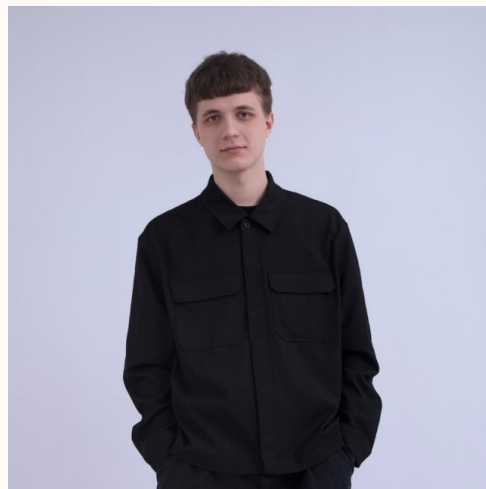


Лукьянчук Ярослав @nmequalmn

Оптимизация покадровой перемотки в Chromium

Обо мне

Работаю разработчиком интерфейсов в
Яндексе
Более трёх лет занимаюсь веб-разработкой
и Node.js



О чём поговорим?

01 Проблема и мотивация

02 Архитектура
медиапайплайна Chromium

03 Локализация проблемы

04 Альтернативные решения

05 А что в других браузерных
движках?

06 Фикс

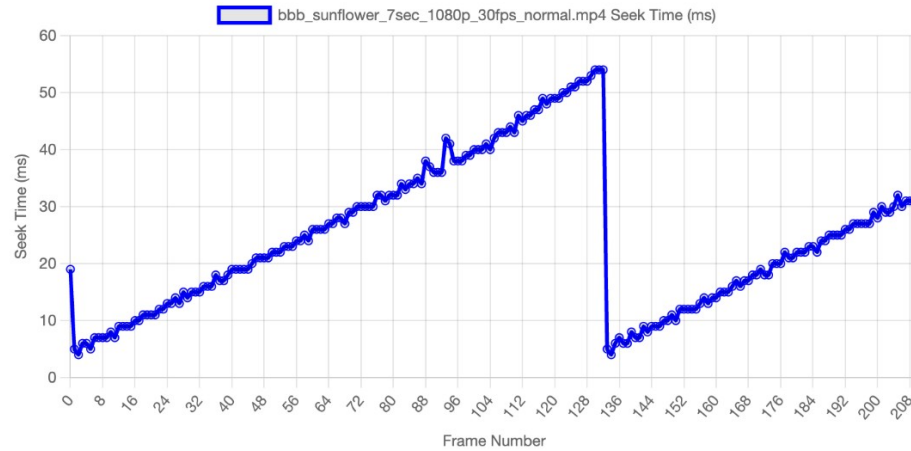
07 Бенчмарки

Проблема и мотивация

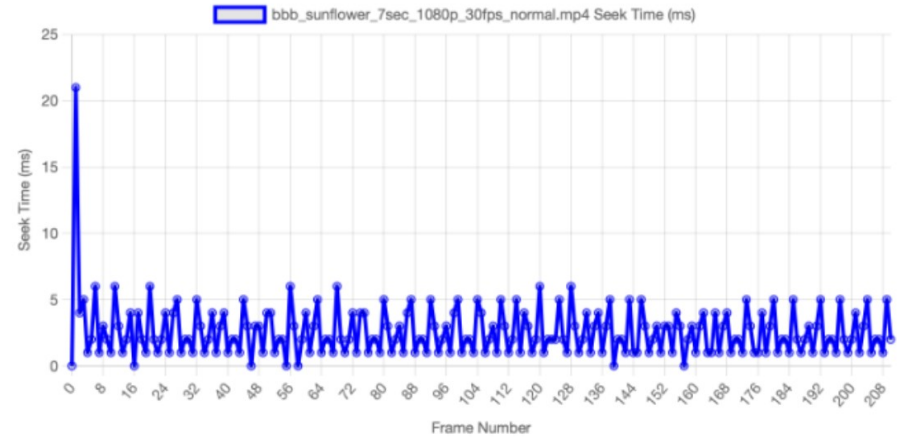
```
1  const video = document.querySelector("video");
2
3  const FPS = 30;
4  const frameDuration = 1 / FPS;
5
6  async function seek(time) {
7    return new Promise(resolve => {
8      const onSeeked = () => {
9        video.removeEventListener("seeked", onSeeked);
10       resolve();
11     };
12
13     video.addEventListener("seeked", onSeeked, { once: true });
14     video.currentTime = time;
15   });
16 }
17
18 async function playFrameByFrame() {
19   if (video.readyState < 1) {
20     await new Promise(resolve => {
21       video.addEventListener("loadedmetadata", resolve, { once: true });
22     });
23   }
24
25   for (let t = 0; t < video.duration; t += frameDuration) {
26     await seek(t);
27
28     await new Promise(resolve => setTimeout(resolve, 33));
29   }
30 }
31
32 playFrameByFrame();
```

→ $O(n^2)$

Chromium



Safari



Optimize HTMLVideoElement forward seeking by avoiding unnecessary decoder resets

+1 10

Hotlists (3)

Mark as Duplicate

Comments (14)

Dependencies (0)

Duplicates (0)

Blocking (0)

Resources (13)

Bug

P3

+ Add Hotlist

Triaged-ET

Needs-Milestone x

STATUS UPDATE

No update yet.

DESCRIPTION

Is...@gmail.com created issue #1

May 18, 2025 08:32AM

What steps will reproduce the problem? Currently, when seeking with an HTMLVideoElement, Chromium always resets the demuxer and decoder to the previous keyframe. This behavior causes a performance bottleneck during forward seeking, especially when seeking frame-by-frame

I've created a demo illustrating this issue: <https://stepancar.github.io/articles/articles/html-video-element-seeking/>

In the demo, you can see that seeking becomes progressively slower when using a video that contains only a single I-frame. This slowdown occurs because Chromium discards decoder state and restarts decoding from the last keyframe—even when the requested seek position is only slightly ahead (e.g., +30ms).

You can prove it your self

[normal video](#)

[video with one I-frame](#)

this video was created by using ffmpeg command:

```
ffmpeg -i bbb_sunflower_7sec_1080p_30fps_normal.mp4 \
-c:v libx264 -g 9999 -keyint_min 9999 -sc_threshold 0 \
-force_key_frames "0" -pix_fmt yuv420p \
-preset fast -crf 18 \
bbb_sunflower_7sec_1080p_30fps_one-iframe.mp4
```

Proposed Change:

We should optimize seeking logic to detect when the next seek target is close to the current decoding position (e.g., forward within 0.5s) and, in such cases, allow the decoder to continue from the current frame instead of resetting to the previous keyframe. This would significantly improve responsiveness in scrubbing scenarios.

Safari seems to implement a more efficient strategy here, as it does not exhibit the same performance degradation in similar cases.

Impact:

This change would:

Reduce latency when seeking forward through a video Improve UX in scrubbing-heavy applications (e.g., video editors, review tools)

What is the expected result?

