LECTURE3 MATCHING CARD

IOS DEVELOPMENT

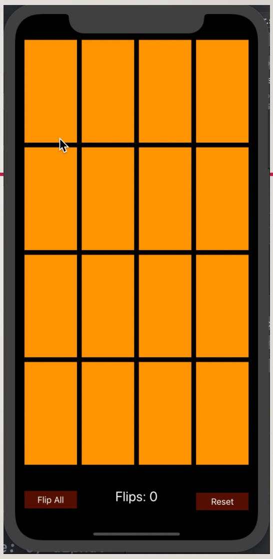
NATIONAL TAIPEI UNIVERSITY OF TECHNOLOGY

PROF. PEIYING CHIANG

1

GOAL: CARD MATCHING GAME

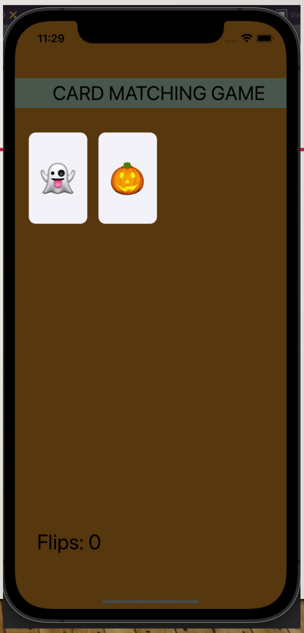
- Flip Card
- Emojis
- Matching a pair of Cards
- Counts



2

FIRST SIMPLIFIED GOAL

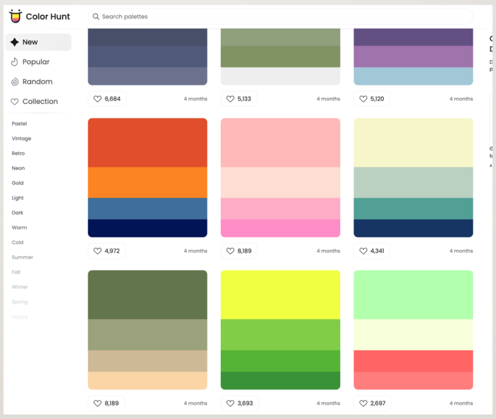
- Two cards
  - 2 labels
  - 2 buttons

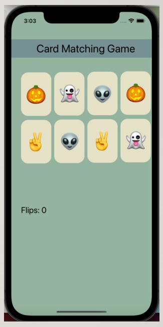


3

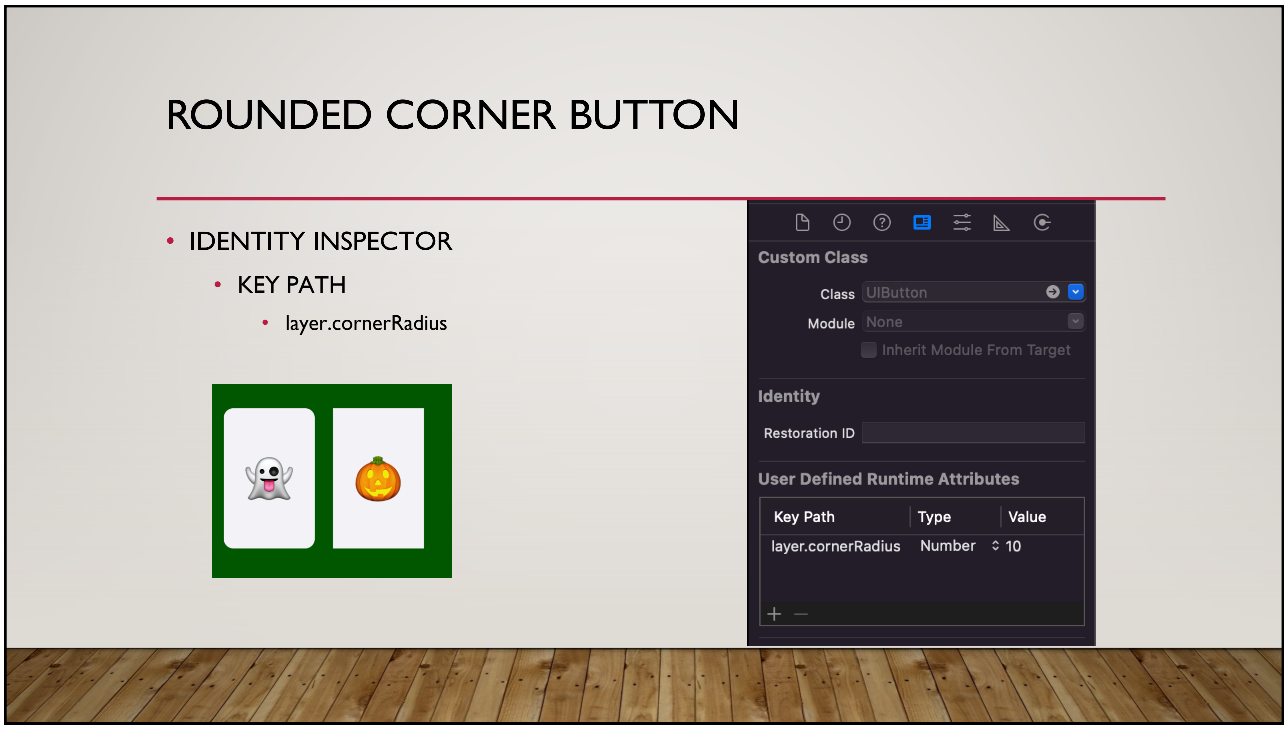
COLOR PALETTE

- https://colorhunt.co/





4



ROUNDED CORNER BUTTON

- IDENTITY INSPECTOR
  - KEY PATH
    - layer.cornerRadius

Custom Class

Class UIButton

Module None

Inherit Module From Target

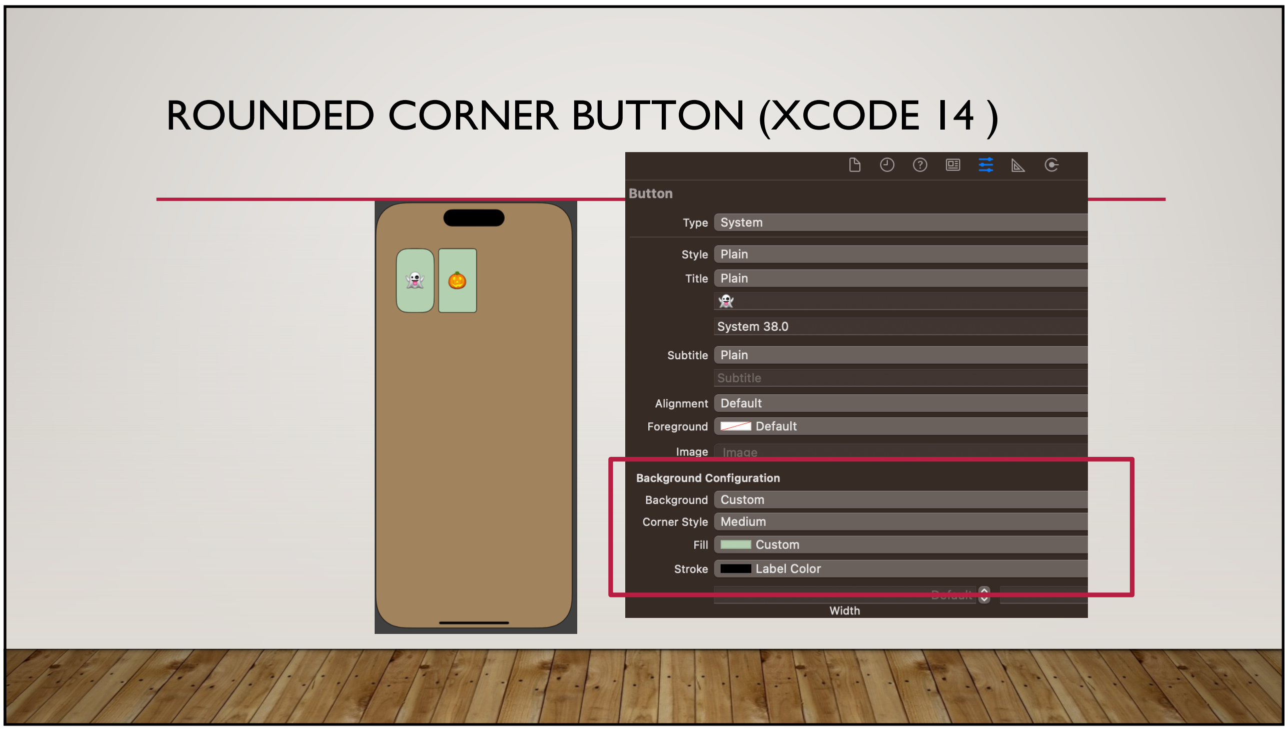
Identity

Restoration ID

User Defined Runtime Attributes

Key Path	Type	Value
layer.cornerRadius	Number	10

5



ROUNDED CORNER BUTTON (XCODE 14)

Button

Type	System
Style	Plain
Title	Plain
	👻
	System 38.0
Subtitle	Plain
	Subtitle
Alignment	Default
Foreground	Default
Image	Image

Background Configuration

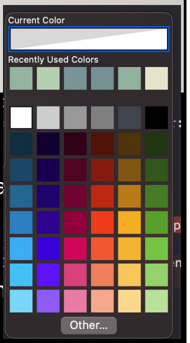
Background	Custom
Corner Style	Medium
Fill	🟩 Custom
Stroke	⬛ Label Color
	Default
	Width

6

#colorLiteral() will turn into a broken image icon and click on it

TEXT WITH NO ATTRIUBTE

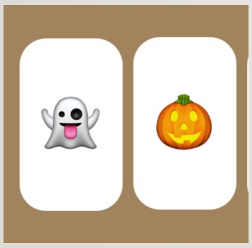
```
@IBAction func touchCard(_ sender: UIButton) {
      if sender.currentTitle == "🎃"{
           sender.setTitle("", for: UIControl.State.normal)
           sender.backgroundColor =   colorLiteral(red: 1, green: 0.5, blue: 0, a
       }else{

           sender.setTitle("🎃", for: UIControl.State.normal)
           sender.backgroundColor =  colorLiteral(red: 1, green: 1, blue: 1, alph
       }
       flipCount += 1

   }
```



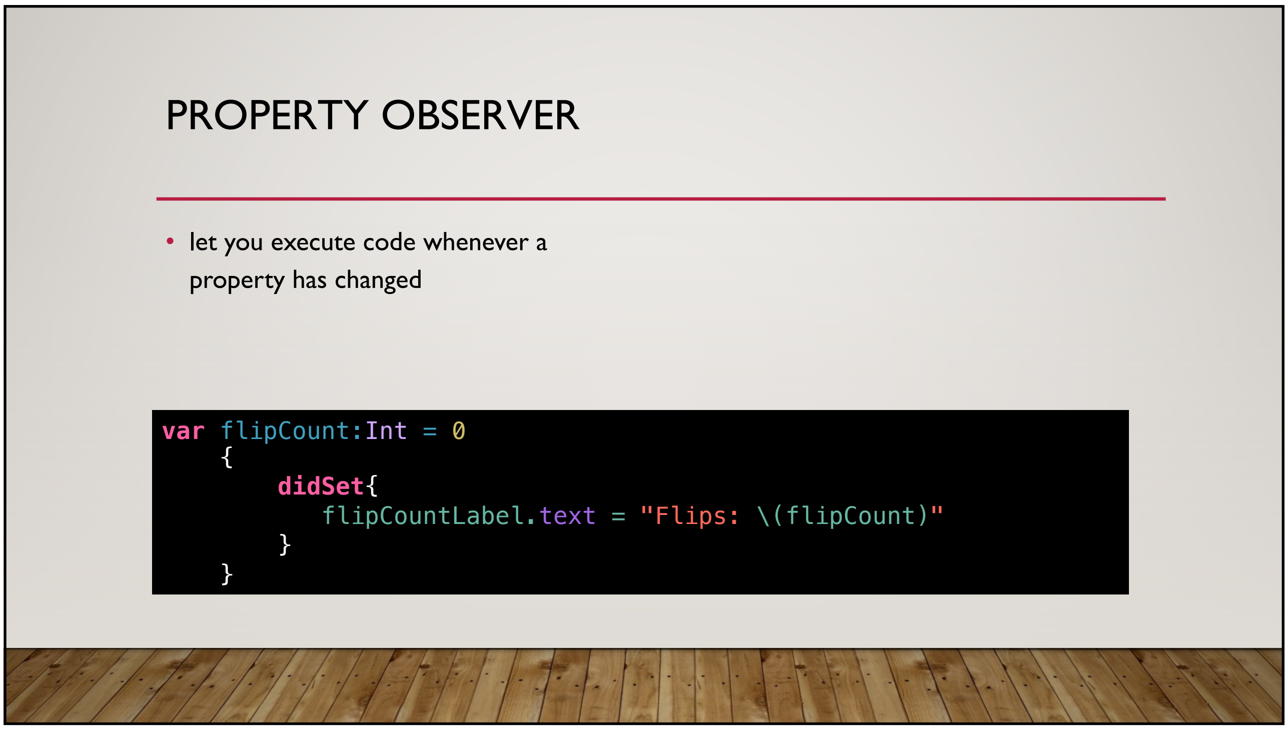
```
sender.backgroundColor = 🟧

//#colorLiteral(red: 1, green: 0.5763723254, blue:
    0, alpha: 1)
```

PRACTICE

- Add another button



- Each card is flippable
- Flip count

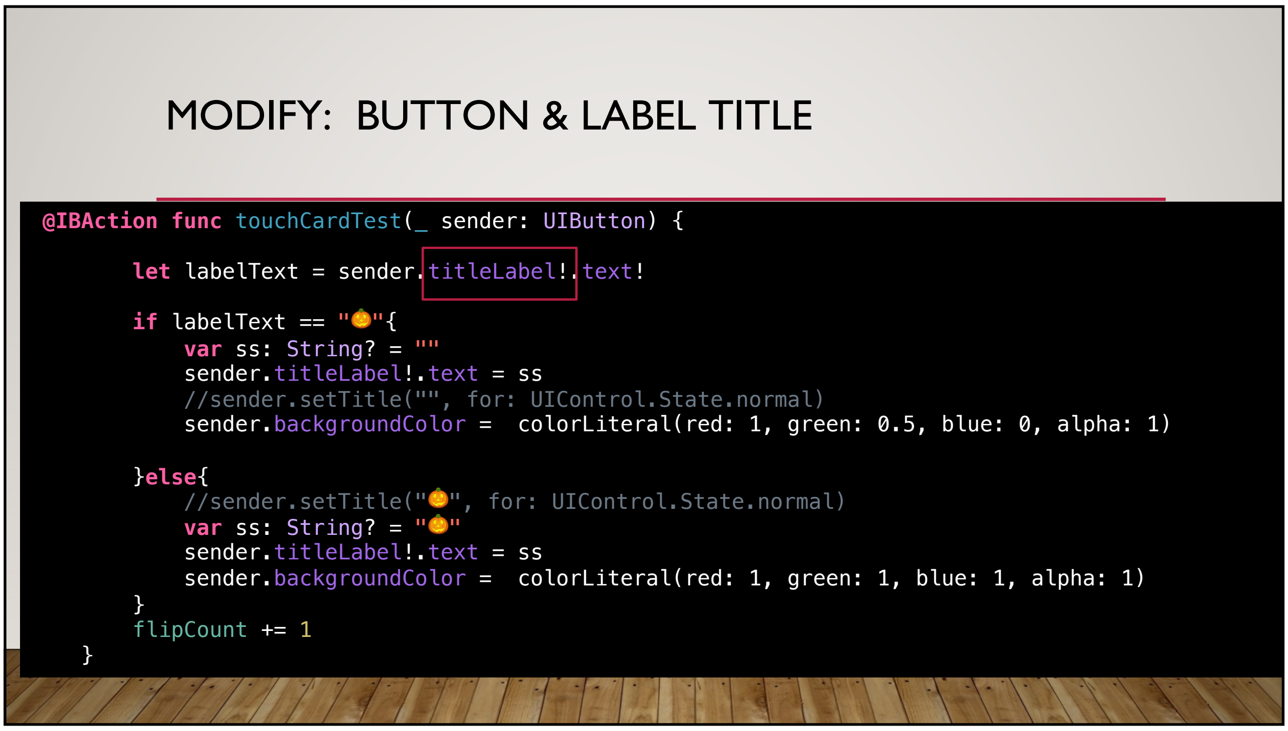


PROPERTY OBSERVER

- let you execute code whenever a property has changed

```
var flipCount:Int = 0
    {
        didSet{
            flipCountLabel.text = "Flips: \(flipCount)"
        }
    }
```

