# [MediaDevices.getUserMedia()](https://developer.mozilla.org/zh-CN/docs/Web/API/MediaDevices/getUserMedia)

我们的志愿者还没有将这篇文章翻译为 中文 (简体)。[加入我们帮助完成翻译！](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia$translate?tolocale=zh-CN)

****This is an experimental technology****  
Because this technology's specification has not stabilized, check the [compatibility table](https://developer.mozilla.org/zh-CN/docs/Web/API/MediaDevices/getUserMedia" \l "Browser_compatibility) for usage in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future versions of browsers as the specification changes.

The **MediaDevices.getUserMedia()** method prompts the user for permission to use one video and/or one audio input device such as a camera or screensharing and/or a microphone. If the user provides permission, then the returned [Promise](https://developer.mozilla.org/en-US/docs/Web/API/Promise" \o "The Promise interface represents a proxy for a value not necessarily known at its creation time. It allows you to associate handlers to an asynchronous action's eventual success or failure. This lets asynchronous methods return values like synchronous methods: instead of the final value, the asynchronous method returns a promise of having a value at some point in the future.) is resolved with the resulting [MediaStream](https://developer.mozilla.org/en-US/docs/WebRTC/MediaStream_API" \l "LocalMediaStream) object. If the user denies permission, or media is not available, then the promise is rejected with PermissionDeniedErroror NotFoundError respectively. Note that it is possible for the returned promise to neither resolve nor reject, as the user is not required to make a choice.

## **Syntax**[EDIT](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia$edit" \l "Syntax)

navigator.mediaDevices.getUserMedia(constraints)

.then(function(mediaStream) { ... })

.catch(function(error) { ... })

### Parameters

**constraints**

Is a [MediaStreamConstraints](https://developer.mozilla.org/en-US/docs/Web/API/MediaStreamConstraints" \o "The documentation about this has not yet been written; please consider contributing!) object specifying the types of media to request, along with any requirements for each type.

The constraints parameter is a MediaStreamConstraints object with two members: video andaudio, describing the media types requested. Either or both must be specified. If the browser cannot find all media tracks with the specified types that meet the constraints given, then the returned promise is rejected with NotFoundError.

The following requests both audio and video without any specific requirements:

{ audio: true, video: true }

While information about a user's cameras and microphones are inaccessible for privacy reasons, an application can request the camera and microphone capabilities it needs and wants, using additional constraints. The following expresses a preference for 1280x720 camera resolution:

{

audio: true,

video: { width: 1280, height: 720 }}

The browser will try to honor this, but may return other resolutions if an exact match is not available, or the user overrides it.

To require a capability, use the keywords min, max, or exact (a.k.a. min == max). The following demands a minimum resolution of 1280x720:

{

audio: true,

video: {

width: { min: 1280 },

height: { min: 720 }

}}

If no camera exists with this resolution or higher, then the returned promise will be rejected withNotFoundError, and the user will not be prompted.

The reason for the difference in behavior is that the keywords min, max, and exact are inherently mandatory, whereas plain values and a keyword called ideal are not. Here's a fuller example:

{

audio: true,

video: {

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

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

}}

An ideal value, when used, has gravity, which means that the browser will try to find the setting (and camera, if you have more than one), with the smallest [fitness distance](https://w3c.github.io/mediacapture-main/" \l "dfn-fitness-distance) from the ideal values given.

Plain values are inherently ideal, which means that the first of our resolution examples above could have been written like this:

{

audio: true,

video: {

width: { ideal: 1280 },

height: { ideal: 720 }

}}

Not all constraints are numbers. For example, on mobile devices, the following will prefer the front camera (if one is available) over the rear one:

{ audio: true, video: { facingMode: "user" } }

To require the rear camera, use:

{ audio: true, video: { facingMode: { exact: "environment" } } }

### Return value

A [Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise" \o "The Promise object is used for asynchronous computations. A Promise represents an operation that hasn't completed yet, but is expected in the future.) that resolves to a [MediaStream](https://developer.mozilla.org/en-US/docs/Web/API/MediaStream" \o "The MediaStream interface represents a stream of media content. A stream consists of several tracks such as video or audio tracks.) object.