EstimatedDay --

s

Flaky-Test --

Merge --

Merge-Request --

Milestone 138

NextAction --

Notice --

OS Linux

Mac

Windows

ReleaseBlock --

Respin --

IRM Link --

Security_Rele ase --

vrp-reward --

Fixed By Code --

Changes

Introduced In --

Found In 136

137

138

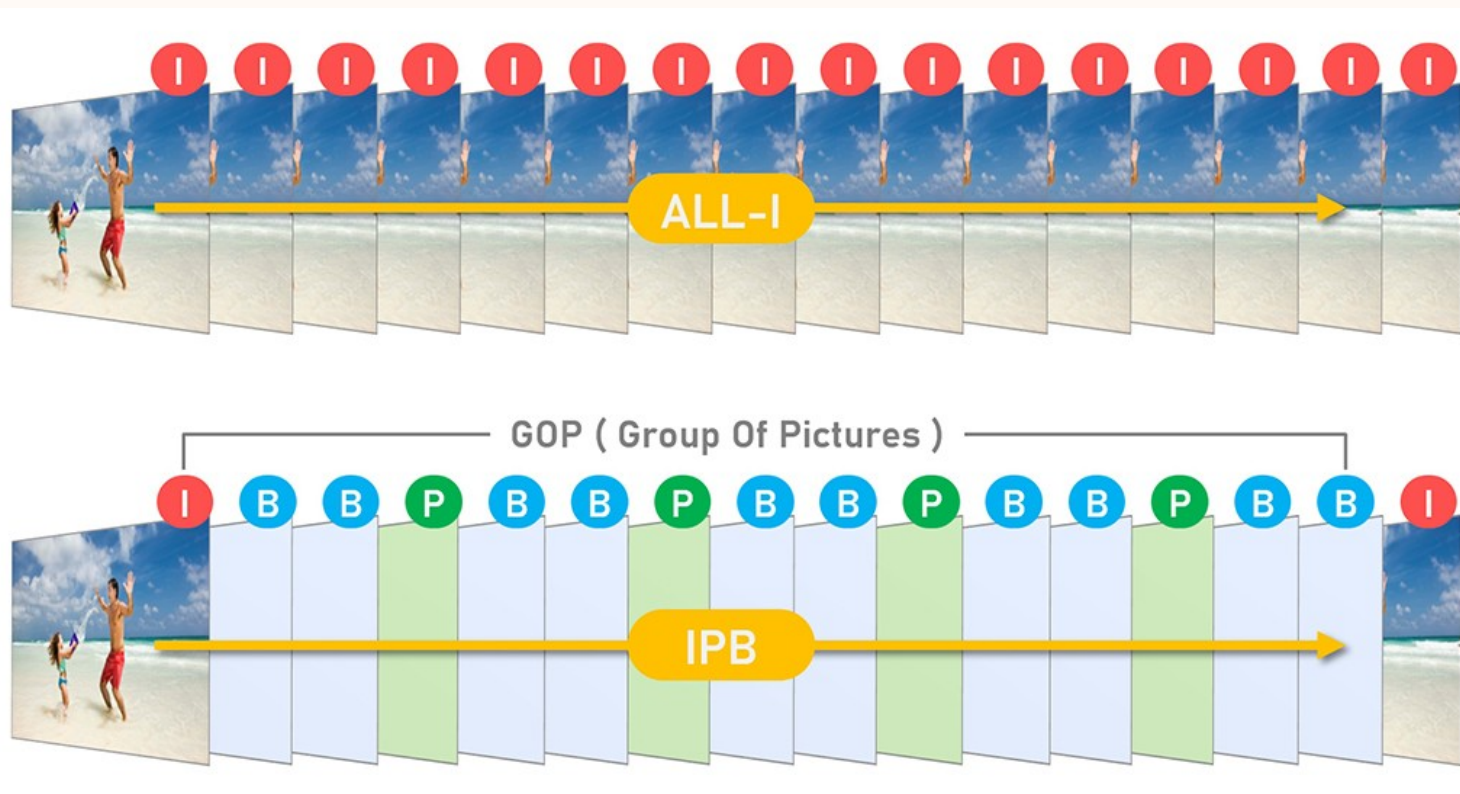
Targeted To 138

Verified In --

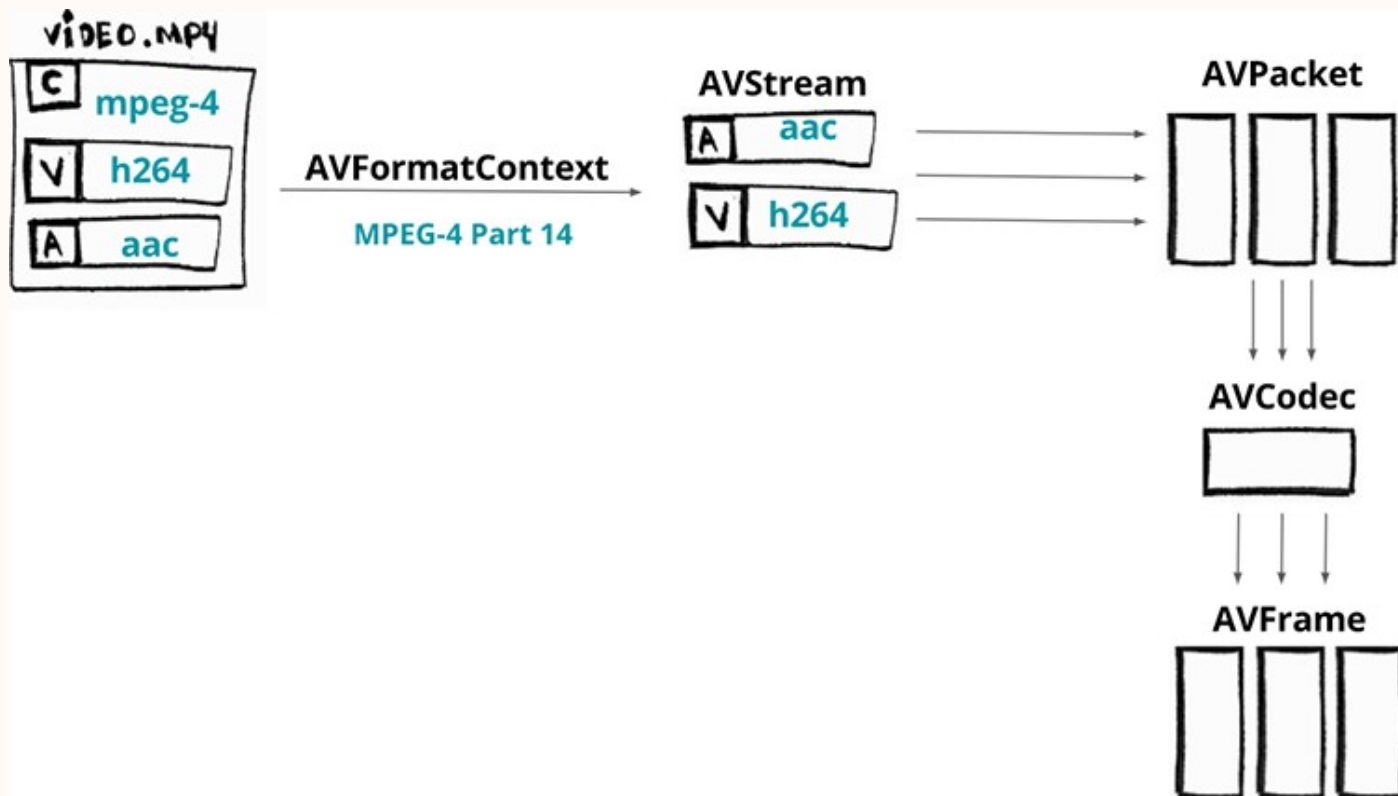
In Prod ☒

Show 1 additional field

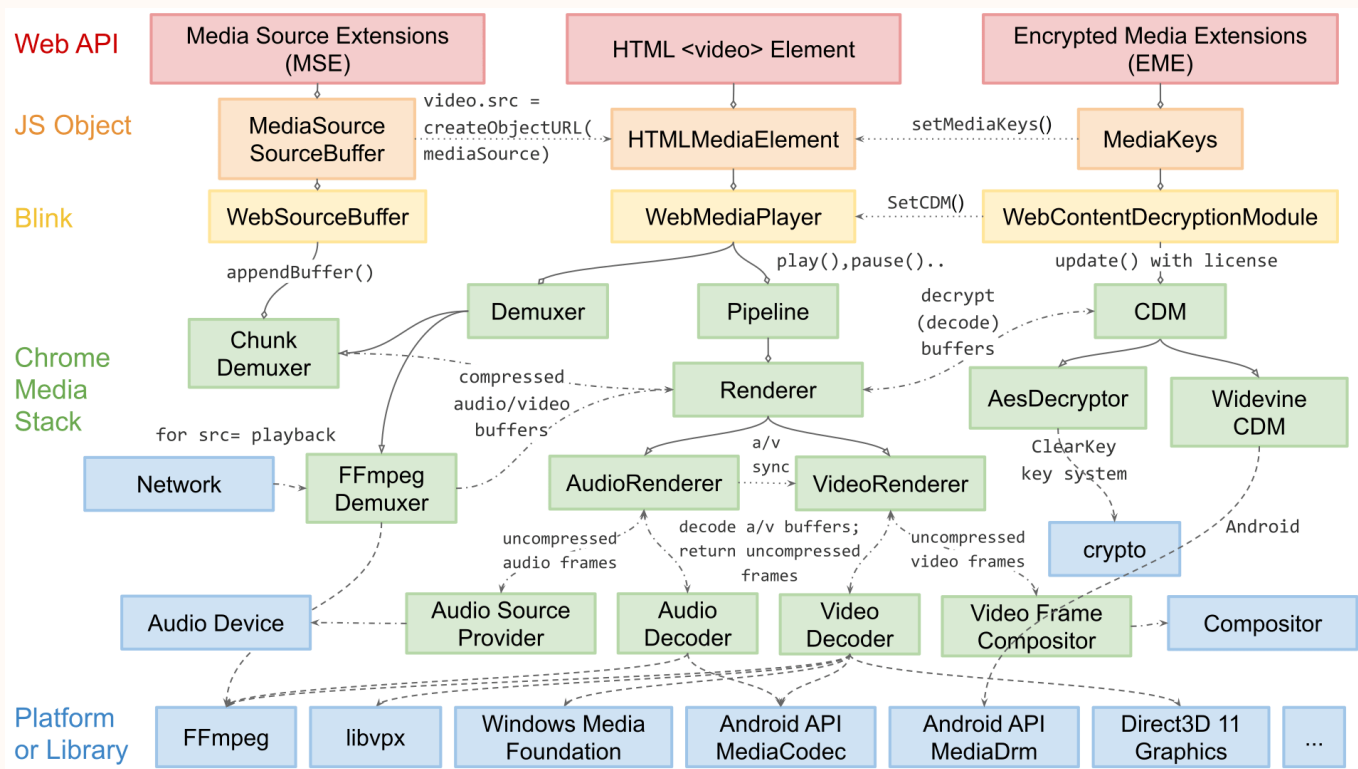
Немного теории видео

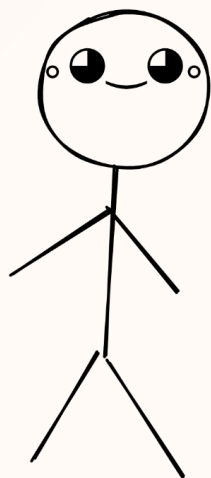


Немного теории видео



Архитектура медиапайплайна Chromium





хочу видео



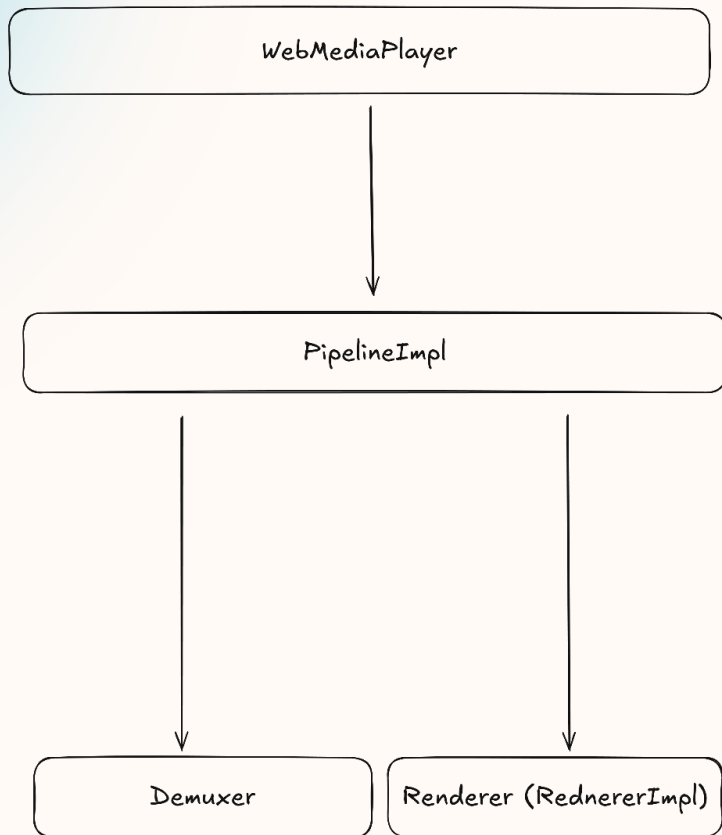
HTMLMediaElement

HTMLMediaElement



WebMediaPlayer

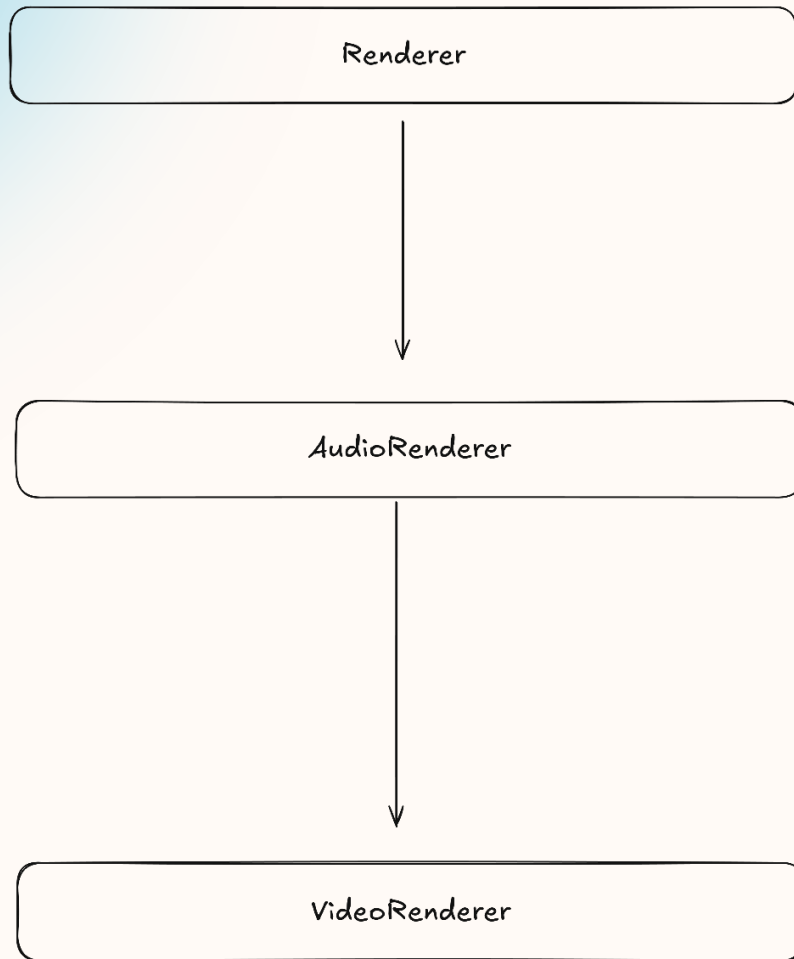
```
3559 std::unique_ptr<blink::WebMediaPlayer> RenderFrameImpl::CreateMediaPlayer(  
3560     const blink::WebMediaPlayerSource& source,  
3561     WebMediaPlayerClient* client,  
3562     blink::MediaInspectorContext* inspector_context,  
3563     WebMediaPlayerEncryptedMediaClient* encrypted_client,  
3564     WebContentDecryptionModule* initial_cdm,  
3565     const blink::WebString& sink_id,  
3566     const cc::LayerTreeSettings* settings,  
3567     scoped_refptr<base::TaskRunner> compositor_worker_task_runner) {  
3568     // `settings` should be non-null since the WebView created for  
3569     // RenderFrameImpl always composites.  
3570     DCHECK(settings);  
3571     return media_factory_.CreateMediaPlayer(  
3572         source, client, inspector_context, encrypted_client, initial_cdm, sink_id,  
3573         GetLocalRootWebFrameWidget()->GetFrameSinkId(), *settings,  
3574         agent_scheduling_group->agent_group_scheduler().CompositorTaskRunner(),  
3575         std::move(compositor_worker_task_runner));  
3576 }
```



```

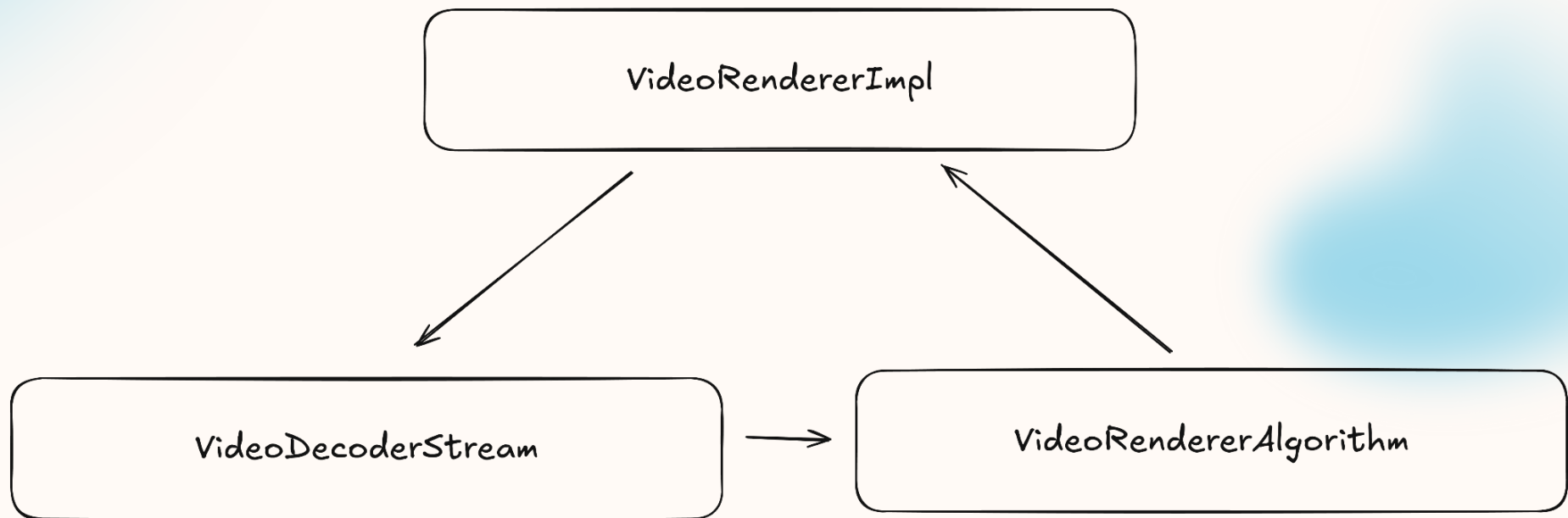
369 void PipelineImpl::RendererWrapper::Start(
370     StartType start_type,
371     Demuxer* demuxer,
372     std::unique_ptr<Renderer> default_renderer,
373     base::WeakPtr<PipelineImpl> weak_pipeline) {
374     DCHECK_CALLED_ON_VALID_SEQUENCE(shared_state_.media_sequence_checker);
375     DCHECK(state_ == State::kCreated || state_ == State::kStopped)
376         << "Received start in unexpected state: " << GetStateString(state_);
377     DCHECK(!demuxer_);
378     DCHECK(!renderer_ended_);
379
380     SetState(State::kStarting);
381     demuxer_ = demuxer;
382     default_renderer_ = std::move(default_renderer);
383     weak_pipeline_ = weak_pipeline;
384
385     // Setup |error_cb_| on the media thread.
386     error_cb_ = base::BindRepeating(&RendererWrapper::OnPipelineError,
387                                     weak_factory_.GetWeakPtr());
388
389     // Queue asynchronous actions required to start.
390     DCHECK(!pending_callbacks_);
