Increase Usage of Your App with Proactive Suggestions

Session 240

Daniel Gross Siri
Sofiane Toudji Software Engineering
You want more users.
You want more users.

Apple wants to offer your app at the right moment.
Siri App Suggestions
Spotlight Search
Handoff
Now Playing

The Power Of Design
TED Radio Hour — May 19, 2016 at

Press home to open
Contextual Siri Reminders
"Remind me about this when I get home"

OK, I'll remind you when you arrive at home:

Reminder

Surreal Desert Landscapes
Painted on a Canvas Made of Sky
Arriving: Home

Remove

Contextual Siri Reminders
QuickType
Multitasking
Proactive Suggestions

Overview
Proactive Suggestions
Overview

To learn when to promote you, the OS needs to know more about your data or app.
Proactive Suggestions

Overview

To learn when to promote you, the OS needs to know more about your data or app.
A few simple APIs deeply integrate your app into iOS and macOS.
Proactive Suggestions

Overview

To learn when to promote you, the OS needs to know more about your data or app.

A few simple APIs deeply integrate your app into iOS and macOS.

Deeper integration with Siri “for free.”
Proactive Suggestions

Agenda
Proactive Suggestions

Agenda

NSUserActivity and schema.org
Proactive Suggestions

Agenda

NSUserActivity and schema.org

Location Suggestions
Proactive Suggestions

Agenda

NSUserActivity and schema.org

Location Suggestions

Media App Suggestions
Proactive Suggestions

Agenda

NSUserActivity and schema.org

Location Suggestions

Media App Suggestions

Summary
Proactive Suggestions

Agenda

NSUserActivity and schema.org

Location Suggestions

Media App Suggestions

Summary
1. Handoff
2. Spotlight Search
3. Contextual Siri Reminders
4. Location Suggestions
5. Contextual Siri Requests
6. Contact Interactions
NSUserActivity
NSUserActivity
Handoff

Swift Playgrounds
Learn serious code on your iPad.
In a seriously fun way.
NSUserActivity
Handoff

Swift Playgrounds
Learn serious code on your iPad.
In a seriously fun way.
Spotlight Search
Location Suggestions

Embarcadero Station
298 Market St, San Francisco, CA

Montgomery St. Station
598 Market St, San Francisco, CA

San Francisco International Airport
San Francisco, CA

Powell St. Station
899 Market St, San Francisco, CA

Union Square
San Francisco, CA

Yank Sing from Yelp
101 Spear St San Francisco CA 94105
“Hey Siri take me there”

Getting directions...

Contextual Siri Requests
How does it work?
Capturing app state with NSUserActivity
NSUserActivity
Capturing app state
NSUserActivity
Capturing app state
NSUserActivity
Capturing app state
NSUserActivity
Capturing app state

Looking for lunch?

Azume Japanese Cuisine

NSUserActivity

NSUserActivity
NSUserActivity
Capturing app state
NSUserActivity
NSUserActivity

Payload

Metadata

Advertised for Handoff
NSUserActivity

Payload

Metadata

Advertised for Handoff

Temporary Context
NSUserActivity

Payload

Metadata

Advertised for Handoff

Temporary Context

Spotlight on Device Index

Item
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Making the Most of Search APIs</td>
<td>Pacific Heights</td>
<td>Thursday 1:00AM</td>
</tr>
<tr>
<td>Adopting Handoff on iOS and OS X</td>
<td>WWDC 2014</td>
<td></td>
</tr>
<tr>
<td>Introducing Search APIs</td>
<td>WWDC 2015</td>
<td></td>
</tr>
</tbody>
</table>
Creating Activities
Create user activity object

```swift
let activity = NSUserActivity(activityType: "com.example.view-location")
activity.title = "Yank Sing"
```
Creating Activities

Create user activity object

```swift
let activity = NSUserActivity(activityType: "com.example.view-location")
activity.title = "Yank Sing"
```

Enable capabilities

```swift
activity.isEligibleForHandoff = true
activity.isEligibleForSearch = true
activity.isEligibleForPublicIndexing = true
```
Creating Activities

Create user activity object

```swift
let activity = NSUserActivity(activityType: "com.example.view-location")
activity.title = "Yank Sing"
```

Enable capabilities

```swift
activity.isEligibleForHandoff = true
activity.isEligibleForSearch = true
activity.isEligibleForPublicIndexing = true
```

Specify userInfo

```swift
activity.userInfo = [
    "id": "yank-sing-san-francisco"
]
```
let attributes = CSSearchableItemAttributeSet(itemContentType: "com.example.location")

// Provide values for all relevant attributes
attributes.thumbnailURL = myThumbnailURL
activity.keywords = ["dim sum", "cantonese", "restaurant"]
activity.contentAttributeSet = attributes
Creating Activities

Provide metadata

```swift
let attributes = CSSearchableItemAttributeSet(itemContentType: "com.example.location")
// Provide values for all relevant attributes
attributes.thumbnailURL = myThumbnailURL
activity.keywords = ["dim sum", "cantonese", "restaurant"]
activity.contentAttributeSet = attributes
```

Restoring on the web

```swift
activity.webpageURL = "http://www.example.com/yank-sing-san-francisco"
```
Creating Activities

