WIKIPEDIA

Clojure

Clojure (/ˈkloʊʒɜːr/, like "closure"[6]) is a dialect of the Lisp programming language.[7] Clojure is a general-purpose programming language with an emphasis on functional programming.[8] It runs on the Java virtual machine and the Common Language Runtime.[9] Like other Lisps, Clojure treats code as data and has a macro system.[10] The current development process is community-driven,[11] overseen by Rich Hickey as its benevolent dictator for life (BDFL).[12]

Clojure encourages immutability and immutable data structures. While its type system is entirely dynamic, recent efforts have also sought the implementation of gradual typing.[13] Clojure encourages programmers to be explicit about managing state and identity.[14] This focus on programming with immutable values and explicit progression-of-time constructs is intended to facilitate developing more robust programs, especially multithreaded ones.[15][16]

Clojure is used in industry by firms such as Funding Circle,[17] Walmart,[18] Puppet,[19] and other large software firms.[20] Commercial support for Clojure is provided by Cognitect.[20] Annual Clojure conferences are organised every year across the globe, the most famous of them being Clojure/conj (US east coast),[21] Clojure/West (US west coast),[22] and EuroClojure (Europe).[23]

The latest stable version of Clojure is 1.9,[24] released on December 8, 2017. The first stable release was version 1.0, released on May 4, 2009.[25] Clojure is free software released under the Eclipse Public License.[26]

Contents

Clojure

Paradigm	functional
Designed by	Rich Hickey
First appeared	2007
Stable release	1.9[1] / December 8, 2017
Typing discipline	dynamic, strong
Platform	JVM, CLR, JavaScript
License	Eclipse Public License
Filename extensions	.clj, .cljs, .cljc, .edn
Website	clojure.org (https://clojure.org)
Influenced by	
C++,[2] C#, Common Lisp, Erlang, Haskell, Mathematica,[3] ML, Prolog, Scheme, Java, Racket,[4] Ruby[5]	
Influenced	
Elixir, Hy, Pixie, Rhine (https://github.com/artagnon/rhine-ml)	
Clojure Programming at	

Wikibooks

History and development process

Rich Hickey is the creator of the Clojure language.[7] Before Clojure, he developed dotLisp, a similar project based on the .NET Framework,[27] and three earlier attempts to provide interoperability between Lisp and Java: a *Java foreign language interface for Common Lisp* (jfli),[28] A *Foreign Object Interface for Lisp* (FOIL),[29] and a *Lisp-friendly interface to Java Servlets* (Lisplets).[30]

Rich Hickey in San Francisco

Hickey spent about 2½ years working on Clojure before releasing it publicly, much of that time working exclusively on Clojure with no outside funding. At the end of this time, Hickey sent an email announcing the language to some friends in the Common Lisp community.

The development process is community-driven[11] and is managed at the Clojure Community website.[31] The website contains planning documents and an issue tracker where bugs may be filed. General development discussion occurs at the Clojure Dev Google Group.[32] While anyone can submit bug reports and ideas, to contribute patches one must sign the Clojure Contributor agreement.[33] JIRA tickets are processed by a team of screeners and finally Rich Hickey approves the changes.[34]

Design philosophy

Rich Hickey developed Clojure because he wanted a modern Lisp for functional programming, symbiotic with the established Java platform, and designed for concurrency.[15][16][35]

Clojure's approach to state is characterized by the concept of identities,[36] which are represented as a series of immutable states over time. Since states are immutable values, any number of workers can operate on them in parallel, and concurrency becomes a question of managing changes from one state to another. For this purpose, Clojure provides several mutable reference types, each having well-defined semantics for the transition between states.[14][36]

Features

Clojure runs on the Java virtual machine and as a result integrates with Java and fully supports calling Java code from Clojure,[46] and Clojure code can be called from Java also.[47] The community uses Leiningen[48] for project automation, providing support for Maven integration. Leiningen handles project package management and dependencies and is configured using Clojure syntax.[48]

Like most other Lisps, Clojure's syntax is built on S-expressions that are first parsed into data structures by a reader before being compiled.[49] Clojure's reader supports literal syntax for maps, sets and vectors in addition to lists, and these are compiled to the mentioned structures directly.[49] Clojure is a Lisp-1 and is not intended to be code-compatible with other dialects of Lisp, since it uses its own set of data structures incompatible with other Lisps.[49][50]

