

CS193P - Lecture 11

iPhone Application Development

**Text Input
Presenting Content Modally**

Announcements

Announcements

- Paparazzi 3 assignment is due Wednesday 2/17

Announcements

- Paparazzi 3 assignment is due Wednesday 2/17
- This Friday's extra session will feature Evan Doll

Today's Topics

- Threading Wrap-Up
- iPhone Keyboards
- Customizing Text Input
- Presenting Content Modally

Alternatives to Threading

Alternatives to Threading

- Asynchronous (nonblocking) functions
 - Specify target/action or delegate for callback
 - **NSURLConnection** has synchronous and asynchronous variants

Alternatives to Threading

- Asynchronous (nonblocking) functions
 - Specify target/action or delegate for callback
 - **NSURLConnection** has synchronous and asynchronous variants
- Timers
 - One-shot or recurring
 - Specify a callback method
 - Managed by the run loop

Alternatives to Threading

- Asynchronous (nonblocking) functions
 - Specify target/action or delegate for callback
 - **NSURLConnection** has synchronous and asynchronous variants
- Timers
 - One-shot or recurring
 - Specify a callback method
 - Managed by the run loop
- Higher level constructs like **operations**

NSOperation

- Abstract superclass
- Manages thread creation and lifecycle
- Encapsulate a **unit of work** in an object
- Specify priorities and dependencies

Creating an NSOperation Subclass

Creating an NSOperation Subclass

- Define a **custom init method**

```
- (id)initWithSomeObject:(id)someObject
{
    self = [super init];
    if (self) {
        self.someObject = someObject;
    }
    return self;
}
```

Creating an NSOperation Subclass

- Define a **custom init method**

```
- (id)initWithSomeObject:(id)someObject
{
    self = [super init];
    if (self) {
        self.someObject = someObject;
    }
    return self;
}
```

- **Override -main method** to do work

```
- (void)main
{
    [someObject doLotsOfTimeConsumingWork];
}
```

NSOperationQueue

- Operations are typically scheduled by **adding to a queue**
- Choose a maximum number of concurrent operations
- Queue runs operations based on priority and dependencies

Using an NSInvocationOperation

- Concrete subclass of NSOperation
- For lightweight tasks where creating a subclass is overkill

Using an NSInvocationOperation

- Concrete subclass of NSOperation
- For lightweight tasks where creating a subclass is overkill

```
- (void)someAction:(id)sender
{
    NSInvocationOperation *operation =
        [[NSInvocationOperation alloc] initWithTarget:self
                                                selector:@selector(doWork:)
                                              object:someObject];

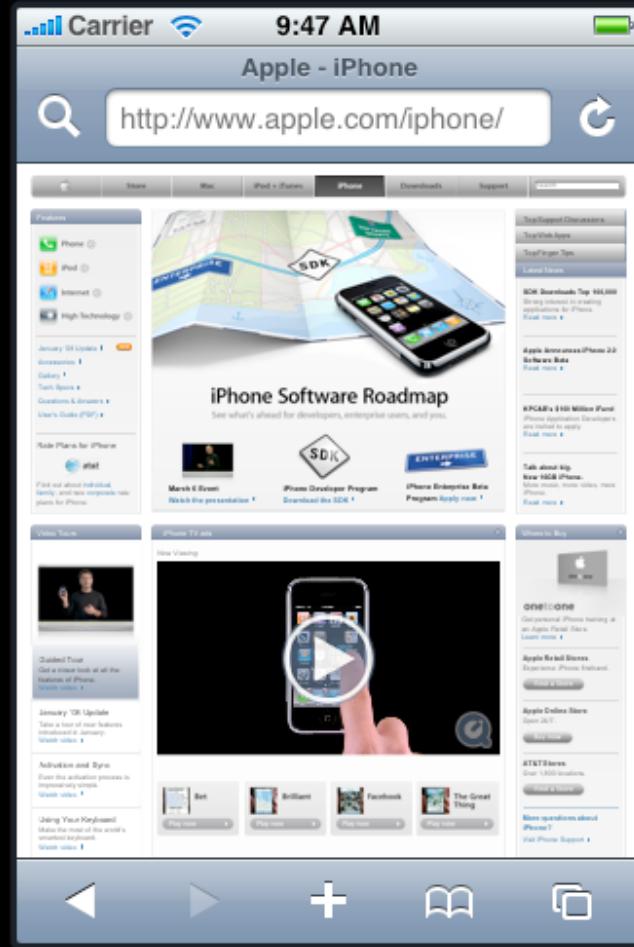
    [queue addObject:operation];

    [operation release];
}
```

