CS193P - Lecture 8

iPhone Application Development

Scroll Views & Table Views

スクロールビューとテーブルビュー

これは iPad です

iPhone SDK 3.2

- Support for iPad
- Beta Release

iPad のアプリ開発もできます
（いまは Beta ではなく本物です）

iPhone SDK 3.2

- Support for iPad
- Beta Release

（いまは（たぶん）University Program でも iPad 開発ができます）

Enrollment in iPhone Developer Standard or Enterprise required

Announcements

- Paparazzi 1 due next Wednesday (2/3)

Today's Topics 今日のトピックス

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

スクロール ビュー

テーブル ビュー
表示データについて
外見と応答の調整

テーブル ビュー コントローラ

テーブル ビュー の セル

Scroll Views

スクロール ビュー

UIScrollView

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

- スクリーンからハミ出すようなコンテンツを表示できる
- ジェスチャ（タッチ）により見渡したりズームしたりできる
- 重要なサブクラス：UITableView と UITextView

Scrolling Examples

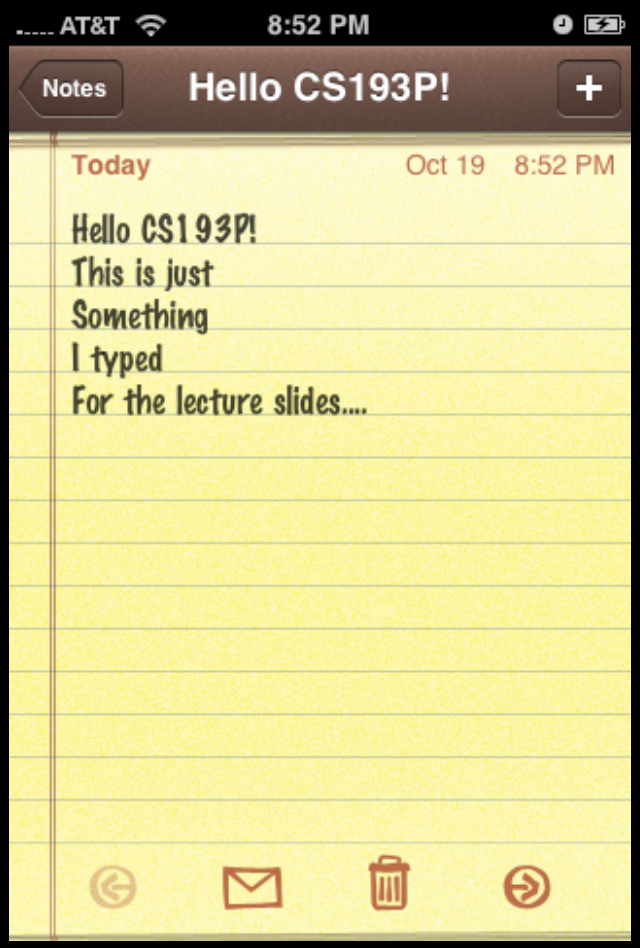
スクロールの例

地図

Scrolling Examples

写真アルバム

Scrolling Examples



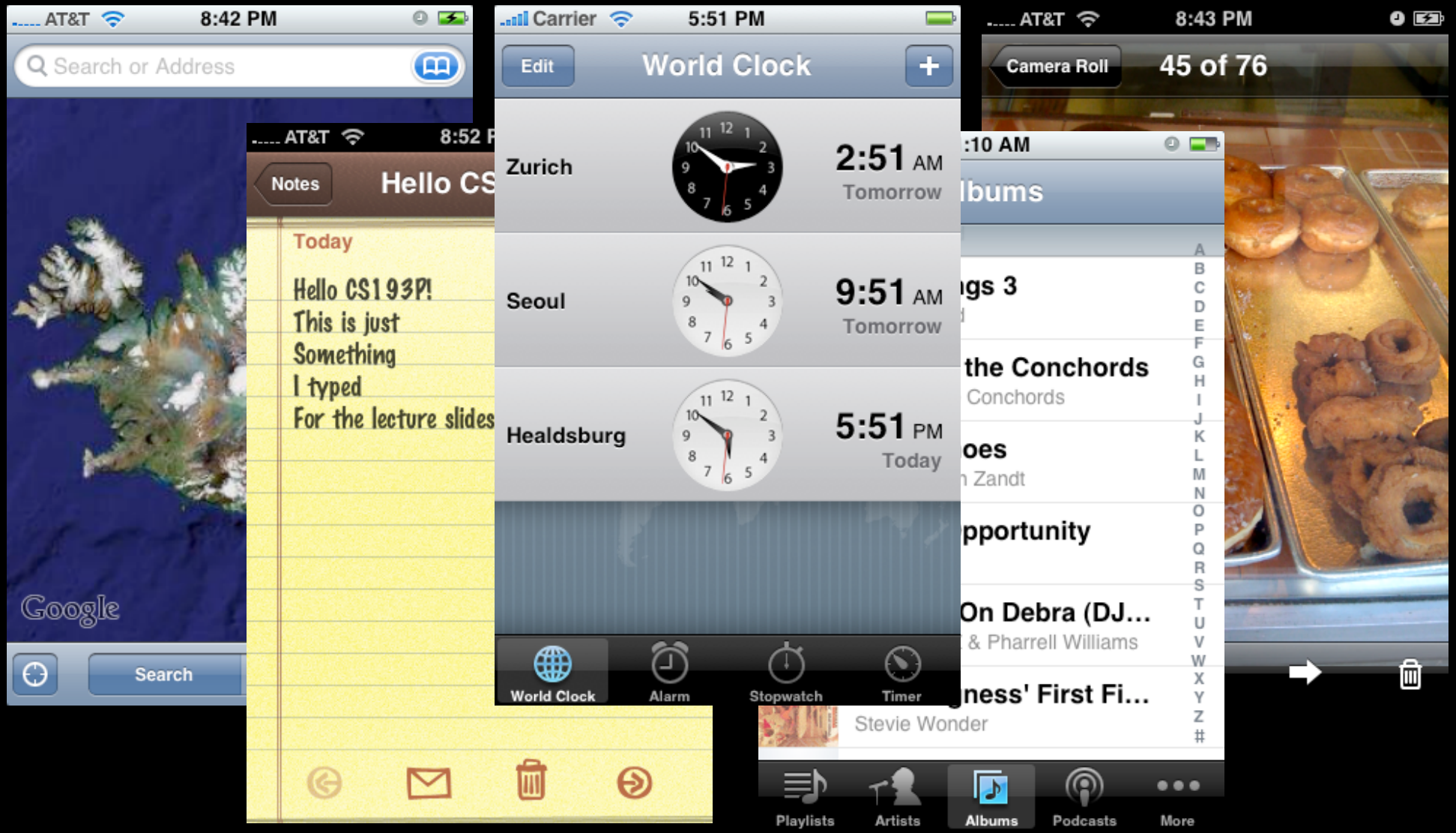
メモ帳

Scrolling Examples

音楽ソフト（アイコン付）

Scrolling Examples

時計（Subview が入る）

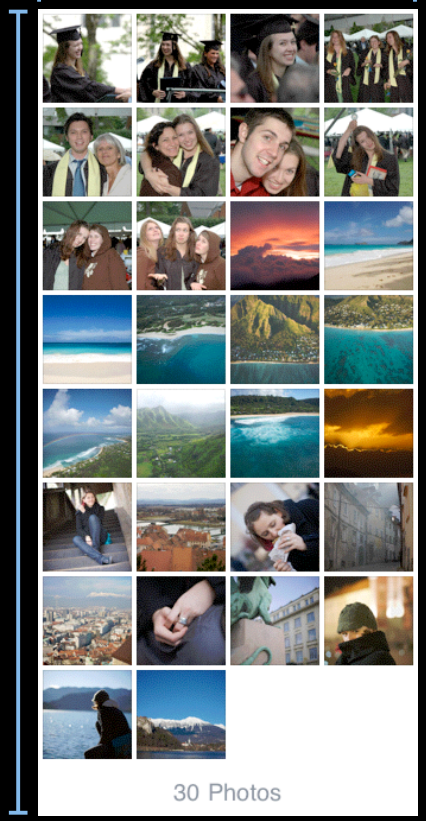


Content Size

表示内容のサイズ

幅

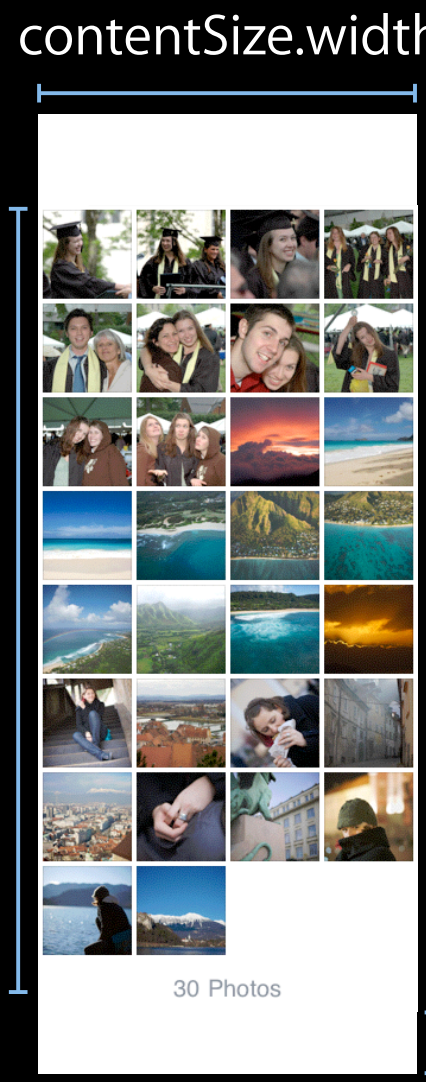
contentSize.width



contentSize.height

高さ

Content Inset



インセット（トップ）
「余白」みたいなー

インセットは左右にも
つけられる

インセット（ボトム）

Content Inset インセット（余白）は何か

Content Inset

画面いっぱいの
写真アルバム

Content Inset

ステータス バー
ナビゲーション バー

Content Inset

