

Customizing labels in World Trotter

Open Main. Storyboard. Create 5 UILabels, center them horizontally on top.

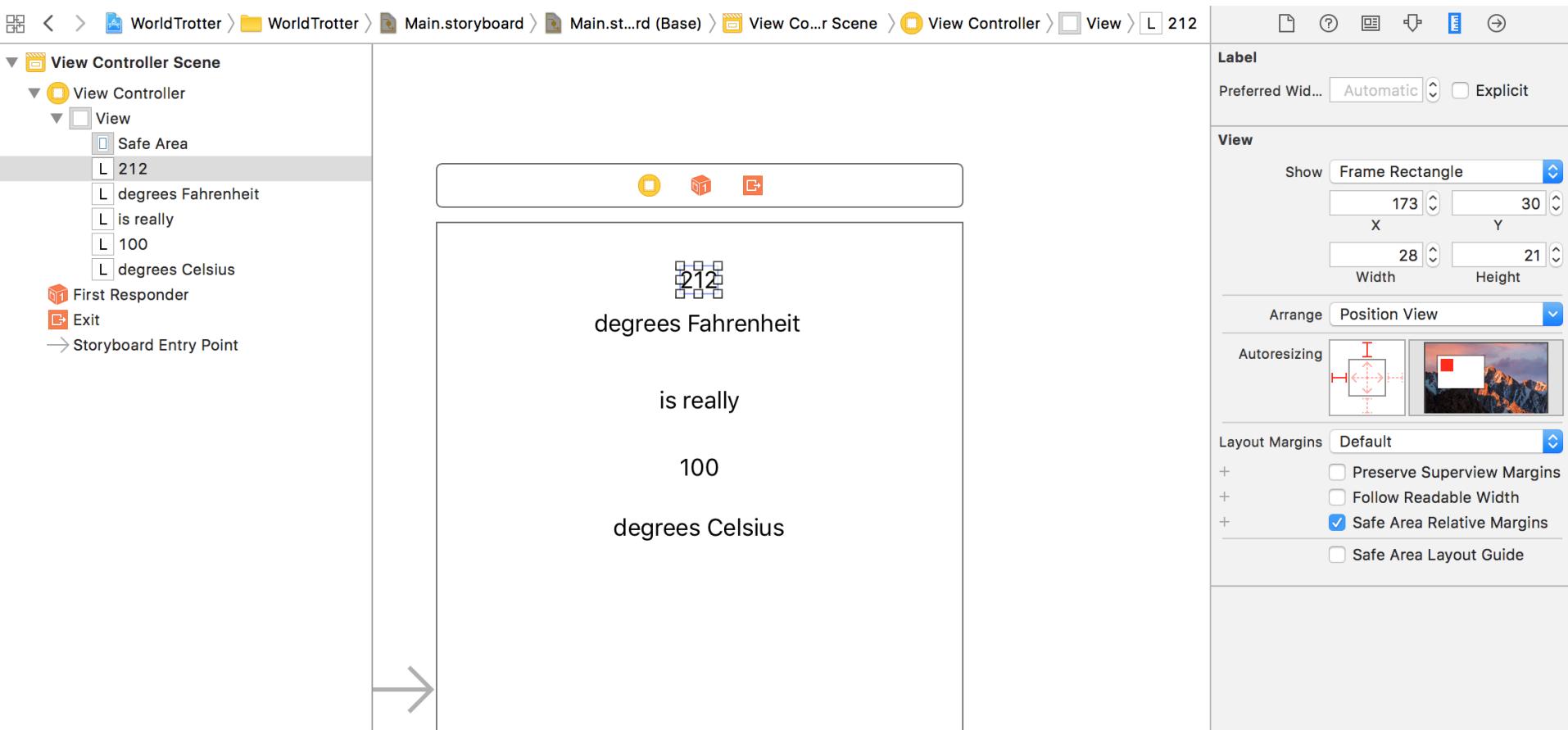
The screenshot shows the Xcode interface with the storyboard open. The storyboard contains a single view controller with a view. Inside the view, there are five labels arranged vertically. The labels contain the text: "212", "degrees Fahrenheit", "is really", "100", and "degrees Celsius". The "degrees Fahrenheit" label is currently selected, as indicated by the highlighted text and the selection handles. The Attributes Inspector on the right is configured for this label, showing the following settings:

- Label** (selected)
- Text**: Plain
- Color**: Default
- Font**: System 17.0
- Dynamic Type**: Automatically Adjusts Font
- Alignment**: Center
- Lines**: 1
- Behavior**: Enabled (checked)
- Baseline**: Align Baselines
- Line Break**: Truncate Tail
- Autoshrink**: Fixed Font Size
- Highlighted**: Default
- Shadow**: Default
- Shadow Offset**: 0 (Width), -1 (Height)
- Width**: 100
- Height**: 100
- View**
- Content Mode**: Left
- Semantic**: Unspecified
- Tag**: 0

The storyboard preview shows the layout of the five labels within the view.

Customizing labels in World Trotter

View Frame values, test on simulator- should look identical.



The screenshot shows the Xcode storyboard editor with the following details:

- Project Navigator:** WorldTrotter > WorldTrotter > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller > View > L 212
- Object Library:** View Controller Scene, View Controller, View, Safe Area, L 212, L degrees Fahrenheit, L is really, L 100, L degrees Celsius, First Responder, Exit, Storyboard Entry Point.
- View Controller Scene:** A view controller with a label "212" and text "degrees Fahrenheit", "is really", "100", and "degrees Celsius".
- Label Properties (L 212):**
 - Label
 - Preferred Width: Automatic (checked)
 - View:
 - Show: Frame Rectangle
 - X: 173, Y: 30
 - Width: 28, Height: 21
 - Arrange: Position View
 - Autoresizing: (red dashed box)
 - Layout Margins: Default
 - Preserve Superview Margins
 - Follow Readable Width
 - Safe Area Relative Margins** (checked)
 - Safe Area Layout Guide

Customizing labels in World Trotter

Customize the view properties. Select bg in the storyboard. Attributes Inspector > Background color > RGB sliders > Hex F5F4F1

