# LEADSHEETS

v0.0 2014/07/24

songbook (draft) (CN)

Clemens NIEDERBERGER

https://github.com/cgnieder/leadsheets/

contact@mychemistry.eu

## Table of Contents

# Part I.
# About the Bundle

> I like beautiful melodies telling me terrible things.
>
> *Tom Waits*

## 1. License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

LEADSHEETS requires the bundles l3kernel [The13a] and l3packages [The13b] ro be available. It also needs the package translations [Nie13].

## 2. Background

Over the years I repeatedly wanted to typeset simple leadsheets of songs, *i. e.*, song lyrics and the corresponding chords.[1] This is not too hard with standard LaTeX commands and environments but it is not very convenient… so looking for existing packages is the logical next step and I found two very promising packages: songs [Ham12] and songbook [Rat10]. However, both were note *quite* what I wanted. Just a bit to inflexible in the wrong places, needing tweaking here and there, and so one. On the other hand I had quite some code lying on my hard drive with various attempts of typesetting leadsheets. This bundle of packages is now the attempt to have a clean, documented and customisable version of my code.

1. I also have had the need to typeset leadsheets in the style of the *The Real Book* – but this is a task where other software than LaTeX is far easier.

# Part II.
# The MUSICSYMBOLS Package

> Music expresses that which cannot be said and on which it is impossible to be silent.
>
> *Victor Hugo*

The MUSICSYMBOLS package is a very small package that makes the music symbol font provided by MusiXTEX available as text font and then uses it to define a number of symbols. This redefines the macros `\sharp`, `\flat` and `\natural`. All defined symbold are listed in table 1.

MUSICSYMBOLS defines three further commands, namely `\musix`, `\textmusix` – a font switch and a text font command – and `\musicsymbol`. Those commands are meant for internal use only which is why they're not explained here.

TABLE 1: Symbols defined by MUSICSYMBOLS.

| Command | Symbol | Command | Symbol |
|---|---|---|---|
| `\sharp` | ♯ | `\flat` | ♭ |
| `\doublesharp` | 𝄪 | `\doubleflat` | 𝄫 |
| `\natural` | ♮ | | |
| `\trebleclef` | 𝄞 | `\bassclef` | 𝄢 |
| `\altoclef` | 𝄡 | | |
| `\allabreve` | 𝄵 | `\meterC` | 𝄴 |
| `\wholerest` | 𝄻 | `\halfrest` | 𝄼 |
| `\quarterrest` | 𝄽 | `\eighthrest` | 𝄾 |
| `\sixteenthrest` | 𝄿 | | |

MUSICSYMBOLS also defines a number of macros for denoting bars. Those macros are listed in table 2.

TABLE 2: Bar symbols.

| Macro | Symbol | Macro | Symbol |
|---|---|---|---|
| `\normalbar` | 𝄀 | `\leftrepeat` | 𝄆 |
| `\rightrepeat` | 𝄇 | `\leftrightrepeat` | 𝄈 |
| `\doublebar` | 𝄁 | `\stopbar` | 𝄂 |

# Part III.
# The CHORDNAMES Package

> I never practice my guitar – from time to time I just open the case and throw in a piece of raw meat.
>
> *Wes Montgomery*

## 3. The \chordname Command

CHORDNAMES provides the command \chordname{⟨*chord*⟩} for convenient typesetting of chords:

```
1 \chordname{Bb7(#9)} \chordname{Bbb6}
2 \chordname{C#7(b9)} \chordname{C##13}
```

B♭$^{7(\sharp 9)}$ B𝄫$^{6}$ C♯$^{7(\flat 9)}$ C𝄪$^{13}$

\chordname{⟨*chord*⟩}
Typesetting chords. Inside the argument every # will be replaced by ♯ and every b is replaced with ♭. Numerals and parentheses are typeset as superscripts. Everything between parentheses is always a superscript: \chordname{F#7(#11)} F♯$^{7(\sharp 11)}$.

There are several token lists that are treated specially inside \chordname:

^ – This token is replaced by \textsuperscript.

ma – The symbol for major chords. Per default this is empty. \chordname{Gma} G.

mi – The symbol for minor chords. Per default this is m. \chordname{Gmi} Gm.

o – The symbol for diminished chords. Per default this is \textsuperscript{o}. \chordname{Go} G$^{\mathrm{o}}$.

+ – The symbol for augmented chords. Per default this is \textsuperscript{+}. \chordname{G+} G$^{+}$.

/o – The symbol for half diminished chords. Per default this is \textsuperscript{\o{}}. \chordname{G/o} G$^{\emptyset}$.

# – The “sharp” symbol. Per default this is \sharp. \chordname{F#} F♯.

## – The “double sharp” symbol. Per default this is \doublesharp. \chordname{F##} F𝄪.

b – The “flat” symbol. Per default this is \flat. \chordname{Eb} E♭.

`bb` – The "double flat" symbol. Per default this is `\doubleflat`. `\chordname{Ebb}` E𝄫.

`b#` – Cancelling flat/sharp combination: this is removed.

`#b` – Cancelling sharp/flat combination: this is removed.

`add` – This is superscripted: `\chordname{Gadd9}` $G^{add9}$.

`sus` – This is superscripted: `\chordname{Gsus4}` $G^{sus4}$.

`dim` – This is superscripted: `\chordname{Gdim5}` $G^{dim5}$.

`maj7` – This is superscripted: `\chordname{Gmaj7}` $G^{maj7}$.

`maj9` – This is superscripted: `\chordname{Gmaj9}` $G^{maj9}$.

How these token lists are treated depends on optional settings:

```
\setchordnames{
  major-seven = $\Delta$ ,
  major-nine  = $\Delta$\textsuperscript{9}
}
\chordname{Gmaj7} \chordname{Gmaj9}
\chordname{G^6} \chordname{G6}
\chordname{G7^#5}
```

---

GΔ $G\Delta^9$ $G^6$ $G^6$ $G^{7\sharp5}$

## 4. Options

Options are set with the command

`\setchordnames{`⟨*options*⟩`}`
where ⟨*options*⟩ is a comma separated list of keyval options.

