视频和音频内容

现在我们可以轻松地向网页添加简单的图像，下一步是开始向您的 HTML 文档添加视频和音频播放器！在本文中，我们将着眼于使用 `<video>` and `<audio>` 元素来做到这一点；然后，我们将通过查看如何为您的视频添加字幕/副标题来结束。

先决条件：	基本的计算机知识、 安装的基本软件、使用文件的基本知识 、熟悉 HTML 基础知识（如 HTML 入门中所述）和 HTML 中的图像。
客观的：	了解如何将视频和音频内容嵌入网页，以及如何为视频添加字幕/副标题。

网络上的视频和音频

Flash 和Silverlight 等专有的基于插件的技术使第一批在线视频和音频成为可能。这两个都存在安全性和可访问性问题，现在已经过时，有利于原生 HTML 解决方案 `<video>` 和元素以及用于控制它们的JavaScript API `<audio>` 的可用性。我们不会在这里看 JavaScript — 只是可以用 HTML 实现的基本基础。

我们不会教您如何制作音频和视频文件——这需要完全不同的技能组合。我们为您提供了示例音频和视频文件以及示例代码 供您自己进行实验，以防您无法掌握自己的实验。

> **注意：** 在开始之前，您还应该知道有很多 OVP（在线视频提供商），如YouTube 、Dailymotion 和Vimeo ，以及在线音频提供商，如Soundcloud 。这些公司提供了一种方便、简单的方式来托管和使用视频，因此您不必担心巨大的带宽消耗。OVP 甚至通常会提供现成的代码，用于在您的网页中嵌入视频/音频；如

> 果您使用该路线，则可以避免我们在本文中讨论的一些困难。我们将在下一篇文章中更多地讨论这种服务。

<video> 元素

该 `<video>` 元素使您可以非常轻松地嵌入视频。一个非常简单的示例如下所示：

```
<video src="rabbit320.webm" controls>
  <p>
    Your browser doesn't support HTML video. Here is a
    <a href="rabbit320.webm">link to the video</a> instead.
  </p>
</video>
```

