1 - Frame based layout VS Auto Layout using constraint

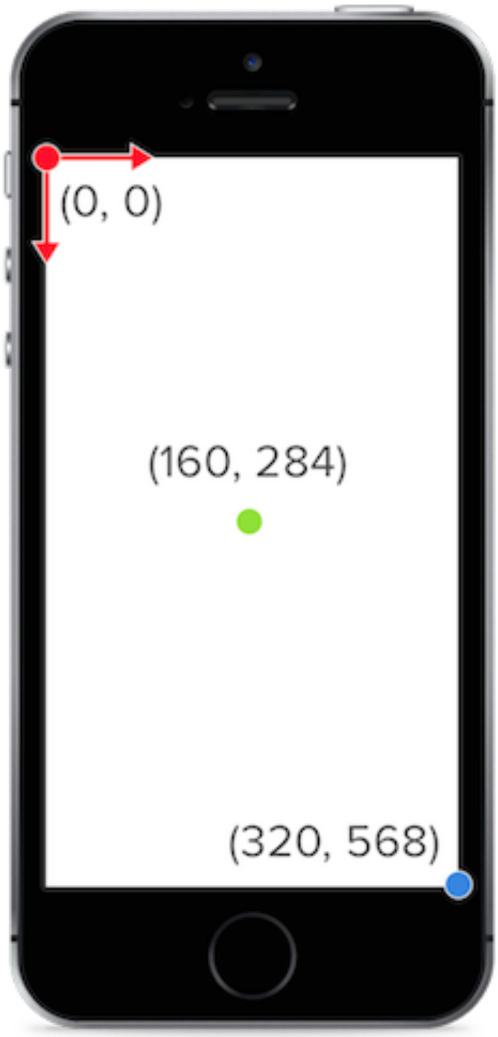
This is the first part of Making Sense of Auto Layout Series.

Before we dive into Auto Layout, lets look at how User Interface is designed prior to Apple inventing Auto Layout concept.

Frame based layout

In the early days of iOS development, developing UI for iOS app is pretty straightforward as there is only one screen size (From the first iPhone until iPhone 4s, their screen resolution is 320x480). In 2010 Apple introduced the iPad (resolution is 1024x768) , it was still manageable at this point, most developer just develop two separate UI (or app), one for iPhone and another for iPad.

In this era, frame based layout is used. Frame based layout uses a coordinate system like this :

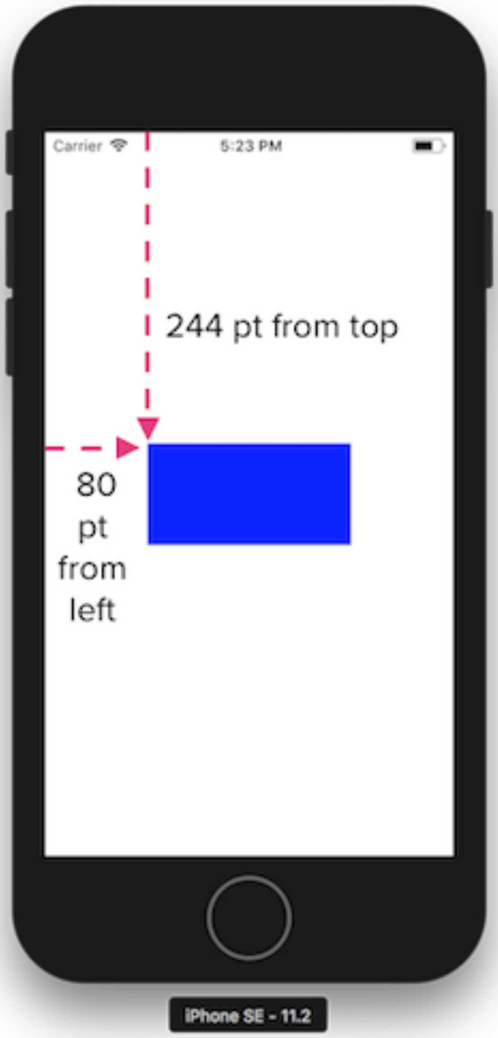


Coordinate system in iOS starts from top left and ends at bottom right. The top left coordinate is (0, 0) , which mean the horizontal (x) position is 0 and vertical (y) position is 0. For this example we used iPhone SE which have a point resolution of 320pt x 568 pt, so the bottom right coordinate is (320, 568), x position is 320 and y position is 568. The center point is (320 / 2 , 568 / 2) = (160, 284).