contentInset.top ⟶

インセット（トップ）

Content Inset

contentInset.top ⟶

ツールバー（らしきもの）

Content Inset

contentInset.top ⟶

contentInset.bottom ⟶

インセット（ボトム）

Scroll Indicator Insets

垂直スクロールならば
右側に「インディケータ」

水平スクロールならば
下側に「インディケータ」

Scroll Indicator Insets

scrollIndicatorInsets.top

インディケータにも
インセットがある

Content Offset

表示内容の「オフセット」

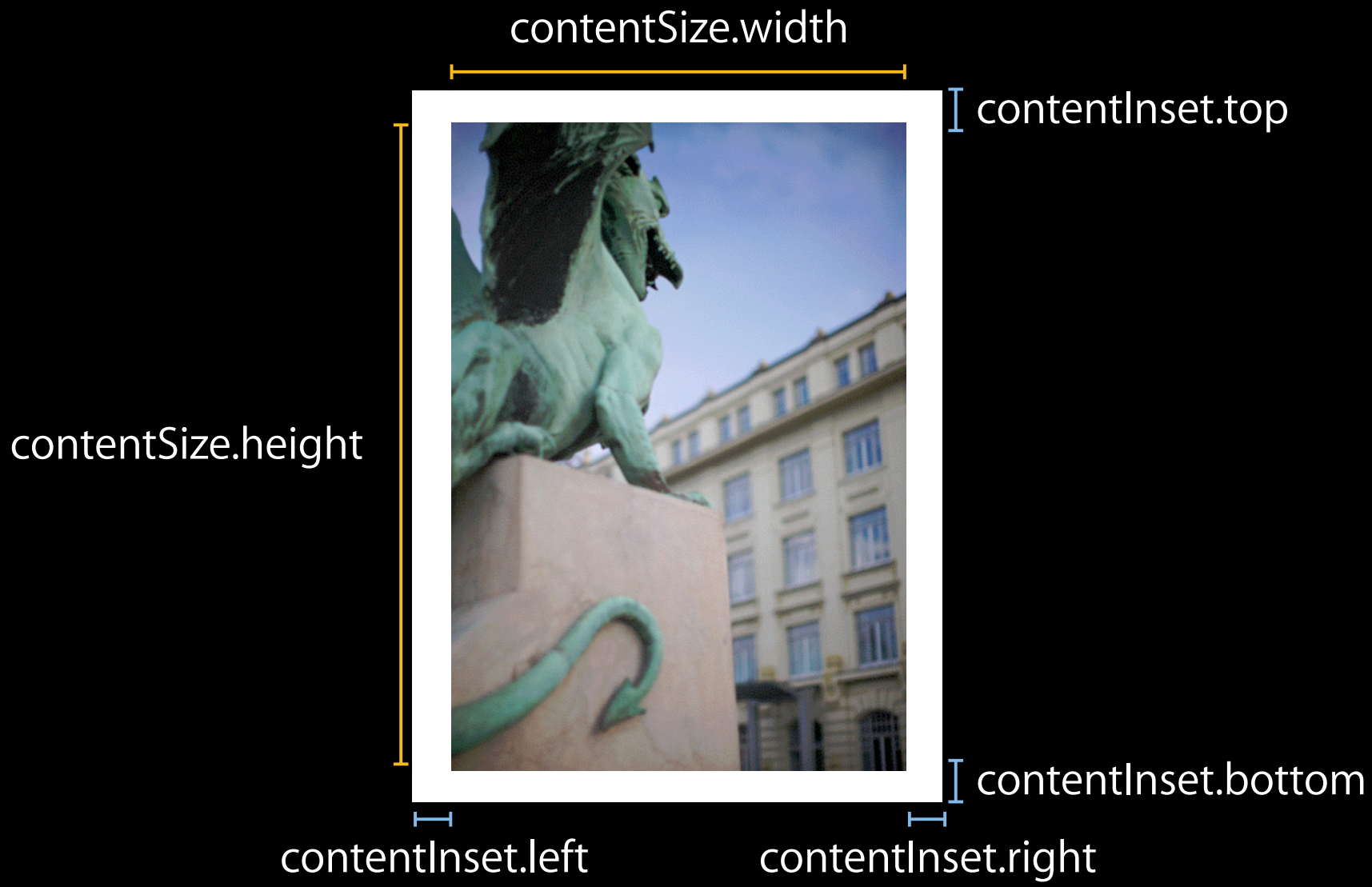
Content Offset

表示内容と
実際のスクリーンの
関係は，こんな感じ．

Content Offset

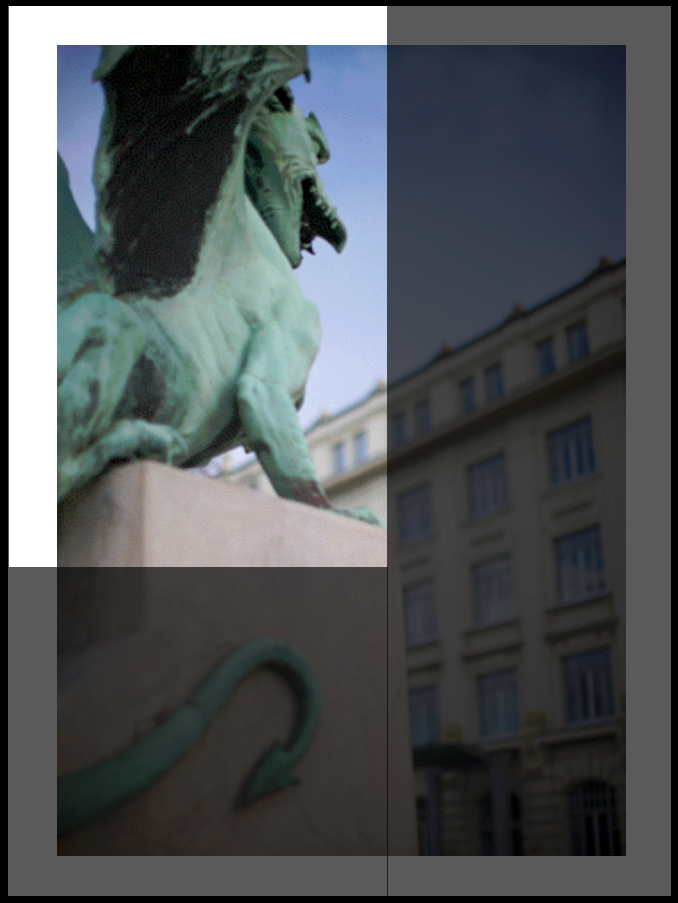
これが
オフセット

表示内容のサイズとインセット



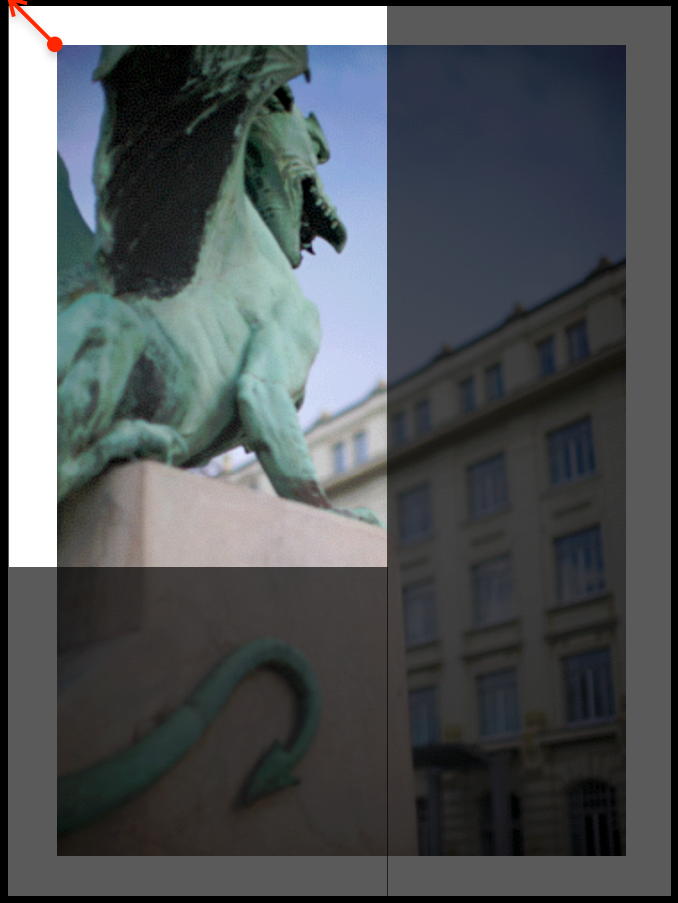
実際に表示されるスクリーンのオフセット

インセットの端ではなく
表示内容（Content）の端から
オフセットをとることに注意



contentOffset.x
(-contentInset.left)

contentOffset.y
(-contentInset.top)



ただし，オフセットは「負」になることがある．
（余白部分に入り込んだとき）

Using a Scroll View スクロール ビュー の使い方

- Create with the desired frame スクロールビューを（長方形領域に）生成する

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

- Add subviews (frames may extend beyond scroll view bounds) 表示するサブビューを入れる

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

ここでは ImageView

- Set the content size 表示内容の大きさを伝える

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

Extending Scroll View Behavior 応答を拡張する

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

・アプリはスクロール（ジェスチャ）イベントを監視したい
  ・（強制的に）オフセットが変更されたとき
  ・ドラッギングが開始・終了したとき
  ・減速が開始・終了したとき

これをどのように実現すればよいか…

（従来の方法）サブクラスを作って拡張する