The options allow detailed customization of how chords are printed. It doesn't change the input syntax.

`format` = {⟨*code*⟩} (initially empty)
Code inserted before a chord within the same group. Can be used for special formatting of the chords, with `\sffamily`, say.

`sharp` = {⟨*code*⟩} Default: `\sharp`
The sharp symbol.

`flat` = {⟨*code*⟩} Default: `\flat`
The flat symbol.

`double-sharp` = {⟨*code*⟩} Default: `\doublesharp`
The double sharp symbol.

`double-flat` = {⟨*code*⟩} Default: `\doubleflat`
The double flat symbol.

`aug` = {⟨*code*⟩} Default: `+`
The augmented symbol.

`half-dim` = {⟨*code*⟩} Default: `\o{}`
The half-diminished symbol.

`full-dim` = {⟨*code*⟩} Default: `o`
The diminished symbol.

`dim` = {⟨*code*⟩} Default: `\textsuperscript{dim}`
The token list `dim`.

`add` = {⟨*code*⟩} Default: `\textsuperscript{add}`
The token list `add`.

`sus` = {⟨*code*⟩} Default: `\textsuperscript{sus}`
The token list `sus`.

`major` = {⟨*code*⟩} (initially empty)
The token list `ma`.

`minor` = {⟨*code*⟩} Default: `m`
The token list `mi`.

`major-seven` = {⟨*code*⟩} Default: `\textsuperscript{maj7}`
The token list `maj7`.

`major-nine` = {⟨*code*⟩} Default: `\textsuperscript{maj9}`
The token list `maj9`.

# Part IV.
# The LEADSHEETS Package

> I don't care about the rules. If I don't break the rules at least ten times every song then I'm not doing my job.
>
> *Jeff Beck*

## 5. Background

The LEADSHEETS package allows for easy creation of leadsheets but it also can be used to create complete songbooks. The basic idea is that songs are typeset in the `song` environment. Each song gets a number of properties (see section 6.4) that determine how the title of the song is typeset. For the typesetting of the titles a template mechansim is used (see section 12). Songs can also be tagged. The tags then allow to typeset only songs matching a list of tags that is specified via an option.

## 6. The song Environment

`\begin{song}[`⟨*options*⟩`]{`⟨*properties*⟩`}`

The main environment used to typeset songs.

### 6.1. A First Example

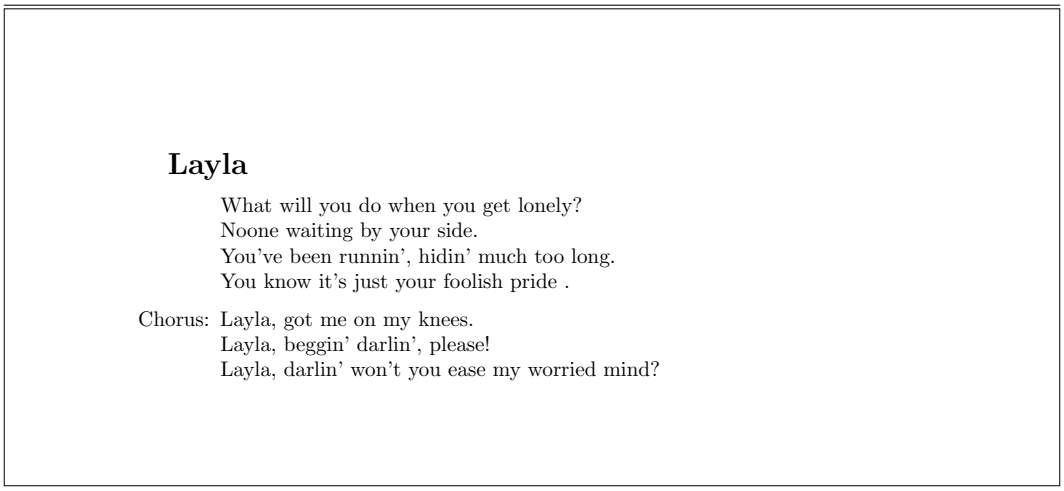
First let's take a look at an example:

```
\documentclass{article}
\usepackage{leadsheets}
\begin{document}

\begin{song}{title=Layla,composer={Eric Clapton and Jim
      Gordon},tags={clapton,unplugged,r&b}}
\begin{verse}
  What will you do when you get lonely? \\
  Noone waiting by your side. \\
  You've been runnin', hidin' much too long. \\
  You know it's just your foolish pride .
\end{verse}
\begin{chorus}
  Layla, got me on my knees. \\
```

```
  Layla, beggin' darlin', please! \\
  Layla, darlin' won't you ease my worried mind?
\end{chorus}
\end{song}

\end{document}
```




Per default the songtitle simply is a `\section*` without any other additions. This is the songtitle template "minimal", see section 12 for more details on those templates and how to create your own.

## 6.2. Using the song Environment

Inside the `song` environment a number of additional environments are used to specify the different parts of a song. They all are basically the same kind of environment, namely an `itemize` environment internally where the only `\item` has the name of the environment as option. The `verse` environment is a little bit different since verses can be numbered. If they are then each usage of `verse` inside `song` will step a vers number and print it (as option to the internal `\item`).

`\begin{verse}[`⟨*options*⟩`]`
An environment for specifying the verses of a song.

`\begin{chorus}[`⟨*options*⟩`]`
An environment for specifying the chorus of a song.
This is the same as `\begin{verse}[type=chorus,`⟨*options*⟩`]`.

`\begin{intro}[`⟨*options*⟩`]`
An environment for specifying the intro of a song.
This is the same as `\begin{verse}[type=intro,`⟨*options*⟩`]`.

`\begin{interlude}[`⟨*options*⟩`]`
An environment for specifying an interlude of a song.
This is the same as `\begin{verse}[type=interlude,`⟨*options*⟩`]`.

`\begin{bridge}[`⟨*bridge*⟩`]`
An environment for specifying a bridge of a song.
This is the same as `\begin{verse}[type=bridge,`⟨*options*⟩`]`.

