AUDIO & VIDEO PRODUCTION

Audio Editing & Production Basics

Reading the Waveform · Noise Reduction & Normalization

Equalization & Compression

Multi-Track Basics & Exporting

Capstone: Cleaning Up and Mixing a Podcast Episode

Philip Osztromok

Generated with Claude

Table of Contents

Audio Editing & Production Basics — 8 Chapters

Video Editing Fundamentals' own Chapter 7 already covered audio work — Fairlight, DaVinci Resolve's built-in DAW-style page, handling narration, ducking, EQ, and compression right inside a video timeline. This course isn't a repeat of that chapter. It's for a genuinely different situation: audio that doesn't start life already living inside a video project.

What Audacity Is

Audacity is a free, open-source, waveform-based audio editor. Unlike Fairlight, it isn't built into a video editor at all — it's a standalone application whose entire job is recording, editing, and cleaning up sound, with no timeline of video frames anywhere in the picture.

Where It Fits: Simpler, Focused Audio Tasks

A podcast episode recorded on its own, a voice narration track being prepped before it ever touches a video project, a quick noise-reduction pass on a single recording — these are audio-only jobs from start to finish. Opening a full non-linear video editor just to clean up one audio file means opening a tool built around syncing sound to picture, for a task that has no picture at all. Audacity is built for exactly this narrower job, and it shows in how much faster it is to get to the actual editing.

What This Course Deliberately Doesn't Cover

Fairlight's own multitrack, video-synced production — mixing dialogue, music, and effects against a picture timeline — stays Video Editing Fundamentals' own territory (its Chapter 7). This course stays honestly scoped to Audacity's own strengths: simpler, standalone audio work, not a competing deep dive into full multitrack film-style mixing. Chapter 6 revisits this boundary directly once multi-track layering comes up.

Why Waveform-Based, Not Timeline-Based

Fairlight's interface is built around clips arranged on tracks, synced to a video timeline running underneath. Audacity's interface is built around a visual waveform — the actual shape of the sound wave, stretched out so silence, clipping, and loud passages are visible at a glance, with selections made directly on that shape rather than by dragging clip blocks. It's a simpler mental model, matched to a simpler set of tasks.

ASPECT	AUDACITY	FAIRLIGHT (IN DAVINCI RESOLVE)
Built for	Standalone audio-only tasks	Audio synced to a video timeline
Core view	Visual waveform	Multitrack timeline, clip-based
Typical use	Podcasts, narration, voice cleanup	Dialogue/music/effects mixed against picture
Where it lives	Its own dedicated application	Built into a video editor

WHERE THIS COURSE CONNECTS TO VIDEO EDITING FUNDAMENTALS

This course's own capstone (Chapter 8) produces a cleaned-up, mixed audio file — the same kind of finished narration or podcast audio that Video Editing Fundamentals' own capstone could plausibly need before final export, matching that course's own honest scope note about leaving deeper audio-only work to this one.

"I ALREADY HAVE A VIDEO EDITOR OPEN" ISN'T A REASON TO SKIP AUDACITY

It's tempting to assume that since DaVinci Resolve already has Fairlight built in, there's never a reason to open a separate audio tool. For audio that starts and ends as audio — a standalone podcast recording, narration prepped before any video exists yet — Audacity's simpler, more focused interface is genuinely faster to work in than opening a full video editor's own DAW page for a job with no video involved at all.

Hands-On Exercises

EXERCISE 1

A friend asks why an entire separate course exists for audio editing when Video Editing Fundamentals already had a full chapter on audio in Fairlight. Using this chapter's own material, explain what distinguishes the two.

View solution

EXERCISE 2

A colleague has DaVinci Resolve open and figures they should just clean up their standalone podcast recording there, in Fairlight, since it's already running. Using this chapter's own warning box, explain why Audacity is likely the better tool for this specific job.

View solution

EXERCISE 3

Someone coming straight from Video Editing Fundamentals opens Audacity for the first time, expects a clip-based timeline like Fairlight's, and is confused to see only a waveform. Using this chapter's own material, explain what's actually different and why.

