

iOS Programming

Lecture 9



Recap

We built a single page app today – fully functioning



View Hierarchy

Actions and Outlets

Home Value: \$200,000

Down Payment: \$20,000

Mortgage Amount:

Loan Duration: Select Duration
15 Years

Rate: To be determined

Reset

Compute

Today

Adaptive UI

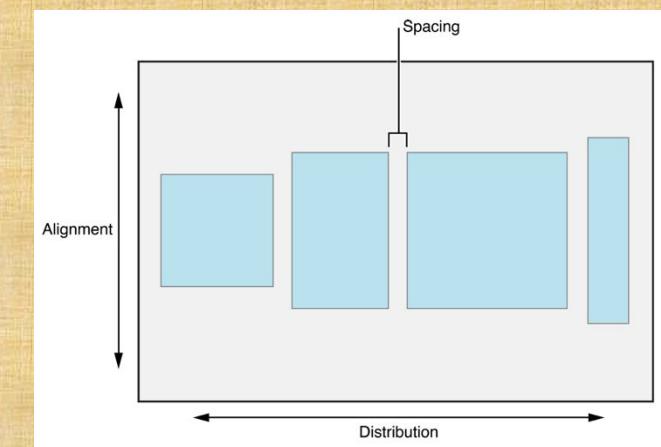
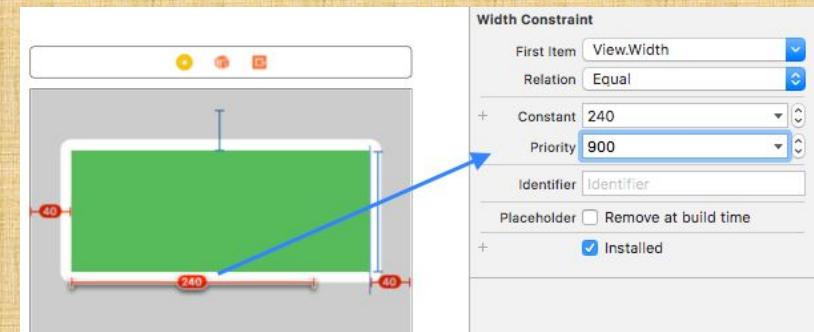


Images

App thinning

Constraints

Stack Views



Deployment Targets



Can I build apps for iPhone 4s still?

AutoLayout

PROJECT AutoLayout

TARGETS AutoLayout

Deployment Target

iOS Deployment Target 14.0

Configurations

Name Based on Configuration File
▶ Debug No Configurations Set
▶ Release No Configurations Set

+

Use Release for command-line builds

Parallelize build for command-line builds (does not apply when using schemes)

Localizations

Localization Resources
Base 2 Files Localized
English — Development Language 0 Files Localized

+

Use Base Internationalization

How to get more simulators

A screenshot of the Xcode Components window. The window title is "Components". Below the title are several icons: gear, user, server, navigation, themes, text editing, key bindings, source control, components (which is selected and highlighted in grey), locations, and server & bots. The main area is titled "Simulators". A table lists various iOS simulators with their sizes:

Simulator	Size
iOS 13.5 Simulator	3.36 GB
iOS 13.4 Simulator	3.36 GB
iOS 13.3 Simulator	3.27 GB
iOS 13.2 Simulator	3.27 GB
iOS 13.1 Simulator	3.24 GB
iOS 13.0 Simulator	3.26 GB
iOS 12.4 Simulator	2.57 GB
iOS 12.2 Simulator	2.54 GB
iOS 12.1 Simulator	2.49 GB
iOS 12.0 Simulator	2.4 GB
iOS 11.4 Simulator	2.14 GB
iOS 11.3 Simulator	2.14 GB
iOS 11.2 Simulator	2.11 GB

Mortgage App – Do we have issues?



4:30

Home Value: \$200,000

Down Payment: \$20,000

Mortgage Amount:

Loan Duration: Select Duration
15 Years

Rate: To be determined

Reset Compute

A screenshot of a mortgage calculator app interface on an iPhone. The screen shows fields for Home Value (\$200,000), Down Payment (\$20,000), and Mortgage Amount. Below these are fields for Loan Duration (set to 15 Years) and Rate (To be determined). At the bottom are 'Reset' and 'Compute' buttons.

Let's try landscape – What do we observe?

Home Value: \$200,000

Down Payment: \$20,000

Mortgage Amount:

Loan Duration: Select Duration
15 Years

A screenshot of the same mortgage calculator app interface, but in landscape mode. The layout is wider and the text is smaller. The 'Select Duration' field is labeled 'Select Duration' and '15 Years'.



Solution 1 – Lock orientation

You can do so programmatically by overriding UIViewController functions



```
// Set the shouldAutorotate to False  
override open var shouldAutorotate: Bool {  
    return false  
}  
  
// Specify the orientation.  
override open var supportedInterfaceOrientations: UIInterfaceOrientationMask {  
    return .portrait  
}
```

Solution 1 – Lock orientation



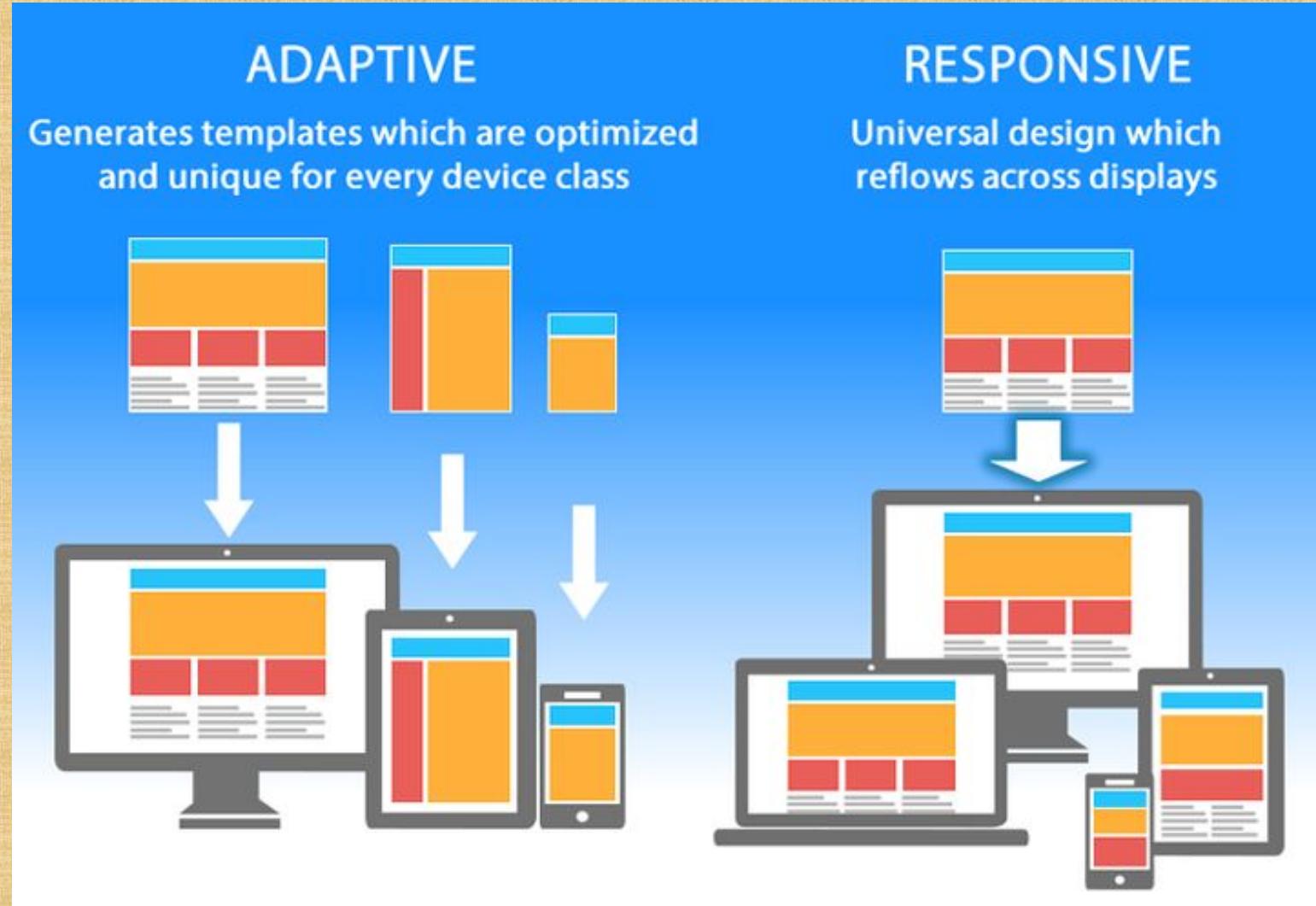
You can do so by configuration for entire app – Info.plist

