

COMMUNICATING WITH TYPES

KRIS JENKINS

@krisajenkins

DISCLAIMER



@krisajenkins

A STORY OF TYPES



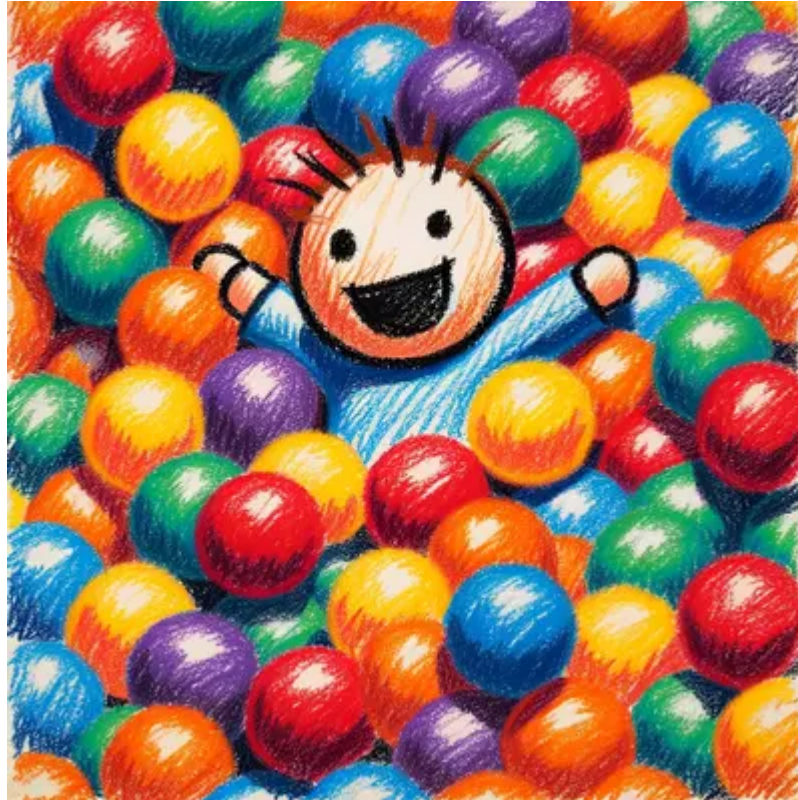
@krisajenkins

MOTIVATION

We're going to look at some ways types help you:

- ...communicate with your team.
- ...now, and in the future.
- ...analyze code from multiple perspectives.

DESCRIBING THINGS



How do we describe data?

@krisajenkins

THAT'S NOT MY DATA DESCRIPTION

```
<xs:element name="order">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="xs:string"></xs:element>
      <xs:element name="value" type="xs:int"></xs:element>
      <xs:element name="payment_method" type="xs:string">
        </xs:element>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

IS THAT MY DATA DESCRIPTION?

```
{  
  "id": "ORD00001",  
  "value": 315.0,  
  "payment_method": "Visa"  
}
```

THAT'S NOT MY DATA DESCRIPTION

```
public class Order {  
    private String id;  
    public String getId() {  
        return this.id;  
    }  
    public void setId(final String id) {  
        this.id = id;  
    }  
  
    private Int value;  
    ...  
}
```

ALGEBRAIC DATA TYPES

@krisajenkins

PRODUCTS - CHOOSE ALL

```
data Order = Order
  { orderId      :: String
  , value        :: Int
  , paymentMethod :: String
  }
```

SUMS - CHOOSE ONE

```
data ClothingSize  
  = Small  
  | Medium  
  | Large  
  | XLarge
```

FREE COMBINATION

```
data OrderResponse
  = PurchaseSuccessful
    { newOrder :: Order }
  | PaymentFailed
    { paymentProvider :: ProviderId
    , failureMessage  :: String
    }
  | NetworkError
    { statusCode :: Int
    , message    :: String
    }
```

REAL EXAMPLE

```
data Events
  = HeatmapItemsLoaded (Array HeatmapItem)
  ...
  | MapCameraMoved { lat :: Float, lng :: Float }
  | ToggleShowHeatmaps
  ...
  | BeginPayment
  | PaymentComplete Stripe.Token
  | PaymentError Stripe.Error
  | PaymentRecorded
```

- We sell heatmap data.

DESCRIBING RELATIONSHIPS



@krisajenkins

NOTATION

```
// JavaScript  
function withdraw(userId, amount) {...}
```

```
// Java  
public String withdraw(String userId, Int amount) {...}
```

...

