sugarpirate_ poteto

Learning to Love Type Systems

Swipe Left, Uncaught TypeError 💔

PRESENTED BY

Lauren Tan (she/her)

Cinemagraph by /u/orbojunglist

sugarpirate_ poteto

Learning to Love Type Systems

Swipe Left, Uncaught TypeError 💔

PRESENTED BY

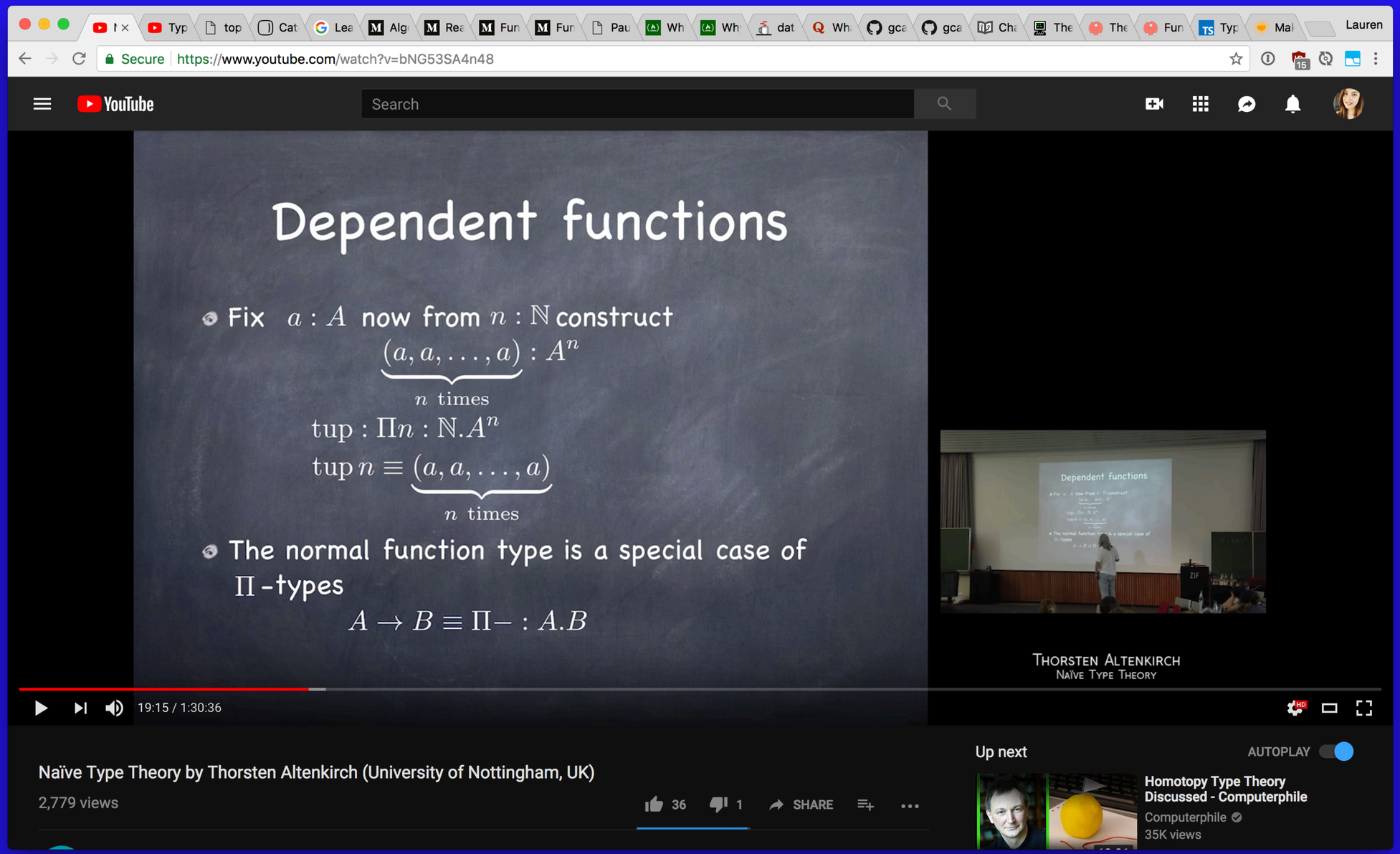
Lauren Tan (she/her)

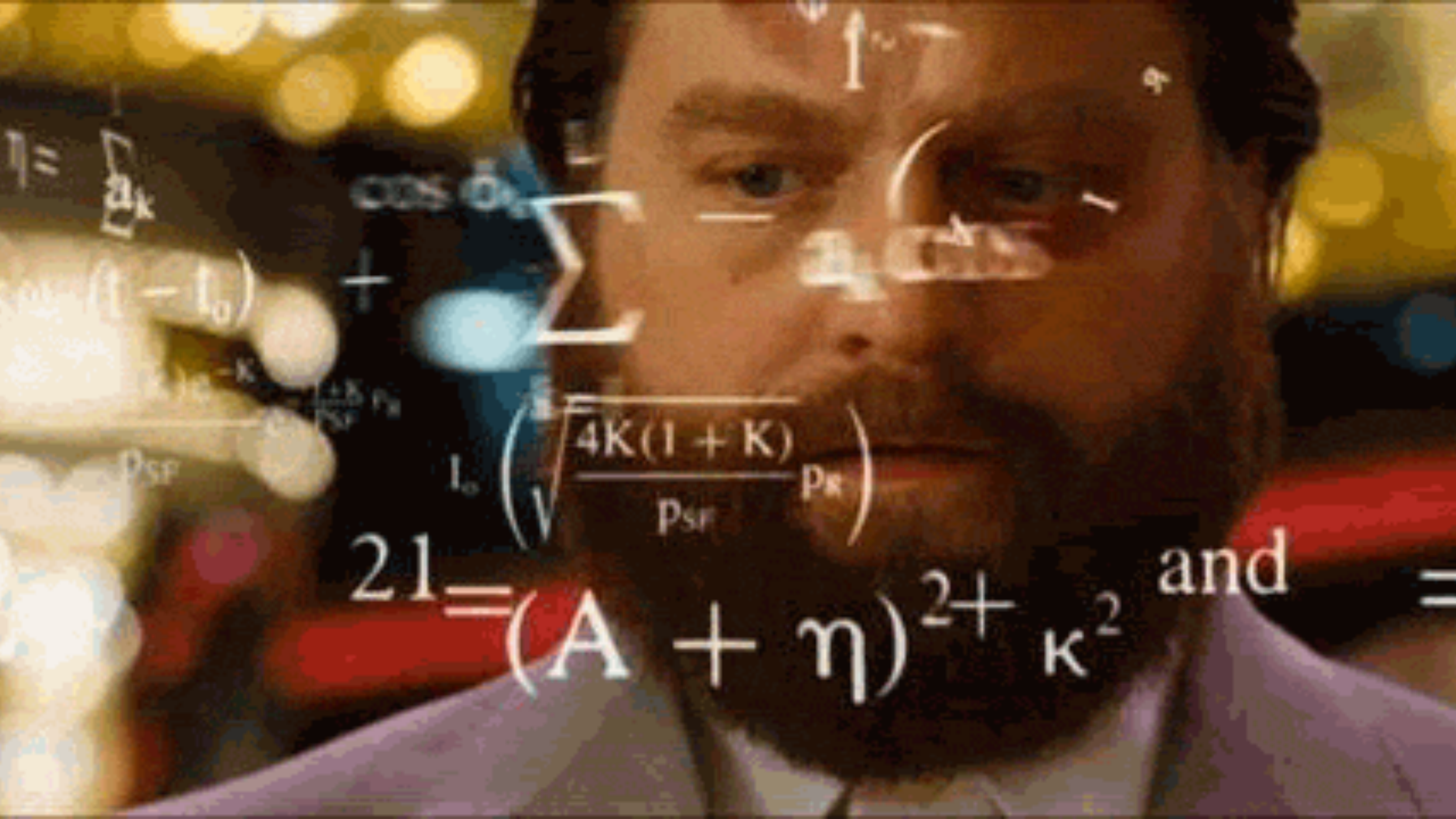
Cinemagraph by /u/orbojunglist

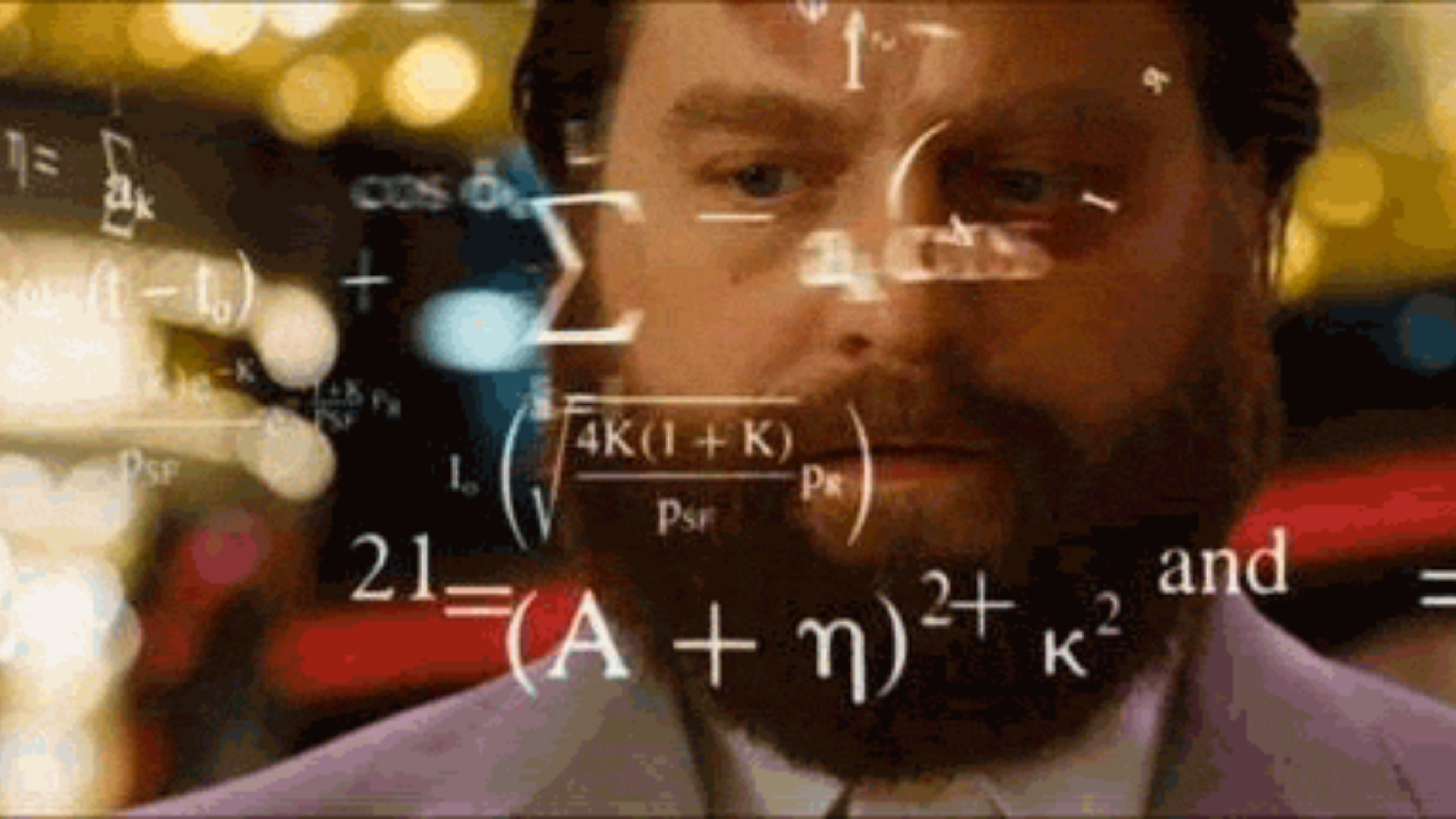
sugarpirate_ poteto

Bonjour

sugarpirate_ poteto




$\eta = \left[a_k \right.$
$\cos \theta$
$\sum$
$\left(t - t_0 \right)$
$\frac{1}{T}$
p_{sr}
$I_o \left(\sqrt{\frac{4K(1+K)}{p_{sr}}} p_k \right)$
$21 = (A + \eta)^2 + \kappa^2$ and

