# INTRO

A function in the world of E-things

World of normal things

World of Elevated things

Int -> String

E<Int> -> E<String>

**map**

A function in the   
world of normal things

A value in the world of E-things

World of normal values

World of Elevated values

**return**

A value in the world of normal things

Int

E<Int>

World of normal things

Int, String, Bool

World of Elevated things

E<Int>, E<String>, E<Bool>

World of normal things

Int->String, Int->Bool

World of Elevated things

E<Int>->E<String>, E<Int>->E<Bool>

# MAP

Elevated World

Normal World

E<a>

a

E<b>

b

**map**

Elevated World

Normal World

E<a>

a

E<b>

b

Elevated World

Normal World

E<a>

a

E<b>

b

A function in the world of E-things

A function in the world of normal things

a->b

E<a>

**map**

E<b>

**output**

# FUNCTOR LAWS

Lifted f

**map**

Lifted g

f

g

**map**

**compose**

**compose**

**map**

Same function!

h

Lifted h

**map**

Lifted id

id

id

**=**

# APPLY

**apply**

Elevated World

E<(a->b)>

Elevated World

E<a>

E<b>

E<(a->b->c)>

E<a>

**apply**

**output**

E<b>

E<c>

**apply**

E<(a->b)>

E<a>

**apply**

E<b>

**output**

**return**

Elevated World

Normal World

E<a>

a

Elevated World

Normal World

E<a>

a

A value in the world of normal things

A value in the world of E-things

E<(a->b)>

a->b

a

b

E<a>

E<b>

**apply**

E<a>

E<b>

**map**

**=**

**lift2**

Elevated World

Normal World

E<a>

a

E<b>

b

Elevated World

E<a>

E<b>

c

E<c>

Normal World

a

b

c

E<c>

E<b>

**combine**

E<(a,b)>

**output**

E<a>

# APPLY LAWS

Lifted f

**return**

Lifted x

f

x

**return**

**apply**

**apply**

**return**

Same value!

y

Lifted y

E (id)

**return**

id

**apply**

? (E<a> -> E<a>)

id

**=**

# BIND

**bind**

Elevated World

Normal World

E<a>

a

E<b>

Elevated World

Normal World

E<a>

a

E<b>

A

E<B>

B

E<C>

C

E<D>

E<C>

**bind**

**bind**

🗶

E<B>

A

E<B>

✓

🗶

E<C>

E<D>

✓

a -> E<b>

E<a>

**bind**

E<b>

**output**

# MONAD LAWS

a

b

E<b>

**return**

E<a>

E<b>

**bind**

a

b

E<a>

E<b>

**map**

**=**

a

E<b>

**f**

a

E<a>

**return**

a

E<b>

**f**

E<a>

**bind**

E<a>

E<b>

E<a>

**apply**

E<b>

**output**

a

**f**

a

a

E<b>

**f**

**apply**

a

E<b>

**output**

a

E<a>

**return**

a

E<a>

**return**

E<a>

**bind**

**=**

id

E<a>

E<b>

E<a>

**apply**

E<b>

**output**

a

**f**

a

# APPLICATIVE VS MONADIC

E<createCustomer>

E<CustomerId>

**apply**

createCustomer

**return**

E<EmailAddress>

**apply**

int

string

createCustomerId

createEmailAddress

E<CustomerId>

E<EmailAddress>

**bind**

int

string

createCustomerId

createEmailAddress

createCustomer

**bind**

E<CustomerInfo>

**return**

E<CustomerInfo>

**output**

Result

Option (cons)

Option<int>

cons

string

parseInt

**apply**

**return**

Option [remaining list]

