Свойства

Введение. Виды свойств.
Свойства (англ. properties) связывают значения с конкретным классом, структурой или перечислением. Различают хранимые и вычисляемые свойства.
Хранимые свойства запоминают значение константы или переменной, в то время как вычисляемые свойства рассчитывают значение при обращении. Вычисляемые свойства доступны классам структурам и перечислениям, а хранимые только классам и структурам.
Хранимые и вычисляемые свойства обычно привязываются к инстансам,но свойства могут быть связаны непосредственно с типом. Такие свойства называются свойствами типов.
Также для хранимых свойств можно определить обозреватель(observers). Они будут наблюдать за изменением значений свойства, а при изменении, выполнять дополнительные действия.

Хранимые свойства
В простейшем понимании хранимое свойство - это константа или переменная принадлежащая инстансу. Для свойства можно указать значение по умолчанию или присвоить значение в инициализаторе.

```
// --- Свойства CODE SNIPPET #1 --- //

struct SimpleStruct {
        var firstValue: Int = 1
        let secondValue: Int
}
var simpleStruct = SimpleStruct(firstValue: 0, secondValue: 2)
simpleStruct.firstValue = 6
```

Хранимые свойства в структурах
Если вы объявите структуру и присвоите значение константе. то значение свойств у этой структуры поменять вы не сможете.

```
// --- Свойства CODE SNIPPET #2 --- //

let constantStruct = SimpleStruct(firstValue: 0, secondValue: 2)
constantStruct.firstValue = 3 // Error: change 'let' to 'var' to make it mutable
```

Это связано с тем, что структуры являются значимыми типами. Если инстанс структуры помечается, как константа, то и свойства структуры будут константами. Для классов поведение отличается, так как классы - ссылочный тип. При присвоении инстанса класса константе, значения свойств, объявленных, как переменные могут быть изменены.

Хранимые свойства с отложенной инициализацией
Отложенная инициализация позволяет установить значение для свойства в момент первого обращения к нему. Такие свойства помечаются ключевым словом lazy.

// --- Свойства CODE SNIPPET #3 --- //

```
class DataStore {
	var counter = 0
}

class DataManager {
	lazy var dataStore = DataStore()
}

let dataManager = DataManager()
dataManager.dataStore.counter = 1
dataManager.dataStore.counter = 2
```

Свойства с отложенной инициализацией всегда объявляются как переменные. Ведь значение подобного свойства может быть определено после инициализации объекта, а для констант это непозволительно.
Использовать подобные свойства удобно, если их значения зависят от других значений и не могут быть определены на этапах создания объекта. Также отложенную инициализацию стоит применять для свойств, чьё создание может занять продолжительное время и потребовать много ресурсов.

Вычисляемые свойства
Помимо хранимых свойств могут быть определены вычисляемые свойства, они не хранят определенное значение, а вычисляют его при обращении к переменной.

// --- Свойства CODE SNIPPET #4 --- //

```
class AnotherDataStore {
	var counter: Int {
	get {
	return numbers.count
	}
	set(newCounter) {
	numbers = []
	for index in 0...newCounter {
		numbers.append(index)
	}
```

```
        }
        }

        var numbers = [Int]()
}

var anotherDataStore = AnotherDataStore()
anotherDataStore.counter = 5
print(anotherDataStore.counter)
```

Если у вычисляемого свойства в сеттере не задается имя для нового значения, оно задается по-умолчанию, с названием newValue. Имена передаваемых переменных также можно назначить самостоятельно.
Когда для вычисляемого свойства устанавливается геттер, но не устанавливается сеттер, оно доступно только для чтения.
Значения вычисляемого свойства устанавливается как переменная, а не как константа, так как значение не фиксировано.
При создании вычисляемого свойства только с геттером, синтаксис можно немного упростить, убрав ключевое слово get.

