

# Unit Conversion in Swift



Knoxville CocoaHeads

April 2019

Gavin Wiggins

~

<https://gavinw.me>

# But there are plenty of unit converter apps 🙄

---

2

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

3

```
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.

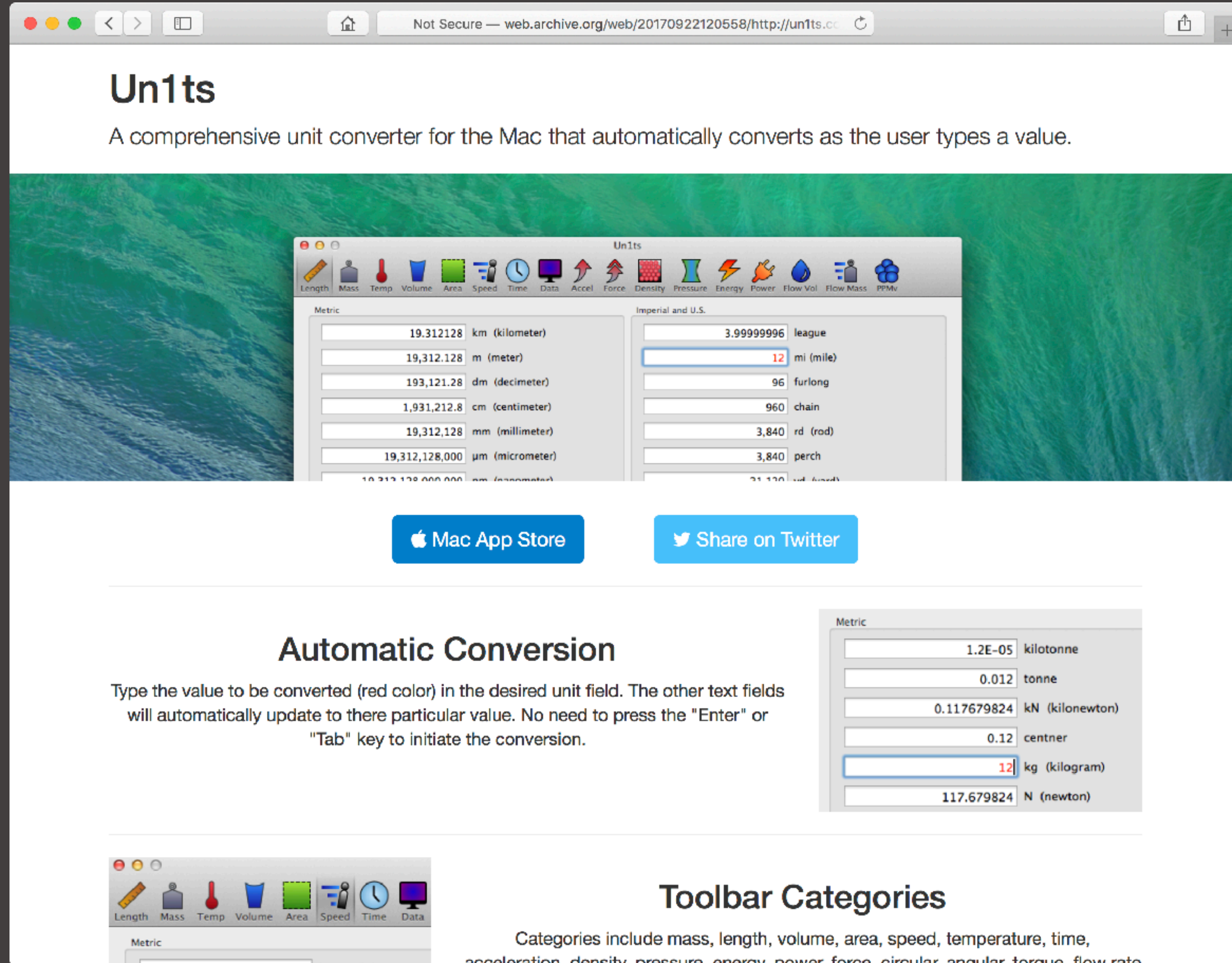
The new MeasurementFormatter formats measurements as locale-specific.

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 4



The screenshot displays the 'Un1ts' web application interface. At the top, the browser address bar shows the URL: `web.archive.org/web/20170922120558/http://un1ts.cc`. The main heading is 'Un1ts', followed by the description: 'A comprehensive unit converter for the Mac that automatically converts as the user types a value.'

The central image shows a screenshot of the 'Un1ts' application window on a Mac desktop. The window has a toolbar with icons for various unit categories: Length, Mass, Temp, Volume, Area, Speed, Time, Data, Accel, Force, Density, Pressure, Energy, Power, Flow Vol, Flow Mass, and PPMv. The main area is divided into two columns: 'Metric' and 'Imperial and U.S.'. The 'Metric' column shows a list of units with their corresponding values: km (kilometer) 19.312128, m (meter) 19,312.128, dm (decimeter) 193,121.28, cm (centimeter) 1,931,212.8, mm (millimeter) 19,312,128, μm (micrometer) 19,312,128,000, and nm (nanometer) 19,312,128,000,000. The 'Imperial and U.S.' column shows: league 3.99999996, mi (mile) 12, furlong 96, chain 960, rd (rod) 3,840, perch 3,840, and yd (yard) 31.12.

Below the application window screenshot, there are two buttons: 'Mac App Store' and 'Share on Twitter'.

The section titled 'Automatic Conversion' explains the functionality: 'Type the value to be converted (red color) in the desired unit field. The other text fields will automatically update to their particular value. No need to press the "Enter" or "Tab" key to initiate the conversion.'

To the right of this text is a screenshot of the 'Metric' column from the application, showing a list of units with their corresponding values: kilotonne 1.2E-05, tonne 0.012, kN (kilonewton) 0.117679824, centner 0.12, kg (kilogram) 12, and N (newton) 117.679824. The 'kg (kilogram)' field is highlighted with a red border, indicating it is the selected unit for conversion.

The section titled 'Toolbar Categories' shows a screenshot of the application's toolbar with icons for Length, Mass, Temp, Volume, Area, Speed, Time, and Data. Below the toolbar, the text states: 'Categories include mass, length, volume, area, speed, temperature, time, acceleration, density, pressure, energy, power, force, circular, angular, torque, flow rate.'

Time for a new macOS unit converter app  
written in Swift



# First approach to unit conversion in a macOS app

---

6

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")  
    }  
}
```