As a Lisp dialect, Clojure supports functions as first-class objects, a read–eval–print loop (REPL), and a macro system.[51] Clojure's macro system is very similar to that in Common Lisp with the exception that Clojure's version of the backquote (called "syntax quote") qualifies symbols with their namespace. This

helps prevent unintended name capture, as binding to namespace-qualified names is forbidden. It is possible to force a capturing macro expansion, but it must be done explicitly. Clojure does not allow user-defined reader macros, but the reader supports a more constrained form of syntactic extension.[52] Clojure supports multimethods[53] and for interface-like abstractions has a protocol[54] based polymorphism and data type system using records,[55] providing high-performance and dynamic polymorphism designed to avoid the expression problem.

Version	Release date	Major features added
	2007-10-16[37]	Initial public release
1.0	2009-05-04[38]	First stable release[39]
1.1	2009-12-31[38]	Futures[40]
1.2	2010-08-19[38]	Protocols[41]
1.3	2011-09-23[38]	Enhanced primitive support[42]
1.4	2012-04-15[38]	Reader literals
1.5	2013-03-01[38]	Reducers
1.6	2014-03-25[43]	Java API, improved hashing algorithms
1.7	2015-06-30[24]	Transducers, reader conditionals
1.8	2016-01-19[44]	Additional string functions, direct linking, socket server
1.9	2017-12-08[45]	Integration with spec, command line tools
Latest version Future release		

Clojure has support for lazy sequences and encourages the principle of immutability and persistent data structures. As a functional language, emphasis is placed on recursion and higher-order functions instead of side-effect-based looping. Automatic tail call optimization is not supported as the JVM does not support it natively;[56] it is possible to do so explicitly by using the `recur` keyword.[57] For parallel and concurrent programming Clojure provides software transactional memory,[58] a reactive agent system,[59] and channel-based concurrent programming.[60]

Recently Clojure introduced reader conditionals by allowing the embedding of Clojure and ClojureScript code in the same namespace.[24][61] Transducers have been added as a way for composing transformations. Transducers enable higher-order functions such as map and fold to generalize over any source of input data, as traditionally these functions operate on sequences, transducers allow them to work on channels and let the user define their own models for transduction.[62][63][64]

Platforms and popularity

The primary platform of Clojure is the JVM,[8][46] but other target implementations exist. The most notable of these are ClojureScript,[65] which compiles to JavaScript, and ClojureCLR,[66] a full port to the Common Language Runtime, interoperable with the .NET ecosystem. A survey of the Clojure community with 1,060 respondents conducted in 2013[67] found that 47% of respondents used both Clojure and ClojureScript when working with Clojure. In 2014 this number had increased to 55%,[68] in 2015, based on 2,445 respondents, to 66%.[69] Popular ClojureScript projects include implementations of the React library such as Reagent (https://reagent-project.github.io/) and Om (https://github.com/omcljs/om).[70]

Clojure has also been used for creative computing, including visual art, music, games, and poetry.[71]

Variant implementations of the Clojure language have been developed for platforms other than the above:

- las3r,[72] a subset of Clojure that runs on the ActionScript Virtual Machine (the Adobe Flash Player platform)
- clojure-py,[73] Clojure in pure Python
- rouge,[74] Clojure atop YARV in Ruby
- CljPerl,[75] Clojure atop Perl
- Pixie, Clojure-inspired Lisp dialect written in RPython

Examples

"Hello, World!" program:

```
(println "Hello world!")
```

Defining a function:

```
(defn square [x]
  (* x x))
```

GUI "Hello world" by calling the Java Swing library:

```
(javax.swing.JOptionPane/showMessageDialog nil "Hello World")
```

Using Unicode (Hello □ ("World") using the CJK code point for that word):

```
(println (str "Hello, " \u4e16)) ; to the console
(javax.swing.JOptionPane/showMessageDialog nil (str "Hello, " \u4e16 "!")) ; using Java GUI
```

A thread-safe generator of unique serial numbers (though, like many other Lisp dialects, Clojure has a built-in `gensym` function that it uses internally):

```
(let [i (atom 0)]
  (defn generate-unique-id
    "Returns a distinct numeric ID for each call."
    []
    (swap! i inc)))
```

