CS193P - Lecture 8

iPhone Application Development

Scroll Views & Table Views

Announcements

- Presence 1 due tomorrow (4/28)
  - Questions?
- Presence 2 due next Tuesday (5/5)

Announcements

- Enrolled students who requested iPod touches can pick them up after class today
  - Need Student ID
  - No grade if not returned!

Today's Topics

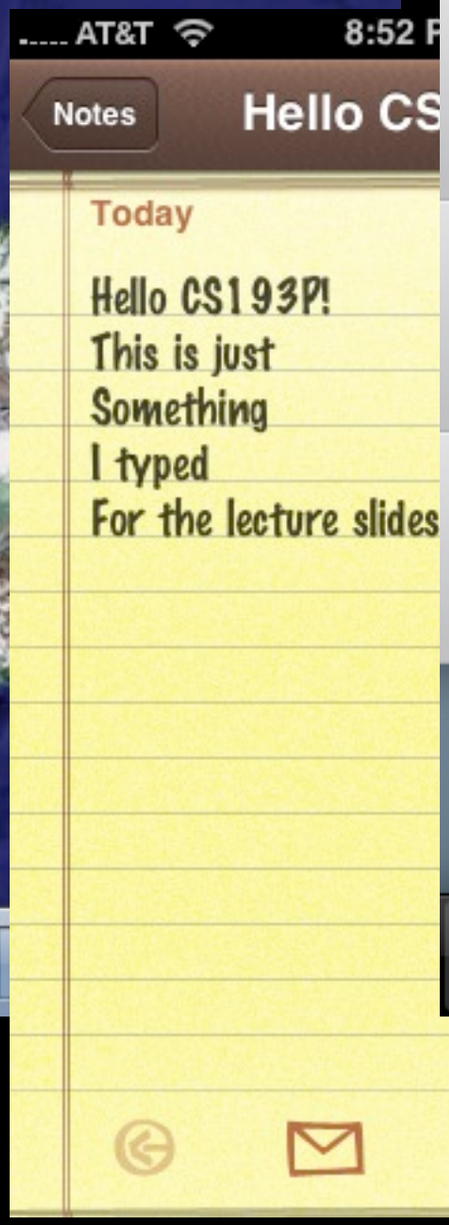
- Scroll views
- Table views
  - Displaying data
  - Controlling appearance & behavior
- UITableViewController
- Table view cells
- Presence - Part 2

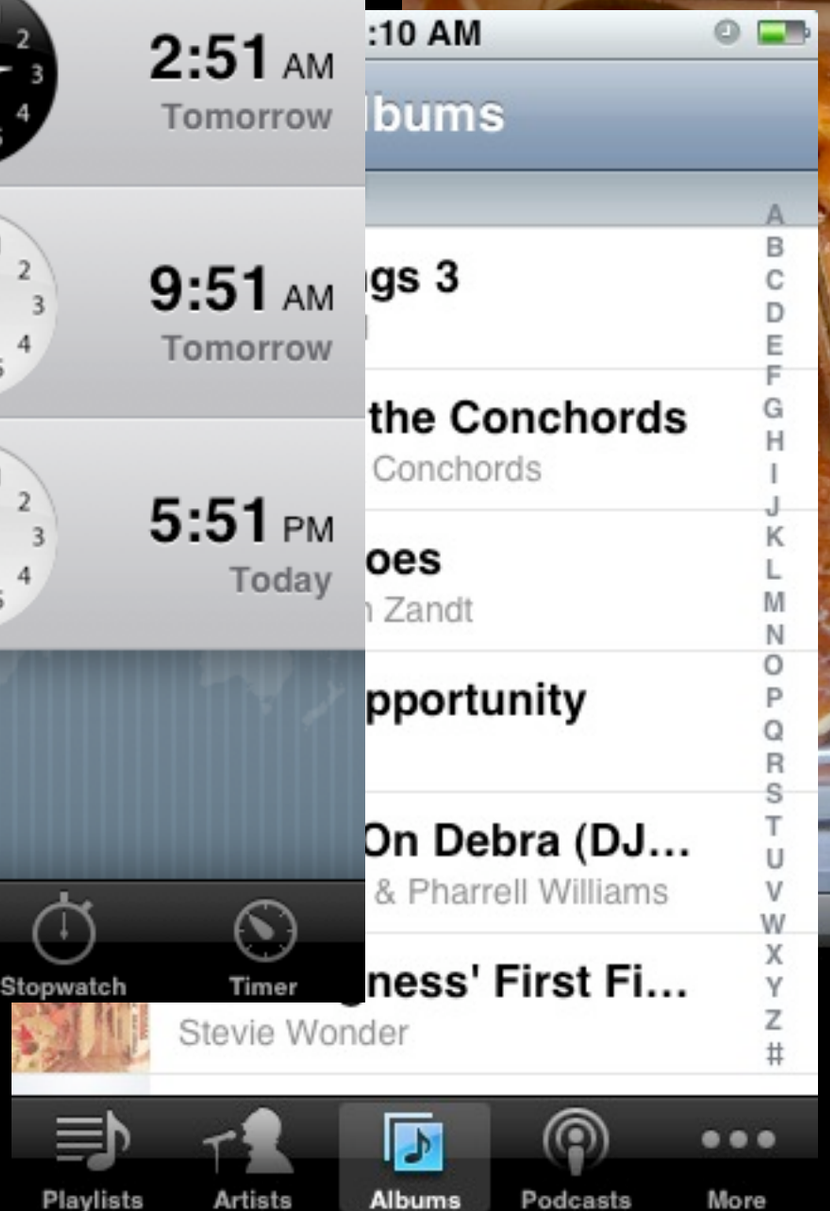
Scroll Views

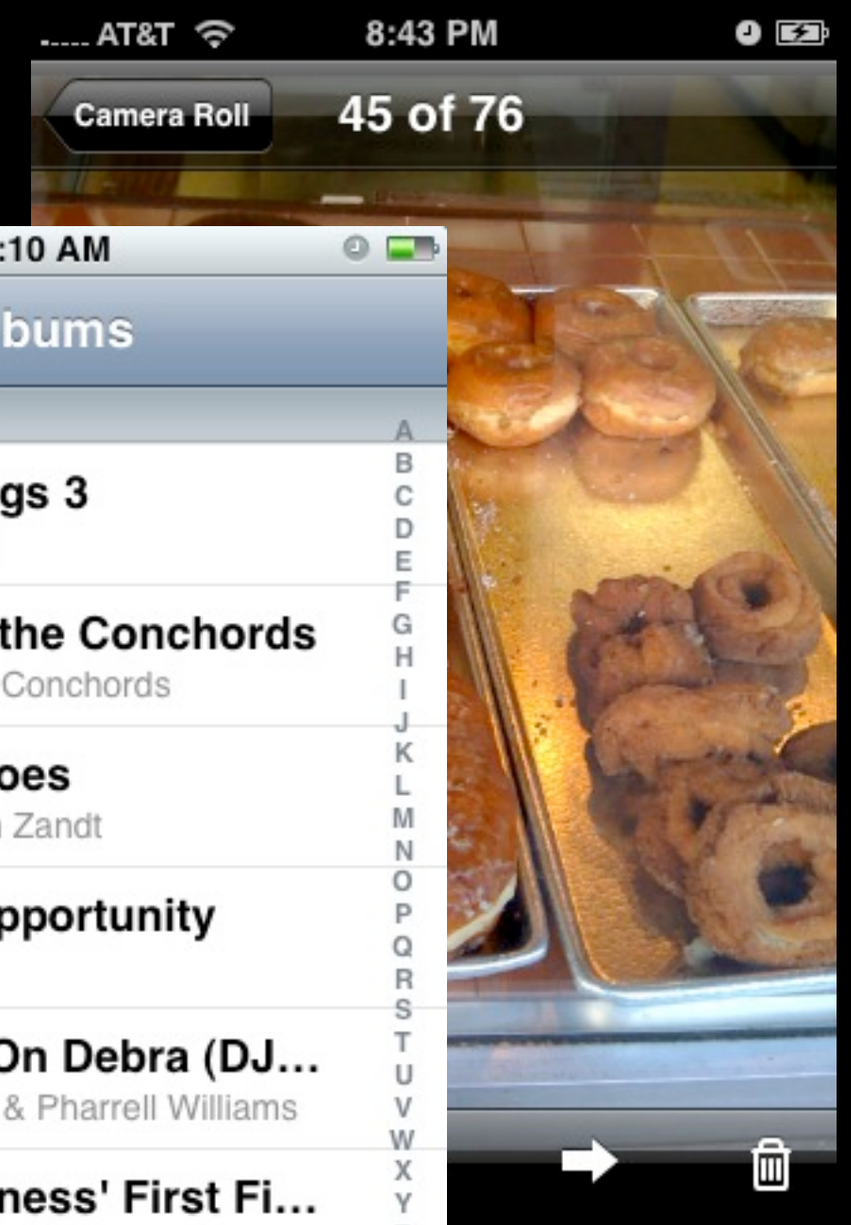
UIScrollView

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

Scrolling Examples







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 frame)

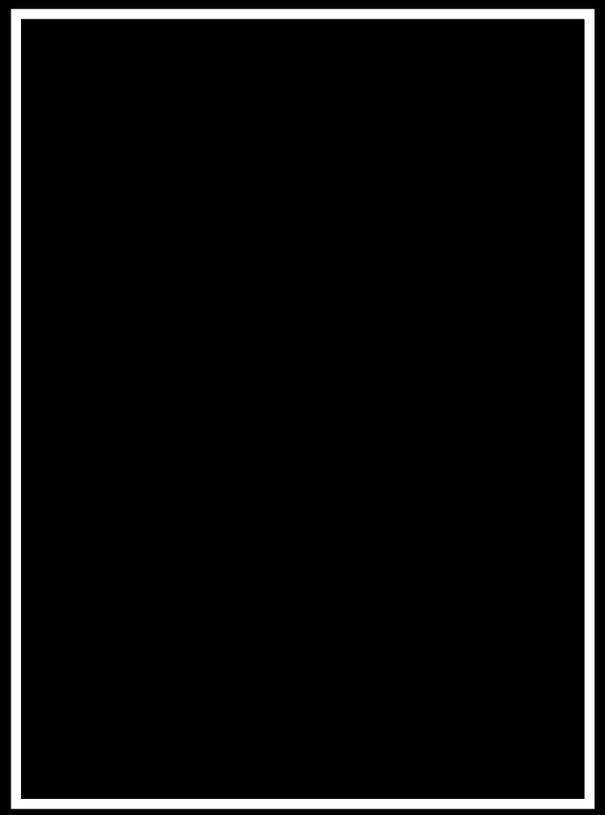
```
frame = CGRectMake(0, 0, 500, 500);
myImageView = [[UIImageView alloc] initWithFrame:frame];
[scrollView addSubview:myImageView];
```

