iPhone Application Programming
Lecture 5: View Programming

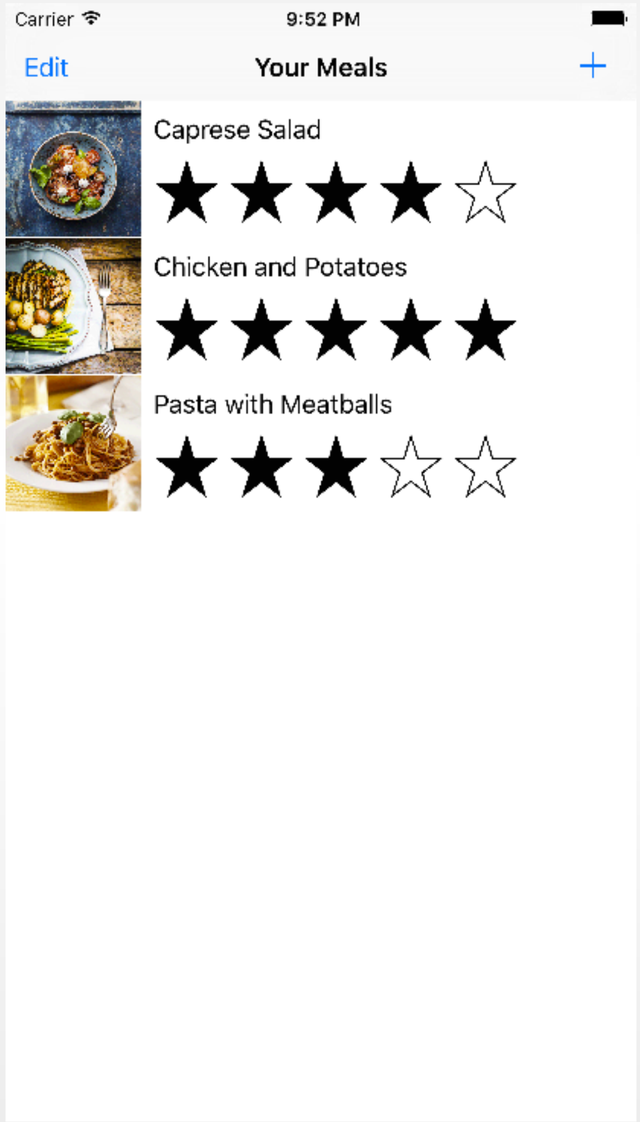
Nur Al-huda Hamdan
Media Computing Group
RWTH Aachen University

Winter Semester 2015/2016

http://hci.rwth-aachen.de/iphone

Media Computing Group | RWTH AACHEN UNIVERSITY

Name the UI Elements In the Screen



View Programming

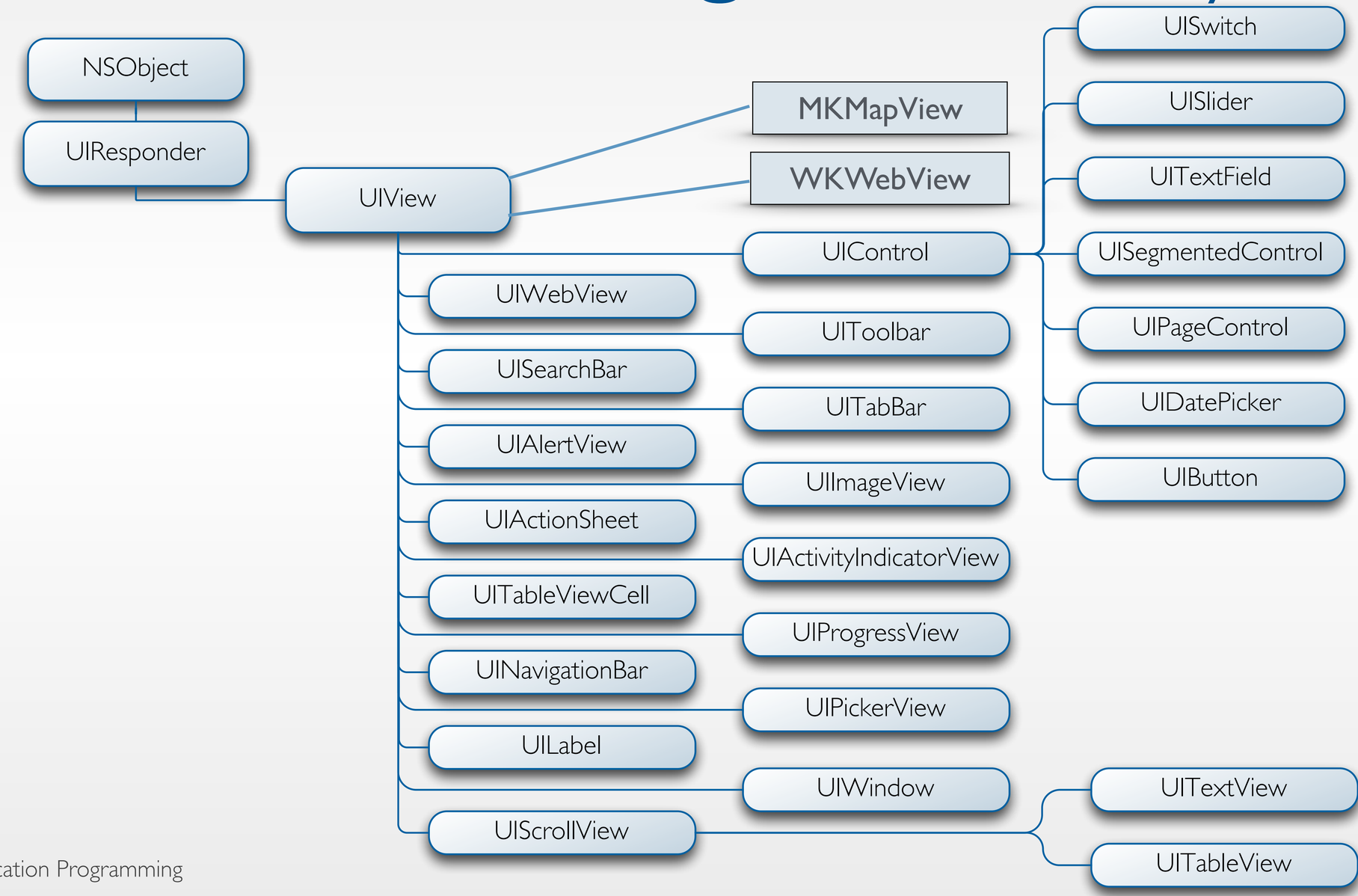
View Programming Concepts

- At run-time Views are organized as a tree
- Use Interface Builder to design your UI and connect it to code
- Geometry of Views are determined by constraints
- SDK provide many types of Views to show your content

iOS Anatomy

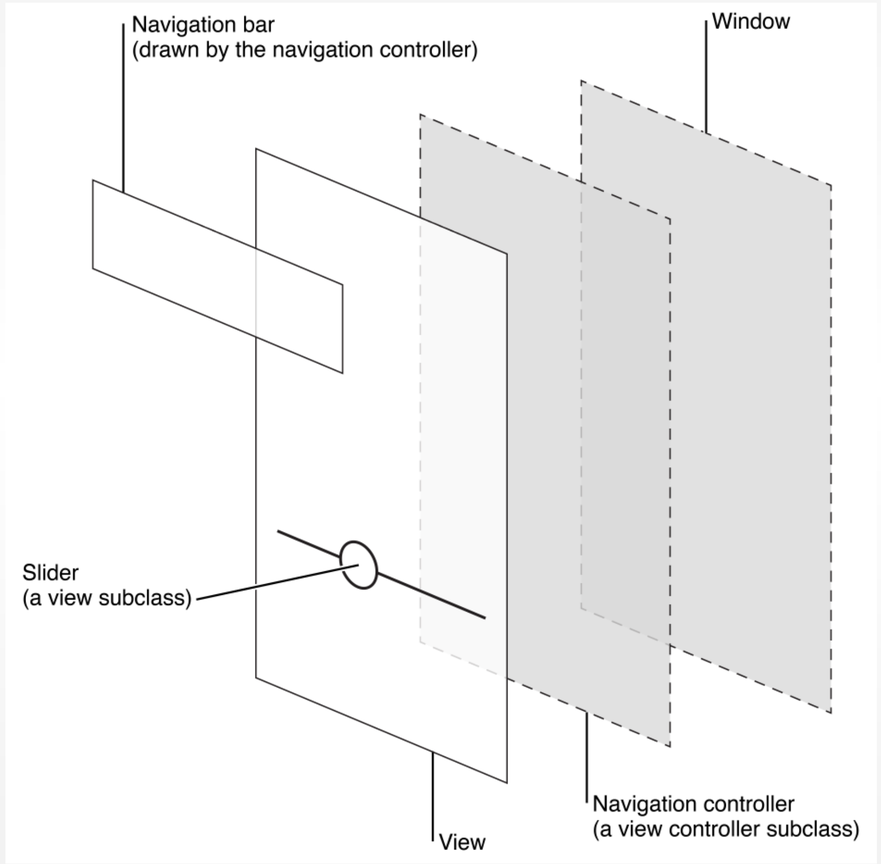
- In iOS windows and views present your app content on the screen.
- Windows have no visible content themselves, they provide the space and distribute events to views
  - rootViewController: provides the content view of the window; screen is mainScreen or external scree (a new windows is created for each external screen)
- UIKit and other frameworks provide predefined views. You can also define custom views and manage the drawing and event handling yourself
- A view is an instance of the UIView class (or one of its subclasses) and manages a rectangular area in the app window
- Views are responsible for drawing content, handling multitouch events, and managing the layout of subviews
- Views can display images, text, shapes, or some combination. You can also use views to organize and manage other views

