张靖元 Jingyuan “Knight” Zhang

Mobile Designer & Developer

CocoaHeads Shanghai 2017-04-27

Profile

Portfolio

Who am I ?

- 张靖元 Jingyuan “Knight” Zhang
- Fudan University Shanghai Institute of Visual Art - Visual Communication Design
- Apple Retail - Creative trainer
- Full Sail University - Mobile Development
- Freelance Mobile Developer

“Simple techniques that make Xcode Storyboard more intuitive.”

CocoaHeads 2017-02-23

Recap Link

“Simple Techniques That Make Your User Interfaces More Flexible.”

Tonight

What to share tonight ?

- Make UI/UX flexible for different layout situations
- Strategies & tools
- How to get started
- Inspiration & comfort for creative UI/UX

Thanks for your support !

Thanks for your support !

Antoine
Cœur

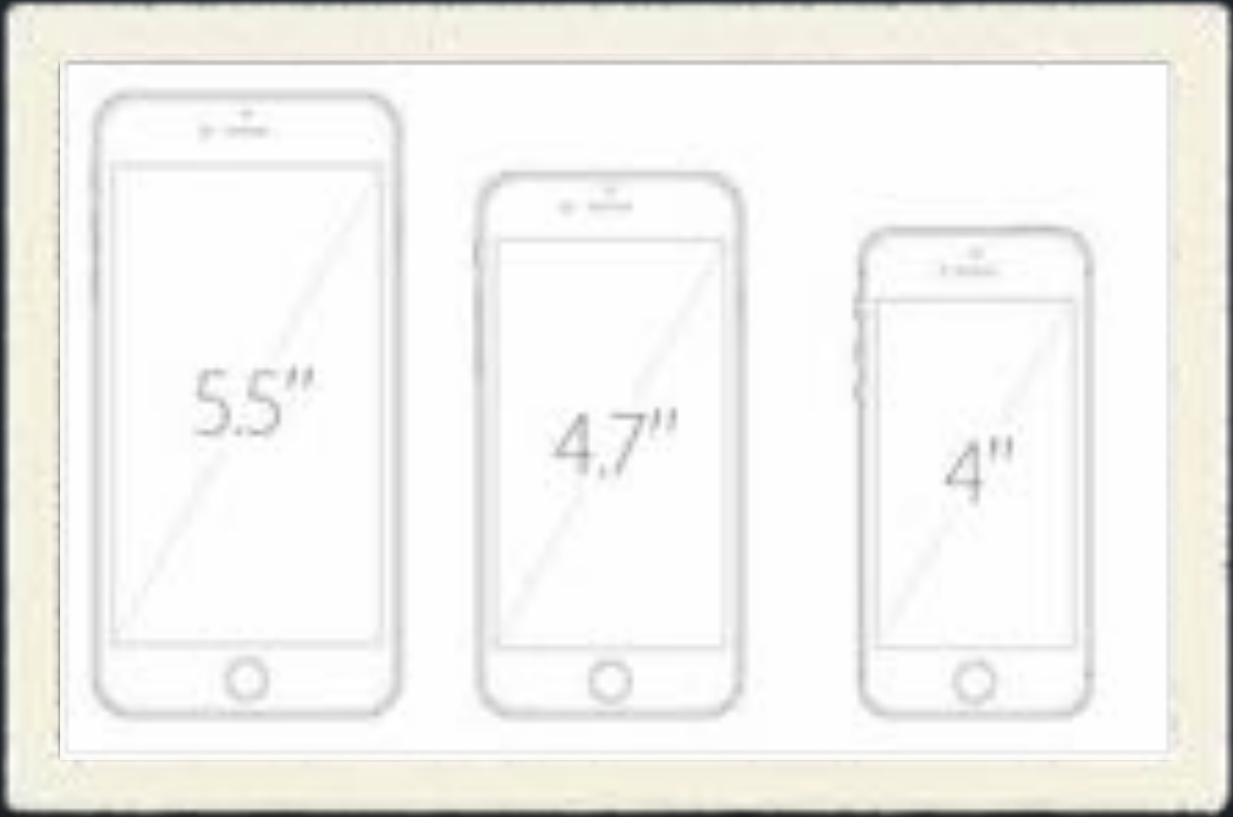
Guanshan
Liu

“Why to make the app’s UI/UX flexible for different layout situations?”