An anonymous subclass of `java.io.Writer (http://docs.oracle.com/javase/7/docs/api/java/io/Writer.html)` that doesn't write to anything, and a macro using it to silence all prints within it:

```
(def bit-bucket-writer
  (proxy [java.io.Writer] []
    (write [buf] nil)
    (close []    nil)
    (flush []    nil)))

(defmacro noprint
  "Evaluates the given expressions with all printing to *out* silenced."
  [& forms]
  `(binding [*out* bit-bucket-writer]
     ~@forms))

(noprint
  (println "Hello, nobody!"))
```

10 threads manipulating one shared data structure, which consists of 100 vectors each one containing 10 (initially sequential) unique numbers. Each thread then repeatedly selects two random positions in two random vectors and swaps them. All changes to the vectors occur in transactions by making use of Clojure's software transactional memory system.

```
(defn run [nvecs nitems nthreads niters]
  (let [vec-refs (->> (range (* nvecs nitems)) (partition nitems) (map (comp ref vec)) vec)
        swap #(let [v1 (rand-int nvecs)
                    v2 (rand-int nvecs)
                    i1 (rand-int nitems)
                    i2 (rand-int nitems)]
                (dosync
                 (let [tmp (nth @(vec-refs v1) i1)]
                   (alter (vec-refs v1) assoc i1 (nth @(vec-refs v2) i2))
                   (alter (vec-refs v2) assoc i2 tmp))))
        report #(let [derefed (map deref vec-refs)]
                  (prn derefed)
                  (println "Distinct:" (->> derefed (apply concat) distinct count)))]
    (report)
    (dorun (apply pcalls (repeat nthreads #(dotimes [_ niters] (swap)))))
    (report)))

(run 100 10 10 100000)
```

Output of prior example:

```
([0 1 2 3 4 5 6 7 8 9] [10 11 12 13 14 15 16 17 18 19] ...
[990 991 992 993 994 995 996 997 998 999])
Distinct: 1000
```

```
([382 318 466 963 619 22 21 273 45 596] [808 639 804 471 394 904 952 75 289 778] ...
[484 216 622 139 651 592 379 228 242 355])
Distinct: 1000
```

See also

- List of JVM languages
- List of CLI languages
- Hy
- Pixie (programming language)

References

1. "Index of /maven2/org/clojure/clojure/1.9.0/" (http://central.maven.org/maven2/org/clojure/clojure/1.9.0/). central.maven.org. 2017-12-08. Retrieved 2017-12-08.
2. "Rich Hickey Q&A on Code Quarterly" (http://codequarterly.com/2011/rich-hickey/). Retrieved 2016-05-08.
3. "Rich Hickey's Amazon Bookshelf of books that influenced Clojure" (https://www.amazon.com/gp/richpub/listmania/fullview/R3LG3ZBZS4GCTH). Retrieved 2016-05-08.
4. Bonnaire-Sergeant, Ambrose (2012). *A Practical Optional Type System for Clojure* (Thesis). The University of Western Australia.
5. "Clojure Programming" (http://cdn.oreilly.com/oreilly/booksamplers/9781449394707_sampler.pdf) (PDF). Retrieved 2013-04-30.
6. "meaning and pronunciation of Clojure" (https://groups.google.com/d/msg/clojure/4uDxeOS8pwY/UHiYp7p1a3YJ). *Rich Hickey*. Retrieved 2012-04-20.

