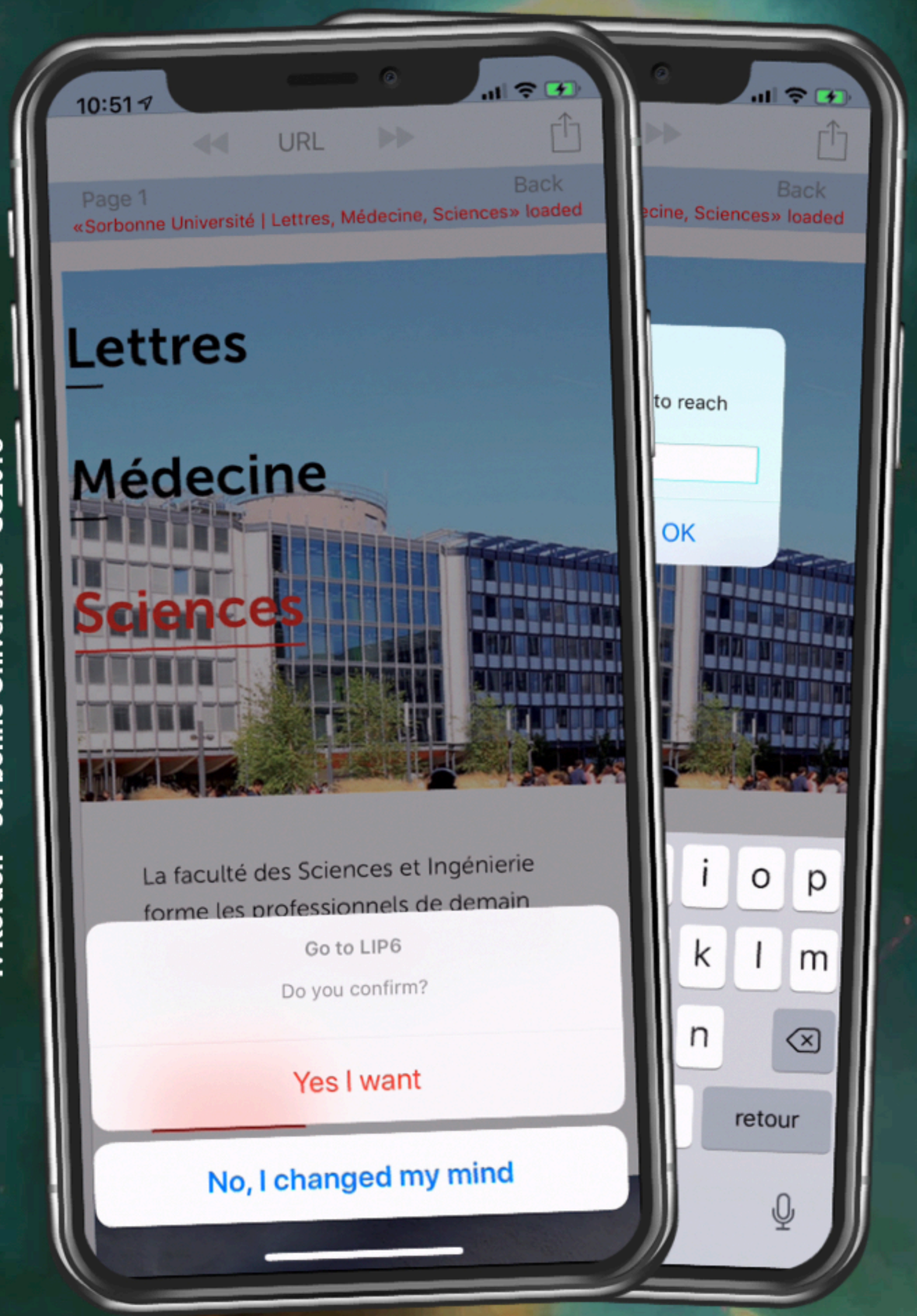
SCIENCES SORBONNE UNIVERSITÉ

UIAlertController

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…



An interaction mechanism

- Alert
- Action
  - **Multiple choice**
  - **Fetch of a string**
- Allocated without frame

Parameterized

- Title, text, buttons
- Text area
- 1 choice among N

Multiple call-backs

- You will like it

How does it work

I — Create a UIAlertViewController

- init(title:message:preferredStyle:) / alertControllerWithTitle:message:preferredStyle:

