

VIDEOS IN HTML5

MMWP2024 - LV08

INHALTSVERZEICHNIS

- [Organisation](#)
- [HTML5 Video Element](#)
- [Weitere Video-Elemente](#)

INHALTSSCHWERPUNKTE

- Einführung von Videos in HTML5
- Gängige Videoformate für Webseiten
- Videostreaming im Webkontext

VORAUSSETZUNG

Der Ausgangspunkt dieser Vorlesungsreihe ist das Wissen über funktionsweise von Inhaltseinbildung in HTML5 und Mediendateien

- Verständnis von CSS-Regeln wie bei Bildern
- Aufbau von Videos und gebräuchliche Videodatentypen
- Funktionsweise von BrowserAPIs und CSS-Filtern

ZIELE

Vorstellung von:

- HTML5 Videoplayer
- Videoformate für Webseiten und Streaming
- Videos und Canvas-API
- Video- und Audio-API

GESCHICHTE DER VIDEOS IN WEBBROWSERN - 1

- Die Geschichte beginnt etwa 1996
- Zuerst waren Applikationen mit Plug-Ins im Webbrowser für die Videowiedergabe erforderlich
 - von RealNetworks: RealPlayer (1997)
 - von Apple: QuickTime 4 (1999)
 - von Microsoft: Windows Media Player
- Die Formate richteten sich nach den Playern
- Das Platzieren der Videos im Kontext war z.B. mit QuickTime unmöglich, da die Darstellung in einem Popup-Fenster erfolgte

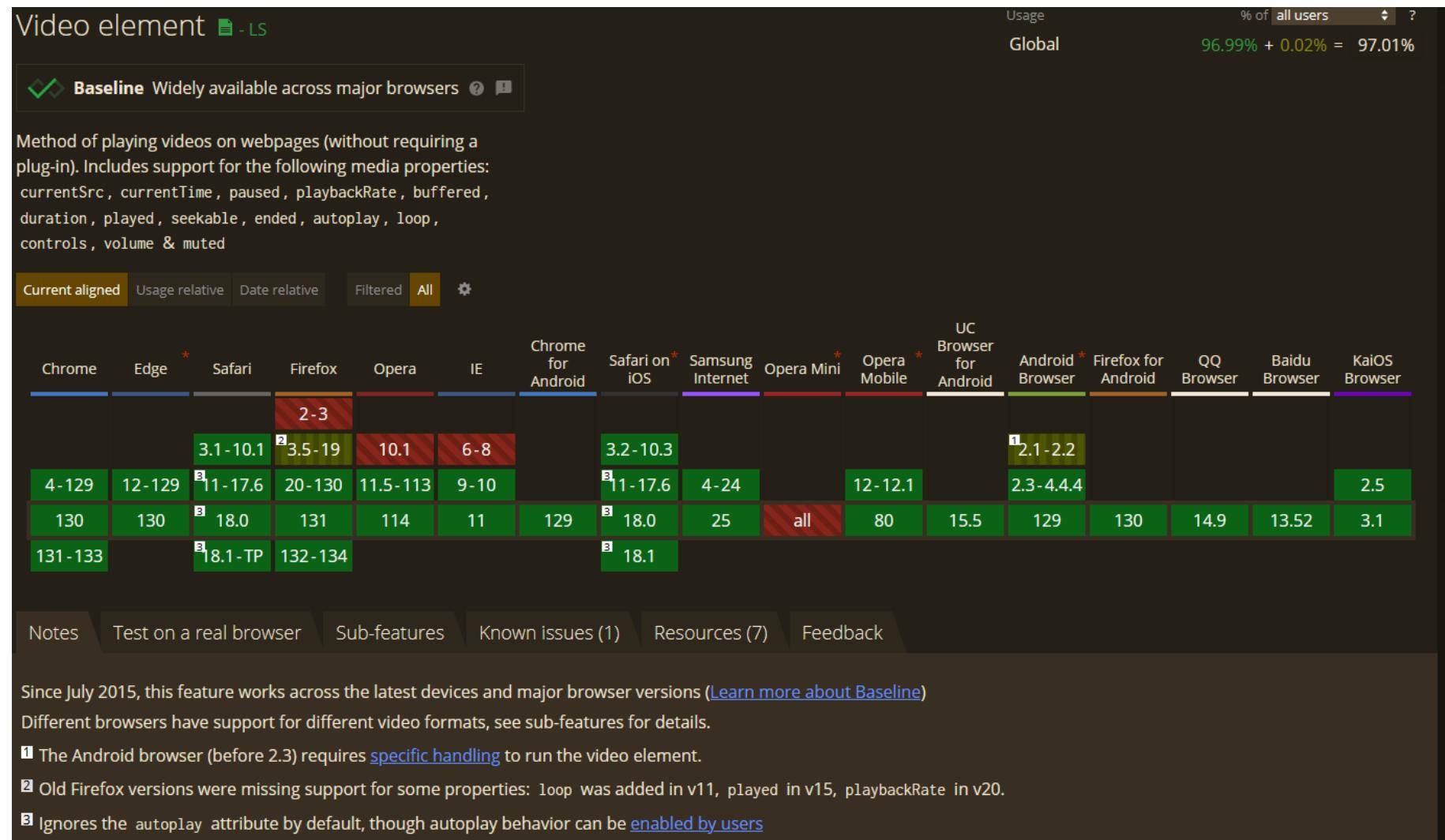
GESCHICHTE DER VIDEOS IN WEBBROWSERN - 2

- Später kam dann die Flash-Technologie von Macromedia / Adobe für Webbrowser, die ursprünglich von Macromedia (Markenname Shockwave) vorangetrieben wurde
- Hier waren ab 1997 der Shockwave-Player oder ab 2005 der Adobe Flash Player die angesagten Plugins
- Formate waren SWF (mit Ziffern für die Formatversionen hinten dran, seit 1997), später FLV als Streamingformat (seit 2002)

GESCHICHTE DER VIDEOS IN WEBBROWSERN - 3

- 2010 kündigte Apple an, den Flash Player auf Apple-Produkten nicht mehr native anzubieten
- Ende 2011 wurde dann die Weiterentwicklung von Flash für mobile Geräte eingestellt
- Es gibt den Flash Player meist noch als App zum Nachinstallieren, aber in veralteter Version (Problematisch für Systemsicherheit)
- Das Element <video> wurde 2007 von Opera für HTML5 vorgeschlagen
- Inzwischen wird es von allen aktuellen Webbrowsern grundlegend unterstützt

