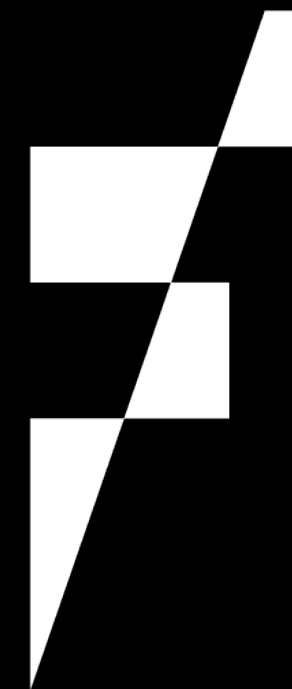




iOS Software engineer
since 2016



FABERNOVEL



Stating the
obvious. ●

1



Type erasure

```
protocol SomethingDelegate: AnyObject {  
    func somethingDelegate(_ something: Something,  
                           handleSomeTask argument: Argument)  
}
```

```
struct Something {  
    weak var delegate: SomethingDelegate?  
    // Implementation  
}
```

2 ○ Constraining for more functionalities

Constraining for more functionalities

```
struct CustomMapAnnotationModel {  
    let id: String  
    let location: CLLocation  
    let color: UIColor  
    // Other relevant properties  
}
```

Constraining for more functionalities

```
struct CustomMapAnnotationModel: Hashable {  
    let id: String  
    let location: CLLocation  
    let color: UIColor  
    // Other relevant properties  
}
```

Set, Dictionary storage
30+ Methods

3 ○ Providing an anchor point for extensions

Providing an anchor point for extensions

```
struct Book: Identifiable {  
    let id: BookID  
  
    var author: String { id.author }  
    var edition: String { id.edition }  
    var title: String { id.title }  
    let chapterList: [String]  
}  
  
struct BookID: Hashable {  
    let title: String  
    let author: String  
    let edition: String  
}
```

Providing an anchor point for extensions

```
extension Collection where Element: Identifiable, Element.ID == BookID {  
    func grouped<T: Hashable>(by keyPath: KeyPath<BookID, T>) -> [T: [Element]] {  
        Dictionary(grouping: self) { element -> T in  
            element.id[keyPath: keyPath]  
        }  
    }  
}
```

Providing an anchor point for extensions

```
extension Collection where Element: Identifiable, Element.ID == BookID {  
    func grouped<T: Hashable>(by keyPath: KeyPath<BookID, T>) -> [T: [Element]] {a  
        Dictionary(grouping: self) { element -> T in  
            element.id[keyPath: keyPath]  
        }  
    }  
}
```

```
let books: [Book] = //Some array of books  
let dictionaryOfBooks = books.grouped(by: \.author)  
let tolkiensBooks = dictionaryOfBooks["J. R. R. Tolkien"]  
let rollingsBooks = dictionaryOfBooks["J. K. Rollings"]
```

Providing an anchor point for extensions

```
struct CustomMapAnnotationModel: Hashable,  
    Identifiable {  
    let id: String  
    let location: CLLocation  
    let icon: MyIconEnum  
    let color: UIColor  
    // Other relevant properties  
}
```

Providing an anchor point for extensions

```
struct CustomMapAnnotationModel: Hashable,  
    Identifiable {  
    let id: String  
    let location: CLLocation  
    let icon: MyIconEnum  
    let color: UIColor  
    // Other relevant properties  
}
```

```
protocol Locatable {  
    var location: CLLocation { get }  
}
```

Providing an anchor point for extensions

```
protocol Locatable {  
    var location: CLLocation { get }  
}
```

```
extension Swift.Collection where Element: Locatable {  
    func sortByDistance(from location: CLLocation) -> [Element] {  
        return map { ($0, location.distance(from: $0.location)) }  
            .sorted { $0.1 < $1.1 }  
            .map { $0.0 }  
    }  
}
```

```
let fiftyClosestElements = locatables  
    .sortByDistance(from: location)  
    .prefix(50)
```

4



When should you conform ?

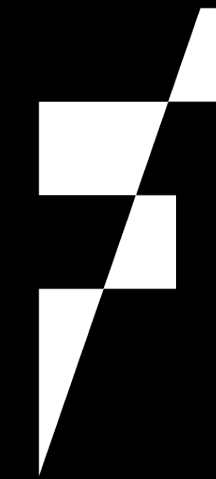
When should you conform ?

Is it useful ?

How much work does it imply?



Denis Poifol



FABERNOVEL

THANK YOU