- Set the content size

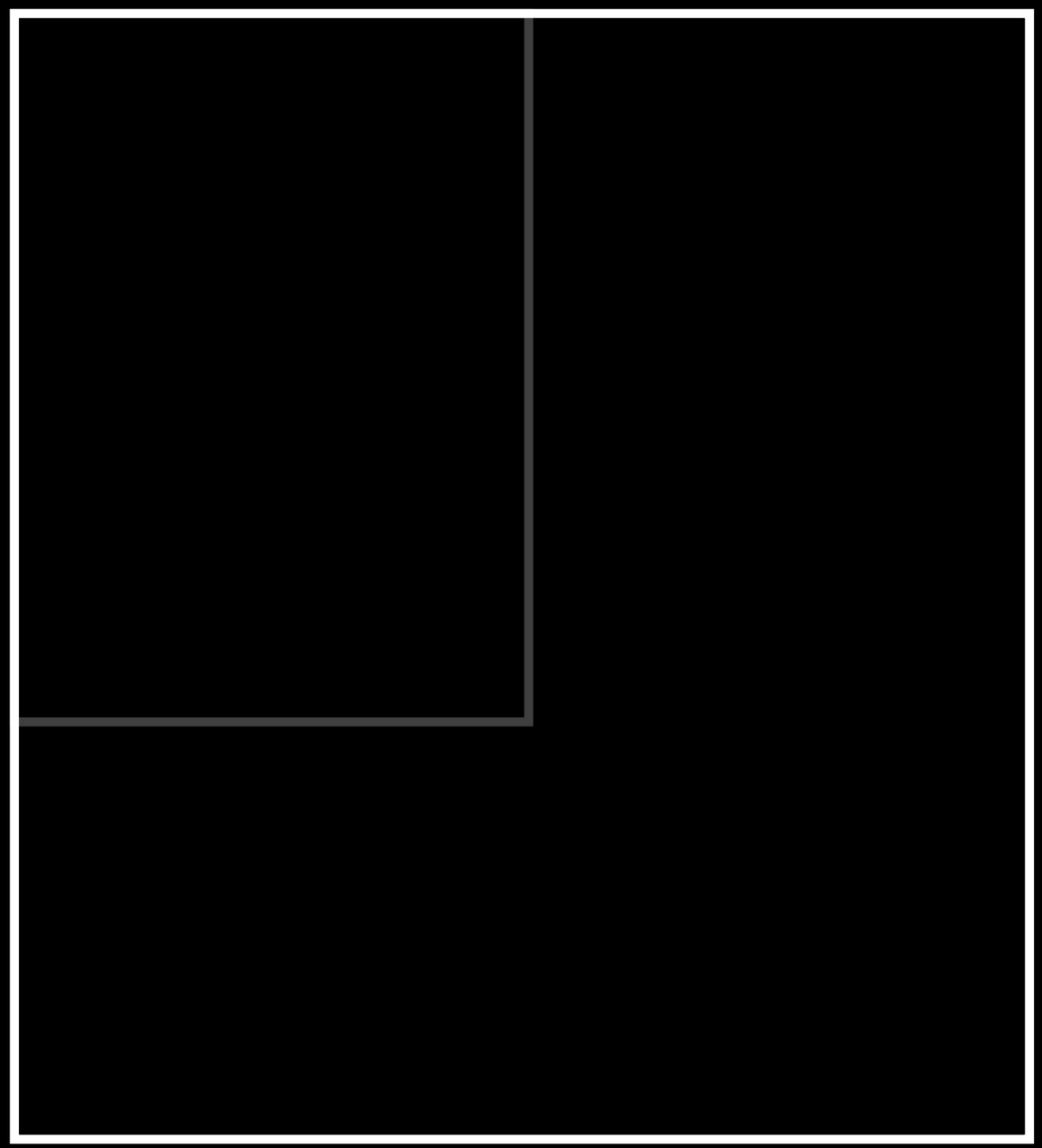
```
scrollView.contentSize = CGSizeMake(500, 500);
```

Frame and Content

scrollView.frame

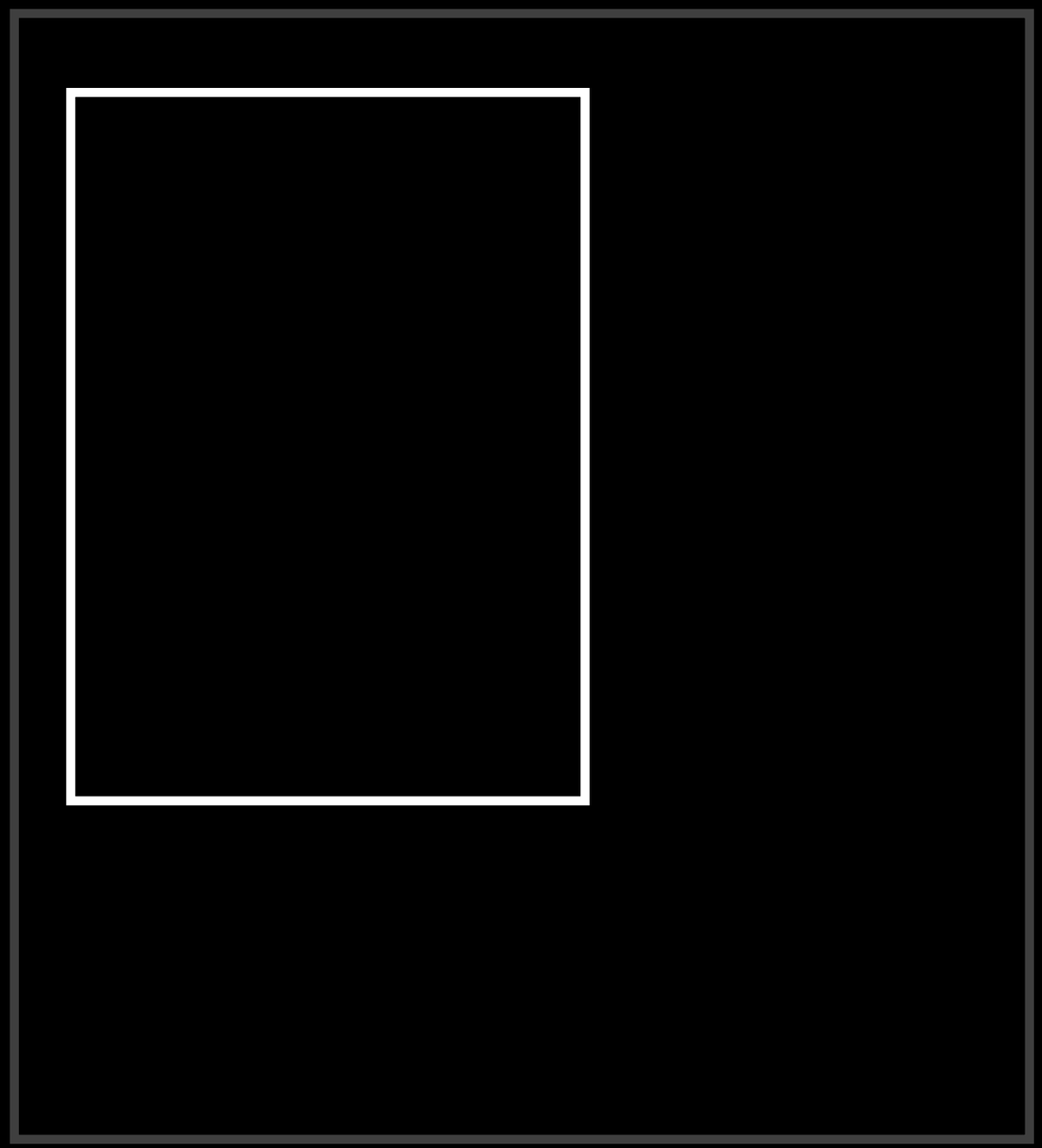


Frame and Content



Frame and Content

`scrollView.contentOffset`



Demo:
Using a UIScrollView

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
- **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

```
@protocol UIScrollViewDelegate<NSObject>

@optional

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

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

@end
```

