МГТУ
им. Н.Э. Баумана
ТЕХНОПАРК
Mail.Ru Group

Layout

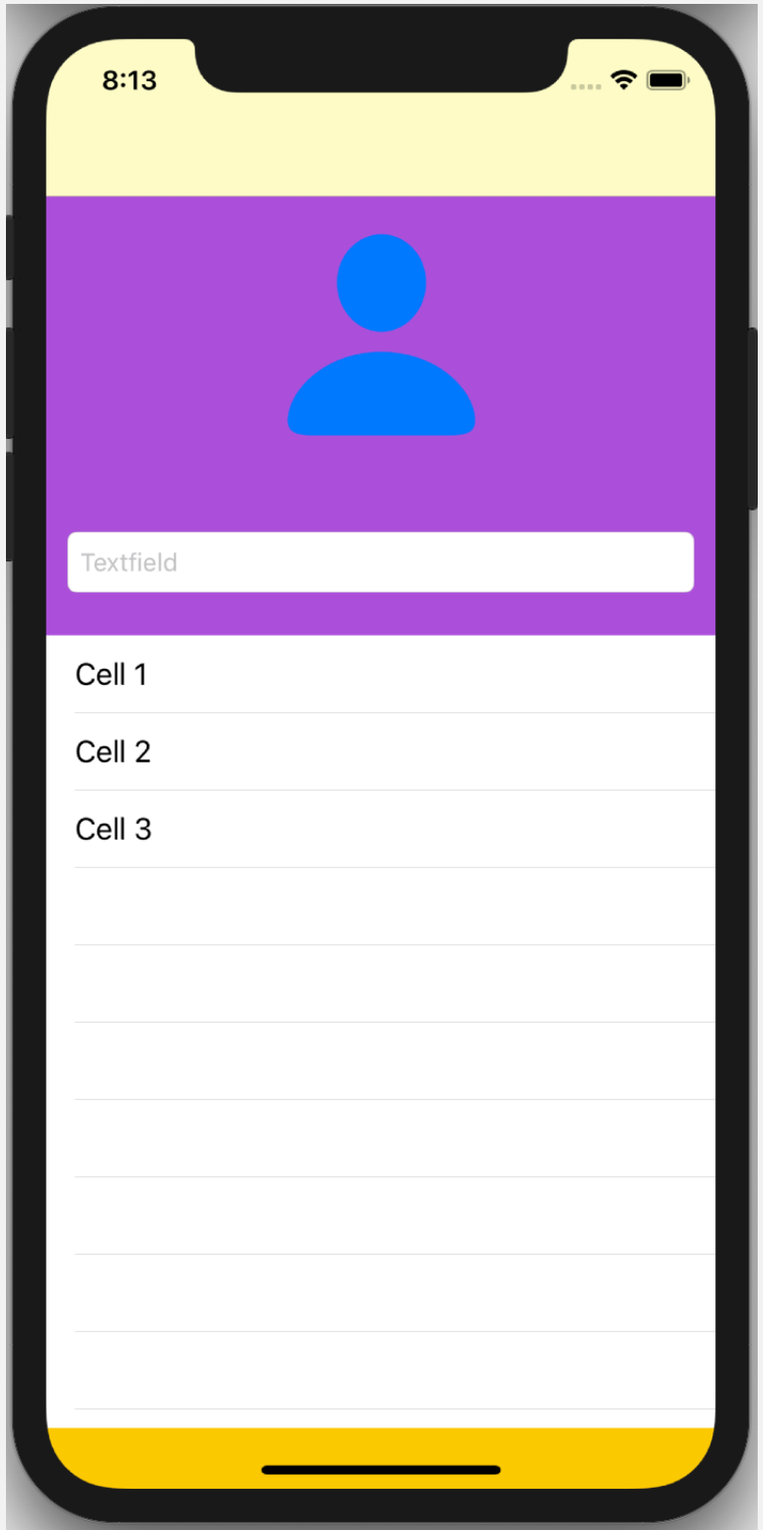
Лекция №3

Павел Тихонов, Артур Сардарян,
Павел Носов, Константин Кулаков, Андрей Бабков

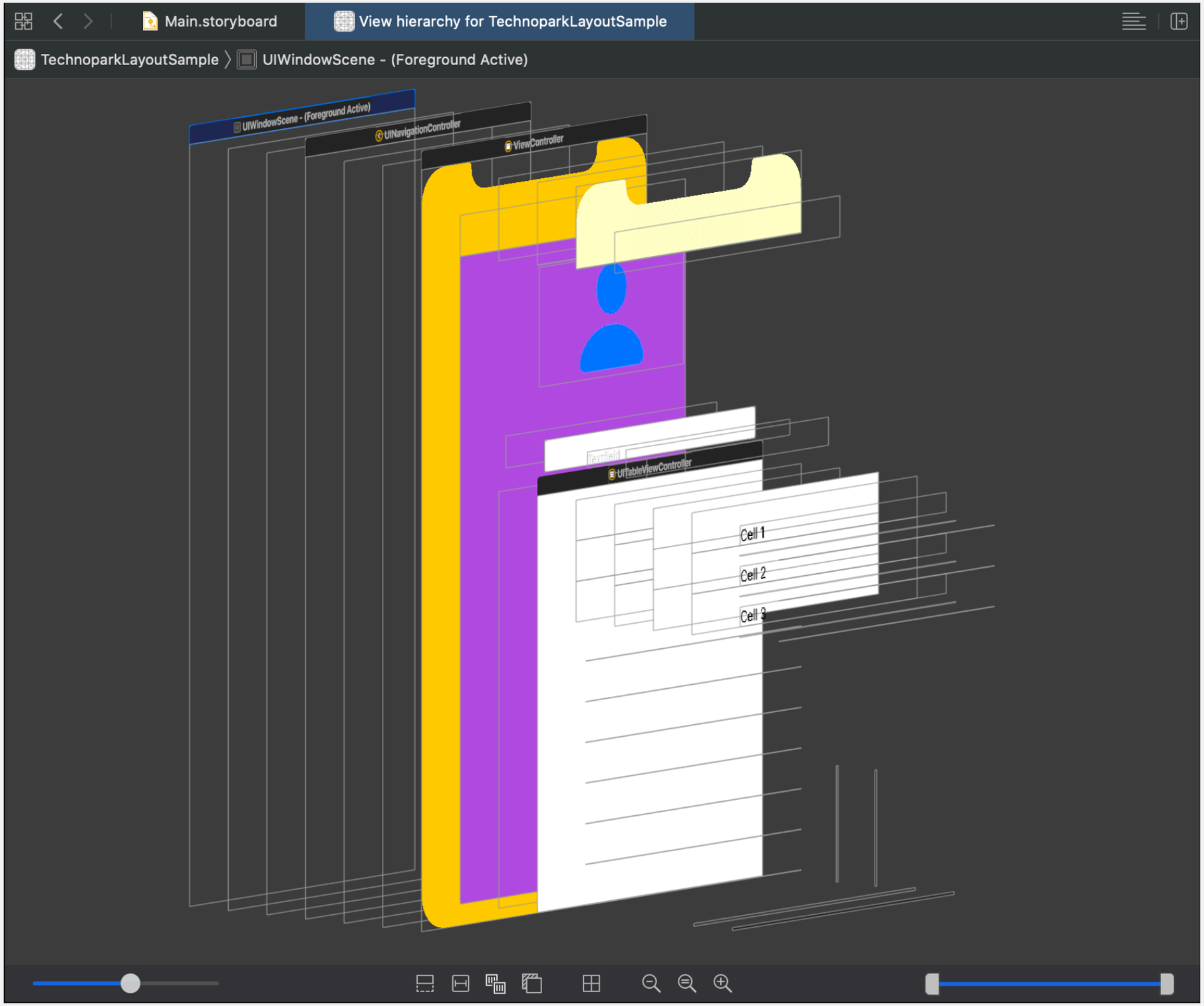
Организационная часть

- Не нарушая традиций — отметиться
- Чем займемся:
  - Рассмотрим базовые элементы UIKit
  - Поговорим о подходах для создания интерфейса
  - Узнаем про responder chain
  - Распознавание жестов
- Оставить отзыв (после занятия)

