

Clojure

Clojure (/ˈkloʊʒər/, like *closure*)^{[15][16]} is a modern, dynamic, and functional dialect of the Lisp programming language on the Java platform.^{[17][18]} Like other Lisp dialects, Clojure treats code as data and has a Lisp macro system.^[19] The current development process is community-driven,^[20] overseen by Rich Hickey as its benevolent dictator for life (BDFL).^[21]

Clojure advocates immutability and immutable data structures and encourages programmers to be explicit about managing identity and its states.^[22] This focus on programming with immutable values and explicit progression-of-time constructs is intended to facilitate developing more robust, especially concurrent, programs that are simple and fast.^{[23][24][15]} While its type system is entirely dynamic, recent efforts have also sought the implementation of gradual typing.^[25]

Commercial support for Clojure is provided by Cognitect.^[26] Clojure conferences are organized every year across the globe, the most famous of them being Clojure/conj.^[27]

Contents

History and development process

Design philosophy

Language overview

Alternate Platforms

Other Implementations

Popularity

Tools

Features by example

Simplicity

Programming at REPL

Names at runtime

Code as data (homoiconicity)

Expressive operators for data transformation

Thread-safe management of identity and state

Macros

Language interoperability with Java

Software transactional memory

See also

References

Clojure



Paradigm	multi-paradigm: agent-oriented ^[1] • <u>concurrent</u> ^{[2][3]} • <u>functional</u> ^[4] • <u>logic</u> ^[5] • <u>macro</u> ^[6] • <u>pipeline</u> ^[7]
Family	<u>Lisp</u>
Designed by	Rich Hickey
First appeared	2007
Stable release	1.10.1 ^[8] / 6 June 2019
Preview release	1.10.2-alpha1 / 5 March 2020
Typing discipline	<u>dynamic</u> • <u>strong</u>
Platform	<u>Java</u> • <u>JavaScript</u> • <u>.NET</u>
License	<u>Eclipse Public License</u>
Filename extensions	.clj • .cljs • .cljc • .edn
Website	<u>clojure.org</u> (<u>http</u> <u>s://clojure.org</u>)
Influenced by	
C# • C++ ^[9] • <u>Common Lisp</u> • <u>Erlang</u> • <u>Haskell</u> • <u>Java</u> • <u>Mathematica</u> ^[10] •	


[Further reading](#)

[External links](#)

[ML](#) · [Prolog](#) · [Racket](#)^[11] · [Ruby](#)^[12] · [Scheme](#)

Influenced

[Elixir](#) · [Hy](#) · [LFE](#) · [Pixie](#)^[13] · [Rhine](#)^[14]

 [Clojure Programming at Wikibooks](#)

History and development process

Rich Hickey is the creator of the Clojure language.^[17] Before Clojure, he developed dotLisp, a similar project based on the .NET platform,^[28] and three earlier attempts to provide interoperability between Lisp and Java: a *Java foreign language interface for Common Lisp* (jfli),^[29] A *Foreign Object Interface for Lisp* (FOIL),^[30] and a *Lisp-friendly interface to Java Servlets* (Lisplets).^[31]

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^[20] and is managed at the Clojure [JIRA](#) project page.^[32] General development discussion occurs at the Clojure Google Group.^[33] Anyone can submit issues and ideas at ask.clojure.org,^[34] but to contribute patches, one must sign the Clojure Contributor agreement.^[35] JIRA issues are processed by a team of screeners and finally Rich Hickey approves the changes.^[36]

Clojure's name, according to Hickey, is a pun on the programming concept "closure" incorporating the letters C, L, and J for C#, Lisp, and Java respectively—three languages which had a major influence on Clojure's design.^[37]

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.^{[23][24][38][15]}

Clojure's approach to state is characterized by the concept of identities,^[22] 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.^[22]

Language overview



Rich Hickey, creator of Clojure

Version	Release date	Major features/improvements
	October 16, 2007 ^[39]	Initial public release
1.0	May 4, 2009 ^[40]	First stable release
1.1	December 31, 2009 ^[41]	Futures
1.2	August 19, 2010 ^[42]	Protocols
1.3	September 23, 2011 ^[43]	Enhanced primitive support
1.4	April 15, 2012 ^[44]	Reader literals
1.5	March 1, 2013 ^[45]	Reducers
1.5.1	March 10, 2013 ^[46]	Fixing a memory leak
1.6	March 25, 2014 ^[47]	Java API, improved hashing algorithms
1.7	June 30, 2015 ^[48]	Transducers, reader conditionals
1.8	January 19, 2016 ^[49]	Additional string functions, direct linking, socket server
1.9	December 8, 2017 ^[50]	Integration with spec, command-line tools
1.10	December 17, 2018 ^[51]	Improved error reporting, Java compatibility
1.10.1	June 6, 2019 ^[8]	Working around a Java performance regression and improving error reporting from clojure.main
 Latest version		

Clojure runs on the Java platform and as a result, integrates with Java and fully supports calling Java code from Clojure,^{[52][15]} and Clojure code can be called from Java also.^[53] The community uses Leiningen for project automation, providing support for Maven integration. Leiningen handles project package management and dependencies and is configured using Clojure syntax.^[54]

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.^{[55][15]} Clojure's reader supports literal syntax for maps, sets and vectors in addition to lists, and these are compiled to the mentioned structures directly.^[55] 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.^[19]

As a Lisp dialect, Clojure supports functions as first-class objects, a read-eval-print loop (REPL), and a macro system.^[6] Clojure's Lisp macro system is very similar to that in Common Lisp with the exception that Clojure's version of the backquote (termed "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.^[56] Clojure supports multimethods^[57] and for interface-like abstractions has a protocol^[58] based polymorphism and data type system using records,^[59] providing high-performance and dynamic polymorphism designed to avoid the expression problem.