7. "Clojure inventor Hickey now aims for Android" (http://www.infoworld.com/article/2619641/java/clojure-inventor-hickey-now-aims-for-android.html).
8. "Clojure - home" (http://clojure.org/). *clojure.org*. Retrieved 2015-09-15.
9. "clojure/clojure-clr" (https://github.com/clojure/clojure-clr). *GitHub*. Retrieved 2015-09-15.
10. "Clojure - lisp" (http://clojure.org/lisp). *clojure.org*. Retrieved 2015-09-15.
11. "Contributing FAQ - Clojure Community - Clojure Development" (http://dev.clojure.org/display/community/Contributing+FAQ). *dev.clojure.org*. Retrieved 2015-09-15.
12. "Clojure - funding" (http://clojure.org/funding). *clojure.org*. Retrieved 2015-09-15.
13. "clojure/core.typed" (https://github.com/clojure/core.typed/wiki/User-Guide). *GitHub*. Retrieved 2015-09-15.
14. "Clojure - state" (http://clojure.org/state). *clojure.org*. Retrieved 2015-09-15.
15. "Rationale" (http://clojure.org/rationale). *Rich Hickey*. clojure.org. Retrieved 2008-10-17.
16. Charles (2009-10-06). "Expert to Expert: Rich Hickey and Brian Beckman – Inside Clojure | Going Deep | Channel 9" (http://channel9.msdn.com/shows/Going+Deep/Expert-to-Expert-Rich-Hickey-and-Brian-Beckman-Inside-Clojure/). Channel9.msdn.com. Retrieved 2012-06-28.
17. "JUXT Blog: Clojure in London: Funding Circle" (https://juxt.pro/blog/posts/clojure-in-fundingcircle.html). *juxt.pro*. Retrieved 2017-02-01.
18. "Walmart Runs Clojure at Scale" (http://blog.cognitect.com/blog/2015/6/30/walmart-runs-clojure-at-scale). Retrieved 2015-09-15.
19. "A New Era of Application Services at Puppet Labs" (https://puppetlabs.com/blog/new-era-application-services-puppet-labs). *Puppet Labs*. Retrieved 2015-09-15.
20. "Clojure Programming Language :: Cognitect, Clojure Consulting, Clojure Support, Functional Programming, JVM" (http://cognitect.com/clojure). *cognitect.com*. Retrieved 2015-09-15.
21. Clojure/conj (http://clojure-conj.org/)
22. Clojure/West (http://clojurewest.org/)
23. EuroClojure (http://euroclojure.org/)
24. "Clojure 1.9" (http://blog.cognitect.com/blog/clojure19). Retrieved 2017-12-08.
25. "Clojure: Clojure 1.0" (http://clojure.blogspot.fi/2009/05/clojure-10.html). *clojure.blogspot.fi*. Retrieved 2015-09-16.
26. "Clojure - license" (http://clojure.org/license). *clojure.org*. Retrieved 2015-09-15.
27. "[ANN] dotLisp - a Lisp dialect for .Net" (https://groups.google.com/forum/#!topic/comp.lang.scheme/ibf6CC6V66o).
28. "jfli, a Java foreign language interface for Common Lisp" (http://jfli.sourceforge.net).
29. "Foil - a *Foreign Object Interface for Lisp*" (http://foil.sourceforge.net).
30. "Lisplets - a Lisp-friendly interface to Java Servlets" (http://lisplets.sourceforge.net).
31. Clojure Community website (http://dev.clojure.org/display/community/Home)
32. Clojure Dev Google Group (https://groups.google.com/forum/#!forum/clojure-dev)
33. "Contributing FAQ - Clojure Community - Clojure Development" (http://dev.clojure.org/display/community/Contributing+FAQ). *dev.clojure.org*. Retrieved 2015-09-16.
34. "JIRA workflow - Clojure Community - Clojure Development" (http://dev.clojure.org/display/community/JIRA+workflow). *dev.clojure.org*. Retrieved 2015-09-16.
35. "Economy Size Geek - Interview with Rich Hickey, Creator of Clojure | Linux Journal" (http://www.linuxjournal.com/article/10708). *www.linuxjournal.com*. Retrieved 2015-09-15.
36. "On State and Identity" (http://clojure.org/state). *Rich Hickey*. clojure.org. Retrieved 2010-03-01.
37. "Clojure: Clojure is Two!" (http://clojure.blogspot.com/2009/10/clojure-is-two.html). *Clojure Blog*. Retrieved 2015-09-16.