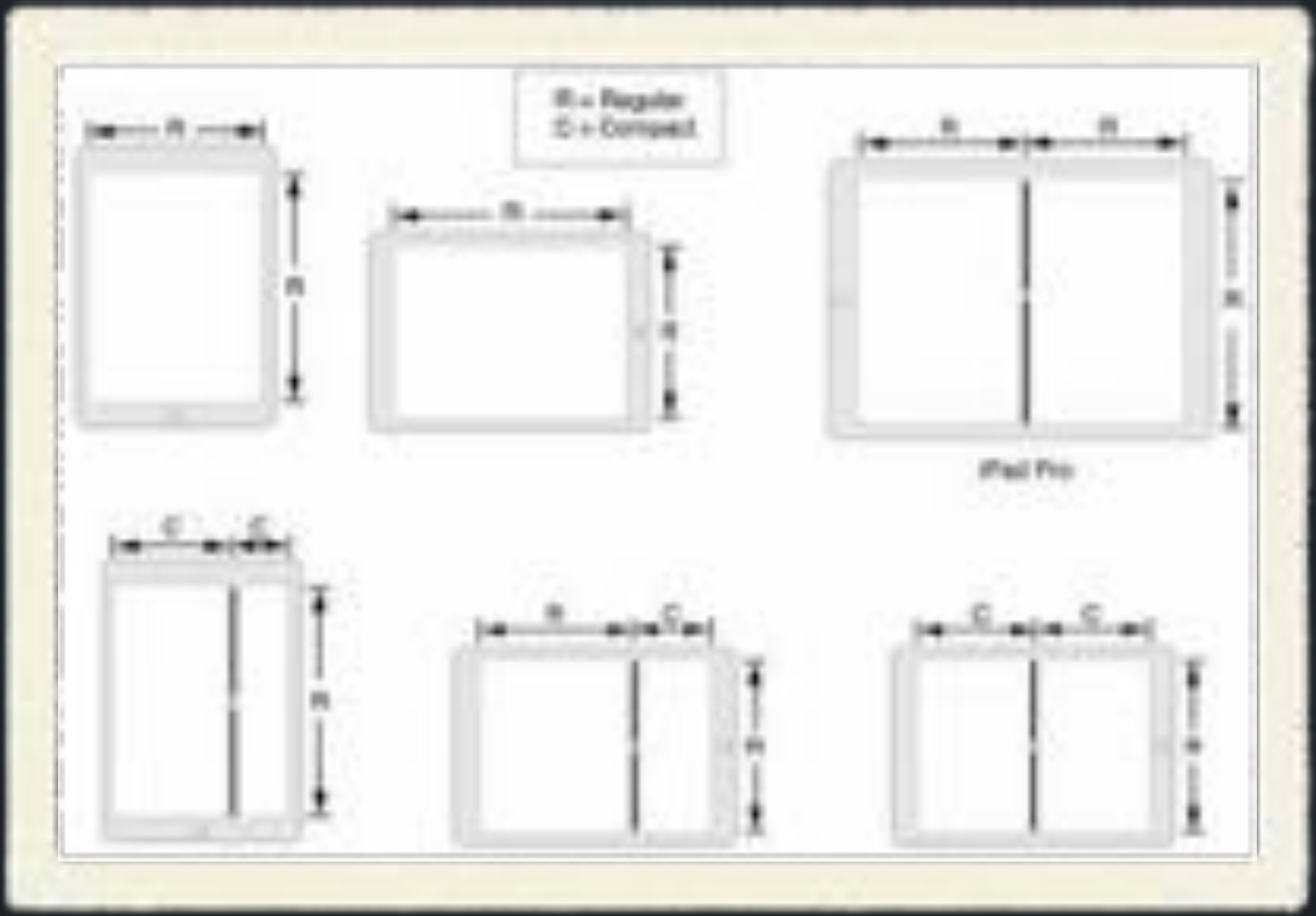
Why to make the app's UI/UX flexible for different layout situations?

- Expected UI/UX on multiple screen-sized devices
- Efficient development
- Easy to scale & manage for the future

Expected UI/UX on multiple screen-sized devices



Source: Everythingicafe



Source: Apple

Efficient development

- More Intuitive, Less Code

Easy to scale & manage for the future

- Less Revision, Faster Update

How to make the UI/UX development so flexible?

How to make the UI/UX development so flexible?

- Use Xcode Interface Builder

What do we use ?

Auto-layout | **StackView** | **Adaptive Layout (Size-Class-Specific Layout)**

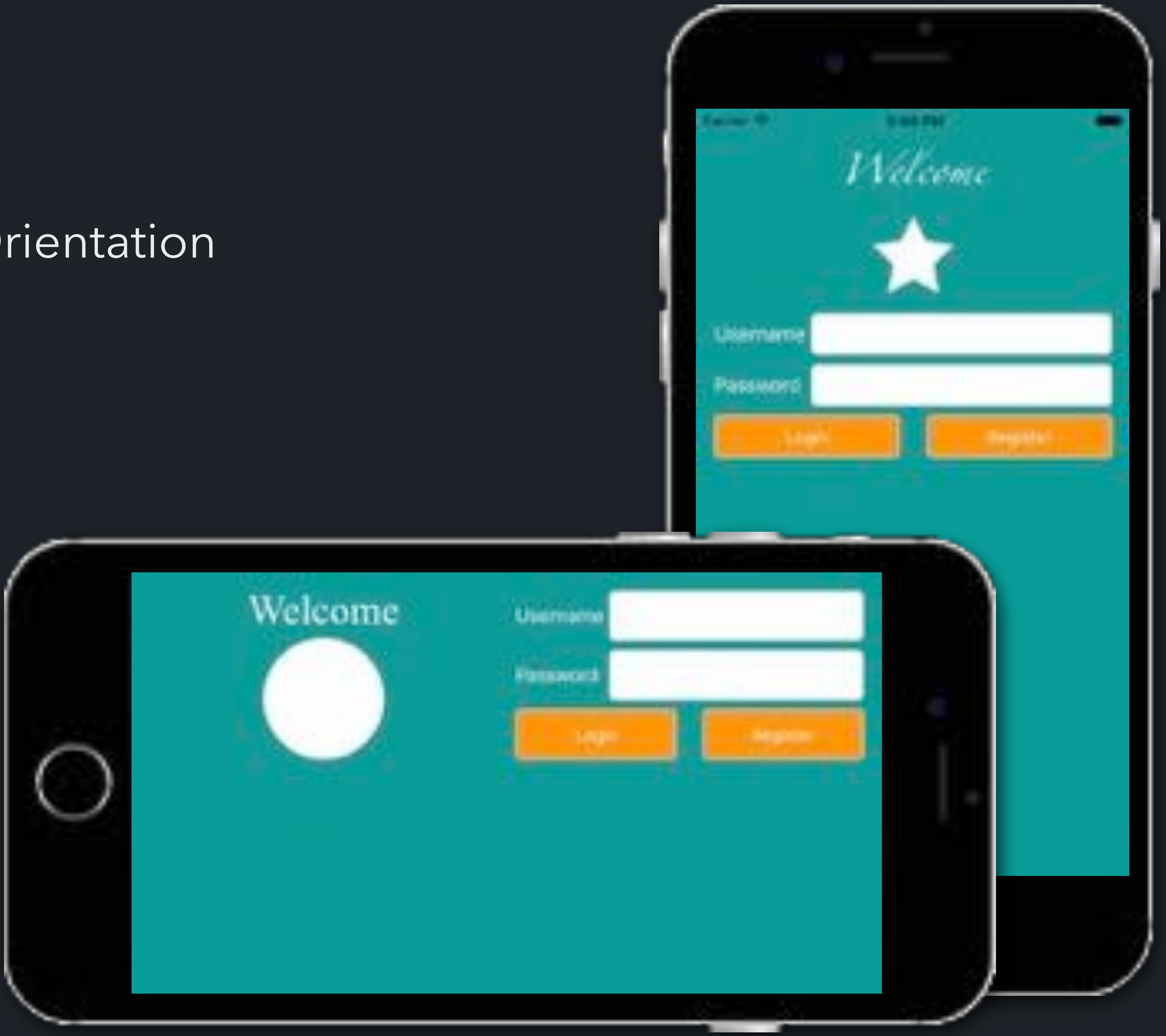
“Auto Layout dynamically calculates the size and position of all the views in your view hierarchy, based on constraints placed on those views.” – Apple

