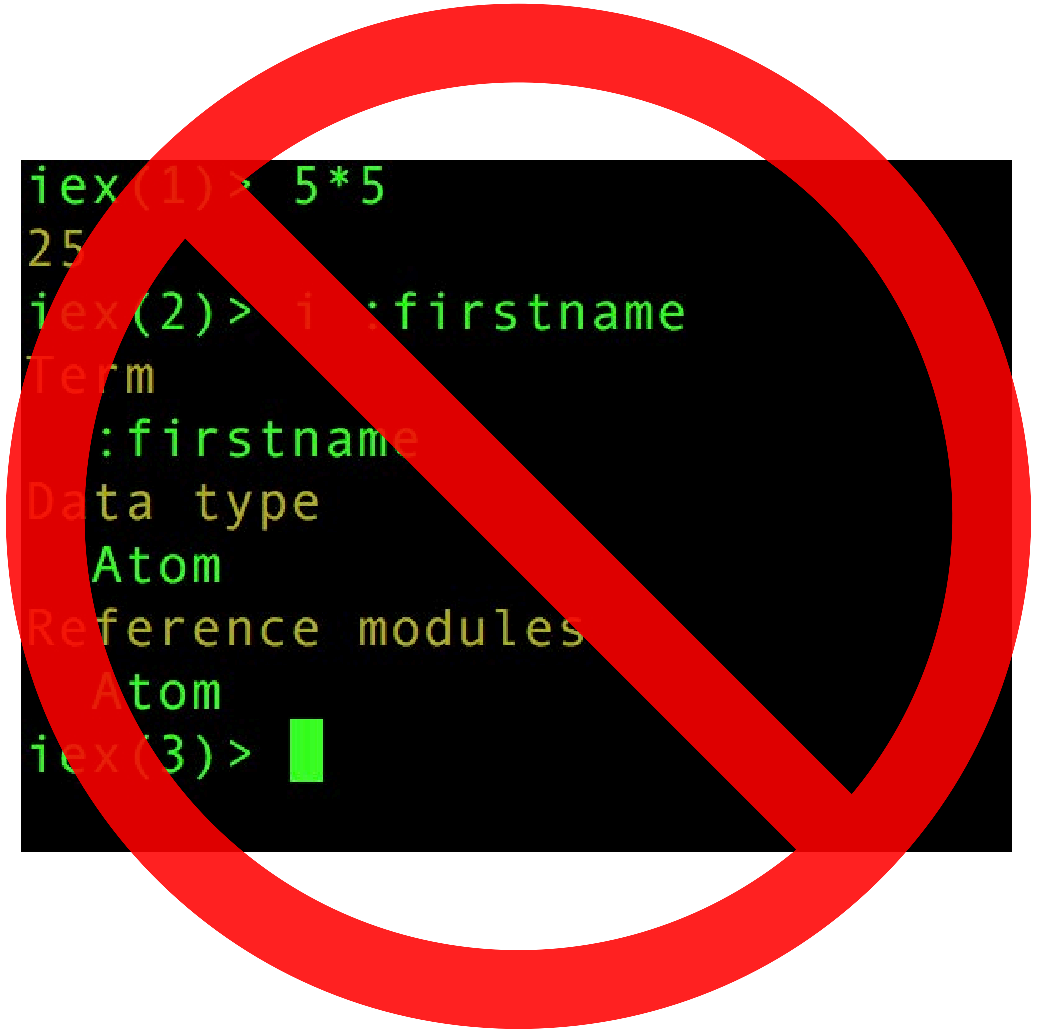
Modules and Functions

Nathan Taylor
SOFTWARE ENGINEER

@taylonr taylonr.com

User Expectations



```
iex(1)> 5*5
25
iex(2)> i :firstname
Term
  :firstname
Data type
  Atom
Reference modules
  Atom
iex(3)> 
```