38. Fingerhut, Andy. "Clojure version history" (https://jafingerhut.github.io/clojure-benchmarks-results/Clojure-version-history.html). *jafingerhut.github.io*. Retrieved 2015-09-16.
39. "Clojure: Clojure 1.0" (http://clojure.blogspot.com/2009/05/clojure-10.html). *Clojure Blog*. Retrieved 2015-09-16.
40. "Clojure: Clojure 1.1 Release" (http://clojure.blogspot.com/2009/12/clojure-11-release.html). *clojure.blogspot.com*. Retrieved 2015-09-16.
41. "Clojure - protocols" (http://clojure.org/protocols). *clojure.org*. Retrieved 2015-09-16.
42. "clojure/clojure" (https://github.com/clojure/clojure/blob/master/changes.md#changes-to-clojure-in-version-13). *GitHub*. Retrieved 2015-09-16.
43. "Google Groups" (https://groups.google.com/forum/#!topic/clojure/pArFVr0fJ0w). *groups.google.com*. Retrieved 2015-09-16.
44. "Google Groups" (https://groups.google.com/forum/#!topic/clojure/O307eyvpwn0). *groups.google.com*. Retrieved 2016-01-25.
45. "Google Groups" (https://groups.google.com/forum/#!topic/clojure/kSOO81ddFr0). *groups.google.com*. Retrieved 2017-12-08.
46. "Clojure - jvm_hosted" (http://clojure.org/jvm_hosted). *clojure.org*. Retrieved 2015-09-15.
47. "Clojure - java_interop" (http://clojure.org/java_interop#Java%2520Interop-Calling%2520Clojure%2520From%2520Java). *clojure.org*. Retrieved 2015-09-15.
48. contributors, Phil Hagelberg and. "Leiningen" (http://leiningen.org/). *leiningen.org*. Retrieved 2015-09-15.
49. "Clojure - reader" (http://clojure.org/reader). *clojure.org*. Retrieved 2015-09-15.
50. "Clojure - lisps" (http://clojure.org/lisps). *clojure.org*. Retrieved 2015-09-15.
51. "Clojure - macros" (http://clojure.org/macros). *clojure.org*. Retrieved 2015-09-15.
52. "edn" (https://github.com/edn-format/edn). *Rich Hickey*. Github.com. Retrieved 2014-05-24.
53. "Clojure - multimethods" (http://clojure.org/multimethods). *clojure.org*. Retrieved 2015-09-15.
54. "Clojure - protocols" (http://clojure.org/protocols). *clojure.org*. Retrieved 2015-09-15.
55. "Clojure - datatypes" (http://clojure.org/datatypes). *clojure.org*. Retrieved 2015-09-15.
56. "Brian Goetz - Stewardship: the Sobering Parts" (https://www.youtube.com/watch?v=2y5Pv4yN0b0&t=1h02m18s). *YouTube*. ClojureTV. Retrieved 2015-09-15.
57. "Clojure - special_forms" (http://clojure.org/special_forms#Special%2520Forms--(recur%2520exprs*)). *clojure.org*. Retrieved 2015-09-15.
58. "Clojure - Refs" (http://clojure.org/Refs). *clojure.org*. Retrieved 2015-09-15.
59. "Clojure - Agents" (http://clojure.org/reference/agents). *clojure.org*. Retrieved 2016-07-04.
60. "Clojure :: Clojure core.async Channels" (http://clojure.com/blog/2013/06/28/clojure-core-async-channels.html). *clojure.com*. Retrieved 2015-09-15.
61. "Clojure - reader" (http://clojure.org/reader#toc5). *clojure.org*. Retrieved 2015-09-15.
62. "Transducers" by Rich Hickey. https://www.youtube.com/watch?v=6mTbuzafcII. Retrieved on 2015-09-15.
63. "Transducers are Coming" (http://blog.cognitect.com/blog/2014/8/6/transducers-are-coming). Retrieved 2015-09-15.
64. "Rich Hickey - Inside Transducers" (https://www.youtube.com/watch?v=4KqUvG8HPYo). *YouTube*. Cognitect Inc. Nov 20, 2014. Retrieved 2015-09-15.
65. "clojure/clojurescript" (https://github.com/clojure/clojurescript). *GitHub*. Retrieved 2015-09-15.
66. "clojure/clojure-clr · GitHub" (https://github.com/clojure/clojure-clr). Github.com. Retrieved 2012-06-28.
67. Emerick, Chas. "Results of the 2013 State of Clojure & ClojureScript survey" (http://cemerick.com/2013/11/18/results-of-the-2013-state-of-clojure-clojurescript-survey/). *cemerick*. Retrieved 2015-09-17.