笔记的特点是：

`src`

与元素相同 `<img>`， `src` (source) 属性包含您要嵌入的视频的路径。它以完全相同的方式工作。

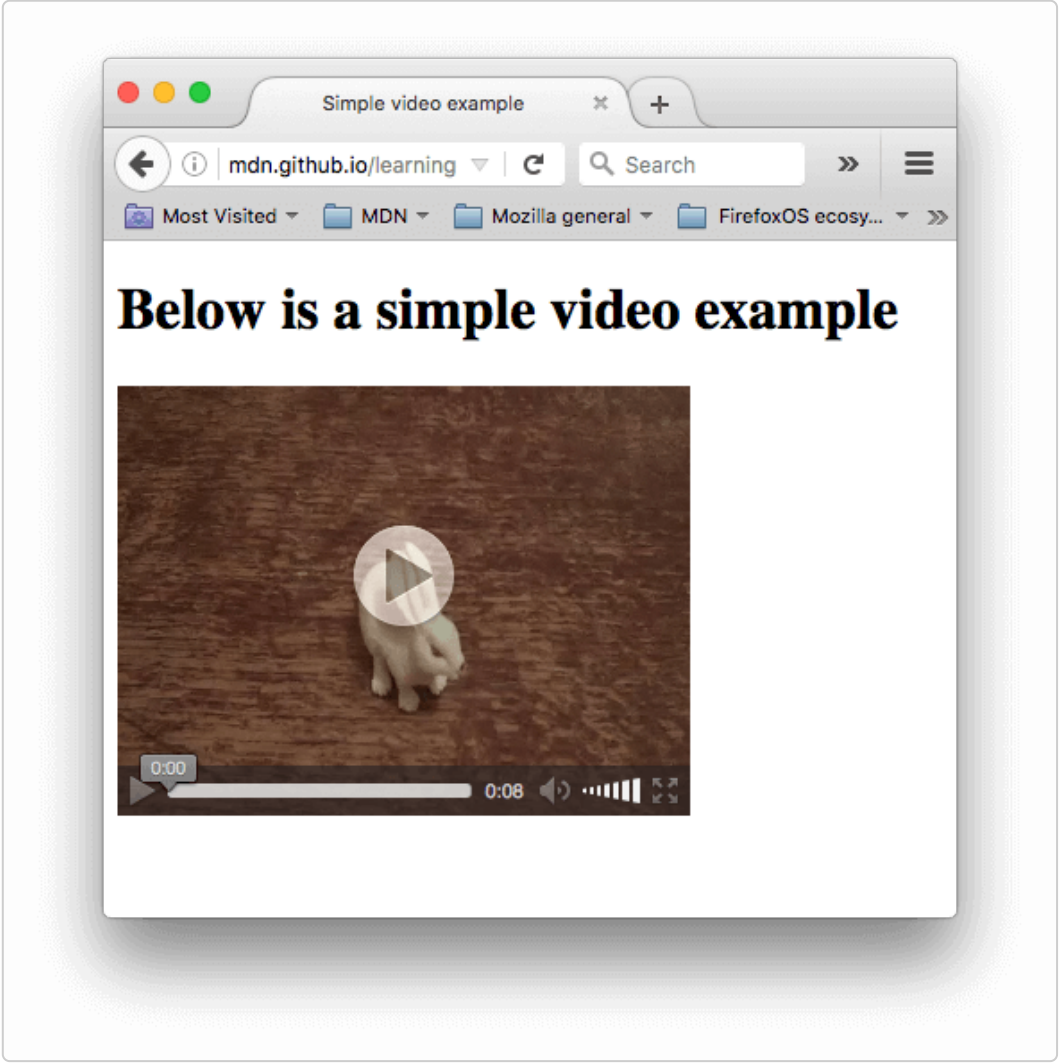
`controls`

用户必须能够控制视频和音频播放（这对患有癫痫症 的人尤其重要。）您必须使用该 `controls` 属性来包含浏览器自己的控制界面，或者使用适当的JavaScript API构建您的界面。至少，界面必须包括启动和停止媒体以及调整音量的方法。

`<video>` 标签内的段落

这称为**回退内容**——如果访问页面的浏览器不支持该元素，则会显示此内容 `<video>`，从而允许我们为旧浏览器提供回退。这可以是任何你喜欢的；在这种情况下，我们提供了一个指向视频文件的直接链接，因此用户至少可以通过某种方式访问它，而不管他们使用的是什么浏览器。

