What’s New in tvOS

Session 206

Hans Kim  tvOS Engineer
Welcome
Welcome
First WWDC for tvOS
Agenda
Agenda

tvOS Overview
Agenda

tvOS Overview
New in tvOS 10
Agenda

tvOS Overview
New in tvOS 10
Diving in
tvOS
Overview
Asset storage
Asset storage
Public and private database
Asset storage
Public and private database
iCloud authentication
Asset storage
Public and private database
iCloud authentication
CloudKit JS
Faster initial launch
Faster initial launch
Minimize wait time
Faster initial launch
Minimize wait time
Larger apps
.app

- Executable
- Resources
Executable

• Executable
• Resources

≤ 200 MB
.app

- Executable
- Resources

≤ 200 MB
• Executable
• Resources

≤ 200 MB

.app

Asset Pack
Asset Pack
Asset Pack
Executable
• Resources
≤ 200 MB

Initial Install Tags
• Asset Pack
• Asset Pack
• Asset Pack
• Executable
• Resources

 ≤ 200 MB

Initial Install Tags

Asset Pack  Asset Pack  Asset Pack

≤ 2 GB
• Executable
• Resources

≤ 200 MB

Initial Install Tags

Asset Pack Asset Pack Asset Pack

≤ 2 GB
Executable
· Resources

≤ 200 MB

Initial Install Tags
· Asset Pack
· Asset Pack
· Asset Pack

≤ 2 GB

Asset Pack
Asset Pack
Asset Pack

Prefetch Tags
- Executable
- Resources

≤ 200 MB

Initial Install Tags
- Asset Pack
- Asset Pack
- Asset Pack

≤ 2 GB

Prefetch Tags
- Asset Pack
- Asset Pack
- Asset Pack

≤ 4 GB
- Executable
- Resources

≤ 200 MB

Initial Install Tags
- Asset Pack
- Asset Pack
- Asset Pack

Prefetch Tags
- Asset Pack
- Asset Pack
- Asset Pack
- Asset Pack
- Asset Pack
- Asset Pack
- Asset Pack

≤ 2 GB

≤ 4 GB

... Asset Pack
- Asset Pack
App Bundle + Content
20.2 GB
App Bundle + Content
Asset Catalog

Parallax Previewer

Parallax Exporter
The Future of TV is Apps
TVML Kit

Template-based
TVMLKit

Template-based
Javascript
TVMLKit

Template-based
Javascript
Rapid development
TVMLKit

Template-based
Javascript
Rapid development
Performance-tuned
Disney presents a heartwarming comedy-adventure set in the modern mammal metropolis of Zootopia. With habitat neighborhoods like ritzy Sahara Square and frigid Tundratown, it’s a melting pot where animals from every environment live together—a place where no matter...
New in tvOS 10
AirDrop lets you instantly share with devices nearby. To send, turn on AirDrop from Control Center in iOS or from Finder on a Mac and make the devices discoverable to everyone.
UserNotification.framework
Building Apps
UIKit
UIKit

Full native development
UIKit

Full native development

Existing iOS app
UIKit

Full native development
Existing iOS app

Mastering UIKit on tvOS

Focus Interaction on tvOS

Presidio
Wednesday 10:00AM

Mission
Wednesday 4:00PM
TVMLKit
TVMLKit

Javascript and XML
TVMLKit

Javascript and XML
TVMLKit

Javascript and XML
Mix in UIKIt
<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developing tvOS Apps Using TVMLKit: Part 1</td>
<td>Mission</td>
</tr>
<tr>
<td></td>
<td>Wednesday 1:40PM</td>
</tr>
<tr>
<td>Developing tvOS Apps Using TVMLKit: Part 2</td>
<td>Mission</td>
</tr>
<tr>
<td></td>
<td>Thursday 4:00PM</td>
</tr>
</tbody>
</table>
Games
Games
Games

GameKit

<table>
<thead>
<tr>
<th>What’s New in GameplayKit</th>
<th>Pacific Heights</th>
<th>Thursday 9:00AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in SpriteKit</td>
<td>Presidio</td>
<td>Thursday 5:00PM</td>
</tr>
</tbody>
</table>
Diving In

Marshall Huss  tvOS Engineer
Diving In

Remotes and devices
Diving In

Remotes and devices

Notifications
Diving In

Remotes and devices
Notifications
Light and dark appearance
Remotes and Devices

New Apple TV Remote app
Remotes and Devices

New Apple TV Remote app
Works just like the Siri Remote
Remotes and Devices

New Apple TV Remote app
Works just like the Siri Remote
Includes game controller layout
Controlling Game Input for Apple TV

Mission

Wednesday 5:00PM
Notifications
“FireplaceWorld” Would Like to Send You Notifications

Allow this app to show an app icon badge when there are new notifications. You can change your selection at any time in Settings.

