

# Saving on energy in a web video environment

Created May 11, 2022  
Updated June 15, 2023



RADIANT  
[MEDIA]  
PLAYER

# Can we measure energy consumption in a browser or WebView?

- Short answer: yes and no (mid-2022)
- Long answer: Battery Status API (JavaScript)
  - Available for mobile and portable devices
  - Changes in battery status: chargingchange & levelchange
  - Battery level in %
  - Charging time and discharging time in seconds
  - Available in Chrome & Edge but NOT in Firefox or Safari

We can get % level data in some environments but no translation to mAh or Watts

# How to optimise energy consumption for an HTML5 video player then?

- Media Capabilities API (JavaScript) to the rescue
- Supported in all modern browsers and WebViews
- Supports many input settings (framerate, resolution, bitrate)
- Gives information about
  - Supported configuration
  - Smooth playback
  - Power efficient

Given several codecs are available in a HLS or MPEG-DASH manifest we can make a decision on which one to use

# Media Capabilities return values

- supported: yes or no for the input configuration (may slightly vary from canPlayType and isTypeSupported)
- smooth: the device evaluate the capacities to playback content without dropped frames
- powerEfficient: is hardware or software decoding available?

Hardware decoders outperform software decoders in power efficiency and decoding speed, but software decoders can remain efficient for low-resolutions

```
1  if ('mediaCapabilities' in navigator) {
2    //Create a video configuration to be tested
3    const videoDecoderConfig = {
4      type: 'file', // 'record', 'transmission', or 'media-source'
5      video: {
6        contentType: 'video/webm;codecs=vp8', // valid content type
7        width: 800, // width of the video
8        height: 600, // height of the video
9        bitrate: 10000, // number of bits used to encode 1s of video
10       framerate: 30 // number of frames making up that 1s.
11     }
12   };
13
14   const audioEncoderConfig = {
15     type: 'file', // 'record', 'transmission', or 'media-source'
16     audio: {
17       contentType: 'audio/ogg', // valid content type
18       channels: 2, // audio channels used by the track
19       bitrate: 132700, // number of bits used to encode 1s of audio
20       samplerate: 5200 // number of audio samples making up that 1s.
21     }
22   };
23
24   navigator.mediaCapabilities.decodingInfo(videoDecoderConfig).then(result => {
25     console.log('This configuration is ' +
26       (result.supported ? '' : 'not ') + 'supported, ' +
27       (result.smooth ? '' : 'not ') + 'smooth, and ' +
28       (result.powerEfficient ? '' : 'not ') + 'power efficient.')
29   }).catch(() => {
30     console.log('decodingInfo error: ' + contentType)
31   });
32 }
33
```

# Testing results - configuration

```
const mediaConfig = {  
  type: 'media-source',  
  audio: {  
    contentType: codec.audioOnly,  
    channels: 2,  
    bitrate: 128000,  
    samplerate: 48000  
  },  
  video: {  
    contentType: codec.videoOnly,  
    width: 1920,  
    height: 1080,  
    bitrate: 5000000,  
    framerate: 30  
  };  
};
```

# Testing results - Desktop

- Chrome 101 on Windows 11
  - AVC & AAC-LC audio: supported, smooth and power efficient
  - HEVC & HEv2-AAC audio: **NOT supported**
  - VP8 & vorbis audio: supported, smooth and **NOT power efficient**
  - VP9 & opus audio: supported, smooth and power efficient
  - AV1 & HEv2-AAC audio: supported, smooth and **NOT power efficient**
  - >> Best energy/quality ratio => VP9 with opus audio**
- Safari 15.4 on macOS 12.3.1
  - AVC & AAC-LC audio: supported, smooth and power efficient
  - HEVC & HEv2-AAC audio: supported, smooth and power efficient
  - VP8 & vorbis audio: **NOT supported**
  - VP9 & opus audio: supported, smooth and power efficient
  - AV1 & HEv2-AAC audio: **NOT supported**
  - >> Best energy/quality ratio => HEVC & HEv2-AAC audio**

# Testing results - Mobile

- Chrome 101 for Android 11
  - AVC & AAC-LC audio: supported, smooth and power efficient
  - HEVC & HEv2-AAC audio: **NOT supported**
  - VP8 & vorbis audio: supported, smooth and power efficient
  - VP9 & opus audio: supported, smooth and power efficient
  - AV1 & HEv2-AAC audio: supported, smooth and power efficient
  - >> Best energy/quality ratio => AV1 & HEv2-AAC audio**
- Safari for iOS 15.4
  - AVC & AAC-LC audio: supported, smooth and power efficient
  - HEVC & HEv2-AAC audio: supported, **NOT smooth** and power efficient
  - VP8 & vorbis audio: **NOT supported**
  - VP9 & opus audio: supported, smooth and power efficient
  - AV1 & HEv2-AAC audio: **NOT supported**
  - >> Best energy/quality ratio => VP9 & opus audio**



# Test Media Capabilities on your device

Go to <https://www.radiantmediaplayer.com/checkcodecs.html> and see what your device can do!

# Media Capabilities API in the real world!

## Server side

Encode your content with at least one next-gen codec and one legacy codec

- Legacy codec: H.264 for its wide support
- Next-gen codecs:
  - H.265 (Apple devices and Smart TVs) or/and
  - AV1 (Emerging support but better performance than H.265)
- Bundle your HLS or MPEG-DASH manifest with your various codecs

## Client-side

Use a HTML5 video player that implements codecs selection with HLS or MPEG-DASH based on Media Capabilities API

- Radiant Media Player
- Shaka Player

# Best practices for building energy-efficient media web applications

- Do not use autoplay
- Do not use media content preloading
- Make your Adaptive Bitrate Streaming (ABR) logic more environment-friendly
  - Do not account for device pixel ratio in your ABR logic
  - Cap resolution to current player size in your ABR logic
  - Compute initial bandwidth with efficiency
  - Adapt buffer length based on content length
  - Make your ABR logic less aggressive
- Use Media Capabilities API
- Leave the viewer with a choice

# Q & A



RADIANT  
[MEDIA]  
PLAYER