AirPrint

Session 234

Howard Miller
Printing Engineering

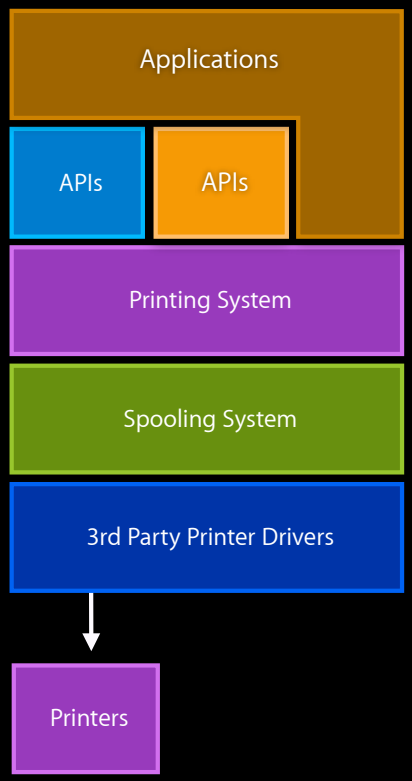
Agenda

- AirPrint Overview
- OS X Printing
- iOS Printing
- Printer Simulator

Printing System Architecture

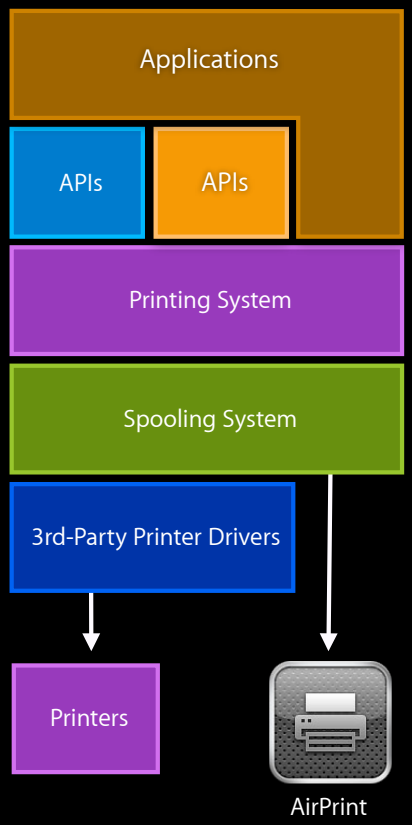
Printing System Architecture

Mac OS X

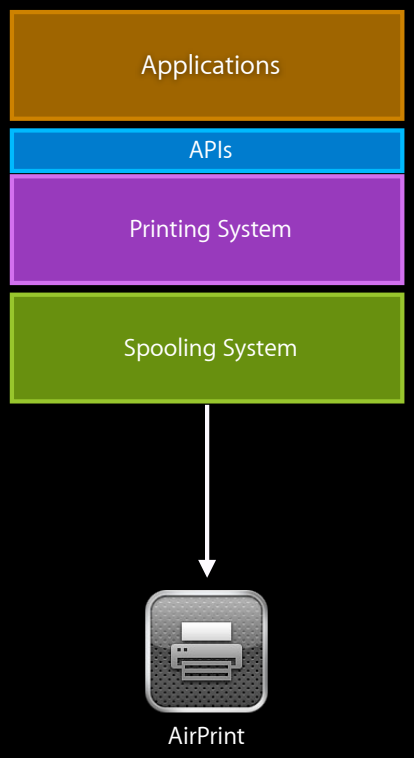


Printing System Architecture

Mac OS X



Printing System Architecture

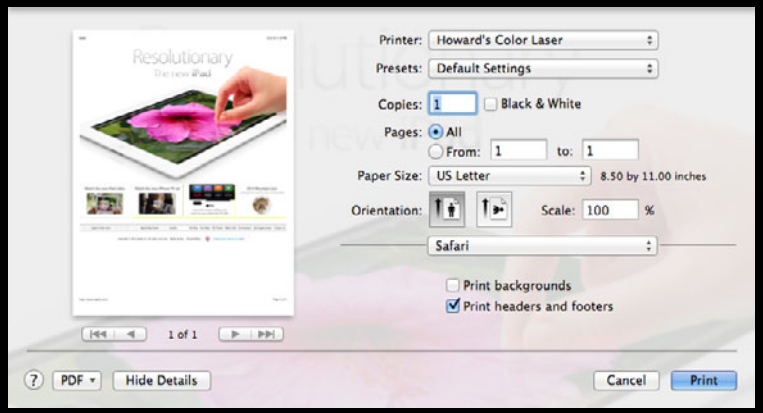


OS X Printing System

- User experience
  - Simple dialog is easy-to-use
  - Advanced dialog has many options
  - No drivers to install for AirPrint
  - Automatic download of drivers
- Great for application developers
  - Flexible and powerful printing system

OS X Printing System

- User experience
  - Simple dialog is easy-to-use
  - Advanced dialog has many options
  - No drivers to install for AirPrint
  - Automatic download of drivers
- Great for application developers
  - Flexible and powerful printing system



iOS Printing System

- User experience
  - Easy to use
  - Consistent high quality output
  - No drivers or software to install, no configuration
- Great for app developers
  - Easy to add printing support to your app
  - Flexible and powerful printing system

AirPrint Technology

- Great user experience
  - No driver, no software to install
  - Full output quality
- Supported on all Apple platforms
  - iOS 4.2 and later
  - OS X 10.7, default on OS X 10.8
- Standards based
- Zero cost license for printer manufacturers

