Measurements and Units

Using locale-appropriate measurements in your app

Session 238

Daphne Larose Software Engineer, Foundation

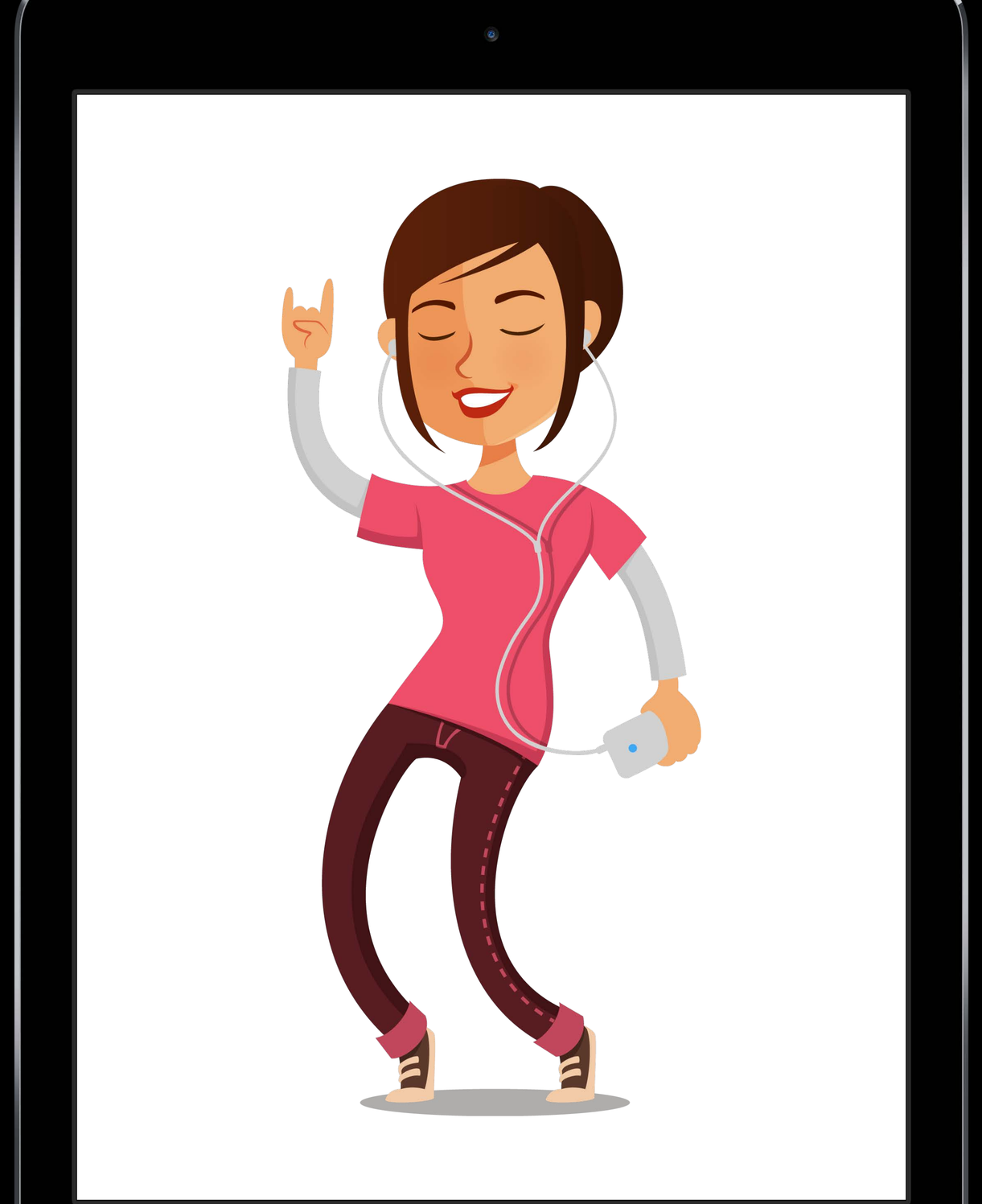
Introduction

Measurements pop up all the time

Measurements should be in preferred units

New API makes it easy to do the right thing for everyone

Jammin’ in the Streetz



Goals

Fun

Lots of emojis

Available in multiple countries

Key Elements of the Game

Jam sessions

Tracks

- Total time
- Distance traveled
- Number of dance movements performed
- Rate of travel

