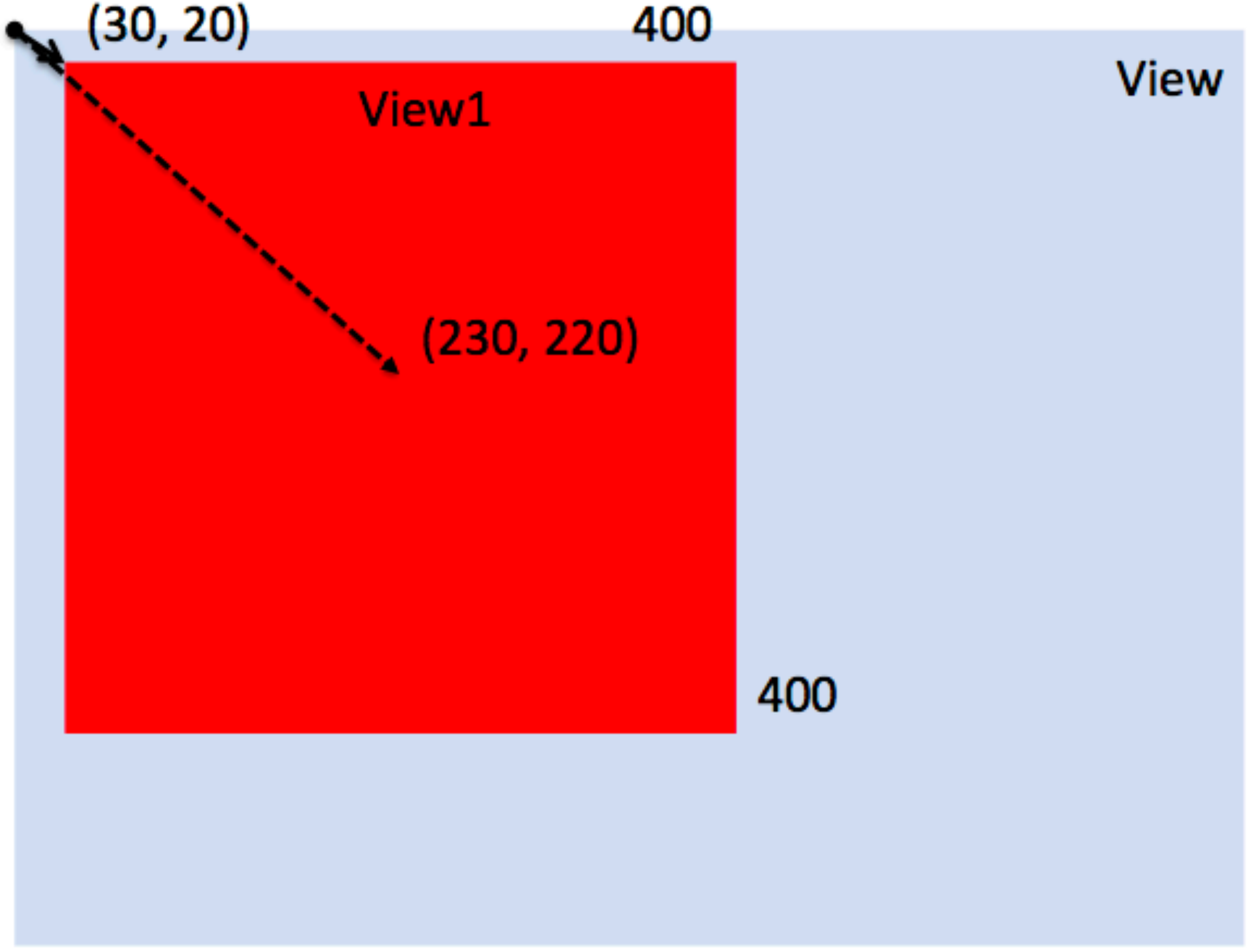
Autolayout

Frame & bounds



View1's Frame
(30, 20, 400, 400)

View1's Bounds
(0,0, 400, 400)

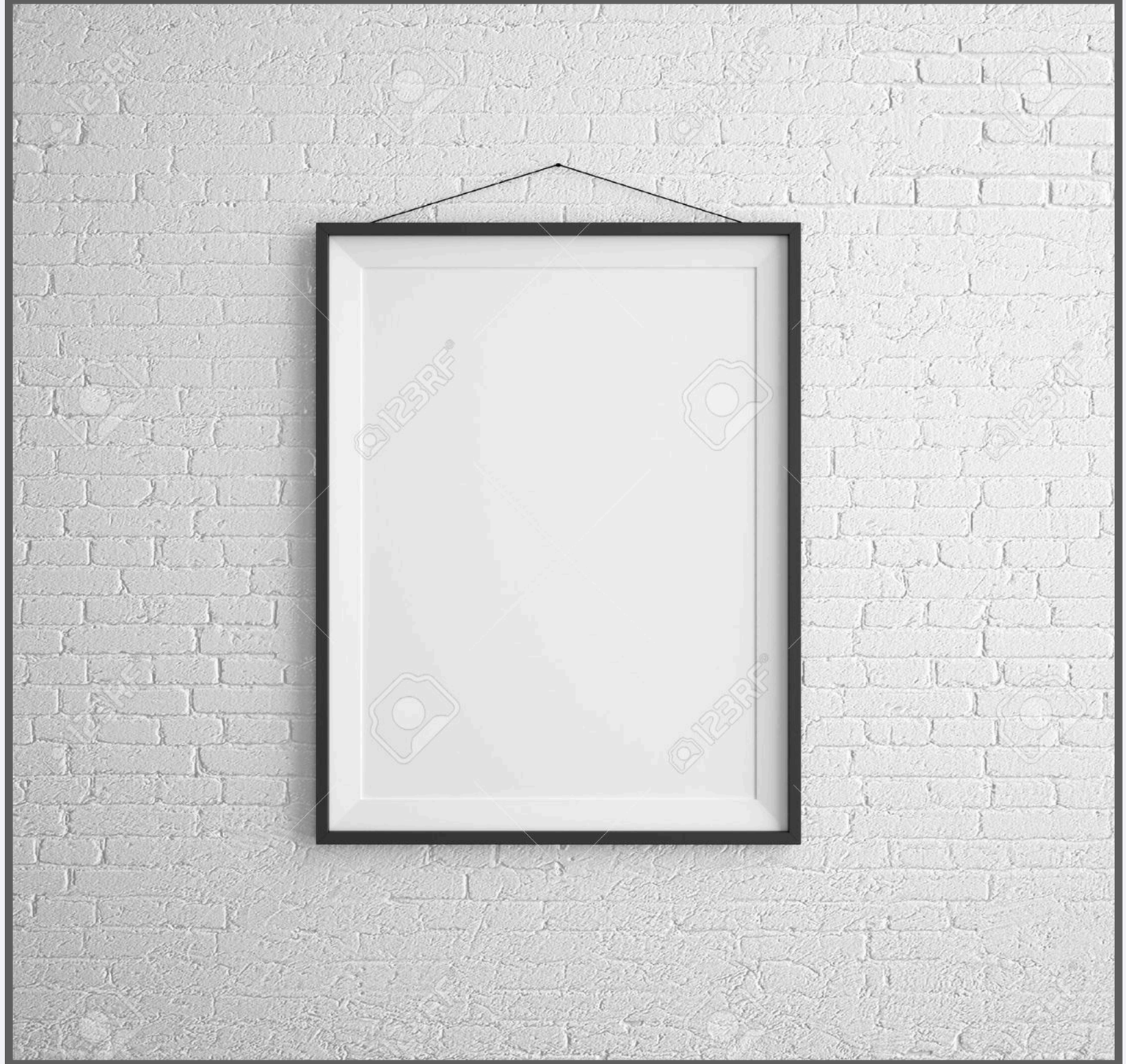
View1's Center
(230, 220)

