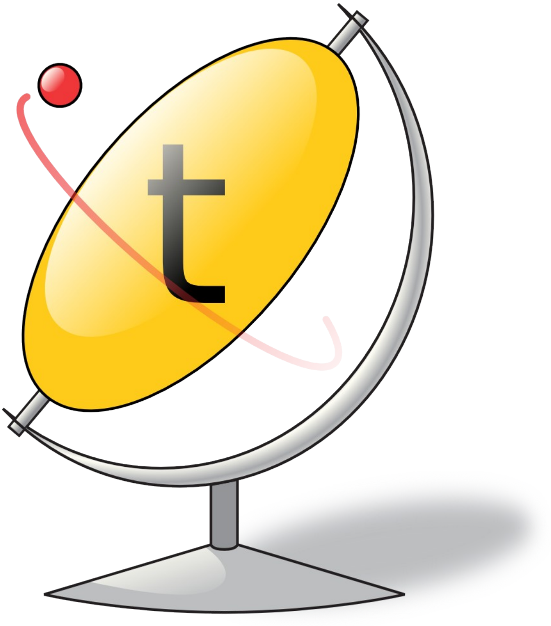
Textadept

A fast, minimalist, and remarkably extensible text editor



Quick Reference

Fourth Edition

Mitchell

Textadept Quick Reference

Textadept is a fast, minimalist, and remarkably extensible cross-platform text editor for programmers. This quick reference contains a wealth of knowledge on how to script and configure Textadept using the Lua programming language. It groups the editor's rich API into a series of tasks in a convenient and easy-to-use manner.

This book covers how to:

- Leverage Textadept's important files and folders
- Adeptly navigate and manipulate text
- Mark lines and text visually
- Show interactive lists and call tips
- Prompt for user input in various ways
- Spawn asynchronous, interactive child processes
- Configure colors, themes, and other settings
- Define lexers for highlighting source code
- And much more

Mitchell is the author and principal developer of Textadept and commands over 10 years of experience with Lua.

foicica.com

FOURTH EDITION

Textadept

Quick Reference

Mitchell

Textadept Quick Reference
by Mitchell

Contact the author at mitchell@foicica.com.

Although great care has been taken in preparing this book, the author assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. All product names mentioned in this book are trademarks of their respective owners.

Editor: Ana Balan
Cover Designer: Mitchell
Interior Designer: Mitchell
Indexer: Mitchell

Printing history:
December 2013: First Edition
May 2015: Second Edition
October 2016: Third Edition
August 2018: Fourth Edition

ISBN: 978-0-9912379-4-4

Preface to the Fourth Edition

This book is an updated version of the third edition of *Textadept Quick Reference*. It includes many of the new features introduced in the Textadept 9.x releases, and covers the backwards-incompatible changes made for Textadept 10.x. In a nutshell, this book covers the following new topics:

- Textadept's new and improved lexer syntax.
- Its simplified configuration file.
- The new color selection dialog.

For a comprehensive list of changes between Textadept versions, please refer to the editor's *CHANGELOG.md* or *doc/CHANGELOG.html* files, which are distributed with the application. The online version is located at *https://foicica.com/textadept/CHANGELOG.html*.

Contents

Introduction

Textadept is a fast, minimalist, and remarkably extensible cross-platform text editor for programmers. Written in a combination of C and Lua[1] and relentlessly optimized for speed and minimalism for over nine years, Textadept is an ideal editor for programmers who want endless extensibility without sacrificing speed or succumbing to code bloat and featuritis.

Textadept runs in both graphical and text-based user interface environments. The text-based version of the editor is referred to as the "terminal version", since it executes within a terminal emulator.

Textadept Quick Reference is designed to help the user "get things done" when it comes to scripting and configuring Textadept. Its pragmatic approach assumes the user has a working knowledge of both Lua and Textadept. This book is broken up into a number of descriptive sections with conveniently grouped tasks that cover nearly every aspect of Textadept's Application Programming Interface (API). For the most part, the contents of each task are not listed in conceptual order. They are listed in procedural order, an order the user would likely follow when writing Lua scripts. This quick reference serves as a complement to Textadept's comprehensive Manual and extensive API documentation.

While this book aims to be a complete reference, it does omit some of the less useful features of Textadept's API. For example, although many of Textadept's table fields are both readable and writable, this reference sometimes chooses to cover only one of those operations. (Unless a field is marked "Read-only" or "Write-only", it is readable and writable.) This book also does not cover Lua's standard libraries.

Finally, the facilities in this book are designed to be used primarily in user-written Lua scripts and in the occasional "one-shot" Lua command. If the user keeps this in mind, he or she can realize Textadept's full potential.

[1] *https://www.lua.org*

Download

Textadept binary packages for Windows, Mac OSX, and Linux platforms are available from *https://foicica.com/textadept*. Each package is self-contained and need not be installed. Textadept's source code is also included in each archive. The user may compile the application manually by following the instructions in the editor's Manual.

Conventions

This book uses the following conventions.

Italic
: Used for filenames and for introducing new terms.

`Constant width`
: Used for environment variables, command line options, and Lua code, including functions, tables, and variables.

`Constant width italic`
: Used for user-specified parameters.

`[]`
: Used for optional function arguments, except in code examples that index Lua tables. Unless otherwise specified, optional arguments default to `nil`.

Terminology

This book uses the following terminology.

Buffer
: An object that contains editable text.

View
: An object that contains a single buffer.

Caret
: Either the visual that represents the text insertion point or the end point of a text selection.

