

# Functional Programming in a Stateful World

Manuel M T Chakravarty

*Applicative &  
Tweag I/O*

-  [mchakravarty](#)
-  [TacticalGrace](#)
-  [justtesting.org](#)



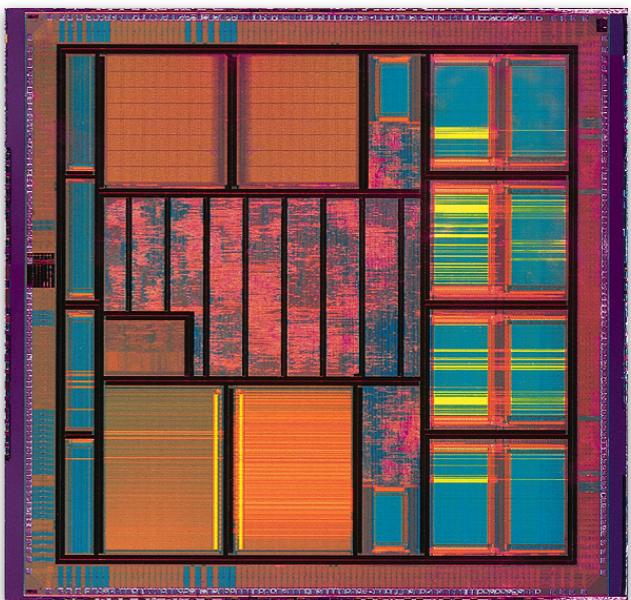
# Abstract

λ

# Abstract

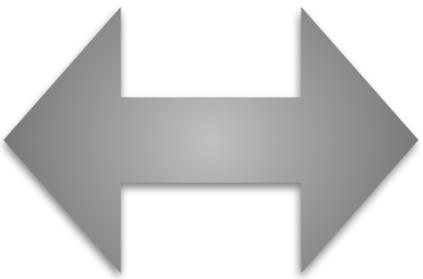
λ

# Concrete

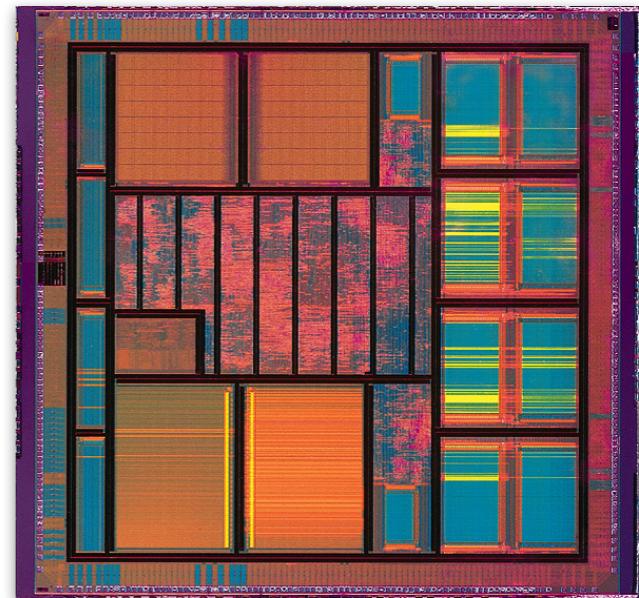


Abstract

$\lambda$

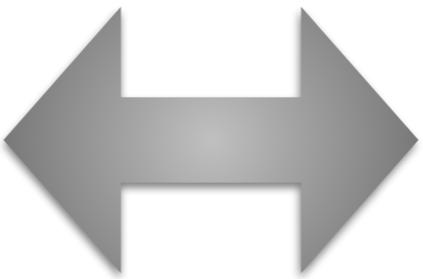


Concrete

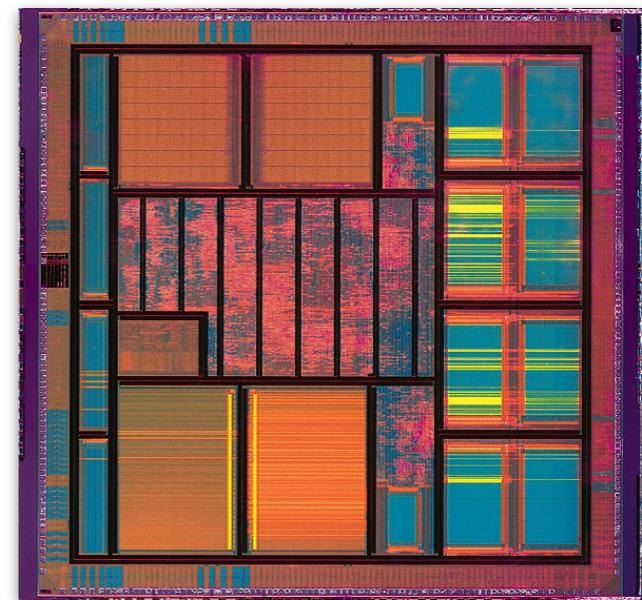


Abstract

$\lambda$



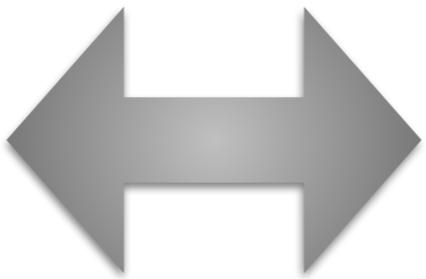
Concrete



