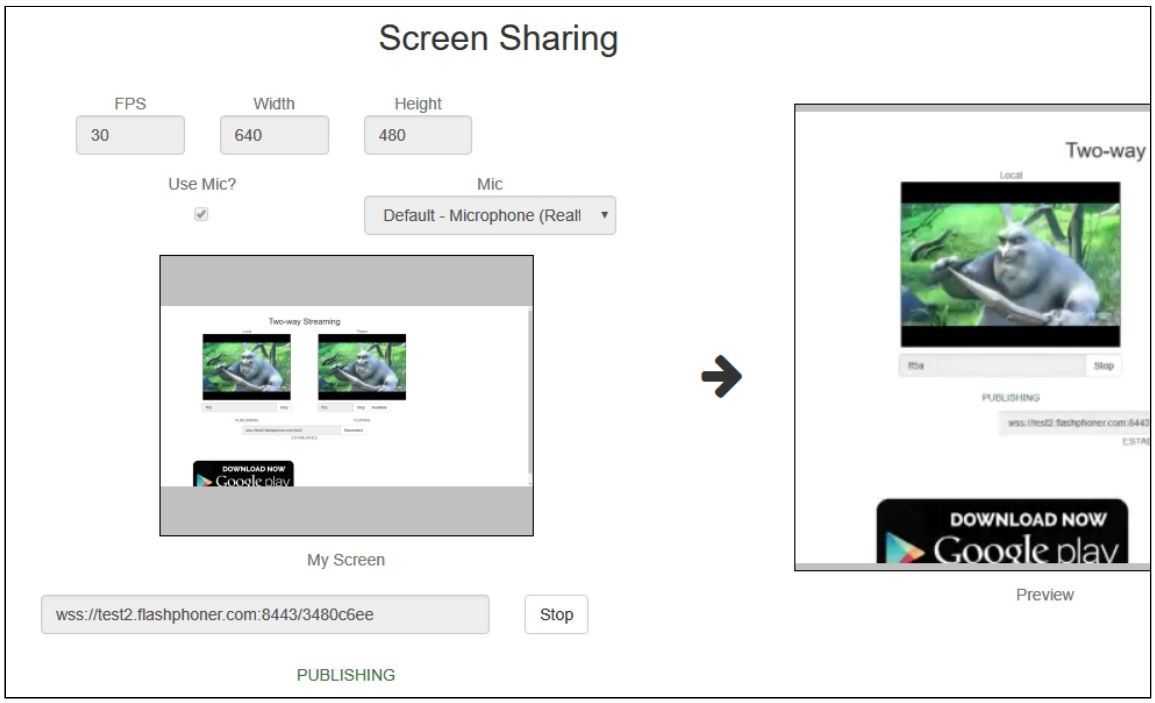
Screen Sharing

Пример демонстрации экрана браузера

Демонстрация экрана доступна в браузерах Chrome и Firefox. Для демонстрации экрана в Chrome до версии 73 необходимо собрать и установить расширение, версии Chrome 73 и выше, а также Firefox и Safari в расширениях не нуждаются. В настоящее время использование расширения не рекомсендуется.



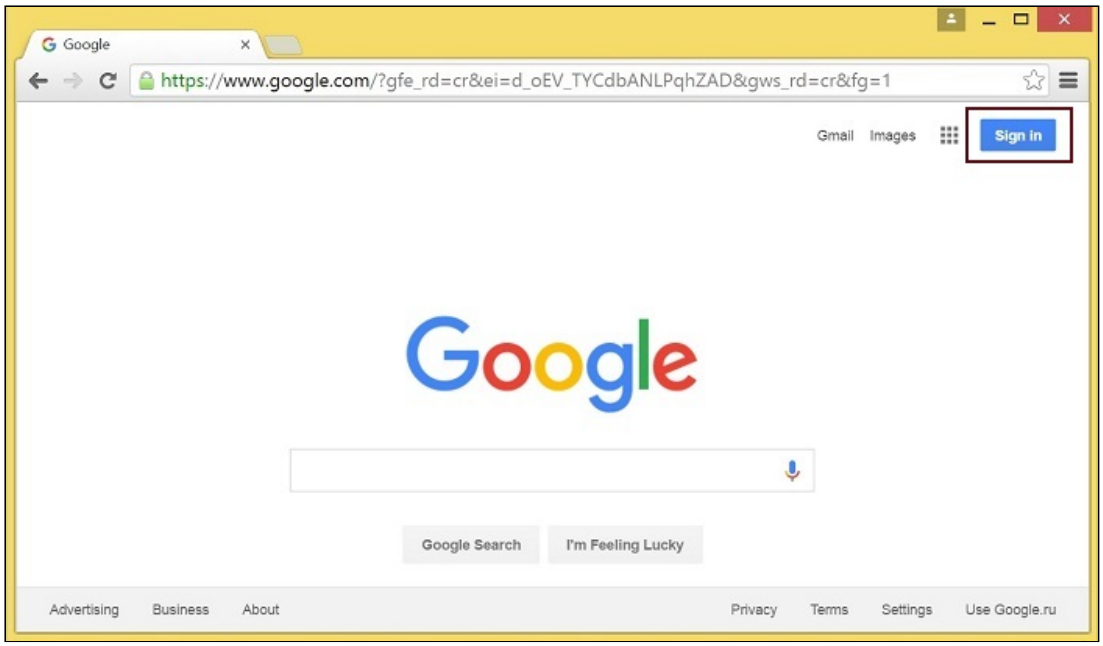
Расширение для Google Chrome с публикацией в Chrome Store

Исходный код расширения для сборки доступен по ссылке ниже:

