

Unit 2—Lesson 3:

Structures

Structures

```
struct Person {  
    var name: String  
}
```

Capitalize type names

Use lowercase for property names

Structures

Accessing property values

```
struct Person {  
    var name: String  
}  
  
let person = Person(name: "Jasmine")  
print(person.name)
```

Jasmine

Structures

Adding functionality

```
struct Person {  
    var name: String  
  
    func sayHello() {  
        print("Hello there! My name is \(name)!")  
    }  
}  
  
let person = Person(name: "Jasmine")  
person.sayHello()
```

Hello there! My name is Jasmine!

Instances

```
struct Shirt {  
  var size: String  
  var color: String  
}  
  
let myShirt = Shirt(size: "XL", color: "blue")  
  
let yourShirt = Shirt(size: "M", color: "red")
```

```
struct Car {  
    var make: String  
    var year: Int  
    var color: String  
  
    func startEngine() {...}  
  
    func drive() {...}  
  
    func park() {...}  
  
    func steer(direction: Direction) {...}  
}  
  
let firstCar = Car(make: "Honda", year: 2010, color: "blue")  
let secondCar = Car(make: "Ford", year: 2013, color: "black")  
  
firstCar.startEngine()  
firstCar.drive()
```

Initializers

```
let string = String.init() // ""  
let integer = Int.init() // 0  
let bool = Bool.init() // false
```

Initializers

```
var string = String() // ""  
var integer = Int() // 0  
var bool = Bool() // false
```


Initializers

Default values

```
struct Odometer {  
    var count: Int = 0  
}  
  
let odometer = Odometer()  
print(odometer.count)
```

0

Initializers

Memberwise initializers

```
let odometer = Odometer(count: 27000)  
print(odometer.count)
```

27000

Initializers

Memberwise initializers

```
struct Person {  
    var name: String  
}
```

Initializers

Memberwise initializers

```
struct Person {  
    var name: String  
  
    func sayHello() {  
        print("Hello there!")  
    }  
}  
  
let person = Person(name: "Jasmine") // Memberwise initializer
```

```
struct Shirt {  
    let size: String  
    let color: String  
}
```

```
let myShirt = Shirt(size: "XL", color: "blue") // Memberwise initializer
```

```
struct Car {  
    let make: String  
    let year: Int  
    let color: String  
}
```

```
let firstCar = Car(make: "Honda", year: 2010, color: "blue") // Memberwise initializer
```

Initializers

Custom initializers

```
struct Temperature {  
    var celsius: Double  
}
```

```
let temperature = Temperature(celsius: 30.0)
```

```
let fahrenheitValue = 98.6  
let celsiusValue = (fahrenheitValue - 32) / 1.8  
  
let newTemperature = Temperature(celsius: celsiusValue)
```

```
struct Temperature {  
    var celsius: Double  
  
    init(celsius: Double) {  
        self.celsius = celsius  
    }  
  
    init(fahrenheit: Double) {  
        celsius = (fahrenheit - 32) / 1.8  
    }  
}  
  
let currentTemperature = Temperature(celsius: 18.5)  
let boiling = Temperature(fahrenheit: 212.0)  
  
print(currentTemperature.celsius)  
print(boiling.celsius)
```

18.5

100.0

Unit 2—Lesson 3

Lab: Structures



Open and complete the following exercises in Lab – Structures.playground:

- Exercise - Structs, Instances, and Default Values
- App Exercise - Workout Tracking

Instance methods

```
struct Size {  
    var width: Double  
    var height: Double  
  
    func area() -> Double {  
        return width * height  
    }  
}  
  
var someSize = Size(width: 10.0, height: 5.5)  
  
let area = someSize.area() // Area is assigned a value of 55.0
```

Mutating methods

```
struct Odometer {  
    var count: Int = 0 // Assigns a default value to the 'count' property.  
}
```

Need to

- Increment the mileage
- Reset the mileage

```
struct Odometer {  
    var count: Int = 0 // Assigns a default value to the 'count' property.  
  
    mutating func increment() {  
        count += 1  
    }  
  
    mutating func increment(by amount: Int) {  
        count += amount  
    }  
  
    mutating func reset() {  
        count = 0  
    }  
}
```

```
var odometer = Odometer() // odometer.count defaults to 0  
odometer.increment() // odometer.count is incremented to 1  
odometer.increment(by: 15) // odometer.count is incremented to 16  
odometer.reset() // odometer.count is reset to 0
```

