Functional Programming

Brian Lonsdorf
@drboolean

http://drboolean.gitbooks.io/mostly-adequate-guide/
https://bit.ly/2QiJ0XG

Functional Programming

Programming with functions

Set theoretically

Every function is

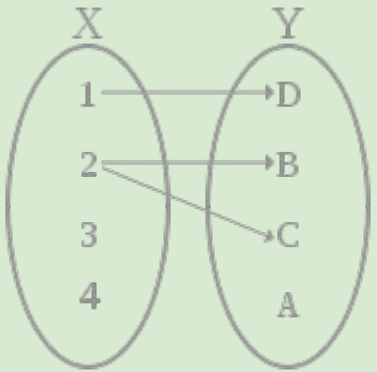
a *single-valued* collection of pairs

THIS

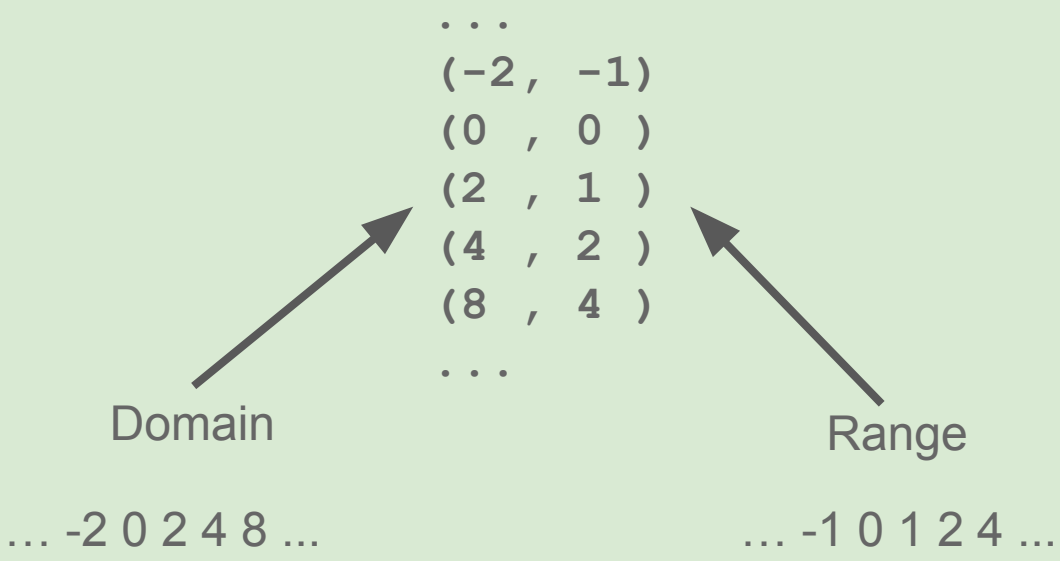
```
...
(-2, -1)
(0 , 0 )
(2 , 1 )
(4 , 2 )
(8 , 4 )
...
```

NOT THIS

```
(1, D)
(2, B)
(2, C)
```

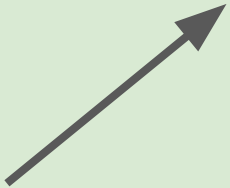


One input, one output



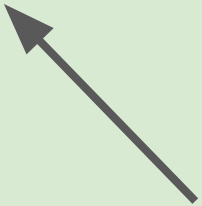
One input, one output

Input	Output
1	2
2	4
3	6



Domain

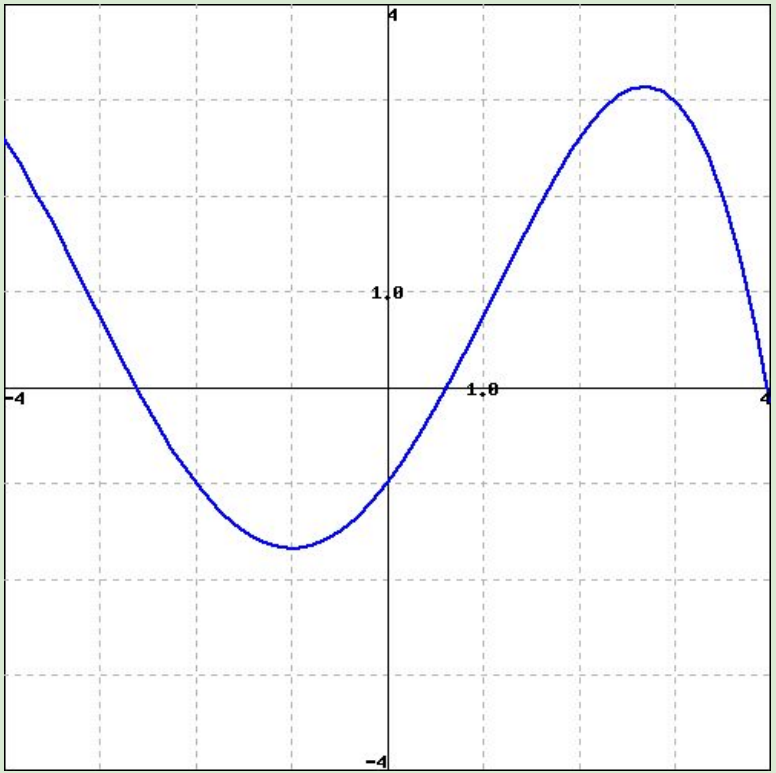
… 1,2,3 ...



