PyTheory Playground

June 2026

7 min read • 1,557 words

Earlier this spring I wrote a few essays about PyTheory, the music theory library I was stuck on for five years and finally unstuck. The short version: it models tones, scales, chords, and fretboards in Python the way a musician actually thinks about them, it grew into a mini DAW with a NumPy synthesizer inside, and I used it to write an album. It hasn't slowed down since. The library that rendered that album has since learned to transcribe audio, engrave sheet music with LilyPond, tune a guitar in real time, and sync its clock with Ableton.

But everything it can do has lived behind one gate: `pip install pytheory`. For a programmer, that's not a gate at all. For everyone else, it's the whole wall.

So the library has a front door now. It's called PyTheory Playground, it runs in your browser, and there is nothing to install. You pick a chord and watch the fingering appear. You hum a melody and get it back harmonized. You tune your guitar against a strobe. Every result on every page is computed live, on the server, by the same library you'd get from `pip`. The browser is just the doorway.

The Tour

The Guitar Tab page is the soul of the thing, because chord identification is why PyTheory exists at all. Pick a chord and you get a diagram for guitar, twelve-string, ukulele, banjo, mandolin, or bass, in any named tuning (drop D, DADGAD, open G), with a capo wherever you like. But the diagram works in both directions: click the dots and pytheory names whatever you're now fingering.

The question I built the library to answer, what chord is this?, no longer requires a REPL. It requires a mouse. Alternative voicings load up the neck with a click, and a twelve-fret board maps any scale, or the chord's own tones, across the whole neck, every note clickable and audible.

The Tuner listens to your microphone and streams pitch readings twenty times a second: note name, signed cents, a needle that goes green within five cents, and as of this morning a spinning strobe wheel, because pytheory 0.49 shipped its native strobe tuner a few hours after the playground went up. You can tune chromatically, or per-string against any instrument and tuning the site knows. And because the tuner runs through pytheory's tonal systems rather than a hardcoded note table, you can tune to Sa if your music lives there.

Chord Lab is for the theory-curious: voicings, interval structure, pitch-class set theory with Forte numbers, tension scoring, tritone substitutions. My favorite part is the temperament switch. The same chord plays in equal temperament, Pythagorean, meantone, and just intonation, and you can hear the beat frequencies slow and then vanish as the thirds become pure. That's not a recording of the difference. The synthesizer is retuning the actual frequencies on each request.

The Scales page is where the library's reach shows. When I wrote about PyTheory in March it spoke six musical systems. It speaks sixteen now: Western modes, Hindustani rāgas, Arabic maqāmāt, gamelan slendro and pelog, 19-tone equal temperament, Bohlen–Pierce, and more, each with microtonal audio, harmonization, a piano view, and fretboard diagrams. Keys & Progressions has an interactive circle of fifths, borrowed chords, secondary dominants, and a modulation planner that finds pivot chords between any two keys. Songwriter sketches a complete arrangement from a vibe (pop, jazz, lofi, latin) using the library's section engine and a hundred drum grooves, fully editable, rendered as audio, MIDI, or engraved sheet music. And the Tools page holds the party trick: hum a melody at your laptop, and pytheory transcribes it, detects the key, picks a chord for every bar, and plays your own voice back over piano and bass.

> The rest of the toolbox: a chord identifier for raw fret positions, Roman-numeral progression analysis, key detection from note names, a MIDI-to-notation converter, and full audio transcription via YIN pitch tracking into LilyPond, ABC, MusicXML, or tab. Everything downloadable.

Nothing Was Rewritten

Here's the part that was genuinely the most fun to build, and it's the part you can't see.

The obvious way to put a music theory toolbox on the web in 2026 is to write it in JavaScript. There are npm packages for this. Or you compile the Python to WebAssembly, or you quietly reimplement the chord logic client-side "for responsiveness" and hope the two copies never disagree. Every one of those paths ends with two implementations of the same theory, drifting.