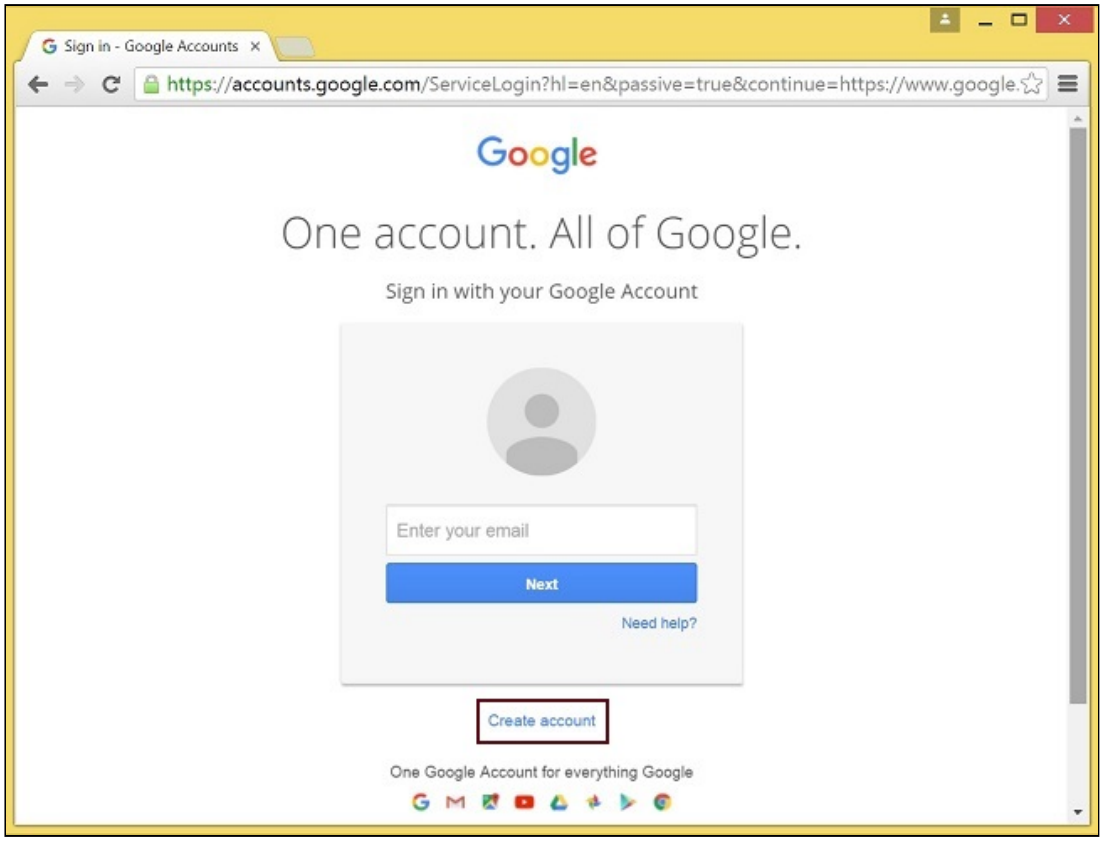
Chrome Screen Sharing Extension

Создайте аккаунт Google

1. Зайдите на google.com и нажмите кнопку `Sign in`



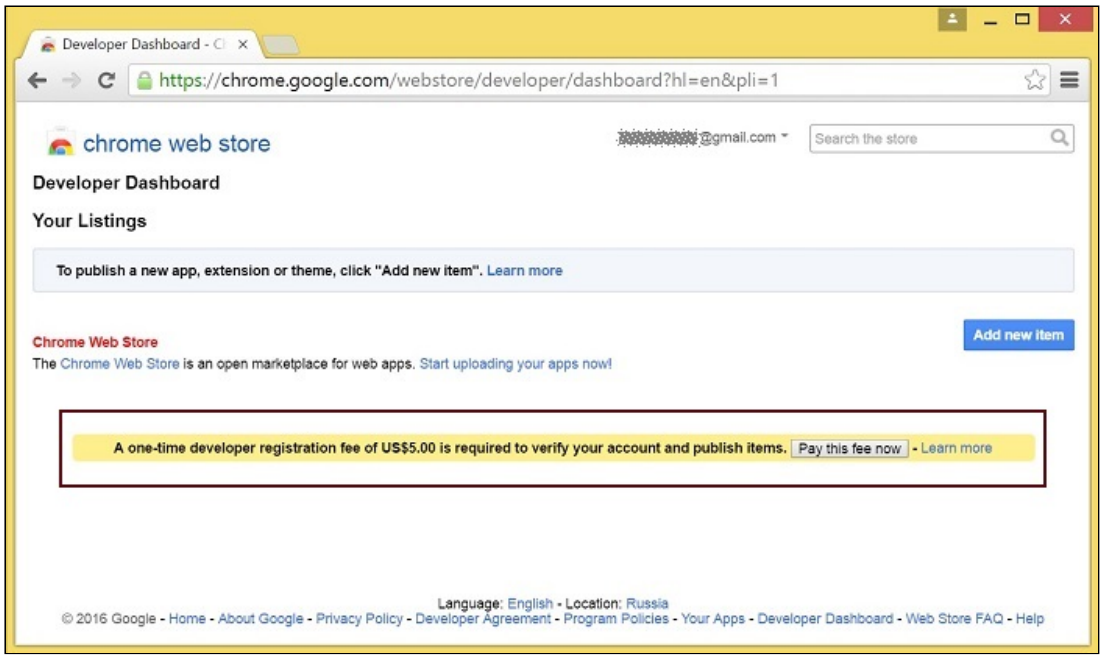
2. Нажмите ссылку `Create account` на странице `Sign in`



3. Будет открыта страница `Create your Google Account`
   Заполните необходимые поля и нажмите кнопку `Next step`, чтобы создать аккаунт.

Зарегистрируйтесь как разработчик Chrome Web Store

1. Войдите в Chrome Developer Dashboard, используя созданный аккаунт Google
2. Внесите единовременную плату в размере US$5, чтобы подтвердить аккаунт



Адаптируйте расширение к вашему домену

Выполните действия, описанные ниже, чтобы использовать расширение со своим доменом. Отредактируйте файл манифеста `manifest.json` расширения для Chrome.

Измените:

- имя (`name`)
- автора (`author`)
- описание (`description`)
- адрес домашней страницы (`homepage_url`)
- в `"externally_connectable":"matches"` замените `flashphoner.com` на свой домен

Добавьте свои иконки в директорию `chrome-extension` и отредактируйте имена файлов в `"icons"` и `"web_accessible_resources"`. (Для дополнительной информации см. Manifest - Icons и Supplying Images.)

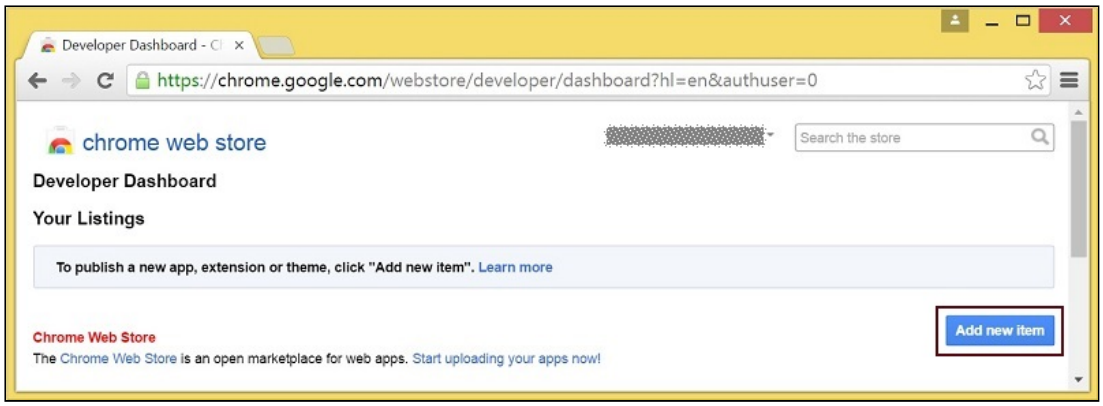
Упакуйте расширение

Упакуйте файлы из директории `chrome-extension` в zip-архив.