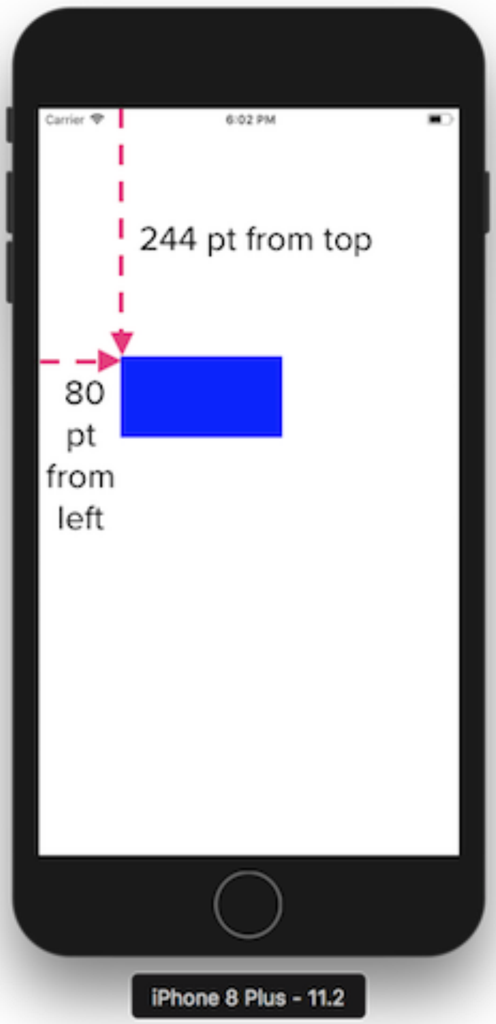
We can create a UIView and define its x, y, width and height like this :

```
let blueView = UIView(frame: CGRect(x: 80.0, y: 244.0,
width: 160.0, height: 80.0))
blueView.backgroundColor = UIColor.blue
self.view.addSubview(blueView)
```

This will put a blue rectangle in X position of 80 (from left), Y position of 244 (from top) and a width of 160 and height of 80. Like this :



I have pre-calculated the position so that the rectangle is placed in the center of the view (horizontally and vertically). It is centered nicely in iPhone SE, but when you view it on iPhone 8 Plus, it is positioned upper left from the center because the iPhone 8 Plus screen width and height is larger than iPhone SE, like this :



To accommodate for iPhone 8 plus screen, you have to adjust the x,y position of the view to get it to the center of the screen. Without using Auto Layout constraint, the code might look like this :

```
// Don't do this! This serve as a bad example
if(screenSize == iPhoneSE){
    blueView.frame = CGRect(x: 80.0, y: 244.0, width: 
160.0, height: 80.0)
} else if (screenSize == iPhone8){
    blueView.frame = CGRect(x: 100.0, y: 284.0, width: 
160.0, height: 80.0)
} else if (screenSize == iPhone8Plus){
....
} // Apple suddenly launch iPhone X
else if (screenSize == iPhoneX){
// ffuuuu...
}
// what if user rotate the screen?!
```

To solve this problem, Apple has introduced Auto Layout with constraint based approach so you wouldn't have to purposely accommodate each screen size differently.

