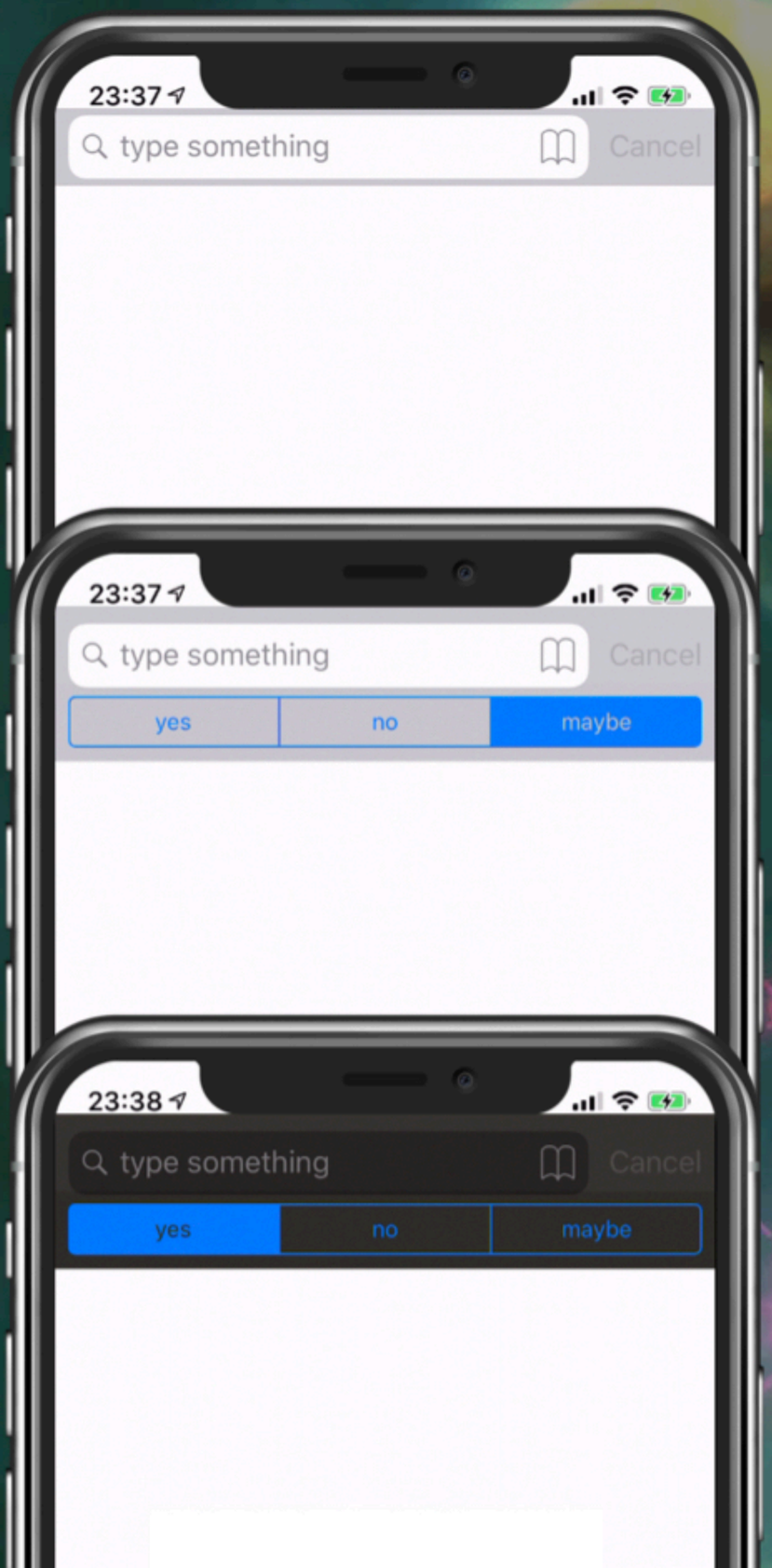
SCIENCES SORBONNE UNIVERSITÉ

UISearchBar

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…



Predefined search bar

- No need to reinvent the wheel
- ScopeBar
- Different styles
  - **UIBarStyleDefault, UIBarStyleBlack**
  - **default, black**

Init with a frame

- Configuration of keyboard
- Change of appearance
  - **tintColor, etc.**

Use of delegation

- UISearchBarDelegate

Principles

As usual

Create a UISearchBar

- Customize it
  - placeHolder, text, barTintColor, tintColor, showsCancelButton, showBookMarkButton…

Handle it thanks to UISearchBarDelegate

- searchBar(_:textDidChange:)
- searchBarBookmarkButtonClicked(_:)
- searchBarCancelButtonClicked(_:)
- searchBarSearchButtonClicked(_:)
- searchBar(_:selectedScopeButtonIndexDidChange:)

