Lesson 5

PDF Slides Lesson 5

Lesson5_slides.pdf https://drive.google.com/file/d/15hR1a0wD3fmxl2-EplqaHMtWJT9EfpDW/view?usp=drivesdk

Basic Views

User Interface Components

- Mostly until now, we've been looking at Swift code in Playgrounds that we interact with in a text-based format.
- Apps, however, are based off rich, graphical user interfaces.
- Apple's UIKit provides us with the components we can use to make our own.

Views

- A view is a visual element that makes up our App.
  - We can create text, graphics, lines and more with them. Most apps contain screens which are made up of many
  - views, which create what is termed a view hierarchy.
  - Views can be nested within views – the concept of parent and child views.
  - Views need a Frame (size and position) first.

UIView

- The **UIView** class is **UIKit**'s foundational class. This is subclassed to make many components we can use to build our app's interface.

- Let’s go through some of the most common.

UILabel

- A **UILabel** allows us to display a piece of short text (in our app) to the user.
    - We can write code to change it – both the text and the look and feel.
    - The text doesn’t change - the user can’t change it or type over it.
    - It is possible to allow the user to change the text by using buttons or similar.

UIImageView

- A **UIImageView** allows us to display an image to the user.
    - What a **UILabel** is to text, a **UIImageView** is to pictures.
    - Images can be photos or drawings, still or animated.
    - In the same manner as a **UILabel**, we can change it using code, but the user can’t change it directly.

UITextView

- A **UITextView** allows a user to interact with multiple lines of text within the App.
    - Users can add, edit, view and change the text.
    - A use of this would be to write a message (of some sort)
    - It can have text already written when loaded.
    - Can also be used to display large amounts of text, **without** allowing the user to change or edit it.

UIScrollView

- A **UIScrollView** allows users to scroll.
    - The benefit of this is to add and explore content larger than the viewport (screen/window).
    - For example, could be used for large detailed images or complex maps.
    - Scrollbars (or indicators as they are termed) only appear when the user scrolls.

UIToolbar

- A **UIToolbar** displays a group of buttons at the bottom of the screen (usually).
    - You may be used to toolbars being at the top of the screen.
    - **UIToolbar** placement varies by device.
    - Each **UITooolbar** has one or more buttons, named **bar button items**.
    - These buttons (or **tools**) allow users to perform an action.

UINavigationBar

- The UINavigationBar can be thought of as the 'title bar' - but isn't the only way to make one.
- It has buttons to navigate through the view hierarchy (back, forwards etc).
- It looks like it sometimes has other buttons (although they are not implemented using a **UINavigationBar**).
- Don’t forget that it has the all important view title!

UITabBar

- Looks similar to a UIToolbar, but serves a **different yet similar purpose.**! Usually appears at the same position on screen as a UIToolbar.
- Rather than performing actions, a **UITabBar** allows the user to select between views.
- It is used in conjunction with a **TabBarController**.
- It is best used for when the app has multiple workflows i.e. doesn’t just run from start to finish like a ‘wizard’.

Basic Controls

Controls (vs Views)

