Introducing the Contacts Framework

For OS X, iOS, and watchOS

Session 223

Bruce Stadnyk iOS Contacts Engineer
Dave Dribin OS X Contacts Engineer
Julien Robert iOS Contacts Engineer

What is Contacts Framework?

What is Contacts Framework?

Objective-C and Swift API

What is Contacts Framework?

Objective-C and Swift API

Optimized for thread-safe, read only usage

What is Contacts Framework?

Objective-C and Swift API

Optimized for thread-safe, read only usage

One API, multiple platforms

What is Contacts Framework?

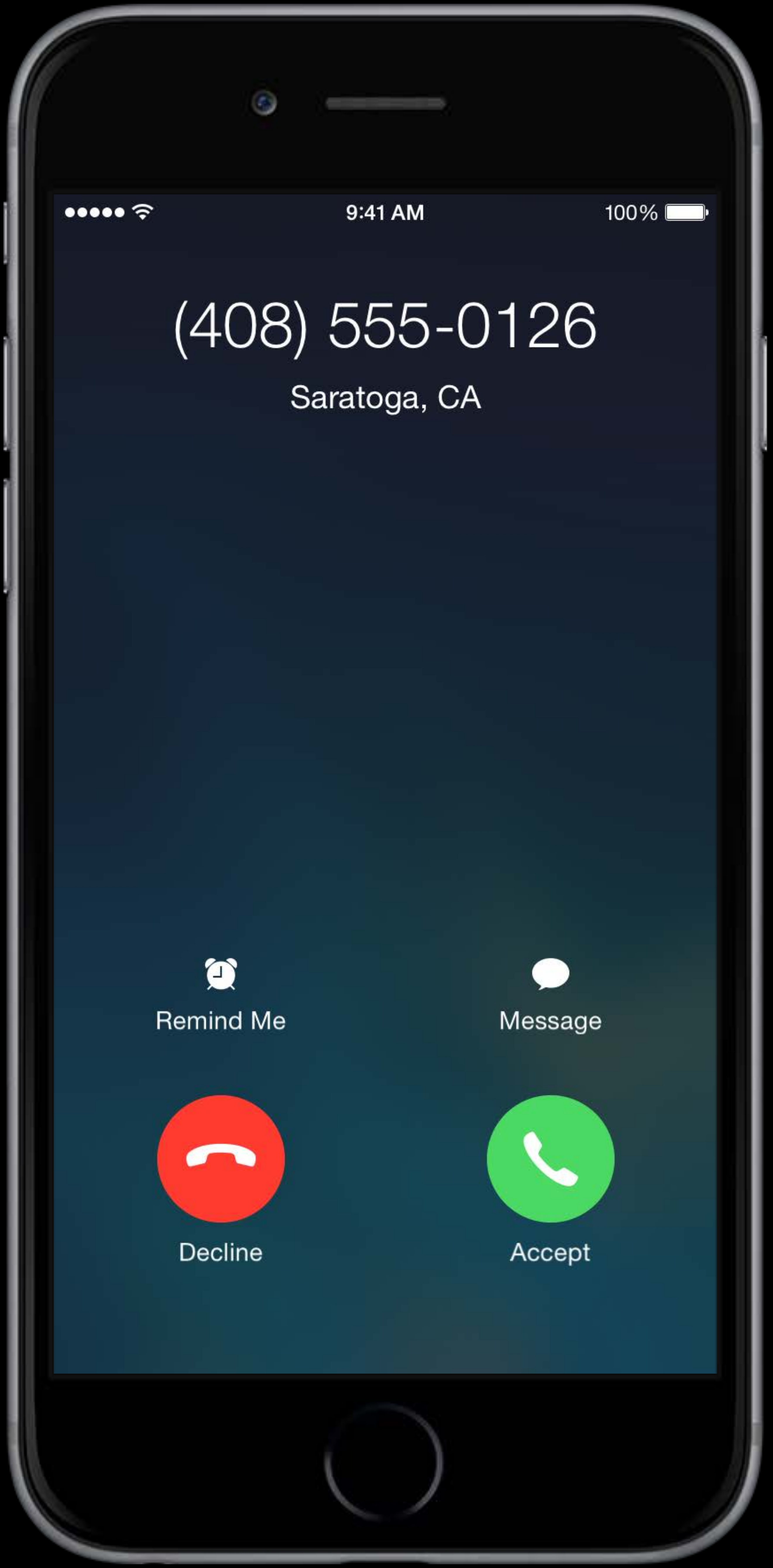
Objective-C and Swift API

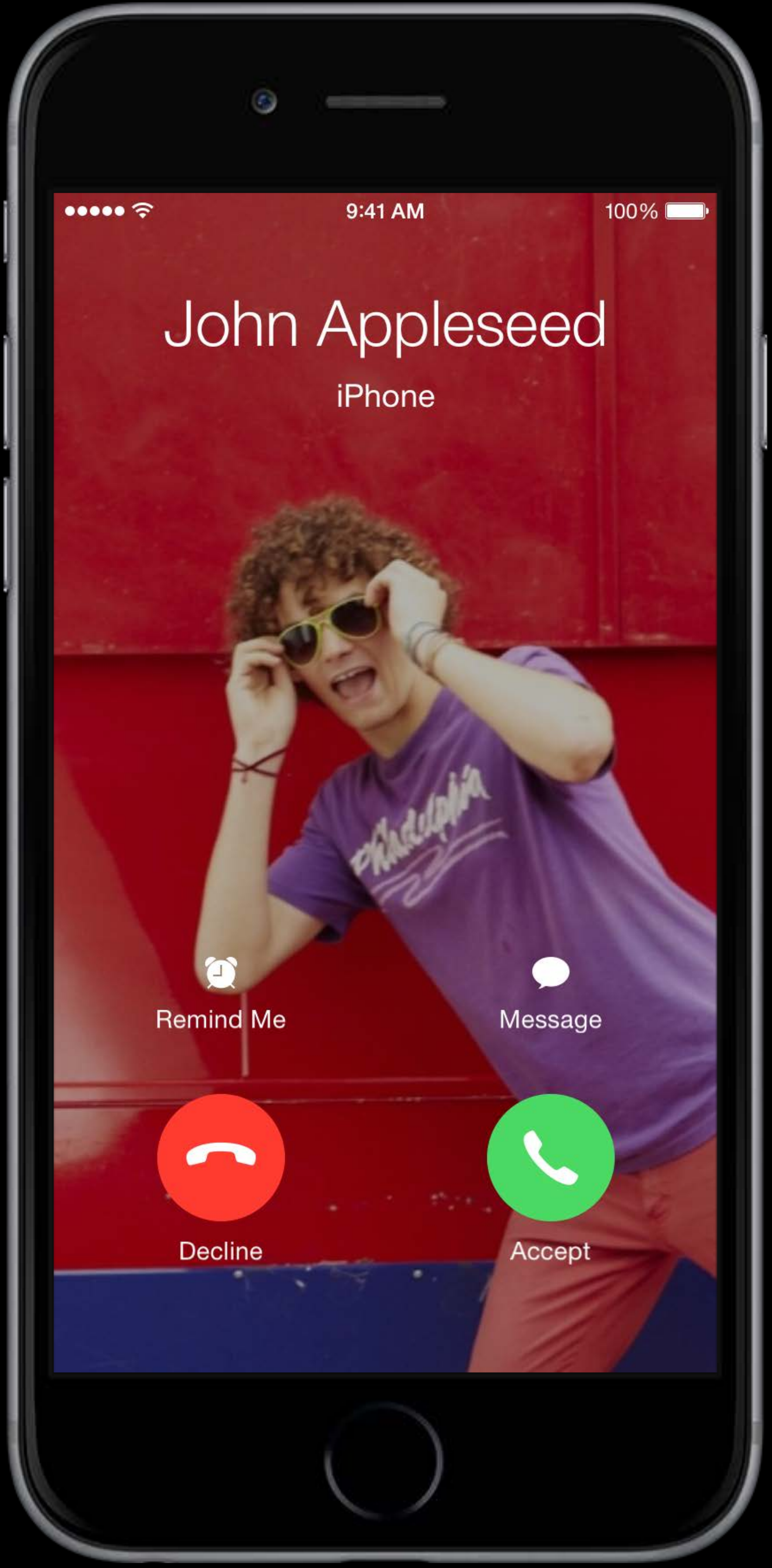
Optimized for thread-safe, read only usage

One API, multiple platforms

AddressBook API being deprecated

9:41 AM
100%
Monday
8
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
Videos
Wallet
Notes
Reminders
Stocks
iTunes Store
App Store
iBooks
News
Health
Settings
Phone
Safari
Mail
Music


9:41 AM
100%
(408) 555-0126
Saratoga, CA
Remind Me
Message
Decline
Accept


9:41 AM
100%
John Appleseed
iPhone
Remind Me
Message
Decline
Accept

What are Contacts?

What are Contacts?

Everyone has Contacts

What are Contacts?

Everyone has Contacts

Phone, Mail, Messages, . . .

What are Contacts?

Everyone has Contacts

Phone, Mail, Messages, …

What are Contacts?

Everyone has Contacts

Phone, Mail, Messages, . . .

Central to the user experience

Contact Properties

Contact Properties

John

Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

Contact Properties

John

Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

Contact Properties

John

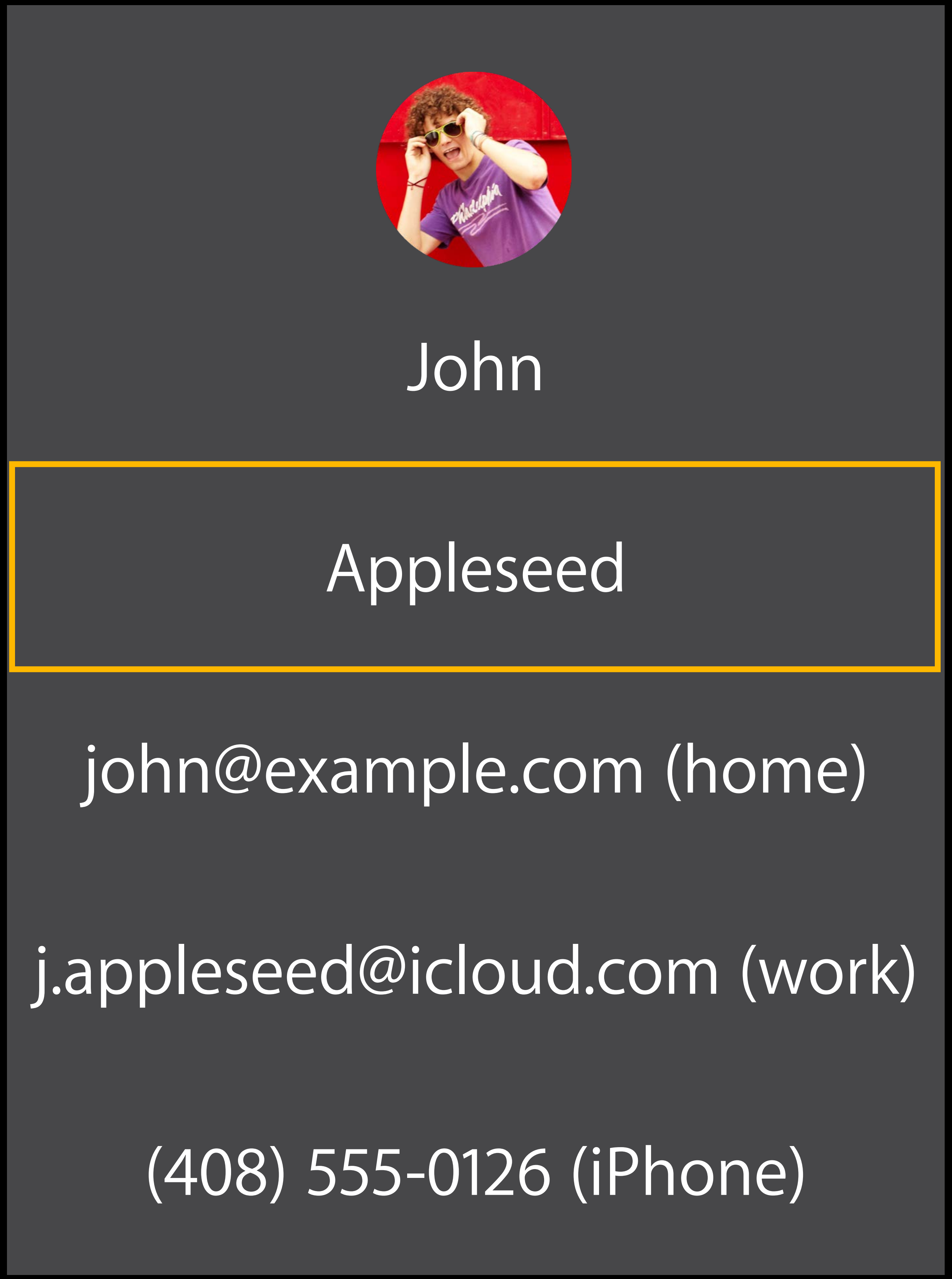
Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

Contact Properties



John

Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

Contact Properties

John

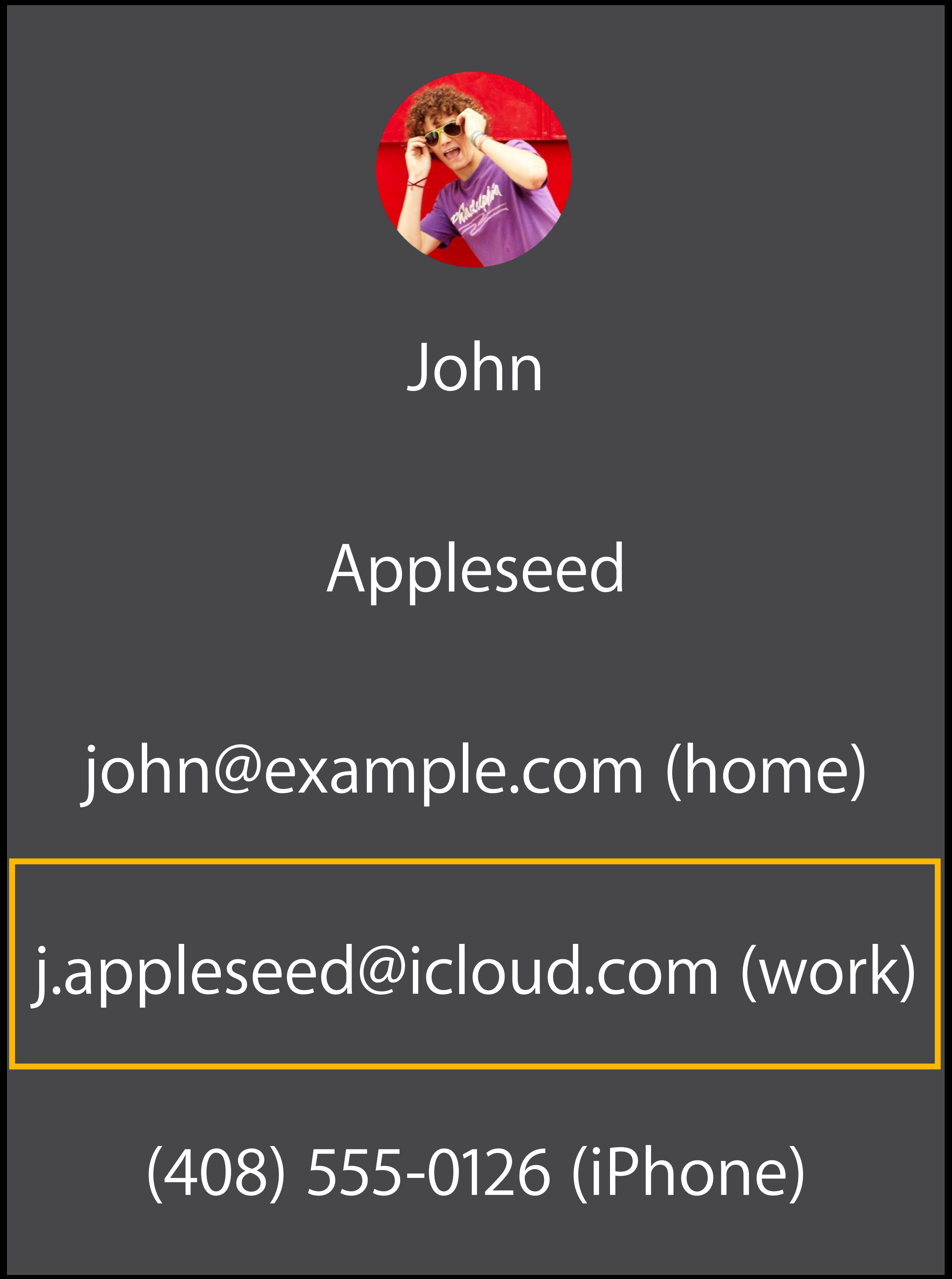
Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

