Introducing

Adaptive Layout

Hands-on Lab

Introducing Adaptive Layout Hands-On Lab

Lab: Adaptive Image and Fonts

You've already seen some of the tools of the trade for dealing with adaptive layouts, time to level-up.

So far you've learnt how to create an adaptive layout using installable constraints and views, but these alone don't solve all your problems. For example, the font size on a compact display should be smaller than that on a regular display.

In this lab you'll learn how to adapt the font size based on the size class, and how to use the asset library to customize images based on size classes.

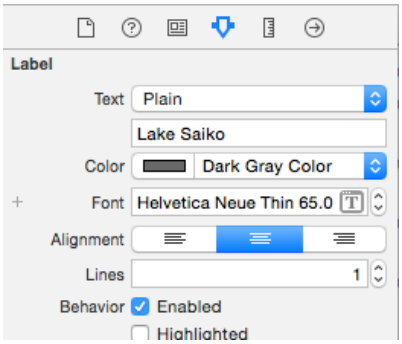
Font Scaling

Take a look at the preview of the weather view controller on an 3.5" iPhone in landscape:

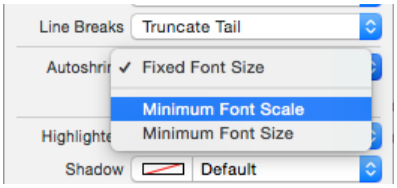
Notice that there isn't enough horizontal space for the **Lake Saiko** title, and that it is being truncated. As the title of this section suggests, one option is to use the new adaptive font functionality, but there is a simpler alternative in this situation.

Ensure that you're working in the **w Any h Any** base size class using the bottom bar:

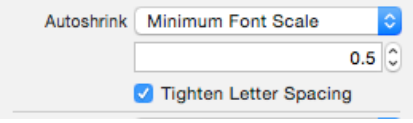
Select the **Lake Saiko** label, and open the **Attributes Inspector**:



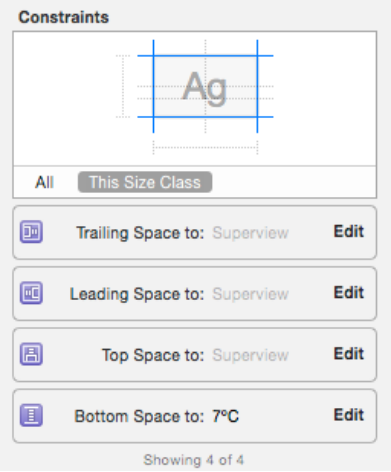
Find the **Autoshrink** section, and change the setting from **Fixed Font Size** to **Minimum Font Scale**:



Ensure that the value is set to **0.5**, and check the **Tighten Letter Spacing** box:



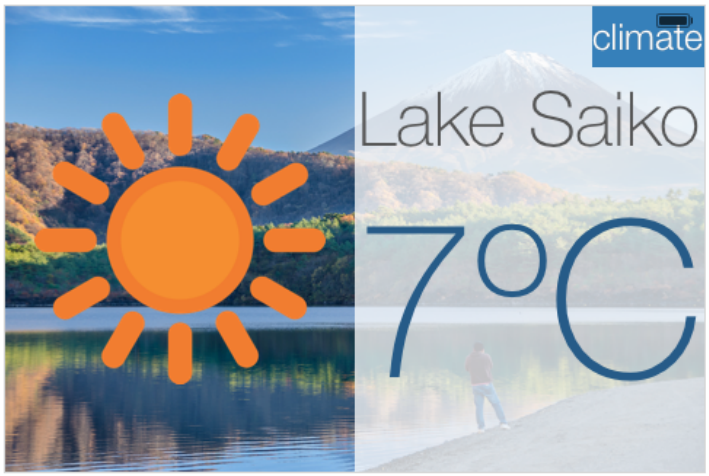
These settings mean that if the layout restricts the width of the box, then the font size will be reduced (down to a limit of half its specified size) in order to get it to fit. Note that the **Lake Saiko** label has layout constraints that do define its width:



This approach will not work if you are relying on the intrinsic size of the label to define the width.

Repeat the same process for the **7ºC** label.

Check out the preview for the 3.5" iPhone again – you can see that the title label now fits within the width available:



iPhone 3.5-inch

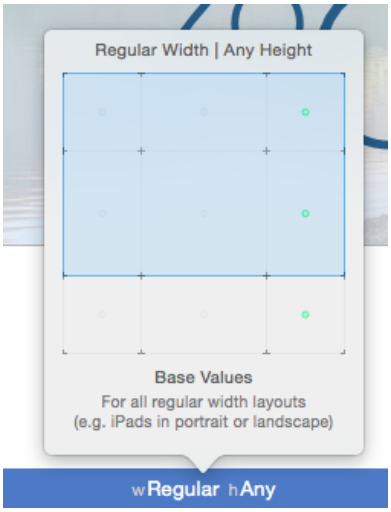
Adaptive Fonts

Rescaling fonts is a simple solution and should probably be your first port of call. However, it's not very powerful. It allows you to *shrink* fonts, but not make them larger, or change them entirely. It's also worth noting that shrinking a font will not affect the height of the label – i.e. the layout won't update and this can result in strange spacing.

