*Completed, but still need to be debugged and improved*

**music.init(custom)**

Loads all of the definitions (chords, scales, etc.) based on the western music scale. Takes an optional parameter ‘custom’ which points to an external file (or string maybe) of custom definitions

**music.fretboard(strings,limit,pattern)**

Constructor to make objects describing fretted instruments and the notes of each fret and string. Takes array of string notes (notes of the open strings), fret limit, and fret scale pattern. If scale pattern is not given, it is assumed to be just semitone after semitone, like most fretted instruments. Find notes with *objectname*.[*string*].[*fret*]. (May, in the future, have a method to display via canvas the instrument)

**music.scale(startingnote,pattern)**

Returns an array of notes (i.e. a scale) starting on *startingnote* and following *pattern*. Examples: C major, D minor, F# dorian. This is also used to construct chords. In this case the pattern argument is a chord pattern.

**music.numInterval(note1,note2)**

Finds the numerical interval (that is, distance in semitones) between two notes. Does not return inverse.

**music.interval(note1,note2)**

Returns strings in plain English describing interval between two notes. (e.g. Major Third). Strings returned are from an array, which can of course be customized.

**music.isNote(note)**

Returns true if note, returns false if not. Based on music.chrScale and could possibly in the future take a custom scale as an argument.

*In progress*

**music.chordfind(notes)**

Takes string of notes and finds possibly chords it could comprise. It does not look it up in a library of chords (that would be a stupid way to do it), but rather finds all arrangements of intervals. As of now, it returns the interval between the tonic note and each other note. It assumes each note is the tonic and builds an array of (numerically sorted) intervals based on that assumption. Thus, it returns an array, with an element for each tonic, each containing an array for the intervals relative to that tonic. Next to do is to build in a comparison mechanism to match to library of chord patterns. After that I want to include a mechanism for finding chords that would match if one note were ignored, and returning the result as “Cmaj7 with an added 6th” for example.

**music.parse(notes)**

Takes notes and uses regex to extract notes (based on music.chrScale) and put them into array. Once this is done, all the functions that take notes will take them in any form and just use this function to make them into an array.

*Not started yet*

***fretboardobject.*chordfind(notes)**

Notes is a 2d array describing each fret and string number being pressed. This method returns the chord formed.

***fretboardobject.*fingerfind(chord)**

Takes a chord (array of notes) and finds various fingering positions, favoring ones that involve nearby frets.

***fretboardobject.*draw(frets)**

Draws, via canvas, the instrument described by the object. Optional parameter causes it to draw dots on fingered frets

**music.transposeChords(chords,oldkey,newkey)**

Takes a chord progression (of form: “Fmaj7,D#min9”, etc.) and transposes it from the oldkey into the newkey

**music.transposeChordsBy(chords,steps)**

Takes a chord progression and transposes it *steps* amount of steps. Steps can be an integer from -11 to 11

**music.transposeMelody**

Think you get the idea

**music.transposeMelodyBy**

Think you get the idea

**music.progressionFinder(chords)**

Takes chord progression and returns roman numeral notation.

*Far in the future, if at all*

**music.song(notes)**

Constructor to build objects describing songs. Parameter is the sequence of notes that make up the song. This would require a more absolute format for inputting notes (that is, not just “G” but “G2”)

***songobject*.draw(timing,staff,tempo)**

Draws, via canvas, a staff displaying a song (described by an instance of music.song).

***songobject*.play(timing,staff,tempo)**

Play a song (described by an instance of music.song).

All the functions above to manipulate notes could be applied to a song object