Sun*

UIControls

Outline

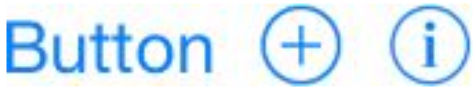
1. UIButton
2. UIDatePicker
3. UISegmentedControl
4. UISlider
5. UIStepper
6. UISwitch

1. UIButton

❖ A control that executes your custom code in response to user interactions.

❖ When you tap a button, or select a button that has focus, the button performs any actions attached to it. You communicate the purpose of a button using a text label, an image, or both.

❖ The appearance of buttons is configurable, so you can tint buttons or format titles to match the design of your app.



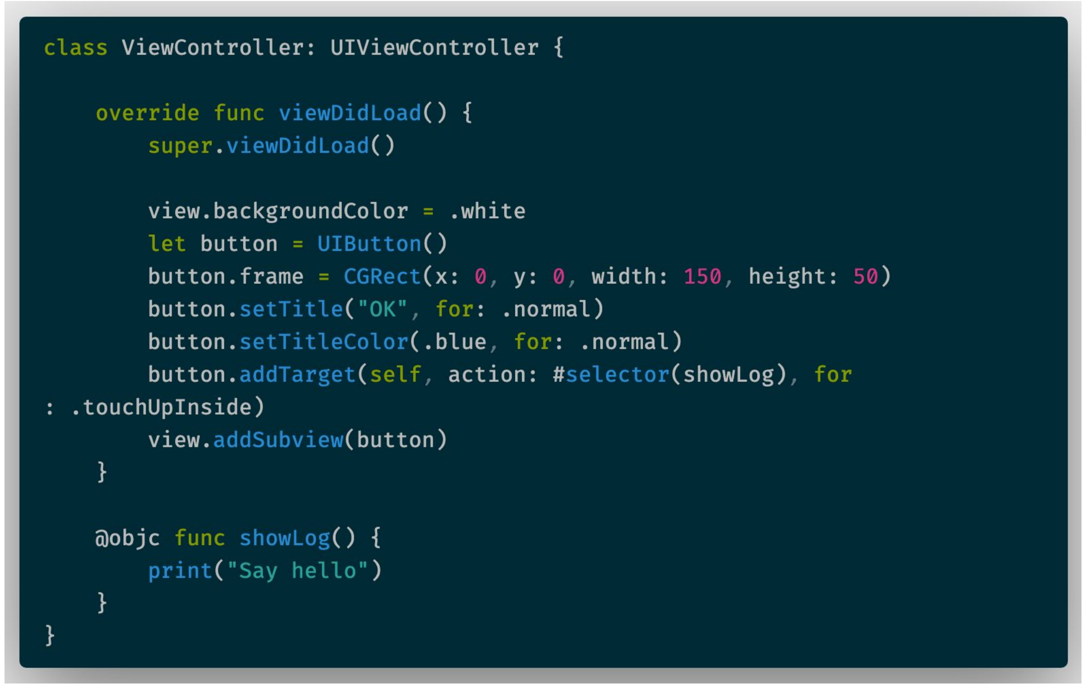
1. UIButton (cont)

❖ When adding a button to your interface, perform the following steps:

- ➔ Set the type of the button at creation time.
- ➔ Supply a title string or image; size the button appropriately for your content.
- ➔ Connect one or more action methods to the button.
- ➔ Set up Auto Layout rules to govern the size and position of the button in your interface.
- ➔ Provide accessibility information and localized strings.

1. UIButton (cont)

❖ Creating a button by code:

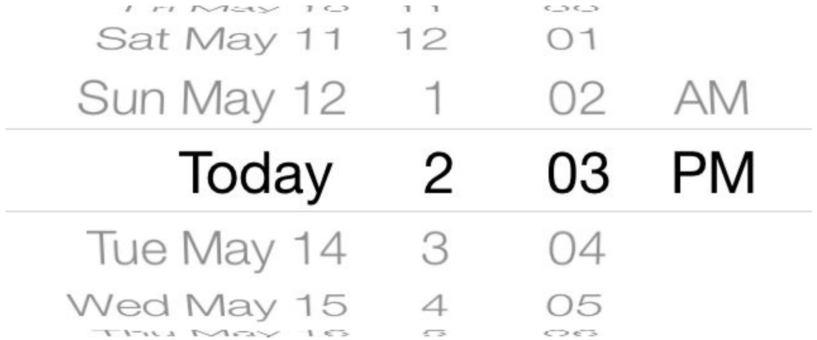


```
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white
        let button = UIButton()
        button.frame = CGRect(x: 0, y: 0, width: 150, height: 50)
        button.setTitle("OK", for: .normal)
        button.setTitleColor(.blue, for: .normal)
        button.addTarget(self, action: #selector(showLog), for
: .touchUpInside)
        view.addSubview(button)
    }

    @objc func showLog() {
        print("Say hello")
    }
}
```

