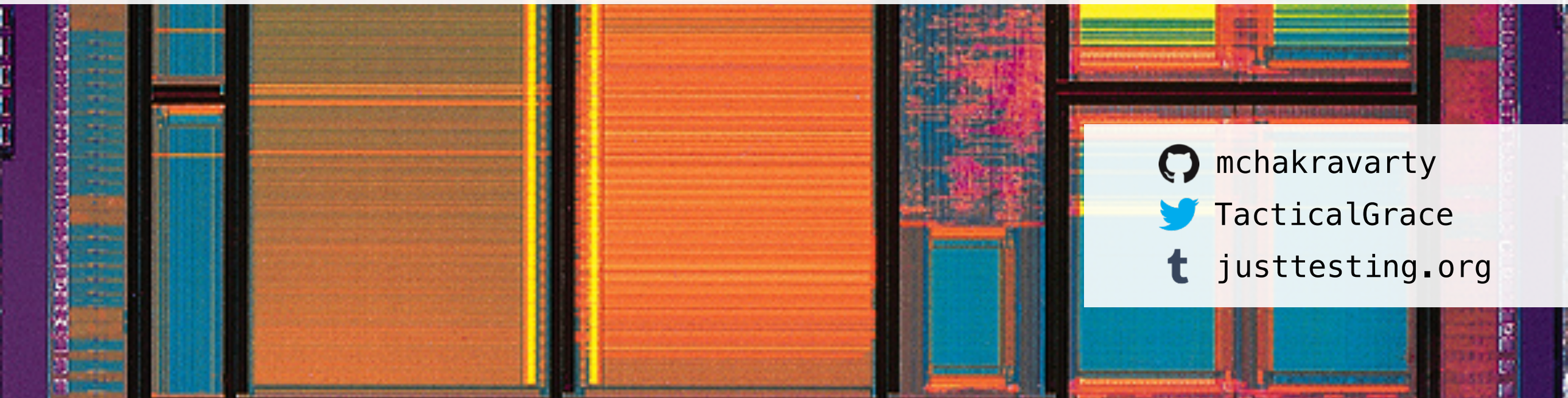
Functional Programming in a Stateful World

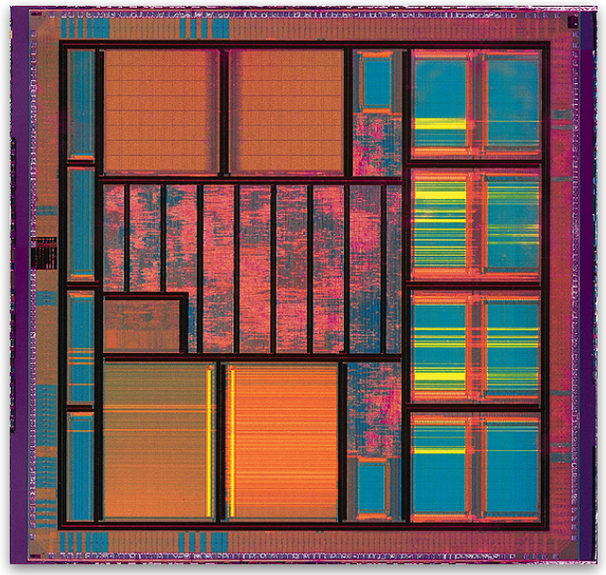
Manuel M T Chakravarty
Applicative &
Tweag I/O