Extending with a Subclass

サブクラスをつくり
メソッドを上書きして
動作をカスタマイズ

- 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

このアプローチのもつ問題：

- 動作ロジック (C) がビュー (V) の一部に入ってしまう．
- スクロールビューごとにサブクラスを書くのは面倒くさい．
- スーパークラスに大きく依存したプログラムになってしまう．

（新しい方法）デリゲーション（委譲）

Extending with Delegation

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

デリゲート（委譲先）は
別オブジェクト

「責任」を明示した上で
動作をカスタマイズ
外見をカスタマイズ

拡張されるオブジェクトと「弱くカップリング」する

UIScrollView Delegate

（参考までに…）
プロトコルの宣言は…

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

従属するクラス名

```
@optional

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

スクロールしたときに呼ばれる

```
...

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

ステータスバーをタップしたとき
上端までスクロールするか否か

```
@end
```

デリゲート（委譲先）をつくるには

Implementing a Delegate

- Conform to the delegate protocol

どのプロトコルに従うか

```
@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 ...) {

    }
}
```

必須メソッドはすべて，
任意メソッドは必要に応じて，
実装していく

拡大・縮小

Zooming with a Scroll View

- Set the minimum, maximum, initial zoom scales

どこまで拡大できるか

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

どこまで
縮小できるか

- Implement delegate method for zooming

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

拡大・縮小されたときに
表示すべき（任意サイズの）ビュー
（自動的にサイズ調整される）

Set Zoom Scale （強制的に）倍率を指定するには

```
- (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 与えられた長方形領域を表示