Contact Properties



John

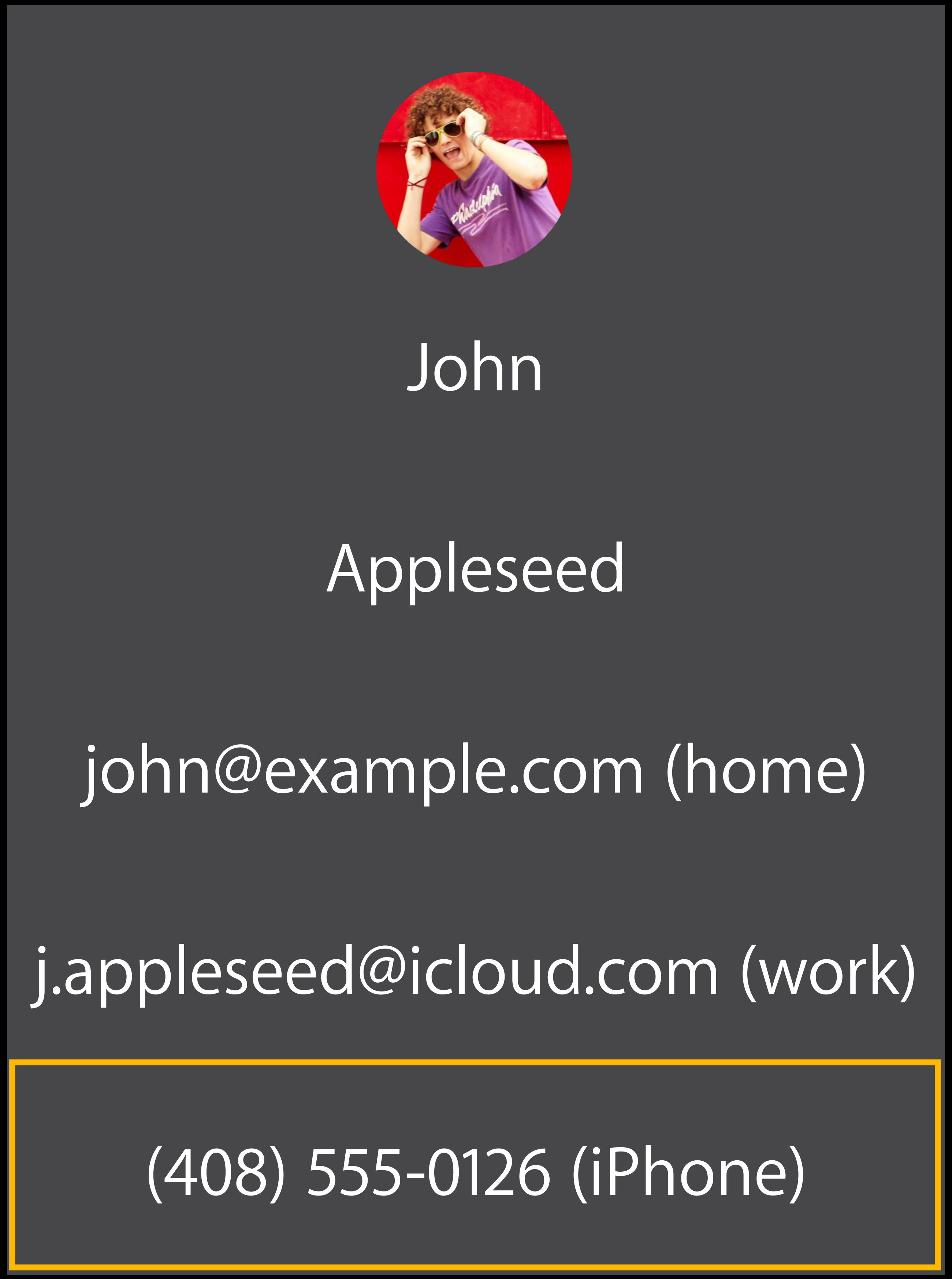
Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

Contact Properties



John

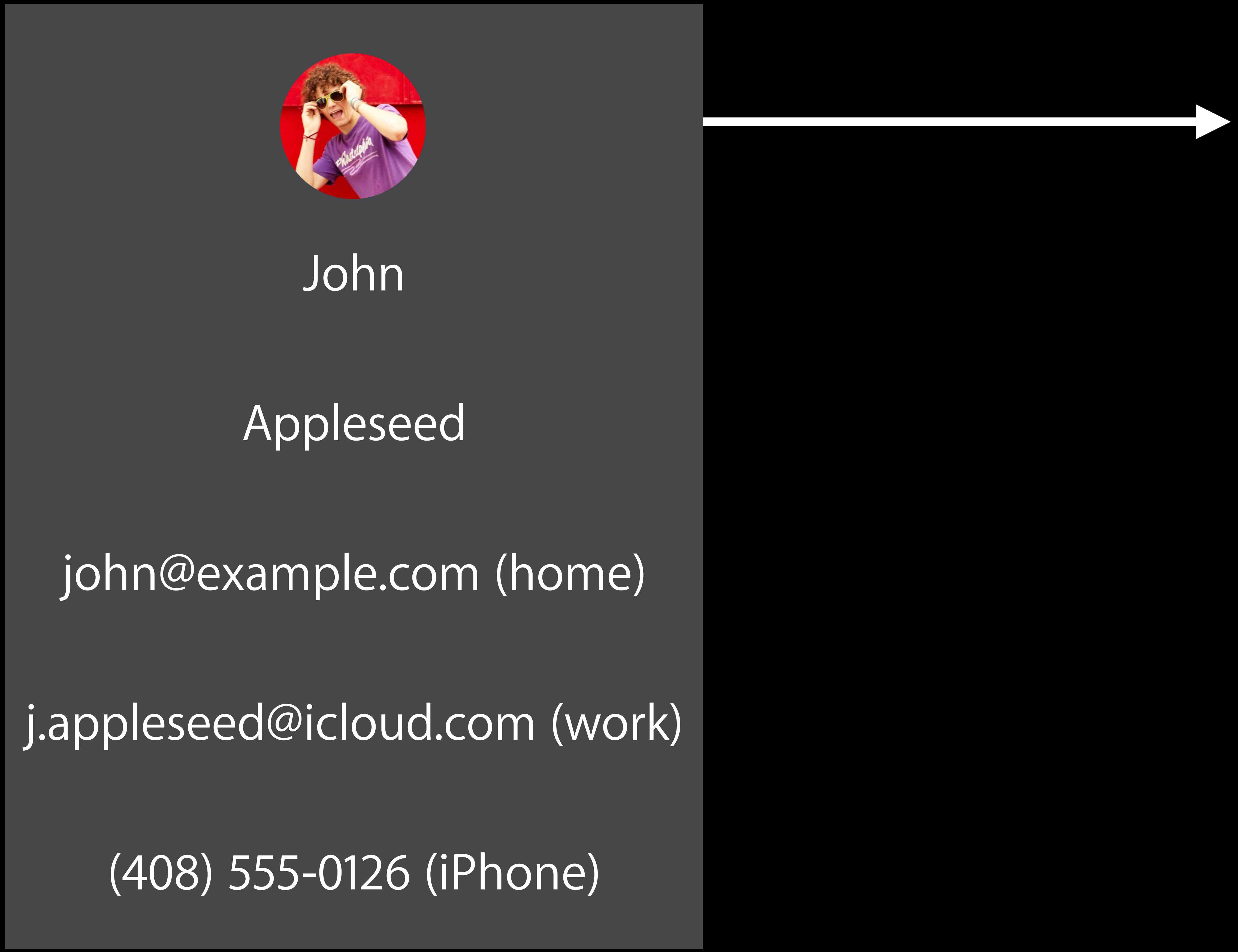
Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

Contact Properties



John

Appleseed

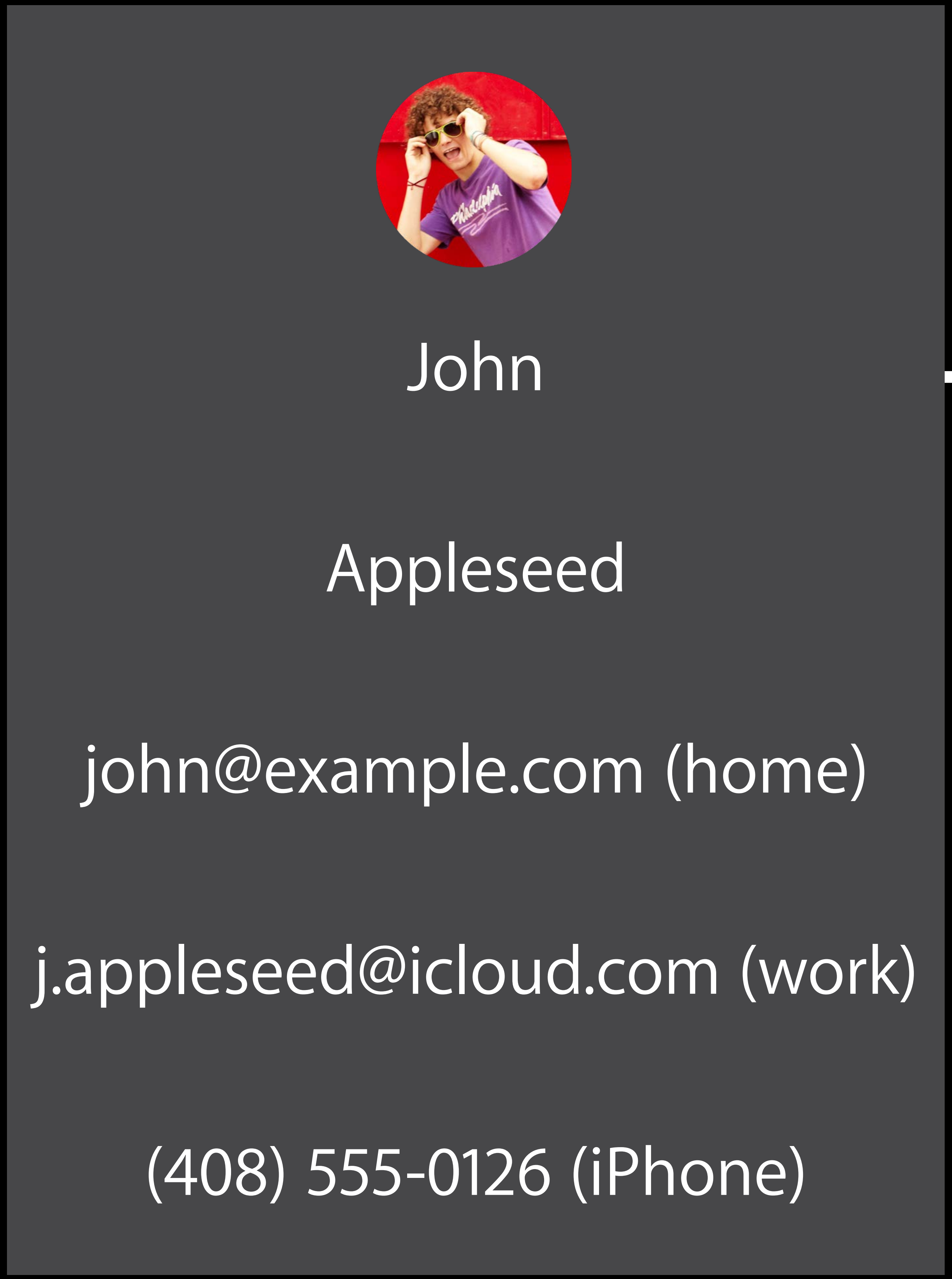
john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

```
contact.imageData

contact.givenName

contact.familyName

contact.emailAddresses

contact.phoneNumbers
```

Contact Properties



John

Appleseed

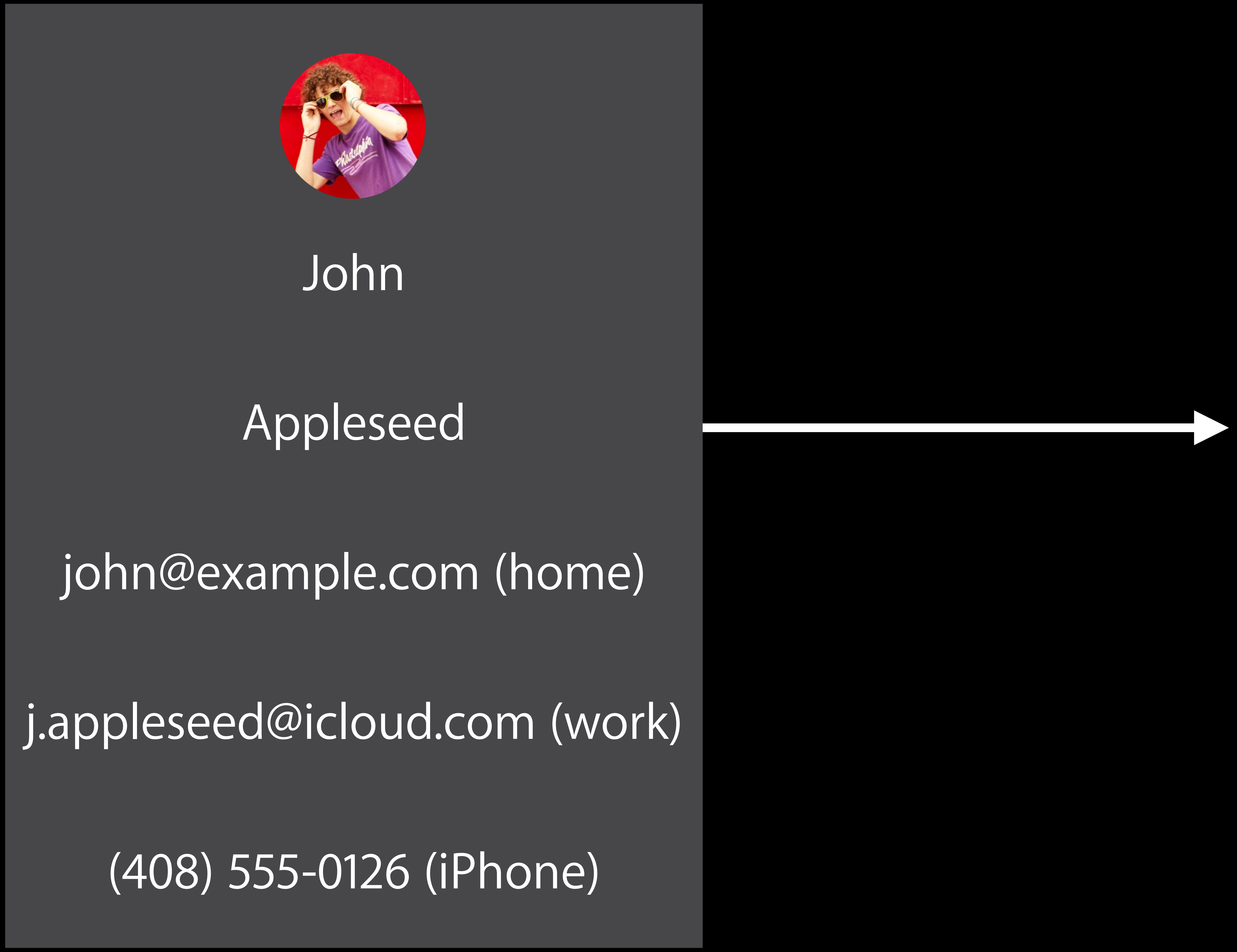
john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