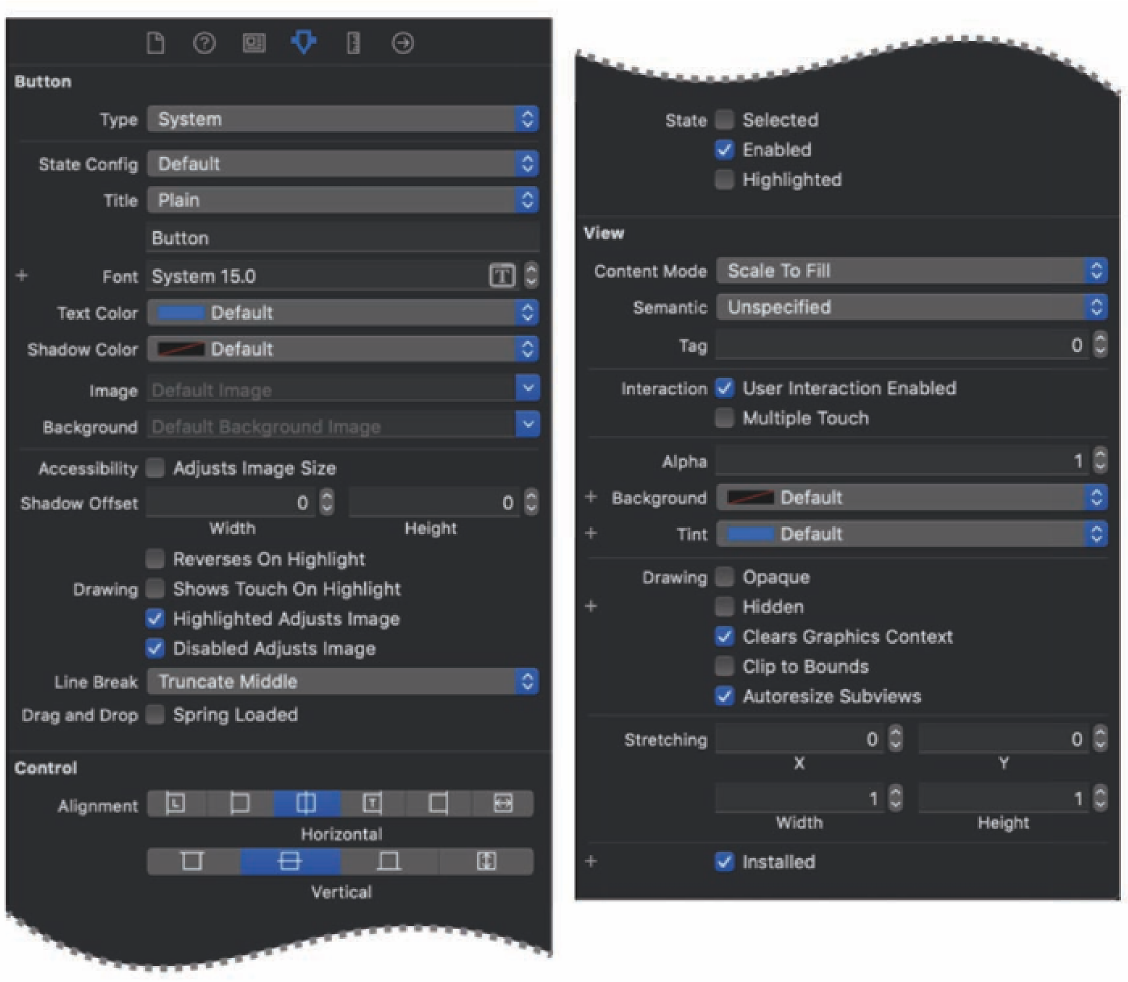
- Views showed things to the user, but often we want to react to the user’s input.
    - This is where controls come in – this is the main difference between the two.

- Controls allow us to setup Actions that respond to events (interactions) with the controls.
- An **Action** is simply a special type of function.
- You'll probably recognise a few of these...

UIButton

- A **UIButton** is a control which the user can 'tap'.
  - A button is just an icon or text which can be tapped.
  - 'Tapping' runs a block of Swift code (a function).
  - Generally bound (runs) on when you 'let go' of the button, but there are many other ways to bind the event as well.

UIButton Attributes Inspector



Button

Type: System
State Config: Default
Title: Plain
Button
Font: System 15.0
Text Color: Default
Shadow Color: Default
Image: Default Image
Background: Default Background Image
Accessibility: ☐ Adjusts Image Size
Shadow Offset: 0 (Width) 0 (Height)
Drawing: ☐ Reverses On Highlight
☐ Shows Touch On Highlight
☑ Highlighted Adjusts Image
☑ Disabled Adjusts Image
Line Break: Truncate Middle
Drag and Drop: ☐ Spring Loaded