### Exceptions

Rejections of the returned promise are made with a [DOMException](https://developer.mozilla.org/en-US/docs/Web/API/DOMException" \o "The DOMException interface represents an abnormal event (called an exception) which occurs as a result of calling a method or accessing a property of a web API. This is basically how error conditions are described in web APIs.) object. Possible exceptions are:

**AbortError**

Although the user and operating system both granted access to the hardware device, and no hardware issues occurred that would throw the NotReadableError exception, some problem occurred which prevented the device from being used.

**NotAllowedError**

The user has specified that the current browsing instance is not permitted access to the device; or the user has denied access for the current session; or the user has denied all access to user media devices globally.

Older versions of the specification used SecurityError for this instead; SecurityError has taken on a new meaning.

**NotFoundError**

No media tracks of the type specified were found that satisfy the given constraints.

**NotReadableError**

Although the user granted permission to use the matching devices, a hardware error occurred at the operating system, browser, or Web page level which prevented access to the device.

**OverConstrainedError**

The specified constraints resulted in no candidate devices which met the criteria requested. The exception is an object of type OverconstrainedError, and has a constraint property whose value is a constraint which was impossible to meet, and a message property containing a human-readable string explaining the problem.

Because this exception can be thrown even when the user has not yet granted permission to use the underlying device, it can potentially be used as a fingerprinting surface.

**SecurityError**

User media support is disabled on the [Document](https://developer.mozilla.org/en-US/docs/Web/API/Document" \o "The Document interface represents any web page loaded in the browser and serves as an entry point into the web page's content, which is the DOM tree. The DOM tree includes elements such as <body> and <table>, among many others. It provides functionality global to the document, like how to obtain the page's URL and create new elements in the document.) on which getUserMedia() was called. The mechanism by which user media support is enabled and disabled is left up to the individual user agent.

**TypeError**

The list of constraints specified is empty, or has all constraints set to false.

## ****Example**s**[EDIT](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia$edit" \l "Examples)

### Using the Promise

This example assigns the returned [MediaStream](https://developer.mozilla.org/en-US/docs/Web/API/MediaStream" \o "The MediaStream interface represents a stream of media content. A stream consists of several tracks such as video or audio tracks.) object to the appropriate element.

var p = navigator.mediaDevices.getUserMedia({ audio: true, video: true });

p.then(function(mediaStream) {

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

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

video.onloadedmetadata = function(e) {

// Do something with the video here.

};

});

p.catch(function(err) { console.log(err.name); }); // always check for errors at the end.

### Width and height

Here's an example of using mediaDevices.getUserMedia(), including a polyfill to cope with older browsers.

var promisifiedOldGUM = function(constraints) {

// First get ahold of getUserMedia, if present

var getUserMedia = (navigator.getUserMedia ||

navigator.webkitGetUserMedia ||

navigator.mozGetUserMedia);

// Some browsers just don't implement it - return a rejected promise with an error

// to keep a consistent interface

if(!getUserMedia) {

return Promise.reject(new Error('getUserMedia is not implemented in this browser'));

}

// Otherwise, wrap the call to the old navigator.getUserMedia with a Promise

return new Promise(function(resolve, reject) {

getUserMedia.call(navigator, constraints, resolve, reject);

});

}

// Older browsers might not implement mediaDevices at all, so we set an empty object firstif(navigator.mediaDevices === undefined) {

navigator.mediaDevices = {};}

// Some browsers partially implement mediaDevices. We can't just assign an object// with getUserMedia as it would overwrite existing properties.// Here, we will just add the getUserMedia property if it's missing.if(navigator.mediaDevices.getUserMedia === undefined) {

navigator.mediaDevices.getUserMedia = promisifiedOldGUM;}

// Prefer camera resolution nearest to 1280x720.var constraints = { audio: true, video: { width: 1280, height: 720 } };

navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {

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

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

video.onloadedmetadata = function(e) {

video.play();

