Clojure Under the Hood

erik

August 10, 2016

Contents

Clojure Under the Hood

- or -

Everything You Wanted to Know about Clojure*
(* But Were Afraid to Ask)

1 Our team had some gaps in our understanding of how Clojure works

This talk intends to cast light on those gaps, in the context of a high-level overview, which will hopefully serve as a practical reference in addressing under-the-hood problems in the future.

Clojure development has a lot of ingredients for new developers to digest:

- the Clojure language
- Leiningen/Maven
- Java/JVM
- Lisp syntax
- Functional Programming
- Emacs/other

So it's not surprising that it's hard to be clear about where one thing ends and another begins.

This will avoid discussing leiningen and maven, to focus on Clojure itself, JVM, and Clojure applications. It *does* assume some basic working knowledge of Clojure.

I am not an expert, so please correct if you hear something wrong or misleading.

2 This topic has been sooo covered by others

Here are just some of the many good resources for these topics:

- Clojure.org
- Clojure Programming (Chas Emerick) (much of this content is from his book, highly recommended!)
- Rich Hickey's "Clojure for Java Programmers Pt 1"

- Decompiling Clojure (Guillermo Winkler) Nice explanation of 3 entry points to compiler: compile, load & eval
- Life of a Clojure Expression (John Hume)
- How Clojure Babies Are Made (Daniel Higgenbotham)

3 Some questions that this will try to make easier to answer

- How do you install Clojure?
- How is it that Clojure is just another .jar file? (that Clojure is packaged as one of many dependencies within a delivered appication?)
- How can Clojure exhibit "interpreted"-like language behavior? For example, lein uberjar, then change .clj files inside, rerun and program's runtime behavior has changed.
- Macros: "built-in" vs. user-defined: how can they have the same status?

4 Clojure and Clojure, what is Clojure?

4.1 Clojure is written in Java and in the Clojure language itself

Which parts are Java?

- **Data structures** (symbol, keyword, vector, etc.)
- **Special operators** (def, fn, var, do, let, etc.)
- **Bytecode compiler** ASM

(The data structures and special operators are in clojure/src/jvm/clojure/lang, and the bytecode compiler is in clojure/src/jvm/clojure/ASM)

The rest is written in Clojure (clojure/src/clj/clojure), that is, written using functions and macros that are composed of those data structures and special operators.

Wait! A bytecode compiler? Yes, Clojure itself includes a bytecode compiler, so Clojure can load and evaluate Clojure expressions at runtime.

4.2 Clojure is a jar

Clojure can be downloaded as a jar: clojure-1.9.0-alpha10.jar. It has .clj & .class files.

The .clj files have already been compiled to .class files, so Clojure itself doesn't need to go through compilation.

5 Clojure at Runtime

5.1 Java basics

Run a simple java application (packaged as a jar file, which is mostly bytecode .class files):

`java -jar HelloWorld.jar`

(executes HelloWorld application contained in jar file)