AirPrint Manufacturers

AirPrint Manufacturers

Canon brother EPSON

SAMSUNG LEXMARK

hiti

FUJI xerox

KYOCERA

PRINICS

zink ZERO INK

csr

DELL

SHARP

Sagemcom

TOSHIBA

RICOH

Astro-Med, Inc TEST & MEASUREMENT PRODUCT GROUP

OKI

Panasonic Document Imaging Company

KONICA MINOLTA

Zebra

MARVELL

xerox

SINDOH

ALPS

AirPrint Printers

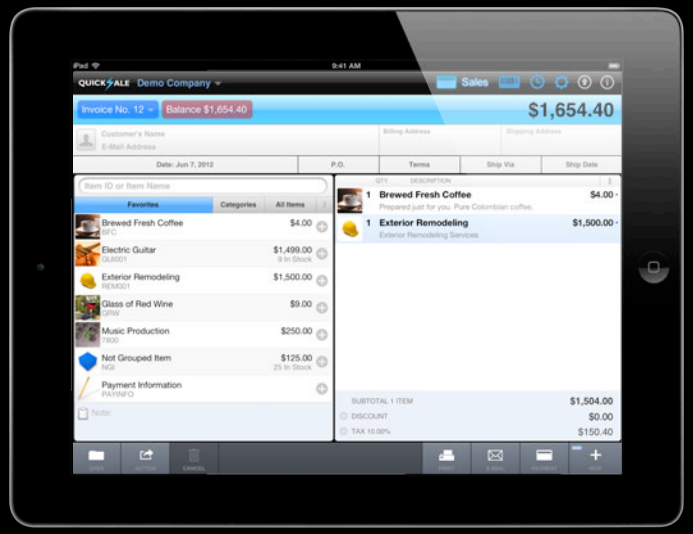
Over 75 Million

AirPrint Printers Sold

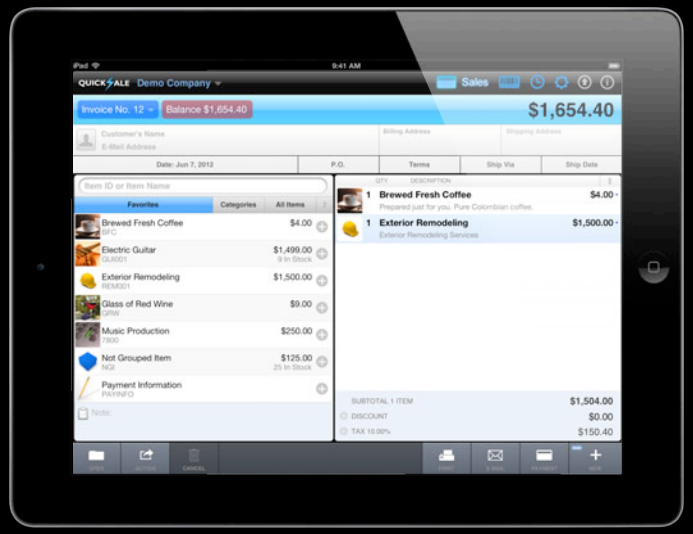
Summing up a Great Printing Application

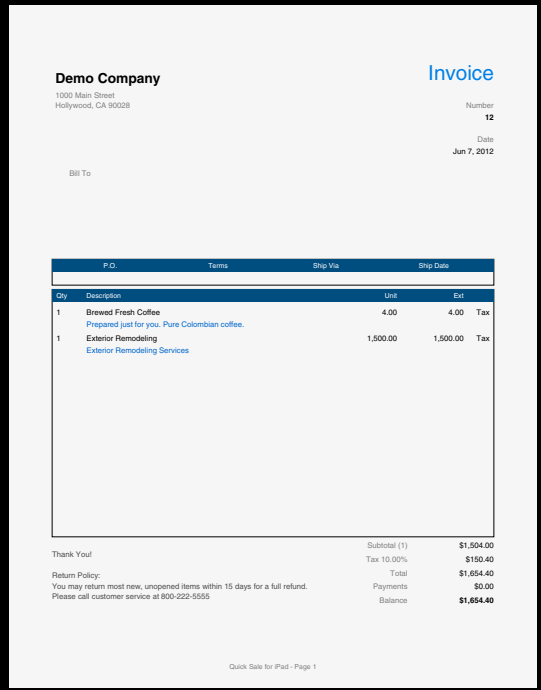
- Format for paper, not display
- Enhanced content
- High quality drawing

Summing up a Great Printing Application



Summing up a Great Printing Application





Demo Company
1000 Main Street
Hollywood, CA 90028

Invoice

Number
12

Date
Jun 7, 2012

Bill To

P.O.	Terms	Ship Via	Ship Date

Qty	Description	Unit	Ext	
1	Brewed Fresh Coffee Prepared just for you. Pure Colombian coffee.	4.00	4.00	Tax
1	Exterior Remodeling Exterior Remodeling Services	1,500.00	1,500.00	Tax

Thank You!

Return Policy:
You may return most new, unopened items within 15 days for a full refund.
Please call customer service at 800-222-5555

Subtotal (1)	$1,504.00
Tax 10.00%	$150.40
Total	$1,654.40
Payments	$0.00
Balance	**$1,654.40**

Quick Sale for iPad - Page 1

OS X Printing API

OS X 10.8 Printing API

- No new printing APIs

iOS Printing API

Session 234

Paul Danbold
Core OS Technologies Evangelist

iOS Printing Is Easy

iOS Printing Classes

```
UIPrintInteractionController
   UIPrintInfo
   UIPrintPaper
UIPrintFormatter
   UISimpleTextPrintFormatter
   UIMarkupTextPrintFormatter
   UIViewPrintFormatter
UIPrintPageRenderer
```

iOS Printing Classes

- UIPrintInteractionController
  - UIPrintInfo
  - UIPrintPaper
- UIPrintFormatter
  - UISimpleTextPrintFormatter
  - UIMarkupTextPrintFormatter
  - UIViewPrintFormatter
- UIPrintPageRenderer

iOS Printing Classes

```
UIPrintInteractionController
  UIPrintInfo
  UIPrintPaper
UIPrintFormatter
  UISimpleTextPrintFormatter
  UIMarkupTextPrintFormatter
  UIViewPrintFormatter
UIPrintPageRenderer
```

iOS Printing Classes

```
UIPrintInteractionController
   UIPrintInfo
   UIPrintPaper
UIPrintFormatter
   UISimpleTextPrintFormatter
   UIMarkupTextPrintFormatter
   UIViewPrintFormatter
UIPrintPageRenderer
```

iOS Printing Classes

```
UIPrintInteractionController
   UIPrintInfo
   UIPrintPaper
UIPrintFormatter
   UISimpleTextPrintFormatter
   UIMarkupTextPrintFormatter
   UIViewPrintFormatter
UIPrintPageRenderer
```

iOS Printing Classes

- UIPrintInteractionController
  - UIPrintInfo
  - UIPrintPaper
- UIPrintFormatter
  - UISimpleTextPrintFormatter
  - UIMarkupTextPrintFormatter
  - UIViewPrintFormatter
- UIPrintPageRenderer

Basic Steps

Basic Steps

- Get the print controller

Basic Steps

- Get the print controller
- Provide content to print

Basic Steps

- Get the print controller
- Provide content to print
- Specify the output type

Basic Steps

- Get the print controller
- Provide content to print
- Specify the output type
- Give the print job a name

Basic Steps

- Get the print controller
- Provide content to print
- Specify the output type
- Give the print job a name
- Present the printing UI

Printing a PDF

```
- (void)printFile:(NSURL *)url {

    if ([UIPrintInteractionController canPrintURL:url]) {

        UIPrintInteractionController *
           controller = [UIPrintInteractionController
sharedPrintController];

        controller.printingItem = url;

        UIPrintInfo *printInfo = [UIPrintInfo printInfo];
        printInfo.outputType = UIPrintInfoOutputGeneral;
        printInfo.jobName    = [url lastPathComponent];
        controller.printInfo = printInfo;

        controller.showsPageRange = YES;

        [controller presentAnimated:YES completionHandler:NULL];
    }
}
```

Printing a PDF

```
- (void)printFile:(NSURL *)url {

    if ([UIPrintInteractionController canPrintURL:url]) {

        UIPrintInteractionController *
           controller = [UIPrintInteractionController
sharedPrintController];

        controller.printingItem = url;

        UIPrintInfo *printInfo = [UIPrintInfo printInfo];
        printInfo.outputType = UIPrintInfoOutputGeneral;
        printInfo.jobName    = [url lastPathComponent];
        controller.printInfo = printInfo;

        controller.showsPageRange = YES;

        [controller presentAnimated:YES completionHandler:NULL];
    }
}
```

Printing a PDF

```
- (void)printFile:(NSURL *)url {

    if ([UIPrintInteractionController canPrintURL:url]) {

        UIPrintInteractionController *
           controller = [UIPrintInteractionController
sharedPrintController];

        controller.printingItem = url;

        UIPrintInfo *printInfo = [UIPrintInfo printInfo];
        printInfo.outputType = UIPrintInfoOutputGeneral;
        printInfo.jobName    = [url lastPathComponent];
        controller.printInfo = printInfo;

        controller.showsPageRange = YES;

        [controller presentAnimated:YES completionHandler:NULL];
    }
}
```

Printing a PDF

```
- (void)printFile:(NSURL *)url {

    if ([UIPrintInteractionController canPrintURL:url]) {

        UIPrintInteractionController *
           controller = [UIPrintInteractionController
sharedPrintController];

        controller.printingItem = url;

        UIPrintInfo *printInfo = [UIPrintInfo printInfo];
        printInfo.outputType = UIPrintInfoOutputGeneral;
        printInfo.jobName    = [url lastPathComponent];
        controller.printInfo = printInfo;

        controller.showsPageRange = YES;

        [controller presentAnimated:YES completionHandler:NULL];
    }
}
```

Printing a PDF

```
- (void)printFile:(NSURL *)url {

    if ([UIPrintInteractionController canPrintURL:url]) {

        UIPrintInteractionController *
           controller = [UIPrintInteractionController
sharedPrintController];

        controller.printingItem = url;

        UIPrintInfo *printInfo = [UIPrintInfo printInfo];
        printInfo.outputType = UIPrintInfoOutputGeneral;
        printInfo.jobName    = [url lastPathComponent];
        controller.printInfo = printInfo;

        controller.showsPageRange = YES;

        [controller presentAnimated:YES completionHandler:NULL];
    }
}
```

Printing a PDF

```
- (void)printFile:(NSURL *)url {

    if ([UIPrintInteractionController canPrintURL:url]) {

        UIPrintInteractionController *
           controller = [UIPrintInteractionController
sharedPrintController];

        controller.printingItem = url;

        UIPrintInfo *printInfo = [UIPrintInfo printInfo];
        printInfo.outputType = UIPrintInfoOutputGeneral;
        printInfo.jobName    = [url lastPathComponent];
        controller.printInfo = printInfo;

        controller.showsPageRange = YES;

        [controller presentAnimated:YES completionHandler:NULL];
    }
}
```