Frame vs Bounds

frame = a view's location and size using the parent view's coordinate system

bounds = a view's location and size using its own coordinate system

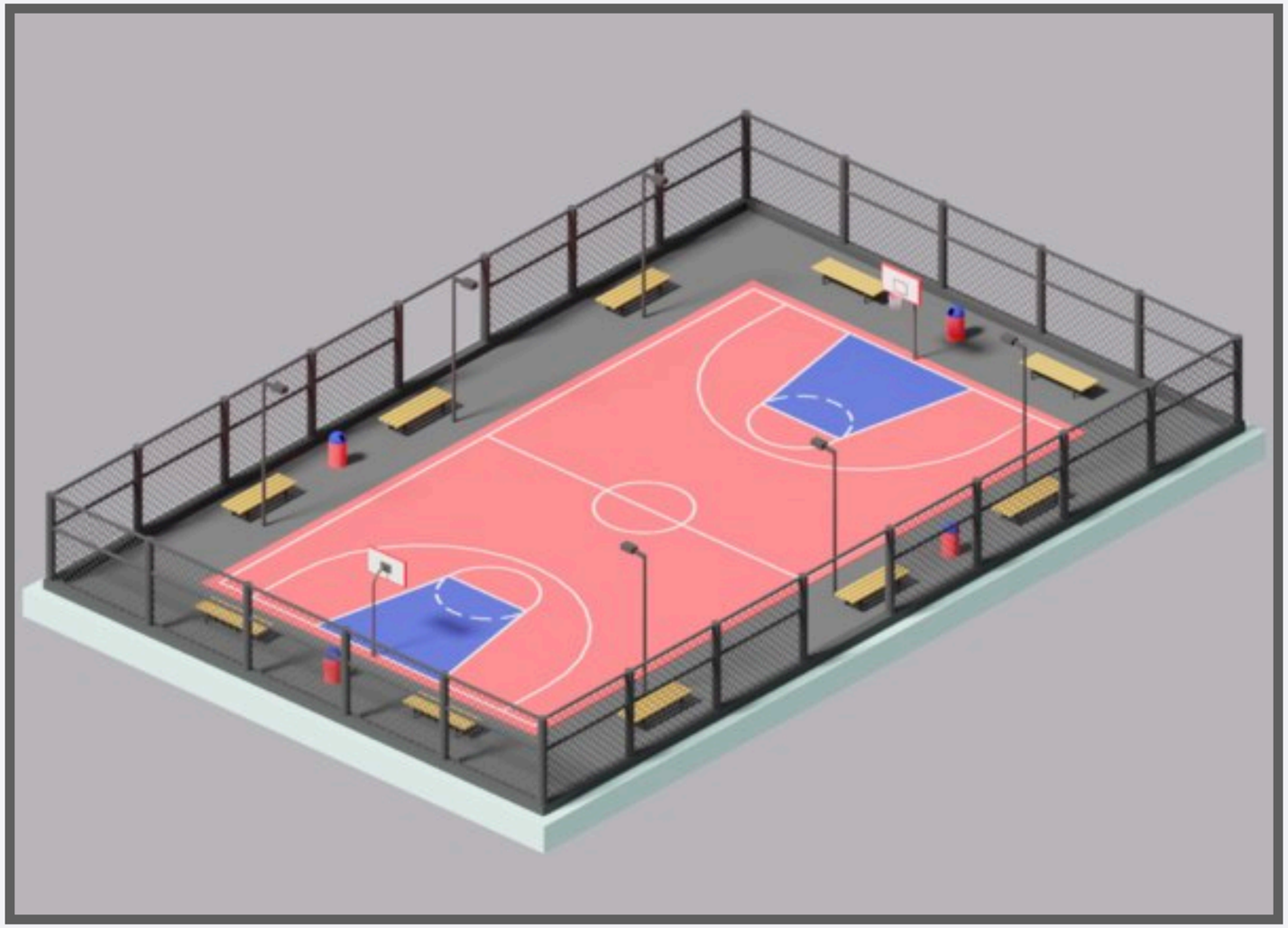
Frame



Сбербанк Онлайн

Devteam

Bounds

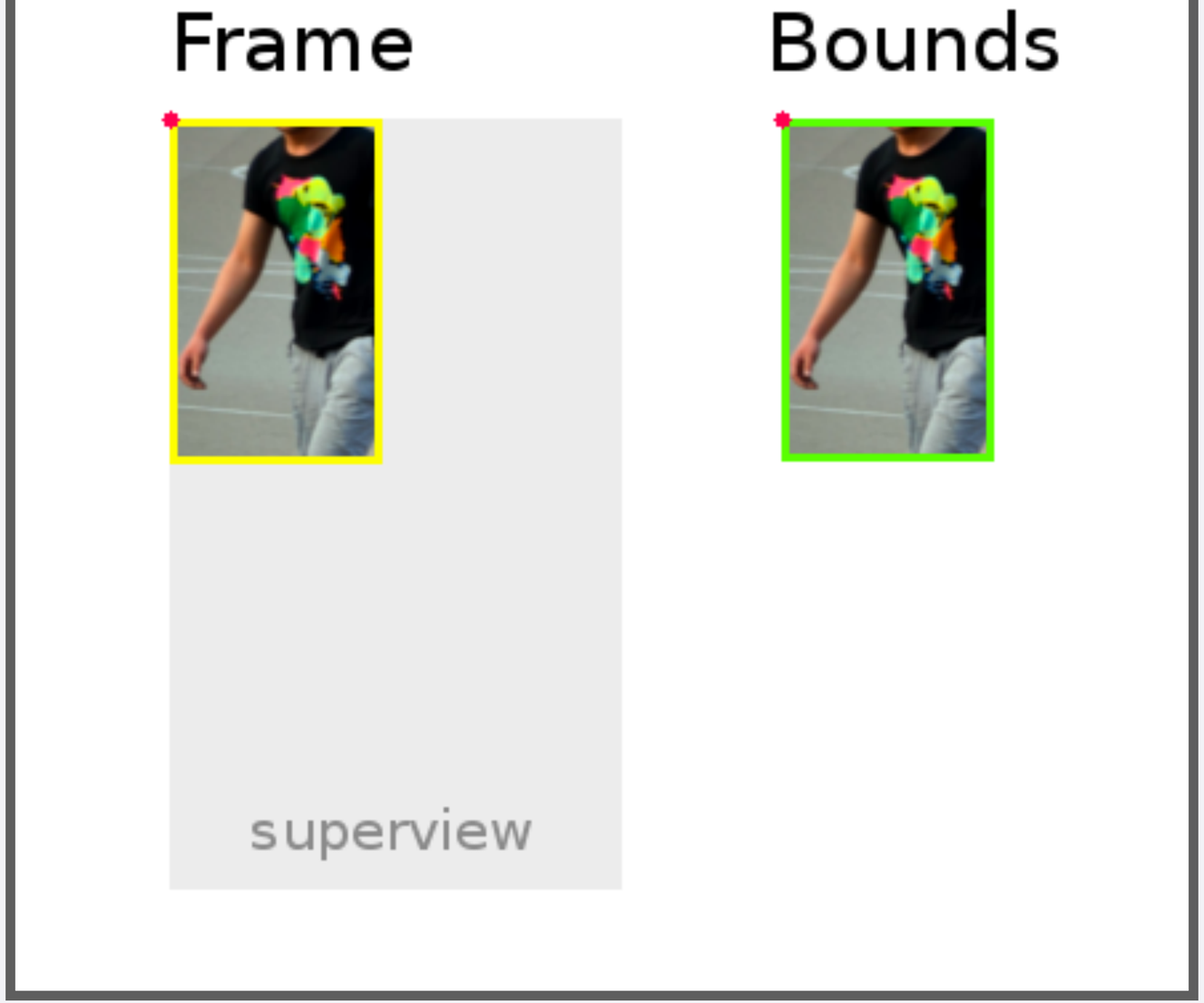


Сбербанк Онлайн

Devteam

Frame vs Bounds

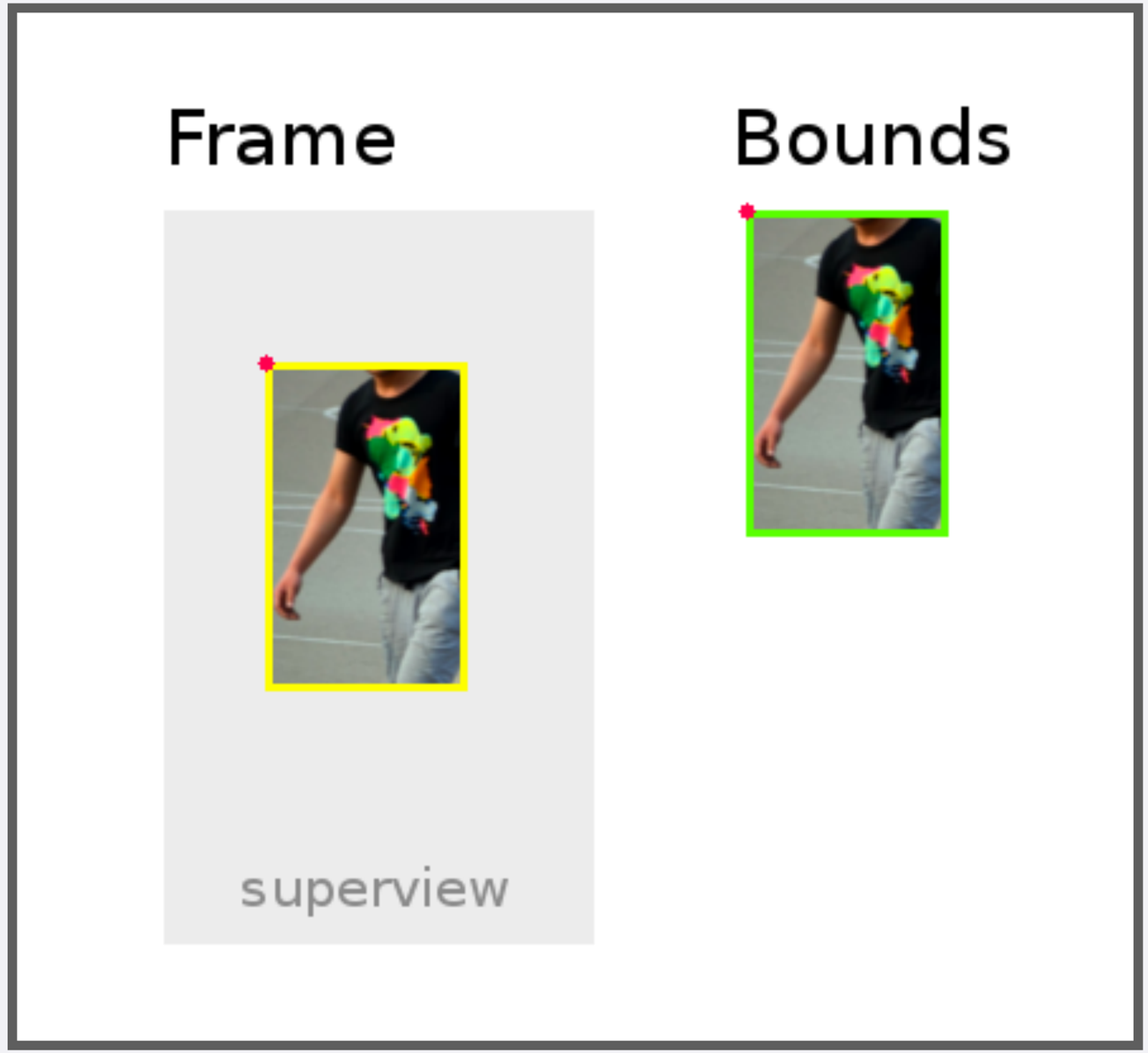
```
Frame
     origin = (0, 0)
     width = 80
     height = 130

Bounds
     origin = (0, 0)
     width = 80
     height = 130
```



Сбербанк Онлайн

Frame vs Bounds

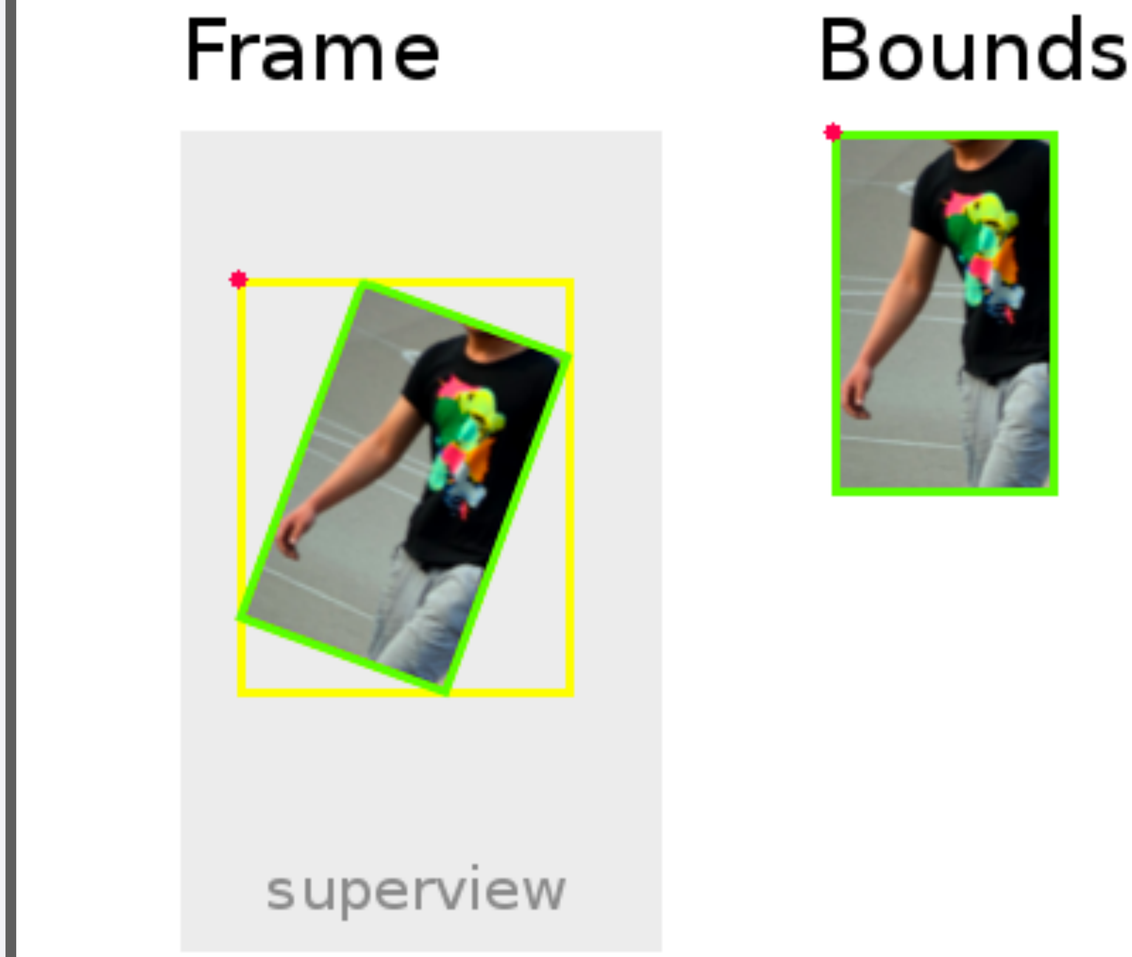
```
Frame
    origin = (40, 60)  // That is, x=40 and y=60
    width = 80
    height = 130

Bounds
    origin = (0, 0)
    width = 80
    height = 130
```



Сбербанк Онлайн

Frame vs Bounds

```
Frame
    origin = (20, 52)  // These are just rough estimates.
    width = 118
    height = 187

Bounds
    origin = (0, 0)
    width = 80
    height = 130
```