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

- 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;
}
```

Demo:
Zooming with a UIScrollView

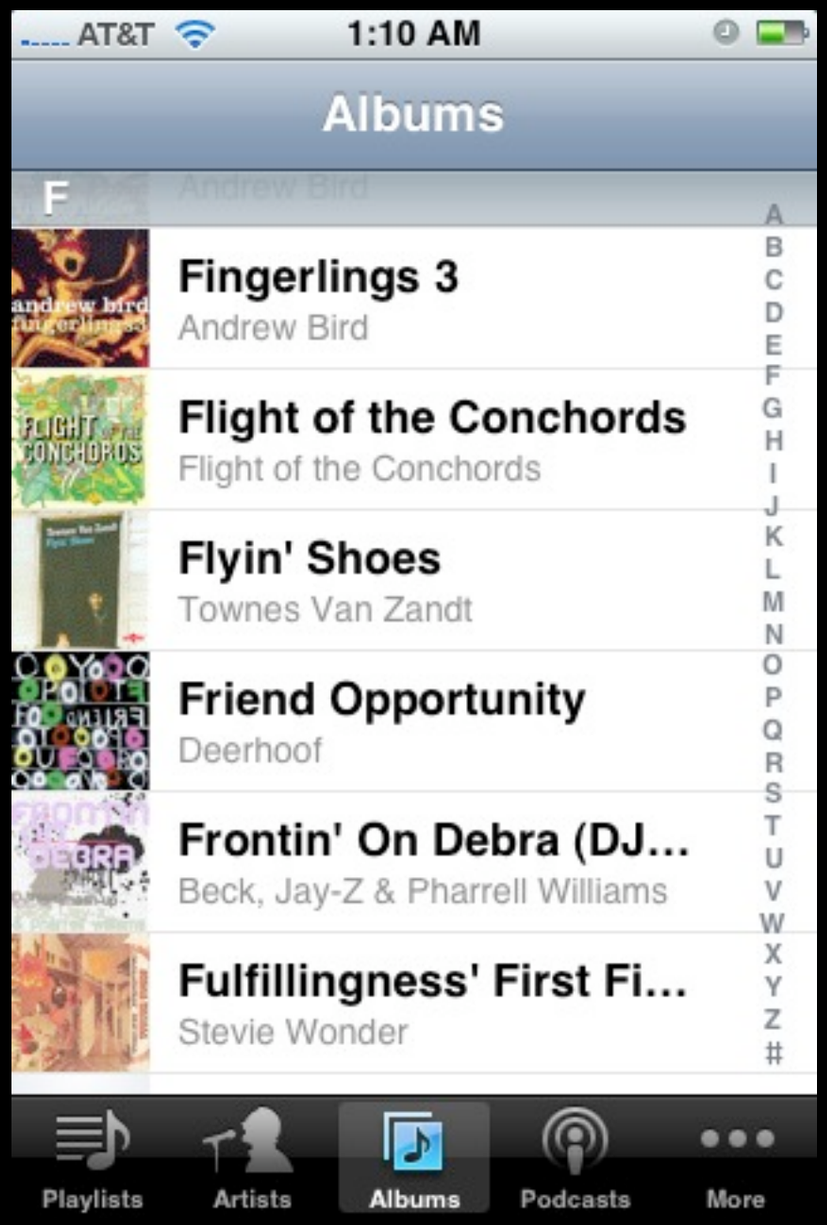
Table Views

Table Views

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

Table View Styles

UITableViewStylePlain



UITableViewStyleGrouped

Table View Anatomy

Plain Style

Table Header

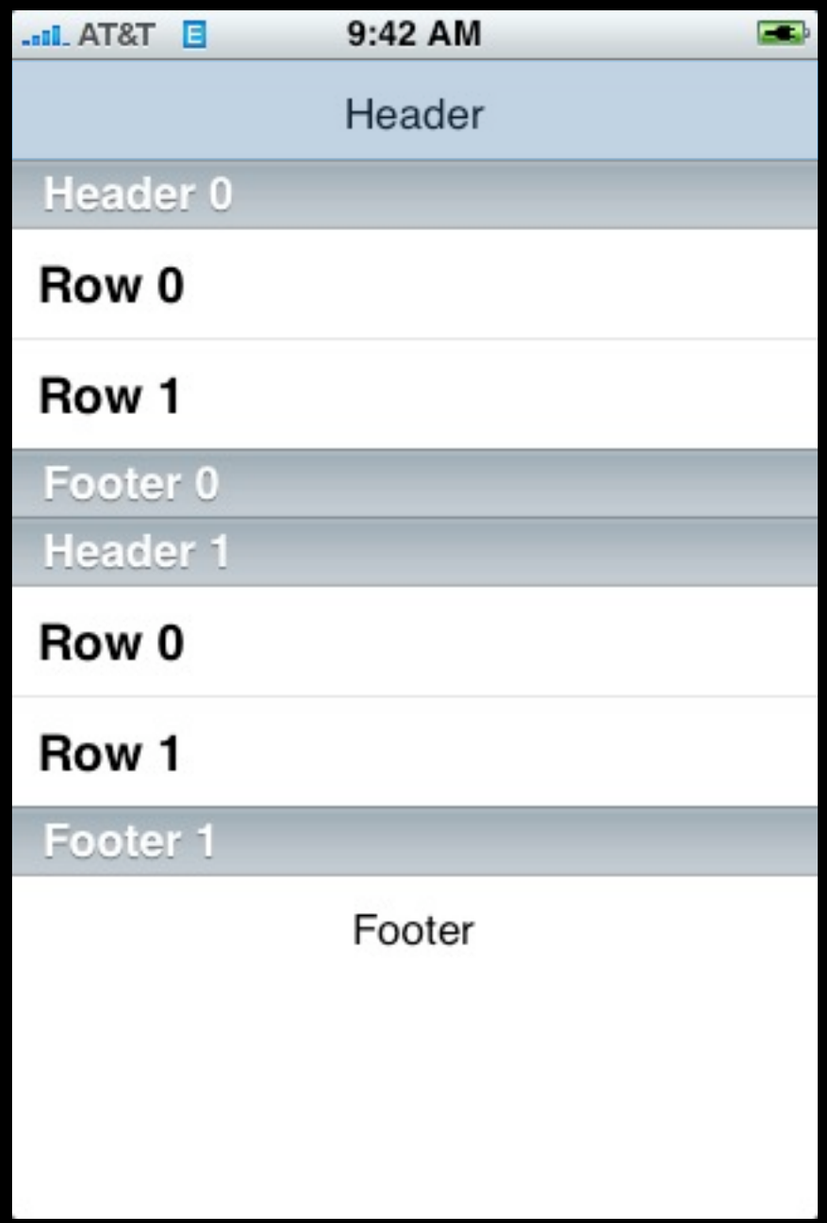


Table View Anatomy

