Stevia Layout

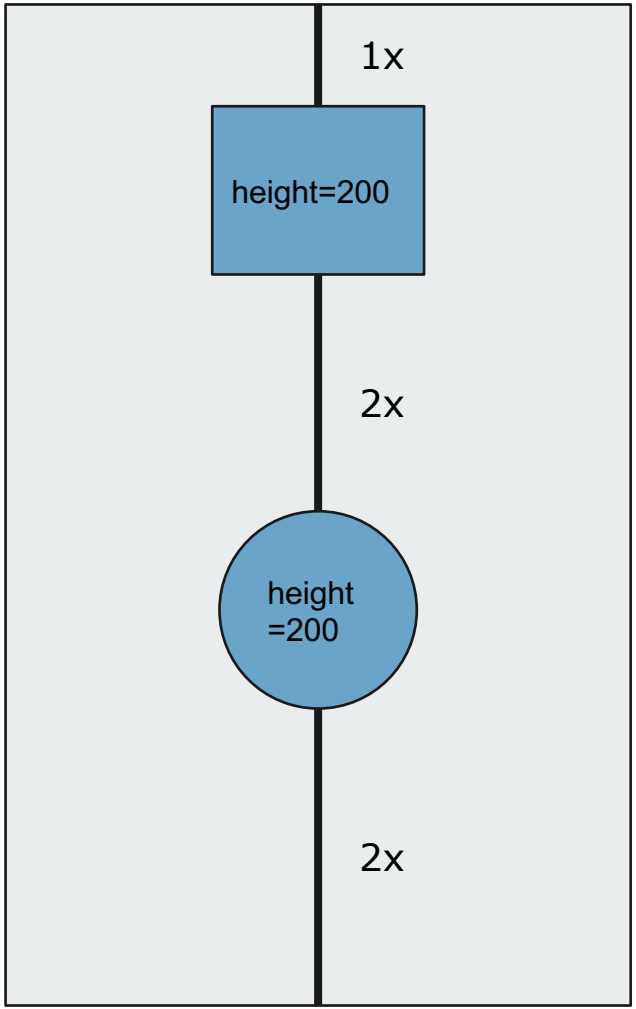
Layout giao diện bằng cú pháp Stevia

cuong@techmaster.vn

Stevia không xử lý các khoảng cách tỷ lệ nhau

- Stevia tạo được ràng buộc width, heigh các UIView tỷ lệ với nhau
- Nhưng chưa có cơ chế tạo tỷ lệ cho các khoảng cách giữa các view
    - Khoảng cách giữa các view không phải là thuộc tính của riêng một view nào cả

Khó khăn khi sử dụng Autolayout chuẩn



Khi Autolayout với những khoảng cách có tỷ lệ với nhau như hình, trong giao diện kéo thả của XCode, chúng ta không ràng buộc được tỷ lệ giữa các khoảng cách từ vật thể này đến vật thể kia.

Cách xử lý là phải kết hợp giữa kéo thả và ràng buộc bằng code việc bảo trì nâng cấp giao diện trở nên phức tạp, mong manh dễ lỗi

Apple biết rõ nhược điểm của Autolayout

Apple nỗ lực học hỏi những kinh nghiệm tốt nhất từ Android, Flutter và Responsive web.

Apple tung ra Swift UI vào tháng 9/2019 để đơn giản việc layout giao diện.

Stevia

View Layout Library for iOS

Cài đặt Stevia #1

Stevia là thư viện layout giao diện bằng cú pháp, nó đóng gói trong CocoaPods. Để nhúng Stevia vào dự án việc đầu tiên phải cài phần mềm quản lý 3rd libraries CocoaPods.

CocoaPods được viết bằng Ruby. Có 2 cách cài:

1. Cài HomeBrew trên MacOSX
2. `$ brew install cocoapods`

```
$ sudo gem install cocoapods
```

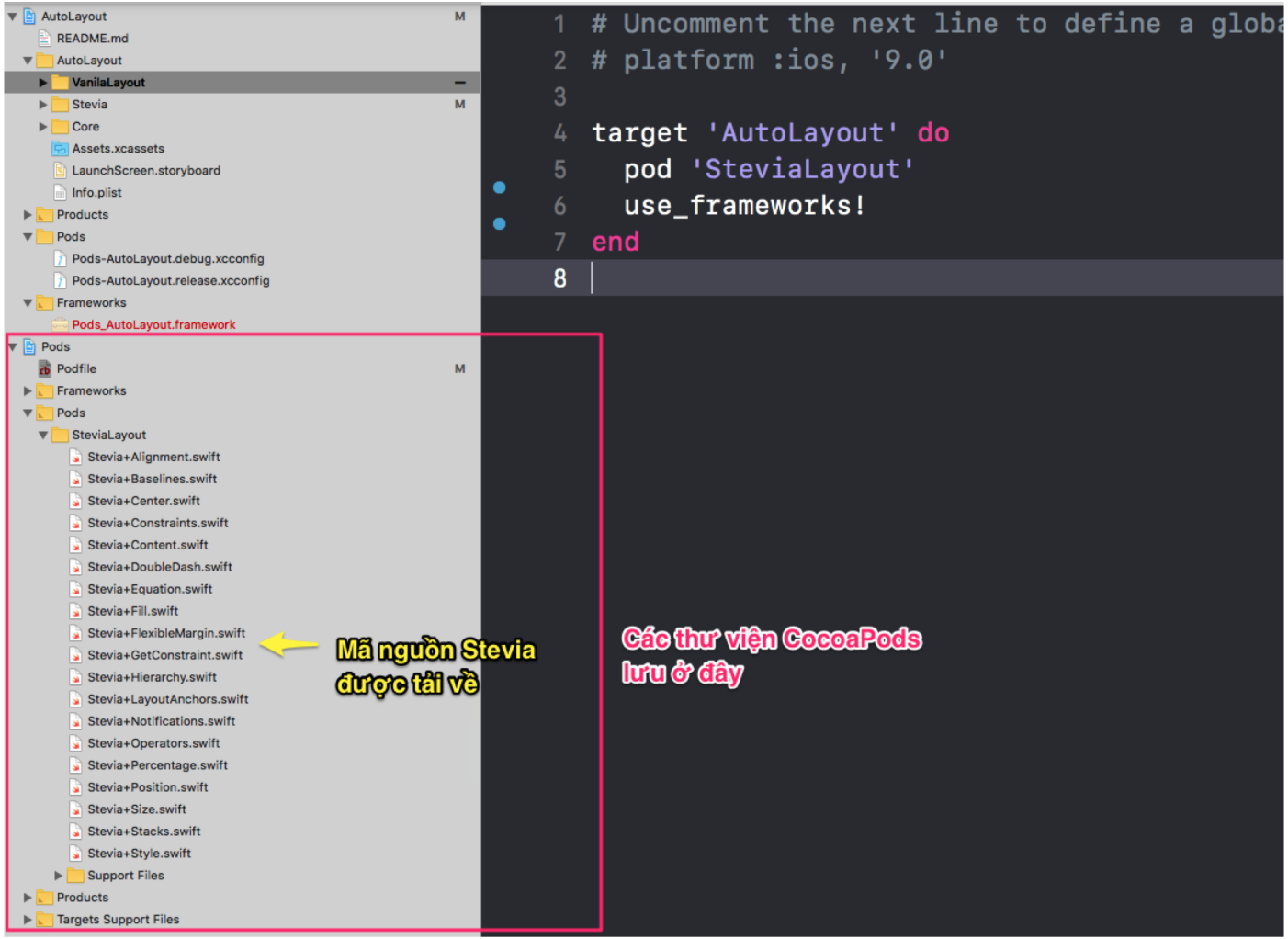
https://brew.sh

Cài đặt Stevia #2

1. Tạo dự án XCode Project
2. Chuyển vào thư mục dự án
3. Gõ lệnh **$ pod init**
4. Sửa **Podfile**
5. Gõ lệnh **$ pod install**
6. XCode Workspace sẽ được tạo ra từ bây giờ bạn chỉ mở XCode Workspace ra để lập trình đừng mở XCode Project ra nữa

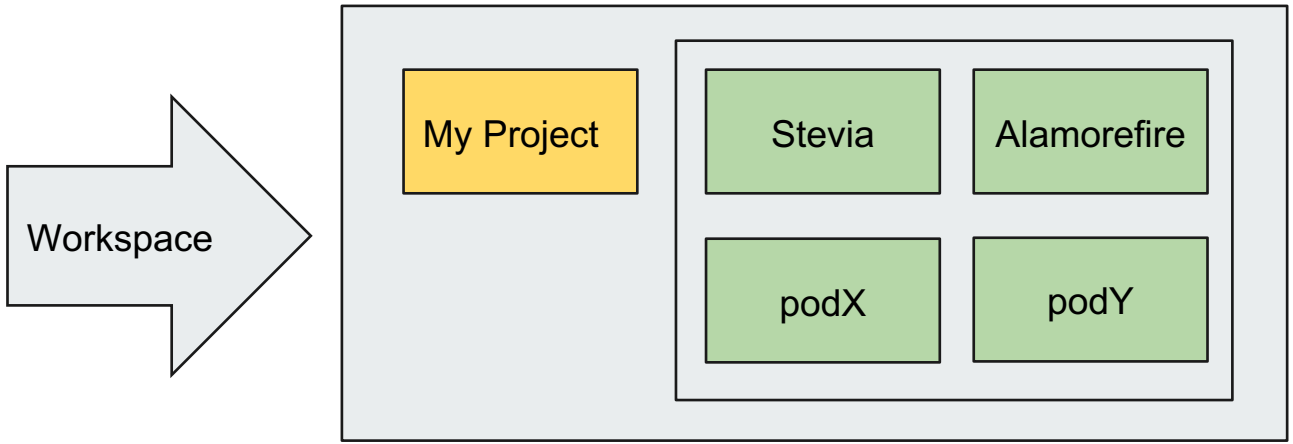
```
target 'AutoLayout' do
  pod 'SteviaLayout'
  use_frameworks!
end
```

```
.../HocAutolayout ✔ ls
HocAutolayout                 HocAutolayout.xcworkspace Podfile.lock
HocAutolayout.xcodeproj       Podfile                   Pods
```


