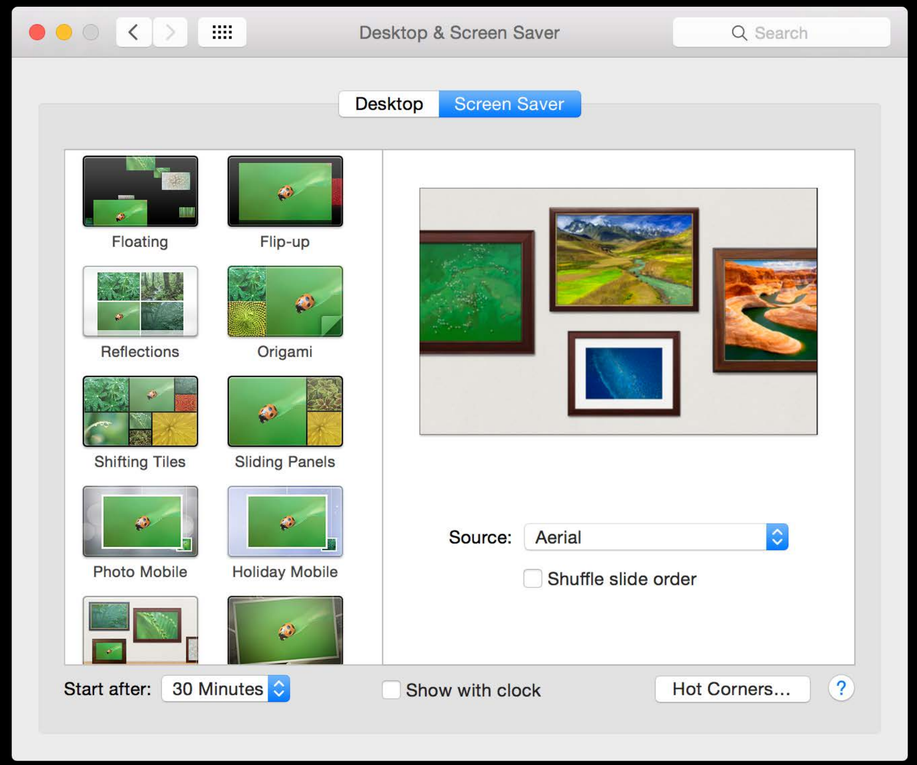
App Frameworks #WWDC15

What's New in NSCollectionView

Session 225

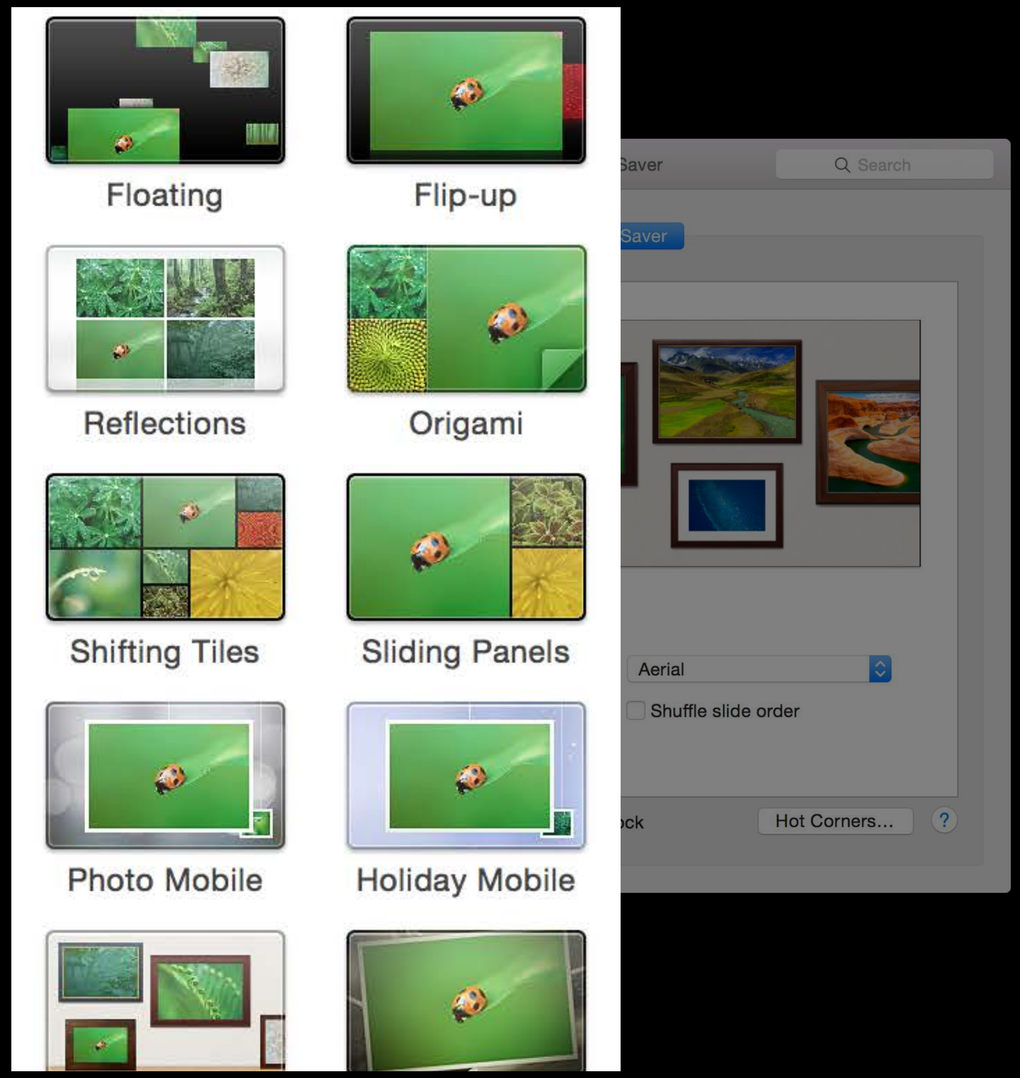
Troy Stephens Application Frameworks Engineer

NSCollectionView



NSCollectionView

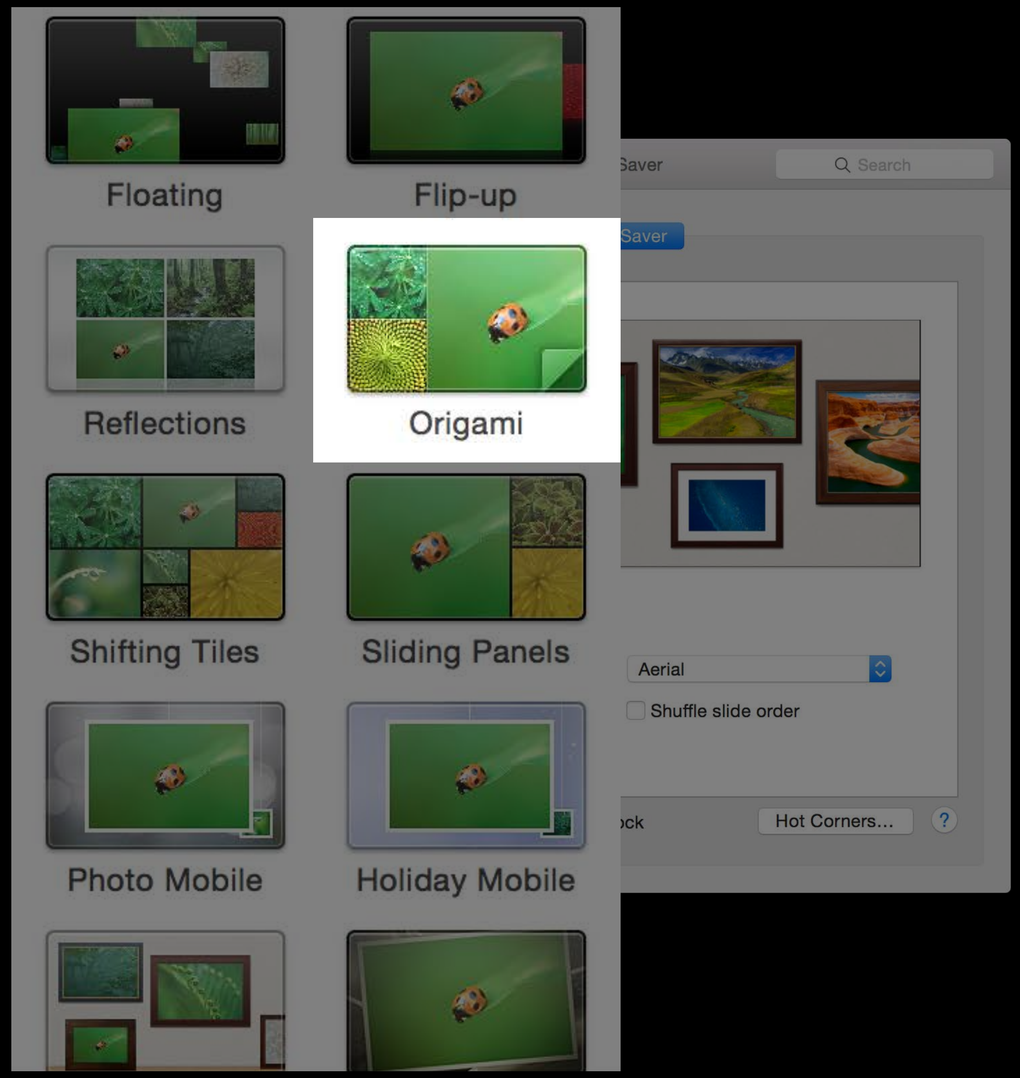
Displays grids of identically sized items



NSCollectionView

Displays grids of identically sized items

Each item cloned from an “itemPrototype”

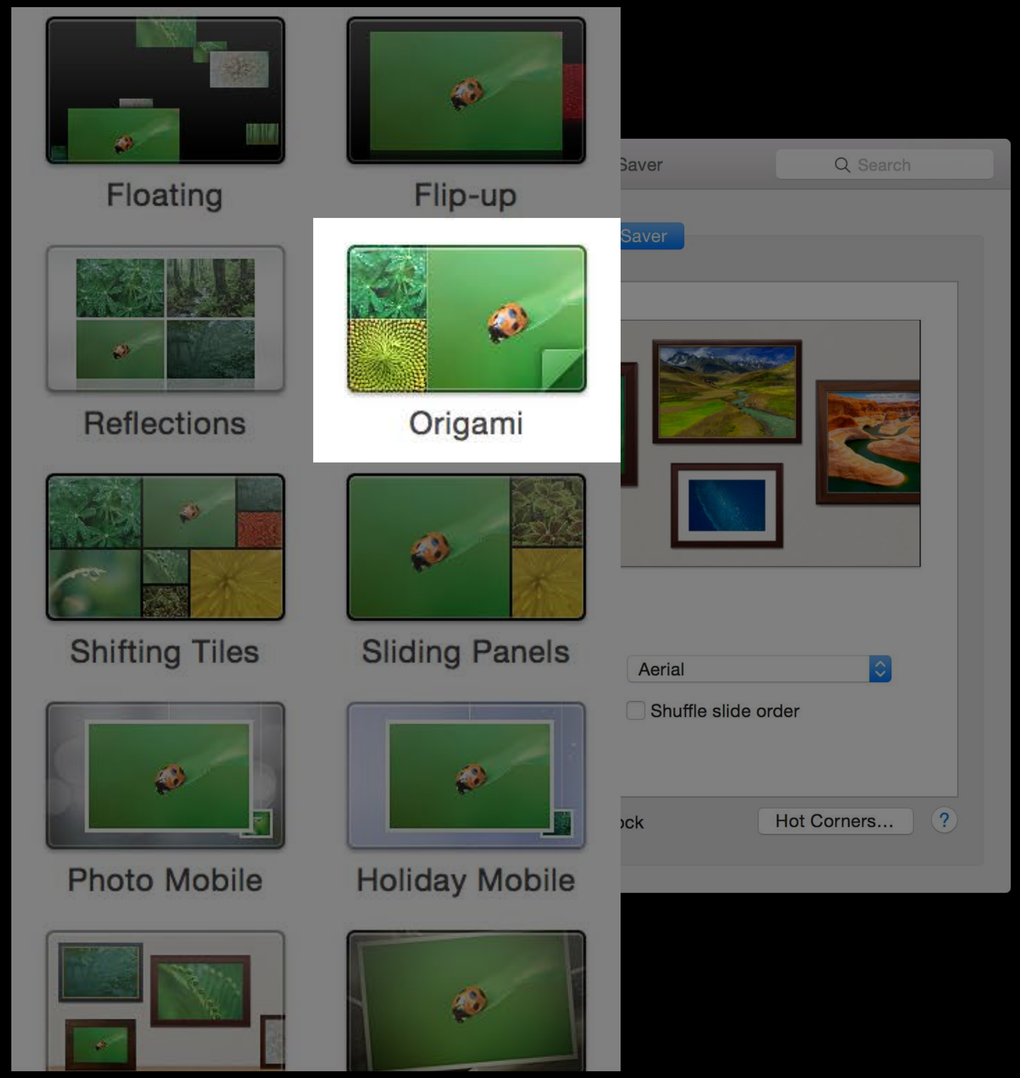


NSCollectionView

Displays grids of identically sized items

Each item cloned from an “itemPrototype”

- ViewController + View subtree



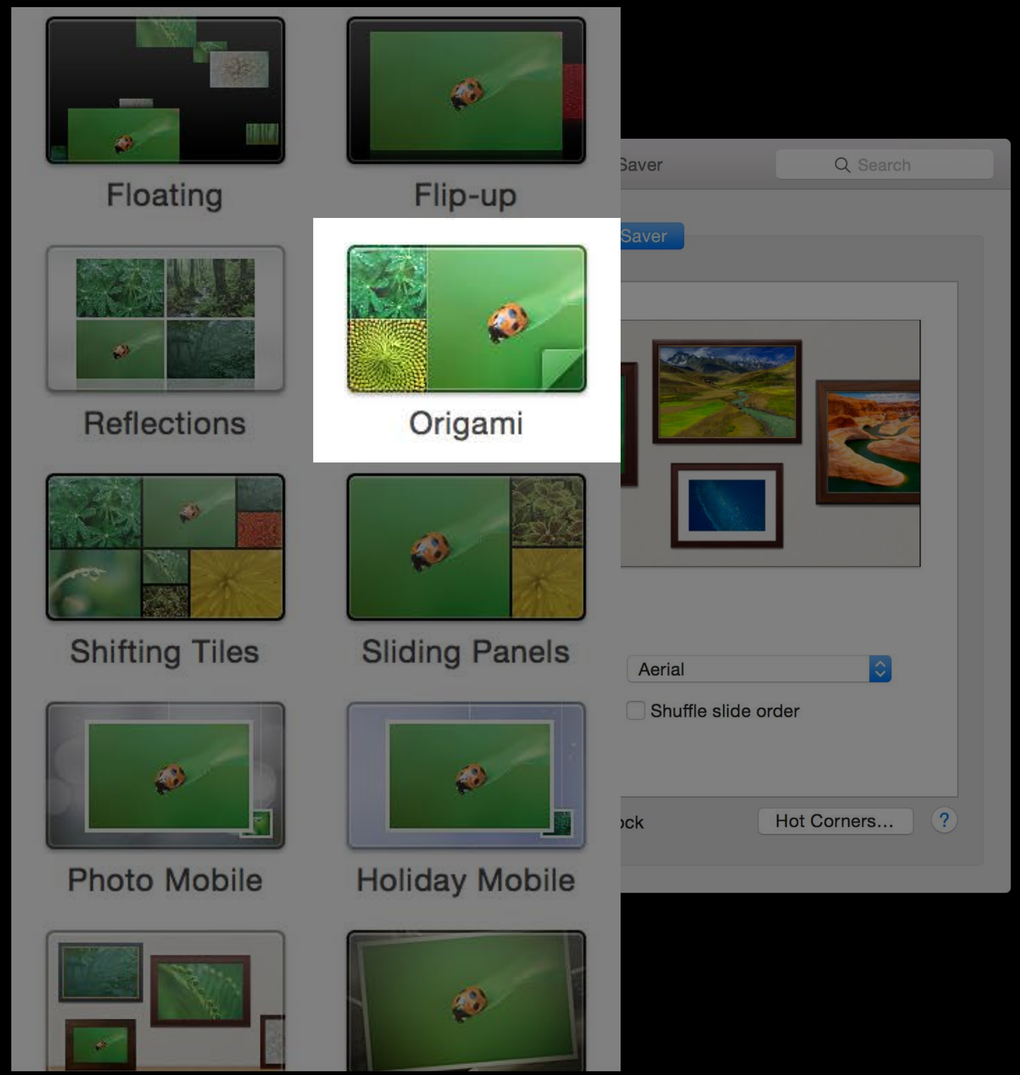
NSCollectionView

Displays grids of identically sized items

Each item cloned from an “itemPrototype”

- ViewController + View subtree

Supports selection, drag-and-drop, animated relayout



iPad 9:41 AM 100%

Edit

World Clock

Cupertino
Today, 3 hours behind

New York
Today

Paris
Today, 6 hours ahead

Beijing
Today, 12 hours ahead

Tokyo
Tomorrow, 13 hours ahead

Moscow
Today, 7 hours ahead

World Clock Alarm Stopwatch Timer

iPad 9:41 AM 100%

Edit

World Clock

Cupertino
Today, 3 hours behind

New York
Today

Paris
Today, 6 hours ahead

Beijing
Today, 12 hours ahead

Tokyo
Tomorrow, 13 hours ahead

Moscow
Today, 7 hours ahead

8:25 AM Cupertino 55°
11:25 AM New York 73°
5:25 PM Paris 64°
6:25 PM Moscow 64°
11:25 PM Beijing 61°
12:25 AM Tokyo 64°

The Weather Channel

World Clock | Alarm | Stopwatch | Timer

UICollectionView

UICollectionView

Displays collections of items

UICollectionView

Displays collections of items

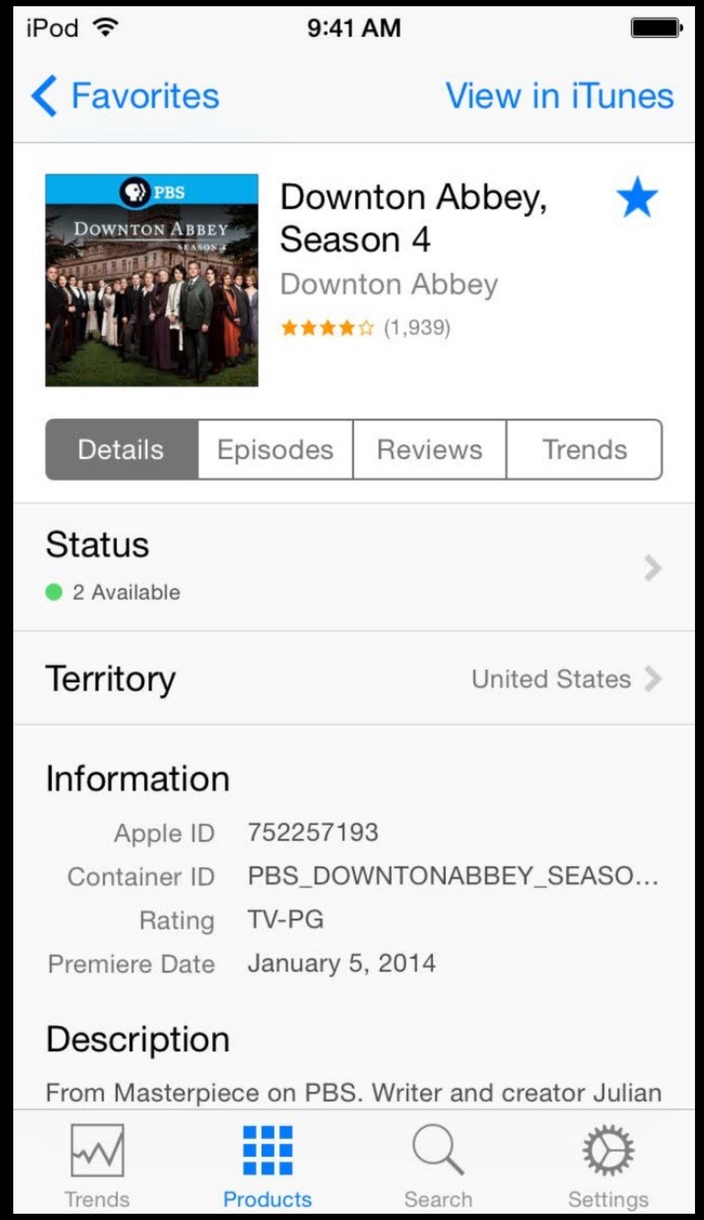
Each item created from a .nib or view class

UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

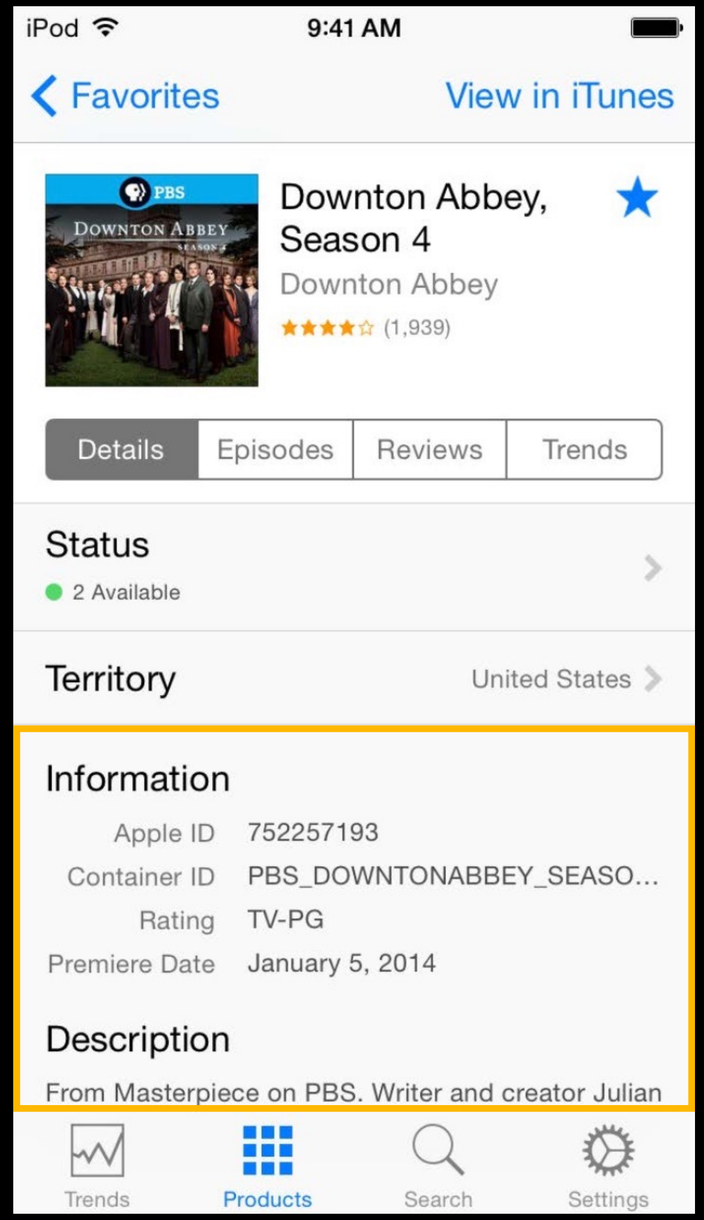


UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types



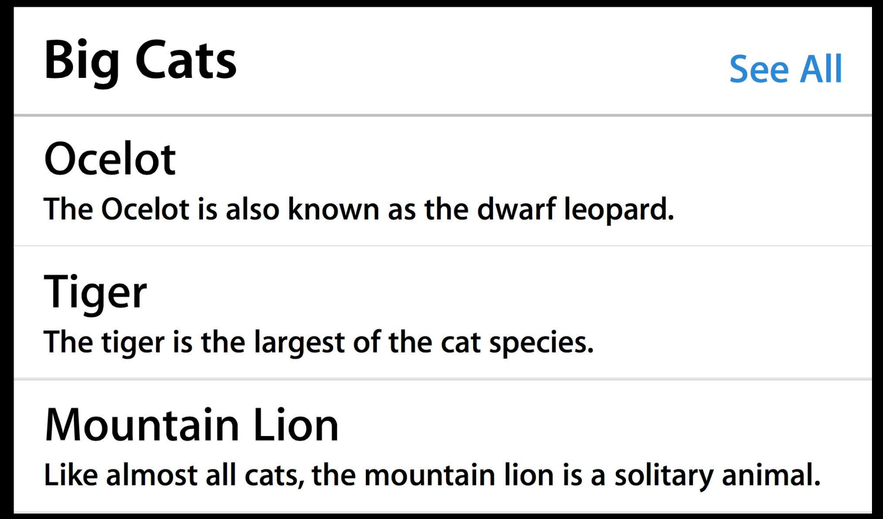
UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

Headers, footers



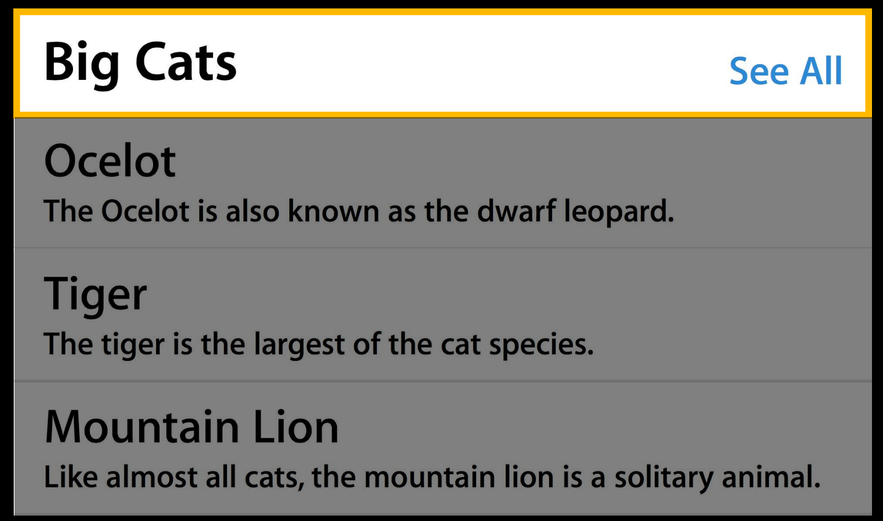
UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

Headers, footers



UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

Headers, footers

Sections

UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

Headers, footers

Sections

Flexible, customizable “Flow” layout

UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

Headers, footers

Sections

Flexible, customizable “Flow” layout

Developer-definable layouts

UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

Headers, footers

Sections

Flexible, customizable “Flow” layout

Developer-definable layouts

Scalable to large numbers of items

UICollectionView

Displays collections of items

Each item created from a .nib or view class

Can mix item types

Headers, footers

Sections

Flexible, customizable “Flow” layout

Developer-definable layouts

Scalable to large numbers of items

Layer-backed

A Better NSCollectionView

NEW

A Better NSCollectionView

NEW

Scalable

A Better NSCollectionView

NEW

Scalable

Optional sectioning, with header/footer views

A Better NSCollectionView

NEW

Scalable

Optional sectioning, with header/footer views

Completely customizable layout

A Better NSCollectionView

NEW

Scalable

Optional sectioning, with header/footer views

Completely customizable layout

Heterogeneous and variable-sized items

A Better NSCollectionView

NEW

Scalable

Optional sectioning, with header/footer views

Completely customizable layout

Heterogeneous and variable-sized items

Customizable appearance

A Better NSCollectionView

NEW

Scalable

Optional sectioning, with header/footer views

Completely customizable layout

Heterogeneous and variable-sized items

Customizable appearance

Animation control

OS X Adaptations

NEW

OS X Adaptations

NEW

Drag and Drop, including Layout-based drop target determination and indication

OS X Adaptations

NEW

Drag and Drop, including Layout-based drop target determination and indication

Rubberband drag-select

OS X Adaptations

NEW

Drag and Drop, including Layout-based drop target determination and indication

Rubberband drag-select

Bulk selection and highlight notifications

OS X Adaptations

NEW

Drag and Drop, including Layout-based drop target determination and indication

Rubberband drag-select

Bulk selection and highlight notifications

Items still represented using ViewControllers (NSCollectionViewItem)

OS X Adaptations

NEW

Drag and Drop, including Layout-based drop target determination and indication

Rubberband drag-select

Bulk selection and highlight notifications

Items still represented using ViewControllers (NSCollectionViewItem)

Automatic nib/class finding, via naming conventions

Today's Goals

Today's Goals

Learn how to wire up and use a new NSCollectionView

Today's Goals

Learn how to wire up and use a new NSCollectionView

Learn what's different on OS X vs. iOS

Today's Goals

Learn how to wire up and use a new NSCollectionView

Learn what's different on OS X vs. iOS

A little something for everybody

Today's Goals

Learn how to wire up and use a new NSCollectionView

Learn what's different on OS X vs. iOS

A little something for everybody

- iOS developers bringing companion apps to OS X

Today's Goals

Learn how to wire up and use a new NSCollectionView

Learn what's different on OS X vs. iOS

A little something for everybody

- iOS developers bringing companion apps to OS X
- OS X developers wanting to leverage the new capabilities

Today's Goals

Learn how to wire up and use a new NSCollectionView

Learn what's different on OS X vs. iOS

A little something for everybody

- iOS developers bringing companion apps to OS X
- OS X developers wanting to leverage the new capabilities
- Those new to all of this