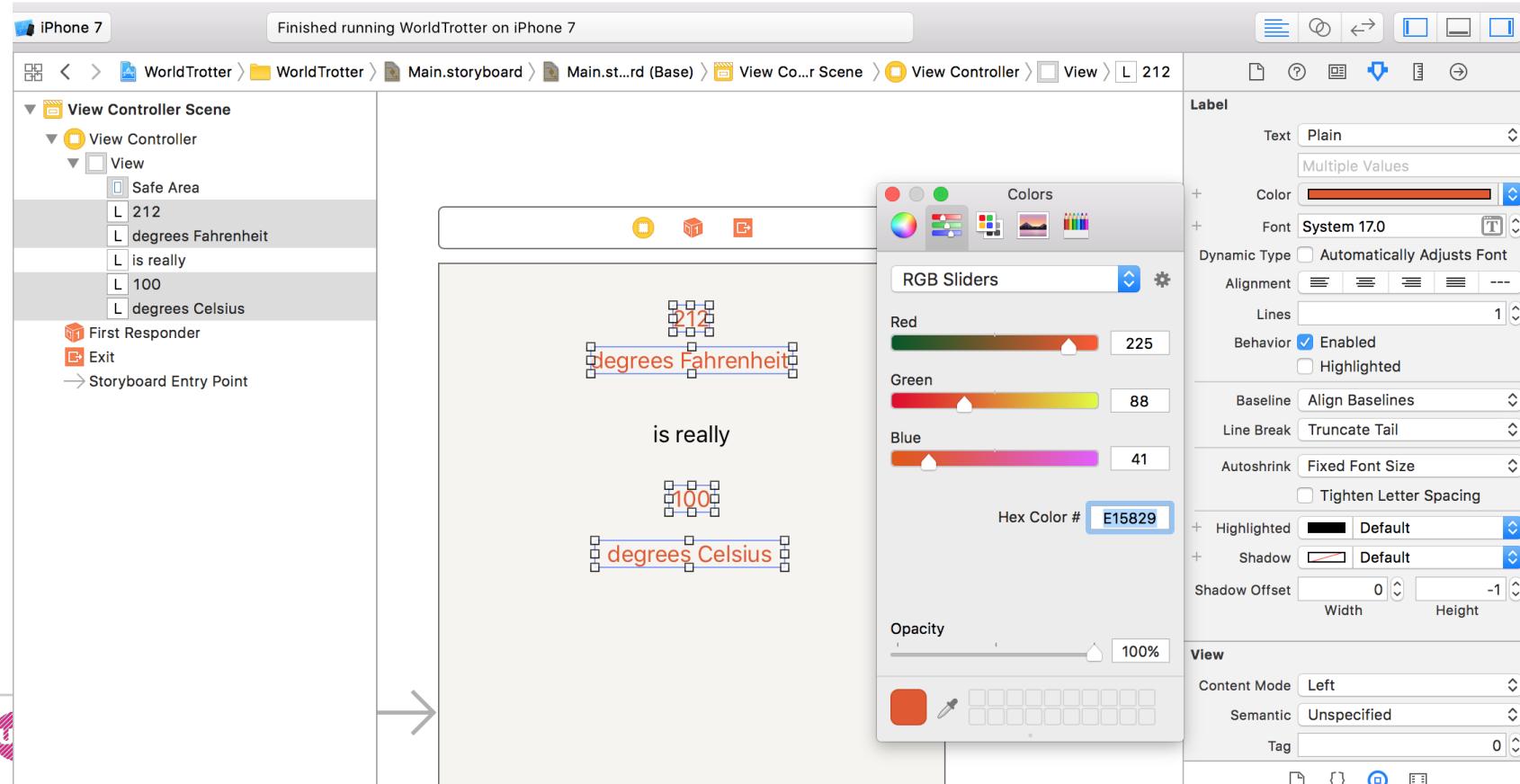
The screenshot shows the Xcode storyboard editor. On the left, the Document Outline shows a View Controller Scene with a View selected. Inside the View, there are several labels with text: "212", "degrees Fahrenheit", "is really", "100", and "degrees Celsius". A color picker window is open in the foreground, titled "Colors", showing "RGB Sliders" with values Red: 245, Green: 244, Blue: 241. Below the sliders is a "Hex Color #" field containing "F5F4F1". To the right, the Attributes Inspector is open for the View, showing the following settings:

- View**
 - Content Mode: Scale To Fill
 - Semantic: Unspecified
 - Tag: 0
- Interaction**
 - User Interaction Enabled
 - Multiple Touch
- Background** (Color well: F5F4F1)
- Tint** (Color well: Default)
- Drawing**
 - Opaque
 - Hidden
 - Clears Graphics Context
 - Clip to Bounds
 - Autoresizes Subviews
- Stretching**
 - X: 0
 - Y: 0
 - Width: 1
 - Height: 1

At the bottom of the Attributes Inspector, there is a "Label" section with the text "Label - A variably sized amount of static text." A small "uic" logo is in the bottom left corner.

