**CSCI E65g: Autolayout**

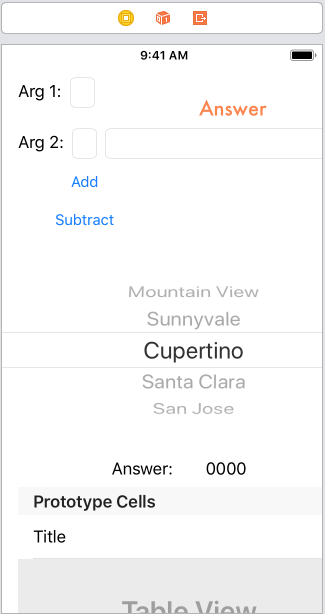
**Fall 2018**

**Autolayout: Why Two Stages?**

There are TWO distinct stages to doing layout.

The first is what I call, informally, the “paste-up” stage. Where you just move things around until they look right, using the dotted guides as optional snap-to alignments.

This should be thought of as unrelated to anything that happens at run-time. It’s just experimental. XCode actually lets you leave it this way, with no errors, but this is very misleading and unfortunate. If an object has no constraints, it will be at an absolute location. This is hopelessly fragile when you consider relationships that need to be obeyed in different device sizes, or in fact anything that changes about the surrounding elements. Consider one of many unusable results when doing a design on iPhone 8 template and then running the App on an iPhone SE:



For the sake of the below, assume the width of a device is exactly 400 points to make the math easy.

The second stage is the constraining stage. This is where you make the location and dimensions of a component precisely determined as a set of constraints. This does make it into the App. Constraints, being equations with interdependent values that are not evaluated until runtime, give immeasurably more power to layouts that can adapt to changing conditions.

When an element is pasted up in Storyboard at absolute location (100, 100) extending to (300, 300), it could be that you want to express that in terms of offsets simple values:

* Left-side(table-view) = Left-side(superview + 100)
* Top-edge(table-view) = Top-edge(superview + 100)
* Width(table-view) = 200
* Height(table-view) = 200

But that’s not the only way to end up with those absolute locations. You have to start thinking about what the layout should do relative to things that might change, such as the device dimensions or orientation. Maybe what you really want is:

* A table that is horizontally centered
* And is exactly 50% of the device width
* And 100 points from the top
* And 200 points tall

So now the constraints would be:

1. Horizontal-Center(table-view) = Horizontal-Center(superview)
2. Top-edge(table-view) = Top-edge(superview + 100)
3. Width(table-view) = 0.5 \* Width(superview)
4. Height(table-view) = 200

And yet the absolute coordinates (solved constraint equations) come out exactly the same…for this particular device in this particular orientation.

So, Storyboard / Autolayout has no idea what you really mean, so you have to work in these two separate stages. And to do that, you have to decide what you mean to yourself, first!

Any errors you see are not referring to paste-up coordinates, which are always known. Instead they refer to missing or conflicting constraints. What it’s doing is a simulation of processing all the constraints. If the math results in unsolved values, you get errors. So if constraint 4 were missing, the height would remain a ‘?’ despite the paste-up size of 200 or whatever.

If you want to widen a button (any object) to have a fixed width, you have two options:

* Edit the constraint manually by typing in a larger number
* Widen it, then observe the pasted-up width in the Constraints inspector, and “make it official” by copying that width value into the constrained width. (And if there’s no width constraint, you’ll need to ctrl-drag inside the element to create one.)

There’s no purpose to adding white space inside a button since it’s not observable; instead increase the horizontal separation values for adjacent element constraints, or think of other constraints for the adjacent elements (maybe they ought to be relative to the side of the superview rather than the edges of the button).