CS193P - Lecture 8

iPhone Application Development

Scroll Views & Table Views


iPad
9:41 AM
27
Calendar
Contacts
Notes
Maps
Videos
YouTube
iTunes
App Store
Settings
Safari
Mail
Photos
iPod
iPad

iPhone SDK 3.2

iPhone SDK 3.2

- Support for iPad

iPhone SDK 3.2

- Support for iPad
- Beta Release

iPhone SDK 3.2

- Support for iPad
- Beta Release

Enrollment in iPhone Developer Standard or Enterprise required

Announcements

- Paparazzi 1 due next Wednesday (2/3)

Today's Topics

- Scroll views
- Table views
  - Displaying data
  - Controlling appearance & behavior
- UITableViewController
- Table view cells

Scroll Views

UIScrollView

- For displaying more content than can fit on the screen

UIScrollView

- For displaying more content than can fit on the screen
- Handles gestures for panning and zooming

UIScrollView

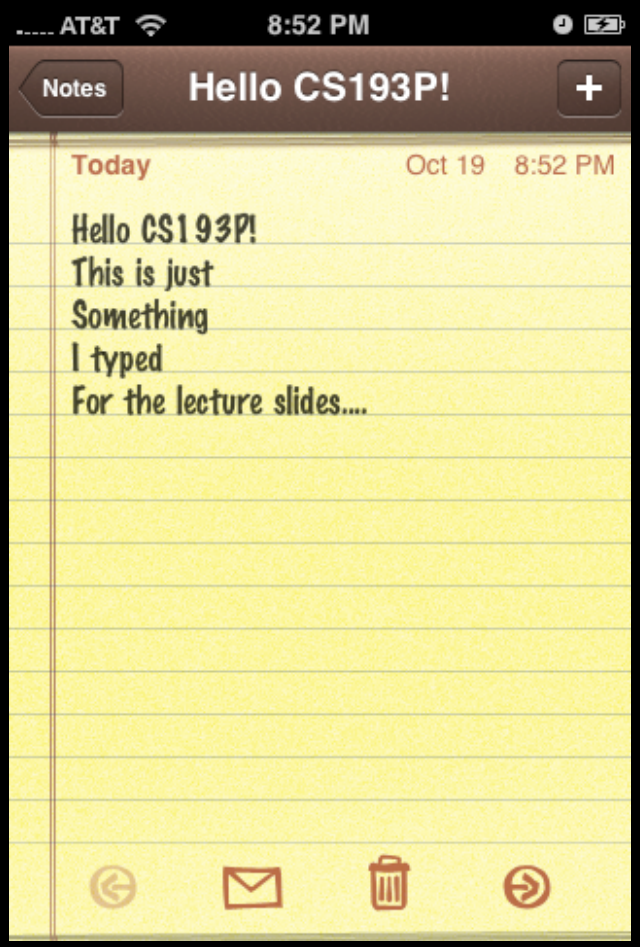
- For displaying more content than can fit on the screen
- Handles gestures for panning and zooming
- Noteworthy subclasses: UITableView and UITextView