VIDEO - SUPPORT

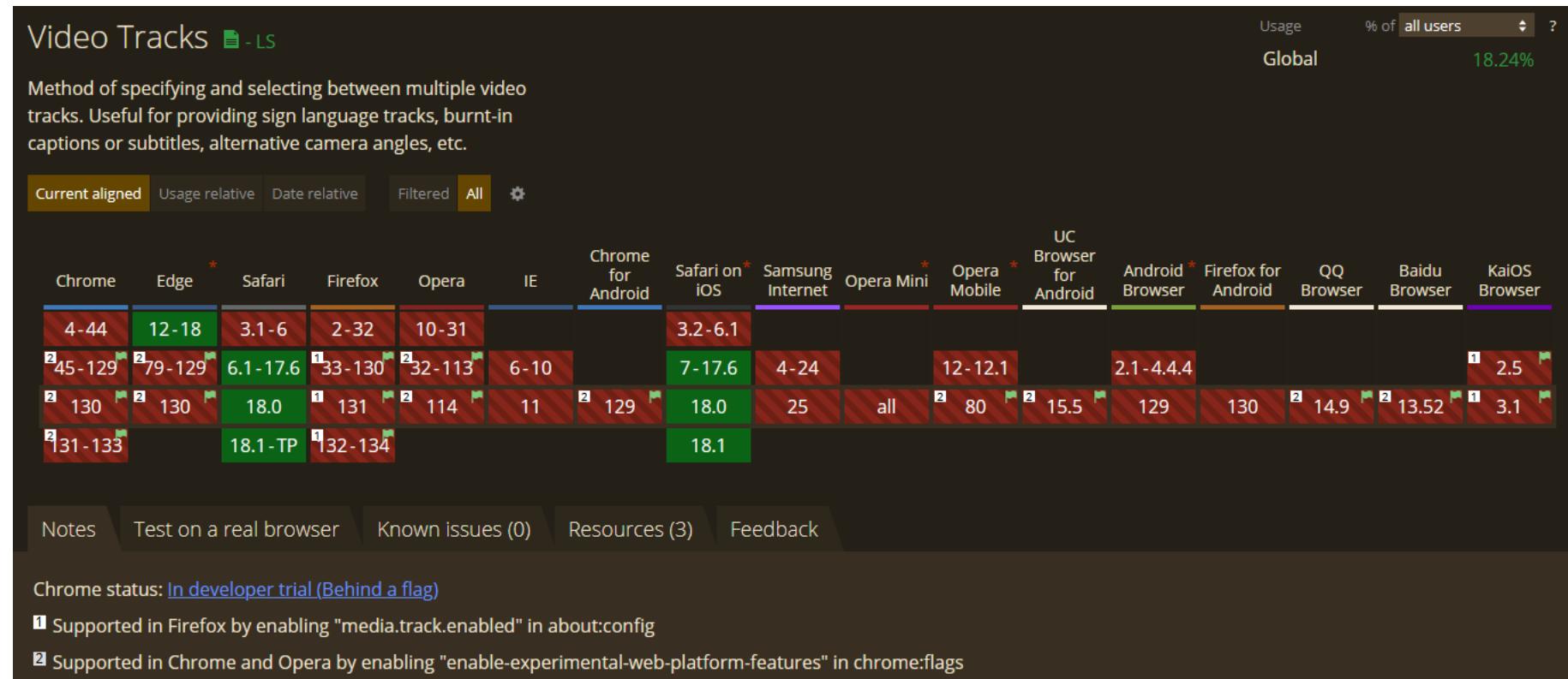


Quelle

BESONDERHEIT VON VIDEO- UND AUDIO-ELEMENTEN

- Die meisten Browser nutzen vorhandene Medienwiedergabe-Frameworks zur Dekodierung und Wiedergabe
- Dadurch ergeben sich verschiedene Kombinationen an Formaten und Codecs - je nach Browser, Betriebssystem und installierten Codecs
- Videodateien in verschiedenen Formaten durch mehrere `<source>`-Elemente, oder durch JavaScript unterstützte Formate abfragen
- Neu: Videotracks welche zurzeit noch in Arbeit sind

VIDEO - TRACK SUPPORT



Quelle

TEST AUF <VIDEO>-UNTERSTÜTZUNG - 1

- Den Test auf Unterstützung des Elementes <video> kann man in JavaScript dynamisch t tigen
- ```
var videoSupport = !!(document.createElement('video').canPlayType);
```
- Durch die doppelte Verneinung wird die Variable boolesch und kann ausgewertet werden
- Auch modernizr.js kann eingesetzt werden (siehe [GitHub](#))

## TEST AUF <VIDEO>-UNTERSTÜTZUNG - 2

- Alternativ kann eine genauere Abfrage erfolgen
- ```
let x = document.createElement("VIDEO");
isSupp = x.canPlayType(vidType+';codecs="'+codType+'"');
```
- "probably" - the browser most likely supports this video type
- "maybe" - the browser might support this video type
- "" - (empty string) the browser does not support this video type
- [Can I play](#)

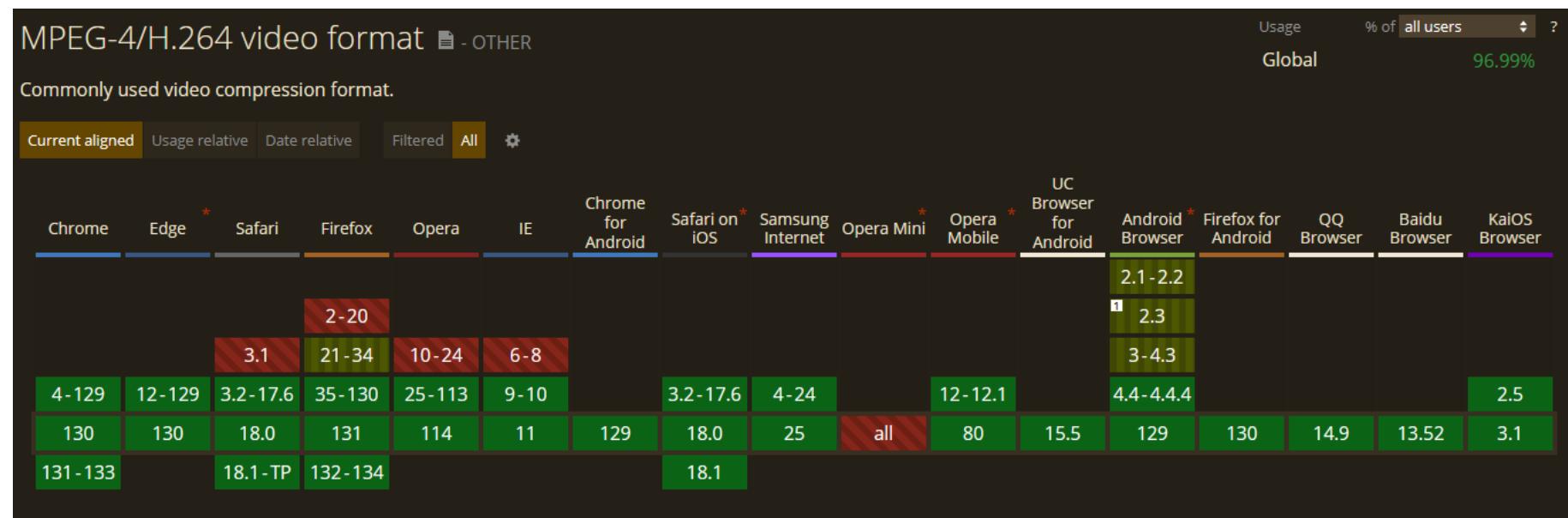
TEST AUF VIDEOFORMATE

```
let videoSupportOgg = !!(document.createElement('video').canPlayType('video/ogg') && document.createElement('video').canPlayType('video/mp4'))  
let videoSupportH264 = !!(document.createElement('video').canPlayType('video/mp4') && document.createElement('video').canPlayType('video/ogg'))  
var videoSupportWebM = !!(document.createElement('video').canPlayType('video/webm') && document.createElement('video').canPlayType('video/ogg'))  
  
codecs="vp8,vorbis").replace(/^no$/,'')));
```

FORMAT MP4

- MP4, bzw. MPEG-4 Teil 14, ist die Weiterentwicklung des Quicktime-Formats. Container-Format
- Videospur: H.264 / Audiospur: AAC
- Beide Codecs sind patentrechtlich geschützt und Lizenzgebührenpflichtig
- MP4 unterstützt DRM-Mechaniken (siehe [Unified streaming](#))
- Es gibt eine weitverbreitete Hardwarebeschleunigung für den H.264-Codec
- X.264 ist der OpenSource-Standard, wird aber praktisch nicht genutzt

H.264 - SUPPORT



Quelle

FORMAT OGG

- Ogg ist ein Containerformat, das lizenzfrei ist
- Videospur: Theora, Audiospur: Vorbis oder Opus
- Vorteil von Ogg ist, das es gestreamt werden kann
- MP4 unterstützt DRM-Mechaniken (siehe [Verschlüsselung](#))
- Es gibt eine weitverbreitete Hardwarebeschleunigung für den H.264-Codec
- X.264 ist der OpenSource-Standard und auch verbreitet

