MIME types (IANA media types)

A **media type** (also known as a **Multipurpose Internet Mail Extensions or MIME type**) indicates the nature and format of a document, file, or assortment of bytes. MIME types are defined and standardized in IETF's RFC 6838 .

The Internet Assigned Numbers Authority (IANA) is responsible for all official MIME types, and you can find the most up-to-date and complete list at their Media Types page.

> **Warning:** Browsers use the MIME type, *not the file extension*, to determine how to process a URL, so it's important that web servers send the correct MIME type in the response's `Content-Type` header. If this is not correctly configured, browsers are likely to misinterpret the contents of files, sites will not work correctly, and downloaded files may be mishandled.

Structure of a MIME type

A simplest MIME type consists of a *type* and a *subtype*. A MIME type comprises these strings concatenated with a slash (`/`). No whitespace is allowed in a MIME type:

```
type/subtype
```

The ***type*** represents the general category into which the data type falls, such as `video` or `text` .

The ***subtype*** identifies the exact kind of data of the specified type the MIME type represents. For example, for the MIME type `text` , the subtype might be `plain` (plain text), `html` (HTML source code), or `calendar` (for iCalendar/ `.ics`) files.

Each type has its own set of possible subtypes. A MIME type always has both a type and a subtype, never just one or the other.

An optional **parameter** can be added to provide additional details:

```
type/subtype;parameter=value
```

For example, for any MIME type whose main type is `text` , you can add the optional

For example, for any MIME type whose main type is `text`, you can add the optional `charset` parameter to specify the character set used for the characters in the data. If no `charset` is specified, the default is ASCII (`US-ASCII`) unless overridden by the user agent's settings. To specify a UTF-8 text file, the MIME type `text/plain;charset=UTF-8` is used.

MIME types are case-insensitive but are traditionally written in lowercase. The parameter values can be case-sensitive.

Types

There are two classes of type: **discrete** and **multipart**. Discrete types are types which represent a single file or medium, such as a single text or music file, or a single video. A multipart type is one which represents a document that's comprised of multiple component parts, each of which may have its own individual MIME type; or, a multipart type may encapsulate multiple files being sent together in one transaction. For example, multipart MIME types are used when attaching multiple files to an email.

Discrete types

The discrete types currently registered with the IANA are:

`application`

Any kind of binary data that doesn't fall explicitly into one of the other types; either data that will be executed or interpreted in some way or binary data that requires a specific application or category of application to use. Generic binary data (or binary data whose true type is unknown) is `application/octet-stream`. Other common examples include `application/pdf`, `application/pkcs8`, and `application/zip`. (Registration at IANA)

`audio`

Audio or music data. Examples include `audio/mpeg`, `audio/vorbis`. (Registration at IANA)

`example`

Reserved for use as a placeholder in examples showing how to use MIME types. These should never be used outside of sample code listings and documentation. `example` can also be used as a subtype; for instance, in an example related to working with audio on the web, the MIME type `audio/example` can be used to indicate that the type is a placeholder and should be replaced with an appropriate one when using the code in the real world.

`font`

Font/typeface data. Common examples include `font/woff`, `font/ttf`, and `font/otf`. (Registration at IANA)

`image`
: Image or graphical data including both bitmap and vector still images as well as animated versions of still image formats such as animated GIF or APNG. Common examples are `image/jpeg`, `image/png`, and `image/svg+xml`. (Registration at IANA)

`model`
: Model data for a 3D object or scene. Examples include `model/3mf` and `model/vrml`. (Registration at IANA)

`text`
: Text-only data including any human-readable content, source code, or textual data such as comma-separated value (CSV) formatted data. Examples include: `text/plain`, `text/csv`, and `text/html`. (Registration at IANA)

`video`
: Video data or files, such as MP4 movies (`video/mp4`). (Registration at IANA)

For text documents without a specific subtype, `text/plain` should be used. Similarly, for binary documents without a specific or known subtype, `application/octet-stream` should be used.

Multipart types

Multipart types indicate a category of document broken into pieces, often with different MIME types; they can also be used — especially in email scenarios — to represent multiple, separate files which are all part of the same transaction. They represent a **composite document**.

With the exception of `multipart/form-data`, used in the `POST` method of HTML Forms, and `multipart/byteranges`, used with `206 Partial Content` to send part of a document, HTTP doesn't handle multipart documents in a special way: the message is transmitted to the browser (which will likely show a "Save As" window if it doesn't know how to display the document).