▼ Supported interface orientations	Array	(3 items)
Item 0	+ - String	Landscape (right home button)
Item 1	+ - String	Landscape (left home button)
Item 2	String	Portrait (bottom home button)
▼ Supported interface orientations...	Array	(4 items)
Item 0	String	Landscape (right home button)
Item 1	String	Landscape (left home button)
Item 2	String	Portrait (bottom home button)
Item 3	String	Portrait (top home button)

Let's discuss pros and
cons of programmatic vs
config



Adaptive vs Responsive





iPhone Screen Sizes

Model	XS Max 11 Pro Max	X, XS 11 Pro	XR 11	8+ 7+ 6+, 6S+	8 7 6, 6S	SE 5, 5S, 5C	4, 4S	3G, 3GS 2G
Points	414x896	375x812	414x896	414x736	375x667	320x568	320x480	320x480
Rendered @	3x	3x	2x	3x	2x	2x	2x	1x
Rendered Pixels	1242x2688	1125x2436	828x1792	1242x2208	750x1334	640x1136	640x960	320x480
Physical Screen	6.5"	5.8"	6.1"	5.5"	4.7"	4"	3.5"	3.5"



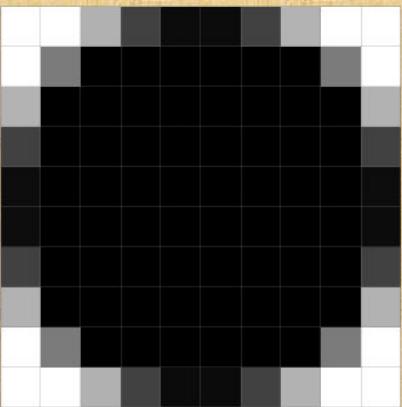
iPhone Screen Sizes



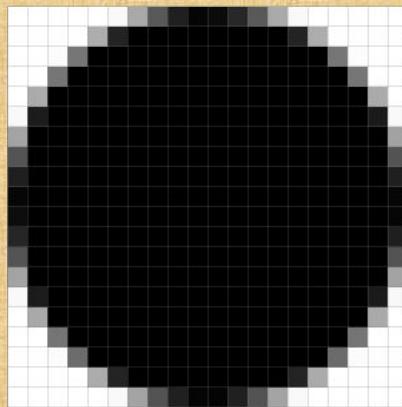


Images – 1x, 2x and 3x

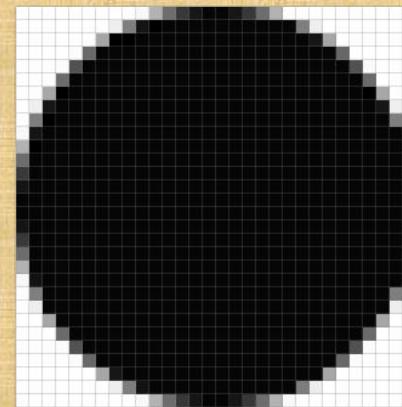
The coordinate system iOS uses to place content onscreen is based on measurements in points, which map to pixels in the display. A standard-resolution display has a 1:1 pixel density (or @1x), where one pixel is equal to one point. High-resolution displays have a higher pixel density, offering a scale factor of 2.0 or 3.0 (referred to as @2x and @3x). As a result, high-resolution displays demand images with more pixels. For example, suppose you have a standard resolution (@1x) image that's 100px × 100px. The @2x version of this image would be 200px × 200px, and the @3x version would be 300px × 300px.



1x
(10 x 10 px)



2x
(20 x 20 px)



3x
(30 x 30 px)

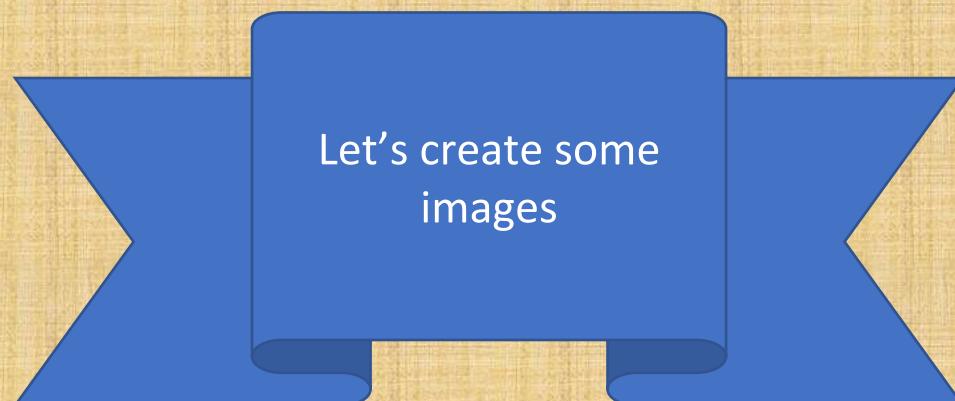
How do I get these images?