Plain Style

Table Header

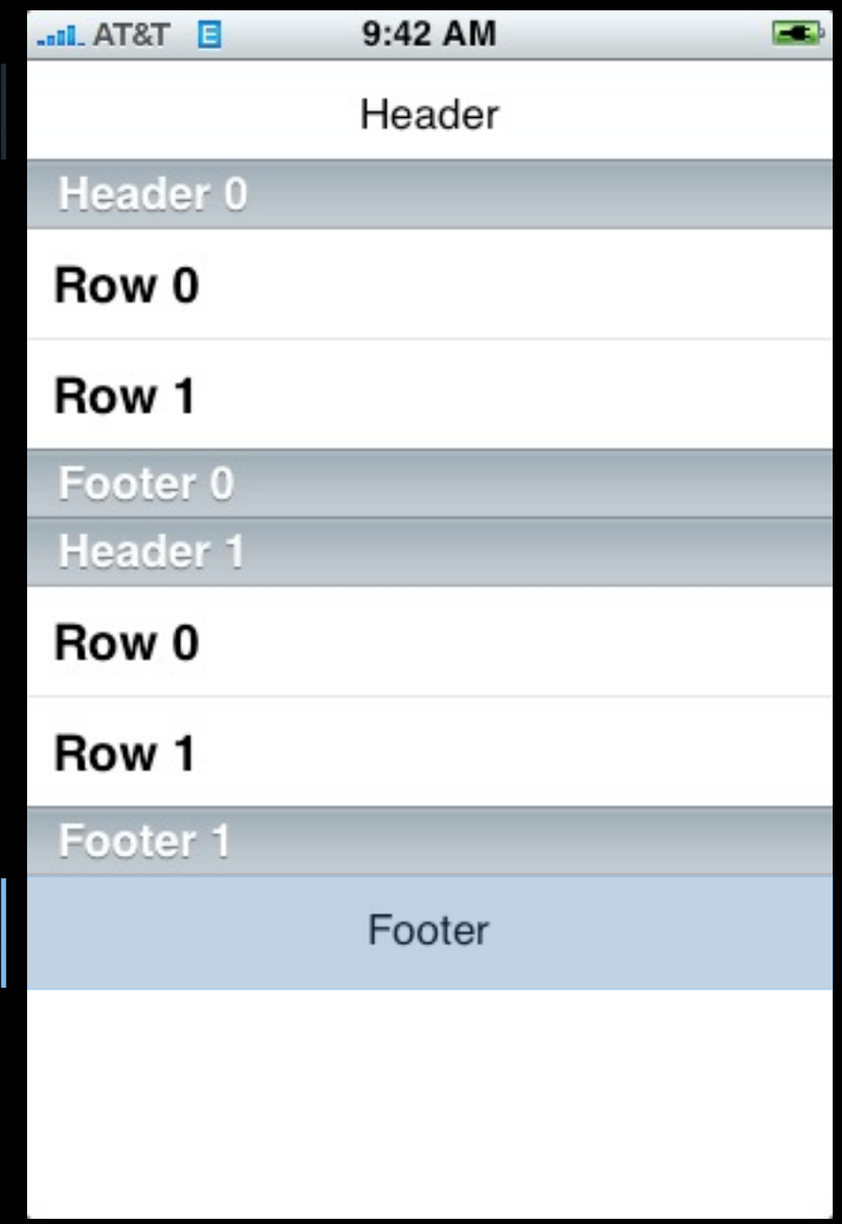


Table Footer

Table View Anatomy

Plain Style

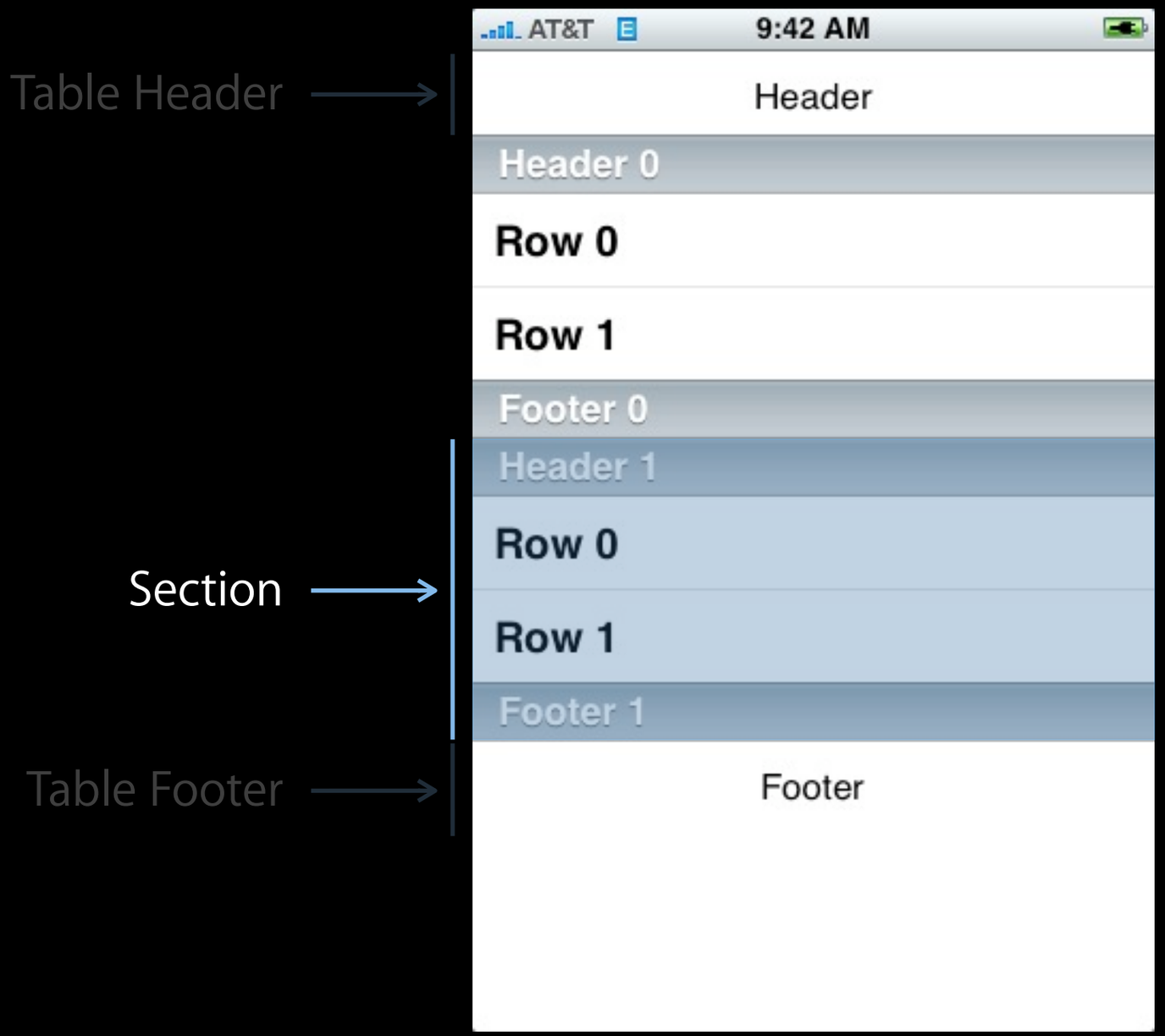


Table View Anatomy

Plain Style

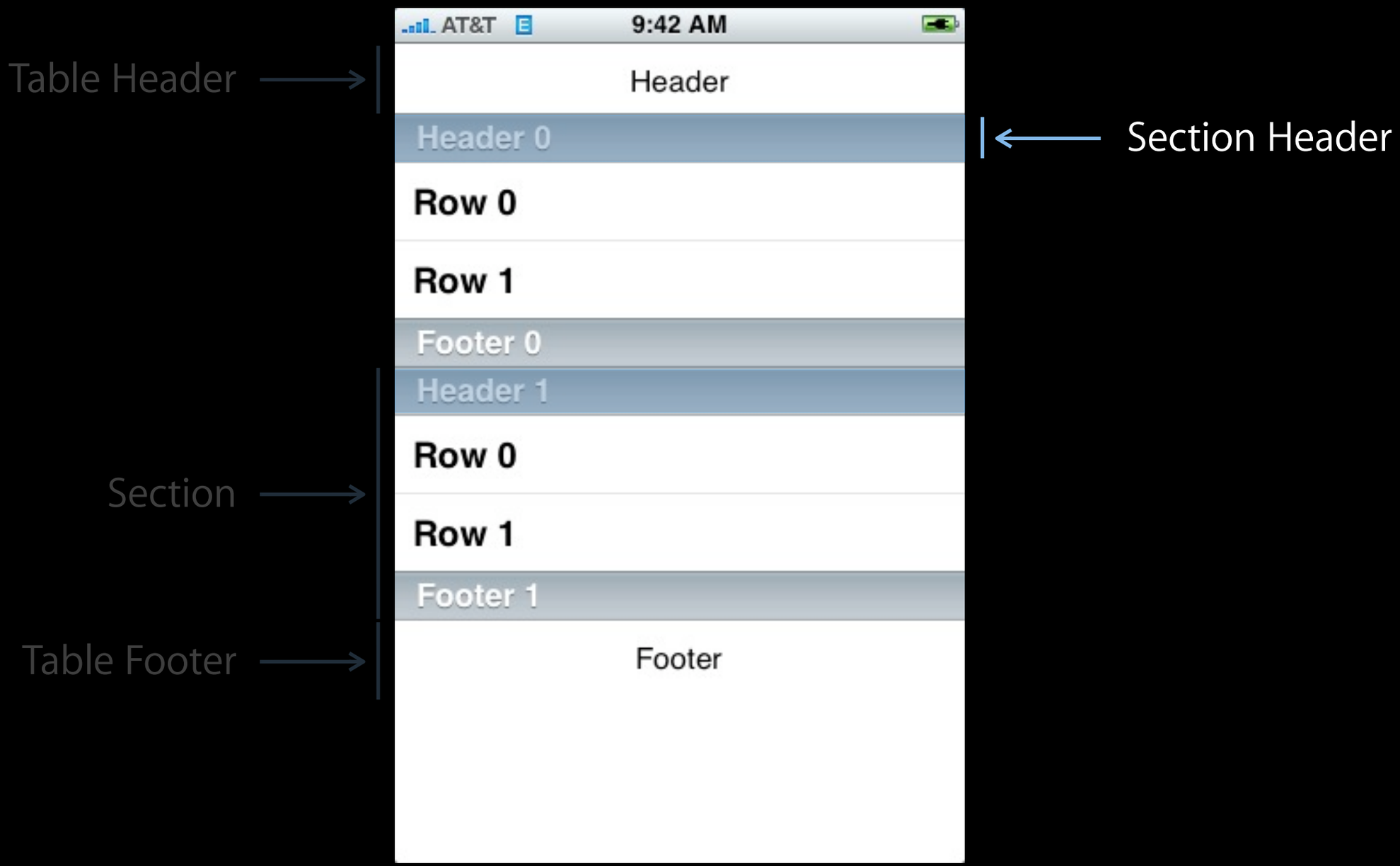


Table View Anatomy

Plain Style

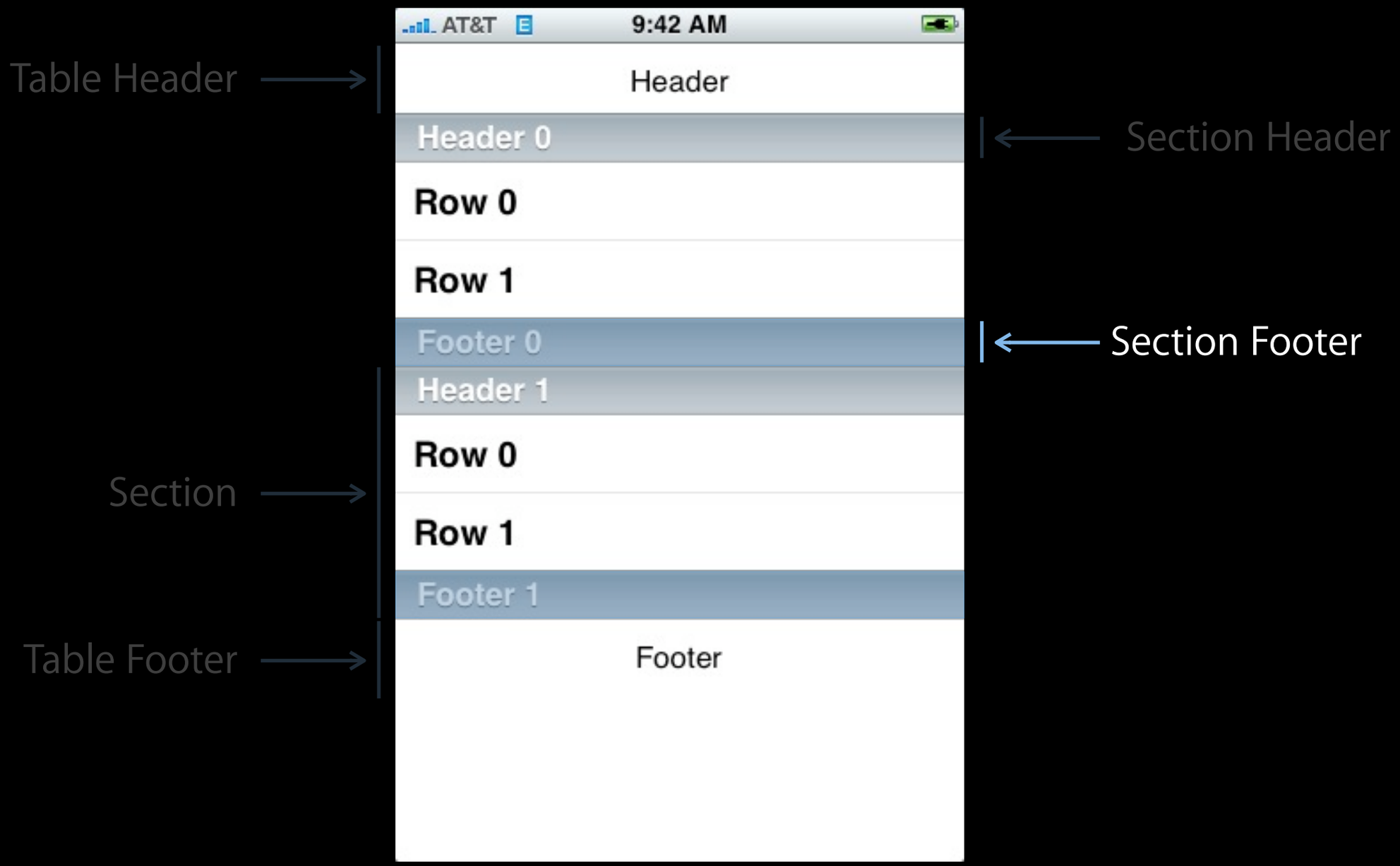


Table View Anatomy

Plain Style

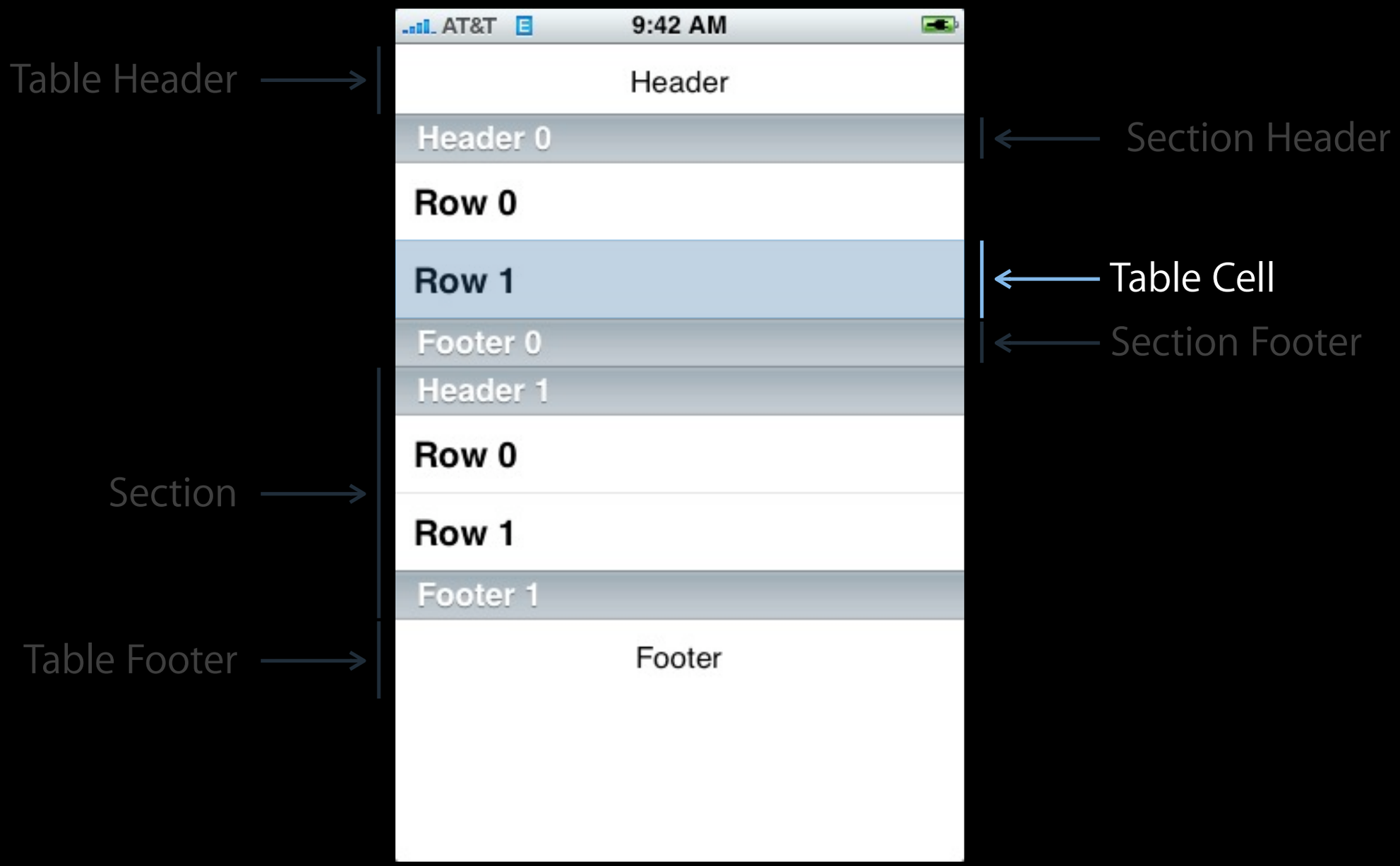


Table View Anatomy

Plain Style

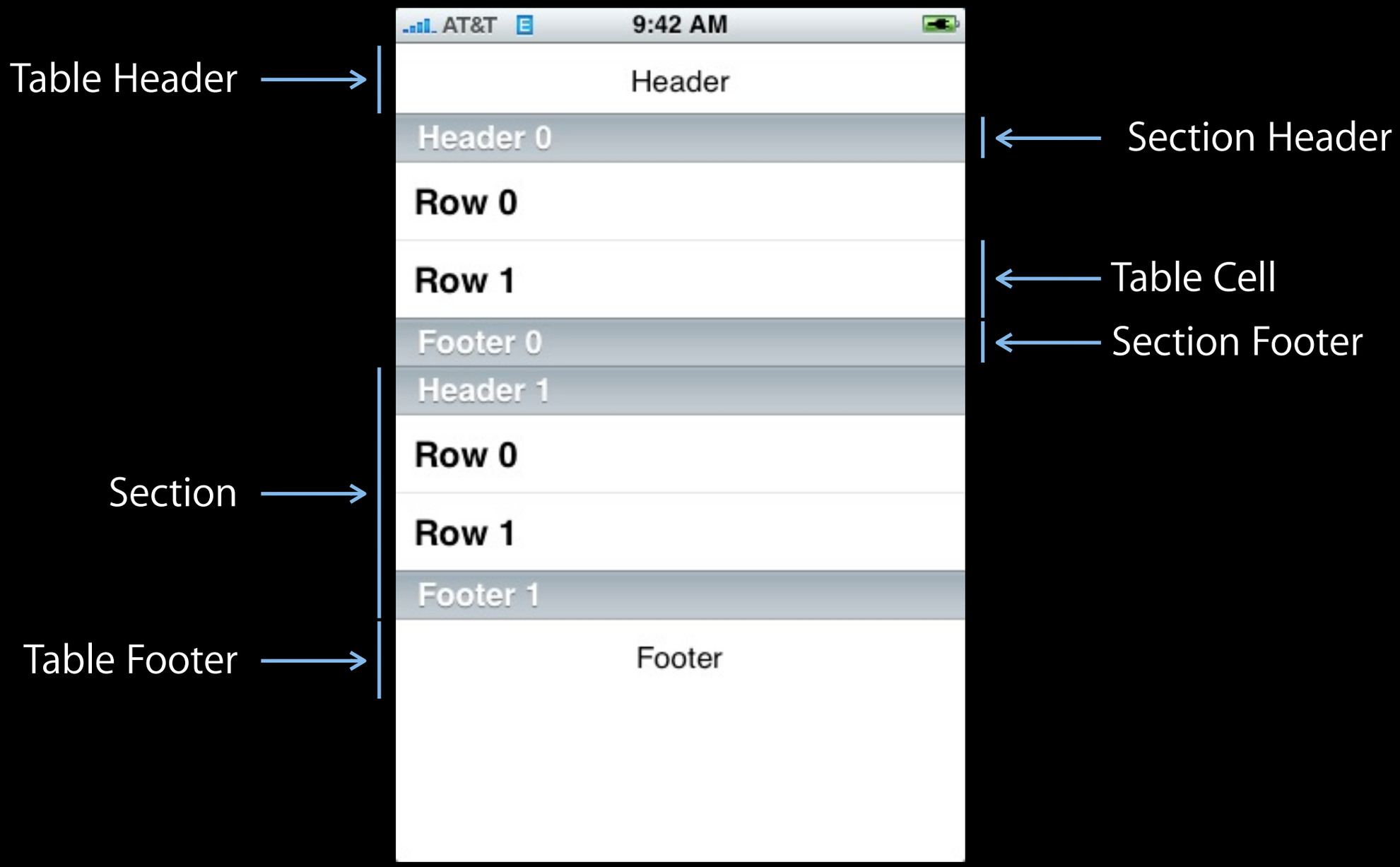