Интерфейс приложения



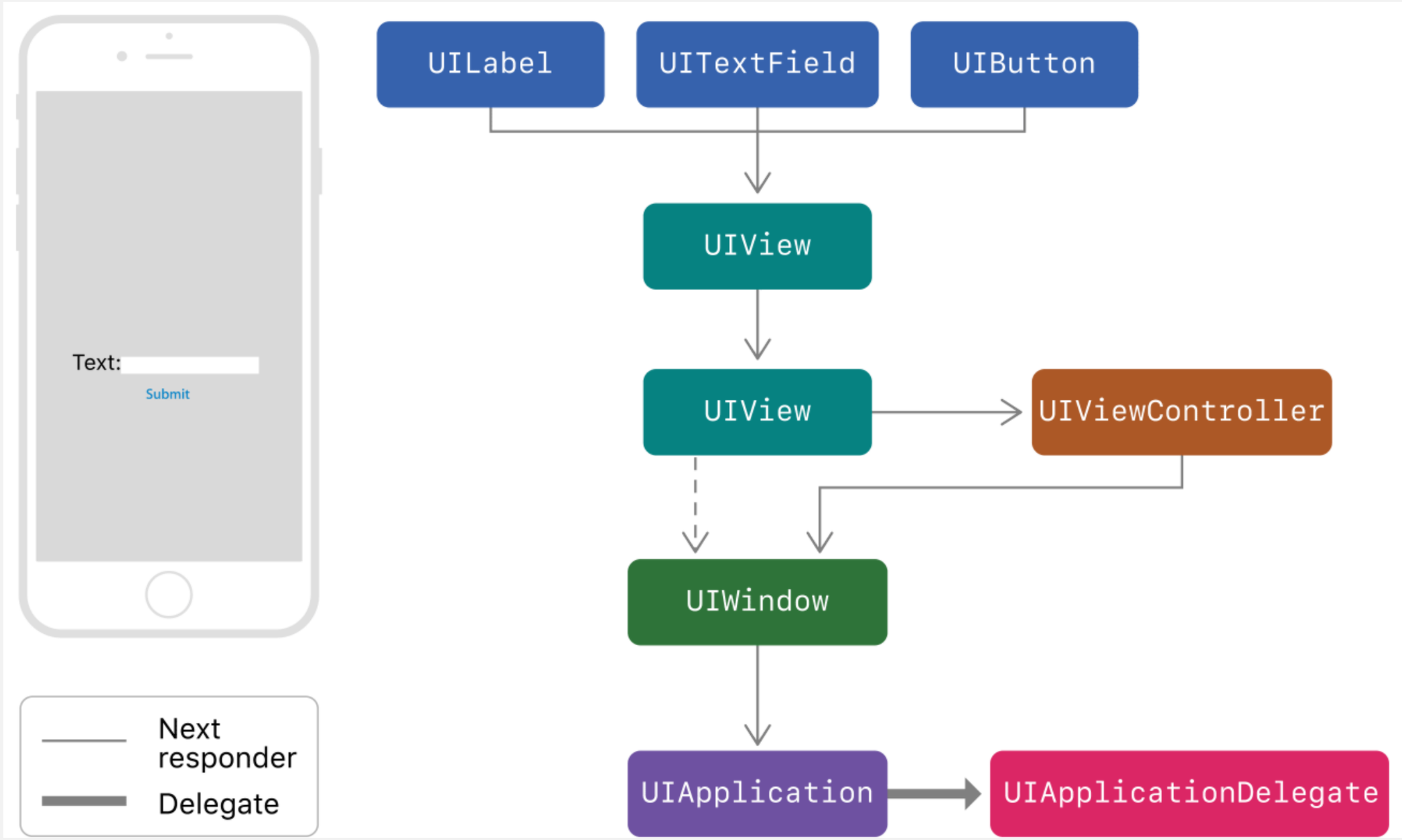
Интерфейс приложения в разрезе



UIView

- Базовый класс UIKit («кирпичи» из которых строится интерфейс)
- Потомок UIResponder’a
  - обрабатывает события:
    - касания (touches)
    - движения (motions), например, встряхивание
- Отображает элементы интерфейса
- Может настраивать размеры и позиции своих subview
- Могут добавлять распознавателей жестов (gesture recognizers) для обработки типичных жестов

Responder Chain



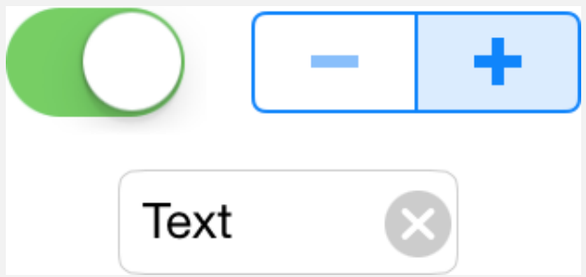
https://developer.apple.com/documentation/uikit/understanding_event_handling_responders_and_the_responder_chain

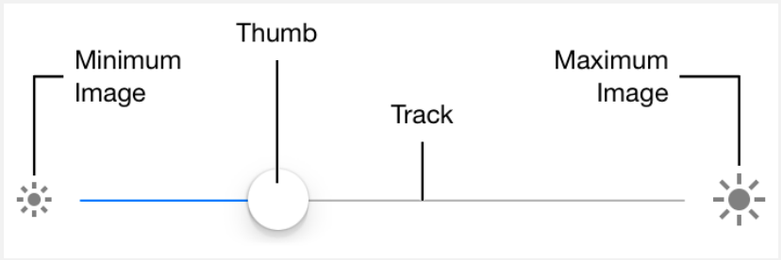
UIView (потомки)

