Keyboard shortcuts

- `M-x cider-jack-in`: Launch an nREPL server and a REPL client. Prompts for a project root if given a prefix argument.
- `M-x cider`: Connect to an already-running nREPL server.

While you're in `clojure-mode`, `cider-jack-in` is bound for convenience to `C-c M-j` and `cider-connect` is bound to `C-c M-c`.

cider-mode

Keyboard shortcut	Description
`C-x C-e` `C-c C-e`	Evaluate the form preceding point and display the result in the echo area. If invoked with a prefix argument, insert the result into the current buffer.
`C-c C-w`	Evaluate the form preceding point and replace it with its result.
`C-c M-e`	Evaluate the form preceding point and output it result to the REPL buffer. If invoked with a prefix argument, takes you to the REPL buffer after being invoked.
`C-c M-p`	Load the form preceding point in the REPL buffer.
`C-c C-p`	Evaluate the form preceding point and pretty-print the result in a popup buffer.
`C-c C-f`	Evaluate the top level form under point and pretty-print the result in a popup buffer.
`C-M-x` `C-c C-c`	Evaluate the top level form under point and display the result in the echo area.
`C-u C-M-x` `C-u C-c C-c`	Debug the top level form under point and walk through its evaluation
`C-c C-r`	Evaluate the region and display the result in the echo area.
`C-c C-b`	Interrupt any pending evaluations.
`C-c C-m`	Invoke `macroexpand-1` on the form at point and display the result in a macroexpansion buffer. If invoked with a prefix argument, `macroexpand` is used instead of `macroexpand-1`.
`C-c M-m`	Invoke `clojure.walk/macroexpand-all` on the form at point and display the result in a macroexpansion buffer.
`C-c C-n`	Eval the ns form.
`C-c M-n`	Switch the namespace of the REPL buffer to the namespace of the current buffer.
`C-c C-z`	Switch to the relevant REPL buffer. Use a prefix argument to change the namespace of the REPL buffer to match the currently visited source file.
`C-u C-u C-c C-z`	Switch to the REPL buffer based on a user prompt for a directory.
`C-c M-d`	Display default REPL connection details, including project directory name, buffer namespace, host and port.
`C-c M-r`	Rotate and display the default nREPL connection.
`C-c M-o`	Clear the entire REPL buffer, leaving only a prompt. Useful if you're running the REPL buffer in a side by side buffer.
`C-c C-k`	Load (eval) the current buffer.
`C-c C-l`	Load (eval) a Clojure file.

Keyboard shortcut	Description
C-c C-d d	Display doc string for the symbol at point. If invoked with a prefix argument, or no symbol is found at point, prompt for a symbol.
C-c C-d j	Display JavaDoc (in your default browser) for the symbol at point. If invoked with a prefix argument, or no symbol is found at point, prompt for a symbol.
C-c M-i	Inspect expression. Will act on expression at point if present.
C-c M-t v	Toggle var tracing.
C-c M-t n	Toggle namespace tracing.
C-c C-u	Undefine a symbol. If invoked with a prefix argument, or no symbol is found at point, prompt for a symbol.
C-c ,	Run tests for namespace.
C-c C-,	Re-run test failures/errors for namespace.
C-c M-,	Run test at point.
C-c C-t	Show the test report buffer.
M-.	Jump to the definition of a symbol. If invoked with a prefix argument, or no symbol is found at point, prompt for a symbol.
C-c M-.	Jump to the resource referenced by the string at point.
C-c C-.	Jump to some namespace on the classpath.
M-,	Return to your pre-jump location.
M-TAB	Complete the symbol at point.
C-c C-d g	Lookup symbol in Grimoire.
C-c C-d a	Apropos search for functions/vars.
C-c C-d A	Apropos search for documentation.