iPhone Keyboards

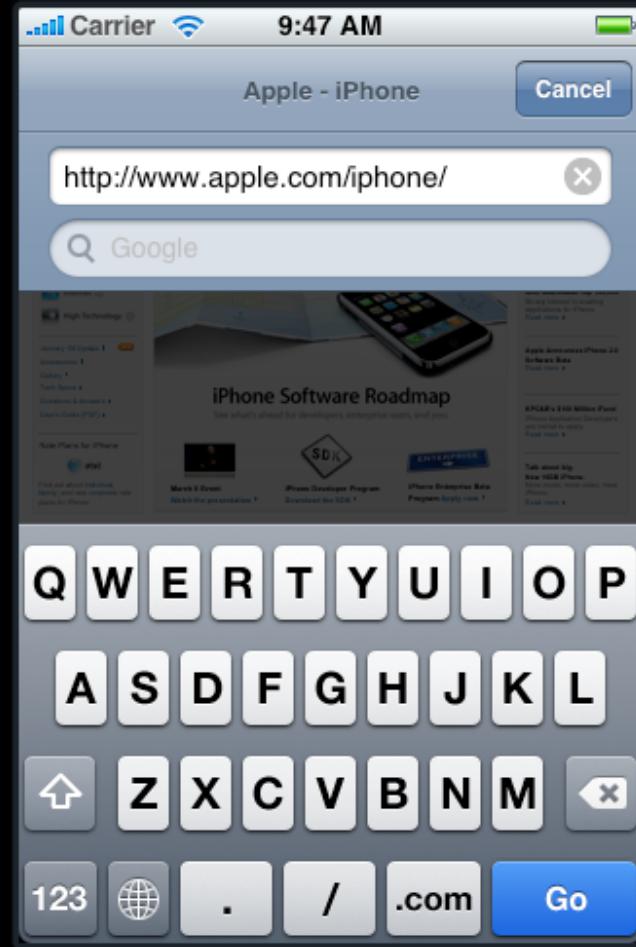
Virtual keyboard

Appears when needed

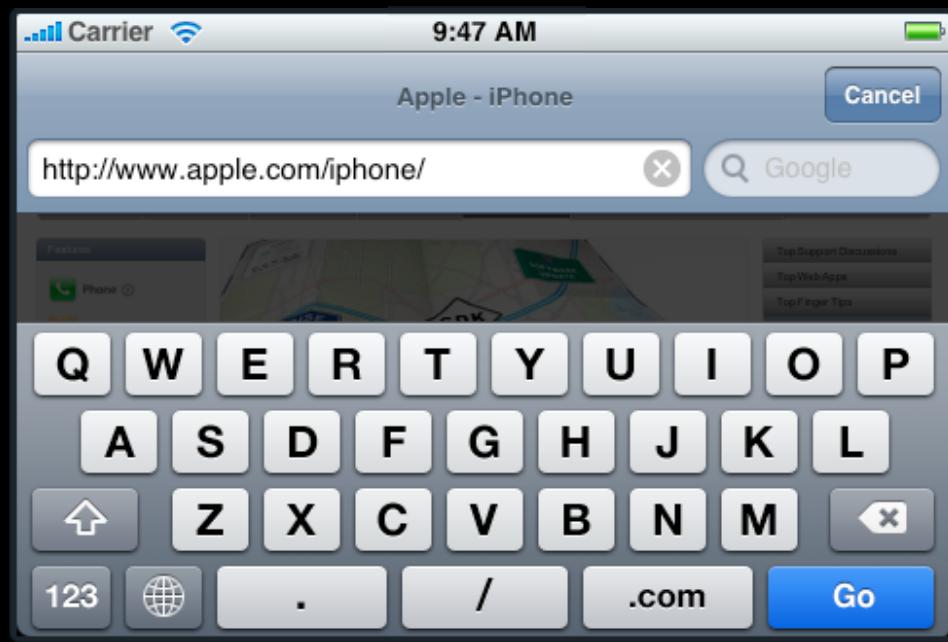


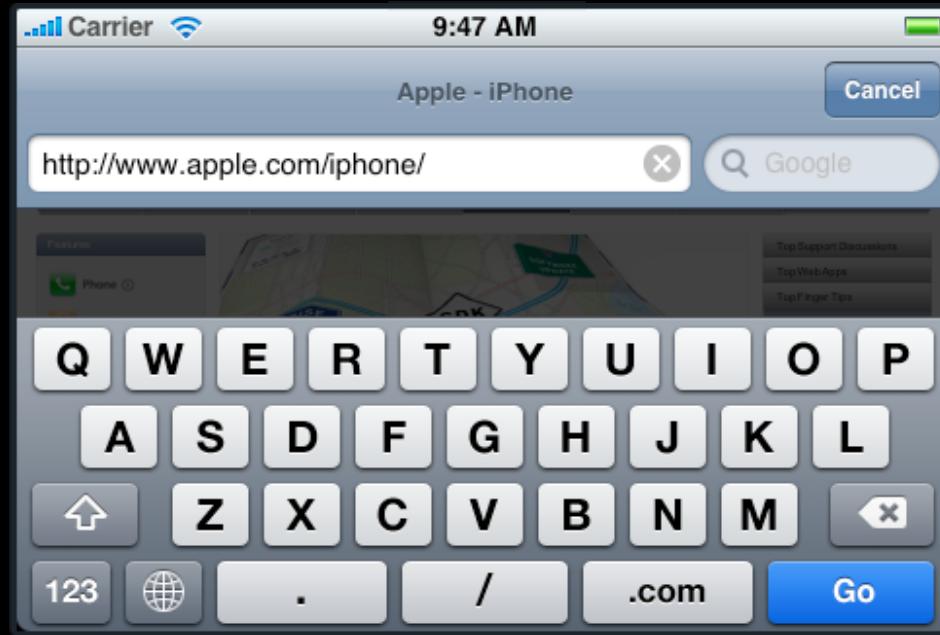
Virtual keyboard

Appears when needed









Portrait and Landscape

Simple selection model

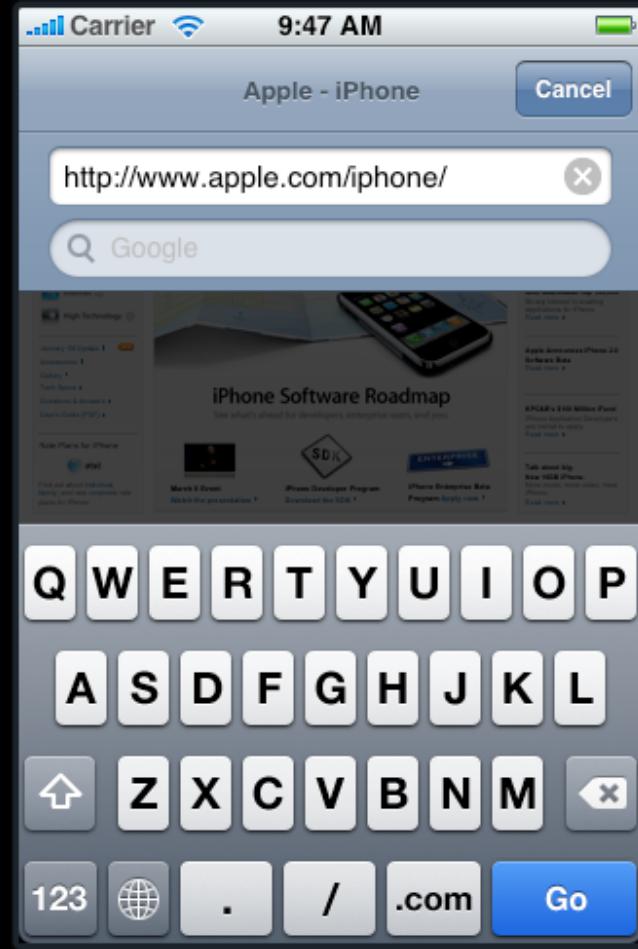
Text loupe/magnifier



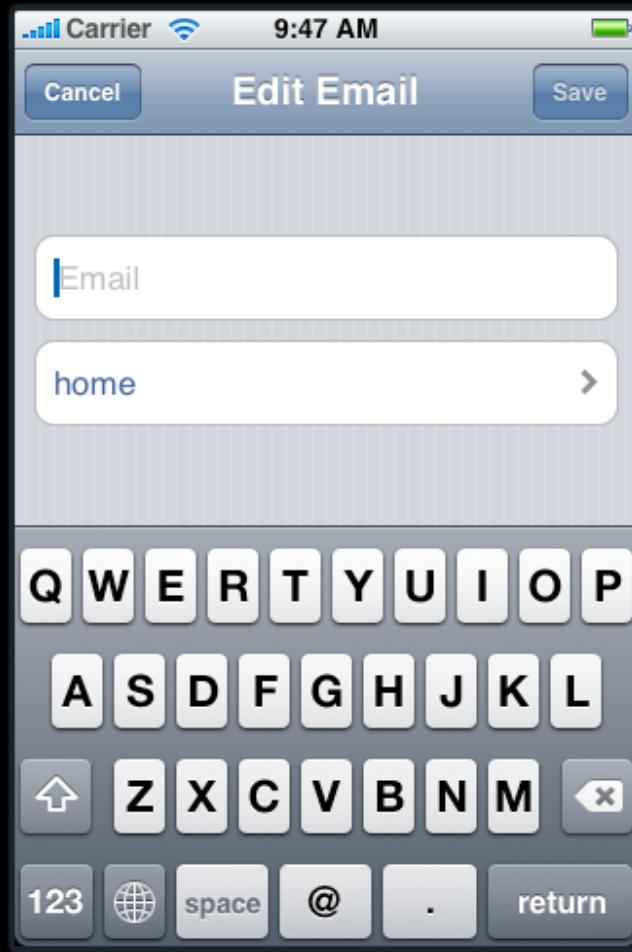
Many keyboard types
Adapted to task



Many keyboard types
Adapted to task



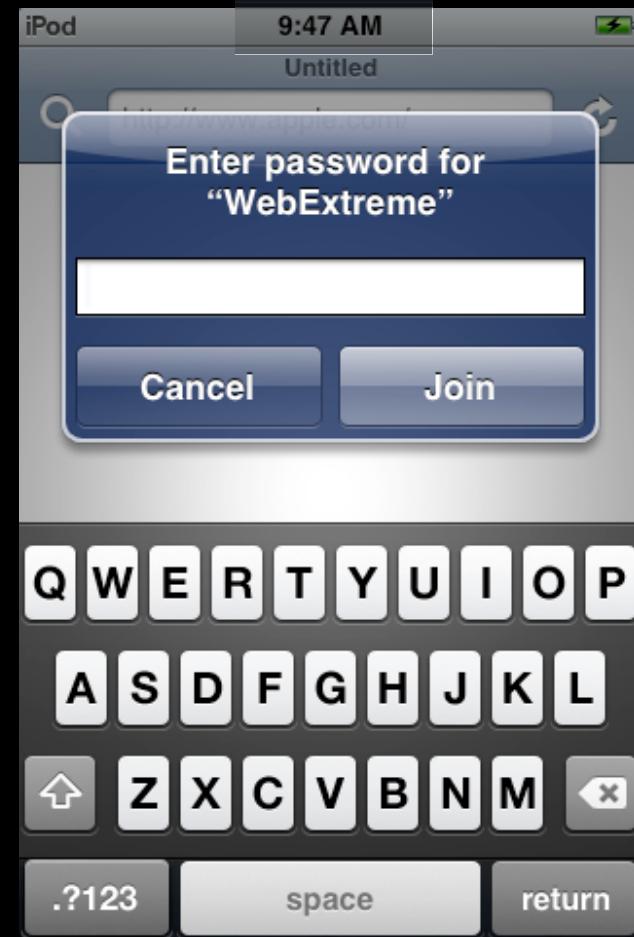
Many keyboard types
Adapted to task



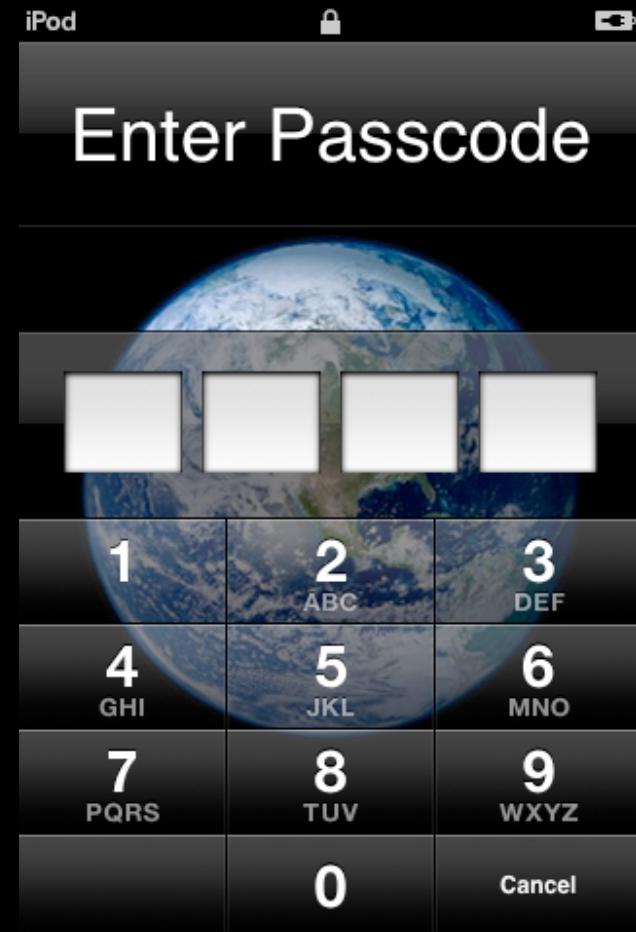
Many keyboard types