嵌入的视频看起来像这样：



您可以在此处试用示例 （另请参阅源代码 。）

使用多种源格式来提高兼容性

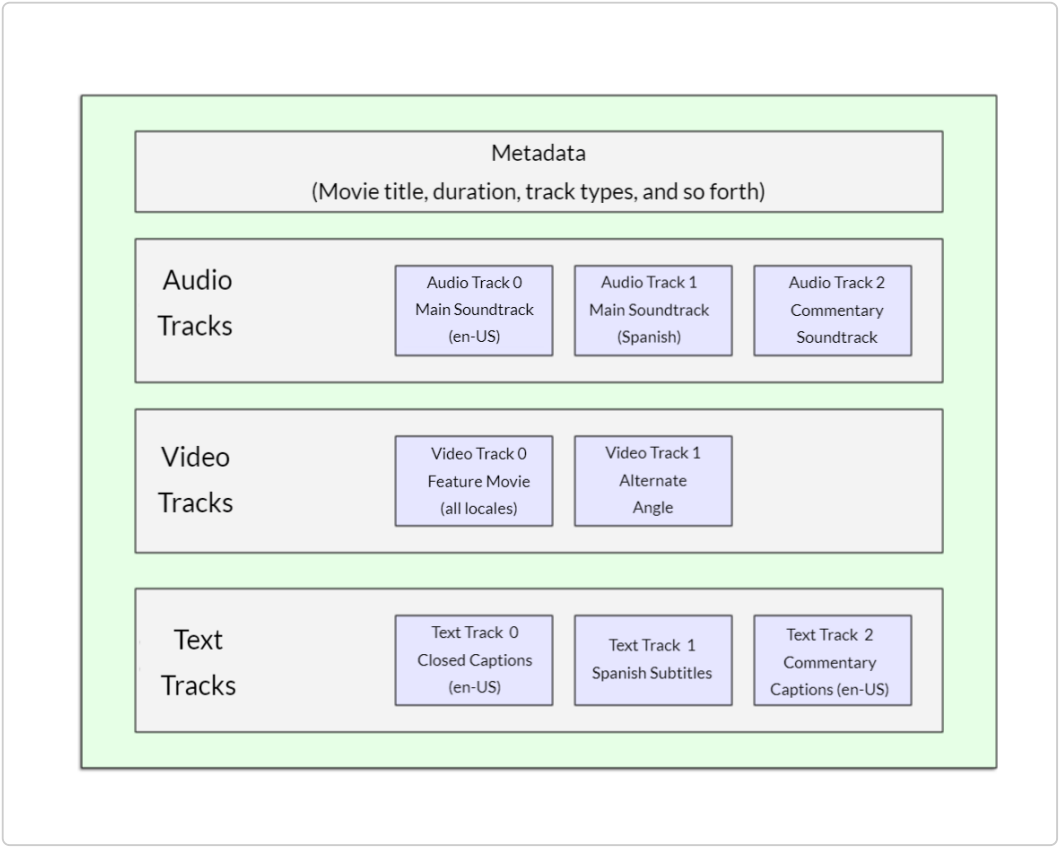
上面的例子有问题。视频可能无法为您播放，因为不同的浏览器支持不同的视频（和音频）格式。幸运的是，您可以采取一些措施来防止这成为一个问题。

媒体文件的内容

首先，让我们快速浏览一下术语。MP3、MP4 和 WebM 等格式称为容器格式。它们定义了一种结构，其中存储构成媒体的音频和/或视频轨道，以及描述媒体的元数据、用于编码其频道的编解码器等。

包含一部电影的 WebM 文件具有一个主视频轨道和一个备用角度轨道，加上英语和西班牙语的音频，以及英语评论轨道的音频，可以如下图所示进行

概念化。还包括文本轨道，其中包含故事片的隐藏式字幕、电影的西班牙文字幕和评论的英文字幕。



容器内的音频和视频轨道以适合用于编码该媒体的编解码器的格式保存数据。音频轨道与视频轨道使用不同的格式。每个音轨都是使用音频编解码器编码的，而视频轨道是使用（您可能已经猜到的）视频编解码器编码的。正如我们之前所说，不同的浏览器支持不同的视频和音频格式，以及不同的容器格式（如 MP3、MP4 和 WebM，它们又可以包含不同类型的视频和音频）。

例如：

- WebM 容器通常将 Vorbis 或 Opus 音频与 VP8/VP9 视频打包在一起。所有现代浏览器都支持此功能，但旧版本可能无法使用。
- MP4 容器通常将 AAC 或 MP3 音频与 H.264 视频打包在一起。所有现代浏览器也支持这一点。
- Ogg 容器倾向于使用 Vorbis 音频和 Theora 视频。这在 Firefox 和 Chrome 中得到了最好的支持，但基本上已经被质量更好的 WebM 格式所取代。

有一些特殊情况。例如，对于某些类型的音频，编解码器的数据通常在没有容器或简化容器的情况下存储。一个这样的例子是 FLAC 编解码器，它最常存储在 FLAC 文件中，这些文件只是原始的 FLAC 曲目。

另一种情况是一直流行的 MP3 文件。“MP3 文件”实际上是存储在 MPEG 或 MPEG-2 容器中的 MPEG-1 音频层 III (MP3) 音轨。这一点特别有趣，因为虽然大多数浏览器不支持在 `<video>` 和 `<audio>` 元素中使用 MPEG 媒体，但由于 MP3 的流行，它们可能仍然支持它。

