Report of language structure of the input language used for the project, "JavaScript Assembler for an Programming Language"

By Gary Reichard
Advisor: Professor James Heliotis

This report explains the language of the input that JavaScript will compile and the grammar used will be in red text throughout. Each executed program will consist of a musical tune which will have just the melody and not the harmony but certainly on a future to-do list. Each melody will consist of 4 major components:

1) time signature
2) key signature
3) tempo
4) note sequence

melody ::= t_sig k_sig tempo seq { t_sig seq } { k_sig seq } { tempo seq}

This particular audio programming language must contain at least one of each component. Each component will be delimited by a keyword to mark it's beginning and a semi-colon to mark it's end to differentiate each component. An explanation, the rules of the grammar, and an example will be given for each component and some sub-components.

Time signature

The first component of a melody is the **time signature,** which specifies how many beats are to be contained in each bar and which note value is to be given one beat. Simple time signatures consist of two numerals, one stacked above the other, as shown in figure 1. The upper numeral indicates how many such beats there are grouped together in a bar. The lower numeral indicates the note value that represents one beat (the beat unit). Beat_num will represent the upper number and the beat_unit will represent the lower, with both having a precise range of allowable integers to be checked after parsing. The time sequence will be delimited with a "@" keyword to mark it's start and a semi-colon marking it's end.

t_sig ::= "**@**" beat_num beat_unit "**;**"

beat_num ::= **integer**

beat unit ::= **integer**

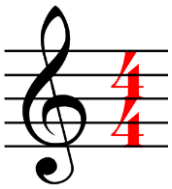

Fig 1 – A Time signature

Key Signature

The second component, **key signature,** is a set of sharp or flat symbols placed together on the staff like in figure 2. A key signature designates notes that are to be played higher or lower than the corresponding natural notes and applies through to the end of the piece or up to the next key signature. The components of the key signature is one of the natural notes (A – G) , a modifier (sharp (#), flat (b), natural (n)), and a key (major or minor). The key signature will be delimited by an exclamation point marking the beginning and a semi-colon marking the end.

key_sig ::= "**!**" natural modifier key "**;**"

natural ::= **A** | **B** | **C** | **D** | **E** | **F** | **G**

modifier ::= **#** | **b** | **n**

key ::= **major** | **minor**

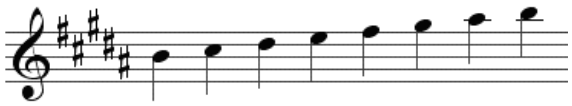

Fig 2 – A key signature example

Tempo

The third component, the **tempo,** is the speed or pace of a given piece and is usually indicated in beats per minute (BPM), indicated in figure 3 above the staff. For a program that just prints the executed audio programming language, this component would just be a written notation, but if the program plays the language using sound, then the tempo can be fine-tuned and the tempo can be modified by the programmer. The tempo is delimited by the percent sign marking the beginning and the semi-colon marking it's end. Each identifier will have it's own range of integers indicating the tempo that would be checked after the parsing.

tempo ::= "**%**" ident integer "**;**"

ident ::= letter { letter }

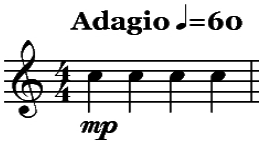

Fig 3 – The tempo Adagio

Note Sequence

The last component contains the note sequence which will be the articulation of this language, containing the notes, rests, durations, and note relationships. Each subcomponent have a complexity of their own and together have infinite combinations sequentially and these subcomponents will discussed further. A note sequence contain notes, slurs, ties, and rests and can be in any order and in any number. They will delimited by an ampersand to mark it's beginning and a semi-colon to mark it's end.

seq ::= "**&**" notes { notes } "**;**"

notes ::= note | slur | tie | rest

Note

The note represents the relative duration and pitch of a sound indicated by a letter name which can be modified by the accidentals, the sharp and flat. A sharp raises a note by a semitone or half-step, and a flat lowers it by the same amount. Each note be start with a carat and all have 4 components, the letter name, a modifier, it's duration and it's octave. A modifier will be mandatory even if it may be implied by the key signature. The duration and octave will be have a small finite set of integers that will be checked after the parsing.