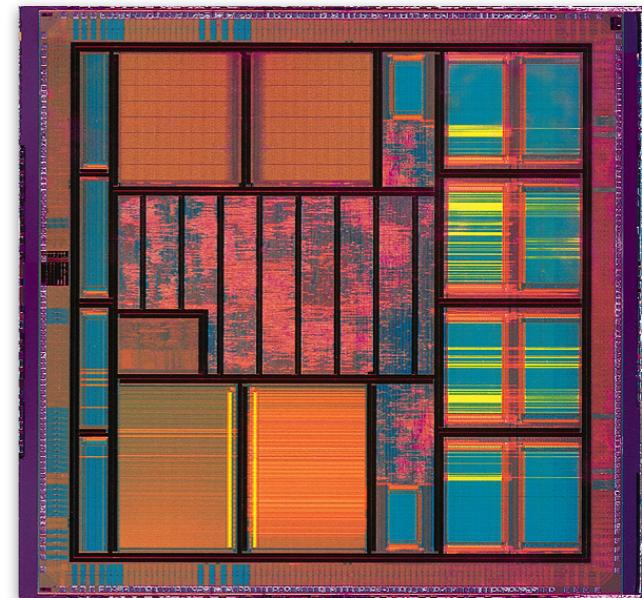
Entangled

Abstract

$\lambda$



Concrete

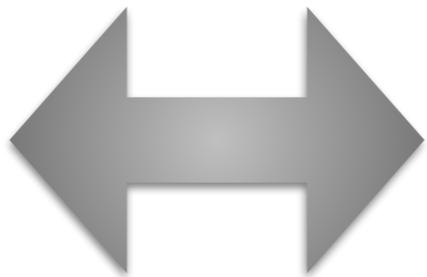


Compositional

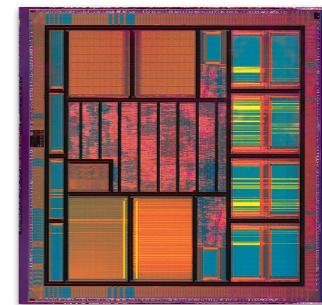
Entangled

# Abstract

$\lambda$



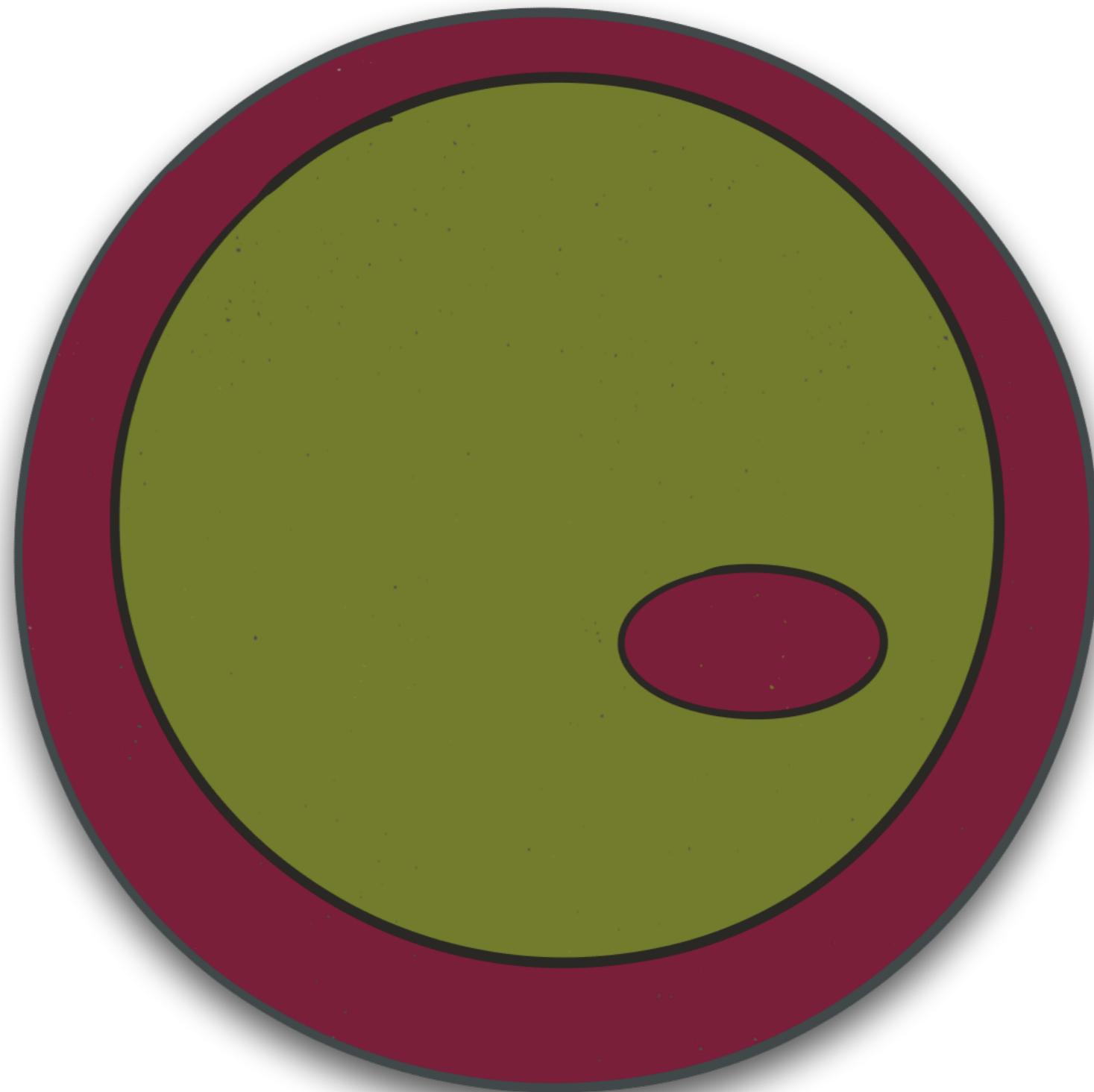
Concrete



# Compositional

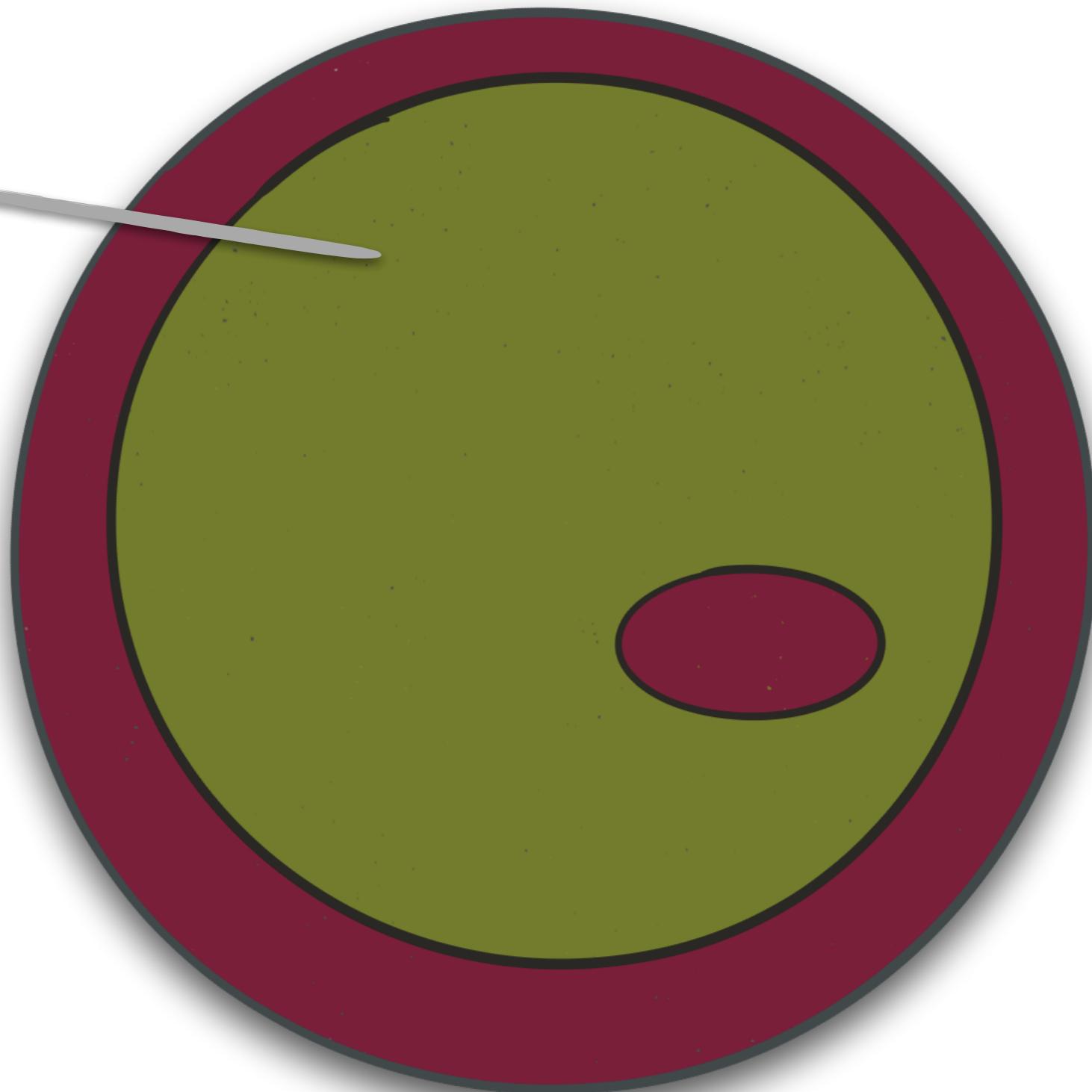
Entangled

# Anatomy of a Haskell Program



# Anatomy of a Haskell Program

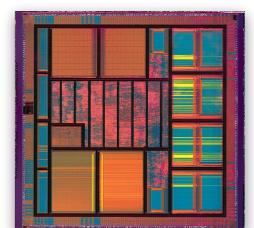
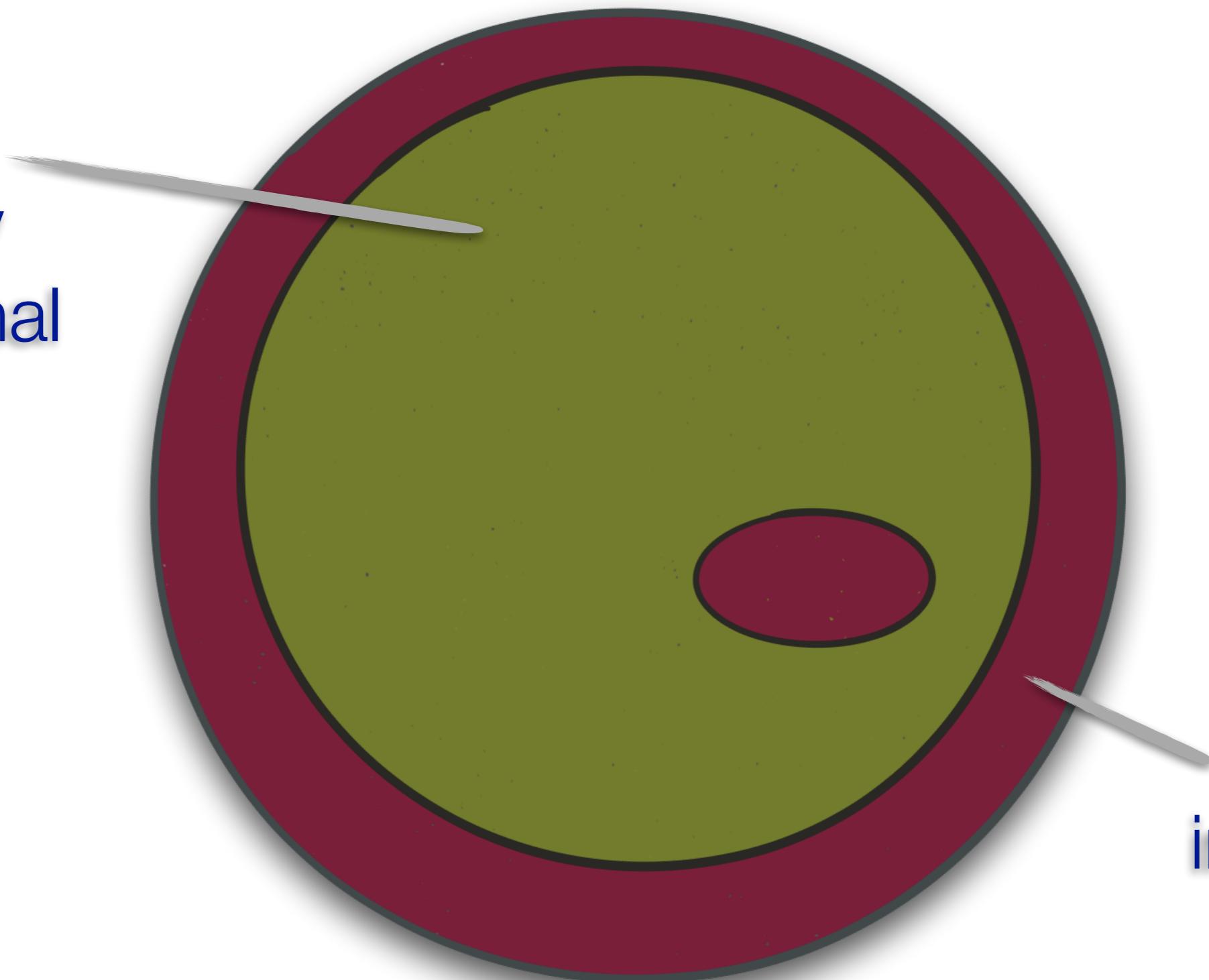
$\lambda$   
purely  
functional  
core