Abstract

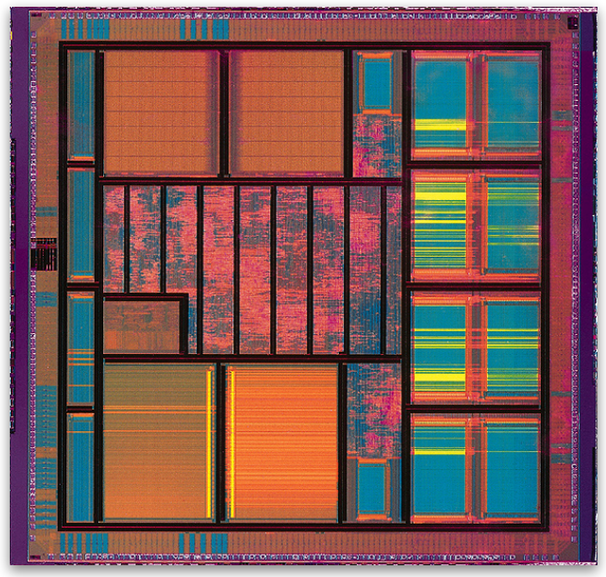
Abstract

Concrete



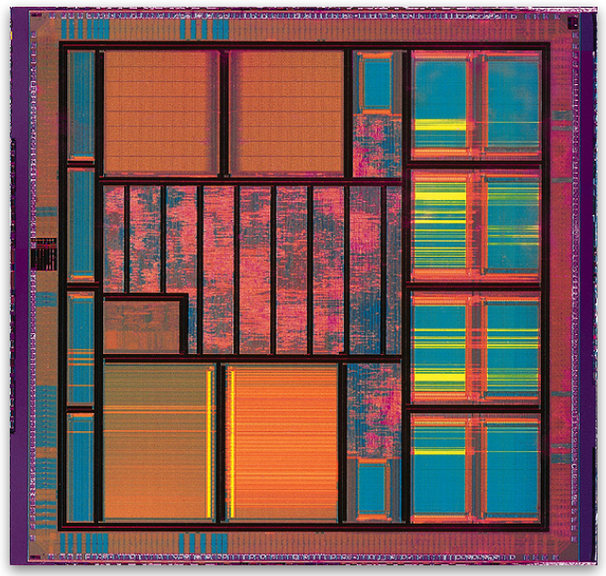
Abstract

Concrete



Abstract

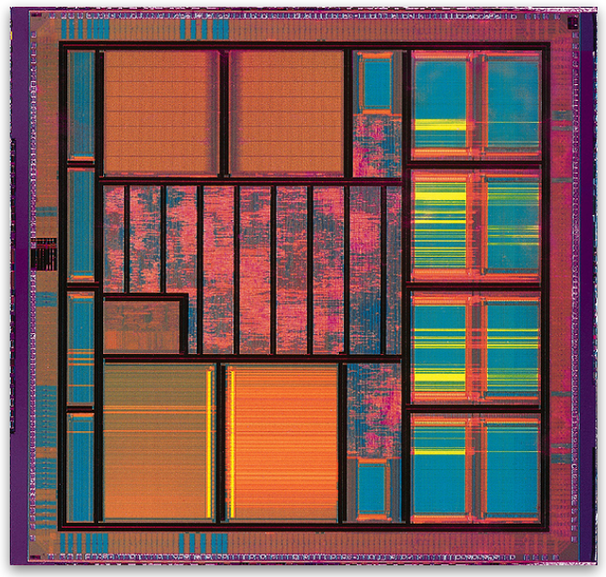
Concrete



Entangled

Abstract

Concrete

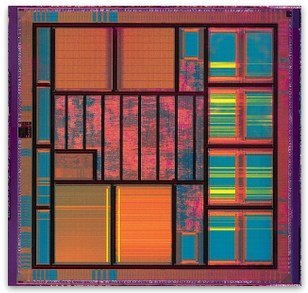


Compositional

Entangled

Abstract

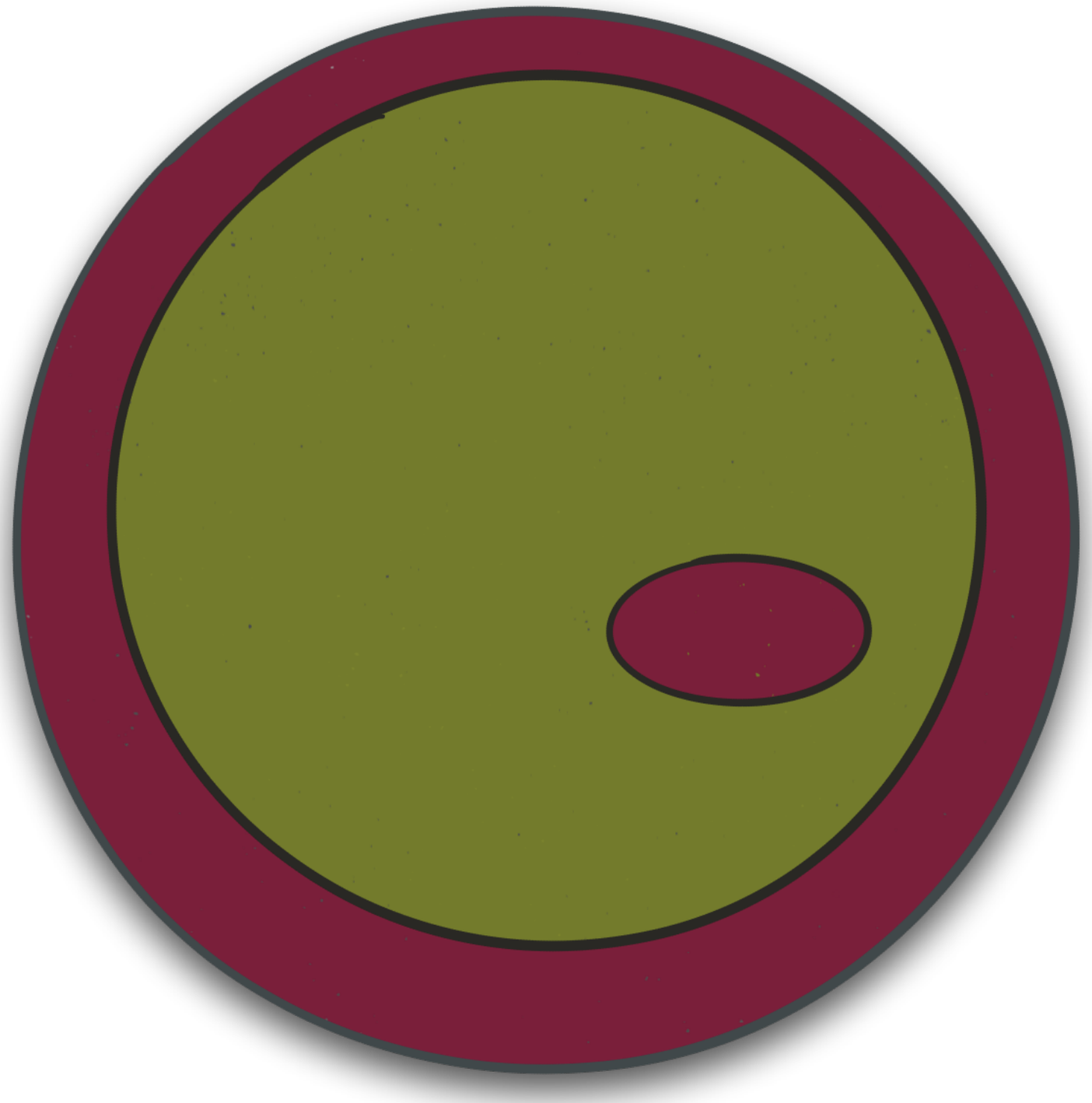
Concrete



Compositional

Entangled

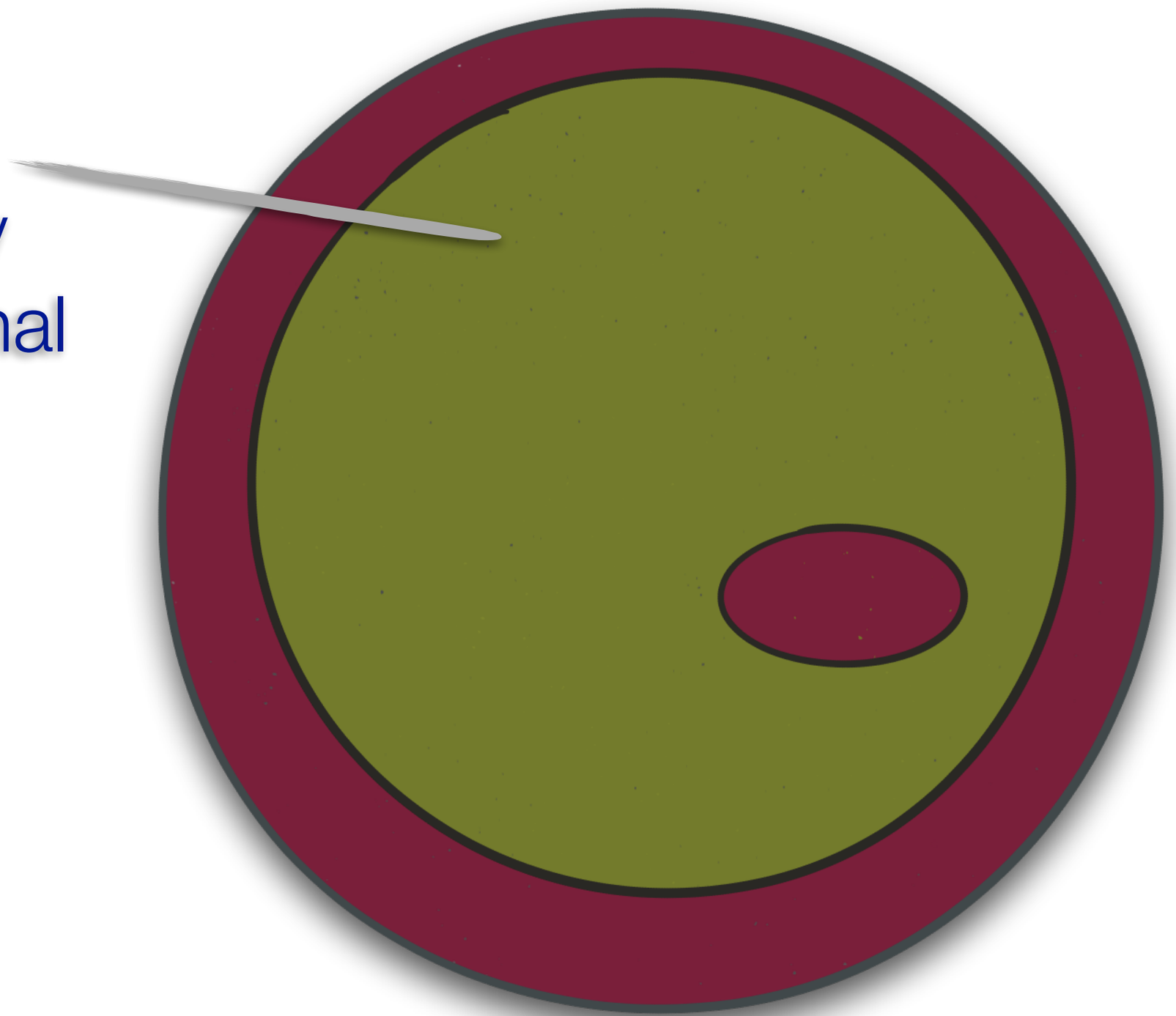
Anatomy of a Haskell Program



Anatomy of a Haskell Program

λ

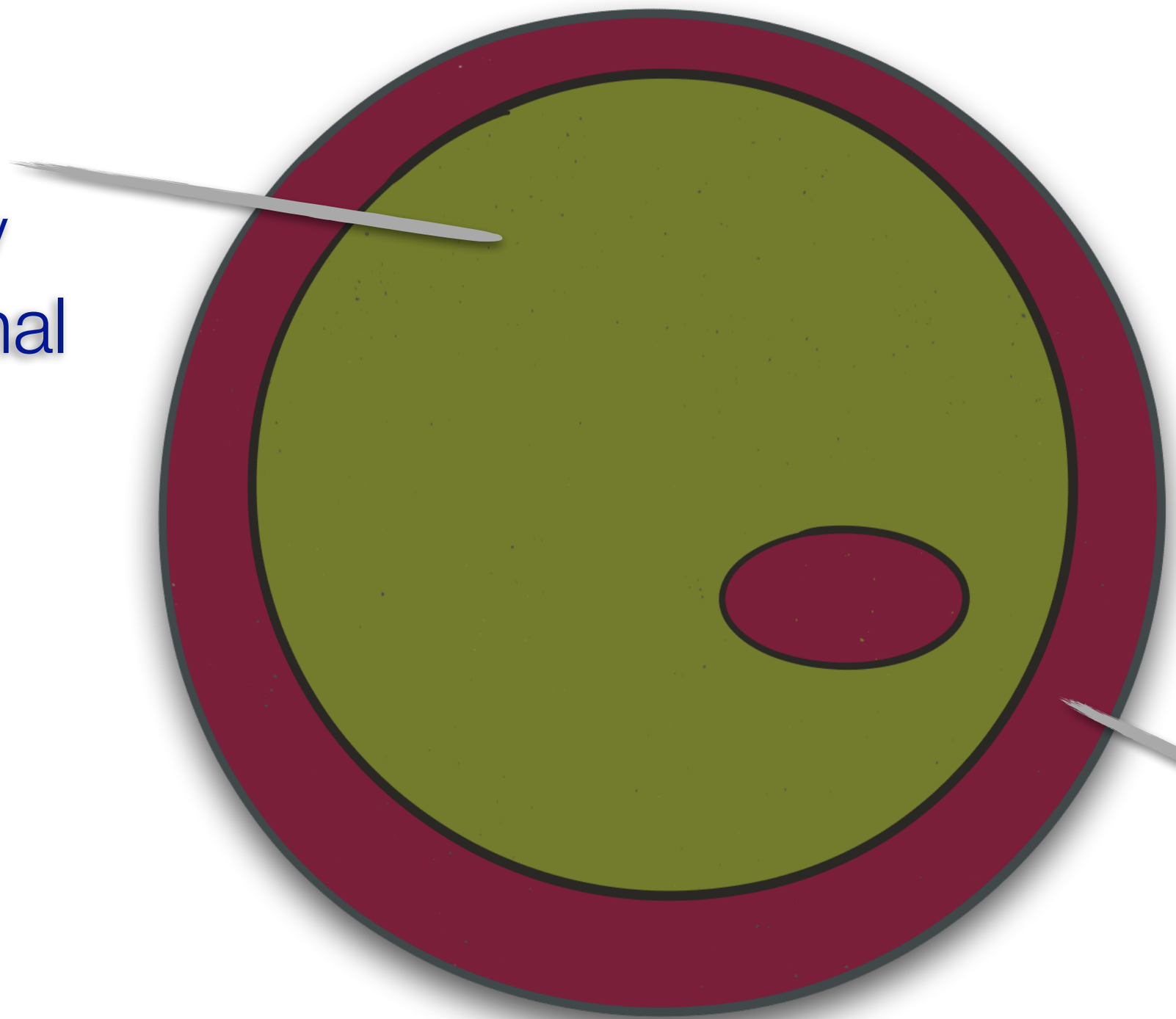
purely functional core

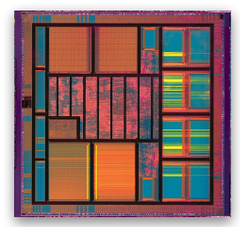


Anatomy of a Haskell Program

λ

purely functional core



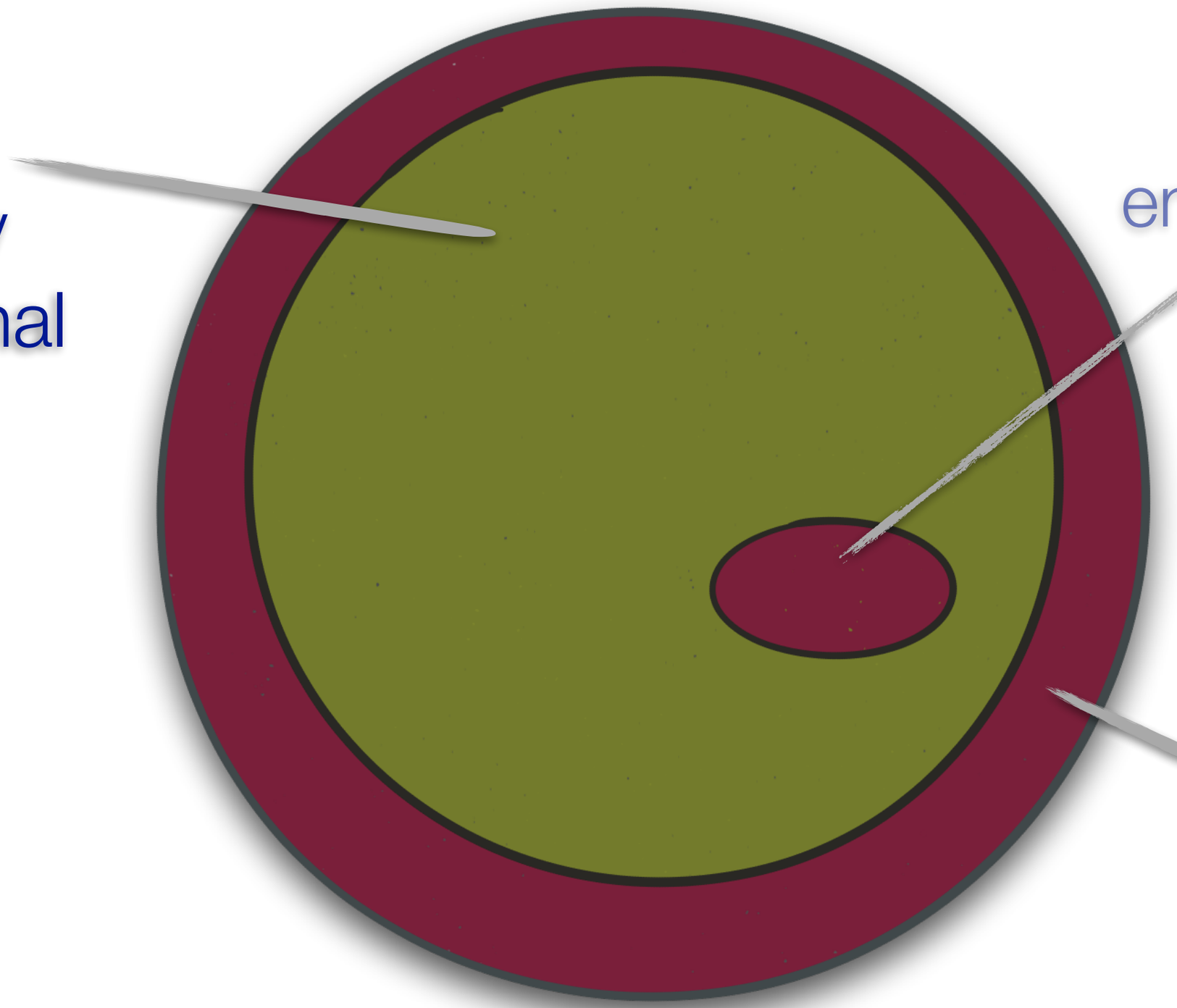


