Second Edition

A Swift Kickstart

DANIEL H STEINBERG

Introducing the
Swift Programming Language

Editors Cut

Second Edition

A Swift Kickstart

DANIEL H STEINBERG

Introducing the
Swift Programming Language

Editors Cut

Copyright

"A Swift Kickstart" Second Edition, by Daniel H Steinberg

ISBN-13: 978-0-9830669-8-9

Recommended Settings

The ePub is best viewed in scrolling mode using the original fonts. The ePub and Mobi versions of this book are best read in single column view.

Legal

TABLE OF CONTENTS

CHAPTER 1

Let's Get Started

Welcome to the Swift 5 release of the second edition of this introductory book about the Swift Programming Language.

The book has been updated with every release of Swift. This was a light update as there isn't much new in Swift 5 for beginners whereas last years update for Swift 4.1 was significant.

I loved Swift from the moment Apple demonstrated it at WWDC 2014.

The language pushed me to improve my code in ways that now feel natural. With the changes introduced in Swift 3 and 4, I now write code that's both more concise and more expressive than what I was able to write before.

This isn't a comprehensive book covering every nook and cranny of Swift. This is a kickstart. It's designed to get you up and coding in this new language. That said, there's a lot of material in this short book. By the end, you'll know enough to go to the docs, forums, and other resources to learn the finer points.

I've updated this book for Swift 5 and Xcode 10 and added support for the latest Swift Playgrounds release. Many of the changes are based on reader feedback and from the input of my students as I've taught this material in dozens of cities, in many countries on three continents since July 2014.

This definitely isn't a book that you read and nod your head to. Code along with me either on an iPad or a Mac. Try things out and see what works and what doesn't work. Join in the fun and you'll be thinking in Swift in no time at all.

We'll start playing with code right away, working through examples using playgrounds, an interactive environment in Xcode 9 and now on iOS. Every year the engineers at Apple add wonderful new features to playgrounds. I love incorporating these new features in my courses and books.

To follow along with the examples, you'll either need to have access to a Mac or an iPad. Only a couple of sections require Swift 5 but I still recommend that if you code along with me on a Mac, it needs to be running the latest developer tools from Apple installed. Xcode 10 is available for free from the Mac App Store.

If you choose to use an iPad, it needs to be running iOS 12 and you'll need to download and install Swift Playgrounds from the App Store.

"Really," you ask, "Do I really need a Mac or an iPad? Isn't Swift open source? Couldn't I just use Linux or something?"

You can. But in this book, all of my descriptions and screenshots are either for Xcode 10 running on OS X 10.14 Mojave or for Swift Playgrounds running on iOS 12.

Please check http://developer.apple.com for additional resources including videos, sample code, documentation, and forums. You'll also find information on what is required to take advantage of these resources.

This book is for experienced developers who are either new to Swift or who haven't been keeping up with all of the changes to Swift. So much has changed in Swift 3 and 4 that it seems as if every line of code we wrote in the past couple of years looks different when we update it to the latest version of Swift. More than that, the way we think about our code has changed. We're beginning to understand what we mean when we say "That code is more Swifty."

Apple open-sourced Swift in December 2015 and involved the community in decisions of how to stabilize and expand the language. The conversations were amazing. We got to hear what Apple had intended with their language and the folks at Apple working on the language got to hear how we had used it.

As a result not only has all of the code has been updated and checked for compliance with Swift 5 as of Xcode 10, but I've also tried to embrace the best practices that came out of the community discussions.

I'm sure you'll enjoy this journey as much as I.

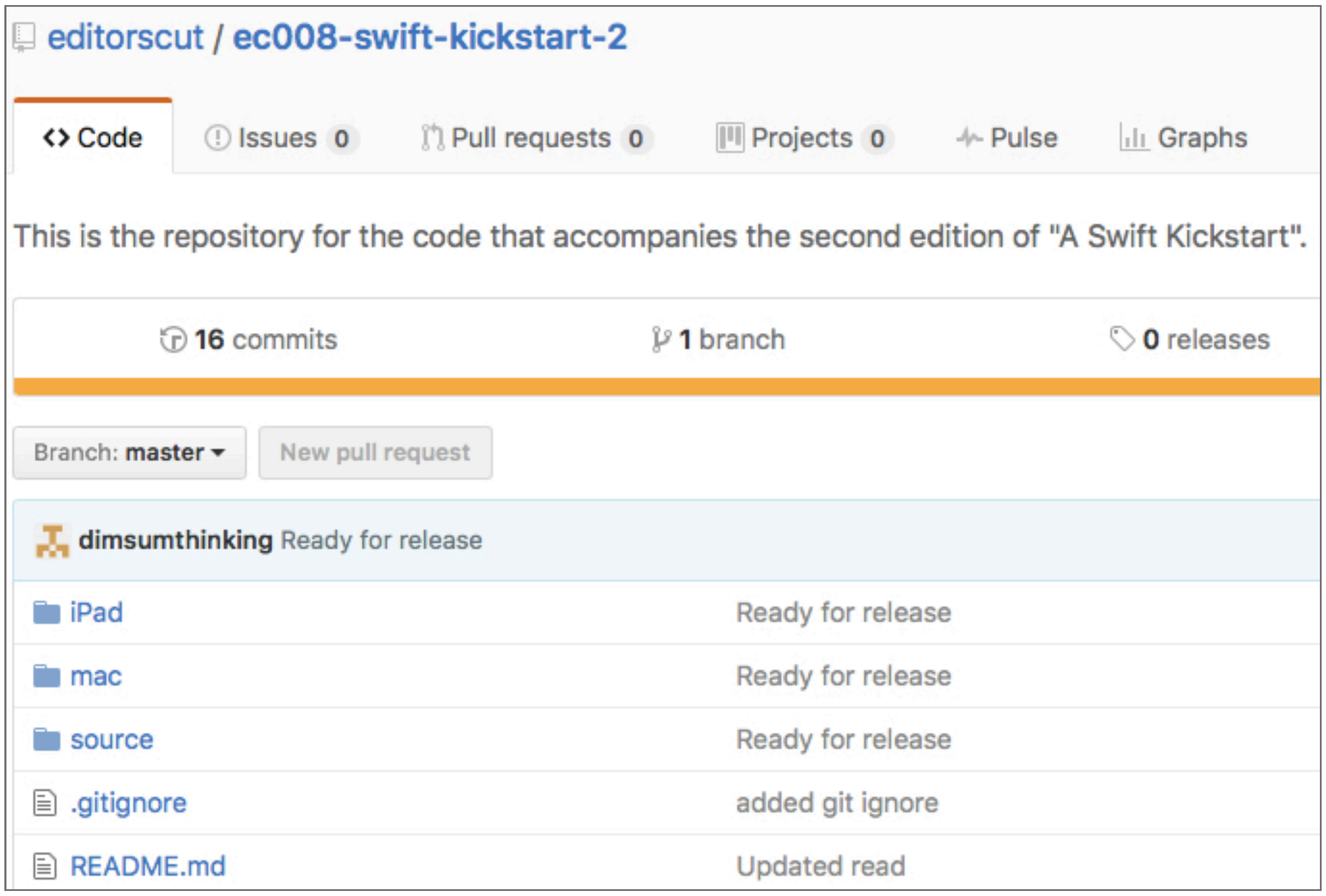
Let's start by downloading the necessary files and establishing some code conventions for this book.

Mac Quickstart

If you plan to code along on a Mac using Xcode Playgrounds, this section is for you.

If instead, you prefer to use an iPad, check out the next section for information on how to use Swift Playgrounds on an iPad.

You can find all of the code for this book at https://github.com/editorscut/ec008-swift-kickstart-2.



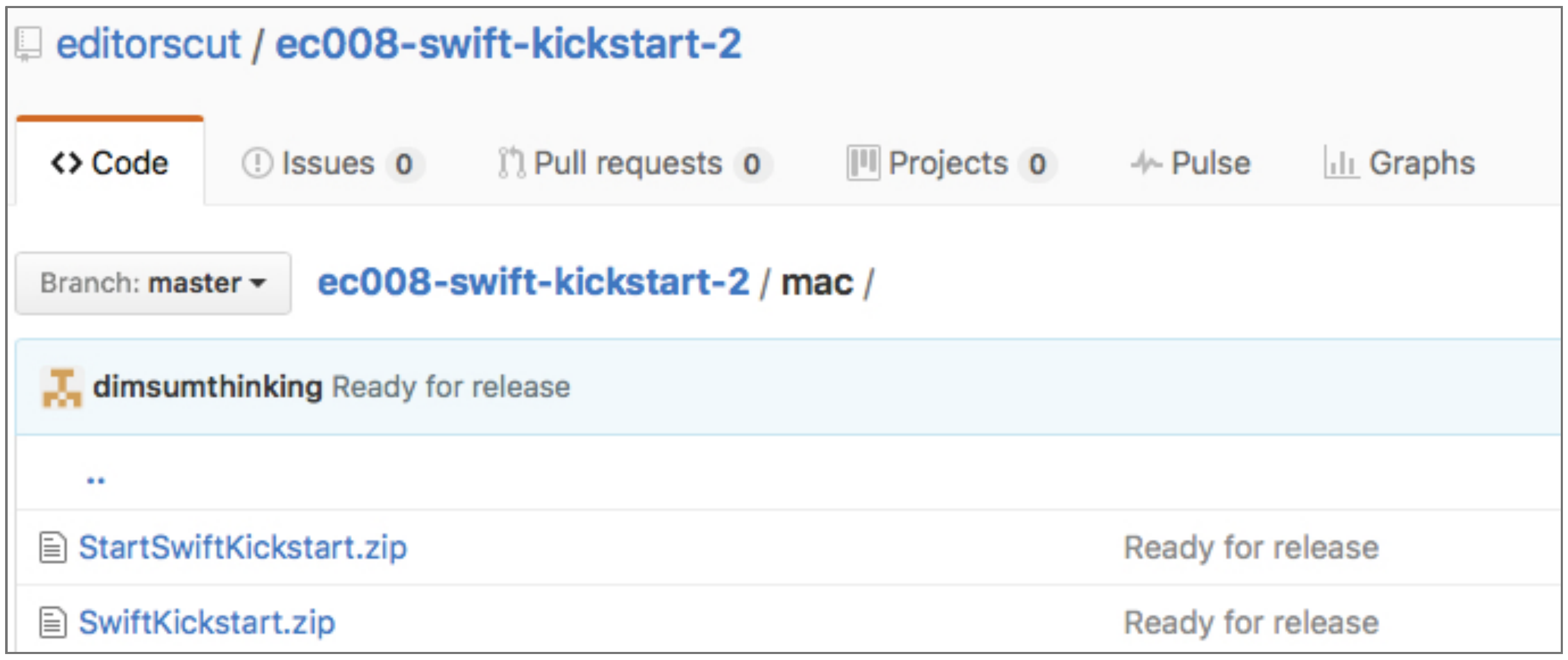
This is the repository for the code that accompanies the second edition of "A Swift Kickstart".

16 commits · 1 branch · 0 releases

dimsumthinking Ready for release

iPad	Ready for release
mac	Ready for release
source	Ready for release
.gitignore	added git ignore
README.md	Updated read

Select the *mac* folder to see the files that you'll need for coding on a Mac.

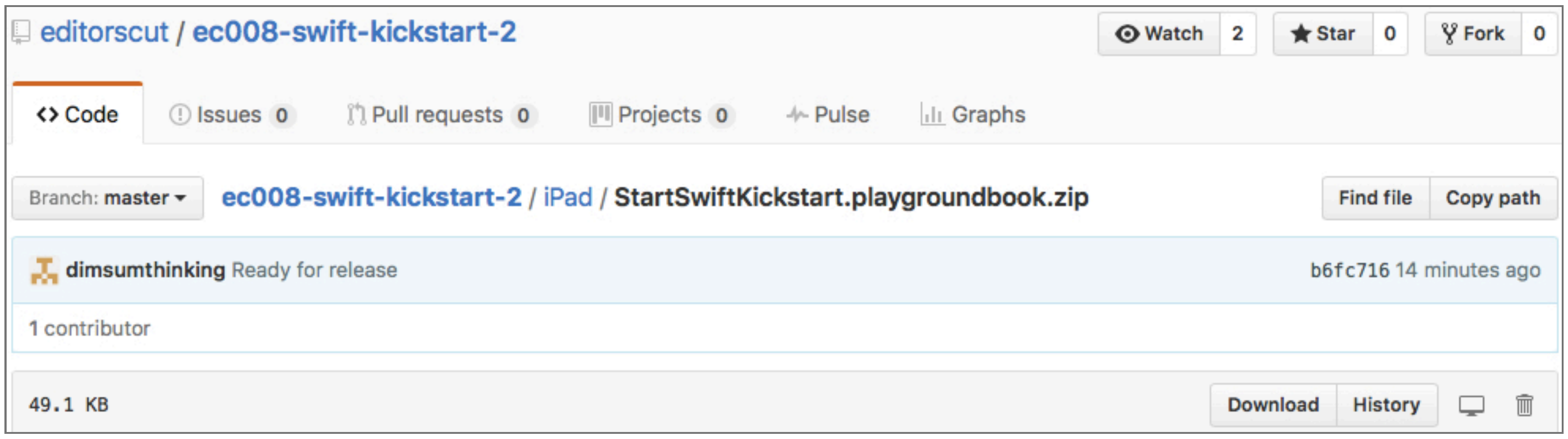

editorscut / ec008-swift-kickstart-2
<> Code
Issues 0
Pull requests 0
Projects 0
Pulse
Graphs
Branch: master
ec008-swift-kickstart-2 / mac /
dimsumthinking Ready for release
..
StartSwiftKickstart.zip Ready for release
SwiftKickstart.zip Ready for release

This directory contains two zip files. Each file unzips into a folder that contains one Xcode playground for each chapter of this book. Each playground contains a table of contents and a page for each section of the chapter. Here's the difference between the two:

- *StartSwiftKickstart.zip* contains pages that are mostly blank. The pages include some basic navigation and from time to time some code that you need from previous sections. To follow along with the book, start with the playgrounds in this folder.
- *SwiftKickstart.zip* contains pages that are filled with the code that you'll enter as you work through the corresponding section.

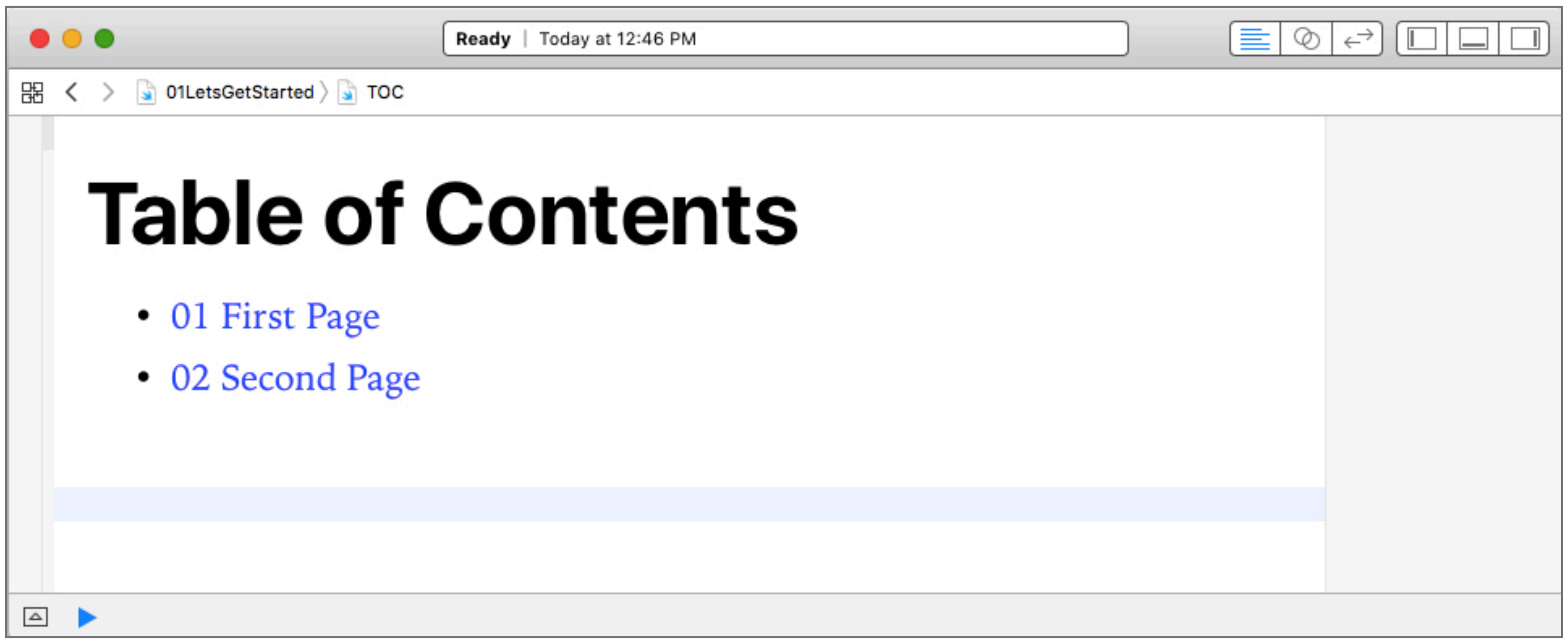
In other words, *StartSwiftKickstart* is the "before" — it's a template created for your convenience — and *SwiftKickstart* is the "after."

Click *StartSwiftKickstart.zip*. You'll be taken to this page.



Select the Download button in the button cluster in the lower-right corner to download the zip file to your Mac. Unzip the file. You should have a *StartSwiftKickstart* folder containing one playground for each chapter in this book.

