# Konstrukce aplikace v iOS

—

# MVC

IZA, Martin Hrubý, FIT VUT, 2018

# Literatura

- Zdroje Apple.
- Web: objc.io
- Ole Begeman: Understanding UIScrollView

# Úvod

- Další koncepty Swiftu budeme probírat v kontextu aspektů programování aplikací.
- Cílem je poznat základy konstrukce aplikace pro iOS.
- Přehled View-Controllerů (VC).
- UIKit — knihovna pro psaní iOS/tvOS app.
  - Swift Standard Library. Foundation. (Cocoa Touch, Obj-C).

# XCode

- IDE pro programování aplikací.
- Editor pro interaktivní návrh konceptu obrazovek aplikace (Storyboard).
- Editace rozvržení Views (geometrie) v okně apod.
- Xcode generuje základní rámce zdrojáků aplikace.

# První kontakt s programem

- XCode nabídne vygenerovat zdrojáky pro základní koncepty UI:
  - Single View, Tab bar, Master-Detail.
- Především generuje třídu pro UIApplication delegate.
  - Případně infrastrukturu pro CoreData a testy.

# UIApplication

- Knihovní třída. Nepředpokládá se odvozování vlastní. Singleton (UIApplication.shared).
- Rozhraní mezi OS a aplikací (události).
- Ovládá základní podobu aplikace v systému (ikonka, horní lišta apod.).
- Referencuje: *app-delegáta*, windows a keyWindow.

# Protokoly a delegáti

- Forma ustanovení dlouhodobější synchronní komunikace mezi objekty A (např. knihovní) a B (např. aplikační).
- Synchronní/asynchronní zaslání zprávy.
- Je třeba ustanovit komunikační *protokol*.
  - Třída implementuje *protokol*. Pak může být delegátem.
  - Povinné/volitelné prvky protokolu. Kontrola překladačem, dynamická kontrola před odesláním zprávy.
  - Protokol jakou součást datových struktur a algoritmů.

# Demo protocol & delegate

```
// funkce z protokolu jsou povinne
protocol MujProtokol {
    //
    func udelejNeco() -> Int;

    //
    var jmeno : String { get }
}

// funkce z protokolu muzou byt nepovinne
@objc protocol MujProtokolD {
    //
    @objc optional func udelejNecoDobrovolne() -> Int;
}
```

```
class TridaZDROJ {
    // weak; MujProtokol : class
    var delegate: MujProtokol?

    //
    func necoDelam() {
        // informuju delegata
        delegate?.udelejNeco()
    }
}
//
class TridaCIL : MujProtokol {
    // pak se hodi zdroj.delegate jako weak
    var zdroj: TridaZDROJ?

    //
    func udelejNeco() -> Int {
        //
        return 3
    }

    //
    var jmeno: String {
        return "Petr"
    }
}
```

# Referencování delegáta

- strong/weak/unowned reference — zvážit cykly a multi-thread přístup.
- unowned(unsafe) var delegate: UIApplicationDelegate? { get set }
- Referoncování celkově:
  - strong — implicitní.
  - weak — vždy optional. Reference nulována, když cíl zanikne.
  - unowned — není optional. Zánik cíle. Zombie objekt.

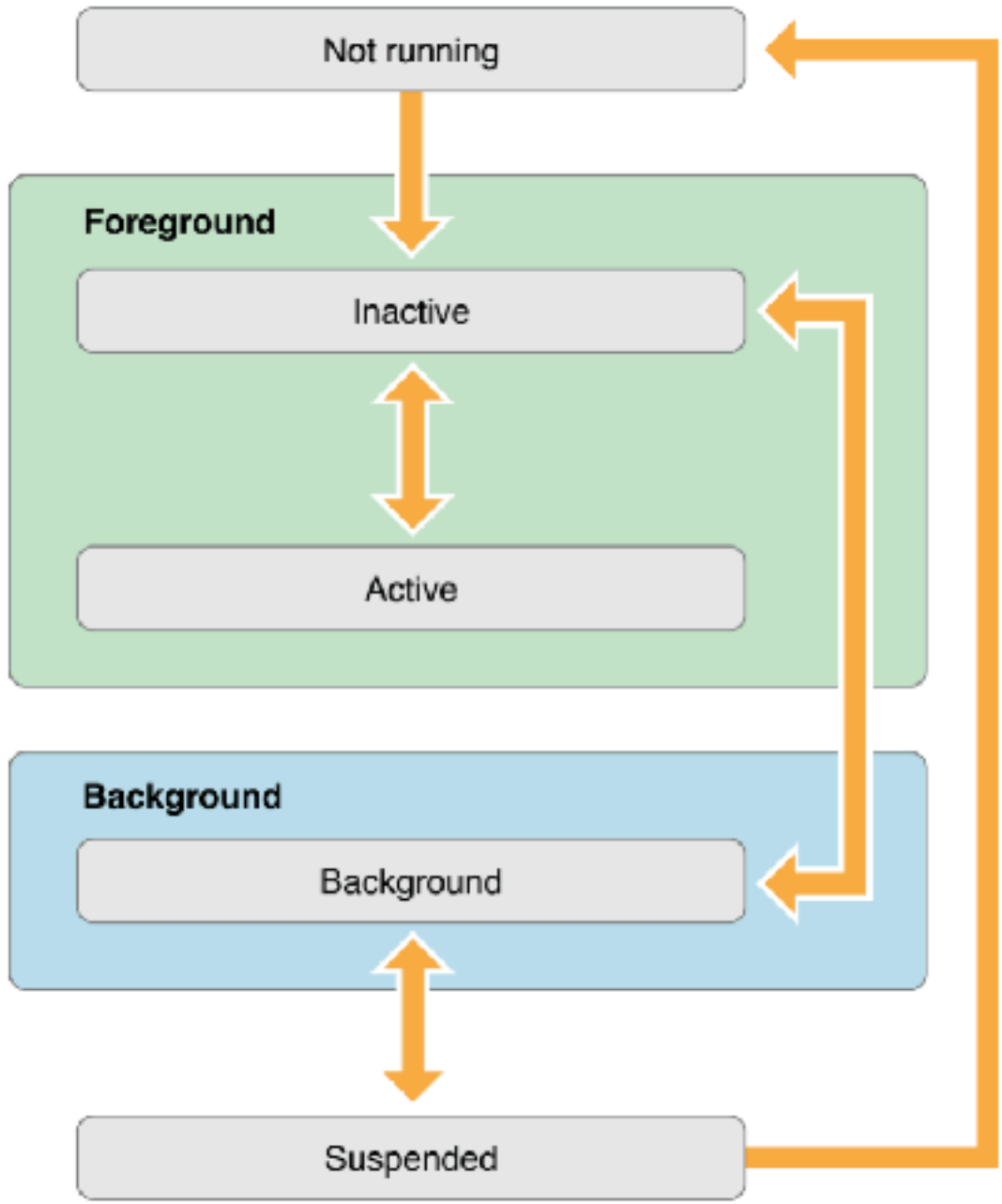
# Demo protokol & struktury

```
//
class Trida : Hashable {
    //
    var cisloObjektu : Int = 0

    //
    var hashValue: Int {
        return cisloObjektu
    }
}

extension Trida : Equatable {
    //
    static func ==(lhs: Trida, rhs: Trida) -> Bool {
        //
        return lhs.cisloObjektu == rhs.cisloObjektu
    }
}

//
typealias Trida_set = Set<Trida>
typealias Trida_dict = [Trida:String]
```

# UIApplication delegate

- Singleton. Může referencovat objekty globálního charakteru (typicky data, CoreData, apod.)
  - didFinishLaunchingWithOptions. Neblokovat příliš start aplikace (globální fronta).
- Události změny stavu aplikace — popředí/pozadí, will/did,
- Referencuje window, window referencuje view-controller, …