imperative shell

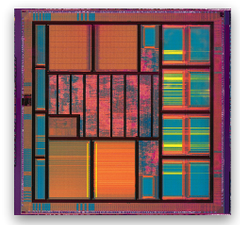
Anatomy of a Haskell Program

λ

purely functional core



encapsulated state



imperative shell

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

Server-side apps

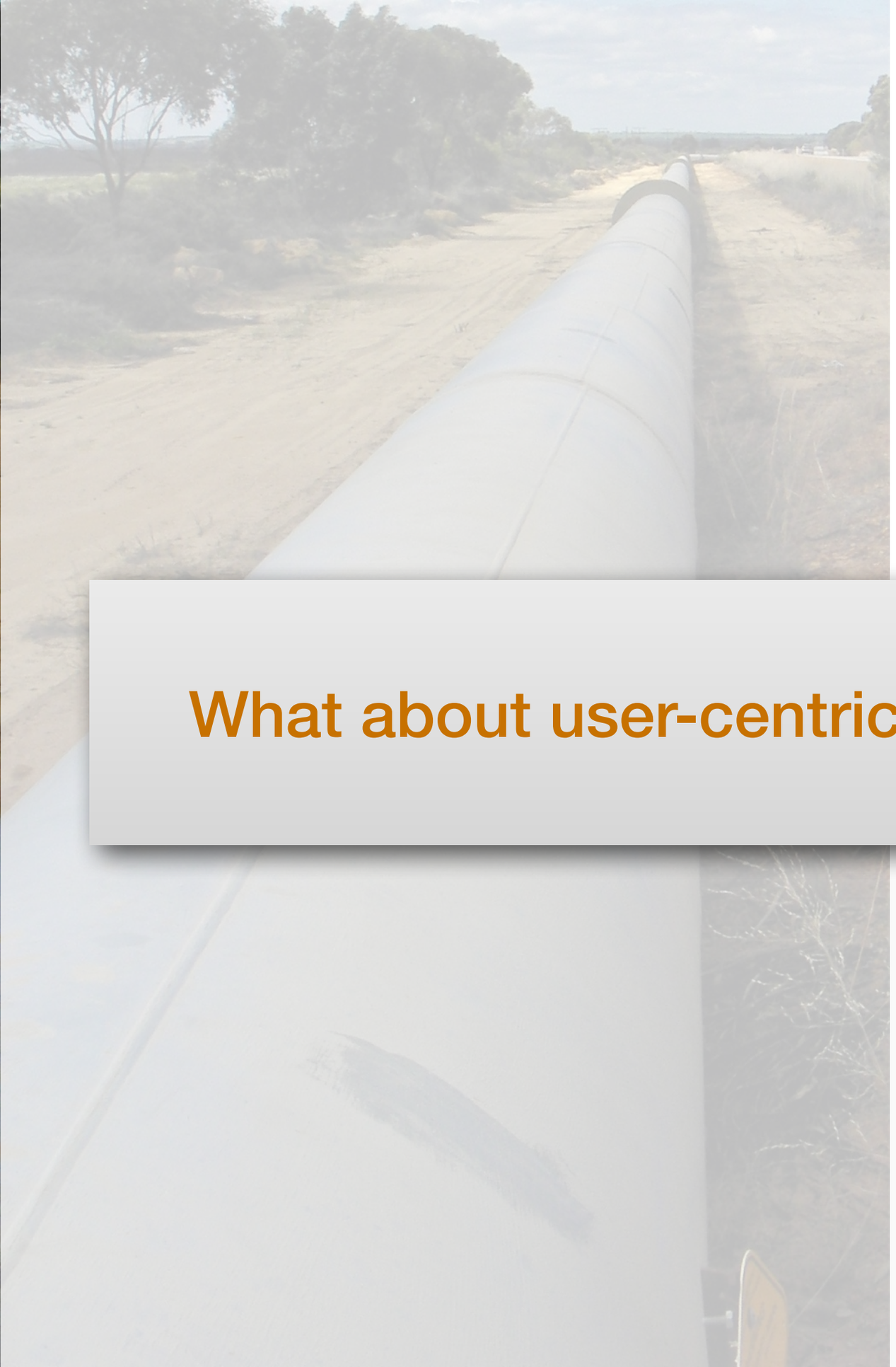
Command-line apps

Clojure

Haskell

OCaml

Scala



Pipeline Applications

Read request or data

Compute

database

What about user-centric & mobile applications?

Clojure

Haskell

Server-side apps

Command-line apps

OCaml

Scala

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

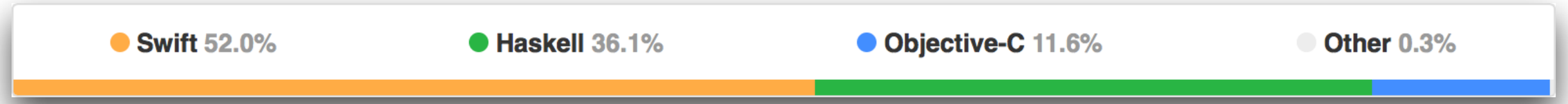

Frequent asynchronous events
Must use OO-based platform libraries
GUI elements have their own derived state