Key Elements of the Game

Jam sessions

Tracks

- Total time
- Distance traveled
- Number of dance movements performed
- Rate of travel

5

5

5 feet

Creating Measurements Easily

```swift
// Measurement

public struct Measurement<UnitType : Unit> : Comparable, Equatable {
```

NEW

```swift
// Measurement

public struct Measurement<UnitType : Unit> : Comparable, Equatable {
    public let unit: UnitType
```

NEW

```swift
// Measurement

public struct Measurement<UnitType : Unit> : Comparable, Equatable {
    public let unit: UnitType
    public var value: Double
```

NEW

```swift
// Measurement

public struct Measurement<UnitType : Unit> : Comparable, Equatable {
    public let unit: UnitType
    public var value: Double
    public init(value: Double, unit: UnitType)
}
```

NEW

Distance to go

Distance traveled

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
```

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
```

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
let totalDistance = distanceTraveled + distanceToGo
```

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
let totalDistance = distanceTraveled + distanceToGo
```

```
value: 11, unit: .feet
```

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
let totalDistance = distanceTraveled + distanceToGo
let tripleDistance = 3 * distanceToGo
```

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
let totalDistance = distanceTraveled + distanceToGo
let tripleDistance = 3 * distanceToGo
```

```
value: 18, unit: .feet
```

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
let totalDistance = distanceTraveled + distanceToGo
let tripleDistance = 3 * distanceToGo
let halfDistance = distanceToGo / 2
```

```swift
// Calculations With Measurements

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
let totalDistance = distanceTraveled + distanceToGo
let tripleDistance = 3 * distanceToGo
let halfDistance = distanceToGo / 2
```

value: 3, unit: .feet

Properties of a Unit

Properties of a Unit

Symbol	“ft”

Properties of a Unit

Symbol	“ft”
Dimension	“Foot is a unit of length”

Properties of a Unit

Symbol	“ft”
Dimension	“Foot is a unit of length”
Equivalence	1ft = 0.348m

```swift
// Unit

public class Unit : NSObject, NSCopying {
    public let symbol : String
    public init(symbol: String)
}
```

NEW

Dimension

Dimension

Categories of units

Dimension

Categories of units

Expressed with different units

- Length: km, m, ft, mi, etc.

Dimension

Categories of units

Expressed with different units

- Length: km, m, ft, mi, etc.

Always has a base unit

```
let meter = UnitLength.baseUnit
```

Dimension

Categories of units

Expressed with different units

- Length: km, m, ft, mi, etc.

Always has a base unit

```
let meter = UnitLength.baseUnit
```

Can perform conversions

- km ⇆ ft, m ⇆ mi

```swift
// Dimension

public class Dimension : Unit, NSCoding {
```

NEW

```swift
// Dimension

public class Dimension : Unit, NSCoding {
    public var converter : UnitConverter { get }
```

NEW

```swift
// Dimension

public class Dimension : Unit, NSCoding {
    public var converter : UnitConverter { get }
    public init(symbol: String, converter: UnitConverter)
```

NEW

```swift
// Dimension

public class Dimension : Unit, NSCoding {
    public var converter : UnitConverter { get }
    public init(symbol: String, converter: UnitConverter)
    public class var baseUnit : Dimension
}
```

NEW

Dimension

Abstract units

Instances as units

Dimension

Abstract units

Instances as units

Singletons for most common units

Dimension

Abstract units

Instances as units

Singletons for most common units

International System of Units

Dimension

Abstract units

Instances as units

Singletons for most common units

International System of Units

```
public class UnitLength : Dimension {
/*
 Base unit - meters
 */
    public class var kilometers: UnitLength { get }
    public class var meters: UnitLength { get }
    public class var feet: UnitLength { get }
    public class var miles: UnitLength { get }

    ...
}
```

Dimension

Provided subclasses

NEW

UnitAcceleration	UnitElectricCurrent	UnitIlluminance
UnitAngle	UnitElectricPotentialDifference	UnitMass
UnitArea	UnitElectricResistance	UnitPower
UnitConcentrationMass	UnitEnergy	UnitPressure
UnitDispersion	UnitFrequency	UnitSpeed
UnitDuration	UnitFuelEfficiency	UnitTemperature
UnitElectricCharge	UnitLength	UnitVolume