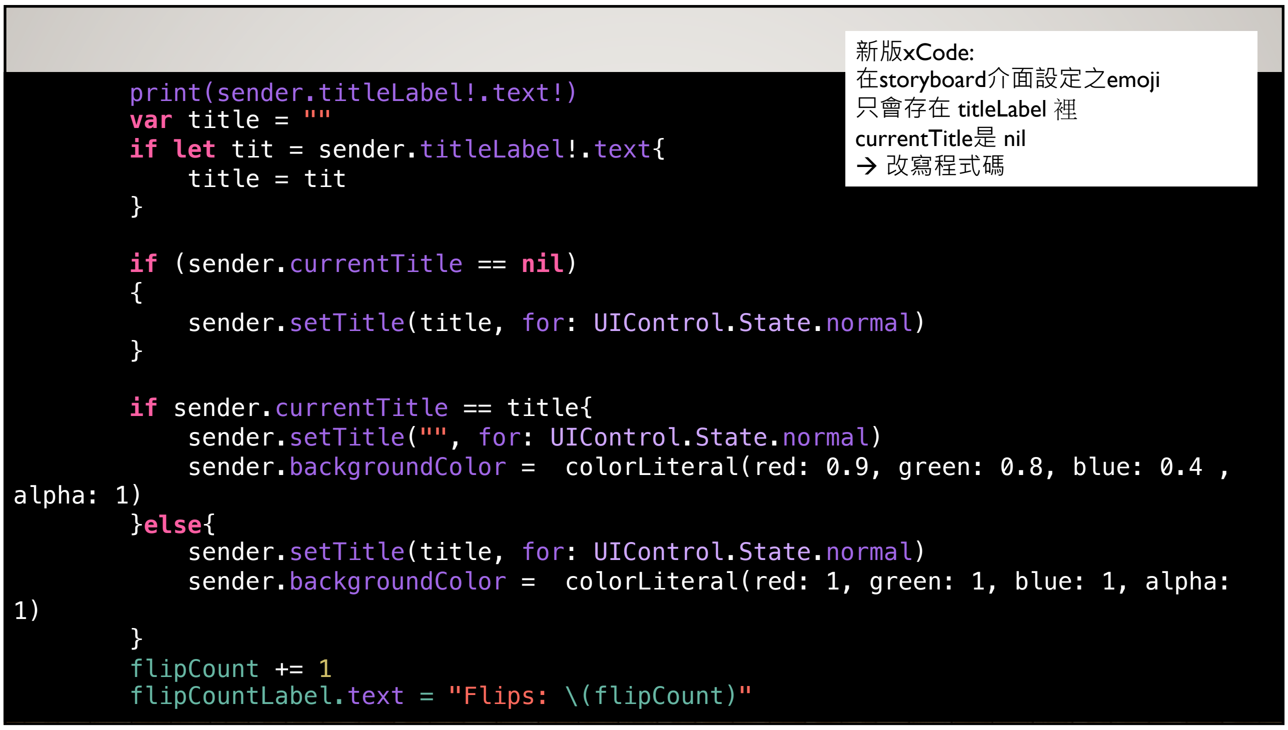
9



MODIFY: BUTTON & LABEL TITLE

```
@IBAction func touchCardTest(_ sender: UIButton) {

        let labelText = sender.titleLabel!.text!

        if labelText == "🎃"{
            var ss: String? = ""
            sender.titleLabel!.text = ss
            //sender.setTitle("", for: UIControl.State.normal)
            sender.backgroundColor =  colorLiteral(red: 1, green: 0.5, blue: 0, alpha: 1)

        }else{
            //sender.setTitle("🎃", for: UIControl.State.normal)
            var ss: String? = "🎃"
            sender.titleLabel!.text = ss
            sender.backgroundColor =  colorLiteral(red: 1, green: 1, blue: 1, alpha: 1)
        }
        flipCount += 1
    }
```

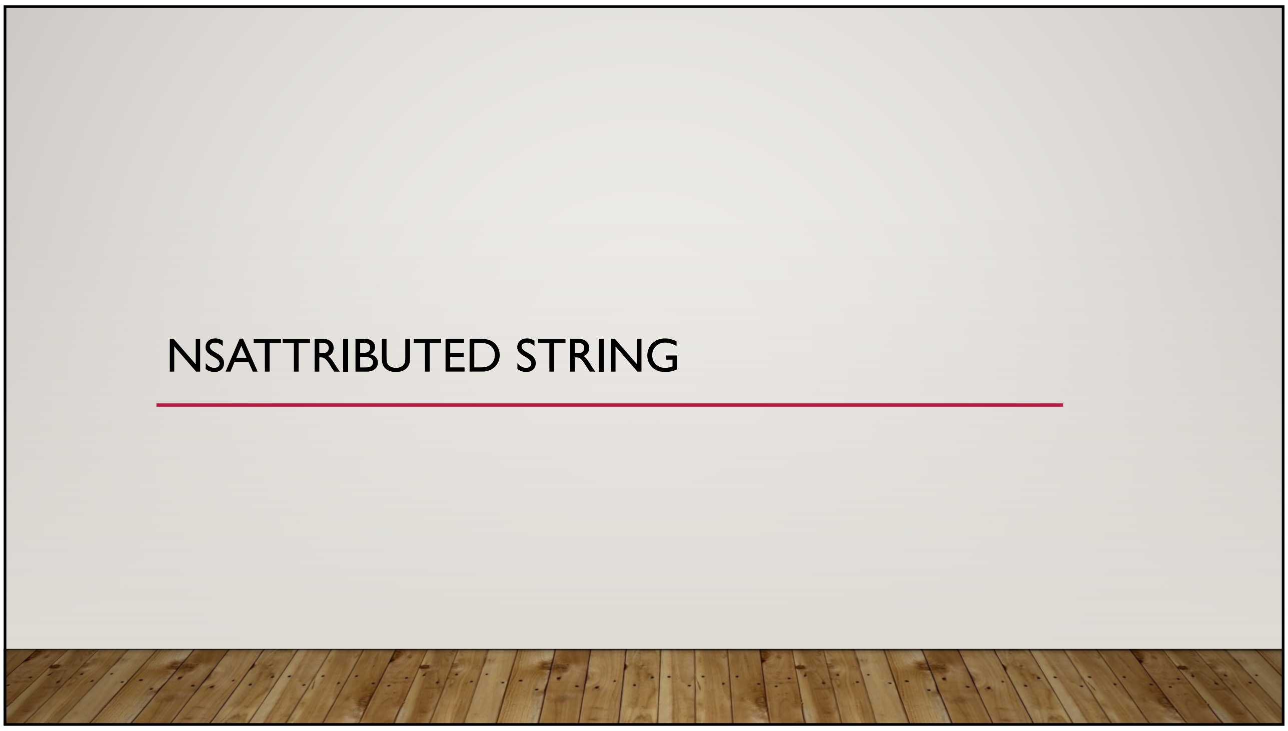
10



新版xCode:
在storyboard介面設定之emoji
只會存在 titleLabel 裡
currentTitle是 nil
→ 改寫程式碼

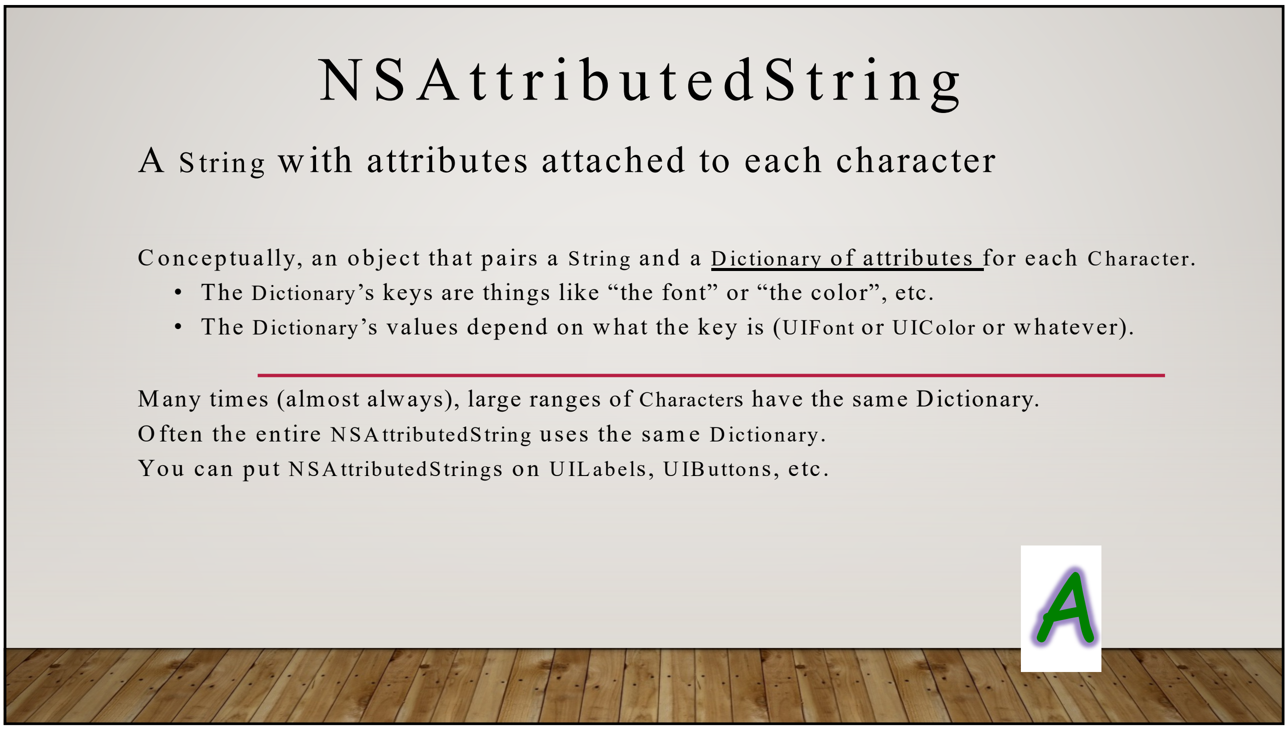
```
        print(sender.titleLabel!.text!)
        var title = ""
        if let tit = sender.titleLabel!.text{
            title = tit
        }

        if (sender.currentTitle == nil)
        {
            sender.setTitle(title, for: UIControl.State.normal)
        }

        if sender.currentTitle == title{
            sender.setTitle("", for: UIControl.State.normal)
            sender.backgroundColor =  colorLiteral(red: 0.9, green: 0.8, blue: 0.4 ,
alpha: 1)
        }else{
            sender.setTitle(title, for: UIControl.State.normal)
            sender.backgroundColor =  colorLiteral(red: 1, green: 1, blue: 1, alpha:
1)
        }
        flipCount += 1
        flipCountLabel.text = "Flips: \(flipCount)"
```

11



NSATTRIBUTED STRING

12



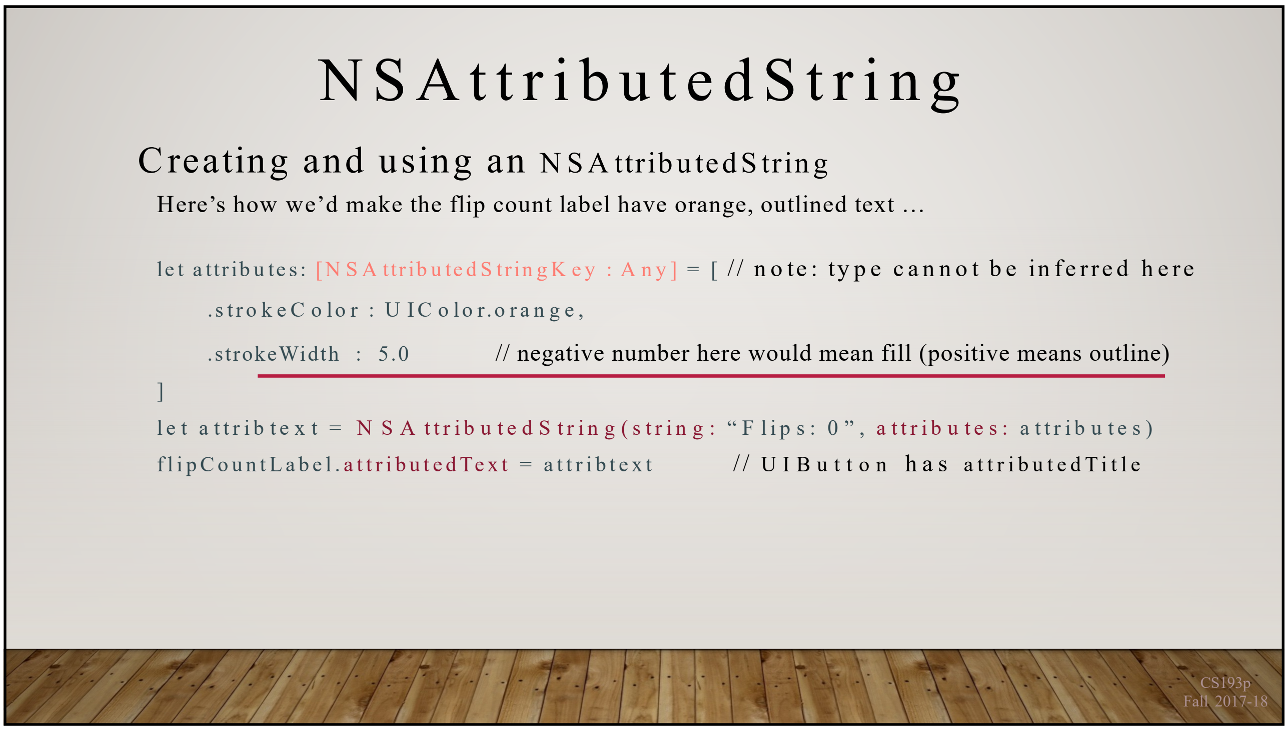
NSAttributedString

A String with attributes attached to each character

Conceptually, an object that pairs a String and a Dictionary of attributes for each Character.
- The Dictionary's keys are things like "the font" or "the color", etc.
- The Dictionary's values depend on what the key is (UIFont or UIColor or whatever).

Many times (almost always), large ranges of Characters have the same Dictionary.
Often the entire NSAttributedString uses the same Dictionary.
You can put NSAttributedStrings on UILabels, UIButtons, etc.

13



NSAttributedString

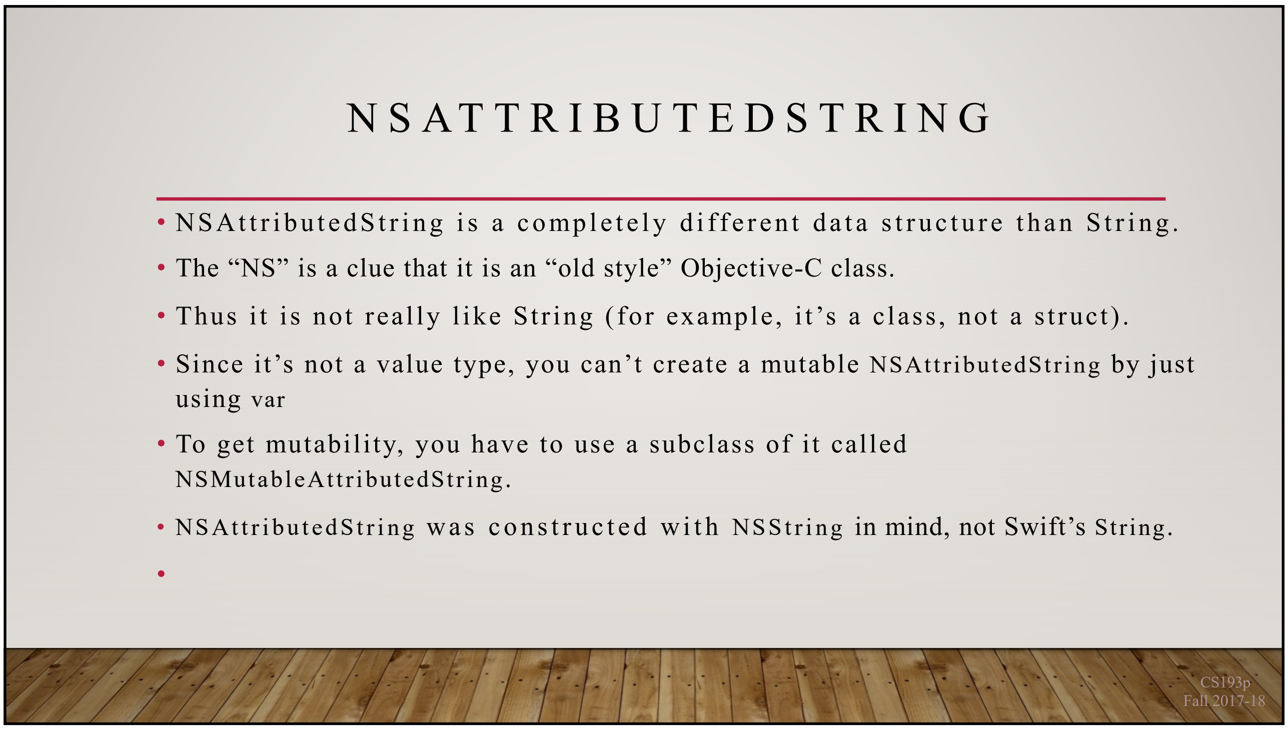
Creating and using an NSAttributedString

Here's how we'd make the flip count label have orange, outlined text ...

```
let attributes: [NSAttributedStringKey : Any] = [ // note: type cannot be inferred here
      .strokeColor : UIColor.orange,
      .strokeWidth  :  5.0          // negative number here would mean fill (positive means outline)
]
let attribtext = NSAttributedString(string: "Flips: 0", attributes: attributes)
flipCountLabel.attributedText = attribtext       // UIButton has attributedTitle
```

CS193p
Fall 2017-18

14

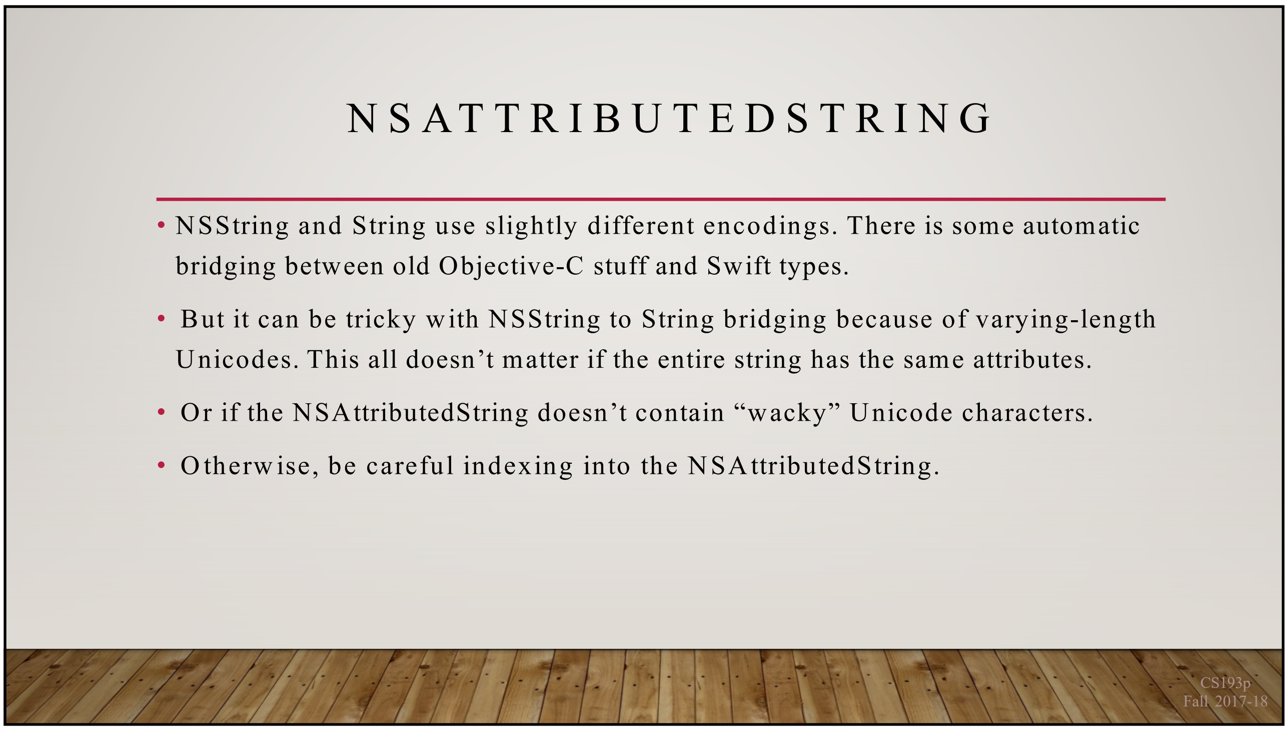


NSATTRIBUTEDSTRING

- NSAttributedString is a completely different data structure than String.
- The “NS” is a clue that it is an “old style” Objective-C class.
- Thus it is not really like String (for example, it’s a class, not a struct).
- Since it’s not a value type, you can’t create a mutable NSAttributedString by just using var
- To get mutability, you have to use a subclass of it called NSMutableAttributedString.
- NSAttributedString was constructed with NSString in mind, not Swift’s String.
-