OGG VORBIS - SUPPORT

Device	OS	Version	Video Codecs				AES scheme		DRM/Max Security Level		
			AVC	HEVC	VP9	AV1	cbcS	cenc	Widevine	PlayReady	FairPlay
Chrome	Win/MacOS	>=68	YES		YES	YES	YES	YES	YES/L3		
Safari	MacOS	>=11	YES	YES	YES		YES				YES
Firefox	Win/MacOS	>=60	YES		YES	YES	YES	YES	YES/L3		
Edge (Chromium)	Win	88	YES	YES	YES	(YES)	YES	YES	YES/L3	(YES/SL3000)	
Edge (Chromium)	MacOS	88	YES		YES		YES	YES	YES/L3		
iPhone/AppleTV	iOS/tvOS	>=10	YES	YES	YES		YES				YES
Safari	iOS/ipadOS	>=10	YES	YES			YES				YES
Chrome	Android	>=7.1	YES		YES		YES	YES	YES/L2		
Mobile	Android	>=7.1	YES	YES	YES		YES	YES	YES/L1		
Sony/Philips	Android TV		YES	YES	YES		YES	YES	YES/L1	YES	
Magenta TV Stick	Android TV	9	YES	YES	YES		YES	YES	YES/L1	YES	
Fire TV	FireOS	>=6 (Android 7.1)	YES	YES	YES		YES	YES	YES/L1	YES/SL3000	
Samsung	Tizen	>=5.5 (2020)	YES	YES	YES	YES	YES	YES	YES/L1	YES/SL3000	
LG	WebOS TV	>=5.0 (2020)	YES	YES	YES	YES	YES	YES	YES/L1	YES/SL3000	
Chromecast Ultra / G	Android TV		YES	YES	YES		YES	YES	YES/L1	YES/SL2000	

Quelle

OGG THEORA - SUPPORT

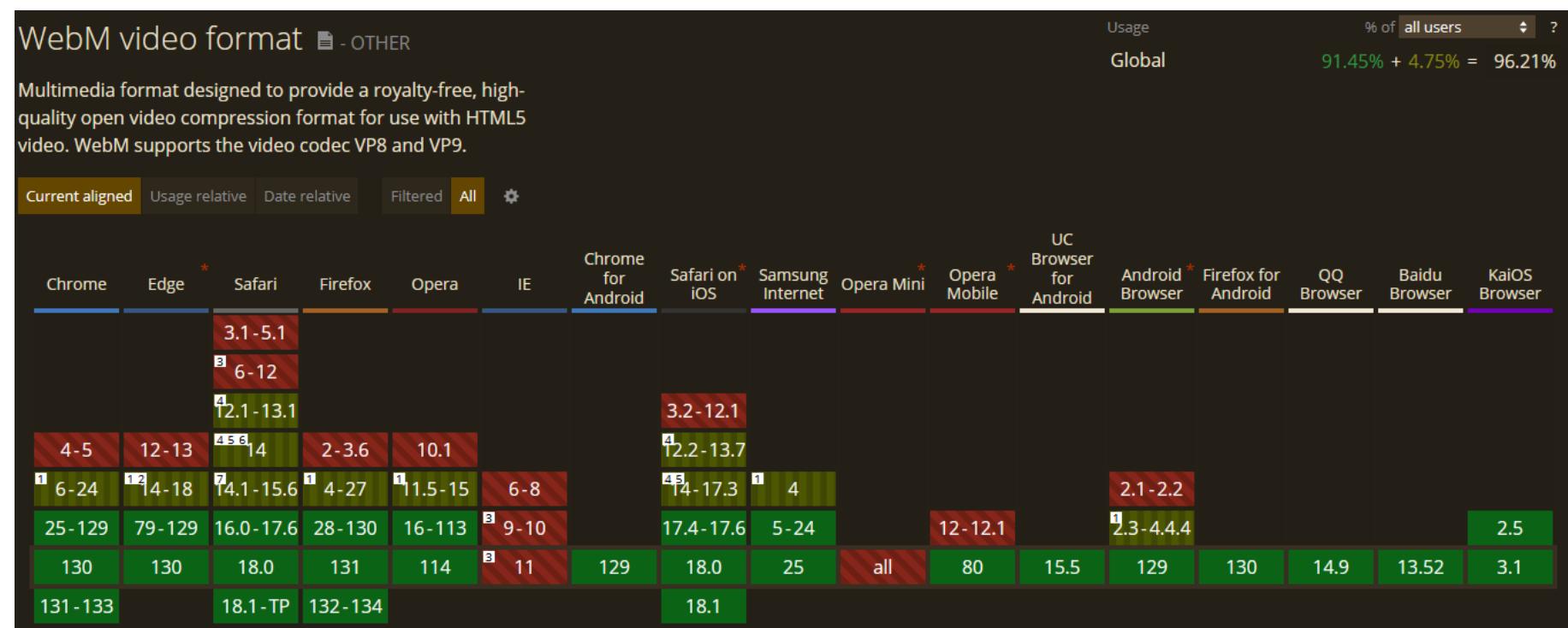
Device	OS	Version	Video Codecs				AES scheme		DRM/Max Security Level		
			AVC	HEVC	VP9	AV1	cbcS	cenc	Widevine	PlayReady	FairPlay
Chrome	Win/MacOS	>=68	YES		YES	YES	YES	YES	YES/L3		
Safari	MacOS	>=11	YES	YES	YES		YES				YES
Firefox	Win/MacOS	>=60	YES		YES	YES	YES	YES	YES/L3		
Edge (Chromium)	Win	88	YES	YES	YES	(YES)	YES	YES	YES/L3	(YES/SL3000)	
Edge (Chromium)	MacOS	88	YES		YES		YES	YES	YES/L3		
iPhone/AppleTV	iOS/tvOS	>=10	YES	YES	YES		YES				YES
Safari	iOS/ipadOS	>=10	YES	YES			YES				YES
Chrome	Android	>=7.1	YES		YES		YES	YES	YES/L2		
Mobile	Android	>=7.1	YES	YES	YES		YES	YES	YES/L1		
Sony/Philips	Android TV		YES	YES	YES		YES	YES	YES/L1	YES	
Magenta TV Stick	Android TV	9	YES	YES	YES		YES	YES	YES/L1	YES	
Fire TV	FireOS	>=6 (Android 7.1)	YES	YES	YES		YES	YES	YES/L1	YES/SL3000	
Samsung	Tizen	>=5.5 (2020)	YES	YES	YES	YES	YES	YES	YES/L1	YES/SL3000	
LG	WebOS TV	>=5.0 (2020)	YES	YES	YES	YES	YES	YES	YES/L1	YES/SL3000	
Chromecast Ultra / G	Android TV		YES	YES	YES		YES	YES	YES/L1	YES/SL2000	

Quelle

FORMAT WEBM

- WebM ist ein Videoformat, das u.a. von Google, Mozilla und Opera als freies Format stetig weiterentwickelt wird
- Erklärtes Ziel ist die Patentfreiheit
- WebM ist ein Containerformat mit P8, VP9, AV1 als Video-Codec und Vorbis, Opus als Audio codec
- Unterstützung von VP9 seit Chrome 25, Firefox 29 und Opera 16
- Ein Grund für die weite Verbreitung von WebM ist, dass Youtube seine Videos seit Dezember 2013 beginnend im Format VP9 anbot

FORMAT WEBM - SUPPORT



Quelle

DATEIDATEN FÜR YOUTUBE - 1

- Upload-Empfehlungen von Youtube am 17.05.2019
- [Supported YouTube file formats](#)
- [YouTube recommended upload encoding settings](#)
- Container: MP4 und "moov"-Atom am Anfang der Datei (Schnellstart)
- Audio-Codec: AAC-LC (Kanäle: Stereo oder Stereo + 5.1, Abtastrate: 96 kHz oder 48 kHz)

