-------------------------------------------Practice – Date 9, --------------------------------June----------------

<https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html>

iOS APP Development

**Auto-Layout : -** Layout to decide exact position of any view/element so that if we change product (iphone se ,12 ,11 ,13,13 pro) the dimensions will auto adjust according to size of the display **.**

.frame : - Use krke kise bhi view ka lenth width x and y Find kr skte h or assign bhi.

.frame = CGrect() ---to decide frame layouts( viewDidAppear main )

Steps : - Always decide all Four constraints

1. Press control button and select view and drag upward then left in blank space and chose given options.

A - select center horizontal (to set center from x direction) (left to right = x)

B – select center vertical (to set center from y direction) (upside to down = y)

A + B = center all Screen or view

1. Downside of the screen there is option add constraints decide here width and height

\*\*\*Kise ek view ko refrence banakr baki views ko uske reference leke decide kr skte h \*\*\*

1. Aspect Ratio – Ratio batata h view ka ex 2:1 height and width

Ise ham same view m drag and drop krke banate h

1. Leading Space – Margin from X Axis to the view/element
2. Trailing Space – margin from Y Axis to the view/element
3. Vertical Spacing – Space between two elements (Drag first element to second and select vertical spacing .
4. Agar 2 ya use jyada constrains add kiye h to unki priority set kr de or us priority k nusar ham us constrain ka outlet banakar different-different sizes ki screen pr run kr skte h priority ko reference deke .

Ex . ----------------

ViewDidLoad main : - isme ek error ayega use fix krna h jisse row - value type ban jayega….

if (self.view.frame.height>800(pixel size) )

{. Widthconstrain1.priority = 900

Widthcontraint2.priority = 1000.