};}).catch(function(err) {

console.log(err.name + ": " + err.message);});

### Frame rate

Lower frame-rates may be desirable in some cases, like WebRTC transmissions with bandwidth restrictions.

var constraints = { video: { frameRate: { ideal: 10, max: 15 } } };

### Front and back camera

On mobile phones.

var front = false;

document.getElementById('flip-button').onclick = function() { front = !front; };

var constraints = { video: { facingMode: (front? "user" : "environment") } };

## **Permissions**[EDIT](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia$edit" \l "Permissions)

To use getUserMedia() in an installable app (for example, a [Firefox OS app](https://developer.mozilla.org/en-US/Apps/Build/Building_apps_for_Firefox_OS/Firefox_OS_app_beginners_tutorial)), you need to specify one or both of the following fields inside your manifest file:

"permissions": {

"audio-capture": {

"description": "Required to capture audio using getUserMedia()"

},

"video-capture": {

"description": "Required to capture video using getUserMedia()"

}}

See [permission: audio-capture](https://developer.mozilla.org/en-US/Apps/Developing/App_permissions" \l "audio-capture) and [permission: video-capture](https://developer.mozilla.org/en-US/Apps/Developing/App_permissions" \l "video-capture) for more information.

## **Specifications**[EDIT](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia$edit" \l "Specifications)

|  |  |  |
| --- | --- | --- |
| **Specification** | **Status** | **Comment** |
| [Media Capture and Streams The definition of 'MediaDevices.getUserMedia()' in that specification.](http://w3c.github.io/mediacapture-main/" \l "dom-mediadevices-getusermedia) | Editor's Draft | Initial definition |

## **Browser compatibility**[EDIT](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia$edit" \l "Browser_compatibility)

* **Desktop**

* Mobile

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature | Chrome | Firefox (Gecko) | Internet Explorer | Opera | Safari (WebKit) |
| Stream API | (Yes)[1][3] 47 | [36](https://developer.mozilla.org/en-US/Firefox/Releases/36" \o "Released on 2015-02-24.) (36) [2] | No support | (Yes)[1] | No support |

[1] Older versions of Chrome and Opera implement navigator.webkitGetUserMedia, the prefixed version of the legacy [navigator.getUserMedia](https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia" \o "The Navigator.getUserMedia() method prompts the user for permission to use 0 or 1 video and 0 or 1 audio input device such as a camera, a shared screen, or a microphone. If the user provides permission, then the successCallback is invoked with the resulting MediaStream object as its argument. If the user denies permission or media is not available, then the errorCallback is called with PermissionDeniedError or NotFoundError respectively. Note that it is possible for neither completion callback to be called, as the user is not required to make a choice.) method.

In Chrome, this promise-based interface is only available through the [adapter.js](https://github.com/webrtc/adapter) polyfill, or using the flag chrome://flags/#enable-experimental-web-platform-features.

Chrome uses an outdated constraint syntax, but the syntax described here is available through the [adapter.js](https://github.com/webrtc/adapter) polyfill.

[2] Older versions of Firefox implement navigator.mozGetUserMedia, the prefixed version of the legacy[navigator.getUserMedia](https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia" \o "The Navigator.getUserMedia() method prompts the user for permission to use 0 or 1 video and 0 or 1 audio input device such as a camera, a shared screen, or a microphone. If the user provides permission, then the successCallback is invoked with the resulting MediaStream object as its argument. If the user denies permission or media is not available, then the errorCallback is called with PermissionDeniedError or NotFoundError respectively. Note that it is possible for neither completion callback to be called, as the user is not required to make a choice.) method.

This promise-based interface and the constraint syntax described here is available as of Firefox 38. Earlier versions (32-37) used an outdated constraint syntax, but the syntax described here, as well as the promise-based interface, is available there through the [adapter.js](https://github.com/webrtc/adapter) polyfill.