# Stavy aplikace

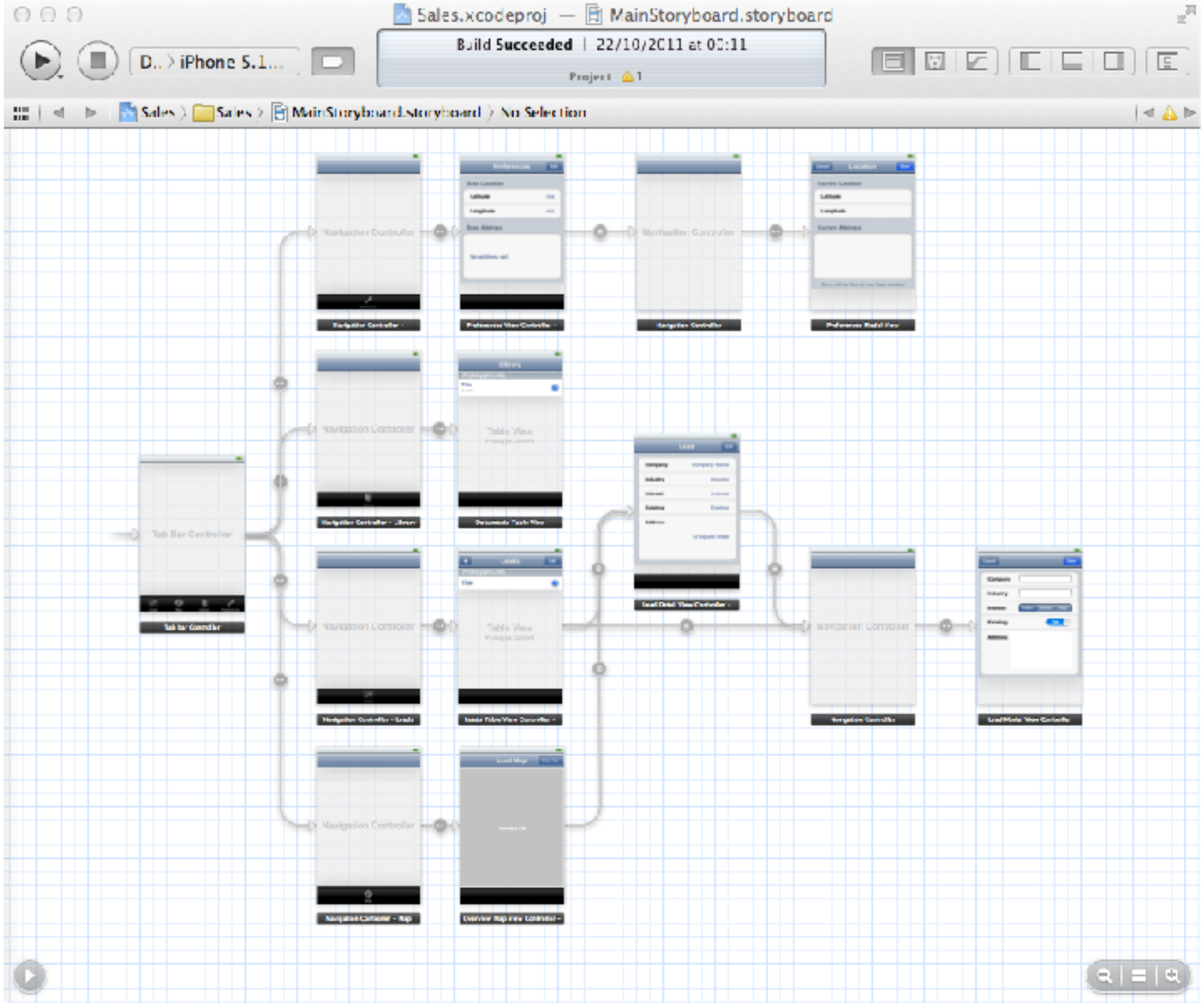


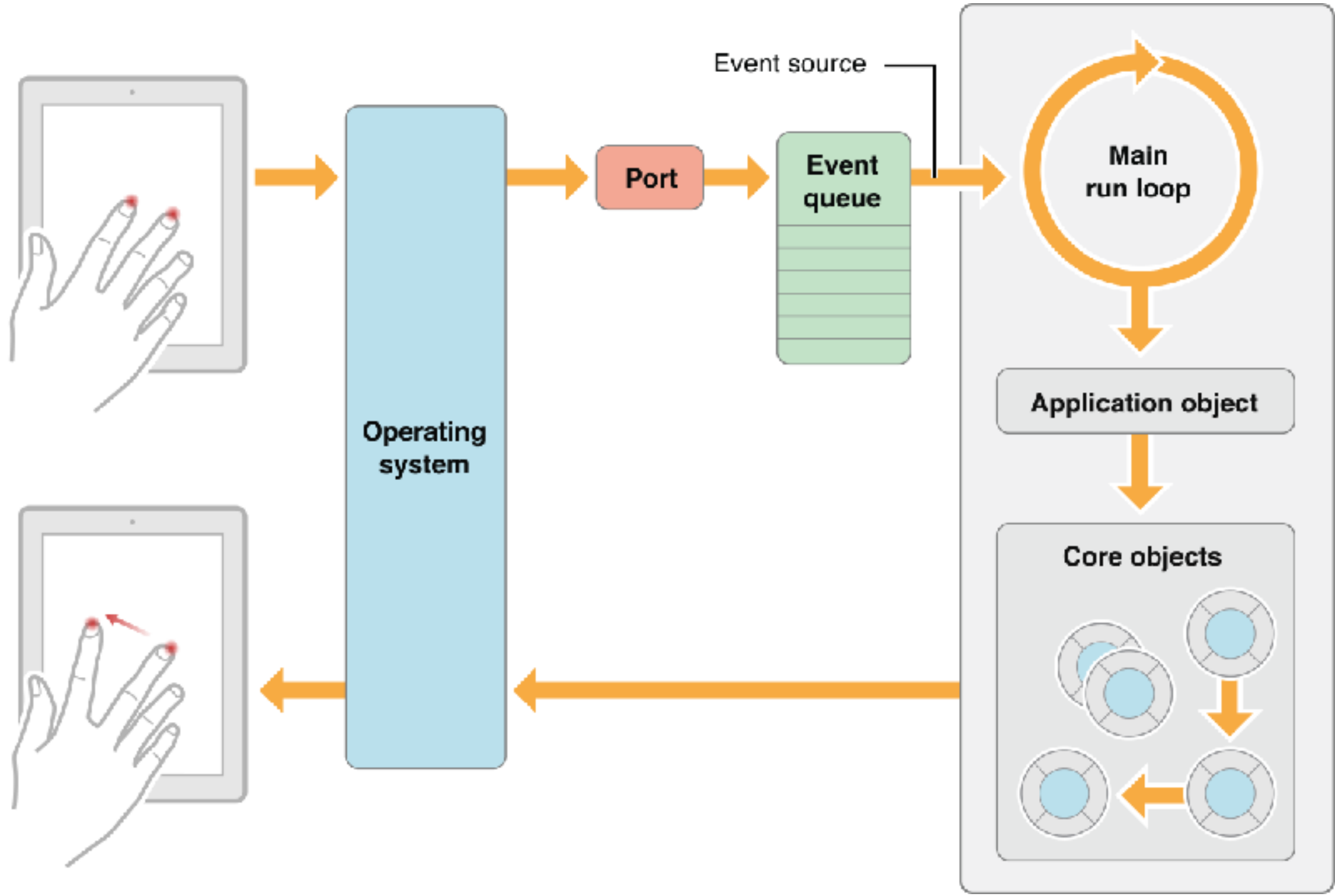
# Stavy aplikace

- Not running — nespuštěna nebo ukončena.
- Popředí (je viditelná):
  - Active — vykonává kód, přijímá údálosti.
  - Inactive — nepřijímá údálosti, běží. Typicky krátký moment.
- Pozadí (není viditelná): smí vykonávat kód.
- Suspended — je v paměti, neběží. Systém smí aplikaci kdykoliv ukončit.

# Systémové objekty aplikace

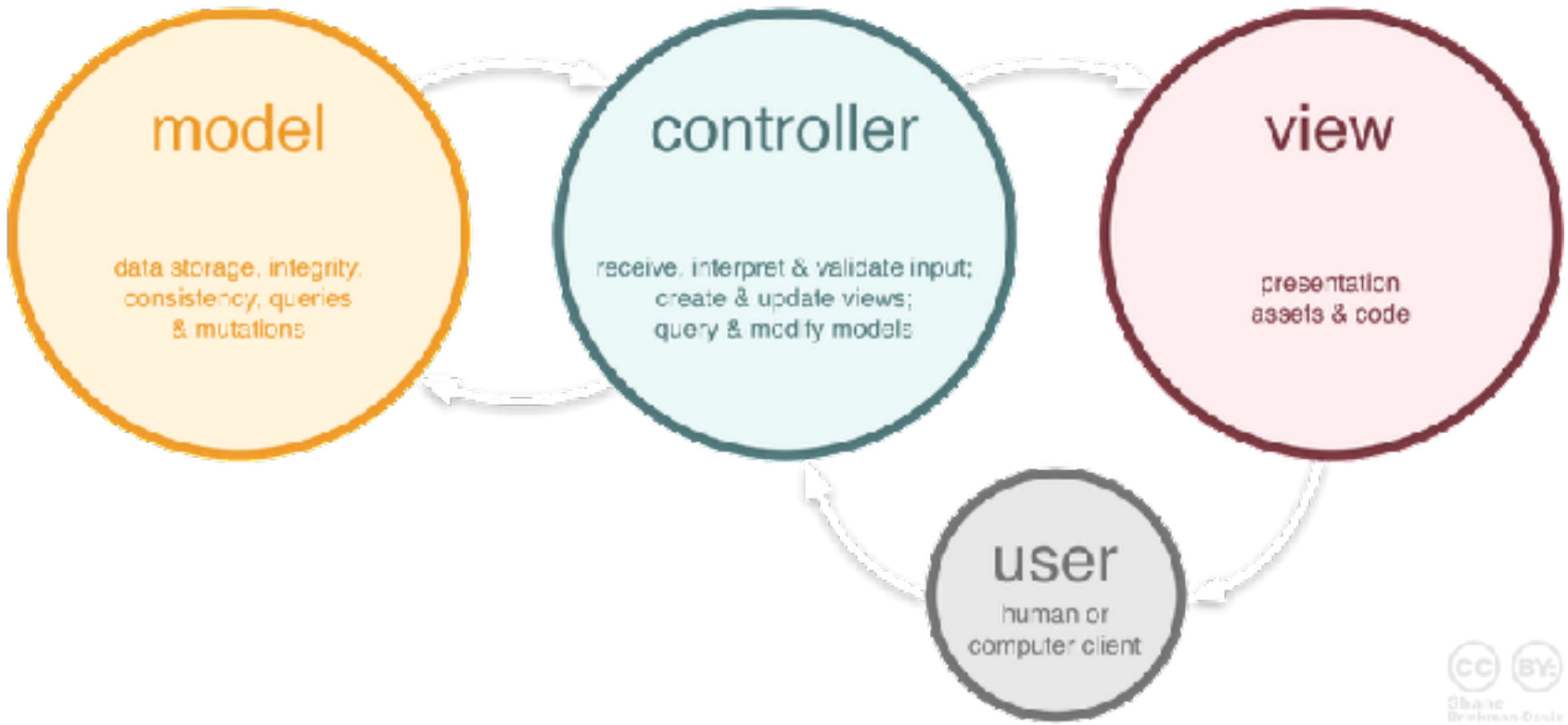
- UIDevice — metadata, orientace, baterie, typ zařízení (phone, pad, tv, carPlay).
- UIScreen — obrazovka zařízení.
  - Hlavní a sada vedlejších. Definuje ”bounds” (orientace).
- UIWindow — kontejner pro UIViews.
  - rootViewController, přeposílá zprávy.
  - Další UIWindow — Alerts, (popovers).


Sales.xcodeproj — MainStoryboard.storyboard
Build Succeeded | 22/10/2011 at 03:11
Project 1
D.. > iPhone 5.1...
Sales > Sales > MainStoryboard.storyboard > No Selection
Tab Bar Controller
Table View
Navigation Controller


Event source
Port
Event queue
Main run loop
Application object
Core objects
Operating system

# Model-View-Controller (MVC)

- Model — datová část aplikace.
- View — zobrazovací prvky do View / Window.
- Controller — řídicí objekty.



# MVC

- MVC koncept byl poprvé použit ve Smalltalku (Xerox) — dále okno, myš, … návaznost na Mac.
- Jde o balík obecně více objektů (M, V, C).
- V systému MVC je reprezentantem tria M-V-C vždy Controller.
  - tj. ostatní objekty referencují Controller,
  - controller drží reference na view a model.

# MVC v aplikaci

- Programujeme (typicky) Model a Controller.
- Je snaha automatizovat přesun dat z Modelu do Views (Bindings, macOS).
- Znovupoužitelnost ViewControllerů (VC).
  - dědičnost,
  - konektory na views (IBOutlets),
  - konektory na model.
- Typizované VC. Styly aplikace.

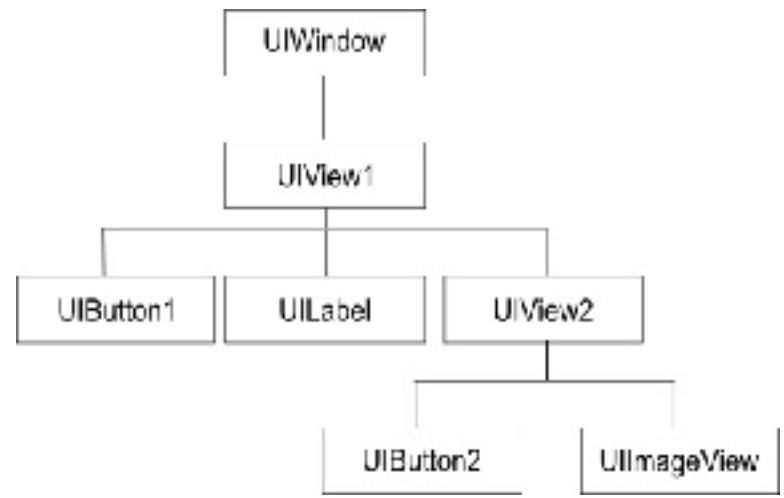
# Views

- UIView, UILabel, UISwitch, UITableViewCell,
- Superview, subviews — hierarchie.
- Smyslem je skládat UI z knihovních komponent (versus drawRect:).
- Provádět změny do UI smí pouze hlavní vlákno.

```
class ViewController: UIViewController {

    @IBOutlet var textik : UILabel?

    func inicializuj() {
        textik?.text = "Napis hello";
    }
```

# Topologie views

- View má: superview a subviews:
  - superview.addSubView(myView)
  - myView.removeFromSuperview()
- Předpokládáme stromovou topologii.



# Geometrie View

- *Frame* — definuje umístění *view* v jeho *superview*.
- *Bounds* — definuje velikost view a jeho lokální souřadný systém.
- Frame a bounds nemusí mít stejné velikosti.
- CGRect(x: y: width: height:)
- Souřadnice (0, 0) — levý horní roh.
- Pozor: jednotkou souřadnice NENÍ 1 pixel.

# Vytvoření objektu view

- var label = UILabel(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
  - Definuje frame,
  - nastaví bounds na origin=(0,0), size=(200,21).
- view.addSubView(label)

# Proces rasterizace a kompozice

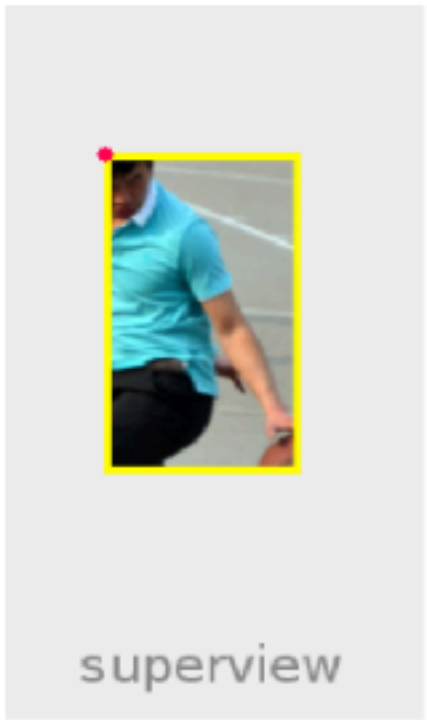
- Rasterizace (drawRect:), bounds((x,y), (w,h)).
  - Alokuje se prázdná bitmapa velikosti (w,h).
- Představa projekce rasterizace:
  - V abstraktním souřadném systému grafického objektu se objekt (myšleně) vykreslí s počátkem (0, 0).
  - Tento (myšlený) obraz se obtiskne do bitmapy (w, h), kterou přiložíme do bodu (x, y) obrazu.
  - Obecně *pouze část obrazu* se obtiskne do bitmapy (w, h).