- UIControl
  - UIButton, UITextField и др.
- UIWindow
  - редко нужен в iOS, но может пригодиться, если нужно что-то показать поверх status bar’a
- UILabel
- списки
  - UIScrollView,
  - UITableView,
  - UICollectionView,
  - UITextView
- UIImageView
- MKMapView
- WKWebView

UIControl

- UIControl
  - UIButton
  - UITextField
  - UISwitch
  - UISegmentedControl
  - UISlider
  - UIProgress
- Добавляет к UIView
  - механизм target/action
  - протокол (UITextFieldDelegate и пр.)





https://developer.apple.com/reference/uikit/uicontrol
https://developer.apple.com/reference/uikit/uislider

UIView (основные понятия)

- один superview
- много subview
- addSubview: (у родительской view)
  - добавляет к нашему view subview
- removeFromSuperview
  - удаляет наш view из его superview

- анимации
  - ```
    @available(iOS 4.0, *)
    open class func animate(withDuration duration: TimeInterval, delay: TimeInterval,
        options: UIView.AnimationOptions = [], animations: @escaping () -> Void,
        completion: ((Bool) -> Void)? = nil)
    ```
    ```

UIView (полезные свойства)

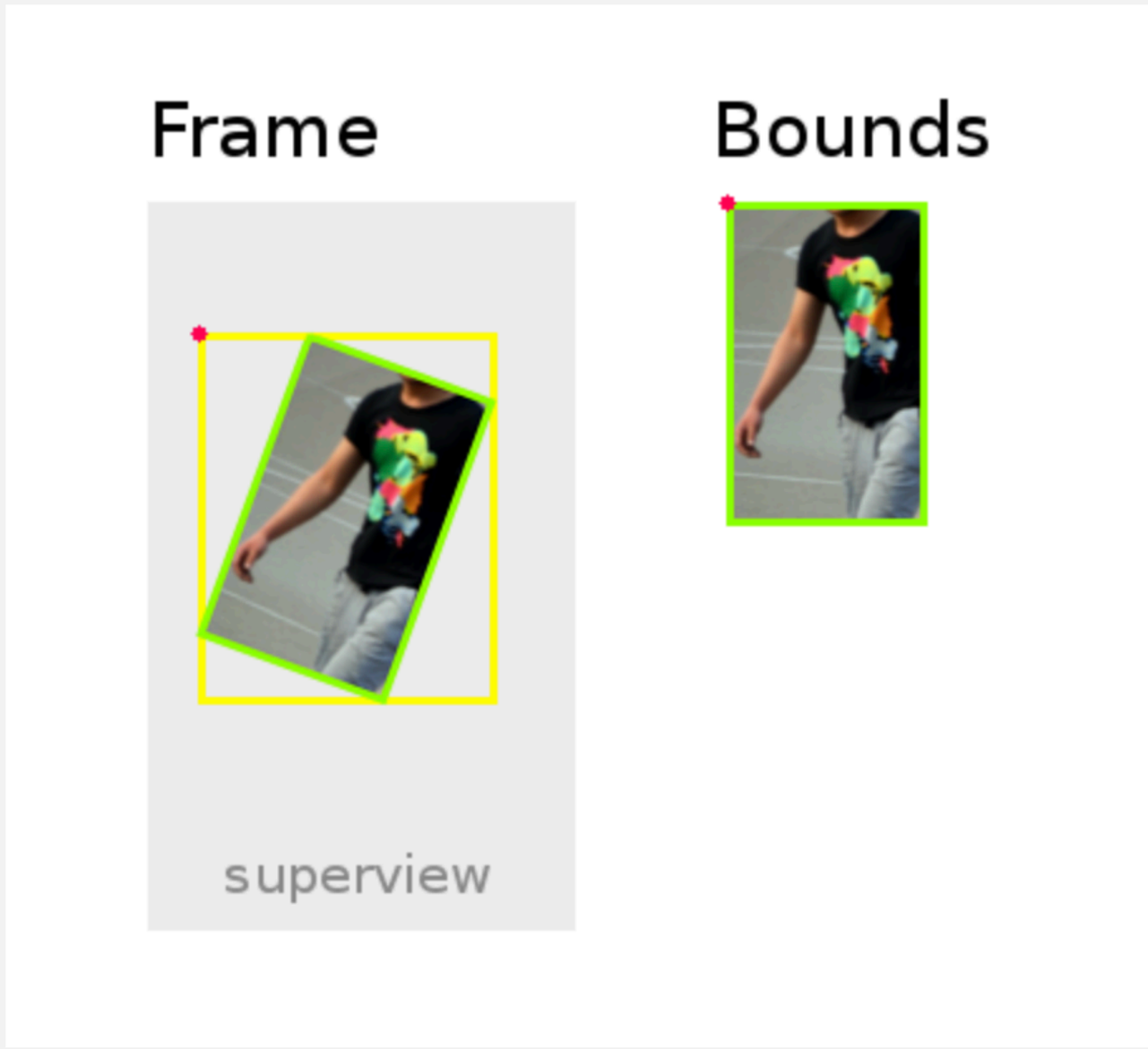
- CGFloat alpha
  - прозрачность (от 0 до 1)
- Bool isOpaque (непрозрачный)
  - true/false
- Bool isHidden (невидимый)
- Bool userInteractionEnabled (отключенный)
- Bool clipsToBounds (обрезать по границам)
- Bool translatesAutoresizingMaskIntoConstraints
  (при использовании auto layout кодом)

UIView (bounds / frame)

- bounds — положение и размер в собственной системе координат
- frame — положение и размер в системе координат родительской вью

- Если нужно спозиционировать view, лучше менять center, а frame пусть считается сам

UIView (bounds / frame)



Распознавание жестов

- Вручную
  - touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)touchesEnded:withEvent:
  - touchesMoved…
  - touchesCancelled…

- Использование UIGestureRecognizer (точнее, его потомков)
  - кодом
  - с помощью Interface Builder

Пример распознавания жеста:


```
.....

let gestureRecognizer =

  UIGestureRecognizer(target: self, action: #selector(viewTapped(_:)))

view.addGestureRecognizer(gestureRecognizer)

.....


@objc func viewTapped(_ sender: UIGestureRecognizer) {
    print("viewTapped: \(gestureRecognizer.view)")
}
```

UIGestureRecognizer — основные виды

UIGestureRecognizer — основные виды

- tap
- swipe
- pan
- long press
- pinch
- rotation

Кастомные UIView

- Кастомизация
  - init(frame:) (могут отличаться у наследников UIView, например, у UITableViewCell)
    - конструктор используется только для создания view кодом
  - init(coder:)
    - для view, созданного через Interface Builder
  - draw(_:)
    - если нужно во view что-то нарисовать

Кастомные UIView: -drawRect:

- В UIView можно рисовать:
  - **UIBezierPath**

```
let path = UIBezierPath()
```

    - moveToPoint:,
    - addLineToPoint
  - **Core Graphics**
    - контекст CGGetCurrentContext()
    - функции для рисования

- Используется CPU, а не GPU

Пример drawRect:


```swift
override func draw(_ rect: CGRect) {
    // получаем ссылку на контекст
    guard let context = UIGraphicsGetCurrentContext() else {
        return
    }

    // очищаем контекст
    context.clear(rect)

    // выставляем цвет
    context.setFillColor(UIColor.yellow.cgColor)

    // заполняем rect
    context.fill(rect)
}
```

```swift
override func draw(_ rect: CGRect) {
    let origin: CGPoint = .init(x: 50, y: 50)
    let size: CGSize = .init(width: 200, height: 400)
    let rect = CGRect(origin: origin, size: size)

    UIColor.green.setFill()
    let path = UIBezierPath(roundedRect: rect, cornerRadius: 10)
    path.fill()
}
```

```swift
override func draw(_ rect: CGRect) {
    let bezierPath = UIBezierPath()

    UIColor.green.setFill()

    bezierPath.move(to: CGPoint(x: 10, y: 10))
    bezierPath.addLine(to: CGPoint(x: 160, y: 10))
    bezierPath.addLine(to: CGPoint(x: 160, y: 160))
    bezierPath.addLine(to: CGPoint(x: 10, y: 160))
    bezierPath.close()

    bezierPath.fill()
}
```

Frame-based верстка

- До iOS 6 (2012) верстка делалась только через изменение frame у view
- С помощью autoresizingMask задавались правила изменения frame, если родительские размеры изменятся

Auto Layout

- Система линейных неравенств

- Constraints (ограничения)
  - основной класс NSLayoutConstraint

- Можно задать разными способами:
  - Interface Builder
  - Код (NSLayoutConstraint)
  - Код (visual format language)
  - Библиотеки (PureLayout, SnapKit)

Пример Auto Layout (VFL)


