Ease your code with Swift Protocol Extensions

Alexandre Boyer Laporte
Co-founder and iOS developer
Sidekick Interactive

SIDE KICK INTERACTIVE

Who is this talk for?

- Someone that wants to learn new ways to reuse code
- Table View / Collection View programmers (everyone)

What you will get out of this

- Have a good example on how to use Protocol Extensions to your advantage
- A new way of implementing Table View and Collection View Data Sources and Delegates

SIDE KICK INTERACTIVE

Agenda

- Case Study: Table and Collection Views
- Demo

SIDE KICK INTERACTIVE

Case Study on Table and Collection Views

- Table Views and Collection Views are everywhere
- You need to implement a data source and delegate in order to make them work
- Implementation logic is almost always the same
- Also almost always requires customization

SIDE KICK INTERACTIVE

Case Study on Table and Collection Views

Conventional ways to solve

- Implement your data source and delegate logic in your view controller
- What if you wanted to reuse your logic in another type of view controller? (UIViewController subclass instead of a UITableViewController subclass)

SIDE KICK INTERACTIVE

Case Study on Table and Collection Views

Conventional ways to solve

- Implement your logic as an external class
  - Reusable across view controllers
- What if you wanted to customize behaviour with ease (without having to learn how the external class works)

Goal

- Provide generic way to solve a problem while still allowing maximum customization

SIDE KICK INTERACTIVE

Case Study on Table and Collection Views

A new way to solve this

- ANSWER: Protocol Extensions

SIDE KICK INTERACTIVE

Protocol Extensions

A better way for abstraction

- Let's you provide default implementation for your protocols
- Still lets your implementation (type or class) override the default implementation
- Allows "multiple inheritance" without complications

How to implement a single data source

1. Conform to TableViewDataSourceType and TableViewDelegateType

2. Provide a number of sections

3. Provide a cell identifier for a row/item

4. Prepare the cell

5. “Copy and paste" the base Data Source and Delegate Code

How to implement a composed Data Source

1. Conform the ComposedSectionedType
2. Create the necessary “nested” data source
3. “Copy and paste" the base Data Source and Delegate Code

SIDE KICK INTERACTIVE

Takeaways

- Protocol Extensions helps you stop writing tedious and repetitive code while giving you the flexibility of customization

SIDE KICK INTERACTIVE

Where to go from here?

General Questions
Alexandre Boyer Laporte
@steam_abl, alex@sideckickinteractive.com

WWDC 2015 Sessions
- What's new in Swift
- Protocol Oriented Programming in Swift
- Swift In Practice

WWDC 2014 Sessions
- Advanced User Interfaces with Collection Views