Сбербанк Онлайн

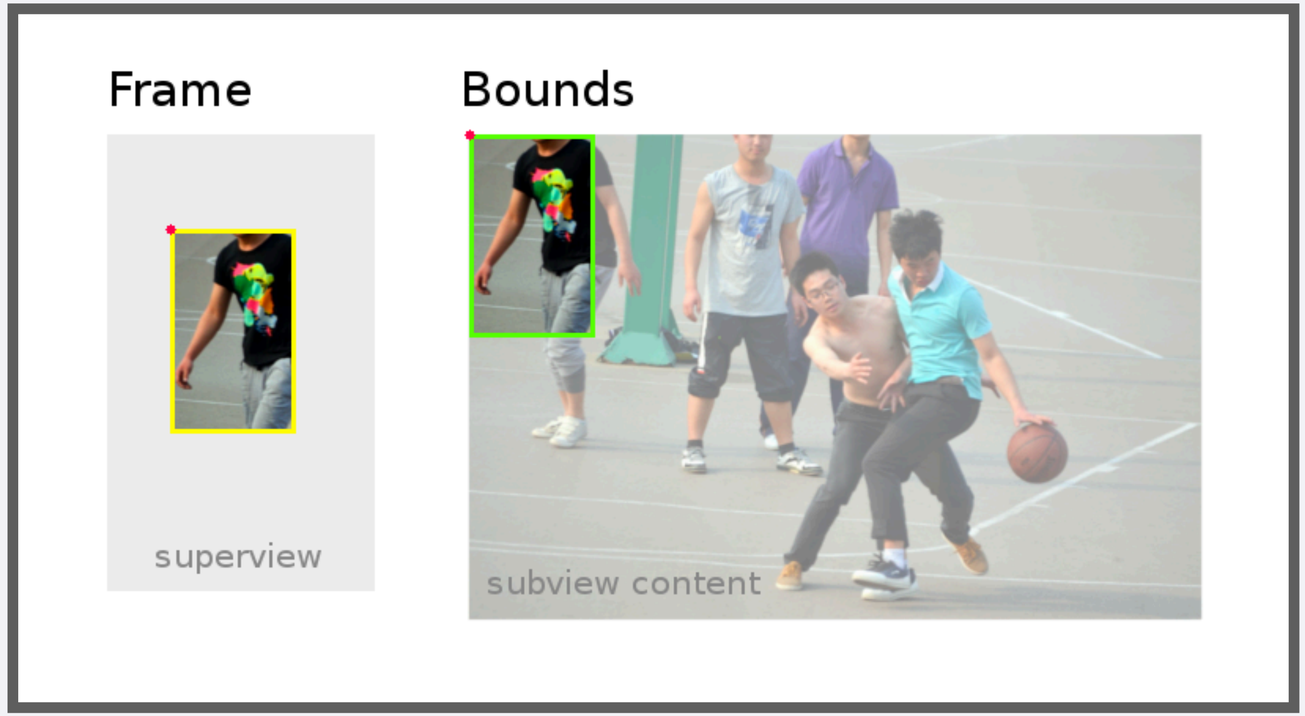
Frame vs Bounds

frame = the smallest bounding box of that view with respect to it’s parents coordinate system, including any transformations applied to that view

bounds = a view's location and size using its own coordinate system

Frame vs Bounds

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

Bounds
     origin = (0, 0)
     width = 80
     height = 130
```



Сбербанк Онлайн

Devteam

Frame vs Bounds

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

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

Frame vs Bounds

frame = outward changes

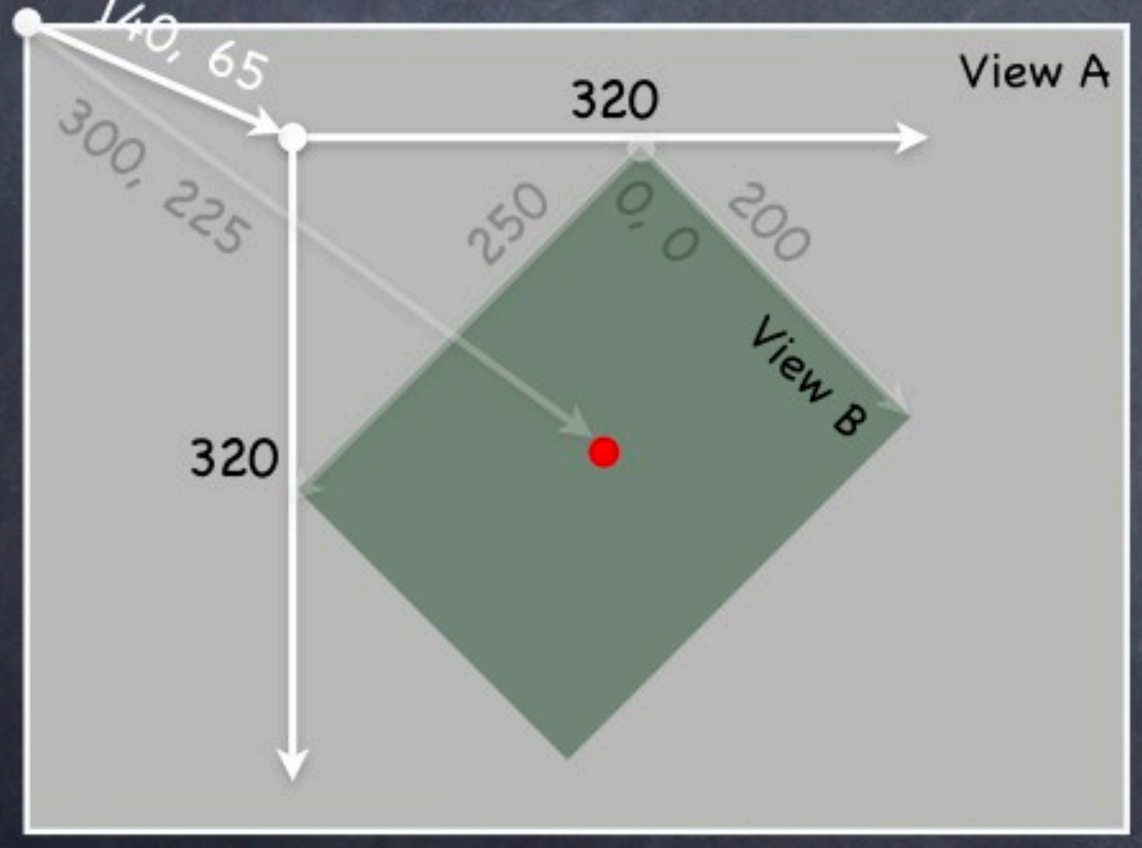
bounds = inward changes

Center

Coordinates

- Use `frame` and `center` to position the view in the hierarchy

  These are used by `superviews`, never inside your `UIView` subclass's implementation.
  You might think `frame.size` is always equal to `bounds.size`, but you'd be wrong ...



Because views can be rotated
(and scaled and translated too).

View B's `bounds` = `((0,0),(200,250))`

View B's `frame` = `((140,65),(320,320))`

View B's `center` = `(300,225)`

View B's middle in its own coordinate space is
`(bound.size.width/2+bounds.origin.x,`
`bounds.size.height/2+bounds.origin.y)`
which is `(100,125)` in this case.

Views are rarely rotated, but don't
misuse `frame` or `center` by assuming that.

Frame vs Bounds vs Center

frame = the smallest bounding box of that view with respect to it’s parents coordinate system, including any transformations applied to that view

bounds = a view's location and size using its own coordinate system

center = a view’s center relative to its superview using the parent view's coordinate system

Autoresizing mask

Autoresizing mask

Instance Property

autoresizesSubviews

A Boolean value that determines whether the receiver automatically resizes its subviews when its bounds change.

Declaration

```
var autoresizesSubviews: Bool { get set }
```

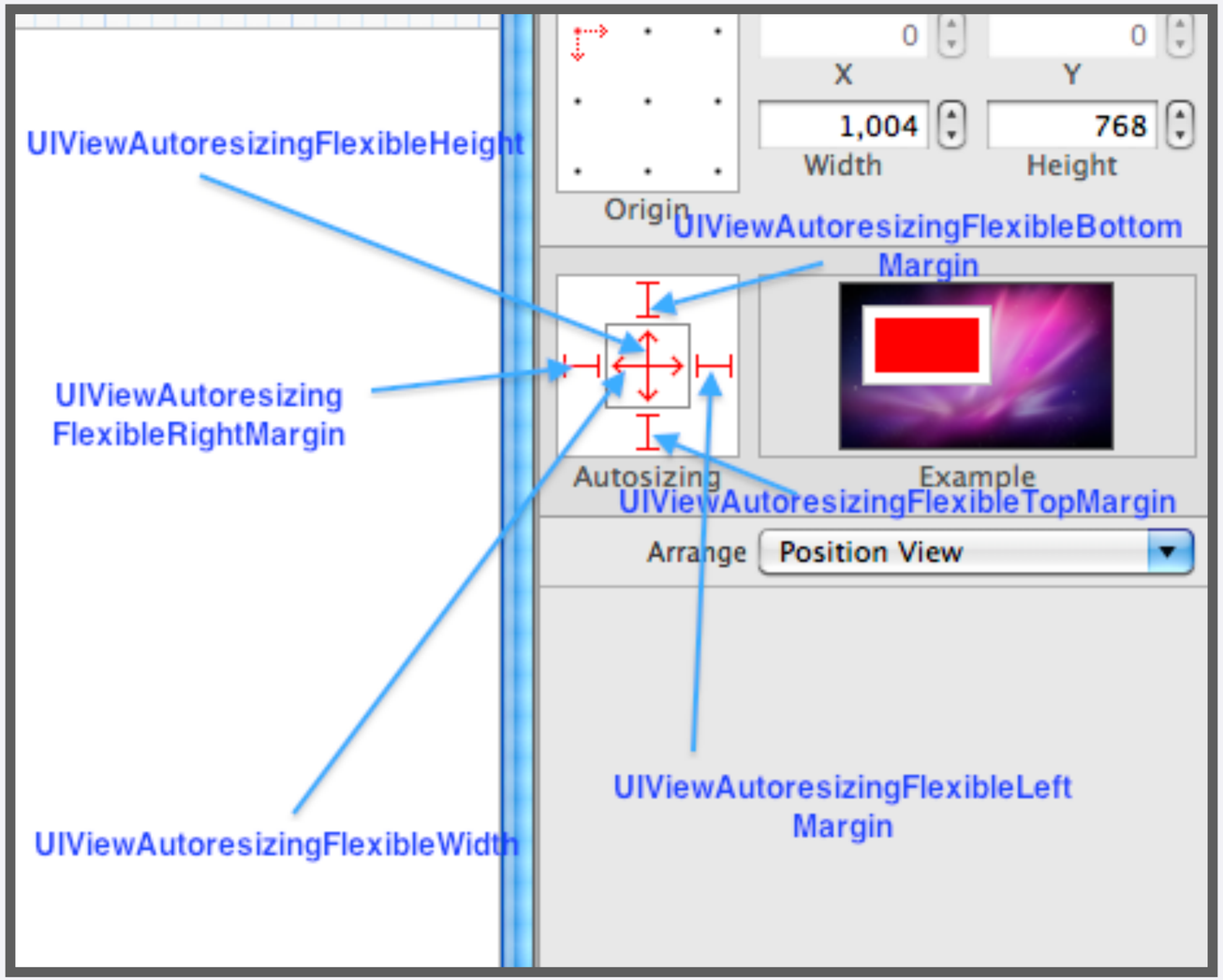
Discussion

When set to `true`, the receiver adjusts the size of its subviews when its bounds change. The default value is `true`.

Сбербанк Онлайн

Devteam

Autoresizing mask



Сбербанк Онлайн

Devteam

Autoresizing mask. OptionSet