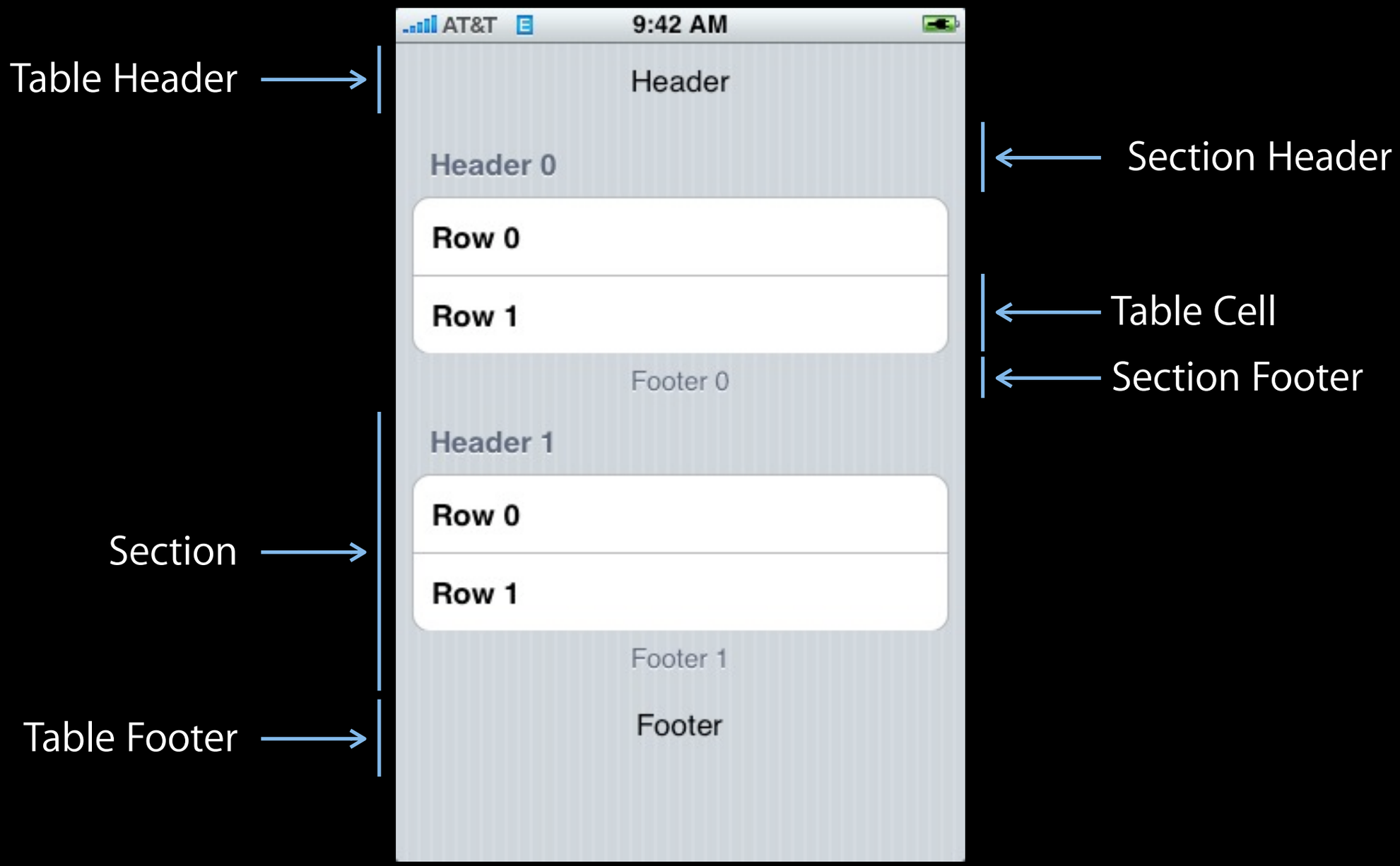


Table View Anatomy

Grouped Style



Using Table Views

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

Displaying Data in a Table View

A Naïve Solution

- 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

- Provide number of sections and rows

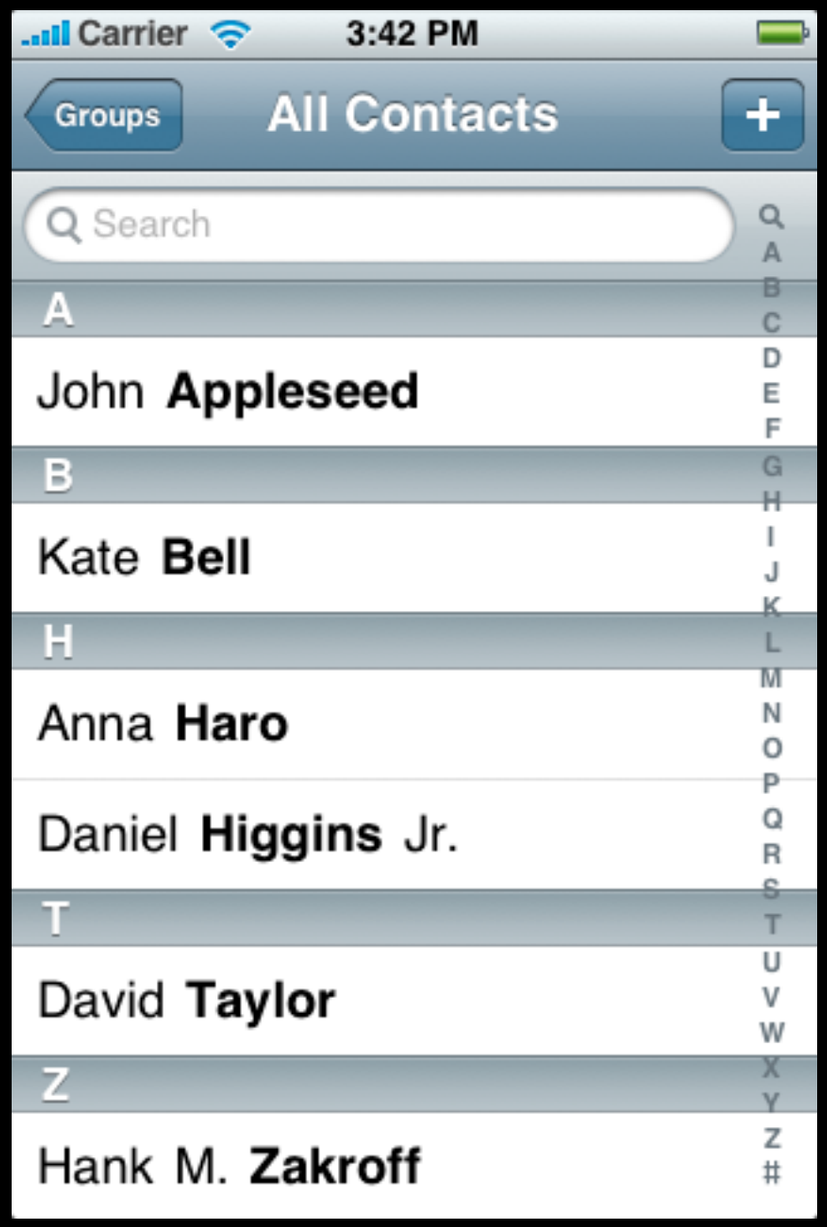
```
// Optional method, defaults to 1 if not implemented
- (NSInteger)numberOfSectionsInTableView:(UITableView *)table;