II — Configure it

- Add actions (buttons, text area, etc)

III — Activate it

- No attachment to a given view hierarchy
- Association to a *completion handler*

```
func present(_ viewControllerToPresent: UIViewController,
             animated: Bool,
             completion: (() -> Void)? = nil)
```

```
- (void)presentViewController:(UIViewController *)viewControllerToPresent
                      animated:(BOOL)flag
                    completion:(void (^)(void))completion;
```

Default alerts

- Minimalist
- You must enrich
  - Text area
  - buttons
    - «cancel»
    - «destructive»
    - «normal»
    - First is default action

Enriching a UIAlertController

Creating UIAlertActions

```
convenience init(title: String?,
                 style: UIAlertActionStyle,
                 handler: ((UIAlertAction) -> Void)? = nil)
```

```
+ (instancetype)actionWithTitle:(NSString *)title
                          style:(UIAlertActionStyle)style
                        handler:(void (^)(UIAlertAction *action))handler;
```

Adding UIAlertActions in an UIAlertController

```
func addAction(_ action: UIAlertAction)
```

```
- (void)addAction:(UIAlertAction *)action;
```

Enriching a UIAlertController

Creating

```
convenience i
                handler: ((UIAlertAction*)
```

Remind about actions!

Associated action in a handler

```
+ (instancetype)actionWithTitle:(NSString *)title
                          style:(UIAlertActionStyle)style
                        handler:(void (^)(UIAlertAction *action))handler;
```

Adding UIAlertActions in an UIAlertController

```
func a
- (voi
```

UIAlertActionStyle

default, cancel, destructive

UIAlertActionStyleDefault, UIAlertActionStyleCancel, UIAlertActionStyleDestructive

An example

ViewController

Sake of simplicity…

Code located in a ViewController

ViewController


```
import UIKit

class ViewController: UIViewController {

    private let a1 = UIAlertController(title: "My alert",
                                       message: "My message",
                                       preferredStyle: .alert)
    private let a2 = UIAlertController(title: "My action",
                                       message: "My message",
                                       preferredStyle: .actionSheet)

    private var textfield : UITextField?
    private let b1 = UIButton(type: .system)
    private let b2 = UIButton(type: .system)
    private let l = UILabel()
```

ViewController


```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view
    self.view = UIView(frame : UIScreen.main.bounds)
    self.view.backgroundColor = UIColor.white
    self.view.addSubview(b1)
    self.view.addSubview(b2)
    self.view.addSubview(l)
    b1.setTitle("Action", for: .normal)
    b1.backgroundColor = UIColor.lightGray
    b2.setTitle("Alert", for: .normal)
    b2.backgroundColor = UIColor.yellow
    b1.addTarget(self, action: #selector(action),
                 for: .touchDown)
    b2.addTarget(self, action: #selector(action),
                 for: .touchDown)
    b1.frame = CGRect(x: 20, y: 50,
                      width: UIScreen.main.bounds.size.width - 40,
                      height: 40)
    b2.frame = CGRect(x: 20, y: 100,
                      width: UIScreen.main.bounds.size.width - 40,
                      height: 40)
    l.frame = CGRect(x: 20, y: 150,
                     width: UIScreen.main.bounds.size.width - 40,
                     height: 40)
    l.textAlignment = .center
    // Prepare the alert (can be done later)
```