```
public struct AutoresizingMask : OptionSet {

    public init(rawValue: UInt)


    public static var flexibleLeftMargin: UIView.AutoresizingMask { get }

    public static var flexibleWidth: UIView.AutoresizingMask { get }

    public static var flexibleRightMargin: UIView.AutoresizingMask { get }

    public static var flexibleTopMargin: UIView.AutoresizingMask { get }

    public static var flexibleHeight: UIView.AutoresizingMask { get }

    public static var flexibleBottomMargin: UIView.AutoresizingMask { get }
}
```

Autoresizing mask

Constants

`static var flexibleLeftMargin: UIView.AutoresizingMask`
Resizing performed by expanding or shrinking a view in the direction of the left margin.

`static var flexibleWidth: UIView.AutoresizingMask`
Resizing performed by expanding or shrinking a view's width.

`static var flexibleRightMargin: UIView.AutoresizingMask`
Resizing performed by expanding or shrinking a view in the direction of the right margin.

`static var flexibleTopMargin: UIView.AutoresizingMask`
Resizing performed by expanding or shrinking a view in the direction of the top margin.

`static var flexibleHeight: UIView.AutoresizingMask`
Resizing performed by expanding or shrinking a view's height.

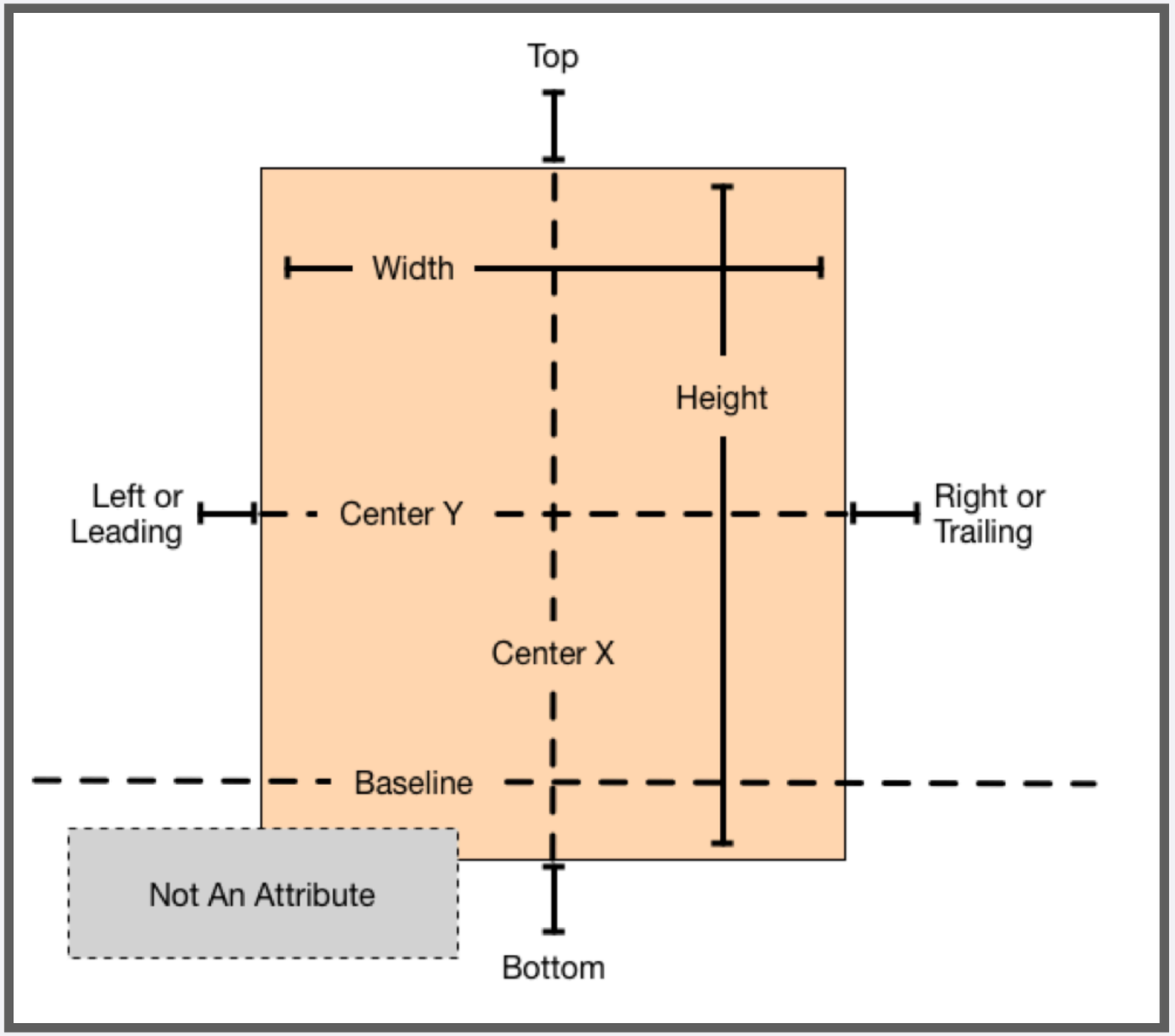
`static var flexibleBottomMargin: UIView.AutoresizingMask`
Resizing performed by expanding or shrinking a view in the direction of the bottom margin.

```
view.autoresizingMask = [.flexibleRightMargin, .flexibleLeftMargin, .flexibleBottomMargin]
```

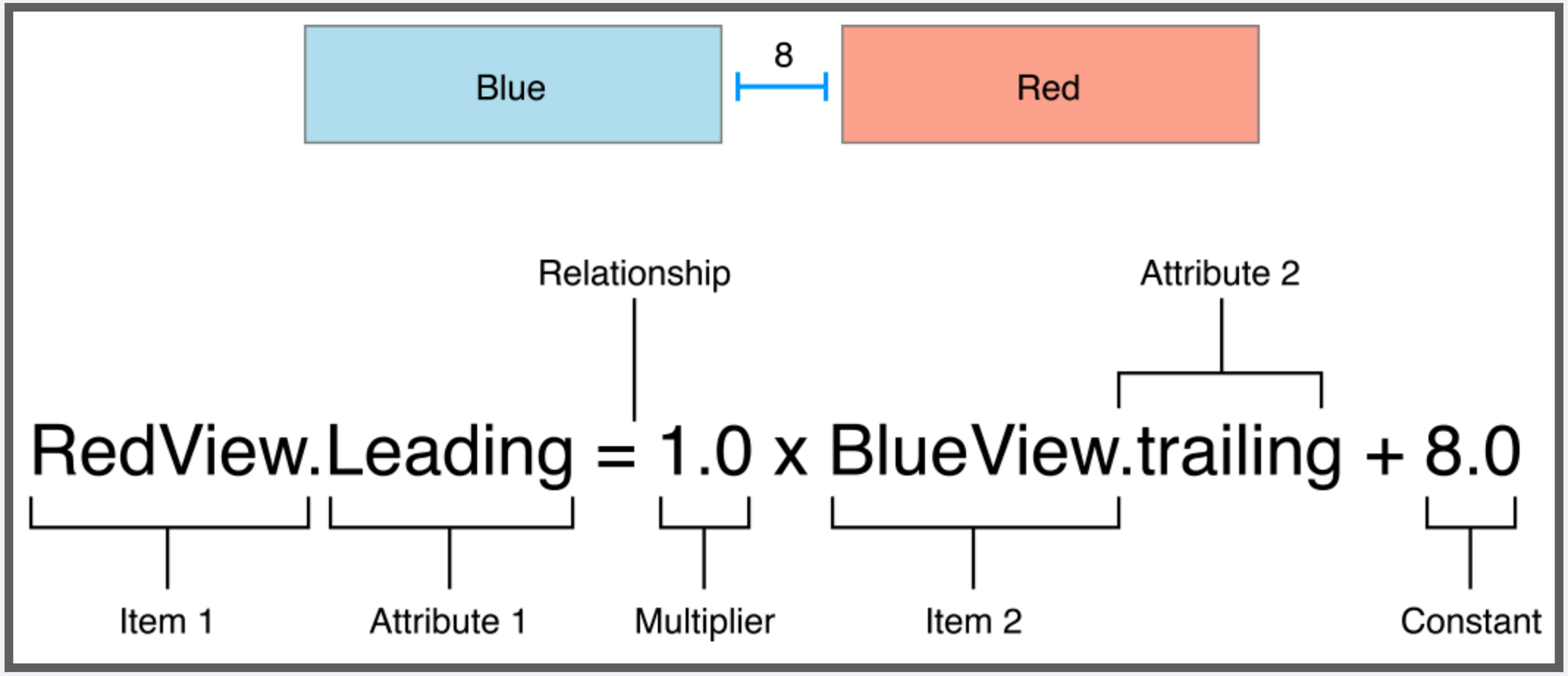
Autolayout

Сбербанк Онлайн

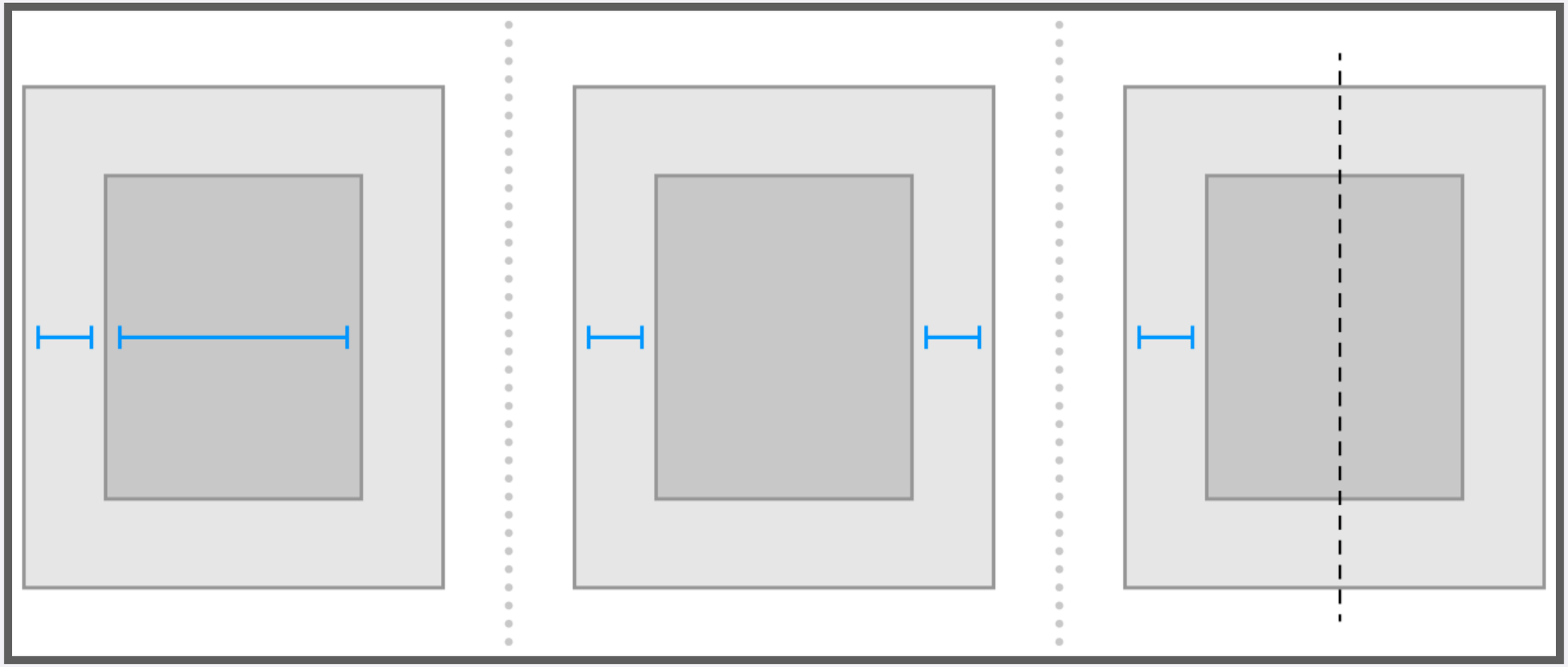
Devteam



Autolayout



Autolayout



Сбербанк Онлайн

Devteam

VS