```swift
// Implicit Conversion

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
let totalDistance = distanceTraveled + distanceToGo
```

```
value: 11, unit: .feet
```

```
// Implicit Conversion

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
// let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
// let totalDistance = distanceTraveled + distanceToGo
```

```
// Implicit Conversion

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
// let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
// let totalDistance = distanceTraveled + distanceToGo

let distanceToGo = Measurement(value: 6, unit: UnitLength.kilometers)
```

```
// Implicit Conversion

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
// let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
// let totalDistance = distanceTraveled + distanceToGo

let distanceToGo = Measurement(value: 6, unit: UnitLength.kilometers)
let totalDistance = distanceTraveled + distanceToGo
```

```swift
// Implicit Conversion

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
// let distanceToGo = Measurement(value: 6, unit: UnitLength.feet)
// let totalDistance = distanceTraveled + distanceToGo

let distanceToGo = Measurement(value: 6, unit: UnitLength.kilometers)
let totalDistance = distanceTraveled + distanceToGo
```

```
value: 6001.524, unit: .meters
```

```
// Comparison Operators

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.kilometers)

var distanceMarker : String

if (distanceTraveled > distanceToGo) {
    distanceMarker = "Almost there!"
} else if (distanceTraveled < distanceToGo) {
    distanceMarker = "Barely started!"
} else {
    distanceMarker = "Halfway!"
}
```

```
// Comparison Operators

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 6, unit: UnitLength.kilometers)

var distanceMarker : String

if (distanceTraveled > distanceToGo) {
    distanceMarker = "Almost there!"
} else if (distanceTraveled < distanceToGo) {
    distanceMarker = "Barely started!"
} else {
    distanceMarker = "Halfway!"
}
```

```
// Comparison Operators

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 5, unit: UnitLength.kilometers)

var distanceMarker : String

if (distanceTraveled > distanceToGo) {
    distanceMarker = "Almost there!"
} else if (distanceTraveled < distanceToGo) {
    distanceMarker = "Barely started!"
} else {
    distanceMarker = "Halfway!"
}

print(distanceMarker)
```

```swift
// Comparison Operators

let distanceTraveled = Measurement(value: 5, unit: UnitLength.feet)
let distanceToGo = Measurement(value: 5, unit: UnitLength.kilometers)

var distanceMarker : String

if (distanceTraveled > distanceToGo) {
    distanceMarker = "Almost there!"
} else if (distanceTraveled < distanceToGo) {
    distanceMarker = "Barely started!"
} else {
    distanceMarker = "Halfway!"
}

print(distanceMarker)
```

“Barely started!”

Unit Definition

Unit Definition

In terms of base unit

Unit Definition

In terms of base unit

Methods to define conversion

Unit Definition

In terms of base unit

Methods to define conversion

Convert within dimension

Creating Units

Creating Units

Only define custom units

Conversion handled implicitly

```
// Create Custom Units on the Fly

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
```

```
// Create Custom Units on the Fly

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
```

Conversion

NEW

Conversion

NEW

baseUnit ⟷ unit

Conversion

NEW

baseUnit ⟷ unit

UnitConverter

- baseUnitValue(fromValue value:)
- value(fromBaseUnitValue baseUnitValue:)

Conversion

NEW

baseUnit ⟷ unit

UnitConverter

- baseUnitValue(fromValue value:)
- value(fromBaseUnitValue baseUnitValue:)

UnitConverterLinear

- baseUnitValue = value * coefficient + constant
- value = (baseUnitValue - constant)/coefficient

```
// “Jammin’ in the Streetz” Game - Custom Units

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
```

```
// “Jammin’ in the Streetz” Game - Custom Units

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
```

```
baseUnitValue = 30 * jamzValue
jamzValue = baseUnitValue/30
```

```
// “Jammin’ in the Streetz” Game - Custom Units

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
```

```
// “Jammin’ in the Streetz” Game - Custom Units

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
```

```
baseUnitValue = 0.75 * hopzValue
hopzValue = baseUnitValue/0.75
```

```
// “Jammin’ in the Streetz” Game - Custom Units

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
let glidez = UnitLength(symbol: “glidez", converter: UnitConverterLinear(coefficient: 1.5))
```

```swift
// “Jammin’ in the Streetz” Game - Custom Units

let jamz = UnitDuration(symbol: “jamz", converter: UnitConverterLinear(coefficient: 30))
let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
let glidez = UnitLength(symbol: “glidez", converter: UnitConverterLinear(coefficient: 1.5))
```