cider-repl-mode

Keyboard shortcut	Description
RET	Evaluate the current input in Clojure if it is complete. If incomplete, open a new line and indent. If invoked with a prefix argument is given then the input is evaluated without checking for completeness.
C-RET	Close any unmatched parenthesis and then evaluate the current input in Clojure.
C-j	Open a new line and indent.
C-c M-o	Clear the entire REPL buffer, leaving only a prompt.
C-c C-o	Remove the output of the previous evaluation from the REPL buffer.
C-c C-u	Kill all text from the prompt to the current point.
C-c C-b C-c C-c	Interrupt any pending evaluations.
C-up C-down	Goto to previous/next input in history.
M-p M-n	Search the previous/next item in history using the current input as search pattern. If M-p/M-n is typed two times in a row, the second invocation uses the same search pattern (even if the current input has changed).
M-s M-r	Search forward/reverse through command history with regex.
C-c C-n C-c C-p	Move between the current and previous prompts in the REPL buffer. Pressing RET on a line with old input copies that line to the newest prompt.
TAB	Complete symbol at point.

Keyboard shortcut	Description
`C-c C-d d`	Display doc string for the symbol at point. If invoked with a prefix argument, or no symbol is found at point, prompt for a symbol
`C-c C-d j`	Display JavaDoc (in your default browser) for the symbol at point. If invoked with a prefix argument, or no symbol is found at point, prompt for a symbol.
`C-c C-d g`	Lookup symbol in Grimoire.
`C-c C-d a`	Apropos search for functions/vars.
`C-c C-d A`	Apropos search for documentation.
`C-c C-z`	Switch to the previous Clojure buffer. This complements `C-c C-z` used in cider-mode.
`C-c M-i`	Inspect expression. Will act on expression at point if present.
`C-c M-n`	Select a namespace and switch to it.
`C-c C-.`	Jump to some namespace on the classpath.
`C-c M-t v`	Toggle var tracing.
`C-c M-t n`	Toggle namespace tracing.

In the REPL you can also use "shortcut commands" by pressing `,` at the beginning of a REPL line. You'll be presented with a list of commands you can quickly run (like quitting, displaying some info, clearing the REPL, etc). The character used to trigger the shortcuts is configurable via `cider-repl-shortcut-dispatch-char`. Here's how you can change it to `:`:

```
(setq cider-repl-shortcut-dispatch-char ?\:)
```

cider-macroexpansion-mode

Keyboard shortcut	Description
`C-c C-m`	Invoke `macroexpand-1` on the form at point and replace the original form with its expansion. If invoked with a prefix argument, `macroexpand` is used instead of `macroexpand-1`.
`C-c M-m`	Invoke `clojure.walk/macroexpand-all` on the form at point and replace the original form with its expansion.
`g`	The prior macroexpansion is performed again and the current contents of the macroexpansion buffer are replaced with the new expansion.
`C-/` `C-x u`	Undo the last inplace expansion performed in the macroexpansion buffer.

cider-inspector-mode

Keyboard shortcut	Description
`Tab` and `Shift-Tab`	navigate inspectable sub-objects
`Return`	inspect sub-objects
`l`	pop to the parent object
`g`	refresh the inspector (e.g. if viewing an atom/ref/agent)

cider-test-report-mode

Keyboard shortcut	Description
C-c ,	Run tests for namespace.
C-c C-,	Re-run test failures/errors for namespace.
C-c M-,	Run test at point.
M-p	Move point to previous test.
M-n	Move point to next test.
t and M-.	Jump to test definition.
d	Display diff of actual vs expected.
e	Display test error cause and stacktrace info.

cider-stacktrace-mode

Keyboard shortcut	Description
M-p	move point to previous cause
M-n	move point to next cause
M-. and Return	navigate to the source location (if available) for the stacktrace frame
Tab	Cycle current cause detail
0 and S-Tab	Cycle all cause detail
1	Cycle cause #1 detail
2	Cycle cause #2 detail
3	Cycle cause #3 detail
4	Cycle cause #4 detail
5	Cycle cause #5 detail
j	toggle display of java frames
c	toggle display of clj frames
r	toggle display of repl frames
t	toggle display of tooling frames (e.g. compiler, nREPL middleware)
d	toggle display of duplicate frames
a	toggle display of all frames

cider-debug

cider-debug (invoked with C-u C-M-x) tries to be consistent with Edebug. So it makes available the following bindings while stepping through code.

Keyboard shortcut	Description
n	Next step
c	Continue without stopping
o	Move out of the current sexp (like up-list)
i	Inject a value into running code
e	Eval code in current context
q	Quit execution