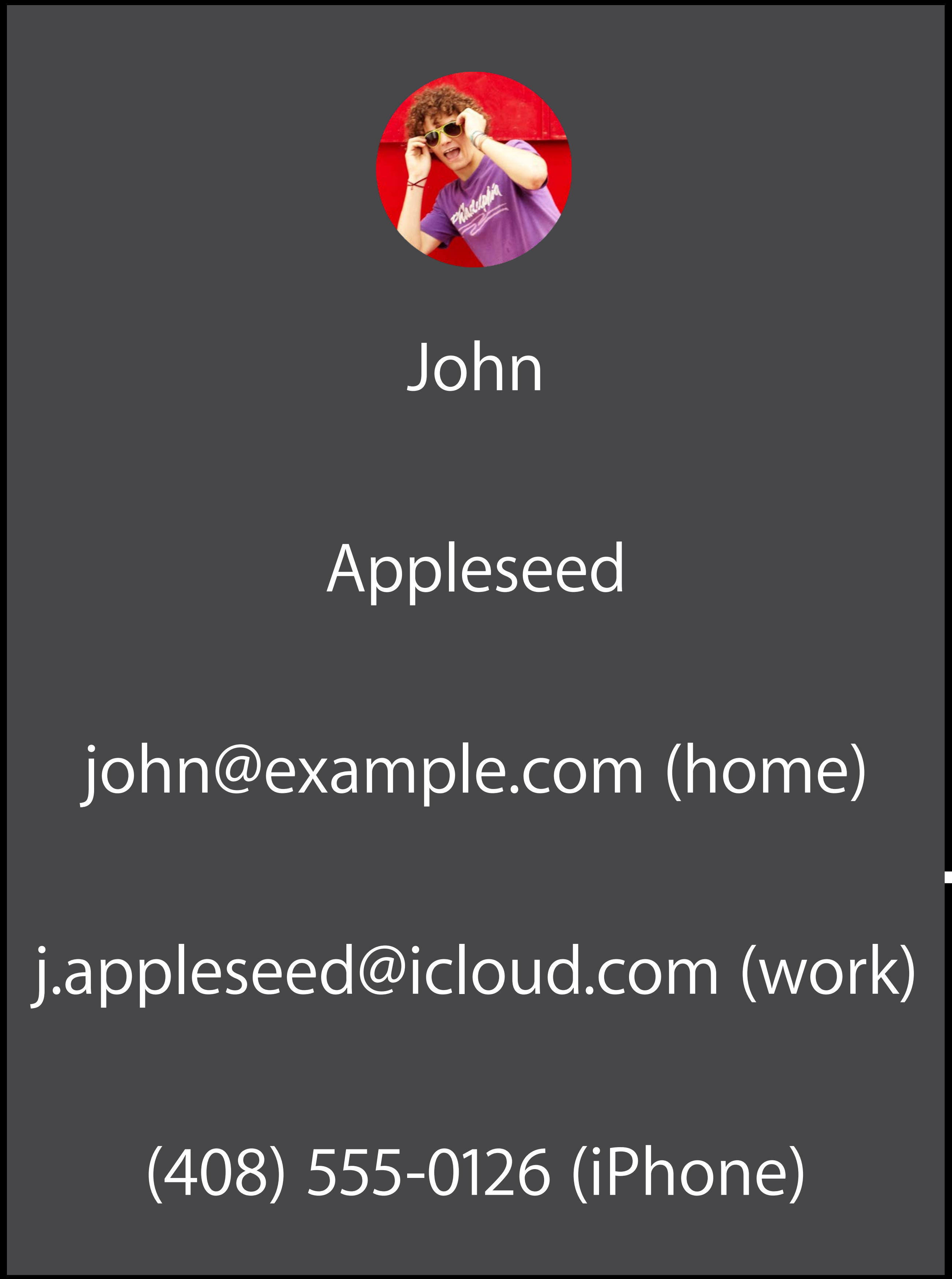
```
contact.imageData
contact.givenName
contact.familyName

contact.emailAddresses

contact.phoneNumbers
```

Contact Properties



```
contact.imageData
contact.givenName
contact.familyName
contact.emailAddresses
contact.phoneNumbers
```

Contact Properties



```
contact.imageData

contact.givenName

contact.familyName

contact.emailAddresses

contact.phoneNumbers
```

Contact Properties

John

Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

```
contact.imageData

contact.givenName

contact.familyName

contact.emailAddresses

contact.phoneNumbers
```

Contact Properties

John

Appleseed

john@example.com (home)

j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

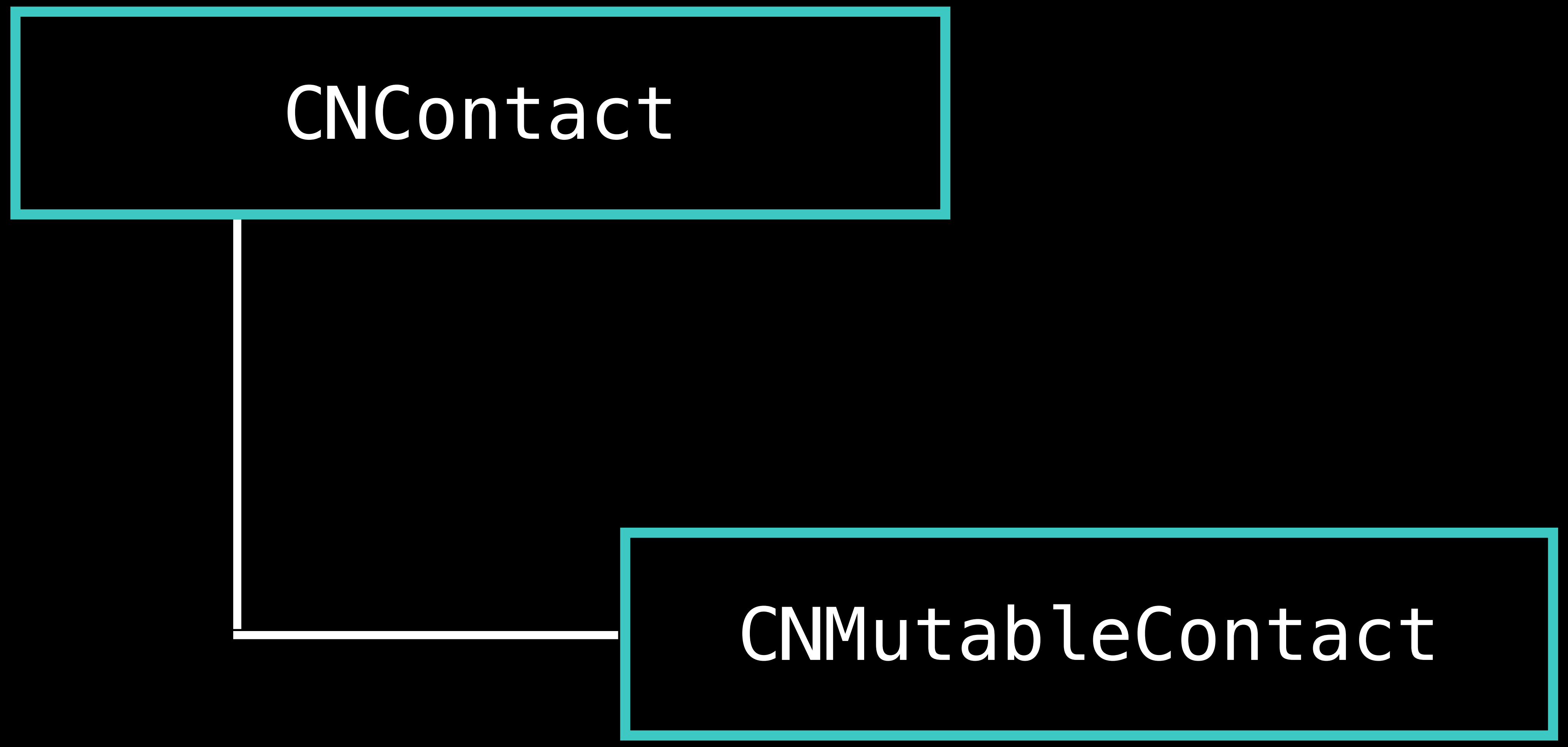
```
contact.imageData

contact.givenName

contact.familyName


contact.emailAddresses


contact.phoneNumbers
```

Contact Objects

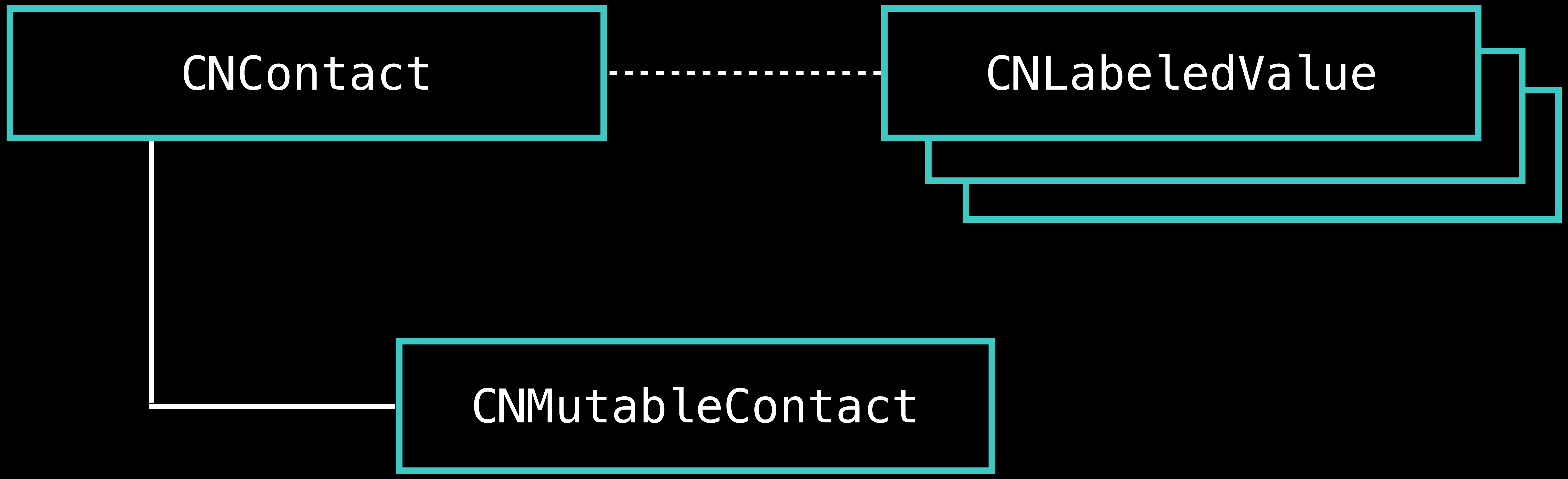
Contact Objects

`CNContact`

Contact Objects



Contact Objects



Creating a New Contact

```swift
import Contacts
```

Creating a New Contact

```swift
import Contacts
```

```swift
// create mutable for adding to the contact
let contact = CNMutableContact()
```

Creating a New Contact

```
import Contacts

// create mutable for adding to the contact
let contact = CNMutableContact()

contact.imageData = // profile picture as NSData
```

Creating a New Contact

```swift
import Contacts

// create mutable for adding to the contact
let contact = CNMutableContact()

contact.imageData = // profile picture as NSData
```

Creating a New Contact

```swift
import Contacts

// create mutable for adding to the contact
let contact = CNMutableContact()

contact.imageData = // profile picture as NSData

contact.givenName = "John"
contact.familyName = "Appleseed"
```

Creating a New Contact

Labeled Values

```
let homeEmail = CNLabeledValue(label: CNLabelHome, value: "john@example.com")
let workEmail = CNLabeledValue(label: CNLabelWork,
    value: "j.appleseed@icloud.com")
```

Creating a New Contact

Labeled Values

```
let homeEmail = CNLabeledValue(label: CNLabelHome, value: "john@example.com")
let workEmail = CNLabeledValue(label: CNLabelWork,
    value: "j.appleseed@icloud.com")
```

```
contact.emailAddresses = [homeEmail, workEmail]
```

Creating a New Contact

Labeled Values

```
let homeEmail = CNLabeledValue(label: CNLabelHome, value: "john@example.com")
let workEmail = CNLabeledValue(label: CNLabelWork,
    value: "j.appleseed@icloud.com")

contact.emailAddresses = [homeEmail, workEmail]
```

```
contact.phoneNumbers = [CNLabeledValue(
    label: CNLabelPhoneNumberiPhone,
    value: CNPhoneNumber(stringValue: "(408) 555-0126"))]
```

Creating a New Contact

Labeled Values

```
let address = CNMutablePostalAddress()
address.street = "774 Loma Vista Ave"
address.city = "Los Gatos"
address.state = "CA"
address.postalCode = "95032"
```

Creating a New Contact

Labeled Values

```
let address = CNMutablePostalAddress()
address.street = "774 Loma Vista Ave"
address.city = "Los Gatos"
address.state = "CA"
address.postalCode = "95032"

contact.postalAddresses = [CNLabeledValue(label: CNLabelHome,
    value: address)]
```

Creating a New Contact

Dates

```
let birthday = NSDateComponents()
birthday.day = 1
birthday.month = 4
birthday.year = 1988  // can omit for a year-less birthday
```

Creating a New Contact

Dates

```
let birthday = NSDateComponents()
birthday.day = 1
birthday.month = 4
birthday.year = 1988  // can omit for a year-less birthday

contact.birthday = birthday
```

Formatting Contact Data

Formatting Contact Data

```
let fullName = CNContactFormatter.stringFromContact(contact,
    style: .FullName)
```

Formatting Contact Data

```
let fullName = CNContactFormatter.stringFromContact(contact,
    style: .FullName)
print(fullName)
// John Appleseed
```

Formatting Contact Data

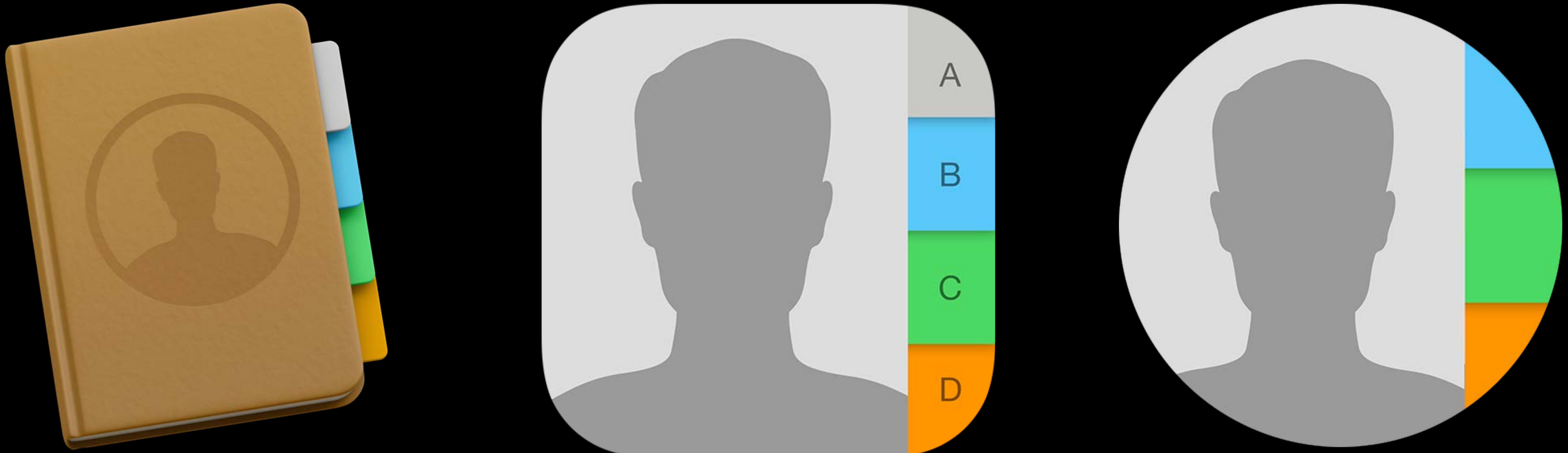
```swift
let fullName = CNContactFormatter.stringFromContact(contact,
    style: .FullName)
print(fullName)
// John Appleseed

let postalString = CNPostalAddressFormatter.stringFromPostalAddress(address)
```

Formatting Contact Data

```
let fullName = CNContactFormatter.stringFromContact(contact,
    style: .FullName)
print(fullName)
// John Appleseed

let postalString = CNPostalAddressFormatter.stringFromPostalAddress(address)
print(postalString)
// 774 Loma Vista Ave
// Los Gatos, CA 95032
```

Using Contacts in Your App

Dave Dribin OS X Contacts Engineer


A
B
C
D

A
B
C
D

Using Contacts in Your App

`CNContactStore`

Fetching User's Contacts

```
class CNContactStore : NSObject {
    func unifiedContactsMatchingPredicate(
        predicate: NSPredicate,
        keysToFetch: [CNKeyDescriptor]) -> [CNContact] throws
    ...
}
```

Predicates

```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
```

Predicates

```swift
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
```

John

Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

April 1, 1988 (birthday)

Jane

Appleseed

jane@example.com (home)

(505) 555-0155 (home)
(408) 555-0166 (work)

