CD 24-AUTO LAYOUT - Afternoon

Documentation could be in the form of screen recording or screenshots (step by step)

Constraint

Group 3

Definition:

Auto Layout dynamically calculates the size and position of all the views in your view hierarchy, based on constraints placed on those views. For example, you can constrain a button so that it is horizontally centered with an Image view and so that the button's top edge always remains 8 points below the image's bottom. If the image view's size or position changes, the button's position automatically adjusts to match.

Constraint is setting a boundaries or restriction for positioning an object

https://youtu.be/m_0_XQEfrGQ

https://youtu.be/m_0_XQEfrGQ

https://youtu.be/emojd8GFB0o

https://youtu.be/emojd8GFB0o

https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/ProgrammaticallyCreatingConstraints.html

Documentation:

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

---

Group 4

Definition:

Auto Layout dynamically calculates the size and position of all the views in your view hierarchy, based on constraints placed on those views. For example, you can constrain a button so that it is horizontally centered with an Image view and so that the button's top edge always remains 8 points below the image's bottom. If the image view's size or position changes, the button's position automatically adjusts to match.

How to use:

There are three main options for setting up Auto Layout constraints in Interface Builder: You can control-drag between views, you can use the Pin and Align tools, and you can let Interface Builder set up the constraints for you and then edit or modify the results.

Documentation:

- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/WorkingwithConstraintsinInterfaceBuidler.html
- https://developer.apple.com/documentation/uikit/uiview/positioning_content_within_layout_margins
- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/

Group 13

Definition:

Constraint is a limitation or restriction.

How to use:

bisa pakai programmatically atau pakai interface builder

Documentation:

https://hackernoon.com/understanding-auto-layout-in-xcode-9-2719710f0706

Understanding Auto Layout in Xcode 9 - Hacker Noon • hackernoon.com

https://www.twilio.com/blog/2018/05/xcode-auto-layout-swift-ios.html

Working with Xcode Auto Layout in Swift and iOS Projects • www.twilio.com

https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/
https://developer.apple.com/documentation/uikit/nslayoutconstraint

---

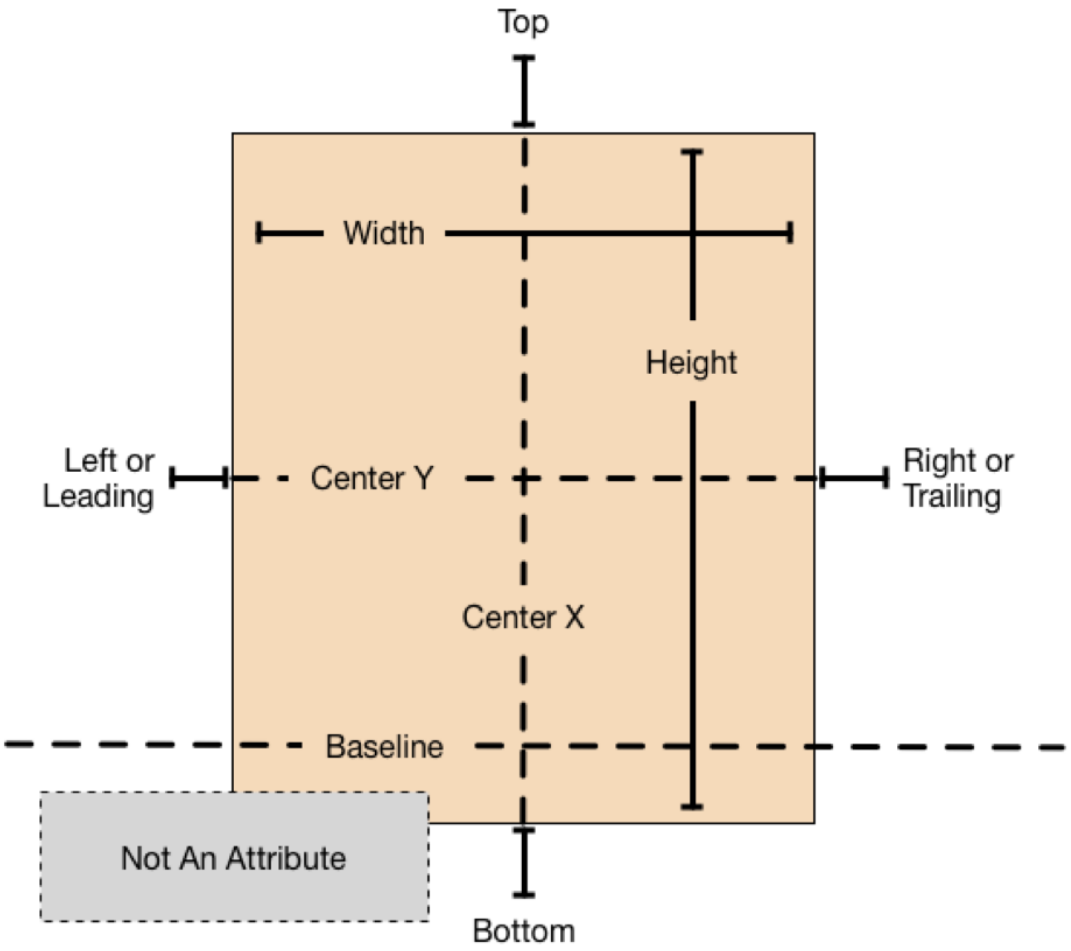
Group 14
Definition:
Auto Layout defines your user interface using a series of constraints. Constraints typically represent a relationship between two views. Auto Layout then calculates the size and location of each view based on these constraints. This produces layouts that dynamically respond to both internal and external changes.
How to use:
interface builter