391     SerialRunner::Queue fns;
392
393     // Initialize demuxer.
394     fns.Push(base::BindOnce(&RendererWrapper::InitializeDemuxer,
395                             weak_factory_.GetWeakPtr()));
396
397     // Once the demuxer is initialized successfully, media metadata must be
398     // available - report the metadata to client. If starting without a renderer
399     // we'll complete initialization at this point.
400     fns.Push(base::BindOnce(&RendererWrapper::ReportMetadata,
401                             weak_factory_.GetWeakPtr(), start_type));
402
403     // Create renderer.
404     fns.Push(base::BindOnce(&RendererWrapper::CreateRenderer,
405                             weak_factory_.GetWeakPtr()));
406
407     // Initialize renderer.
408     fns.Push(base::BindOnce(&RendererWrapper::InitializeRenderer,
409                             weak_factory_.GetWeakPtr()));
410
411     // Run tasks.
412     pending_callbacks_ = SerialRunner::Run(
413         std::move(fns),
414         base::BindOnce(&RendererWrapper::CompleteSeek, weak_factory_.GetWeakPtr(),
415                       base::TimeDelta()));
416 }

```



```
145 void RendererImpl::Initialize(MediaResource* media_resource,
146                               RendererClient* client,
147                               PipelineStatusCallback init_cb) {
148     DVLOG(1) << __func__;
149     DCHECK(task_runner_>RunsTasksInCurrentSequence());
150     DCHECK_EQ(state_, STATE_UNINITIALIZED);
151     DCHECK(init_cb);
152     DCHECK(client);
153     TRACE_EVENT_BEGIN("media", "RendererImpl::Initialize", GetTracingTrack(this));
154
155     client_ = client;
156     media_resource_ = media_resource;
157     init_cb_ = std::move(init_cb);
158
159     if (HasEncryptedStream() && !cdm_context_) {
160         DVLOG(1) << __func__ << ": Has encrypted stream but CDM is not set.";
161         state_ = STATE_INIT_PENDING_CDM;
162         OnWaiting(WaitingReason::kNoCdm);
163         return;
164     }
165
166     state_ = STATE_INITIALIZING;
167     InitializeAudioRenderer();
168 }
169
```

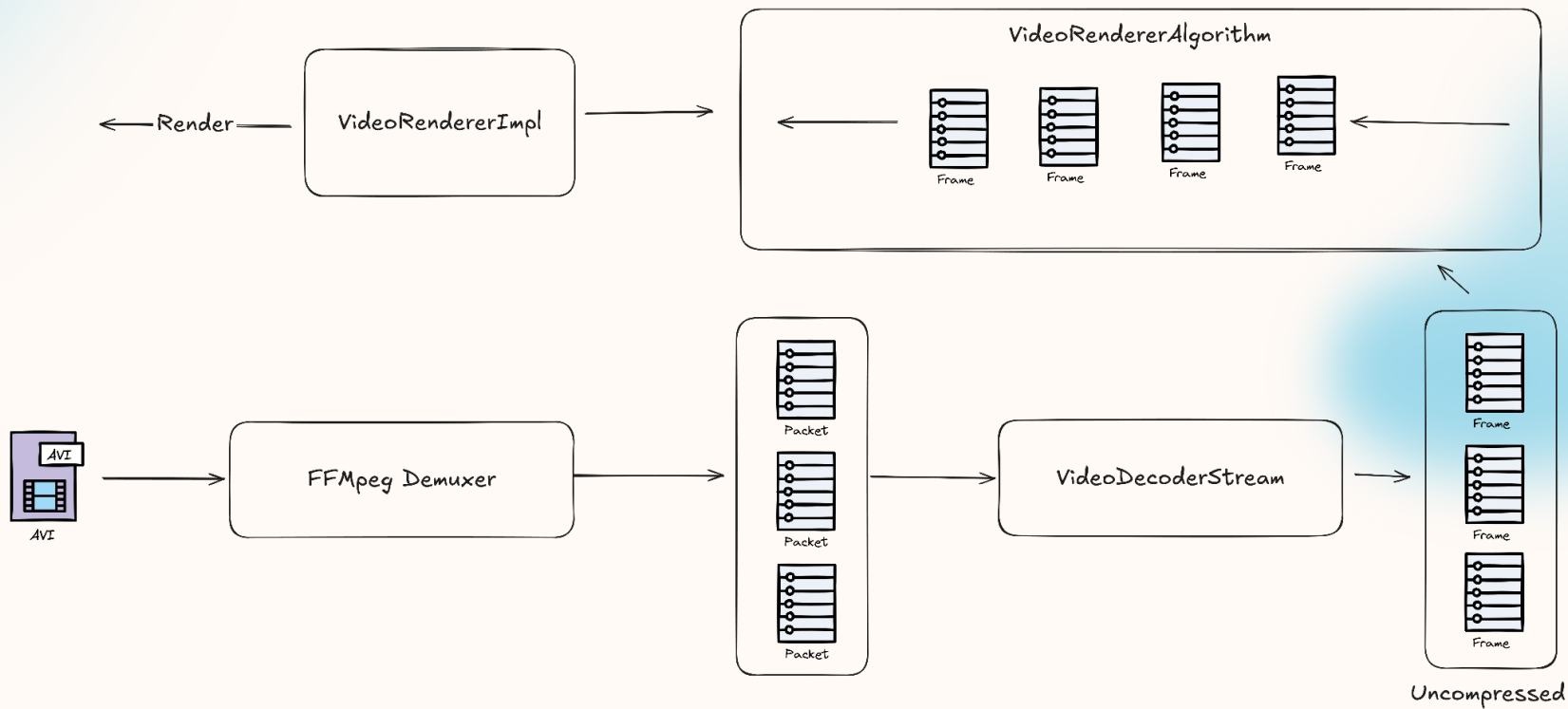
```
430 void RendererImpl::OnAudioRendererInitializeDone(PipelineStatus status) {
431     DVLOG(1) << __func__ << ": " << status;
432     DCHECK(task_runner_>RunsTasksInCurrentSequence());
433
434     // OnError() may be fired at any time by the renderers, even if they thought
435     // they initialized successfully (due to delayed output device setup).
436     if (state_ != STATE_INITIALIZING) {
437         DCHECK(!init_cb_);
438         audio_renderer_.reset();
439         return;
440     }
441
442     if (status != PIPELINE_OK) {
443         FinishInitialization(status);
444         return;
445     }
446
447     DCHECK(init_cb_);
448     InitializeVideoRenderer();
449 }
```