Sun*

2. UIDatePicker

- A control used for the inputting of date and time values.
- You can use a date picker to allow a user to enter either a point in time (calendar date, time value or both) or a time interval (for example for a timer).
- The date picker reports interactions to its associated target object.



Sun*

2. UIDatePicker (cont)

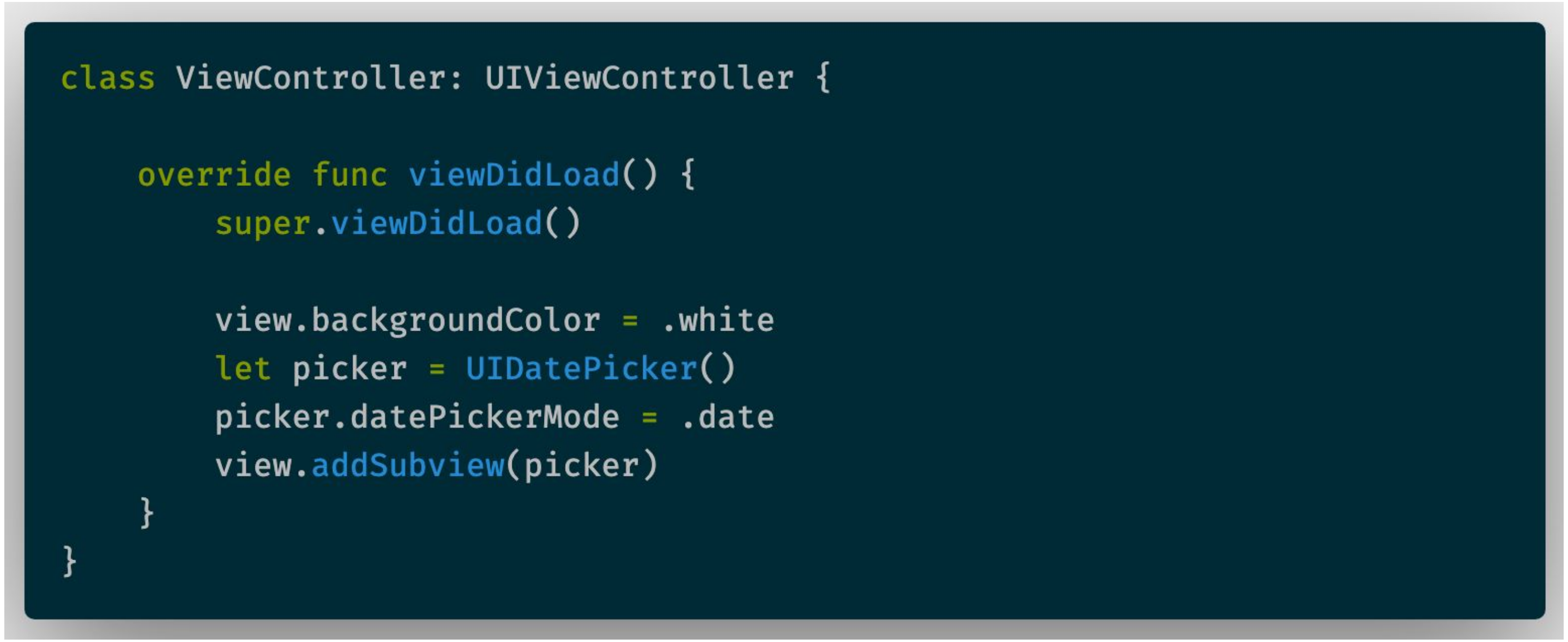
❖ To add a date picker to your interface:

- ➔ Set the date picker mode at creation time.
- ➔ Supply additional configuration options such as minimum and maximum dates if required.
- ➔ Connect an action method to the date picker.