View solution

CHAPTER 1 QUICK REFERENCE

- **Audacity** is a free, open-source, standalone, waveform-based audio editor — not built into a video editor
- It fits **standalone audio-only tasks**: podcasts, narration, quick cleanup — not video-synced multitrack production
- Fairlight's own **multitrack, video-synced work** stays Video Editing Fundamentals' own territory (its Chapter 7)
- Audacity's **waveform view** is a simpler mental model than a clip-based timeline, matched to simpler tasks
- This course's own capstone produces audio Video Editing Fundamentals' own capstone could plausibly need

Chapter 1 established that Audacity's core view is the waveform itself, not a clip-based timeline. This chapter puts that to use — reading a waveform's shape to spot real problems at a glance, then using that same shape to make accurate selections and trims.

Reading a Waveform: Amplitude Over Time

A waveform plots amplitude (loudness, on the vertical axis) against time (on the horizontal axis). A taller peak means a louder moment; a flat, thin line means quiet or silence. Once this reading habit is second nature, a lot of editing decisions become visible before a single second of playback.

Spotting Clipping at a Glance

Clipping happens when a recording's peaks are pushed past the maximum level the format can represent, flattening the tops of what should be rounded waveform peaks into hard, flat-topped blocks. On the waveform, this looks like the peaks visibly slamming into the top and bottom edges of the track and getting chopped off flat — a shape that's unmistakable once it's been pointed out, and a genuine problem, since clipped audio is often distorted in a way that can't be fully repaired after the fact.

Spotting Silence and Dead Air

Silence and near-silence show up as a thin, flat line with barely any visible height at all — a long pause before someone starts speaking, dead air between podcast segments, or the quiet room tone recorded before a narrator says a word. These flat stretches are exactly where Chapter 3's own noise-reduction work starts, since a clean sample of "what silence sounds like" in this specific recording is what a noise profile is built from.

Basic Selection Techniques

A selection is made by clicking and dragging directly across the waveform — the selected range is highlighted, and any edit (delete, trim, apply an effect) only affects that highlighted range. Double-clicking selects a single continuous sound; zooming in (horizontally, to see more time detail, or vertically, to see quiet passages more clearly) makes precise selections around exact moments far easier than working at a zoomed-out, whole-episode view.

Trimming: Cutting Without a Multitrack Timeline

Trimming in Audacity means selecting the unwanted range — a long pause, a cough, a false start — and deleting it, which closes the gap automatically since there's no video picture underneath that needs to stay in sync. This is simpler than Fairlight's own Ripple/Roll/Slip trim types from Video Editing Fundamentals' own Chapter 4, precisely because there's no second track of picture that a trim could ever knock out of alignment.

WAVEFORM SHAPE	WHAT IT MEANS	WHAT TO DO ABOUT IT
Flat-topped, chopped-off peaks	Clipping — the signal was too loud	Flag for re-recording if possible; avoid pushing gain further
Thin, near-flat line	Silence or near-silence	Trim if it's dead air; sample it for a noise profile (Ch.3) if it's room tone
Tall, rounded peaks well within the track	Healthy signal, good headroom	No action needed
Consistently short, low peaks throughout	Recording was too quiet	Normalize to a target level (Ch.3)

THIS CHAPTER FEEDS DIRECTLY INTO CHAPTER 3

Selecting a clean stretch of silence, by eye, off the waveform is the very first step of noise reduction — Chapter 3 samples that exact selection as a "noise profile" before removing that same background noise from the rest of the recording.

A TALL WAVEFORM ISN'T AUTOMATICALLY A GOOD RECORDING

It's tempting to read "big, tall waveform" as "strong, healthy audio" on sight. Past a certain height, tall usually means clipped, not strong — the flat-topped shape described above is a warning sign, not a sign of a loud, confident recording. A well-recorded track has healthy peaks with visible room left above them, not peaks slamming into the very top of the track.

Hands-On Exercises

EXERCISE 1

A recording's waveform shows peaks that look like flat-topped blocks instead of rounded curves. Using this chapter's own material, identify what's happening and why it matters.

View solution

EXERCISE 2

A beginner proudly points out that their recording's waveform is "nice and tall" across the whole file, and assumes this means it's a strong recording. Using this chapter's own warning box, explain why this isn't necessarily good news.

View solution

EXERCISE 3

Someone coming from Video Editing Fundamentals expects trimming a pause out of a podcast recording to require the same kind of ripple/roll trim-type decisions as Fairlight. Using this chapter's own material, explain why trimming in Audacity is simpler.