The Plan

The Plan

Overview

The Plan

Overview

Nuts and bolts

The Plan

Overview

Nuts and bolts

Conclusion

Overview

Old API

Old API

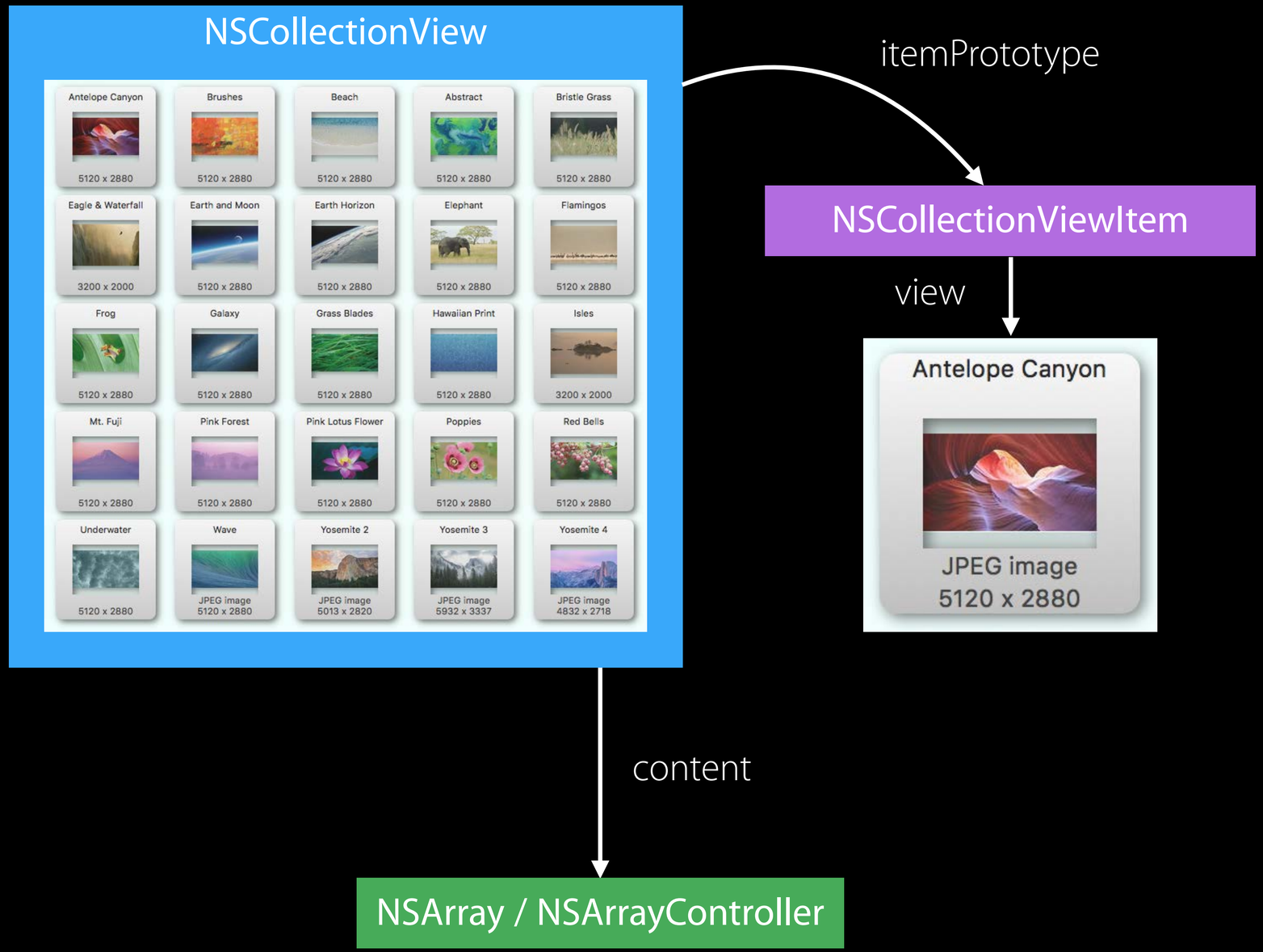

NSCollectionView
Antelope Canyon
5120 x 2880
Brushes
5120 x 2880
Beach
5120 x 2880
Abstract
5120 x 2880
Bristle Grass
5120 x 2880
Eagle & Waterfall
3200 x 2000
Earth and Moon
5120 x 2880
Earth Horizon
5120 x 2880
Elephant
5120 x 2880
Flamingos
5120 x 2880
Frog
5120 x 2880
Galaxy
5120 x 2880
Grass Blades
5120 x 2880
Hawaiian Print
5120 x 2880
Isles
3200 x 2000
Mt. Fuji
5120 x 2880
Pink Forest
5120 x 2880
Pink Lotus Flower
5120 x 2880
Poppies
5120 x 2880
Red Bells
5120 x 2880
Underwater
5120 x 2880
Wave
JPEG image
5120 x 2880
Yosemite 2
JPEG image
5013 x 2820
Yosemite 3
JPEG image
5932 x 3337
Yosemite 4
JPEG image
4832 x 2718

Old API

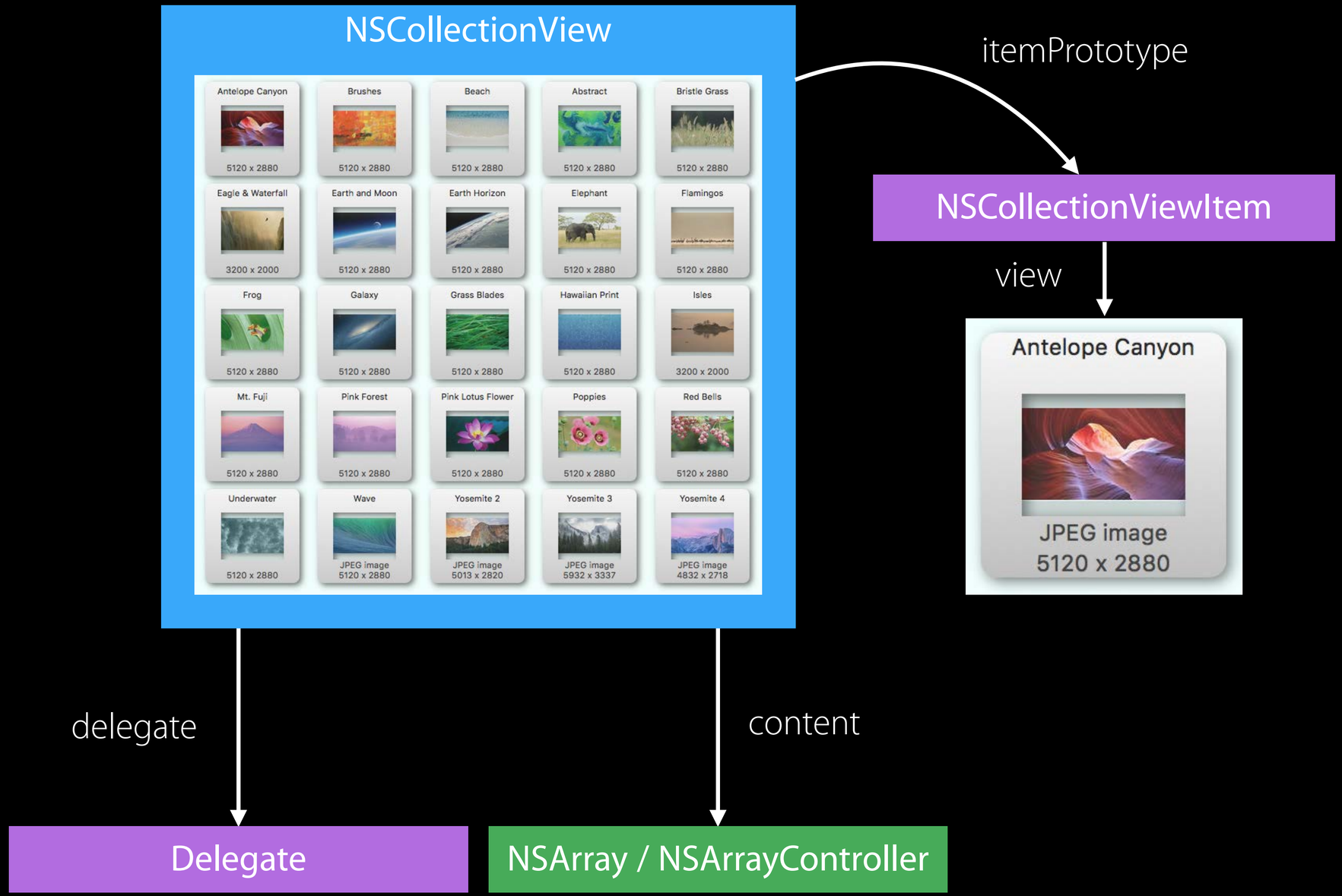
content

NSArray / NSArrayController

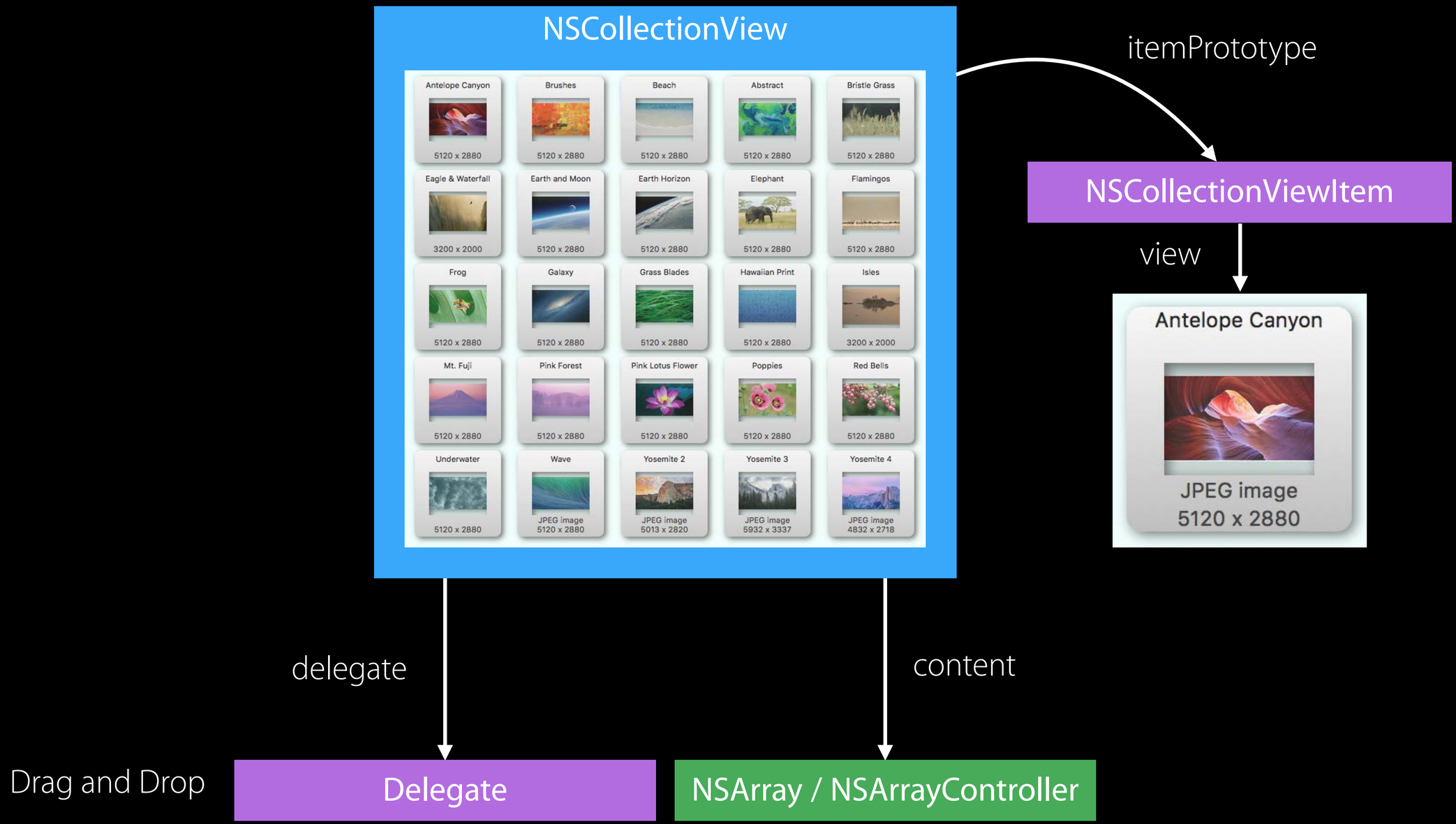
Old API


NSCollectionView
Antelope Canyon 5120 x 2880
Brushes 5120 x 2880
Beach 5120 x 2880
Abstract 5120 x 2880
Bristle Grass 5120 x 2880
Eagle & Waterfall 3200 x 2000
Earth and Moon 5120 x 2880
Earth Horizon 5120 x 2880
Elephant 5120 x 2880
Flamingos 5120 x 2880
Frog 5120 x 2880
Galaxy 5120 x 2880
Grass Blades 5120 x 2880
Hawaiian Print 5120 x 2880
Isles 3200 x 2000
Mt. Fuji 5120 x 2880
Pink Forest 5120 x 2880
Pink Lotus Flower 5120 x 2880
Poppies 5120 x 2880
Red Bells 5120 x 2880
Underwater 5120 x 2880
Wave JPEG image 5120 x 2880
Yosemite 2 JPEG image 5013 x 2820
Yosemite 3 JPEG image 5932 x 3337
Yosemite 4 JPEG image 4832 x 2718
itemPrototype
NSCollectionViewItem
view
Antelope Canyon
JPEG image
5120 x 2880
content
NSArray / NSArrayController

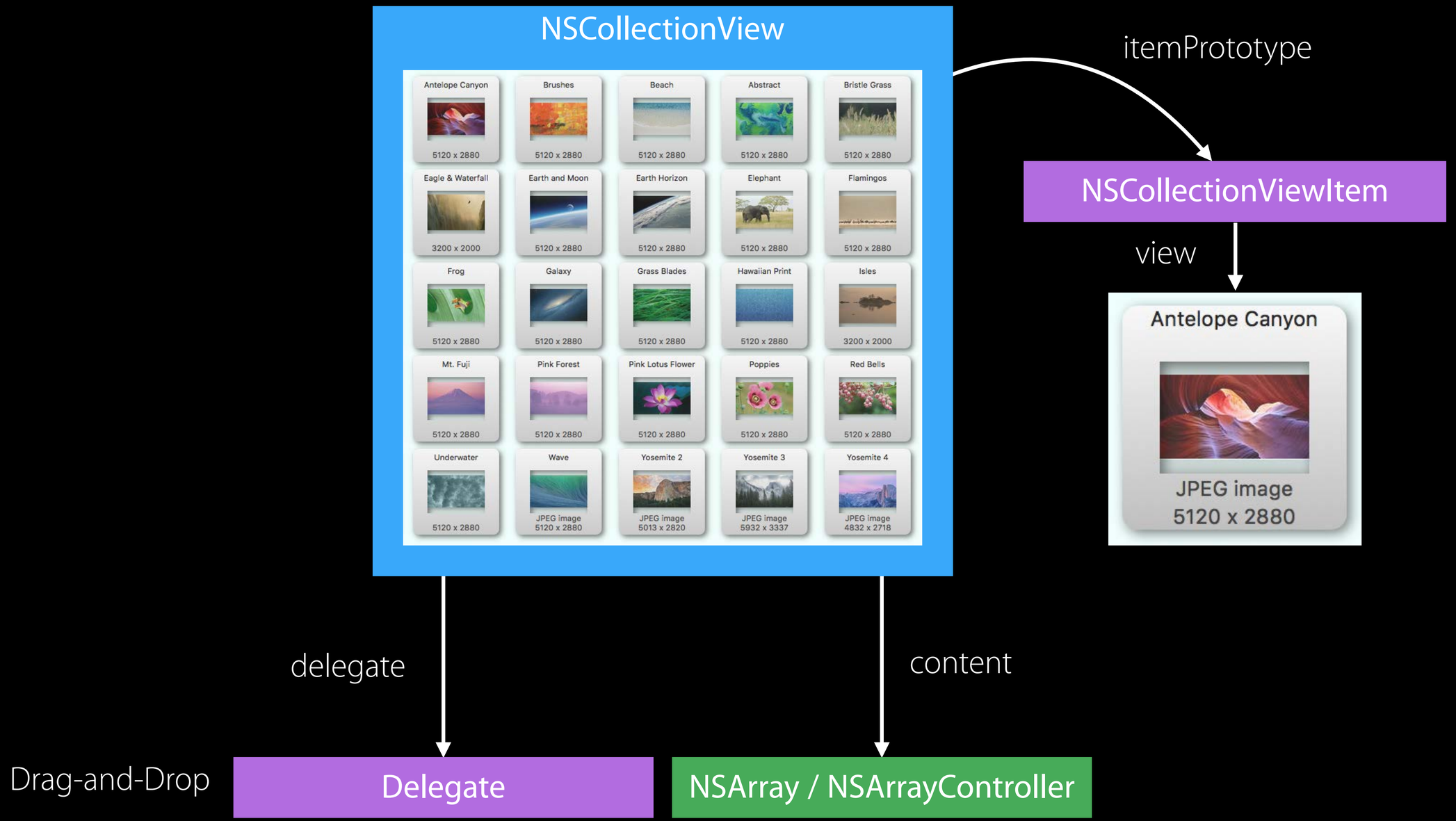
Old API


NSCollectionView
Antelope Canyon
Brushes
Beach
Abstract
Bristle Grass
5120 x 2880
Eagle & Waterfall
Earth and Moon
Earth Horizon
Elephant
Flamingos
3200 x 2000
Frog
Galaxy
Grass Blades
Hawaiian Print
Isles
Mt. Fuji
Pink Forest
Pink Lotus Flower
Poppies
Red Bells
Underwater
Wave
Yosemite 2
Yosemite 3
Yosemite 4
JPEG image
5120 x 2880
5013 x 2820
5932 x 3337
4832 x 2718
itemPrototype
NSCollectionViewItem
view
Antelope Canyon
JPEG image
5120 x 2880
delegate
content
Delegate
NSArray / NSArrayController

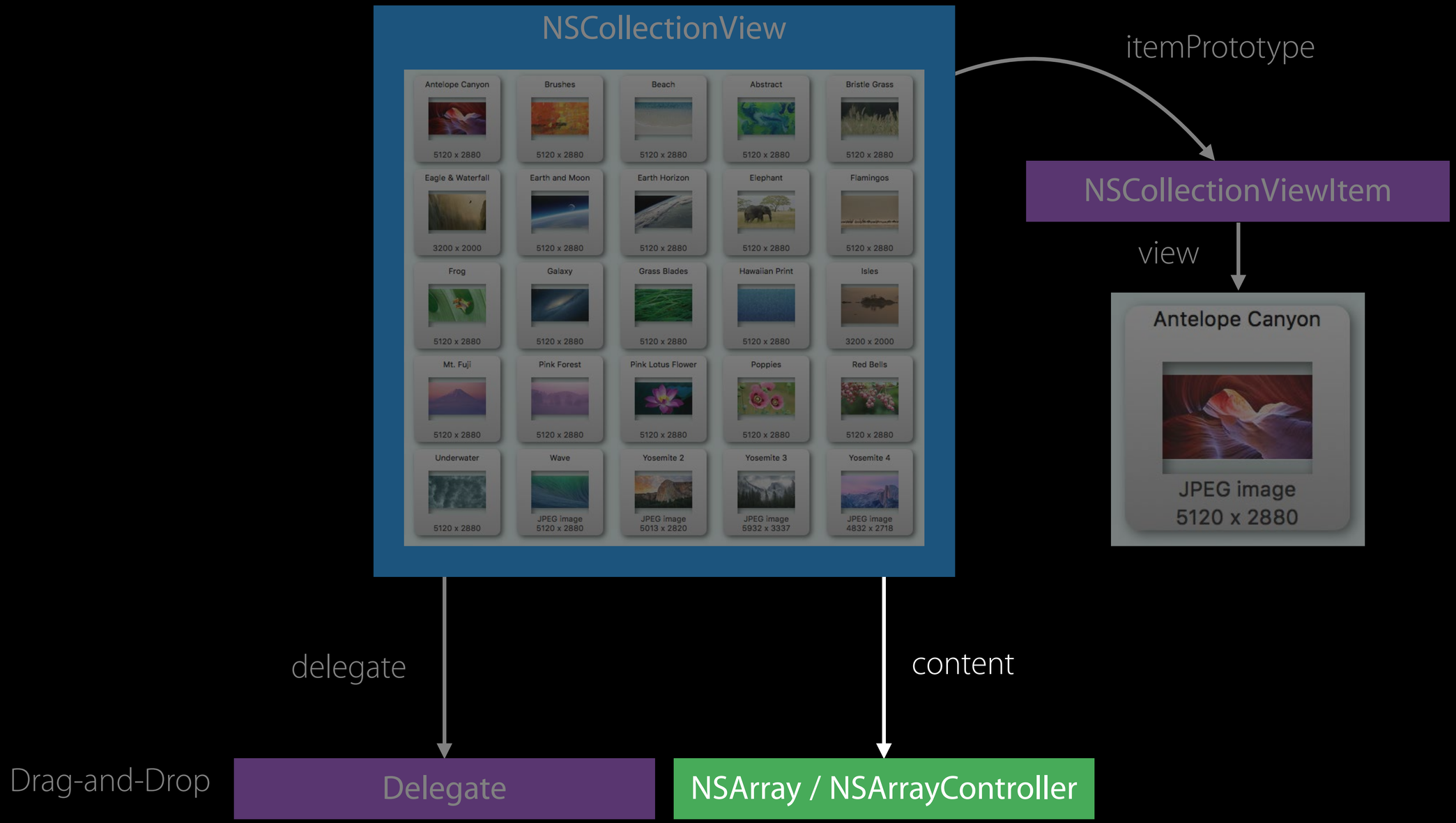
Old API


NSCollectionView
Antelope Canyon
Brushes
Beach
Abstract
Bristle Grass
5120 x 2880
Eagle & Waterfall
Earth and Moon
Earth Horizon
Elephant
Flamingos
3200 x 2000
Frog
Galaxy
Grass Blades
Hawaiian Print
Isles
Mt. Fuji
Pink Forest
Pink Lotus Flower
Poppies
Red Bells
Underwater
Wave
Yosemite 2
Yosemite 3
Yosemite 4
JPEG image
5120 x 2880
5013 x 2820
5932 x 3337
4832 x 2718
itemPrototype
NSCollectionViewItem
view
Antelope Canyon
JPEG image
5120 x 2880
delegate
content
Drag and Drop
Delegate
NSArray / NSArrayController

New API

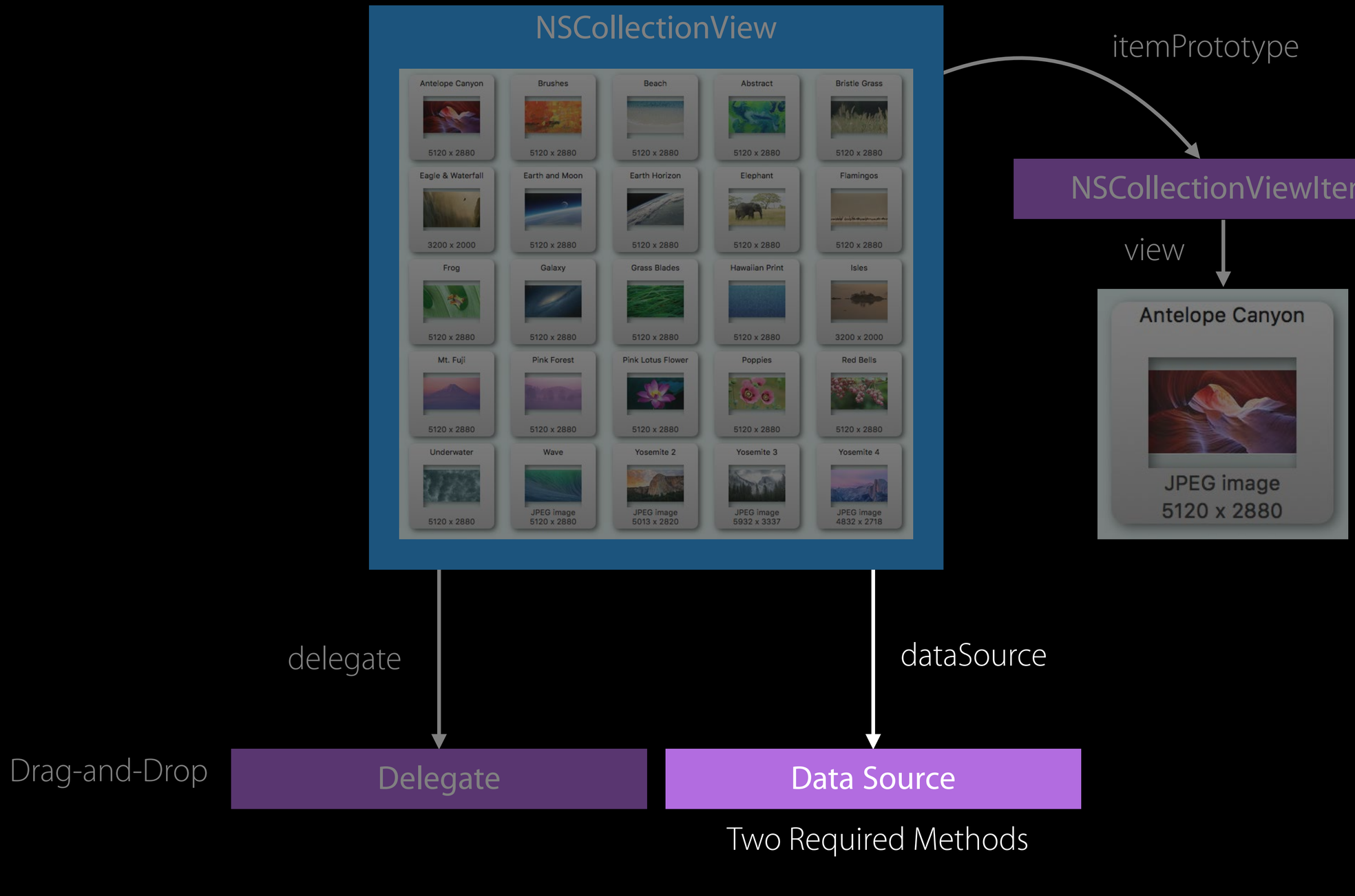

NSCollectionView
Antelope Canyon
5120 x 2880
Brushes
5120 x 2880
Beach
5120 x 2880
Abstract
5120 x 2880
Bristle Grass
5120 x 2880
Eagle & Waterfall
3200 x 2000
Earth and Moon
5120 x 2880
Earth Horizon
5120 x 2880
Elephant
5120 x 2880
Flamingos
5120 x 2880
Frog
5120 x 2880
Galaxy
5120 x 2880
Grass Blades
5120 x 2880
Hawaiian Print
5120 x 2880
Isles
3200 x 2000
Mt. Fuji
5120 x 2880
Pink Forest
5120 x 2880
Pink Lotus Flower
5120 x 2880
Poppies
5120 x 2880
Red Bells
5120 x 2880
Underwater
5120 x 2880
Wave
JPEG image
5120 x 2880
Yosemite 2
JPEG image
5013 x 2820
Yosemite 3
JPEG image
5932 x 3337
Yosemite 4
JPEG image
4832 x 2718
itemPrototype
NSCollectionViewItem
view
Antelope Canyon
JPEG image
5120 x 2880
delegate
content
Drag-and-Drop
Delegate
NSArray / NSArrayController

New API


NSCollectionView
Antelope Canyon
Brushes
Beach
Abstract
Bristle Grass
Eagle & Waterfall
Earth and Moon
Earth Horizon
Elephant
Flamingos
Frog
Galaxy
Grass Blades
Hawaiian Print
Isles
Mt. Fuji
Pink Forest
Pink Lotus Flower
Poppies
Red Bells
Underwater
Wave
Yosemite 2
Yosemite 3
Yosemite 4
itemPrototype
NSCollectionViewItem
view
Antelope Canyon
JPEG image
5120 x 2880
delegate
content
Drag-and-Drop
Delegate
NSArray / NSArrayController

New API

NEW


NSCollectionView
Antelope Canyon
Brushes
Beach
Abstract
Bristle Grass
Eagle & Waterfall
Earth and Moon
Earth Horizon
Elephant
Flamingos
Frog
Galaxy
Grass Blades
Hawaiian Print
Isles
Mt. Fuji
Pink Forest
Pink Lotus Flower
Poppies
Red Bells
Underwater
Wave
Yosemite 2
Yosemite 3
Yosemite 4
itemPrototype
NSCollectionViewItem
view
Antelope Canyon
JPEG image
5120 x 2880
delegate
dataSource
Drag-and-Drop
Delegate
Data Source
Two Required Methods