CS193p
Fall 2017-18

15

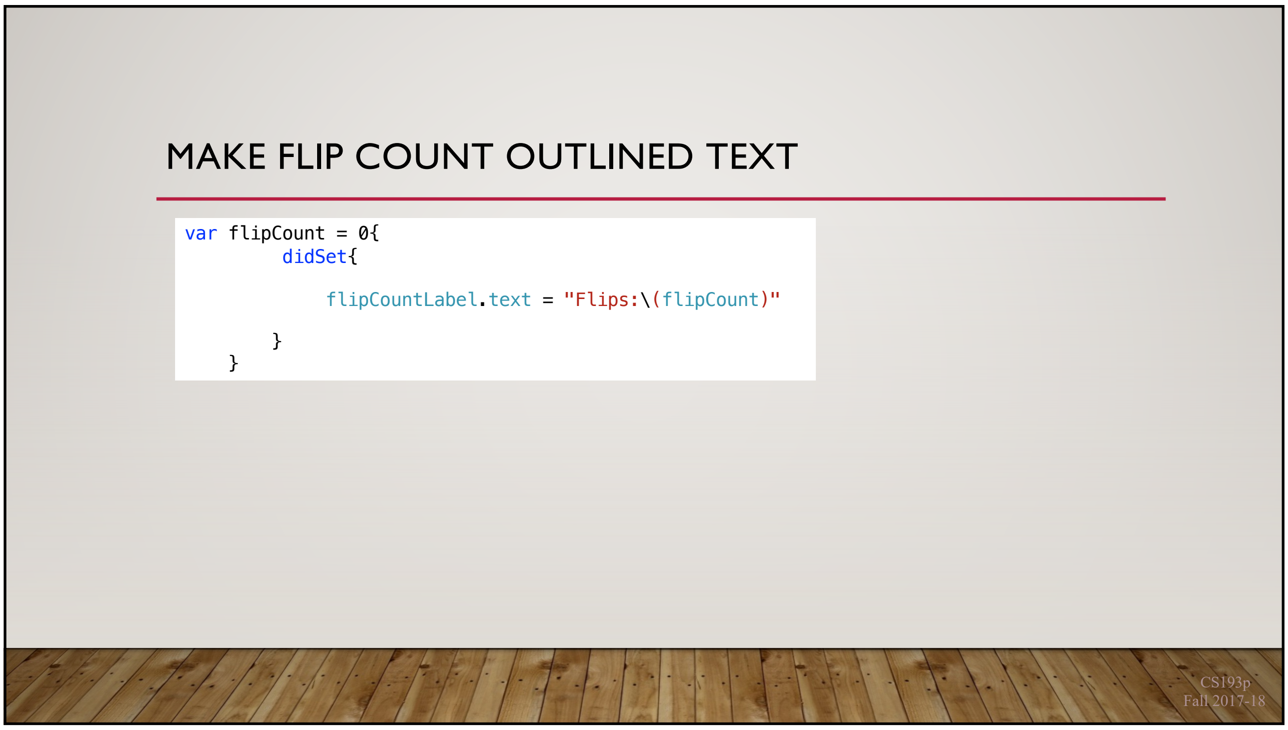


NSATTRIBUTEDSTRING

- NSString and String use slightly different encodings. There is some automatic bridging between old Objective-C stuff and Swift types.
- But it can be tricky with NSString to String bridging because of varying-length Unicodes. This all doesn’t matter if the entire string has the same attributes.
- Or if the NSAttributedString doesn’t contain “wacky” Unicode characters.
- Otherwise, be careful indexing into the NSAttributedString.

CS193p
Fall 2017-18

16

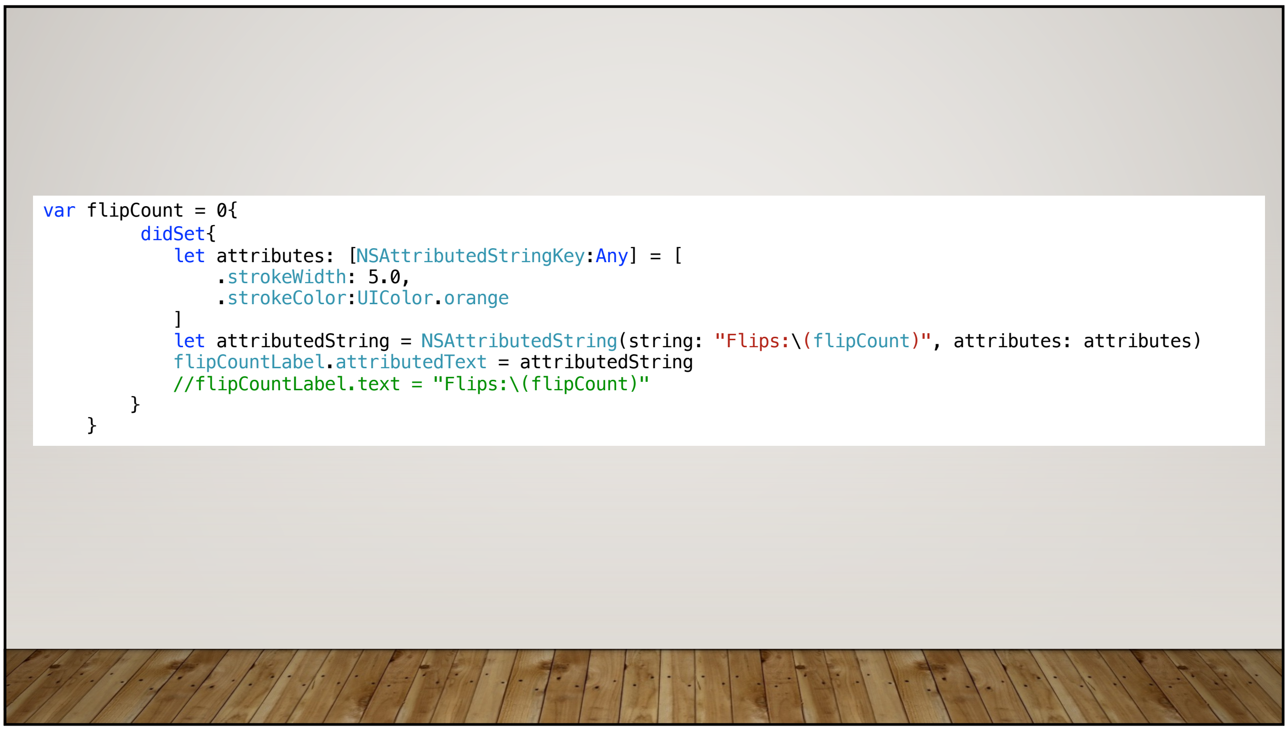


MAKE FLIP COUNT OUTLINED TEXT

```
var flipCount = 0{
         didSet{

             flipCountLabel.text = "Flips:\(flipCount)"

        }
    }
```

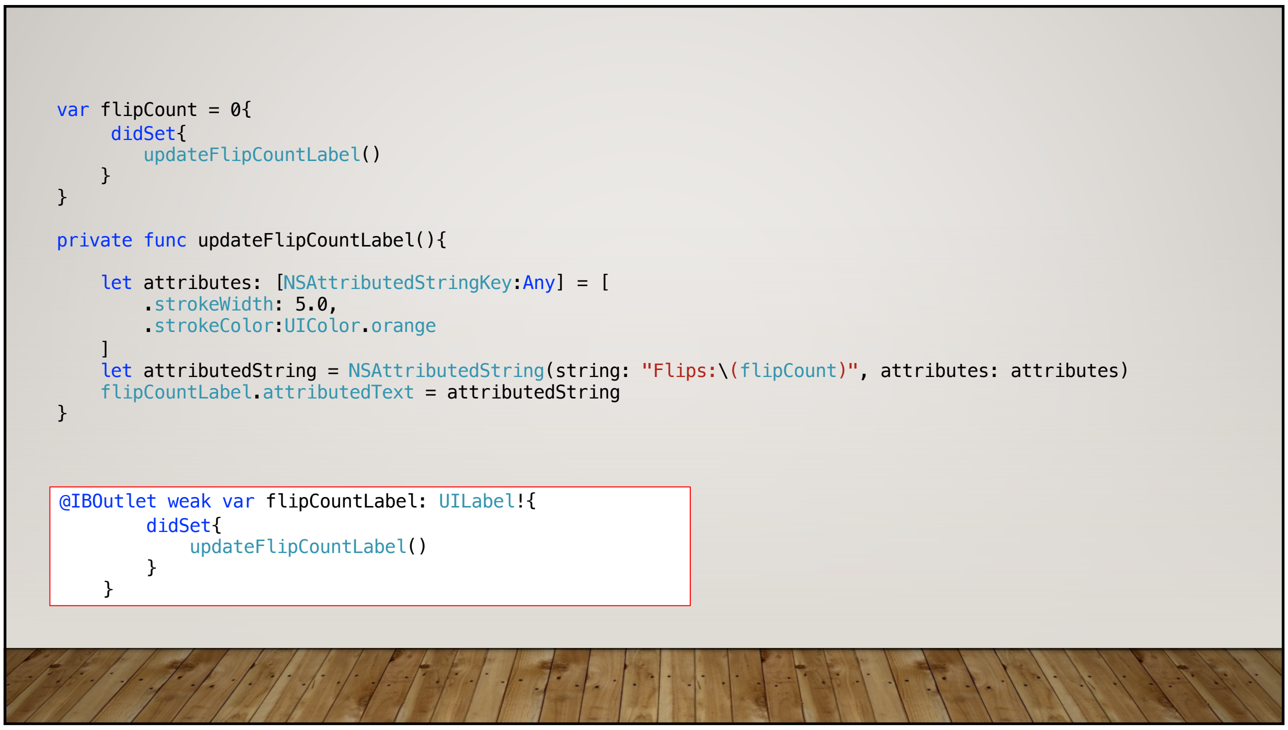
CS193p
Fall 2017-18

17



```
var flipCount = 0{
         didSet{
            let attributes: [NSAttributedStringKey:Any] = [
                .strokeWidth: 5.0,
                .strokeColor:UIColor.orange
            ]
            let attributedString = NSAttributedString(string: "Flips:\(flipCount)", attributes: attributes)
            flipCountLabel.attributedText = attributedString
            //flipCountLabel.text = "Flips:\(flipCount)"
        }
    }
```

18

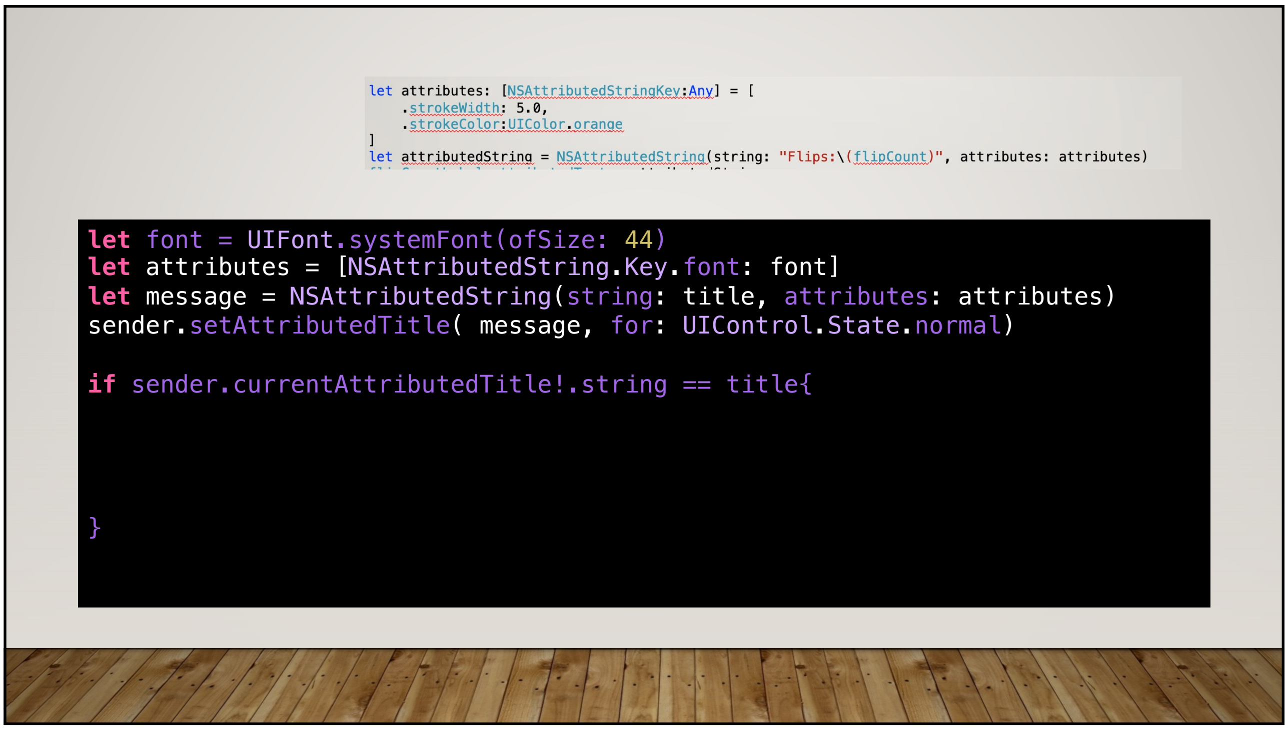


```
var flipCount = 0{
     didSet{
        updateFlipCountLabel()
    }
}

private func updateFlipCountLabel(){

    let attributes: [NSAttributedStringKey:Any] = [
        .strokeWidth: 5.0,
        .strokeColor:UIColor.orange
    ]
    let attributedString = NSAttributedString(string: "Flips:\(flipCount)", attributes: attributes)
    flipCountLabel.attributedText = attributedString
}


@IBOutlet weak var flipCountLabel: UILabel!{
        didSet{
            updateFlipCountLabel()
        }
    }
```

19

BUTTON + ATTRIBUTEDSTRING

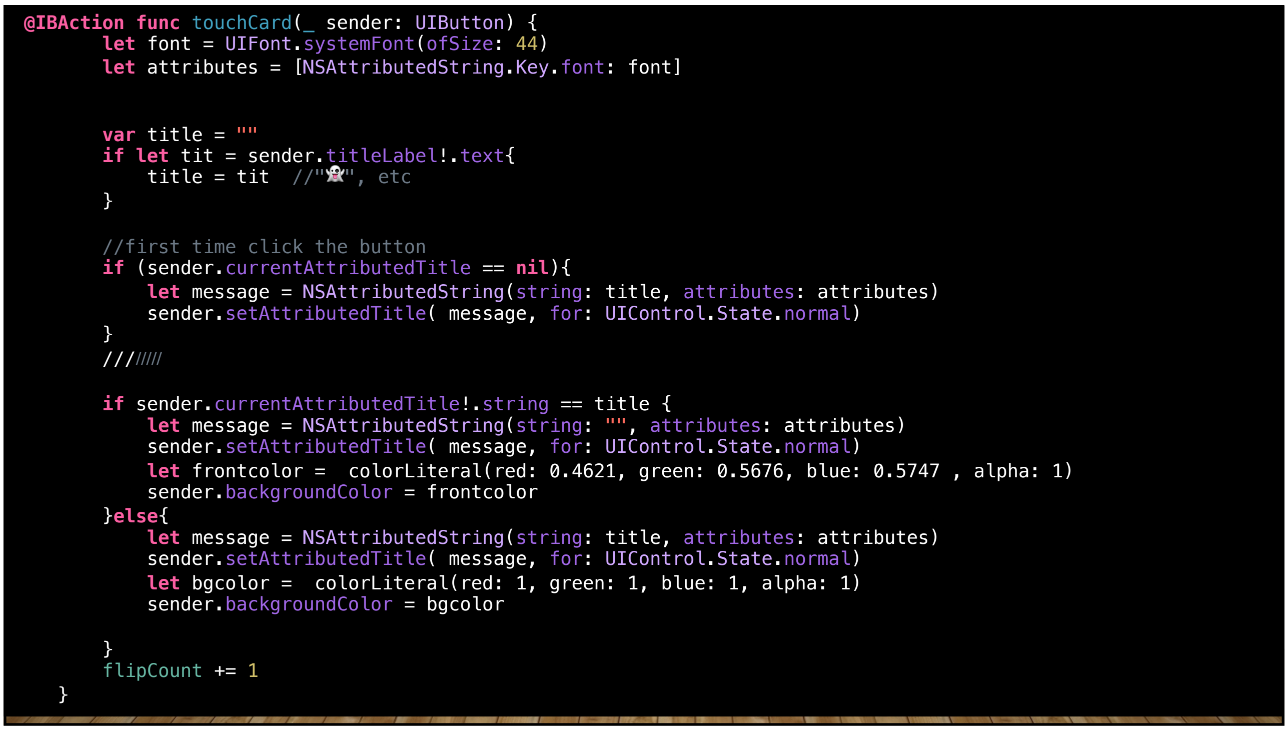
20



```
let attributes: [NSAttributedStringKey:Any] = [
    .strokeWidth: 5.0,
    .strokeColor:UIColor.orange
]
let attributedString = NSAttributedString(string: "Flips:\(flipCount)", attributes: attributes)
```

```
let font = UIFont.systemFont(ofSize: 44)
let attributes = [NSAttributedString.Key.font: font]
let message = NSAttributedString(string: title, attributes: attributes)
sender.setAttributedTitle( message, for: UIControl.State.normal)

if sender.currentAttributedTitle!.string == title{




}
```

