Introducing PDFKit on iOS

PDF on macOS and iOS

Session 241

Jeremy Bridon, Software Engineer
Nicki Brower, Software Engineer

Portable Document Format

Portable Document Format

Framework Overview

Portable Document Format

Framework Overview

Document, Page, and Annotations Model

Portable Document Format

Framework Overview

Document, Page, and Annotations Model

Deep-Dive: Annotations

Portable Document Format

Framework Overview

Document, Page, and Annotations Model

Deep-Dive: Annotations

Best Practices

pages in PDF specification

1,310

pages in PDF specification

Portable Document Format

Government, medical, financial, and business documents

Strong encryption with permissions model

User interactive with widgets and annotations

Printing what you see is what you get

Portable Document Format

Government, medical, financial, and business documents

Strong encryption with permissions model

User interactive with widgets and annotations

Printing what you see is what you get

Complex binary format, big specification

CoreGraphics PDF Framework

Same drawing model as PDF graphics

Read and write features

C-language functions

CoreGraphics PDF Framework

Same drawing model as PDF graphics

Read and write features

C-language functions

No AppKit primitives

No document interaction

No accessibility support

PDFKit Framework

PDFKit is based on CoreGraphics PDF features

Modernized Swift and Objective-C API

AppKit support

Easy to open, modify, draw, and save documents

Select and search text

PDFKit Framework

NEW

PDFKit is based on CoreGraphics PDF features

Modernized Swift and Objective-C API

AppKit support **and UIKit support**

Easy to open, modify, draw, and save documents

Select and search text

Improved accessibility support

PDFKit Today

PDFKit Today

macOS

PDFKit Today

- macOS

PDFKit Today

macOS

PDFKit Today

macOS






PDFKit Today

macOS






iOS

PDFKit Today

macOS

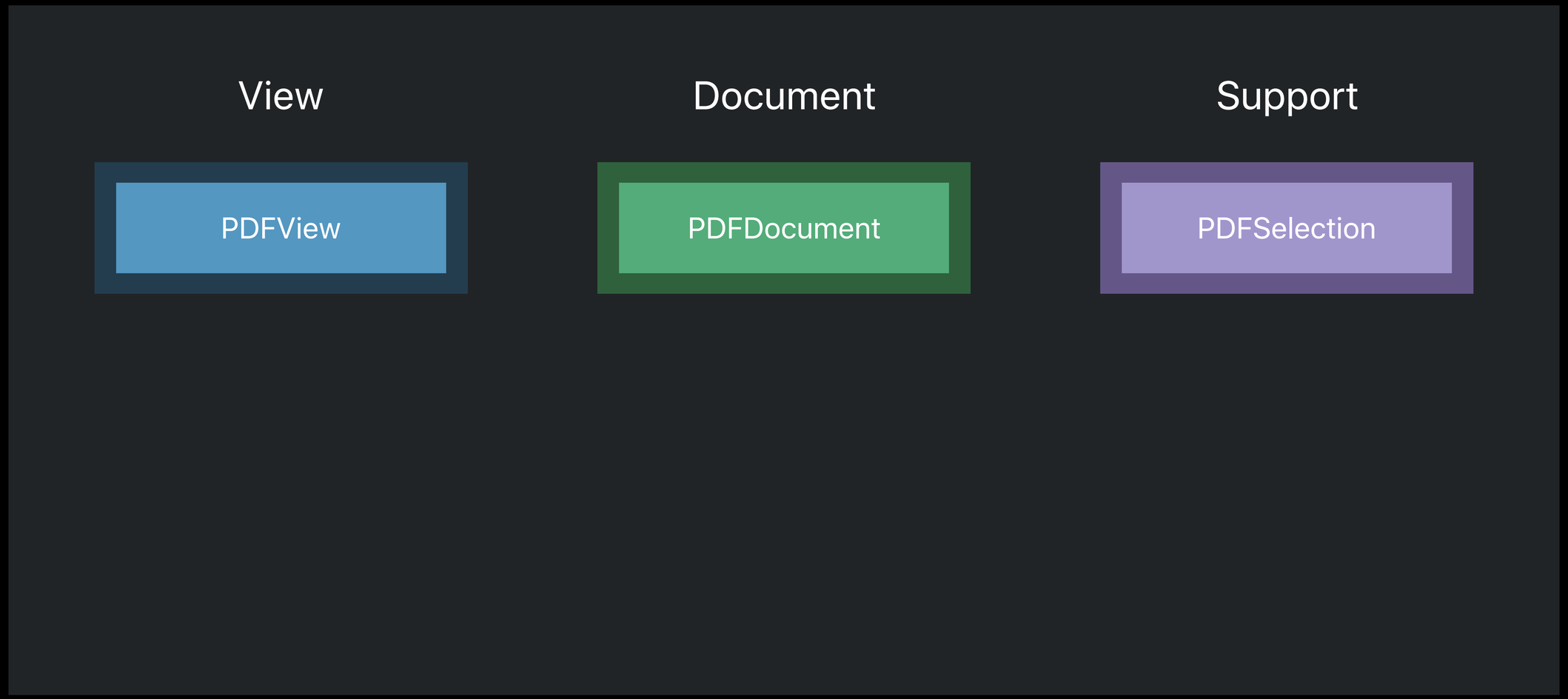




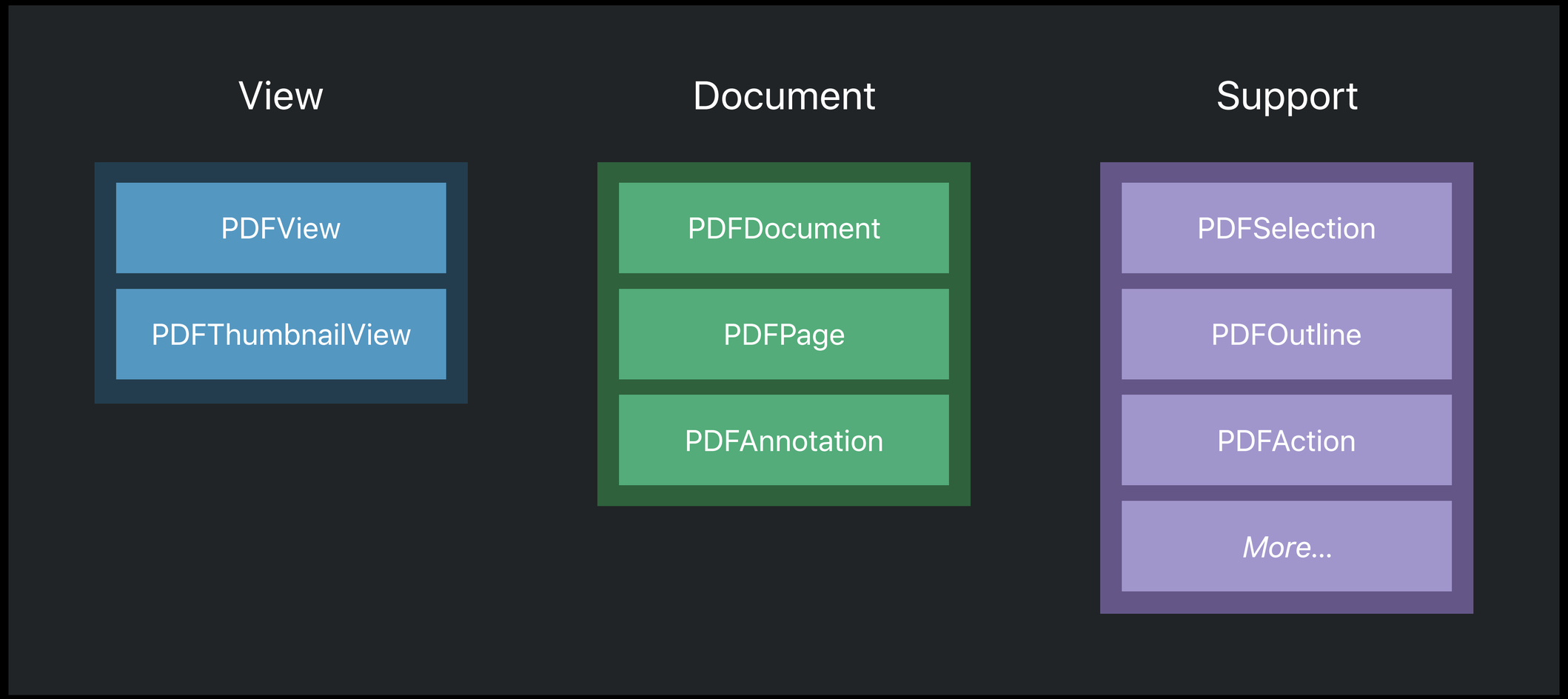

iOS



Framework Overview



Framework Overview



Framework Overview

Four core classes do most of what you need

PDFView | **PDFDocument** | **PDFPage** | **PDFAnnotation**

Demo

PDFView in action

Jeremy Bridon, Software Engineer

PDFView

Customizable PDF document view

Allows full user interaction with pages and widgets

Layout, direction, spacing, zoom factors, and auto-zoom

View-to-page, page-to-view coordinate conversion

PDFView

Customizable PDF document view

Allows full user interaction with pages and widgets

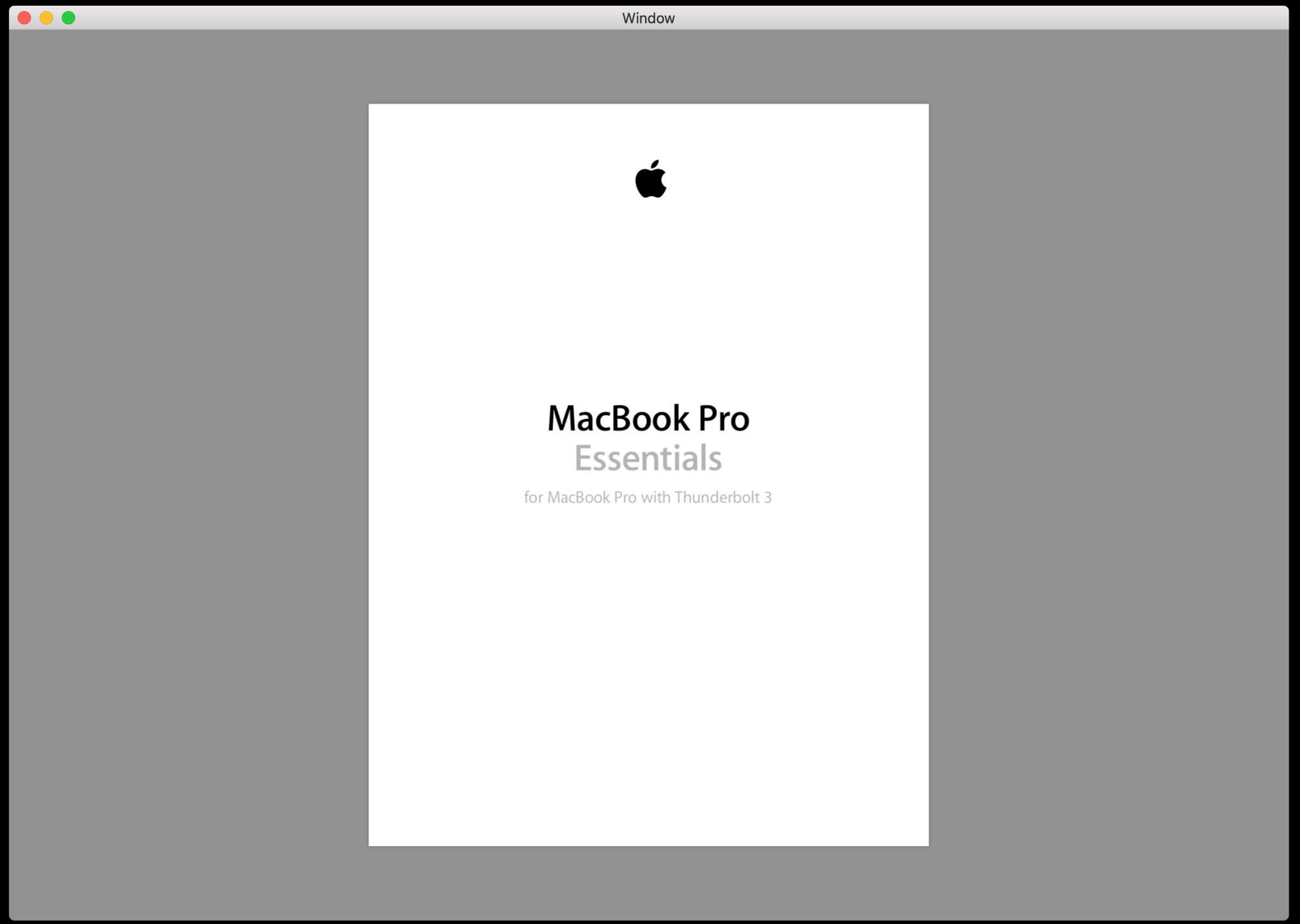
Layout, direction, spacing, zoom factors, and auto-zoom