# Anatomy of a Haskell Program

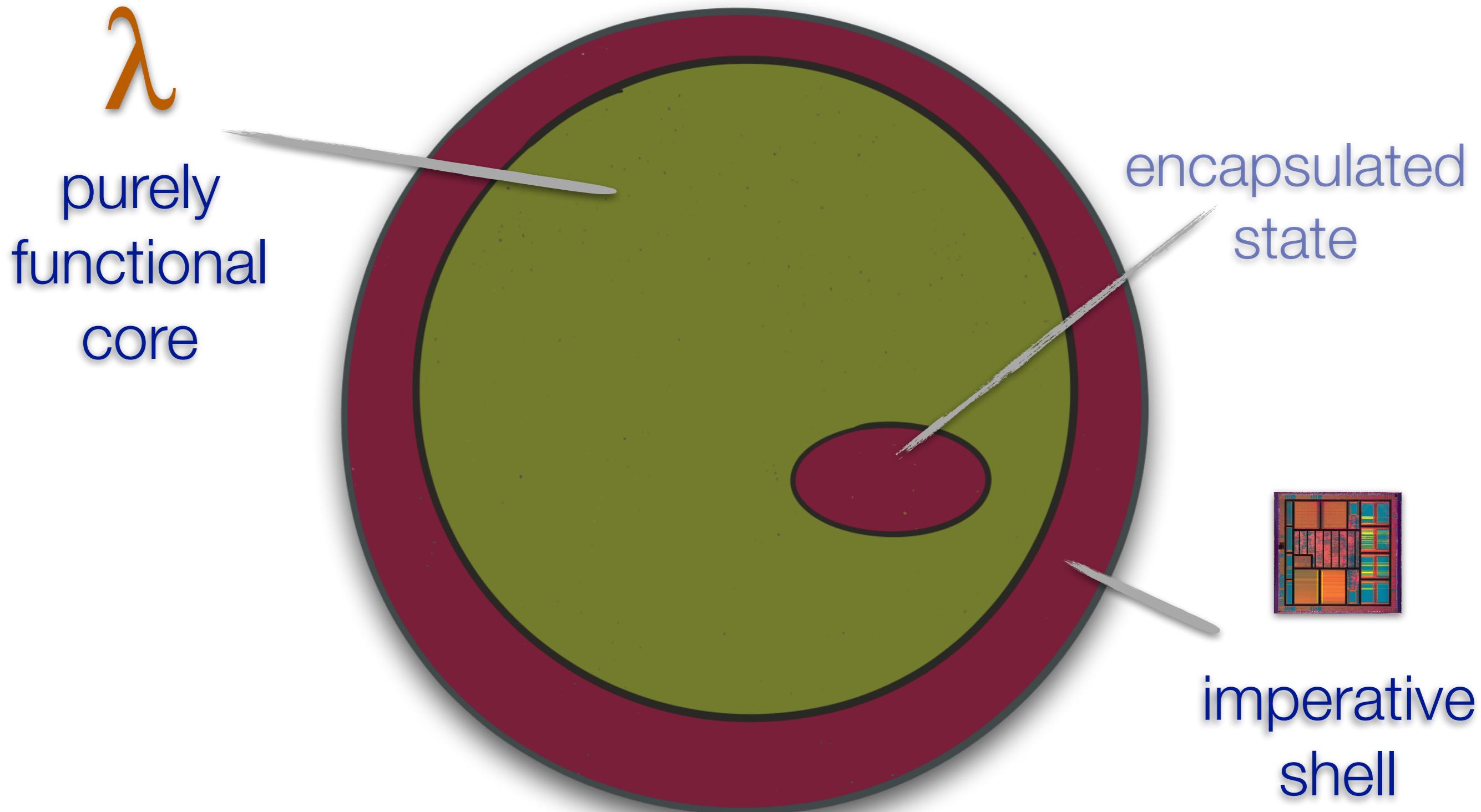
$\lambda$

purely  
functional  
core



imperative  
shell

# Anatomy of a Haskell Program





# Pipeline Applications



# Pipeline Applications

Read request or data



# Pipeline Applications

Read request or data

Compute .....

database



# Pipeline Applications

Read request or data

Compute .....

database

Emit result



# Pipeline Applications

Read request or data

Compute .....  
database

Emit result

Clojure

Haskell

Server-side apps

Command-line apps

Ocaml

Scala



# Pipeline Applications

Read request or data

Compute .....  
database

**What about user-centric & mobile applications?**

Clojure      Haskell  
Ocaml      Scala  
Server-side apps  
Command-line apps

# Swift



# Swift



# Swift

Can we extend the pipeline architecture?





GUI elements have their own derived state



Must use OO-based platform libraries

GUI elements have their own derived state

The background image shows a detailed view of an industrial facility, likely a refinery or chemical plant. It features a complex network of large, rusted metal pipes, some with white protective coatings. These pipes are supported by a steel framework of beams and ladders. Several vertical cylindrical tanks and smaller structures are visible against a clear blue sky.

Frequent asynchronous events

Must use OO-based platform libraries

GUI elements have their own derived state

The background image shows a detailed view of an industrial facility, likely a refinery or chemical plant. It features a complex network of large, rusted metal pipes of various sizes, some with flanges and valves. These pipes are supported by a dense framework of blue-painted steel beams and walkways. In the center, there's a tall, cylindrical structure, possibly a reactor or distillation column, with multiple levels and platforms. The sky is clear and blue.

Must be responsive

Frequent asynchronous events

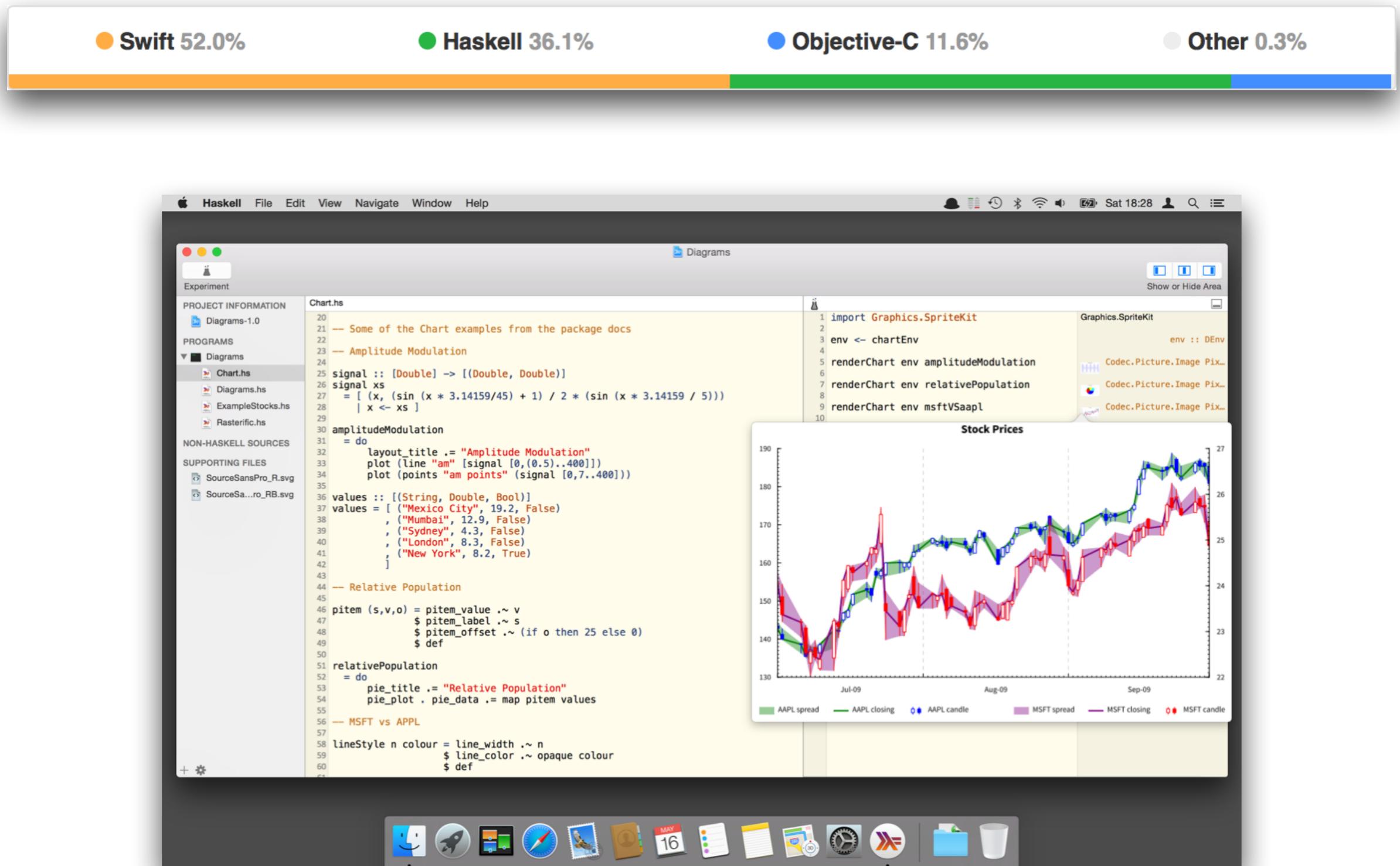
Must use OO-based platform libraries

GUI elements have their own derived state

“How can We Use Functional  
Programming in this context?”

