

*nix-info*<sup>1</sup>

*Eric Bailey*

*March 29, 2017*

<sup>1</sup> a brew info clone for Nix.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special contents, but the length of words should match the language.

THE MOTIVATION FOR USING HASKELL to write nix-info is its strong, static typing, etc, etc...

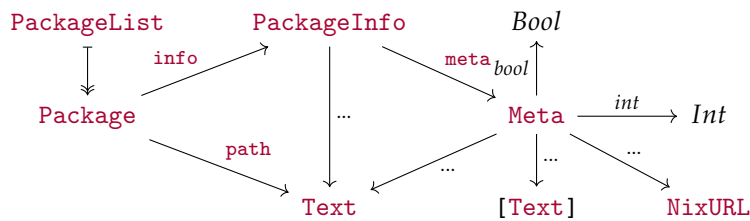
fixme: obviously

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift - not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special contents, but the length of words should match the language.

## Data Types

$\langle \text{Data Types } 1 \rangle \equiv$	(9 16)	
$\langle \text{Meta } 2 \rangle$		<b>Meta</b> “standard meta-attributes” [Con17]
$\langle \text{PackageInfo } 3 \rangle$		<b>PackageInfo</b> name, system and meta
$\langle \text{Package } 4 \rangle$		<b>Package</b> path and info
$\langle \text{PackageList } 5 \rangle$		<b>PackageList</b> [Package]
$\langle \text{NixURL } 6 \rangle$		<b>NixURL</b> URL

The standard meta-attributes are documented in the Nixpkgs Contributors Guide [Con17]. `nix-env`, which is called by `nix-info` in  $\langle \text{nixQuery } 12 \rangle$ , returns a nested **Object**, with relationships as described by the following diagram.



Flesh this out.

use better types than just **Text** everywhere ...

$\langle \text{Meta } 2 \rangle \equiv$  (1)

```

data Meta = Meta
  { description      :: Maybe Text
  , longDescription  :: Maybe Text
  , branch           :: Maybe Text
  , homepage         :: Maybe NixURL
  , downloadPage     :: Maybe NixURL
  , maintainers      :: Maybe [Text]
  , priority         :: Maybe Int
  , platforms        :: Maybe [Text]
  , hydraPlatforms   :: Maybe [Text]
  , broken           :: Maybe Bool
  , updateWalker     :: Maybe Bool
  , outputsToInstall :: Maybe [Text]
  , position         :: Maybe Text
  }
deriving (Show)
  
```

describe this

```

3  <PackageInfo 3>≡ (1)
    data PackageInfo = PackageInfo
      { name    :: Text
      , system  :: Text
      , meta    :: Meta
      }
    deriving (Show)

```

describe this

```

4  <Package 4>≡ (1)
    data Package = Package
      { path :: Text
      , info :: PackageInfo
      }
    deriving (Show)

```

This **newtype** is a cheap trick to avoid using **FlexibleInstances** for the automagically derived *<FromJSON Instances 8>*.

describe why

```

5  <PackageList 5>≡ (1)
    newtype PackageList = PackageList [Package]

```

Mention the avoidance of the orphan instance.

```

6  <NixURL 6>≡ (1)
    newtype NixURL = NixURL URL deriving (Show)

```

describe this

```

7  <magically derive ToJSON and FromJSON instances 7>≡ (8)
    $(deriveJSON defaultOptions "Meta")

    $(deriveJSON defaultOptions "PackageInfo")

```

describe this

```

8  <FromJSON Instances 8>≡ (9 16)
    <magically derive ToJSON and FromJSON instances 7>

    instance FromJSON PackageList where
      parseJSON (Object v) =
        PackageList <$> traverse \(p,y) -> Package p <$> parseJSON y (HM.toList v)
      parseJSON _          = fail "non-object"

    instance FromJSON NixURL where
      parseJSON (String t) = case importURL (T.unpack t) of
        Just url -> pure $ NixURL url
        Nothing  -> fail "no parse"
      parseJSON _      = fail "non-string"

    instance ToJSON NixURL where
      toJSON (NixURL url)    = String (T.pack (exportURL url))
      toEncoding (NixURL url) = text (T.pack (exportURL url))