Control

Alignment: Horizontal, Vertical

State: ☐ Selected
☑ Enabled
☐ Highlighted

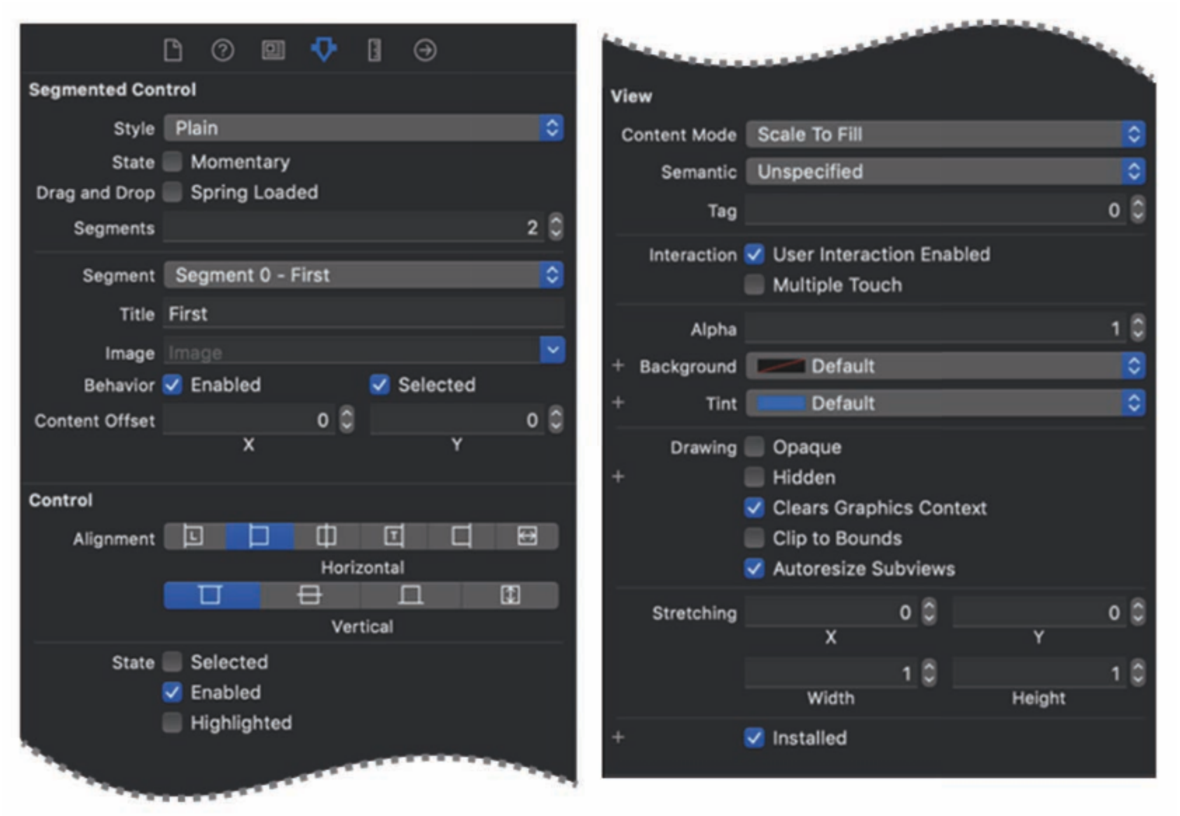
View

Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ☑ User Interaction Enabled
☐ Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: ☐ Opaque
☐ Hidden
☑ Clears Graphics Context
☐ Clip to Bounds
☑ Autoresize Subviews
Stretching: 0 (X) 0 (Y)
1 (Width) 1 (Height)
☑ Installed

UISegmentedControl

- Think of a **UISegmentedControl** as a group of buttons where only one can be selected.
    - Similar to an ‘option group’ for those who make web pages.
    - Code is run when the user selects a different button.
    - Generally, this is to choose between varying content.

UISegmentedControl Attributes Inspector



Segmented Control

- Style: Plain
- State: ☐ Momentary
- Drag and Drop: ☐ Spring Loaded
- Segments: 2
- Segment: Segment 0 - First
- Title: First
- Image: Image
- Behavior: ✓ Enabled ✓ Selected
- Content Offset: X 0, Y 0