Наблюдатели свойств
Свойства следят за своими значениями и могут сообщать об этом. Обозреватели свойств вызываются при установке значения, даже если значение не изменилось.
Обозреватели можно использовать для любых хранимых свойств, кроме свойств с отложенной инициализацией.
Также обозреватели могут быть добавлены для унаследованных свойств.
Для наблюдения за изменения можно использовать ***willSet*** или ***didSet***.

```
// --- Свойства CODE SNIPPET #5 --- //

class ExplicitCounter {
        var total: Int = 0 {
                willSet(newTotal) {
                        print("Скоро установится новое значение \(newTotal)")
                }
                didSet {
                        if total > oldValue  {
                                print("Значение изменилось на:\(total - oldValue)")
                        }
                }
        }
}

let explicitCounter = ExplicitCounter()
explicitCounter.total = 5
explicitCounter.total = 9
```

WillSet вызывается непосредственно перед изменение свойства
DidSet вызывается сразу после изменения свойства.
В willSet передается константное значение. Для значение может быть использовано уникальное имя, если оно не установлено, значение передается с именем newValue. Для didSet также передается старое значение, но оно имеет другое имя по-умолчанию, oldValue. Если внутри didSet установить новое значение наблюдаемой переменной, оно заменит только что установленное.
Когда свойство с обозревателями передается функции в качестве in-out параметра, у свойства будут вызваны методы willSet и didSet. Это связано с особенностью реализации in-out параметров. При завершении функции всегда записывается новое значение и соответственно вызываются обозреватели.

Глобальные и локальные переменные.
Ранее описанные возможности доступны для глобальных и локальных переменных.
Глобальные переменные задаются вне метода, функции или контекста класса.
Локальные переменные объявляются внутри функции или замыкания.
Для локальных и глобальных переменных можно задать обозреватели или сделать переменные вычисляемыми. Все точно также, как и для свойств. Значения вычисляются при обращении к переменной.
Глобальные переменные и константы всегда вычисляются при обращении, в отличии от свойств глобальные переменные не нужно помечать ключевым словом lazy.
Локальные переменные не могут обладать отложенной инициализацией.
Свойства для типов.
Свойства объектов принадлежат объектам и создаются каждый раз при инициализации нового объекта.
Но можно создать свойство, которое будет единственным для всех объектов заданного типа и значение которого будет инициализировано только 1 раз. Такие свойства, называются свойствами типов.
Они аналогичны переменным и константам из языка Си, объявленным со словом static.
Вычисляемые свойства типов могут быть объявлены только, как переменные.
Хранимые свойства типов могут быть объявлены, как переменные, так и как константы. В отличии от свойств у инстансов, для переменных обязательно указание значения по-умолчанию потому, что у типов нет инициализатора. Соответственно нет возможности для задания первоначального значения.
Хранимые свойства получают значения при первом обращении к ним. Это происходит только 1 раз

Синтаксис
В Swift свойства типов описываются внутри описания типа. Каждое свойство привязно к типу в котором оно описано.
Каждое свойство привязывается к типу, для которого оно описывается.

// --- Свойства CODE SNIPPET #6 --- //

```
struct SomeStruct {
		static var storedTypeProperty = 1
		static var squareOfProperty: Int {
		return storedTypeProperty * storedTypeProperty
	}
}

class SomeClass {
	static var storedTypeProperty = 2
	static var squareOfProperty: Int {
		return storedTypeProperty * storedTypeProperty
	}
	static var overridableHalfOfProperty: Int {
		get {
		return storedTypeProperty * storedTypeProperty
		}
		set {
		storedTypeProperty = newValue / 2
		}
	}
}

print (SomeStruct.storedTypeProperty)

SomeClass.overridableHalfOfProperty = 10
print(SomeClass.squareOfProperty)
```

Обозначаются они ключевым словом static, либо в случае с классами можно использовать ключевое слово class, что позволит наследникам переопределять свойство.
Обращение осуществляется при помощи dot-синтаксиса. Однако, обращение идет не к инстансу, а к типу.