Adapted to task



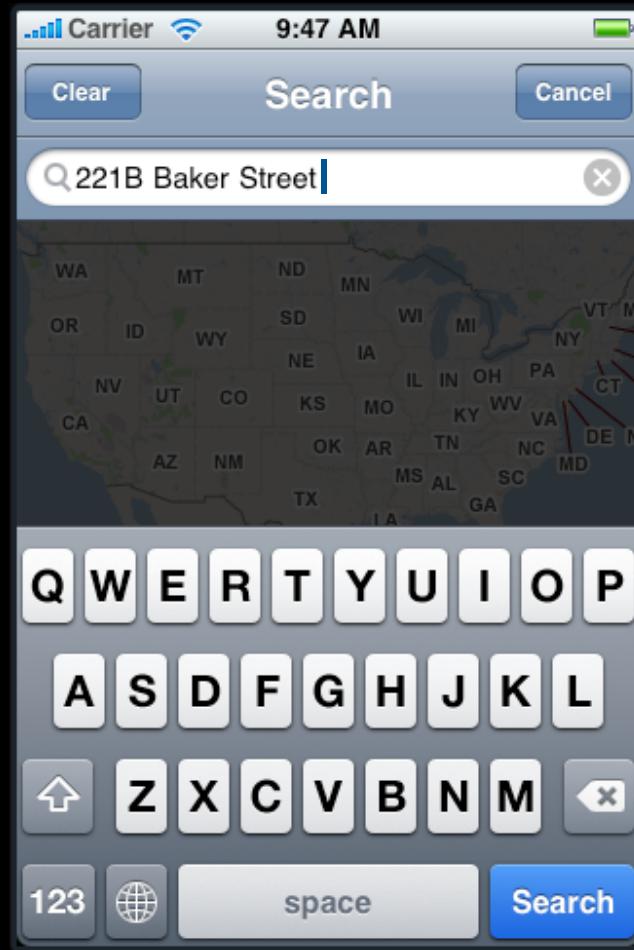
Many keyboard types
Adapted to task



Many keyboard types
Adapted to task



Single line editing

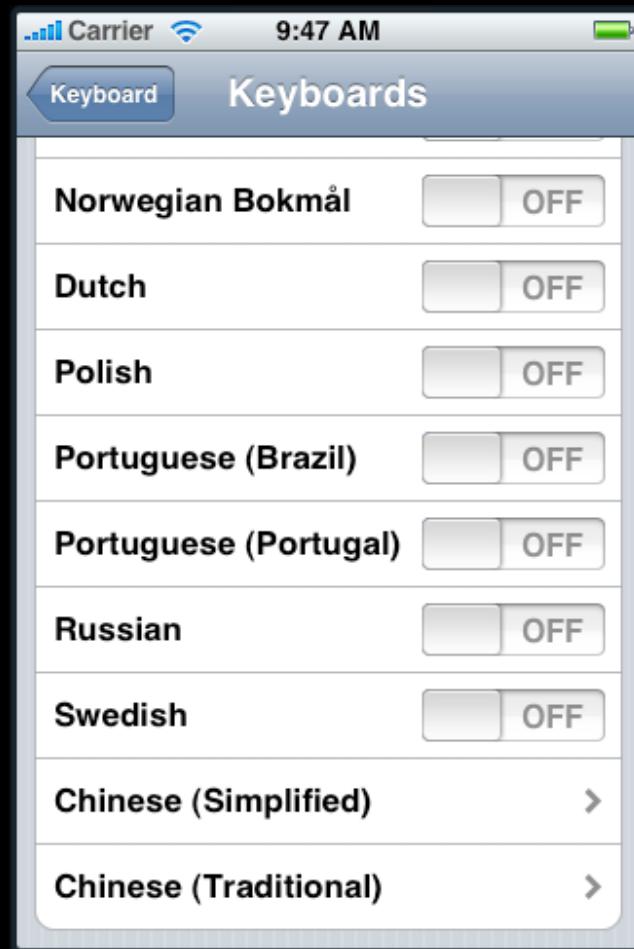


Multi-line editing



40

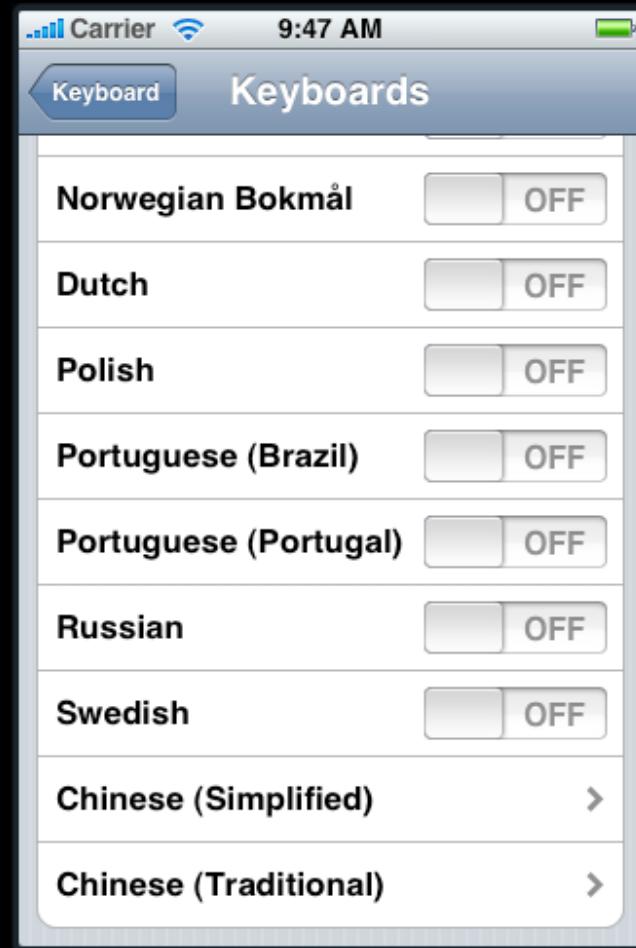
Languages



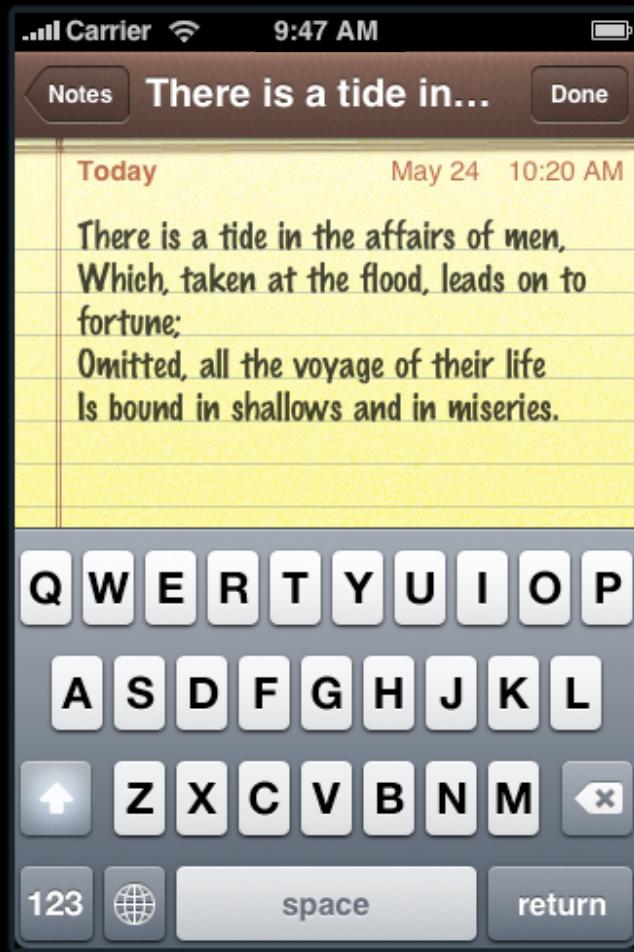
40

Languages

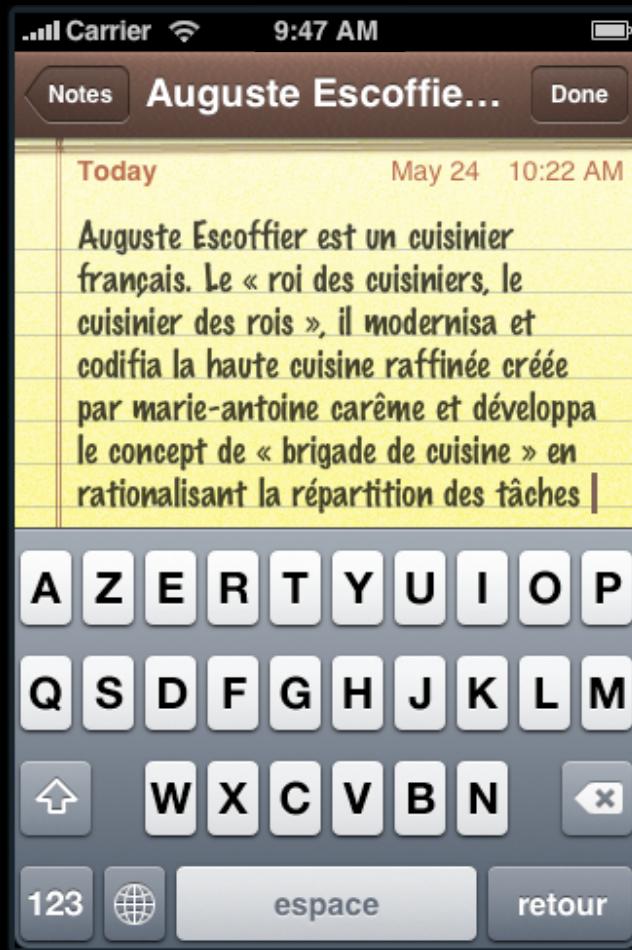
Full dictionary support



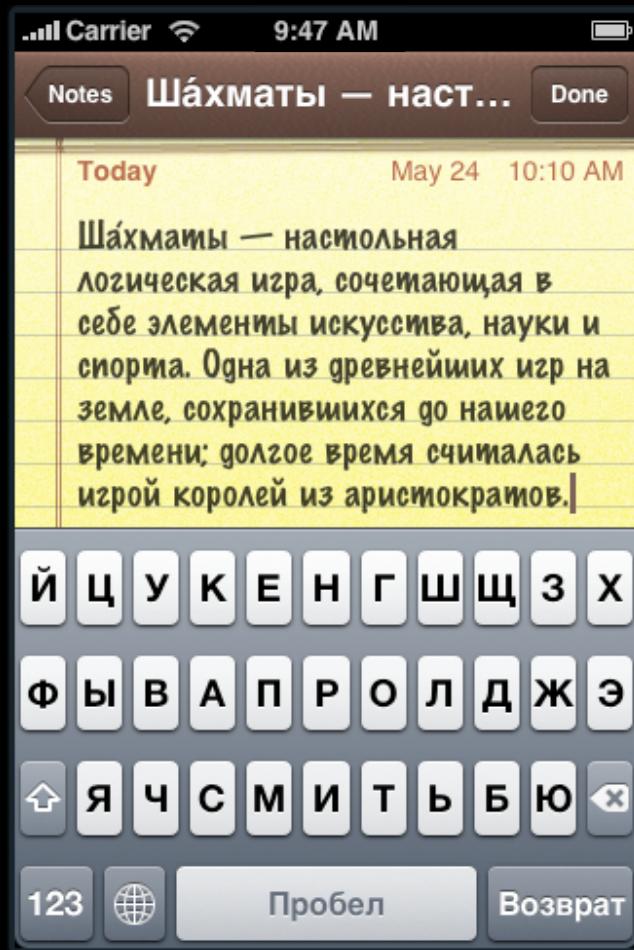
English



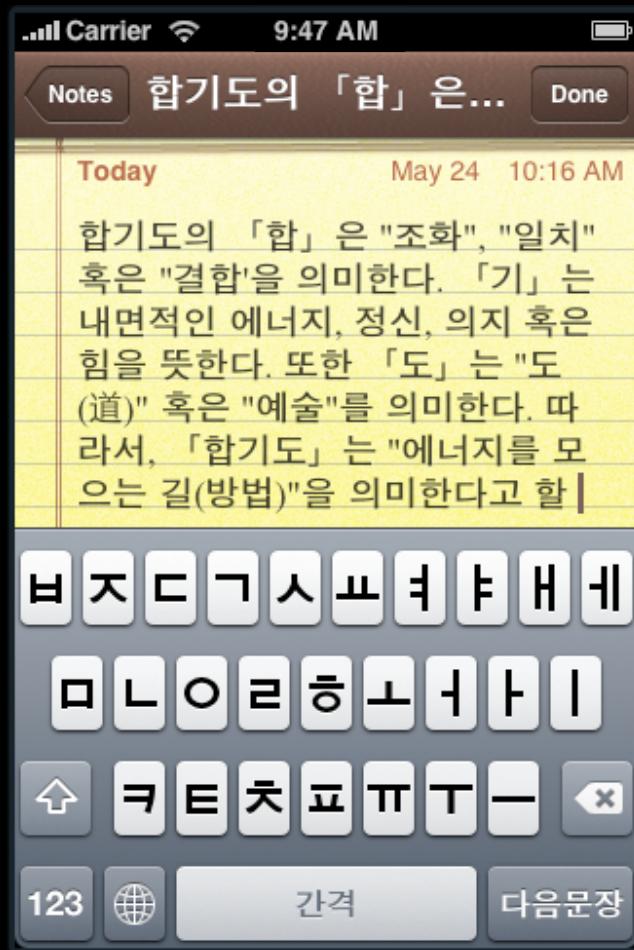
French



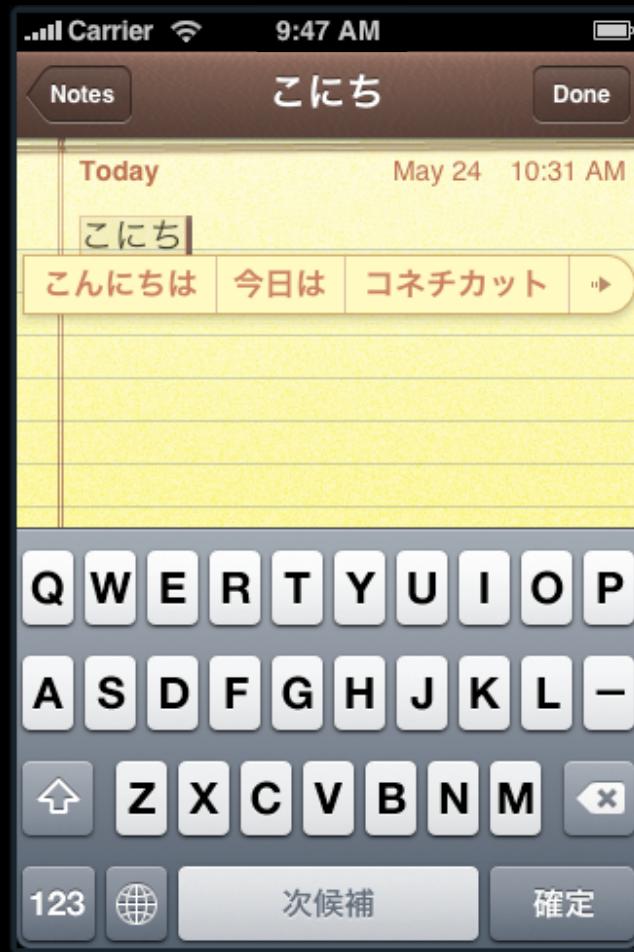
Russian



Korean



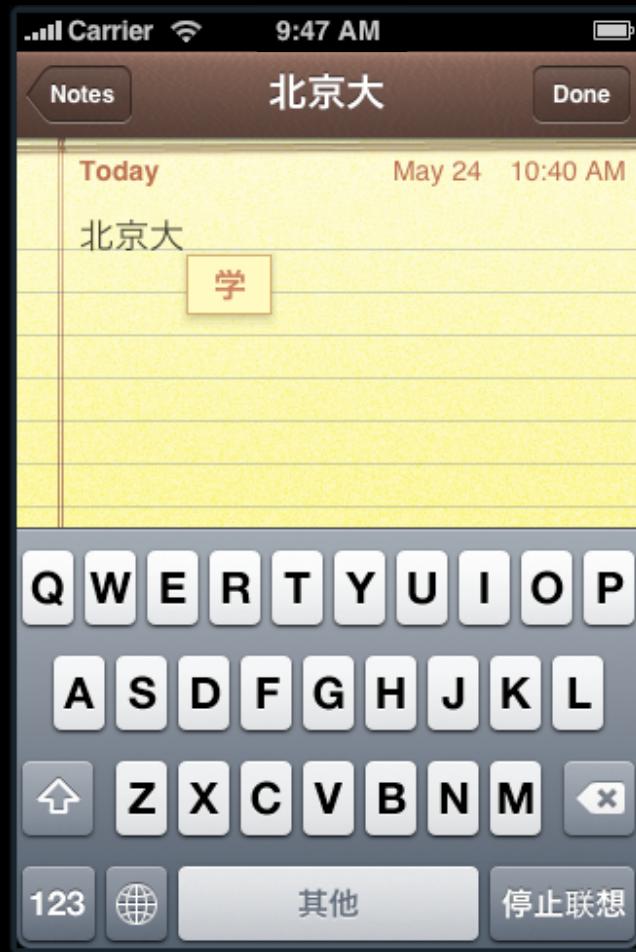