// 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

`numberOfSectionsInTableView:`

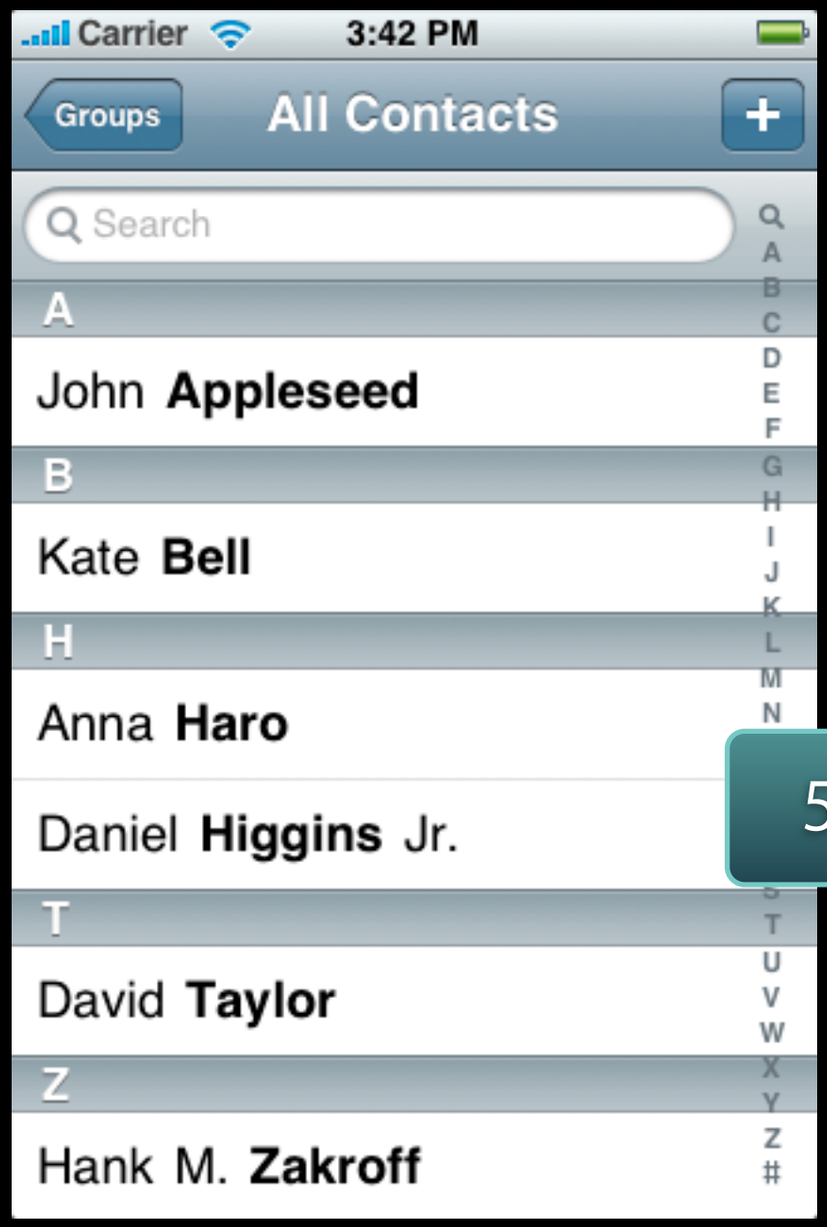


How many sections?