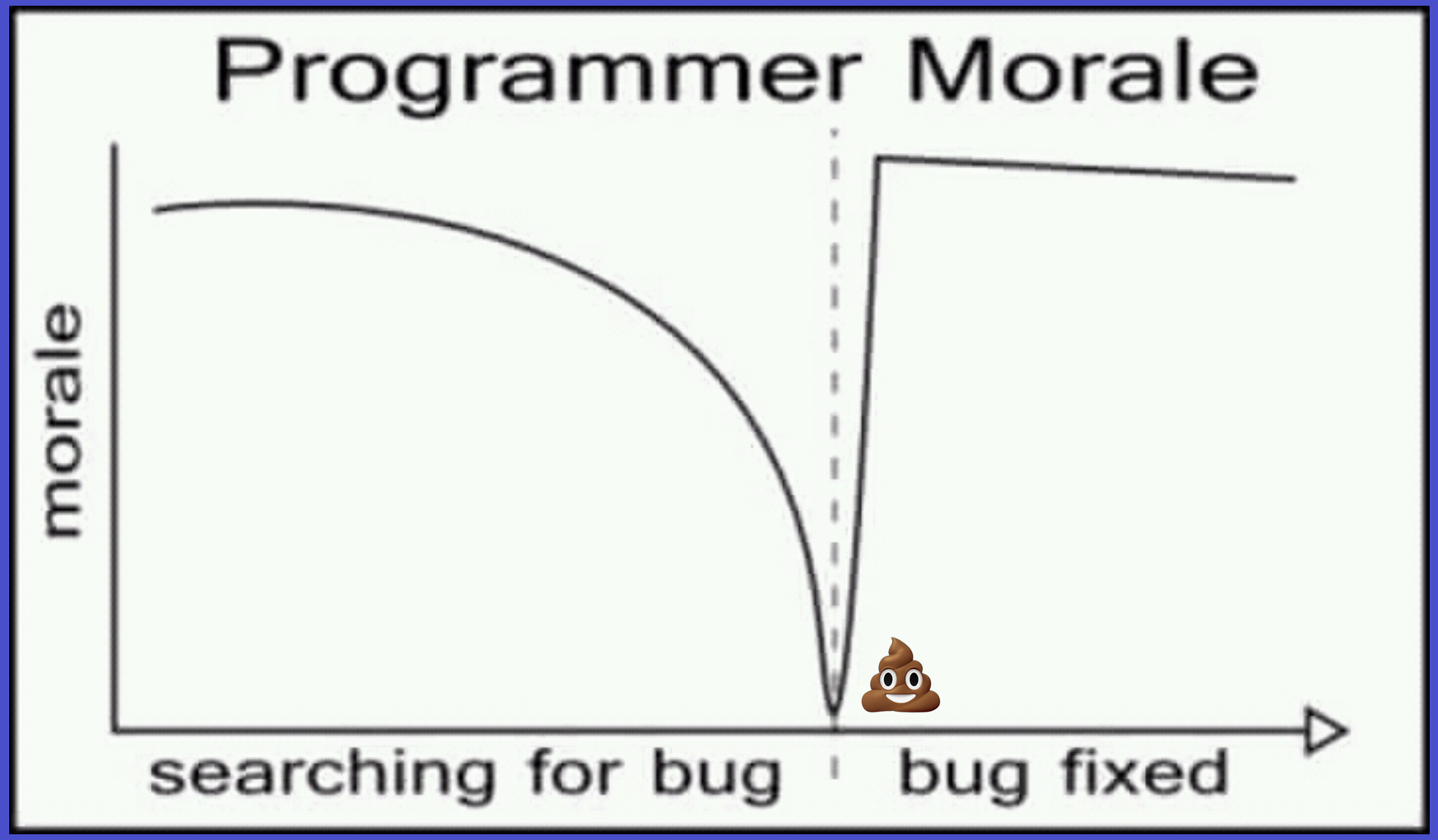

$\eta = \left[a_k \right.$
$\cos \theta$
$\sum$
$\left(t - t_0 \right)$
$\frac{1}{T}$
p_{sr}
$I_o \left(\sqrt{\frac{4K(1+K)}{p_{sr}}} p_k \right)$
$21 = (A + \eta)^2 + \kappa^2$ and

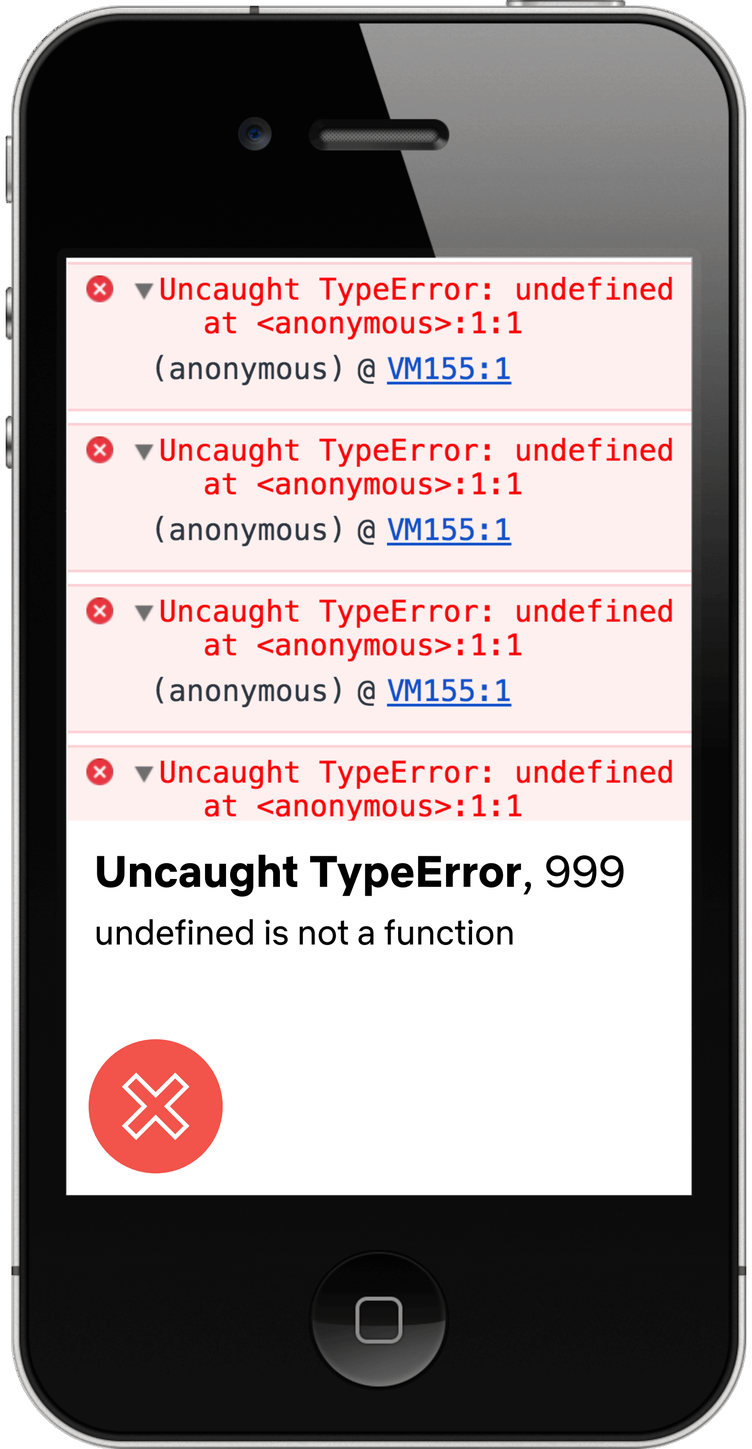
sugarpirate_ poteto

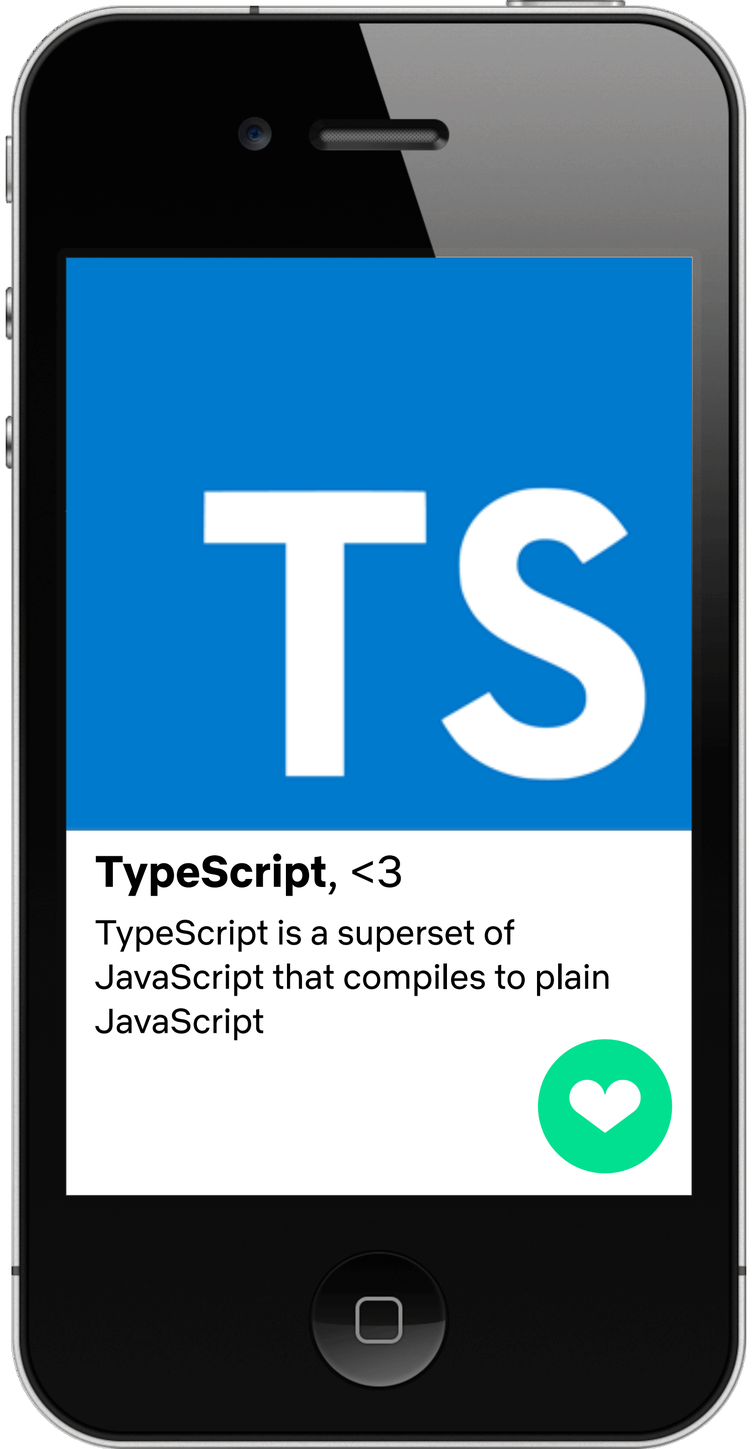
"If debugging is the process of removing software bugs, then programming must be the process of putting them in."

