Pixie Documentation

Release 0.1

Pixielang.org

June 20, 2017

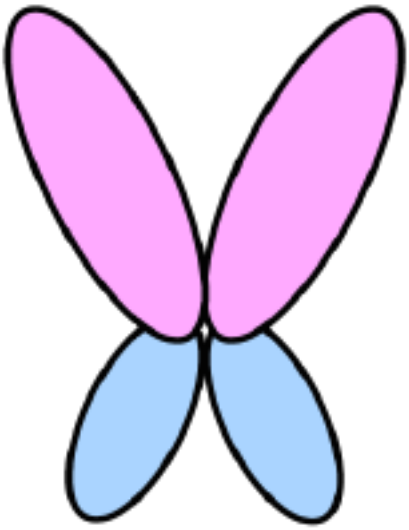
Contents

1 A small, fast, native lisp with "magical" powers **3**

2 Table of Contents **5**

2.1 Introduction 5
2.2 Getting started 5
2.3 Hello World 6
2.4 Pixie Namespaces 7

3 Indices **11**



CHAPTER 1

A small, fast, native lisp with "magical" powers

Pixie is a lightweight lisp suitable for both general use as well as shell scripting. The standard library is heavily inspired by Clojure as well as several other functional programming languages.

Pixie implements its own virtual machine. It does not run on the JVM, CLR or Python VM. It implements its own bytecode, has its own GC and JIT.

Pixie is still in a "pre-alpha" phase and as such changes fairly quickly.

This documentation is also still in development

```
(print "Welcome to Neverland!")
```

Table of Contents

Introduction

Pixie is a lightweight lisp suitable for both general use as well as shell scripting. The standard library is heavily inspired by Clojure as well as several other functional programming languages. It is written in RPython and as such supports a fairly fast GC and an amazingly fast tracing JIT.

Pixie implements its own virtual machine. It does not run on the JVM, CLR or Python VM. It implements its own bytecode, has its own GC and JIT. And it's small. Currently the interpreter, JIT, GC, and stdlib clock in at about 10.3MB once compiled down to an executable

If you like Clojure, but are unhappy with the start-up time, or if you want something outside of the JVM ecosystem, then Pixie may be for you.

Pixie is still in a "pre-alpha" phase and as such changes fairly quickly.

This documentation is also still in development

Getting started

Pixie implements its own virtual machine therefore in order to use Pixie the Pixie-VM is required. Pixie-VM may either be built from source or a release may be downloaded.

Quickstart

The latest Pixie-VM (PVM) is available binary is available here. The PVM is currently in alpha.

Building from source

Building from source is currently possible on 4 platforms:

1. Arch
2. Debian
3. FreeBSD
4. OS X

Builing on Arch

```
pacman -S libuv libffi python2
mkdir ~/projects/pixie-lang
cd ~/projects/pixie-lang
git clone https://github.com/pixie-lang/pixie.git
cd pixie
PYTHON="/usr/bin/python2" make build_with_jit
mkdir ~/bin
cd ~/bin
```

Building on Debian

```
sudo apt-get install build-essential pkg-config
sudo apt-get install curl libffi-dev libedit-dev
sudo apt-get install libboost-all-dev
sudo apt-get install libuv-dev
git clone https://github.com/pixie-lang/pixie.git
cd pixie
make build_with_jit
```

Building on FreeBSD 10.1

```
pkg install libuv libffi gmake readline python
git clone https://github.com/pixie-lang/pixie.git
cd pixie
make build_with_jit
```

Building on OS X via Homebrew

```
brew install libuv libffi boost
git clone https://github.com/pixie-lang/pixie.git
cd pixie
make build_with_jit
```

Installation

Once the PVM has been obtained it is advised that a symbolic link from the PVM to the */usr/bin/pixie* directory be created.

This documentation assumes that an alias (pixie) to the PVM has been created

Verify the installion of Pixie by entering the following in the terminal:

```
$ pixie -v
```

If installed correctly the installed version of pixie will echoed.

Hello World

The most basic Pixie program

```
;Pixie comments are preceded by a semicolon.
;This whole line is a comment

(println "Hello Neverland")
```

The following program highlights some functions in Pixie

```
;This code snippet highlights a basic program in Pixie

(ns hello-neverland) ;declare the namespace

(print "Goodbye world!\n")
;print does not terminate the output with a new line

(loop [x 5]
;a basic loop
  (when (> x 0)
  ;this loop runs while x > 0
    (print (str x "\n"))
    ;string can be built using (str string1 string2 ... stringN)
    (recur (- x 1))))
    ;x is decremented on each interation

(println "Hello Neverland!")
;println terminate the output with a new line
```

Pixie Namespaces

pixie.async

No documentation

pixie.buffers

The `pixie.buffers` namespace provides functionality relating to the use of *pixie.stdlib.Buffer*

pixie.buffers.acopy

Copies `len` elements from the source buffer (`src`) starting at the `nth` element to the destination buffer (`dest`) starting at the `mth` element

```
(acopy src nth dest mth len)
```

Example usage

```
(ns buffer-test)

;TODO
```

pixie.channels

No documentation

pixie.csp

The `pixie.csp` namespace provides functionality for communicating sequential processes.

pixie.csp.close!

Closes the channel `c`, future writes will be rejected, future reads will drain the channel before returning nil.

```
(close! c)
```

Example usage

```
(ns csp-test)

;TODO
```

pixie.csp.put!

Puts the value `v` into the channel `c`, calling the optional callback `f` when the operation has completed.

```
(put! c v)
(put! c v f)
```

Example usage

```
(ns csp-test)

;TODO
```

pixie.csp.take!

"Takes a value from a channel `c`, calling the provided callback `f` when completed"

```
(take! c)
(take! c f)
```

Example usage

```
(ns csp-test)

;TODO
```

pixie.csp.alts!

No documentation

pixie.csp.>!

No documentation

pixie.csp.<!

No documentation

pixie.data

No documentation

pixie.ffi-infer

No documentation

pixie.fs

The `pixie.fs` namespace provides functionality for interfacing with the file system.

pixie.fs.dir

Returns a dir `d` if the path is a dir or does not exist. If a different filesystem object exists at the path an error will be thrown.

```
(dir d)
```

Example usage

```
(ns fs-test)

(dir "/etc")
;=> <Dir /etc>
```

pixie.fs.file

Returns a file if the path `p` is a file or does not exist. If a different filesystem object exists at the path an error will be thrown.

```
(file p)
```

Example usage

```
(ns fs-test)

(file "~/.bashrc")
;=> <File /home/pixie/src/~/.bashrc>
```

pixie.fs.fspath

Returns either a File or Dir `fd` if they exist at the path

```
(fspath fd)
```

Example usage

```
(ns fs-test)

(fspath "/etc")
;=> <Dir /etc>
(fspath "~/.bashrc")
;=> <File /home/pixie/src/~/.bashrc>
```

pixie.fs.rel-path

Get the common root of the two paths, `path1` and `path2`, and the bits that diverge

```
(rel-path path1 path2)
```

Example usage

```
(ns fs-test)

;TODO
```

CHAPTER 3

Indices

- genindex
- modindex
- search