https://youtu.be/7iT9fueKCJM

https://youtu.be/7iT9fueKCJM
Documentation:
https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/

Trailing & Leading

Group 5
Definition:
The values increase as you move towards the trailing edge. For a left-to-right layout directions, the values increase as you move to the right. For a right-to-left layout direction, the values increase as you move left.



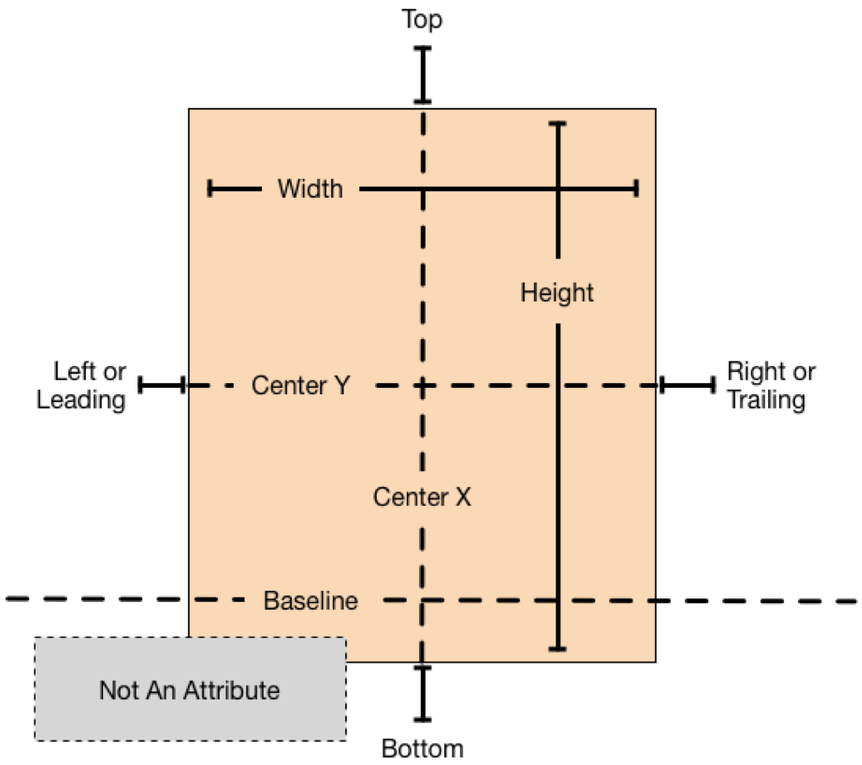
How to use:

- https://www.youtube.com/watch?v=I1Z3vvSad8Y

Documentation:
https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/AnatomyofaConstraint.html

---

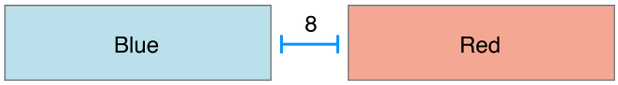
Group 6
Definition:
Leading is the LEFT EDGE of CONSTRAINED ATTRIBUTE in Auto Layout
Trailing is the RIGHT EDGE of CONSTRAINTED ATTRIBUTE in Auto Layout

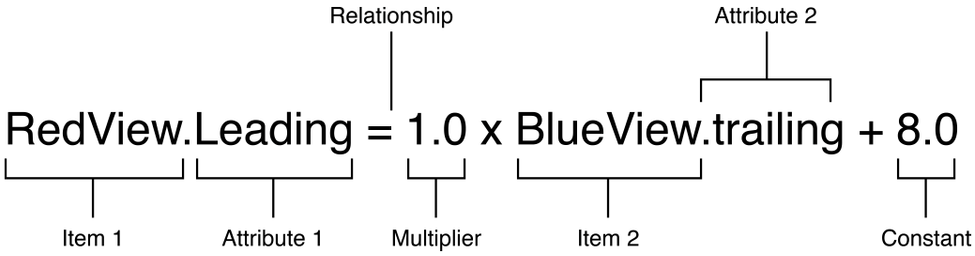


The values increase as you move towards the trailing edge. For a left-to-right layout directions, the values increase as you move to the right. For a right-to-left layout direction, the values increase as you move left. (Developer Apple)

I think the question you're asking yourself is why the heck Apple is not just naming it left and right. What's with the trailing and leading.
The reason behind this is that there might be 2 different layouts. Starting with iOS 9, the UI layout for left-to-right languages (like English) is.. well left-to-right. But in case of Arabic for example, it's right-to-left.
However Autolayout is smart enough that you don't need to setup your layout twice for these 2 types of layouts. You just set it up once and the system auto-inverts it in case your app supports right-to-left languages. (Serghei Catraniuc)

How to use:





Documentation:
https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/AnatomyofaConstraint.html#//apple_ref/doc/uid/TP40010853-CH9-SW1