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;^{[60][61][62]} it is possible to do so explicitly by using the `recur` keyword.^[63] For parallel and concurrent programming Clojure provides software transactional memory,^[64] a reactive agent system,^[1] and channel-based concurrent programming.^[65]

Clojure 1.7 introduced reader conditionals by allowing the embedding of Clojure and ClojureScript code in the same namespace.^{[48][55]} Transducers were added as a method for composing transformations. Transducers enable higher-order functions such as `map` and `fold` to generalize over any source of input data. While traditionally these functions operate on sequences, transducers allow them to work on channels and let the user define their own models for transduction.^{[66][67][68]}

Alternate Platforms

The primary platform of Clojure is Java,^{[18][52]} but other target implementations exist. The most notable of these is ClojureScript,^[69] which compiles to ECMAScript 3,^[70] and ClojureCLR,^[71] a full port on the .NET platform, interoperable with its ecosystem. A survey of the Clojure community with 1,060 respondents conducted in 2013^[72] found that 47% of respondents used both Clojure and ClojureScript when working with Clojure. In 2014 this number had increased to 55%,^[73] in 2015, based on 2,445 respondents, to 66%.^[74] Popular ClojureScript projects include implementations of the React library such as Reagent,^[75] re-frame,^[76] Rum,^[77] and Om.^{[78][79]}

Other Implementations

Other implementations of Clojure on different platforms include:

- CljPerl,^[80] Clojure atop Perl
- Clojerl,^[81] Clojure on BEAM, the Erlang virtual machine
- clojure-py,^[82] Clojure in pure Python
- Ferret,^[83] compiles to self-contained C++11 that can run on microcontrollers
- Joker,^[84] an interpreter and linter written in Go
- Las3r,^[85] a subset of Clojure that runs on the ActionScript Virtual Machine (the Adobe Flash Player platform)
- Pixie,^[86] Clojure-inspired Lisp dialect written in RPython
- Rouge,^[87] Clojure atop YARV in Ruby

Popularity

With continued interest in functional programming, Clojure's adoption by software developers using the Java platform has continued to increase. The language has also been recommended by software developers such as Brian Goetz,^{[88][89][90]} Eric Evans,^{[91][92]} James Gosling,^[93] Paul Graham,^[94] and Robert C. Martin.^{[95][96][97][98]} ThoughtWorks, while assessing functional programming languages for their Technology Radar,^[99] described Clojure as "a simple, elegant implementation of Lisp on the JVM" in 2010 and promoted its status to "ADOPT" in 2012.^[100]

In the "JVM Ecosystem Report 2018" (which was claimed to be "the largest survey ever of Java developers"), that was prepared in collaboration by Snyk and Java Magazine, ranked Clojure as the 2nd most used programming language on the JVM for "main applications".^[101] Clojure is used in industry by firms^[102] such as Apple,^{[103][104]} Atlassian,^[105] Funding Circle,^[106] Netflix,^[107] Puppet,^[108] and Walmart^[109] as well as government agencies such as NASA.^[110] It has also been used for creative computing, including visual art, music, games, and poetry.^[111]

Tools

Tooling for Clojure development has seen significant improvement over the years. The following is a list of some popular IDEs and text editors with plug-ins that add support Clojure development:^[112]

- Atom, with Chlorine^[113]
- Emacs, with CIDER^[114]
- IntelliJ IDEA, with Clojure-Kit^[115] or Cursive^[116] (a free license is available for non-commercial use)
- Light Table
- Vim, with fireplace.vim,^{[117][118]} vim-iced,^[119] or Conjure (Neovim only)^{[120][121]}
- Visual Studio Code, with Calva^[122]

In addition to the tools provided by the community, the official Clojure CLI tools^[123] have also become available on GNU/Linux, macOS, and Windows since Clojure 1.9.^[124]

Features by example

The following examples can be run in a Clojure REPL such as one started with the Clojure CLI tools^[123] or an online REPL such as one available on REPL.it.^[125]

Simplicity

Because of the strong emphasis on simplicity, typical Clojure programs consist of mostly functions and simple data structures (i.e., lists, vectors, maps, and sets):

```
;; A typical entry point of a Clojure program:
;; ~~~~~
;;   `~main` function
(defn -main ; name
  [& args] ; (variable) parameters
  (println "Hello, World!")) ; body
```

