iPhone Programming

CMSC 498i – Spring 2010

Text

Lecture #11 – Chuck Pisula

This Week

- **Midterm Exam**
  - This Thursday
  - Material
    - Anything from lectures, labs, reading assignments
    - Questions focus on concepts and pseudo code, not APIs
- **Lab This Friday**
  - Work on “View Controller Lab”, or …
  - New **extra credit** “Picker Lab” assignment
  - Both due Thursday after the break

Today's Topics

- Getting Text Input
- Configuring UI and Behavior
- Editable Text Containers – `UITextField`, `UITextView`

- If Time…
  - Understanding unicode and encodings
  - NSString's companion classes

Keyboard

Display and Configuration

Virtual Keyboard

- Appears When Needed

Virtual Keyboard

- Appears When Needed

Virtual Keyboard

- Appears when needed
- Portrait and Landscape

Virtual Keyboard

- Appears when needed
- Portrait and Landscape

Virtual Keyboard

- Appears when needed
- Portrait and Landscape
- Adapted to task

Virtual Keyboard

- Appears when needed
- Portrait and Landscape
- Adapted to task

URL Entry

Virtual Keyboard

- Appears when needed
- Portrait and Landscape
- Adapted to task

Email Addresses

Virtual Keyboard

- Appears when needed
- Portrait and Landscape
- Adapted to task

Phone Numbers

Virtual Keyboard

- Appears when needed
- Portrait and Landscape
- Adapted to task

Numbers

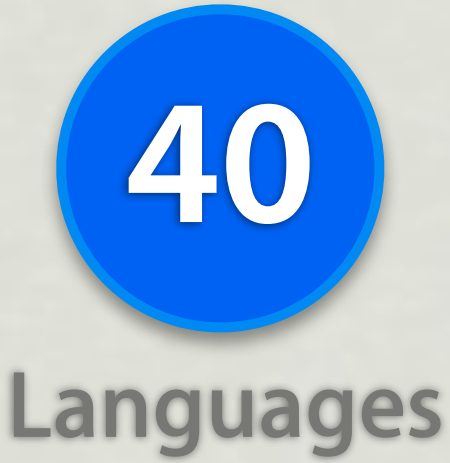
Virtual Keyboard

- Appears when needed
- Portrait and Landscape
- Adapted to task

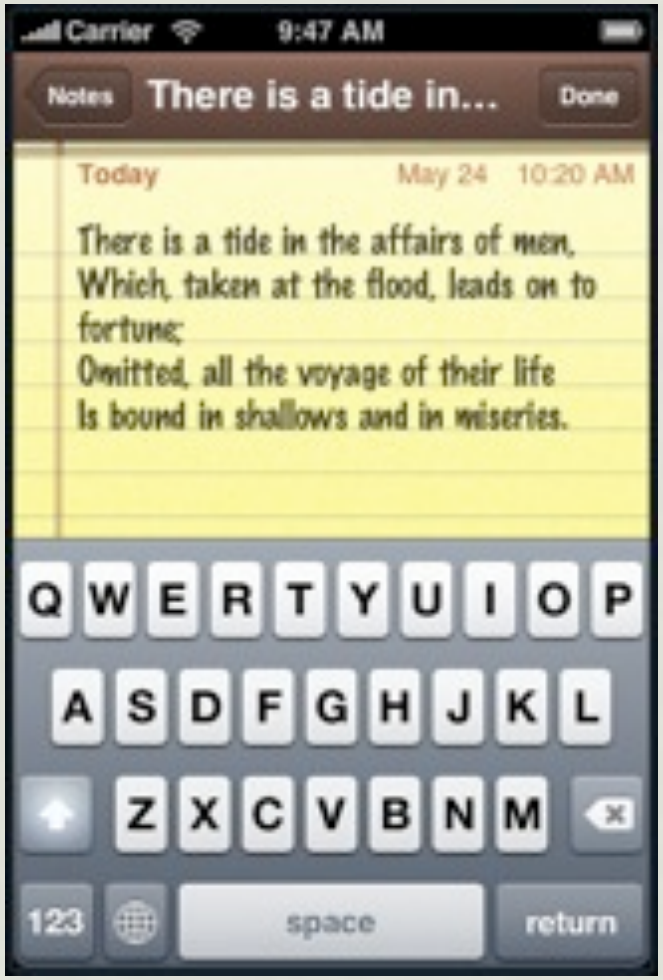
Multi–Line Input

Virtual Keyboard

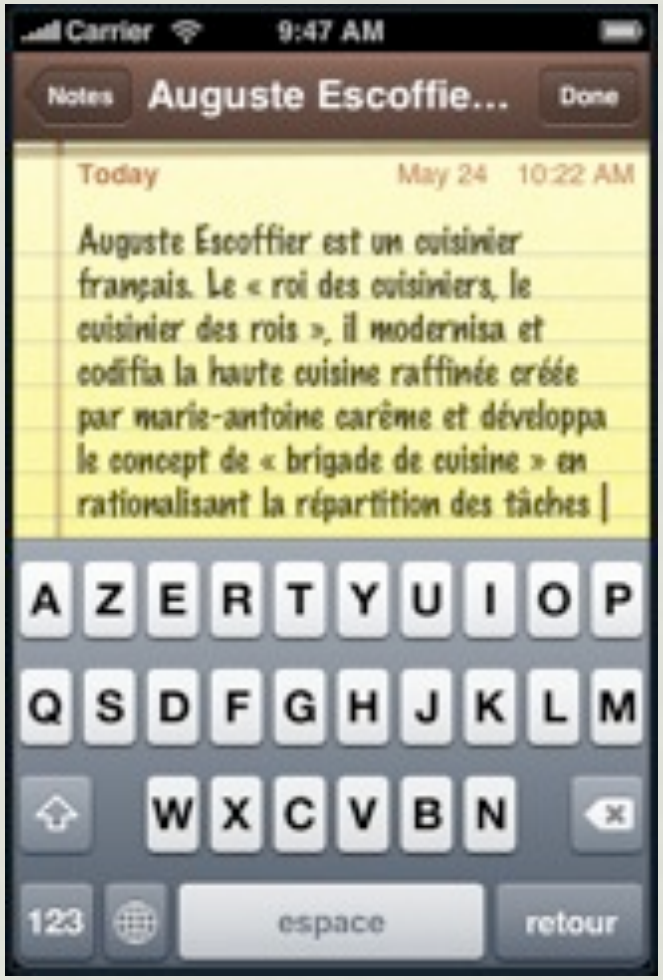
- Appears when needed
- Portrait and Landscape
- Adapted to task
- One device – Many languages



