View Concepts

- SDK provide many types of Views to show your content
- At run-time Views are organized as a tree
- Use Interface Builder to design your UI and connect it to code
- Geometry of Views are determined by constraints
Finding the Right View

A label displays static text.

View Concepts

- SDK provides many types of Views to show your content
- At run-time Views are organized as a tree
- Use Interface Builder to design your UI and connect it to code
- Geometry of Views is determined by constraints

Demo: Hacking Calendar

```assembly
(llb) po [[UITextView keyWindow] recursiveDescription]
<UITextView
    | <UILayoutContainerView
    |    | <UINavigationTransitionView
    |    |    | <UIViewControllerWrapperView
    |    |    |    | <EKCalendarItemEditorTableView: 0xf870200; baseClass = UITableView;
    |    |    |    |    | <UITableViewWrapperView
    |    |    |    |    |    | <UITableViewCell: 0xb19ad90
    |    |    |    |    |    |    ...
    |    |    | <UINavigationBar
    |    |    |    | ...
    |    |    | <UINavigationItemView
    |    |    |    | <UILabel
    |    |    |    | <UINavigationButton
    |    |    |    |    ...
(llab) expr ((UIView *)0xb19ad90).backgroundColor = [UIColor redColor]
(llab) expr ((UIView *)0xf870200).backgroundColor = [UIColor greenColor]
```
View Concepts

✓ SDK provide many types of Views to show your content
✓ At run-time Views are organized as a tree
  • Use Interface Builder to design your UI and connect it to code
  • Geometry of Views are determined by constraints

Interface Builder

  • Graphical tool to layout user interfaces
  • Create the widget hierarchy
  • Set attributes of widgets
  • Set up connections between the widgets
  • Store these informations in nib files

The Anatomy of a xib File
The Source of a `.xib`

```xml
<archive
    type="com.apple.InterfaceBuilder3.CocoaTouch.XIB"
    version="8.00">
    !
    <data>
        !!
        <int
            key="IBDocument.SystemTarget">
            1280
        </int>
        !!
        <string
            key="IBDocument.SystemVersion">
            11C74
        </string>
        !!
        <string
            key="IBDocument.InterfaceBuilderVersion">
            1938
        </string>
        !!
        <string
            key="IBDocument.AppKitVersion">
            1138.23
        </string>
        !!
        <string
            key="IBDocument.HIToolboxVersion">
            567.00
        </string>
        !!
        <object
            class="NSMutableDictionary"
            key="IBDocument.PluginVersions">
            !
            <string
                key="NS.key.0">
                com.apple.InterfaceBuilder.IBCocoaTouchPlugin
            </string>
            !
            <string
                key="NS.object.0">
                933
            </string>
        </object>
        !!
        <array
            key="IBDocument.IntegratedClassDependencies">
            !!
            <string>
                IBUITextView
            </string>
            !!
            <string>
                IBUISwitch
            </string>
            !!
            <string>
                IBUIButton
            </string>
            !!
            <string>
                IBUIView
            </string>
            !!
            <string>
                IBUILabel
            </string>
            !!
            <string>
                IBProxyObject
            </string>
        </array>
        !!
        <array
            key="IBDocument.PluginDependencies">
            !!
            <string>
                com.apple.InterfaceBuilder.IBCocoaTouchPlugin
            </string>
        </array>
        !!
        <object
            class="NSMutableDictionary"
            key="IBDocument.Metadata">
            !
            <string
                key="NS.key.0">
                PluginDependencyRecalculationVersion
            </string>
            !!
            <integer
                key="NS.object.0" value="1"/>
        </object>
        !!
        <array
            class="NSMutableArray"
            key="IBDocument.RootObjects"
            id="1000">
            !!
            <object
                class="IBProxyObject"
                id="841351856">
                !!
                <string
                    key="IBProxiedObjectIdentifier">
                    IBFilesOwner
                </string>
                !!
                <string
                    key="targetRuntimeIdentifier">
                    IBCocoaTouchFramework
                </string>
            </object>
            !!
            <object
                class="IBProxyObject"
                id="371349661">
                !!
                <string
                    key="IBProxiedObjectIdentifier">
                    IBFirstResponder
                </string>
                !!
                <string
                    key="targetRuntimeIdentifier">
                    IBCocoaTouchFramework
                </string>
            </object>
            !!
            <object
                class="IBUIView"
                id="474857037">
                !!
                <reference
                    key="NSNextResponder" ref="474857037"/>
                !!
                <int
                    key="NSvFlags" value="292"/>
                !!
                <array
                    class="NSMutableArray"
                    key="NSSubviews">
                    !!
                    <object
                        class="IBUITextView"
                        id="694905917">
                        !!
                        <reference
                            key="NSNextResponder" ref="474857037"/>
                        !!
                        <int
                            key="NSvFlags" value="274"/>
                        !!
                        <string
                            key="NSFrame">{{20, 20}, {280, 385}}</string>
                        !!
                        <reference
                            key="NSSuperview" ref="474857037"/>
                    </object>
                </array>
            </object>
        </array>
    </data>
</archive>
```

Laying out the User Interface

- The library contains all UI Widgets
- Drag them to your view
- See instantly what your UI looks like
- Test your UI in the iPhone Simulator