Printing a PDF

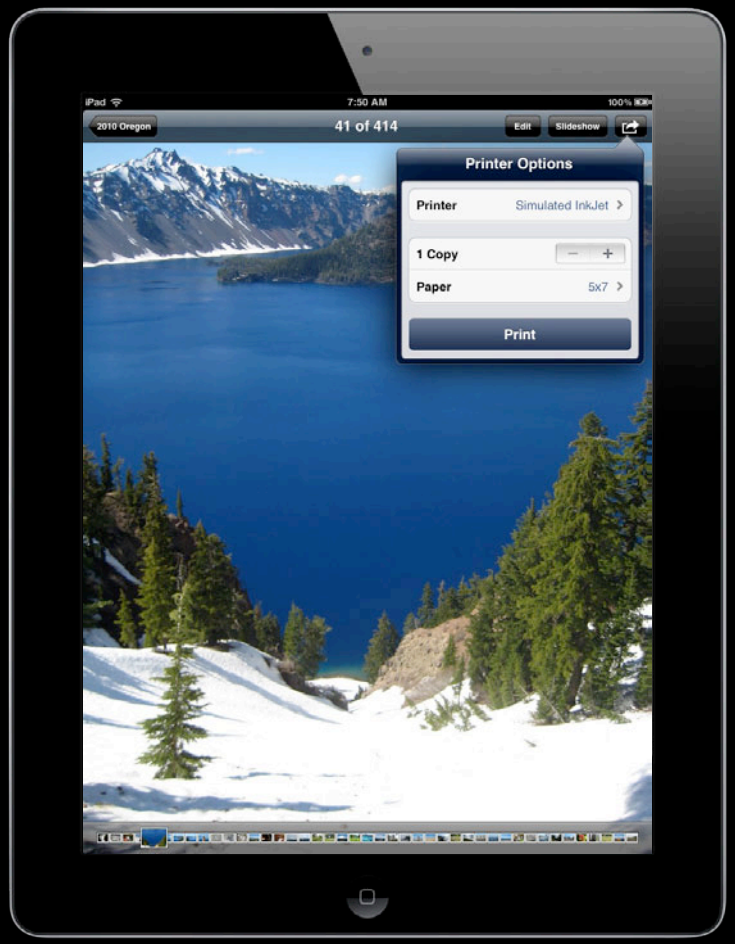
```
- (void)printFile:(NSURL *)url {

    if ([UIPrintInteractionController canPrintURL:url]) {

        UIPrintInteractionController *
           controller = [UIPrintInteractionController
sharedPrintController];

        controller.printingItem = url;

        UIPrintInfo *printInfo = [UIPrintInfo printInfo];
        printInfo.outputType = UIPrintInfoOutputGeneral;
        printInfo.jobName    = [url lastPathComponent];
        controller.printInfo = printInfo;

        controller.showsPageRange = YES;

        [controller presentAnimated:YES completionHandler:NULL];
    }
}
```

Output Type

- Tell the printing system about the type of content to be printed
- Allows the printing system to choose appropriate
  - Paper size
  - Print quality mode
  - Appropriate UI

UIPrintInfoOutputPhoto

- High quality
- Photo paper size
- Borderless if available
- No duplex mode
- No page range



UIPrintInfoOutputPhoto

- High quality
- Photo paper size
- Borderless if available
- No duplex mode
- No page range

UIPrintInfoOutputGeneral

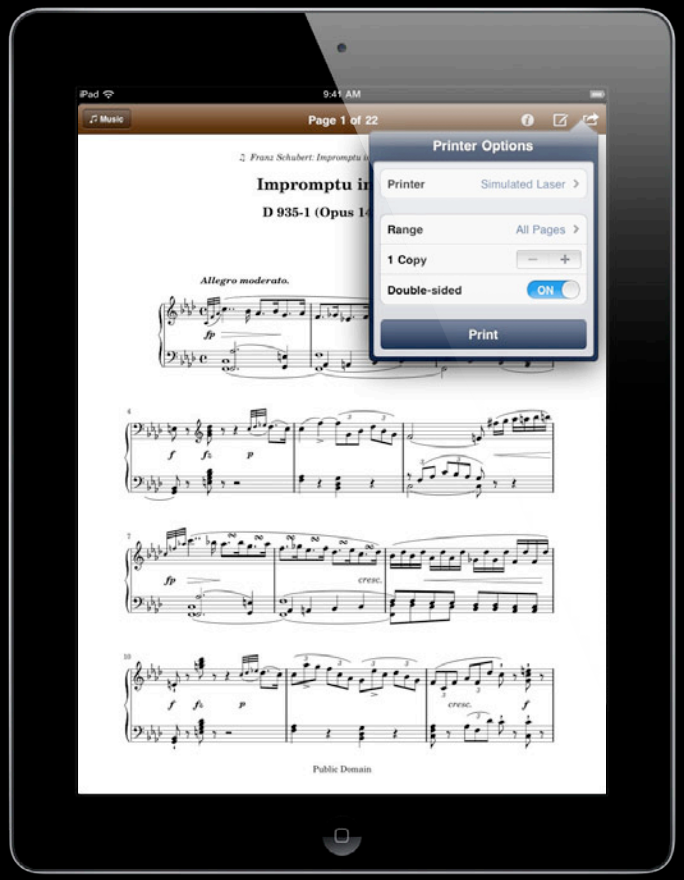
- Mixed text and graphics
- Normal quality
- Document paper size
- Duplex allowed
- Page range allowed

UIPrintInfoOutputGeneral

- Mixed text and graphics
- Normal quality
- Document paper size
- Duplex allowed
- Page range allowed

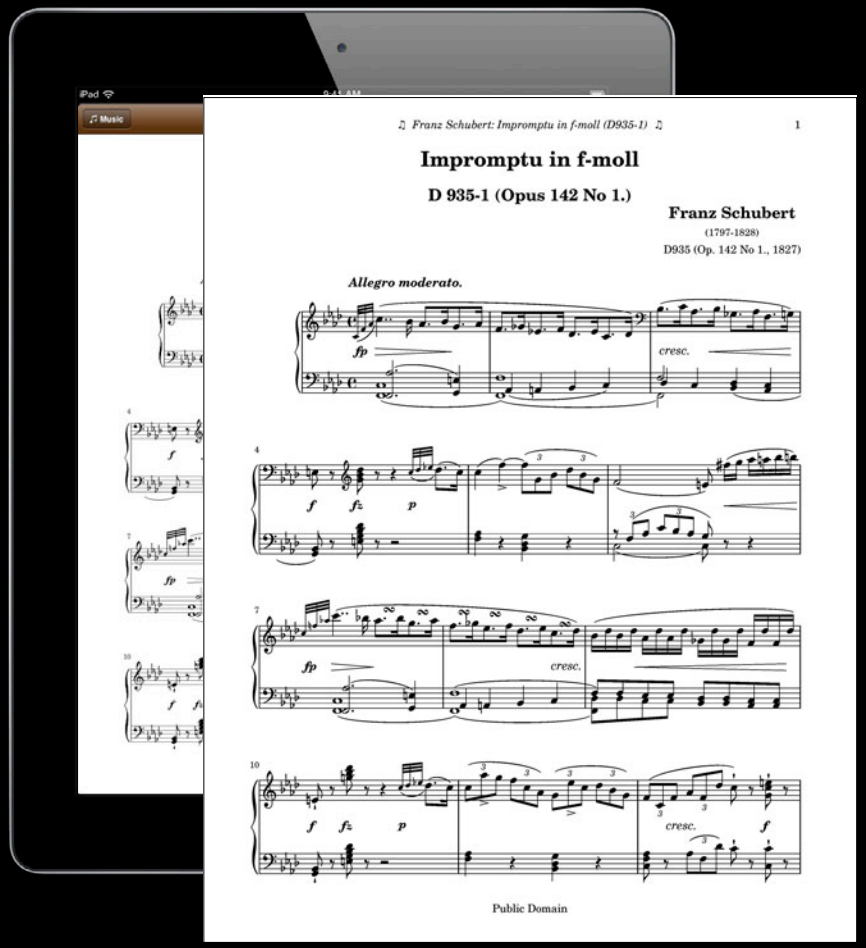
UIPrintInfoOutputGrayscale

- Monochrome text and graphics
- Improved print speed
- Reduced ink usage
- Document paper size
- Duplex allowed
- Page range allowed



UIPrintInfoOutputGrayscale

- Monochrome text and graphics
- Improved print speed
- Reduced ink usage
- Document paper size
- Duplex allowed
- Page range allowed



Smart Paper Size Selection

- Paper size option for printers that report loaded paper sizes

Smart Paper Size Selection

- Paper size option for printers that report loaded paper sizes

Setting the Job Name

Setting the Job Name

Setting the Job Name

Printing Items

- Single item or array of items
  - PDF, JPEG, other image types (PNG, etc.)