View solution

CHAPTER 2 QUICK REFERENCE

- A waveform plots **amplitude over time** — tall peaks mean loud, a flat thin line means quiet or silent
- **Clipping** shows up as flat-topped, chopped peaks — a real problem, not a sign of a strong recording
- **Silence** shows up as a near-flat line — dead air to trim, or room tone worth sampling for Chapter 3's noise profile
- Selections are made by **clicking and dragging** directly on the waveform; zoom in for precision
- Trimming is simpler than Fairlight's Ripple/Roll/Slip, since there's **no video picture** underneath to stay in sync with

Chapter 2 pointed out that a flat, near-silent stretch of waveform is exactly where noise reduction starts. This chapter puts that stretch to work — sampling it as a noise profile, removing that same background noise from the rest of the recording, then bringing the whole file up to a consistent, target volume.

Noise Reduction: Sampling a Profile

Every recording made outside a perfectly silent studio picks up some constant background noise — a fan, room hum, a faint hiss from the microphone itself. Noise reduction starts by selecting a stretch of the recording that contains only that background noise and nothing else — ideally a few seconds of the room tone or pause identified back in Chapter 2 — and using it to build a "noise profile," a fingerprint of exactly what that unwanted sound looks like.

Applying Noise Reduction

Once a profile exists, it's applied across the rest of the selection (often the entire track): Audacity looks for that same fingerprint throughout the recording and reduces it, leaving the wanted signal — the voice, the instrument — largely intact. This only works well because the profile was sampled from real silence in this specific recording; a generic, one-size-fits-all noise profile wouldn't match this room's own actual background sound.

Normalization: Setting a Target Level

Normalization raises or lowers a recording's overall volume so its loudest point sits at a specific target level, without changing the relative loudness of quieter and louder moments within the file. This matters for consistency: two podcast guests recorded on different microphones, or two episodes recorded on different days, can end up sounding evenly matched once both are normalized to the same target rather than one being noticeably quieter than the other.

Why Order Matters: Noise Reduction Before Normalization

Doing these two steps in the wrong order causes a real problem. Normalizing first raises the background noise right along with the wanted signal, since normalization can't tell the difference between the two — it only reads the overall waveform. Removing the noise first, then normalizing the now-cleaner recording, means the target volume is being set based on the signal that's actually wanted, not on a mix of that signal and the noise sitting underneath it.

STEP	WHAT IT DOES	DO THIS FIRST?
Sample a noise profile	Selects a silent stretch as a fingerprint of unwanted background noise	Yes — first
Apply noise reduction	Removes that fingerprint from the rest of the recording	Yes — second
Normalize	Sets overall volume to a target level, without clipping	No — last, after cleanup

THIS CHAPTER FEEDS CHAPTERS 4 AND 5

A recording that's already had its background noise removed and its overall level normalized is a much better starting point for the frequency-shaping work in Chapter 4 (Equalization) and the volume-evening work in Chapter 5 (Compression) — both build directly on top of this chapter's own cleanup pass, not on raw, unprocessed audio.

AGGRESSIVE NOISE REDUCTION CAN SOUND WORSE THAN THE NOISE IT REMOVED

It's tempting to push noise reduction as hard as possible, assuming more removal is always better. Pushed too far, noise reduction introduces its own artifacts — a warbly, "underwater" quality to voices, especially on quieter or breathier sounds. A moderate setting that leaves a faint trace of noise usually sounds far more natural than an aggressive setting that removes the noise completely but damages the voice along with it.

Hands-On Exercises

EXERCISE 1

A colleague normalizes their recording first, then tries to apply noise reduction, and is confused that the background hum still sounds noticeably loud. Using this chapter's own material, explain what went wrong.

View solution

EXERCISE 2

A friend pushes their noise reduction setting to maximum strength "just to be safe," and now their narrator's voice sounds strange and warbly during quiet passages. Using this chapter's own warning box, explain what happened and what to do instead.

View solution

EXERCISE 3

Explain why a noise profile sampled from one recording's own silent stretch works better on that recording than a generic, pre-made noise profile would, using this chapter's own material on how a profile is built.

View solution

CHAPTER 3 QUICK REFERENCE

- A **noise profile** is sampled from a silent stretch of the specific recording being cleaned up
- Noise reduction uses that profile to remove the same **background noise** from the rest of the recording
- **Normalization** sets overall volume to a target level without changing relative loudness within the file
- Always **reduce noise before normalizing** — normalizing first raises the noise along with the signal
- Overly aggressive noise reduction introduces **artifacts** — moderate settings usually sound more natural

Chapter 3 cleaned up background noise and set a consistent overall level. This chapter goes a layer deeper than "how loud" — into "what does it actually sound like" — using equalization (EQ) to boost or cut specific frequency ranges within an already-cleaned recording.

