FREE eBook

LEARNING webrtc

Free unaffiliated eBook created from
Stack Overflow contributors.

#webrtc

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: webrtc

It is an unofficial and free webrtc ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official webrtc.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with webrtc

Remarks

WebRTC is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs. The WebRTC components have been optimized to best serve this purpose.

Checkout following links to get more information about WebRTC
webrtc.org
webrtc architecture
check live demo here

Examples

Setting up a WebRTC-based communication system

To setup a WebRTC-based communication system, you need three main components:

1. **A WebRTC signaling server**

   To establish a WebRTC connections, peers need to contact a signaling server, which then provides the address information the peers require to set up a peer-to-peer connection. Signaling servers are for example:

   - signalmaster: Lightweight, JavaScript-based signaling server
   - NextRTC: Java-based signaling server
   - Kurento: Comprehensive WebRTC framework
   - Janus: General purpose WebRTC Gateway

2. **A WebRTC client application**

   The client accesses either a browser's WebRTC implementation through a JavaScript API or uses a WebRTC library (i.e. as part of a desktop or mobile app). To establish the connection to a peer, the client first needs to connect to the signaling server. Examples for WebRTC clients are:

   - Several Kurento projects
   - OpenWebRTC, a cross-platform client with mobile focus
   - Peer.js A browser-based client (Peer.js also provides a light-weight server)
   - Janus Demo examples

3. **A STUN/TURN server**

Session Traversal Utilities for NAT (STUN) enables peers to exchange address information even if they are behind routers employing Network Adress Translation (NAT). If network restrictions prevent peers from communication directly at all, the traffic is routed via a Traversal Using Relays around NAT (TURN) server. You find a detailed and graphical explanation of STUN and TURN at http://www.avaya.com/blogs/archives/2014/08/understanding-webrtc-media-connections-ice-stun-and-turn.html. Examples for WebRTC STUN/TURN servers are:

- coturn combines STUN and TURN and is typically part of a fully-fledged WebRTC infrastructure.
- Janus WebRTC Gateway comes with an integrated STUN/TURN server.

Introduction to WebRTC

WebRTC is an open framework for the web that enables Real Time Communications in the browser. It includes the fundamental building blocks for high-quality communications on the web, such as network, audio and video components used in voice and video chat applications.

These components, when implemented in a browser, can be accessed through a JavaScript API, enabling developers to easily implement their own RTC web app.

The WebRTC effort is being standardized on an API level at the W3C and at the protocol level at the IETF.

- A key factor in the success of the web is that its core technologies – such as HTML, HTTP, and TCP/IP – are open and freely implementable. Currently, there is no free, high-quality, complete solution available that enables communication in the browser. WebRTC enables this.
- Already integrated with best-of-breed voice and video engines that have been deployed on millions of endpoints over the last 8+ years. Google does not charge royalties for WebRTC.
- Includes and abstracts key NAT and firewall traversal technology, using STUN, ICE, TURN, RTP-over-TCP and support for proxies.
- Builds on the strength of the web browser: WebRTC abstracts signaling by offering a signaling state machine that maps directly to PeerConnection. Web developers can therefore choose the protocol of choice for their usage scenario (for example, but not limited to, SIP, XMPP/Jingle, etc).

Read more about WebRTC from here

Get access to your audio and video using getUserMedia() API, Hello WebRTC!

`navigator.mediaDevices` is the common method adapted in Chrome and FF to getUserMedia as of now.

A promised based call back which returns local stream on success

```
navigator.mediaDevices.getUserMedia({ audio: true, video: true })
    .then(stream => {
```

```
        // attach this stream to window object so you can reuse it later
        window.localStream = stream;
        // Your code to use the stream
    })
    .catch((err) =>{
        console.log(err);
    });
```

You can pass audio and video constraints to getUserMedia to control capture settings like resolution, framerate, device preference, and more.

Attach the stream to a video element

```
// Set the video element to autoplay to ensure the video appears live
videoElement.autoplay = true;
// Mute the local video so the audio output doesn't feedback into the input
videoElement.muted = true;
// attach the stream to the video element
```

stop both video and audio

```
localStream.getTracks().forEach((track) => {
    track.stop();
});
```

stop only audio

```
localStream.getAudioTracks()[0].stop();
```

stop only video

```
localStream.getVideoTracks()[0].stop();
```

Live demo

Read Getting started with webrtc online: https://riptutorial.com/webrtc/topic/4623/getting-started-with-webrtc

Chapter 2: Using getUserMedia() to request camera and microphone access

Examples

Using getUserMedia()

As we knew, WebRTC is all based on the JavaScript development and coding and for more information and examples please refer here and here.