Static Widget Hierarchy



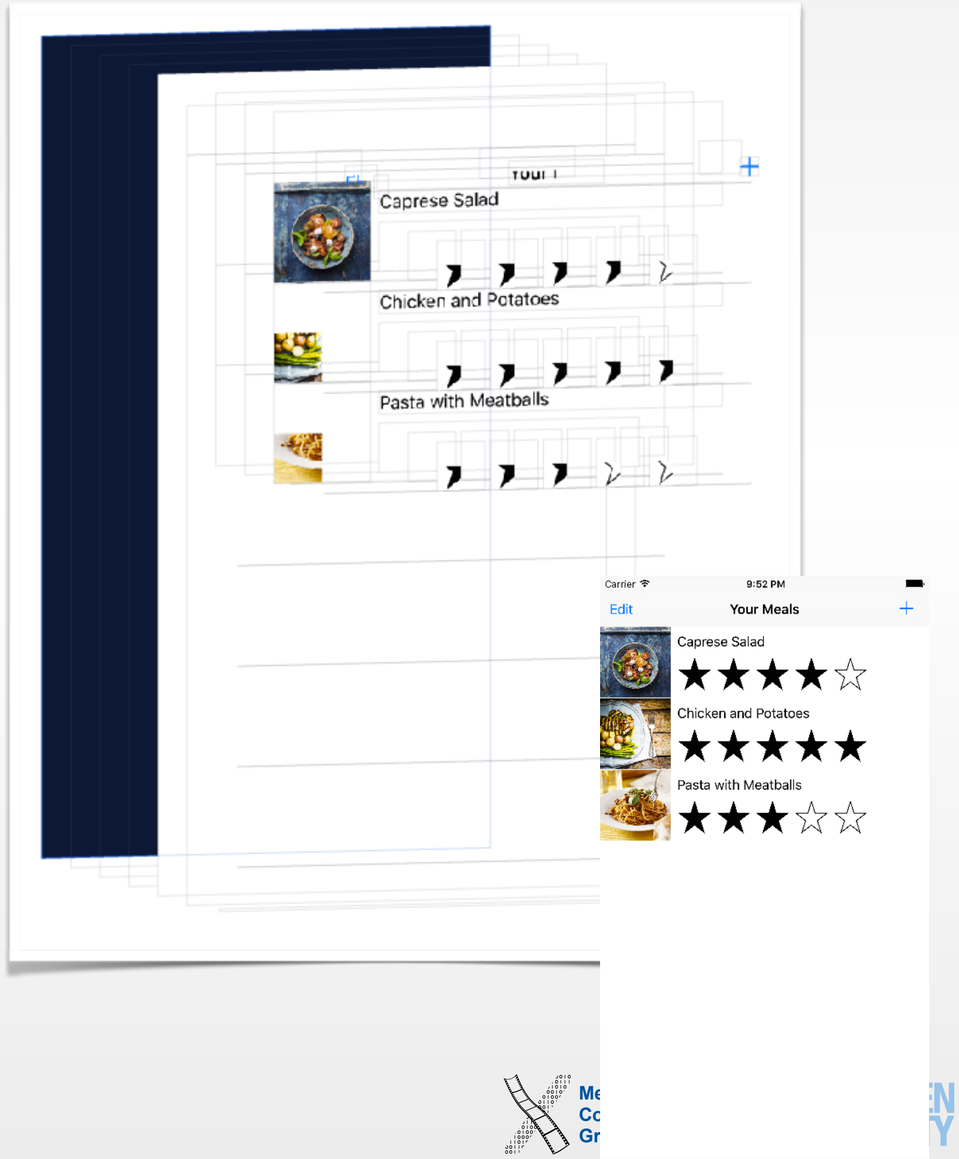
Dynamic Widget Hierarchy

- An app is composed of a dynamic hierarchy of views; you can create, add, remove, and manipulate views at run time
- View controllers mange view hierarchies: coordinate the display of views, implement user interactions functionality, and manage transitions from one screen to another
- As a developer you deal with views and view controllers



View Hierarchy at Runtime

```
po [[UIWindow keyWindow] recursiveDescription]
<UIWindow:
    | <UILayoutContainerView:
    |     | <UINavigationTransitionView:
    |     |     | <UIViewControllerWrapperView:
    |     |     |     | <UITableView:
    |     |     |     |     | <UITableViewWrapperView:
    |     |     |     |     |     | <FoodTracker.MealTableViewCell:
    |     |     |     |     |     |     | <UITableViewCellContentView:
    |     |     |     |     |     |     |     | <UIImageView:
    |     |     |     |     |     |     |     | <UILabel:
    |     |     |     |     |     |     |     | <FoodTracker.RatingControl:
    |     |     |     |     |     |     |     |     | <UIButton:
    |     |     |     |     |     |     |     |     |     | <UIImageView:
…
    |     | <UINavigationBar:
    |     |     | <_UINavigationBarBackground:
…
    |     |     | <_UINavigationBarBackIndicatorView:
```



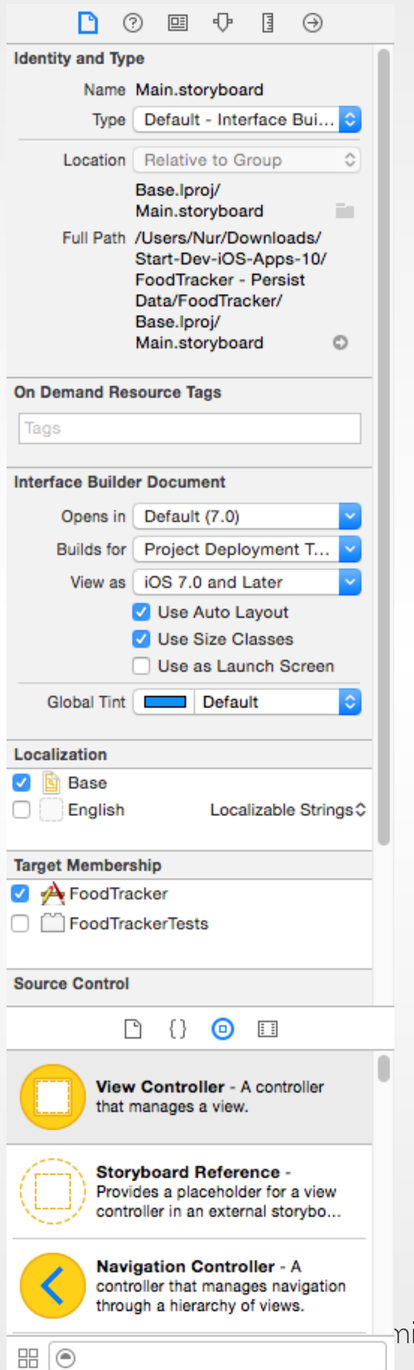
View Animation

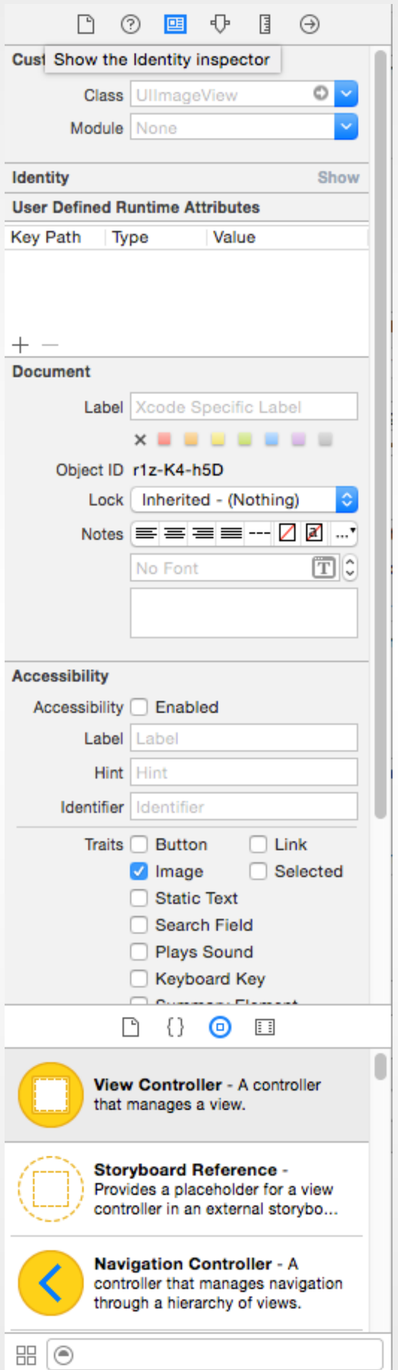
- UINavigationTransitionView and UIViewControllerWrapperView are used to animate UINavigationController child views. The wrapper contains your view
- Each view has an underlaying Core Animation **layer** object, which allows you to perform other animations
- Animations provide users with visible feedback about changes in view hierarchy
- The system defines standard animations for presenting modal views and transitioning between views
- You can animate the changes of view attributes, e.g., transparency of a view, change of position on the screen, size, background color, etc.

```
po [[UIWindow keyWindow] recursiveDescription]
<UIWindow:
   | <UILayoutContainerView:
   |    | <UINavigationTransitionView:
   |    |    | <UIViewControllerWrapperView:
```

Interface Builder

- Graphical tool to layout user interfaces
- Create the widget hierarchy
- Set attributes of widgets
- Set up connections between the widgets
- Store these informations in nib files
  - The contents of .xib and .storyboard files are stored in XML format and compiled to nibs (binary files). At runtime, nibs are loaded and instantiated to create new views


Identity and Type
Name Main.storyboard
Type Default - Interface Bui...
Location Relative to Group
Base.lproj/
Main.storyboard
Full Path /Users/Nur/Downloads/
Start-Dev-iOS-Apps-10/
FoodTracker - Persist
Data/FoodTracker/
Base.lproj/
Main.storyboard
On Demand Resource Tags
Tags
Interface Builder Document
Opens in Default (7.0)
Builds for Project Deployment T...
View as iOS 7.0 and Later
Use Auto Layout
Use Size Classes
Use as Launch Screen
Global Tint Default
Localization
Base
English Localizable Strings
Target Membership
FoodTracker
FoodTrackerTests
Source Control
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storybo...
Navigation Controller - A controller that manages navigation through a hierarchy of views.


Show the Identity inspector
Class UIImageView
Module None
Identity Show
User Defined Runtime Attributes
Key Path Type Value
Document
Label Xcode Specific Label
Object ID r1z-K4-h5D
Lock Inherited - (Nothing)
Notes
No Font
Accessibility
Accessibility Enabled
Label Label
Hint Hint
Identifier Identifier
Traits Button Link
Image Selected
Static Text
Search Field
Plays Sound
Keyboard Key
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storybo...
Navigation Controller - A controller that manages navigation through a hierarchy of views.