ViewController

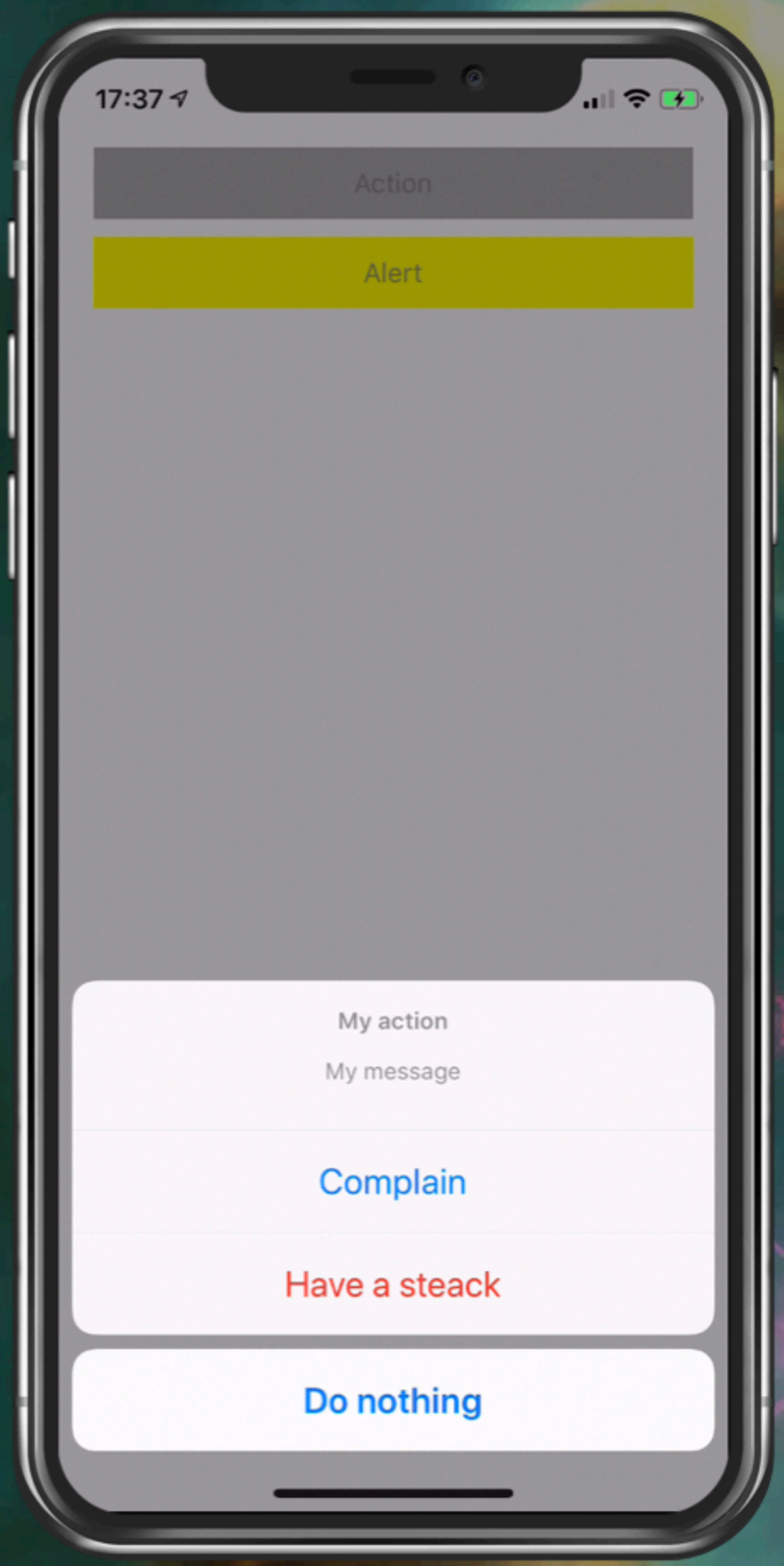

```
        // Prepare the alert (can be done later)
        a2.addAction(UIAlertAction(title: "Complain",
                                   style:.default,
                                   handler: {(a) -> Void in
                                    self.l.text = "I complain!!!!"}))
        a2.addAction(UIAlertAction(title: "Have a steack",
                                   style:.destructive,
                                   handler: haveASteack))
        a2.addAction(UIAlertAction(title: "Do nothing",
                                   style: .cancel, handler: nil))
        // Prepare the action
        a1.addAction(UIAlertAction(title: "OK",
                                   style: .default,
                                   handler: {(a) -> Void in
                                    self.l.text = ""}))
    }
```

ViewController


```
@objc func haveASteack (a : UIAlertAction) {
    l.text = "I go out having a steack"
}

@objc func action (sender : UIButton) {
    // when you have no access to a UIViewController…
    let vc = UIApplication.shared.windows[0].rootViewController
    if sender === b1 {
        vc?.present(a2, animated: true, completion: nil)
    } else {
        vc?.present(a1, animated: true, completion: {
            self.l.text = "waiting for a tap"
        })
    }
}
}
```

