Functional programming, Seminar No. 1

Danya Rogozin
Lomonosov Moscow State University,
Serokell OÜ

Higher School of Economics
Faculty of Computer Science

General words on Haskell

- The language is named after Haskell Curry, an American logician
- The first implementation: 1990
- The language standard: Haskell2010
- Default compiler: Glasgow Haskell compiler
- Haskell is a strongly-typed, polymorphic, and purely functional programming language

General words on Haskell

- The language is named after Haskell Curry, an American logician
- The first implementation: 1990
- The language standard: Haskell2010
- Default compiler: Glasgow Haskell compiler
- Haskell is a strongly-typed, polymorphic, and purely functional programming language
- This course is quite introductory. We partially piggyback the ITMO Haskell course.

General words on Haskell

- The language is named after Haskell Curry, an American logician
- The first implementation: 1990
- The language standard: Haskell2010
- Default compiler: Glasgow Haskell compiler
- Haskell is a strongly-typed, polymorphic, and purely functional programming language
- This course is quite introductory. We partially piggyback the ITMO Haskell course.
- Vox populi:

Chris Burnor
@chrisburnor

Is Haskell the Rick and Morty of programming languages? 🤔 twitter.com/thejameskyle/s...

The Haskell Platform installation

There are several ways to install the Haskell platform on Mac:

- Download the `.pkg` file and install the corresponding package

The Haskell Platform installation

There are several ways to install the Haskell platform on Mac:

- Download the `.pkg` file and install the corresponding package
- Run the script `curl -sSL https://get.haskellstack.org/ | sh`

The Haskell Platform installation

There are several ways to install the Haskell platform on Mac:

- Download the `.pkg` file and install the corresponding package
- Run the script `curl -sSL https://get.haskellstack.org/ | sh`
- Install ghc, stack, and cabal via Homebrew

The Haskell Platform installation

There are several ways to install the Haskell platform on Mac:

- Download the `.pkg` file and install the corresponding package
- Run the script `curl -sSL https://get.haskellstack.org/ | sh`
- Install ghc, stack, and cabal via Homebrew

Choose any way you prefer. All these ways are equivalent to each other.

The Haskell Platform installation

There are several ways to install the Haskell platform on Mac:

- Download the `.pkg` file and install the corresponding package
- Run the script `curl -sSL https://get.haskellstack.org/ | sh`
- Install ghc, stack, and cabal via Homebrew

Choose any way you prefer. All these ways are equivalent to each other.

I'm a Mac user, but I believe that you'll manage to install the Haskell Platform on NixOs/Windows/Linux/etc quite quickly.

GHC

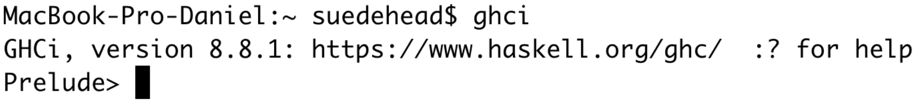
- GHC is a default Haskell compiler as we told above
- GHC is an open-source project. Don't hesitate to contribute!
- GHC is mostly implemented on Haskell
- GHC is developed under the GHC Steering committee control

GHC

- GHC is a default Haskell compiler as we told above
- GHC is an open-source project. Don't hesitate to contribute!
- GHC is mostly implemented on Haskell
- GHC is developed under the GHC Steering committee control
- Very roughly, compiling pipeline is arranged as follows:
  parsing ⇒ compile-time (type-checking mostly) ⇒ runtime (program execution)

GHCi

- GHCi is a Haskell interpreter based on GHC
- One may run GHCi with a quite simple command `ghci` on a shell
- You play with GHCi as a calculator, the ordinary arithmetical operators are written in a usual way
- Take a look at the GHCi chapter in the GHC User's Guide to be familiar with GHCi closely



```
MacBook-Pro-Daniel:~ suedehead$ ghci
GHCi, version 8.8.1: https://www.haskell.org/ghc/  :? for help
Prelude> █
```

Cabal

- Cabal is a system of library and dependency management
- A `.cabal` file describes the version of a package and its dependencies
- Cabal is also a packaging tool
- Keep in mind that Cabal is known as a reason of so-called dependency hell