Programming at REPL

Like other Lisps, one of the iconic features of Clojure is interactive programming at the REPL.^[126] Note that, in the following examples, `;;` starts a line comment and `;; =>` indicates output:

```
;; define a var
(def a 42)
;; => #'user/a

;; call a function named `+`
(+ a 8)
```

```

;; => 50

;; call a function named `even?`
(even? a)
;; => true

;; define a function that returns the remainder of `n` when divided by 10
(defn foo [n] (rem n 10))
;; => #'user/foo

;; call the function
(foo a)
;; => 2

;; print the docstring of `rem`
(doc rem)
;; =>
-----
clojure.core/rem
([num div])
  remainder of dividing numerator by denominator.

;; print the source of `rem`
(source rem)
;; =>
(defn rem
  "remainder of dividing numerator by denominator."
  {:added "1.0"
   :static true
   :inline (fn [x y] `( . clojure.lang.Numbers (remainder ~x ~y)))}
  [num div]
    (. clojure.lang.Numbers (remainder num div)))

```

Names at runtime

Unlike other runtime environments where names get compiled away, Clojure's runtime environment is easily introspectable using normal Clojure data structures:

```

;; define a var
(def a 42)
;; => #'user/a

;; get a map of all public vars interned in the `user` namespace
(ns-publics 'user)
;; => {a #'user/a}

;; reference the var via `#'` (reader macro) and
;; its associated, namespace-qualified symbol `user/a`
#'user/a
;; => #'user/a

;; de-reference (get the value of) the var
(deref #'user/a)
;; => 42

;; define a function (with a docstring) that
;; returns the remainder of `n` when divided by 10
(defn foo "returns `(rem n 10)`" [n] (rem n 10))
;; => #'user/foo

;; get the metadata of the var `#'user/foo`
(meta #'user/foo)
;; =>
{:arglists ([n]),
 :doc "returns `(rem n 10)`",
 :line 1,
 :column 1,
 :file "user.clj",
 :name foo,
 :ns #namespace[user]}

```

Code as data (homoiconicity)

Similar to other Lisps, Clojure is homoiconic (also known as code as data). In the example below, we can see how easy it is to write code that modifies code itself:

```
;; call a function (code)
(+ 1 1)
;; => 2

;; quote the function call
;; (turning code into data, which is a list of symbols)
(quote (+ 1 1))
;; => (+ 1 1)

;; get the first element on the list
;; (operating on code as data)
(first (quote (+ 1 1)))
;; => +

;; get the last element on the list
;; (operating on code as data)
(last (quote (+ 1 1)))
;; => 1

;; get a new list by replacing the symbols on the original list
;; (manipulating code as data)
(map (fn [form]
      (case form
        1 'one
        + 'plus))
     (quote (+ 1 1)))
;; => (plus one one)
```

Expressive operators for data transformation

The threading macros (->, ->>, and friends) can syntactically express the abstraction of piping a collection of data through a series of transformations:

```
(->> (range 10)
     (map inc)
     (filter even?))
;; => (2 4 6 8 10)
```

This can also be achieved more efficiently using transducers:

```
(sequence (comp (map inc)
                (filter even?))
          (range 10))
;; => (2 4 6 8 10)
```

Thread-safe management of identity and state

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):

```
(def i (atom 0))

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

Macros

An anonymous subclass of `java.io.Writer` 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 `forms` with all printing to `*out*` silenced."
  [& forms]
  `(binding [*out* bit-bucket-writer]
    ~@forms))

(noprint
  (println "Hello, nobody!"))
;; => nil
```

Language interoperability with Java

Clojure was created from the ground up to embrace its host platforms as one of its design goals and thus provides excellent language interoperability with Java:

```
;; call an instance method
(.toUpperCase "apple")
;; => "APPLE"

;; call a static method
(System/getProperty "java.vm.version")
;; => "12+33"

;; create an instance of `java.util.HashMap` and
;; add some entries
(doto (java.util.HashMap.)
  (.put "apple" 1)
  (.put "banana" 2))
;; => {"banana" 2, "apple" 1}

;; create an instance of `java.util.ArrayList` and
;; increment its elements with `clojure.core/map`
(def al (doto (java.util.ArrayList.)
  (.add 1)
  (.add 2)
  (.add 3)))

(map inc al)
;; => (2 3 4)

;; show a message dialog using Java Swing
(javax.swing.JOptionPane/showMessageDialog
  nil
  "Hello, World!")
;; => nil
```

Software transactional memory

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
        (->> (* nvecs nitems)
              (range)
              (into [] (comp (partition-all nitems)
                             (map vec)
                             (map ref)))))]

    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
    #(->> vec-refs
      (into [] (comp (map deref)
                     (map (fn [v] (prn v) v))
                     cat
                     (distinct))))
      (count)
      (println "Distinct:"))])

  (report))

(->> #(dotimes [_ niters] (swap)))
(repeat nthreads)
(apply pcalls)
(dorun))