Range

… 2,4,6...

One input, one output



One input, one output

```
const toLowerCase = {"A":"a", "B": "b", "C": "c", "D": "d", "E": "e", "D": "d"}

toLowerCase["C"]
//=> "c"
```

```
const isPrime = {1: false, 2: true, 3: true, 4: false, 5: true, 6: false}

isPrime[3]
//=> true
```

Functions

1) *Total*
2) *Deterministic*
3) *No Observable Side-Effects*

Total

For every input there is a corresponding output

Total

```
const inc = i => {
    if(i === 0) return 1
    if(i === 1) return 2
    if(i === 2) return 3
}
```

```
const inc = i => {
    return i + 1
}
```

Total

```
const inc = i => {
    if(i === 0) return 1
    if(i === 1) return 2
    if(i === 2) return 3
}
```

```
const inc = i => {
    if(i === 0) return 1
    if(i === 1) return 2
    if(i === 2) return 3
    return 100
}
```

Deterministic

Always receive the same output for a given input

Deterministic

```
const timeSince = comment => {
    const now = new Date()
    const then = new Date(comment.createdAt)
    return getDifference(now, then)
}
```

```
const getDifference = (now, then) => {
    const days = Math.abs(now.getDate() - then.getDate());
    const hours = Math.abs(now.getHours() - then.getHours());
    return {days, hours}
}
```

No Side Effects

No observable effects besides computing a value

No Side Effects

```
const add = (x, y) => {
    console.log(`Adding ${x} ${y}`)
    return x + y
}
```

```
const add = (x, y) => {
    return {result: x + y, log: `Adding ${x} ${y}`}
}
```

```
var xs = [1,2,3,4,5]
```

```
// not a function
xs.splice(0,3)
//=> [1,2,3]

xs.splice(0,3)
//=> [4,5]

xs.splice(0,3)
//=> []
```

```
// function
xs.slice(0,3)
//=> [1,2,3]

xs.slice(0,3)
//=> [1,2,3]

xs.slice(0,3)
//=> [1,2,3]
```

```
// not a function
const toSlug = (title) => {
    const urlFriendly = title.replace(/\W+/ig, '-')
    if(urlFriendly.length < 1) {
        throw new Error('is bad')
    }
    return urlFriendly
}
```

```
// function
const toSlug = (title) => {
    return new Promise((res, rej) => {
        const urlFriendly = title.replace(/\W+/ig, '-')

        if(urlFriendly.length < 1) {
            rej(new Error('is bad'))
        }
        return res(urlFriendly)
    })
}
```

```
// not a function
const signUp = (attrs) => {
    let user = saveUser(attrs)
    welcomeUser(user)
}
```

```
// function
const signUp = (attrs) => {
    return () => {
        let user = saveUser(attrs)
        welcomeUser(user)
    }
}
```

Let's play function or no function

Let’s play function or no function

```
const birthday = user => {
    user.age += 1;
    return user;
}
```

Let’s play function or no function

```
const shout = word =>
    word.toUpperCase().concat("!")
```

Let’s play function or no function

```
const headerText = header_selector =>
    querySelector(header_selector).text()
```

Let’s play function or no function

```
const parseQuery = () =>
    location.search.substring(1).split('&').map(x => x.split('='))
```

Let's play function or no function

```
var parseQueryString = function(queryString) {
    var params = {}, queries, temp, i, l;

    queries = queryString.split("&");

    for ( i = 0, l = queries.length; i < l; i++ ) {
        temp = queries[i].split('=');
        params[temp[0]] = temp[1];
    }

    return params;
};
```

Why?