```swift
@objc private func didPullRefreshControl() {
    delegate?.didPullToRefresh()
    refreshControl.endRefreshing()
}

private func setupLoadingView() {
    addSubview(loadingView)
    constrain(loadingView, self) { view, superview in
        view.edges == superview.edges
    }
}

private func setupEmptyStateView() {
    addSubview(emptyStateView)
    constrain(emptyStateView, self) { view, superview in
        view.edges == superview.edges
    }
}

func setup(delegate: WalletViewDelegate) {
    self.delegate = delegate
    tableView.dataSource = delegate
    tableView.delegate = delegate
    emptyStateView.setup(delegate: delegate)
}

func setupBackground(with color: UIColor) {
    loadingView.bgColor = color
}

func renderLoading() {
    emptyStateView.isHidden = true
    loadingView.show(animated: false, opacity: 1)
}
```

Сбербанк Онлайн

Devteam

Autolayout. Code

- Layout anchors
- NSLayoutConstraint class
- Visual Format Language

Generic Class

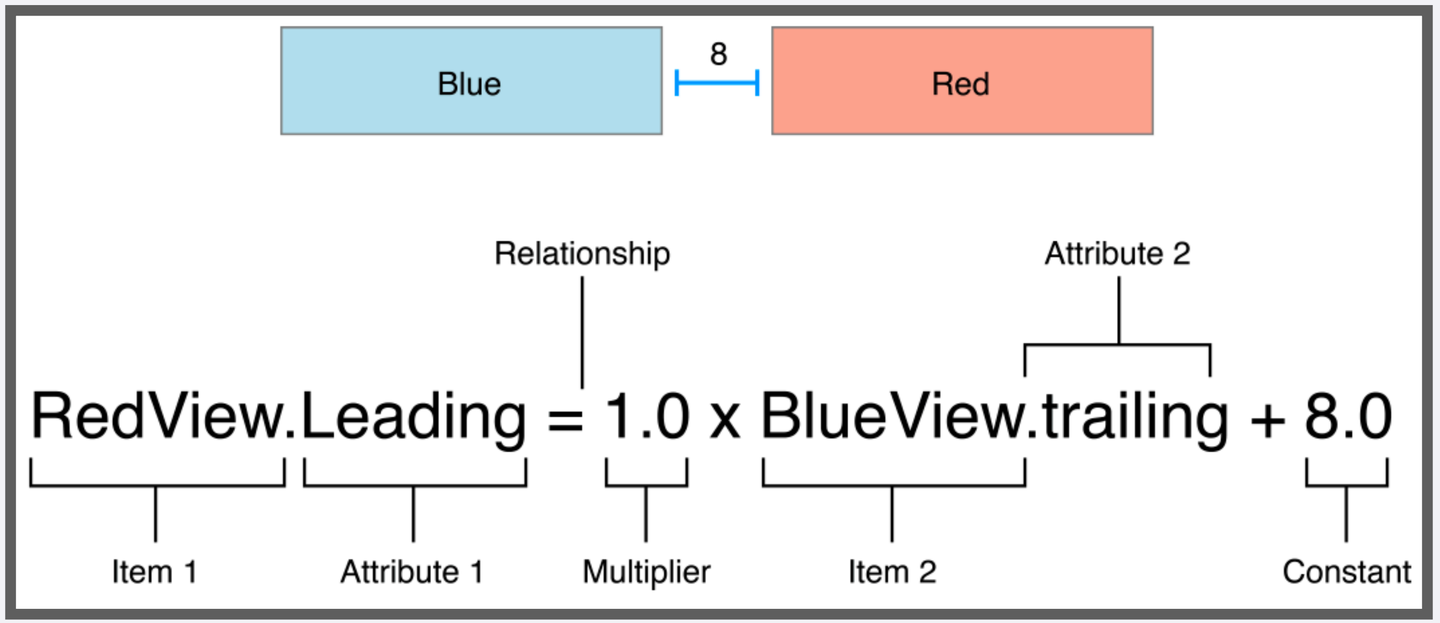
NSLayoutAnchor

A factory class for creating layout constraint objects using a fluent API.

Declaration

```
class NSLayoutAnchor<AnchorType> : NSObject where AnchorType : AnyObject
```

Autolayout. Anchor

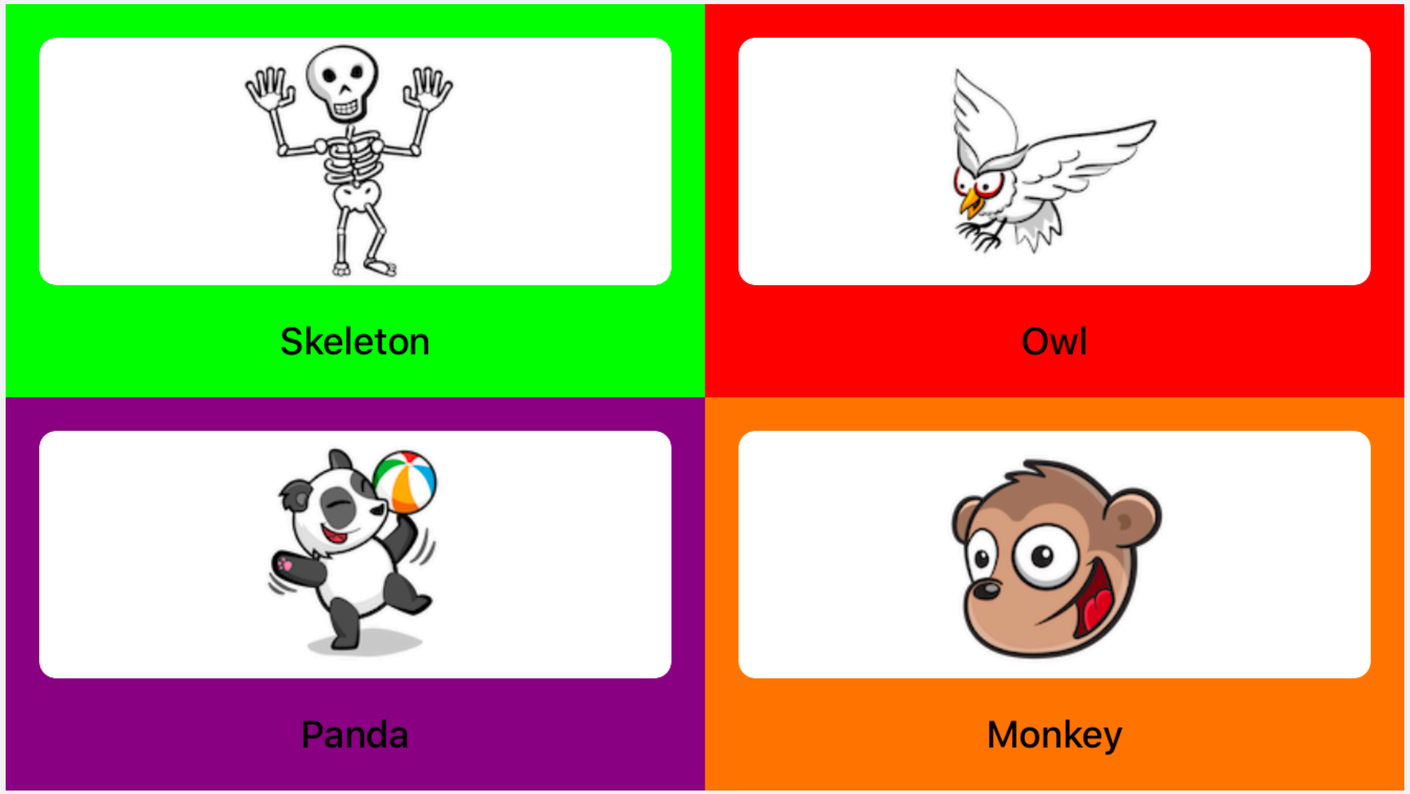



```
myView.leadingAnchor.constraint(equalTo: margins.leadingAnchor).isActive = true
```

Autolayout. Anchor

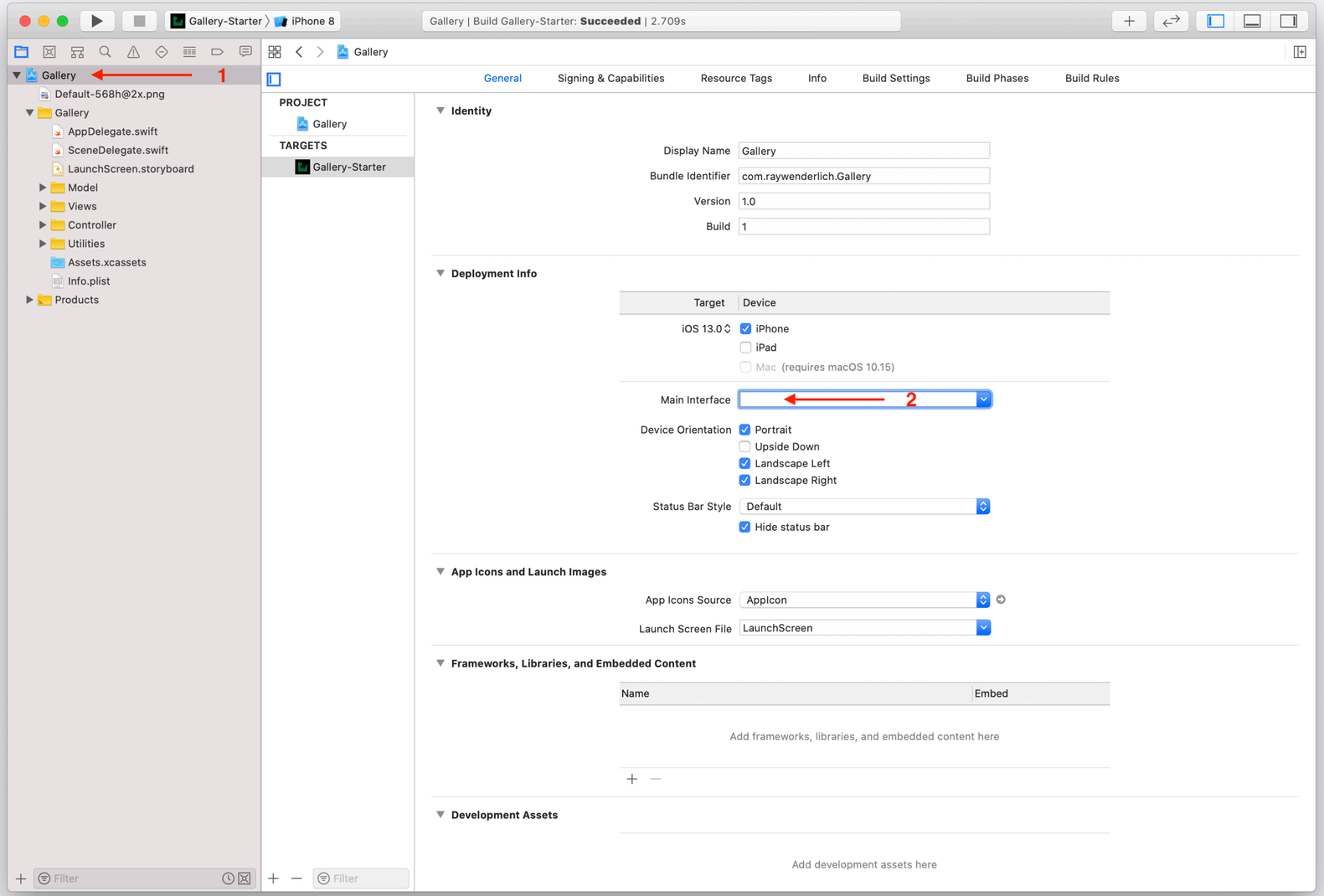
```swift
// Creating constraints using NSLayoutConstraint
NSLayoutConstraint(item: subview,
                   attribute: .leading,
                   relatedBy: .equal,
                   toItem: view,
                   attribute: .leadingMargin,
                   multiplier: 1.0,
                   constant: 0.0).isActive = true

NSLayoutConstraint(item: subview,
                   attribute: .trailing,
                   relatedBy: .equal,
                   toItem: view,
                   attribute: .trailingMargin,
                   multiplier: 1.0,
                   constant: 0.0).isActive = true


