Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Мануал по работе с ClosureScript

Москва — 2023 г.

Содержание

Введение

Что такое ClosureScript?

ClosureScript [1] — компилятор для Closure [2], выдающий в результате код на JavaScript.

Что такое Closure?

Closure — диалект языка LISP, являющийся динамическим компилируемым языком программирования, поддерживающий доступ к фреймворкам, написанным на Java. Из-за своего родства с LISP поддерживает функциональное программирование и использование макросов.

Что нужно для того, чтобы начать писать на Closure?

Во-первых, среда разработки или текстовый редактор для Closure — подходящих несколько, например Emacs, Intellij IDEA, VS Code. В рамках данного мануала будет рассмотрено использование текстового редактора VS Code для работы с Closure.

Во-вторых, сам Closure — он доступен для установки под MacOS, Linux и Windows.

В-третьих,

Установка

VS Code и Calva

Установить VS Code под свою платоформу можно по ссылке: `https://code.visualstudio.com/Download`

Для тех, кто не имеет опыта использования IDE или желает научиться использовать VS Code, рекомендуются к прочтению следующие статьи:

1) `https://habr.com/ru/post/490754/` — статья на русском языке

2) `https://code.visualstudio.com/docs/introvideos/basics` — статья на английском языке

Далее необходимо установить Calva — расширение для VS Code, поддерживающее ClojureScript и помогающее разрабатывать ПО на Clojure. Для этого в левом меню VS Code необходимо перейти в раздел "Расширения" (см. скриншот 1), сделать поиск "Calva" и установить найденное расширение (см. скриншот 2).