These environments and their options are described in more detail in sections 7 and 10.

## 6.3. Options

The options to the `song` environment are the same as the general options of LEADSHEETS. This means you can set the following options either local to a song or global for the whole document with this command:

`\setleadsheets{`⟨*options*⟩`}`
Setup command for LEADSHEETS.

Although I used the word "global" above *all options are local to the current scope*!

`title-template` = `{`⟨*template name*⟩`}` Default: `minimal`
The songtitle template, see section 12 for details.

`chord-cs` = `{`⟨*cs*⟩`}` Default: `\chordname`
The command that is used to parse the chords. See section 8 for details. ⟨*cs*⟩ needs to be a command that takes a mandatory argument.

`song-format` = `{`⟨*TEX code*⟩`}` (initially empty)
⟨*TEX code*⟩ is inserted *before* the song title at the beginning of the `song` environment.

`text-format` = `{`⟨*TEX code*⟩`}` (initially empty)
⟨*TEX code*⟩ is inserted *after* the song title at the beginning of the `song` environment.

`numbered` = `true|false` Default: `false`
Determines wether verses are numbered or not.

`print-tags` = `{`⟨*list of tags*⟩`}`
A comma separated list of tags. When specified a song will only be printed if it is tagged with at least one of the tags in ⟨*list of tags*⟩.

`obey-lines` = `true|false` Default: `false`
An experimental option. Use at your own risk! If set to `true` then inside the `verse` like environments end-of-lines will be obeyed and start a new line. This comes with a price when using chords, see section 8.3.

`bar-shortcuts` = `true|false` Default: `false`
Makes the characters `:` and `|` active inside the `song` environment. See sections 6.5 and 11 for more details.

## 6.4. Song Properties

Songs can have a number of properties which basically are used in songtitle templates (see section 12). One specific property, `tags`, plays a different role, though.

`title` = {⟨*title*⟩}
This is the main title of the song.

`subtitle` = {⟨*subtitle*⟩}
A subtitle.

`short-title` = {⟨*short song title*⟩}
A short title (may be useful in a template that writes the titles in `\section`s for a version to be used in the table of contents).

`sort-title` = {⟨*song title*⟩}
If not set explicitly this property holds the same value as `title`.

`sort-short-title` = {⟨*short song title*⟩}
If not set explicitly this property holds the same value as `short-title`.

`composer` = {⟨*composer*⟩}
The composer of the song. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-composer` = {⟨*composer*⟩}
If not set explicitly this property holds the same value as `composer`.

`lyrics` = {⟨*writer*⟩}
Whoever wrote the lyrics if different from the composer. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-lyrics` = {⟨*writer*⟩}
If not set explicitly this property holds the same value as `writer`.

`arr` = {⟨*arranger*⟩}
Whoever arranged the song. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-arr` = {⟨*arranger*⟩}
If not set explicitly this property holds the same value as `arr`.

`band` = {⟨*band*⟩}
The band who plays or played the song.

`sort-band` = {⟨*band*⟩}
If not set explicitly this property holds the same value as `band`.

`interpret` = {⟨*interpret*⟩}
The interpret of the song. As of now this accepts an arbitrary entry but maybe this will not be supported any more when indexing will be implemented. No promises.

`sort-interpret` = {⟨*interpret*⟩}
If not set explicitly this property holds the same value as `interpret`.

`genre` = {⟨*genre*⟩}
The genre of the song.

`key` = {⟨*key*⟩}
The key of the song. This property will be used when transposing will be implemented, see section 9.

`tempo` = {⟨*tempo*⟩}
The tempo of the song.

`tags` = {⟨*tags*⟩}
A comma separated list of tags. Those tags play a role for the option `print-tags`. When that option is used a song is only printed if it has at least one of the tags specified in the option.

There are two more properties, `counter` and `ID` that cannot be set but are set automatically for each song. The `counter` simply holds the number of the current song starting from `1` for the first song. The `ID` currently always ia `song`⟨*counter*⟩ where ⟨*counter*⟩ is the current `counter` value.

In principle all properties can get list of entries where items are separated with ␣`and`␣. Of course this doesn't make sense for each property – a song does only have one title. But a song can very well have more than one composer: think of the Beatles where most songs are written by Paul McCartney and John Lennon.[2]

It is possibly to define further such properties. For details see section 12.3.1.

## 6.5. Special Characters

Inside the `song` environment several characters don't have their usual catecory codes:

`^` – category code 13 (active). It is a shortcut for the `\chord` command.

`_` – category code 13 (active). It is a shortcut for the `\writechord` command.

`|` – category code 13 (active). Used for typesetting bars.

`:` – category code 13 (active). Used for typesetting bars.

`#` – category code 12 (other). Used fo chord names.

Actually the characters `|` and `:` are *not* changed per default. In order to do that you have to use the option `bar-shortcuts`.

For details on the usage of the characters `|` and `:` see section 11. The usage of chords is explained in section 8.

2. This is not quite true: most songs were written either by Paul or John but legally usually both are the composers.

# 7. The verse Environment

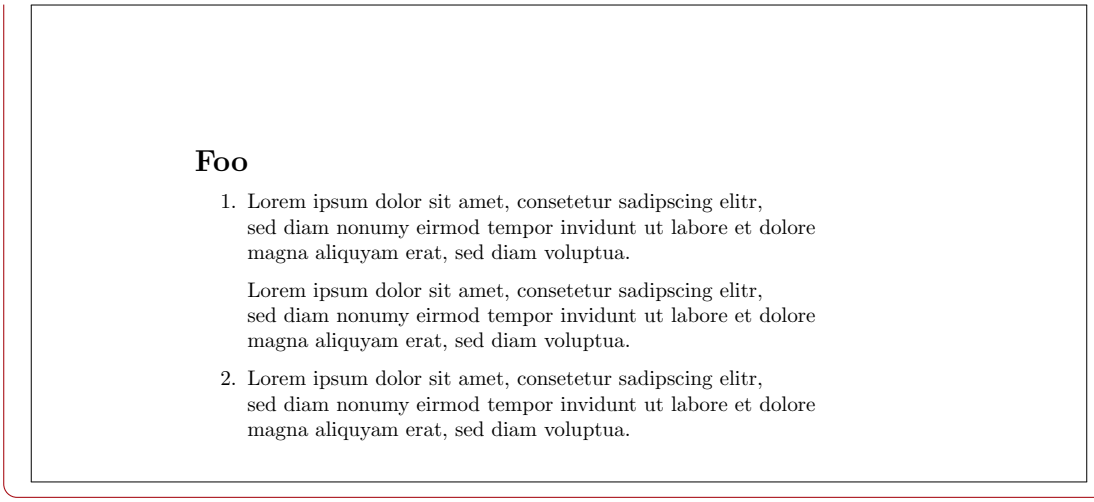
`\begin{verse}[`⟨*options*⟩`]`
An environment for specifying the verses of a song.