The playground does none of that. The frontend is about nineteen hundred lines of vanilla JavaScript, no framework, no build step, no `node_modules`, and it contains no music theory whatsoever. The closest it comes is a twelve-entry lookup table mapping note names to piano keys, so the drawing code knows which key to light up. Everything else, every chord name, every fingering, every frequency, every engraved PDF, comes from one place: pytheory, running on the server, answering plain HTTP requests.

Which means the playground isn't really an app with an API. It's an API wearing an app:

```
$ curl "https://playground.kennethreitz.org/api/tools/identify?frets=x,3,2,0,1,0"
{"name": "C major", "tones": ["E4", "C4", "G3", "E3", "C3"], ...}
```

Every button on the site resolves to a request like that one. The chord diagrams are JSON. The audio is WAV synthesized per-request by NumPy. The sheet music is a PDF engraved by LilyPond on demand.

> LilyPond embeds a full Scheme interpreter, so engraving arbitrary user-supplied source would be remote code execution as a service. The server only engraves LilyPond it generated itself, proven by an HMAC signature minted alongside each conversion. Security as a design constraint, even on a toy.

Even the tuner refuses to do its thinking in the browser: your microphone samples stream to the server, YIN pitch detection runs in Python, and the readings stream back over a WebSocket. The browser draws a needle. That's its whole job.

I keep coming back to why this felt so satisfying, and I think it's this: the playground can't lie. A marketing site describes what a library does; a demo approximates what it does. This thing has no choice but to be accurate, because it's not a representation of pytheory. It is pytheory, with buttons. When the library learns a new trick, the playground learns it by bumping one line in `pyproject.toml`. The strobe tuner went from library release to live feature in the time it took to upgrade a dependency. There is one source of truth, and it's the same one you can `pip install`.

The Last Gate

I've written before about why music theory needs this. It's a notoriously gatekept subject, wrapped in Italian terminology and the silent assumption that you already spent a decade in conservatory, when the underlying ideas are beautiful and not actually that complicated. PyTheory was my attempt to make theory as accessible as HTTP: you shouldn't need a music degree to ask what chord you're playing, any more than you should need to understand TCP/IP to fetch a URL.

But "accessible to anyone who can write Python" is a strange definition of accessible. It serves the programmer-musician, a real and beloved demographic of approximately me. It does nothing for the kid with a ukulele and a question, or the choir director who wants to hear what just intonation actually sounds like, or anyone whose curiosity is musical rather than computational. For them, the install step isn't friction. It's a locked door with a sign that says developers only.

Requests said you shouldn't need to understand the protocol. PyTheory said you shouldn't need the degree. The playground says you shouldn't even need Python. Each one is the same move: find the remaining gate, and remove it.

Built in the Right Hours

The commit log says the playground came together between two and six-twenty in the morning, with Claude doing the typing while I did the wanting, the same division of labor that has powered everything I've shipped this year. When I first wrote about PyTheory, I said it existed for the weird chord shape you find at 2am and don't know what to call. It feels right that the library's house got built in exactly those hours.

And it's my whole stack, top to bottom, which I won't pretend isn't part of the joy. The theory is pytheory. The server is Responder, the framework with a userbase of one, whose user count may now technically be two if you count the playground as its own customer. The styling is borrowed from this site. The analytics are self-hosted on my own machines. Nobody's product roadmap is anywhere in the building. It's infrastructure for one that happens to have a public entrance, which might be my favorite genre of software: built entirely to my own taste, and then the door left open.

I don't know who needs a modulation planner with pivot chords, or microtonal audio for Bohlen–Pierce scales, or a strobe tuner that speaks Hindustani solfège. Statistically, almost nobody. But the marginal cost of sharing it is zero, and somewhere out there is a person who will hum a melody into their laptop, hear it come back harmonized, and feel the little jolt of oh, that's what I was singing. That person doesn't care that it's Python. They shouldn't have to.

The weird chord at 2am finally has a doorbell. Ring it: playground.kennethreitz.org.