    `NSURL, NSData, UIImage, CIImage`
    - Asset library

    `ALAsset, ALAssetURL`
- Each item is a separate print job

Formatters

Formatters

- Use with `UIPrintInteractionController` or in a renderer
- For plain text use `UISimpleTextFormatter` and specify
  - Font
  - Color
  - Alignment
- For HTML markup text use `UIMarkupTextFormatter`

Formatters

```
- (void)printHTMLText:(NSString *)text {

    UIPrintInteractionController *
           controller = [UIPrintInteractionController sharedPrintController];

    UIMarkupTextFormatter *formatter
                      = [[UIMarkupTextFormatter alloc] initWithText:text];

    controller.printFormatter = formatter;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;
    printInfo.jobName    = [urlField webPage];
    controller.printInfo = printInfo;

    [controller presentAnimated:YES completionHandler:NULL];

}
```

Formatters

```
- (void)printHTMLText:(NSString *)text {

    UIPrintInteractionController *
           controller = [UIPrintInteractionController sharedPrintController];

    UIMarkupTextFormatter *formatter
                      = [[UIMarkupTextFormatter alloc] initWithText:text];

    controller.printFormatter = formatter;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;
    printInfo.jobName    = [urlField webPage];
    controller.printInfo = printInfo;

    [controller presentAnimated:YES completionHandler:NULL];

}
```

Formatters

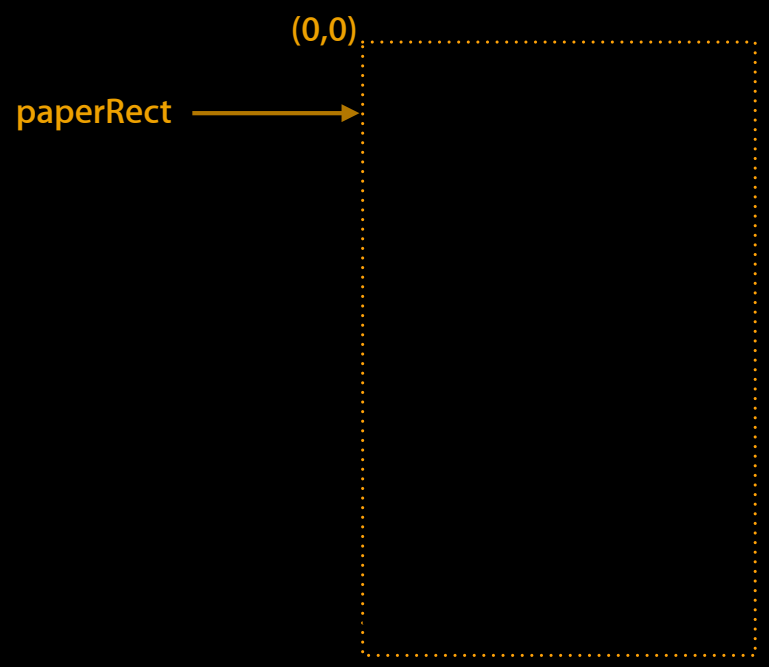
```
- (void)printHTMLText:(NSString *)text {

    UIPrintInteractionController *
           controller = [UIPrintInteractionController sharedPrintController];

    UIMarkupTextFormatter *formatter
                      = [[UIMarkupTextFormatter alloc] initWithText:text];

    controller.printFormatter = formatter;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;
    printInfo.jobName    = [urlField webPage];
    controller.printInfo = printInfo;

    [controller presentAnimated:YES completionHandler:NULL];

}
```

Formatters

```
- (void)printHTMLText:(NSString *)text {

    UIPrintInteractionController *
           controller = [UIPrintInteractionController sharedPrintController];

    UIMarkupTextFormatter *formatter
                      = [[UIMarkupTextFormatter alloc] initWithText:text];

    controller.printFormatter = formatter;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;
    printInfo.jobName    = [urlField webPage];
    controller.printInfo = printInfo;

    [controller presentAnimated:YES completionHandler:NULL];

}
```