**apply**

[remaining list]

mapOption

Async

List

Result

List

Async

List<Async<Result<a>>>

Async<Result<List<a>>>

Option<int>

Option<int>

**bind**

string

string

parseInt

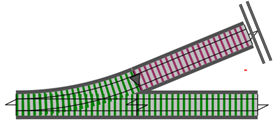
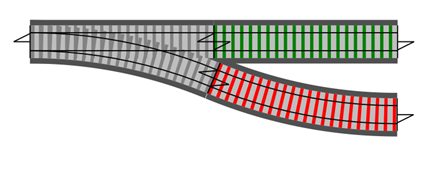
parseInt

createCustomer

**bind**

E<CustomerInfo>

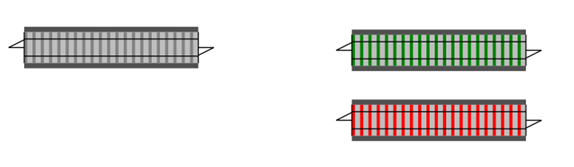
**return**



Validate

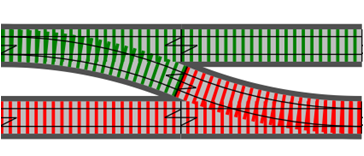
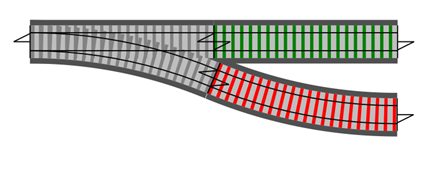
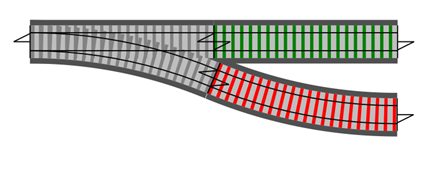
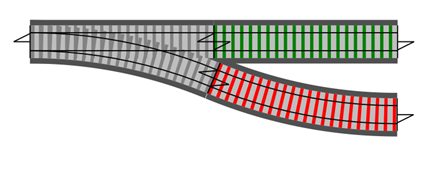
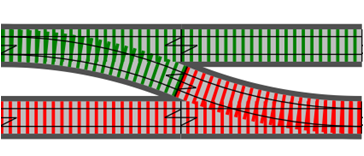
Canonicalize

DbFetch



DbUpdate

DbUpdate

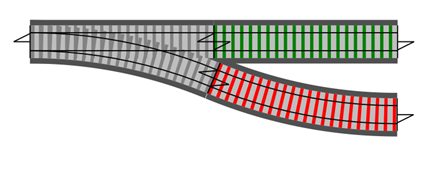
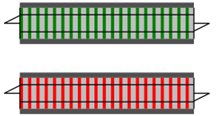
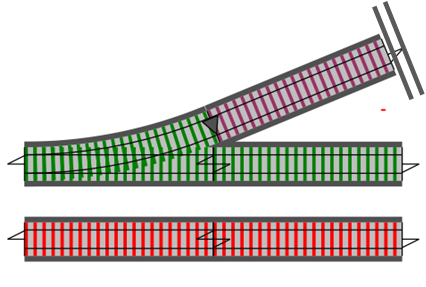
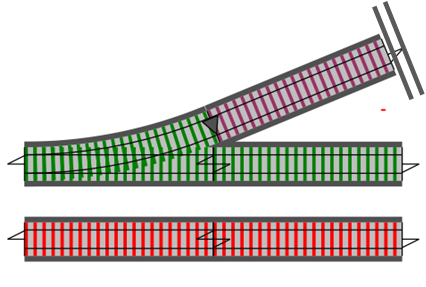


**bind**

**map**

**tee**

**map**

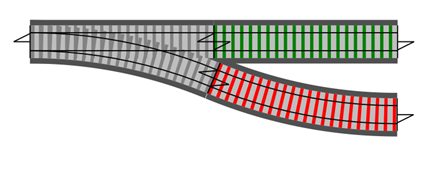
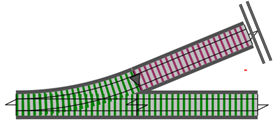
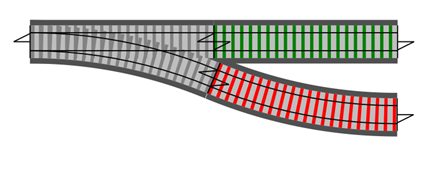
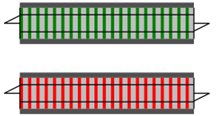
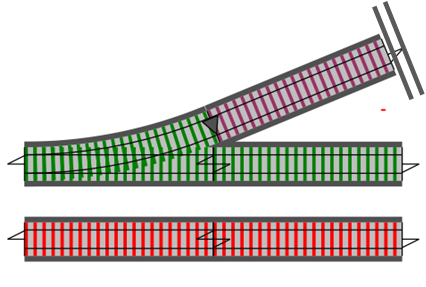
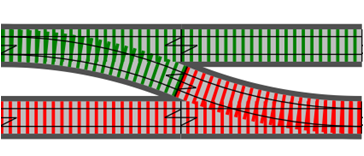