“Stack views are a powerful tool for quickly and easily designing your user interfaces. Their attributes allow a high degree of control over how they lay out their arranged views.” – Apple

“Size classes … provide a rough indication of the element’s size. Interface Builder lets you customize many of your layout’s features based on the current size class.” – Apple

A simple challenge

- Generic login view
- Different Layout each Orientation
- 3 Labels
- 2 Text Views
- 1 Image View
- 2 Button



A simple challenge of Generic Login View

- **StackView**
  - Alignment
  - Distribution
  - Spacing

- **Adaptive Layout**
  - Set attributes based on the “Size-class”
  - Set constraint by “Varying for Size-class Traits”
  - Set image based on the “Size-class” in Xcode Assets.

Start from the basic

- Putting Items on the Canvas
- Set the color
- Set custom class if needed

StackView

- **"WelcomeStack"**
  - Welcome Label
  - ImageView

StackView

- **“InputStackCell”**
  - Label & TextView
  - Buttons

StackView

- **“InputStack”**
    - 3 sets of “InputStackCell”

StackView

- **“MasterStack”**
  - “WelcomeStack”
  - “InputStack”

Auto-layout

- **Non-ambiguous & Satisfiable**
  - Width and Height
  - Horizontal and Vertical Position

Auto-layout

- **Pin panel**

Auto-layout

- **Pin panel**
  - “Constrain to margins”