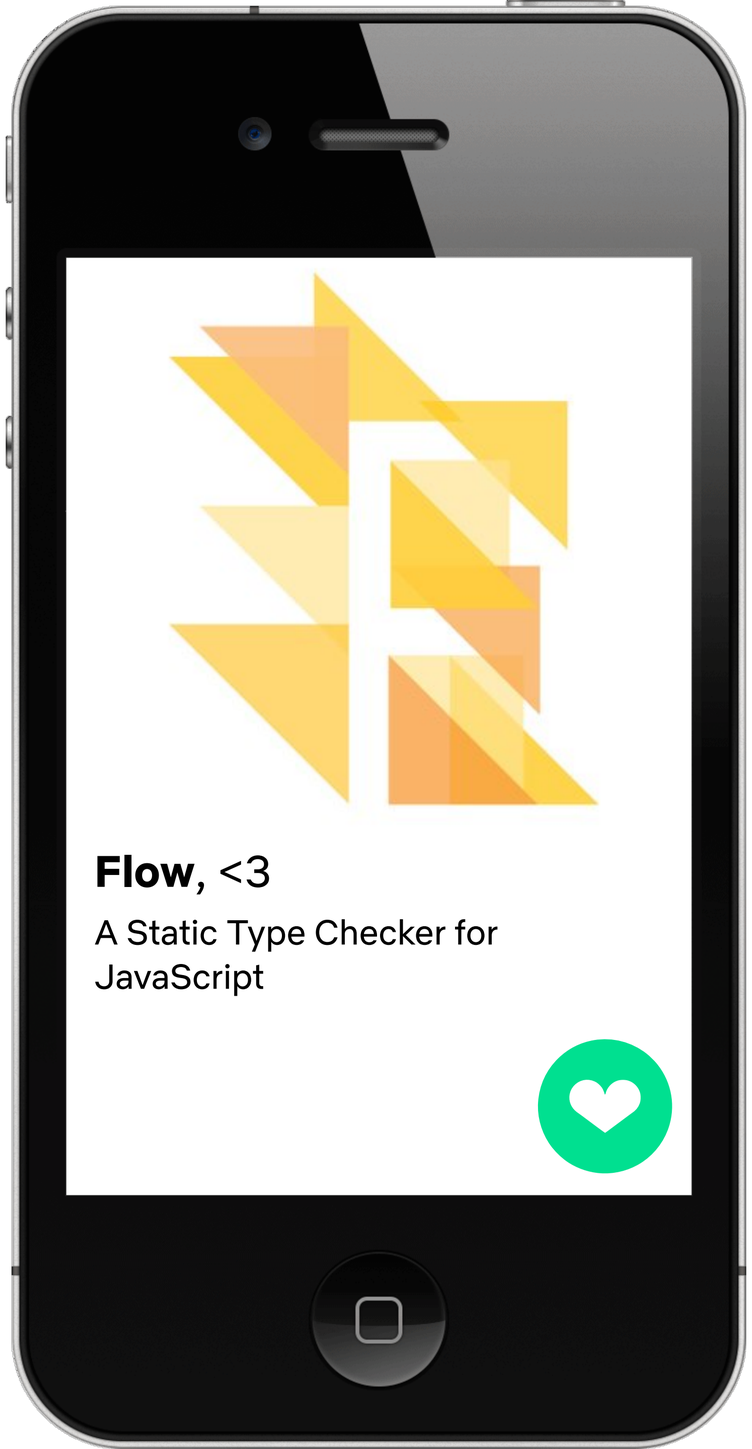
Djikstra, supposedly 🤷‍♀️

sugarpirate_ poteto




Uncaught TypeError: undefined
at <anonymous>:1:1
(anonymous) @ VM155:1
Uncaught TypeError: undefined
at <anonymous>:1:1
(anonymous) @ VM155:1
Uncaught TypeError: undefined
at <anonymous>:1:1
(anonymous) @ VM155:1
Uncaught TypeError: undefined
at <anonymous>:1:1
Uncaught TypeError, 999
undefined is not a function


TS
TypeScript, <3
TypeScript is a superset of JavaScript that compiles to plain JavaScript


Flow, <3
A Static Type Checker for
JavaScript

sugarpirate_ poteto

"A type system is a tractable syntactic method for **proving the absence of certain program behaviors** *by classifying phrases according to the kinds of values they compute."*

Types and Programming Languages, Benjamin C. Pierce

sugarpirate_ poteto

How many ways can this program fail?

sugarpirate_ poteto

```
const half = x ⟹ x / 2;
```

sugarpirate_ poteto

```
const TEST_CASES = [
  null,
  undefined,
  Symbol(1),
  10,
  '10',
  'hello world',
  { name: 'Lauren' },
  [1, 2, 3],
  x => x * x
];
```

sugarpirate_ poteto

```
TEST_CASES.map(testValue ⟹ {
  return {
    result: half(testValue),
    test: testValue.toString()
  }
});
```

sugarpirate_ poteto

not my type!

```
Uncaught TypeError: Cannot read property 'toString' of null
    at TEST_CASES.map.testValue (<anonymous>:17:21)
    at Array.map (<anonymous>)
    at <anonymous>:14:12
```


OK


OK

sugarpirate_ poteto

donavon/undefined-is-a-funct

Lauren

GitHub, Inc. [US] | https://github.com/donavon/undefined-is-a-function

donavon / undefined-is-a-function

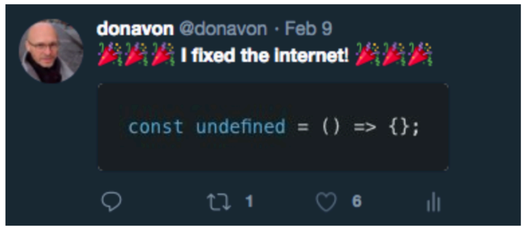
master

- src
- .babelrc
- .gitignore
- .npmignore
- .travis.yml
- LICENSE.md
- README.md
- package-lock.json
- package.json

"undefined is not a function"? — It is now!

build passing npm v0.1.0

A while ago a tweeted that I had fixed the internet.



Well, I guess that wasn't entirely accurate. I gave people the ability for *them* to fix the internet, but not everyone was on board.

So today, I'm making it even easier for you to fix your little corner of the interwebs.

Get the code (i.e. fix the internet)

All you have to do is this:

```
$ npm i --save undefined-is-a-function
```

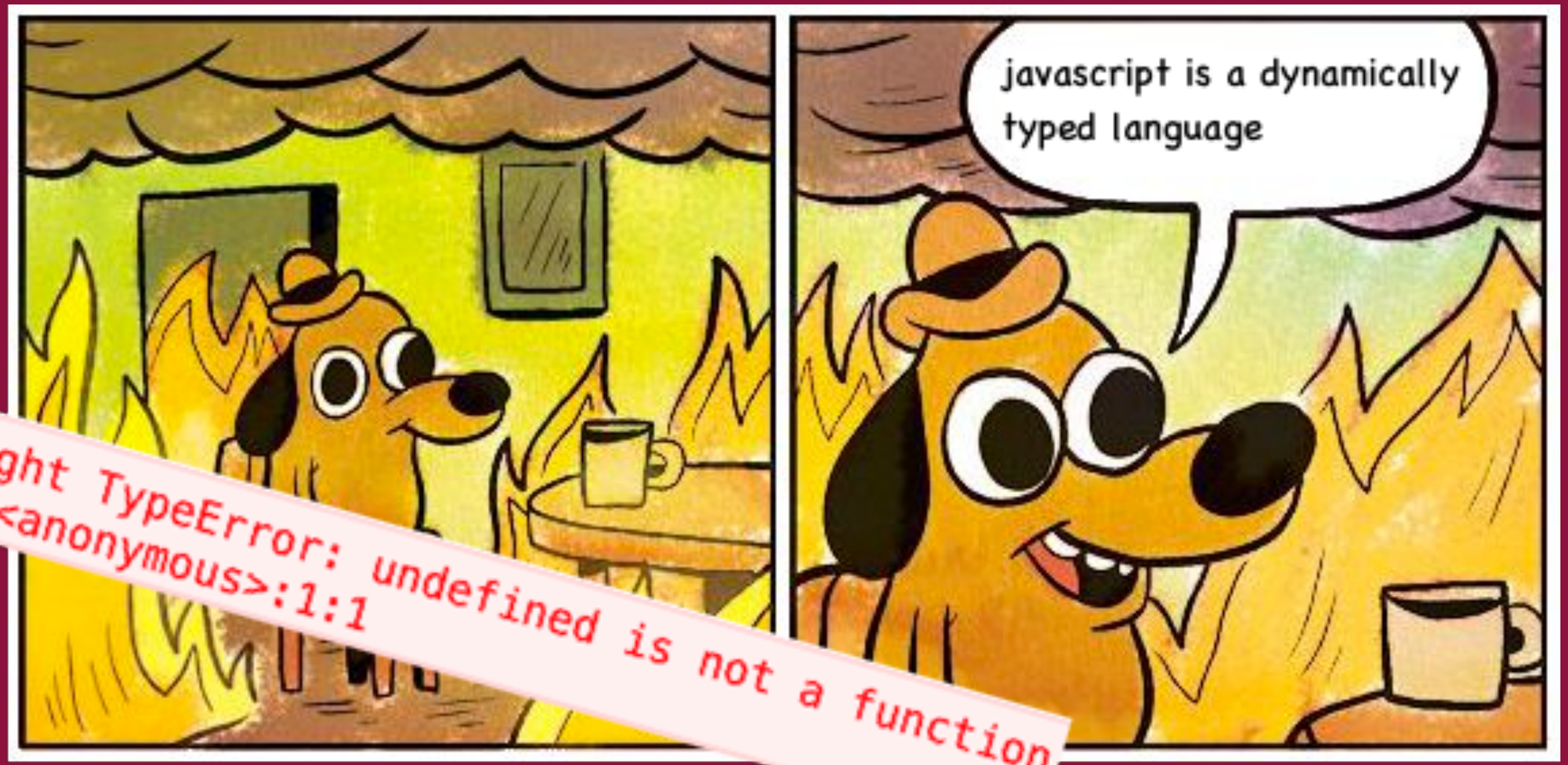
(or if you like yarn)

```
$ yarn add undefined-is-a-function
```

Then in your code:

```
import { undefined } from 'undefined-is-a-function';
```

sugarpirate_ poteto



sugarpirate_ poteto

```
const TEST_CASES = [
  null, // Uncaught TypeError
  undefined, // Uncaught TypeError
  Symbol(1), // Uncaught TypeError
  10, // 5
  '10', // 5 🤷‍♀️
  'hello world', // NaN
  { name: 'Lauren' }, // NaN
  [1, 2, 3], // NaN
  x => x * x // NaN
];
```

sugarpirate_ poteto

How many ways can this program fail?

sugarpirate_ poteto

(Infinity)

How fail?

sugarpirate_ poteto

```
const half = (x: number) ⇒ x / 2;
```

sugarpirate_ poteto

How many ways can this program fail (at compile time)?

sugarpirate_ poteto

0*


Why Less Is Better
Precise Types Means Less Bugs
Cinemagraph by /u/orbojunglist


Why Less Is Better
Precise Types Means Less Bugs
Cinemagraph by /u/orbojunglist

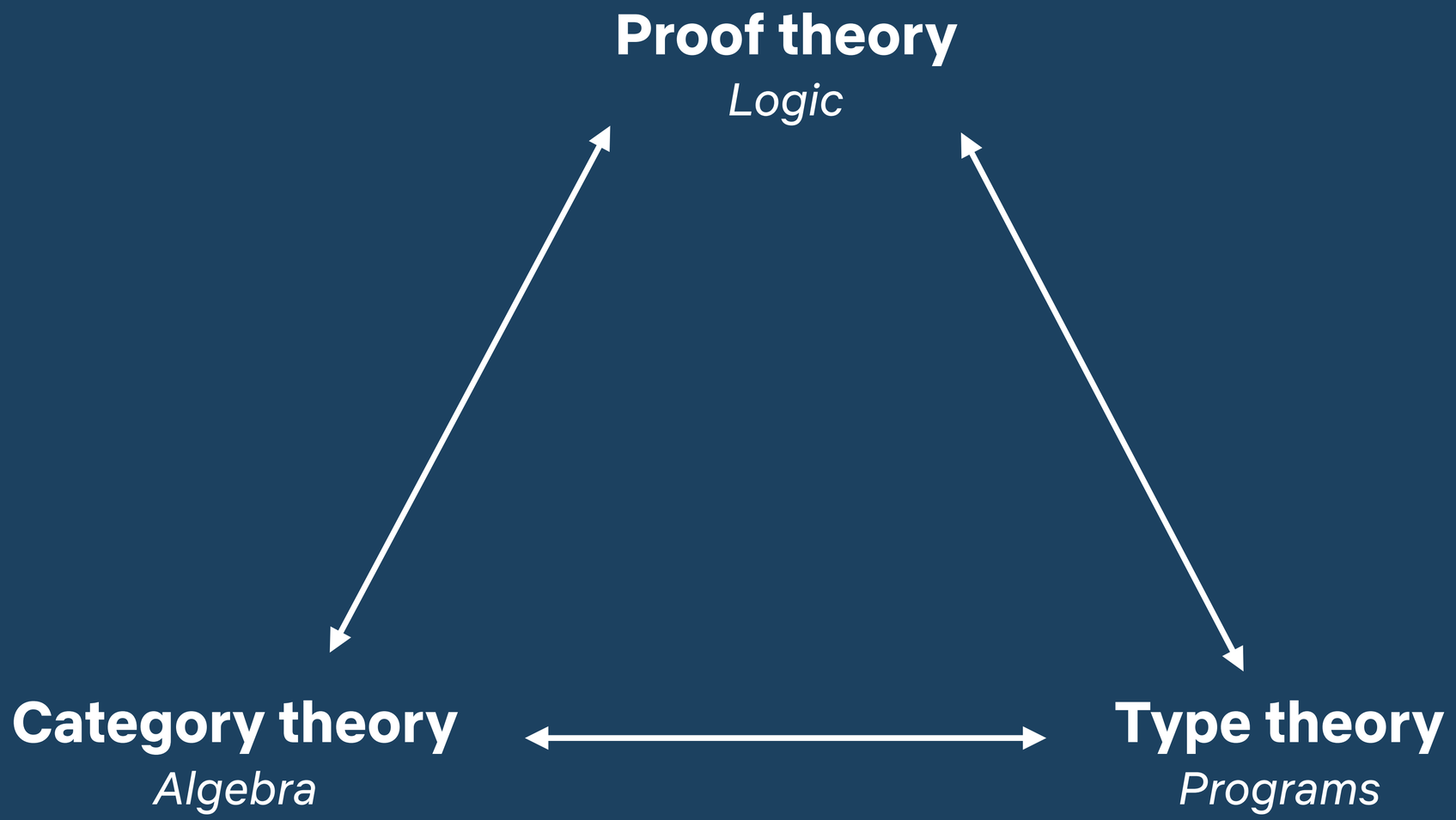
sugarpirate_ poteto

Learning from Functional Programming

https://www.youtube.com/watch?v=ev7AYsLljxk&index=5&list=PL8Ky8IYL8-Oh7awp0sqa82o7Ggt4AGhyf