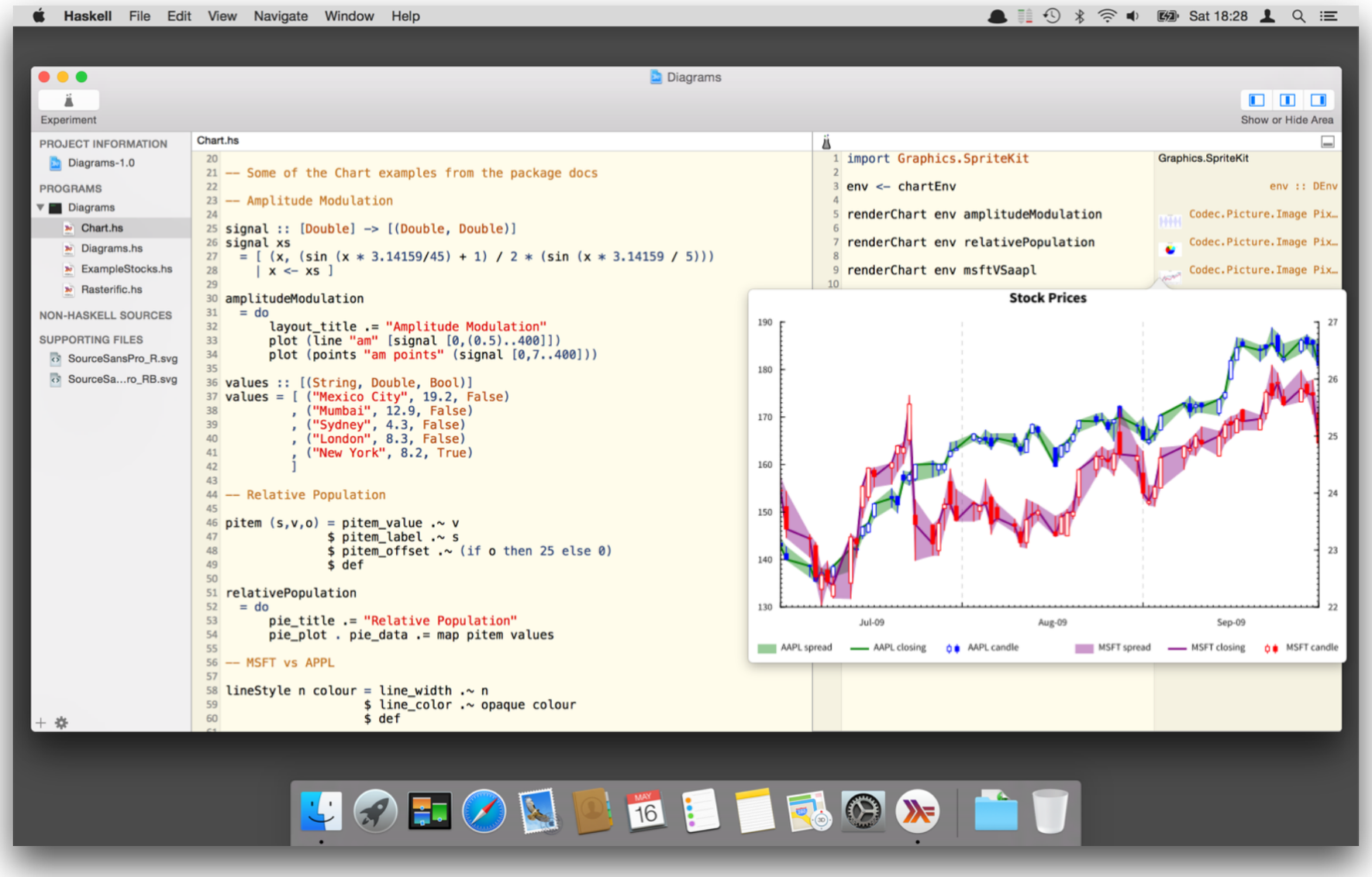

Must be responsive
Frequent asynchronous events
Must use OO-based platform libraries
GUI elements have their own derived state

“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

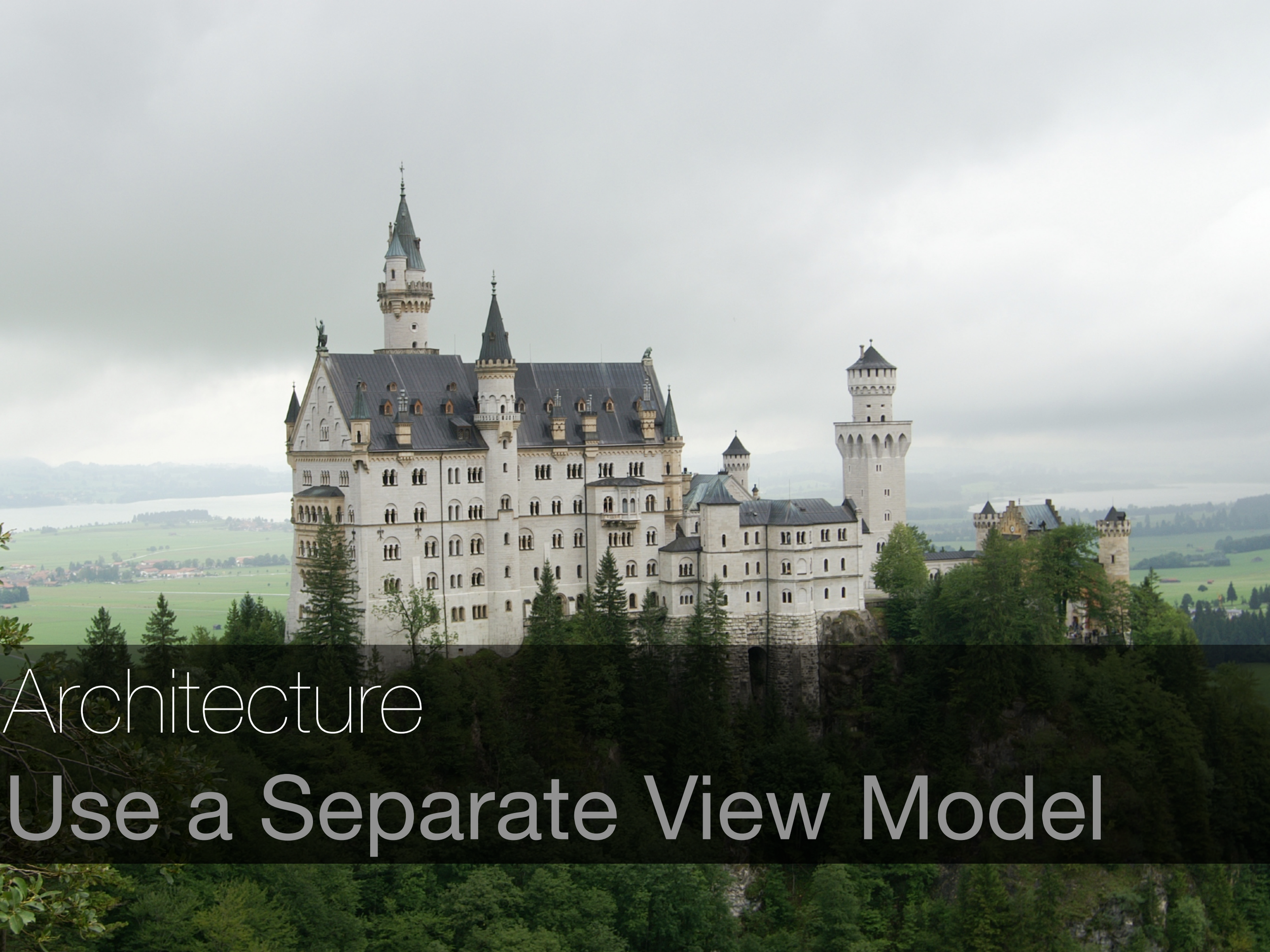
Isolate side effects with a view model

Immutable values

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

Time series

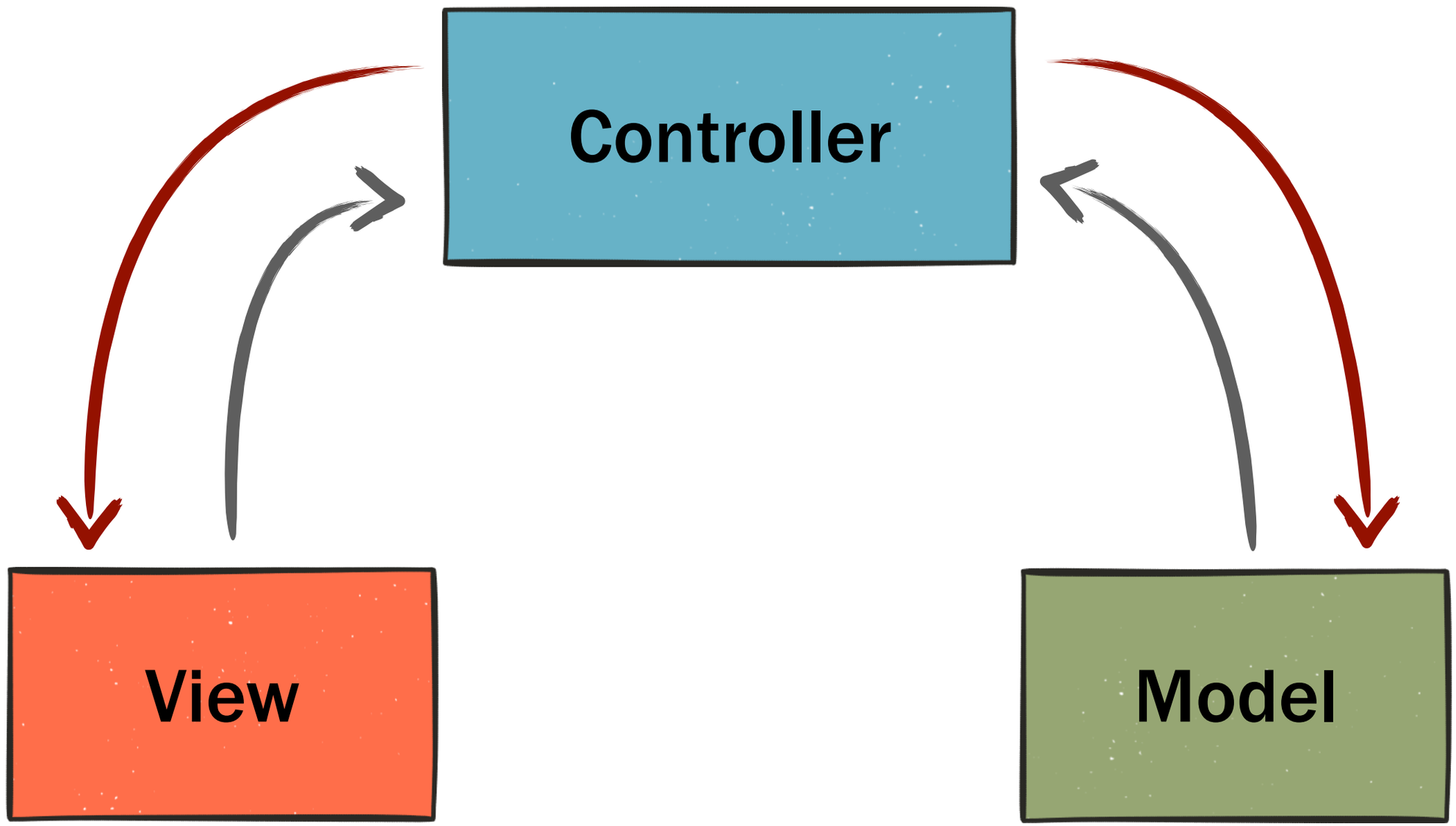
Streams of values over time replace mutable state



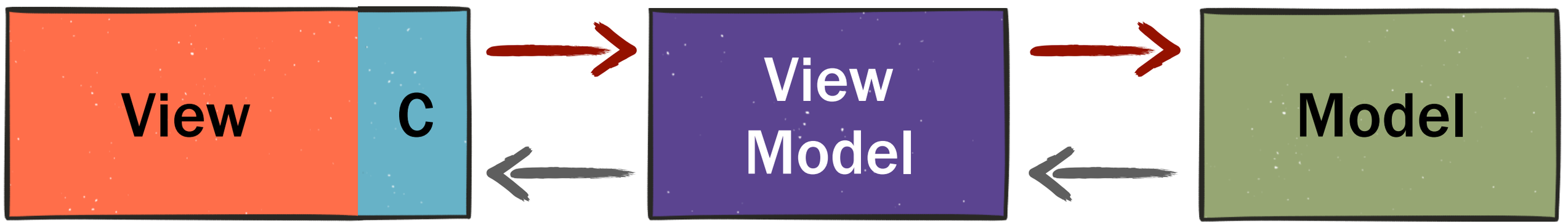
Architecture

Use a Separate View Model

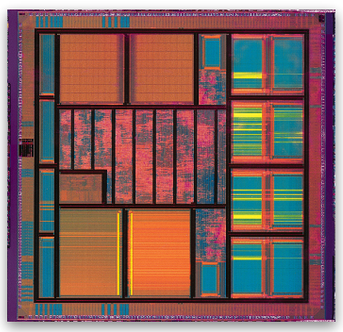
What is a View Model?