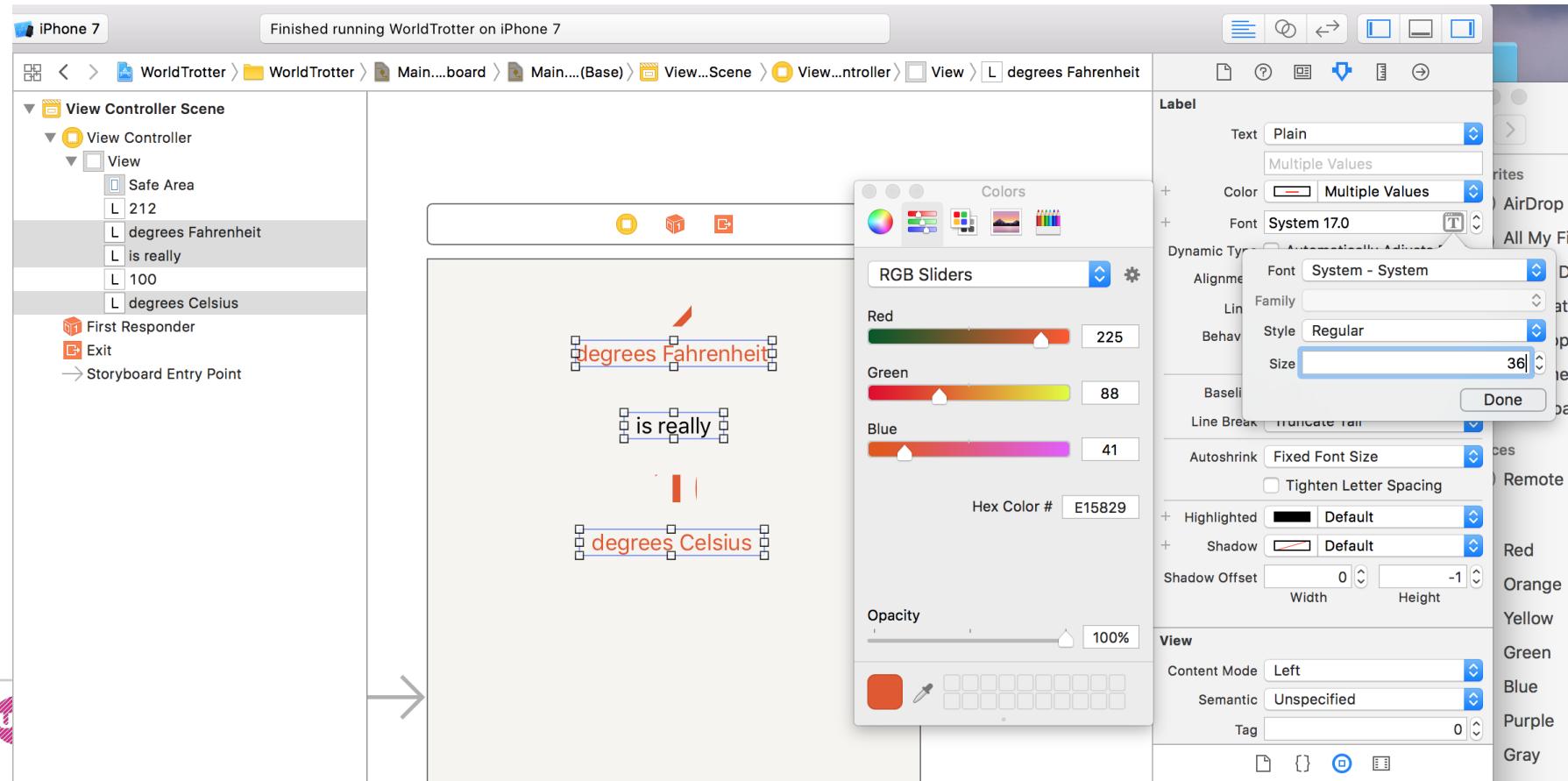
Customizing labels in World Trotter

Select top two and bottom two labels >Color>E15829



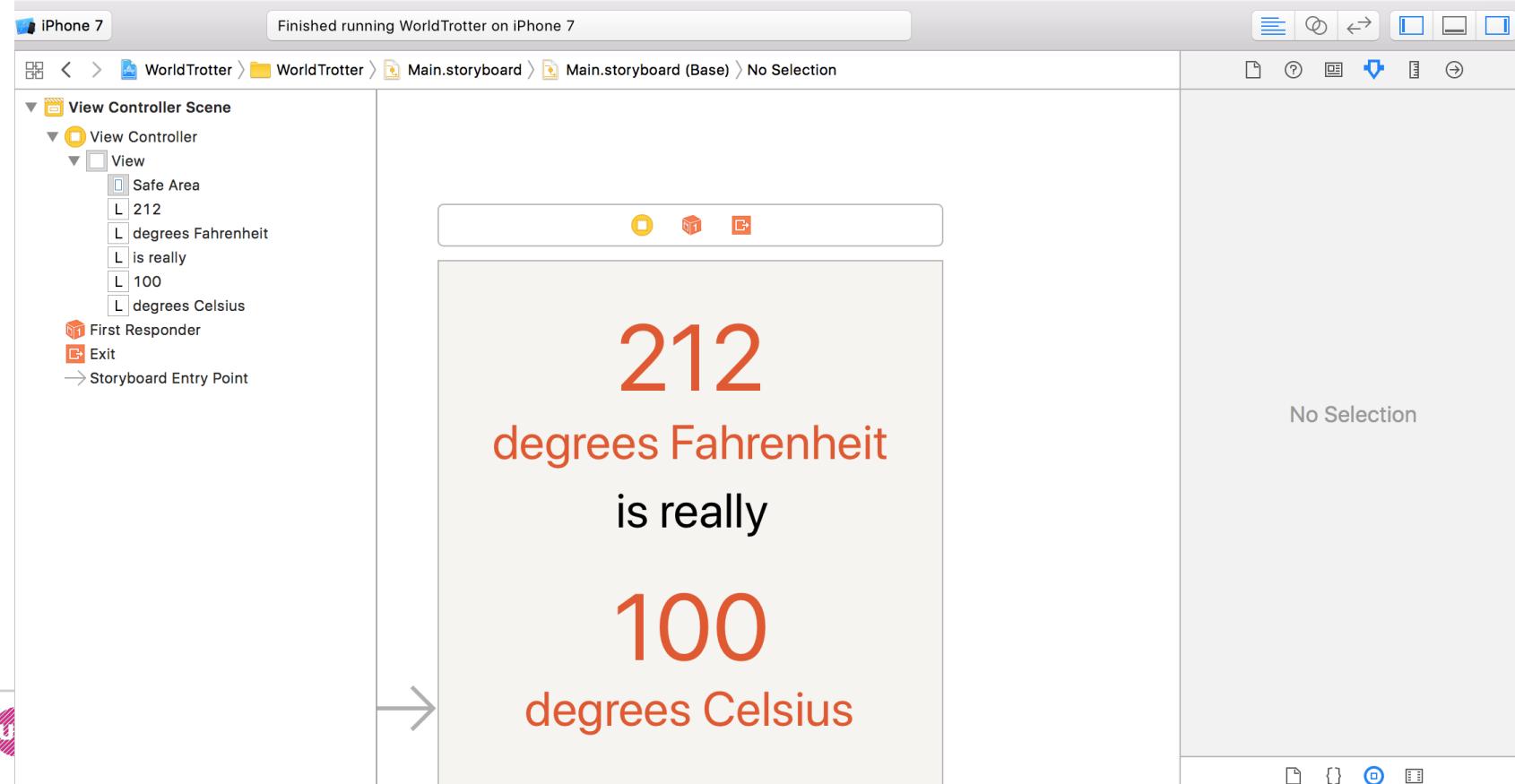
Customizing labels in World Trotter

212 and 100 labels > Font Size > 70. Degrees F and Degrees C > Font > Size > 36



Customizing labels in World Trotter

Select all labels > Scale to fit using (Command -=) Arrange vertically. Test in simulator



Customizing labels in World Trotter

The labels will appear slightly shifted to the left.