An Overview

Basic Building Block: Function

- hd
- elem
- List.insert_at

Maintainability

- Logical Grouping

Create Custom Modules

Write Named Functions

Defining a Module

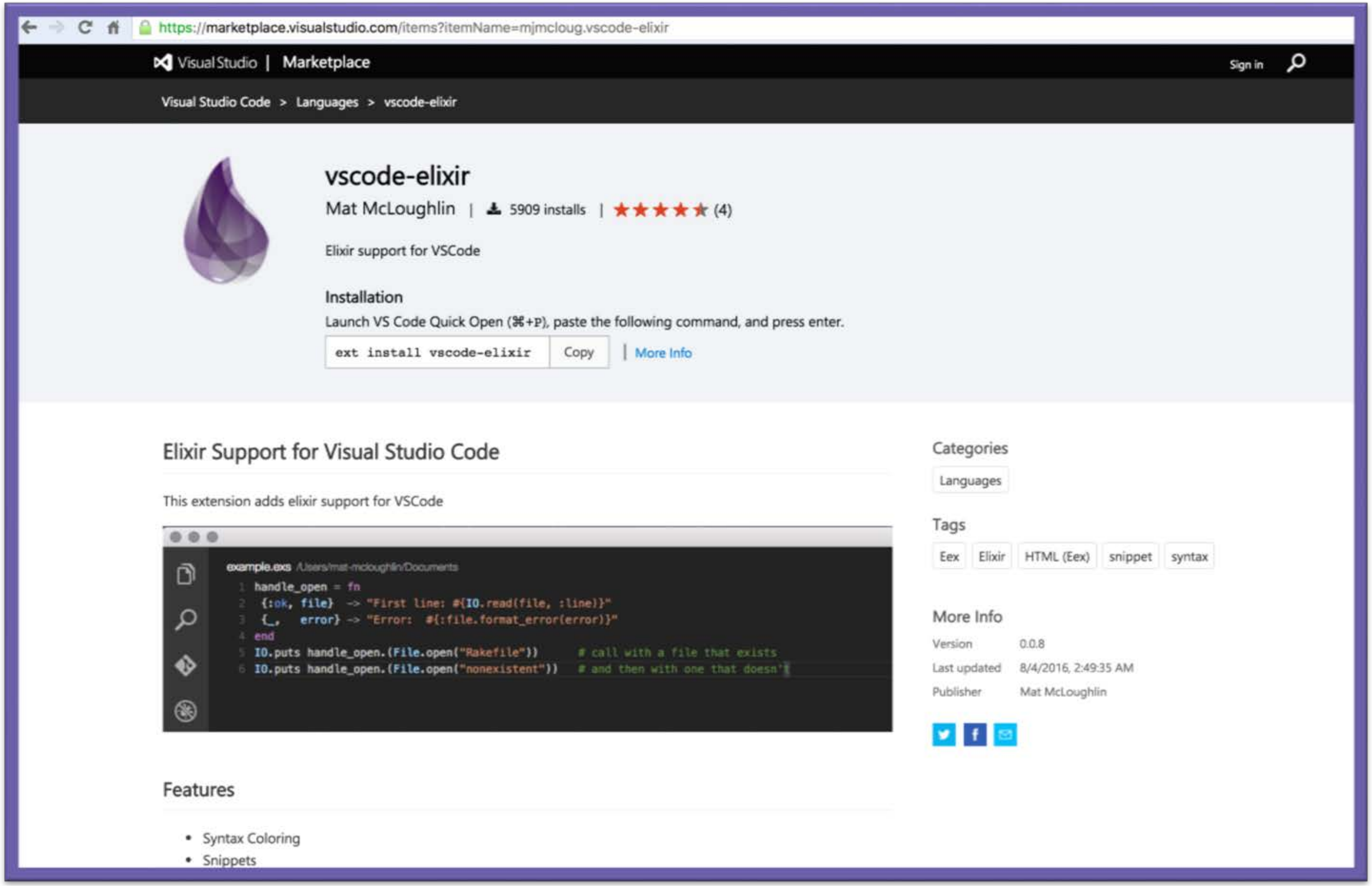
Editors

Sublime

Atom

Emacs

Visual Studio Code



https://marketplace.visualstudio.com/items?itemName=mjmcloug.vscode-elixir

Visual Studio | Marketplace   Sign in

Visual Studio Code > Languages > vscode-elixir

vscode-elixir

Mat McLoughlin | 5909 installs | ★★★★★ (4)

Elixir support for VSCode

Installation

Launch VS Code Quick Open (⌘+P), paste the following command, and press enter.

```
ext install vscode-elixir
```

