Developer Tools

#WWDC17

What's New in Swift Playgrounds

Session 408

Connor Wakamo, Playgrounds Engineer
Grace Kendall, Playgrounds Engineer

Najla Bulous, Playgrounds Engineer

Playground book review

Enhancements for playground books

New Bluetooth API for playgrounds

Playground Book Review


BOOK

Playground Book

Split into chapters and pages

Playground Book

Split into chapters and pages

May contain resources

Playground Book

Split into chapters and pages

May contain resources

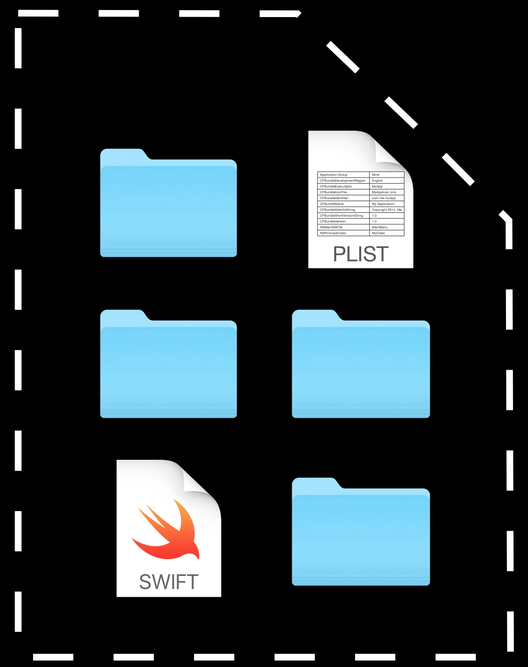
Package-based format

Playground Book

Split into chapters and pages

May contain resources

Package-based format



Manifest Files

Like an app’s Info.plist

Manifest Files

Like an app’s Info.plist

Provide book/chapter/page-level metadata

- Name
- Icon

Manifest Files

Like an app's Info.plist

Provide book/chapter/page-level metadata

- Name
- Icon

Specify options for the book/chapter/page

- Initial live view state
- Playground logging

Swift Files

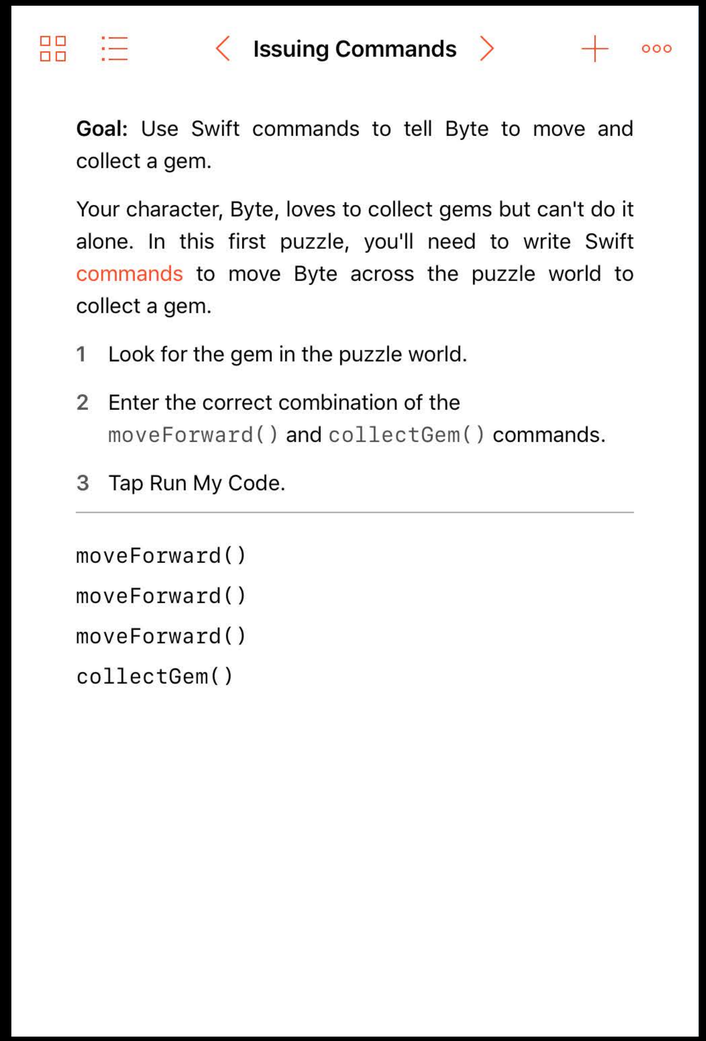
Swift Files

Three kinds of Swift files

Swift Files

Three kinds of Swift files

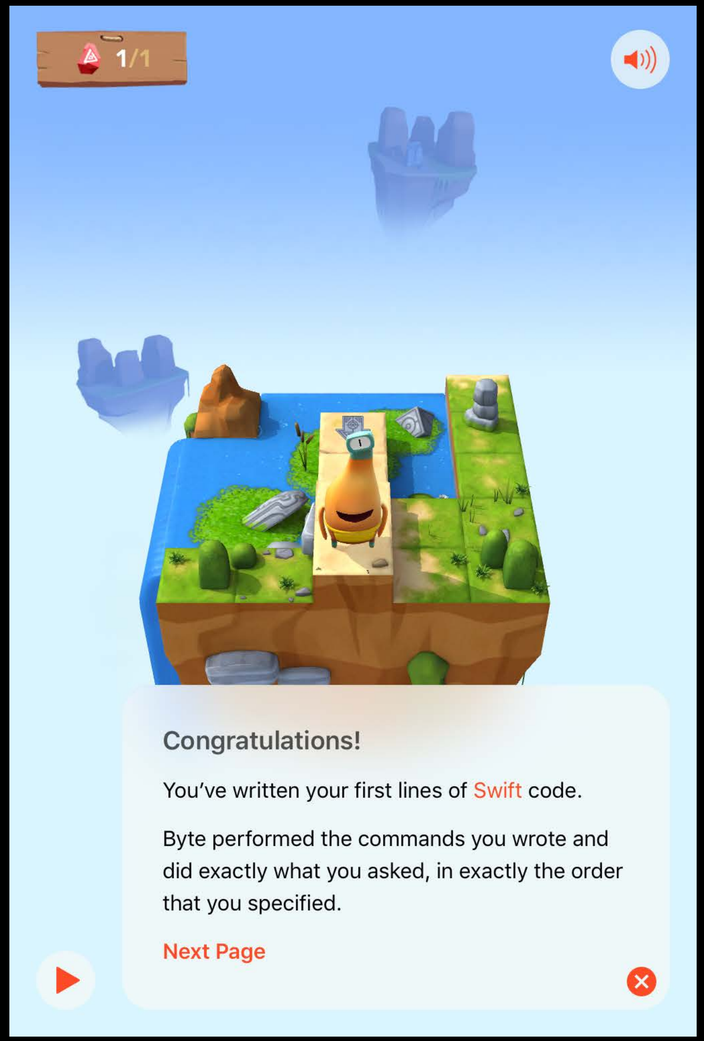
- Contents.swift



Swift Files

Three kinds of Swift files

- Contents.swift
- LiveView.swift



Swift Files

Three kinds of Swift files

- Contents.swift
- LiveView.swift
- Auxiliary sources

Swift Files

Three kinds of Swift files

- Contents.swift
- LiveView.swift
- Auxiliary sources

New Features in Swift Playgrounds

Add Actions

Another way to add an action is by defining an array. You include actions that the new action beats within the `[` and `]` brackets, separated by commas.

> Example
>
> ```
> hardRock.beats([rock, scissors])
> ```

1. Add an action to the game that beats two or more other actions.
2. Add an action that loses to all other actions. **Tip:** You can get an array of all the actions in the game by calling `game.actions`.

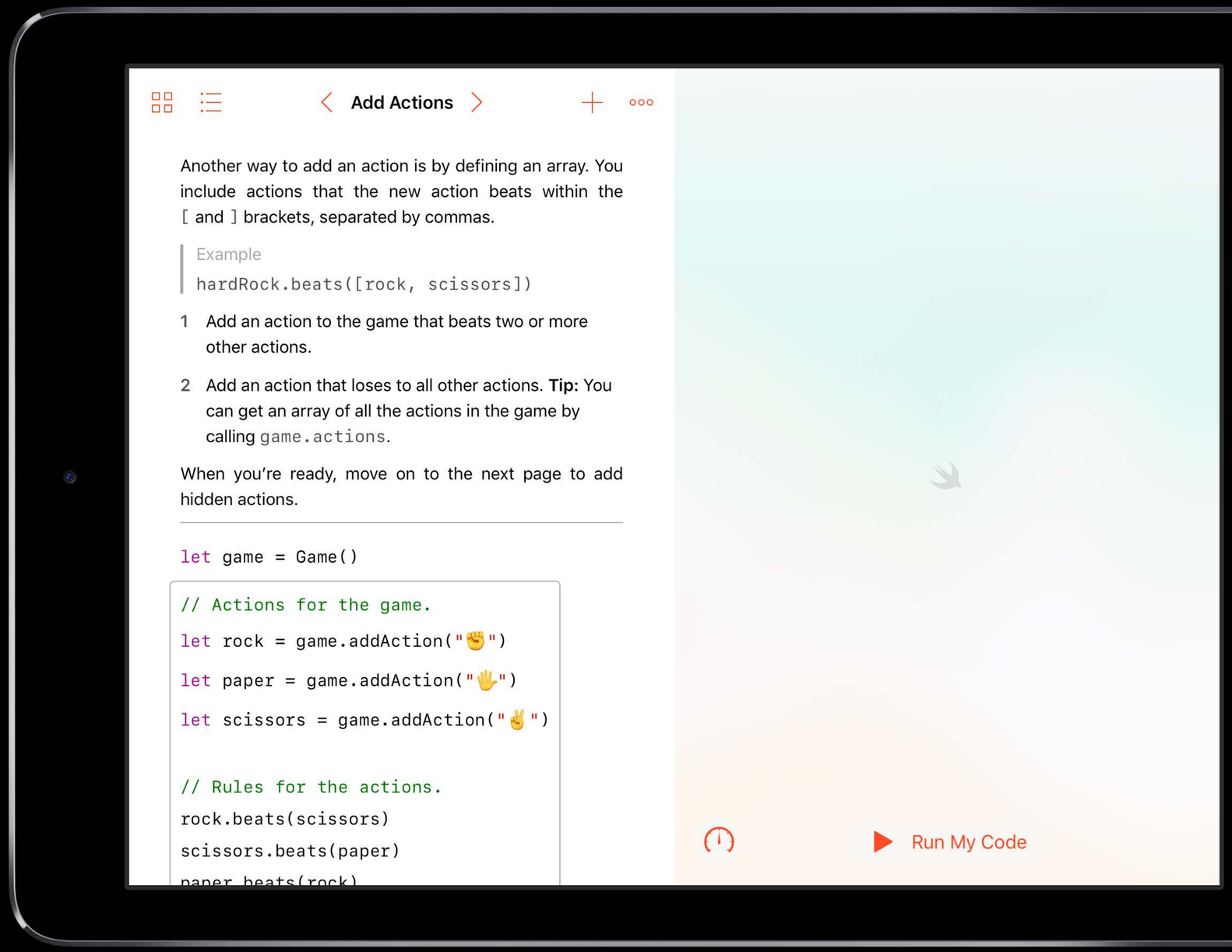
When you're ready, move on to the next page to add hidden actions.

You can bring over your personalized code from the previous page to continue improving it.