```

```

⟨src/NixInfo/Types.hs 9⟩≡
- |
- Module      : NixInfo.Types
- Copyright   : (c) 2017, Eric Bailey
- License     : BSD-style (see LICENSE)
-
- Maintainer  : eric@ericb.me
- Stability   : experimental
- Portability : portable
-
- Data types and JSON parsers for nix-info

```

⟨OverloadedStrings 18⟩

⟨TemplateHaskell 19⟩

module NixInfo.Types where

⟨NixInfo.Types Imports 24⟩

⟨Data Types 1⟩

⟨FromJSON Instances 8⟩

## Helper Functions

```

⟨printPackage 10⟩≡
- printPackage :: MonadIO io => Package -> io ()
printPackage :: Package -> IO ()
printPackage (Package pkgPath (PackageInfo pkgName _pkgSystem pkgMeta)) =
  traverse_ putStrLn $
  catMaybes
  [ Just pkgName
  - , Just pkgSystem
  , description pkgMeta
  , T.pack . exportURL . (\(NixURL url) -> url) <$> homepage pkgMeta
  - , T.unwords . T.words <$> longDescription pkgMeta
  , T.unwords <$> maintainers pkgMeta
  - , T.unwords <$> outputsToInstall pkgMeta
  - , T.unwords <$> platforms pkgMeta
  , Just pkgPath
  , position pkgMeta
  ]

```

(11 16)

```

11  <src/NixInfo.hs 11>≡
    - |
    - Module      : NixInfo
    - Copyright   : (c) 2017, Eric Bailey
    - License     : BSD-style (see LICENSE)
    -
    - Maintainer  : eric@ericb.me
    - Stability   : experimental
    - Portability : portable
    -
    - brew info clone for Nix

module NixInfo (printPackage) where

import           NixInfo.Types

<hide Prelude.putStrLn 20>

<import traverse_, catMaybes 26>

import qualified Data.Text           as T
<import Data.Text.IO 25>

import           Network.URL         (exportURL)

<printPackage 10>

```

## Main Executable

```

12  <nixQuery 12>≡ (14 16)
    nixQuery :: Text -> Shell (Maybe PackageList)
    nixQuery arg =
        procStrict "nix-env" ["-qa", arg, "-json" ] empty >= \case
            (ExitSuccess,txt) -> pure $ decode (cs txt)
            (status,_)        -> exit status

13  <main 13>≡ (14 16)
    main :: IO ()
    main =
        sh $ arguments >= \case
            [arg] -> nixQuery arg >= \case
                Just (PackageList pkgs) -> liftIO $ traverse_ printPackage pkgs
                Nothing                  -> exit $ ExitFailure 1
            _      -> do echo "TODO: usage"
                exit $ ExitFailure 1

```

*<app/Main.hs 14>*≡

```
- |
- Module      : Main
- Copyright   : (c) 2017, Eric Bailey
- License     : BSD-style (see LICENSE)
-
- Maintainer  : eric@ericb.me
- Stability   : experimental
- Portability : portable
-
- Main executable for nix-info.
```

*<LambdaCase 17>*

*<OverloadedStrings 18>*

module Main (main) where

```
import      NixInfo                (printPackage)
import      NixInfo.Types
```

*<import Data.Aeson 21>*

```
import      Data.Foldable          (traverse_)
import      Data.String.Conversions (cs)
import      Data.Text              (Text)
```

*<import Turtle 27>*

*<nixQuery 12>*

*<main 13>*

## *As a Script*

*<shebang 15>*≡

(16)

```
#!/usr/bin/env nix-shell
#! nix-shell -i runhaskell -p "haskellPackages.ghcWithPackages (h: [ h.turtle h.aeson h.string-conversions h.url ])"
```

```

16  <script/nix-info 16>≡
    <shebang 15>

    <LambdaCase 17>

    <OverloadedStrings 18>
    <TemplateHaskell 19>

    module Main (main) where

    <hide Prelude.putStrLn 20>

    <NixInfo.Types Imports 24>

    <import traverse_, catMaybes 26>
    import           Data.String.Conversions (cs)

    <import Data.Text.IO 25>

    <import Turtle 27>

    <Data Types 1>

    <FromJSON Instances 8>

    <printPackage 10>

    <nixQuery 12>

    <main 13>