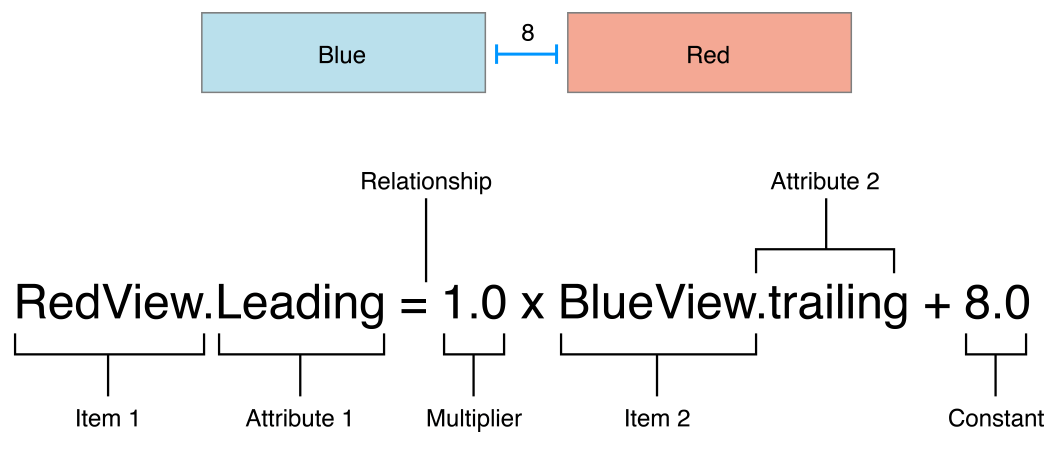
} else

{

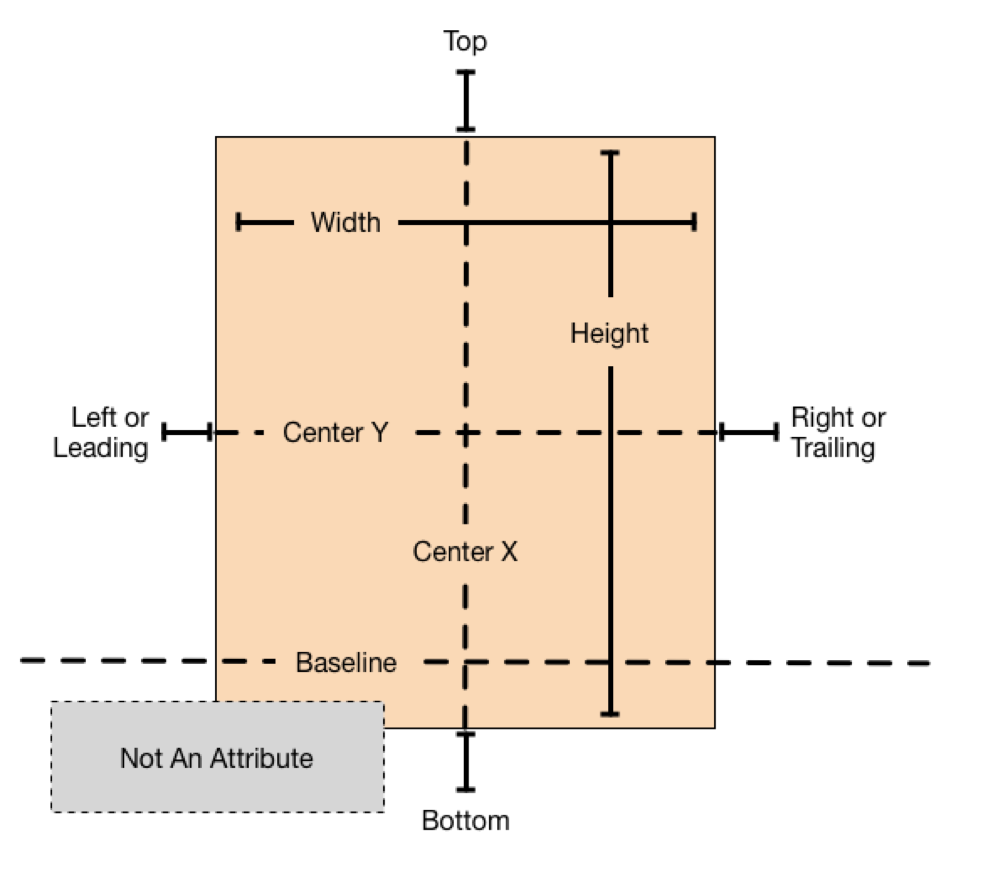
Widthconstrain1.priority = 1000

Widthcontraint2.priority = 900

}



his constraint states that the red view’s leading edge must be 8.0 points after the blue view’s trailing edge. Its equation has a number of parts:

* **Item 1**. The first item in the equation—in this case, the red view. The item must be either a view or a layout guide.
* **Attribute 1**. The attribute to be constrained on the first item—in this case, the red view’s leading edge.
* **Relationship**. The relationship between the left and right sides. The relationship can have one of three values: equal, greater than or equal, or less than or equal. In this case, the left and right side are equal.
* **Multiplier**. The value of attribute 2 is multiplied by this floating point number. In this case, the multiplier is 1.0.
* **Item 2**. The second item in the equation—in this case, the blue view. Unlike the first item, this can be left blank.
* **Attribute 2**. The attribute to be constrained on the second item—in this case, the blue view’s trailing edge. If the second item is left blank, this must be Not an Attribute.
* **Constant**. A constant, floating-point offset—in this case, 8.0. This value is added to the value of attribute 2.
* 

**Listing 3-1** Sample equations for common constraints

1. // Setting a constant height
2. View.height = 0.0 \* NotAnAttribute + 40.0
3. // Setting a fixed distance between two buttons
4. Button\_2.leading = 1.0 \* Button\_1.trailing + 8.0
5. // Aligning the leading edge of two buttons
6. Button\_1.leading = 1.0 \* Button\_2.leading + 0.0
7. // Give two buttons the same width
8. Button\_1.width = 1.0 \* Button\_2.width + 0.0
9. // Center a view in its superview
10. View.centerX = 1.0 \* Superview.centerX + 0.0
11. View.centerY = 1.0 \* Superview.centerY + 0.0
12. // Give a view a constant aspect ratio
13. View.height = 2.0 \* View.width + 0.0

**Listing 3-2** Inverted equations

1. // Setting a fixed distance between two buttons
2. Button\_1.trailing = 1.0 \* Button\_2.leading - 8.0
3. // Aligning the leading edge of two buttons
4. Button\_2.leading = 1.0 \* Button\_1.leading + 0.0
5. // Give two buttons the same width
6. Button\_2.width = 1.0 \* Button.width + 0.0
7. // Center a view in its superview
8. Superview.centerX = 1.0 \* View.centerX + 0.0
9. Superview.centerY = 1.0 \* View.centerY + 0.0
10. // Give a view a constant aspect ratio
11. View.width = 0.5 \* View.height + 0.0

## Types of Errors

Errors in Auto Layout can be divided into three main categories:

* **Unsatisfiable Layouts**. Your layout has no valid solution. For more information, see [Unsatisfiable Layouts](https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/ConflictingLayouts.html#//apple_ref/doc/uid/TP40010853-CH19-SW1).
* **Ambiguous Layouts**. Your layout has two or more possible solutions. For more information, see [Ambiguous Layouts](https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/AmbiguousLayouts.html#//apple_ref/doc/uid/TP40010853-CH18-SW1).
* **Logical Errors**. There is a bug in your layout logic. For more information, see [Logical Errors](https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/LogicalErrors.html#//apple_ref/doc/uid/TP40010853-CH20-SW1).

Most of the time, the real problem is just determining what went wrong. You added the constraints you thought you needed, but when you ran the app, things did not turn out as you had hoped.

Usually, as soon as you understand the problem, the solution is obvious. Remove conflicting constraints, add missing constraints, and adjust tied priorities so that there is a clear winner. Of course, getting to the point where you can easily understand the problem may take some trial and error. Like any skill, it gets easier with practice.

Sometimes, however, things get more complicated. That’s where the [Debugging Trickßs and Tips](https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/DebuggingTricksandTips.html#//apple_ref/doc/uid/TP40010853-CH21-SW1) chapter comes in.