Serious Functional Programming in TypeScript: 
*Putting Category Theory to Work for You*

Christian Di Lorenzo
Data Science Engineer • JupiterOne

https://l.rcd.zone/news-repo
2019 Fast Company - Best Workplaces for Innovators

2019 Outside Magazine – The 50 Best Places to Work
Objectives

• Evaluate software correctness and reliability
• Observe “computational patterns”
• Synthesize mathematics and computer science
Top News in US

Britain sets December date for an election in which Brexit will dominate - The Washington Post
Karla Adam, William Booth • washingtonpost.com • an hour ago
Voters will be offered some stark choices over Brexit alongside the usual enthusiastic promises, scary scenarios, misrepresentations and foggy numbers. The Conservative Party under Johnson will run as the 100 percent for Brexit party, under the banner, Lets ...

NFL Trade Deadline Tracker 2019: Rams move on from Aqib Talib, but other rumored deals don't materialize - CBSSports.com
Tyler Sullivan • cbssports.com • an hour ago
The NFL’s midseason game of musical chairs wrapped up on Tuesday afternoon at 4 p.m. ET with the trade deadline. Leading up to this year’s deadline, we’d already seen a solid amount of movement throughout the league. Some of the headliners included Los Angel...

How to share your audio in iOS 13 - The Verge
Jay Peters • theverge.com • an hour ago
It works with AirPods and Beats headphones Last year, when I thought about sharing my audio, I wanted to use an easy workaround ...
Roadmap

1. Introduction to problem space
2. Approaches to computational abstractions
3. Program correctness and reliability
4. Functors, monoids, and monads
5. Putting it all together
1. Introduction to Problem Space

\[ f(x) = x^2 + 4x \]
Composition: The Building Blocks

\[ f \circ g = h \]
Composition: The Building Blocks

```javascript
1 const addOne = (x: number) => x + 1;
2 const subtractOne = (x: number) => x - 1;
3
4 const compose = <A, B, C>(g: (input: A) => B, f: (input: B) => C) => {
5    return (input: A) => f(g(input));
6 }
7
8 const addTwo = compose(addOne, addOne);
9 const subtractTwo = compose(subtractOne, subtractOne);
10 const identity = compose(addOne, subtractOne);
11
12 console.log(addTwo(0));  // => 2
13 console.log(identity(0)); // => 0
14 console.log(subtractTwo(0)); // => -2
```
2. Approaches to Computational Abstractions
Procedural

```javascript
1 // ...
2 } else {
3   for (const e of entities) {
4     let entityId;
5     let entityIds;
6
7     if (e.entityId) {
8       entityId = e.entityId;
9     } else if (e.entityKey) {
10        const query = `Find * with _key='${e.entityKey}'`;
11        const res = await client.queryVI(query);
12        if (res.length === 1) {
13          entityId = res[0].entity._id;
14        } else if (res.length === 0) {
15          console.log(`Skipping entity with _key='${e.entityKey}' - NOT FOUND`);  
16          continue;
17        } else if (res.length > 0) {
18          if (operation !== "delete" && !program.deleteDuplicates) {
19            console.log(`Skipping '${e.entityKey}' - KEY NOT UNIQUE`);
20            continue;
21          }
22          entityIds = res.map(r => r.entity._id);
23       } else {
24         console.log(`Skipping entity with _key='${e.entityKey}'`);
25         continue;
26      }
27  } // ..
28 ```
Object-Oriented Programming

```javascript
export class ArticleList extends AbstractList<Article> {
  constructor(props: any) {
    super(props);
    this.loadData = apiCall;
  }

  renderItem(article: Article, index: number) {
    return (
      <article>
        <a href={article.url}>{article.title}</a>
      </article>
    );
  }
}
```
Object-Oriented Programming

```javascript
export class AbstractList<T> extends React.Component<{}, AbstractListState<T>> {
  loadData: () => Promise<T[]>

  constructor(props: any) {
    super(props);
    this.state = { items: [] }
    this.loadData = () => Promise.resolve([]);
  }

  componentDidMount() {
    this.loadData().then(items => this.setState({ items }));
  }

  renderItem(item: T, index: number) {
    return (<span>TODD: Implement renderItem</span>);
  }

  render() {
    return (
      <div>{this.state.items.map((item, index) => this.renderItem(item, index))}</div>
    );
  }
}
```
```javascript
const loadInitialEvidenceData = async (input: CollectEvidenceBaseInput) => {
    const dataLoader = /* ... */;

    return Promise.all([[
        loadQuestions(input, dataLoader),
        loadSpec(input, dataLoader),
        loadSecurityItems(input, dataLoader),
        loadControlsMapping(input, dataLoader),
        loadRequirements(input, dataLoader)
    ].then(requirements => ({ requirements }))).then(([questions, spec, securityItems, controlsMapping, remaining]) => ({
        ...remaining,
        questions,
        spec,
        securityItems,
        controlsMapping
    }));
};
```
3. Program Correctness and Reliability
Mars Rover Curiosity Snaps Beautiful Selfie After Rare Chemistry Experiment - Space.com

Mike Wall • space.com • 3 hours ago

NASA's Curiosity rover took this selfie on Oct. 11, 2019, the 2,553rd Martian day, or sol, of its mission. The rover drilled twice in this location, which is nicknamed "Glen Etive." NASA's Curiosity Mars rover performed some rare science work recently, the... [+3493 chars]
import axios from 'axios';

const API_KEY = /* actual api key */;

const loadTopHeadlines = async (): Promise<Article[]> => {
  const response = await axios.get(`https://newsapi.org/v2/top-headlines?country=us&api_key=${API_KEY}`);
  return response.data.articles as Article[];
};
Rendering Subtitles