New API

NEW

itemPrototype

NSCollectionViewItem

view

delegate

dataSource

Drag-and-Drop

Selection and Highlighting

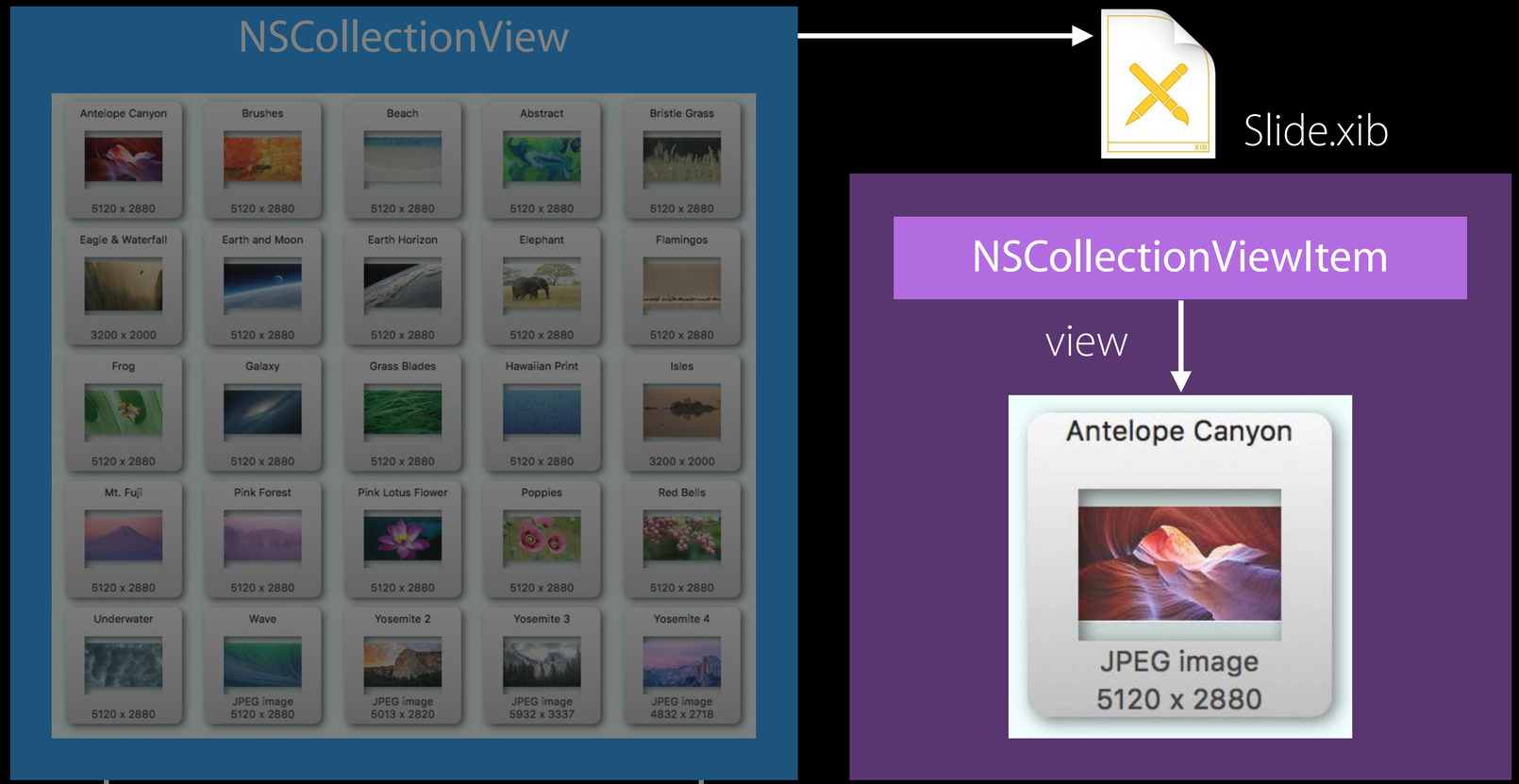
Delegate

Data Source

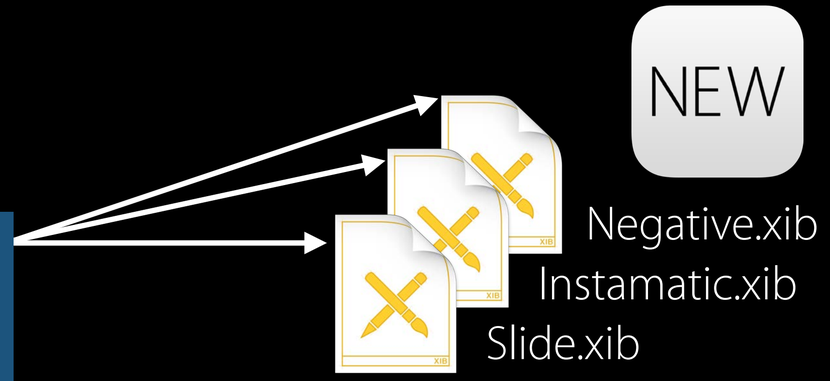
Two Required Methods

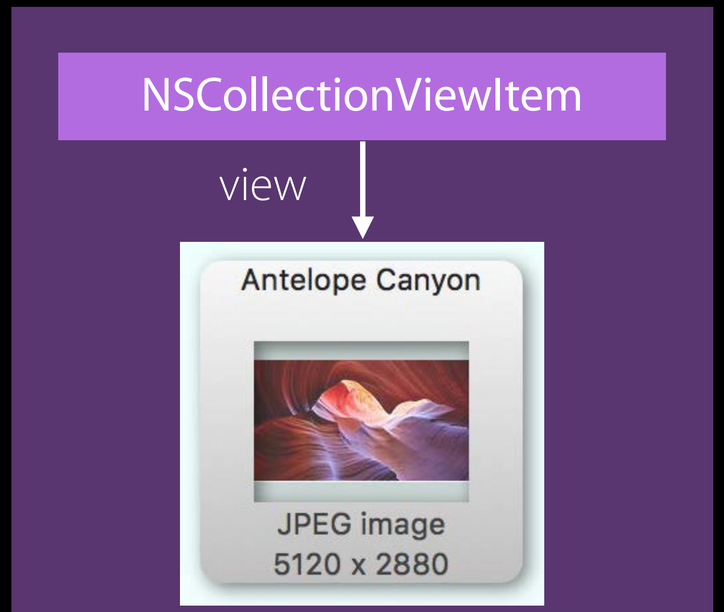
New API

NEW


NSCollectionView
Antelope Canyon 5120 x 2880
Brushes 5120 x 2880
Beach 5120 x 2880
Abstract 5120 x 2880
Bristle Grass 5120 x 2880
Eagle & Waterfall 3200 x 2000
Earth and Moon 5120 x 2880
Earth Horizon 5120 x 2880
Elephant 5120 x 2880
Flamingos 5120 x 2880
Frog 5120 x 2880
Galaxy 5120 x 2880
Grass Blades 5120 x 2880
Hawaiian Print 5120 x 2880
Isles 3200 x 2000
Mt. Fuji 5120 x 2880
Pink Forest 5120 x 2880
Pink Lotus Flower 5120 x 2880
Poppies 5120 x 2880
Red Bells 5120 x 2880
Underwater 5120 x 2880
Wave JPEG image 5120 x 2880
Yosemite 2 JPEG image 5013 x 2820
Yosemite 3 JPEG image 5932 x 3337
Yosemite 4 JPEG image 4832 x 2718
Slide.xib
NSCollectionViewItem
view
Antelope Canyon
JPEG image
5120 x 2880
delegate
dataSource
Drag-and-Drop
Selection and Highlighting
Delegate
Data Source
Two Required Methods