Scrolling Examples

Scrolling Examples

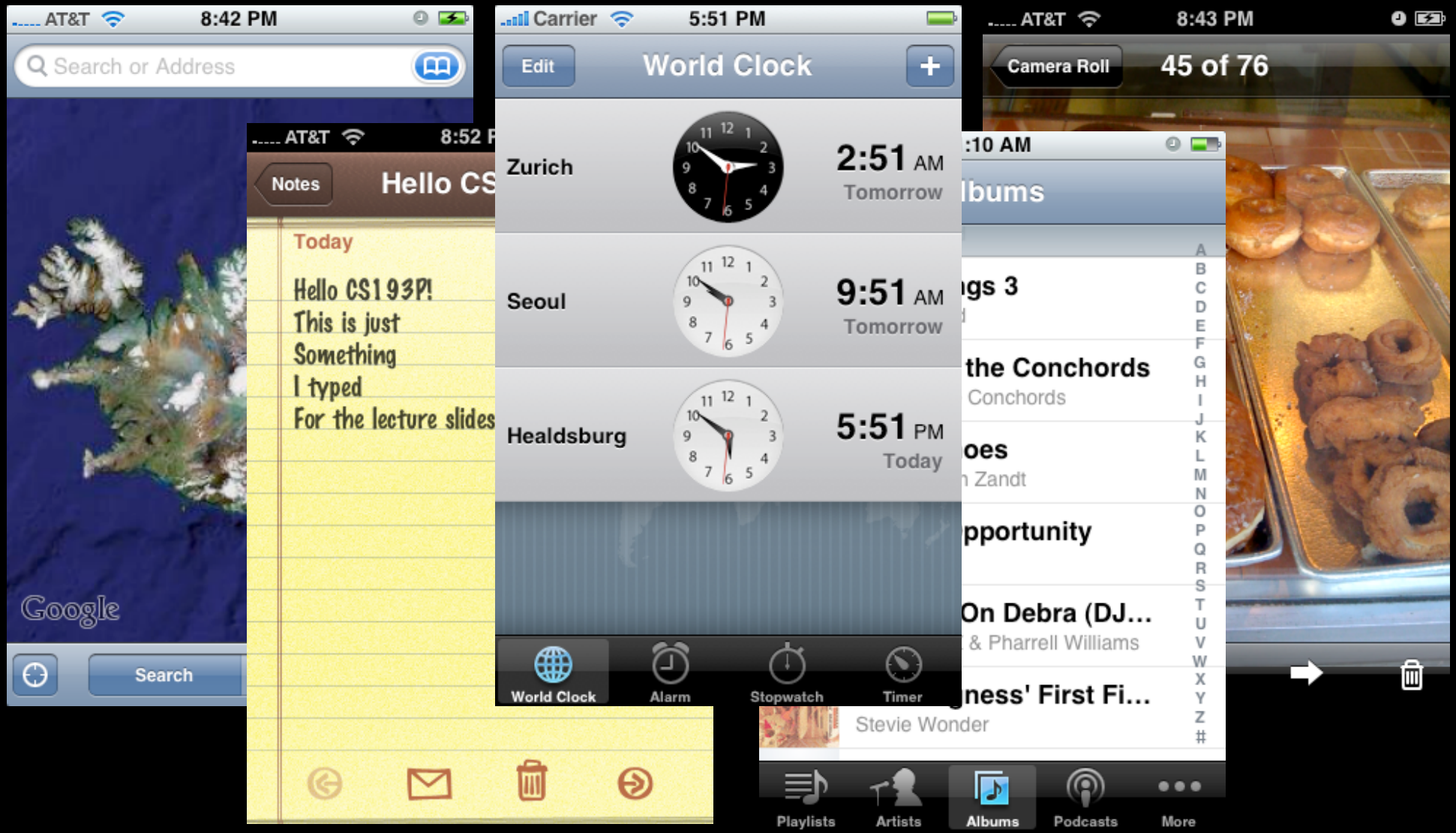
Scrolling Examples

Scrolling Examples

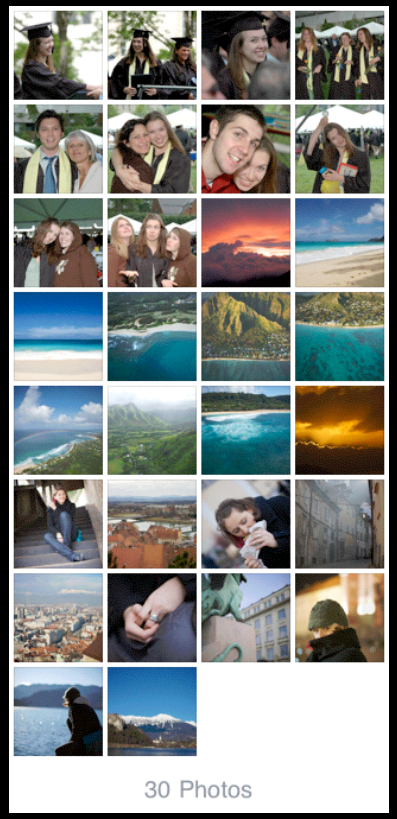


Scrolling Examples

Scrolling Examples



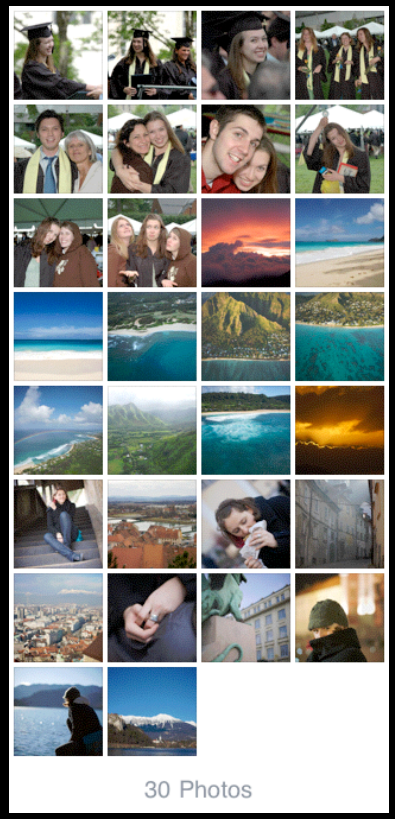
Content Size



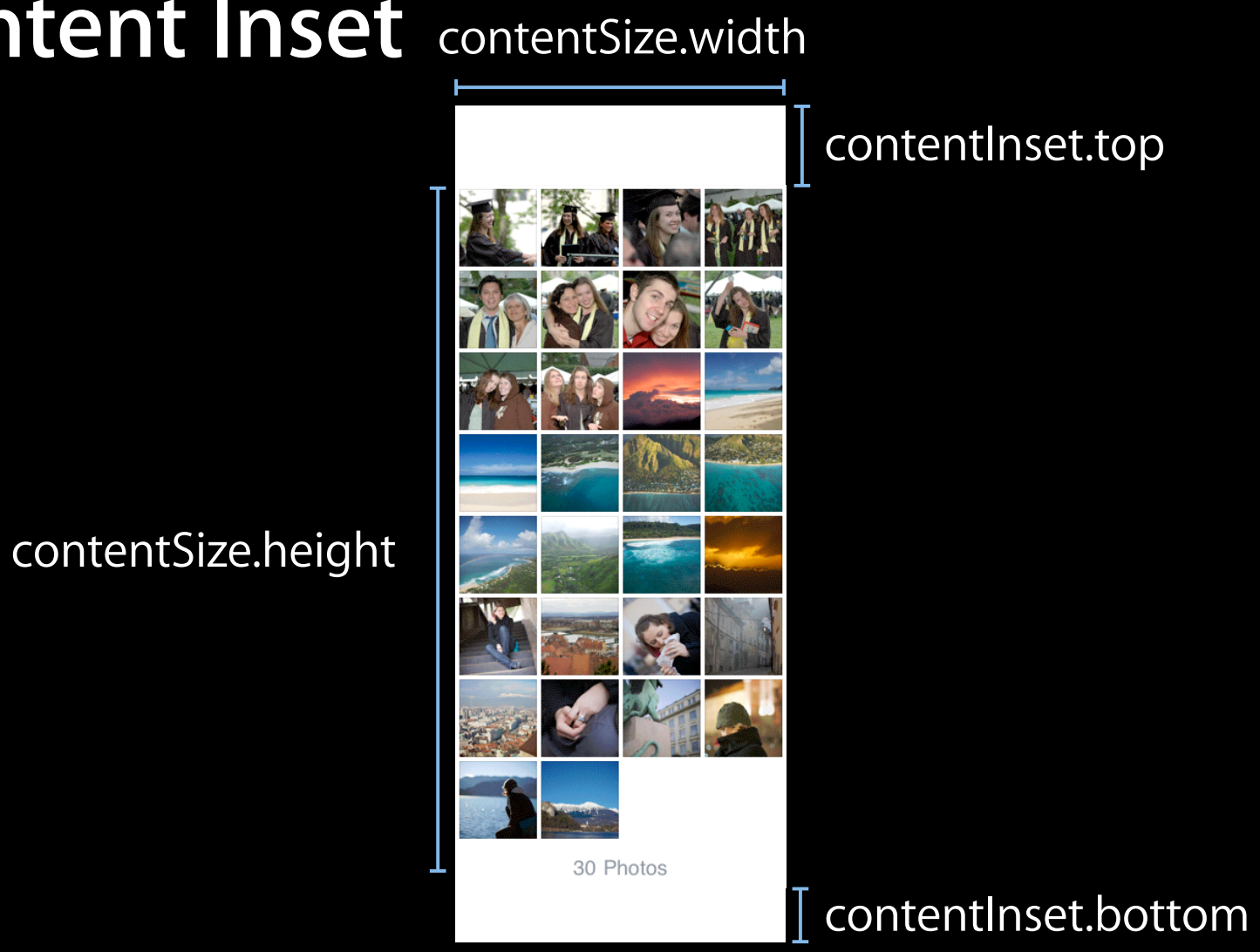
Content Size

contentSize.width

contentSize.height



Content Inset



Content Inset

Content Inset

Content Inset

Content Inset

contentInset.top

Content Inset

contentInset.top

Content Inset

contentInset.top

contentInset.bottom

Scroll Indicator Insets

Scroll Indicator Insets

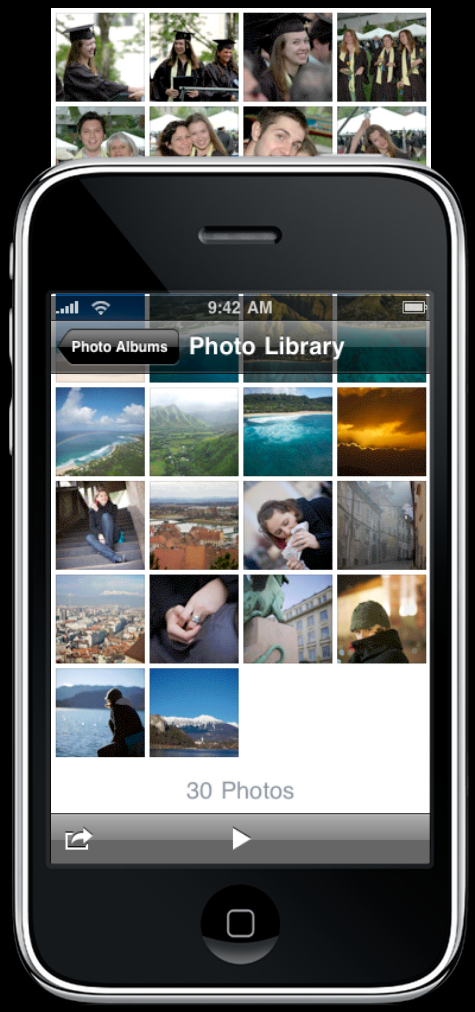
Scroll Indicator Insets

Scroll Indicator Insets

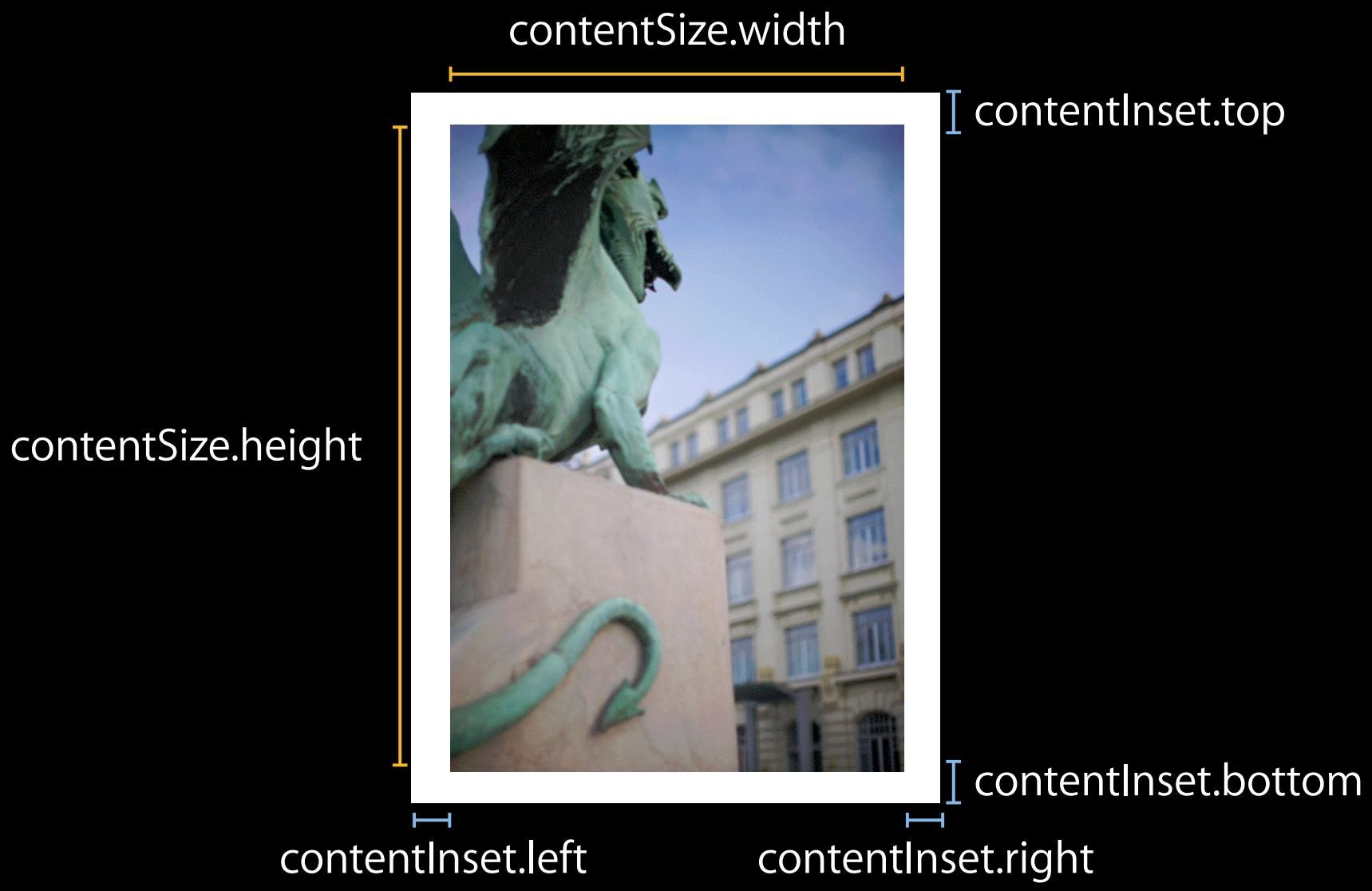
scrollIndicatorInsets.top

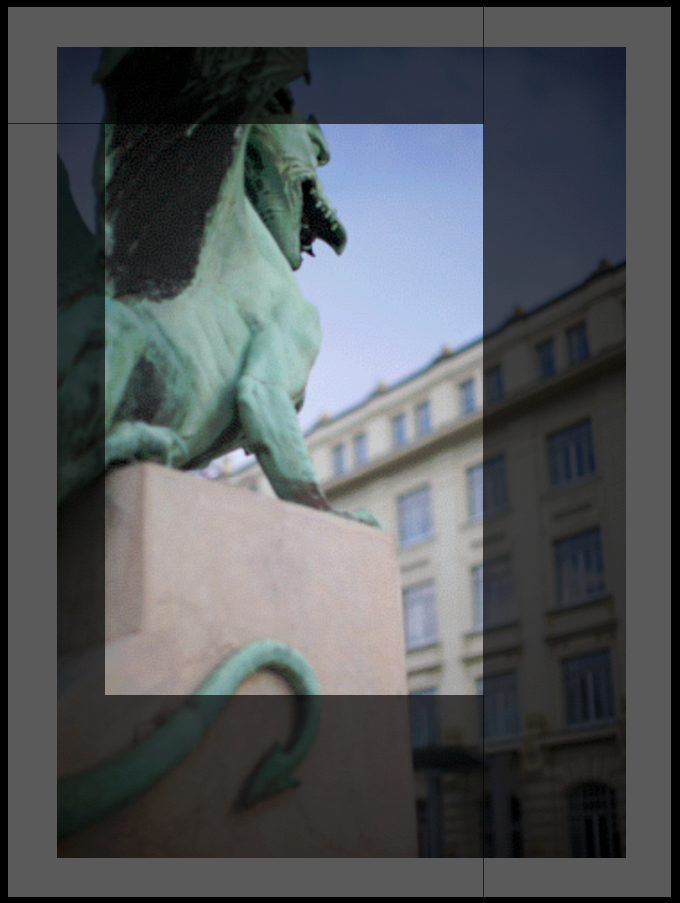
Content Offset

Content Offset

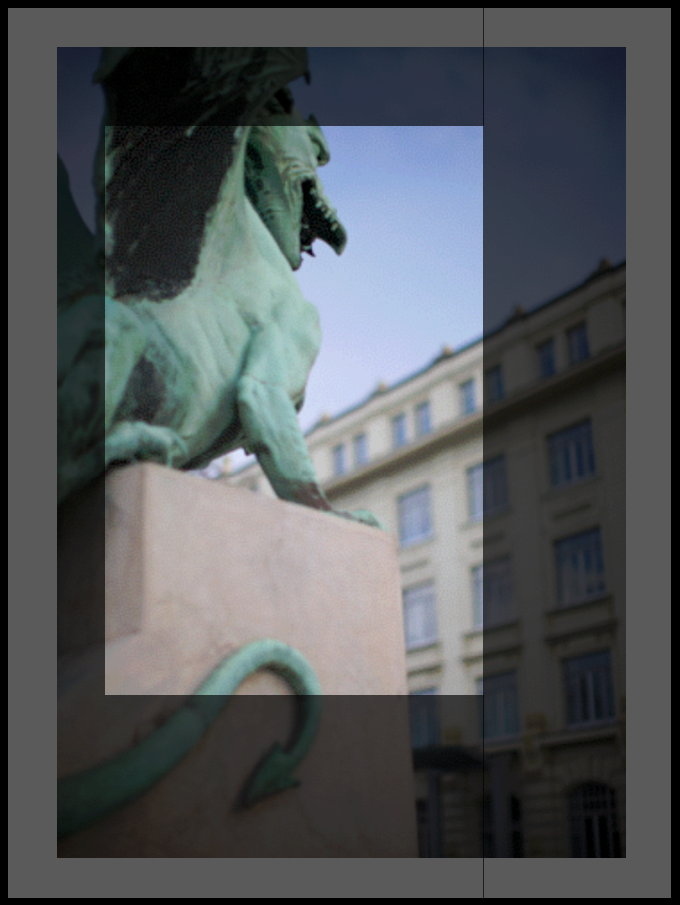


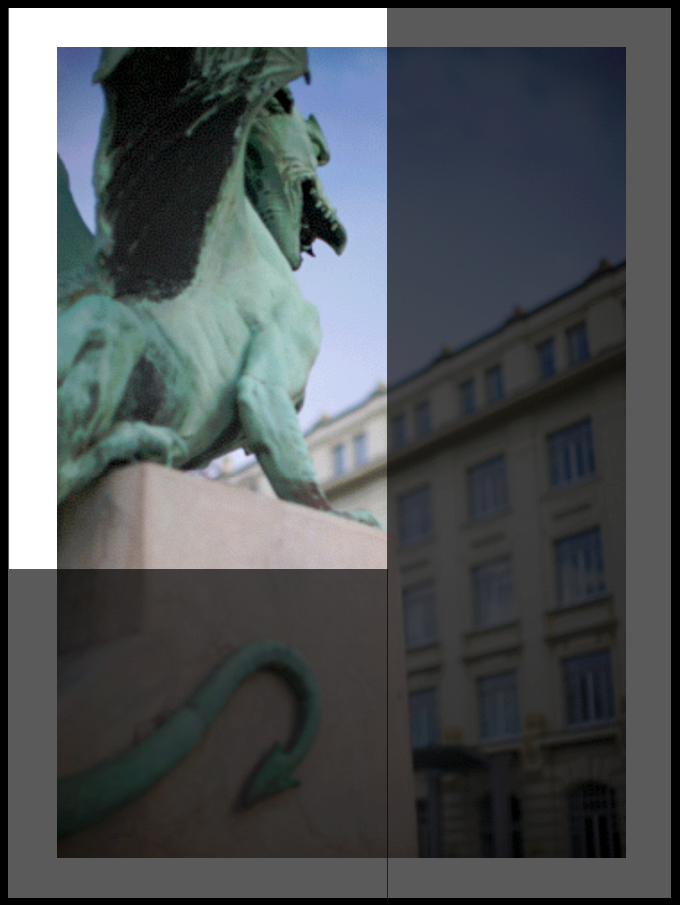
Content Offset

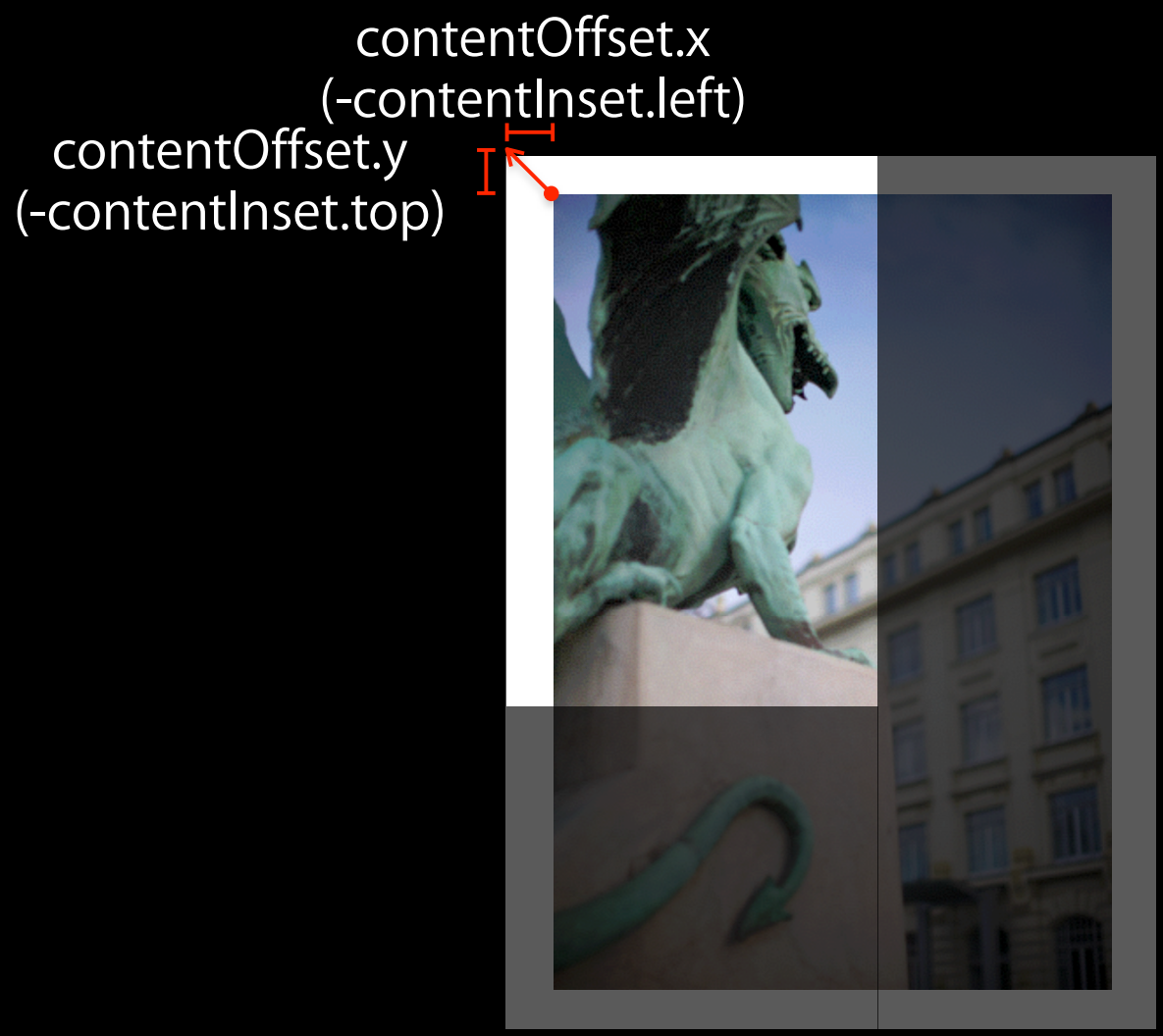

contentSize.width
contentInset.top
contentSize.height
contentInset.bottom
contentInset.left
contentInset.right