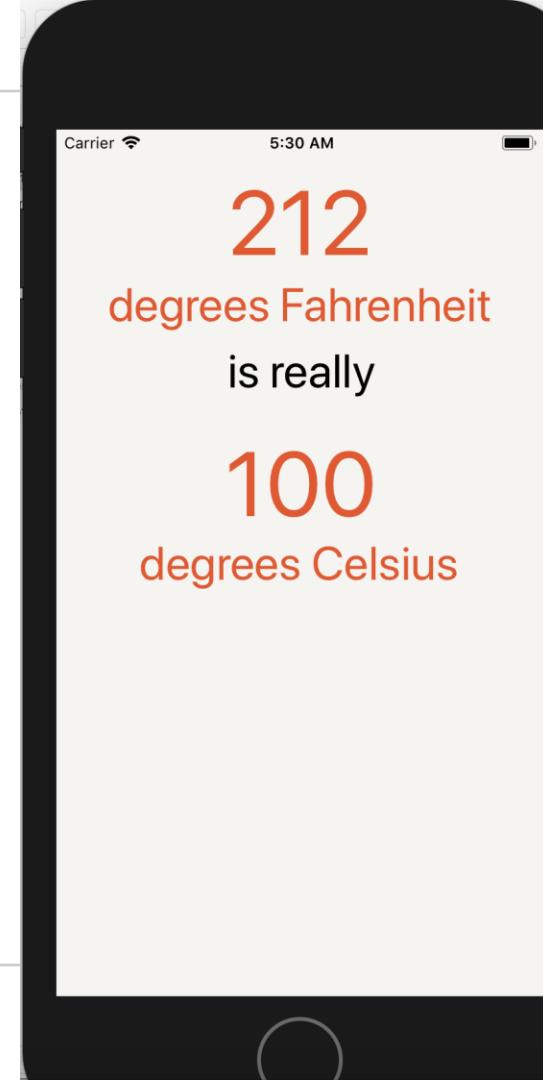
Absolute Frames have major problems:

- When content is resized, the frames do not update automatically.
- The view does not look equally good on different screen sizes

Do not use absolute frame for views.

Use Auto Layout which flexibly computes frames based on the constraints specified for each view.

Label should remain the same distance from the top of the screen; and horizontally centered within their superview. They should update if font or text or labels change.

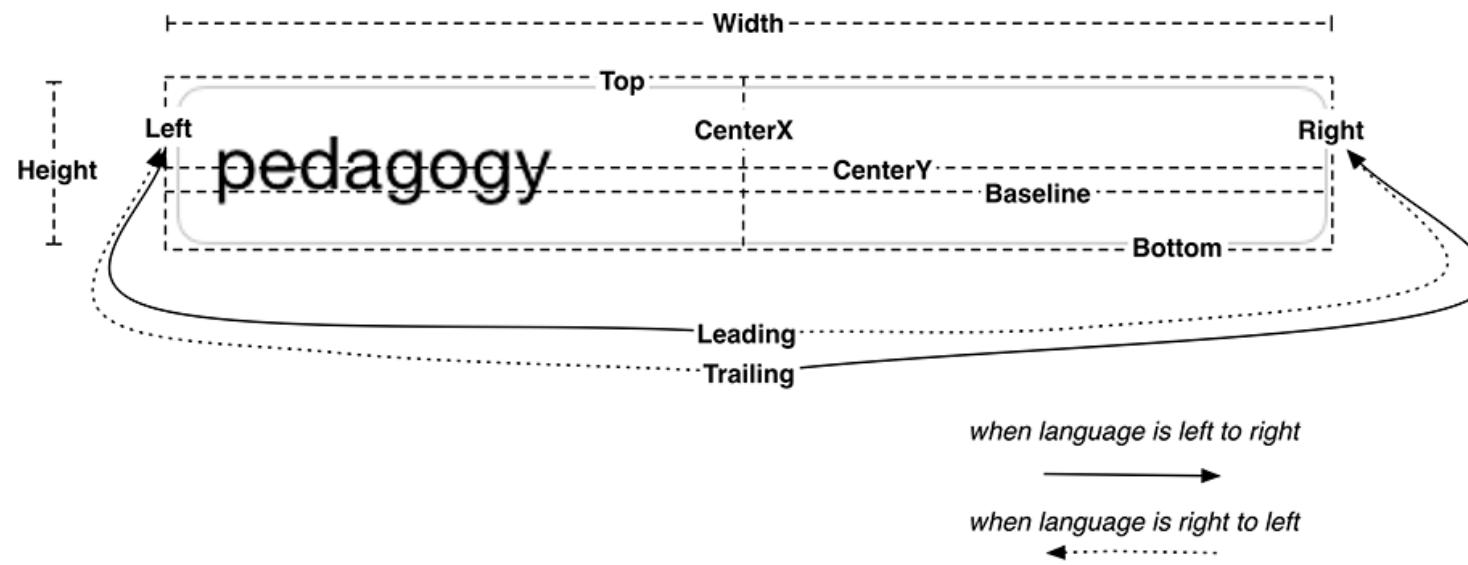


Using Auto Layout system

Auto Layout describes the layout of your views in a relative way that enables their frames to be determined at a runtime, so that the frames' definitions can take into account the screen of the device that the app is running on.

Auto Layout allows Responsive design

The Auto Layout system is based on the *alignment rectangle*. This rectangle is defined by several *layout attributes*



Using Auto Layout system

Width / height

determine the alignment rectangle's size.

Top/Bottom/Left/Right

determine the spacing between the given edge of the alignment rectangle and the alignment rectangle of another view in the hierarchy.

CenterX/CenterY

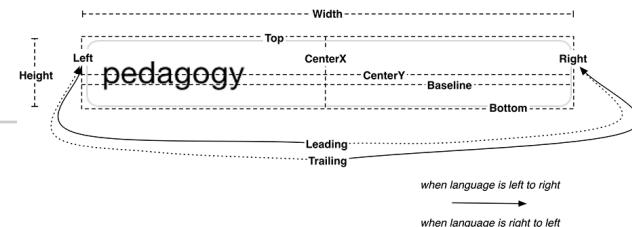
Determine the center point of the alignment rectangle.

Baseline

This value is the same as the bottom attribute for most, but not all, views. For example, **UITextField** defines its baseline as the bottom of the text it displays rather than the bottom of the alignment rectangle. This keeps “descenders” (letters like ‘g’ and ‘p’ that descend below the baseline) from being obscured by a view right below the text field. e the center point of the alignment rectangle.

Leading/Trailing

These values are language-specific attributes. If the device is set to a language that reads left to right (e.g., English), then the leading attribute is the same as the left attribute and the trailing attribute is the same as the right attribute. If the language reads right to left (e.g., Arabic), then the leading attribute is on the right and the trailing attribute is on the left. **Interface Builder** automatically prefers leading and trailing over left and right, and, in general, you should as well.



Using Auto Layout system

By default, every view has an alignment rectangle, and every view hierarchy uses Auto Layout.

The alignment rectangle is very similar to the frame. In fact, these two rectangles are often the same. Whereas the frame encompasses the entire view, the alignment rectangle only encompasses the content that you wish to use for alignment purposes. Frame vs. alignment rectangle:



Frame



Alignment rectangle

Constraints

You cannot define a view's alignment rectangle directly. You do not have enough information (like screen size) to do that. Instead, you provide a set of *constraints*. Taken together, these constraints enable the system to determine the layout attributes, and thus the alignment rectangle, for each view in the view hierarchy.