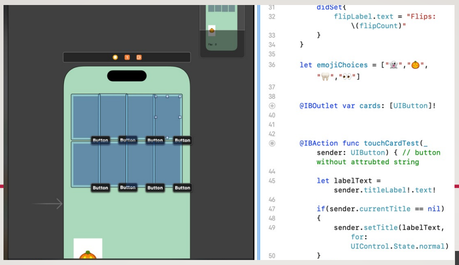
21



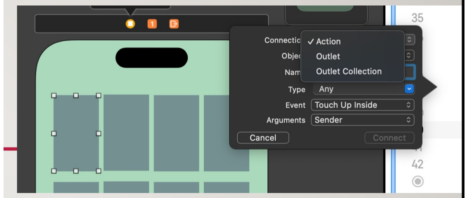
```
@IBAction func touchCard(_ sender: UIButton) {
        let font = UIFont.systemFont(ofSize: 44)
        let attributes = [NSAttributedString.Key.font: font]


        var title = ""
        if let tit = sender.titleLabel!.text{
            title = tit  //"👻", etc
        }

        //first time click the button
        if (sender.currentAttributedTitle == nil){
            let message = NSAttributedString(string: title, attributes: attributes)
            sender.setAttributedTitle( message, for: UIControl.State.normal)
        }
        ///////

        if sender.currentAttributedTitle!.string == title {
            let message = NSAttributedString(string: "", attributes: attributes)
            sender.setAttributedTitle( message, for: UIControl.State.normal)
            let frontcolor =  colorLiteral(red: 0.4621, green: 0.5676, blue: 0.5747 , alpha: 1)
            sender.backgroundColor = frontcolor
        }else{
            let message = NSAttributedString(string: title, attributes: attributes)
            sender.setAttributedTitle( message, for: UIControl.State.normal)
            let bgcolor =  colorLiteral(red: 1, green: 1, blue: 1, alpha: 1)
            sender.backgroundColor = bgcolor

        }
        flipCount += 1
    }
```

22

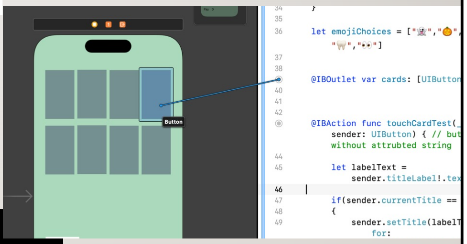
EMOJIS



```
        didSet{
            flipLabel.text = "Flips:
                \(flipCount)"
        }
    }

    let emojiChoices = ["👻","🎃",
        "🦷","👀"]


    @IBOutlet var cards: [UIButton]!



    @IBAction func touchCardTest(_
        sender: UIButton) { // button
        without attrubted string

        let labelText =
            sender.titleLabel!.text!

        if(sender.currentTitle == nil)
        {
            sender.setTitle(labelText,
                for:
                UIControl.State.normal)
        }
```



Type: Any
Event: Touch Up Inside
Arguments: Sender



```
    }

    let emojiChoices = ["👻","🎃",
        "🦷","👀"]


    @IBOutlet var cards: [UIButton



    @IBAction func touchCardTest(_
        sender: UIButton) { // but
        without attrubted string

        let labelText =
            sender.titleLabel!.tex

        if(sender.currentTitle ==
        {
            sender.setTitle(labelT
                for:
```

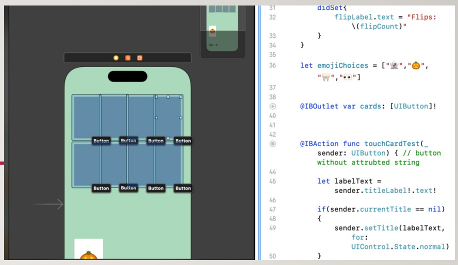
```
let emojiChoices = ["👻","🎃”, "👻","🎃"]
```

```
@IBOutlet var cards: [UIButton]!
```

```
  if let id = cards.firstIndex(of: sender){
                print(id)
                title = emojiChoices[id]
        }
```

23

EMOJIS



```
        didSet{
            flipLabel.text = "Flips:
                \(flipCount)"
        }
    }

    let emojiChoices = ["👻","🎃",
        "🦷","👀"]


    @IBOutlet var cards: [UIButton]!



    @IBAction func touchCardTest(_
        sender: UIButton) { // button
        without attrubted string

        let labelText =
            sender.titleLabel!.text!

        if(sender.currentTitle == nil)
        {
            sender.setTitle(labelText,
                for:
                UIControl.State.normal)
        }
```

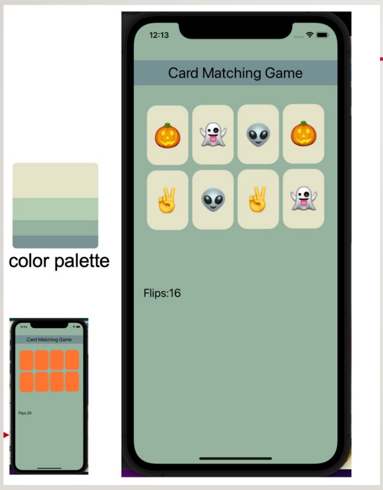
```
let emojiChoices = ["👻","🎃”, "👻","🎃"]
```

```
@IBOutlet var cards: [UIButton]!
```

```
  if let id = cards.firstIndex(of: sender){
                print(id)
                title = emojiChoices[id]
        }
```

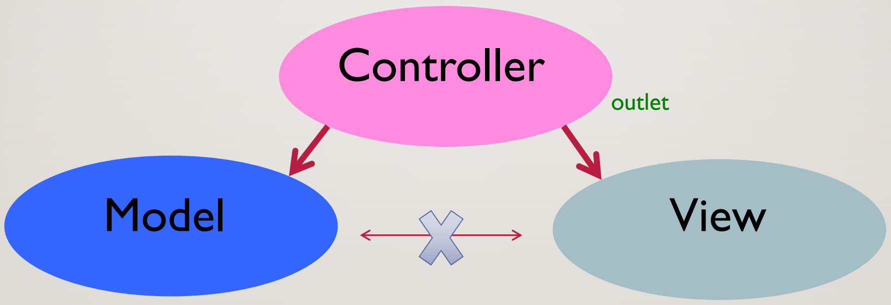
24

LAB02



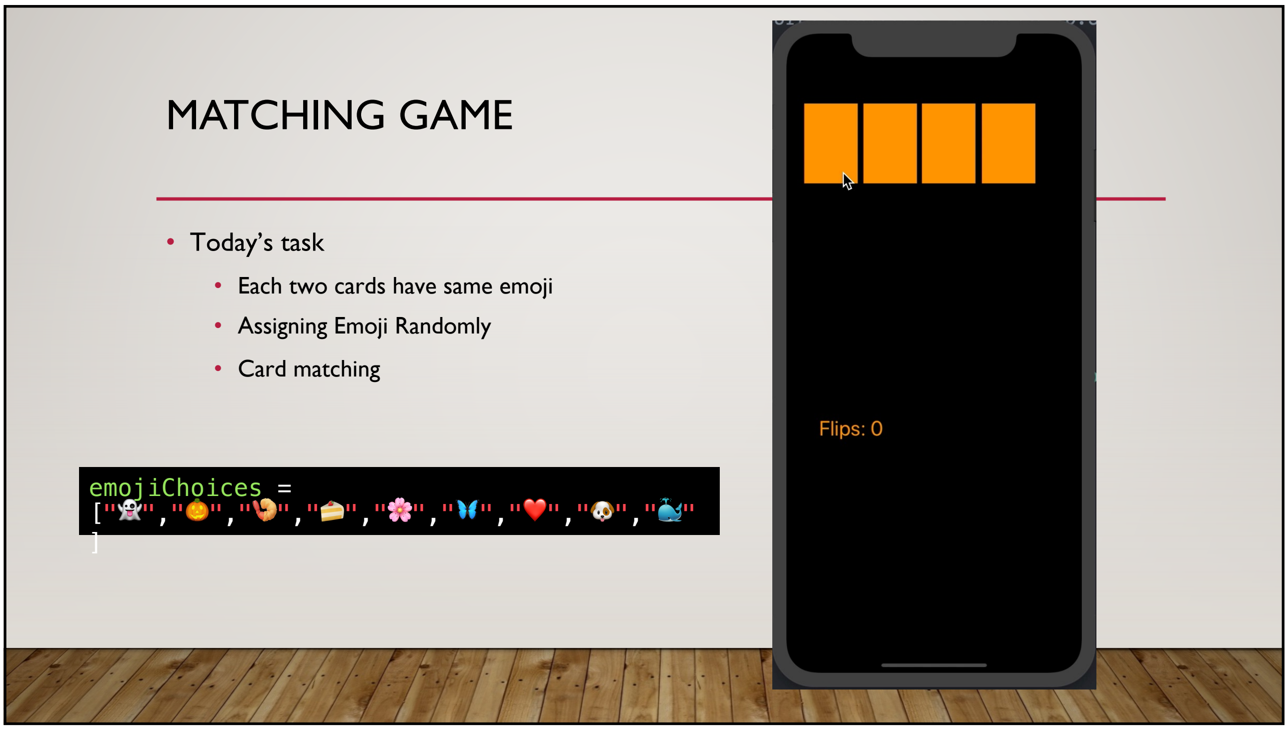
25

MVC



- Controllers can always talk directly to their Model
- Controllers can also talk directly to their View
- The Model and View should never speak to each other

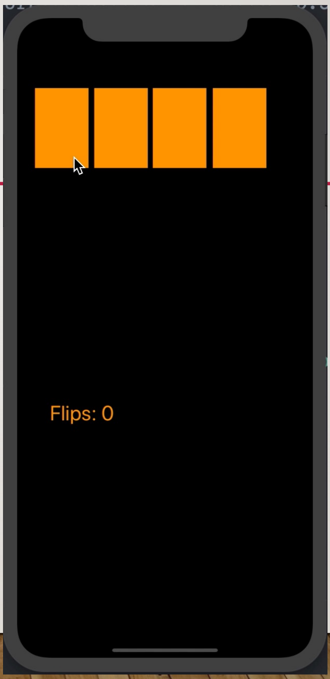
26



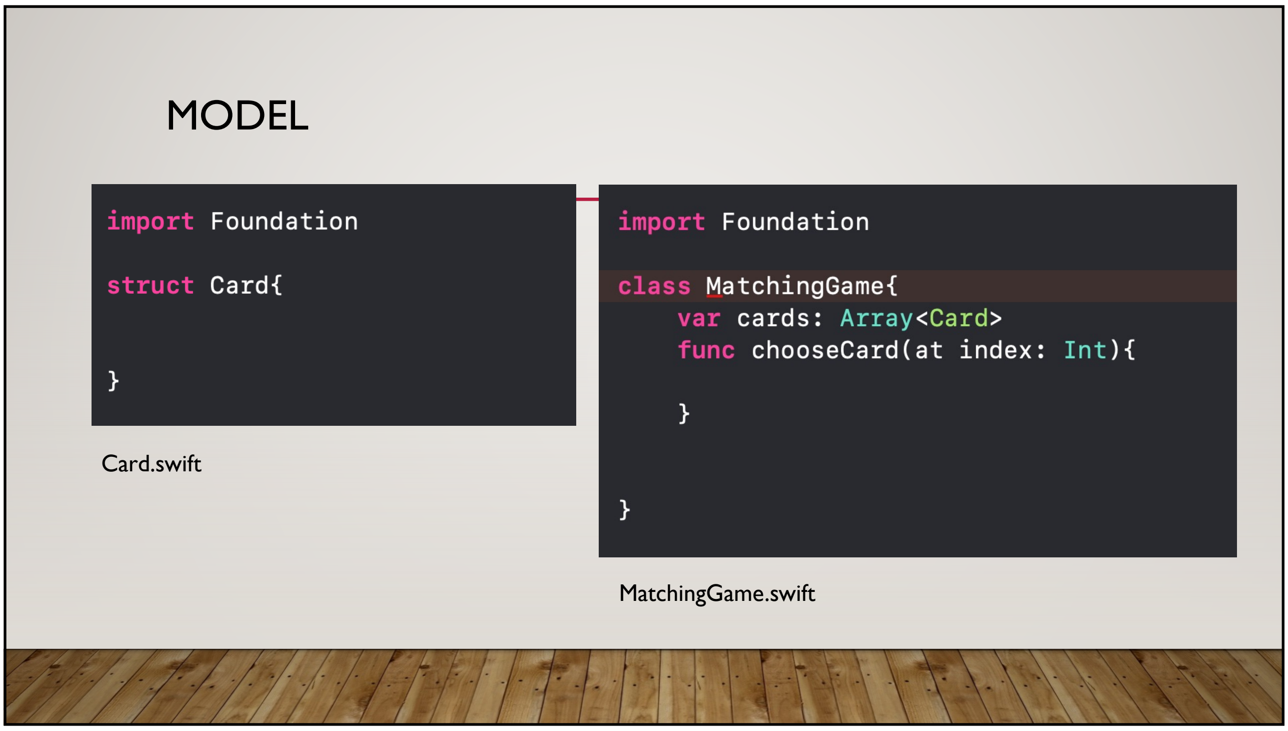
MATCHING GAME

- Today's task
  - Each two cards have same emoji
  - Assigning Emoji Randomly
  - Card matching

```
emojiChoices = ["👻","🎃","🦐","🍰","🌸","🦋","❤️","🐶","🐳"]
```



27



MODEL

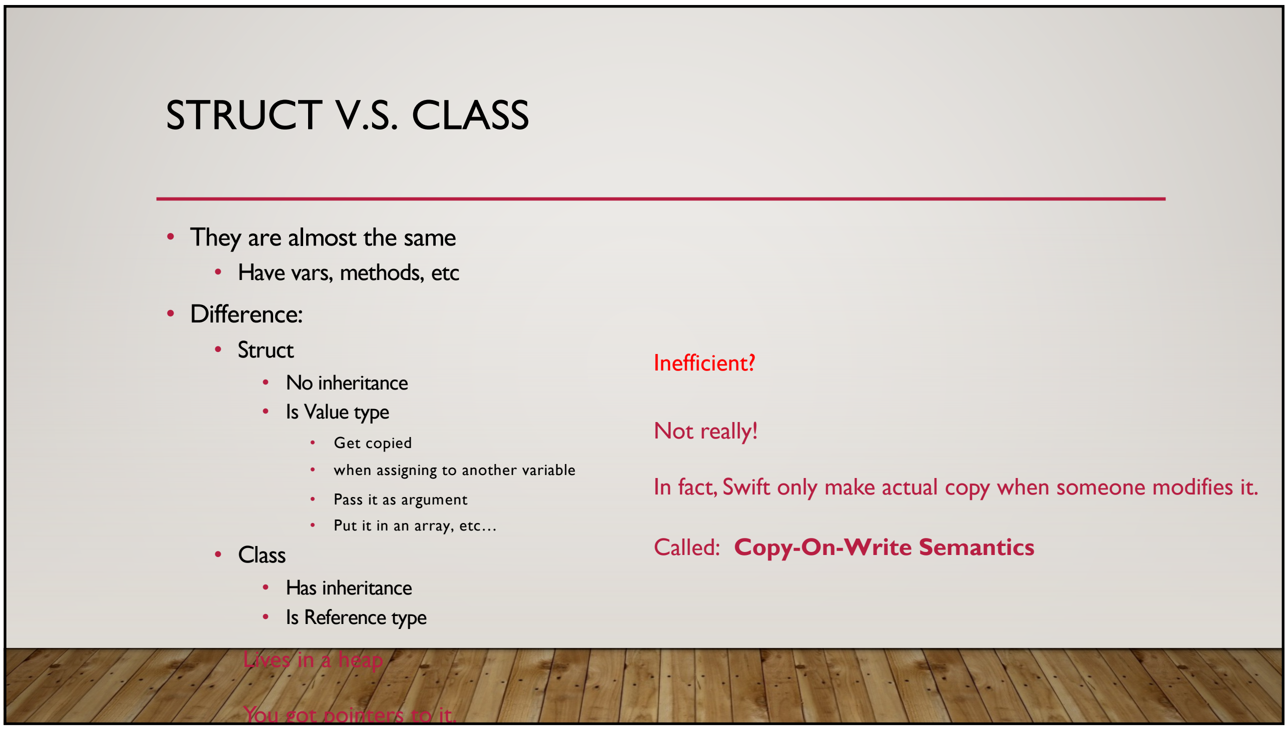
```swift
import Foundation

struct Card{


}
```

Card.swift

```swift
import Foundation

class MatchingGame{
    var cards: Array<Card>
    func chooseCard(at index: Int){

    }


}
```

MatchingGame.swift

28



STRUCT V.S. CLASS

- They are almost the same
  - Have vars, methods, etc
- Difference:
  - Struct
    - No inheritance
    - Is Value type
      - Get copied
      - when assigning to another variable
      - Pass it as argument
      - Put it in an array, etc…
  - Class
    - Has inheritance
    - Is Reference type

Lives in a heap

You got pointers to it.

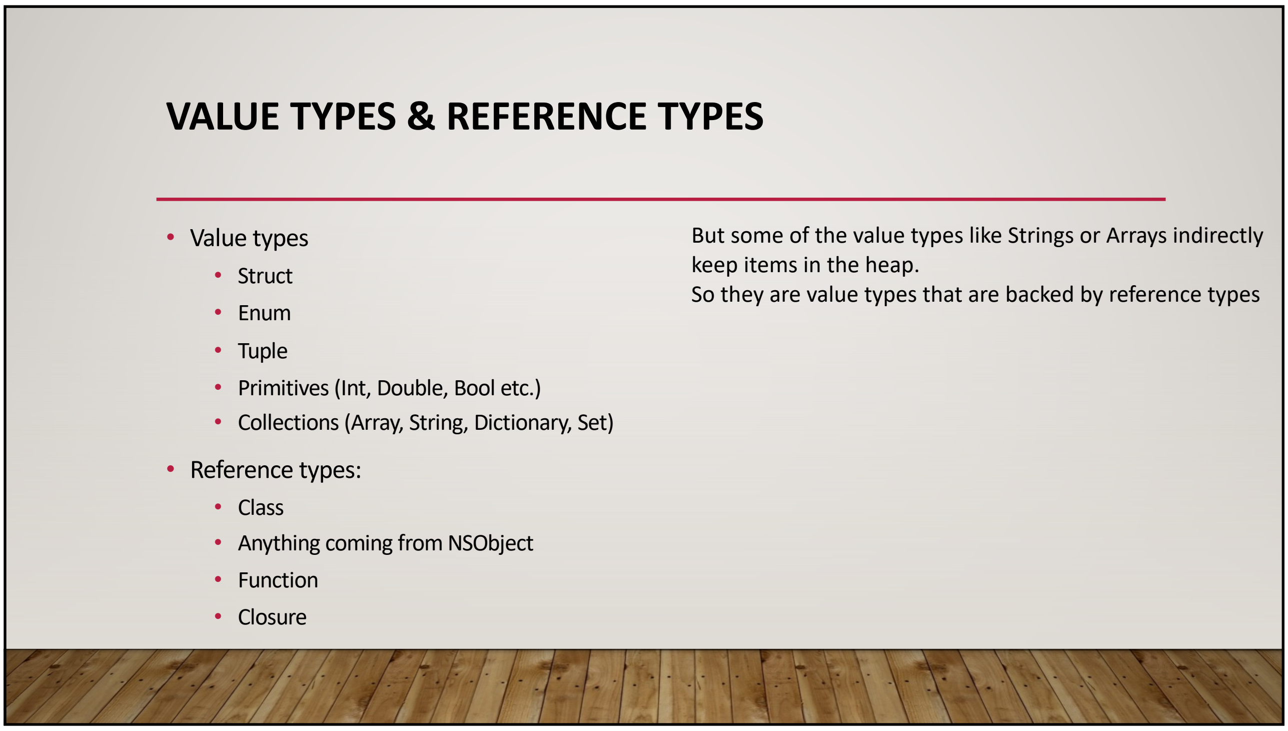
Inefficient?

