Video Filters

extending Webcodecs

Rijubrata Bhaumik

Web Platform Architect, Intel

WebCodecs Extension API

- Post processing Video in the Web via new WebCodecs extension for both Real-Time (streaming) and **Offline** (Video Editing) use cases.
- Expose Filter capabilities based on VideoFrames to avoid Memory copies
- Removing Noise, enhancing color contrast
- More complex filters such as background blur, face retouch and AI based filters.

Video Processing Options – Today (Link) - Francois & Dom

Option	Capability	Pro	Con
JavaScript	Manipulate pixels/images on CPU	Easy	Slowest Memory copies
Web Assembly	Manipulate pixels/images on CPU	Wide support Near native speed Cross compile	Not as fast as GPU Memory copies
WebGL	Leverages GPU	Wide Support	Write shaders Debugging is hard
WebGPU	Leverages GPU	Low level	Memory Copies ? importExternalTexture
WebNN	Can run models on images/Frames	XPU (future)	
WebCodecs-extension (this proposal)	Leverages h/w (GPU) XPU (future)	Easy for anybody u... WebCodecs	PnP analysis vs WebGPU ?

I can't write shaders ☹

Video Processing Options – one day

Option	Capability	Pro	Con
JavaScript	Manipulate pixels/images on CPU	Easy	Slowest Memory copies
Web Assembly	Manipulate pixels/images on CPU	Wide support Near native speed Cross compile	Not as fast as GPU Memory copies
WebGL	Leverages GPU	Wide support	Write shaders Debugging is hard
WebGPU	Leverages GPU	Low level	Memory Copies ? importExternalTexture
WebNN	Can run models on images/Frames	XPU	
WebCodecs-extension (this proposal)	Leverages h/w (GPU) XPU (future)	Easy for anybody using WebCodecs.	PnP analysis vs WebGPU ?

Add VideoFilters to *your* existing web app EASILY

Platform support today

- Windows: Microsoft Media Foundation/DXVA
- Linux: VA-API
- MacOS: Apple Core Image Filter

Filter	Windows	Linux	MacOS
Brightness, Contrast, Hue, Saturation	✓	✓	✓
White balance	✓	✓	✓
Color space conversion	✓	✓	✓
Denoise		✓	✓
Scaling	✓	✓	✓
Deinterlace	✓	✓	✓
Blur			✓
Sharpen		✓	✓
Transpose/rotation		✓	✓
Image blending	✓		✓
Video composition	✓		✓
Skin color enhancement		✓	
Gamma correction		✓	✓

Stakeholders - *whom should we talk to ?*

Adobe Creative cloud

Climpchamp

Screencastify, inVideo, PlayPlay, Kapwing

OBS ?

How - WebCodecs Extension API

```
[Exposed=(Window,DedicatedWorker), SecureContext]
interface VideoFilter {
  constructor(VideoFilterInit init);

  readonly attribute FilterState state;
  readonly attribute unsigned long filterQueueSize;

  undefined configure(VideoFilterConfig config);
  undefined filter(VideoFrame frame);
  Promise<undefined> flush();
  undefined reset();
  undefined close();

  static Promise<VideoFilterSupport> isConfigSupported(VideoFilterConfig config);
};
```

VideoFilterConfig

```
dictionary VideoFilterConfig {
  required DOMString filter;
  [EnforceRange] unsigned long codedWidth;
  [EnforceRange] unsigned long codedHeight;
  [EnforceRange] unsigned long displayAspectWidth;
  [EnforceRange] unsigned long displayAspectHeight;
  VideoColorSpaceInit colorSpace;
  HardwareAcceleration hardwareAcceleration = "no-preference";
  boolean optimizeForLatency;
};

// When filter == "denoising"
dictionary VideoFilterDenoisingConfig : VideoFilterConfig {
  float strength;
};
```