# A GUI-centric Mac app in Swift & Haskell

# A GUI-centric Mac app in Swift & Haskell



# Three Methods to Facilitate FP

# Three Methods to Facilitate FP

Application architecture

**Isolate side effects with a view model**



# Three Methods to Facilitate FP

Application architecture

**Isolate side effects with a view model**



Immutable values

**Structs & enums over classes; and use let (not var)**



# Three Methods to Facilitate FP

Application architecture

**Isolate side effects with a view model**



Immutable values

**Structs & enums over classes; and use let (not var)**



Time series

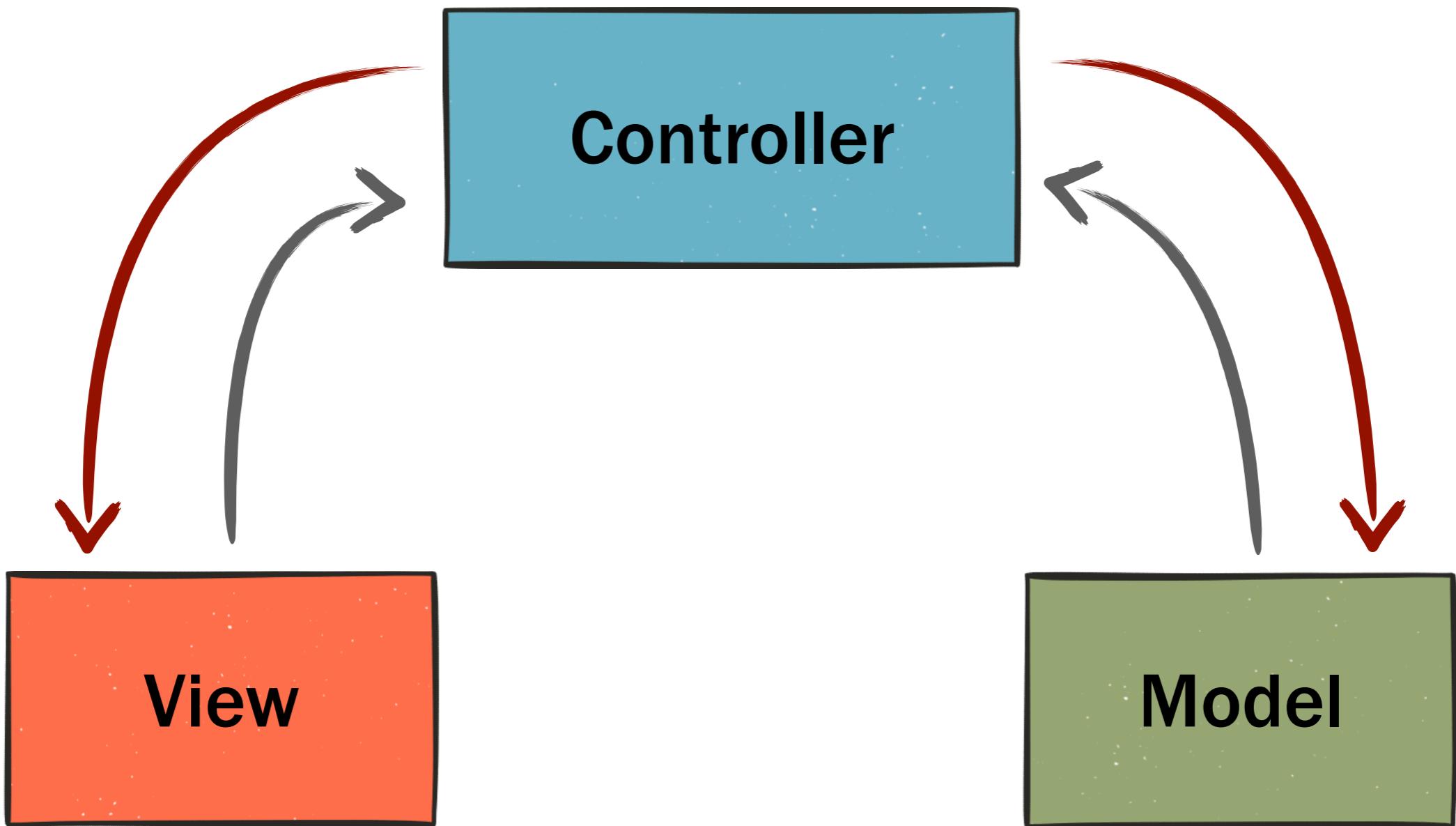
**Streams of values over time replace mutable state**



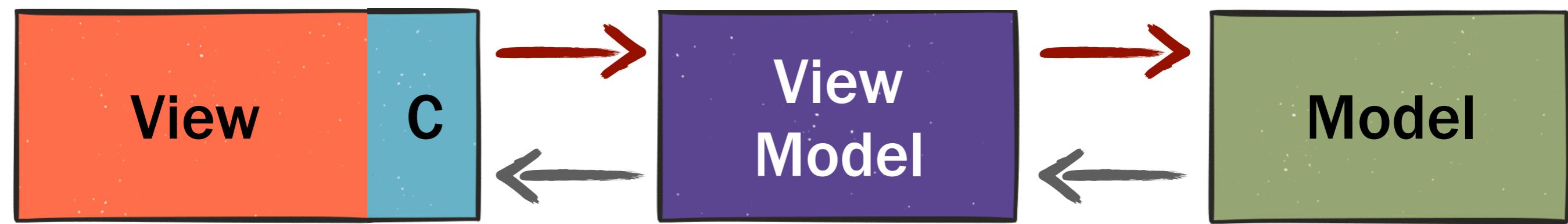


Architecture  
Use a Separate View Model

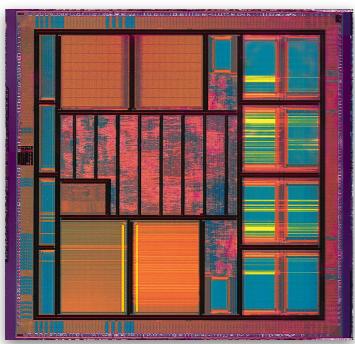
# What is a View Model?



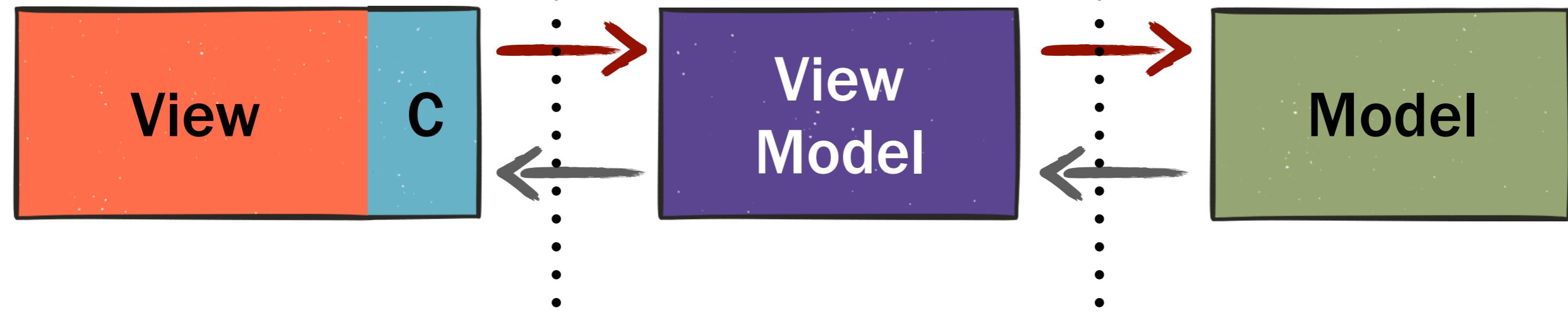
# What is a View Model?



# What is a View Model?



# Mutable



# Project Descriptions in Haskell for Mac

# Project Descriptions in Haskell for Mac

Serialised      Model      Data

```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et
15 al
16 description: This project demonstrates the use
17 of the Rasterific, Diagrams, and Chart libraries
18 in Haskell for Mac's interactive playgrounds.
19 category: Sample project
20 author: Manuel M T Chakravarty
21 tested-with:
22 data-files: SourceSansPro_R.svg
23 SourceSansPro_RB.svg
24 data-dir: ""
25 extra-source-files:
26 Executable Diagrams
27 main-is: Diagrams.hs
28 buildable: True
29 other-modules: Chart ExampleStocks Rasterific
```

# Project Descriptions in Haskell for Mac

View

Serialised

Model

Data

PROJECT INFORMATION

Diagrams.cabal

PROGRAMS

- Diagrams
  - Chart.hs
  - Diagrams.hs
  - ExampleStocks.hs
  - Rasterific.hs

NON-HASKELL SOURCES

SUPPORTING FILES

- SourceSansPro\_R.svg
- SourceSansPro\_RB.svg

Identity

Name: Diagrams

Version: 1.0

Category: Sample project

Synopsis: Demonstrate the use of the Diagrams et al

Full Description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.

Author: Manuel M T Chakravarty

Maintainer: Names of current package maintainers

Licensing

Copyright: Copyright 2015 Manuel M T Chakravarty

License Type: BSD3

License File:

```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et
15 al
16 description: This project demonstrates the use
17 of the Rasterific, Diagrams, and Chart libraries
18 in Haskell for Mac's interactive playgrounds.
19 category: Sample project
20 author: Manuel M T Chakravarty
21 tested-with:
22 data-files: SourceSansPro_R.svg
23 SourceSansPro_RB.svg
24 data-dir: ""
25 extra-source-files:
26 Executable Diagrams
27 main-is: Diagrams.hs
28 buildable: True
29 other-modules: Chart ExampleStocks Rasterific
```

# Project Descriptions in Haskell for Mac

View

Serialised

Model

Data

PROJECT INFORMATION

Diagrams.cabal

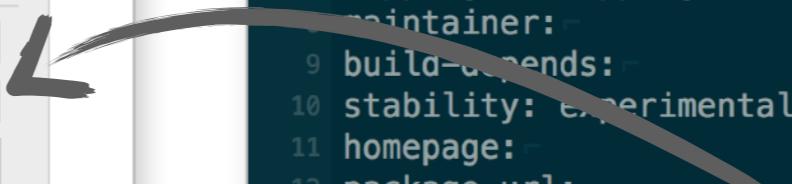
PROGRAMS

- Diagrams
  - Chart.hs
  - Diagrams.hs
  - ExampleStocks.hs
  - Rasterific.hs

NON-HASKELL SOURCES

SUPPORTING FILES

- SourceSansPro\_R.svg
- SourceSansPro\_RB.svg



Identity

Name:

Version:

Category:

Synopsis:

Full Description:

Author:

Maintainer:

Licensing

Copyright:

License Type:

License File:

```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al
15 description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
   SourceSansPro_RB.svg
20 data-dir: ""
21 extra-source-files:
22 Executable Diagrams
23 main-is: Diagrams.hs
24 buildable: True
25 other-modules: Chart ExampleStocks Rasterific
```

# Project Descriptions in Haskell for Mac

**View**

**Serialised**   **Model**   **Data**

The diagram illustrates the flow of project information from the Haskell for Mac interface to the Cabal file and then to the serialised representation.

**Haskell for Mac Interface:** On the left, the "PROJECT INFORMATION" pane shows a project named "Diagrams-1.0" with files like Chart.hs, Diagrams.hs, ExampleStocks.hs, and Rasterific.hs. The "PROGRAMS" section lists "Diagrams". The "NON-HASKELL SOURCES" section lists "SourceSansPro\_R.svg" and "SourceSansPro\_RB.svg". The "SUPPORTING FILES" section lists "SourceS...ro\_R.svg" and "SourceS...o\_RB.svg".