In enterprises – UX Designer will share those

For our example we will use:

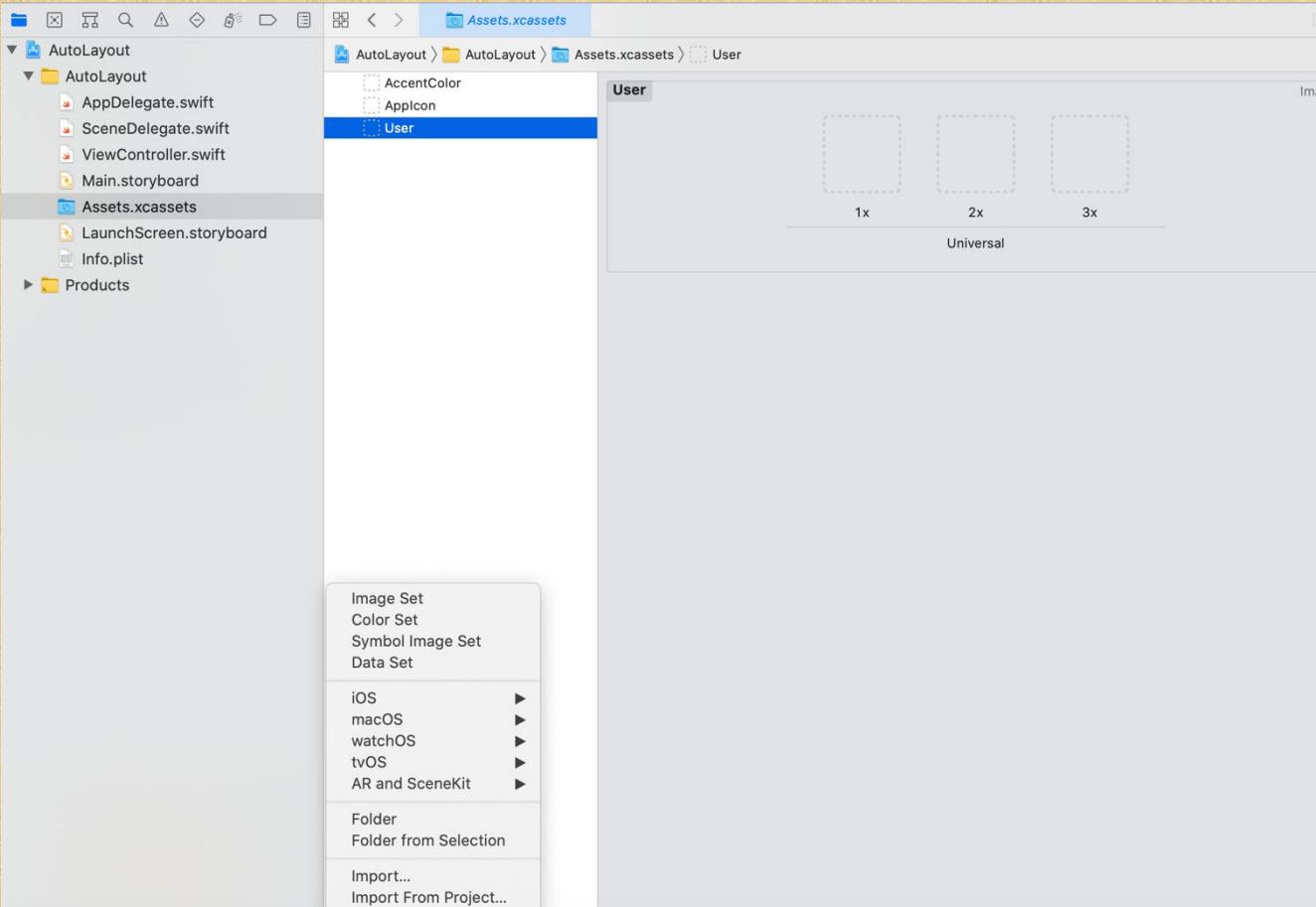
<https://hotpot.ai/icon-resizer>



	AppleWatch-Icon-24@2x.png
	AppleWatch-Icon-27.5@2x.png
	AppleWatch-Icon-29@2x.png
	AppleWatch-Icon-29@3x.png
	AppleWatch-Icon-40@2x.png
	AppleWatch-Icon-44@2x.png
	AppleWatch-Icon-86@2x.png
	AppleWatch-Icon-98@2x.png
	Icon-20.png
	Icon-20@2x.png
	Icon-20@3x.png
	Icon-29.png
	Icon-29@2x.png
	Icon-29@3x.png
	Icon-40.png
	Icon-40@2x.png
	Icon-40@3x.png
	Icon-50.png
	Icon-50@2x.png
	Icon-57.png
	Icon-57@2x.png
	Icon-60@2x.png
	Icon-60@3x.png
	Icon-72.png
	Icon-72@2x.png
	Icon-76.png
	Icon-76@2x.png
	Icon-83.5@2x.png
	iTunesArtwork-1024.png

Images – Asset Catalogue

You really don't have to worry about figuring which asset to use on which device type. iOS will take care of this.



App Thinning



App thinning is a concept for modern day interactive apps where there are a lot of resources. The App Store and OS install the app according to the device, with a minimal footprint. This helps in making the app which occupies less space, are easy to download and make use of all features. Faster downloads and minimum space occupancy gives a better app experience.

App Thinning - Slicing



Slicing is the process of creating different variations of app bundle for different target devices. A variant contains only the executable architecture and resources those are needed for the target device. The store will create and deliver different variants based on the devices your app supports. Image resources are sliced according to their resolution and device family. GPU resources are sliced according to device capabilities. For tvOS apps, assets in catalogs shared between iOS and tvOS targets are sliced and large app icons are removed. When the user installs an app, a variant for the user's device is downloaded and installed.

Note: Sliced apps are supported on devices running 9.0 and later; otherwise, the store delivers universal apps to customers.

App Thinning - Bitcode



Bitcode is an intermediate representation of a compiled program. Apps you upload to iTunes Connect that contain bitcode will be compiled and linked on the store. Including bitcode will allow Apple to re-optimize your app binary in the future without the need to submit a new version of your app to the store.

