Data Stream from your Webcam and Microphone: Video Call with WebRTC Step 1

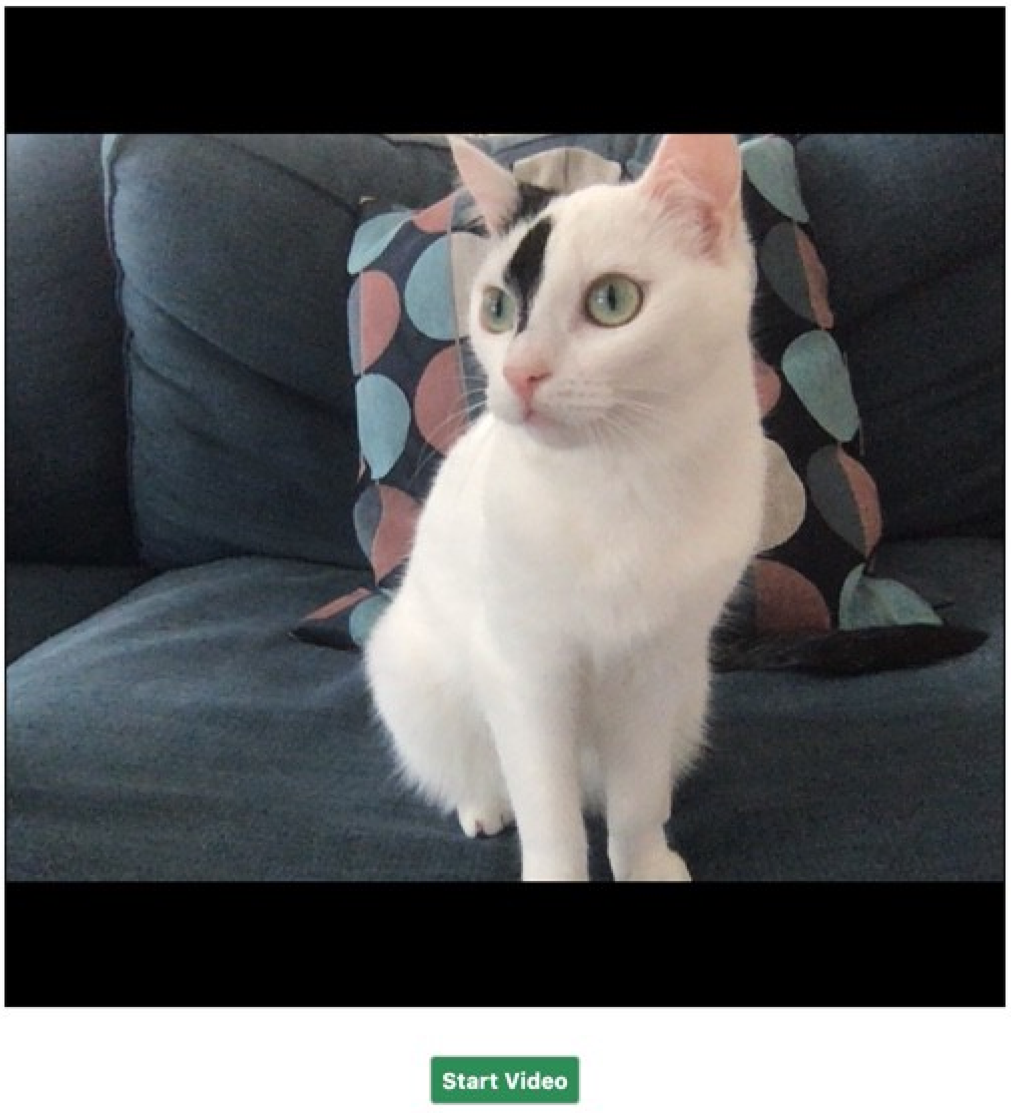
Dornhoth
Mar 18 · 4 min read ★

Photo by Austin Distel on Unsplash

Real time communication between browsers has been made possible in 2011 by WebRTC. This API enables you to create a P2P connection between two users and allow them to share data, for example to communicate over a video chat and share their screen. In this series of articles, we are going to create such a video chat. You can check the next articles:

- Step 2: Set up a Connection over WebSocket
- Step 3: Establish the WebRTC Connection
- Step 3: Find your contact