English


Carrier
9:47 AM
Notes
There is a tide in...
Done
Today
May 24 10:20 AM
There is a tide in the affairs of men,
Which, taken at the flood, leads on to fortune;
Omitted, all the voyage of their life
Is bound in shallows and in miseries.
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

French


Carrier
9:47 AM
Notes
Auguste Escoffie...
Done
Today
May 24 10:22 AM
Auguste Escoffier est un cuisinier français. Le « roi des cuisiniers, le cuisinier des rois », il modernisa et codifia la haute cuisine raffinée créée par marie-antoine carême et développa le concept de « brigade de cuisine » en rationalisant la répartition des tâches
A Z E R T Y U I O P
Q S D F G H J K L M
W X C V B N
123
espace
retour

Russian


Carrier
9:47 AM
Notes
Шáхматы — наст...
Done
Today
May 24 10:10 AM
Шáхматы — настольная логическая игра, сочетающая в себе элементы искусства, науки и спорта. Одна из древнейших игр на земле, сохранившихся до нашего времени; долгое время считалась игрой королей из аристократов.
Й Ц У К Е Н Г Ш Щ З Х
Ф Ы В А П Р О Л Д Ж Э
Я Ч С М И Т Ь Б Ю
123
Пробел
Возврат

Japanese Kana

Chinese Pinyin

Chinese Handwriting

- Simplified and Traditional

Getting Text Input

- Can not interact with Keyboard directly

Getting Text Input

- Can not interact with Keyboard directly
- Interacts with editable “Text Containers” instead of keyboard
  - Configure keyboard – set attributes on the container
  - Gather input – notifications, delegation, and action messages

Getting Text Input

- Can not interact with Keyboard directly
- Interacts with editable “Text Containers” instead of keyboard
  - Configure keyboard – set attributes on the container
  - Gather input – notifications, delegation, and action messages
- Example editable text containers
  - `UITextField`, `UITextView`
  - `UILabel` is not editable, and does not interact with the keyboard

Presenting The Keyboard

- UIKit presents keyboard
    - When an editable text container **becomes first responder**
- Keyboard will slide up into place at the bottom
- May need to make sure your text view stays visible

Presenting The Keyboard

- UIKit presents keyboard
  - When an editable text container **becomes first responder**
- Keyboard will slide up into place at the bottom
- May need to make sure your text view stays visible
  - Watch for keyboard visibility notifications

```
NSString *const UIKeyboardWillShowNotification;
NSString *const UIKeyboardDidShowNotification;
NSString *const UIKeyboardWillHideNotification;
NSString *const UIKeyboardDidHideNotification;
```

Presenting The Keyboard

- UIKit presents keyboard
  - When an editable text container **becomes first responder**
- Keyboard will slide up into place at the bottom
- May need to make sure your text view stays visible
  - Watch for keyboard visibility notifications

```
NSString *const UIKeyboardWillShowNotification;
NSString *const UIKeyboardDidShowNotification;
NSString *const UIKeyboardWillHideNotification;
NSString *const UIKeyboardDidHideNotification;
```

  - Get information about the keyboard animation

```
NSString *const UIKeyboardBoundsUserInfoKey
NSString *const UIKeyboardAnimationDurationUserInfoKey
NSString *const UIKeyboardAnimationCurveUserInfoKey
```

Becoming First Responder

- View must be able to become first responder

- View must be told to become first responder

Becoming First Responder

- View must be able to become first responder

```
- (BOOL)canBecomeFirstResponder;
```

- View must be told to become first responder

```
[aView becomeFirstResponder]
```

Becoming First Responder

- View must be able to become first responder

```
- (BOOL)canBecomeFirstResponder;
```

- View must be told to become first responder

```
[aView becomeFirstResponder]
```

- Active first responder sent `-resignFirstResponder`

Becoming First Responder

- View must be able to become first responder

```
- (BOOL)canBecomeFirstResponder;
```

- View must be told to become first responder

```
[aView becomeFirstResponder]
```

- Active first responder sent `-resignFirstResponder`
- Automatic for editable text containers (`UITextField`, etc...)
  - `-canBecomeFirstResponder` returns `YES`
  - `-becomeFirstResponder` sent when tapped

Presenting The Keyboard

- What if you want don’t want the keyboard to animate up?

