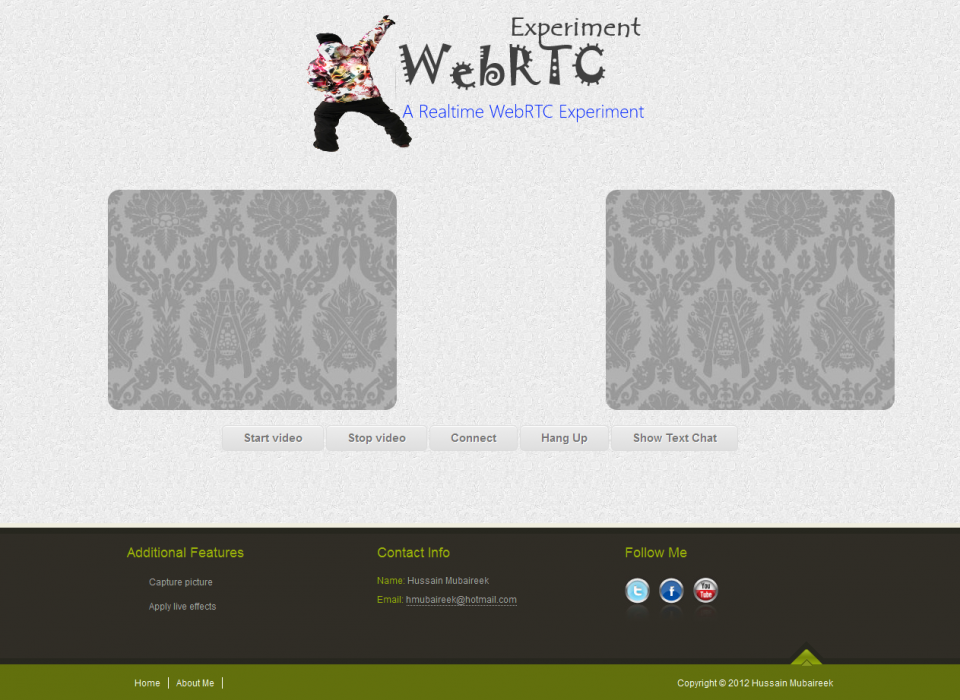
Video Conferencing using WebRTC

**[Hussain Mubaireek](https://www.codeproject.com/script/Membership/View.aspx?mid=9627297)**, 11 Jan 2013

|  |  |
| --- | --- |
| https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-fill-md.png  https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-empty-md.png | 4.81 (22 votes) |
|  | |  |  |  | | --- | --- | --- | | Rate this: | [vote 1](https://www.codeproject.com/Articles/515192/Video-Conferencing-using-WebRTC" \o "vote 1)[vote 2](https://www.codeproject.com/Articles/515192/Video-Conferencing-using-WebRTC" \o "vote 2)[vote 3](https://www.codeproject.com/Articles/515192/Video-Conferencing-using-WebRTC" \o "vote 3)[vote 4](https://www.codeproject.com/Articles/515192/Video-Conferencing-using-WebRTC" \o "vote 4)[vote 5](https://www.codeproject.com/Articles/515192/Video-Conferencing-using-WebRTC" \o "vote 5) |  | |

Video Conference in HTML5 using WebRTC with websockets and javascript only.



Introduction

This project is implementing a simple video conference application, where you can do an online video conference between two persons opening the same html page. This is done via WebRTc and Websockets.

Background

Video Conferencing is hard and needs many things like Wideband codecs (voice and video), Echo cancellation, Automatic Gain Control, Noise reduction/suppression, Dynamic jitter buffers, Error concealment, Network Traversal, P2P protocols, and Session setup.

However, now it became much easier with WebRTC as you can do video conferencing in browser without plugins!

 First let's know **what is WebRTC?**

WebRTC is a free, open project that enables web browsers with Real-Time Communications (RTC) capabilities via simple Javascript APIs.

The mission of WebRTC is to enable rich, high quality, RTC applications to be developed in the browser via simple Javascript APIs and HTML5.

The work is still going on WebRTC and it is supported by Google Chrome Chanary/Firefox Aurora/Opera Labs.

What you need to do it?

To do video conferencing we need to do five things:

1. Set up video element in an html page:

Hide   Copy Code

<video></video>

2. Access local devices (camera,microphone):

Hide   Copy Code

navigator.getUserMedia();

3.  Display a/v from local or remote peer:

Hide   Copy Code

createObjectURL

4. Connect to remote peers

Hide   Copy Code

PeerConnection

Using the code

**First, to access local device and display it in your html page we use this code:**

Hide   Copy Code

*// Replace the source of the video element with the stream from the camera*

try

{

navigator.webkitGetUserMedia({audio: true, video: true}, successCallback, errorCallback);

}

catch (e)

{

navigator.webkitGetUserMedia("video,audio", successCallback, errorCallback);

}

function successCallback(stream) {

sourcevid.src = window.webkitURL.createObjectURL(stream);

sourcevid.style.webkitTransform = "rotateY(180deg)";

localStream = stream;

}

function errorCallback(error) {

logg('An error occurred: [CODE ' + error.code + ']');

}

*Browsers implementations for GetUserMedia():*

WebKit (Chrome):  
webkitGetUserMedia()  
window.webkitURL.createObjectURL()

Opera:  
getUserMedia()  
set video.src directly

Firefox:  
mozGetUserMedia()  
window.URL.createObjectURL()  
  
IE:  
Not implemented until now.