```
baseUnitValue = 1.5 * glidezValue
glidezValue = baseUnitValue/1.5
```

```
// “Jammin’ in the Streetz” Game – Custom Dimension

public class UnitDanceMove : Dimension {
```

```
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                  converter: UnitConverterLinear(coefficient: 1))
```

```
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                  converter: UnitConverterLinear(coefficient: 1))
```

1 💪 = 1 💪

```swift
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                  converter: UnitConverterLinear(coefficient: 1))
    static let robot = UnitDanceMove(symbol: "🤖",
                                     converter: UnitConverterLinear(coefficient: 4))
```

```
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                 converter: UnitConverterLinear(coefficient: 1))
    static let robot = UnitDanceMove(symbol: "🤖",
                                     converter: UnitConverterLinear(coefficient: 4))
```

1 🤖 = 4 💪

```swift
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                  converter: UnitConverterLinear(coefficient: 1))
    static let robot = UnitDanceMove(symbol: "🤖",
                                     converter: UnitConverterLinear(coefficient: 4))
    static let cabbagePatch = UnitDanceMove(symbol: "👼",
                                            converter: UnitConverterLinear(coefficient: 3))
```

```
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                 converter: UnitConverterLinear(coefficient: 1))
    static let robot = UnitDanceMove(symbol: "🤖",
                                     converter: UnitConverterLinear(coefficient: 4))
    static let cabbagePatch = UnitDanceMove(symbol: "👼",
                                            converter: UnitConverterLinear(coefficient: 3))
```

1 👼 = 3 💪

```swift
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                 converter: UnitConverterLinear(coefficient: 1))
    static let robot = UnitDanceMove(symbol: "🤖",
                                     converter: UnitConverterLinear(coefficient: 4))
    static let cabbagePatch = UnitDanceMove(symbol: "👼",
                                            converter: UnitConverterLinear(coefficient: 3))
    static let jazzHands = UnitDanceMove(symbol: "👐",
                                         converter: UnitConverterLinear(coefficient: 2))
}
```

```swift
// “Jammin’ in the Streetz” Game - Custom Dimension

public class UnitDanceMove : Dimension {
    static let wackyArmMovements = UnitDanceMove(symbol: "💪",
                                                 converter: UnitConverterLinear(coefficient: 1))
    static let robot = UnitDanceMove(symbol: "🤖",
                                     converter: UnitConverterLinear(coefficient: 4))
    static let cabbagePatch = UnitDanceMove(symbol: "👼",
                                            converter: UnitConverterLinear(coefficient: 3))
    static let jazzHands = UnitDanceMove(symbol: "👐",
                                         converter: UnitConverterLinear(coefficient: 2))
}
```

1 👐 = 2 💪

```
// “Jammin’ in the Streetz” Game – Jam Session

public struct JamSession {
```

```
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
```

```swift
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
```

.hopz

```
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
```

```
// “Jammin’ in the Streetz” Game – Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
```

.jamz

```
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
    public var danceMoves : Measurement<UnitDanceMove>
```

```
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
    public var danceMoves : Measurement<UnitDanceMove>
```

```
.robot
```

```
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
    public var danceMoves : Measurement<UnitDanceMove>
    public var danceRate : Measurement<UnitSpeed> {
```

```swift
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
    public var danceMoves : Measurement<UnitDanceMove>
    public var danceRate : Measurement<UnitSpeed> {
```

```
.metersPerSecond
```

```
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
    public var danceMoves : Measurement<UnitDanceMove>
    public var danceRate : Measurement<UnitSpeed> {
        let stepsInMeters = stepsTaken.converted(to: .meters)
```

```
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
    public var danceMoves : Measurement<UnitDanceMove>
    public var danceRate : Measurement<UnitSpeed> {
        let stepsInMeters = stepsTaken.converted(to: .meters)
        let jamTimeInSeconds = jamTime.converted(to: .seconds)
```