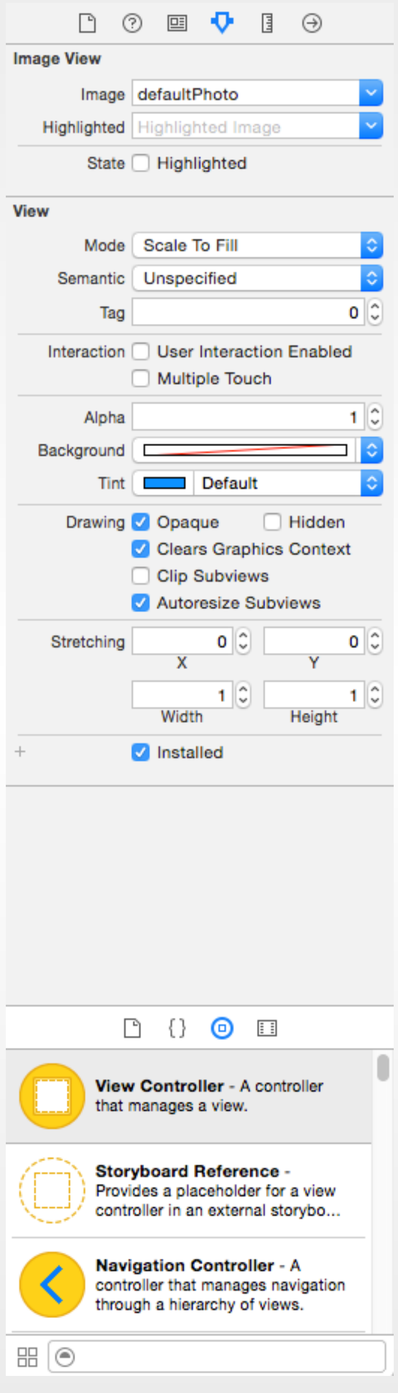
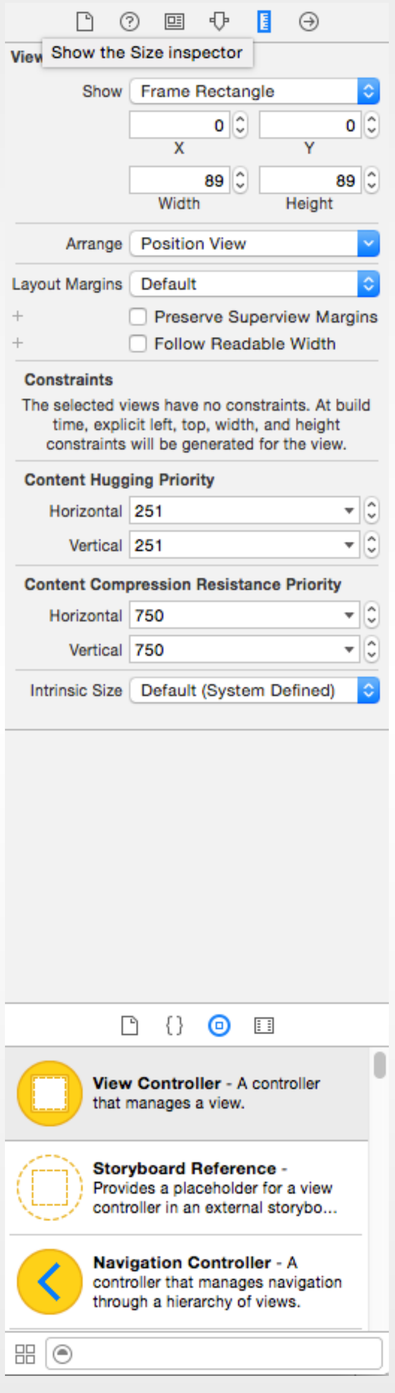
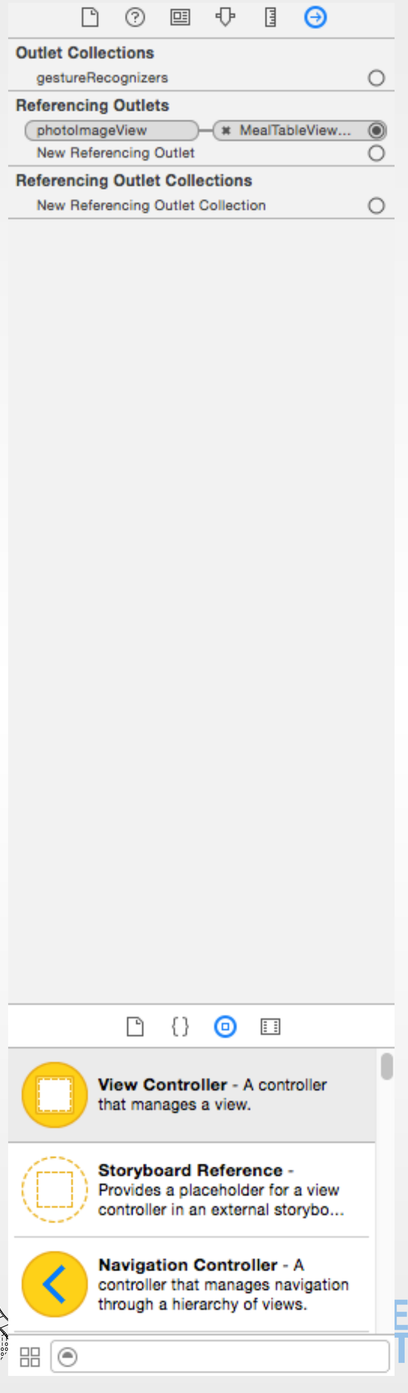


Image View
Image defaultPhoto
Highlighted Highlighted Image
State Highlighted
View
Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 0
X Y
1 1
Width Height
Installed
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storybo...
Navigation Controller - A controller that manages navigation through a hierarchy of views.


Show the Size inspector
Show Frame Rectangle
0 0
X Y
89 89
Width Height
Arrange Position View
Layout Margins Default
Preserve Superview Margins
Follow Readable Width
Constraints
The selected views have no constraints. At build time, explicit left, top, width, and height constraints will be generated for the view.
Content Hugging Priority
Horizontal 251
Vertical 251
Content Compression Resistance Priority
Horizontal 750
Vertical 750
Intrinsic Size Default (System Defined)
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storybo...
Navigation Controller - A controller that manages navigation through a hierarchy of views.


Outlet Collections
gestureRecognizers
Referencing Outlets
photoImageView MealTableView...
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storybo...
Navigation Controller - A controller that manages navigation through a hierarchy of views.

Object Library

- The library contains all UI Widgets
- Drag them to your view
- Modify their attributes
- Connect them to code
- See instantly what your UI looks like
- Test your UI in the Simulator

View Programming Concepts

- ✓ At run-time Views are organized as a tree
- ✓ Use Interface Builder to design your UI and connect it to code
- Geometry of Views are determined by constraints
- SDK provide many types of Views to show your content

Superview-Subview Relationship

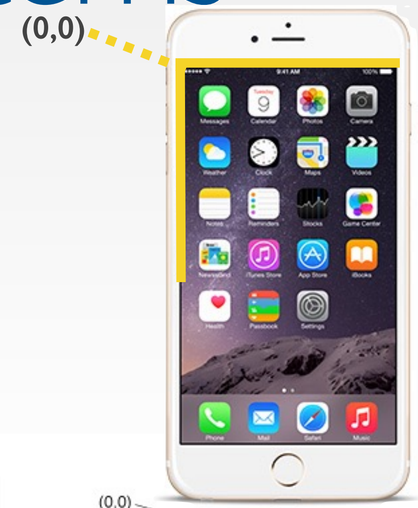
- A view can have subview(s) and a superview. This has implications on the appearance and behaviours of subviews
- Hierarchies can be built graphically or programatically (in loadView)
- A view maintains an array of subviews. It can add, insert at index, bring back/front, remove
- Changing the size of a parent view affects the size and position of subviews. You can configure the resizing behavior of each subview
- Other effects: hiding a superview, changing a superview's alpha (transparency), or applying a mathematical transform to a superview's coordinate system

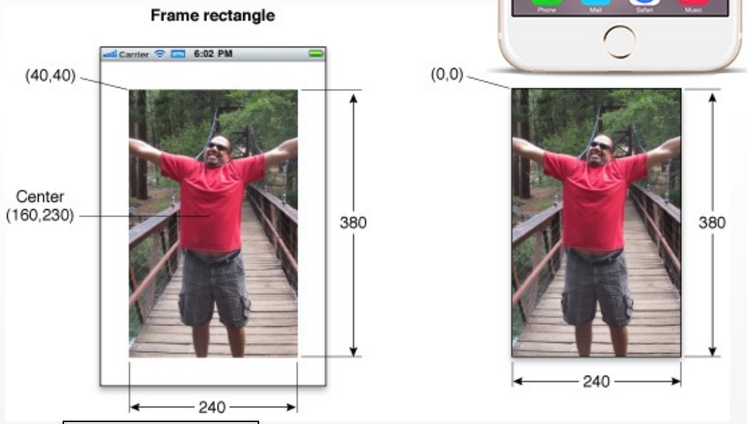
The View Drawing Cycle

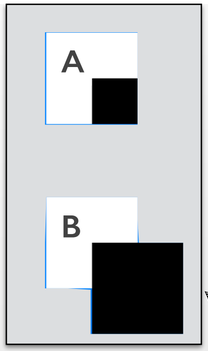
- When a view first appears on the screen, the system asks it to draw its content and captures a snapshot of the view
- When the contents of view change, do not redraw directly, but use setNeedsDisplay or setNeedsDisplayInRect: methods to tell the system view needs to be redrawn
- Changing a view's geometry does not automatically cause redraw. Based on contentMode property the existing view snapshot might be stretched or repositioned only
- drawRect method

View Geometry and Coordinate Systems

- Origin in the top-left corner. Floating-point numbers (points not pixels, 1 point = 1, 2, or 3 pixels)
- A view object tracks its size and location using its frame, v, bounds properties:
  - Frame: the size and location of the view in its superview's coordinate system
  - Centre: centre point of the view in the superview's coordinate system
  - Bounds: specifies the size of the view (and its content origin (0,0)) in the view's own local coordinate system
- Use the frame and centre to build the view hierarchy or changing the position or size of a view at runtime
- Use bounds during drawing the visible content of the view
- By default,a view's frame is not clipped to its superview's frame, subviews that lie outside of their superview's frame are rendered in their entirety





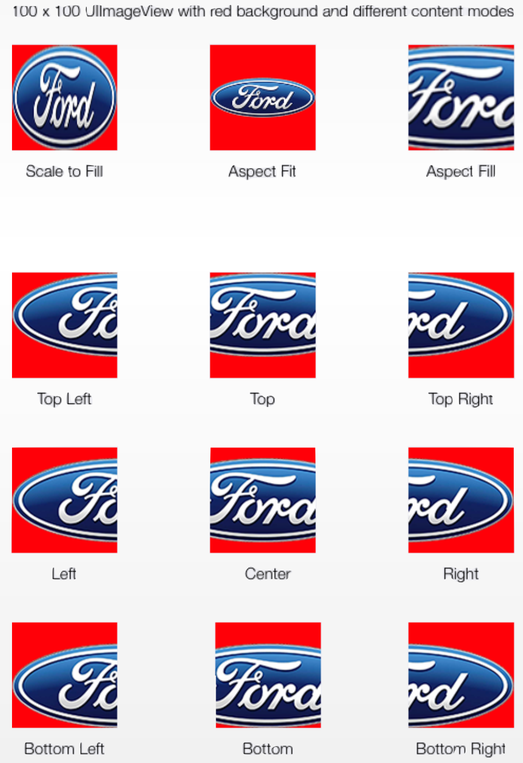


Responding to Geometry Change

- Each view has a content mode to adjust its content in response to changes in the view's geometry
  - Change the width or height of the view's frame or bounds
  - Assign a transform that includes a scaling factor to the view's transform property
- Default value: UIViewContentMode.ScaleToFill (do not use .Redraw with standard UI)
- Intrinsic content size property describes the minimum space needed to express the full view content without squeezing or clipping that data (used for auto layout)