Select:

- Allow
- Don’t Allow
Remote Notifications

Use Apple Push Notification service to send remote notifications to your app
Remote Notifications

Use Apple Push Notification service to send remote notifications to your app

Supported notification types
Remote Notifications

Use Apple Push Notification service to send remote notifications to your app

Supported notification types

- Badge
Remote Notifications

Use Apple Push Notification service to send remote notifications to your app

Supported notification types

- Badge
- Content-available
Remote Notifications

Use Apple Push Notification service to send remote notifications to your app

Supported notification types

- Badge
- Content-available

Notifications will not wake up your app in the background
content-available

When are they delivered?

APNs

badge:
content-available: true

tvOS

background
content-available
When are they delivered?

APNs
badge: 1
content-available: true

tvOS
notif 1

background
content-available

When are they delivered?

- badge: 2
- content-available: true

- tvOS
- notif 2

- background
content-available

When are they delivered?

APNs

badge: 3
content-available: true

tvOS

notif 3

background

3
content-available

When are they delivered?
content-available

When are they delivered?

APNs

tvOS

notif 3

foreground
Use the new UserNotification.framework
UserNotification.framework

- Use the new UserNotification.framework
- UIKit API for notifications prohibited on tvOS
Use the new UserNotification.framework

UIKit API for notifications prohibited on tvOS

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Location</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to Notifications</td>
<td>Pacific Heights</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>Advanced Notifications</td>
<td>Pacific Heights</td>
<td>Wednesday 10:00AM</td>
</tr>
</tbody>
</table>
Appearance
Overview
Light / Dark

System appearance is changed by the user
Overview

Light / Dark

System appearance is changed by the user

New API on UI Trait Collection
Overview

Light / Dark

System appearance is changed by the user
New API on UITraitCollection
You must opt-in for dark appearance
Overview
Light / Dark

System appearance is changed by the user
New API on UITraitCollection
You must opt-in for dark appearance
• If you don’t, then your app will always use the light appearance
Adopting Appearance
Opting In

Info.plist: user interface style

Add new Info.plist key UIUserInterfaceStyle to opt in
Opting In

Info.plist: user interface style

Add new Info.plist key UIUserInterfaceStyle to opt in

- "Light" resolves in always light option
Opting In

Info.plist: user interface style

Add new Info.plist key UIUserInterfaceStyle to opt in

• “Light” resolves in always light option
• “Dark” resolves in always dark option
Opting In

Info.plist: user interface style

Add new Info.plist key UIUserInterfaceStyle to opt in

- “Light” resolves in always light option
- “Dark” resolves in always dark option
- “Automatic” resolves to the system setting
Opting In

Info.plist: user interface style

Add new Info.plist key UIUserInterfaceStyle to opt in

- “Light” resolves in always light option
- “Dark” resolves in always dark option
- “Automatic” resolves to the system setting

New projects set to “Automatic” by default
`.userInterfaceStyle`

New property on `UITraitCollection`
New property on UITraitCollection

```swift
public enum UIUserInterfaceStyle : Int {
    case unspecified
    case light
    case dark
}
```
Adopting Appearance

Developer tools support
Adopting Appearance

Developer tools support

Standard controls will automatically change their appearance
Adopting Appearance

Developer tools support
Standard controls will automatically change their appearance
Appearance proxy can customize based on UITraitCollection
Adopting Appearance

Developer tools support

Standard controls will automatically change their appearance

Appearance proxy can customize based on UITraitCollection

Handle changes in -traitCollectionDidChange: in your view and view controllers
Specify colors based on user interface style in the Inspector
Specify colors based on user interface style in the Inspector
Light and dark style preview for storyboards
Developer Tools Support

Specify colors based on user interface style in the Inspector
Light and dark style preview for storyboards
Simulator shortcut to quickly switch system appearance
UIKit Controls
Apple TV Name

You can name your Apple TV to make it easy to recognize when you use AirPlay and Home Sharing.