Bring Over My Code

Start Coding on This Page

Run My Code


Add Actions
Another way to add an action is by defining an array. You include actions that the new action beats within the [and] brackets, separated by commas.
Example
hardRock.beats([rock, scissors])
1 Add an action to the game that beats two or more other actions.
2 Add an action that loses to all other actions. Tip: You can get an array of all the actions in the game by calling game.actions.
When you're ready, move on to the next page to add hidden actions.
let game = Game()
// Actions for the game.
let rock = game.addAction("✊")
let paper = game.addAction("🖐")
let scissors = game.addAction("✌️")
// Rules for the actions.
rock.beats(scissors)
scissors.beats(paper)
paper.beats(rock)
Run My Code

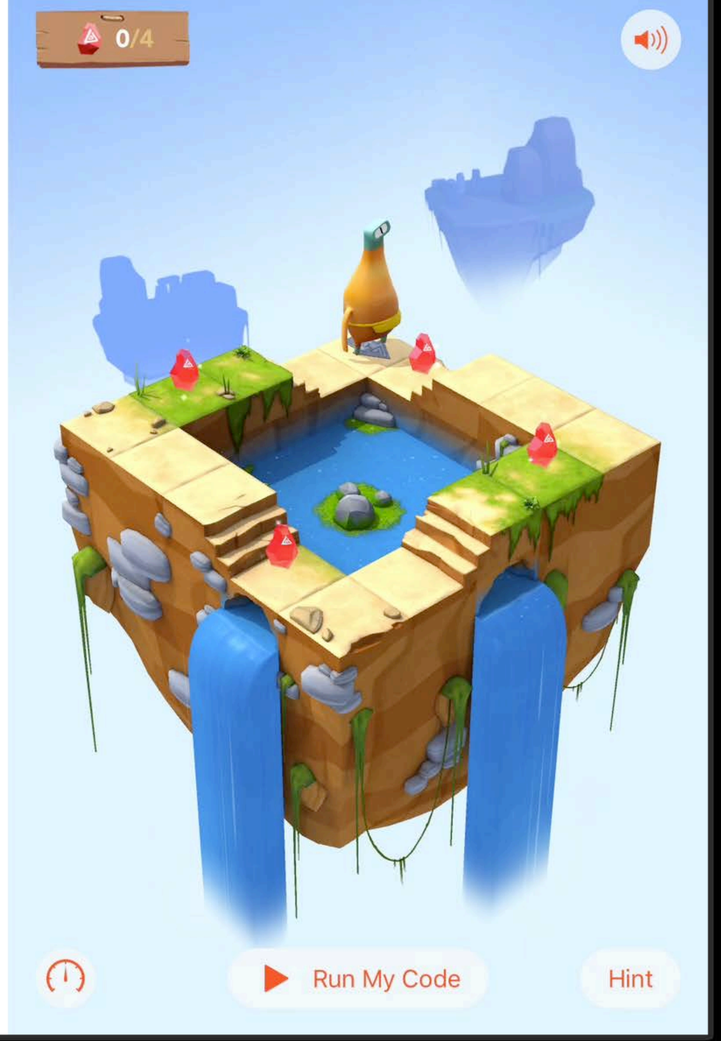
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```



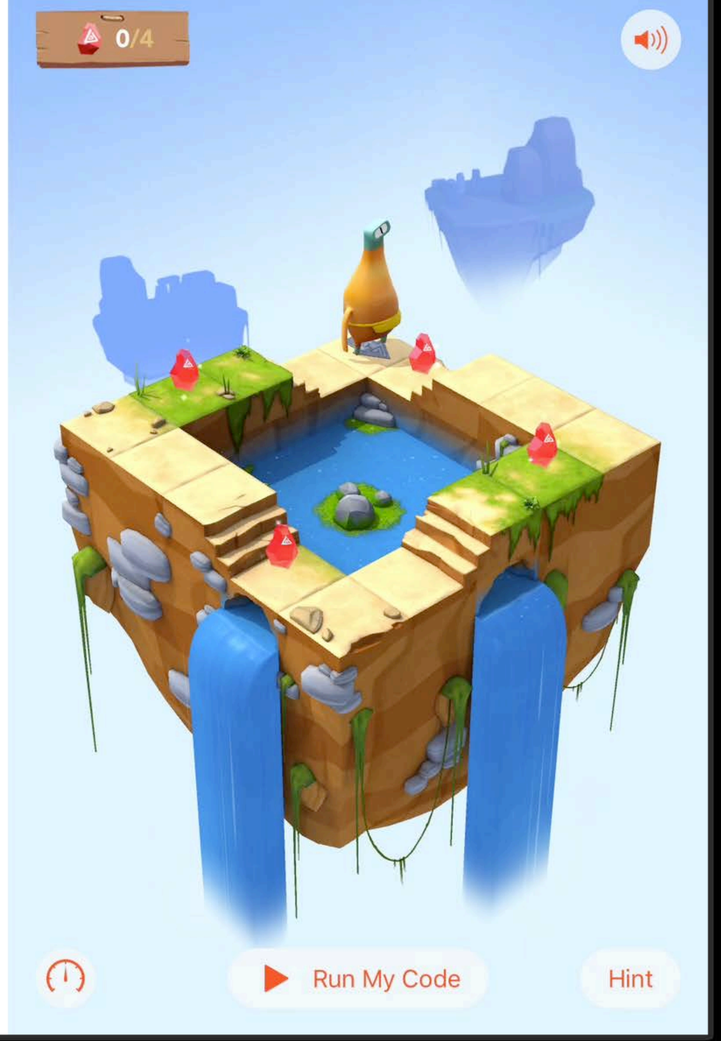
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```



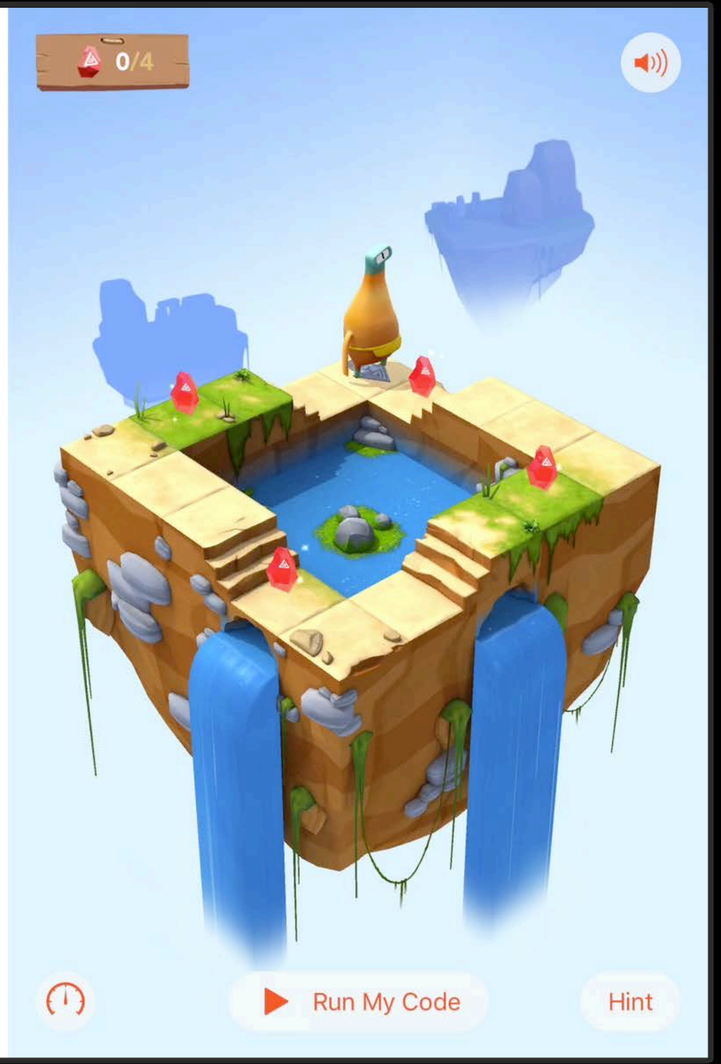
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```



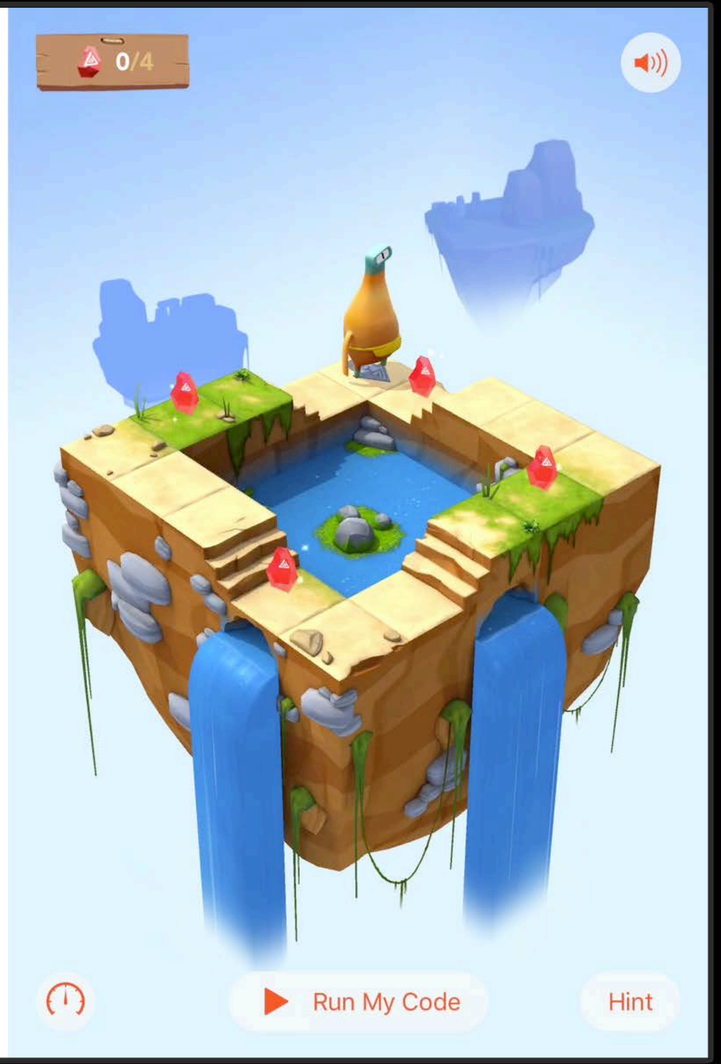
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

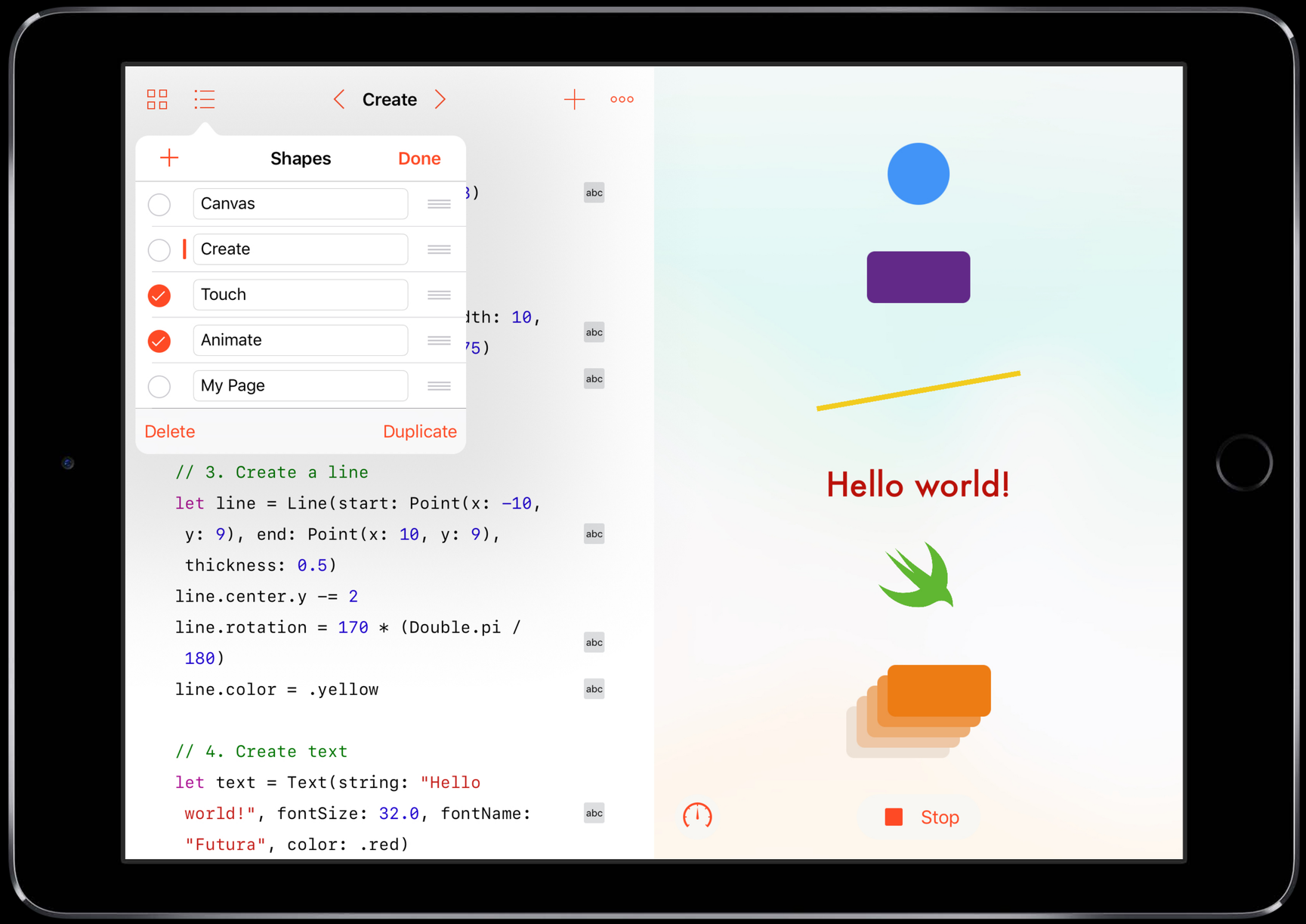
In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```




