Unit Conversion in Swift

Knoxville CocoaHeads

April 2019

Gavin Wiggins

~

https://gavinw.me

But there are plenty of unit converter apps 🤷

Why make another one?

- Online unit converters are not comprehensive, just the basics
- Online converters are not customizable
- Conversion is typically not automatic
- Most applications don’t account for localization

Foundation provides Units and Measurement types


```
let a = Measurement(value: 2.8, unit: UnitLength.kilometers)
let b = Measurement(value: 400.1, unit: UnitLength.centimeters)

let x = a.converted(to: .meters)    // 2800.041 m
let y = a + b                       // 2804.001 m
```

Added to Foundation in iOS 10 and macOS 10.12.

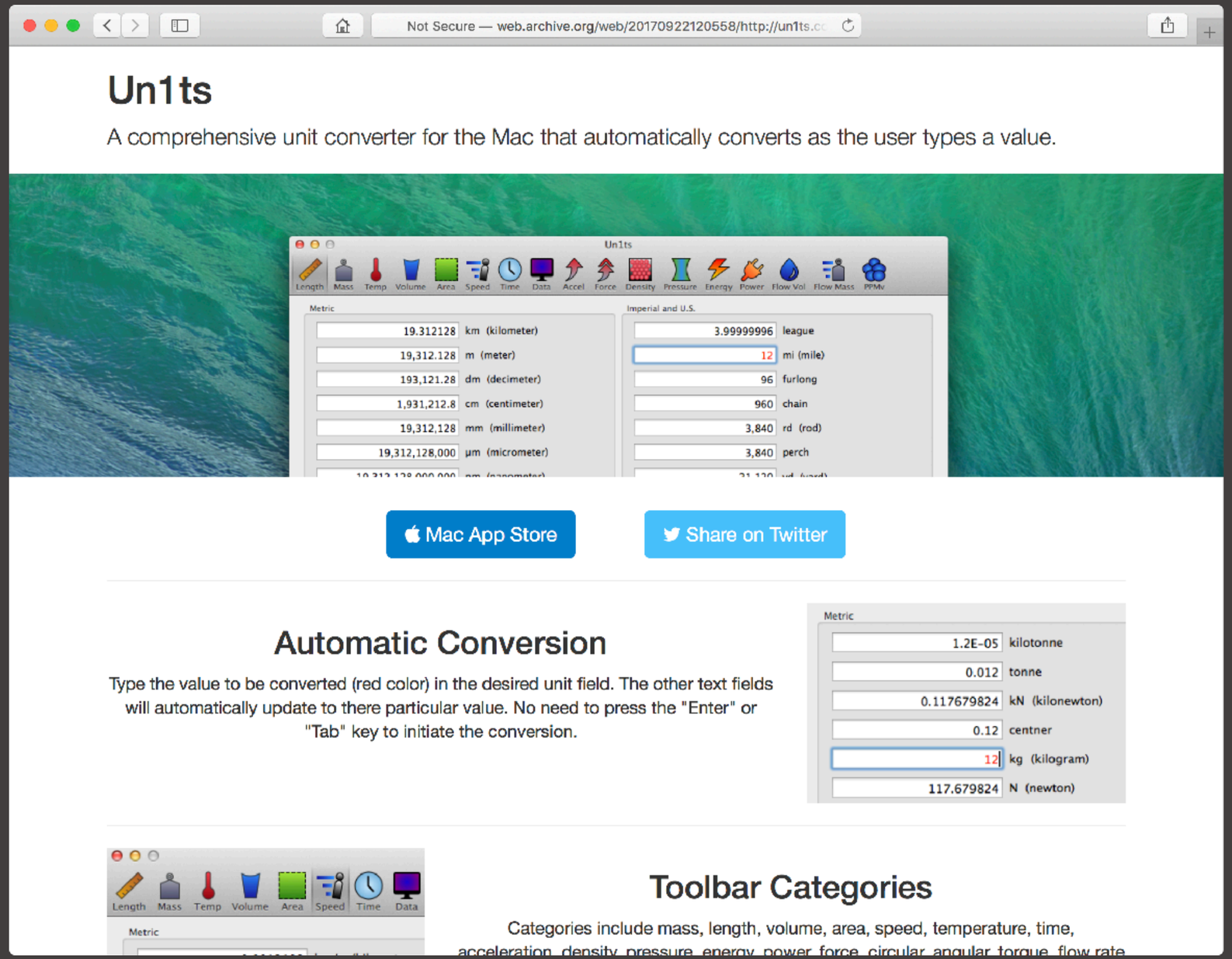
Compiler checks for conversion errors, e.g. can’t add length to temperature.

The new MeasurementFormatter formats measurements as locale-specific.

API is verbose and can be tedious to write.