# Kompozice

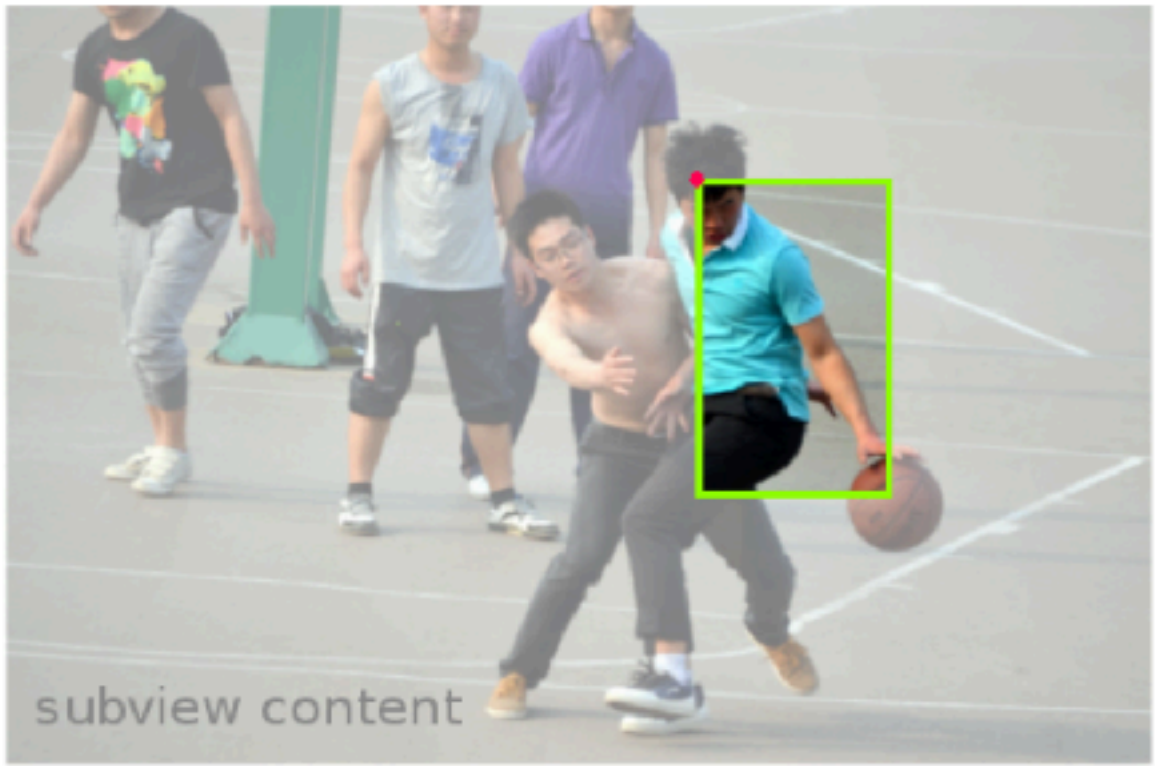
- Mějme superview S, S.frame, S.bounds,
- Představa: kompozice bitmapy X subview je pokračování rastrizace do myšleného obrazu.
  - Tj. do myšleného souř. sys. umístím na X.frame.origin bitmapu X (oříznutou X.frame.size).
  - Pro S mám bitmapu S.bounds.size, kterou obtisknu umístěnou do S.bounds.origin.
  - Efektivně X jde na X.frame.origin - S.bounds.origin

```
Frame
    origin = (40, 60)
    width = 80
    height = 130

Bounds
    origin = (280, 70)
    width = 80
    height = 130
```