AutoLayout
README.md
AutoLayout
VanilaLayout
Stevia
Core
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
Pods
Pods-AutoLayout.debug.xcconfig
Pods-AutoLayout.release.xcconfig
Frameworks
Pods_AutoLayout.framework
Pods
Podfile
Frameworks
Pods
SteviaLayout
Stevia+Alignment.swift
Stevia+Baselines.swift
Stevia+Center.swift
Stevia+Constraints.swift
Stevia+Content.swift
Stevia+DoubleDash.swift
Stevia+Equation.swift
Stevia+Fill.swift
Stevia+FlexibleMargin.swift
Stevia+GetConstraint.swift
Stevia+Hierarchy.swift
Stevia+LayoutAnchors.swift
Stevia+Notifications.swift
Stevia+Operators.swift
Stevia+Percentage.swift
Stevia+Position.swift
Stevia+Size.swift
Stevia+Stacks.swift
Stevia+Style.swift
Support Files
Products
Targets Support Files
Uncomment the next line to define a globa
platform :ios, '9.0'
target 'AutoLayout' do
pod 'SteviaLayout'
use_frameworks!
end
Mã nguồn Stevia
được tải về
Các thư viện CocoaPods
lưu ở đây

Project khác WorkSpace như thế nào

WorkSpace gồm nhiều Project

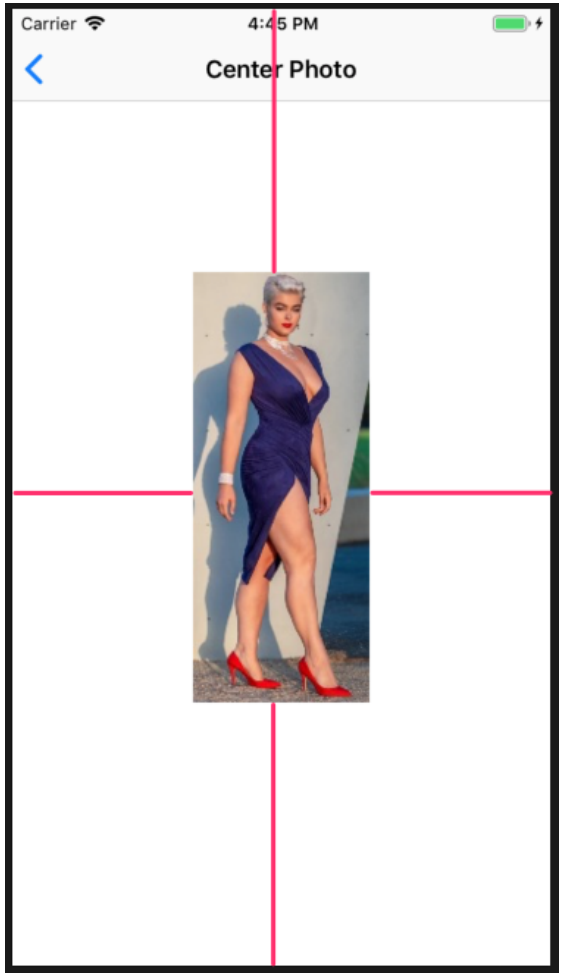


Ví dụ Stevia đơn giản

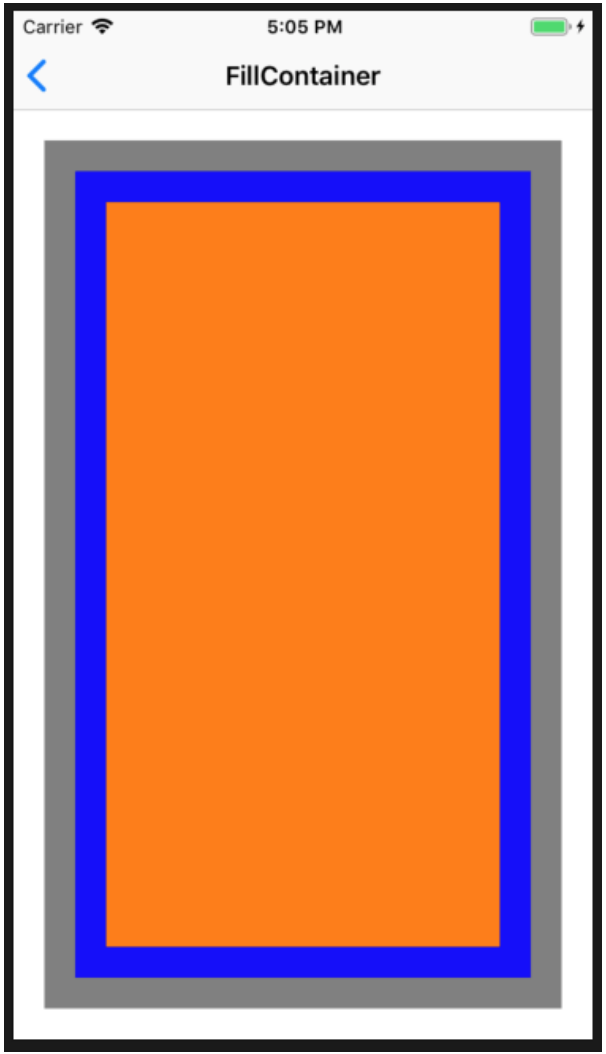
```swift
import UIKit
import Stevia
class CenterPhoto: UIViewController {

 let photo = UIImageView(image: UIImage.init(named:
"stefania"))
 override func viewDidLoad() {
   super.viewDidLoad()
   view.backgroundColor = UIColor.white
   view.sv(
     photo
   )
   /*
     photo.centerHorizontally()
     photo.centerVertically()
   */

   //Có thể thay bằng lệnh này
   photo.centerInContainer()
 }
}
```