```

## Language Extensions

For brevity:

```

17  <LambdaCase 17>≡ (14 16)
    {-# LANGUAGE LambdaCase #-}

```

To manage juggling **Text**, (lazy) **ByteString**, and **Line** values, use the *<OverloadedStrings 18>* language extension [Cha14].

```

18  <OverloadedStrings 18>≡ (9 14 16)
    {-# LANGUAGE OverloadedStrings #-}

```

Enable the *<TemplateHaskell 19>* language extension [Wes14] to *<magically derive ToJSON and FromJSON instances 7>* from record definitions via **Data.Aeson.TH**

```

19  <TemplateHaskell 19>≡ (9 16)
    {-# LANGUAGE TemplateHaskell #-}

```

## Imports

Hide `Prelude.putStrLn`, so we can `<import Data.Text.IO 25> (putStrLn)`.

```

<hide Prelude.putStrLn 20>≡ (11 16)
import Prelude hiding (putStrLn)

<import Data.Aeson 21>≡ (14 24)
import Data.Aeson

<import Data.Aeson.Encoding 22>≡ (24)
import Data.Aeson.Encoding (text)

<import Data.Aeson.TH 23>≡ (24)
import Data.Aeson.TH (defaultOptions, deriveJSON)

<NixInfo.Types Imports 24>≡ (9 16)
<import Data.Aeson 21>
<import Data.Aeson.Encoding 22>
<import Data.Aeson.TH 23>

import qualified Data.HashMap.Lazy as HM
import Data.Text (Text)
import qualified Data.Text as T

import Network.URL (URL, exportURL, importURL)

<import Data.Text.IO 25>≡ (11 16)
import Data.Text.IO (putStrLn)

<import traverse_, catMaybes 26>≡ (11 16)
import Data.Foldable (traverse_)
import Data.Maybe (catMaybes)

<import Turtle 27>≡ (14 16)
import Turtle (ExitCode (..), Shell, arguments,
echo, empty, exit, liftIO,
procStrict, sh)

```



## Package Setup

```

28 <package.yaml 28>≡
    name: nix-info
    version: '0.1.0.0'
    synopsis: brew info clone for Nix
    description: See README at <https://github.com/nix-hackers/nix-info#readme>
    category: Development
    stability: experimental
    homepage: https://github.com/nix-hackers/nix-info
    github: nix-hackers/nix-info
    author: Eric Bailey
    maintainer: eric@ericb.me
    license: BSD3
    extra-source-files:
      - ChangeLog.md

    ghc-options: -Wall

    dependencies:
      - base >=4.9 && <4.10
      - aeson >=1.0 && <1.2
      - string-conversions >=0.4 && <0.5
      - text >=1.2 && <1.3
      - turtle >=1.3 && <1.4
      - unordered-containers >=0.2 && <0.3
      - url >=2.1 && <2.2

    library:
      source-dirs: src
      exposed-modules:
        - NixInfo
        - NixInfo.Types

    executables:
      nix-info:
        main: Main.hs
        source-dirs: app
        dependencies:
          - nix-info
29 <Setup.hs 29>≡
    import Distribution.Simple

    main :: IO ()
    main = defaultMain

```

## Chunks

<app/Main.hs [14](#)>  
 <Data Types [1](#)>  
 <FromJSON Instances [8](#)>  
 <hide Prelude.putStrLn [20](#)>  
 <import Data.Aeson [21](#)>  
 <import Data.Aeson.Encoding [22](#)>  
 <import Data.Aeson.TH [23](#)>  
 <import Data.Text.IO [25](#)>  
 <import traverse\_, catMaybes [26](#)>  
 <import Turtle [27](#)>  
 <LambdaCase [17](#)>  
 <magically derive ToJSON and FromJSON instances [7](#)>  
 <main [13](#)>  
 <Meta [2](#)>  
 <NixInfo.Types Imports [24](#)>  
 <nixQuery [12](#)>  
 <NixURL [6](#)>  
 <OverloadedStrings [18](#)>  
 <Package [4](#)>  
 <package.yaml [28](#)>  
 <PackageInfo [3](#)>  
 <PackageList [5](#)>  
 <printPackage [10](#)>  
 <script/nix-info [16](#)>  
 <Setup.hs [29](#)>  
 <shebang [15](#)>  
 <src/NixInfo.hs [11](#)>  
 <src/NixInfo/Types.hs [9](#)>  
 <TemplateHaskell [19](#)>