Principles

As usual

Create a UISearchBar

- Customize it
  - placeHolder, text, barTintColor, tintColor, showsCancelButton, showBookMarkButton…

Handle it thanks to UISe arDelegate

- searchBar(_:textDidChange:)
- searchBarBookmarkButtonClicked(_:
- searchBarCancelButto :)
- searchBarSearchButto
- searchBar(_:selected lexDidChange:)

An example

MyView

I am lazy…

Do not show the ViewController code
(it is as usual)

MyView


```
import UIKit

class MyView: UIView, UISearchBarDelegate {

    private let myBar = UISearchBar()
    private let l = UILabel()
    private let b = UIButton(type: .system)
    private var extraH = 0

    override init(frame: CGRect) {
        myBar.showsBookmarkButton = true
        myBar.showsCancelButton = true
        myBar.barStyle = .black
        myBar.placeholder = "type something"
        l.text = "no action on the search bar yet"
        l.textAlignment = .center
        b.setTitle("Update", for: .normal)
        super.init(frame:frame)
        self.backgroundColor = .white
        myBar.delegate = self
        b.addTarget(self, action: #selector(updateBar),
                    for: .touchDown)
        self.addSubview(myBar)
        self.addSubview(b)
        self.addSubview(l)
        self.drawInSize(size: frame.size)
}
```

MyView


```
required init?(coder aDecoder: NSCoder) {
    fatalError("init(coder:) has not been implemented")
}

@objc func updateBar (sender : UIButton) {
    if myBar.showsScopeBar {
        extraH = 0
        myBar.scopeButtonTitles = nil
        myBar.showsScopeBar = false
    } else {
        extraH = 44
        myBar.scopeButtonTitles = ["yes", "no", "maybe"]
        myBar.showsScopeBar = true
    }
    self.drawInSize(size: UIScreen.main.bounds.size)
}
```

MyView


```
func drawInSize (size: CGSize) {
    var top = 20;
    if UIDevice.current.userInterfaceIdiom == .phone &&
        size.height >= 812 {
        top = 30
    } else if UIDevice.current.userInterfaceIdiom == .phone &&
        size.width > size.height {
        top = 0
    }
    myBar.frame = CGRect(x: 0, y: top,
                         width: Int(size.width), height: 44 + extraH)
    l.frame = CGRect(x: 20, y: Int(size.height) / 3 - 10,
                     width: Int(size.width) - 40, height: 30)
    b.frame = CGRect(x: 50, y: Int(size.height) / 3 + 30,
                     width: Int(size.width) - 100, height: 30)
}

func searchIfAny (s : String?) {
    if s != nil {
        l.text = "Search with «\(s!)»"
    } else {
        l.text = "Nothing to search for"
    }

}
```

MyView


```
// UISearchBarDelegate protocol

func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {
    l.text = "typed «\(searchText)»"
}

func searchBarBookmarkButtonClicked(_ searchBar: UISearchBar) {
    let a = UIAlertController(title: "BookMark",
                              message: "You typed on the BookMark symbol",
                              preferredStyle: .alert)
    a.addAction(UIAlertAction(title: "Sorbonne Université",
                              style: .default,
                              handler: {(a: UIAlertAction) -> Void in
        self.myBar.text = "Sorbonne Université"
        self.searchIfAny(s: self.myBar.text)
        }))
    a.addAction(UIAlertAction(title: "LIP6",
                              style: .default,
                              handler: {(a: UIAlertAction) -> Void in
        self.myBar.text = "LIP6"
        self.searchIfAny(s: self.myBar.text)
    }))
    a.addAction(UIAlertAction(title: "Cancel",
                              style: .default,
                              handler: nil))
    let vc = UIApplication.shared.windows[0].rootViewController
    vc?.present(a, animated: true, completion: nil)
}
```

MyView


```
func searchBarCancelButtonClicked(_ searchBar: UISearchBar) {
    l.text = "Search canceled" // when Cancel is pressed
    searchBar.resignFirstResponder()
}

func searchBarSearchButtonClicked(_ searchBar: UISearchBar) {
    self.searchIfAny(s: searchBar.text) // when Search is pressed
    searchBar.resignFirstResponder()
}

func searchBar(_ searchBar: UISearchBar,
               selectedScopeButtonIndexDidChange selectedScope: Int) {
        switch selectedScope {
            case 0: l.text = l.text!+" + «yes»"
            case 1: l.text = l.text!+" + «no»"
            case 2: l.text = l.text!+" + «maybe»"
            default: break
        }
    }
}
```

As a conclusion…

Yet another useful stuff

- Standard interaction
- What is the «semantics» of the «scope view»
  - **Change the search scale**
    e.g. local, remote

So, it is easy but…

… you still have to program the research