Build frontend applications that never crash with fp-ts ecosystem

 AntonSutarmin

Truly reliable frontend applications with fp-ts ecosystem

 AntonSutarmin

Devexperts


dxtrade.pro
dxTrade
+ Add Widgets
Balance $1,020.43
Equity $1,954.81
Net Liquidity $1,954.81
BP $1,861.39
Day RPL $0.00
Deposit...
Account REAL USD 4365478397
Stocks Options Futures Account
DOW 30
Symbol Bid Ask Last Close
AAPL 189.53 189.56 189.55 190.35
CAT 138.99 139.09 139.00 141.25
CSCO 42.63 42.64 42.63 42.86
CVX 125.87 125.94 125.91 127.59
DIS 105.65 105.71 105.66 106.03
GE 14.07 14.08 14.08 14.17
GS 226.23 226.45 226.40 226.85
IBM 145.72 145.76 145.76 144.71
INTC 51.63 51.64 51.64 52.16
JNJ 127.19 127.21 127.21 127.38
JPM 106.64 106.65 106.65 106.62
KO 44.97 44.98 44.98 44.97
MCD 159.13 159.30 159.13 160.62
MMM 198.91 199.09 199.01 201.48
MRK 62.07 62.09 62.07 62.30
MSFT 101.65 101.67 101.66 102.12
NKE 77.16 77.20 77.17 77.57
PFE 37.28 37.29 37.28 37.43
PG 79.67 79.68 79.69 79.82
TRV 125.33 125.51 125.37 125.87
UNH 254.08 254.29 254.35 255.54
UTX 124.72 124.81 124.76 127.06
V 137.47 137.56 137.56 136.69
VZ 51.38 51.39 51.40 51.34
Chart AAPL Sell 207.99 1 207.99 Buy 1D
29/08/18 09:58
Open 207.03
High 208.74
Low 205.48
Close 207.99
Volume (20)
Value 269
Ichimoku (9, 26, 52, 26)
Tenkan 198.9050
Kijun 195.8250
Chikou 195.8250
SenkouA 186.4550
SenkouB 186.4550
Stochastic (14,3, 3)
%K 96.3371
%D 71.4161
MAR APR MAY JUN JUL AUG
Time and Sales
Time Exchange Name PD Price Size
17:36:47 02/07/18 Direct Edge A 186.38 100
17:36:47 02/07/18 NYSE Arca 186.38 100
17:36:47 02/07/18 BATS 186.38 100
17:36:47 02/07/18 NASDAQ OMX 186.38 10
17:36:47 02/07/18 BATS 186.38 200
17:36:47 02/07/18 NYSE 186.39 100
17:36:46 02/07/18 Direct Edge X 186.39 100
17:36:46 02/07/18 NYSE 186.39 100
17:36:46 02/07/18 FINRA 186.41 100
17:36:46 02/07/18 NYSE Arca 186.39 100
17:36:46 02/07/18 BATS 186.40 100
17:36:46 02/07/18 NASDAQ OMX 186.40 20
17:36:46 02/07/18 NASDAQ OMX 186.40 100
17:36:46 02/07/18 NASDAQ OMX 186.40 100
17:36:46 02/07/18 NASDAQ OMX 186.40 10
17:36:46 02/07/18 NASDAQ OMX 186.40 100
17:36:46 02/07/18 NYSE 186.40 100
17:36:46 02/07/18 BATS 186.40 100
17:36:46 02/07/18 NASDAQ OMX 186.41 33
17:36:46 02/07/18 NASDAQ OMX 186.41 59
17:36:46 02/07/18 NASDAQ OMX 186.41 41
17:36:46 02/07/18 NASDAQ OMX 186.41 2
17:36:44 02/07/18 NASDAQ OMX 186.40 24
17:36:44 02/07/18 NASDAQ OMX 186.40 100
Order Entry Account 4365478397 OSO Buy 500 AAPL @ 181.96 Limit BRK
Chart
Symbol Side Quantity Order Type Limit Price Stop Price Duration Account
OSO AAPL Buy 500 Limit 181.96 DAY 4365478397
BRK AAPL Sell 500 Limit 183.30 DAY 4365478397
BRK AAPL Sell 500 Stop Market 180.25 DAY 4365478397
Oct 2016 Apr Jul Oct 2017 Apr Jul Oct 2018 Apr Jul Aug
Clear data
Confirm and Send

