A picture containing text, appliance

Description automatically generated**Auto Layout**

To position an element X number of points away from the top of bottom of the screen, you must **control + click** it and drag it to the side you want to add a constraint, you can then **click Bottom space to safe area** to constraint the bottom space.

In the constraint inspector, you can modify the **Constant** property so it says the following: "make it at least 20, but you can make it more to fill up space."

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, application

Description automatically generatedIf we need to squeeze multiple elements, we can do the next two steps: 1. **Control + click** each one of them **with the other** and **click Equal Heights,** which will maintain all the buttons equal height. 2. Then, to preserve aspect, **control + click** each one of them with **themselves** and **click Aspect Ratio.**

**Visual Format Language**

On the next line of code

view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat: "H:|[\(label)]|", options: [], metrics: **nil**, views: viewsDictionary)) the tricky parameter is the first one, the V means vertical, while the pipes (|) mean “the edge of the screen” and the [ ] means “the edge of the view”. So that means, "horizontally, I want my label1 to go edge to edge in my view." The parameter views requires a dictionary to know where to map the first parameter.

We can do the same for

view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat: "V:|[label1]-[label2]-[label3]-[label4]-[label5]", options: [], metrics: **nil**, views: viewsDictionary)) which adds vertical constraints, the - means a spacing, which is 10 points by default but can be customized. At the end we did not put a | so we don’t force the labels to stretch all the way down, we are rather leaving a space. We could add a height to each label by [label1(==88)] to specify its height. And add a final space to the page, that is greater or equal than 10 by adding this at the end (>=10)-|.

But for specifying sizes, it is better to use the metrics parameter and adjust it as [label1(labelHeight)], you can also make labels height relative to other labels + adjustable, to do this 1st specify the priority as (labelHeight@999) where 999 means not required BUT really really wanted, and then use label1 to say you want the same heights as other labels [label2(label1)].

**Auto Layout Anchors**

Every **UIView** has a set of anchors that define its layouts rules. The most important ones are **widthAnchor**, **heightAnchor**, **topAnchor**, **bottomAnchor**, **leftAnchor**, **rightAnchor**, **leadingAnchor**, **trailingAnchor**, **centerXAnchor**, and **centerYAnchor**. Note that we would use **leftAnchor** and **rightAnchor** if we want our app to always look the same, but we use **leadingAnchor** and **trailingAnchor** when we want our app to look different on idioms like Hebrew and so.

[Auto layout cheat sheet](mailto:https://www.hackingwithswift.com/articles/140/the-auto-layout-cheat-sheet)