view chứa photo

photo chính giữa màn hình

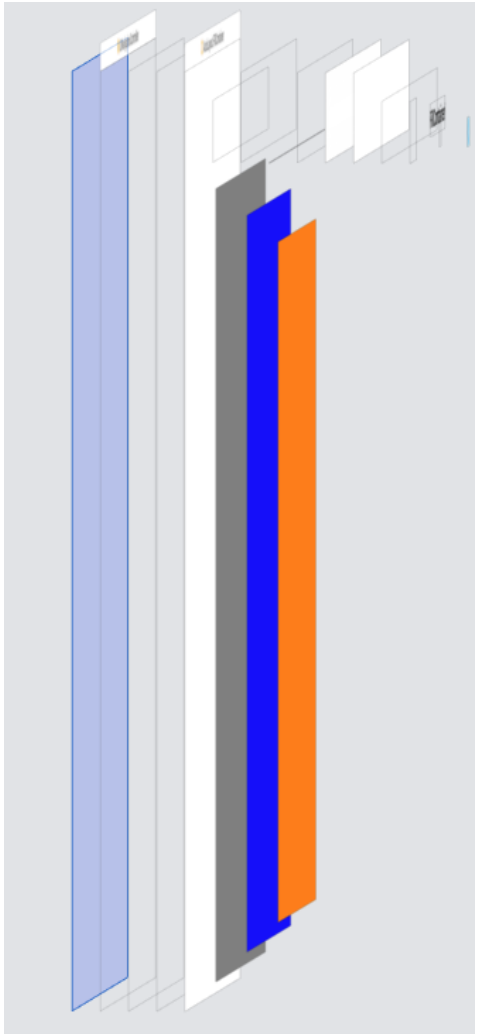


```
view.sv(
  rec1.sv(
    rec2.sv(
      rec3
    )
  )
)
rec1.Top == view.safeAreaLayoutGuide.Top + 20
view.layout (
 |-20-rec1-20-|,
 20
)

rec2.fillContainer(20)
rec3.fillContainer(20)
```



```
view.sv(
  rec1.sv(
    rec2.sv(
      rec3
   )
 )
)
```

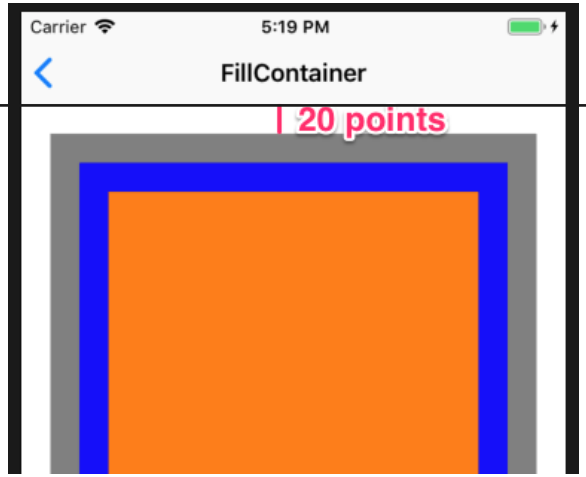
- view chứa rec1
- rec1 chứa rec2
- rec2 chứa rec3



view.layout

view.layout

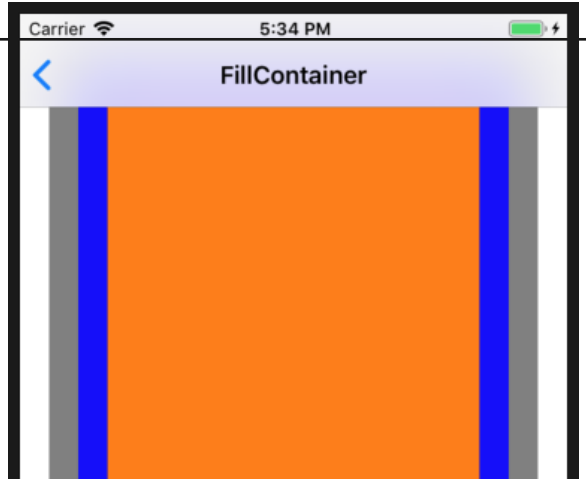
- view.sv mô tả cấu trúc cha – con giữa các UIView: ai chứa ai
- view.layout mô tả ràng buộc sắp xếp giữa các UIView
    - Quy ước theo các dòng từ trên xuống dưới, từ trái sang phải
    - Ký tự pipe | mô tả viền trái hoặc phải màn hình
    - Khi bỏ 1 trong 2 ký tự | có nghĩa chỉ tập trung rang buộc bám theo viền màn hình còn lại
    - Định được kích thước của một dòng bằng ~