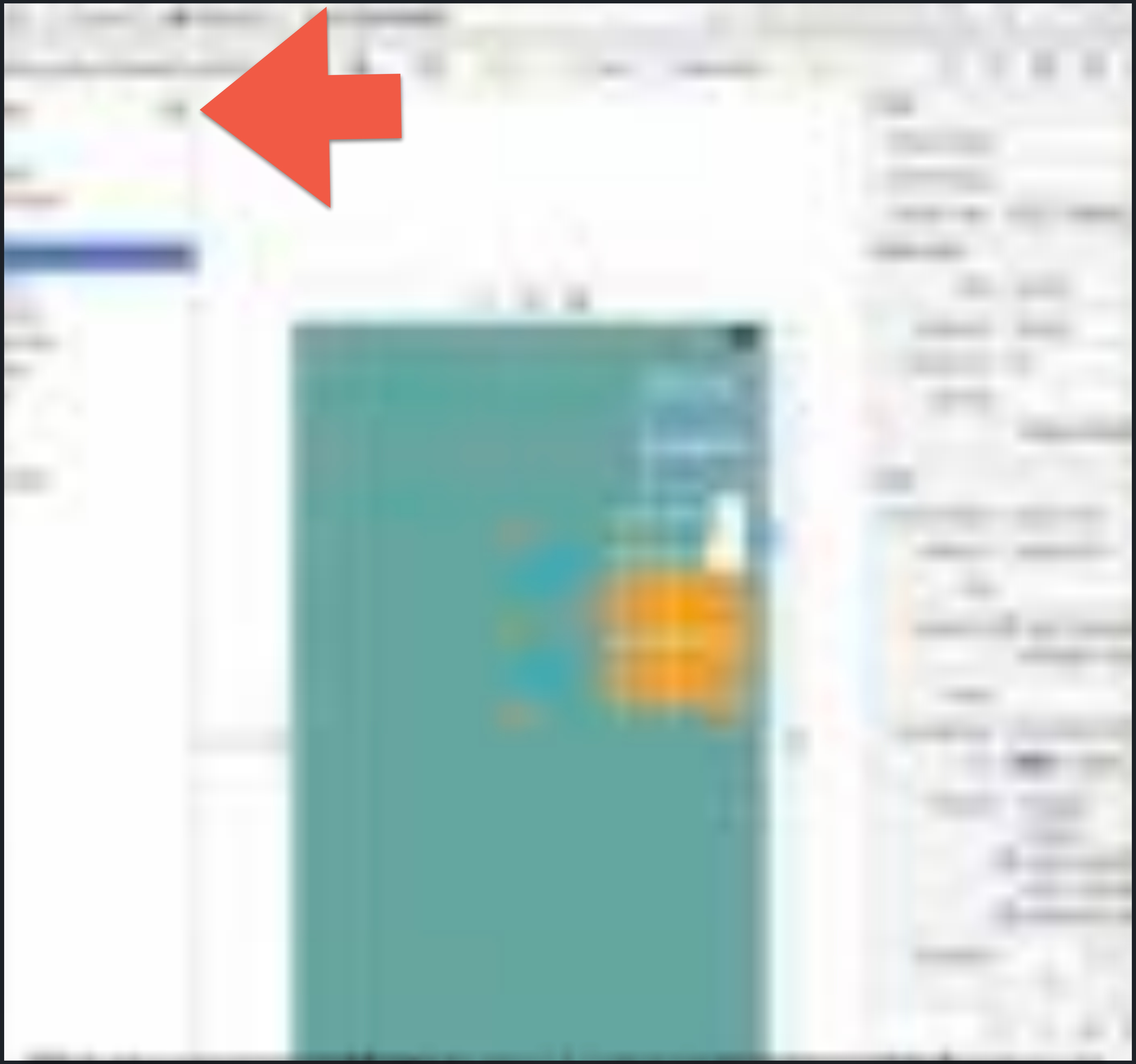
Auto-layout

- **Equal Heights**
    - “MasterStack”
    - BackgroundView

Auto-layout

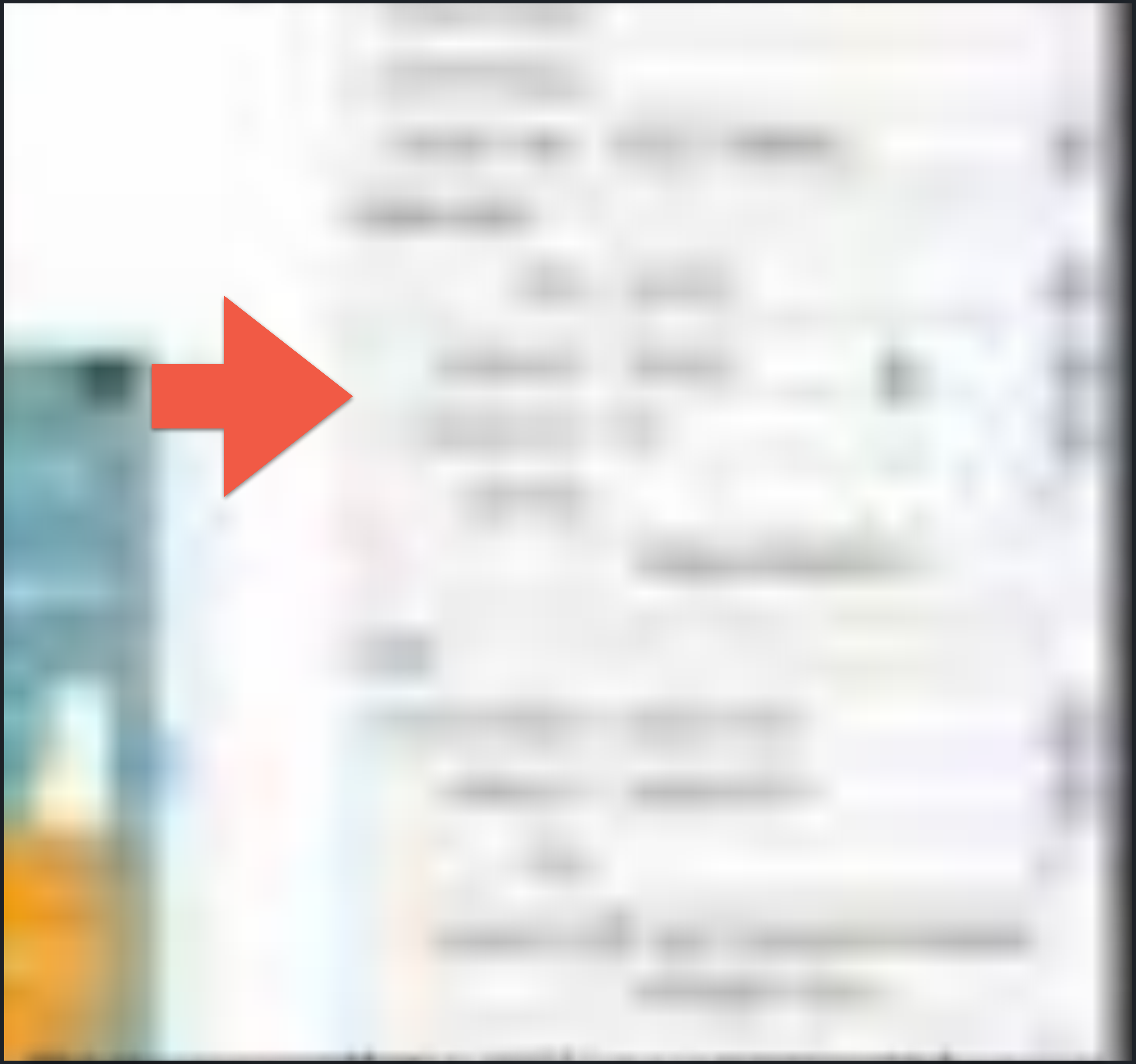
- **Proportional Height**
  - "MasterStack"
  - BackgroundView
  - Multiplier 2/5

StackView



- **Unsatisfied yet**
    - Remove the “Red Warning”