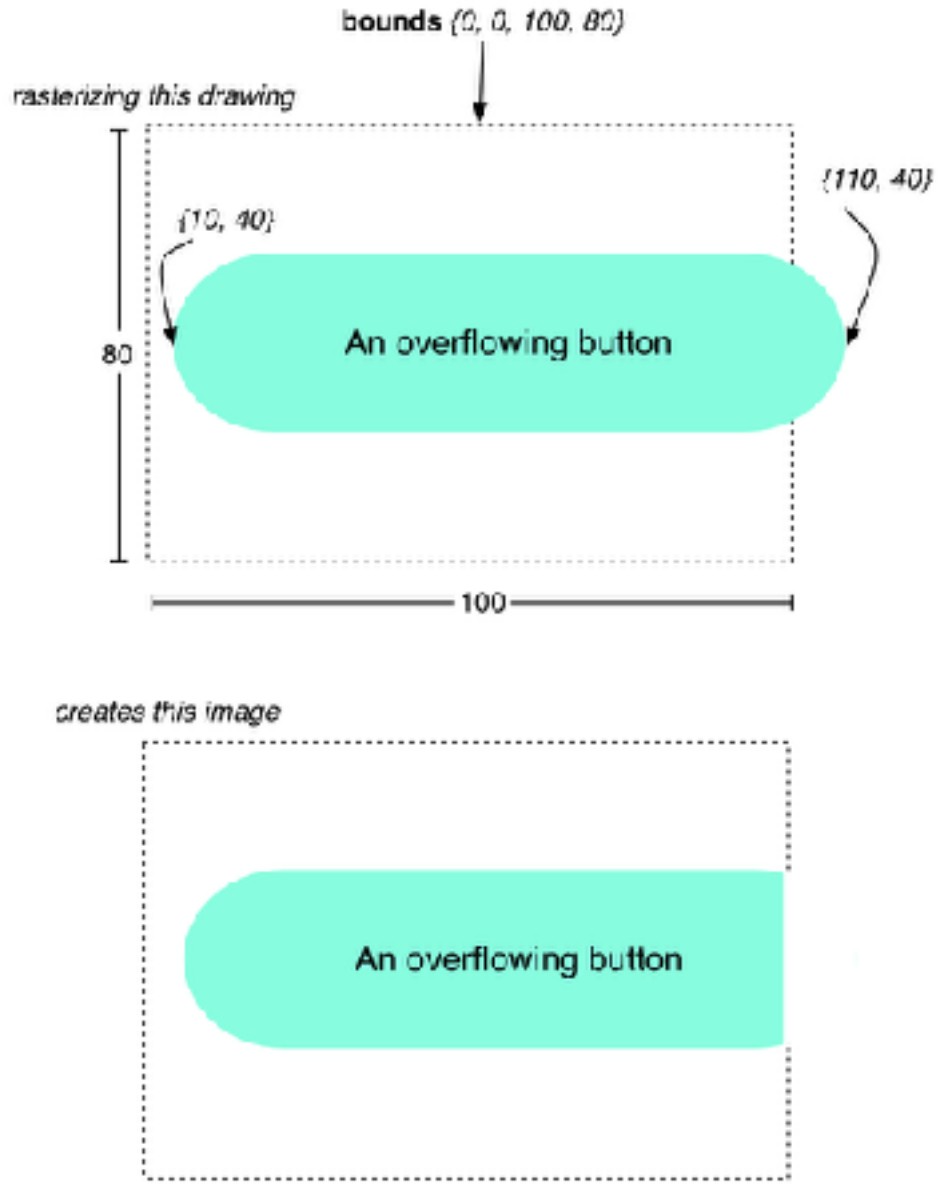
## Frame



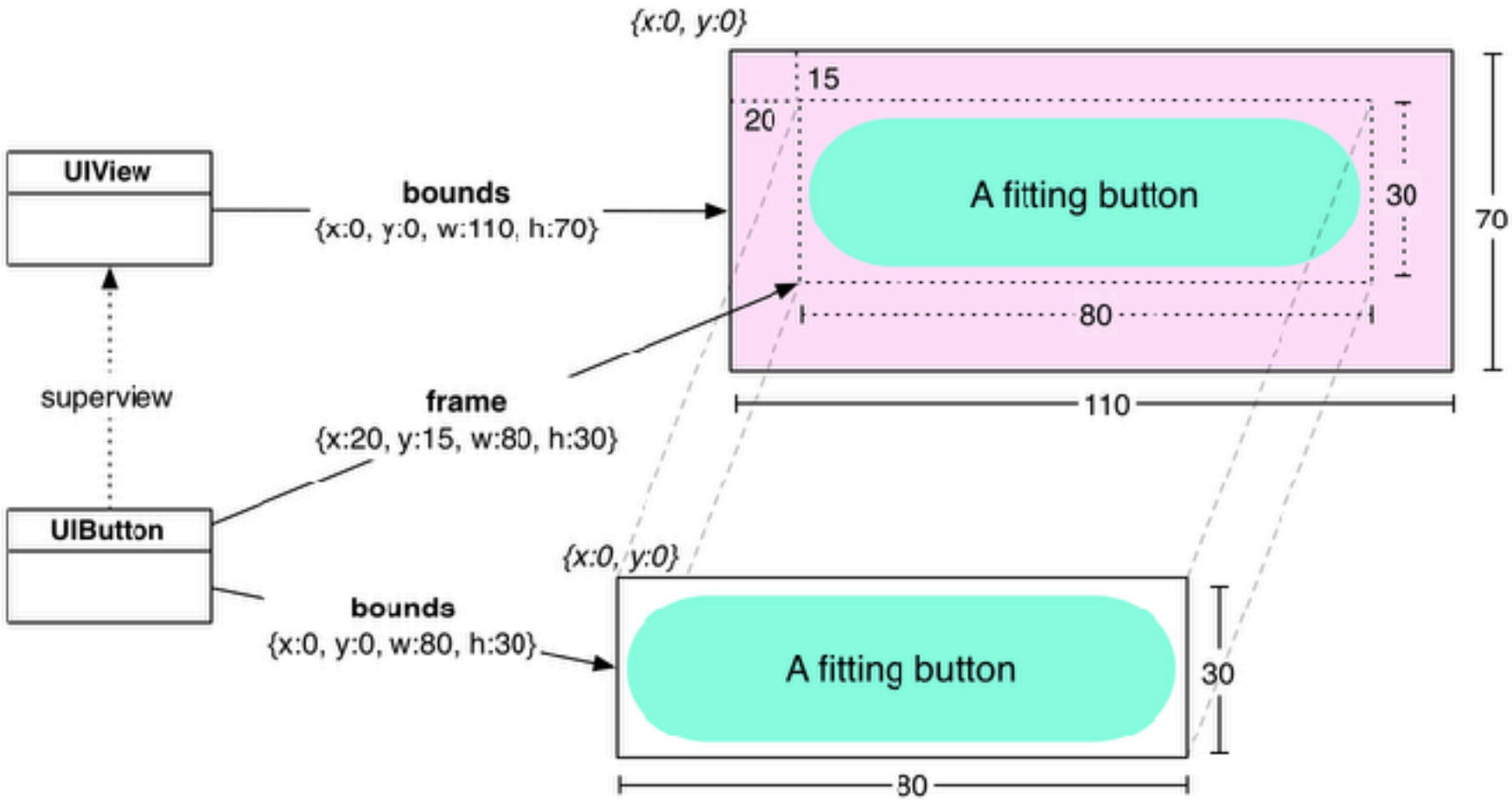

## Bounds



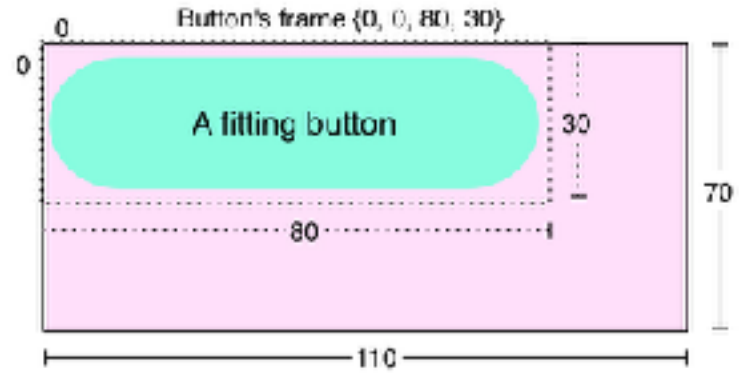
# Rastrizace do bounds

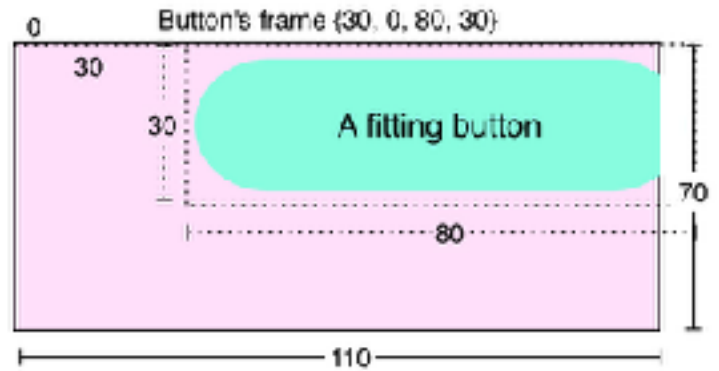


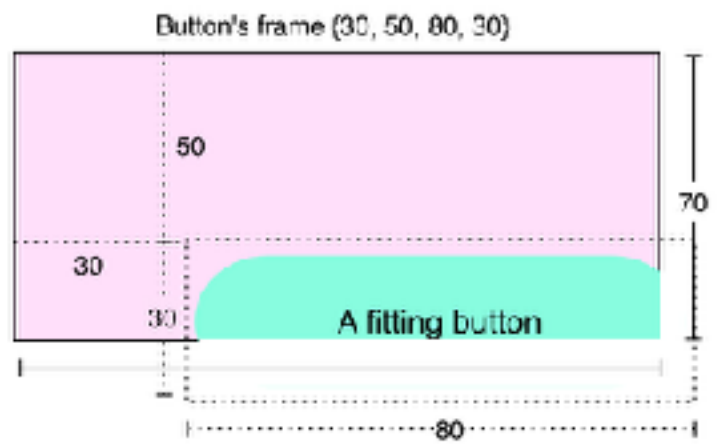
# Kompozice view



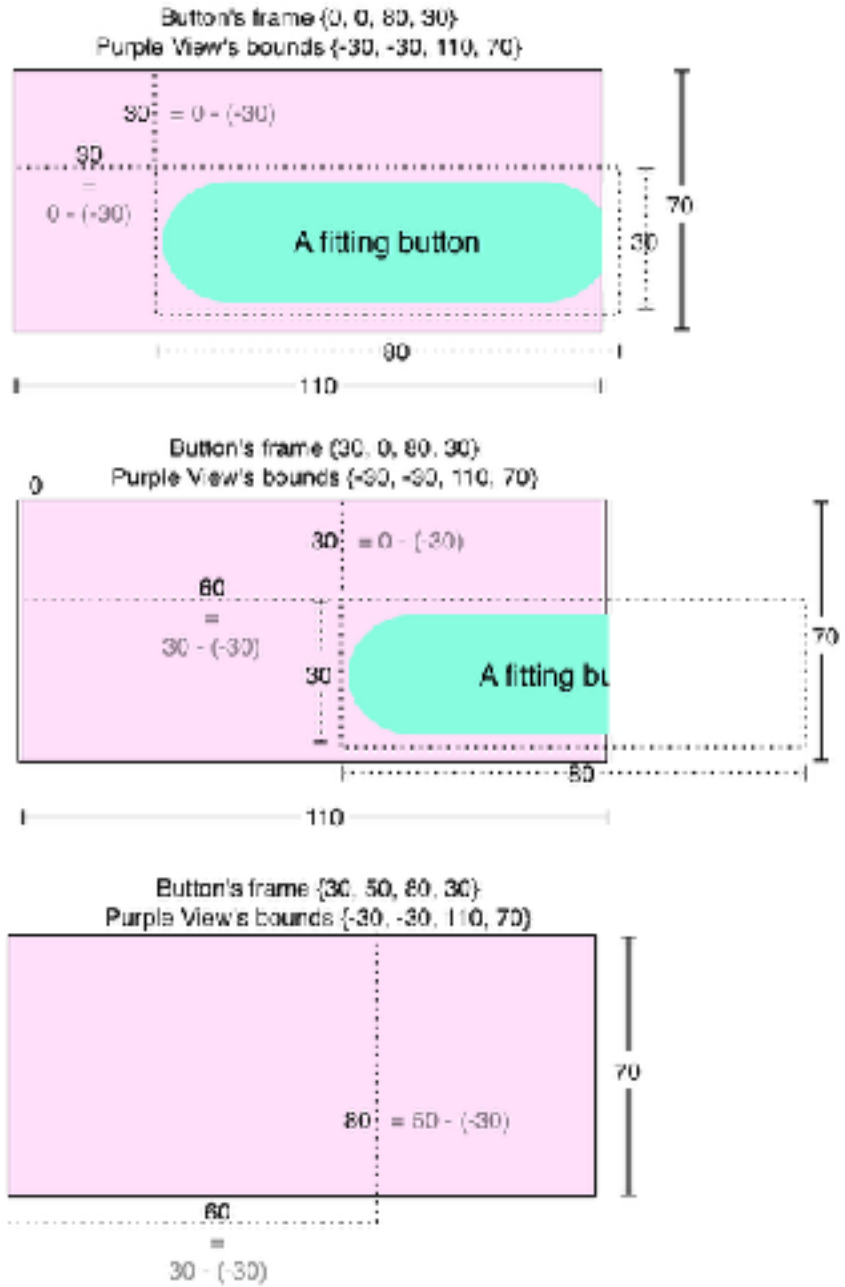
# Kompozice view



# Kompozice, superview bounds

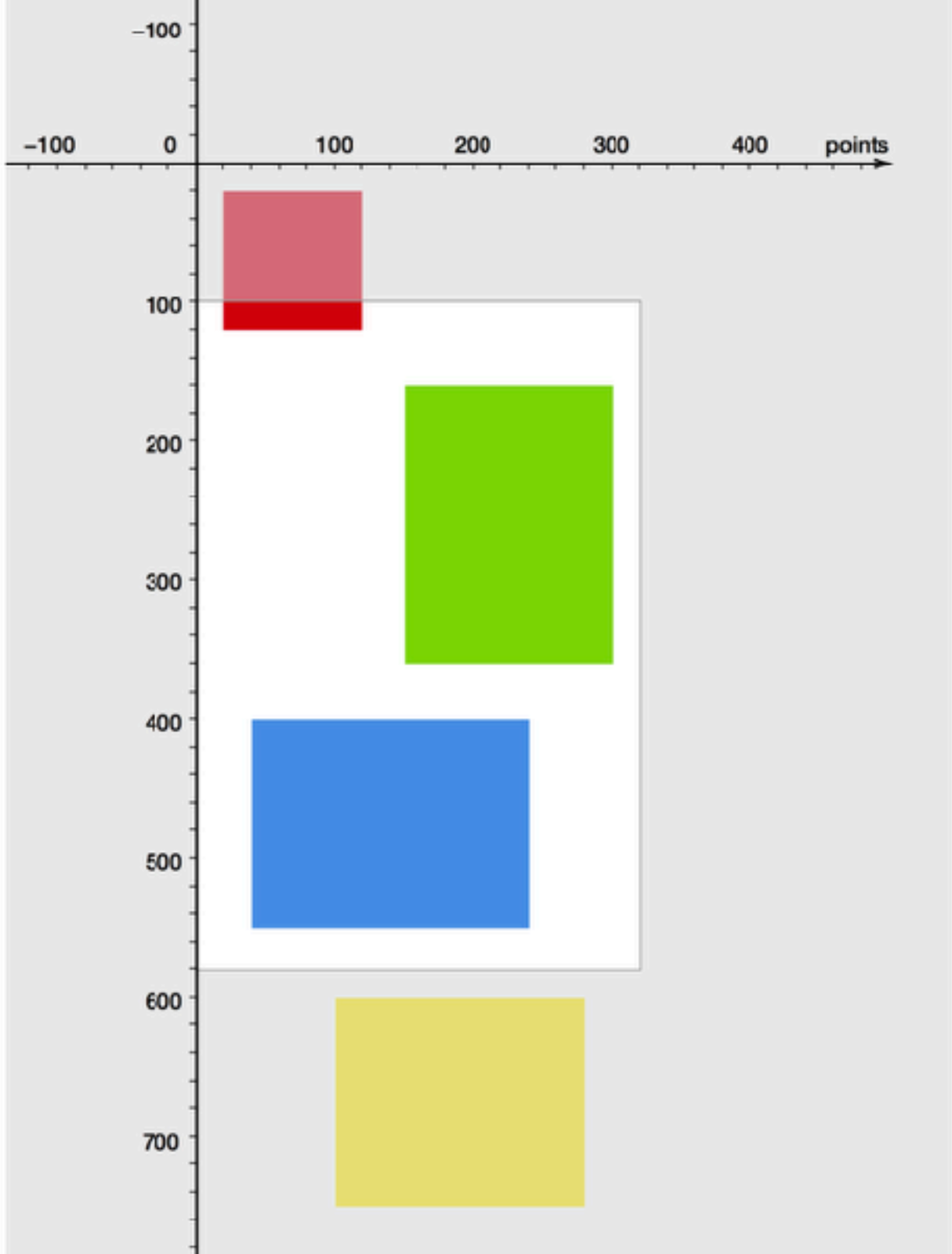


# Jak pracuje UIScrollView

- Potřebujeme View s rozměrem přesahujícím rozměr obrazovky. Typicky TableView, ImageView, …
- Takže na událost posuvu pohledu (výřez do ScrollView) bychom asi modifikovali frame.x/y všech subviews…to by bylo ovšem peklo…
- Nějaký nápad, jak efektivně změnit výsledek rastrizace ScrollView při scrollování?

# ScrollView

- Je to opět view, tj. má frame a bounds (zobrazovanou velikost).
- Chceme však virtualizovat jeho velikost, tj. contentView > bounds.size;
- contentOffset — viditelná část view (bounds) se nastaví levým horním rohem na contentOffset.
  - tj. zprostředkovaně nastavuji bounds.origin.


−100
−100
0
100
200
300
400
points
100
200
300
400
500
600
700

# Stav objektů View

- Uvažujme UILabel (property: text), odvozen od UIView.
- V proceduře kontroleru chceme změnit obsah UILabelu (prováděno nějakým vláknem).
- o.text = newValue;
- o.setNeedsDisplay()
- Změny obsahu UI se provádí pouze vláknem zpracovávajícím MainQueue!