Control

- Alignment: Horizontal, Vertical
- State: ☐ Selected, ✓ Enabled, ☐ Highlighted

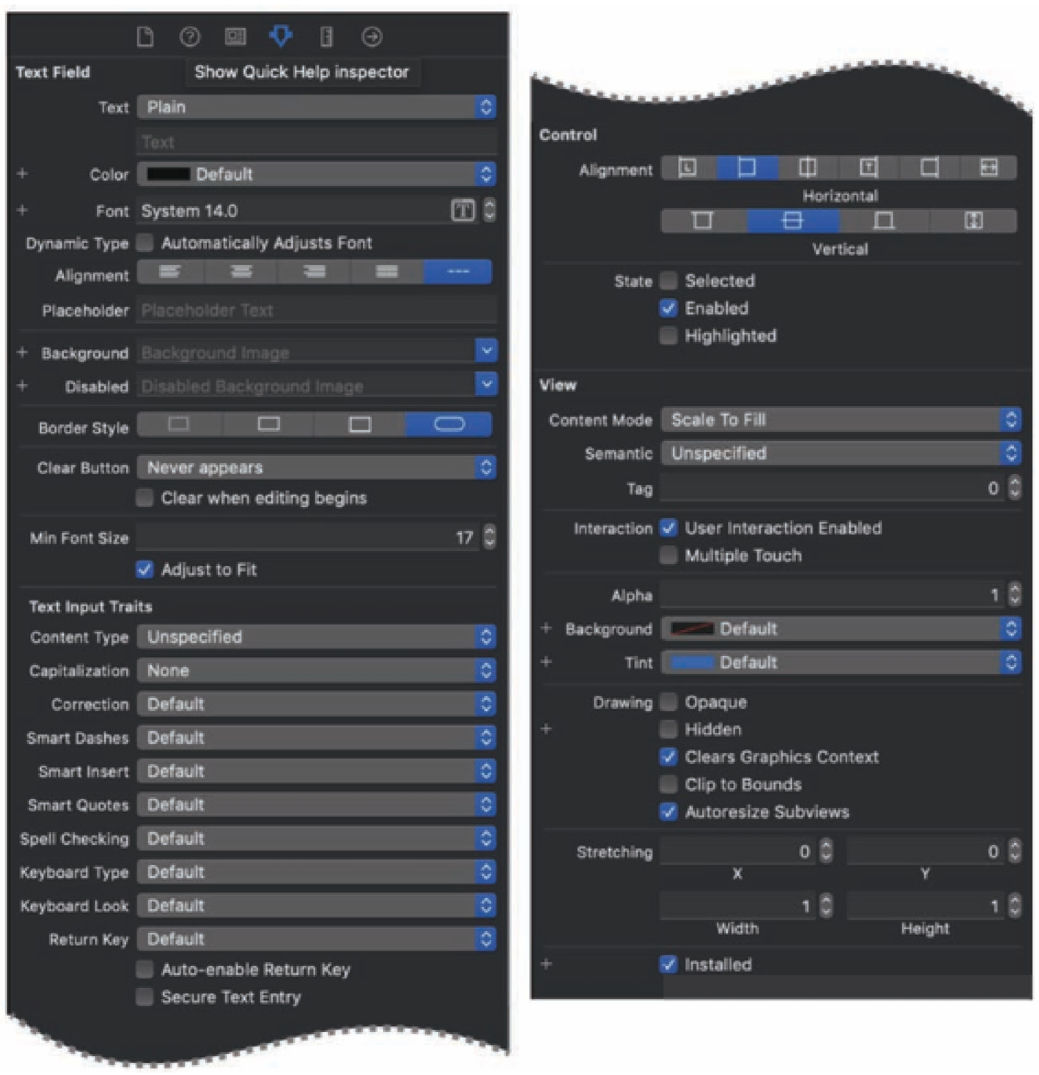
View

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- Interaction: ✓ User Interaction Enabled, ☐ Multiple Touch
- Alpha: 1
- Background: Default
- Tint: Default
- Drawing: ☐ Opaque, ☐ Hidden, ✓ Clears Graphics Context, ☐ Clip to Bounds, ✓ Autoresize Subviews
- Stretching: X 0, Y 0, Width 1, Height 1
- ✓ Installed

UITextField

- A **UITextField** is similar to a UITextView or a UILabel.
    - In that it allows the user to enter a single line of
    - text – these are properties of those Views.
    - When the user presses ‘Return’ or taps ‘Done’, this will run a function.
    - Code can also be bound to after each keystroke (use this carefully!)

