Advanced Dark Mode

Session 218

Matt Jacobson, Cocoa Frameworks
Jeff Nadeau, Cocoa Frameworks
Adopting Dark Mode

Build with macOS 10.14 SDK

Dynamic colors

Template images

Materials

Asset Catalog assets in Xcode 10
Adopting Dark Mode

Build with macOS 10.14 SDK

Dynamic colors

Template images

Materials

Asset Catalog assets in Xcode 10
Adopting Dark Mode

Most UI will look great using just those techniques

Some cases may require more work

We'll explore those in this session
Appearance
Materials
Vibrancy
Background styles
Backward deployment
Tips and tricks
Appearance
Materials
Vibrancy
Background styles
Backward deployment
Tips and tricks
NSAppearance

Cocoa’s UI theme system

One view hierarchy for both light and dark

Underlies Dark Mode, High Contrast mode, and Touch Bar
NSAppearance

Light . aqua

Popup
NSAppearance

Light .aqua

Dark .darkAqua

NEW
NSAppearance

Light .aqua

- labelColor
- windowBackground
- NSImage(named:)
- NSPopupButton

Popup

Dark .darkAqua

- labelColor
- windowBackground
- NSImage(named:)
- NSPopupButton

NEW
Chloe was very good today. She watched me practice my presentation and only fell asleep once! Either I'm getting better or she's being more patient.

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water 🌿.

May 25, 2018
Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorful 🌈!

May 18, 2018
Chloe's second birthday 🎇🎉
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 😊.

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is a such thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.
Dark Aqua

Chloe
2 years old

Photos
Vitals

Notes

June 4, 2018
Chloe was very good today. She watched me practice my presentation and only fell asleep for a bit! Either I'm getting better or she's being more patient.

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water.

May 25, 2018
Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorful!

May 18, 2018
Chloe’s second birthday 🎅🎁
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 🙄

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is such a thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.
June 4, 2018
Chloe was very good today. She watched me practice my presentation and only fell asleep once! Either I'm getting better or she's being more patient.

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water.

May 25, 2018
Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorful!

May 18, 2018
Chloe's second birthday🎉🎊
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 😛

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is a such thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.
NSAppearanceCustomization

Adopted by NSView and NSWindow

In 10.14, NSApplication conforms too

// Specify an appearance, or nil (by default) to inherit from parent
var appearance: NSAppearance?

// Appearance used for drawing, possibly inherited
var effectiveAppearance: NSAppearance { get }
NSAppearanceCustomization

Adopted by NSView and NSWindow

In 10.14, NSApplication conforms too

```swift
// Specify an appearance, or nil (by default) to inherit from parent
var appearance: NSAppearance?

// Appearance used for drawing, possibly inherited
var effectiveAppearance: NSAppearance { get }
```
NSAppearanceCustomization

Adopted by NSView and NSWindow

In 10.14, NSApplication conforms too

```swift
// Specify an appearance, or nil (by default) to inherit from parent
var appearance: NSAppearance?

// Appearance used for drawing, possibly inherited
var effectiveAppearance: NSAppearance { get }
```
NSAppearance

Light .aqua

Dark .darkAqua

Popup

NSAppearance(named: .aqua)

NSAppearance(named: .darkAqua)
Chloe was very good today. She watched me practice my presentation and only fell asleep once after feeling tired. Either I'm getting better or she's being more patient.

June 1, 2018
Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water.

May 25, 2018
Took Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorful!

May 18, 2018
Chloe's second birthday! 🎉🎂
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 😄

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is such a thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.
Chloe
2 years old

Photos

Vitals

Notes

June 4, 2018
Chloe was very good today. She watched me practice my presentation and only fell asleep twice! Either I'm getting better or she's being more patient.

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water.

May 25, 2018
Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorful!

May 18, 2018
Chloe's second birthday 🎂🌱
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 😜

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is a such thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.

textView.appearance = NSAppearance(named: .aqua)
June 4, 2018

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water.

May 25, 2018
T ook Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorfull!

May 18, 2018
Chloe’s second birthday 🎉🎈
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 😊

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is a such thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.
June 4, 2018

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something, and to just make sure she gets plenty of water.

May 25, 2018
Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along well with the neighborhood. It was very colorful!!

May 18, 2018
Chloe's second birthday 🎂🎁🎉
We got her a dozen crickets.
June 4, 2018

Chloe's little chameleon, Chloe, is growing a little green. I think she must have a bug or something. She always wakes up with a big yawn, and she gets plenty of water.

Chloe is very happy. Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along well with the chameleons, and Chester was a little shy. It was very colorful!!

May 18, 2018

Chloe's second birthday 🍰🎁🎉
We got her a dozen crickets.
Appearance Source

Makes a window inherit appearance from another object

```swift
extension NSWindow {
    var appearanceSource: NSAppearanceCustomization!
}
```

Assign `nil` to reset to default inheritance.
June 4, 2018

Chloe

Chloe has been looking a little green. I think she must have a bug or something. Got to make sure she gets plenty of water.

May 18, 2018

Chloe's second birthday 🎂🎁🎉

We got her a dozen crickets.
June 4, 2018

Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along with everything except the chameleon that was a little green. I think she must have a bug or something he gets plenty of water.

Chloe's second birthday 🎂🎁
We got her a dozen crickets.
June 4, 2018
Chloe

Chloe's tail is going a little green. I think she must have a bug or something. I gave her some water. Make sure she gets plenty of water.

Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along well with the other chameleons in the neighborhood. It was very colorful!!

May 18, 2018
Chloe's second birthday 🎂🎁🎉
We got her a dozen crickets.
**June 4, 2018**

Chloe

Chloe is having a little party today. She gets a little excited, she thrives on attention. She loves playing hide and seek and playing with her toys. She got a new toy for her birthday this year. We bought her a new ball to play with in the backyard.

**May 18, 2018**

Chloe's second birthday 🎂🎁

We got her a dozen crickets.
Outline View Effective Appearance:
Custom Views
Chloe
2 years old
Chloe
2 years old

Light

Chloe
2 years old

Dark
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
}

}
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}

class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}

// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}

class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}
class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}

// Custom view using an asset catalog color
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}
class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}
Custom Views

Do appearance-sensitive work in

```swift
func updateConstraints()
func layout()
func draw(_ dirtyRect: NSRect)
func updateLayer()
```

Appearance changes will cause update

Set `needsUpdateConstraints`, `needsLayout`, or `needsDisplay` to trigger manually
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        layer.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}

// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}

// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}

// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {

    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            layer?.backgroundColor = headerColor?.cgColor
        }
    }
}
/ Custom view using an asset catalog color

```swift
class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
        }
    }
}
```
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }

    var headerColor = NSColor(named: "ChameleonGreen") {
        didSet {
            needsDisplay = true
        }
    }
}
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }
}

var headerColor = NSColor(named: "ChameleonGreen") {
    didSet {
        needsDisplay = true
    }
}
// Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }
}

var headerColor = NSColor(named: "ChameleonGreen") {
    didSet {
        needsDisplay = true
    }
}
/ Custom view using an asset catalog color

class ChameleonHeaderView : NSView {
    override func updateLayer() {
        layer?.backgroundColor = headerColor?.cgColor
    }
}

var headerColor = NSColor(named: "ChameleonGreen") {
    didSet {
        needsDisplay = true
    }
}
class GlowView : NSView {
 ... 
}

class ChameleonAvatarView : NSView {
}

class GlowView : NSView {
    ...
}

class ChameleonAvatarView : NSView {
    var glowView: GlowView
    var imageView: NSImageView

    override func layout() {
        switch effectiveAppearance.bestMatch(from: [.aqua, .darkAqua]) {
            ...
        }
    }
}
class GlowView : NSView {
  ...
}

class ChameleonAvatarView : NSView {
  var glowView: GlowView
  var imageView: NSImageView

  override func layout() {
    switch effectiveAppearance.bestMatch(from: [.aqua, .darkAqua]) {
    }
  }
}

class GlowView : NSView {
    ...
}

class ChameleonAvatarView : NSView {
    var glowView: GlowView
    var imageView: NSImageView

    override func layout() {
        switch effectiveAppearance.bestMatch(from: [.aqua, .darkAqua]) {
        }
    }
}
class GlowView : NSView {
    ...
}

class ChameleonAvatarView : NSView {
    var glowView: GlowView
    var imageView: NSImageView

    override func layout() {
        switch effectiveAppearance.bestMatch(from: [.aqua, .darkAqua]) {
            case .aqua?:
                subviews = [imageView]
        }
    }
}

NEW
class GlowView : NSView {
    ...
}

class ChameleonAvatarView : NSView {
    var glowView: GlowView
    var imageView: NSImageView

    override func layout() {
        switch effectiveAppearance.bestMatch(from: [.aqua, .darkAqua]) {
        case .aqua?:
            subviews = [ imageView ]
        case .darkAqua?:
            subviews = [ glowView, imageView ]
        }
    }
}

class GlowView : NSView {
    ...
}

class ChameleonAvatarView : NSView {
    var glowView: GlowView
    var imageView: NSImageView

    override func layout() {
        switch effectiveAppearance.bestMatch(from: [.aqua, .darkAqua]) {
            case .aqua?:
                subviews = [ imageView ]
            case .darkAqua?:
                subviews = [ glowView, imageView ]
            default:
                subviews = [ imageView ]
        }
    }
}
Chloe
2 years old
Chloe
2 years old
Customizing for High Contrast
Customizing for High Contrast

Aqua

Dark Aqua
Customizing for High Contrast
Customizing for High Contrast

- Aqua
- Dark Aqua

High-Contrast Aqua
High-Contrast Dark Aqua
Customizing for High Contrast

High-contrast appearance names:

extension NSAppearance.Name {
    static let accessibilityHighContrastAqua: NSAppearance.Name
    static let accessibilityHighContrastDarkAqua: NSAppearance.Name
}

Customizing for High Contrast

Not available directly

```swift
NSAppearance(named: .accessibilityHighContrastAqua)
NSAppearance(named: .accessibilityHighContrastDarkAqua)
```

Only via System Preferences
Customizing for High Contrast

Pass to `bestMatch(from:)` to provide custom behavior

```swift
switch effectiveAppearance.bestMatch(from: [.aqua, .accessibilityHighContrastAqua, ... ]) {
```
Managing Custom Sublayers
Managing Custom Sublayers

Custom sublayers don’t inherit appearance automatically

If possible, switch to using views

Otherwise, manage layers manually with

• `viewDidChangeEffectiveAppearance()`
• `NSAppearance.current`
Detecting Appearance Change

```swift
extension NSView {
    func viewDidChangeEffectiveAppearance()
}
```