Provide metadata

```swift
let attributes = CSSearchableItemAttributeSet(itemContentType: "com.example.location")
// Provide values for all relevant attributes
attributes.thumbnailURL = myThumbnailURL
activity.keywords = ["dim sum", "cantonese", "restaurant"]
activity.contentAttributeSet = attributes
```

Restoring on the web

```swift
activity.webpageURL = "http://www.example.com/yank-sing-san-francisco"
activity.becomeCurrent()
```
Continuing Activities

UIApplicationDelegate

```swift
func application(_ UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: [AnyObject]? -> Void) -> Bool {
    if userActivity.activityType == "com.example.view-location" {
        // Restore state for userActivity and userInfo
    }
    return true
}
```
Continuing Activities

UIApplicationDelegate

```swift
func application(UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: [AnyObject]?> -> Void) -> Bool {

    if userActivity.activityType == "com.example.view-location" {
        // Restore state for userActivity and userInfo
    }

    return true
}
```
Continuing Activities

UIFontApplicationDelegate

```swift
func applicationWill(UIUIApplication, continueUserActivity userActivity: 
    NSUserActivity, restorationHandler: [AnyObject]? -> Void) -> Bool {

    if userActivity.activityType == "com.example.view-location" {

        // Restore state for userActivity and userInfo
    }

    return true
}
```
Location Suggestions

Contextual Siri Requests

Contact Interactions

1. Handoff
2. Spotlight Search
3. Contextual Siri Reminders
4. Location Suggestions
5. Contextual Siri Requests
6. Contact Interactions
Handoff
Spotlight Search
Contextual Siri Reminders
Location Suggestions
Contextual Siri Requests
Contact Interactions
Find a Place to Eat

1 Steps
Find a Place to Eat

*Switch App*

Text Friends

yelp®

3 Steps
Find a Place to Eat

*Switch App*

Copy Address

*Switch App*

Text Friends

Steps

5
Hey
Here’s a good one
How many developers does it take to change a light bulb?
None, that’s a hardware problem.

Anyway, I found a place to eat...

Let’s meet at
Yank Sing from Yelp
101 Spear St
San Francisco CA 94105
"Hey Siri take me there"

Getting directions...
From Yelp
From Your App
Location Suggestions

Background

Capture locations viewed by the user

MapKit and CoreSpotlight APIs

Integrate with Siri, Maps, Keyboards, Multitasking, and more
Location Suggestions
MapKit-based apps

Reuse the same user activity

```swift
let activity = NSUserActivity(activityType: "com.example.view-location")
```
Location Suggestions
MapKit-based apps

Reuse the same user activity

```
let activity = NSUserActivity(activityType: "com.example.view-location")
```

Set MKMapItem

```
activity.mapItem = myMapItem
```
Location Suggestions
Apps adopting app search

```swift
let attributes = CSSearchableItemAttributeSet(itemContentType: "location")
attributes.namedLocation = "Apple Inc."
attributes.subThoroughfare = "1"
attributes.thoroughfare = "Infinite Loop"
attributes.city = "Cupertino"
attributes.stateOrProvince = "CA"
attributes.country = "United States"
attributes.latitude = 37.33072
attributes.longitude = -122.029674
attributes.phoneNumbers = ["(800) 275-2273"]
attributes.supportsPhoneCall = true
attributes.supportsNavigation = true
```
Location Suggestions
Apps adopting app search

```swift
let attributes = CSSearchableItemAttributeSet(itemContentType: "location")
attributes.namedLocation = "Apple Inc."
attributes.subThoroughfare = "1"
attributes.thoroughfare = "Infinite Loop"
attributes.city = "Cupertino"
attributes.stateOrProvince = "CA"
attributes.country = "United States"
attributes.latitude = 37.33072
attributes.longitude = -122.029674
attributes.phoneNumbers = ["(800) 275-2273"]
attributes.supportsPhoneCall = true
attributes.supportsNavigation = true
```
let attributes = CSSearchableItemAttributeSet(itemContentType: "location")
attributes.namedLocation = "Apple Inc."
attributes.subThoroughfare = "1"
attributes.thoroughfare = "Infinite Loop"
attributes.city = "Cupertino"
attributes.stateOrProvince = "CA"
attributes.country = "United States"
attributes.latitude = 37.33072
attributes.longitude = -122.029674
attributes.phoneNumbers = ["(800) 275-2273"]
attributes.supportsPhoneCall = true
attributes.supportsNavigation = true
let attributes = CSSearchableItemAttributeSet(itemContentType: "location")
attributes.namedLocation = "Apple Inc."
attributes.subThoroughfare = "1"
attributes.thoroughfare = "Infinite Loop"
attributes.city = "Cupertino"
attributes.stateOrProvince = "CA"
attributes.country = "United States"
attributes.latitude = 37.33072
attributes.longitude = -122.029674
attributesphoneNumber = ["(800) 275-2273"]
attributes.supportsPhoneCall = true
attributes.supportsNavigation = true
let attributes = CSSearchableItemAttributeSet(itemContentType: "location")
attributes.namedLocation = "Apple Inc."
attributes.subThoroughfare = "1"
attributes.thoroughfare = "Infinite Loop"
attributes.city = "Cupertino"
attributes.stateOrProvince = "CA"
attributes.country = "United States"
attributes.latitude = 37.33072
attributes.longitude = -122.029674
attributes.phoneNumbers = ["(800) 275-2273"]
attributes.supportsPhoneCall = true
attributes.supportsNavigation = true
let attributes = CSSearchableItemAttributeSet(itemContentType: "location")
attributes.namedLocation = "Apple Inc."
attributes.subThoroughfare = "1"
attributes.thoroughfare = "Infinite Loop"
attributes.city = "Cupertino"
attributes.stateOrProvince = "CA"
attributes.country = "United States"
attributes.latitude = 37.33072
attributes.longitude = -122.029674
attributes.phoneNumbers = ["(800) 275-2273"]
attributes.supportsPhoneCall = true
attributes.supportsNavigation = true
Handoff
Spotlight Search
Contextual Siri Reminders
Location Suggestions
Contextual Siri Requests
Contact interactions
Handoff
Spotlight Search
Contextual Siri Reminders
Location Suggestions
Contextual Siri Requests
Contact Interactions
John Appleseed

