From Stack: <http://stackoverflow.com/questions/24217586/structure-vs-class-in-swift-language>

Here's an example with a class. Note how when the name is changed, the instance referenced by both variables is updated. Bob is now Sue, everywhere that Bob was ever referenced.

**class SomeClass {**

**var name: String**

**init(name: String) {**

**self.name = name**

**}**

**}**

**var aClass = SomeClass(name: "Bob")**

**var bClass = aClass // aClass and bClass now reference the same instance!**

**bClass.name = "Sue"**

**println(aClass.name) // "Sue"**

**println(bClass.name) // "Sue"**

And now with a struct we see that the values are copied and each variable keeps it's own set of values. When we set the name to Sue, the Bob struct in aStruct does not get changed.

**struct SomeStruct {**

**var name: String**

**init(name: String) {**

**self.name = name**

**}**

**}**

**var aStruct = SomeStruct(name: "Bob")**

**var bStruct = aStruct // aStruct and bStruct are two structs with the same value!**

**bStruct.name = "Sue"**

**println(aStruct.name) // "Bob"**

**println(bStruct.name) // "Sue"**

So for representing a stateful complex entity, a class is awesome. But for values that are simply a measurement or bits of related data, a struct makes more sense so that you can easily copy them around and calculate with them or modify the values without fear of side effects.