Craig

Bromley

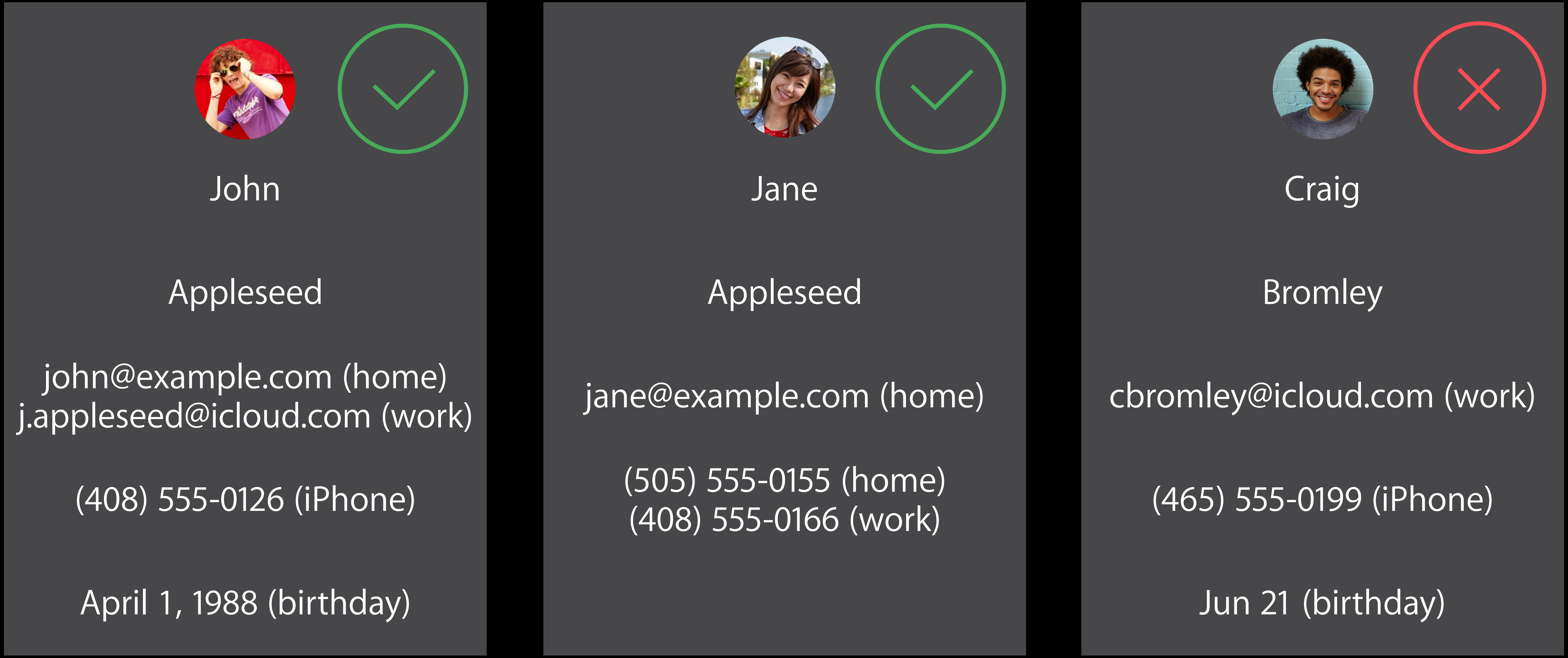
cbromley@icloud.com (work)

(465) 555-0199 (iPhone)

Jun 21 (birthday)

Predicates

```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
```



✓ John

Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

April 1, 1988 (birthday)

✓ Jane

Appleseed

jane@example.com (home)

(505) 555-0155 (home)
(408) 555-0166 (work)

✗ Craig

Bromley

cbromley@icloud.com (work)

(465) 555-0199 (iPhone)

Jun 21 (birthday)

Keys to Fetch

```swift
let keysToFetch = ["givenName", "familyName"]
```

		imageData
John	Jane	givenName
Appleseed	Appleseed	familyName
john@example.com (home) j.appleseed@icloud.com (work)	jane@example.com (home)	emailAddresses
(408) 555-0126 (iPhone)	(505) 555-0155 (home) (408) 555-0166 (work)	phoneNumbers
April 1, 1988 (birthday)		birthday

Keys to Fetch

```
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
```

John

Appleseed

john@example.com (home)
j.appleseed@icloud.com (work)

(408) 555-0126 (iPhone)

April 1, 1988 (birthday)

Jane

Appleseed

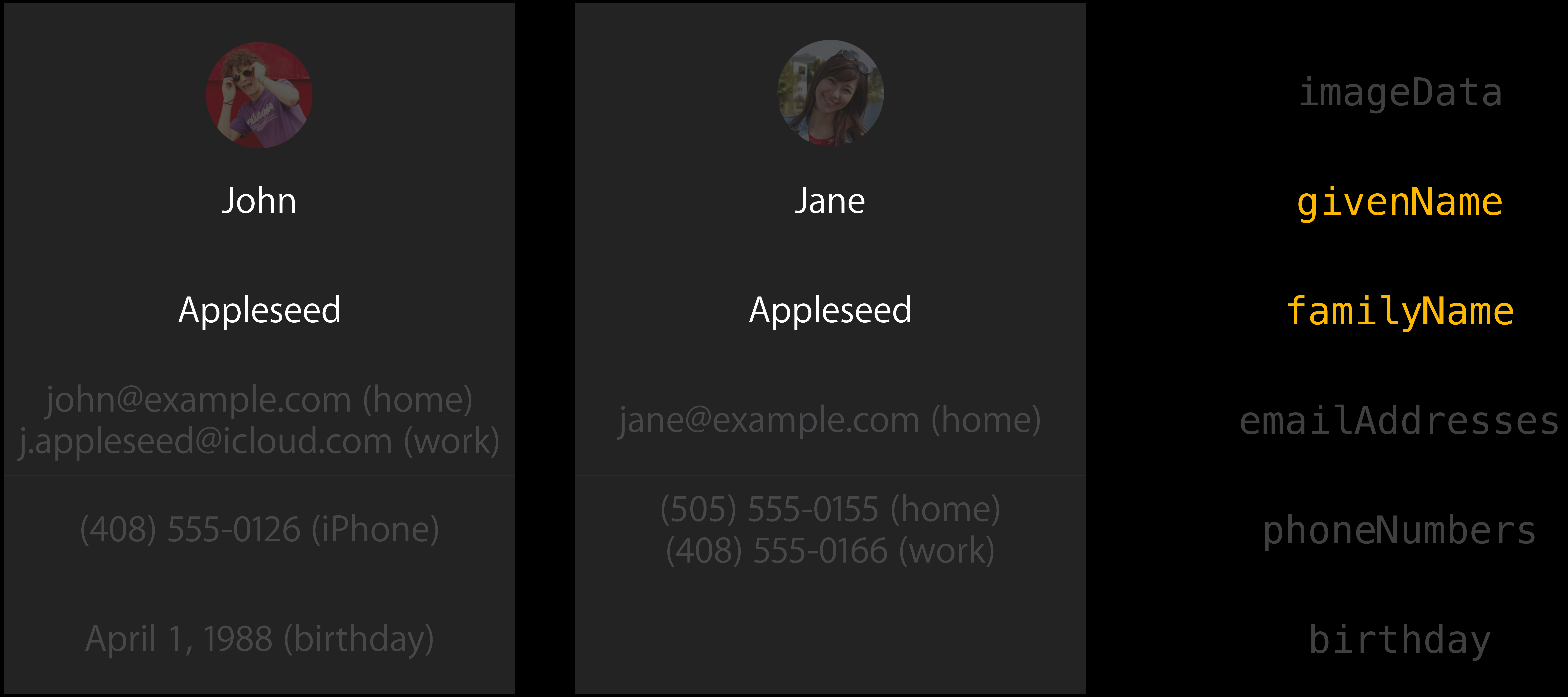
jane@example.com (home)

(505) 555-0155 (home)
(408) 555-0166 (work)

```
imageData
givenName
familyName
emailAddresses
phoneNumbers
birthday
```

Keys to Fetch

```
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
```



How to Fetch

```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
```

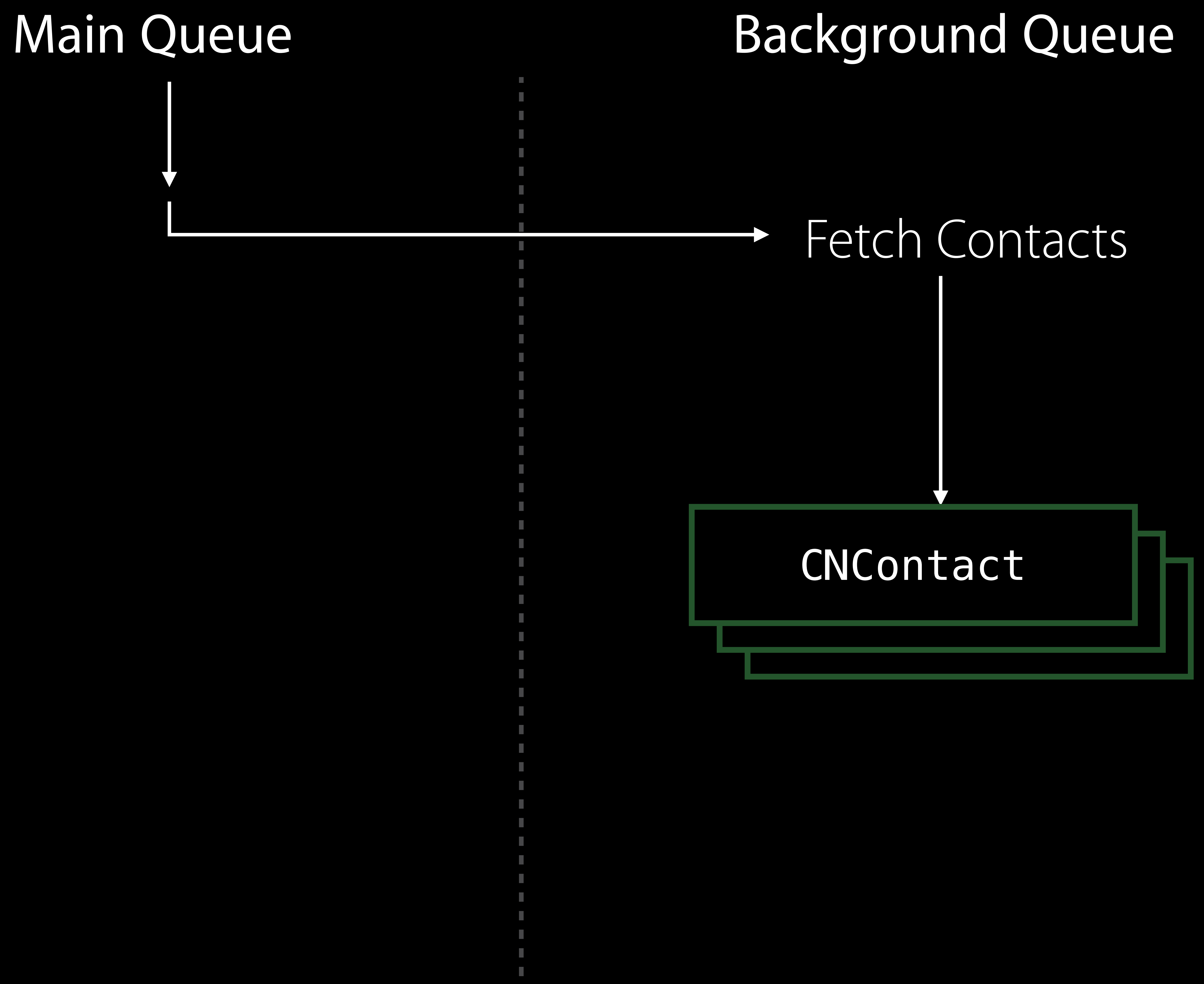
How to Fetch

```swift
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]

let store = CNContactStore()
let contacts = try store.unifiedContactsMatchingPredicate(predicate,
    keysToFetch: keysToFetch)
```

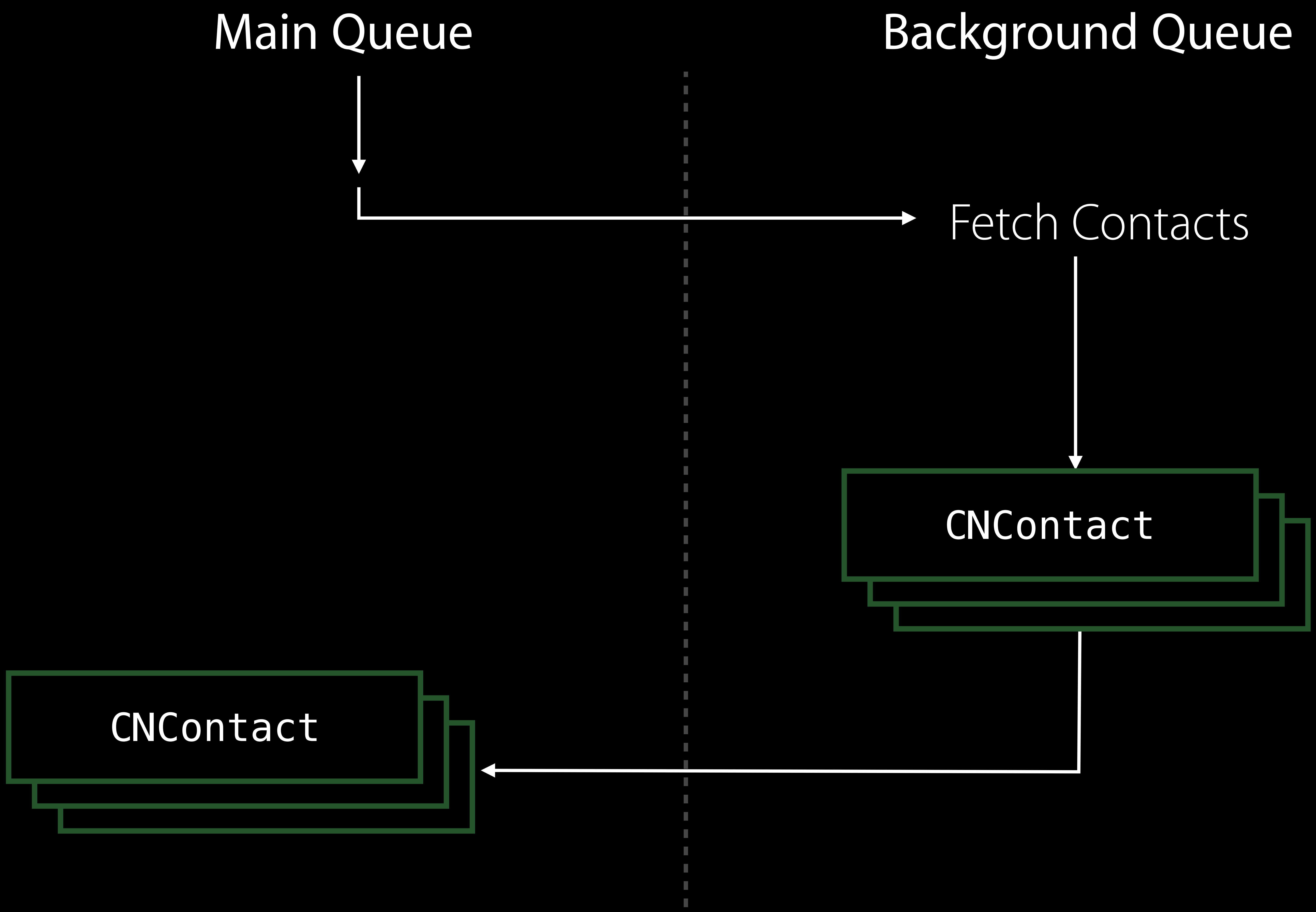
How to Fetch

```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]

let store = CNContactStore()
let contacts = try store.unifiedContactsMatchingPredicate(predicate,
    keysToFetch: keysToFetch)

for contact in contacts {
   print("\(contact.givenName) \(contact.familyName)")
}
```

