Making Apps Accessible

Making Apps Accessible

- Standard UIKit controls and views have accessibility built in, require no configuration
  - You may want to add hints in rare cases
- NSObject adopts UIAccessibility protocol
- Custom views require work
  - UIAccessiblityElement class

Custom Views

```
@implementation MyCustomView
- (BOOL)isAccessibilityElement
{
    return YES;
}
```

OR

```
@implementation MyViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.myCustomView.isAccessibilityElement = YES;
}
```

Custom Individual Views

- UIAccessibility Informal Protocol

Determining Accessibility

`isAccessibilityElement` property

Configuring an Accessibility Element

`accessibilityLabel` property
`accessibilityHint` property
`accessibilityValue` property
`accessibilityTraits` property
`accessibilityFrame` property
`accessibilityLanguage` property

Custom Container Views

- Make contained views accessible
- Make container view NOT accessible
- UIAccessibilityContainer Protocol
    - `– accessibilityElementCount`
    - `– accessibilityElementAtIndex:`
    - `– indexOfAccessibilityElement:`

TableViewControllers

- Your application should always set the `accessibilityLabel` property for new table cell elements

UIAccessibilityElement

Creating an Accessibility Element

– `initWithAccessibilityContainer:`

Accessing the Containing View

`accessibilityContainer` property

Determining Accessibility

`isAccessibilityElement` property

Accessing the Attributes of an Accessibility Element

`accessibilityLabel` property
`accessibilityHint` property
`accessibilityValue` property
`accessibilityFrame` property
`accessibilityTraits` property

Hints

- Tell the user what will happen if they activate the selected control
- Use punctuation (e.g., begin with capital letter and end with period) for voice inflection
- Don't include name or type of control or view
- Use localized strings