Double-Click *01LetsGetStarted.playground* to open it with Xcode. You should see something like this.

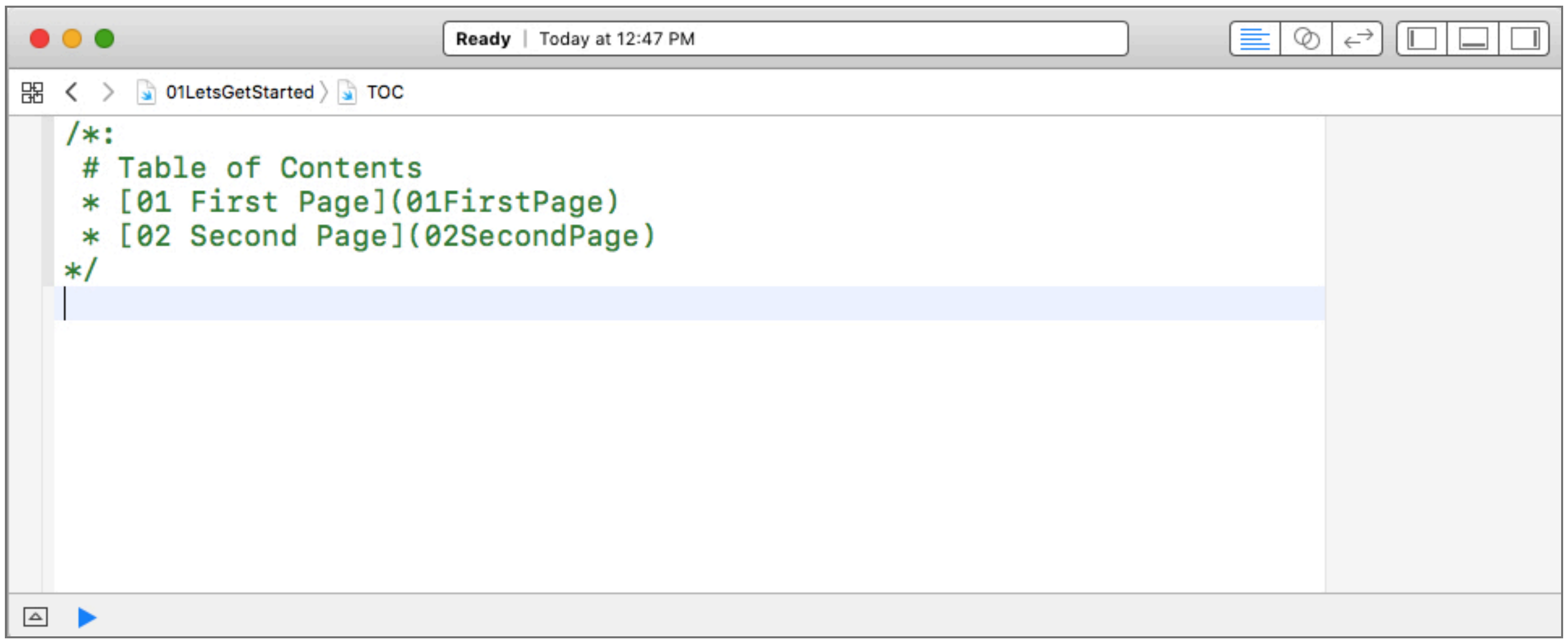

Ready | Today at 12:46 PM

01LetsGetStarted 〉 TOC

Table of Contents

- 01 First Page
- 02 Second Page

There's a slight chance that you'll instead see raw markup like this.


Ready | Today at 12:47 PM

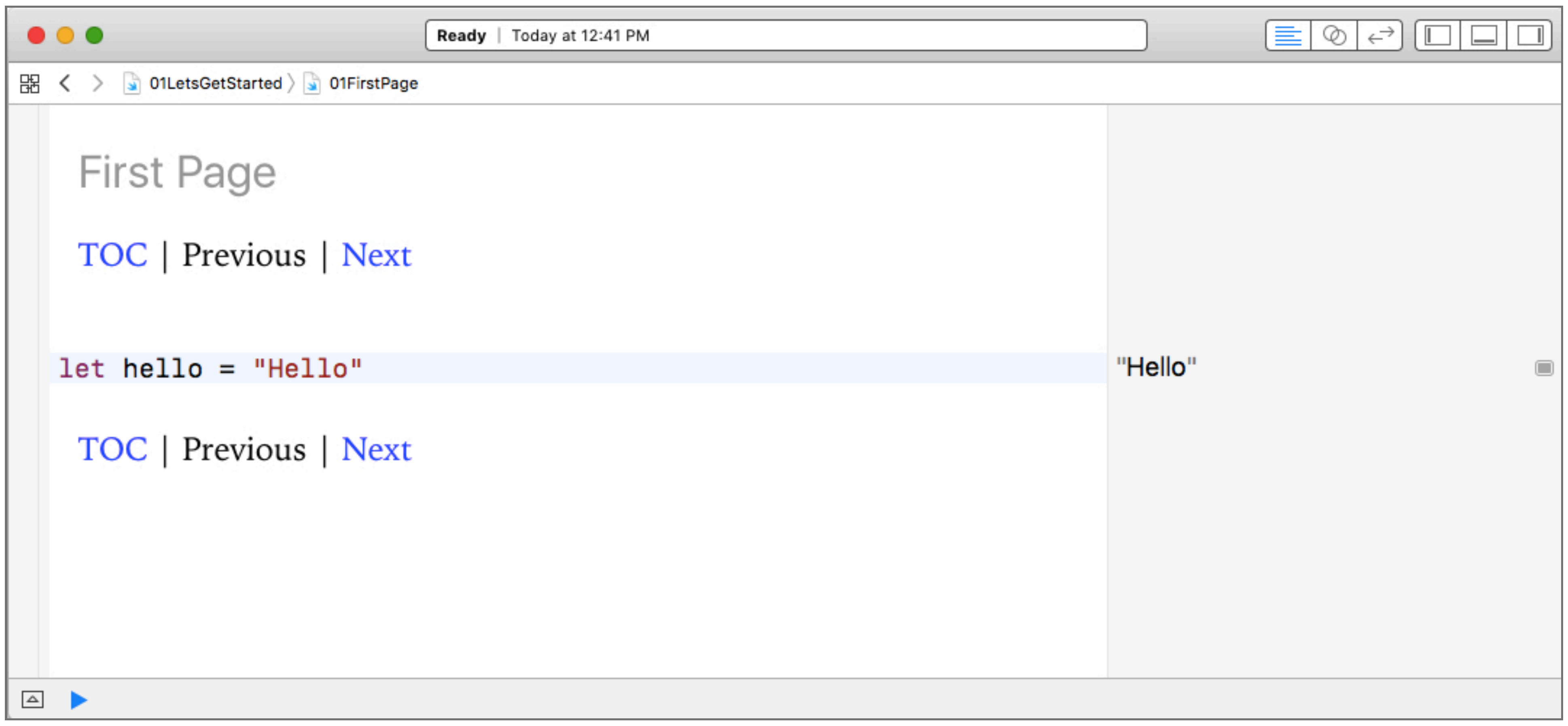
01LetsGetStarted 〉 TOC

```
/*:
 # Table of Contents
 * [01 First Page](01FirstPage)
 * [02 Second Page](02SecondPage)
*/
```

If you do, select the menu item Editor > Show Rendered Markup to view the rendered markup.

Click the links to navigate to the other pages in the playground. Navigate to *01FirstPage*. Place the cursor in between the two lines that begin "TOC" and type `let hello = "Hello"`. You're declaring a `String` named `hello` and assigning it the value `"Hello"`. We'll discuss what that means in greater depth soon enough.

You should see this on your playground page.



First Page

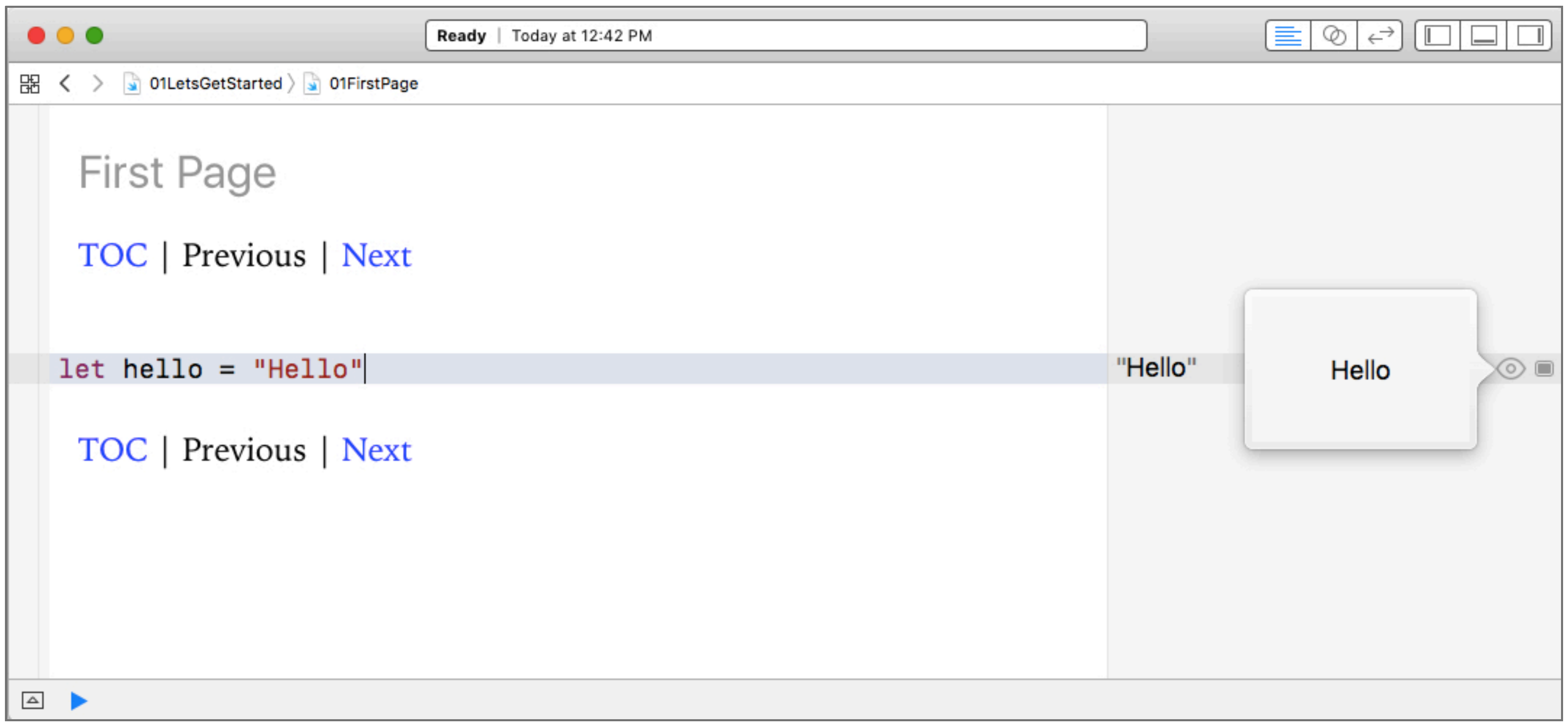
TOC | Previous | Next

```swift
let hello = "Hello"
```

"Hello"

TOC | Previous | Next

On the right side of the playground, you can see where the value of `hello` is echoed as `"Hello"`. Mouse over that line and you'll see two icons to the right of the echoed `"Hello"`. The eye-shaped one is for Quick Look. Click it and you'll see this popover.



First Page

TOC | Previous | Next

```swift
let hello = "Hello"
```

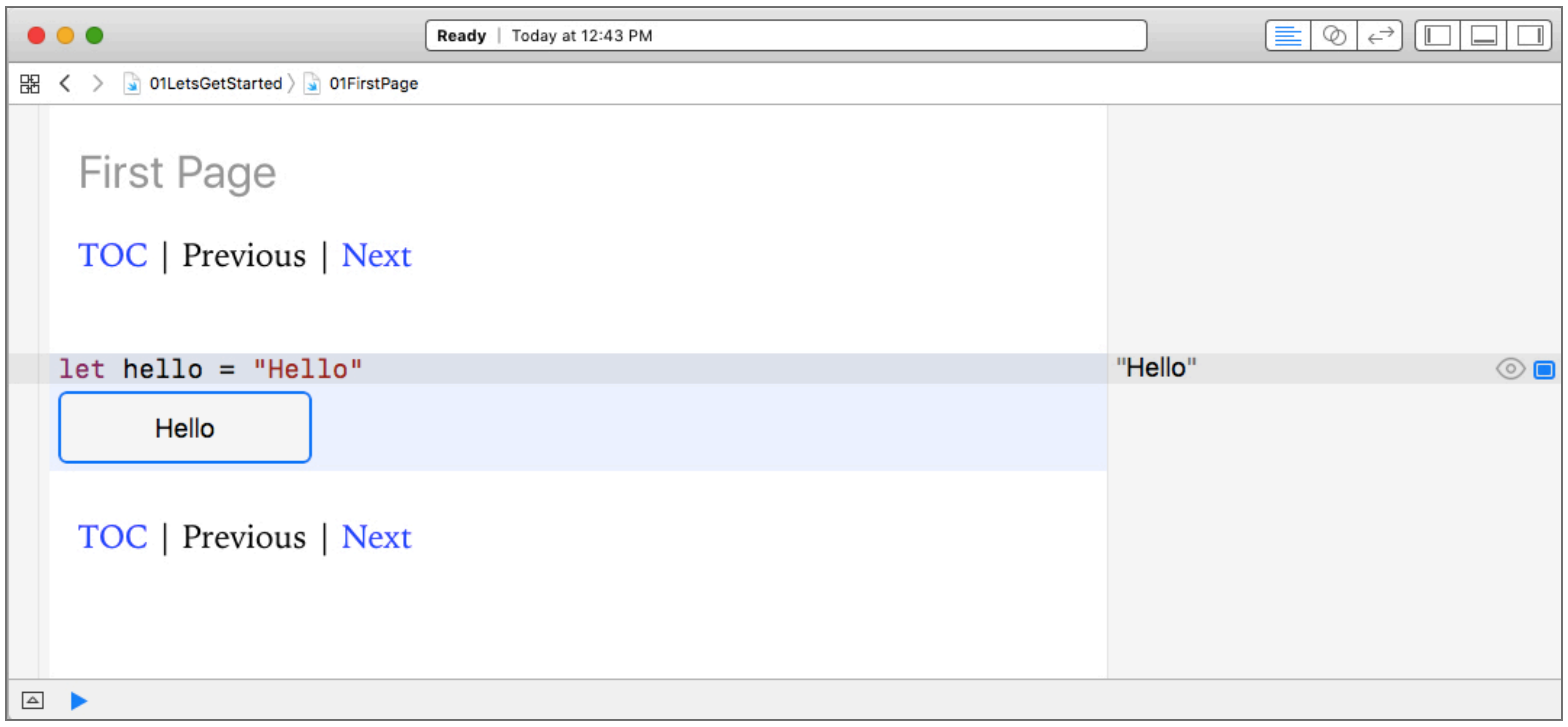
"Hello"

Hello

TOC | Previous | Next

In this case, `"Hello"` is just a `String`, so there's not much more to see.

The second icon is a rectangle. This is the ***Show Result*** icon. Click it and you'll see more detail underneath the code. The results box isn't framed in blue unless you select it. Go ahead and do so, and you'll see something like this.

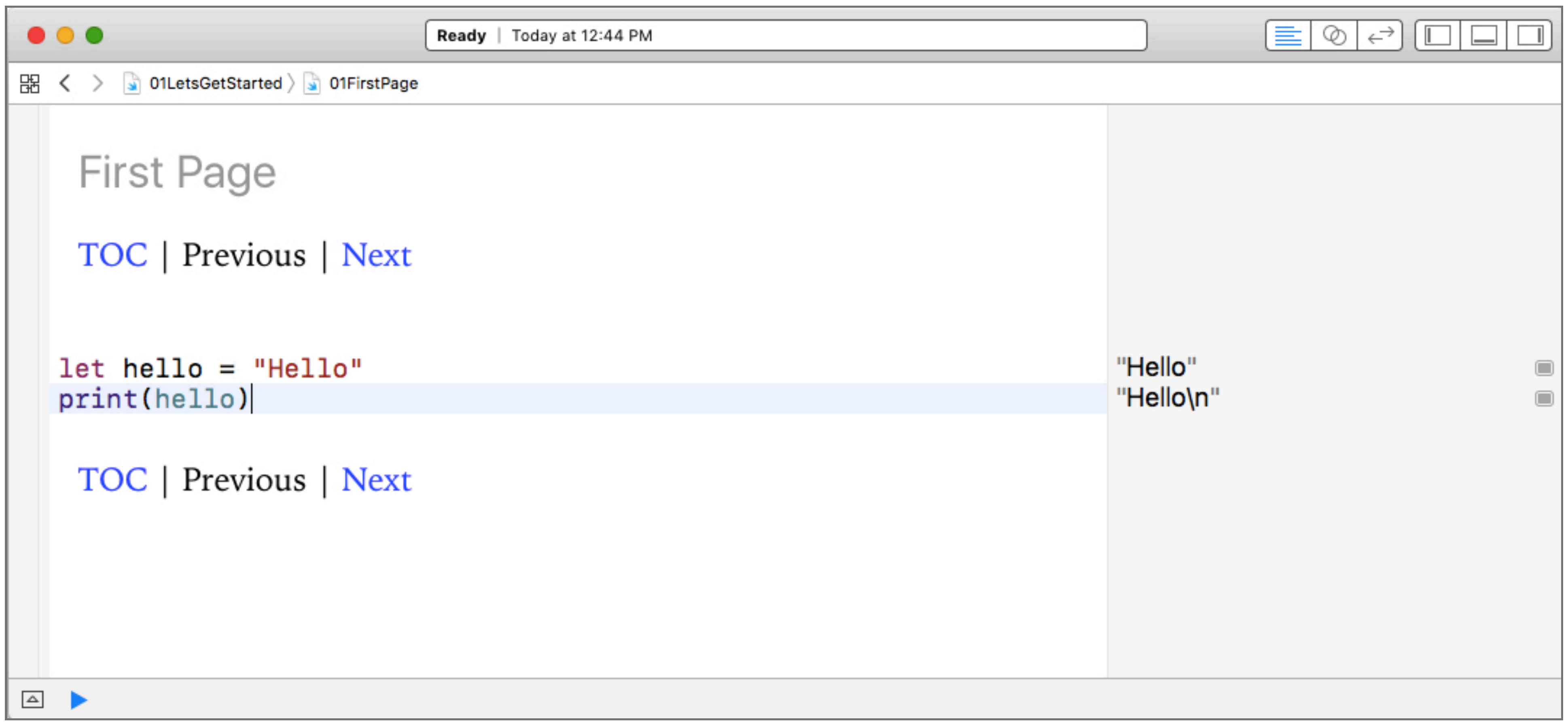


```
let hello = "Hello"
```

Dismiss the results box by tapping the rectangle again.

