SCIENCES SORBONNE UNIVERSITÉ

# Layout & constraints (VFL)

Fabrice.Kordon@lip6.fr

LIP6

# Auto Layout?



## Introduced with iOS6

- Constraints management
  - **Relatives or fixed**
- A way to reason without considering the screen size

## Several ways to reach Auto Layout

- The «very complex» way

  - **Storyboard**
- The complex way


  - **Direct use of the NSLayoutConstraint class**
  - **Not presented**
- The easy way : Visual Format Language (VFL)

  - **Contraints expressed using a dedicated language**
  - **Reference to objects/distances thanks to dictionaries**

# Principles (VFL)



## Create view in a view

- UIView embeds useful mechanisms
  - **translatesAutoresizingMaskIntoConstraints**
  - **Attribute set to false = use of Auto Layout**
  - **Default value = true**

## Build constraints

- NSLayoutConstraint (as for any constraints)
  - **constraints(withVisualFormat:options:metrics:views:) / constraintsWithVisualFormat:options:metrics:views:**
  - **The way to build constraints with VFL**

## Apply to the enclosed views

- activate(_:) / activateConstraints:
  - **Using an array of NSLayoutConstraint**
- Constraints can be reused
  - **Involved dictionaries too**

# VFL, Grammar

**<axis><super><dist><view><<dist><view>>*<dist><super>**

*one occurrence or none* *mandatory* *keyword*

## Components

- axis ::= H: or V: or ε *(none means «H:»)*
- super ::= |
- dist ::= -<val>-
- view ::= [<id><dim>]
- <id> ::= any identifier *(in a dictionary)*
- <dim> ::= (<val>)
- <val> ::= <vprio> or <id> *(<id> in a dictionary)*
- <vprio> ::= <int value> or <int value>@<int value>
  - **priorities between 0-250 (lowest), 250-750 (high) up to 1000 (required)**
- Two dictionaries
  - **For views: associates identifiers to views**
  - **For metrics: associates identifiers to integer values**

# VFL, Grammar



**<axis><super><dist><view><<dist><view>>*<dist><super>**

*or ...rrence or none  mandatory  keyword*

## Components

- axis ::= H:
- super ::= |
- dist ::= -<
- view ::= [<id><
- <id> ::= any ide
- <dim> ::= (<
- <val> ::= <vprio> or <id>  *(<id> in a dictionary)*
- <vprio> ::= <int value> or <int value>@<int value>
  - **priorities between 0-250 (lowest), 250-750 (high) up to 1000 (required)**
- Two dictionaries
  - **For views: associates identifiers to views**
  - **For metrics: associates identifiers to integer values**

**Other operators**

== (equal to something)
<= or >=

**Do not be worried…**

Concrete examples are coming…

# Demo (small device)

# Demo (large device)

# View controller's basis



**Warning!**

do not worry about the «tabBar» management (later in this course)

**Sake of simplicity…**

Code located in a ViewController

# View controller's basis



```
import UIKit

class ViewController: UIViewController {

    private var selectedDisplay = 0

    private let red = UIView()
    private let yellow = UIView()
    private let label = UILabel()

    convenience init (_ withDisplay: Int) {
        self.init()
        selectedDisplay = withDisplay
        let  tbitem = UITabBarItem(title: "display \(withDisplay)",
            image: UIImage(named: "viewIcon"), tag: 0)
        self.tabBarItem = tbitem
}
```

# View controller's basis



```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    self.view = UIView(frame: UIScreen.main.bounds)
    self.view.backgroundColor = .white
    red.translatesAutoresizingMaskIntoConstraints = false
    red.backgroundColor = .red
    yellow.translatesAutoresizingMaskIntoConstraints = false
    yellow.backgroundColor = .yellow
    label.text = "Hello!!!"
    label.textColor = .white
    label.font = UIFont(name: "Helvetica-Bold", size: 16.0)
    label.textAlignment = .center
    label.translatesAutoresizingMaskIntoConstraints = false
    label.backgroundColor = .blue
    self.view.addSubview(red)
    self.view.addSubview(yellow)
    self.view.addSubview(label)
    switch selectedDisplay {
        case 1: self.display1()
        case 2: self.display2()
        case 3: self.display3()
        case 4: self.display4()
        case 5: self.display5()
        case 6: self.display6()
        case 7: self.display7()
        default: break
    }
}
```