Japanese Romaji



Japanese Kana



Chinese Pinyin

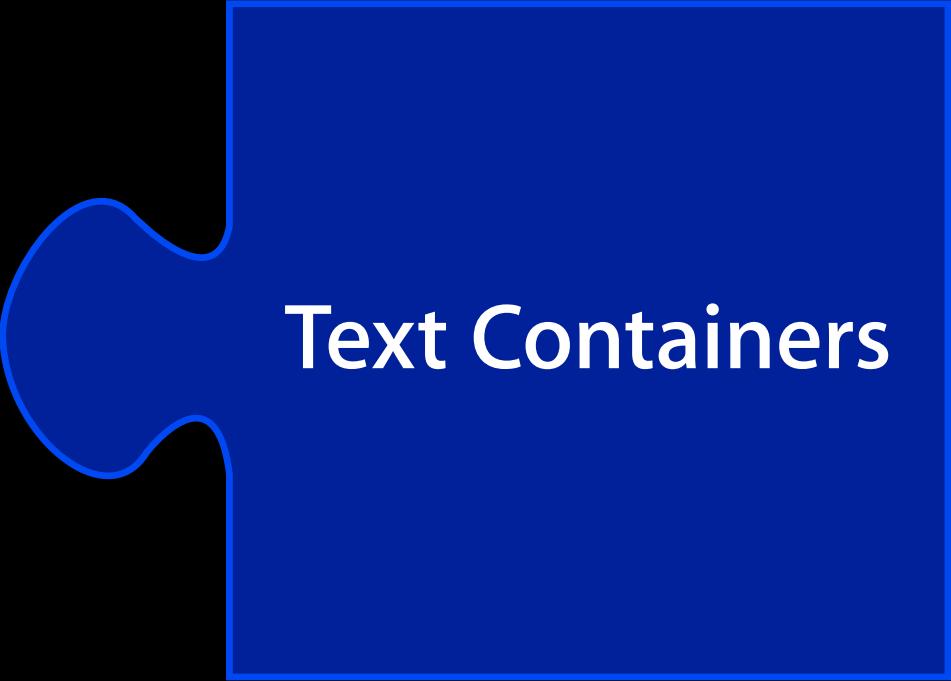


Chinese Handwriting

Simplified
Traditional



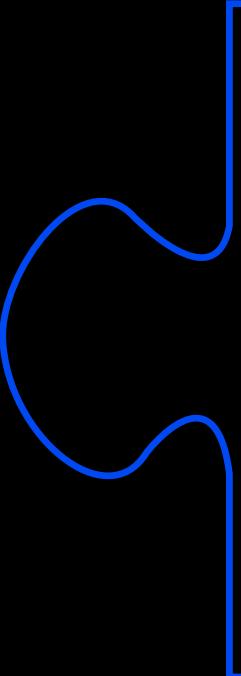
Customizing Text Input



Text Containers

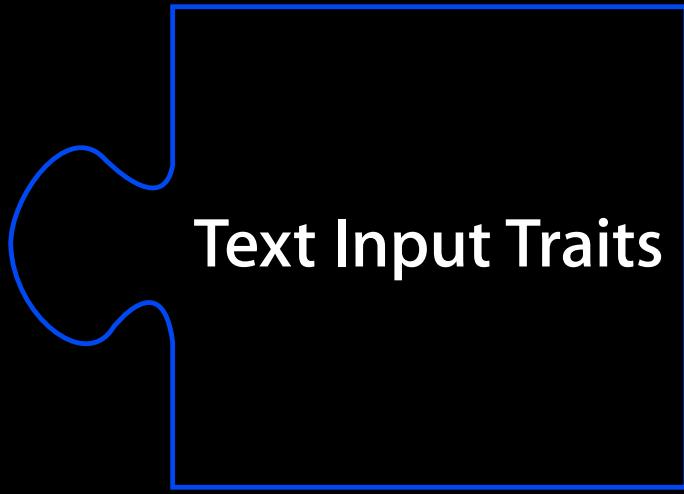
Text Containers

Delegates
Notifications
Methods



Text Containers

Text Input Traits



Protocol
UITextField
UITextView



Text Input Traits

Autocapitalization

Autocorrection

Keyboard Type

Keyboard Appearance

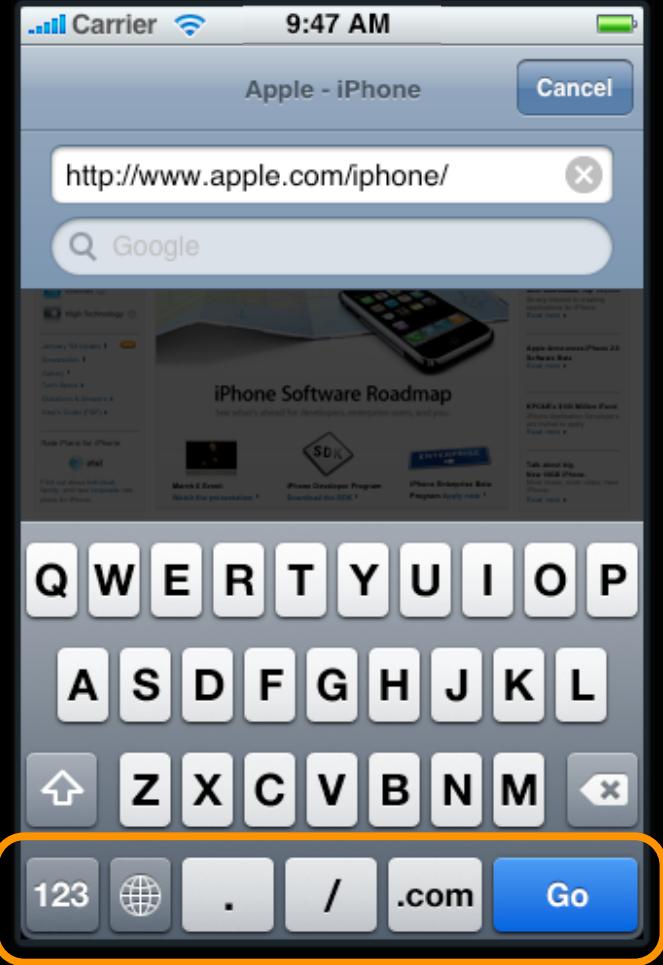
Return Key Type

Return Key Autoenabling

Secure Text Entry

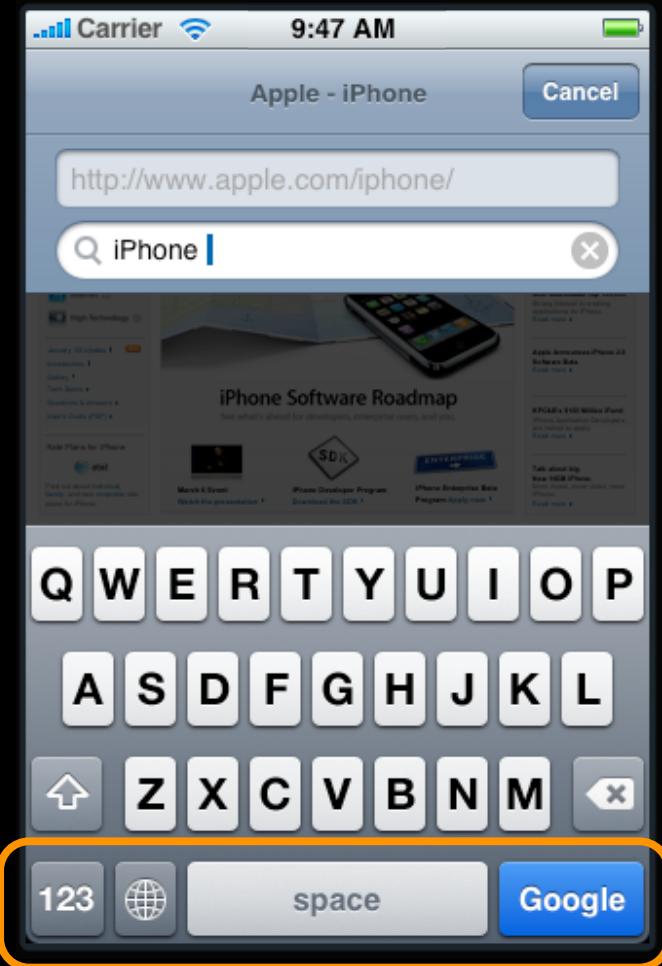