View-to-page, page-to-view coordinate conversion

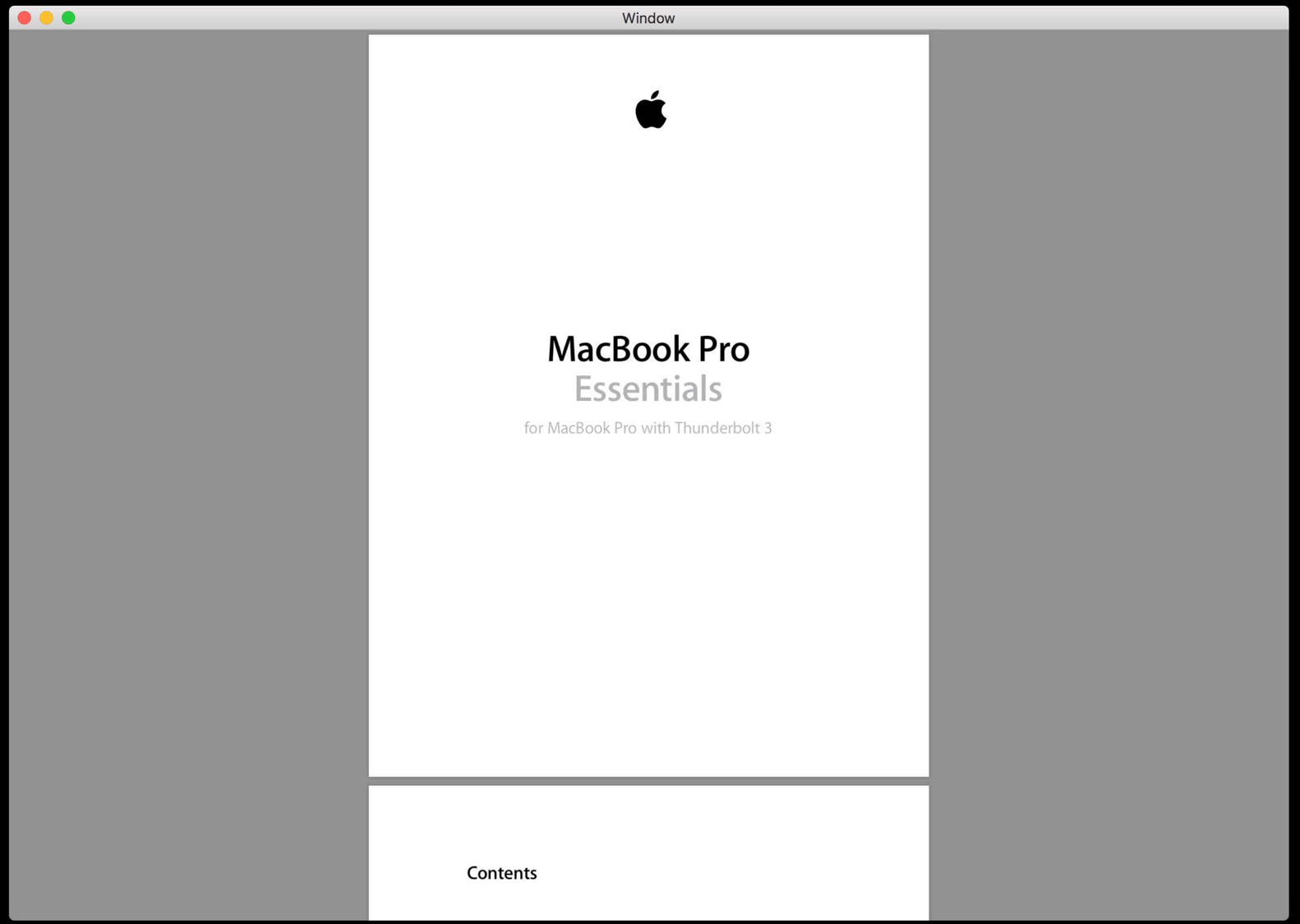
```
// Create our document and set it to the view
if let document = PDFDocument(url: documentURL) {
    pdfView.document = document
}
```

PDFDisplayMode

PDFDisplayMode.singlePage



PDFDisplayMode.singlePageContinuous



PDFDisplayMode.twoUp

Window

MacBook Pro
Essentials

for MacBook Pro with Thunderbolt 3

Contents

PDFDisplayMode.twoUpContinuous

Window

MacBook Pro
Essentials

for MacBook Pro with Thunderbolt 3

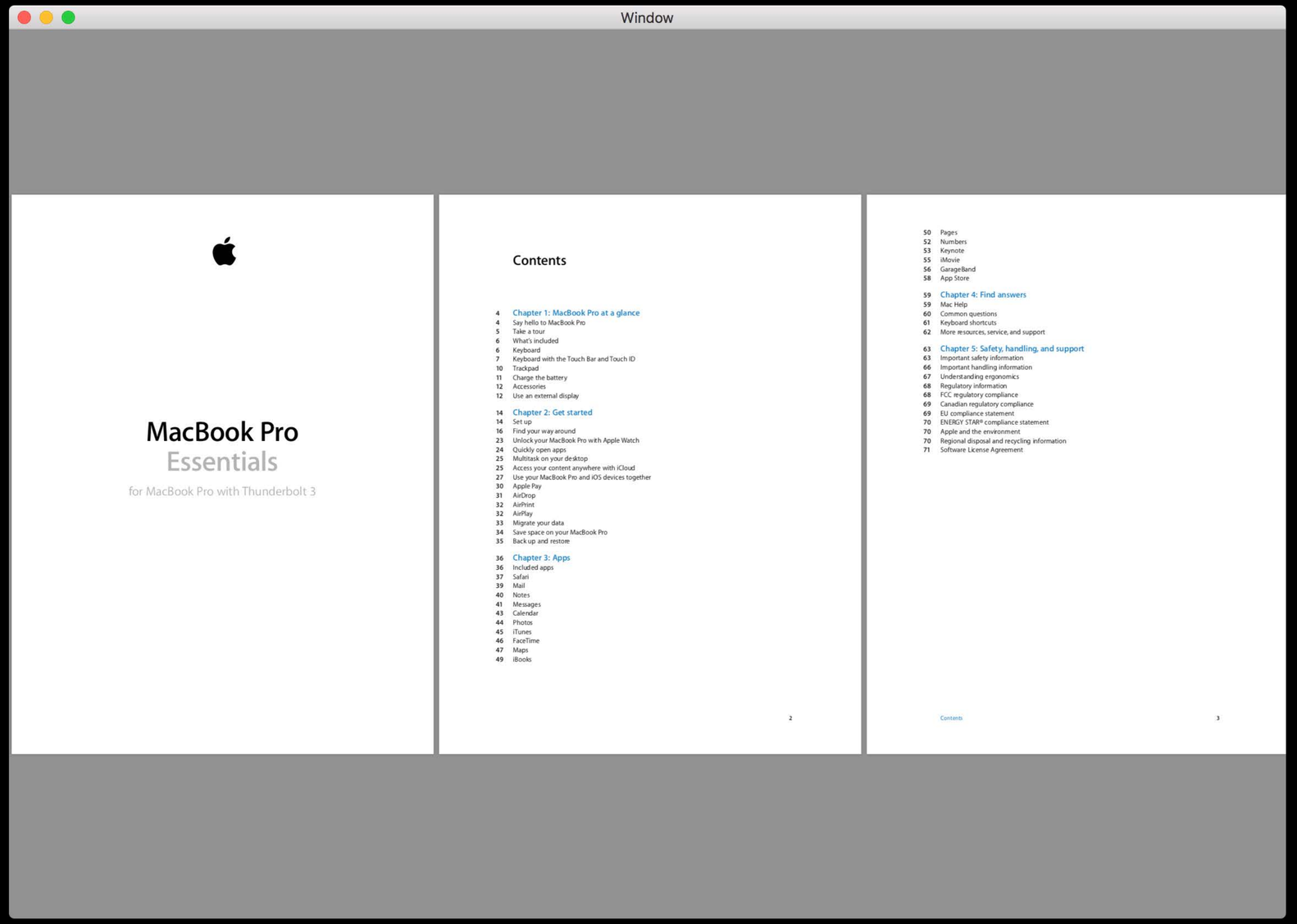
Contents

MacBook Pro at a glance 1

PDFDisplayDirection

PDFDisplayDirection.horizontal

NEW



View Pagification

MacBook Pro at a glance 1

Say hello to MacBook Pro

This guide provides the essential information you need in order to get the most from your MacBook Pro. The sections described below cover the hardware features, the software setup process and highlights, what you can do with apps on your Mac, and how to find more information about any topic.

Take a look around. Want a quick intro to the features of your MacBook Pro? Go to the next section, Take a tour.

Get started. Start your MacBook Pro by lifting the lid or connecting it to power, or by pressing the power button or Touch ID. Follow the Setup Assistant prompts, and you're up and running. For details, see Set up. To migrate your information from an older computer, see Migrate your data.

Make the most of shortcuts. If your MacBook Pro has a Touch Bar, shortcuts for common tasks are right at your fingertips. Change settings, use typing suggestions for text and messages, add an emoji, edit photos, and much more, with just a touch. See Meet the Touch Bar and Touch ID.

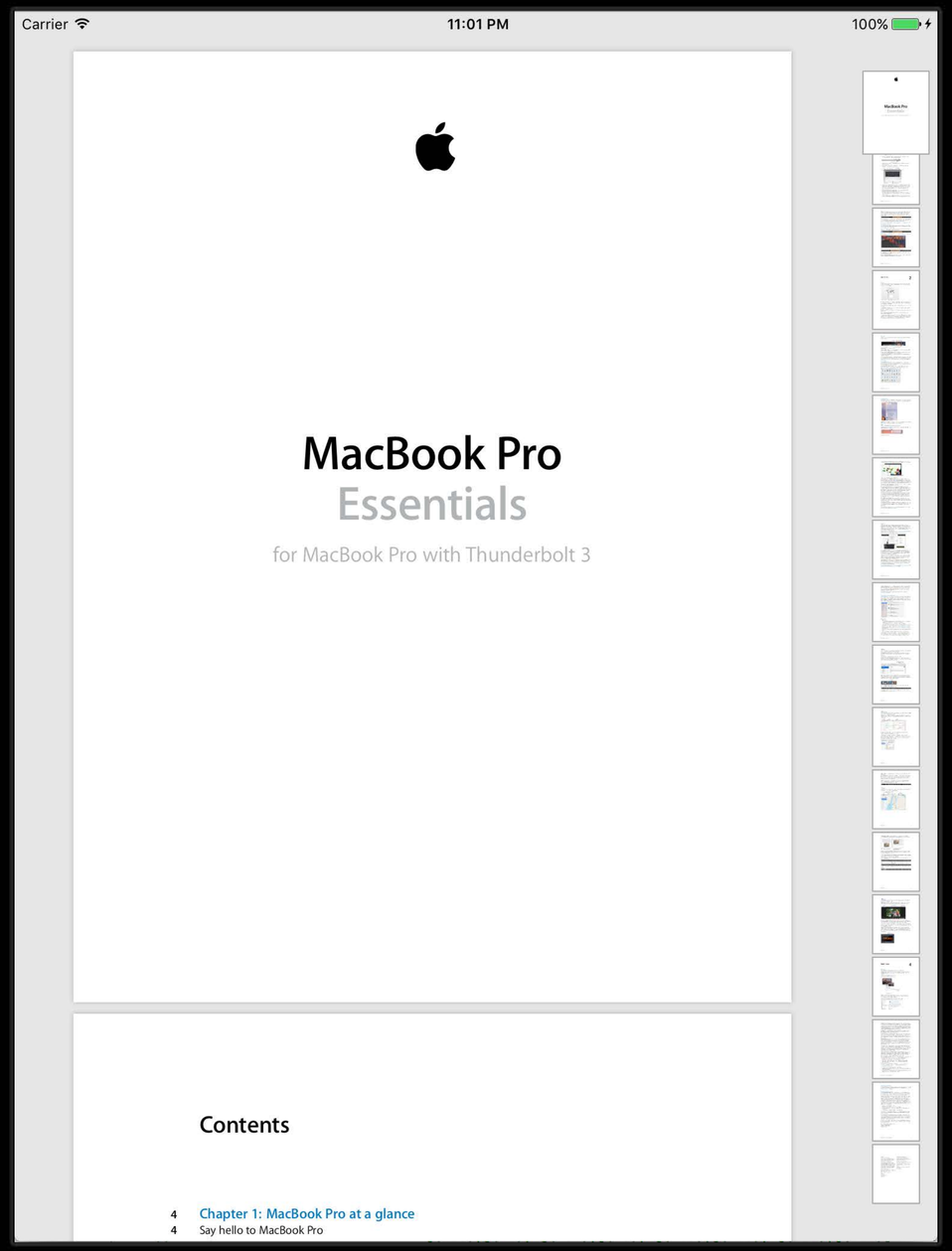
Stay in sync. Access your documents, photos, music, apps, contacts, and calendars across all your devices with iCloud. And use your MacBook Pro with your iOS devices to make and receive phone calls and texts, copy and paste across devices, or create an Instant Hotspot. Learn more in Access your content anywhere with iCloud and Continuity.