note ::= "**^**" natural modifier duration octave

natural ::= **A | B | C | D | E | F | G**

modifier ::= **# | b | n**

duration ::= **integer**

octave ::= integer

Slur

A slur is a symbol indicating that the notes it embraces are to be played without separation. A slur is denoted with a curved line generally placed over the notes if the stems point downward, and under them if the stems point upwards, as shown in figure 4. This sequence of notes will be delimited by the "*" character to mark the start and end of the slur.

slur ::= "*****" note note { note } "*****"

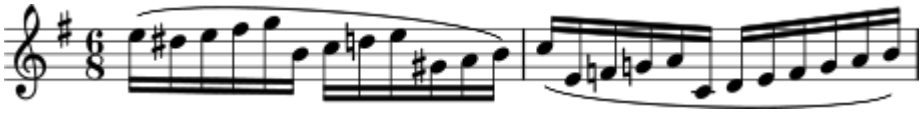

Fig 4 – The slur

Tie

In music notation, a tie is a curved line connecting the heads of two notes of the same pitch and name (figure 5), indicating that they are to be played as a single note with a duration equal to the sum of the individual notes' note values. The tie will be indicated by the dollar sign, the note and the two durations to be added together.

tie ::= "**$**" note duration duration

duration ::= **integer**

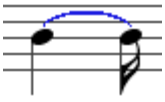

Fig 5 – The tie

Rest

A rest is an interval of silence marked by a symbol indicating the length of silence. Figure 6 shows some such symbols with their equivalent note durations. The rest will be indicated by a plus sign and the silence duration.

rest ::= "✚" duration

duration ::= **integer**

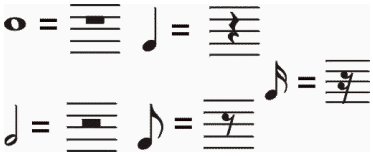

Fig 6 – Rests with equivalent note durations

An example

Using the grammar in this report with the number 16 indicating 16 th notes and the 4 and 5 indicating the octaves, an example code was produced from the music in figure 7.

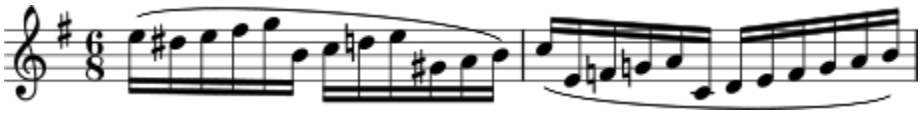

Fig 7 – A code example

```
@6 8; !G n major; %Adagio 64;
&*^E n 16 5 ^D # 16 5 ^E n 16 5 ^F # 16 5 ^G n 16 5
    ^B n 16 4 ^C n 16 5 ^D n 16 5 ^E n 16 5 ^G # 16 4
    ^A n 16 4 ^B n 16 4*
 *^C n 16 5 ^E n 16 4 ^F n 16 4 ^G n 16 4 ^A n 16 4
   ^C n 16 4 ^D n 16 4 ^E n 16 4 ^F n 16 4 ^G n 16 4
   ^A n 16 4 ^B n 16 4*;
```

The Grammar

melody ::= t_sig k_sig tempo seq { t_sig seq } { k_sig seq } { tempo seq}

t_sig ::= "**@**" beat_num beat_unit "**;**"

beat_num ::= **integer**

beat unit ::= **integer**

key_sig ::= "**!**" natural modifier key "**;**"

natural ::= **A** | **B** | **C** | **D** | **E** | **F** | **G**

modifier ::= **#** | **b** | **n**

key ::= **major** | **minor**

tempo ::= "**%**" ident integer "**;**"

ident ::= letter { letter }

seq ::= "**&**" notes { notes } "**;**"

notes ::= note | slur | tie | rest

note ::= "**^**" natural modifier duration octave

natural ::= **A** | **B** | **C** | **D** | **E** | **F** | **G**

modifier ::= **#** | **b** | **n**

duration ::= **integer**

octave ::= integer

slur ::= "*****" note note { note } "*****"

tie ::= "**$**" note duration duration

rest ::= "**+**" duration