音频播放器倾向于直接播放音轨，例如 MP3 或 Ogg 文件。这些不需要容器。

浏览器中的媒体文件支持

> **注意：**一些流行的格式，如 MP3 和 MP4/H.264，非常好，但受专利保护；也就是说，有些专利涵盖了它们所基于的部分或全部技术。在美国，专利涵盖 MP3 到 2017 年，H.264 至少到 2027 年都受专利保护。
>
> 由于这些专利，希望实现对这些编解码器的支持的浏览器通常必须支付巨额许可费用。此外，有些人更喜欢避免使用受限制的软件，而更喜欢只使用开放格式。由于这些法律和优惠原因，Web 开发人员发现他们必须支持多种格式才能吸引所有受众。

上一节中描述的编解码器用于将视频和音频压缩成可管理的文件，因为原始音频和视频都非常大。每个网络浏览器都支持各种**编解码器**，如 Vorbis 或 H.264，用于将压缩的音频和视频转换为二进制数据并返回。每个编解码器都有自己的优点和缺点，每个容器也可能提供自己的正面和负面特征，影响您决定使用哪个。

事情变得稍微复杂一些，因为每个浏览器不仅支持一组不同的容器文件格式，而且每个浏览器都支持不同的编解码器选择。为了最大限度地提高您的网站或应用在用户浏览器上运行的可能性，您可能需要以多种格式提供您使用的每个媒体文件。如果您的站点和用户的浏览器不共享共同的媒体格式，您的媒体将无法播放。

由于确保您的应用程序的媒体在您希望访问的各种浏览器、平台和设备组合中可见的复杂性，选择编解码器和容器的最佳组合可能是一项复杂的任务。请参阅选择合适的容器以帮助选择最适合您需要的容器文件格式；同样，请参阅选择视频编解码器和选择音频编解码器以帮助选择第一个用于您的内容和目标受众的媒体编解码器。

要记住的另一件事：移动浏览器可能支持其桌面等效项不支持的其他格式，就像它们可能不支持桌面版本所支持的所有相同格式一样。最重要的是，桌面和移动浏览器都*可以*设计为卸载媒体播放的处理（针对所有媒体或仅针对它无法在内部处理的特定类型）。这意味着媒体支持部分取决于用户安装的软件。

那么我们该怎么做呢？查看以下更新的示例 （也可以在此处试用）：

```
<video controls>
  <source src="rabbit320.mp4" type="video/mp4" />
  <source src="rabbit320.webm" type="video/webm" />
  <p>
    Your browser doesn't support this video. Here is a
    <a href="rabbit320.mp4">link to the video</a> instead.
  </p>
</video>
```

在这里，我们 src 从实际 <video> 标签中取出属性，而是包含 <source> 指向其自身来源的单独元素。在这种情况下，浏览器将遍历这些 <source> 元素并播放其编解码器支持的第一个元素。如今，包括 WebM 和 MP4 源应该足以在大多数平台和浏览器上播放您的视频。

每个 <source> 元素也有一个 type 属性。这是可选的，但建议您包含它。该 type 属性包含指定的文件的MIME 类型 <source> ，浏览器可以使用 type 立即跳过他们不理解的视频。如果 type 不包括在内，浏览器将加载并尝试播放每个文件，直到找到一个可用的文件，这显然需要时间并且是对资源的不必要使用。

请参阅我们的媒体类型和格式指南，以帮助选择最适合您需要的容器和编解码器，以及查找正确的 MIME 类型以指定每个类型。

其他 <video> 功能

在显示 HTML 视频时，您还可以包含许多其他功能。看看我们的下一个例子：

```
<video
  controls
  width="400"
  height="400"
  autoplay
  loop
  muted
  preload="auto"
  poster="poster.png">
  <source src="rabbit320.mp4" type="video/mp4" />
  <source src="rabbit320.webm" type="video/webm" />
  <p>
    Your browser doesn't support this video. Here is a
    <a href="rabbit320.mp4">link to the video</a> instead.
  </p>
</video>
```