```swift
let subview = UIView(frame: .zero)
subview.backgroundColor = .purple
subview.translatesAutoresizingMaskIntoConstraints = false

view.addSubview(subview)

let views = ["subview": subview]
let metrics = ["bottomMargin": 50]
let horizontalConstraints: [NSLayoutConstraint] = NSLayoutConstraint.constraints(withVisualFormat: "H:|[subview]|",
                                                                                 options: [],
                                                                                 metrics: metrics,
                                                                                 views: views)

view.addConstraints(horizontalConstraints)

let verticalConstraints: [NSLayoutConstraint] = NSLayoutConstraint.constraints(withVisualFormat: "V:|[subview]-bottomMargin-|",
                                                                               options: [],
                                                                               metrics: metrics,
                                                                               views: views)

view.addConstraints(verticalConstraints)
```

Layout Anchors


```swift
// создаем constraints через NSLayoutConstraint-конструктор

NSLayoutConstraint(item: self.view!,
                   attribute: .top,
                   relatedBy: .equal,
                   toItem: subview,
                   attribute: .top,
                   multiplier: 1,
                   constant: 0).isActive = true

NSLayoutConstraint(item: self.view!,
                   attribute: .leading,
                   relatedBy: .equal,
                   toItem: subview,
                   attribute: .leading,
                   multiplier: 1,
                   constant: 0).isActive = true