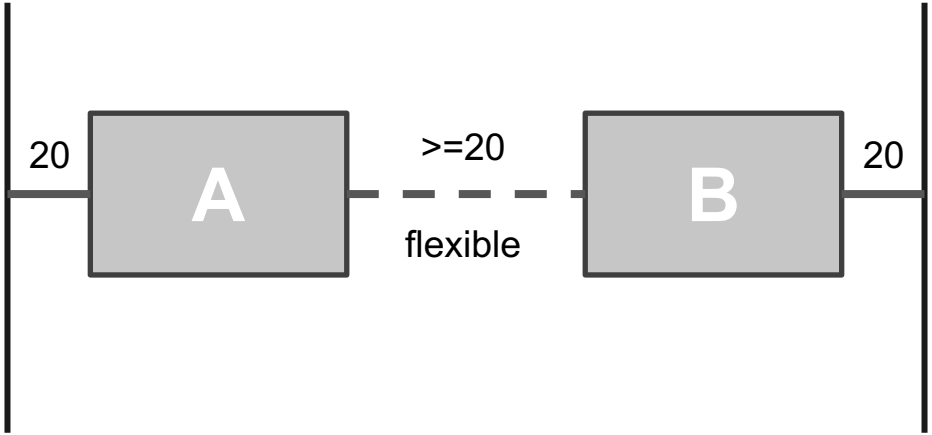
view.safeAreaLayoutGuide.top

```
rec1.Top == view.safeAreaLayoutGuide.Top + 20
view.layout (
  |-20-rec1-20-|,
  20
)
```

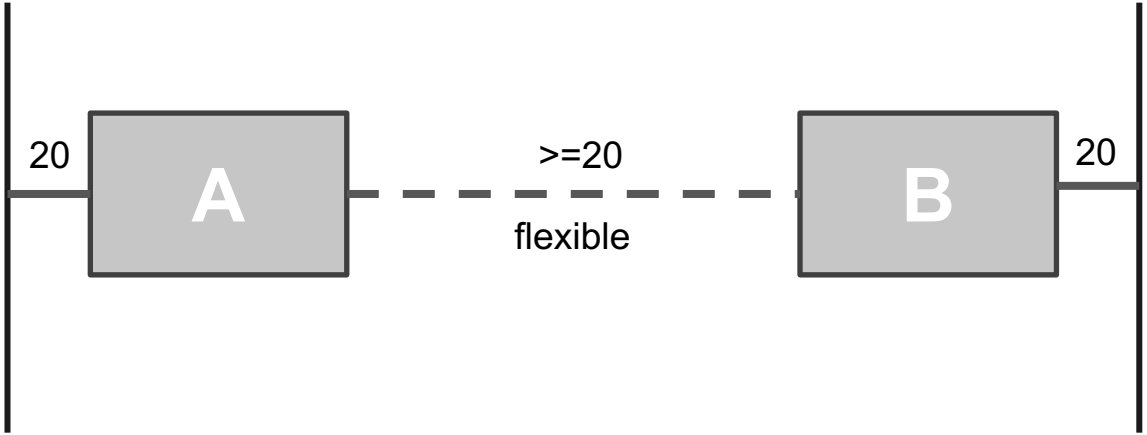
(0,0)

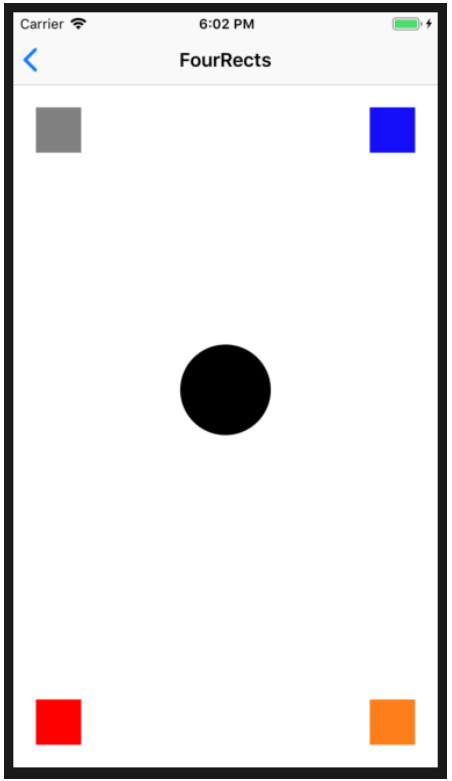



```
view.layout (
 20,
 |-20-rec1-20-|,
 20
)
```




```
view.layout (
    [redacted]
    |-20-A-(>=20)-B-20-|,
    [redacted]
)
```






```
import UIKit
import Stevia
class FourRects: UIViewController {

    let rec1 = UIView()
    let rec2 = UIView()
    let rec3 = UIView()
    let rec4 = UIView()
    let circle = UIView()
    let margin: CGFloat = 20
    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = UIColor.white
        view.sv(
            rec1,
            rec2,
            rec3,
            rec4,
            circle
        )
```

```
rec1.Top == view.safeAreaLayoutGuide.Top + margin
rec2.Top == view.safeAreaLayoutGuide.Top + margin

rec3.Bottom == view.safeAreaLayoutGuide.Bottom - margin
rec4.Bottom == view.safeAreaLayoutGuide.Bottom - margin

//width(80).height(80) có thể viết lại thành size(80)
view.layout (
    |-margin-rec1.size(40)-(>=20)-rec2.size(40)-margin-|,
    |-margin-rec3.size(40)-(>=20)-rec4.size(40)-margin-|
)

//circle.width(80).height(80).centerVertically().centerHorizontally()
circle.size(80).centerVertically().centerHorizontally()

rec1.backgroundColor = UIColor.gray
rec2.backgroundColor = UIColor.blue
rec3.backgroundColor = UIColor.red
rec4.backgroundColor = UIColor.orange
circle.backgroundColor = UIColor.black
circle.layer.cornerRadius = 40
```