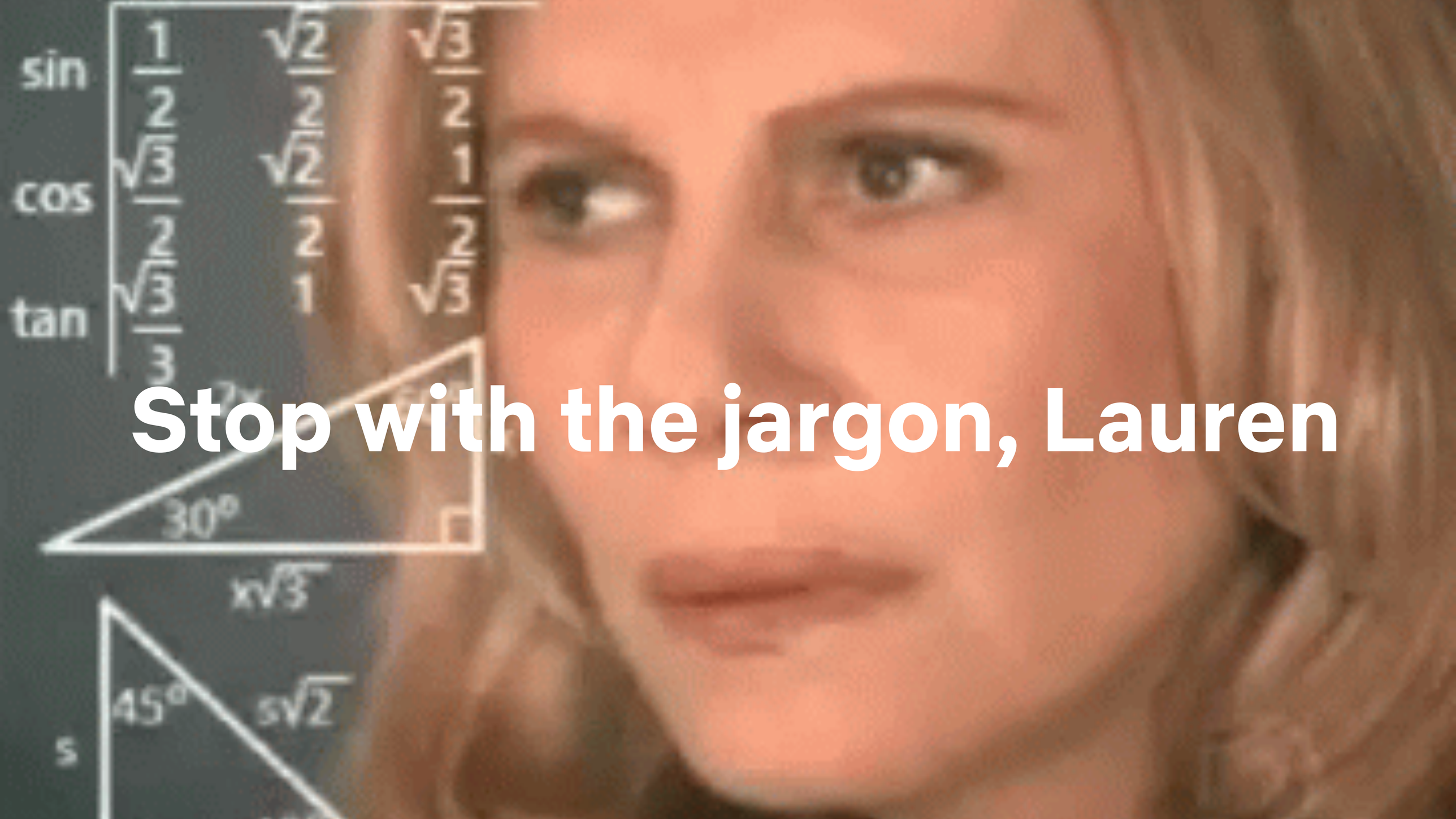
sugarpirate_ poteto

sugarpirate_ poteto

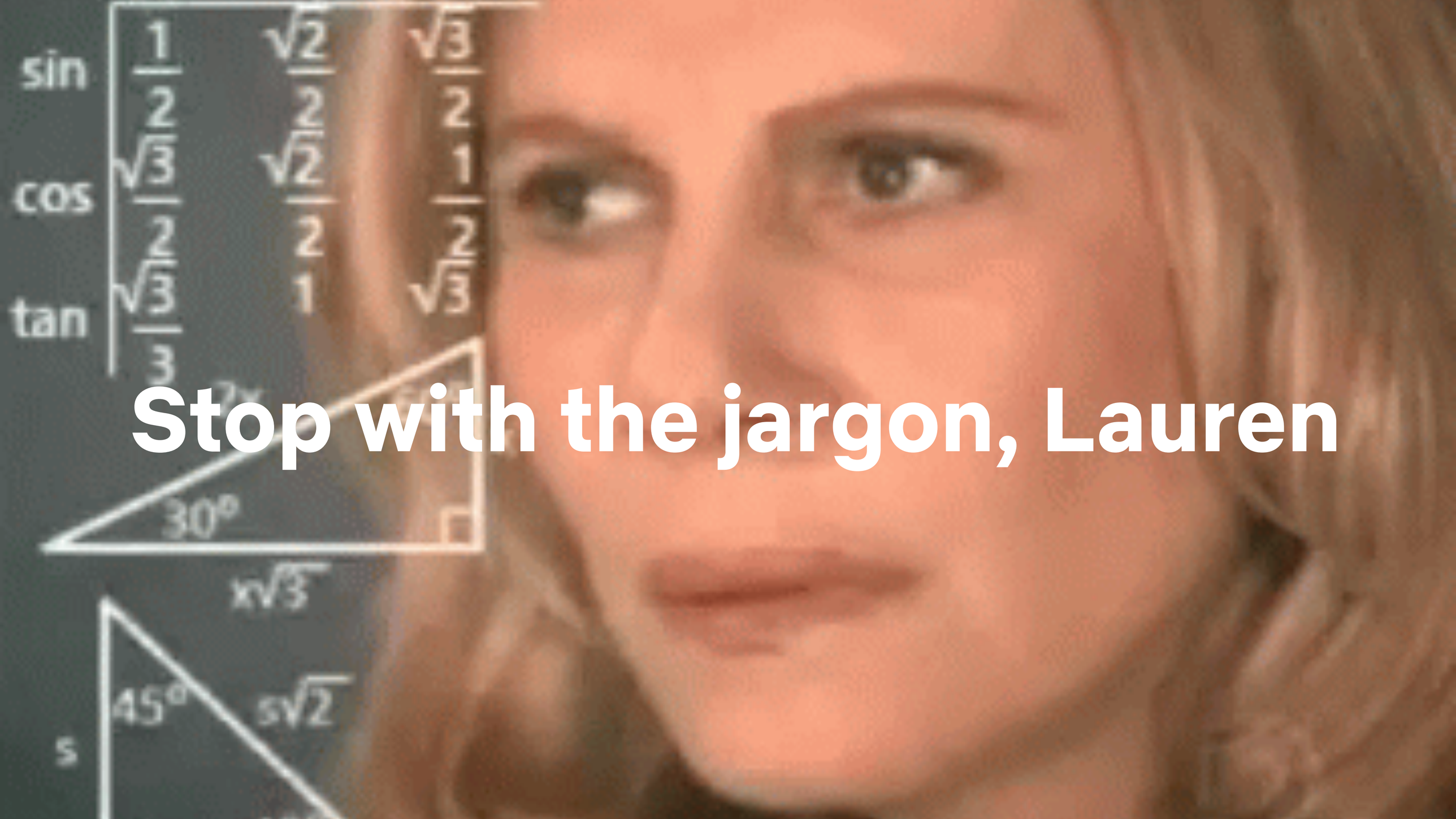


sugarpirate_ poteto

Curry–Howard–Lambek correspondence



Stop with the jargon, Lauren



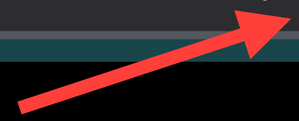
Stop with the jargon, Lauren

sugarpirate_ poteto

```
declare function Addition(x: number, y: number): number;      // proposition
function add(x: number, y: number): number { return x + y; }  // proof
```

sugarpirate_ poteto

Proposition: If x and y are numbers, a number exists



```
declare function Addition(x: number, y: number): number;      // proposition
function add(x: number, y: number): number { return x + y; }  // proof
```

sugarpirate_ poteto

Proposition: If x and y are numbers, a number exists

```
declare function Addition(x: number, y: number): number;      // proposition
function add(x: number, y: number): number { return x + y; }  // proof
```

Proof: x + y is a proof that a number exists

sugarpirate_ poteto

What is a function?

sugarpirate_ poteto

$$f : A \to B$$

sugarpirate_ poteto

```
f :: function from type A to type B
```

** not a JS object*

sugarpirate_ poteto

Types are propositions
Programs are proofs

Curry-Howard Correspondence

sugarpirate_ poteto

Let the type system suggest the implementation

sugarpirate_ poteto

```
function head<T>(list: T[]): T {
  // ...
}
```

sugarpirate_ poteto

```
[ts] Type 'T[]' is not assignable to type 'T'.
(parameter) list: T[]
```

```ts
function head<T>(list: T[]): T {
  return list;
}
```