A *constraint* defines a specific relationship in a view hierarchy that can be used to determine a layout attribute for one or more views. For example, you might add a constraint like, "The vertical space between these two views should always be 8 points," or, "These views must always have the same width." A constraint can also be used to give a view a fixed size, like, "This view's height should always be 44 points."

You do not need a constraint for every layout attribute. Some values may come directly from a constraint; others will be computed by the values of related layout attributes. For example, if a view's constraints set its left edge and its width, then the right edge is already determined (left edge + width = right edge, always). As a general rule of thumb, you need at least two constraints per dimension (horizontal and vertical).

Constraints

If, after all of the constraints have been considered, there is still an ambiguous or missing value for a layout attribute, then there will be errors and warnings from Auto Layout and your interface will not look as you expect on all devices. Debugging these problems is important/

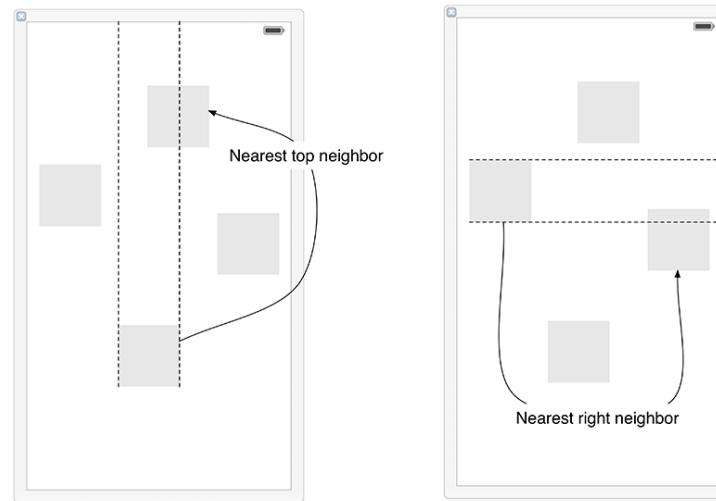
First, describe what you want the view to look like independent of screen size. For example, you might say that you want the top label to be:

- 8 points from the top of the screen
- centered horizontally in its superview
- as wide and as tall as its text

Constraints

To turn this description into constraints in **Interface Builder**, it will help to understand how to find a view's *nearest neighbor*. The nearest neighbor is the closest sibling view in the specified direction.

If a view does not have any siblings in the specified direction, then the nearest neighbor is its superview, also known as its container.



Constraints

the constraints for the label:

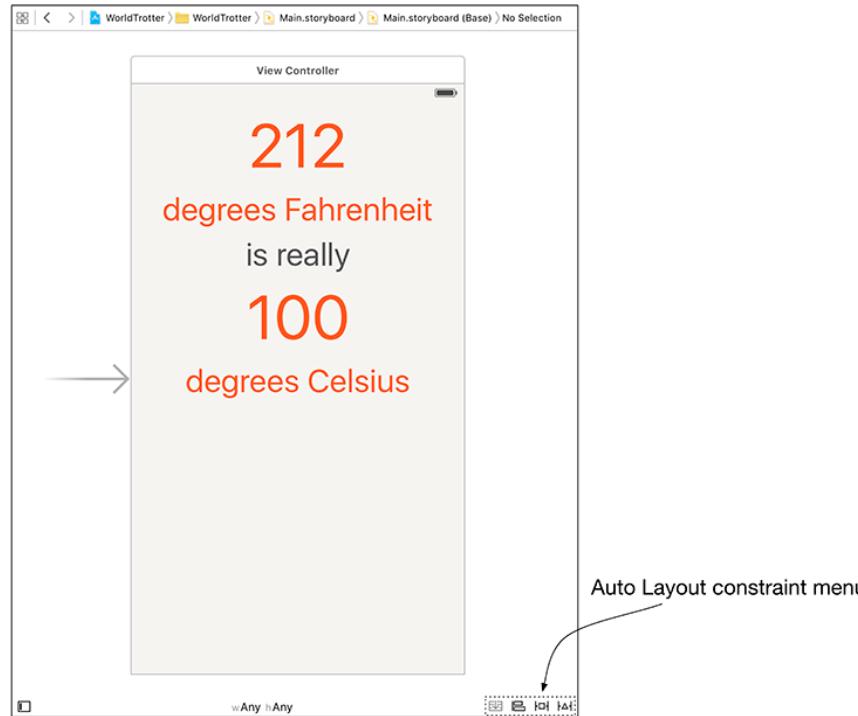
- The label's top edge should be 8 points away from its nearest neighbor (which is its container – the view of the **ViewController**).
- The label's center should be the same as its superview's center.
- The label's width should be equal to the width of its text rendered at its font size.
- The label's height should be equal to the height of its text rendered at its font size.

If you consider the first and fourth constraints, you can see that there is no need to explicitly constrain the label's bottom edge. It will be determined from the constraints on the label's top edge and the label's height. Similarly, the second and third constraints together determine the label's right and left edges.

Constraints can be added using **Interface Builder** or in code. Apple recommends that you add constraints using **Interface Builder** whenever possible, and that is what you will do here. However, if your views are created and configured programmatically, then you can add constraints in code.

Adding Constraints in Interface Builder

Select the top label on the canvas. In the bottom righthand corner of the canvas, find the Auto Layout constraint menu 

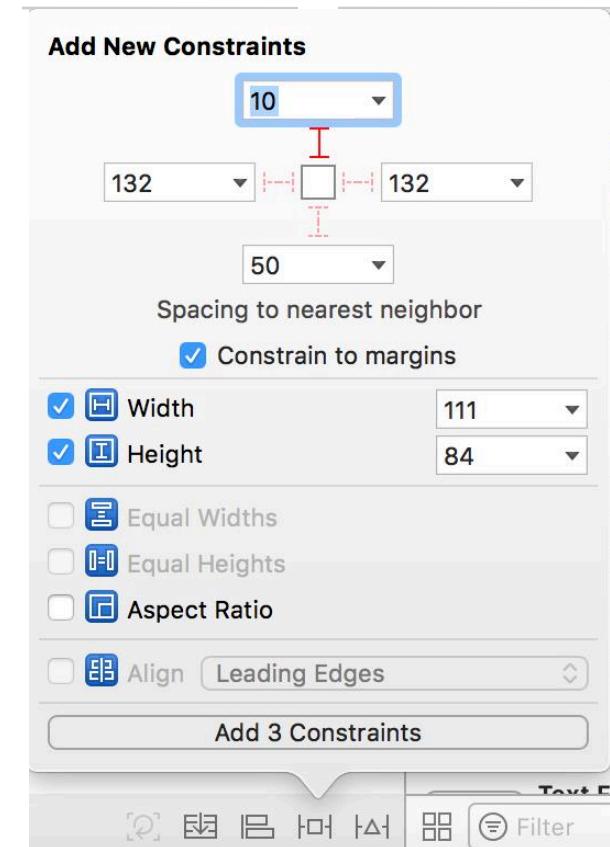


Adding Constraints in Interface Builder

At the top of the **Pin** menu are four values that describe the label's current spacing from its nearest neighbor on the canvas. For this label, you are only interested in the top value. To turn this value into a constraint, click the top red strut separating the value from the square in the middle. The strut will become a solid red line.

