Haskell Israel Meetup
30.8.2015

Haskell: From Theory to Enterprise

How Haskellers are seen

- With apologies to Chris Done
- http://chrisdone.com/posts/haskellers

How Haskellers are seen

How Haskellers are seen:

The type system and separated IO is an awkward, restricting space suit:

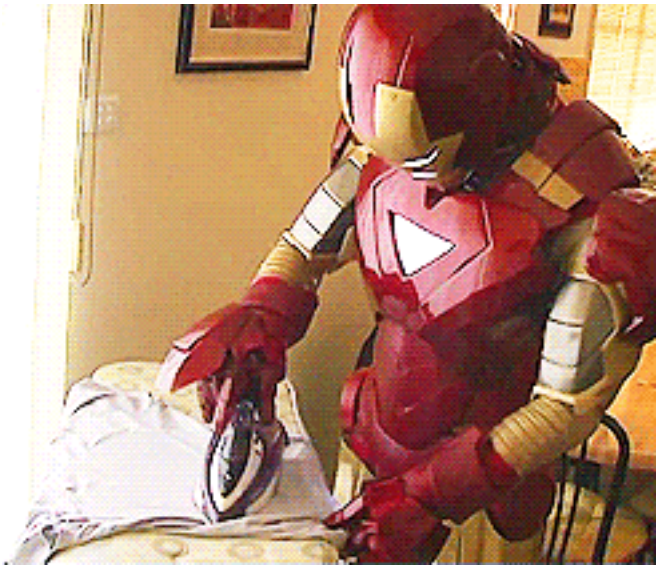
How Haskellers are seen

How Haskellers see themselves:

No, it's not a restrictive suit. It's a rocket suit!

How Haskellers are seen

Reality:



Theory

- Two approaches to the foundations of Computer Science:
  - Combinator Theory / Lambda Calculus
  - Finite autonoma / Turing machines
- Category Theory
- Curry – Howard isomorphism

From Theory to Practice

- Strong type system
- Laziness – purity, referential transparency
- Monads – helpful, but not necessary
- Useful structures
  - Functor
  - Applicative
  - Monoid / Semigroup
- Useful polymorphic types
  - Foldable
  - Traversable
  - Lens
- Dependent types

Practice

- Semantic preciseness
- High-level compile-time error checking
  - Quick development
  - Agile refactoring
  - Do it right – but quickly
- Security and privacy guarantees
- Simple concurrency
  - "Almost free" parallelism
  - STM
- Great unit tests

Trade-offs

- Performance
  - More optimization opportunities
  - Tweakable “almost” down to the metal, but
  - High level
  - Fundamentally different execution model
  - Case study: Facebook
- Memory
  - More sharing opportunities
  - Advanced garbage collection
  - Avoid generating what you don’t need / Fusion, but
  - Fundamentally different execution model

Example

```
data SuitehelpStage = Dita | TempXml | DitaOTXml |
                      XHtml | Suitehelp

data SuitehelpArgs = SuitehelpArgs
    { argInputType :: SuitehelpStage
    , argOutputType :: SuitehelpStage
    -- ...
    }
```

```
$ suitehelp --input-type=dita --output-type=suitehelp ...
```

Example

```
suitehelpMainWithConf :: SuitehelpArgs -> Conf -> IO ()
suitehelpMainWithConf args conf = do
    dita <- maybeInput Dita readDita Nothing
    maybeInput Dita (writeStatics dita) Nothing
    maybeInput Dita (createOutdirs dita) Nothing
    maybeOutput DitaOTXml writeDitaOTXml dita
    maybeOutput TempXml writeTempXml dita
    dita' <- maybeInput TempXml readTempXml dita
    html <- maybeDo createHtml dita'
    html' <- maybeInput XHtml readHtml html
    maybeOutput XHtml writeHtml html'
    maybeOutput Suitehelp writeSuitehelp html'
    error "suitehelp: no data available for output"
```

Example

```
suitehelpMainWithConf :: SuitehelpArgs -> Conf -> IO ()
suitehelpMainWithConf args conf = -- …
where
    itype = argInputType args
    otype = argOutputType args
    maybeInput typ action x
      | typ == itype = Just <$> action args conf
      | otherwise    = return x
    maybeOutput typ action x
      | typ == otype = maybeDoAndExit action x
      | otherwise    = return ()
    maybeDo action = maybe (return Nothing) $ fmap Just . action args conf
    maybeDoAndExit = maybe (return ()) . doAndExit
    doAndExit action x = do
      action args conf x
      exitSuccess
```