What is a View Model?



What is a View Model?

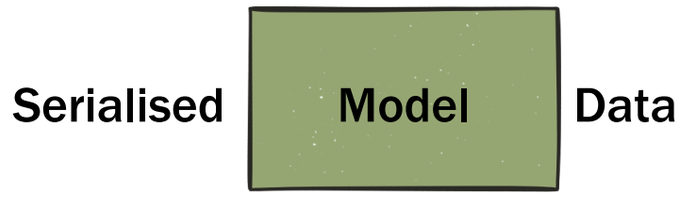


Mutable **Hybrid** **Immutable**

View | C → View Model → Model

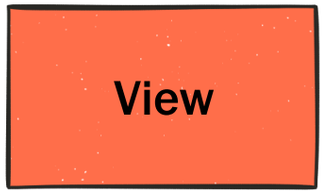
Project Descriptions in Haskell for Mac

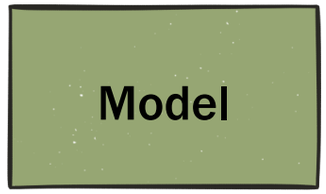
Project Descriptions in Haskell for Mac

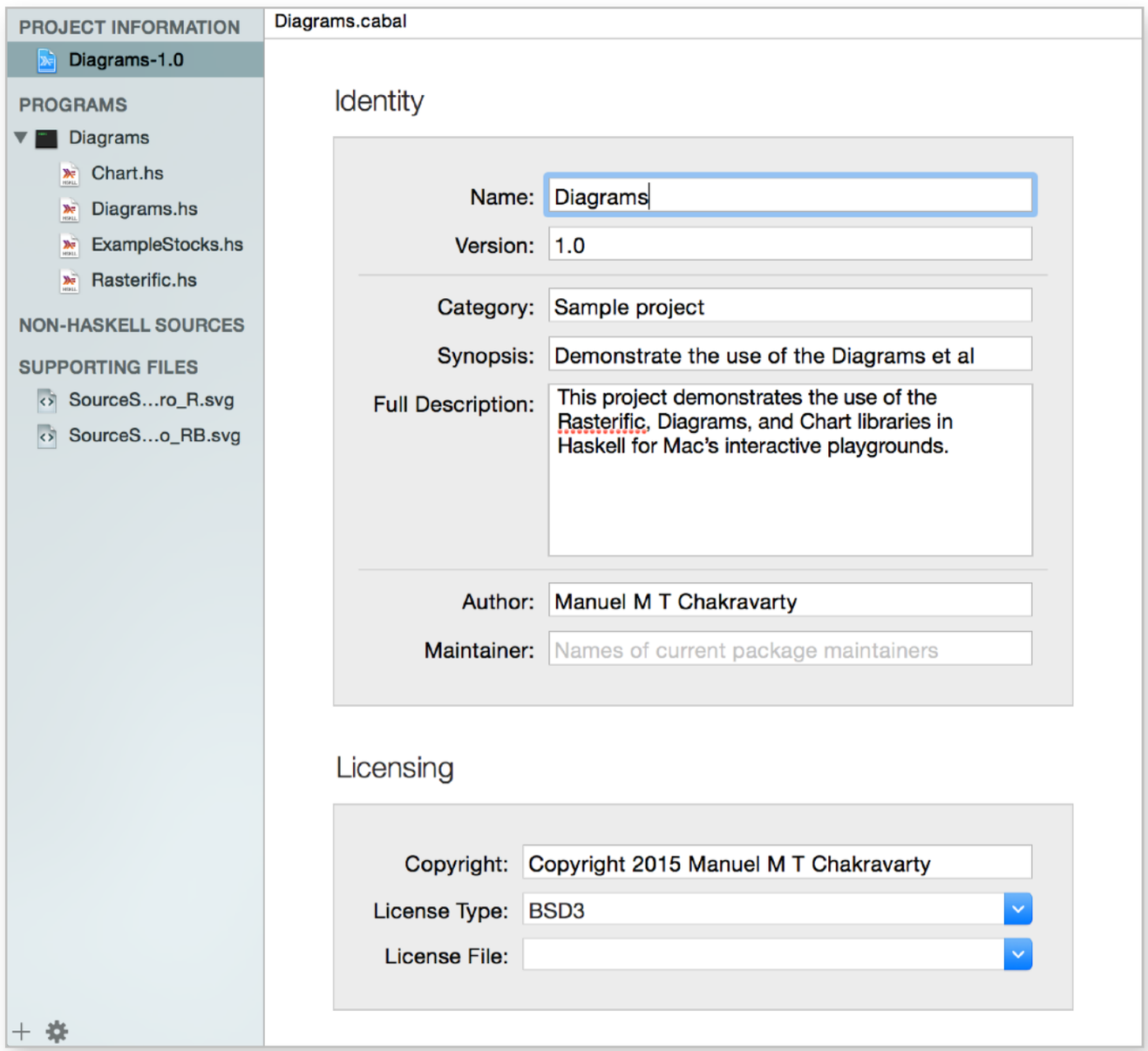



```
name: Diagrams
version: 1.0
cabal-version: 1.6
build-type: Simple
license: BSD3
license-file: ""
copyright: Copyright 2015 Manuel M T Chakravarty
maintainer:
build-depends:
stability: experimental
homepage:
package-url:
bug-reports:
synopsis: Demonstrate the use of the Diagrams et
al
description: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

Project Descriptions in Haskell for Mac



Serialised  Data

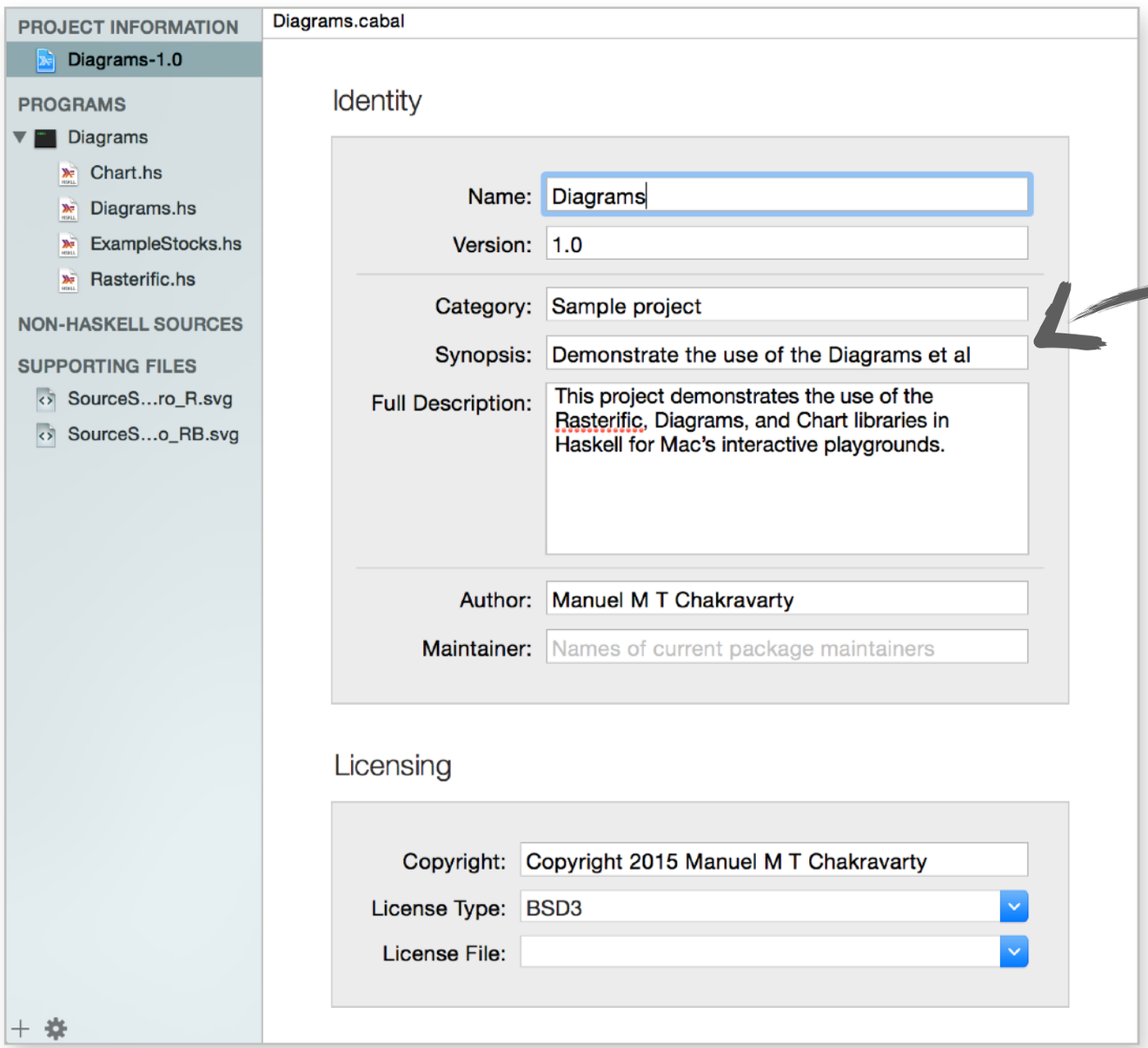