Apple TV

[Keyboard input area with letters, numbers, and special characters]

Done
Apple TV Name

You can name your Apple TV to make it easy to recognize when you use AirPlay and Home Sharing.

Apple TV

Your current name is 'Apple TV'.

Done
UILabel & UITextView

Customizing text

UILabel and UITextView will change their text color on appearance change.
UILabel & UITextView

Customizing text

UILabel and UITextView will change their text color on appearance change

• Black for light appearance
UILabel & UITextView

Customizing text

UILabel and UITextView will change their text color on appearance change

- Black for light appearance
- White for dark appearance
UILabel & UITextView

Customizing text

UILabel and UITextView will change their text color on appearance change

• Black for light appearance
• White for dark appearance

If you set a text color you’re responsible for light and dark colors
Blurs
Prominent and regular
Blurs
Prominent and regular

New automatic blur styles
• UIBlurEffectStyleRegular
• UIBlurEffectStyleProminent
Blurs
Prominent and regular

New automatic blur styles
• UILBlurEffectStyleRegular
• UILBlurEffectStyleProminent

<table>
<thead>
<tr>
<th>.regular</th>
<th>.light</th>
<th>.dark</th>
</tr>
</thead>
<tbody>
<tr>
<td>.prominent</td>
<td>.extraLight</td>
<td>.extraDark</td>
</tr>
</tbody>
</table>
Appearance Proxy
Appearance Proxy

UIButton.appearance().setTitleColor(.red(), for: [])
Appearance Proxy

```swift
UIButton.appearance().setTitleColor(.red(), for: [])

let light = UITraitCollection(userInterfaceStyle: .light)
let dark = UITraitCollection(userInterfaceStyle: .dark)

UIButton.forTraitCollection(light).setTitleColor(.red(), for: [])
UIButton.forTraitCollection(dark).setTitleColor(.blue(), for: []
```
Responding to Changes Directly
Responding to Changes Directly

traitCollectionDidChange
Responding to Changes Directly

traitCollectionDidChange

override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
    super.traitCollectionDidChange(previousTraitCollection)
}
Responding to Changes Directly

traitCollectionDidChange

override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
    super.traitCollectionDidChange(previousTraitCollection)
    // Is userInterfaceStyle available?
    guard traitCollection.responds(to: #selector(getter: UITraitCollection.userInterfaceStyle)))
    else { return }
}
Responding to Changes Directly

traitCollectionDidChange

override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
    super.traitCollectionDidChange(previousTraitCollection)

    // Is userInterfaceStyle available?
    guard (traitCollection.responds(to: #selector(getter: UITraitCollection.userInterfaceStyle))) {
        return
    }

    // Did the userInterfaceStyle change?
    guard (traitCollection.userInterfaceStyle != previousTraitCollection?.userInterfaceStyle) {
        return
    }
}
override func traitCollectionDidChange(_ previousTraitCollection:UITraitCollection?) {
    super.traitCollectionDidChange(previousTraitCollection)
    // Is userInterfaceStyle available?
    guard(traitCollection.responds(to: #selector(getter: UITraitCollection.userInterfaceStyle)))
        else { return }
    // Did the userInterfaceStyle change?
    guard(traitCollection.userInterfaceStyle != previousTraitCollection?.userInterfaceStyle)
        else { return }
    if traitCollection.userInterfaceStyle == .dark {
        // Handle .dark
    }
    else {
        // Handle .light
    }
}
Overriding Trait Collections
Disney presents a heartwarming comedy-adventure set in the modern mammal metropolis of Zootopia. With habitat neighborhoods like ritzy Sahara Square and frigid Tundratown, it's a melting pot where animals from every environment live together—a place where no matter...
Overriding Trait Collections
Overriding Trait Collections

```swift
public func setOverrideTraitCollection(
    _ collection: UITraitCollection?,
    forChildViewController childViewController: UIViewController
)
```
class AppearanceViewController: UIViewController {
    var style: UIUserInterfaceStyle = .light

    override init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {...}
    required init?(coder aDecoder: NSCoder) {...}

    var viewController: UIViewController {
        get { return self }
        set {
            // override trait collection
            let traitCollection = UITraitCollection(userInterfaceStyle: style)
            self.setOverrideTraitCollection(traitCollection, forChildViewController: newValue)