Text Input Traits

URL Keyboard
Go button



Text Input Traits

Default Keyboard
Google button



Text Containers

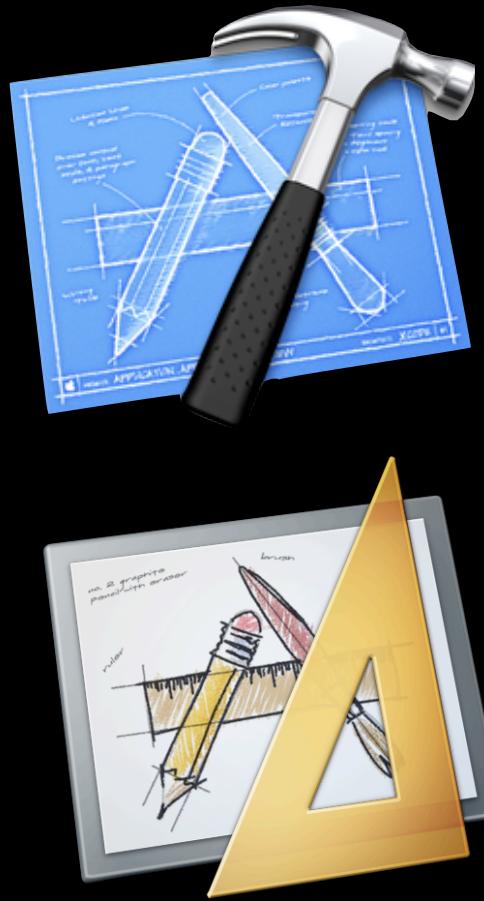
Text Input Traits

Delegates

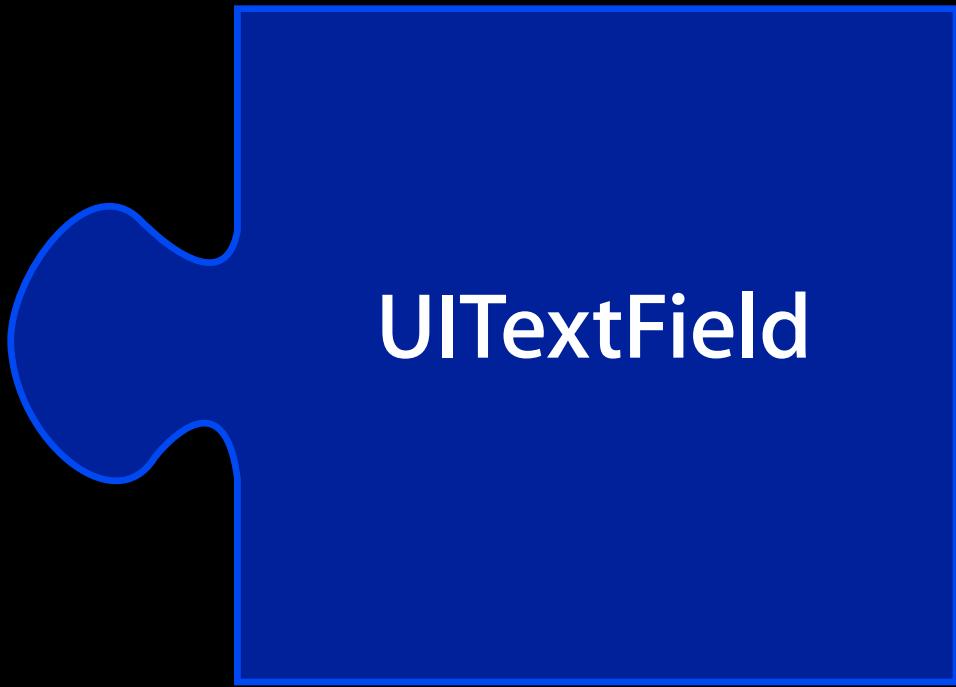
Notifications

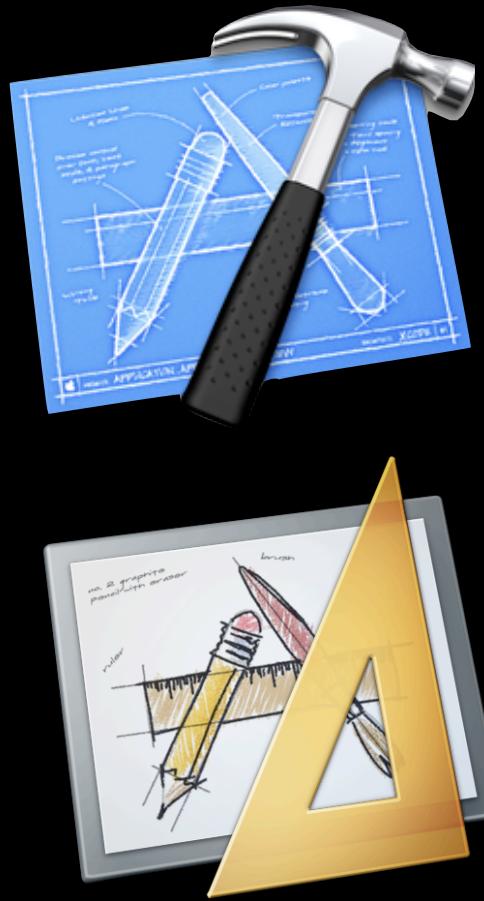
Methods

Text Containers



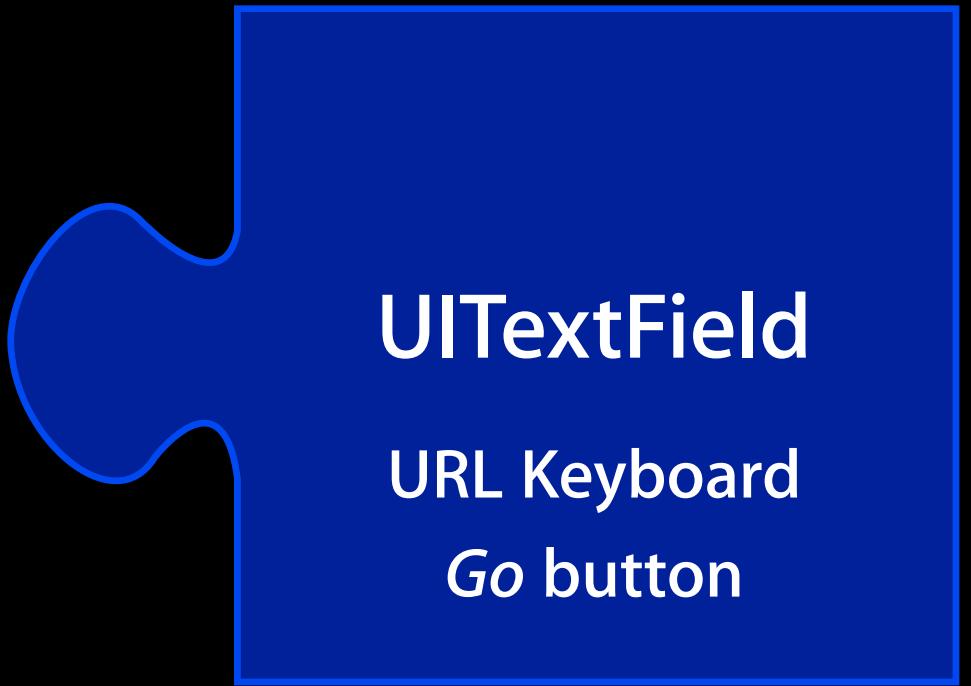
Design time





Design time

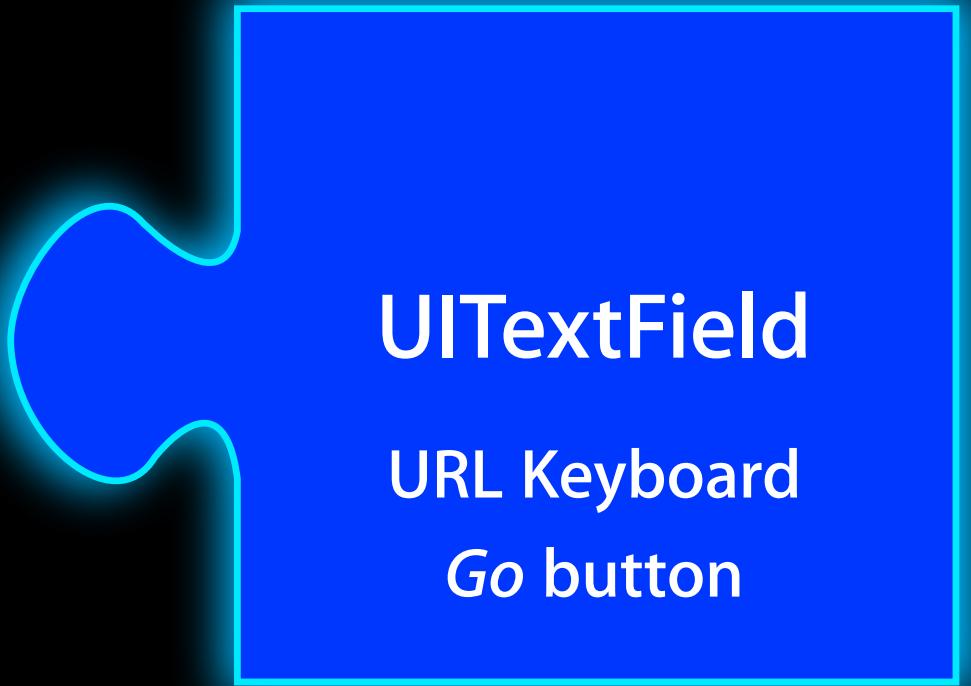
UITextField
URL Keyboard
Go button



UITextField

URL Keyboard
Go button

Run time

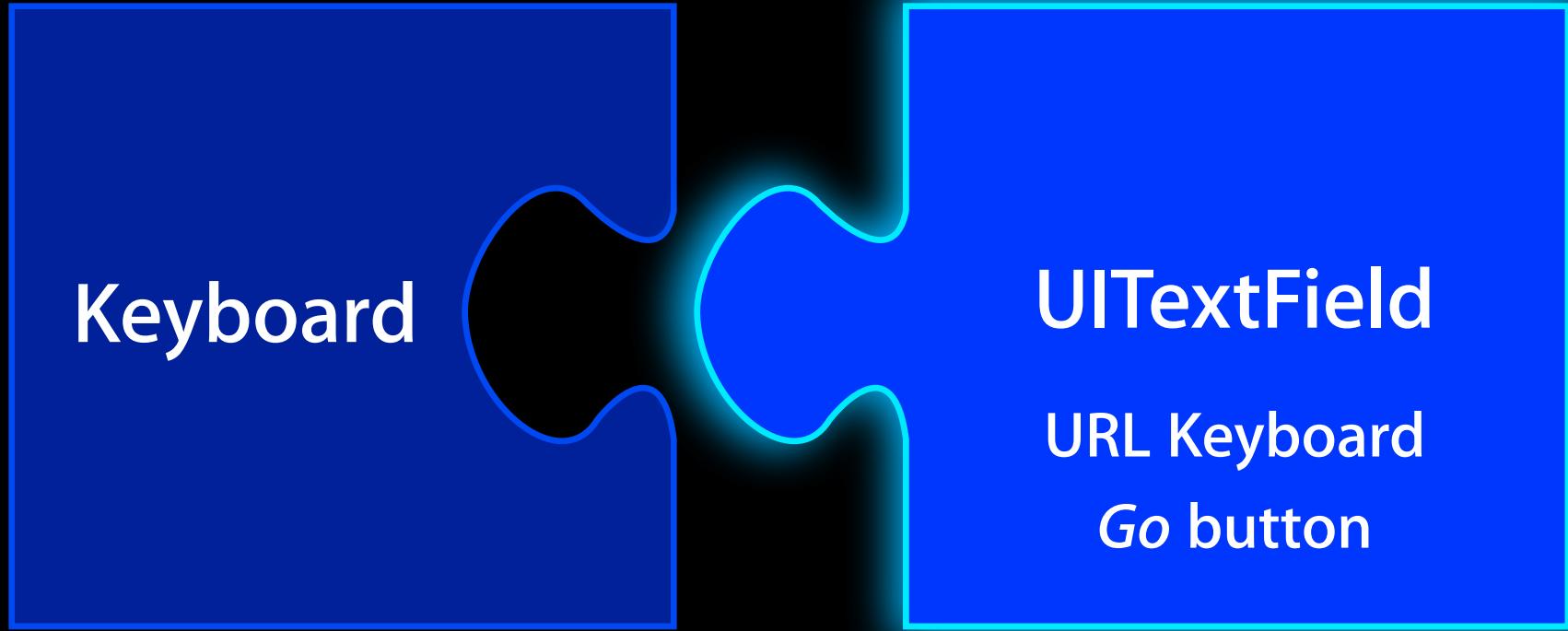


UITextField

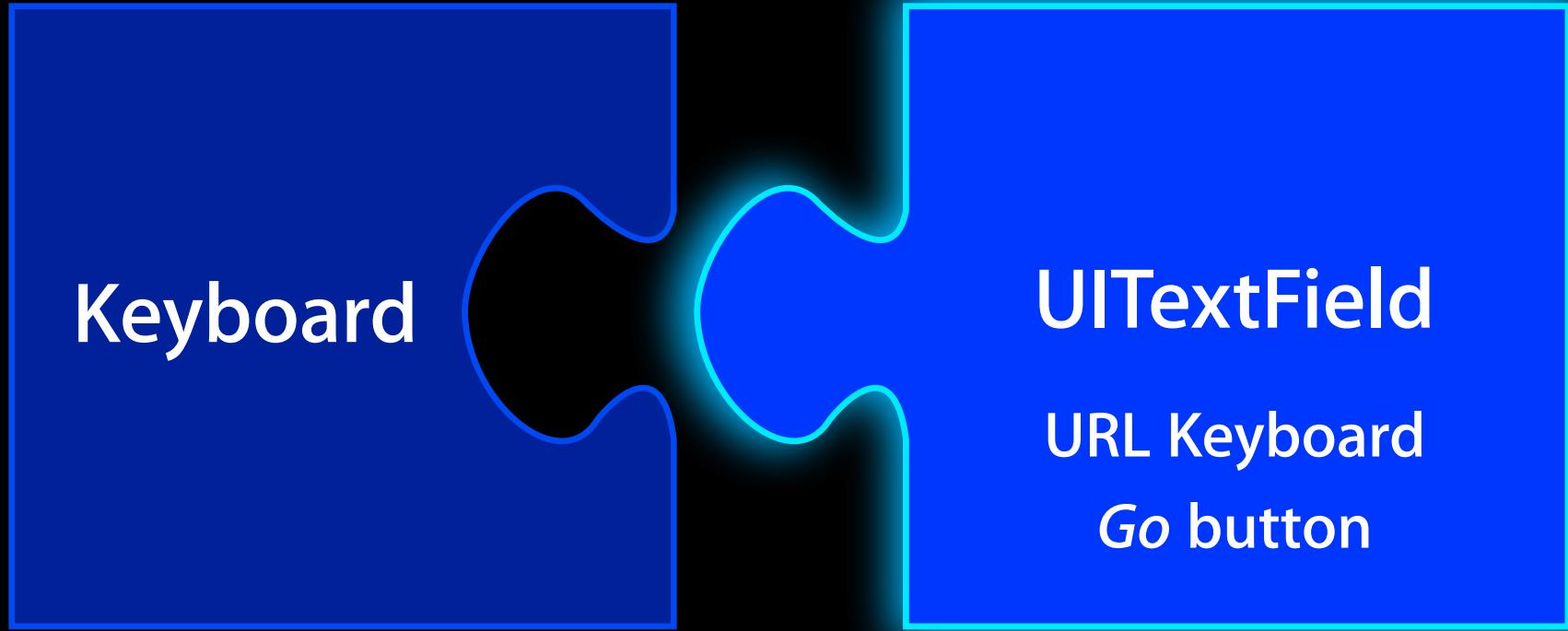