(run 100 10 10 100000)
;; =>
[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
nil

```

See also

- [List of JVM languages](#)
- [List of CLI languages](#)
- [Comparison of programming languages](#)

References

1. "Agents and Asynchronous Actions" (<https://clojure.org/reference/agents>). *Clojure.org*. Retrieved 2019-07-07.
2. "Concurrent Programming" (https://clojure.org/about/concurrent_programming). *Clojure.org*. Retrieved 2019-07-07.
3. Hickey, Rich; contributors. "core.async" (<https://github.com/clojure/core.async>). *GitHub.com*. Retrieved 2019-07-07.
4. "Functional Programming" (https://clojure.org/about/functional_programming). *Clojure.org*. Retrieved 2019-07-07.

5. Nolen, David; Hickey, Rich; contributors. "[core.logic](https://github.com/clojure/core.logic/)" (<https://github.com/clojure/core.logic/>). *GitHub.com*. Retrieved 2019-07-07.
6. "[Macros](https://clojure.org/reference/macros)" (<https://clojure.org/reference/macros>). *Clojure.org*. Retrieved 2019-07-07.
7. Esterhazy, Paulus. "[Threading Macros Guide](https://clojure.org/guides/threading_macros)" (https://clojure.org/guides/threading_macros). *Clojure.org*. Retrieved 2019-07-07.
8. Miller, Alex (2019-06-06). "[Clojure 1.10.1 release](https://clojure.org/news/2019/06/06/clojure1-10-1)" (<https://clojure.org/news/2019/06/06/clojure1-10-1>). *Clojure.org*.
9. Fogus, Michael (2011). "[Rich Hickey Q&A](https://web.archive.org/web/20170111184835/http://www.codequarterly.com/2011/rich-hickey/)" (<https://web.archive.org/web/20170111184835/http://www.codequarterly.com/2011/rich-hickey/>). *CodeQuarterly.com*. Archived from the original (<http://www.codequarterly.com/2011/rich-hickey/>) on 2017-01-11.
10. Hickey, Rich. "[Clojure Bookshelf](https://web.archive.org/web/20171003001051/https://www.amazon.com/gp/richpub/listmania/fullview/R3LG3ZBZS4GCTH)" (<https://web.archive.org/web/20171003001051/https://www.amazon.com/gp/richpub/listmania/fullview/R3LG3ZBZS4GCTH>). *Amazon.com*. Archived from the original (<https://www.amazon.com/gp/richpub/listmania/fullview/R3LG3ZBZS4GCTH>) on 2017-10-03. Retrieved 2019-07-07.
11. Bonnaire-Sergeant, Ambrose (2012). *A Practical Optional Type System for Clojure* (Thesis). The University of Western Australia.
12. "[Clojure Programming](http://cdn.oreilly.com/oreilly/booksamplers/9781449394707_sampler.pdf)" (http://cdn.oreilly.com/oreilly/booksamplers/9781449394707_sampler.pdf) (PDF). *OReilly.com*. Retrieved 2013-04-30.
13. Baldridge, Timothy. "[Pixie](https://www.pixielang.org/)" (<https://www.pixielang.org/>). *PixieLang.org*. Retrieved 2019-07-07.
14. Ramachandra, Ramkumar. "[Rhine](https://github.com/artagnon/rhine-ml)" (<https://github.com/artagnon/rhine-ml>). *GitHub.org*. Retrieved 2019-07-07.
15. Edwards, Kathryn (2009-08-10). "[The A-Z of Programming Languages: Clojure](https://www.computerworld.com.au/article/313989/a-z_programming_languages_clojure/)" (https://www.computerworld.com.au/article/313989/a-z_programming_languages_clojure/). *Computerworld.com.au*.
16. Hickey, Rich (2009-01-05). "[meaning and pronunciation of Clojure](https://groups.google.com/d/msg/clojure/4uDxeOS8pwY/UHiYp7p1a3YJ)" (<https://groups.google.com/d/msg/clojure/4uDxeOS8pwY/UHiYp7p1a3YJ>). *Google.com*.
17. Krill, Paul (2012-03-22). "[Clojure inventor Hickey now aims for Android](https://www.infoworld.com/article/2619641/clojure-inventor-hickey-now-aims-for-android.html)" (<https://www.infoworld.com/article/2619641/clojure-inventor-hickey-now-aims-for-android.html>). *InfoWorld.com*.
18. "[Clojure](https://clojure.org/)" (<https://clojure.org/>). *Clojure.org*. Retrieved 2019-07-07.
19. "[Differences with other Lisps](https://clojure.org/reference/lisps)" (<https://clojure.org/reference/lisps>). *Clojure.org*. Retrieved 2019-07-07.
20. "[Development](https://clojure.org/dev/dev)" (<https://clojure.org/dev/dev>). *Clojure.org*. Retrieved 2019-07-07.
21. Hickey, Rich (2018-11-26). "[Open Source is Not About You](https://gist.github.com/richhickey/1563cddea1002958f96e7ba9519972d9)" (<https://gist.github.com/richhickey/1563cddea1002958f96e7ba9519972d9>). *GitHub.com*.