There are two multipart types:

`message`
: A message that encapsulates other messages. This can be used, for instance, to represent an email that includes a forwarded message as part of its data, or to allow sending very large messages in chunks as if it were multiple messages. Examples include `message/rfc822` (for forwarded or replied-to message quoting) and `message/partial` to allow breaking a large message into smaller ones automatically

to be reassembled by the recipient. (Registration at IANA)

`multipart`

Data that is comprised of multiple components which may individually have different MIME types. Examples include `multipart/form-data` (for data produced using the `FormData` API) and `multipart/byteranges` (defined in RFC 7233: 5.4.1 and used with HTTP's `206` "Partial Content" response returned when the fetched data is only part of the content, such as is delivered using the `Range` header). (Registration at IANA)

Important MIME types for Web developers

application/octet-stream

This is the default for binary files. As it means *unknown binary* file, browsers usually don't execute it, or even ask if it should be executed. They treat it as if the `Content-Disposition` header was set to `attachment`, and propose a "Save As" dialog.

text/plain

This is the default for textual files. Even if it really means "unknown textual file," browsers assume they can display it.

> **Note:** `text/plain` does not mean "any kind of textual data." If they expect a specific kind of textual data, they will likely not consider it a match. Specifically if they download a `text/plain` file from a `<link>` element declaring a CSS file, they will not recognize it as a valid CSS file if presented with `text/plain`. The CSS mime type `text/css` must be used.

text/css

CSS files used to style a Web page **must** be sent with `text/css`. If a server doesn't recognize the `.css` suffix for CSS files, it may send them with `text/plain` or `application/octet-stream` MIME types. If so, they won't be recognized as CSS by most browsers and will be ignored.

text/html

All HTML content should be served with this type. Alternative MIME types for XHTML (like `application/xhtml+xml`) are mostly useless nowadays.

> **Note:** Use `application/xml` or `application/xhtml+xml` if you want XML's strict parsing rules, `<![CDATA[…]]>` sections, or elements that aren't from HTML/SVG/MathML namespaces.

text/javascript

Per the HTML specification, JavaScript files should always be served using the MIME type `text/javascript`. No other values are considered valid, and using any of those may result in scripts that do not load or run.

For historical reasons, the MIME Sniffing Standard (the definition of how browsers should interpret media types and figure out what to do with content that doesn't have a valid one) allows JavaScript to be served using any MIME type that essentially matches any of the following:

- `application/javascript`
- `application/ecmascript`
- `application/x-ecmascript` 👎
- `application/x-javascript` 👎
- `text/javascript`
- `text/ecmascript`
- `text/javascript1.0` 👎
- `text/javascript1.1` 👎
- `text/javascript1.2` 👎
- `text/javascript1.3` 👎
- `text/javascript1.4` 👎
- `text/javascript1.5` 👎
- `text/jscript` 👎
- `text/livescript` 👎
- `text/x-ecmascript` 👎
- `text/x-javascript` 👎

> **Note:** Even though any given user agent may support any or all of these, you should only use `text/javascript`. It's the only MIME type guaranteed to work now and into the future.

Some content you find may have a `charset` parameter at the end of the

Some content you find may have a `charset` parameter at the end of the `text/javascript` media type, to specify the character set used to represent the code's content. This is not valid, and in most cases will result in a script not being loaded.

Image types

Files whose MIME type is `image` contain image data. The subtype specifies which specific image file format the data represents.

The following image types are used commonly enough to be considered *safe* for use on web pages:

- `image/apng` : Animated Portable Network Graphics (APNG)
- `image/avif` : AV1 Image File Format (AVIF)
- `image/gif` : Graphics Interchange Format (GIF)
- `image/jpeg` : Joint Photographic Expert Group image (JPEG)
- `image/png` : Portable Network Graphics (PNG)
- `image/svg+xml` : Scalable Vector Graphics (SVG)
- `image/webp` : Web Picture format (WEBP)

The Image file type and format guide provides information and recommendations about when to use the different image formats.

Audio and video types

As is the case for images, HTML doesn't mandate that web browsers support any specific file and codec types for the `<audio>` and `<video>` elements, so it's important to consider your target audience and the range of browsers (and versions of those browsers) they may be using when choosing the file type and codecs to use for media.

Our media container formats guide provides a list of the file types that are commonly supported by web browsers, including information about what their special use cases may be, any drawbacks they have, and compatibility information, along with other details.

The audio codec and video codec guides list the various codecs that web browsers often support, providing compatibility details along with technical information such as how many audio channels they support, what sort of compression is used, and what bit rates and so forth they're useful at. The codecs used by WebRTC guide expands upon this by specifically covering the codecs supported by the major web browsers, so you can choose the codecs that best cover the range of browsers you wish to support.

As for MIME types of audio or video files, they typically specify the container format (file type). The optional codecs parameter can be added to the MIME type to further specify which codecs

to use and what options were used to encode the media, such as codec profile, level, or other such information.

The most commonly used MIME types used for web content are listed below. This isn't a complete list of all the types that may be available, however. See the media container formats guide for that.

MIME type	Audio or video type

`audio/wave`
`audio/wav`
`audio/x-wav`
`audio/x-pn-wav`
	An audio file in the WAVE container format. The PCM audio codec (WAVE codec "1") is often supported, but other codecs have limited support (if any).