contentOffset.x
contentOffset.y






contentOffset.x
(-contentInset.left)
contentOffset.y
(-contentInset.top)

Using a Scroll View

Using a Scroll View

- Create with the desired frame

```
CGRect frame = CGRectMake(0, 0, 200, 200);
scrollView = [[UIScrollView alloc] initWithFrame:frame];
```

Using a Scroll View

- Create with the desired frame

```
CGRect frame = CGRectMake(0, 0, 200, 200);
scrollView = [[UIScrollView alloc] initWithFrame:frame];
```

- Add subviews (frames may extend beyond scroll view bounds)

```
frame = CGRectMake(0, 0, 500, 500);
myImageView = [[UIImageView alloc] initWithFrame:frame];
[scrollView addSubview:myImageView];
```

Using a Scroll View

- Create with the desired frame

```
CGRect frame = CGRectMake(0, 0, 200, 200);
scrollView = [[UIScrollView alloc] initWithFrame:frame];
```

- Add subviews (frames may extend beyond scroll view bounds)

```
frame = CGRectMake(0, 0, 500, 500);
myImageView = [[UIImageView alloc] initWithFrame:frame];
[scrollView addSubview:myImageView];
```

- Set the content size

```
scrollView.contentSize = CGSizeMake(500, 500);
```

Extending Scroll View Behavior

- Applications often want to know about scroll events

Extending Scroll View Behavior

- Applications often want to know about scroll events
  - When the scroll offset is changed

Extending Scroll View Behavior

- Applications often want to know about scroll events
  - When the scroll offset is changed
  - When dragging begins & ends

Extending Scroll View Behavior

- Applications often want to know about scroll events
  - When the scroll offset is changed
  - When dragging begins & ends
  - When deceleration begins & ends

Extending with a Subclass

- Create a subclass
- Override methods to customize behavior

Extending with a Subclass

- Create a subclass
- Override methods to customize behavior
- **Issues with this approach**

Extending with a Subclass

- Create a subclass
- Override methods to customize behavior
- **Issues with this approach**
  - Application logic and behavior is now part of a View class

Extending with a Subclass

- Create a subclass
- Override methods to customize behavior
- **Issues with this approach**
  - Application logic and behavior is now part of a View class
  - Tedious to write a one-off subclass for every scroll view instance

Extending with a Subclass

- Create a subclass
- Override methods to customize behavior
- **Issues with this approach**
  - Application logic and behavior is now part of a View class
  - Tedious to write a one-off subclass for every scroll view instance
  - Your code becomes **tightly coupled** with superclass

Extending with Delegation

- Delegate is a separate object
- Clearly defined points of responsibility
  - Change behavior
  - Customize appearance
- **Loosely coupled** with the object being extended

UIScrollView Delegate

UIScrollView Delegate

```
@protocol UIScrollViewDelegate<NSObject>
```

UIScrollView Delegate

```
@protocol UIScrollViewDelegate<NSObject>

@optional
```

UIScrollView Delegate

```
@protocol UIScrollViewDelegate<NSObject>

@optional

// Respond to interesting events
- (void)scrollViewDidScroll:(UIScrollView *)scrollView;
```

UIScrollView Delegate

```
@protocol UIScrollViewDelegate<NSObject>

@optional

// Respond to interesting events
- (void)scrollViewDidScroll:(UIScrollView *)scrollView;

...

// Influence behavior
- (BOOL)scrollViewShouldScrollToTop:(UIScrollView *)scrollView;

@end
```

Implementing a Delegate

Implementing a Delegate

- Conform to the delegate protocol

```
@interface MyController : NSObject <UIScrollViewDelegate>
```

Implementing a Delegate

- Conform to the delegate protocol

```
@interface MyController : NSObject <UIScrollViewDelegate>
```

- Implement **all required methods** and **any optional methods**

```
- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
  // Do something in response to the new scroll position
  if (scrollView.contentOffset ...) {

  }
}
```

Zooming with a Scroll View

Zooming with a Scroll View

- Set the minimum, maximum, initial zoom scales

```
scrollView.maximumZoomScale = 2.0;
scrollView.minimumZoomScale = scrollView.size.width /
                              myImage.size.width;
```

Zooming with a Scroll View

- Set the minimum, maximum, initial zoom scales

```
scrollView.maximumZoomScale = 2.0;
scrollView.minimumZoomScale = scrollView.size.width /
                              myImage.size.width;
```

- Implement delegate method for zooming

```
- (UIView *)viewForZoomingInScrollView:(UIView *)view
{
  return someViewThatWillBeScaled;
}
```

Set Zoom Scale

```
- (void)setZoomScale:(float)scale animated:(BOOL);
```

Set Zoom Scale

```
- (void)setZoomScale:(float)scale animated:(BOOL);
```

Set Zoom Scale

```
- (void)setZoomScale:(float)scale animated:(BOOL);
```

Zoom to Rect

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL);
```

Zoom to Rect

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL);
```

Zoom to Rect

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL);
```

Zoom to Rect

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL);
```

Zoom to Rect

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL);
```

Zoom to Rect

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL);
```

Demo

Table Views

Table Views

- Display lists of content
  - Single column, multiple rows
  - Vertical scrolling
  - Large data sets
- Powerful and ubiquitous in iPhone applications

Table View Styles

Table View Styles

UITableViewStylePlain

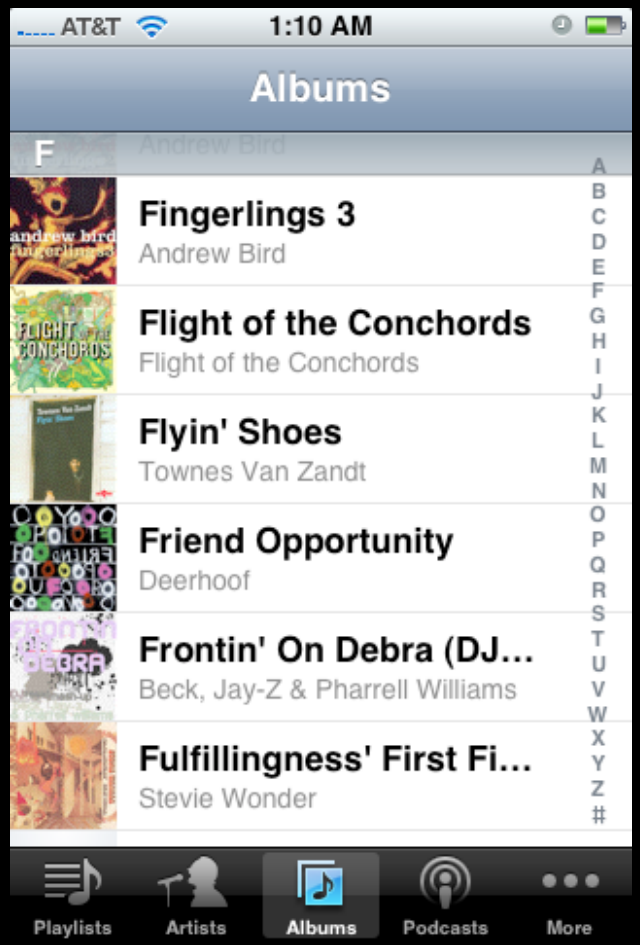
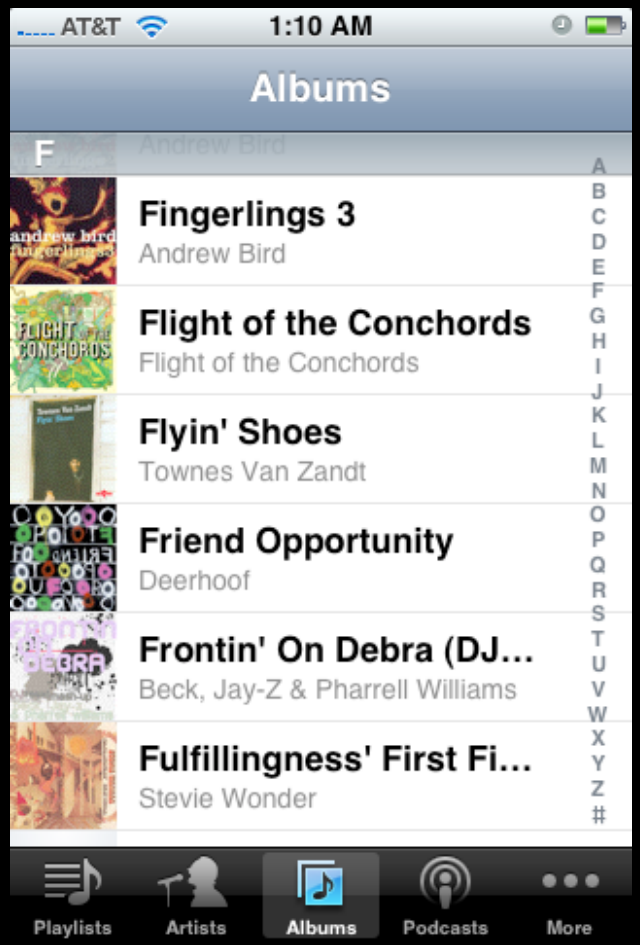