We've got one more stop on our quick tour: the console.

Type `print(hello)` in the line below `let hello = "Hello"`.



```
let hello = "Hello"
print(hello)
```

```
"Hello"
"Hello\n"
```

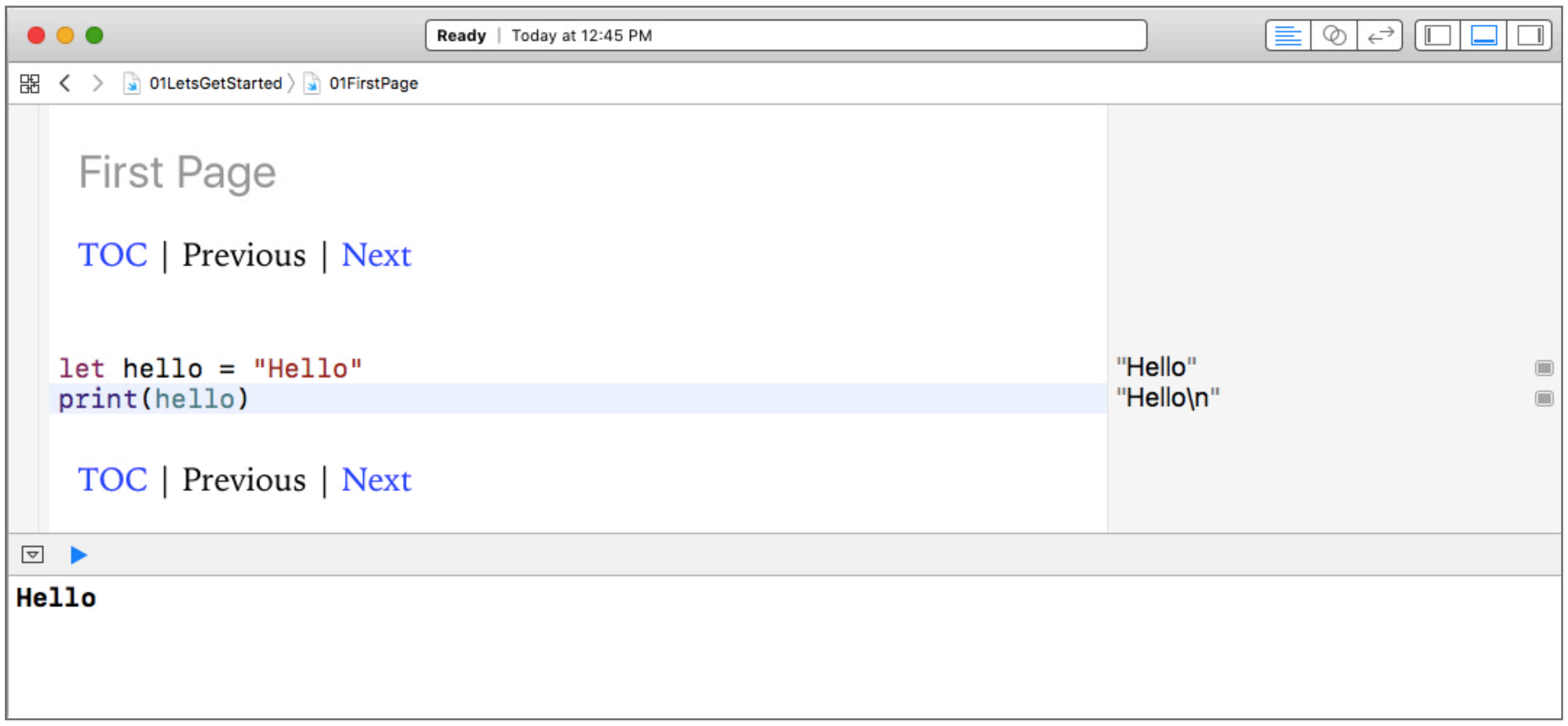
You should see `"Hello\n"` on the right side of the playground to indicate that the result of `print(hello)` is the `String` `"Hello"` followed by a newline character.

The `print()` command results in output to the console. To toggle the console, you can use the keyboard shortcut Shift - Command - Y.

You might prefer to use the middle of the three buttons at the top-right corner of Xcode.

Click the middle button to toggle the console to appear at the bottom of the playground. The button is highlighted in blue to indicate that it's selected.

You should now see the console at the bottom of the playground, and it should contain the output `Hello`.



First Page

TOC | Previous | Next

```
let hello = "Hello"
print(hello)
```

"Hello"
"Hello\n"

TOC | Previous | Next

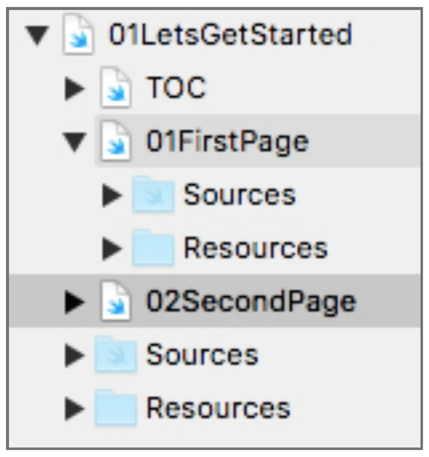
```
Hello
```

If you don't see console output as you work your way through various examples in this book, click the blue triangle to run the playground again.

That's it!

You can check your work by comparing these results to the completed playground in the *ThinkingInSwift* directory.

One more thing. Later in this book you're going to need to look at the file directory. You click the middle button to toggle the console. Click the left button to toggle the navigators.



Above you can see the *TOC*, *01FirstPage*, and *02SecondPage* hierarchy. There are also top-level *Source* and *Resource* directories. Also, when you click the disclosure triangle next to any of the pages, you see that each page has its own *Source* and *Resource* directories. In Chapter 8 on protocols you'll look at files that I'll place in the *Source* directory for one of the pages.

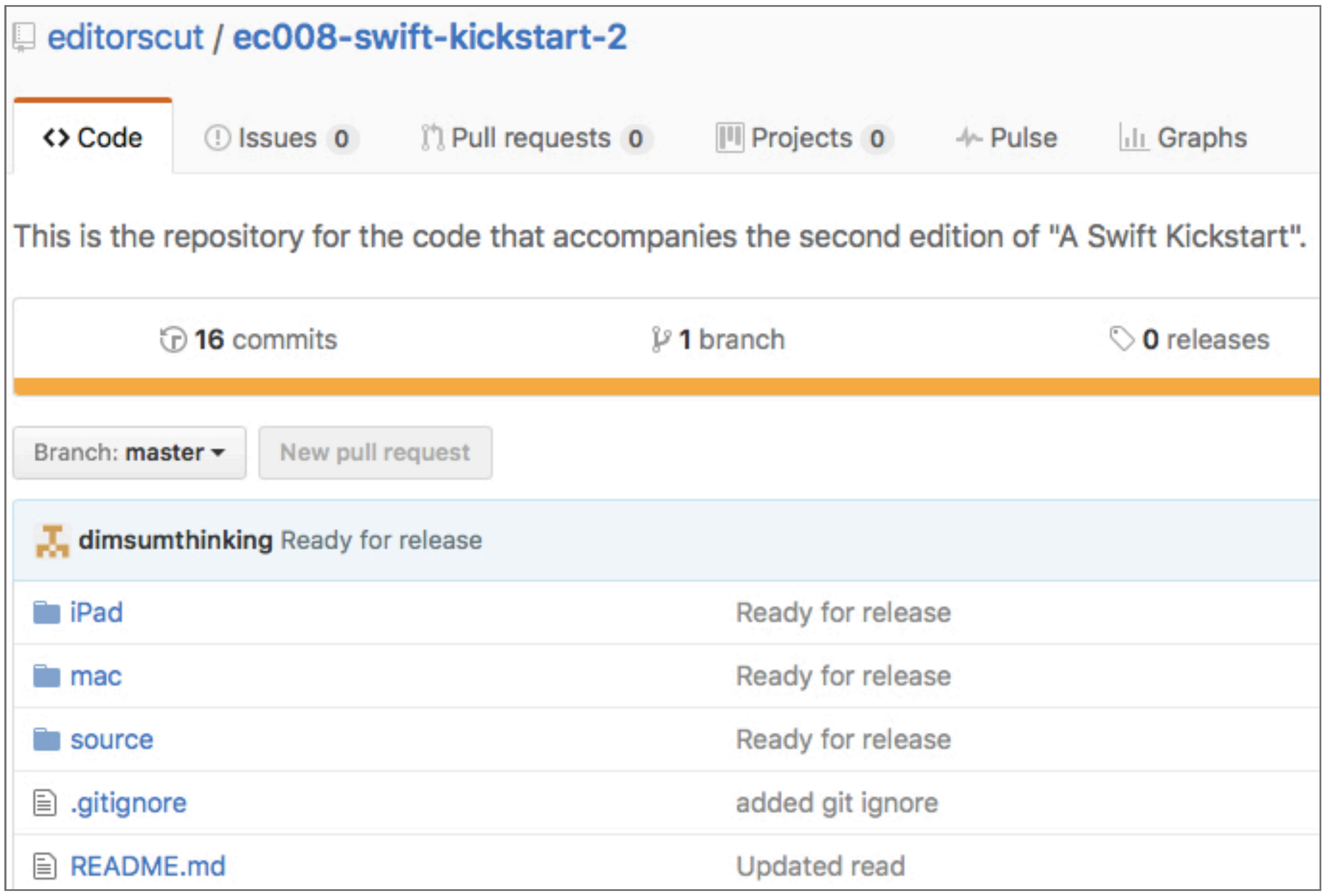
In the next section we'll follow similar steps to demonstrate how to code along on an iPad using Swift playgrounds. If you're going to only follow along on a Mac, you can skip this section.

iPad Quickstart

Use this section if you plan to code along on an iPad using Swift Playgrounds. Otherwise, check out the previous section for information on how to use Xcode Playgrounds on a Mac.

Before you go any further, you need to download and install Swift Playgrounds from the App Store on an iPad running iOS 10.

You can find all of the code for this book at https://github.com/editorscut/ec008-swift-kickstart-2.



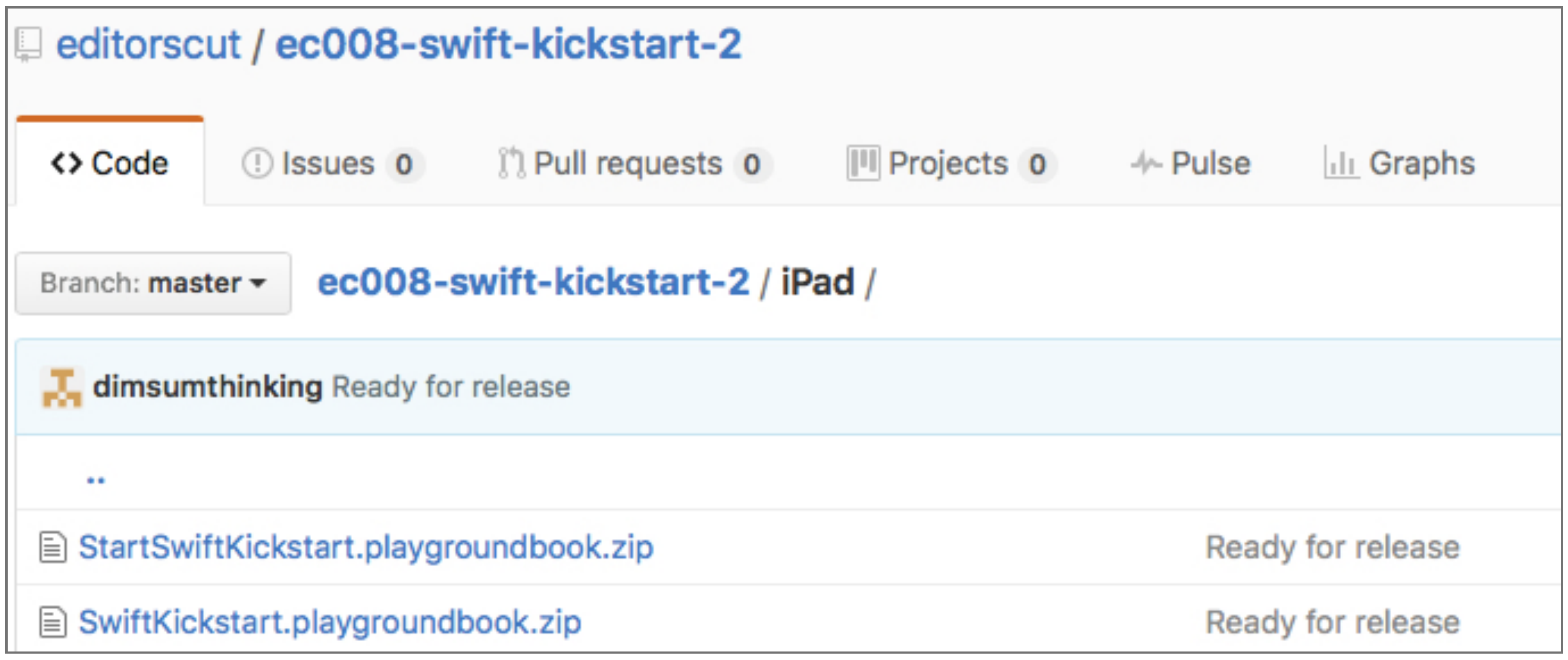
This is the repository for the code that accompanies the second edition of "A Swift Kickstart".

16 commits
1 branch
0 releases

dimsumthinking Ready for release

iPad	Ready for release
mac	Ready for release
source	Ready for release
.gitignore	added git ignore
README.md	Updated read

Select the *iPad* folder to see the files that you'll need if you're going to code on an iPad.


editorscut / ec008-swift-kickstart-2

ec008-swift-kickstart-2 / iPad /

dimsumthinking Ready for release

..

StartSwiftKickstart.playgroundbook.zip	Ready for release
SwiftKickstart.playgroundbook.zip	Ready for release

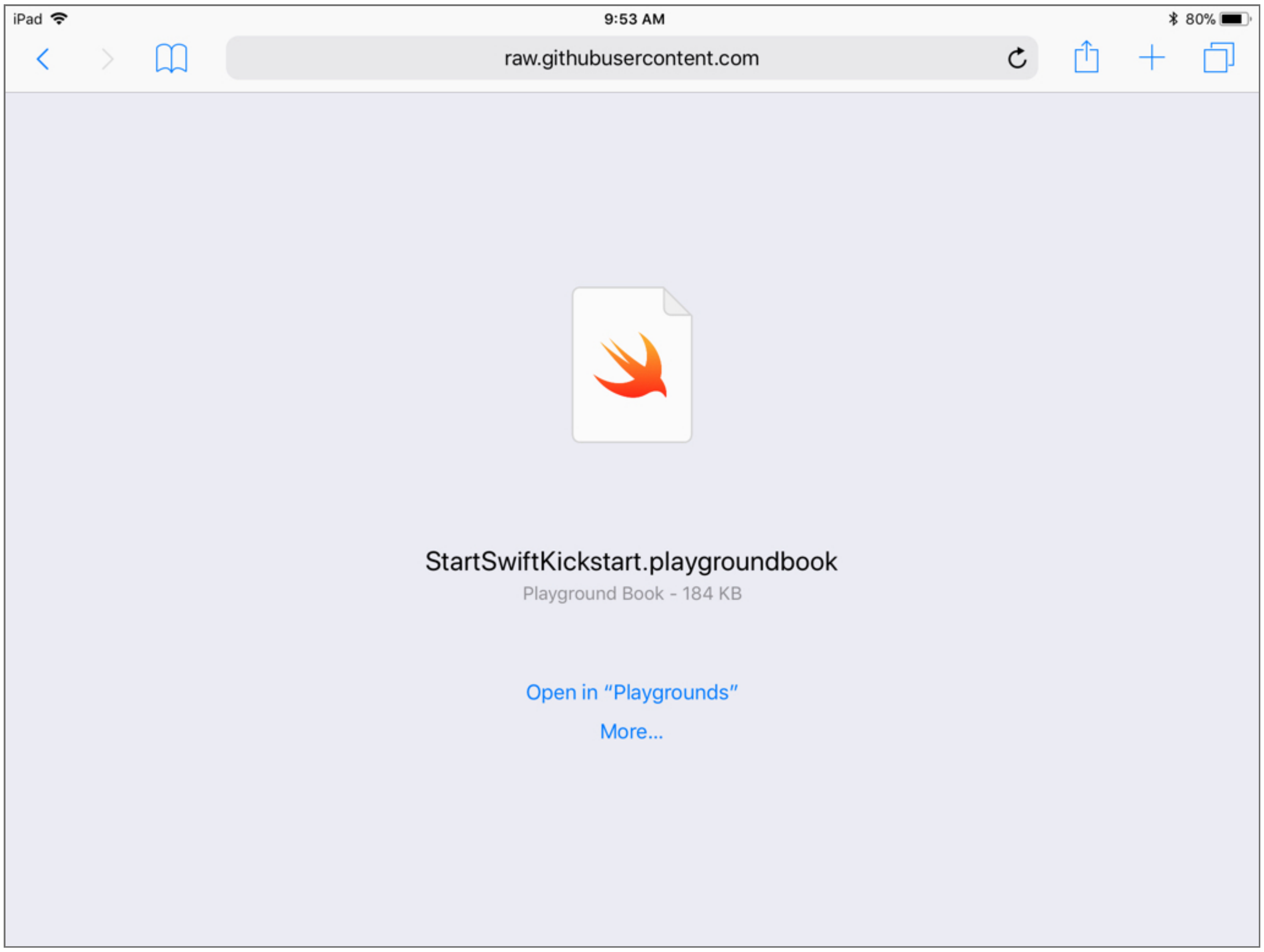
This directory contains two zip files, and each file unzips into a Swift playground. This playground is a *playgroundbook* file that has a chapter for each chapter of this book. It has a page for each section of this book. The difference between the two playgrouhds are as follows:

- *StartSwiftKickstart.playgroundbook.zip* contains pages that are blank other than some occasional bits of starter code to allow you to easily continue from previous sections. Start with this playground to follow along with this book.
- *SwiftKickstart.playgroundbook.zip* contains pages that are filled with the code that you'll enter as you work through each section of this book.

