names don’t matter

why this is not as trivially false as it may seem

firstly…

this is not to say that naming things isn't useful, and that we should all go around calling each other whatever we feel like… that's what Twitter is for

in other words, Identifiers gotta Identify!

names don’t matter

you may have heard this uttered by some programmers, generally by people working on strongly typed functional libraries

these same people also often demand fidelity to difficult names from mathematics, such as **Functor**, **Monad** or **Isomorphism**

clearly they are hypocrites!

hypocrites!

if you demand that names *don't matter*, then what is your problem with changing the name to better suit the audience?

conversely, if you demand that names *do matter*, why are you so willing to change an existing name?

who is the hypocrite?

what’s in a name?

“what’s in a name? that which we call a rose
by any other name would smell as sweet.”

–Juliet Capulet. Balcony Scene, Romeo & Juliet

what is a name?

name
noun.

1. a word or a combination of words by which a person, place, or thing, a body or class, or any object of thought is designated, called, or known

3. an appellation, title, or epithet, applied descriptively, in honor, abuse, etc.

what is a name?

word
noun.

1. a unit of language, consisting of one or more spoken sounds or their written representation, that **functions as a principal carrier of meaning**.

Words are composed of one **or more morphemes** and are either the smallest units susceptible of independent use or consist of two or three such units combined under certain linking conditions, as with the loss of primary accent that distinguishes black·bird· from black· bird·. Words are usually separated by spaces in writing, and are distinguished phonologically, as by accent, in many languages.

a principal carrier of meaning

words carry meaning based on the shared understanding between two communicating parties

without shared understanding, the value of the use of the word is reduced

yet, the actual grouping of morphemes is arbitrary, and the *meaning* is built by usage – the *essence*, or the object, is not affected by the specific morphemes

without shared understanding, we have two options: use another word in the hope that it will convey our intended meaning, or teach the meaning of the original

a principal carrier of meaning

our usage of words is nevertheless imprecise

we use synonyms; multiple words that have the same or very similar meaning

we overload meanings, so the different words mean the same thing

we speak multiple languages, each with their own entire dictionaries, with different grammatical and morphological themes

each word is simply an *association* of the morphemes to the actual meaning

if a word is merely an association, can the actual meaning may be referenced in a more stable or canonical form?

why do names matter
to programmers?

“Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a *computer* what to do, let us concentrate rather on explaining to *human beings* what we want a computer to do.”

–Donald Knuth, Literate Programming

“There are 2 hard problems in computer science:
cache invalidation, naming things, and off-by-1 errors.”

–Leon Bambrick @secretGeek

many of the abstractions programmers have traditionally built have been only informally or poorly specified

lack of abstractive power of our languages and type-systems have made them insufficiently general and reusable

in attempting to convey meaning we have two options, *documentation* and *names*

we constantly search for better names to describe a particular unit, class, function or module – names that will better give the reader a sense of its utility, often leaning heavily on metaphor

in many programming contexts, names are all you've got

special AKA

in searching for the perfect name, programmers often reach for baroque synonyms

the assumption seems to be that programmers basically already know everything, and if only we can find the right metaphor to relate to their existing body of knowledge, then they'll understand the unit under consideration

this leads to every language, every library, providing different names for the same thing, **because names matter**!

>>=

`>>=` AKA ‘bind’ (Haskell)

`flatMap` (Scala)

`thenCompose`, `thenComposeAsync` (Java `CompletionStage`)

`SelectMany` (C# LiNQ)

`then` (Javascript Promises A+)

`smooshMap` (ECMAScript TC39 Proposal)

>>=

these are (almost) all *the same thing*.

how do we know? let's look at the type!