```javascript
interface Article {
  title: string;
  url: string;
  author: string;
  content: string;
  publishedAt: string;
}

const subtitles = (articles: Article[]) => {
  return articles.map(article => {
    const domain = article.url.match(/https?://w?w?w?\.(?([\w\.]+)[\/$]{1,})\![1];
    const relativeTime = dayjs(new Date(article.publishedAt)).fromNow();
    return [article.author, domain, relativeTime].join(' • ');
  });
};
```
interface Article {
  title: string;
  url: string;
  author: Option<string>;
  content: Option<string>;
  publishedAt: Date;
}

export const articleSubtitle = (article: Article): Option<string> => {
  const elements = [
    article.author,
    articleDomainName(article),
    some(articlePublishedAt(article))
  ];
  return pipe(
    elements
    A.map(chain(fromEmptyNullable)),
    A.compact,
    join(' • '),
    fromEmptyNullable
  );
};
4. Functors, Monoids, and Monads
4a. Functors

Functors, Monoids, and Monads
Functors (Haskell)

```haskell
class Functor f where
  fmap :: (a -> b) -> f a -> f b

-- Example implementation: Maybe
data Maybe a = Just a | Nothing

instance Functor Maybe where
  fmap func (Just value) = Just (func value)
  fmap func Nothing = Nothing

-- Example usage
fmap (+1) [1, 2, 3] -- => [2, 3, 4]
fmap (*2) (Just 3) -- => Just 6
fmap (*4) Nothing -- => Nothing

-- Alternate Syntax
(+1) <$> [1, 2, 3]
(*2) <$> (Just 3)
```
Functors (TypeScript)

```
1 // Functor.ts
2 // Instances must satisfy the following laws:
3 //     Identity:
4 //     F.map(fa, a => a) = fa
5 //     Composition:
6 //     F.map(fa, a => bc(ab(a))) = F.map(F.map(fa, ab), bc)
7 export interface Functor<F> {
8     readonly URI: F
9     readonly map: <A, B>(fa: HKT<F, A>, f: (a: A) => B) => HKT<F, B>
10 }
11
12 // Array.ts (converted to be pipeable)
13 export const map = <A, B>(f: (a: A) => B) => (fa: A[]) => B[];
```
Usage

```javascript
export const articleSubtitle = (article: Article): Option<string> => {
  const elements = [
    article.author,
    articleDomainName(article),
    some(articlePublishedAt(article))
  ];

  return pipe(
    elements
    A.map(chain(fromEmptyNullable)),
    A.compact,
    join(' • ')
  );
};
```
4b. Monoids

Functors, Monoids, and Monads
class Monoid m where
  mempty :: m
  mappend :: m -> m -> m

mconcat :: [m] -> m
mconcat = foldr mappend mempty

-- Example implementation: Array
instance Monoid [a] where
  mempty = []
  mappend = (++)

-- Example implementation: Sum
newtype Sum n = Sum n

instance Monoid (Sum n) where
  mempty = 0
  mappend = (+)

-- Example implementation: String
instance Monoid String where
  mempty = ""
  mappend = (++)

-- Example usage
mconcat [(Sum 1), (Sum 2)] -- => (Sum 3)
"Hello" <> " world!" -- => "Hello world!"
Monoids (TypeScript)

```typescript
// Magma.ts, Semigroup.ts, Monoid.ts

export interface Magma<A> {
  readonly concat: (x: A, y: A) => A
}

export interface Semigroup<A> extends Magma<A> {}

export interface Monoid<A> extends Semigroup<A> {
  readonly empty: A
}

const join = (sep: string) => (items: string[]) =>
  intercalate(monoidString, array)(sep, items);

const Url: Monoid<string> = {
  empty: '',
  concat: (left, right) => pipe(
    [left, right],
    map(trimSlash),
    compact,
    join('/')
  )
};

Url.concat('https://newsapi.org', 'v2')
// => https://newsapi.org/v2

Url.concat('https://newsapi.org/', '')
// => https://newsapi.org
```
4c. Monads

Functors, Monoids, and Monads
Monads (Haskell)

```haskell
class Monad m where
  (>>=) :: m a -> (a -> m b) -> m b
  -- ...

-- Example implementation: Maybe
instance Monad (Maybe x) where
  (Just x) >>= k = k x
  Nothing >>= _ = Nothing
  -- ...

lastName :: String -> Maybe String
-- ...implementation of lastName

-- Example usage
(Just "Christian Di Lorenzo") >>= lastName
-- => (Just "Di Lorenzo")
Nothing >>= lastName
-- => Nothing
```
Monads (TypeScript)

```
1 // Apply.ts, Chain.ts, Applicative.ts, Monad.ts
2 export interface Apply<F> extends Functor<F> {
4 }
5
6 export interface Chain<F> extends Apply<F> {
7   readonly chain: <A, B>(fa: HKT<F, A>, f: (a: A) => HKT<F, B>) => HKT<F, B>
8 }
9
10 export interface Applicative<F> extends Apply<F> {
11   readonly of: <A>(a: A) => HKT<F, A>
12 }
13
14 export interface Monad<F> extends Applicative<F>, Chain<F> {}
```
export const articleSubtitle = (article: Article): Option<string> => {
  const elements = [
    article.author,
    articleDomainName(article),
    some(articlePublishedAt(article))
  ];

  return pipe(
    elements
    A.map(chain(fromEmptyNullable)),
    A.compact,
    join(' • ')
  );
};
5. Putting It All Together

Demo
• “fp-ts”: https://gcanti.github.io/fp-ts


• Monads (fastest): https://dev.to/bobbypriambodo/comment/j27

• Haskell: https://wiki.haskell.org

• Category Theory by Steve Awodey