```
name: Diagrams
version: 1.0
cabal-version: 1.6
build-type: Simple
license: BSD3
license-file: ""
copyright: Copyright 2015 Manuel M T Chakravarty
maintainer:
build-depends:
stability: experimental
homepage:
package-url:
bug-reports:
synopsis: Demonstrate the use of the Diagrams et
al
description: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

Project Descriptions in Haskell for Mac

View

Serialised Model Data

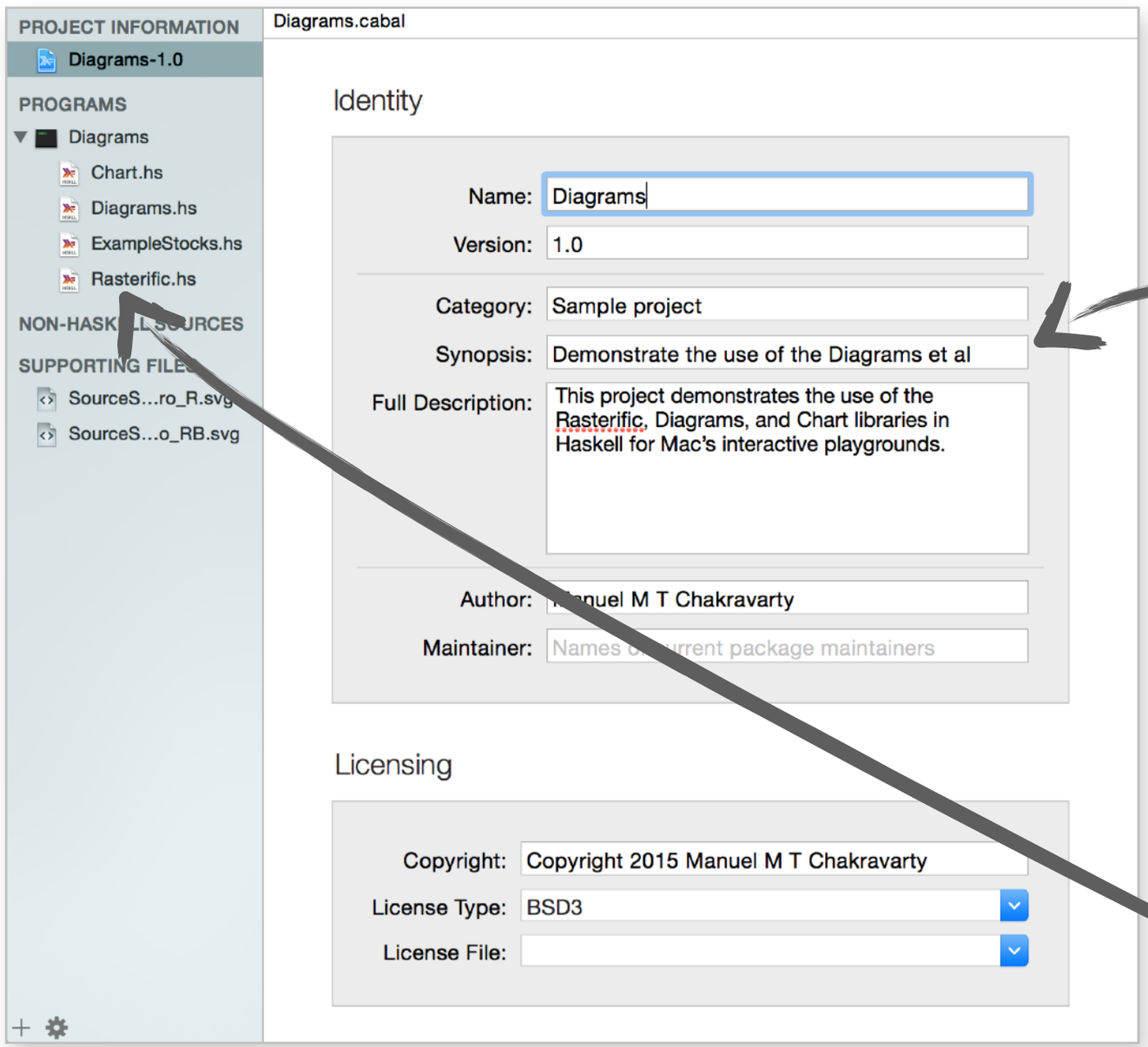



```
1  name: Diagrams
2  version: 1.0
3  cabal-version: 1.6
4  build-type: Simple
5  license: BSD3
6  license-file: ""
7  copyright: Copyright 2015 Manuel M T Chakravarty
8  maintainer:
9  build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate the use of the Diagrams et
   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
```

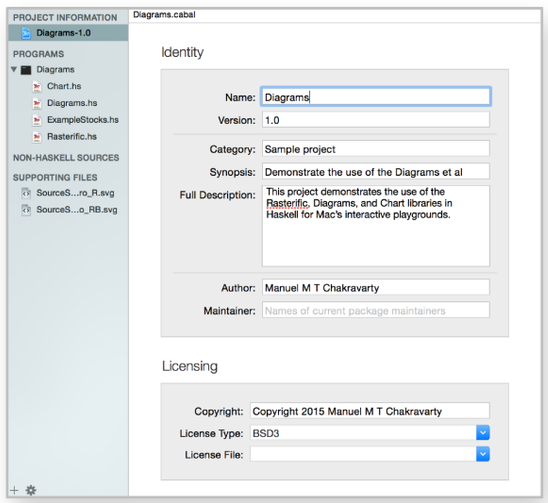
Project Descriptions in Haskell for Mac

View

Serialised **Model** **Data**



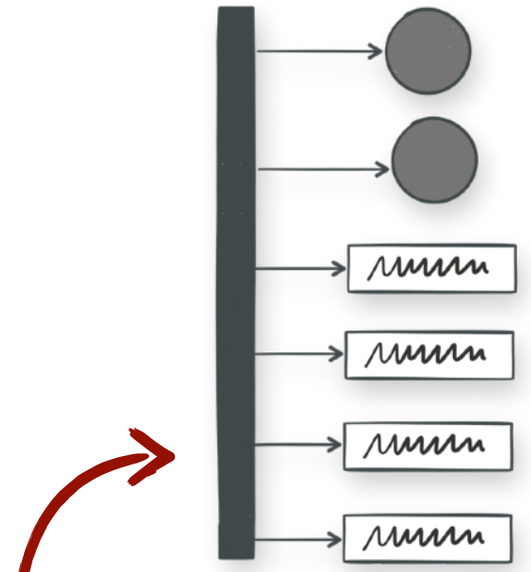

```
1  name: Diagrams
2  version: 1.0
3  cabal-version: 1.6
4  build-type: Simple
5  license: BSD3
6  license-file: ""
7  copyright: Copyright 2015 Manuel M T Chakravarty
8  maintainer:
9  build-depends:
10 stability: experimental
11 homepage:
12 package-url:
13 bug-reports:
14 synopsis: Demonstrate 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
```

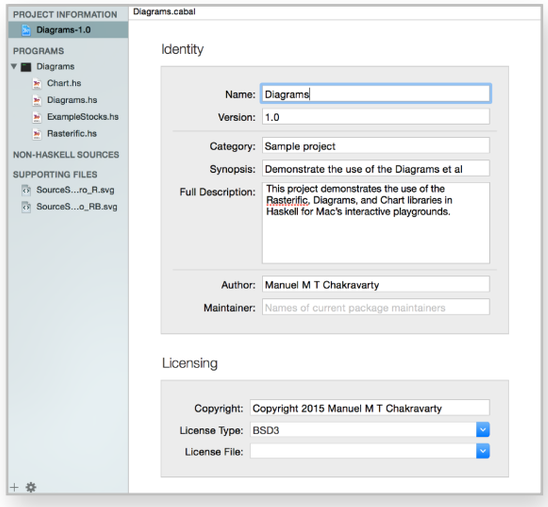

PROJECT INFORMATION
Diagrams-1.0
PROGRAMS
Diagrams
Chart.hs
Diagrams.hs
ExampleStocks.hs
Rasterific.hs
NON-HASKELL SOURCES
SUPPORTING FILES
SourceS…ro_R.svg
SourceS…o_RB.svg
Diagrams.cabal
Identity
Name: Diagrams
Version: 1.0
Category: Sample project
Synopsis: Demonstrate the use of the Diagrams et al
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:


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

Model

Immutable



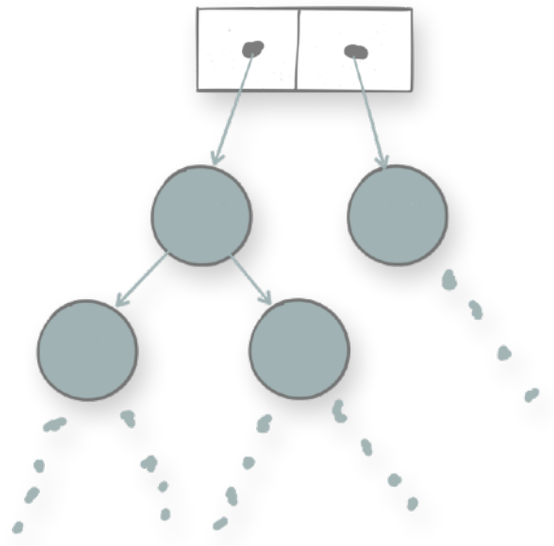


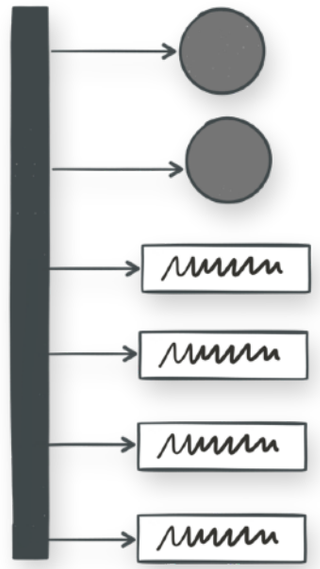

```
name: Diagrams
version: 1.0
cabal-version: 1.6
build-type: Simple
license: BSD3
license-file: ""
copyright: Copyright 2015 Manuel M T Chakravarty
maintainer:
build-depends:
stability: experimental
homepage:
package-url:
bug-reports:
synopsis: Demonstrate the use of the Diagrams et
al
description: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

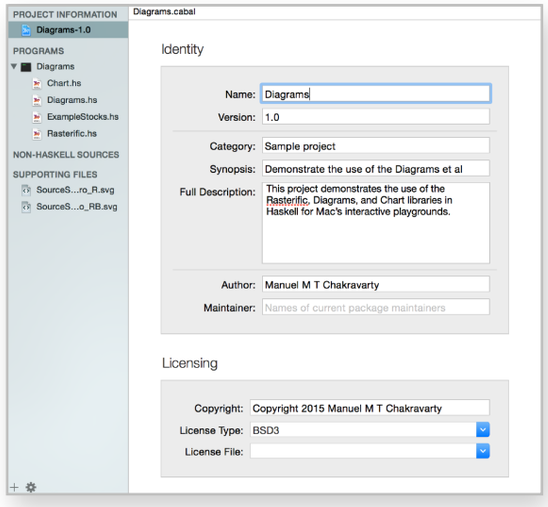
View Model

Model

Immutable







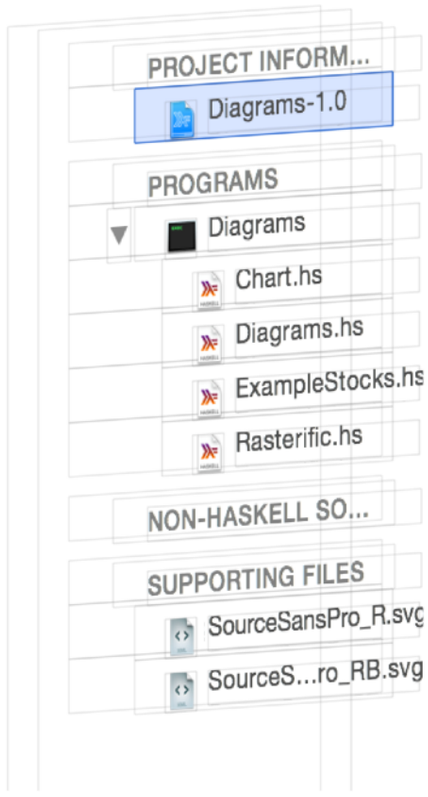

```
name: Diagrams
version: 1.0
cabal-version: 1.6
build-type: Simple
license: BSD3
license-file: ""
copyright: Copyright 2015 Manuel M T Chakravarty
maintainer:
build-depends:
stability: experimental
homepage:
package-url:
bug-reports:
synopsis: Demonstrate the use of the Diagrams et
al
description: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

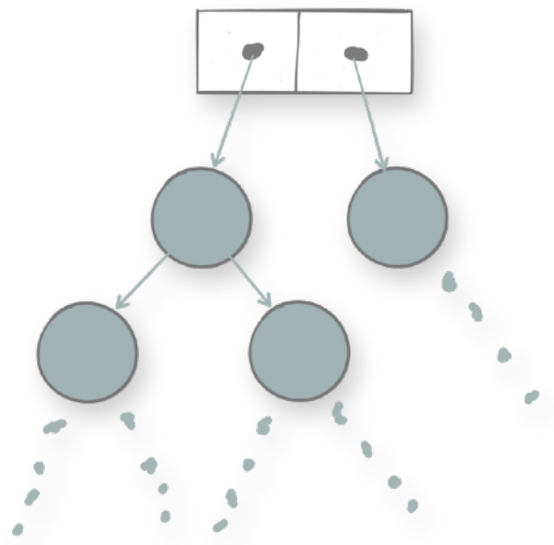
View | C

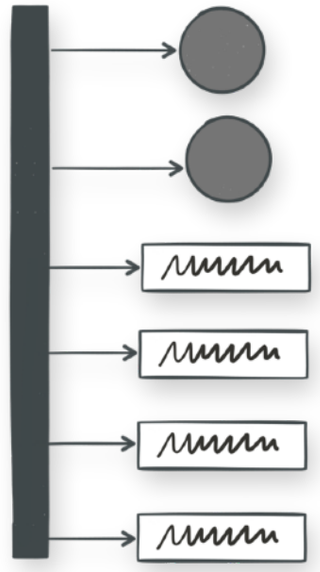
View Model

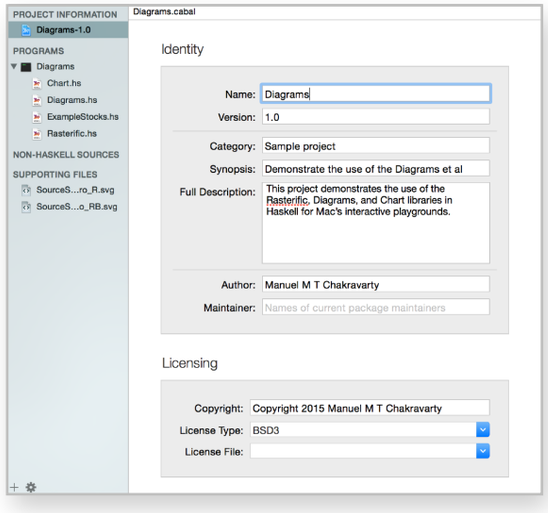
Model

Immutable

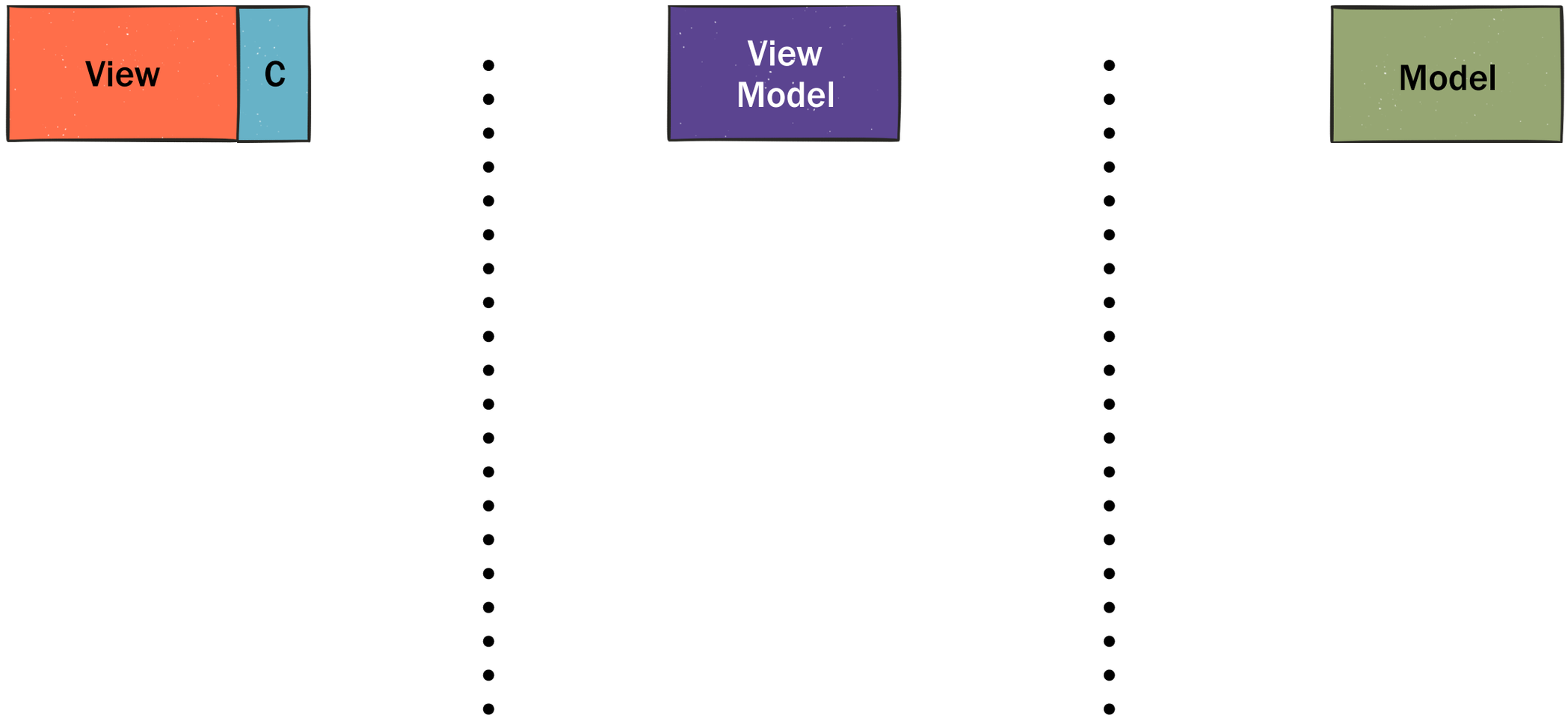






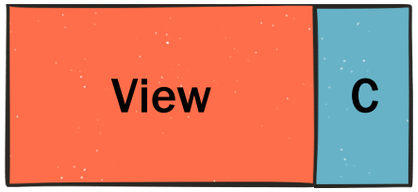


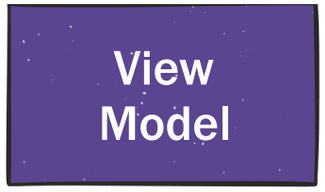

```
name: Diagrams
version: 1.0
cabal-version: 1.6
build-type: Simple
license: BSD3
license-file: ""
copyright: Copyright 2015 Manuel M T Chakravarty
maintainer:
build-depends:
stability: experimental
homepage:
package-url:
bug-reports:
synopsis: Demonstrate the use of the Diagrams et
al
description: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

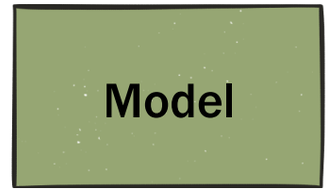

View
C
View
Model
Model

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,
    …
```








```haskell
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) }
 }
 …
```

```
n
{
ckageIdentifier,
cense,
ilePath],
ring,
ring,
ring,
ring,

,VersionRange)],
ring,
ring,
ring,
ourceRepo],
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) }
 }
 …
```

```
n
{
ckageIdentifier,
cense,
ilePath],
ring,
ring,
ring,
ring,

,VersionRange)],
ring,
ring,
ring,
ourceRepo],
ring,
ring,
ring,
```