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

with the addition of some laws, we can recognise that all these different names actually refer to the same concept, the *sequential composition of two actions*

the type, and the laws, are the more important identifying feature!

fred

here is a function:

```
fred :: forall a. a -> a
```

if we believe the hypothesis that Fast & Loose Reasoning is Morally Correct* then we already know that this function *must* be the same as:

```
definitely_not_fred :: forall a. a -> a
```

aka…

* http://www.cse.chalmers.se/~nad/publications/danielsson-et-al-popl2006.html

upset

here is a function:

```
upset :: forall a. [a] -> [a]
```

this function's signature is not enough to tell us everything about its implementation, but we can still tell quite a lot about it. For instance, all elements in the result were present in the argument

if we add a law:

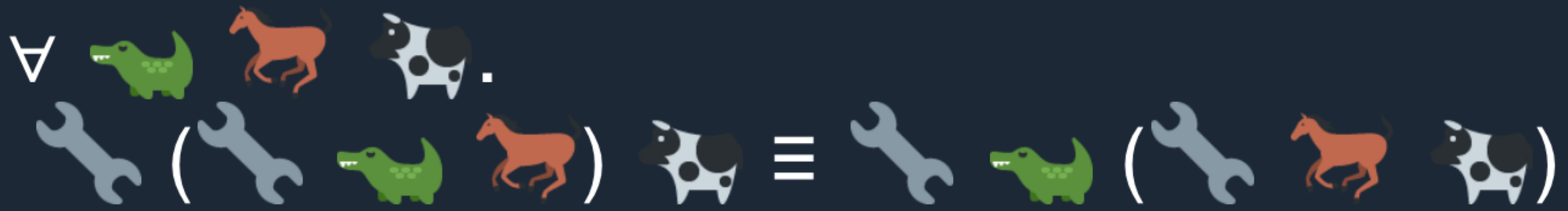
```
forall a, bs :: [a], cs :: [a]. upset (bs ++ cs) == upset cs ++ upset bs
```

we now know precisely what this function does

Brian McKenna
@puffnfresh

Following

such that



10:27 PM - 8 Apr 2018 from Deloraine, Tasmania

arguments

argument names are important if looking them up is difficult. If we have long methods (maybe in Java) and the declaration site and usage sites are far apart, maybe we need descriptive names

in monomorphic code, our arguments mean concrete things, longer names may have better ergonomics

in polymorphic code, arguments are often for all types, and thus lack any definition

```
fred :: a -> a
fred theArgumentToTheFunctionFred = theArgumentToTheFunctionFred
fred a = a
```

arguments

often the best argument name is no name at all

```
class Monad m where
  return :: a -> m a
  >>=    :: m a -> (a -> m b) -> m b
  fmap   :: m a -> (a -> b)   -> m b
```

we can implement `fmap` using `return` and `>>=`:

```
  fmap ma f = ma >>= (\a -> return (f a))
```

or using function composition with no explicit named argument at all:

```
  fmap ma f = ma >>= (return . f)
```

arguments

although, we can take this to unnecessary extremes:

```
fmap ma f = ma >>= (\a -> return (f a))

fmap ma f = ma >>= (return . f)

fmap = (. (return .)) . (>>=)
```

hard names

people often balk at difficult names, often ones from mathematics (please ask Dr Cheng why mathematicians are *so bad* at naming things)

but, using these names gives us access to much literature on their meaning, which is unavailable if we rename them

Ed Kmett tells how using pre-existing names has allowed others to show him where he is using them incorrectly, or how to add useful additional functionality implied by the mathematics

the Twitter “Bijection” library inspired much comment about its incorrect usage of the term, and that led to material improvements in the library for everyone’s benefit

what does *matters* mean?

if we accept the hypothesis that names matter, what is our definition of “matter”

I propose that something matters where if you change it, there is a significant difference

I can add multiple names to things without changing their essential meaning:

my Mum usually calls me a string of names of ex-husbands, boyfriends and dogs… I am habituated to answer regardless

if we change types, or laws, we fundamentally change their meaning and the behaviour of the program

what if names don’t matter?

does this mean we are free to rename things willy nilly? of course not!

“names don’t matter” means we should carefully choose the ergonomic factors we care about in the names we use

names that already have precise meaning should be preferred to new, made-up names and metaphorical allusions

names that already have precise meaning should be avoided for only tangentially related concepts (Isomorphic Javascript?)

stability of identifiers is important, tying multiple communities with the same concept

naming new concepts is hard! let’s go to the pub and argue about them

thanks!