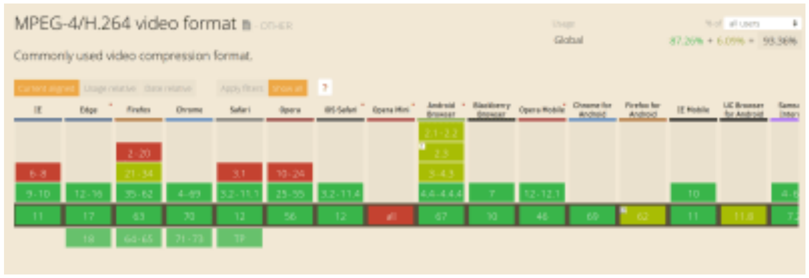
Convert Animated GIFs To Video

The first step is to convert GIF files to a video format. MP4 is the most widely supported format in browsers with almost 94% of all browsers enjoying support, so that's a safe default.



MPEG-4/H.264 video format - OTHER

Usage % of all users

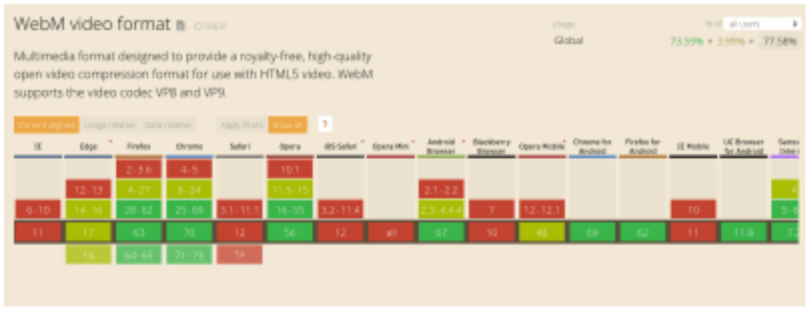
Global 87.26% + 6.09% = 93.36%

Commonly used video compression format.

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsu
								2.1-2.2							
		2-20						2.3							
6-8		21-34		3.1	10-24			3-4.3							
9-10	12-16	35-62	4-69	3.2-11.1	25-55	3.2-11.4		4.4-4.4.4	7	12-12.1			10		4-
11	17	63	70	12	56	12	all	67	10	46	69	62	11	11.8	7.
	18	64-65	71-73	TP											

94% of all browsers support the MP4 format (Large preview)

Another option is the WebM format which offers high quality videos, often comparable to an MP4, but usually at a reduced file size. However, at this time, browser support is not as widespread so you can't just go replacing MP4 files with their WebM equivalents.



WebM video format - OTHER

Usage % of all users

Global 73.59% + 3.99% = 77.58%

Multimedia format designed to provide a royalty-free, high-quality open video compression format for use with HTML5 video. WebM supports the video codec VP8 and VP9.

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsu
		2-3.6	4-5		10.1										
	12-13	4-27	6-24		11.5-15			2.1-2.2							4
6-10	14-16	28-62	25-69	3.1-11.1	16-55	3.2-11.4		2.3-4.4.4	7	12-12.1			10		5-
11	17	63	70	12	56	12	all	67	10	46	69	62	11	11.8	7.
	18	64-65	71-73	TP											

Internet Explorer and Safari are notable browsers without WebM support (Large preview)

However, because the `<video>` tag supports multiple `<source>` files, we can serve WebM videos to browsers that support them while falling back to MP4 everywhere else.

Let's go ahead and convert an animated GIF to both MP4 and WebM. There are several online tools that can help you do this, but many of them

use ffmpeg under the hood so we'll skip the middle man and just use that instead. `ffmpeg` is a free and open source command line tool that is designed for the processing of video and audio files. It can also be used to convert an animated GIF to video formats.
To find out if you have `ffmpeg` on your machine, fire up a terminal and run the `ffmpeg` command. This should display some diagnostic information, otherwise, you'll need to install it. Installation instructions for Windows, macOS and Linux can be found on this page. Since we'll be converting to is WebM, you need to make sure that whatever `ffmpeg` build you install is compiled with libvpx.
To follow along with the commands that are included in this article, you can use any animated GIF file lying around on your computer or grab this one which is just over 28MB. Let's begin by converting a GIF to MP4 in the next section.