DATEIDATEN FÜR YOUTUBE - 2

- Video-Codec: H.264
- Progressiver Scan (kein Zeilensprungverfahren) zusammen mit Profil "High"
- [YouTube recommended upload encoding settings](#)
- Container: MP4 und "moov"-Atom am Anfang der Datei (Schnellstart)
- Audio-Codec: AAC-LC (Kanäle: Stereo oder Stereo + 5.1, Abtastrate: 96 kHz oder 48 kHz)

FORMAT H.265 BZW. HEVC

- In Konkurrenz zu VP9 und VP10 von Google entwickelte die MPEG-Gruppe 2011-2013 das Format H.265 als Nachfolger von H. 264
- Offizielle Bezeichnung: MPEG-H Teil 2
- ITU-Standard seit 2013 (Lizenzpflichtig!)
- Neben der Ermöglichung effektiver Internetübertragungen können höhere Auflösungen von Videos bis 8K UHD effektiv kodiert werden

H.265 - SUPPORT



Quelle

VIDEO TESTS

[Html5 video Testseite](#)

PLAYER UND PLUG-INS

- Google Chrome: Plug-In libde265 (X.265) (in Versionen) [libde265](#)
- Alle Browser: DivX-Plug-In [hevc](#)
- Die Lizenzgebühren sind ein ständiges Problem für Browserproduzenten

ALLIANCE FOR OPEN MEDIA (AOMEDIA)

- 2015 wurde die Alliance for Open Media gegründet, um für das Internet ein garantiert lizenzgebührenfreies Videoformat zu finden
- Die Alliance for Open Media wird von den Gründerfirmen geleitet: Amazon, ARM, Cisco, Google, Intel, Microsoft, Mozilla, Netflix and NVIDIA
- Die Lizenzgebühren sind ein ständiges Problem für Browserproduzenten
- AO Media

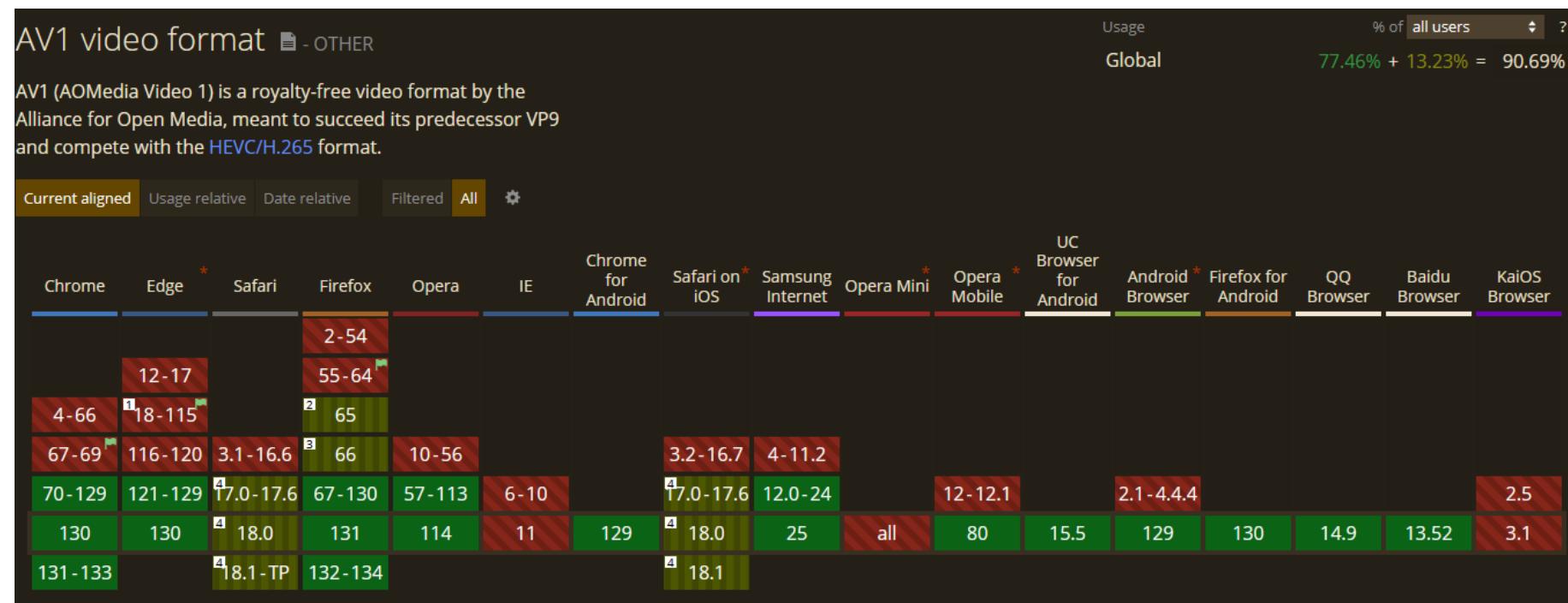
AOMEDIA VIDEO 1 (AV1)

- Es ist ein offener, lizenzfreier Video-Codec. Er ist für die Video-Übertragung über das Internet konstruiert
- AV1 ist als der Nachfolger von VP9 und als Konkurrent zu HEVC/H.265 aufgestellt
- Er wurde am 28. März 2018 veröffentlicht
- AV1 soll mit Opus zusammen in WebM für HTML5-Video genutzt werden können
- Es gibt Konkurrenten: NetVC

VIDEOOKONVERTER FÜR AV1

- ffmpeg ([FFMPEG](#))
- Miro Video Converter ([Miro Video](#))
- VLC media player ([VLC Player](#))
- Handbrake: [Handbrake](#)

AV1 - SUPPORT



Quelle

VIDEO EINBINDEN - SYNTAX - 1

- Videos lassen sich einfach mittels <video>-Elements einbinden
- Optional kann man sie durch viele Attribute, mögliche Unterelemente, JavaScript-Methoden und Ereignisse für den Ereignismanager genauer konfigurieren

VIDEO EINBINDEN - SYNTAX - 2

```
// Das Bild zur Vorschau wird beim Laden angezeigt.  
<video src="videofile.mp4" poster="vorschau.jpg" controls>  
Nachricht, falls der Browser das Videoelement nicht  
unterst&uuml;tzt.  
</video>
```

VIDEO EINBINDEN - SYNTAX - 3

Attribut	Bedeutung
autoplay	Sofort spielen nach dem Laden (nicht für mobile Geräte).
height	Höhe des Videoplayers in Pixeln.
width	Breite des Videoplayers in Pixeln.
loop	Endlosschleife des Videos = ein.
muted	Video ohne Ton abspielen.
preload	auto (= default) – Video vorladen, metadata – nur Metadaten vorladen, none – nichts vorladen.

VIDEO EINBINDEN - SYNTAX - 4

```
<video controls>
  <source src="videofile.webm" type="video/webm" />
  <source src="videofile.ogg" type="video/ogg" />
  <source src="videofile.mp4" type="video/mp4" />
  The <code>video</code> element is not supported.
</video>
```

- Der Browser wählt automatisch das Format aus, welches er am besten unterstützt
- Zusätzlich kann noch ein Attribut media hinzugefügt werden, welches das Zielmedium angibt
- Es gibt 30 JavaScript-Eigenschaften der Media JavaScript API, 26 Ereignisse, 5 Methoden