```
- (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);
```

Table Views

テーブル ビュー

Table Views

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

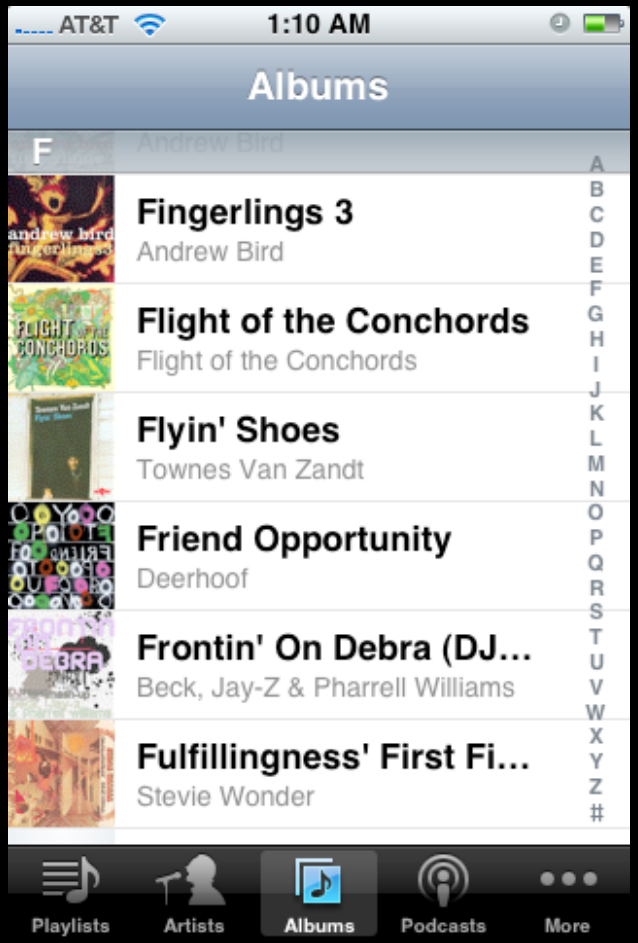
たくさんのコンテンツのリストを表示

単一カラム（縦１列）
垂直にスクロール
大量のデータセット

iPhone アプリでよく使われる
強力なビュー

Table View Styles スタイルの違い

プレーン
UITableViewStylePlain



グループ
UITableViewStyleGrouped

解剖

Table View Anatomy

Plain Style

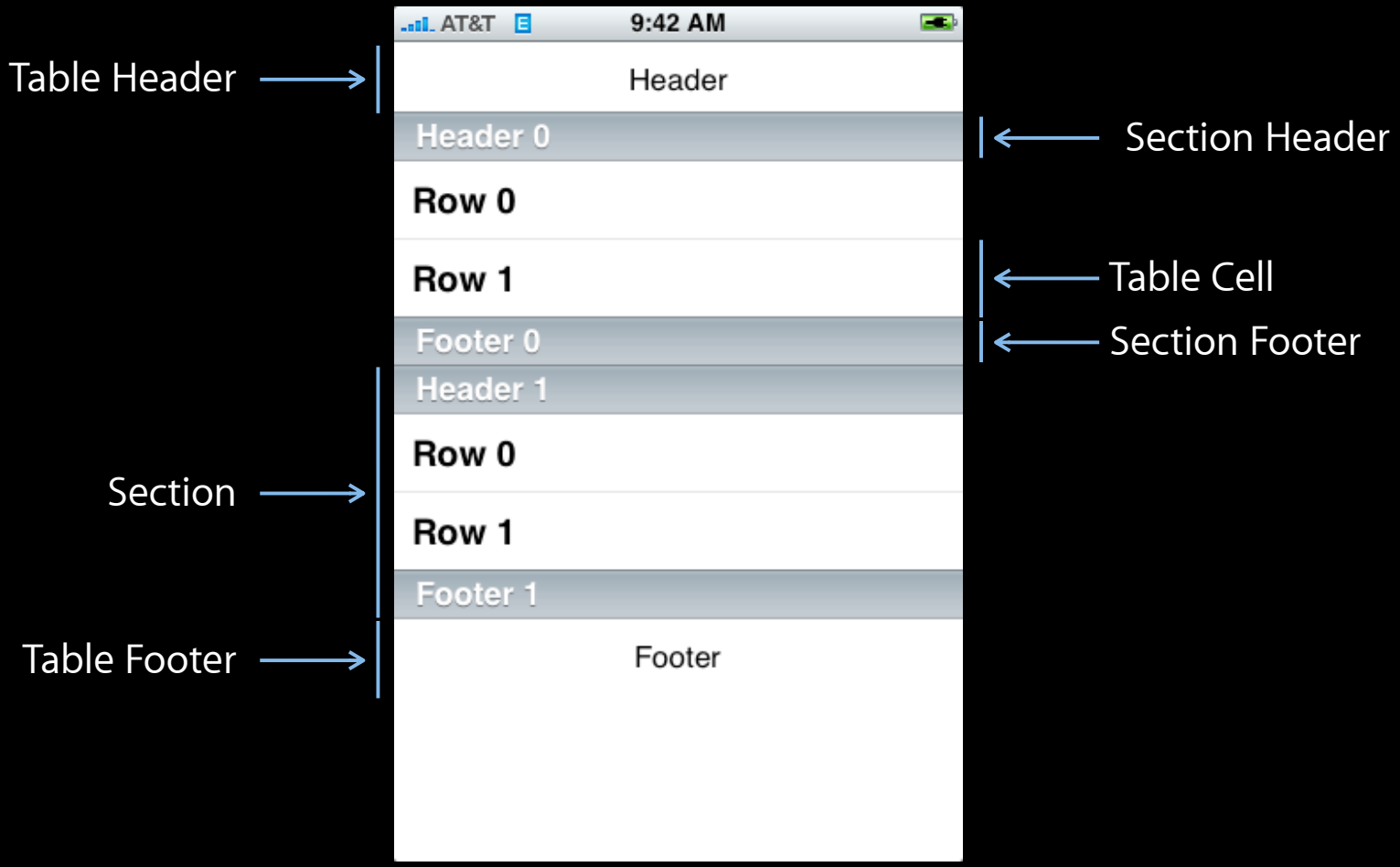
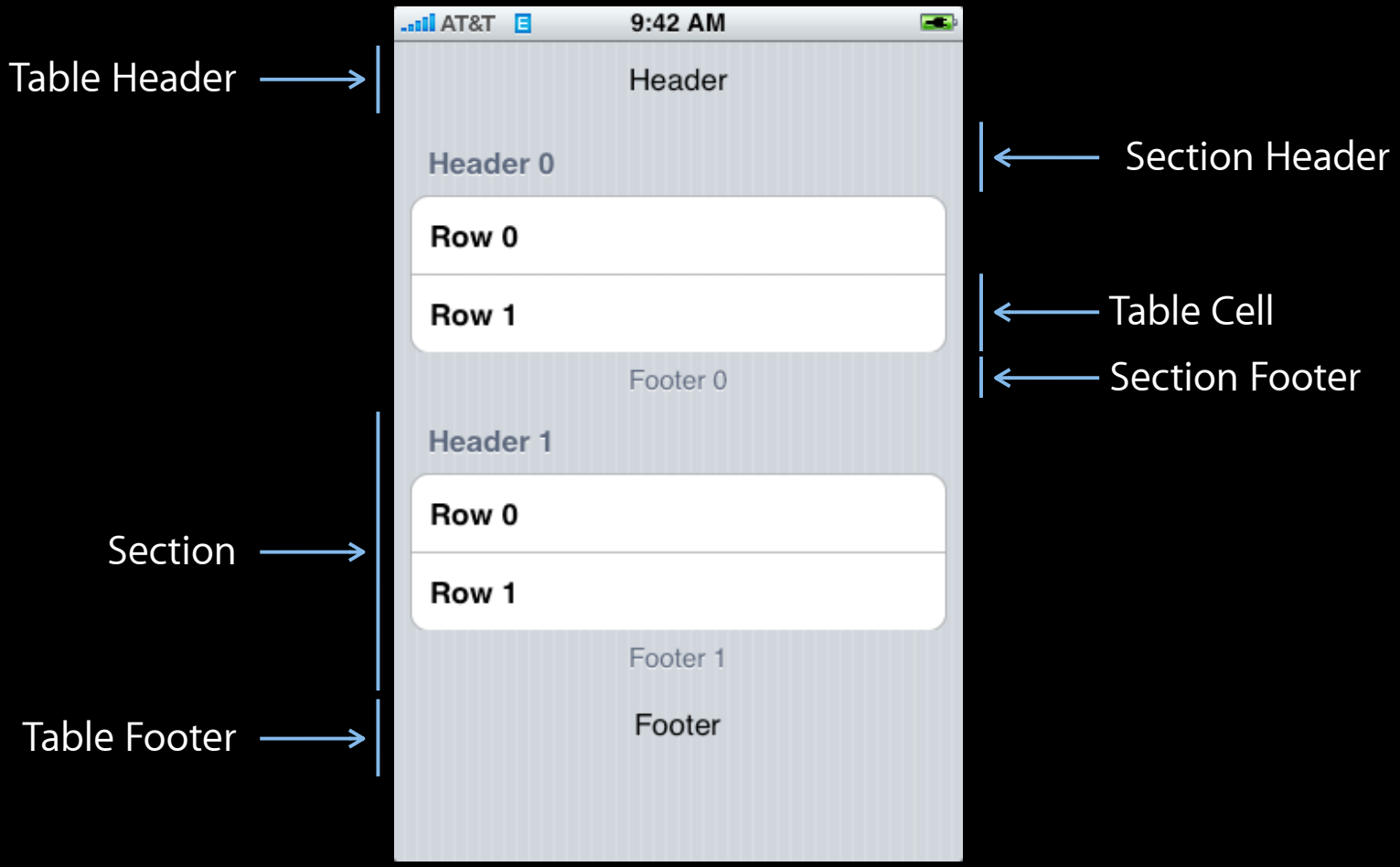


Table View Anatomy

Grouped Style



Using Table Views

- Displaying your data in the table view　データの表示
- Customizing appearance & behavior　外見と動作のカスタマイズ

Displaying Data in a Table View

データの表示

A Naïve Solution ナイーブな解決法

- Table views display a list of data, so use an array
  `[myTableView setList:myListOfStuff];` 配列で与える？
- **Issues with this approach**
  - All data is loaded upfront
  - All data stays in memory