Analyzing the execution

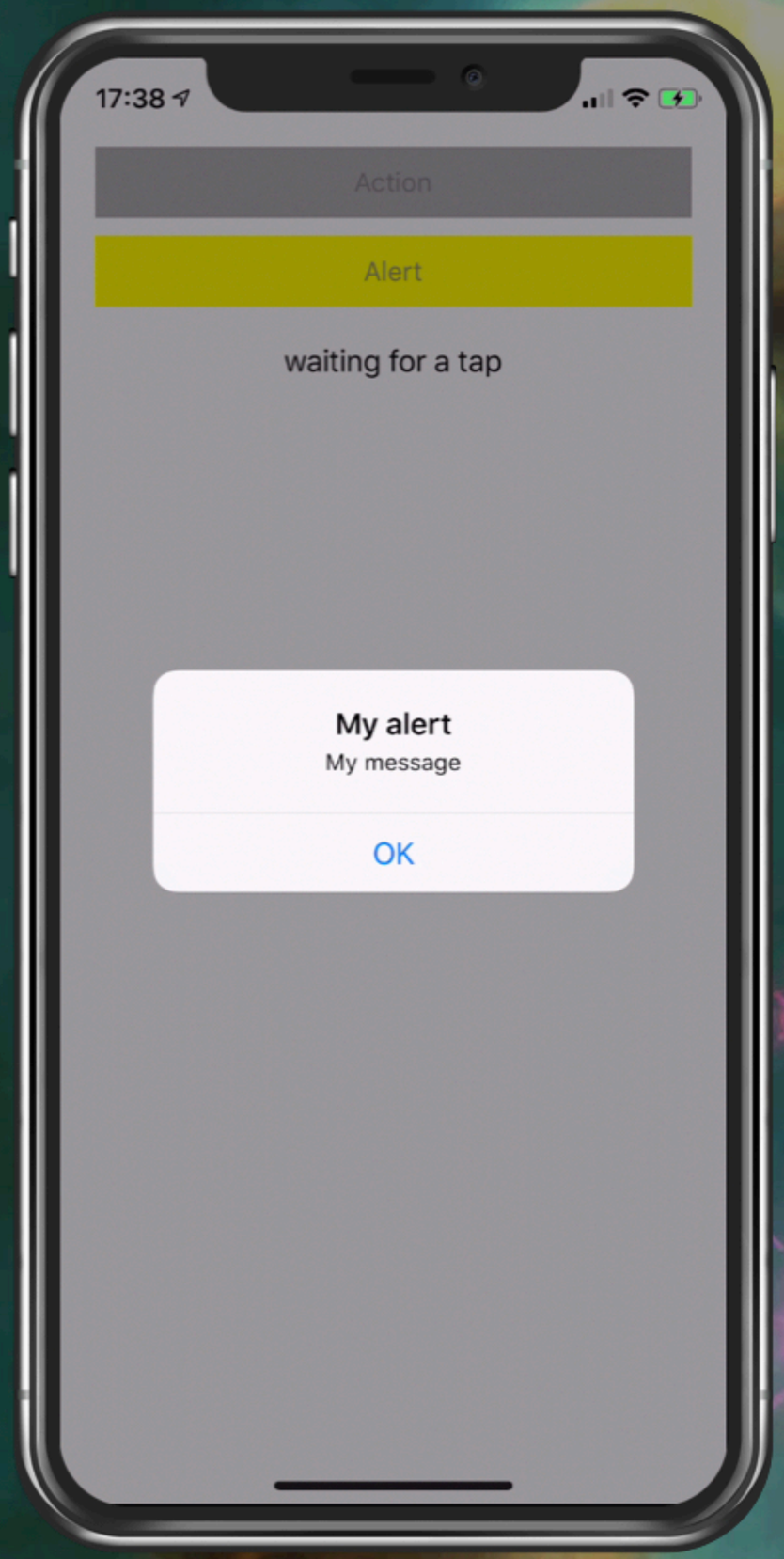


A cascade of call backs

tap on the Action button

call `action()`

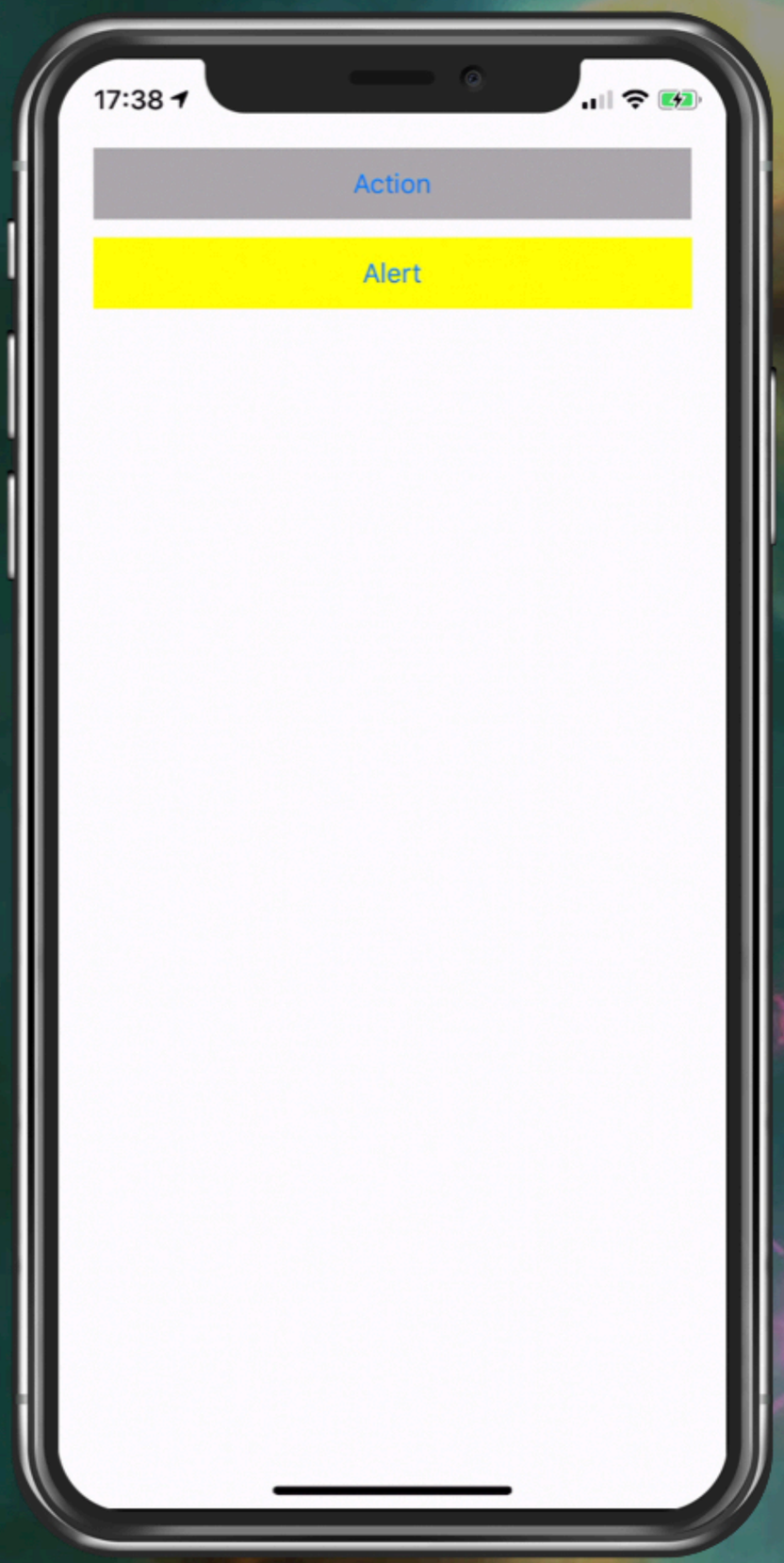
Analyzing the execution



A cascade of call backs

- tap on the Action button
  - call `action()`
- tap on «Have a steak»
  - call `haveASteack()`
    - Dismiss of the controller
- tap on the Alert button
  - call `action()`
    - call anonymous function

Analyzing the execution



A cascade of call backs

- tap on the Action button
  - call `action()`
- tap on «Have a steak»
  - call `haveASteack()`
    - Dismiss of the controller
- tap on the Alert button
  - call `action()`
    - call anonymous function
- tap on «OK»
  - Dismiss of the controller
    - call anonymous function

Retrieving a user input

Add a textField

```
func addTextField(configurationHandler: ((UITextField) -> Void)? = nil)
```

```
- (void)addTextFieldWithConfigurationHandler:
                    (void (^)(UITextField *textField))configurationHandler;
```

How to fetch the value?

- Set an external link to the embedded UITextField
  - **In the completion handler**
- Access via the textFields attribute (in he AlertController)
  - **Array of UITextField**

As a conclusion…

Another useful interaction mechanism

- Easy to use
- Allows to fetch simple text from the user
  - **To be detailed later**

Important for large devices

- Such alerts must be embedded in a «popover»
  - **Presentation option of the controller**
  - **See next video**