What EQ Actually Does: Frequency, Not Volume

Normalization (Chapter 3) changes a recording's overall volume evenly across every frequency at once. EQ works differently — it targets specific frequency ranges within the recording, boosting some and cutting others, without touching the overall volume the same way. Two recordings can be normalized to the exact same overall level and still sound completely different, because EQ is shaping tone, not loudness.

The Frequency Spectrum in Plain Terms

Human hearing spans roughly 20 Hz to 20,000 Hz (20 kHz), and a voice recording's useful range sits mostly within that span: low frequencies carry bass and body, low-mid frequencies carry warmth and (in excess) muddiness, mid frequencies carry most of a voice's actual clarity and presence, and high frequencies carry crispness, air, and — pushed too far — harshness. Knowing roughly where a problem sound lives on this spectrum is what makes an EQ adjustment a targeted fix rather than a guess.

Reducing Muddiness

A "muddy" or "boxy" voice recording usually has too much energy built up in the low-mid range — often somewhere around 200–500 Hz — which makes speech sound thick, indistinct, or like it was recorded inside a small box. Cutting a few decibels in that range, rather than boosting anything, is usually the fix: removing the buildup that's masking the clearer mid-range frequencies where actual vocal clarity lives.

Taming Harsh Sibilance

Sibilance is the sharp, hissy emphasis on "s" and "sh" sounds that some microphones and voices produce, typically concentrated in a narrow high-frequency band around 5–8 kHz. A targeted cut in that specific band — sometimes with a dedicated de-essing tool rather than a broad EQ cut — reduces the harshness on those specific sounds without dulling the rest of the recording's high-frequency crispness.

Boost vs. Cut: Why Cutting Is Usually Safer

EQ can either boost a frequency range (make it louder relative to the rest) or cut it (make it quieter relative to the rest). Cutting a problem frequency is usually the safer, more natural-sounding move, since it removes something that's actually causing a problem rather than adding artificial energy to a range that may introduce a new one. Boosting is still useful — adding a little presence in the upper-mids can help a recording feel clearer — but it's easier to overdo and make a recording sound thin, harsh, or unnatural.

FREQUENCY RANGE	COMMON PROBLEM	TYPICAL FIX
Low-mid (~200–500 Hz)	Muddiness, "boxy" sound	Cut a few dB
Mid (~1–4 kHz)	Lacking clarity or presence	Small, careful boost
High (~5–8 kHz)	Harsh sibilance on "s"/"sh"	Narrow cut, or de-essing
Very low (below ~80 Hz)	Rumble, handling noise	Cut — rarely useful content in a voice track

THIS CHAPTER FEEDS CHAPTER 5

EQ shapes which frequencies stand out; Chapter 5's compression evens out how consistently loud those frequencies stay over time. The two are complementary — a well-EQed recording still benefits from compression, and compression doesn't fix a muddy or harsh tone the way EQ does.

EQ CAN'T FIX A FUNDAMENTALLY BAD RECORDING

It's tempting to treat EQ as a cure-all for a poorly recorded file — heavily boosting to compensate for a weak, distant, or noisy original recording. Aggressive boosting on a fundamentally thin or noisy signal usually just makes its flaws more audible, not less. EQ works best as a refinement on a recording that's already reasonably solid, not as a rescue for one that fundamentally wasn't recorded well.

Hands-On Exercises

EXERCISE 1

A recording sounds thick and indistinct, almost like it was recorded inside a box. Using this chapter's own material, identify which frequency range is likely the culprit and what to do about it.

View solution

EXERCISE 2

A colleague's narration has a harsh, hissy quality every time they say a word with "s" in it. Using this chapter's own material, explain what's happening and how to address it without dulling the rest of the recording.

View solution

EXERCISE 3

A beginner has a weak, distant-sounding recording and tries to fix it by heavily boosting several frequency ranges with EQ. Using this chapter's own warning box, explain why this likely won't work the way they hope.

View solution

CHAPTER 4 QUICK REFERENCE

- EQ shapes **tone** by targeting specific frequency ranges — unlike normalization, which changes overall volume evenly
- **Muddiness** usually lives in the low-mid range (~200–500 Hz) — cut, don't boost
- **Sibilance** ("s"/"sh" harshness) usually lives around 5–8 kHz — a narrow, targeted cut or de-essing
- **Cutting** a problem frequency is generally safer and more natural-sounding than boosting
- EQ **refines** an already-decent recording — it can't rescue a fundamentally weak or noisy one

Chapter 4 shaped which frequencies stand out at any given moment. This chapter deals with a different axis entirely: how consistently loud a recording stays over time. Compression evens out the gap between a recording's loudest and quietest moments — a technique used constantly in podcasting and voiceover work.

