Issue #1: Playback Doesn't Stop On Button Press

When the start/stop button is pressed, playback doesn't stop, and multiple playback loops can occur simultaneously if the user clicks the start button a third time.

I used the clearTimeout() method with unique IDs for each timeout to stop playback on a second button press. I also reworked the visuals of the button so that it was intuitive to use, appearing as a green “Start” button when playback is paused and appearing as a red “Stop” button when playback is running.

Issue #2: Metronome Latency

The metronome audio file is loaded every time before it is played, so there is some latency between the time it is called and the time it plays, messing up the sync between the chords and metronome.

I separated the load and play functions so that the metronome audio file only loads once when the webpage is initialized and is then referenced by the play functions.

Issue #3: Playback Requirements Failsafe

The playback loop requires two conditions to successfully function. The total of the attack, decay, and release times must equal no more than the total duration time of a chord and at least one chord of each of three categories must be selected.

I used if statements to prevent playback if either requirement is not met and output a message to the user detailing what they must do to fulfill the playback requirements. I also made it so chords could not be added or removed from selection during playback.

Issue #4: Missing Chord Tones + Parallel Octaves

When ensuring a voice doesn't move to an already represented pitch so that all chord tones are represented, notes of the same pitch class but different octaves are still considered different notes. Very quickly, two voices end up on the same pitch class an octave apart and then almost indefinitely move in parallel motion. This also means only two of the three chord tones are represented.

I added a condition that ensured a new array made of the newVoicing array elements moduloed 12 (converted to pitch class) did not include the pitch class of the new target tone.

Issue #5: Jumping Voices

Voices always move in the same order (voice 1 always moves first; voice 2 always second; and voice 3 always third). Because of this, voices 2 and 3 often end up moving to less ideal, sometimes distant, notes because the closer tones are already represented.

I developed a function in which the voices move in all 6 possible orders, storing the total distances between the current tones and the target tones, and then choosing the voicing which has the least total distance (difference between current chord tones and target chord tones).

Issue #6: Unlimited Voice Ranges

There is currently no restriction on the range of the pitches. Progressions tend to eventually reach a very high register. With most progressions, this is hardly an issue as it takes a long time to do so, but when select chords, particularly the vii*, are used, the progression very quickly reaches too high of a register.

I think this issue is beyond the scope of my project for now.