Sun*

2. UIDatePicker (cont)

❖ Creating a DatePicker by code:

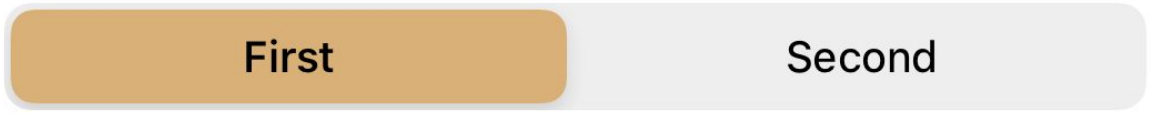


```
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = .white
        let picker = UIDatePicker()
        picker.datePickerMode = .date
        view.addSubview(picker)
    }
}
```

Sun*

3. UISegmentedControl

- ❖ A horizontal control made of multiple segments, each segment functioning as a discrete button.
- ❖ A segmented control can display a title (an `NSString` object) or an image (`UIImage` object). The `UISegmentedControl` object automatically resizes segments to fit proportionally within their superview unless they have a specific width set.
- ❖ When you add and remove segments, you can request that the action be animated with sliding and fading effects.



3. UISegmentedControl (cont)

❖ How you configure a segmented control can affect its display behavior:

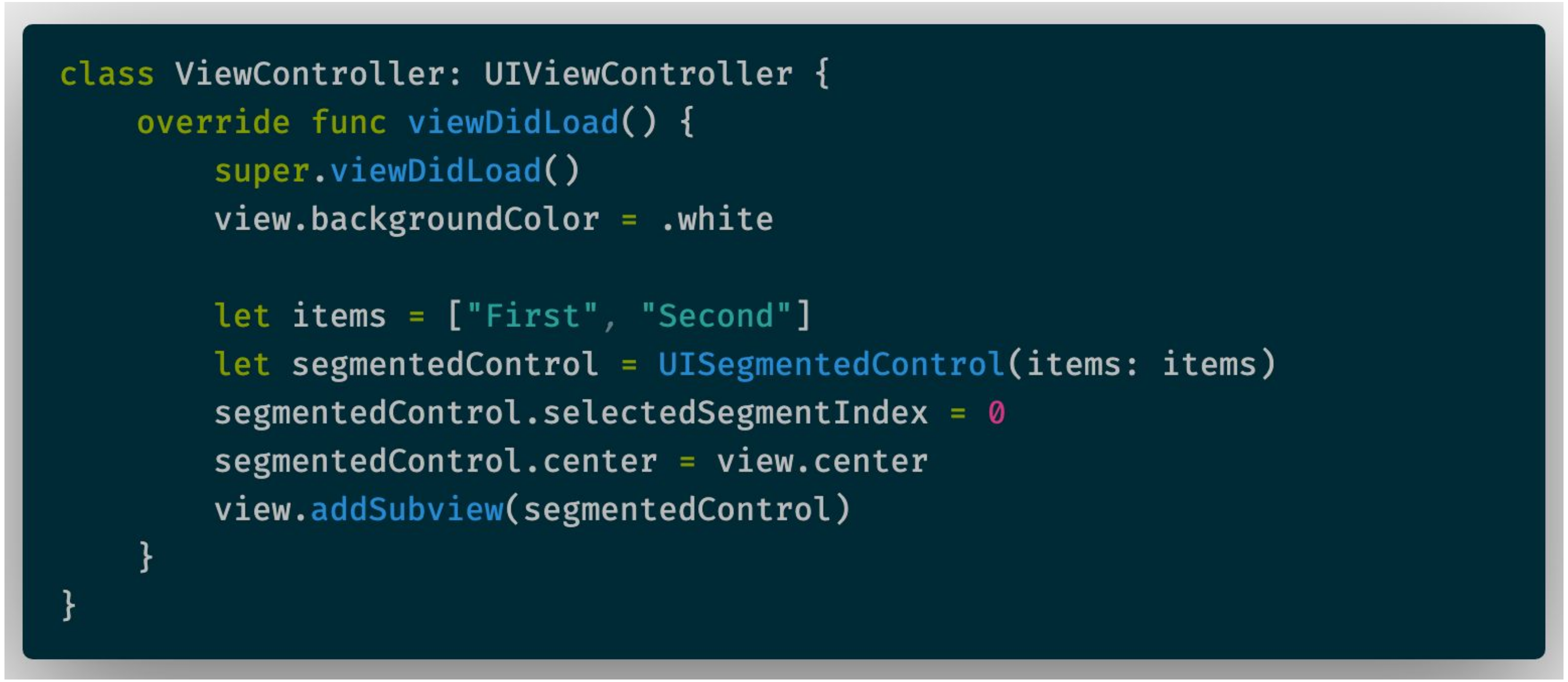
- ➔ If you set a segmented control to have a momentary style, a segment doesn’t show itself as selected (blue background) when the user touches it. The disclosure button is always momentary and doesn’t affect the actual selection.