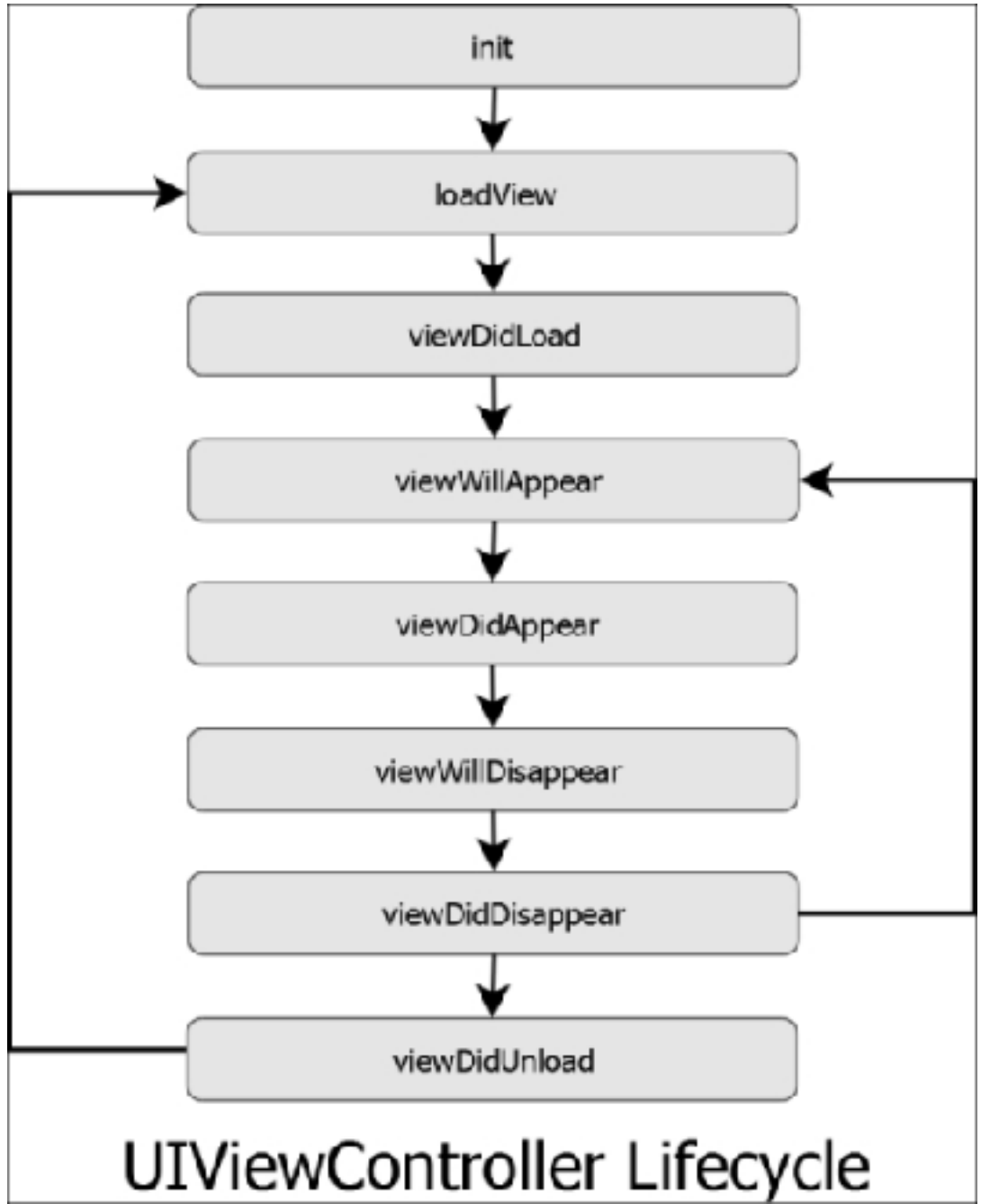
# Stav objektů view

- Když uživatel změní stav objektu view, view posílá zprávu svému ”target” (target-action mechanism). Cílem je typicky controller.
- UISwitch (od UIControl).
- Typicky:
  - func nejakaZmenaTlacitka(sender: UISwitch);
  - @IBAction

# ViewController (VC)

- Je vlastník objektu ”view”. Ten dále tvoří hierarchii dalších subview.
- V aplikaci se manipuluje s VC.
- UIWindow referencuje svůj ”root” VC.
  - Ten může organizovat další VC: navigation, tabbar, …
- UIViewController.

# Životní cyklus ViewController

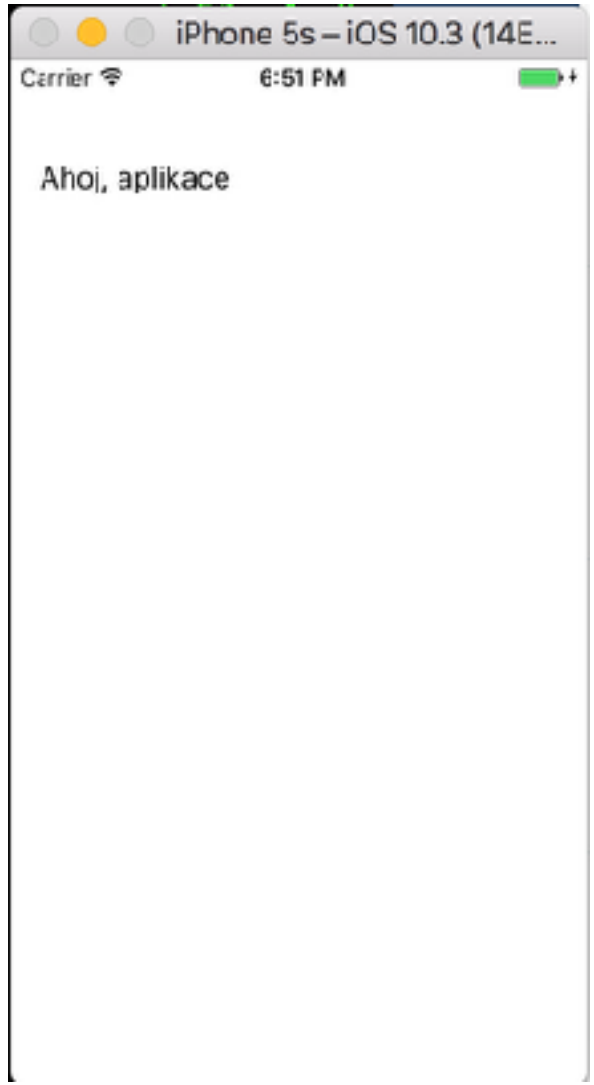


# Single-VC aplikace

- UIWindow.rootViewController je nějaký jednoduchý VC, např. UITableViewController
- Jinak uvažujeme vždy přepínání VC prostřednictvím nějakého řídicího VC (container VC): Navigation, TabBar, …

# M-V-C, Základní demo

```
class MujModel {
    var obsah : String = "Ahoj, aplikace"
}

class ViewController: UIViewController {

    @IBOutlet var lei : UILabel?

    let mujmodel = MujModel()

    override func viewDidLoad() {
        super.viewDidLoad()

        //
        lei?.text = mujmodel.obsah
    }
}
```



# M-V-C, KVO verze

```
//
class MujModel : NSObject {
    dynamic var obsah : String = "Ahoj, aplikace"
}
//
class ViewController: UIViewController {
    @IBOutlet var lei : UILabel?
    let mujmodel = MujModel()
    //
    override func viewDidLoad() {
        super.viewDidLoad()

        lei?.text = mujmodel.obsah

        mujmodel.addObserver(self, forKeyPath: #keyPath(MujModel.obsah), options:
[.new, .old], context: nil);
    }
    //
    override func observeValue(forKeyPath keyPath: String?,
                               of object: Any?,
                               change: [NSKeyValueChangeKey : Any]?,
                               context: UnsafeMutableRawPointer?)
    {
        DispatchQueue.main.async {
            self.lei?.text = self.mujmodel.obsah
        }
    }
}
```

# M-V-C, didSet verze

```
class MujModel {
    //
    weak var vc : ViewController? = nil

    var obsah : String = "Ahoj, aplikace" {
        didSet {
            //
            vc?.lei?.text = obsah;
        }
    }
}
```

```
class ViewController: UIViewController {
    //
    @IBOutlet var lei : UILabel?
    //
    let mujmodel = MujModel() ;
    //
    override func viewDidLoad() {
        super.viewDidLoad()
        //
        mujmodel.vc = self;
        //
        lei?.text = mujmodel.obsah
    }
```

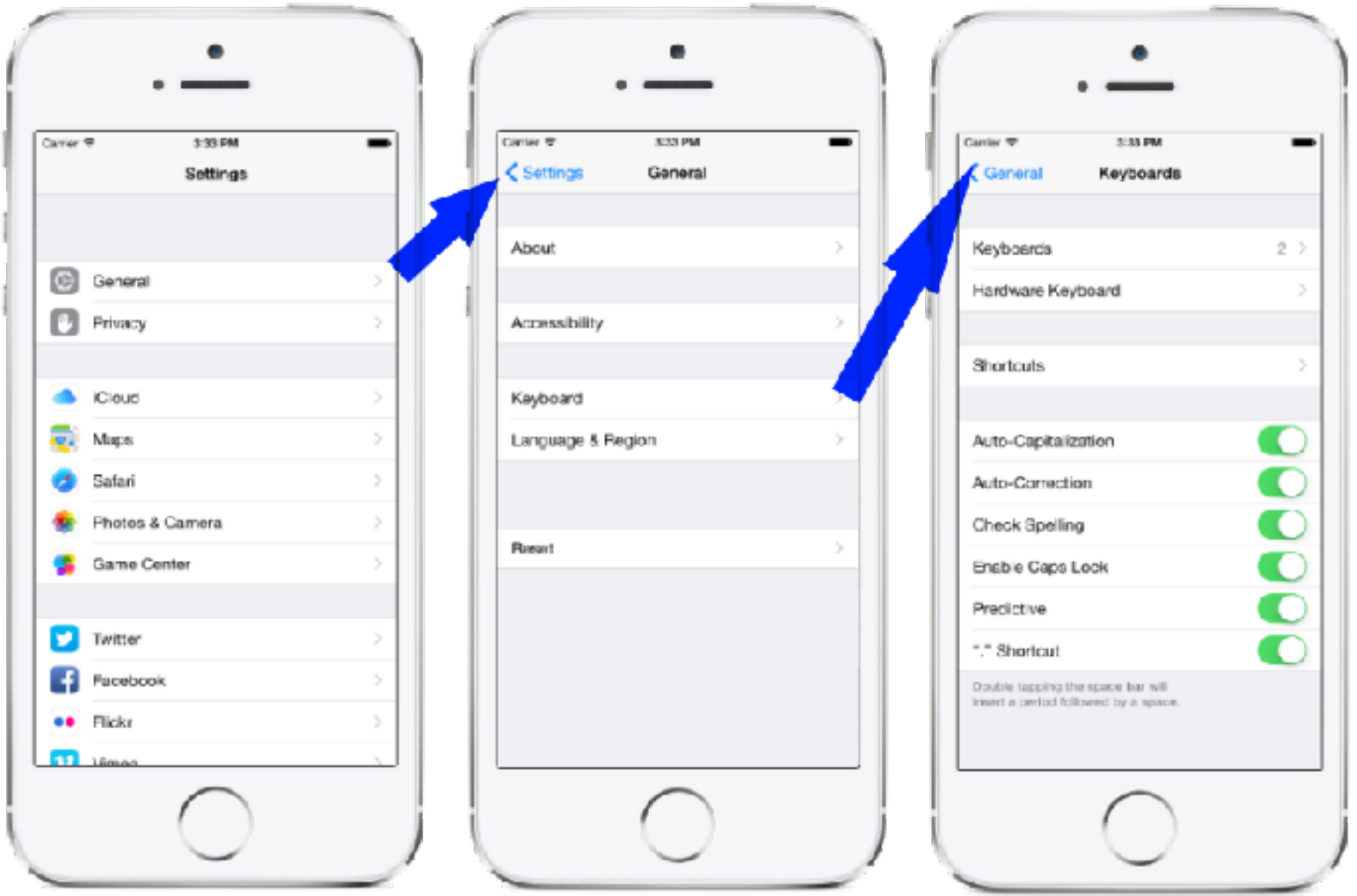
# M-V-C, NotificationCenter

```
var obsah : String = "Ahoj, aplikace" {
        didSet {
            //
            NotificationCenter.default.post(name:
                NSNotification.Name(rawValue: "mojeZprava"), object:
nil)
        }
    }
//
    override func viewDidLoad() {
        super.viewDidLoad()

        //
        NotificationCenter.default.addObserver(self, selector:
#selector(update), name: NSNotification.Name(rawValue: "mojeZprava"),
object: nil)
    }

    func update() {
        //
        lei?.text = mujmodel.obsah
    }
```