UITextField Attributes Inspector

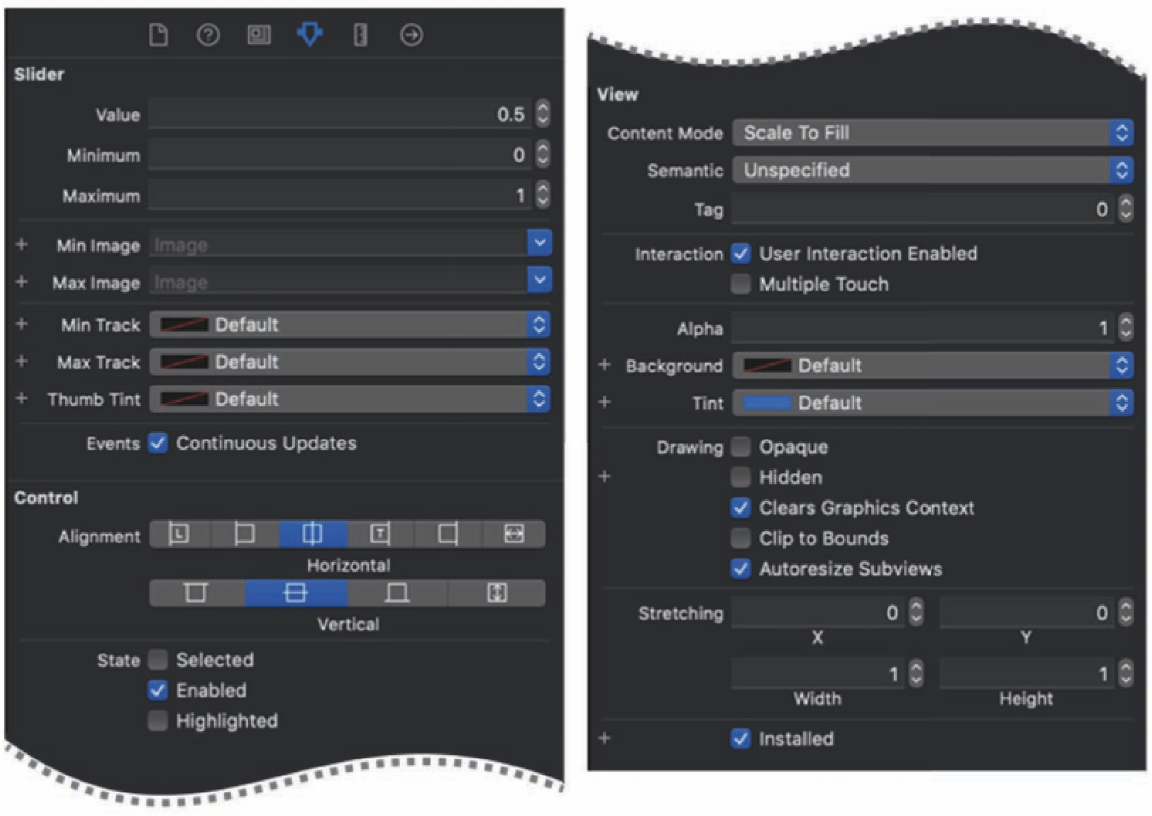

Text Field
Show Quick Help inspector
Text: Plain
Text
Color: Default
Font: System 14.0
Dynamic Type: Automatically Adjusts Font
Alignment
Placeholder: Placeholder Text
Background: Background Image
Disabled: Disabled Background Image
Border Style
Clear Button: Never appears
Clear when editing begins
Min Font Size: 17
✓ Adjust to Fit
Text Input Traits
Content Type: Unspecified
Capitalization: None
Correction: Default
Smart Dashes: Default
Smart Insert: Default
Smart Quotes: Default
Spell Checking: Default
Keyboard Type: Default
Keyboard Look: Default
Return Key: Default
Auto-enable Return Key
Secure Text Entry

Control
Alignment: Horizontal
Vertical
State: Selected
✓ Enabled
Highlighted
View
Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ✓ User Interaction Enabled
Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: Opaque
Hidden
✓ Clears Graphics Context
Clip to Bounds
✓ Autoresize Subviews
Stretching: X 0, Y 0
Width 1, Height 1
✓ Installed

UISlider

- A **UISlider** allows the user to select a continuous value between a lower and upper range.
  - As the user drags the slider toggle, code is run.
  - Hence, dragging can trigger an event many times.
  - Examples are volume sliders, number choosers...

UISlider Attributes Inspector



Slider
Value 0.5
Minimum 0
Maximum 1
Min Image Image
Max Image Image
Min Track Default
Max Track Default
Thumb Tint Default
Events ✓ Continuous Updates