68. "State of Clojure 2014 Survey Results" (https://cognitect.wufoo.com/reports/state-of-clojure-2014-results/). *Cognitect Blog*. Retrieved 2015-09-17.
69. "State of Clojure 2015 Survey Results" (http://blog.cognitect.com/blog/2016/1/28/state-of-clojure-2015-survey-results). *Cognitect Blog*. Retrieved 2016-09-08.
70. "Om: Enhancing Facebook's React with Immutability" (http://www.infoq.com/news/2014/01/om-react). *InfoQ*. Retrieved 2015-09-17.
71. Meier, Carin. "Creative computing with Clojure - O'Reilly Radar" (http://radar.oreilly.com/2015/05/creative-computing-with-clojure.html). *radar.oreilly.com*. Retrieved 2015-09-17.
72. aemoncannon (2010-12-30). "Home · aemoncannon/las3r Wiki · GitHub" (https://github.com/aemoncannon/las3r/wiki). Github.com. Retrieved 2012-06-28.
73. "drewr/clojure-py · GitHub" (https://github.com/drewr/clojure-py). Github.com. Retrieved 2017-01-10.
74. "rouge-lang/rouge · GitHub" (https://github.com/ecmendenhall/rouge). Github.com. Retrieved 2015-12-19.
75. "A lisp on Perl . MetaCPAN" (https://metacpan.org/pod/CljPerl). metacpan.org. Retrieved 2014-05-25.

Further reading

- Rochester, Eric (2015), *Clojure Data Analysis Cookbook* (https://www.packtpub.com/application-development/clojure-data-analysis-cookbook-second-edition) (2nd ed.), Packt Publishing, ISBN 9781784390297
- Gamble, Julian (2015), *Clojure Recipes* (http://www.informit.com/store/clojure-recipes-9780321927736) (1st ed.), Pearson Publishing, ISBN 9780321927736
- Rochester, Eric (2014), *Mastering Clojure Data Analysis* (https://www.packtpub.com/big-data-and-business-intelligence/mastering-clojure-data-analysis) (1st ed.), Packt Publishing, ISBN 9781783284139
- Fogus, Michael; Houser, Chris (2014), *The Joy of Clojure* (2nd ed.), Manning, ISBN 1-617291-41-2
- Fogus, Michael; Houser, Chris (2010), *The Joy of Clojure* (1st ed.), Manning, ISBN 1-935182-64-1
- Halloway, Stuart (2012), *Programming Clojure* (2nd ed.), Pragmatic Bookshelf, ISBN 978-1-93435-686-9
- Rathore, Amit (2011), *Clojure in Action* (1st ed.), Manning, ISBN 1-935182-59-5
- VanderHart, Luke; Sierra, Stuart (June 7, 2010), *Practical Clojure* (1st ed.), Apress, ISBN 1-4302-7231-7
- Emerick, Chas; Carper, Brian; Grand, Christophe (April 19, 2012), *Clojure Programming* (1st ed.), O'Reilly Media, ISBN 1-4493-9470-1

External links

- Official website (http://clojure.org)
- GitHub code repository for Clojure (https://github.com/clojure/clojure/)
- A comprehensive overview of Clojure (http://java.ociweb.com/mark/clojure/article.html)
- An overview of Clojure 1.2 in reference format (http://faustus.webatu.com/clj-quick-ref.html)
- Full Disclojure – Screencast (http://vimeo.com/channels/fulldisclojure)
- Clojure talks on Youtube (https://www.youtube.com/user/clojuretv)
- Clojure talks on Blip.tv (https://web.archive.org/web/20100316101626/http://clojure.blip.tv/posts?view=archive&nsfw=dc)
- clojuredocs.org – Community-powered documentation and examples (http://clojuredocs.org/)
- clojure-doc.org – Community-driven documentation site for the Clojure programming language (http://clojure-doc.org/)
- 4clojure.com – Interactive Clojure Problems (http://4clojure.com/)
- TryClojure – An online REPL for Clojure (http://tryclj.com/)
- Clojure on infoq.com (http://www.infoq.com/clojure/)
- Clojure community and resources on Facebook (http://www.facebook.com/clojurian)

- R.Hickey presentation "Are We There Yet?" where he advocates for the reexamination of basic principles like state, identity, value, time, types, genericity, complexity, as they are used by OOP. 2009 (http://www.infoq.com/presentations/Are-We-There-Yet-Rich-Hickey)