New API





delegate

dataSource

Drag-and-Drop

Selection and Highlighting

Delegate

Data Source

Two Required Methods

New API

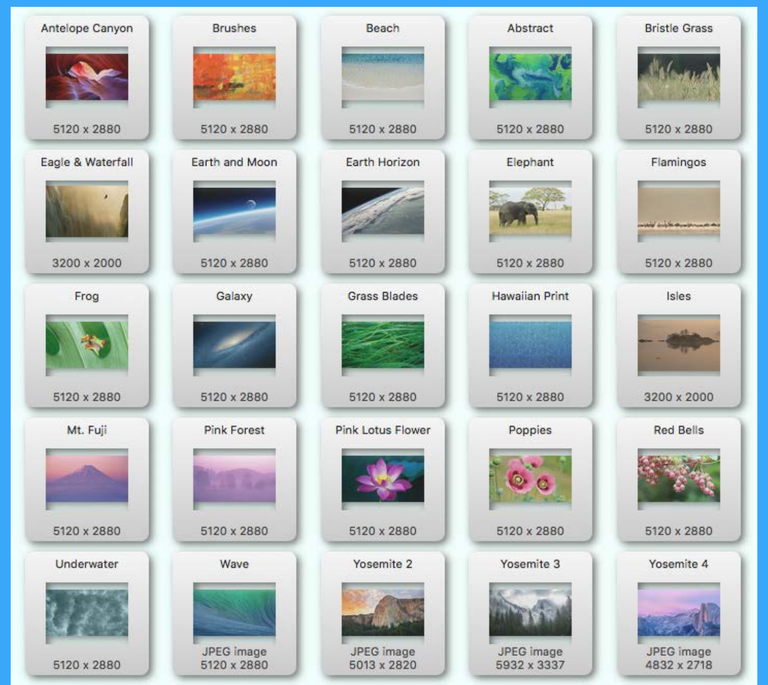
NEW

Negative.xib
Instamatic.xib
Slide.xib

collectionViewLayout

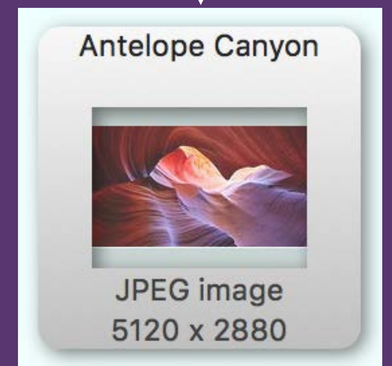
NSCollectionViewLayout

NSCollectionView



NSCollectionViewItem

view



delegate

dataSource

Drag-and-Drop
Selection and Highlighting

Delegate

Data Source

Two Required Methods

New API

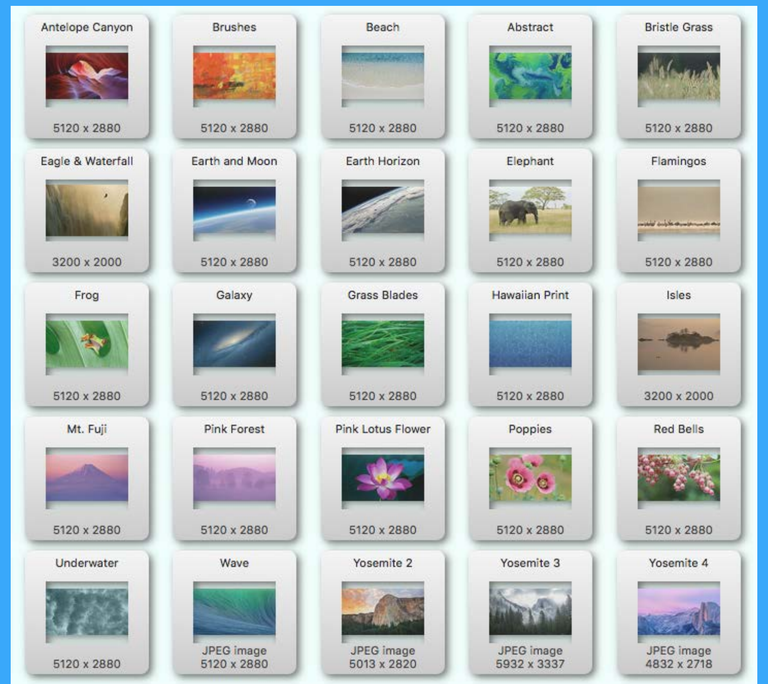
NEW

Negative.xib
Instamatic.xib
Slide.xib

collectionViewLayout

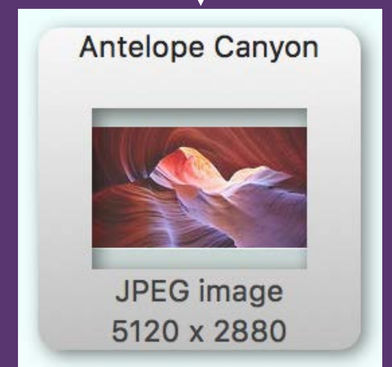
NSCollectionViewLayout

NSCollectionView



NSCollectionViewItem

view



delegate

dataSource

Drag-and-Drop
Selection and Highlighting
Per-Item Layout Adjustments

Delegate

Data Source

Two Required Methods

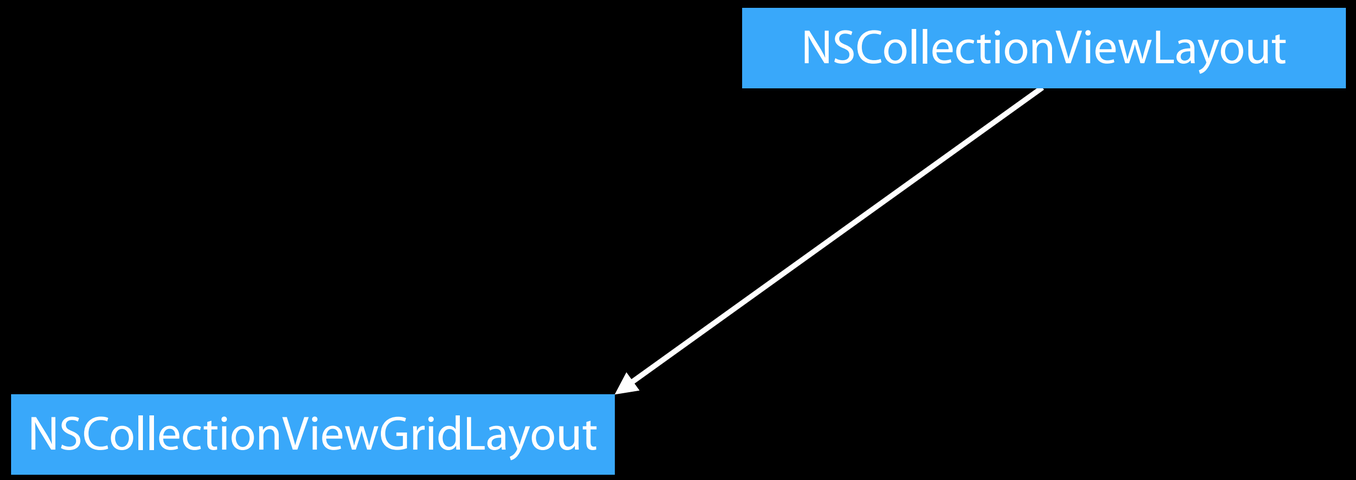
Layouts

Sizing and positioning items

NSCollectionViewLayout

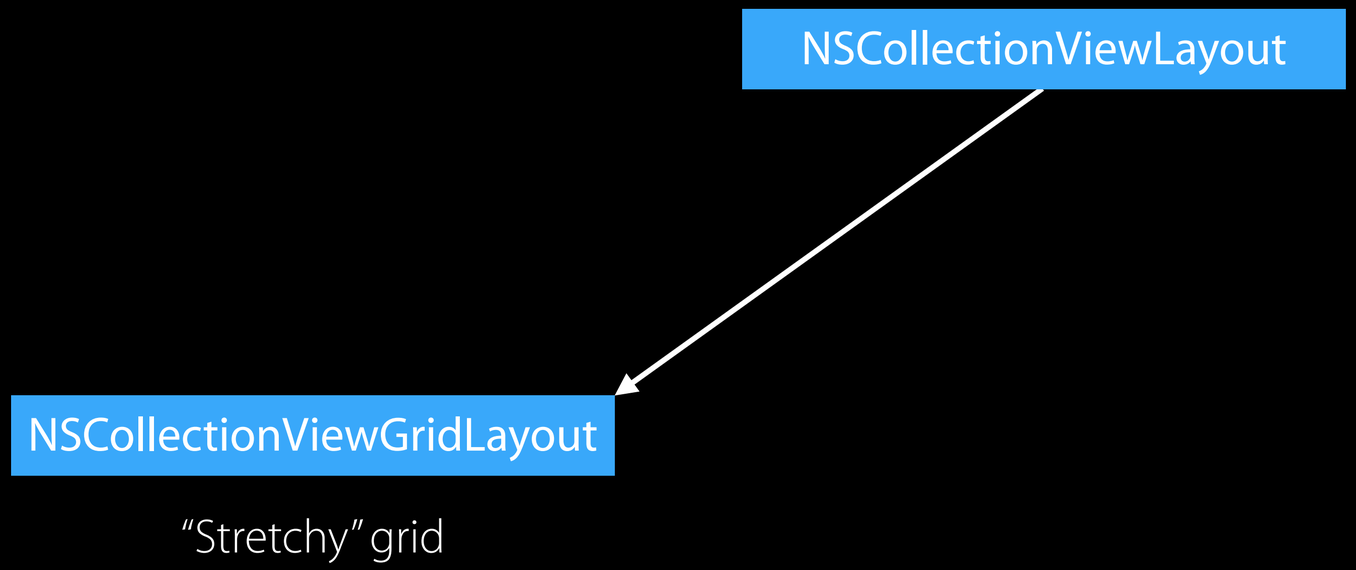
Layouts

Sizing and positioning items



Layouts

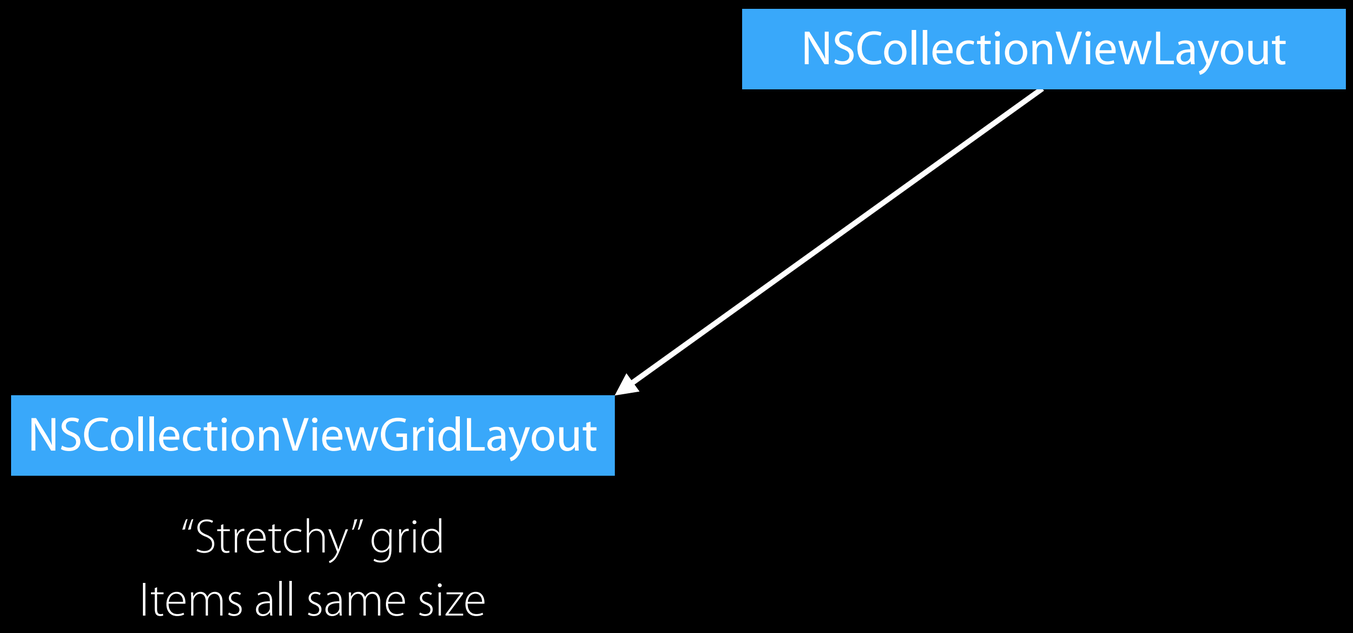
Sizing and positioning items



"Stretchy" grid

Layouts

Sizing and positioning items

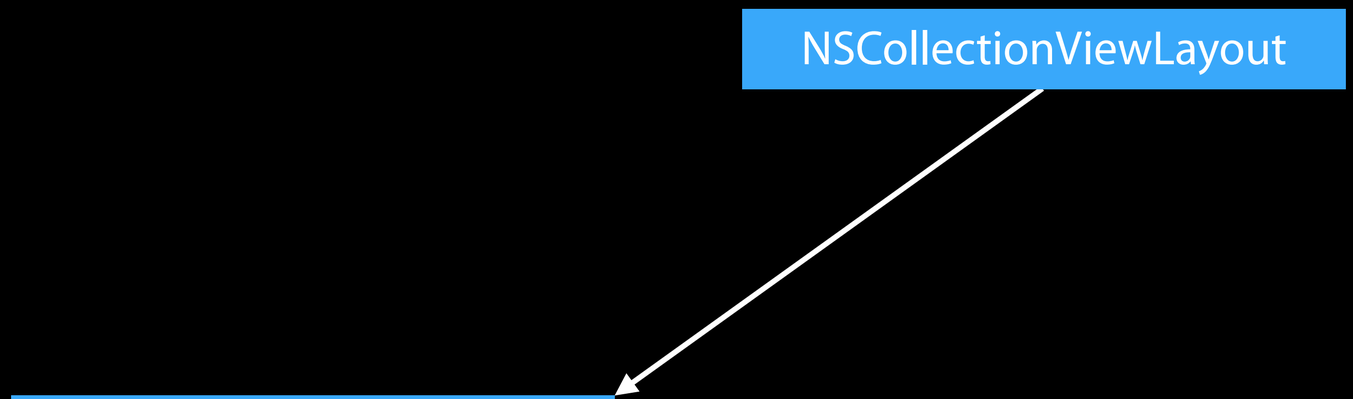


"Stretchy" grid

Items all same size

Layouts

Sizing and positioning items



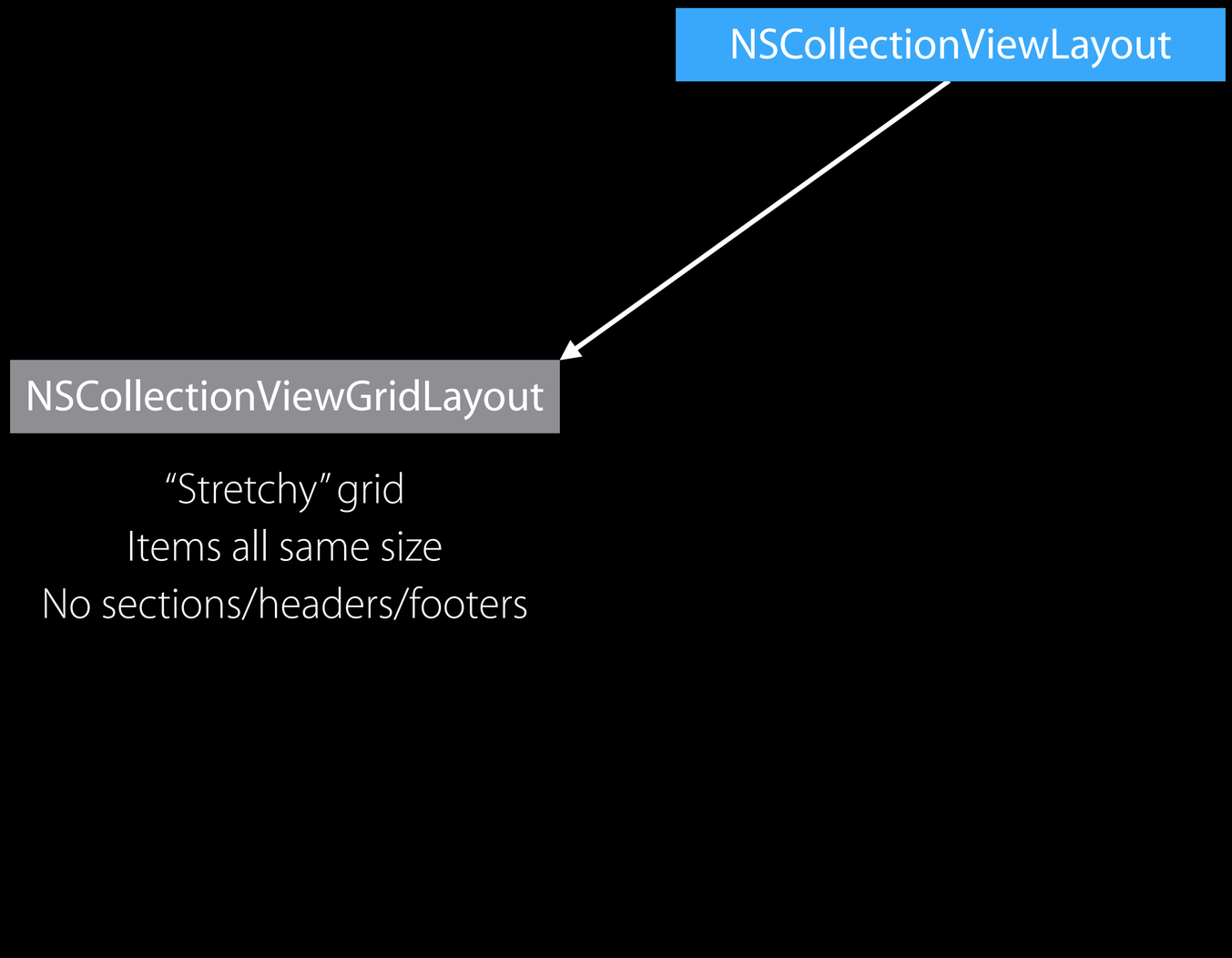
"Stretchy" grid

Items all same size

No sections/headers/footers

Layouts

Sizing and positioning items



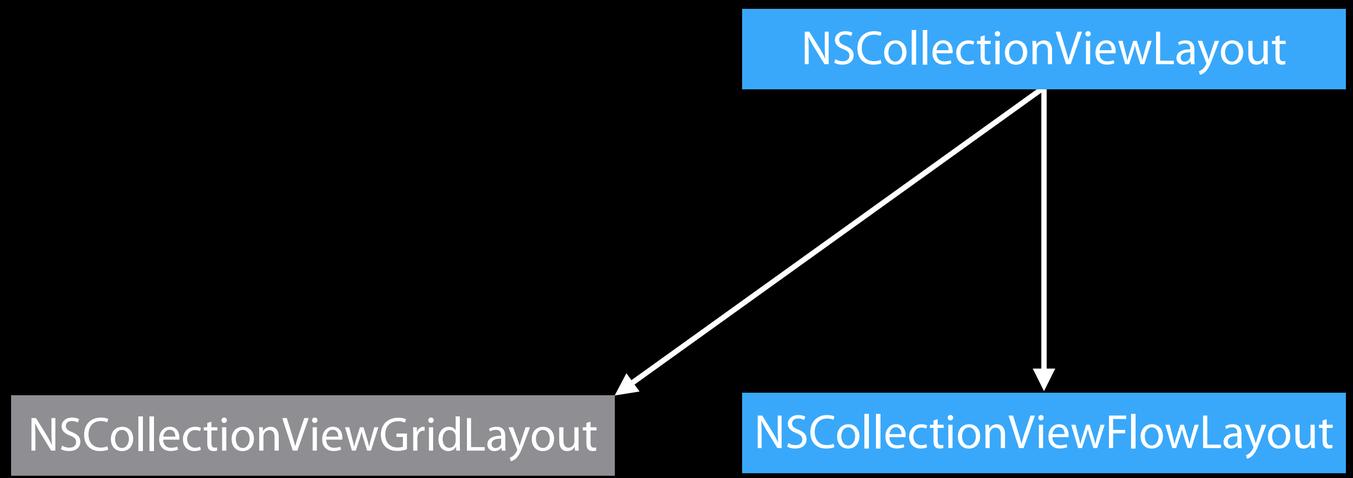
"Stretchy" grid

Items all same size

No sections/headers/footers

Layouts

Sizing and positioning items



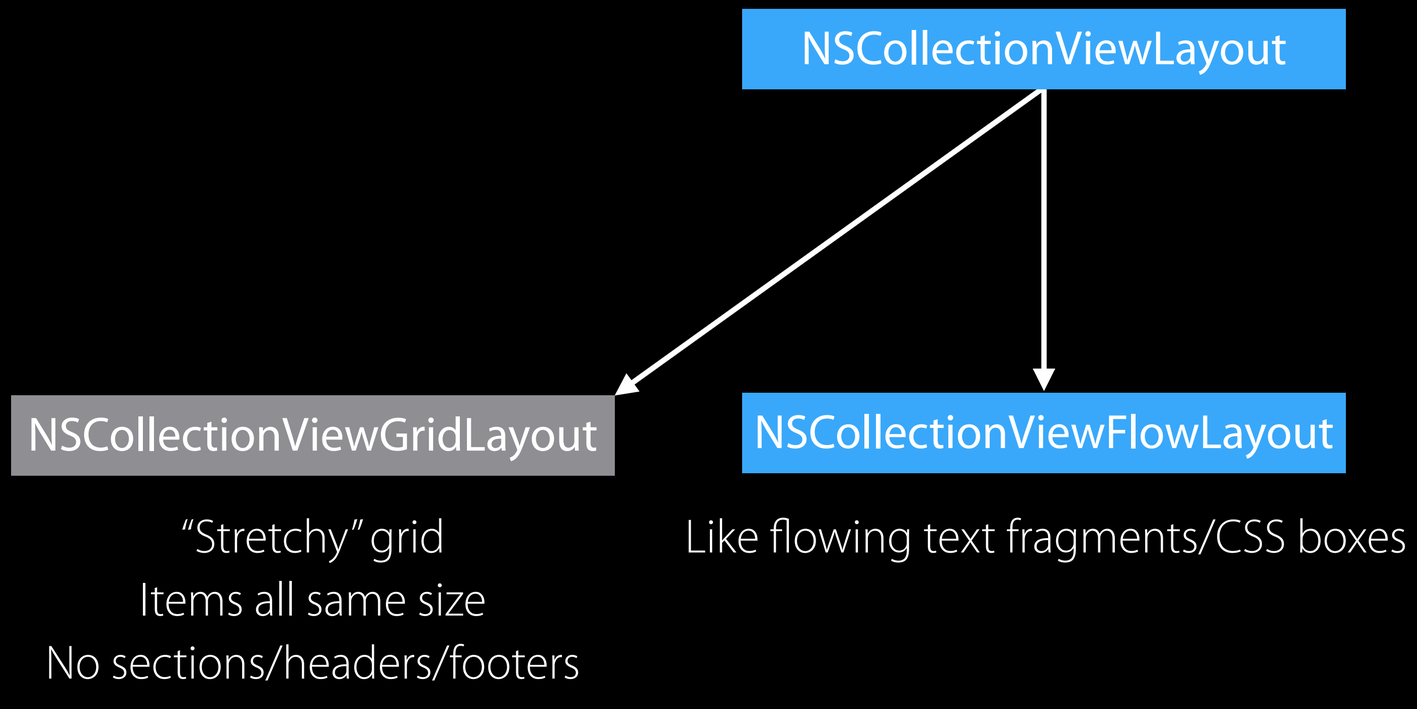
"Stretchy" grid

Items all same size

No sections/headers/footers

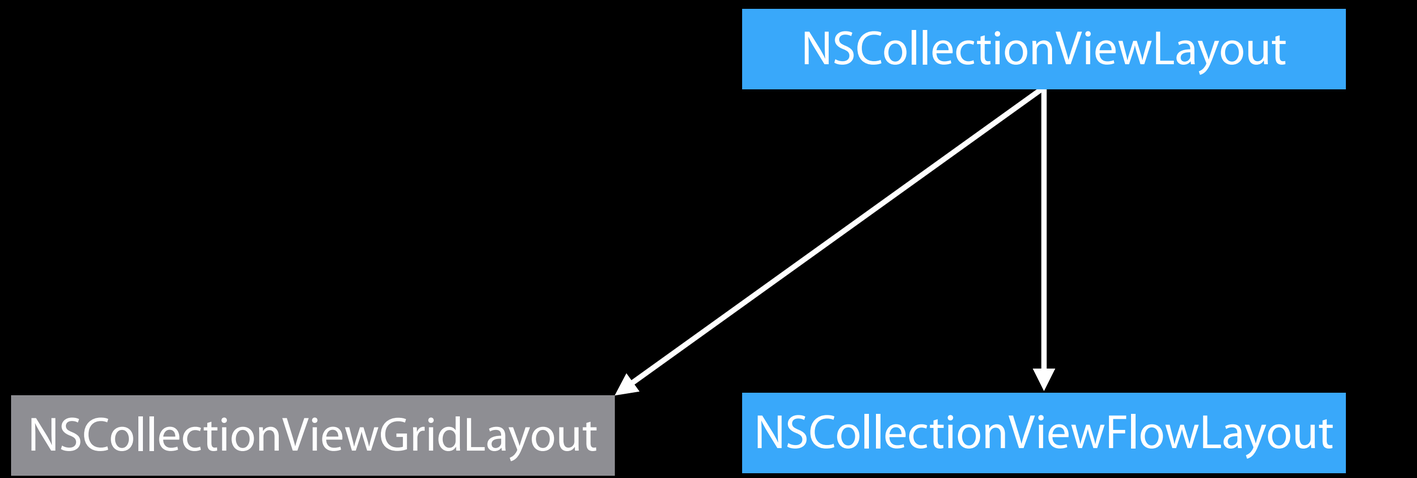
Layouts

Sizing and positioning items



Layouts

Sizing and positioning items

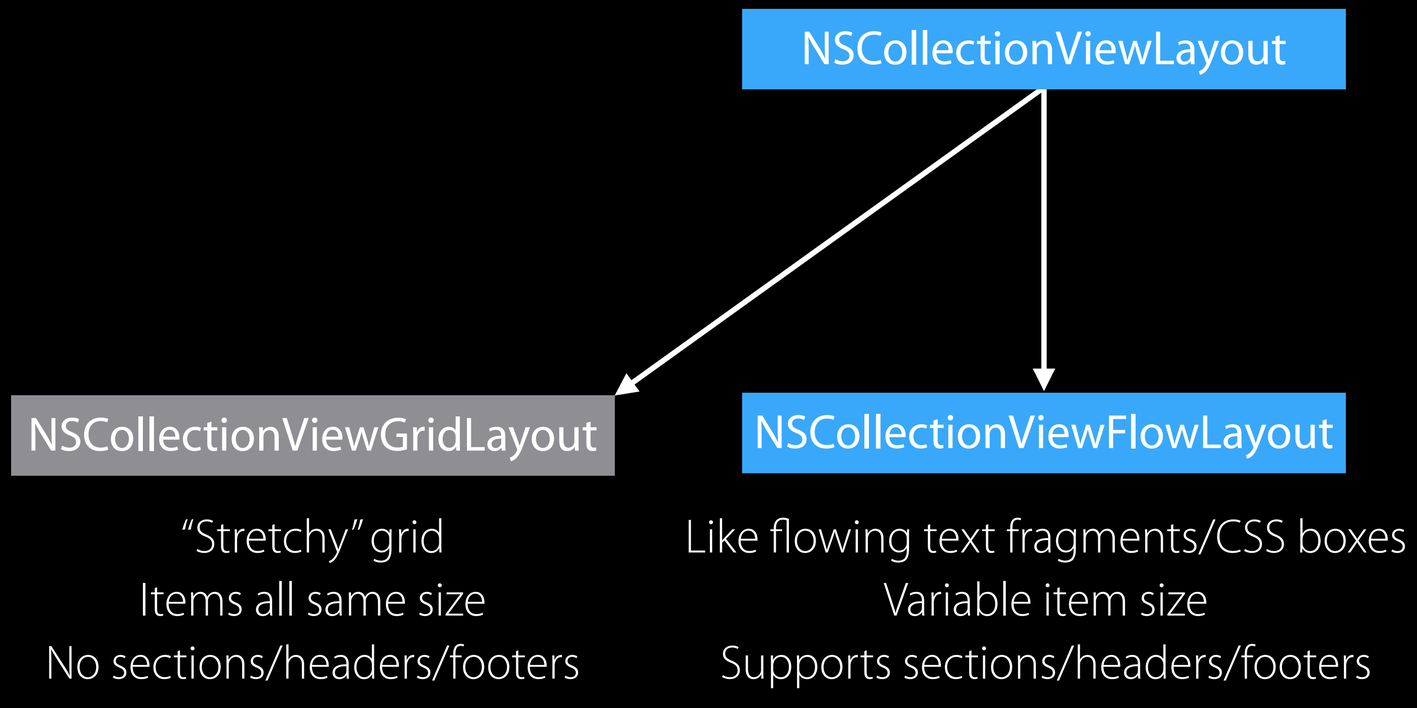


"Stretchy" grid
Items all same size
No sections/headers/footers

Like flowing text fragments/CSS boxes
Variable item size

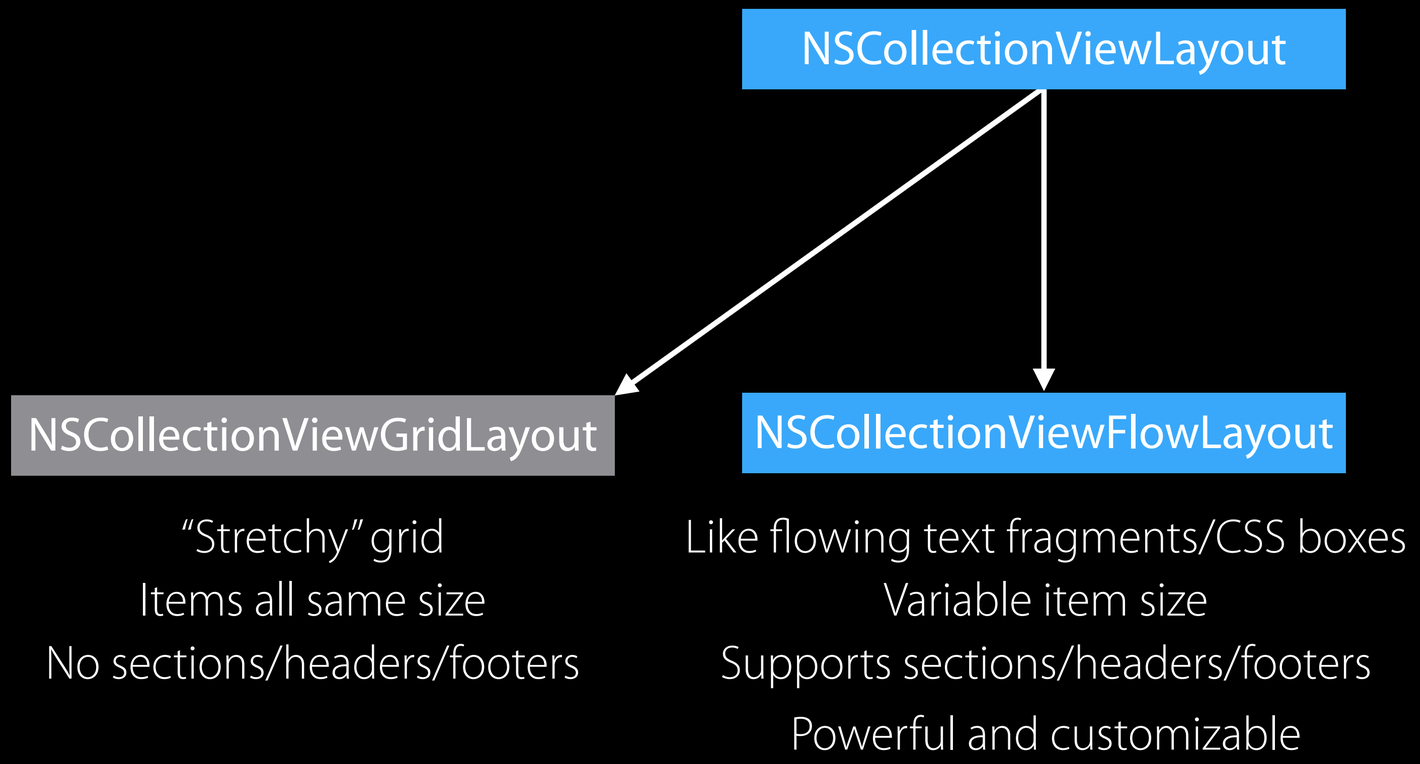
Layouts

Sizing and positioning items



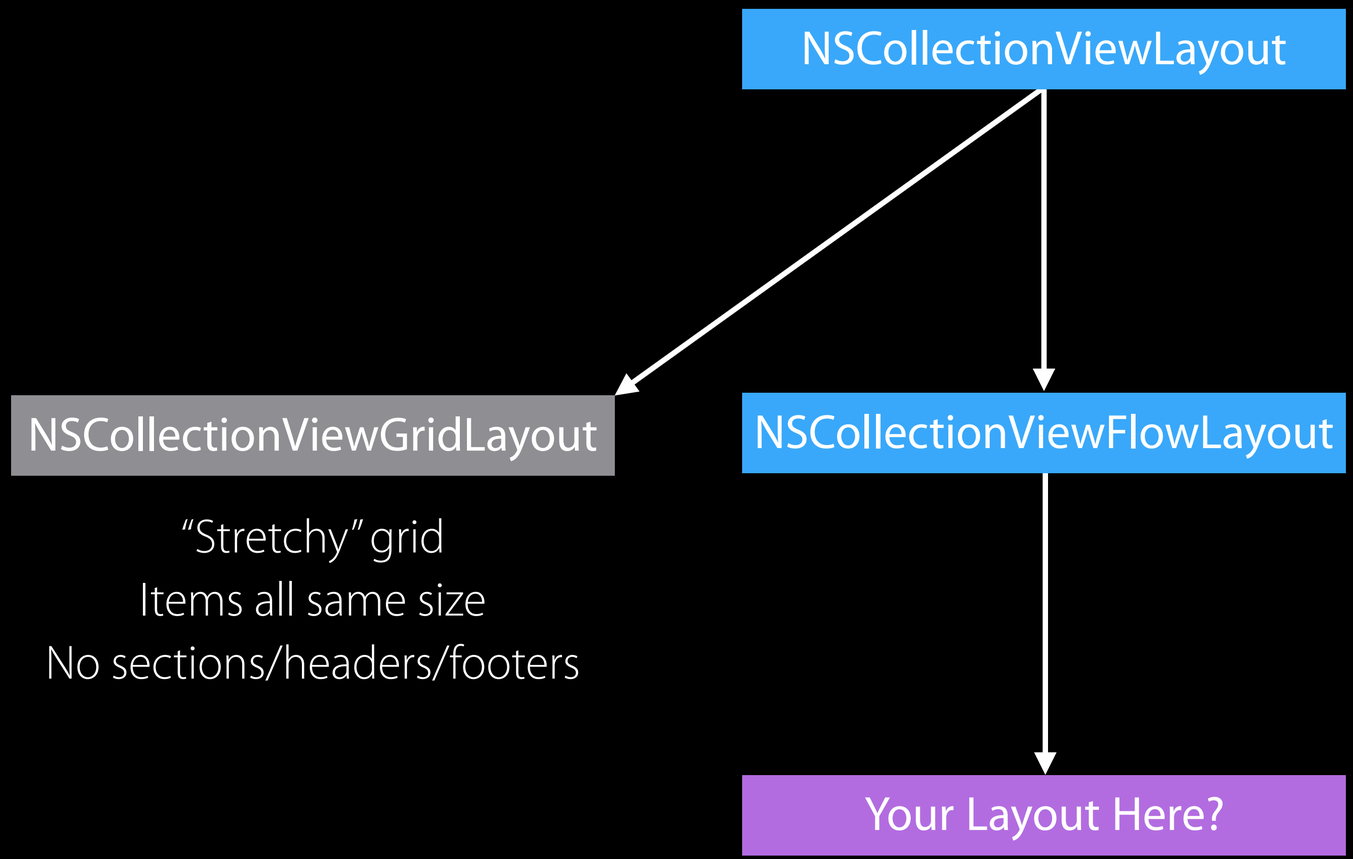
Layouts

Sizing and positioning items



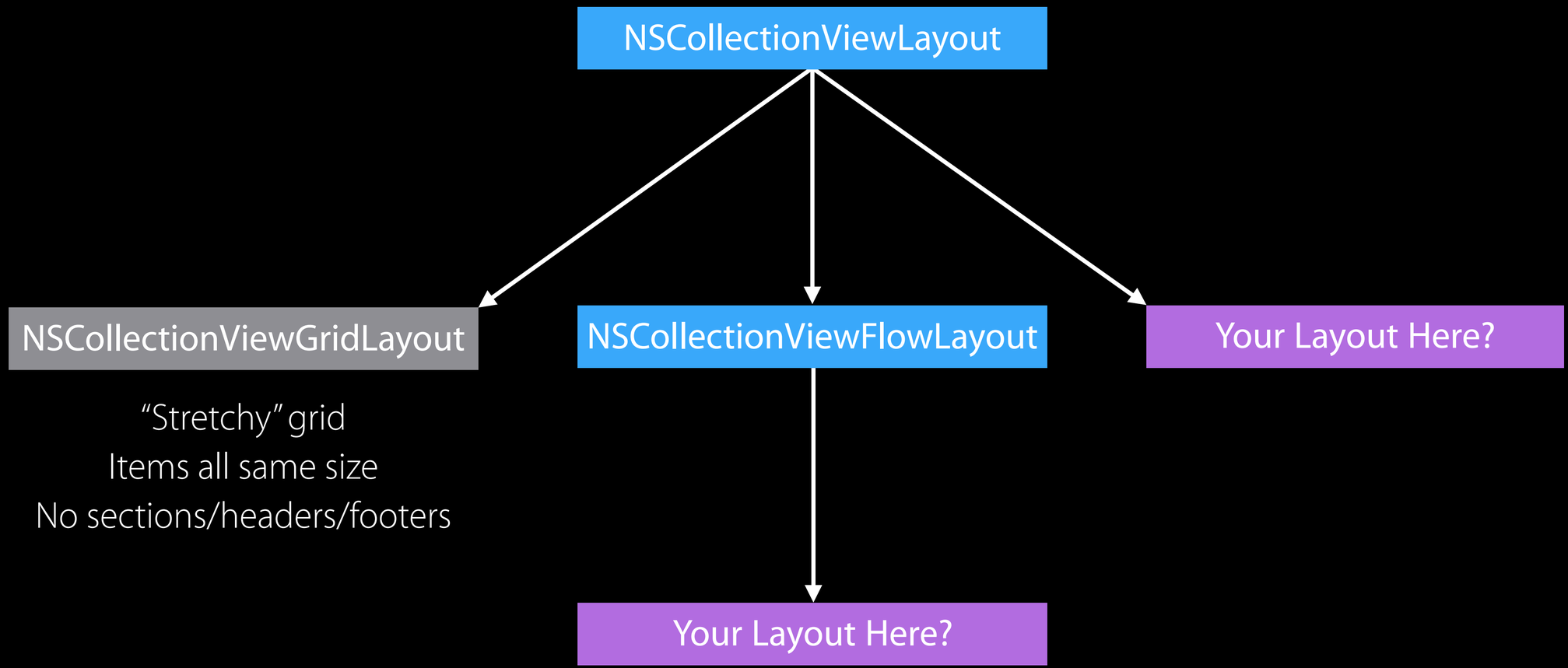
Layouts

Sizing and positioning items



Layouts

Sizing and positioning items



Understanding “Layout Attributes” Objects

NSCollectionViewLayoutAttributes

Understanding “Layout Attributes” Objects

NSCollectionViewLayoutAttributes

Encapsulates `frame`, `alphaValue`, and other states that can be applied to a view

```
class NSCollectionViewLayoutAttributes : NSObject, NSCopying {
    var frame: NSRect
    var size: NSSize
    var alpha: CGFloat
    var zIndex: Int // default is 0
    var hidden: Bool // As an optimization, NSCollectionView might not
create a view for items whose hidden attribute is YES
    var indexPath: NSIndexPath?
```

Understanding “Layout Attributes” Objects

NSCollectionViewLayoutAttributes

Encapsulates `frame`, `alphaValue`, and other states that can be applied to a view

Enables CollectionView APIs to reason about items not currently instantiated

```
class NSCollectionViewLayoutAttributes : NSObject, NSCopying {
    var frame: NSRect
    var size: NSSize
    var alpha: CGFloat
    var zIndex: Int // default is 0
    var hidden: Bool // As an optimization, NSCollectionView might not
create a view for items whose hidden attribute is YES
    var indexPath: NSIndexPath?
```

Understanding “Layout Attributes” Objects

NSCollectionViewLayoutAttributes

Encapsulates `frame`, `alphaValue`, and other states that can be applied to a view

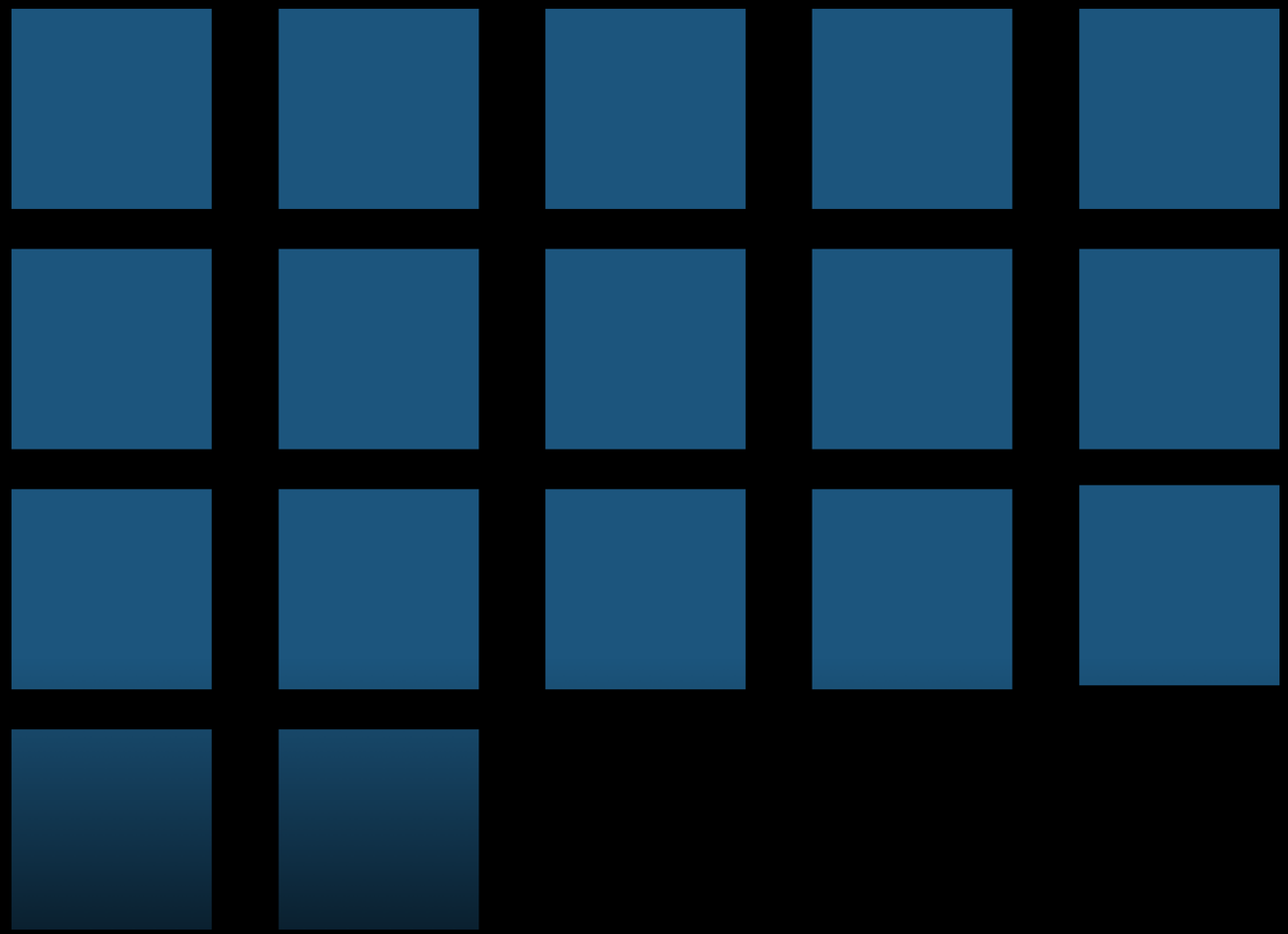
Enables CollectionView APIs to reason about items not currently instantiated

Applied to items/views at layout time

```
class NSCollectionViewLayoutAttributes : NSObject, NSCopying {
    var frame: NSRect
    var size: NSSize
    var alpha: CGFloat
    var zIndex: Int // default is 0
    var hidden: Bool // As an optimization, NSCollectionView might not
create a view for items whose hidden attribute is YES
    var indexPath: NSIndexPath?
```

