Marius Jurgelenas & Karolis Masiulis

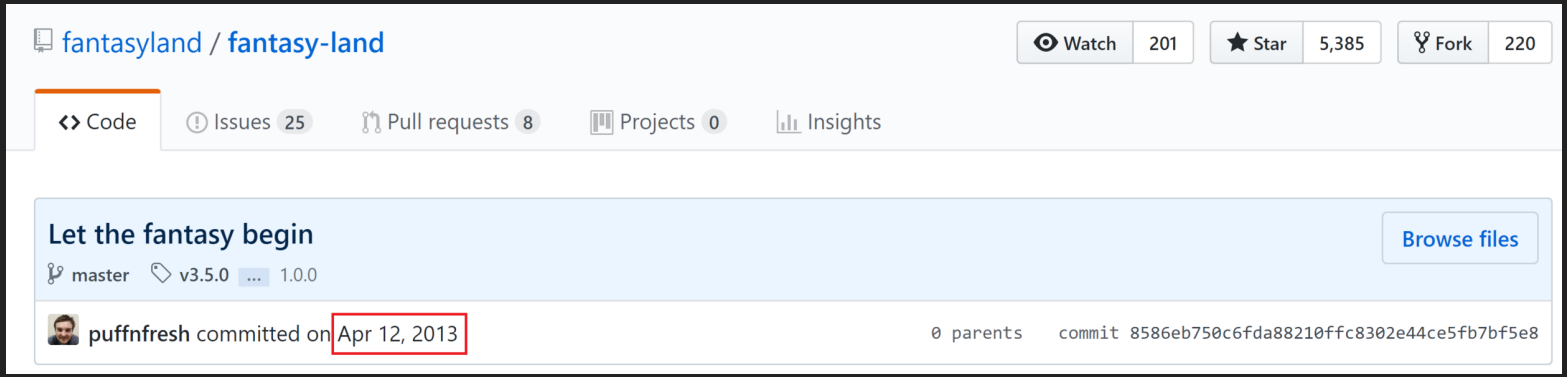
Functional Programming in the Fantasy Land

VilniusJs

2018

Functional Programming in the Fantasy Land
By Marius Jurgelenas

```
(x) => x
```