Source

Original Image



Custom Views

- init(frame: CGRect) if programmatically OR required init?(coder aDecoder: NSCoder) if graphically
- Set the autoresizingMask
- Create subviews during your view's initialization and set their autoresizingMask property
- Can override the following:
  - touchesBegan:withEvent:; touchesMoved:withEvent:; touchesEnded:withEvent:; touchesCancelled:withEvent:
  - The layoutSubviews method. Geometry of a view changed, setNeedsLayout, layoutIfNeeded. Adjust the position and size of any subviews (can call for redraw)
  - The drawRect: method. setNeedsDisplay, setNeedsDisplayInRect:. Redraw the specified area of the view as quickly as possible and nothing else

View Programming Concepts

- ✓ At run-time Views are organized as a tree
- ✓ Use Interface Builder to design your UI and connect it to code
- ✓ Geometry of Views are determined by constraints
- • SDK provide many types of Views to show your content

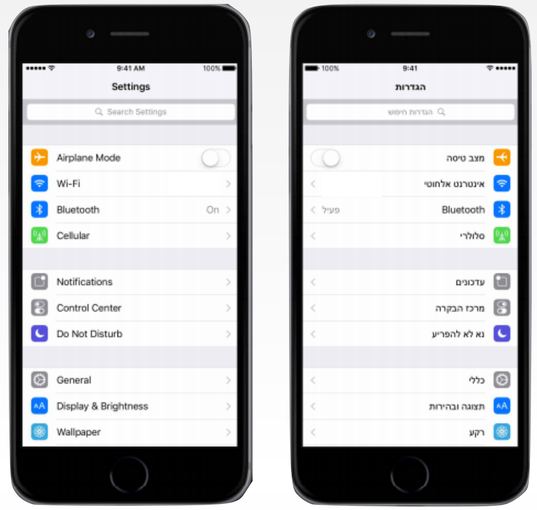
UIKit

UIBarItem and UIBarButtonItem are not UIViews

- Almost all iOS apps use UI components from UIKit framework
- UIKit defines UI elements' look, properties, behaviour, gesture recognition, drawing, accessibility, print support
- UI elements fall in 4 categories:
  - **Content views** provide app-specific content and can enable user interaction
  - **Temporary views** appear briefly to give users information or functionality
  - **Controls** perform actions or display information
  - **Bars** provide contextual information to navigate or initiate actions

UI Elements

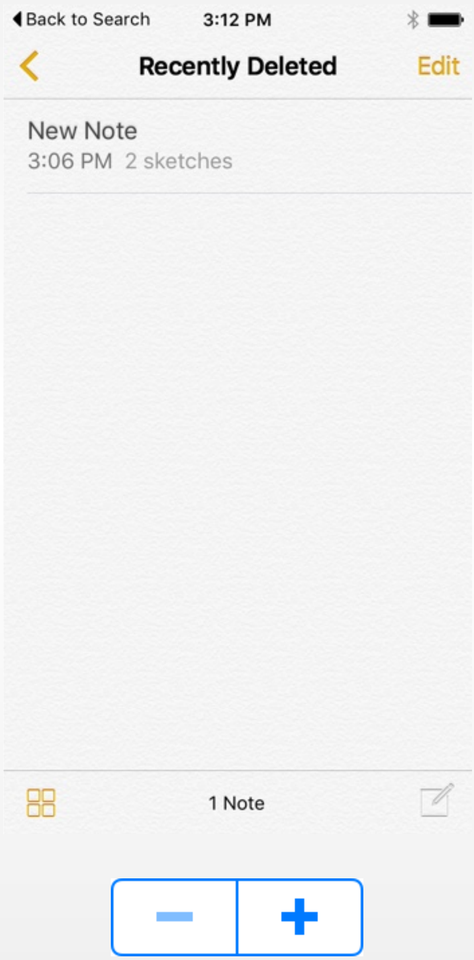
Internationalizing in iOS 9



- Purpose
- Content
- Behaviour
- Appearance
- Accessibility: by default all views are accessible, you can modify this in Accessibility section of the Identity inspector (check out Accessibility Programming Guide for iOS)
- Internationalizing (input processing, formats, e.g. dates, lengths, weights, prices, and currency symbols, New! RTL UIKit support) and localization (text files in multiple languages)

View Appearance

- You can customize the appearance of all instances of a class by sending appearance modification messages to the class's appearance proxy
  - UISlider.appearance().backgroundColor = UIColor.redColor()
  - UISlider.appearanceWhenContainedInInstancesOfClasses([UIView.self]).backgroundColor = UIColor.redColor()
- Tint color defines the app skin colour. Used to visually indicate which controls on the screen are active or have an action. Defaults to blue, e.g., Note is yellow
- Each view inherits its superview's tint color if its own tint color is nil
- You cannot use the appearance proxy with the tint color property on UIView



Images in Views

- View can have images as content, as foreground, and as background (w/o shadow image)
- If you define bg image for navigation bar or segmented control need to provide image in all control states and provide divider images
- A property for UIImage is UIImageRenderingMode
  - .AlwaysTemplate: ignore the colour information of image and use superview tint color for all non-transparent parts
  - .AlwaysOriginal: not effected by tint color
  - .Automatic: automatically decides which rendering mode to use based on where the image is being displayed
- Navigation bars, tab bars, toolbars, segmented controls automatically treat their foreground images as templates, and their background images as original
- Image views and web views treat their images as originals

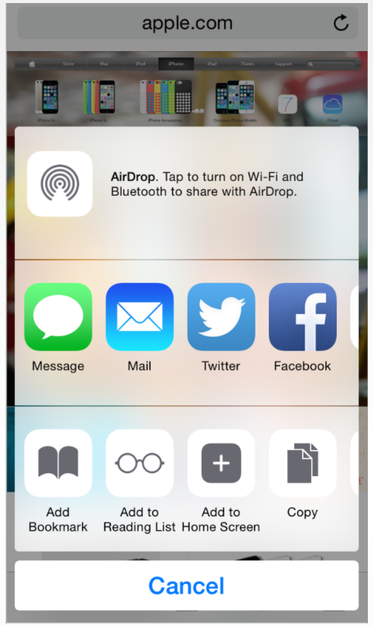
Content Views

- Provide app-specific content and can enable user interaction
- Associated with view controllers
- Have delegation protocols to communicate user interaction to a view controller

Activity

- UIActivity (UIActivityViewController)
- Trigger using the Action button 
- Gives users access to a custom service or task that your app can perform on selected content
- Has an icon and a title (make short or iOS tries to shirk first then truncates)
- Can appear in an action sheet or a popover (in iPad)
- iOS provides several built-in *actions* and *share* extensions, such as Print, Twitter, Message, and AirPlay
- You can add and exclude activities

70 x 70 pixels
(high resolution)



Task Time Indictors

- UIActivityIndicatorView
- If an action takes noticeable and indeterminate amount of time to process, display an activity indicator
- Shouldn't have a stationer location in a view. Use startAnimating and stopAnimating methods (automatically shows/hides onscreen)
- Typically, activity indicators appear before a label or centered within a view.
- If the task takes a determinate amount of time, and to let the user know how long until an operation completes, use progress view UIProgressView
- You can set the initial progress as a float between 0 and 1 (in viewDidLoad)

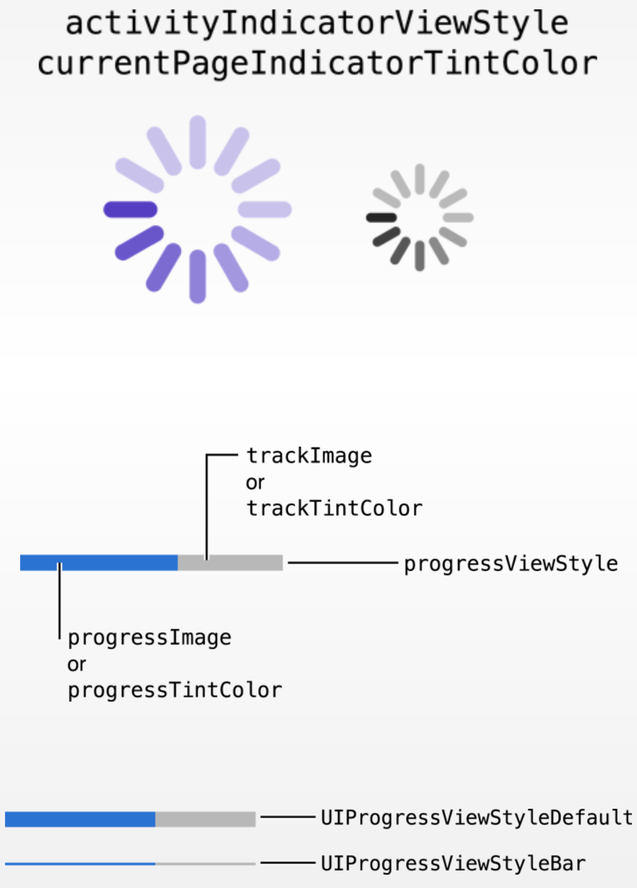
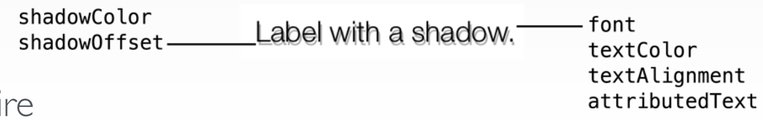


Image View

- UIImageView
- Displays an image or an animated sequence of images (array of images with the same scale)
- For a sequence of images yo can set the animation duration and the repeat count
- Scales the images automatically to fit within the current size of the view
  - Resizing modes: tile or stretch (with content mode UIViewContentModeScaleToFill)
  - Rendering modes
- Has 2 states: highlighted and not, e.g., ☐☑
- Can access image by name if in app bundle, or using URL if in the file system