In the middle of the menu, find the label's **Width** and **Height**. The values next to **Width** and **Height** indicate the current canvas values. To constrain the label's width and height to the current canvas values, check the boxes next to **Width** and **Height**.

Click button **Add 3 Constraints**.



WorldTrotter > WorldTrotter > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller > View > L 212

View Controller Scene

View Controller

View

Safe Area

L 212

L degrees Fahrenheit

L is really

L 100

L degrees Celsius

First Responder

Exit

Storyboard Entry Point

View as: iPhone 8 (wC hR) 100% Filter

Label

Text: Plain 212

Color: Red

Font: System 70.0

Dynamic Type: Automatically Adjusts Font

Alignment: Center

Lines: 1

Behavior: Enabled

Baseline: Align Baselines

Line Break: Truncate Tail

Autoshrink: Fixed Font Size

Tighten Letter Spacing

Highlighted: Default

Shadow: Default

Shadow Offset: 0 -1

Add New Constraints

10

132 132 50

Constrain to margins

Width: 111

Height: 84

Equal Widths

Equal Heights

Aspect Ratio

Align: Leading Edges

Add 3 Constraints

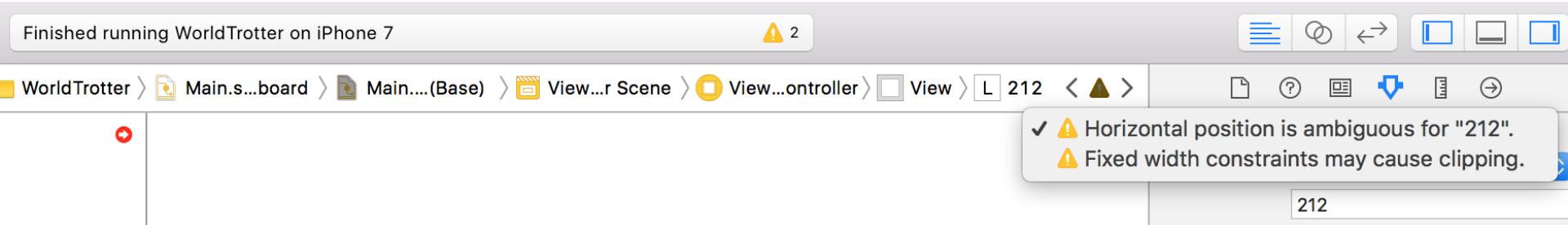
35

The screenshot shows a Xcode storyboard interface. On the left, the Document Outline shows a hierarchy of View Controller, View, and various labels (L 212, L degrees Fahrenheit, L is really, L 100, L degrees Celsius) and storyboard elements (First Responder, Exit, Storyboard Entry Point). The main canvas displays a storyboard scene with five text labels: '212' (large red font, selected), 'degrees Fahrenheit', 'is really', '100', and 'degrees Celsius'. The '212' label has a bounding box with handles. The right panel shows the 'Label' attributes for the selected '212' label, including Text (Plain), Color (Red), Font (System 70.0), Alignment (Center), Lines (1), Behavior (Enabled), Baseline (Align Baselines), Line Break (Truncate Tail), and Autoshrink (Fixed Font Size). A 'Add New Constraints' dialog is open, showing constraints for the '212' label relative to the Safe Area (top: 10, left: 132, right: 132, bottom: 50). The storyboard is set to 'iPhone 8 (wC hR)' and 100% zoom.

Adding Constraints in Interface Builder

At this point, you have not specified enough constraints to fully determine the alignment rectangle. **Interface Builder** will help you determine what the problem is.

In the top right corner of **Interface Builder**, notice the yellow warning sign. Click on this icon to reveal the issue: "Horizontal position is ambiguous for "212"."



Adding Constraints in Interface Builder

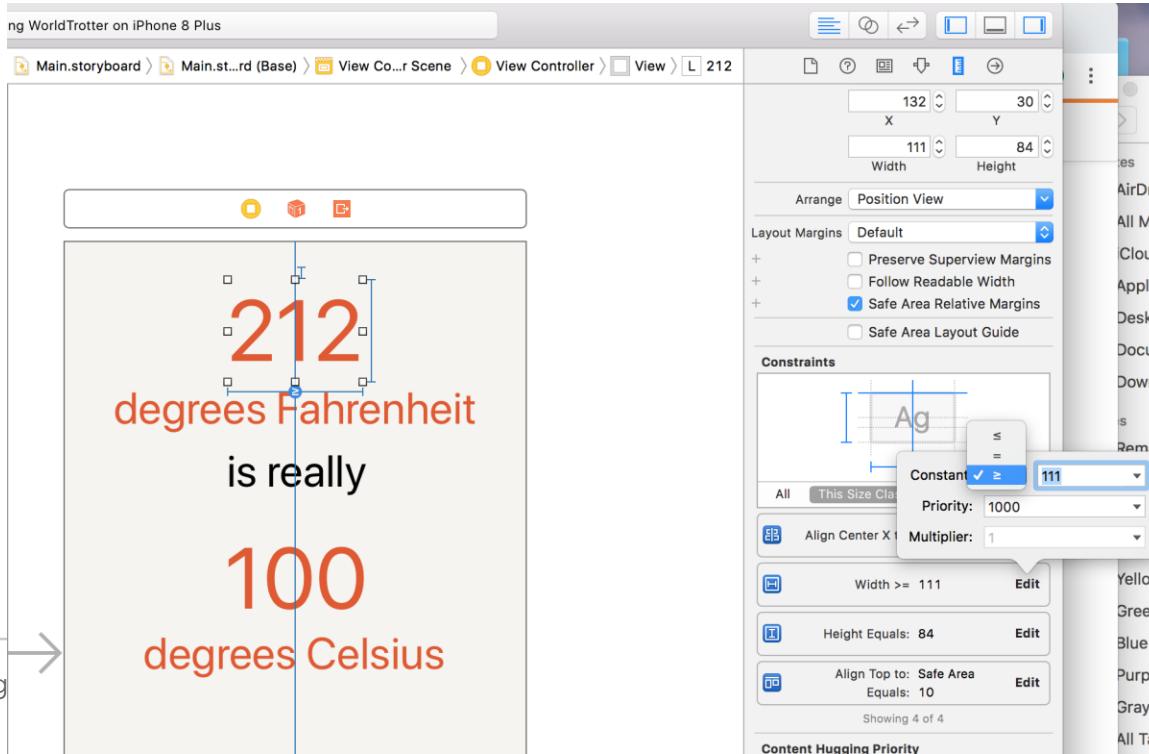
With the top label still selected, click the align_  icon (the second from the left in the Auto Layout constraints menu) to reveal the Align menu. If you have multiple views selected, this menu will allow you to align attributes among the views. Since you have only selected one label, the only options you are given are to align the view within its container.