My Playground
let array = ["first", "second", "third", "fourth ", "fifth"]
array[5]
Index out of range
Run My Code


Create
Shapes
Done
Canvas
Create
Touch
Animate
My Page
Delete
Duplicate
// 3. Create a line
let line = Line(start: Point(x: -10,
y: 9), end: Point(x: 10, y: 9),
thickness: 0.5)
line.center.y -= 2
line.rotation = 170 * (Double.pi /
180)
line.color = .yellow
// 4. Create text
let text = Text(string: "Hello
world!", fontSize: 32.0, fontName:
"Futura", color: .red)
Hello world!
Stop

发出命令

目标： 使用 Swift 命令，让 Byte 动起来，去收集宝石。

你的角色 Byte 喜欢收集宝石，但它一个人做不到。在第一关里，你需要编写 Swift 命令，让 Byte 在关卡世界中动起来，去收集宝石。

1. 找到关卡世界中的宝石。
2. 输入正确的 `moveForward()`（向前走）和 `collectGem()`（收集宝石）命令组合，让 Byte 向前走，去收集宝石。
3. 轻点“运行我的代码”。

```
moveForward()
moveForward()
moveForward()
collectGem()
```

コマンドを使う

目標: Swiftコマンドを使ってByteを動かして、宝石を集めましょう。

主人公のByteは宝石を集めるのが大好き。でも自分一人ではできません。Swiftコマンドを使ってステージ内でByteを動かして、宝石を取るお手伝いをしてください。

1 宝石の位置を確かめます。

2 `moveForward()`（進む）コマンドと`collectGem()`（宝石を取る）コマンドを正しい順序で入力します。

3 “コードを実行”をタップします。

```
moveForward()
moveForward()
moveForward()
collectGem()
```

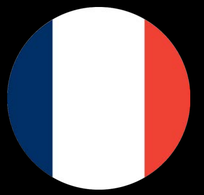
1/1

おめでとうございます!

はじめてのSwiftコードを書きましたね。

書いたコマンドをByteが実行して、頼んだことを指示通りの順序で実行してくれました。

次のページ



Création de commandes

Objectif : Utiliser les commandes Swift pour dire à Octet de se déplacer et de collecter une gemme.

Ton personnage, Octet, aime collecter les gemmes, mais il ne peut pas y arriver tout seul. Dans ce premier puzzle, tu devras écrire des commandes Swift pour déplacer Octet à travers le puzzle et collecter une gemme.

1. Cherche la gemme dans le monde du puzzle.
2. Saisis la combinaison correcte de commandes `moveForward()` et `collectGem()`.
3. Touche Exécuter mon code.

```
moveForward()
moveForward()
moveForward()
collectGem()
```

Félicitations !

Tu as rédigé tes premières lignes de code Swift.

Octet a effectué les commandes que tu as rédigées, et a fait exactement ce que tu lui as indiqué, dans l'ordre exact que tu lui avais signifié.

Page suivante

Befehle erteilen

Ziel: Verwende Swift-Befehle, um Byte zu sagen, dass er sich bewegen und die Edelsteine einsammeln soll.

Dein Charakter Byte liebt es, Edelsteine zu sammeln, kann das aber nicht alleine. Im ersten Rätsel musst du Swift-Befehle schreiben, um Byte durch die Rätselwelt zu bewegen und einen Edelstein einzusammeln.

1. Suche in der Rätselwelt nach dem Edelstein.
2. Gib die korrekte Kombination aus den Befehlen `moveForward()` und `collectGem()` ein.
3. Tippe auf „Meinen Code ausführen".

```
moveForward()
moveForward()
moveForward()
collectGem()
```

1/1

Herzlichen Glückwunsch!

Du hast deine ersten Codezeilen in Swift geschrieben.

Byte hat die Befehle, die du geschrieben hast, ausgeführt, und genau das in genau der Reihenfolge gemacht, wie von dir angegeben.

Nächste Seite

Creación de comandos

Objetivo: usa comandos Swift para hacer que Byte se mueva y recolecte una gema.

A tu personaje, Byte, le encanta recolectar gemas. Sin embargo, no lo puede hacer solo. En este primer rompecabezas, deberás escribir comandos Swift para hacer que Byte se mueva por el rompecabezas y recolecte una gema.

1. Busca la gema en el rompecabezas.
2. Ingresa la combinación correcta de comandos `moveForward()` (“avanzar”) y `collectGem()` (“recolectar gema”).
3. Toca “Ejecutar mi código”.

```
moveForward()
moveForward()
moveForward()
collectGem()
```

1/1

¡Felicidades!

Escribiste tus primeras líneas de código Swift.

Byte realizó los comandos que escribiste e hizo exactamente lo que pediste y en el orden en que lo especificaste.

Siguiente página

iOS 11 SDK

BETA

iOS 11 SDK

BETA

New frameworks added to the SDK in Swift Playgrounds

- ARKit
- CoreML
- IOSurface
- PDFKit
- Vision

iOS 11 SDK

BETA

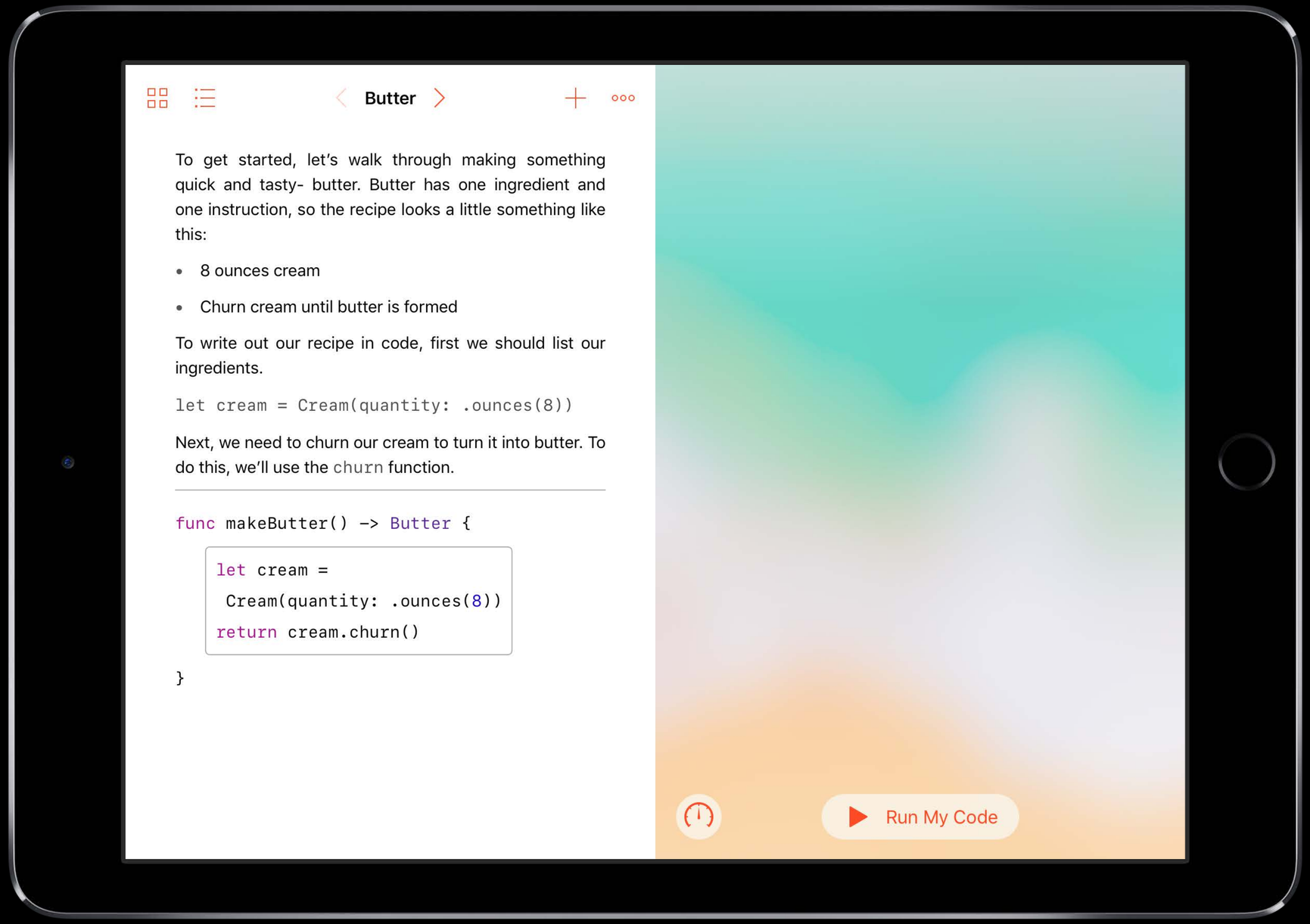
New frameworks added to the SDK in Swift Playgrounds