Grouping Items

Using sections



Grouping Items

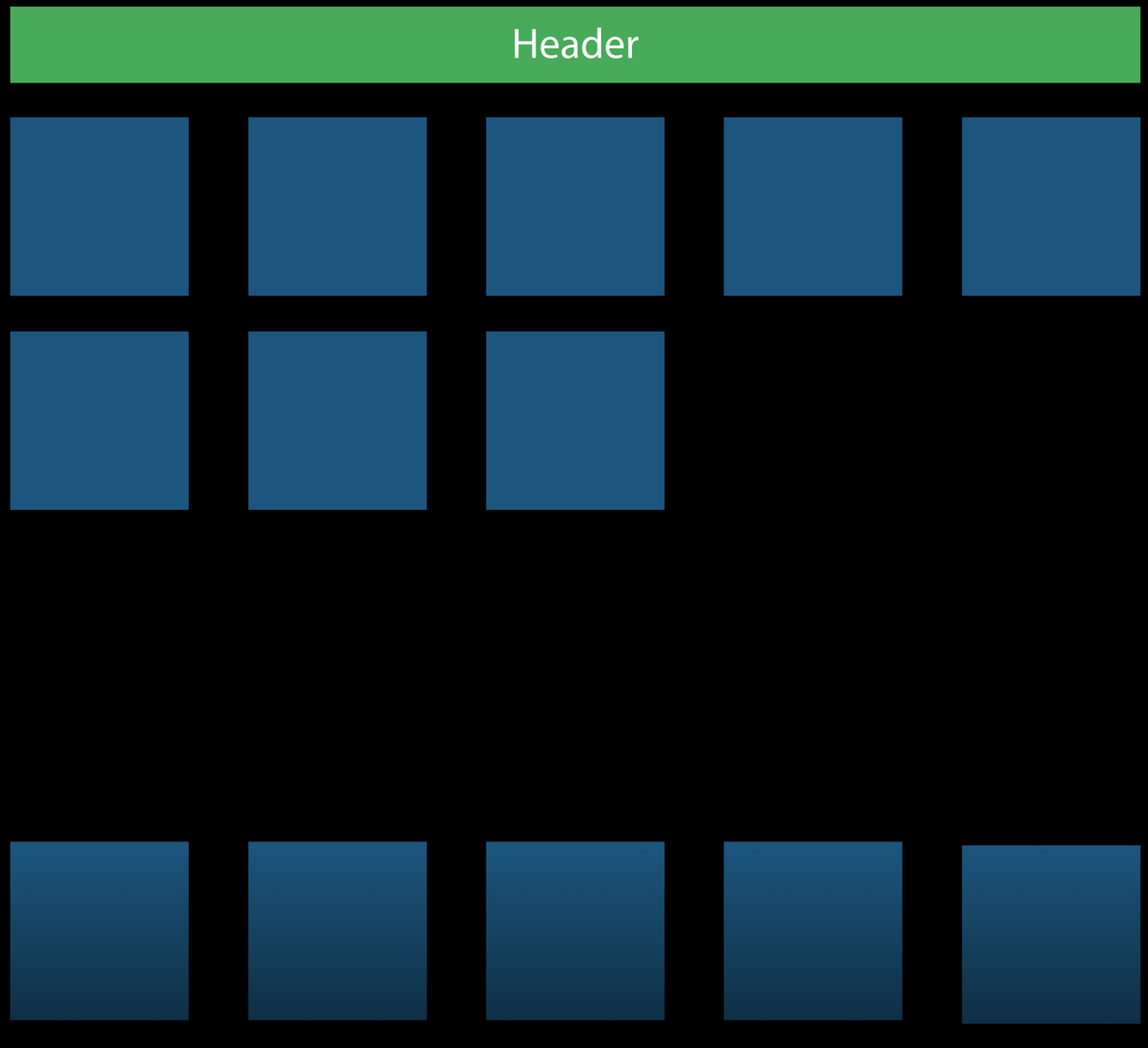
Using sections

Grouping Items

Using sections

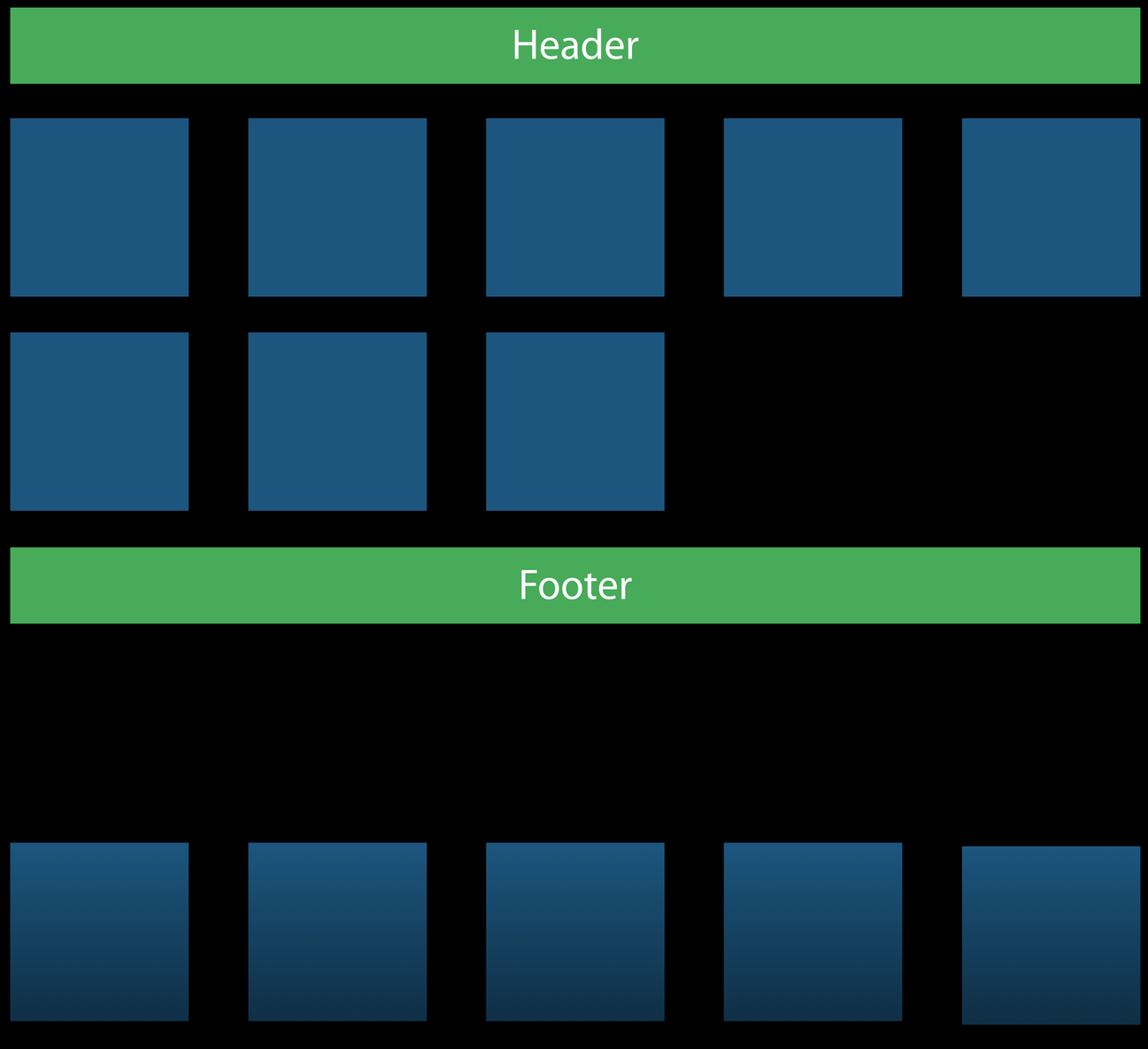
Grouping Items

Using sections



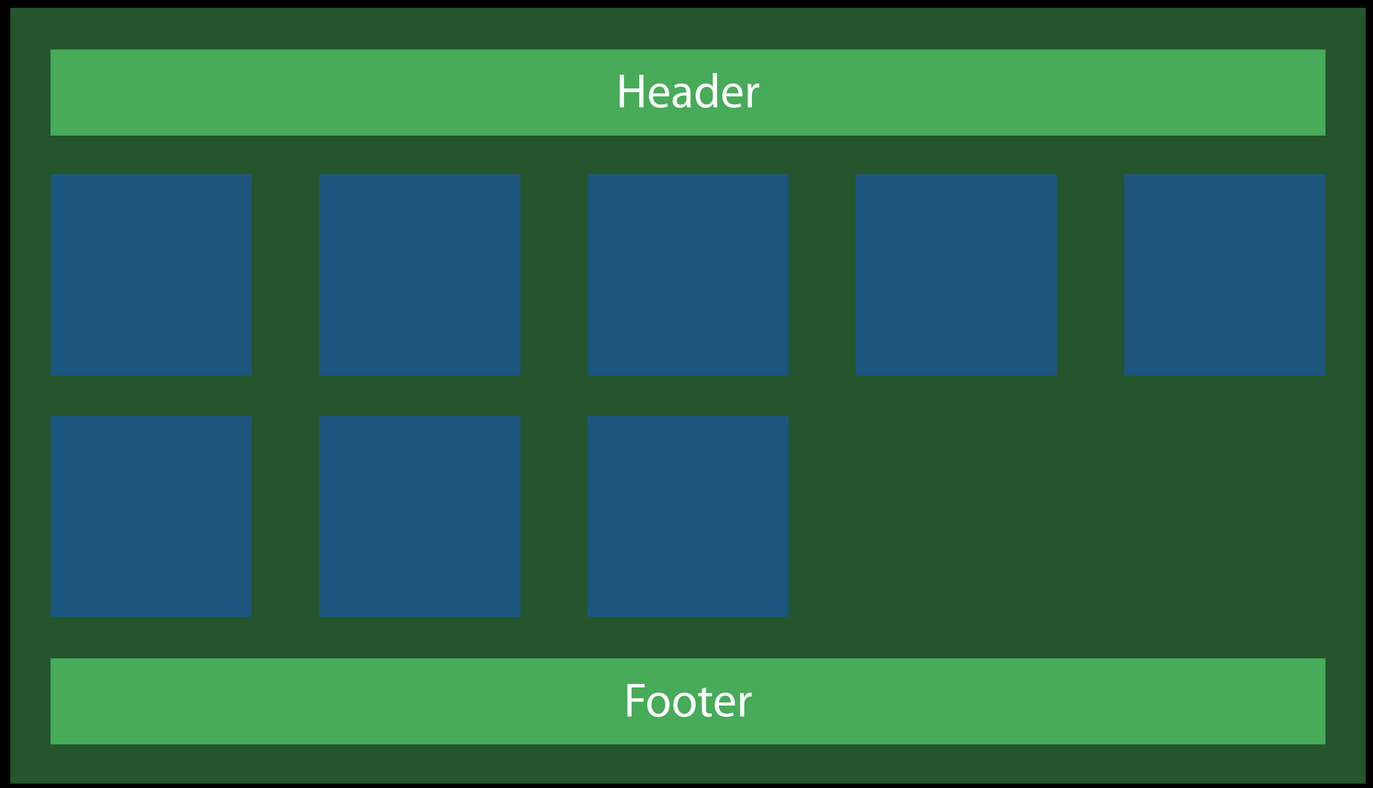
Grouping Items

Using sections



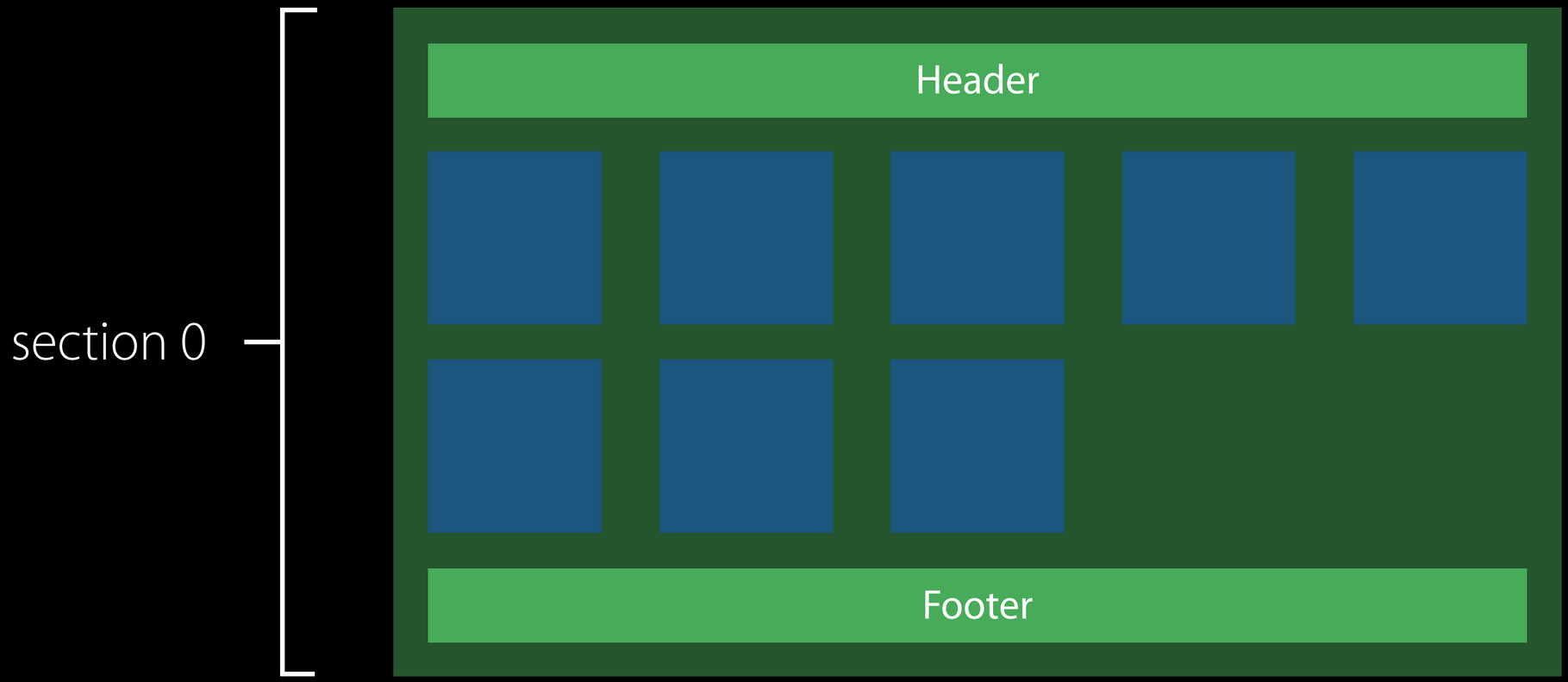
Grouping Items

Using sections



Grouping Items

Using sections



Grouping Items

Using sections

section 0

Header

item 0 | item 1 | item 2 | item 3 | item 4

item 5 | item 6 | item 7

Footer

Grouping Items

Using sections

section 0

Header

item 0 | item 1 | item 2 | item 3 | item 4

item 5 | item 6 | item 7

Footer

Header

Grouping Items

Using sections

section 0

Header

item 0 | item 1 | item 2 | item 3 | item 4

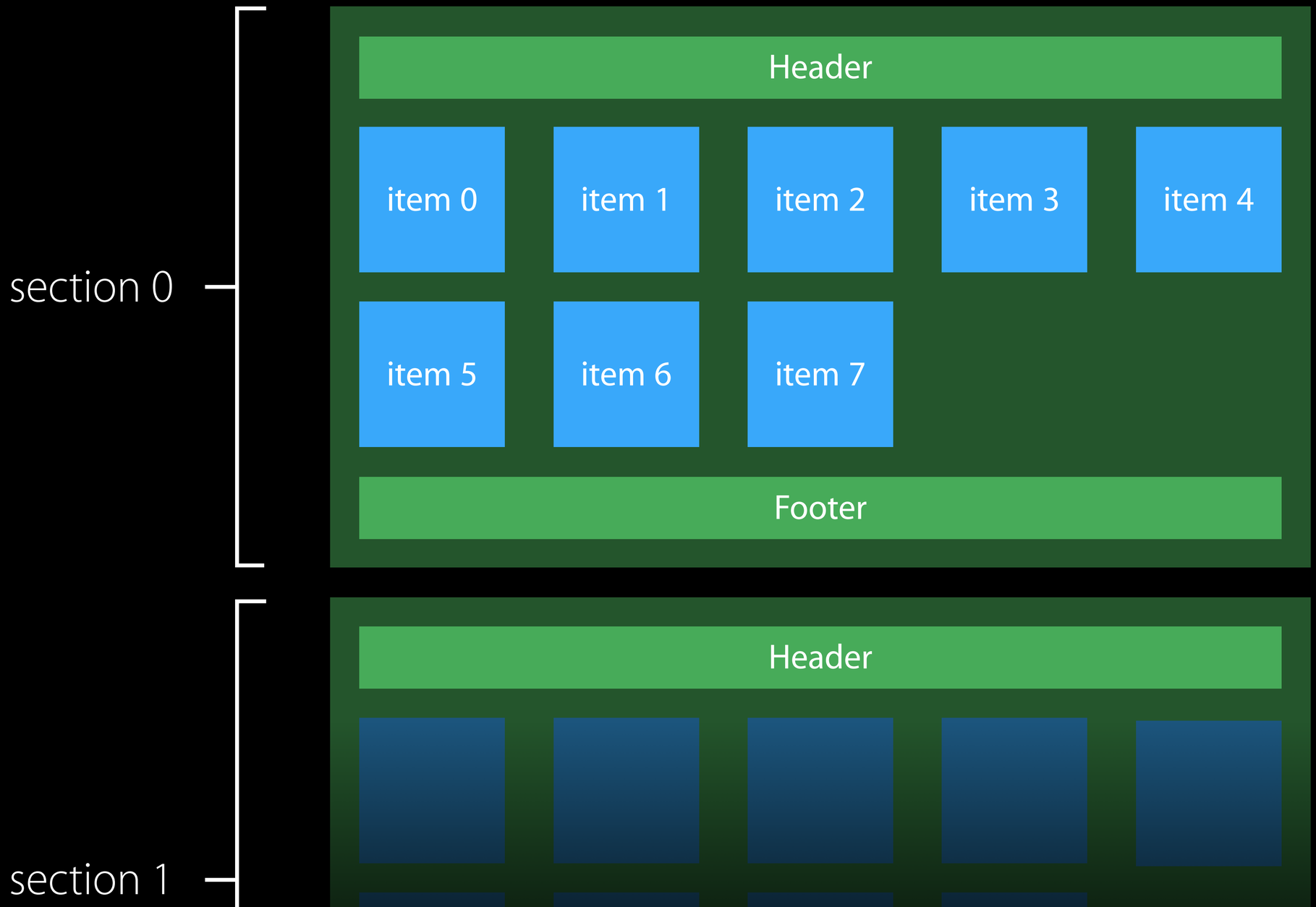
item 5 | item 6 | item 7

Footer

Header

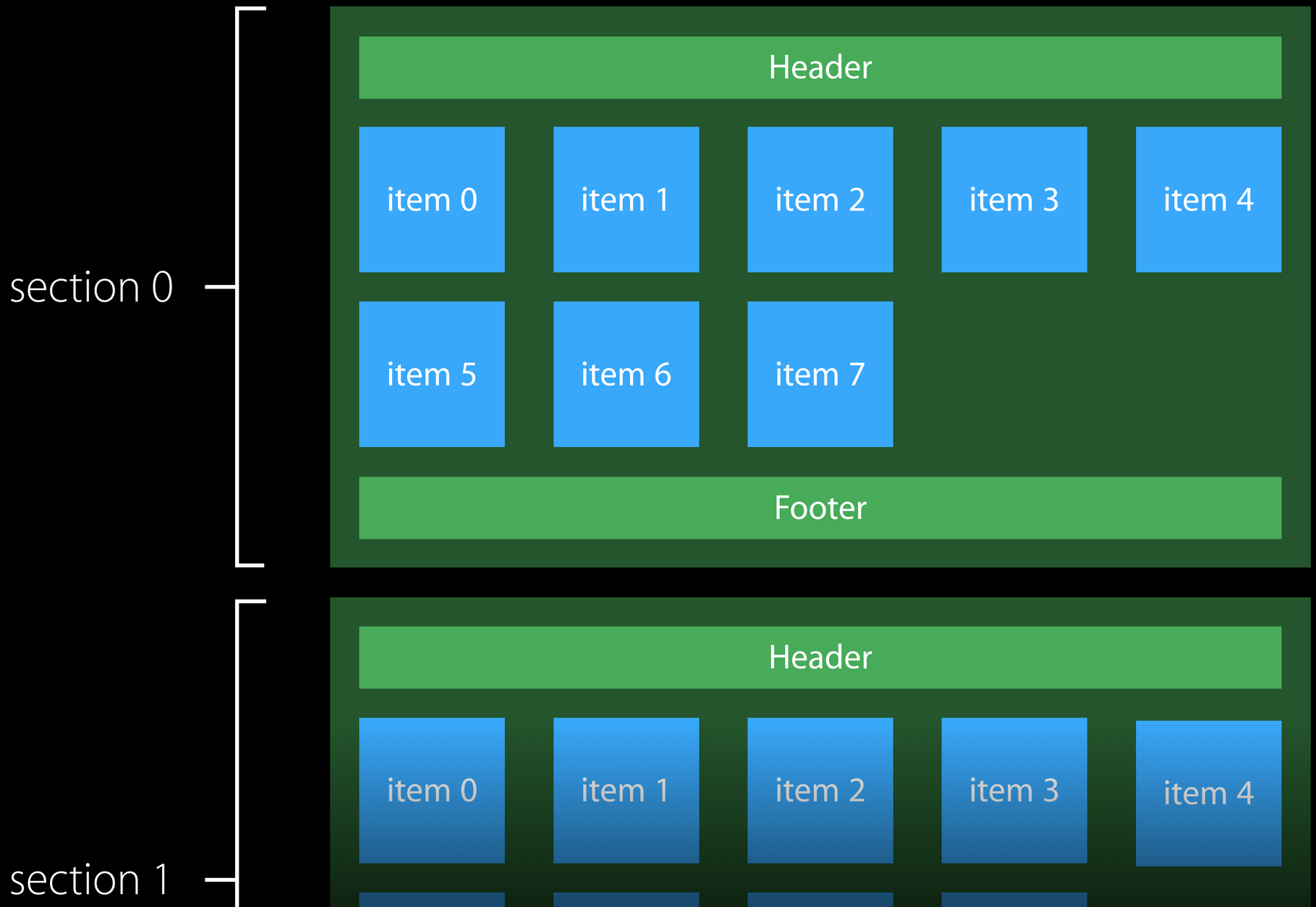
Grouping Items

Using sections



Grouping Items

Using sections



Referencing Items

Allowing for sections

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

```
itemAtIndexPath(NSIndexPath) -> NSCollectionViewItem?
```

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

```
itemAtIndexPath(NSIndexPath) -> NSCollectionViewItem?
```

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

```
itemAtIndexPath(NSIndexPath) -> NSCollectionViewItem?
```

(section, item)

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

```
itemAtIndexPath(NSIndexPath) -> NSCollectionViewItem?
```

NSIndexPath

(section, item)

Referencing Items

Allowing for sections

```
func itemAtIndex(index: Int) -> NSCollectionViewItem?
```

```
itemAtIndexPath(NSIndexPath) -> NSCollectionViewItem?
```

NSIndexPath

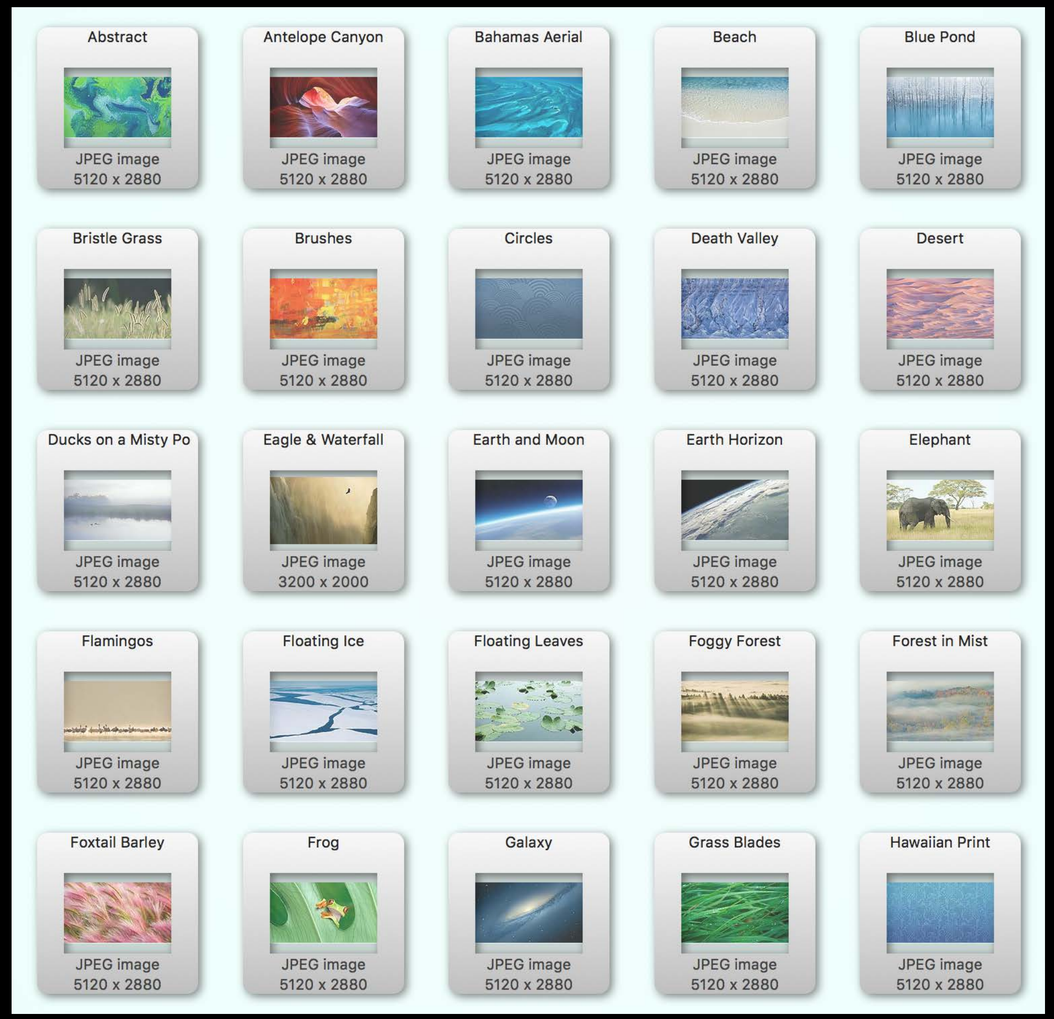
(section, item)

Nuts and Bolts

Putting CollectionView to work

Example

“Cocoa Slide Collection”

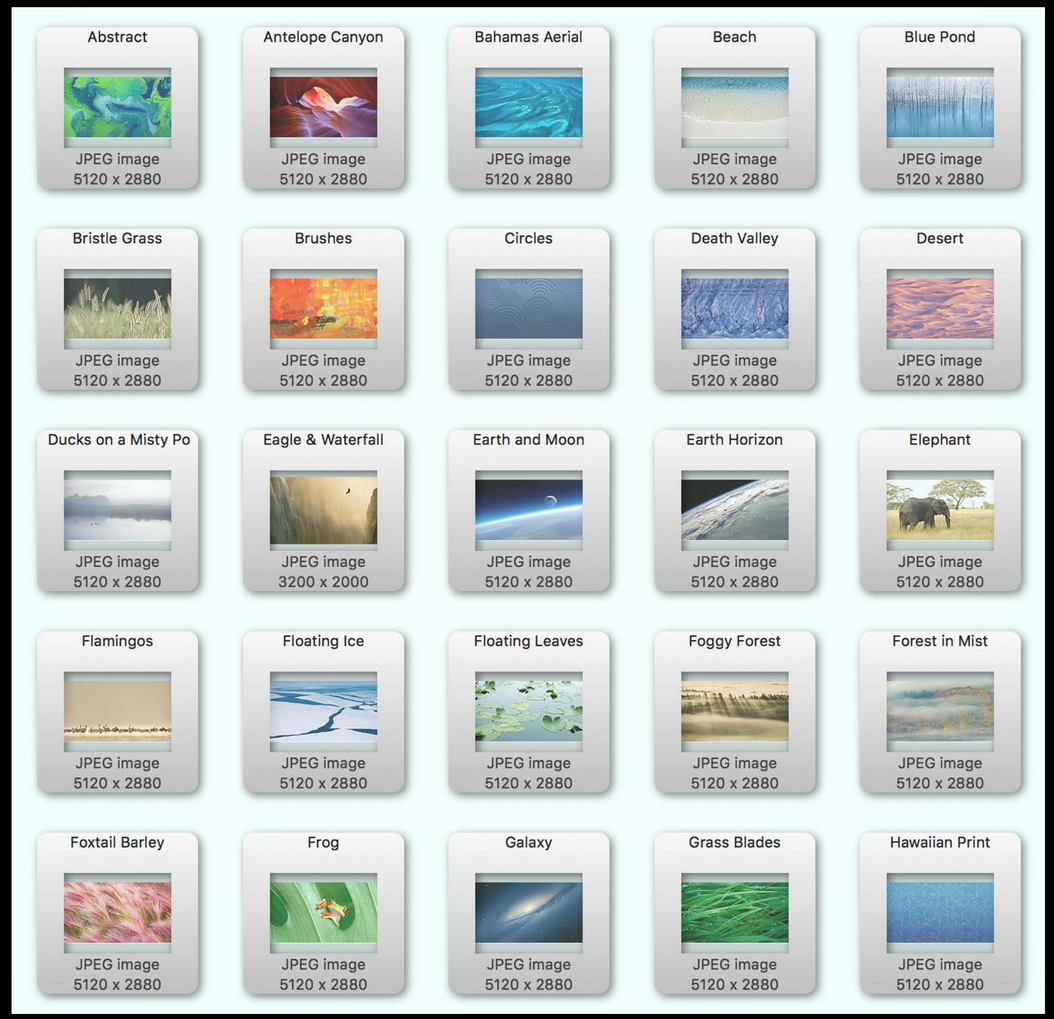


https://developer.apple.com/library/prerelease/mac/samplecode/CocoaSlideCollection

Example

“Cocoa Slide Collection”

Browse a folder of image files



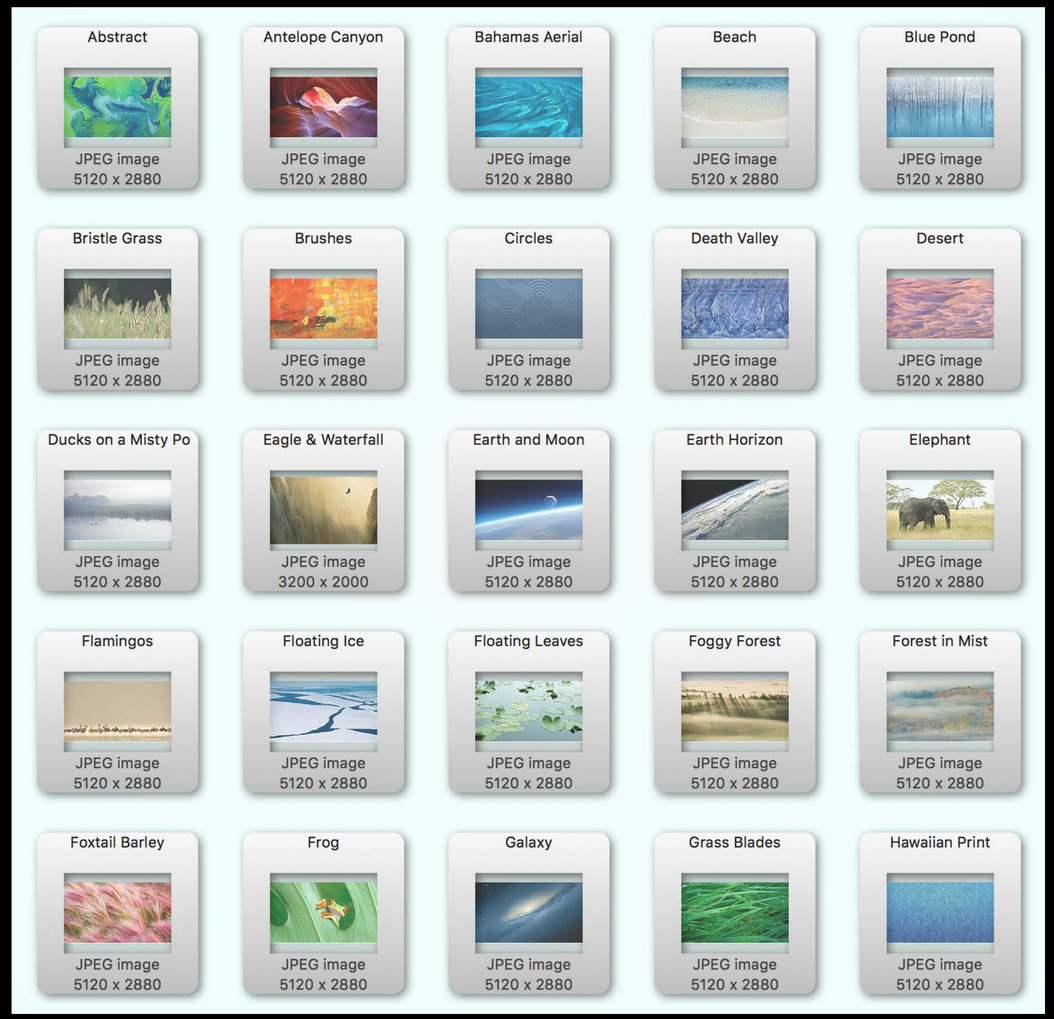
https://developer.apple.com/library/prerelease/mac/samplecode/CocoaSlideCollection

Example

“Cocoa Slide Collection”

Browse a folder of image files

Show thumbnails and image info



https://developer.apple.com/library/prerelease/mac/samplecode/CocoaSlideCollection

Example

“Cocoa Slide Collection”

Browse a folder of image files

Show thumbnails and image info

Position using Flow and custom layouts

https://developer.apple.com/library/prerelease/mac/samplecode/CocoaSlideCollection

Example

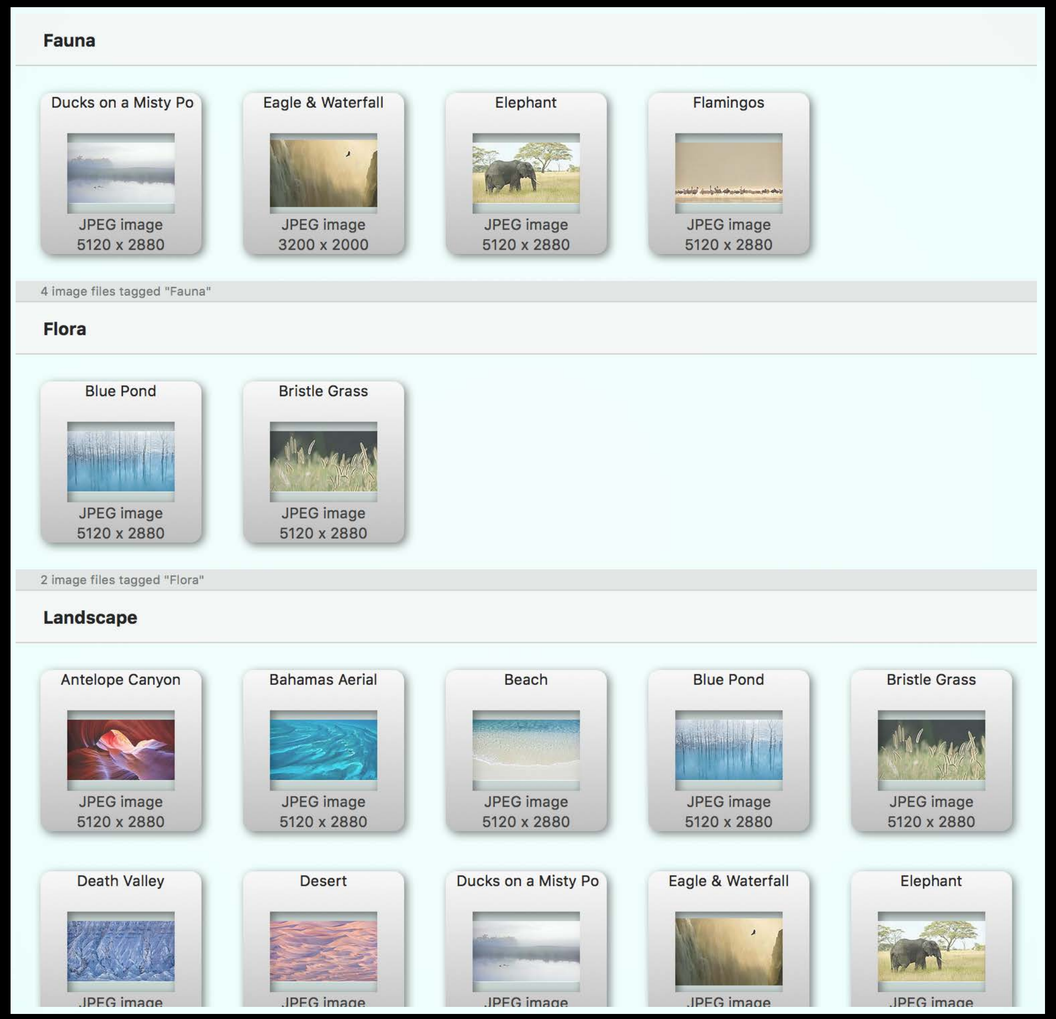
“Cocoa Slide Collection”