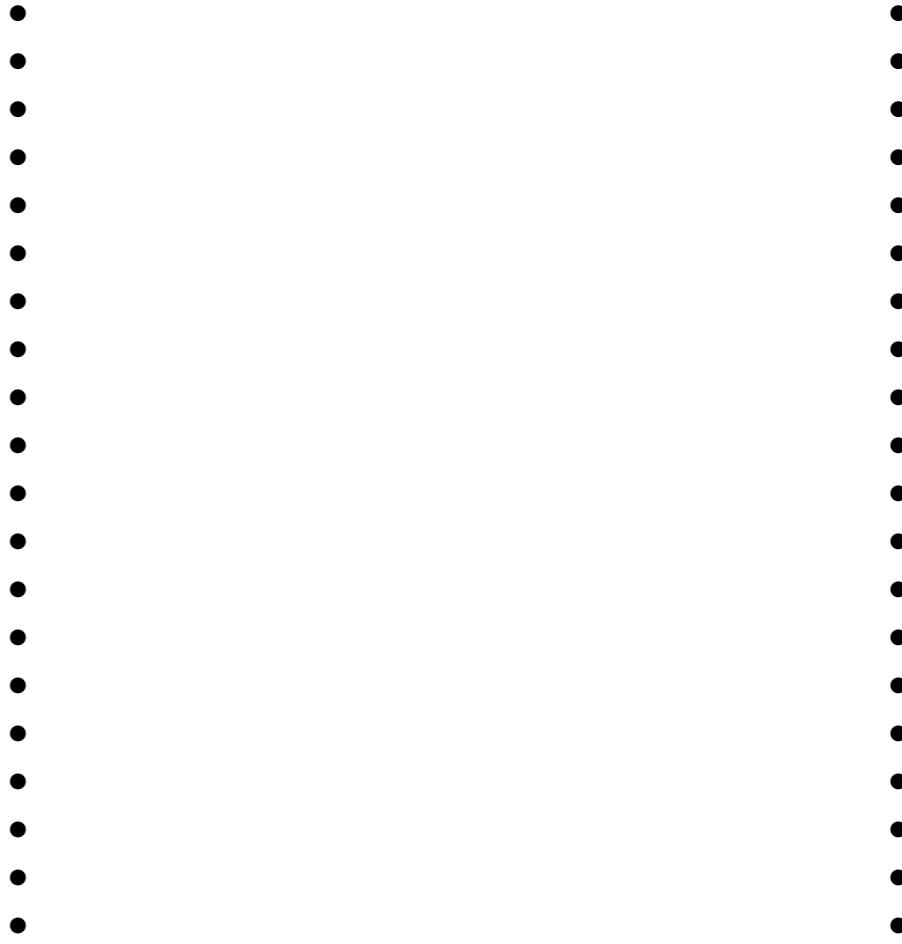
**Cabal File:** In the center, the "Diagrams.cabal" file is shown with the following content:

```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al
15 description: This project demonstrates the use
of the Rasterific, Diagrams, and Chart libraries
in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
SourceSansPro_RB.svg
20 data-dir: ""
21 extra-source-files:
22 Executable Diagrams
23 main-is: Diagrams.hs
24 buildable: True
25 other-modules: Chart ExampleStocks Rasterific
```

**Serialised Representation:** On the right, the serialised representation of the project is shown as a JSON object:

```
{ "name": "Diagrams", "version": "1.0", "cabal-version": "1.6", "build-type": "Simple", "license": "BSD3", "license-file": "", "copyright": "Copyright 2015 Manuel M T Chakravarty", "maintainer": null, "build-depends": null, "stability": "experimental", "homepage": null, "package-url": null, "bug-reports": null, "synopsis": "Demonstrate the use of the Diagrams et al", "description": "This project demonstrates the use\nof the Rasterific, Diagrams, and Chart libraries\nin Haskell for Mac's interactive playgrounds.", "category": "Sample project", "author": "Manuel M T Chakravarty", "tested-with": null, "data-files": ["SourceSansPro_R.svg", "SourceSansPro_RB.svg"], "data-dir": "", "extra-source-files": null, "executables": [{"name": "Diagrams", "main-is": "Diagrams.hs", "buildable": true}], "other-modules": ["Chart", "ExampleStocks", "Rasterific"] }
```

Arrows indicate the flow of data from the Haskell for Mac interface through the Cabal file to the serialised representation. The "Name" field in the interface is highlighted in blue, matching the "name" field in the Cabal file and the "name" key in the JSON object. The "Synopsis" and "description" fields in the Cabal file are circled in green, corresponding to the "synopsis" and "description" keys in the JSON object.



PROJECT INFORMATION

**Diagrams-1.0**

PROGRAMS

- Diagrams
  - Chart.hs
  - Diagrams.hs
  - ExampleStocks.hs
  - Rasterific.hs

SUPPORTING FILES

- SourceS...\_R.svg
- SourceS...\_RB.svg

Diagrams.cabal

**Identity**

Name: **Diagrams**  
Version: 1.0  
Category: Sample project  
Synopsis: Demonstrate the use of the Diagrams et al  
Full Description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.  
Author: Manuel M T Chakravarty  
Maintainer: Names of current package maintainers

**Licensing**

Copyright: Copyright 2015 Manuel M T Chakravarty  
License Type: BSD3  
License File:



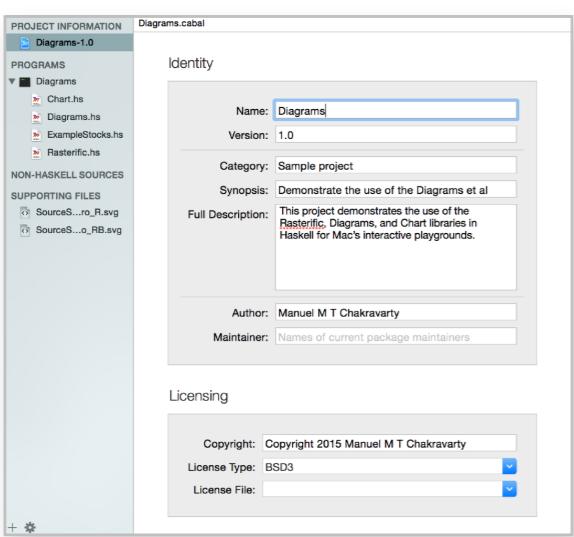
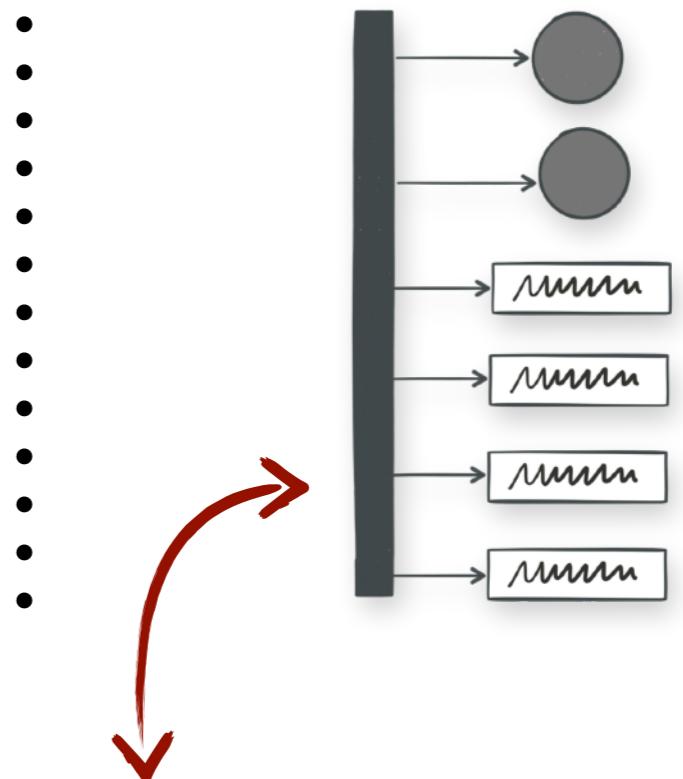
```

1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al
15 description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
20 SourceSansPro_RB.svg
21 data-dir: ""
22 extra-source-files:
23 Executable Diagrams
24 main-is: Diagrams.hs
25 buildable: True
26 other-modules: Chart ExampleStocks Rasterific

```

Model

## Immutable



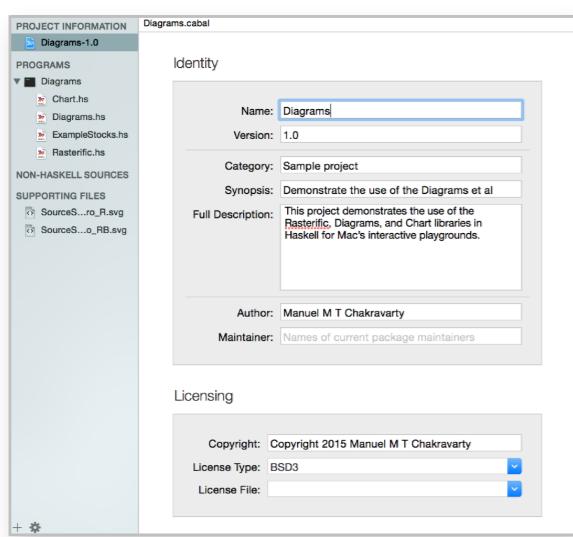
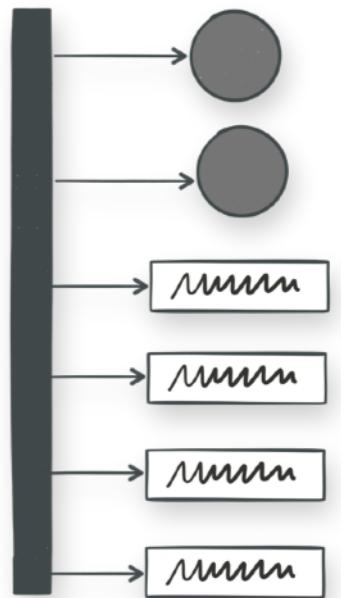
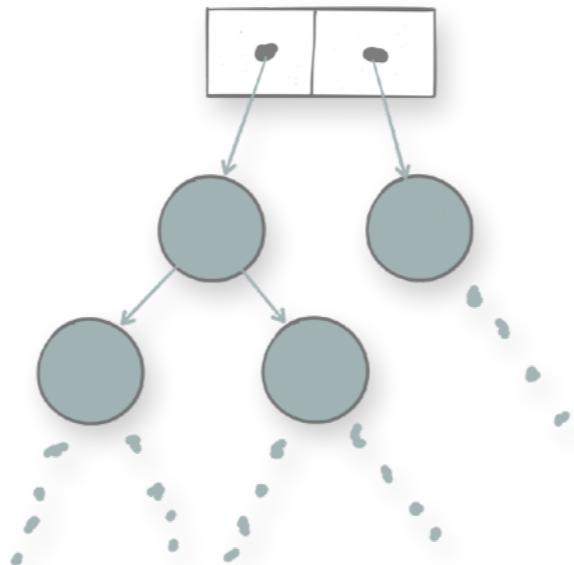
A screenshot of a Haskell code editor. At the top, there is a Haskell logo consisting of three overlapping white rectangles with a purple 'X' symbol. Below the logo, the word 'HASKELL' is written in a bold, sans-serif font. To the right of the logo, the contents of the Diagrams.cabal file are displayed in a code editor window:

```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al.
15 description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
20 SourceSansPro_RB.svg
21 data-dir: ""
22 extra-source-files:
23 Executable Diagrams
24 main-is: Diagrams.hs
25 buildable: True
26 other-modules: Chart ExampleStocks Rasterific
```