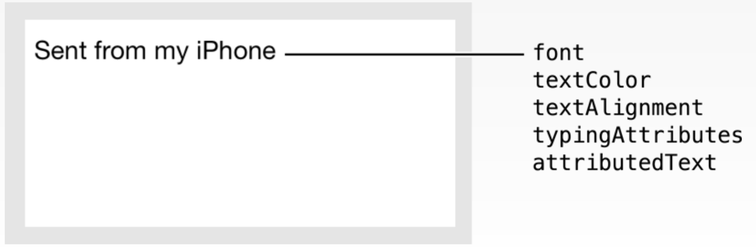
Label

- UILabel
- Displays static text
- Used in conjunction with controls to describe their intended purpose
- Allows for plain text and attributed text
  - Plain text supports a single set of formatting attributes—font, size, color, and so on—for the entire string
  - Attributed text supports multiple sets of attributes that apply to individual characters or ranges of characters in the string
- By default, a label is a single line. Make multiline with numberOfLines property
- Fitting text policy
  - If you set adjustsFontSizeToFitWidth, the font size would be reduced in order to fit into the label's bounding rectangle.
  - The lineBreakMode property specifies what happens when a line is too long for its container (wrap, clip or truncate the text to fit)



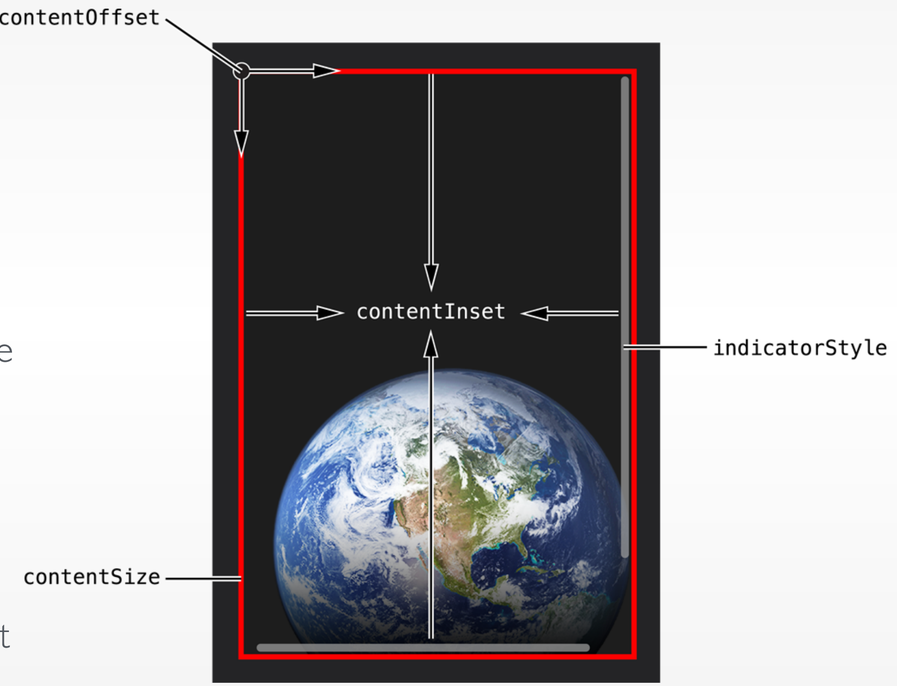
Text Views

- UITextView
- Accepts and displays multiple lines of text
- Support scrolling and text editing
- Plain and attributed text
- By default, users can add, remove, or change text within a text view (you can disable that)
- A text view is capable of recognizing when text is formatted as a link, address, phone number, or event.
  - If you enable dataDetectorTypes checkboxes, users will be able to trigger the appropriate actions form text



Scroll View

- UIScrollView, UIScrollViewDelegate
- Allows users to see content that is larger than the scroll view's boundaries
- Responds to the speed and direction of gestures to reveal content in a way that feels natural to people
- Set scroll view content programmatically by adding subviews to its content view
- Can also operate in paging mode, in which each drag or flick gesture reveals one app-defined page
- Scroll views need a delegate to handle scrolling, dragging, and zooming
- Can enable/diable vertical or horizontal scrolling, an both directions, including diagonal spanning. Can set the max and min zooming allowed
- Paging enabled make the scroll view stops on multiples of the scroll view's bounds, giving the effect of scrolling through a single page at a time
- Number of options that dictate how content is laid out: size of the content, contentInset property to specify the distance that the content is padded, contentOffset to set the point at which the origin of the content view is offset from the origin of the scroll view



Picker

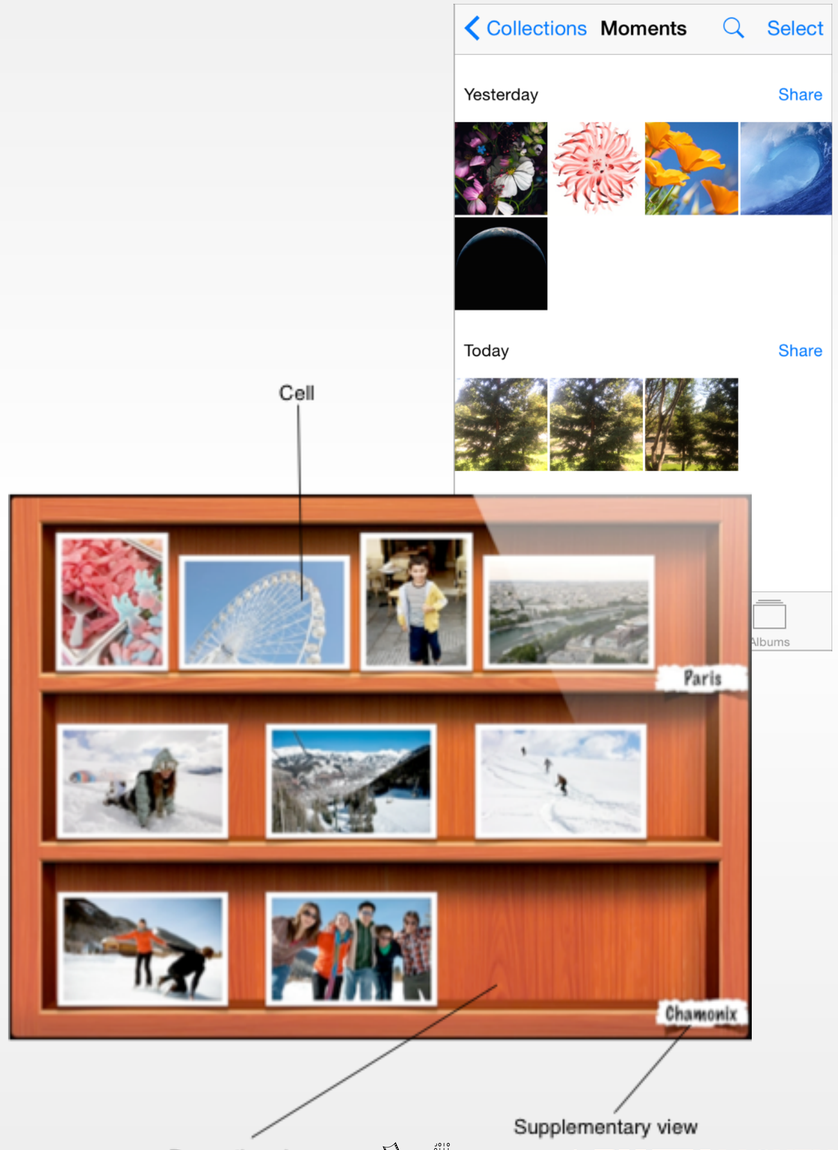
- UIPickerView, UIPickerViewDelegate, UIPickerViewDataSource
- Lets the user choose between certain options by spinning a wheel on the screen
- Requires both a data source (number of components/columns , number of rows in each component) and a delegate (content for each component's row)
- After setting the data source and delegate, in viewDidLoad, set the initial selection
- You can dynamically change the rows of a component by calling the reloadComponent: method, or dynamically change the rows of all components by calling the reloadAllComponents method
- You can only manipulate the pickers background color

Web View

- WKWebView, WKNavigationDelegate (optional)
- A web view is a region that can display rich HTML content
- Allows for swiping between the page history
- You can set your web view to automatically scale web content to fit on the screen of the user's device, which allows the user to zoom in and out in the web view
- A web view is capable of recognizing when web text is formatted as a link, address, phone number, or event (enable dataDetectorTypes to allow interaction)
- The delegate provides methods for tracking the progress of the web page loading, and navigations data for deciding on load policy
- In iOS 9: SFSafariViewController, an embedded Safari inside your app, with shared cookies, AutoFill of forms, and Reader Mode
- SFSafariViewControllerDelegate
  - let webView = SFSafariViewController(URL: NSURL(string: "http://www.hci.rwth-aachen.de")!, entersReaderIfAvailable: true)
  - presentViewController(webView, animated: true, completion: nil)

Collection View

- UICollectionView, UICollectionViewCell,UICollectionViewDataSource, UICollectionViewDelegate, and more…
- Can display items in a grid or in a custom layout that you design (sections available)
- Purpose
  - View a catalog of variably sized items
  - Add to, rearrange, and edit a collection of items
  - Choose from a frequently changing display of items
- Recognises tap (to select an item) and touch-and-hold (to edit an item) (can add more)
- A collection view allows you to change the layout of items while users are viewing and interacting with them
- Users can add, remove, insert, reorder, and edit items



StackView

- UIStackView (iOS 9)
- Allows for linear layout. Add views to a superview and they automatically are placed in a column or a row
- Leverage the power of Auto Layout. You determine the size of the view, and it determines the size and position of the subviews
- You only need to specify how much spacing you want between the items, arrange horizontally or vertically (that can update with device orientation), and if the items have equal or proportional sizes
- They do not scroll!