22. "[Values and Change: Clojure's approach to Identity and State](https://clojure.org/about/state)" (<https://clojure.org/about/state>). *Clojure.org*. Retrieved 2019-07-07.
23. Hickey, Rich. "[Rationale](https://clojure.org/about/rationale)" (<https://clojure.org/about/rationale>). *Clojure.org*. Retrieved 2019-07-07.
24. Torre, Charles (2009-10-06). "[Expert to Expert: Rich Hickey and Brian Beckman – Inside Clojure](https://channel9.msdn.com/shows/Going+Deep/Expert-to-Expert-Rich-Hickey-and-Brian-Beckman-Inside-Clojure/)" (<https://channel9.msdn.com/shows/Going+Deep/Expert-to-Expert-Rich-Hickey-and-Brian-Beckman-Inside-Clojure/>). *MSDN.com*.
25. "[core.typed](https://github.com/clojure/core.typed)" (<https://github.com/clojure/core.typed>). *GitHub.com*. Retrieved 2019-07-07.
26. "[Investing in A Better Way](https://cognitect.com/technologies.html)" (<https://cognitect.com/technologies.html>). *Cognitect.com*. Retrieved 2019-07-07.
27. "[Clojure/conj](http://clojure-conj.org/)" (<http://clojure-conj.org/>). *Clojure-Conj.org*. Retrieved 2019-07-07.
28. Hickey, Rich (2002-10-16). "[\[ANN\] dotLisp – a Lisp dialect for .Net](https://groups.google.com/forum/#!topic/comp.lang.scheme/ibf6CC6V66o)" (<https://groups.google.com/forum/#!topic/comp.lang.scheme/ibf6CC6V66o>). *Google.com*.
29. Hickey, Rich (2013-04-15). "[jfli](https://sourceforge.net/projects/jfli/)" (<https://sourceforge.net/projects/jfli/>). *SourceForge.net*.

30. Hickey, Rich (2013-04-03). "foil - Foreign Object Interface for Lisp" (<https://sourceforge.net/projects/foil/>). *SourceForge.net*.
31. Hickey, Rich (2013-03-07). "Lisplets" (<https://sourceforge.net/projects/lisplets/>). *SourceForge.net*.
32. "Clojure" (<https://clojure.atlassian.net/projects/CLJ>). *Atlassian.net*. Retrieved 2019-07-07.
33. "Clojure" (<https://groups.google.com/forum/#!forum/clojure>). *Google.com*. Retrieved 2019-07-07.
34. "Clojure Forum" (<https://clojure.org/news/2019/07/25/clojure-forum>). *clojure.org*. Retrieved 2020-03-20.
35. "Contributor Agreement" (https://clojure.org/dev/contributor_agreement). *Clojure.org*. Retrieved 2019-07-07.
36. "Workflow" (<https://clojure.org/dev/workflow>). *Clojure.org*. Retrieved 2019-07-07.
37. Rich, Hickey (2009-01-05). "Clojure Mailing List: meaning and pronunciation of Clojure" (<https://groups.google.com/forum/#!msg/clojure/4uDxeOS8pwY/UHiYp7p1a3YJ>). *groups.google.com*. Archived (<https://archive.today/20200109152435/https://groups.google.com/forum/#!msg/clojure/4uDxeOS8pwY/UHiYp7p1a3YJ>) from the original on 2020-01-09. Retrieved 2020-01-09.
38. Elmendorf, Dirk (2010-04-01). "Economy Size Geek – Interview with Rich Hickey, Creator of Clojure" (<https://www.linuxjournal.com/article/10708>). *LinuxJournal.com*.
39. Hickey, Rich (2007-10-16). "Clojure is Two!" (<https://clojure.blogspot.com/2009/10/clojure-is-two.html>). *BlogSpot.com*.
40. Hickey, Rich (2009-05-04). "Clojure 1.0" (<https://clojure.blogspot.com/2009/05/clojure-10.html>). *BlogSpot.com*.
41. Hickey, Rich (2009-12-31). "Clojure 1.1 Release" (<https://clojure.blogspot.com/2009/12/clojure-11-release.html>). *BlogSpot.com*.
42. Hickey, Rich (2010-08-19). "Clojure 1.2 Release" (<https://groups.google.com/forum/#!topic/clojure/tXII-vxyJpc>). *Google.com*.
43. Redinger, Christopher (2011-09-23). "[ANN] Clojure 1.3 Released" (<https://groups.google.com/forum/#!topic/clojure/w5Nmx5rPaQs>). *Google.com*.
44. Dipert, Alan (2012-04-17). "[ANN] Clojure 1.4 Released" (<https://groups.google.com/forum/#!topic/clojure/H4f2nbB6gWI>). *Google.com*.
45. Halloway, Stuart (2013-03-01). "ANN: Clojure 1.5" (<https://groups.google.com/forum/#!topic/clojure/kzF5O0Yfdhc>). *Google.com*.
46. Halloway, Stuart (2013-03-10). "Clojure 1.5.1" (<https://groups.google.com/forum/#!topic/clojure/PDENUpc44IY>). *Google.com*.
47. Miller, Alex (2014-03-25). "[ANN] Clojure 1.6" (<https://groups.google.com/forum/#!topic/clojure/pArFVr0fJ0w>). *Google.com*.
48. Miller, Alex (2015-06-30). "Clojure 1.7 is now available" (<https://clojure.org/news/2015/06/30/clojure-17>). *Clojure.org*.
49. Miller, Alex (2016-01-19). "Clojure 1.8 is now available" (<https://clojure.org/news/2016/01/19/clojure18>). *Clojure.org*.
50. Miller, Alex (2017-12-08). "Clojure 1.9 is now available" (<https://clojure.org/news/2017/12/08/clojure19>). *Clojure.org*.