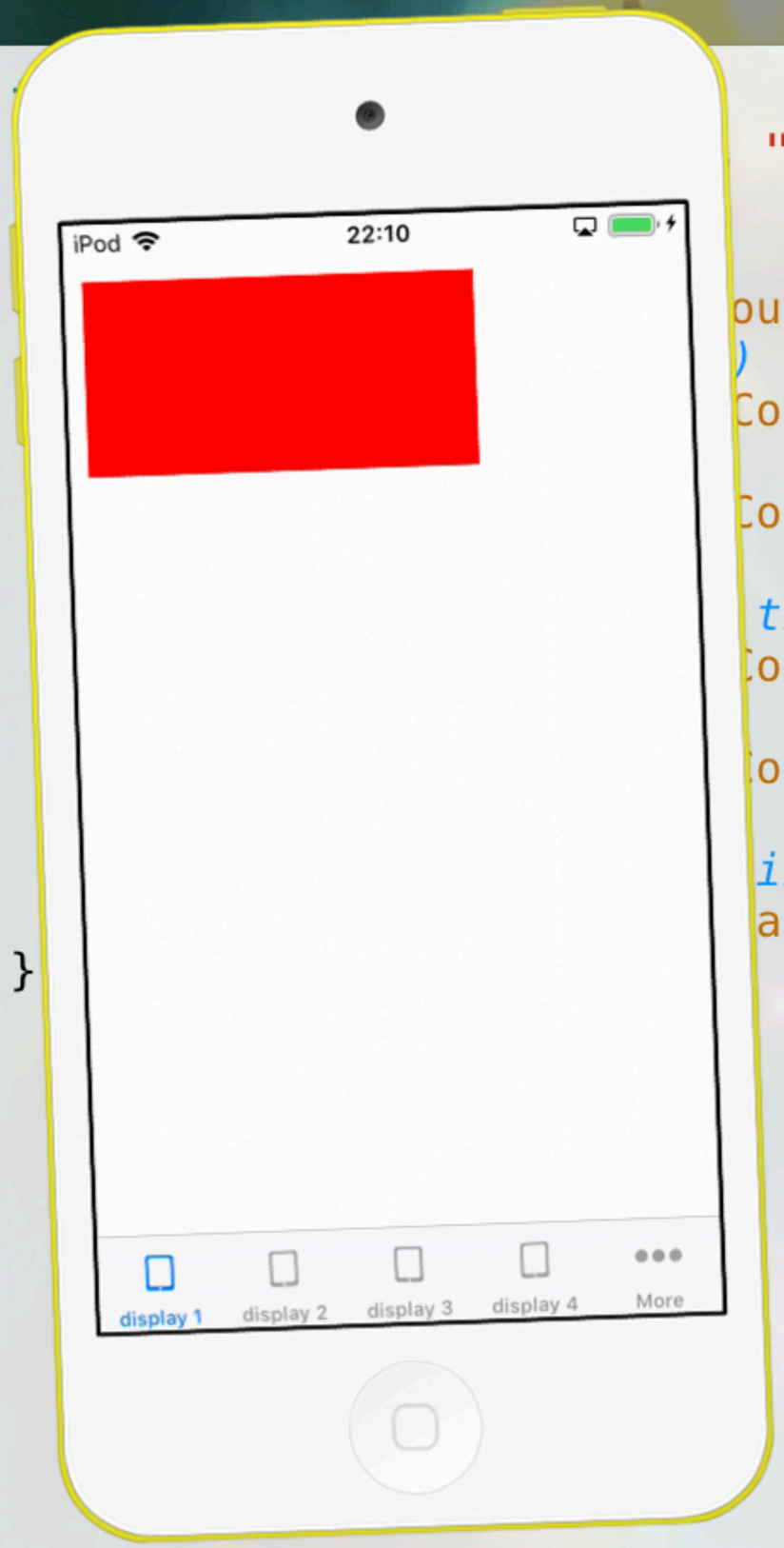
# display1()



```swift
func display1() {
   let dico = ["R":self.red, "Y":self.yellow, "L":self.label]
   yellow.isHidden = true
  label.isHidden = true
    var constraints: [NSLayoutConstraint] = []
    // Size (width & height)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:[R(100)]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:[R(200)]",
                                                  metrics: nil, views: dico)
    // Positions (x & y) in the super view
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-30-[R]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:|-10-[R]",
                                                  metrics: nil, views: dico)
    // Activate constraints in the view
    NSLayoutConstraint.activate(constraints)
}
```

# display1()






```
                                "Y":self.yellow, "L":self.label]


                            outConstraint] = []
                            )
                            Constraint.constraints(withVisualFormat: "V:[R(100)]",
                                                   metrics: nil, views: dico)
                            Constraint.constraints(withVisualFormat: "H:[R(200)]",
                                                   metrics: nil, views: dico)
                            the super view
                            onstraint.constraints(withVisualFormat: "V:|-30-[R]",
                                                   metrics: nil, views: dico)
                            onstraint.constraints(withVisualFormat: "H:|-10-[R]",
                                                   metrics: nil, views: dico)
                            in the view
                            ate(constraints)
}
```