Datasource

Datasource Message Flow

`numberOfSectionsInTableView:`

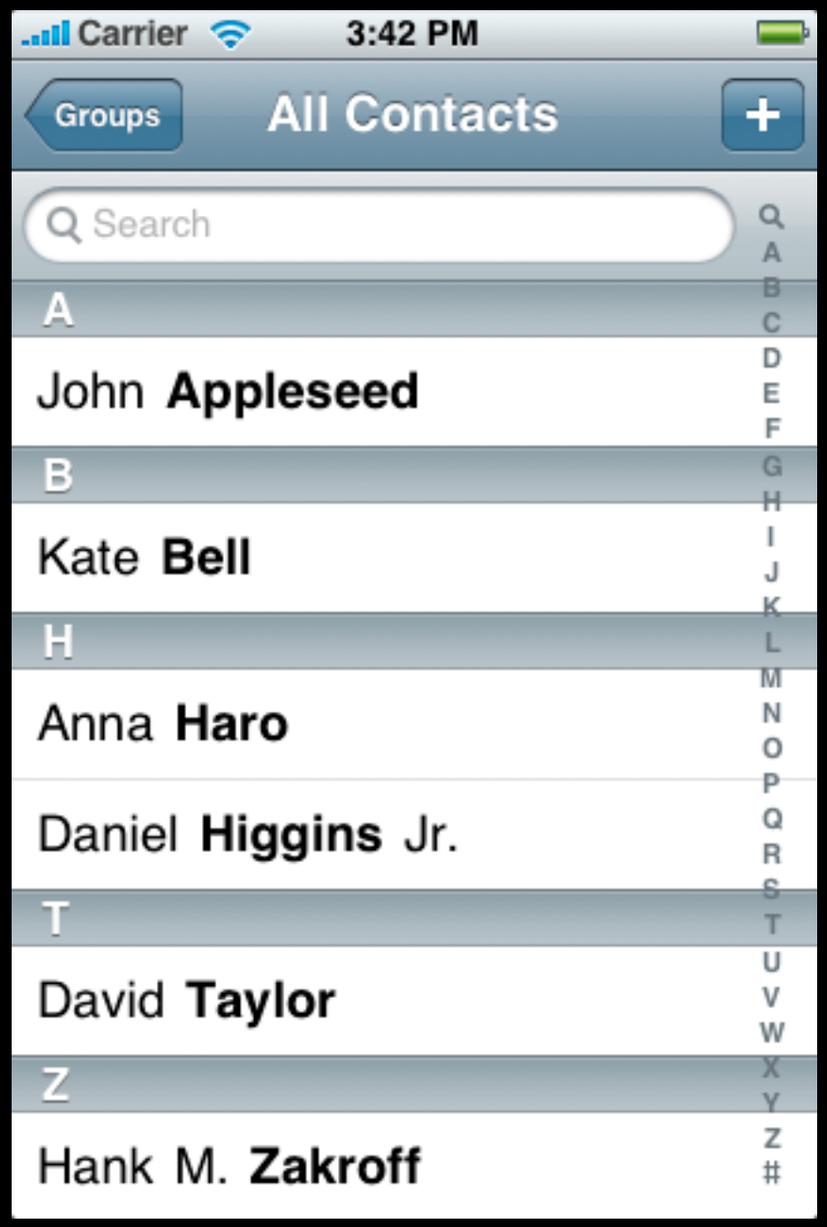


5

Datasource

Datasource Message Flow

`tableView:numberOfRowsInSection:`

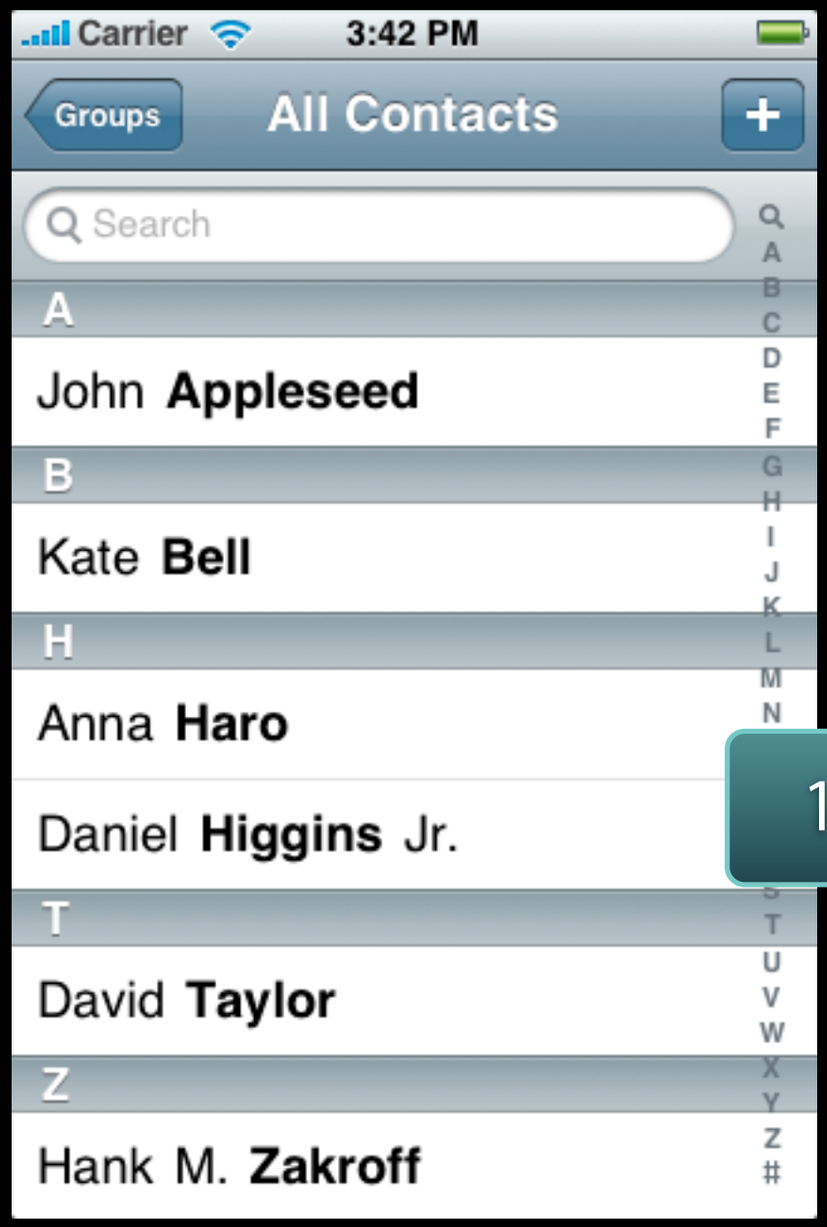


How many rows in section 0?

Datasource

Datasource Message Flow

`tableView:numberOfRowsInSection:`

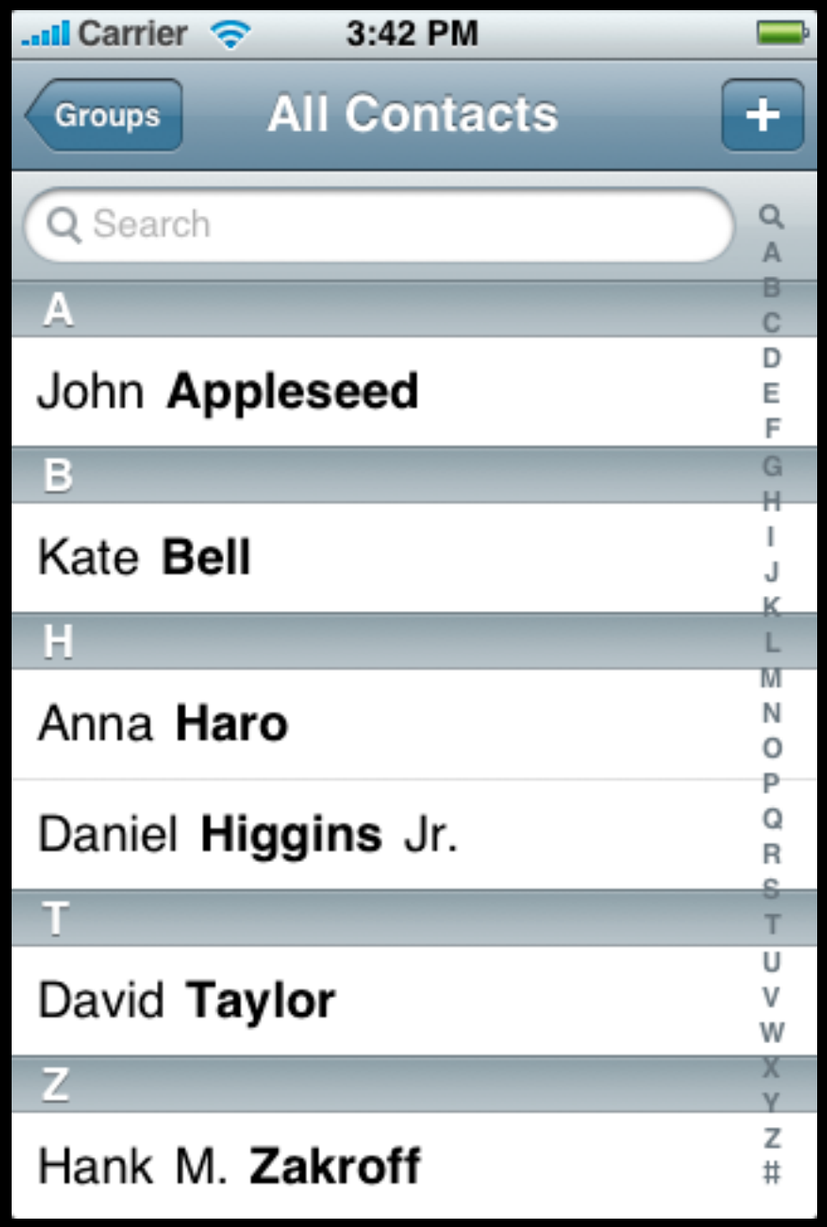


1

Datasource

Datasource Message Flow

`tableView:cellForRowAtIndexPath:`

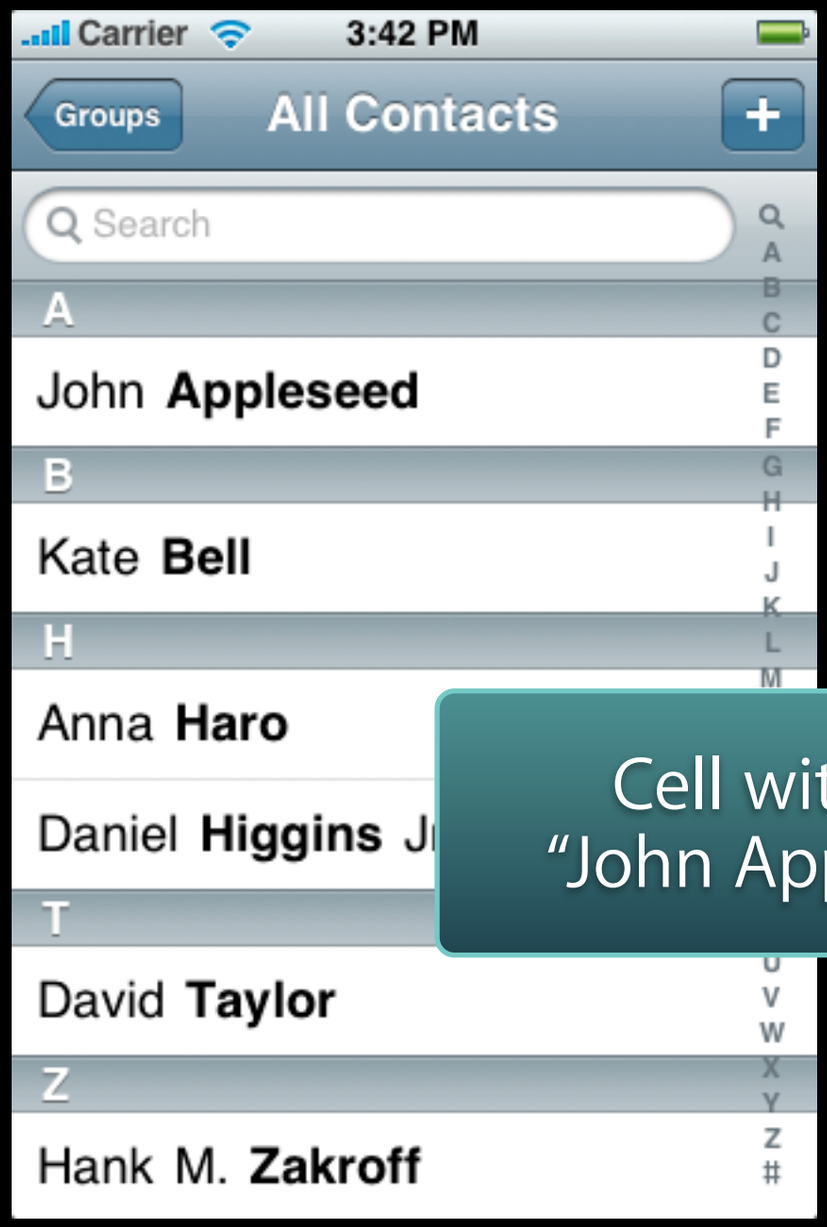


What to display at section 0, row 0?

