Contents

1 Insert Source Code Block for Org Babel—ask for a name **1**

1.1 Introduction . . . 2
1.2 Function to insert an Org Babel code block . . . 2
1.3 Ask for a name and update variable . . . 2
1.4 Update other variables dependent on the name . . . 2
1.5 Update section name header variable . . . 3
1.6 Get the section name header . . . 3
1.7 Update noweb-ref block header variable . . . 3
1.8 Get noweb-ref block header . . . 3
1.9 Insert the code block . . . 3
1.10 Insert the language . . . 4
1.11 Languages . . . 4
1.12 Insert the block name, if non-empty . . . 5
1.13 Insert the section name header, if non-empty . . . 5
1.14 Insert the noweb-ref block header, if non-empty . . . 5
1.15 Add a keybinding . . . 6

2 License **6**

1 Insert Source Code Block for Org Babel—ask for a name

This project is a function for Org Babel that, before inserting a source code block, ask the user for a name. If the user does not inform a name, nothing changes. But if the user does inform a name, the function will automatically

- decorate the block with a nicelly formatted string,
- insert a `:noweb-ref` for the block, and
- create an Org heading for the code block section.

The idea is to make it easier to do literate programming on Emacs.

All the text below *is* literate programming (LP) itself: the ideas and divagations are mixed with the source code, and the final product is automatically extracted from this text via *tangling*.

Thus, the whole entire source code for the program is listed below! For me, this was the first not-so-short project that I used LP, and I loved it!

1.1 Introduction

Org Babel has this nice little functionality of

> Insert[ing] a block structure of the type `#+begin_foo/#+end_foo`.

using the function `org-insert-structure-template`. This is very practical, because I have to type less.

What I *wish* Org Babel had was a way to insert the *name* of the code blocks. This would help a lot in my literate programming activities.

```
<<Function to insert an Org Babel code block>>
```

1.2 Function to insert an Org Babel code block

The code block may or may not have a name, and this will determine several things about the tangling process. The general structure of the function is depicted below.

⟨ Function to insert an Org Babel code block ⟩ ≡

```
(defun my-org-babel-insert-src-block ()
  (interactive)
  <<Ask for a name and update variable>>
  <<Update other variables dependent on the name>>
  <<Insert the code block>>)
```

1.3 Ask for a name and update variable

⟨ Block name ⟩ ≡

```
my-org-babel-insert-src-block-name
```

⟨ Ask for a name and update variable ⟩ ≡

```
(setq <<Block name>>
      (read-string "Name: "))
```

1.4 Update other variables dependent on the name

⟨ Update other variables dependent on the name ⟩ ≡

```
<<Update section name header variable>>
<<Update noweb-ref block header variable>>
```

1.5 Update section name header variable

⟨ Section name header ⟩ ≡

```
my-org-babel-insert-src-block-section-name-header
```

⟨ Update section name header variable ⟩ ≡

```
(setq <<Section name header>>
      <<Get the section name header>>)
```

1.6 Get the section name header

⟨ Get the section name header ⟩ ≡

```
(if (string= <<Block name>> "") ""
  (concat "@@latex: \\noindent@@ \\langle " <<Block name>> " \\rangle\\equiv"))
```

1.7 Update noweb-ref block header variable

⟨ noweb-ref block header ⟩ ≡

```
my-org-babel-insert-src-block-noweb-ref-block-header
```

⟨ Update noweb-ref block header variable ⟩ ≡

```
(setq <<noweb-ref block header>>
      <<Get noweb-ref block header>>)
```

1.8 Get noweb-ref block header

⟨ Get noweb-ref block header ⟩ ≡

```
(if (string= <<Block name>> "") ""
  (concat " :noweb-ref " <<Block name>>))
```

1.9 Insert the code block

⟨ Insert the code block ⟩ ≡

```
<<Insert the block name, if non-empty>>
<<Insert the section name header, if non-empty>>
(insert "#+begin_src ")
<<Insert the language>>
<<Insert the noweb-ref block header, if non-empty>>
(insert "\n\n#+end_src\n")
(previous-line)(previous-line)
```

1.10 Insert the language

Insert the language ⟨ Insert the language ⟩ ≡

```
(insert
 (completing-read "Language: "
                  <<Languages>>))
```

1.11 Languages

These are the languages supported by Org Babel, as of November, 2022.
⟨ Languages ⟩ ≡

```
(list
 "C"
 "D"
 "F90"
 "R"
 "awk"
 "calc"
 "clojure"
 "comint"
 "cpp"
 "css"
 "ditaa"
 "dot"
 "elisp"
 "emacs-lisp"
 "eshell"
 "forth"
 "gnuplot"
 "groovy"
 "haskell"
 "java"
 "js"
 "julia"
 "latex"
 "lisp"
 "lua"
 "ly"
 "makefile"
```

```
"matlab"
"max"
"ocaml"
"octave"
"org"
"perl"
"plantuml"
"processing"
"python"
"ruby"
"sass"
"scheme"
"screen"
"sed"
"shell"
"sql"
"sqlite")
```

1.12 Insert the block name, if non-empty

⟨ Insert the block name, if non-empty ⟩ ≡

```
(unless (string= <<Block name>> "")
  (org-insert-heading)
  (insert <<Block name>> "\n\n"))
```

1.13 Insert the section name header, if non-empty

⟨ Insert the section name header, if non-empty ⟩ ≡

```
(unless (string= <<Section name header>> "")
  (insert <<Section name header>> "\n"))
```

1.14 Insert the noweb-ref block header, if non-empty

⟨ Insert the noweb-ref block header, if non-empty ⟩ ≡

```
(unless (string= <<noweb-ref block header>> "")
  (insert <<noweb-ref block header>>))
```

1.15 Add a keybinding

Add a keybinding
⟨ Add a keybinding ⟩ ≡

```
(define-key org-mode-map
  (kbd "C-. s")
  'my-org-babel-insert-src-block)
```

2 License