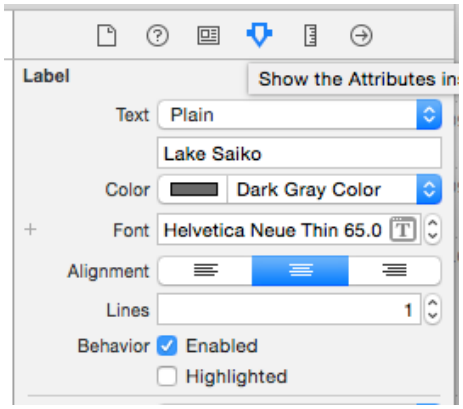
Adaptive fonts are a much more powerful tool, and allow you to define specified font overrides for particular size classes.

In this section you will update the layout to change the font for the regular width size class, i.e. when there is more space then a larger, bolder font will be more appropriate.

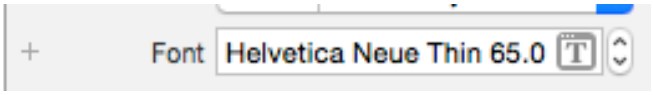
Use the size class selector on the bottom bar to enter the **Regular Width | Any Height** size class override:



Select the **Lake Saiko** label, and open the **Attributes Inspector**:

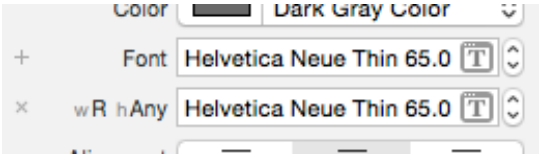


Click the **+** to the left hand side of the **Font** field, to add a size class override:

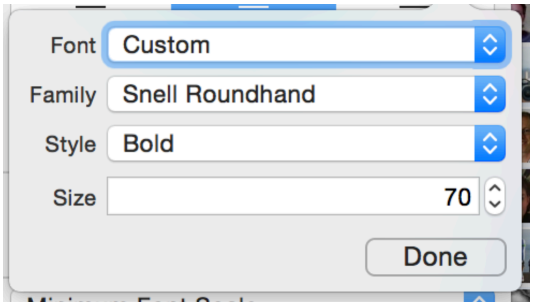


This displays a menu that allows you to choose any size class override, with the current one highlighted. Select the current size class – **Regular Width | Any Height**:

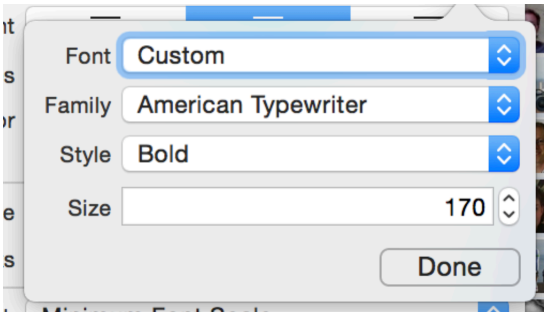
This adds a new **Font** field for the specified size class override:



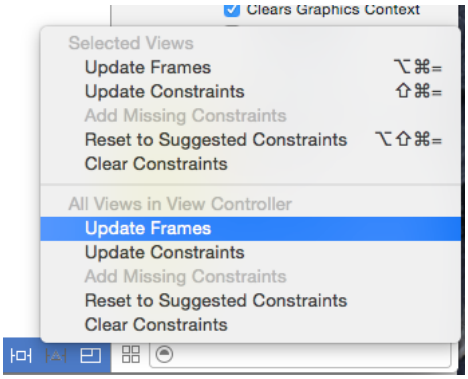
Use the font dialog to set the font to **Snell Roundhand Bold** with a size of **70**:



Repeat exactly the same process for the **7ºC** label, this time setting the font to **American Typewriter Bold** with a size of **170**:



Since the intrinsic size of the labels has now changed, there are constraints which are no longer satisfied by the current layout of the canvas. As previously, use the **Resolve Auto Layout Issues** menu to update the frames of all the views in the view controller:



Check out the results in the preview assistant editor. You'll notice that the font changes on the 5.5" iPhone (6 Plus) depending on its orientation, and that the new font is present on both orientations of the iPad:

ıe 5.5-inch

iPhone 5.5-inch

iPad

Pretty cool, huh? It maybe isn't the best typographical design you've ever seen, but you get the idea of what's possible. This technique gives you an immense amount of control over the different layouts, whilst remaining in the universal storyboard.