- ARKit
- CoreML
- IOSurface
- PDFKit
- Vision

Playgrounds may now access the camera

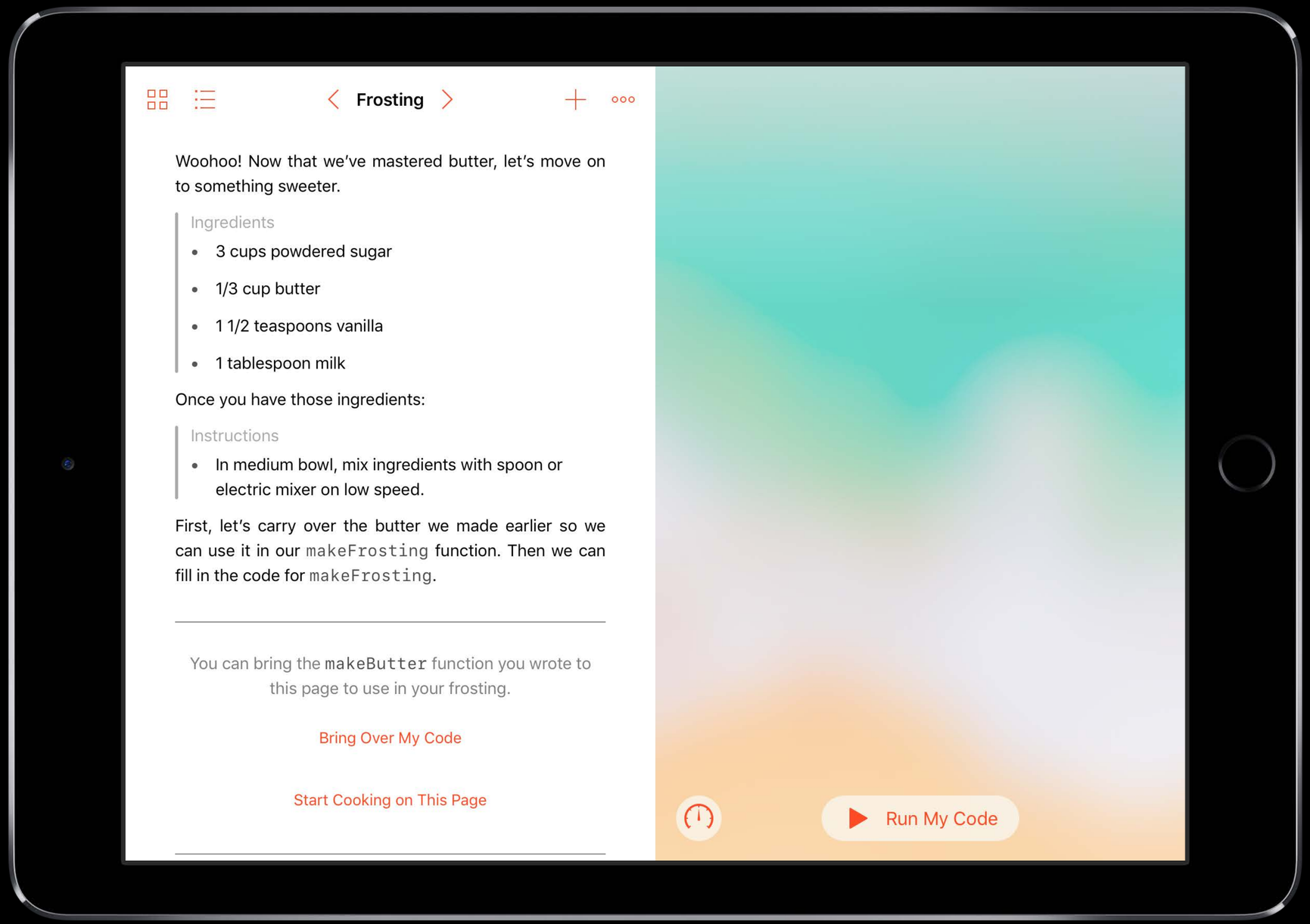
Demo

Copying Code Between Pages


Butter
To get started, let's walk through making something quick and tasty- butter. Butter has one ingredient and one instruction, so the recipe looks a little something like this:
8 ounces cream
Churn cream until butter is formed
To write out our recipe in code, first we should list our ingredients.
let cream = Cream(quantity: .ounces(8))
Next, we need to churn our cream to turn it into butter. To do this, we'll use the churn function.
func makeButter() -> Butter {
let cream =
Cream(quantity: .ounces(8))
return cream.churn()
}
Run My Code


Butter
To get started, let's walk through making something quick and tasty- butter. Butter has one ingredient and
//#-copy-source(butter)
func makeButter() -> Butter {
//#-editable-code Tap to enter code
let cream = Cream(quantity: .ounces(8))
return cream.churn()
//#-end-editable-code
}
//#-end-copy-source
Run My Code


Butter
To get started, let's walk through making something quick and tasty- butter. Butter has one ingredient and
//#-copy-source(butter)
func makeButter() -> Butter {
//#-editable-code Tap to enter code
let cream = Cream(quantity: .ounces(8))
return cream.churn()
//#-end-editable-code
}
//#-end-copy-source
Run My Code


Frosting
Woohoo! Now that we've mastered butter, let's move on to something sweeter.
Ingredients
3 cups powdered sugar
1/3 cup butter
1 1/2 teaspoons vanilla
1 tablespoon milk
Once you have those ingredients:
Instructions
In medium bowl, mix ingredients with spoon or electric mixer on low speed.
First, let's carry over the butter we made earlier so we can use it in our makeFrosting function. Then we can fill in the code for makeFrosting.
You can bring the makeButter function you wrote to this page to use in your frosting.
Bring Over My Code
Start Cooking on This Page
Run My Code

Frosting

Woohoo! Now that we've mastered butter, let's move on to something sweeter.

> Ingredients
>
> - 3 cups powdered sugar
> - 1/3 cup butter
> - 1 1/2 teaspoons vanilla
> - 1 tablespoon milk

Once you have those ingredients:

> Instructions
>
> - In medium bowl, mix ingredients with spoon or electric mixer on low speed.

First, let's carry over the butter we made earlier so we can use it in our `makeFrosting` function. Then we can fill in the code for `makeFrosting`.

Go back to finish making butter?

Return to Previous Page

Start Cooking on This Page

Run My Code

Frosting

Woohoo! Now that we've mastered butter, let's move on to something sweeter.

> Ingredients
>
> - 3 cups powdered sugar
> - 1/3 cup butter
> - 1 1/2 teaspoons vanilla
> - 1 tablespoon milk

Once you have those ingredients:

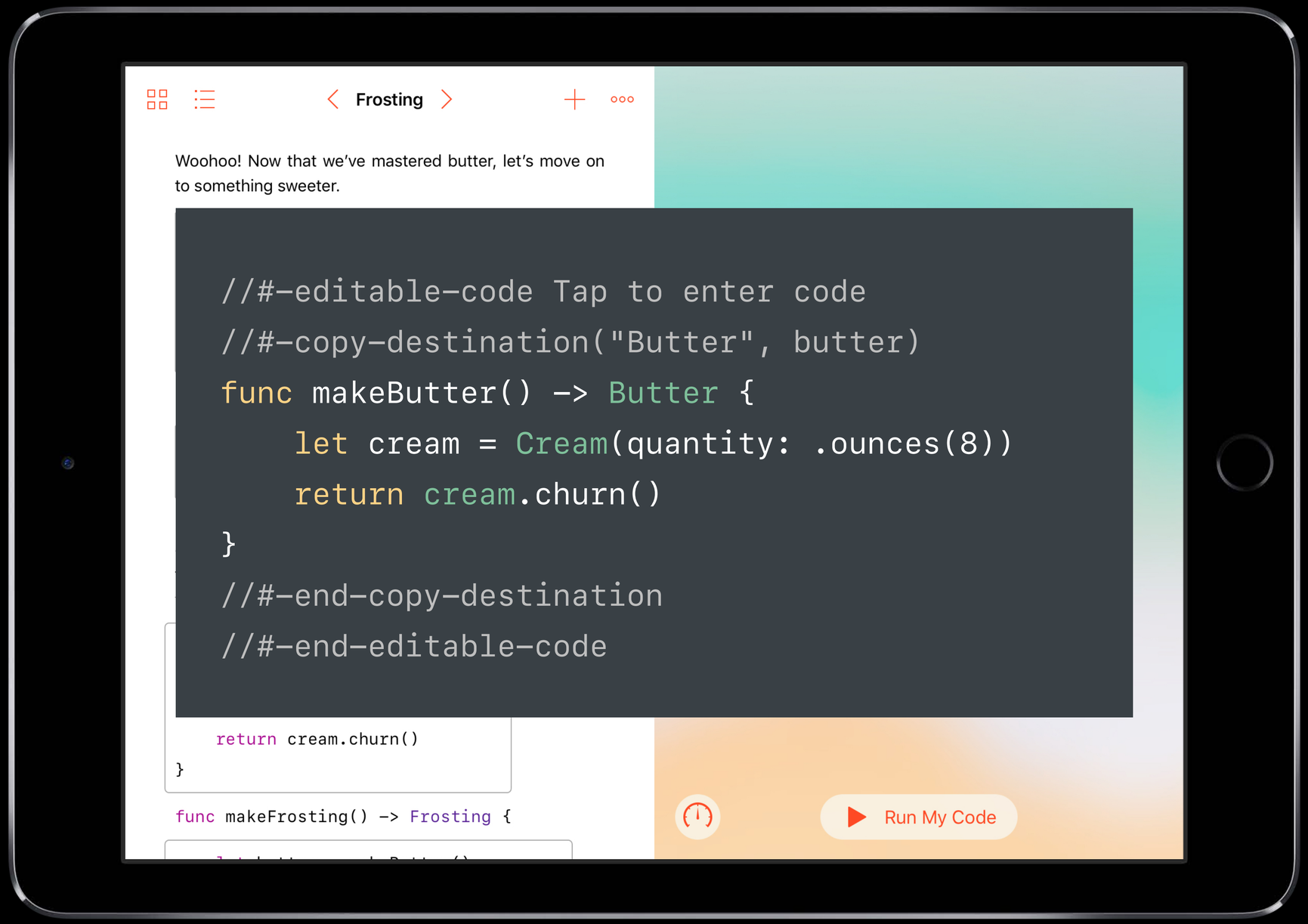
> Instructions
>
> - In medium bowl, mix ingredients with spoon or electric mixer on low speed.

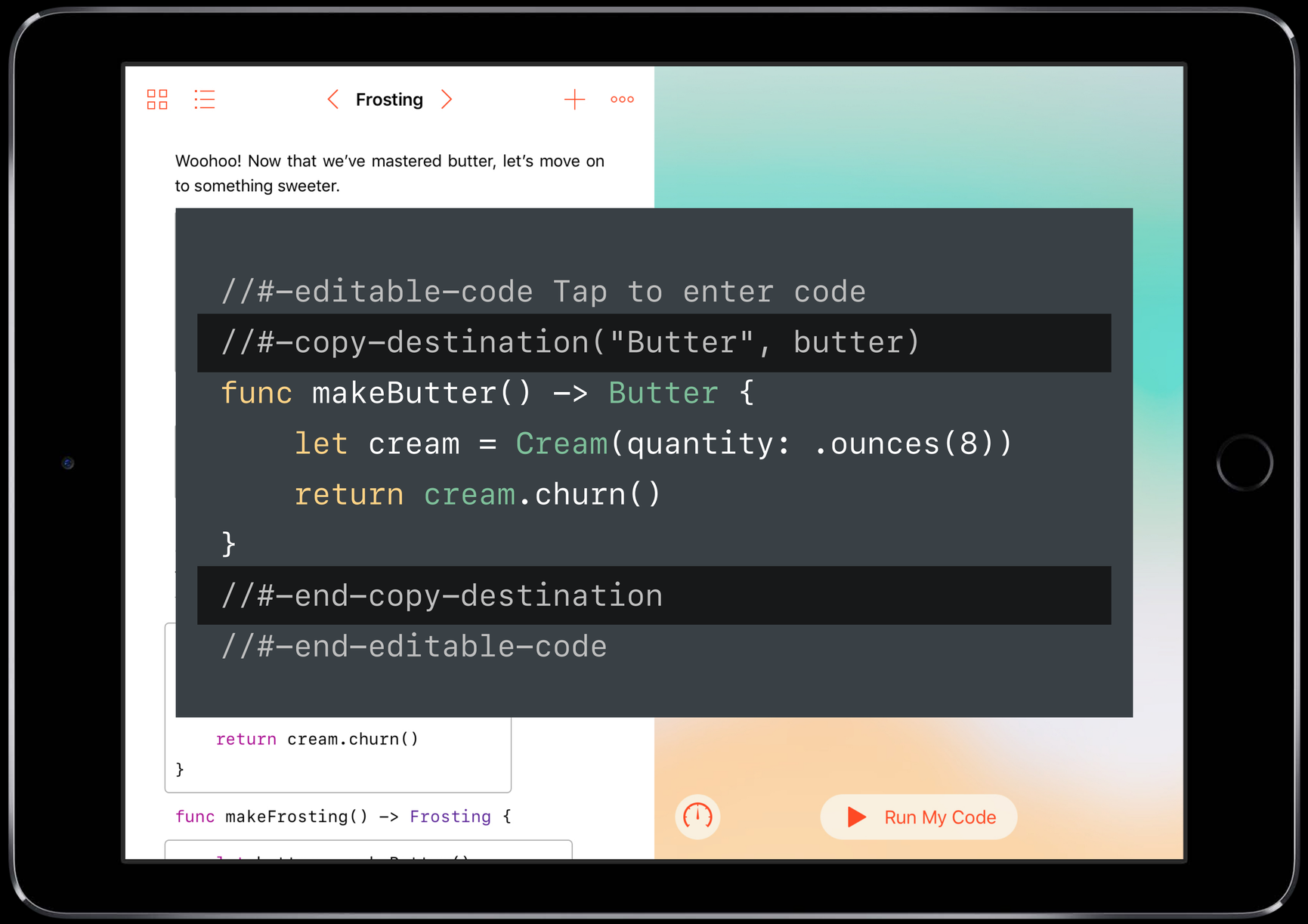
First, let's carry over the butter we made earlier so we can use it in our `makeFrosting` function. Then we can fill in the code for `makeFrosting`.

```
func makeButter() -> Butter {
    let cream =
     Cream(quantity: .ounces(8))
    return cream.churn()
}
```

```
func makeFrosting() -> Frosting {
```

Run My Code


Frosting
Woohoo! Now that we've mastered butter, let's move on to something sweeter.
//#-editable-code Tap to enter code
//#-copy-destination("Butter", butter)
func makeButter() -> Butter {
let cream = Cream(quantity: .ounces(8))
return cream.churn()
}
//#-end-copy-destination
//#-end-editable-code
return cream.churn()
}
func makeFrosting() -> Frosting {
Run My Code


Frosting
Woohoo! Now that we've mastered butter, let's move on to something sweeter.
//#-editable-code Tap to enter code
//#-copy-destination("Butter", butter)
func makeButter() -> Butter {
let cream = Cream(quantity: .ounces(8))
return cream.churn()
}
//#-end-copy-destination
//#-end-editable-code
return cream.churn()
}
func makeFrosting() -> Frosting {
Run My Code

Copying Code Between Pages

Required Manifest.plist Keys

Key	Type	Value
▼ Root	Dictionary	
▼ CodeCopySetup	Dictionary	
ReadyToCopyInstructions	String	You can bring the `makeButter` function you wrote to this page to...
NotReadyToCopyInstructions	String	Go back to finish making butter?