Set Widget Attributes

Connecting Widgets and Code

- `IBActions`
  - Tags a method as a target for an interface action
- `IBOutlets`
- `IBOutletCollection`
  - Variables to populate with objects from a nib file
Connecting Widgets and Code

```objective-c
@interface iCalcView: UIView {
    ...
}
- (IBAction)addDigit:(id)sender;
- (IBAction)calculateResult:(id)sender;

//declared properties
//textfield
@property (nonatomic, weak) IBOutlet UITextField *textField;
//operators: + =
@property (nonatomic, weak) IBOutlet UIButton *operatorPlus;
@property (nonatomic, weak) IBOutlet UIButton *operatorEqual;
//digits
@property (nonatomic, weak) IBOutletCollection (UIButton) NSArray *digitButtons;
```

Interface Builder Demo

View Concepts

- SDK provides many types of Views to show your content
- At run-time Views are organized as a tree
- Use Interface Builder to design your UI and connect it to code
  - Geometry of Views are determined by constraints

Auto Layout

- Preferred layout management
- Allows you to create views that work both in portrait and landscape mode
- Available in iOS 6 and higher
- Spatial relationships expressed by constraints
Auto Layout Constraints

- Constraints are mathematical expressions
  - $\leq$, $\geq$
- Constraints have a priority level
- The runtime tries to solve the system of equations

Adding Constraints

View Concepts

- SDK provides many types of Views to show your content
- At run-time Views are organized as a tree
- Geometry of Views are determined by constraints
- Use Interface Builder to design your UI and connect it to code
Review

• A calendar app reminds user of a scheduled event
• A news reader app loads updates in background and updates the UI upon completion
• An app determines a path for an image that is shipped with the app itself
• Base class for objects that respond to UI events
• Base class for managing screen content

View Concepts

✓ SDK provide many types of Views to show your content
✓ At run-time Views are organized as a tree
✓ Geometry of Views are determined by constraints
✓ Use Interface Builder to design your UI and connect it to code

UIView

• Defines a rectangular area on the screen
• Two responsibilities
  • Render content
  • React to user input
  • Manage subviews
• Layout as view hierarchy

View Programming
View Hierarchies

Managing Views

- `addSubview`
- `insertSubview:atIndex:`
- `insertSubview:aboveSubview:`
- `removeFromSuperview`
- `bringSubviewToFront:`
- `sendSubviewToBack:`
- `exchangeSubviewAtIndex:withSubviewAtIndex:`
**View Coordinate System**

- Origin at top-left corner
- Coordinates use points as unit
  - iPhone: 320x480
  - iPhone 5: 320x568
  - iPad: 768x1024

**View Geometry**

- Frame
  - A rectangle with origin and size relative to the superview
- Bounds
  - A rectangle with origin (0,0) and the size of the view
- Center
  - The center point of the frame rectangle

**Frame vs. Bounds**

Frame (looking from outside)
- Origin: (140.0, 35.0)
- Size: (320.0, 480.0)

Bound (looking from the view itself)
- Origin: (0.0, 0.0)
- Size: (320.0, 480.0)

**Setting Values**

- Setting `frame`:
  - `bounds` matches the size
  - `center` is adjusted
- Setting `center`:
  - The origin of `frame` is set accordingly
- Setting the size of `bounds`
  - The size of `frame` is set accordingly
Content Modes & Scaling

### Content Mode

- **UIViewContentModeScaleToFfill**: Distorting
- **UIViewContentModeScaleAspectFit**: Nondistorting
- **UIViewContentModeScaleAspectFill**: Nondistorting

Creating Views

```swift
// Create a view with blue background
CGRect viewFrame = CGRectMake(160, 240, 150, 150);
UIView *blueView = [[UIView alloc] initWithFrame:viewFrame];
blueView.backgroundColor = [UIColor blueColor];

// add it to the main window
[window addSubview:blueView];
```

Subclassing UIView

```swift
// Only override drawRect: if you perform custom drawing.
// An empty implementation adversely affects performance during animation.
-(void)drawRect:(CGRect)rect {
    [view setNeedsDisplay];
}

// Overridden by subclasses to layout subviews when layoutIfNeeded is invoked.
// The default implementation of this method does nothing
-(void)layoutSubviews{
    [view setNeedsLayout];
}
```
### Reaction to Events

- Adjust properties of the view and its subviews
- Mark the view as needing a change in its layout
- Mark the view as needing to be redrawn
- Notify a controller that data has changed

### Scroll Views

- Container view
- Displays content larger than the app window
- Support for scrolling
- Support for zooming

---

**UIScrollView**

```objective-c
UIScrollView *scrollView = [[UIScrollView alloc] initWithFrame:window.bounds];
window.addSubview:scrollView;
// window retains its subview, thus we can release the view here
[scrollView release];
CGRect contentFrame = CGRectMake(0, 0, window.bounds.size.width*2, window.bounds.size.height*2);
scrollView.contentSize = contentFrame.size;
scrollView.contentOffset = window.center;
// add the content view
[scrollView addSubview:aView];
```

---

**UIScrollView: Zooming**

```objective-c
// Enable zooming
scrollView.minimumZoomScale = 0.5;
scrollView.maximumZoomScale = 2.5;
scrollView.delegate = self;

#pragma mark ScrollView delegate methods
- (UIView*)viewForZoomingInScrollView:(UIScrollView *)scrollView;
{
    return [window viewWithTag:1];
}
```
Summary

• Interface Builder
• UIView
• View hierarchies

• Reading Assignment:
  - View Programming Guide
  - Interface Builder User Guide
  - UIView Class Reference
  - Taking control of Auto Layout in Xcode 5

UIKitViewController

• Manages typically one screen
• Flushes the view on low-memory situations
• Resizes the view on orientation change
• Creates modal views on top of the current view