# take the first n objects from a list

take = (n, [x, ...xs]:list) -->

| n <= 0 => []

| empty list => []

| otherwise => [x] ++ take n - 1, xs

take 2, [1, 2, 3, 4, 5]

# Curried functions

take-three = take 3

take-three [6, 7, 8, 9, 10]

# Function composition

last-three = reverse >> take-three >> reverse

last-three [1 to 8]

# List comprehensions and piping

const t1 =

\* id: 1

name: 'george'

\* id: 2

name: 'mike'

\* id: 3

name: 'donald'

const t2 =

\* id: 2

age: 21

\* id: 1

age: 20

\* id: 3

age: 26

[{id:id1, name, age}

for {id:id1, name} in t1

for {id:id2, age} in t2

where id1 is id2]

|> sort-by \id

|> JSON.stringify

~function add x, y

@result = x + y

class A

(num) ->

@x = num

property: 1

method: (y) ->

@x + @property + y

a = new A 3

a.x #=> 3

a.property #=> 1

a.method 6 #=> 10

f = !-> 2

g = (x) !-> x + 2

result = switch 'test'

case 'blatant'

'effort'

fallthrough

case 'at'

'increasing'

fallthrough

case 'relevance'

void