Structs vs. Classes in Swift

Structs and classes are both data types in Swift, but they have different properties and uses.

Structs are value types, which means that when they are copied or passed to a function, a new copy of the struct is created. Structs are typically used to represent small, immutable pieces of data, such as a point in 2D space or a user's name.

Classes are reference types, which means that when they are copied or passed to a function, a reference to the class is copied instead of the class itself. Classes are typically used to represent complex data structures, such as a user account or a view in a user interface.

Here is a table that summarizes the key differences between structs and classes:

Property	Struct	Class
Type	Value type	Reference type
Copy behavior	New copy is created	Reference to the class is copied
Inheritance	Not supported	Supported
Mutability	Immutable	Mutable
Memory management	Automatic memory management is used	Automatic reference counting (ARC) is used

When to use structs:

- To represent small, immutable pieces of data, such as a point in 2D space or a user's name.
- To pass data to functions without modifying the original data.
- To create immutable copies of data.

When to use classes:

- To represent complex data structures, such as a user account or a view in a user interface.
- To pass data to functions and allow the function to modify the data.
- To inherit from other classes.

Examples:

Here is an example of a struct:

```swift
struct Point {
    var x: Double
    var y: Double
}
```

Here is an example of a class:

```swift
class User {
    var name: String
    var email: String

    init(name: String, email: String) {
        self.name = name
        self.email = email
    }
}
```

Here is an example of how to use a struct:

Swift

```swift
var point1 = Point(x: 10.0, y: 20.0)
var point2 = point1

point1.x = 30.0

print(point1.x) // 30.0
print(point2.x) // 10.0
```

In this example, the point1 and point2 variables are both assigned the same Point struct. When the point1.x property is changed, the point2.x property is not affected, because structs are value types.

Here is an example of how to use a class:

Swift

```swift
var user1 = User(name: "John Doe", email: "john.doe@example.com")
var user2 = user1

user1.name = "Jane Doe"

print(user1.name) // Jane Doe
print(user2.name) // Jane Doe
```

In this example, the user1 and user2 variables are both assigned the same User class. When the user1.name property is changed, the user2.name property is also affected, because classes are reference types.

Conclusion

Structs and classes are both powerful data types in Swift. The best data type to use depends on the specific needs of your application. If you need to represent small, immutable pieces of data, then a struct is a good choice. If you need to represent complex data structures or allow data to be modified by functions, then a class is a good choice.