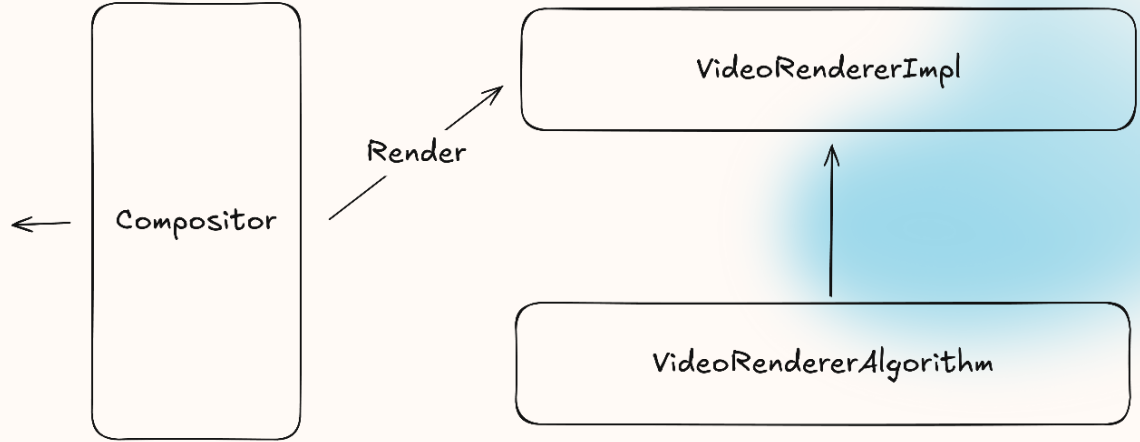
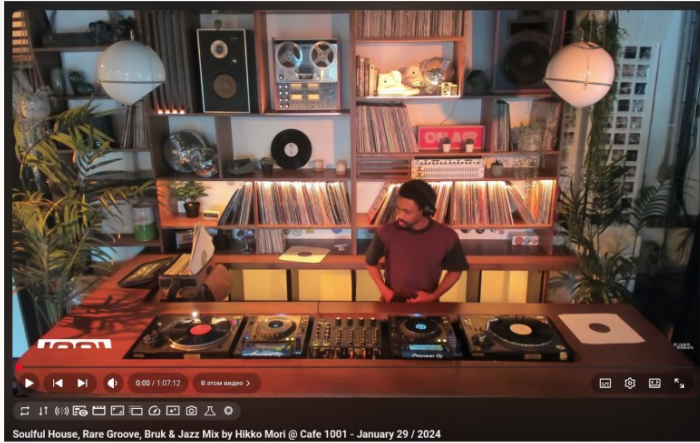


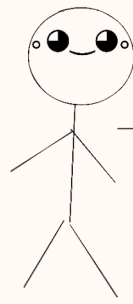
```

150 void VideoRendererImpl::StartPlayingFrom(base::TimeDelta timestamp) {
151     DVLOG(1) << __func__ << "(" << timestamp.InMicroseconds() << ")";
152     DCHECK(task_runner_>RunsTasksInCurrentSequence());
153     base::AutoLock auto_lock(lock_);
154     DCHECK_EQ(state_, kFlushed);
155     DCHECK(!pending_read_);
156     DCHECK_EQ(buffering_state_, BUFFERING_HAVE_NOTHING);
157
158     state_ = kPlaying;
159     start_timestamp_ = timestamp;
160     painted_first_frame_ = false;
161     last_render_time_ = last_frame_ready_time_ = base::TimeTicks();
162     video_decoder_stream_>SkipPrepareUntil(start_timestamp_);
163     AttemptRead_Locked();
164 }

```







хочу видео

HTMLMediaElement

WebMediaPlayer

PipelineImpl

Demuxer

Renderer (RendererImpl)