51. Miller, Alex (2018-12-17). "Clojure 1.10 release" (<https://clojure.org/news/2018/12/17/clojure110>). *Clojure.org*.
52. "Hosted on the JVM" (https://clojure.org/about/jvm_hosted). *Clojure.org*. Retrieved 2019-07-07.
53. "Java Interop" (https://clojure.org/reference/java_interop#_calling_clojure_from_java). *Clojure.org*. Retrieved 2019-07-07.
54. Hagelberg, Phil; contributors. "Leiningen" (<https://leiningen.org/>). *Leiningen.org*. Retrieved 2019-07-07.

55. "The Reader" (<https://clojure.org/reference/reader>). *Clojure.org*. Retrieved 2019-07-07.
56. Hickey, Rich. "edn" (<https://github.com/edn-format/edn>). *GitHub.com*. Retrieved 2019-07-07.
57. "Multimethods and Hierarchies" (<https://clojure.org/reference/multimethods>). *Clojure.org*. Retrieved 2019-07-07.
58. "Protocols" (<https://clojure.org/reference/protocols>). *Clojure.org*. Retrieved 2019-07-07.
59. "Datatypes: deftype, defrecord and reify" (<https://clojure.org/reference/datatypes>). *Clojure.org*. Retrieved 2019-07-07.
60. Goetz, Brian (2014-11-20). "Stewardship: the Sobering Parts" (<https://www.youtube.com/watch?v=2y5Pv4yN0b0&t=1h02m18s>). *YouTube.com*.
61. Rose, John (2007-07-12). "tail calls in the VM" (<https://blogs.oracle.com/jrose/tail-calls-in-the-vm>). *Oracle.com*.
62. Rose, John (2009-02-11). "Some languages need to be able to perform tail calls" (<https://bugs.openjdk.java.net/browse/JDK-6804517>). *Java.net*.
63. "Special Forms" (https://clojure.org/reference/special_forms#recur). *Clojure.org*. Retrieved 2019-07-07.
64. "Refs and Transactions" (<https://clojure.org/refs>). *Clojure.org*. Retrieved 2019-07-07.
65. Hickey, Rich (2013-06-28). "Clojure core.async Channels" (<https://clojure.org/news/2013/06/28/clojure-core-async-channels>). *Clojure.org*.
66. Hickey, Rich (2014-09-17). "Transducers" (<https://www.youtube.com/watch?v=6mTbuzafclI>). *YouTube.com*.
67. Hickey, Rich (2014-08-06). "Transducers are Coming" (<https://blog.cognitect.com/blog/2014/8/6/transducers-are-coming>). *Cognitect.com*.
68. Hickey, Rich (2014-11-20). "Inside Transducers" (<https://www.youtube.com/watch?v=4KqUvG8HPYo>). *YouTube.com*.
69. "ClojureScript" (<https://clojurescript.org>). *ClojureScript.org*. Retrieved 2019-07-06.
70. "ClojureScript – FAQ (for JavaScript developers)" (<https://clojurescript.org/guides/faq-js#does-clojurescript-work-in-old-browsers>). *ClojureScript.org*. Retrieved 2018-02-04.
71. "ClojureCLR" (<https://github.com/clojure/clojure-clr>). *GitHub.com*. Retrieved 2012-06-28.
72. Emerick, Chas (2013-11-18). "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.com*.
73. "State of Clojure 2014 Survey Results" (<https://cognitect.wufoo.com/reports/state-of-clojure-2014-results/>). *WuFoo.com*. Retrieved 2015-09-17.
74. Gehrtland, Justin (2016-01-28). "State of Clojure 2015 Survey Results" (<https://blog.cognitect.com/blog/2016/1/28/state-of-clojure-2015-survey-results>). *Cognitect.com*.
75. "Reagent" (<https://reagent-project.github.io/>). *GitHub.io*. Retrieved 2019-07-06.
76. "re-frame" (<https://github.com/Day8/re-frame>). *GitHub.com*. Retrieved 2019-07-06.
77. Prokopov, Nikita. "Rum" (<https://github.com/tonsky/rum>). *GitHub.com*. Retrieved 2019-07-06.
78. Nolen, David. "Om" (<https://github.com/omcljs/om>). *GitHub.com*. Retrieved 2019-07-06.
79. Coupland, Tom (2014-01-17). "Om: Enhancing Facebook's React with Immutability" (<https://www.infoq.com/news/2014/01/om-react/>). *InfoQ.com*.
80. Hu, Wei. "A Lisp on Perl" (<https://metacpan.org/pod/CljPerl>). *MetaCPAN.org*. Retrieved 2019-07-06.
81. Facorro, Juan. "Clojerl" (<https://github.com/clojerl/clojerl>). *GitHub.com*. Retrieved 2019-07-06.
82. Baldridge, Timothy. "clojure-py" (<https://github.com/drewr/clojure-py>). *GitHub.com*. Retrieved 2019-07-06.
83. Akkaya, Nurullah. "Ferret" (<https://ferret-lang.org/>). *Ferret-Lang.org*. Retrieved 2019-07-06.
84. Bataev, Roman. "Joker" (<https://joker-lang.org/>). *Joker-Lang.org*. Retrieved 2019-07-06.

85. Cannon, Aemon. "Laz3r" (<https://github.com/aemoncannon/laz3r>). *GitHub.com*. Retrieved 2019-07-06.
86. Baldrige, Timothy. "Pixie" (<http://www.pixielang.org/>). *PixieLang.org*. Retrieved 2019-07-06.