Good time to

• Perform custom invalidation
• Drop caches

No need to invalidate view itself!
Current Appearance

```swift
extension NSAppearance {
    class var current: NSAppearance!
}
```

Thread-local state

- Like `NSGraphicsContext.current`

Appearance used to resolve colors and images
Current Appearance

Current appearance is set up automatically for

```swift
func updateConstraints()  
func layout() 
func draw(_ dirtyRect: NSRect) 
func updateLayer()
```

You can also set it yourself
// Updating a Custom Sublayer

class CustomViewWithSublayers : NSView, CALayerDelegate {
    override func viewDidChangeEffectiveAppearance() {
        sublayer.setNeedsDisplay()
    }

    func display(_ sublayer: CALayer) {
        let oldAppearance = NSAppearance.current
        NSAppearance.current = effectiveAppearance

        // Update layer using our new effective appearance
        sublayer.borderColor = NSColor.systemRed.cgColor

        NSAppearance.current = oldAppearance
    }
}

// Updating a Custom Sublayer

class CustomViewWithSublayers: NSView, CALayerDelegate {
    override func viewDidChangeEffectiveAppearance() {
        sublayer.setNeedsDisplay()
    }

    func display(_ sublayer: CALayer) {
        let oldAppearance = NSAppearance.current
        NSAppearance.current = effectiveAppearance

        // Update layer using our new effective appearance
        sublayer.borderColor = NSColor.systemRed.cgColor

        NSAppearance.current = oldAppearance
    }
}

// Updating a Custom Sublayer

class CustomViewWithSublayers: NSView, CALayerDelegate {
    override func viewDidChangeEffectiveAppearance() {
        sublayer.setNeedsDisplay()
    }

    func display(_ sublayer: CALayer) {
        let oldAppearance = NSAppearance.current
        NSAppearance.current = effectiveAppearance

        // Update layer using our new effective appearance
        sublayer.borderColor = NSColor.systemRed.cgColor

        NSAppearance.current = oldAppearance
    }
}
// Updating a Custom Sublayer

class CustomViewWithSublayers : NSView, CALayerDelegate {
    override func viewDidChangeEffectiveAppearance() {
        sublayer.setNeedsDisplay()
    }

    func display(_ sublayer: CALayer) {
        let oldAppearance = NSAppearance.current
        NSAppearance.current = effectiveAppearance

        // Update layer using our new effective appearance
        sublayer.borderColor = NSColor.systemRed.cgColor

        NSAppearance.current = oldAppearance
    }
}

// Updating a Custom Sublayer

class CustomViewWithSublayers : NSView, CALayerDelegate {
    override func viewDidChangeEffectiveAppearance() {
        sublayer.setNeedsDisplay()
    }
}

func display(_ sublayer: CALayer) {
    let oldAppearance = NSAppearance.current
    NSAppearance.current = effectiveAppearance

    // Update layer using our new effective appearance
    sublayer.borderColor = NSColor.systemRed.cgColor

    NSAppearance.current = oldAppearance
}
}
// Updating a Custom Sublayer

class CustomViewWithSublayers: NSView, CALayerDelegate {
    override func viewDidChangeEffectiveAppearance() {
        sublayer.setNeedsDisplay()
    }

    func display(_ sublayer: CALayer) {
        let oldAppearance = NSAppearance.current
        NSAppearance.current = effectiveAppearance

        // Update layer using our new effective appearance
        sublayer.borderColor = NSColor.systemRed.cgColor

        NSAppearance.current = oldAppearance
    }
}
// Updating a Custom Sublayer

class CustomViewWithSublayers: NSView, CALayerDelegate {
    override func viewDidChangeEffectiveAppearance() {
        sublayer.setNeedsDisplay()
    }

    func display(_ sublayer: CALayer) {
        let oldAppearance = NSAppearance.current
        NSAppearance.current = effectiveAppearance

        // Update layer using our new effective appearance
        sublayer.borderColor = NSColor.systemRed.cgColor

        NSAppearance.current = oldAppearance
    }
}
NSImage as Layer Contents

```swift
layer.contents = someNSImage
```

```swift
let scale = someNSImage.recommendedLayerContentsScale(window.backingScaleFactor)
layer.contents = someNSImage.layerContents(forContentsScale: scale)
```

These won’t inherit appearance!
NSImage as Layer Contents

If possible, switch to NSImageView

Automatically updates for appearance
Create a CGImage in `updateLayer()`

```swift
override func updateLayer() {
    var rect = layer.bounds
    layer.contents = image.cgImage(forProposedRect: &rect, context: nil, hints: [
        .CTM : AffineTransform(scale: window?.backingScaleFactor) 
    ])}
```
Appearance
Materials
Vibrancy
Background styles
Backward deployment
Tips and tricks
Materials

Dynamic backgrounds

Use effects like blur, tinting, and gradients

Provide depth, context, and beauty to the interface
Materials

Automatically used in
• Window titlebars and backgrounds
• Table views
• Sidebar split view items
• Popovers
• Menus
NSVisualEffectView