Table View Styles

UITableViewStylePlain



UITableViewStyleGrouped

Table View Anatomy

Plain Style

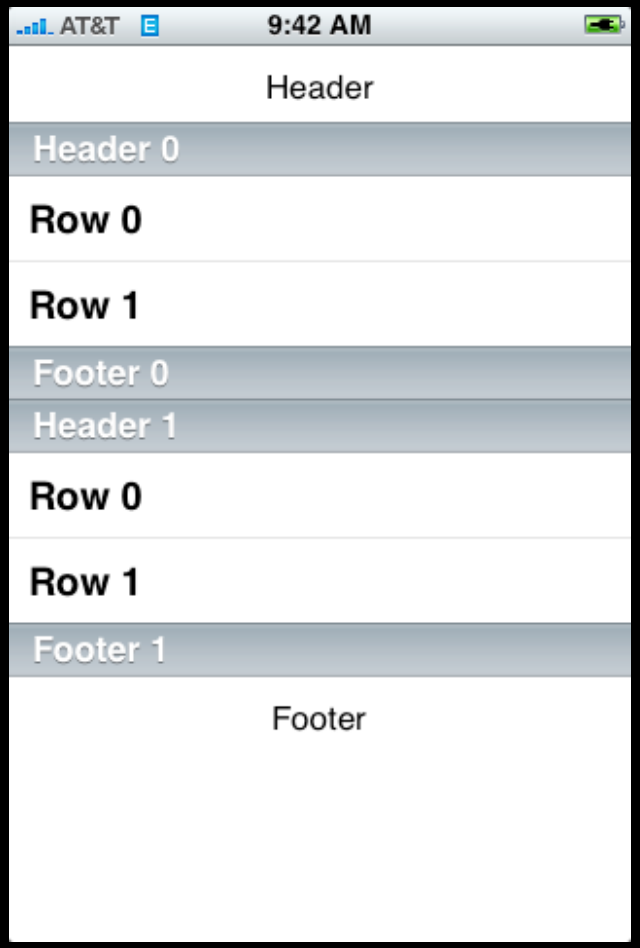


Table View Anatomy

Plain Style

Table Header

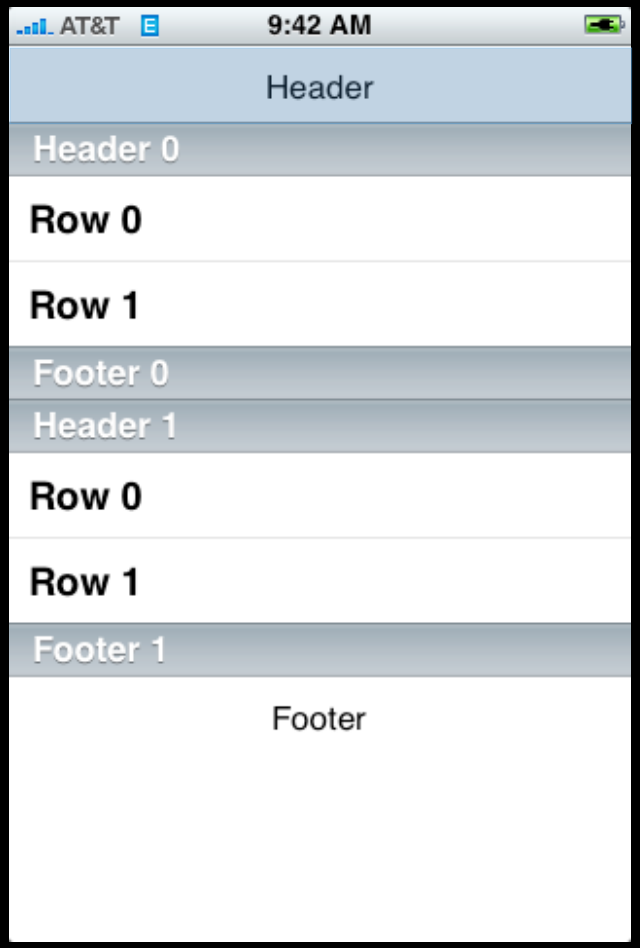


Table View Anatomy

Plain Style

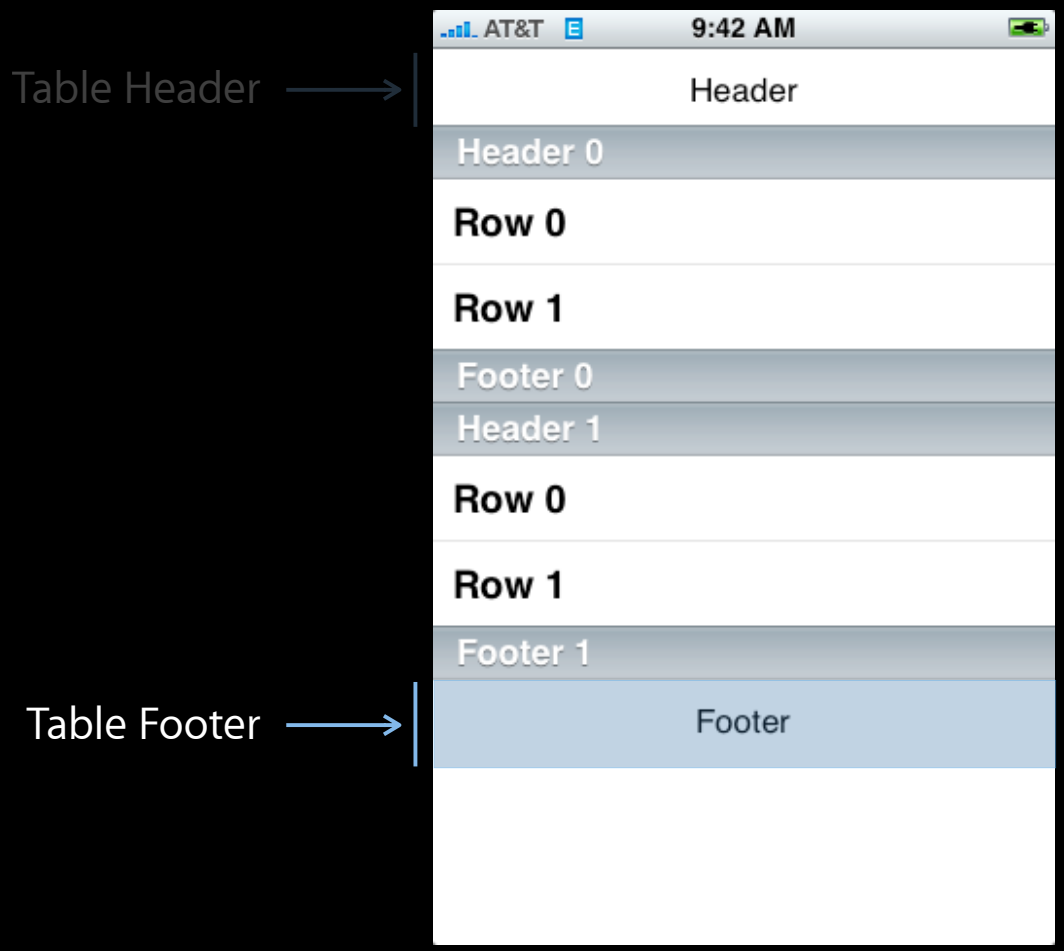


Table View Anatomy

Plain Style

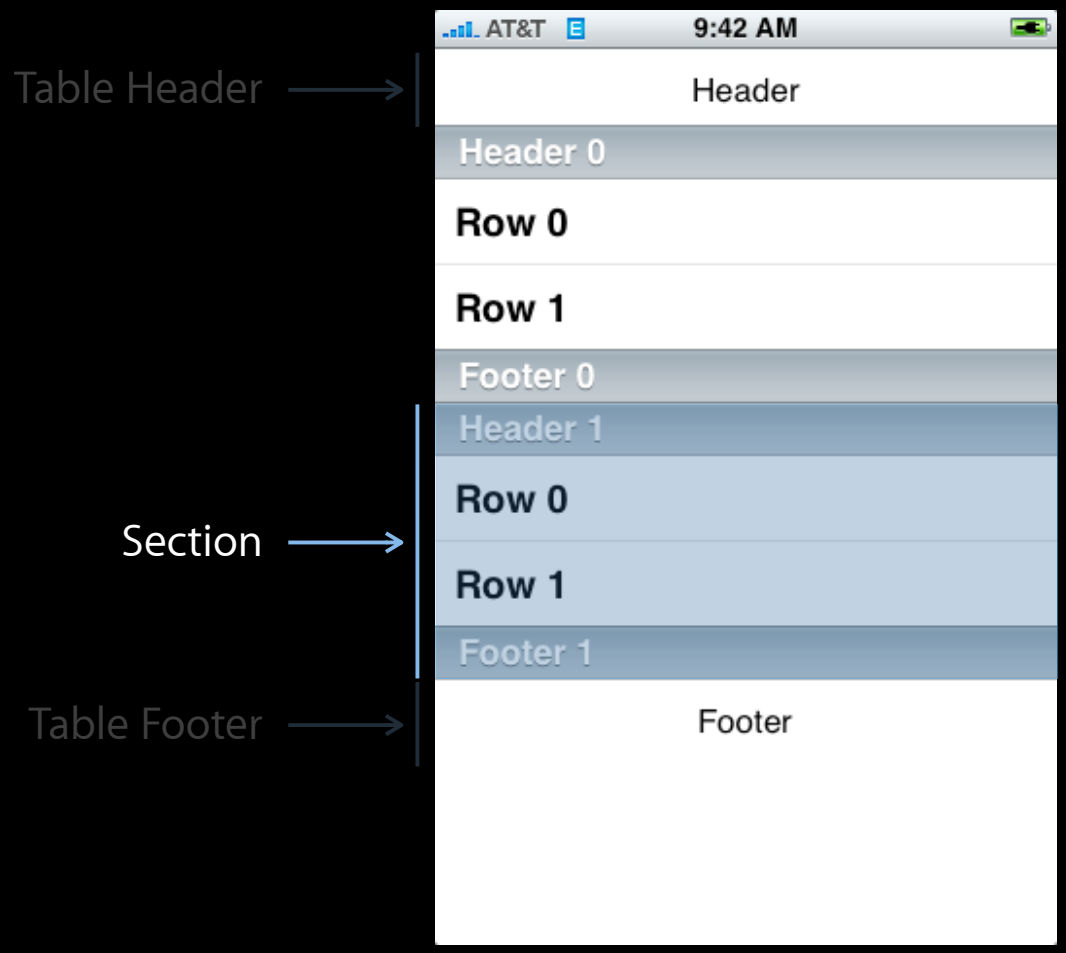