Опубликуйте расширение в Chrome Web Store

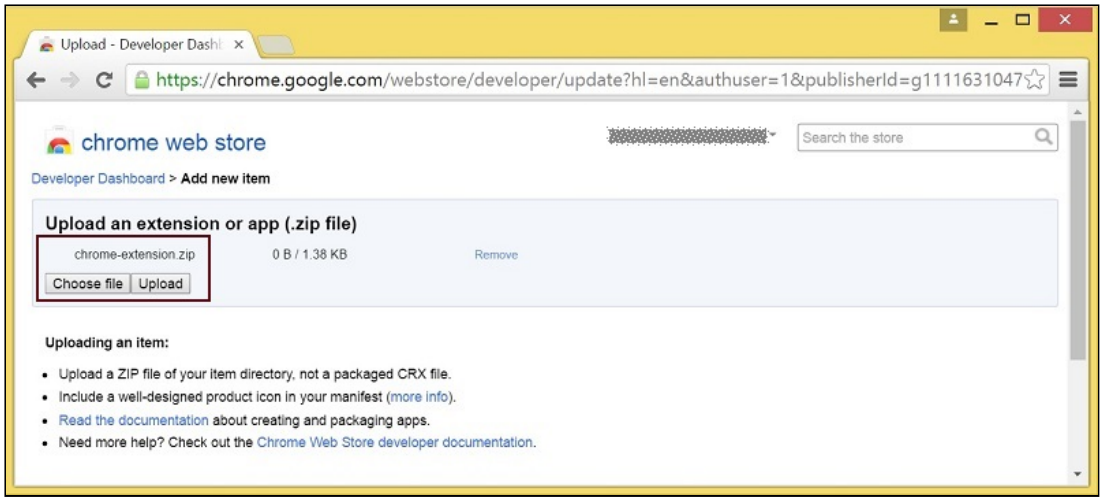
1. Войдите в Chrome Developer Dashboard

2. Нажмите кнопку `Add new item`

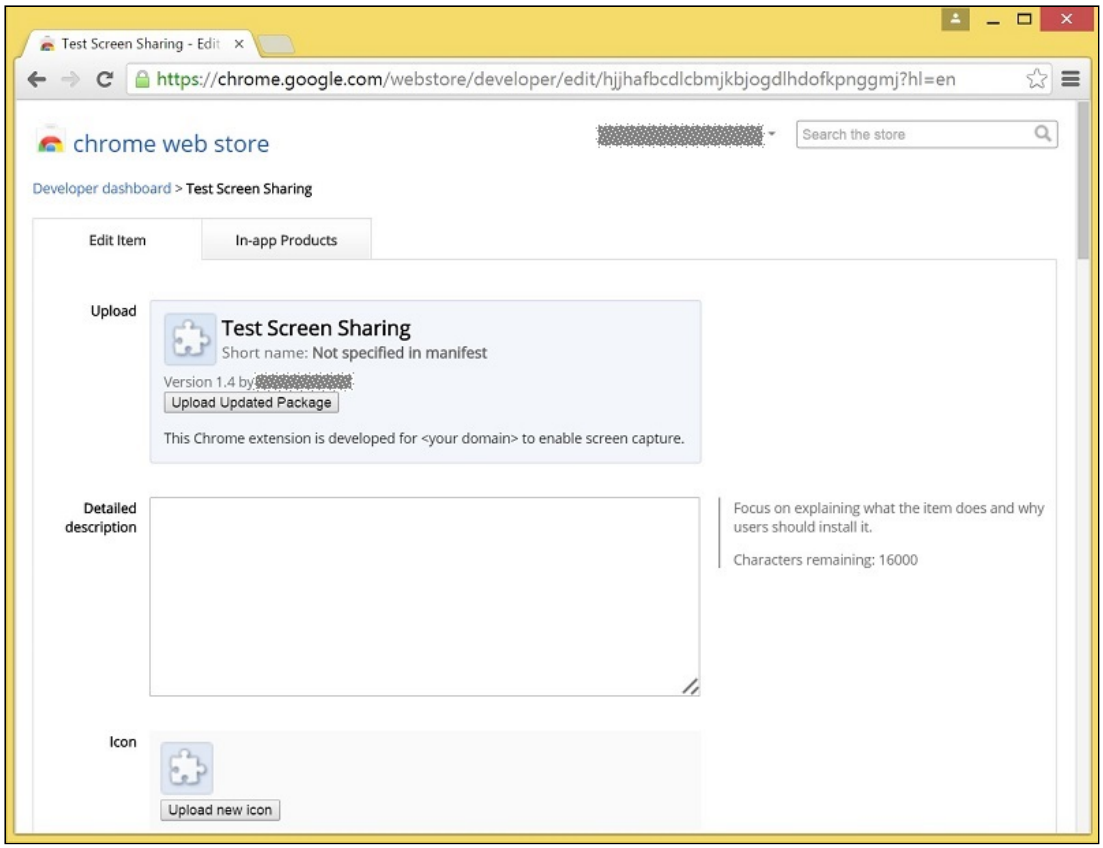


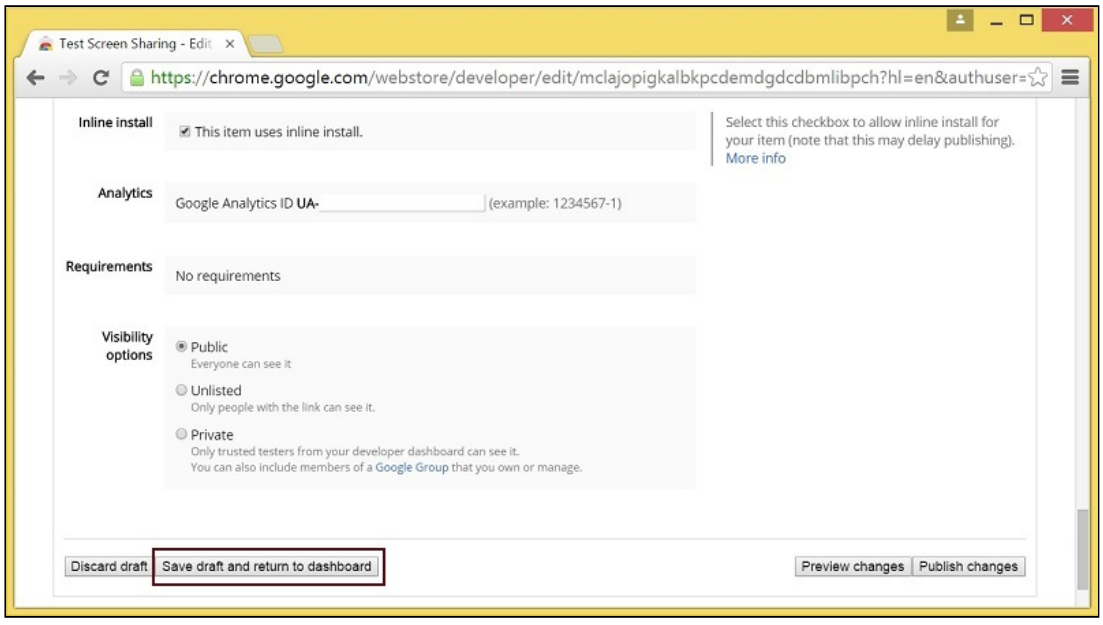
3. Примите соглашение разработчика

4. Выберите файл `chrome-extension.zip` и нажмите кнопку `Upload` на странице загрузки