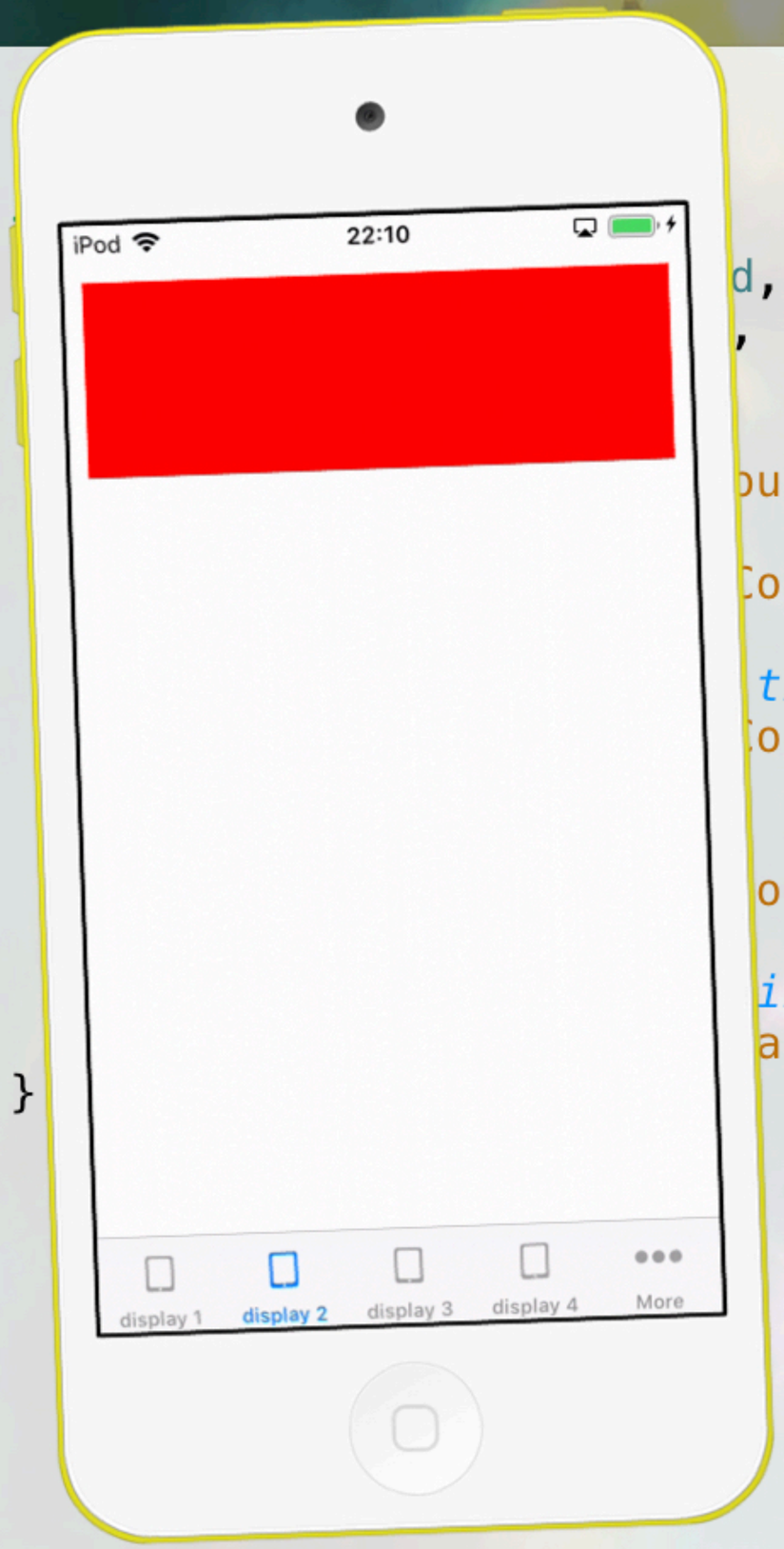
# display2()



```
func display2() {
    let dico = ["R":self.red, "Y":self.yellow, "L":self.label]
    let dist = ["vspace":30, "hspace":10, "hred":100]
    yellow.isHidden = true
    label.isHidden = true
    var constraints: [NSLayoutConstraint] = []
    // Size
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:[R(hred)]",
                                                  metrics: dist, views: dico)
    // Positions (x & y) in the super view
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-vspace-[R]",
                                                  metrics: dist, views: dico)
    constraints +=
        NSLayoutConstraint.constraints(withVisualFormat: "H:|-hspace-[R]-hspace-|",
                                       metrics: dist, views: dico)
    // Activate constraints in the view
    NSLayoutConstraint.activate(constraints)
}
```

# display2()






```
d, "Y":self.yellow, "L":self.label]
, "hspace":10, "hred":100]


utConstraint] = []

Constraint.constraints(withVisualFormat: "V:[R(hred)]",
                                    metrics: dist, views: dico)
the super view
onstraint.constraints(withVisualFormat: "V:|-vspace-[R]",
                                    metrics: dist, views: dico)

onstraints(withVisualFormat: "H:|-hspace-[R]-hspace-|",
                    metrics: dist, views: dico)
in the view
ate(constraints)
}
```

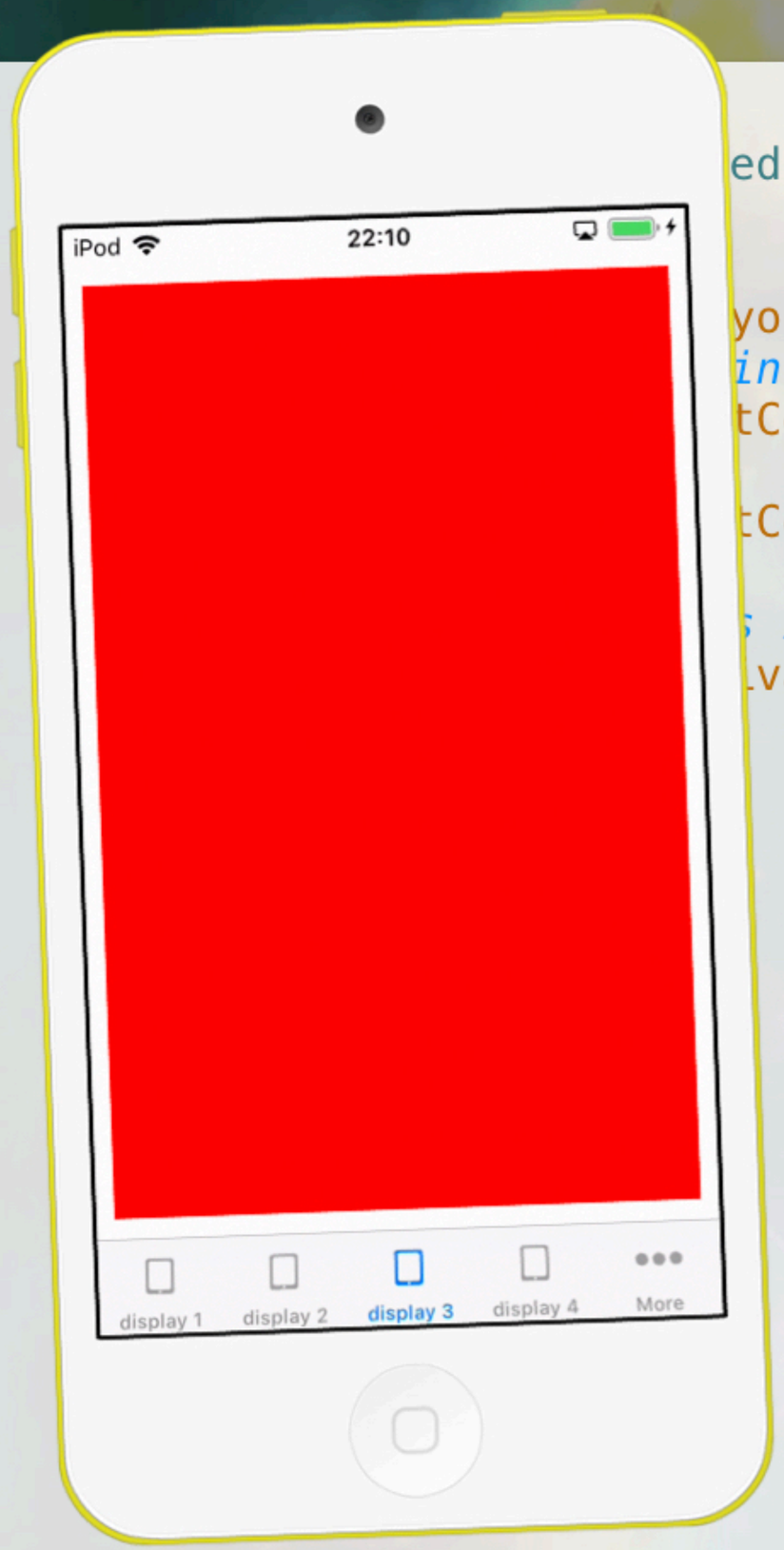
# display3()



```
func display3() {
    let dico = ["R":self.red, "Y":self.yellow, "L":self.label]
    yellow.isHidden = true
    label.isHidden = true
    var constraints: [NSLayoutConstraint] = []
    // Relative positions in the super view
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-30-[R]-60-|",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:|-10-[R]-10-|",
                                                  metrics: nil, views: dico)
    // Activate constraints in the view
    NSLayoutConstraint.activate(constraints)
}
```