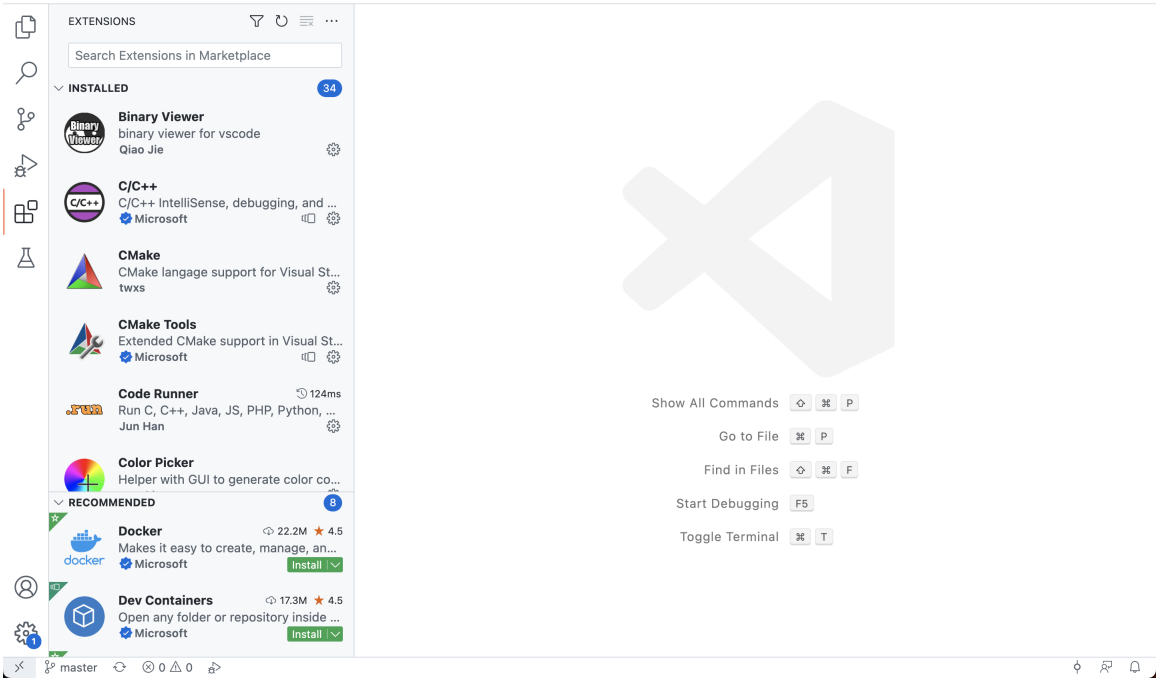

Рисунок 1 – Раздел "Расширения" VS Code

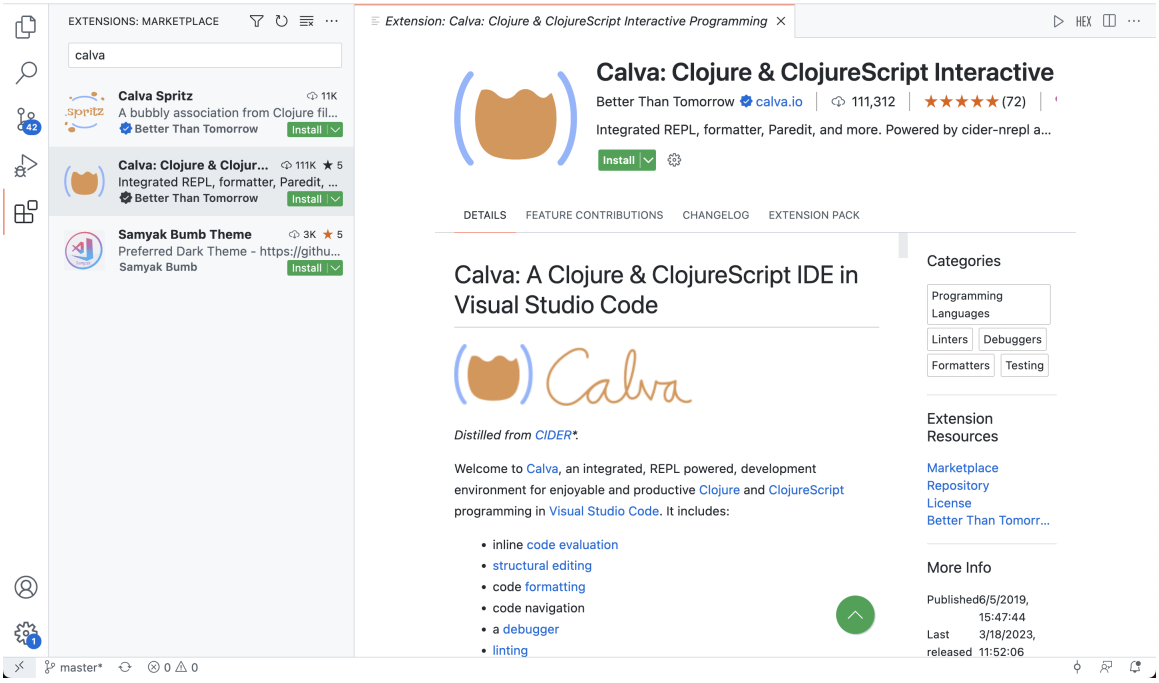

Рисунок 2 – Расширение Calva

Clojure

Для того, чтобы установить Clojure, воспользуемся инструкциями с официального сайта: `https://clojure.org/guides/install_clojure`.

В приведённой статье (на английском языке) присутствую инструкции по установке Clojure на ОС MacOS, Linux-подобные ОС (Ubuntu, Debian), а также Windows. В данном мануале будет рассмотрена установка Clojure на Linux-подобные ОС посредством установщика пакетов brew.

Для установки необходимо открыть терминал и выполнить в нём следующие команды:

Листинг 1 – Установка brew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/
    Homebrew/install/HEAD/install.sh)"
(echo; echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"'
    ) >> /home/<YOURUSERNAME>/.profile
eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
sudo apt-get install build-essentials
```

После установки brew необходимо установить и сам Clojure:

Листинг 2 – Установка Clojure

```
1 brew install clojure/tools/clojure
```

LISP

Поскольку Clojure является диалектом LISP, стоит также установить и базовый интерпретатор LISP, например, Common Lisp. Для ОС семейства Linux это можно сделать следующей командой:

Листинг 3 – Установка Common Lisp

