iOS Foundations II
Session 5

- AutoLayout
- UITextFields

Autolayout

- Constraint-based layout system used in building user interfaces.
- You can tell the system two things about every view (widget):
    1. *Size* of the widget
    2. *Location* where the widget is placed within its **super** (“parent”) view
- Once told the rules, AutoLayout will enforce the rules.
- Rules are setup using **constraints**.

Constraints

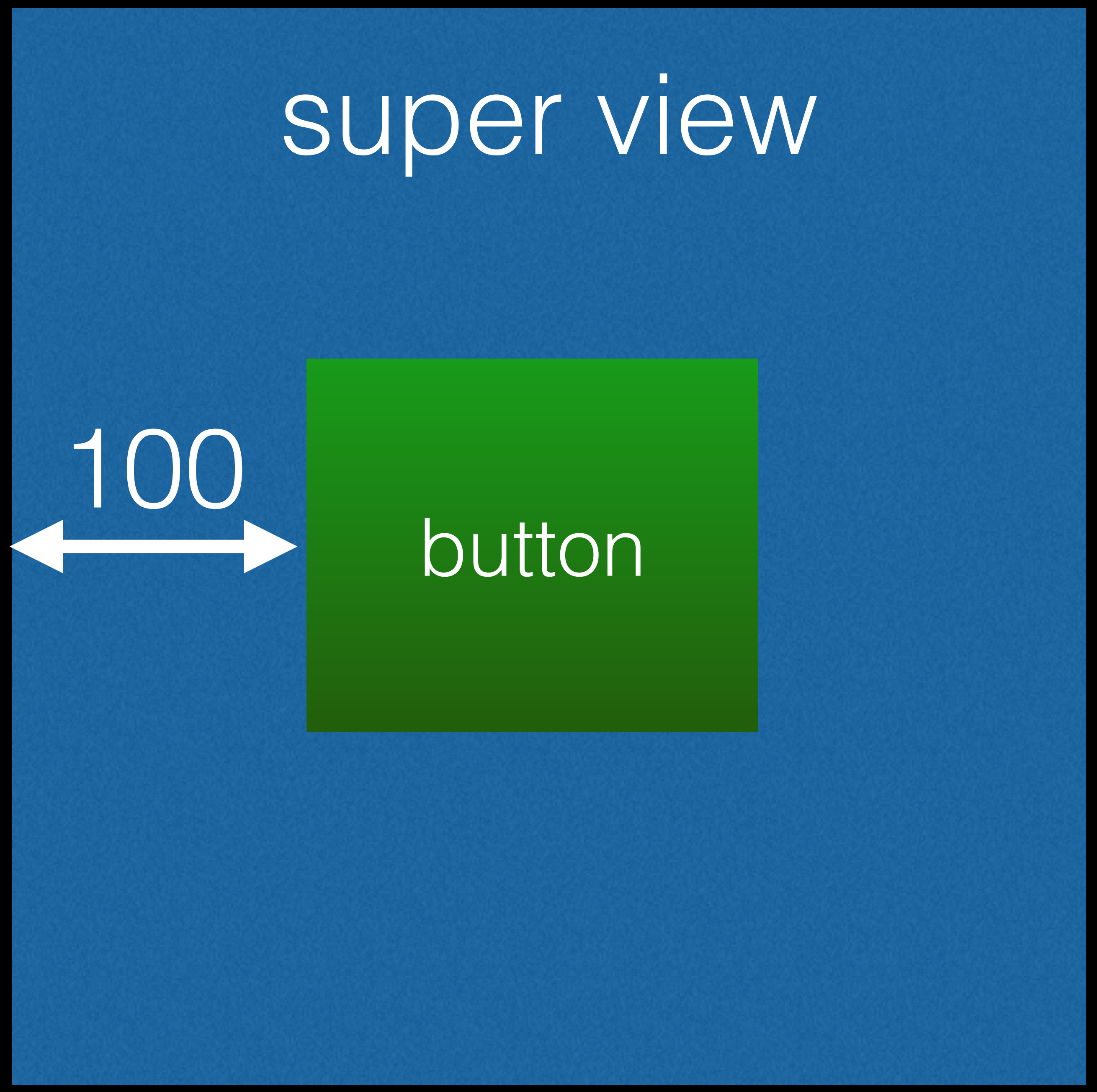
- Constraints are the fundamental building block of autolayout.
- Constraints contain rules for the layout of your UI's elements.
- You can give a 50 point height constraint to an UIImageView object, which constrains the view to always have a 50 point height (**size** constraint). And/or you can give it a constraint to always be located 20 points from the bottom of its superview (**location** constraint).
- Constraints can work together, but sometimes conflict with each other.
- At runtime, AutoLayout considers all constraints, then calculates sizes and positions that best satisfies all of the constraints.