StackView



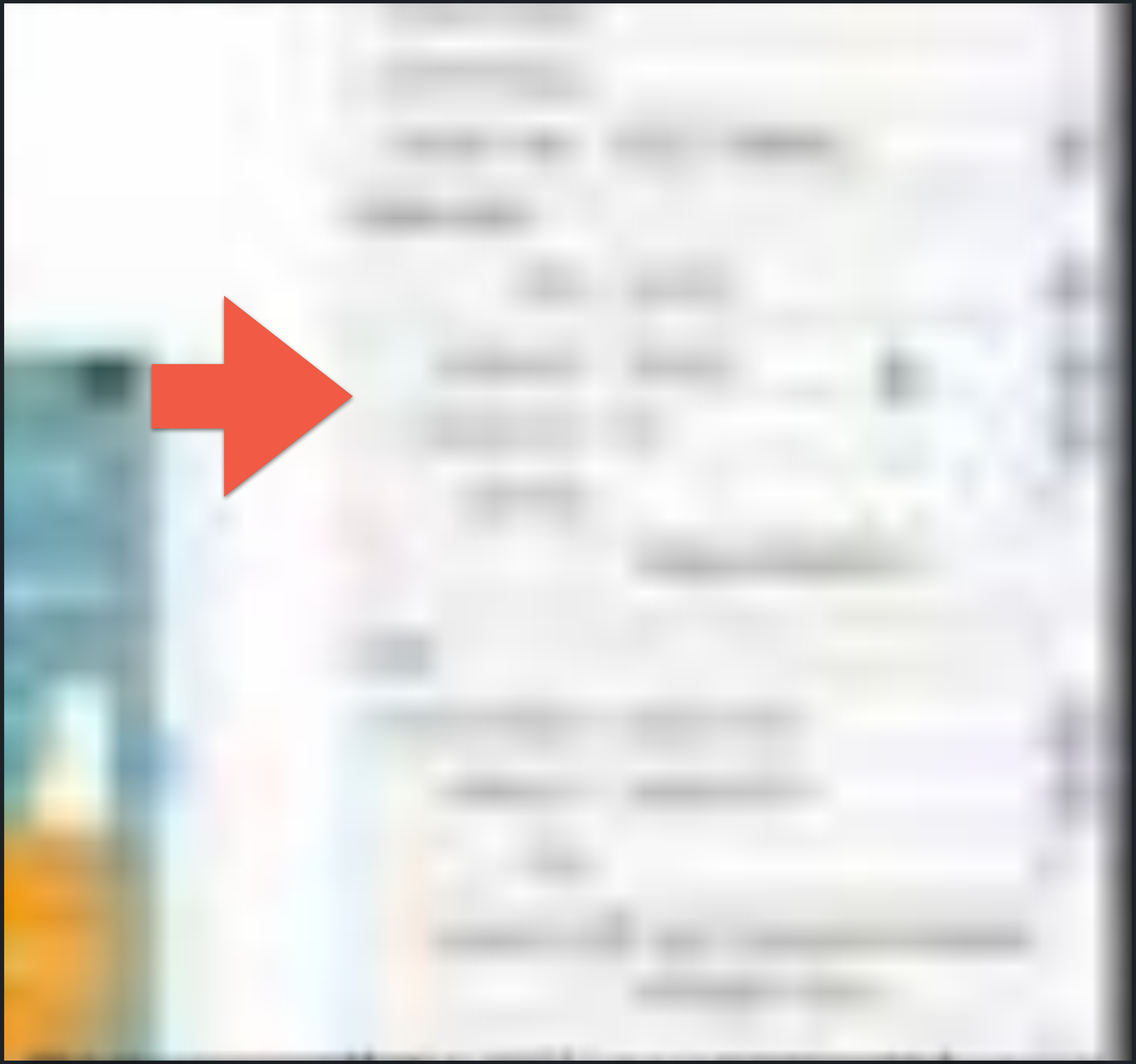
- **StackView Attributes**
  - Alignment
  - Distribution
  - Spacing

StackView

Video Demo

Count the time

StackView



- **StackView Attributes**
    - Alignment
    - Distribution
    - Spacing

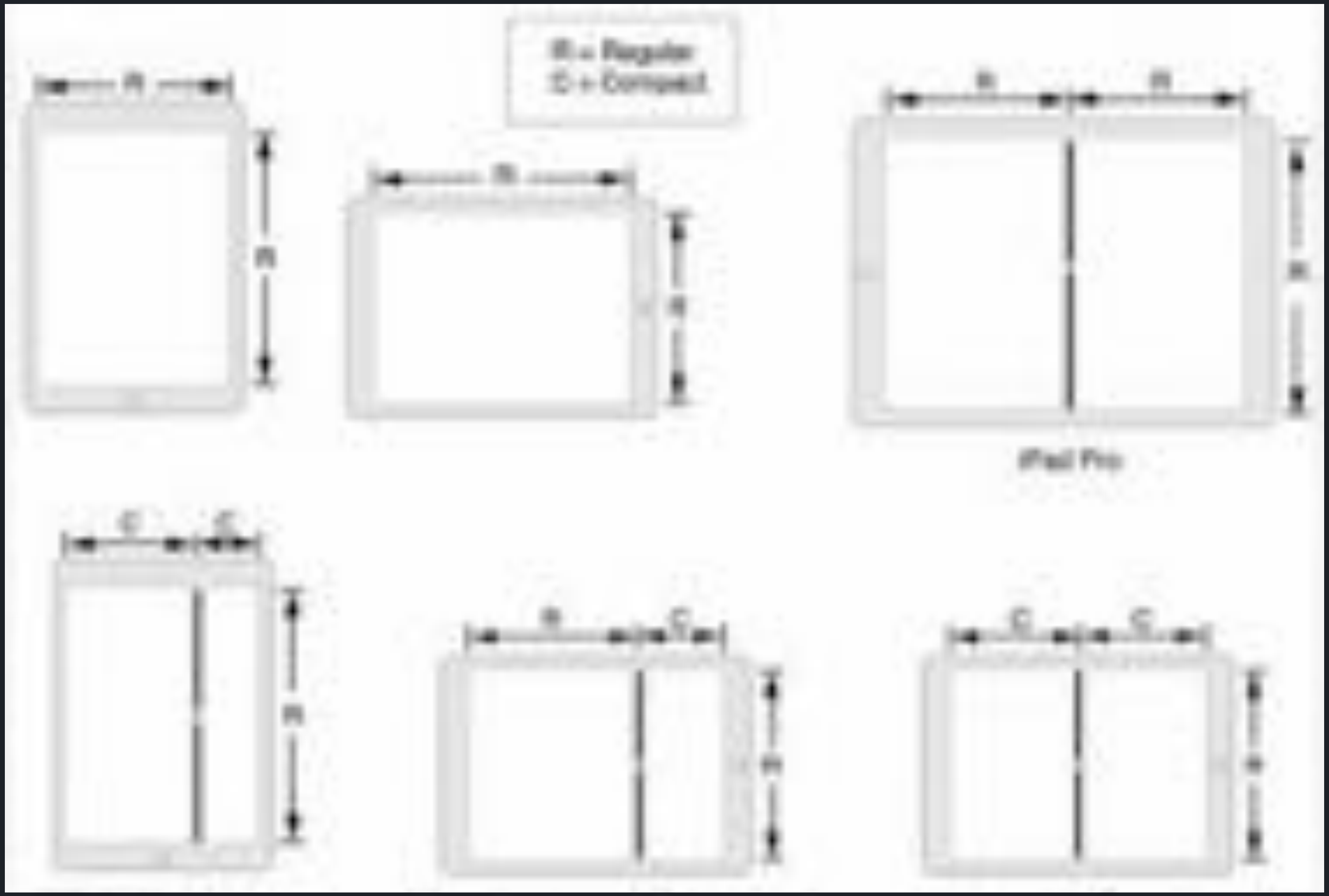
Adaptive Layout

- **Reposition**
  - “WelcomeStack”
  - “InputStack”