Computed properties

```
struct Temperature {  
  let celsius: Double  
  let fahrenheit: Double  
  let kelvin: Double  
}  
  
let temperature = Temperature(celsius: 0, fahrenheit: 32, kelvin: 273.15)
```

```
struct Temperature {  
    var celsius: Double  
    var fahrenheit: Double  
    var kelvin: Double  
  
    init(celsius: Double) {  
        self.celsius = celsius  
        fahrenheit = celsius * 1.8 + 32  
        kelvin = celsius + 273.15  
    }  
  
    init(fahrenheit: Double) {  
        self.fahrenheit = fahrenheit  
        celsius = (fahrenheit - 32) / 1.8  
        kelvin = celsius + 273.15  
    }  
  
    init(kelvin: Double) {  
        self.kelvin = kelvin  
        celsius = kelvin - 273.15  
        fahrenheit = celsius * 1.8 + 32  
    }  
}  
  
let currentTemperature = Temperature(celsius: 18.5)  
let boiling = Temperature(fahrenheit: 212.0)  
let freezing = Temperature(kelvin: 273.15)
```

```
struct Temperature {  
    var celsius: Double  
  
    var fahrenheit: Double {  
        return celsius * 1.8 + 32  
    }  
}
```

```
let currentTemperature = Temperature(celsius: 0.0)  
print(currentTemperature.fahrenheit)
```

32.0

Challenge



Add support for Kelvin

Modify the following to allow the temperature to be read as Kelvin

```
struct Temperature {  
    let celsius: Double  
  
    var fahrenheit: Double {  
        return celsius * 1.8 + 32  
    }  
  
}
```

Hint: Temperature in Kelvin is Celsius + 273.15

```
struct Temperature {  
    let celsius: Double  
  
    var fahrenheit: Double {  
        return celsius * 1.8 + 32  
    }  
  
    var kelvin: Double {  
        return celsius + 273.15  
    }  
}  
  
let currentTemperature = Temperature(celsius: 0.0)  
print(currentTemperature.kelvin)
```

273.15

Property observers

```
struct StepCounter {  
    var totalSteps: Int = 0 {  
        willSet {  
            print("About to set totalSteps to \ (newValue)")  
        }  
        didSet {  
            if totalSteps > oldValue {  
                print("Added \ (totalSteps - oldValue) steps")  
            }  
        }  
    }  
}
```

Property observers

```
var stepCounter = StepCounter()  
stepCounter.totalSteps = 40  
stepCounter.totalSteps = 100
```

```
About to set totalSteps to 40  
Added 40 steps  
About to set totalSteps to 100  
Added 60 steps
```

Type properties and methods

```
struct Temperature {  
    static var boilingPoint = 100.0  
  
    static func convertedFromFahrenheit(_ temperatureInFahrenheit: Double) -> Double {  
        return(((temperatureInFahrenheit - 32) * 5) / 9)  
    }  
}  
  
let boilingPoint = Temperature.boilingPoint  
  
let currentTemperature = Temperature.convertedFromFahrenheit(99)  
  
let positiveNumber = abs(-4.14)
```

Copying

```
var someSize = Size(width: 250, height: 1000)
var anotherSize = someSize

someSize.width = 500

print(someSize.width)
print(anotherSize.width)
```

500

250

self

```
struct Car {  
    var color: Color  
  
    var description: String {  
        return "This is a \(self.color) car."  
    }  
}
```

self

When not required

Not required when property or method names exist on the current object

```
struct Car {  
  var color: Color  
  
  var description: String {  
    return "This is a \(color) car."  
  }  
}
```

self

When required

```
struct Temperature {  
    var celsius: Double  
  
    init(celsius: Double) {  
        self.celsius = celsius  
    }  
}
```

Unit 2—Lesson 3

Lab: Structures



Open and complete the remaining exercises in
Lab – Structures.playground