A first step to create a video chat is to have access to the user's devices, like webcam and microphone, and their stream of data. In this first article, we are going to access our devices and display the video (with audio) in our browser. The end result will look like this:



Access User Audio and Video Stream

We first need to have access to the user’s webcam and microphone. The navigator has access to the media devices through the MediaDevices interface.

```
window.navigator.mediaDevices
```

We can get a list of all the devices the browser has access to by calling in the console:

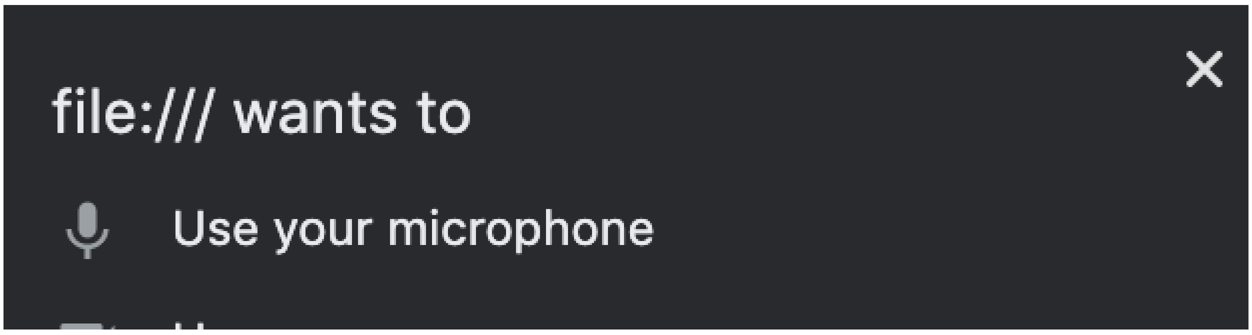
```
const devices = await window.navigator
                            .mediaDevices
                            .enumerateDevices();
```

In my browser, I get an array of five devices, one video input and five audio input. But we are not interested in the devices themselves, we want to get the data stream from them.

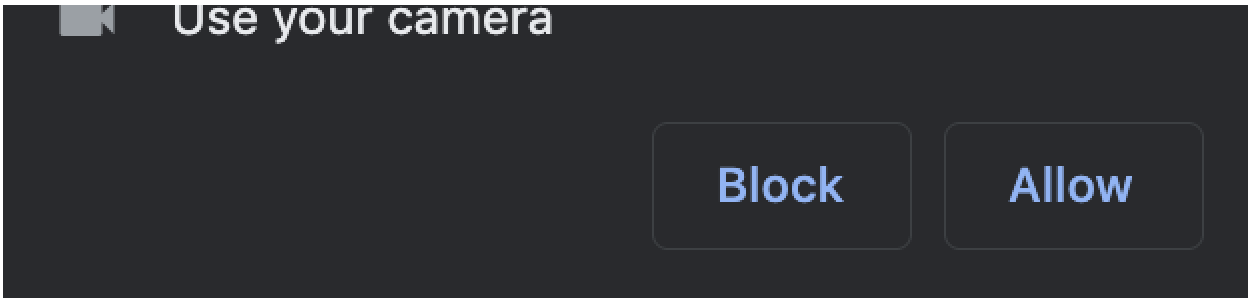
Media content, like audio and video, are represented by the MediaStream interface. To get access to the data stream, you need to call the *getUserMedia* function on the *mediaDevices* object. We need to give the types of media we want as a parameter, here audio and video. We can give a bunch of constraints as parameter, not only the media type, but also the size of the video we want, the orientation, … We keep it simple here:

```
const stream = await window.navigator.mediaDevices.getUserMedia(
  {
    video: true,
    audio: true,
  },
);
```

This returns a Promise, as you have to allow access to your devices. On Google Chrome, you should see such a dialog:


file:/// wants to

Use your microphone



If you say yes, you'll receive the MediaStream object. If your webcam has this feature, you should see a little light next to it, telling you that it is now on.

Our MediaStream object consists of two tracks, one track corresponding to one media type. You can access them by calling

```
stream.getAudioTracks();
stream.getVideoTracks();
```

These methods return arrays of tracks, each track containing some info like the kind (audio or video for example), the label, the idea, but also if the track is enabled, muted, etc.