AudioRenderer

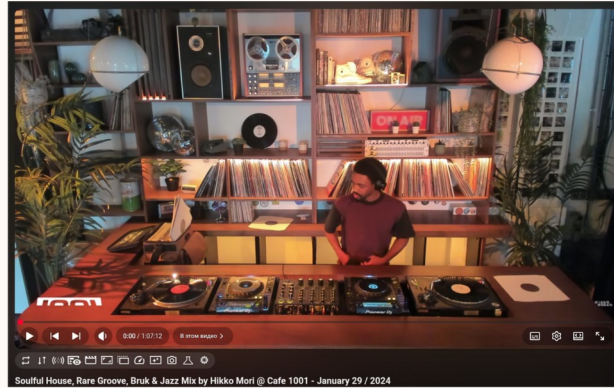
VideoRenderer

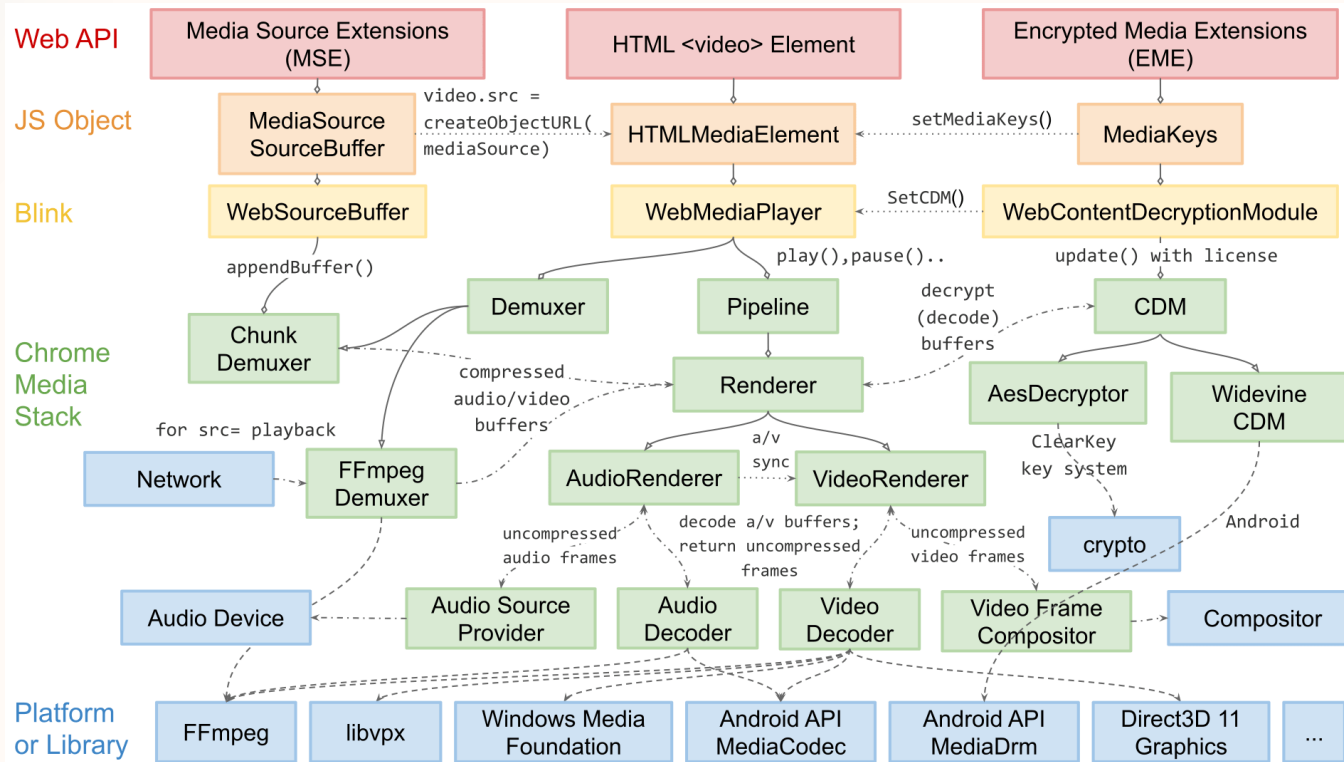
VideoRendererImpl

Compositor

VideoDecoderStream

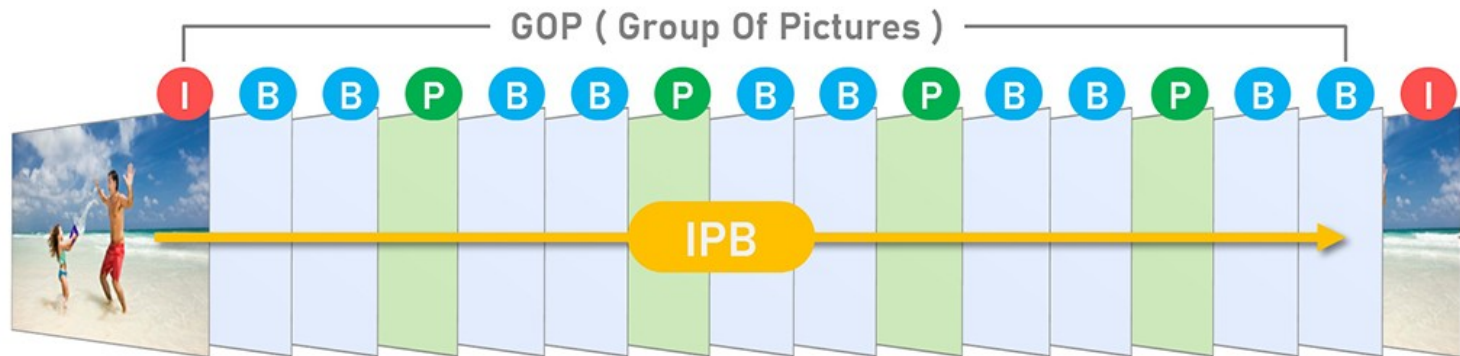
VideoRendererAlgorithm





Локализация проблемы

```
447 void PipelineImpl::RendererWrapper::Seek(base::TimeDelta time) {  
448     // Suppress seeking if we're not fully started.  
449     if (state_ != State::kPlaying) {  
450         DCHECK(state_ == State::kStopping || state_ == State::kStopped)  
451             << "Receive seek in unexpected state: " << GetStateString(state_);  
452         OnPipelineError(PIPELINE_ERROR_INVALID_STATE);  
453         return;  
454     }  
455     base::TimeDelta seek_timestamp = std::max(time, demuxer_->GetStartTime());  
456     SetState(State::kSeeking);  
457     renderer_ended_ = false;  
458     // Queue asynchronous actions required to start.  
459     DCHECK(!pending_callbacks_);  
460     SerialRunner::Queue bound_fns;  
461     // Abort any reads the renderer may be blocked on.  
462     demuxer_->AbortPendingReads();  
463     // Flush.  
464     DCHECK(shared_state_.media().renderer);  
465     bound_fns.Push(  
466         base::BindOnce(&Renderer::Flush,  
467             base::Unretained(shared_state_.media().renderer.get())));  
468     // Seek demuxer.  
469     bound_fns.Push(base::BindOnce(&Demuxer::Seek, base::Unretained(demuxer_),  
470         seek_timestamp));  
471     // Run tasks.  
472     pending_callbacks_ = SerialRunner::Run(  
473         std::move(bound_fns),  
474         base::BindOnce(&RendererWrapper::CompleteSeek, weak_factory_.GetWeakPtr(),  
475             seek_timestamp));  
476 }
```



```
69  int AVSeekFrame(AVFormatContext* s, int stream_index, int64_t timestamp) {  
70      // Seek to a timestamp <= to the desired timestamp.  
71      int result = av_seek_frame(s, stream_index, timestamp, AVSEEK_FLAG_BACKWARD);  
72      if (result >= 0) {  
73          return result;  
74      }  
75  
76      // Seek to the nearest keyframe, wherever that may be.  
77      return av_seek_frame(s, stream_index, timestamp, 0);  
78  }  
79
```

Если целевой кадр — следующий, можно избежать полного seek

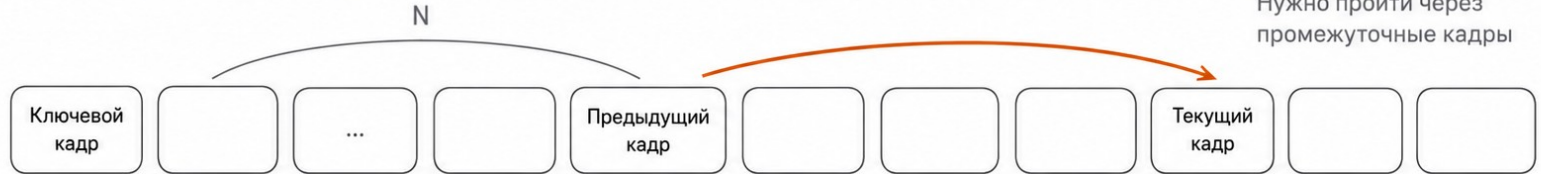
ShouldFastForward = true

Целевой кадр — следующий

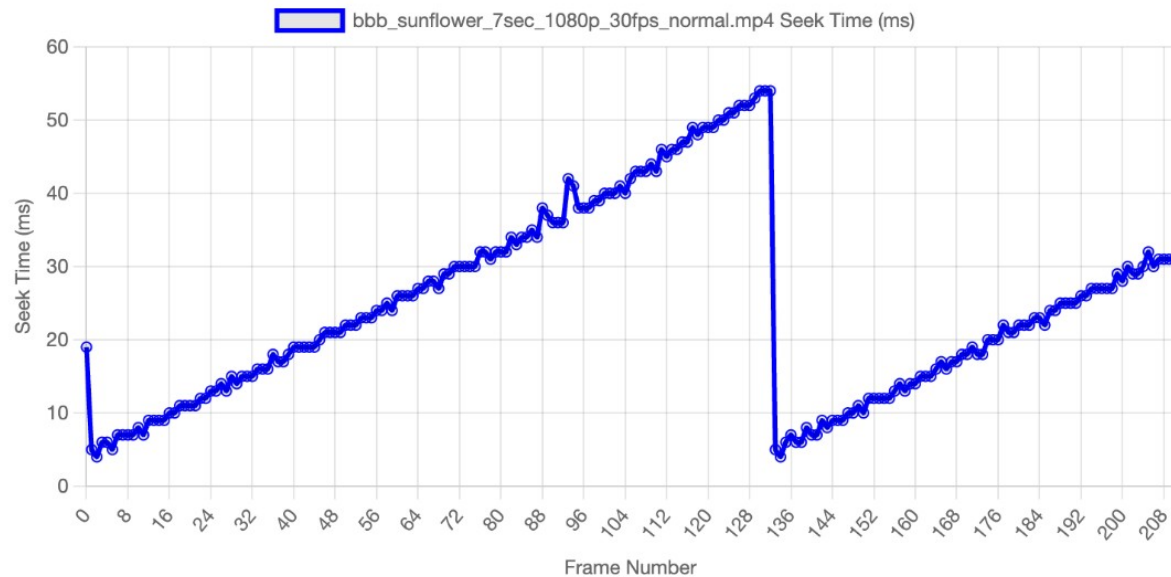


ShouldFastForward = false

Целевой кадр находится дальше



Chromium



HTMLMediaElement: seekToNextFrame() method

 **Deprecated:** This feature is no longer recommended. Though some browsers might still support it, it may have already been removed from the relevant web standards, may be in the process of being dropped, or may only be kept for compatibility purposes. Avoid using it, and update existing code if possible; see the [compatibility table](#) at the bottom of this page to guide your decision. Be aware that this feature may cease to work at any time.

 **Non-standard:** This feature is not standardized. We do not recommend using non-standard features in production, as they have limited browser support, and may change or be removed. However, they can be a suitable alternative in specific cases where no standard option exists.

The `HTMLMediaElement.seekToNextFrame()` method asynchronously advances the current play position to the next frame in the media.

 **Warning:** This non-standard method is part of an experimentation process around support for non-real-time access to media for tasks including filtering, editing, and so forth. You should *not* use this method in production code, because its implementation may change—or be removed outright—without notice. You are, however, invited to experiment with it.

This method lets you access frames of video media without the media being performed in real time. This also lets you access media using frames as a seek unit rather than timecodes (albeit only by seeking one frame at a time until you get to the frame you want). Possible uses for this method include filtering and editing of video content.

This method returns immediately, returning a `Promise`, whose fulfillment handler is called when the

ement > fastSeek()

HTMLMediaElement: fastSeek() method



Limited availability



This feature is not Baseline because it does not work in some of the most widely-used browsers.

Want more browser support for this feature? [Tell us why.](#)

[Learn more](#) [See full compatibility](#)

The `HTMLMediaElement.fastSeek()` method quickly seeks the media to the new time with precision tradeoff.

🕒 **Note:** If you need to seek with precision, you should set `HTMLMediaElement.currentTime` instead.

Syntax

JS

Copy

```
fastSeek(time)
```

Parameters

`time`

Альтернативные решения



ffmpeg.wasm

+

- Поддерживает demux, decode, фильтры, encode и transcode
- Привычная модель FFmpeg и аргументы командной строки
- Не требует серверного видеопайплайна

-

- WebAssembly-версия заметно медленнее native FFmpeg
- Нужно загрузить WASM core перед началом работы
- Высокое потребление CPU и памяти; multi-thread требует ещё больше ресурсов

WebCodecs

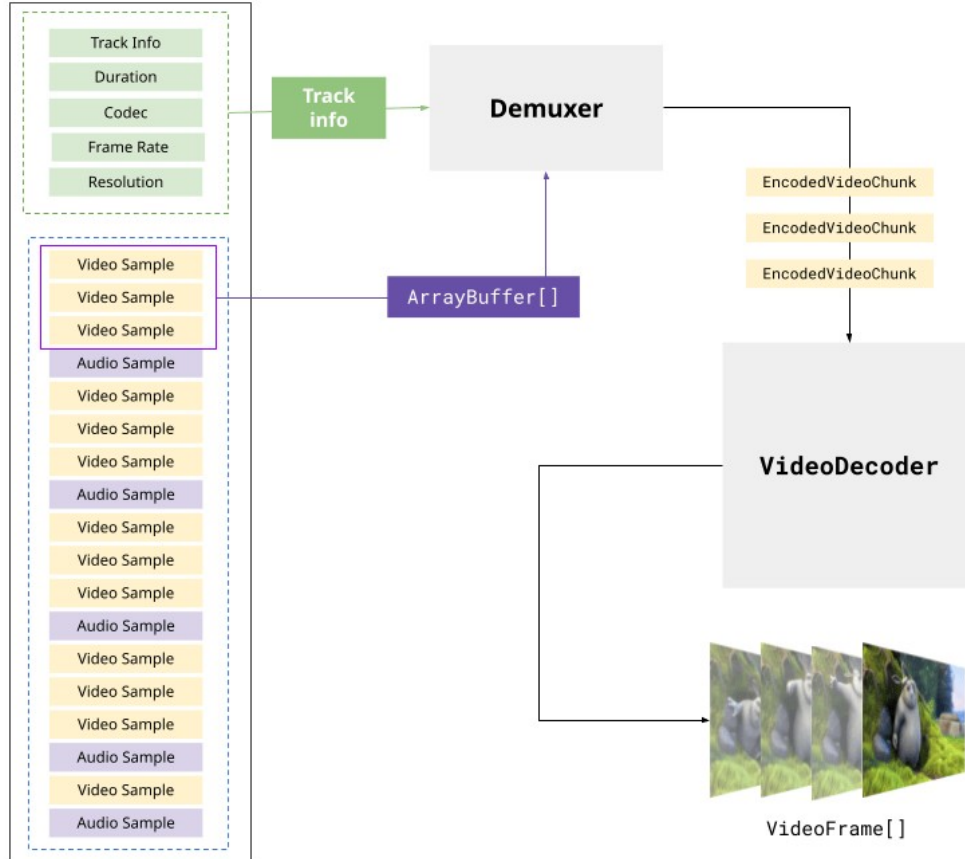
+

- Прямой доступ к VideoDecoder, VideoEncoder и VideoFrame
- Не нужно загружать собственную реализацию кодеков
- Возможна аппаратная акселерация
- Приложение контролирует очередь декодирования и время жизни кадров
- Хорошо подходит для редакторов, streaming и обработки кадров

-

- WebCodecs не является видеоплеером
- Контейнер нужно демультимплексировать самостоятельно
- После кодирования данные нужно самостоятельно собрать в MP4/WebM
- Синхронизация аудио, видео и рендеринга лежит на приложении
- Набор доступных кодеков зависит от браузера и устройства
- VideoFrame необходимо вовремя закрывать, иначе быстро расходуются CPU/GPU-ресурсы

Video File



Покадровая перемотка через WebCodecs

The logo for 'Big Buck Bunny' is displayed in a bold, blue, sans-serif font. The words 'Big Buck' are on the top line, and 'BUNNY' is on the bottom line. A small, stylized white rabbit is integrated into the letter 'N' of 'BUNNY'.