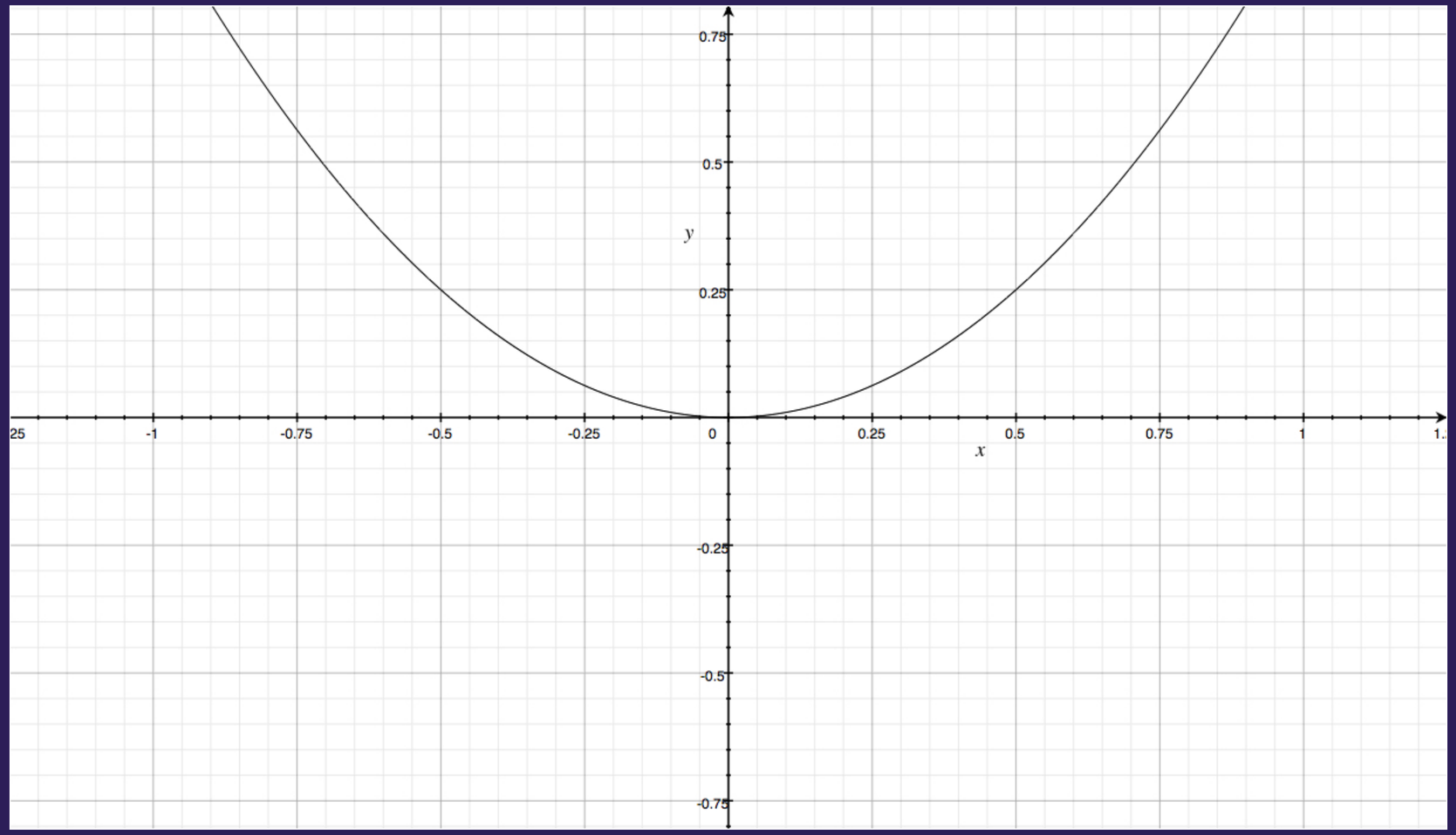
sugarpirate_ poteto

```
function head<T>(list: T[]): T {
  return list[0];
}
```

sugarpirate_ poteto

Writing better functions

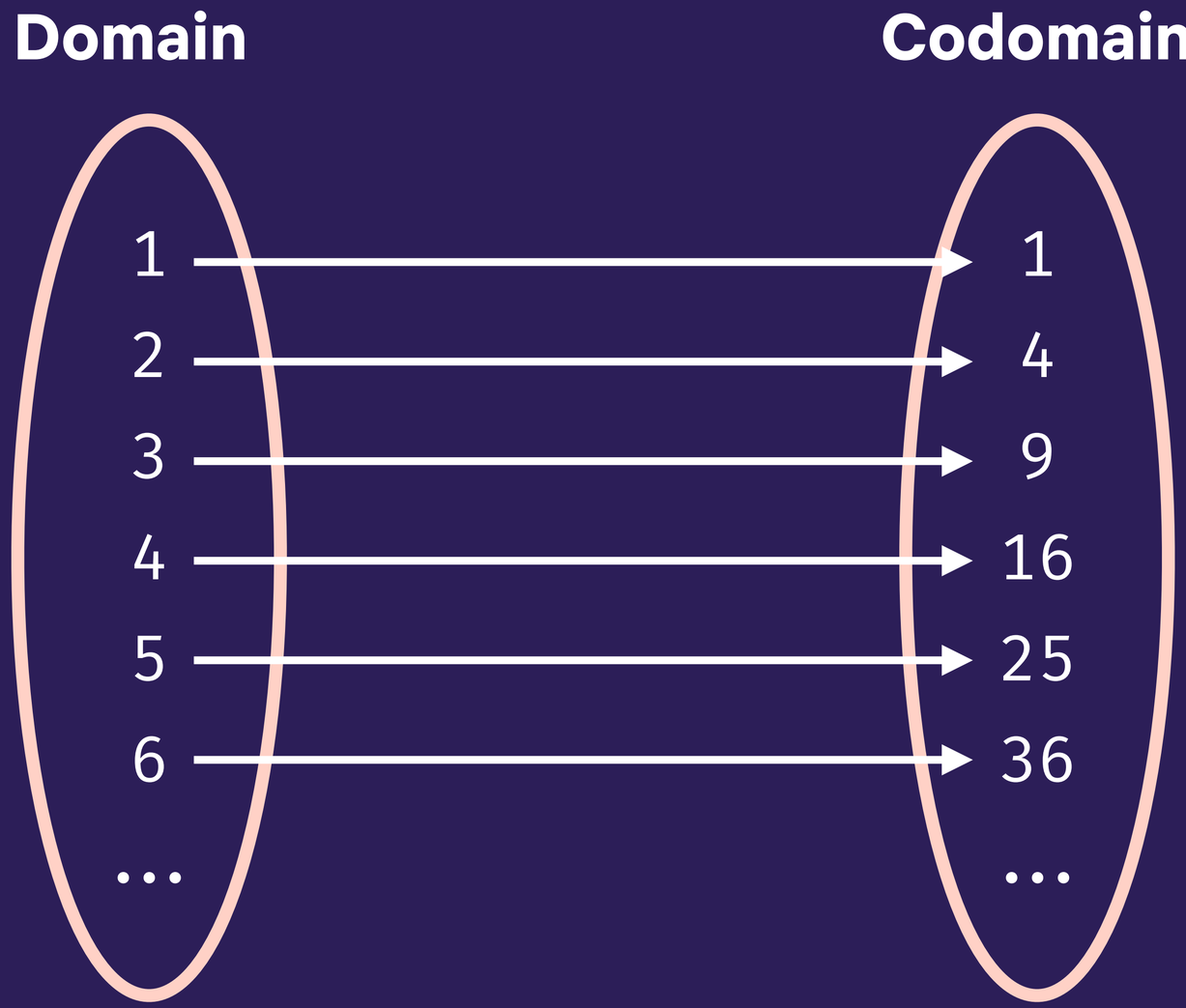
sugarpirate_ poteto



$$f(x) = x^2$$

sugarpirate_ poteto

$$f(x) = x^2$$



su… pirate_ poteto

Total vs Partial functions

sugarpirate_ poteto

	Partial	Total
Pure	Pure & Partial	Pure & Total
Impure	Impure & Partial	Impure & Total

sugarpirate_ poteto

Partial

A partial function is a function that is not defined for all possible input values.