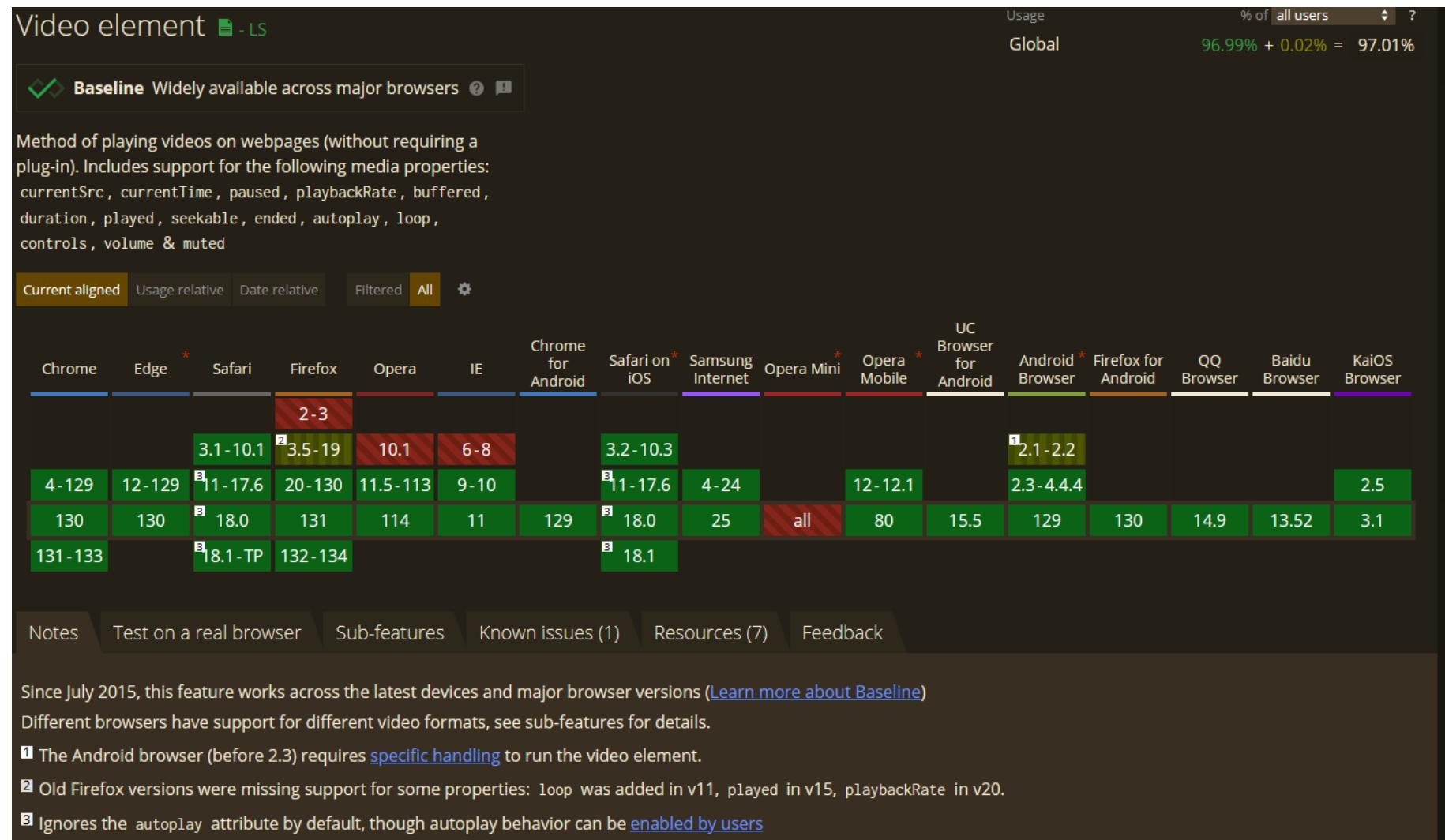
TRACKS EINBINDEN - SYNTAX - 1

- Mit dem Unterelement <track> können Audio- und Video-Elementen Untertitel, Beschreibungen für die Ausgabe per Screenreader, Szenenüberschriften etc. zum Teil mehrsprachig mitgegeben werden
- ```
<track src="de_subtitles.vtt" srclang="de" kind="subtitles" label="German" />
```
- Zusätzlich gibt es noch das Attribut default für <track>-Informationen, die zuerst genommen werden sollen

## TRACKS EINBINDEN - SYNTAX - 2

Werte für das Attribut kind	Bedeutung
subtitles	Untertitel z.B. zur Handlung, zu Dialogen
captions	Untertitel für Soundeffekte, Audiohinweise
chapters	Kapitelüberschriften, nur zur Navigation
metadata	nicht angezeigter Text zur Nutzung mit JavaScript
descriptions	Beschreibung des Videoinhalts in Textform

# VIDEO - SUPPORT



Quelle

# TRACKS - SUPPORT



## VTT – VIDEO TEXT TRACKS

- Aufbau einer WebVTT-Datei
  - Mit Kopfzeile „WEBVTT“ evtl. mit Titel
  - Eine Leerzeile
  - durch Leerzeilen getrennte cues [Zeiger der Art: VTTcue( startTime, endTime, Text )]
- Dateiendung: \*.vtt
- VTT-Dateien können aber auch sehr komplexe Syntax enthalten
- [Mozilla Video Tracks](#)
- [WebBTT](#)

## VTT – VIDEO TEXT TRACKS

WEBVTT – Beispiel

14

00:01:14.815 --> 00:01:18.114

What? - Where are we now?

15

00:01:18.171 --> 00:01:20.991

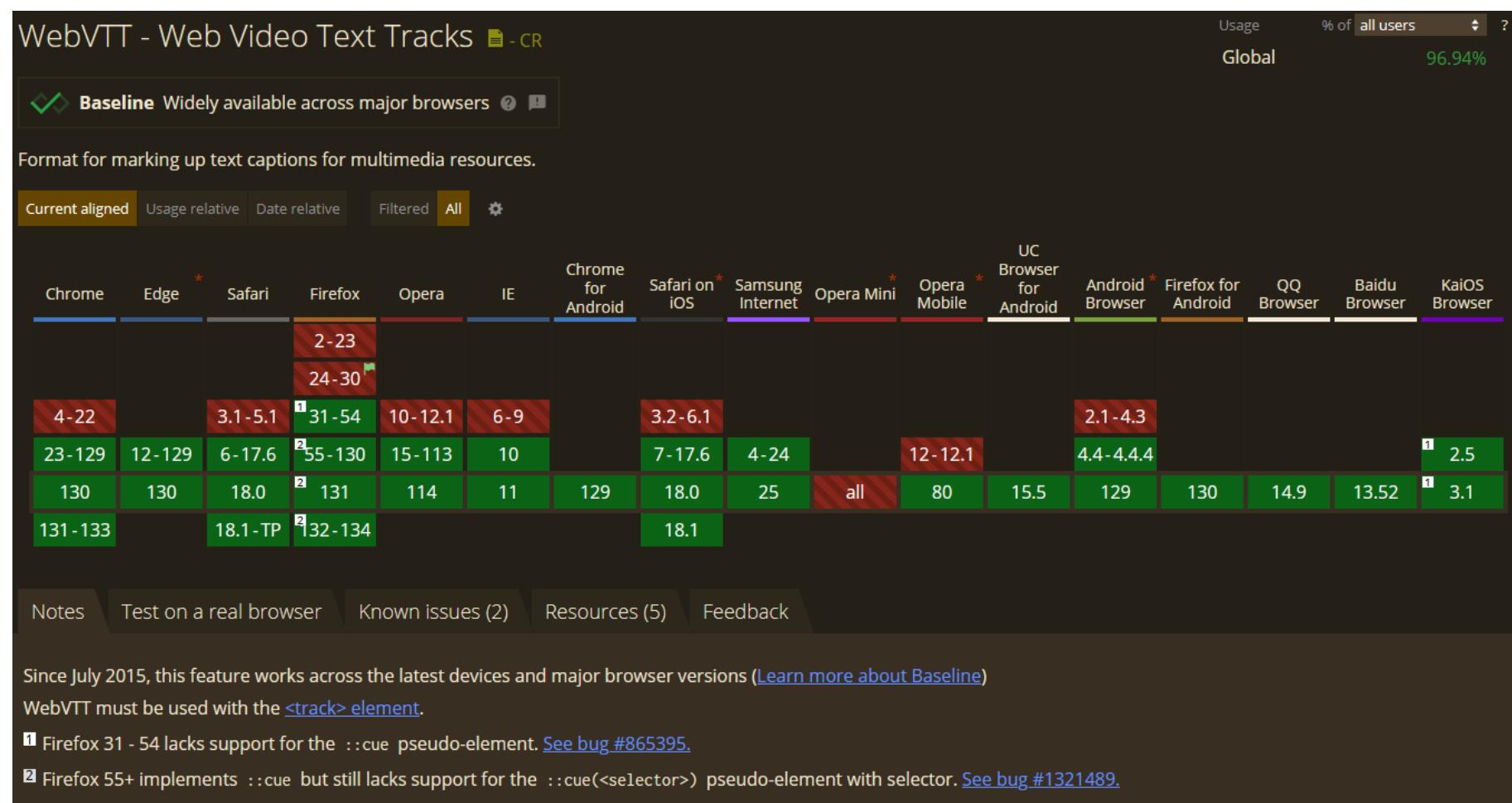
This is big bat country.