In other words, *StartSwiftKickstart.playgroundbook* is the "before" and *SwiftKickstart.playgroundbook* is the "after".

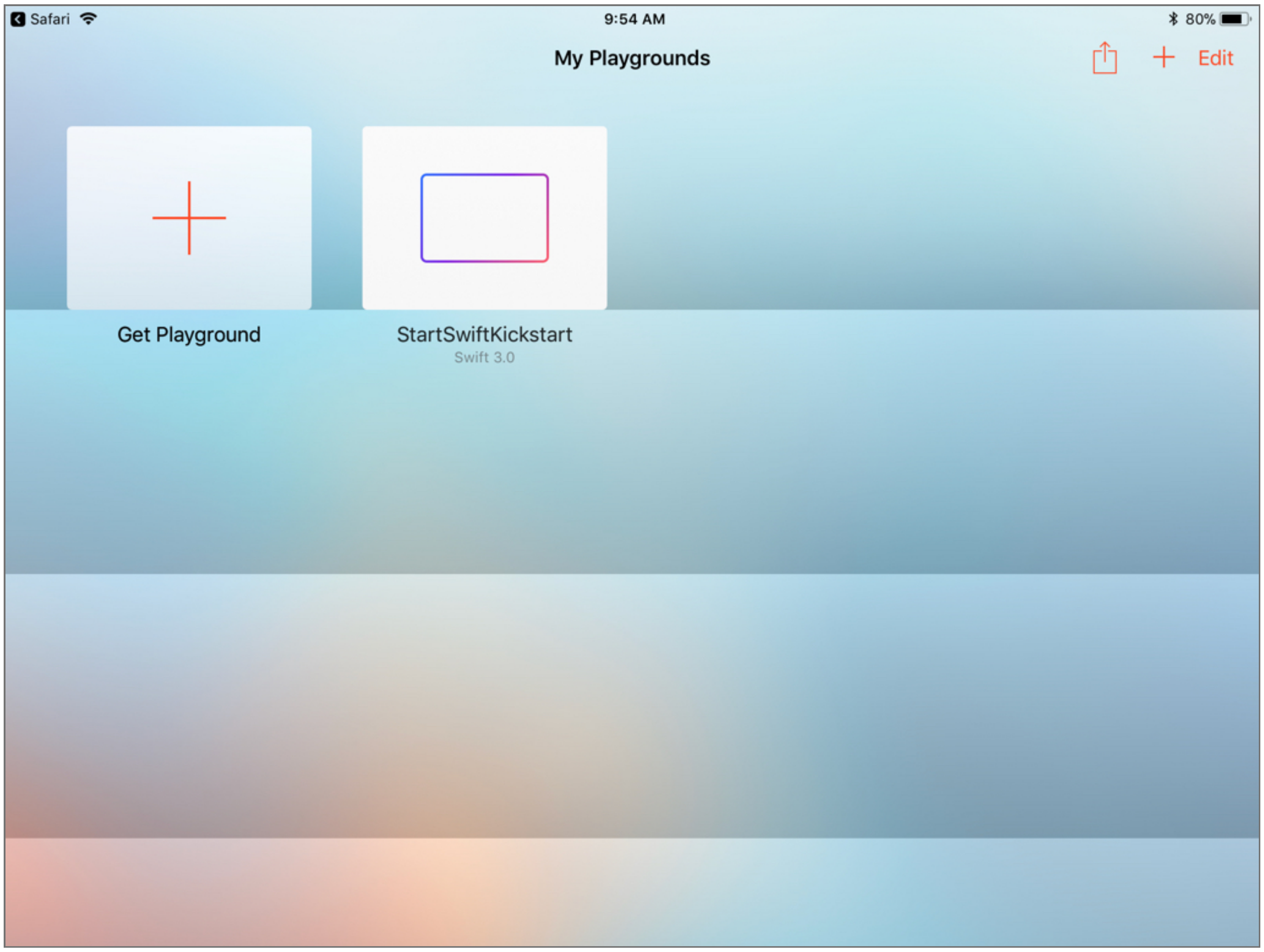
While you're on your iPad tap *StartSwiftKickstart.playgroundbook.zip*, and you'll be taken to this page then once you are taken to a new page tap the Download button in cluster of buttons in the lower right to download the zip file to your iPad.

You should see this prompt.

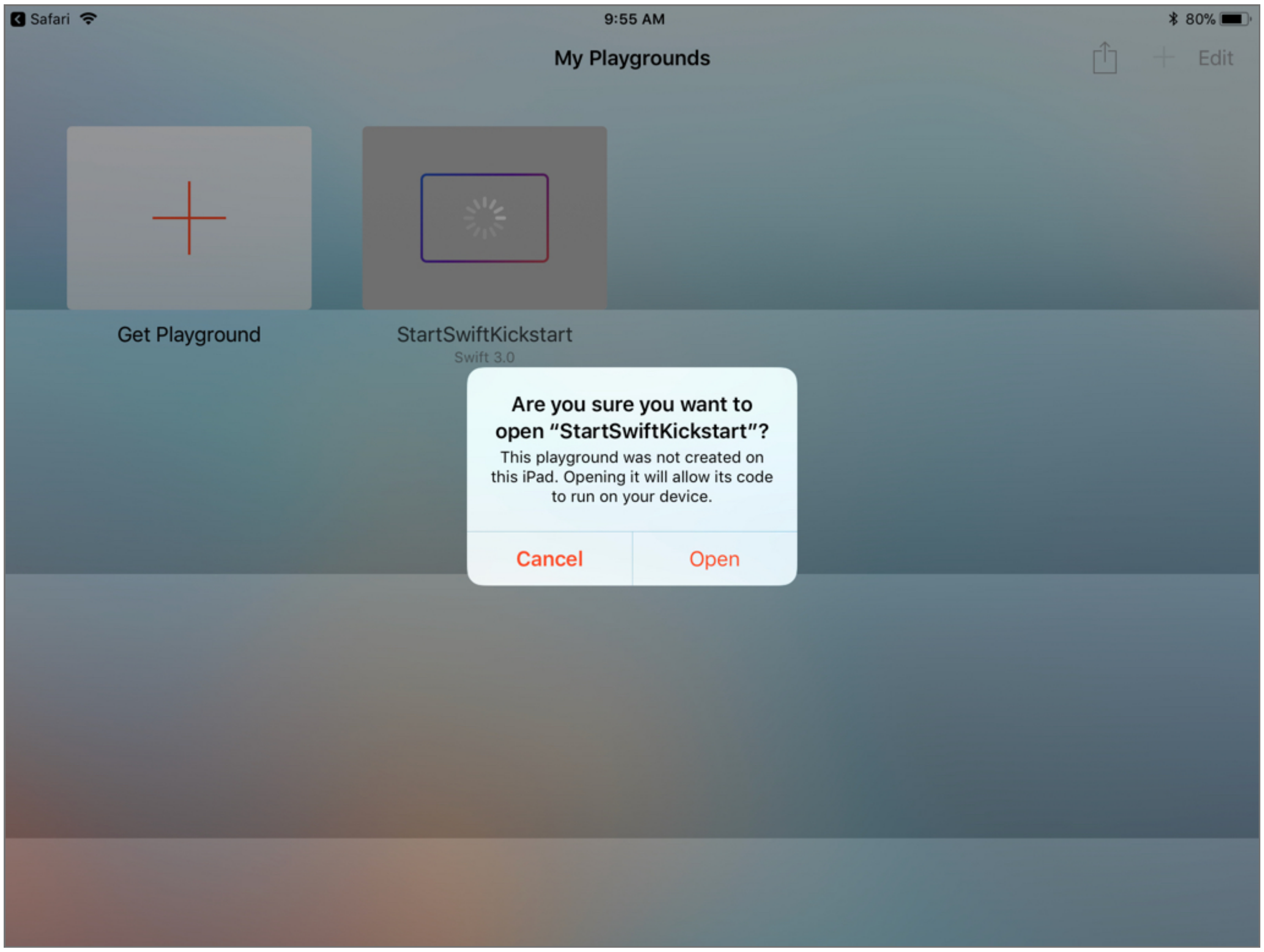


Choose Open in "Playgrounds" to open the Swift Playgrounds app and install this playground.

Here's the ***StartSwiftKickstart*** playground installed in the ***My Playgrounds*** tab.

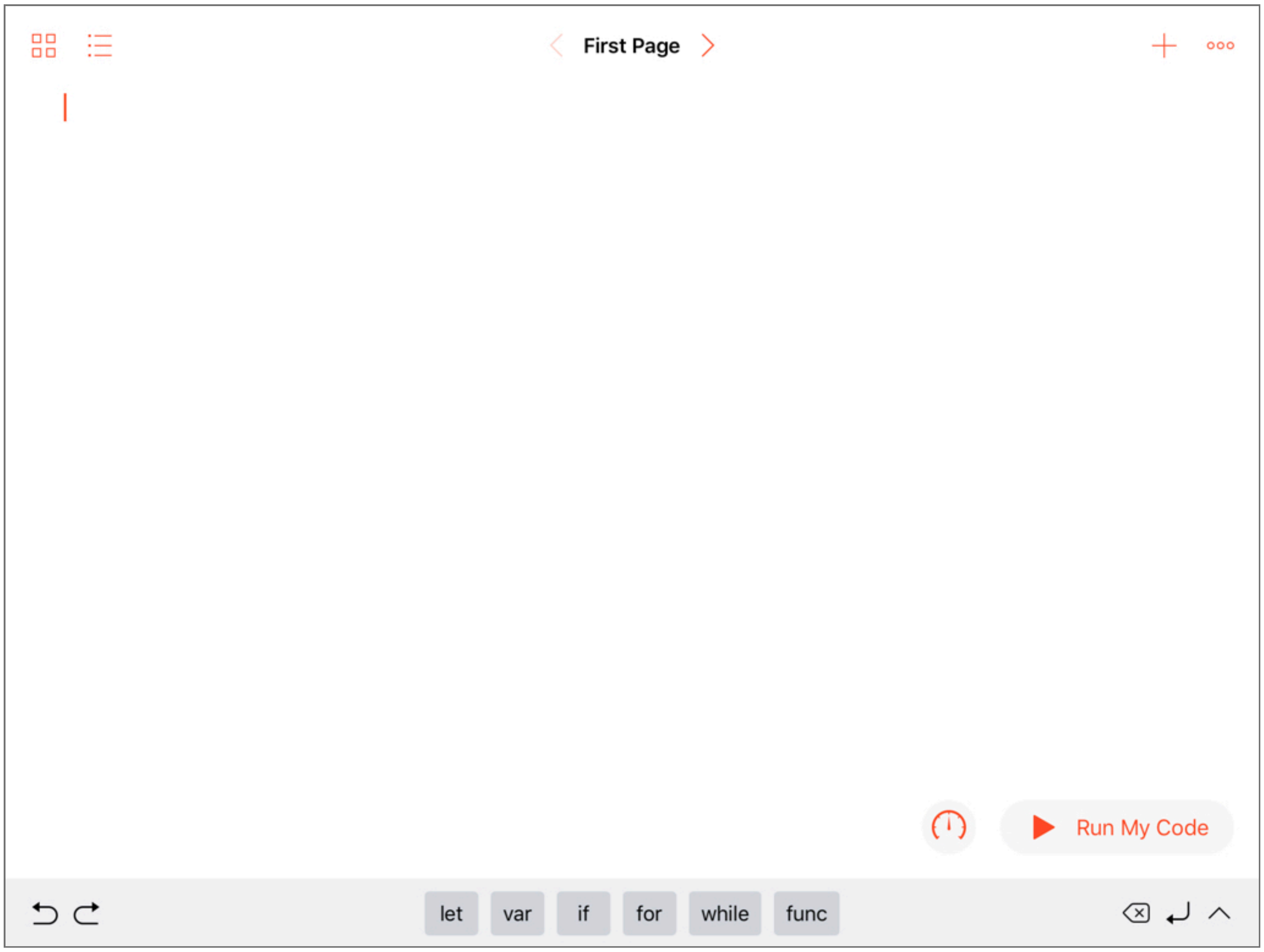


Tap the *StartSwiftKickstart* playground and you'll be asked to confirm that you really want to open it.

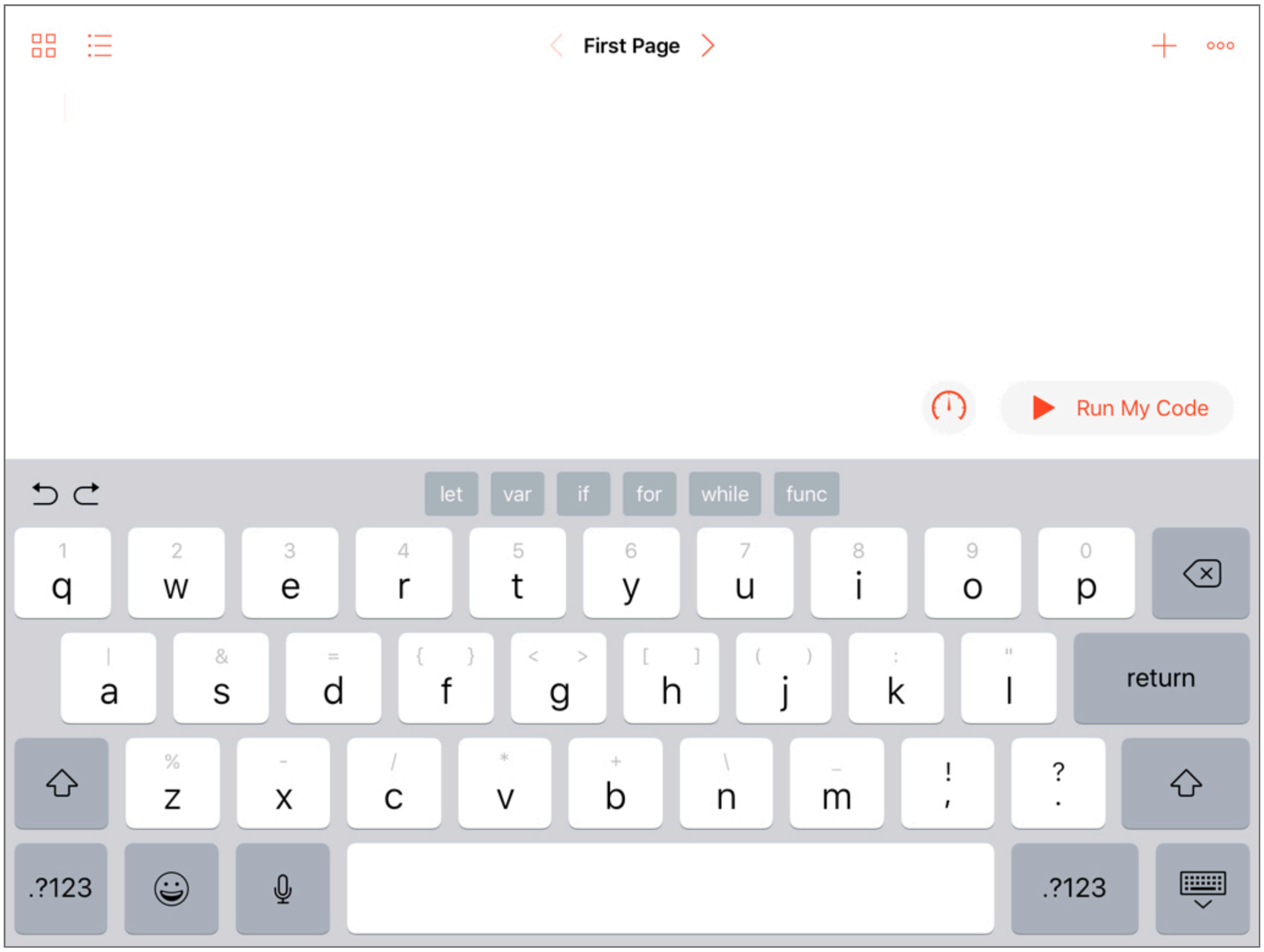


Are you sure you want to open "StartSwiftKickstart"?
This playground was not created on this iPad. Opening it will allow its code to run on your device.