// создаем те же constraints через Anchor-API

self.view.topAnchor.constraint(equalTo: subview.topAnchor).isActive = true
self.view.leadingAnchor.constraint(equalTo: subview.leadingAnchor).isActive = true
```

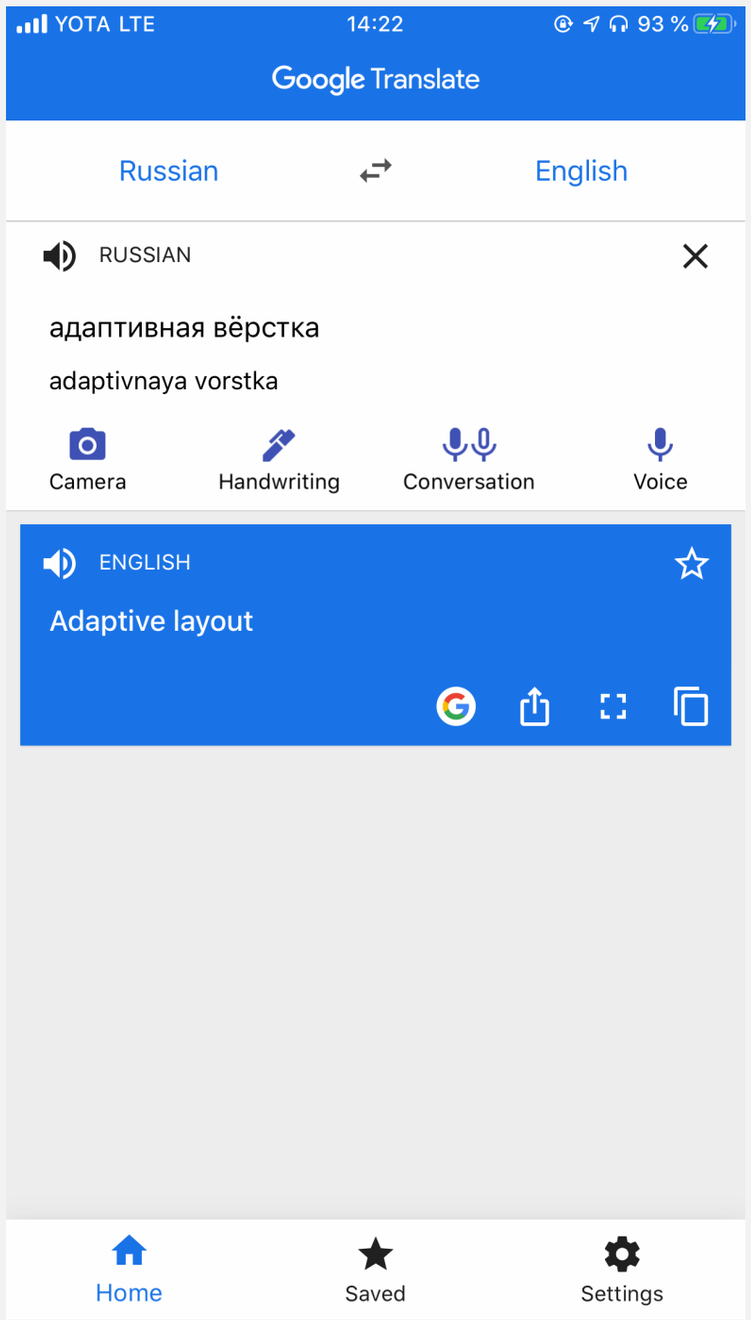
CALayer

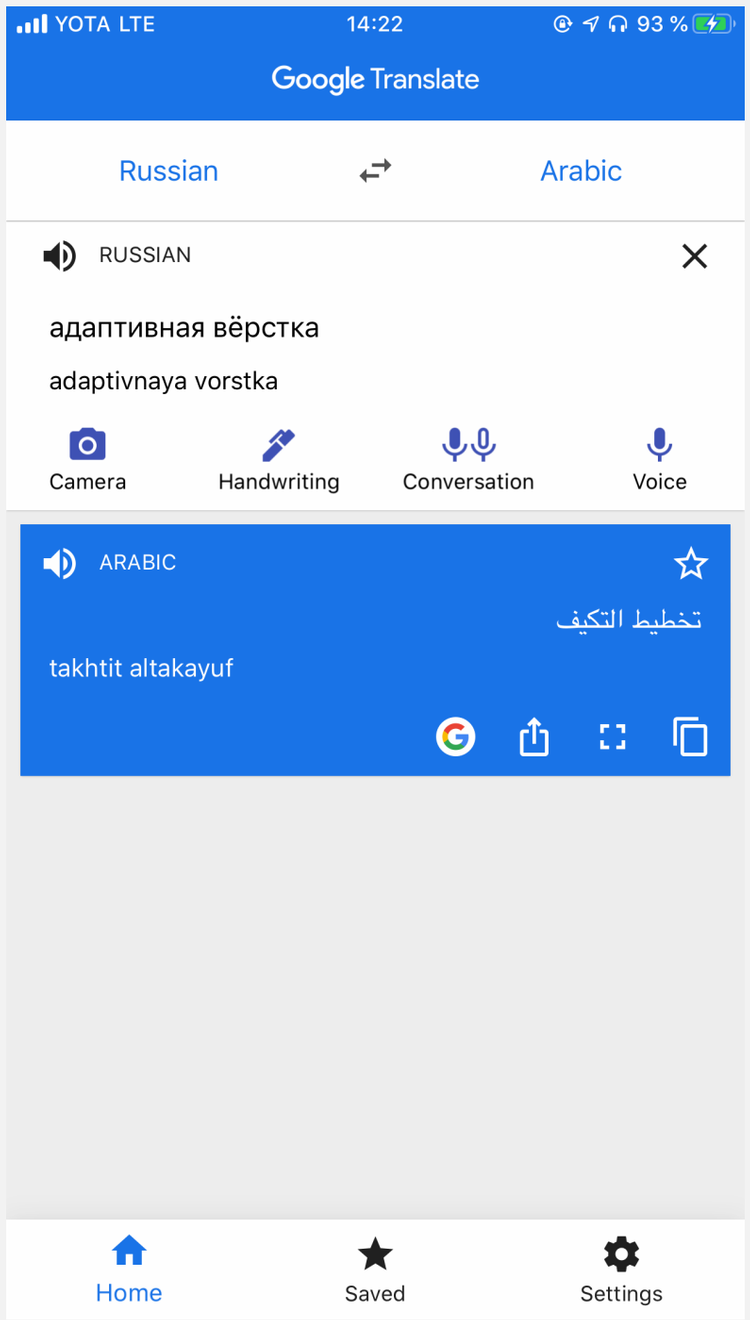
- Входит в состав CoreAnimation framework
- Создается UIView и UIView ставится его делегатом по дефолту
- Не умеет обрабатывать события (не наследуется от UIResponder)
- Нужен для большего контроля над отображением интерфейса и анимациями
- Пригодится:
  - cornerRadius
  - borderColor, borderWidth
  - shadowPath

Adaptivity

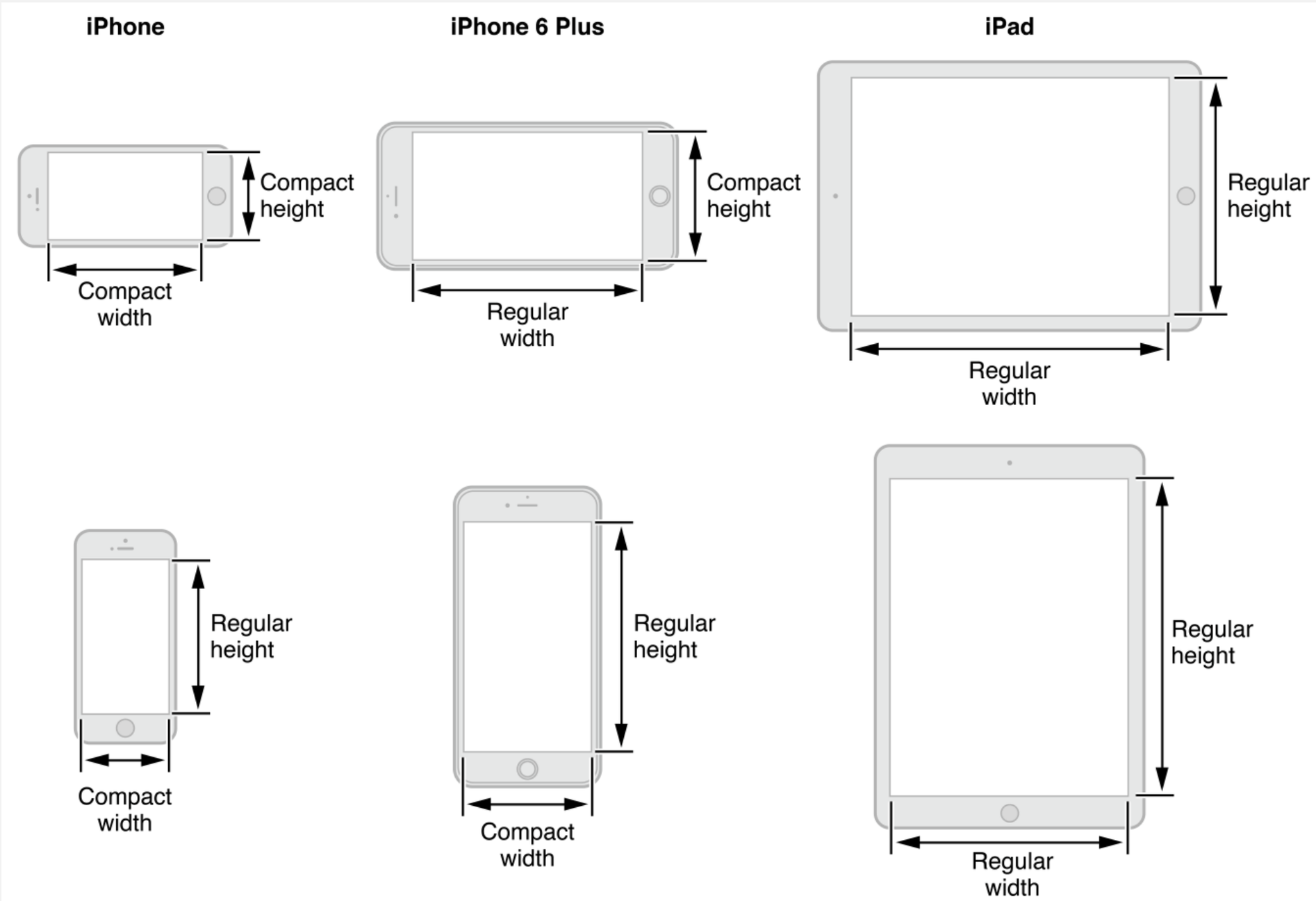
- Разные размеры экранов
- Разные ориентации (вертикальная, горизонтальная)
- Split View (iPad)
- Меняющийся текст
- Layout direction в зависимости от страны

Adaptivity


YOTA LTE
14:22
93 %
Google Translate