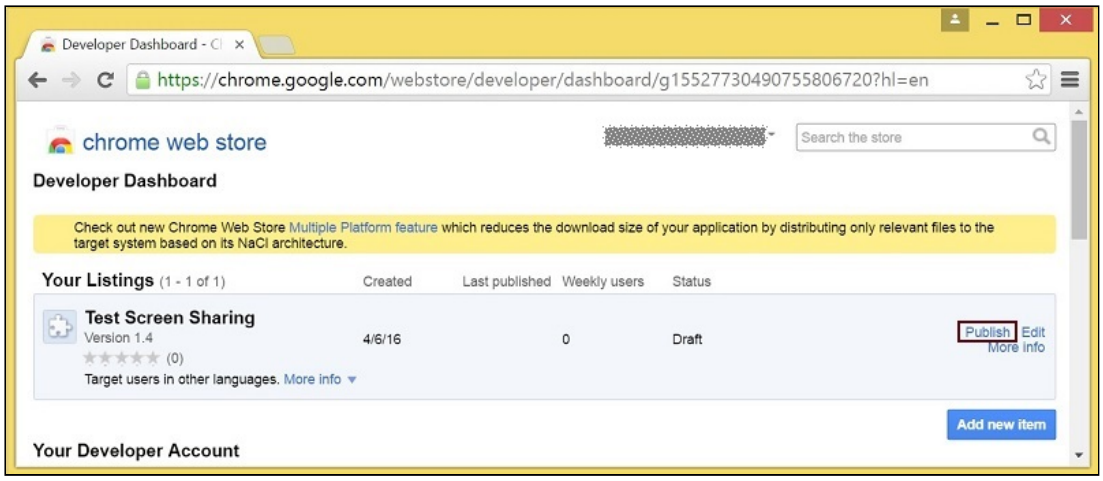


5. После загрузки расширения будет открыта страница для редактирования опций Выберите необходимые опции и нажмите кнопку `Save draft and return to dashboard` внизу страницы

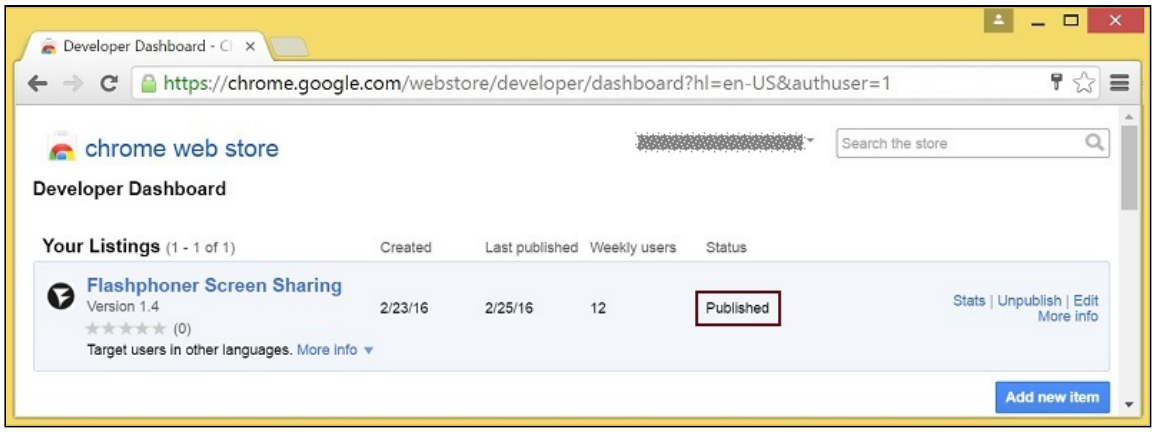




6. Расширение появится в панели разработчика.
   Для публикации расширения нажмите ссылку `Publish`



Опубликованное расширение будет иметь статус `Published`, как на изображении ниже



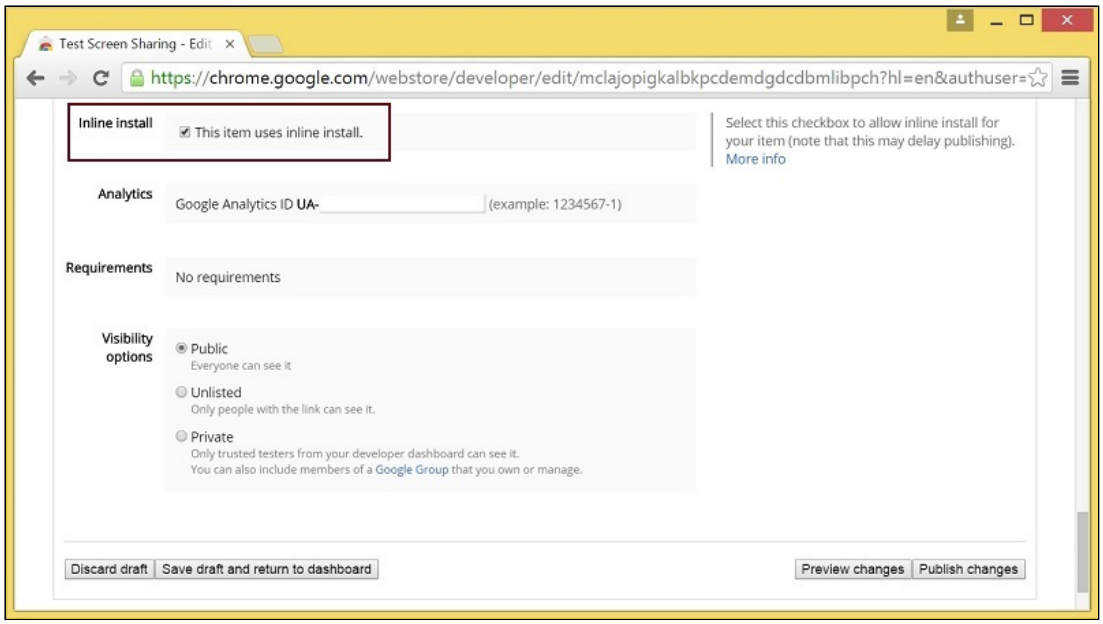
Для дополнительной информации см. Chrome Web Store Publishing Tutorial.