その問題点：
- 最初に全データのロードが必要
- 全データがメモリを占拠

A More Flexible Solution より柔軟な解決法

- Another object provides data to the table view データ供給オブジェクトを使う
  - Not all at once 一度に全部ではなく，
  - Just as it's needed for display 表示する必要に応じて
- Like a delegate, but purely data-oriented

UITableViewDataSource データ供給オブジェクト

- Provide number of sections and rows

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

セクションの数を返す（上書きしなければ１）

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

指定セクション内の行数を返す

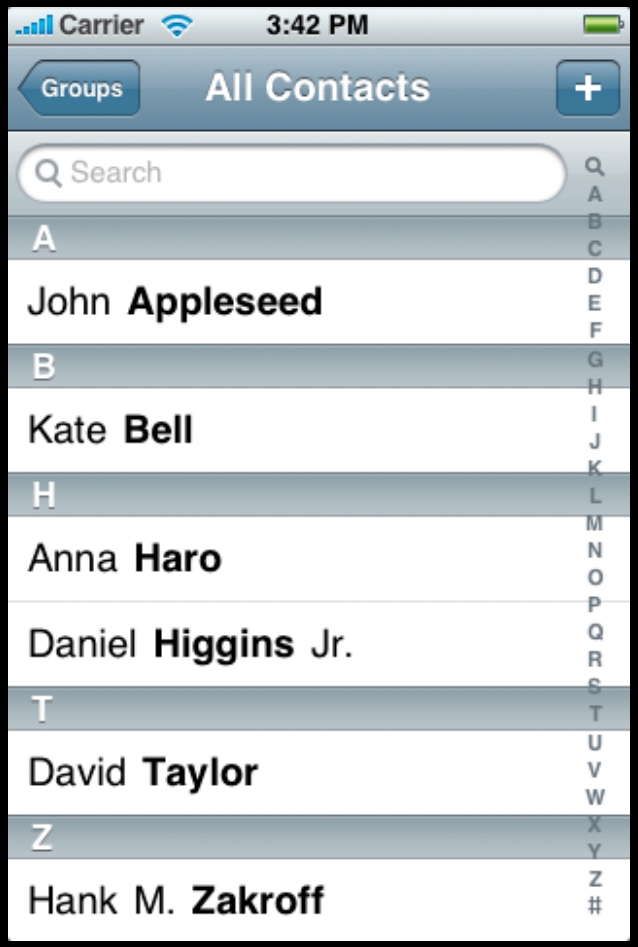
- Provide cells for table view as needed

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

指定セクションの指定した行についてデータ（セル）を返す

Datasource Message Flow

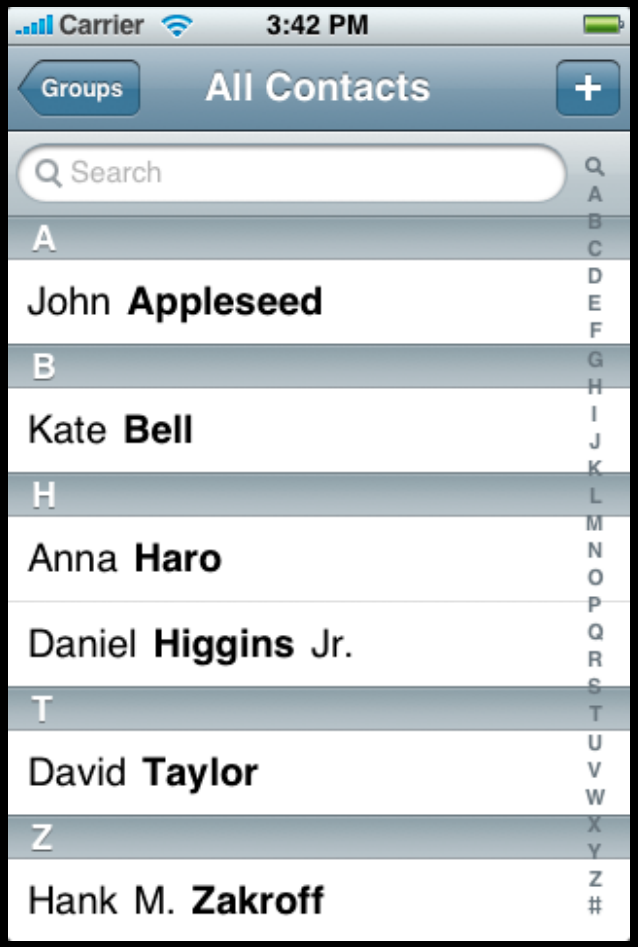
データ供給オブジェクトのメッセージの流れ



Datasource

Datasource Message Flow

`numberOfSectionsInTableView:`

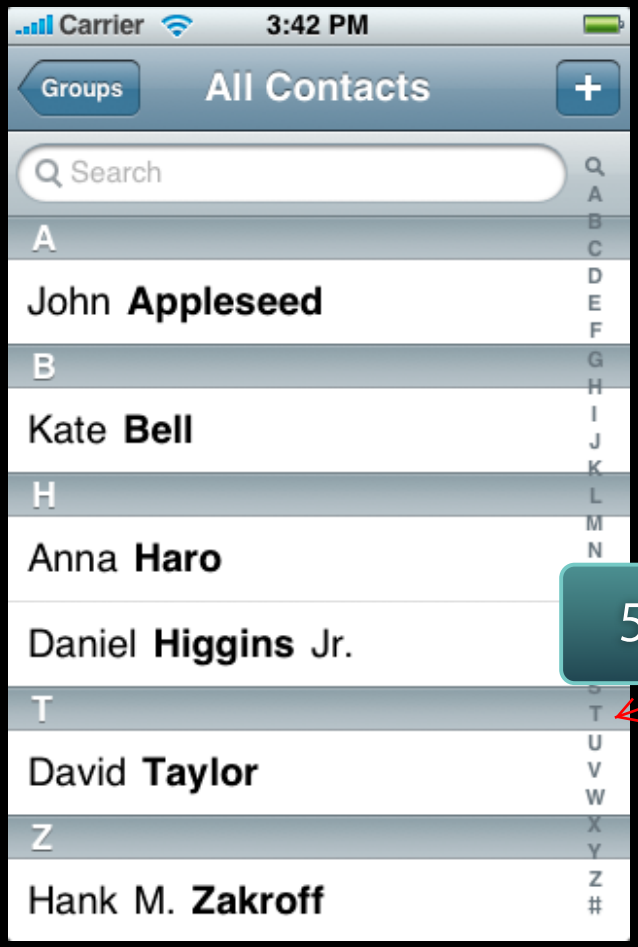


How many sections?

Datasource