A view that shows a material

```swift
var state: NSVisualEffectView.State
var blendingMode: NSVisualEffectView.BlendingMode
var material: NSVisualEffectView.Material
```
NSVisualEffectView

var state: NSVisualEffectView.State

Controls whether material uses active window look

By default, matches enclosing window
NSVisualEffectView

```swift
var blendingMode: NSVisualEffectView.BlendingMode
```

Controls whether material “punches through” window
NSVisualEffectView

var blendingMode: NSVisualEffectView.BlendingMode

Controls whether material “punches through” window

Behind-window by default
NSVisualEffectView

var material: NSVisualEffectView.Material

Encapsulates effect definition
Semantic Materials

Popover
Menu
Sidebar
Selection
Titlebar
<table>
<thead>
<tr>
<th>Semantic Materials</th>
<th>Sheet</th>
<th>Tool tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popover</td>
<td>Sheet</td>
<td>Tool tip</td>
</tr>
<tr>
<td>Menu</td>
<td>HUD window</td>
<td>Content background</td>
</tr>
<tr>
<td>Sidebar</td>
<td>Header view</td>
<td>Window background</td>
</tr>
<tr>
<td>Selection</td>
<td>Under window background</td>
<td>Under page background</td>
</tr>
<tr>
<td>Titlebar</td>
<td>Full-screen UI</td>
<td></td>
</tr>
</tbody>
</table>
Materials

Nonsemantic materials are deprecated:

• Light
• Dark
• Medium light
• Ultra dark
Hi John!

We just got back from our trip to Vietnam and I wanted to share some of our photos with you. We backpacked through the country for two weeks and had an experience we’ll never forget. The people were all incredibly friendly and we made a lot of new friends as we traveled around the country. You should join us next time!

Let’s schedule a lunch date soon so that we can catch up. I’ll show you the rest of the trip photos then!
Hi John!

We just got back from our trip to Vietnam and I wanted to share some of our photos with you. We backpacked through the country for two weeks and had an experience we'll never forget. The people were all incredibly friendly and we made a lot of new friends as we traveled around the country. You should join us next time!

Let's schedule a lunch date soon so that we can catch up. I'll show you the rest of the trip photos then!
June 4, 2018
Chloe was very good today. She watched me practice my presentation and only fell asleep twice! Either I’m getting better or she’s being more patient.

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water.

May 25, 2018
Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorful!

May 18, 2018
Chloe’s second birthday 😊🎉
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 😊

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is a such thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.
June 4, 2018
Chloe was very good today. She watched me practice my presentation and only fell asleep twice! Either I’m getting better or she’s being more patient.

June 1, 2018
Today Chloe was feeling a little green. I think she must have a bug or something. The doctor said it was nothing to worry about, and to just make sure she gets plenty of water.

May 25, 2018
Tuesday! Took Chloe and Chester to the chameleon park today. Chloe got along great with all the chameleons in the neighborhood. It was very colorful!

May 18, 2018
Chloe’s second birthday 😃🎉
We got her a dozen crickets.

May 14, 2018
Looked around for Chloe for three hours. Turns out she was just blending into my shirt the whole time 😊

May 9, 2018
Chloe got a bath today.

May 8, 2018
It turns out that there is a such thing as Chameleon flu. Had to give Chloe 5 spritzes from the water bottle to cool her down.
Desktop-Tinted Materials
Automatic support

NSWindow
NSScrollView
NSTableView
NSCollectionView
## Desktop-Tinted Materials

Custom NSBox support

<table>
<thead>
<tr>
<th></th>
<th>NSBox fill color</th>
<th>NSVisualEffectView material</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Window Background</strong></td>
<td><code>windowBackgroundColor</code></td>
<td><code>windowBackground</code></td>
</tr>
<tr>
<td><strong>Content Background</strong></td>
<td><code>.controlBackgroundColor</code></td>
<td><code>.contentBackground</code></td>
</tr>
<tr>
<td></td>
<td><code>textBackgroundColor</code></td>
<td></td>
</tr>
<tr>
<td><strong>Under Page Background</strong></td>
<td><code>.underPageBackgroundColor</code></td>
<td><code>.underPageBackground</code></td>
</tr>
</tbody>
</table>
Desktop-Tinted Materials
NSBox and NSVisualEffectView support

// Using NSBox
box.boxType = .custom
box.fillColor = .underPageBackgroundColor

// Using NSVisualEffectView
visualEffectView.material = .underPageBackground
Desktop-Tinted Materials

Untinted color in light

Desktop-tinted in dark

Tint effect might be disabled
<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appearance</td>
<td></td>
</tr>
<tr>
<td>Accent color</td>
<td></td>
</tr>
<tr>
<td>Highlight color</td>
<td>Blue</td>
</tr>
<tr>
<td>Sidebar icon size</td>
<td>Medium</td>
</tr>
<tr>
<td>Automatically hide and show the menu bar</td>
<td></td>
</tr>
<tr>
<td>Show scroll bars</td>
<td></td>
</tr>
<tr>
<td>When scrolling</td>
<td></td>
</tr>
<tr>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>Click in the scroll bar to</td>
<td></td>
</tr>
<tr>
<td>Jump to the next page</td>
<td></td>
</tr>
<tr>
<td>Jump to the spot that's clicked</td>
<td></td>
</tr>
<tr>
<td>Default web browser</td>
<td>Safari</td>
</tr>
<tr>
<td>Ask to keep changes when closing documents</td>
<td></td>
</tr>
<tr>
<td>Close windows when quitting an app</td>
<td></td>
</tr>
<tr>
<td>When selected, open documents and windows will not be restored when you re-open an app.</td>
<td></td>
</tr>
<tr>
<td>Recent Items</td>
<td>10 Documents, Apps, and Servers</td>
</tr>
<tr>
<td>Allow Handoff between this Mac and your iCloud devices</td>
<td></td>
</tr>
<tr>
<td>Use LCD font smoothing when available</td>
<td></td>
</tr>
</tbody>
</table>
General