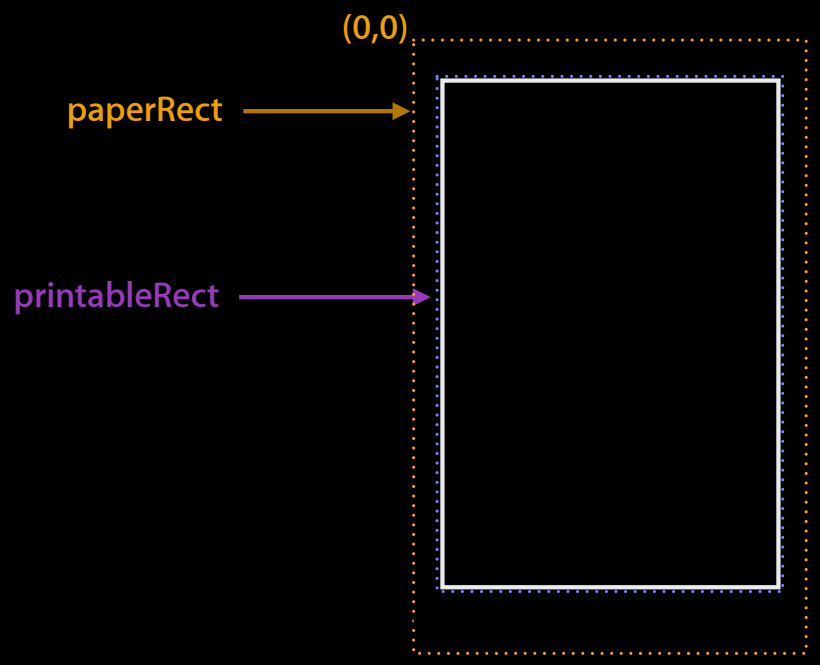
Layout

Content is drawn inside the printable rect



Layout

Content is drawn inside the printable rect

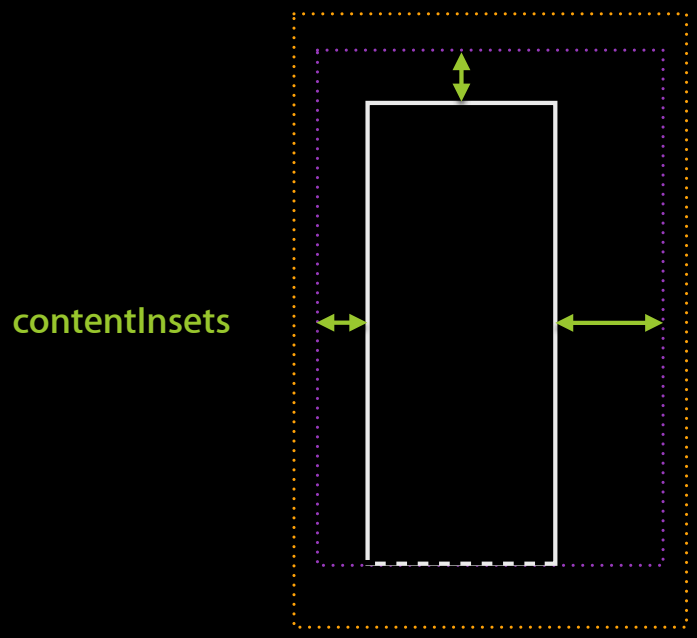


Layout

Content insets

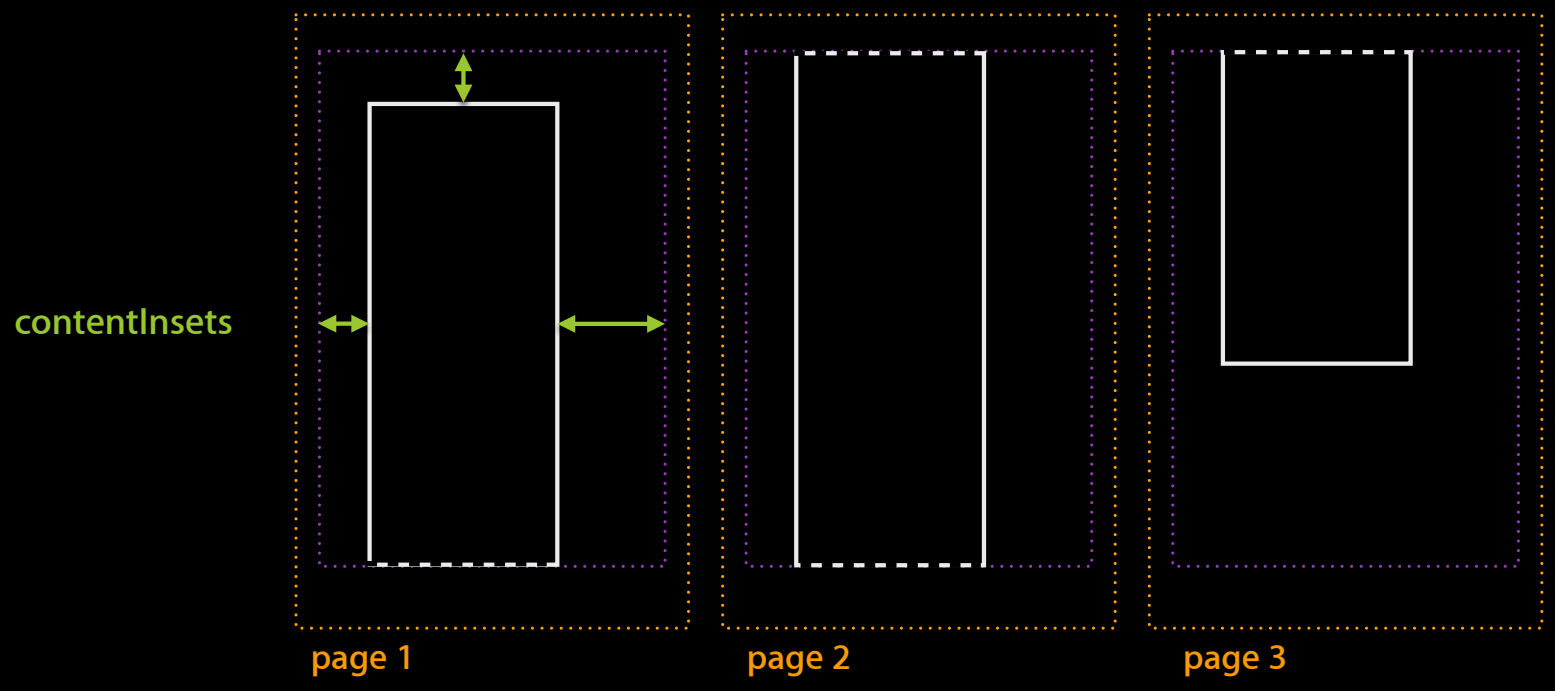
Layout

Content insets



Layout

Content insets



View Formatters

```
- (void)printWebView:(id) sender {

    UIPrintInteractionController *
           controller = [UIPrintInteractionController sharedPrintController];

    UIViewPrintFormatter *formatter
                      = [self.myWebView viewPrintFormatter];

    controller.printFormatter = formatter;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;
    printInfo.jobName    = [urlField myWebPage];
    controller.printInfo = printInfo;

    [controller presentAnimated:YES completionHandler:NULL];

}
```

View Formatters

```
- (void)printWebView:(id) sender {

    UIPrintInteractionController *
           controller = [UIPrintInteractionController sharedPrintController];

    UIViewPrintFormatter *formatter
                      = [self.myWebView viewPrintFormatter];

    controller.printFormatter = formatter;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;
    printInfo.jobName    = [urlField myWebPage];
    controller.printInfo = printInfo;

    [controller presentAnimated:YES completionHandler:NULL];

}
```

View Formatters

```
- (void)printWebView:(id) sender {

    UIPrintInteractionController *
           controller = [UIPrintInteractionController sharedPrintController];

    UIViewPrintFormatter *formatter
                      = [self.myWebView viewPrintFormatter];

    controller.printFormatter = formatter;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;
    printInfo.jobName    = [urlField myWebPage];
    controller.printInfo = printInfo;

    [controller presentAnimated:YES completionHandler:NULL];

}
```

Renderers

Print Page Renderer

- Full drawing control

Print Page Renderer

- Full drawing control
- Custom page-drawing object
  - Calculates page count
  - Draws page contents

Print Page Renderer

- Full drawing control
- Custom page-drawing object
  - Calculates page count
  - Draws page contents
- Add space for headers and footers

Print Page Renderer

- Full drawing control
- Custom page-drawing object
  - Calculates page count
  - Draws page contents
- Add space for headers and footers
- Add formatters

Basic Rendering

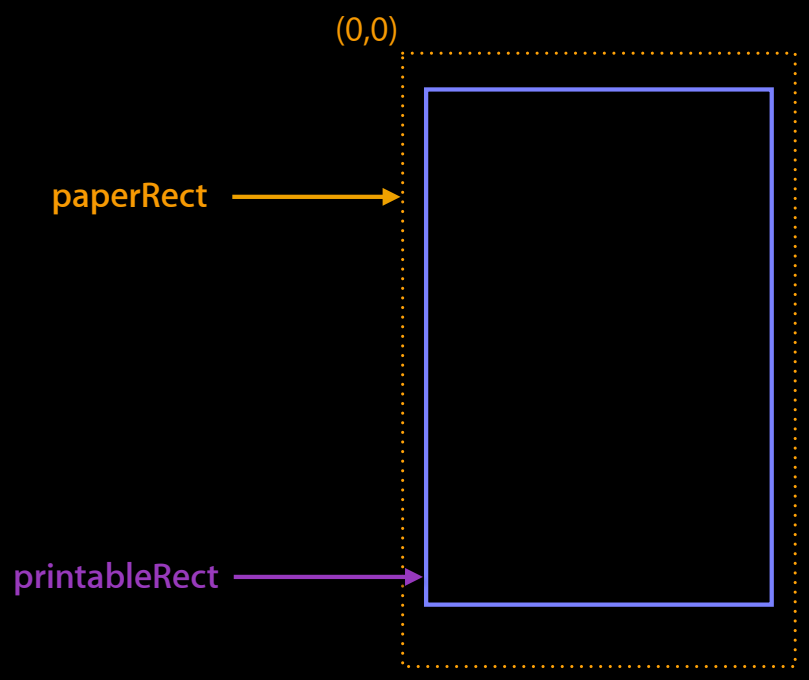
- Subclass `UIPrintPageRenderer`
- Override

  `-numberOfPages`
  `-drawContentForPageAtIndex:inRect:`

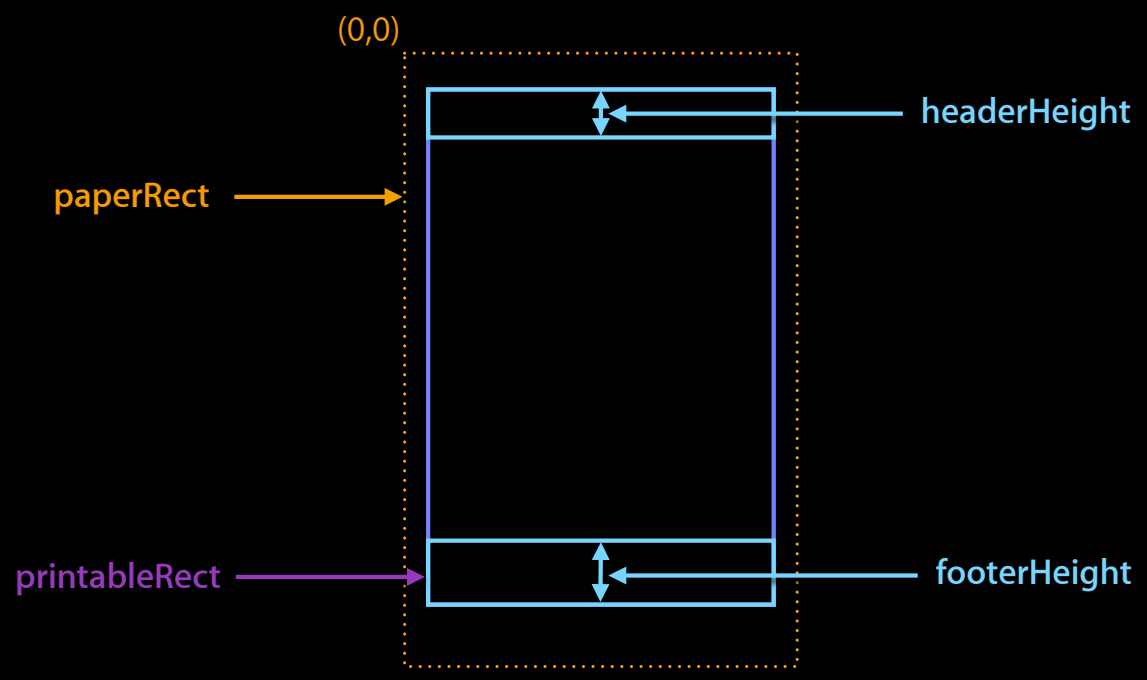
- Set `UIPrintInteractionController.printPageRenderer`

