The Not-A-Wat in Haskell

Brian Hurt

18 Oct 2015

<RANT>

Wat Definition

What is a "Wat"?

Wat Definition

https://www.destroyallsoftware.com/talks/wat

Wat Definition

My definition:
A "wat" (w.r.t. programming languages) is a behavior that is both unexpected, and inconsistent with all the other behaviors of the language.

The Wat In Question

```
Prelude> length (1, 2)
```

The Wat In Question

```
Prelude> length (1, 2)
1
Prelude>
```

Wat Definition

This is not a Wat!

This behavior is consitent with the rest of the language.

Working Through the Logic

Length gives the number of elements held by a container.

```
Prelude> length [ 1, 2, 3 ]
3
Prelude>
```

Length just folds over the container, counting the elements as they go by.

Working Through the Logic

Can you have a container that can *not* hold an arbitrary number of elements, but can only hold at most one?

Working Through the Logic

Maybe works as a "at most one" container:

```
Prelude> length Nothing
0
Prelude> length (Just "foo")
1
Prelude> fmap show Nothing
Nothing
Prelude> fmap show (Just 1)
Just "1"
```

Working Through the Logic

You can partially apply multi-param types just like functions.
So `Either String` works as a "at most one" container:

```
Prelude> foldr (+) 0 (Left "foo")
0
Prelude> foldr (+) 0 (Right 2)
2
Prelude> fmap show (Right 3)
Right "3"
Prelude>
```

Working Through the Logic

It is foldable, there for length is defined on it:

```
Prelude> :t length
length ::  Foldable t => t a -> Int
Prelude> length (Left "error message")
0
Prelude> length (Right 1)
1
Prelude>
```

Important Concept

Tuples

Are

Not

Lists

Important Concept

- `(,)` is a type, **just like Either**
- `(,)` takes two type parameters, **just like Either**
- `(,)` as a type can be partially applied, **just like Either**

So, if we apply `(,)` to one type (like we do with Either), what do we get?

Important Concept

We get a container that holds exactly one value, as the second value of the tuple.

Tuples are a Container

```
Prelude> foldr (+) 0 (1, 2)
2
Prelude> foldr (+) 0 ("foo", 2)
2
Prelude> fmap show (1, 2)
(1,"2")
Prelude> fmap show (False, 2)
(False,"2")
Prelude> length (False, 2)
1
Prelude>
```

Tuples are a Container

What value did you expect `length` to return?

nil?

NaN?

</RANT>