Unleash your creativity. Plan events and share info and photos with Notes; organize and listen to music, books, movies, and more with iTunes; create presentations with Keynote; and check out all the apps available on the App Store, to express yourself in as many ways as you have ideas.

Dig deeper. Explore your MacBook Pro and get your questions answered. Go to Mac Help.

PDFThumbnailView

NEW


Carrier
11:01 PM
100%
MacBook Pro
Essentials
for MacBook Pro with Thunderbolt 3
Contents
4 Chapter 1: MacBook Pro at a glance
4 Say hello to MacBook Pro

NEW

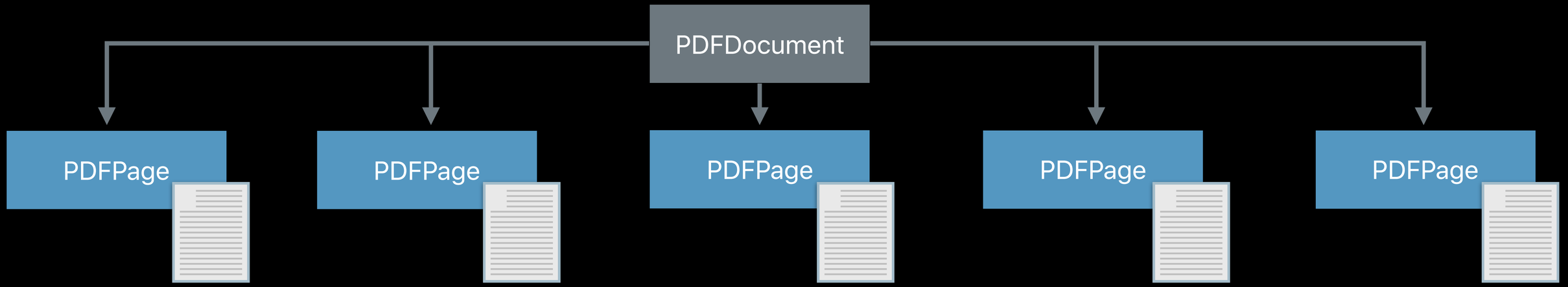
Document, Page, and Annotations Model

Document, Page, and Annotations Model

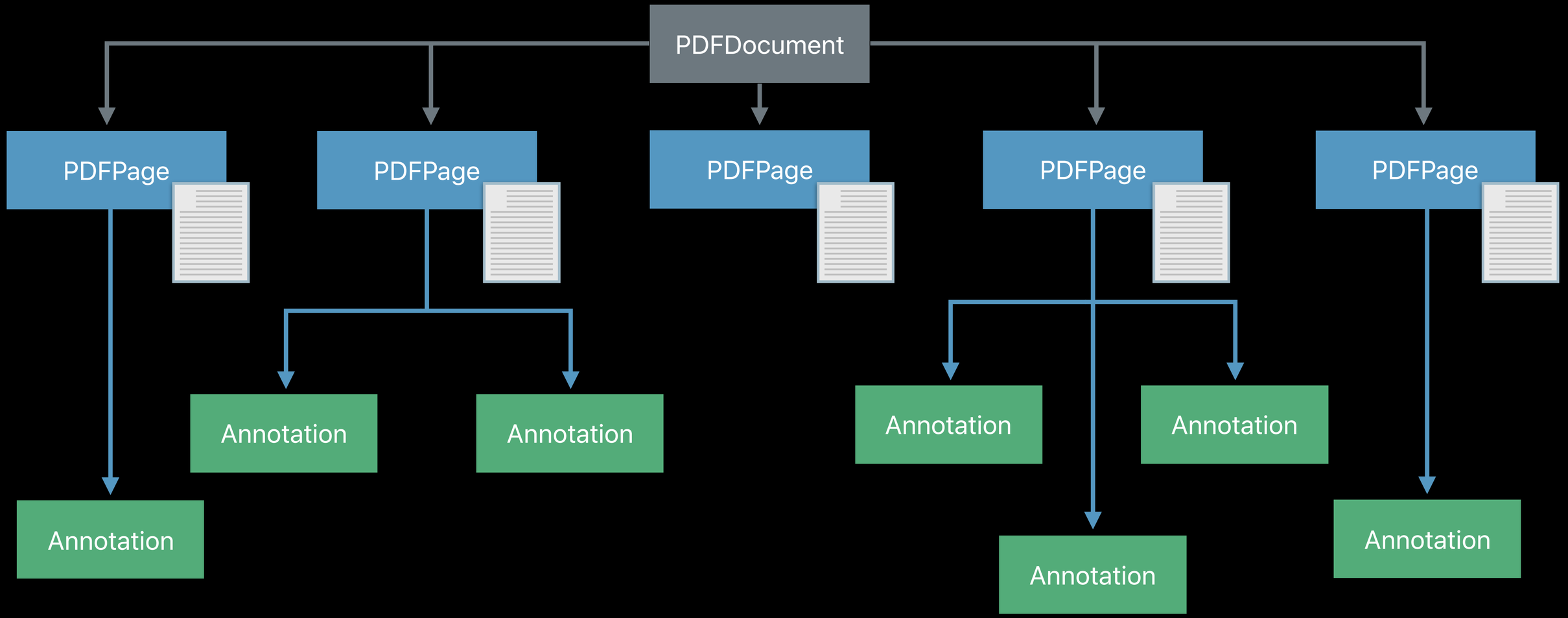
Document, Page, and Annotations Model

PDFDocument

Document, Page, and Annotations Model



Document, Page, and Annotations Model



PDFDocument

Page container: add, swap, and remove pages

Decrypt and verify permissions

Document attributes

Search strings

PDFDocument

Read and write

```
// Create a PDFDocument from a file
if let documentURL = Bundle.main.url(forResource: "Sample", withExtension: "pdf"),
   let document = PDFDocument(url: documentURL) {
    pdfView.document = document
}
```

PDFDocument

Read and write

```
// Create a PDFDocument from a file
if let documentURL = Bundle.main.url(forResource: "Sample", withExtension: "pdf"),
   let document = PDFDocument(url: documentURL) {
    pdfView.document = document
}

// Save file
document.write(to: documentURL)
```

PDFDocument

Read and write

```
// Create a PDFDocument from a file
if let documentURL = Bundle.main.url(forResource: "Sample", withExtension: "pdf"),
   let document = PDFDocument(url: documentURL) {
    pdfView.document = document
}

// Save file
document.write(to: documentURL)

// Save file with encryption
document.write(to: documentURL,
      withOptions: [.ownerPassword: "apple"])
```

PDFDocument

Page manipulation

```
// Retrieve a page
let myPage = document.page(at: 0)
```

PDFDocument

Page manipulation

```
// Retrieve a page
let myPage = document.page(at: 0)

// Add a page to end of document
document.insert(newPage, at: document.pageCount)
```

PDFDocument

Page manipulation

```
// Retrieve a page
let myPage = document.page(at: 0)

// Add a page to end of document
document.insert(newPage, at: document.pageCount)

// Exchange page pair
document.exchangePage(at: 0, withPageAt: 1)
```

PDFDocument

Page manipulation

```
// Retrieve a page
let myPage = document.page(at: 0)

// Add a page to end of document
document.insert(newPage, at: document.pageCount)

// Exchange page pair
document.exchangePage(at: 0, withPageAt: 1)

// Remove last page
document.removePage(at: document.pageCount - 1)
```

PDFDocument

Decryption

```
let document = PDFDocument(url: documentURL)

// Handle encrypted documents
if document.isEncrypted && document.unlock(withPassword: "apple") {
    if document.permissionsStatus == .owner {
        // Owner..
    } else {
        // User..
        if document.allowsCopying { /* ... */ }
        if document.allowsPrinting { /* ... */ }
    }
}
```

PDFDocument

Decryption

```
let document = PDFDocument(url: documentURL)

// Handle encrypted documents
if document.isEncrypted && document.unlock(withPassword: "apple") {
    if document.permissionsStatus == .owner {
        // Owner..
    } else {
        // User..
        if document.allowsCopying { /* ... */ }
        if document.allowsPrinting { /* ... */ }
    }
}
```

PDFDocument

Notifications Key:

```
Notification.Name.PDFDocumentDidUnlock
Notification.Name.PDFDocumentDidBeginWrite
...
```

PDFDocumentDelegate:

```
func documentDidFindMatch(_ notification: Notification)
func documentDidEndDocumentFind(_ notification: Notification)
...
```

PDFPage

PDFPage

Content container

Retrieved from document, initialized empty, or with image

Annotations container: add, retrieve, and remove annotations

Customize size, rotation, and custom drawing

Text selection

PDFPage

```
// Create a US letter sized page with an image
let image = UIImage(named: "image")
let newPage = PDFPage(image: image!)
```

PDFPage

```
// Create a US letter sized page with an image
let image = UIImage(named: "image")
let newPage = PDFPage(image: image!)

// Get the page contents
print("Page contents: \(newPage.string)”)
```

PDFPage

```
// Create a US letter sized page with an image
let image = UIImage(named: "image")
let newPage = PDFPage(image: image!)

// Get the page contents
print("Page contents: \(newPage.string)”)
textView.textStorage.setAttributedString(newPage.attributedString)
```

PDFPage

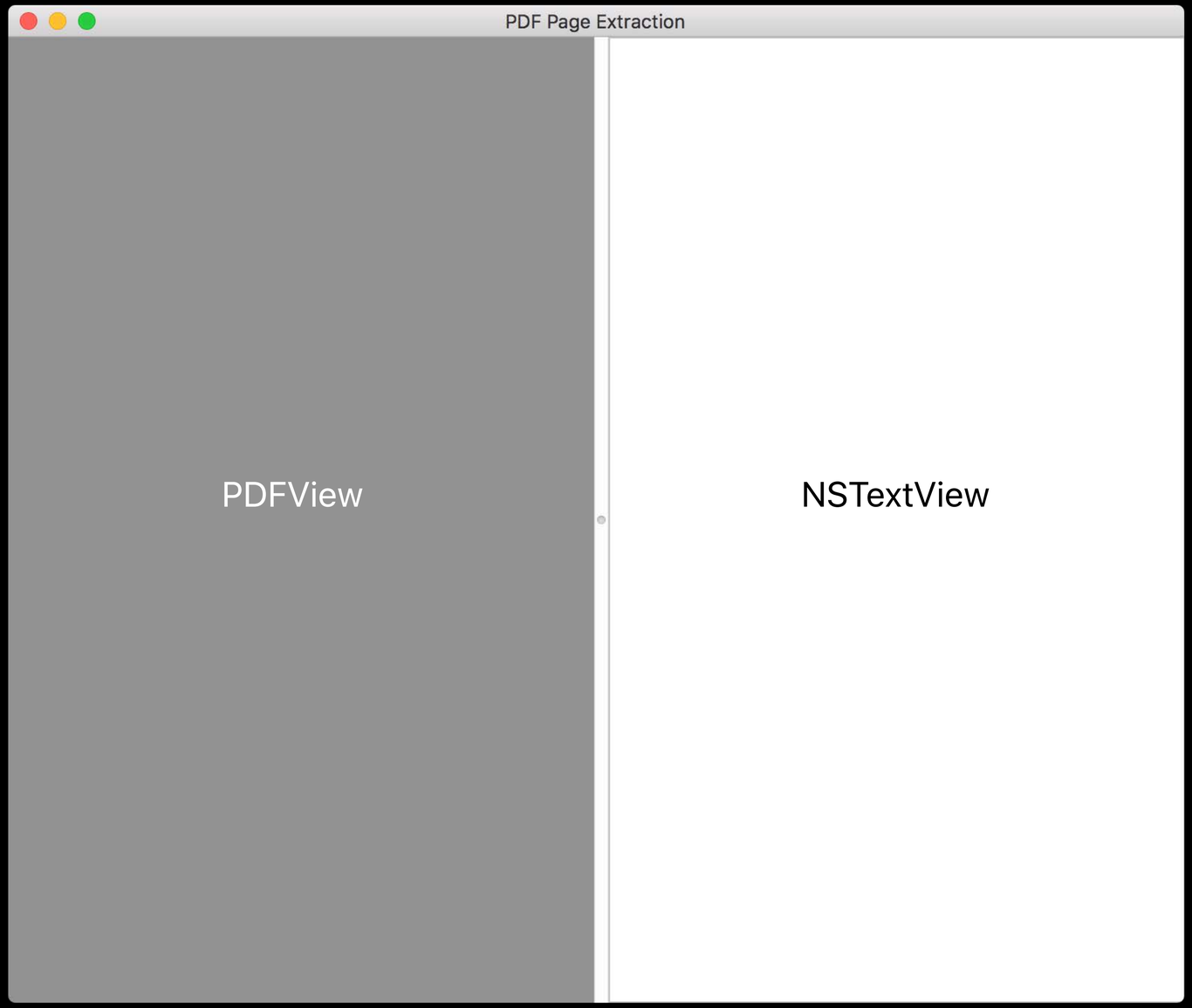
```
// Create a US letter sized page with an image
let image = UIImage(named: "image")
let newPage = PDFPage(image: image!)

// Get the page contents
print("Page contents: \(newPage.string)”)
textView.textStorage.setAttributedString(newPage.attributedString)

// Extract PDFSelection from substring range on page
let stringSelection = newPage.selection(for: NSRange(location: 10, length: 5))
```