iPhone
(408) 555-0621

work
j.appleseed@icloud.com

WhatsApp
j.appleseed
NSUserActivity + Intents

Introducing SiriKit

Extending Your Apps with SiriKit

Presidio

Wednesday 5:00PM

Presidio

Thursday 1:40PM
Contact Interactions
Donating interactions
Contact Interactions
Donating interactions

INIntent
Participants
Metadata
Contact Interactions
Donating interactions
Contact Interactions

Donating interactions
Contact Interactions

Donating interactions

Specify sender and recipient

```swift
let sender = INPerson(handle: "j.appleseed@icloud.com",
                       displayName: "John Appleseed",
                       contactIdentifier: nil)

let recipient = INPerson(handle: "janedaniels@icloud.com",
                          displayName: "Jane Daniels",
                          contactIdentifier: nil)
```
Create an intent

```swift
let intent = INSendMessageIntent(recipients: [recipient],
    content: nil
    serviceName: INMessageServiceNameWhatsApp
    sender: sender)
```
Create an intent

```
let intent = INSendMessageIntent(recipients: [recipient],
content: nil
serviceName: INMessageServiceNameWhatsApp
sender: sender)
```
Contact Interactions

Donating interactions

Create an intent

```swift
let intent = INSendMessageIntent(
  recipients: [recipient],
  content: nil
  serviceName: INMessageServiceNameWhatsApp,
  sender: sender)
```

Three communication intents

- `INSendMessageIntent`
- `INStartAudioCallIntent`
- `INStartVideoCallIntent`
Contact Interactions

Donating interactions

Create an intent

```swift
let intent = INSendMessageIntent(recipients: [recipient],
    content: nil
    serviceName: INMessageServiceNameWhatsApp
    sender: sender)
```

Three communication intents

- INSendMessageIntent
- INStartAudioCallIntent
- INStartVideoCallIntent
Contact Interactions

Donating interactions

Create an interaction

```swift
let interaction = INInteraction(intent: intent,
                                 response: INIntent.responseSuccess)

interaction.direction = INInteractionDirectionOutgoing
interaction.intentHandlingStatus = INIntentHandlingStatusDone
```
Create an interaction

```swift
let interaction = INInteraction(intent: intent,
                                  response: INIntent.responseSuccess)

interaction.direction = INInteractionDirectionOutgoing
interaction.intentHandlingStatus = INIntentHandlingStatusDone
```
Create an interaction

```swift
let interaction = INInteraction(intent: intent,
                                  response: INIntent.responseSuccess)

interaction.direction = INInteractionDirectionOutgoing
interaction.intentHandlingStatus = INIntentHandlingStatusDone
```

Donate interaction

```swift
interaction.donateInteraction(completion: { (error) in
  // Interaction donated
})
```
func application(UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: [AnyObject]? -> Void) -> Bool {

    if let intent = userActivity.interaction?.intent as? INSendMessageIntent {
        // Start communication with recipient
    }

    return true
}
func application(UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: [AnyObject]?) -> Void {
    if let intent = userActivity.interaction?.intent as? INSendMessageIntent {
        // Start communication with recipient
    }
    return true
}
Handoff
Spotlight Search
Contextual Siri Reminders
Location Suggestions
Contextual Siri Requests
Contact Interactions
Location Suggestions
Contextual Siri Requests
Contact Interactions
Spotlight Search
Contextual Siri Reminders
Handoff
Best Practices
Use needsSave for lazy payload updates

```swift
func updateActivityForEachKeystroke() {
    activity.userInfo = [
        "textContent": textfield.text
    ]
}
```
Best Practices

Lazy payload update

Use needsSave for lazy payload updates

```swift
func updateActivityForEachKeystroke() {
    activity.userInfo = [
        "textContent": textfield.text
    ]
}
```
Use needsSave for lazy payload updates

```swift
func updateActivityForEachKeystroke() {
    activity.userInfo = 
        ["textContent": textfield.text]
}

func updateActivityForEachKeystroke() {
    activity.needsSave = true
}

override func updateUserActivityState(_ activity: NSUserActivity) {
    activity.addUserInfoEntries(from: 
        ["textContent": textfield.text])
}
```
Best Practices

Scope of current activity

Keep a strong reference to the current activity

```swift
func updateActivity() {
    let activity = NSUserActivity(activityType: "com.example.my-activity")
    // setup user activity
    activity.becomeCurrent()
}

// activity is released
```
Best Practices