Следующий кадр →

Кадр 7 / 212

```

25  async function load() {
26      status.value = "Скачиваем видео...";
27      const response = await fetch(videoUrl);
28      if (!response.ok) throw new Error(`Не удалось загрузить видео: ${response.status}`);
29
30      const videoFile = new File([await response.blob()], "bbb.mp4", {
31          type: "video/mp4",
32      });
33      status.value = "Открываем видео...";
34      const demuxer = new WebDemuxer({ wasmFilePath: wasmUrl });
35      await demuxer.load(videoFile);
36
37      const stream = await demuxer.getMediaStream("video");
38      frameCount = Number(stream.nb_frames);
39      canvas.width = stream.width;
40      canvas.height = stream.height;
41      reader = demuxer.read("video").getReader();
42      decoder = new VideoDecoder({
43          output: (videoFrame) => frames.push(videoFrame),
44          error: (error) => (decoderError = error),
45      });
46      const config = demuxer.genDecoderConfig("video", stream);
47      decoder.configure(config);
48      await showNextFrame();
49  }

```

```

50
51 async function showNextFrame() {
52   while (!frames.length) {
53     const { done, value } = await reader.read();
54     if (done) {
55       await decoder.flush();
56       break;
57     }
58     await decode(value);
59     if (decoderError) throw decoderError;
60   }
61
62   if (!frames.length) throw new Error("Кадры закончились");
63   const videoFrame = frames.shift();
64   context.drawImage(videoFrame, 0, 0, canvas.width, canvas.height);
65   videoFrame.close();
66   frame += 1;
67   status.value = `Кадр ${frame} / ${frameCount}`;
68   next.disabled = frame === frameCount;
69 }
70
71 async function decode(chunk) {
72   decoder.decode(chunk);
73   if (decoder.decodeQueueSize) {
74     await new Promise((resolve) => decoder.addEventListener("dequeue", resolve, { once: true }));
75   }
76 }

```

Mediabunny

Complete media toolkit

A JavaScript library for reading, writing, and converting video and audio files. Directly in the browser, and faster than anybunny else.


[Hop in](#)
[API](#)
[Examples](#)
[Blog](#)
[Sponsors](#)
[License](#)

“

The missing part in the JS ecosystem for building web-first media apps.



Yonatan, Gling AI

“

Mediabunny gives you low-level control with high-level ease. It's fast, lightweight, and finally feels like media processing built for the web, not ported to it.



Konstantin, Diffusion Studio

```
npm install mediabunny
```

```
bash
```

Рендеринг на сервере

+

- Native FFmpeg, GPU и специализированное оборудование
- Предсказуемое окружение независимо от устройства пользователя
- Можно обрабатывать большие файлы и тяжёлые проекты
- Централизованные логи, мониторинг и повторяемые результаты
- Клиентское устройство почти не нагружается

-

- Вычисления, хранение и исходящий трафик стоят денег
- Нужно масштабировать workers, очереди и хранилище
- Плохо подходит для мгновенного покадрового взаимодействия

А что в других браузерных движках?



```
3942 void HTMLMediaElement::seek(const MediaTime& time)
3943 {
3944     ALWAYS_LOG(LOGIDENTIFIER, time);
3945     seekWithTolerance({ time, MediaTime::zeroTime(), MediaTime::zeroTime() }, true);
3946 }
3947
```



[AVFoundation](#) / [AVPlayer](#) / [seek\(to:toleranceBefore:toleranceAfter:\)](#)

Instance Method

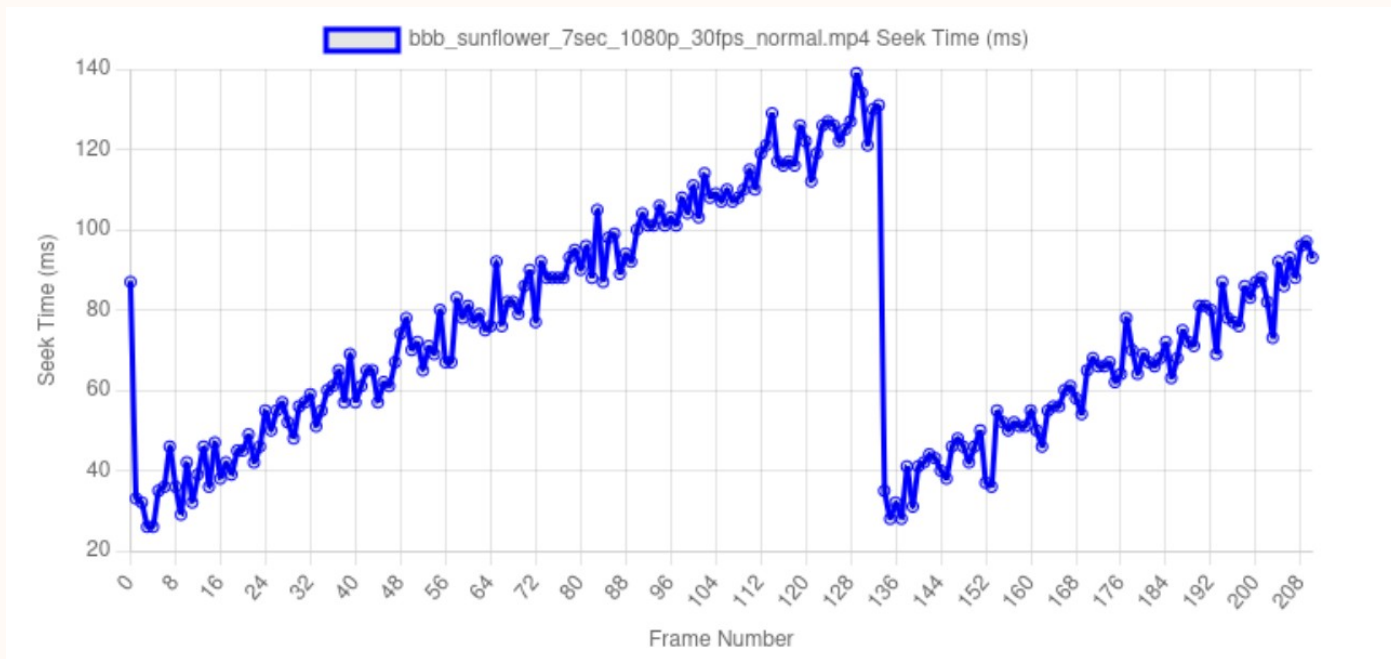
seek(to:toleranceBefore:toleranceAfter:)

Requests that the player seek to a specified time with the amount of accuracy specified by the time tolerance values.

iOS | iPadOS | Mac Catalyst | macOS | tvOS | visionOS | watchOS

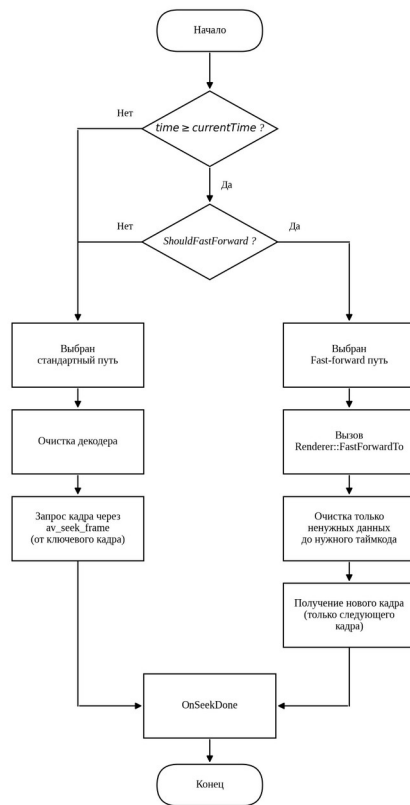
```
nonisolated
func seek(
    to time: CMTime,
    toleranceBefore: CMTime,
    toleranceAfter: CMTime
)
```

Firefox (Gecko)



Фикс

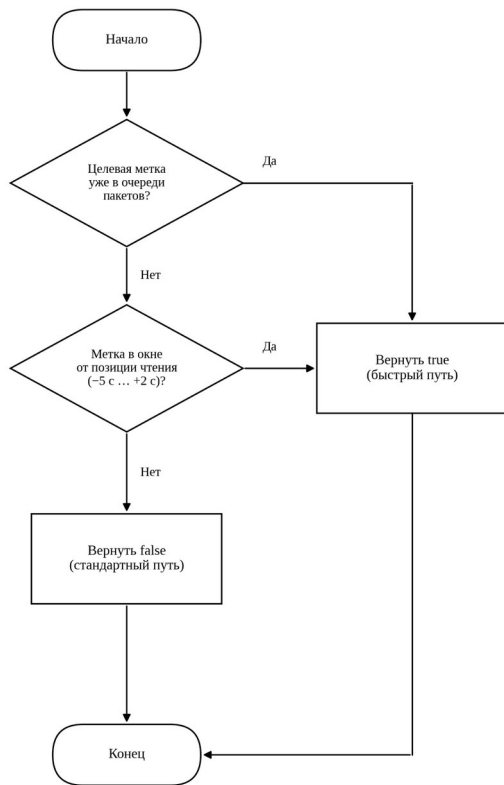
- Если целевой кадр следующий (ShouldFastForward), пропускаем полную перемотку с помощью операции FastForward
- Рендереры сами отбрасывают ненужные данные без Flush()
- Результат: переход к следующему кадру как при обычном воспроизведении



```

media / base / pipeline_impl.cc
451 void PipelineImpl::RendererWrapper::Seek(base::TimeDelta time) {
461
462     base::TimeDelta seek_timestamp = std::max(time, demuxer_->GetStartTime())
463
464     SetState(State::kSeeking);
465     renderer_ended_ = false;
466
467     DCHECK(!pending_callbacks_);
468     DCHECK(shared_state_.media().renderer);
469
470     // If both the demuxer and renderer can fast-forward to |seek_timestamp|,
471     // keep the renderer and demuxer state intact and discard old buffered da
472     Renderer* renderer = shared_state_.media().renderer.get();
473
474     const base::TimeDelta renderer_time = renderer->GetMediaTime();
475     const bool forward_seek = seek_timestamp >= renderer_time;
476     const bool demuxer_fast_forward =
477         forward_seek && demuxer_->ShouldFastForward(seek_timestamp);
478     const bool renderer_fast_forward =
479         forward_seek && demuxer_fast_forward &&
480         renderer->SupportsFastForward(seek_timestamp);
481
482     if (forward_seek && demuxer_fast_forward && renderer_fast_forward) {
483         renderer->FastForwardTo(
484             seek_timestamp,
485             base::BindOnce(&RendererWrapper::CompleteFastForwardSeek,
486                 weak_factory_.GetWeakPtr(), seek_timestamp));
487         return;
488     }

