# Cheat sheet

## Basics

### Numbers

| | |
|---|---|
| **Literals** | `42 -1.5 22/7 1N 0xff 2r1010 1e9 ##Inf` |
| **Arithmetic** | `+ - * / quot rem mod inc dec abs max min` |
| **Overflow** | `+' -' *' inc' unchecked-add unchecked-multiply` |
| **Compare** | `= not= == < > <= >= compare` |
| **Test** | `zero? pos? neg? even? odd? number? integer? int? double? ratio? NaN?` |
| **Convert** | `int long double bigint rationalize numerator denominator parse-long parse-double` |
| **Random** | `rand rand-int rand-nth shuffle random-uuid` |
| **Bits** | `bit-and bit-or bit-xor bit-not bit-shift-left bit-shift-right` |
| **Math** | `Math/sqrt Math/pow Math/floor Math/ceil Math/round Math/log Math/sin Math/PI` |

### Strings

| | |
|---|---|
| **Build** | `str format pr-str prn-str with-out-str str/join` |
| **Use** | `count subs str/split str/split-lines str/index-of str/includes? str/starts-with? str/ends-with?` |
| **Search** | `str/last-index-of re-find` |
| **Change** | `str/upper-case str/lower-case str/capitalize str/trim str/triml str/trimr str/replace str/reverse` |
| **Test** | `string? str/blank? char?` |
| **Parse** | `read-string parse-long parse-double parse-boolean keyword symbol name namespace` |

### Regex

| | |
|---|---|
| **Literal** | `#"a+b"` |
| **Match** | `re-find re-matches re-seq re-groups re-matcher re-pattern` |
| **Replace** | `str/replace str/replace-first` |

### Other scalars

| | |
|---|---|
| **Keywords** | `keyword keyword? name namespace` |
| **Symbols** | `symbol symbol? gensym` |
| **nil and bools** | `nil? some? true? false? boolean not` |

## Collections

### Any collection

| | |
|---|---|
| **Ask** | `count empty? not-empty seq coll? contains? every? some not-any?` |
| **Change** | `conj into empty` |
| **Compare** | `= identical? distinct? hash` |
| **Kind** | `coll? sequential? associative? sorted? map? vector? list? set? seq? fn?` |

### Transients

| | |
|---|---|
| **Use** | `transient persistent! conj! assoc! dissoc! pop! disj!` |

### Vectors

| | |
|---|---|
| **Make** | `[] vector vec mapv filterv` |
| **Read** | `get nth peek first last subvec` |
| **Change** | `conj assoc pop update` |

### Lists

| | |
|---|---|
| **Make** | `'() list list*` |
| **Read** | `first rest peek` |
| **Change** | `cons conj pop` |

### Arrays

| | |
|---|---|
| **Use** | `int-array long-array double-array object-array aget aset alength` |

### Maps

| | |
|---|---|
| **Make** | `{} hash-map array-map sorted-map sorted-map-by zipmap frequencies group-by` |
| **Read** | `get get-in keys vals find select-keys contains?` |
| **Change** | `assoc assoc-in dissoc update update-in merge merge-with` |
| **Whole map** | `update-keys update-vals reduce-kv` |
| **Entries** | `key val` |

### Sets

| | |
|---|---|
| **Make** | `#{} hash-set set sorted-set sorted-set-by` |
| **Change** | `conj disj` |
| **clojure.set** | `set/union set/intersection set/difference set/subset? set/superset? set/rename-keys set/index` |

### Nested data

| | |
|---|---|
| **Paths** | `get-in assoc-in update-in` |
| **Walk** | `clojure.walk/postwalk clojure.walk/prewalk clojure.walk/keywordize-keys` |
| **Diff** | `clojure.data/diff` |

## Sequences

### Make a seq

| | |
|---|---|
| **From data** | `seq vals keys rseq subseq rsubseq` |
| **Generate** | `range repeat repeatedly iterate cycle lazy-seq` |
| **Combine** | `concat interleave interpose` |

### Take part of one

| | |
|---|---|
| **From the front** | `first second next rest take drop take-while drop-while nthrest` |
| **From the back** | `last butlast take-last drop-last` |
| **By value** | `filter remove keep distinct dedupe` |
| **By position** | `nth take-nth rand-nth` |

### Transform

| | |
|---|---|
| **Each item** | `map map-indexed keep-indexed mapcat` |
| **Reorder** | `sort sort-by reverse shuffle` |
| **Group** | `partition partition-all partition-by group-by split-at split-with` |
| **Flatten** | `flatten tree-seq` |

### Consume

| | |
|---|---|
| **Reduce** | `reduce reductions reduce-kv transduce frequencies` |
| **Find** | `some first filter every? not-every?` |
| **Side effects** | `doseq dorun doall run!` |