# display3()






```
ed, "Y":self.yellow, "L":self.label]


youtConstraint] = []
in the super view
tConstraint.constraints(withVisualFormat: "V:|-30-[R]-60-|",
                                        metrics: nil, views: dico)
tConstraint.constraints(withVisualFormat: "H:|-10-[R]-10-|",
                                        metrics: nil, views: dico)
 in the view
vate(constraints)
```

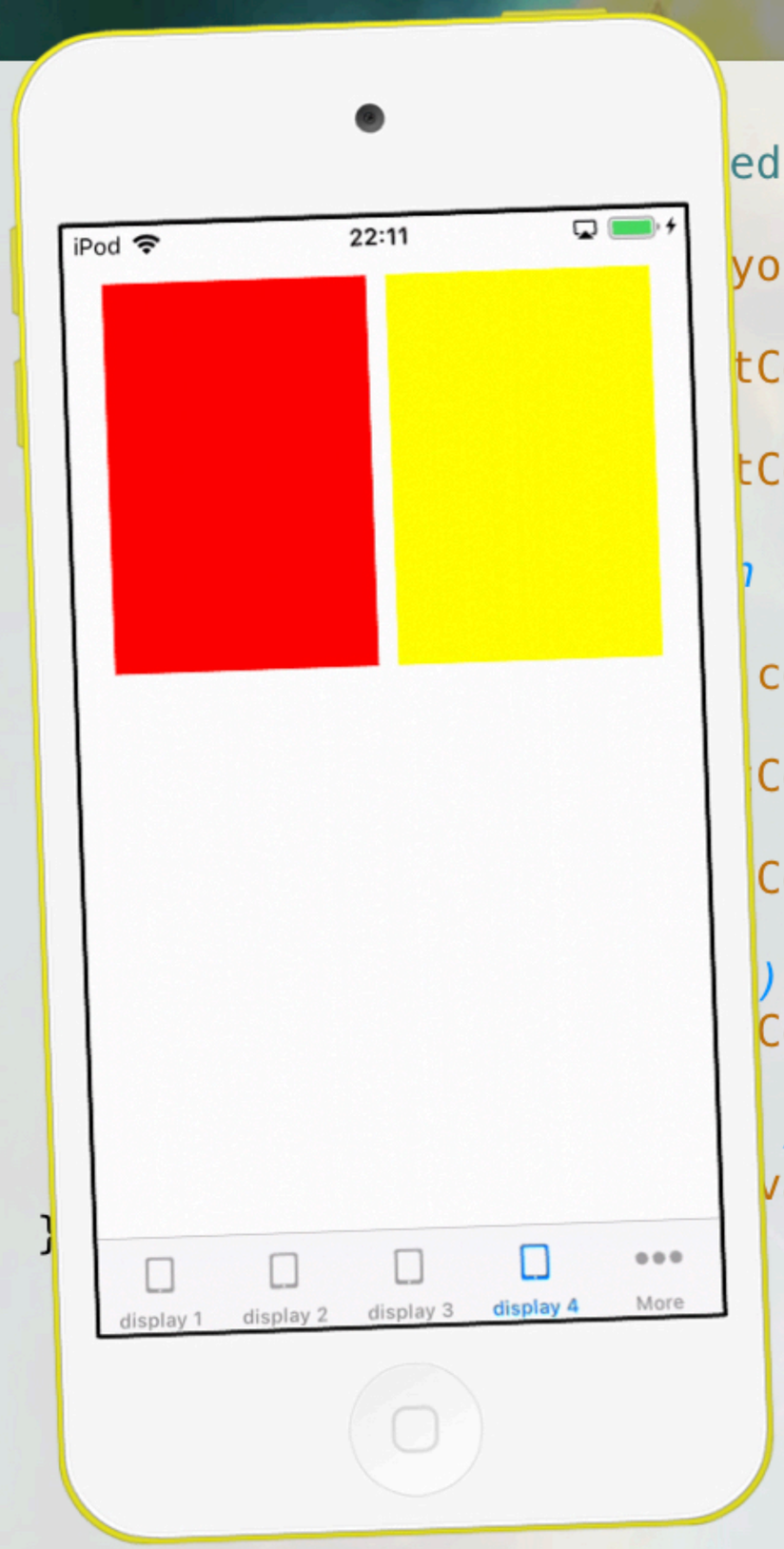
# display4()



```swift
func display4() {
    let dico = ["R":self.red, "Y":self.yellow, "L":self.label]
    label.isHidden = true
    var constraints: [NSLayoutConstraint] = []
    // Size
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:[R(200)]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:[Y(200)]",
                                                  metrics: nil, views: dico)
    // Positions (x & y) in the super view
    constraints +=
        NSLayoutConstraint.constraints(withVisualFormat: "H:|-20-[R]-10-[Y]-20-|",
                                       metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-30-[R]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-30-[Y]",
                                                  metrics: nil, views: dico)
    // Size (width & height)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:[R(==Y)]",
                                                  metrics: nil, views: dico)
    // Activate constraints in the view
    NSLayoutConstraint.activate(constraints)
}
```