Adaptive Layout

- **Sizing Class**
    - Set attributes based on the "Size-class"

Sizing Class



- **Sizing Class**
  - Any
  - Regular
  - Compact

Example: iPad Sizing Class

Adaptive Layout

- **Sizing Class**
  - Add Variation

Adaptive Layout

- **Sizing Class**
    - Different attributes based on the "Size-class"

Adaptive Layout

- **Sizing Class**
  - Vary for Traits

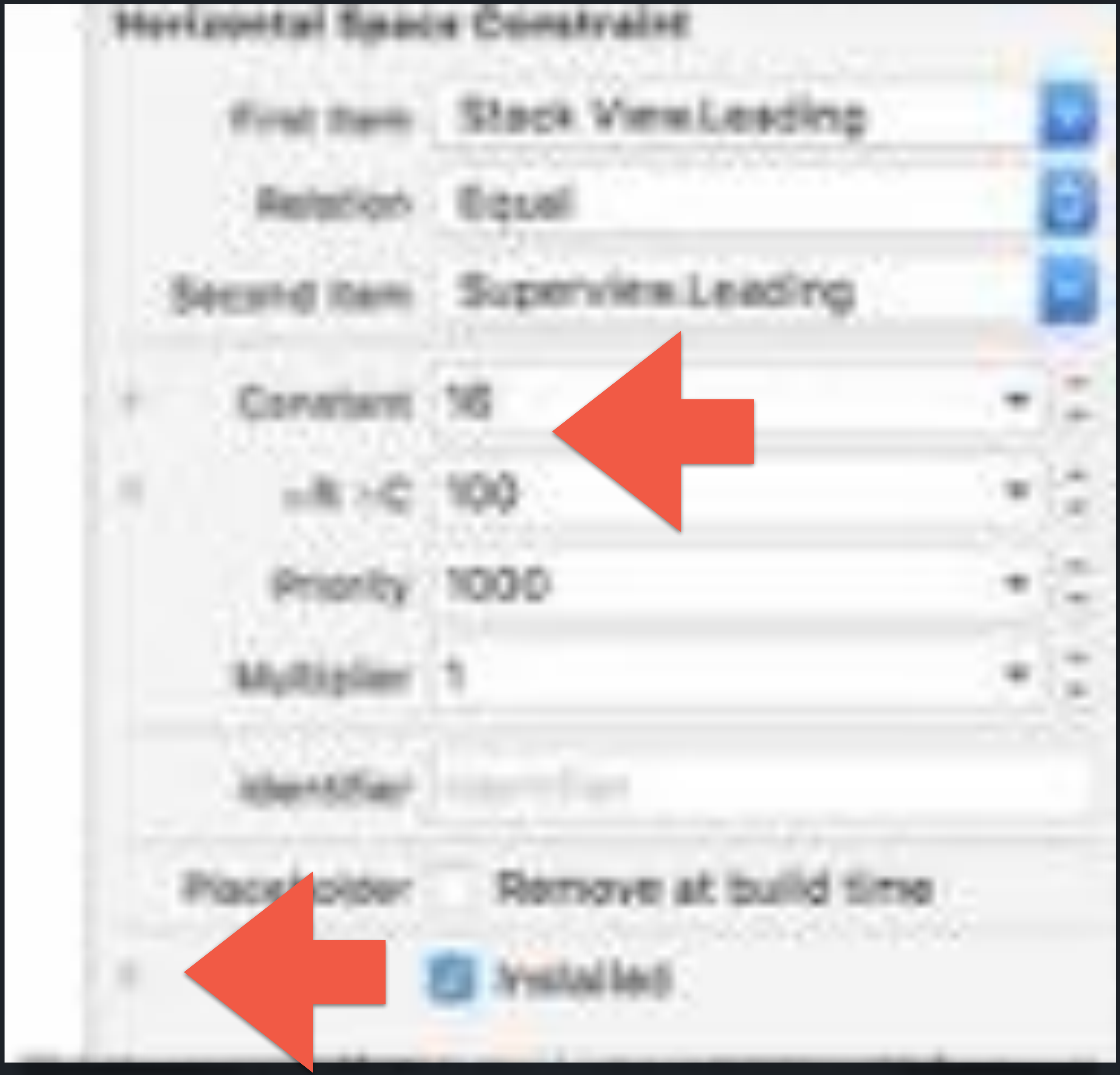
Adaptive Layout

- **Sizing Class**
  - Set constraint by "Varying for Size-class Traits"

Adaptive Layout

- **Sizing Class**
  - Set constraint by “Varying for Size-class Traits”