Table View Anatomy

Plain Style

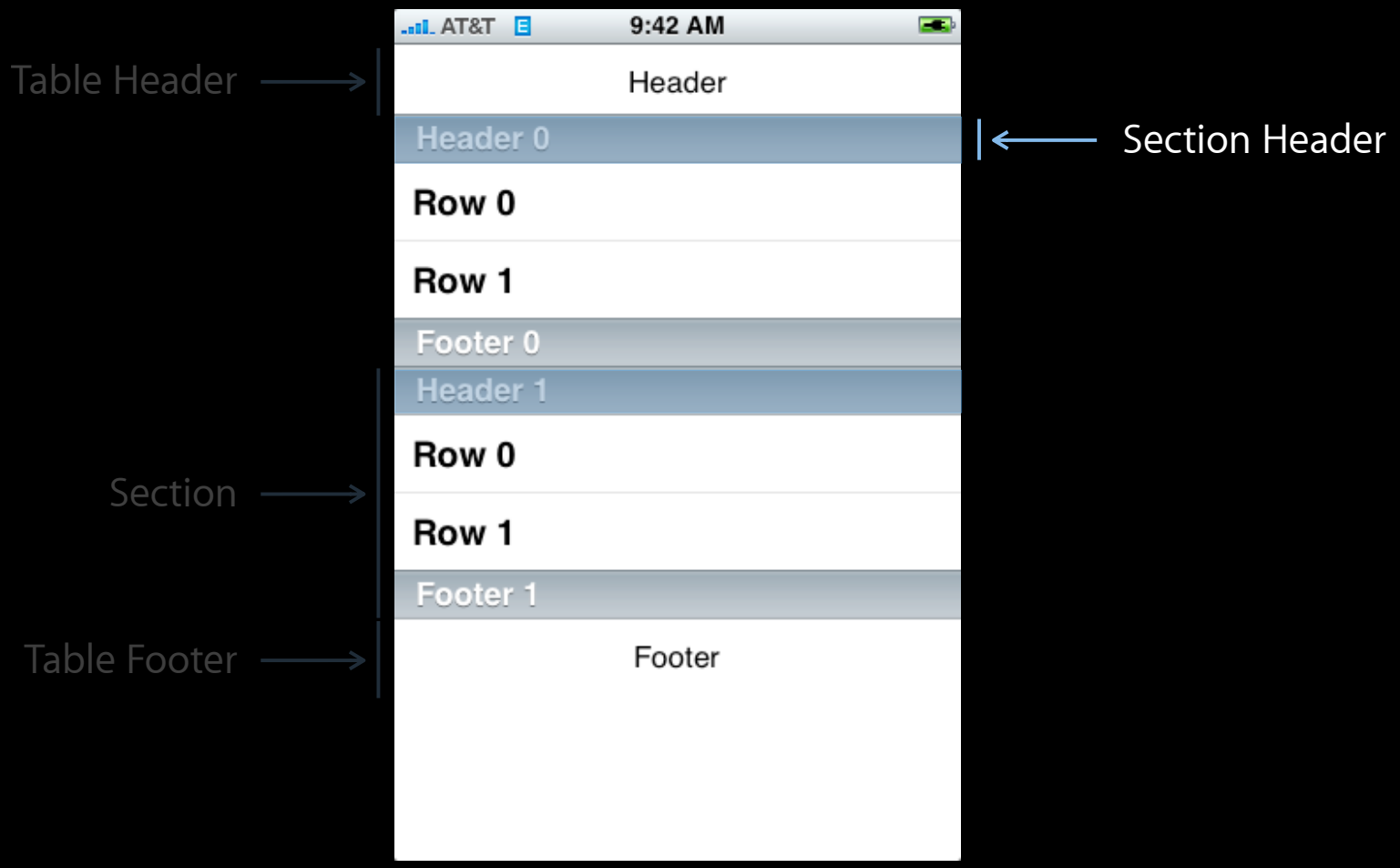


Table View Anatomy

Plain Style

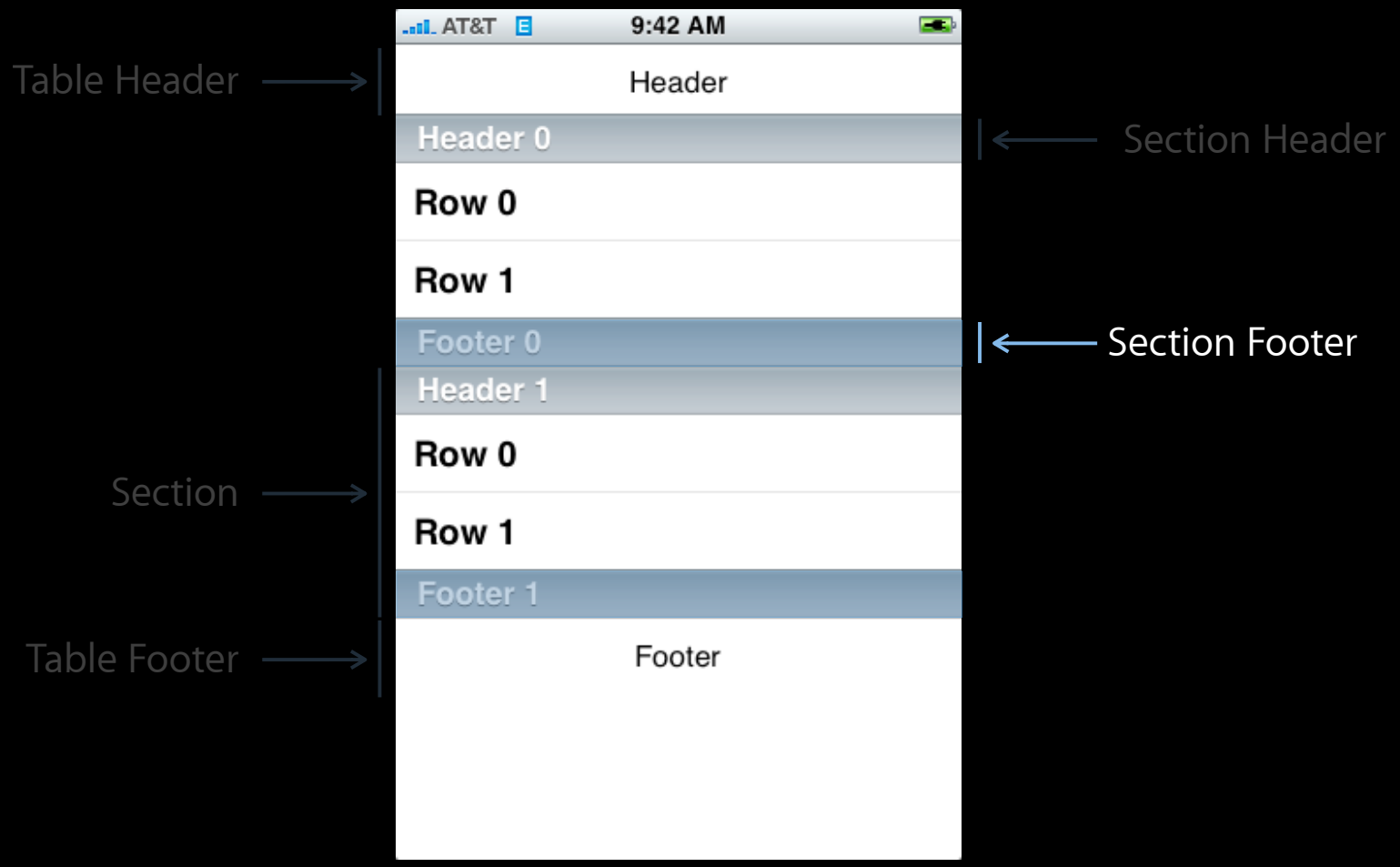


Table View Anatomy

Plain Style

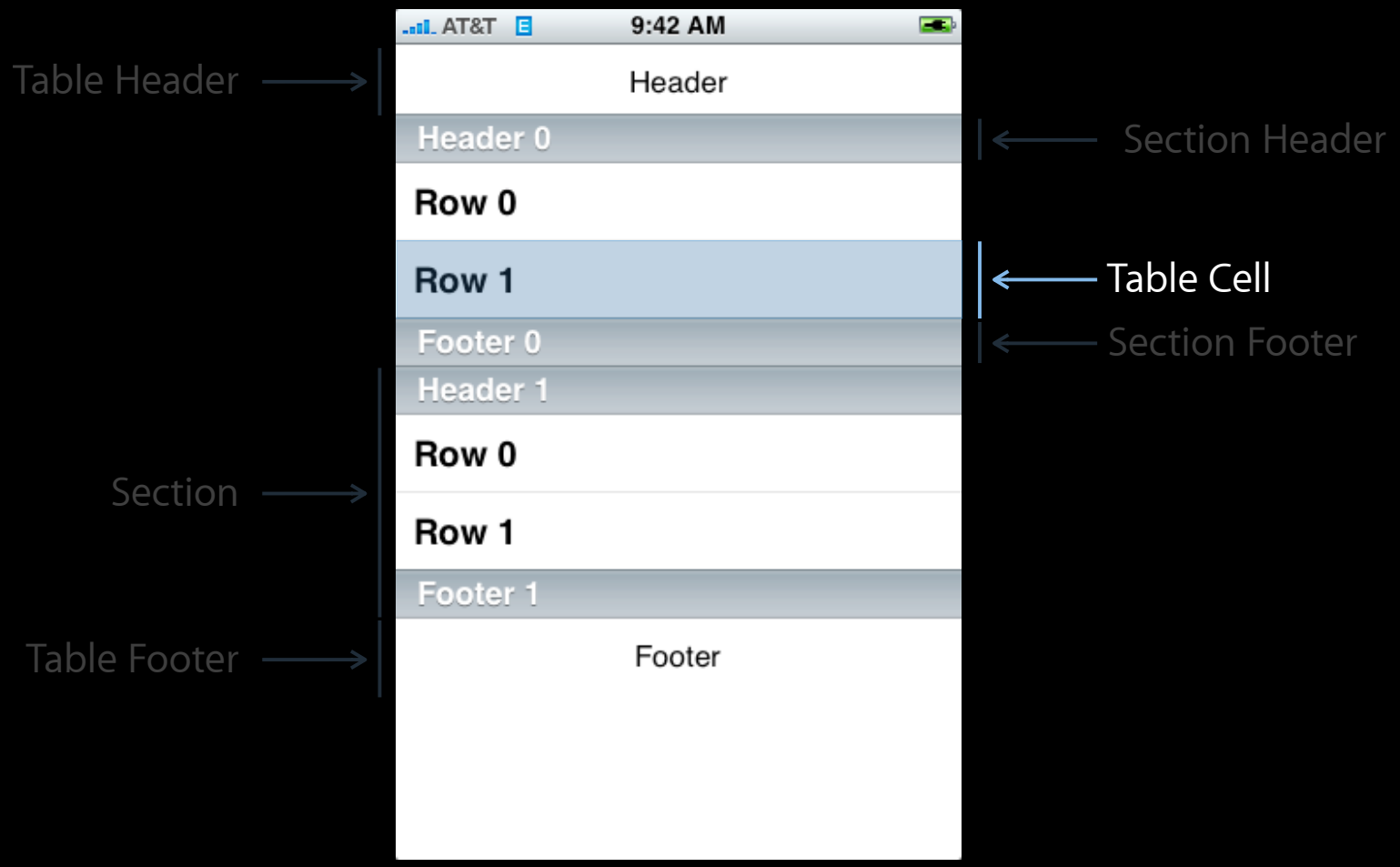


Table View Anatomy

Plain Style

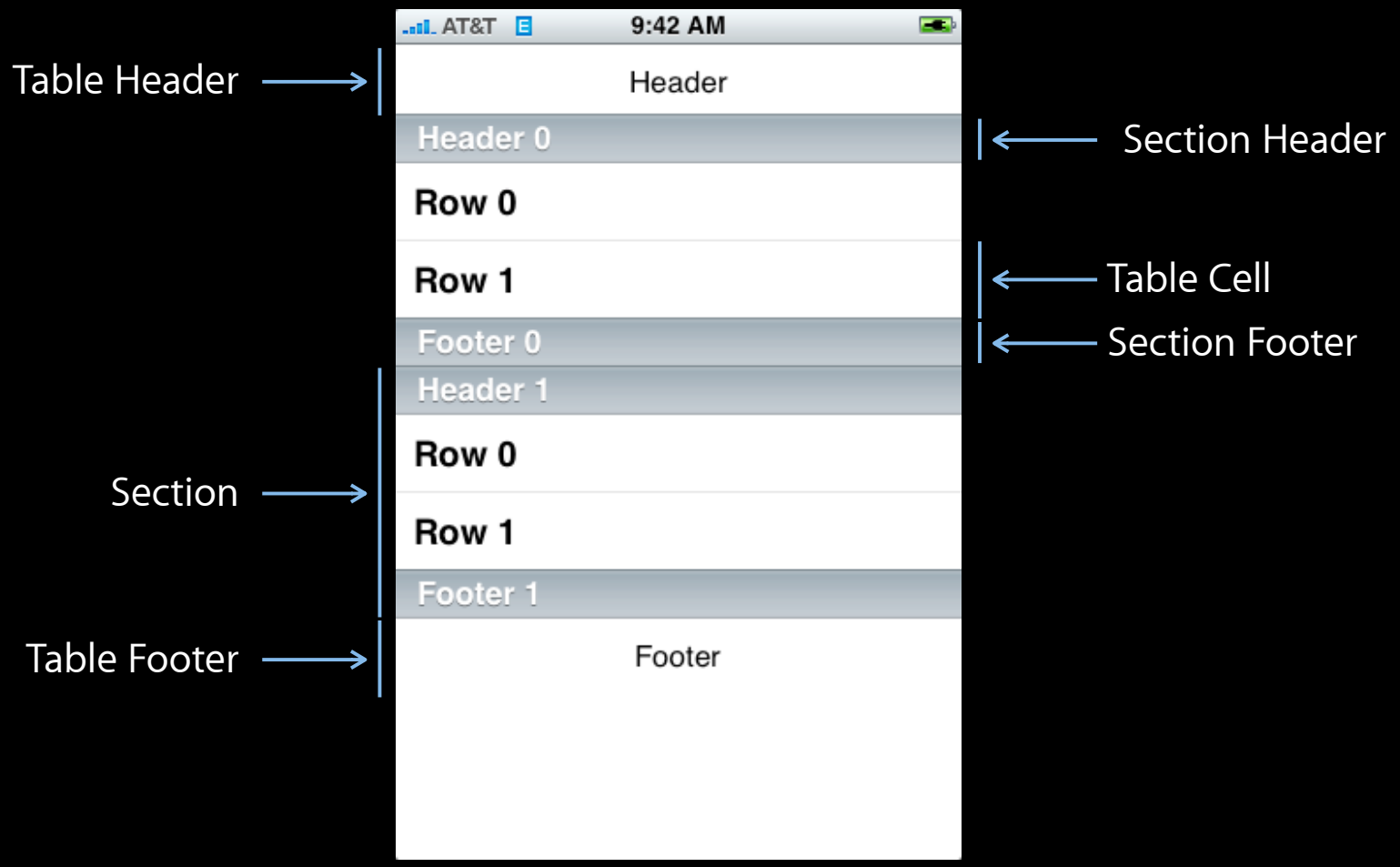
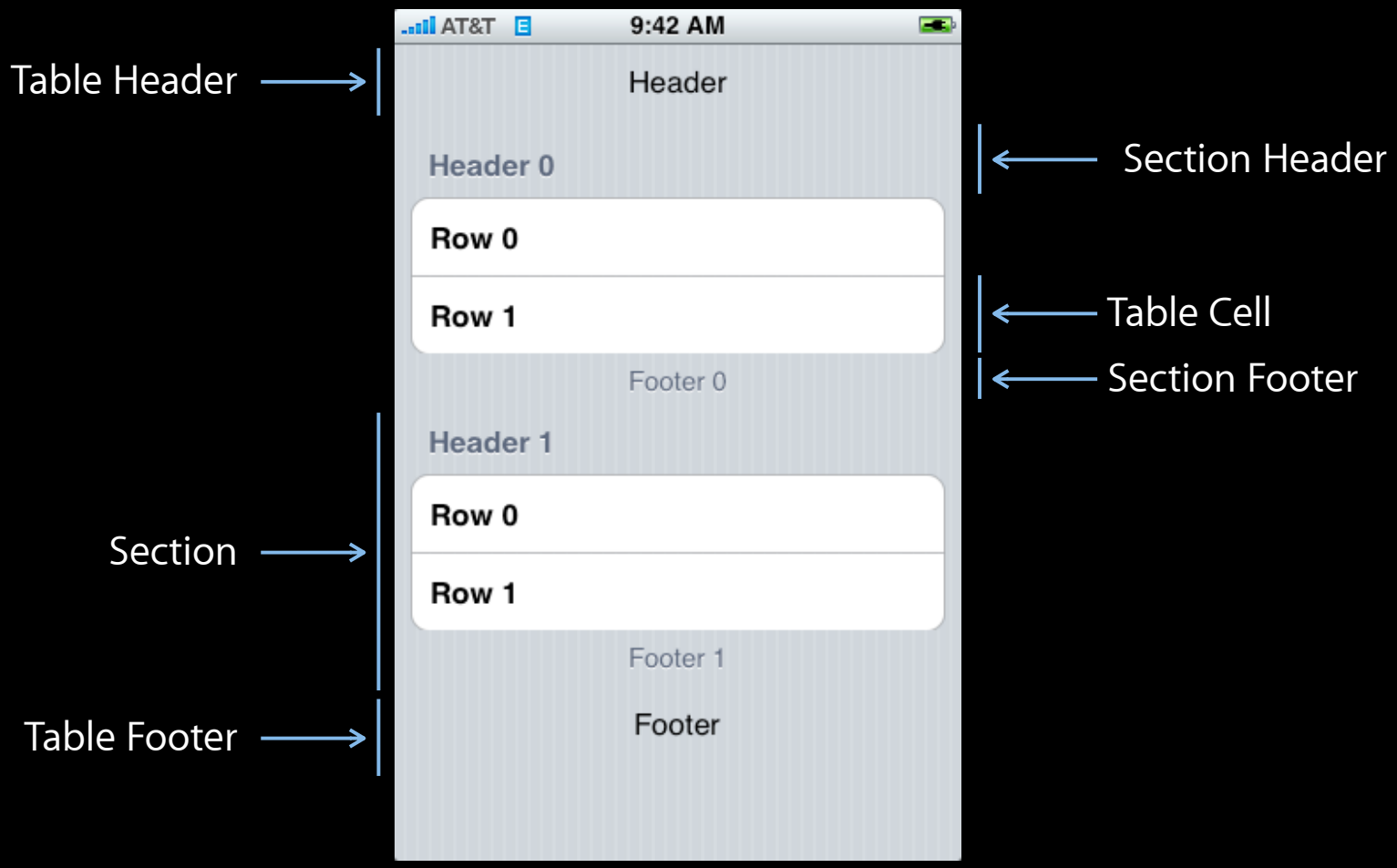