# display4()






```
ed, "Y":self.yellow, "L":self.label]

youtConstraint] = []

tConstraint.constraints(withVisualFormat: "V:[R(200)]",
                        metrics: nil, views: dico)
tConstraint.constraints(withVisualFormat: "V:[Y(200)]",
                        metrics: nil, views: dico)
 the super view

constraints(withVisualFormat: "H:|-20-[R]-10-[Y]-20-|",
            metrics: nil, views: dico)
Constraint.constraints(withVisualFormat: "V:|-30-[R]",
                       metrics: nil, views: dico)
Constraint.constraints(withVisualFormat: "V:|-30-[Y]",
                       metrics: nil, views: dico)
)
Constraint.constraints(withVisualFormat: "H:[R(==Y)]",
                       metrics: nil, views: dico)
 in the view
vate(constraints)
```

# display5()



```swift
func display5() {
    let dico = ["R":self.red, "Y":self.yellow, "L":self.label]
    label.isHidden = true
    var constraints: [NSLayoutConstraint] = []
    // Positions (x & y) in the super view
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:|[Y][R]|",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-80-[R(100)]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-100-[Y(80)]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:[R(==Y)]",
                                                  metrics: nil, views: dico)
    // Activate constraints in the view
    NSLayoutConstraint.activate(constraints)

}
```

# display5()




```
ed, "Y":self.yellow, "L":self.label]

youtConstraint] = []
n the super view
tConstraint.constraints(withVisualFormat: "H:|[Y][R]|",
                                    metrics: nil, views: dico)
tConstraint.constraints(withVisualFormat: "V:|-80-[R(100)]",
                                    metrics: nil, views: dico)
tConstraint.constraints(withVisualFormat: "V:|-100-[Y(80)]",
                                    metrics: nil, views: dico)
tConstraint.constraints(withVisualFormat: "H:[R(==Y)]",
                                    metrics: nil, views: dico)
 in the view
vate(constraints)
```

# display6()



```swift
func display6() {
    let dico = ["R":self.red, "Y":self.yellow, "L":self.label]
    label.isHidden = true
    var constraints: [NSLayoutConstraint] = []
    // Positions (x & y) in the super view
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:|[Y][R]|",
                                                  metrics: nil, views: dico)
    constraints +=
        NSLayoutConstraint.constraints(withVisualFormat: "V:|-80-[R(100)][Y]-60-|",
                                       metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "H:[R(==Y)]",
                                                  metrics: nil, views: dico)

    // Activate constraints in the view
    NSLayoutConstraint.activate(constraints)
}
```

# display6()




```
ed, "Y":self.yellow, "L":self.label]

youtConstraint] = []
n the super view
tConstraint.constraints(withVisualFormat: "H:|[Y][R]|",
                                  metrics: nil, views: dico)

.constraints(withVisualFormat: "V:|-80-[R(100)][Y]-60-|",
                        metrics: nil, views: dico)
tConstraint.constraints(withVisualFormat: "H:[R(==Y)]",
                                  metrics: nil, views: dico)
 in the view
vate(constraints)
```