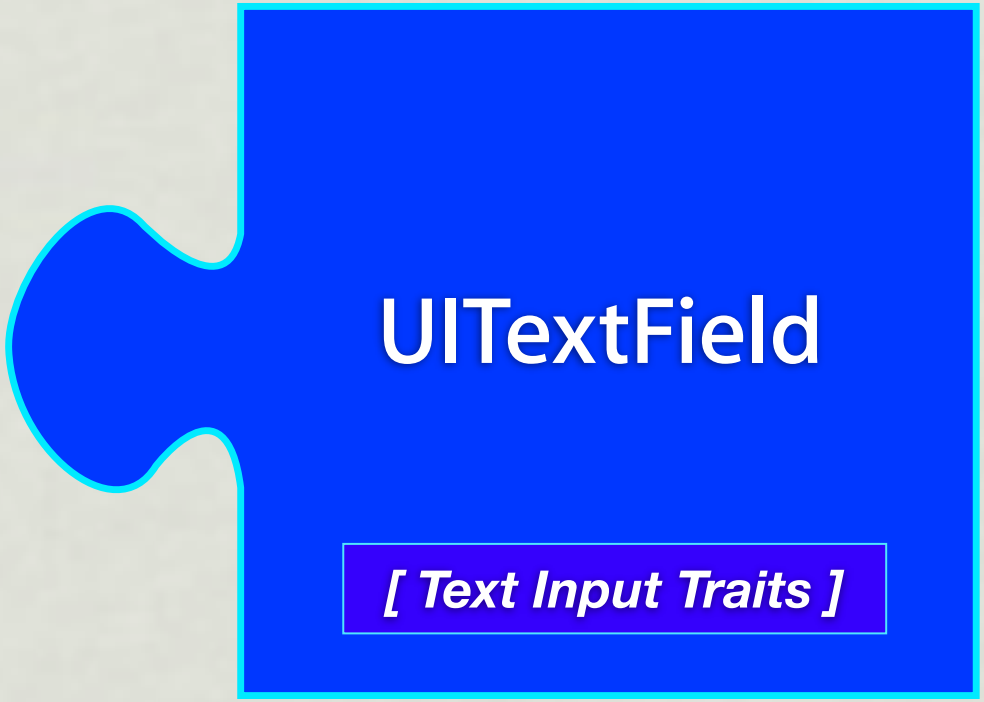
Presenting The Keyboard

- What if you want don't want the keyboard to animate up?

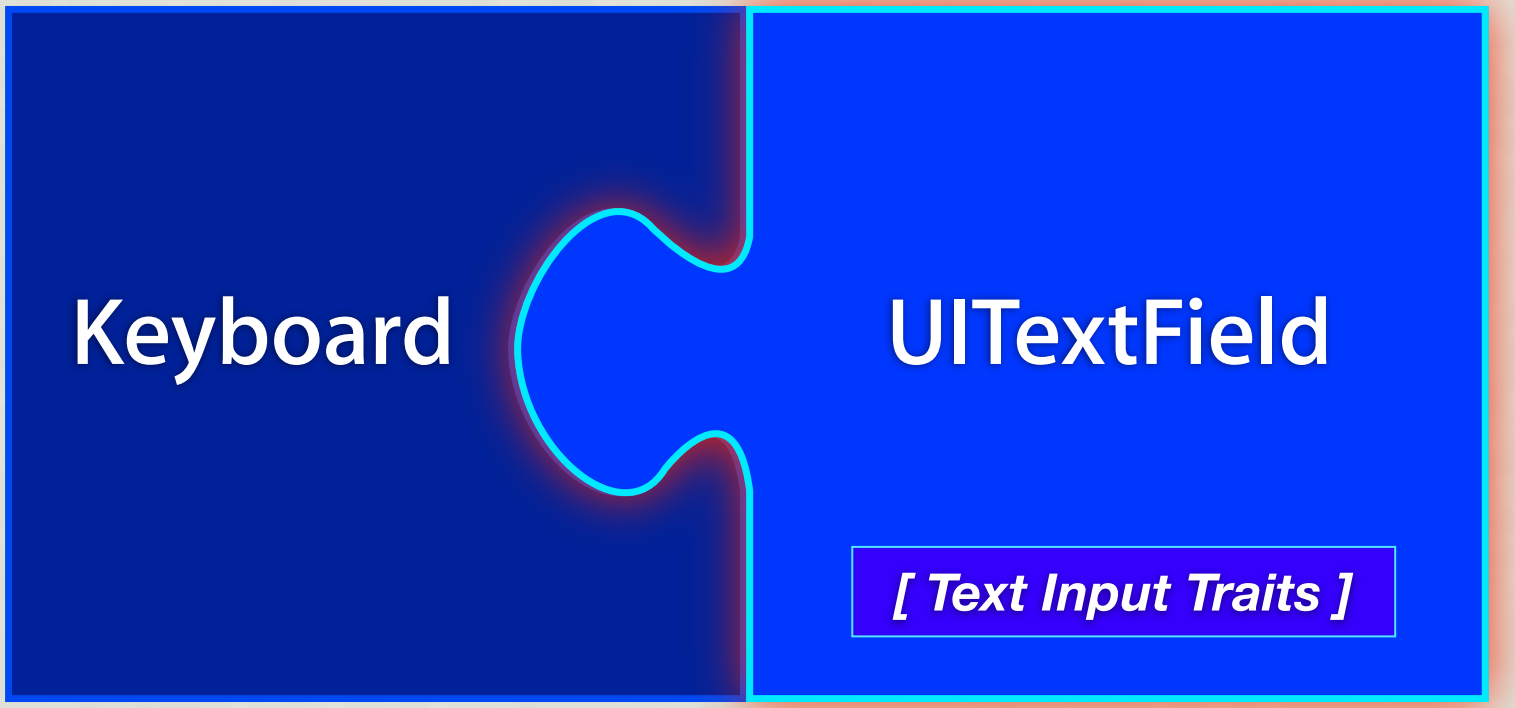
Presenting The Keyboard

- What if you want don't want the keyboard to animate up?
- In your `UIViewController -viewWillAppear:`
  - This is before the view is added to the window…
  - Make your text view first responder
  - UIKit will handle the rest