### Transducers

| | |
|---|---|
| **Use** | `transduce into sequence eduction` |
| **Compose** | `comp map filter remove take drop mapcat cat partition-all dedupe` |
| **Stop early** | `reduced reduced? unreduced` |

## Functions

### Define

| | |
|---|---|
| **Named** | `defn defn- defmulti defmethod` |
| **Anonymous** | `fn #(* % 2) letfn` |

### Combine

| | |
|---|---|
| **Build** | `comp partial complement juxt constantly identity fnil memoize every-pred some-fn` |
| **Call** | `apply trampoline` |

### Threading

| | |
|---|---|
| **Thread** | `-> ->> as-> some-> some->> cond-> cond->> doto` |

### Polymorphism

| | |
|---|---|
| **Protocols** | `defprotocol extend-protocol extend-type satisfies? reify` |
| **Types** | `defrecord deftype instance? type` |
| **Multimethods** | `defmulti defmethod derive isa?` |

## Control

### Binding

| | |
|---|---|
| **Local** | `let` `if-let` `when-let` `if-some` `when-some` `letfn` |
| **Global** | `def` `defonce` `declare` |
| **Dynamic** | `binding` `with-redefs` |

### Branching

| | |
|---|---|
| **Two ways** | `if` `if-not` `when` `when-not` |
| **Many ways** | `cond` `condp` `case` |
| **Logic** | `and` `or` `not` |
| **In order** | `do` |

### Looping

| | |
|---|---|
| **Loop** | `loop` `recur` `dotimes` `while` |
| **Comprehend** | `for` `doseq` |

### Errors

| | |
|---|---|
| **Throw** | `throw` `ex-info` |
| **Catch** | `try` `ex-message` `ex-data` `ex-cause` |
| **Check** | `assert` |

### Macros

| | |
|---|---|
| **Define** | `defmacro` `gensym` |
| **Inspect** | `macroexpand` `macroexpand-1` |
| **Code as data** | `quote` `eval` `read-string` `var` |

### Metadata

| | |
|---|---|
| **Use** | `meta` `with-meta` `vary-meta` |
| **Literal** | `^:private` `^{:doc "…"}` |

## State and concurrency

### Atoms

| | |
|---|---|
| **Use** | `atom` `deref` `reset!` `swap!` `swap-vals!` `reset-vals!` `compare-and-set!` |
| **Watch** | `add-watch` `remove-watch` `set-validator!` |

### Refs and agents

| | |
|---|---|
| **Refs** | `ref` `dosync` `alter` `commute` `ref-set` `ensure` |
| **Agents** | `agent` `send` `send-off` `await` `agent-error` `restart-agent` |

### Later

| | |
|---|---|
| **Delay** | `delay` `force` `realized?` |
| **Future** | `future` `future-done?` `future-cancel` |
| **Promise** | `promise` `deliver` |

### Volatiles

| | |
|---|---|
| **Use** | `volatile!` `vreset!` `vswap!` |

## Environment

### Namespaces

| | |
|---|---|
| **Declare** | `ns` `require` `in-ns` |
| **Look around** | `*ns*` `find-ns` `all-ns` `ns-publics` `resolve` |

### Files and processes

| | |
|---|---|
| **Read** | `slurp` `line-seq` `read-line` |
| **Write** | `spit` `println` `print` `prn` `printf` |
| **Run** | `sh` |
| **Args** | `*args*` `*command-line-args*` |

### REPL

| | |
|---|---|
| **Look up** | `doc` `source` `dir` `apropos` `find-doc` |
| **Results** | `*1` `*2` `*3` `*e` `pst` |
| **Debug** | `dbg` `time` `tap>` `add-tap` |
| **Browse** | `pprint` `inspect` |
| **Skip** | `comment` `#_` |

## Batteries

### Files (fs)

| | |
|---|---|
| **Paths** | `fs/cwd` `fs/parent` `fs/file-name` `fs/extension` `fs/temp-dir` |
| **Ask** | `fs/exists?` `fs/directory?` `fs/regular-file?` `fs/list-dir` |
| **Change** | `fs/create-dir` `fs/copy` `fs/move` `fs/delete` |

### Processes (process)

| | |
|---|---|
| **Run** | `process/sh` `process/shell` `process/out` `process/escape` |

### JSON (json)

| | |
|---|---|
| **Use** | `json/parse` `json/write` |

### HTTP (http)

| | |
|---|---|
| **Client** | `http/get` `http/post` `http/request` |
| **Server** | `http/run-server` `http/serve` `http/router` |

### Grammars (peg)

| | |
|---|---|
| **Use** | `peg/match` `peg/compile` `peg/find` `peg/find-all` `peg/replace` `peg/replace-all` |