16

00:01:21.058 --> 00:01:23.868

[ Bats Screeching ] - They won't get in your hair. They're after the bugs.

# TRACKS - SUPPORT



Quelle

## VIDEOS - FALLBACKS - 1

Wenn keine der angebotenen Videovarianten über das Element <video> greift, muss man Fallback-Lösungen anbieten, z.B.:

```
<video controls>
 <source src="videofile.mp4" type="video/mp4">
 </source> <!-- weitere Quellen -->

 <p>Click image to play a video demo.</p>
 </video>
```

## VIDEOS - FALLBACKS - 2

Fallback tricks

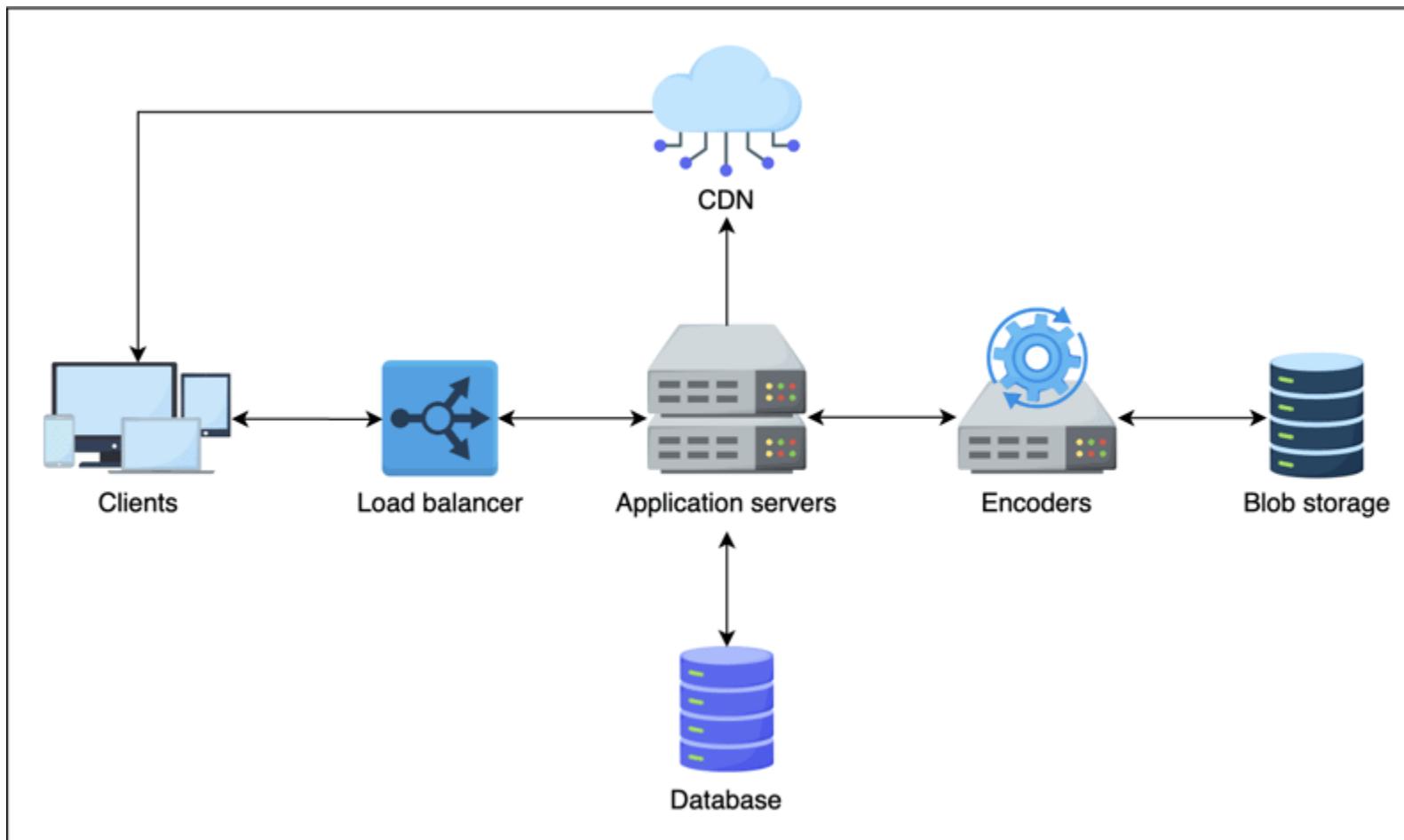
## AUFNAHME VON MEDIEN

- Seit 2009 wird an einem Standard zum Aufnehmen von Bild, Video und Audio aus Webseiten heraus gearbeitet
- Ein Tutorial mit Demos: [User media capture](#)
- WebRTC demos und samples: [WebRTC Beispiele](#)
- Devices and Sensors Working Group: [Device and Sensors Working Group](#)

## VERSCHIEDENE STREAMS API

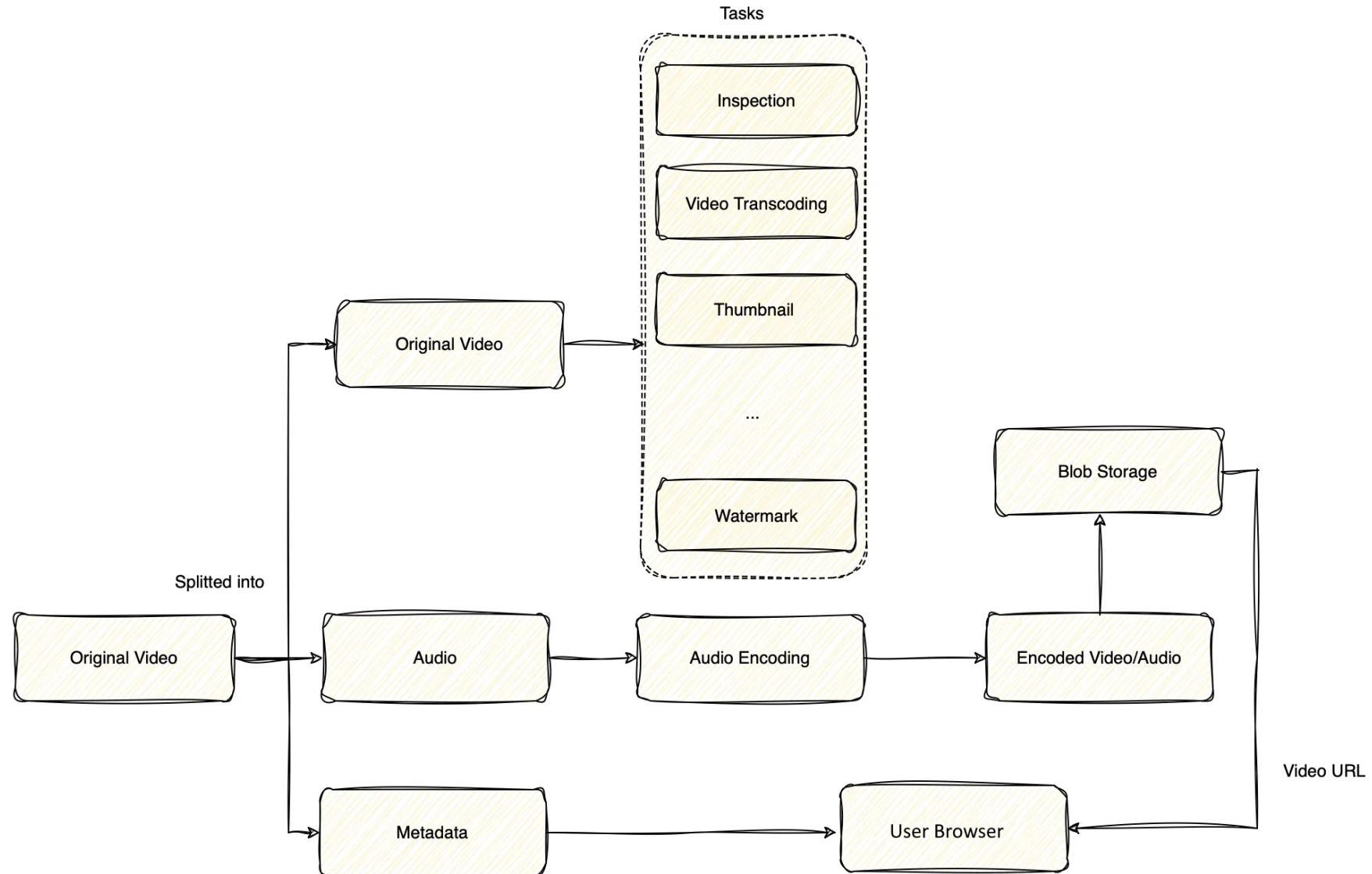
- Vorgängertechnologie (deprecated):  [GetUserMedia](#)
- Aktuelle Anleitung:  [GetUserMedia \(neu\)](#)
- W3C vom 03.10.2017:  [Quelle](#)
- Die Implementierung ist browserabhängig und wird von gängigen Browsern vielseitig unterstützt