Keeping UI Responsive



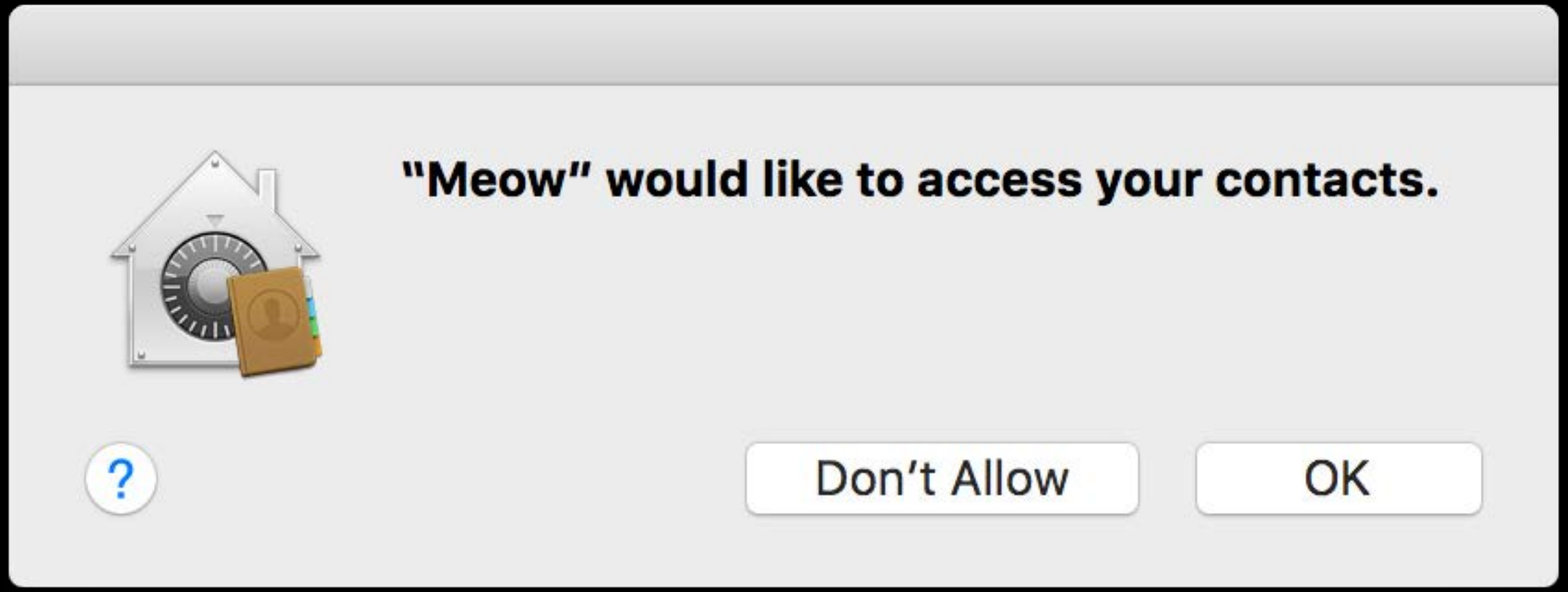
Keeping UI Responsive



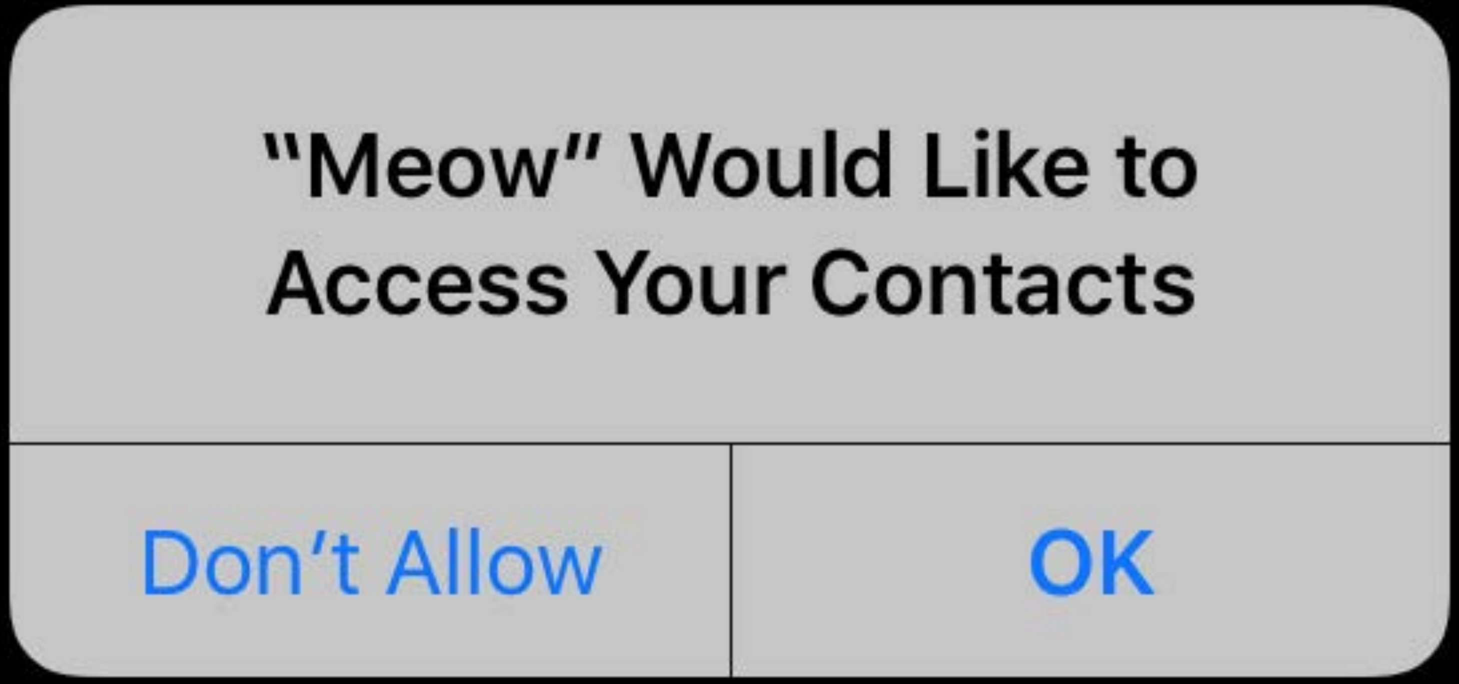
Data Privacy

Data Privacy

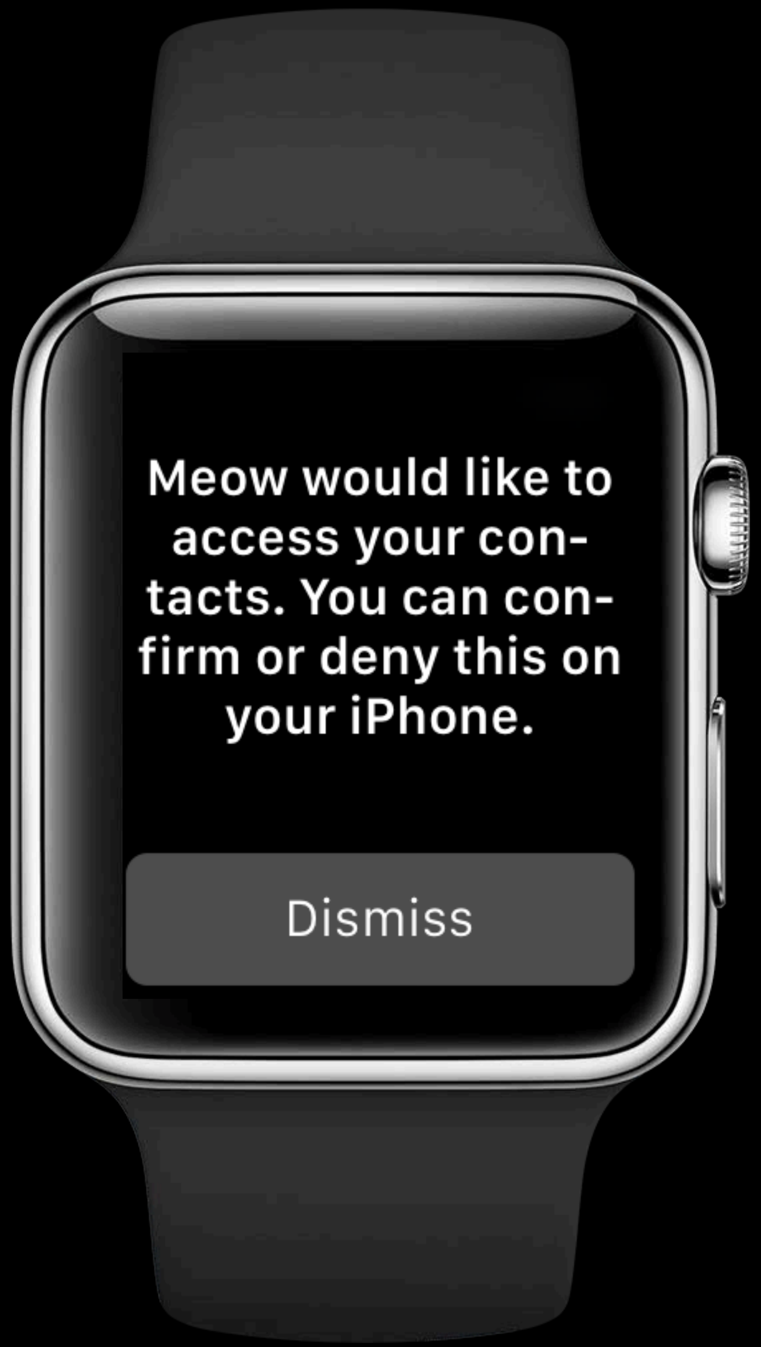
```
class CNContactStore : NSObject {
    func requestAccessForEntityType(..., completionHandler:)
}
```



"Meow" would like to access your contacts.



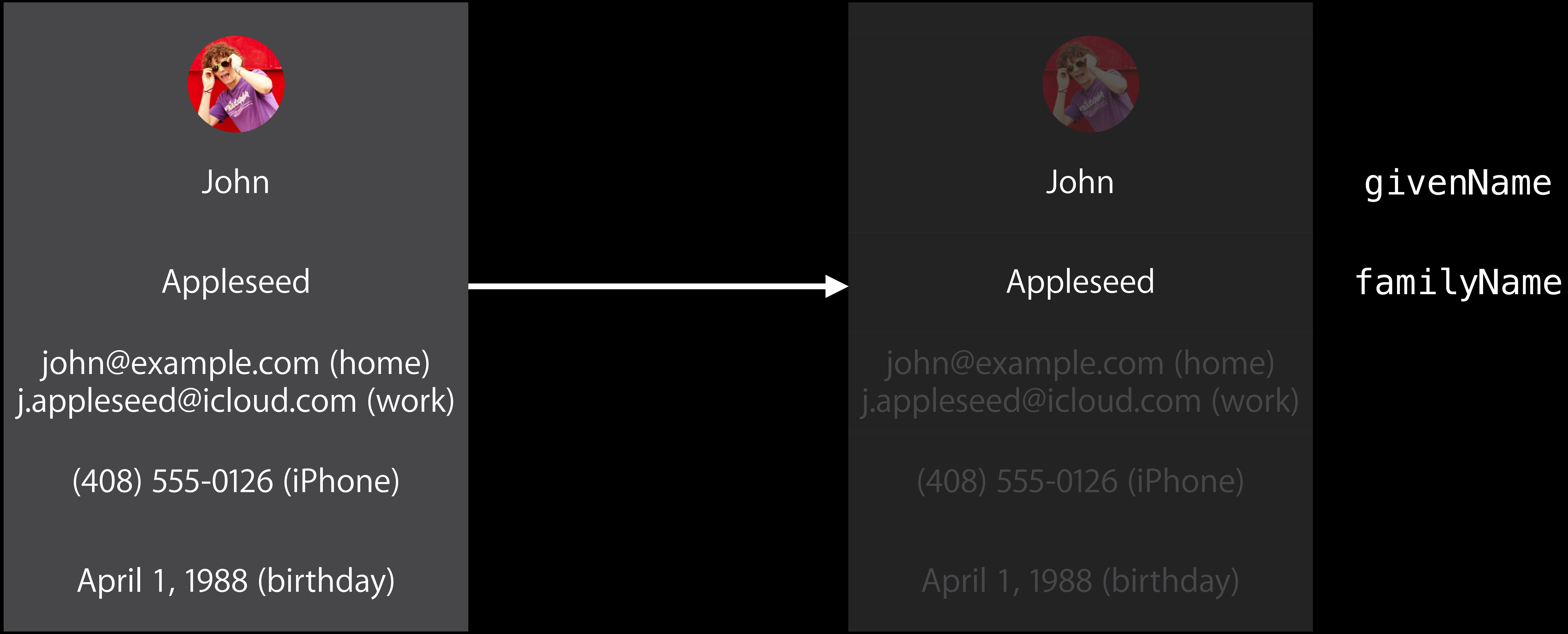
"Meow" Would Like to Access Your Contacts



Meow would like to access your contacts. You can confirm or deny this on your iPhone.

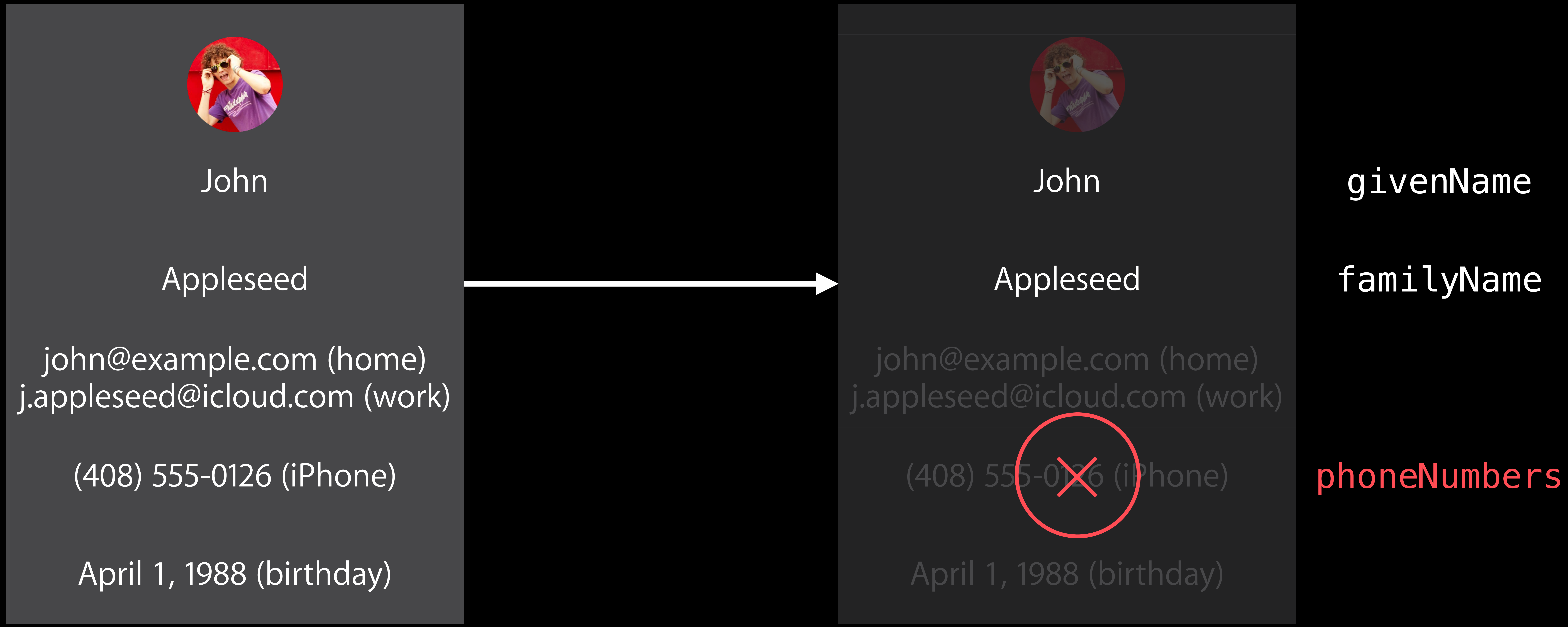
Partial Contacts

```swift
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
```



Partial Contacts

```
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
```



Partial Contacts

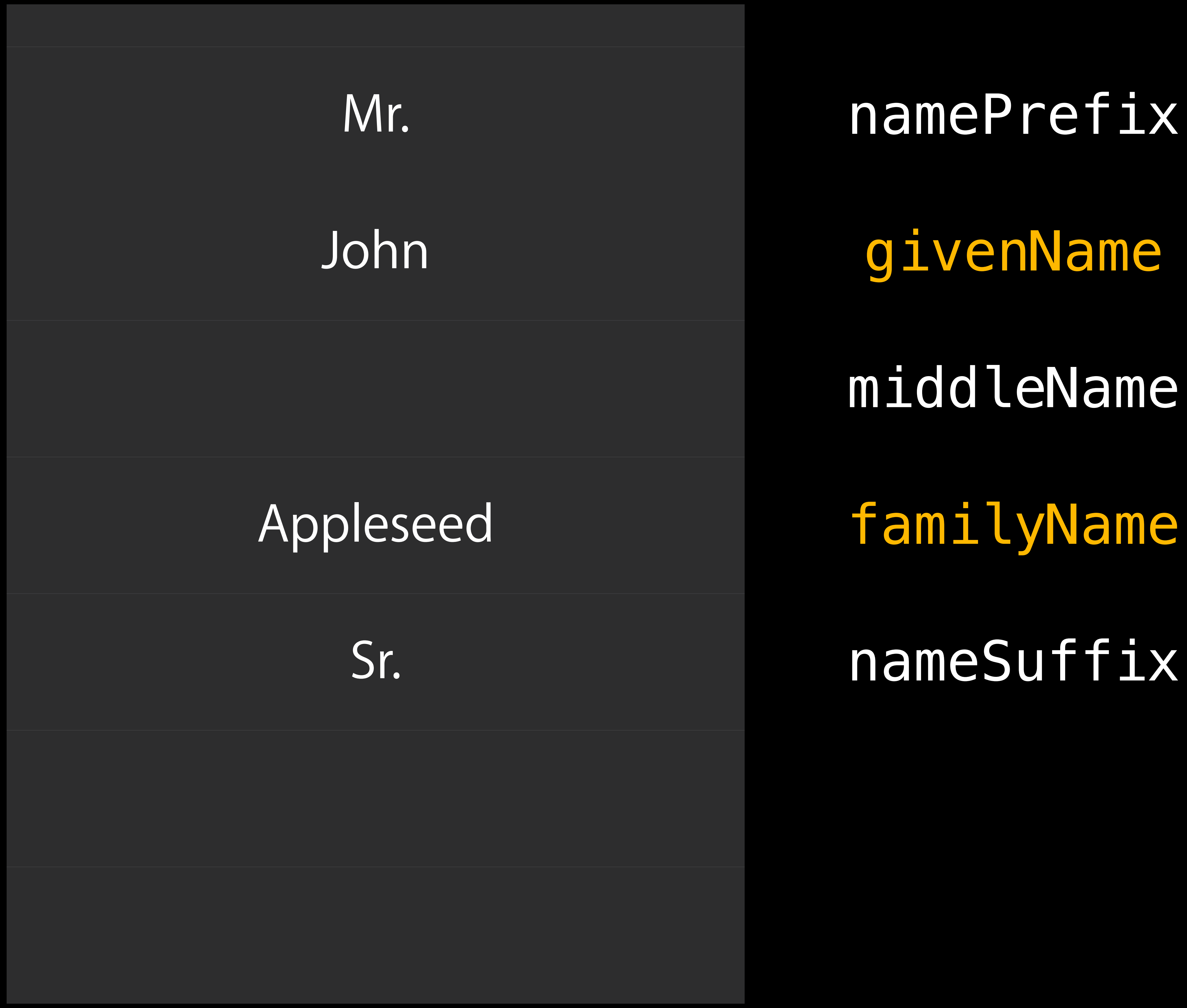
```
if (contact.isKeyAvailable(CNContactPhoneNumbersKey)) {
    print("\(contact.phoneNumbers)")
}
```