`\begin{verse*}[`⟨*options*⟩`]`
The same as the `verse` environment but will always be unnumbered regardless of any option settings.

```
\documentclass{article}
\usepackage{leadsheets}
\begin{document}

\begin{song}[numbered]{title=Foo}
\begin{verse}
  Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
  sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
  magna aliquyam erat, sed diam voluptua.
\end{verse}
\begin{verse*}
  Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
  sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
  magna aliquyam erat, sed diam voluptua.
\end{verse*}
\begin{verse}
  Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
  sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
  magna aliquyam erat, sed diam voluptua.
\end{verse}
\end{song}

\end{document}
```




## 7.1. Options

The verse environment and all related environments have the following option:

format = {⟨*code*⟩} (initially empty)
⟨*code*⟩ is inserted at the beginning of the environment and can thus be used to add formatting, *e. g.*, format = {\itshape}.

label-format = {⟨*code*⟩} (initially empty)
The same for the environment labels.

This can be used either locally, *i. e.*, as option to the corresponding environment, or for all environments of the same type using the setup command using the following syntax:

\setleadsheets{⟨*env name*⟩/format = ⟨*code*⟩}

```
\begin{verse}[format=\itshape]
  Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
  sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
  magna aliquyam erat, sed diam voluptua.
\end{verse}
```

---

*Lorem ipsum dolor sit amet, consetetur sadipscing elitr,*
*sed diam nonumy eirmod tempor invidunt ut labore et dolore*
*magna aliquyam erat, sed diam voluptua.*

It is also possible so set the formatting for all related environments at once:

`verses-format` = {⟨*code*⟩} (initially empty)
Sets the format for all `verse` like environments.

`verses-label-format` = {⟨*code*⟩} (initially empty)
Sets the label format for all `verse` like environments.

Both options are overwritten if the options for a specific environment are set. That is, if you want all environments to have italic shape except for choruses, then you could do

```
\setleadsheets{
  verses-format = \itshape ,
  chorus/format =
}
```

# 8. Placing Chords

## 8.1. The Commands

Inside the `song` environment the characters ^ and _ are active characters.[3] ^ is a shortcut for the command `\chord`, _ is a shortcut for `\writechord`. Those commands have the following functions:

`\chord*-{`⟨*chord*⟩`}`⟨*text*⟩␣
Places ⟨*chord*⟩ centered above ⟨*text*⟩. The starred version gobbles the trailing space while the unstarred version does not. Like the star the dash is optional. It sets the option `smash-next-chord` to `true`. ⟨*text*⟩ may be empty but the trailing space *must* be there. If ⟨*text*⟩ is empty then the chord is place above some horizontal space which can be set with the option `empty-chord-dim`.

`\writechord{`⟨*chord*⟩`}`
This command transforms the chord according to the options `transpose` and `enharmonic` before printing it. This command can/should be used for placing chords inline or for typesetting the `key` property in a template. The command is used by `\chord` for the actual printing.

```
Text \chord{E7}text \chord*{B7}lon ger
   text
```

$E^7$ $B^7$
Text text longer text

3. There are more characters with a special function, see section 6.5

## 8.2. Usage

Note that per default the width of a chord is not ignored:

```
text \chord{Gbmi7(b5)}text text
```

G♭m$^{7(♭5)}$
text text text

However, there is an option which sets the width of a chord to zero:

`smash-chords` = `true`|`false` Default: `false`

If set to true the width of the chords set with `\chord` is set to zero.

`smash-next-chord` = `true`|`false` Default: `false`

If set to true the width of the next chord set with `\chord` is set to zero.

```
\setleadsheets{smash-next-chord=true}
text \chord{Gbmi7(b5)}text text \par
text \chord{Gbmi7(b5)}text text \par
\setleadsheets{smash-chords=true}
text \chord{Gbmi7(b5)}text text \par
text \chord{Gbmi7(b5)}text text
```

G♭m$^{7(♭5)}$
text text text
G♭m$^{7(♭5)}$
text text text
G♭m$^{7(♭5)}$
text text text
G♭m$^{7(♭5)}$
text text text

Before we forget:

`empty-chord-dim` = {⟨*dim*⟩} Default: `1em`

The horizontal space that is inserted if the ⟨*text*⟩ argument of `\chord` is empty.

While `\chord` is available in the whole document the ^ syntax is – as mentioned before – only available inside of the `song` environment.

```
\documentclass{article}
\usepackage{leadsheets}
\begin{document}

\begin{song}{title=Layla,composer={Eric Clapton and Jim
      Gordon},tags={clapton,unplugged,r&b}}
\begin{verse}
  ^{C#mi7} What will you do when you get ^*{G#7}lone ly? \\
  ^{C#mi7} Noone ^*{C}wai ting ^{D}by your ^{E}side. ^{E7} \\
  ^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. \\
```

```
  ^{F#mi} You know it's ^{B}just your foolish ^{E}pride .
\end{verse}
\begin{chorus}
  ^*{A}Lay ---^-{Dmi7}la, \quad ^{Bb} ^{C}got me on my knees. \\
  Lay^-{Dmi7}la, \quad ^{Bb} ^*{C}beg gin' darlin', ^{Dmi7}please, Layla. \\
  Darlin' won't you ease my worried ^{Dmi7}mind? ^{Bb} ^{C}
\end{chorus}
\end{song}

\end{document}
```

**Layla**

C♯m7 G♯7
What will you do when you get lonely?
C♯m7 C D E E7
Noone waiting by your side.
F♯m B E A
You've been runnin', hidin' much too long.
F♯m B E
You know it's just your foolish pride .