Datasource Message Flow

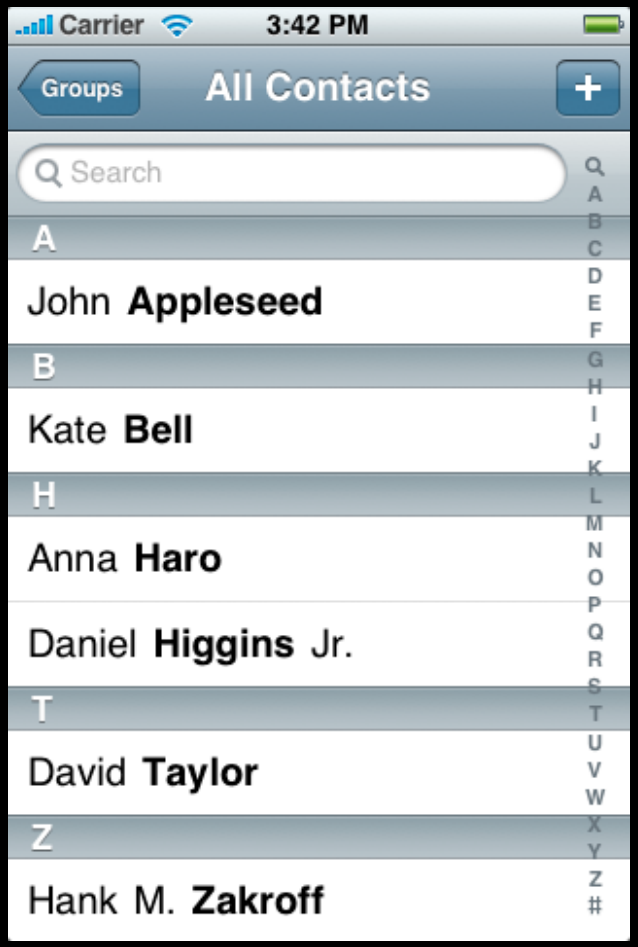
`numberOfSectionsInTableView:`



5

Datasource

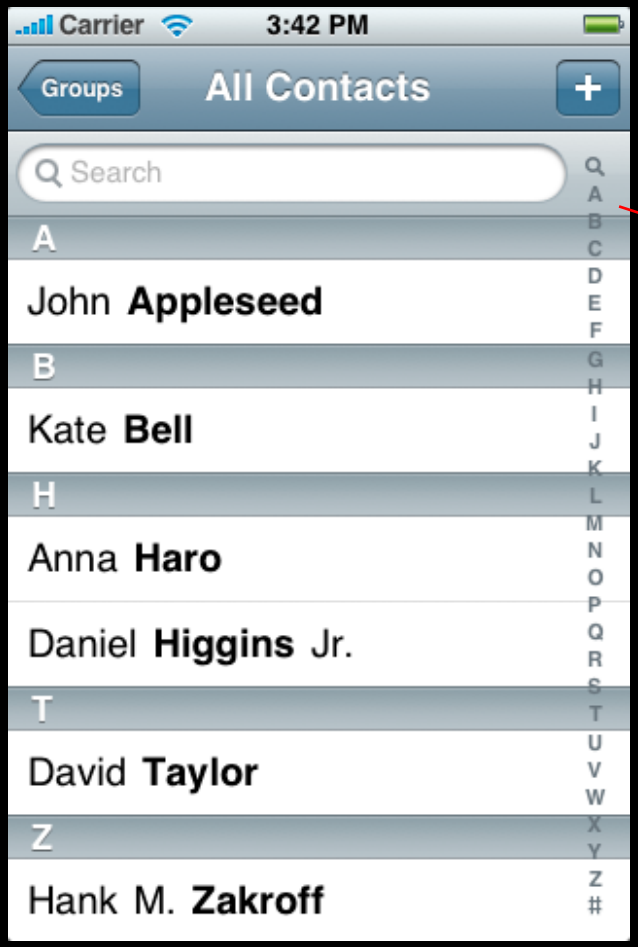
Datasource Message Flow



Datasource

Datasource Message Flow

`tableView:numberOfRowsInSection:`

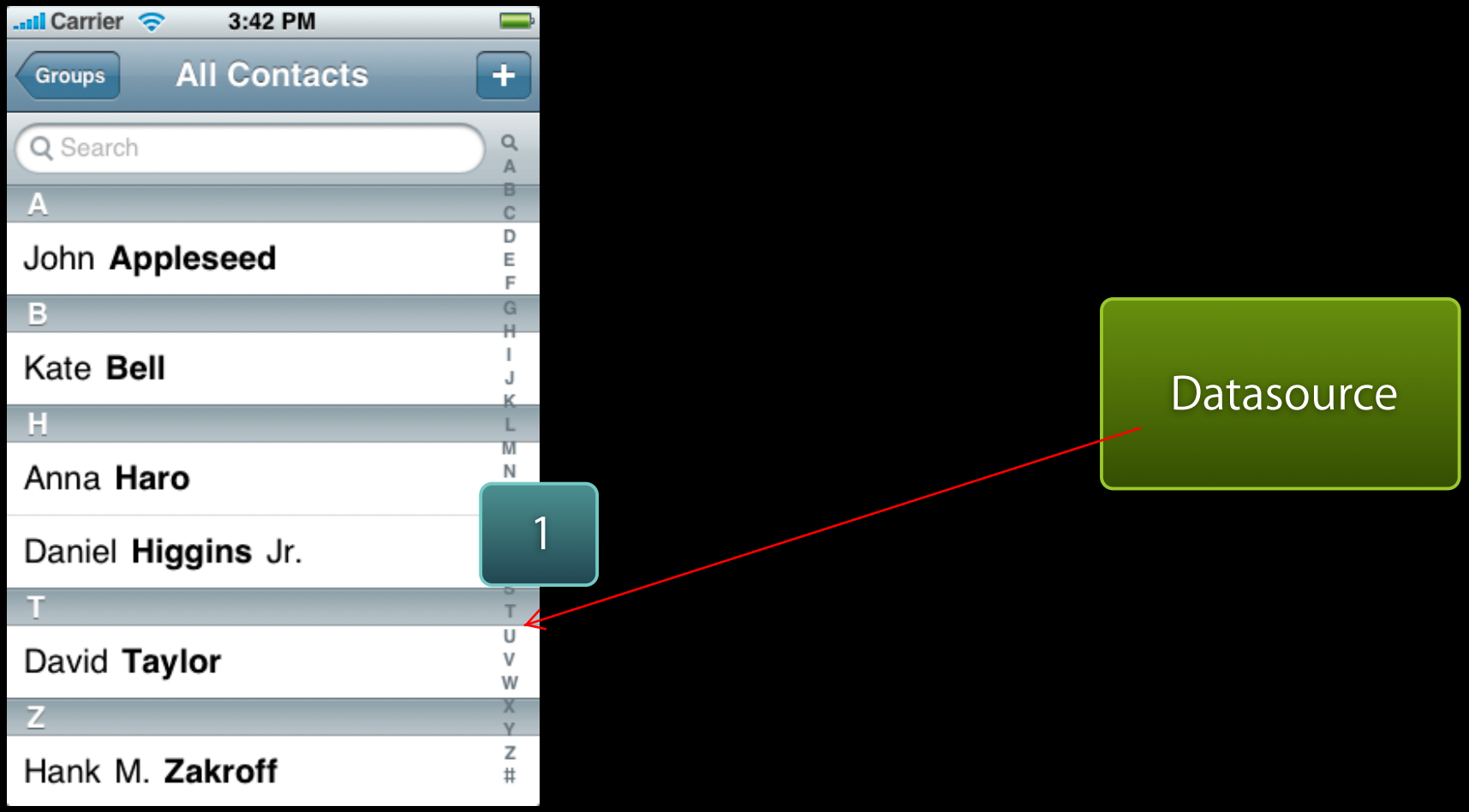


How many rows in section 0?

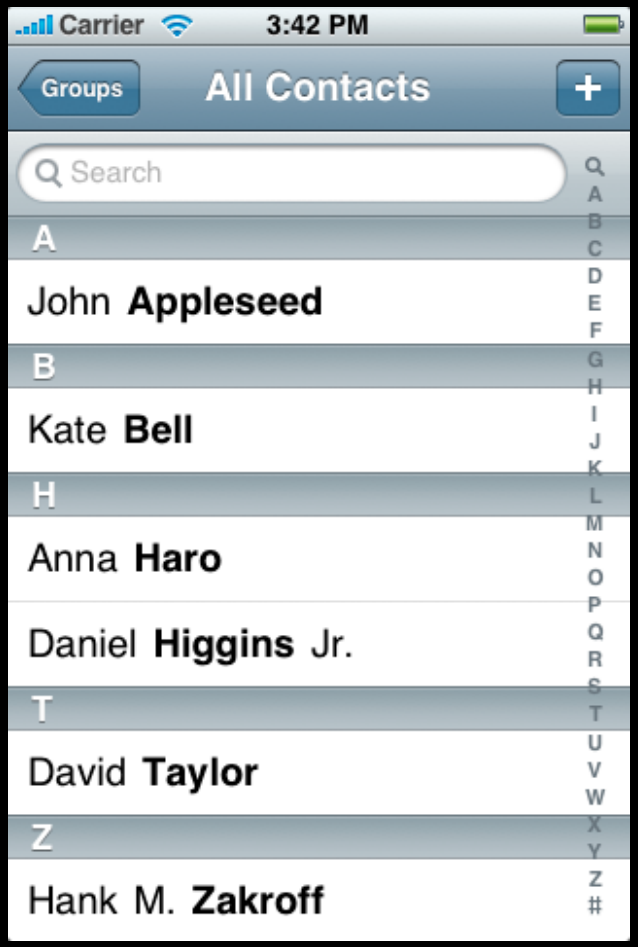
Datasource

Datasource Message Flow

`tableView:numberOfRowsInSection:`



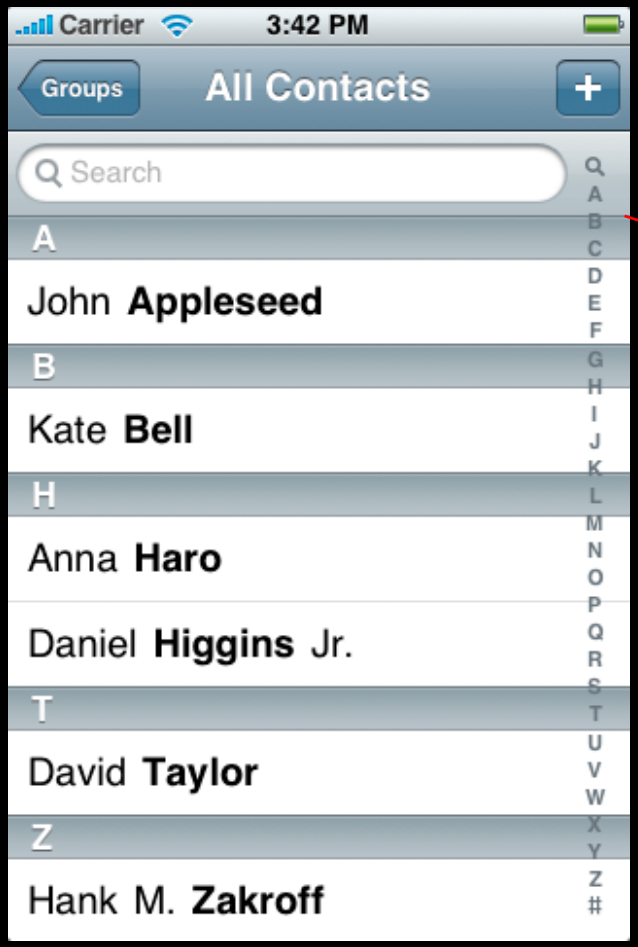
Datasource Message Flow



Datasource

Datasource Message Flow

`tableView:cellForRowAtIndexPath:`

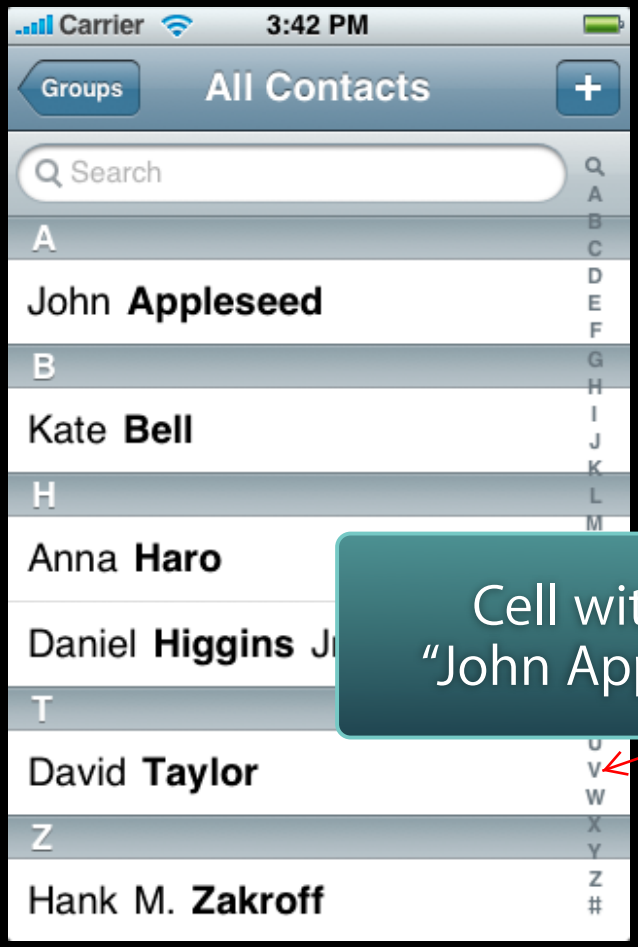


What to display at section 0, row 0?

Datasource

Datasource Message Flow

`tableView:cellForRowAtIndexPath:`



Cell with text
"John Appleseed"

Datasource

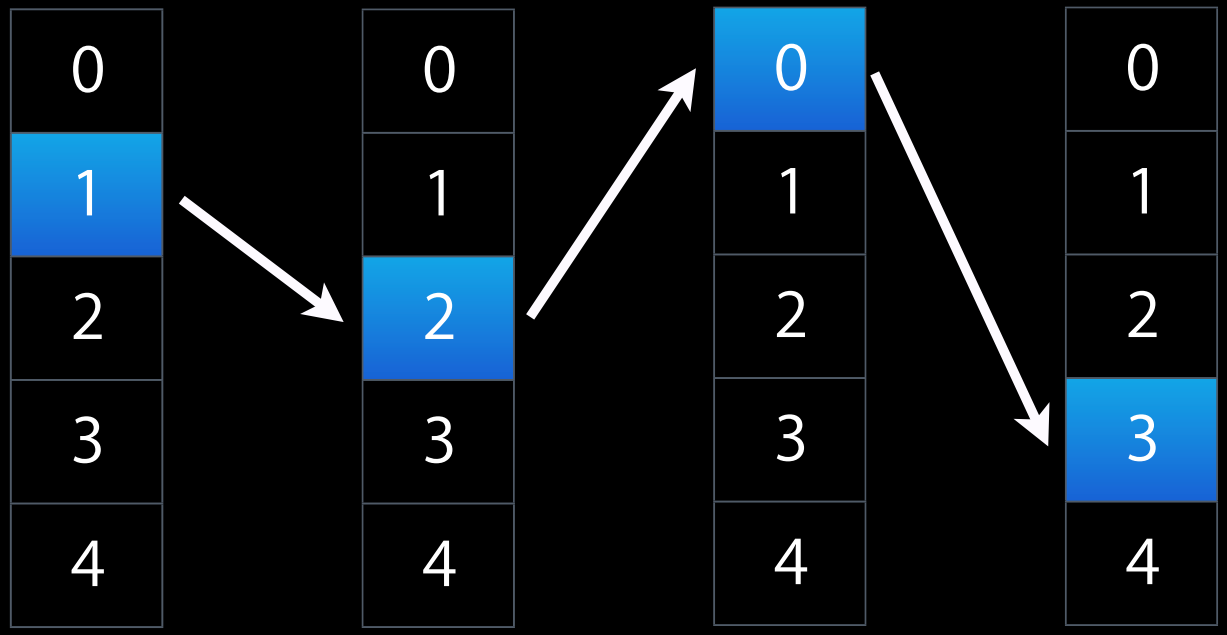
NSIndexPath インデックス パス というデータ構造

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

配列の配列（の配列…）の
特定要素への「パス」

NSIndexPath

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



array[1][2][0][3]

NSIndexPath and Table Views

- Cell location described with an index path
  - Section index + row index 「セクション＋行」の指定に使える
- 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

例として，単一セクション テーブル ビュー を考える

- Return the number of rows 行数を返すメソッド

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

セクションは０

String 配列の長さを返す

- Provide a cell when requested セル（内容）を返すメソッド

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

パス

セルを生成し，

textLabel に内容を入れ，
autorelease 設定して返す．

Cell Reuse セルの再利用

- When asked for a cell, it would be expensive to create a new cell each time. セルが要求される度，毎回新しく生成するのは無駄

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

文字列で
セル溜まりを
区別する

再利用可能セル溜まりから
セルを取り出す

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;
}
```