// Creating the same constraints using Layout Anchors
let margins = view.layoutMarginsGuide

subview.leadingAnchor.constraint(equalTo: margins.leadingAnchor).isActive = true
subview.trailingAnchor.constraint(equalTo: margins.trailingAnchor).isActive = true
```

Autolayout



Сбербанк Онлайн

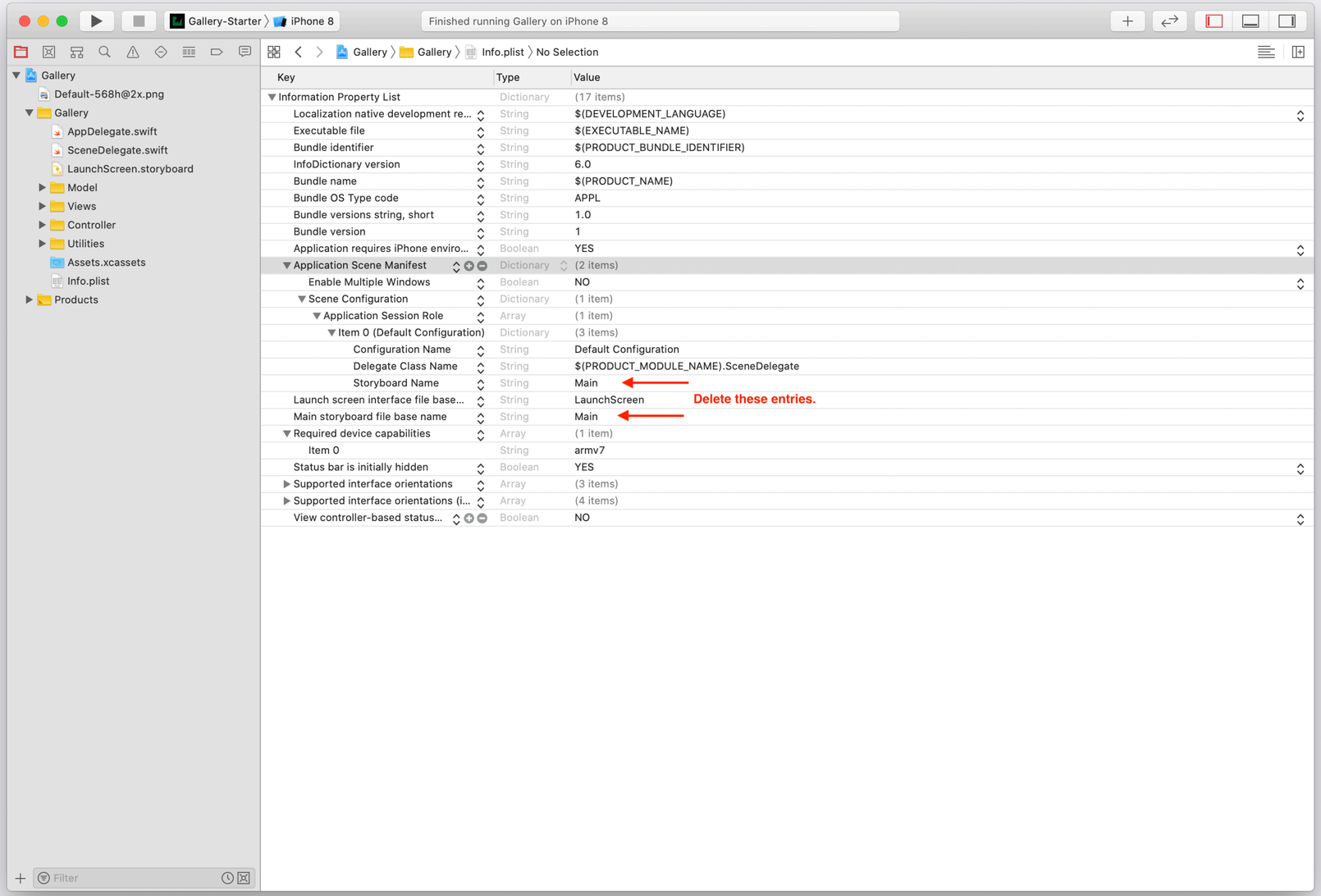
Autolayout. Code. 1/3



Сбербанк Онлайн

Devteam

Autolayout. Code. 2/3



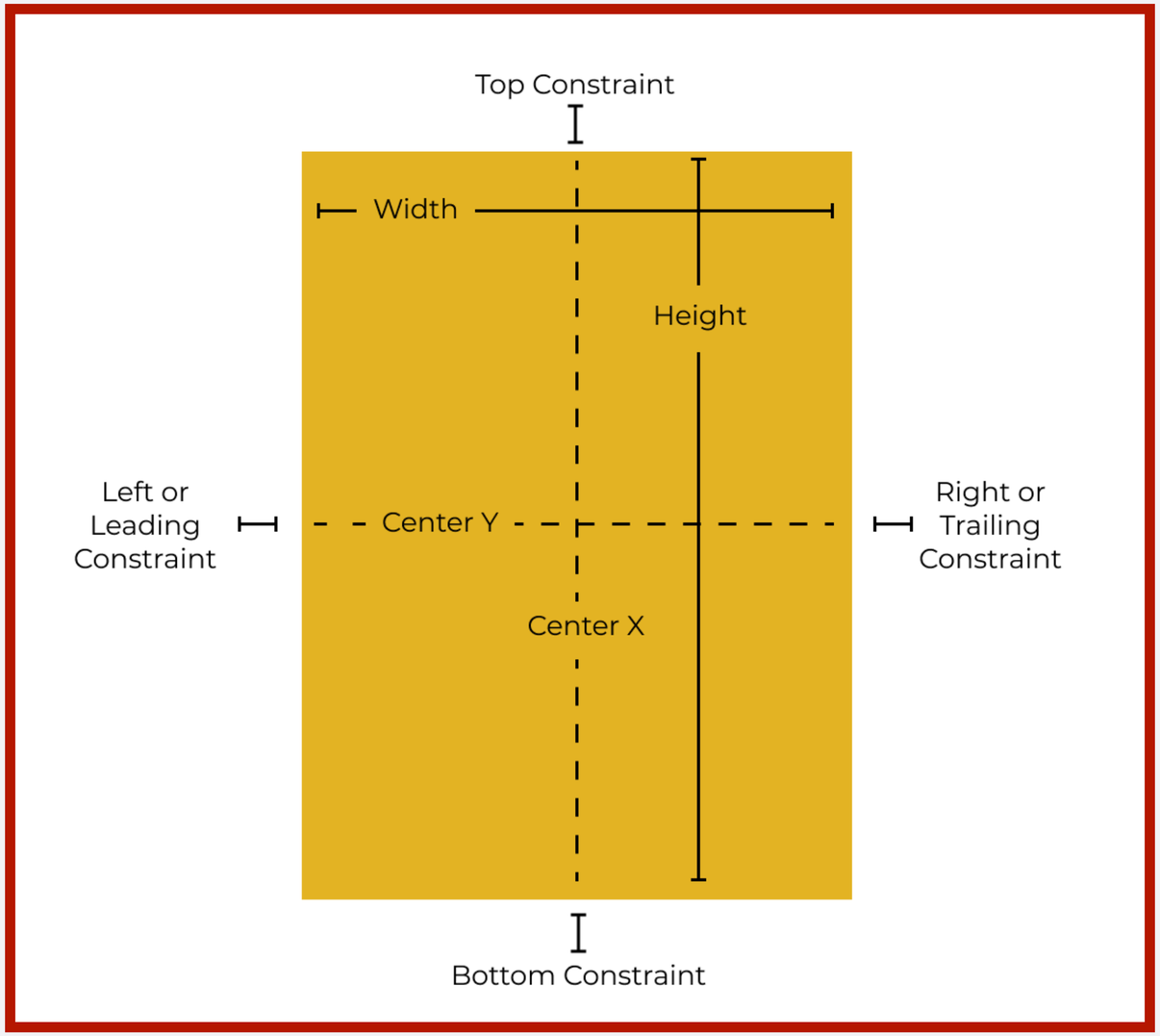
Сбербанк Онлайн

Devteam

Autolayout. Code. 3/3

```
  func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options
connectionOptions: UIScene.ConnectionOptions) {
    guard let windowScene = scene as? UIWindowScene else { return }

    window = UIWindow(frame: windowScene.coordinateSpace.bounds)

    window?.windowScene = windowScene

    window?.rootViewController = GalleryController()

    window?.makeKeyAndVisible()
  }
```

Autolayout



Сбербанк
Онлайн

Devteam

Autolayout

```
lazy var cardView1: CardView = {
  let cardView = CardView(with: .skeleton)
  cardView.translatesAutoresizingMaskIntoConstraints = false
  return cardView
}()
```

Autolayout

Instance Property

translatesAutoresizingMaskIntoConstraints

A Boolean value that determines whether the view’s autoresizing mask is translated into Auto Layout constraints.

Declaration

```
var translatesAutoresizingMaskIntoConstraints: Bool { get set }
```

Autolayout

```
private func setupConstraints() {
  let safeArea = view.safeAreaLayoutGuide

  NSLayoutConstraint.activate([
    cardView1.leadingAnchor.constraint(equalTo: safeArea.leadingAnchor),
    cardView1.topAnchor.constraint(equalTo: safeArea.topAnchor),
    cardView1.widthAnchor.constraint(equalTo: safeArea.widthAnchor, multiplier: 0.5),
    cardView1.heightAnchor.constraint(equalTo: safeArea.heightAnchor, multiplier: 0.5),
  ])
}
```

Autolayout

Type Method

activate(_:)

Activates each constraint in the specified array.

Declaration

```
class func activate(_ constraints: [NSLayoutConstraint])
```

Parameters

constraints

An array of constraints to activate.

Сбербанк Онлайн

Devteam

Autolayout. Content Hugging

Сбербанк Онлайн

Devteam

Autolayout. Content Hugging

