MIME 类型

▸ **在本文章中**

MIME类型是一种通知客户端其接收文件的多样性的机制:文件后缀名在网页上并没有明确的意义。因此，使服务器设置正确的传输类型非常重要，所以正确的MIME类型与每个文件一同传输给服务器。在网络资源进行连接时，浏览器经常使用MIME类型来决定执行何种默认行为。

因为有很多种文件的存在，所以也有很多种MIME类型。在这个词条中，我们只列出对于开发者最重要的一些类型，但是你可以在 MIME类型表 中找到所有的文件类型。

为了传递文件类型信息，MIME类型并不是唯一的选择:

- 在微软Windows系统中，文件命后缀经常会被使用。但并不是所有操作系统认为这些后缀是有意义的（尤其是Linux和Mac OS），它们更偏好于外部MIME类型，但是也不能保证它们就是对的。
- Magic numbers. 在一些文件的语法中，允许使用其构造类型来区分。比如说GIF文件以16进制值47 49 46 38开始 [GIF89] ，而PNG文件是以89 50 4E 47 [.PNG] 。并不是所有类型的文件都会有magic numbers，所以这也不是一个100%可靠的方式。

正因如此，在web中MIME类型的正确设置非常重要。浏览器和服务器尝试使用后缀名或magic number去定义MIME类型，以用来检测其对应的一致性，以及尝试对一个通用的类型寻找正确的MIME类型。

语法

通用结构

```
type/subtype
```

MIME的组成结构非常简单；由类型与子类型两个字符串中间用“/”分隔而组成。并不允许空格存在。type 表示可以被分为复数子类的独立类型。subtype 表示细分后的每个类型。

MIME类型对大小写不敏感，但是传统写法都是小写。

独立类型

```
text/plain
text/html
image/jpeg
image/png
audio/mpeg
audio/ogg
audio/*
video/mp4
application/octet-stream
application/json
…
```

独立 类型表明了对文件的分类，例如如下所示：

类型	描述	典型示例
`text`	表明文件是普通文本，理论上是可读的语言	`text/plain`, `text/html`, `text/css`, `text/javascript`
`image`	表明是某种图像。不包括视频，但是动态图（比如动态gif）也使用image类型	`image/gif`, `image/png`, `image/jpeg`, `image/bmp`, `image/webp`
`audio`	表明是某种音频文件	`audio/midi`, `audio/mpeg`, `audio/webm`, `audio/ogg`, `audio/wav`
`video`	表明是某种视频文件	`video/webm`, `video/ogg`
`application`	表明是某种二进制数据	`application/octet-stream`, `application/pkcs12`, `application/vnd.mspowerpoint`, `application/xhtml+xml`, `application/xml`, `application/pdf`, `application/json`

对于text文件类型若没有特定的subtype，就使用 `text/plain`。类似的，二进制文件没有特定或已知的 `subtype`，即使用 `application/octet-stream` 。

Multipart types

```
multipart/form-data
multipart/byteranges
```

Multipart types 表示细分领域的文件类型的种类，经常对应不同的 MIME类型。这是 复合文 件的一种表现方式。对于 `multipart/form-data` 的例外部分，可以使用HTML Forms 和 `POST` 方法来解决，以及使用状态码`206` `Partial Content` 来发送整个文件的子集，而HTTP不能处理的复合文件使用一个特殊的方式：将信息直接传送给浏览器（这时可能会建立一个“另存为”窗口，但是却不知道如何去显示内联文件。）

重要的MIME类型

`application/octet-stream`

这是应用程序文件的默认值。意思是 未知的应用程序文件 ，浏览器一般不会自动执行或询问执行。浏览器会像对待 设置了 HTTP头`Content-Disposition` 值为“附件”的文件一样来对待这类文件。

`text/plain`

文本文件默认值。意思是 未知的文本文件 ，浏览器认为是可以直接展示的。

> `text/plain` 并不是意味着 某种文本数据 。如果浏览器想要一个文本文件的明确类型，浏览器并不会考虑他们是否匹配。比如说，如果通过一个表明是下载CSS文件的`<link>`链接下载了一个 `text/plain` 文件。如果提供的信息是 text/plain，浏览器并不会认出这是有效的CSS文件。CSS类型需要使用 text/css。

`text/css`

任何一个CSS文件想要在网页上被解释执行就必须为`text/css` 文件。但是服务器经常不会分辨出使用`.css`后缀的CSS文件，并且将其MIME类型设置为`text/plain` 或 `application/octet-stream` 发送：在这种情况下，文件并不能被浏览器识别为CSS文件并且会被直接忽略。所以特别注意要给CSS文件设置正确的类型。

`text/html`

HTML 文件的类型.另一个用于XHTML的MIME类型是 application/xml+html. 但是现在很少用了（ HTML5统一了这个格式 ）。

图片类型

图片类型是在网页中使用的，唯一被广泛识别以及考虑过web安全的类型：

MIME 类型	图片类型

MIME 类型	图片类型
`image/gif`	GIF 图片 (无损耗压缩方面被PNG所替代)
`image/jpeg`	JPEG 图片
`image/png`	PNG 图片
`image/svg+xml`	SVG图片 (矢量图)