Scope of current activity

Keep a strong reference to the current activity

```swift
func updateActivity() {
    let activity = NSUserActivity(activityType: "com.example.my-activity")
    // setup user activity
    activity.becomeCurrent()
}

// activity is released
```
Best Practices

Scope of current activity

Keep a strong reference to the current activity

```swift
func updateActivity() {
    let activity = NSUserActivity(activityType: "com.example.my-activity")
    // setup user activity
    self.userActivity = activity
    activity.becomeCurrent()
}
```
Best Practices

Scope of current activity

Keep a strong reference to the current activity

```swift
func updateActivity() {
    let activity = NSUserActivity(activityType: "com.example.my-activity")
    // setup user activity
    self.userActivity = activity
    activity.becomeCurrent()
}
```
Best Practices

Transfer a small userInfo payload

Transfer a small payload in the userInfo dictionary

```swift
let imageData: Data // Downloaded image data from a web service
let photo = UIImage(data: imageData)

activity.userInfo = ["photoData": imageData]
```
Best Practices

Transfer a small userInfo payload

Transfer a small payload in the userInfo dictionary

```swift
let imageData: Data // Downloaded image data from a web service
let photo = UIImage(data: imageData)

activity.userInfo = ["photoData": imageData]
```
Best Practices

Transfer a small userInfo payload

Transfer a small payload in the userInfo dictionary

```swift
let imageData: Data // Downloaded image data from a web service
let photo = UIImage(data: imageData)

activity.userInfo = [
  "photoData": imageData
]

activity.userInfo = [
  "photoRemoteURL": imageRemoteURL
]
```
Best Practices

Unique activity types

Use reverse-DNS notation for your activity types

```swift
let activity = NSUserActivity(activityType: "my-activity")
```
Best Practices

Unique activity types

Use reverse-DNS notation for your activity types

let activity = NSUserActivity(activityType: "my-activity")
Best Practices
Unique activity types

Use reverse-DNS notation for your activity types

```swift
let activity = NSUserActivity(activityType: "my-activity")
```

```swift
let activity = NSUserActivity(activityType: "com.example.view-location")
let activity = NSUserActivity(activityType: "com.example.search-location")
```
Demo
Proactive Toolbox app
NSUserActivity
Native apps

schema.org
Web apps
schema.org
Background

Open web markup vocabulary standard
Structured metadata on web pages
500+ schemas representing various concepts
Some of the benefits of NSUserActivity for the web
schema.org

Structure

- **Thing**
  - **Organization**
    - **LocalBusiness**
      - ... (Nested elements)
    - ... (Additional elements)
    - **Restaurant**
  - ... (Additional elements)
  - **Rating**
    - ... (Additional elements)
  - **AggregateRating**
schema.org
Structure

- **Organization**
  - **LocalBusiness**
  - **Restaurant**
  - **Rating**
  - **AggregateRating**
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Defined in Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Text</td>
<td>Thing</td>
</tr>
<tr>
<td>aggregateRating</td>
<td>AggregateRating</td>
<td>Organization</td>
</tr>
<tr>
<td>address</td>
<td>PostalAddress</td>
<td>LocalBusiness</td>
</tr>
<tr>
<td>telephone</td>
<td>Text</td>
<td>LocalBusiness</td>
</tr>
<tr>
<td>openingHours</td>
<td>Text</td>
<td>FoodEstablishment</td>
</tr>
<tr>
<td>priceRange</td>
<td>Text</td>
<td>FoodEstablishment</td>
</tr>
<tr>
<td>acceptsReservations</td>
<td>Boolean</td>
<td>Restaurant</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Web apps promoting locations

<script type="application/ld+json">
{
    "@context": "http://schema.org",
    "@type": "Restaurant",
    "name": "Yank Sing",
    "telephone": "(415) 781-1111",
    "url": "http://www.yanksing.com",
    "address": {
        "@type": "PostalAddress",
        "streetAddress": "101 Spear St",
        "addressLocality": "San Francisco",
        "postalCode": "94105",
        "addressRegion": "CA"
    },
    "aggregateRating": {
        "@type": "AggregateRating",
        "ratingValue": "3.5",
        "reviewCount": "2022"
    }
}
</script>
Yank Sing
3.5 stars - based on 2022 reviews
101 Spear St, San Francisco, CA 94105
(408) 714-1489
<a href="http://www.yanksing.com">www.yanksing.com</a>
Yank Sing
3.5 stars - based on 2022 reviews
101 Spear St, San Francisco, CA 94105
(408) 714-1489
<a href="http://www.yanksing.com">www.yanksing.com</a>
Yank Sing
3.5 stars - based on 2022 reviews
101 Spear St, San Francisco, CA 94105
(408) 714-1489
<a href="http://www.yanksing.com">www.yanksing.com</a>

<script type="application/ld+json">
{
   "@context": "http://schema.org",
   "@type": "Restaurant",
   "name": "Yank Sing",
   "telephone": "(415) 781-1111",
   "url": "http://www.yanksing.com",
   "address": {
      "@type": "PostalAddress",
      "streetAddress": "101 Spear St",
      "addressLocality": "San Francisco",
      "postalCode": "94105",
      "addressRegion": "CA"
   },
   "aggregateRating": {
      "@type": "AggregateRating",
      "ratingValue": "3.5",
      "reviewCount": "2022"
   }
}
</script>
<div>
  Yank Sing<br />
  3.5 stars - based on 2022 reviews<br />
  101 Spear St, San Francisco, CA 94105<br />
  (408) 714-1489<br />
  <a href="http://www.yanksing.com">www.yanksing.com</a>
</div>