Attributes

- When you attach constraints to a view, you attach them using the view’s attributes.
- Attributes: **left/leading, right/trailing, top, bottom, width, height, centerX, centerY**
- Attributes tell the constraint the parts of the view the constraint should manipulate.

Attributes

- E.g. A constraint of 100 points from a button's *left* attribute to its container's *left* attribute, is equivalent to saying, "I want the left side of this button to be 100 points to the right of the left side of the button's super view."



Demo 5.1:
Basic AutoLayout

Constraints + Attributes = Math time

- “…think of a constraint as a mathematical representation of a human-expressable statement”
- “The left edge should be 20 points from the left edge of its containing view” translates to

  button.left = container.left x 1.0 + 20
- This is in the form y=mx+b:
  - **First attribute = Second attribute * multiplier + constant**
- In the case of an absolute value, like pinning height, width, centerX, or centerY, the second attribute is nil.
- You can change the constants in code as an easy way to programmatically adjust your interface.

Demo 5.2:
Autolayout w/ multipliers

Storyboard and Autolayout

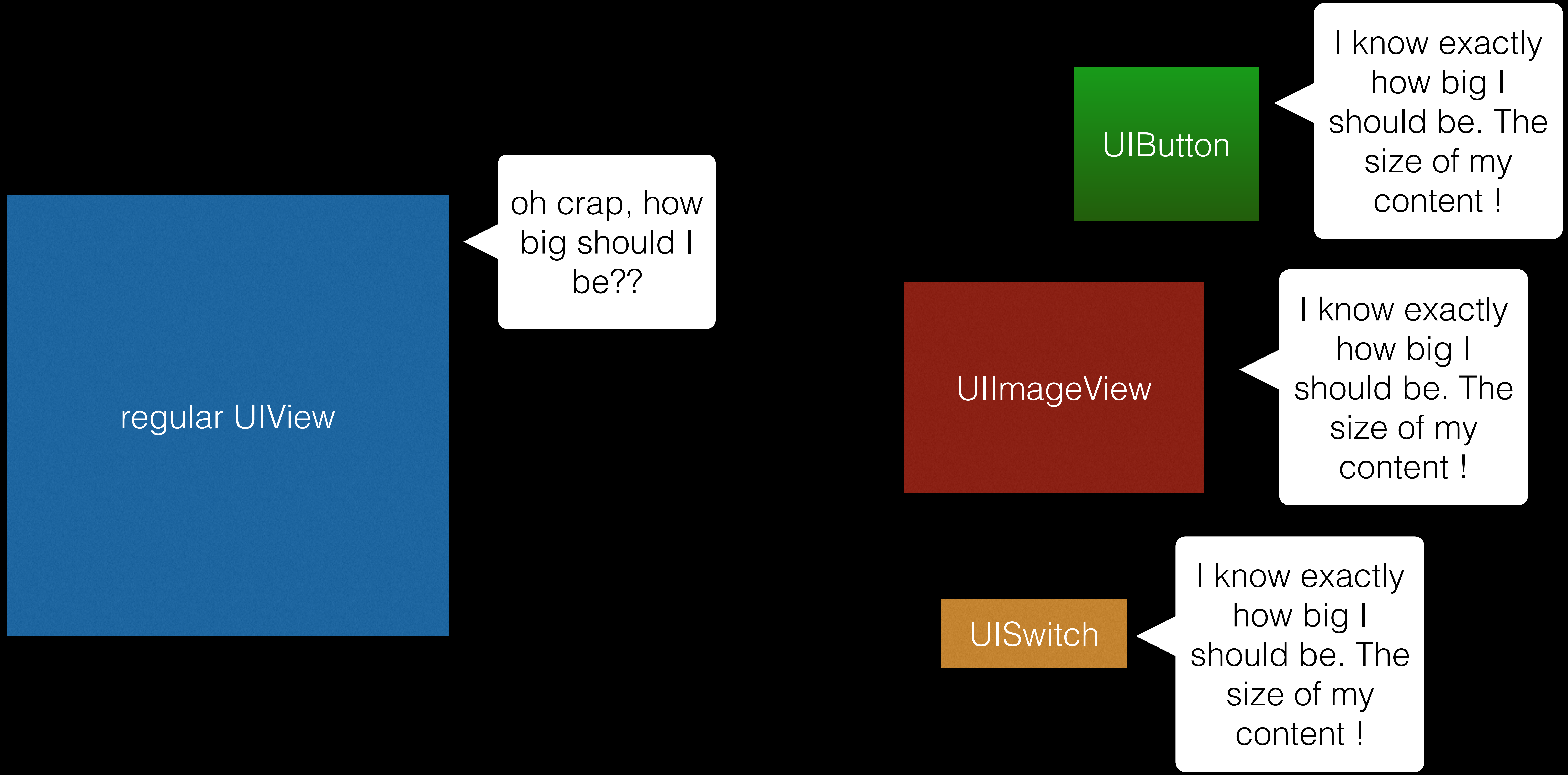
- Use storyboard!
  - It makes setting up autolayout relatively easy. You can set up autolayout using only code, but Apple strongly recommends using storyboard.
- Conflicting constraints
  - Xcode lets you build your app even if constraints conflict or are incorrect, but Apple says to never ship an app like that.
- When you drag a object onto your interface, **it starts with no constraints.**
- Once you apply one constraint, that view needs to have constraints dictating both **size AND location**

Demo 5.3: Constraints: “All or nothing”

Intrinsic Content Size

- Intrinsic content size is the minimum size a view needs to display its content.