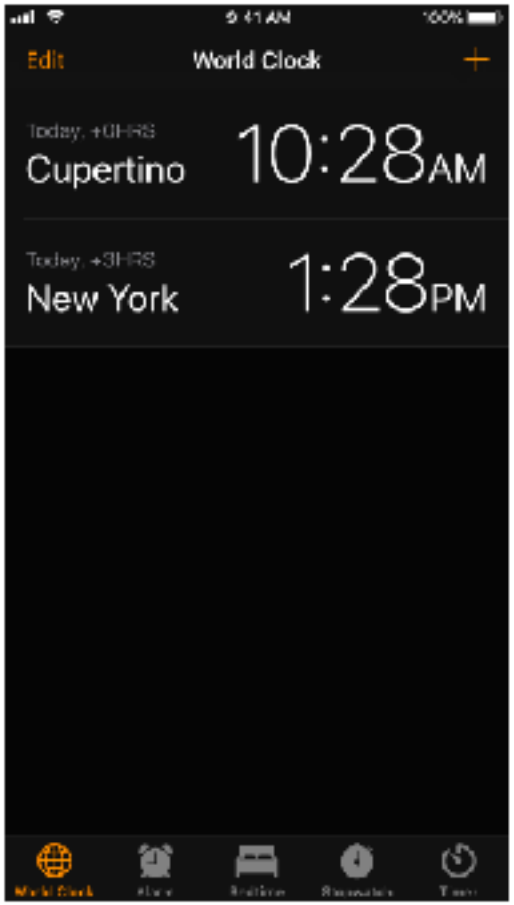
# Multi-VC aplikace

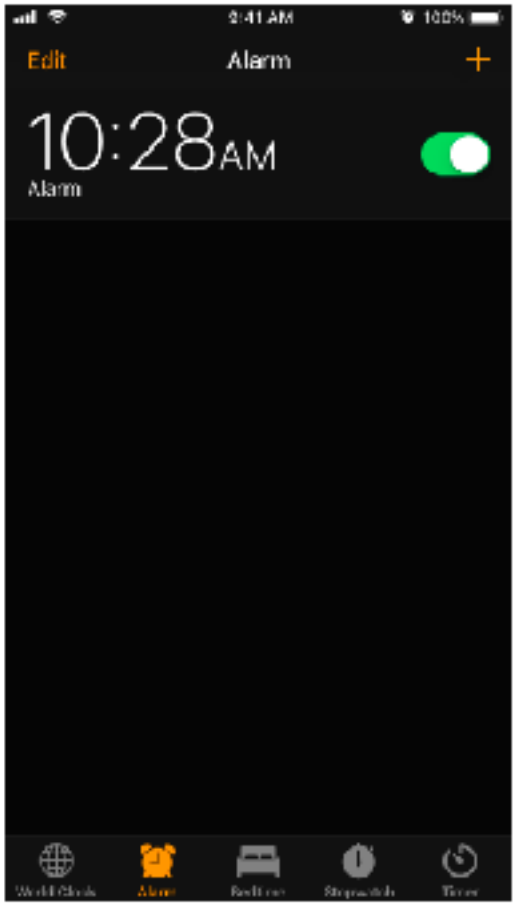
- UINavigationController,
- UITabBarController,
- Styl Master-Detail — 2 VC spojené přes UISplitViewController.

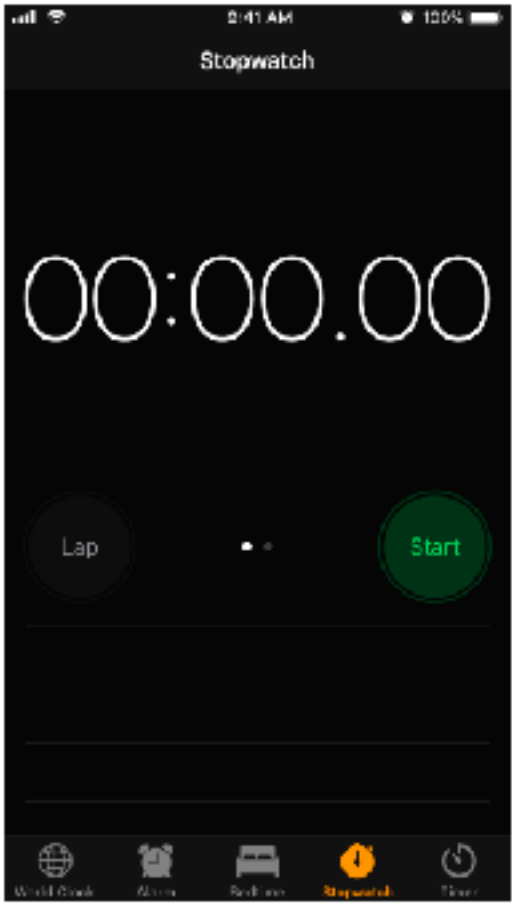
# Navigation VC

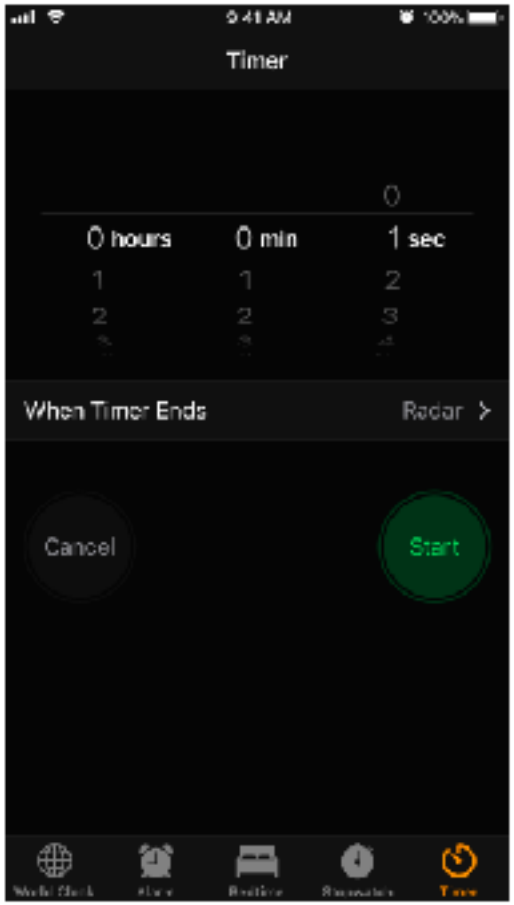


# TabBar VC


Edit
World Clock
Today, +0HRS
Cupertino
10:28AM
Today, +3HRS
New York
1:28PM

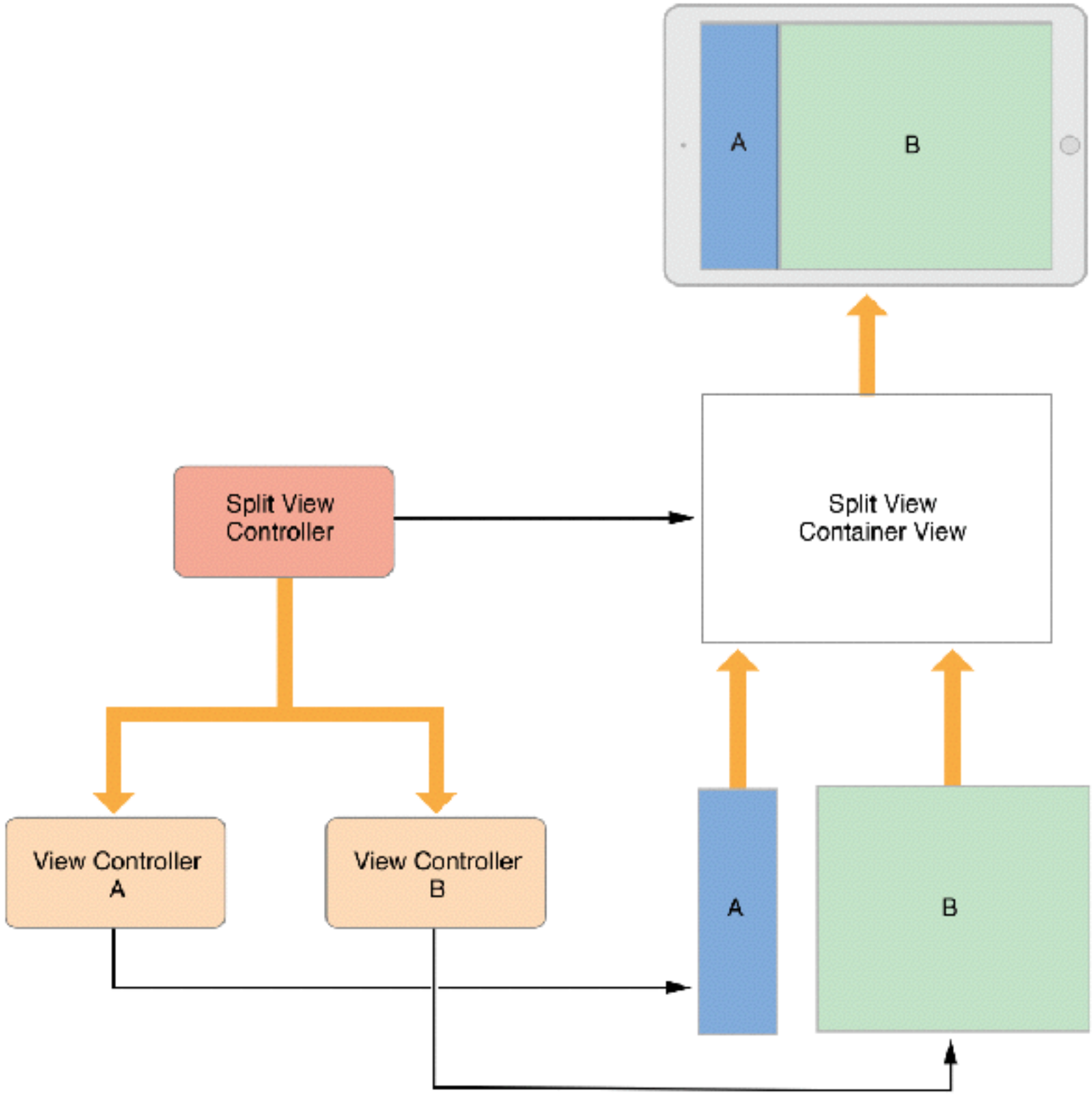

Edit
Alarm
10:28AM
Alarm


Stopwatch
00:00.00
Lap
Start


Timer
0 hours
0 min
1 sec
When Timer Ends
Radar
Cancel
Start

# Funkce Nav-VC

- Tzv. ”container vc”. Má:
  - rootViewController — základní/počáteční VC.
  - viewControllers : [UIViewController]
  - contentView — View pro vnitřní obsah.
- pushViewController(o:VC, animated:Bool)
- popViewController(animated:Bool)
- Obdobně s TabBarVC.


A
B
Split View
Controller
Split View
Container View
View Controller
A
View Controller
B
A
B

# Master-Detail styl

- Je složením dvou VC:
  - Master — typicky přehledový VC, tabulka se záznamy.
  - Detail — typicky záběr na detail záznamu z Master.
- Je směřován na tablety, ovšem lze jej konfigurovat i na iPhone.
  - Není SplitView, ale N-VC s obsahem Master, pak push na Detail.

# M-D v režimu Landscape

# M-D v režimu Portrait


Plays
Julius Caesar
King Lear
Othello
Henry IV, Pt 1
The Tempest
Title
Plays
Julius Caesar
King Lear
Othello
Henry IV, Pt 1
The Tempest

# VC referencuje container VCs

- Vazba na NVC, TabBarVC, SplitVC je v aplikacích tak typická, že UIViewController automaticky v hierarchii VC najde/referencuje:
  - Navigation VC,
  - Tab Bar VC,
  - Split VC.

# Přechod na druhý VC

- Deaktivace původního vc : UIViewController:
  - vc.willMove(toParentViewController: nil)
  - vc.view.removeFromSuperview()
  - vc.removeFromParentViewController() — ??