Встроенная установка расширения

Встроенная установка позволяет инициировать установку расширения для демонстрации экрана щелчком по ссылке на странице клиента. Для работы встроенной установки расширение должно быть адаптировано, опубликовано и одобрено.

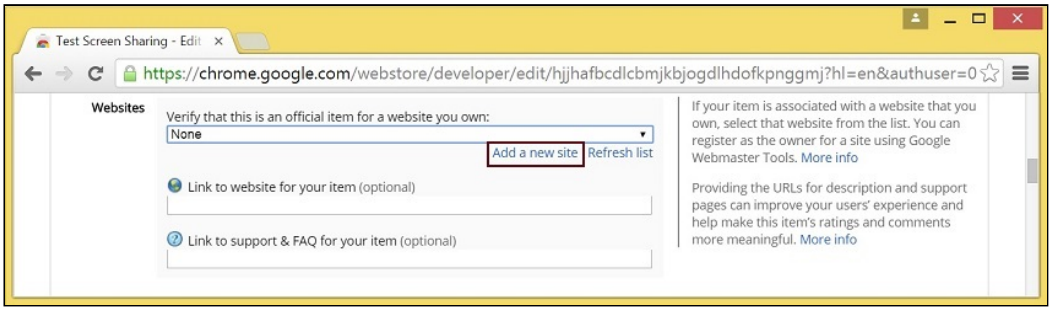
Выполните действия, описанные ниже, чтобы использовать клиент со своими расширениями.

1. При публикации выберите опцию `Inline Install`

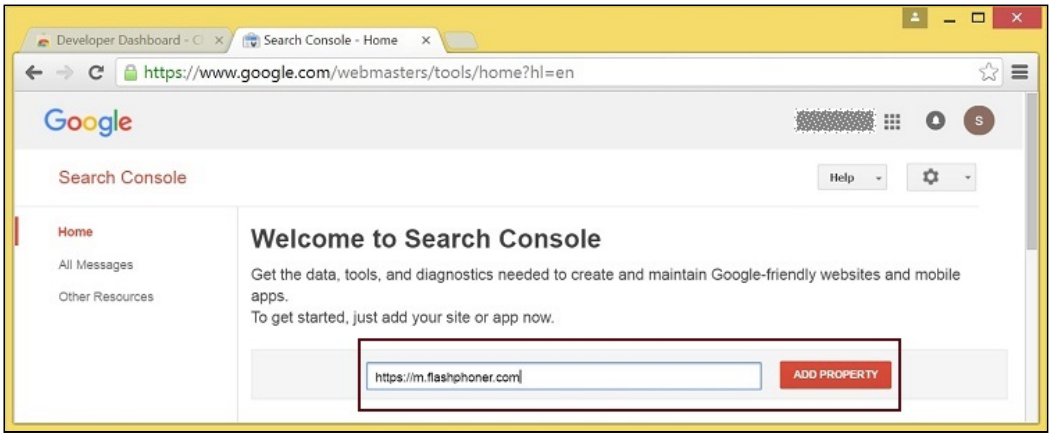


2. Подтвердите и привяжите к расширению сайт со свойм доменом

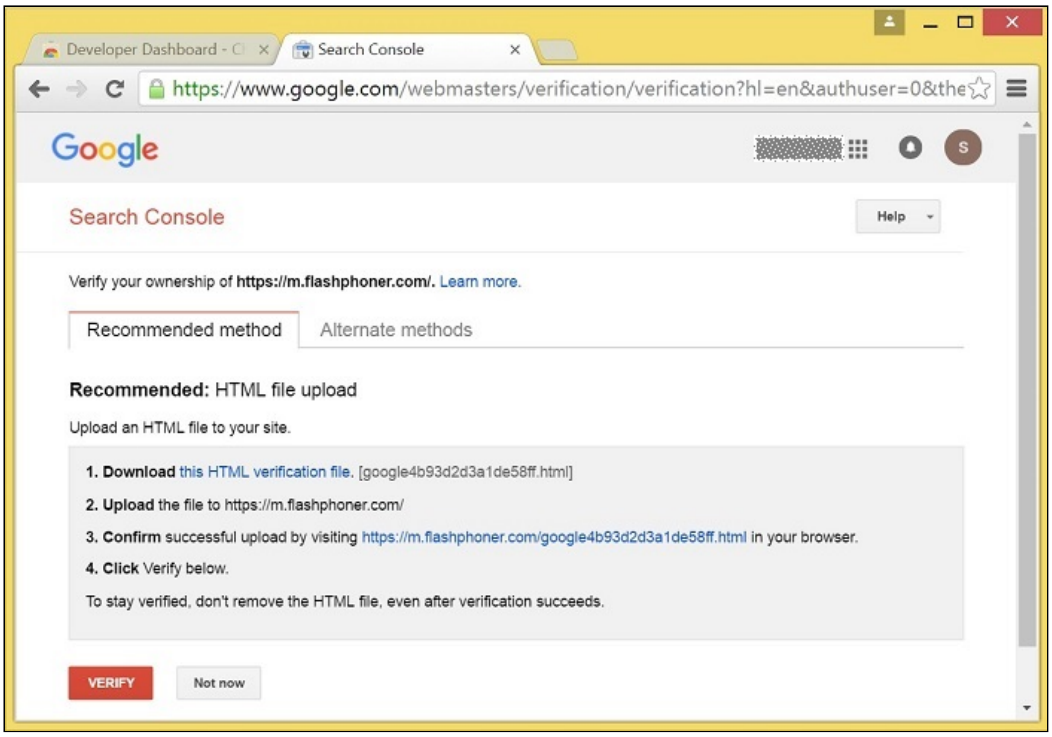
- Нажмите `Add a New Site`



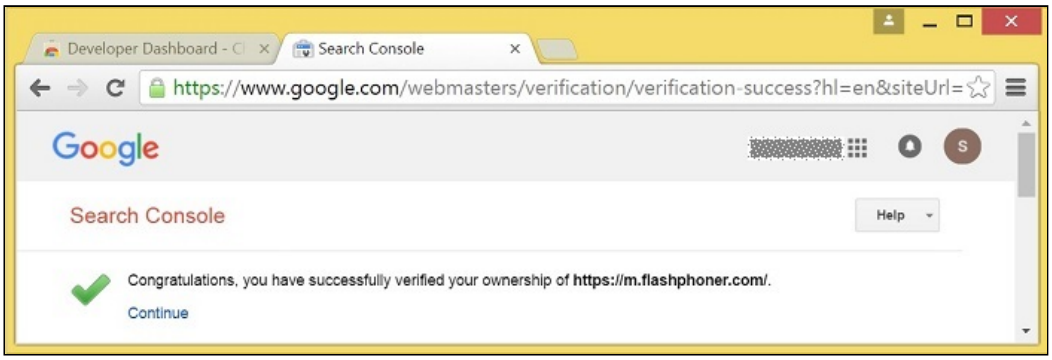
- В новой вкладке браузера будет открыта страница Google Search Console. Введите URL со своим доменом