Presenting The Keyboard

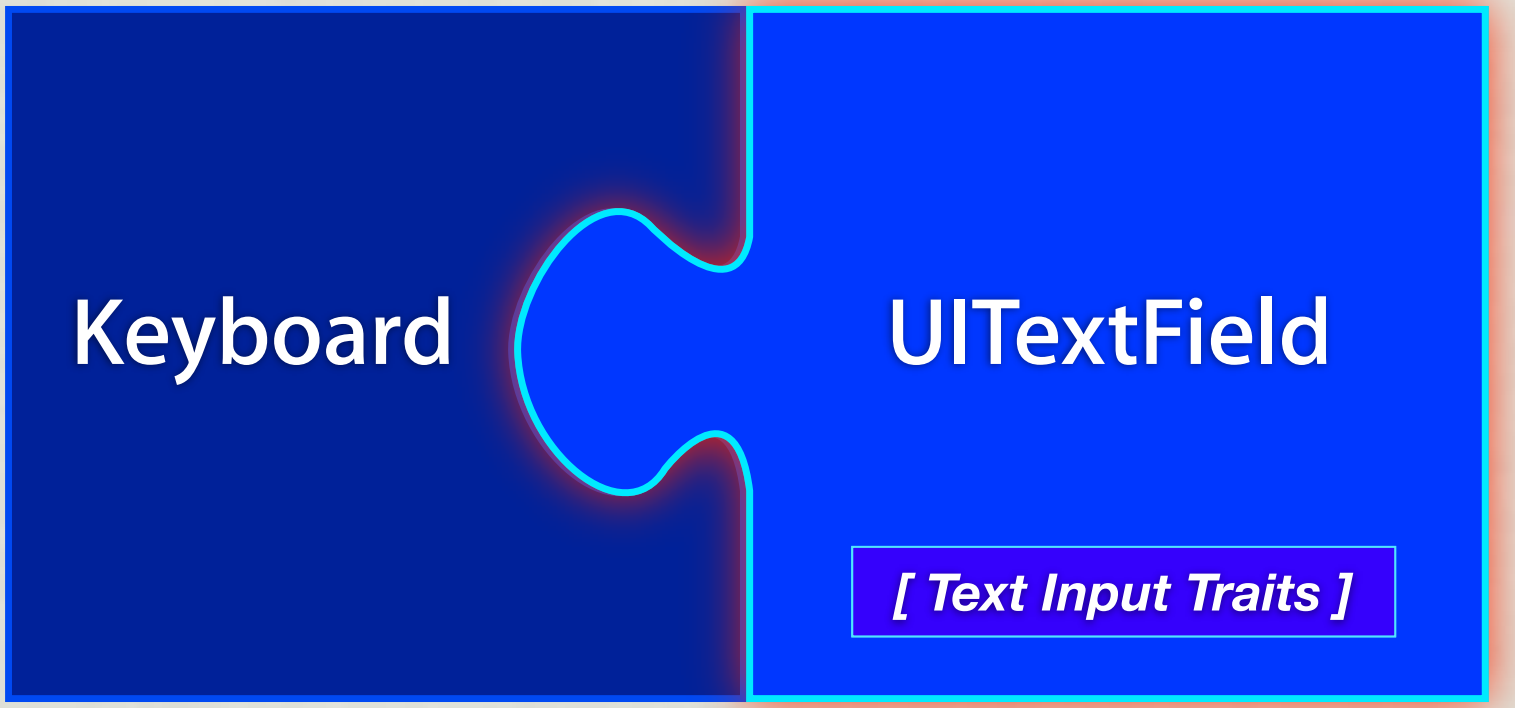


Presenting The Keyboard



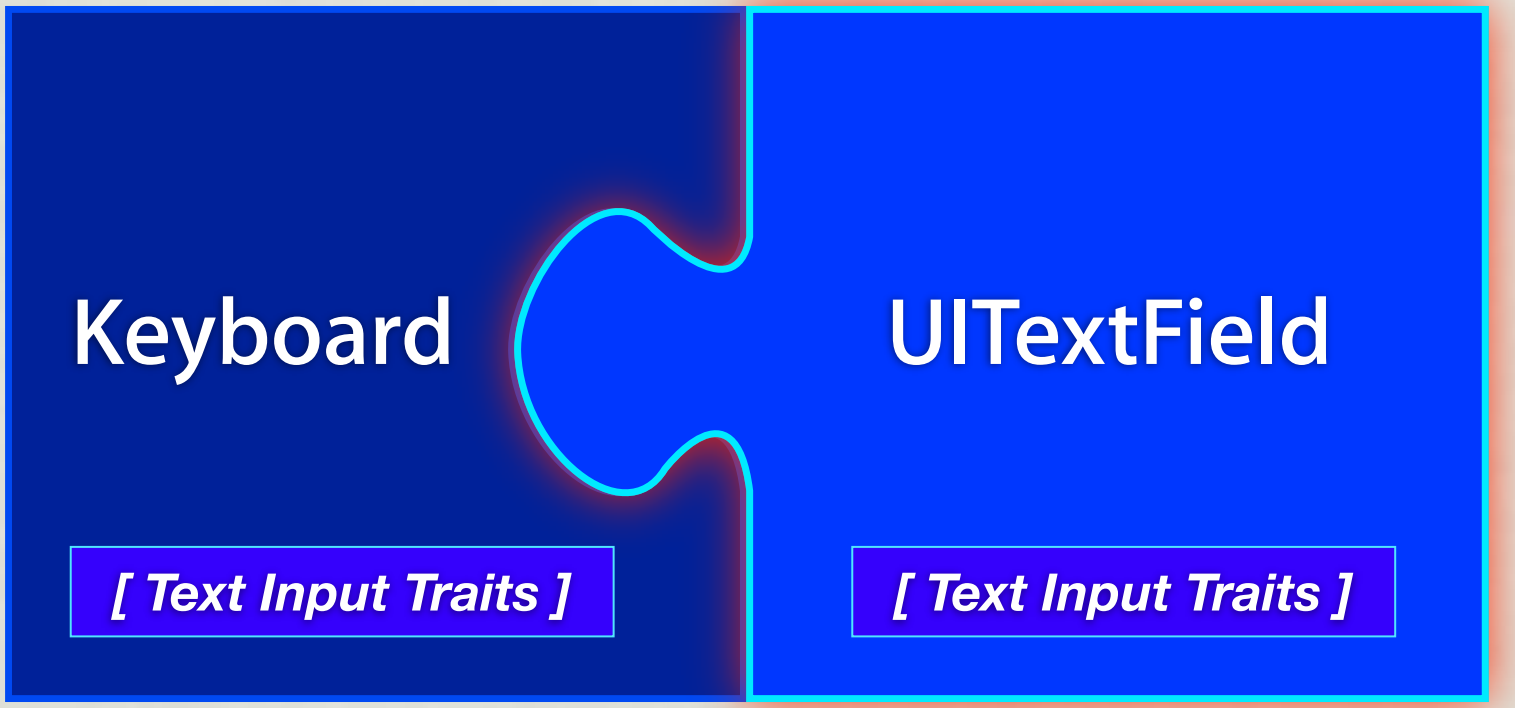
Become First Responder

Configuring The Keyboard



Become First Responder

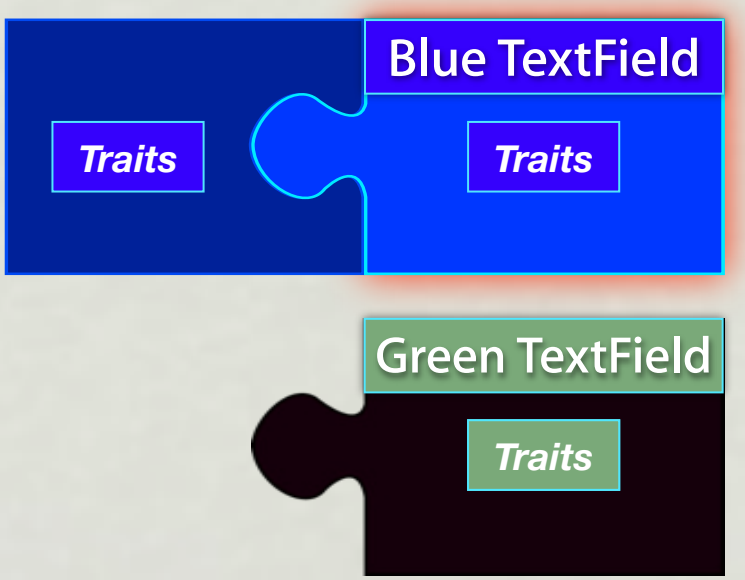
Configuring The Keyboard



Become First Responder

Changing First Responder

- Blue already first responder…
- Green told to “become first responder” for some reason

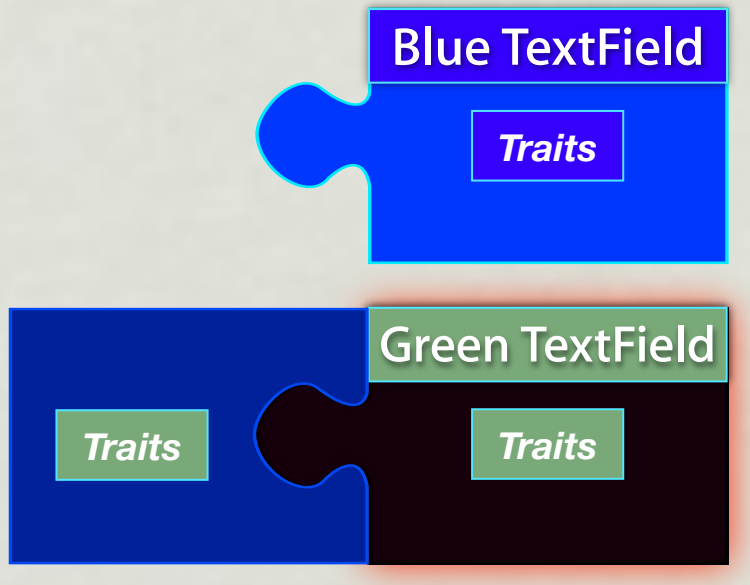


Changing First Responder

- Blue already first responder…
- Green told to “become first responder” for some reason
- Blue automatically told to “resign first responder” status

Changing First Responder

- Blue already first responder…
- Green told to “become first responder” for some reason
- Blue automatically told to “resign first responder” status
- Finally, Green is the official first responder



Dismissing The Keyboard

- No “dismiss” keyboard API

Dismissing The Keyboard

- No “dismiss” keyboard API
- Keyboard visible when the first responder has text input traits

Dismissing The Keyboard

- No “dismiss” keyboard API
- Keyboard visible when the first responder has text input traits
- To dismiss the keyboard resign first responder status