            // add child view controller
            self.addChildViewController(newValue)
            newValue.view.frame = view.bounds
            self.view.addSubview(newValue.view)
            newValue.didMove(toParentViewController: self)
        }
    }
}
class AppearanceViewController: UIViewController {
    var style: UIUserInterfaceStyle = .light
    override init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {
    }
    required init?(coder aDecoder: NSCoder) {
    }

    var viewController: UIViewController {
        get { return self }
        set {
            // override trait collection
            let traitCollection = UITraitCollection(userInterfaceStyle: style)
            self.setOverrideTraitCollection(traitCollection, forChildViewController: newValue)

            // add child view controller
            self.addChildViewController(newValue)
            newValue.view.frame = view.bounds
            self.view.addSubview(newValue.view)
            newValue.didMove(toParentViewController: self)
        }
    }
}
class AppearanceViewController: UIViewController {

var style: UIUserInterfaceStyle = .light

override init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {...
required init?(coder aDecoder: NSCoder) {...}

var viewController: UIViewController {
    get { return self }
    set {
        // override trait collection
        let traitCollection = UITraitCollection(userInterfaceStyle: style)
        self.setOverrideTraitCollection(traitCollection, forChildViewController: newValue)

        // add child view controller
        self.addChildViewController(newValue)
        newValue.view.frame = view.bounds
        self.view.addSubview(newValue.view)
        newValue.didMove(toParentViewController: self)
    }
}
class AppearanceViewController: UIViewController {
    var style: UIUserInterfaceStyle = .light

    override init?(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {...}
    required init?(coder aDecoder: NSCoder) {...}

    var viewController: UIViewController {
        get { return self }
        set {
            // override trait collection
            let traitCollection = UITraitCollection(userInterfaceStyle: style)
            self.setOverrideTraitCollection(traitCollection, forChildViewController: newValue)

            // add child view controller
            self.addChildViewController(newValue)
            newValue.view.frame = view.bounds
            self.view.addSubview(newValue.view)
            newValue.didMove(toParentViewController: self)
        }
    }
}
class AppearanceViewController: UIViewController {

    var style: UIUser Interface Style = .light

    override init?(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {...}

    required init?(coder aDecoder: NSCoder) {...}

    var viewController: UIViewController {
        get { return self }

        set {
            // override trait collection
            let traitCollection =UITraitCollection(userInterfaceStyle: style)
            self.setOverrideTraitCollection(traitCollection, forChildViewController: newValue)

            // add child view controller
            self.addChildViewController(newValue)
            newValue.view.frame = view.bounds
            self.view.addSubview(newValue.view)
            newValue.didMove(toParentViewController: self)
        }
    }
}

class AppearanceViewController: UIViewController {
    var style: UIUserInterfaceStyle = .light

    override init?(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {
        ...
    }

    required init?(coder aDecoder: NSCoder) {
        ...
    }

    var viewController: UIViewController {
        get { return self }
        set {

            // override trait collection
            let traitCollection = UITraitCollection(userInterfaceStyle: style)
            self.setOverrideTraitCollection(traitCollection, forChildViewController: newValue)

            // add child view controller
            self.addChildViewController(newValue)
            newValue.view.frame = view.bounds
            self.view.addSubview(newValue.view)
            newValue.didMove(toParentViewController: self)
        }
    }
}
let appearanceViewController = AppearanceViewController()
appearanceViewController.style = .dark
appearanceViewController.viewController = viewController;
navController.pushViewController(appearanceViewController, animated: true)
let appearanceViewController = AppearanceViewController()
appearanceViewController.style = .dark
appearanceViewController.viewController = viewController;
navController.pushViewController(appearanceViewController, animated: true)
let appearanceViewController = AppearanceViewController()
appearanceViewController.style = .dark
appearanceViewController.viewController = viewController;
navController.pushViewController(appearanceViewController, animated: true)
let appearanceViewController = AppearanceViewController()
appearanceViewController.style = .dark
appearanceViewController.viewController = viewController;
navController.pushViewController(appearanceViewController, animated: true)
let appearanceViewController = AppearanceViewController()
appearanceViewController.style = .dark
appearanceViewController.viewController = viewController;
navController.pushViewController(appearanceViewController, animated: true)
Demo