Chorus: A Dm7 B♭ C
Lay—la, got me on my knees.
Dm7 B♭ C Dm7
Layla, beggin' darlin', please, Layla.
Dm7 B♭ C
Darlin' won't you ease my worried mind?

You've probably noticed: chords are printed with `\chordname` in the default setting. You can change this with the option `chord-cs`. If you do then keep in mind that the input syntax will also change.

## 8.3. Caveat

If you use `obey-lines = {true}` you have to be careful when you place chords. If you place a chord over the last word in a line

```
^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long.
```

then the end of line that is used as the mandatory space argument for `\chord` won't be recognized as an end of line. Even worse: at the end of a `verse` like environment this may cause non-obvious errors. So in these cases you should always insert an explicit space by one of the following methods:

```
^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. {}
^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. \empty
^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. \relax
```

## 9. Transposing

Provided a song has the property `key` *and* the key is given as one of the twelve "usual" keys, *i. e.*, one of the keys given in table 3, the chords of a song can be transposed.

`transpose` = `{`⟨*number*⟩`}`
Transposes the chords of a song by ⟨*number*⟩ of semitones. ⟨*number*⟩ can be a negative number, then the chords are transposed down.

`enharmonic` = `sharp|flat`
Suppose you transpose a song in the key of E down a semitone. LEADSHEETS will then transpose to the key of E♭. It always chooses the key whose signature has less accidentals. You can force a variant, though, by using this option. With `enharmonic = {sharp}` LEADSHEETS would have chosen D♯ instead of E♭.

The transposing mechanism relies on the CHORDNAMES input syntax which means that if you change `chord-cs` horrible things may happen. *So don't change `chord-cs` and use `transpose` at the same time!*

TABLE 3: Allowed keys for the `key` property.

| Key | Input | Key | Input | Key | Input | Key | Input | Key | Input | Key | Input |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C | `C` | | | C | `Cma` | | | Am | `Ami` | | |
| G | `G` | F | `F` | G | `Gma` | F | `Fma` | Em | `Emi` | Dm | `Dmi` |
| D | `D` | B♭ | `Bb` | D | `Dma` | B♭ | `Bbma` | Bm | `Bmi` | Gm | `Gmi` |
| A | `A` | E♭ | `Eb` | A | `Ama` | E♭ | `Ebma` | F♯m | `F#mi` | Cm | `Cmi` |
| E | `E` | A♭ | `Ab` | E | `Ema` | A♭ | `Abma` | C♯m | `C#mi` | Fm | `Fmi` |
| B | `B` | D♭ | `Db` | B | `Bma` | D♭ | `Dbma` | G♯m | `G#mi` | F♭m | `Fbmi` |
| F♯ | `F#` | G♭ | `Gb` | F♯ | `F#ma` | G♭ | `Gbma` | D♯m | `D#mi` | E♭m | `Ebmi` |

```
\documentclass{article}
\usepackage{leadsheets}
\begin{document}

\begin{song}[transpose=2]{
```

```
    title=Layla,
    composer={Eric Clapton and Jim Gordon},
    tags={clapton,unplugged,r&b},
    key = Dmi
  }
\begin{verse}
  ^{C#mi7} What will you do when you get ^*{G#7}lone ly? \\
  ^{C#mi7} Noone ^*{C}wai ting ^{D}by your ^{E}side. ^{E7} \\
  ^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. \\
  ^{F#mi} You know it's ^{B}just your foolish ^{E}pride .
\end{verse}
\begin{chorus}
  ^*{A}Lay ---^-{Dmi7}la, \quad ^{Bb} ^{C}got me on my knees. \\
  Lay^-{Dmi7}la, \quad ^{Bb} ^*{C}beg gin' darlin', ^{Dmi7}please, Layla. \\
  Darlin' won't you ease my worried ^{Dmi7}mind? ^{Bb} ^{C}
\end{chorus}
\end{song}

\end{document}
```

**Layla**

D♯m$^7$ A♯$^7$
What will you do when you get lonely?
D♯m$^7$ D E F♯ F♯$^7$
Noone waiting by your side.
G♯m B F♯ B
You've been runnin', hidin' much too long.
G♯m B F♯
You know it's just your foolish pride .

B Em$^7$ C D
Chorus: Lay—la, got me on my knees.
Em$^7$ C D Em$^7$
Layla, beggin' darlin', please, Layla.
Em$^7$ C D
Darlin' won't you ease my worried mind?

## 10. Other verse-like Environments

Songs can have lots of different kinds of parts: verses, choruses, bridges, intros, outros, and so on. Typographically they're all more or less the same, at least for the purpose of this package. This means we'd ideally have environments for all of these parts with a distinct name in order to get a clean source. At the same time these environments should all behave basically the same. This is what the environments described in the following sections are for.

## 10.1. Available Environments

`\begin{chorus}[`⟨*options*⟩`]`
An environment for specifying the chorus of a song.

`\begin{chorus*}[`⟨*options*⟩`]`
The same as `chorus` but does not display the label.

`\begin{intro}[`⟨*options*⟩`]`
An environment for specifying the intro of a song.

`\begin{intro*}[`⟨*options*⟩`]`
The same as `intro` but does not display the label.

`\begin{interlude}[`⟨*options*⟩`]`
An environment for specifying an interlude of a song.

`\begin{bridge}[`⟨*bridge*⟩`]`
An environment for specifying a bridge of a song.

`\begin{info}[`⟨*options*⟩`]`
An environment for specifying arbitrary information. This environment has no label.

`\begin{solo}[`⟨*options*⟩`]`
An environment for specifying a solo to a song.

`\begin{solo*}[`⟨*options*⟩`]`
The same as `solo` but does not display the label.