https://stackoverflow.com/users/1103592/serghei-catraniuc

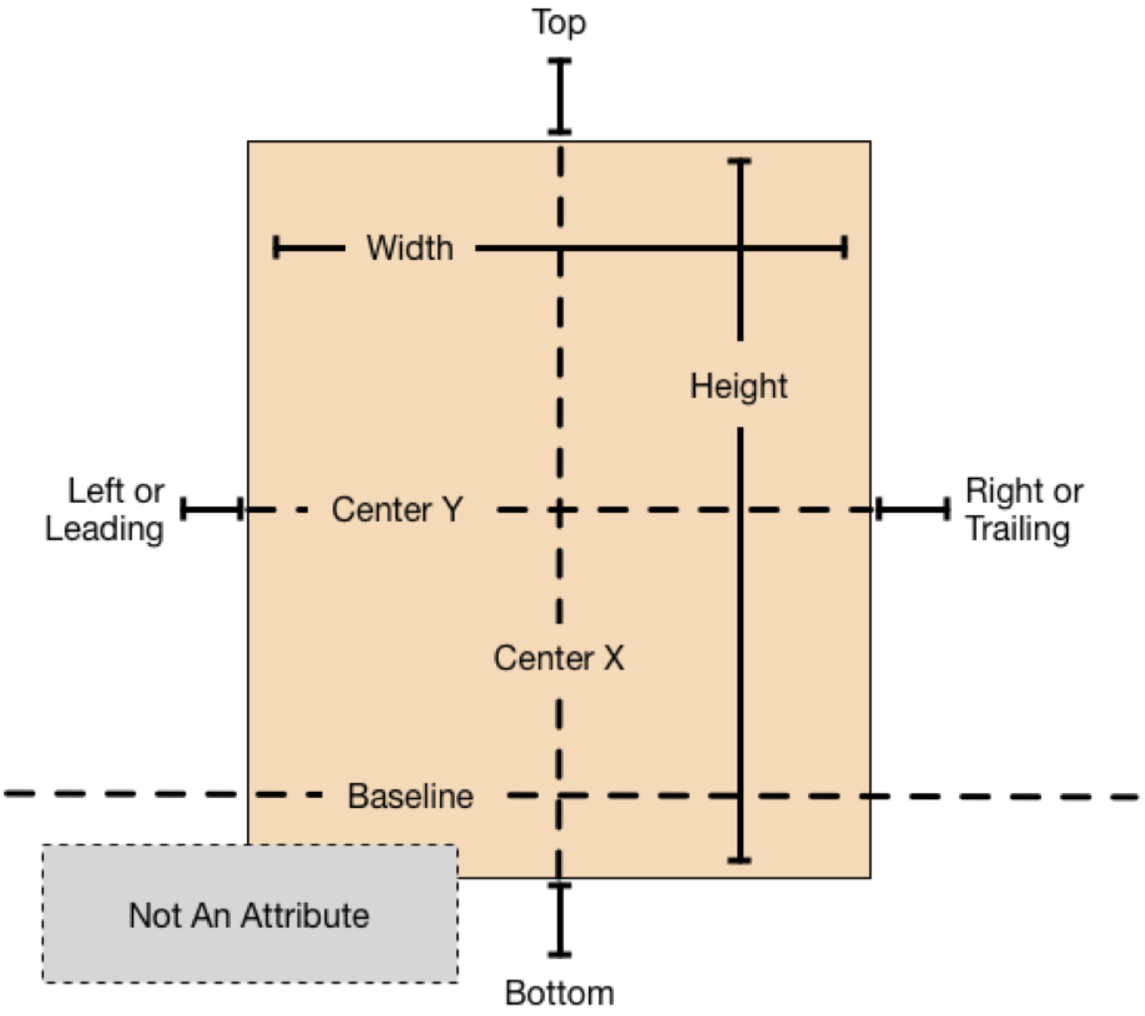
User Serghei Catraniuc • stackoverflow.com

Group 15

Trailing & Leading

Definition:
How to use:
Documentation:

Group 16
Definition:



How to use:

Documentation:

https://youtu.be/NzYeoiODoeQ

https://youtu.be/NzYeoiODoeQ

https://youtu.be/WwFBoekYNyQ

https://youtu.be/WwFBoekYNyQ

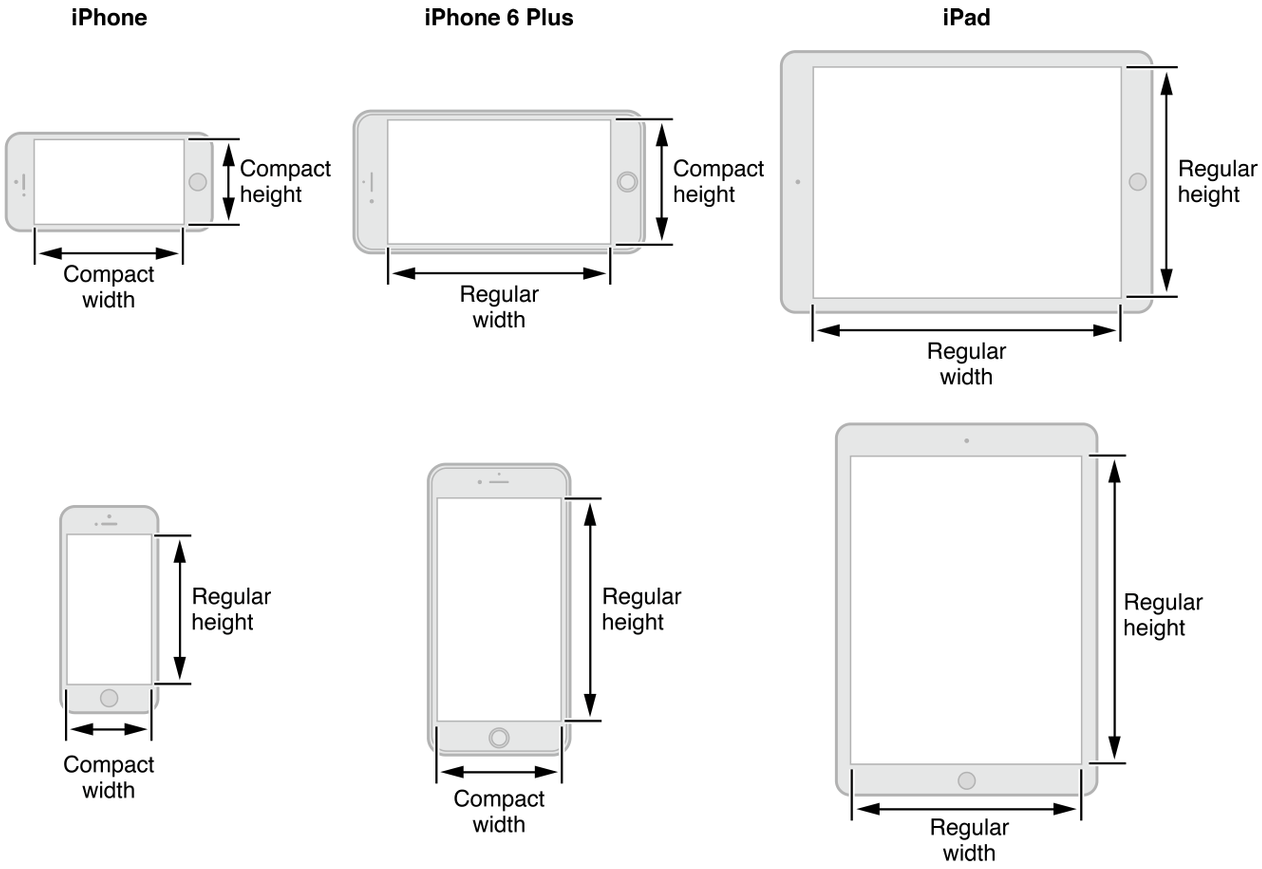
Size Class

Group 7
Definition:

Size classes are traits that are automatically assigned to content areas based on their size. The system defines two size classes, *regular* (denotes expansive space) and *compact*(denotes constrained space), which describe the height and width of a view.

A view may possess any combination of size classes:

- Regular width, regular height
- Compact width, compact height
- Regular width, compact height
- Compact width, regular height



In iOS, Size Classes are groups of screen sizes that are applied to the width and height of the device screen. The two Size Classes that exist currently are Compact and Regular.
The *Compact Size Class* refers to a constrained space. It is denoted in Xcode as wC (Compact width) and hC (Compact height).
The *Regular Size Class* refers to a non-constrained space. It is denoted in Xcode as wR (Regular width) and hR (Regular height).

Semakin kecil semakin compact semakin besar semakin regular
How to use:

https://youtu.be/Uc8IJz6zsME

https://youtu.be/Uc8IJz6zsME
Documentation:

https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/Size-ClassSpecificLayout.html

---

Group 8

Definition: https://www.hackingwithswift.com/example-code/uikit/what-are-size-classes

Size Classes are the iOS method of creating adaptable layouts that look great on all sizes and orientations of iPhone and iPad. For example, you might want to say that your UI looks mostly the same in portrait and landscape, but on landscape some extra information is visible. You could do this in code by checking for a change in the size of your view controller and trying to figure out what it means, but that's a huge waste of time – particularly now that iPad has multiple different sizes thanks to multitasking in iOS 9.