- Its available for certain UIView subclasses:
    - UIButton & UILabel: these views are as large as they need to display their full text
    - UIImageView: image views have a size big enough to display their entire image. This can change with its content mode.
- You will know a view has an intrinsic content size if autolayout doesn't require its size to be described with constraints.

Intrinsic Content Size


regular UIView
oh crap, how big should I be??
UIButton
I know exactly how big I should be. The size of my content !
UIImageView
I know exactly how big I should be. The size of my content !
UISwitch
I know exactly how big I should be. The size of my content !

Demo 5.4:
Progress Bar:
No Inherent Width

`UITextField`

- “A `UITextField` object is a control that displays editable text and sends an action message to a target object when the user presses the return button.”
- A text field can have a delegate to handle editing related notifications.
- When a user taps into a text field, the text field becomes the first responder and it brings the keyboard on screen.
- You are responsible for making sure the text field you are editing is not covered by the keyboard.
- **A UITextField is a subclass of UIView, so it is put on screen the same as a UIView**

UITextField

- UITextfield has an intrinsic content size!
- It’s size depends on the text in it. It can have two types of text:
  - “Text” box in Xcode: Normal text (as if user typed it in)
  - “Placeholder” box inXcode: Greyed-out text, used to tell the user what info to type into the field (e.g., “Name” or “E-mail Address”). Once the user types a single character into the textfield, the placeholder text vanishes and is replaced by the character and subsequently typed characters.
- If you provide no text or placeholder, the textfield intrinsically takes a very small width of 26 points.

Demo