PDFPage

```
// Create a US letter sized page with an image
let image = UIImage(named: "image")
let newPage = PDFPage(image: image!)

// Get the page contents
print("Page contents: \(newPage.string)")
textView.textStorage.setAttributedString(newPage.attributedString)

// Extract PDFSelection from substring range on page
let stringSelection = newPage.selection(for: NSRange(location: 10, length: 5))

// Extract PDFSelection from page-space rect
let rectSelection = newPage.selection(for: CGRect(x: 0, y: 0, width: 600, height: 200))
```

PDFPage

String extraction and Accessibility



PDFPage

String extraction and Accessibility

NEW

PDF Page Extraction

MacBook Pro at a glance 1

Say hello to MacBook Pro

This guide provides the essential information you need in order to get the most from your MacBook Pro. The sections described below cover the hardware features, the software setup process and highlights, what you can do with apps on your Mac, and how to find more information about any topic.

Take a look around. Want a quick intro to the features of your MacBook Pro? Go to the next section, Take a tour.

Get started. Start your MacBook Pro by lifting the lid or connecting it to power, or by pressing the power button or Touch ID. Follow the Setup Assistant prompts, and you're up and running. For details, see Set up. To migrate your information from an older computer, see Migrate your data.

Make the most of shortcuts. If your MacBook Pro has a Touch Bar, shortcuts for common tasks are right at your fingertips. Change settings, use typing suggestions for text and messages, add an emoji, edit photos, and much more, with just a touch. See Meet the Touch Bar and Touch ID.

Stay in sync. Access your documents, photos, music, apps, contacts, and calendars across all your devices with iCloud. And use your MacBook Pro with your iOS devices to make and receive phone calls and texts, copy and paste across devices, or create an Instant Hotspot. Learn more in Access your content anywhere with iCloud and Continuity.

Unleash your creativity. Plan events and share info and photos with Notes; organize and listen to music, books, movies, and more with iTunes; create presentations with Keynote; and check out all the apps available on the App Store, to express yourself in as many ways as you have ideas.

Dig deeper. Explore your MacBook Pro and get your questions answered. Go to Mac Help.

4

Take a tour

This guide is for MacBook Pro models with Thunderbolt 3 (USB-C). (Not all features are available on all models.)

MacBook Pro at a glance 1

Say hello to MacBook Pro

This guide provides the essential information you need in order to get the most from your MacBook Pro. The sections described below cover the hardware features, the software setup process and highlights, what you can do with apps on your Mac, and how to find more information about any topic.

Take a look around. Want a quick intro to the features of your MacBook Pro? Go to the next section, Take a tour.

Get started. Start your MacBook Pro by lifting the lid or connecting it to power, or by pressing the power button or Touch ID. Follow the Setup Assistant prompts, and you're up and running. For details, see Set up. To migrate your information from an older computer, see Migrate your data.

Make the most of shortcuts. If your MacBook Pro has a Touch Bar, shortcuts for common tasks are right at your fingertips. Change settings, use typing suggestions for text and messages, add an emoji, edit photos, and much more, with just a touch. See Meet the Touch Bar and Touch ID.

Stay in sync. Access your documents, photos, music, apps, contacts, and calendars across all your devices with iCloud. And use your MacBook Pro with your iOS devices to make and receive phone calls and texts, copy and paste across devices, or create an Instant Hotspot. Learn more in Access your content anywhere with iCloud and Continuity.

Unleash your creativity. Plan events and share info and photos with Notes; organize and listen to music, books, movies, and more with iTunes; create presentations with Keynote; and check out all the apps available on the App Store, to express yourself in as many ways as you have ideas.

Dig deeper. Explore your MacBook Pro and get your questions answered. Go to Mac Help.

4

PDFPage
Thumbnails

NEW

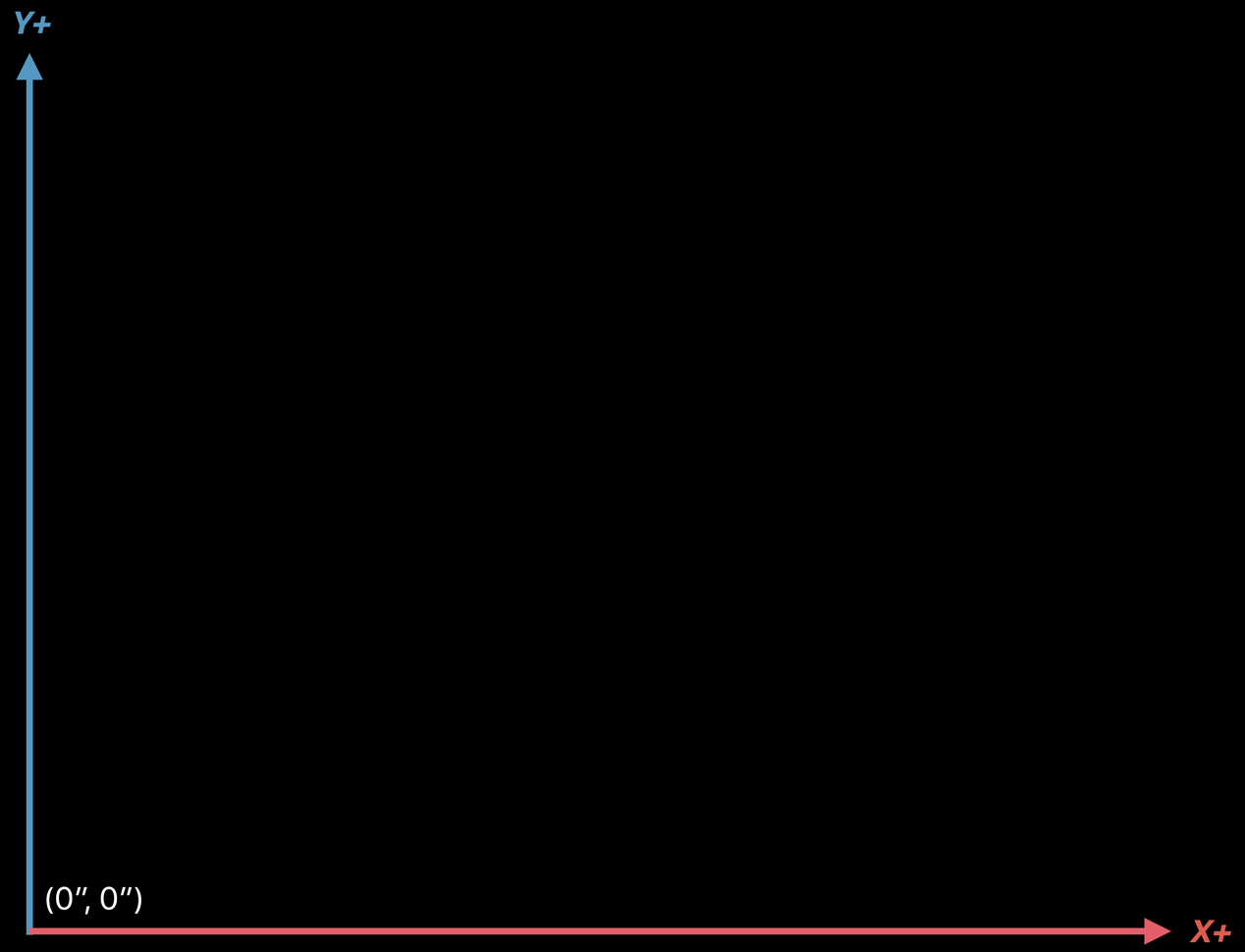
```
// For each page, create an image
for pageIndex in 0..<document.pageCount {
    if let page = document.page(at: pageIndex) {
        let pageThumbnail = page.thumbnail(of: CGSize(width: 100, height: 100),
                                           for: .cropBox)
            // pageThumbnail ready to use
    }
}
```

PDFPage
Thumbnails

NEW

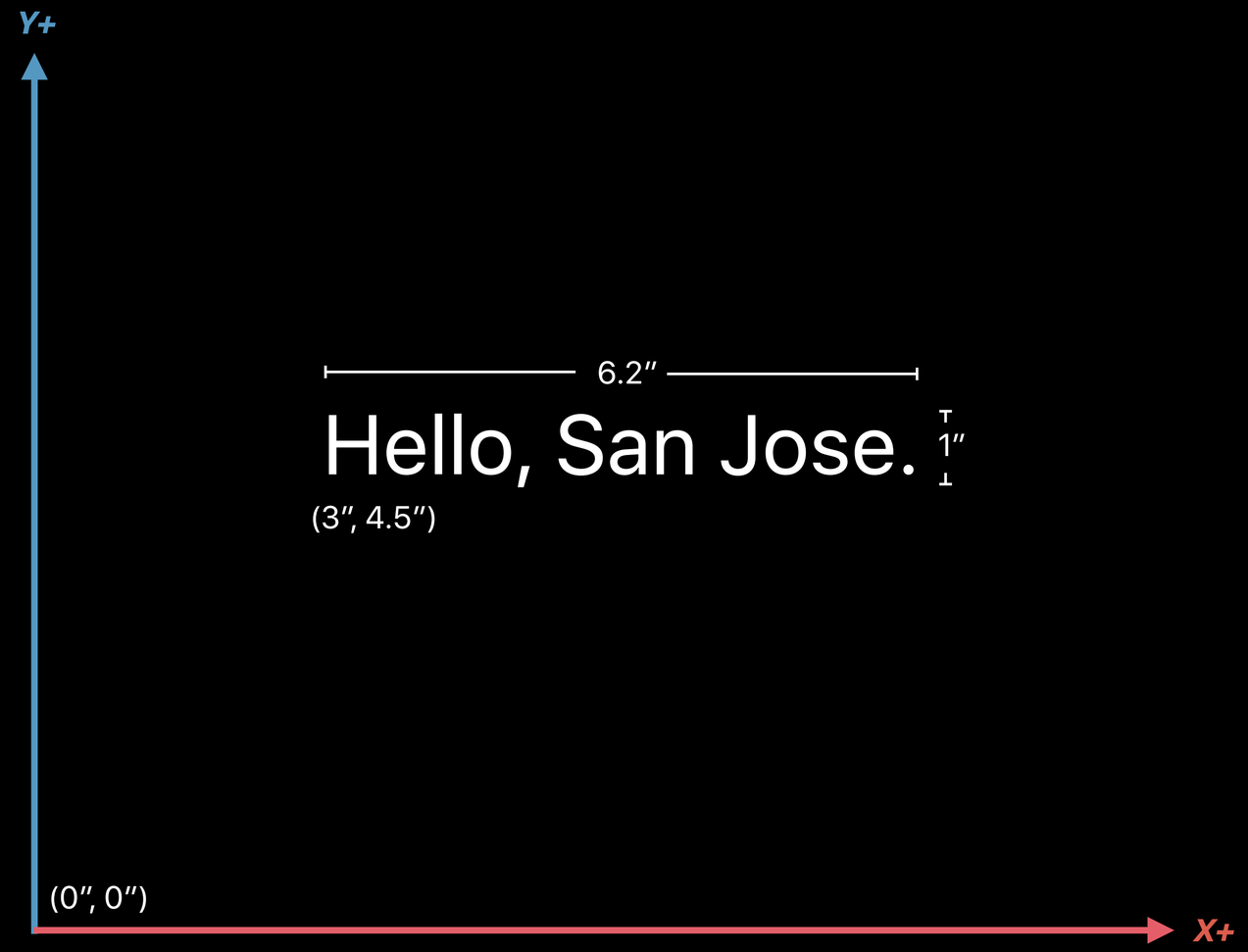
```
// For each page, create an image
for pageIndex in 0..<document.pageCount {
    if let page = document.page(at: pageIndex) {
        let pageThumbnail = page.thumbnail(of: CGSize(width: 100, height: 100),
                                           for: .cropBox)
            // pageThumbnail ready to use
    }
}
```