Renderer Layout

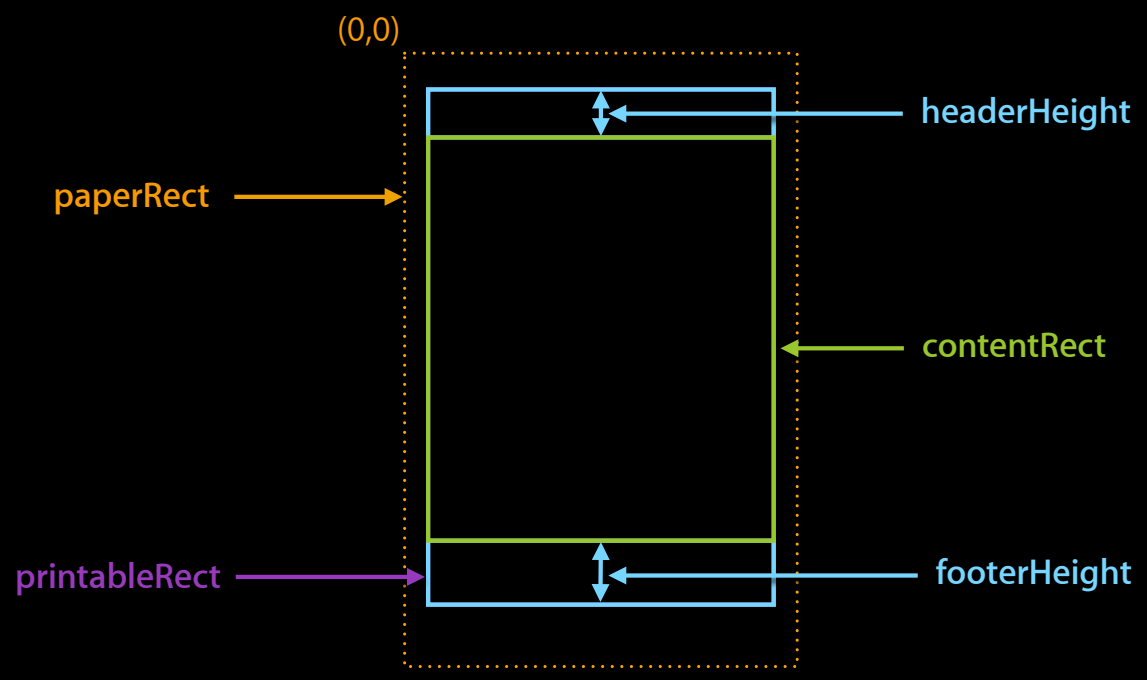
Renderer Layout



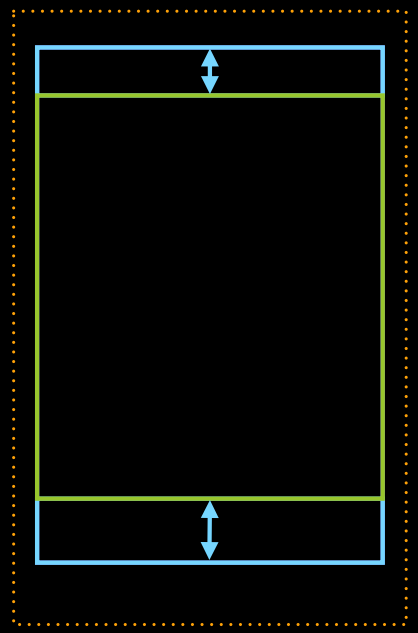
Renderer Layout



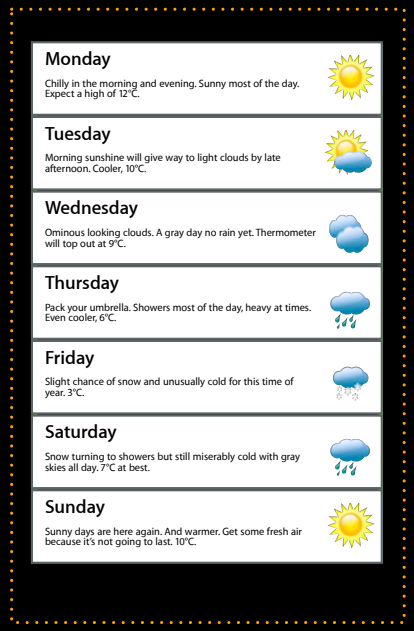
Renderer Layout



Renderer Layout



Renderer Layout



Basic Rendering

```
@interface ItemRenderer : UIPrintPageRenderer { }

// New

- (void)drawGraphicAtIndex:(NSInteger)index
                atOffset:(CGPoint)offset;
@property NSInteger graphicsPerPage;

// Overrides

- (NSInteger)numberOfPages;
- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                           inRect:(CGRect)contentRect
@end
```

Basic Rendering

```
@interface ItemRenderer : UIPrintPageRenderer { }

// New

- (void)drawGraphicAtIndex:(NSInteger)index
                atOffset:(CGPoint)offset;
@property NSInteger graphicsPerPage;

// Overrides

- (NSInteger)numberOfPages;
- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                           inRect:(CGRect)contentRect
@end
```

Basic Rendering

```
@interface ItemRenderer : UIPrintPageRenderer { }

// New

- (void)drawGraphicAtIndex:(NSInteger)index
                atOffset:(CGPoint)offset;
@property NSInteger graphicsPerPage;

// Overrides

- (NSInteger)numberOfPages;
- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                           inRect:(CGRect)contentRect
@end
```

Basic Rendering

```
- (NSInteger)numberOfPages {

   self.graphicsPerPage = floorf(self.printableRect.size.height /
IMAGE_HEIGHT);

   return ceilf(self.numberOfImages / self.graphicsPerPage);
}

- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                          inRect:(CGRect)contentRect {

   UIRectClip(contentRect);

   for (int i = 0; i < self.graphicsPerPage; i++) {
      [self drawGraphicAtIndex:pageIndex * self.graphicsPerPage + i
                    atOffset:CGPointMake(contentRect.origin.x,
                                         contentRect.origin.y+i*IMAGE_HEIGHT];
   }
}
```

Basic Rendering

```
- (NSInteger)numberOfPages {

   self.graphicsPerPage = floorf(self.printableRect.size.height /
IMAGE_HEIGHT);

   return ceilf(self.numberOfImages / self.graphicsPerPage);
}

- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                          inRect:(CGRect)contentRect {

   UIRectClip(contentRect);

   for (int i = 0; i < self.graphicsPerPage; i++) {
      [self drawGraphicAtIndex:pageIndex * self.graphicsPerPage + i
                    atOffset:CGPointMake(contentRect.origin.x,
                                         contentRect.origin.y+i*IMAGE_HEIGHT];
   }
}
```

Basic Rendering

```
- (NSInteger)numberOfPages {

   self.graphicsPerPage = floorf(self.printableRect.size.height /
IMAGE_HEIGHT);

   return ceilf(self.numberOfImages / self.graphicsPerPage);
}

- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                           inRect:(CGRect)contentRect {

   UIRectClip(contentRect);

   for (int i = 0; i < self.graphicsPerPage; i++) {
      [self drawGraphicAtIndex:pageIndex * self.graphicsPerPage + i
                    atOffset:CGPointMake(contentRect.origin.x,
                                         contentRect.origin.y+i*IMAGE_HEIGHT];
   }
}
```