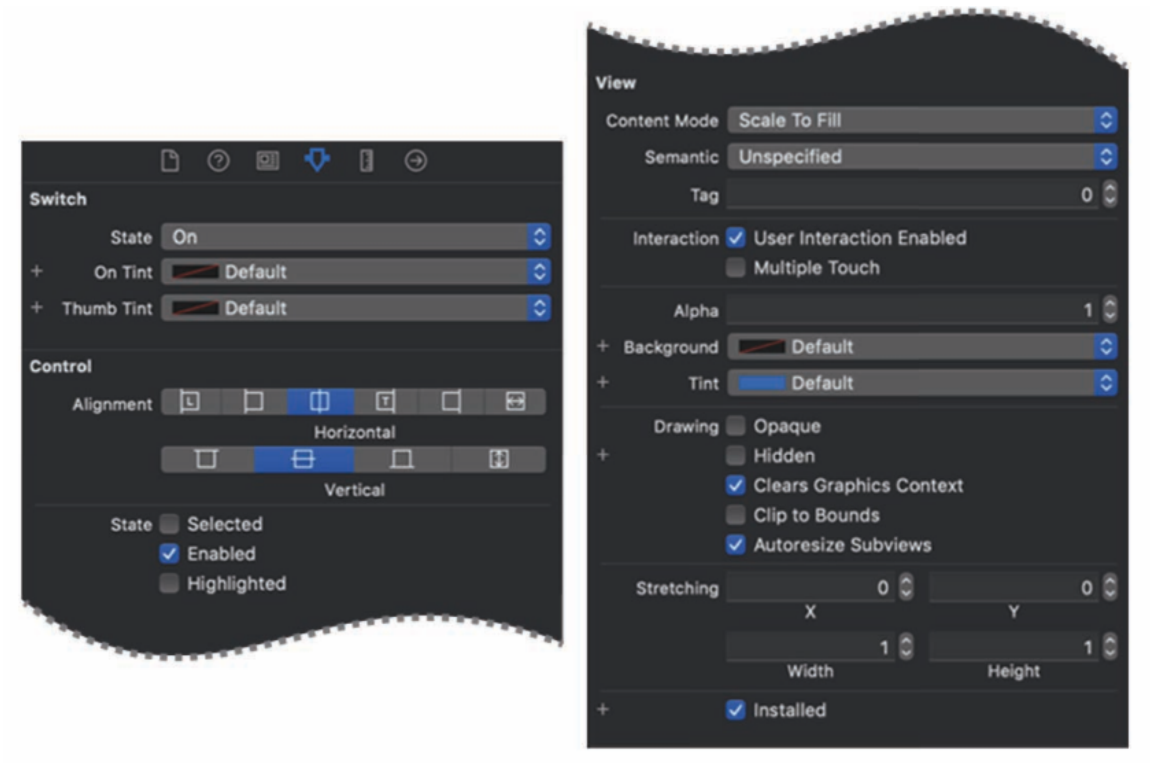
Control
Alignment
Horizontal
Vertical
State Selected
✓ Enabled
Highlighted

View
Content Mode Scale To Fill
Semantic Unspecified
Tag 0
Interaction ✓ User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque
Hidden
✓ Clears Graphics Context
Clip to Bounds
✓ Autoresize Subviews
Stretching 0 0
X Y
1 1
Width Height
✓ Installed

UISwitch

- **UISwitch**es are to controls what Boolean values are to variables.
  - They can be a 'yes or no'.
  - Code can run for when the state changes or when it is set to a specific value (yes and/or no).