5.2 Launch a Clojure REPL

```
java -cp clojure.jar clojure.main
```

-cp: add this jar to classpath

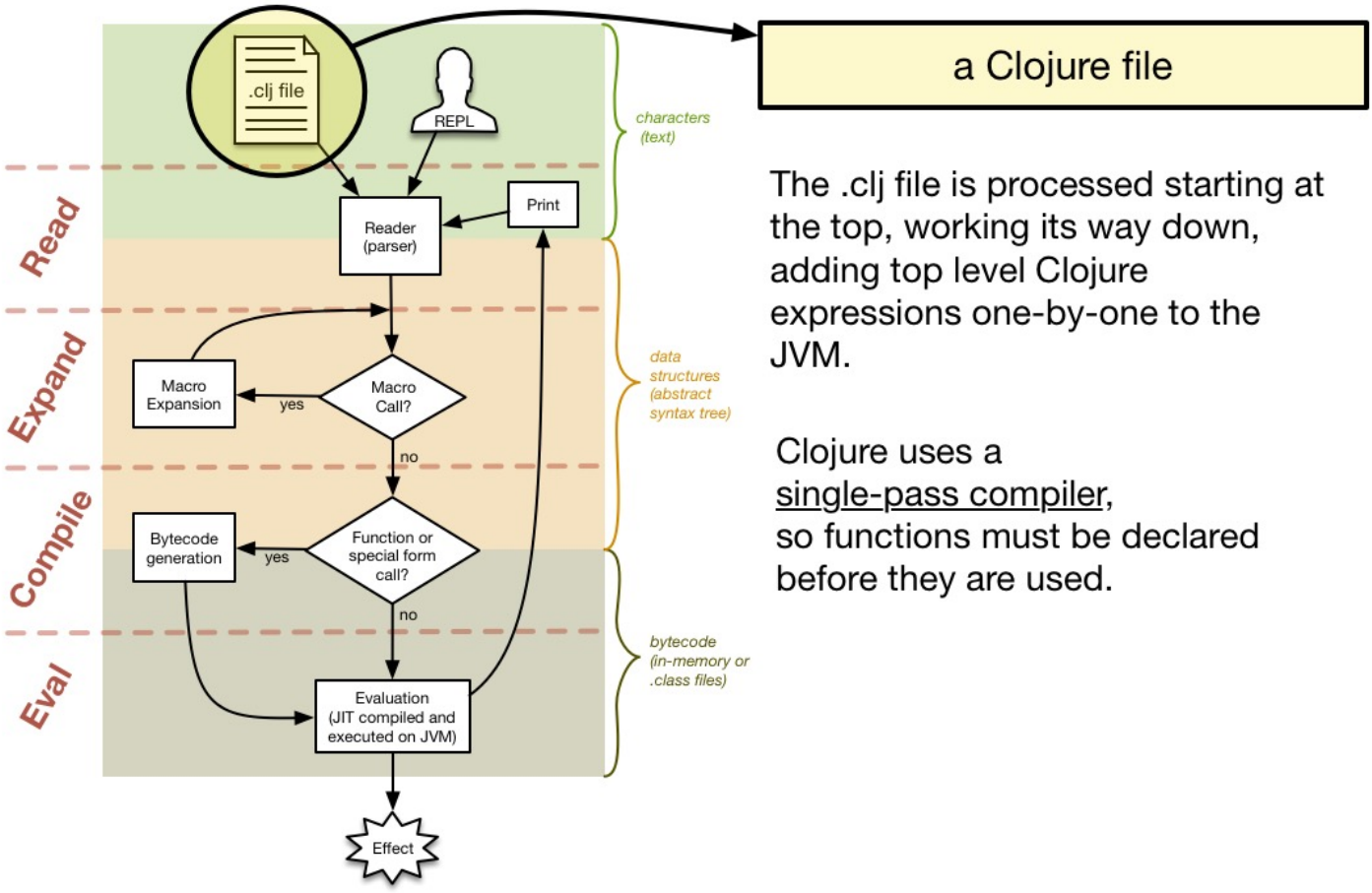
(What is a classpath? The search path that the JVM will use when looking for user-defined classes and resources. It can include both directories and .zip archives, including .jar files.)

clojure.main: the entrypoint for Clojure

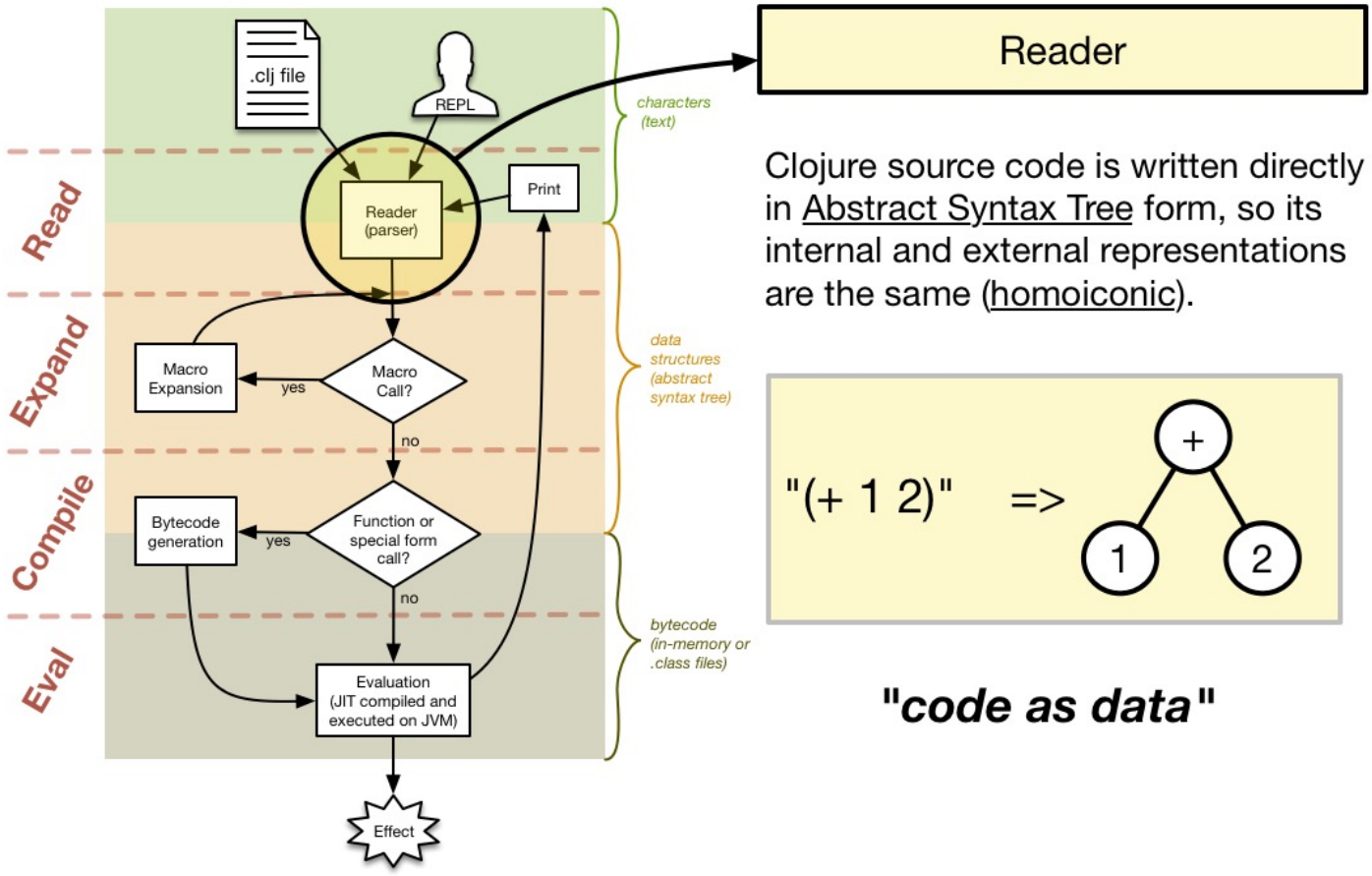
5.3 Run a file full of Clojure code as a script

```
java -cp clojure.jar clojure.main /path/to/myscript.clj
```