Adaptive Layout

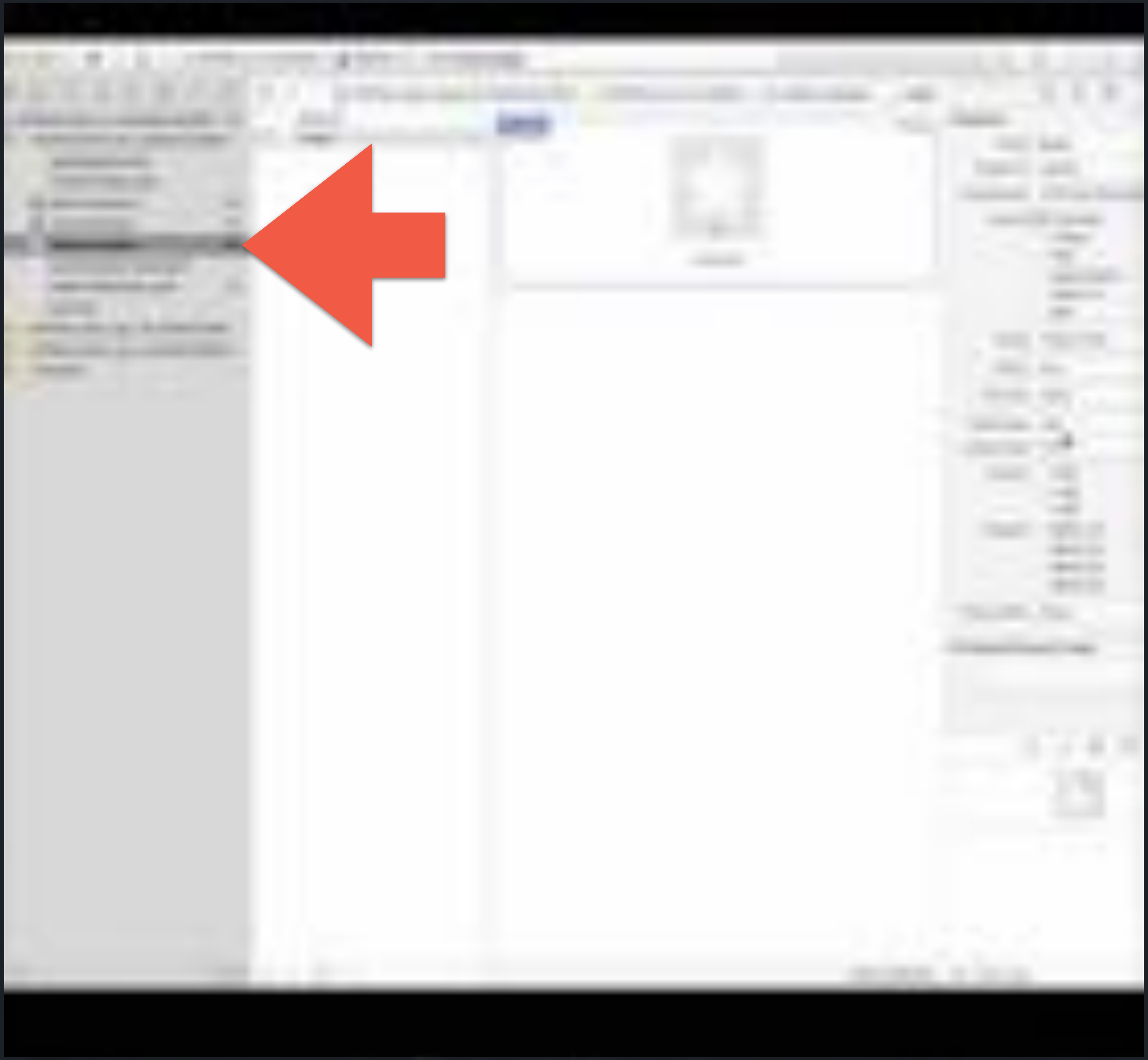


Horizontal Space Constraint

First Item	Stack View.Leading
Relation	Equal
Second Item	Superview.Leading
Constant	16
wR hC	100
Priority	1000
Multiplier	1
Identifier	Identifier
Placeholder	Remove at build time
	✓ Installed

- **Sizing Class**
  - Set constraint by "Varying for Size-class Traits"
  - Install constraint by "Varying for Size-class Traits"

Adaptive Layout



- **Sizing Class**
  - Set image based on the “Size-class” in Xcode Assets.

Adaptive Layout

- **Sizing Class**
  - Set image based on the “Size-class” in Xcode Assets.

Adaptive Layout

- **Sizing Class**
  - Set image based on the "Size-class" in Xcode Assets.

Adaptive Layout

- **Sizing Class**
  - Set image based on the "Size-class" in Xcode Assets.

Adaptive Layout

- **Sizing Class**
  - Set image based on the “Size-class” in Xcode Assets.