```
String withdraw(String      , Int      )
```

```
withdraw(String      , Int      ) -> String
```

```
--| Haskell  
withdraw :: String -> Int -> String
```

FUNDAMENTAL RELATIONSHIP

```
withdraw :: String -> Int -> String
```



SOME EXAMPLES

in Elm

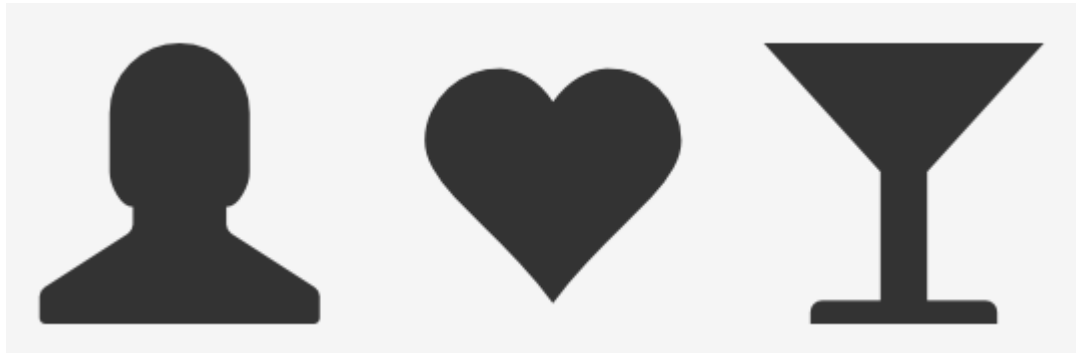


@krisajenkins

WE NEED TO GENERATE

```
<span class="glyphicon glyphicon-user"></span>  
<span class="glyphicon glyphicon-heart"></span>  
<span class="glyphicon glyphicon-glass"></span>
```

PRODUCES



@krisajenkins

REAL CODE

```
iconClass name =  
    "glyphicon glyphicon-" ++ name
```

@krisajenkins

THE TYPE SIGNATURE

```
iconClass : String -> String  
iconClass name =  
    "glyphicon glyphicon-" ++ name
```

WORLD'S VAGUEST SIGNATURE

```
foo :: String -> String
```

Is foo?

- iconClass?
- toUpper or toLower?
- reverse or abbreviate?
- chatGPT?
- turingMachine!

@krisajenkins

BROKEN

```
iconClass : String -> String
```

```
> iconClass "uesr"
```

```
--> "glyphicon glyphicon-uesr"
```

@krisajenkins

BROKEN (II)

```
iconClass : String -> String
```

```
> let user = iconClass "user"
```

```
> ...
```

```
> iconClass user
```

```
--> "glyphicon glyphicon-glyphicon glyphicon-user"
```

FIX THIS VAGUE SIGNATURE

```
iconClass : String -> String
```

with:

```
iconClass : Icon -> Html ()
```

IMPLEMENTATION

```
type Icon = User | Heart | Glass

showIcon : Icon -> Html ()
showIcon icon =
  let name = toLower (toString icon)
  in span
    [ class ("glyphicon glyphicon-" ++ name) ]
    []
```

INFORMATION & CLUES

The type signature:

- ...reveals the structure.
- ...flags up problems.
- ...structures a solution.
- ...gives clarity of scope.

ANOTHER REAL EXAMPLE

```
decodeUri uri =  
  case uri of  
    "/account" -> "dashboard"  
    "/account/settings" -> "settings"  
    "/" -> "frontpage"  
    _ -> "404"
```

LOOK AT THE SIGNATURE

```
decodeUri : String -> String
```

SHOULD BE

```
type View = Dashboard | Settings | Frontpage | NotFound  
decodeUri : String -> View
```

AFTER

```
type View = Dashboard | Settings | Frontpage | NotFound

decodeUri : String -> View
decodeUri uri =
  case uri of
    "/account" -> Dashboard
    "/account/settings" -> Settings
    "/" -> Frontpage
    _ -> NotFound
```

COMPILER HELP AS A BONUS

```
rootView : View -> Html msg
rootView view =
  case view of
    Dashboard -> h1 [] [ text "Dashboard..." ]
    Settings -> h1 [] [ text "Settings..." ]
    Frontpage -> h1 [] [ text "Welcome..." ]
```

COMPILER HELP AS A BONUS (II)

```
-- MISSING PATTERNS ----- ./src/View.elm
```

This `case` does not have branches for all possibilities.