此处的类型划分有一定的争议，有人认为此处应该增加 WebP (`image/webp`)，但是每个新增的图片类型都会增加代码的数量，这会带来一些新的安全问题，所以浏览器供应商对于添加类型非常小心。

另外的一些图片种类可以在Web文档中找到。比如很多浏览器支持 icon 类型的图标作为 favicons或者类似的图标，并且浏览器在MIME类型中的 image/x-icon 支持ICO图像。

音频与视频类型

HTML并没有明确定义被用于`<audio>` 和`<video>`元素所支持的文件类型，所以在web上使用的只有相对较小的一组类型。 Media formats supported by the HTML audio and video elements 这篇文章解释了可以被使用的解码器或视频文件格式。

在web环境最常用的视频文件的格式，是以下这些这些文件类型：

MIME 类型	音频或视频类型

`audio/wave`
`audio/wav`
`audio/x-wav`
`audio/x-pn-wav`
	音频流媒体文件。一般支持PCM音频编码，其他解码器有限支持（如果有的话）。

`audio/webm`	WebM 音频文件格式。Vorbis 和 Opus 是其最常用的解码器。
`video/webm`	采用WebM视频文件格式的音视频文件。VP8 和 VP9是其最常用的视频解码器。Vorbis 和 Opus 是其最常用的音频解码器。
`audio/ogg`	采用OGG多媒体文件格式的音频文件。 Vorbis 是这个多媒体文件格式最常用的音频解码器。
`video/ogg`	采用OGG多媒体文件格式的音视频文件。常用的视频解码器是 Theora；音频解码器为Vorbis 。
`application/ogg`	采用OGG多媒体文件格式的音视频文件。常用的视频解码器是 Theora；音频解码器为Vorbis 。

MIME 类型	音频或视频类型
`application/json`	application/json (MIME_type) https://en.wikipedia.org/wiki/Media_type#Common_examples https://www.iana.org/assignments/media-types/application/json

`multipart/form-data`

`multipart/form-data` 可用于HTML表单从浏览器发送信息给服务器。 作为多部分文档格式，它由边界线（一个由'--'开始的字符串）划分出的不同部分组成。每一部分有自己的实体，以及自己的 HTTP 请求头，`Content-Disposition`和 `Content-Type` 用于文件上传领域，最常用的（`Content-Length` 因为边界线作为分隔符而被忽略）。

```
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--
```

如下所示的表单:

```
<form action="http://localhost:8000/" method="post" enctype="multipart/for
  <input type="text" name="myTextField">
  <input type="checkbox" name="myCheckBox">Check</input>
  <input type="file" name="myFile">
  <button>Send the file</button>
</form>
```

会发送这样的请求:

```
POST / HTTP/1.1
Host: localhost:8000
```

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
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=---------------------------872
Content-Length: 465

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

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

on
-----------------------------8721656041911415653955004498
Content-Disposition: form-data; name="myFile"; filename="test.txt"
Content-Type: text/plain

Simple file.
-----------------------------8721656041911415653955004498--
```

multipart/byteranges

`multipart/byteranges` 用于把部分的响应报文发送回浏览器。当发送状态码 `206 Partial Content` 时，这个MIME类型用于指出这个文件由若干部分组成，每一个都有其请求范围。就像其他很多类型 `Content-Type` 使用分隔符来制定分界线。每一个不同的部分都有 `Content-Type` 这样的HTTP头来说明文件的实际类型，以及 `Content-Range` 来说明其范围。

```
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
```

```
Content-Range: bytes 500-600/1270

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

设置正确的MIME类型的重要性

很多web服务器使用默认的 `application/octet-stream` 来发送未知类型。出于一些安全原因，对于这些资源浏览器不允许设置一些自定义默认操作，导致用户必须存储到本地以使用。常见的导致服务器配置错误的文件类型如下所示：

- RAR编码文件。在这种情况，理想状态是，设置真实的编码文件类型；但这通常不可能（可能是服务器所未知的类型或者这个文件包含许多其他的不同的文件类型）。这这种情况服务器将发送 application/x-rar-compressed 作为MIME类型，用户不会将其定义为有用的默认操作。
- 音频或视频文件。只有正确设置了MIME类型的文件才能被 `<video>` 或 `<audio>` 识别和播放。 可参照 use the correct type for audio and video。
- 专有文件类型。是专有文件时需要特别注意。使用 application/octet-stream 作为特殊处理是不被允许的：对于一般的MIME类型浏览器不允许定义默认行为（比如“在Word中打开”）

MIME 嗅探

在缺失 MIME 类型或客户端认为文件设置了错误的 MIME 类型时，浏览器可能会通过查看资源来进行MIME嗅探。每一个浏览器在不同的情况下会执行不同的操作。因为这个操作会有一些安全问题，有的 MIME 类型表示可执行内容而有些是不可执行内容。浏览器可以通过请求头 `Content-Type` 来设置 `X-Content-Type-Options` 以阻止MIME嗅探。

其他

- Properly configuring server MIME types
- Media formats supported by the HTML audio and video elements
- https://www.iana.org/assignments/media-types/application/json
- https://en.wikipedia.org/wiki/Media_type#Common_examples