Adopting system appearance

Brandon Kieft  tvOS Engineer
Summary
Summary

Familiar frameworks, tools, and services
Summary

Familiar frameworks, tools, and services
Focus-based Interaction, Layered Images, Top Shelf
Summary

Familiar frameworks, tools, and services
Focus-based Interaction, Layered Images, Top Shelf
Universal Purchase, Auto Download
Summary

Familiar frameworks, tools, and services
Focus-based Interaction, Layered Images, Top Shelf
Universal Purchase, Auto Download
New Apple TV Remote app
Summary

Familiar frameworks, tools, and services
Focus-based Interaction, Layered Images, Top Shelf
Universal Purchase, Auto Download
New Apple TV Remote app
ReplayKit, and SpriteKit support in Focus Engine
Summary

Familiar frameworks, tools, and services
Focus-based Interaction, Layered Images, Top Shelf
Universal Purchase, Auto Download
New Apple TV Remote app
ReplayKit, and SpriteKit support in Focus Engine
Light and dark system appearance
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mastering UIKit on tvOS</td>
<td>Presidio</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Developing tvOS Apps Using TVMLKit: Part 1</td>
<td>Mission</td>
<td>Wednesday 1:40PM</td>
</tr>
<tr>
<td>Focus Interaction on tvOS</td>
<td>Mission</td>
<td>Wednesday 4:00PM</td>
</tr>
<tr>
<td>Developing tvOS Apps Using TVMLKit: Part 2</td>
<td>Mission</td>
<td>Thursday 4:00PM</td>
</tr>
</tbody>
</table>
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go Live with ReplayKit</td>
<td>Mission</td>
<td>Tuesday 10:00AM</td>
</tr>
<tr>
<td>Introduction to Notifications</td>
<td>Pacific Heights</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>Advanced Notifications</td>
<td>Pacific Heights</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Controlling Game Input for Apple TV</td>
<td>Mission</td>
<td>Wednesday 5:00PM</td>
</tr>
<tr>
<td>Optimizing On-Demand Resources</td>
<td>Mission</td>
<td>Thursday 10:00AM</td>
</tr>
<tr>
<td>AVKit on tvOS</td>
<td>Presidio</td>
<td>Friday 11:00AM</td>
</tr>
<tr>
<td>Topic</td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>Kickoff</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>Designing for Apple TV</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>Focus-Driven Interfaces with UIKit</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>Siri Remote and Game Controllers</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>On-Demand Resources and Data Storage</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>Media Playback</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>Topic</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>Introduction to TVMLKit</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>Best Practices for Designing tvOS Apps</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>Tuning Your tvOS App</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>The Top Shelf</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
<tr>
<td>App Store Distribution and Marketing</td>
<td>Tech Talks 2016</td>
<td></td>
</tr>
</tbody>
</table>
# tvOS Tech Talks


<table>
<thead>
<tr>
<th>Title</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to TVMLKit</td>
<td>Tech Talks 2016</td>
</tr>
<tr>
<td>Best Practices for Designing tvOS Apps</td>
<td>Tech Talks 2016</td>
</tr>
<tr>
<td>Tuning Your tvOS App</td>
<td>Tech Talks 2016</td>
</tr>
<tr>
<td>The Top Shelf</td>
<td>Tech Talks 2016</td>
</tr>
<tr>
<td>App Store Distribution and Marketing</td>
<td>Tech Talks 2016</td>
</tr>
<tr>
<td>Labs</td>
<td>Locations</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>tvOS Lab</td>
<td>Frameworks Lab A</td>
</tr>
<tr>
<td>tvOS Graphics and Games Lab</td>
<td>Graphics, Games, and Media Lab B</td>
</tr>
<tr>
<td>tvOS Lab</td>
<td>Frameworks Lab D</td>
</tr>
<tr>
<td>TVMLKit Lab</td>
<td>Graphics, Games, and Media Lab B</td>
</tr>
<tr>
<td>tvOS Lab</td>
<td>Frameworks Lab D</td>
</tr>
<tr>
<td>On-Demand Resources Lab</td>
<td>Frameworks Lab B</td>
</tr>
<tr>
<td>TVMLKit Lab</td>
<td>Graphics, Games, and Media Lab C</td>
</tr>
</tbody>
</table>
More Information