## 10.2. Own verse-like Environments

All environments mentioned in the previous section were defined with this command:

`\newversetype*{`⟨*environment name*⟩`}{`⟨*displayed name*⟩`}`
This defines an environment ⟨*environment name*⟩ with the text ⟨*displayed name*⟩ at the start of the environment left to the text. The starred version defines an environment which hides the displayed name. The environment also defines a translation string (see section 13) `leadsheets/`⟨*environment name*⟩ with ⟨*displayed name*⟩ both as translation fallback and as English translation.

As mentioned before in section 7.1 all environments defined this way have the options `format` and `label-format`. They also have another option:

`after-label` = `{`⟨*code*⟩`}` Default: `:`
⟨*code*⟩ is inserted in the label after the label text.

There's also a general option for setting the default of the `after-label` for all environments:

`verses-after-label` = `{`⟨*code*⟩`}` Default: `:`
Defeault ⟨*code*⟩ that is inserted in the label after the label text of `verse` like environments.

Let's summarize: the label text of these environments is built of three items in the following order:

1. The ⟨*code*⟩ set with the corresponding `label-format` option.
2. The label text as defined as second argument to `\newversetype` or as declared through the corresponding translation.
3. The ⟨*code*⟩ set with the corresponding `after-label` option.

```
\newversetype{foo}{Foo}
\setleadsheets{
  foo/label-format = \bfseries ,
  foo/after-label  = ~$\Rightarrow$
}
\begin{foo}
  Lorem ipsum dolor sit amet, consetetur sadipscing elitr,\\
  sed diam nonumy eirmod tempor invidunt ut labore et dolore\\
  magna aliquyam erat, sed diam voluptua.
\end{foo}
```

---

**Foo** ⇒ Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et dolore
magna aliquyam erat, sed diam voluptua.

## 11. Typesetting Bars

Sometimes it can be useful to typeset the chord scheme of a song. Then one should be able to indicate start and beginnings of bars, maybe indicate repeats and so on. While this is obviously possible with the macros provided by the MUSICSYMBOLS package listed in table 2 it may be more convenient to have a shorter syntax. This is why inside the `song` environment some characters can be made (or are) active (see section 6.5). For the typesetting of bars this are the characters `:` nor `|`. Per default they are not active, though. If you want to use the shortcut syntax you have to use the option `bar-shortcuts`. Here's a short example that emulates the behaviour by setting the characters active explicitly:

```
\catcode`|=\active
\catcode`:=\active
|: repeat | this :|: and | this :| \par
```

```
| this | part | ends | here || \par
| the | song | is over | now |||
```

---

|: repeat | this :||: and | this :|
| this | part | ends | here ||
| the | song | is over | now |

All possibly combinations that have a special definition are shown in the example above. The replacements that are done internally are these:

`|` – `\normalbar\space` (the space is there because otherwise it eats following spaces which would be annoying)

`|:` – `\leftrepeat`

`:|` – `\rightrepeat`

`:|:` – `\leftrightrepeat`

`||` – `\doublebar`

`|||` – `\stopbar`

# 12. Title Templates

## 12.1. Background

The titles of songs set with the `song` environment are displayed according to the chosen title template. It is chosen through the option `title-template` which can be set with `\setleadsheets` or as option to a specific `song` environment. LEADSHEETS provides few predefined templates and an easy mechansim to define own templates.

## 12.2. Existing Templates

Currently LEADSHEETS provides two templates:

**minimal** This only typesettes the song title in a `\section*`.

**tabular** This typesets the song title in a `\section` and prints some song properties in a `tabular` below it. This template needs the array [MCo8] package loaded.

## 12.3. Own Templates

### 12.3.1. The Principle and Available Commands

The principle is pretty straight forward: templates are defined with the following command:

`\definesongtitletemplate{`⟨*name*⟩`}{`⟨*code*⟩`}`
This defines the template ⟨*name*⟩.

Inside of ⟨*code*⟩ any code can be used. The idea is that you use the commands presented below to insert song properties where you want them.

First there are two commands related to defining new properties:

`\definesongproperty{`⟨*property*⟩`}`
Defines a new property ⟨*property*⟩. All existing properties have been defined this way. The command can only be used in the preamble.

`\copysongproperty{`⟨*from*⟩`}{`⟨*to*⟩`}`
Copies the values of property ⟨*from*⟩ to property ⟨*to*⟩ if property ⟨*to*⟩ has not been set but property ⟨*from*⟩ has been. For example all `sort-`⟨*property*⟩ properties have been treated this way so they have the ⟨*property*⟩ value as fallback. The command can only be used in the preamble.

Then there are a number of commands related to retrieving and using the values of properties. All these commands only make sense inside a template definition (see section 12). Some of the commands are expandable which means they can be used in an `\edef` like context, *i. e.*, they are also suitable for writing the property values to the table of contents or other auxiliary files.

* `\songproperty{`⟨*property*⟩`}`
Retrieves property ⟨*property*⟩.

`\printsongpropertylist[`⟨*code*⟩`]{`⟨*property*⟩`}{`⟨*between two*⟩`}{`⟨*between more*⟩`}{`⟨*between last two*⟩`}` Default: `\@firstofone`
Prints a property list ⟨*property*⟩ separated with ⟨*between two*⟩ if the list contains only two items and separated with ⟨*between more*⟩ and ⟨*between last two*⟩ if the list contains more than two items. ⟨*code*⟩ is placed directly in front of each item and items are surrounded with braces which means that the last token in ⟨*code*⟩ may be a macro with a mandatory argument.

`\usesongpropertylist[`⟨*code*⟩`]{`⟨*property*⟩`}{`⟨*between*⟩`}` Default: `\@firstofone`
Like `\printsongpropertylist` but separates items with ⟨*between*⟩ regardless of the length of the list.

* `\forsongpropertylist{`⟨*property*⟩`}{`⟨*code*⟩`}`
Places all items of the property list ⟨*property*⟩ in the input stream, each item preceded with ⟨*code*⟩. Items are surrounded with braces which means that the last token in ⟨*code*⟩ may be a macro with a mandatory argument.