Not really!

In fact, Swift only make actual copy when someone modifies it.

Called: **Copy-On-Write Semantics**

29



VALUE TYPES & REFERENCE TYPES

- Value types
  - Struct
  - Enum
  - Tuple
  - Primitives (Int, Double, Bool etc.)
  - Collections (Array, String, Dictionary, Set)
- Reference types:
  - Class
  - Anything coming from NSObject
  - Function
  - Closure

But some of the value types like Strings or Arrays indirectly keep items in the heap.
So they are value types that are backed by reference types

30

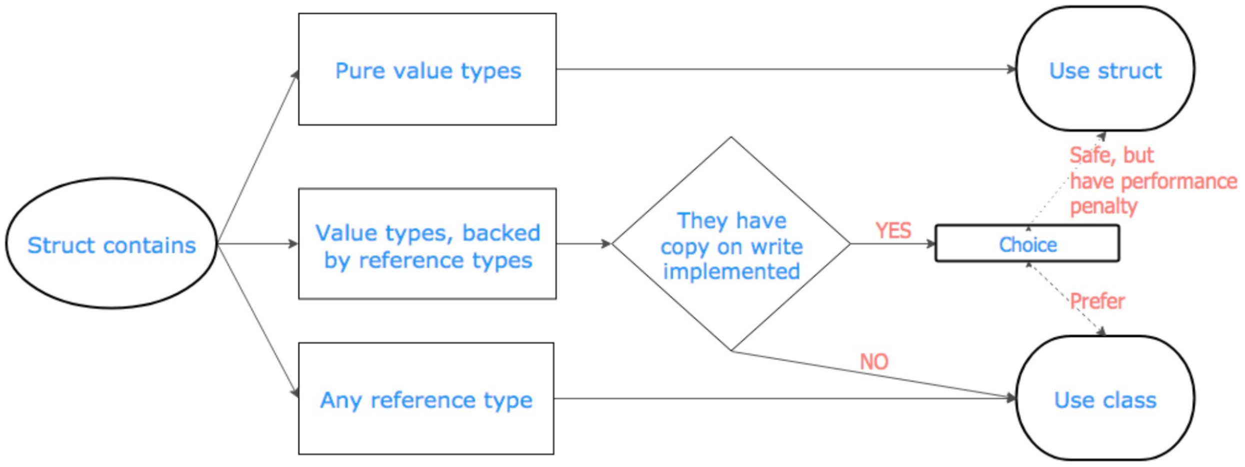
CARD

```swift
import Foundation


struct Card{

    var isFaceUp = false
    var isMatched = false
    var identifier:Int      //use ID, not emoji

}
```

Card.swift

31

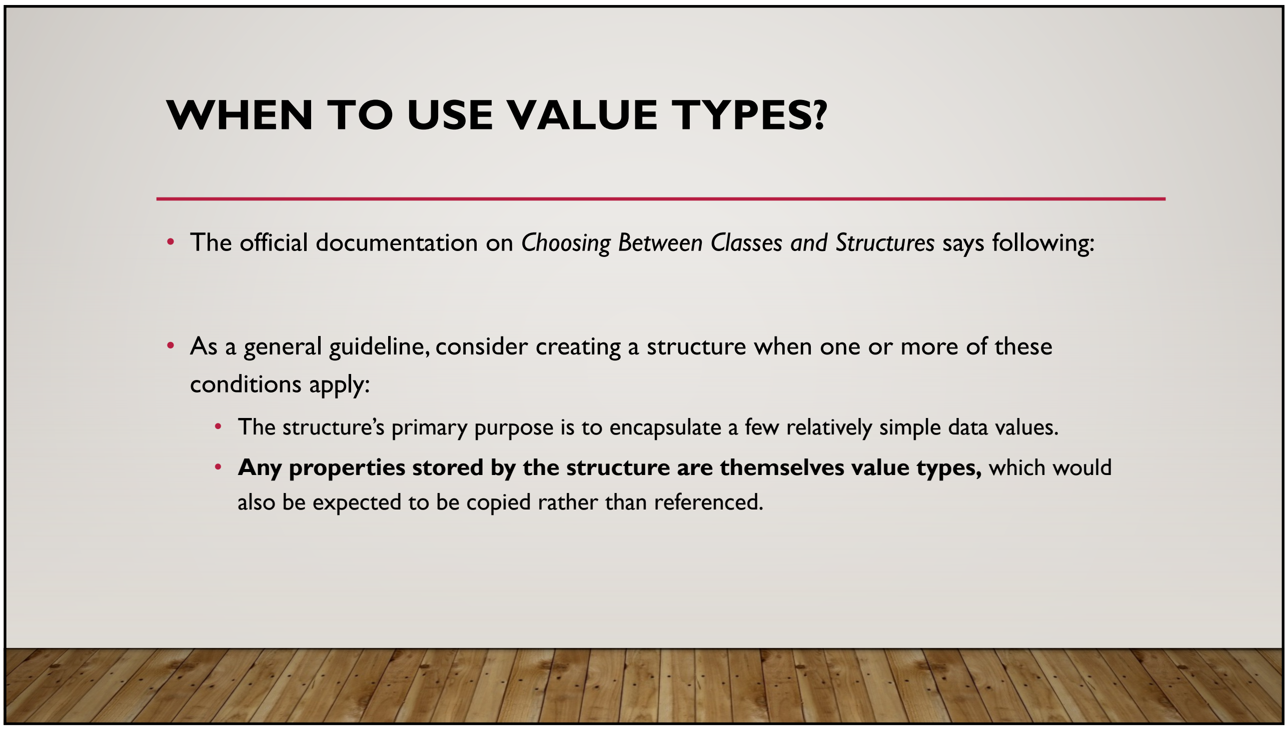
WHEN TO USE VALUE TYPES?



Struct contains
Pure value types
Value types, backed by reference types
Any reference type
They have copy on write implemented
YES
NO
Choice
Safe, but have performance penalty
Prefer
Use struct
Use class

https://medium.com/commencis/stop-using-structs-e1be9a86376f

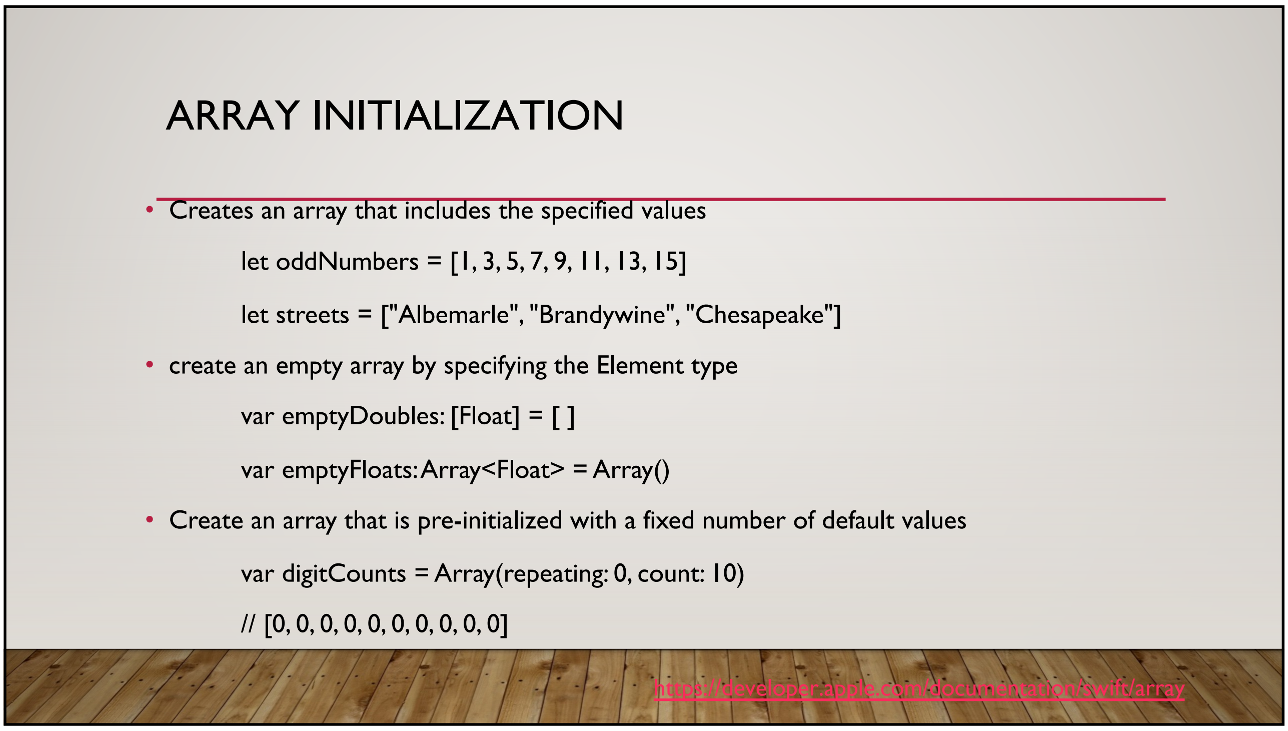
32



WHEN TO USE VALUE TYPES?

- The official documentation on *Choosing Between Classes and Structures* says following:
- As a general guideline, consider creating a structure when one or more of these conditions apply:
  - The structure's primary purpose is to encapsulate a few relatively simple data values.
  - **Any properties stored by the structure are themselves value types,** which would also be expected to be copied rather than referenced.

33



ARRAY INITIALIZATION

- Creates an array that includes the specified values

```
let oddNumbers = [1, 3, 5, 7, 9, 11, 13, 15]
let streets = ["Albemarle", "Brandywine", "Chesapeake"]
```

- create an empty array by specifying the Element type

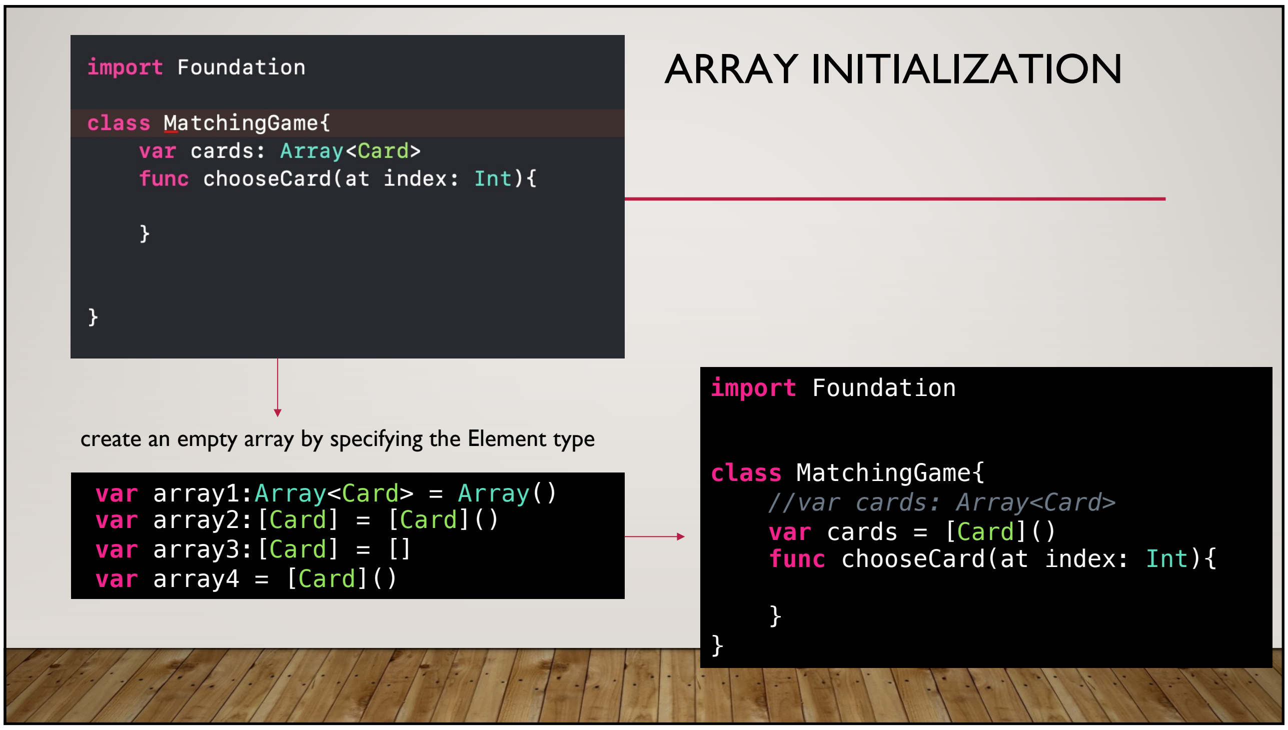
```
var emptyDoubles: [Float] = [ ]
var emptyFloats: Array<Float> = Array()
```

- Create an array that is pre-initialized with a fixed number of default values

```
var digitCounts = Array(repeating: 0, count: 10)
// [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

https://developer.apple.com/documentation/swift/array

34



ARRAY INITIALIZATION

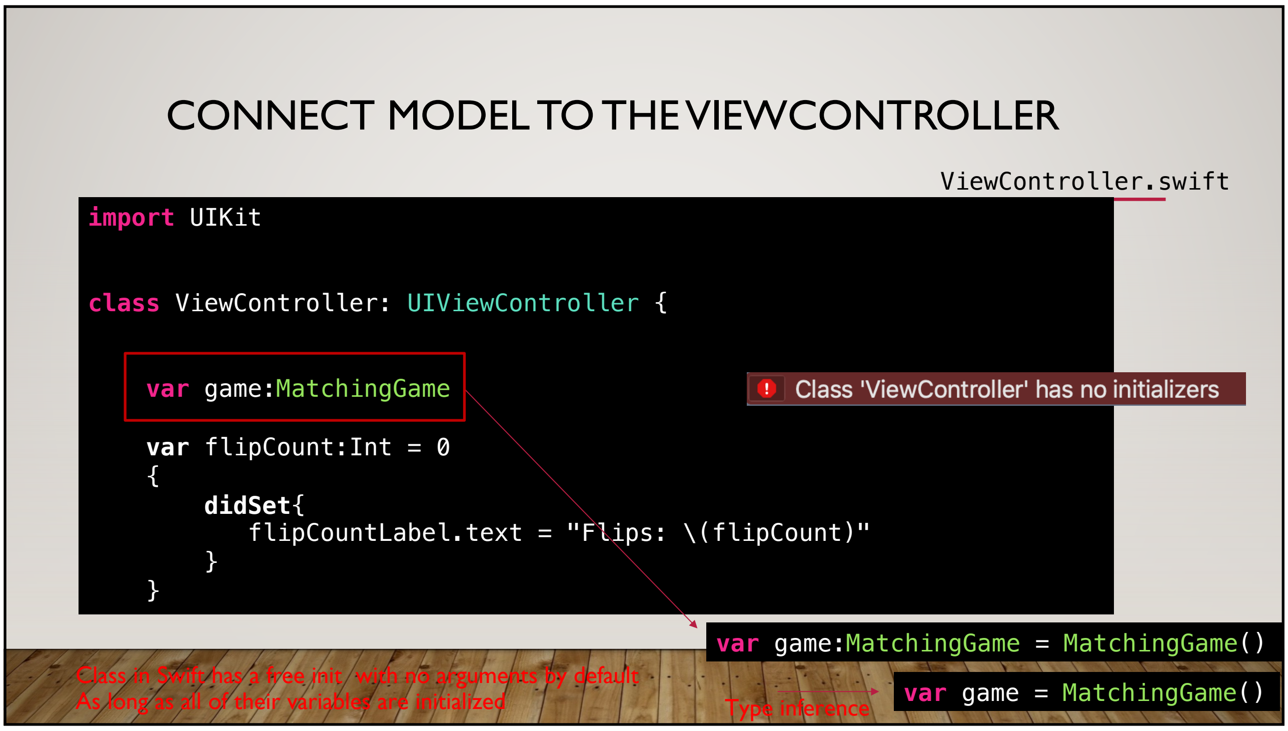
```
import Foundation

class MatchingGame{
    var cards: Array<Card>
    func chooseCard(at index: Int){

    }


}
```

create an empty array by specifying the Element type

```
var array1:Array<Card> = Array()
var array2:[Card] = [Card]()
var array3:[Card] = []
var array4 = [Card]()
```

```
import Foundation


class MatchingGame{
    //var cards: Array<Card>
    var cards = [Card]()
    func chooseCard(at index: Int){