Cabal

- Cabal is a system of library and dependency management
- A `.cabal` file describes the version of a package and its dependencies
- Cabal is also a packaging tool
- Keep in mind that Cabal is known as a reason of so-called dependency hell

That's how this dependency hell might look like:

Stack

- Stack is a cross-platform build tool for Haskell projects
- Stack allows one to

Stack

- Stack is a cross-platform build tool for Haskell projects
- Stack allows one to
    - install packages and version of GHC (and their concrete versions) you need

Stack

- Stack is a cross-platform build tool for Haskell projects
- Stack allows one to
  - install packages and version of GHC (and their concrete versions) you need
  - build, execute, and test projects

Stack

- Stack is a cross-platform build tool for Haskell projects
- Stack allows one to
    - install packages and version of GHC (and their concrete versions) you need
    - build, execute, and test projects
    - reproduce builds

Stack

- Stack is a cross-platform build tool for Haskell projects
- Stack allows one to
  - install packages and version of GHC (and their concrete versions) you need
  - build, execute, and test projects
  - reproduce builds
  - create an isolated location

Snapshots

- Snapshot is a curated package set used by Stack

Snapshots

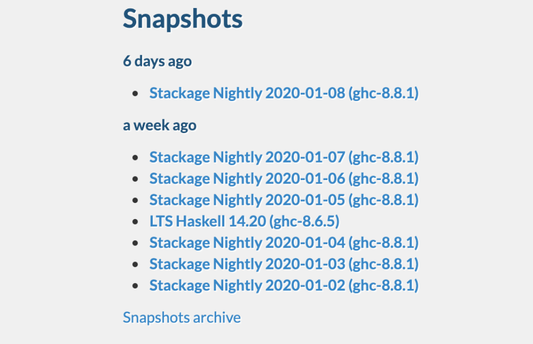
- Snapshot is a curated package set used by Stack
- Stackage is a stable repository that stores snapshots

Snapshots

- Snapshot is a curated package set used by Stack
- Stackage is a stable repository that stores snapshots
- Resolver is a reference to a required snapshot

Snapshots

- Snapshot is a curated package set used by Stack
- Stackage is a stable repository that stores snapshots
- Resolver is a reference to a required snapshot
- Let us take a look at the screenshot from Stackage:



Snapshots

6 days ago

- Stackage Nightly 2020-01-08 (ghc-8.8.1)

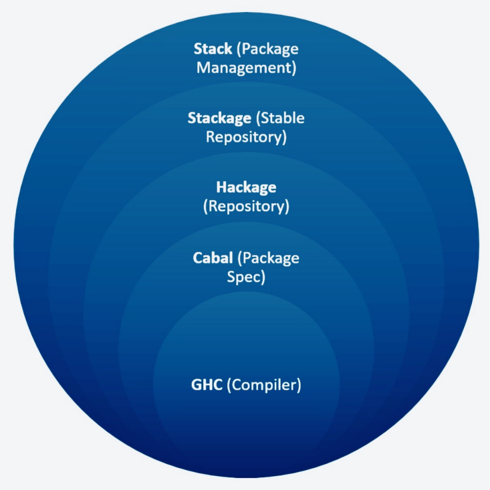
a week ago

- Stackage Nightly 2020-01-07 (ghc-8.8.1)
- Stackage Nightly 2020-01-06 (ghc-8.8.1)
- Stackage Nightly 2020-01-05 (ghc-8.8.1)
- LTS Haskell 14.20 (ghc-8.6.5)
- Stackage Nightly 2020-01-04 (ghc-8.8.1)
- Stackage Nightly 2020-01-03 (ghc-8.8.1)
- Stackage Nightly 2020-01-02 (ghc-8.8.1)

Snapshots archive

Ecosystem encapsulation

The Haskell ecosystem encapsulation might be described as the following sequence:



Creating a Haskell project via Stack

- Figure out how to call your project and run the script `stack new <projectname>`
- You will see the following story after the command `tree .` in the project directory:

Creating a Haskell project via Stack

- Figure out how to call your project and run the script `stack new <projectname>`
- You will see the following story after the command `tree .` in the project directory:

```
MacBook-Pro-Daniel:myFirstProject suedehead$ tree .
.
├── ChangeLog.md
├── LICENSE
├── README.md
├── Setup.hs
├── app
│   └── Main.hs
├── myFirstProject.cabal
├── package.yaml
├── src
│   └── Lib.hs
├── stack.yaml
└── test
    └── Spec.hs

3 directories, 10 files
```

stack.yaml

Let us discuss dependencies files in a Haskell project. First of all, we observe the stack.yaml file:

stack.yaml

Let us discuss dependencies files in a Haskell project. First of all, we observe the stack.yaml file:

```
resolver: lts-14.19

# User packages to be built.
# Various formats can be used as shown in the example below.
#
# packages:
# - some-directory
# - https://example.com/foo/bar/baz-0.0.2.tar.gz
#   subdirs:
#   - auto-update
#   - wai
packages:
- .

# extra-deps:
# - acme-missiles-0.3
# - git: https://github.com/commercialhaskell/stack.git
#   commit: e7b331f14bcffb8367cd58fbfc8b40ec7642100a
#
# extra-deps: []
```

Cabal file

As we told above, the `.cabal` file describe the relevant version of a project and its dependencies:

Cabal file

As we told above, the `.cabal` file describe the relevant version of a project and its dependencies:

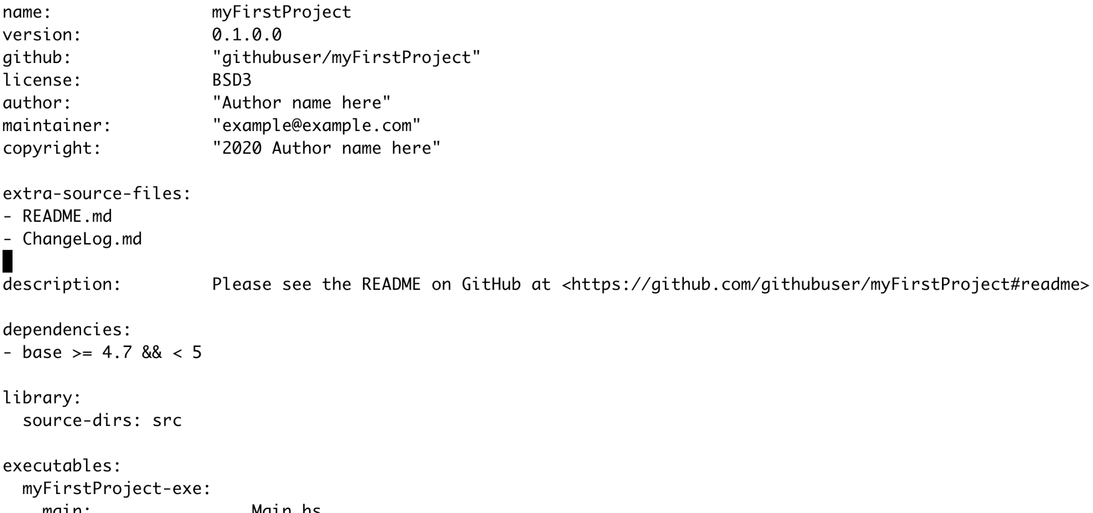
```
cabal-version: 1.12
name:           myFirstProject
version:        0.1.0.0
description:    Please see the README on GitHub at <https://github.com/githubuser/myFirstProject#readme>
homepage:       https://github.com/githubuser/myFirstProject#readme
bug-reports:    https://github.com/githubuser/myFirstProject/issues
author:         Author name here
maintainer:     example@example.com
copyright:      2020 Author name here
license:        BSD3
license-file:   LICENSE
build-type:     Simple
extra-source-files:
    README.md
    ChangeLog.md

source-repository head
  type: git
  location: https://github.com/githubuser/myFirstProject

library
```

package.yaml

The package.yaml generates automatically from the stack.yaml and .cabal files:

package.yaml

The package.yaml generates automatically from the stack.yaml and .cabal files:



```
name:                myFirstProject
version:             0.1.0.0
github:              "githubuser/myFirstProject"
license:             BSD3
author:              "Author name here"
maintainer:          "example@example.com"
copyright:           "2020 Author name here"

extra-source-files:
- README.md
- ChangeLog.md

description:         Please see the README on GitHub at <https://github.com/githubuser/myFirstProject#readme>

dependencies:
- base >= 4.7 && < 5

library:
  source-dirs: src

executables:
  myFirstProject-exe:
    main:                Main.hs
```