select **Horizontally in Container (do not click Add 1 Constraint yet)**. Once you add this constraint, there will be enough constraints to fully determine the alignment rectangle. To ensure that the frame of the label matches the constraints specified, open the **Update Frames** pop-up menu from the Align menu and select Items of New Constraints. This will reposition the label to match the constraints that have been added. Now click on Add 1 Constraint to add the centering constraint and reposition the label.

The label's constraints are all blue now that the alignment rectangle for the label is fully specified. Additionally, the warning at the top right corner of Interface Builder is now gone.

“Fixed Width Constraints May Cause Clipping” Warning

to fix the fixed-width warnings in your app - change the width of the object spacings from fixed width to greater than or equal or less than or equal. (<=. Or >=) select the object in interface builder > go to the size inspector > constraints > width>changing to <=



Intrinsic content size - flexible size

because we added explicit width and height constraints to the label, its size is not flexible. If the text or font were to change, the label will not look centered.

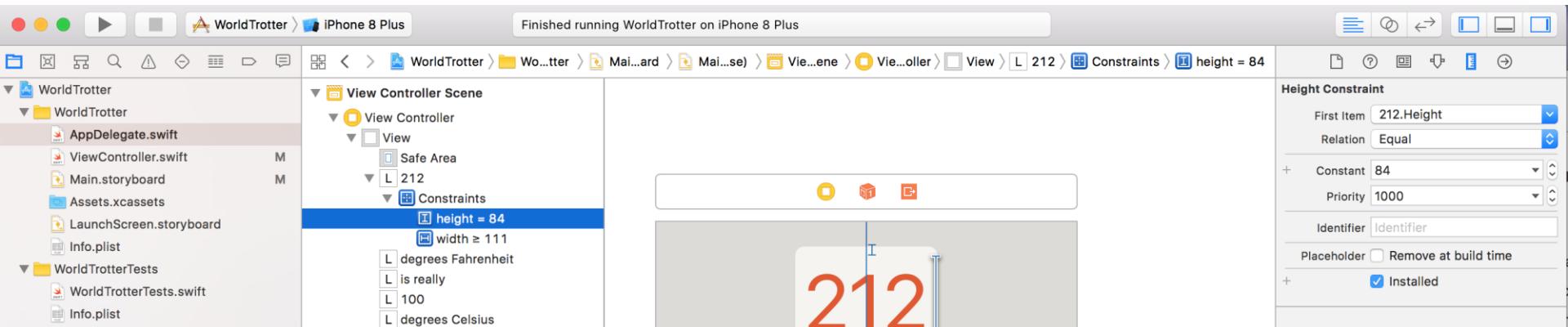
the intrinsic content size is the size that a view “wants” to naturally be.

For labels, this size is the size of the text rendered at the given font. For images, this is the size of the image itself. A view’s intrinsic content size acts as implicit width and height constraints.

If you do not specify constraints that explicitly determine the width, the view will be its intrinsic width. The same goes for the height.

To remove the explicit width and height constraints:

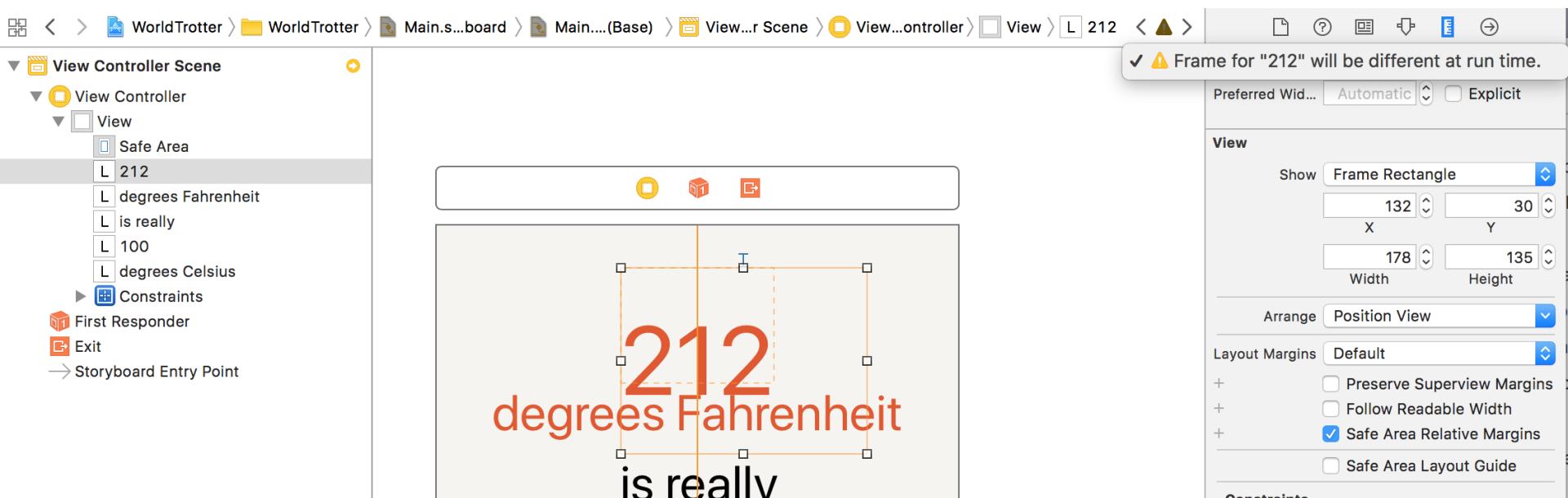
select the width constraint on the label, press the Delete key. Do the same for the height constraint.



Misplaced views

blue constraints indicate that the alignment rectangle for a view is fully specified. Orange constraints often indicate a **misplaced view**. This means that the frame for the view in Interface Builder is different than the frame that Auto Layout has computed.