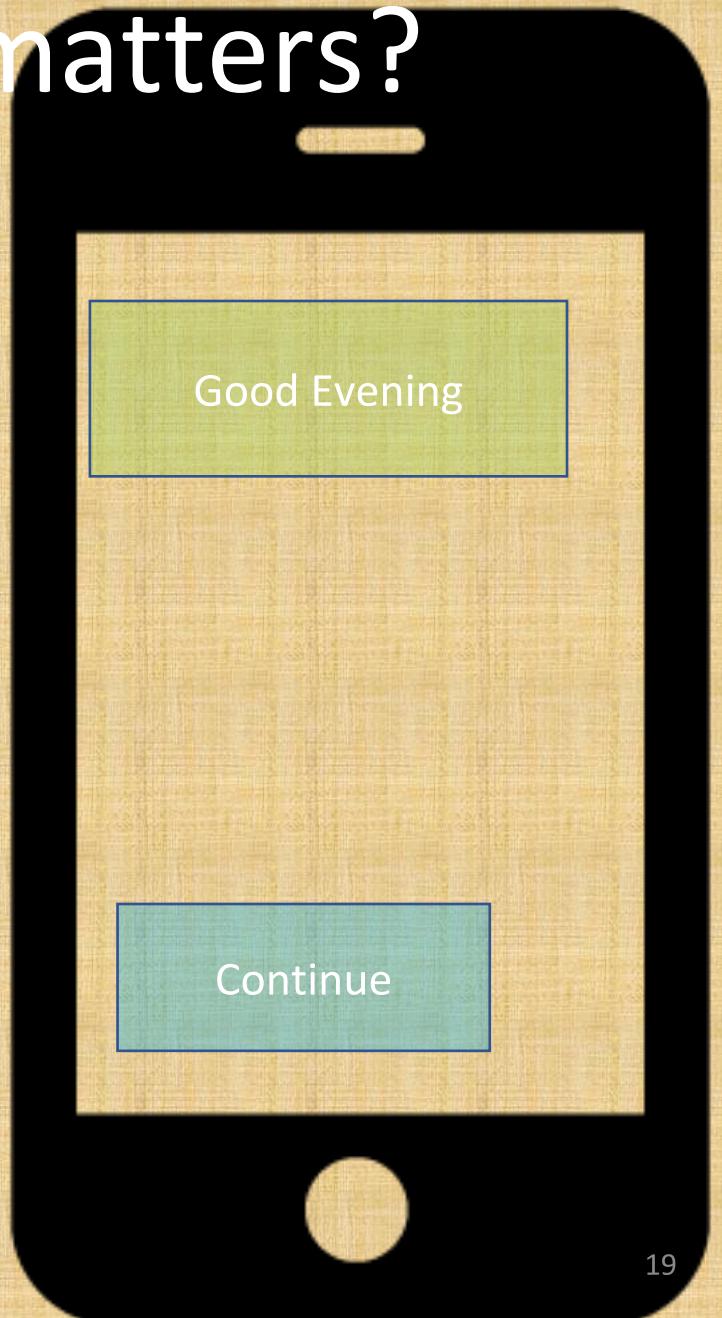
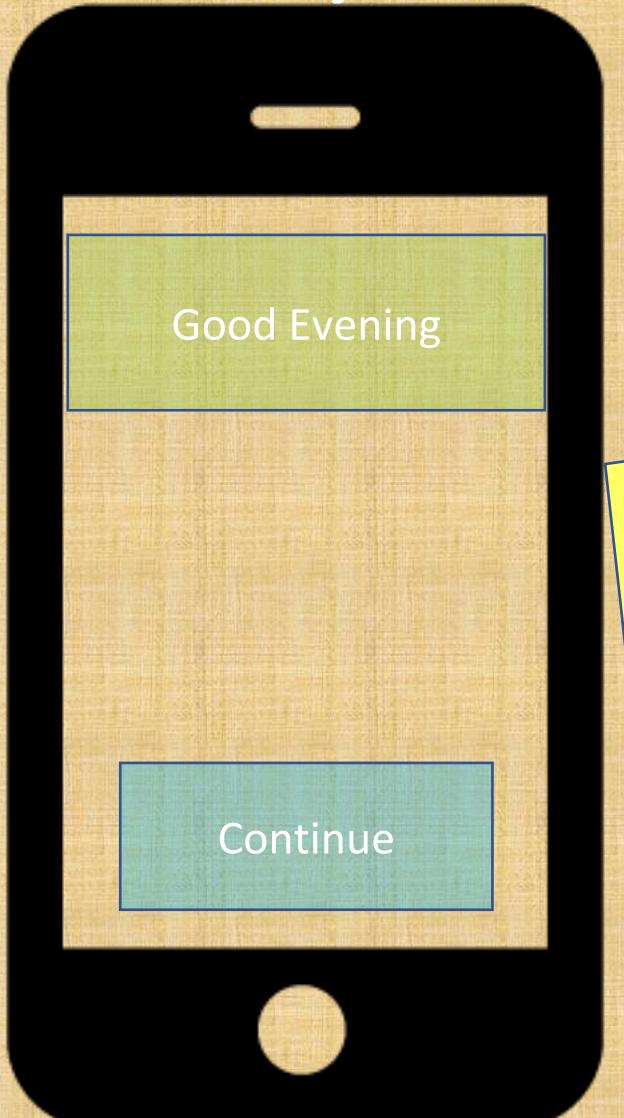
Xcode hides symbols generated during build time by default, so they are not readable by Apple. Only if you choose to include symbols when uploading your app to iTunes Connect would the symbols be sent to Apple. You must include symbols to receive crash reports from Apple.

App Thinning - On Demand Resources



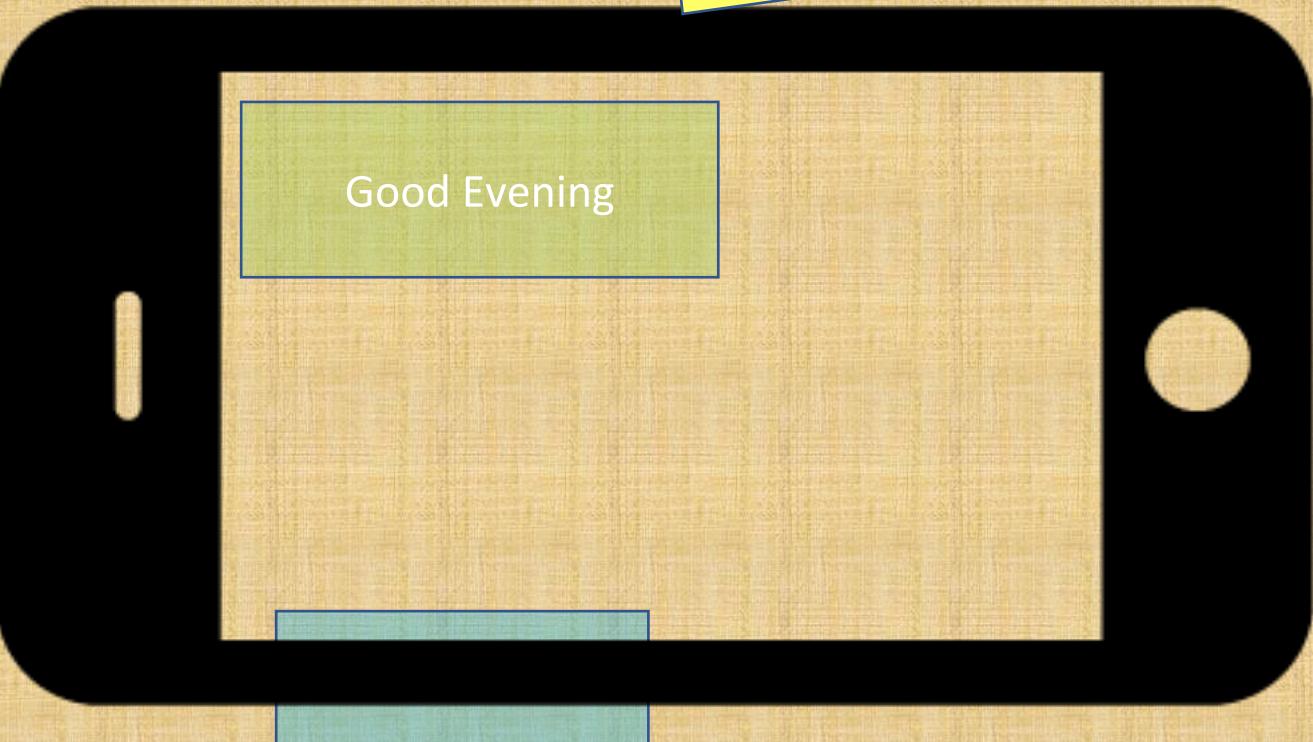
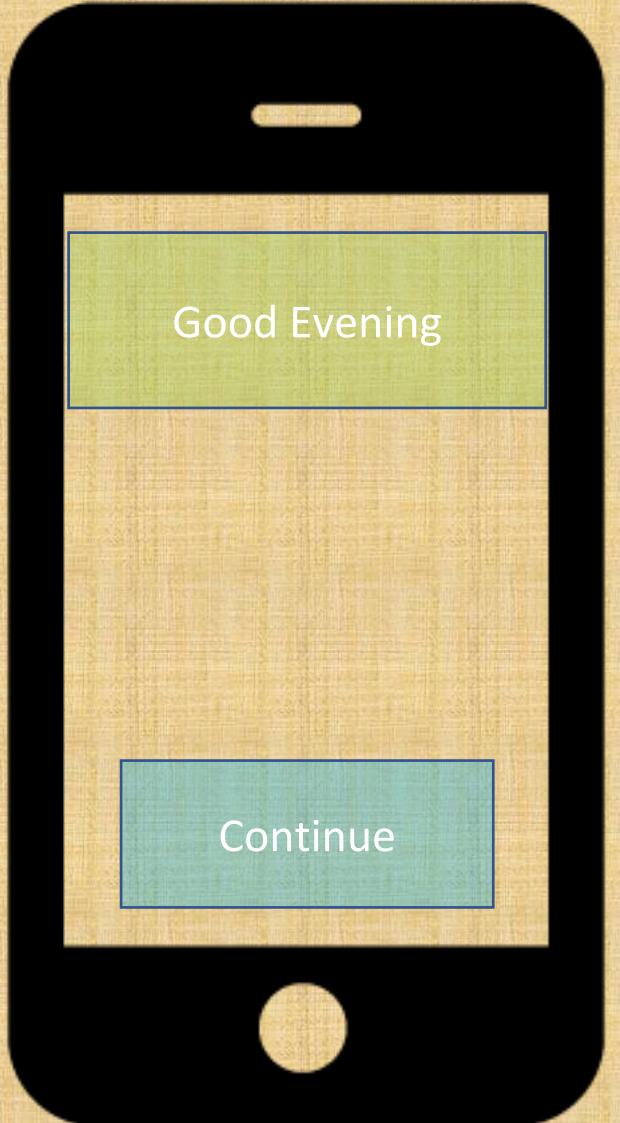
On-demand resources are resources—such as images and sounds—that you can tag with keywords and request in groups, by tag. The store hosts the resources on Apple servers and manages the downloads for you. On-demand resources enable faster downloads and smaller app sizes, improving the first-time launch experience. For example, a game app may divide resources into game levels and request the next level of resources only when the app anticipates that the user will move to that level. Similarly, the app can request In-App Purchase resources only when the user buys the corresponding in-app purchase.