Ràng buộc tọa độ safeAreaLayoutGuide

Lệnh format có thể nối chuỗi rất tiện

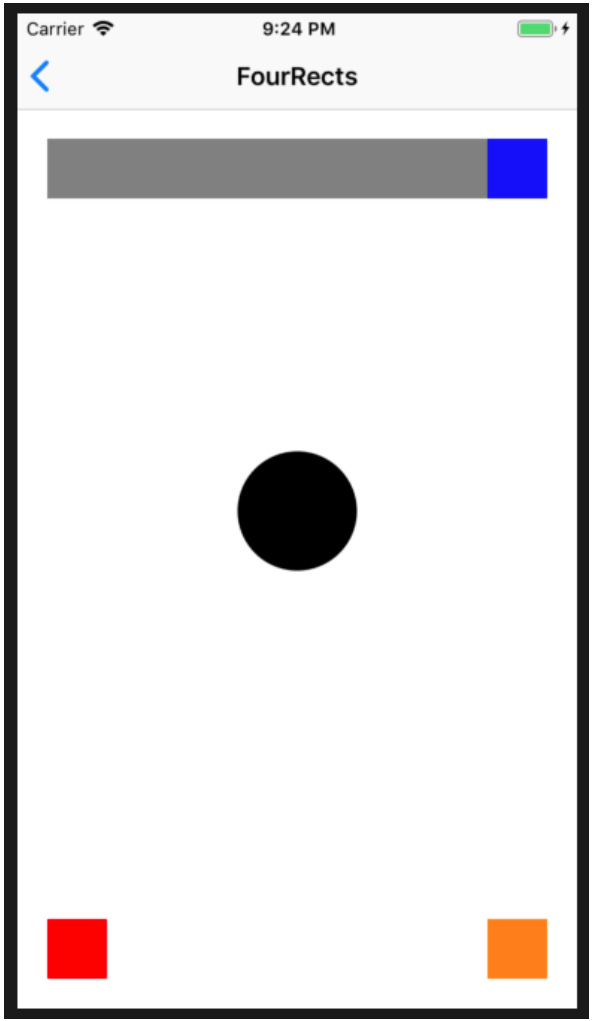
Cú pháp nối chuỗi

- Stevia cho phép nối chuỗi các ràng buộc đặt thuộc tính lên đối tượng
- Rút ngắn dòng code, lệnh nhìn trực quan dễ hiểu hơn

```
//circle.width(80).height(80).centerVertically().centerHorizontally()

circle.size(80).centerVertically().centerHorizontally()
```

Khoảng cách 2 đối tượng

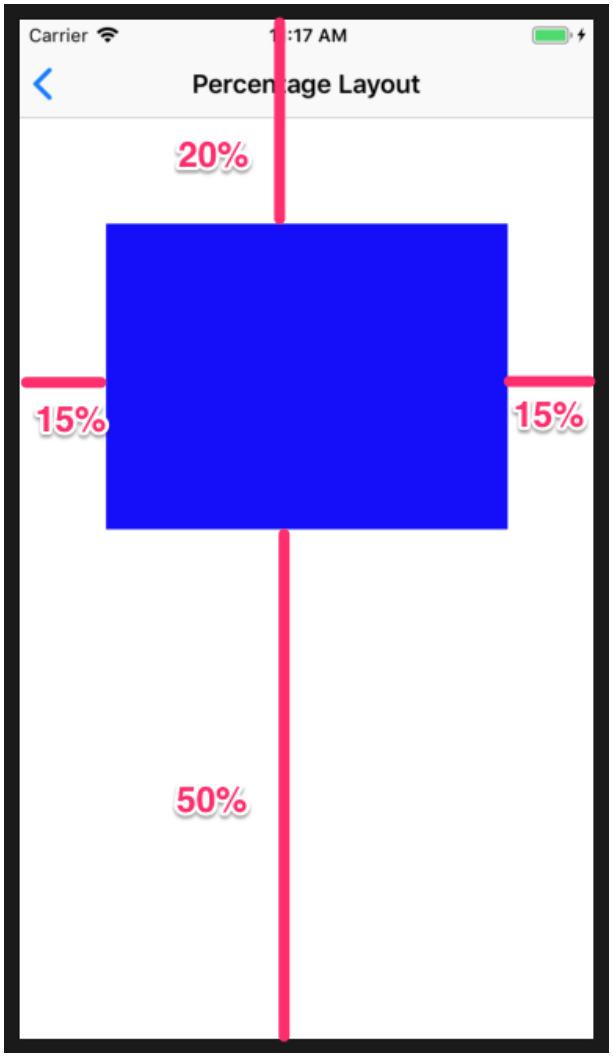
- Được mô tả bởi –(biểu thức giá trị)–
- Biểu thức giá trị có thể là
  - 0: hai đối tượng nằm sát nhau
  - >=xxx: lớn hơn hoặc bằng xxx
  - <=xxx: nhỏ hơn hoặc bằng xxx
  - xxx: bằng đúng xxx