Copying Code Between Pages

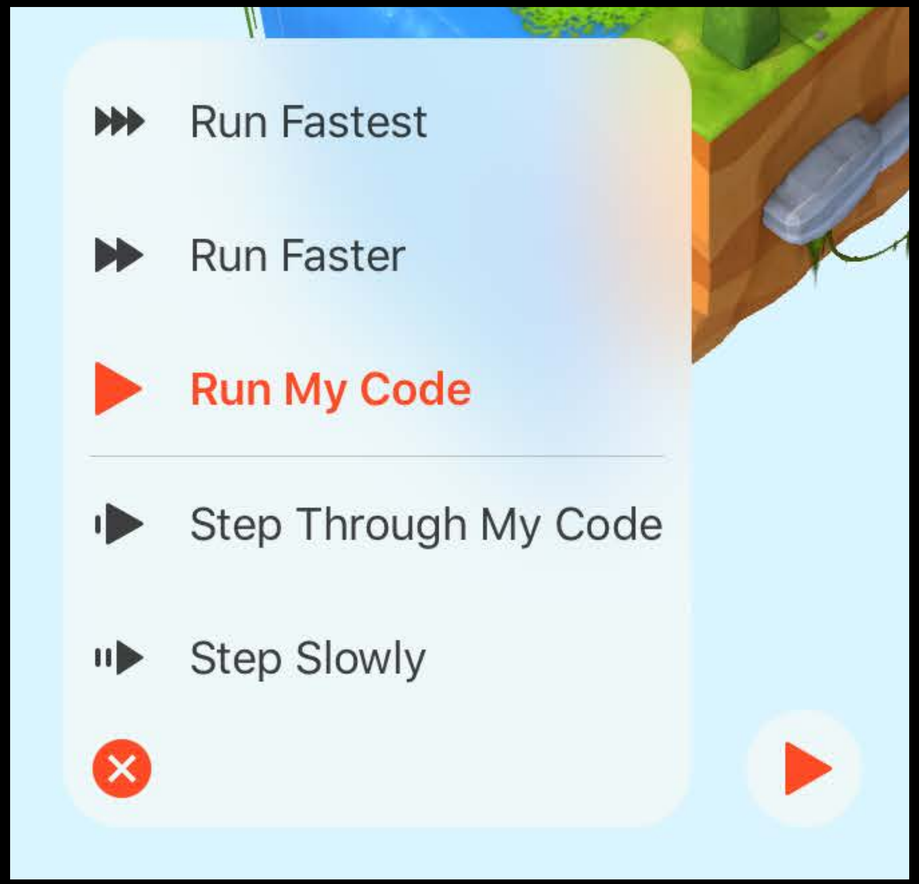
Optional Manifest.plist Keys

Key	Type	Value
▼ Root	Dictionary	
▼ CodeCopySetup	Dictionary	
CopyCommandButtonTitle	String	Bring Over My Code
NavigateCommandButtonTitle	String	Return to Previous Page
DefaultCommandButtonTitle	String	Start Cooking on This Page

Controlling Speed in Playgrounds

Controlling Speed in Playgrounds

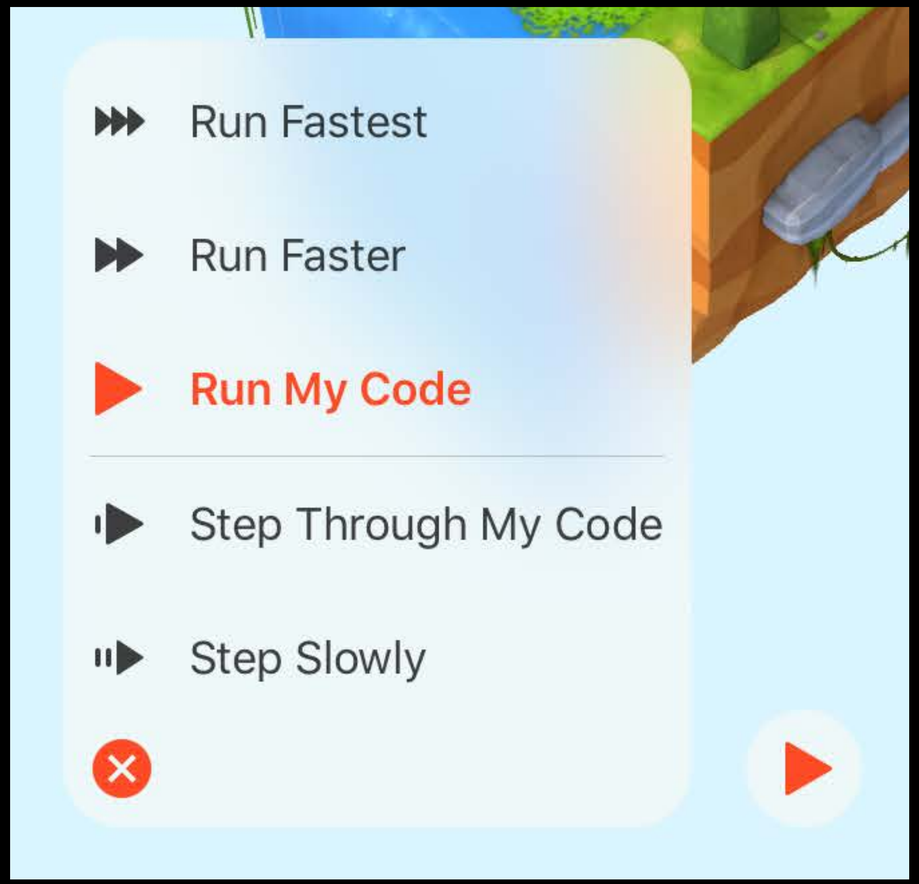
Swift Playgrounds now supports stepping through code and running faster



Controlling Speed in Playgrounds

Swift Playgrounds now supports stepping through code and running faster

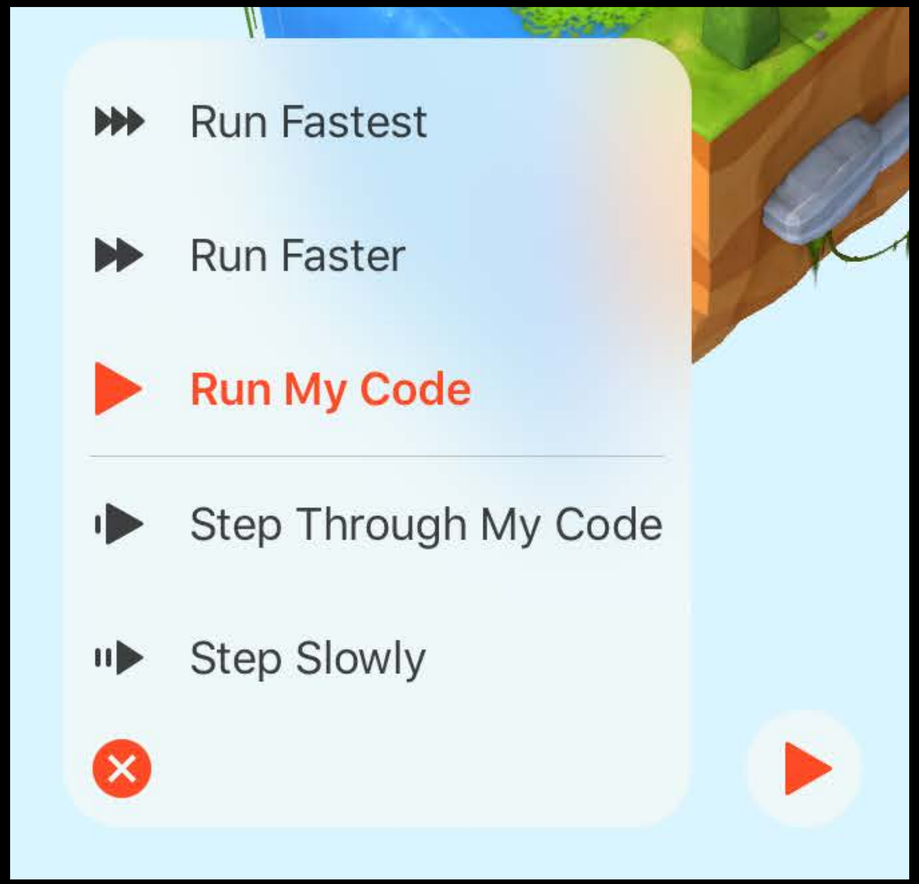
- Step Through My Code is enabled for all playgrounds



Controlling Speed in Playgrounds

Swift Playgrounds now supports stepping through code and running faster

- Step Through My Code is enabled for all playgrounds
- Playground books may opt in to Run Faster and Run Fastest



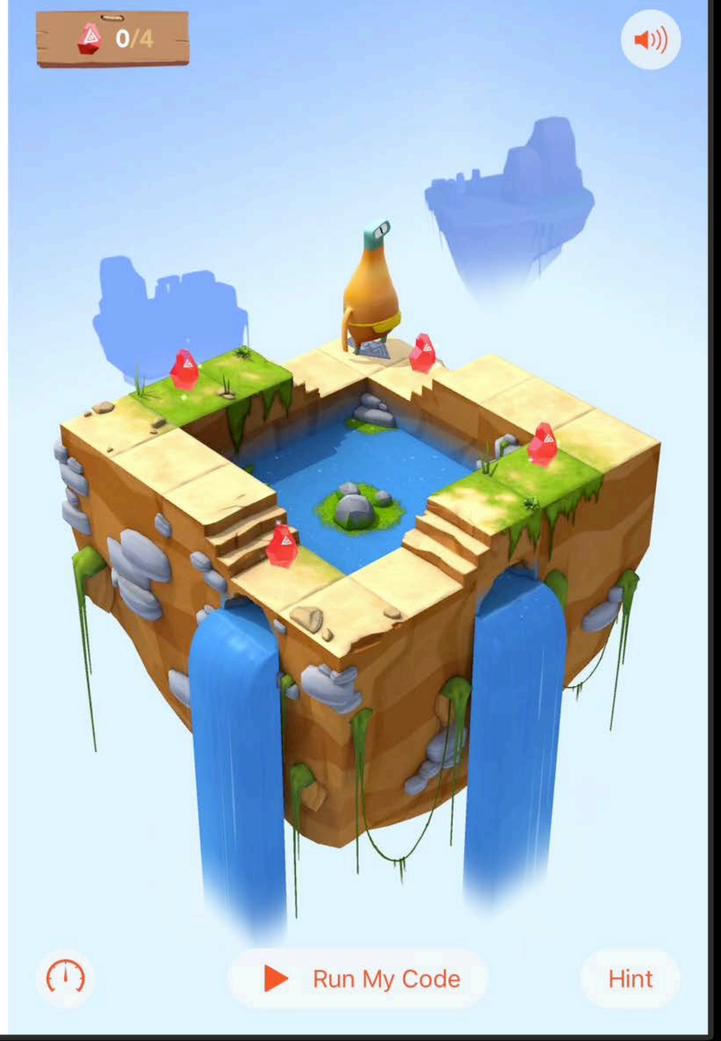
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```