sugarpirate_ poteto

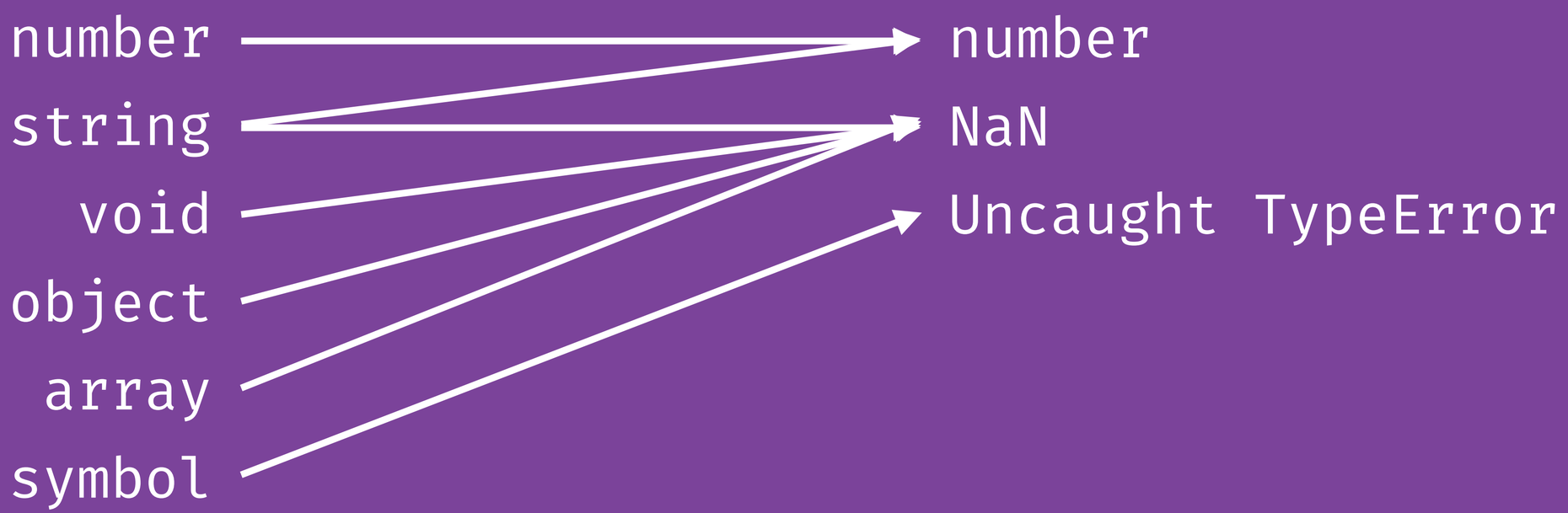
```
const half = x ⟹ x / 2;
```

sugarpirate_ poteto

```
const half = x ⇒ x / 2;
```

Possible Domains

Possible Codomains

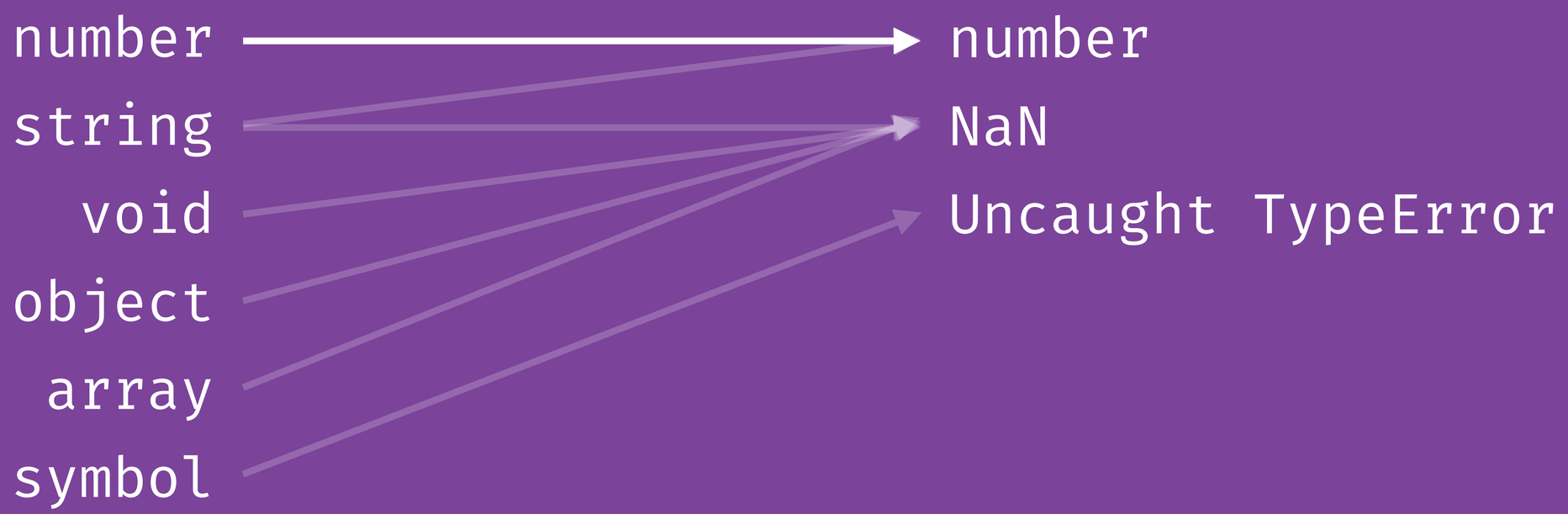


sugarpirate_ poteto

```
const half = x ⇒ x / 2;
```

Possible Domains

Possible Codomains

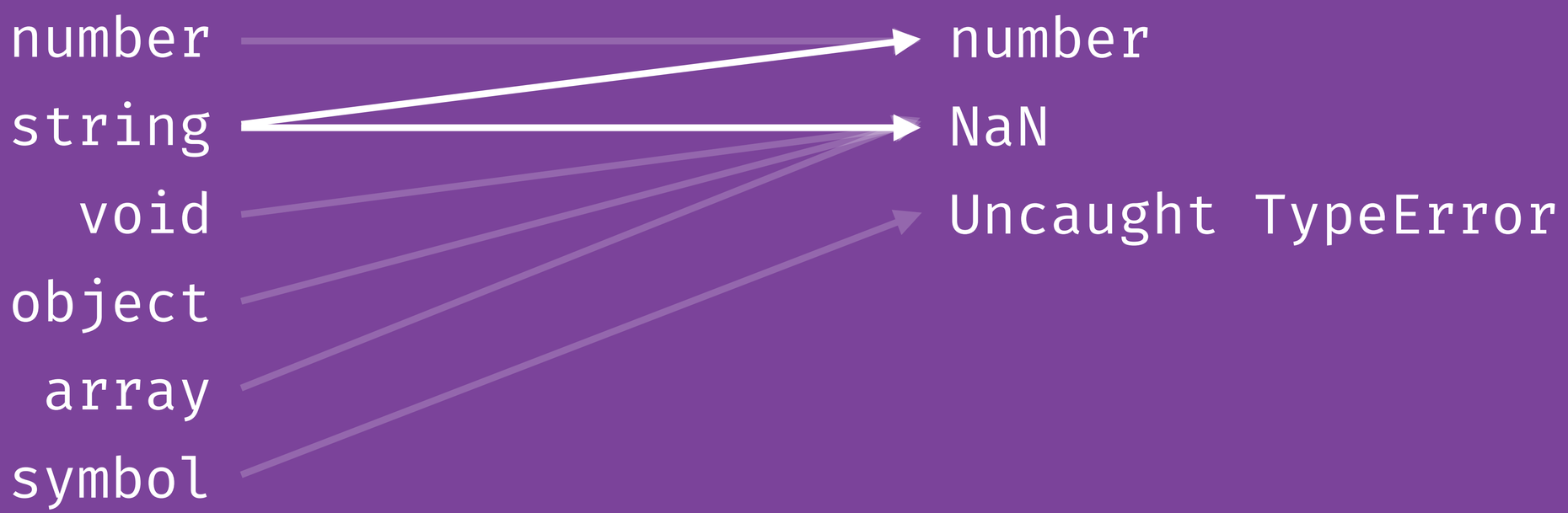


sugarpirate_ poteto

```
const half = x ⇒ x / 2;
```

Possible Domains

Possible Codomains



sugarpirate_ poteto

```
const half = x ⇒ x / 2;
```

Possible Domains

Possible Codomains

```
half('10') // 5
half('hello world') // NaN
```

void

object

array

symbol

Uncaught TypeError

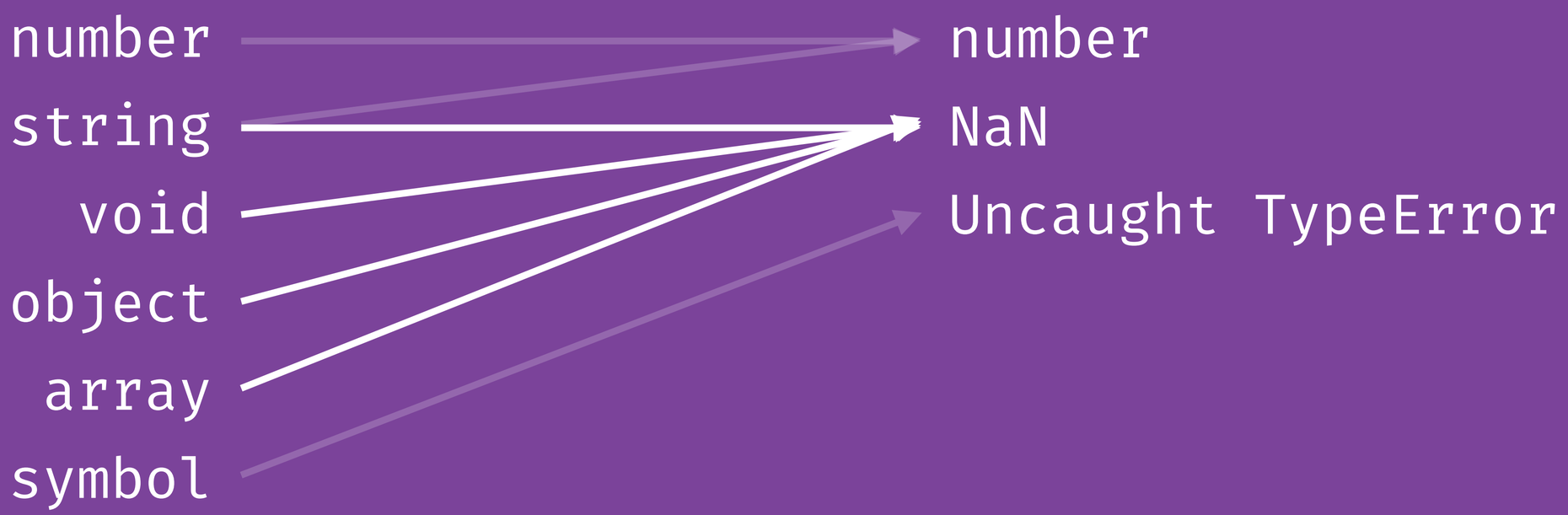
sugarpirate_ poteto

sugarpirate_ poteto

```
const half = x ⇒ x / 2;
```

Possible Domains

Possible Codomains

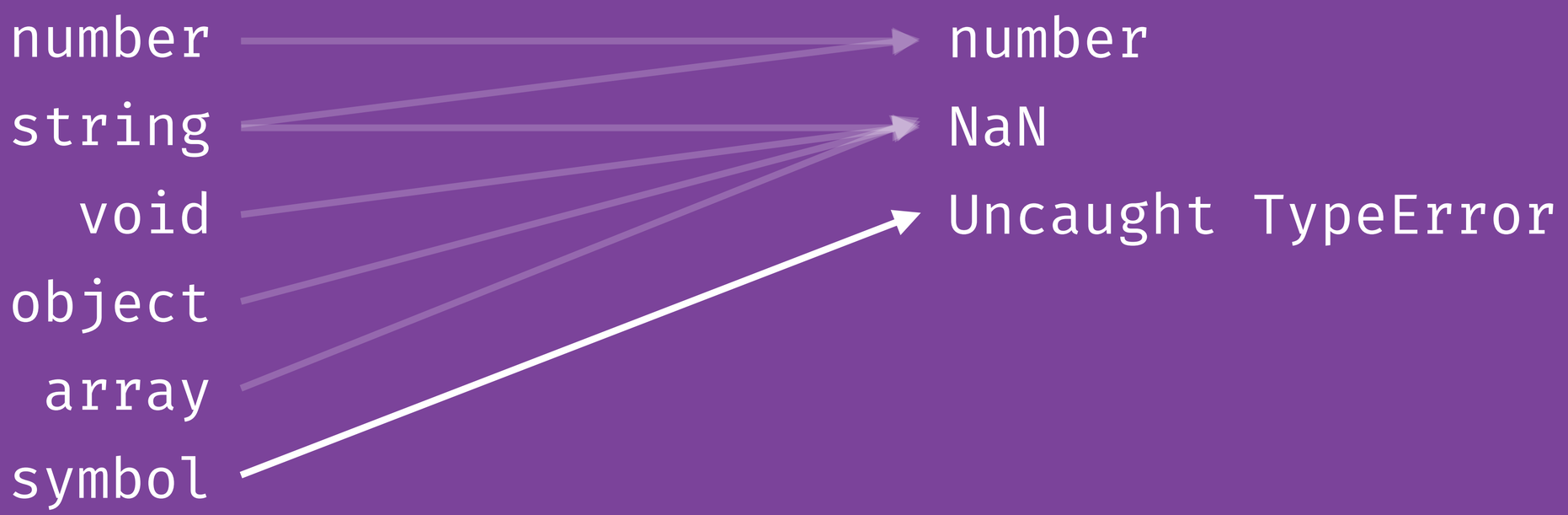


sugarpirate_ poteto

```
const half = x ⇒ x / 2;
```

Possible Domains

Possible Codomains



sugarpirate_ poteto

```
const half = (x: number) ⇒ x / 2;
```

Possible Domains

number
string
void
object
array
symbol

Possible Codomains

number
NaN
Uncaught TypeError

sugarpirate_ poteto

Total

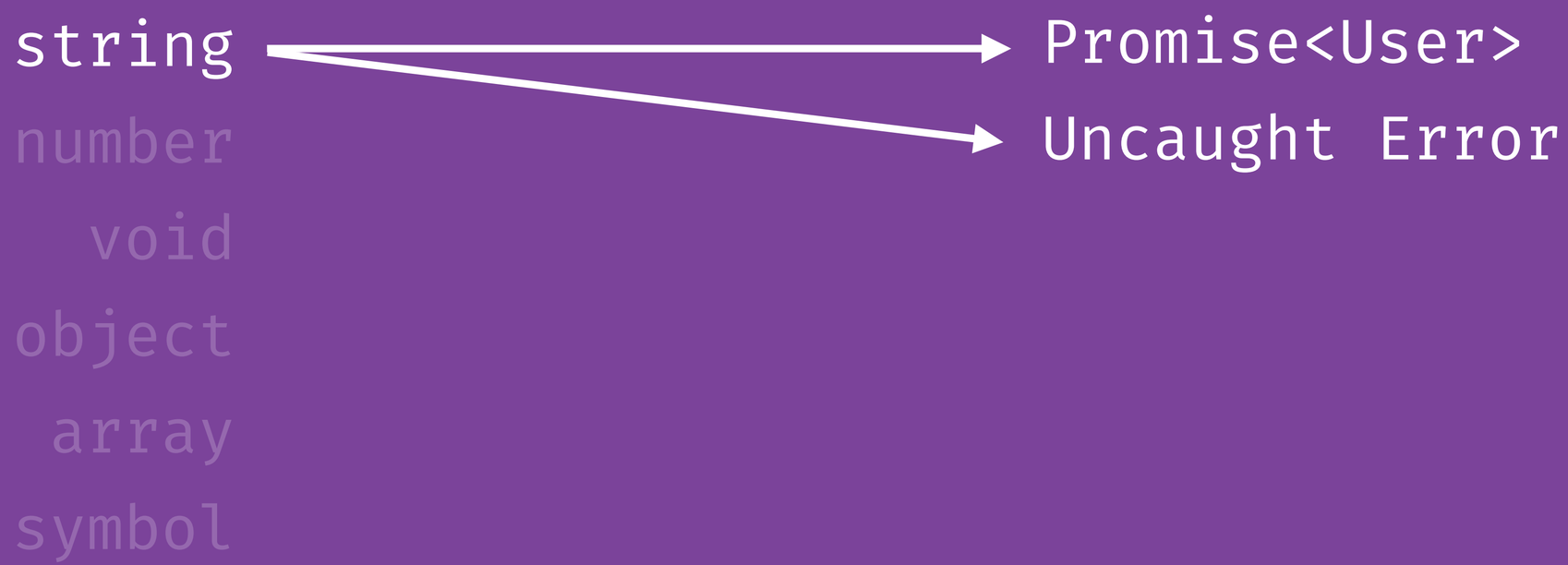
A total function is a function that is defined for all possible values of its input. That is, it terminates and returns a value.