生成的 UI 看起来像这样：

功能包括：

`width` 和 `height`

您可以使用这些属性或CSS控制视频大小。在这两种情况下，视频都保持其原始的宽高比——称为**纵横比**。如果纵横比不符合您设置的尺寸，视频将水平增长以填充空间，默认情况下未填充的空间将被赋予纯色背景。

`autoplay`

使音频或视频立即开始播放，同时加载页面的其余部分。建议您不要在您的网站上使用自动播放视频（或音频），因为用户会觉得这很烦人。

`loop`

使视频（或音频）在结束时再次开始播放。这也可能很烦人，所以只有在真正需要时才使用。

`muted`

使媒体播放时默认关闭声音。

`poster`

将在播放视频之前显示的图像的 URL。它旨在用于初始屏幕或广告屏幕。

`preload`

用于缓冲大文件；它可以采用以下三个值之一：

- `"none"` 不缓冲文件
- `"auto"` 缓冲媒体文件
- `"metadata"` 仅缓冲文件的元数据

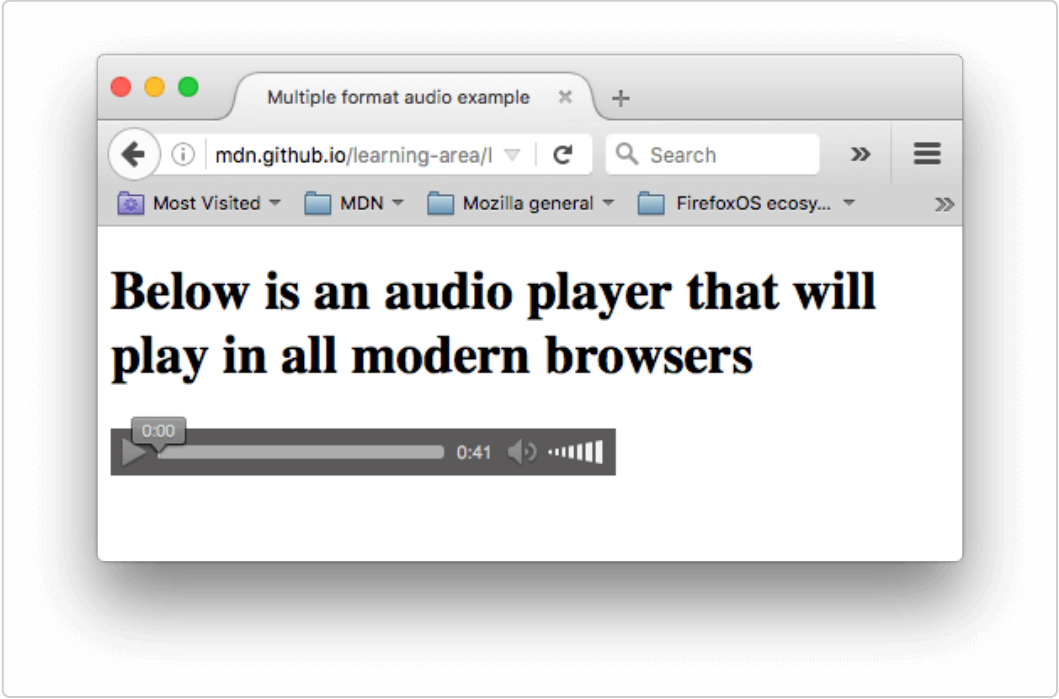
您可以在 GitHub 上 找到可以实时播放的上述示例（另请参阅源代码。）请注意，我们尚未 `autoplay` 在实时版本中包含该属性——如果视频在页面加载后立即开始播放，则您不会看不到海报！