```swift
// “Jammin’ in the Streetz” Game - Jam Session

public struct JamSession {
    public var stepsTaken : Measurement<UnitLength>
    public var jamTime : Measurement<UnitDuration>
    public var danceMoves : Measurement<UnitDanceMove>
    public var danceRate : Measurement<UnitSpeed> {
        let stepsInMeters = stepsTaken.converted(to: .meters)
        let jamTimeInSeconds = jamTime.converted(to: .seconds)
        return Measurement(value: (stepsInMeters.value / jamTimeInSeconds.value),
                           unit: .metersPerSecond)
    }
}
```

Formatting Measurements

5 km

Formatting Is Hard

Country	Expected String

Formatting Is Hard

Country	Expected String
Canada	“5 km”

Formatting Is Hard

Country	Expected String
Canada	“5 km”
China	“5公里“

Formatting Is Hard

Country	Expected String
Canada	“5 km”
China	“5公里“
Egypt	“٥ كم“

Formatting Is Hard

Country	Expected String
Canada	“5 km”
China	“5公里“
Egypt	“٥ كم“
United States	“3.1 mi”

Let Us Do the Work

Let Us Do the Work

New formatter

Let Us Do the Work

New formatter

Measurements and units

Let Us Do the Work

New formatter

Measurements and units

Locale-aware formatting

```
// MeasurementFormatter

public class MeasurementFormatter : Formatter, NSSecureCoding {
```

NEW

```swift
// MeasurementFormatter

public class MeasurementFormatter : Formatter, NSSecureCoding {
    public var unitOptions: MeasurementFormatter.UnitOptions
```

NEW

```swift
// MeasurementFormatter

public class MeasurementFormatter : Formatter, NSSecureCoding {
    public var unitOptions: MeasurementFormatter.UnitOptions
    public var unitStyle: Formatter.UnitStyle
```

NEW

```swift
// MeasurementFormatter

public class MeasurementFormatter : Formatter, NSSecureCoding {
    public var unitOptions: MeasurementFormatter.UnitOptions
    public var unitStyle: Formatter.UnitStyle
    @NSCopying public var locale: Locale!
```

NEW

```swift
// MeasurementFormatter

public class MeasurementFormatter : Formatter, NSSecureCoding {
    public var unitOptions: MeasurementFormatter.UnitOptions
    public var unitStyle: Formatter.UnitStyle
    @NSCopying public var locale: Locale!
    @NSCopying public var numberFormatter: NumberFormatter!
```

NEW

```swift
// MeasurementFormatter

public class MeasurementFormatter : Formatter, NSSecureCoding {
    public var unitOptions: MeasurementFormatter.UnitOptions
    public var unitStyle: Formatter.UnitStyle
    @NSCopying public var locale: Locale!
    @NSCopying public var numberFormatter: NumberFormatter!
    public func string(from measurement: Measurement<Unit>) -> String
```

NEW

```
// MeasurementFormatter
```

NEW

```
public class MeasurementFormatter : Formatter, NSSecureCoding {
    public var unitOptions: MeasurementFormatter.UnitOptions
    public var unitStyle: Formatter.UnitStyle
    @NSCopying public var locale: Locale!
    @NSCopying public var numberFormatter: NumberFormatter!
    public func string(from measurement: Measurement<Unit>) -> String
    public func string(from unit: Unit) -> String
}
```

Unit Options

Unit Options

Formats preferred unit of locale by default

Takes purpose into account

Unit Options

UnitOptions	Measurement	Locale	Example String

Unit Options

UnitOptions	Measurement	Locale	Example String
.providedUnit	value: 5, unit: .kilometers	“en_US”	“5 km”

Unit Options

UnitOptions	Measurement	Locale	Example String
.providedUnit	value: 5, unit: .kilometers	“en_US”	“5 km”
.naturalScale	value: 1000, unit: .meters	“fr_FR”	“1 km”

Unit Options

UnitOptions	Measurement	Locale	Example String
.providedUnit	value: 5, unit: .kilometers	“en_US”	“5 km”
.naturalScale	value: 1000, unit: .meters	“fr_FR”	“1 km”
.temperatureWithoutUnit	value: 90, unit: .fahrenheit	“en_US”	“90°“

```
// “Jammin’ in the Streetz” Game - Formatting Units

let formatter = MeasurementFormatter()
```

```swift
// “Jammin’ in the Streetz” Game - Formatting Units

let formatter = MeasurementFormatter()
let distance = Measurement(value: 5, unit: UnitLength.kilometers)
```

```swift
// “Jammin’ in the Streetz” Game - Formatting Units

let formatter = MeasurementFormatter()
let distance = Measurement(value: 5, unit: UnitLength.kilometers)
let result = formatter.string(from: distance)
```