# Přechod na druhý VC

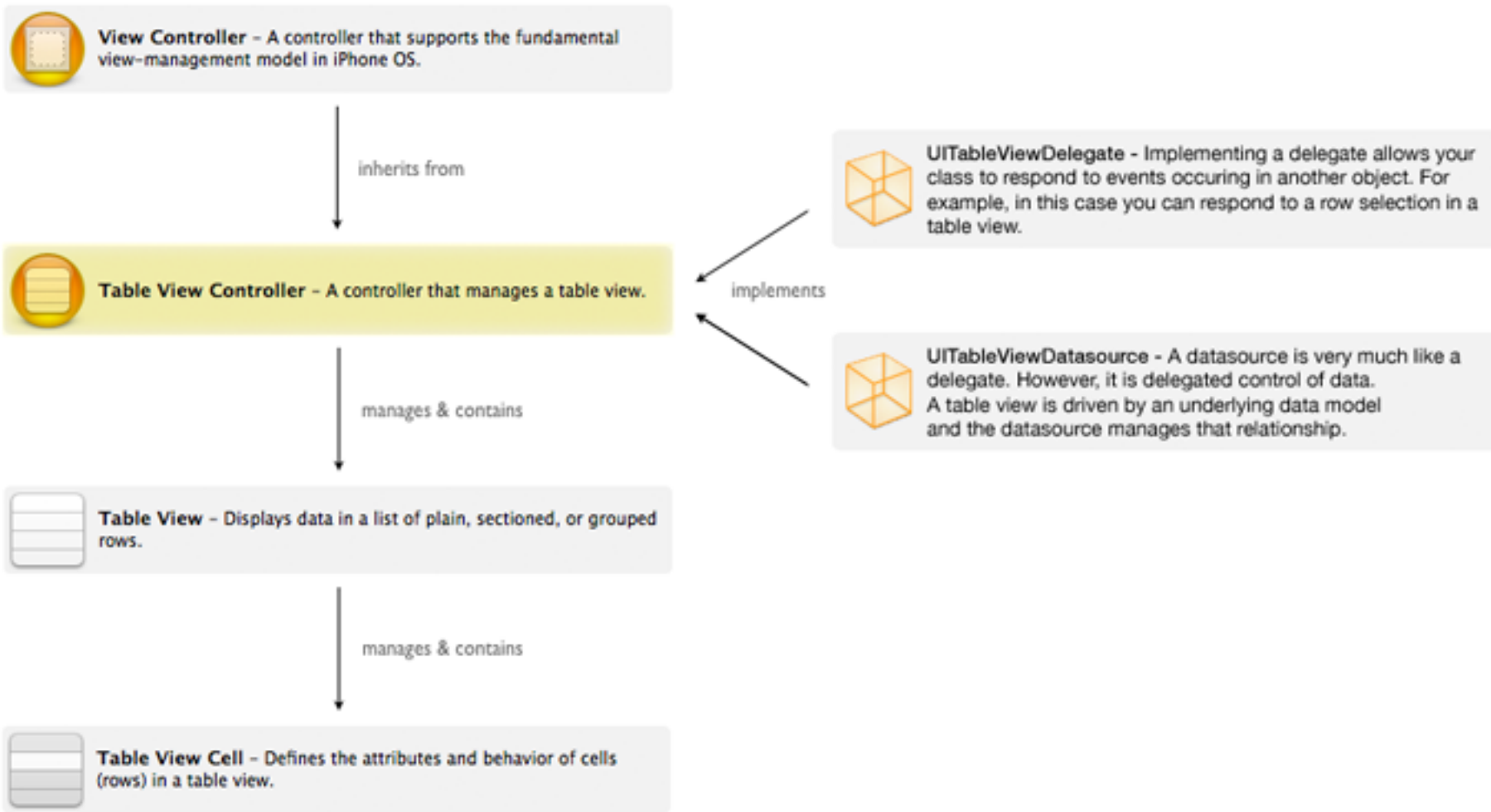
- Aktivace nového VC, nvc:
  - addChildViewController(nvc) — ??
  - view.addSubView(nvc.view)
  - nvc.view.frame = view.bounds
  - nvc.view.autoresizingMask = [.flexibleWidth, .flexibleHeight]
  - nvc.didMove(toParentViewController: self)

# Table View Controller

- Nejdůležitejší prvek UI iOS.
- Seznam buněk (TableViewCell).


UITableViewCellStyleDefault UITableViewCellStyleSubtitle UITableViewCellStyleValue1 UITableViewCellStyleValue2

# UITableViewController

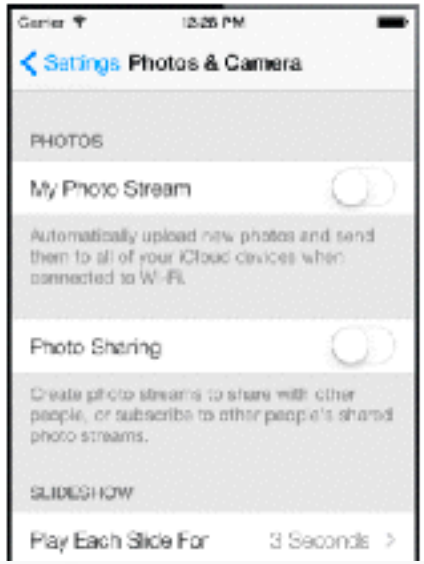


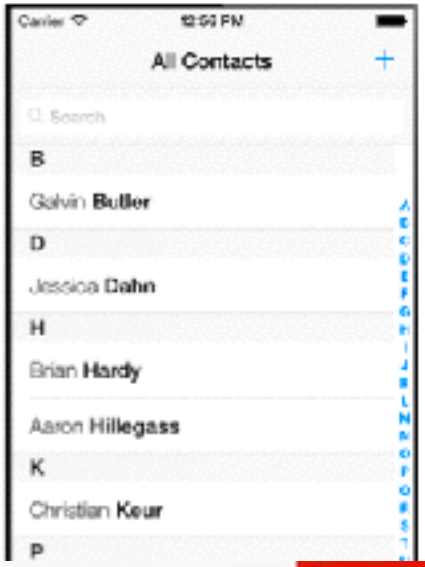
# UITableViewController

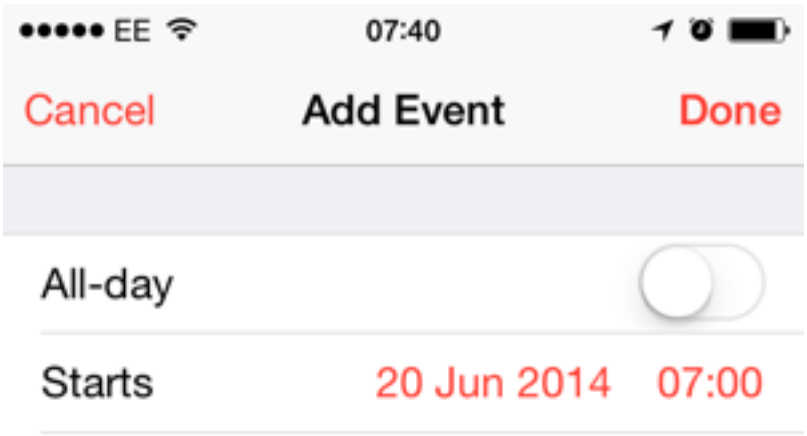
- Implementuje dataSource a delegate protokoly tabulky.
- Uživatelský VC dědí z TVC, přepisuje potřebné metody dataSource a delegate protokolů.

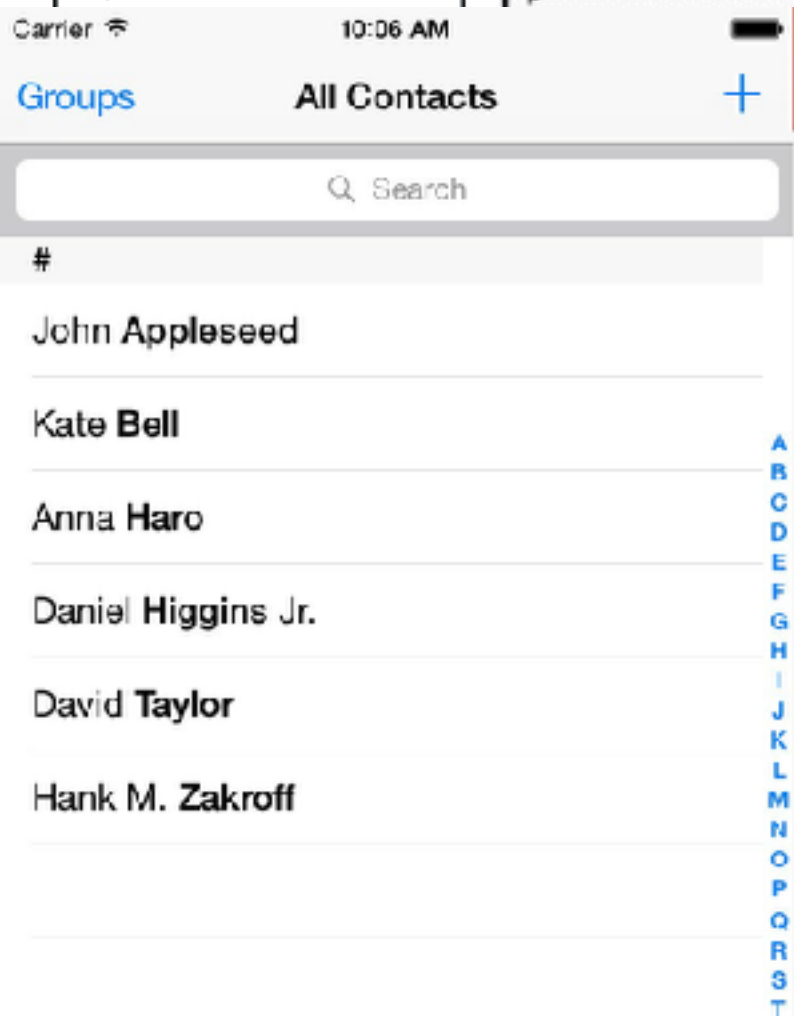
# UITableView

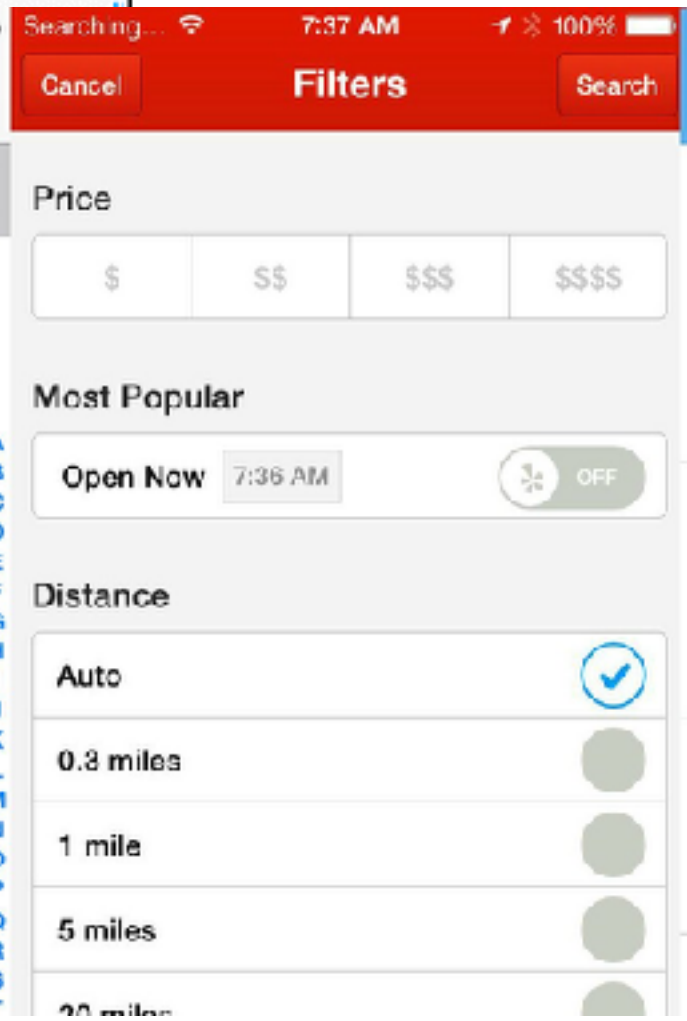
- Odvozen od UIScrollView (přesahuje rámec obrazovky).
- Provádí rozmístění (layout) buněk (Cell) ve skupinách.
- Obecný heterogenní dynamický soupis buněk vertikálně řazených.