# VIDEOSTREAMING - TOPOLOGIE



Quelle

# AUDI-VIDEOSTREAMING - TOPOLOGIE



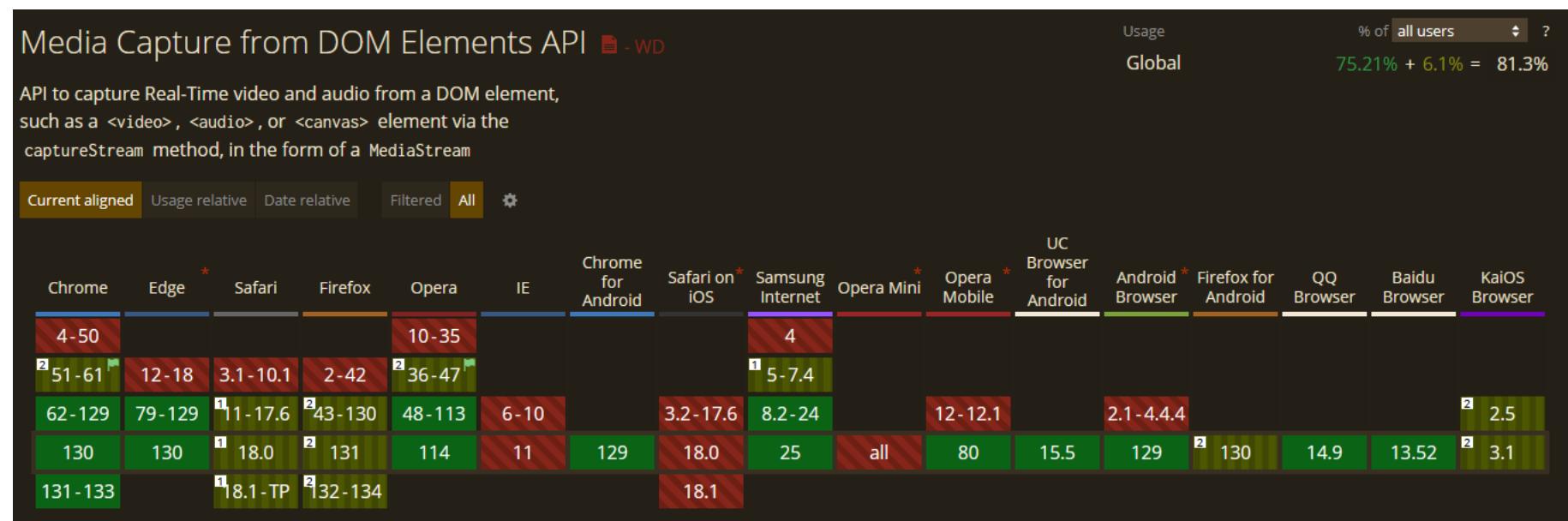
# Quelle

# STREAM API



Quelle

# STREAM API - SUPPORT



# Quelle

## AUDIO UND VIDEO MIT CANVAS API

- Der Durchbruch der Einbettung und Veränderung von Video und Audio in der Canvas API war die Idee des Zugriffes auf den Frame-Buffer einmal geladener Dateien
- [Beispiel Canvas Video](#)
- [Manipulating video using canvas](#)

## WEITERE AUDIO UND VIDEO BEISPIELE

- Open source HTML5 player framework
- jQuery HTML5 Audio / Video Library
- W3Schools - HTML videos
- W3Schools - YouTube videos

## VIDEOS ALS HINTERGRUND

- [W3Schools fullscreen video](#)
- [Fullscreen video](#)
- [CSS-Tricks Background videos](#)

## FILTER AUF BILDER UND VIDEOS - 1

- [Filter function list](#)
- [Mozilla Filter Guide](#)
- [CSS-Tricks Bilder](#)

## FILTER AUF BILDER UND VIDEOS - 2

```
.blur {
 filter: blur(4px);
}

.brightness {
 filter: brightness(0.30);
}

.contrast {
 filter: contrast(180%);
}

.grayscale {
 filter: grayscale(100%);
}
```

## BACKDROP FILTER - 1

- [W3Schools Backdrop Filter](#)
- [Mozilla Beispiele](#)
- [CSS-Tricks Filter](#)