<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Restaurant",
  "name": "Yank Sing",
  "telephone": "(415) 781-1111",
  "url": "http://www.yanksing.com",
  "address": {
    "@type": "PostalAddress",
    "streetAddress": "101 Spear St",
    "addressLocality": "San Francisco",
    "postalCode": "94105",
    "addressRegion": "CA"
  },
  "aggregateRating": {
    "@type": "AggregateRating",
    "ratingValue": "3.5",
    "reviewCount": "2022"
  }
}
</script>
<div itemscope itemtype="http://schema.org/Restaurant">
  <span itemprop="name">Yank Sing</span>
  <div itemprop="aggregateRating" itemscope itemtype="http://schema.org/AggregateRating">
    <span itemprop="ratingValue">3.5</span> stars -
    based on <span itemprop="reviewCount">2022</span> reviews
  </div>
  <div itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
    <span itemprop="streetAddress">101 Spear St</span>,
    <span itemprop="addressLocality">San Francisco</span>,
    <span itemprop="addressRegion">CA</span> <span itemprop="postalCode">94105</span>
  </div>
  <span itemprop="telephone">(415) 781-1111</span>
  <a itemprop="url" href="http://www.yanksing.com">www.yanksing.com</a>
</div>
schema.org
Web apps promoting locations

Safari relies on location-related schemas for location suggestions
Some of the benefits of NSUserActivity for the web
Support for JSON-LD and Microdata
schema.org

Supported schemas

Any schemas with:

• PostalAddress
• GeoCoordinates
• A telephone property
Summary

NSUserActivity
Native apps

schema.org
Web apps
Summary

NSUserActivity
Native apps
Promote locations
Contact interactions
Spotlight Search and Handoff

schema.org
Web apps
Summary

**NSUserActivity**
Native apps

- Promote locations
- Contact interactions
- Spotlight Search and Handoff

**schema.org**
Web apps

- Promote locations on the web
NSUserActivity and schema.org

Location Suggestions

Media App Suggestions

Summary
1. Location Suggestions in QuickType
2. Routing Apps
1 Location Suggestions in QuickType

2 Routing Apps
<table>
<thead>
<tr>
<th>Location</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embarcadero Station</td>
<td>298 Market St, San Francisco, CA</td>
</tr>
<tr>
<td>Montgomery St. Station</td>
<td>598 Market St, San Francisco, CA</td>
</tr>
<tr>
<td>San Francisco International Airport</td>
<td>San Francisco, CA</td>
</tr>
<tr>
<td>Powell St. Station</td>
<td>899 Market St, San Francisco, CA</td>
</tr>
<tr>
<td>Union Square</td>
<td>San Francisco, CA</td>
</tr>
</tbody>
</table>

Yank Sing from Yelp
101 Spear St San Francisco CA 94105
Yank Sing from Yelp
101 Spear St San Francisco CA 94105
Locations Suggestions
QuickType

Hint the semantic intent for eligible text containers
Get proactive suggestions in your app
Benefit from enhanced autocorrection
New UITextInputTraits `textContentType` API

```swift
let textField = UITextField()
textField.textContentType = UITextContentTypeFullStreetAddress
```
**UITextInputTraits**

Text content type

<table>
<thead>
<tr>
<th>UITextContentTypeName</th>
<th>UITextContentTypeFullStreetAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>UITextContentTypeGivenName</td>
<td>UITextContentTypeAddressCityAndState</td>
</tr>
<tr>
<td>UITextContentTypeFamilyName</td>
<td>UITextContentTypeTelephoneNumber</td>
</tr>
<tr>
<td>UITextContentTypeLocation</td>
<td>UITextContentTypeEmailAddress</td>
</tr>
</tbody>
</table>
# UITextInputTraits

**Text content type**

<table>
<thead>
<tr>
<th>UITextContentTypeName</th>
<th>UITextContentTypeFullStreetAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>UITextContentTypeGivenName</td>
<td>UITextContentTypeAddressCityAndState</td>
</tr>
<tr>
<td>UITextContentTypeFamilyName</td>
<td>UITextContentTypeTelephoneNumber</td>
</tr>
<tr>
<td>UITextContentTypeLocation</td>
<td>UITextContentTypeEmailAddress</td>
</tr>
</tbody>
</table>
UI Text Input Traits

Text content type

<table>
<thead>
<tr>
<th>UITextContentTypeName</th>
<th>UITextContentTypeFullStreetAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>UITextContentTypeGivenName</td>
<td>UITextContentTypeAddressCityAndState</td>
</tr>
<tr>
<td>UITextContentTypeFamilyName</td>
<td>UITextContentTypeTelephoneNumber</td>
</tr>
<tr>
<td>UITextContentTypeLocation</td>
<td>UITextContentTypeEmailAddress</td>
</tr>
</tbody>
</table>
# UITextInputTraits