Reliability

The quality of being able to be trusted to do what somebody wants or needs

Reliability

Compliance with tech specs

Edge case coverage

No runtime crashes

How to achieve better reliability?

Static types

Functional paradigm

Runtime type safety

How to achieve better reliability?

Static types

Functional paradigm

Runtime type safety

Static types

- Easier maintenance
- Self-documented code

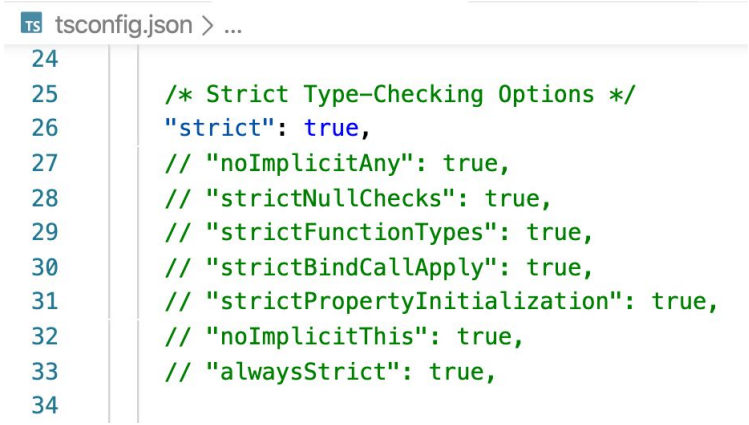
Dynamic types

- Faster initial development
- Better code reusability and expressiveness

https://twitter.com/01k/status/1067788059989684224

Typescript

- Very expressive type system
- Typed superset of JavaScript
- Compiles to plain JS
- Could be used in very different ways



strictNullChecks: false

- null/undefined is a part of every type
- Works like in most modern programming languages

```
type User = {
    id: string;
    name: string;
};

let user: User;

// ...

console.log(user.id);
```

strictNullChecks: false

- null/undefined is a part of every type
- Works like in most modern programming languages

```
type User = {
    id: string;
    name: string;
};

let user: User = null;

// ...

console.log(user.id);
```

strictNullChecks: true

```ts
type User = {
    id: string;
```

```
let user: User
Type 'null' is not assignable to type 'User'. ts(2322)
Peek Problem    No quick fixes available
```

```ts
let user: User = null;

// ...

console.log(user.id);
```

strictNullChecks: true

```ts
type User = {
    id: string;
    name: string;
};

let user: User;
```

```
let user: User
Variable 'user' is used before being assigned. ts(2454)
Peek Problem    No quick fixes available
```

```ts
console.log(user.id);
```

strictNullChecks: true

Works without deceiving a programmer

```
type User = {
    id: string;
    name: string;
};

let user: User;

user = { id: '1', name: 'Anton' };

console.log(user.id);
```

TS: gotchas

Array indexing

```
const arr = [1, 2, 3];

console.log(arr[5].toFixed());
```

TS: gotchas

Array indexing

any type

```
const res1 = Object.create(null);

const res2 = JSON.parse('{"id": 1 }');
```

TS: gotchas

Array indexing

any type

```
type User = {
    id: string;
    name: string;
};

const parsed: User =
     JSON.parse('{"id": 1 }');

console.log(parsed.name);
```

Static types: takeouts

Use responsibly

Don’t try to trick

How to achieve better reliability?