PDF Coordinate Space

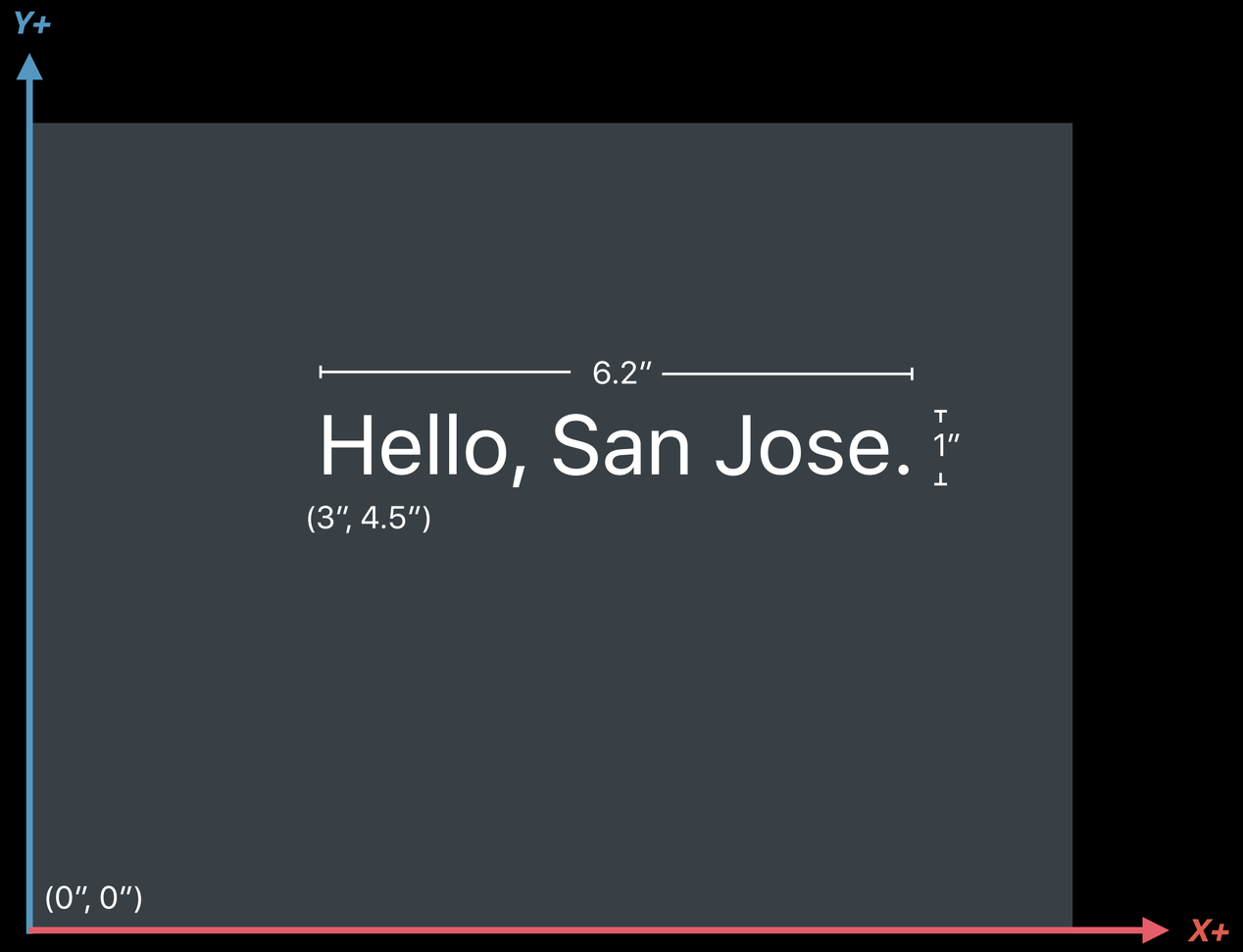


PDF Coordinate Space

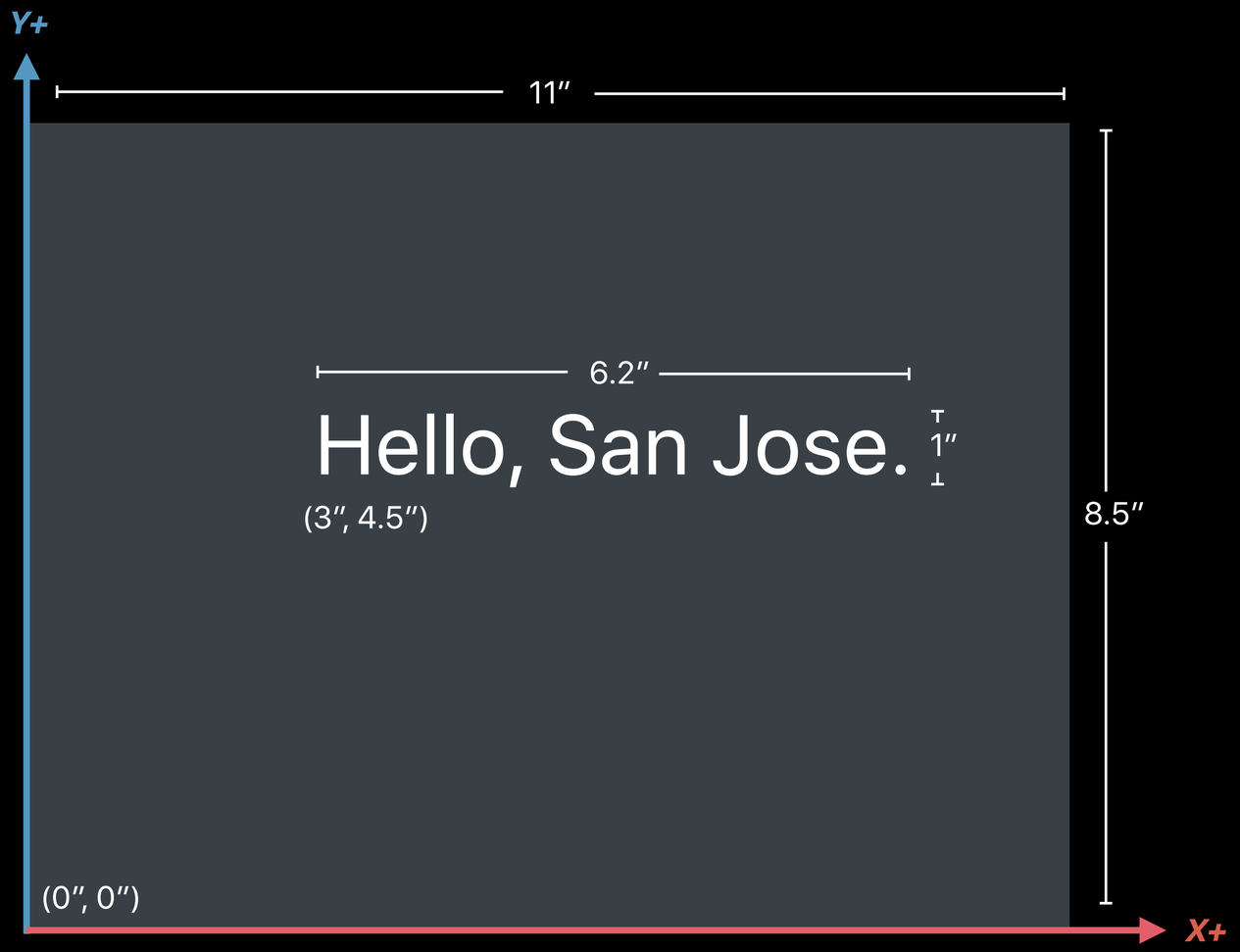
PDF Coordinate Space



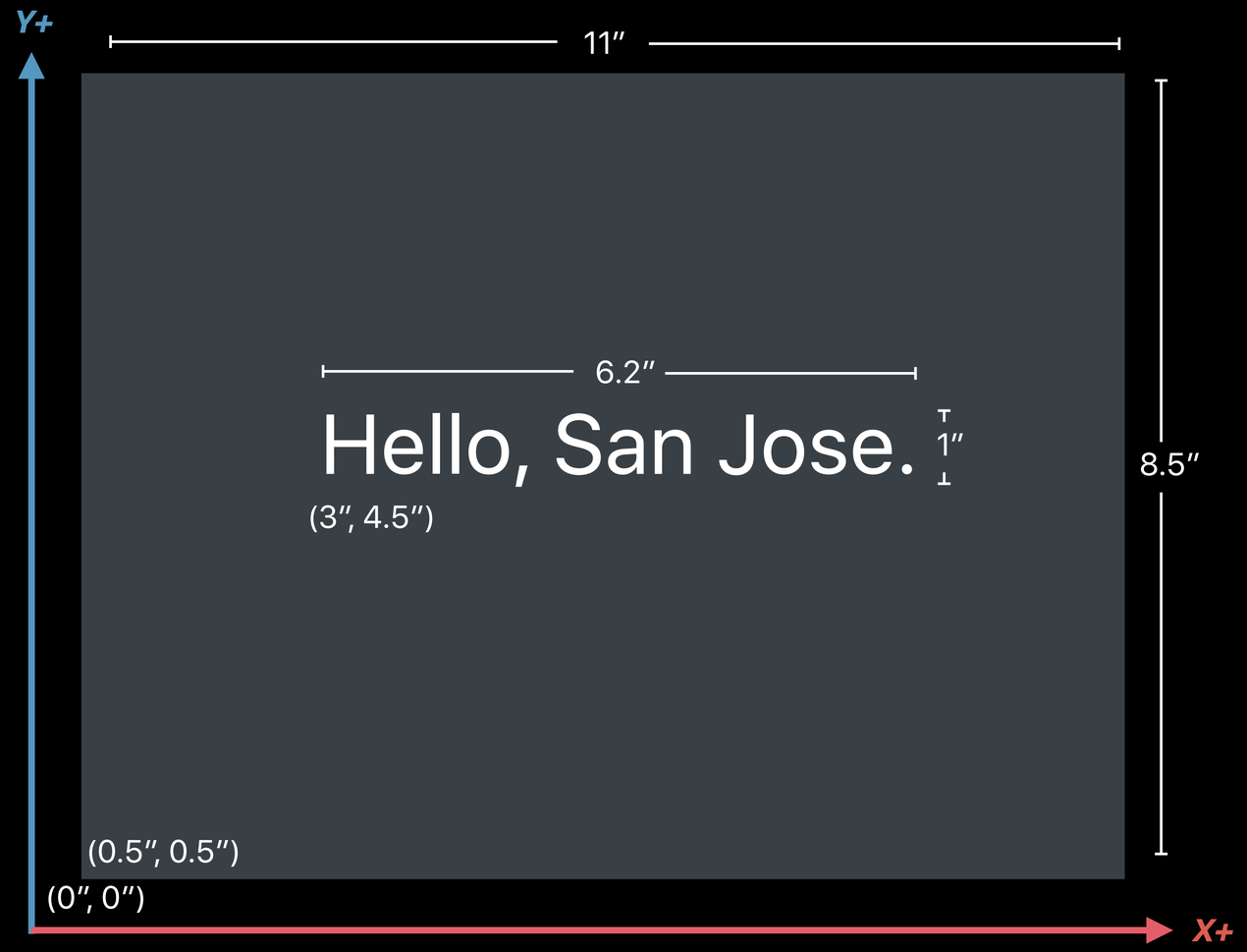
PDF Coordinate Space



PDF Coordinate Space



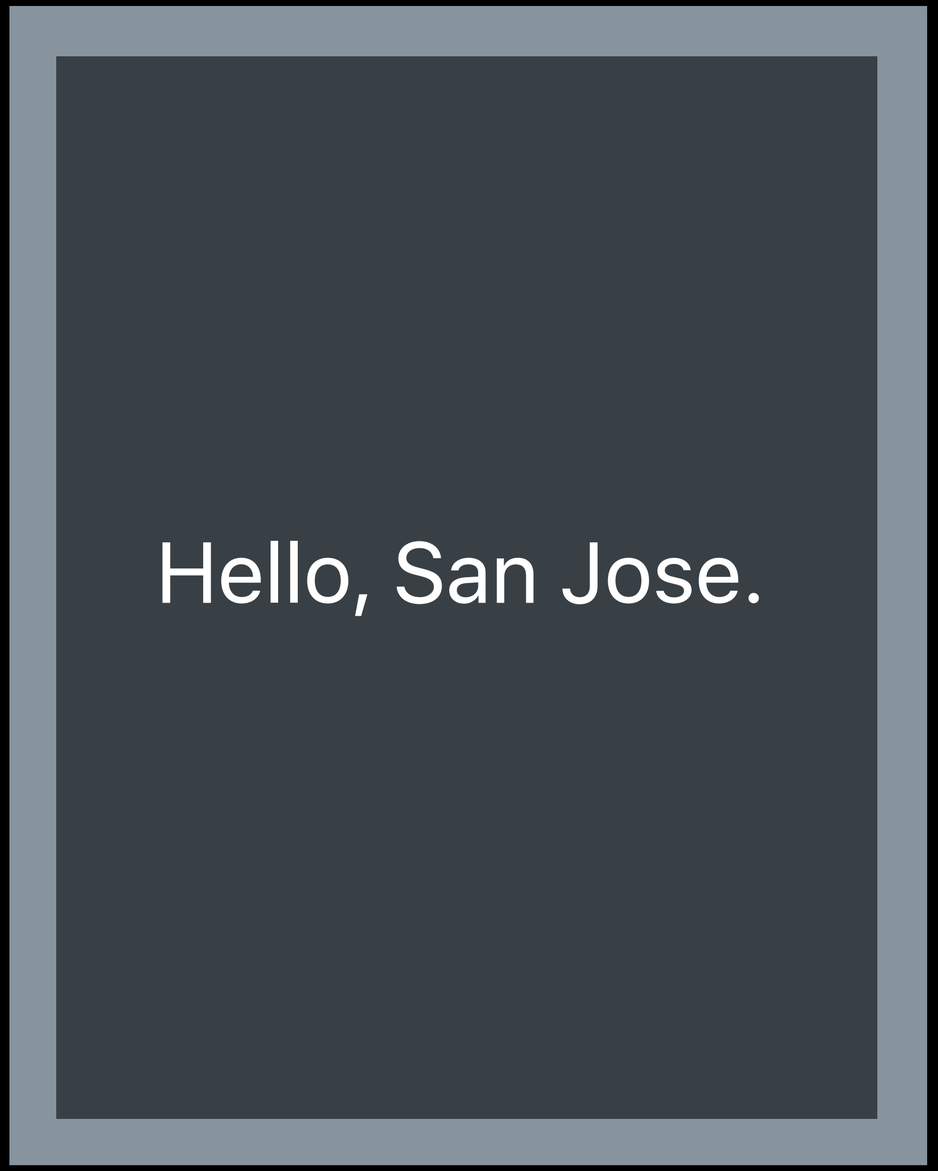
PDF Coordinate Space



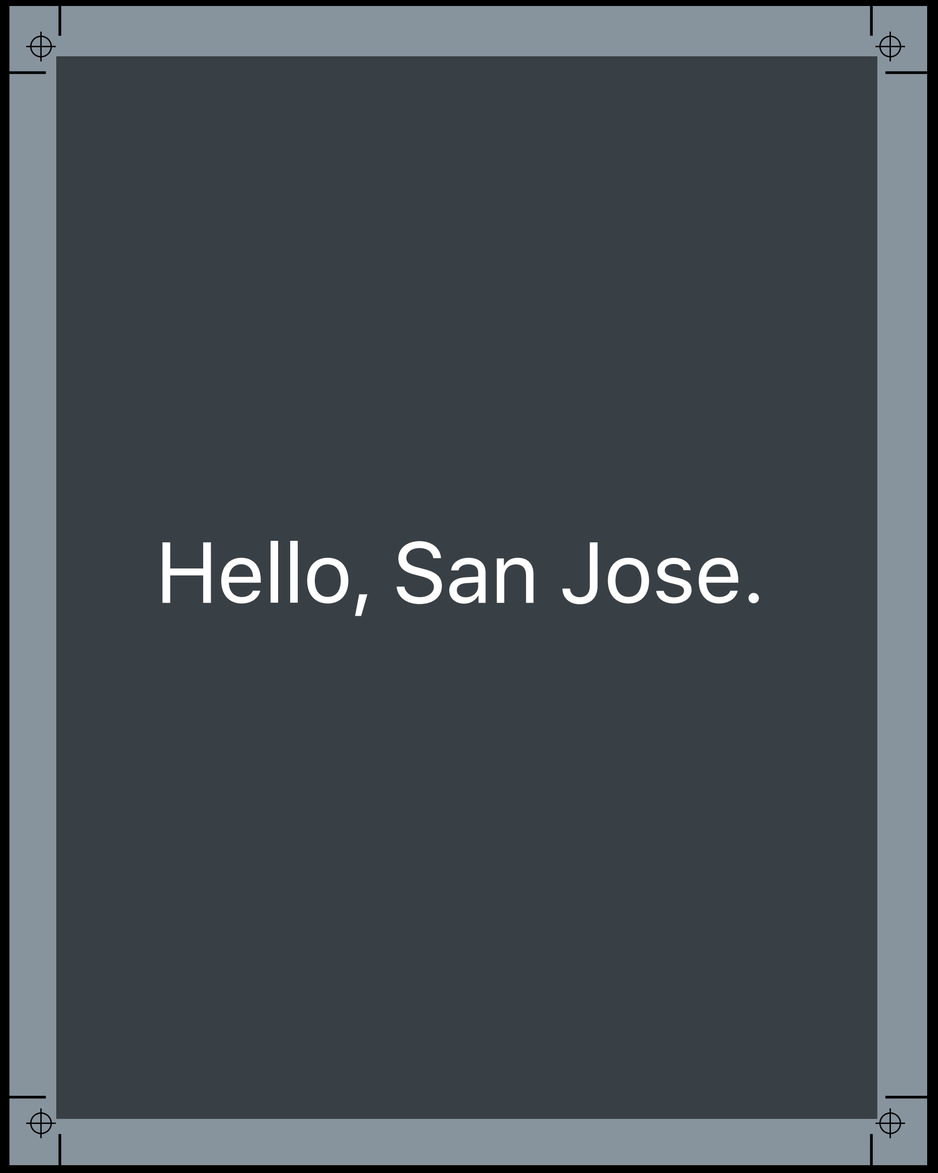
PDF Coordinate Space

PDF Coordinate Space

PDF Coordinate Space



PDF Coordinate Space



PDF Coordinate Space

Custom PDFPage Drawing

```
// 1. Register PDFDocument delegate
document.delegate = self
```

Custom PDFPage Drawing

```
// 1. Register PDFDocument delegate
document.delegate = self

// 2. Implement delegate method classForPage()
func classForPage() -> AnyClass {
    return WatermarkPage.self
}
```

Custom PDFPage Drawing

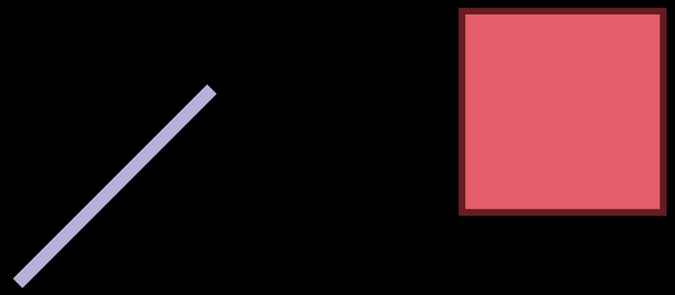