Auto layout – Why it matters?



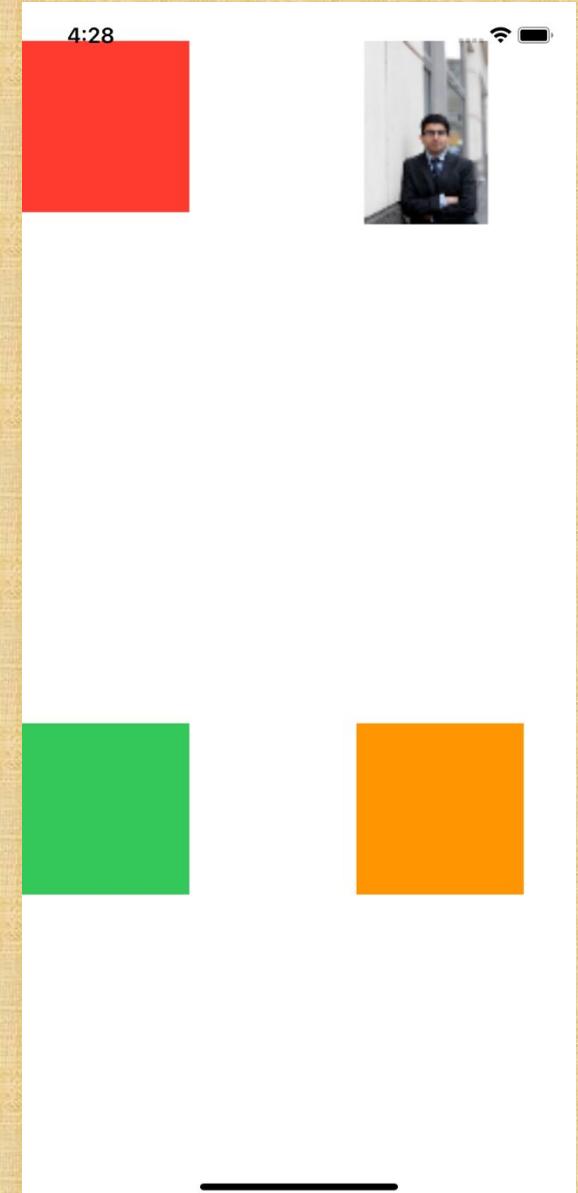
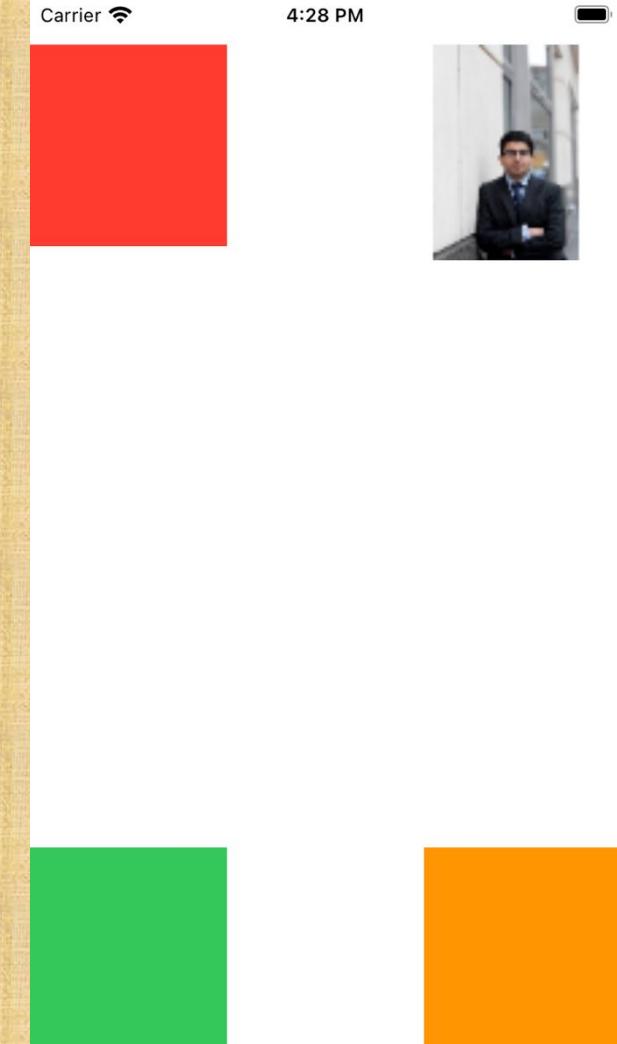