Table View Anatomy

Grouped Style



Using Table Views

- Displaying your data in the table view
- Customizing appearance & behavior

Displaying Data in a Table View

A Naïve Solution

A Naïve Solution

- Table views display a list of data, so use an array

```
[myTableView setList:myListOfStuff];
```

A Naïve Solution

- Table views display a list of data, so use an array

```
[myTableView setList:myListOfStuff];
```

- **Issues with this approach**

A Naïve Solution

- Table views display a list of data, so use an array

```
[myTableView setList:myListOfStuff];
```

- **Issues with this approach**
  - All data is loaded upfront

A Naïve Solution

- Table views display a list of data, so use an array

```
[myTableView setList:myListOfStuff];
```

- **Issues with this approach**
  - All data is loaded upfront
  - All data stays in memory

A More Flexible Solution

- Another object provides data to the table view
  - Not all at once
  - Just as it's needed for display
- Like a delegate, but purely data-oriented

UITableViewDataSource

UITableViewDataSource

- Provide number of sections and rows

```
// Optional method, defaults to 1 if not implemented
- (NSInteger)numberOfSectionsInTableView:(UITableView *)table;

// Required method
- (NSInteger)tableView:(UITableView *)tableView
 numberOfRowsInSection:(NSInteger)section;
```

UITableViewDataSource

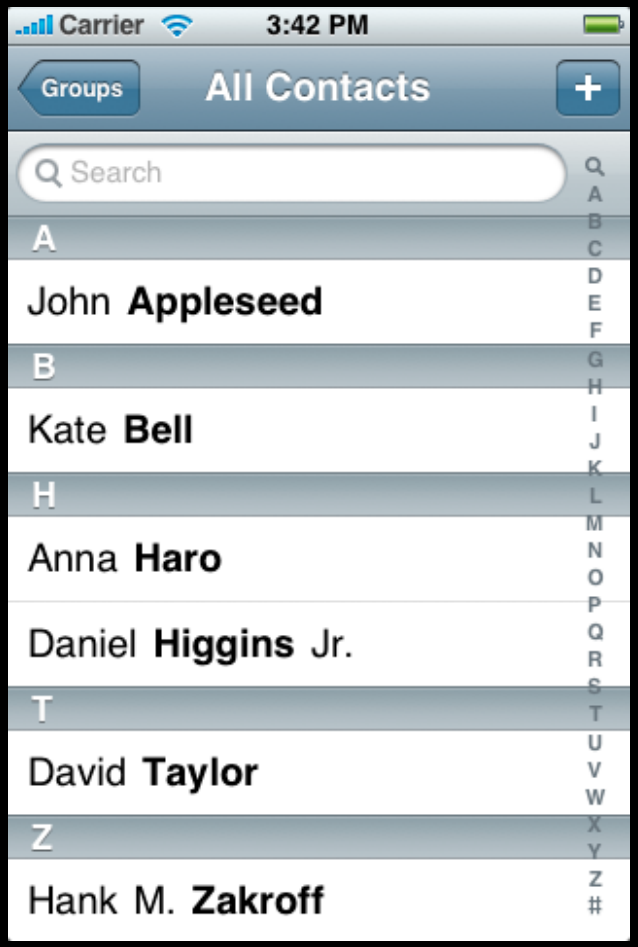
- Provide number of sections and rows

```
// Optional method, defaults to 1 if not implemented
- (NSInteger)numberOfSectionsInTableView:(UITableView *)table;

// Required method
- (NSInteger)tableView:(UITableView *)tableView
 numberOfRowsInSection:(NSInteger)section;
```

- Provide cells for table view as needed

```
// Required method
- (UITableViewCell *)tableView:(UITableView *)tableView
        cellForRowAtIndexPath:(NSIndexPath *)indexPath;
```

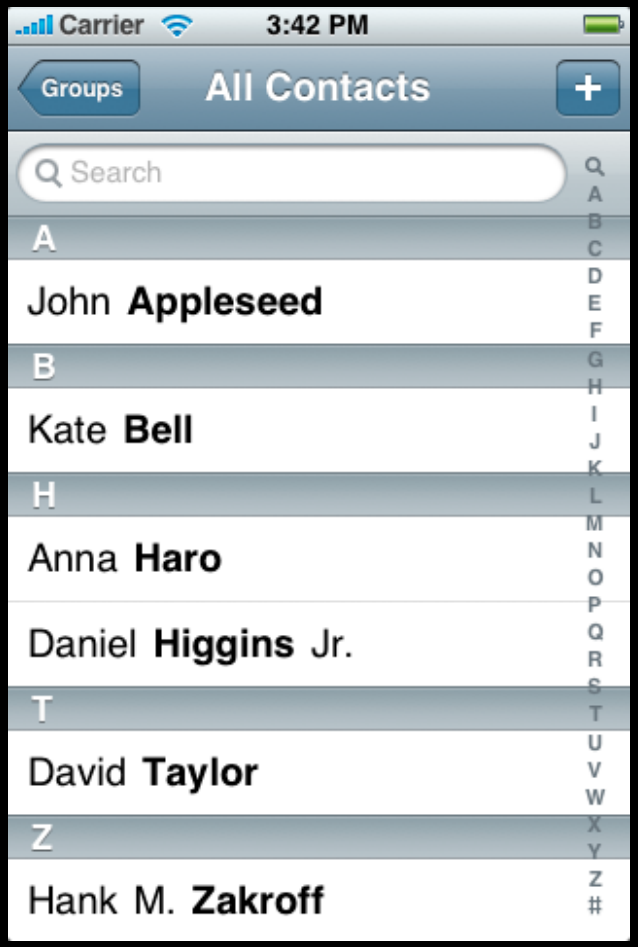
Datasource Message Flow



Datasource

Datasource Message Flow

`numberOfSectionsInTableView:`

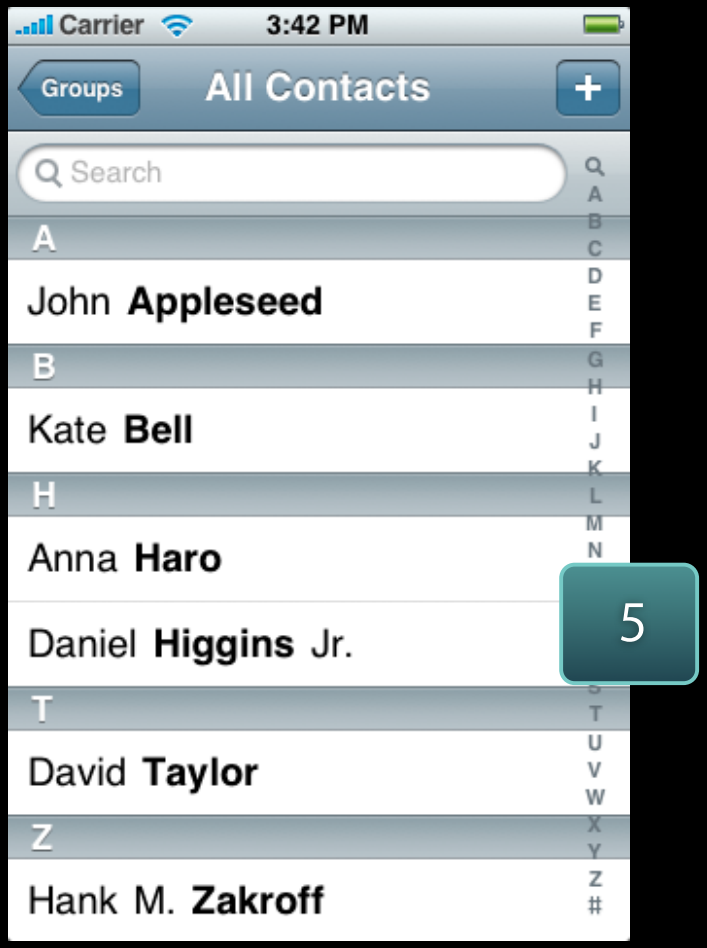


How many sections?

Datasource

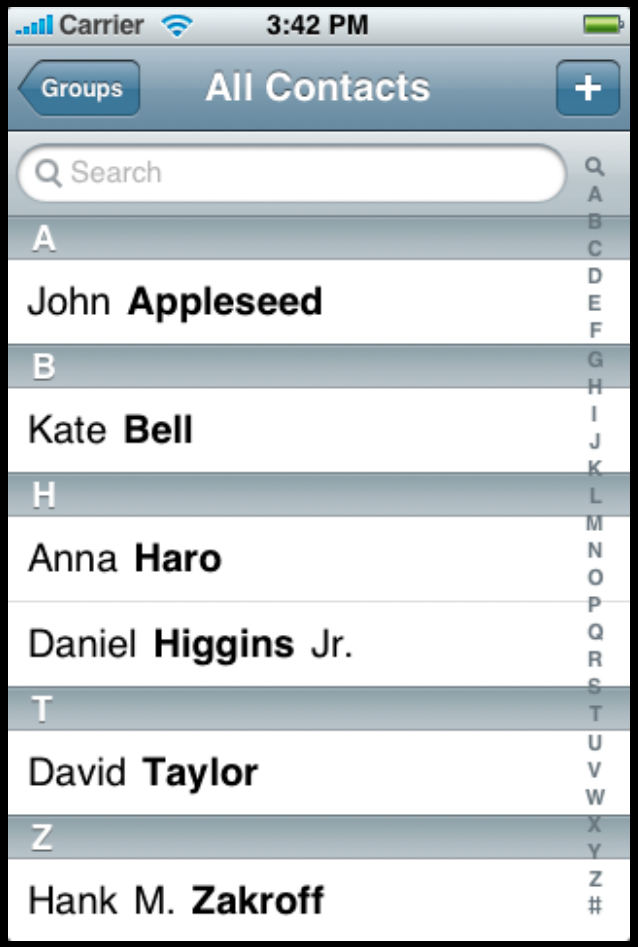
Datasource Message Flow

`numberOfSectionsInTableView:`



Datasource

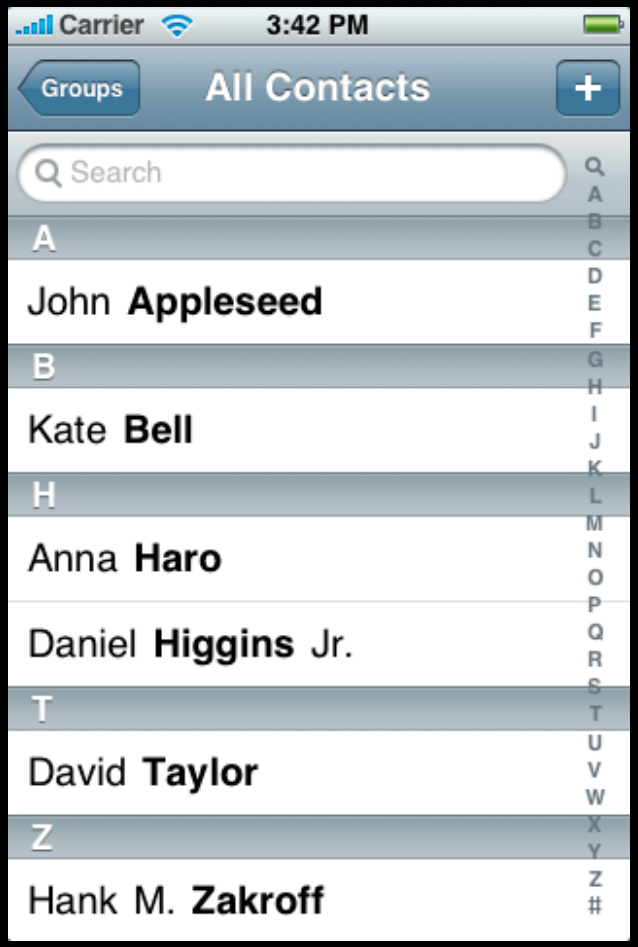
Datasource Message Flow



Datasource

Datasource Message Flow

`tableView:numberOfRowsInSection:`

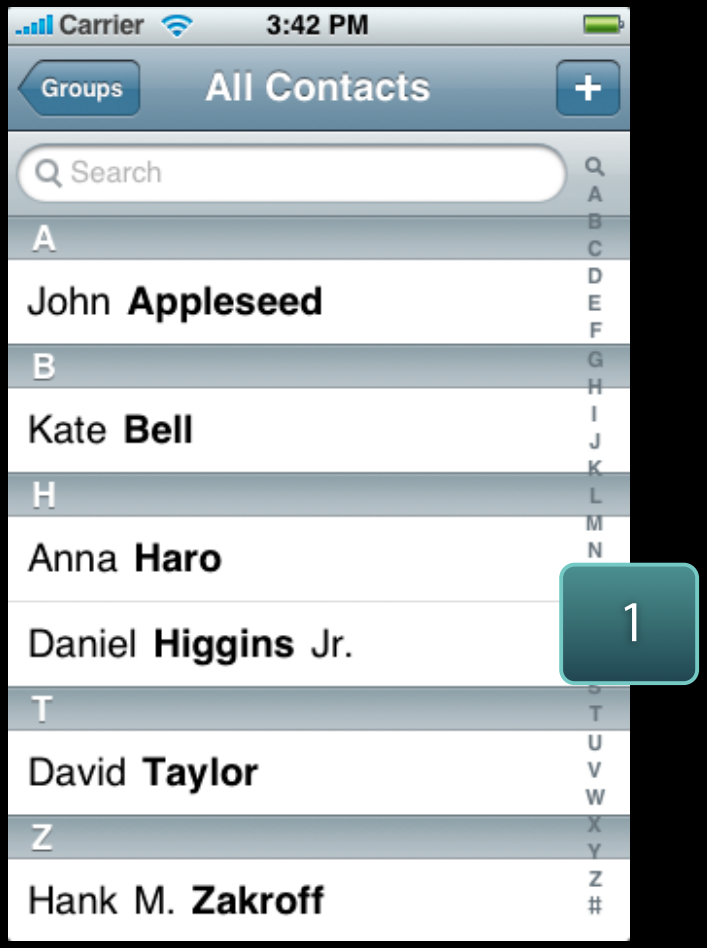


How many rows in section 0?