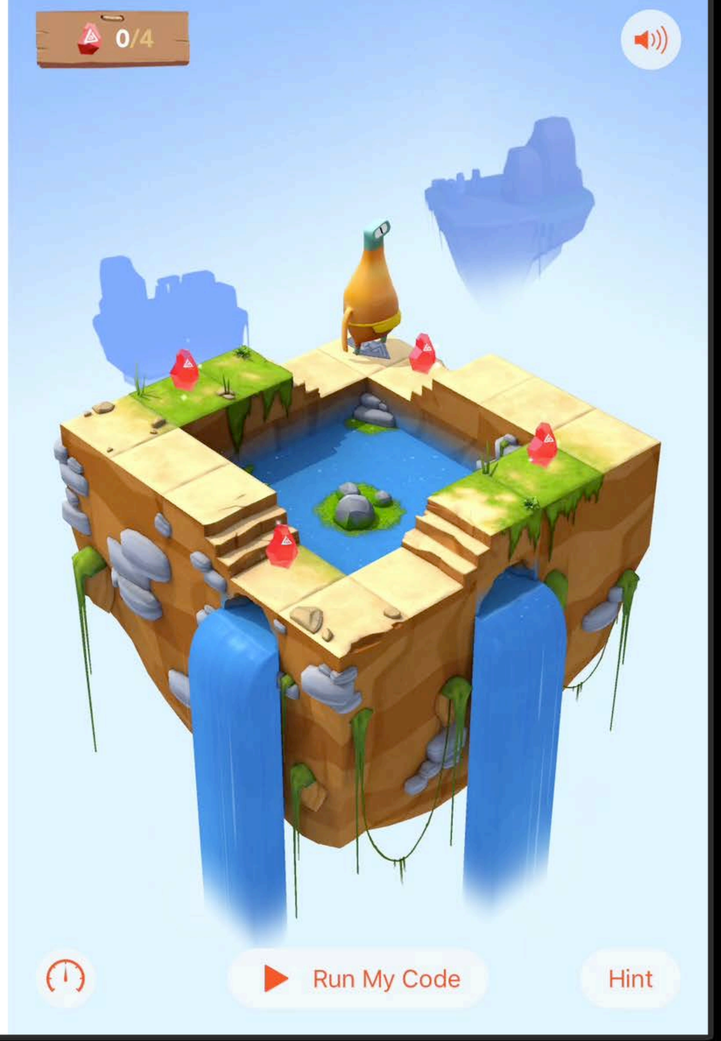
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```



```
// Step Through My Code

let string = "Hello, world"
for character in string {
    print(character)
}
```

```
// Step Through My Code

public func moveForward() {
    DispatchQueue.global(.default).async {
        // Do work
    }
}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work




    }



}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work




    }



}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work




    }
    while !didFinish {
        CFRunLoopRun()
    }
}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work




    }
    while !didFinish {
        CFRunLoopRun()
    }
}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work
        didFinish = true
        CFRunLoopPerformBlock(runLoop) {
            CFRunLoopStop(runLoop)
        }
        CFRunLoopWakeUp(runLoop)
    }
    while !didFinish {
        CFRunLoopRun()
    }
}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work
        didFinish = true
        CFRunLoopPerformBlock(runLoop) {
            CFRunLoopStop(runLoop)
        }
        CFRunLoopWakeUp(runLoop)
    }
    while !didFinish {
        CFRunLoopRun()
    }
}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work
        didFinish = true
        CFRunLoopPerformBlock(runLoop) {
            CFRunLoopStop(runLoop)
        }
        CFRunLoopWakeUp(runLoop)
    }
    while !didFinish {
        CFRunLoopRun()
    }
}
```

```
// Step Through My Code

public func moveForward() {
    let runLoop = CFRunLoopGetCurrent()
    var didFinish = false
    DispatchQueue.global(.default).async {
        // Do work
        didFinish = true
        CFRunLoopPerformBlock(runLoop) {
            CFRunLoopStop(runLoop)
        }
        CFRunLoopWakeUp(runLoop)
    }
    while !didFinish {
        CFRunLoopRun()
    }
}
```

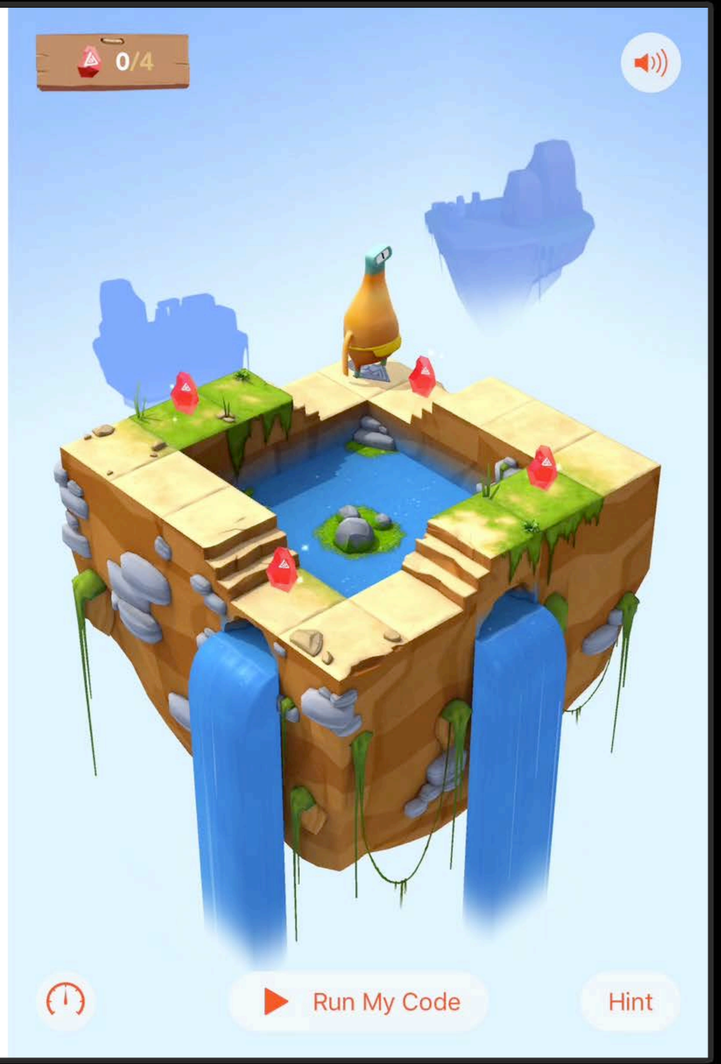
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```



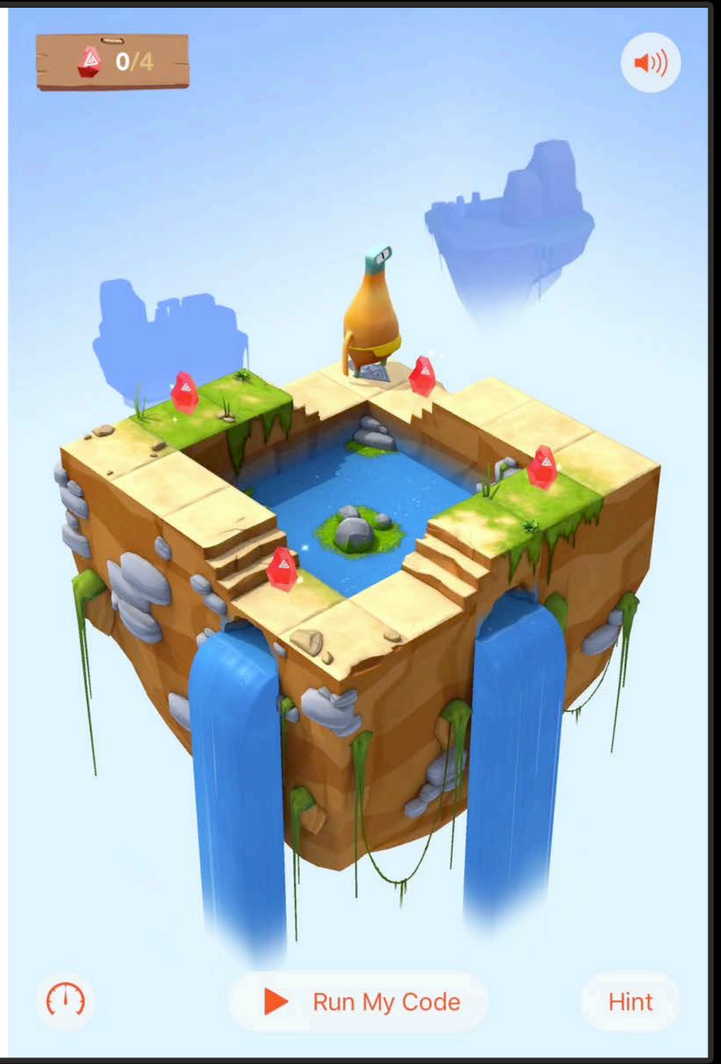
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a `for` loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```



Run Faster and Run Fastest

Manifest.plist Key

Key	Type	Value
▼ Root	Dictionary	
MaximumSupportedExecutionSpeed	String	Fastest

Allowed execution speeds

- Normal (default)
- Faster
- Fastest

```
// Run Faster and Run Fastest

let observerToken =
NotificationCenter.default.addObserver(.playgroundPageExecutionModeDidChange,
                                       object: PlaygroundPage.current,
                                       queue: .main) {
    let newMode = PlaygroundPage.current.executionMode
    // Update animation speed etc. for new mode
}
```

```
// Run Faster and Run Fastest

let observerToken =
NotificationCenter.default.addObserver(.playgroundPageExecutionModeDidChange,
                                       object: PlaygroundPage.current,
                                       queue: .main) {
    let newMode = PlaygroundPage.current.executionMode
    // Update animation speed etc. for new mode
}
```

```
// Run Faster and Run Fastest

let observerToken =
NotificationCenter.default.addObserver(.playgroundPageExecutionModeDidChange,
                                       object: PlaygroundPage.current,
                                       queue: .main) {
    let newMode = PlaygroundPage.current.executionMode
    // Update animation speed etc. for new mode
}
```

Playground Book Enhancements

Grace Kendall, Playgrounds Engineer

Minimum Swift Playgrounds Version

Key	Type	Value
▼ Root	Dictionary	
Version	String	3.0
DevelopmentRegion	String	en
Name	String	RecipeBook
MinimumSwiftPlaygroundsVersion	String	1.5
DeploymentTarget	String	ios10.3
SwiftVersion	String	3.1

Deployment Target and Swift Version

Key	Type	Value
▼ Root	Dictionary	
Version	String	3.0
DevelopmentRegion	String	en
Name	String	RecipeBook
MinimumSwiftPlaygroundsVersion	String	1.5
DeploymentTarget	String	ios10.3
SwiftVersion	String	3.1

Subtitles for Your Document

Playground books may now encode a subtitle

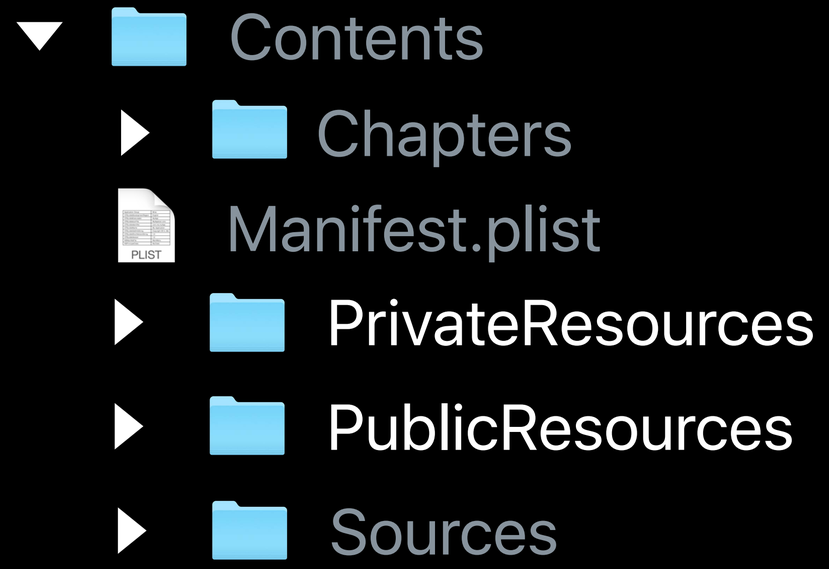
Subtitle key in book-level Manifest.plist