And now, let me show you a very simple example to use `getUserMedia();`

For what `getUserMedia()` is used?

getUserMedia() is used to get the user/visitor's camera and microphone detection.

Supported browsers for `getUserMedia()`

- Mozilla Firefox 22 (PC) or higher.
- Microsoft Edge 21 (PC) or higher.
- Google Chrome 23 (PC) or higher.
- Opera 18 (PC) or higher.
- Google Chrome 28 (Android) or higher.
- Mozilla Firefox 24 (Android) or higher.
- Opera Mobile 12 (Android) or higher.
- iOS (**Bowser**).
- Chrome OS
- Firefox OS
- Default BlackBerry 10 browser.

Required files to use `getUserMedia`

- https://github.com/webrtc/samples/blob/gh-pages/src/content/getusermedia/gum/js/test.js

- https://github.com/webrtc/samples/blob/gh-pages/src/content/getusermedia/gum/js/main.js
- https://github.com/webrtc/samples/blob/gh-pages/src/js/adapter.js
- https://github.com/webrtc/samples/blob/gh-pages/src/js/common.js
- https://github.com/webrtc/samples/blob/gh-pages/src/js/lib/ga.js

---

1. Start coding in a normal HTML file.
2. In the `<body></body>` tags, include the required WebRTC API files:

I'm not able to use the Stackoverflow Editor's Code, so here is the code:
http://pastebin.com/2fQzJhuG

Good job! you're just a great starter now and it should work normally.

Last code:

```
<html>
<body>
<script src="js/adapter.js"></script>
<script src="js/common.js"></script>
<script src="js/main.js"></script>
<script src="js/lib/ga.js"></script>
<body>
</html>
```

It was very easy, right?

Read Using getUserMedia() to request camera and microphone access online:
https://riptutorial.com/webrtc/topic/6134/using-getusermedia---to-request-camera-and-microphone-access

Chapter 3: WebRTC simple examples

Parameters

getUserMedia() Paramters	Description
Constraints	It consist of array which allows us to specify which media devices to use i.e audio or video or both
Success callback	Create a function for success callback which will provide you the stream which you get from your media devices
Error callback	This callback get invoked when there is problem like there are no media devices, or user has denied the permission to use them

Examples

Get camera and microphone permission and display preview on webpage

In order to begin using WebRTC you need to get camera and microphone permission.For that you need following things:

1. `adapter.js`, you can get it from here
2. A html webpage with a video tag and little bit of js code

The adapter.js is a JavaScript shim for WebRTC, maintained by Google with help from the WebRTC community, that abstracts vendor prefixes, browser differences and spec changes.

Now once you have this file, create a html file with following code:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My first webrtc example</title>
        <script src="adapter.js"></script>
        <script type="text/javascript">
            function gotStream(stream) {
                window.AudioContext = window.AudioContext || window.webkitAudioContext;
                var audioContext = new AudioContext();

                // Create an AudioNode from the stream
                var mediaStreamSource = audioContext.createMediaStreamSource(stream);

                // Connect it to destination to hear yourself
                // or any other node for processing!
                mediaStreamSource.connect(audioContext.destination);
                var video = document.querySelector('video');
                var videoTracks = stream.getVideoTracks();
```

```
                window.stream = stream; // make variable available to browser console
                video.srcObject = stream;
            }
            function onfail(error) {
                console.log("permission not granted or system don't have media
devices."+error.name);
            }
            navigator.getUserMedia({audio:true,video:true}, gotStream,onfail);


        </script>
    </head>
    <body>
        Welcome to webrtc
        <video id="local" autoplay=""></video>
    </body>
</html>
```

Once done, save this file and run in the browser. When you run the browser will ask you to allow this webpage to access your webcam and microphone. Allow it and whola!, you will see the preview on the webpage.

Read WebRTC simple examples online: https://riptutorial.com/webrtc/topic/5641/webrtc-simple-examples

Credits

S. No	Chapters	Contributors
1	Getting started with webrtc	AJ., Community, Griffin, Ichigo Kurosaki, mpromonet, Olga Khylkouskaya, Sasi Varunan, Timotheus.Kampik, xdumaine
2	Using getUserMedia() to request camera and microphone access	protld
3	WebRTC simple examples	Ichigo Kurosaki