- ➔ In versions of iOS prior to 3.0, if a segmented control has only two segments, then it behaves like a switch—tapping the currently-selected segment causes the other segment to be selected. On iOS 3.0 and later, tapping the currently-selected segment does not cause the other segment to be selected.

Sun*

3. UISegmentedControl (cont)

❖ Creating a `UISegmentedControl` by code:

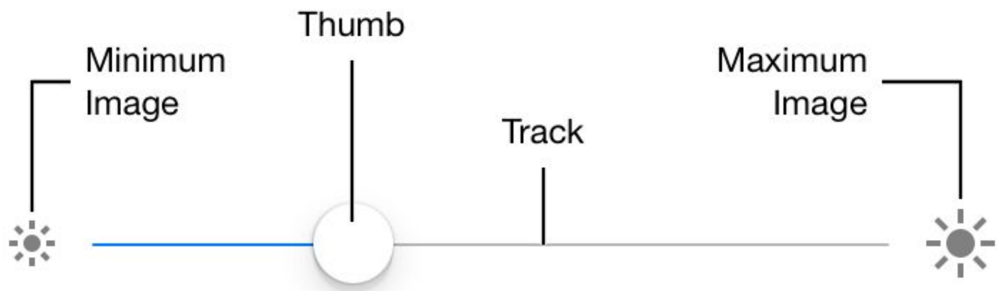


```swift
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .white

        let items = ["First", "Second"]
        let segmentedControl = UISegmentedControl(items: items)
        segmentedControl.selectedSegmentIndex = 0
        segmentedControl.center = view.center
        view.addSubview(segmentedControl)
    }
}
```

Sun*

4. UISlider

- ❖ A control used to select a single value from a continuous range of values.
- ❖ As you move the thumb of a slider, it passes its updated value to any actions attached to it. The appearance of sliders is configurable; you can tint the track and the thumb, and provide images to appear at the ends of the slider.

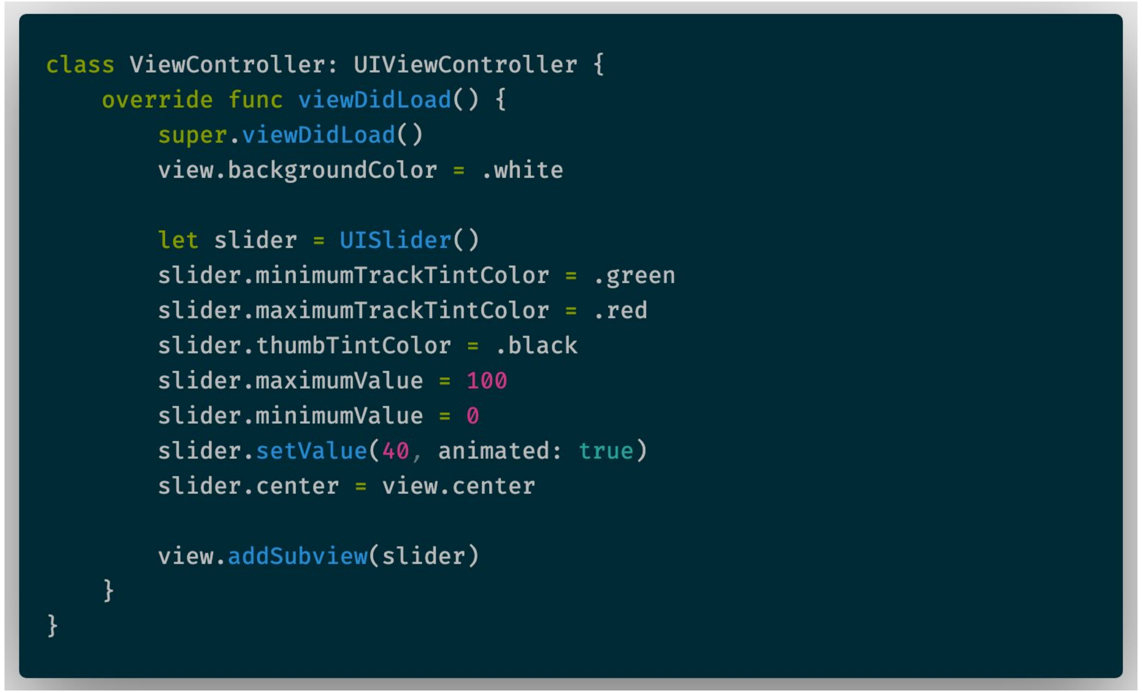


4. UISlider (cont)

- ❖ The following steps are required to add a slider to your interface:
  - ➔ Specify the range of values the slider represents.
  - ➔ Optionally, configure the appearance of the slider with appropriate tint colors, and limit images.
  - ➔ Connect one or more action methods to the slider.
  - ➔ Set up Auto Layout rules to govern the size and position of the slider in your interface.