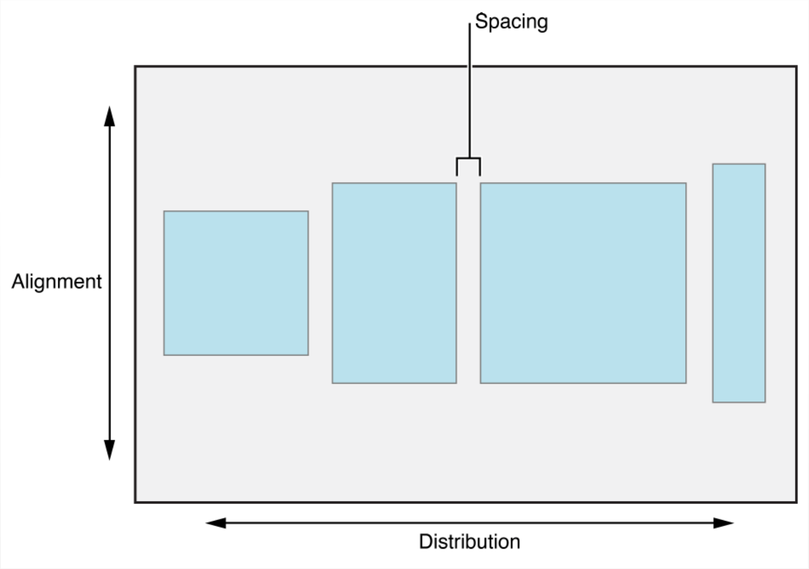
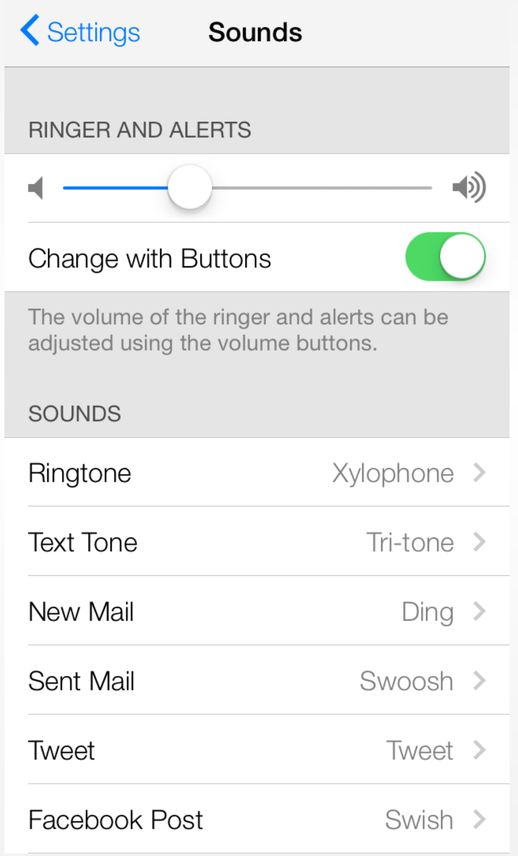
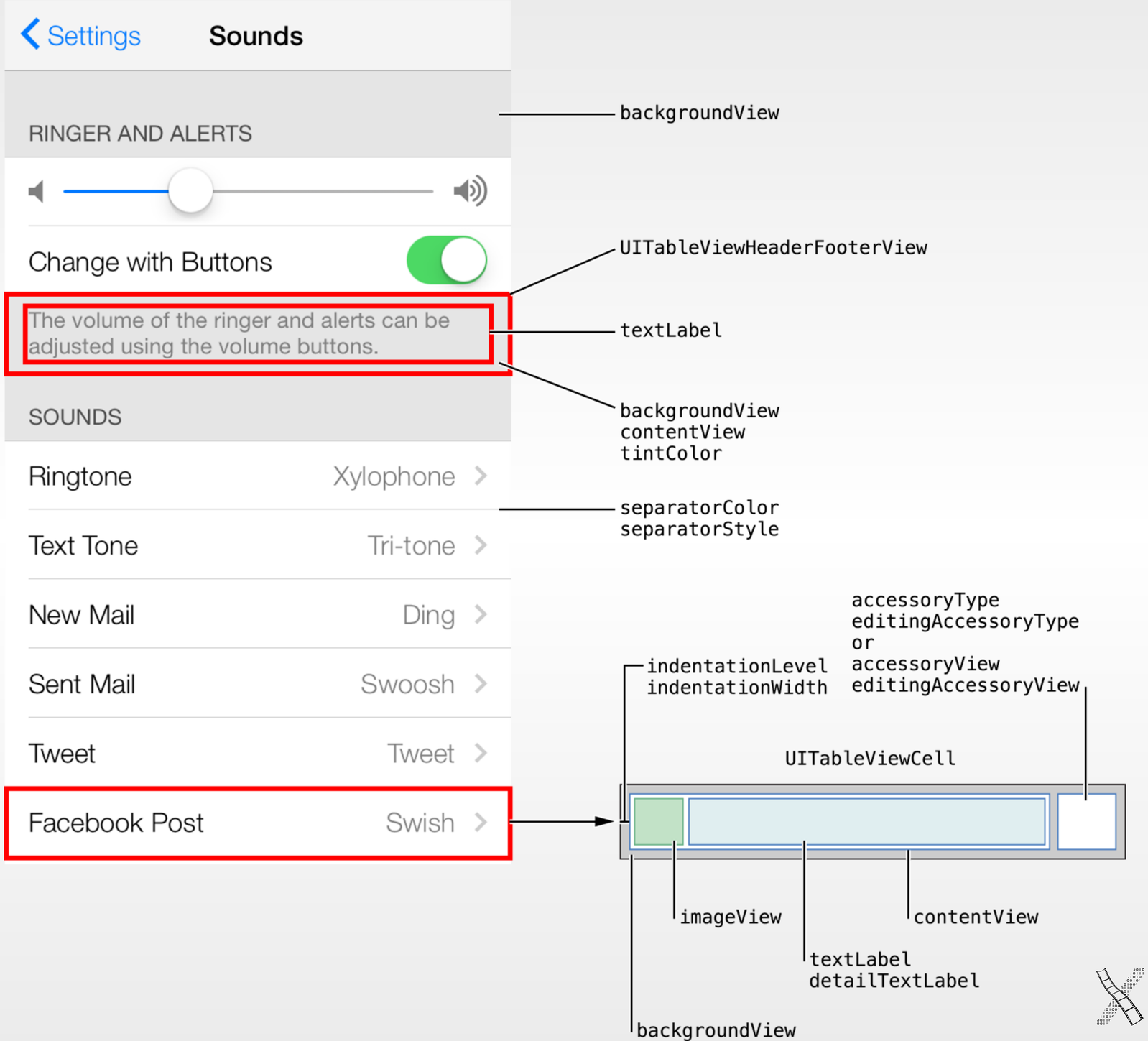


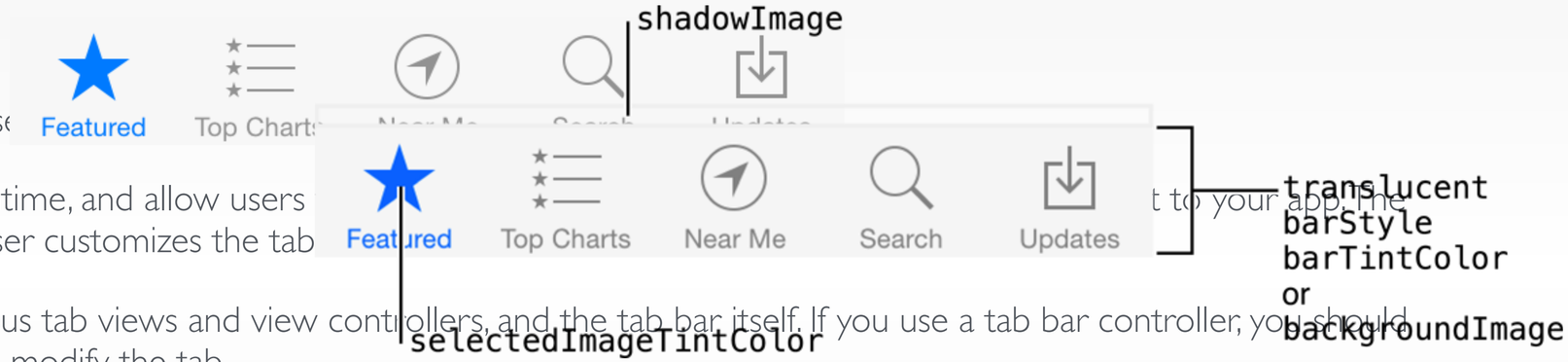
Table View

- Presents data in a scrollable list of multiple rows that may be divided into sections
- UITableView, UITableViewCell, UITableViewHeaderFooterView, UITableViewDataSource, UITableViewDelegate
- Cells have basic styles: image and text label. Can customise this
- A cell can contain image, text, and any custom views
- Each section within a table can have a header and a footer that displays text or custom content
- UITableViewDelegate manages selections, configure section headings and footers, help to delete and reorder cells, and perform other actions
- Three types of selection styles: single, multiple, or none. (e.g., in normal mode and editing mode)