**Text content type**

<table>
<thead>
<tr>
<th>UITextContentTypeName</th>
<th>UITextContentTypeFullStreetAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>UITextContentTypeGivenName</td>
<td>UITextContentTypeAddressCityAndState</td>
</tr>
<tr>
<td>UITextContentTypeFamilyName</td>
<td>UITextContentTypeTelephoneNumber</td>
</tr>
<tr>
<td>UITextContentTypeLocation</td>
<td>UITextContentTypeEmailAddress</td>
</tr>
</tbody>
</table>
UITextContentTypeFullStreetAddress

UITextContentTypeCityAndState
Location Suggestions
Granularity

UITextContentTypeFullStreetAddress
Location Suggestions

Granularity

UITextContentTypeFullStreetAddress

Yank Sing from Yelp
101 Spear St San Francisco CA 94105
Location Suggestions

Granularity

UITextContentTypeFullStreetAddress

101 Spear St San Francisco CA 94105

UITextContentTypeAddressCityAndState

San Francisco, CA
1. Location Suggestions in QuickType
2. Routing Apps
Location Suggestions in QuickType

Routing Apps
From this Business

Explore the Menu
Peking Duck, Sesame Ball, SF

More Info
Hours, Website, Attire, Noise

Get Directions to "Yank Sing" in Uber
Recently viewed in Yelp
Consuming Locations
For routing apps

Promote your app in Multitasking
Register as a routing app
Handle launch with MapKit’s MKDirectionsRequest
iOS learns to suggest your app based on user engagement
MKDirectionsRequest

Overview

Configure your app to accept directions requests in Xcode
Overview

MKDirectionsRequest

Configure your app to accept directions requests in Xcode

Declare the map regions that your app supports
MKDirectionsRequest

Overview

Configure your app to accept directions requests in Xcode
Declare the map regions that your app supports
Process direction request URLs when they are sent to your app
func application(app: UIApplication, url: NSURL, options: [String : AnyObject]) -> Bool {
    if (MKDirectionsRequest.isDirectionsRequestURL(url)) {
        let directionsRequest = MKDirectionsRequest(contentsOfURL: url)
        // Handle routing request
        guard
            let coordinate = directionsRequest.destination?.placemark.location?.coordinate,
            let addressDictionary = directionsRequest.destination?.placemark.addressDictionary
        where !CLLocationCoordinate2DIsValid(coordinate)
        else { return true }
        let geocoder = CLGeocoder()
        geocoder.geocodeAddressDictionary(addressDictionary, completionHandler: { (place, err) in
            // Handle launch
        })
    } else { return true }
    return true
}
func application(app: UIApplication, url: NSURL, options: [String : AnyObject]) -> Bool {
    if MKDirectionsRequest.isDirectionsRequestURL(url) {
        let directionsRequest = MKDirectionsRequest(contentsOfURL: url)
        // Handle routing request
        guard
            let coordinate = directionsRequest.destination?.placemark.location?.coordinate,
            let addressDictionary = directionsRequest.destination?.placemark.addressDictionary
        where !CLLocationCoordinate2DIsValid(coordinate)
        else { return true }
        let geocoder = CLGeocoder()
        geocoder.geocodeAddressDictionary(addressDictionary, completionHandler: { (place, err) in
            // Handle launch
        })
    }
    return true
}
// Handle launch with MKDirectionsRequest

func application(app: UIApplication,
                 url: NSURL,
                 options: [String : AnyObject]) -> Bool {

    if (MKDirectionsRequest.isDirectionsRequestURL(url)) {
        let directionsRequest = MKDirectionsRequest(contentsOfURL: url)

        // Handle routing request
        guard
            let coordinate = directionsRequest.destination?.placemark.location?.coordinate,
            let addressDictionary = directionsRequest.destination?.placemark.addressDictionary
        where !CLLocationCoordinate2DIsValid(coordinate)
        else { return true }

        let geocoder = CLGeocoder()
        geocoder.geocodeAddressDictionary(addressDictionary, completionHandler: { (place, err) in
            // Handle launch
            })
    }

    return true
}
// Handle launch with MKDirectionsRequest

func application(app: UIApplication,
    url: NSURL,
    options: [String : AnyObject]) -> Bool {
    if (MKDirectionsRequest.isDirectionsRequestURL(url)) {
        let directionsRequest = MKDirectionsRequest(contentsOfURL: url)
        // Handle routing request
        guard
            let coordinate = directionsRequest.destination?.placemark.location?.coordinate,
            let addressDictionary = directionsRequest.destination?.placemark.addressDictionary
        where !CLLocationCoordinate2DIsValid(coordinate)
        else { return true }
        let geocoder = CLGeocoder()
        geocoder.geocodeAddressDictionary(addressDictionary, completionHandler: { (place, err) in
            // Handle launch
        })
    } else { return true }
    return true
}
Location Suggestions in QuickType

Routing Apps
Location Suggestions in QuickType
Routing Apps
NSUserActivity and schema.org

Location Suggestions

Media App Suggestions

Summary
Promoting Your Media App

Overview
Promoting Your Media App

Overview

iOS promotes the app a user is likely to use based on behavior
Promoting Your Media App

Overview

iOS promotes the app a user is likely to use based on behavior.
Suggestions are offered in Spotlight and Today View.
Promoting Your Media App

Overview

iOS promotes the app a user is likely to use based on behavior.

Suggestions are offered in Spotlight and Today View.