What Compression Actually Does

A raw voice recording naturally varies in loudness — a word spoken with emphasis is louder than one trailing off at the end of a sentence. Compression automatically turns down the loud moments once they cross a set level, narrowing the overall gap between the loudest and quietest parts of the recording. The result is a track that sounds more consistently present throughout, without a listener needing to constantly adjust their own volume.

The Core Controls: Threshold, Ratio, Attack, Release

Threshold sets the loudness level above which compression starts acting — anything below the threshold is left alone. Ratio sets how strongly the compressor turns down whatever crosses that threshold (a higher ratio means more aggressive turning-down). Attack sets how quickly the compressor reacts once the threshold is crossed, and release sets how quickly it lets go once the signal drops back below it. These four controls together shape both how much compression happens and how naturally it's felt.

Why Podcasting and Voiceover Rely on Compression So Heavily

Spoken-word content is especially prone to big loudness swings — a host leaning toward the microphone, a guest speaking more quietly than the host, natural emphasis on certain words. Compression is what keeps a podcast episode listenable in a car, on headphones, or through a phone speaker, without a listener needing to ride the volume knob themselves. This is exactly why it's treated as a near-constant technique in this kind of work, rather than an occasional creative effect.

Makeup Gain: Restoring Volume After Compression

Since compression turns the loud parts down, the overall recording often ends up quieter than it started. Makeup gain raises the whole signal back up afterward, bringing the now-narrower dynamic range up to a comfortable overall level — similar in spirit to Chapter 3's normalization, but applied after compression has already narrowed the gap between loud and quiet.

Over-Compression: The "Squashed" Sound

Pushed too hard — a very low threshold combined with a very high ratio — compression flattens a recording's natural dynamics almost entirely, producing a "squashed," fatiguing sound where every word lands at nearly the same loudness with none of the natural rise and fall of real speech. A more moderate setting still narrows the loudness gap meaningfully while leaving some natural variation intact.

CONTROL	WHAT IT SETS
Threshold	The loudness level above which compression starts acting
Ratio	How strongly the signal is turned down once it crosses the threshold
Attack	How quickly the compressor reacts once the threshold is crossed
Release	How quickly the compressor lets go once the signal drops back below threshold
Makeup gain	Raises overall volume back up after compression narrows the range

THIS CHAPTER FEEDS CHAPTER 6

A compressed narration track holds a more consistent volume, which matters directly once Chapter 6 introduces layering background music underneath it — a voice with big, unpredictable loudness swings is far harder to balance against a music bed than one that's already been evened out here.

HEAVIER COMPRESSION ISN'T AUTOMATICALLY BETTER

It's tempting to assume that more compression always makes a recording sound more polished or professional. Pushed too far, the "squashed" result described above actually sounds worse — flat, lifeless, and fatiguing to listen to over a full episode, even though it's technically more consistent in loudness. A moderate setting that keeps some natural dynamic variation usually sounds far more like a real, professionally produced voice than an aggressively flattened one.

Hands-On Exercises

EXERCISE 1

A podcast host's recording has one guest who speaks much more quietly than the host, making the episode uncomfortable to listen to without constantly adjusting the volume. Using this chapter's own material, explain how compression addresses this.

View solution

EXERCISE 2

After applying compression, a colleague notices their recording sounds noticeably quieter overall than before, even though the loud and quiet parts are now closer together. Using this chapter's own material, explain what step they're missing.

View solution

EXERCISE 3

A beginner sets an extremely low threshold and a very high ratio, assuming maximum compression will sound the most professional. Using this chapter's own warning box, explain what they'll likely hear instead, and why.

View solution

CHAPTER 5 QUICK REFERENCE

- Compression narrows the gap between a recording's **loudest and quietest** moments over time
- **Threshold, ratio, attack, and release** together control how much compression happens and how it's felt
- **Makeup gain** restores overall volume after compression has narrowed the dynamic range
- A near-constant technique in **podcasting and voiceover**, where loudness naturally swings a lot
- Pushed too far, compression produces a **"squashed," fatiguing** sound — moderate settings usually sound more natural

Every chapter so far has worked with a single track of audio — cleaning it up, shaping its tone, evening out its dynamics. This chapter adds a second track: background music, layered underneath the narration that's now been cleaned, EQed, and compressed by everything that came before it.