Adaptive Images

The final piece of the adaptive layout collection comes in the shape of adaptive images. This means being able to switch out different images depending on the

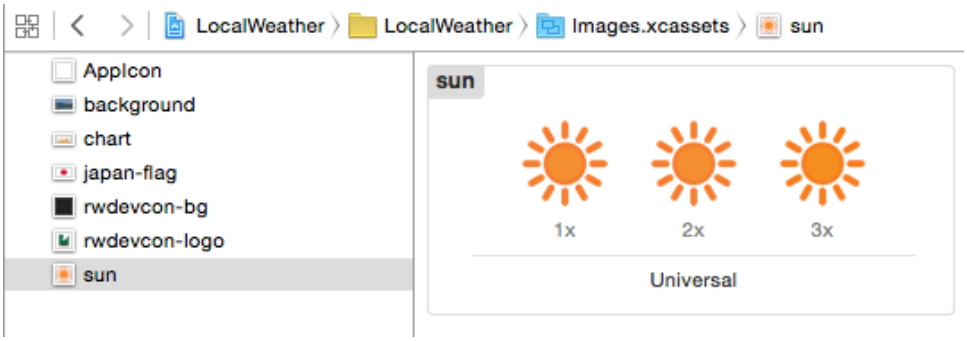
current size class. For example, you might want to show less-detailed icons for compact size classes than for regular ones. In fact, that's exactly what you're going to do in this section of the lab.

The sun weather icon that is currently in the LocalWeather app was designed for size classes of regular width, and you've been provided with another for compact width:

This icon is less detailed, and looks better than the existing one on small screens. You can find the three required versions of this icon with the materials supplied with the starter project, in a directory called **3-lab-images**.

Asset catalogs have built-in support for adaptive images, so you'll use that to implement this feature.

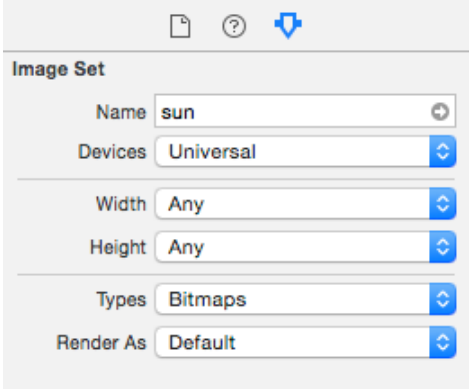
Open **Images.xcassets**, and select the **sun** image set:



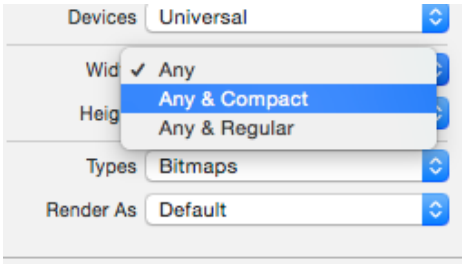
You can see the three existing versions of the sun image.

> **Note:** If you haven't seen an asset catalog since iOS8 you might be wondering why there are 3 different versions:
> * **1x** represents the non-retina version. Used on iPad 2.
> * **2x** represents the retina version. Used on every device apart from iPhone 6 Plus
> * **3x** represents the retina HD version. Used on the iPhone 6 Plus (rescaled).

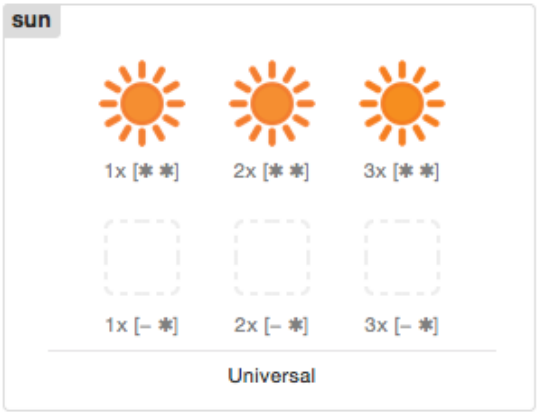
Open the **Attributes Inspector** to reveal details about the image set:



The **Width** and **Height** fields allow you to specify whether you'd like to provide overrides for the different size classes. Since you want to use the new images for compact width, select **Any & Compact** from the **Width** field:



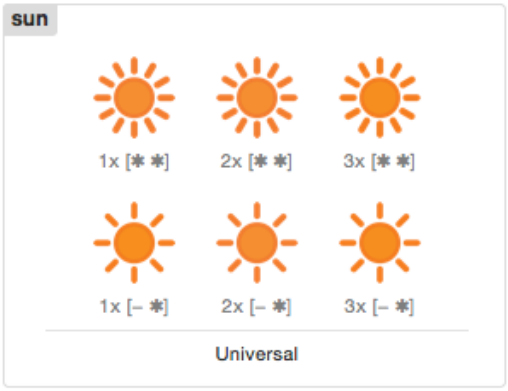
This updates the appearance of the image set in the asset catalog – creating three new, empty cells:



> **Note:** The symbols involved here are a little confusing; the horizontal and vertical size classes are represented using the follow form `[h v]`, where `h` and `v` take one of the following:
>
> * **Any** size class – i.e. not overridden
> + **Regular** size class
> - **Compact** size class

> Therefore, `[– *]` represents the compact width size class, and `[* *]` represents the base size class, as expected.

Locate the supplied source images in Finder, and then drag each of them in turn to the appropriate cell in the image set. Once you've done it will look like this:



Head back to the storyboard, and open up the preview assistant editor to see your new images in action:

You can see that the original image remains for regular width size classes, but has been switched out for the new version when the width becomes compact. Mega!