URL Keyboard

Go button

Become first responder



Become first responder



Become first responder

Keyboard

UITextField

URL Keyboard

Go button

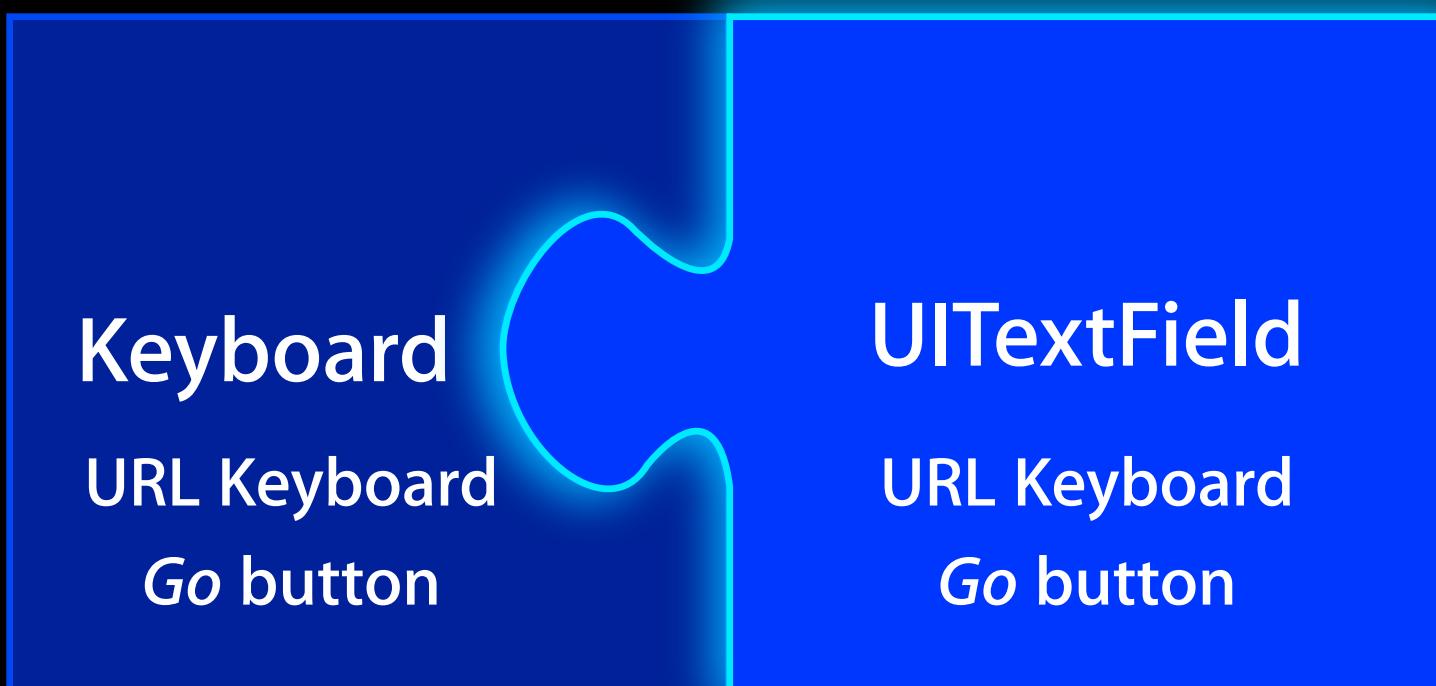
Become first responder

The diagram illustrates the concept of trait adoption. It features a large blue rounded rectangle labeled "Keyboard". To its right is a smaller blue rounded rectangle containing three items: "UITextField", "URL Keyboard", and "Go button". A curved blue line connects the "Keyboard" label to the "UITextField" label, symbolizing that the Keyboard adopts the traits of UITextField.