Appearance:

Access:

Highlight colors:

Sidebar icon size:

Automatically hide and show the menu bar

Show scroll bars:

Automatically based on mouse or trackpad
When scrolling
Always

Click in the scroll bar to:

Jump to the next page
Jump to the spot that’s clicked

Default web browser:

Safari

Ask to keep changes when closing documents
Close windows when quitting an app
When selected, open documents and windows will not be restored when you re-open an app.

Recent items:

10 Documents, Apps, and Servers

Allow Handoff between this Mac and your iCloud devices

Use LCD font smoothing when available
visualEffectView.material = .contentBackground
Hello, world!

visualEffectView.material = .contentBackground
// Using `controlBackgroundColor` to fill the bubble

class BubbleView : NSView {

    override func draw(_ dirtyRect: NSRect) {
        let bubblePath: NSBezierPath = makeBubblePath()

        NSColor.controlBackgroundColor.setFill()
        bubblePath.fill()
    }

}
// Using controlBackgroundColor to fill the bubble

class BubbleView : NSView {

    override func draw(_ dirtyRect: NSRect) {
        let bubblePath: NSBezierPath = makeBubblePath()

        NSColor.controlBackgroundColor.setFill()
        bubblePath.fill()
    }
}
class BubbleView : NSView {

    override func draw(_ dirtyRect: NSRect) {
        let bubblePath: NSBezierPath = makeBubblePath()
        NSColor.controlBackgroundColor.setFill()
        bubblePath.fill()
    }
}
// Using controlBackgroundColor to fill the bubble

class BubbleView : NSView {

    override func draw(_ dirtyRect: NSRect) {
        let bubblePath: NSBezierPath = makeBubblePath()

        NSColor.controlBackgroundColor.setFill()
        bubblePath.setFill()

    }
}
Hey Chloe. How are you feeling today?

A little blue. 😞

Hey — I just installed Mojave, and Dark Mode looks so cool! 😢😭

What?! I am green with envy!

Never change, Chloe.

I can't help it!

Should have seen that one coming.
Hey Chloe. How are you feeling today?

A little blue. 😞

Hey — I just installed Mojave, and Dark Mode looks so cool! 😍😍

What?! I am green with envy!

Never change, Chloe.

Should have seen that one coming.
Desktop-Tinted Materials

Effect is provided by Quartz window server

Updates asynchronously from your app

But can’t draw with the color (or get RGB values)

Instead, use `maskImage`
Mask Image

extension NSVisualEffectView {
    var maskImage: NSImage?
}

Masks the material to a shape

Drawing-handler images supported
// Using a masked NSVisualEffectView for the bubble

class BubbleView: NSView {
    override func layout() {
        let bubblePath: NSBezierPath = makeBubblePath()
    }
}
class BubbleView : NSView {

    override func layout() {
        let bubblePath: NSBezierPath = makeBubblePath()

        let materialView = NSVisualEffectView(frame: bounds)
        materialView.material = .contentBackground

        let mask = NSImage(size: bounds.size, flipped: false) { (_) in
            NSColor.white.setFill() // Any opaque color will do
            bubblePath.fill()
            return true
        }

        materialView.maskImage = mask
    }
// Using a masked NSVisualEffectView for the bubble

class BubbleView: NSView {
    override func layout() {
        let bubblePath: NSBezierPath = makeBubblePath()

        let materialView = NSVisualEffectView(frame: bounds)
        materialView.material = .contentBackground

        let mask = NSImage(size: bounds.size, flipped: false) { (_) in
            NSColor.white.setFill() // Any opaque color will do
            bubblePath.fill()
            return true
        }

        materialView.maskImage = mask
    }
}
// Using a masked NSVisualEffectView for the bubble

class BubbleView: NSView {
    override func layout() {
        let bubblePath: NSBezierPath = makeBubblePath()

        let materialView = NSVisualEffectView(frame: bounds)
        materialView.material = .contentBackground

        let mask = NSImage(size: bounds.size, flipped: false) { (_) in
            NSColor.white.setFill() // Any opaque color will do
            bubblePath.fill()
            return true
        }
        materialView.maskImage = mask
    }
}
class BubbleView : NSView {
    override func layout() {
        let bubblePath: NSBezierPath = makeBubblePath()

        let materialView = NSVisualEffectView(frame: bounds)
        materialView.material = .contentBackground

        let mask = NSImage(size: bounds.size, flipped: false) {
            NSColor.white.setFill() // Any opaque color will do
            bubblePath.fill()
            return true
        }
        materialView.maskImage = mask
    }
}
// Using a masked NSVisualEffectView for the bubble

class BubbleView : NSView {
    override func layout() {
        let bubblePath: NSBezierPath = makeBubblePath()

        let materialView = NSVisualEffectView(frame: bounds)
        materialView.material = .contentBackground

        let mask = NSImage(size: bounds.size, flipped: false) { (_) in
            NSColor.white.setFill() // Any opaque color will do
            bubblePath.fill()
            return true
        }
        materialView.maskImage = mask
    }
}
Hey Chloe. How are you feeling today?