With Size Classes, you don't think about orientation or even device size. You care about whether you are running in a compact size or regular size, and iOS takes care of mapping that to various device sizes and orientations. iOS will also tell you when your size class changes so you can update your UI.

Perhaps you may want:

- a bigger font size in the huge iPad Pro screen than on the tiny iPhone SE.
- a view to be laid out differently on iPhones when in landscape or portrait mode.
- to provide additional buttons in the iPad version of your app.
- to layout content differently when your app is in slide over or split view modes.

How to use:

https://youtu.be/VLh0RIQmCr4

https://youtu.be/VLh0RIQmCr4

Documentation:
https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/Size-ClassSpecificLayout.html

Group 17
Definition:
How to use:

Documentation: https://developer.apple.com/videos/play/wwdc2017/812/?time=28

Orientation

Group 9
Definition:
How to use:

- The `orientation` property uses these constants to identify the device orientation. These constants identify the physical orientation of the device and are not tied to the orientation of your application's user interface.
- The value of the property is a constant that indicates the current orientation of the device. This value represents the physical orientation of the device and may be different from the current orientation of your application's user interface.

Documentation:
https://developer.apple.com/documentation/uikit/uidevice/1620053-orientation

---

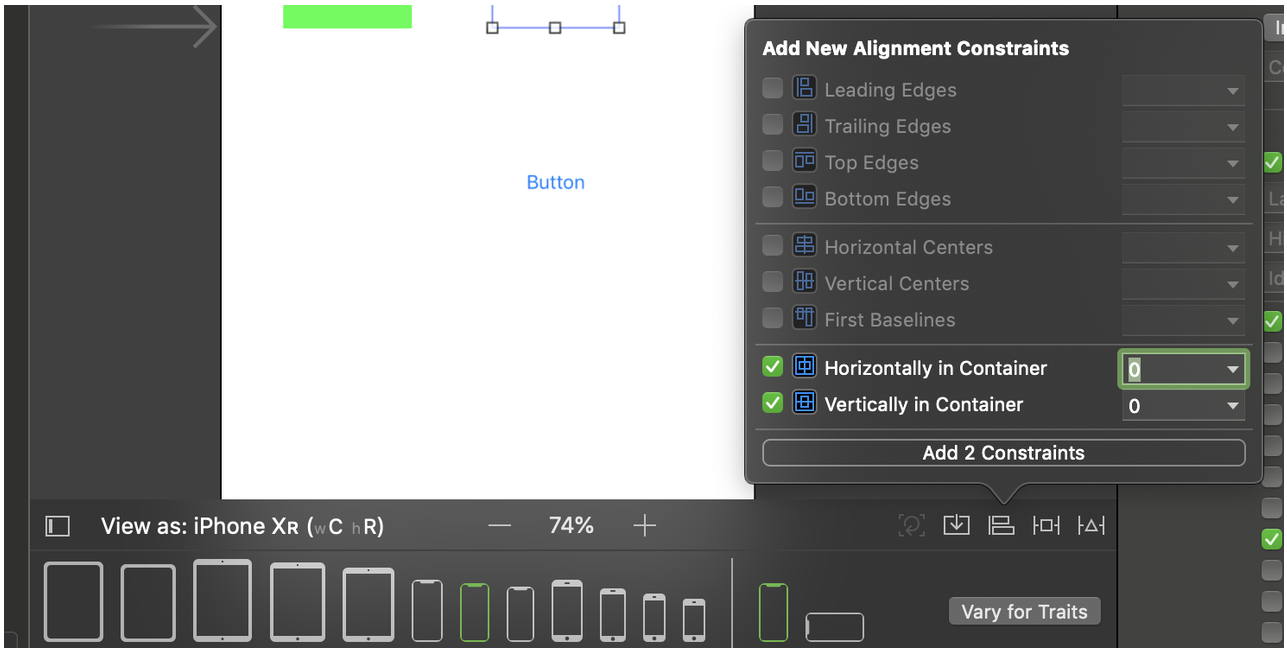
Group 10
Definition:
The `orientation` property uses these constants to identify the device orientation. These
constants identify the physical orientation of the device and are not tied to the orientation of
your application's user interface.