<audio> 元素

该 `<audio>` 元素的工作方式与 元素类似 `<video>`，但存在如下所述的一些小差异。一个典型的例子可能是这样的：

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3" />
  <source src="viper.ogg" type="audio/ogg" />
  <p>
    Your browser doesn't support this audio file. Here is a
    <a href="viper.mp3">link to the audio</a> instead.
  </p>
</audio>
```

这会在浏览器中产生如下内容：



> **注意：**您可以在 GitHub 上实时运行音频演示 （另请参阅音频播放器源代码 。）

这比视频播放器占用更少的空间，因为没有视觉组件——你只需要显示控件来播放音频。与 HTML 视频的其他区别如下：

- 该 `<audio>` 元素不支持 `width` / `height` 属性——同样，没有可视化组件，因此无法为其分配宽度或高度。
- 它也不支持该 `poster` 属性——同样，没有可视化组件。

除此之外， `<audio>` 支持与以下所有相同的功能 `<video>` ——查看以上部分以获取有关它们的更多信息。

显示视频文本轨道

现在我们将讨论一个稍微高级一点的概念，了解它确实很有用。许多人不能或不想听到他们在 Web 上找到的音频/视频内容，至少在某些时候是这样。例如：

- 许多人有听觉障碍（例如听力障碍或失聪），因此根本无法清楚地听到音频。
- 其他人可能无法听到音频，因为他们处于嘈杂的环境中（例如正在播放体育比赛时拥挤的酒吧）。
- 同样，在播放音频会分散注意力或造成干扰的环境中（例如在图书馆或伙伴试图睡觉时），字幕可能非常有用。
- 不会说视频语言的人可能需要文本抄本甚至翻译来帮助他们理解媒体内容。

能够向这些人提供音频/视频中所说的文字的抄本不是很好吗？好吧，感谢 HTML 视频，您可以。为此，我们使用WebVTT文件格式和 `<track>` 元素。

> **注意：** “转录”的意思是“将口语记录为文本”。生成的文本是“抄本”。

WebVTT 是一种用于编写文本文件的格式，其中包含多个文本字符串以及元数据，例如视频中每个文本字符串应显示的时间，甚至是有限的样式/位置信息。这些文本串称为**提示**，有多种提示用于不同的目的。最常见的提示是：

字幕

外国材料的翻译，供听不懂音频中所说的话的人使用。

字幕

同步转录对话或重要声音的描述，让听不到音频的人了解发生了什么。

定时描述

媒体播放器应该说出的文本，以便向盲人或其他视障用户描述重要的视觉效果。

一个典型的 WebVTT 文件看起来像这样：

```
WEBVTT

1
00:00:22.230 --> 00:00:24.606
This is the first subtitle.

2
00:00:30.739 --> 00:00:34.074
This is the second.

…
```

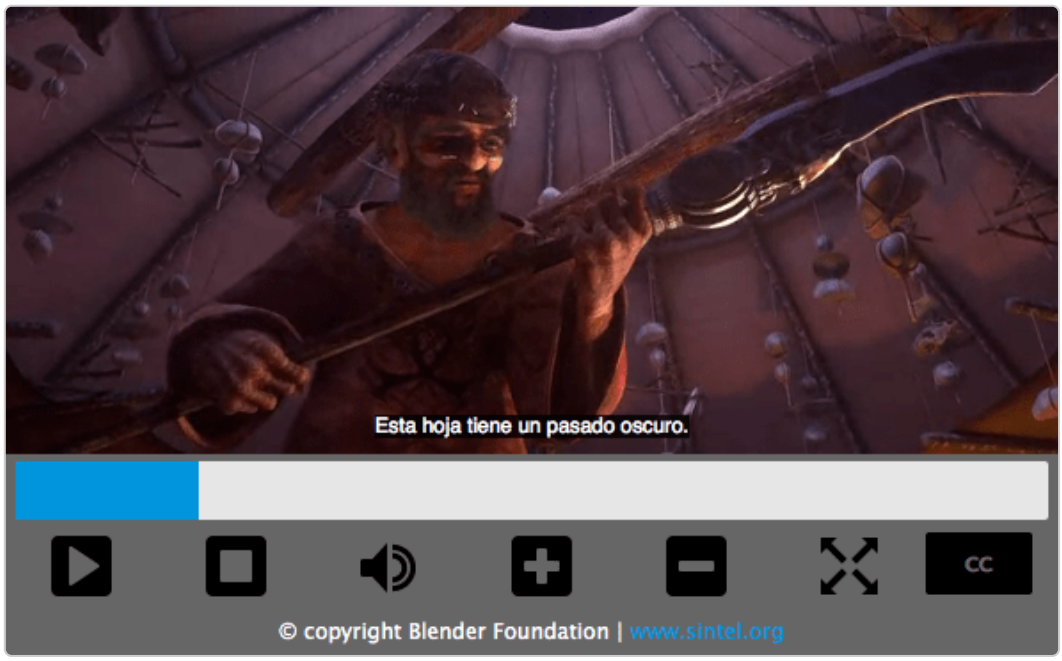
要使它与 HTML 媒体播放一起显示，您需要：

1. 将它作为 `.vtt` 文件保存在一个合理的地方。
2. 链接到 `.vtt` 包含该 `<track>` 元素的文件。 `<track>` 应该放在 `<audio>` or之内 `<video>` ，但毕竟是 `<source>` 元素。使用 `kind` 属性指定提示是 `subtitles` 、 `captions` 还是 `descriptions` 。此外，使用 `srclang` 告诉浏览器您用什么语言编写了字幕。最后，添加 `label` 以帮助读者识别他们正在搜索的语言。

这是一个例子：