If they follow a particular trigger, suggestions may be elevated to the lock screen.
Promoting Your Media App

Overview

iOS promotes the app a user is likely to use based on behavior

Suggestions are offered in Spotlight and Today View

If they follow a particular trigger, suggestions may be elevated to the lock screen

For example:

• After plugging in headphones or a BT device (Podcasts, Music)
• After getting in a car (Maps)
• After arriving at home or work (Agenda app)
Spotlight
Spotlight
Lockscreen
With a simple API, you can build a far more engaging lock screen experience.
Before
The Power Of Design
TED Radio Hour — May 19, 2016 at

After
10X
Increase in conversion
How?
MPPlayableContentManager
struct MediaItem {
    var title: String
    var artist: String
    var album: String
    var duration: Double
    var albumArtwork: UIImage
    var albumArtworkSize: CGSize
}

import MediaPlayer

class MyPlayer: MPPlaybleContentDelegate, NSObject {
    override init() {
        super.init()
        MPPlaybleContentManager.shared().delegate = self
    }
}

import MediaPlayer

class MyPlayer: MPPlayabeContentDelegate, NSObject {
    override init() {
        super.init()
        MPPlayabeContentManager.shared().delegate = self
    }
}

import MediaPlayer

class MyPlayer: MPPlaybleContentDelegate, NSObject {
    override init() {
        super.init()
        MPPlayableContentManager.shared().delegate = self
    }
}

import MediaPlayer

class MyPlayer: MPPlayableContentDelegate, NSObject {
    override init() {
        super.init()
        MPPlayableContentManager.shared().delegate = self
    }
}
func playableContentManager(contentManager: MPPlayableContentManager,  
initializePlaybackQueueWithContentItems contentItems: [AnyObject]?,  
completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }

    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
func playableContentManager(contentManager: MPPlaybleContentManager,
    initializePlaybackQueueWithContentItems contentItems: [AnyObject]?,
    completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }

    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
func playableContentManager(contentManager: MPPlayContentManager,
    initializePlaybackQueueWithContentItems contentItems: [AnyObject]?,
    completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }
    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }
    completionHandler(nil)
}
func playableContentManager(contentManager: MPPlayabeContentManager, 
      initializePlaybackQueueWithContentItems contentItems: [AnyObject]?, 
      completionHandler: (NSError?) -> Void) {

  guard let mediaItem = findMediaItemToPlay() else {
    // Handle error, call completion handler with error and return
  }

  populateNowPlayingInfo(mediaItem: mediaItem)

  guard let status = preparePlayback(mediaItem: mediaItem) else {
    // Handle error, call completion handler with error and return
  }

  // Do other things that are required for success

  completionHandler(nil)
}
func playableContentManager(contentManager: MPPlayableContentManager, 
    initializePlaybackQueueWithContentItems contentItems: [AnyObject]?, 
    completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }

    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
func playableContentManager(contentManager: MPPlayableContentManager,
    initializePlaybackQueueWithContentItems contentItems: [AnyObject]?,
    completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }

    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize,
                                           requestHandler: { (size) -> UIImage in
                                                               return image
                                           })

        return image
    }

    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title,
        MPMediaItemPropertyArtist: item.artist,
        MPMediaItemPropertyAlbumTitle: item.album,
        MPMediaItemPropertyPlaybackDuration: item.duration,
        MPMediaItemPropertyArtwork: albumArt
    ]

    infoCenter.nowPlayingInfo = nowPlayingInfo
}
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize,
            requestHandler: { (size) -> UIImage in
                return image
            })
    
        return image
    }

    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title,
        MPMediaItemPropertyArtist: item.artist,
        MPMediaItemPropertyAlbumTitle: item.album,
        MPMediaItemPropertyPlaybackDuration: item.duration,
        MPMediaItemPropertyArtwork: albumArt
    ]

    infoCenter.nowPlayingInfo = nowPlayingInfo
}
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize,
                                          requestHandler: { (size) -> UIImage in
                                              return image
                                          })

        return image
    }

    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title,
        MPMediaItemPropertyArtist: item.artist,
        MPMediaItemPropertyAlbumTitle: item.album,
        MPMediaItemPropertyPlaybackDuration: item.duration,
        MPMediaItemPropertyArtwork: albumArt
    ]

    infoCenter.nowPlayingInfo = nowPlayingInfo
}
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize,
            requestHandler: { (size) -> UIImage in
                return image
            })
        return image
    }
    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title,
        MPMediaItemPropertyArtist: item.artist,
        MPMediaItemPropertyAlbumTitle: item.album,
        MPMediaItemPropertyPlaybackDuration: item.duration,
        MPMediaItemPropertyArtwork: albumArt
    ]
    infoCenter.nowPlayingInfo = nowPlayingInfo
}
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize,
        requestHandler: { (size) -> UIImage in
            return image
        })
        return image
    }

    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title,
        MPMediaItemPropertyArtist: item.artist,
        MPMediaItemPropertyAlbumTitle: item.album,
        MPMediaItemPropertyPlaybackDuration: item.duration,
        MPMediaItemPropertyArtwork: albumArt
    ]

    infoCenter.nowPlayingInfo = nowPlayingInfo
}
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize,
                                           requestHandler: { (size) -> UIImage in
                                               return image
                                           })
        return image
    }

