SCIENCES SORBONNE UNIVERSITÉ

Identifying the different types of devices

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…

It is important to detect devices

- Allows to build Universal applications
  - **And adapt layouts / behaviors to the device**

Tis informations stands in iOS

- UIDevice provides such information
  - **Device name**
  - **Device type**
  - **OS version**
  - **Current orientation**
  - **Battery level**
  - **Battery state**
  - **Existence + state of the proximity sensor (iPhones)**
- Etc.

In short, your friends…

UIDevice & UIScreen

UIDevice (1st glance)

Shared class (you fetch a reference)

- + currentDevice / current

Useful properties

- name, systemName, systemVersion, model, localizedModel, **userInterfaceIdiom**, identifierForVendor

Device orientation

- orientation
  - **unknown, portrait, portraitUpsideDown, landscapeLeft, landscapeRight, faceUp, faceDown**

Battery state

- batteryLevel
- batteryState

UIDevice (1st glance)

- **Shared class (you fetch a reference)**
  - + currentDevice / current
- **Useful pro… …s**
  - name, syste… …odel, **userInterfac…**
- **Device o…**
  - orientation
    - **unknown, p… landscapeL…**
- **Battery st…**
  - batteryLevel
  - batteryState

userInterfaceIdiom

Objective-C
- UIUserInterfaceIdiomUnspecified,
- UIUserInterfaceIdiomPhone,
- UIUserInterfaceIdiomPad
- UIUserInterfaceIdiomTV,
- UIUserInterfaceIdiomCarPlay

Swift
- unspecified, phone, pad, tv, carPlay

UIScreen

Shared class (you fetch a reference)

- +mainScreen / main
- +screens / screen() if several screens
  - **Caution, in carPlay, size might be non standard**

Properties

- Screen size & density
  - **bounds (in points) & scale (CGFloat)**
- Brightness
  - **brightness**
- Suspend/activate brightness adaptation
  - **wantsSoftwareDimming**

Device detection (swift)


```swift
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var lab: UILabel!

    enum terminalType {
        case undefined // oups ;-)
        case iphone35 // screen 3.5" = iPhone <= 4s (being deprecated)
        case iphone40 // screen 4" = iPhone 5, 5s
        case iphone47 // screen 4.7" = iPhone 6, 6s, 7, 8
        case iphone55 // screen 5.5" = iPhone 6+, 6s+, 7+, 8+
        case iphone58 // screen 5.8" = iPhone X/XS
        case iphone6x // screen 6.1" 6.5" &  = iPhone XR & XS Max
        case ipad97 // iPadmini, ipad & ipad pro (9.7")
        case ipad105 // iPadPro 10.5"
        case ipad127 // iPad pro (12.7")
        case tv
        case carplay
    }

    var myDevice = terminalType.undefined
```