sugarpirate_ poteto

```
function fetchUser(username: string): Promise<User>
```

Possible Domains

Possible Codomains

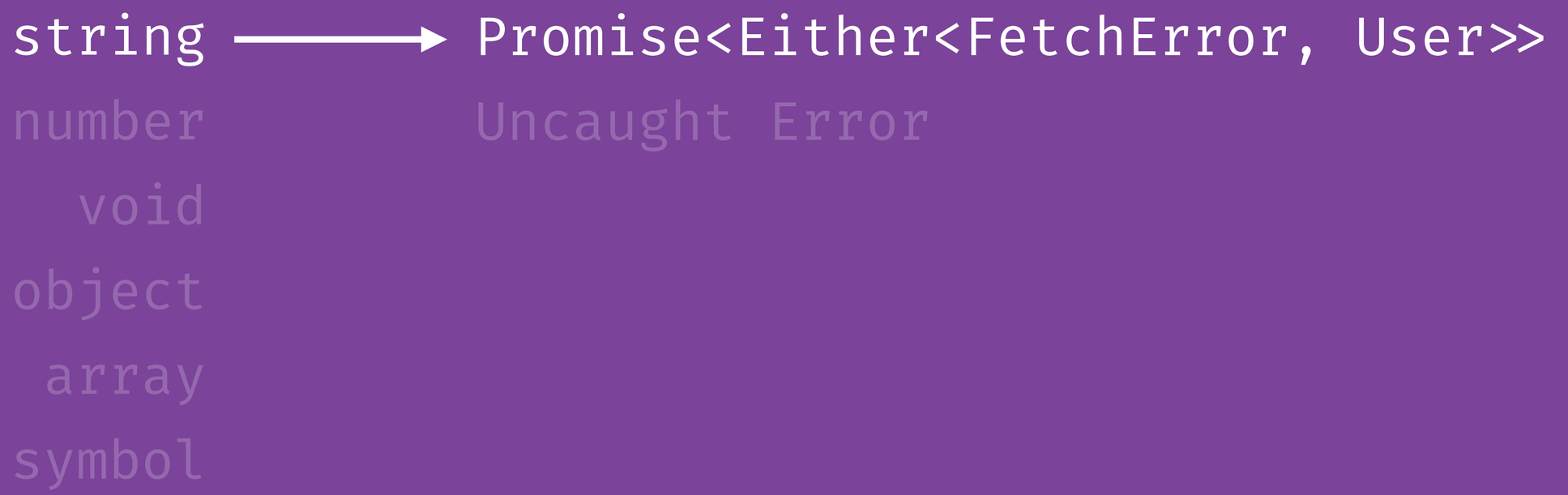


sugarpirate_ poteto

```
function fetchUser(username: string): Promise<Either<FetchError, User>>
```

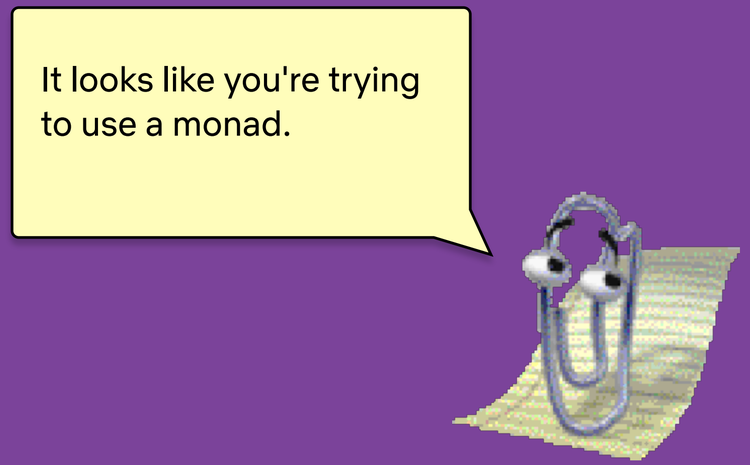
Possible Domains

Possible Codomains



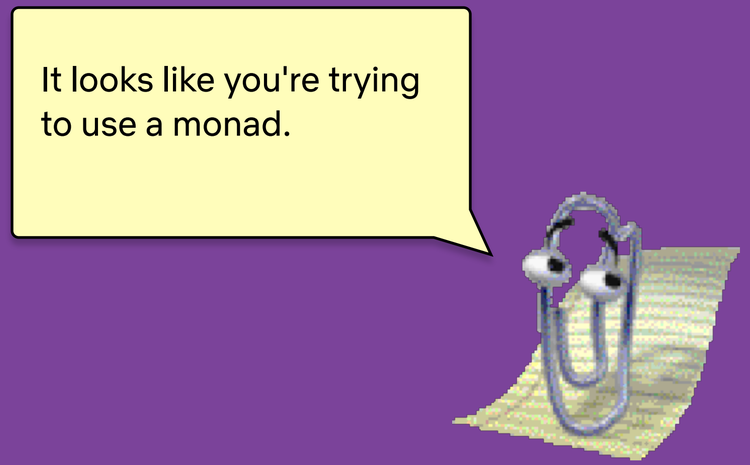
sugarpirate_ poteto

```
type Either<L, A> = Left<L, A> | Right<L, A>
```



sugarpirate_ poteto

```
type Either<L, A> = Left<L, A> | Right<L, A>
```



sugarpirate_ poteto

```ts
import { Either, left, right } from 'fp-ts/lib/Either';
import fetch from 'node-fetch';

async function fetchUser(username: string): Promise<Either<FetchError, User>> {
  const res = await fetch(`https://api.sugarpirate.com/users/${username}`);
  if (!res.ok) { return left(new FetchError(`[${res.status}] ${res.statusText}`)) }
  return right(await res.json());
}
```

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

sugarpirate_ poteto

```
import { Either, left, right } from 'fp-ts/lib/Either';
import fetch from 'node-fetch';

async function fetchUser(username: string): Promise<Either<FetchError, User>> {
  const res = await fetch(`https://api.sugarpirate.com/users/${username}`);
  if (!res.ok) { return left(new FetchError(`[${res.status}] ${res.statusText}`)) }
  return right(await res.json());
}
```

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

sugarpirate_ poteto

```ts
import { Either, left, right } from 'fp-ts/lib/Either';
import fetch from 'node-fetch';

async function fetchUser(username: string): Promise<Either<FetchError, User>> {
  const res = await fetch(`https://api.sugarpirate.com/users/${username}`);
  if (!res.ok) { return left(new FetchError(`[${res.status}] ${res.statusText}`)) }
  return right(await res.json());
}
```

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

sugarpirate_ poteto

```
import { Either, left, right } from 'fp-ts/lib/Either';
import fetch from 'node-fetch';

async function fetchUser(username: string): Promise<Either<FetchError, User>> {
  const res = await fetch(`https://api.sugarpirate.com/users/${username}`);
  if (!res.ok) { return left(new FetchError(`[${res.status}] ${res.statusText}`)) }
  return right(await res.json());
}
```

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

sugarpirate_ poteto

```
async function doIt() {
  const maybeLauren = await fetchUser('lauren');
  const maybeNoOne = await fetchUser('asdjasjdashjdkahjksd');
  maybeLauren
    .map(lauren ⇒ lauren.projects)
    .map(projects ⇒ console.log(projects.map(p ⇒ p.name)));
  maybeNoOne
    .map(noOne ⇒ noOne.projects)
    .map(projects ⇒ console.log(projects.map(p ⇒ p.name)));
}
```

sugarpirate_ poteto

```
async function doIt() {
  const maybeNoOne = await fetchUser('asdjasjdashjdkahjksd');
  maybeNoOne
    .mapLeft(e ⇒ console.log(e.message)); // e: FetchError
}
```

sugarpirate_ poteto

```
export function firstVisibleElement(
  selector: string,
  scrollableAreaSelector: string
): Element | undefined
```

Possible Domains

[string, string]

number

void

object

array

symbol

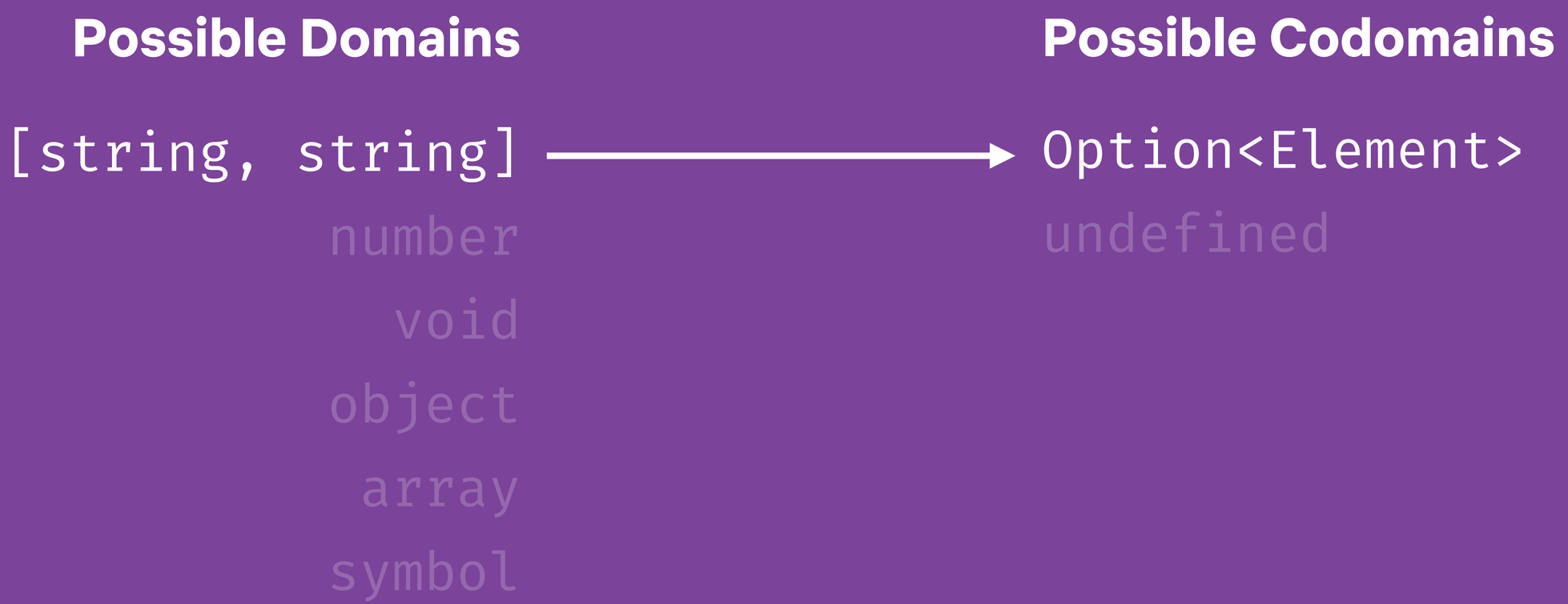
Possible Codomains

Element

undefined

sugarpirate_ poteto

```
export function firstVisibleElement(
  selector: string,
  scrollableAreaSelector: string
): Option<Element>
```



sugarpirate_ poteto

```
type Option<A> = None<A> | Some<A>
```

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

sugarpirate_ poteto

```
export function firstVisibleElement(
  selector: string,
  scrollableAreaSelector: string
): Option<Element> {
  const scrollableElement = document.querySelector(scrollableAreaSelector);
  if (!scrollableElement) return none;
  const scrollableBounds = scrollableElement.getBoundingClientRect();
  const firstVisibleItem = Array.from(document.querySelectorAll(selector)).find(
    el ⇒ {
      const elBounds = el.getBoundingClientRect();
      const isInViewport = detectInViewport(elBounds, scrollableBounds);
      return isInViewport && elBounds.top - scrollableBounds.top ⩽ 0;
    }
  );
  return firstVisibleItem ? some<Element>(firstVisibleItem) : none;
}
```

sugarpirate_ poteto

```
export function firstVisibleElement(
  selector: string,
  scrollableAreaSelector: string
): Option<Element> {
  const scrollableElement = document.querySelector(scrollableAreaSelector);
  if (!scrollableElement) return none;
  const scrollableBounds = scrollableElement.getBoundingClientRect();
  const firstVisibleItem = Array.from(document.querySelectorAll(selector)).find(
    el => {
      const elBounds = el.getBoundingClientRect();
      const isInViewport = detectInViewport(elBounds, scrollableBounds);
      return isInViewport && elBounds.top - scrollableBounds.top <= 0;
    }
  );
  return firstVisibleItem ? some<Element>(firstVisibleItem) : none;
}
```

sugarpirate_ poteto

```
export function firstVisibleElement(
  selector: string,
  scrollableAreaSelector: string
): Option<Element> {
  const scrollableElement = document.querySelector(scrollableAreaSelector);
  if (!scrollableElement) return none;
  const scrollableBounds = scrollableElement.getBoundingClientRect();
  const firstVisibleItem = Array.from(document.querySelectorAll(selector)).find(
    el => {
      const elBounds = el.getBoundingClientRect();
      const isInViewport = detectInViewport(elBounds, scrollableBounds);
      return isInViewport && elBounds.top - scrollableBounds.top <= 0;
    }
  );
  return firstVisibleItem ? some<Element>(firstVisibleItem) : none;
}
```

sugarpirate_ poteto

```
export function firstVisibleElement(
  selector: string,
  scrollableAreaSelector: string
): Option<Element> {
  const scrollableElement = document.querySelector(scrollableAreaSelector);
  if (!scrollableElement) return none;
  const scrollableBounds = scrollableElement.getBoundingClientRect();
  const firstVisibleItem = Array.from(document.querySelectorAll(selector)).find(
    el => {
      const elBounds = el.getBoundingClientRect();
      const isInViewport = detectInViewport(elBounds, scrollableBounds);
      return isInViewport && elBounds.top - scrollableBounds.top <= 0;
    }
  );
  return firstVisibleItem ? some<Element>(firstVisibleItem) : none;
}
```

sugarpirate_ poteto

```
firstVisibleElement('.item', '.item-container').map(el ⇒
  el.getAttribute('data-whatever') // string
);
```

sugarpirate_ poteto

```
(method) map<string>(f: (a: Element) ⇒ string):
Option<string>
```

Takes a function `f` and an `Option` of `A`. Maps `f` either on `None` or `Some`, Option's data constructors. If it maps on `Some` then it will apply the `f` on `Some`'s value, if it maps on `None` it will return `None`.

