1. Working with Camera Input

To ensure the cross-browser compatibility of any device handling code in this article, use the latest WebRTC adapter kit:

```
<script src="https://webrtchacks.github.io/adapter/adapter-latest.js"></script>
```

When displaying the camera stream within a video element, make sure you flip the video element horizontally (to mirror it). This applies only when you want to capture an image from the user-facing camera (selfie or webcam):

```
video {
  transform: scale(-1, 1); /*For Firefox (& IE) */
  -webkit-transform: scale(-1, 1); /*for Chrome & Opera (& Safari) */
}
```

Now we're ready to request the stream and access from the Camera device:

```
let constraints = {}
let track

// set desired audio/video configurations
constraints = {
  audio: false,
  video: {
    width: { min: 800, ideal: 1920 },
    height: { min: 600, ideal: 1080 }
  }
};

// reference the video tag
video = document.getElementById('video_id')

// handle the video object ONLY when the actual stream starts playing
video.addEventListener('play', onVideoStart)

// request access to input devices
navigator.mediaDevices.getUserMedia(constraints)
.then( stream =>
{
  // keep the reference to the stream so it could be stopped when not needed anymore
  track = stream.getVideoTracks()[0];
```

```
  // attach the stream to the video object
  try {
    video.srcObject = stream;
  } catch (error) {
    video.srcObject = URL.createObjectURL(stream);
  }

  //Now enumerate devices
  navigator.mediaDevices.enumerateDevices().then(gotDevices)

})
.catch( err =>
{
  console.error('getUserMedia error!', error);
});

function onVideoStart()
{
  // video is now ready for handling
  // and/or to capture image from video output
}
```

In case there are multiple cameras available its a good practice to store them in case we need to offer them to the user for selection:

```
var devices

gotDevices(deviceInfos)
{
  devices = []

  for (let i = 0; i !== deviceInfos.length; ++i)
  {
    const deviceInfo = deviceInfos[i];

    if (deviceInfo.kind == 'videoinput')
    {
      var option = {};
      option.value = deviceInfo.deviceId;
      option.text = deviceInfo.label || `camera ${i}`;
      devices.push( option )
    }
  }
}
```

When you no longer need the camera of the user, and in order to respect the user's privacy, it's a good practice to stop accessing the stream input:

```
if( track ) track.stop();
video.srcObject = null;
```