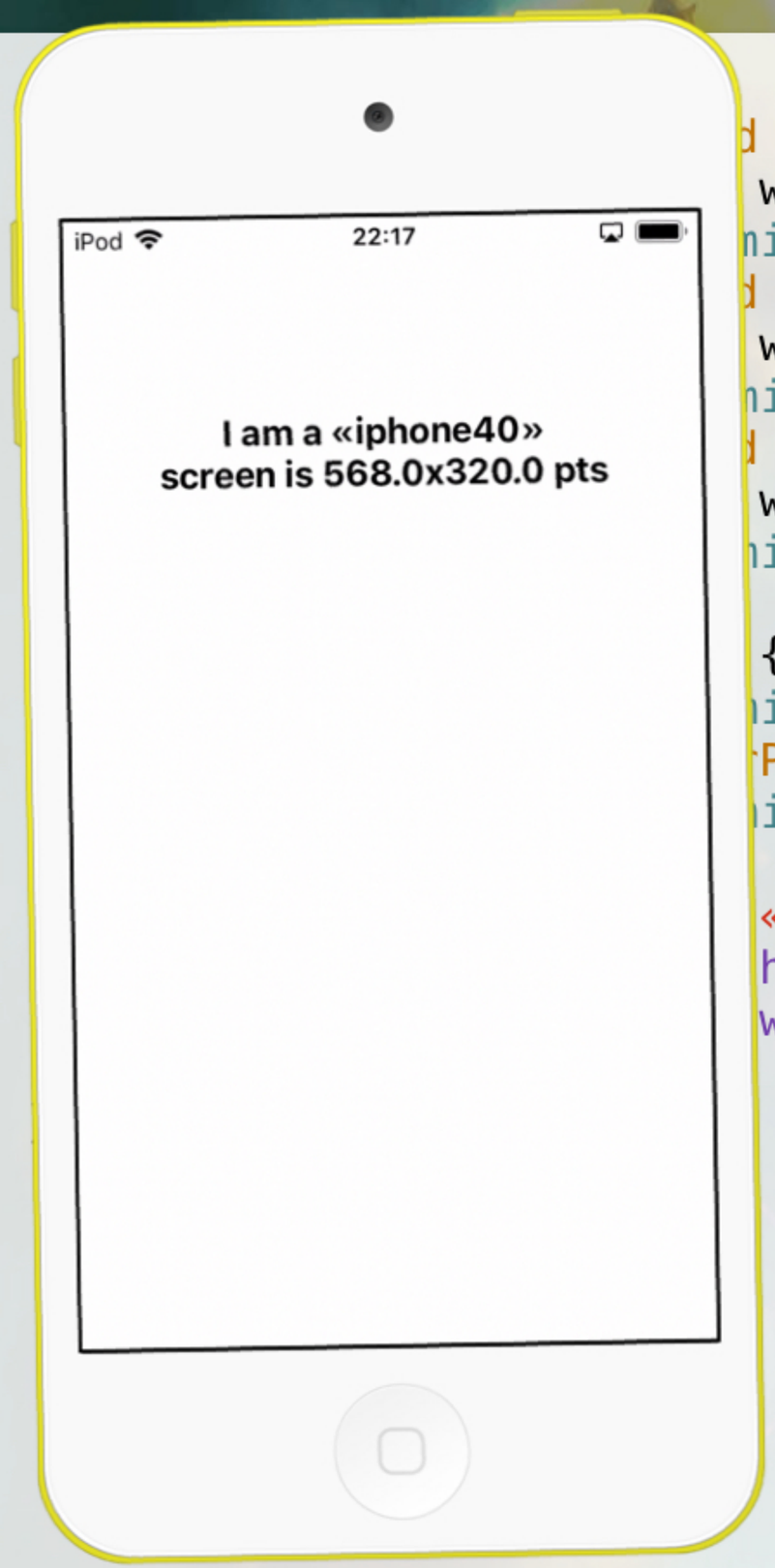
Device detection (swift)


```swift
override func viewDidLoad() {
    super.viewDidLoad()

    // just simplifying the expressions
    let h = UIScreen.main.bounds.size.height
    let w = UIScreen.main.bounds.size.width
    let i = UIDevice.current.userInterfaceIdiom

    // iPhone / iPod touch
    if i == .phone &&
        ((h < 568 && w == 320) || (w < 568 && h == 320)) {
        myDevice = terminalType.iphone35
    } else if i == .phone &&
        ((h == 568 && w == 320) || (w == 568 && h == 320)) {
        myDevice = terminalType.iphone40
    } else if i == .phone &&
        ((h == 667 && w == 375) || (w == 667 && h == 375)) {
        myDevice = terminalType.iphone47
    } else if i == .phone &&
        ((h == 736 && w == 414) || (w == 736 && h == 414)){
        myDevice = terminalType.iphone55
    } else if i == .phone &&
        ((h == 812 && w == 375) || (w == 812 && h == 375)) {
        myDevice = terminalType.iphone58
    } else if i == .phone &&
        ((h == 896 && w == 414) || (w == 896 && h == 414)) {
        myDevice = terminalType.iphone6x
```

Device detection (swift)