@example

```
assert.deepEqual(some(1).map(n ⇒ n * 2),
some(2))
```

sugarpirate_ poteto

Cardinality

cardinality · number of elements of the set

sugarpirate_ poteto

Lower cardinality = Less bugs*

sugarpirate_ poteto

Lower cardinality = Less bugs*

sugarpirate_ poteto

Pragmatic Set Theory

set · collection of objects

sugarpirate_ poteto

```
type Conferences = 'QConSF' | 'dotJS' | 'React Rally';
```

sugarpirate_ poteto

(not real syntax)

```
|Conferences| = 3
```

sugarpirate_ poteto

```
type Conferences = string;
```

sugarpirate_ poteto

(not real syntax)

```
|Conferences| = Infinity
```

sugarpirate_ poteto

Primitive types are not precise

sugarpirate_ poteto

(not real syntax)

```
|string|    = Infinity
|number|    = Infinity
|symbol|    = Infinity
|boolean|   = 2
|null|      = 1
|undefined| = 1
```

sugarpirate_ poteto

(not real syntax)

```
|object| = Infinity
```

sugarpirate_ poteto

Be precise

sugarpirate_ poteto

```
function toString<T>(x: T): string { return x.toString(); }
toString(undefined);
toString(null);
```

sugarpirate_ poteto

```
function toString<undefined>(x: undefined): string
```

```
function toString<T>(x: T): string { return x.toString(); }
toString(undefined);
toString(null);
```

```
function toString<null>(x: null): string
```

sugarpirate_ poteto

```
function toString<T>(x: T): string
```

Possible Domains

- string
- number
- object
- array
- symbol
- void

Possible Codomains

- string
- Uncaught TypeError

sugarpirate_ poteto

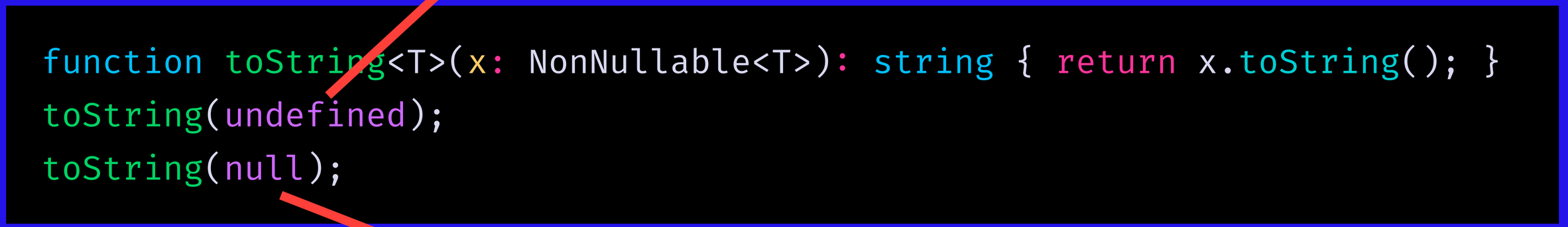
```
function toString<T>(x: NonNullable<T>): string { return x.toString(); }
toString(undefined);
toString(null);
```

sugarpirate_ poteto

```
function toString<T>(x: NonNullable<T>): string { return x.toString(); }
toString(undefined);
toString(null);
```

sugarpirate_ poteto

```
[ts] Argument of type 'undefined' is not
assignable to parameter of type '{}'.
```



```
function toString<T>(x: NonNullable<T>): string { return x.toString(); }
toString(undefined);
toString(null);
```

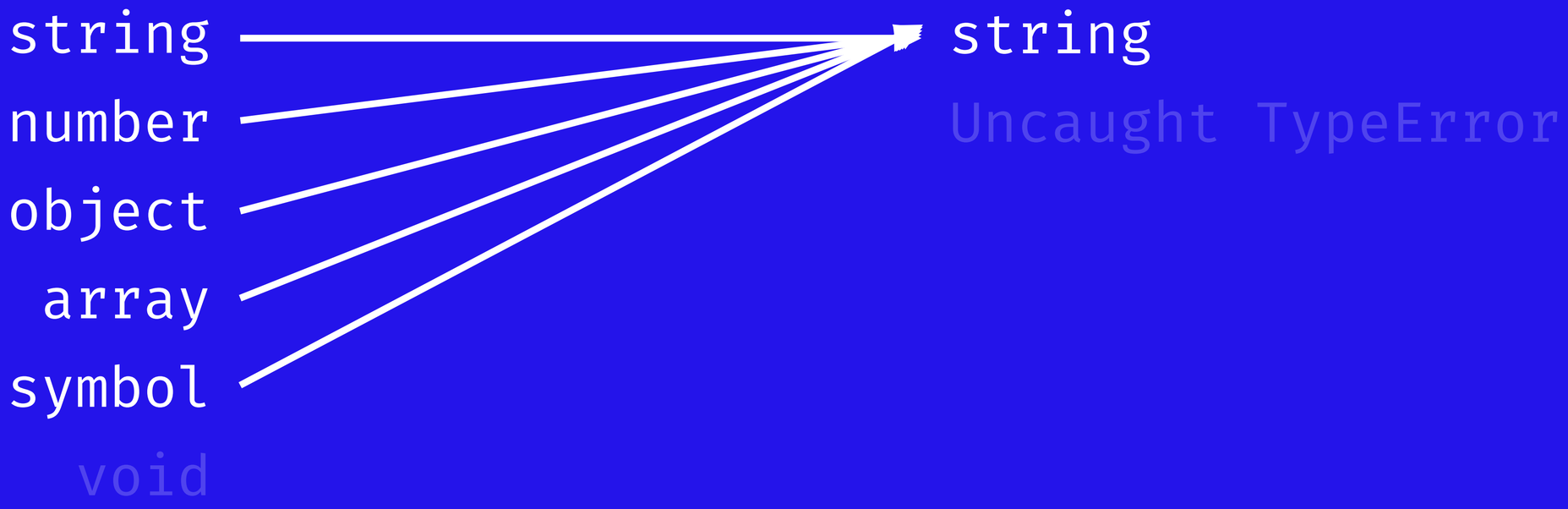
```
[ts] Argument of type 'null' is not
assignable to parameter of type '{}'.
```

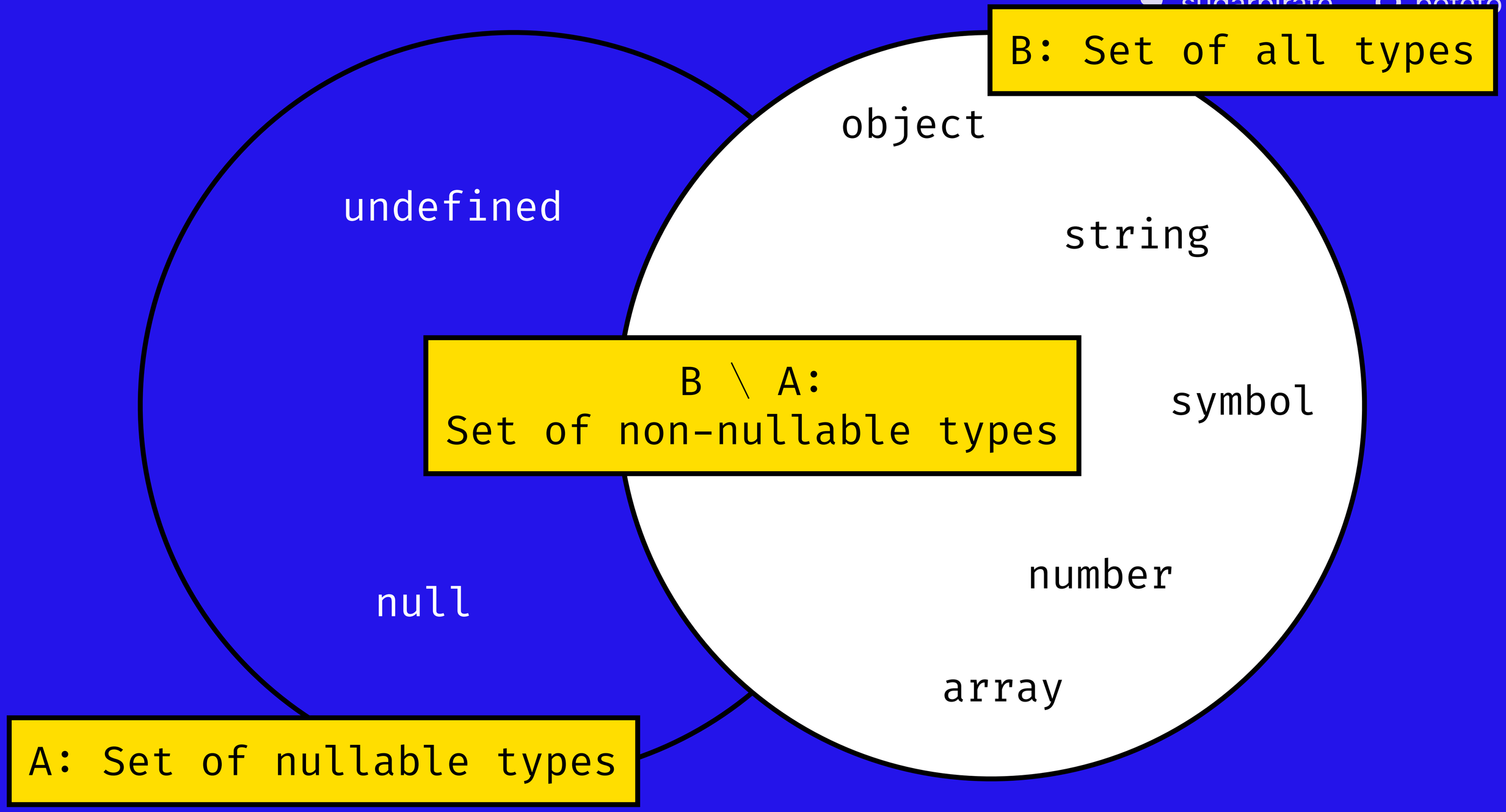
sugarpirate_ poteto

```
function toString<T>(x: NonNullable<T>): string
```

Possible Domains

Possible Codomains




B: Set of all types
undefined
object
string
B \ A:
Set of non-nullable types
symbol
number
null
array
A: Set of nullable types

sugarpirate_ poteto

```
type NonNullable<T> = T extends null | undefined ? never : T
type T34 = NonNullable<string | number | undefined>;  // string | number
type T35 = NonNullable<string | string[] | null | undefined>;  // string | string[]
```

sugarpirate_ poteto

```ts
type Partial<T> = { [P in keyof T]?: T[P]; };
type Required<T> = { [P in keyof T]-?: T[P]; };
type Readonly<T> = { readonly [P in keyof T]: T[P]; };
type Pick<T, K extends keyof T> = { [P in K]: T[P]; };
type Record<K extends keyof any, T> = { [P in K]: T; };
type Exclude<T, U> = T extends U ? never : T;
type Extract<T, U> = T extends U ? T : never;
type NonNullable<T> = T extends null | undefined ? never : T;
type ReturnType<T extends (...args: any[]) => any> = T extends (...args: any[]) => infer R ? R : any;
```

https://github.com/Microsoft/TypeScript/blob/v3.0.1/src/lib/es5.d.ts

sugar te_
eto

Be pragmatic

"No matter what language you work in, programming in a functional style provides benefits. You should do it whenever it is convenient, and you should think hard about the decision when it isn't convenient."

John Carmack


sugarpirate_
poteto
Merci beaucoup
Cinemagraph by /u/fezzo


sugarpirate_
poteto
Merci beaucoup
Cinemagraph by /u/fezzo