Tap Open. The first page of the first chapter has the title ***First Page***. Other than this title, the page should be empty. Tap in the body of the first page and you'll see the cursor as a vertical orange bar, like this.

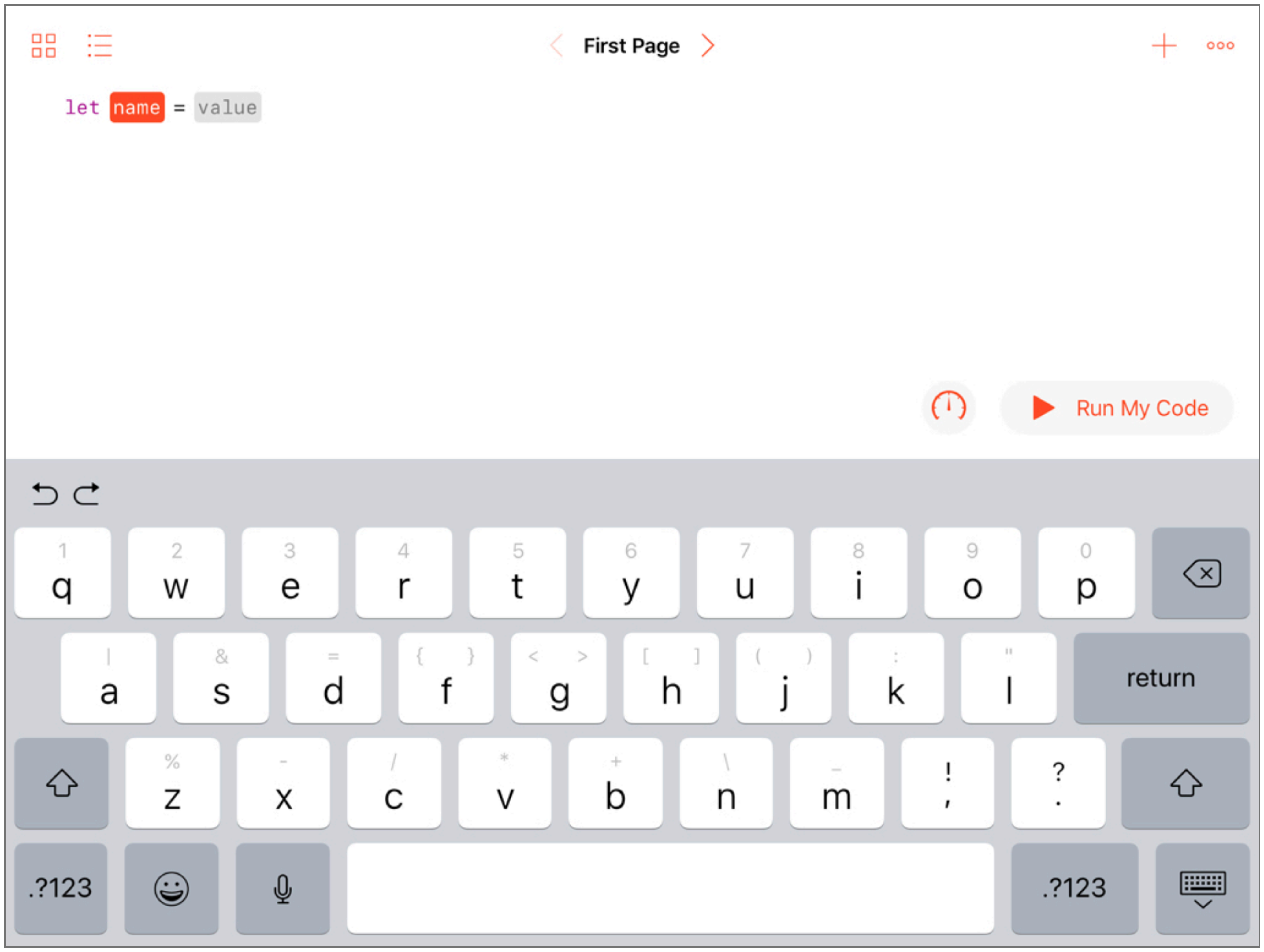


If your iPad has a physical keyboard, you can use it to type the code. Otherwise, Swift Playgrounds has a special keyboard that you can make visible by tapping the ^ in the lower-right corner.

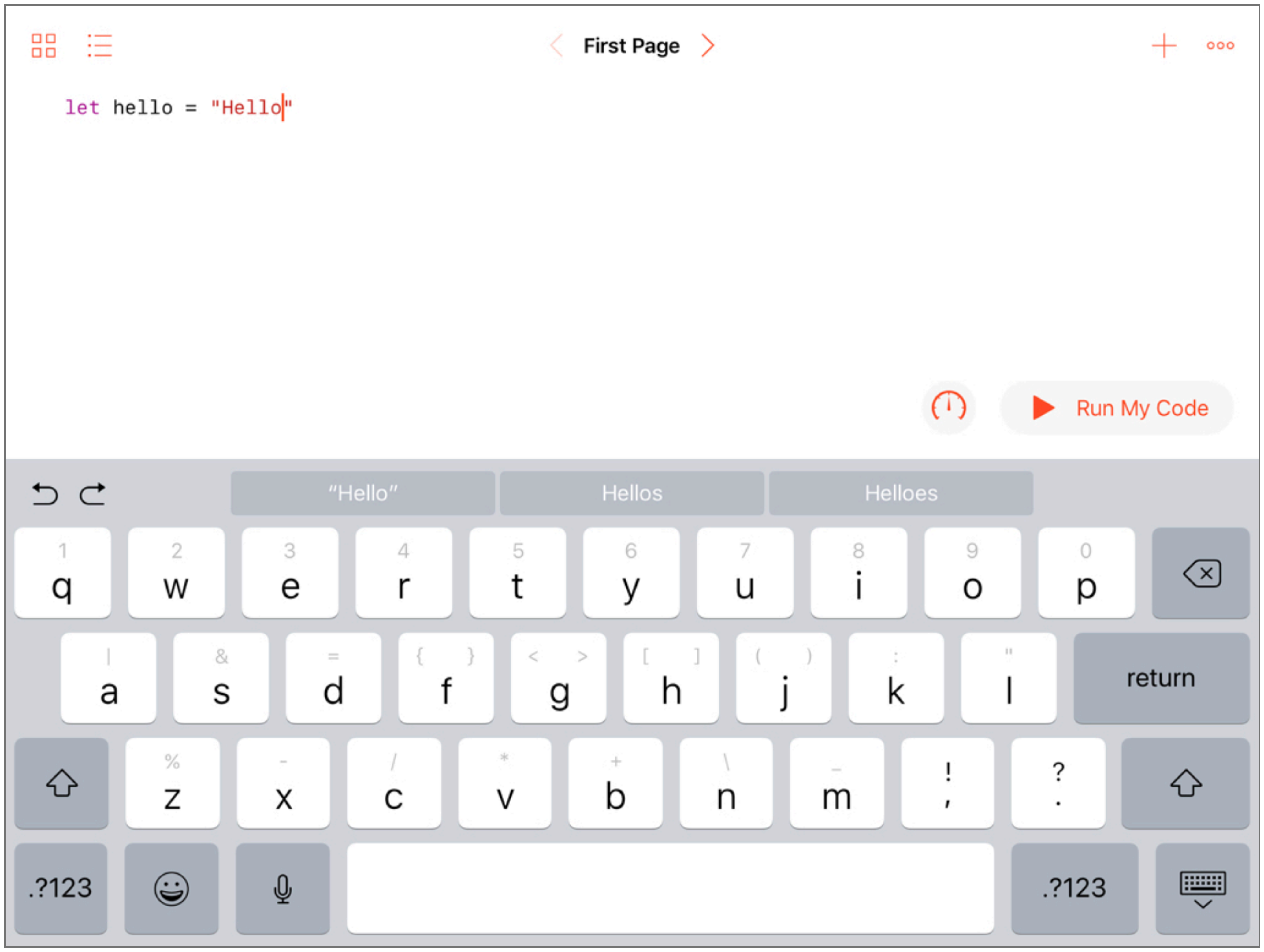


We're going to type `let hello = "Hello"`. You're declaring a `String` named `hello` and assigning it the value `"Hello"`. We'll look at this syntax in depth later in this book.

To start let's enter the word `let`. You can type `let` using the keyboard or you can choose not to. You may notice that it's one of the words that appears above the keyboard. You can just tap `let`. So tap or type `let` and you'll see a code completion prompt like this.

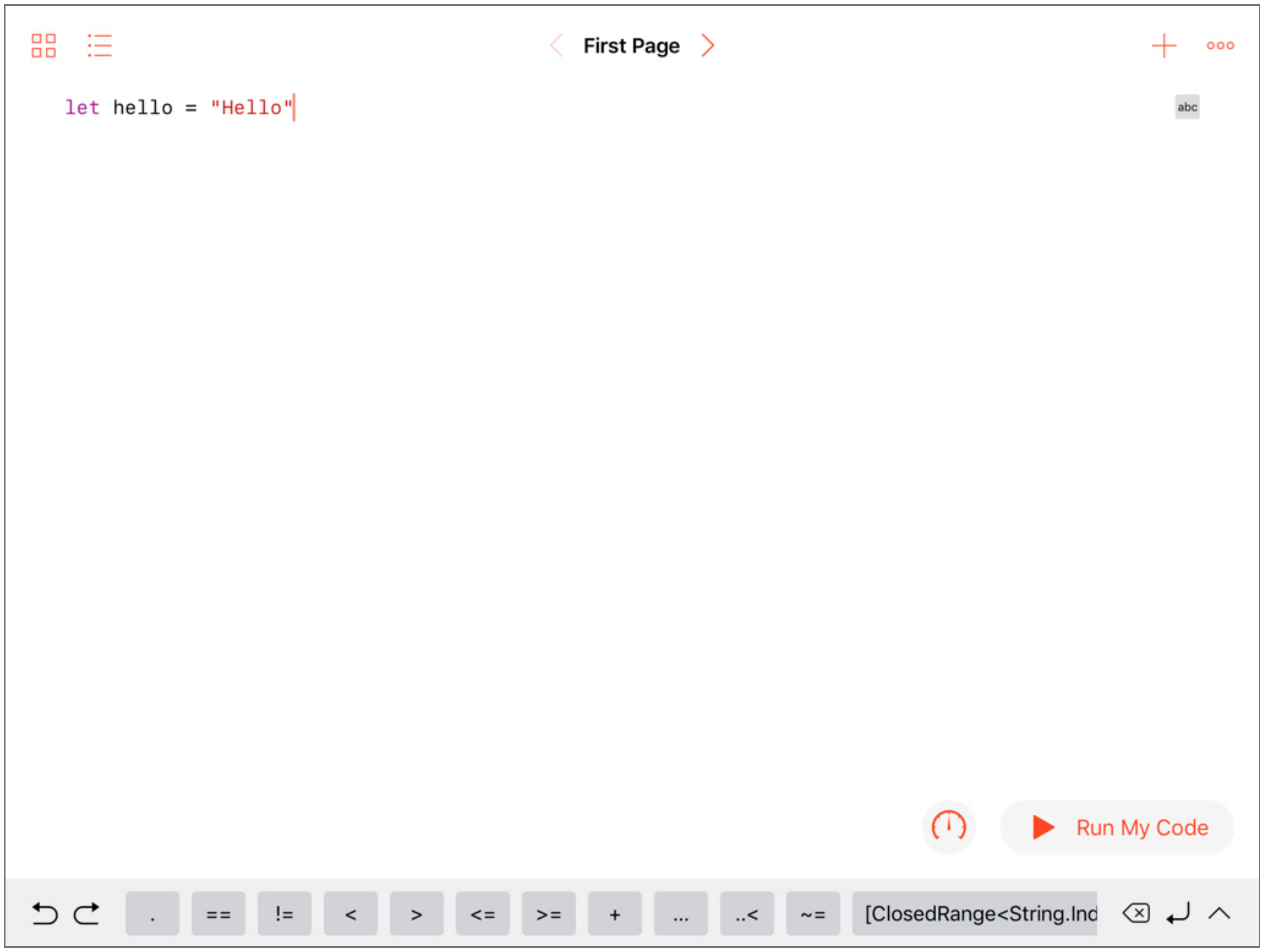

```
let name = value
```

Let's enter the rest of the command `let hello = "Hello"`. You may have a hard time locating `"` on this special keyboard. It's on the same key as the `l`. If you tap the Shift key, you'll get an uppercase `L`. To get the `"`, put your finger on the `l` key and drag it downwards.



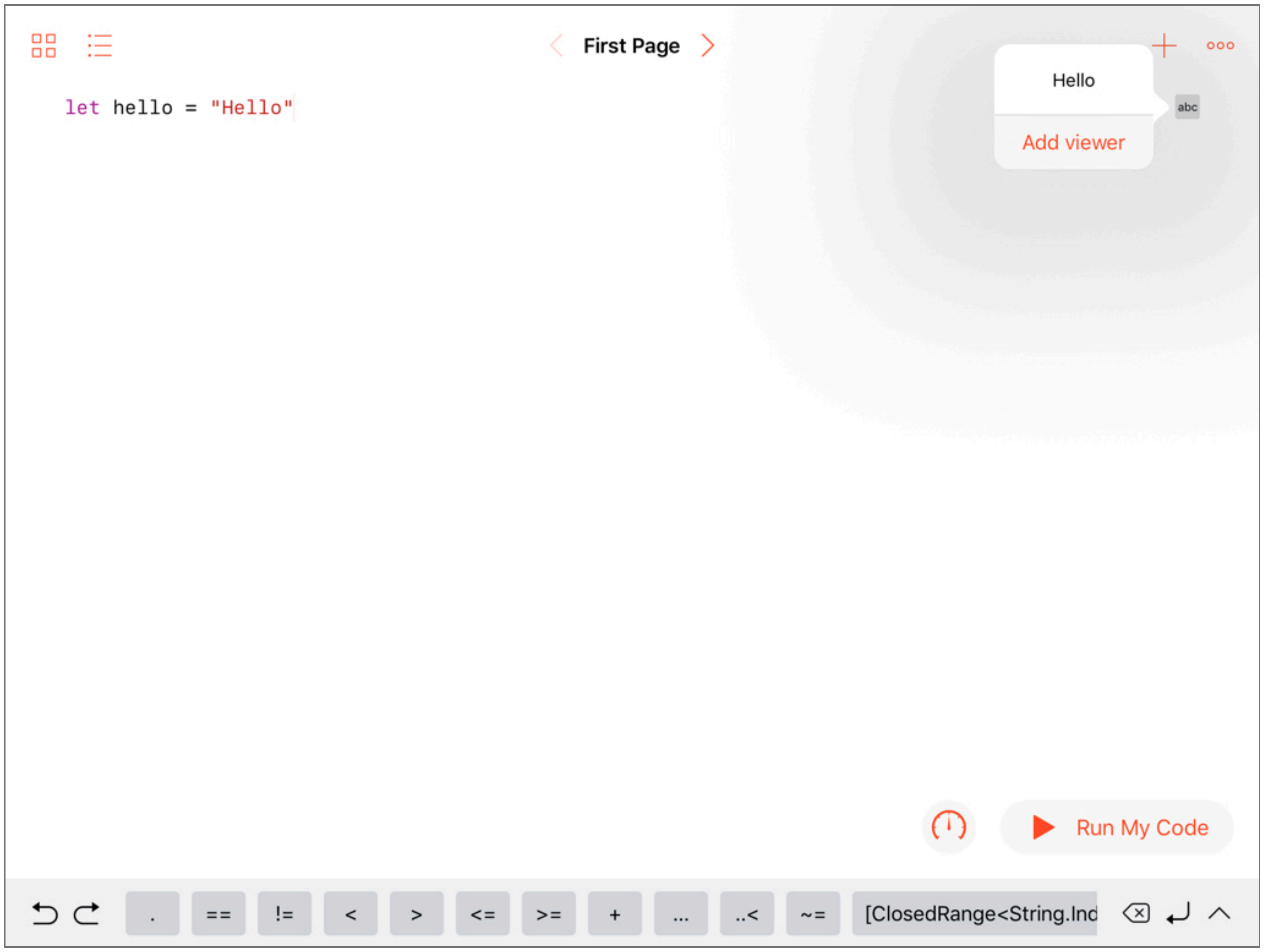
```
let hello = "Hello"
```

We need to get some feedback in our playground. To do so tap Run My Code and your screen should include a box with abc in it as shown below.