```
1 sudo apt-get install sbcl
```

Начало работы и примеры

Hello World, Clojure

Откроем VS Code и нажмём Ctrl + Shift + P (данное сочетание клавиш откроет паллет управления в VS Code) и введём следующий текст: Calva: Fire up the Getting Started REPL Среди появившихся результатов выберем единственный:

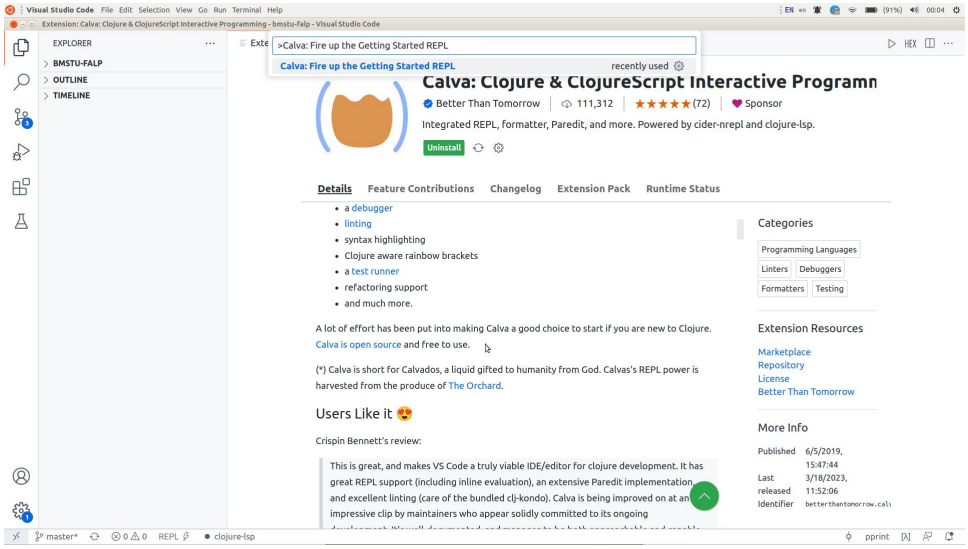

Рисунок 3 – Запуск Calva

После запуска Calva мы увидим примерно следующее:

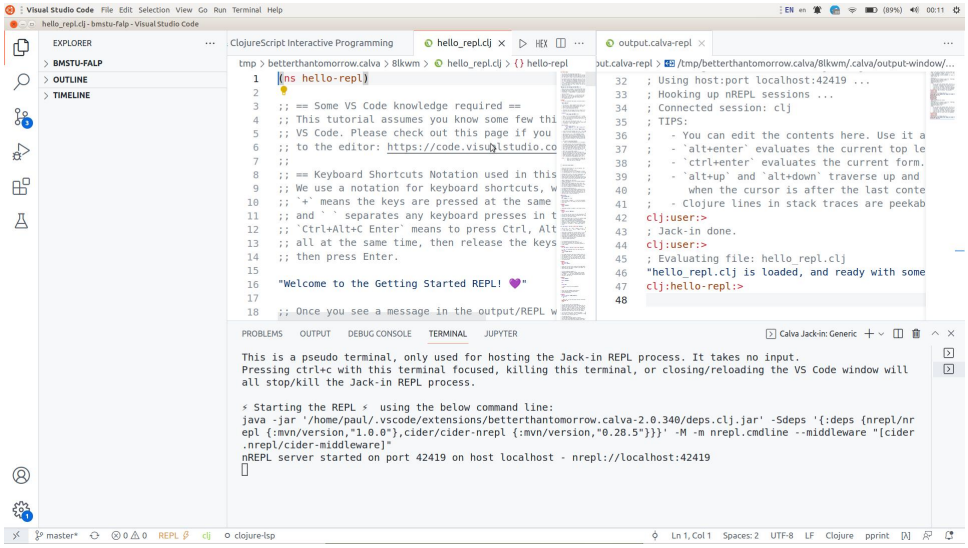

Рисунок 4 – После запуска Calva