Keyboard

UITextField
URL Keyboard
Go button

Keyboard adopts traits



Keyboard

URL Keyboard

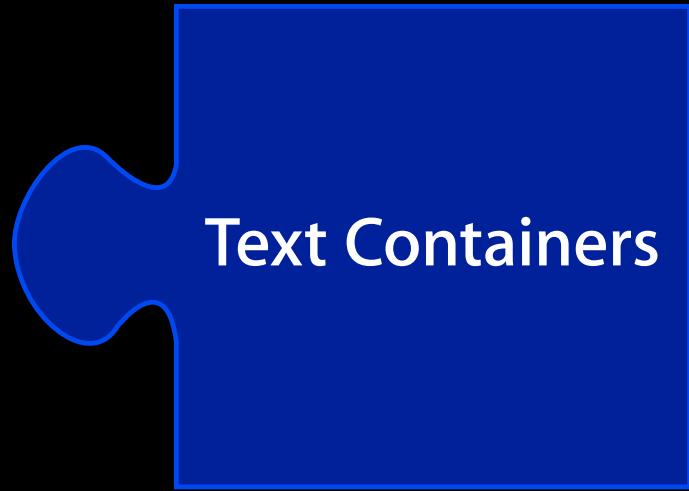
Go button

UITextField

URL Keyboard

Go button

Keyboard adopts traits



UITextField

UITextView

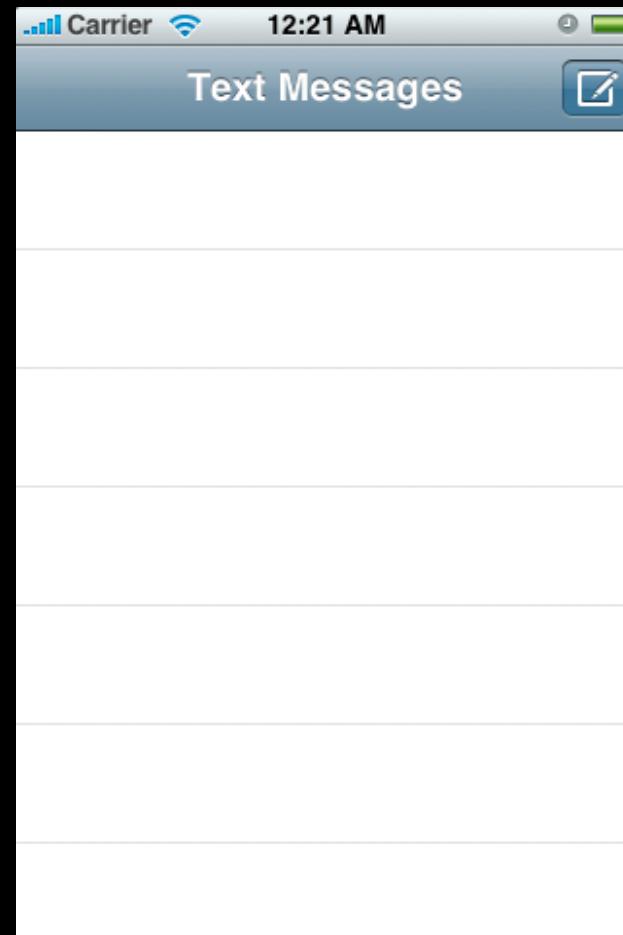
Web Forms

Demo: Text Input

Presenting Content Modally

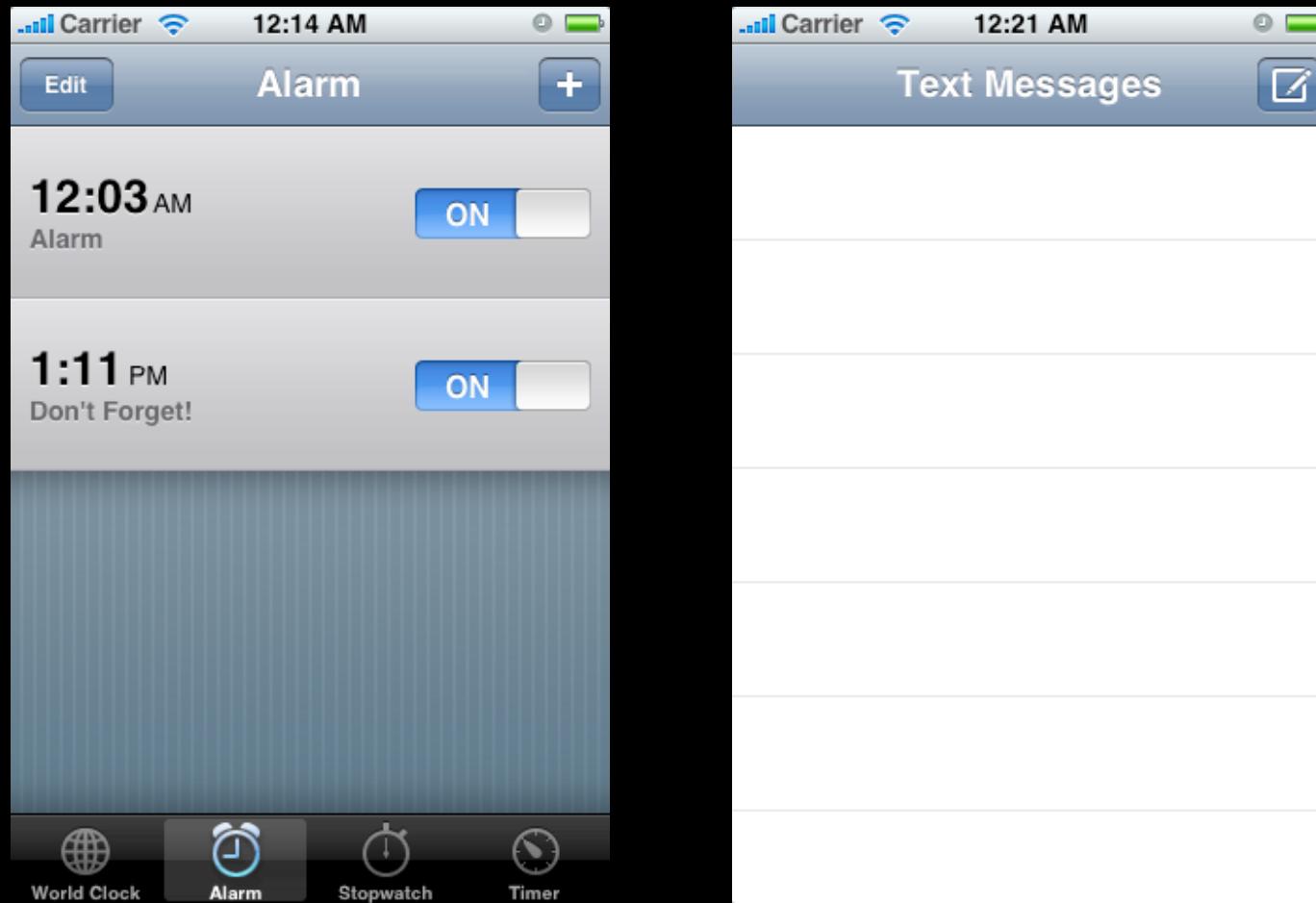
Presenting Content Modally

- For **adding** or **picking** data



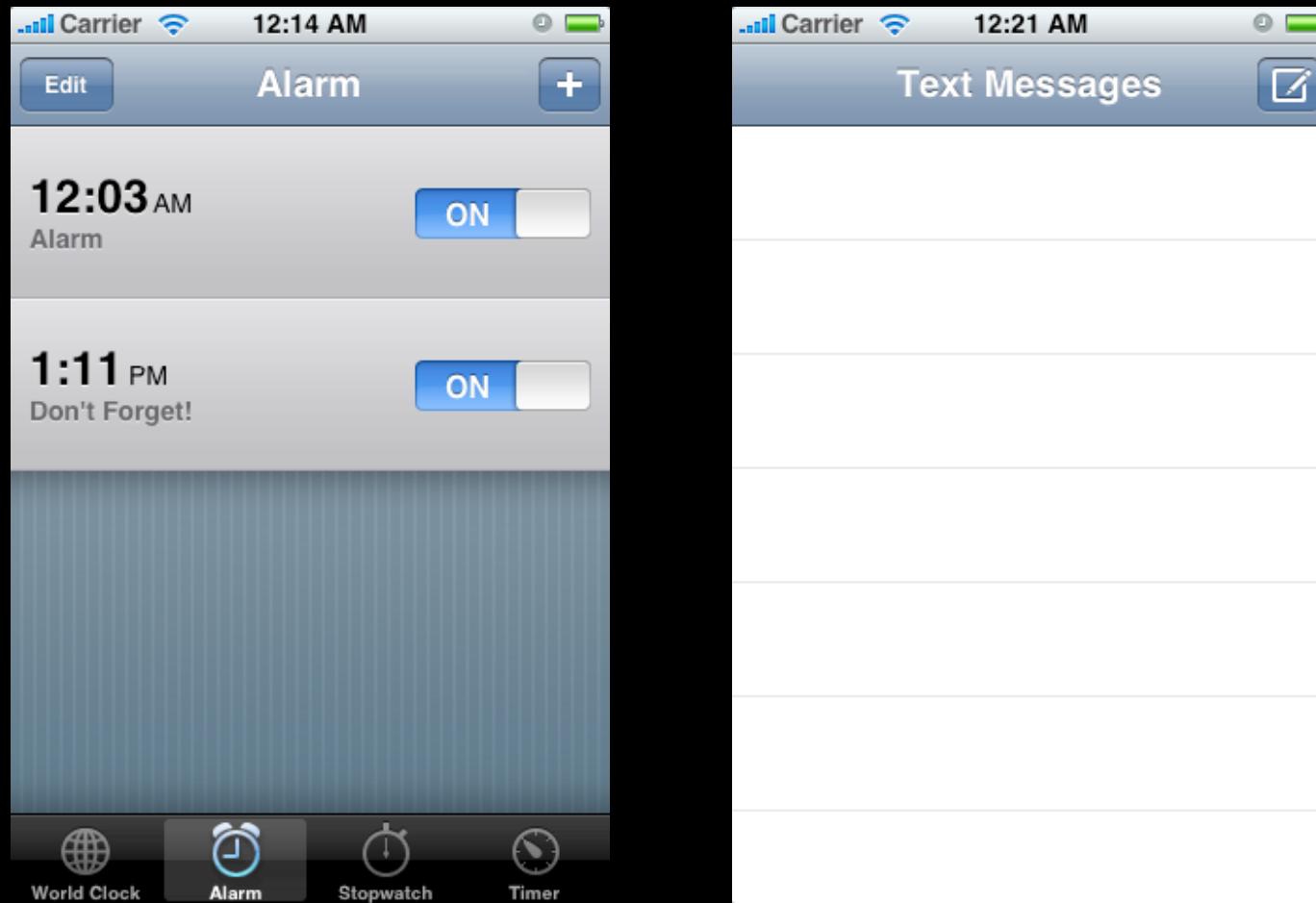
Presenting Content Modally

- For **adding** or **picking** data



Presenting Content Modally

- For **adding** or **picking** data



Presenting a View Controller



Presenting a View Controller

```
// Recipe list view controller
- (void)showAddRecipe {
    RecipeAddViewController *viewController = ...;
    [self presentModalViewController:viewController animated:YES];
}
```



Presenting a View Controller

```
// Recipe list view controller
- (void)showAddRecipe {
    RecipeAddViewController *viewController = ...;
    [self presentModalViewController:viewController animated:YES];
}
```



Dismissing a View Controller



Dismissing a View Controller

```
// Recipe list view controller  
- (void)didAddRecipe {  
    [self dismissModalViewControllerAnimated:YES];  
}
```