Too much copy and paste.

This ain't gonna cut it. 🙄

```
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)"  
}
```



# Swiftly approach using enum and struct

---

8

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

9

```
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

10

```
// 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

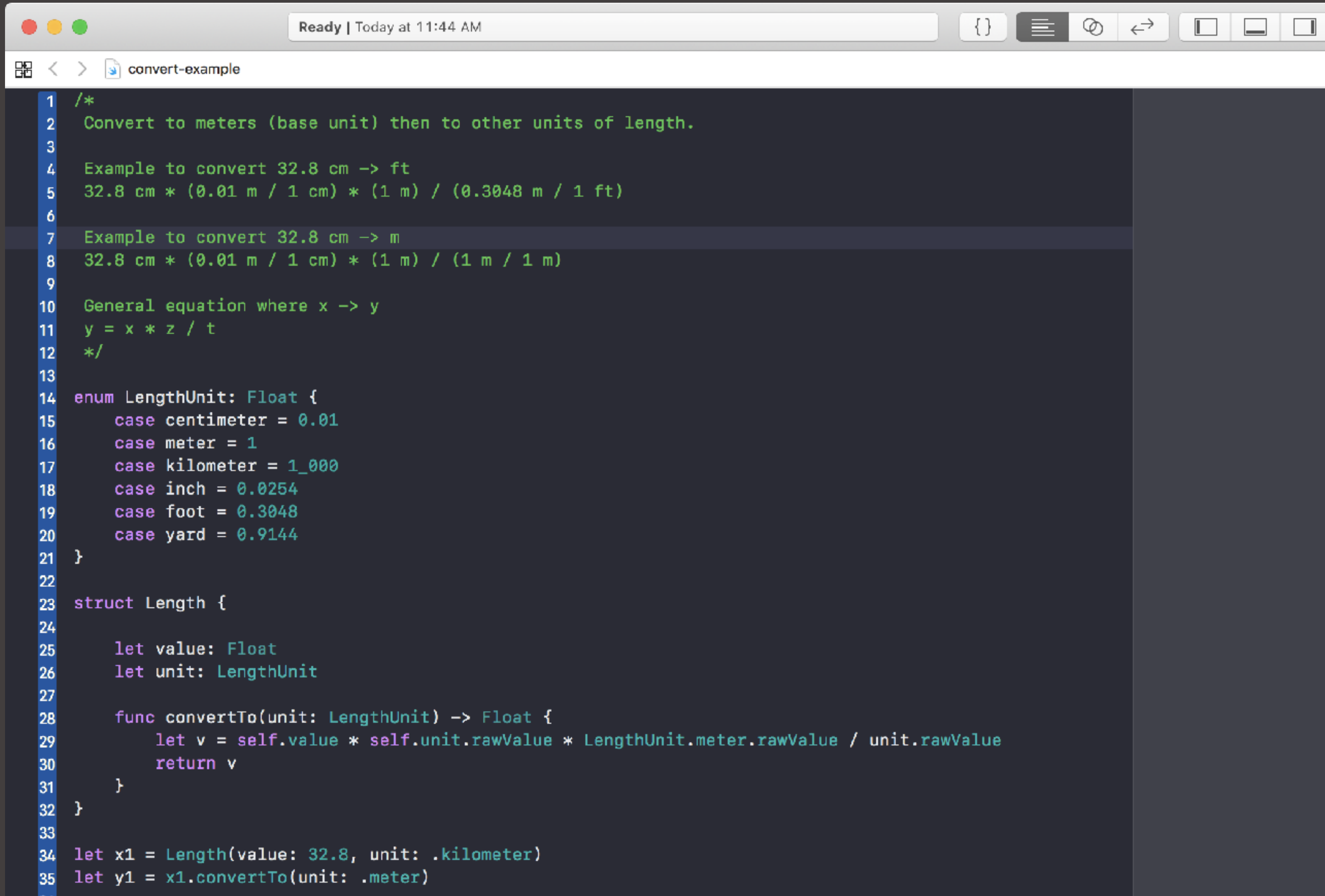
---

11

```
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

12



The screenshot shows a code editor window titled "convert-example" with a status bar indicating "Ready | Today at 11:44 AM". The code is written in Swift and implements a unit conversion system. It includes a multi-line comment explaining the purpose and providing examples, an enumeration for length units, a struct for length values with a conversion method, and two final lines demonstrating the usage of the struct.

```
1  /*
2   Convert to meters (base unit) then to other units of length.
3
4   Example to convert 32.8 cm -> ft
5   32.8 cm * (0.01 m / 1 cm) * (1 m) / (0.3048 m / 1 ft)
6
7   Example to convert 32.8 cm -> m
8   32.8 cm * (0.01 m / 1 cm) * (1 m) / (1 m / 1 m)
9
10  General equation where x -> y
11  y = x * z / t
12  */
13
14  enum LengthUnit: Float {
15      case centimeter = 0.01
16      case meter = 1
17      case kilometer = 1_000
18      case inch = 0.0254
19      case foot = 0.3048
20      case yard = 0.9144
21  }
22
23  struct Length {
24
25      let value: Float
26      let unit: LengthUnit
27
28      func convertTo(unit: LengthUnit) -> Float {
29          let v = self.value * self.unit.rawValue * LengthUnit.meter.rawValue / unit.rawValue
30          return v
31      }
32  }
33
34  let x1 = Length(value: 32.8, unit: .kilometer)
35  let y1 = x1.convertTo(unit: .meter)
```