navigator.getUserMedia =

navigator.getUserMedia ||

navigator.mozGetUserMedia ||

navigator.webkitGetUserMedia ||

navigator.msGetUserMedia;

function hasGetUserMedia() {

// Note: Opera builds are unprefixed.

return !!(navigator.getUserMedia || navigator.webkitGetUserMedia ||

navigator.mozGetUserMedia || navigator.msGetUserMedia);

}

**The HTML**

Please read my note about the HTML structure below:

<!--

Ideally these elements aren't created until it's confirmed that the

client supports video/camera, but for the sake of illustrating the

elements involved, they are created with markup (not JavaScript)

-->

<video id="video" width="640" height="480" autoplay></video>

<button id="snap">Snap Photo</button>

<canvas id="canvas" width="640" height="480"></canvas>

Each of these elements should be created once confirmation of camera support is confirmed, but for the sake of this tutorial, I wanted to show you what the elements look like with basic HTML.  Do note that the dimensions we're working with are 640x480.

**The JavaScript**

Since the HTML elements above are already created, the JavaScript portion will look smaller than you think:

// Grab elements, create settings, etc.

var video = document.getElementById('video');

// Get access to the camera!

if(navigator.mediaDevices && navigator.mediaDevices.getUserMedia) {

// Not adding `{ audio: true }` since we only want video now

navigator.mediaDevices.getUserMedia({ video: true }).then(function(stream) {

video.src = window.URL.createObjectURL(stream);

video.play();

});

}

/\* Legacy code below: getUserMedia

else if(navigator.getUserMedia) { // Standard

navigator.getUserMedia({ video: true }, function(stream) {

video.src = stream;

video.play();

}, errBack);

} else if(navigator.webkitGetUserMedia) { // WebKit-prefixed

navigator.webkitGetUserMedia({ video: true }, function(stream){

video.src = window.webkitURL.createObjectURL(stream);

video.play();

}, errBack);

} else if(navigator.mozGetUserMedia) { // Mozilla-prefixed

navigator.mozGetUserMedia({ video: true }, function(stream){

video.src = window.URL.createObjectURL(stream);

video.play();

}, errBack);

}

\*/

Once it's been established that the browser supports navigator.mediaDevices.getUserMedia, a simple method sets the video element's src to the user's live camera/webcam.  Calling the play method of the video then starts the element's live streaming video connection.  That's all that's required to connect your camera to the browser!

Taking a photo is only marginally more difficult.  Simply add a click listener to a generic button and and draw an image from video!

// Elements for taking the snapshot

var canvas = document.getElementById('canvas');

var context = canvas.getContext('2d');

var video = document.getElementById('video');

// Trigger photo take

document.getElementById("snap").addEventListener("click", function() {

context.drawImage(video, 0, 0, 640, 480);

});

Of course you could add some sexy image filters and make a billion dollars...but I'll save that for another post.  At minimum you could convert the [canvas snapshot to an image](https://davidwalsh.name/convert-canvas-image) though!  I'll talk about canvas image filters in the future...

[View Demo](https://davidwalsh.name/demo/camera.php)

Being able to access the camera within the browser without using third party software is an incredible advancement.  Paired with canvas and a bit of JavaScript, the camera has become quickly and easily accessible.  Not only it the camera accessible, but since canvas is ultra-flexible, we'll be able to add sexy Instagram-style image filters in the future.  For now, however, simply accessing the camera in our browser moves us miles ahead.  Have fun taking images within your browser!

### navigator.GetUserMedia 参数及使用

2016-06-20 15:06:26|  分类： [JS](http://blog.163.com/cs380637384@126/blog/#m=0&t=1&c=fks_084067082085088069085086083095082082083070080087082070) |  标签：[webrtc](http://blog.163.com/cs380637384@126/blog/#m=0&t=3&c=webrtc)  [getusermedia](http://blog.163.com/cs380637384@126/blog/#m=0&t=3&c=getusermedia)  [html5获取视频流](http://blog.163.com/cs380637384@126/blog/#m=0&t=3&c=html5%E8%8E%B7%E5%8F%96%E8%A7%86%E9%A2%91%E6%B5%81)   |举报 |字号 订阅





[下载LOFTER](http://www.lofter.com/app?act=qbbkrzydb_20150408_01) [我的照片书  |](http://yxp.163.com)

先说一下兼容情况：

chrome：21+

firefox:    17+

IE及Safari暂不支持

兼容写法：

var getUserMedia = (navigator.getUserMedia || navigator.webkitGetUserMedia ||navigator.mozGetUserMedia);

getUserMedia参数：

getUserMedia(constraints, successCallback, errorCallback);

constraints 指定请求的媒体类型，必需为一个对象类型：例如{ audio: true, video:true}  
successCallback 成功回调函数

errorCallback      失败回调函数

下面主要介绍下constraints类型的详细介绍：

兼容情况：

chrome：47+

firefox:    36+

IE及Safari暂不支持

移动端目前只有Gecko内核的36+版本支持

设置相机的分辨率：

{

 audio: true,

 video: { width: 1280, height: 720 }

}

设置相机最小分辨率：

{

 audio: true,

 video: {

width: { min:1024 },

height: { min:768 }

 }

}

设置理想的分辨率：

{

audio: true,

video: {

width: { min: 1024, ideal: 1280, max: 1920 },

height: { min: 776, ideal: 720, max: 1080 }

}

}

设置ideal后，浏览器会尝试设置，达到与ideal最近的值；

在移动设备上还可以这样设置：

{ audio: true, video: { facingMode: "user" } } //设置前置摄像头

{ audio: true, video: { facingMode: { exact: "environment" } } } //设置后置摄像头

var flag=false;

{ video: { facingMode: (flag? "user" : "environment") } }; //不解释了

最后附一个例子：

var getUserMedia = navigator.getUserMedia || navigator.webkitGetUserMedia || navigator.mozGetUserMedia;

if (getUserMedia) {

getUserMedia({ audio: true, video: { width: 1280, height: 720 } },

function(stream) {

var video = document.querySelector('video');

video.src = window.URL.createObjectURL(stream);

//firefox不支持src,由下面代替

//video.mozSrcObject= stream

video.play();

},

function(err) {

console.log("出错信息: " + err.name);

}

);

} else {

console.log("不支持getUserMedia);

}