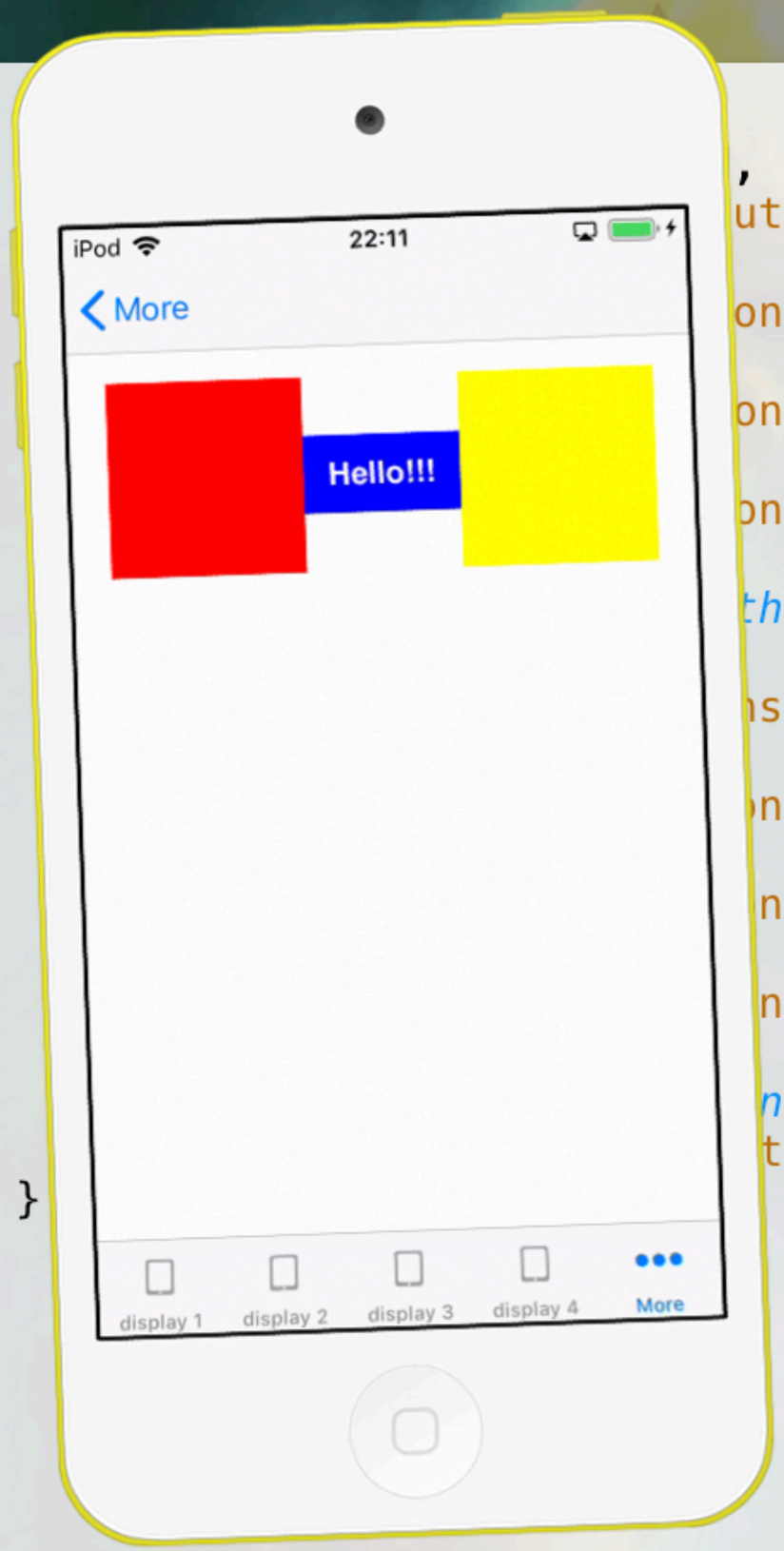
# display7()



```
func display7() {
    let dico = ["R":self.red, "Y":self.yellow, "L":self.label]
    var constraints: [NSLayoutConstraint] = []
    // Size
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:[R(100)]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:[Y(100)]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:[L(40)]",
                                                  metrics: nil, views: dico)
    // Positions (x & y) in the super view
    constraints +=
       NSLayoutConstraint.constraints(withVisualFormat: "H:|-20-[R(100)][L][Y(100)]-20-|",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-80-[R]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-80-[Y]",
                                                  metrics: nil, views: dico)
    constraints += NSLayoutConstraint.constraints(withVisualFormat: "V:|-110-[L]",
                                                  metrics: nil, views: dico)
    // Activate constraints in the view
    NSLayoutConstraint.activate(constraints)
}
```

# display7()






```
, "Y":self.yellow, "L":self.label]
utConstraint] = []

onstraint.constraints(withVisualFormat: "V:[R(100)]",
                      metrics: nil, views: dico)
onstraint.constraints(withVisualFormat: "V:[Y(100)]",
                      metrics: nil, views: dico)
onstraint.constraints(withVisualFormat: "V:[L(40)]",
                      metrics: nil, views: dico)
the super view

nstraints(withVisualFormat: "H:|-20-[R(100)][L][Y(100)]-20-|",
                      metrics: nil, views: dico)
nstraint.constraints(withVisualFormat: "V:|-80-[R]",
                      metrics: nil, views: dico)
nstraint.constraints(withVisualFormat: "V:|-80-[Y]",
                      metrics: nil, views: dico)
nstraint.constraints(withVisualFormat: "V:|-110-[L]",
                      metrics: nil, views: dico)
n the view
te(constraints)
}
```