A little blue. 😞

Hey — I just installed Mojave, and Dark Mode looks so cool! 😱��

What?! I am green with envy!

Never change, Chloe.

I can't help it!

Should have seen that one coming.
Hey Chloe. How are you feeling today?

A little blue. 😞

Hey --- I just installed Mojave, and Dark Mode looks so cool! 😍

What?! I am green with envy!

Never change, Chloe.

Should have seen that one coming.
Hey Chloe. How are you feeling today?

A little blue. 😞

Hey — I just installed Mojave, and Dark Mode looks so cool! 😍😍

What?! I am green with envy!

Never change, Chloe.

I can't help it!

Should have seen that one coming.
I can’t help it!
Mask Image

Works with any material

Alpha channel used for mask

Only material (not subviews) masked

Provide a resizable image `capInsets` and `resizingMode`
Appearance
Materials
Vibrancy
Background styles
Backward deployment
Tips and tricks
Vibrancy

Lightens or darkens the content behind it

Similar to dodge and burn in a photo editor
Normal Blending
Source Over

Vibrant Blending
Plus Lighter
Normal Blending
Source Over

0%  20%  40%
60%  80%  100%
Vibrant Blending
Plus Lighter

0%  20%  40%
60%  80%  100%
Vibrancy

Provided by “vibrant” NSAppearance objects
Vibrancy

Provided by “vibrant” NSAppearance objects
Vibrancy

Provided by “vibrant” NSAppearance objects

NSAppearance(named: .vibrantDark)  NSAppearance(named: .vibrantLight)
Vibrancy

Override allows Vibrancy to return true.

Blending effect automatically applies to your view and its descendants.
Vibrancy

Typically, draw with a label color depending on prominence

Avoid non grayscale colors
class MoodMeterView : NSView {

    override var allowsVibrancy: Bool {
        return true
    }

    override func layout() {
        subviews = [ sadFaceView, ambivalentFaceView, happyFaceView, sliderView ]
    }
}

Washed out colors

Vibrant slider
Meter View

- Faces
- Slider

Allows Vibrancy
Meter View

Faces

Slider

Allows Vibrancy

Blends Vibrantly
Meter View

Faces

Slider

Allows Vibrancy

Blends Vibrantly
Meter View

Faces

Slider

Non-vibrant

Vibrant

Allows Vibrancy

Blends Vibrantly
Vibrant Blending Tips

Vibrant blend mode applies to subtree

Typically only leaf views should be vibrant

Break non vibrant drawing out into a sibling view
Vibrant Blending Tips

Avoid overlapping vibrant and non-vibrant views

Don't subclass Cocoa controls to override `allowsVibrancy`

Only override `allowsVibrancy` if you also override drawing
Appearance
Materials
Vibrancy
Background styles
Backward deployment
Tips and tricks
<table>
<thead>
<tr>
<th>Aqua</th>
<th>Normal</th>
<th>Selected / Emphasized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🌊 Katie Abeles</td>
<td>🌊 Katie Abeles</td>
</tr>
<tr>
<td></td>
<td>Pictures from Thailand</td>
<td>Pictures from Thailand</td>
</tr>
<tr>
<td></td>
<td>Here are some of the pictures that John and I took on vacation to Thailand last su...</td>
<td>Here are some of the pictures that John and I took on vacation to Thailand last su...</td>
</tr>
<tr>
<td>Background Styles</td>
<td>Normal</td>
<td>Selected / Emphasized</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Aqua</td>
<td><img src="#" alt="Katie Abeles" /> 6/4/18&lt;br&gt;Pictures from Thailand&lt;br&gt;Here are some of the pictures that John and I took on vacation to Thailand last su...</td>
<td><img src="#" alt="Katie Abeles" /> 6/4/18&lt;br&gt;Pictures from Thailand&lt;br&gt;Here are some of the pictures that John and I took on vacation to Thailand last su...</td>
</tr>
<tr>
<td>Dark Aqua</td>
<td><img src="#" alt="Katie Abeles" /> 6/4/18&lt;br&gt;Pictures from Thailand&lt;br&gt;Here are some of the pictures that John and I took on vacation to Thailand last su...</td>
<td><img src="#" alt="Katie Abeles" /> 6/4/18&lt;br&gt;Pictures from Thailand&lt;br&gt;Here are some of the pictures that John and I took on vacation to Thailand last su...</td>
</tr>
</tbody>
</table>
// Emphasis with NSBackgroundColorStyle

eenum NSView.BackgroundColorStyle {
    case light
    case dark
    // ...
}

class NSTableCellView: NSView {
    var backgroundColorStyle: NSView.BackgroundColorStyle
}
// Emphasis with NSBackgroundStyle

elect  

```swift
enum NSView.BackgroundStyle {
    case normal
    case emphasized
    // ...
}
```

```swift
class NSTableViewCell: NSView {
    var backgroundStyle: NSView.BackgroundStyle
}
```
NSTableView now sets background style recursively
Label colors automatically adapt to emphasized style

labelColor
secondaryLabelColor
tertiaryLabelColor
quaternaryLabelColor
NSTableView now sets background style recursively