How does it impact



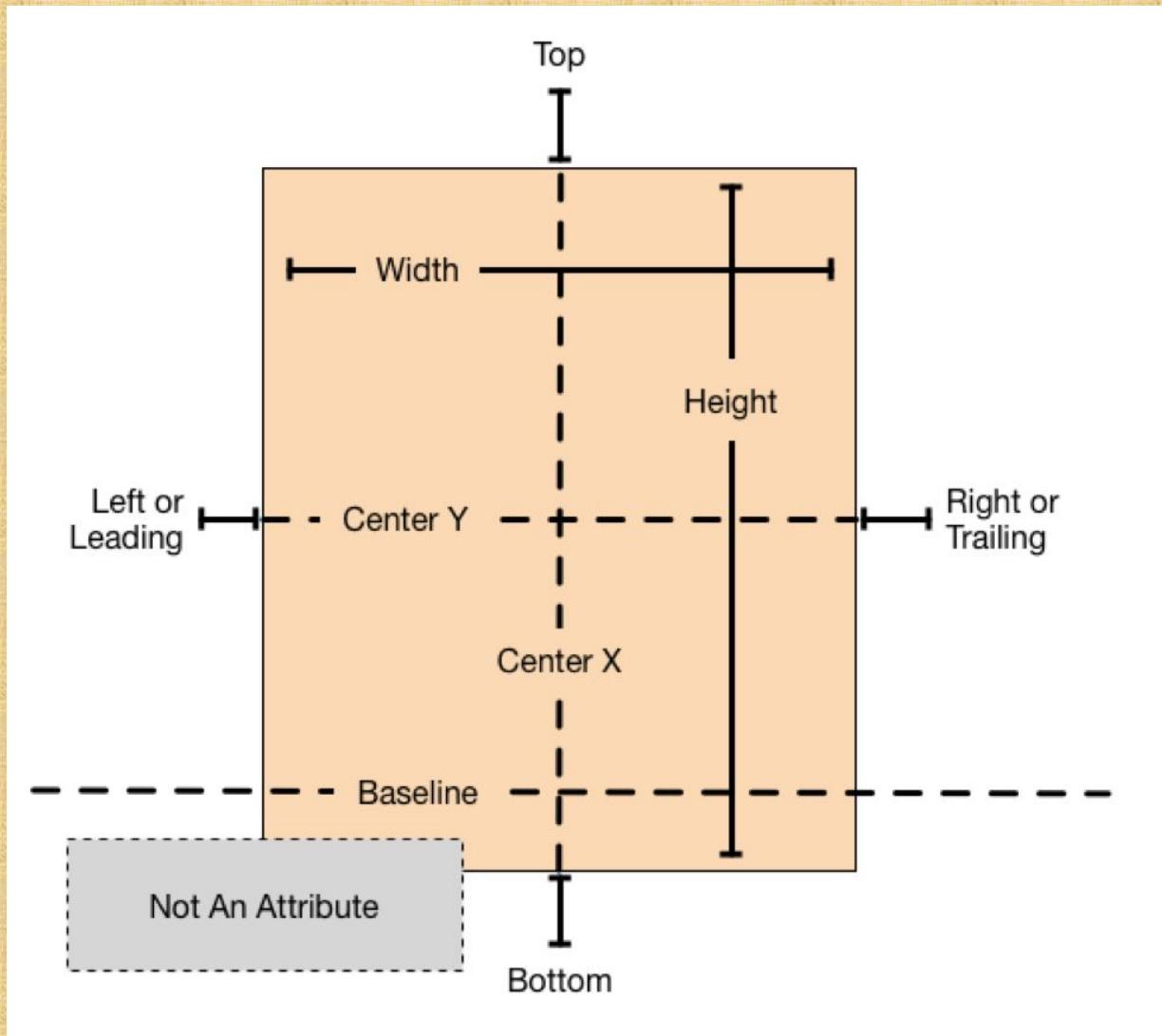
and landscape
layout

Let's try with a corner example





Constraints

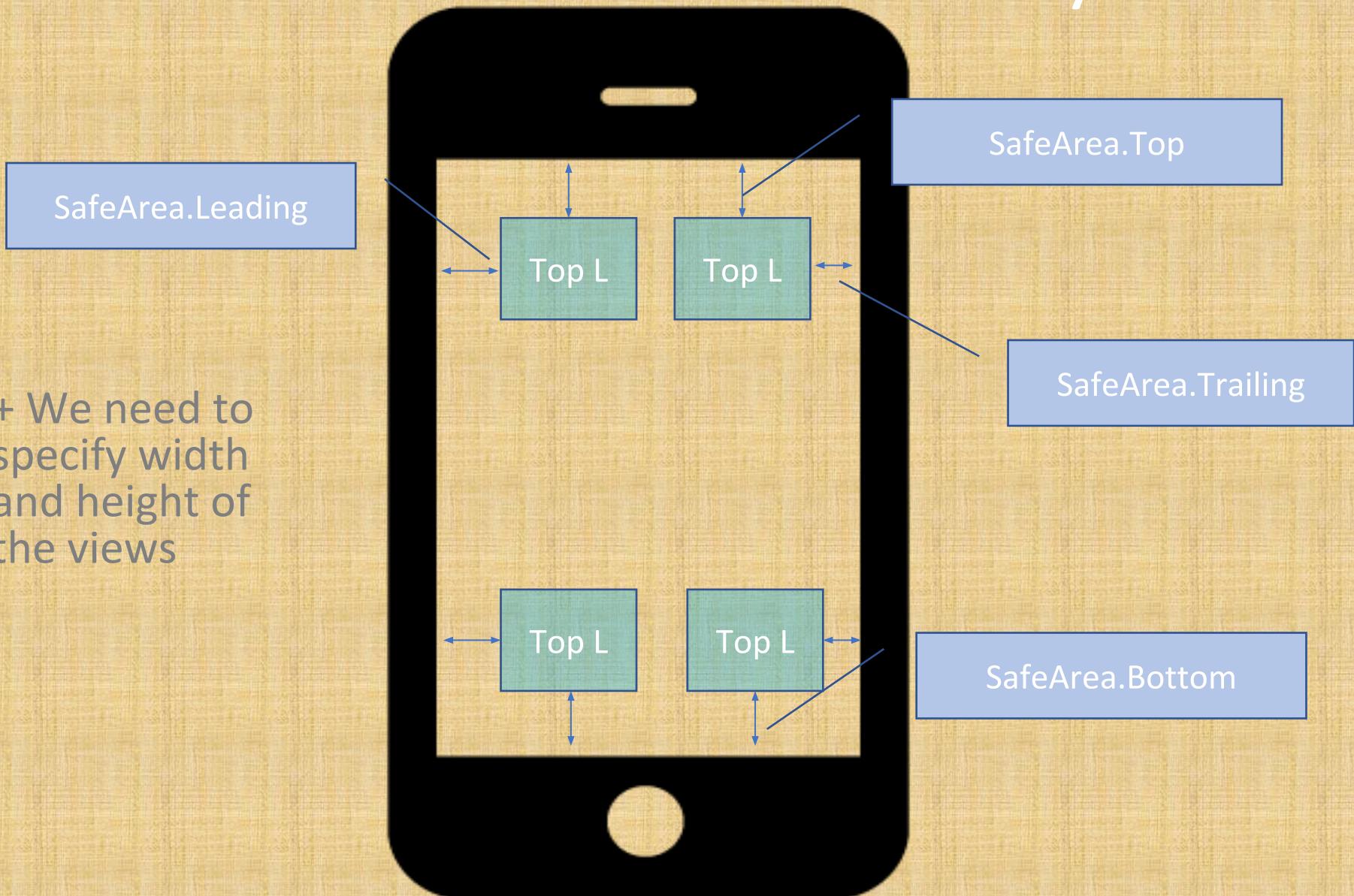