Thanks Apple, but it’s more fun to create my own unit conversion types 🤪

My previous Mac unit converter written in Objective-C



Time for a new macOS unit converter app written in Swift

First approach to unit conversion in a macOS app

Each category (length, mass, etc.) has its own view and view controller.

NSTextFields for each unit.

Delegate method to determine when text field changes.

Identifier in Interface Builder for each text field associated with a unit.

Number formatter for each text field.

```
func controlTextDidChange(_ obj: Notification) {
    guard let txtField = obj.object as? NSTextField else { return }
    guard let id = txtField.identifier?.rawValue else { return }

    switch id {
    case "kilometers":
        convertFromKilometers(txtField.stringValue)
    case "meters":
        convertFromMeters(txtField.stringValue)
    case "centimeters":
        convertFromCentimeters(txtField.stringValue)
    default:
        print("id unknown")
    }
}
```

```
func convertFromKilometers(_ s: String) {
    guard let kilometers = Float(s) else { return }
    mTextField.stringValue = "\(kilometers * 1000)"
    cmTextField.stringValue = "\(kilometers * 100_000)"
    ydTextField.stringValue = "\(kilometers * 1_093.61)"
    ftTextField.stringValue = "\(kilometers * 3_280.84)"
    inTextField.stringValue = "\(kilometers * 39_370.1)"
}
```

Too much copy and paste.

This ain’t gonna cut it. 😒

Swifty approach using enum and struct

Each category (length, mass, etc.) has its own view and view controller.

NSTextfields for each unit.

Delegate method to determine when text field changes.

Identifier for each text field associated with a unit.

Number formatter for each text field.

Enum and struct for unit conversion.

Model units and conversion with enum and struct


```
enum LengthUnit: String {

    case kilometer
    case meter
    case centimeter
    case inch
    case foot
    case yard
    case mile

    var value: Float {
        switch self {
        case .kilometer: return 1_000
        case .meter: return 1
        case .centimeter: return 0.01
        case .inch: return 0.0254
        case .foot: return 0.3048
        case .yard: return 0.9144
        case .mile: return 1_609.344
        }
    }
}
```

```
struct Length {

    let value: Float
    let unit: LengthUnit

    func convert(to unit: LengthUnit) -> Float {
        return self.value * self.unit.value / unit.value
    }
}
```

View controller handles text fields and formatter


```
// Connect delegate and set identifier for each text field in IB.
@IBOutlet weak var kmTextField: NSTextField!
@IBOutlet weak var mTextField: NSTextField!
@IBOutlet weak var cmTextField: NSTextField!
@IBOutlet weak var ydTextField: NSTextField!
@IBOutlet weak var ftTextField: NSTextField!
@IBOutlet weak var inTextField: NSTextField!

let formatter = NumberFormatter()
var allFields = [NSTextField]()

override func viewDidLoad() {
    super.viewDidLoad()
    formatter.numberStyle = .decimal
    formatter.usesSignificantDigits = true

    // Array of all editable text fields in view (does not include labels).
    allFields = self.view.subviews.compactMap { $0 as? NSTextField }.filter { $0.isEditable }
}
```

Update text fields with appropriate units


```
func controlTextDidChange(_ obj: Notification) {

    guard let txtField = obj.object as? NSTextField,
        let txtId = txtField.identifier?.rawValue,
        let txtValue = Float(txtField.stringValue),
        let txtUnit = LengthUnit(rawValue: txtId) else { return }

    let x = Length(value: txtValue, unit: txtUnit)
    let fields = allFields.filter { $0 != txtField }

    for field in fields {
        if let id = field.identifier?.rawValue,
            let unit = LengthUnit(rawValue: id),
            let str = formatter.string(from: x.convert(to: unit) as NSNumber) {
            field.stringValue = str
        }
    }
}
```

Demo

Ready | Today at 11:44 AM

convert-example

```
/*
 Convert to meters (base unit) then to other units of length.

 Example to convert 32.8 cm -> ft
 32.8 cm * (0.01 m / 1 cm) * (1 m) / (0.3048 m / 1 ft)

 Example to convert 32.8 cm -> m
 32.8 cm * (0.01 m / 1 cm) * (1 m) / (1 m / 1 m)

 General equation where x -> y
 y = x * z / t
 */

enum LengthUnit: Float {
    case centimeter = 0.01
    case meter = 1
    case kilometer = 1_000
    case inch = 0.0254
    case foot = 0.3048
    case yard = 0.9144
}

struct Length {

    let value: Float
    let unit: LengthUnit

    func convertTo(unit: LengthUnit) -> Float {
        let v = self.value * self.unit.rawValue * LengthUnit.meter.rawValue / unit.rawValue
        return v
    }
}

let x1 = Length(value: 32.8, unit: .kilometer)
let y1 = x1.convertTo(unit: .meter)
```