```
// 1. Register PDFDocument delegate
document.delegate = self

// 2. Implement delegate method classForPage()
func classForPage() -> AnyClass {
    return WatermarkPage.self
}

// 3. Subclass PDFPage class, override custom draw function
class WatermarkPage: PDFPage {
    override func draw(with box: PDFDisplayBox, to context: CGContext) {
        ...
```

Demo

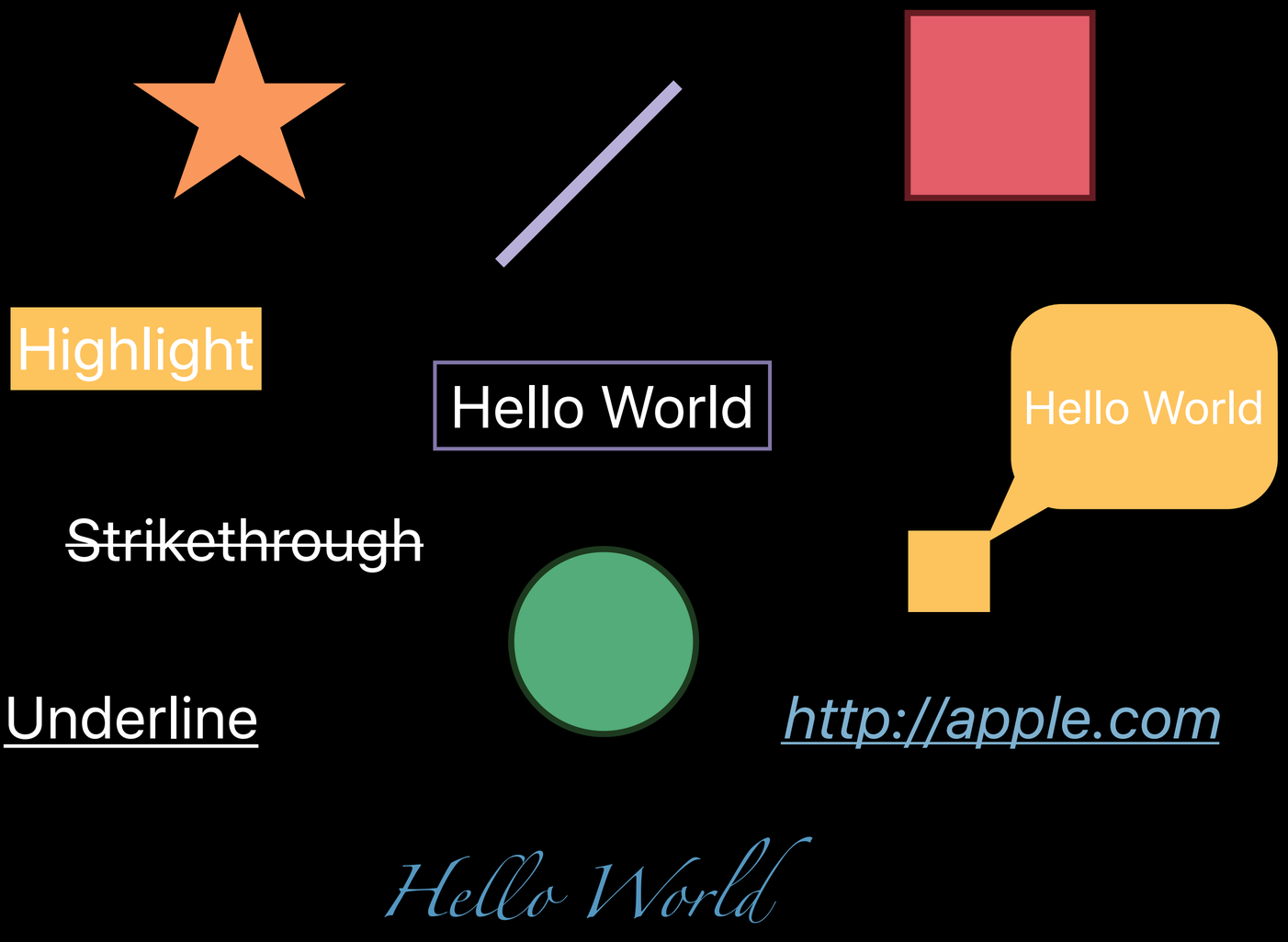
Watermarked pages

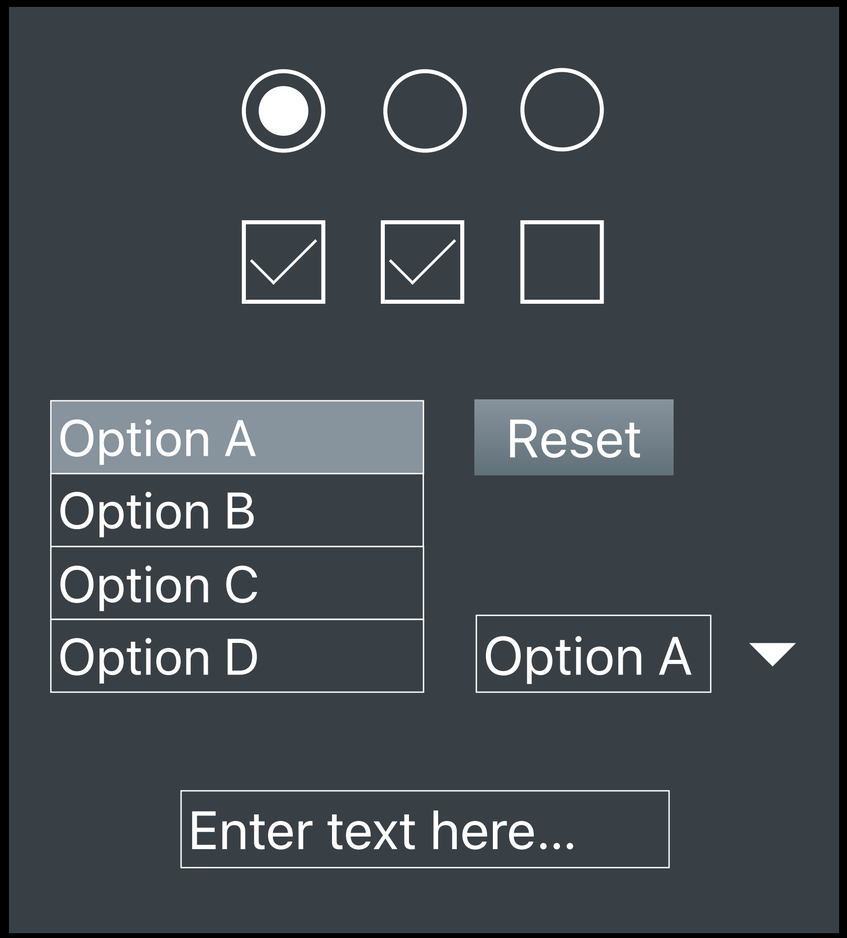
Annotations Supported by PDFKit

Annotations Supported by PDFKit



Annotations Supported by PDFKit





PDFAnnotation

PDFPages own annotations. You can add, modify, and remove

PDFView will update for value changes

Universal support via key-value pairs

- What you set in the dictionary gets set in the file
- Allows use of undefined annotations