```
<video controls>
  <source src="example.mp4" type="video/mp4" />
  <source src="example.webm" type="video/webm" />
  <track kind="subtitles" src="subtitles_es.vtt" srclang="es"
label="Spanish" />
</video>
```

这将导致显示有字幕的视频，有点像这样：



有关更多详细信息，包括如何添加标签，请阅读向 HTML 视频添加字幕和副标题。您可以在 GitHub 上找到本文附带的示例，该示例由 Ian Devlin 编写（另请参阅 源代码 。）该示例使用一些 JavaScript 来允许用户在不同的字幕之间进行选择。请注意，要打开字幕，您需要按“CC”按钮并选择一个选项——英语、德语或西班牙语。

> **注意：** 文本轨道还可以帮助您进行SEO，因为搜索引擎尤其依赖于文本。文本轨道甚至允许搜索引擎直接链接到视频中途的一个点。

主动学习：嵌入您自己的音频和视频

For this active learning, we'd (ideally) like you to go out into the world and record some of your own video and audio — most phones these days allow you to record audio and video very easily and, provided you can transfer it on to your computer, you can use it. You may have to do some conversion to end up with a WebM and MP4 in the case of video, and an MP3 and Ogg in the case of audio, but there are enough programs out there to allow you to do this without too much trouble, such as Miro Video Converter and Audacity . We'd like you to have a go!

If you are unable to source any video or audio, then you can feel free to use our sample audio and video files to carry out this exercise. You can also use our sample code for reference.

We would like you to:

1. Save your audio and video files in a new directory on your computer.
2. Create a new HTML file in the same directory, called `index.html`.
3. Add `<audio>` and `<video>` elements to the page; make them display the default browser controls.
4. Give both of them `<source>` elements so that browsers will find the audio format they support best and load it. These should include `type` attributes.
5. Give the `<video>` element a poster that will be displayed before the video starts to be played. Have fun creating your own poster graphic.

For an added bonus, you could try researching text tracks, and work out how to add some captions to your video.

Test your skills!

You've reached the end of this article, but can you remember the most important information? You can find some further tests to verify that you've retained this information before you move on — see Test your skills: Multimedia and embedding. Note that the third assessment question in this test assumes knowledge of some of the techniques covered in the next article, so you may want to read that before attempting it.

Summary

And that's a wrap — we hope you had fun playing with video and audio in web pages! In the next article, we'll look at other ways of

embedding content on the Web, using technologies like `<iframe>` and `<object>`.

See also

- The HTML media elements: `<audio>`, `<video>`, `<source>`, and `<track>`
- Adding captions and subtitles to video
- Audio and Video delivery: A LOT of detail about putting audio and video onto web pages using HTML and JavaScript.
- Audio and Video manipulation: A LOT of detail about manipulating audio and video using JavaScript (for example adding filters.)
- Web media technologies
- Guide to media types and formats on the web
- Event reference > Media

This page was last modified on Mar 12, 2023 by MDN contributors.