Validate

Canonicalize

DbFetch

DbUpdate

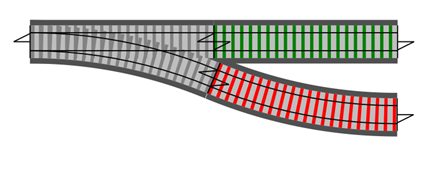
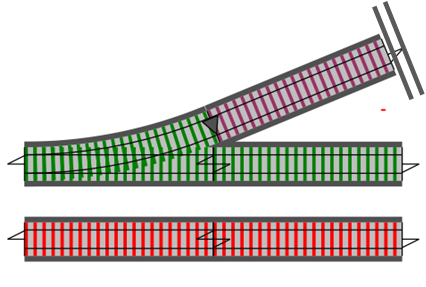


**>=>**

**toSwitch**

**tee**

**toSwitch**

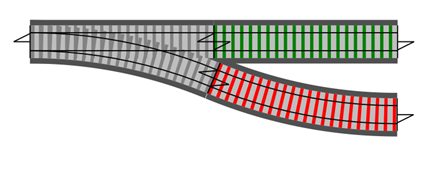
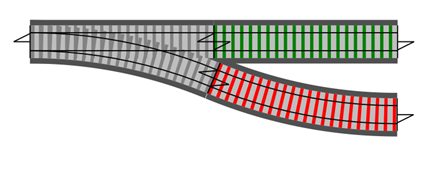


Validate

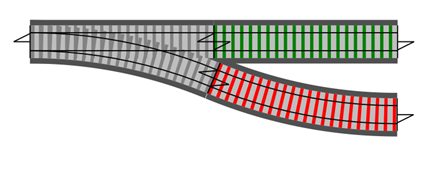
Canonicalize

DbFetch

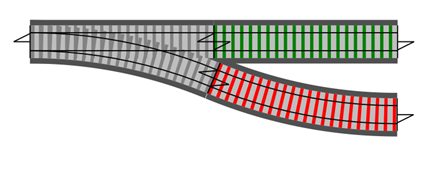
DbUpdate



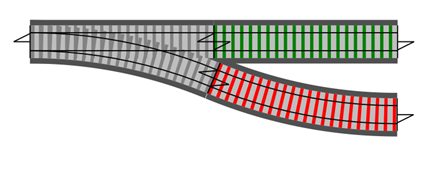
**>=>**



**=**



**>=>**



**>=>**

# Traversable

**traverse**

Elevated World

Normal World

a

E<b>

Elevated World

Normal World

E<b list>

a list

Normal World

Applicative World

Normal World

Result<List<b>>

**traverse f**

Traversable World

Normal World

List<a>

List<Result<b>>

**map f**

Traversable World

Normal World

Traversable World

List<a>

where f =   
(a->Result<b>)

Traversable World

Applicative World

Applicative World

Normal World

Result<b>

**cross world  
function “f”**

Normal World

a

f = (a->Result<b>)

Applicative World

Normal World

Result<a>

Normal World

a

Normal World

List<a>

Traversable World

Normal World

List<Result<a>>

Applicative World

Traversable World

where f =   
(a->Result<b>)

Normal World

Result<List<a>>

Traversable World

Applicative World

**sequence**

# Sequence

Normal World

List<Result<a>>

Applicative World

Traversable World

Normal World

Result<List<a>>

Traversable World

Applicative World

**sequence**

# Sequence Example

List

List<Result<a>>

Result<List<a>>

List

**List.sequenceResult**

Result

AsyncResult

List

List

AsyncResult

**List.sequenceAsyncResult**

Result

Normal

Result

Async

List

Result

List

Async

List<Async<Result<a>>>

Async<Result<List<a>>>

List<Async<Result<a>>>

Async<List<Result<a>>>

**List.sequenceAsync**

Result

List

Async<List<Result<a>>>

Async<Result<List<a>>>

List

**List.sequenceResult**

Result

Async

Async

**Async.map**

Result

Async

List

Result

List

Async

List

Async<Result<List<a>>>

Async<Result<a>>

**List.maxBy**

Result

Async

Async

**Async.map**

Result

**Result.map**

Tuple (Applicative)