Adding a Second Track: Background Music

Audacity supports multiple tracks stacked in the same project — narration on one track, background music on another, both playing back together. Importing a music track alongside an existing narration track is straightforward; the real skill in this chapter is what happens next, getting the two tracks to sit together properly rather than fighting each other.

Basic Mixing: Balancing Two Tracks Relative to Each Other

Mixing, at this basic level, means setting each track's own relative volume so the narration stays clearly understandable while the music still contributes atmosphere rather than disappearing entirely. As a rough starting point, background music typically needs to sit noticeably quieter than narration — several decibels down — since its job is supporting the voice, not competing with it.

Ducking: Automatically Lowering Music Under Speech

Ducking automatically lowers the music track's volume specifically during moments when narration is present, then lets it rise back up during pauses or once the narration ends — the same underlying concept Video Editing Fundamentals' own Chapter 7 described in Fairlight, applied here on a standalone, non-video-synced pair of tracks. Even a simple, manually-drawn volume envelope over the music track — turning it down under speech, back up in the gaps — accomplishes the same basic goal without needing an automated ducking tool.

Honest Scope: Where Audacity's Multitrack Stops and Fairlight's Begins

This is genuinely basic multitrack work — two tracks, a simple volume balance, straightforward ducking. Fairlight's own multitrack environment (Video Editing Fundamentals' own Chapter 7) handles far more: many simultaneous tracks, automation curves, and — critically — everything kept in sync against a moving video picture. Audacity was never built to compete with that depth, and this course doesn't pretend otherwise; standalone podcast and voiceover mixing is a genuinely smaller job than full video-synced audio post-production.

ASPECT	AUDACITY (THIS CHAPTER)	FAIRLIGHT (VIDEO EDITING FUNDAMENTALS CH.7)
Typical track count	A small handful (narration + music)	Many, mixed for a full production
Synced to video?	No — standalone audio	Yes — synced to the picture timeline
Ducking	Manual envelope, or basic automated ducking	Fader automation built for video-synced mixing
Intended scope	Podcasts, narration, simple mixes	Full audio post-production for video

THIS CHAPTER FEEDS CHAPTER 7

A properly balanced, ducked mix of narration and music is exactly what gets bounced down to a single exported file in Chapter 7 — export format and platform considerations only matter once there's a finished mix ready to render.

MUSIC SITTING AT THE SAME VOLUME AS NARRATION USUALLY ISN'T A MIX, IT'S A FIGHT

It's tempting to bring music in at a volume that "sounds good on its own" without checking it against the narration. Music competing at a similar volume to speech forces a listener to work harder to follow the words, even if the music sounds perfectly fine in isolation. Checking the balance with both tracks actually playing together — not just previewing the music alone — is the only reliable way to catch this.

Hands-On Exercises

EXERCISE 1

A podcast producer adds background music at a volume that sounds great when previewed by itself, but listeners complain they can't follow the narration once both tracks play together. Using this chapter's own material, explain what went wrong.

View solution

EXERCISE 2

Explain, using this chapter's own material, how ducking in Audacity relates to the ducking Video Editing Fundamentals' own Chapter 7 described in Fairlight — what's the same idea, and what's different about the situation it's applied in?

View solution

EXERCISE 3

A student asks why this course doesn't just teach Audacity's multitrack features as a full replacement for Fairlight. Using this chapter's own honest scope section, explain why that comparison doesn't hold up.

View solution

CHAPTER 6 QUICK REFERENCE

- Audacity supports **multiple tracks** — narration and background music can play back together
- Basic mixing means setting each track's **relative volume** so music supports, rather than competes with, narration
- **Ducking** lowers music under speech and raises it back during pauses — manually or via a basic automated tool
- This is **genuinely basic** multitrack work — Fairlight's own deeper, video-synced multitrack production stays its own territory
- Always check a mix with **both tracks playing together**, not just the music previewed alone

Chapter 6 produced a balanced, ducked mix of narration and music. That mix still lives inside an Audacity project — this chapter turns it into an actual delivered file, choosing the right format and settings for wherever it's actually headed next.

WAV vs. MP3: Lossless vs. Lossy

WAV is an uncompressed, lossless format — it stores the audio data essentially as-is, with no data thrown away, at the cost of a much larger file size. MP3 is a lossy, compressed format — it discards some audio data, chosen to be the least noticeable to human hearing, in exchange for a dramatically smaller file. Neither is universally "better"; the right choice depends entirely on what happens to the file next.

