What is the relationship between Common Lisp, Scheme, and Clojure?

What's the easiest way to create a Clojure project?

What is Clojure's function invocation syntax?

What is odd about Clojure operators?

How do you calculate remainders?

What is the advantage of doing math in prefix notation?

How can you concatenate strings?

Create a string without quotes.

Show the Java class of an expression.

How are conditionals expressed?

What evaluates to "true" for the purpose of `if` expressions?

What conventionally separates lists and vectors?

What separates them in terms of implementation?

How are non-function lists constructed?

How do you get the head/tail of a list?

The last element?

Access an arbitrary index in an list.

Combine two lists.

What is the syntax of vectors, sets, and maps?

How do you assign to a variable?

Are re-assignments allowed?

How can you access the size of collections?

Show that vectors and sets are really functions.

What syntactic convenience does Clojure allow for maps?

What are the two kinds of symbols in Clojure?

Retrieve a value from a map.

Add to a map.

What is Clojure's function definition syntax?

Create and retrieve documentation.

Where can destructuring be used?

How is it different from pattern matching?

What does `let` do?

Why are higher-order functions baked right into Clojure?

How can you create anonymous functions?

How can you apply a function that is the result of an expression?

How does Clojure work around the JVM's lack of tail recursion optimization?

Translate Scala's `forall`.

Translate Scala's `exists`.

Translate Java's `null` check.

How are list comprehensions done?

What is idiomatic Clojure for Java's `isX` functions?

What is idiomatic Clojure's capitalization system for functions.

Scala's `foldLeft` goes by what in other languages?

How do you create ranges?