Browse a folder of image files

Show thumbnails and image info

Position using Flow and custom layouts

Group images by tag



https://developer.apple.com/library/prerelease/mac/samplecode/CocoaSlideCollection

Example

“Cocoa Slide Collection”

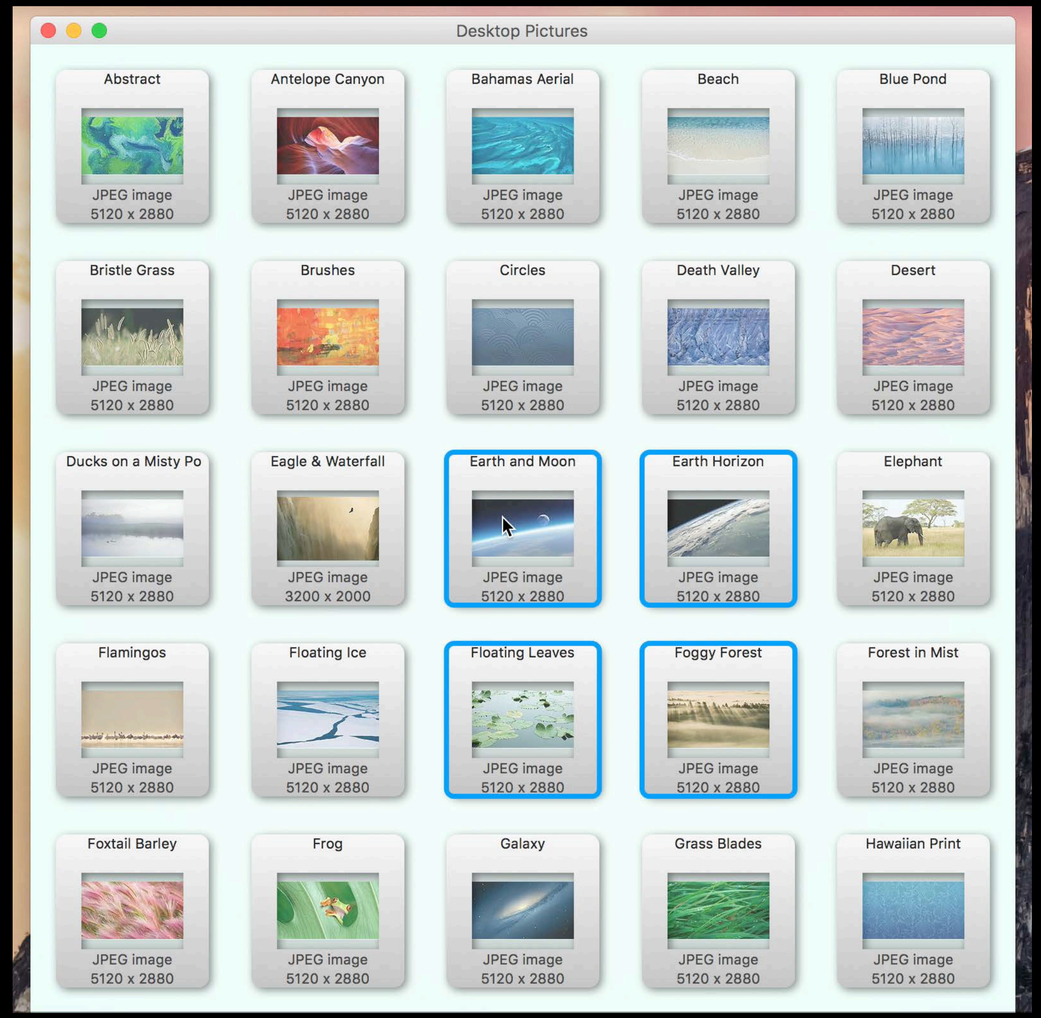
Browse a folder of image files

Show thumbnails and image info

Position using Flow and custom layouts

Group images by tag

Support selection, Drag and Drop



https://developer.apple.com/library/prerelease/mac/samplecode/CocoaSlideCollection

Roadmap

Roadmap

Make items appear

Roadmap

Make items appear

Group items into sections

Roadmap

Make items appear

Group items into sections

Update when model changes

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag and Drop

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag and Drop

Customize layout

Roadmap

Make items appear

Group items into sections

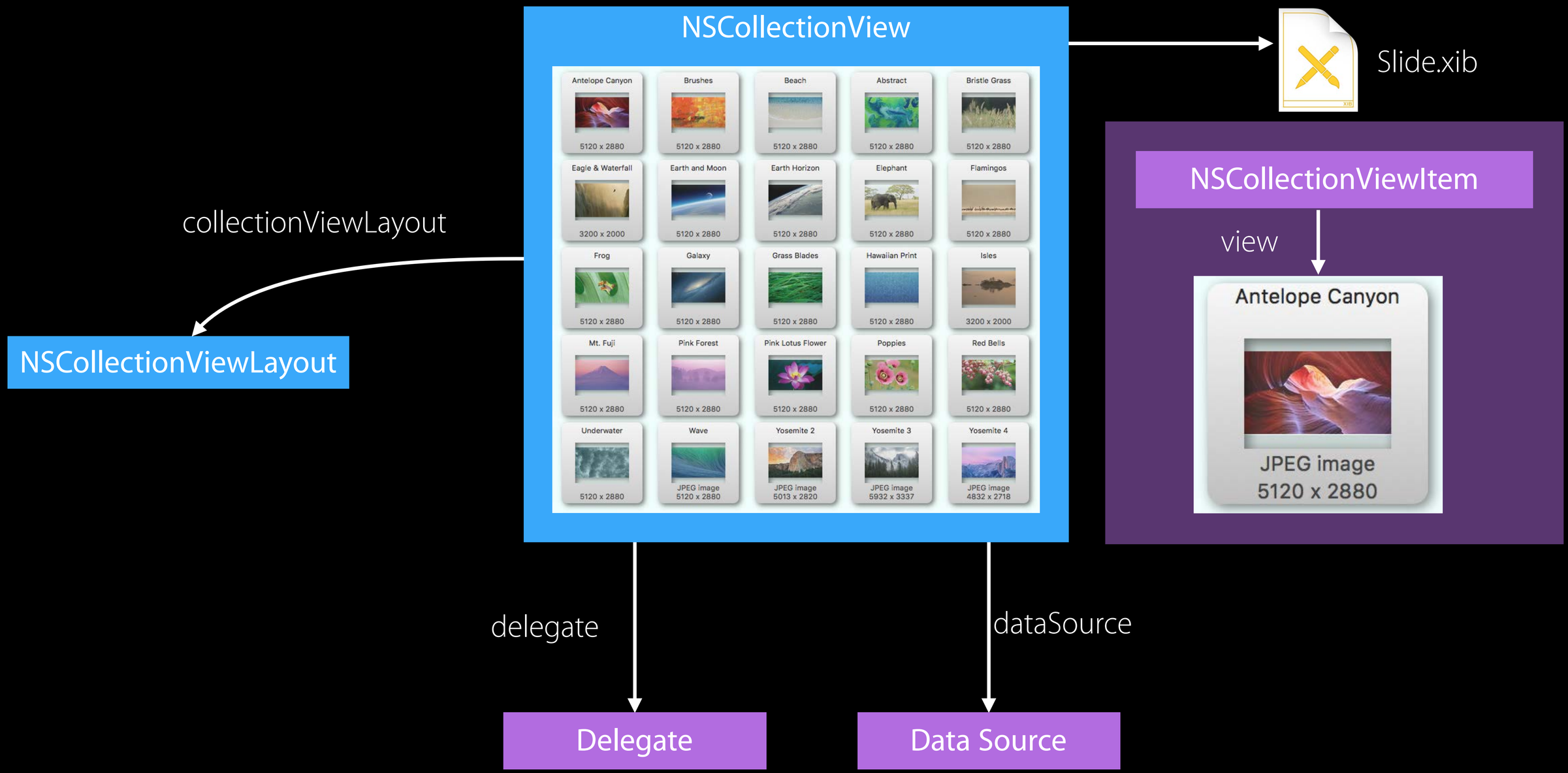
Update when model changes

Handle selection and highlighting

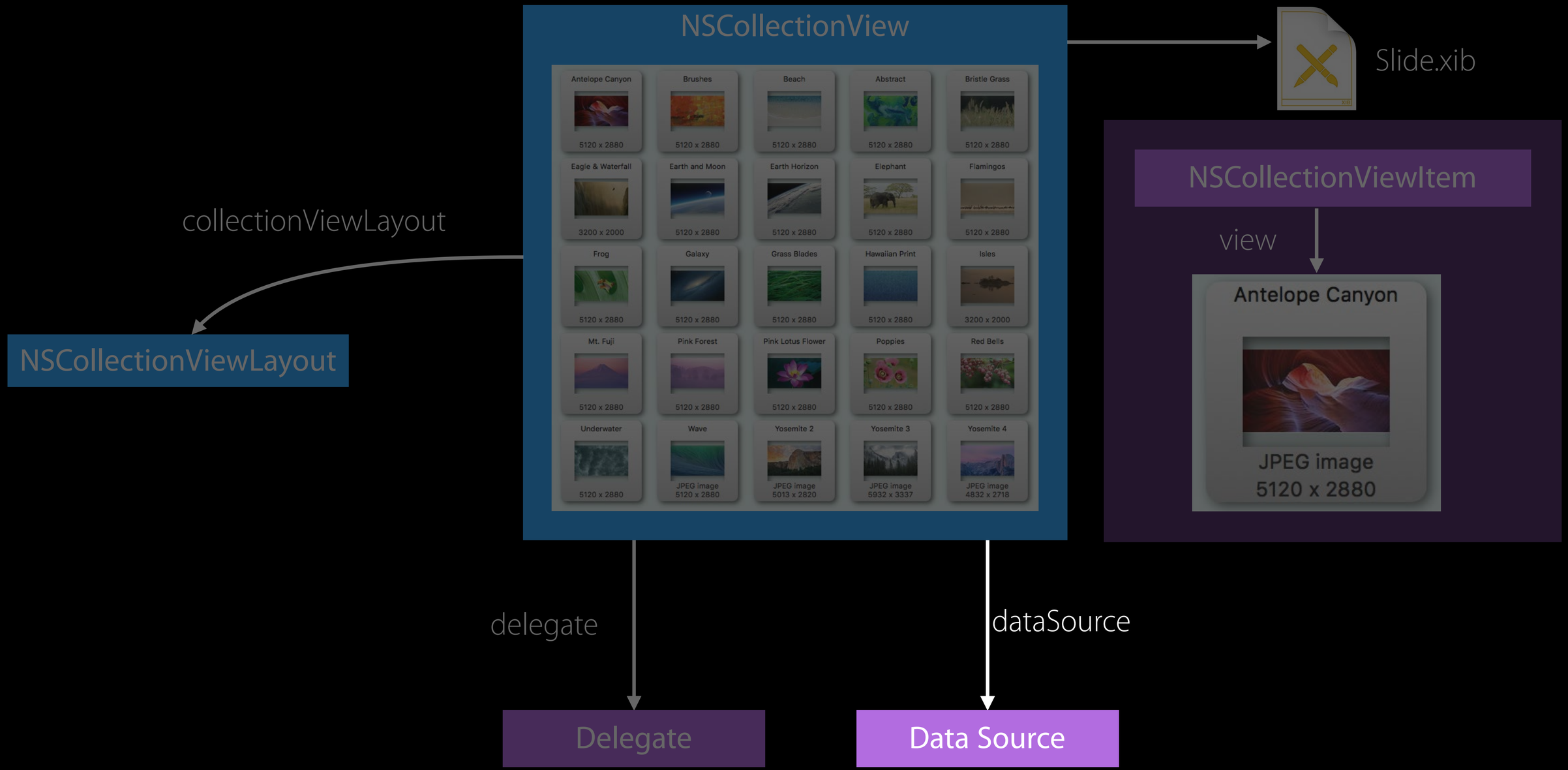
Handle Drag-and-Drop

Customize Layout

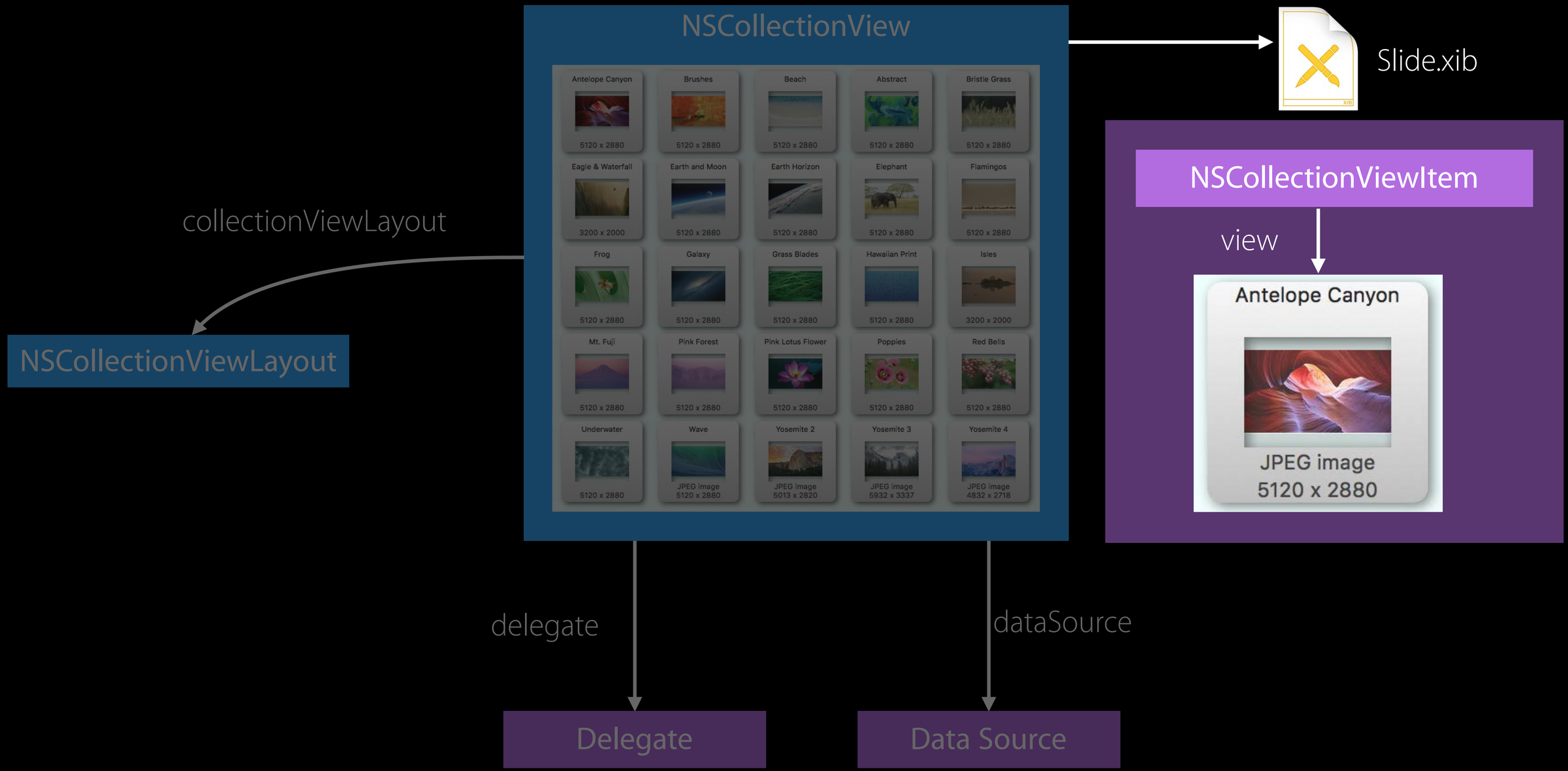
Make Items Appear



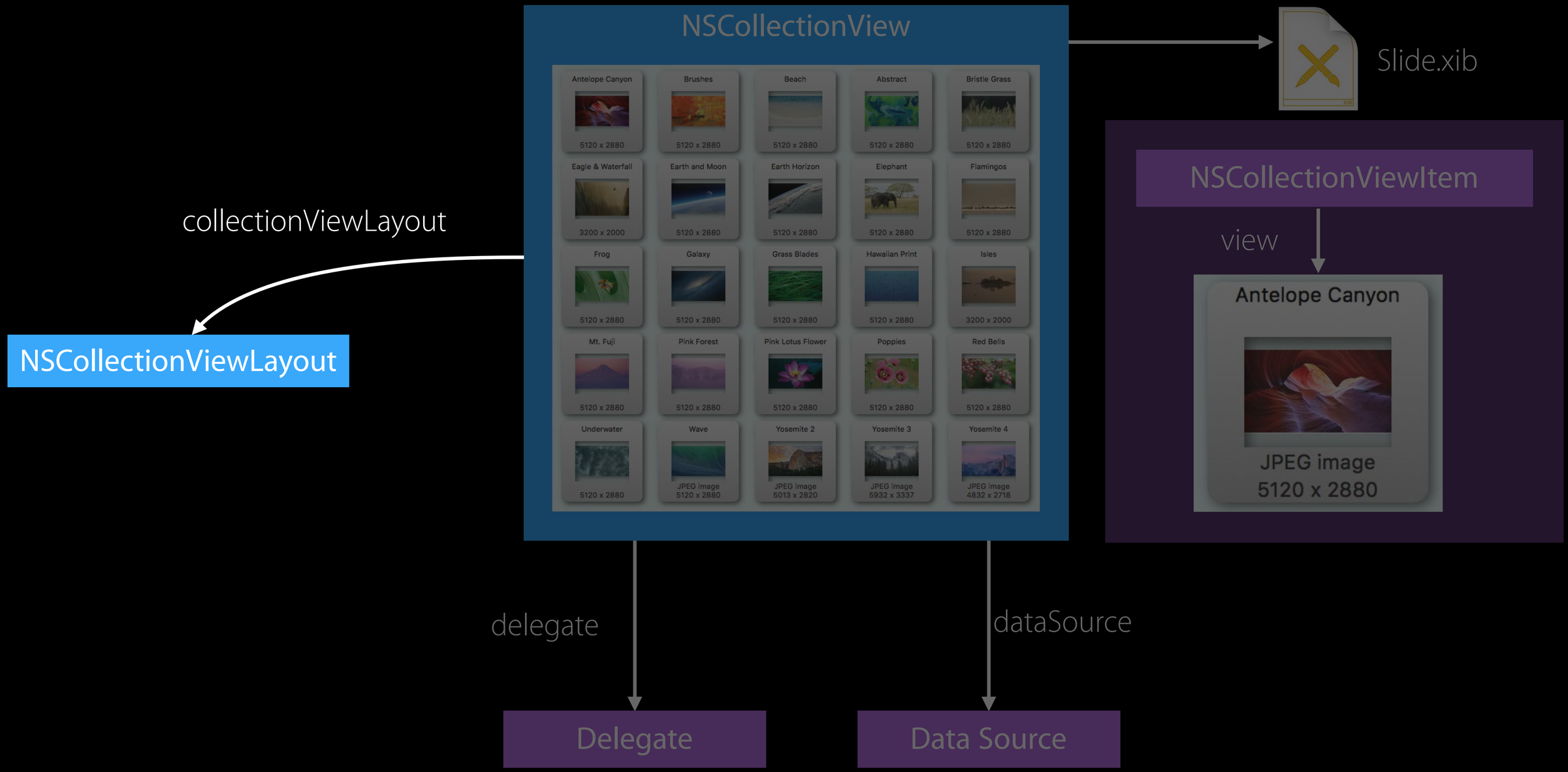
Make Items Appear



Make Items Appear



Make Items Appear



NSCollectionViewDataSource

Two required methods

NSCollectionViewDataSource

Two required methods

```
func numberOfItemsInSection(Int) -> Int

func collectionView(NSCollectionView,
     itemForRepresentedObjectAtIndexPath: NSIndexPath) ->
                                                NSCollectionViewItem
```

Example

"Cocoa Slide Collection"

Example

“Cocoa Slide Collection”

ImageFile–our Model object

Example

“Cocoa Slide Collection”

ImageFile–our Model object

- `url` (includes filename)

Example

“Cocoa Slide Collection”

ImageFile–our Model object

- `url` (includes filename)
- `fileType` (UTI)

Example

“Cocoa Slide Collection”

ImageFile–our Model object

- `url` (includes filename)
- `fileType` (UTI)
- `pixelsWide`

Example

“Cocoa Slide Collection”

ImageFile–our Model object

- `url` (includes filename)
- `fileType` (UTI)
- `pixelsWide`
- pixelsHigh

Example

“Cocoa Slide Collection”

ImageFile–our Model object

- `url` (includes filename)
- `fileType` (UTI)
- `pixelsWide`
- pixelsHigh
- previewImage

Demo

Making items appear

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Grouping Items into Sections

Grouping Items into Sections

We'll group our **ImageFiles** by tag

Grouping Items into Sections

We'll group our **ImageFiles** by tag

- For each tag, we have an **NSArray** of **ImageFiles**

Grouping Items into Sections

We'll group our **ImageFiles** by tag

- For each tag, we have an **NSArray** of **ImageFiles**
- An **ImageFile** with many tags will appear many times

Grouping Items into Sections

We'll group our **ImageFiles** by tag

- For each tag, we have an **NSArray** of **ImageFiles**
- An **ImageFile** with many tags will appear many times

We'll give each section a header and footer view

Grouping Items into Sections

We'll group our **ImageFiles** by tag

- For each tag, we have an **NSArray** of **ImageFiles**
- An **ImageFile** with many tags will appear many times

We'll give each section a header and footer view

- As with item types, we provide a nib or class

Grouping Items into Sections

We'll group our **ImageFiles** by tag

- For each tag, we have an **NSArray** of **ImageFiles**
- An **ImageFile** with many tags will appear many times

We'll give each section a header and footer view

- As with item types, we provide a nib or class
- A header or footer is considered a "supplementary view"

Grouping Items into Sections

We'll group our **ImageFiles** by tag

- For each tag, we have an **NSArray** of **ImageFiles**
- An **ImageFile** with many tags will appear many times

We'll give each section a header and footer view

- As with item types, we provide a nib or class
- A header or footer is considered a "supplementary view"
- Implement `NSCollectionViewDataSource`'s `collectionView(_:viewForSupplementaryElementOfKind:indexPath:)`

Demo

Grouping items into sections

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Keeping Model and View in Sync

Keeping Model and View in Sync

Basic operations: Insert, delete, move, reload

Keeping Model and View in Sync

Basic operations: Insert, delete, move, reload

Applicable to both items and sections

Keeping Model and View in Sync

Basic operations: Insert, delete, move, reload

Applicable to both items and sections

Similar approach to View-based NSOutlineView

Keeping Model and View in Sync

Basic operations: Insert, delete, move, reload

Applicable to both items and sections

Similar approach to View-based NSOutlineView

- When model changes, dataSource must notify CollectionView, describing the change

Keeping Model and View in Sync

Basic operations: Insert, delete, move, reload

Applicable to both items and sections

Similar approach to View-based NSOutlineView

- When model changes, dataSource must notify CollectionView, describing the change

By default, changes appear instantly

```
collectionView.insertItemsAtIndexPaths(insertionIndexPaths)
```

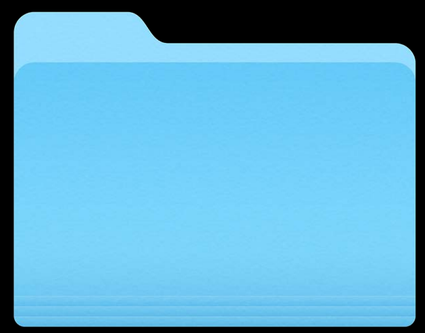
To animate a change, message through the CollectionView's animator (inherited from NSView)

```
collectionView.animator.insertItemsAtIndexPaths(insertionIndexPaths)
```

In CocoaSlideCollection

In CocoaSlideCollection

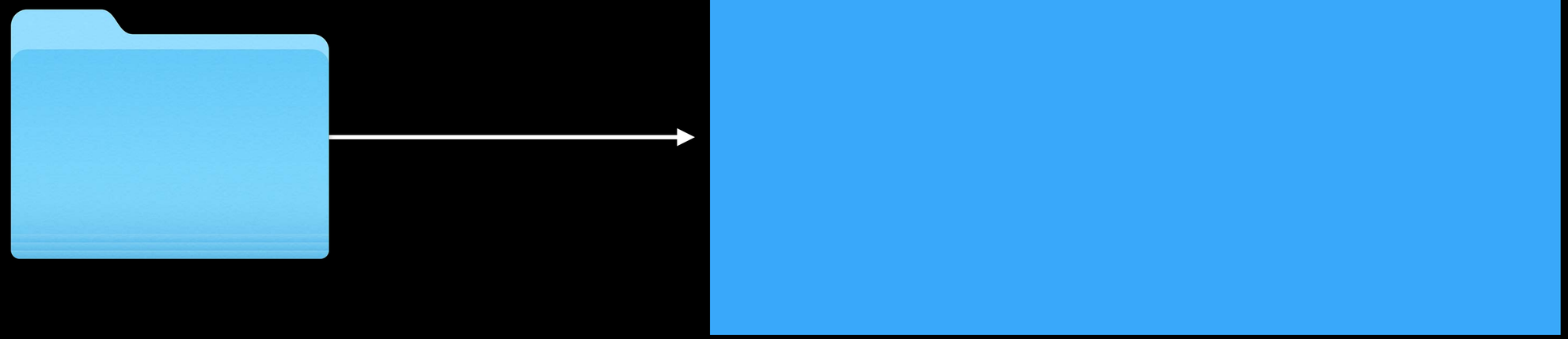
Watch image folder for changes



In CocoaSlideCollection

Watch image folder for changes

Update our CollectionView to match

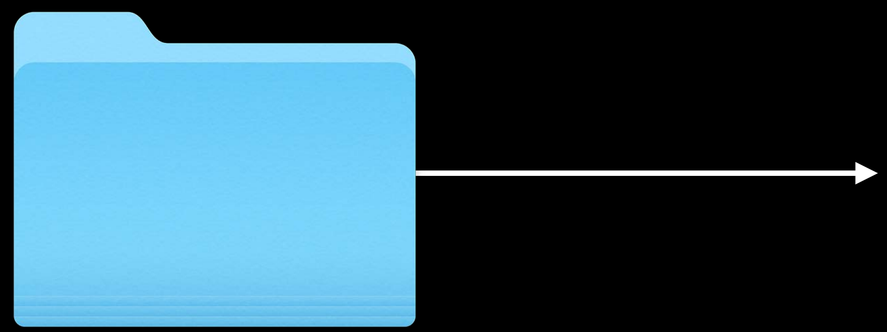


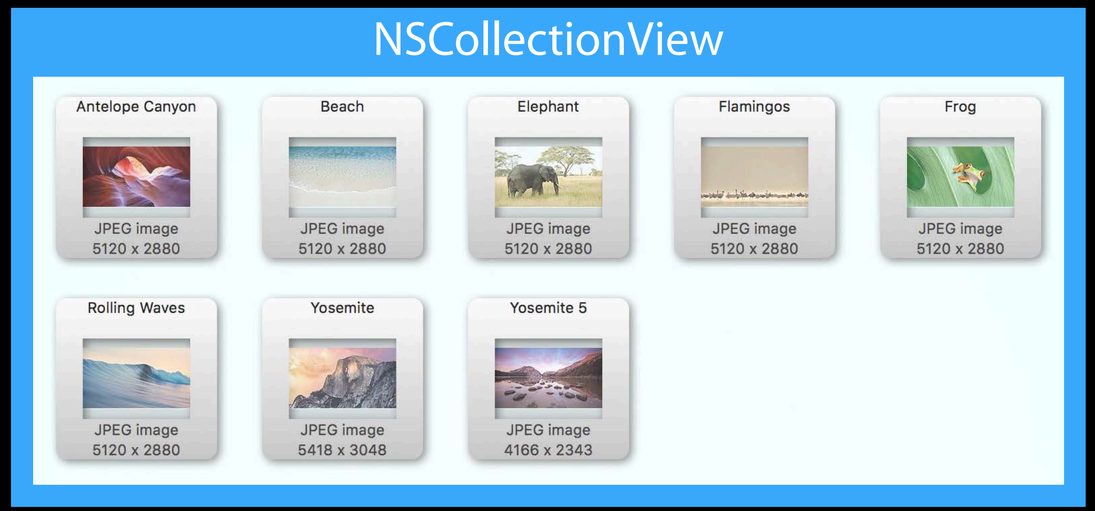
In CocoaSlideCollection

Watch image folder for changes

Update our CollectionView to match

ImageFiles may be added, deleted, or changed





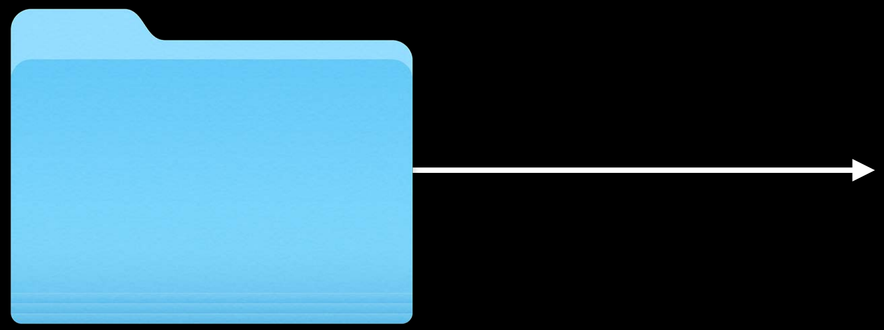
In CocoaSlideCollection

Watch image folder for changes

Update our CollectionView to match

ImageFiles may be added, deleted, or changed

Key-Value Observing (KVO) is our friend



Demo

Keeping model and view in sync

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Roadmap

Make items appear

Group items into sections

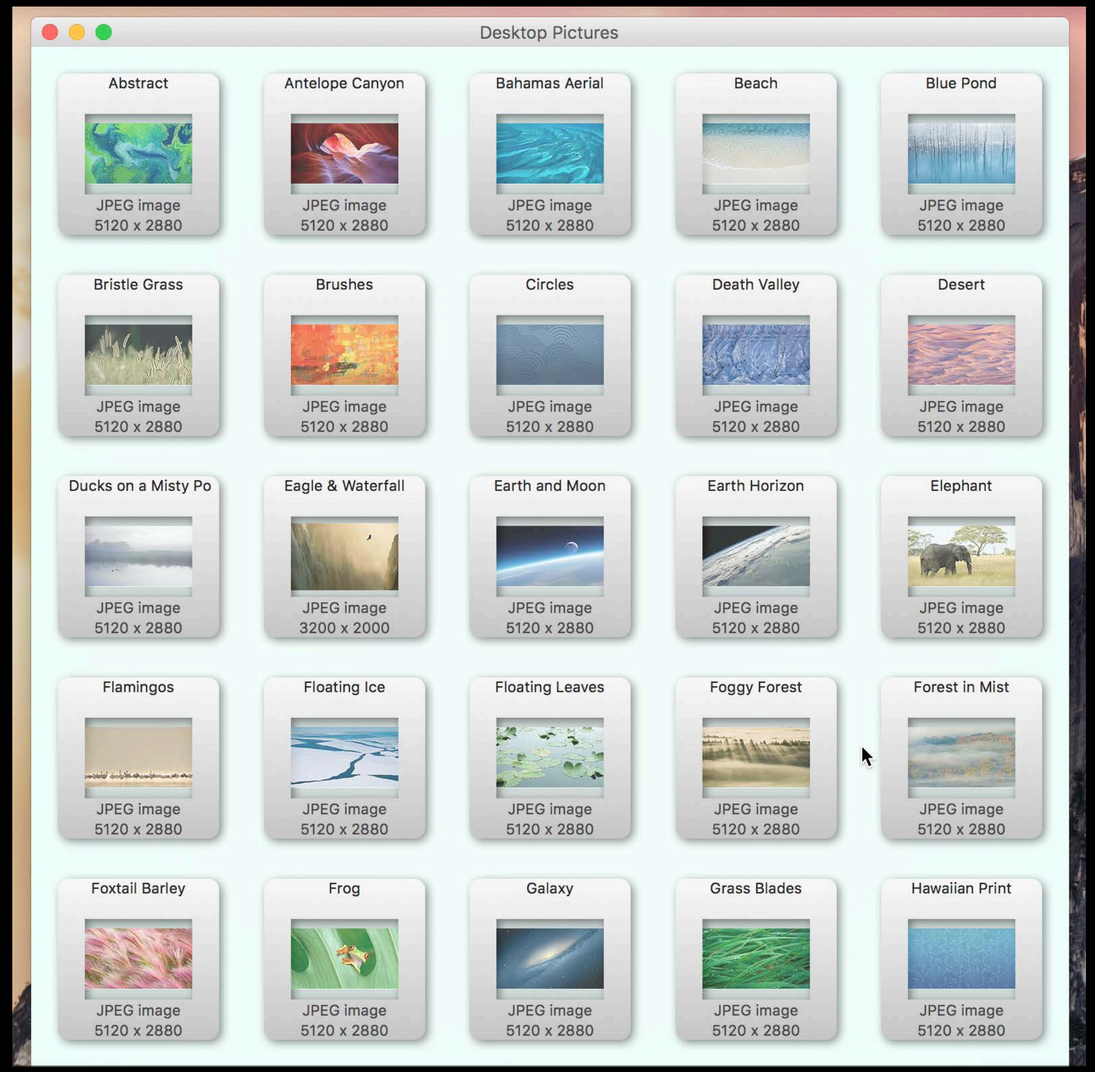
Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