Datasource

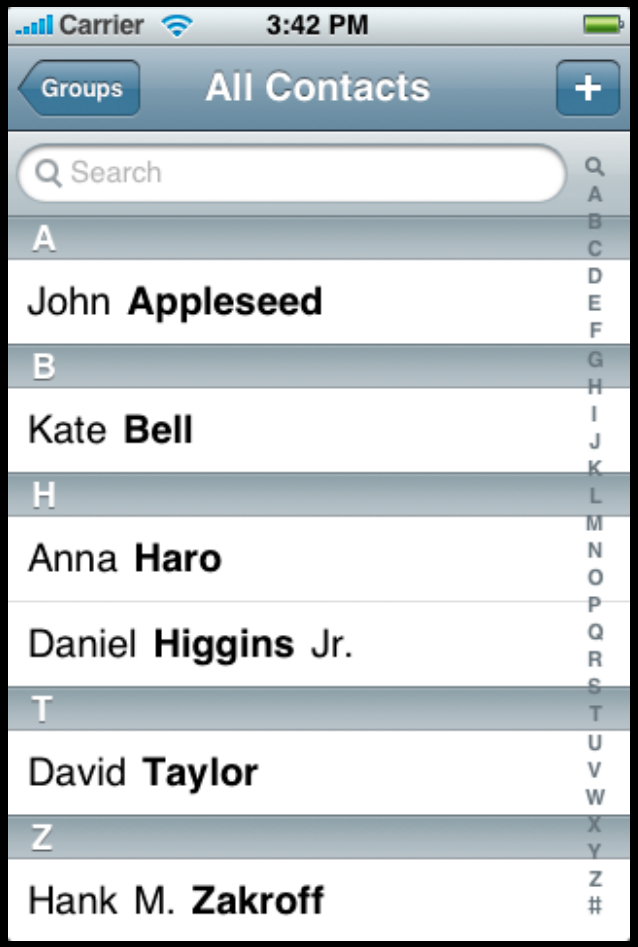
Datasource Message Flow

`tableView:numberOfRowsInSection:`



Datasource

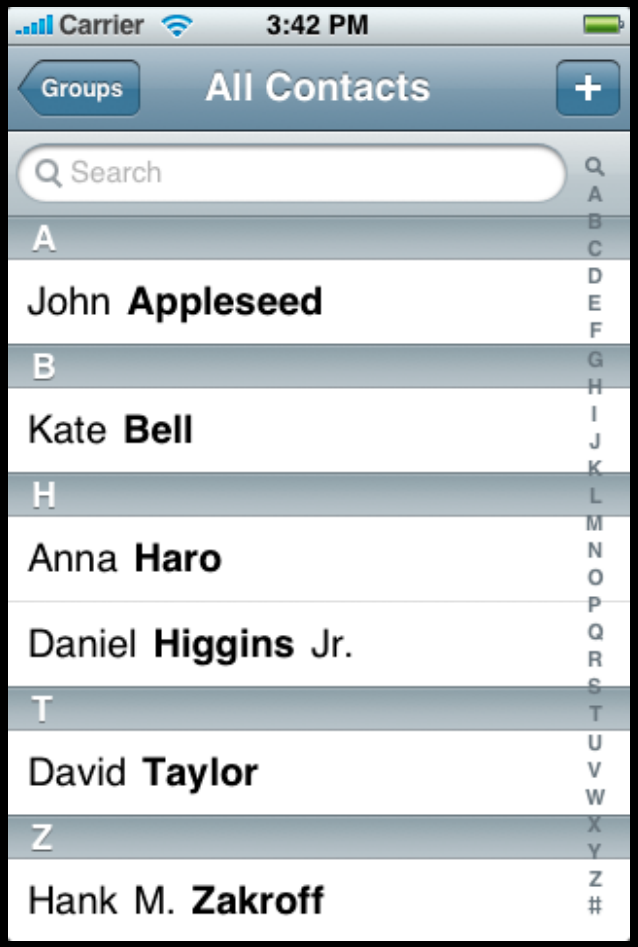
Datasource Message Flow



Datasource

Datasource Message Flow

`tableView:cellForRowAtIndexPath:`

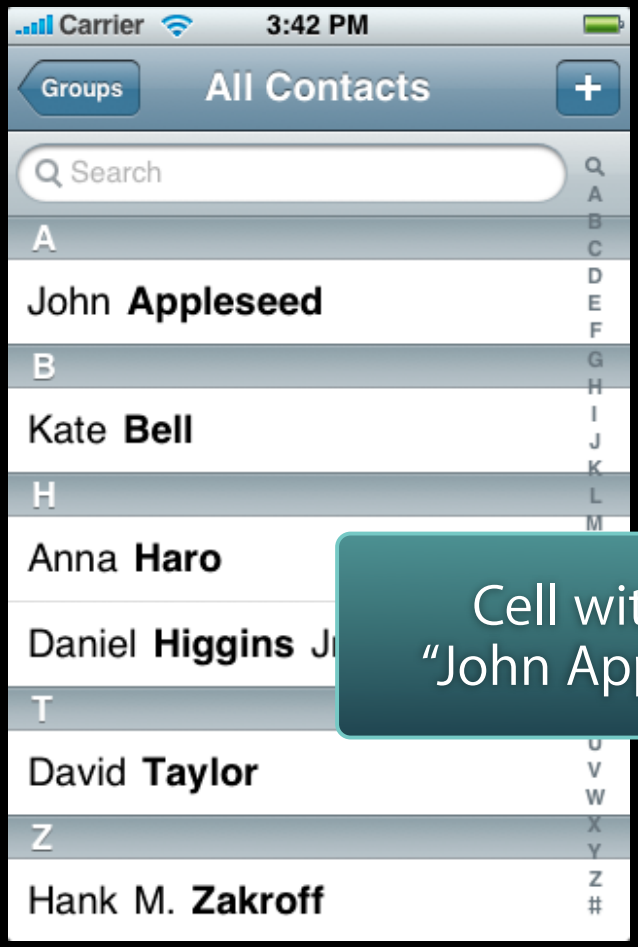


What to display at section 0, row 0?

Datasource

Datasource Message Flow

`tableView:cellForRowAtIndexPath:`



Cell with text
"John Appleseed"

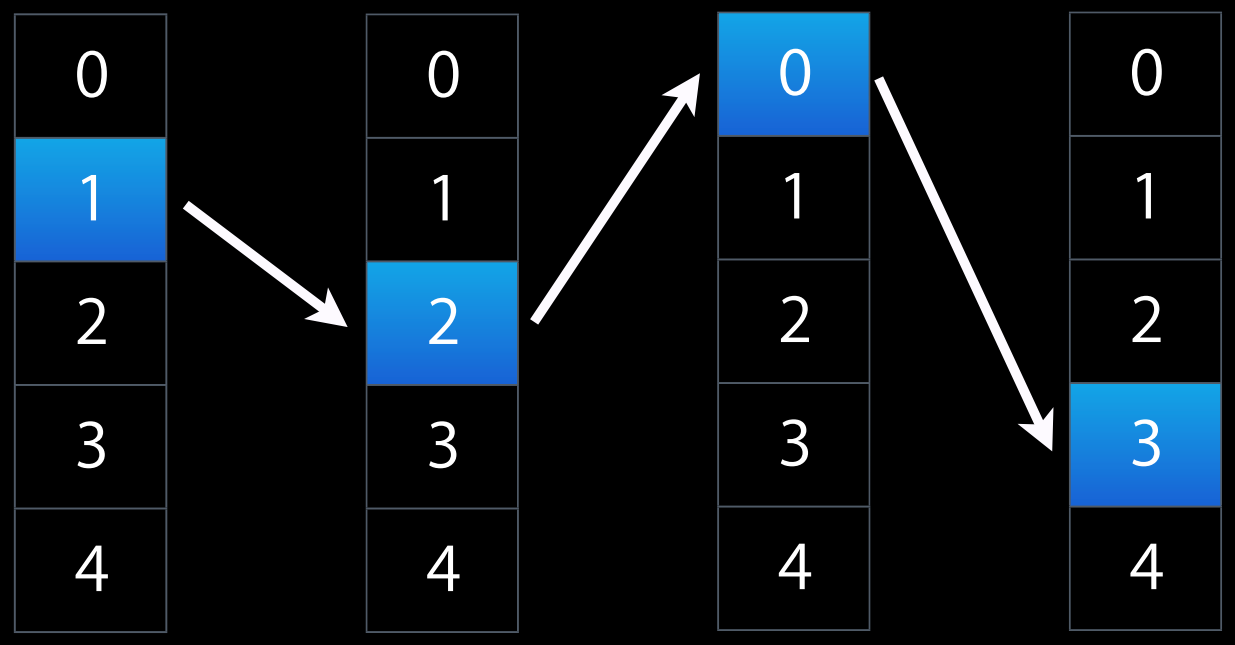
Datasource

NSIndexPath

- Generic class in Foundation
- Path to a specific node in a tree of nested arrays

NSIndexPath

- Generic class in Foundation
- Path to a specific node in a tree of nested arrays



NSIndexPath and Table Views

- Cell location described with an index path
  - Section index + row index

NSIndexPath and Table Views

- Cell location described with an index path
  - Section index + row index
- Category on NSIndexPath with helper methods

```
@interface NSIndexPath (UITableView)

+ (NSIndexPath *)indexPathForRow:(NSUInteger)row
                       inSection:(NSUInteger)section;

@property(nonatomic,readonly) NSUInteger section;
@property(nonatomic,readonly) NSUInteger row;

@end
```

Single Section Table View

Single Section Table View

- Return the number of rows

```
- (NSInteger)tableView:(UITableView *)tableView
 numberOfRowsInSection:(NSInteger)section
{
  return [myStrings count];
}
```

Single Section Table View

- Return the number of rows

```
- (NSInteger)tableView:(UITableView *)tableView
 numberOfRowsInSection:(NSInteger)section
{
  return [myStrings count];
}
```

- Provide a cell when requested

```
- (UITableViewCell *)tableView:(UITableView *)tableView
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
  UITableViewCell *cell = ...;
  cell.textLabel.text = [myStrings objectAtIndex:indexPath.row]
  return [cell autorelease];
}
```

Cell Reuse

Cell Reuse

- When asked for a cell, it would be expensive to create a new cell each time.

Cell Reuse

- When asked for a cell, it would be expensive to create a new cell each time.

```
- (UITableViewCell *)dequeueReusableCellWithIdentifier:
(NSString *)identifier;
```

Cell Reuse

- When asked for a cell, it would be expensive to create a new cell each time.

```
- (UITableViewCell *)tableView:(UITableView *)tableView
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
  UITableViewCell *cell = [tableView
  dequeueReusableCellWithIdentifier:@“MyIdentifier”];

  if (cell == nil) {
      cell = [[[UITableViewCell alloc]
  initWithStyle:... reuseIdentifier:@“MyIdenifier”]
  autorelease];
  }

  cell.text = [myStrings objectAtIndex:indexPath.row]

  return cell;
}
```

Triggering Updates

- When is the datasource asked for its data?