## Index

arguments: [13](#), [27](#)  
 branch: [2](#)  
 broken: [2](#)  
 catMaybes: [10](#), [26](#)  
 cs: [12](#), [14](#), [16](#)  
 Data.Aeson: [21](#), [22](#), [23](#)  
 Data.Aeson.Encoding: [22](#)  
 Data.Aeson.TH: [23](#)  
 Data.Foldable: [14](#), [26](#)  
 Data.HashMap.Lazy: [24](#)

Data.Maybe: [26](#)  
Data.Text: [11](#), [14](#), [24](#), [25](#)  
Data.Text.IO: [25](#)  
defaultOptions: [7](#), [23](#)  
deriveJSON: [7](#), [23](#)  
description: [2](#), [10](#), [28](#)  
downloadPage: [2](#)  
echo: [13](#), [27](#)  
empty: [12](#), [27](#)  
exit: [12](#), [13](#), [27](#)  
ExitCode: [27](#)  
exportURL: [8](#), [10](#), [11](#), [24](#)  
FromJSON: [8](#), [9](#)  
HM: [8](#), [24](#)  
homepage: [2](#), [10](#), [28](#)  
hydraPlatforms: [2](#)  
importURL: [8](#), [24](#)  
info: [4](#), [9](#), [11](#), [14](#), [28](#)  
liftIO: [13](#), [27](#)  
longDescription: [2](#), [10](#)  
Main: [14](#), [16](#), [28](#)  
main: [13](#), [14](#), [16](#), [28](#), [29](#)  
maintainers: [2](#), [10](#)  
Meta: [2](#), [3](#)  
meta: [3](#)  
name: [3](#), [28](#)  
Network.URL: [11](#), [24](#)  
NixInfo: [9](#), [11](#), [14](#), [28](#)  
NixInfo.Types: [9](#), [11](#), [14](#), [28](#)  
nixQuery: [12](#), [13](#)  
NixURL: [2](#), [6](#), [8](#), [10](#)  
outputsToInstall: [2](#), [10](#)  
Package: [4](#), [5](#), [8](#), [10](#)  
PackageInfo: [3](#), [4](#), [10](#)  
PackageList: [5](#), [8](#), [12](#), [13](#)  
path: [4](#)  
platforms: [2](#), [10](#)  
position: [2](#), [10](#)  
priority: [2](#)  
procStrict: [12](#), [27](#)  
putStrLn: [10](#), [20](#), [25](#)  
sh: [13](#), [27](#)  
Shell: [12](#), [27](#)  
system: [3](#)

T: 8, 10, 11, 24  
Text: 2, 3, 4, 11, 12, 14, 24, 25  
text: 8, 22, 28  
traverse\_: 10, 13, 14, 26  
Turtle: 27  
updateWalker: 2  
URL: 6, 11, 24

## References

- [Cha14] Oliver Charles. *24 Days of GHC Extensions: Overloaded Strings*. Dec. 17, 2014. URL: <https://ocharles.org.uk/blog/posts/2014-12-17-overloaded-strings.html> (visited on 03/18/2017).
- [Con17] Nix Contributors. *Nixpkgs Contributors Guide*. 2017. URL: <https://nixos.org/nixpkgs/manual/#sec-standard-meta-attributes> (visited on 03/18/2017).
- [Wes14] Sean Westfall. *24 Days of GHC Extensions: Template Haskell*. Dec. 22, 2014. URL: <https://ocharles.org.uk/blog/guest-posts/2014-12-22-template-haskell.html> (visited on 03/18/2017).