Datasource

Datasource Message Flow

`tableView:cellForRowAtIndexPath:`

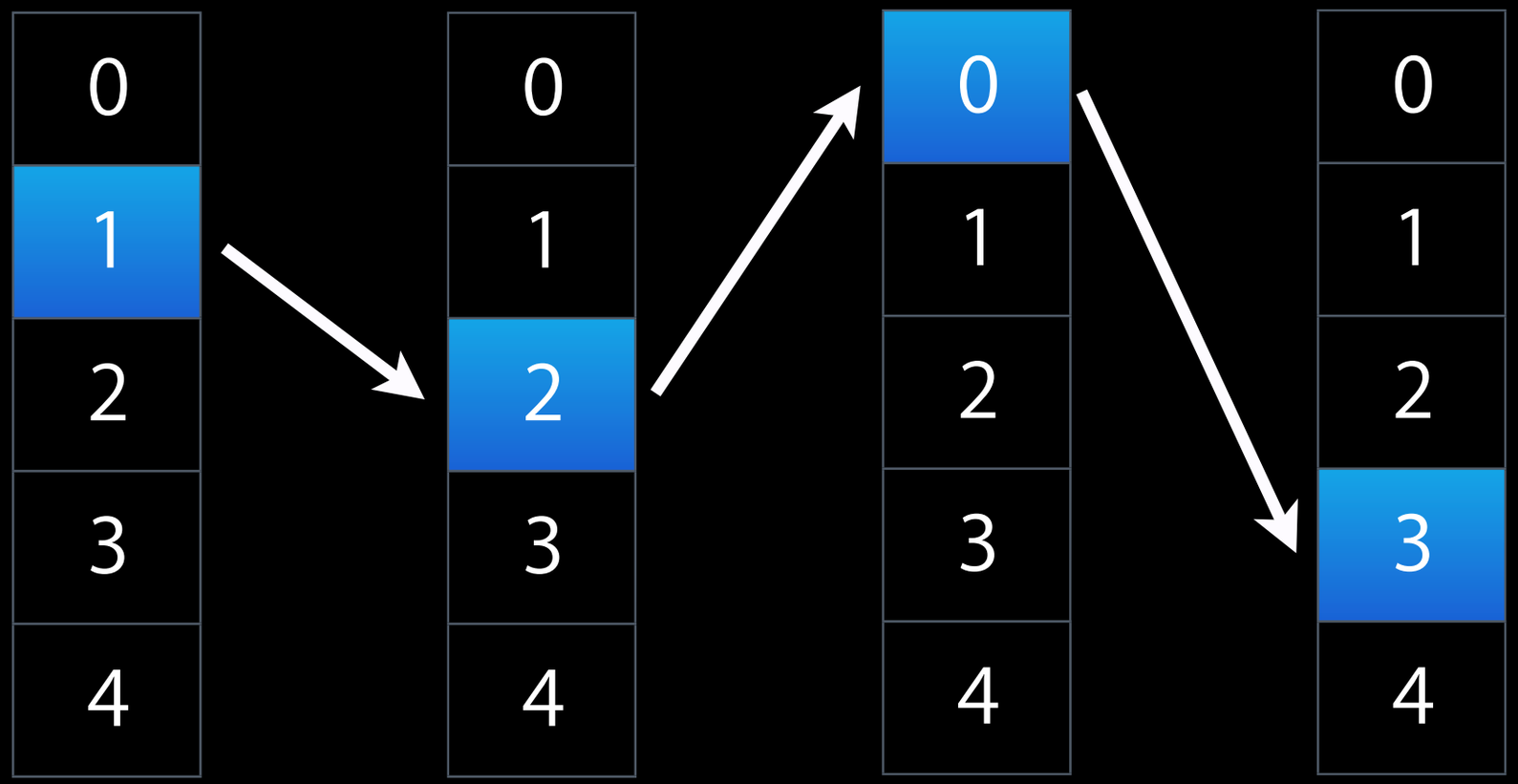


Cell with text
“John Appleseed”

Datasource

NSIndexPath

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



NSIndexPath 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

- 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.text = [myStrings objectAtIndex:indexPath.row]
  return [cell autorelease];
}
```

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]
    initWithFrame:... reuseIdentifier:@”MyIdenifier”];
    }

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

    return cell;
}
```

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];
}
```

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

- Customize appearance of table view cell

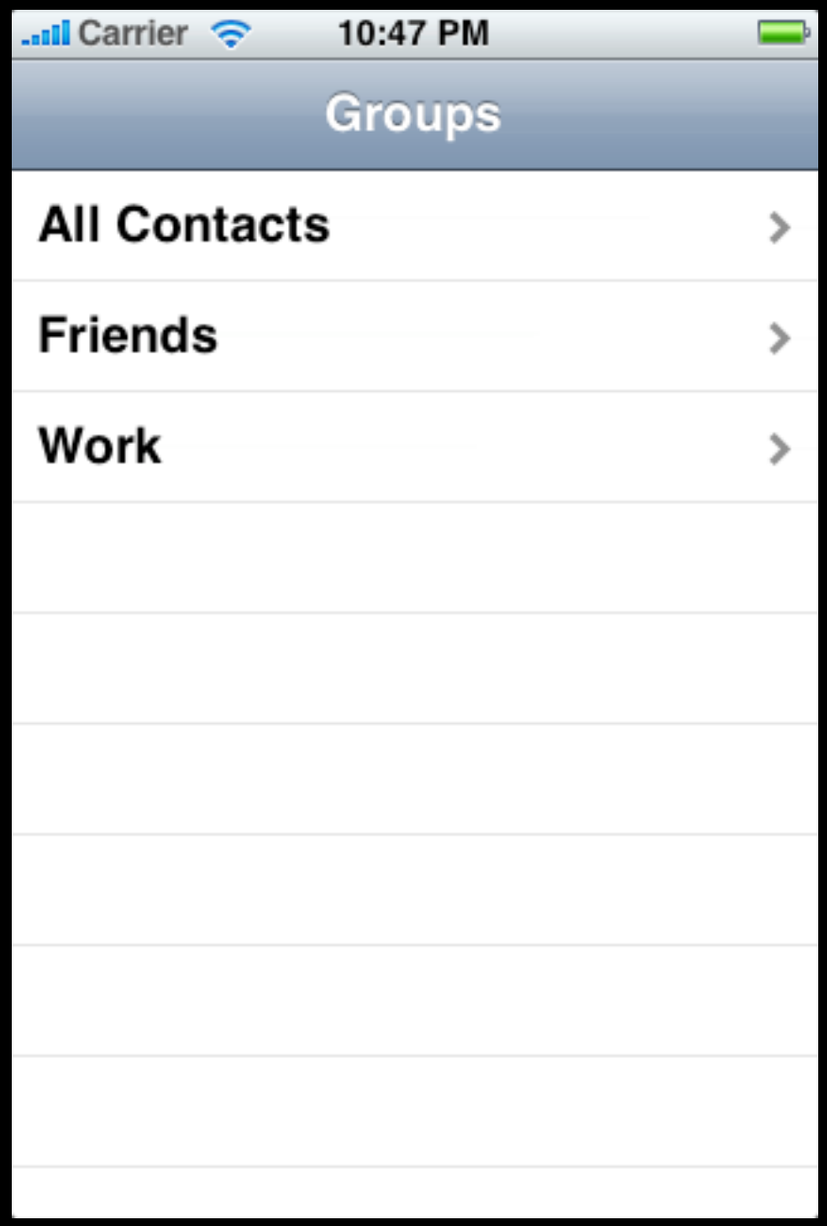
```
- (void)tableView:(UITableView *)tableView
  willDisplayCell:(UITableViewCell *)cell
forRowAtIndexPath:(NSIndexPath *)indexPath;
```

- 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



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

- 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

Demo: UITableViewController

Table View Cells

Basic properties

- UITableViewCell has image and text properties

```
cell.image = [UIImage imageNamed:@“obama.png”];
cell.text = @“Barack Obama”;
```

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**

```
- (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];
}
```

Custom Row Heights

- Rows in a table view may have variable heights
- NSString category in **UIStringDrawing.h** is very useful for computing text sizes

```
- (CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    NSString *text = ...;
    UIFont *font = [UIFont systemFontOfSize:...];
    CGSize withinSize = CGSizeMake(tableView.width, 1000];

    CGSize size = [text sizeWithFont:font
                   constrainedToSize:withinSize
                       lineBreakMode:UILineBreakModeWordWrap];

    return size.height + somePadding;
}
```

Questions?

Presence 2

- Due next Tuesday 5/5 at 11:59PM
  - Displaying dynamic data with table views
  - Fetching data over the Internet using web services

Demo:
Presence - Part 2