Label colors automatically adapt to emphasized style
Custom Selections

Chameleon Diary

Downloading...

labelColor

secondaryLabelColor
Selection Material

Selection material follows the accent color

Avoid drawing custom blue selections

Use NSVisualEffectView instead:

```swift
visualEffectView.material = .selection
```
Appearance
Materials
Vibrancy
Background styles
Backward deployment
Tips and tricks
Backward Deployment

System colors
Backward Deployment

System colors

- alternateSelectedControlText
- controlBackgroundColor
- controlColor
- controlTextColor
- disabledControlTextColor
- gridColor
- keyboardFocusIndicatorColor
- labelColor
- linkColor
- placeholderTextColor
- quaternaryLabelColor
- secondaryLabelColor
- selectedControlColor
- selectedControlTextColor
- selectedMenuItemTextColor
- selectedTextBackgroundColor
- selectedTextColor
- systemBlueColor
- systemBrownColor
- systemGrayColor
- systemGreenColor
- systemOrangeColor
- systemPinkColor
- systemPurpleColor
- systemRedColor
- systemYellowColor
- tertiaryLabelColor
- textColor
- underPageBackgroundColor
- windowBackgroundColor
- windowFrameTextColor
- controlAccentColor
- selectedContentBackgroundColor
- separatorColor
- unemphasizedSelectedTextColor
- textBackgroundColor
- textBackgroundColor
## Backward Deployment

### System colors

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>alternateSelectedControlText</td>
<td>textColor</td>
</tr>
<tr>
<td>controlBackgroundColor</td>
<td>selectedMenuItemTextColor</td>
</tr>
<tr>
<td>controlColor</td>
<td>selectedTextBackgroundColor</td>
</tr>
<tr>
<td>controlTextColor</td>
<td>selectedTextColor</td>
</tr>
<tr>
<td>disabledControlTextColor</td>
<td>systemBlueColor</td>
</tr>
<tr>
<td>gridColor</td>
<td>systemBrownColor</td>
</tr>
<tr>
<td>keyboardFocusIndicatorColor</td>
<td>systemGrayColor</td>
</tr>
<tr>
<td>labelColor</td>
<td>systemGreenColor</td>
</tr>
<tr>
<td>linkColor</td>
<td>systemOrangeColor</td>
</tr>
<tr>
<td>placeholderTextColor</td>
<td>systemPinkColor</td>
</tr>
<tr>
<td>quaternaryLabelColor</td>
<td>systemPurpleColor</td>
</tr>
<tr>
<td>secondaryLabelColor</td>
<td>systemRedColor</td>
</tr>
<tr>
<td>selectedControlColor</td>
<td>systemYellowColor</td>
</tr>
<tr>
<td>selectedControlTextColor</td>
<td>tertiaryLabelColor</td>
</tr>
<tr>
<td>selectedControlTextColor</td>
<td>textBackgroundColor</td>
</tr>
<tr>
<td>selectedMenuItemTextColor</td>
<td>underPageBackgroundColor</td>
</tr>
<tr>
<td>selectedTextBackgroundColor</td>
<td>textBackgroundColor</td>
</tr>
<tr>
<td>selectedTextColor</td>
<td>underPageBackgroundColor</td>
</tr>
<tr>
<td>systemBlueColor</td>
<td>windowBackgroundColor</td>
</tr>
<tr>
<td>systemBrownColor</td>
<td>windowFrameTextColor</td>
</tr>
<tr>
<td>systemGrayColor</td>
<td>controlAccentColor</td>
</tr>
<tr>
<td>systemGreenColor</td>
<td>selectedContentBackgroundColor</td>
</tr>
<tr>
<td>systemOrangeColor</td>
<td>separatorColor</td>
</tr>
<tr>
<td>systemPinkColor</td>
<td>unemphasizedSelectedTextColor</td>
</tr>
<tr>
<td>systemPurpleColor</td>
<td></td>
</tr>
<tr>
<td>systemRedColor</td>
<td></td>
</tr>
<tr>
<td>systemYellowColor</td>
<td></td>
</tr>
</tbody>
</table>
class NSColor {
    @available(OSX 10.13, *)
    public init?(named name: NSColor.Name)
}

Backward Deployment
Asset catalog colors
Backward Deployment
Asset catalog colors

class NSColor {
    @available(OSX 10.13, *)
    public init?(named name: NSColor.Name)
}

extension NSColor {
    class var chameleonHeaderColor: NSColor {
        if #available(OSX 10.13, *) {
            return NSColor(named: "ChameleonHeaderColor")!
        } else {
            return NSColor(srgbRed: 0.585, green: 0.971, blue: 0.762, alpha: 1.0)
        }
    }
}
## Backward Deployment

### Desktop-tinted materials

Desktop-tinted NSVisualEffectView materials are new in 10.14.