セル溜まりの識別子

nil なら「在庫なし」
新しくセルをつくる

内容をセットして返す

Triggering Updates 更新はいつ起こるか

- 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];
}
```

いつデータ要求されるか
- ある行が「見える」ようになったとき
- -reloadData によって強制的に更新されるとき
  （たとえば最初に表示するとき）

Section and Row Reloading

セクションと行のリロード

```
- (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

外見と動作（のカスタマイズ）

デリゲート

UITableView Delegate

- Customize appearance and behavior　外見と動作のカスタマイズ
- **Keep application logic separate from view**　**C を V から分離**
- Often the same object as datasource　データ供給元と同じオブジェクトになることが多い

Table View Appearance & Behavior

- Customize appearance of table view cell

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

セルが表示される直前に呼び出される（外見を変更できる）

- Validate and respond to selection changes

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

セルを選択する直前

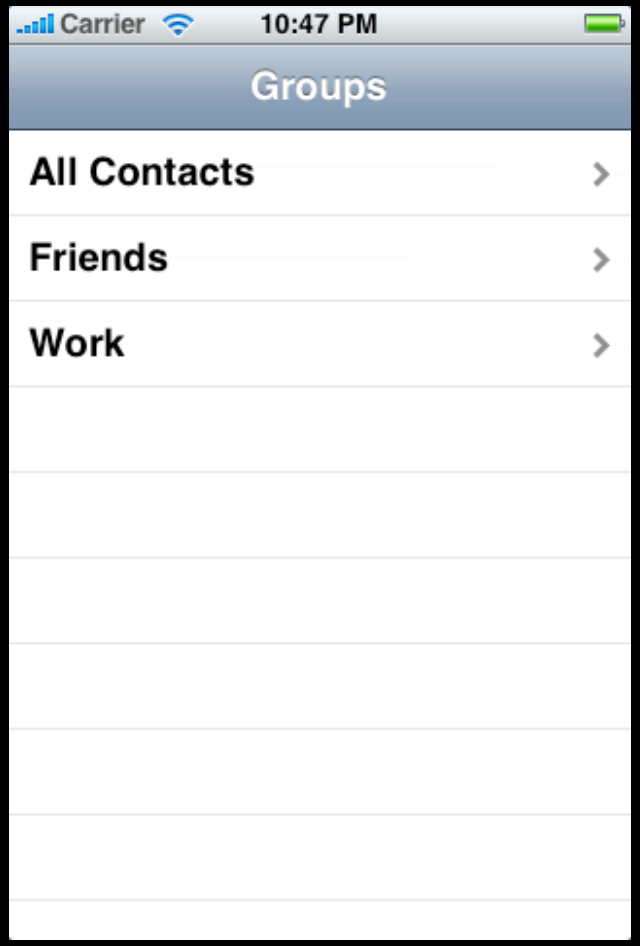
```
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath;
```

選択した直後

Row Selection in Table Views

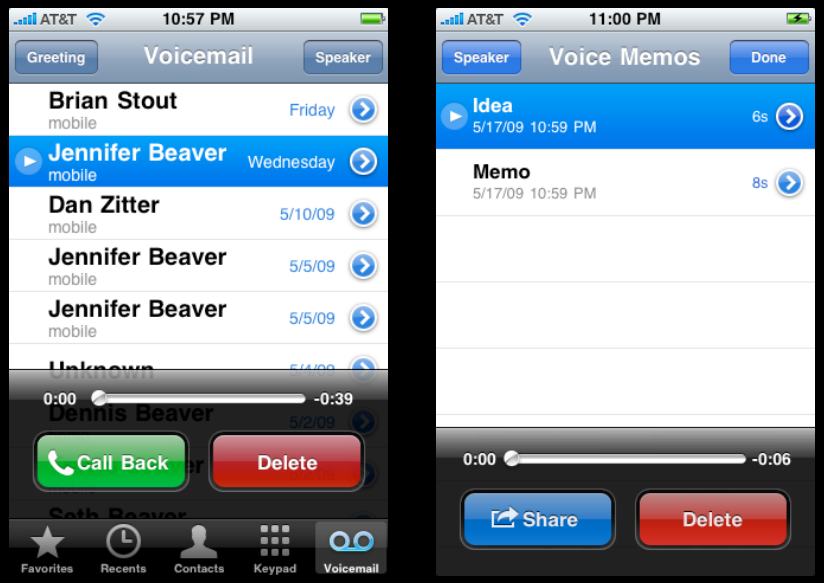
- In iPhone applications, **rows rarely stay selected**
- Selecting a row usually triggers an event

行（セル）が
選択され続ける
ことはまずない



行（セル）が
選択されることで
何らかのイベントが
発生する

Persistent Selection 選択され続ける場合



Responding to Selection

ある行が選択されると
このメソッドが呼ばれる

```
// 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];
}
```

パス

行の番号

データを
取り出す

新しい VC を生成

データをセット

push することで
表示画面をスライドイン

Altering or Disabling Selection

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

UITableViewController

UITableViewController テーブル ビュー コントローラ

- Convenient starting point for view controller with a table view
  - Table view is automatically created テーブル ビュー を自動生成
  - **Controller is table view's delegate and datasource**
- Takes care of some default behaviors
  - Calls -reloadData the first time it appears 最初に reload
  - Deselects rows when user navigates back 親画面に戻ったとき元の行を非選択化
  - Flashes scroll indicators インジケータをフラッシュ

Table View Cells

Designated Initializer

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

Designated Initializer

DEPRECATED

古いので使わない

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

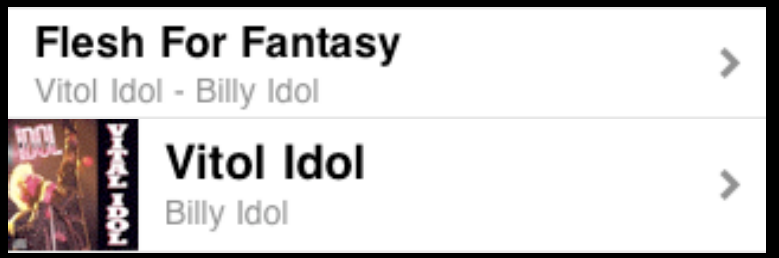
スタイル

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

Cell Styles

UITableViewCellStyleDefault

UITableViewCellStyleSubtitle



アイコン可

UITableViewCellStyleValue1

UITableViewCellStyleValue2

Basic properties 基本プロパティ

- 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”;
```

こんな風になります

Accessory Types アクセサリのいろいろ

```
// 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;
    ...
}
```

行に「ビュー」をぶちこむ（何でも表示できる）

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]; ぶちこむ

  [myLabel release];
}
```

テーブルビューの中に
スクロールビューを入れるなど
いろいろできる

Questions?