Partial Contacts

```
if (contact.isKeyAvailable(CNContactPhoneNumbersKey)) {
    print("\(contact.phoneNumbers)")
} else {
    let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey,
                       CNContactPhoneNumbersKey]
    var refetchedContact = try store.unifiedContactWithIdentifier(
        contact.identifier, keysToFetch: keysToFetch)
    print("\(refetchedContact.phoneNumbers)")
}
```

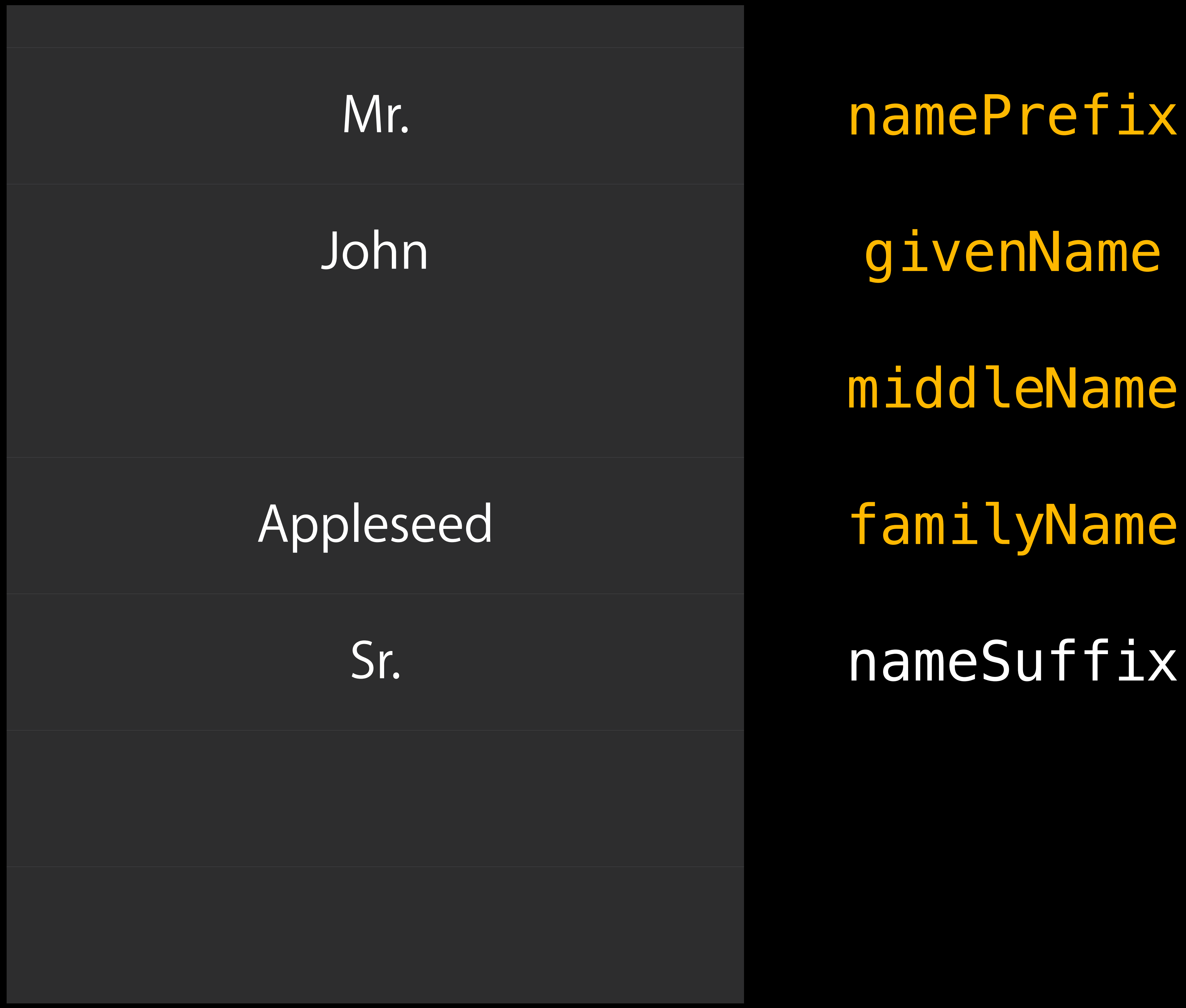
Formatting Partial Contacts

```
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey]
```



Formatting Partial Contacts

```
let keysToFetch = [CNContactGivenNameKey, CNContactFamilyNameKey,
    CNContactNamePrefixKey, CNContactMiddleNameKey, ...]
```



Formatting Partial Contacts

```
let keysToFetch =
   [CNContactFormatter.descriptorForRequiredKeysForStyle(.FullName)]
```

Mr.	namePrefix
John	givenName
	middleName
Appleseed	familyName
Sr.	nameSuffix
	+ Any Others

Key Descriptors

```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [
    CNContactFormatter.descriptorForRequiredKeysForStyle(.FullName),
    CNContactEmailAddressesKey]
```

Key Descriptors

```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [
    CNContactFormatter.descriptorForRequiredKeysForStyle(.FullName),
    CNContactEmailAddressesKey]

let contacts = try store.unifiedContactsMatchingPredicate(predicate,
    keysToFetch: keysToFetch)
```

Key Descriptors

```
let predicate = CNContact.predicateForContactsMatchingName("Appleseed")
let keysToFetch = [
    CNContactFormatter.descriptorForRequiredKeysForStyle(.FullName),
    CNContactEmailAddressesKey]

let contacts = try store.unifiedContactsMatchingPredicate(predicate,
    keysToFetch: keysToFetch)

for contact in contacts {
   let fullName = CNContactFormatter.stringFromContact(
        contact, style: .FullName) ?? "No Name"
    print("\(fullName): \(contact.emailAddresses)")
}
```

Unified Contacts

iCloud	Facebook
John	John
Appleseed	Appleseed
j.appleseed@icloud.com (work)	john@example.com (home)
(408) 555-0126 (iPhone)	
	April 1, 1988 (birthday)

Unified Contacts

iCloud	Facebook	Unified
John	John	John
Appleseed	Appleseed	Appleseed
j.appleseed@icloud.com (work)	john@example.com (home)	j.appleseed@icloud.com (work) john@example.com (home)
(408) 555-0126 (iPhone)		(408) 555-0126 (iPhone)
	April 1, 1988 (birthday)	April 1, 1988 (birthday)

Adding a New Contact

```
let john = CNMutableContact()
john.givenName = "John"
john.familyName = "Appleseed"
```

Adding a New Contact

```
let john = CNMutableContact()
john.givenName = "John"
john.familyName = "Appleseed"
```

```
let saveRequest = CNSaveRequest()
saveRequest.addContact(john, toContainerWithIdentifier: nil)
try store.executeSaveRequest(saveRequest)
```

Updating an Existing Contact

```
let updatedContact = contact.mutableCopy()
let newEmail = CNLabeledValue(label: CNLabelHome,
    value: "john@example.com")
updatedContact.emailAddresses.append(newEmail)
```

Updating an Existing Contact

```swift
let updatedContact = contact.mutableCopy()
let newEmail = CNLabeledValue(label: CNLabelHome,
    value: "john@example.com")
updatedContact.emailAddresses.append(newEmail)

let saveRequest = CNSaveRequest()
saveRequest.updateContact(updatedContact)
try store.executeSaveRequest(saveRequest)
```

Contacts in the UI

Julien Robert iOS Contacts Engineer

ContactsUI

New framework

ContactsUI

New framework

iOS	OS X

ContactsUI

New framework

iOS	OS X
`CNContactPickerViewController`	`CNContactPicker`

ContactsUI

New framework

iOS	OS X
`CNContactPickerViewController`	`CNContactPicker`
`CNContactViewController`	`CNContactViewController`

Picking Contacts

CNContactPickerViewController

Picking Contacts

CNContactPickerViewController

Modern replacement for `ABPeoplePickerNavigationController`

Picking Contacts

CNContactPickerViewController

Modern replacement for `ABPeoplePickerNavigationController`

Must be presented, not pushed

Picking Contacts

CNContactPickerViewController

Modern replacement for `ABPeoplePickerNavigationController`

Must be presented, not pushed

Always out of process, no contacts access dialog

Picking Contacts

CNContactPickerViewController

Modern replacement for `ABPeoplePickerNavigationController`

Must be presented, not pushed

Always out of process, no contacts access dialog

May return partial contacts

Picking Contacts

CNContactPickerViewController

Modern replacement for `ABPeoplePickerNavigationController`

Must be presented, not pushed

Always out of process, no contacts access dialog

May return partial contacts

Behavior based on delegate methods and predicates

Picking Contacts

CNContactPickerViewController

Modern replacement for `ABPeoplePickerNavigationController`

Must be presented, not pushed

Always out of process, no contacts access dialog

May return partial contacts

Behavior based on delegate methods and predicates

Supports multi-selection

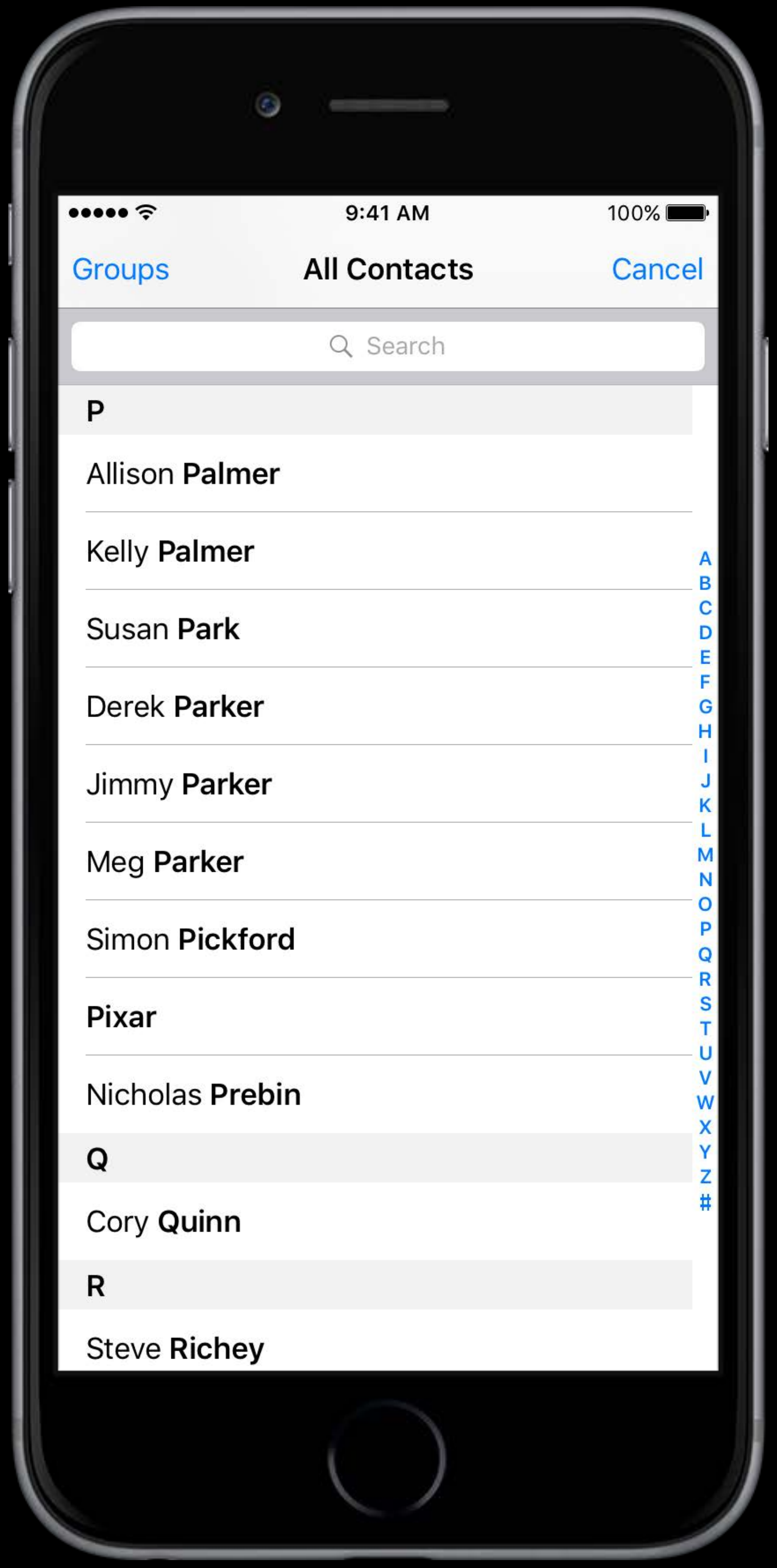
Picking Contacts

Delegate methods

Picking Contacts

Delegate methods

Single contact



Picking Contacts

Delegate methods

Single contact

```
contactPicker(picker, didSelectContact contact: CNContact)
```

Picking Contacts

Delegate methods

Single contact

```
contactPicker(picker, didSelectContact contact: CNContact)
```

Single property

```
contactPicker(picker, didSelectContactProperty property: CNContactProperty)
```

Picking Contacts

Delegate methods

Single contact

```
contactPicker(picker, didSelectContact contact: CNContact)
```

Single property

```
contactPicker(picker, didSelectContactProperty property: CNContactProperty)
```

```
class CNContactProperty {
   var contact: CNContact
   var key: NSString
   var value: AnyObject?
   var identifier: NSString?
}
```

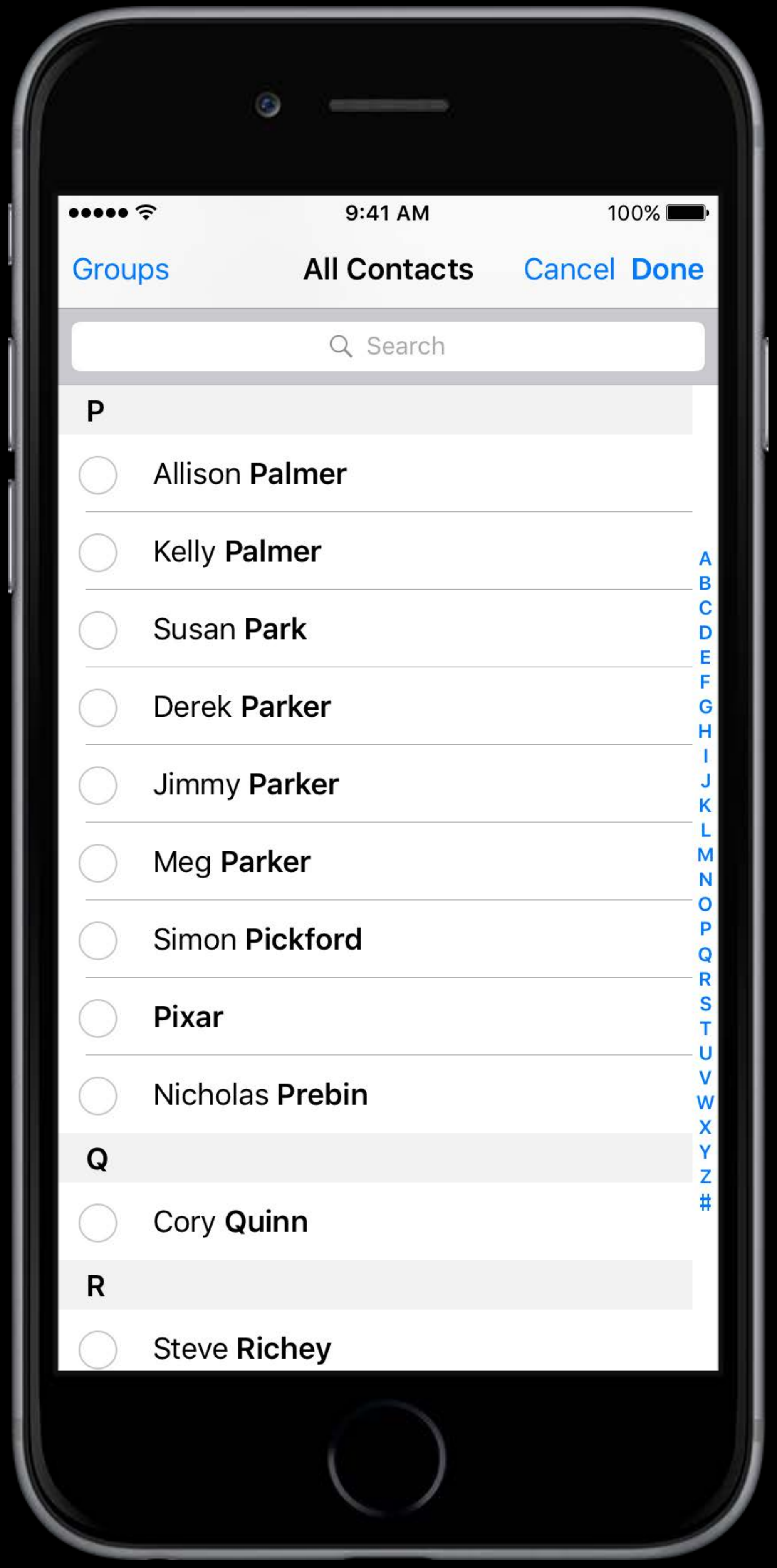
Picking Contacts

Delegate methods

Picking Contacts

Delegate methods

Multiple contacts



Picking Contacts

Delegate methods

Multiple contacts

```
contactPicker(picker, didSelectContacts contacts: [CNContact])
```