### Classes providing automatic materials:

<table>
<thead>
<tr>
<th>10.0+</th>
<th>10.5+</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSWindow</td>
<td>NSBox (Custom)</td>
</tr>
<tr>
<td>NSScrollView</td>
<td>NSCollectionView</td>
</tr>
<tr>
<td>NSTableView</td>
<td></td>
</tr>
</tbody>
</table>
Backward Deployment
Manually enable or disable Dark Mode

Automatically opted in with 10.14 SDK

Info.plist key: NSRequiresAquaSystemAppearance

- Set to NO to enable Dark Mode with earlier SDKs
- Set to YES to disable it temporarily
Appearance
Materials
Vibrancy
Background styles
Backward deployment
Tips and tricks
Find Hardcoded Colors

Static named colors

```
label.textColor = .black
```

Hardcoded components

```
customBox.fillColor = NSColor(red: 1.0, green: 0.95, blue: 0.90, alpha: 1.0)
```
Find Hardcoded Colors

Static named colors

```swift
textColor = .labelColor
```

Hardcoded components

```swift
fillColor = NSColor(red: 1.0, green: 0.95, blue: 0.90, alpha: 1.0)
```
Find Hardcoded Colors

Static named colors

```swift
label.textColor = .labelColor
```

Hardcoded components

```swift
customBox.fillColor = NSColor(named: "ChameleonHeaderColor")
```
Prefer block-based NSImage over lockFocus

```swift
let badgedImage = NSImage(size: imageSize)
badgedImage.lockFocus()

// Draw chameleon
baseImage.draw(in: rect, from: .zero, operation: .sourceOver,
fraction: 1.0)

// Draw badge
NSColor(named: "ChameleonBadgeFill")?.set()
badgePath.fill()
badgedImage.unlockFocus()
```
Offscreen Drawing

Prefer block-based NSImage over lockFocus

```swift
NSImage(size: imageSize, flipped: false) { imageRect -> Bool in
    // Draw chameleon
    baseImage.draw(in: imageRect, from: .zero,
                   operation: .sourceOver, fraction: 1.0)

    // Draw badge
    NSColor(named: "ChameleonBadgeFill")?.set()
    badgePath.fill()

    return true
}
```
Offscreen Drawing

Prefer block-based NSImage over lockFocus

```swift
NSImage(size: imageSize, flipped: false) { imageRect -> Bool in
    // Draw chameleon
    baseImage.draw(in: imageRect, from: .zero,
                  operation: .sourceOver, fraction: 1.0)
    // Draw badge
    NSColor(named: "ChameleonBadgeFill")?.set()
    badgePath.fill()
    return true
}
```
Offscreen Drawing

Other offscreen drawing techniques, including:

NSGraphicsContext(bitmapImageRep: NSBitmapImageRep)

CGBitmapContext()

NSView.cacheDisplay(in: NSRect, to: NSBitmapImageRep)
let attributes: [NSAttributedString.Key : Any] = [
    .font : NSFont.systemFont(ofSize: 36.0)
]

The chameleon is capable of blending in to its environment.
let attributes: [NSAttributedString.Key : Any] = [
    .font : NSFont.systemFont(ofSize: 36.0),
    .foregroundColor : NSColor.labelColor
]

The chameleon is capable of blending in to its environment.
let attributes: [NSAttributedString.Key : Any] = [
    .font : NSFont.systemFont(ofSize: 36.0),
    .foregroundColor : NSColor.labelColor
]

The chameleon is capable of blending in to its environment.
let attributes: [NSAttributedString.Key : Any] = [
  .font : NSFont.systemFont(ofSize: 36.0),
  .foregroundColor : NSColor.labelColor
]

textView.performValidatedReplacement(in: range,
  with: attributedStr)

The chameleon is capable of blending in to its environment.
Appearances in Interface Builder

Views set to Aqua won’t change dynamically
Choose “Inherited” to match the superview
Appearances in Interface Builder

Views set to Aqua won’t change dynamically

Choose “Inherited” to match the superview
Appearances in Interface Builder

Views set to Aqua won’t change dynamically
Choose “Inherited” to match the superview
Appearances in Interface Builder

Views set to Aqua won’t change dynamically

Choose “Inherited” to match the superview
NSVisualEffectView Appearance

NSVisualEffectView now automatically uses the correct vibrant appearance

Avoid overriding the appearance

• Interface Builder: select "Inherited"

• In code: set `appearance` to `nil`
Designing in Interface Builder

Chloe
2 years old
Designing in Interface Builder
Designing in Interface Builder
Testing Appearances

Switch appearances from the Debug Bar and Touch Bar
Testing Appearances

Switch appearances from the Debug Bar and Touch Bar
Testing Appearances

Switch appearances from the Debug Bar and Touch Bar

NEW
Using the View Debugger
Using the View Debugger
Using the View Debugger
Using the View Debugger
Using the View Debugger

Enhancements in Xcode 10

• System color names
• Asset Catalog color names
• NSAppearance configuration
Using the View Debugger

Enhancements in Xcode 10

- System color names
- Asset Catalog color names
- NSAppearance configuration
Summary

Work with NSAppearance to adapt automatically
Add depth with new visual effect materials
Adapt drawing for vibrancy and selections
Take advantage of Xcode design and debugging tools
More Information

https://developer.apple.com/wwdc18/218

<table>
<thead>
<tr>
<th>Lab Name</th>
<th>Technology Lab</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cocoa and Dark Mode Lab</td>
<td>Technology Lab 9</td>
<td>Wednesday 2:00PM</td>
</tr>
<tr>
<td>Cocoa Lab</td>
<td>Technology Lab 7</td>
<td>Friday 11:00AM</td>
</tr>
</tbody>
</table>