Anchor
: The start point of a text selection or search.

Virtual Space
: The space past the ends of lines.

Lexer
: A Lua module that highlights the syntax of source code written in a particular programming language. Textadept refers to a programming language by its lexer's name.

Style
: A collection of display settings specific to source code comments, strings, keywords, and other ranges of text.

Language Module
: A Lua module automatically loaded by Textadept when editing source code in a particular programming language. The module's name matches the language's lexer name. Not all languages have language modules.

Environment Variables

Textadept utilizes the following environment variables.

`HOME` or `USERHOME`
: The user's home directory. Textadept's user data and preferences exist in a *.textadept/* sub-directory, denoted as *~/.textadept/* throughout this book.

  On Windows, this directory is typically *C:\Users\username\*. On Mac OSX, it is */Users/username/*. On Linux and BSD it is often */home/username/*.

`LANG`
: The user's default locale. Textadept will display localized text and messages in it if possible.

`TA_LUA_PATH`
`TA_LUA_CPATH`
: The Textadept equivalent of `LUA_PATH` and `LUA_CPATH`. Used by Lua's `require()` function for finding modules.

Important Files and Directories

Textadept allows the user to configure and customize the editor using several important files and directories contained

Search for Text

Textadept supplies a variety of tools to search for text: a simple search API, a more complex API for search and replace, a Find & Replace Pane for interactive search and replace, and an incremental find entry. The first two tools make use of the search flags defined in Table 2 and the regular expression syntax in Table 3. The last two tools make use of the search flags defined in the section “Interact with the Find & Replace Pane” on page 42 and the regular expression syntax in Table 3.

NOTE

Textadept’s regular expressions are based on the C++11 standard’s ECMAScript syntax[2].

Table 2. Buffer search flags

Bit Flag	Description
`buffer.FIND_MATCHCASE`	Match search text case sensitively.
`buffer.FIND_WHOLEWORD`	Match search text only when it is surrounded by non-word characters. `buffer.word_chars` is a string that contains all word characters.
`buffer.FIND_WORDSTART`	Match search text only when the previous character is a non-word character.
`buffer.FIND_REGEXP`	Interpret search text as a regular expression. (See Table 3.)

Table 3. Regular expression and Lua pattern special characters

Regex	Lua	Meaning
.	.	Matches any character.
[[:alpha:]]	%a	Matches any letter.
\d	%d	Matches any digit.

[2] *http://www.cplusplus.com/reference/regex/ECMAScript/*

Regex	Lua	Meaning
[[:lower:]]	%l	Matches any lower case character.
[[:punct:]]	%p	Matches any punctuation character.
\s	%s	Matches any space character.
[[:upper:]]	%u	Matches any upper case character.
\w	%w	Matches any alphanumeric character.
[[:xdigit:]]	%x	Matches any hexadecimal digit.
[*set*]	[*set*]	Matches any character in *set*, including ranges (e.g. `[A-Za-z]`).
[^*set*]	[^*set*]	Matches the complement of *set*.
*	*	Matches the previous item (Regex) or *character class* (Lua) zero or more times. The previous Lua expressions listed are character classes.
+	+	Matches the previous item or class one or more times.
*?	-	Matches the previous item or class zero or more times, but as few times as possible.
+?		Matches the previous item one or more times, but as few times as possible.
?	?	Matches the previous item or class once, or not at all.
{*m*,*n*}		Matches the previous item between *m* and *n* times.
{*m*,}		Matches the previous item at least *m* times.
{*m*}		Matches the previous item exactly *m* times.
|		Matches either the previous item or the next item.
	%b*xy*	Matches a balanced string that starts with *x* and ends with *y*.
	%f[*set*]	Matches a position where the next character belongs to *set*, but the previous character does not.
\<		Matches the beginning of a word.
\>		Matches the end of a word.
\b		Matches a word boundary.
^	^	Matches the beginning of a line unless inside a set.

Regex	Lua	Meaning
$	$	Matches the end of a line unless inside a set.
(	(	The beginning of a captured matching region.
)	)	The end of a captured matching region.
(?: ...)		Consider the matched sequence as a single, uncaptured item.
\n	%n	Represents the *n*th captured matching region's text. In replacement text, "\0" represents all matched text.
\x	%x	Represents non-alphanumeric character *x*, ignoring any special meaning it may have by itself.

Simple Search

The user can perform simple searches anchored at the caret position. Example 8 demonstrates a simple (but not comprehensive) method of jumping to the beginning of the next occurrence of the word under the caret.

Example 8. Find the next instance of the current word

```
-- Determine the word under the caret.
local pos = buffer.current_pos
local s = buffer:word_start_position(pos, true)
local e = buffer:word_end_position(pos, true)
local word = buffer:text_range(s, e)
-- Perform the search.
local flags = buffer.FIND_MATCHCASE +
              buffer.FIND_WHOLEWORD
buffer:search_anchor()
pos = buffer:search_next(flags, word)
if pos ~= -1 then buffer:goto_pos(pos) end
```

`buffer:search_anchor()`

Anchors the position that `buffer:search_next()` and `buffer:search_prev()` start at to the caret position.

`buffer:search_next(`*`flags`*`, `*`text`*`)`
`buffer:search_prev(`*`flags`*`, `*`text`*`)`

Searches for and selects the next or previous occurrence of string ***text*** using search flags bit-mask ***flags***, returning that occurrence's position or `-1` if ***text*** was not found.

Concept Index