Swift Philosophy

Safe

Fast

Expressive

History

Introduced at WWDC 2014

Swift 2 at WWDC 2015

Open-sourced December 2015

Optionals

```
//Obj-C
- (NSString *)fullNameFromFirstName:(NSString *) lastName:(NSString *) {
    return [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
}
```

Optionals

```
//Obj-C
- (NSString *)fullNameFromFirstName:(NSString *) lastName:(NSString *) {
    return [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
}

[self fullNameFromFirstName:nil lastName:@"Doe"];
[self fullNameFromFirstName:@"Jane" lastName:nil];
```

Optionals

```
//Obj-C
- (NSString *)fullNameFromFirstName:(NSString *) lastName:(NSString *) {
    return [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
}

[self fullNameFromFirstName:nil lastName:@"Doe"];  //(null)
[self fullNameFromFirstName:@"Jane" lastName:nil]; //NSInvalidArgumentException
```

Optionals

```
//Obj-C
- (NSString *)fullNameFromFirstName:(NSString *) lastName:(NSString *) {
    return [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
}

//Swift
func fullName(firstName: String, lastName: String) -> String {
    return firstName + " " + lastName
}
```

Optionals

```
//Obj-C
- (NSString *)fullNameFromFirstName:(NSString *) lastName:(NSString *) {
    return [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
} //breaks at runtime

//Swift
func fullName(firstName: String, lastName: String) -> String {
    return firstName + " " + lastName
} //fails to compile
```

Optionals

```
//Obj-C
- (NSString *)fullNameFromFirstName:(NSString *) lastName:(NSString *) {
    return [[firstName stringByAppendingString:@" "] stringByAppendingString:lastName];
} //breaks at runtime

//Swift
func fullName(firstName: String?, lastName: String?) -> String {
    return firstName + " " + lastName
} //fails to compile
```

Constants and Variables

```
let name = "Jimmy D. Snitch"
```

Constants and Variables

```
let name = "Jimmy D. Snitch"
name = "Gerald N. Witnessprotection" //nope!
```

Constants and Variables

```
var name = "Jimmy D. Snitch"
name = "Gerald N. Witnessprotection" //okay!
```

Constants and Variables

```
var directors = ["Chris Brummel", "Rhonda Hoskins", "Rob Howard", "Ben Kotovic", "Brett McGinnis"]
```

Constants and Variables

```
var directors = ["Chris Brummel", "Rhonda Hoskins", "Rob Howard", "Ben Kotovic", "Brett McGinnis"]
directors.append("Andrew Smith") //okay!
```

Constants and Variables

```
var directors = ["Chris Brummel", "Rhonda Hoskins", "Rob Howard", "Ben Kotovic", "Brett McGinnis"]
directors.append("Andrew Smith") //okay!

let founders = ["Brandon Albers", "Bruce James", "Keith O'Neill"]
```

Constants and Variables

```
var directors = ["Chris Brummel", "Rhonda Hoskins", "Rob Howard", "Ben Kotovic", "Brett McGinnis"]
directors.append("Andrew Smith") //okay!

let founders = ["Brandon Albers", "Bruce James", "Keith O'Neill"]
founders.append("Brett McGinnis") //totally not okay!
```

Constants and Variables

```
class Singleton {
    static let sharedInstance = Singleton()
}

Singleton.sharedInstance
```

Types

```
class Person: NSObject {
    let firstName: String
    let lastName: String

    init(first: String, last: String) {
        firstName = first
        lastName = last
    }
}
```

Types

```
struct Person {
    let firstName: String
    let lastName: String

    init(first: String, last: String) {
        firstName = first
        lastName = last
    }
}
```

Types

```
struct Person {
    let firstName: String
    let lastName: String

    init(first: String, last: String) {
        firstName = first
        lastName = last
    }

    func fullName() -> String {
        return "\(firstName) \(lastName)"
    }
}
```

Types

```
enum Team {
    case Design,
    Development,
    Product,
    ProjectManagement,
    QA
}
```

Types

```
enum Team: Int {
    case Design,
    Development,
    Product,
    ProjectManagement,
    QA
}
```

Types

```
enum Team: Int {
	case Design,
	Development,
	Product,
	ProjectManagement,
	QA
}

let team = Team(rawValue: 0)
debugPrint(team) //Optional(Team.Design)
```

Types

```
enum Team: String {
    case Design,
    Development,
    Product,
    ProjectManagement,
    QA
}
```

Types

```
enum Team: String {
    case Design = "design",
    Development = "dev",
    Product = "product",
    ProjectManagement = "pm",
    QA = "qa"
}
```

Types

```
enum Team: String {
	case Design = "design",
	Development = "dev",
	Product = "product",
	ProjectManagement = "pm",
	QA = "qa"
}

let team = Team(rawValue: "dev")
debugPrint(team) //Optional(Team.Development)
```

Types

```
enum Team: String {
    case Design = "design",
    Development = "dev",
    Product = "product",
    ProjectManagement = "pm",
    QA = "qa"

    func isTechnical() -> Bool {
        return self == .Design || self == .QA
    }
}
```

Swift 3

Linux support

New community-driven features

Package manager

Rewritten Foundation