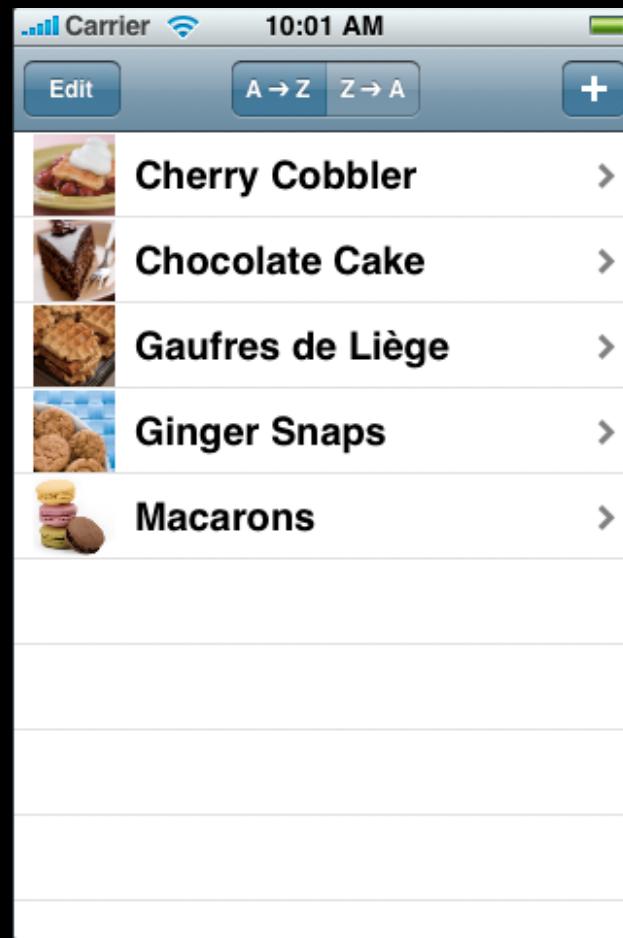


Dismissing a View Controller

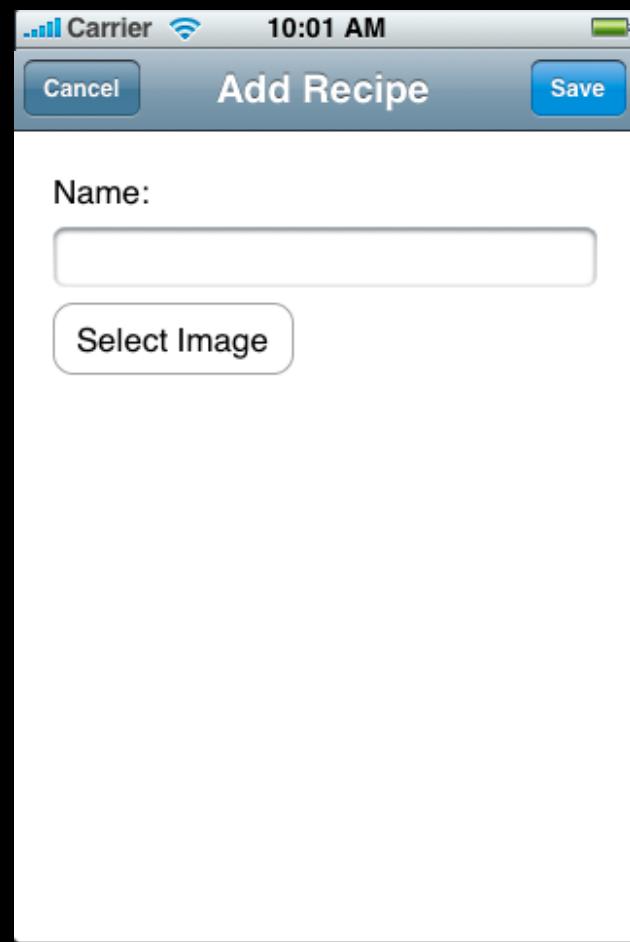
```
// Recipe list view controller  
- (void)didAddRecipe {  
    [self dismissModalViewControllerAnimated:YES];  
}
```



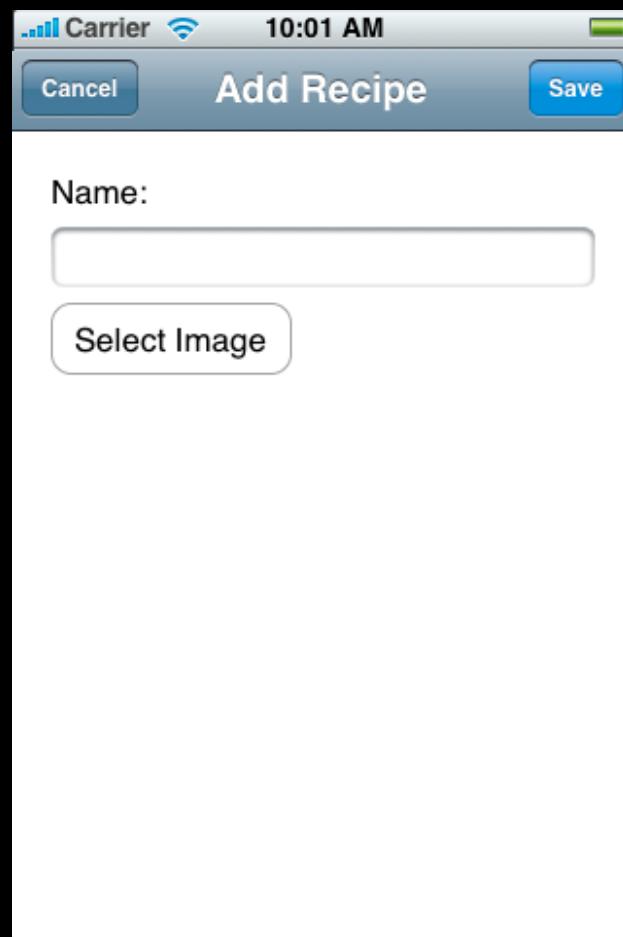
Separate Navigation Stacks



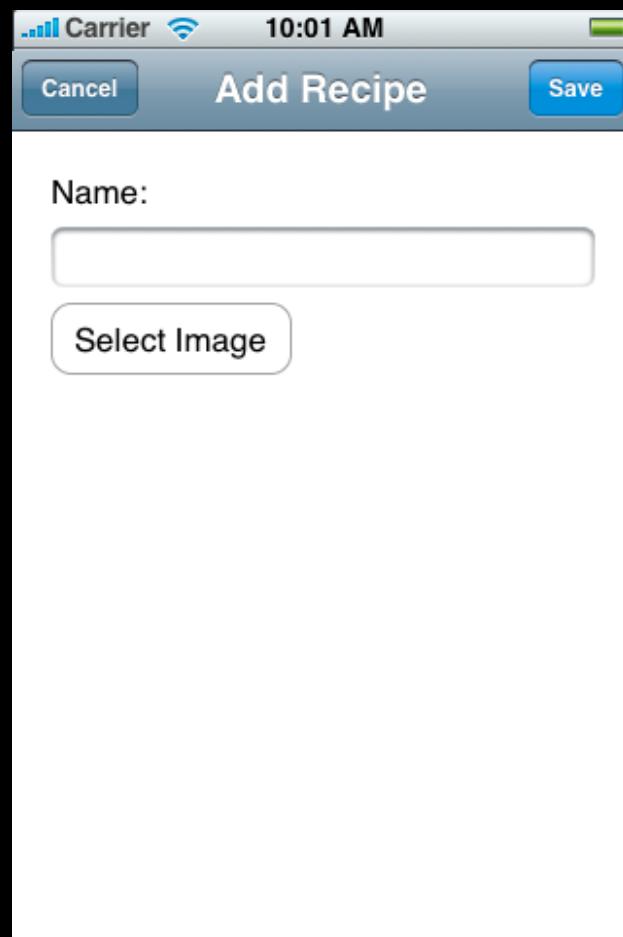
Separate Navigation Stacks



Separate Navigation Stacks



Separate Navigation Stacks



Separate Navigation Stacks



Dismissing a Modal View Controller

Dismissing a Modal View Controller

- Who should do it?

Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss

Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller

Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller
 - Tell the delegate when the presented controller is done

Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller
 - Tell the delegate when the presented controller is done
 - The delegate makes the call to dismiss

Dismissing a Modal View Controller

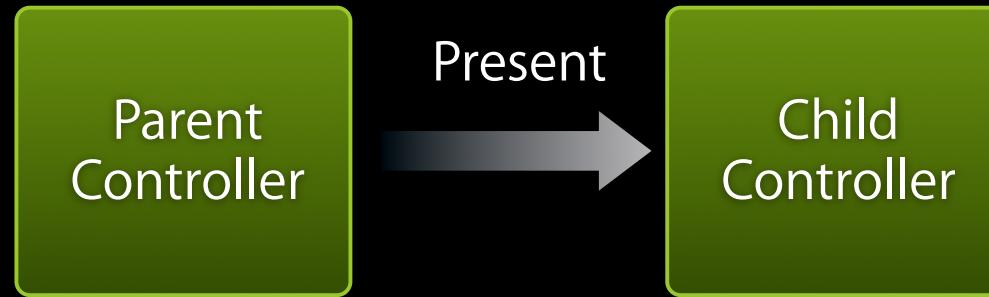
- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller
 - Tell the delegate when the presented controller is done
 - The delegate makes the call to dismiss



Parent
Controller

Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller
 - Tell the delegate when the presented controller is done
 - The delegate makes the call to dismiss



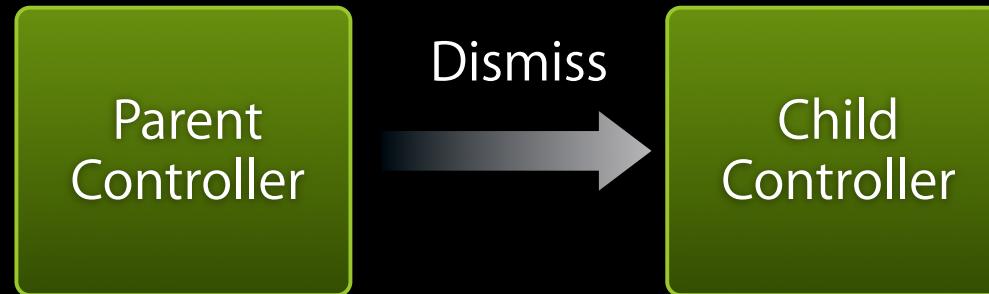
Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller
 - Tell the delegate when the presented controller is done
 - The delegate makes the call to dismiss



Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller
 - Tell the delegate when the presented controller is done
 - The delegate makes the call to dismiss



Dismissing a Modal View Controller

- Who should do it?
- Best practice is for the **same object** to call present and dismiss
- **Define delegate methods** for the presented controller
 - Tell the delegate when the presented controller is done
 - The delegate makes the call to dismiss



Parent
Controller

Demo: Presenting Content Modally

Questions?