Basic Rendering

```
- (NSInteger)numberOfPages {

   self.graphicsPerPage = floorf(self.printableRect.size.height /
IMAGE_HEIGHT);

   return ceilf(self.numberOfImages / self.graphicsPerPage);
}

- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                          inRect:(CGRect)contentRect {

   UIRectClip(contentRect);

   for (int i = 0; i < self.graphicsPerPage; i++) {
      [self drawGraphicAtIndex:pageIndex * self.graphicsPerPage + i
                    atOffset:CGPointMake(contentRect.origin.x,
                                         contentRect.origin.y+i*IMAGE_HEIGHT];
   }
}
```

Basic Rendering

```
- (NSInteger)numberOfPages {

   self.graphicsPerPage = floorf(self.printableRect.size.height /
IMAGE_HEIGHT);

   return ceilf(self.numberOfImages / self.graphicsPerPage);
}

- (void)drawContentForPageAtIndex:(NSInteger)pageIndex
                          inRect:(CGRect)contentRect {

   UIRectClip(contentRect);

   for (int i = 0; i < self.graphicsPerPage; i++) {
      [self drawGraphicAtIndex:pageIndex * self.graphicsPerPage + i
                    atOffset:CGPointMake(contentRect.origin.x,
                                         contentRect.origin.y+i*IMAGE_HEIGHT];
   }
}
```

Renderer Drawing Methods

```
-drawPageAtIndex:inRect:
    -drawHeaderForPageAtIndex:inRect:
    -drawContentForPageAtIndex:inRect:
    -drawPrintFormatter:forPageAtIndex:
    -drawFooterForPageAtIndex:inRect:
```

Renderer Drawing Methods

```
-drawPageAtIndex:inRect:
```

```
    -drawHeaderForPageAtIndex:inRect:

    -drawContentForPageAtIndex:inRect:

    -drawPrintFormatter:forPageAtIndex:

    -drawFooterForPageAtIndex:inRect:
```

Renderer Drawing Methods

```
-drawPageAtIndex:inRect:
    -drawHeaderForPageAtIndex:inRect:
    -drawContentForPageAtIndex:inRect:
    -drawPrintFormatter:forPageAtIndex:
    -drawFooterForPageAtIndex:inRect:
```

Renderer Drawing Methods

```
-drawPageAtIndex:inRect:
    -drawHeaderForPageAtIndex:inRect:
    -drawContentForPageAtIndex:inRect:
    -drawPrintFormatter:forPageAtIndex:
    -drawFooterForPageAtIndex:inRect:
```

Renderer Drawing Methods

```
-drawPageAtIndex:inRect:
    -drawHeaderForPageAtIndex:inRect:
    -drawContentForPageAtIndex:inRect:
    -drawPrintFormatter:forPageAtIndex:
    -drawFooterForPageAtIndex:inRect:
```

Renderer Drawing Methods

```
-drawPageAtIndex:inRect:
    -drawHeaderForPageAtIndex:inRect:
    -drawContentForPageAtIndex:inRect:
    -drawPrintFormatter:forPageAtIndex:
    -drawFooterForPageAtIndex:inRect:
```

UI Considerations

Printing From a Sheet or Bar Button Item

- iPhone

  `-presentAnimated:completionHandler:`

- iPad

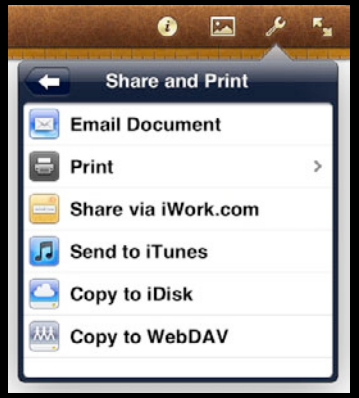
  `-presentFromRect:inView:animated:completionHandler:`
  `-presentFromBarButtonItem:animated:completionHandler:`

Printing as a Menu Item

- Implement—`printInteractionControllerParentViewController:`
  - `UINavigationController`—push
  - `UIViewController`—modal
- Do not peek

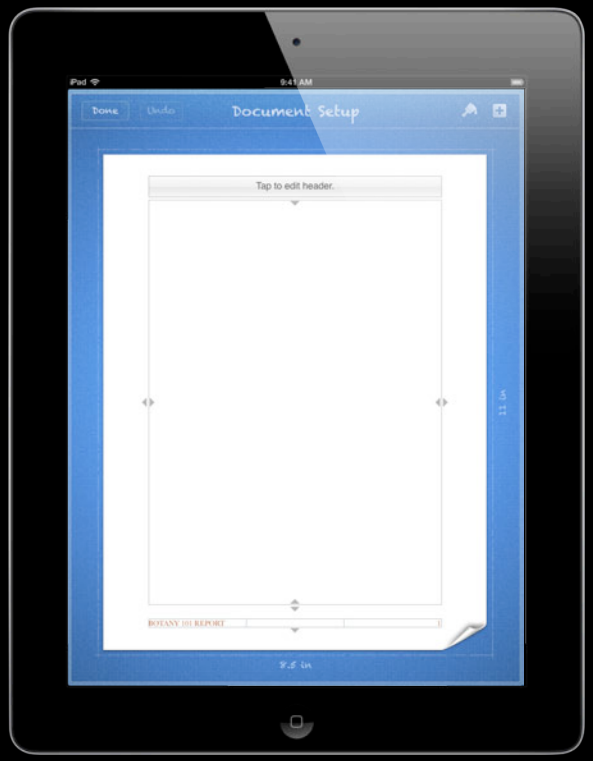
Printing as a Menu Item

- Implement—`printInteractionControllerParentViewController:`
  - `UINavigationController`—push
  - `UIViewController`—modal
- Do not peek



Paper Size

- For document-centric apps that require a paper size in order to format the content



Paper Size

- Provide your own paper selection UI

Paper Size

- Provide your own paper selection UI
- Use delegate method
  - `-printInteractionController:choosePaper:`
  - Called after user selects a printer
  - You ask for a paper size that is a good match to the user selected paper

Paper Size

```
- (UIPrintPaper *)printInteractionController:(UIPrintInteractionController *)
                printInteractionController choosePaper:(NSArray *)paperList {

    CGSize paperSize = CGSizeMake(8.5 * 72.0, 11.0 * 72.0);

    return [UIPrintPaper bestPaperForPageSize:pageSize
                          withPapersFromArray:paperList];
}
```

Paper Size

```
- (UIPrintPaper *)printInteractionController:(UIPrintInteractionController *)
                printInteractionController choosePaper:(NSArray *)paperList {

    CGSize paperSize = CGSizeMake(8.5 * 72.0, 11.0 * 72.0);

    return [UIPrintPaper bestPaperForPageSize:pageSize
                          withPapersFromArray:paperList];
}
```

Paper Size

```
- (UIPrintPaper *)printInteractionController:(UIPrintInteractionController *)
                printInteractionController choosePaper:(NSArray *)paperList {

    CGSize paperSize = CGSizeMake(8.5 * 72.0, 11.0 * 72.0);

    return [UIPrintPaper bestPaperForPageSize:pageSize
                         withPapersFromArray:paperList];
}
```

Printing from the Activity Sheet

Printing from the Activity Sheet

```
NSArray *activityItems = [NSArray arrayWithObjects: printInfo, myRenderer,
[urlField text], nil ];

UIActivityViewController *viewController = [[UIActivityViewController alloc]
initWithActivityItems:activityItems applicationActivities:nil];

viewController.completionHandler = ^(NSString * activityType, BOOL completed)
{
   [viewController release];
   [myRenderer release];
 };

... present using standard view controller present methods
```

Printing from the Activity Sheet

```
NSArray *activityItems = [NSArray arrayWithObjects: printInfo, myRenderer,
[urlField text], nil ];

UIActivityViewController *viewController = [[UIActivityViewController alloc]
initWithActivityItems:activityItems applicationActivities:nil];

viewController.completionHandler = ^(NSString * activityType, BOOL completed)
{
   [viewController release];
   [myRenderer release];
 };

... present using standard view controller present methods
```

Printing from the Activity Sheet

```
NSArray *activityItems = [NSArray arrayWithObjects: printInfo, myRenderer,
[urlField text], nil ];

UIActivityViewController *viewController = [[UIActivityViewController alloc]
initWithActivityItems:activityItems applicationActivities:nil];

viewController.completionHandler = ^(NSString * activityType, BOOL completed)
{
   [viewController release];
   [myRenderer release];
 };

... present using standard view controller present methods
```

Printing from the Activity Sheet

```
NSArray *activityItems = [NSArray arrayWithObjects: printInfo, myRenderer,
[urlField text], nil ];

UIActivityViewController *viewController = [[UIActivityViewController alloc]
initWithActivityItems:activityItems applicationActivities:nil];

viewController.completionHandler = ^(NSString * activityType, BOOL completed)
{
   [viewController release];
   [myRenderer release];
 };

... present using standard view controller present methods
```

Printer Simulator

Todd Ritland
Printing Engineer

How do I start?