87. Connor, Ashe. "Rouge" (<https://github.com/ecmendenhall/rouge>). *GitHub.com*. Retrieved 2019-07-06.
88. Goetz, Brian (2020-05-24). "Brian Goetz' favorite non-Java JVM language (Part 1 of 3)" (<https://www.twitch.tv/nipafx/clip/BloodyUglySharkFailFish>). *Twitch.tv*.
89. Goetz, Brian (2020-05-24). "Brian Goetz' favorite non-Java JVM language (Part 2 of 3)" (<https://www.twitch.tv/nipafx/clip/GrotesqueWonderfulPigeonEleGiggle>). *Twitch.tv*.
90. Goetz, Brian (2020-05-24). "Brian Goetz' favorite non-Java JVM language (Part 3 of 3)" (<https://www.twitch.tv/nipafx/clip/EphemeralAdorableWalletGingerPower>). *Twitch.tv*.
91. Evans, Eric (2018-08-14). "Modelling Time - Eric Evans - Domain-Driven Design Europe 2018" (<https://www.youtube.com/watch?v=T29WzvaPNc8&t=926>). *YouTube.com*.
92. Evans, Eric (2014-11-21). "Eric Evans on Twitter" (<https://twitter.com/ericevans0/status/535742147098853376>). *Twitter.com*.
93. "James Gostling meetup with London Java Community" (<https://www.youtube.com/watch?v=-ktUXFkSkI&t=24m14s>). *YouTube.com*. 2016-10-11.
94. Graham, Paul (2016-05-06). "Paul Graham on Twitter" (<https://twitter.com/paulg/status/728831131534024704>). *Twitter.com*.
95. Martin, Robert (2019-08-22). "Why Clojure?" (<http://blog.cleancoder.com/uncle-bob/2019/08/22/WhyClojure.html>). *CleanCoder.com*.
96. Martin, Robert (2018-11-29). "Unble Bob Martin on Twitter" (<https://twitter.com/unclebobmartin/status/1068205421737857024>). *Twitter.com*.
97. Martin, Robert (2018-08-01). "Introduction To Functional Programming" (<https://cleancoders.com/video-details/clean-code-episode-53>). *CleanCoders.com*.
98. Martin, Robert (2017-07-11). "Pragmatic Functional Programming" (<https://blog.cleancoder.com/uncle-bob/2017/07/11/PragmaticFunctionalProgramming.html>). *CleanCoder.com*.
99. "Frequently Asked Questions - Technology Radar - ThoughtWorks" (<https://www.thoughtworks.com/radar/faq>). *ThoughtWorks.com*. Retrieved 2019-02-10.
00. "Clojure - Technology Radar - ThoughtWorks" (<https://www.thoughtworks.com/radar/languages-and-frameworks/clojure>). *ThoughtWorks.com*. Retrieved 2019-02-10.
01. Maple, Simon; Binstock, Andrew (2018-10-17). "JVM Ecosystem Report 2018" (<https://snky.io/blog/jvm-ecosystem-report-2018/>). *Snyk.io*.
02. "Success Stories" (https://clojure.org/community/success_stories). *Clojure.org*. Retrieved 2018-10-27.
03. Liutikov, Roman (2017-12-17). "Roman Liutikov on Twitter" (<https://twitter.com/roman01la/status/942469177444569089>). *Twitter.com*.
04. "Jobs at Apple" (<https://jobs.apple.com/en-us/search?sort=relevance&search=clojure>). *Apple.com*. Retrieved 2019-07-06.
05. Borges, Leonardo (2015-07-07). "Realtime Collaboration with Clojure" (<https://www.youtube.com/watch?v=3QR8meTrh5g>). *YouTube.com*.
06. Pither, Jon (2016-10-04). "Clojure in London: Funding Circle – Lending some Clojure" (<https://juxt.pro/blog/posts/clojure-in-fundingcircle.html>). *JUXT.pro*.
07. Williams, Alex (2014-08-03). "The New Stack Makers: Adrian Cockcroft on Sun, Netflix, Clojure, Go, Docker and More" (<https://thenewstack.io/the-new-stack-makers-adrian-cockcroft-on-sun-netflix-clojure-go-docker-and-more/>). *TheNewStack.io*.
08. Price, Chris (2014-04-11). "A New Era of Application Services at Puppet Labs" (<https://puppet.com/blog/a-new-era-of-application-services-at-puppet-labs/>). *Puppet.com*. Retrieved 2020-08-06.

09. Phillips, Marc (2015-07-14). "Walmart Runs Clojure at Scale" (<https://blog.cognitect.com/blog/2015/6/30/walmart-runs-clojure-at-scale>). *Cognitect.com*.
10. "Common-Metadata-Repository" (<https://github.com/nasa/Common-Metadata-Repository>). *GitHub.com*. Retrieved 2019-07-06.
11. Meier, Carin (2015-05-06). "Creative computing with Clojure" (<http://radar.oreilly.com/2015/05/creative-computing-with-clojure.html>). *OReilly.com*.
12. Miller, Alex (2019-02-04). "State of Clojure 2019" Results" (<https://www.clojure.org/news/2019/02/04/state-of-clojure-2019>). *Clojure.org*.
13. Szabo, Maurício. "Chlorine: Socket REPL Client for Clojure and ClojureScript" (<https://atom.io/packages/chlorine>). *Atom.io*. Retrieved 2019-07-05.
14. Batsov, Bozhidar; contributors. "CIDER: The Clojure Interactive Development Environment that Rocks" (<https://cider.mx/>). *CIDER.mx*. Retrieved 2019-07-05.