PDFAnnotation

NEW

PDFPages own annotations. You can add, modify, and remove

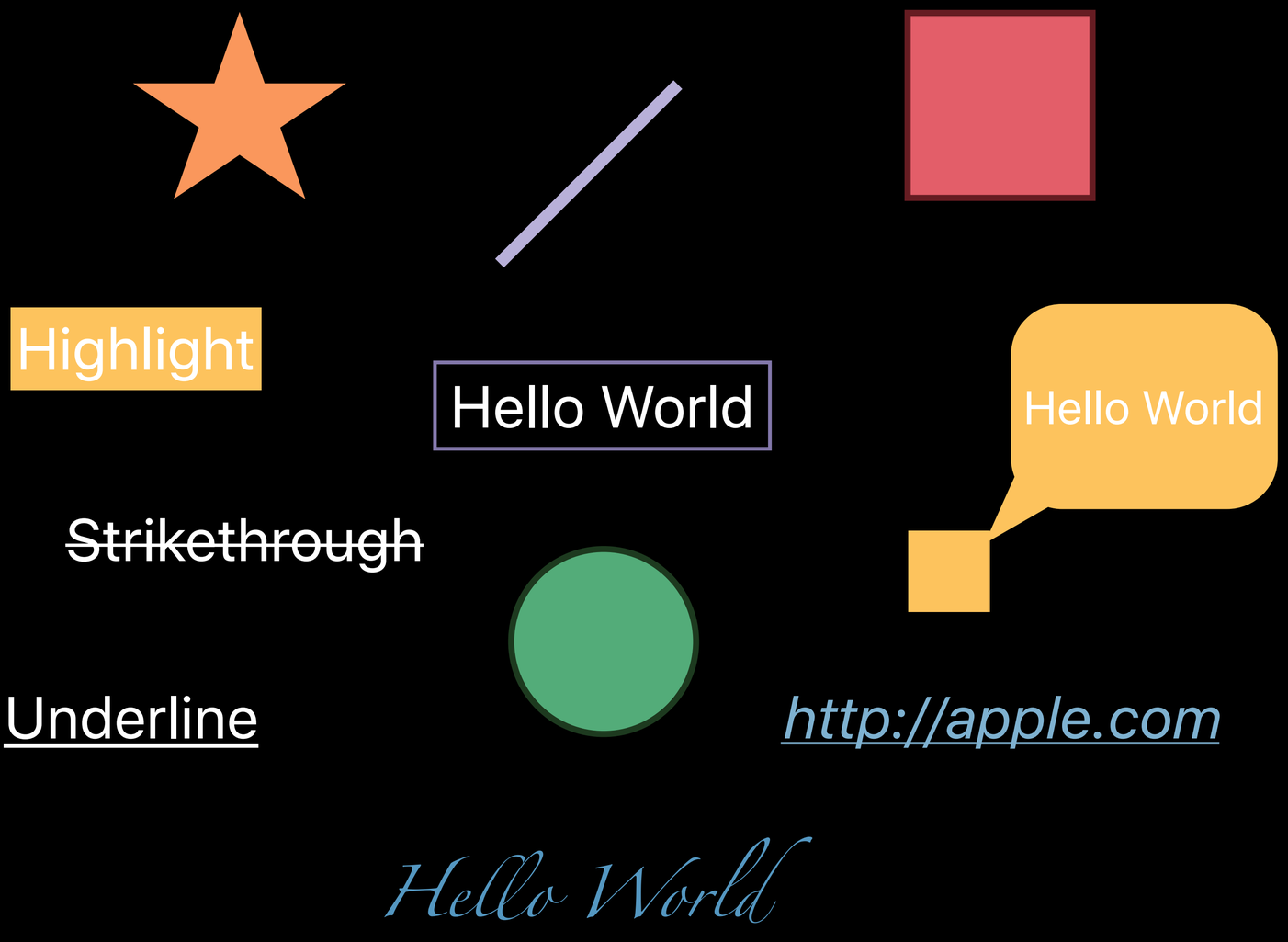
PDFView will update for value changes

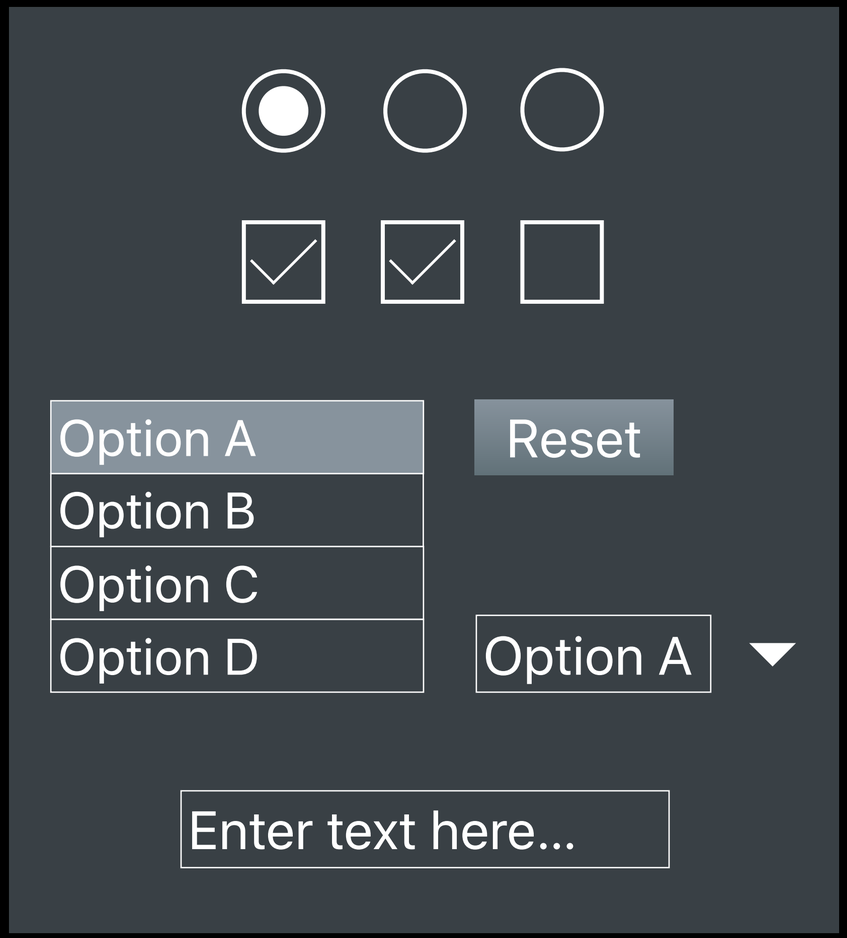
Universal support via key-value pairs

- What you set in the dictionary gets set in the file
- Allows use of undefined annotations

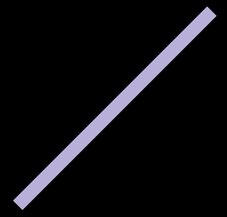
PDFAnnotationUtilities category methods

Annotations Supported by PDFKit

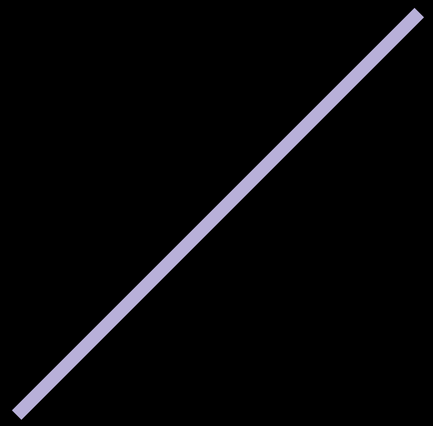




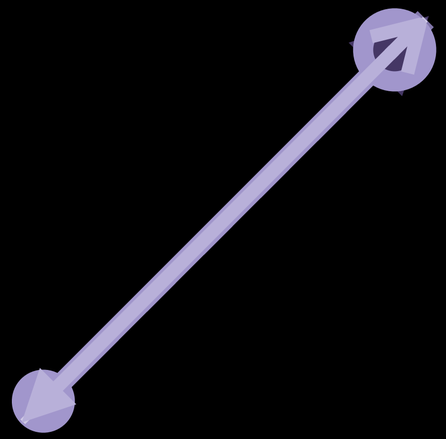
Annotations Supported by PDFKit



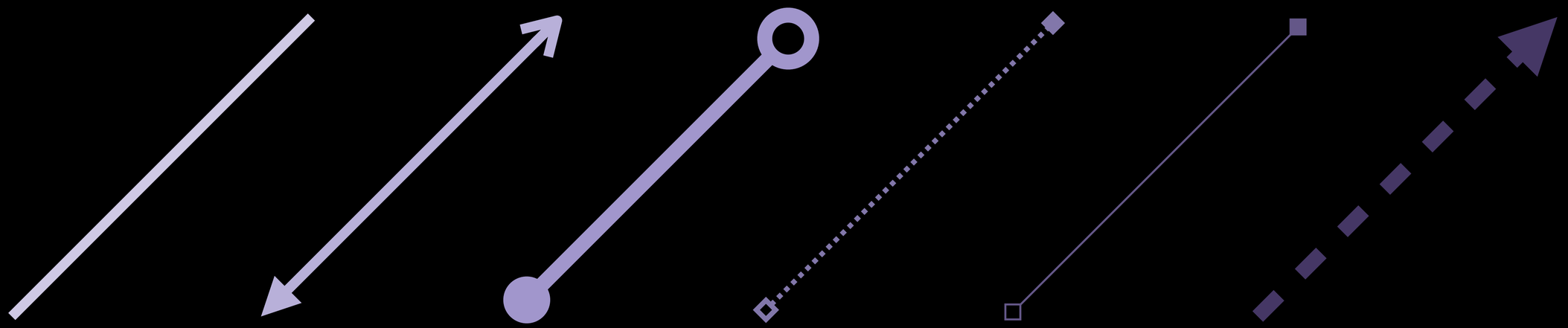
Annotations Supported by PDFKit



Annotations Supported by PDFKit



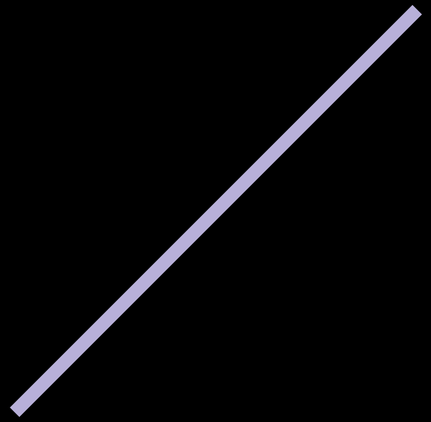
Annotations Supported by PDFKit



Annotations Supported by PDFKit

Properties of a line annotation:

- Start and end points

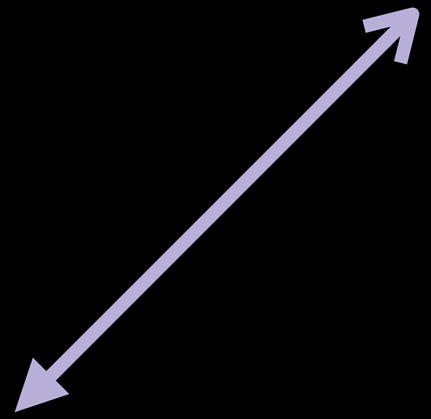


```
line.setValue([0, 0, 100, 100], forAnnotationKey: .linePoints)
```