What tools are available
to make my job easier?

Printer Simulator

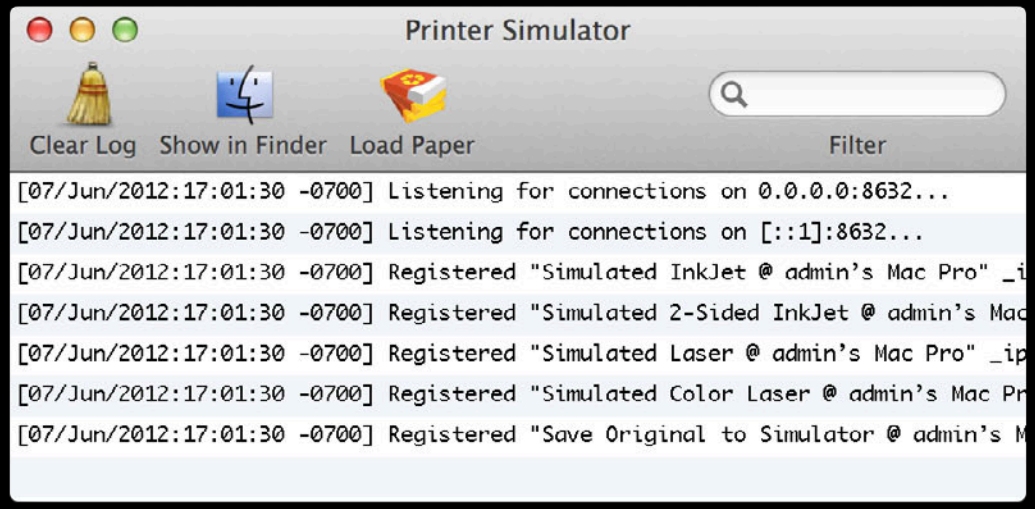
- Creates virtual printers on your Mac
- Avoid wasting paper while developing your printing code
- Outputs PDF
- We use it as our reference implementation of AirPrint

Demo

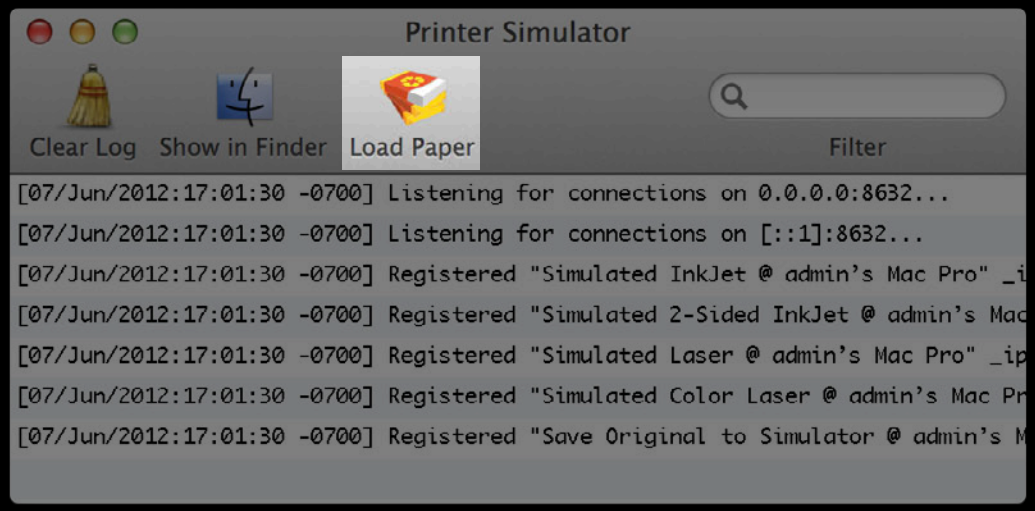
Using Printer Simulator for Testing

- Use all 4 simulated printers
- Test with double-sided option on and off
- Any content in the yellow area of the page will be clipped by a printer

New in iOS 6 SDK



New in iOS 6 SDK



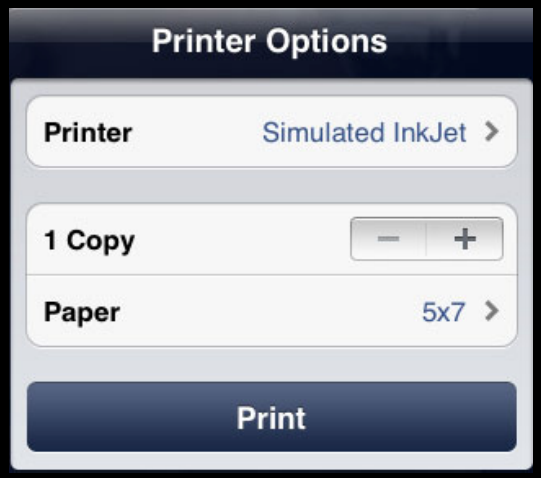
New in iOS 6 SDK

New in iOS 6 SDK

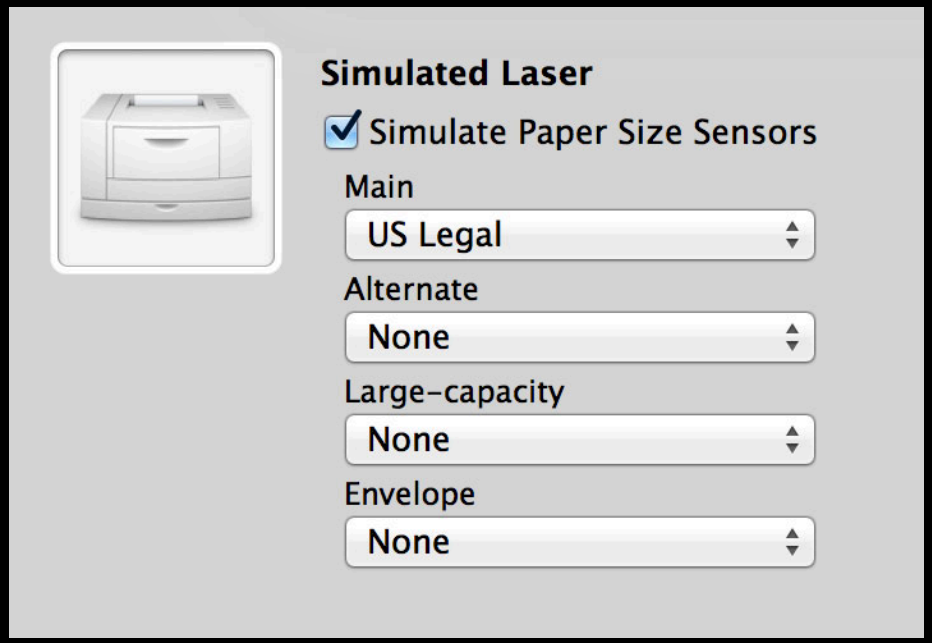
Load Paper?

Paper Size Sensors

- Some printers sense what size is loaded
- If your app prints items, the OS automatically sizes the PDF or photo
- If your app uses formatters or renderers, the `paperRect` and `printableRect` represent the size loaded at the printer



Simulate Paper Sensing



Demo

Summary

- AirPrint overview
- OS X printing
- iOS printing
- Printer Simulator

More Information

Paul Danbold
Core OS Evangelist
danbold@apple.com

iOS Printing Documentation
http://developer.apple.com/search/index.php?q=printing

iOS Printing Sample Code
http://developer.apple.com/library/ios/#samplecode/PrintWebView
http://developer.apple.com/library/ios/#samplecode/PrintPhoto
http://developer.apple.com/library/ios/#samplecode/Recipes_+_Printing

More Information

Apple Developer Forums
http://devforums.apple.com

AirPrint 101
http://support.apple.com/kb/ht4356

Related Sessions

What's New in Cocoa Touch	Mission Tuesday 9:00AM

Labs

Printing Lab	Core OS Lab A Thursday 2:00PM

WWDC2012