Очистим содержимое левого файла и напишем в него ”Hello, World!”:

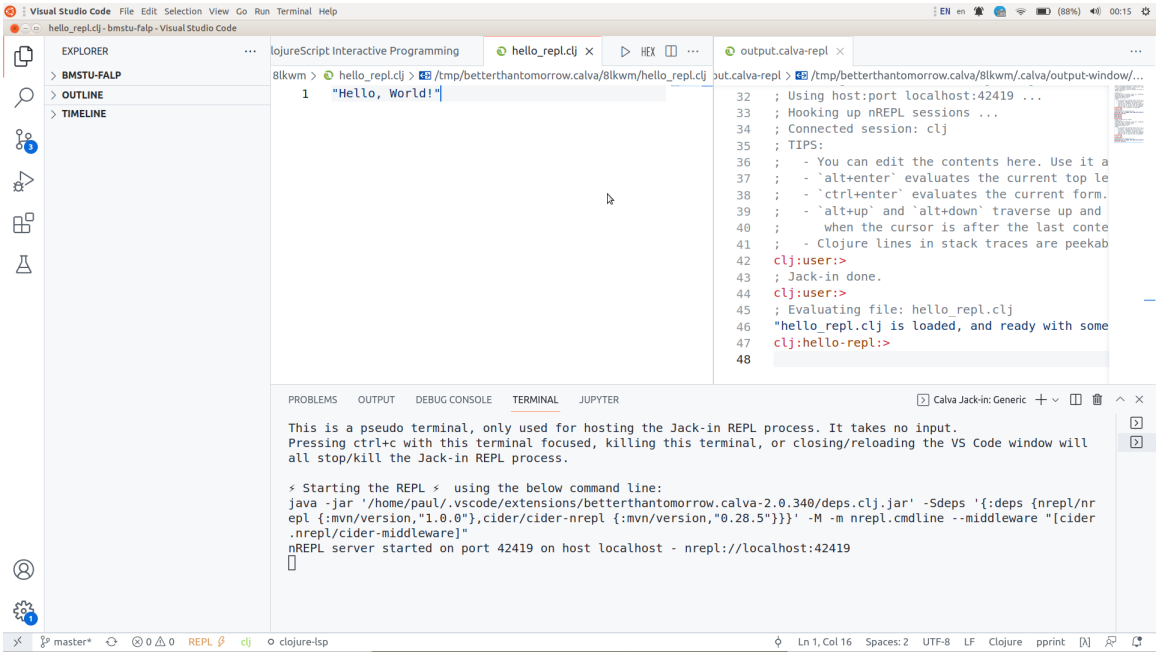

Рисунок 5 – Напишем ”Hello, World!”

Нажмём Alt + Enter и увидим результат:

Рисунок 6 – Hello, World!

Hello World, ClojureScript

Чтобы запустить Hello World с использованием ClojureScript, мы будем выполнять действия по следующему гайду:

`https://clojurescript.org/guides/quick-start`

Для этого создадим папку hello-world и настроим её содержимое следующим образом:

Листинг 4 – ClojureScirpt