Triggering Updates

- When is the datasource asked for its data?
  - When a row becomes visible

Triggering Updates

- When is the datasource asked for its data?
  - When a row becomes visible
  - When an update is explicitly requested by calling -reloadData

```
- (void)viewWillAppear:(BOOL)animated
{
  [super viewWillAppear:animated];

  [self.tableView reloadData];
}
```

Section and Row Reloading

Section and Row Reloading

```
- (void)insertSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;
```

Section and Row Reloading

```
- (void)insertSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;
- (void)deleteSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;
```

Section and Row Reloading

```
- (void)insertSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)deleteSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)reloadSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;
```

Section and Row Reloading

```
- (void)insertSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)deleteSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)reloadSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;


- (void)insertRowsAtIndexPaths:(NSArray *)indexPaths
      withRowAnimation:(UITableViewRowAnimation)animation;
```

Section and Row Reloading

```
- (void)insertSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)deleteSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)reloadSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;


- (void)insertRowsAtIndexPaths:(NSArray *)indexPaths
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)deleteRowsAtIndexPaths:(NSArray *)indexPaths
      withRowAnimation:(UITableViewRowAnimation)animation;
```

Section and Row Reloading

```
- (void)insertSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)deleteSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)reloadSections:(NSIndexSet *)sections
      withRowAnimation:(UITableViewRowAnimation)animation;


- (void)insertRowsAtIndexPaths:(NSArray *)indexPaths
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)deleteRowsAtIndexPaths:(NSArray *)indexPaths
      withRowAnimation:(UITableViewRowAnimation)animation;

- (void)reloadRowsAtIndexPaths:(NSArray *)indexPaths
              withRowAnimation:(UITableViewRowAnimation)animation;
```

Additional Datasource Methods

- Titles for section headers and footers
- Allow editing and reordering cells

Appearance & Behavior

UITableView Delegate

- Customize appearance and behavior
- **Keep application logic separate from view**
- Often the same object as datasource

Table View Appearance & Behavior

Table View Appearance & Behavior

- Customize appearance of table view cell

```
- (void)tableView:(UITableView *)tableView
  willDisplayCell:(UITableViewCell *)cell
forRowAtIndexPath:(NSIndexPath *)indexPath;
```

Table View Appearance & Behavior

- Customize appearance of table view cell

```
- (void)tableView:(UITableView *)tableView
  willDisplayCell:(UITableViewCell *)cell
forRowAtIndexPath:(NSIndexPath *)indexPath;
```

- Validate and respond to selection changes

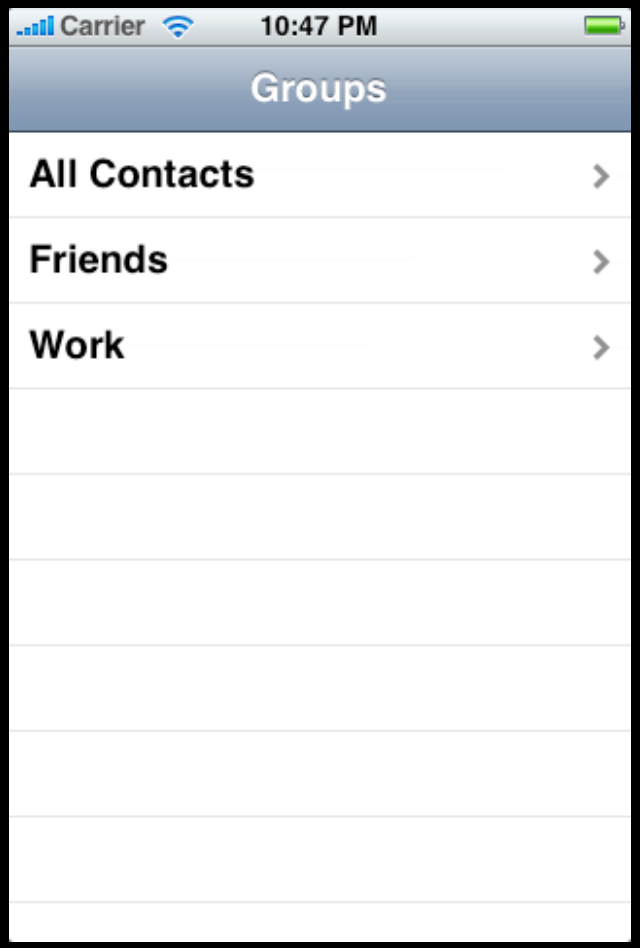
```
- (NSIndexPath *)tableView:(UITableView *)tableView
willSelectRowAtIndexPath:(NSIndexPath *)indexPath;

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath;
```

Row Selection in Table Views

- In iPhone applications, **rows rarely stay selected**
- Selecting a row usually triggers an event

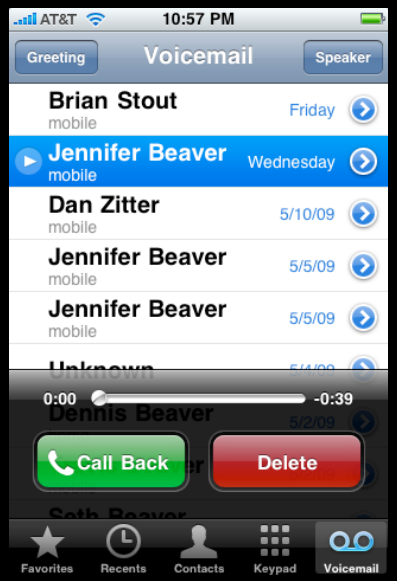
Row Selection in Table Views

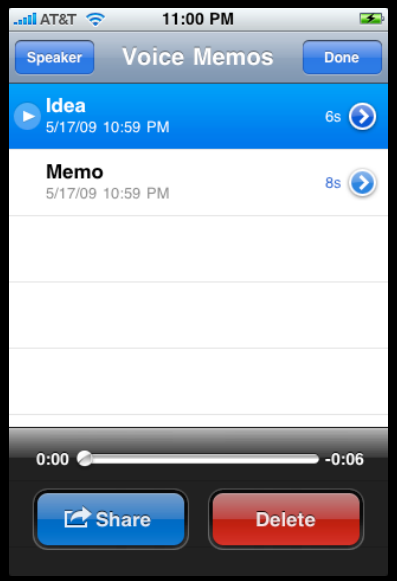
- In iPhone applications, **rows rarely stay selected**
- Selecting a row usually triggers an event



Persistent Selection

Persistent Selection





Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
```

Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Get the row and the object it represents
```

Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Get the row and the object it represents
  NSUInteger row = indexPath.row
```

Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Get the row and the object it represents
  NSUInteger row = indexPath.row
  id objectToDisplay = [myObjects objectAtIndex:row];
```

Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Get the row and the object it represents
  NSUInteger row = indexPath.row
  id objectToDisplay = [myObjects objectAtIndex:row];

  // Create a new view controller and pass it along
```

Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Get the row and the object it represents
  NSUInteger row = indexPath.row
  id objectToDisplay = [myObjects objectAtIndex:row];

  // Create a new view controller and pass it along
  MyViewController *myViewController = ...;
```

Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Get the row and the object it represents
  NSUInteger row = indexPath.row
  id objectToDisplay = [myObjects objectAtIndex:row];

  // Create a new view controller and pass it along
  MyViewController *myViewController = ...;
  myViewController.object = objectToDisplay;
```

Responding to Selection

```
// For a navigation hierarchy...
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Get the row and the object it represents
  NSUInteger row = indexPath.row
  id objectToDisplay = [myObjects objectAtIndex:row];

  // Create a new view controller and pass it along
  MyViewController *myViewController = ...;
  myViewController.object = objectToDisplay;

  [self.navigationController
   pushViewController:myViewController animated:YES];
}
```

Altering or Disabling Selection

```
- (NSIndexPath *)tableView:(UITableView *)tableView
willSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  // Don't allow selecting certain rows?
  if (indexPath.row == ...) {
    return nil;
  } else {
    return indexPath;
  }
}
```

UITableViewController

UITableViewController

UITableViewController

- Convenient starting point for view controller with a table view
  - Table view is automatically created
  - **Controller is table view's delegate and datasource**

UITableViewController

- Convenient starting point for view controller with a table view
  - Table view is automatically created
  - **Controller is table view's delegate and datasource**
- Takes care of some default behaviors
  - Calls -reloadData the first time it appears
  - Deselects rows when user navigates back
  - Flashes scroll indicators

Table View Cells

Designated Initializer

Designated Initializer

```
- (id)initWithFrame:(CGRect)frame
    reuseIdentifier:(NSString *)reuseIdentifier;
```

Designated Initializer

DEPRECATED

```
- (id)initWithFrame:(CGRect)frame
    reuseIdentifier:(NSString *)reuseIdentifier;
```

```
- (id)initWithStyle:(UITableViewCellStyle)style
    reuseIdentifier:(NSString *)reuseIdentifier;
```

Cell Styles

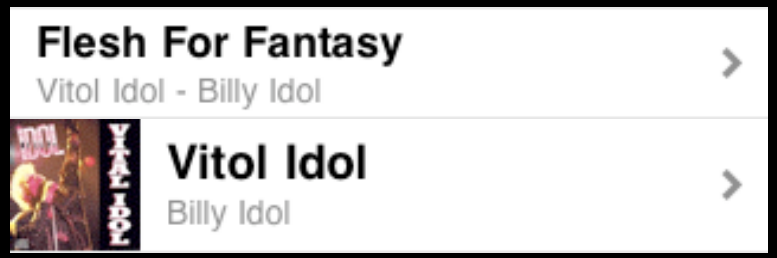
Cell Styles

UITableViewCellStyleDefault

Cell Styles

UITableViewCellStyleDefault

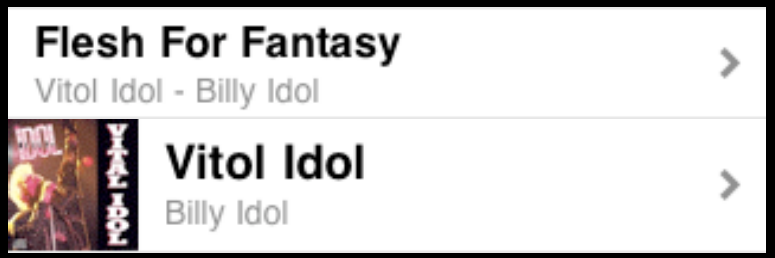
UITableViewCellStyleSubtitle



Cell Styles

UITableViewCellStyleDefault

UITableViewCellStyleSubtitle

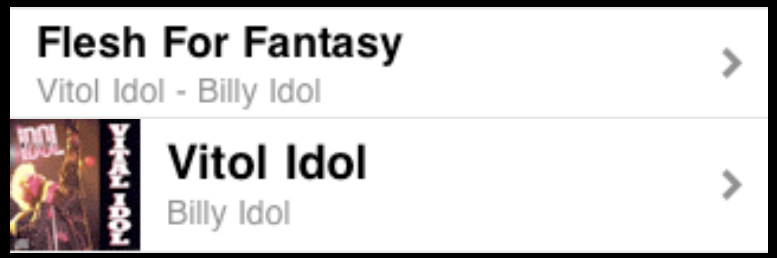


UITableViewCellStyleValue1

Cell Styles

UITableViewCellStyleDefault

UITableViewCellStyleSubtitle



UITableViewCellStyleValue1

UITableViewCellStyleValue2

Basic properties

- UITableViewCell has an image view and one or two text labels

Basic properties

- UITableViewCell has an image view and one or two text labels

```
cell.imageView.image = [UIImage imageNamed:@“vitolidol.png”];
cell.textLabel.text = @“Vitol Idol”;
cell.detailTextLabel.text = @“Billy Idol”;
```

Accessory Types

```
// UITableView delegate method
- (UITableViewCellAccessoryType)tableView:(UITableView *)table
accessoryTypeForRowWithIndexPath:(NSIndexPath *)indexPath;
```

Accessory Types

```
// UITableView delegate method
- (UITableViewCellAccessoryType)tableView:(UITableView *)table
accessoryTypeForRowWithIndexPath:(NSIndexPath *)indexPath;
```

UITableViewCellAccessoryDisclosureIndicator

Accessory Types

```
// UITableView delegate method
- (UITableViewCellAccessoryType)tableView:(UITableView *)table
accessoryTypeForRowWithIndexPath:(NSIndexPath *)indexPath;
```

UITableViewCellAccessoryDisclosureIndicator

UITableViewCellAccessoryDetailDisclosureButton

Accessory Types

```
// UITableView delegate method
- (UITableViewCellAccessoryType)tableView:(UITableView *)table
accessoryTypeForRowWithIndexPath:(NSIndexPath *)indexPath;
```

UITableViewCellAccessoryDisclosureIndicator

UITableViewCellAccessoryDetailDisclosureButton


```
- (void)tableView:(UITableView *)tableView
accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath
{
  // Only for the blue disclosure button
  NSUInteger row = indexPath.row;
  ...
}
```

Accessory Types

```
// UITableView delegate method
- (UITableViewCellAccessoryType)tableView:(UITableView *)table
accessoryTypeForRowWithIndexPath:(NSIndexPath *)indexPath;
```

UITableViewCellAccessoryDisclosureIndicator

UITableViewCellAccessoryDetailDisclosureButton

UITableViewCellAccessoryCheckmark


```
- (void)tableView:(UITableView *)tableView
accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath
{
  // Only for the blue disclosure button
  NSUInteger row = indexPath.row;
  ...
}
```

Customizing the Content View

- For cases where a simple image + text cell doesn't suffice
- UITableViewCell has a content view property
  - **Add additional views to the content view**

Customizing the Content View

- For cases where a simple image + text cell doesn't suffice
- UITableViewCell has a content view property
  - **Add additional views to the content view**

```
- (UITableViewCell *)tableView:(UITableView *)tableView
  cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
  UITableViewCell *cell = ...;
  CGRect frame = cell.contentView.bounds;

  UILabel *myLabel = [[UILabel alloc] initWithFrame:frame];
  myLabel.text = ...;
  [cell.contentView addSubview:myLabel];

  [myLabel release];
}
```


9:42 AM
Messages
Calendar
Photos
YouTube
Stocks
Maps
Weather
Clock
Calculator
Notes
Settings
Phone
Contacts
Mail
Safari
iPod

Questions?