CONVERT GIF TO MP4

Open up a terminal instance and navigate to the directory where the test gif is located then run the command below to convert it to an MP4 video file:

```
ffmpeg -i animated.gif video.mp4
```

This should output a new video file in the current directory after a few seconds depending on the size of the GIF file you're converting. The `-i` flag specifies the path to the input GIF file and the output file is specified afterwards (video.mp4 in this instance). Running this command on my 28MB GIF produces an MP4 file that is just 536KB in size, **a 98% reduction** in file size with roughly the same visual quality.
But we can go even further than that. `ffmpeg` has so many options that you can use to regulate the video output even further. One way is to employ an encoding method known as Constant Rate Factor (CRF) to trim the size of the MP4 output even further. Here's the command you need to run:

```
ffmpeg -i animated.gif -b:v 0 -crf 25 video.mp4
```

As you can see, there are a couple of new flags in above command compared to the previous one. `-b:v` is normally used to limit the output bitrate, but when using CRF mode, it must be set to 0. The `-crf` flag controls

the quality of the video output. It accepts a value between 0 and 51; the lower the value, the higher the video quality and file size.
Running the above command on the test GIF, trims down the video output to just 386KB with no discernable difference in quality. If you want to trim the size even further, you could increase the CRF value. Just keep in mind that higher values will lower the quality of the video file.

CONVERT GIF TO WEBM

You can convert your GIF file to WebM by running the command below in the terminal:

```
ffmpeg -i animated.gif -c vp9 -b:v 0 -crf 41 video.webm
```

This command is almost the same as the previous one, with the exception of a new `-c` flag which is used to specify the codec that should be used for this conversion. We are using the `vp9` codec which succeeds the `vp8` codec. In addition, I've adjusted the CRF value to 41 in this case since CRF values don't necessarily yield the same quality across video formats. This particular value results in a WebM file that is 16KB smaller than the MP4 with roughly the same visual quality.
Now that we know how to convert animated GIFs to video files, let's look at how we can imitate their behavior in the browser with the HTML5 `<video>` tag.

Replace Animated GIFs With Video In The Browser

Making a video act like a GIF on a webpage is not as easy as dropping the file in an `<img>` tag, but it's not so difficult either. The major qualities of animated GIFs to keep in mind are as follows:

- They play automatically
- They loop continuously
- They are silent

While you get these qualities by default with GIF files, we can cause a video file to act the exact same way using a handful of attributes. Here's how you'll embed a video file to behave like a GIF:

```
<video autoplay loop muted playsinline src="video.mp4"></video>
```

This markup instructs the browser to automatically start the video, loop it continuously, play no sound, and play inline without displaying any video controls. This gives the same experience as an animated GIF but with better performance.
To specify more that once source for a video, you can use the `<source>` element within the `<video>` tag like this:

```
<video autoplay loop muted playsinline>
    <source src="video.webm" type="video/webm">
    <source src="video.mp4" type="video/mp4">
</video>
```

This tells the browser to choose from the provided video files depending on format support. In this case, the WebM video will be downloaded and played if it's supported, otherwise the MP4 file is used instead.
To make this more robust for older browsers which do not support HTML5 video, you could add some HTML content linking to the original GIF file as a fallback.

```
<video autoplay loop muted playsinline>
    <source src="video.webm" type="video/webm">
    <source src="video.mp4" type="video/mp4">

    Your browser does not support HTML5 video.
    <a href="/animated.gif">Click here to view original GIF</a>
</video>
```

Or you could just add the GIF file directly in an `<img>` tag:

```html
<video autoplay loop muted playsinline>
    <source src="video.webm" type="video/webm">
    <source src="video.mp4" type="video/mp4">
    <img src="animated.gif">
</video>
```

Now that we've examined how to emulate animated GIFs in the browser with HTML5 video, let's consider a few potential drawbacks to doing so in the next section.

Potential Drawbacks

There are a couple of drawbacks you need to consider before adopting HTML5 video as a GIF replacement. It's clearly not as convenient as simply uploading a GIF to a page and watch it just work everywhere. You need to encode it first, and it may be difficult to implement an automated solution that works well in all scenarios.

The safest thing would be to convert each GIF manually and check the result of the output to ensure a good balance between visual quality and file size. But on large projects, this may not be practical. In that case, it may be better to look to a service like Cloudinary to do the heavy lifting for you.

Another problem is that unlike images, browsers do not preload video content. Because video files can be of any length, they're often skipped until the main thread is ready to parse their content. This could delay the loading of a video file by several hundreds of milliseconds.

Additionally, there are quite a few restrictions on autoplaying videos especially on mobile. The `muted` attribute is actually required for videos to autoplay in Chrome for Android and iOS Safari even if the video does not contain an audio track, and where autoplay is disallowed, the user will only see a blank space where the video should have been. An example is Data Saver mode in Chrome for Android where autoplaying videos will not work even if you set up everything correctly.

To account for any of these scenarios, you should consider setting a placeholder image for the video using the `poster` attribute so that the video area is still populated with meaningful content if the video does not autoplay for some reason. Also consider using the controls attribute which allows the user to initiate playback even if video autoplay is disallowed.

Wrap Up

By replacing animated GIFs with HTML5 video, we can provide awesome GIF-like experiences without the performance and quality drawbacks associated with GIF files. Doing away with animated GIFs is worth serious consideration especially if your site is GIF-heavy.