`audio/webm`	An audio file in the WebM container format. Vorbis and Opus are the codecs officially supported by the WebM specification.
`video/webm`	A video file, possibly with audio, in the WebM container format. VP8 and VP9 are the most common video codecs; Vorbis and Opus the most common audio codecs.
`audio/ogg`	An audio file in the Ogg container format. Vorbis is the most common audio codec used in such a container; however, Opus is now supported by Ogg as well.
`video/ogg`	A video file, possibly with audio, in the Ogg container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec, although Opus is becoming more common.
`application/ogg`	An audio or video file using the Ogg container format. Theora is the usual video codec used within it; Vorbis is the usual audio codec.

multipart/form-data

The `multipart/form-data` type can be used when sending the values of a completed HTML Form from browser to server.

As a multipart document format, it consists of different parts, delimited by a boundary (a string starting with a double dash `--`). Each part is its own entity with its own HTTP headers, `Content-Disposition`, and `Content-Type` for file uploading fields.

```
Content-Type: multipart/form-data; boundary=aBoundaryString
(other headers associated with the multipart document as a whole)

--aBoundaryString
Content-Disposition: form-data; name="myFile"; filename="img.jpg"
Content-Type: image/jpeg

(data)
--aBoundaryString
Content-Disposition: form-data; name="myField"

(data)
--aBoundaryString
(more subparts)
--aBoundaryString--
```

The following `<form>`:

```
<form action="http://localhost:8000/" method="post" enctype="mult
  <label>Name: <input name="myTextField" value="Test"></label>
  <label><input type="checkbox" name="myCheckBox"> Check</label>
  <label>Upload file: <input type="file" name="myFile" value="test.tx
  <button>Send the file</button>
</form>
```

will send this message:

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecl
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=--------------------------
Content-Length: 465

-----------------------------8721656041911415653955004498
Content-Disposition: form-data; name="myTextField"

Test
-----------------------------8721656041911415653955004498
Content-Disposition: form-data; name="myCheckBox"
```

```
on
----------------------------872165604191141565395500449 8
Content-Disposition: form-data; name="myFile"; filename="test.txt"
Content-Type: text/plain


Simple file.
----------------------------872165604191141565395500449 8--
```

multipart/byteranges

The `multipart/byteranges` MIME type is used to send partial responses to the browser.

When the `206 Partial Content` status code is sent, this MIME type indicates that the document is composed of several parts, one for each of the requested ranges. Like other multipart types, the `Content-Type` uses a `boundary` to separate the pieces. Each piece has a `Content-Type` header with its actual type and a `Content-Range` of the range it represents.

```
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Type: multipart/byteranges; boundary=3d6b6a416f9b5
Content-Length: 385

--3d6b6a416f9b5
Content-Type: text/html
Content-Range: bytes 100-200/1270

eta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="vieport" content
--3d6b6a416f9b5
Content-Type: text/html
Content-Range: bytes 300-400/1270

-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica
--3d6b6a416f9b5--
```

Importance of setting the correct MIME type

Most web servers send unrecognized resources as the `application/octet-stream` MIME type. For security reasons, most browsers do not allow setting a custom default action

for such resources, forcing the user to save it to disk to use it.

Some common incorrect server configurations:

- RAR-compressed files. In this case, the ideal would be the true type of the original files; this is often impossible as .RAR files can hold several resources of different types. In this case, configure the server to send `application/x-rar-compressed`.
- Audio and video. Only resources with the correct MIME Type will be played in `<video>` or `<audio>` elements. Be sure to specify the correct media type for audio and video.
- Proprietary file types. Avoid using `application/octet-stream` as most browsers do not allow defining a default behavior (like "Open in Word") for this generic MIME type. A specific type like `application/vnd.mspowerpoint` lets users open such files automatically in the presentation software of their choice.

MIME sniffing

In the absence of a MIME type, or in certain cases where browsers believe they are incorrect, browsers may perform *MIME sniffing* — guessing the correct MIME type by looking at the bytes of the resource.

Each browser performs MIME sniffing differently and under different circumstances. (For example, Safari will look at the file extension in the URL if the sent MIME type is unsuitable.) There are security concerns as some MIME types represent executable content. Servers can prevent MIME sniffing by sending the `X-Content-Type-Options` header.

Other methods of conveying document type

MIME types are not the only way to convey document type information:

- Filename suffixes are sometimes used, especially on Microsoft Windows. Not all operating systems consider these suffixes meaningful (such as Linux and MacOS), and there is no guarantee they are correct.
- Magic numbers. The syntax of different formats allows file-type inference by looking at their byte structure. For example, GIF files start with the `47 49 46 38 39` hexadecimal value (`GIF89`), and PNG files with `89 50 4E 47` (`.PNG`). Not all file types have magic numbers, so this is not 100% reliable either.

See also

- Web media technologies
- Guide to media types used on the web
- Properly configuring server MIME types

- Properly configuring server MIME types

Last modified: Dec 1, 2021, by MDN contributors