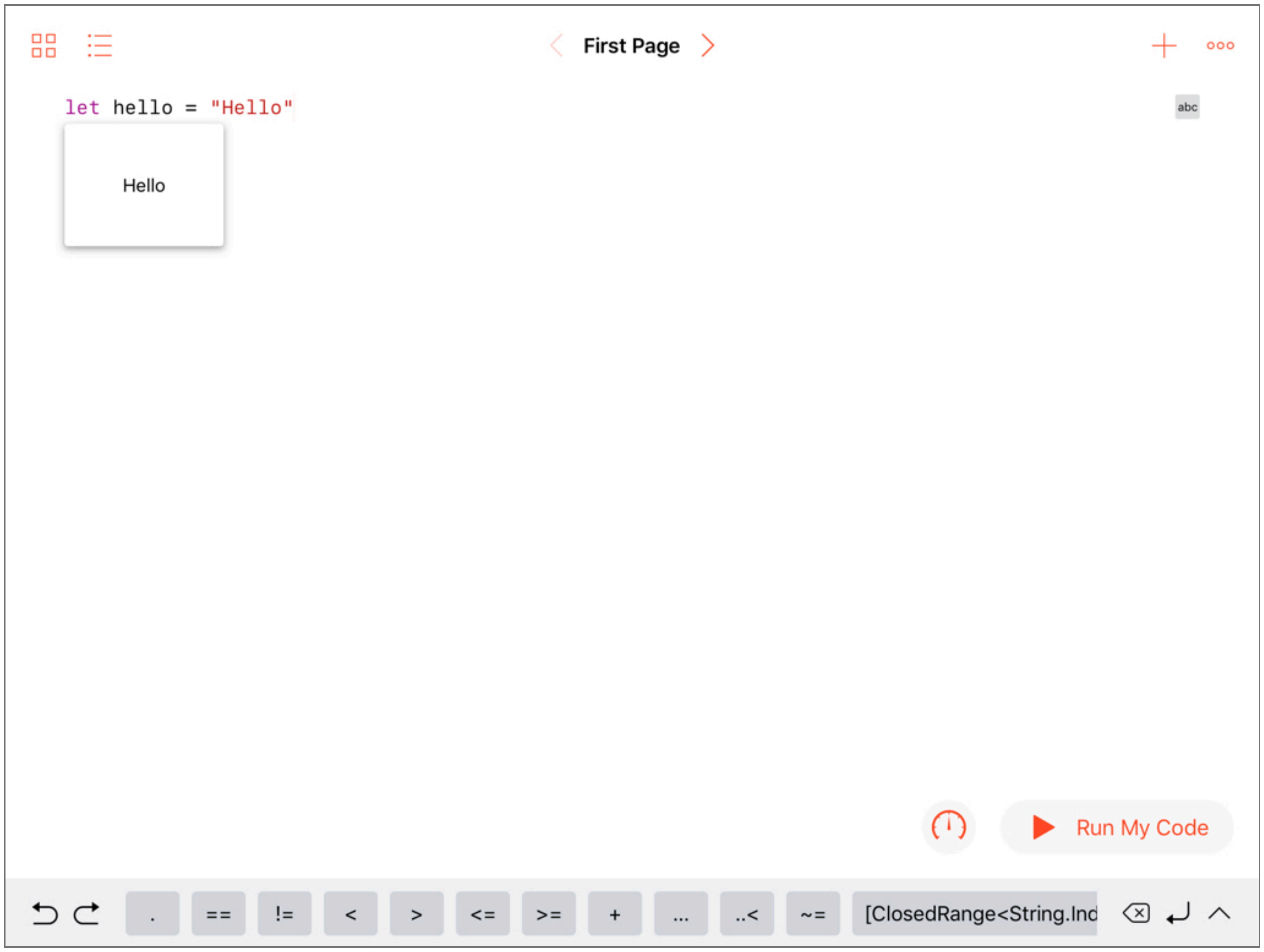


```
let hello = "Hello"
```

Tap the abc box to bring up this popover.

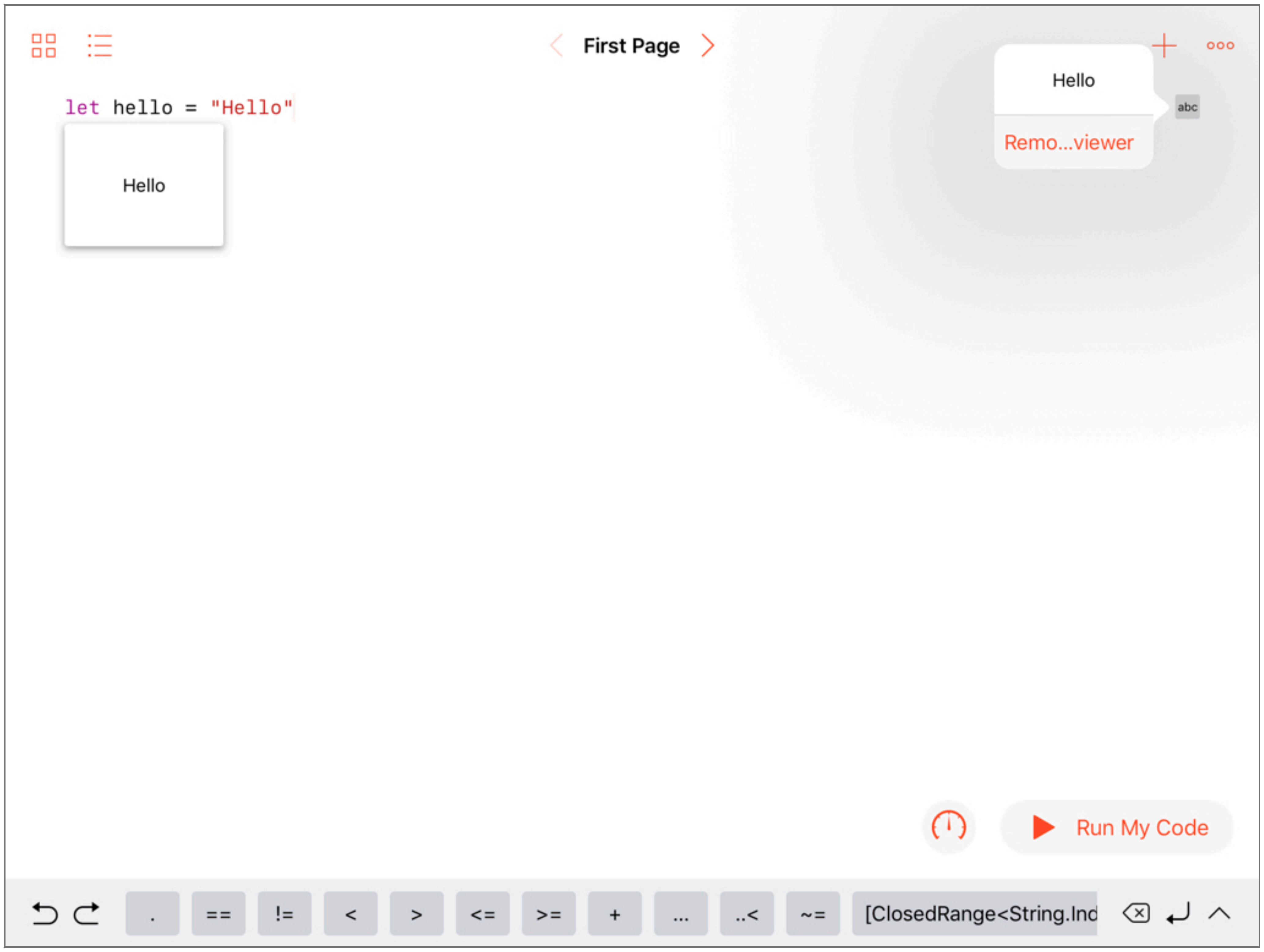

```
let hello = "Hello"
```

The `Hello` in the popover is the value of the `hello` variable. Below it you see Add viewer. Tap Add viewer to open this viewer below the line of code. The viewer contains the `String` with value `Hello`.



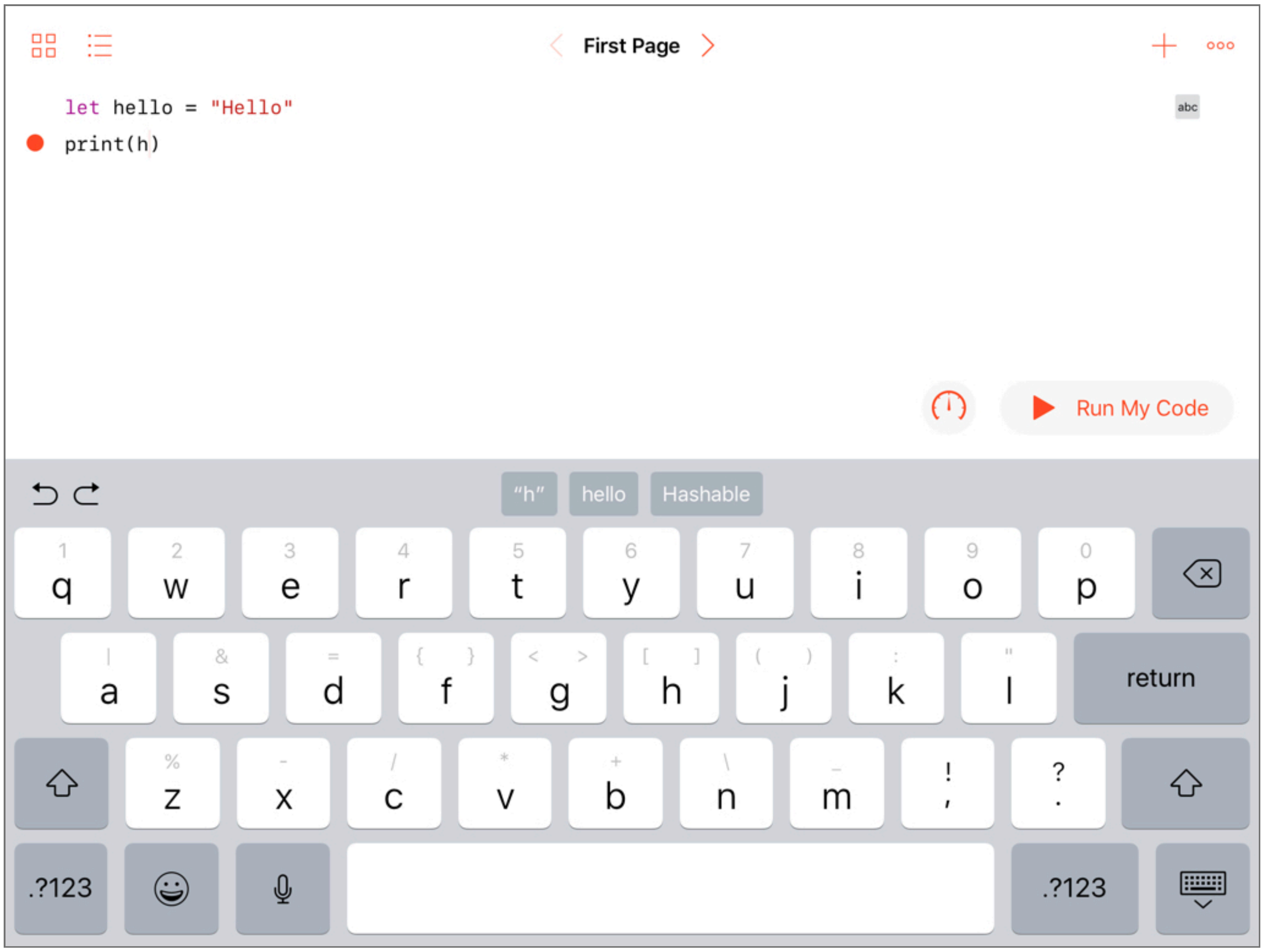
```
let hello = "Hello"
```

Tap abc again and you can now choose to Remove viewer.



```
let hello = "Hello"
```

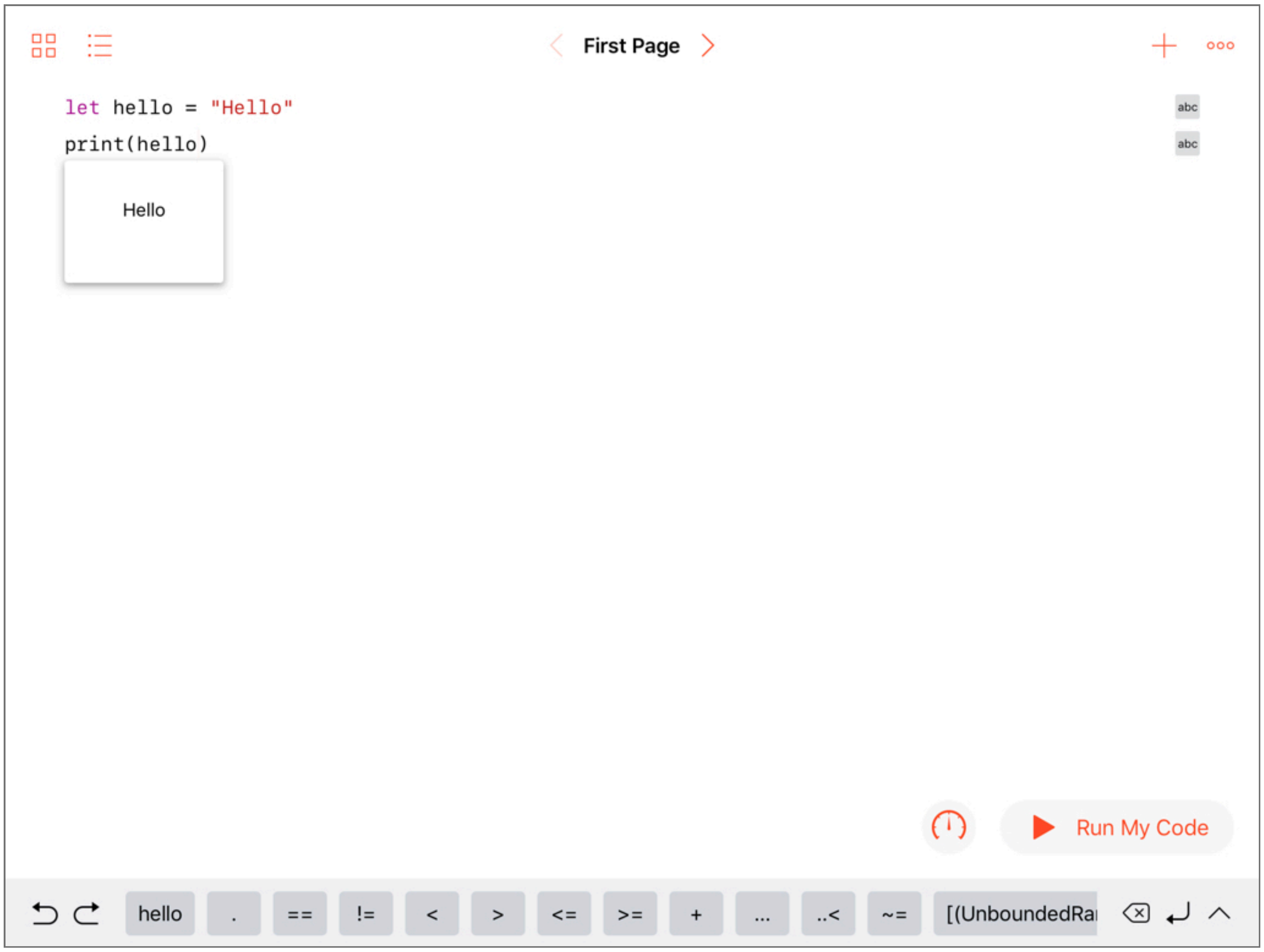
Begin to type `print(hello)` in the line below `let hello = "Hello"`. Stop when you get to the `h` in `hello`.


```
let hello = "Hello"
print(h)
```

You can see `hello` above the keyboard. Again, you can tap `hello` or you can type out all of the letters.

Once you have `print(hello)`, tap Run My Code.

On the Mac, output from `print()` appears in the console. There is no console in Swift Playgrounds on the iPad. We can see a little of what's going on by adding a viewer as before. Tap the abc box to the right of this `print()` line. You should see what the call to `print(hello)` would print to the console in this viewer.



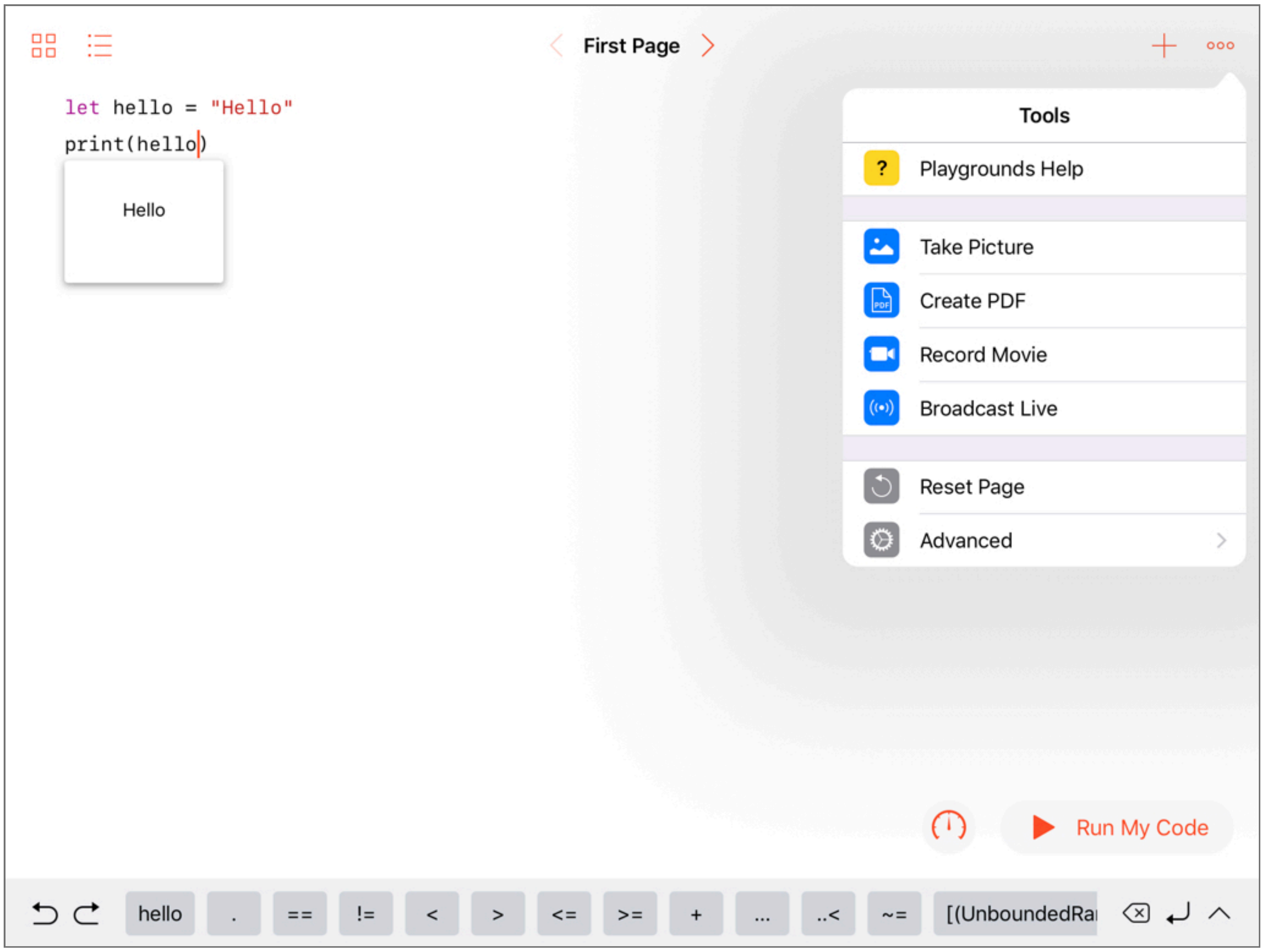
```
let hello = "Hello"
print(hello)
```

```
Hello
```

That's it!