* `\ifsongproperty{`⟨*property*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Checks if property ⟨*property*⟩ has been set.

`\ifsongpropertiesequal{`⟨*property 1*⟩`}{`⟨*property 2*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Checks if properties ⟨*property 1*⟩ and ⟨*property 2*⟩ have been set to the same value.

`\expandsongpropertycode{`⟨*code*⟩`}`
Exhaustively expands ⟨*code*⟩. Experienced users won't need this. It is essentially

`\begingroup\edef\x{\endgroup`⟨*code*⟩`}\x`.

(More precisely it is a wrapper for the expl3 function `\use:x`.) This means that any # needs to doubled. Inside the argument of this commands non-robust macros that should not be expanded need to be prefixed with `\noexpand`.

With the right template definition you can index composers, interprets, song titles, … You can write tables of contents for properties such as song titles, and so on, and so on. LEADSHEETS does not do this for you and it may require some experience to create templates which do all this.

### 12.3.2. Examples

In order to give you an idea on how to use templates I'll show you how the existing ones are defined and one new definition.

**The "minimal" template** This is quite short and self-explaining.

```
\definesongtitletemplate{minimal}{\section*{\songproperty{title}}}
```

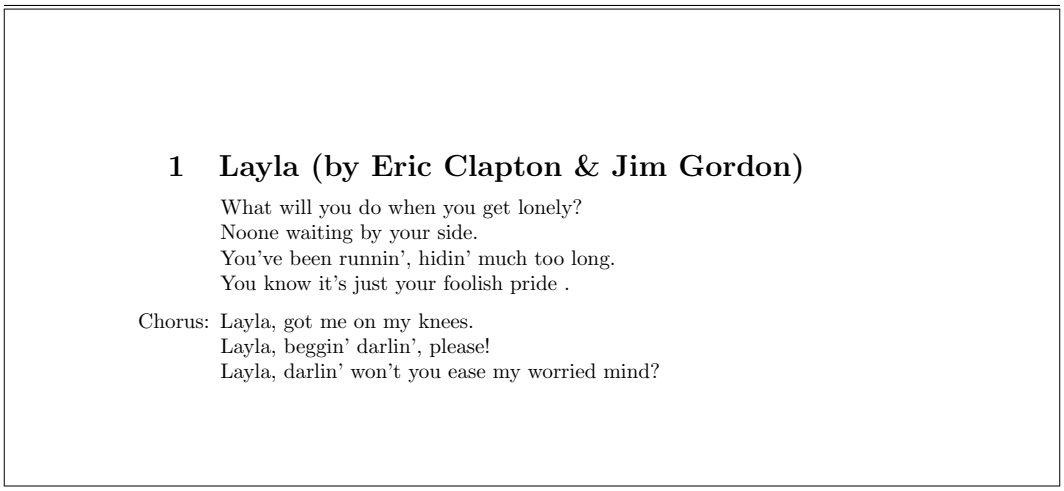
**A custom template** Now let's see an example for a newly defined template. It's nearly as simple as the "minimal" template.

```
\documentclass{article}
\usepackage{leadsheets}
\definesongtitletemplate{custom}{
  \section{%
    \songproperty{title}%
    \ifsongproperty{composer}
      { (by \printsongpropertylist{composer}{ \& }{, }{ \& })}
      {}%
  }
}
\setleadsheets{title-template = custom}
\begin{document}

\begin{song}{title=Layla,composer={Eric Clapton and Jim
      Gordon},tags={clapton,unplugged,r&b}}
\begin{verse}
```

```
  What will you do when you get lonely? \\
  Noone waiting by your side. \\
  You've been runnin', hidin' much too long. \\
  You know it's just your foolish pride .
\end{verse}
\begin{chorus}
  Layla, got me on my knees. \\
  Layla, beggin' darlin', please! \\
  Layla, darlin' won't you ease my worried mind?
\end{chorus}
\end{song}

\end{document}
```



1 Layla (by Eric Clapton & Jim Gordon)

What will you do when you get lonely?
Noone waiting by your side.
You've been runnin', hidin' much too long.
You know it's just your foolish pride .

Chorus: Layla, got me on my knees.
Layla, beggin' darlin', please!
Layla, darlin' won't you ease my worried mind?

**The "tabular" template** This one is a lot more advanced and demonstrates various of the available commands.

```
\definesongtitletemplate{tabular}{
  \section{\songproperty{title}}
  \begingroup\footnotesize
  \begin{tabular}{
      @{}
      >{\raggedright\arraybackslash}p{.5\linewidth}
      @{}
      >{\raggedleft\arraybackslash}p{.5\linewidth}
      @{}
    }
    \ifsongproperty{interpret}
```

```
      {\GetTranslation{leadsheets/interpret}}
      {}%
    \ifsongproperty{composer}
      {%
        &
        \GetTranslation{leadsheets/composer}: %
        \printsongpropertylist{composer}{ \& }{,~}{ \& }
        \ifsongproperty{lyrics}
          {
            \\ &
            \GetTranslation{leadsheets/lyrics}: %
            \printsongpropertylist{lyrics}{ \& }{,~}{ \& }
          }
          {}%
      }
      {}%
    \ifsongproperty{interpret}{\\}{\ifsongproperty{composer}{\\}{}}%
    \ifsongproperty{genre}
      {& Genre:~ \songproperty{genre} \\}
      {}%
    \ifsongproperty{tempo}
      {& Tempo:~ \songproperty{tempo} \\}
      {}%
    \ifsongproperty{key}
      {%
        & \setchordnames{
            major = -\GetTranslation{leadsheets/major} ,
            minor = -\GetTranslation{leadsheets/minor}
          }%
        \GetTranslation{leadsheets/key}: %
        \expandsongpropertycode{\writechord{\songproperty{key}}} \\%
      }
      {}%
  \end{tabular}
  \par\endgroup
}
```