Dismissing The Keyboard

- No “dismiss” keyboard API
- Keyboard visible when the first responder has text input traits
- To dismiss the keyboard resign first responder status
- Keyboard will then automatically slide out

Keyboard

Configuration

Configuring The Keyboard

- Keyboard takes on traits of its target, the first responder
- `UITextInputTraits`
  - Protocol that all editable text containers support
  - Defines configurable options

UITextInputTraits

- Keyboard Type
- Appearance
- Secure Entry
- Capitalization
- Auto Correction
- Return Key Behavior

Configuration – Keyboard Type

- `UIKeyboardTypeDefault`
  - Displays users choice of keyboard (whatever language they want)
- `UIKeyboardTypeASCIICapable`
  - Not a generic Unicode input method, ASCII only
- `UIKeyboardTypeURL`
  - Features “.”, “/”, and “.com” prominently
- `UIKeyboardTypeNumberPad`
  - Displays numbers 0 - 9. Designed for PIN entry.
- Others – `UIKeyboardTypeNamePhonePad`, etc…

Configuration – Appearance

- UIKeyboardAppearanceDefault

Configuration – Appearance

- **UIKeyboardAppearanceDefault**
- **UIKeyboardAppearanceAlert** – darker keyboard UI suitable for displaying with an alert panel

Configuration – Behavior

- `UITextAutocapitalizationType`
  - None, Words, Sentences, All Characters
- `UITextAutocorrectionType`
  - Default – Use the user's choice from the Settings App
  - No, Yes – Force a particular setting

Configuration – Return Key

- `UIReturnKeyType`
  - Default – set the return key text to “return”
  - Go, Search, Send
  - Google, Yahoo
- Return Key Enabling
  - `BOOL enablesReturnKeyAutomatically`
  - Set this to `YES` if you want the return key disabled when there is no text in the editable text container

Text Containers

Working With Text Containers

- `UITextField` – single line editor
- `UITextView` – multi–line editor

Working With Text Containers

- `UITextField` – single line editor
- `UITextView` – multi–line editor
- API to grab text from the editable text containers

```
// UITextField.h
@property(nonatomic,copy) NSString *text;

// UITextView.h
@property(nonatomic,copy) NSString *text;
```

Working With Text Containers

- `UITextField` – single line editor
- `UITextView` – multi–line editor
- API to grab text from the editable text containers

```
// UITextField.h
@property(nonatomic,copy) NSString *text;

// UITextView.h
@property(nonatomic,copy) NSString *text;
```

- Editing Sequence and Change “Events”
  - Did begin editing
  - Did end editing
  - Did change text

Text “Events”

- Mechanism you choose depends on your situation…
- “Did” events available to all mechanisms
- “Should” available only to the **delegate**

Event	Notification	Delegation
did begin editing	√ √	√ √
should begin editing		√ √
did end editing	√ √	√ √
should end editing		√ √
did change	√ √	√ √
should change		√ √
– other options –		√ √

Text “Events”

- Mechanism you choose depends on your situation…
- “Did” events available to all mechanisms
- “Should” available only to the **delegate**
- Target / Action only for UITextField

Event	Notification	Delegation	Target / Action
did begin editing	√ √	√ √	√
should begin editing		√ √	
did end editing	√ √	√ √	√
should end editing		√ √	
did change	√ √	√ √	√
should change		√ √	
– other options –		√ √	

UITextField “Events”

- Target / Action

```
// UIControl.h
UIControlEventEditingDidBegin
UIControlEventEditingChanged
UIControlEventEditingDidEnd
UIControlEventEditingDidEndOnExit
```

- Example

```
// Set up action to be called when editing ends (could do this in IB)
UITextField *myTextField = ...;

[myTextField addTarget:self
                action:@selector(textDidEndEditing)
      forControlEvents:UIControlEventEditingDidEnd];
```

UITextField “Events”

- Notifications

```
NSString *const UITextFieldTextDidBeginEditingNotification;
NSString *const UITextFieldTextDidChangeNotification;
NSString *const UITextFieldTextDidEndEditingNotification;
```

UITextField “Events”

- Notifications

```
NSString *const UITextFieldTextDidBeginEditingNotification;
NSString *const UITextFieldTextDidChangeNotification;
NSString *const UITextFieldTextDidEndEditingNotification;
```

- Delegation

```
// Begin
- (BOOL)textFieldShouldBeginEditing:(UITextField *)textField;
- (void)textFieldDidBeginEditing:(UITextField *)textField;

// End
- (BOOL)textFieldShouldEndEditing:(UITextField *)textField;
- (void)textFieldDidEndEditing:(UITextField *)textField;

// Change
- (BOOL)textField:(UITextField *)textField
    shouldChangeCharactersInRange:(NSRange)range
                replacementString:(NSString *)string;
```

UITextField – Editing Sequence

- Keyboard shows, but does not automatically dismiss for you
- Ending Editing
  - To force editing to end, `-resignFirstResponder`
- Moving focus (first responder) to another text field
  - Delegate's `-textFieldShouldReturn:` called when return key tapped
  - Either just `-resignFirstResponder` there to dismiss the keyboard
  - …or, pick another text field to `-becomeFirstResponder`

UITextField – Editing Sequence

- Keyboard shows, but does not automatically dismiss for you
- Ending Editing
  - To force editing to end, `-resignFirstResponder`
- Moving focus (first responder) to another text field
  - Delegate's `-textFieldShouldReturn:` called when return key tapped
  - Either just `-resignFirstResponder` there to dismiss the keyboard
  - …or, pick another text field to `-becomeFirstResponder`