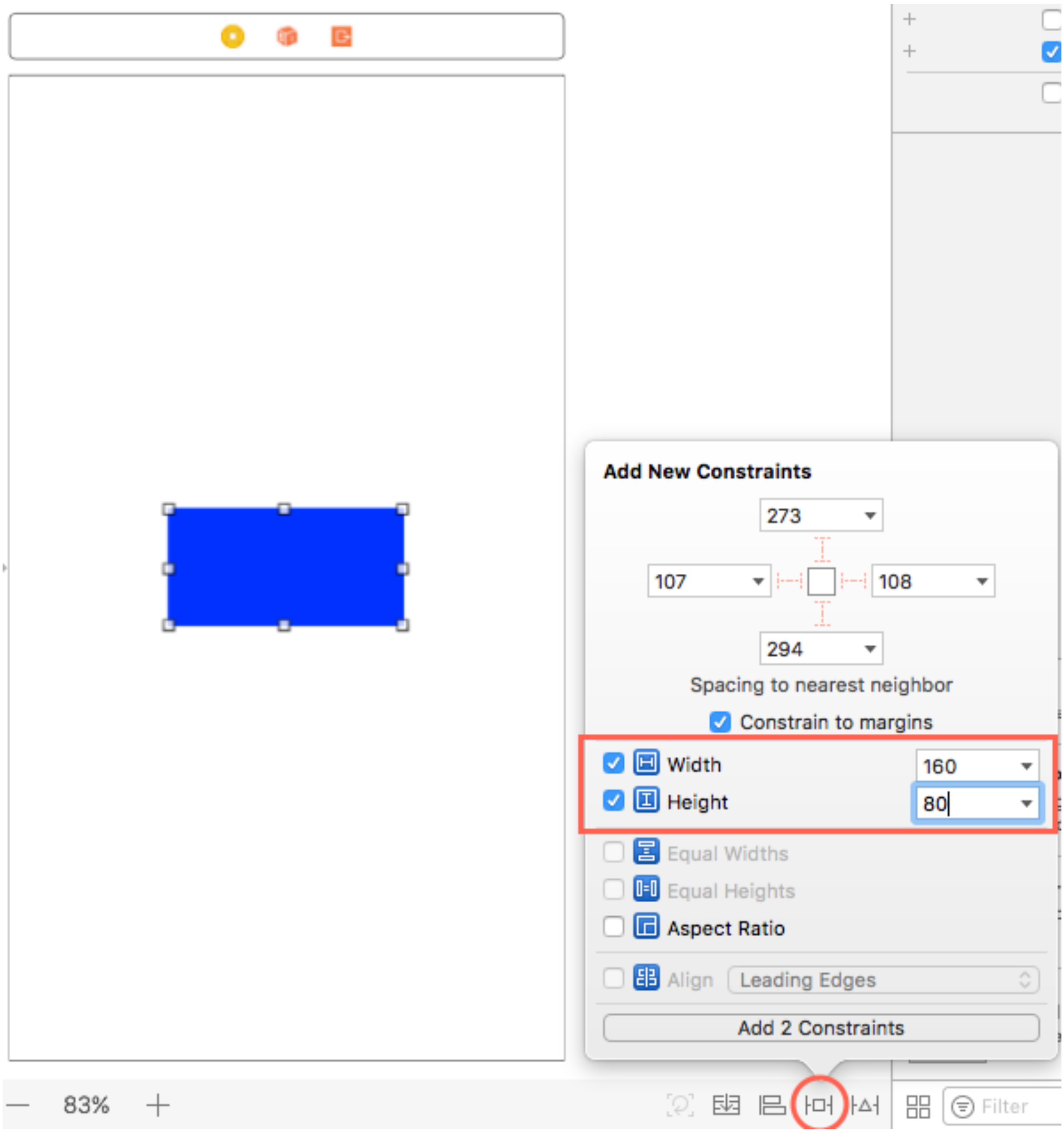
Auto Layout using Constraint

Things start to get complicated after iPhone 5 is released, then iPhone 6 and 6 plus... and then iPhone X , its certainly possible to design multiple UI layout to fit each of them, but its a pain in the ass to do so. In my previous work, I inherited a project that has a storyboard for each of the iphone screen size (iPhone4s.storyboard , iPhone5.storyboard, iPhone6.storyboard and iPhone6plus.storyboard) , it was a nightmare (kill me) and I spent days to refactor them to use Auto Layout.

Auto Layout is a concept which the size and position of all the views in the view hierarchy are dynamically calculated on run time, based on constraints placed on those views.

For easier understanding, let's call the thing that will dynamically calculates the size and position of all the views as **Auto Layout Engine**.

Following the previous example shown in frame based layout, to position the blue rectangle to center horizontally and vertically using Auto Layout, we add constraints to the blue view in Interface Builder / Storyboard like this :