HTML5 Video Tag

We now have access to the data stream from the microphone and webcam, we just want to display them in our browser to make sure it works. This used to be a pain in the neck, but the HTML5 video tag allows us to build a video in a page pretty easily.

```html
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>VideoChat</title>
  <script src="index.js"></script>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <div>
    <video id="video" controls autoplay></video>
  </div>
  <div>
    <button id="button">Start Video</button>
  </div>
</body>

</html>
```

index.html hosted with ❤ by GitHub view raw

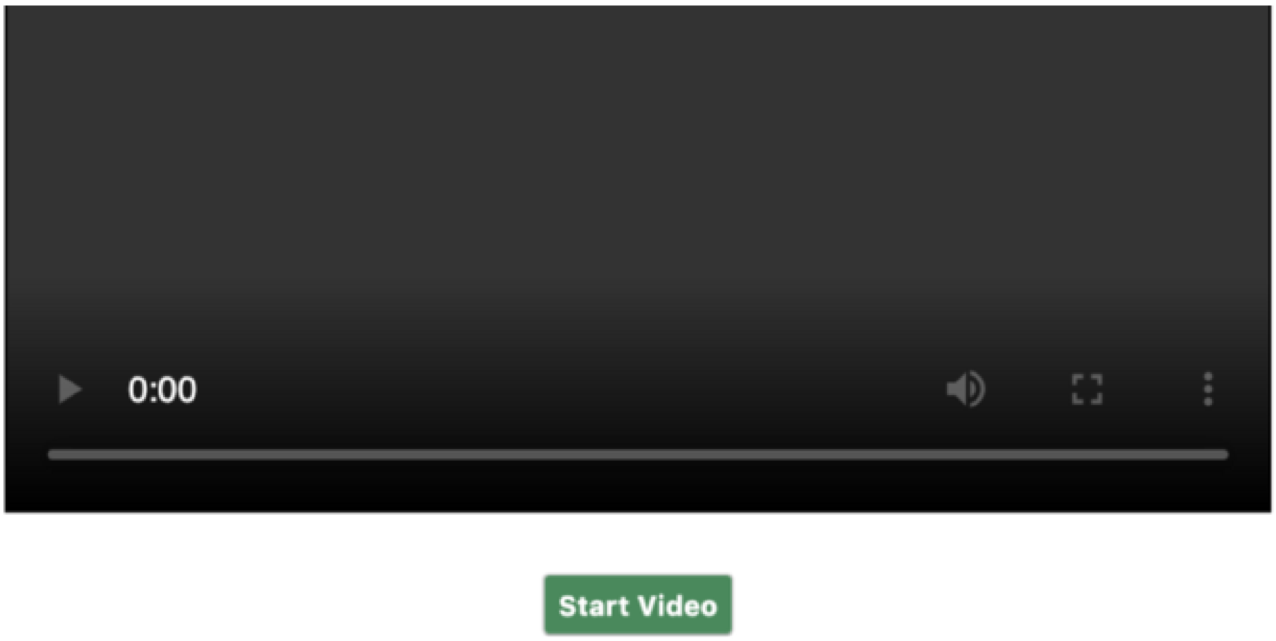
```
body {
  text-align: center;
}

video {
  width: 500px;
  height: 500px;
  margin: 20px;
}

button {
  padding: 5px;
  background-color: seagreen;
  color: white;
  border-radius: 3px;
  font-weight: bold;
}
```

styles.css hosted with ❤ by GitHub view raw

We have a video tag with controls (the play, mute, and stop buttons) and a button to start the video. Before clicking, the screen looks like this:



Let’s look at the JavaScript code:

```
(function () {
  "use strict";

  document.addEventListener('click', async event => {
    if (event.target.id === 'button') {
      const stream = await window.navigator.mediaDevices.getUserMedia({ video: true, au
      const video = document.getElementById('video');
      video.srcObject = stream;
      video.play();
    }
  });

})();
```

When you click on the button, we create the stream as we did in the previous section. Your browser should ask for your permission to start the video (we don’t handle the case where the user refuses here). Once the stream is available, it is set as the source object for the video tag and the video is started. You should now see and hear yourself.

. . .

We now have access to the data stream from the browser. The next step is to create a connection between the browsers of the users wanting to video chat. In the next article, we create a WebSocket connection between two clients.

JavaScript Web Development Programming Software Development Tech