```
textLabel.setContentHuggingPriority(.init(rawValue: 252), for: .vertical)
textLabel.setContentCompressionResistancePriority(.init(rawValue: 751), for: .vertical)
```

Сбербанк Онлайн

Devteam

Работа в группах

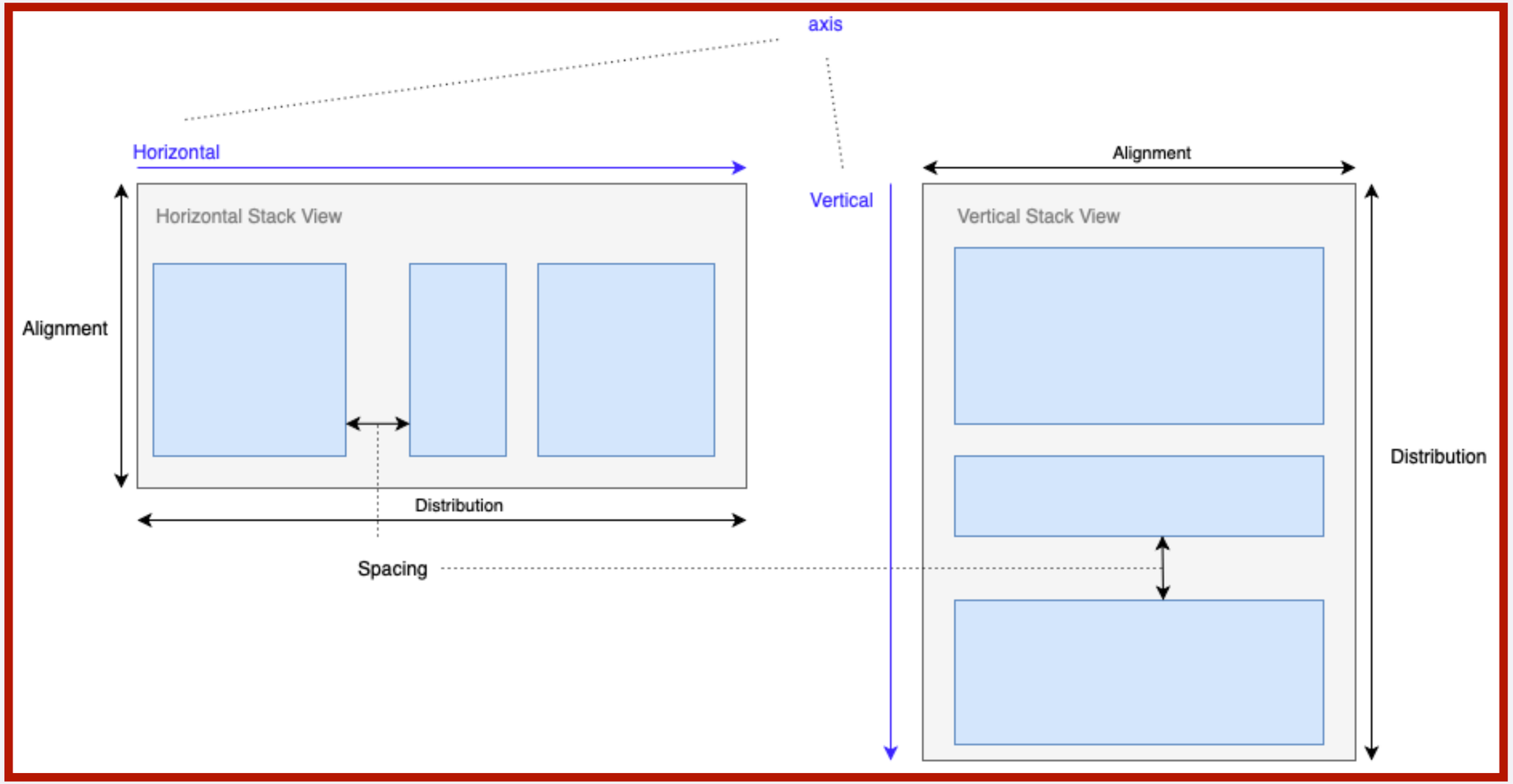
- Сделать тестовый проект, в котором надо управлять позицией и размером **3 view**.
- Используем 2 подхода: **autoresizing mask** и **autolayout+anchors**.
- Верстка только в коде.
- Надо поддержать поворот экрана
- Выведите размеры frame, bounds и значение center.

Сбербанк Онлайн

Devteam

StackView

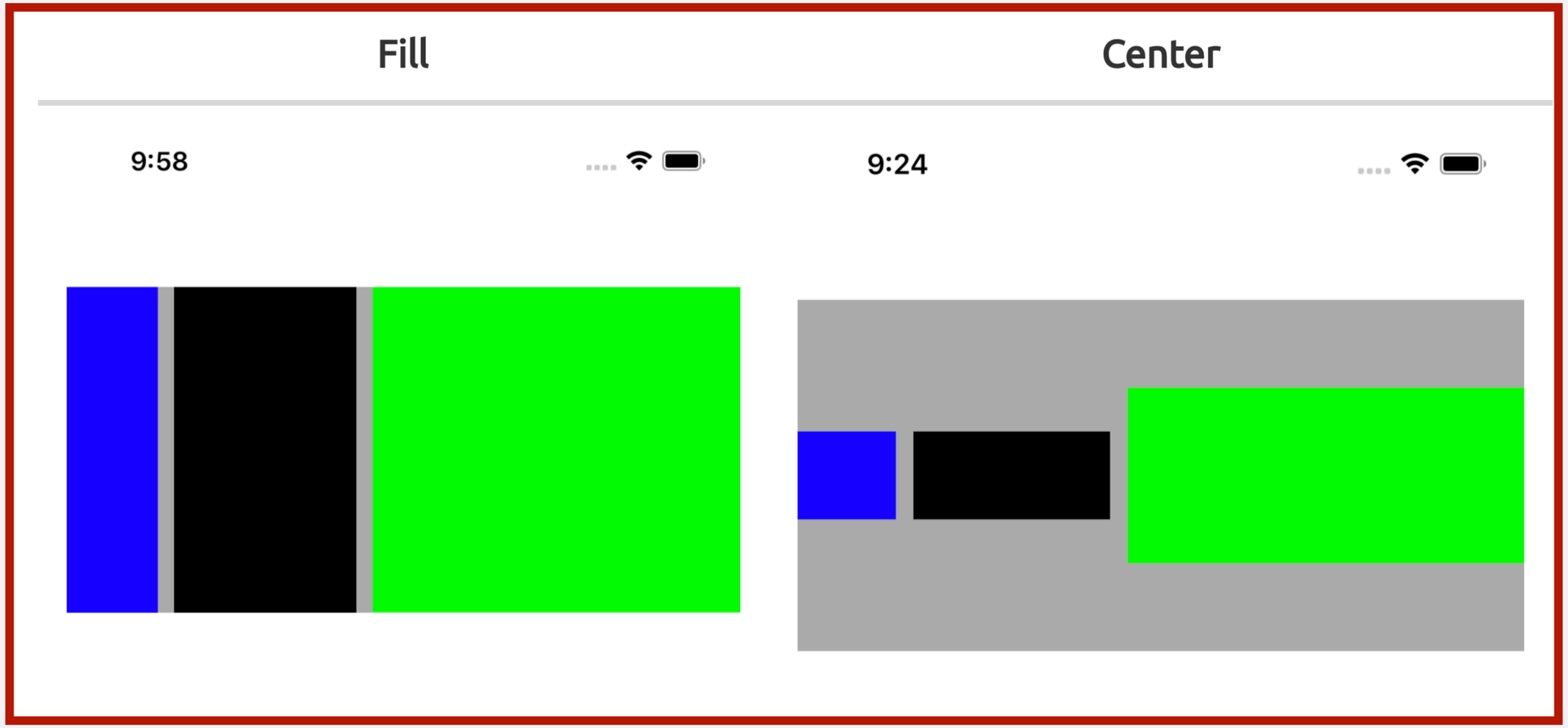
StackView



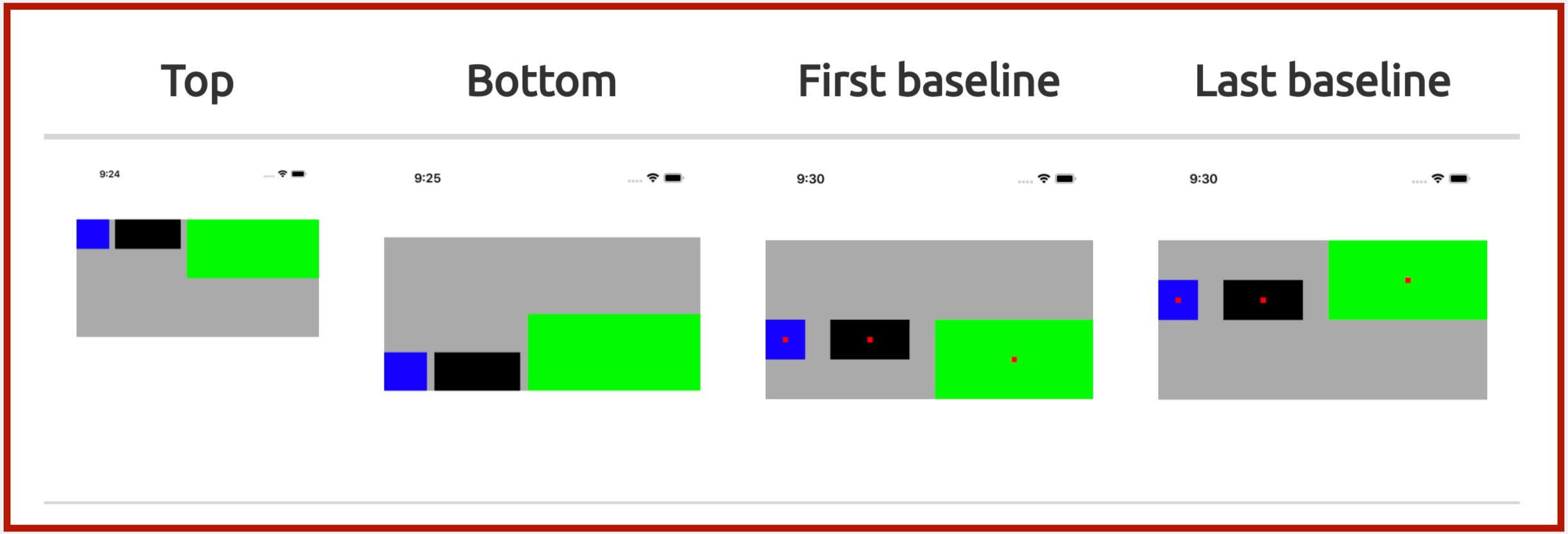
Сбербанк Онлайн

Devteam

StackView



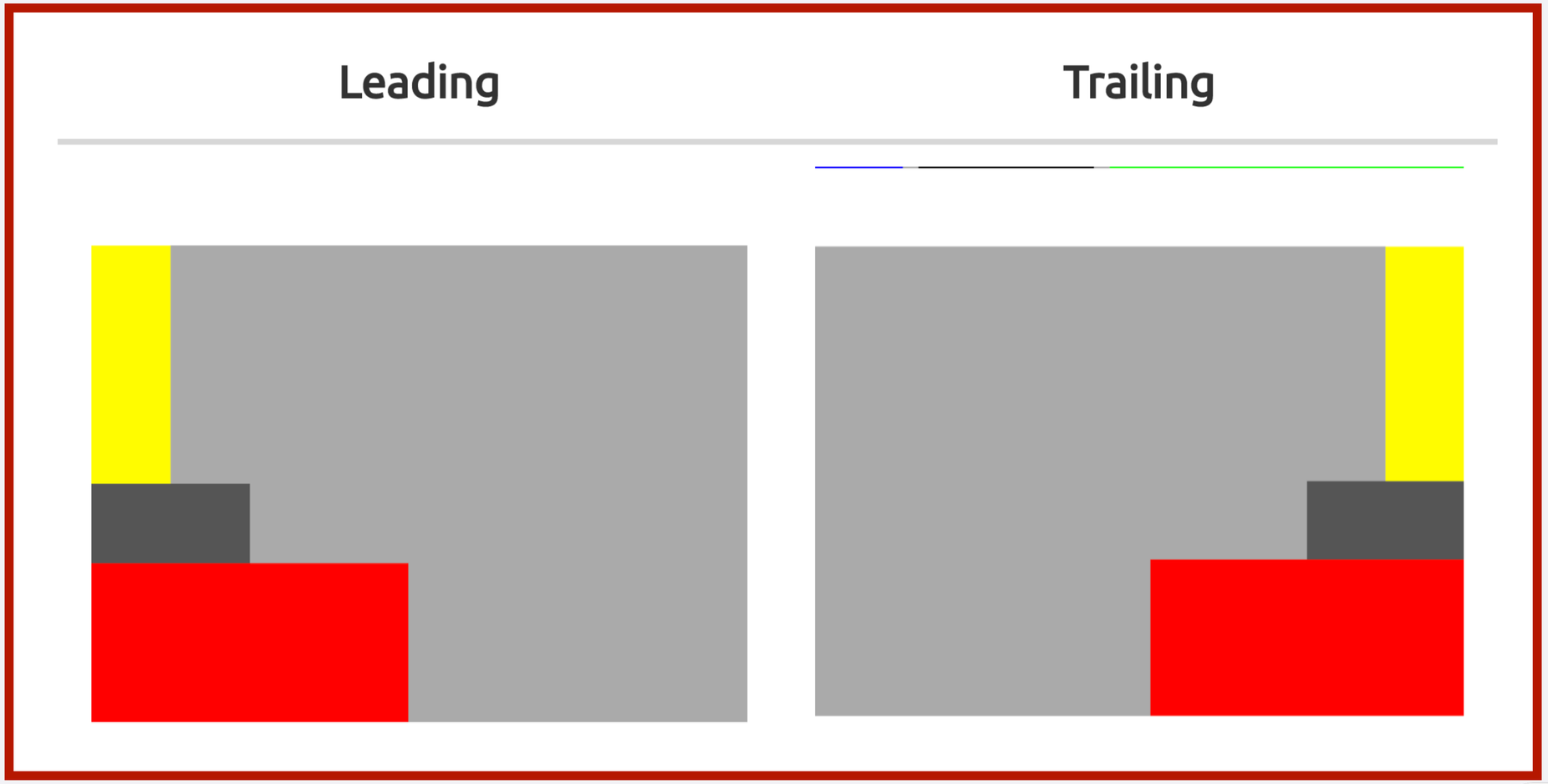
StackView. Horizontal



Сбербанк Онлайн

Devteam

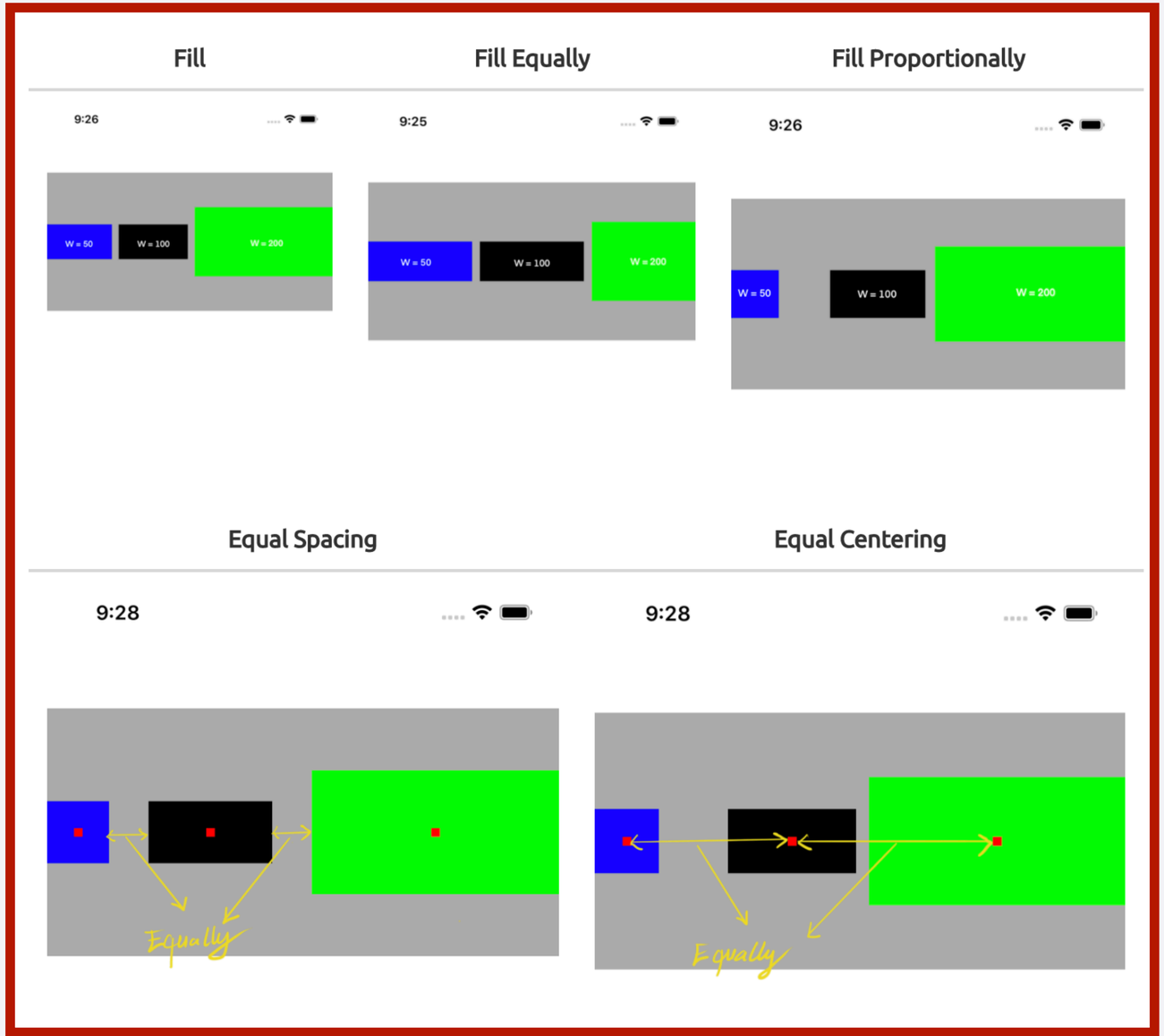
StackView. Vertical



Сбербанк Онлайн

Devteam

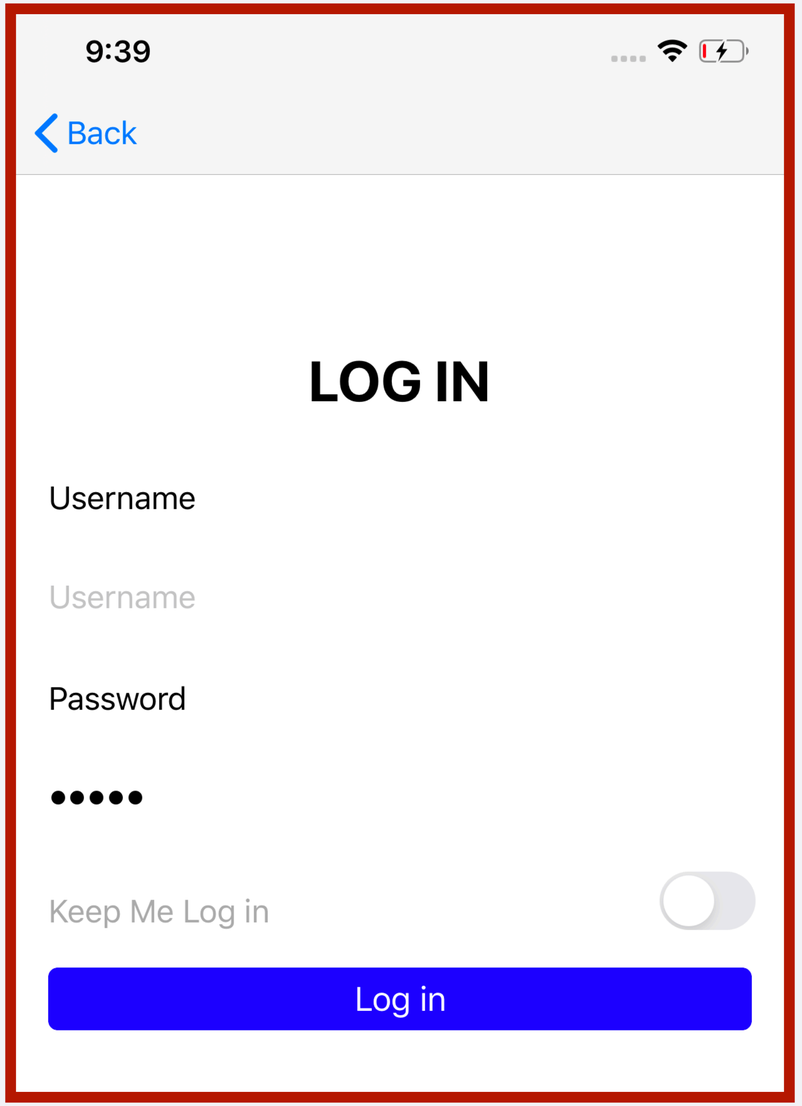
StackView. Distribution



Сбербанк Онлайн

Devteam

StackView. Example

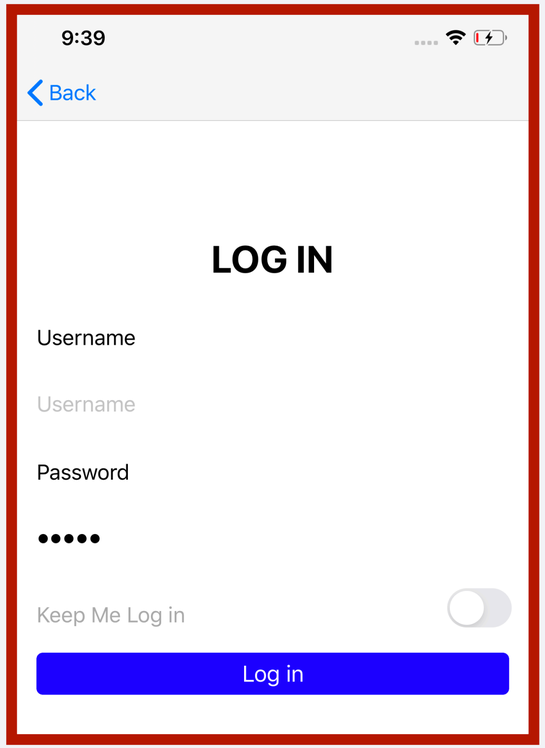


Сбербанк Онлайн

Devteam

StackView. Example