```swift
// “Jammin’ in the Streetz” Game - Formatting Units

let formatter = MeasurementFormatter()
let distance = Measurement(value: 5, unit: UnitLength.kilometers)
let result = formatter.string(from: distance)
```

“3.1 mi”

```
// “Jammin’ in the Streetz” Game - Formatting Custom Units

let formatter = MeasurementFormatter()
let distance = Measurement(value: 5, unit: UnitLength.kilometers)
let result = formatter.string(from: distance)

let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
```

```
// “Jammin’ in the Streetz” Game - Formatting Custom Units

let formatter = MeasurementFormatter()
let distance = Measurement(value: 5, unit: UnitLength.kilometers)
let result = formatter.string(from: distance)

let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
let hopzDistance = Measurement(value: 1000, unit: hopz)
```

```
// “Jammin’ in the Streetz” Game - Formatting Custom Units

let formatter = MeasurementFormatter()
let distance = Measurement(value: 5, unit: UnitLength.kilometers)
let result = formatter.string(from: distance)

let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
let hopzDistance = Measurement(value: 1000, unit: hopz)
let resultingHopz = formatter.string(from: hopzDistance)
```

```swift
// “Jammin’ in the Streetz” Game - Formatting Custom Units

let formatter = MeasurementFormatter()
let distance = Measurement(value: 5, unit: UnitLength.kilometers)
let result = formatter.string(from: distance)

let hopz = UnitLength(symbol: “hopz", converter: UnitConverterLinear(coefficient: 0.75))
let hopzDistance = Measurement(value: 1000, unit: hopz)
let resultingHopz = formatter.string(from: hopzDistance)
```

“0.466 mi”

```
// “Jammin’ in the Streetz” Game - Formatting Provided Unit

formatter.unitOptions = [.providedUnit]
```

```
// “Jammin’ in the Streetz” Game - Formatting Provided Unit

formatter.unitOptions = [.providedUnit]
let hopzDistance = Measurement(value: 1000, unit: hopz)
```

```swift
// “Jammin’ in the Streetz” Game - Formatting Provided Unit

formatter.unitOptions = [.providedUnit]
let hopzDistance = Measurement(value: 1000, unit: hopz)
let resultingHopz = formatter.string(from: hopzDistance)
```

```swift
// “Jammin’ in the Streetz” Game - Formatting Provided Unit

formatter.unitOptions = [.providedUnit]
let hopzDistance = Measurement(value: 1000, unit: hopz)
let resultingHopz = formatter.string(from: hopzDistance)
```

“1000 hopz”

```
// “Jammin’ in the Streetz” Game - Formatting Provided Unit

formatter.unitOptions = [.providedUnit]
let hopzDistance = Measurement(value: 1000, unit: hopz)
let resultingHopz = formatter.string(from: hopzDistance)

let robotDance = Measurement(value: 30, unit: UnitDanceMove.robot)
```

```
// “Jammin’ in the Streetz” Game - Formatting Provided Unit

formatter.unitOptions = [.providedUnit]
let hopzDistance = Measurement(value: 1000, unit: hopz)
let resultingHopz = formatter.string(from: hopzDistance)

let robotDance = Measurement(value: 30, unit: UnitDanceMove.robot)
let resultingRobotDances = formatter.string(from: robotDance)
```

```swift
// “Jammin’ in the Streetz” Game - Formatting Provided Unit

formatter.unitOptions = [.providedUnit]
let hopzDistance = Measurement(value: 1000, unit: hopz)
let resultingHopz = formatter.string(from: hopzDistance)

let robotDance = Measurement(value: 30, unit: UnitDanceMove.robot)
let resultingRobotDances = formatter.string(from: robotDance)
```

“30 🤖”

Demo

Displaying locale-aware measurements in your app

Peter Hosey Software Engineer, Foundation

Summary

Summary

New model objects

Summary

New model objects

MeasurementFormatter for formatting

Summary

New model objects

MeasurementFormatter for formatting

Powerful localization for free

More Information

https://developer.apple.com/wwdc16/238

Related Sessions

Internationalization Best Practices	Mission	Tuesday 9:00AM
What's New in Cocoa	Nob Hill	Tuesday 11:00AM
What's New in Foundation for Swift	Mission	Tuesday 4:00PM

WWDC16