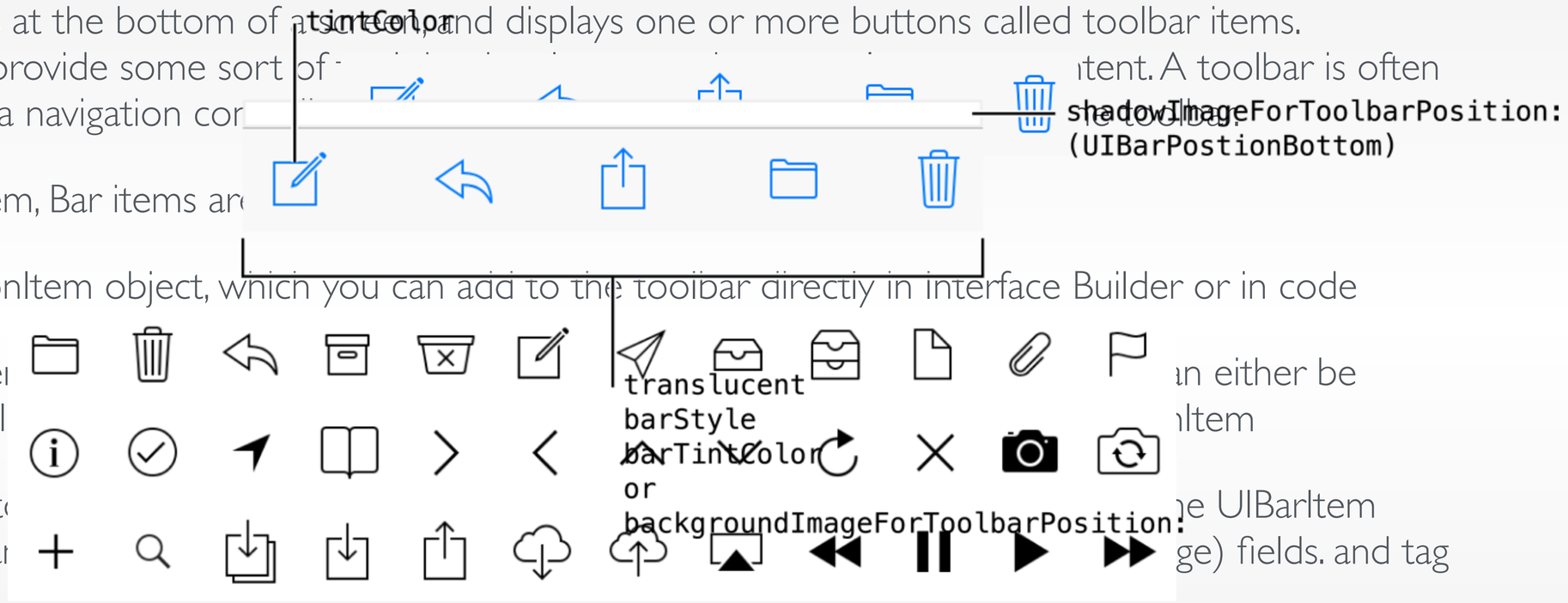




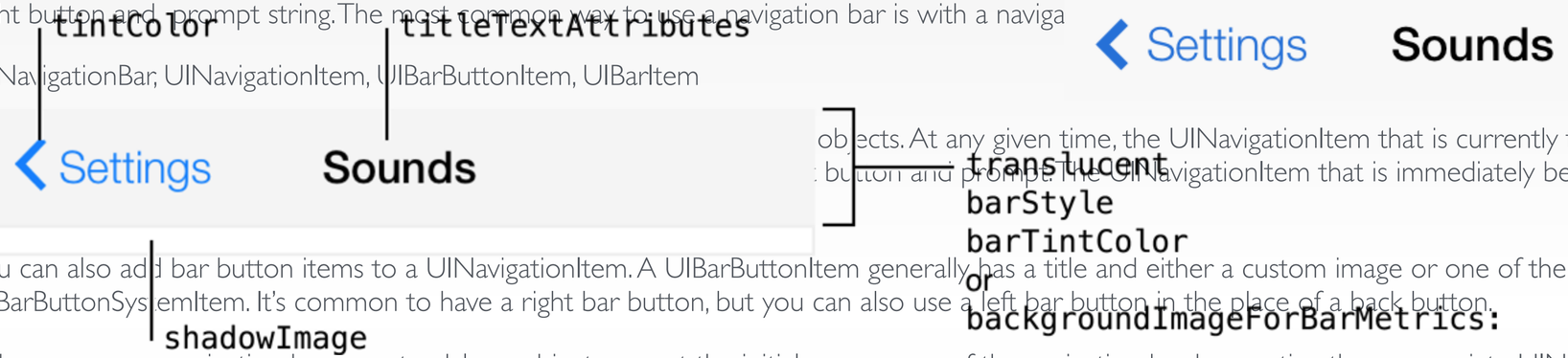
- different views in an app. Use a tab bar to organize information in your app by subtask. The most common way to use a tab bar is with a tab bar controller. You can also use a tab bar as a standalone object in your app.
- UITabBar, UITabBarItem
- Each tab bar item has a title, selected image, unse
- You can change the contents of a tab bar at runtime, and allow users t to your app. The tab bar delegate receives messages when the user customizes the tab
- A UITabBarController object manages the various tab views and view controllers, and the tab bar itself. If you use a tab bar controller, you should not use the UITabBar methods or properties to modify the tab
- two standard appearance styles: translucent white with dark text (default) or translucent black with light text. Use the barStyle
- translucent allows your content to show through underneath the bar.
- a tab bar item image will be automatically rendered as a template image within a tab bar, unless you explicitly set its rendering mode to UIImageRenderingModeAlwaysOriginal.



- A toolbar usually appears at the bottom of a screen and displays one or more buttons called toolbar items. Generally, these buttons provide some sort of [illegible] ntent. A toolbar is often used in conjunction with a navigation cor[illegible]
- UIToolbar, UIBarButtonItem, Bar items ar[illegible]
- Each item is a UIBarButtonItem object, which you can add to the toolbar directly in Interface Builder or in code
- You can specify the conte[illegible]n either be custom or take on the val[illegible]Item
- If you are using a bar butt[illegible]e UIBarItem level. For example, you ca[illegible]ge) fields. and tag
- A common way to create and manage a toolbar is in conjunction with a navigation controller. The navigation controller displays the toolbar and populates it with items from the currently visible view controller. Using a navigation controller is ideal for an app design where you want to change the contents of the toolbar dynamically. However, you should not use a navigation controller if your app does not have or need a navigation bar.



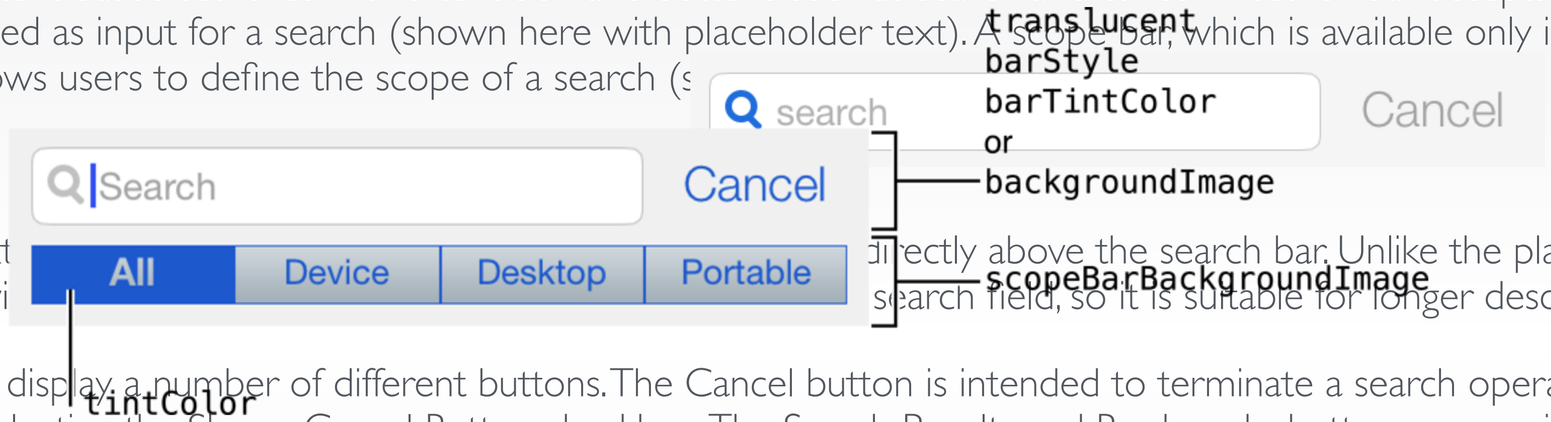
- A navigation bar is displayed at the top of the screen, and contains buttons for navigating through a hierarchy of screens. A navigation bar generally has a back button, a title, and a right button and a prompt string. The most common way to use a navigation bar is with a naviga in your app.



- UINavigationBar, UINavigationItem, UIBarButtonItem, UIBarItem
- Th objects. At any given time, the UINavigationItem that is currently the topItem of the stack determines the tit button and prompt. The UINavigationItem that is immediately below the topItem is the backItem, which de
- You can also add bar button items to a UINavigationItem. A UIBarButtonItem generally has a title and either a custom image or one of the system-supplied images listed in UIBarButtonSystemItem. It's common to have a right bar button, but you can also use a left bar button in the place of a back button.
- When you use a navigation bar as a standalone object, you set the initial appearance of the navigation bar by creating the appropriate UINavigationItem objects and adding them to the navigation bar object stack.
- You are also responsible for managing the stack of UINavigationItem objects when you use a navigation bar as a standalone object. You push new navigation items onto the stack using the pushNavigationItem:animated: method and pop items off the stack using the popNavigationItemAnimated: method. In addition to pushing and popping items, you can also set the contents of the stack directly using either the items property or the setItems:animated: method. You might use these methods at launch time to restore your interface to its previous state or to push or pop more than one navigation item at a time.

- Assign a custom delegate object to the delegate property and use that object to intercept messages sent by the navigation bar. Delegate objects must conform to the UINavigationBarDelegate protocol. The delegate notifications let you track when navigation items are pushed or popped from the stack. You use these notifications to update the rest of your app's user interface
- You can adjust the vertical position of a navigation bar's title, You can specify the font, text color, text shadow color, and text shadow offset for the title

- an interface for text-based searches with a text box and buttons such as search and cancel. A search bar accepts text from users, which can be used as input for a search (shown here with placeholder text). A scope bar, which is available only in conjunction with a search bar—allows users to define the scope of a search (s



- UISearchBar

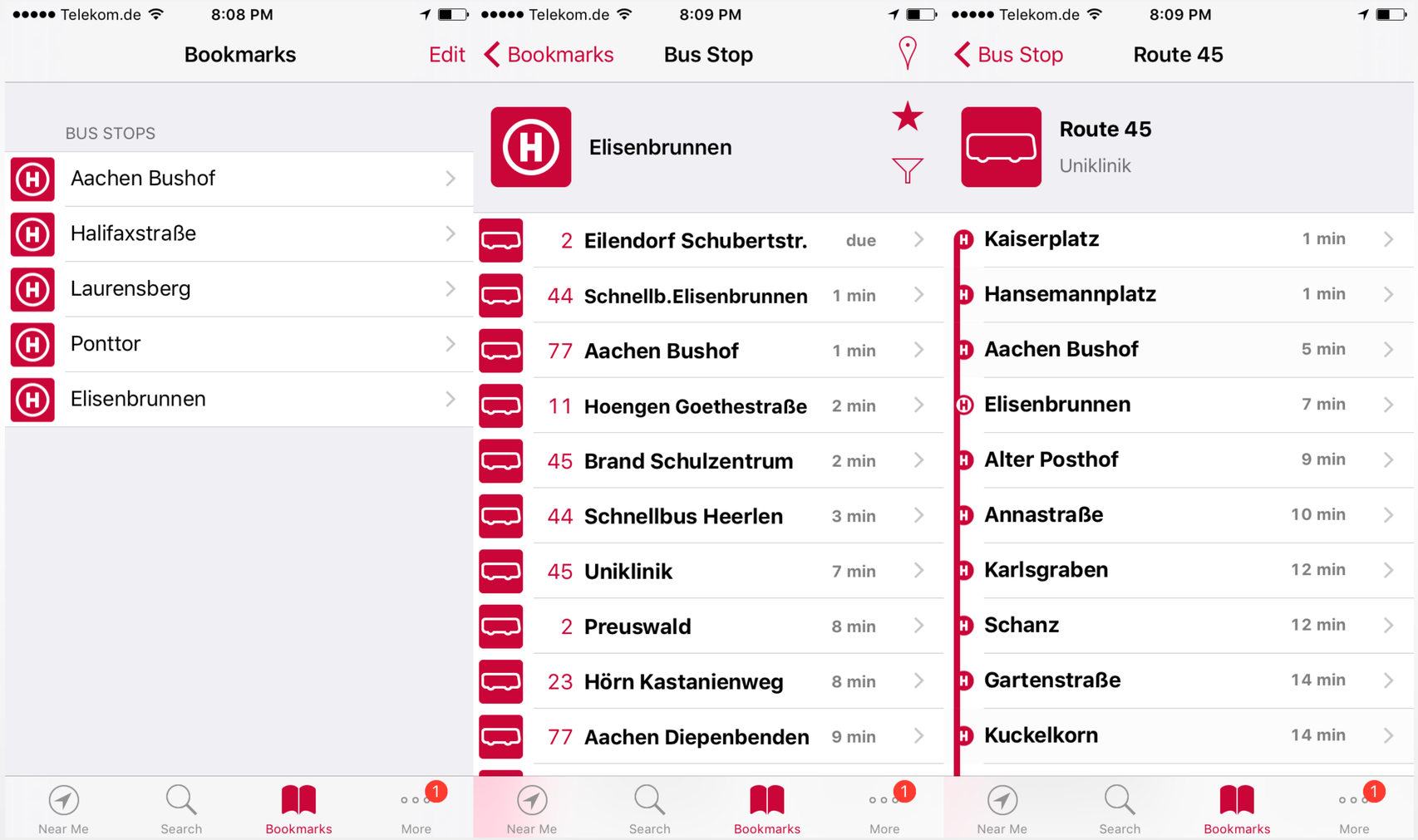
- The prompt text di rectly above the search bar. Unlike the placeholder text, the prompt text is vi search field, so it is suitable for longer descriptions or directions.