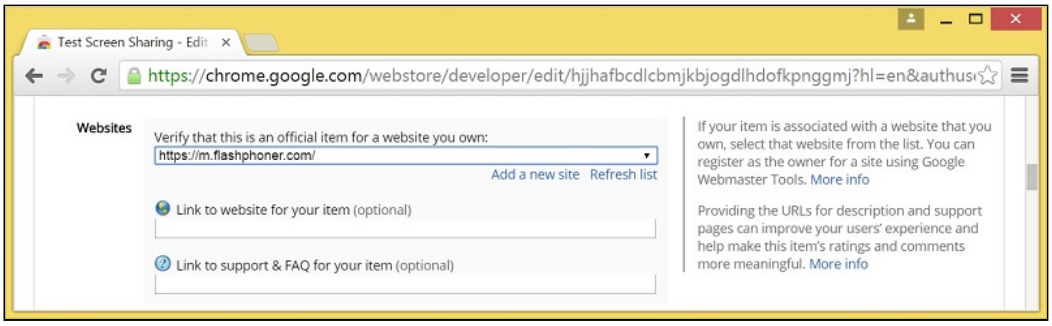
- Будет открыта страница с инструкцией для подтверждения сайта. Выполните требуемые шаги и нажмите кнопку `Verify`



- Если проверка пройдена, будет открыта страница с подтверждением успешной проверки



- Подтвержденный сайт появится в списке в опциях расширения; далее расширение может быть проассоциировано с этим сайтом



Настройка клиента

Отредактируйте `Screen-sharing.html` и `Screen-sharing.js`

- В `Screen-sharing.html` `chrome-webstore-item` должен указывать на ваше расширение в интернет-магазине Chrome
- В `Screen-sharing.js` замените значение переменной `chromeScreenSharingExtensionId` на ID вашего расширения

Чтобы узнать ID расширения, нажмите `More info` для этого расширения в Chrome Developer Dashboard.

Параметры источника медиа

Для настройки параметров источника медиа экрана могут быть использованы параметры объекта `Configuration`, передаваемые методу `init()` при инициализации экземпляра Flashphoner API.

```
var f = Flashphoner.getInstance();
var configuration = new Configuration();
....
configuration.screenSharingVideoWidth = 1920;
configuration.screenSharingVideoHeight = 1080;
configuration.screenSharingVideoFps = 10;
f.init(configuration);
```

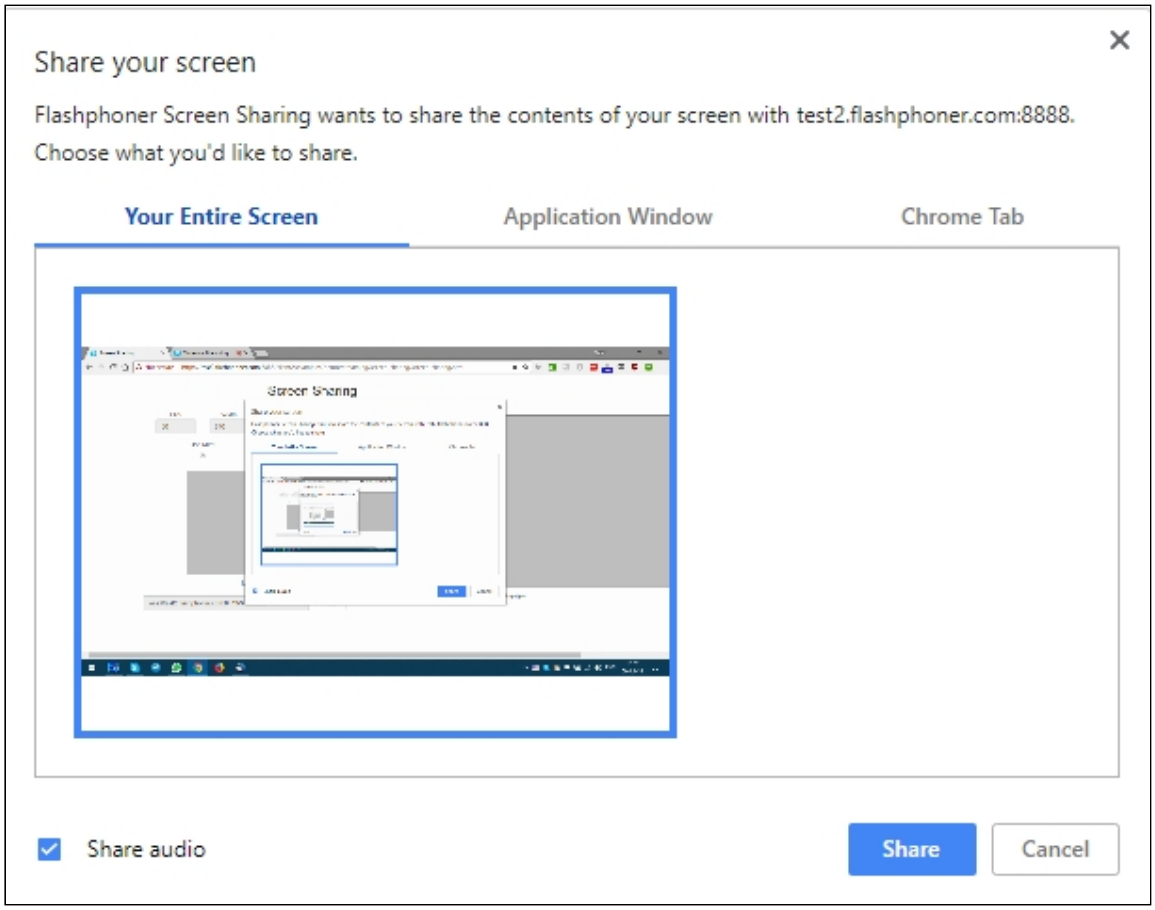
Список параметров

Параметр	Описание
screenSharingVideoWidth	Ширина источника медиа экрана
screenSharingVideoHeight	Высота источника медиа экрана
screenSharingVideoFps	Частота кадров источника медиа эк ран

Данные параметры задают предельные значения разрешения и количество кадров в секунду (FPS). Например, `screenSharingVideoWidth = 1080` означает, что ширина исходного видео не может быть более 1080 пикселей, но может быть меньше (напр., в случае передачи потока с изображением окна приложения с шириной 720 пикселей.)