Static types

Functional paradigm

Runtime type safety

Functional paradigm

Immutable data

Pure functions

```
// pure
function add(a: number, b: number) {
    return a + b;
}
```

Functional paradigm

Immutable data

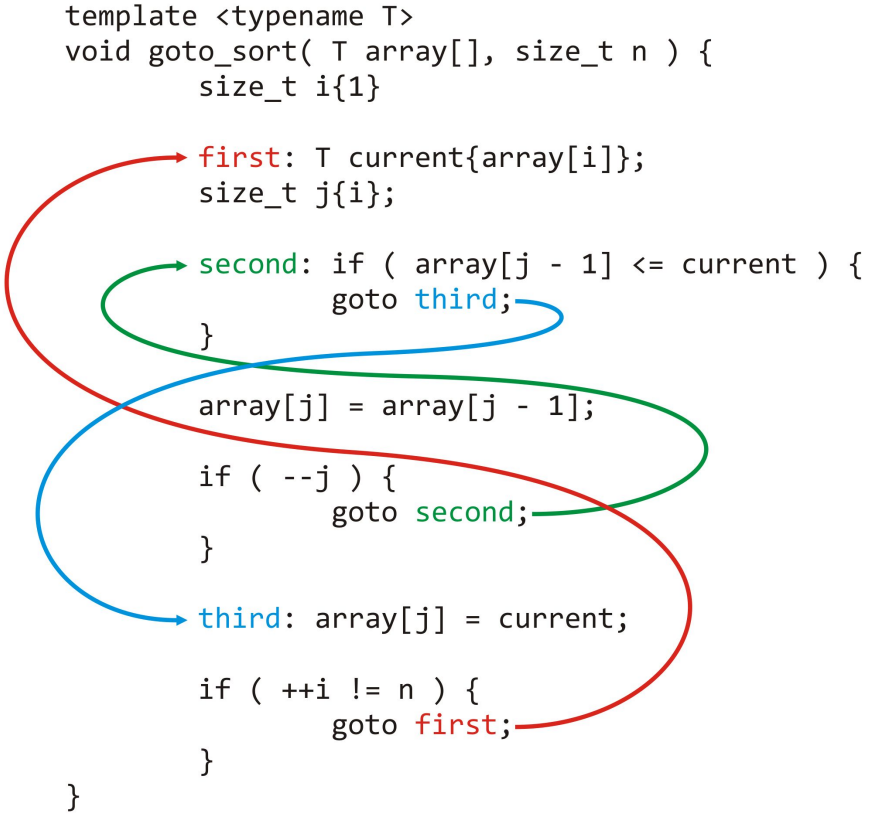
Pure functions

```
// impure
function getValue() {
   return Math.random(); // effect
}

// impure
function getData(url: string) {
   return fetch(url).then(/*...*/);  // effect
}

// impure
function maybeGetValue(arg: boolean) {
   if (arg === false) {
       throw new Error('No value today, sir');  // effect
   }
   // ...
}
```

goto statement



throw is new goto

```
try {

function getUser(id: string): User { /*...*/ };

function getUserOrError(id: string): User { /*...*/ };

function getUser_MAY_THROW(id: string): User { /*...*/ };

} catch (e) { /* good enough? */}
```

FP-way to pass errors

```
function getUser(id: string): Error | User {/*...*/}
```