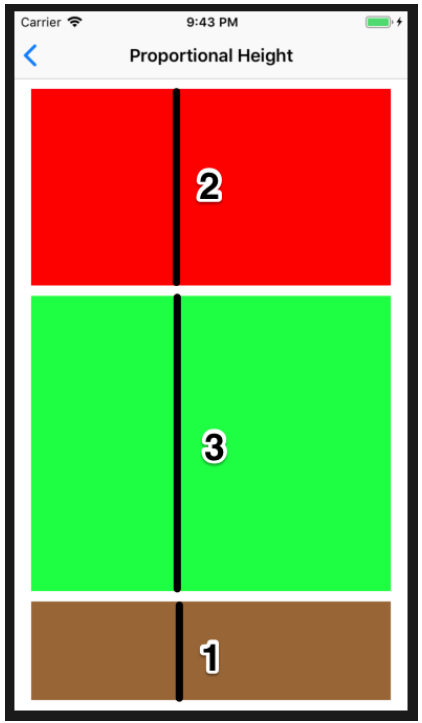

```
view.layout (
 |-margin-rec1-(0)-rec2.size(40)-margin-|,
 |-margin-rec3.size(40)-(>=20)-rec4.size(40)-margin-|
)
```

- rec1 và rec2 sát nhau
- rec3 và rec4 cách nhau 1 khoảng tối thiểu 20 points

Đặt kích thước, tọa độ theo tỷ lệ

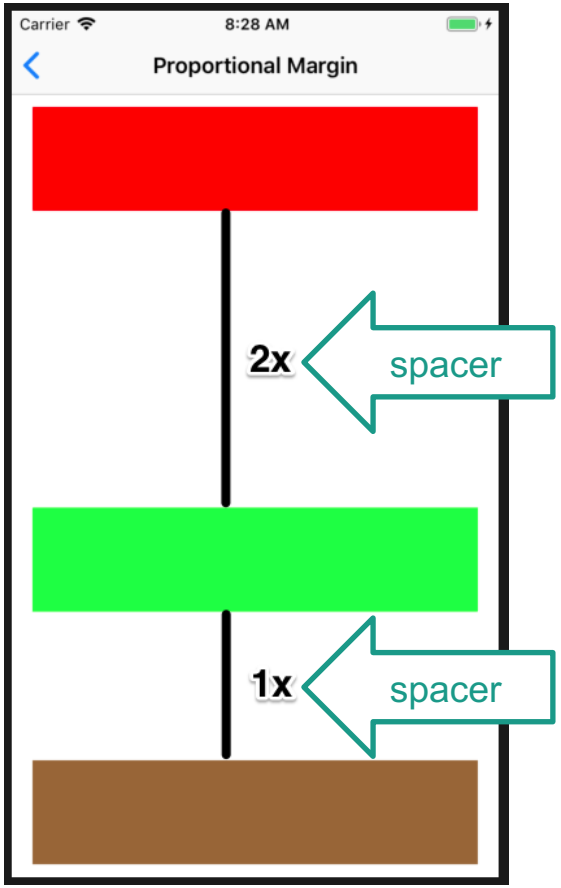



```
view.sv(view1)
view1.backgroundColor = UIColor.blue
view1.top(20%)
view1.left(15%)
view1.bottom(50%)
view1.right(15%)
```




```
view.sv(
  red_rect1,
  red_rect2,
  red_rect3
)
let margin: CGFloat = 10.0
view.layout(
  |-(16)-red_rect1-(16)-|,
  margin,
  |-(16)-red_rect2-(16)-|,
  margin,
  |-(16)-red_rect3-(16)-|
)
red_rect1.Top == view.safeAreaLayoutGuide.Top + margin
red_rect3.Bottom == view.safeAreaLayoutGuide.Bottom - margin
red_rect1.Height == red_rect3.Height * 2.0
red_rect2.Height == red_rect3.Height * 3.0
```

