1. **lazy**

* 지연 저장 프로퍼티, 클래스와 구조체에 사용가능
* 호출이 있어야 값을 초기화 함
* 인스턴스가 완전히 생성되기 전에 초기화 해야하므로 var키워드 사용

1. 언제사용하는가?

* 프로퍼티 초기화시에 다른 인스턴스를 참조해야 할때
* 인스턴스내에 모든 프로퍼티를 사용할 필요가 없다면
* 다중스레드 환경에서 동시다발적으로 접근하는 경우에는 여러번 초기화가 될 수 있다.

1. 클로저를 활용한 초기화

* 클로져를 활용하여 값을 초기화 할수 있음
* 디폴트 값을 연산하는데 한 줄로 되지 않고 여러 라인의 코드가 필요한 경우

class Avatar {

static let defaultSmallSize = CGSize(width: 64, height: 64)

lazy var smallImage: UIImage = self.largeImage.resizedTo(Avatar.defaultSmallSize)

var largeImage: UIImage

init(largeImage: UIImage) {

self.largeImage = largeImage

}

}

* 클로져 내부에서 self를 참조 할수있다.
* lazy의 뜻은 self가 완전히 초기화 과정을 끝낸시점 이후에 호출되는 것이기 때문에 self를 참조하는데 아무런 문제가 없으며 unowned self와 같은 캡쳐과정이 불필요 하며 **순환참조가 발생할 걱정이 없다**.

class HTMLElement {

let name: String

let text: String?

lazy var asHTML: Void -> String = {

if let text = self.text {

return "<\(self.name)>\(text)</\(self.name)>"

} else {

return "<\(self.name) />"

}

}

init(name: String, text: String? = nil) {

self.name = name

self.text = text

}

deinit {

print("\(name) is being deinitialized")

}

}

* 단 예제와 같이 그 자체가 클로져인 지연변수인 경우 lazy변수를 참조시 self에 대한 강한 참조가 발생되면서 순환참조가 일어나므로 weak self로 캡쳐리스트를 사용해야 한다.

1. Weak와 사용할수 없는 이유는?

* lazy는 지연저장 프로퍼티인 대신 인스턴스가 모두 초기화 된 이후에 초기화되므로 다른 프로퍼티에 대해서 참조해야할 경우가 생긴다. 이때 weak는 rc카운터를 발생시키지 않으므로 참조하지 않는것에 대한 가정을 하고 있어서 lazy키워드와 사용할수 없다