Shown below the book in the document picker

Public vs. Private Resources

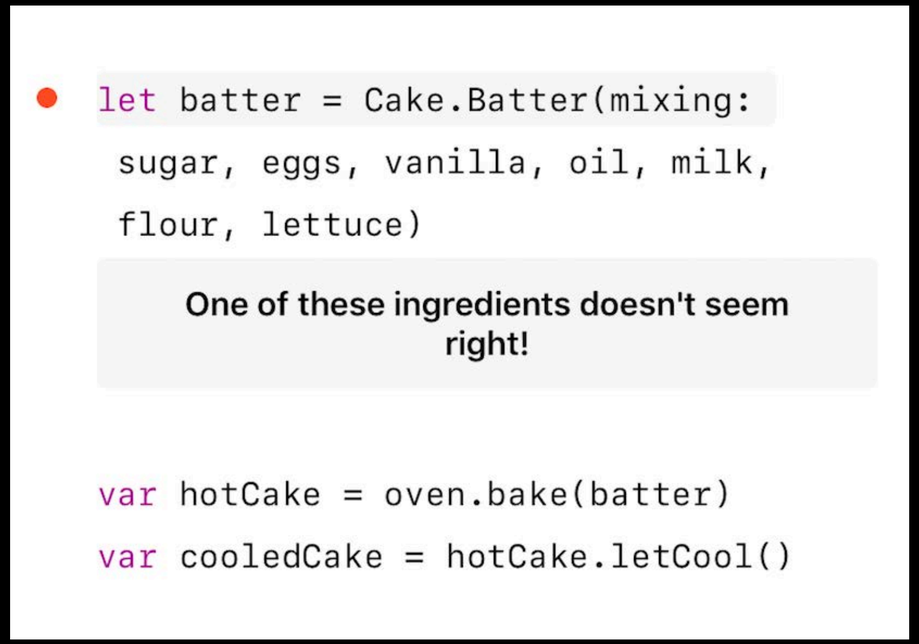
Replaced single Resources directory with PublicResources and PrivateResources

All resources available at runtime or in markup



Specifying Runtime Issues

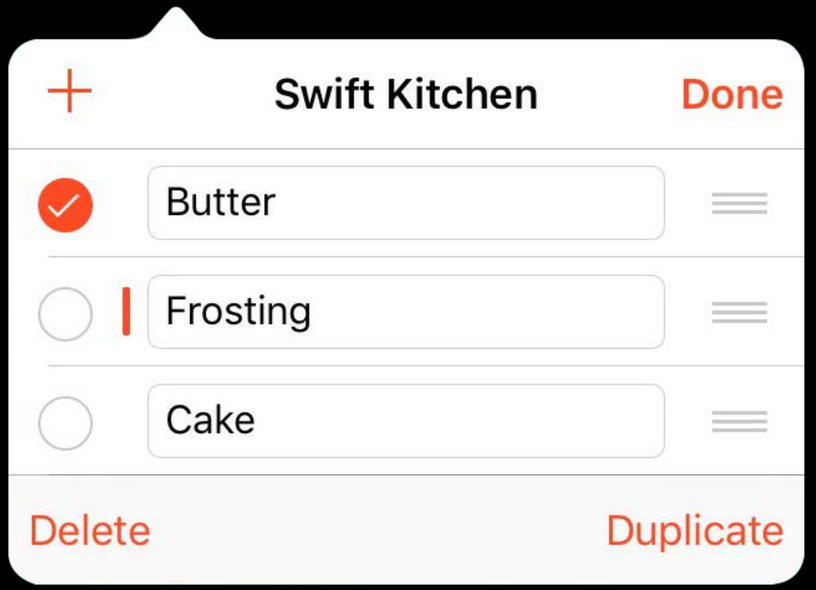
```
fatalError("One of these ingredients
doesn’t seem right!")
```



Templates in Swift Playgrounds

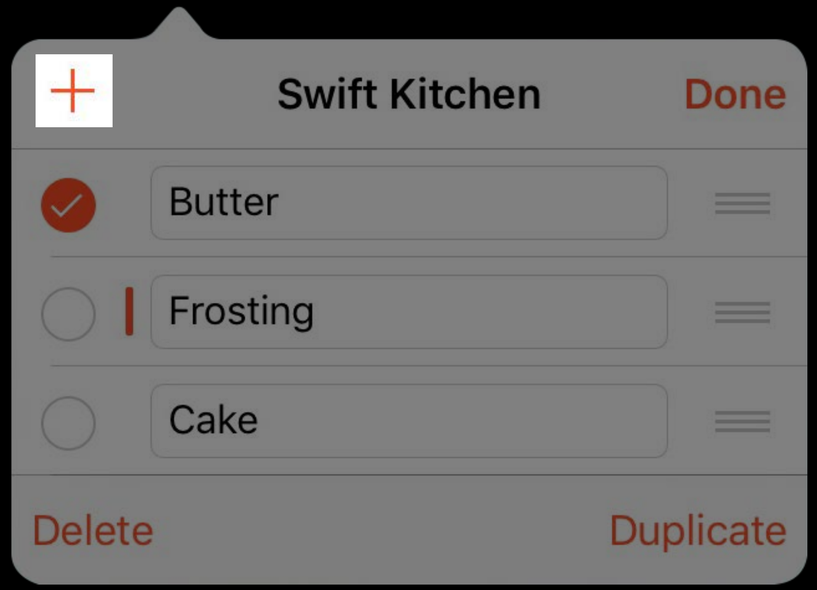
Supporting user-editing in playground books

User Editable Books



User Editable Books

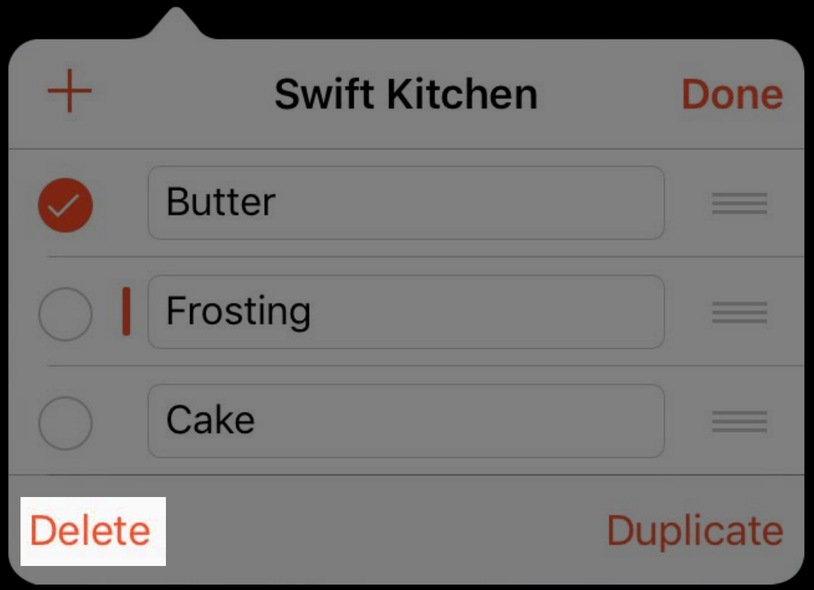
Add



User Editable Books

Add

Remove

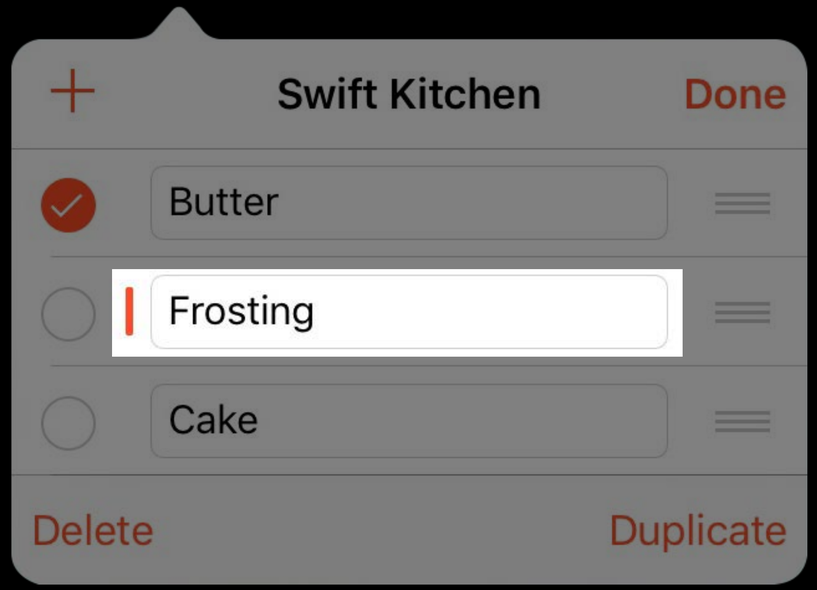


User Editable Books

Add

Remove

Rename



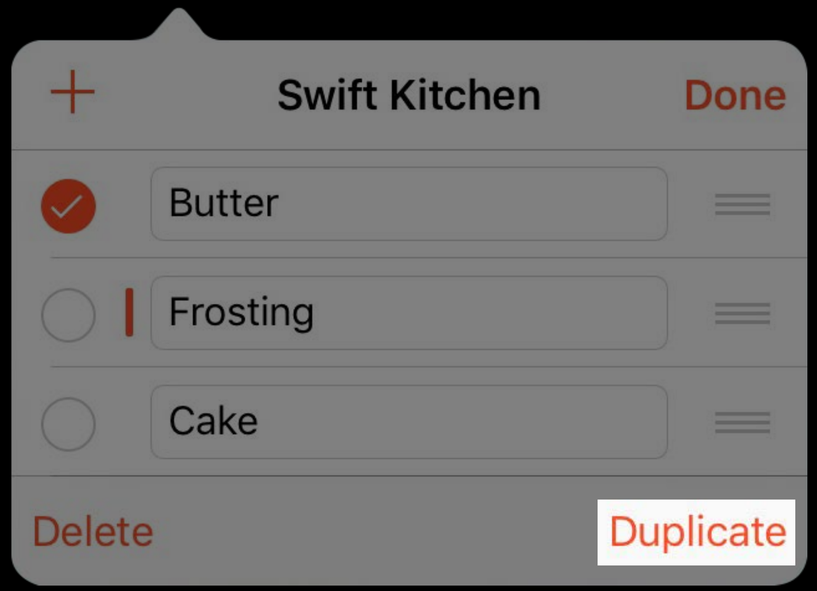
User Editable Books

Add

Remove

Rename

Duplicate



User Editable Books

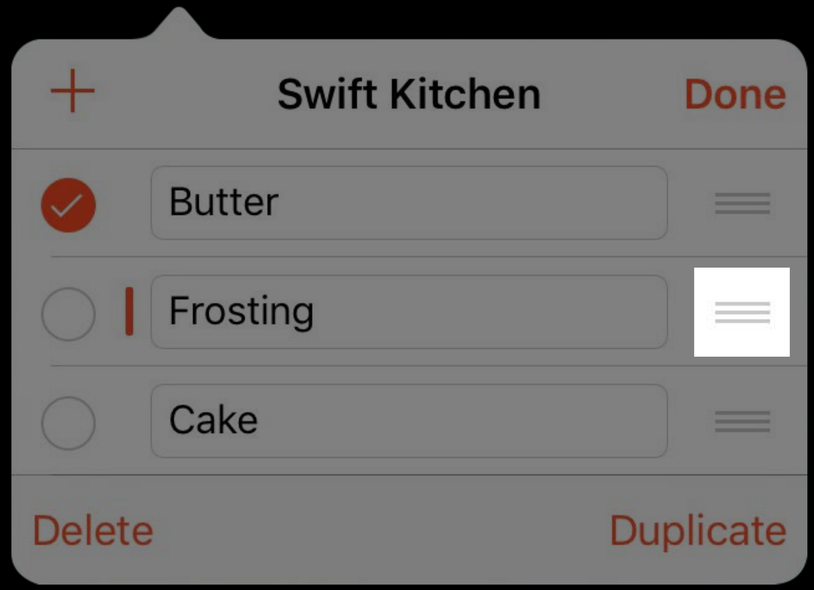
Add

Remove

Rename

Duplicate

Reorder



Demo

Specifying a Template Page

Manifest.plist

Key	Type	Value
▼ Root	Dictionary	
TemplatePageFilename	String	Template.playgroundpage
Version	String	1.0
Name	String	RecipeBook
▼ InitialUserPages	Array	(3 Items)
Item 0	String	Butter.playgroundpage
Item 1	String	Frosting.playgroundpage
Item 2	String	Cake.playgroundpage