UISwitch Attributes Inspector



Switch
State: On
On Tint: Default
Thumb Tint: Default

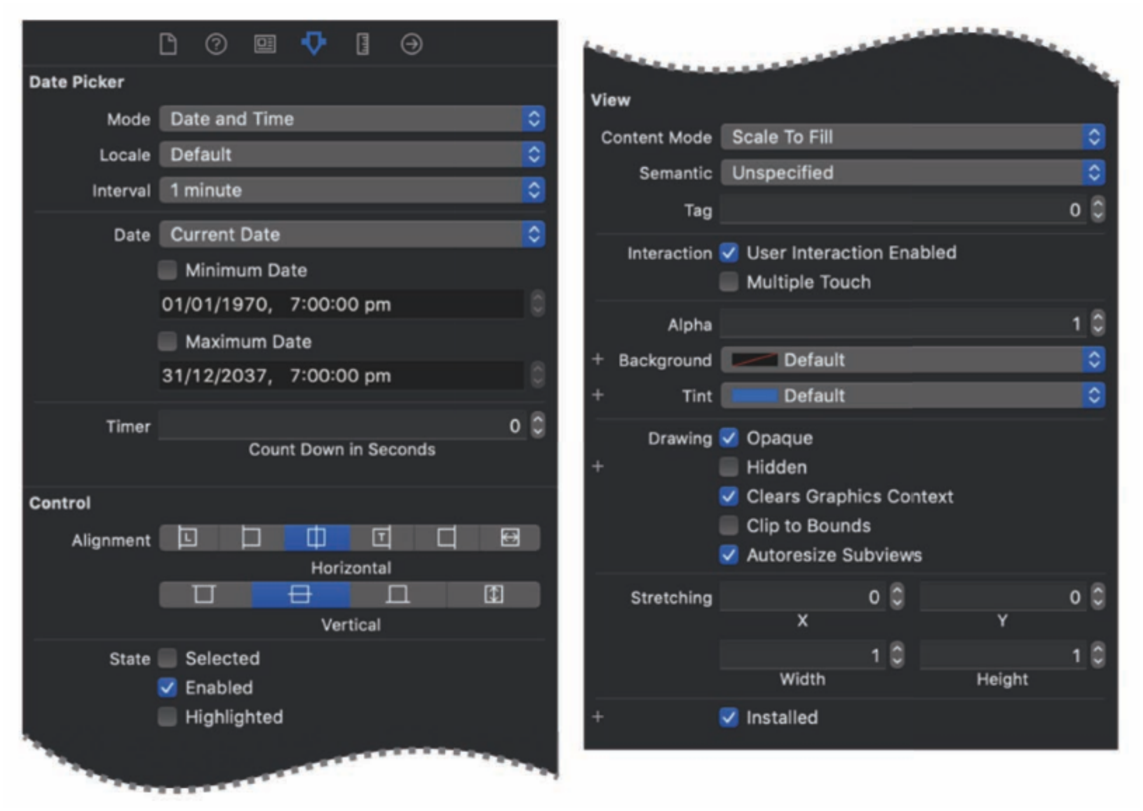
Control
Alignment: Horizontal, Vertical
State: Selected ☐, Enabled ✓, Highlighted ☐

View
Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ✓ User Interaction Enabled, ☐ Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: ☐ Opaque, ☐ Hidden, ✓ Clears Graphics Context, ☐ Clip to Bounds, ✓ Autoresize Subviews
Stretching: X 0, Y 0, Width 1, Height 1
✓ Installed

UIDatePicker

- A **UIDatePicker** provides a convenient way to deal with a complex but common problem.
  - that is, allowing the user to select a date and time.
  - Can be used for calendars, dates of birth.
  - Code is executed when date and/or time is changed.

UIDatePicker Attributes Inspector


Date Picker
Mode: Date and Time
Locale: Default
Interval: 1 minute
Date: Current Date
☐ Minimum Date
01/01/1970, 7:00:00 pm
☐ Maximum Date
31/12/2037, 7:00:00 pm
Timer: 0
Count Down in Seconds

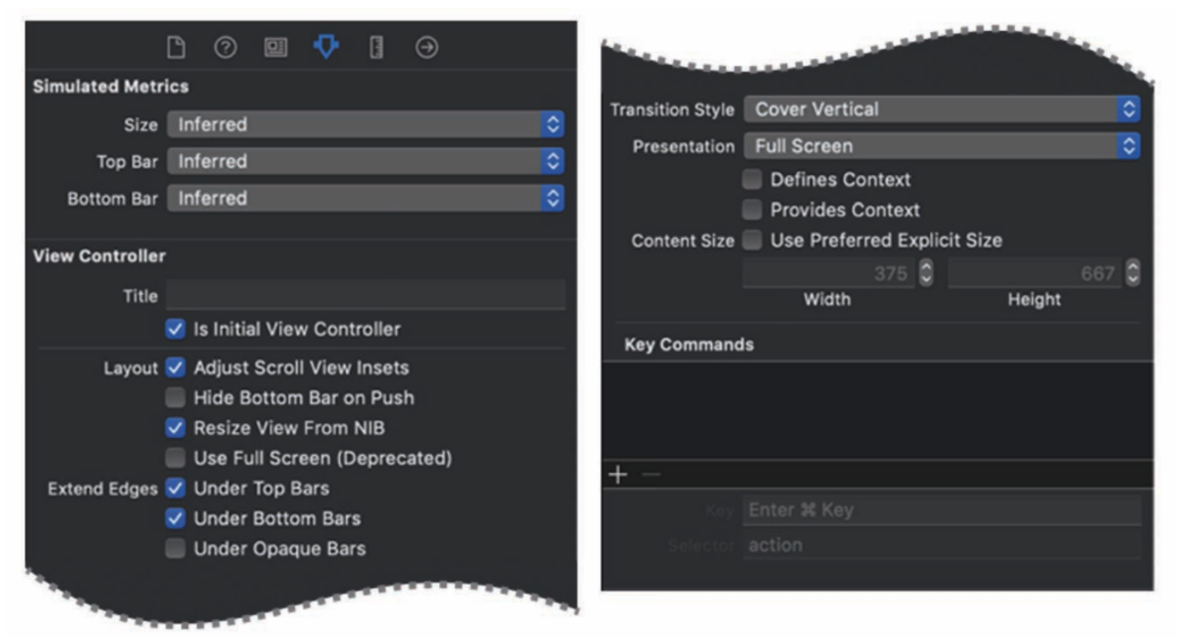
Control
Alignment
Horizontal
Vertical
State: ☐ Selected
✓ Enabled
☐ Highlighted