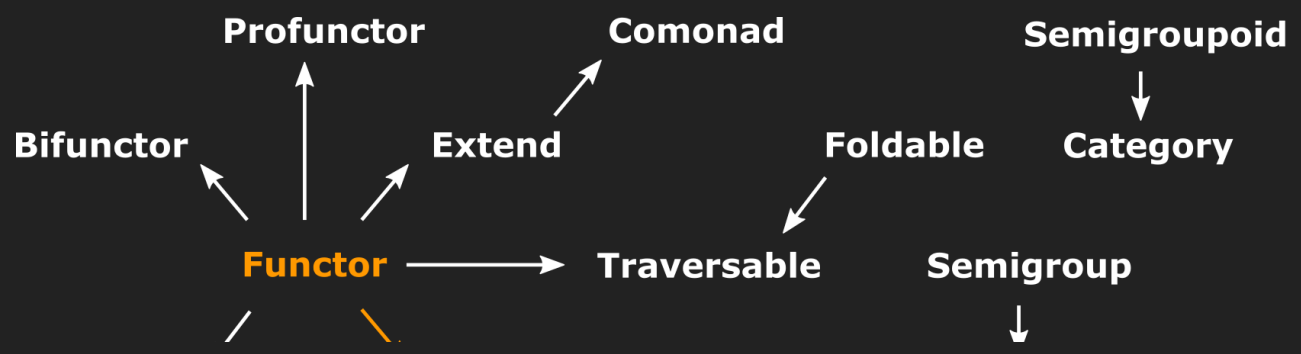


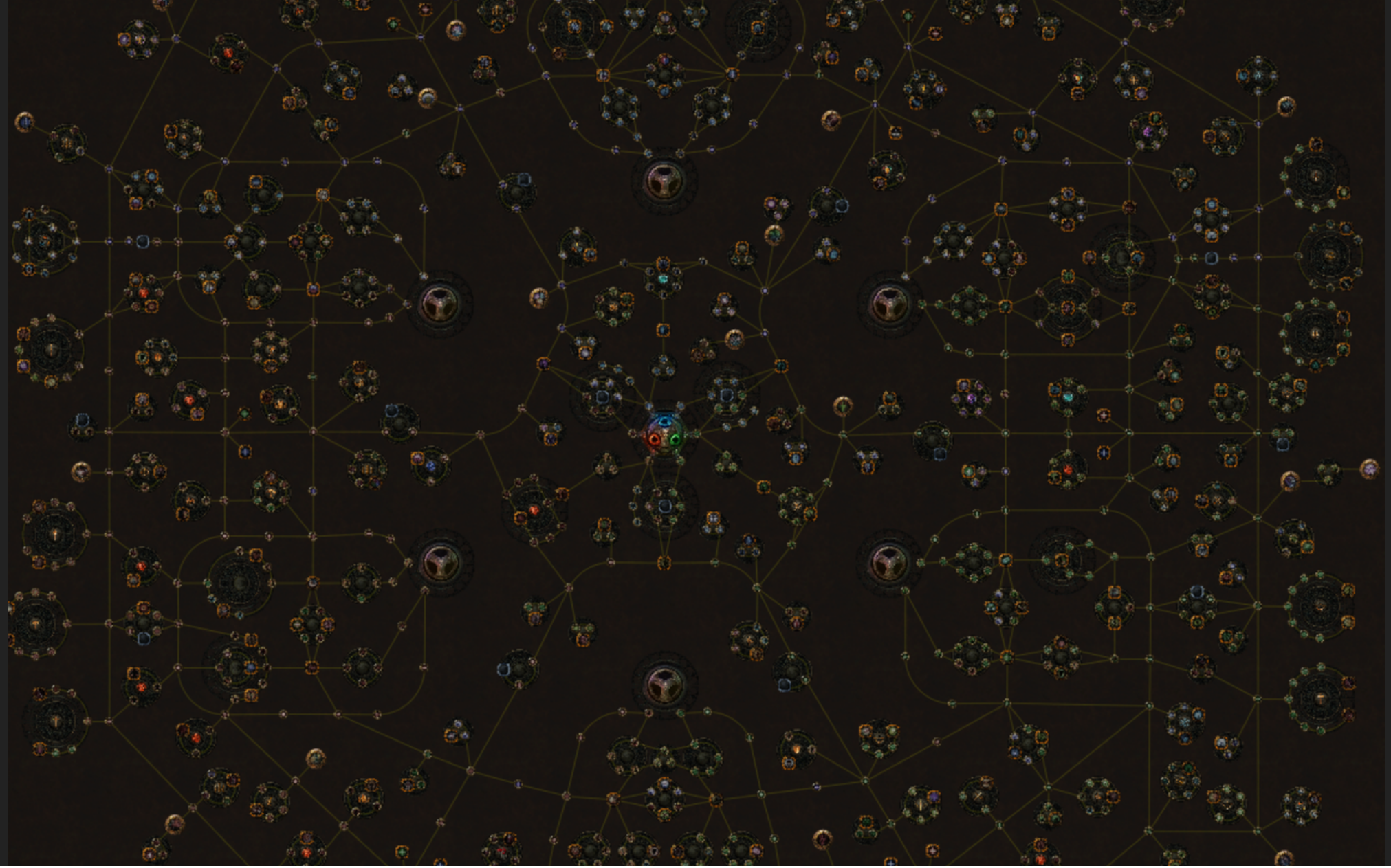
Fantasy Land

https://github.com/fantasyland/fantasy-land

- Method Signatures
- Laws
- Constraints

Specified in JavaScript by Fantasy Land spec





Functor

Applicative

Apply

Chain

Monad

Functor

Method signatures

```
map :: Functor f => f a ~> (a -> b) -> f b
```

Laws

Identity

```
u.map(a => a) === u
```

Composition

```
u.map(x => f(g(x))) === u.map(g).map(f)
```

Extended Hindley-Milner type system

method type type return

```
[1,2,3].map(x => x + 1)  // [2,3,4]
```

Functor

Constraints

```
u.map(f)
```

1. f must be a function,
    - If f is not a function, the behaviour of map is unspecified.
    - f can return any value.
    - No parts of f's return value should be checked.
2. map must return a value of the same Functor

Functor

Applicative

Apply

Chain

Monad

Applicative

Method signature

```
of :: Applicative f => a -> f a
```

A value which has an Applicative must provide an of function on its type representative. The of function takes one argument:

```
F.of(a)
```

Functor

Applicative

Apply

Chain

Monad

Apply

Method signature

```
ap :: Apply f => f a ~> f (a -> b) -> f b
```

Functor vs Apply

```
map :: Functor f => f a ~>   (a -> b) -> f b

ap  ::   Apply f => f a ~> f (a -> b) -> f b
```

Functor

Applicative

Apply

Chain

Monad

Chain

Method signature

```
chain :: Chain f => f a ~> (a -> f b) -> f b
```

Functor vs Apply vs Chain

```
map :: Functor f => f a ~>   (a -> b) -> f b

 ap ::   Apply f => f a ~> f (a -> b) -> f b

chain ::   Chain f => f a ~> (a -> f b) -> f b
```

Functor

Applicative

Apply

Chain

Monad

Monad

A Monad must implement the Applicative and Chain specifications.

There is one operation to manipulate values of the monad (chain), and an operation to put values into a monad (of)

Thank you!