## View Model

## Model

# Immutable

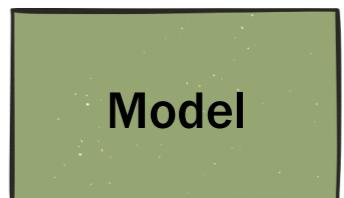
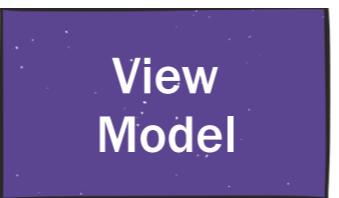


**Haskell**

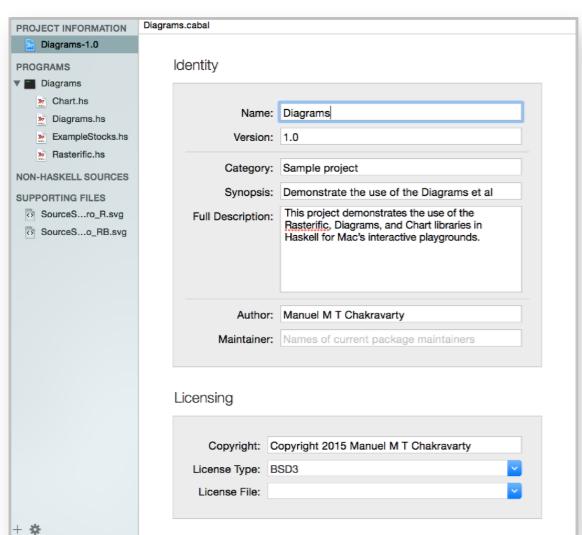
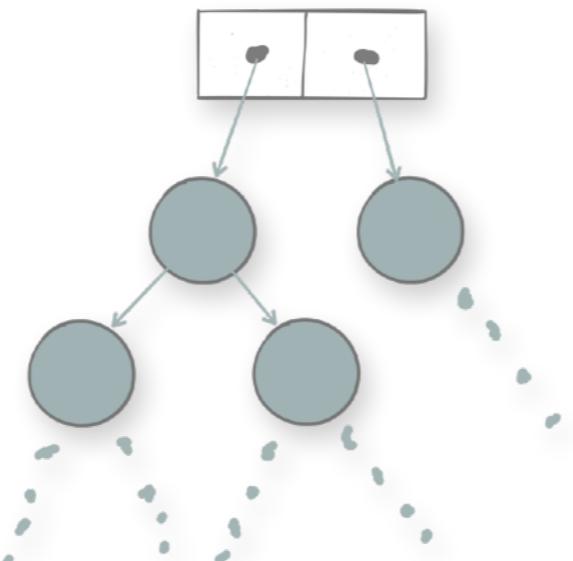
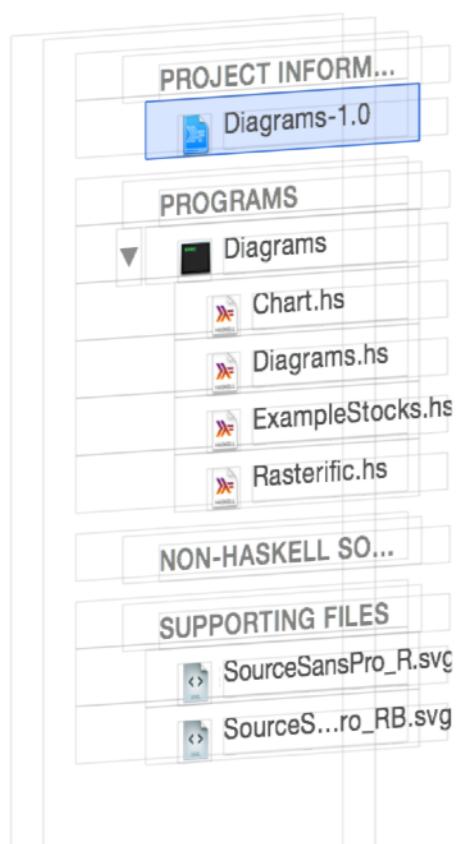
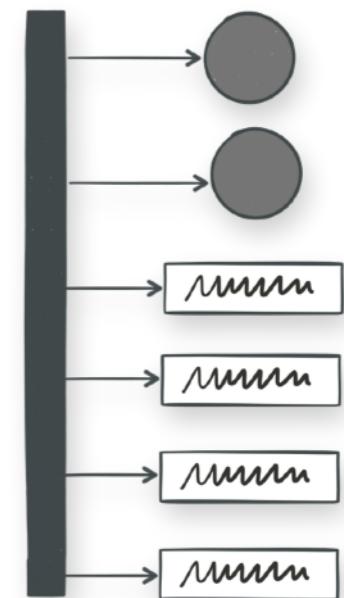
```

1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al.
15 description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
20 SourceSansPro_RB.svg
21 data-dir: ""
22 extra-source-files:
23 Executable Diagrams
24 main-is: Diagrams.hs
25 buildable: True
26 other-modules: Chart ExampleStocks Rasterific

```



## Immutable

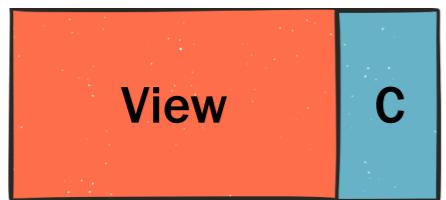


The Haskell logo, which consists of three overlapping white rectangles with a purple 'GHC' monogram in the center.

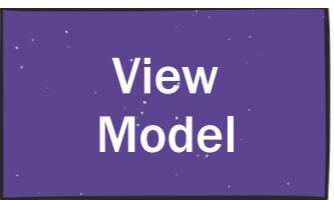
```

1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al
15 description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
20 SourceSansPro_RB.svg
21 data-dir: ""
22 extra-source-files:
23 Executable Diagrams
24 main-is: Diagrams.hs
25 buildable: True
26 other-modules: Chart ExampleStocks Rasterific

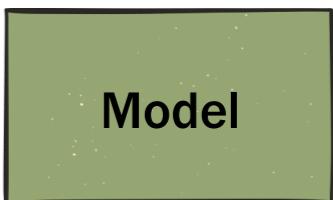
```



•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•



•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•

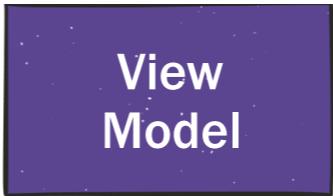


View

```
data PackageDescription
= PackageDescription {
    package          :: PackageIdentifier,
    license          :: License,
    licenseFiles    :: [FilePath],
    copyright        :: String,
    maintainer      :: String,
    author           :: String,
    stability         :: String,
    testedWith       :: [(CompilerFlavor, VersionRange)],
    homepage         :: String,
    pkgUrl          :: String,
    bugReports       :: String,
    sourceRepos     :: [SourceRepo],
    synopsis          :: String,
    description       :: String,
    category          :: String,
    ...
}
```



•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•



•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•  
•

```
data PackageDescription
= PackageDescription {
    package      :: PackageIdentifier,
    license      :: License,
    licenseFiles :: [FilePath],
    copyright    :: String,
    maintainer   :: String,
    author       :: String,
    stability    :: String,
    testedWith   :: [(CompilerFlavor, VersionRange)],
    homepage     :: String,
    pkgUrl       :: String,
    bugReports   :: String,
    sourceRepos  :: [SourceRepo],
    synopsis     :: String,
    description  :: String,
    category     :: String,
    ...
}
```

View

```
struct Package {  
    private var modelPackage: CBLPackage  
  
    ...  
  
    var name: String {  
        get { return modelPackage.name }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                      withName: newValue) }  
    }  
  
    ...  
  
    var richTextDesc: NSAttributedString{  
        get {  
            return NSAttributedString(string:  
                modelPackage.fullDescription) }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                      withFullDescription:  
                        newValue.string) }  
    }  
  
    ...
```

```
    packageIdentifier,  
    license,  
    filePath],  
    ring,  
    ring,  
    ring,  
    ring,  
    ring,  
    VersionRange)],  
    ring,  
    ring,  
    ring,  
    ring,  
    sourceRepo],  
    ring,  
    ring,  
    ring,
```

View

```
struct Package {  
    private var modelPackage: CBLPackage  
  
    ...  
  
    var name: String {  
        get { return modelPackage.name }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withName: newValue) }  
    }  
  
    ...  
  
    var richTextDesc: NSAttributedString{  
        get {  
            return NSAttributedString(string:  
                modelPackage.fullDescription) }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withFullDescription:  
                           newValue.string) }  
    }  
  
    ...
```

```
    packageIdentifier,  
    license,  
    filePath],  
    ring,  
    ring,  
    ring,  
    ring,  
    ring,  
    VersionRange)],  
    ring,  
    ring,  
    ring,  
    ring,  
    sourceRepo],  
    ring,  
    ring,  
    ring,
```

