Unit Testing and Debugging Clojure Code

Zachary Bennett
Software Engineer

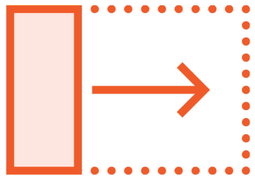
@z_bennett_ zachbennettcodes.com

Unit Testing

The practice of writing code that verifies that the smallest “unit” of a program performs appropriately in isolation.

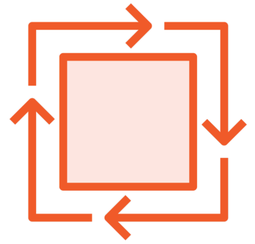
Functions!

Benefits of Unit Testing



Functionality guarantees

Easier integration and refactor



Find bugs early

```
(deftest a-test
  (testing "FIXME, I fail."
    (is (= 0 1))))
```

Example Unit Test

The above code is the auto-generated unit test that Leiningen creates when you start a new project.

Creating a Unit Test

clojure.test

```
(ns my-app.core-test
  (:require [clojure.test :refer :all]
            [my-app.core :refer :all]))


(deftest a-test
  (testing "FIXME, I fail."
    (is (= 0 1))))
```

Example Unit Test

The above code is the auto-generated unit test that Leiningen creates when you start a new project.

```
;; Unit Test Breakdown


(ns my-app.core-test
  (:require [clojure.test :refer :all]
            [my-app.core :refer :all]))


(deftest a-test

  (testing "FIXME, I fail.”

    (is (= 0 1))))
```

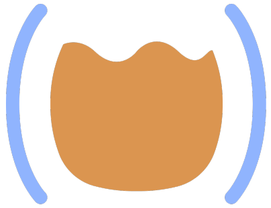
- Set the testing namespace and import the clojure.test and my-app.core namespaces
- Define a new unit test function
- Add a new testing context – this helps you easily track where tests failed
- Provide an assertion to validate using “is”

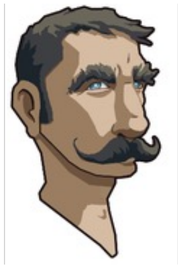
```
;; Composing Test
(deftest division
  (testing "dividing one by one."
    (is (= (/ 1 1) 1))))

(deftest multiplication
  (testing "multiplying one by one."
    (is (= (* 1 1) 1))))

(deftest math-works
  (division)
  (multiplication))
```

- **Define one unit testing function**
- **Define another unit testing function**
- **Compose them!**

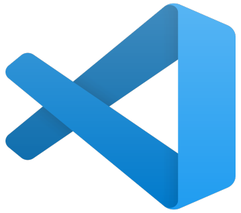
Debugging Clojure

Debugging Environment

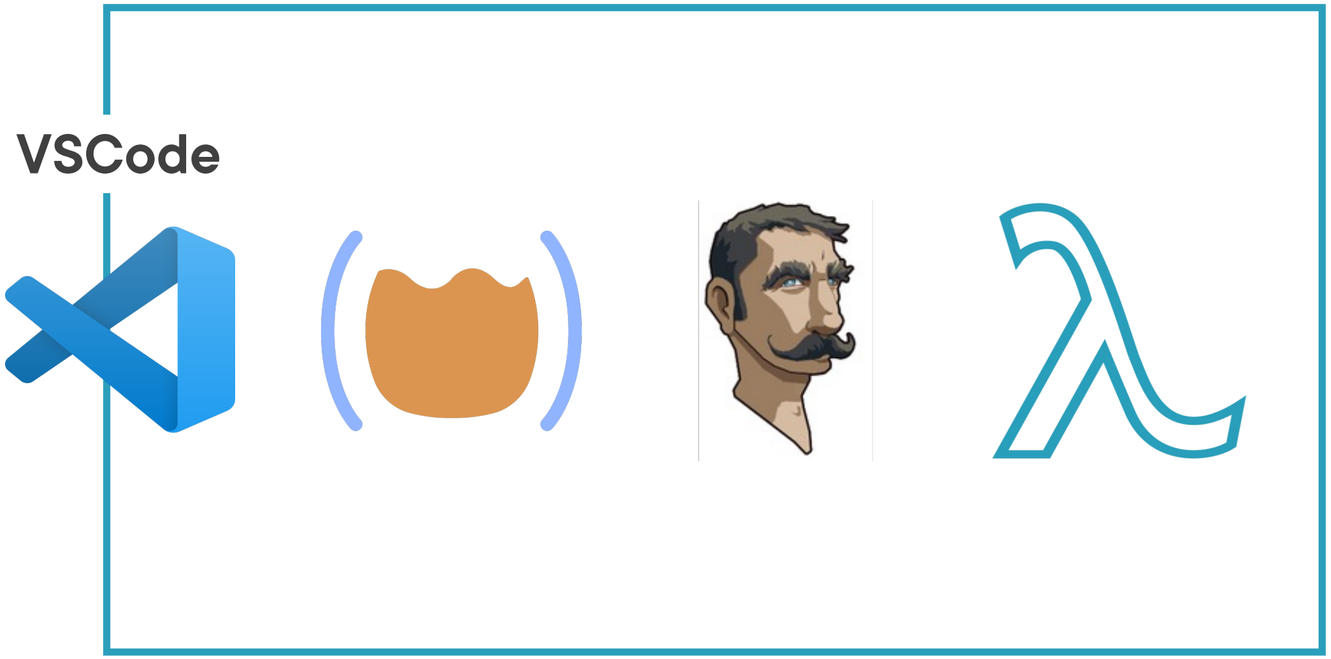




VSCode



Debugging Environment



Jacking-in

The process by which Calva starts a Clojure REPL and then connects it to your Visual Studio Code environment

```
;; Instrumentation

; Define a function

(deftest a-test

  (testing "FIXME, I fail."

    (is (= 0 1))))


; Select then press “ctrl+alt+c i”


; Hit the first breakpoint

(a-test)
```

- First, define a function that you want to instrument

- Use the right key combination to automatically instrument the function with breakpoints

- Call the function and hit the first breakpoint – from there you have the standard debugging tools available in Visual Studio Code

Demo

Running Unit Tests

- Create a unit test
- Run our unit tests using Leiningen
- Debug broken code
- Fix a broken function and validate the fix