You can check your work by comparing these results to the completed playground in *ThinkingInSwift.playgroundbook*.

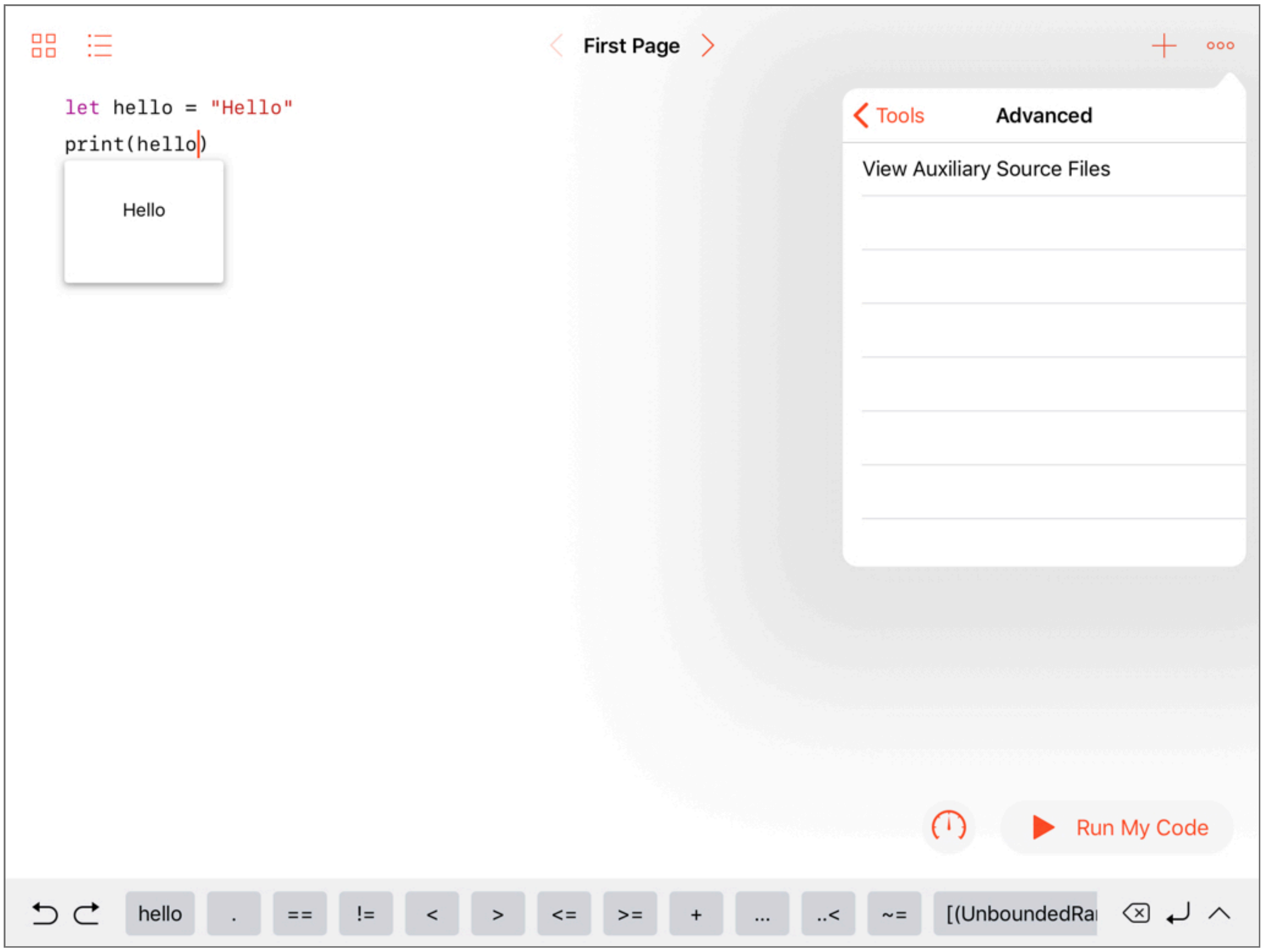
Before we move on, I want to show you how to look behind the scenes. Sometimes the files that you see on a playground page are assisted by code or images that are kept in `Sources` and `Resources` directories for the whole playground, for individual chapters, or for a single page. To find these, tap the three dots in the top-right corner.



```
let hello = "Hello"
print(hello)
```

```
Hello
```

Choose Advanced, and then choose View Auxiliary Source Files.

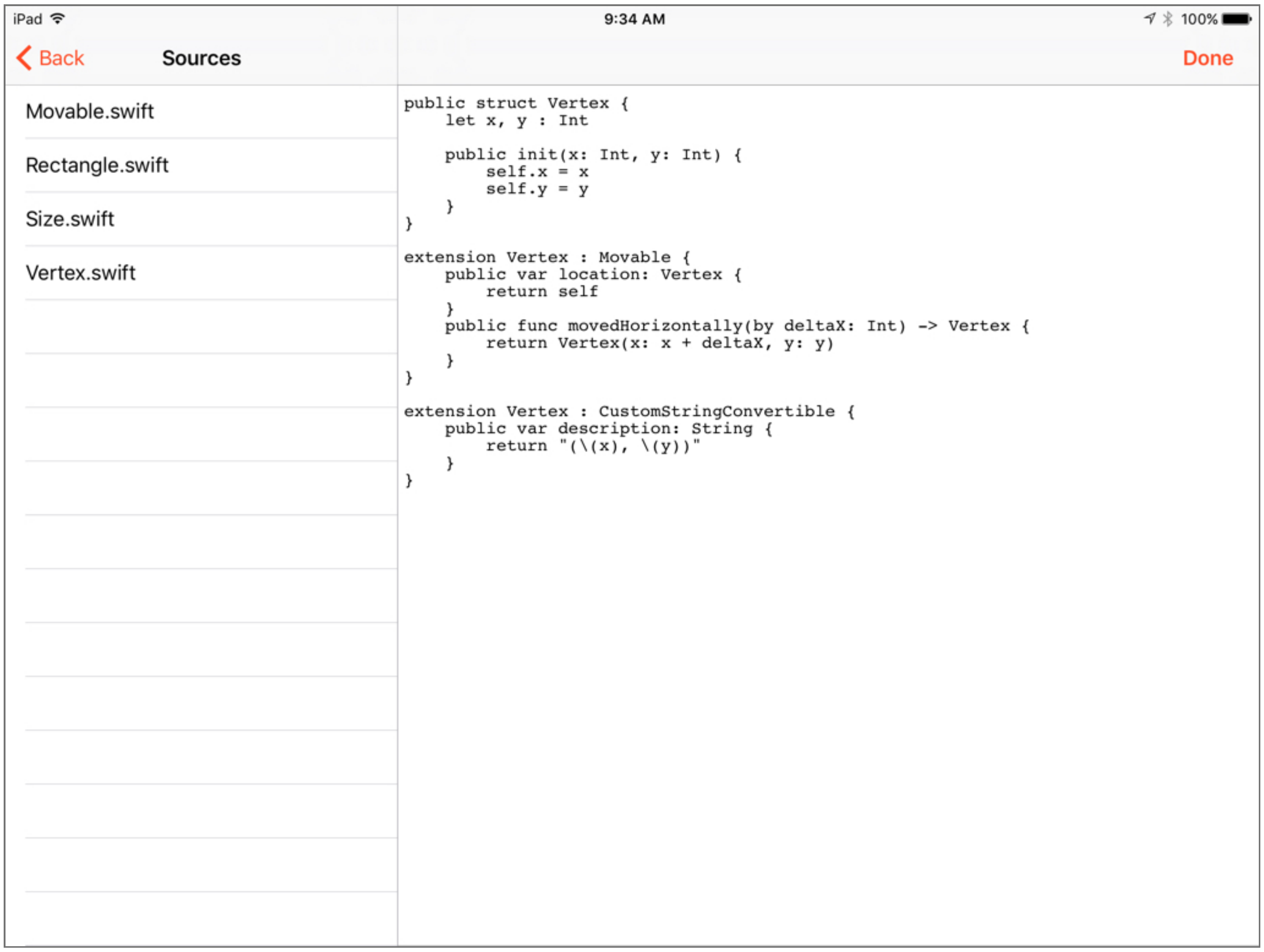


```
let hello = "Hello"
print(hello)
```

```
Hello
```

First Page
Tools
Advanced
View Auxiliary Source Files
Run My Code
hello . == != < > <= >= + ...< ~= [(UnboundedRa

You'll see the navigator on the left side. Follow the path from *Contents* to *Chapters* to *Chapter8.playgroundchapter* to *Pages* to *Playground9.playgroundpage* to *Sources*. The `Sources` directory contains source files that are helper files for the playground page. Tap `Sources` to see four *.swift* files and tap *Vertex.swift* to view its contents.



```
public struct Vertex {
    let x, y : Int

    public init(x: Int, y: Int) {
        self.x = x
        self.y = y
    }
}

extension Vertex : Movable {
    public var location: Vertex {
        return self
    }
    public func movedHorizontally(by deltaX: Int) -> Vertex {
        return Vertex(x: x + deltaX, y: y)
    }
}

extension Vertex : CustomStringConvertible {
    public var description: String {
        return "(\(x), \(y))"
    }
}
```

You can't modify files that you see in this view but it sometimes helps to examine the code. You'll need to do that, for example, at the end of Chapter 8.

In the next section, we adopt some code conventions for this book so that we don't have to show screenshots for Mac and for iPad for each step of the way.

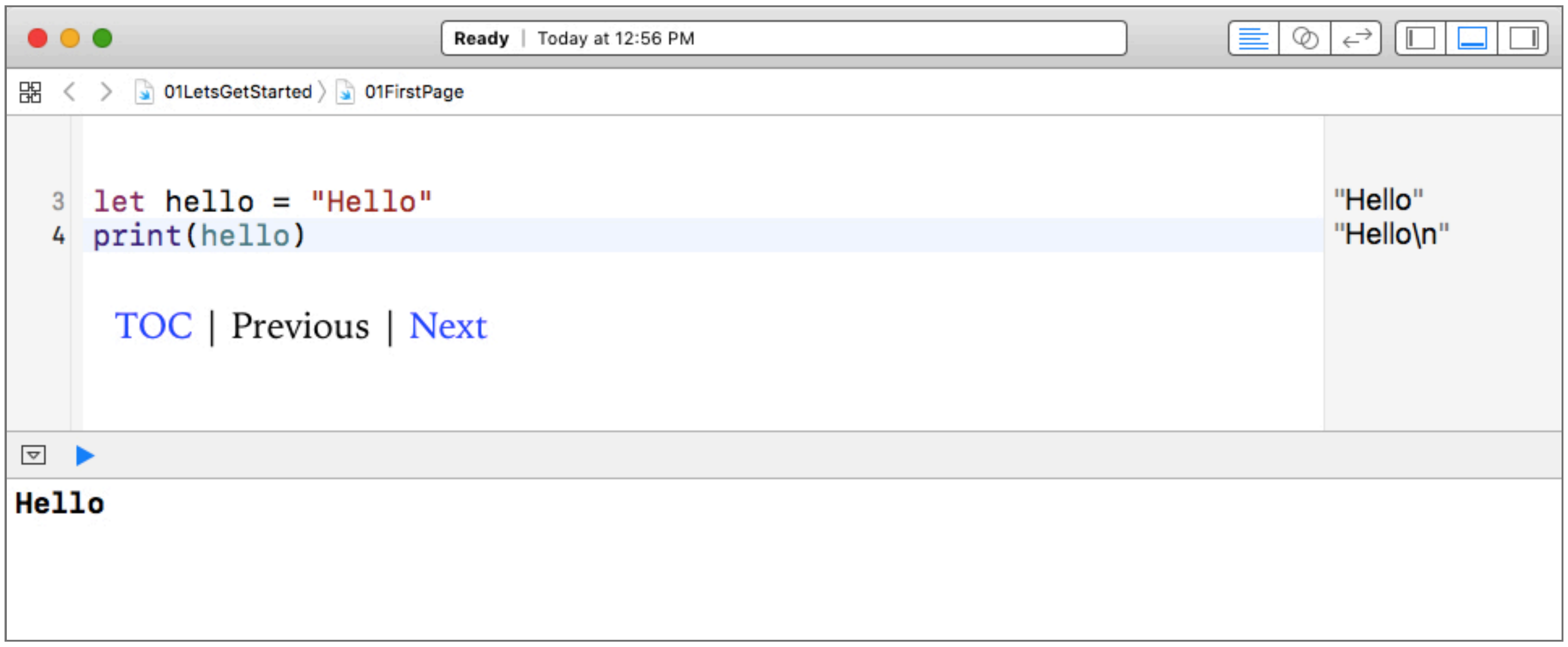
Code Conventions

As you've seen in the previous two sections, there are some differences between Xcode playgrounds on the Mac and Swift playgrounds on the iPad. The code area is essentially the same, but the feedback is a bit different.

In either case, this immediate feedback is what makes playgrounds so special. This feedback lets you easily experiment with and quickly modify your code. If you have a question about Swift or how to code something, pause before you ask someone or head to the Internet to ask, “What would happen if ...”, and try whatever it is you're wondering about. The playground is a great learning tool.

Let's take a moment to review the differences between playgrounds on the Mac and iPad.

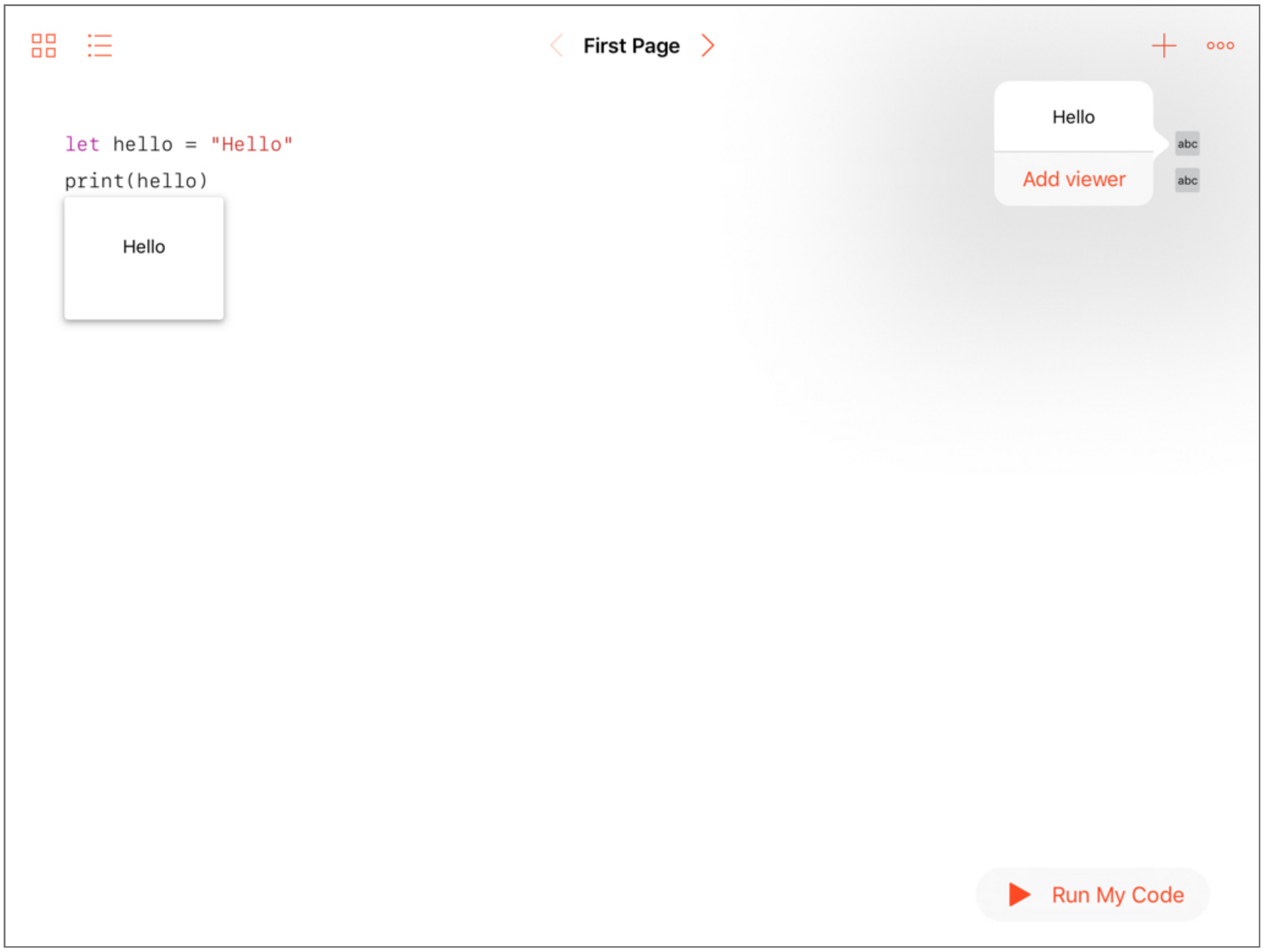
Here's a screenshot of our first page on the Mac. The column on the right reflects the value of various lines. The output of the calls to the `print()` function appears in the console below.



```
let hello = "Hello"        "Hello"
print(hello)               "Hello\n"
```

```
Hello
```

Here's the same page on the iPad. If you want to see the value for the line of code you need to tap the box on the right or choose to show the viewer. There's no overall console on the iPad. The calls to the `print()` function appear in the viewer for that particular call.


```
let hello = "Hello"
print(hello)
```

To simplify things in this book, instead of using syntax highlighting, I'll use a much simpler color scheme to call out the new code that we add to the playground in each step.