    }
}
```

35



CONNECT MODEL TO THE VIEWCONTROLLER

ViewController.swift

```
import UIKit


class ViewController: UIViewController {

    var game:MatchingGame

    var flipCount:Int = 0
    {
        didSet{
            flipCountLabel.text = "Flips: \(flipCount)"
        }
    }
```

Class 'ViewController' has no initializers

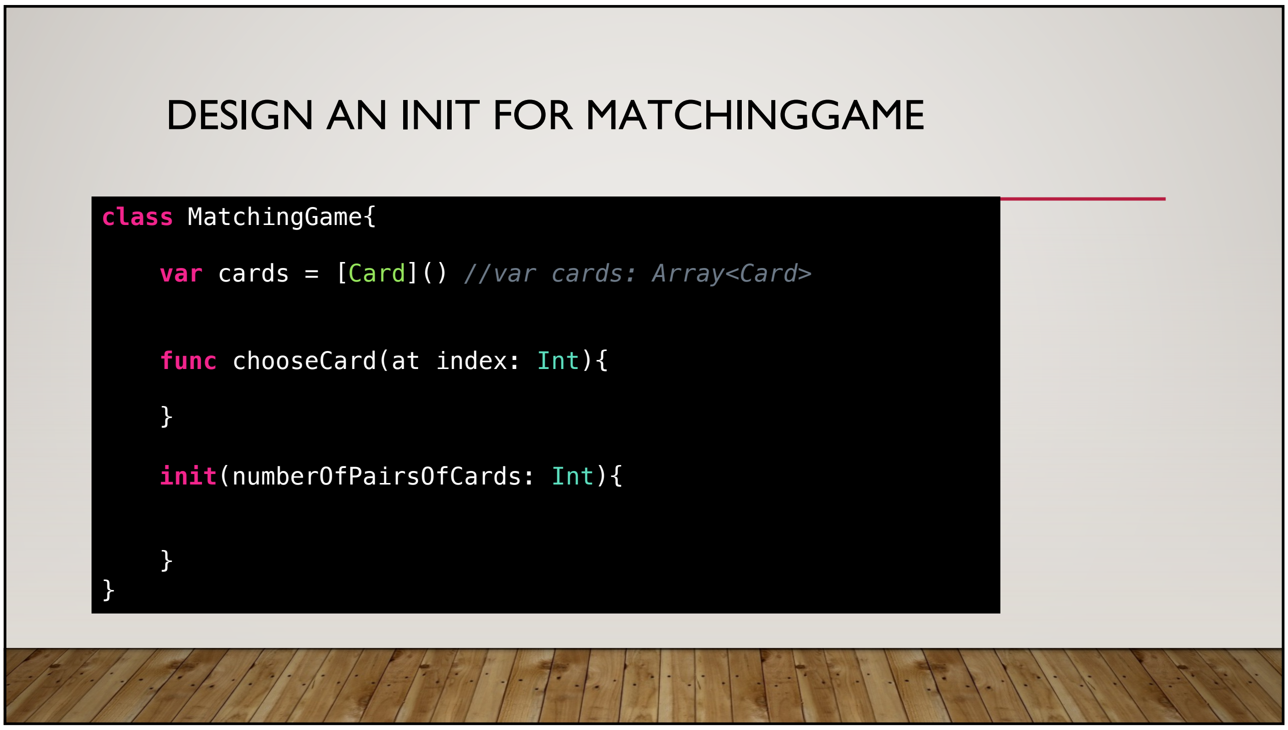
```
var game:MatchingGame = MatchingGame()
```

```
var game = MatchingGame()
```

Class in Swift has a free init with no arguments by default
As long as all of their variables are initialized

Type inference

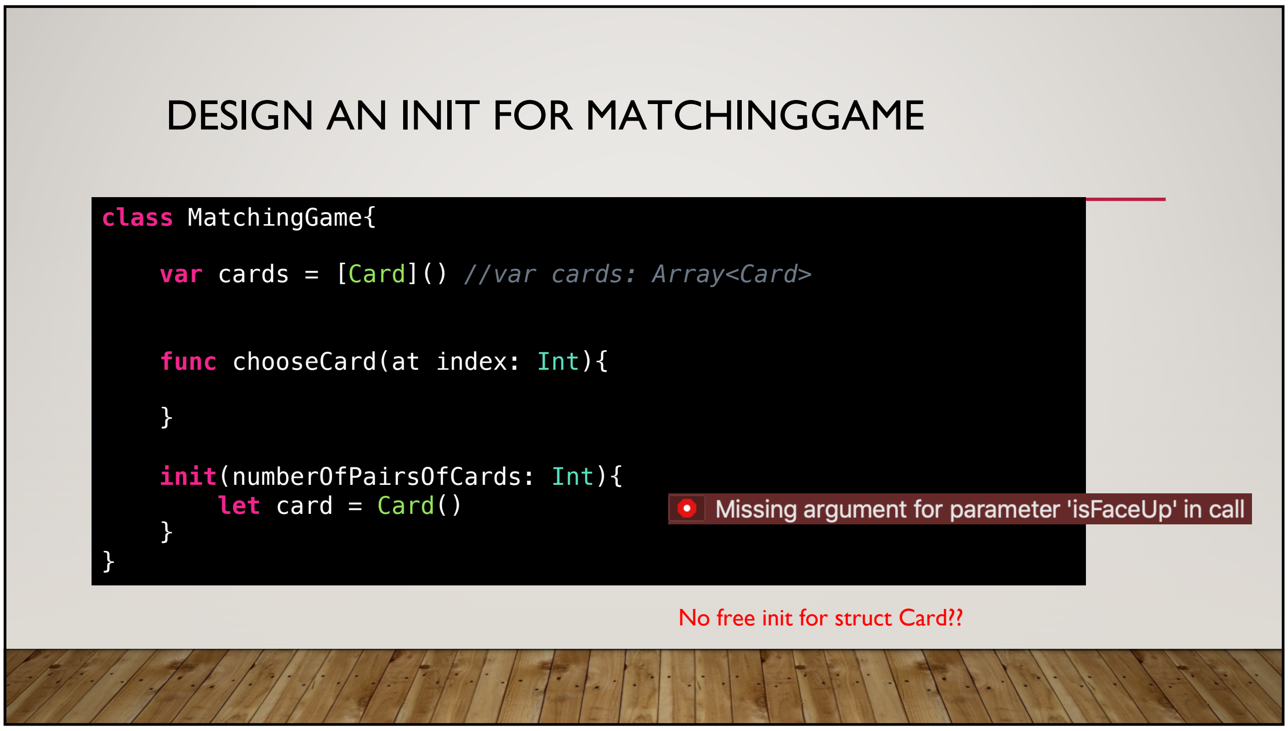
36



DESIGN AN INIT FOR MATCHINGGAME

```
class MatchingGame{
    var cards = [Card]() //var cards: Array<Card>

    func chooseCard(at index: Int){
    }
    init(numberOfPairsOfCards: Int){

    }
}
```

37



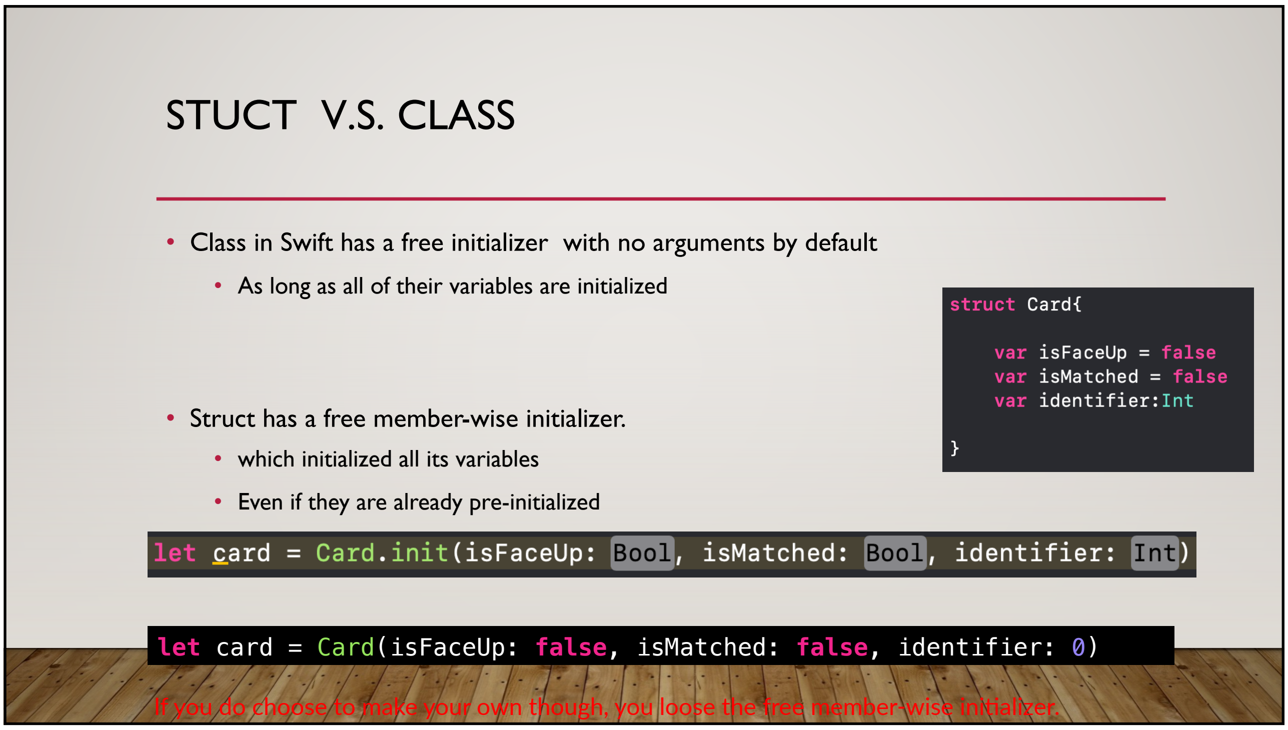
DESIGN AN INIT FOR MATCHINGGAME

```
class MatchingGame{
    var cards = [Card]() //var cards: Array<Card>

    func chooseCard(at index: Int){
    }
    init(numberOfPairsOfCards: Int){
        let card = Card()
    }
}
```

● Missing argument for parameter 'isFaceUp' in call

No free init for struct Card??

38



STUCT V.S. CLASS

- Class in Swift has a free initializer with no arguments by default
  - As long as all of their variables are initialized
- Struct has a free member-wise initializer.
  - which initialized all its variables
  - Even if they are already pre-initialized

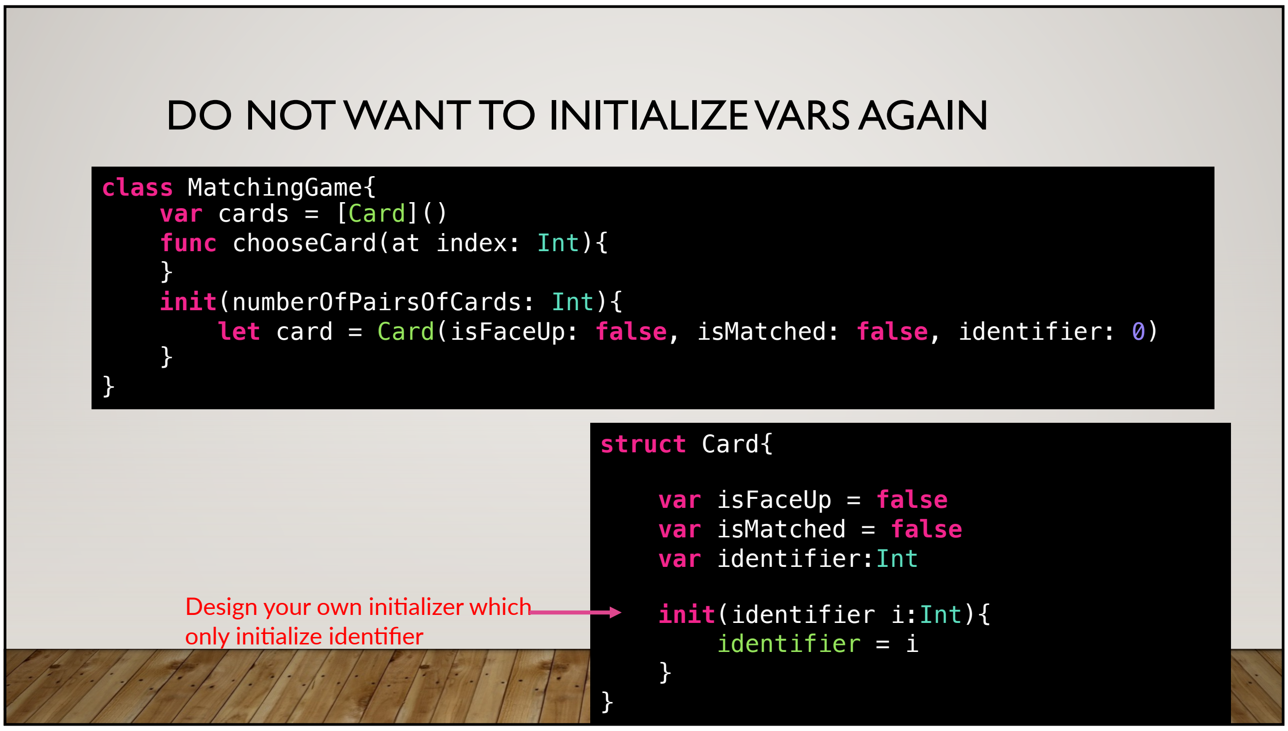
```
struct Card{

    var isFaceUp = false
    var isMatched = false
    var identifier:Int

}
```

```
let card = Card.init(isFaceUp: Bool, isMatched: Bool, identifier: Int)
```

```
let card = Card(isFaceUp: false, isMatched: false, identifier: 0)
```

If you do choose to make your own though, you loose the free member-wise initializer.

39



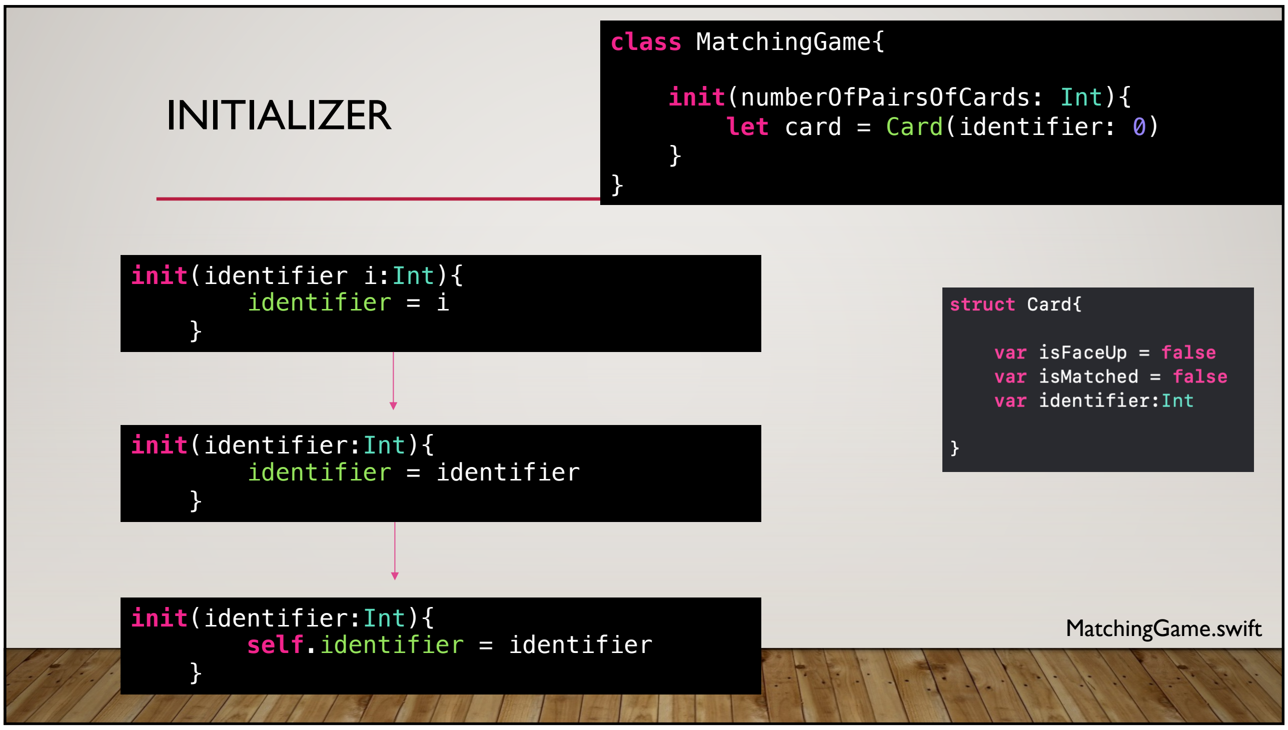
DO NOT WANT TO INITIALIZE VARS AGAIN

```
class MatchingGame{
    var cards = [Card]()
    func chooseCard(at index: Int){
    }
    init(numberOfPairsOfCards: Int){
        let card = Card(isFaceUp: false, isMatched: false, identifier: 0)
    }
}
```

```
struct Card{

    var isFaceUp = false
    var isMatched = false
    var identifier:Int

    init(identifier i:Int){
        identifier = i
    }
}
```

Design your own initializer which only initialize identifier

41



INITIALIZER

```
class MatchingGame{

    init(numberOfPairsOfCards: Int){
        let card = Card(identifier: 0)
    }
}
```

```
init(identifier i:Int){
        identifier = i
    }
```

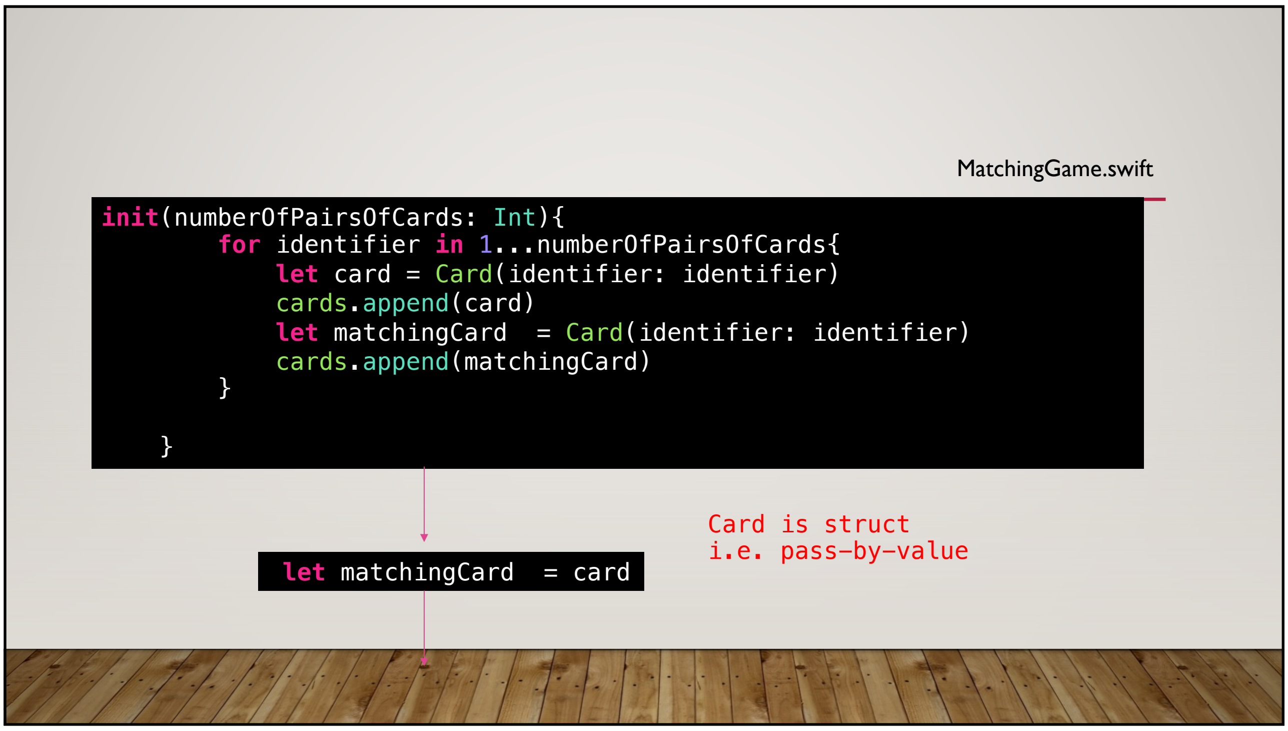
```
init(identifier:Int){
        identifier = identifier
    }
```