**Second, we need to set up a peer connection:**

We can establish the connection via Websockets, or use a channel management server on Google’s appengine (Like the open source apprtc application). I decided to use Websocket server to be working as two-way communication channel.

WebRTC uses PeerConnection to communicate streams of data. However, we need a mechanism to send control messages between peers, a process known as signalling. You can choose whatever messaging protocol you prefer, such as SIP.

* Signaling Procedure:  
  - Caller sends **offer**.  
  - Callee receives **offer**.  
  - Callee sends **answer**.  
  - Caller receives **answer**.

To start a session WebRTC client needs local & remote configuration information so those configuration information are sent by SessionDiscreption which conforms to SDP. The SessionDescription sent by the caller   
is known as an offer, and the response from the callee is an answer. This architecture is called JSEP, JavaScript Session Establishment Protocol.  
  
*Enabling chat messaging via WebRTC*  
 I was interested to get text chat working with WebRTC as I knew that it is not an available feature since you can stream video/audio only. What I did to get it working was modifying the signaling procedure a little bit. So, when the Caller wants to send a text message I make him send a **chat**message in SessionDescription message and once the reciever recieves it he add it to the chatting iframe he have.

**Third, we need to add the remote stream after creating the peer connection:**

Hide   Copy Code

var remotevid = document.getElementById('remotevid');

try {

logg("Creating peer connection");

var servers = [];

servers.push({'url':'stun:' + stunServer});

var pc\_config = {'iceServers':servers};

peerConn = new webkitRTCPeerConnection(pc\_config);

peerConn.onicecandidate = onIceCandidate;

}

catch (e) {

logg("Failed to create PeerConnection, exception: " + e.message);

}

peerConn.onaddstream = onRemoteStreamAdded;

peerConn.onremovestream = onRemoteStreamRemoved;

function onRemoteStreamAdded(event) {

logg("Added remote stream");

remotevid.src = window.webkitURL.createObjectURL(event.stream);

}

Points of Interest

This project was interesting as I learned something new and I also was even interested in implementing and trying something like capturing pictures from the camera stream.

So, to take a picture from a camera stream do create canvas element:

Hide   Copy Code

<canvas id="canvas"></canvas>

And also we need a buton to take a picture and an img tag to set the captured picture to it:

Hide   Copy Code

<button id="btn">Take photo</button>

<img src="" id="photo" alt="photo" style="margin-left:90px;">

And ofcourse you have to have video element as we learned above. Then:

Hide   Copy Code

startbutton = document.querySelector('#btn');

startbutton.addEventListener('click', function(ev){

takepicture();

ev.preventDefault();

}, false);

function takepicture() {

canvas.width = width;

canvas.height = height;

canvas.getContext('2d').drawImage(video, 0, 0, width, height);

var data = canvas.toDataURL('image/png');

photo.setAttribute('src', data);

}

Also, something more interesting was to apply live effects to the camera stream using CSS.

How to run the code

1. Download the source code.  
2. Move the source files to your www directory (You need to have an http server running locally).  
3. Install node.js (From ***nodejs****.org*) to be able to run the server using it.  
4. After installing node.js (And the required commands with it) open Node.js Command Prompt.  
5. Move to the directory were the websocket server is located (websocket-server.js).  
6. Write this command: node websocket-server.js.   
7. Now the server is running and listening on port 1337.  
8. Open VideoConference.htm and modify the ip address in the code to match your computer's network ip (or you can write 127.0.0.1 ).  
9. Now, the other client should do steps 1-2 & 8 (In step 8 he should put the ip address of the computer who has the server running).  
10. Now, after opening VideoConference.htm (In Chrom Canary browser), the server should detect the connection from both of you.  
11. After both clicking on start and now you can see youself. Click on Connect to see the remote client also.

Acknowledgment

This work was part of a project for SWE 344 Internet Protocols and Client-Server Programming (Fall 2012).   
The project was suggested by my instructor,  Dr. [Nasir Darwish.](http://www.codeproject.com/Members/Darwish)

References

1. GingerTech:  
http://blog.gingertech.net/2012/06/04/video-conferencing-in-html5-webrtc-via-web-sockets/  
  
He implemented the websocket server and I adapted it to my application (The demo there is not working because it is an old one and uses the old depricated APIs.

2. apprtc applicatino done by Chrom developers:

<http://code.google.com/p/webrtc-samples/source/browse/trunk/apprtc/>

To understand the process and see the new APIs.  
  
3. WebRTC.org:  
[www.webrtc.org/home](http://www.webrtc.org/home)   
  
For general information and demos:  
  
4. HTML5Rocks:  
<http://www.html5rocks.com/en/tutorials/webrtc/basics/>   
  
To get general information and understand the process.

 5. Photobooth example:  
<http://html5-demos.appspot.com/static/getusermedia/photobooth.html>   
  
For the CSS filters to apply on live stream.

Source Code

<https://www.dropbox.com/sh/4dj54jhnfqv4hx6/c0UUxS75Dp/VideoConferenceWebRTC.zip>

Mirror:  [www.mediafire.com/?noc33ur1eo4x37k](https://www.codeproject.com/Articles/515192/www.mediafire.com/?noc33ur1eo4x37k%20)

History

v1.0 : Initial release - 24th of December 2012.

License

This article, along with any associated source code and files, is licensed under [A Public Domain dedication](http://creativecommons.org/licenses/publicdomain/)

Share