Recording Audio from the User

By Paul Kinlan
Paul is a Developer Advocate

Many browsers now have the ability to access video and audio input from the user. However, depending on the browser it might be a full dynamic and inline experience, or it could be delegated to another app on the user's device.

Start simple and progressively

The easiest thing to do is simply ask the user for a pre-recorded file. Do this by creating a simple file input element and adding an `accept` filter that indicates we can only accept audio files, and a `capture` attribute that indicates we want to get it direct from the microphone.

```
<input type="file" accept="audio/*" capture>
```

This method works on all platforms. On desktop it will prompt the user to upload a file from the file system (ignoring the `capture` attribute). In Safari on iOS it will open up the microphone app, allowing you to record audio and then send it back to the web page; on Android it will give the user the choice of which app to use record the audio in before sending it back to the web page.

Once the user has finished recording and they are back in the website, you need to somehow get ahold of the file data. You can get quick access by attaching an **onchange** event to the input element and then reading the `files` property of the event object.

```
<input type="file" accept="audio/*" capture id="recorder">
<audio id="player" controls></audio>
<script>
  var recorder = document.getElementById('recorder');
  var player = document.getElementById('player');

  recorder.addEventListener('change', function(e) {
    var file = e.target.files[0];
    // Do something with the audio file.
    player.src =  URL.createObjectURL(file);
  });
</script>
```

Once you have access to the file you can do anything you want with it. For example, you can:

- Attach it directly to an `<audio>` element so that you can play it
- Download it to the user's device
- Upload it to a server by attaching to an `XMLHttpRequest`
- Pass it through the Web Audio API and apply filters on to it

Whilst using the input element method of getting access to audio data is ubiquitous, it is the least appealing option. We really want to get access to the microphone and provide a nice experience directly in the page.

Access the microphone interactively

Modern browsers can have a direct line to the microphone allowing us to build experiences that are fully integrated with the web page and the user will never leave the browser.

Acquire access to the microphone

We can directly access the Microphone by using an API in the WebRTC specification called `getUserMedia()`. `getUserMedia()` will prompt the user for access to their connected microphones and cameras.

If successful the API will return a `Stream` that will contain the data from either the camera or the microphone, and we can then either attach it to an `<audio>` element, attach it to a WebRTC stream, attach it to a Web Audio `AudioContext`, or save it using the `MediaRecorder` API.

To get data from the microphone we just set `audio: true` in the constraints object that is passed to the `getUserMedia()` API

```
<audio id="player" controls></audio>
<script>
  var player = document.getElementById('player');

  var handleSuccess = function(stream) {
    if (window.URL) {
      player.src = window.URL.createObjectURL(stream);
    } else {
      player.src = stream;
    }
  };

  navigator.mediaDevices.getUserMedia({ audio: true, video: false })
      .then(handleSuccess);
</script>
```

If you want to choose a particular microphone you can first enumerate the available microphones.

```
navigator.mediaDevices.enumerateDevices().then((devices) => {
  devices = devices.filter((d) => d.kind === 'audioinput');
});
```

You can then pass the deviceId that you wish to use when you call `getUserMedia`.

```
navigator.mediaDevices.getUserMedia({
  audio: {
    deviceId: devices[0].deviceId
  }
});
```

By itself, this isn't that useful. All we can do is take the audio data and play it back.

Access the raw data from the microphone

To access the raw data from the microphone we have to take the stream created by `getUserMedia()` and then use the Web Audio API to process the data. The Web Audio API is a simple API that takes input sources and connects those sources to nodes which can process the audio data (adjust Gain etc) and ultimately to a speaker so that the user can hear it.

One of the nodes that you can connect is a `ScriptProcessorNode`. This node will emit an `onaudioprocess` event every time the audio buffer is filled and you need to process it. At this point you could save the data into your own buffer and save it for later use.

```
<script>
  var handleSuccess = function(stream) {
    var context = new AudioContext();
    var source = context.createMediaStreamSource(stream);
    var processor = context.createScriptProcessor(1024, 1, 1);

    source.connect(processor);
    processor.connect(context.destination);
    processor.onaudioprocess = function(e) {
      // Do something with the data, i.e Convert this to WAV
      console.log(e.inputBuffer);
    };
  };

  navigator.mediaDevices.getUserMedia({ audio: true, video: false })
      .then(handleSuccess);
</script>
```

The data that is held in the buffers is the raw data from the microphone and you have a number of options with what you can do with the data:

- Upload it straight to the server
- Store it locally
- Convert to a dedicated file format, such as WAV, and then save it to your servers or locally

Save the data from the microphone

The easiest way to save the data from the microphone is to use the `MediaRecorder` API.

The `MediaRecorder` API will take the stream created by `getUserMedia` and then progressively save the data that is on the stream in to you preferred destination.

```
<a id="download">Download</a>
<button id="stop">Stop</button>
```