```
init(identifier:Int){
        self.identifier = identifier
    }
```

```
struct Card{

    var isFaceUp = false
    var isMatched = false
    var identifier:Int

}
```

MatchingGame.swift

42



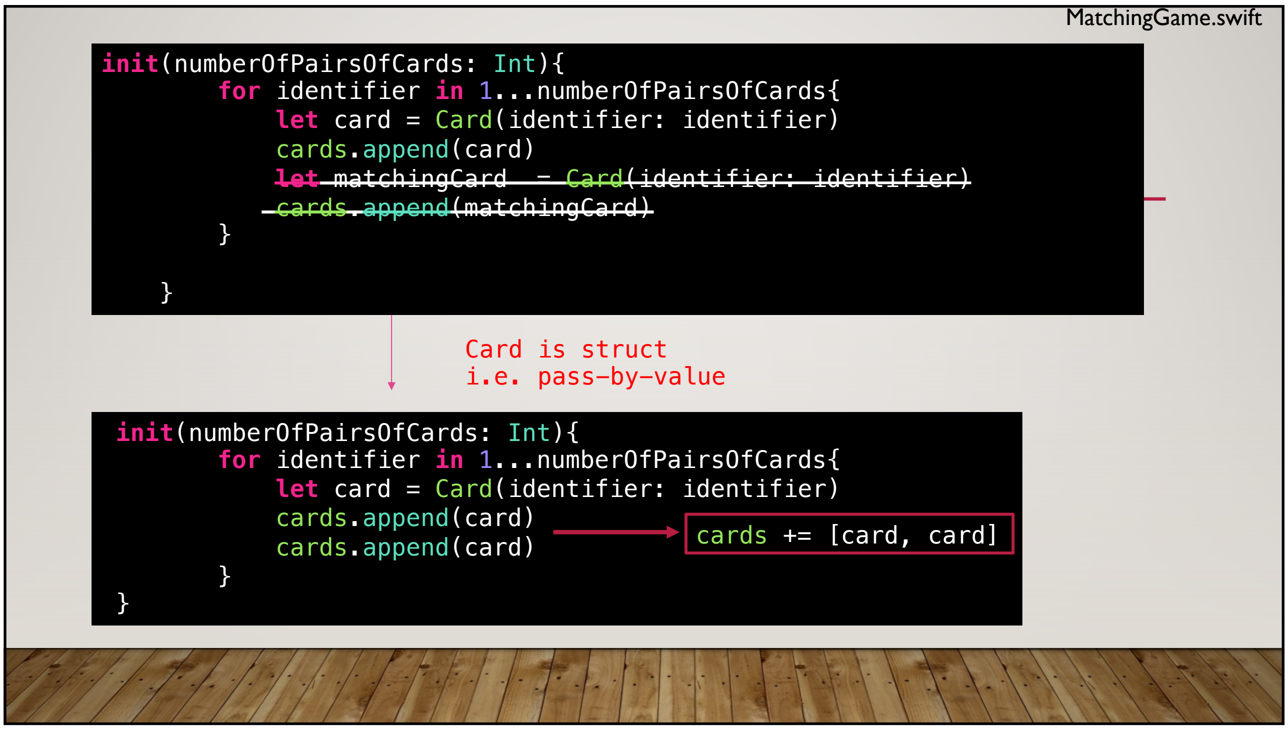
MatchingGame.swift

```
init(numberOfPairsOfCards: Int){
        for identifier in 1...numberOfPairsOfCards{
            let card = Card(identifier: identifier)
            cards.append(card)
            let matchingCard  = Card(identifier: identifier)
            cards.append(matchingCard)
        }

    }
```

```
let matchingCard  = card
```

Card is struct
i.e. pass-by-value

43



MatchingGame.swift

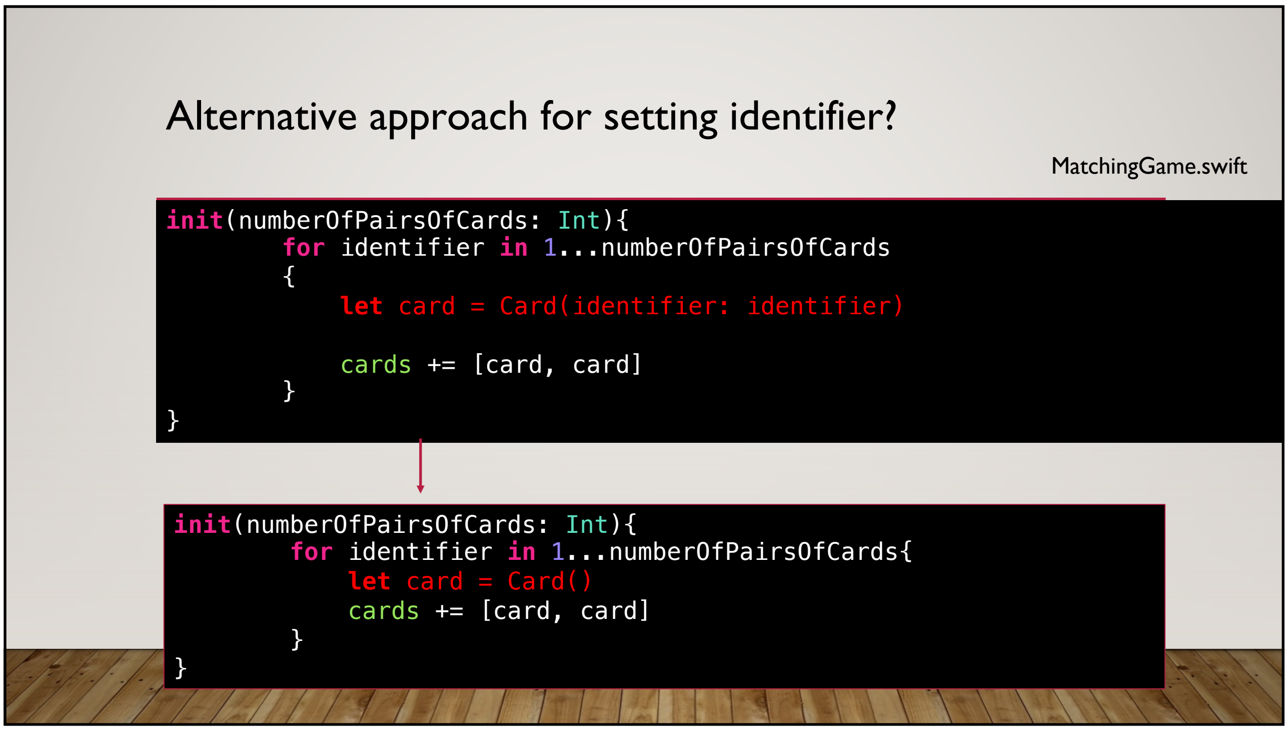
```
init(numberOfPairsOfCards: Int){
        for identifier in 1...numberOfPairsOfCards{
            let card = Card(identifier: identifier)
            cards.append(card)
            ~~let matchingCard  = Card(identifier: identifier)~~
            ~~cards.append(matchingCard)~~
        }

    }
```

Card is struct
i.e. pass-by-value

```
 init(numberOfPairsOfCards: Int){
        for identifier in 1...numberOfPairsOfCards{
            let card = Card(identifier: identifier)
            cards.append(card)
            cards.append(card)
        }
 }
```

```
cards += [card, card]
```

44



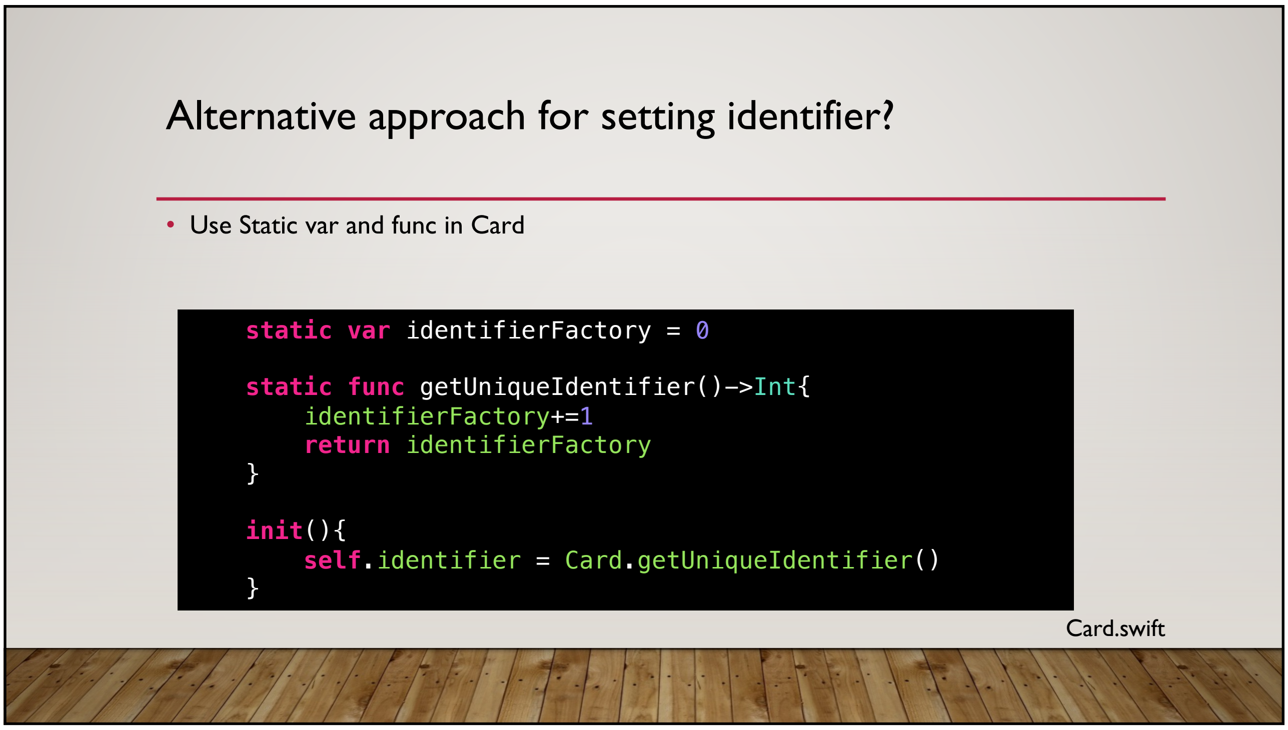
Alternative approach for setting identifier?

MatchingGame.swift

```
init(numberOfPairsOfCards: Int){
        for identifier in 1...numberOfPairsOfCards
        {
            let card = Card(identifier: identifier)

            cards += [card, card]
        }
}
```

```
init(numberOfPairsOfCards: Int){
        for identifier in 1...numberOfPairsOfCards{
            let card = Card()
            cards += [card, card]
        }
}
```

45



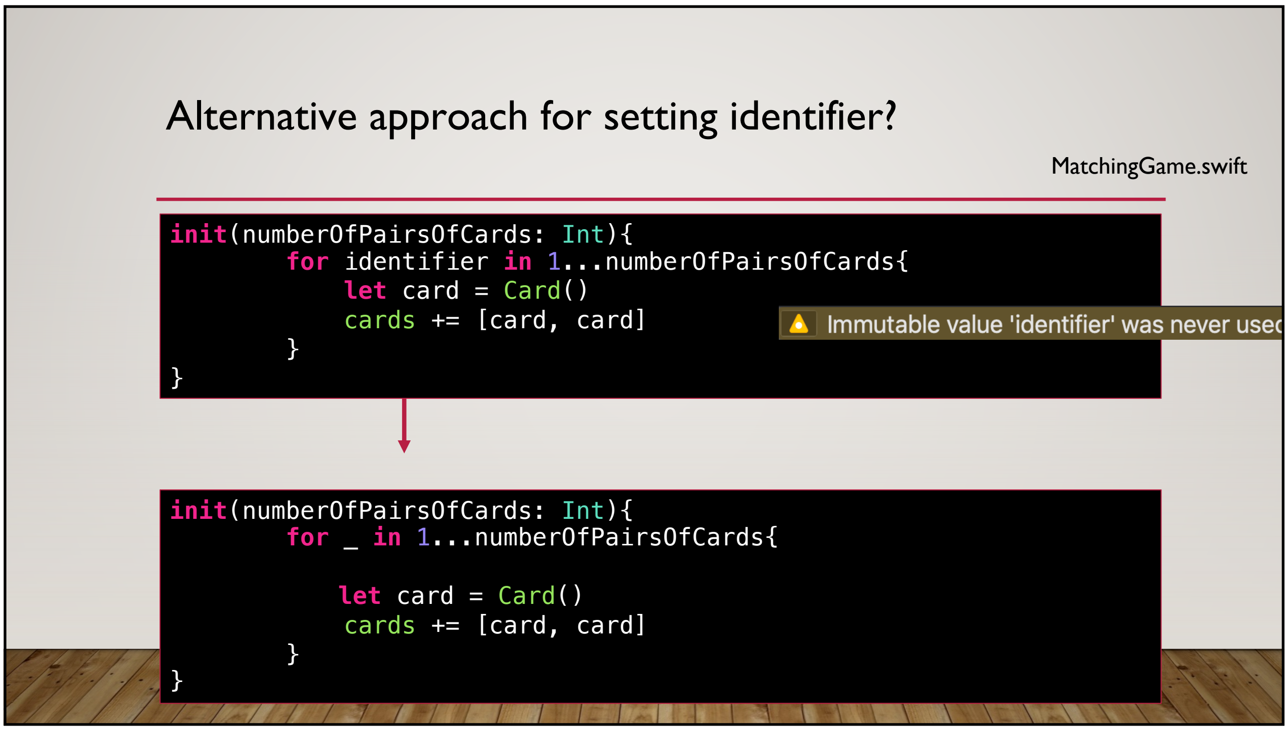
Alternative approach for setting identifier?

- Use Static var and func in Card

```
static var identifierFactory = 0

static func getUniqueIdentifier()->Int{
    identifierFactory+=1
    return identifierFactory
}

init(){
    self.identifier = Card.getUniqueIdentifier()
}
```

Card.swift

46



Alternative approach for setting identifier?

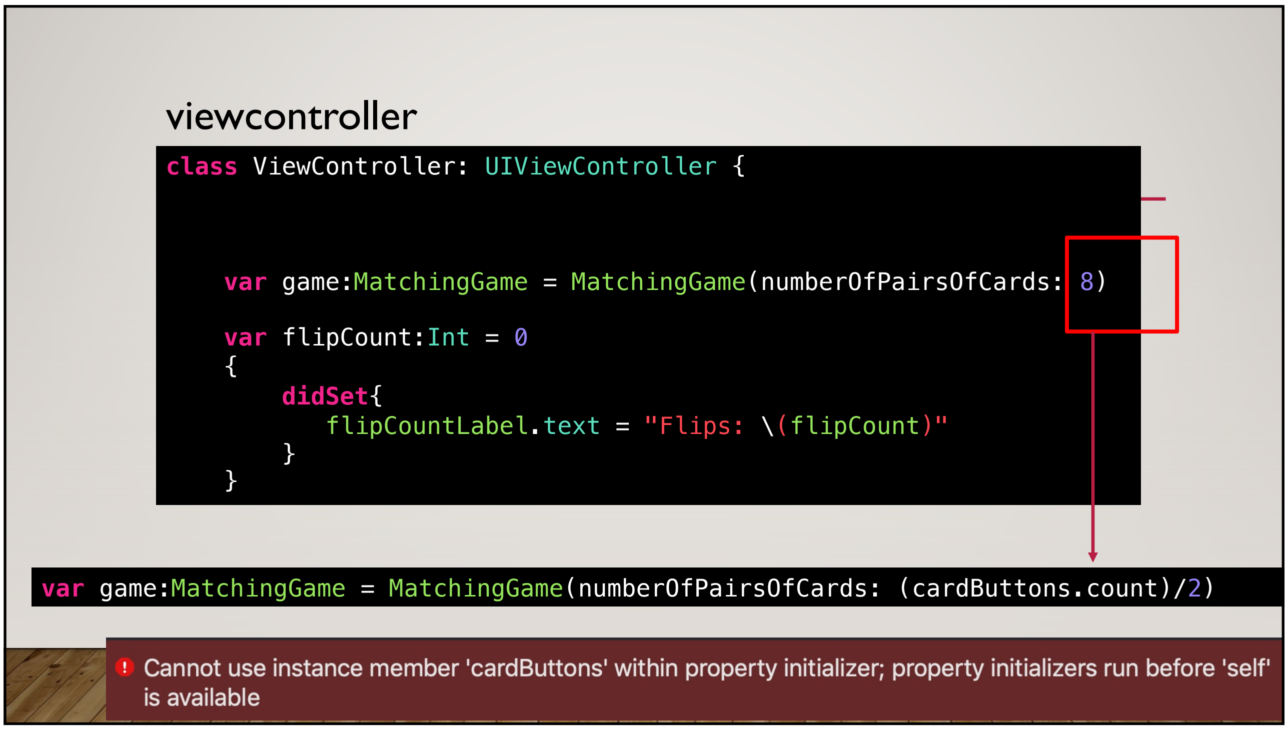
MatchingGame.swift