View
Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ✓ User Interaction Enabled
☐ Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: ✓ Opaque
☐ Hidden
✓ Clears Graphics Context
☐ Clip to Bounds
✓ Autoresize Subviews
Stretching: X 0, Y 0
Width 1, Height 1
✓ Installed

UIViewController

- A **UIViewController** is not really a control, but a special case view.
  - Each screen in an app is usually a scene in Storyboards.
  - **UIViewController**s manage each view – they contain the Actions and the Storyboards used to generate each screen or scene.
  - These are subclassed and overridden to allow us to make our own scenes.

UIViewController Attributes Inspector


Simulated Metrics
Size: Inferred
Top Bar: Inferred
Bottom Bar: Inferred

View Controller
Title:
[✓] Is Initial View Controller
Layout: [✓] Adjust Scroll View Insets
[ ] Hide Bottom Bar on Push
[✓] Resize View From NIB
[ ] Use Full Screen (Deprecated)
Extend Edges: [✓] Under Top Bars
[✓] Under Bottom Bars
[ ] Under Opaque Bars

Transition Style: Cover Vertical
Presentation: Full Screen
[ ] Defines Context
[ ] Provides Context
Content Size: [ ] Use Preferred Explicit Size
Width: 375 Height: 667

Key Commands
Key: Enter ⌘ Key
Selector: action

Exercises

If you would like to practice some more coding, try the following extension exercises.

You will find that these two exercises build on the demonstrations provided earlier, so you may like to go back and watch (or create) the demos before you begin.

Alternatively, here is the playground file that contains the code written during the screen capture. Modify this file to complete the exercises detailed below.

Exercise One

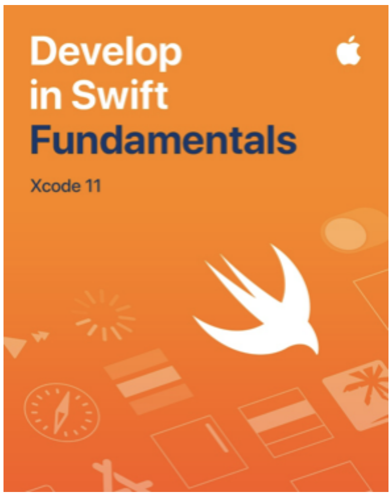
Modify the Playgrounds code to use a **UIView** rather than a **UIScrollView**. Adjust the placement of the **UIDatePicker** such that there is no need for the **UIScrollView**.

Exercise Two

Add a **UISlider** with a range between 1 and 10 inclusive between your button and **UIDatePicker**. Output a message when the value is changed, in the same manner as the **UIDatePicker**. Comment on any interesting behaviour you notice.

Extra Resources

- In Apple Books:



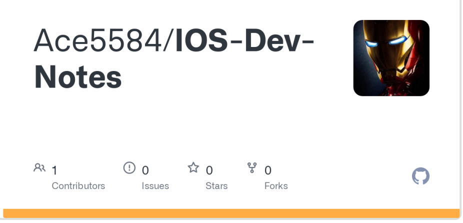
- 2.7 Introduction to UIKit
- 2.8 Displaying Data
- 2.9 Controls in Action
- 2.10 Auto Layout and Stack Views

Source Code:

Ace5584/IOS-Dev-Notes

Contribute to Ace5584/IOS-Dev-Notes development by creating an account on GitHub.

https://github.com/Ace5584/IOS-Dev-Notes/tree/main


Ace5584/**IOS-Dev-Notes**

1 Contributors 0 Issues 0 Stars 0 Forks