```



```

bool FFmpegDemuxer::ShouldFastForward(base::TimeDelta media_time) const {
    // Prefer video stream as reference; fall back to audio.
    const FFmpegDemuxerStream* ref_stream =
        GetFirstEnabledFFmpegStream(DemuxerStream::VIDEO);
    if (!ref_stream) {
        ref_stream = GetFirstEnabledFFmpegStream(DemuxerStream::AUDIO);
    }
    if (!ref_stream) {
        return false;
    }

    // Data still queued in the demuxer can be consumed without an FFmpeg seek.
    // Do not use GetBufferedRanges() here: it is historical buffered metadata and
    // may include packets already handed off to downstream decoders.
    if (ref_stream->HasQueuedPacketFor(media_time)) {
        return true;
    }

    // Targets near the read head can be reached without a full FFmpeg seek:
    // behind it by dropping already-read data, or ahead by reading forward.
    const base::TimeDelta last_read = ref_stream->GetLastPacketTimestamp();
    if (last_read == kNoTimestamp) {
        return false;
    }

    constexpr base::TimeDelta kMaxReadAhead = base::Seconds(5);
    if (media_time <= last_read) {
        return (last_read - media_time) <= kMaxReadAhead;
    }

    constexpr base::TimeDelta kMaxLookahead = base::Seconds(2);
    return (media_time - last_read) <= kMaxLookahead;
}

```

```

1094 void RendererImpl::FastForwardTo(base::TimeDelta media_time,
1095                                PipelineStatusCallback seek_done_cb) {
1096     DCHECK(task_runner_ ->RunsTasksInCurrentSequence());
1097     DCHECK_EQ(state_, STATE_PLAYING);
1098
1099     const bool restart_ticking = time_ticking;
1100     if (time_ticking) {
1101         time_ticking = false;
1102         time_source_ ->StopTicking();
1103         if (playback_rate_ > 0 && video_renderer_) {
1104             video_renderer_ ->OnTimeStopped();
1105         }
1106     }
1107     // When audio is present, `time_source` is the AudioRendererImpl. Its
1108     // SetMediaTime() contract is flushed-only, while the fast-forward path keeps
1109     // renderers playing. AudioRendererImpl::FastForwardTo() updates its own
1110     // media time below. Video-only playback uses WallClockTimeSource and still
1111     // needs to be moved here.
1112     if (!audio_renderer_) {
1113         time_source_ ->SetMediaTime(media_time);
1114     }
1115
1116     // Wait for all sub-renderers.
1117     if (!audio_renderer_ && !video_renderer_) {
1118         CompleteFastForwardTo(restart_ticking, std::move(seek_done_cb));
1119         return;
1120     }
1121
1122     // Wrap seek_done_cb in a BarrierClosure so it fires once both audio and video
1123     // have finished discarding data before |media_time|.
1124     int count = (audio_renderer_ ? 1 : 0) + (video_renderer_ ? 1 : 0);
1125     base::RepeatingClosure barrier = base::BarrierClosure(
1126         count, base::BindOnce(&RendererImpl::CompleteFastForwardTo, weak_this_,
1127                               restart_ticking, std::move(seek_done_cb)));
1128
1129     if (audio_renderer_) {
1130         audio_renderer_ ->FastForwardTo(media_time, barrier);
1131     }
1132     if (video_renderer_) {
1133         video_renderer_ ->FastForwardTo(media_time, barrier);
1134     }
1135 }

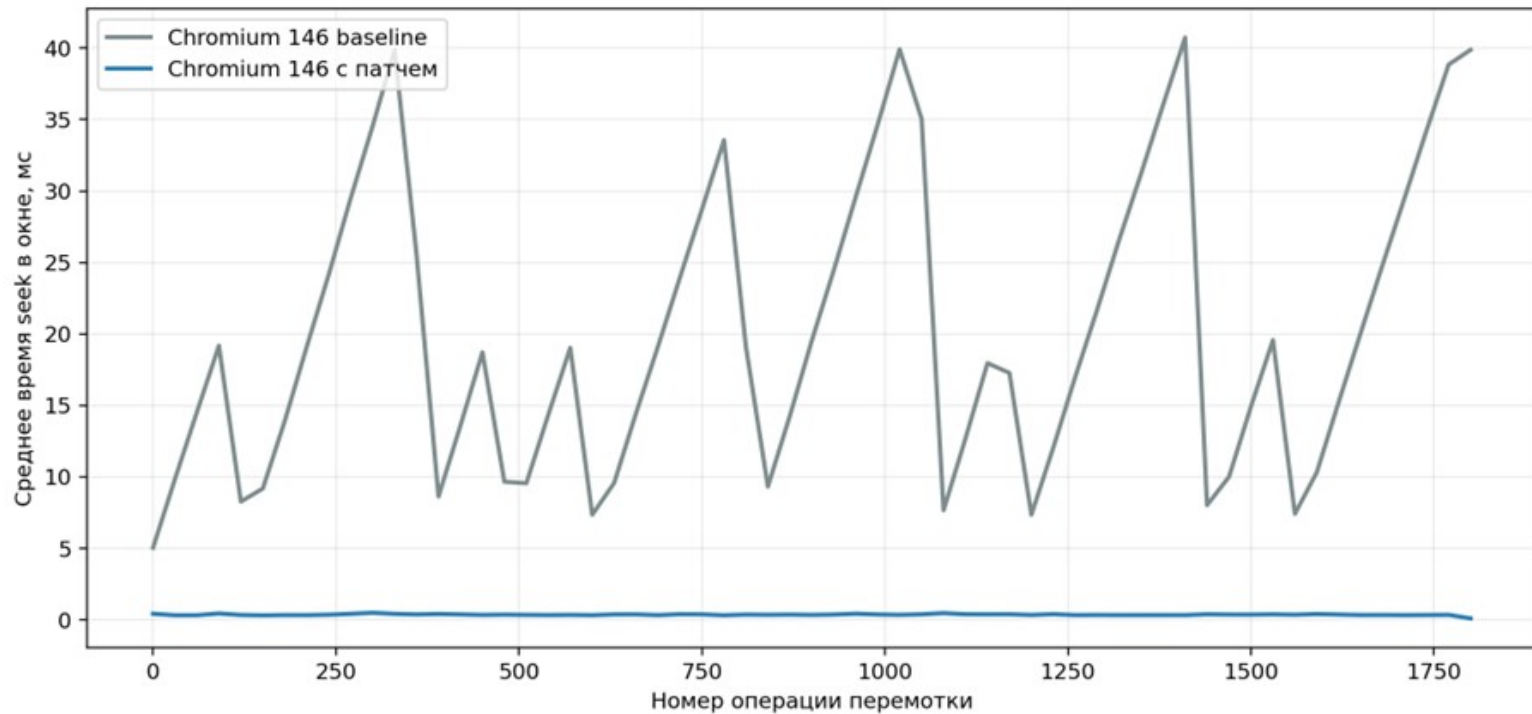
```

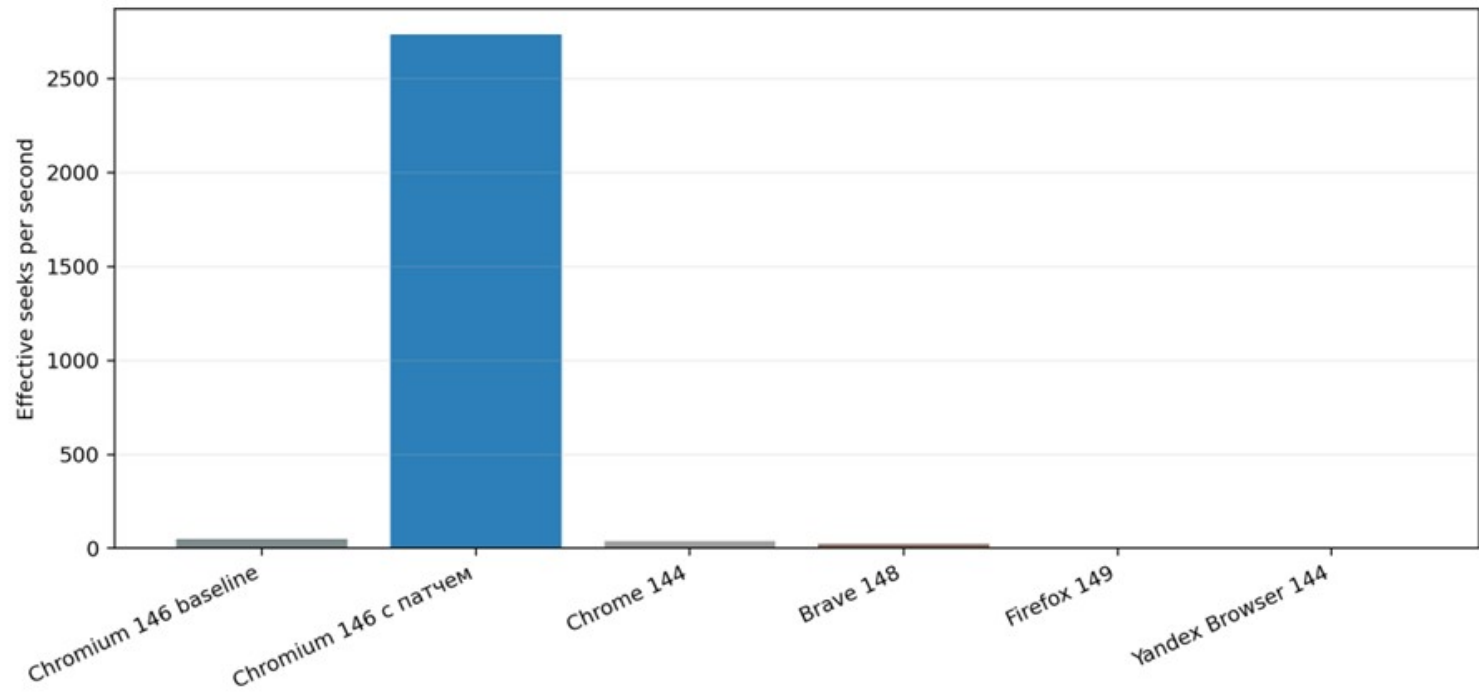
```

166 void VideoRendererImpl::FastForwardTo(base::TimeDelta time,
167                                     base::OnceClosure callback) {
168     DCHECK(task_runner_>RunsTasksInCurrentSequence());
169
170     base::OnceClosure ready_cb;
171     {
172         base::AutoLock auto_lock(lock_);
173         DCHECK_EQ(state_, kPlaying);
174         DCHECK(!fast_forward_cb_);
175         fast_forward_cb_ = std::move(callback);
176
177         // Cancel any pending first-frame paint that was scheduled for the previous
178         // position; it will be re-triggered once the new position is rendered.
179         paint_first_frame_cb_.Cancel();
180
181         start_timestamp_ = time;
182         painted_first_frame_ = false;
183         last_render_time_ = last_frame_ready_time_ = base::TimeTicks();
184
185         received_end_of_stream_ = false;
186         rendered_end_of_stream_ = false;
187
188         // Keep already-decoded frames that can be used for the target timestamp.
189         algorithm_>DiscardFramesBefore(start_timestamp_);
190         const bool has_usable_frame = algorithm_>HasFrameForTime(start_timestamp_);
191
192         if (has_usable_frame) {
193             video_decoder_stream_>FastForwardTo(start_timestamp_);
194             ready_cb = TakeFastForwardCallbackIfReady_Locked();
195         } else {
196             // Request fresh decoded output when the retained queue has no frame
197             // usable for the target timestamp.
198             video_decoder_stream_>FastForwardTo(start_timestamp_);
199             AttemptRead_Locked();
200         }
201
202         if (ready_cb && !sink_started_ && !painted_first_frame_ &&
203             algorithm_>frames_queued()) {
204             PaintFirstFrame_Locked(/*force_repaint=*/true);
205         }
206
207         AttemptRead_Locked();
208     }
209 }