    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title,
        MPMediaItemPropertyArtist: item.artist,
        MPMediaItemPropertyAlbumTitle: item.album,
        MPMediaItemPropertyPlaybackDuration: item.duration,
        MPMediaItemPropertyArtwork: albumArt
    ]

    infoCenter.nowPlayingInfo = nowPlayingInfo
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize,
            requestHandler: { (size) -> UIImage in
                return image
            })
        return image
    }

    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title,
        MPMediaItemPropertyArtist: item.artist,
        MPMediaItemPropertyAlbumTitle: item.album,
        MPMediaItemPropertyPlaybackDuration: item.duration,
        MPMediaItemPropertyArtwork: albumArt
    ]

    infoCenter.nowPlayingInfo = nowPlayingInfo
}
func populateNowPlayingInfo(item: MediaItem) {
    let infoCenter = MPNowPlayingInfoCenter.defaultCenter()
    if let image = item.albumArtwork ?? UIImage(named: "EmptyAlbum") {
        let albumArt = MPMediaItemArtwork(boundsSize: item.albumArtworkSize, 
                                        requestHandler: { (size) -> UIImage in
                                            return image
                                        })
        return image
    }

    let nowPlayingInfo = [
        MPMediaItemPropertyTitle: item.title, 
        MPMediaItemPropertyArtist: item.artist, 
        MPMediaItemPropertyAlbumTitle: item.album, 
        MPMediaItemPropertyPlaybackDuration: item.duration, 
        MPMediaItemPropertyArtwork: albumArt
    ]

    infoCenter.nowPlayingInfo = nowPlayingInfo
}
func playableContentManager(contentManager: MPPlayableContentManager, initializePlaybackQueueWithContentItems contentItems: [AnyObject]?, completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }

    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
func playableContentManager(contentManager: MPPlaybleContentManager,
    initializePlaybackQueueWithContentItems contentItems: [AnyObject]?,
    completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }

    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
func playableContentManager(contentManager: MPPlaybleContentManager,
    initializePlaybackQueueWithContentItems contentItems: [AnyObject]?,
    completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }
    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
func playableContentManager(contentManager: MPPlayableContentManager, 
    initializePlaybackQueueWithContentItems contentItems: [AnyObject]?, 
    completionHandler: (NSError?) -> Void) {

    guard let mediaItem = findMediaItemToPlay() else {
        // Handle error, call completion handler with error and return
    }

    populateNowPlayingInfo(mediaItem: mediaItem)

    guard let status = preparePlayback(mediaItem: mediaItem) else {
        // Handle error, call completion handler with error and return
    }

    // Do other things that are required for success

    completionHandler(nil)
}
Promoting Your Media App

Overview

iOS promotes the app a user is likely to use based on behavior. Suggestions are offered in Spotlight and Today View. If they follow a particular trigger, suggestions may be elevated to the lock screen. For example:

- After plugging in headphones or a BT device (Podcasts, Music)
- After getting in a car (Maps)
- After arriving at home or work (Agenda app)

With a simple API, you can build a far more engaging lock screen experience.
NSUserActivity and schema.org

Location Suggestions

Media App Suggestions

Summary
Proactive Suggestions

Summary
Proactive Suggestions

Summary

Simple APIs that deeply integrate your app to the OS
Proactive Suggestions

Summary

Simple APIs that deeply integrate your app to the OS

• NSUserActivity—The “eyes” of the OS
Proactive Suggestions

Summary

Simple APIs that deeply integrate your app to the OS

• NSUserActivity—The “eyes” of the OS
• schema.org—Like NSUserActivity, but for the web
Proactive Suggestions

Summary

Simple APIs that deeply integrate your app to the OS

• NSUserActivity—The “eyes” of the OS
• schema.org—Like NSUserActivity, but for the web
• MKDirectionsRequest and a new UIKit API—Build a seamless experience for handling locations
Proactive Suggestions

Summary

Simple APIs that deeply integrate your app to the OS

• NSUserActivity—The “eyes” of the OS
• schema.org—Like NSUserActivity, but for the web
• MKDirectionsRequest and a new UIKit API—Build a seamless experience for handling locations
• MPPlayableContentManager—Enhanced experience for media suggestions
Proactive Suggestions

Summary

Simple APIs that deeply integrate your app to the OS

- NSUserActivity—The “eyes” of the OS
- schema.org—Like NSUserActivity, but for the web
- MKDirectionsRequest and a new UIKit API—Build a seamless experience for handling locations
- MPPlayableContentManager—Enhanced experience for media suggestions

Easy to adopt, easy to test
More Information

## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing SiriKit</td>
<td>Presidio</td>
<td>Wednesday 5:00PM</td>
</tr>
<tr>
<td>Making the Most of Search APIs</td>
<td>Pacific Heights</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>Extending Your Apps with SiriKit</td>
<td>Presidio</td>
<td>Thursday 1:40PM</td>
</tr>
<tr>
<td>Adopting Handoff on iOS and OS X</td>
<td>WWDC 2014</td>
<td></td>
</tr>
<tr>
<td>Introducing Search APIs</td>
<td>WWDC 2015</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proactive Suggestions Lab</td>
<td>Frameworks Lab A</td>
<td>Friday 3:00PM</td>
</tr>
</tbody>
</table>
The more the system knows about your app,
The more the system knows about your app, the more opportunity it will have to promote it.