```
hello-world          # Our project folder
    src              # The CLJS source code for our project
        hello_world  # Our hello_world namespace folder
            core.cljs # Our main file
    cljs.jar         # (Windows only) The standalone Jar you
      downloaded earlier
    deps.edn         # (macOS/Linux only) A file for listing our
      dependencies
```

Содержимое deps.edn:

Листинг 5 – Содержимое deps.edn

```
{:deps {org.clojure/clojurescript {:mvn/version "1.11.54"}}}
```

Содержимое src/hello_world/core.cljs:

Листинг 6 – Содержимое src/hello_world/core.cljs

```
(ns hello-world.core)

(println "Hello world!")
```

После чего запустим в терминал следующую команду:

Листинг 7 – Запуск в терминале

```
clj -M --main cljs.main --compile hello-world.core --repl
```

В браузере при этом должна открыться следующая страница:

После этого в терминале, из которого произходил запуск, должно появиться сообщение Hello World!

Попробуем добавить несколько функций. Для этого изменим содержимое src/hello_world/core.cljs:

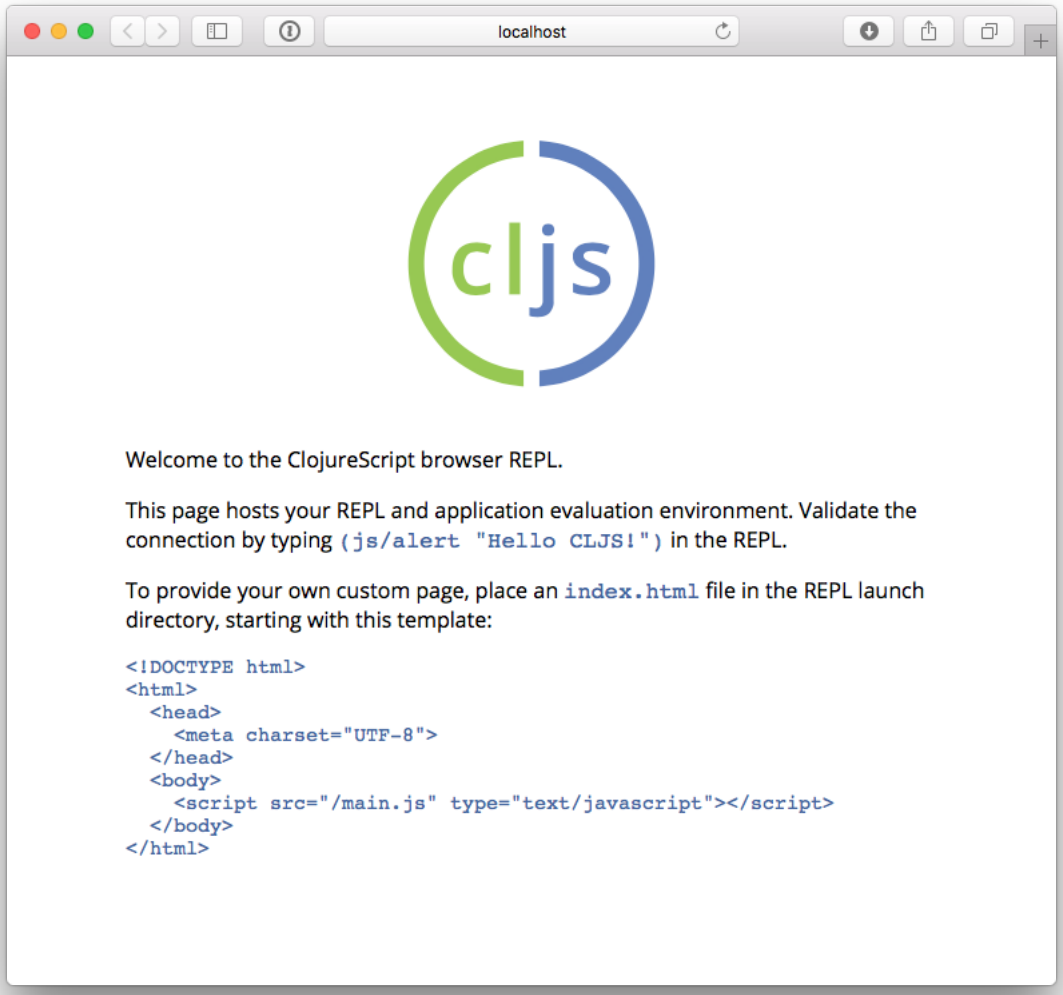

Рисунок 7 – Hello, World!

Листинг 8 – Содержимое src/hello_world/core.cljs

```
1  (ns hello-world.core)
2
3  (println "Hello world!")
4
5  ;; ADDED
6  (defn average [a b]
7      (/ (+ a b) 2.0))
8
9  (defn plus [a b]
10     (+ a b))
```

Перекомпилируем рабочее пространство, выполнив в терминале следующие команды:

Листинг 9 – Обновление страницы проверка новых функций

```
1 (require '[hello-world.core :as hello] :reload)
2 (hello/average 20 13)
3 (hello/plus 1 21)
```

В терминале должны отобразиться результаты работы функций - числа 16.5 и 22.

Отдельно отметим, что в папке out можно найти скомпилированный JavaScript код. Исследовав её содержимое, можно придти к выводу, что полученный код является ни слишком оптимизированным. Для того, чтобы получить более оптимизированную версию кода, необходимо запустить ClojureScript компилятор со следующими опциями:

Листинг 10 – Получени оптимизированной сборки

```
1 clj -M -m cljs.main --optimizations advanced -c hello-world.core
```

Список использованных источников

1. ClosueScript [Электронный ресурс]. Режим доступа: `https://clojurescript.org/` (дата обращения: 22.03.2023).

2. Closure [Электронный ресурс]. Режим доступа: `https://clojure.org/` (дата обращения: 22.03.2023).

3. Баварин И. И. Высшая математика: учебник по естественно-научным направлениям и специальностям. – М.: Гуманит. изд. центр ВЛАДОС, 2003. С. 45–49.

4. Kakaradov B. Ultra-Fast Matrix Multiplication: An Empirical Analysis of Highly Optimized Vector Algorithms, 2004 [Электронный ресурс]. Режим досступа: `https://cs.stanford.edu/people/boyko/pubs/MatrixMult_SURJ_2004.pdf` (дата обращения: 04.10.2022).