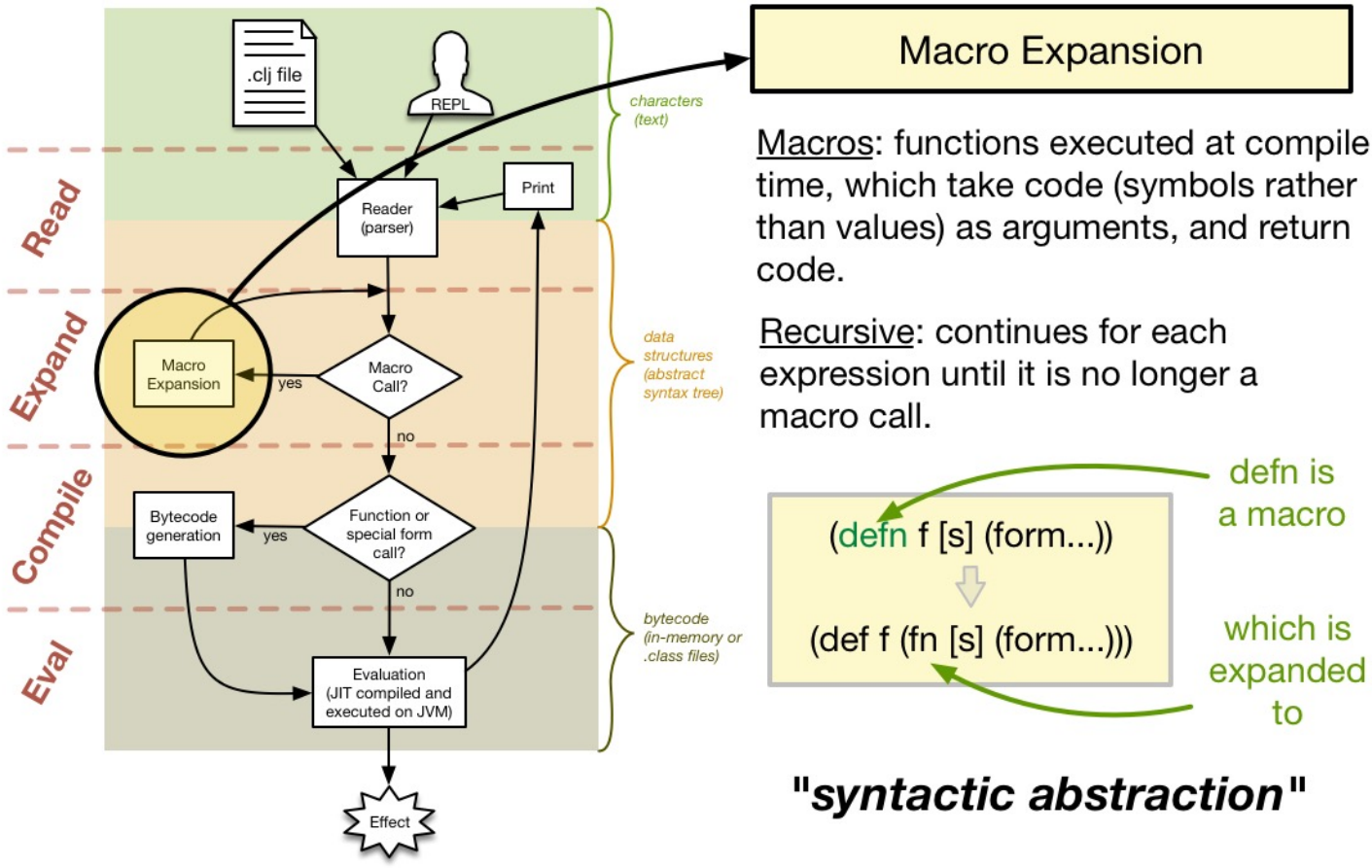
6 Clojure Compilation Model (a Clojure file)



