# Compositional UI Styling

Jason Zurita

# UI Styling Goals

Reusable

Simple to use

Discoverable

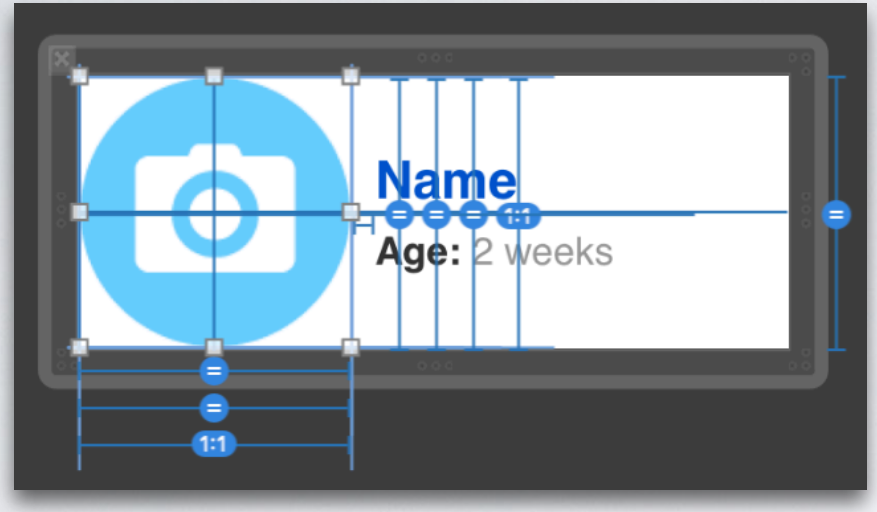
# Ways to Style UI

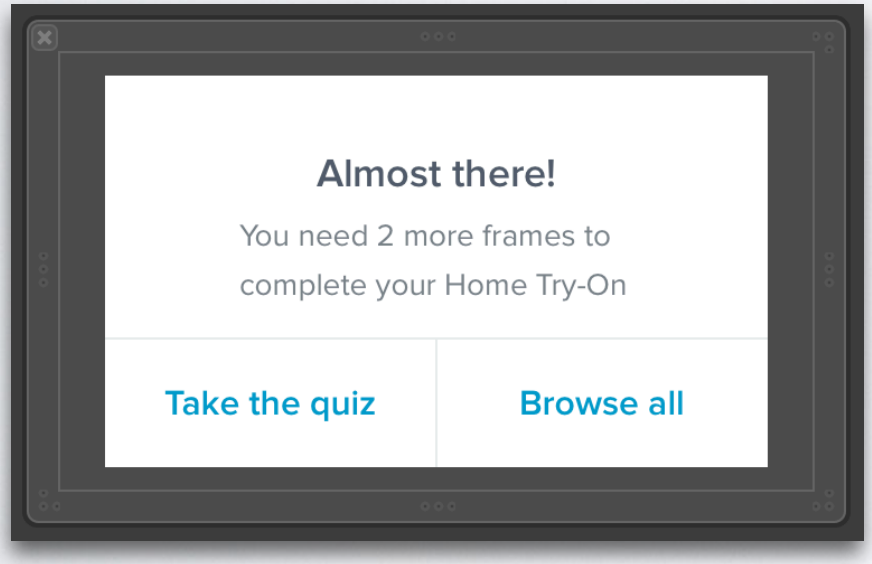
Visually

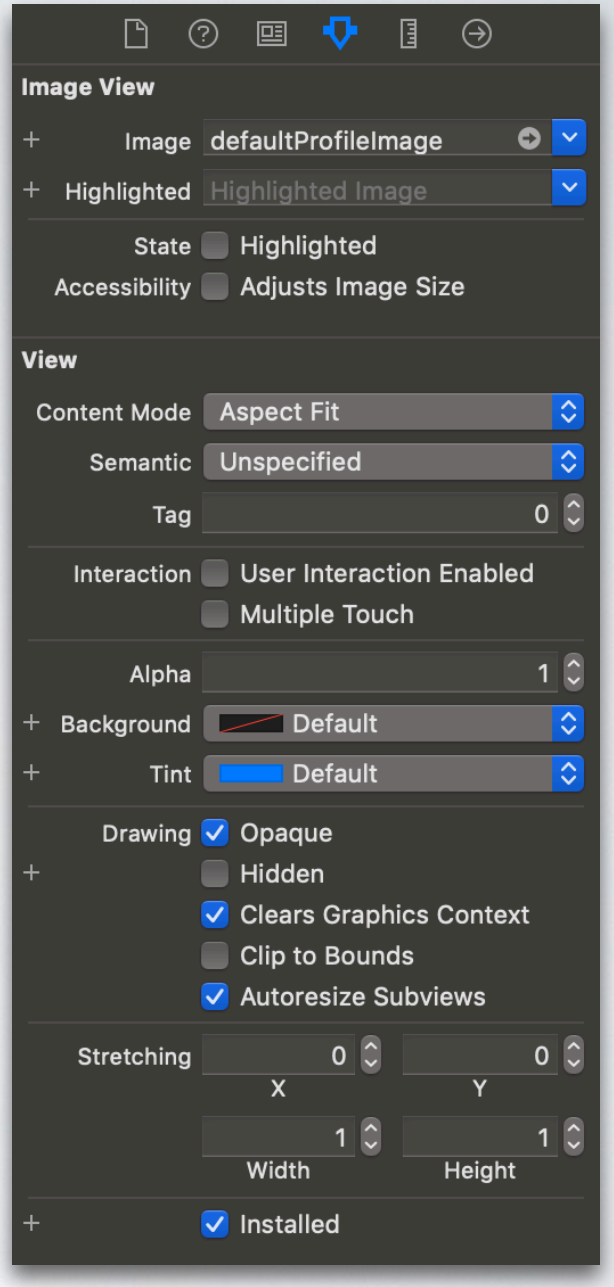
Programmatically

# Visually

# Ways to Style UI / Visually



# Programmatically

# One off views

```
@IBOutlet var imageView: UIImageView! {
    didSet {
        imageView.contentMode = .scaleAspectFit
        imageView.image = UIImage(named: "demo_image")
        imageView.layer.masksToBounds = true
        imageView.layer.cornerRadius = 8.0
        if #available(iOS 11.0, *) {
            imageView.layer.maskedCorners =
                [.layerMinXMinYCorner, .layerMaxXMinYCorner]
        }
    }
}
```