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    // When the return button is pressed, dismiss the keyboard!
    [textField resignFirstResponder]; // end editing, send end edit actions, etc…
    return YES; // dismiss keyboard
}
```

UITextView “Events”

- Not a `UIControl`, so no target / action
- Notification and delegation API ~ same as `UITextField's`

UITextView “Events”

- Not a `UIControl`, so no target / action
- Notification and delegation API ~ same as `UITextField's`

```
NSString *const UITextViewTextDidBeginEditingNotification;
NSString *const UITextViewTextDidChangeNotification;
NSString *const UITextViewTextDidEndEditingNotification;
```

UITextView “Events”

- Not a `UIControl`, so no target / action
- Notification and delegation API ~ same as `UITextField's`

```
NSString *const UITextViewTextDidBeginEditingNotification;
NSString *const UITextViewTextDidChangeNotification;
NSString *const UITextViewTextDidEndEditingNotification;
```

```
// Begin
- (BOOL)textViewShouldBeginEditing:(UITextView *)textView;
- (void)textViewDidBeginEditing:(UITextView *)textView;

// End
- (BOOL)textViewShouldEndEditing:(UITextView *)textView;
- (void)textViewDidEndEditing:(UITextView *)textView;

// Change
- (BOOL)textView:(UITextView *)textView
         shouldChangeTextInRange:(NSRange)range
                 replacementText:(NSString *)string;
```

UITextView “Events”

- Other delegate methods