Copy | More Info

Elixir Support for Visual Studio Code

This extension adds elixir support for VSCode

example.exs /Users/mat-mcloughlin/Documents

```
handle_open = fn
 {:ok, file}  -> "First line: #{IO.read(file, :line)}"
 {_,   error} -> "Error:  #{:file.format_error(error)}"
end
IO.puts handle_open.(File.open("Rakefile"))      # call with a file that exists
IO.puts handle_open.(File.open("nonexistent"))   # and then with one that doesn't
```

Categories

Languages

Tags

Eex | Elixir | HTML (Eex) | snippet | syntax

More Info

Version	0.0.8
Last updated	8/4/2016, 2:49:35 AM
Publisher	Mat McLoughlin

Features

- Syntax Coloring
- Snippets

Module Directives

Directives Comparison

Import	• Include module functions • Include/Exclude specific functions
Alias	• Reduce typing • Rename a module in your module
Require	• Allows using macros in your module

Functions

Function Arity

```
def first(list) do
   hd(list)
end


first/1


{function name} / {number of parameters}
```

Matching

```
def some_func(quantity, {_, _, price}) do
  quantity * price
end

def some_func(quantity, book) do
  quantity * elem(book, 2)
end
```

Guard Clauses

Operators

comparison (==, !=, ===, !==, >, >=, <, <=)

boolean (and, or, not)

arithmetic (+, -, *, /)

arithmetic (+, -)

the binary concatenation operator <>

the **in** operator as long as the right side is a range or a list

Type Check Functions

- is_atom/1
- is_binary/1
- is_bitstring/1
- is_boolean/1
- is_float/1
- is_function/1
- is_function/2
- is_integer/1
- is_list/1
- is_map/1
- is_nil/1
- is_number/1
- is_pid/1
- is_port/1
- is_reference/1
- is_tuple/1

Additional Functions

abs(number)

binary_part(binary, start, length)

bit_size(bitstring)

byte_size(bitstring)

div(integer, integer)

elem(tuple, n)

hd(list)

length(list)

map_size(map)

node()

node(pid | ref | port)

rem(integer, integer)

round(number)

self()

tl(list)

trunc(number)

tuple_size(tuple)