Annotations Supported by PDFKit

Properties of a line annotation:

- Start and end points
- Line ending styles

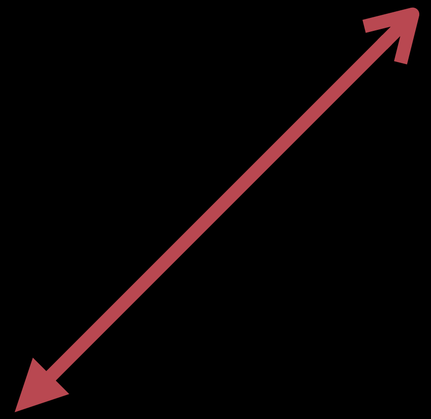


```
line.setValue([0, 0, 100, 100], forAnnotationKey: .linePoints)
line.setValue(["Closed", "Open"], forAnnotationKey: .lineEndingStyles)
```

Annotations Supported by PDFKit

Properties of a line annotation:

- Start and end points
- Line ending styles
- Color



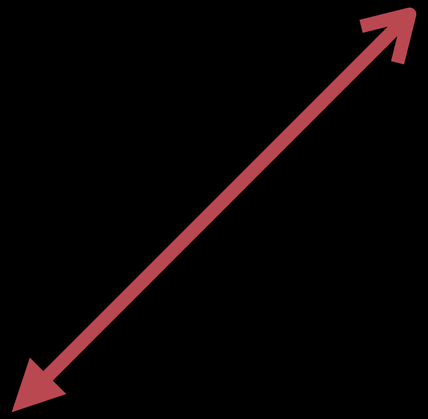
```
line.setValue([0, 0, 100, 100], forAnnotationKey: .linePoints)
line.setValue(["Closed", "Open"], forAnnotationKey: .lineEndingStyles)
line.setValue(UIColor.red, forAnnotationKey: .color)
```

Annotations Supported by PDFKit

NEW

Properties of a line annotation:

- Start and end points
- Line ending styles
- Color



```
line.startPoint = CGPoint(x: 0, y: 0)
line.endPoint = CGPoint(x: 100, y: 100)
line.startLineStyle = .closedArrow
line.endLineStyle = .openArrow
line.color = UIColor.red
```

Annotations Supported by PDFKit

NEW

```
// Universal key-value pairs:

line.setValue([0, 0, 100, 100], forAnnotationKey: .linePoints)
line.setValue(["Closed", "Open"], forAnnotationKey: .lineEndingStyles)
line.setValue(UIColor.red, forAnnotationKey: .color)
```

```
// PDFAnnotationUtilities:

line.startPoint = CGPoint(x: 0, y: 0)
line.endPoint = CGPoint(x: 100, y: 100)
line.startLineStyle = .closedArrow
line.endLineStyle = .openArrow
line.color = UIColor.red
```

PDFAnnotation

```
// Create an annotation to add to a page (empty)
let newAnnotation = PDFAnnotation(bounds: CGRect(x: 10, y: 10, width: 100, height: 100),
                                  forType: .square,
                              withProperties: nil)
```

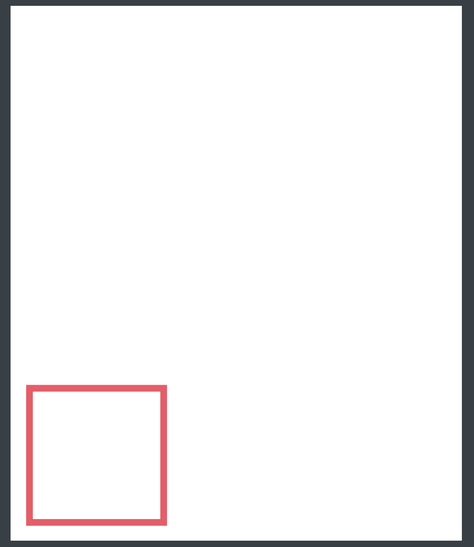
PDFAnnotation

```
// Create an annotation to add to a page (empty)
let newAnnotation = PDFAnnotation(bounds: CGRect(x: 10, y: 10, width: 100, height: 100),
                                  forType: .square,
                              withProperties: nil)

// Add additional properties to the annotation
newAnnotation.color = UIColor.red
let border = PDFBorder()
border.lineWidth = 2.0
newAnnotation.border = border
```

PDFAnnotation

```
// Create an annotation to add to a page (empty)
let newAnnotation = PDFAnnotation(bounds: CGRect(x: 10, y: 10, width: 100, height: 100),
                                  forType: .square,
                              withProperties: nil)

// Add additional properties to the annotation
newAnnotation.color = UIColor.red
let border = PDFBorder()
border.lineWidth = 2.0
newAnnotation.border = border
```

```
page.addAnnotation(newAnnotation)
```



PDFAnnotation

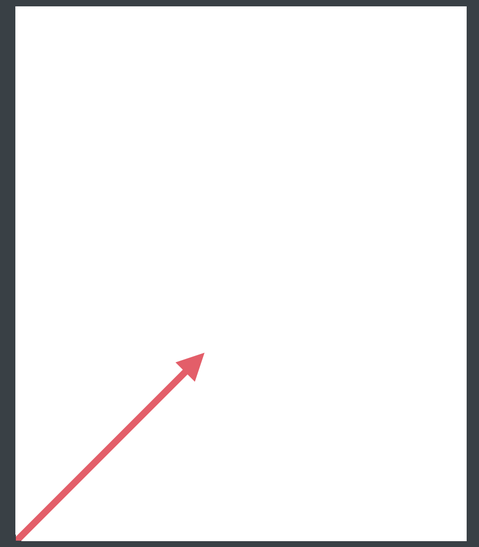
```
// Create dictionary of annotation properties
let lineAttributes: [PDFAnnotationKey: Any] = [
    .linePoints: [0, 0, 200, 200],
    .lineEndingStyles: [PDFAnnotationLineEndingStyle.none,
                        PDFAnnotationLineEndingStyle.closedArrow],
    .color: UIColor.red,
    .border: PDFBorder()
]
```

PDFAnnotation

```
// Create dictionary of annotation properties
let lineAttributes: [PDFAnnotationKey: Any] = [
    .linePoints: [0, 0, 200, 200],
    .lineEndingStyles: [PDFAnnotationLineEndingStyle.none,
                        PDFAnnotationLineEndingStyle.closedArrow],
    .color: UIColor.red,
    .border: PDFBorder()
]

let lineAnnotation = PDFAnnotation(bounds: CGRect(x: 0, y: 0, width: 200, height: 200),
                                   forType: .line,
                                   withProperties: lineAttributes)
```

PDFAnnotation

```
// Create dictionary of annotation properties
let lineAttributes: [PDFAnnotationKey: Any] = [
    .linePoints: [0, 0, 200, 200],
    .lineEndingStyles: [PDFAnnotationLineEndingStyle.none,
                        PDFAnnotationLineEndingStyle.closedArrow],
    .color: UIColor.red,
    .border: PDFBorder()
]

let lineAnnotation = PDFAnnotation(bounds: CGRect(x: 0, y: 0, width: 200, height: 200),
                                   forType: .line,
                               withProperties: lineAttributes)

page.addAnnotation(lineAnnotation)
```



PDFAnnotation

PDFAction and PDFDestination

```
// Create an action that allows the user to open a URL
let appleURL = URL(string: "http://apple.com")
let actionURL = PDFActionURL(url: appleURL)
linkAnnotation.action = actionURL

// Create an action that allows the user to jump to a PDFDestination
let destination = PDFDestination(page: myPage, at: CGPoint(x: 35, y: 275))
let actionGoTo = PDFActionGoTo(destination: destination)
linkAnnotation.action = actionGoTo
```

PDFAnnotation

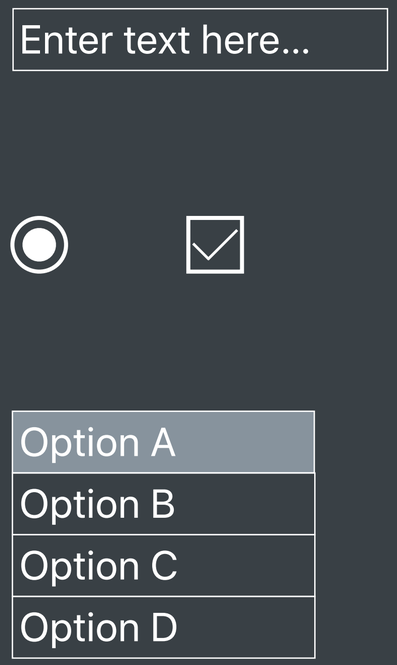
PDFAction and PDFDestination

```
// Create an action that allows the user to open a URL
let appleURL = URL(string: "http://apple.com")
let actionURL = PDFActionURL(url: appleURL)
linkAnnotation.action = actionURL

// Create an action that allows the user to jump to a PDFDestination
let destination = PDFDestination(page: myPage, at: CGPoint(x: 35, y: 275))
let actionGoTo = PDFActionGoTo(destination: destination)
linkAnnotation.action = actionGoTo
```

PDFAnnotation

Widgets

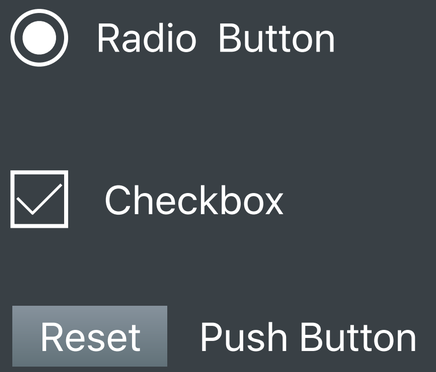
```
// Widget field types are important: PDFAnnotationWidgetSubtype

// Create a text widget
textWidget.widgetFieldType = .text

// Create a button widget
buttonWidget.widgetFieldType = .button

// Create a choice widget
choiceWidget.widgetFieldType = .choice
```



PDFAnnotation

Widgets

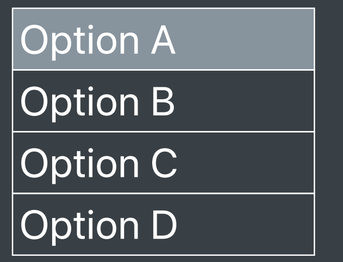
```
// Flavors of button widgets: PDFWidgetControlType

// Create a button widget
buttonWidget.widgetFieldType = .button

// Set button widget control type
buttonWidget.widgetControlType = .radioButtonControl

buttonWidget.widgetControlType = .checkBoxControl

buttonWidget.widgetControlType = .pushButtonControl
```



PDFAnnotation

Widgets

```
// Flavors of choice widgets

// Create a choice widget
choiceWidget.widgetFieldType = .text

// Create list box (default)
choiceWidget.isListChoice = true



// Create combo box
choiceWidget.isListChoice = false
```



PDFAnnotation

Widgets

```
// Create a widget annotation
let textField = PDFAnnotation(bounds: CGRect(x: 100, y: 200, width: 50, height: 20),
                              forType: .widget,
                        withProperties: nil)
```

PDFAnnotation

Widgets

```
// Create a widget annotation
let textField = PDFAnnotation(bounds: CGRect(x: 100, y: 200, width: 50, height: 20),
                              forType: .widget,
                       withProperties: nil)

// Use PDFAnnotation category methods to set text widget properties
textField.widgetFieldType = .text
textField.backgroundColor = UIColor.blue
textField.font = UIFont.systemFont(ofSize: 14.0)
textField.widgetStringValue = "WWDC 2017"
```

PDFAnnotation

Widgets

```
// Create a widget annotation
let textField = PDFAnnotation(bounds: CGRect(x: 100, y: 200, width: 50, height: 20),
                              forType: .widget,
                       withProperties: nil)

// Use PDFAnnotation category methods to set text widget properties
textField.widgetFieldType = .text
textField.backgroundColor = UIColor.blue
textField.font = UIFont.systemFont(ofSize: 14.0)
textField.widgetStringValue = "WWDC 2017"

page.addAnnotation(textField)
```

Demo

Advanced widget annotations

Best Practices

Recommended

- Use annotations for custom or real-time drawing
- Use PDFAnnotationUtilities for easy access to properties
- Custom draw functions (PDFPage and PDFView) must be thread-safe
- Custom PDFPage drawing should call super for original page content

Not Recommended

- Do not call PDFView's setNeedsDisplay to update content
- Do not mutate PDFPage from different threads
- Do not use deprecated drawing methods

Summary

Easy and extensible PDF application using AppKit/UIKit views

Easy to read, modify, and write PDF files

Secure with the latest encryption standard

Create your own forms, extract filled forms

Accessibility enabled

More Information

https://developer.apple.com/wwdc17/241


WWDC17