View

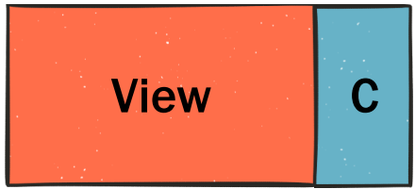
```
struct Package {
 private var modelPackage: CBLPackage
 …
 var name: String {
  get { return modelPackage.name }
  set { modelPackage =
          CBLPackage(modelPackage,
                     withName: newValue) }
 }
 …
 var richTextDesc: NSAttributedString{
  get {
    return NSAttributedString(string: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

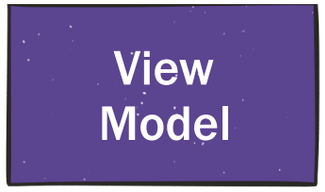
```
n
{
ckageIdentifier,
cense,
ilePath],
ring,
ring,
ring,
ring,

,VersionRange)],
ring,
ring,
ring,
ourceRepo],
ring,
ring,
ring,
```

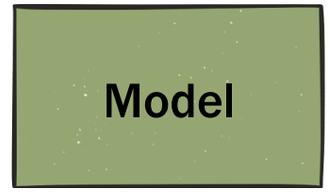
View

```
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) }
 }
 …
```

```
n
{
ckageIdentifier,
cense,
ilePath],
ring,
ring,
ring,
ring,

,VersionRange)],
ring,
ring,
ring,
ourceRepo],
ring,
ring,
ring,
```

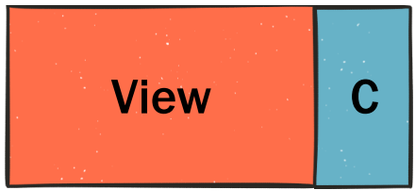


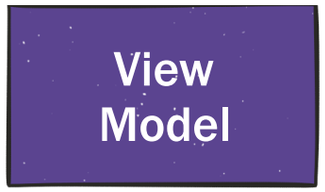


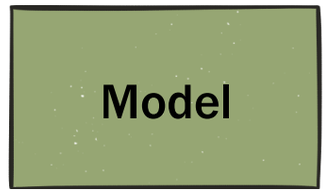



```
struct Package {
 private var modelPackage: CBLPackage
 …
 var name: String {
  get { return modelPackage.name }
  set { modelPackage =
          CBLPackage(modelPackage,
                     withName: newValue) }
 }
 …
 var richTextDesc: NSAttributedString{
  get {
    return NSAttributedString(string: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

```
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,
    …
```



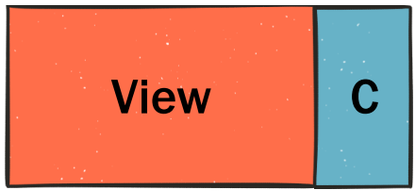


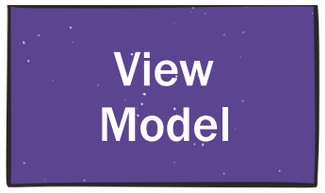


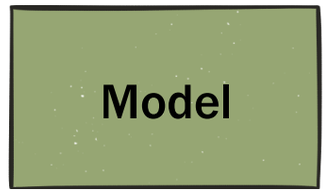

```
struct Package {
 private var modelPackage: CBLPackage
```

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

```
n
{
ckageIdentifier,
cense,
ilePath],
ring,
ring,
ring,

,VersionRange)],
ring,
ring,
ring,
ourceRepo],
ring,
ring,
ring,
```



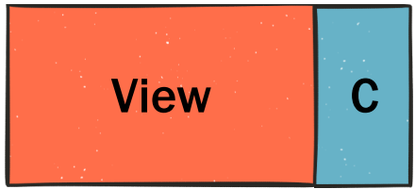


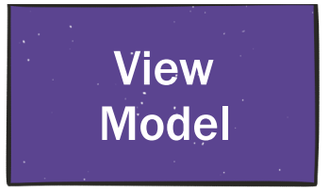


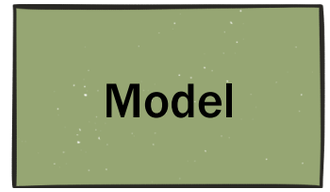

```
struct Package {
 private var modelPackage: CBLPackage
```

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

```
n
{
ckageIdentifier,
cense,
ilePath],
ring,
ring,
ring,

,VersionRange)],
ring,
ring,
ring,
ourceRepo],
ring,
ring,
ring,
```



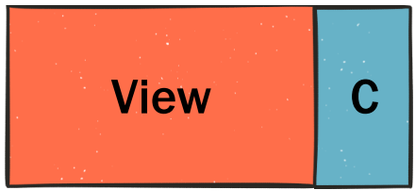


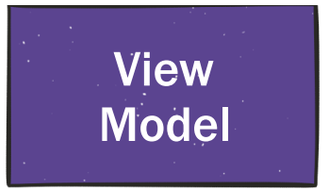


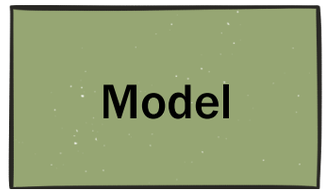

```
struct Package {
 private var modelPackage: CBLPackage
 …
 var name: String {
  get { return modelPackage.name }
  set { modelPackage =
          CBLPackage(modelPackage,
                     withName: newValue) }
 }
 …
 var richTextDesc: NSAttributedString{
  get {
    return NSAttributedString(string: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

```
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: CBLPackage
```

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

```
n
{
ckageIdentifier,
cense,
ilePath],
ring,
ring,
ring,

,VersionRange)],
ring,
ring,
ring,
ourceRepo],
ring,
ring,
ring,
```

```
}
```

View | C

View Model

Model

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

```
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: NSFileWrapper?
  …
}
```

```
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

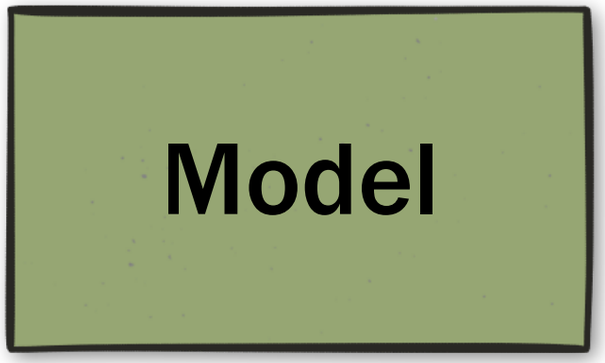
Sometimes performance requires mutability

Beware of premature optimisation!

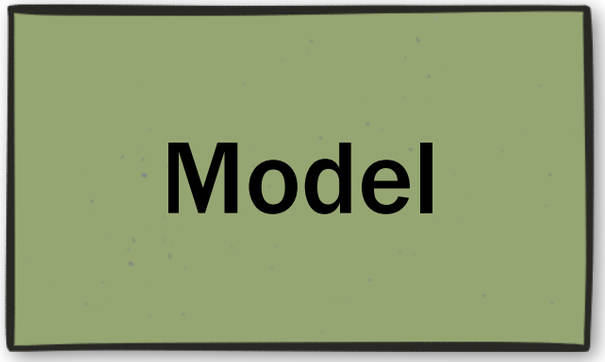
Sometimes an existing API is based on mutability

Can you wrap the mutable API?

Immutable

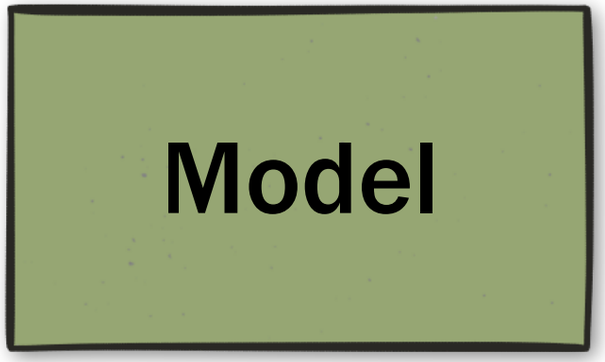


Immutable



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

Immutable



✓ Cabal package in Haskell

✓ Source files as `NSFileWrapper`

Directories are not explicit in the model

Immutable Model

✓ Cabal package in Haskell

✓ Source files as `NSFileWrapper`

Directories are not explicit in the model

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: NSFileWrapper?
  …
}
```

Time Series

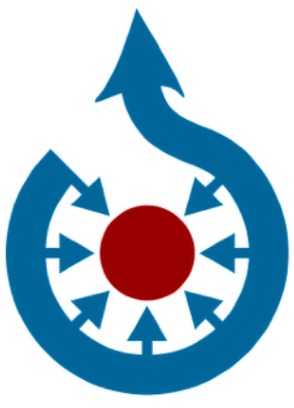
Changes as Streams of Values

Three Principles for FP in a Stateful World

haskellformac.com
mchakravarty
TacticalGrace
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:NeuSchwanstein_7.JPG

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

http://commons.wikimedia.org/wiki/File:Stopwatch_A.jpg

dribbble

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