```
// Variables

// var name = "Kevin"
var name = "Kevin"

// var CONSTANT = "MyConstant" - Not really a constant
let constant = "MyConstant"
// ES6
// const constant = "MyConstant"
// let blockLevel = "MyBlockLevel"

// var age = 21
var age = 21
// var speed = 2.5
var speed = 2.5
// var flag = true
var flag = true
// var combined = name + " is age " + age
var combined = "\(name) is age \(age)"

// Emojis?
var 🍕 = "Pizza"
var 🍣 = "Sushi"
var sushiPizza = "\(🍣) \(🍕)"


// Arrays


// var names = ["Kevin", "George", "Bob"]
var names = ["Kevin", "George", "Bob"]
// names.push("Emily")
names.append("Emily")
// name.pop()
names.removeLast()
// var someName = names[0]
var someName = names[0]


// Dictionaries


// var nameDictionary = { first: "Kevin", last: "Kazmierczak" }
var nameDictionary = [ "first": "Kevin", "last": "Kazmierczak"]
// var firstName = nameDictionary.first
var firstName = nameDictionary["first"]
// nameDictionary["address"] = "Marlborough"
nameDictionary["address"] = "Marlborough"
nameDictionary

// Loops

// for ( var x = 1; x <= 5; x++ ) {
//      console.log( x * 5 )
// }
```

```
for var x = 1; x <= 5; x++ {
    println(x * 5)
}

for x in 1...5 {
    println(x * 5)
}

// Array Looping

// var names = ["Kevin", "Bob", "George"]
// names.forEach( function ( name ) {
//      console.log( name )
// } );

var names1 = ["Kevin", "Bob", "George"]
for name in names {
    println(name)
}

for index in 0..<names.count {
    println(names[index])
}

// Dictionary Loops

// var nameObject = { first:"Kevin", last:"Kazmierczak" }
// for ( var key in nameObject ) {
//      console.log( nameObject[key] )
// }

var nameObject = ["first":"Kevin", "last":"Kazmierczak"]
for (key, value) in nameObject {
    println("\(key):\(value)")
}

// Create an optional
var someNumber:Int?
someNumber = 10

// Output that optional
if someNumber != nil {
    println(someNumber! * 10)
} else {
    println("No value")
}

// Optional binding
if let unwrappedNumber = someNumber {
    println(unwrappedNumber * 10 )
} else {
    println("Unable to unwrap")
}

// Forced unwrap
```

```
let possibleNumber = "57"
if let actualNumber = possibleNumber.toInt() {
    println(actualNumber)
} else {
    println("Possible number isn't a number")
}

// Forced unwrap
let anotherActualNumber = possibleNumber.toInt()
let someAddition = anotherActualNumber! + 50
let someMoreAddition = anotherActualNumber! + 75

// Implicit unwrap
let implicitActualNumber:Int! = possibleNumber.toInt()
let someMultiplication = implicitActualNumber * 50
let someMoreMultiplication = implicitActualNumber * 75

// Fancy - nil coalescing operator

var optionalNumber = "43".toInt()
let fancyConversion = optionalNumber ?? 21 // is the same as doing
let notSoFancy = optionalNumber != nil ? optionalNumber! : 21


// Basic Functions

// function squareRoot( input ) {
//      return input * input;
// }
// var result = squareRoot( 7 );

func squareRoot(input:Int) -> Int {
    return input * input
}
let result = squareRoot(7)

// Functions returning undefined values

// function optionalFunction( flag ) {
//      if ( flag ) {
//          return "Some data";
//      } else {
//          return null;
//      }
// }
// var optionalReturn = optionalFunction(false);

// Show optional return
func optionalFunction(flag:Bool) -> String? {
    if flag {
        return "Some data"
    } else {
        return nil
    }
}
let optionalReturn = optionalFunction(true)
```

```
// Tuples

// function getName() {
//  return {
//      first: "Kevin",
//      last: "Kazmierczak"
//  };
// }
// var myName = getName();

func getName() -> (first:String, last:String) {
    return ("kevin", "kazmierczak")
}
let myName = getName()
let (first, last) = myName

// ES6 Destructuring
// var {first, last} = myName

// Named parameters

// function combineString(obj){
//  // Logic to ensure properties exist with defaults
//  return obj.first + " " + obj.mi + " " + obj.last;
// }
// var combined = combineString({first:"Kevin", last:"Kazmierczak", mi:"R"})

func combineStrings(first:String, last:String, mi:String = "") -> String {
    return "\(first) \(mi) \(last)"
}
let combined1 = combineStrings("Kevin", "Kazmierczak", mi:"R")

// Closures

// var names = ["Kevin", "Andrea"];
// var sorted = names.map( function(name) {
//      return name + " was here";
// });

let names2 = ["Kevin", "Andrea", "Emily"]
let sorted = names2.map({
    (name: String) -> String in
    return "\(name) is here"
})

let sorted2 = names.map({ "\($0) is here" })

// var numbers = [1,2,3,4,5];
// var sum = numbers.reduce( function(number, start){
//      return start + number;
// }, 0);

let numbers = [1,2,3,4,5]
let sum = numbers.reduce(0, combine: { (total:Int, value:Int) -> Int in
    return total + value
```

```
})

let sum2 = numbers.reduce(10){$0 + $1}
sum2

func getFullName() -> (first:String, last:String) {
    return ("Kevin", "Kazmierczak")
}
let name3 = getFullName().first // returns "Kevin"


// Classes Example

// var Person = function (firstName, lastName){
//      this.firstName = firstName;
//      this.lastName = lastName;
//      this.address = undefined;
// };
// Person.prototype.getNameAndAddress = function () {
//      if ( this.address ) {
//          return this.firstName + " " + this.lastName + " lives in " +
    this.address;
//      } else {
//          return this.firstName + " " + this.lastName;
//      }
// };

// var p = new Person("Kevin", "Kazmierczak");
// p.address = "Marlborough";
// var full = p.getNameAndAddress();

class Person {
    var firstName = ""
    var lastName = ""
    var address:String?
    var fullName:String {
        return "\(firstName) \(lastName)"
    }
    init(fName:String){
        firstName = fName
    }
    func getNameAndAddress() -> String {
        if address != nil {
            return "\(firstName) \(lastName) lives in \(address!)"
        } else {
            return "\(firstName) \(lastName)"
        }
    }
}

class Programmer: Person{
    var favoriteLanguage = "Swift"
}

let myPerson = Person(fName: "Kevin")
myPerson.lastName = "Kazmierczak"
```

```
myPerson.address = "Marlborough"
let full = myPerson.getNameAndAddress()

let myProgrammer = Programmer(fName: "Emily")
myProgrammer.favoriteLanguage = "HTML"
```