Bitrate Considerations for MP3

MP3's bitrate — how much data is used per second of audio — directly trades file size against audio quality. A low bitrate (around 96–128 kbps) produces a small file with audible quality loss, especially noticeable on music; a higher bitrate (192–320 kbps) sounds close to indistinguishable from the original for most listeners, at a larger file size. Spoken-word content like podcasts tolerates lower bitrates better than music does, since speech has less complex frequency content to preserve.

Choosing an Export Format for Podcasts/Distribution

Most podcast hosting platforms expect MP3, and for good reason: smaller files download and stream faster, and the format is universally supported by every podcast app and browser. A bitrate in the 128–192 kbps range is a common, sensible default for spoken-word podcast content — small enough for fast, reliable streaming, high enough that listeners won't notice compression artifacts.

Exporting Audio to Accompany a Video Edit

When audio produced here is headed into a video project — narration or a cleaned-up dialogue track being handed off to Video Editing Fundamentals' own workflow — WAV is usually the better choice instead. A video editor's own Fairlight page (Video Editing Fundamentals' own Chapter 7) will process that audio further, and starting from an uncompressed file avoids stacking a second round of lossy compression on top of whatever the video's own final export format ends up being.

ASPECT	WAV	MP3
Compression	None — lossless	Lossy — some data discarded
File size	Large	Much smaller
Best for	Handoff into further editing (e.g. a video project)	Final distribution (podcast hosting, streaming)
Typical bitrate choice	Not applicable — uncompressed	128–192 kbps for spoken-word podcasts

THIS CHAPTER FEEDS THE CHAPTER 8 CAPSTONE

The capstone's own finished podcast episode gets exported exactly the way this chapter describes — MP3, at a sensible bitrate, ready for direct upload to a podcast host — closing the loop on everything from cleanup through mixing.

EXPORTING DOESN'T FIX A MIX — IT JUST DELIVERS WHATEVER MIX ALREADY EXISTS

It's tempting to think a higher-quality export format or bitrate can compensate for problems left over from earlier steps — leftover noise, an unbalanced music level, harsh sibilance. Export settings only control how faithfully the existing mix is delivered; they can't retroactively fix mixing or cleanup decisions made in earlier chapters. Any real problems need to be addressed before export, not covered for by picking a "better" format afterward.

Hands-On Exercises

EXERCISE 1

A podcaster exports their episode as an uncompressed WAV file and uploads it directly to their podcast host, then wonders why episodes take so long to download for listeners. Using this chapter's own material, explain what format choice would serve them better and why.

View solution

EXERCISE 2

Someone is handing off a cleaned-up narration track to be imported into a Video Editing Fundamentals project, and exports it as an MP3 at 128 kbps to keep the file small. Using this chapter's own material, explain why WAV would be the better choice here instead.

View solution

EXERCISE 3

A colleague, unhappy with some leftover background hiss in their mix, decides to export at the highest possible bitrate instead of going back to fix the noise reduction. Using this chapter's own warning box, explain why this won't solve their actual problem.

View solution

CHAPTER 7 QUICK REFERENCE

- **WAV** is uncompressed and lossless — larger files, no quality lost
- **MP3** is lossy and compressed — much smaller files, some quality traded away
- MP3 **bitrate** trades file size against quality — 128–192 kbps is a sensible range for spoken-word podcasts
- Export as **MP3** for podcast distribution; export as **WAV** when handing audio off into a video edit
- Export settings can only **deliver** an existing mix — they can't fix cleanup or mixing problems from earlier chapters

Every prior chapter covered one piece of a standalone audio workflow. This capstone takes a raw, two-host podcast recording all the way through that workflow — reading the waveform, cleaning it up, shaping its tone, evening out its dynamics, mixing in intro music, and exporting a finished episode — touching every chapter's own technique in the order a real editing session would actually use them.

Step 1 — Reading the Raw Waveform (Chapter 2)

The raw recording is opened and read visually first: one host's track runs noticeably quieter than the other's, a few flat-topped peaks show where a laugh briefly clipped, and several thin, near-flat stretches mark long pauses and a stretch of room tone recorded before either host started speaking.

Step 2 — Trimming Dead Air (Chapter 2)

The longest pauses and a false start at the very beginning are selected directly on the waveform and deleted, closing the gaps automatically — no ripple/roll trim-type decisions needed, since there's no video picture underneath to keep in sync.