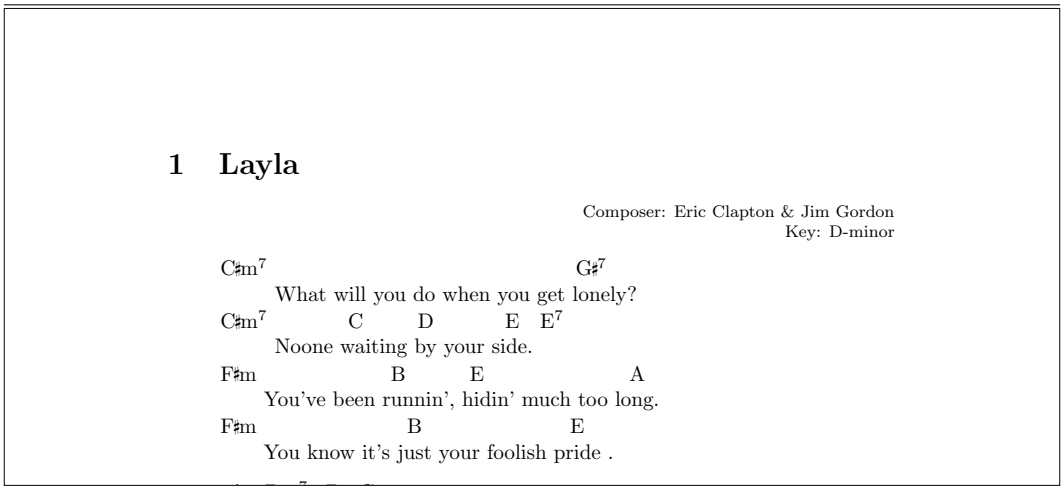
A song using the “tabular” template:

```
\documentclass{article}
\usepackage{leadsheets}
\usepackage{array}
\setleadsheets{title-template = tabular}
\begin{document}

\begin{song}{
```

```
    title = Layla ,
    composer = {Eric Clapton and Jim Gordon} ,
    tags = {clapton,unplugged,r&b} ,
    key = Dmi
  }
\begin{verse}
  ^{C#mi7} What will you do when you get ^*{G#7}lone ly? \\
  ^{C#mi7} Noone ^*{C}wai ting ^{D}by your ^{E}side. ^{E7} \\
  ^{F#mi} You've been ^*{B}run nin', ^*{E}hid in' much too ^{A}long. \\
  ^{F#mi} You know it's ^{B}just your foolish ^{E}pride .
\end{verse}
\begin{chorus}
  ^*{A}Lay ---^-{Dmi7}la, \quad ^{Bb} ^{C}got me on my knees. \\
  Lay^-{Dmi7}la, \quad ^{Bb} ^*{C}beg gin' darlin', ^{Dmi7}please, Layla. \\
  Darlin' won't you ease my worried ^{Dmi7}mind? ^{Bb} ^{C}
\end{chorus}
\end{song}

\end{document}
```



1 Layla

Composer: Eric Clapton & Jim Gordon
Key: D-minor

C♯m7 G♯7
What will you do when you get lonely?
C♯m7 C D E E7
Noone waiting by your side.
F♯m B E A
You've been runnin', hidin' much too long.
F♯m B E
You know it's just your foolish pride .

## 13. Internationalization

Th environments described in sections 7 and 10 as well as a few other words used in LEADSHEETS are translated with the help of the translations [Nie13] package. All predefined and available translation strings are listed in table 4. You can change those translations or add translations for other languages with this command:

`\DeclareTranslation{`⟨*language*⟩`}{`⟨*string*⟩`}{`⟨*translation*⟩`}`

The command provided by the translations package for translating strings.

Those translations can be used for example in song title templates (see section 12). One of the strings listed in table 4 is a little different: the string `leadsheets/interpret` is declared as

```
\DeclareTranslation{English}{leadsheets/interpret}
  {as interpreted by \printsongpropertylist{interpret}{ \& }{,~}{ \& }}
\DeclareTranslation{German}{leadsheets/interpret}
  {wie von \printsongpropertylist{interpret}{ \& }{,~}{ \& } interpretiert}
```

which means it uses the song property `interpret`. As a consequence it only really can be used inside a `song` environment. In other cases as for example in table 4 the property part expands to nothing (but the spaces around it are of course there). Also keep in mind that `\printsongpropertylist` is not expandable.

Table 4: Predefined translation strings.

| String | English | German |
|---|---|---|
| `leadsheets/major` | major | Dur |
| `leadsheets/minor` | minor | Moll |
| `leadsheets/chorus` | Chorus | Refrain |
| `leadsheets/composer` | Composer | Komponist |
| `leadsheets/lyrics` | Lyrics | Text |
| `leadsheets/key` | Key | Tonart |
| `leadsheets/interpret` | as interpreted by | wie von  interpretiert |
| `leadsheets/intro` | Intro | Intro |
| `leadsheets/interlude` | Interlude | Interlude |
| `leadsheets/bridge` | Bridge | Bridge |
| `leadsheets/solo` | Solo | Solo |

# Part V.
# Appendix

> Talking about music is like dancing about architecture.
>
> *Thelonious Monk*

## A. References


[Ham12] Kevin W. Hamlen. songs. version 2.14, Mar. 17, 2012.
URL: http://mirror.ctan.org/macros/latex/contrib/songs/.

[MC08] Frank Mittelbach and David Carlisle. array. version 2.4c, Sept. 9, 2008.
URL: http://mirror.ctan.org/macros/latex/required/tools/.

[Nie13] Clemens Niederberger. translations. version 1.1a, Sept. 30, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/translations/.

[Rat10] Christopher Rath. songbook. version 4.5, Apr. 30, 2010.
URL: http://mirror.ctan.org/macros/latex/contrib/songbook/.

[The13a] The LaTeX3 Project Team. l3kernel. version SVN 4582, July 28, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/l3kernel/.

[The13b] The LaTeX3 Project Team. l3packages. version SVN 4582, July 28, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/l3packages/.

[Varoo] Various. *The Real Book. C Edition.* 6th edition. Vol. I.
Hal Leonard Publishing Corporation, Jan. 1, 2000. ISBN: 978-0634060380.


## B. Index