FP-way to pass errors

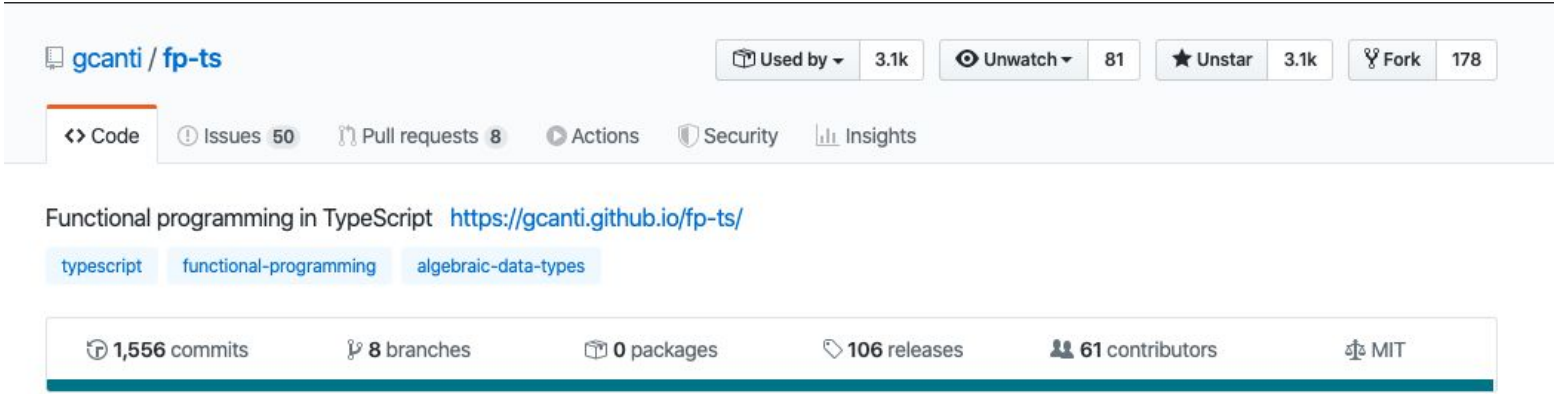
```
function getUser(id: string): Either<Error, User> {/*...*/}
```

Either monad

```
type Left<A> = {
   left: A;
}

type Right<B> = {
   right: B;
}

type Either<A, B> = Left<A> | Right<B>;
```



https://github.com/gcanti/fp-ts

fp-ts

- Library for FP in TS
- Collection of data types and utilities
- Higher Kinded Types in Typescript!

fp-ts/lib/Either

- Can be used for error handling

```ts
import * as E from "fp-ts/lib/Either";

type User = {
   id: string;
   name: string;
};

declare const users: Map<string, User>;

function getUser(id: string): E.Either<Error, User> {
   const user = users.get(id);
   if (user === undefined) {
       return E.left(new Error("Can't find a user"));
   } else {
       return E.right(user);
   }
}
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API

```
import * as E from "fp-ts/lib/Either";
import { pipe } from "fp-ts/lib/pipeable";

const user: E.Either<Error, User> = getUser("1");

const userName = pipe(
   user,
   E.map(user => user.name)
);
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API

```
import * as E from "fp-ts/lib/Either";
import { pipe } from "fp-ts/lib/pipeable";

const user: E.Either<Error, User> = getUser("1");

const userName = pipe(
   user,
   E.map(user => user.name),
   E.filterOrElse(
       name => name.length > 0,
       () => new Error("Name is empty")
   )
);
```

fp-ts/lib/*

- Pipeable API

- Option
- Task
- Reader
- Writer
- IO
- TaskEither
- NonEmptyArray
- State
- Store

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API

```
import * as E from "fp-ts/lib/Either";
import { pipe } from "fp-ts/lib/pipeable";

const user: E.Either<Error, User> = getUser("1");

const userName = pipe(
   user,
   E.map(user => user.name),
   E.filterOrElse(
       name => name.length > 0,
       () => new Error("Name is empty")
   )
);
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API

```
const greeting = pipe(
   userName,
   E.fold(
       () => `Hello, stranger!`,
       name => `Hey, ${name}! What's up?`
   )
);
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API
- No runtime exceptions at all

```
const parsed: E.Either<Error, any> = E.tryCatch(
    () => {
        /* ... */
        return JSON.parse(str);
    },
    () => new Error("Unable to parse value")
);
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API
- No runtime exceptions at all