```
init(numberOfPairsOfCards: Int){
        for identifier in 1...numberOfPairsOfCards{
            let card = Card()
            cards += [card, card]
        }
}
```

⚠ Immutable value 'identifier' was never use

```
init(numberOfPairsOfCards: Int){
        for _ in 1...numberOfPairsOfCards{

            let card = Card()
            cards += [card, card]
        }
}
```

47



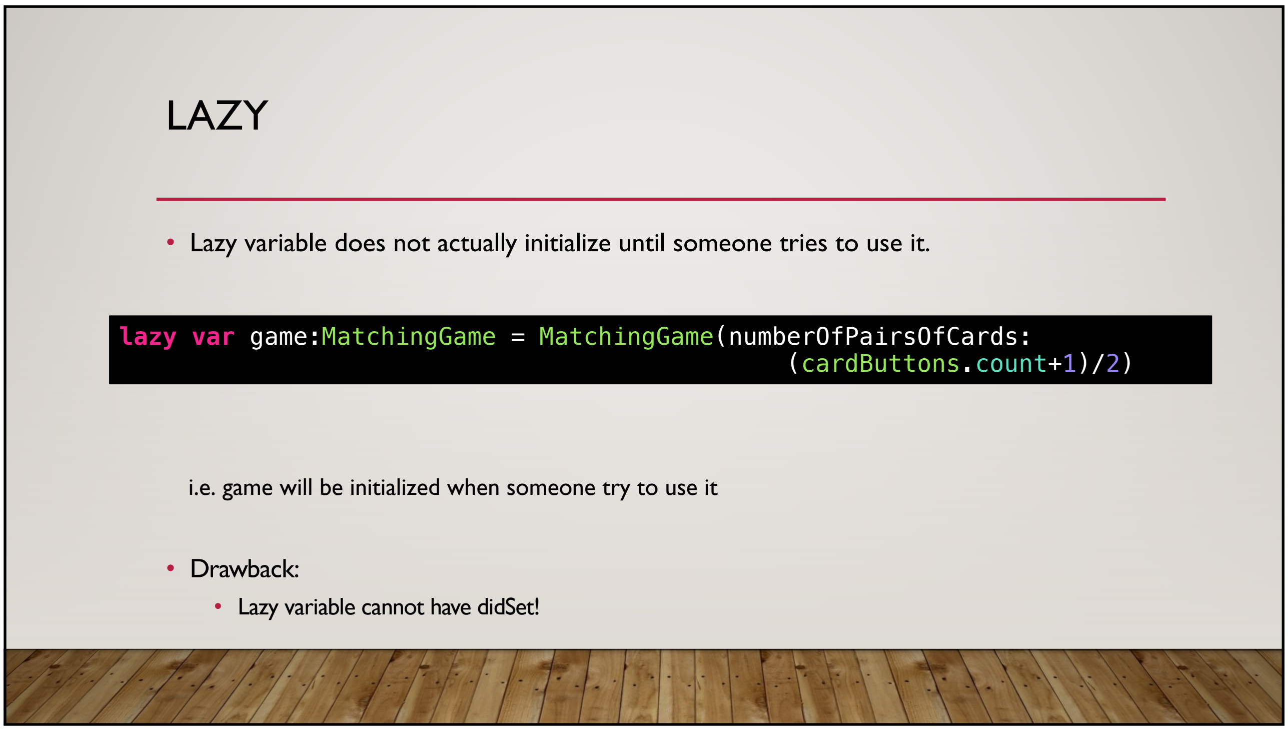
viewcontroller

```
class ViewController: UIViewController {


    var game:MatchingGame = MatchingGame(numberOfPairsOfCards: 8)

    var flipCount:Int = 0
    {
        didSet{
            flipCountLabel.text = "Flips: \(flipCount)"
        }
    }
```

```
var game:MatchingGame = MatchingGame(numberOfPairsOfCards: (cardButtons.count)/2)
```

Cannot use instance member 'cardButtons' within property initializer; property initializers run before 'self' is available

48



LAZY

- Lazy variable does not actually initialize until someone tries to use it.

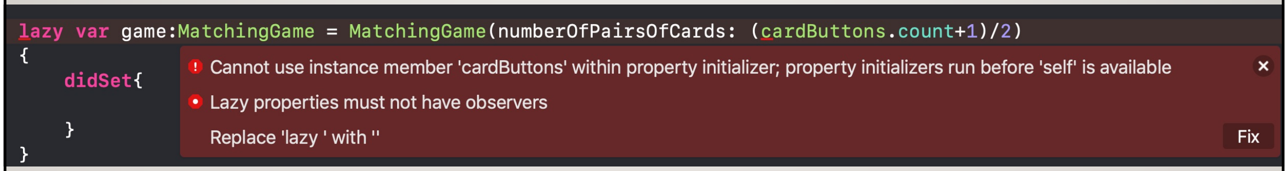
```
lazy var game:MatchingGame = MatchingGame(numberOfPairsOfCards:
                                                     (cardButtons.count+1)/2)
```

i.e. game will be initialized when someone try to use it

- Drawback:
  - Lazy variable cannot have didSet!

49

LAZY



```
lazy var game:MatchingGame = MatchingGame(numberOfPairsOfCards: (cardButtons.count+1)/2)
{
    didSet{

    }
}
```

```
Cannot use instance member 'cardButtons' within property initializer; property initializers run before 'self' is available
Lazy properties must not have observers
Replace 'lazy ' with ''
```

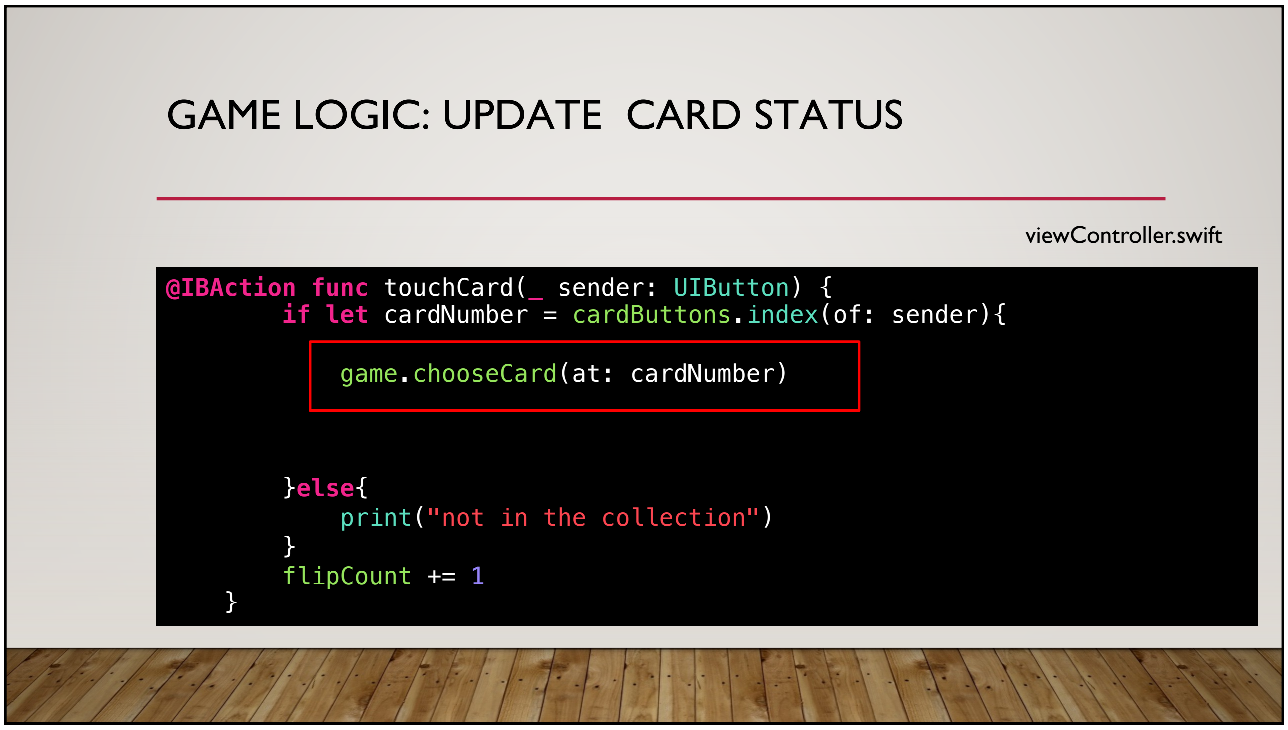
50

GAME LOGIC: UPDATE CARD STATUS

```
func chooseCard(at index: Int)->Card{
        if cards[index].isFaceUp{
            cards[index].isFaceUp = false
        }else{
            cards[index].isFaceUp = true
        }
          return cards[index]

    }
```

MatchingGame.swift

51

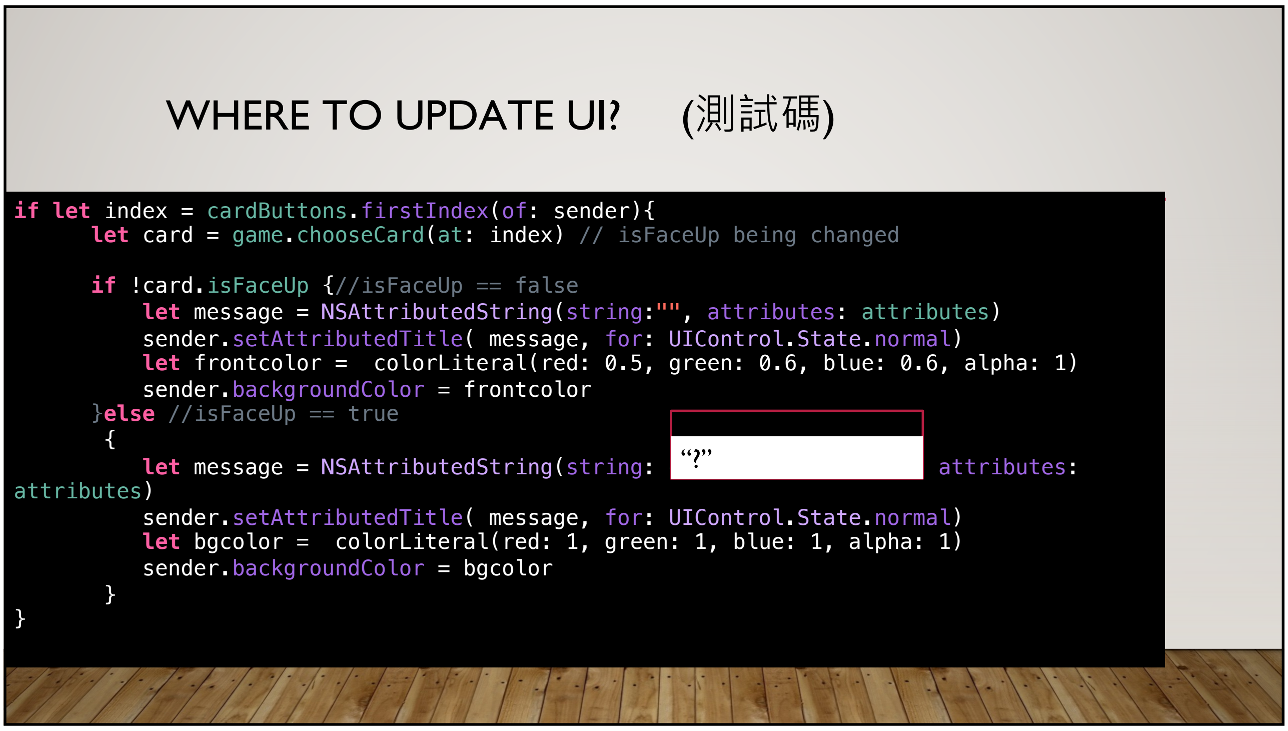


GAME LOGIC: UPDATE CARD STATUS

viewController.swift

```
@IBAction func touchCard(_ sender: UIButton) {
        if let cardNumber = cardButtons.index(of: sender){

            game.chooseCard(at: cardNumber)


        }else{
            print("not in the collection")
        }
        flipCount += 1
    }
```

52



WHERE TO UPDATE UI?   (測試碼)

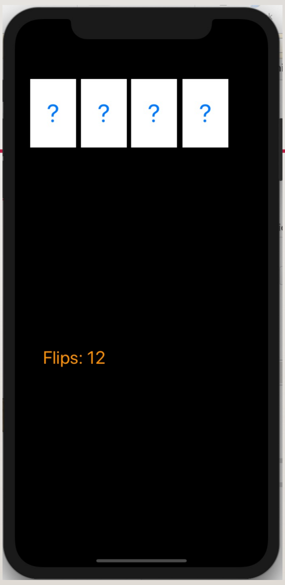
```
if let index = cardButtons.firstIndex(of: sender){
      let card = game.chooseCard(at: index) // isFaceUp being changed

      if !card.isFaceUp {//isFaceUp == false
          let message = NSAttributedString(string:"", attributes: attributes)
          sender.setAttributedTitle( message, for: UIControl.State.normal)
          let frontcolor =  colorLiteral(red: 0.5, green: 0.6, blue: 0.6, alpha: 1)
          sender.backgroundColor = frontcolor
      }else //isFaceUp == true
       {
          let message = NSAttributedString(string: “?”  attributes:
attributes)
          sender.setAttributedTitle( message, for: UIControl.State.normal)
          let bgcolor =  colorLiteral(red: 1, green: 1, blue: 1, alpha: 1)
          sender.backgroundColor = bgcolor
       }
}
```

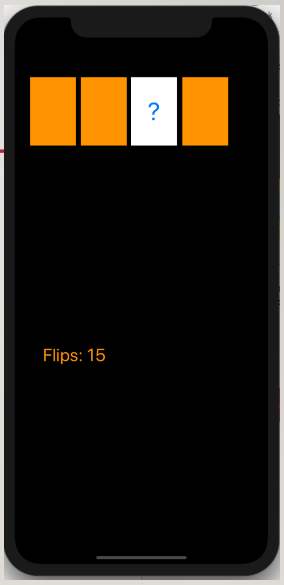
53

TEST

```
var emojiChoices =
["🎃","👻","👽","🎃","✌️","👽","❤️","👻" ]
```

```
func getEmoji(at index:Int)->String{
        if index<emojiChoices.count{
            title = emojiChoices[index]
        }else{
            title = "?"
        }
        return title ?? "?"
    }
```





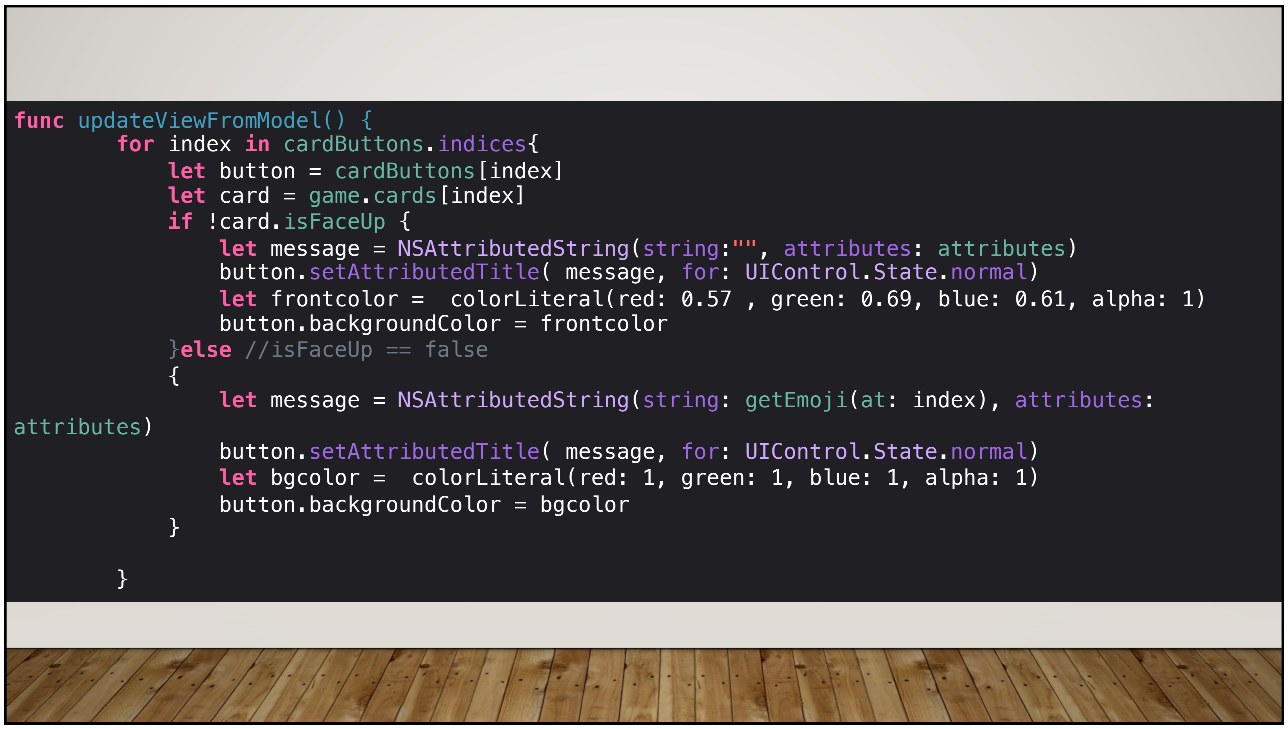
54

WHERE TO UPDATE UI?

```
@IBAction func touchCard(_ sender: UIButton) {

    if let index = cardButtons.firstIndex(of: sender){
       let card = game.chooseCard(at: index) // isFaceUp being changed
       updateViewFromModel()


    }
    flipCount +=1
}
```

```
func updateViewFromModel(){


    }
```

55



```swift
func updateViewFromModel() {
        for index in cardButtons.indices{
            let button = cardButtons[index]
            let card = game.cards[index]
            if !card.isFaceUp {
                let message = NSAttributedString(string:"", attributes: attributes)
                button.setAttributedTitle( message, for: UIControl.State.normal)
                let frontcolor =  colorLiteral(red: 0.57 , green: 0.69, blue: 0.61, alpha: 1)
                button.backgroundColor = frontcolor
            }else //isFaceUp == false
            {
                let message = NSAttributedString(string: getEmoji(at: index), attributes:
attributes)
                button.setAttributedTitle( message, for: UIControl.State.normal)
                let bgcolor =  colorLiteral(red: 1, green: 1, blue: 1, alpha: 1)
                button.backgroundColor = bgcolor
            }

        }
```

56