round tripping balls (with partial isomorphisms & haskell)

For everyone's sake, please interrupt for clarification on any of these concepts during the talk (after they are introduced).

1. Printer. [*]

```
type Printer a = a -> Doc
```

2. Parser. [†]

```
newtype Parser a = Parser (String -> [(a, String)])
```

3. Partial isomorphisms[‡].

```
data Iso a b = Iso
    { apply   :: a -> Maybe b
    , unapply :: b -> Maybe a
    }
```

4. IsoFunctor: the functor[§] from Iso to Hask (restricted to f).

```
class IsoFunctor f where
  (<$>) :: Iso a b -> f a -> f b
```

5. ProductFunctor: a way to merge the output/input of two f's.

```
class ProductFunctor f where
  -- Left associative, applies before <$>
  infixr 6 <*>
  (<*>) :: f a -> f b -> f (a, b)
```

6. Alternative: try one failing that, the other.

```
class Alternative where
  (<|>) :: f a -> f a -> f a
```

7. Syntax: putting it all together.

```
class (IsoFunctor s, ProductFunctor s, Alternative s) => Syntax s where
  -- (<$>) :: Iso a b -> f a -> f b
  -- (<*>) :: f a -> f b -> f (a, b)
  -- (<|>) :: f a -> f a -> f a
  pure :: Eq a => a -> s a -- Eq for checking the value at runtime

class Syntax s => JsonSyntax s where
    -- Nest the first syntax within the second.
    runSub :: s v -> s Value -> s v

    -- We need a concrete way to access the underlying Aeson Value types in
    -- order to work with them.
    value :: s Value
```

[*] Doc is an abstract document representation.

[†] The list here allows non-determinism (backtracking). The string in the result is the unparsed remainder.

[‡] Isomorphism: A pair of functions $f : A \to B$ and $g : B \to A$ that are inverses such that:

1. $f \circ g \equiv id_B$
2. $g \circ f \equiv id_A$

[§] Functor: A principled way of taking a "thing" and "moving" it into a different context whilst preserving some notion of structure.

The canonical Haskell example is the Functor typeclass. Given two types ($\forall a, b \in Hask$) that have a function between them ($\exists f : a \to b$) we can produce a functored function ($\forall F \in Functor.\ F(f) : F(a) \to F(b)$).

More formally: a functor, F, is a transformation between categories $\mathcal{C}$ and $\mathcal{D}$ that maps morphisms and objects in $\mathcal{C}$ to morphisms and objects in $\mathcal{D}$ such that a few rules hold given objects $A, B \in \mathcal{C}$:

1. $F(f : A \to B) = F(f) : F(A) \to F(B)$
2. $F(id_A) = id_{F(A)}$
3. $F(f \circ g) = F(f) \circ F(g)$

Christian Marie <christian@ponies.io>

https://github.com/anchor/roundtrip-aeson/
https://hackage.haskell.org/package/roundtrip/
http://www.informatik.uni-marburg.de/ rendel/unparse/