Define relationship
between container
and in between
controls

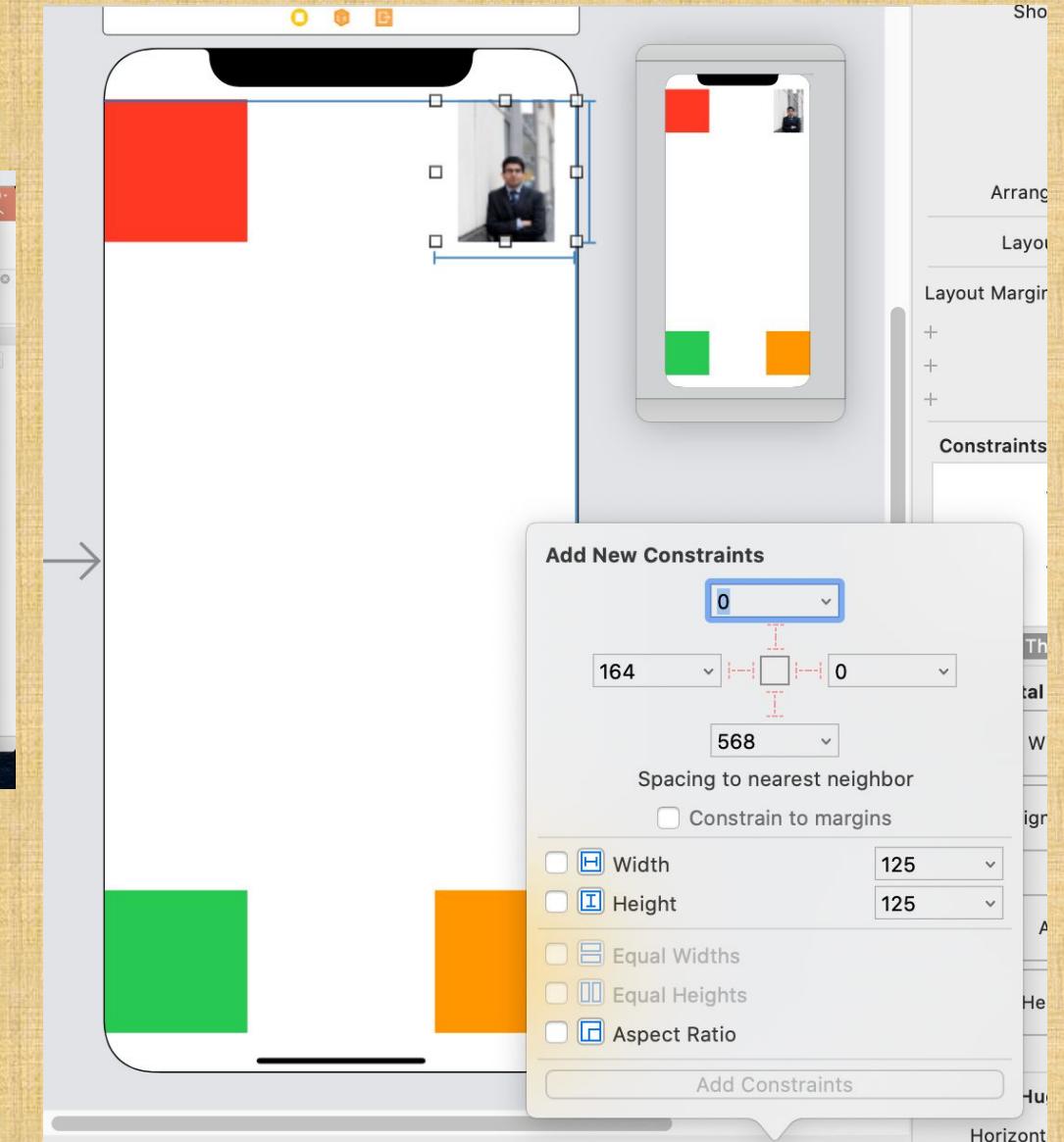
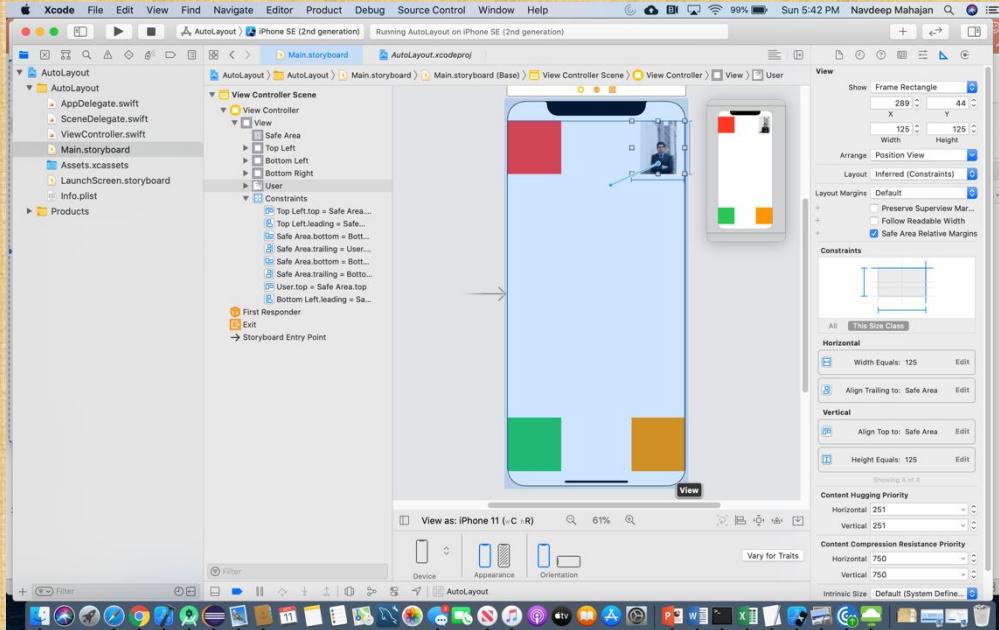
Now let's see how we should lay it out