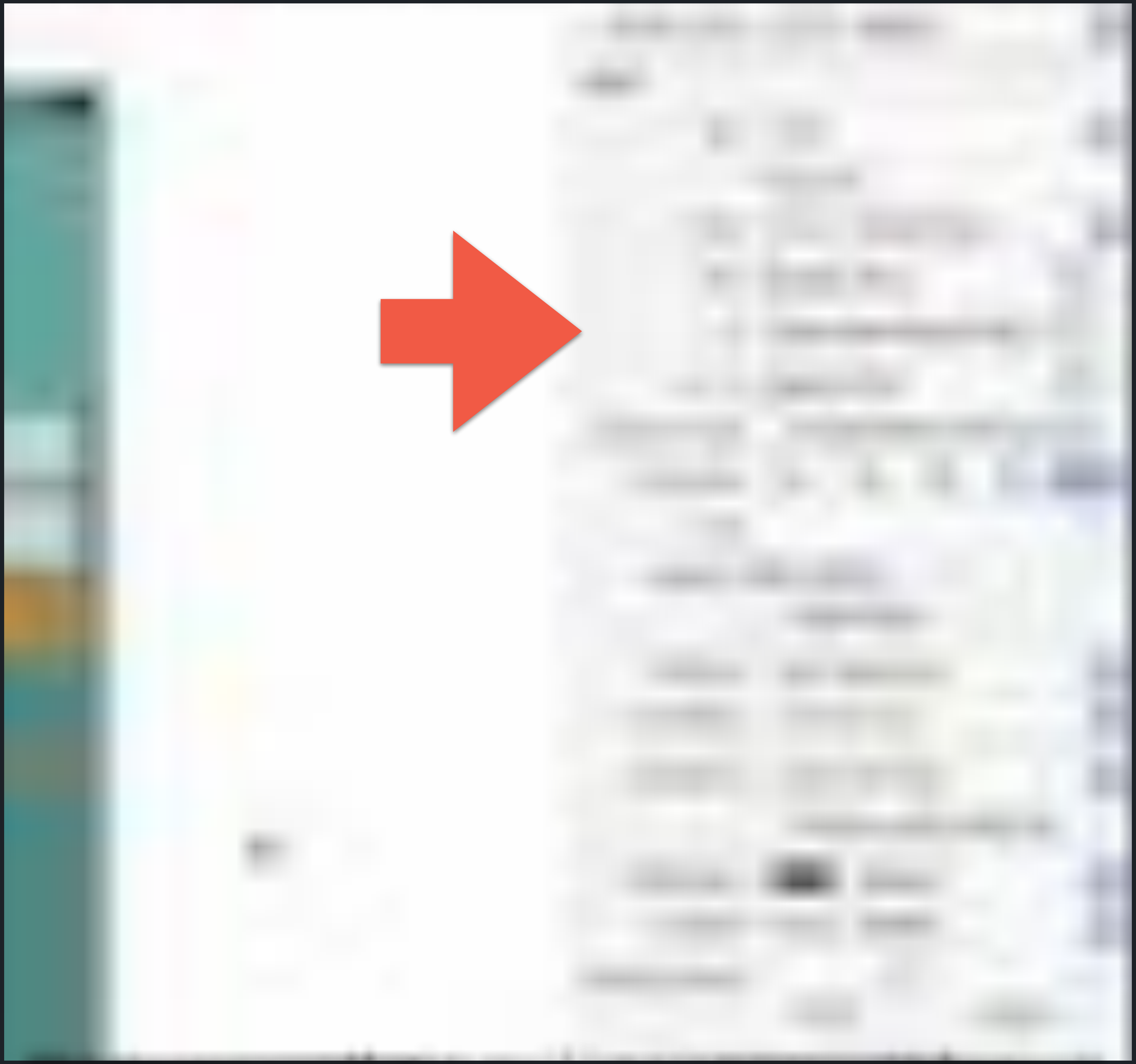
Adaptive Layout

- **Sizing Class**
  - Just set the image
  - Xcode Assets handle for you

Adaptive Layout

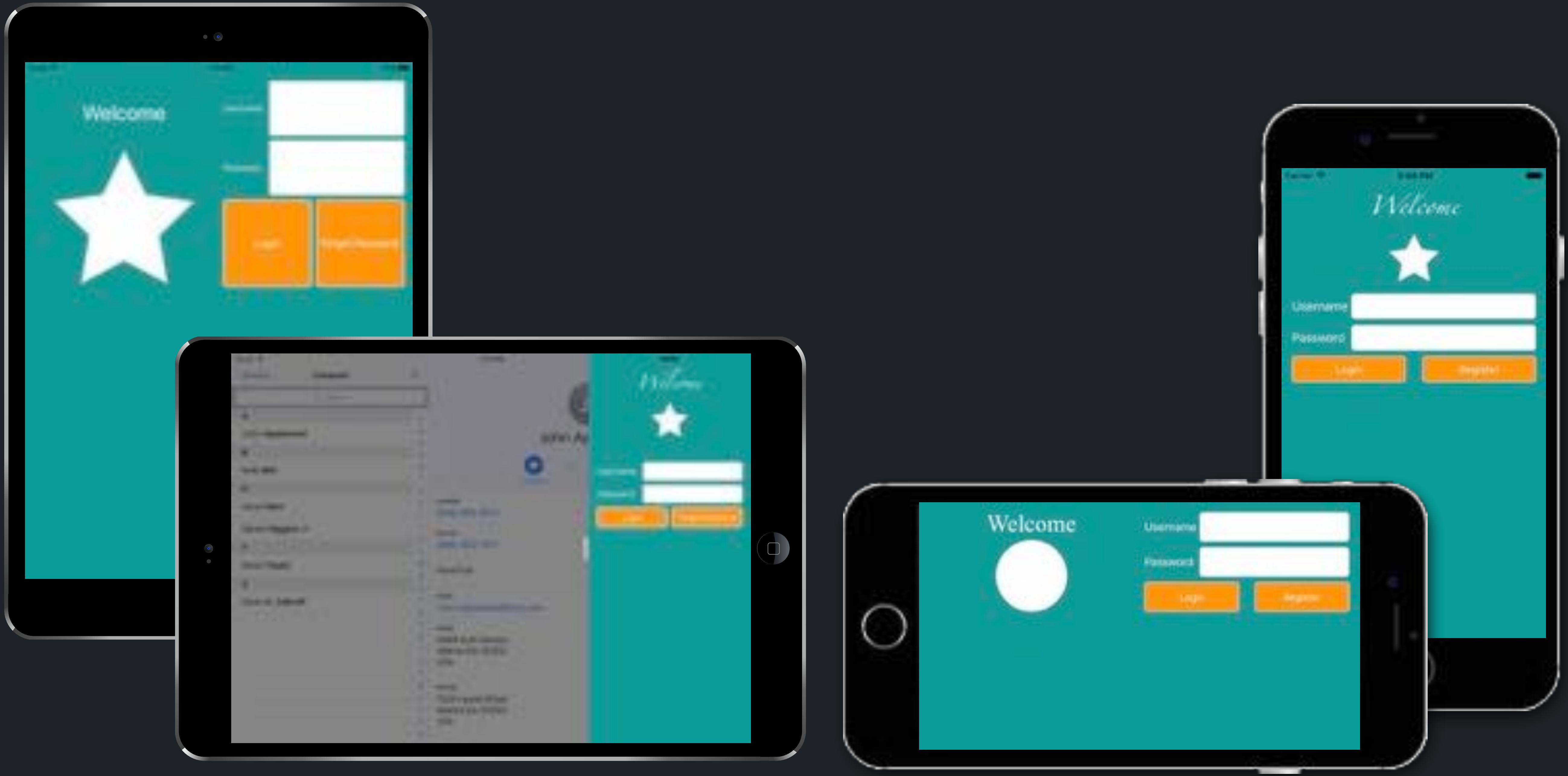
- **Sizing Class**
    - Just set the image
    - Xcode Assets handle for you

Adaptive Layout



- **Sizing Class**
  - Set attributes based on the "Size-class"
  - Be creative
  - Such fonts

Adaptive Layout



Summary

- **Auto-layout** - dynamically calculates the size and position of all the views

- **StackView** - Their attributes allow a high degree of control

- **Adaptive Layout** - customize many of your layout’s features based on the current size class

“Simple Techniques That Make Your User Interfaces More Flexible.”

Thank You

Useful Resources

张靖元 Jingyuan “Knight” Zhang

Mobile Designer & Developer

CocoaHeads Shanghai 2017-04-27

Profile

Portfolio