Захват системного звука в браузере Chrome

В браузере Chrome существует возможность при захвате экрана транслировать аудиопоток из системного источника звука. Такая возможность полезна, например, при скринкастинге. Для захвата системного звука при выборе окна или вкладки в диалоговом окне расширения Chrome установите опцию `Share audio`:



Код расширения code:

```
callback({sourceId: sourceId, systemSoundAccess: opts.canRequestAudioTrack});
```

Управление источником захвата в браузере Firefox

В браузере Firefox можно выбрать экран или окно программы в качестве источника видео при помощи параметра `constraints.video.mediaSource`

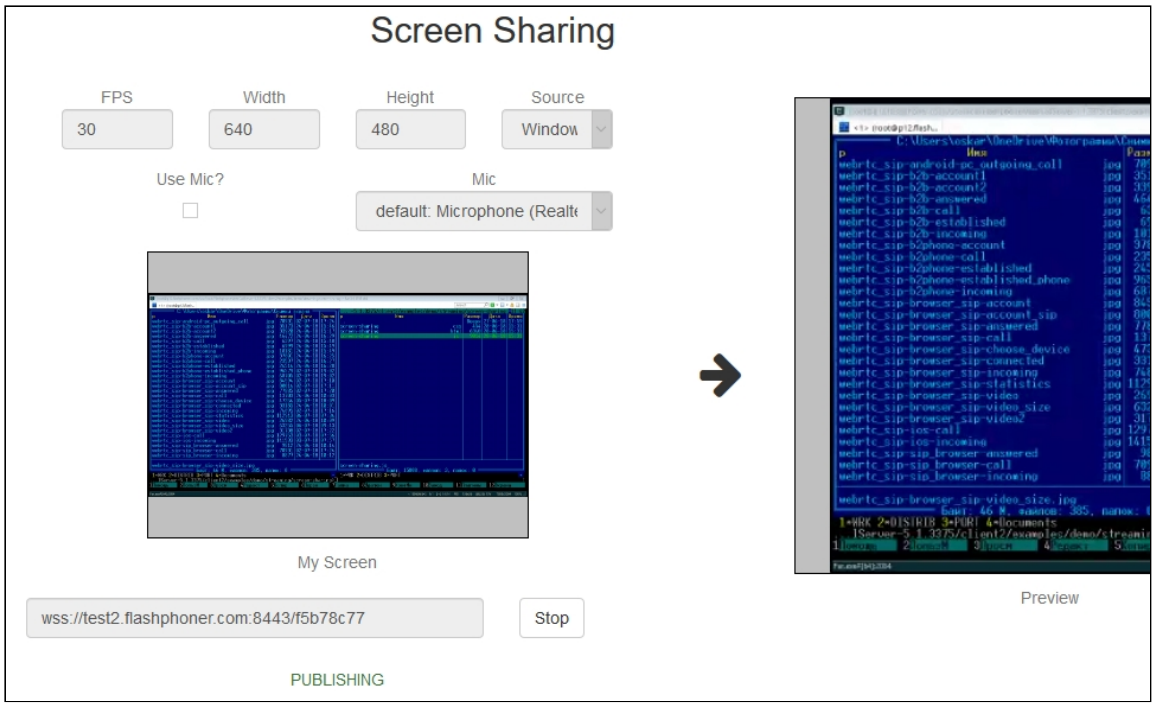
code:

```
constraints.video.type = "screen";
if (Browser.isFirefox()){
    constraints.video.mediaSource = $('#mediaSource').val();
}
session.createStream({
    name: streamName,
    display: localVideo,
```

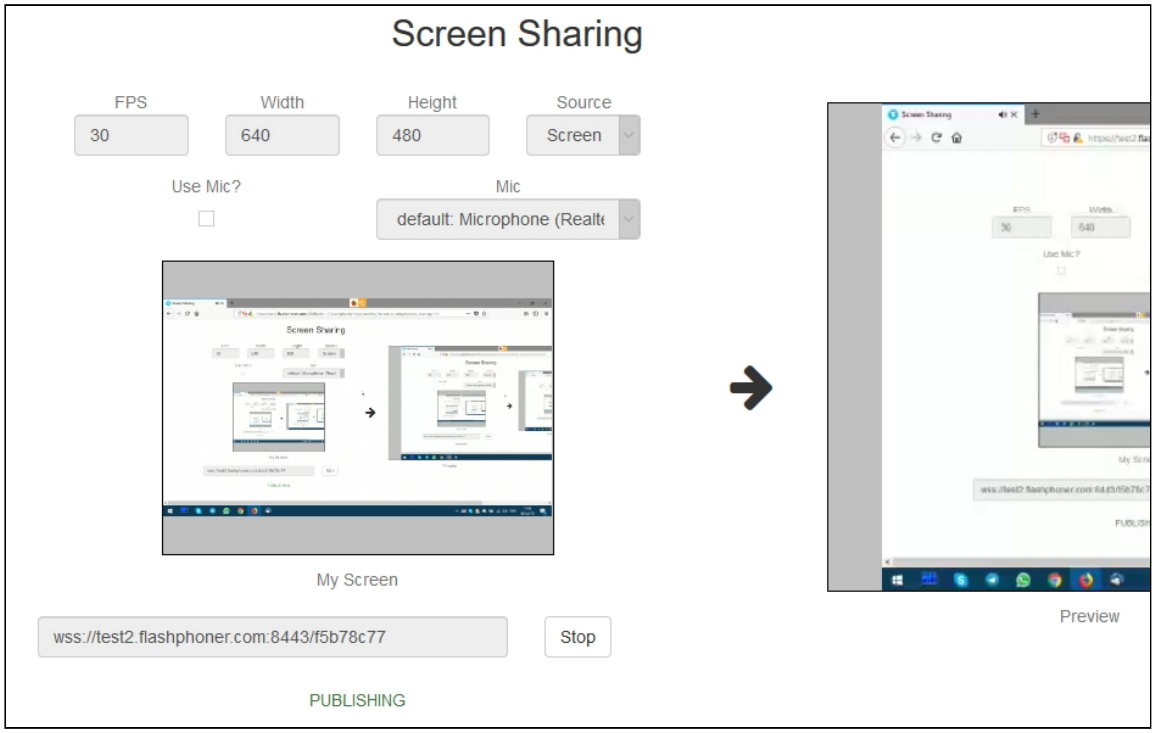
```
    constraints: constraints
})
```

Пример интерфейса для выбора источника

Захват окна программы



Захват экрана



Демонстрация экрана без использования расширения

Браузер Firefox

Браузер Firefox не использует расширение для захвата экрана

Браузеры на основе Chrome

Начиная с версии Chrome 73 и версии Flashphoner WebSDK 0.5.28.2753.86 возможна демонстрация экрана без использования расширения. Для этого необходимо при создании потока передать параметр `constraints.video.withoutExtension`

code

```
if ($("#woChromeExtension").prop('checked')) {
 constraints.video.withoutExtension = true;
}
```