Khoảng cách giữa các đối tượng cần theo tỷ lệ

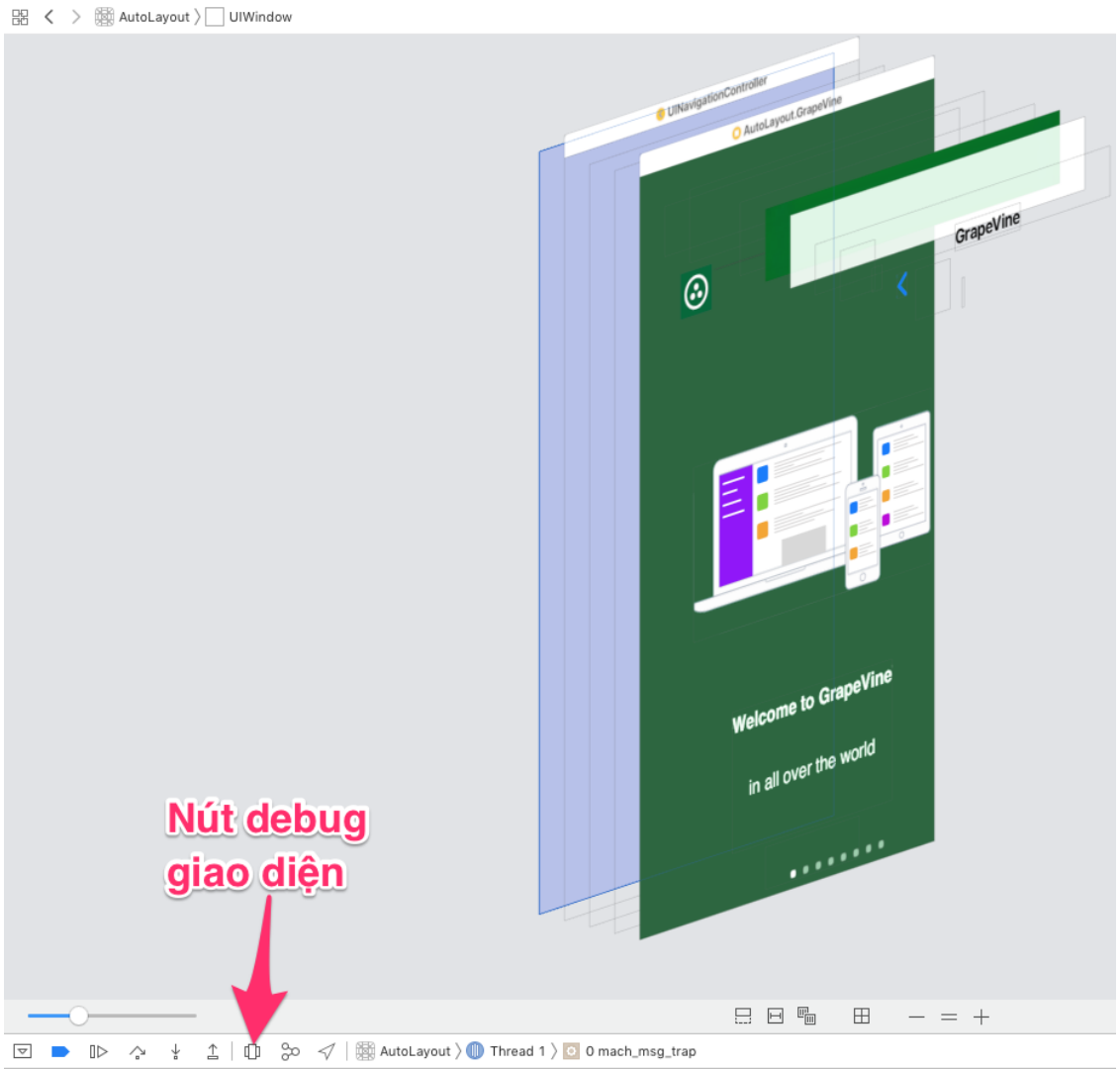


- Cả Stevia và Apple Autolayout để không có cú pháp trực tiếp để tạo ràng buộc tỷ lệ khoảng cách (ration between spacers)
- Hướng xử lý là tạo UIView ảo làm bộ đệm khoảng cách. UIView ảo không hiển thị mà chỉ làm nhiệm vụ giữa khoảng cách.

```
view.sv(
  red_rect1,
  spacer1,
  red_rect2,
  spacer2,
  red_rect3 )

let margin: CGFloat = 10.0

view.layout(
  |-(16)-red_rect1-(16)-| ~ 80,
  spacer1.width(0),
  |-(16)-red_rect2-(16)-| ~ 80,
  spacer2.width(0),
  |-(16)-red_rect3-(16)-| ~ 80
)

red_rect1.Top == view.safeAreaLayoutGuide.Top + margin
red_rect3.Bottom == view.safeAreaLayoutGuide.Bottom - margin
spacer1.Height == spacer2.Height * 2.0
```

Kỹ thuật tìm lỗi khi layout


Carrier
11:23 AM
GrapeVine
Skip
Welcome to GrapeVine
in all over the world


AutoLayout
UIWindow
UINavigationController
AutoLayout.GrapeVine
GrapeVine
Welcome to GrapeVine
in all over the world
Nút debug
giao diện
AutoLayout
Thread 1
0 mach_msg_trap

Một UIView biến mất vì

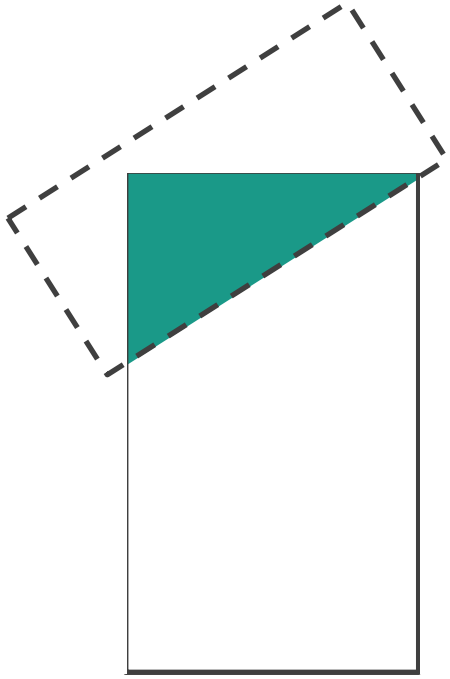
- Hoặc width hoặc height bằng 0
- backgroundColor chưa gán màu , transparent color
- alpha = 0.0
- bị một UIView khác đè lên

Tại sao bài giảng này chỉ dạy những layout rất đơn giản, cục mịch?

Giao diện tinh tế bắt đầu từ layout giao diện căn bản đúng cách

- Chia giao diện thành những khối chức năng độc lập hoặc có thể tái sử dụng.

- Dùng CALayer để vẽ những hình phức tạp tròn, elipse. Dùng kỹ thuật rotate để tạo góc chéo vát



Bắt đầu với layout tĩnh nhưng luôn kết thúc với transition để giao diện trở nên sống động, dẫn dắt người dung hơn

https://github.com/HeroTransitions/Hero