```
view.addSubview(lblLogin)
lblLogin.snp.makeConstraints { (make) in
    make.centerX.equalToSuperview()
    make.centerY.equalToSuperview().offset(-250)
    make.left.equalToSuperview().offset(20)
    make.right.equalToSuperview().offset(-20)
    make.height.equalTo(30)
}

view.addSubview(lblUsername)
lblUsername.snp.makeConstraints { (make) in
    make.centerX.left.right.equalTo(lblLogin)
    make.top.equalTo(lblLogin.snp.bottom).offset(30)
    make.height.equalTo(30)
}

view.addSubView(btnLogin)
//...
// The rest omitted
```

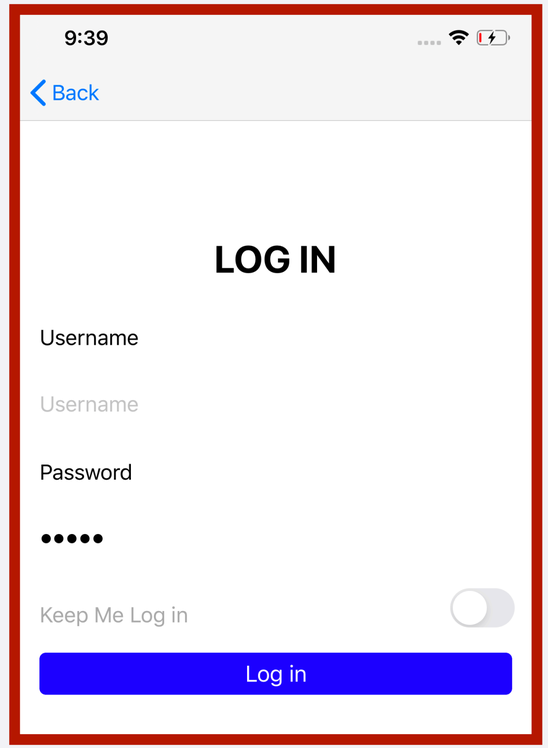


Сбербанк Онлайн

Devteam

StackView. Example

```
lazy var stackView: UIStackView = {
    let stack = UIStackView()
    stack.axis = .vertical
    stack.spacing = 20.0
    stack.alignment = .fill
    stack.distribution = .fillEqually
    [self.lblUsername,
        self.txtUserName,
        self.lblPassword,
        self.txtPassword,
        self.btnLogin].forEach { stack.addArrangedSubview($0) }
    return stack
}()
```

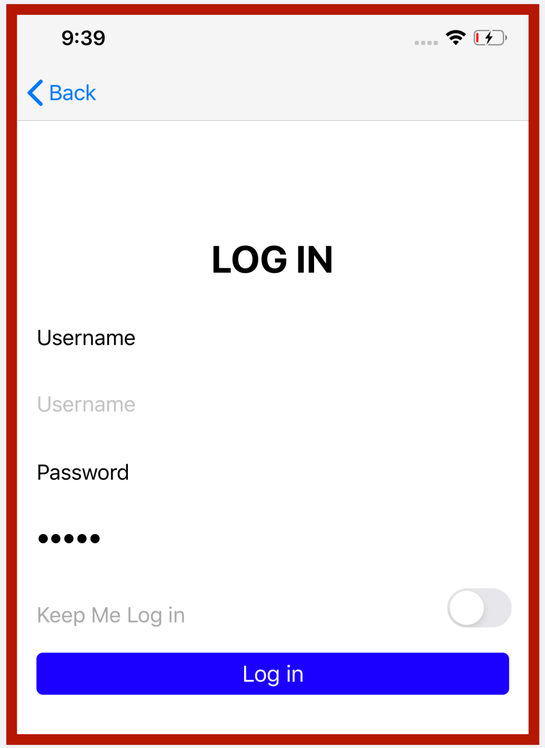


Сбербанк Онлайн

Devteam

StackView. Example

```
override func viewDidLoad() {
    super.viewDidLoad()
    // ...
    view.addSubview(stackView)
    stackView.snp.makeConstraints { (make) in
        make.centerX.left.right.equalTo(lblLogin)
        make.top.equalTo(lblLogin.snp.bottom).offset(30)
        make.height.equalTo(280)
    }
}
```



Сбербанк Онлайн

Devteam

Read

- https://ashfurrow.com/blog/i-totally-didnt-understand-frames-and-bounds/
- https://code.tutsplus.com/tutorials/ios-fundamentals-frames-bounds-and-cggeometry--cms-21196
- https://developer.apple.com/documentation/uikit/uiview/autoresizingmask
- https://developer.apple.com/library/archive/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/CreatingViews/CreatingViews.html
- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html
- https://sarunw.com/posts/history-of-auto-layout-constraints/
- https://habr.com/ru/company/oleg-bunin/blog/437584/
- https://www.vadimbulavin.com/effective-auto-layout-programmatically-in-swift/
- https://developer.apple.com/library/archive/technotes/tn2154/_index.html
- https://uynguyen.github.io/2020/07/18/iOS-Introducing-Stack-Views/