- *Reliable*
- *Portable*
- *Reusable*
- *Testable*
- *Composable*
- *Properties/Contract*

```
// associative
add(add(x, y), z) == add(x, add(y, z))

// commutative
add(x, y) == add(y, x)

// identity
add(x, 0) == x

// distributive
add(multiply(x, y), multiply(x, z)) == multiply(x, add(y,z))
```

```
const url = t => `http://gdata.youtube.com/feeds/api/videos?q=${t}&alt=json`

const src = _.compose(_.prop('url'), _.head, _.prop('media$thumbnail'), _.prop('media$group'))

const srcs = _.compose(_.map(src), _.prop('entry'), _.prop('feed'))

const images = _.compose(_.map(imageTag), srcs)

const widget = _.compose(_.map(images), getJSON, url)



widget('cats').fork(log, setHtml(document.querySelector('#youtube')))
```

```
const doStuff = _.compose(
  join(''),
  _.filter(x => x.length > 3),
  reverse,
  _.map(trim),
  split(' '),
  toLowerCase
)
```

```
const doStuff = _.compose(
  join(''),
  _.filter(x => x.length > 3),
  reverse,
  _.map(trim),
  split(' '),
  toLowerCase
)
```

'Chain Dot Com '

```
const doStuff = _.compose(
  join(''),
  _.filter(x => x.length > 3),
  reverse,
  _.map(trim),
  split(' '),
  toLowerCase
)
```

'chain dot com '

```
const doStuff = _.compose(
  join(''),
  _.filter(x => x.length > 3),
  reverse,
  _.map(trim),
  split(' '),
  toLowerCase
)
```

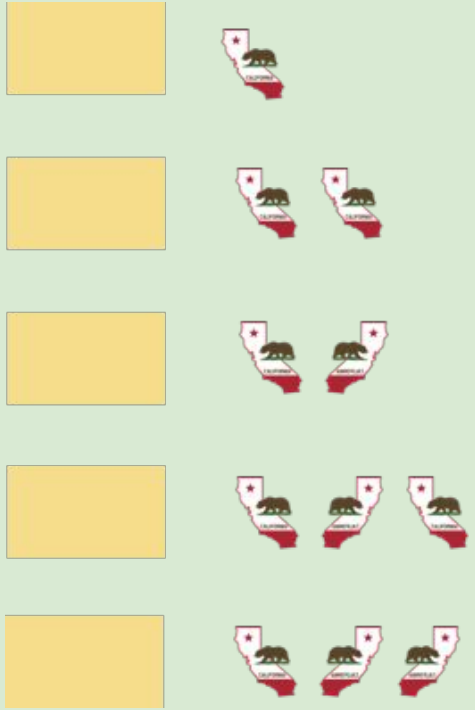
```
['chain', 'dot', 'com ']
```

```
const doStuff = _.compose(
  join(''),
  _.filter(x => x.length > 3),
  reverse,
  _.map(trim),
  split(' '),
  toLowerCase
)
```

