HTML Media Capture

W3C Recommendation 01 February 2018

This version:
https://www.w3.org/TR/2018/REC-html-media-capture-20180201/

Latest published version:
https://www.w3.org/TR/html-media-capture/

Latest editor's draft:
https://w3c.github.io/html-media-capture/

Test suite:
https://w3c-test.org/html-media-capture/

Implementation report:
https://www.w3.org/2009/dap/wiki/ImplementationStatus

Previous version:
https://www.w3.org/TR/2017/PR-html-media-capture-20171128/

Editors:
Anssi Kostiainen, Intel
Ilkka Oksanen, Nokia (until May 10, 2012)
Dominique Hazaël-Massieux, W3C (until May 10, 2012)

Translations:
ру́сский язы́к
한국어
日本語

Participate:
public-device-apis@w3.org
GitHub w3c/html-media-capture
GitHub w3c/html-media-capture/issues
GitHub w3c/html-media-capture/commits

Please check the **errata** for any errors or issues reported since publication.

The English version of this specification is the only normative version. Non-normative **translations** may also be available.

Abstract

The *HTML Media Capture* specification defines an HTML form extension that facilitates user access to a device's media capture mechanism, such as a camera, or microphone, from within a file upload control.

Status of This Document

Status Update (April 2018): This paragraph is informative. The specification was updated in-place to include links to translations and to incorporate the existing errata, which only contains a non-normative change.

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

An HTML Media Capture Proposed Recommendation was published on 28 November 2017, no further normative changes have been made since then. Errata for this document are recorded as issues. The implementation report produced for this version demonstrates there are two independent interoperable implementations.

This document was published by the Device and Sensors Working Group as a Recommendation. Comments regarding this document are welcome. Please send them to public-device-apis@w3.org (subscribe, archives) or file an issue on GitHub.

Please see the Working Group's implementation report.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 March 2017 W3C Process Document.

Table of Contents

1. Introduction

This section is non-normative.

The *HTML Media Capture* specification extends the `HTMLInputElement` interface with a `capture` attribute. The `capture` attribute allows authors to declaratively request use of a media capture mechanism, such as a camera or microphone, from within a file upload control, for capturing media on the spot.

This extension is specifically designed to be simple and declarative, and covers a subset of the media capture functionality of the web platform. Specifically, the extension does not provide detailed author control over capture. Use cases requiring more fine-grained author control may be met by using another specification, *Media Capture and Streams* [MEDIACAPTURE-STREAMS]. For example, access to real-time media streams from the hosting device is out of scope for this specification.

2. Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words *MUST*, *MUST NOT*, and *SHOULD* are to be interpreted as described in [RFC2119].

This specification defines conformance criteria that apply to a single product: the ***user agent*** that implements the interfaces that it contains.

Implementations that use ECMAScript to implement the APIs defined in this specification must implement them in a manner consistent with the ECMAScript Bindings defined in the Web IDL specification [WEBIDL-1], as this specification uses that specification and terminology.

3. Terminology

The ***`input`*** element, its ***`type`*** attribute, ***`HTMLInputElement`*** interface, ***`accept`*** attribute, ***File Upload*** state, ***enumerated attribute***, ***missing value default***, ***invalid value default***, and ***reflect*** are defined in [HTML51].

The ***`[CEReactions]`*** WebIDL extended attribute is defined in [custom-elements].

The ***`VideoFacingModeEnum`*** enumeration is defined in [MEDIACAPTURE-STREAMS].

The ***`FileList`*** interface is defined in [FILE-API].

In this specification, the term ***capture control type*** refers to a specialized type of a file picker control that is optimized, for the user, for directly capturing media of a MIME type specified by the `accept` attribute, using a media capture mechanism in its preferred facing mode.

The term ***media capture mechanism*** refers to a device's local media capture device, such as a camera or microphone.

The ***preferred facing mode*** is a hint for the direction of the device's media capture mechanism to be used.

4. Security and privacy considerations

This section is non-normative.