Default Parameters

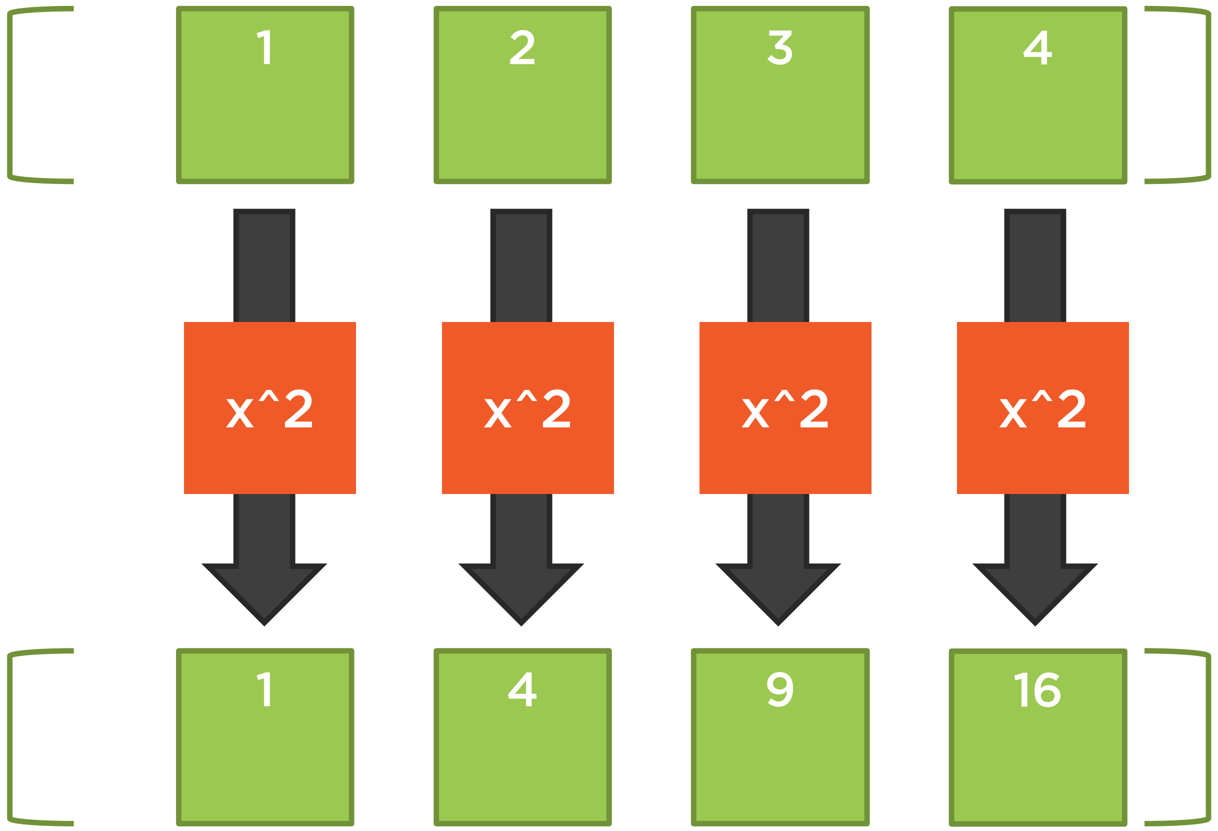
Private Functions

Functions as First Class Citizens

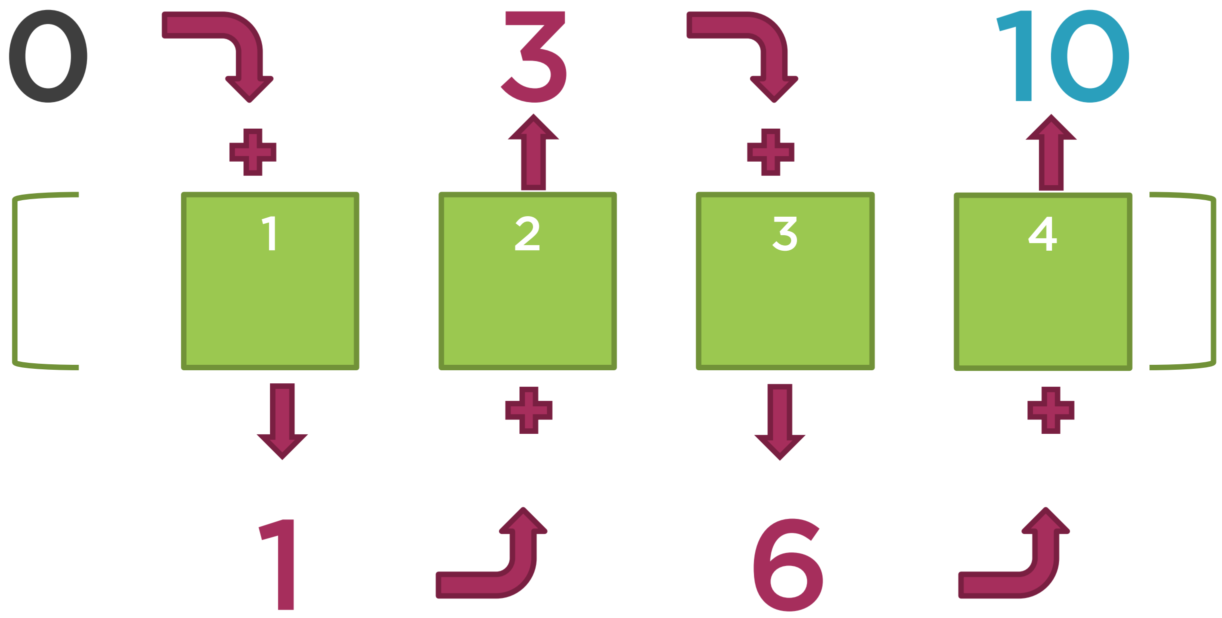
First Class Citizen

supports passing functions as arguments to other functions, returning them as the values from other functions, and assigning them to variables

Map



Reduce



Anonymous Functions

Summary

Module

- Import
- Alias

Pattern Matching

Guard Clauses

Default Parameters

Functions as Parameters

Coming Up:

Control Flow