Objective C	Swift
Declare Constant & variable	
int highScore;	var highScore: Int
NSString *playerName;	var playerName: String
Assigning to a variable	
HighScore = 1000;	highScore = 1000
playerName = @"Joe Chan";	playerName = "Joe Chan"
NSString *const skyColor = @"Blue";	let skyColor = "Blue"
Defining a class	
@interface CustomClass : NSObject	class CustomClass { }
@implementation CustomClass	
Creating new class instance	
CustomClass *instance = [[CustomClass alloc] init];	var instance = CustomClass()
CustomClass *secondInstance = [CustomClass new];	var instance = CustomClass()
Declaring methods	

- (int) getOdometerReading;	func getOdometerReading() -> Int {
	return 2000
	}

- (int) getOdometerReading{
return 2000;
}	

Calling methods	
SpecificCar *myCar = [[SpecificCar alloc] init]; [myCar getOdometerReading];	var myCar = SpecificCar() myCar.getOdometerReading()
Declaring methods with multiple parameter	
- (void)changeEngineOil:(int)oil	func changeEngineOil(oil:Int,

transmissionFluid:(int)fluid
{
 	
}
	transmissionFluid:Int)
	{
	
	}

Calling & overriding method with multiple parameter	
[myCar changeEngineOil:10 transmissionFluid:10];	myCar.changeEngineOil(10, transmissionFluid:10);
- (void)someMethod	override func someMethod()
If statement	

if (score > 25)
{
}
	if score > 25
	{
	}

Variable with optional value?(string may be empty)	

NSString *aString;
if (aString) {
}
	var aString: String?
	if let myString = aString {
	}

loop	
for (int i = 0; i < 10; i++)	for var i = 0; i < 10; i++
	for i in 0..10

NSArray *array = @[@"item 1",
@"item 2"];
for (NSString *item in array)
{
}
	var array = ["item 1", "item 2"]
	
	for item in array {
	}

Collection	
NSMutableArray *myArray = [[NSMutableArray alloc] init];	var myArray:[String] = [String]()
NSMutableArray *myArray2 = [@[@"item 1", @"item 2"] mutableCopy];	var myArray2 = ["item 1", "item 2"]
Adding and inserting into array	
myArray[0] = @"item 1";	myArray[0] = "item 1"

`[myArray addObject:@"item 3"];`	`myArray.append("item 3") or myArray += "item 3"`
`[myArray insertObject:@"item 4" atIndex:1];`	`myArray.insert("item 4", atIndex:1)`
Declaring Dictionary	
`NSMutableDictionary *myDictionary = [[NSMutableDictionary alloc] init];`	`var myDictionary = Dictionary<String, String>()`
`[myDictionary setObject:@"object 3" forKey:@"key3"];`	`myDictionary["key3"] = "object 3"`
`myDictionary[@"key2"] = @"object 4";`	`myDictionary["key2"] = "object 4"`
Declaring protocol	
`@protocol SampleProtocol <NSObject> - (void)someMethod;`	`protocol SampleProtocol { func someMethod() }`
Conforming protocol	
`@interface MyClass : NSObject<SampleProtocol>`	`class MyClass: SampleProtocol`
Declaring the delegate	
`@interface FirstClass : NSObject @property (nonatomic, weak) id<SampleProtocol> delegate;`	`class FirstClass{ var delegate:SampleProtocol? }`
Calling a delegate	
`if (self.delegate) [self.delegate someMethod];`	`delegate?.someMethod()`

Defining the block

```
- (void)SomethingWithFirstValue:(int)firstvalue
secondValue:(int)secondvalue block:(int (^)(int,
int))block {
int result = block(firstvalue, secondvalue);
NSLog(@"%d", result);
}
```

Calling the block

```
[self doSomethingWithFirstValue:3
                    secondValue:7
                    block:^(int firstValue, int secondValue)
{
return firstValue + secondValue;
}];
```

Defining the block

```
func SomethingWithClosure(firstValue:Int,secondValue:Int,
closure:(Int,Int) -> Int) {
let result = closure(firstValue, secondValue)
println(result)
```

calling the block

```
doSomethingWithClosure(3, 7,{(firstValue:Int,secondValue:Int)-
>;Int in
return firstValue + secondValue
})
```