```

Бенчмарки





Полезное

DeepWiki chromium/chromium

Index your code with Devin

Edit Wiki

Share

Last indexed: 1 July 2026 (43efce)

Overview

Repository Architecture

Development Workflow Integration

AI Assistant Integration

Agents System

Prompts and Templates

Skills System

Extensions and MCP Servers

Prompt Evaluation Framework

Common Utilities

Modernization and Code Health Projects

Claude Code Skills

GitHub Copilot Integration

Cursor AI Configuration

Development Environment Configuration

Code Formatting Configuration

Static Analysis Configuration

Language Server Configuration

Overview

> Relevant source files

This repository provides AI-assisted development infrastructure and tooling for the Chromium project. It contains automated workflow commands, code quality configurations, and AI assistant integrations that streamline common development tasks, enforce coding standards, and provide intelligent access to Chromium's extensive technical documentation [README.md](#) 1-22

For detailed information about specific subsystems:

- Repository structure and organization: see [Repository Architecture](#)
- How systems integrate into the development lifecycle: see [Development Workflow Integration](#)

AI Assistant Integrations

The Chromium development tools repository integrates multiple AI assistant systems to enhance developer productivity and code quality. These systems include the Gemini CLI, Claude Code, GitHub Copilot, and Cursor AI. The core of these integrations is the `agents/` directory, which serves as a centralized hub for AI coding agent resources [agents/README.md](#) 1-9

AI Coding Policy

The use of AI tools in Chromium is governed by a specific policy that emphasizes human

Ask Devin about chromium/chromium

Fast

Refresh this wiki

Enter email to refresh

On this page

Overview

AI Assistant Integrations

AI Coding Policy

Agents System

Gemini Command System

Development Automation Systems

Code Ownership and Governance

System Mapping: AI Agent Entities

Code Quality Tooling

Core Configuration Files

Tool Integration Flow

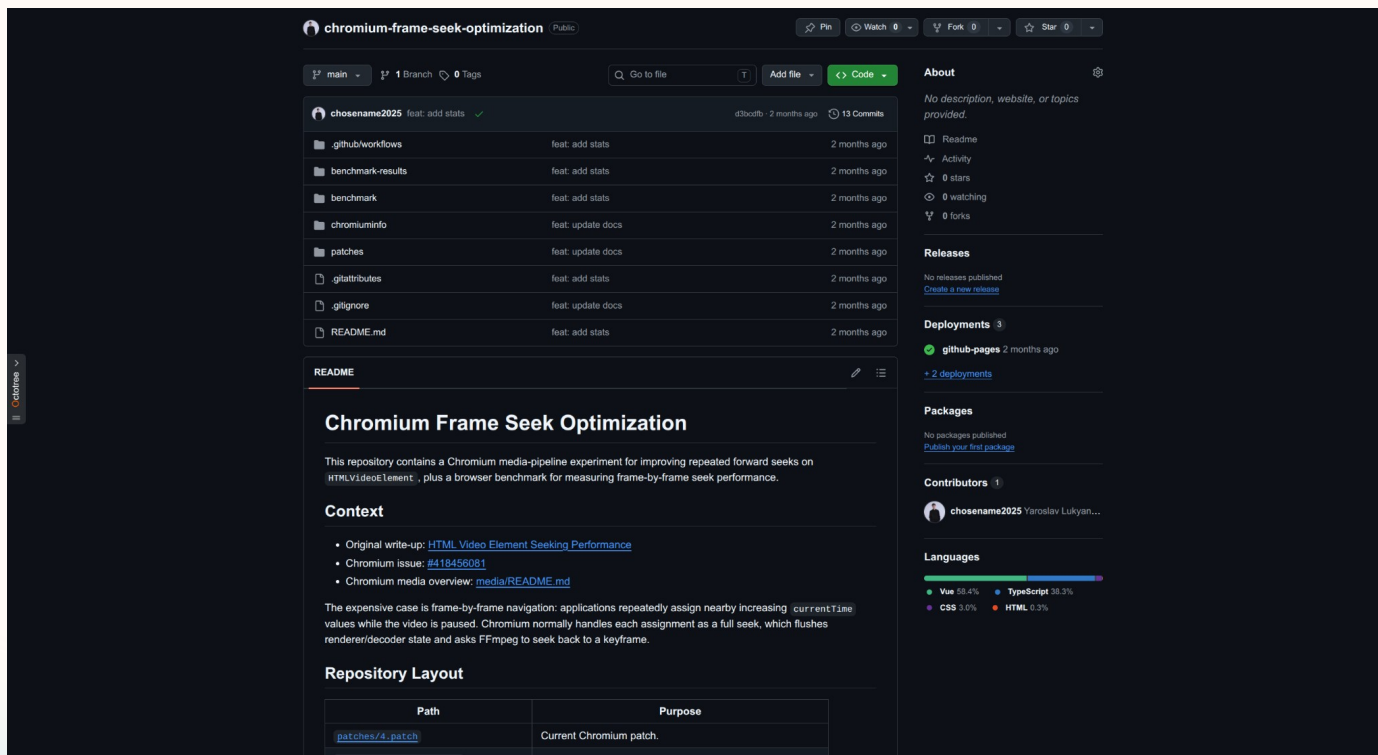
Repository Architecture

Development Workflow Integration

A square QR code with a black and white pixelated pattern, used for quick access to the repository content.

44

Полезное



The screenshot shows the GitHub repository page for 'chromium-frame-seek-optimization'. The repository is public and has 13 commits, 0 stars, and 0 forks. The main branch is 'main'. The repository contains several files and folders, including 'github/workflows', 'benchmark-results', 'benchmark', 'chromiuminfo', 'patches', '.gitattributes', '.gitignore', and 'README.md'. The README file is selected and shows the following content:

Chromium Frame Seek Optimization

This repository contains a Chromium media-pipeline experiment for improving repeated forward seeks on `HTMLVideoElement`, plus a browser benchmark for measuring frame-by-frame seek performance.

Context

- Original write-up: [HTML_Video_Element_Seeking_Performance](#)
- Chromium issue: [#418456081](#)
- Chromium media overview: [media/README.md](#)

The expensive case is frame-by-frame navigation: applications repeatedly assign nearby increasing `currentTime` values while the video is paused. Chromium normally handles each assignment as a full seek, which flushes `renderVideoDecoder` state and asks FFmpeg to seek back to a keyframe.

Repository Layout

Path	Purpose
patches/4_patch	Current Chromium patch.
patches/README.md	Problem summary and links to the original context.



Полезное

Active 759833 Add fast-forward seek path to media pipeline

Change Info

Owner Yaroslav Lukyanchuk (Mer...)

Reviewers Dale Curtis

CC mark a. foltz feature-media- chromium-revi...

Repo | Branch chromium/src | main

Topic

Submit Requirements

Code-Review No votes

No-Unresolved-Comments Unsatisfied

Code-Coverage No votes

Code-Owners 24 pending Add owners

Review-Enforcement Unsatisfied

Reply

Add fast-forward seek path to media pipeline

Allow sequential forward seeks to reuse buffered demuxer, decoder, and renderer state when the target timestamp can be reached safely without a full seek.

PipelineImpl now checks whether the seek is forward, the demuxer can reach the target without av_seek_frame, and the renderer can discard stale data without Flush(). When all checks pass, Renderer::FastForwardFo() advances audio/video state and completes the seek through the fast path. Otherwise, the existing Flush + Demuxer::Seek path is used unchanged.

Show All Edit

Comments 1 unresolved 1 resolved Create AI Review Prompt

Checks No results Choose Tryjobs

Tree Status Reverted the culprit CL Optimistically opening

Files

Comments

Checks

Patchset 1 → Patchset 2 26501ba Go to latest patch set

Diff view Download Expand All

	Comments	Size	Delta	I(Cov)	ΔCov	I(Cov(U))	ΔCov(U)	
1 → 2 File								
Commit message								
M → M media/renderers/renderer_impl.cc			-1 +8	-	-	-	-	Reviewed
			-1 +8					

Change Log

Show all entries (2 hidden) Expand All

Yaroslav Lukyanc... Added to cc: chromium-reviews@chromi... feature-media-reviews@chr...

Patchset 1 | May 20 2:47 AM

Yaroslav Lukyanc... Added to reviewer: Dale Curtis mark a. foltz (OOO until 8/5...)

Patchset 1 | May 20 11:05 AM

Yaroslav Lukyanc... Dale Curtis, who was added as reviewer owns the following files: * AUTHORS * media/base/audio_renderer.h * media/base/decoder_buffer_queue.cc * media/base/decoder_buffer_queue.h * media/base/demuxer.cc * media/base/de...

Patchset 1 | May 20 11:05 AM

Dale Curtis Moved from reviewer to cc: mark a. foltz (OOO until 8/5...)

Patchset 1 | May 20 8:54 PM

Dale Curtis 1 This is a lot of state to add for a niche use case. I'm inclined to decline this submission, but will dig through it more thoroughly this week and next.

Patchset 1 | May 20 8:54 PM



**Спасибо за
внимание!**