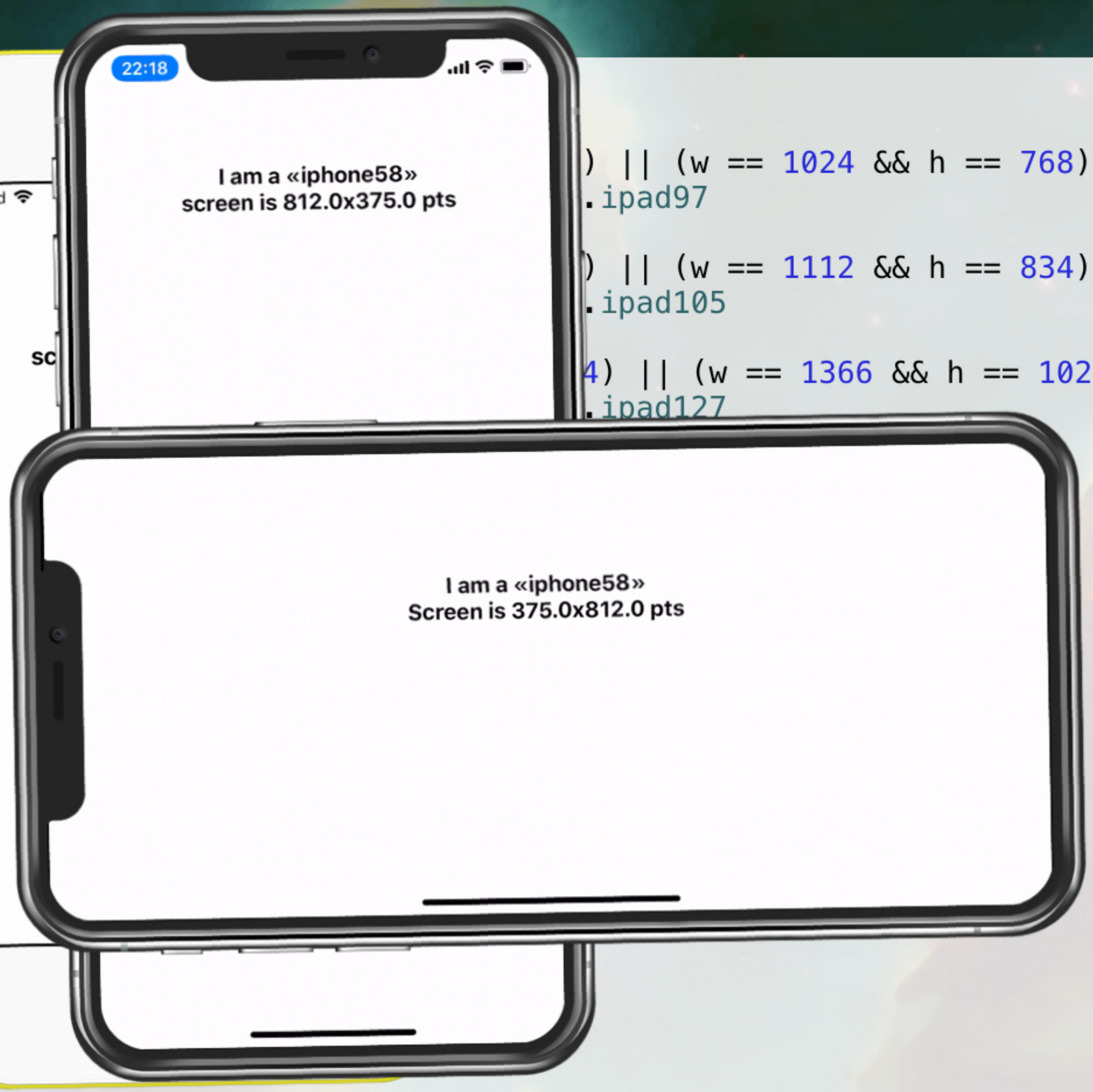

```
// iPads
 } else if i == .pad &&
     ((h == 1024 && w == 768) || (w == 1024 && h == 768)) {
     myDevice = terminalType.ipad97
 } else if i == .pad &&
     ((h == 1112 && w == 834) || (w == 1112 && h == 834)) {
     myDevice = terminalType.ipad105
 } else if i == .pad &&
     ((h == 1366 && w == 1024) || (w == 1366 && h == 1024)) {
     myDevice = terminalType.ipad127
 // others
 } else if i == .tv {
         myDevice = terminalType.tv
     } else if i == .carPlay {
         myDevice = terminalType.carplay
     }
     lab.text = "I am a «\(myDevice)»\nScreen is \
(UIScreen.main.bounds.size.height)x\
(UIScreen.main.bounds.size.width) pts")
    }
}
```

Device detection (swift)



