервера события, подтверждающего успешное воспроизведение потока

`StreamStatusEvent PLAYING` [code](#)

```
session.createStream({
  name: streamName,
  display: remoteVideo
  ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
  setStatus("#playStatus", stream.status());
  onPlaying(stream);
}).on(STREAM_STATUS.STOPPED, function() {
  ...
}).on(STREAM_STATUS.FAILED, function() {
  ...
}).play();
```

## 8. Остановка воспроизведения видеопотока

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

```
function onPlaying(stream) {
  $("#playBtn").text("Stop").off('click').click(function(){
    $(this).prop('disabled', true);
    stream.stop();
  }).prop('disabled', false);
  $("#playInfo").text("");
}
```

## 9. Получение от сервера события, подтверждающего успешную остановку воспроизведения потока

`StreamStatusEvent STOPPED` [code](#)

```
session.createStream({
  ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
  ...
}).on(STREAM_STATUS.STOPPED, function() {
  setStatus("#playStatus", STREAM_STATUS.STOPPED);
  onStopped();
}).on(STREAM_STATUS.FAILED, function() {
```

```
...
}).play();
```

## 10. Остановка публикации видеопотока

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

```
function onPublishing(stream) {
  $("#publishBtn").text("Stop").off('click').click(function(){
    $(this).prop('disabled', true);
    stream.stop();
  }).prop('disabled', false);
  $("#publishInfo").text("");
}
```

## 11. Получение от сервера события, подтверждающего успешную остановку публикации потока

`StreamStatusEvent UNPUBLISHED` [code](#)

```
session.createStream({
  ...
}).on(STREAM_STATUS.PUBLISHING, function(stream){
  ...
}).on(STREAM_STATUS.UNPUBLISHED, function(){
  setStatus("#publishStatus", STREAM_STATUS.UNPUBLISHED);
  onUnpublished();
}).on(STREAM_STATUS.FAILED, function(){
  ...
}).publish();
```

## 12. Отрисовка изображения на канвасе и применение фильтра

`draw()` [code](#)

```
function draw() {
  let localVideo = document.getElementById('localVideo');
  let canvas = localVideo.children[0];
  if (canvas) {
    let ctx = canvas.getContext('2d');
    // First need to draw video on the canvas
    ctx.drawImage(canvas.children[0], 0, 0);
    // next get image data
    let imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
    // next need to apply filter to the image
    let filtered = currentFilter(imageData);
    // and finally draw filtered image on the canvas
    ctx.putImageData(filtered, 0, 0);
  }
}
```

### 13. Инициализация списка и выбор фильтра

`applyFilter()` code

```
var filters = [empty, sepia, threshold, invert];
var currentFilter = empty;
...
function applyFilter() {
  let filter = $('#filter').val();
  currentFilter = filters[filter];
}

function empty(imageData) {
  return imageData;
}
```