Picking Contacts

Delegate methods

Multiple contacts

```
contactPicker(picker, didSelectContacts contacts: [CNContact])
```

Multiple properties

```
contactPicker(picker, didSelectContactProperties properties: [CNContactProperty])
```

Picking Contacts
Predicates

Picking Contacts
Predicates

`predicateForEnablingContact`

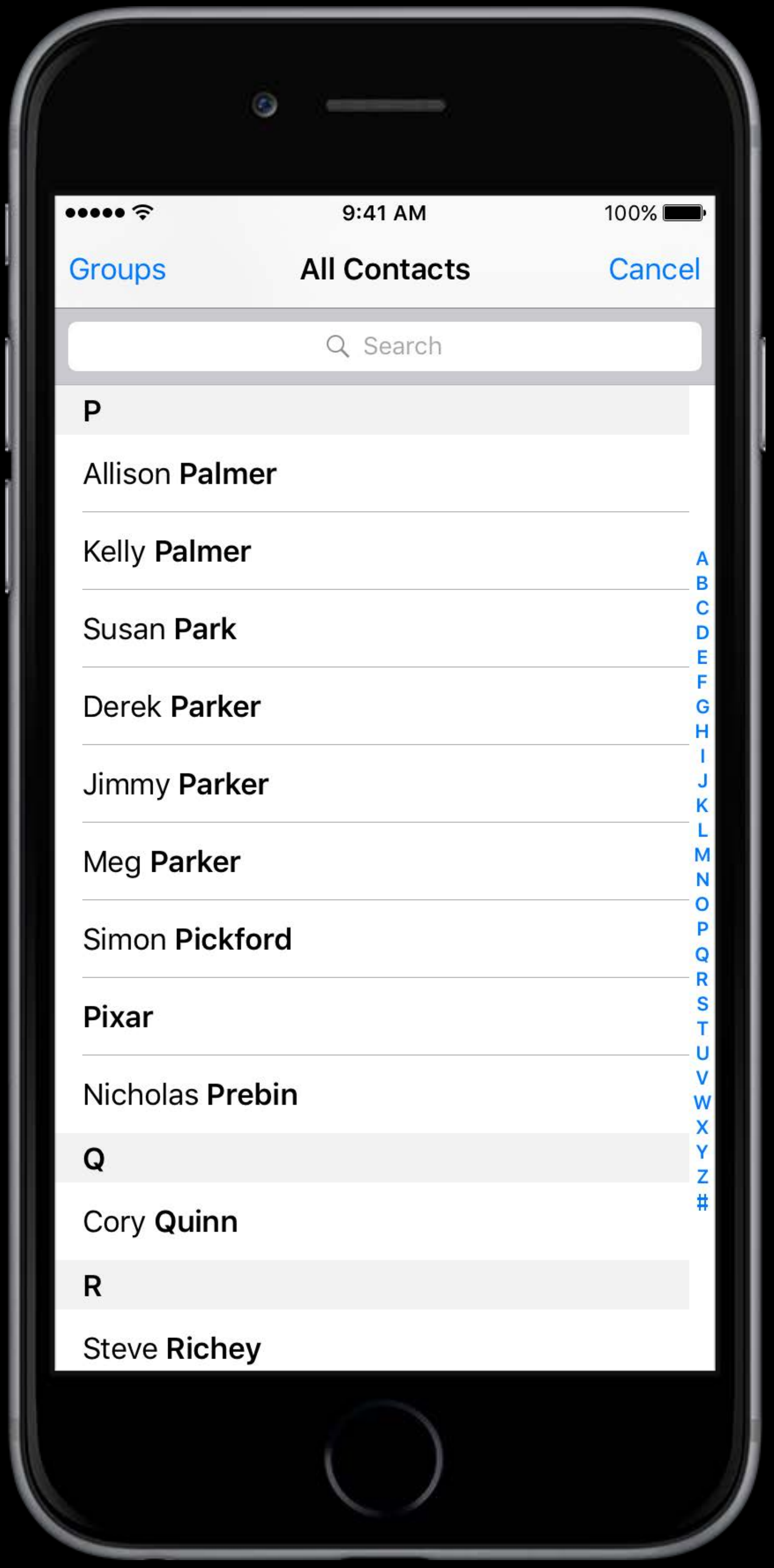
- Which contacts are available
- Evaluated on `CNContact`

Picking Contacts

Predicates

`predicateForEnablingContact`

- Which contacts are available
- Evaluated on `CNContact`



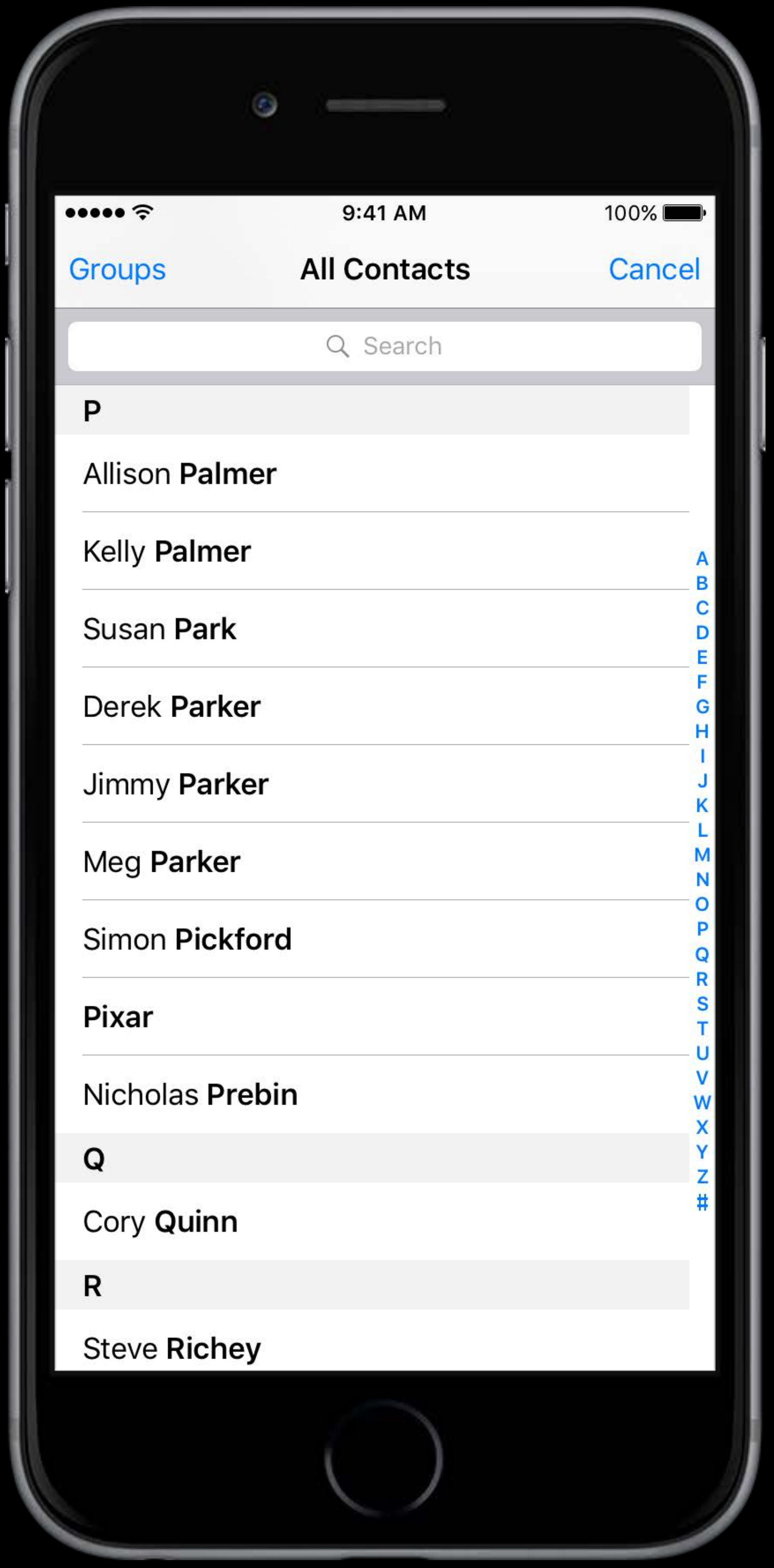
Picking Contacts

Predicates

`predicateForEnablingContact`

- Which contacts are available
- Evaluated on `CNContact`

```
let predicate = NSPredicate(format: "familyName LIKE[cd] 'parker'")
```



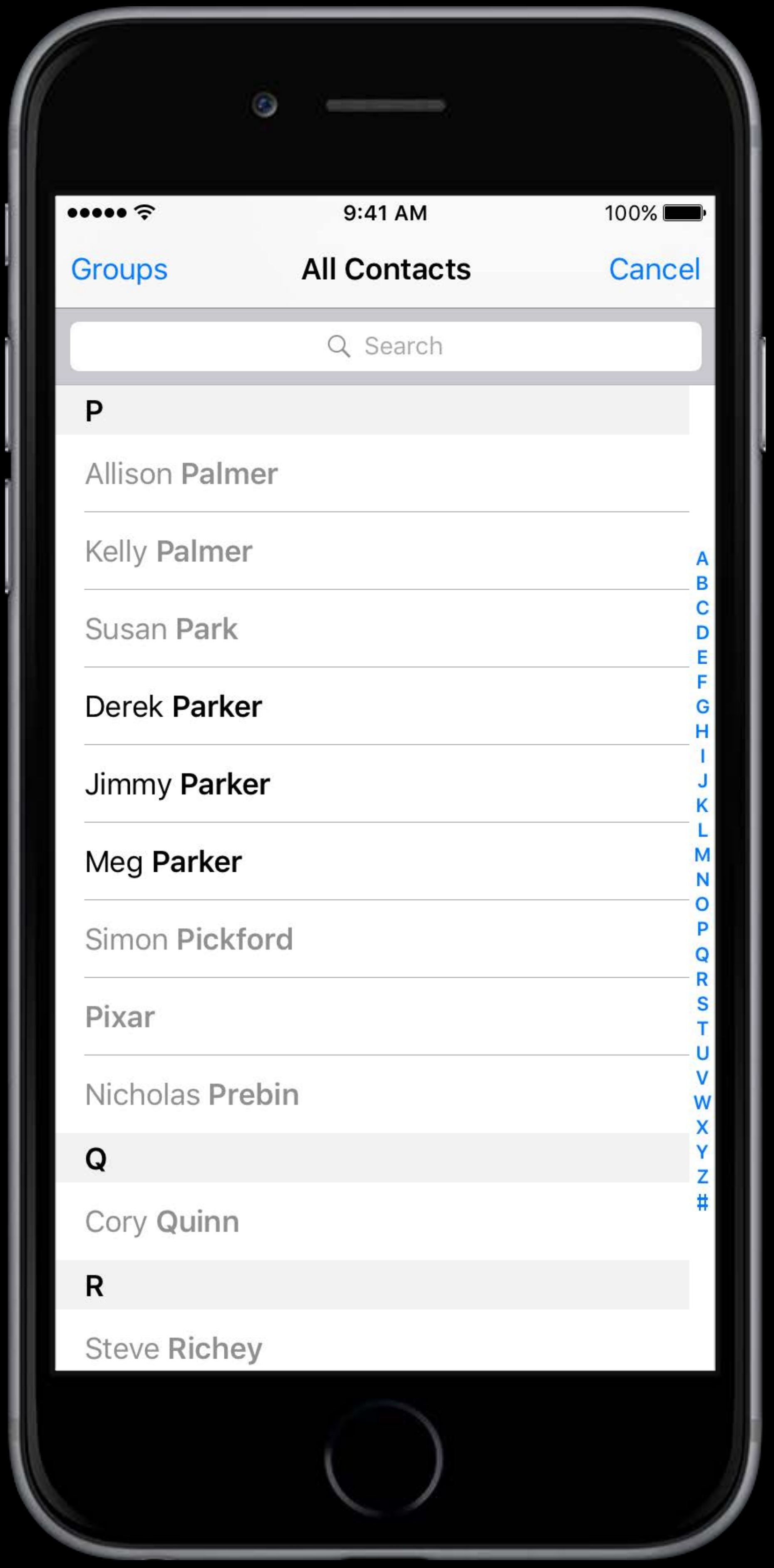
Picking Contacts

Predicates

`predicateForEnablingContact`

- Which contacts are available
- Evaluated on `CNContact`

```
let predicate = NSPredicate(format: "familyName LIKE[cd] 'parker'")
contactPicker.predicateForEnablingContact = predicate
```



Picking Contacts
Predicates

Picking Contacts

Predicates

`predicateForSelectionOfContact`

- Which contacts are returned when tapped, others push the card

Picking Contacts

Predicates

`predicateForSelectionOfContact`

- Which contacts are returned when tapped, others push the card

`predicateForSelectionOfProperty`

- Which properties are returned when tapped, others perform the default action
- Evaluated on `CNContactProperty`

Picking Contacts

Predicates

`predicateForSelectionOfContact`

- Which contacts are returned when tapped, others push the card

`predicateForSelectionOfProperty`

- Which properties are returned when tapped, others perform the default action
- Evaluated on `CNContactProperty`

Coherence between predicates and delegate methods

Viewing Contacts

CNContactViewController

Viewing Contacts

CNContactViewController

One class to replace

- `ABPersonViewController`
- `ABNewPersonViewController`
- `ABUnknownPersonViewController`