```
const parsed = E.tryCatch(
    () => {
        /* ... */
        return JSON.parse(str);
    },
    () => new Error("Unable to parse value")
);
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API
- No runtime exceptions at all

```
const parsed: E.Either<Error, any> = E.tryCatch(
    () => {
        /* ... */
        return JSON.parse(str);
    },
    () => new Error("Unable to parse value")
);
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API
- No runtime exceptions at all

```
const parsed: E.Either<Error, ???> = E.tryCatch(
    () => {
        /* ... */
        return JSON.parse(str);
    },
    () => new Error("Unable to parse value")
);
```

fp-ts/lib/Either

- Can be used for error handling
- Pipeable API
- No runtime exceptions at all

```ts
const parsed: E.Either<Error, User> = E.tryCatch(
    () => {
        /* ... */
        return JSON.parse(str);
    },
    () => new Error("Unable to parse value")
);
```

How to achieve better reliability?

Static types

Functional paradigm

Runtime type safety

Runtime validation

```
function safeParseUser(str: string): E.Either<string, User> {
   const parsedUser = JSON.parse(str);
       if (
           typeof parsedUser !== "object" ||
           parsedUser === null ||
           !parsedUser.hasOwnProperty("id") ||
           typeof parsedUser.id !== "string" ||
           !parsedUser.hasOwnProperty("name") ||
           typeof parsedUser.name !== "string"
       ) {
           return E.left(`Object is not a valid user`);
       }
       return E.right(parsedUser);
}
```

Runtime validation

```
function safeParseUser(str: string): E.Either<string, User> {
    const parsedUser = JSON.parse(str);
    if (
        typeof parsedUser !== "object" ||
        parsedUser === null
    ) {
        return E.left(`Parsed value is not an object`);
    }
    if (
        !parsedUser.hasOwnProperty("id") ||
        typeof parsedUser.id !== "string"
    ) {
        return E.left(`Parsed value must have an id property of t
    }
    if (
        !parsedUser.hasOwnProperty("name") ||
        typeof parsedUser.name !== "string"
    ) {
        return E.left(`Parsed value must have a name property of
```

io-ts

- Part of fp-ts ecosystem
- Runtime type validation

```ts
import * as t from 'io-ts';

const userCodec = t.type({
   id: t.string,
   name: t.string
});

function safeParseUser(str: string): E.Either<t.Errors, User> {
   const parsedUser = JSON.parse(str);
   return userCodec.decode(parsedUser);
}
```

github.com/gcanti/io-ts

Implemented types / combinators

Type	TypeScript	codec / combinator
null	`null`	`t.null` or `t.nullType`
undefined	`undefined`	`t.undefined`
void	`void`	`t.void` or `t.voidType`
string	`string`	`t.string`
number	`number`	`t.number`
boolean	`boolean`	`t.boolean`
unknown	`unknown`	`t.unknown`
array of unknown	`Array<unknown>`	`t.UnknownArray`
array of type	`Array<A>`	`t.array(A)`

io-ts: composability

```
import * as t from 'io-ts';

const userCodec = t.type({
   id: t.string,
   name: t.string
});


const postCodec = t.type({
   id: t.string,
   text: t.string,
   author: userCodec,
});
```

io-ts: use cases

- http-requests

```
const user =
fetch("https://localhost/user")
   .then(res => res.json())
   .then(userCodec.decode);
```

io-ts: use cases

- http-requests
- local storage data

```
const user =
fetch("https://localhost/user")
   .then(res => res.json())
   .then(userCodec.decode);
```

```
const userFromLS = JSON.parse(
   localStorage.getItem('user')
);
const user = userCodec.decode(userFromLS);
```

We
are
safe!

Static types

Functional paradigm

Runtime type safety

And there is a lot more!

- fp-ts-routing
- newtype-ts
- ...


dxTechTalk

Stay tuned!

twitter.com/devexperts

facebook.com/devexperts

linkedin.com/company/devexperts

github.com/devexperts

@AntonSutarmin

Questions?

@AntonSutarmin