A User Agent implementation of this specification is advised to seek user consent before initiating capture of content by microphone or camera. This may be necessary to meet regulatory, legal and best practice requirements related to the privacy of user data. In addition, the User Agent implementation is advised to provide an indication to the user when an input device is enabled and make it possible for the user to terminate such capture. Similarly, the User Agent is advised to offer user control, such as to allow the user to:

- select the exact media capture device to be used if there exist multiple devices of the same type (e.g. a front-facing camera in addition to a primary camera).
- disable sound capture when in the video capture mode.

This specification builds upon the security and privacy protections provided by the `<input type="file">` [HTML51] and the [FILE-API] specifications; in particular, it is expected that any offer to start capturing content from the user's device would require a specific user interaction on an HTML element that is entirely controlled by the user agent.

Implementors should take care to prevent additional leakage of privacy-sensitive data from captured media. For instance, embedding the user's location in the metadata of captured media (e.g. EXIF) might transmit more private data than the user is expecting.

5. The *`capture`* attribute

When an `input` element's `type` attribute is in the File Upload state, and its `accept` attribute is specified, the rules in this section apply.

WebIDL

```
partial interface HTMLInputElement {
    [CEReactions]
    attribute DOMString capture;
};
```

The `capture` attribute is an enumerated attribute whose state specifies the preferred facing mode for the media capture mechanism.

The attribute's keywords are ***`user`*** and ***`environment`***, which map to the respective states *user* and *environment*. The semantics of the states *user* and *environment* mirror the similarly named enumeration values defined in `VideoFacingModeEnum`.

In addition, there is a third state, the *implementation-specific* state.

The missing value default is the *implementation-specific* state. The invalid value default is also the *implementation-specific* state.

> NOTE
>
> If the user agent is unable to support the preferred facing mode, it can fall back to the implementation-specific default facing mode that maps to the *implementation-specific* state that indicates the implementation is to act according to its default behavior.

The `capture` IDL attribute *MUST* reflect the respective content attribute of the same name.

When the `capture` attribute is specified, the user agent *SHOULD* invoke a file picker of the specific capture control type.

When the `capture` attribute is specified, the user agent *MUST NOT* save the captured media to any data storage, local or remote.

> NOTE
> When scripts gain access to the files selected from the file picker (represented by a `FileList` object), they can use various mechanisms to store the captured media. These mechanisms are out of scope for this specification.

If the `accept` attribute's value is set to a MIME type that has no associated capture control type, the user agent *MUST* act as if there was no `capture` attribute.

A. Examples

This section is non-normative.

The following examples demonstrate how to give hints that it is preferred for the user to capture media of a specific MIME type using the media capture capabilities of the hosting device. Both a simple declarative example using an HTML form, as well as a more advanced example including scripting, are presented.

- To take a picture using the device's user-facing camera, and upload the picture taken using an HTML form:

EXAMPLE 1

```
<form action="server.cgi" method="post" enctype="multipart/form-data">
  <input type="file" name="image" accept="image/*" capture="user">
  <input type="submit" value="Upload">
</form>
```

- Or alternatively, to capture video using the device's local video camera facing the environment:

EXAMPLE 2

```
<form action="server.cgi" method="post" enctype="multipart/form-data">
  <input type="file" name="video" accept="video/*" capture="environment">
  <input type="submit" value="Upload">
</form>
```

- Or alternatively, to capture audio using the device's local microphone (without preferred facing mode defined, falls back to the implementation-specific default facing mode):

EXAMPLE 3

```
<form action="server.cgi" method="post" enctype="multipart/form-data">
  <input type="file" name="audio" accept="audio/*" capture>
  <input type="submit" value="Upload">
</form>
```

- For more advanced use cases, specify the `capture` attribute in markup:

EXAMPLE 4

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

And handle the file upload in script via `XMLHttpRequest`:

EXAMPLE 5

```
var input = document.querySelector('input[type=file]'); // see Example 4

input.onchange = function () {
  var file = input.files[0];

  upload(file);
  drawOnCanvas(file);   // see Example 6
  displayAsImage(file); // see Example 7
};

function upload(file) {
  var form = new FormData(),
      xhr = new XMLHttpRequest();

  form.append('image', file);
  xhr.open('post', 'server.php', true);
  xhr.send(form);
}
```

The image can also be displayed on the client-side without uploading it e.g. for client-side image editing purposes, using the `FileReader` and a `canvas` element:

EXAMPLE 6

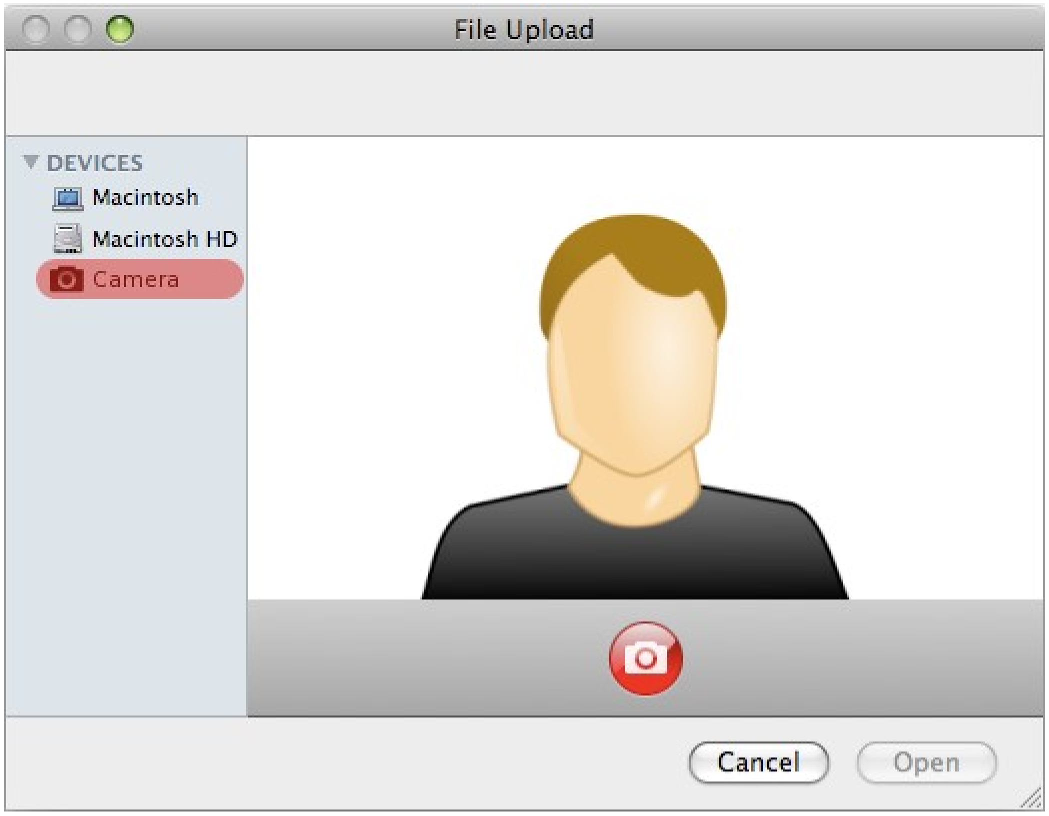
```
function drawOnCanvas(file) {
  var reader = new FileReader();

  reader.onload = function (e) {
    var dataURL = e.target.result,
        c = document.querySelector('canvas'), // see Example 4
        ctx = c.getContext('2d'),
        img = new Image();

    img.onload = function() {
      c.width = img.width;
      c.height = img.height;
      ctx.drawImage(img, 0, 0);
    };

    img.src = dataURL;
  };

  reader.readAsDataURL(file);
}
```

Or alternatively, to just display the image, using the `createObjectURL()` method and an `img` element:

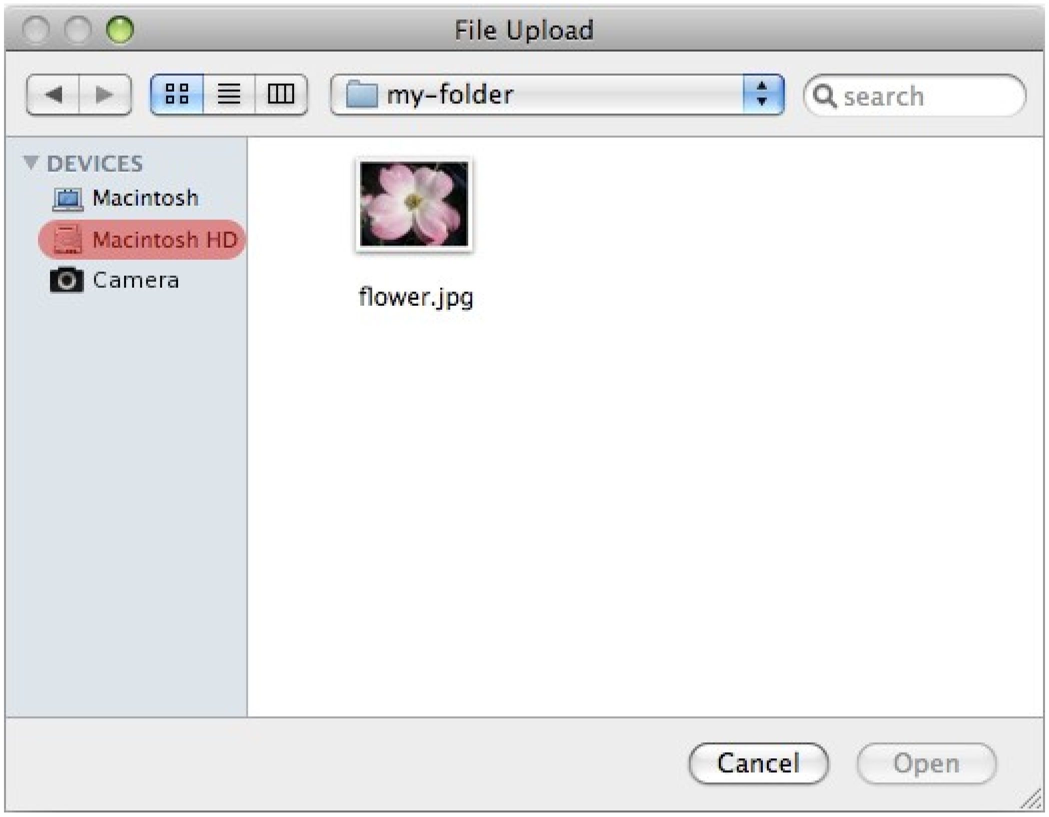
EXAMPLE 7

```
function displayAsImage(file) {
  var imgURL = URL.createObjectURL(file),
      img = document.createElement('img');

  img.onload = function() {
    URL.revokeObjectURL(imgURL);
  };

  img.src = imgURL;
  document.body.appendChild(img);
}
```

When an `input` element's `accept` attribute is set to `image/*` and the `capture` attribute is specified as in the Example 1 or Example 4, the file picker can be rendered as presented below:



When the attribute is not specified, the file picker can be rendered as represented below:



B. References

B.1 Normative references

[custom-elements]
Custom Elements. Domenic Denicola. W3C. 13 October 2016. W3C Working Draft. URL: https://www.w3.org/TR/custom-elements/

[HTML51]
HTML 5.1 2nd Edition. Steve Faulkner; Arron Eicholz; Travis Leithead; Alex Danilo. W3C. 3 October 2017. W3C Recommendation. URL: https://www.w3.org/TR/html51/

[MEDIACAPTURE-STREAMS]
Media Capture and Streams. Daniel Burnett; Adam Bergkvist; Cullen Jennings; Anant Narayanan; Bernard Aboba. W3C. 3 October 2017. W3C Candidate Recommendation. URL: https://www.w3.org/TR/mediacapture-streams/

[RFC2119]
Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[WEBIDL-1]
WebIDL Level 1. Cameron McCormack. W3C. 15 December 2016. W3C Recommendation. URL: https://www.w3.org/TR/2016/REC-WebIDL-1-20161215/

B.2 Informative references

[FILE-API]
File API. Marijn Kruisselbrink. W3C. 26 October 2017. W3C Working Draft. URL: https://www.w3.org/TR/FileAPI/

[HTML]
HTML Standard. Anne van Kesteren; Domenic Denicola; Ian Hickson; Philip Jägenstedt; Simon Pieters. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[WEBIDL]
Web IDL. Cameron McCormack; Boris Zbarsky; Tobie Langel. W3C. 15 December 2016. W3C Editor's Draft. URL: https://heycam.github.io/webidl/

↑