15. Shrago, Greg. "Clojure-Kit: Clojure and ClojureScript plugin for IntelliJ-based tools" (<https://plugins.jetbrains.com/plugin/8636-clojure-kit>). *JetBrains.com*. Retrieved 2019-07-05.
16. Fleming, Colin. "Cursive: Provides full Clojure and ClojureScript language support" (<https://plugins.jetbrains.com/plugin/8090-cursive>). *JetBrains.com*. Retrieved 2019-07-05.
17. Pope, Tim. "fireplace.vim: Clojure REPL Support" (https://www.vim.org/scripts/script.php?script_id=4978). *VIM.org*. Retrieved 2019-07-05.
18. Monroe, Dominic (2016-12-13). "Clojure and Vim: An overview – It's very possible" (<https://juxt.pro/blog/posts/vim-1.html>). *JUXT.pro*.
19. Masashi, Iizuka. "vim-iced: Clojure Interactive Development Environment for Vim8/Neovim" (<https://liquidz.github.io/vim-iced/>). *GitHub.com*. Retrieved 2020-03-13.
20. Caldwell, Oliver. "Neovim Clojure(Script) tooling over prepl" (<https://github.com/Olical/conjure>). *GitHub.com*. Retrieved 2019-11-09.
21. Caldwell, Oliver (2019-11-06). "Getting started with Clojure, Neovim and Conjure in minutes" (<https://oli.me.uk/getting-started-with-clojure-neovim-and-conjure-in-minutes/>). *oli.me.uk*.
22. Strömberg, Peter. "Calva: Clojure & ClojureScript Interactive Programming" (<https://marketplace.visualstudio.com/items?itemName=betterthantomorrow.calva>). *VisualStudio.com*. Retrieved 2019-07-05.
23. Miller, Alex. "Deps and CLI Guide" (https://clojure.org/guides/deps_and_cli). *Clojure.org*. Retrieved 2019-07-08.
24. Miller, Alex (2017-12-08). "Clojure 1.9" (<http://blog.cognitect.com/blog/clojure19>). *Cognitect.com*.
25. "Online Clojure REPL" (<https://repl.it//clojure>). *REPL.it*. Retrieved 2019-07-08.
26. "Programming at the REPL: Introduction" (<https://clojure.org/guides/repl/introduction>). *Clojure.org*. Retrieved 2018-12-04.

Further reading

- Sotnikov, Dmitri (2020), *Web Development with Clojure* (<https://pragprog.com/titles/dswdcloj3/>) (3rd ed.), Pragmatic Bookshelf, ISBN 978-1-68050-682-2
- Olsen, Russ (2018), *Getting Clojure* (<https://pragprog.com/titles/roclojure/>), Pragmatic Bookshelf, ISBN 978-1-68050-300-5
- Miller, Alex; Halloway, Stuart; Bedra, Aaron (2018), *Programming Clojure* (<https://pragprog.com/titles/shcloj3/>) (3rd ed.), Pragmatic Bookshelf, ISBN 978-1-68050-246-6
- Rathore, Amit; Avila, Francis (2015), *Clojure in Action* (2nd ed.), Manning, ISBN 978-1-61729-152-4
- Higginbotham, Daniel (2015), *Clojure for the Brave and True* (<https://www.braveclojure.com/clojure-for-the-brave-and-true/>), No Starch Press, ISBN 978-1-59327-591-4

- Gamble, Julian (2015), *Clojure Recipes* (<http://www.informit.com/store/clojure-recipes-9780321927736>), Pearson Publishing, ISBN 978-0-32192-773-6
- Vandgrift, Ben; Miller, Alex (2015), *Clojure Applied* (<https://pragprog.com/titles/vmclojeco/>), Pragmatic Bookshelf, ISBN 978-1-68050-074-5
- Rochester, Eric (2015), *Clojure Data Analysis Cookbook* (<https://www.packtpub.com/application-development/clojure-data-analysis-cookbook-second-edition>) (2nd ed.), Packt Publishing, ISBN 978-1-78439-029-7
- Fogus, Michael; Houser, Chris (2014), *The Joy of Clojure* (<http://www.joyofclojure.com/>) (2nd ed.), Manning, ISBN 1-617291-41-2
- Kelker, Ryan (2013), *Clojure for Domain-specific Languages* (<https://www.packtpub.com/application-development/clojure-domain-specific-languages>), Packt Publishing, ISBN 978-1-78216-650-4
- Rochester, Eric (2014), *Mastering Clojure Data Analysis* (<https://www.packtpub.com/big-data-and-business-intelligence/mastering-clojure-data-analysis>), Packt Publishing, ISBN 978-1-78328-413-9
- Emerick, Chas; Carper, Brian; Grand, Christophe (April 19, 2012), *Clojure Programming* (<http://www.clojurebook.com/>), O'Reilly Media, ISBN 1-4493-9470-1
- VanderHart, Luke; Sierra, Stuart (June 7, 2010), *Practical Clojure* (<https://www.apress.com/us/book/9781430272311>), Apress, ISBN 978-1-4302-7231-1

External links

Official website (<https://clojure.org/>) 

Retrieved from "<https://en.wikipedia.org/w/index.php?title=Clojure&oldid=983440049>"

This page was last edited on 14 October 2020, at 07:17 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.