```
91|>     case view of
92|>         Dashboard -> h1 [] [ text "Dashboard..." ]
93|>         Settings  -> h1 [] [ text "Settings..." ]
94|>         Frontpage -> h1 [] [ text "Welcome..." ]
... 
```

You need to account for the following values:

`View.NotFound`

Add a branch to cover this pattern!

DESCRIBING CONTEXT

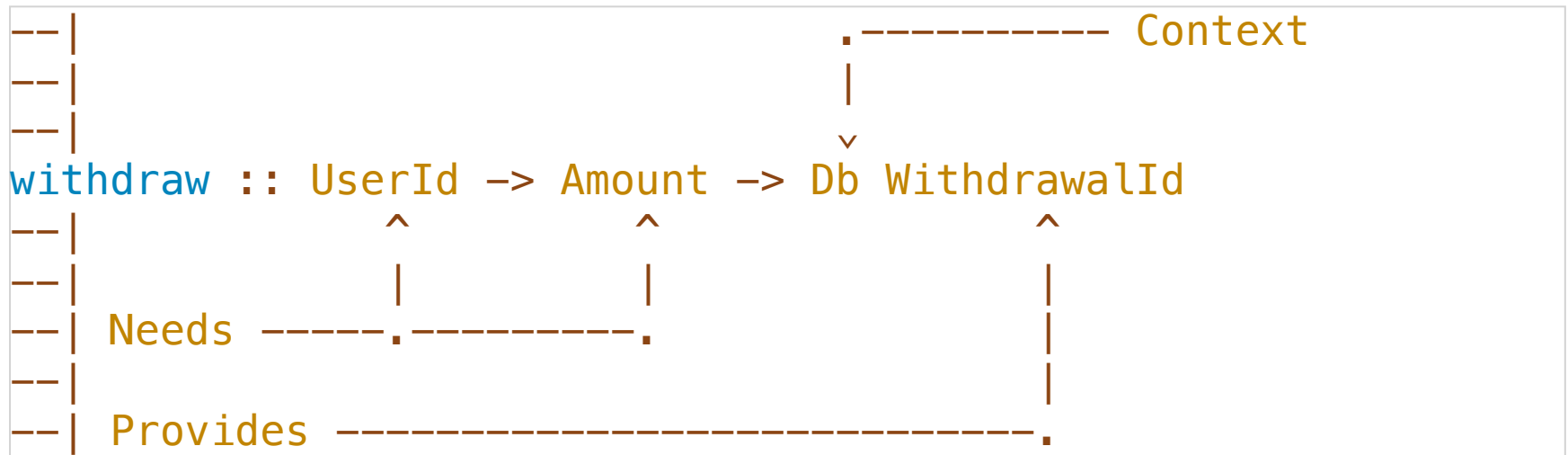


@krisajenkins

WHAT CONTEXT ARE WE IN?

```
showGreeting  :: User          -> String
fetchUser     :: UserId        -> User
createInvoice :: OrderId -> UserId -> Pdf
```

TALKING ABOUT CONTEXT



A FEW EXAMPLES

```
showGreeting  :: User          -> String
fetchUser     :: UserId        -> Http User
createInvoice :: OrderId -> UserId -> Db Pdf
```

- Reveals dependencies.
- Reveals absence of dependencies.
- The compiler checks it's true.
- When it's accurate, it's worth analysing.

RESTRICTING CONTEXT

```
createInvoice :: OrderId -> UserId -> Db Pdf
```

Better:

```
createInvoice :: OrderId -> UserId -> DbRead Pdf
```

- Tame complexity by reducing its footprint.

DECOUPLING CONTEXT

```
createInvoice :: OrderId -> UserId -> DbRead Pdf
```

Better still:

```
fetchInvoice :: OrderId -> UserId -> DbRead Invoice
```

```
createInvoice :: Invoice -> Pdf
```

- Spot opportunities to decouple.
- Make the hard bits more testable.
- Limit the scope of bugs.

A GOOD TYPE SYSTEM LETS YOU:

- Describe **things**.
- Describe **relationships**.
- Describe **context**.

WHERE NEXT?



@krisajenkins

PROPAGANDA!



- But Also TypeScript, Rust, Swift, Gleam
- Book: Type Driven Development with Idris
- Book: Thinking with Types

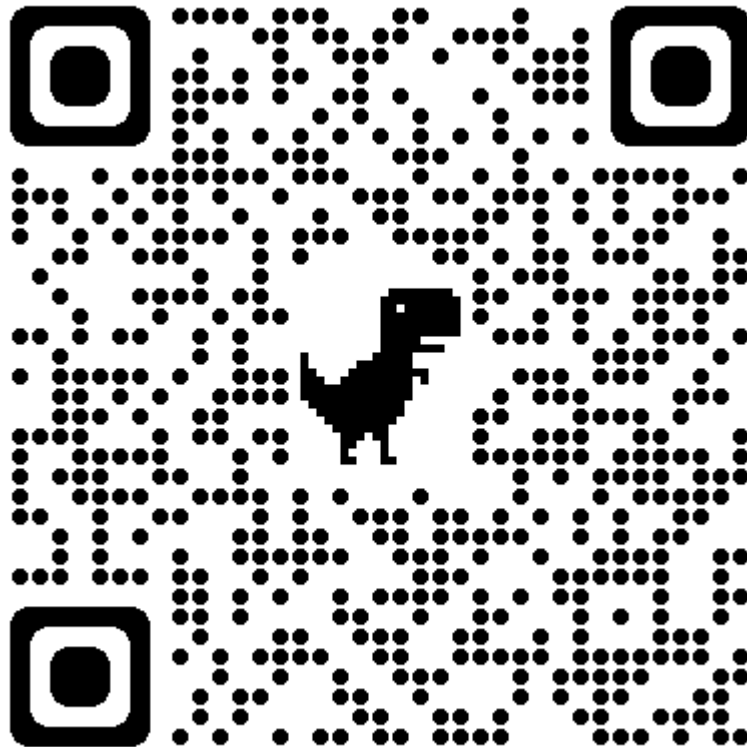
DEVELOPER VOICES



@krisajenkins

THANK YOU

linktr.ee/krisajenkins



@krisajenkins