Add New Constraints

273

107 108

294

Spacing to nearest neighbor

☑ Constrain to margins

☑ Width 160

☑ Height 80

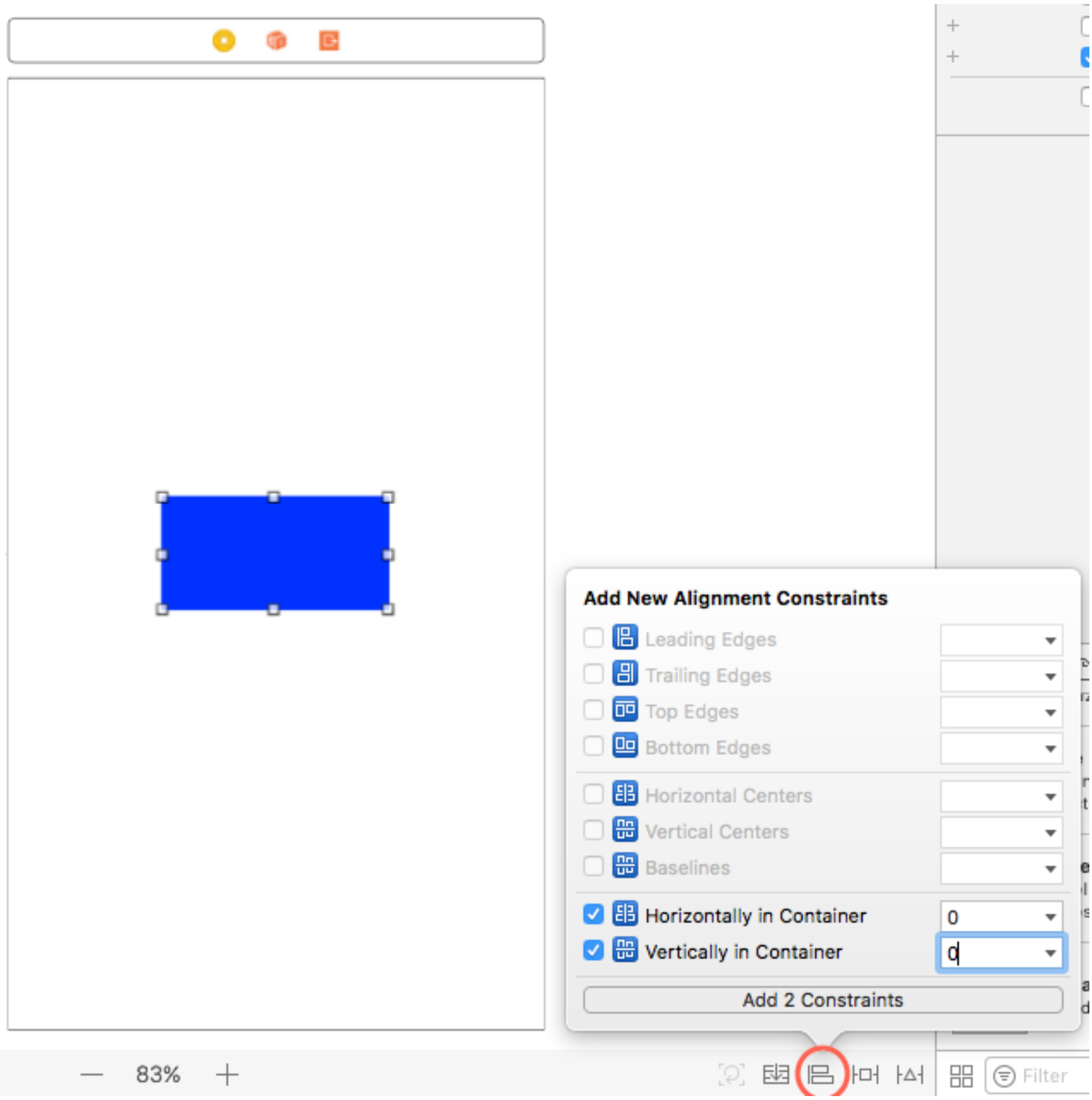
□ Equal Widths

□ Equal Heights

□ Aspect Ratio

□ Align Leading Edges

Add 2 Constraints

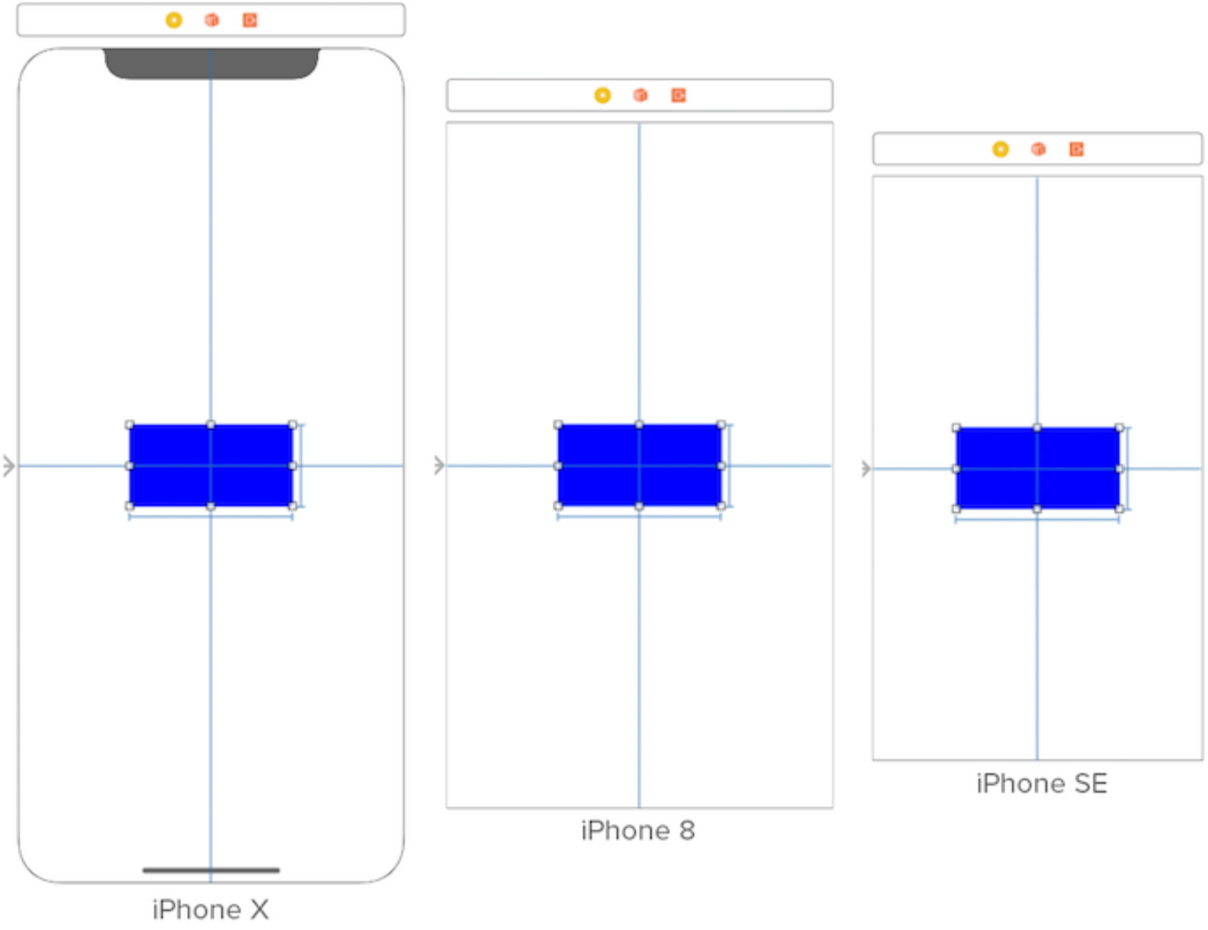


Add New Alignment Constraints

- □ Leading Edges
- □ Trailing Edges
- □ Top Edges
- □ Bottom Edges
- □ Horizontal Centers
- □ Vertical Centers
- □ Baselines
- ☑ Horizontally in Container 0
- ☑ Vertically in Container 0

Add 2 Constraints

This will set the rectangle size (width and height) and also center it both horizontally and vertically to the screen. The constraint works across multiple devices seamlessly like this :



When your app is launched in iPhone X, Auto Layout Engine know that the point resolution of iPhone X is (375x812), then it will calculate

horizontal center point = 375 / 2 = 187.5

vertical center point = 812 / 2 = 406

To position the blue view nicely in the center, Auto Layout Engine will use the width and height of the blue view to calculate what position it should place the blue view on, its calculation formula might look like this :

```
screenCenterX = 187.5
screenCenterY = 406
// x position of blue view, left half of the blue view
is on the left of the x center point
x = screenCenterX - (width of blue view)/2.0
// y position of blue view, top half of the blue view
is
on top of the y center point
y = screenCenterY - (height of blue view)/2.0
```

Now when you launch your app in iPhone 8, Auto Layout Engine will use the point resolution of iPhone 8 (375 x 667) and calculate the center points and position of the blue view again using the formula above.

Since Auto Layout Engine always calculate the position of the view before rendering it on screen, you can rest assured that the layout will work nicely if you define the all the constraint correctly.

Other than different phone size, Auto Layout will also recalculate position and size of a view when the followings happen :

1. Device is rotated
2. Split screen view is enabled in iPad
3. The active call red top bar appears on iPhone

Imagine manually writing code to handle all of above situation, it must be hell lot of work 😱. Auto Layout allows us to save a lot of time on designing UI.

We will look into how Auto Layout Engine work in detail in the next chapter.