https://www.twilio.com/blog/2018/05/xcode-auto-layout-swift-ios.html

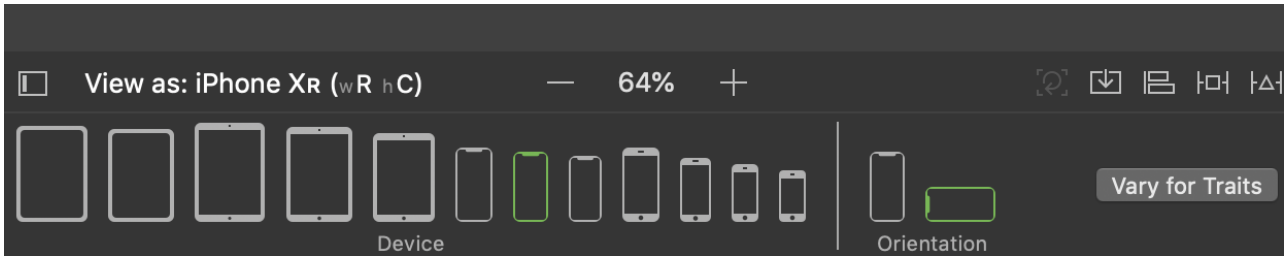
Working with Xcode Auto Layout in Swift and iOS Projects • www.twilio.com

How to use:

1. Choose/ Click your object
2. Click “Align” icon, set Horizontally/Vertically in Container (0,0) → center



3.Check your object by changing orientation device:



https://agilewarrior.wordpress.com/2017/01/26/how-to-handle-portrait-and-landscape-in-autolayout/

How to handle portrait and landscape in autolayout • agilewarrior.wordpress.com

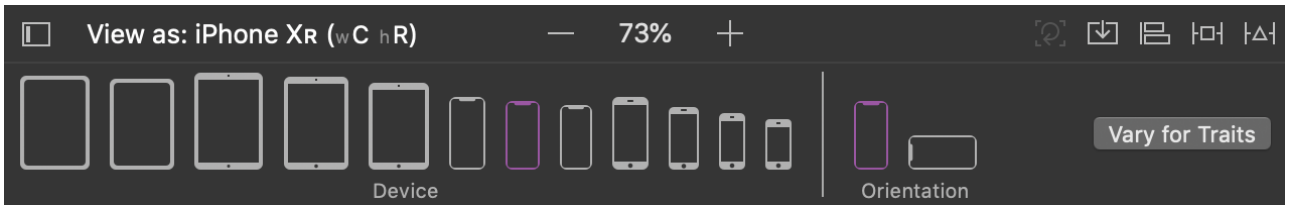
https://youtu.be/YY06LNJ1mGY

https://youtu.be/YY06LNJ1mGY

Documentation:
https://developer.apple.com/documentation/uikit/uidevice

Group 19
Definition:

- Autolayout adalah cara untuk mengatur tampilan, ukuran dan posisi suatu komponen dalam aplikasi IOS agar dapat beradaptasi dalam berbagai ukuran dan orientasi pada berbagai device.



How to use:

- https://www.raywenderlich.com/811496-auto-layout-tutorial-in-ios-getting-started
- https://agilewarrior.wordpress.com/2017/01/26/how-to-handle-portrait-and-landscape-in-autolayout/

Documentation:

-

Group 20

Definition:

- The `orientation` property uses these constants to identify the device orientation. These constants identify the physical orientation of the device and are not tied to the orientation of your application's user interface.
- The value of the property is a constant that indicates the current orientation of the device. This value represents the physical orientation of the device and

may be different from the current orientation of your application's user interface.

How to use:

Documentation:
https://developer.apple.com/documentation/uikit/uidevice/1620053-orientation

https://youtu.be/YY06LNJ1mGY

https://youtu.be/YY06LNJ1mGY