

hls.js production checklist

Versions, controllers, ABR knobs, LL-HLS config, DRM matrix, error recovery, pitfalls.

Controllers you actually touch

Stream controller: state machine (IDLE → FRAG_LOADING → PARSING → BUFFER_APPENDING). Every event you listen to fires at a known transition.

Level controller: owns master + media playlists; reloads on schedule for live.

ABR controller: throughput EWMA × safety factor; replaceable via `hls.abrController = ...`.

Buffer controller: MSE SourceBuffer wrapper; serialises append / remove / endOfStream.

EME controller: `drmSystems` entry point; opens `MediaKeySession`, fetches license.

Loader: fetch / xhr; override with `xhrSetup` / `fetchSetup` for tokens, headers, rewrites.

Seven-line player (canonical pattern)

```
import Hls from 'hls.js';
const v = document.querySelector('video');
if (Hls.isSupported()) {
  const hls = new Hls();
  hls.loadSource('/master.m3u8');
  hls.attachMedia(v);
} else { v.src = '/master.m3u8'; }
```

Safari fallback uses native AVFoundation — preferred for HLS on iOS.

ABR knobs that matter

`abrBandwidthFactor` (0.95) safety factor for steady-state pick.

`abrBandwidthUpFactor` (0.7) stricter factor for an upshift; asymmetry is intentional.

`abrSwitchInterval` (v1.7-alpha) caps switches per second to stop ABR flapping.

`startLevel` force a specific rung at startup; -1 = auto.

`capLevelToPlayerSize` true → never load a rung higher than the `<video>` element.

`maxBufferLength` (30s) forward buffer target; pair with `maxMaxBufferLength`.

LL-HLS configuration

`lowLatencyMode` true (default since v1.4); enables parts + preload hints.

`liveSyncDuration` 3 → target ~3s behind live edge for a low-latency profile.

`liveMaxLatencyDuration` 10 → max drift before a seek to live edge.

`maxLiveSyncPlaybackRate` 1.1 → catch up by playing slightly faster.

CDN must support blocking playlist reload and chunked-transfer parts.

Apple removed HTTP/2 push from LL-HLS in 2023; current spec uses blocking reload + preload hints.

Error recovery rule (one sentence each)

`NETWORK_ERROR` → `hls.startLoad()`; cap retries with a counter.

`MEDIA_ERROR` (1st) → `hls.recoverMediaError()`; clear flag after 5s.

`MEDIA_ERROR` (2nd in 5s) → `swapAudioCodec()` + `recoverMediaError()`.

`KEY_SYSTEM_ERROR` → no auto-recovery; surface 'not available' message.

`MUX / OTHER_ERROR` → `destroy()` and reload.

Calling `destroy()` on every error is a one-line bug; read `data.type` first.

Top pitfalls in production

Not branching on Safari → `hls.js` fails or wastes battery; use `canPlayType` fallback.

Calling `destroy()` on every error → users get a reload prompt for a transient 5xx.

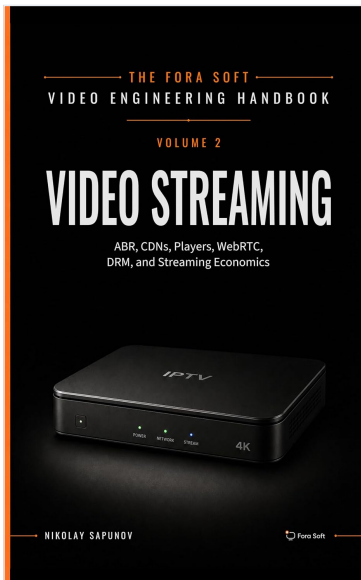
Listening for `MANIFEST_LOADED` instead of `MANIFEST_PARSED` → intermittent Firefox failures.

Assuming live `currentTime` starts at 0 → use `seekable.start(0)` / `seekable.end(0)` for live.

Default timeouts on satellite or 3G → loosen `manifestLoadingTimeout` / `fragLoadingTimeout`.

Multi-DRM configuration (drmSystems map)

Key system	drmSystems key (hls.js)	Browsers / platforms	Schemes
Widevine	<code>com.widevine.alpha</code>	Chrome, Edge, Firefox, Android, ChromeOS	cenc, cbcs
FairPlay	<code>com.apple.fps</code>	Safari (macOS, iOS, iPadOS), tvOS	cbcs only
PlayReady	<code>com.microsoft.playready</code>	Edge on Windows, Xbox, Tizen, webOS	cenc, cbcs



GO DEEPER · VOLUME 3 OF 7

Video Streaming: A Complete Guide to Video Delivery

From ABR and CDNs to Players, WebRTC, DRM, and the Economics of Streaming

By Nikolay Sapunov, CEO at Fora Soft · 1,182 pages · Kindle & paperback on Amazon

[Read it on Amazon →](#)
amazon.com/dp/B0HH76ZYWC

This checklist covers the hls.js controllers, ABR knobs, LL-HLS settings, DRM keys and error recovery you need in production. Volume 3 goes deeper into the player side: ABR algorithms and bitrate ladders, recovery after stalls, DRM systems, CDN caching and QoE monitoring.

WHAT YOU WILL FIND INSIDE

- ABR algorithms: buffer, throughput, hybrid
- DRM with Widevine, FairPlay, PlayReady
- CDN caching for low-latency HLS
- Player recovery after stalls and errors
- QoE monitoring for web players

THE FULL HANDBOOK · 7 VOLUMES · 9,000+ PAGES · ALL ON AMAZON

Vol. 1 · Video Encoding — pixels and color to H.264, HEVC, AV1, quality metrics and AI

Vol. 2 · Video Streaming I — network transport, contribution and ingest, HLS, MPEG-DASH, CMAF

Vol. 3 · Video Streaming II — ABR, CDNs, players, WebRTC, DRM and the economics of delivery

Vol. 4 · Audio for Video I — PCM, codecs, loudness, accessibility, real-time voice processing

Vol. 5 · Audio for Video II — jitter buffers, lip sync, Atmos, quality metrics, neural codecs

Vol. 6 · AI for Video I — models and inference, computer vision, speech, search, generative video

Vol. 7 · AI for Video II — real-time pipelines, agents, serving, cost, compliance, verticals

Shipping an hls.js player that has to survive real networks and DRM?

We will review your player config, ABR and error handling, and test it on throttled networks and the browsers your viewers use. 250+ video products shipped since 2005.

[Book a 30-min call →](#)
[WhatsApp →](#)
[Email us →](#)