4. UISlider (cont)

- Creating a `UISlider` by code:

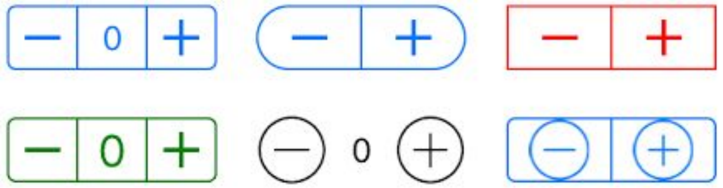


```swift
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .white

        let slider = UISlider()
        slider.minimumTrackTintColor = .green
        slider.maximumTrackTintColor = .red
        slider.thumbTintColor = .black
        slider.maximumValue = 100
        slider.minimumValue = 0
        slider.setValue(40, animated: true)
        slider.center = view.center

        view.addSubview(slider)
    }
}
```

Sun*

5. UIStepper

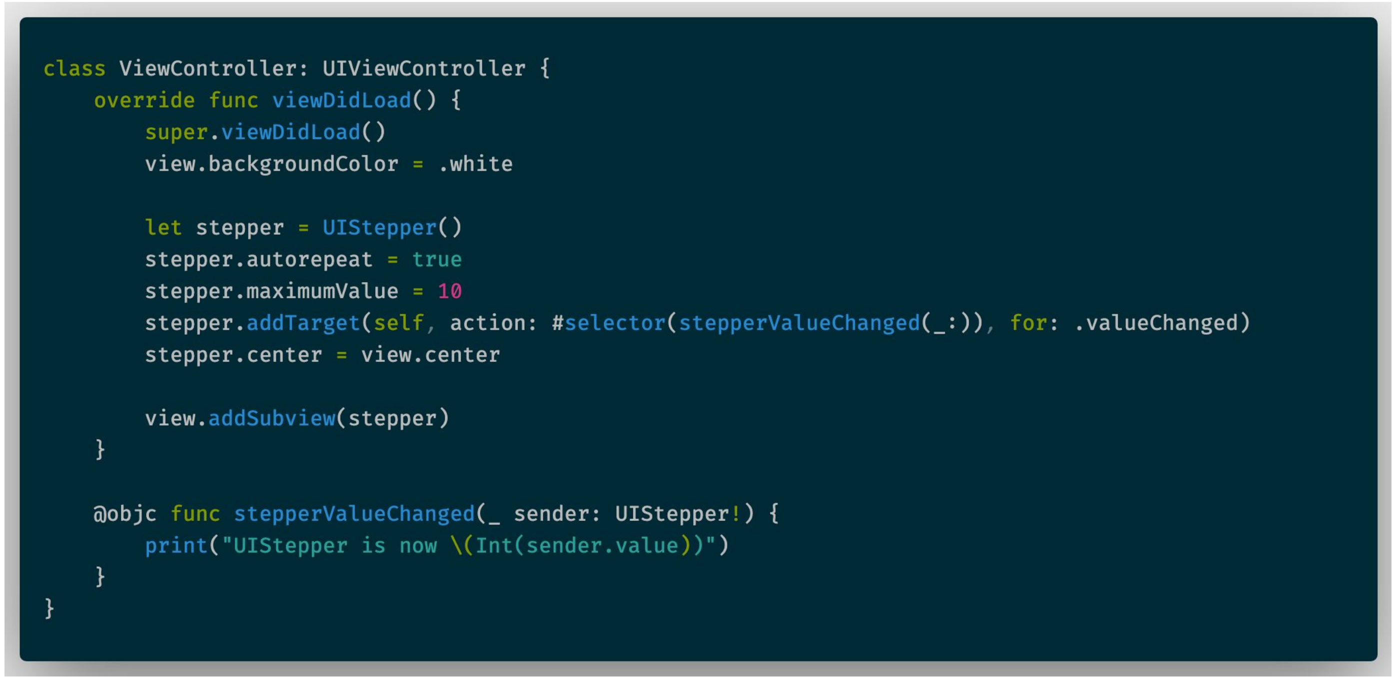
- ❖ A control used to increment or decrement a value.
- ❖ If you set stepper behavior to “autorepeat” (which is the default), pressing and holding one of its buttons increments or decrements the stepper’s value repeatedly. The rate of change depends on how long the user continues pressing the control.
- ❖ The maximum value must be greater than or equal to the minimum value. If you set a maximum or minimum value that would break this invariant, both values are set to the new value.