Браузер Safari для MacOS

Начиная с версии Safari 13 и версии Flashphoner WebSDK 0.5.28.2753.152 возможна демонстрация экрана без использования расширения. Для этого необходимо при создании потока передать параметр `constraints.video.withoutExtension`

code

```
if ($("#woChromeExtension").prop('checked') || Browser.isSafari()) {
 constraints.video.withoutExtension = true;
}
```

Известные ограничения

1. При публикации из Chrome, разрешение и FPS устанавливаются не в соответствии с заданными в `constraints` при создании потока, а по размерам источника (экрана, окна или вкладки браузера) и по реальной частоте изменения картинки. Эта проблема исправлена, начиная со сборки Web SDK 0.5.28.2753.152
2. Захват системного звука работает начиная с версии Chrome 74

Код примера

Пример использует расширение Flashphoner для работы с доменом *.flashphoner.com. Для того, чтобы пример работал с Вашим доменом, необходимо собрать и выложить Ваше расширение, как описано выше.

Как временное решение, вы можете прописать IP-адрес вашего WCS-сервера в файле `C:\Windows\System32\drivers\etc\hosts` (на Windows системе) как test.flashphoner.com. Таким образом, вы сможете тестировать ваш WCS-сервер с доменом test.flashphoner.com, пока не соберете расширение под Ваш собственный домен.

Пример работает в Chrome только через HTTPS.

Код данного примера находится на сервере по следующему пути:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/screen-sharing

- screen-sharing.css - файл стилей
- screen-sharing.html - веб-интерфейс
- screen-sharing.js - скрипт, обеспечивающий работу интерфейса

Тестировать пример можно по следующему адресу:

https://host:8888/client2/examples/demo/streaming/screen-sharing/screen-sharing.html

Здесь host - адрес вашего WCS-сервера.

Для Chrome ссылка на расширение указывается прямо в файле `screen-sharing.html` line 17

```
<link rel="chrome-webstore-item"
href="https://chrome.google.com/webstore/detail/nlbaajplpmleofphigmgaifhoikjmbkg
```

Работа с кодом примера

1. Инициализация API с указанием идентификатора расширения для Chrome браузера

`Flashphoner.init()` code

```
Flashphoner.init({screenSharingExtensionId: extensionId});
```

2. Подключение к WCS-серверу

`Flashphoner.createSession()` code

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function(session){
  //session connected, start streaming
  startStreaming(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
  setStatus(SESSION_STATUS.DISCONNECTED);
  onStopped();
}).on(SESSION_STATUS.FAILED, function(){
  setStatus(SESSION_STATUS.FAILED);
  onStopped();
});
```

3. Получение от сервера события, подтверждающего успешное соединение

`ConnectionStatusEvent ESTABLISHED` code

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function(session){
  //session connected, start streaming
  startStreaming(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
  ...
}).on(SESSION_STATUS.FAILED, function(){
  ...
});
```

4. Настройка параметров публикации

разрешение и fps code

```
var constraints = {
  video: {
      width: parseInt($('#width').val()),
      height: parseInt($('#height').val()),
      //WCS-2014. fixed window/tab sharing
      frameRate: parseInt($('#fps').val())
  }
};
```

использование микрофона code

```
if ($("#useMic").prop('checked')) {
  constraints.audio = {
      deviceId: $('#audioInput').val()
  };
}
```

тип источника видео и возможность публикации без расширения code

```
constraints.video.type = "screen";
if ($("#woChromeExtension").prop('checked')) {
  constraints.video.withoutExtension = true;
}
```

источник медиа в браузере Firefox code

```
if (Browser.isFirefox()){
  constraints.video.mediaSource = $('#mediaSource').val();
}
```

5. Публикация видеопотока

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

```
session.createStream({
  name: streamName,
  display: localVideo,
  constraints: constraints
  ...
}).publish();
```

6. Получение от сервера события, подтверждающего успешную публикацию потока

`StreamStatusEvent PUBLISHING` code

При получении данного события запускается проигрывание превью в отдельном элементе на странице при помощи `Session.createStream()` и `Stream.play()`

```
session.createStream({
  name: streamName,
  display: localVideo,
  constraints: constraints
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
  /*
   * User can stop sharing screen capture using Chrome "stop" button.
   * Catch onended video track event and stop publishing.
   */
  document.getElementById(publishStream.id()).srcObject.getVideoTracks()
[0].onended = function (e) {
      publishStream.stop();
  };
  document.getElementById(publishStream.id()).addEventListener('resize',
function(event){
      resizeVideo(event.target);
  });
  setStatus(STREAM_STATUS.PUBLISHING);
  //play preview
  session.createStream({
      name: streamName,
      display: remoteVideo
      ...
  }).play();
}).on(STREAM_STATUS.UNPUBLISHED, function(){
  ...
}).on(STREAM_STATUS.FAILED, function(){
  ...
}).publish();
```

7. Получение от сервера события, подтверждающего успешное воспроизведение превью-потока

`StreamStatusEvent PLAYING` code

```
session.createStream({
   name: streamName,
   display: remoteVideo
}).on(STREAM_STATUS.PLAYING, function(previewStream){
   document.getElementById(previewStream.id()).addEventListener('resize',
function(event){
       resizeVideo(event.target);
   });
   //enable stop button
   onStarted(publishStream, previewStream);
}).on(STREAM_STATUS.STOPPED, function(){
   ...
}).on(STREAM_STATUS.FAILED, function(){
   ...
}).play();
```

8. Остановка воспроизведения видеопотока

`Stream.stop()` code

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

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

`StreamStatusEvent_STOPPED` code

При получении этого события останавливается публикация потока при помощи `Stream.stop()`

```
session.createStream({
   name: streamName,
   display: remoteVideo
}).on(STREAM_STATUS.PLAYING, function(previewStream){
   ...
}).on(STREAM_STATUS.STOPPED, function(){
   publishStream.stop();
}).on(STREAM_STATUS.FAILED, function(){
   ...
}).play();
```

10. Остановка публикации видеопотока по кнопке расширения Chrome

`publishStream.stop()` code

```
document.getElementById(publishStream.id()).srcObject.getVideoTracks()
[0].onended = function (e) {
   publishStream.stop();
};
```

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

`StreamStatusEvent.UNPUBLISHED` code

```
session.createStream({
   name: streamName,
   display: localVideo,
   constraints: constraints
}).on(STREAM_STATUS.PUBLISHING, function(publishStream){
   ...
}).on(STREAM_STATUS.UNPUBLISHED, function(){
   setStatus(STREAM_STATUS.UNPUBLISHED);
   //enable start button
   onStopped();
}).on(STREAM_STATUS.FAILED, function(){
   ...
}).publish();
```