Elixir Cheatsheet

Data Types

Integers: 55 1_420_000
Floats: 31.35 1_101_123.53

Atoms: :atom, nil, true, false

Binaries (strings): “Hello”

Maps: str = %{first_name: “John”, last_name: “Doe”}

Tuples: {“John”, 52}
{:ok, “Hello World”}

List: [1 ,2, 3]

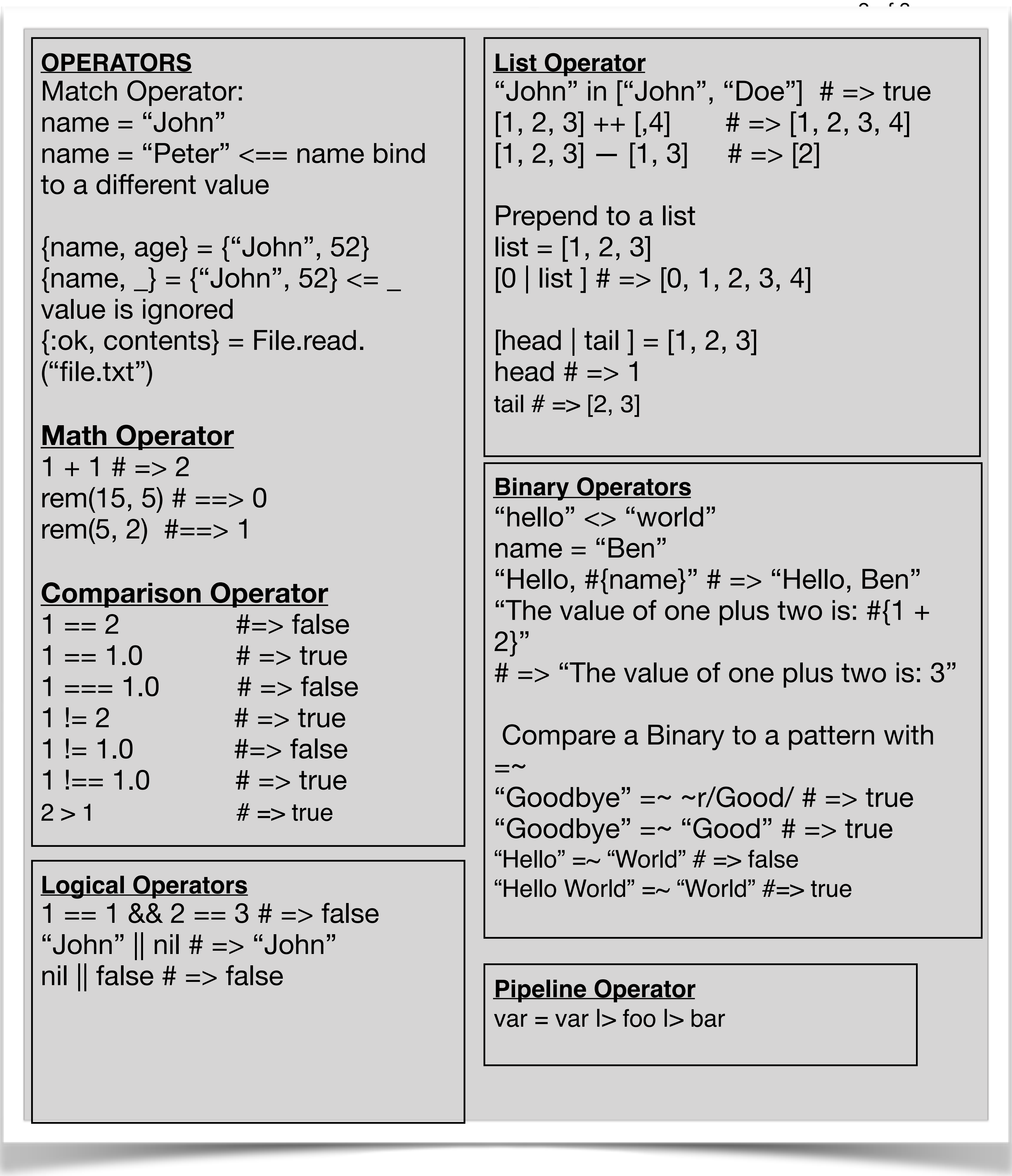
Functions: add = fn(alb) -> a + b end

Character List: ‘hello’

Structs:

```
%Members{
    id: “1234”,
    first_name: “John”,
    last_name: “Doe”
}
```

Ranges: 1..100



OPERATORS

Match Operator:

```
name = “John”
name = “Peter” <== name bind
to a different value

{name, age} = {“John”, 52}
{name, _} = {“John”, 52} <= _
value is ignored
{:ok, contents} = File.read.
(“file.txt”)
```

Math Operator

```
1 + 1 # => 2
rem(15, 5) # ==> 0
rem(5, 2)  #==> 1
```

Comparison Operator

```
1 == 2            #=> false
1 == 1.0          # => true
1 === 1.0         # => false
1 != 2            # => true
1 != 1.0          #=> false
1 !== 1.0         # => true
2 > 1             # => true
```

Logical Operators

```
1 == 1 && 2 == 3 # => false
“John” || nil # => “John”
nil || false # => false
```

List Operator

```
“John” in [“John”, “Doe”]  # => true
[1, 2, 3] ++ [,4]       # => [1, 2, 3, 4]
[1, 2, 3] — [1, 3]      # => [2]
```

Prepend to a list

```
list = [1, 2, 3]
[0 | list ] # => [0, 1, 2, 3, 4]

[head | tail ] = [1, 2, 3]
head # => 1
tail # => [2, 3]
```

Binary Operators

```
“hello” <> “world”
name = “Ben”
“Hello, #{name}” # => “Hello, Ben”
“The value of one plus two is: #{1 +
2}”
# => “The value of one plus two is: 3”
```

Compare a Binary to a pattern with =~

```
“Goodbye” =~ ~r/Good/ # => true
“Goodbye” =~ “Good” # => true
“Hello” =~ “World” # => false
“Hello World” =~ “World” #=> true
```

Pipeline Operator

```
var = var l> foo l> bar
```