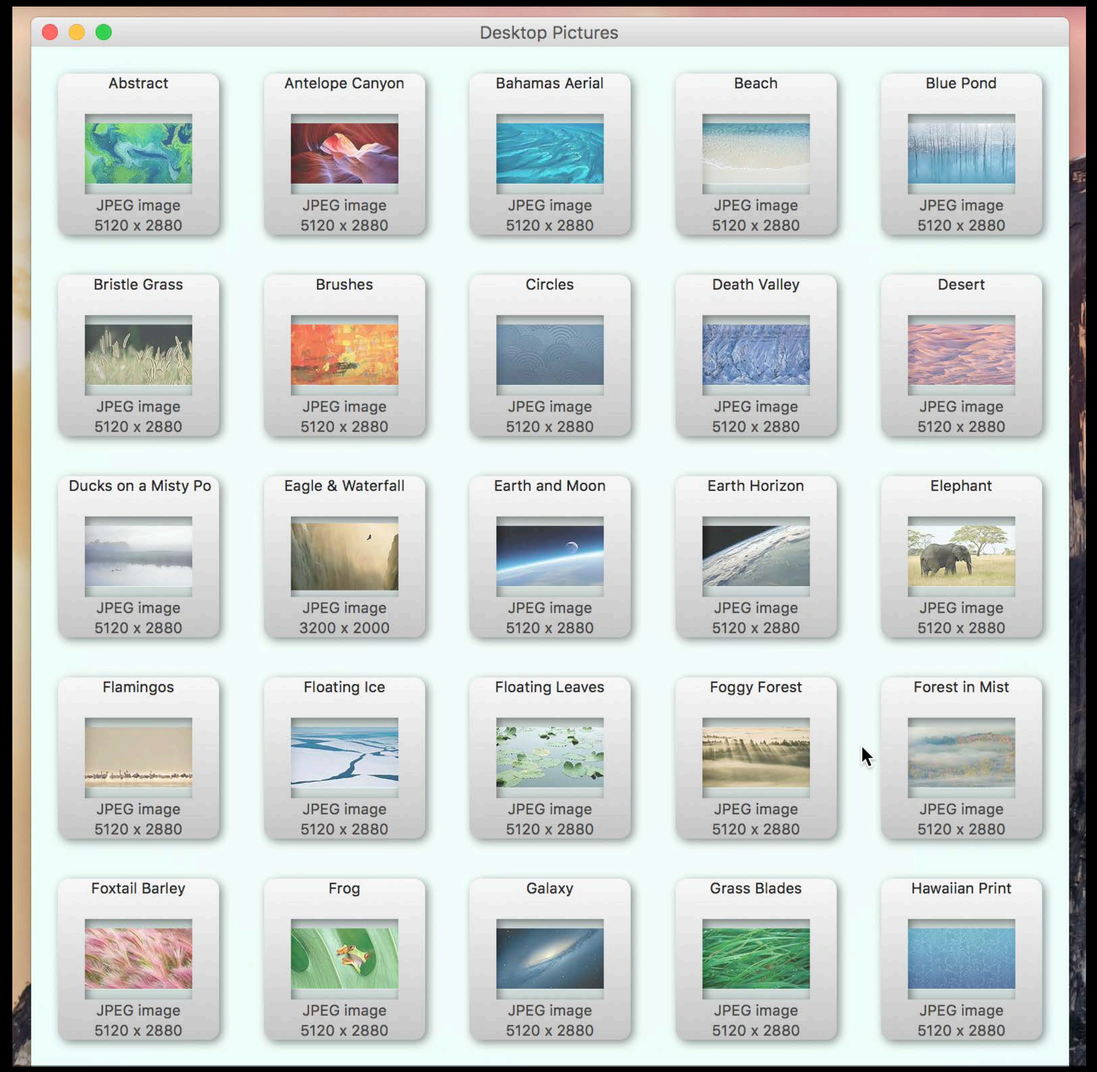
Customize Layout

Selection and Highlighting



Selection and Highlighting

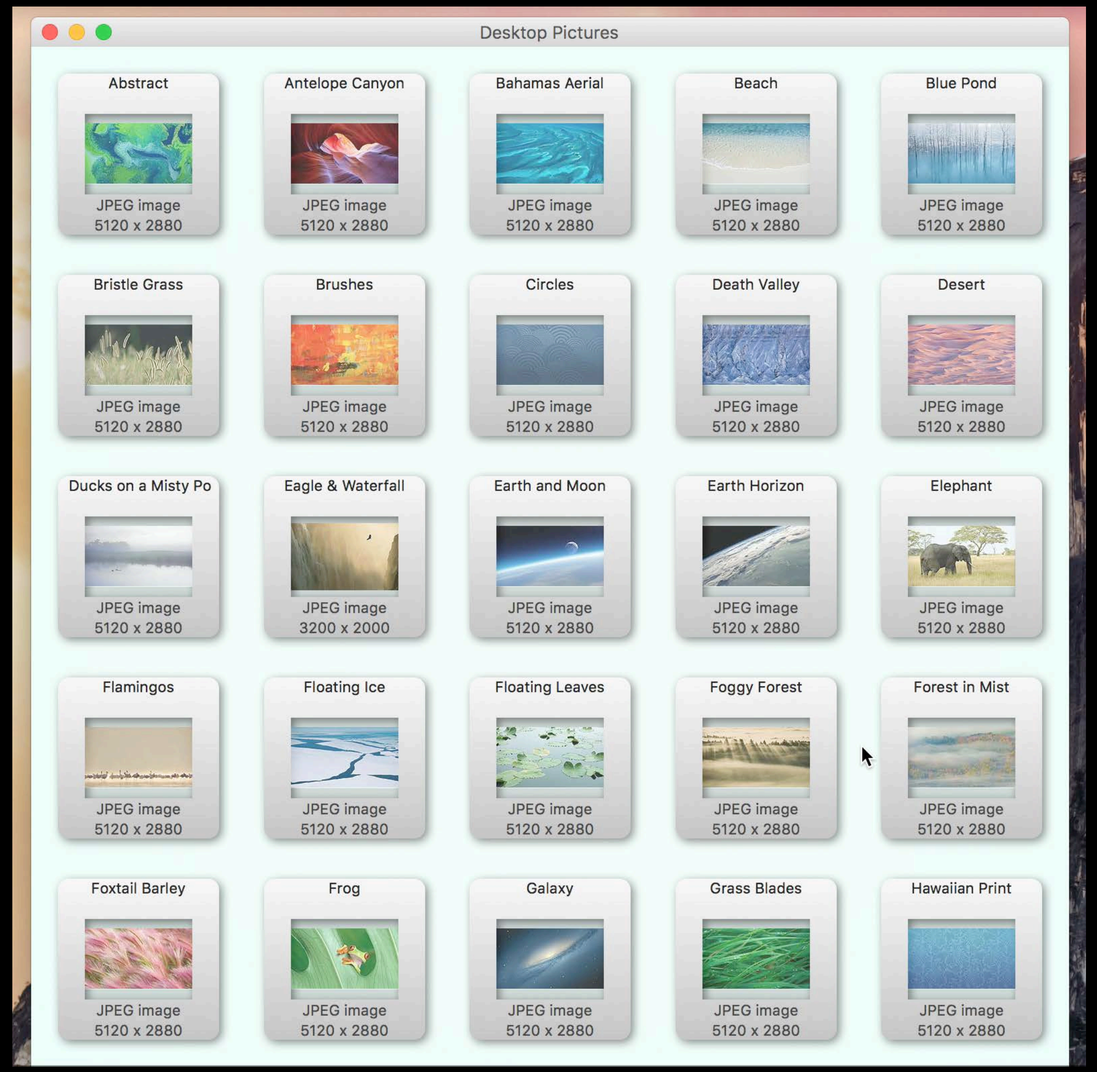
Visually indicated states



Selection and Highlighting

Visually indicated states

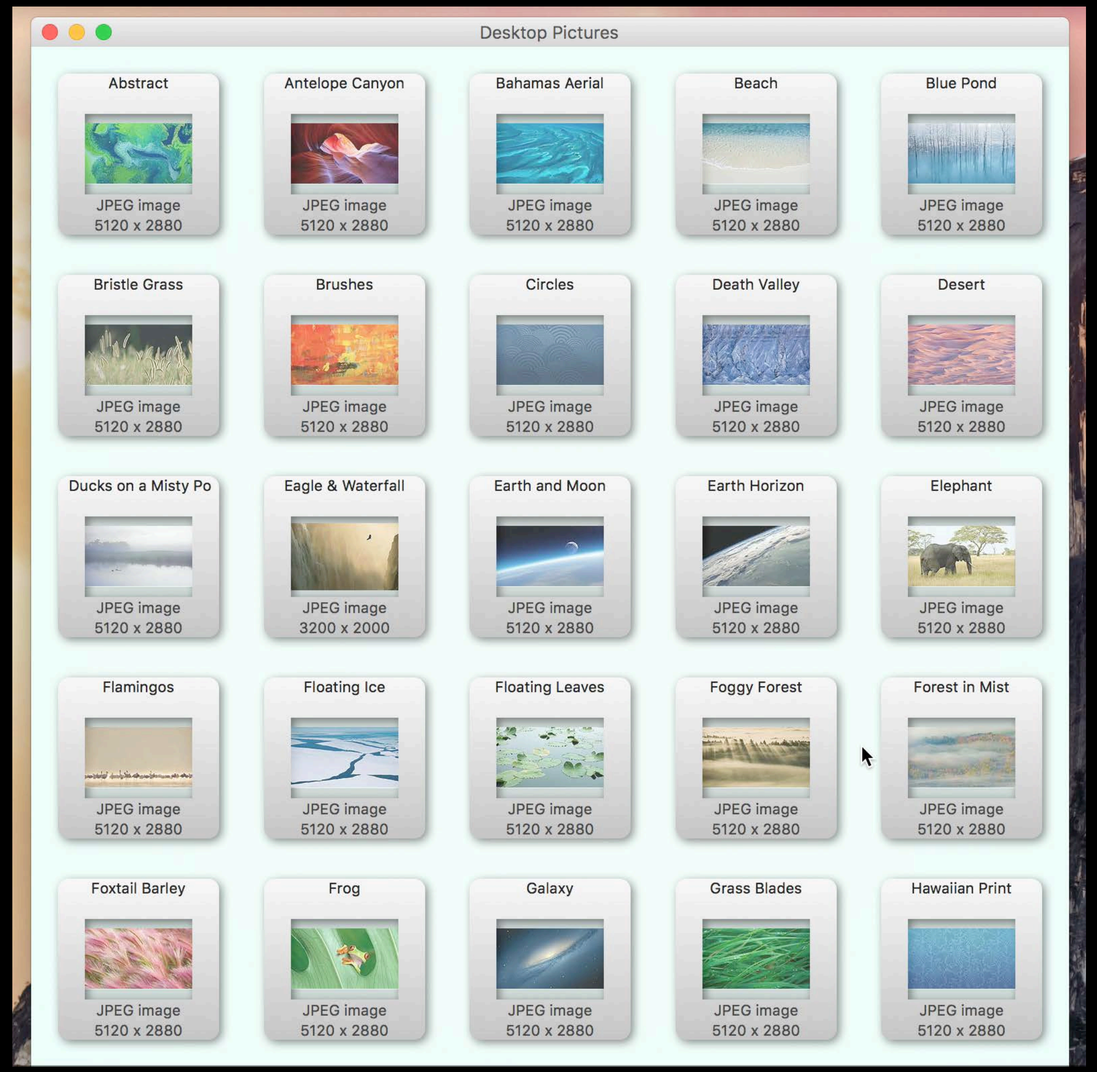
Highlighting precedes selection



Selection and Highlighting

Visually indicated states

Highlighting precedes selection



Highlighting

NEW

Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

```
class NSCollectionViewItem {
    var highlightState: NSCollectionViewItemHighlightState
```

Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

```
class NSCollectionViewItem {
    var highlightState: NSCollectionViewItemHighlightState
```

```
enum NSCollectionViewItemHighlightState : Int {

    case     None,       ForSelection,  ForDeselection,  AsDropTarget
}
```

Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

```
class NSCollectionViewItem {
    var highlightState: NSCollectionViewItemHighlightState
```

```
enum NSCollectionViewItemHighlightState : Int {

    case     None,      ForSelection,  ForDeselection,  AsDropTarget
}
```

Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

```
class NSCollectionViewItem {
    var highlightState: NSCollectionViewItemHighlightState
```

```
enum NSCollectionViewItemHighlightState : Int {
```


```
    case     None,       ForSelection,  ForDeselection,  AsDropTarget
}
```

Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

```
class NSCollectionViewItem {
    var highlightState: NSCollectionViewItemHighlightState
```

```
enum NSCollectionViewItemHighlightState : Int {
```


```
    case None, ForSelection, ForDeselection, AsDropTarget
}
```

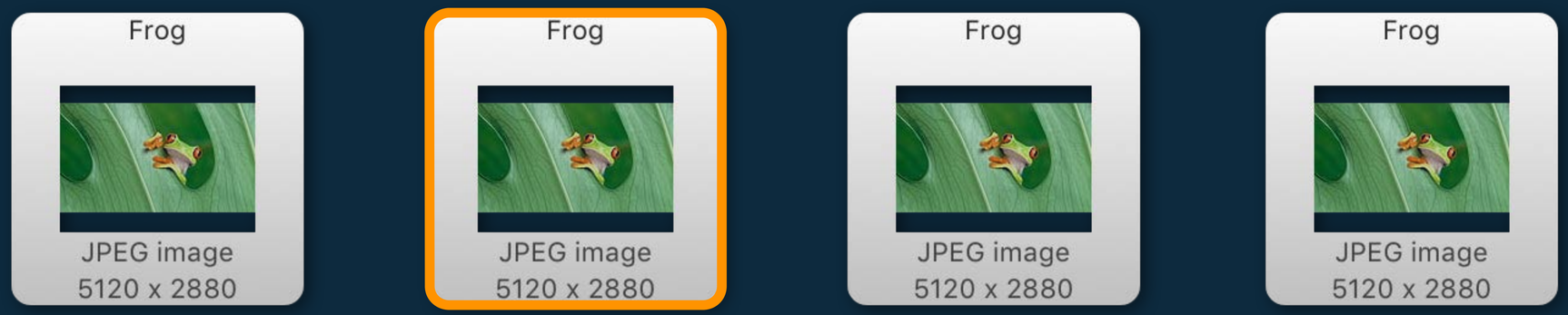
Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

```
class NSCollectionViewItem {
    var highlightState: NSCollectionViewItemHighlightState
```

```
enum NSCollectionViewItemHighlightState : Int {
```




```
    case None, ForSelection, ForDeselection, AsDropTarget
}
```

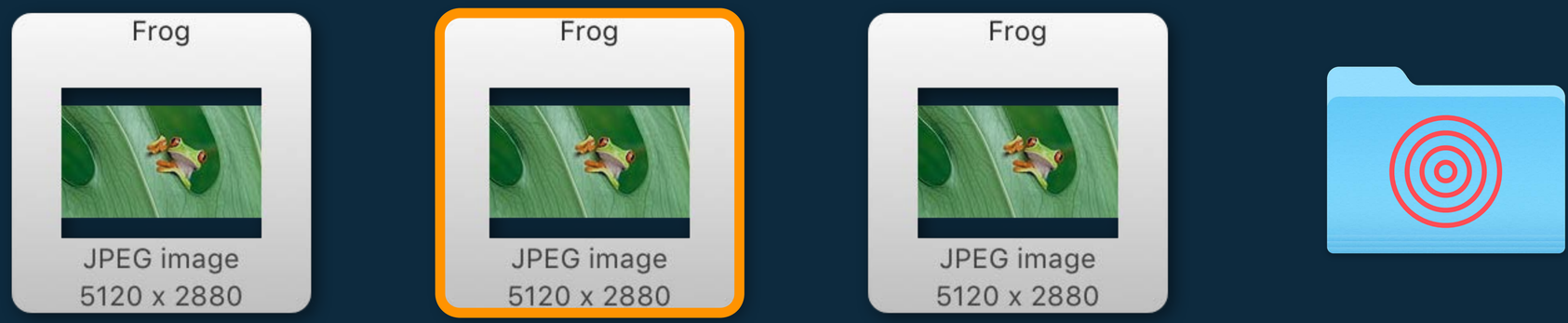
Highlighting

NEW

`item.highlightState` indicates proposed selection, deselection, or drop target status

```
class NSCollectionViewItem {
    var highlightState: NSCollectionViewItemHighlightState
```

```
enum NSCollectionViewItemHighlightState : Int {
```




```
    case None, ForSelection, ForDeselection, AsDropTarget
}
```

Highlighting Using Layer Properties

Highlighting Using Layer Properties

New NSCollectionViews always layer-backed

Highlighting Using Layer Properties

New NSCollectionViews always layer-backed

Can use layer properties to easily change item appearance without redraw

Highlighting Using Layer Properties

New NSCollectionViews always layer-backed

Can use layer properties to easily change item appearance without redraw

- `backgroundColor`, `borderColor`, `borderWidth`, `cornerRadius`

Highlighting Using Layer Properties

New NSCollectionViews always layer-backed

Can use layer properties to easily change item appearance without redraw

- `backgroundColor`, `borderColor`, `borderWidth`, `cornerRadius`

Highlighting Using Layer Properties

New NSCollectionViews always layer-backed

Can use layer properties to easily change item appearance without redraw

- `backgroundColor`, `borderColor`, `borderWidth`, `cornerRadius`

```
item.view.layer.backgroundColor = brightColor
```

Highlighting Using Layer Properties

New NSCollectionViews always layer-backed

Can use layer properties to easily change item appearance without redraw

- `backgroundColor`, `borderColor`, `borderWidth`, `cornerRadius`

```
item.view.layer.backgroundColor = brightColor
item.view.layer.cornerRadius = 8
```

When to Apply Highlighting

When to Apply Highlighting

When "`highlightState`" changes

When to Apply Highlighting

When "highlightState" changes

```
class MyItem : NSCollectionViewItem {
    override var highlightState: NSCollectionViewItemHighlightState {
        didSet {
            // Update this item's appearance accordingly.
        }
    }
}
```

Selection

Selection

Items can be selected

Selection

Items can be selected

NSCollectionView supports single or multiple selection

Selection

Items can be selected

NSCollectionView supports single or multiple selection

```
var selectable: Bool
```

Selection

Items can be selected

NSCollectionView supports single or multiple selection

```
var selectable: Bool

var allowsMultipleSelection: Bool
```

Selection

Items can be selected

NSCollectionView supports single or multiple selection

```
var selectable: Bool

var allowsMultipleSelection: Bool

var allowsEmptySelection: Bool
```

Tracking the Selection

Tracking the Selection

`selectionIndexPaths` tracks the set of selected items

Tracking the Selection

`selectionIndexPaths` tracks the set of selected items

```
var selectionIndexPaths: Set<NSIndexPath>
```

Tracking the Selection

`selectionIndexPaths` tracks the set of selected items

```
var selectionIndexPaths: Set<NSIndexPath>
```

An item knows whether it is part of the selection

Tracking the Selection

`selectionIndexPaths` tracks the set of selected items

```
var selectionIndexPaths: Set<NSIndexPath>
```

An item knows whether it is part of the selection

```
var selected: Bool
```

Programmatic Selection

```
var selectionIndexPaths: Set<NSIndexPath>
```

Programmatic Selection

```
var selectionIndexPaths: Set<NSIndexPath>
```

Programmatic Selection

```
var selectionIndexPaths: Set<NSIndexPath>

func selectItemsAtIndexPaths(Set<NSIndexPath>,
  scrollPosition: NSCollectionViewScrollPosition)

func deselectItemsAtIndexPaths(Set<NSIndexPath>)
```

User Selection

User Selection

Delegate can approve selection and deselection

User Selection

Delegate can approve selection and deselection

```
optional func collectionView(NSCollectionView,
             shouldSelectItemsAtIndexPaths: Set<NSIndexPath>)
             -> Set<NSIndexPath>

optional func collectionView(NSCollectionView,
             shouldDeselectItemsAtIndexPaths: Set<NSIndexPath>)
             -> Set<NSIndexPath>
```

User Selection

Delegate can approve selection and deselection

Return a different set of NSIndexPaths to override the proposed change

```
optional func collectionView(NSCollectionView,
             shouldSelectItemsAtIndexPaths: Set<NSIndexPath>)
             -> Set<NSIndexPath>

optional func collectionView(NSCollectionView,
             shouldDeselectItemsAtIndexPaths: Set<NSIndexPath>)
             -> Set<NSIndexPath>
```

User Selection

Delegate is notified of selection and deselection

```
optional func collectionView(NSCollectionView,
             didSelectItemsAtIndexPaths: Set<NSIndexPath>)

optional func collectionView(NSCollectionView,
             didDeselectItemsAtIndexPaths: Set<NSIndexPath>)
```

Highlighting

Highlighting

Delegate can also approve `highlightState` changes

Highlighting

Delegate can also approve `highlightState` changes

```
optional func collectionView(NSCollectionView,
             shouldChangeItemsAtIndexPaths: Set<NSIndexPath>,
             toHighlightState: NSCollectionViewItemHighlightState)
             -> Set<NSIndexPath>

optional func collectionView(NSCollectionView,
             didChangeItemsAtIndexPaths: Set<NSIndexPath>,
             toHighlightState: NSCollectionViewItemHighlightState)
```