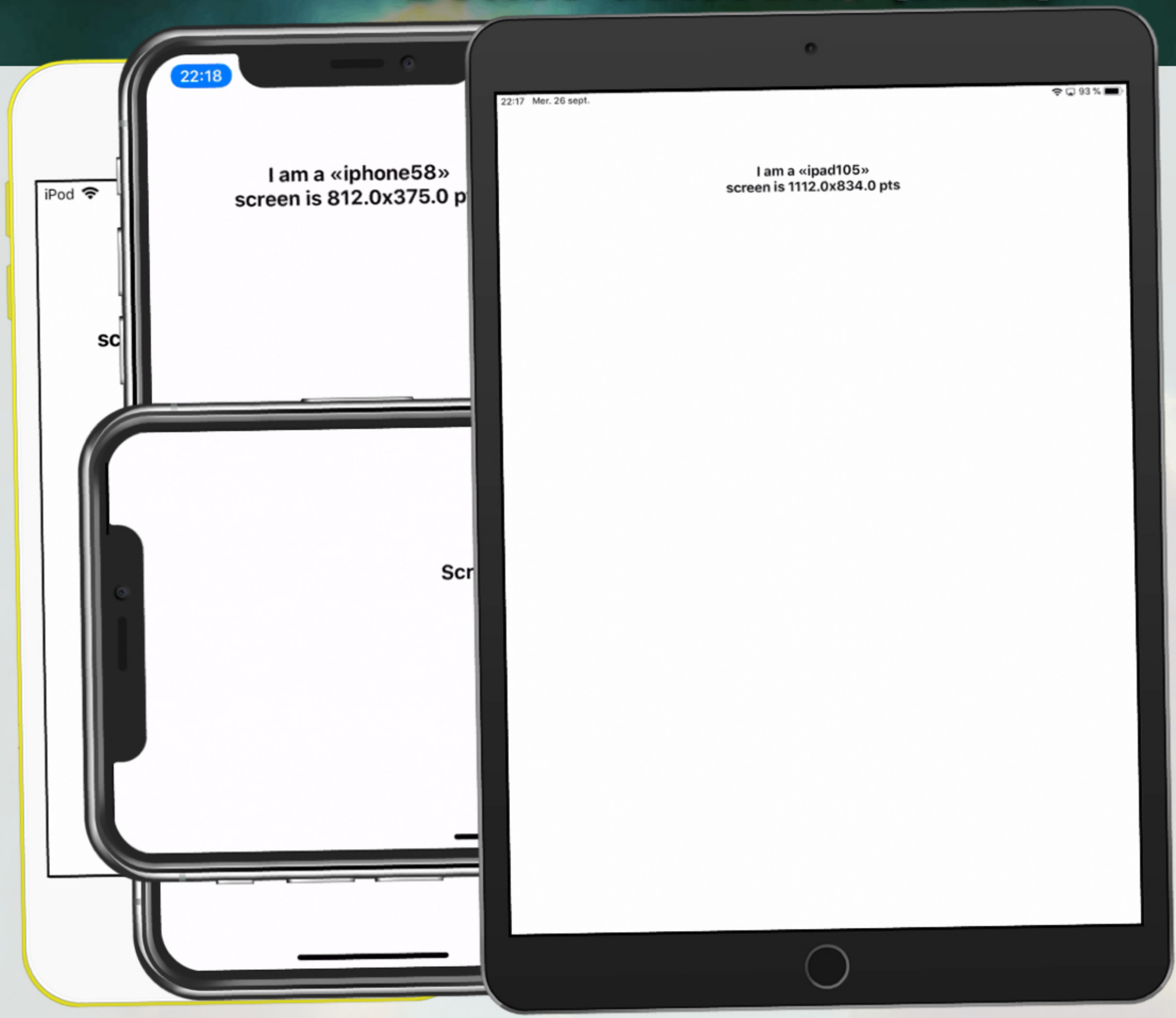

```
d &&
 w == 768) || (w == 1024 && h == 768)) {
minalType.ipad97
d &&
 w == 834) || (w == 1112 && h == 834)) {
minalType.ipad105
d &&
 w == 1024) || (w == 1366 && h == 1024)) {
minalType.ipad127

{
minalType.tv
Play {
minalType.carplay

«\(myDevice)»\nScreen is \
height)x\
width) pts")
```

Device detection (swift)




```
) || (w == 1024 && h == 768)) {
.ipad97

) || (w == 1112 && h == 834)) {
.ipad105

4) || (w == 1366 && h == 1024)) {
.ipad127
```

Device detection (swift)



Objective-C?

Main changes…

- The sequence

```swift
let h = UIScreen.main.bounds.size.height
let w = UIScreen.main.bounds.size.width
let i = UIDevice.current.userInterfaceIdiom
```

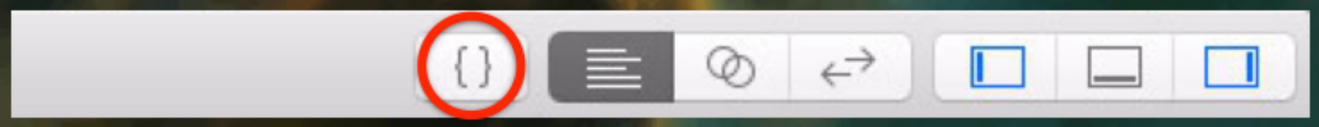
- Becomes

```objc
CGFloat h = [[UIScreen mainScreen] bounds].size.height;
CGFloat w = [[UIScreen mainScreen] bounds].size.width;
CGFloat i = [[UIDevice currentDevice] userInterfaceIdiom];
```

As a conclusion…

It is easy to detect what device runs your App

- Advice, setup a function store as a «snippet»
  - **Only update this function when new sides are out…**



You are ready for programmatic device detection