# Inheritance

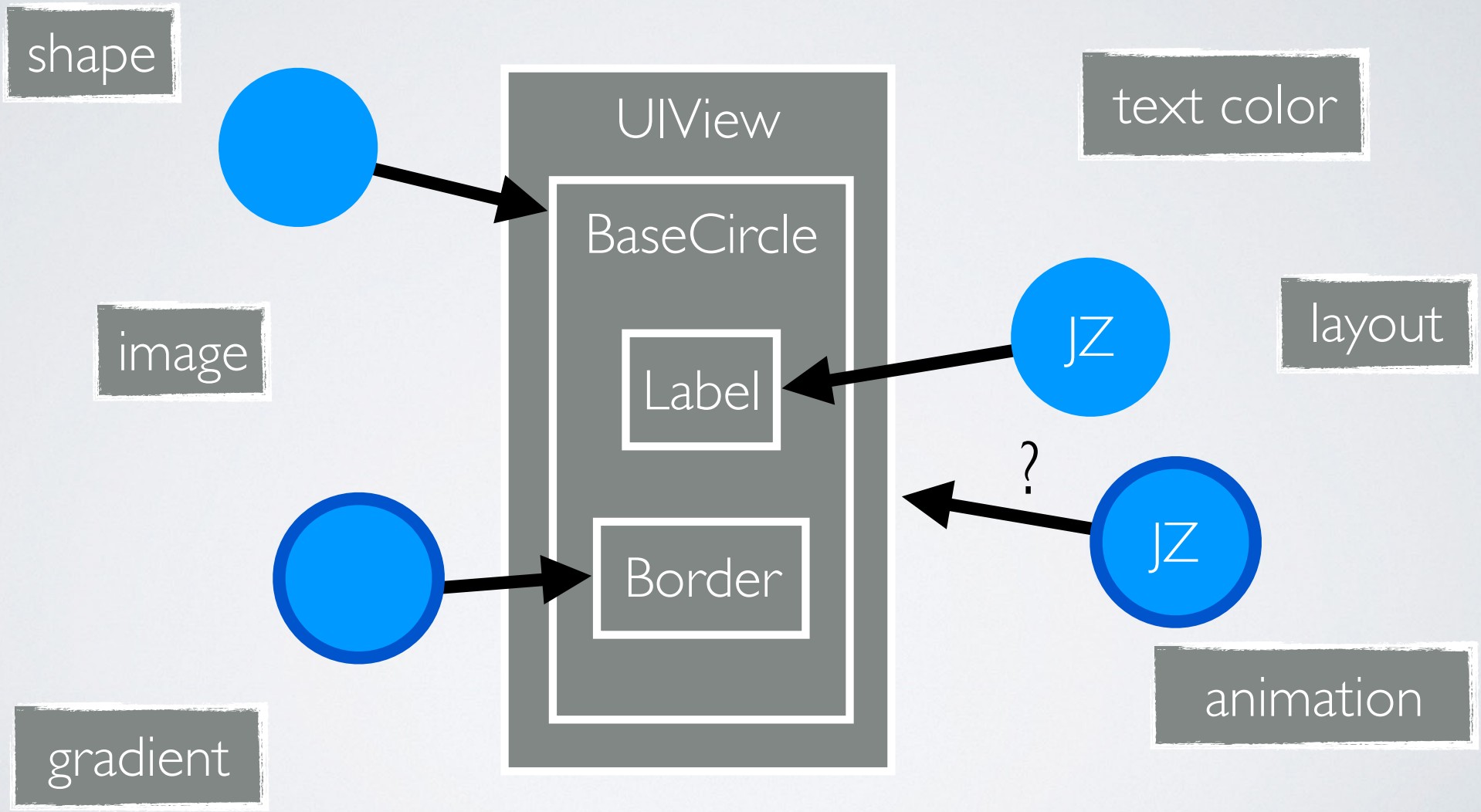
```
class TextCircle: BaseCircle {
    let label = UILabel()

    override init(frame: CGRect) {
        super.init(frame: frame)
        label.text = "jz"

        addSubview(label)
        label.translatesAutoresizingMaskIntoConstraints = false
        label.textAlignment = .center
        NSLayoutConstraint.activate([
            label.centerXAnchor.constraint(equalTo: centerXAnchor),
            label.centerYAnchor.constraint(equalTo: centerYAnchor),
            label.widthAnchor.constraint(equalTo: widthAnchor),
            label.heightAnchor.constraint(equalTo: heightAnchor),
            ])
        label.textColor = .white
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("\(#function) has not been implemented")
    }
}
```

# Inheritance



# Composition Over Inheritance

$f_1(\text{view}) + \ldots + f_x(\text{view}) =$ JZ

# The Diamond Operator (<>)

```
precedencegroup SingleTypeComposition {
    associativity: right
}

infix operator<>: SingleTypeComposition

func <> <A: AnyObject>(f: @escaping (A) -> Void, g: @escaping (A) -> Void) -> (A) -> Void {
    return { a in
        f(a)
        g(a)
    }
}
```

Swift 4.2

# Demo

# Takeaways

- ✅ Reusable | ✅ Simple to use | ✅ Discoverable
- Light weight
- Simple to introduce (or remove) from an existing code base
- The file with all the styles can be thought of as your app's style sheet (playground to show style!)
- Flexible - change the style of your UI easily

# References


- UIKit Styling with Functions — Point-Free
- Styling with Overture — Point-Free
- Beyond Types in Swift — Brandon Kase
- Custom Operators — Swift Language Guide
- The End of Object Inheritance & The Beginning of a New Modularity — Augie Fackler & Nathaniel Manista

# Thanks!

Jason Zurita — @jasonalexzurita