# BACKDROP FILTER VS FILTER



Backdrop Filters



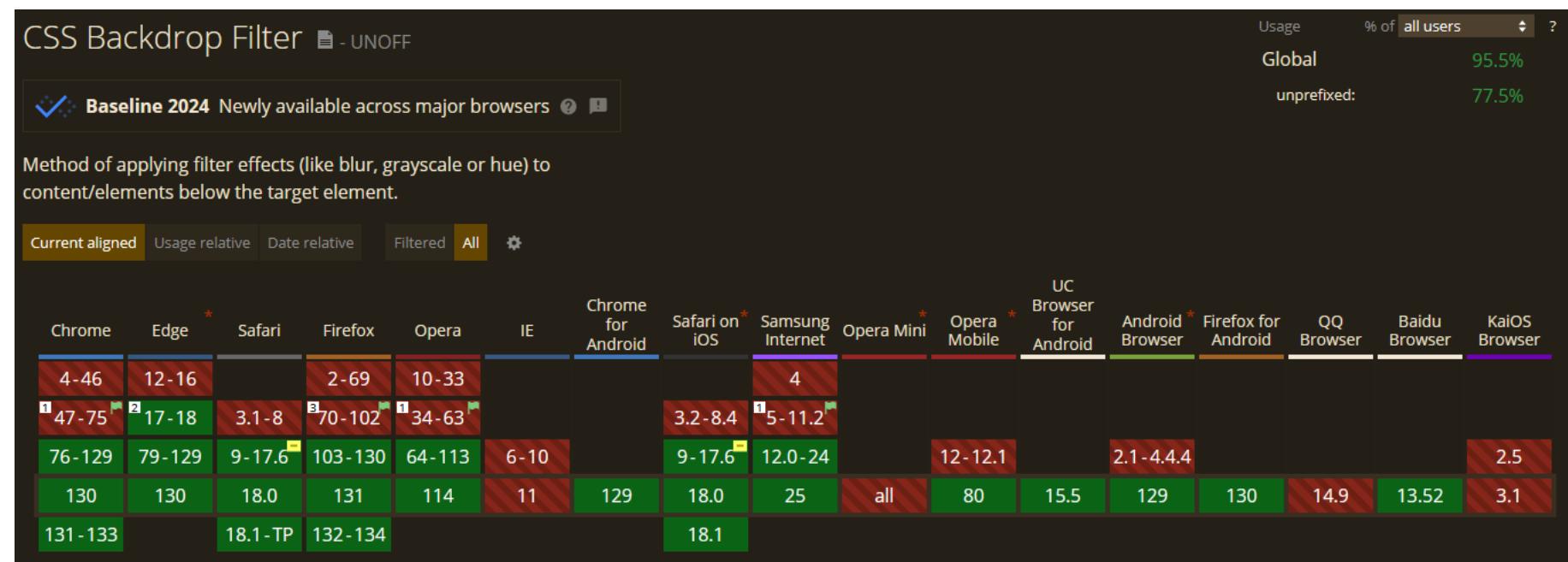
Filter Beispiel

# FILTER - SUPPORT



## CSS Filter

# BACKDROP FILTER - SUPPORT



CSS Backdrop Filter

## VIDEOS UND FILTERANWENDUNGEN

- Durch Canvas, Filter und WebStreaming können Videos im Browser bearbeitet werden
- Z.B. [Etrojs.dev](#)

# NETFLIX - ERSTER MEDIASERVER 262TB HDD CACHE



Quelle

# NETFLIX - AUDIO-VIDEO-STREAMING SERVER - 1



Quelle

# NETFLIX - AUDIO-VIDEO-STREAMING SERVER - 2



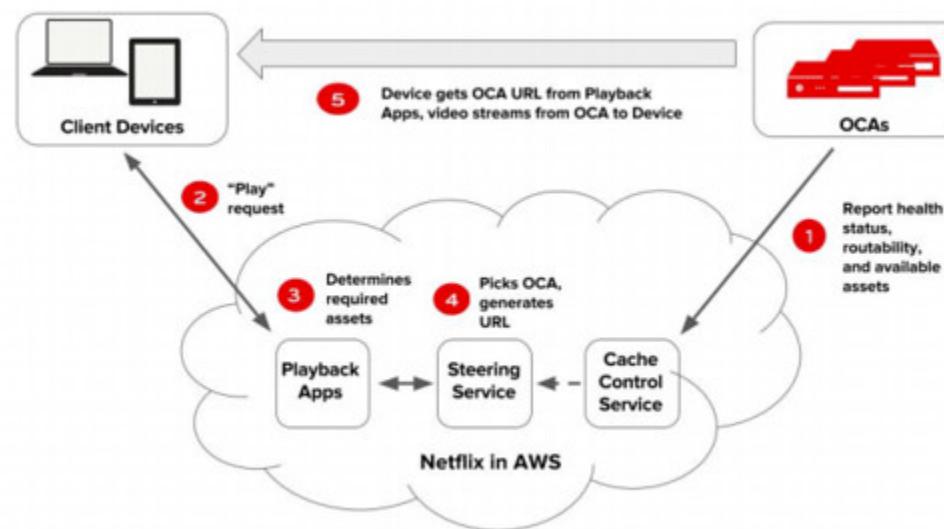
Quelle

## NETFLIX - AUDIO-VIDEO-STREAMING SERVER - 3



Quelle

# NETFLIX - AUDIO-VIDEO-STREAMING SERVER - 4



Quelle

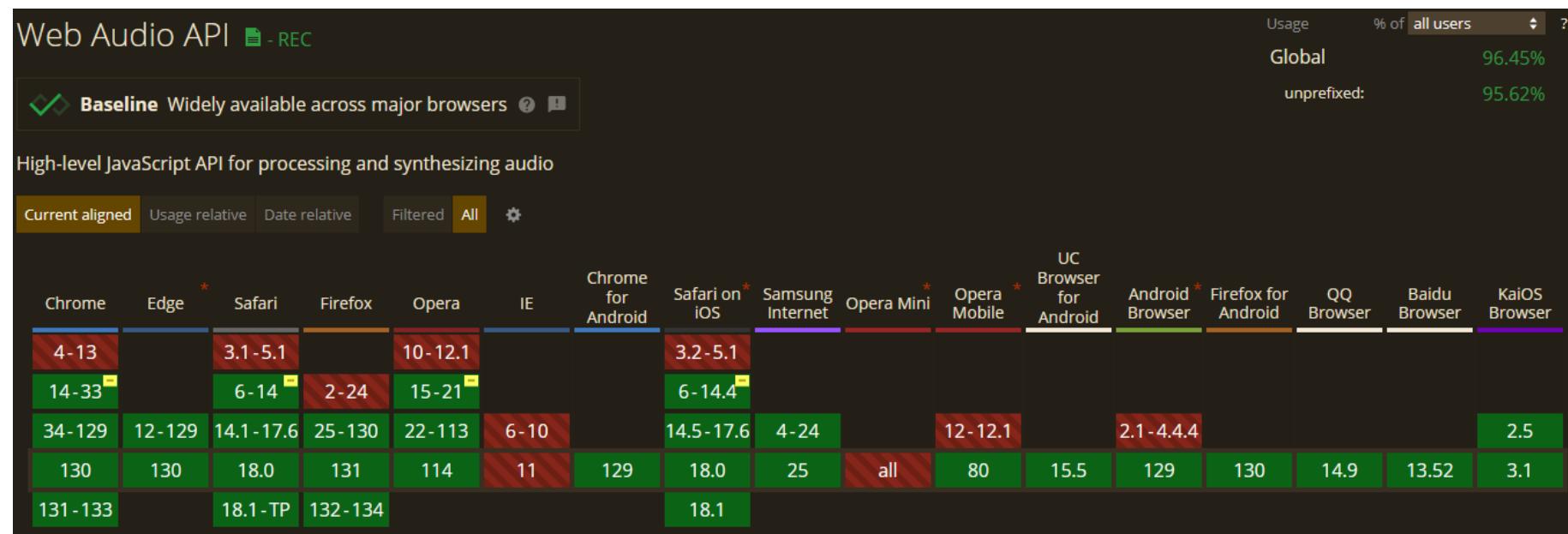
## WEB AUDIO API

- Für Audio-Anwendungen gibt es das umfangreiche Web Audio API des W3C
- [Web Audio API](#)
- Empfehlung: [W3 Webaudio](#)
- [Mozilla Web audio API](#)
- [Google Web audio API](#)
- [Web audio API Beispiel](#)

## WEB AUDIO API - WEITERE BEISPIELE

- [Web Audio Demos](#)
- [Building a synthesizer](#)
- [Web Audio API](#)
- [Quelle](#)

# WEB AUDIO API - SUPPORT



Quelle

## WEB AUDIO API - FILTER

[BiquadFilterNode](#)

## QUELLEN - 1

- Mark Kotsarev, „Analyse der HTML5-Videofunktion und deren Anwendung an einem praxisorientierten Beispiel“, Bachelorarbeit, Juni 2014
- Tutorial: [Quelle](#)
- Medienformate, die Browser akzeptieren: [Quelle](#)
- Ereignisse und Methoden des HTMLMediaElement: [Quelle](#)
- Media Events im Kontext von <audio> und <video>: [Quelle](#)

## QUELLEN - 2

- Erfinder der Technik, den Video-Buffer der Canvas API auszunutzen, um Kopien eines Videos effektiv zu rendern: [Quelle](#)
- Experiment von Jekyll: [Quelle](#)
- Anwendung zahlreicher Filter auf Videos, von Jekyll: [Quelle](#)
- Experiment von Jekyll: [Quelle](#)
- Damian Brock, Bachelorarbeit zum Web Audio API, HTWK, April 2015

## ABSPANN

Achtes Level geschafft fünf weitere Folgen!

Fragen und Feedback?