Resize the top label on the canvas using the resize controls and look for the yellow warning in the top right corner of the canvas. Click on this warning icon to reveal the problem: “Frame for “212” will be different at run time”



Misplaced views

As the warning says, the frame at runtime will not be the same as the frame specified on the canvas.
an orange dotted line that indicates what the runtime frame will be.

Build and run the application. Notice that the label is still centered despite the new frame that you gave it in Interface Builder. However, the disconnect between what you have specified in Interface Builder and the constraints computed by Auto Layout will cause problems down the line as you continue to build your views.

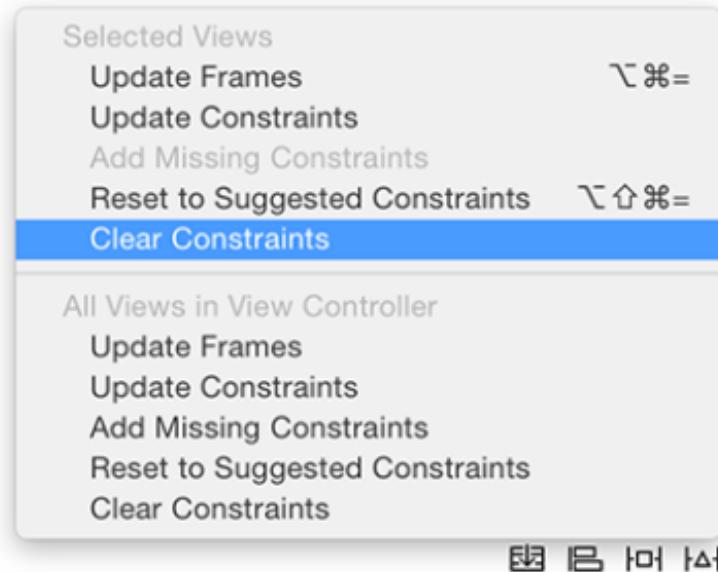
to fix the misplaced view:

- select the top label on the canvas
- click Update Frames icon. This will update the frame of the label to match the frame that the constraints will compute.

word of caution: if you try to update the frames for a view that does not have enough constraints, you will almost certainly get unexpected results. If that happens, undo the change and inspect the constraints to see what is missing.

Removing and adding more constraints

Select the top label on the canvas. Open the Resolve Auto Layout Issues menu and select Clear Constraints from the Selected Views section.



Removing and adding more constraints

to add the constraints to all of the views in two steps:

1. center the top label horizontally within the superview.
2. add constraints that pin the top of each label to its nearest neighbor while aligning the centers of all of the labels.



Removing and adding more constraints

Select the top label. Open the Align menu and choose Horizontally in Container with a constant of 0. Make sure that *Update Frames* has *None* selected; remember that you do not want to update the frame of a view that does not have enough constraints, and this one constraint will certainly not provide enough information to compute the alignment rectangle. Add 1 Constraint.

select all five labels on the canvas to add constraints to multiple views simultaneously.

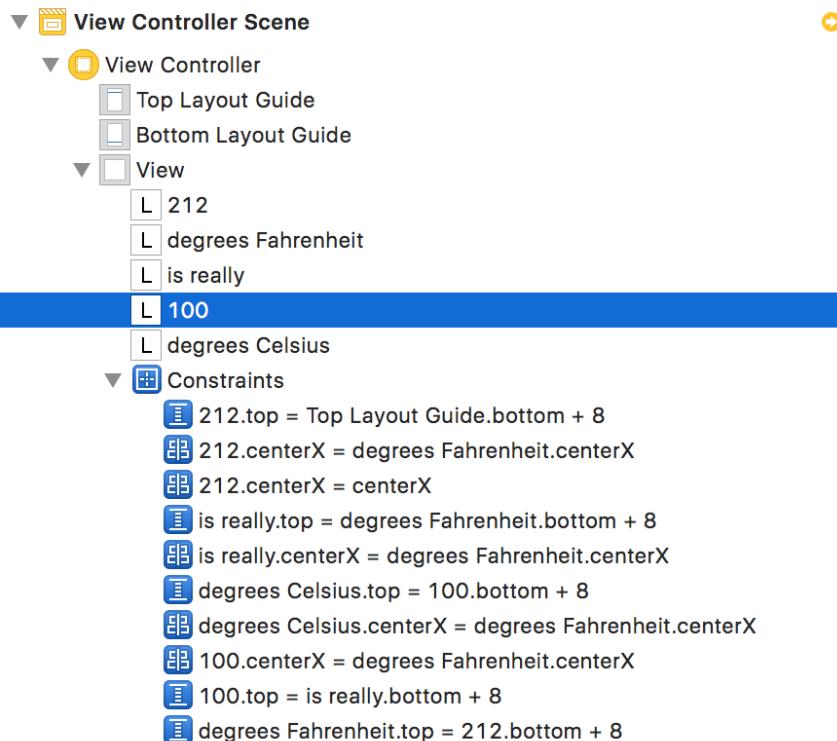
Open Add Constraints menu :

- Select the top strut and make sure it has a constant of 8.
- From the Align menu, choose Horizontal Centers.
- From the Update Frames menu, choose Items of New Constraints.



Removing and adding more constraints

WorldTrotter > WorldTrotter > Main.storyboard > Main....



The screenshot shows the Xcode storyboard hierarchy. At the top, there is a navigation bar with icons for file, back, forward, and project navigation. Below the navigation bar is a breadcrumb trail: WorldTrotter > WorldTrotter > Main.storyboard > Main.... The main area shows the storyboard hierarchy:

- View Controller Scene (selected)
- View Controller
 - Top Layout Guide
 - Bottom Layout Guide
 - View
 - L 212
 - L degrees Fahrenheit
 - L is really
 - L 100** (highlighted with a blue bar)
 - L degrees Celsius
 - Constraints
 - 212.top = Top Layout Guide.bottom + 8
 - 212.centerX = degrees Fahrenheit.centerX
 - 212.centerX = centerX
 - is really.top = degrees Fahrenheit.bottom + 8
 - is really.centerX = degrees Fahrenheit.centerX
 - degrees Celsius.top = 100.bottom + 8
 - degrees Celsius.centerX = degrees Fahrenheit.centerX
 - 100.centerX = degrees Fahrenheit.centerX
 - 100.top = is really.bottom + 8
 - degrees Fahrenheit.top = 212.bottom + 8
- First Responder
- Exit

First Responder

Exit

Storyboard Entry Point

Daria Tsoupikova

Removing and adding more constraints

Assignment 8

Assignment 8: Quiz app (based on Chapter 1)

Read chapters 1, 2 and 3



Good Code **is as little** **code as possible.**

