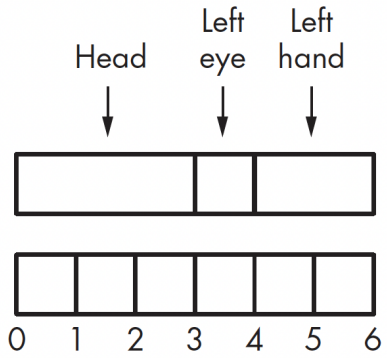
Clojure for the Brave and True

Learn the Ultimate Language and Become a Better Programmer

by Daniel Higginbotham

errata updated to print 8

Page	Error	Correction	Print corrected
11	If you don't follow the thorough Emacs instructions in this chapter, or if you choose to use a different editor, it's worthwhile to at least invest some time in setting up your editor to work with a REPL.	If you don't follow the thorough Emacs instructions in this chapter, or if you choose to use a different editor, it's worthwhile to at least invest some time in setting up your editor to work with a REPL. **Two alternatives that I recommend and that are well regarded in the community are Cursive (*https://cursive-ide.com/*) and Nightcode (*https://sekao.net/nightcode/*).**	Print 4
13	I've created a repository of all the files you need to configure Emacs for Clojure, available at *https://github.com/flyingmachine/emacs-for-clojure/archive/book1.zip*. Do the following to delete your existing Emacs configuration and install the Clojure-friendly one:	I've created a repository of all the files you need to configure Emacs for Clojure, available at *https://github.com/flyingmachine/emacs-for-clojure/archive/book1.zip*. **NOTE** **These tools are constantly being updated, so if the instructions below don't work for you or you want to use the latest configuration, please read the instructions at *https://github.com/flyingmachine/emacs-for-clojure/*.** Do the following to delete your existing Emacs configuration and install the Clojure-friendly one:	Print 4
13	1. Close Emacs. 2. Delete *~/.emacs* or *~/.emacs.d* if they exist. This is where Emacs looks for configuration files, and deleting these files and directories will ensure that you start with a clean slate. 3. Download the Emacs configuration zip file from the book's resource page and unzip it. Its contents should be a folder, *emacs-for-clojure-book1*. Run `mv path/to/emacs-for-clojure-book1 ~/.emacs.d`. **4. Create the file *~/.lein/profiles.clj* and add this line to it:** `{:user {:plugins [[cider/cider-nrepl "0.8.1"]]}}` 5. Open Emacs. When you open Emacs, you might see a lot of activity as Emacs downloads a bunch of useful packages. Once the activity stops, go ahead and just quit Emacs, and then open it again.	1. Close Emacs. 2. Delete *~/.emacs* or *~/.emacs.d* if they exist. **(Windows users, your Emacs files will probably live in *C:\Users\<your_user_name>\AppData\Roaming*. So, for example, you would delete *C:\Users\jason\AppData\Roaming\.emacs.d*.)** This is where Emacs looks for configuration files, and deleting these files and directories will ensure that you start with a clean slate. 3. Download the Emacs configuration zip file from the book's resource page and unzip it. Its contents should be a folder, *emacs-for-clojure-book1*. Run `mv path/to/emacs-for-clojure-book1 ~/.emacs.d`. 4. Open Emacs. When you open Emacs, you might see a lot of activity as Emacs downloads a bunch of useful packages. Once the activity stops, go ahead and just quit Emacs, and then open it again. **(If you don't see any activity, quit and restart Emacs just for funsies.)**	Print 4
26	In the *core.clj* buffer, use **C-c M-n**.	In the *core.clj* buffer, use **C-c M-n n**.	Print 4

Page	Error	Correction	Print corrected
68	Figure 3-1 replacement		Print 2
68	The body part is hit **if** the accumulated size **is greater than** the target, so if the target is **0, 1, or 2**, then the head was hit. Otherwise, you take the next part, the left eye, and increase the accumulated size to 4, yielding a hit if the target is **3**. Similarly, the left hand gets hit if the target is **4 or 5**.	The body part is hit **when** the accumulated size **exceeds** the target, so if the target is **less than 3**, then the head was hit. Otherwise, you take the next part, the left eye, and increase the accumulated size to 4, yielding a hit if the target is **greater than or equal to 3 and less than 4**. Similarly, the left hand gets hit if the target is **greater than or equal to 4 and less than 6**.	Print 2
82	Each entry has the **year,** month, day, and what you ate.	Each entry has the month, day, and what you ate.	Print 2
95	First, map creates a seq of key-value pairs like `([:name "Bella Swan"] [:glitter-index] 0)`.	First, map creates a seq of key-value pairs like `([:name "Bella Swan"] [:glitter-index 0])`.	Print 2
118	. . . positions 1, 6, **and 11** have valid moves . . .	. . . positions 1, 6, **11, and 13** have valid moves . . .	Print 2
142	`(str (:lng latlng) "," (:lat latlng)))`	`(str (:lat latlng) "," (:lng latlng)))`	Print 2
143	```;; These two <g> tags flip the coordinate system "<g transform=\"translate(0," height ")\">" "<g transform=\"scale(1,-1)\">"```	```;; These two <g> tags change the coordinate system so that ;; 0,0 is in the lower-left corner, instead of SVG's default ;; upper-left corner "<g transform=\"translate(0," height ")\">" "<g transform=\"rotate(-90)\">"```	Print 2
162	. . . into one that **can** Clojure can evaluate, . . .	. . . into one that Clojure can evaluate, . . .	Print 9
163	However, the second one might be easier understand . . .	However, the second one might be easier **to** understand . . .	Print 9
182	. . . the pairs are arranged in `order-details-validation`).	. . . the pairs are arranged in `order-details-validations`).	Print 3
183	`(if-valid order-details order-details-validation errors`	`(if-valid order-details order-details-validations errors`	Print 3
184	`(when-valid order-details order-details-validation`	`(when-valid order-details order-details-validations`	Print 3
201	`@butter-promise`	`butter-promise`	Print 3

Page	Error	Correction	Print corrected
212	`(future (println (:percent-deteriorated zombie-state)))))`	`(future (println (:cuddle-hunger-level zombie-state)))))`	Print 3
217	`; This throws "IllegalStateException That's not mathy!"`	`; This throws "IllegalStateException: That's not mathy!"`	Print 3
227	Using `with-redefs` may be more appropriate than using bindings for setting up a test environment. It's also more widely applicable, in that you can use it for any var, not just dynamic ones.	`with-redefs` can be used with any `var`, not just dynamic ones. Because it has such far-reaching effects, you should only use it during testing. For example, you could use it to redefine a function that returns data from a network call, so that the function returns mock data without having to actually make a network request.	Print 3
270, 271	`feelings`	`feelings-about`	Print 3
286	`; => "I don't like 2"`	`; => "I'm not going to bother doing anything to that"`	Print 3