Building and running a project

The following commands are crucially important:

Building and running a project

The following commands are crucially important:

- `stack build`

Building and running a project

The following commands are crucially important:

- `stack build`
- `stack run`

Building and running a project

The following commands are crucially important:

- `stack build`
- `stack run`
- `stack exec`

Building and running a project

The following commands are crucially important:

- `stack build`
- `stack run`
- `stack exec`
- `stack ghci`

Building and running a project

The following commands are crucially important:

- `stack build`
- `stack run`
- `stack exec`
- `stack ghci`
- `stack clean`

Building and running a project

The following commands are crucially important:

- `stack build`
- `stack run`
- `stack exec`
- `stack ghci`
- `stack clean`
- `stack test`

Building and running a project

The following commands are crucially important:

- `stack build`
- `stack run`
- `stack exec`
- `stack ghci`
- `stack clean`
- `stack test`

The roles of these commands follow from their quite self-explanatory names.

Hackage

According to its description, ’Hackage is the Haskell community’s central package archive of open source software‘.

Hackage

According to its description, 'Hackage is the Haskell community's central package archive of open source software'.

- Webpage: `https://hackage.haskell.org`

Hackage

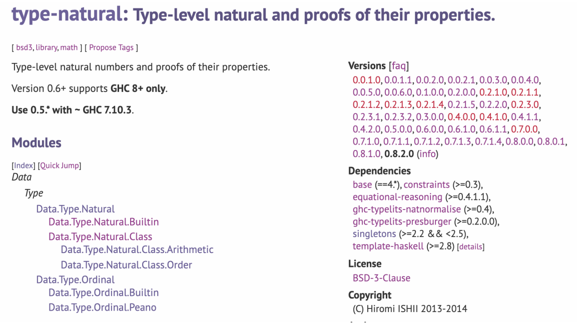
According to its description, 'Hackage is the Haskell community's central package archive of open source software'.

- Webpage: `https://hackage.haskell.org`
- Browsing packages, simplified package search, current uploads.

Hackage

According to its description, 'Hackage is the Haskell community's central package archive of open source software'.

- Webpage: `https://hackage.haskell.org`
- Browsing packages, simplified package search, current uploads.



type-natural: Type-level natural and proofs of their properties.

[bsd3, library, math] [Propose Tags]

Type-level natural numbers and proofs of their properties.

Version 0.6+ supports **GHC 8+ only**.

Use 0.5.* with ~ GHC 7.10.3.

Modules

[Index] [Quick Jump]

Data
- *Type*
  - Data.Type.Natural
    - Data.Type.Natural.Builtin
    - Data.Type.Natural.Class
      - Data.Type.Natural.Class.Arithmetic
      - Data.Type.Natural.Class.Order
  - Data.Type.Ordinal
    - Data.Type.Ordinal.Builtin
    - Data.Type.Ordinal.Peano

Versions [faq]
0.0.1.0, 0.0.1.1, 0.0.2.0, 0.0.2.1, 0.0.3.0, 0.0.4.0, 0.0.5.0, 0.0.6.0, 0.1.0.0, 0.2.0.0, 0.2.1.0, 0.2.1.1, 0.2.1.2, 0.2.1.3, 0.2.1.4, 0.2.1.5, 0.2.2.0, 0.2.3.0, 0.2.3.1, 0.2.3.2, 0.3.0.0, 0.4.0.0, 0.4.1.0, 0.4.1.1, 0.4.2.0, 0.5.0.0, 0.6.0.0, 0.6.1.0, 0.6.1.1, 0.7.0.0, 0.7.1.0, 0.7.1.1, 0.7.1.2, 0.7.1.3, 0.7.1.4, 0.8.0.0, 0.8.0.1, 0.8.1.0, **0.8.2.0** (info)

Dependencies
base (==4.*), constraints (>=0.3), equational-reasoning (>=0.4.1.1), ghc-typelits-natnormalise (>=0.4), ghc-typelits-presburger (>=0.2.0.0), singletons (>=2.2 && <2.5), template-haskell (>=2.8) [details]