Option (Traversable)

List<Option<Tuple<int>>>

List<Tuple<Option<int>>>

Option (Traversable)

**optionSequenceTuple**

Tuple (Applicative)

List (Traversable)

List (Traversable)

**List.map**

Option (Traversable)

Tuple (Applicative)

List (Traversable)

Option (Traversable)

List (Traversable)

Tuple (Applicative)

**listSequenceTuple**

List<Tuple<Option<int>>>

Tuple<List<Option< int>>>

Tuple

Option

List

Option

List

Tuple

List<Option<Tuple<int>>>

Tuple<List<Option<int>>>

retn cons

Option<int>

cons

head

parseInt

**<\*>**

**return**

Option [remaining list]

**<\*>**

tail

mapOption

**bind**

AsyncResult

Normal World

E<a>

a

AsyncResult<b>

<b>

AsyncResult

Normal World

AsyncResult<a>

<b>

a

AsyncResult<b>

<b>

a

Result<b>

Result<a>

Result<b>

**Result.bind**

a

Async<Result<a>>

Async<Result<b>>

**Async.map**

Result<a>

Result<b>

a

ProductId

List

Result

ApiAction

ProductId

🗶

Output of “getPurchaseIds”

Input of “getProductInfo”

ProductId

List

CustId

ApiAction<Result<List<ProductId>>>

CustId

Result

ApiAction

ProductInfo

ProductId

ApiAction<Result<ProductInfo>>

ProductId

Result

ApiAction

ProductInfo

List

CustId

ApiAction<Result<List<ProductInfo>>>

CustId

Result

ApiAction

**getPurchaseIds**

**getProductInfo**

**getPurchaseInfo**

ProductId

List

CustId

ApiAction<Result<List<ProductId>>>

CustId

ApiActionResult

ProductInfo

ProductId

ApiAction<Result<ProductInfo>>

ProductId

ApiActionResult

**getPurchaseIds**

**getProductInfo**

getPurchaseIds

List.Map getProductInfo

ProductId list

CustId

ProductInfo list

ProductInfo

ProductId

**ApiActionResult.bind getProductInfo**

List<ProductId>

ApiAction<Result<List<ProductInfo>>>

ApiActionResult

ProductInfo

ProductId

ApiActionResult

**getProductInfo**

ProductInfo

ProductId

**List.map getProductInfo**

List<ProductId>

ApiAction<Result<List<ProductInfo>>>

List

ApiActionResult

List

🗶

ProductInfo

ProductId

**traverse getProductInfo**

List<ProductId>

ApiAction<Result<List<ProductInfo>>>

List

List

ApiActionResult

✓

**ApiActionResult.bind**

🗶

ApiActionResult

ProductInfo

ProductId

ApiAction<Result<List<ProductId>>>

ApiAction<Result<List<ProductInfo>>>

List

List

ApiActionResult

ApiActionResult

API call

Input

Output

ApiAction generating function

ApiClient

**ApiAction**

ApiClient

Input

Input

Output

ApiClient

Input

Output

ApiClient

NormalWorld

NormalWorld

bind f

E<A>

E<B>

ElevatedWorld

NormalWorld

NormalWorld

ElevatedWorld

f

NormalWorld

NormalWorld

List.map f

List<A>

List<E<B>>

List

ElevatedWorld

List

NormalWorld

NormalWorld

traverse f

List<A>

E<List<B>>

List

List

ElevatedWorld

ElevatedWorld

A

E<B>

ProductId

CustId

ProductId

List

ApiActionResult

ProductInfo

ApiActionResult

**getPurchaseIds**

**getProductInfo**

**???**

CustId

ProductInfo

List

ApiActionResult

**getPurchaseInfo**