Settings Photos & Camera
PHOTOS
My Photo Stream
Photo Sharing
SLIDESHOW
Play Each Slide For
3 Seconds


All Contacts
B
Galvin Butler
D
Jessica Dahn
H
Brian Hardy
Aaron Hillegass
K
Christian Keur
P


EE
07:40
Cancel
Add Event
Done
All-day
Starts
20 Jun 2014
07:00

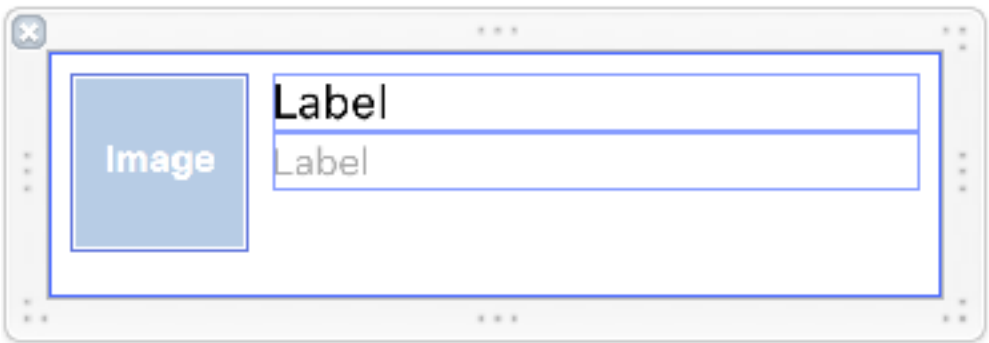

Carrier
10:06 AM
Groups
All Contacts
Search
#
John Appleseed
Kate Bell
Anna Haro
Daniel Higgins Jr.
David Taylor
Hank M. Zakroff
A B C D E F G H I J K L M N O P Q R S T

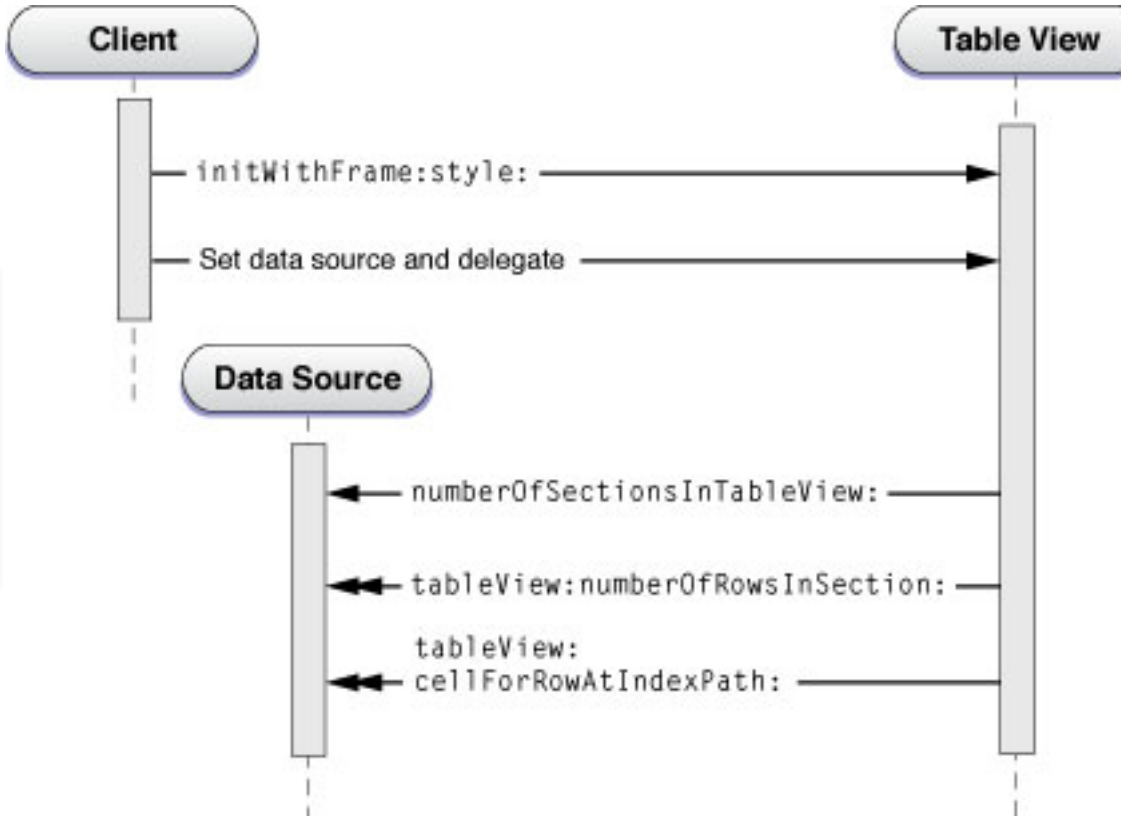

Searching...
7:37 AM
100%
Cancel
Filters
Search
Price
$
$$
$$$
$$$$
Most Popular
Open Now
7:36 AM
OFF
Distance
Auto
0.3 miles
1 mile
5 miles


Searching...
7:36 AM
100%
Home
Wall Street Journal retweeted
Gabby Stern @gabbystern 1h
Why Brazilians & expats fill their suitcases when in the US: Tips for coping w/ Brazil's crazy prices on.wsj.com/1J36C6o via @WSJexpat
27
16
Cara Delevingne @Caradelevin... 16m
I miss you Gaga aka Angela Delevingne 1912-2014
instagram.com/p/yFJ7HcjKCz/
191
485
KATY PERRY @katyperry 17m
Hey @JJWatt what do you think of our @ESPNMag cover? TOUCHDOWN?!
Timelines
Notifications
Messages
Me

# Řízení — dataSource

- TVC zprostředkovává obsah (Model) do View.
- Model zde vystupuje jako ”dataSource” API:
  - počet sekcí, počet řádků v sekci.
  - sestavení TableViewCell pro zadanou souřadnici (IndexPath).

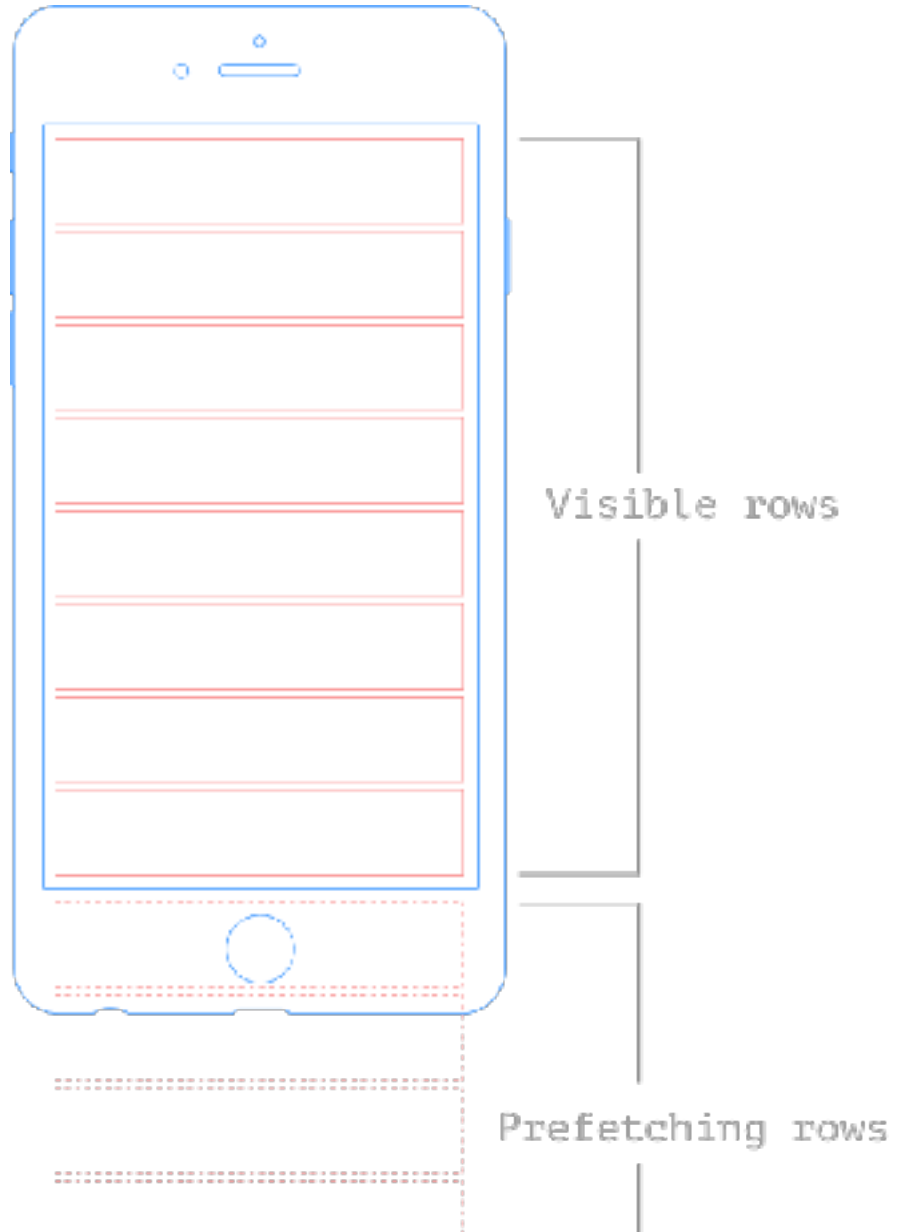


# Řízení TV, dynamika

- TV zjistí datový rozsah (sekce, řádky).
- Předpokládá se, že viditelná je pouze část buněk.
  - Ty jsou alokovány a inicializovány.
  - Při posuvu tabulkou se dynamicky volá dataSource na doplňování obsahu buněk.
  - Znovupoužití objektů buněk (pool). Identifikátory buněk.

# Dynamika přístupu na dataSource



# Demo

```
class SimpleTab: UITableViewController {

    var seznam = ["Jeden", "Druhy", "Treti"];

    override func tableView(_ tableView: UITableView,
                numberOfRowsInSection section: Int) -> Int
    {
        return seznam.count
    }

    override func tableView(_ tableView: UITableView,
                cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell",
for: indexPath)

        cell.textLabel?.text = seznam[indexPath.row]

        return cell
    }
}
```

# Demo, oddělený dataSource

```
class MyArrayModel: NSObject, UITableViewDataSource {
    var seznam : [String]
    init(withStrings : [String]) {
        self.seznam = withStrings;
        super.init();
    }
    func tableView(_ tableView: UITableView,
                   numberOfRowsInSection section: Int) -> Int
    {
        return seznam.count
    }
    func tableView(_ tableView: UITableView,
                   cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell",
for: indexPath)

        cell.textLabel?.text = seznam[indexPath.row]

        return cell
    }
}
```

# Demo, oddělený dataSource

```
class ModelTab: UITableViewController {
    override func viewDidLoad() {
        super.viewDidLoad();
        //
        self.tableView.dataSource = MyArrayModel(withStrings:
["1","2","3"]);
    }
}
```

# Dynamika, posuv v Table

- Chod aplikace musí být hladký.
- Inicializace buněk může brzdit chod tabulky (rychlý posuv).
- Při náročnější inicializaci se přechází do bočních vláken (GCD).

```
override func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell
    {
        let cell = tableView.dequeueReusableCell(withIdentifier:
"tabik", for: indexPath)

        cell?.imageView?.image = placeHolder;

        DispatchQueue.global().async {
            //
            if let loaded = myModel.load(cosi) {
                //
                if let ncell = tableView.cellForRow(at: indexPath) {
                    //
                    DispatchQueue.main.async {
                        ncell.imageView?.image = loaded;
                    }
                }
            }
        }

        return cell;
    }
```

# TableView, Delegate

- Dostává zprávy o událostech nad tabulkou.
  - Označení buňky — will/did. Zrušení značení — will/did.
  - Typicky se provede přesun do dalšího VC (detail obsahu buňky).
- Geometrie buněk, dodatečné texty, …
- Rozsáhlý protokol — UITableViewController ho implementuje.

# Navigation VC, push VC

```
override func tableView(_ tableView: UITableView,
                didSelectRowAt indexPath: IndexPath)
    {
        //
        let story = UIStoryboard(name: "Main",
                                 bundle: Bundle.main);

        let det = story.instantiateViewController(withIdentifier:
"jeden")

        self.navigationController?.pushViewController(det, animated:
true);
    }
```

# Závěr

- Příště podrobněji o Views a ViewControllers.
- Seminář. Program?