+ We need to specify width and height of the views



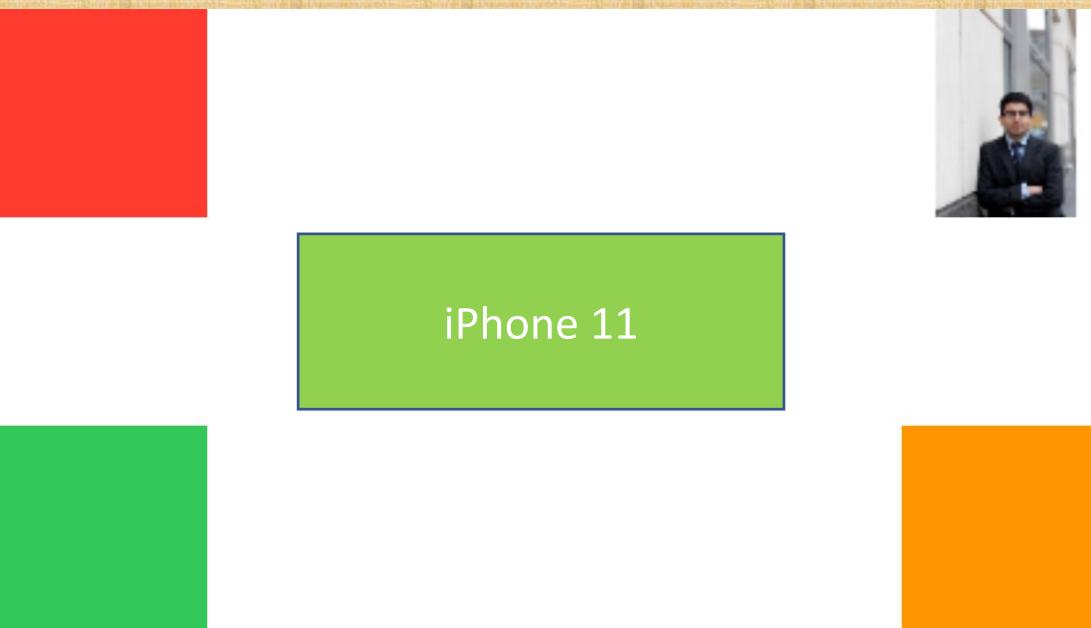
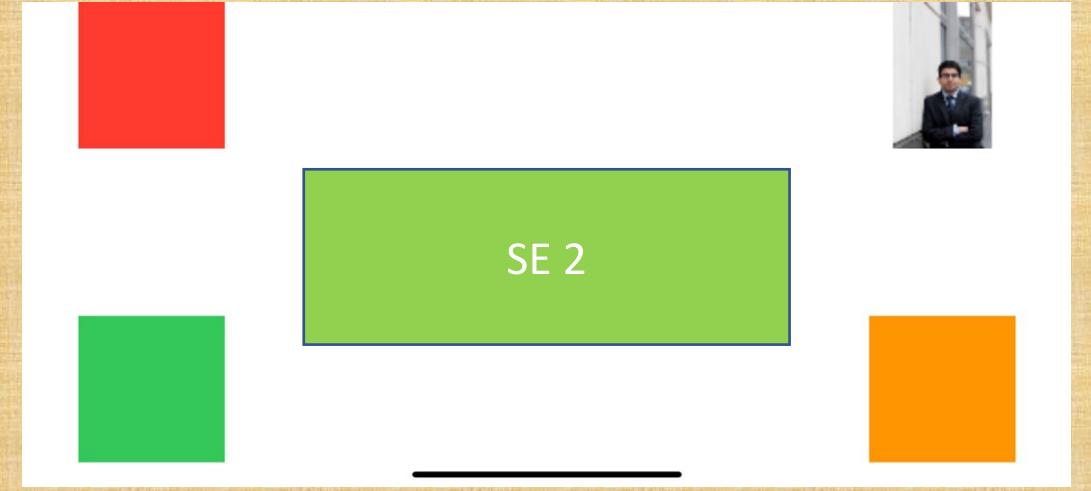
How to add constraints



Post Constraints



Post Constraints - Landscape



Read more

[Apple Cookbook for AutoLayout - Click Here](#)



Documentation Archive Developer

Auto Layout Guide Search Documentation Archive On This Page ▾

Getting Started

Understanding Auto Layout

- Auto Layout Without Constraints
- Anatomy of a Constraint
- Working with Constraints in Interface Builder

Auto Layout Cookbook

Debugging Auto Layout

Advanced Auto Layout

Appendix

Understanding Auto Layout

Auto Layout dynamically calculates the size and position of all the views in your view hierarchy, based on constraints placed on those views. For example, you can constrain a button so that it is horizontally centered with an Image view and so that the button's top edge always remains 8 points below the image's bottom. If the image view's size or position changes, the button's position automatically adjusts to match.

This constraint-based approach to design allows you to build user interfaces that dynamically respond to both internal and external changes.

External Changes

External changes occur when the size or shape of your superview changes. With each change, you must update the layout of your view hierarchy to best use the available space. Here are some common sources of external change:

- The user resizes the window (OS X).

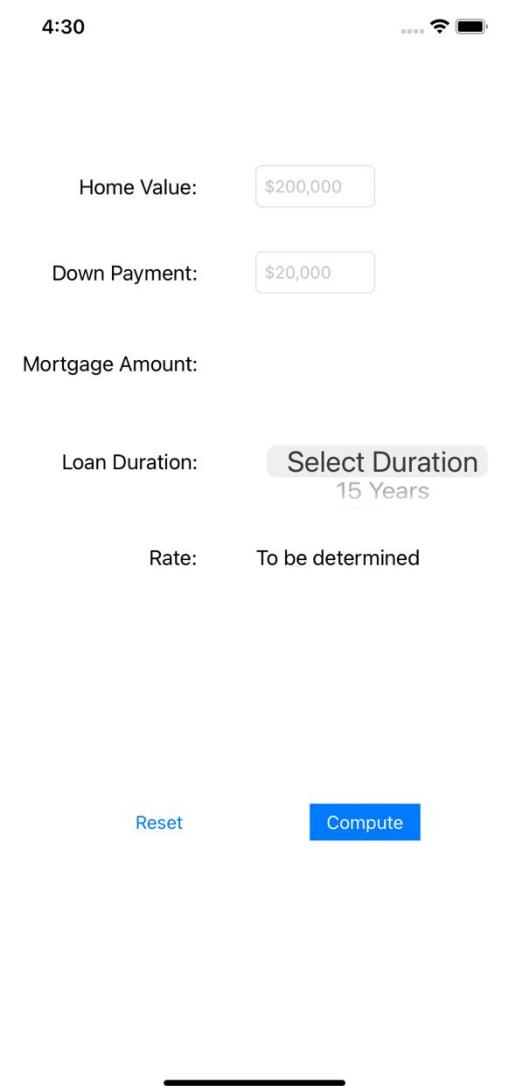


Parting Notes

Exercise:

Let's take our Mortgage calculator and do the following:

- Lock in Portrait Only
- Add Scroll View to Mortgage Calculator
- Use Auto layout and constraints to fix display on multiple form factors



A screenshot of an iPhone displaying a mortgage calculator application. The screen shows the following input fields and settings:

- Home Value: \$200,000
- Down Payment: \$20,000
- Mortgage Amount: (not explicitly shown)
- Loan Duration: Select Duration (15 Years)
- Rate: To be determined

At the bottom right are two buttons: "Reset" and "Compute". The status bar at the top indicates the time is 4:30.