

Web RTC introduced by World Wide Web Consortium W3C. That supports browser-to-browser applications for voice calling, video chat, and P2P file sharing.

If you want to try out? web RTC available for Chrome, opera, and firefox. A good place to start is the simple video chat application at [here](#). Web RTC implements three API's as shown below –

- **MediaStream** – get access to the user's camera and microphone.
- **RTCPeerConnection** – get access to audio or video calling facility.
- **RTCDataChannel** – get access to peer-to-peer communication.

MediaStream

The MediaStream represents synchronized streams of media, For an example, Click on HTML5 Video player in HTML5 demo section or else click [here](#).

The above example contains stream.getAudioTracks and stream.VideoTracks. If there is no audio tracks, it returns an empty array and it will check video stream, if webcam connected, stream.getVideoTracks returns an array of one MediaStreamTrack representing the stream from the webcam. A simple example is chat applications, a chat applications gets stream from web camera, rear camera, microphone.

Sample code of MediaStream

```
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);
}
navigator.getUserMedia({audio:true}, gotStream);
```

Screen capture

It's also possible in chrome browser with mediaStreamSource and it required HTTPS. This feature is not yet available in opera. Sample demo is available at [here](#)

Session Control, Network & Media Information

Web RTC required peer-to-peer communication between browsers. This mechanism required signalling, network information, session control and media information. Web developers can choose different mechanism to communicate between the browsers such as SIP or XMPP or any two way communications. A sample example of XHR is [here](#).

Sample code of createSignalingChannel

```
var signalingChannel = createSignalingChannel();
var pc;
var configuration = ...;

// run start(true) to initiate a call
function start(isCaller) {
    pc = new RTCPeerConnection(configuration);

    // send any ice candidates to the other peer
    pc.onicecandidate = function (evt) {
```

```

    signalingChannel.send(JSON.stringify({ "candidate": evt.candidate }));
};

// once remote stream arrives, show it in the remote video element
pc.onaddstream = function (evt) {
    remoteView.src = URL.createObjectURL(evt.stream);
};

// get the local stream, show it in the local video element and send it
navigator.getUserMedia({ "audio": true, "video": true }, function (stream) {
    selfView.src = URL.createObjectURL(stream);
    pc.addStream(stream);

    if (isCaller)
        pc.createOffer(gotDescription);

    else
        pc.createAnswer(pc.remoteDescription, gotDescription);

    function gotDescription(desc) {
        pc.setLocalDescription(desc);
        signalingChannel.send(JSON.stringify({ "sdp": desc }));
    }
});

signalingChannel.onmessage = function (evt) {
    if (!pc)
        start(false);
    var signal = JSON.parse(evt.data);

    if (signal.sdp)
        pc.setRemoteDescription(new RTCSessionDescription(signal.sdp));

    else
        pc.addIceCandidate(new RTCIceCandidate(signal.candidate));
};

```

Loading [Mathjax]/jax/output/HTML-CSS/jax.js