# Changing layout with device?



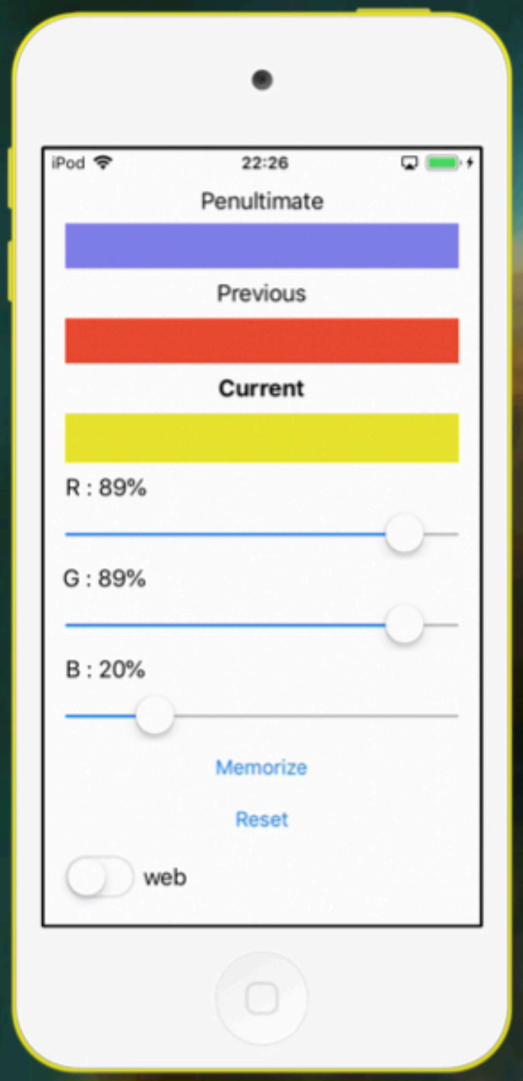
## Trick 1 — conditional code you do

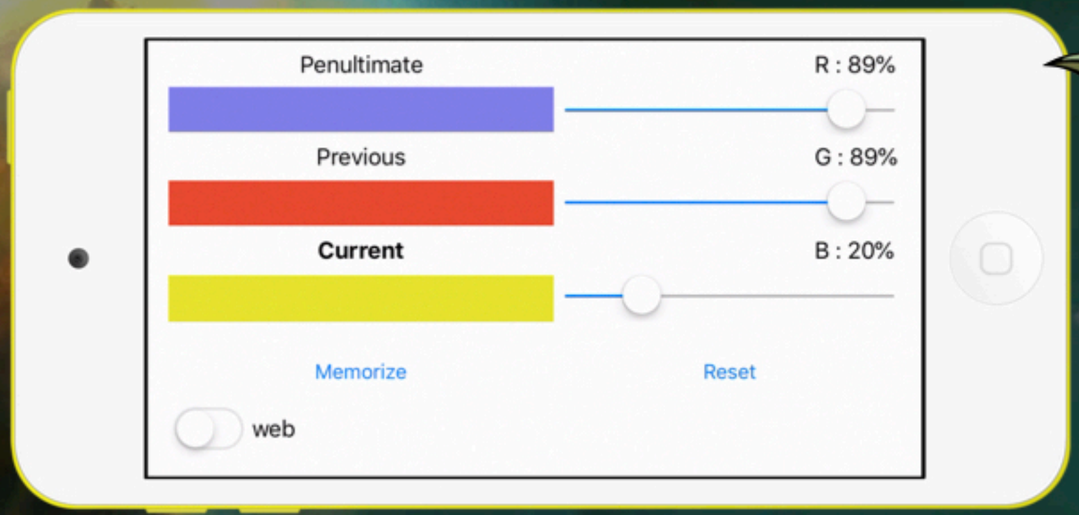
- Changing constraints according to the detected device

## Trick 2 — changing dictionaries you can

- The metric dictionary is intended for such a purpose
- One set of constraints + adapted dictionaries

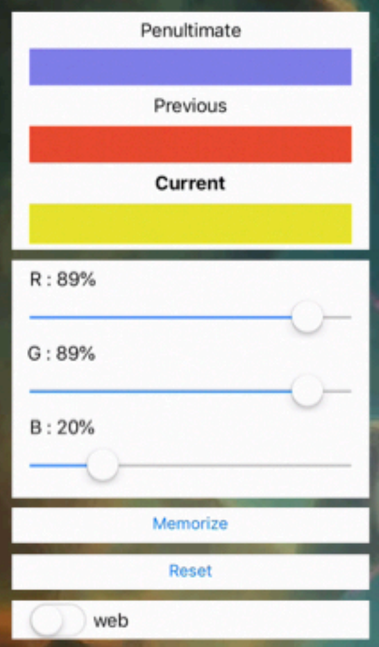
# Complex layouts changes?

