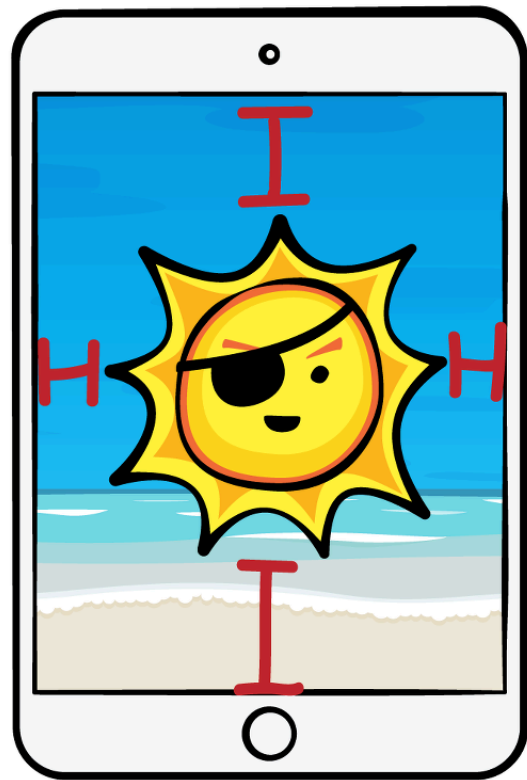


BEGINNING

AUTO LAYOUT



HANDS-ON CHALLENGES

Beginning Auto Layout

Jerry Beers

Copyright ©2016 Razeware LLC.

Notice of Rights

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written permission of the copyright owner.

Notice of Liability

This challenge and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connection with the software or the use of other dealing in the software.

Trademarks

All trademarks and registered trademarks appearing in this book are the property of their own respective owners.

Challenge #2: Autoresizing Masks

By Jerry Beers

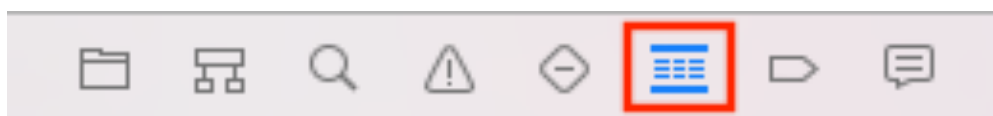
In this challenge, you'll look at one of the tools Xcode 8 provides for debugging views, **Debug View Hierarchy**, and use it to see how the autoresizing mask is reflected on a view.

Debug View Hierarchy

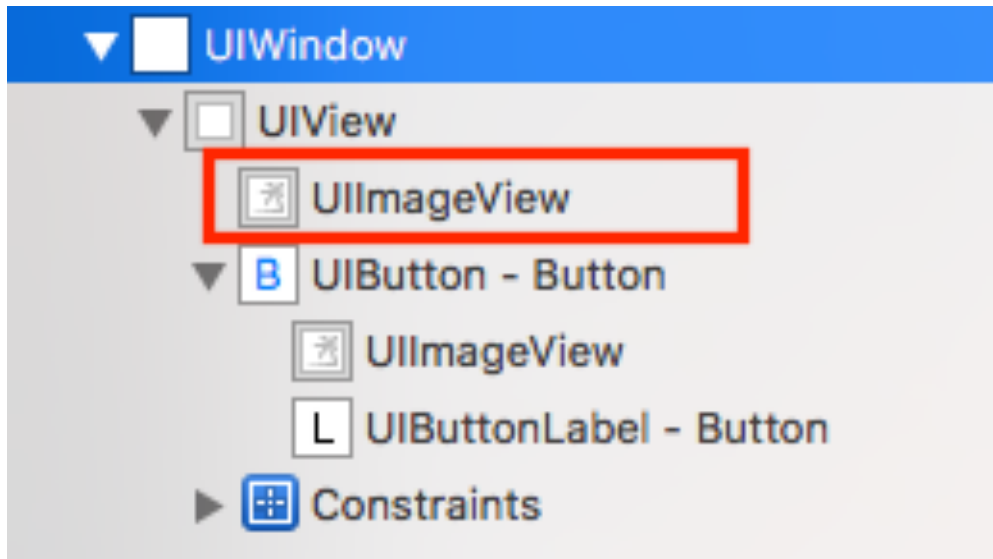
Whenever your app is running, at the bottom of the editor window in Xcode, you'll see a row of buttons:



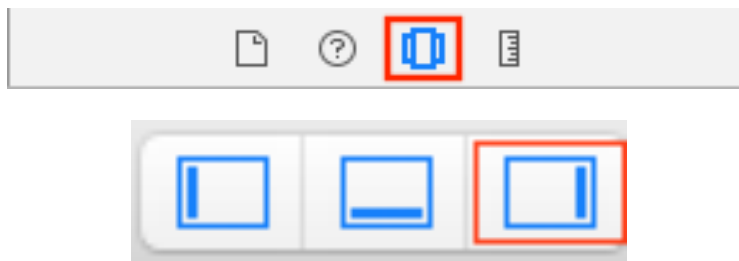
If you click on the highlighted button, your app will pause, just the same as if you had hit a breakpoint, and Xcode will show you a list of all the views in the view hierarchy. Make sure you have the Debug navigator selected in the left pane:



You should now see a list of all the views:

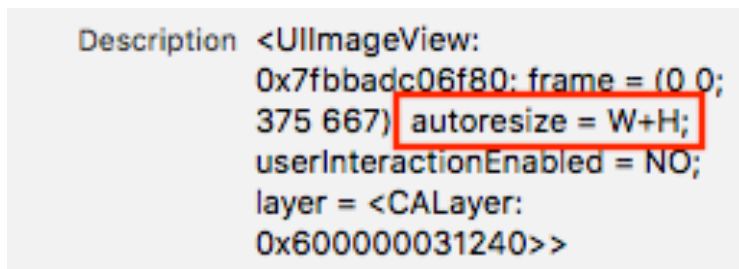


Either click on the highlighted row or anywhere on the background image in the editor window. Then, in the right pane, make sure the Object inspector is selected:



Note: If the right pane is not visible, click the Utilities button in the upper-right corner

At the bottom of the Object inspector, you should see a **Description** section. In that description is an **autoresize** string:



The description shows which elements are *flexible*, in this case the width and the height. Notice that this is sometimes different from Interface builder, where setting the margin elements indicates that they are *fixed*. Now click on the UIButton in the view hierarchy and notice its description:

```
Description <UIButton: 0x7fbbadc08510;  
frame = (144 617; 87 30);  
opaque = NO; autoresize =  
LM+RM+TM; layer = <CALayer:  
0x600000032640>>
```

This tells you that the left margin, right margin, and top margin are all flexible, but that everything else is fixed.

Once you know the autoresizing mask for a view, if `translatesAutoresizingMaskIntoConstraints` is true for that view, you can know how the framework will create its constraints.