Russian
English
RUSSIAN
адаптивная вёрстка
adaptivnaya vorstka
Camera
Handwriting
Conversation
Voice
ENGLISH
Adaptive layout
Home
Saved
Settings


YOTA LTE
14:22
93 %
Google Translate
Russian
Arabic
RUSSIAN
адаптивная вёрстка
adaptivnaya vorstka
Camera
Handwriting
Conversation
Voice
ARABIC
تخطيط التكيف
takhtit altakayuf
Home
Saved
Settings

Size Classes



Size Classes

- Доступно в Storyboard
- В зависимости от size class можно:
  - добавлять/удалять constraint
  - добавлять/удалять view
  - менять атрибуты (например, шрифт)

- В коде

```
if view.traitCollection.horizontalSizeClass == .compact {
    // какая-то логика
}
```

Демо

- gesture recognizers
- autolayout
- draw rect

Ссылки

- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#//apple_ref/doc/uid/TP40010853 — Autolayout
- https://www.raywenderlich.com/443-auto-layout-tutorial-in-ios-11-getting-started - Autolayout
- https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/ — Adaptivity
- https://developer.apple.com/documentation/uikit/nslayoutanchor — Anchors
- https://developer.apple.com/ios/human-interface-guidelines/ — Apple Human Interface Guidelines
- https://www.objc.io/issues/3-views/custom-controls/ — о кастомных control'ах
- http://nshipster.com/ibinspectable-ibdesignable/ — IBDesignable / IBInspectable (редко используется, для фанатов Interface Builder'а)
- https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/VisualFormatLanguage.html — visual format language (для auto layout)
- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/Size-ClassSpecificLayout.html — Size Classes
- https://developer.apple.com/library/archive/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/BezierPaths/BezierPaths.html — рисование с помощью Bezier Paths