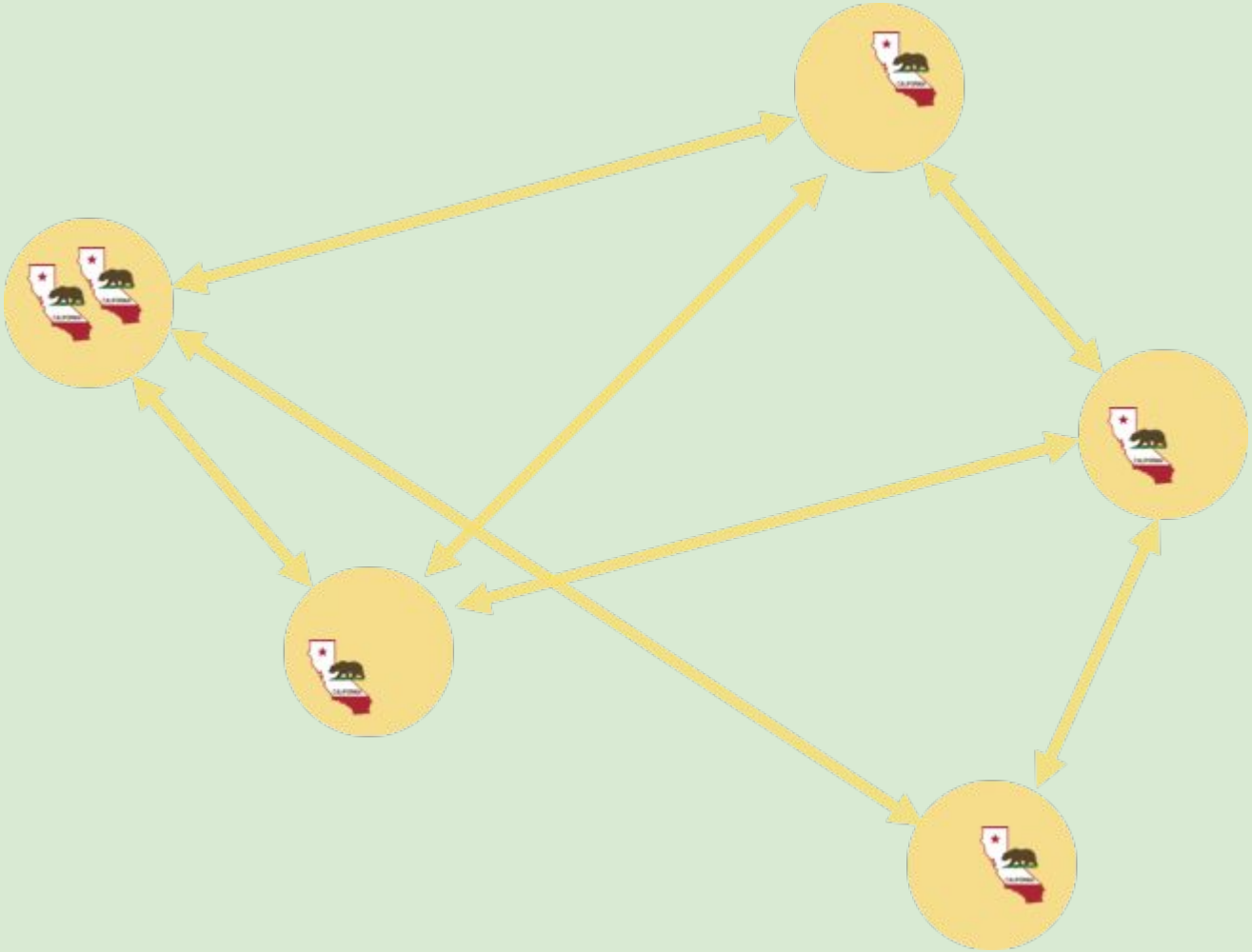
```
['com', 'dot', 'chain']
```

```
const doStuff = str => {
  const lower = str.toLowerCase()
  const words = lower.split(' ')

  words.reverse()

  for(let i in words) {
    words[i] = words[i].trim()
  }

  let keepers = []

  for(let i in words) {
    if(words[i].length > 3) {
      keepers.push(words[i])
    }
  }

  return keepers.join('')
}
```

```
const doStuff = str => {
  const lower = str.toLowerCase()
  const words = lower.split(' ')

  words.reverse()

  for(let i in words) {
    words[i] = words[i].trim()
  }

  let keepers = []

  for(let i in words) {
    if(words[i].length > 3) {
      keepers.push(words[i])
    }
  }

  return keepers.join('')
}
```



```
class AppMailer {

  constructor() {
    this.emailer = new Emailer()
  }

  removeInvalidAddresses() {
    for(let i in this.addresses) {
      if(!this.addresses[i].match(/@/)) {
        this.addresses.splice(i, 1)
      }
    }
  }

  sendEmail({from, to}) {
    this.addresses = to
    this.emailer.setSender(from)
    this.removeInvalidAddresses()
    this.emailer.setRecipients(this.addresses)
    this.emailer.send()
  }
}
```



```
const doStuff = str =>
  str
  .toLowerCase()
  .split(' ')
  .map(c => c.trim())
  .reverse()
  .filter(x => x.length > 3)
  .join('')
```

```
const doStuff = _.compose(
  join(''),
  _.filter(x => x.length > 3),
  reverse,
  _.map(trim),
  split(' '),
  toLowerCase
)
```

Exercises

- Curry: https://codepen.io/drboolean/pen/OJJOQMx?editors=0010
- Compose: https://codepen.io/drboolean/pen/zYYPmZO?editors=0010
- Box: https://codepen.io/drboolean/pen/poodxOm?editors=0010
- Either: https://codepen.io/drboolean/pen/xgoeWR?editors=0010
- Task: https://codepen.io/drboolean/pen/Mparbp?editors=0010

Code
https://drive.google.com/file/d/1x5R9nq13smlXP2R0B75LtZQ1nz6EREUv/view?usp=sharing