Firefox 49 includes changes to bring the thrown [exceptions](https://developer.mozilla.org/zh-CN/docs/Web/API/MediaDevices/getUserMedia" \l "Exceptions) up to date with the specification, including the change to the meaning of the SecurityError exception.

Opera uses an outdated constraint syntax, but the syntax described here is available through the [adapter.js](https://github.com/webrtc/adapter) polyfill.

[3] Chrome throws error if the page serving the script is loaded from insecure origin (i.e. HTTP).

## **See also**[EDIT](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia$edit" \l "See_also)

* The older [navigator.getUserMedia](https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia) legacy API.
* [navigator.enumerateDevices](https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/enumerateDevices) - learn the types and number of devices the user has available.
* [WebRTC](https://developer.mozilla.org/en-US/docs/WebRTC) - the introductory page to the API
* [MediaStream API](https://developer.mozilla.org/en-US/docs/WebRTC/MediaStream_API) - the API for the media stream objects
* [Taking webcam photos](https://developer.mozilla.org/en-US/docs/WebRTC/taking_webcam_photos) - a tutorial on using getUserMedia() for taking photos rather than video.

## **文档标签和贡献者**

**标签：**

* [API](https://developer.mozilla.org/zh-CN/docs/tag/API)

* [audio](https://developer.mozilla.org/zh-CN/docs/tag/audio)

* [Audio](https://developer.mozilla.org/zh-CN/docs/tag/Audio)

* [Experimental](https://developer.mozilla.org/zh-CN/docs/tag/Experimental)

* [MediaDevices](https://developer.mozilla.org/zh-CN/docs/tag/MediaDevices)

* [Method](https://developer.mozilla.org/zh-CN/docs/tag/Method)

* [Reference](https://developer.mozilla.org/zh-CN/docs/tag/Reference)

* [video](https://developer.mozilla.org/zh-CN/docs/tag/video)

* [Video](https://developer.mozilla.org/zh-CN/docs/tag/Video)

**此页面的贡献者：** [Jib](https://developer.mozilla.org/zh-CN/profiles/Jib), [ottoetc](https://developer.mozilla.org/zh-CN/profiles/ottoetc), [fscholz](https://developer.mozilla.org/zh-CN/profiles/fscholz), [Sheppy](https://developer.mozilla.org/zh-CN/profiles/Sheppy), [j1mr10rd4n](https://developer.mozilla.org/zh-CN/profiles/j1mr10rd4n), [fippo](https://developer.mozilla.org/zh-CN/profiles/fippo), [myakura](https://developer.mozilla.org/zh-CN/profiles/myakura), [jamix](https://developer.mozilla.org/zh-CN/profiles/jamix), [Sole](https://developer.mozilla.org/zh-CN/profiles/Sole), [createMo](https://developer.mozilla.org/zh-CN/profiles/createMo), [drob](https://developer.mozilla.org/zh-CN/profiles/drob),[connormckelvey](https://developer.mozilla.org/zh-CN/profiles/connormckelvey), [jpmedley](https://developer.mozilla.org/zh-CN/profiles/jpmedley), [mborn319](https://developer.mozilla.org/zh-CN/profiles/mborn319), [jwhitlock](https://developer.mozilla.org/zh-CN/profiles/jwhitlock), [Sebastianz](https://developer.mozilla.org/zh-CN/profiles/Sebastianz), [shamakry](https://developer.mozilla.org/zh-CN/profiles/shamakry), [teoli](https://developer.mozilla.org/zh-CN/profiles/teoli), [Mkmelin](https://developer.mozilla.org/zh-CN/profiles/Mkmelin), [yati](https://developer.mozilla.org/zh-CN/profiles/yati)

**最后编辑者:** [Jib](https://developer.mozilla.org/zh-CN/profiles/Jib), Jul 21, 2016, 10:43:19 AM