License
BSD-3-Clause

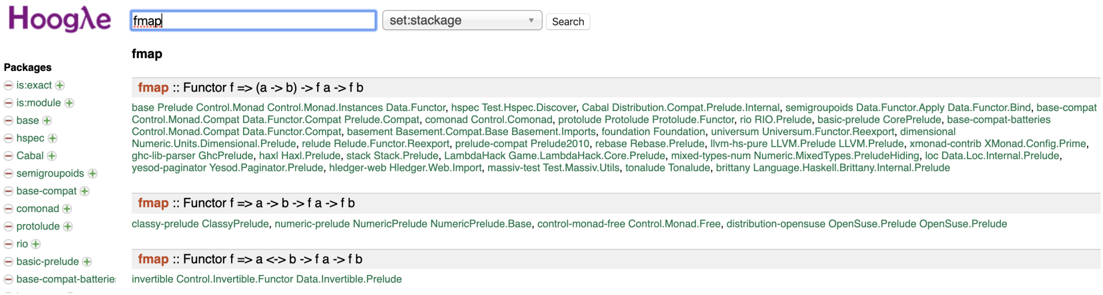
Copyright
(C) Hiromi ISHII 2013-2014

Hoogle

Hoogle is a sort of Haskell search engine. Webpage: `https://hoogle.haskell.org`.

Hoogle

Hoogle is a sort of Haskell search engine. Webpage: `https://hoogle.haskell.org`.



fmap | set:stackage ▾ | Search

fmap

Packages

- ⊖ is:exact ⊕
- ⊖ is:module ⊕
- ⊖ base ⊕
- ⊖ hspec ⊕
- ⊖ Cabal ⊕
- ⊖ semigroupoids ⊕
- ⊖ base-compat ⊕
- ⊖ comonad ⊕
- ⊖ protolude ⊕
- ⊖ rio ⊕
- ⊖ basic-prelude ⊕
- ⊖ base-compat-batterie

fmap :: Functor f => (a -> b) -> f a -> f b

base Prelude Control.Monad Control.Monad.Instances Data.Functor, hspec Test.Hspec.Discover, Cabal Distribution.Compat.Prelude.Internal, semigroupoids Data.Functor.Apply Data.Functor.Bind, base-compat Control.Monad.Compat Data.Functor.Compat Prelude.Compat, comonad Control.Comonad, protolude Protolude Protolude.Functor, rio RIO.Prelude, basic-prelude CorePrelude, base-compat-batteries Control.Monad.Compat Data.Functor.Compat, basement Basement.Compat.Base Basement.Imports, foundation Foundation, universum Universum.Functor.Reexport, dimensional Numeric.Units.Dimensional.Prelude, relude Relude.Functor.Reexport, prelude-compat Prelude2010, rebase Rebase.Prelude, llvm-hs-pure LLVM.Prelude LLVM.Prelude, xmonad-contrib XMonad.Config.Prime, ghc-lib-parser GhcPrelude, haxl Haxl.Prelude, stack Stack.Prelude, LambdaHack Game.LambdaHack.Core.Prelude, mixed-types-num Numeric.MixedTypes.PreludeHiding, loc Data.Loc.Internal.Prelude, yesod-paginator Yesod.Paginator.Prelude, hledger-web Hledger.Web.Import, massiv-test Test.Massiv.Utils, tonalude Tonalude, brittany Language.Haskell.Brittany.Internal.Prelude

fmap :: Functor f => a -> b -> f a -> f b

classy-prelude ClassyPrelude, numeric-prelude NumericPrelude NumericPrelude.Base, control-monad-free Control.Monad.Free, distribution-opensuse OpenSuse.Prelude OpenSuse.Prelude

fmap :: Functor f => a <-> b -> f a -> f b

invertible Control.Invertible.Functor Data.Invertible.Prelude

Summary

We observed today such topics as

1. General aspects of GHC and GHCi
2. The Haskell Platform installation
3. Dependency management via Stack and Cabal

Summary

We observed today such topics as

1. General aspects of GHC and GHCi
2. The Haskell Platform installation
3. Dependency management via Stack and Cabal
4. In other words, the Haskell ecosystem in a nutshell

Summary

We observed today such topics as

1. General aspects of GHC and GHCi
2. The Haskell Platform installation
3. Dependency management via Stack and Cabal
4. In other words, the Haskell ecosystem in a nutshell

On the next seminar, we will discuss:

1. The basic Haskell syntax
2. The underlying aspects of the Haskell type system
3. Functions and lambdas
4. Immutability and Laziness