Sample Code

```
class WebCodecTransform {
  constructor() {
    this.decoder_ = null;
    this.encoder_ = null;
    this.controller_ = null;

    /** TODO(riju) : Put up multiple filters */
    this.filter_brightness_ = null;
    /**
    this.filter_contrast_ = null;
    this.filter_hue_ = null;
    **/
    this.frame_count_ = 0;
    this.start_time_ = false;
  }
/** @override */
  async init() {
    ...
    this.filter_brightness_ = new VideoFilter({
      output: frame => this.handleFilteredFrame(frame),
      error: this.error
    });
    ...
    this.filter_brightness_.configure({filter: 'brightness',
                                       codedWidth: 720,
                                       codedHeight: 576 });
  }
```

```
async transform(frame, controller) {
  if (!this.filter_brightness_) {
    frame.close();
    return;
  }
  try {
    this.controller_ = controller;
    this.filter_brightness_.filter(frame);
  } finally {
    frame.close();
  }
}

handleFilteredFrame(videoFrame) {
  if (!this.controller_) {
    videoFrame.close();
    return;
  }
  this.controller_.enqueue(videoFrame);
```

Chainable Filtering API

```
interface FilterPipeline {
    constructor(FilterPipelineConfig config);

    Promise<undefined> configurePipeline(FilterPipelineConfig config);
        Promise<FilterElement> createElement(FilterElementConfig config);
    Promise<undefined> deleteElement(FilterElement element);
    Promise<WritableStream> importPort(FilterElement element, DOMString port);
    Promise<ReadableStream> exportPort(FilterElement element, DOMString port);
    Promise<sequence<DOMString>> getAvailableFilterTypes();

    readonly attribute FilterPipelineConfig config;
    readonly attribute sequence<FilterElement> elements;
    readonly attribute sequence<FilterPipelineImport> in;
    readonly attribute sequence<FilterPipelineExport> out;
    readonly attribute FilterPipelineState state;
};

dictionary FilterPipelineImport {
    WritableStream writable;
    FilterElement element;
    DOMString port;
};

dictionary FilterPipelineExport {
    ReadableStream readable;
    FilterElement element;
    DOMString port;
};

dictionary FilterPipelineConfig {
    // TBD
};

enum FilterState {
    "unconfigured",
    "configured",
    "closed"
};
```

```
interface FilterElement {
    Promise<undefined> configureElement(FilterElementConfig config);

    readonly attribute FilterElementConfig config;
    readonly attribute sequence<FilterPort> in;
    readonly attribute sequence<FilterPort> out;
};

partial dictionary FilterElementConfig {
    required DOMString type;
    // TBD
};

partial dictionary FilterElementConfig {
    // when type == "denoise"
    float strength;
};

partial dictionary FilterElementConfig {
    // when type == "scale"
    unsigned long outputWidth;
    unsigned long outputHeight;
};

interface FilterPort {
    Promise<undefined> configurePort(FilterPortConfig config);
    Promise<undefined> connectPort(FilterElement element, DOMString port);
    Promise<undefined> detachPort();

    readonly attribute DOMString name;
    readonly attribute FilterElement peer;
    readonly attribute DOMString peerPort;
    readonly attribute FilterPortConfig config;
};

partial dictionary FilterPortConfig {
};
```

Example of Filter Chains - Scaling and Then Denoising

```
// Build a pipeline such that:   [writable] -> scaler -> denoiser -> [readable]
const pipeline = new FilterPipeline({});
const scaler = pipeline.createElement({ type: "scale", outputWidth: 640, outputHeight: 480 });
const denoiser = pipeline.createElement({ type: "denoise", strength: 0.6 });
await scaler.out[0].connectPort(denoiser, denoiser.in[0].name);
const inport = await pipeline.importPort(scaler, scaler.in[0].name);
const outport = await pipeline.exportPort(denoiser, denoiser.out[0].name);

// Use insertable streams to get and push video data
const stream = await getUserMedia({video:true});
const videoTrack = stream.getVideoTracks()[0];
const processor = new MediaStreamTrackProcessor({track: videoTrack});
const generator = new MediaStreamTrackGenerator({kind: 'video'});

processor.readable.pipeThrough({ writable: inport, readable: outport }).pipeTo(generator.writable);
const videoBefore = document.getElementById('video-before');
const videoAfter = document.getElementById('video-after');
videoBefore.srcObject = stream;
const streamAfter = new MediaStream([generator]);
videoAfter.srcObject = streamAfter;
```

What next ?