For example, in each case we first entered the line `let hello = "Hello"`. We then added the line `print(hello)`.

We'll present this sequence in a slightly stylized way.

We'll use the syntax coloring demonstrated in this code listing throughout this book.

```
let hello = "Hello" // This is a comment
                                                                "Hello"
print(hello)
                                                              "Hello\n"
```

- In the code listing above, `print(hello)` is presented in the color that I'll use to represent `new code that is being added to the playground` if it needs to be called out as such.

- The code for `let hello = "Hello"` is presented in the color that I'll use to represent `existing code in the playground`.
- The text in the above listing that reads `// This is a comment` is a comment. I'll use `this color for comments in the code listings`.
- I won't always show the playground response, but when I do, I'll place the response in the line following the line that it decorates. This may seem a bit odd, but I'm trying to help people who may need to increase the font size of this book and might end up with confusing line breaks.

  To further set the response apart, I'll align it right and render `playground response in this color`, as I have with `"Hello"` in the code listing above.

  You may notice additional differences. For example, on the Mac, the feedback includes quote marks and newline characters that don't appear in the Xcode version. The feedback on the right side of the iPad Swift Playgrounds is essentially the same as the Quick Look feedback in the Mac Xcode playgrounds. I'll often leave off the quotes and newline characters, and instead render them as they appear on the iPad.
- The `console and viewer output will also be rendered in this color` but will appear apart from code listings like this.

```
Hello
```

Well that's a lot of the details and mechanics out of the way.

The rest of this chapter has sections with links to resources, credits to those who helped with this book, a list of the versions and what was added or changed with each release, and a road map of the rest of the book.

If you'd like to get right to coding, proceed to the next chapter where we begin our journey with a look at functions in Swift.

Links

Download the source code for this book from GitHub and run it in Xcode 10 or higher. All of the source code for this book is distributed in either macOS playgrounds or iOS playground books. All code is written in Swift.

Submit your errata here for the book or for the source code by selecting New Issue. Please provide the book version (currently Version 0.7s), chapter, section, and page number in your issue so that I can find it and, if possible, resolve it quickly.

Apple has posted videos, slides, and sample code from the Worldwide Developers Conference. Swift was introduced in 2014 and there were many important talks about the new language. Although much of the Swift syntax has changed since then, the big ideas are the same. You may want to watch the Introduction to Swift and Intermediate Swift talks from 2014. Swift 2.0 and Xcode 7 were announced at the 2015 conference. Swift 3.0 and Xcode 8 were announced at the 2016 conference. The talks in the Developer Tools category are invaluable. In particular, you should watch What's New in Swift (each year), Protocol Oriented Programming in Swift, and Building Better Apps with Value Types in Swift. You can access these in your browser or by downloading the WWDC app.

Apple has also made two books available for free as ebooks. You should download The Swift Programming Language for sure and Using Swift with Cocoa and Objective-C if you have existing iOS or Mac apps. In addition, you can keep up with what's happening in Swift on the open source homepage swift.org.

When deciding how your Swift should look, I recommend Erica Sadun's wonderful book Swift Style and Ray Wenderlich's Swift Style Guide.

Subscribe to the Editors Cut newsletter for occasional email about this book as well as announcements of other books in the Editors Cut series.

Visit the Editors Cut site for more information about our books, training, and conference appearances.

Credits

Tools

I used Coda, BBEdit, Safari, Anvil, and Terminal to produce this book.

All of the examples in the book have been checked against the latest public version of Xcode on the latest public version of OS X. As of the time of this writing, that's Xcode 8.3 on Sierra.

The screenshots in this book were captured with Snapz Pro X.

Wonderful People

Thanks to the many reviewers who filed issues as this book was being written and updated. In particular, thank you to Tyler Morrison, Adrian Kosmaczewski, Srdjan Markovic, Michael Mayer, Helen McManus, and Simon Wolf. Special thanks to Regine Horteur-Edjlali, Aijaz Ansari, and Liz Marley, who submitted so many suggestions and corrections.

This book is vastly improved by the contributions of the editor, Jill Steinberg, and the designer, Alison Brown.

Alison was amazingly patient and helpful during the transition to ePub. It's only my limited abilities with CSS that keep this from looking as good as she wanted.

I can't imagine writing a book without Jill. Her comments help me say what it is I thought I was saying.

Thanks to the Xcode team for all of the improvements in playgrounds. That alone has made this book so much better.

Thanks most of all to my late wife, Kim, and my daughter, Maggie.

As I wrapped up the first edition of this book, Maggie was preparing to head for college. Such a huge moment in her life and in Kim's and mine.

I've told Kim that now we can hang out more and talk.

She nods and says, maybe I should go up to my office and start my next book.

As I return to write the second edition, Kim is no longer with us. She was killed in an automobile accident in the summer of 2016. It makes me happy to see the reference to her above - it reminds me that she was a part of every aspect of my life.

Thank you Kim for everything. I miss you.

The Author

Daniel H Steinberg writes short, single-topic books designed to be read on your iPad and Mac available from The Editor's Cut (http://editorscut.com).

He's been writing iPhone and iPad apps since the beginning and coding in Swift since the announcement.

Daniel presents iOS and OS X training at conferences and in public and private training sessions. See more at http://dimsumthinking.com. You can book him for private training or consulting through inquiries@dimsumthinking.com.

Read Daniel's blog posts and subscribe to the Editor's Cut.

Before moving to Swift, Daniel enjoyed the world of Mac and iOS development using Objective-C for many years. He played with BASIC as a kid and Pascal as an undergrad. He then learned to program in C to model the mathematics that he was investigating for his Ph.D. research on elastic curves. Daniel moved to Java in the early days, co-wrote the Java 2 Bible, and became a frequent contributor to JavaWorld Magazine. He was the founding editor of java.net for O'Reilly Media and Sun Microsystems. Daniel created the Mac and iOS line of books for The Pragmatic Programmers. Along the way, he has written dozens of books and hundreds of articles.

Version History

Second Edition

These are the release notes for the second edition from most recent to the initial release.

Version 0.7

This is the Swift 5 / Xcode 10 release.

I've added a mention of Raw Strings to the print() section

One of the custom operators was changed from `>>>` to `|>` as it is now widely accepted for piping forward. `>>>` is still used for function composition.

I've updated the flatMap material to use the new name `compactMap` for the one that was renamed.

I wanted to deprecate and remove the error section but didn't after customer feedback.

I didn't add Swift's `Result` type to the Error chapter. I will be covering it in another book.

All of the code has been tested in Xcode 10 and on Swift playgrounds.

Version 0.6

This is the Swift 4.1 / Xcode 9.3 release.

All of the code has been updated to support the new features in Swift 4.1. In particular, the new synthesized Equatable is used

The section on Strings in the Collections section has been expanded.

The chapter on Sequences has been deprecated and is available in the code download.

All of the code has been tested in Xcode 9.3 and on Swift playgrounds 2.

Version 0.5

This is the Swift 4 / Xcode 9 initial release.

All of the code has been updated to support the new features in Swift 4.

There is an additional section on Strings in the Collections section.

I debated removing chapters 11 and 12 on Sequences and Errors. If I were writing the book today I don't think I would have added this material as it isn't central to what you need to know about Swift. I left it in as there are a lot of nice nuggets in those two chapters.

All of the code has been tested in Xcode 9 and on Swift playgrounds 2.

Also there are a dozen or more errata thanks to helpful submissions form readers.

Version 0.4

There are some small updates for Swift 3.1 including improvements in the Generic Sequence section.

There are two additional sections.

The first expands on access levels to include the `open` keyword and the Swift technique for having a property's getter be more visible than its setter.

The second covers capture lists in closures so you can control the memory type for the values captured in a closure.

Also there are a dozen or more errata thanks to helpful submissions form readers.

Version 0.3

I originally planned for this to be a small update where I responded to errata submitted by readers.

The good news is that of course, I've done that. Thank you all for submitting issues.

The better news is that I decided to add four new chapters.

I teach a two day introduction to Swift. Day one is supported by parts one and two of this book. I've added a part three and filled it with material from day two of the training. This new part includes chapters called ***Flexible Functions***, ***Higher-Order Functions***, ***Sequences***, and ***Errors***.

Enjoy!

Version 0.2

This is the conversion of the book to Swift 3.0 and Xcode 8.

In addition the other big change is that this version and the source code support people who want to learn on the Mac or on the iPad. There's complete support for Xcode playgrounds on the Mac and Swift playgrounds on the iPad.

Oh my goodness. This required a lot more than I expected.

I completely rethought each chapter. I stopped referring to earlier versions of Swift or, for the most part, to Objective-C. As Swift finds its own voice, I decided not to talk about its awkward teen years when it was finding its way.

I eliminated and added sections. I may bring some of them back. I eliminated most of the screen shots other than those in Chapter 1 because I want to support both Xcode on the Mac and Swift Playgrounds on iOS. This also meant that I had to rethink some things that Swift Playgrounds doesn't support well. I eliminated a lot of `print()` statements because iOS doesn't have a console. I mostly moved code from ***Source*** folders directly into the playground page because you can't edit ***Source*** files on iOS.

Version 0.1

In early 2016 I decided to move this book to ePub. At that time, there was no mechanism for offering it as a free version to folks who currently owned the book, so I decided to start the version numbers for this second edition at 0.1.

This first release is mainly a port of Version 2.1 of the first edition, but it paved the way for a full reworking of the book for Swift 2.2 this spring and Swift 3.0 later this year.

First Edition

The initial release of the first edition appeared in July 2014 just one month after Swift was announced and was marked version 1.0. Here's the history of the changes in each subsequent version from most recent to the first release.

Version 2.1

Everything is updated to Swift 2.1. Screenshots and console feedback have been recaptured to reflect the latest version of Xcode. Thanks to your continued help and feedback, I've been able to incorporate yet another batch of errata kindly contributed by readers.

Version 2.0

Everything is updated to the final version of Swift 2.0. This is the version that corresponds to the first public release of Xcode 7. Thanks to your continued help and feedback, I've been able to incorporate dozens more errata.

Version 1.9

Everything is updated to Xcode 7 b6. This included changes to `print()`, a new `try?` keyword, and modifications to the code and playground feedback throughout the book. I added a section on the `print()` function itself as well as sections on error handling, custom operators, and failable and required initializers. Thanks to your feedback, I've been able to incorporate dozens of errata.

Version 1.8

Everything was updated to Swift 2.0 and Xcode 7 b5. I thought that this update would take a day or two, but so many great things have been added to the language and the tools that this is a major rewrite. It's essentially the second edition of this book.

All of the playgrounds were updated to support the new playground format with multiple pages and rich text. The code download now contains a single playground for each chapter.

Many of the examples have been streamlined and simplified based on my experience teaching this material at workshops over the past year. I changed some of the examples because the context didn't resonate with some people. Yes, the basketball example is gone.

The variables, collections, and flexible functions chapters were significantly reworked. The enumerations, structs, and classes chapters were essentially rewritten. The final chapter of the last version was filled with odds and ends. This material was integrated into other chapters. A chapter on protocols was added.

Version 1.7

This was mainly a bug fix release. I included fixes to dozens of errata submitted by readers. Thank you! The result was corrections to the code and the prose in the book and to a few slightly expanded sections.

Version 1.6

I updated the code for Xcode 6.1 and the Yosemite release. This included modifying the way to access and use raw values in enumerations. All of the screenshots were updated from Mavericks to Yosemite. There are also two new introductory examples of using subscripts. I added example of Swift's `MARK:`, `TODO:`, and `FIXME:`.

Version 1.5

I updated the code for Version 1.0 of the Swift Programming Language. I rewrote Chapter 2 and portions of 1 and 3 to move from the REPL to the playground as the REPL support changed in Xcode 6 GM. I added two sections to Chapter 9 Flexible Types on typealias. I also added screenshots as the NDA was dropped.

Version 1.4

I added Chapter 9: Flexible Types. In Chapter 4, I adjusted the in-out parameter example and added a section on functional programming. Added material on using protocols with structs and using them with generics to Chapter 7. In Chapter 8, the advice on structs vs classes has been updated.

Version 1.3

I added Chapter 7 Structs and Chapter 8 Classes

Version 1.2

I added Chapter 5: Flexible Functions. Chapter 6: Enumerations was revised and moved with two additional sections. A section on overloading functions was added to Chapter 2.

Version 1.1

I added Chapter 4: Collections, which covers the `Array` and `Dictionary` types. I updated the book to conform to the recent Swift release in Xcode beta 3. This meant changing the array types from `String[]` to `[String]` and changing the half-open range indicator from `1..4` to `1..<4`. The `Array` sections also reflect the new immutability rules for Swift arrays.

Version 1.0

This was the initial release of the book. The content chapters were Chapter 2: Functions and Chapter 3: Variables and Constants.

Road Map

Here's a quick overview of the book. You can think of the book as falling into roughly two parts.

Part One: Functions and Variables

This part describes the basic components of any Swift application: functions, variables, and collections. As Swift has matured, even these fundamental building blocks have changed quite a bit.

Chapter 2: Functions

We introduce the basic syntax of Swift functions. We use playgrounds to explore how we create and call functions that may take zero or more parameters and return zero or more values.

Chapter 3: Variables and Constants

Sometimes we need to store state. Constants are more than just a way to store magic numbers. In Swift, we think of constants as being immutable variables and use them to make it clear when we have a variable whose value will never change. You'll learn to create and use variables and constants.

Chapter 4: Collections

There are two fundamental collections in Swift: Array and Dictionary. These collections must be homogeneous. In other words, they must contain elements that all have the same type. We look at how to create arrays and dictionaries and how to modify them. You'll learn to enumerate arrays and dictionaries and see how to use optional types. At the end we take a brief look at a third type of collections: Set.

Part Two: Types

In Part One, we mainly used the types provided to us in the Swift Standard Library and in the Cocoa and Cocoa Touch libraries. In Part Two, we learn to create our own types. Even the simplest of Swift applications requires that we create our own types and teach them to play nicely together.

Chapter 5: Enumerations

There are a surprising range of applications for this simplest of Swift constructs. An enumeration is essentially just a collection of options in a given category. Enumeration members can have associated values for storing simple data or raw values for distinguishing the members. We can even add computed properties and methods to enumerations.

Chapter 6: Structs

Swift offers a powerful solution for encapsulating small bundles of data: structs. A struct can have both stored and computed properties. We look at how to create an instance of a struct and how to add methods to make structs easier to work with. Structs have a lot more power in Swift than they do in other languages. Often, we'll reach for them when instead you may be tempted to use a class.

Chapter 7: Classes

The third type is Swift's reference type: classes. We create a base class and subclasses and walk through the complexities of initializing stored properties up and down a hierarchy. We can add and override methods and properties in subclasses and add observers that respond when a property value is set. In this chapter, we work a bit more with optionals and look at how to use optional chaining.

Chapter 8: Protocols

Swift endeavors to change your mind about inheritance. Mostly, we want to inherit the signature of methods and not their implementations. Protocols collect method signatures and property declarations into convenient packages that enumerations, structs, and classes can choose to conform to. Swift also gives us the ability to specify behavior in a protocol extension.

Part Three: Thinking in Swift

Chapter 9: Flexible Functions

A recurring theme with Swift is that it makes code easier to reason about. In this chapter you see that by default a function makes immutable copies of the parameters that are passed to it. We then focus on creating non-mutating versions of functions that create a new instance of the type rather than changing its value.

Chapter 10: Higher-Order Functions

This chapter is so much fun! First we learn about functions that can return or consume other functions. We then study four canonical examples from the Swift Standard Library: `map()`, `filter()`, `reduce()`, and `flatMap()`.

Chapter 11: Errors

Of course we all try to write code that "just works" but often things go wrong. In this final chapter, we look at using

`Optional`s, asserts, and Swift `Error`s to handle issues. As with so many other parts of the language, Swift's `Error` mechanism is simple, clean, and powerful.

Wow. That's a lot to learn.

Let's get started with functions.