Viewing Contacts

CNContactViewController

Viewing Contacts
CNContactViewController

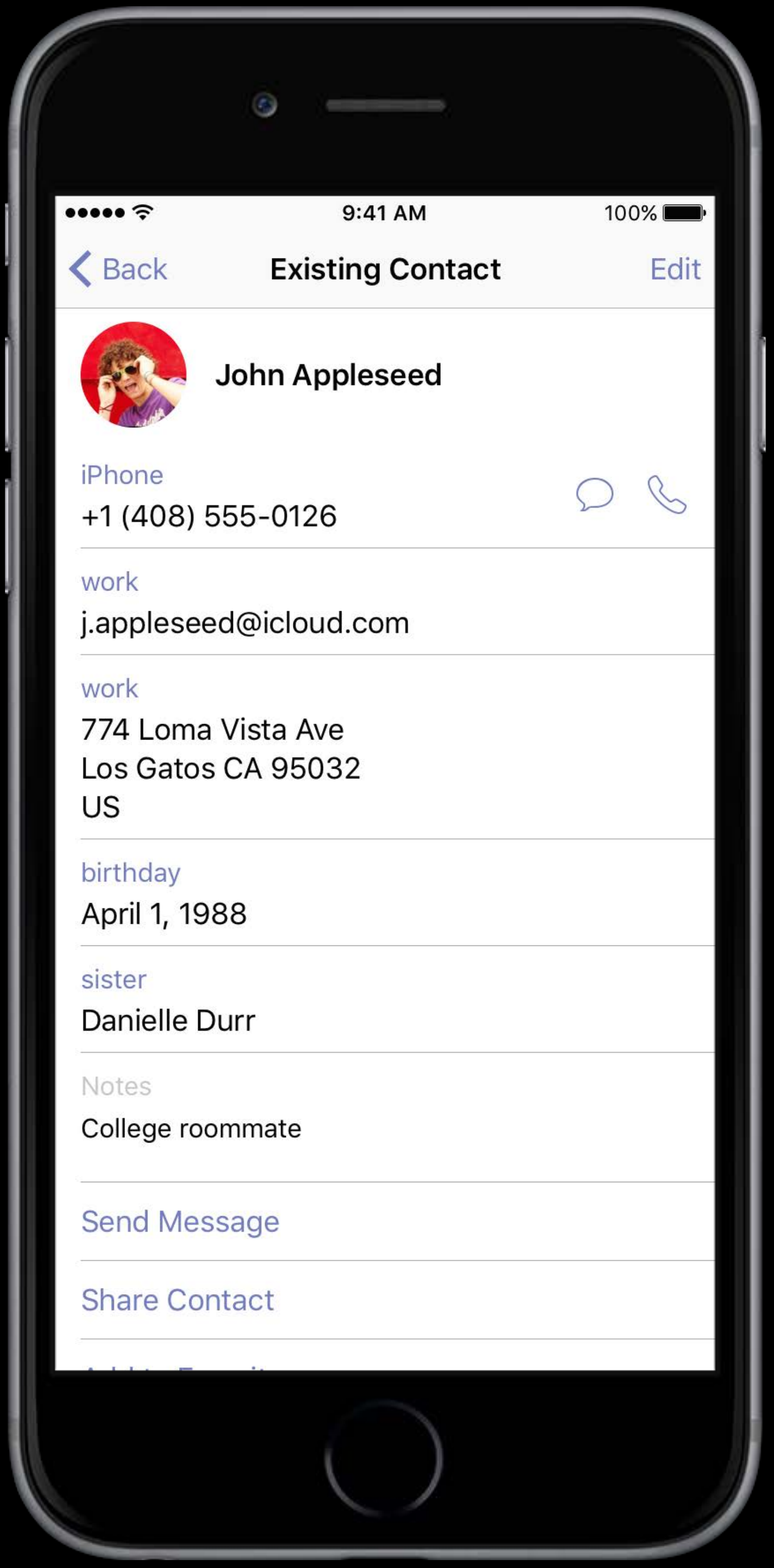
Use appropriate creation method

Viewing Contacts

CNContactViewController

Use appropriate creation method

- `viewControllerForContact:`

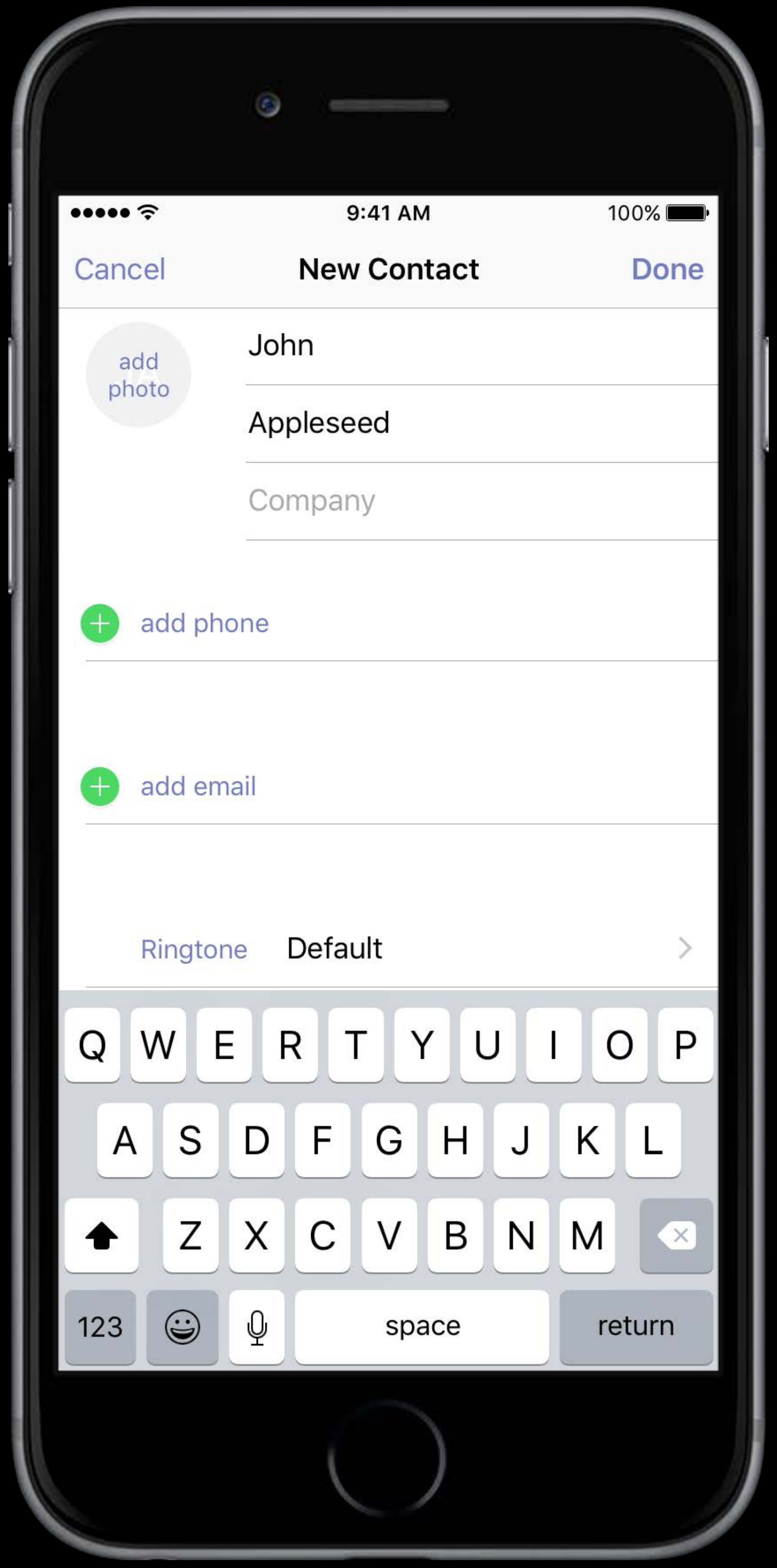


9:41 AM
100%
Back
Existing Contact
Edit
John Appleseed
iPhone
+1 (408) 555-0126
work
j.appleseed@icloud.com
work
774 Loma Vista Ave
Los Gatos CA 95032
US
birthday
April 1, 1988
sister
Danielle Durr
Notes
College roommate
Send Message
Share Contact

Viewing Contacts
CNContactViewController

Use appropriate creation method

- `viewControllerForContact:`
- `viewControllerForNewContact:`

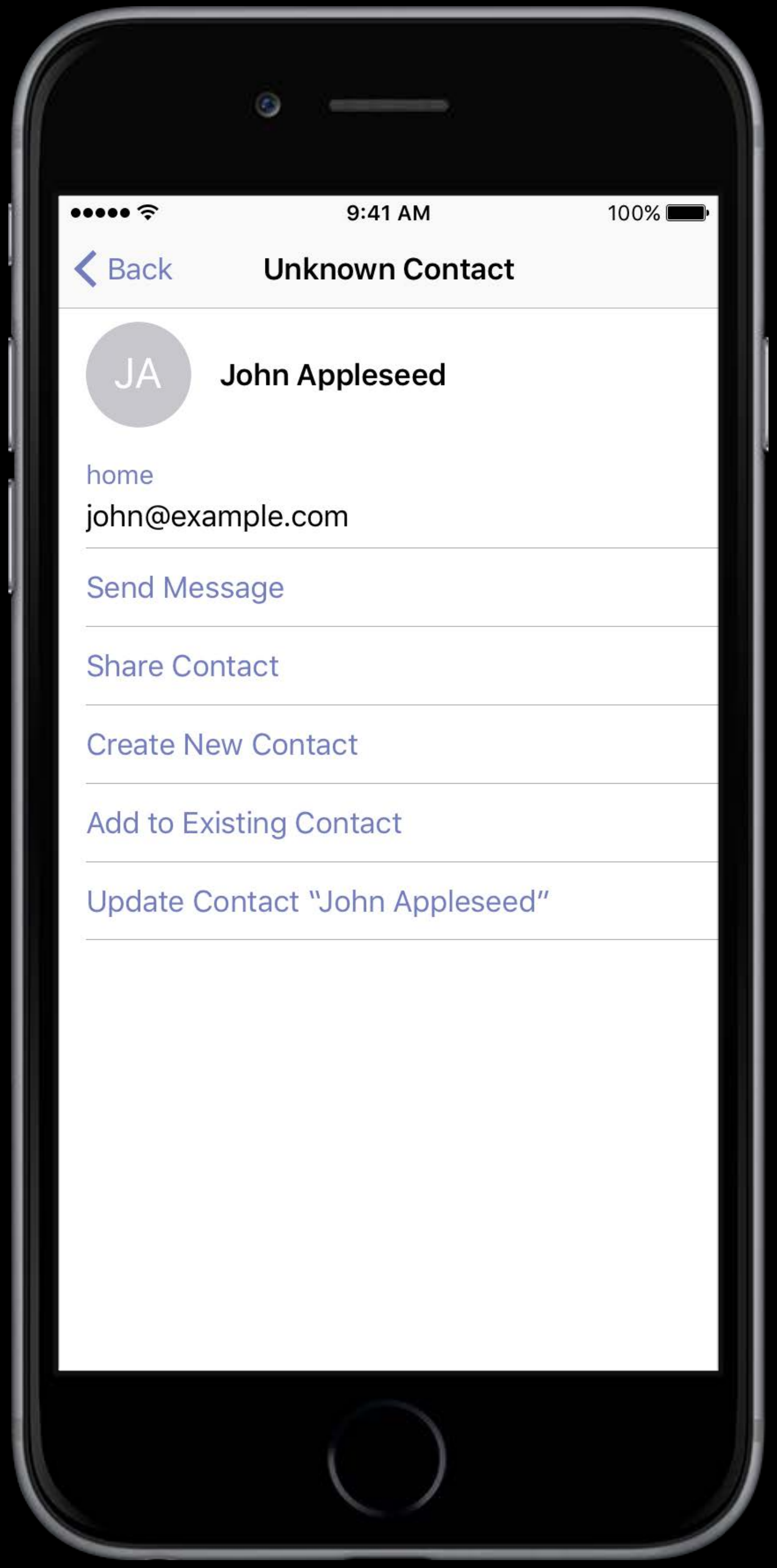


Viewing Contacts

CNContactViewController

Use appropriate creation method

- `viewControllerForContact:`
- `viewControllerForNewContact:`
- `viewControllerForUnknownContact:`

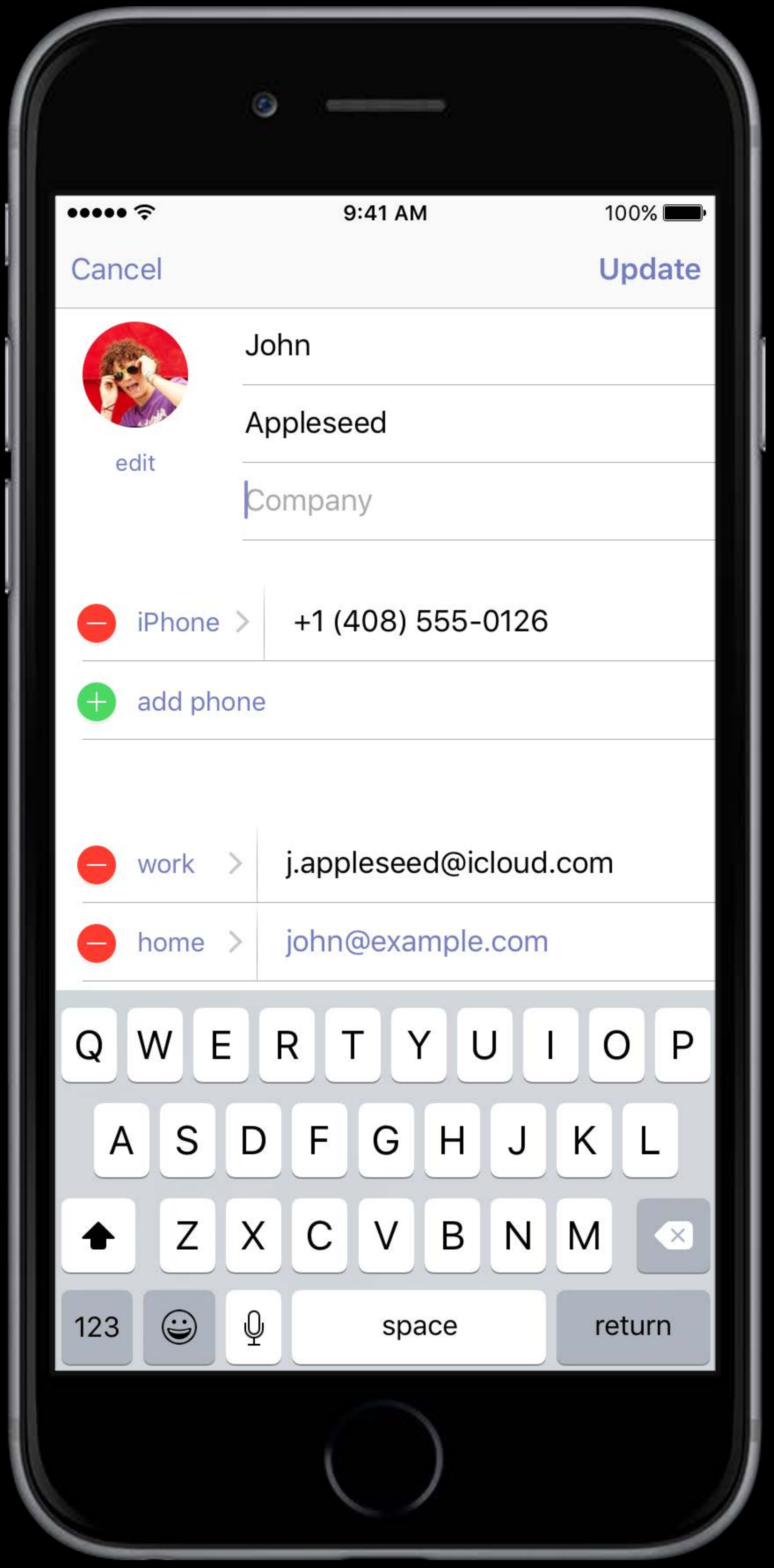


Viewing Contacts

CNContactViewController

Use appropriate creation method

- `viewControllerForContact:`
- `viewControllerForNewContact:`
- `viewControllerForUnknownContact:`



Viewing Contacts

CNContactViewController

Use appropriate creation method

- `viewControllerForContact:`
- `viewControllerForNewContact:`
- `viewControllerForUnknownContact:`

Always out of process

Viewing Contacts

CNContactViewController

Use appropriate creation method

- `viewControllerForContact:`
- `viewControllerForNewContact:`
- `viewControllerForUnknownContact:`

Always out of process

Contact must be fetched with `descriptorForRequiredKeys`

Viewing Contacts

Example

```
let contact = try contactStore.unifiedContactWithIdentifier(identifier,
keysToFetch: [CNContactViewController.descriptorForRequiredKeys])
```

Viewing Contacts

Example

```
let contact = try contactStore.unifiedContactWithIdentifier(identifier,
keysToFetch: [CNContactViewController.descriptorForRequiredKeys])
```

```
let viewController = CNContactViewController(forContact: contact)
```

Viewing Contacts

Example

```
let contact = try contactStore.unifiedContactWithIdentifier(identifier,
keysToFetch: [CNContactViewController.descriptorForRequiredKeys])

let viewController = CNContactViewController(forContact: contact)

viewController.contactStore = self.contactStore
viewController.delegate = self
```

Viewing Contacts

Example

```
let contact = try contactStore.unifiedContactWithIdentifier(identifier,
keysToFetch: [CNContactViewController.descriptorForRequiredKeys])

let viewController = CNContactViewController(forContact: contact)

viewController.contactStore = self.contactStore
viewController.delegate = self

self.pushViewController(viewController)
```

Viewing Contacts

Example

```
let contact = try contactStore.unifiedContactWithIdentifier(identifier,
keysToFetch: [CNContactViewController.descriptorForRequiredKeys])

let viewController = CNContactViewController(forContact: contact)

viewController.contactStore = self.contactStore
viewController.delegate = self

self.pushViewController(viewController)

func contactViewController(vc, didCompleteWithContact: contact) {
    // do something with the modified contact
}
```

Demo

Picking and Viewing Contacts

Meow

Summary

A new modern Contacts API

Common across all platforms

Adopt now!

More Information

Documentation

Contacts Framework Reference
ContactsUI Framework Reference
http://developer.apple.com/library

Technical Support

Apple Developer Forums
http://developer.apple.com/forums

General Inquiries

Paul Marcos, App Frameworks Evangelist
pmarcos@apple.com

Labs

Contacts, Calendar and Reminders Lab	Frameworks Lab A	Thursday 4:30PM
Contacts, Calendar and Reminders Lab	Frameworks Lab A	Friday 9:00AM

WWDC15