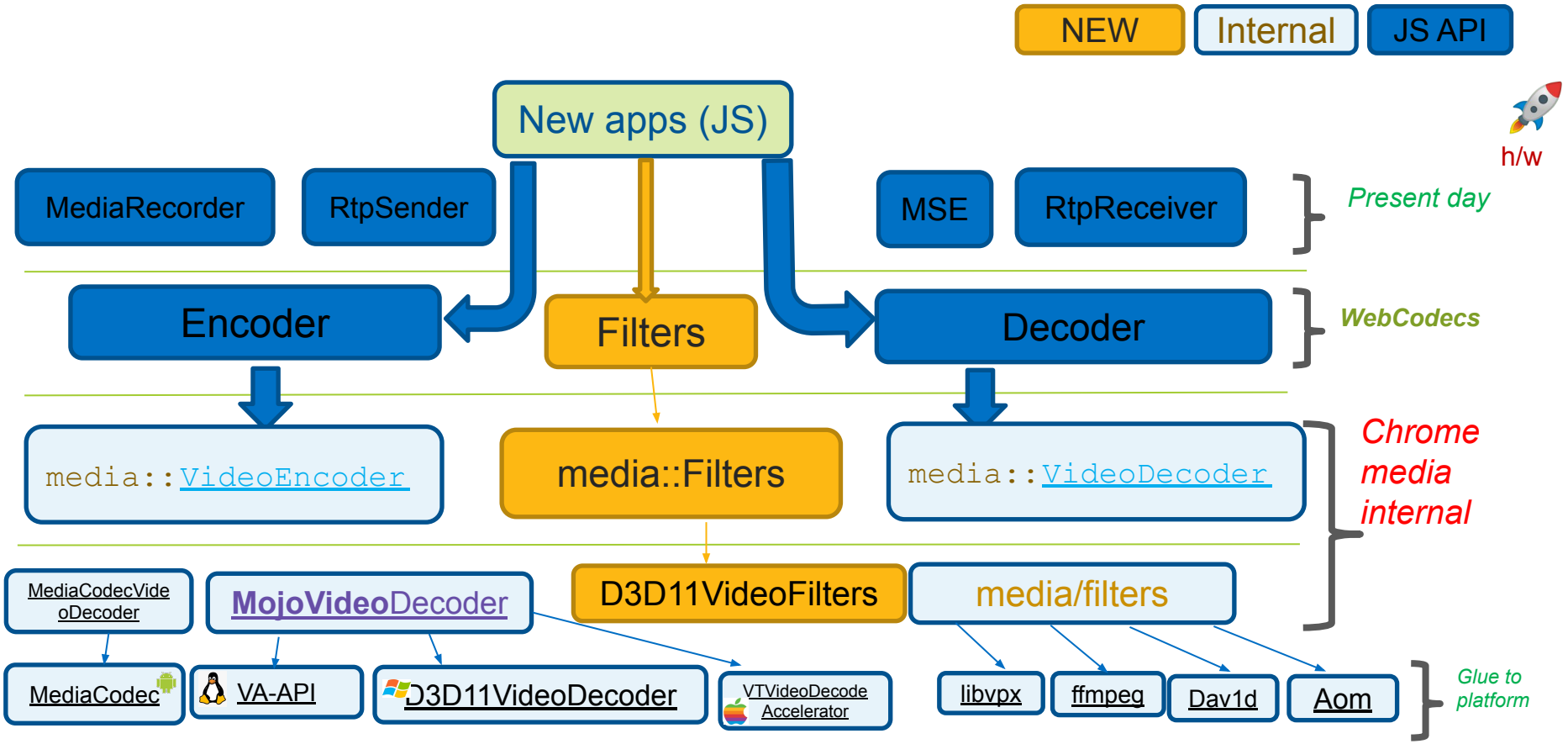
Demo app

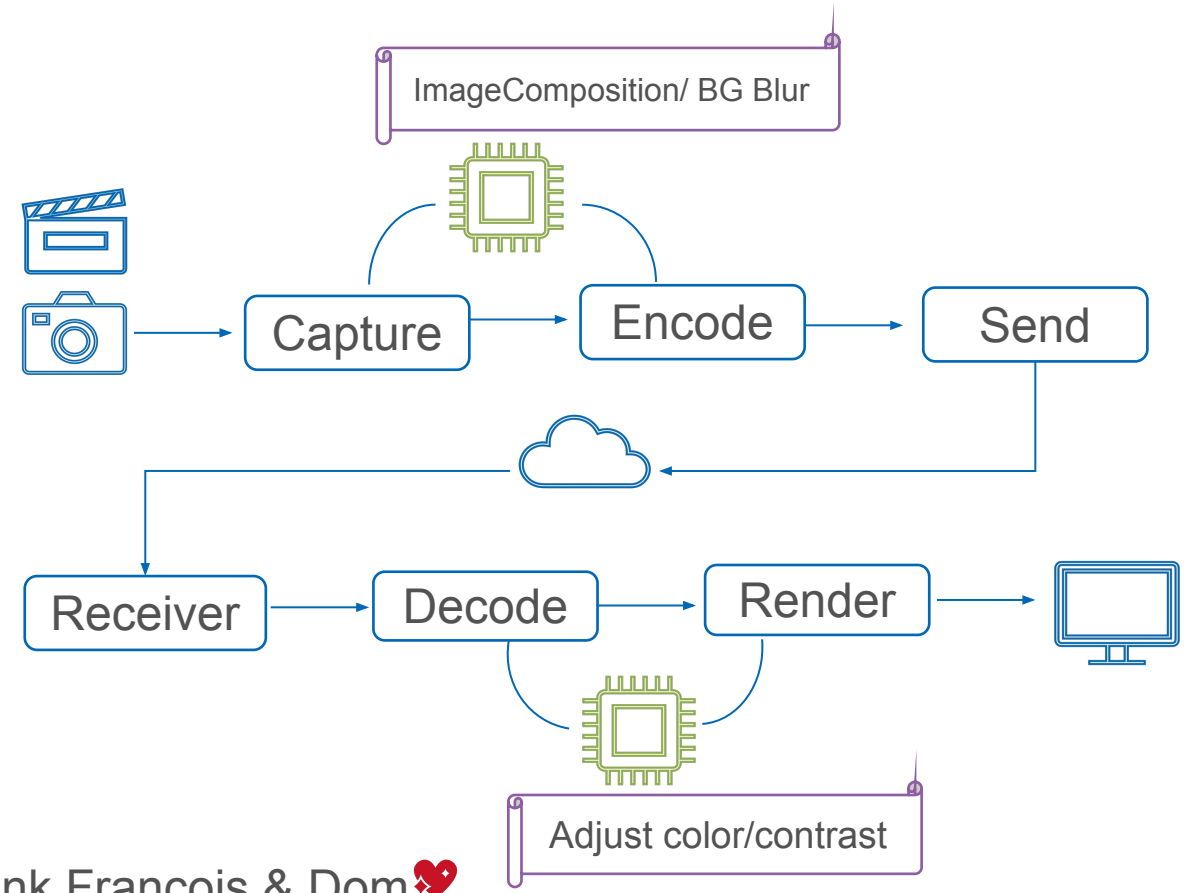
Make a stable PoC

Collect PnP Data against various options

Feedback from stakeholders

Thanks !


NEW
Internal
JS API
New apps (JS)
h/w
MediaRecorder
RtpSender
MSE
RtpReceiver
Present day
Encoder
Filters
Decoder
WebCodecs
media::VideoEncoder
media::Filters
media::VideoDecoder
Chrome media internal
MediaCodecVideoDecoder
MojoVideoDecoder
D3D11VideoFilters
media/filters
MediaCodec
VA-API
D3D11VideoDecoder
VTVideoDecodeAccelerator
libvpx
ffmpeg
Dav1d
Aom
Glue to platform



Processing : Video Filters API

Source:

HTMLMediaElement,

Media Source Extensions

WebRTC, getUserMedia,

- Processing:

WebAssembly,

WebGPU,

Web Neural Network API

Link Francois & Dom