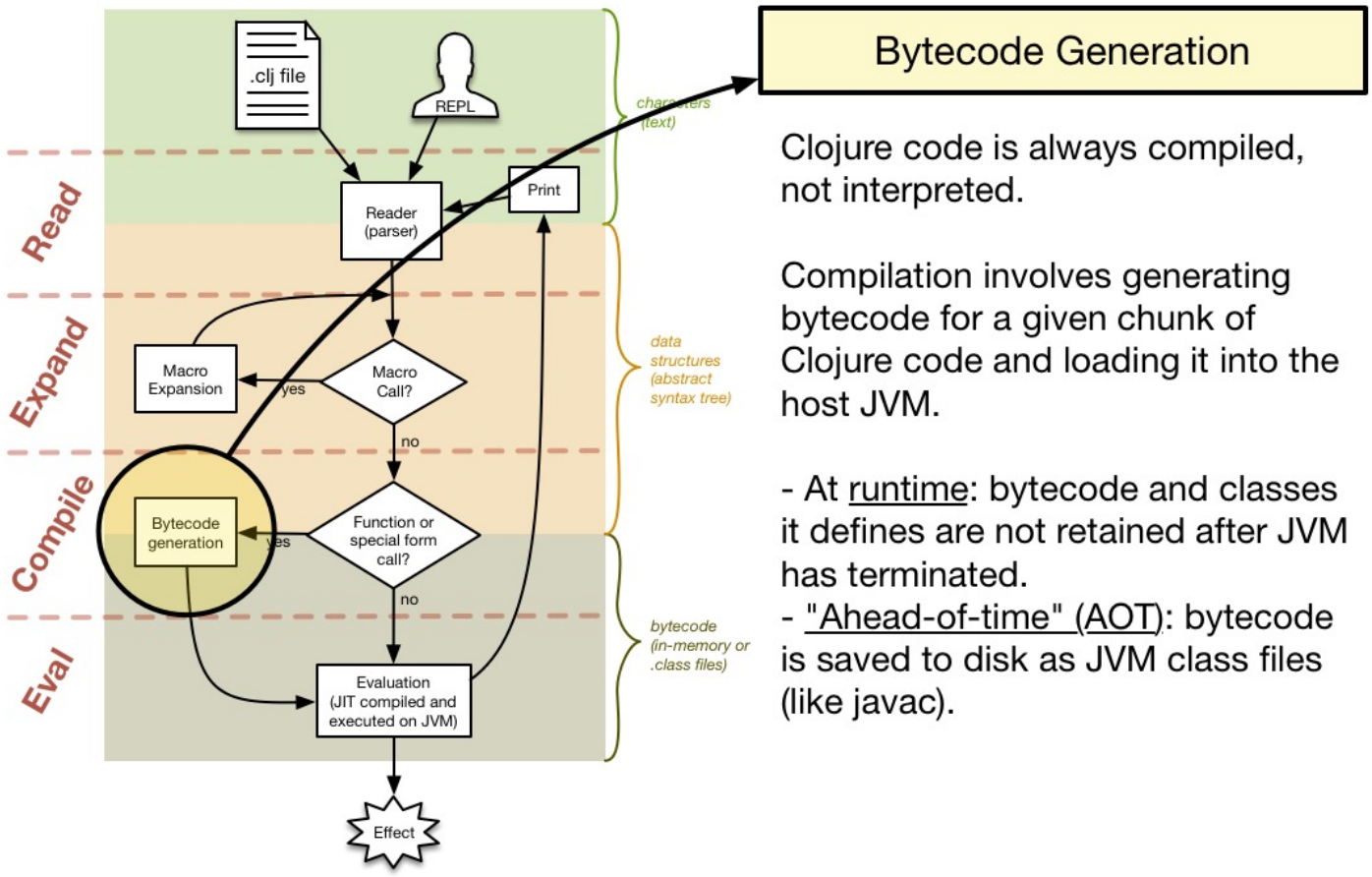
7 Clojure Compilation Model (Reader)



8 Clojure Compilation Model (Macro Expansion)



9 Clojure Compilation Model (Bytecode Generation)



10 Clojure Compilation Model (Evaluation)