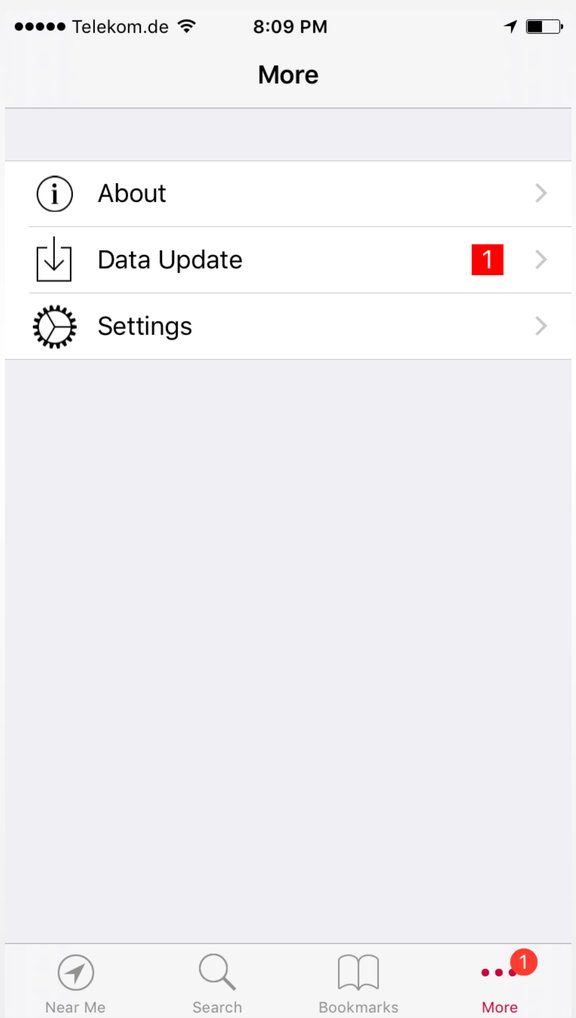
- Search bars can display a number of different buttons. The Cancel button is intended to terminate a search operation; you can display this button by selecting the Shows Cancel Button checkbox. The Search Results and Bookmarks buttons appear in the right side of search bar, and can be toggled to display those respective views.

- These buttons are merely user interface elements and have no functionality. You must implement the appropriate functionality yourself using the corresponding UISearchBarDelegate methods.

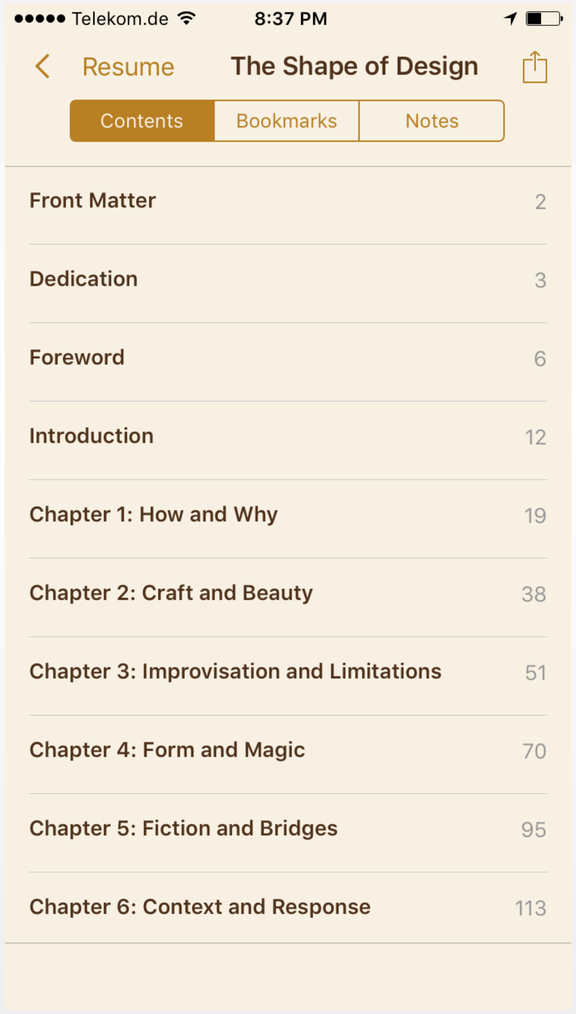
- Search bars need a delegate to handle user interaction. You implement the UISearchBarDelegate protocol on a delegate object to respond to user actions—for example, performing the search. Every search bar needs a delegate object that implements the UISearchBarDelegate protocol. The delegate is responsible for taking actions in response to user input such as editing the search text, starting or canceling a search, and tapping in the scope bar. At the very minimum, the delegate needs to perform a search after text is entered in the text field.

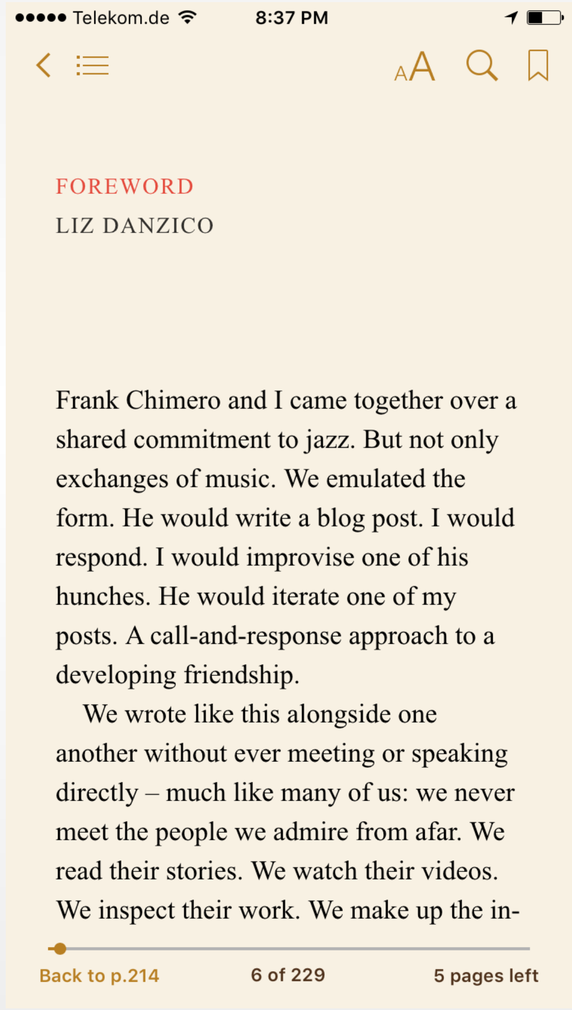
Hierarchical and Flat Nav.

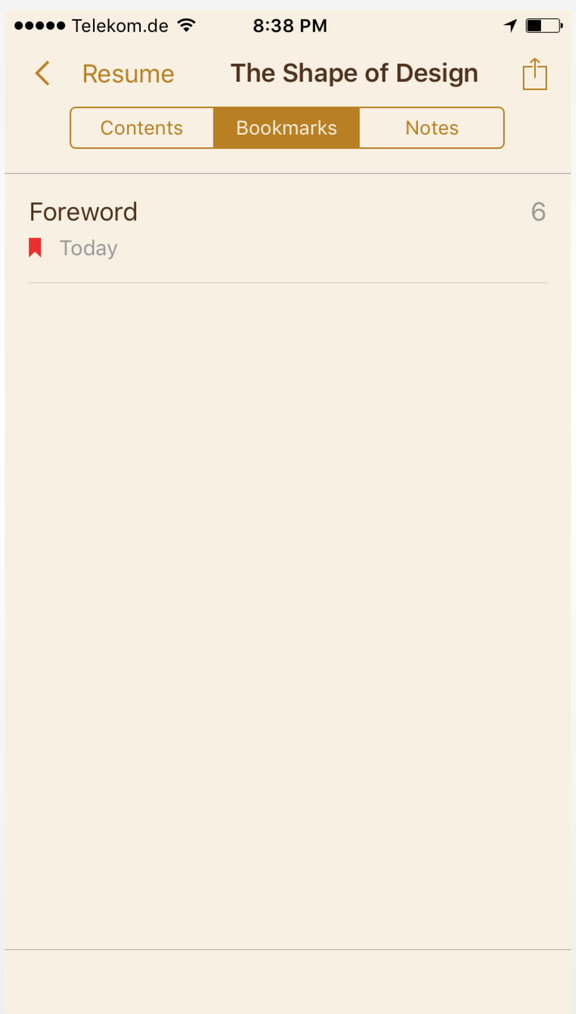




Content or Experience Driven Nav.







Pseudo Nav.

- Page control allows you to move between screens, each representing an instance of the same type or page. Indicates number of pages, and the se

What is iBooks controoler?

- Segmented control allows you to see gories of the content on the screen (it doesn't enable navigation to a new screen)

- Use a *temporary view* for a screen that users want to access from different contexts

Auto Layout

- Preferred layout management
- Allows you to create views that work both in portrait and landscape mode
- Available in iOS 6 and higher
- Spatial relationships expressed by constraints

Auto Layout Constraints

- Constraints are mathematical expressions
  - <=, ==, =>
- Constraints have a priority level
- The runtime tries to solve the system of equations

Adding Constraints