Demo

Selection and highlighting

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Roadmap

Make items appear

Group items into sections

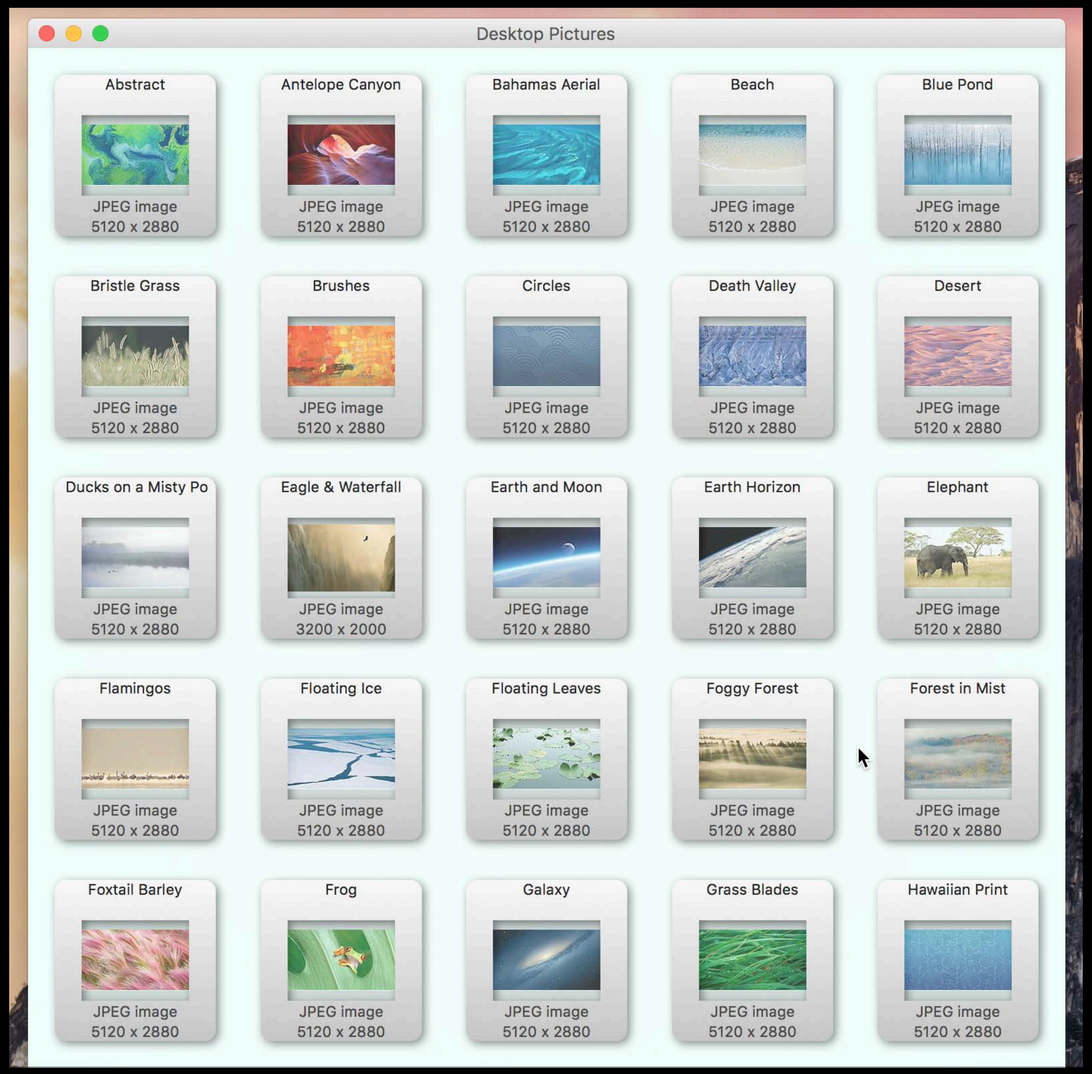
Update when model changes

Handle selection and highlighting

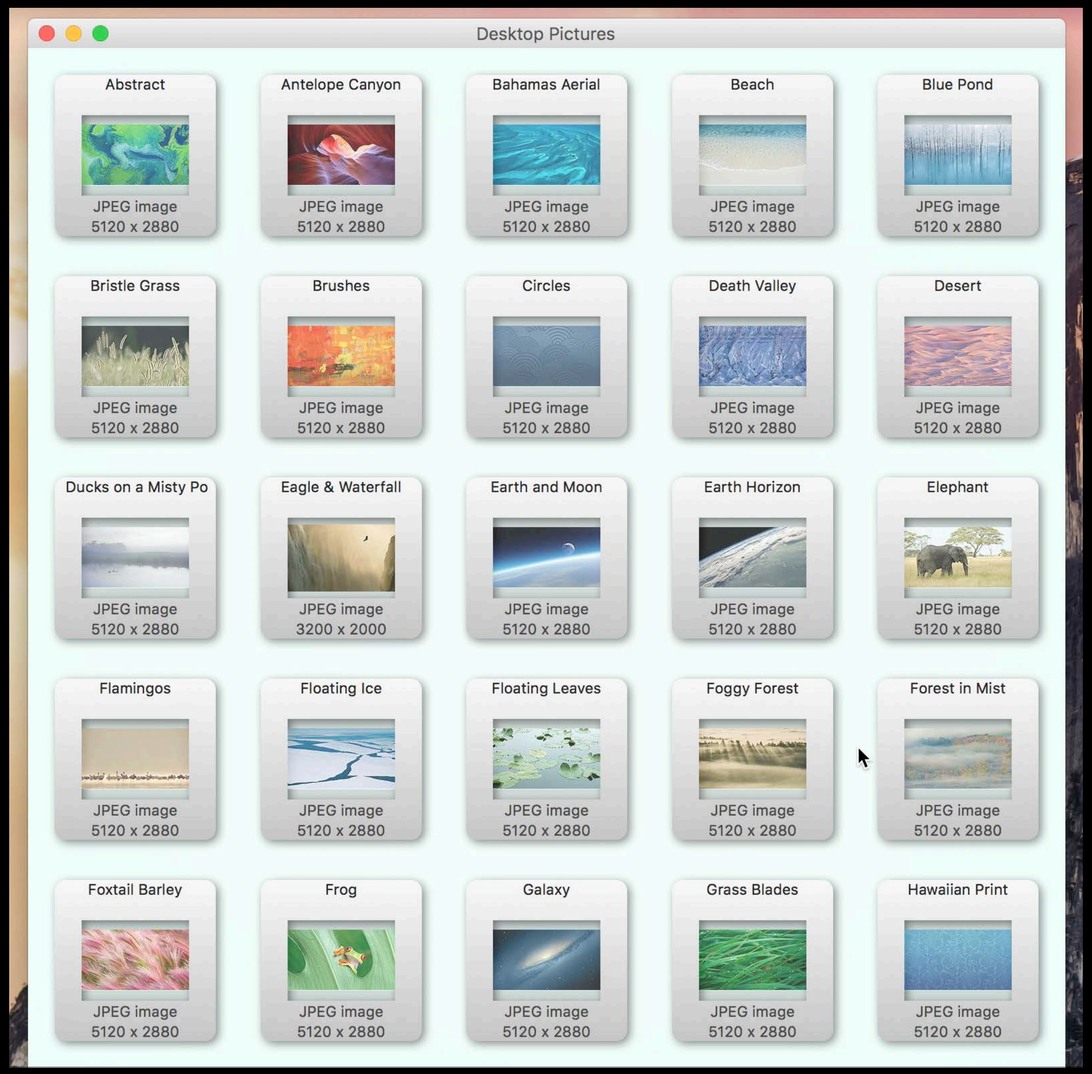
Handle Drag and Drop

Customize Layout

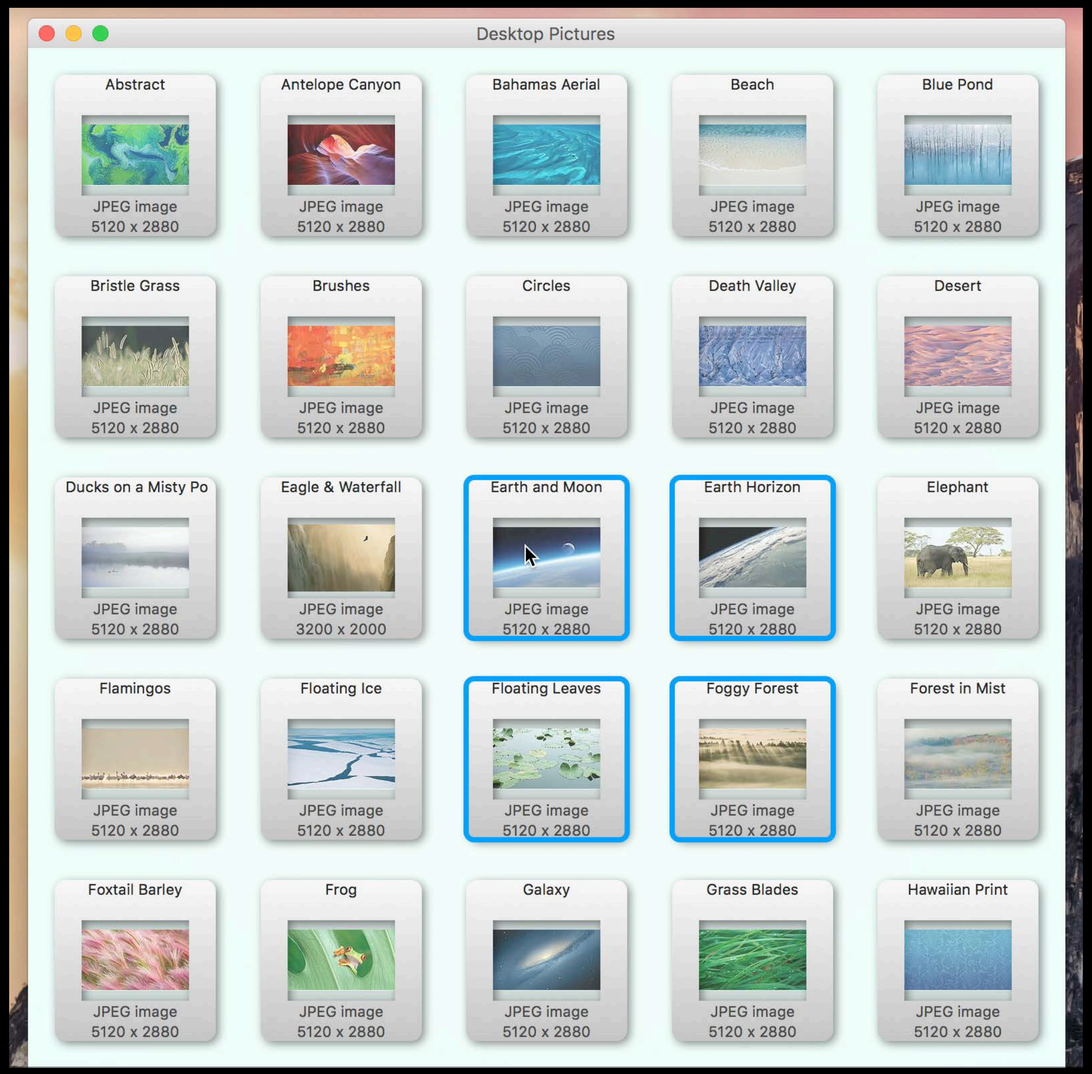
Handling Drag and Drop



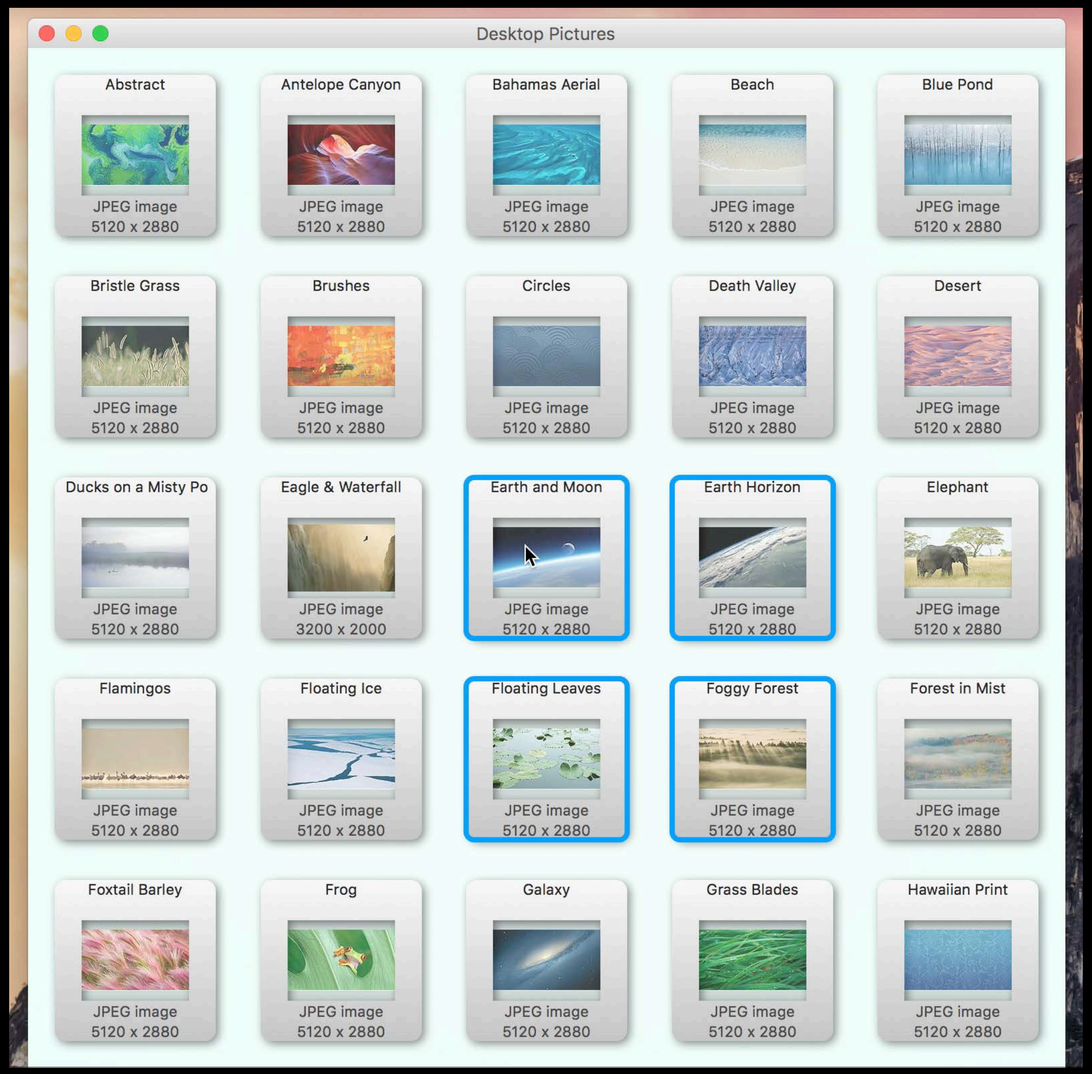
Handling Drag and Drop



Handling Drag and Drop



Handling Drag and Drop



Handling Drag and Drop

NSCollectionView's `delegate` implements Drag-and-Drop response

Handling Drag and Drop

NSCollectionView's `delegate` implements Drag-and-Drop response

Similar to NSOutlineView's API (c.f., "DragAndDropOutlineView")

Handling Drag and Drop

NSCollectionView's `delegate` implements Drag-and-Drop response

Similar to NSOutlineView's API (c.f., "DragAndDropOutlineView")

Dragging Source

Handling Drag and Drop

NSCollectionView's `delegate` implements Drag-and-Drop response

Similar to NSOutlineView's API (c.f., "DragAndDropOutlineView")

Dragging Source

- Put items on pasteboard when requested

Handling Drag and Drop

NSCollectionView's `delegate` implements Drag-and-Drop response

Similar to NSOutlineView's API (c.f., "DragAndDropOutlineView")

Dragging Source

- Put items on pasteboard when requested

Dragging Destination

Handling Drag and Drop

NSCollectionView's `delegate` implements Drag-and-Drop response

Similar to NSOutlineView's API (c.f., "DragAndDropOutlineView")

Dragging Source

- Put items on pasteboard when requested

Dragging Destination

- Assess proposed drop objects, target position, and operation; optionally override

Handling Drag and Drop

NSCollectionView's `delegate` implements Drag-and-Drop response

Similar to NSOutlineView's API (c.f., "DragAndDropOutlineView")

Dragging Source

- Put items on pasteboard when requested

Dragging Destination

- Assess proposed drop objects, target position, and operation; optionally override
- Implement drop acceptance

Drag-and-Drop Requirements

Drag-and-Drop Requirements

```
registerForDraggedTypes(_:)
```

Drag-and-Drop Requirements

```
registerForDraggedTypes(_:)
setDraggingSourceOperationMask(_:forLocal:)
```

Drag-and-Drop Requirements

`registerForDraggedTypes(_:)`

`setDraggingSourceOperationMask(_:forLocal:)`

Implement the required delegate methods

Drag-and-Drop Requirements

`registerForDraggedTypes(_:)`

`setDraggingSourceOperationMask(_:forLocal:)`

Implement the required delegate methods

- Source

Drag-and-Drop Requirements

`registerForDraggedTypes(_:)`

`setDraggingSourceOperationMask(_:forLocal:)`

Implement the required delegate methods

- Source
  - `collectionView(_:pasteboardWriterForItemAtIndexPath:)`, or

Drag-and-Drop Requirements

`registerForDraggedTypes(_:)`

`setDraggingSourceOperationMask(_:forLocal:)`

Implement the required delegate methods

- Source
  - `collectionView(_:pasteboardWriterForItemAtIndexPath:)`, or
  - `collectionView(_:writeItemsAtIndexPaths:toPasteboard:)`

Drag-and-Drop Requirements

`registerForDraggedTypes(_:)`

`setDraggingSourceOperationMask(_:forLocal:)`

Implement the required delegate methods

- Source
  - `collectionView(_:pasteboardWriterForItemAtIndexPath:)`, or
  - `collectionView(_:writeItemsAtIndexPaths:toPasteboard:)`
- Destination

Drag-and-Drop Requirements

`registerForDraggedTypes(_:)`

`setDraggingSourceOperationMask(_:forLocal:)`

Implement the required delegate methods

- Source
  - `collectionView(_:pasteboardWriterForItemAtIndexPath:)`, or
  - `collectionView(_:writeItemsAtIndexPaths:toPasteboard:)`
- Destination
  - `collectionView(_:validateDrop:proposedIndexPath:dropOperation:)`

Drag-and-Drop Requirements

`registerForDraggedTypes(_:)`

`setDraggingSourceOperationMask(_:forLocal:)`

Implement the required delegate methods

- Source
  - `collectionView(_:pasteboardWriterForItemAtIndexPath:)`, or
  - `collectionView(_:writeItemsAtIndexPaths:toPasteboard:)`
- Destination
  - `collectionView(_:validateDrop:proposedIndexPath:dropOperation:)`
  - `collectionView(_:acceptDrop:indexPath:dropOperation:)`

Drag-and-Drop Tips

Drag-and-Drop Tips

It’s worth handling drag within your CollectionView specially

Drag-and-Drop Tips

It's worth handling drag within your CollectionView specially

This lets you *move* items, rather than *remove* and re*insert*

Drag-and-Drop Tips

It’s worth handling drag within your CollectionView specially

This lets you *move* items, rather than *remove* and re*insert*

Stash indexPaths being dragged to know drag came from same CollectionView

Drag-and-Drop Tips

It's worth handling drag within your CollectionView specially

This lets you *move* items, rather than *remove* and re*insert*

Stash indexPaths being dragged to know drag came from same CollectionView

```
collectionView(_:draggingSession:willBeginAtPoint:forItemsAtIndexPaths:)
```

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag and Drop

Customize Layout

Roadmap

Make items appear

Group items into sections

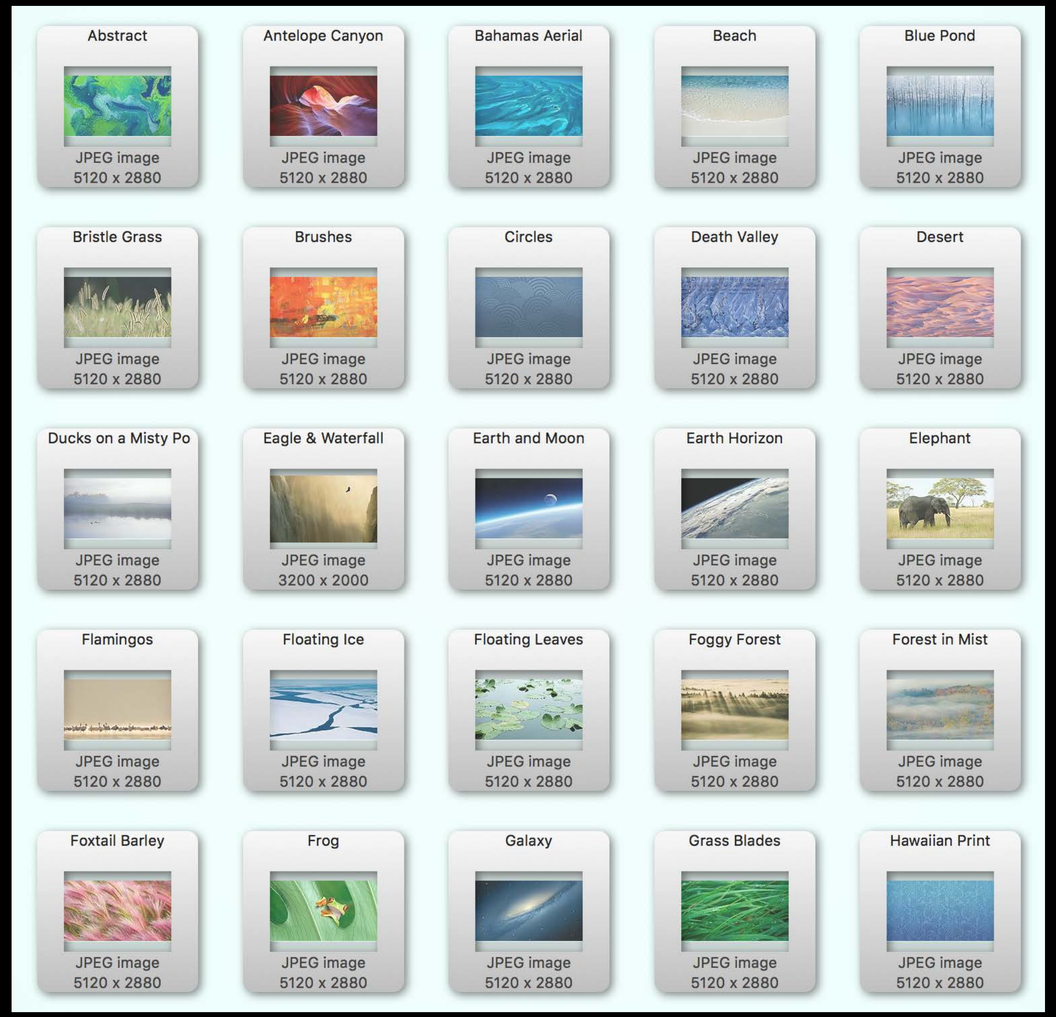
Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

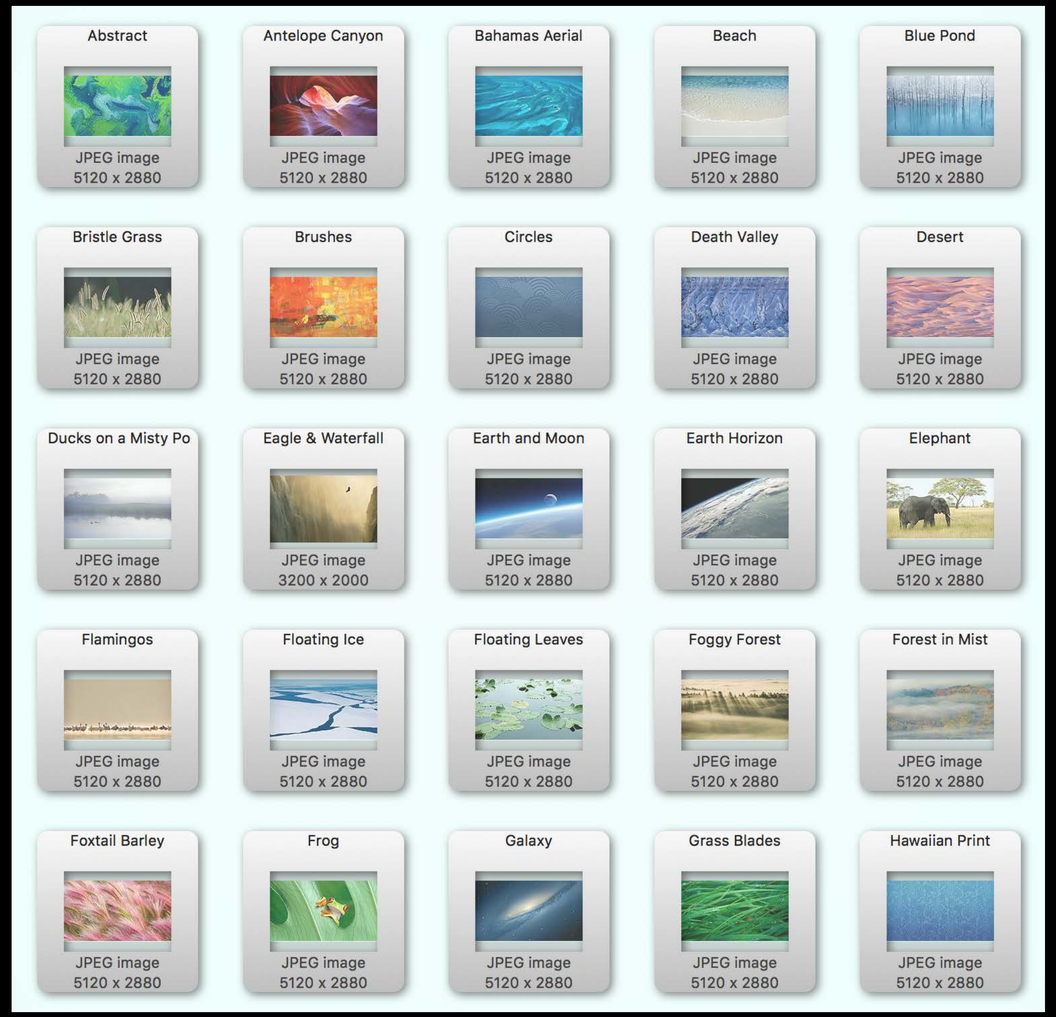
Customize Layout

Customizing Layout



Customizing Layout

Adjust an existing layout



Customizing Layout

Adjust an existing layout

Implement a completely new layout

Adjust an Existing Layout

Adjust an Existing Layout

Subclass `NSCollectionViewFlowLayout` to adjust item positioning

Adjust an Existing Layout

Subclass `NSCollectionViewFlowLayout` to adjust item positioning

Override

Adjust an Existing Layout

Subclass `NSCollectionViewFlowLayout` to adjust item positioning

Override

- `func layoutAttributesForElementsInRect(rect: NSRect) -> [NSCollectionViewLayoutAttributes]`

Adjust an Existing Layout

Subclass `NSCollectionViewFlowLayout` to adjust item positioning

Override

- `func layoutAttributesForElementsInRect(rect: NSRect) -> [NSCollectionViewLayoutAttributes]`
- `func layoutAttributesForItemAtIndexPath(indexPath: NSIndexPath) -> NSCollectionViewLayoutAttributes?`

Adjust an Existing Layout

Subclass `NSCollectionViewFlowLayout` to adjust item positioning

Override

- `func layoutAttributesForElementsInRect(rect: NSRect) -> [NSCollectionViewLayoutAttributes]`
- `func layoutAttributesForItemAtIndexPath(indexPath: NSIndexPath) -> NSCollectionViewLayoutAttributes?`
- `func invalidateLayoutWithContext(context: NSCollectionViewLayoutInvalidationContext)`

Adjust an Existing Layout

Subclass `NSCollectionViewFlowLayout` to adjust item positioning

Override

- `func layoutAttributesForElementsInRect(rect: NSRect) -> [NSCollectionViewLayoutAttributes]`
- `func layoutAttributesForItemAtIndexPath(indexPath: NSIndexPath) -> NSCollectionViewLayoutAttributes?`
- `func invalidateLayoutWithContext(context: NSCollectionViewLayoutInvalidationContext)`

Examine and adjust super-proposed item frames

Completely Custom Layouts

Completely Custom Layouts

Can subclass `NSCollectionViewLayout` directly

Completely Custom Layouts

Can subclass `NSCollectionViewLayout` directly

Implement same methods as for customizing Flow, plus

Completely Custom Layouts

Can subclass `NSCollectionViewLayout` directly

Implement same methods as for customizing Flow, plus

- `func collectionViewContentSize() -> NSSize`

Completely Custom Layouts

Can subclass `NSCollectionViewLayout` directly

Implement same methods as for customizing Flow, plus

- `func collectionViewContentSize() -> NSSize`
- `func shouldInvalidateLayoutForBoundsChange(newBounds: NSRect) -> Bool`

Hit-Testing for a Drop Target

NEW

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

```
attributes.representedElementCategory = .Item
```

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

```
attributes.representedElementCategory = .Item

attributes.indexPath = /* NSIndexPath of the item we’re dropping onto */
```

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

```
attributes.representedElementCategory = .Item
attributes.indexPath = /* NSIndexPath of the item we're dropping onto */
attributes.frame = /* NSRect item frame */
```

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

```
attributes.representedElementCategory = .Item
attributes.indexPath = /* NSIndexPath of the item we're dropping onto */
attributes.frame = /* NSRect item frame */
```

Target is a gap between items

```

```

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

```
attributes.representedElementCategory = .Item
attributes.indexPath = /* NSIndexPath of the item we're dropping onto */
attributes.frame = /* NSRect item frame */
```

Target is a gap between items

```
attributes.representedElementCategory = .InterItemGap
```

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

```
attributes.representedElementCategory = .Item
attributes.indexPath = /* NSIndexPath of the item we're dropping onto */
attributes.frame = /* NSRect item frame */
```

Target is a gap between items

```
attributes.representedElementCategory = .InterItemGap
attributes.indexPath = /* NSIndexPath of the item we're dropping before */
```

Hit-Testing for a Drop Target

NEW

```
func layoutAttributesForDropTargetAtPoint(pointInCollectionView: NSPoint)
                                                -> NSCollectionViewLayoutAttributes?
```

Target is an item

```
attributes.representedElementCategory = .Item
attributes.indexPath = /* NSIndexPath of the item we're dropping onto */
attributes.frame = /* NSRect item frame */
```

Target is a gap between items

```
attributes.representedElementCategory = .InterItemGap
attributes.indexPath = /* NSIndexPath of the item we're dropping before */
attributes.frame = /* NSRect gap frame */
```

Finding Inter-Item Gaps by NSIndexPath

NEW

Finding Inter-Item Gaps by NSIndexPath

NEW

```
func layoutAttributesForInterItemGapBeforeIndexPath(indexPath: NSIndexPath)
                                                    -> NSCollectionViewLayoutAttributes?
```

Finding Inter-Item Gaps by NSIndexPath

NEW

```
func layoutAttributesForInterItemGapBeforeIndexPath(indexPath: NSIndexPath)
                                                    -> NSCollectionViewLayoutAttributes?
```

Result

Finding Inter-Item Gaps by NSIndexPath

NEW

```
func layoutAttributesForInterItemGapBeforeIndexPath(indexPath: NSIndexPath)
                                                    -> NSCollectionViewLayoutAttributes?
```

Result

```
attributes.representedElementCategory = .InterItemGap
```

Finding Inter-Item Gaps by NSIndexPath

NEW

```
func layoutAttributesForInterItemGapBeforeIndexPath(indexPath: NSIndexPath)
                                                    -> NSCollectionViewLayoutAttributes?
```

Result

```
attributes.representedElementCategory = .InterItemGap

attributes.representedElementKind = .InterItemGapIndicator
```

Finding Inter-Item Gaps by NSIndexPath

NEW

```
func layoutAttributesForInterItemGapBeforeIndexPath(indexPath: NSIndexPath)
                                                    -> NSCollectionViewLayoutAttributes?
```

Result

```
attributes.representedElementCategory = .InterItemGap
attributes.representedElementKind = .InterItemGapIndicator
attributes.indexPath = /* NSIndexPath of the item gap comes before */
```

Finding Inter-Item Gaps by NSIndexPath

NEW

```
func layoutAttributesForInterItemGapBeforeIndexPath(indexPath: NSIndexPath)
                                                    -> NSCollectionViewLayoutAttributes?
```

Result

```
attributes.representedElementCategory = .InterItemGap
attributes.representedElementKind = .InterItemGapIndicator
attributes.indexPath = /* NSIndexPath of the item gap comes before */
attributes.frame = /* NSRect gap frame */
```

Demo

Custom item layouts

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Roadmap

Make items appear

Group items into sections

Update when model changes

Handle selection and highlighting

Handle Drag-and-Drop

Customize Layout

Conclusion

New NSCollectionView is Ready!

…to handle your toughest projects!

Coming Up

Need help or guidance? Come to our lab!

Cocoa and NSCollectionView Lab	Frameworks Lab B	Friday 9:00AM

More Information

Documentation
http://developer.apple.com/library/mac

Technical Support
App Developer Forums
http://developer.apple.com/forums

General Inquiries
Paul Marcos, App Frameworks Evangelist
pmarcos@apple.com

Related Sessions

What's New in Cocoa	Presidio	Tuesday 1:30PM
Mysteries of Auto Layout, Part 1	Presidio	Thursday 11:00AM
Mysteries of Auto Layout, Part 2	Presidio	Thursday 1:30PM

WWDC15