View

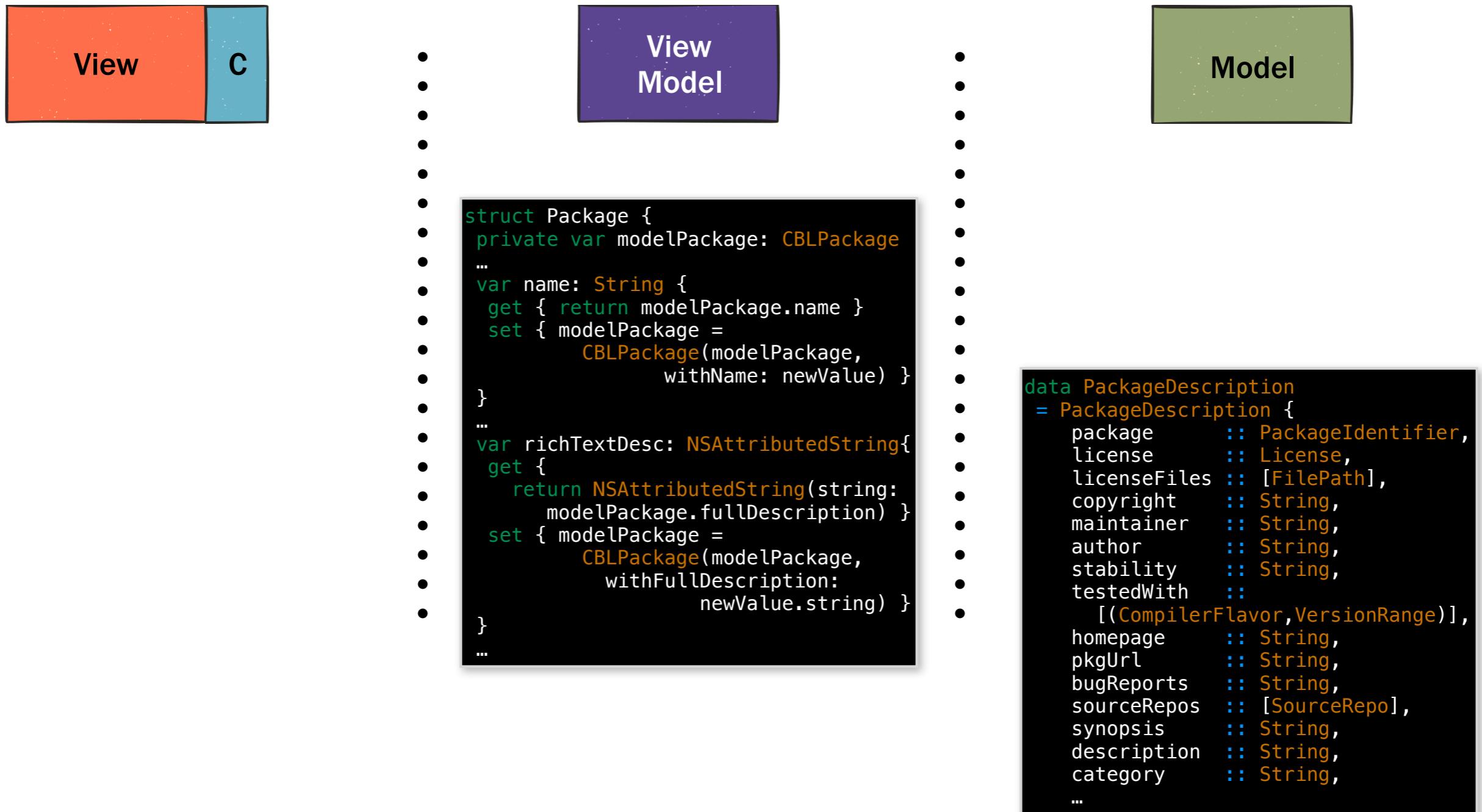
```
struct Package {  
    private var modelPackage: CBLPackage  
  
    ...  
  
    var name: String {  
        get { return modelPackage.name }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withName: newValue) }  
    }  
  
    ...  
  
    var richTextDesc: NSAttributedString{  
        get {  
            return NSAttributedString(string:  
                modelPackage.fullDescription) }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withFullDescription:  
                           newValue.string) }  
    }  
  
    ...
```

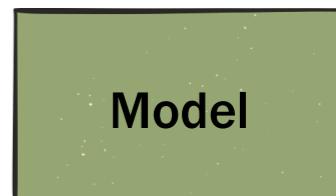
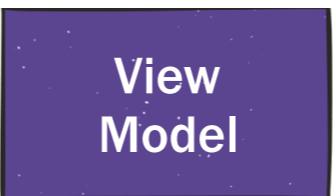
```
    packageIdentifier,  
    license,  
    filePath],  
    ring,  
    ring,  
    ring,  
    ring,  
    ring,  
    VersionRange)],  
    ring,  
    ring,  
    ring,  
    ring,  
    sourceRepo],  
    ring,  
    ring,  
    ring,
```

View

```
struct Package {  
    private var modelPackage: CBLPackage  
  
    ...  
  
    var name: String {  
        get { return modelPackage.name }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                      withName: newValue) }  
    }  
  
    ...  
    var richTextDesc: NSAttributedString {  
        get {  
            return NSAttributedString(string:  
                modelPackage.fullDescription) }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                      withFullDescription:  
                        newValue.string) }  
    }  
  
    ...
```

```
    packageIdentifier,  
    license,  
    filePath],  
    ring,  
    ring,  
    ring,  
    ring,  
    VersionRange)],  
    ring,  
    ring,  
    ring,  
    ring,  
    sourceRepo],  
    ring,  
    ring,  
    ring,
```

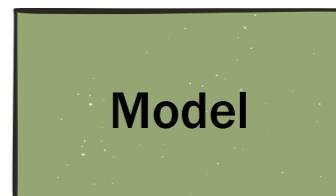
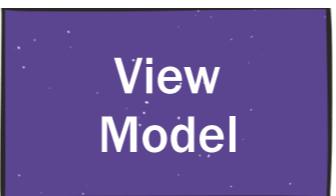




```
struct Package {  
    private var modelPackage: CBI.Package
```

```
class ProjectItem: NSObject {  
    let category: ProjectItemCategory  
    var identifier: String  
    ...  
    weak var parent: ProjectItem?  
    var children: [ProjectItem]  
    var fileWrapper: NSFfileWrapper?  
    ...  
}
```

```
n  
{  
    packageIdentifier,  
    cense,  
    ilePath],  
    ring,  
    ring,  
    ring,  
    ring,  
    ring,  
    VersionRange)],  
    ring,  
    ring,  
    ring,  
    ourceRepo],  
    ring,  
    ring,  
    ring,
```



```
struct Package {  
    private var modelPackage: CBI.Package
```

```
class ProjectItem: NSObject {  
    let category: ProjectItemCategory  
    var identifier: String  
  
    ...  
    weak var parent: ProjectItem?  
    var children: [ProjectItem]  
    var fileWrapper: NSFfileWrapper?  
  
    ...  
}
```

```
n  
{  
    packageIdentifier,  
    cense,  
    ilePath],  
    ring,  
    ring,  
    ring,  
    ring,  
    ring,  
    VersionRange)],  
    ring,  
    ring,  
    ring,  
    ourceRepo],  
    ring,  
    ring,  
    ring,
```

View

C

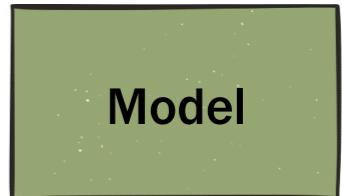
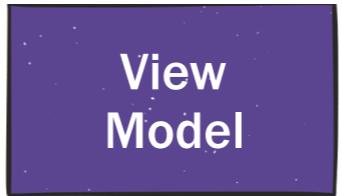
## View Model

Model

```
•
•
•
•
•
struct Package {
    private var modelPackage: CBLPackage
    ...
    var name: String {
        get { return modelPackage.name }
        set { modelPackage =
            CBLPackage(modelPackage,
            withName: newValue) }
    }
    ...
    var richTextDesc: NSAttributedString{
        get {
            return NSAttributedString(string:
                modelPackage.fullDescription) }
        set { modelPackage =
            CBLPackage(modelPackage,
            withFullDescription:
            newValue.string) }
    }
    ...
}
```

```
class ProjectItem: NSObject {
    let category: ProjectItemCategory
    var identifier: String
    ...
    weak var parent: ProjectItem?
    var children: [ProjectItem]
    var fileWrapper: NSFfileWrapper?
}
...
```

```
data PackageDescription
= PackageDescription {
    package :: PackageIdentifier,
    license :: License,
    licenseFiles :: [FilePath],
    copyright :: String,
    maintainer :: String,
    author :: String,
    stability :: String,
    testedWith :: [(CompilerFlavor, VersionRange)],
    homepage :: String,
    pkgUrl :: String,
    bugReports :: String,
    sourceRepos :: [SourceRepo],
    synopsis :: String,
    description :: String,
    category :: String,
    ...
}
```



```
struct Package {  
    private var modelPackage: CBI.Package
```

```
class HeaderEditorController:  
    NSViewController {  
        @IBOutlet  
        weak var nameTextField: NSTextField!  
        ...  
        @IBOutlet  
        var fullDescriptionTextView:  
            NSTextView!  
        ...  
        let packageItem:  
            ProjectItem  
        ...  
    }
```

```
    n  
    {  
        packageIdentifier,  
        cense,  
        ilePath],  
        ring,  
        ring,  
        ring,  
        ring,  
        ring,  
        VersionRange)],  
        ring,  
        ring,  
        ring,  
        sourceRepo],  
        ring,  
        ring,  
        ring,
```

View

C

```
class HeaderEditorController: NSViewController {
    @IBOutlet weak var nameTextField: NSTextField!
    ...
    @IBOutlet var fullDescriptionTextView: NSTextView!
    ...
    let packageItem: ProjectItem
}
```

View  
Model

```
...
struct Package {
    private var modelPackage: CBLPackage
    ...
    var name: String {
        get { return modelPackage.name }
        set { modelPackage =
            CBLPackage(modelPackage,
                       withName: newValue) }
    }
    ...
    var richTextDesc: NSAttributedString {
        get {
            return NSAttributedString(string:
                modelPackage.fullDescription) }
        set { modelPackage =
            CBLPackage(modelPackage,
                       withFullDescription:
                           newValue.string) }
    }
}
...
```

```
class ProjectItem: NSObject {
    let category: ProjectItemCategory
    var identifier: String
    ...
    weak var parent: ProjectItem?
    var children: [ProjectItem]
    var fileWrapper: NSFfileWrapper?
}
...
```