```
// Called selected text range changes, including cursor position
- (void)textViewDidChangeSelection:(UITextView *)textView;
```

- Notice there is no equivalent to “`-textFieldShouldReturn:`”
  - Return key needs to go to the `UITextView` since it is multi–line!

UIWebView

- Not covering in detail today…
- Use to embed web content in an application
  - Safari in a `UIView`!
- Functionality
  - Request URL (http, local file, …) to be loaded
  - Control UI – Back, Forward, Reload, Stop
  - Invoke javascript in the loaded page
  - Track loading progress as `UIWebView's` delegate

Summary

Putting It All Together

Putting It All Together

- `UITextView` or `UITextField` becomes first responder
- Keyboard slides up
  - Appearance notifications posted
  - Keyboard adopts the “input traits”
- While typing
  - Should change, did change notifications / delegate messages
- Dismiss the keyboard
  - User taps in another view that can become first responder
  - Manually resign, or transfer first responder in code

NSString

Unicode and Companion Classes

NSString And Unicode

- Why Understand Unicode?
  - An industry standard for representing all the worlds languages
  - `NSString` represents an array of Unicode characters
  - APIs and `NSString` documentation use terminology from Unicode
  - Understand which APIs to use

Terms

- **Unicode** – standard which defines a “world” character set (UCS) mapping, and encoding schemes

Terms

- **Unicode** – standard which defines a “world” character set (UCS) mapping, and encoding schemes
- **Code Point** – numeric value assigned to a unicode character

Terms

- **Unicode** – standard which defines a “world” character set (UCS) mapping, and encoding schemes
- **Code Point** – numeric value assigned to a unicode character
- **Glyph** – graphical representation of a character, the on screen rendering; there is a many-to-many relation between characters and glyphs

Terms

- **Unicode** – standard which defines a “world” character set (UCS) mapping, and encoding schemes
- **Code Point** – numeric value assigned to a unicode character
- **Glyph** – graphical representation of a character, the on screen rendering; there is a many-to-many relation between characters and glyphs
- **UTF-8** – UCS encoding maximally compatible with ASCII

Terms

- **Unicode** – standard which defines a “world” character set (UCS) mapping, and encoding schemes
- **Code Point** – numeric value assigned to a unicode character
- **Glyph** – graphical representation of a character, the on screen rendering; there is a many-to-many relation between characters and glyphs
- **UTF-8** – UCS encoding maximally compatible with ASCII
- **Precomposed** – character that has its own a code point and equivalent decomposed sequences of others.

Terms

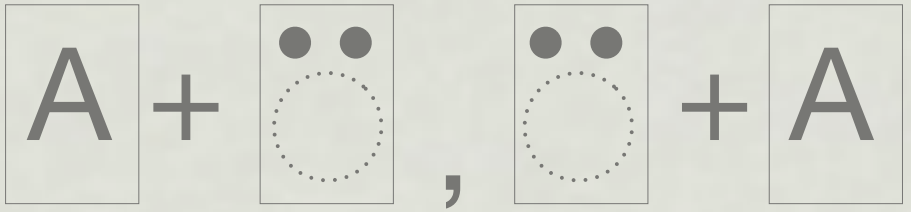
- **Unicode** – standard which defines a “world” character set (UCS) mapping, and encoding schemes
- **Code Point** – numeric value assigned to a unicode character
- **Glyph** – graphical representation of a character, the on screen rendering; there is a many-to-many relation between characters and glyphs
- **UTF-8** – UCS encoding maximally compatible with ASCII
- **Precomposed** – character that has its own a code point and equivalent decomposed sequences of others.
- **Canonical Equivalence** – Character orders may differ, but a “canonical” representation of them may be equivalent

Composed Characters

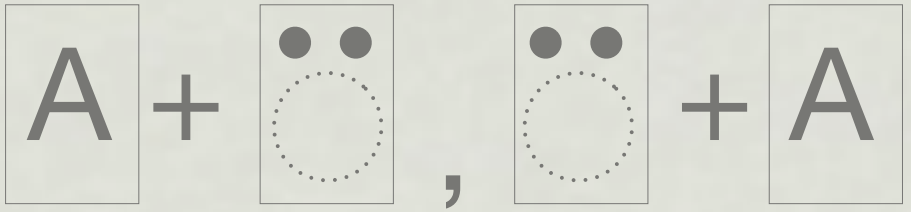
- Precomposed

Composed Characters

- Precomposed 

- Decomposed 

Composed Characters

- Precomposed 
- Decomposed 
- Canonical 

Code Point

- **Definition** - Numeric value used to represent a character

Code Point

- **Definition** - Numeric value used to represent a character
- Unicode defines a table representing practically all languages
  - Most commonly used characters fall in a 16-bit range
  - Unicode defines 21 bits of its 32 bit space; >1 million characters

Code Point

- **Definition** - Numeric value used to represent a character
- Unicode defines a table representing practically all languages
  - Most commonly used characters fall in a 16-bit range
  - Unicode defines 21 bits of its 32 bit space; >1 million characters
- Each abstract character, given a name and numeric value
  - U+0391 - GREEK CAPITAL LETTER ALPHA
  - **Using in an NSString** `@"\u0391"`
  - **Find Tables at** http://www.fileformat.info/info/unicode

Encodings

- **Definition** – specifies how codes map to sequences of bytes

Encodings

- **Definition** – specifies how codes map to sequences of bytes
- ASCII encoding – 7 bit encoding of english/control characters

Encodings

- **Definition** – specifies how codes map to sequences of bytes
- ASCII encoding – 7 bit encoding of english/control characters
- Unicode specifies several encoding forms
  - UTF-8, UTF-16, UTF-32
  - Each maps a unicode character to one or more “units”
    - For example, UTF-8 “unit” is one byte, sequence of up to 4
    - UTF-8/16 are use variable number, UTF-32 uses one unit always
  - Each has advantages in different environments

Encodings

- **Definition** – specifies how codes map to sequences of bytes
- ASCII encoding – 7 bit encoding of english/control characters
- Unicode specifies several encoding forms
  - UTF-8, UTF-16, UTF-32
  - Each maps a unicode character to one or more “units”
    - For example, UTF-8 “unit” is one byte, sequence of up to 4
    - UTF-8/16 are use variable number, UTF-32 uses one unit always
  - Each has advantages in different environments
- **UTF-8 is typically most compact, and is ASCII compatible**

Character / Letter

- The word “character” is used in multiple ways
  - `char` – ASCII base type in C programming
  - `(unichar)characterAtIndex:` – UTF-16 unit returned by NSString
  - **Neither of these is what the user thinks of as a “letter”**

Character / Letter

- The word “character” is used in multiple ways
  - `char` – ASCII base type in C programming
  - `(unichar)characterAtIndex:` – UTF-16 unit returned by NSString
  - **Neither of these is what the user thinks of as a “letter”**

- **“letter” may require multiple UTF-8 or UTF-16 characters**
  - Code point could require more than 16 bits, or…
  - Decomposition: Ä can legally be represented as Ä, or A + ◌̈

What To Understand

What To Understand

- **NSString API presented in terms of UTF-16 units**
  - -length returns the number of UTF-16 units making up the string
  - -characterAtIndex: returns UTF-16 represented units (unichar)
    - Internal storage is not necessarily UTF-16
    - The actual Unicode character might span multiple indexes

What To Understand

What To Understand

- **Direct ‘character’ APIs should be avoided**
  - Use NSString companion classes
  - “character” != byte
  - Do not break up characters that are part of a decomposed seq.

What To Understand

- **Direct ‘character’ APIs should be avoided**
  - Use NSString companion classes
  - “character” != byte
  - Do not break up characters that are part of a decomposed seq.
- Example: Given the NSString storing: A + 
  - **User considers this a single “Letter” (displays as Ä)**

What To Understand

- **Direct ‘character’ APIs should be avoided**
  - Use NSString companion classes
  - “character” != byte
  - Do not break up characters that are part of a decomposed seq.
- Example: Given the NSString storing: A + 
  - **User considers this a single “Letter” (displays as Ä)**
- `-characterAtIndex:0`, returns “A”

What To Understand

- **Direct ‘character’ APIs should be avoided**
  - Use NSString companion classes
  - “character” != byte
  - Do not break up characters that are part of a decomposed seq.
- Example: Given the NSString storing: A + 
  - **User considers this a single “Letter” (displays as Ä)**
- `-characterAtIndex:0`, returns “A”
- `-length` returns 2

What To Understand

- **Direct ‘character’ APIs should be avoided**
  - Use NSString companion classes
  - “character” != byte
  - Do not break up characters that are part of a decomposed seq.
- Example: Given the NSString storing: A + 
  - **User considers this a single “Letter” (displays as Ä)**
- `-characterAtIndex:0`, returns “A”
- `-length` returns 2
- `-rangeOfComposedCharacterSequenceAtIndex:0` returns `{0,2}`

API Explanations

- `unichar` – value will be a UTF-16 unit
- **composed characters** – characters that should be combine
  - Ä is a precomposed version of A + 

API Explanations

- `unichar` – value will be a UTF-16 unit
- **composed characters** – characters that should be combine
  - Ä is a precomposed version of A + 
  - **What about this?** 其他

NSString Companion Classes

- `NSScanner` – Scans and interpret values from an `NSString`
- `NSCharacterSet` – Represents a set of unicode characters
  - An example – process file data by line

```
// lineSeparatorSet represents ‘\n’, ‘\r’, ...
NSCharacterSet *lineSeparatorSet = [NSCharacterSet newlineCharacterSet]

// Load a string from a file
NSString *string = [NSString stringWithContentsOfFile:...];

// Separate into individual lines
NSArray *lines = [string componentsSeparatedByCharactersInSet:lineSeparatorSet];
```

Midterm

This Thursday…