Step 3 — Noise Reduction (Chapter 3)

The stretch of room tone identified in Step 1 is selected and sampled as a noise profile, then applied across the full recording — removing the room's own faint background hum before anything else happens to the file.

Step 4 — Normalizing (Chapter 3)

With the noise already removed, both hosts' tracks are normalized to a consistent target level — done after cleanup, not before, so the target volume reflects the actual wanted signal rather than a mix of signal and noise.

Step 5 — EQ: Clarity and De-Essing (Chapter 4)

A small cut in the low-mid range clears up a slightly boxy quality on one host's microphone, and a narrow cut around the sibilance band tames the other host's harsh "s" sounds — two targeted fixes, not a single blanket adjustment applied to both tracks identically.

Step 6 — Compression and Makeup Gain (Chapter 5)

This is where the loudness gap between the two hosts, spotted back in Step 1, actually gets addressed: a moderate threshold and ratio narrow that gap so neither host constantly needs the listener to reach for the volume knob, followed by makeup gain to bring the now-narrower dynamic range back up to a comfortable overall level.

Step 7 — Layering and Ducking Intro Music (Chapter 6)

A short intro music track is added on its own track, sitting several decibels below the hosts' own dialogue, with a simple ducking pass lowering it further under the opening spoken introduction and letting it fade out once the conversation begins in earnest.

Step 8 — Exporting for the Podcast Host (Chapter 7)

The finished mix is exported as an MP3 at 160 kbps — comfortably within the sensible range for spoken-word content — ready for direct upload to a podcast hosting platform, rather than as a WAV, since this episode's own final destination is a listener's feed, not further editing in another project.

CAPSTONE STEP	CHAPTER IT DRAWS FROM
Step 1 — Reading the raw waveform	Chapter 2
Step 2 — Trimming dead air	Chapter 2
Step 3 — Noise reduction	Chapter 3
Step 4 — Normalizing	Chapter 3
Step 5 — EQ and de-essing	Chapter 4
Step 6 — Compression and makeup gain	Chapter 5
Step 7 — Layering and ducking music	Chapter 6
Step 8 — Exporting for distribution	Chapter 7

IF THIS EPISODE NEEDED TO FEED INTO A VIDEO EDIT INSTEAD

Had this episode been narration for a video rather than a standalone podcast, Step 8 would export a WAV instead of an MP3, handing off to Video Editing Fundamentals' own Fairlight page for further processing — exactly the distinction Chapter 7 drew between distribution and handoff exports.

THIS WORKFLOW IS MEANT TO BECOME A HABIT, NOT A ONE-TIME FIX

It's tempting to treat a cleanup-and-mix pass like this as a one-off rescue for a particularly rough recording. In practice, a consistent podcast or channel benefits from running roughly this same sequence — waveform check, cleanup, EQ, compression, mixing, export — on every episode, not just the ones that sound obviously broken. Consistency across episodes, not just quality within one, is what actually keeps an audience comfortable.

Hands-On Exercises

EXERCISE 1

Explain why this capstone's own Step 3 (noise reduction) happens before Step 4 (normalizing), referencing Chapter 3's own material directly.

View solution

EXERCISE 2

Explain why Step 6's compression pass is specifically what resolves the loudness difference between the two hosts identified back in Step 1, referencing Chapter 5's own material on threshold and ratio.

View solution

EXERCISE 3

Write a short chapter-attribution summary (2-3 sentences) explaining how this capstone's own step-by-step shape compares to OBS Studio's and Video Editing Fundamentals' own capstones, and why all three courses independently arrived at the same kind of shape.

View solution

COURSE COMPLETE

Audio Editing & Production Basics — 8 of 8 chapters complete. This closes out the Audio & Video Production subject: OBS Studio, Video Editing Fundamentals, and Audio Editing & Production Basics are all now complete.

CHAPTER 8 QUICK REFERENCE

- This capstone takes a raw, two-host podcast recording from a **waveform read-through** to a finished, exported episode
- The order mirrors real practice: **read, trim, reduce noise, normalize, EQ, compress, mix in music, export**
- Step 6's compression pass is what actually **resolves the loudness gap** between the two hosts spotted in Step 1
- Export destination decides the format: **MP3 for distribution**, WAV for a handoff into further video editing
- This sequence works best as a **repeatable habit** applied to every episode, not a one-time fix for a rough recording