Extracted from:

Xcode Treasures

Master the Tools to Design, Build, and Distribute Great Apps

The Pragmatic Bookshelf
Raleigh, North Carolina

The Pragmatic Programmers

Xcode Treasures

Master the Tools to Design, Build, and Distribute Great Apps

Chris Adamson

edited by Tammy Coron

Xcode Treasures

Master the Tools to Design, Build, and Distribute Great Apps

Chris Adamson

The Pragmatic Bookshelf
Raleigh, North Carolina

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Brian MacDonald
Development Editor: Tammy Coron
Copy Editor: Jasmine Kwityn
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-586-3
Book version: P1.0—October 2018

Coding with Snippets

One other handy tool for coding lives over on the right side of the window, in the bottom portion of of the Utilities Pane. In the small toolbar, click the curly-braces icon (or just press ^⌥⌘2) to show the Code Snippet Library. *Code snippets* are small, reusable code templates, meant to reduce boilerplate coding. They're also a big help when you can't quite remember the syntax for something, but have the general gist of how it works.

The provided snippets are arranged alphabetically, and their names start with their language, so you'll see C and Objective-C before you get to Swift snippets. You can also type into the Filter text field at the bottom to narrow things down to just Swift, or a keyword you're looking for. Click any of the snippets to pop up an info window showing a description of the snippet and its code. For example, the figure on page 6 shows the code snippet for a Swift computed variable, with its get and set blocks.

Using a snippet couldn't be easier: just drag it from the Code Snippet Library into your code and drop.

Creating Code Snippets

Of course, the Xcode team can't know which snippets will be useful to any given developer. What's easy for you to remember might be hard for me, or vice versa. Fortunately, you can easily create your own snippets. The process is simple: select and drag some code into the Code Snippet Library, then add some metadata to find it later.

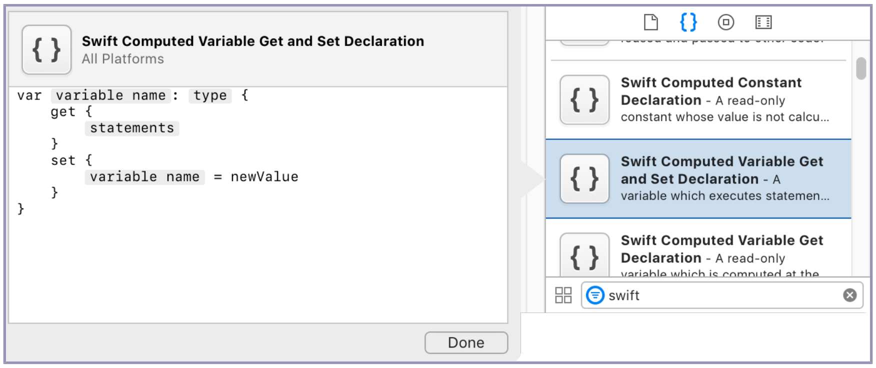
One piece of code I often write is the "Weak-Strong Dance", in which a weak self optional is unwrapped inside a closure, so there's a strong instance I can use (or I just exit early from the closure if self no longer exists). The reasons for this technique are explained much later, in *How the Strong-Delegate Memory Leak Turns Up in Closures,* on page ?. For now, have a look at the code:

```
foo.methodThatTakesAClosure() { [weak self] parameter1, parameter2, ... in
    guard let strongSelf = self else { return }

}
```

In time, you can learn to type this from memory, but there's a good argument to be made for not having to—especially when you could just make it a snippet. To do so, you just write this in the editor, select it, and drag it to the Code Snippets Library.

This creates a new snippet called My Code Snippet, and immediately pops up an editor that allows you to rename the snippet and enter some metadata. You


Swift Computed Variable Get and Set Declaration
All Platforms
Done
Swift Computed Constant Declaration - A read-only constant whose value is not calcu...
Swift Computed Variable Get and Set Declaration - A variable which executes statemen...
Swift Computed Variable Get Declaration - A read-only variable which is computed at the
swift

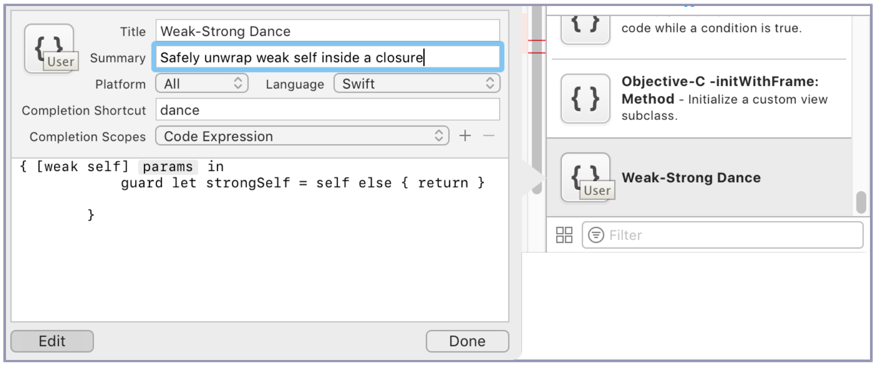
```
var variable name : type {
    get {
        statements
    }
    set {
        variable name  = newValue
    }
}
```

can set the language appropriate for the snippet, the platforms it can be used on, and more.

Most importantly, you can enter a completion shortcut that will offer the snippet as a code completion. While you can drag and drop a snippet anywhere, code completion will honor any metadata you enter for the snippet's language, supported platforms, and completion scopes. That way, if you declare that a snippet only applies at the top level of a file, it won't be offered as a completion while you're coding inside a method.

One other technique you can use when creating a snippet is that any text in the form <#placeholder#> will become placeholder text that you can tab onto and replace, just like when accepting auto-completion for a method or function call.

The following figure shows the snippet editor with the Weak-Strong Dance snippet filled out (note that while there's no visual indication in the editor, params is written with the previously described placeholder syntax):



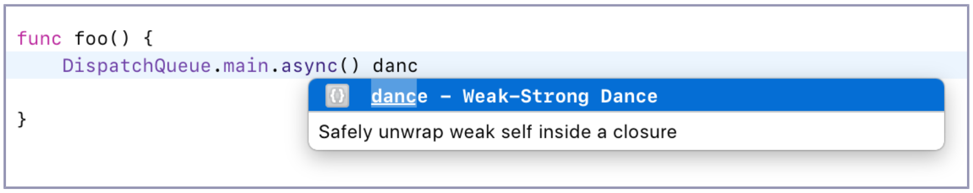
Title: Weak-Strong Dance
Summary: Safely unwrap weak self inside a closure
Platform: All
Language: Swift
Completion Shortcut: dance
Completion Scopes: Code Expression

```
{ [weak self] params  in
            guard let strongSelf = self else { return }

        }
```

User
Edit
Done
code while a condition is true.
Objective-C -initWithFrame: Method - Initialize a custom view subclass.
Weak-Strong Dance
User
Filter

Now, while you're coding, you no longer have to remember the fancy closure syntax. As shown in the following figure, all you need to do is start typing the word dance and you'll get the automatic code completion:



```
func foo() {
    DispatchQueue.main.async() danc
}
```

dance - Weak-Strong Dance

Safely unwrap weak self inside a closure

Any time you find yourself looking up boilerplate syntax (something you're going to use briefly and then not again for another couple months), you should probably consider whether it's worth a few minutes of your time to make it a code snippet. It could save you time and hassle later.