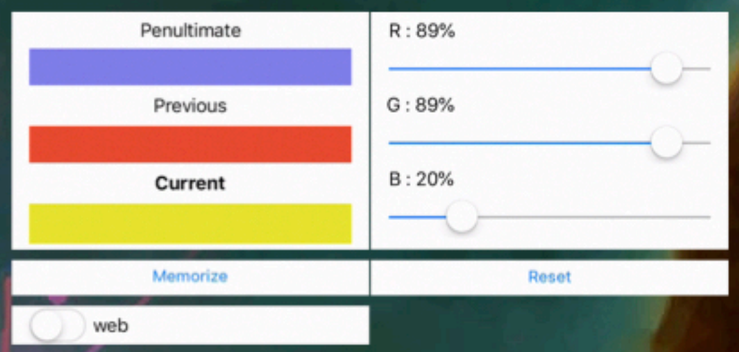



## UIStackView, your friend is

- You update group of views
  - **Trick used in WatchOS**

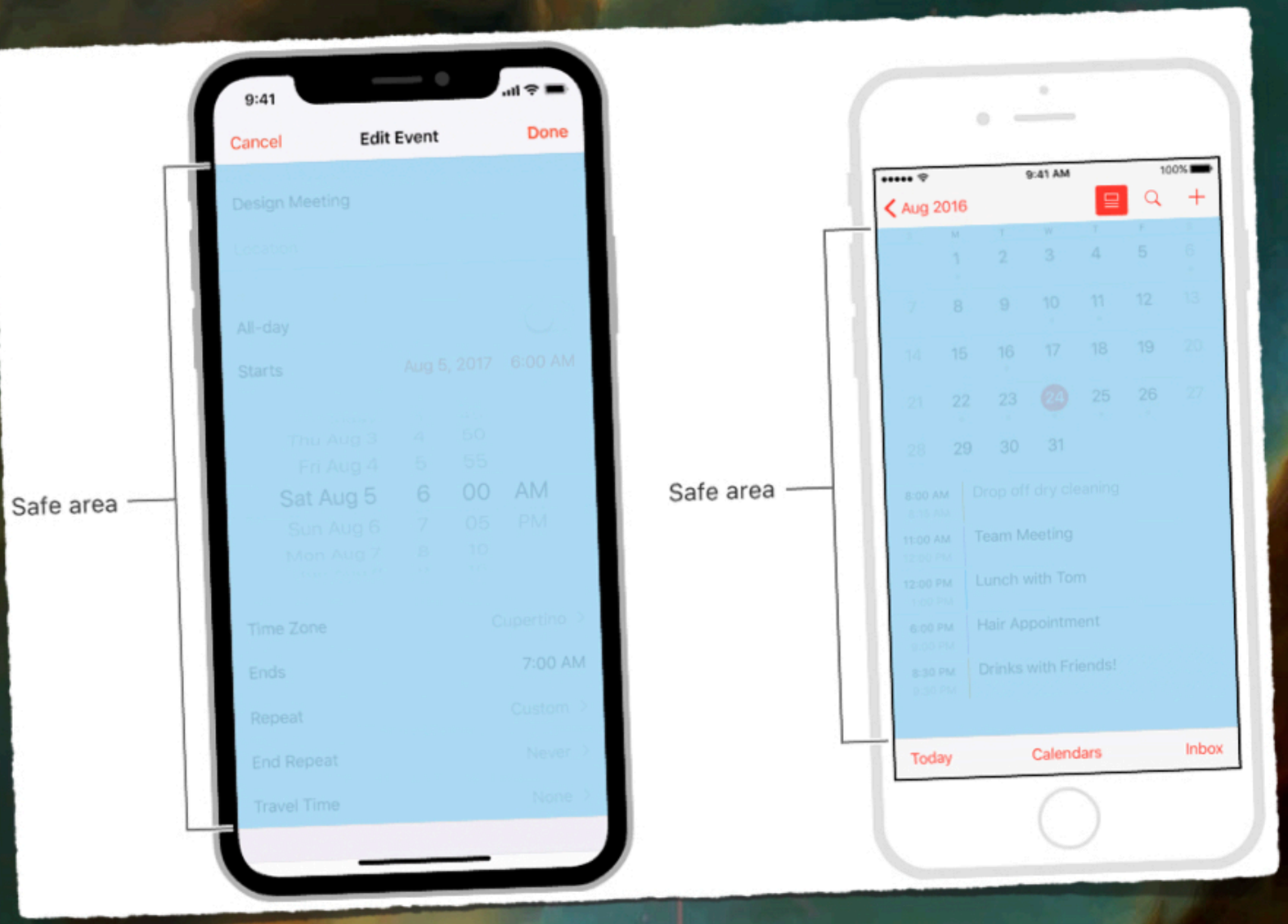


# Last Yoda's advice

## The safe area you will use…

- Avoids the bevel & corners
- Property safeAreaLayoutGuide



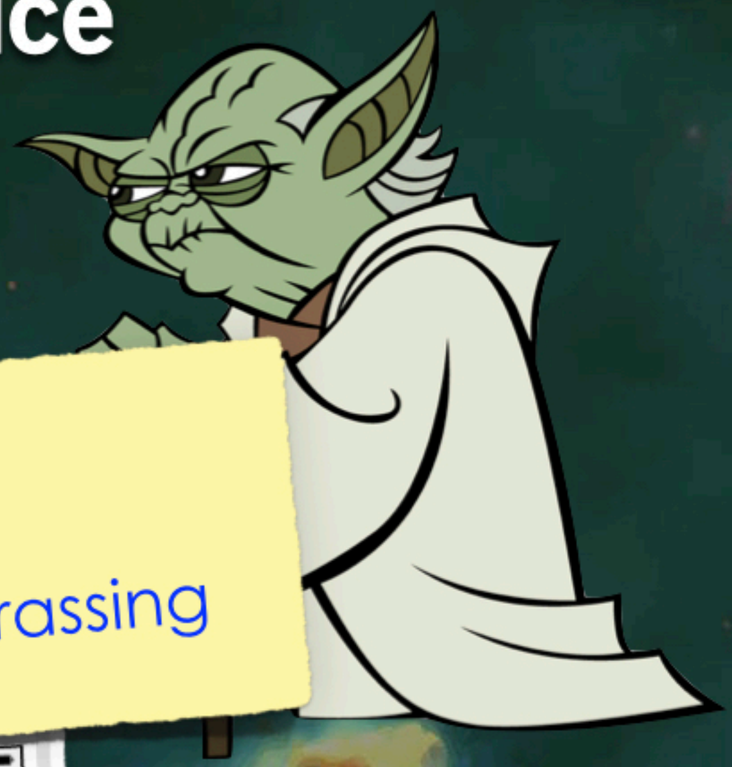
# Last Yoda's advice



- **The safe area you will use…**
  - Avoids the bevel & ers
  - Property safeA

**Avoid this…**

… because this is embarrassing



# As a conclusion…



## You liked

- Kindergarten mode
- Geek mode
- You will like Auto Layout!!!

## Be careful

- Constraints conflicts can exist too!
  - **And you can have fun**
  - **Look at the Xcode console for explanation**
- Still important to draw your interface first


## And for free…

- Orientation management
  - **When organization does not change…**
  - **Otherwise you change the set of constraints**