Sun*

5. UIStepper (cont)

❖ Creating a `UIStepper` by code:



```swift
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .white

        let stepper = UIStepper()
        stepper.autorepeat = true
        stepper.maximumValue = 10
        stepper.addTarget(self, action: #selector(stepperValueChanged(_:)), for: .valueChanged)
        stepper.center = view.center

        view.addSubview(stepper)
    }

    @objc func stepperValueChanged(_ sender: UIStepper!) {
        print("UIStepper is now \(Int(sender.value))")
    }
}
```

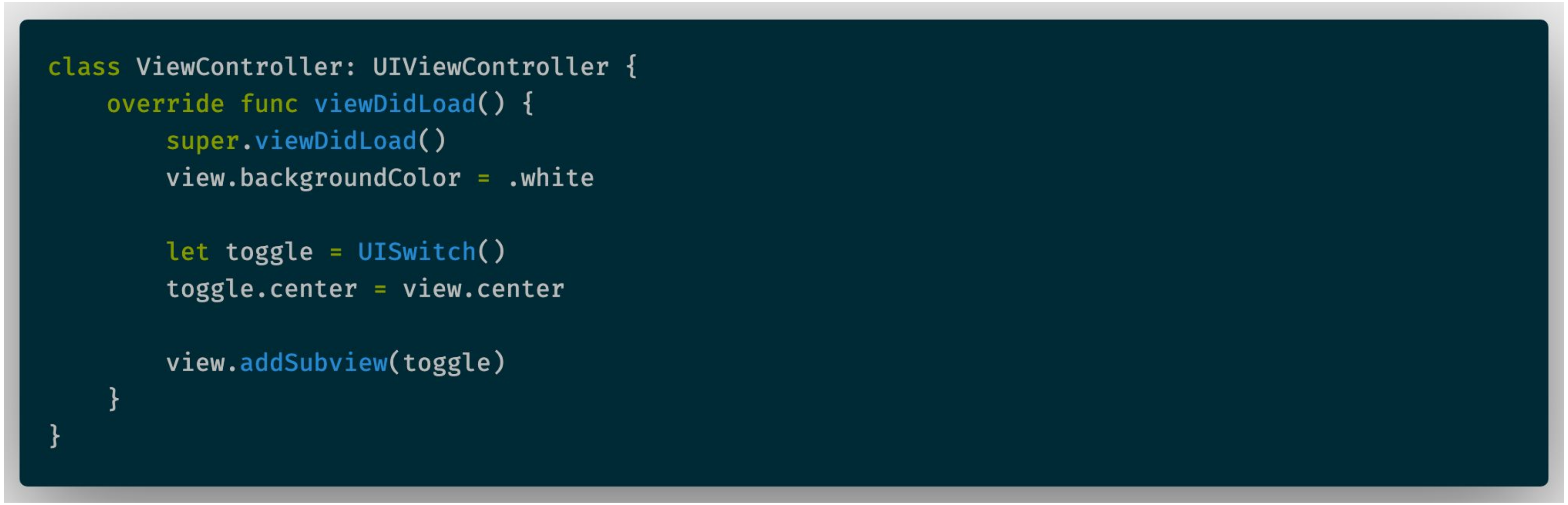
6. UISwitch

- ❖ A control that offers a binary choice, such as On/Off.
- ❖ The `UISwitch` class declares a property and a method to control its on/off state. As with `UISlider`, when the user manipulates the switch control (“flips” it) a `valueChanged` event is generated, which results in the control (if properly configured) sending an action message.
- ❖ You can customize the appearance of the switch by changing the color used to tint the switch when it is on or off.

Sun*

6. UISwitch (cont)

- Creating a `UISwitch` by code:



```swift
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .white

        let toggle = UISwitch()
        toggle.center = view.center

        view.addSubview(toggle)
    }
}
```

Sun*

Question & Answer?

Sun*