Model

```
data PackageDescription
 = PackageDescription {
    package      :: PackageIdentifier,
    license      :: License,
    licenseFiles :: [FilePath],
    copyright   :: String,
    maintainer  :: String,
    author       :: String,
    stability    :: String,
    testedWith   :: [(CompilerFlavor, VersionRange)],
    homepage    :: String,
    pkgUrl      :: String,
    bugReports   :: String,
    sourceRepos  :: [SourceRepo],
    synopsis     :: String,
    description  :: String,
    category     :: String,
}
```



# Immutable Values

## Minimise Mutable State

# Immutable by Default

# Immutable by Default

Sometimes performance requires mutability

# Immutable by Default

Sometimes performance requires mutability

Sometimes an existing API is based on mutability

# Immutable by Default

Sometimes performance requires mutability  
**Beware of premature optimisation!**

Sometimes an existing API is based on mutability

# Immutable by Default

Sometimes performance requires mutability  
**Beware of premature optimisation!**

Sometimes an existing API is based on mutability  
**Can you wrap the mutable API?**

# **Immutable**

**Model**

# Immutable

Model

- ✓ Cabal package in Haskell
- ✓ Source files as `NSFileWrapper`



```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al
15 description: This project demonstrates the use
of the Rasterific, Diagrams, and Chart libraries
in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
SourceSansPro_RB.svg
20 data-dir: ""
21 extra-source-files:
22 Executable Diagrams
23 main-is: Diagrams.hs
24 buildable: True
25 other-modules: Chart ExampleStocks Rasterific
```

# Immutable

## Model

- ✓ Cabal package in Haskell
- ✓ Source files as `NSFileWrapper`  
**Directories are not explicit in the model**



```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al
15 description: This project demonstrates the use
of the Rasterific, Diagrams, and Chart libraries
in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
SourceSansPro_RB.svg
20 data-dir: ""
21 extra-source-files:
22 Executable Diagrams
23 main-is: Diagrams.hs
24 buildable: True
25 other-modules: Chart ExampleStocks Rasterific
```

# Immutable

## Model

- ✓ Cabal package in Haskell
- ✓ Source files as `NSFileWrapper`  
**Directories are not explicit in the model**



```
1 name: Diagrams
2 version: 1.0
3 cabal-version: 1.6
4 build-type: Simple
5 license: BSD3
6 license-file: ""
7 copyright: Copyright 2015 Manuel M T Chakravarty
8 maintainer:
9 build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et al
15 description: This project demonstrates the use of the Rasterific, Diagrams, and Chart libraries in Haskell for Mac's interactive playgrounds.
16 category: Sample project
17 author: Manuel M T Chakravarty
18 tested-with:
19 data-files: SourceSansPro_R.svg
SourceSansPro_RB.svg
20 data-dir: ""
21 extra-source-files:
22 Executable Diagrams
23 main-is: Diagrams.hs
24 buildable: True
25 other-modules: Chart ExampleStocks Rasterific
```

At least provide an immutable interface

# Hybrid

View  
Model

```
struct Package {  
    private var modelPackage: CBLPackage  
    ...  
    var name: String {  
        get { return modelPackage.name }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withName: newValue) }  
    }  
    ...  
    var richTextDesc: NSAttributedString {  
        get {  
            return NSAttributedString(string:  
                modelPackage.fullDescription) }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withFullDescription:  
                           newValue.string) }  
    }  
    ...
```

```
struct Package {  
    private var modelPackage: CBLPackage  
    ...  
    var name: String {  
        get { return modelPackage.name }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withName: newValue) }  
    }  
    ...  
    var richTextDesc: NSAttributedString {  
        get {  
            return NSAttributedString(string:  
                modelPackage.fullDescription) }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withFullDescription:  
                           newValue.string) }  
    }  
    ...
```

Don't copy

```
struct Package {  
    private var modelPackage: CBLPackage  
    ...  
    var name: String {  
        get { return modelPackage.name }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withName: newValue) }  
    }  
    ...  
    var richTextDesc: NSAttributedString {  
        get {  
            return NSAttributedString(string:  
                modelPackage.fullDescription) }  
        set { modelPackage =  
            CBLPackage(modelPackage,  
                       withFullDescription:  
                           newValue.string) }  
    }  
    ...
```

Don't copy

Use lets  
& structs

# Hybrid

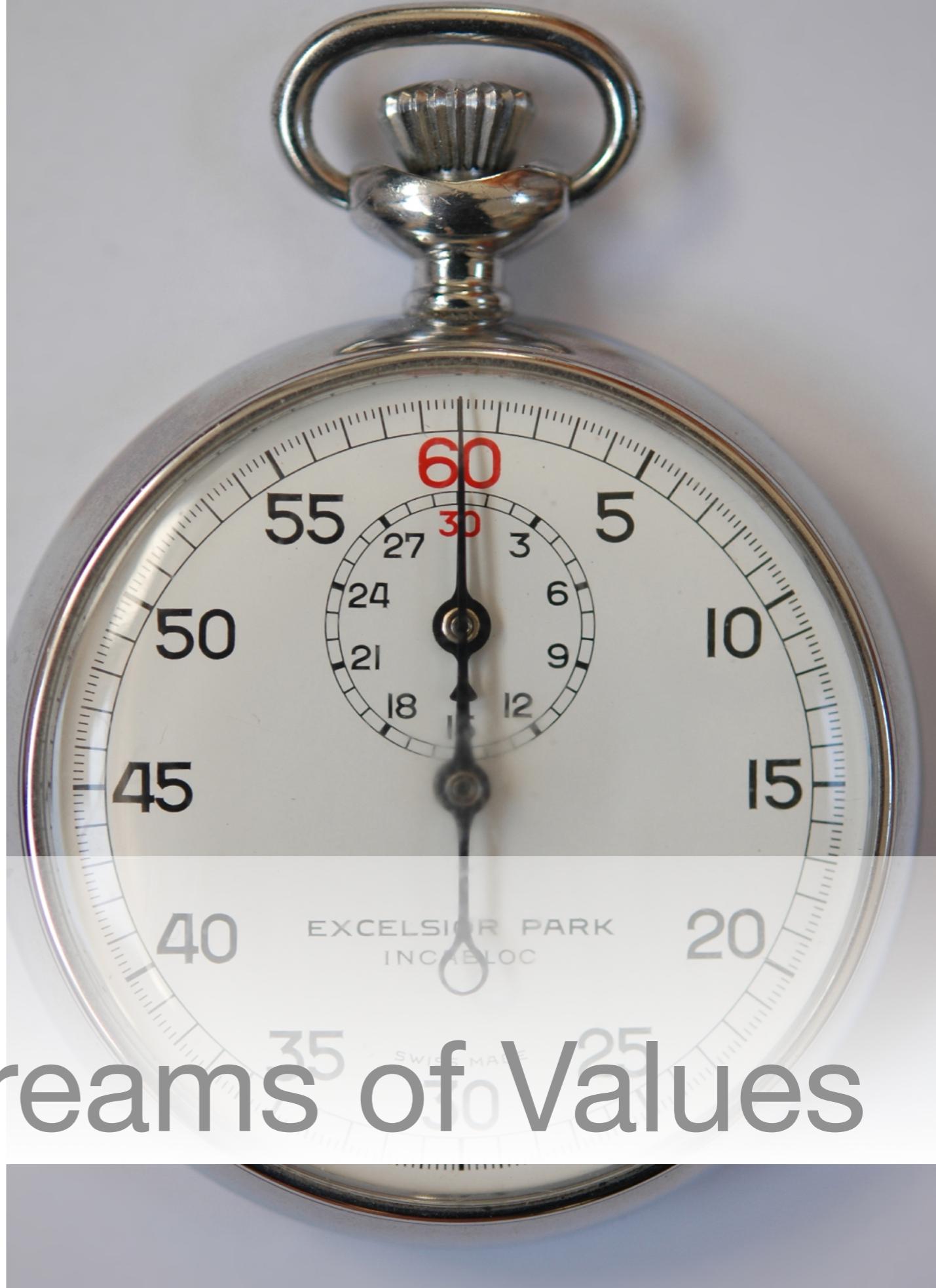
## View Model

Don't copy

Use lets  
& structs

```
struct Package {  
    private var modelPackage: CBLPackage  
  
class ProjectItem: NSObject {  
    let category: ProjectItemCategory  
    var identifier: String  
  
    ...  
    weak var parent: ProjectItem?  
    var children: [ProjectItem]  
    var fileWrapper: NSFfileWrapper?  
    ...  
}
```

# Time Series Changes as Streams of Values



# Three Principles for FP in a Stateful World



[haskellformac.com](http://haskellformac.com)



[mchakravarty](https://github.com/mchakravarty)



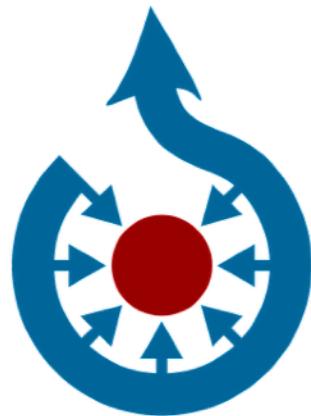
[TacticalGrace](https://twitter.com/TacticalGrace)



[justtesting.org](http://justtesting.org)

Thank you!

# Free Image Sources



Wikimedia

<http://commons.wikimedia.org/wiki/File:Diopsis.jpg>

[http://commons.wikimedia.org/wiki/File:Goldfields\\_Pipeline\\_SMC.JPG](http://commons.wikimedia.org/wiki/File:Goldfields_Pipeline_SMC.JPG)

[http://commons.wikimedia.org/wiki/File:NeuSchwanstein\\_7.JPG](http://commons.wikimedia.org/wiki/File:NeuSchwanstein_7.JPG)

<https://en.wikipedia.org/wiki/File:HeTube.jpg>

[http://commons.wikimedia.org/wiki/File:Stopwatch\\_A.jpg](http://commons.wikimedia.org/wiki/File:Stopwatch_A.jpg)

dribbble

<https://dribbble.com/shots/1667698-Free-vector-Macbook-Ipad-and-Iphone>