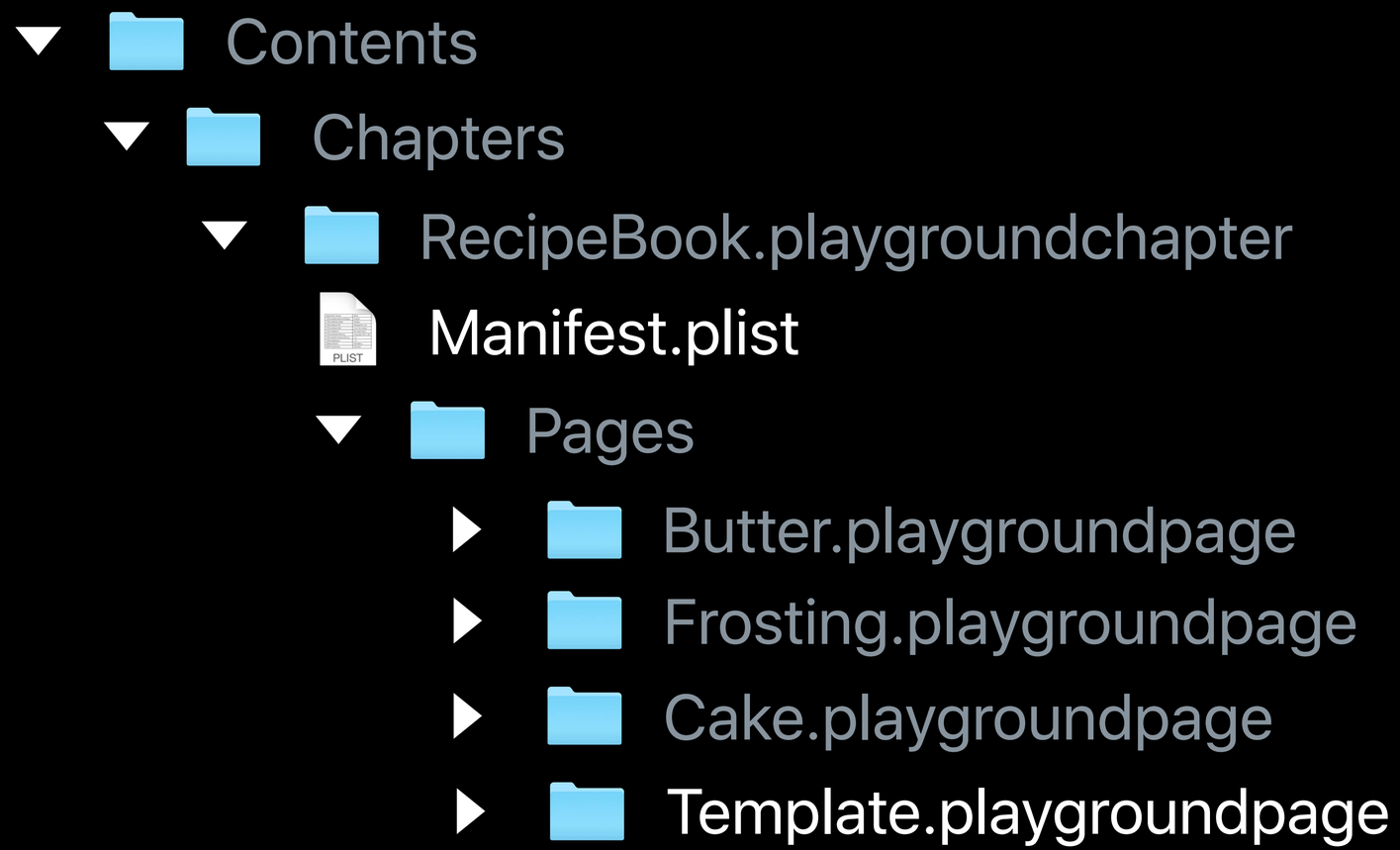
Specifying a Template Page

Manifest.plist

Key	Type	Value
▼ Root	Dictionary	
TemplatePageFilename	String	Template.playgroundpage
Version	String	1.0
Name	String	RecipeBook
▼ InitialUserPages	Array	(3 Items)
Item 0	String	Butter.playgroundpage
Item 1	String	Frosting.playgroundpage
Item 2	String	Cake.playgroundpage

Specifying a Template Page

File Structure



Localization

Najla Bulous, Playgrounds Engineer


iPad
9:41 AM
100%
Cancelar
Aprende a programar
Aprende a programar
Aprende a programar en serio de forma divertida
APRENDE A PROGRAMAR 1
Fundamentos de Swift
APRENDE A PROGRAMAR 2
Más allá de lo básico
APRENDE A PROGRAMAR 3
Explorando el universo
Aprende a progr...
Swift 3.1 Edition
OBTENER
Aprende a progr...
Swift 3.1 Edition
OBTENER
Aprende a progr...
Swift 3.1 Edition
OBTENER
Aprende a programar
Retos
Accesorios
Puntos de partida

Localization

DevelopmentRegion

Specifies the base language for the book

Playground book equivalent of `CFBundleDevelopmentRegion`

Localization

.strings files

Support localizing content via strings files

Localization

.strings files

Support localizing content via strings files

- ManifestPlist.strings

Localization

.strings files

Support localizing content via strings files

- ManifestPlist.strings
- Prose.strings

Localization

.strings files

Support localizing content via strings files

- ManifestPlist.strings
- Prose.strings
- EditableFields.strings

Localization

.strings files

Support localizing content via strings files

- ManifestPlist.strings
- Prose.strings
- EditableFields.strings
- QuickHelp.strings

Localization

.strings files

Support localizing content via strings files

- ManifestPlist.strings
- Prose.strings
- EditableFields.strings
- QuickHelp.strings

Must be in the PrivateResources directory

Localization

ManifestPlist.strings

Support localizing Manifest.plist content via a strings file

ManifestPlist.strings may contain:

- Name
- Subtitle

Localization

ManifestPlist.strings

Support localizing Manifest.plist content via a strings file

ManifestPlist.strings may contain:

- Name
- Subtitle

Manifest

Key	Type	Value
▼ Root	Dictionary	
LiveViewEdgeToEdge	Boolean	YES
LiveViewMode	String	VisibleByDefault
MaximumSupportedExecutionSpeed	String	Faster
Name	String	Issuing Commands
PlaygroundLoggingMode	String	Off
PosterReference	String	LiveViewPoster.png
Version	String	1.0

```
/* Localizable Manifest Content */

/* ManifestPlist.strings (zh_CN) */

/* Name of the "Issuing Commands" page. */
"Name" = "发出命令";
```

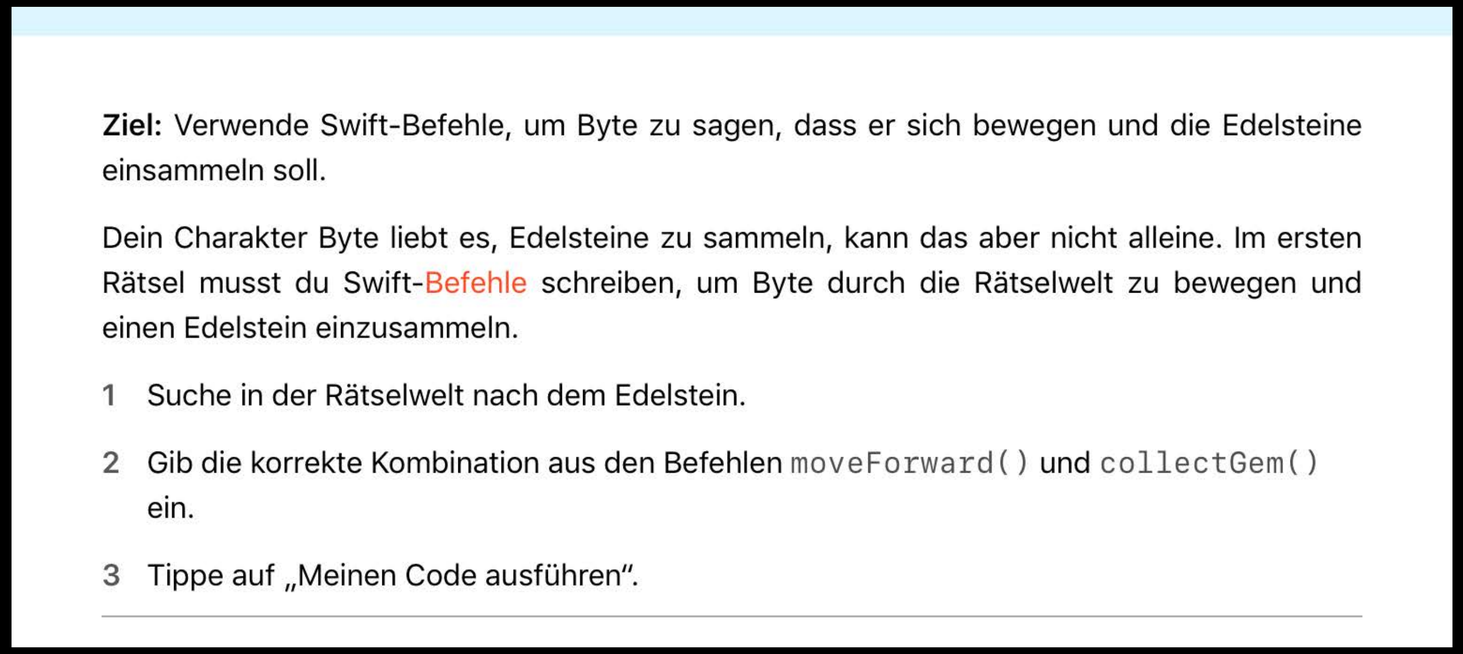
```
/* Localizable Manifest Content */

/* ManifestPlist.strings (zh_CN) */

/* Name of the "Issuing Commands" page. */
"Name" = "发出命令";
```

Localization

Prose.strings

Playground markup blocks may be localized

Markup content stored in Prose.strings at the page level



Ziel: Verwende Swift-Befehle, um Byte zu sagen, dass er sich bewegen und die Edelsteine einsammeln soll.

Dein Charakter Byte liebt es, Edelsteine zu sammeln, kann das aber nicht alleine. Im ersten Rätsel musst du Swift-Befehle schreiben, um Byte durch die Rätselwelt zu bewegen und einen Edelstein einzusammeln.

1. Suche in der Rätselwelt nach dem Edelstein.
2. Gib die korrekte Kombination aus den Befehlen `moveForward()` und `collectGem()` ein.
3. Tippe auf „Meinen Code ausführen“.

```
// Localizable Playground Markup

// Contents.swift

//:#localized(key: "com.apple.LearnToCode1.Commands.IssuingCommands")
//: **Goal:** Use Swift commands to tell Byte to move and collect a gem.
//:
//: Your character, Byte, loves to collect gems but can't do it alone.
//: In this first puzzle…

moveForward()
moveForward()
moveForward()
turnLeft()
```

```
// Localizable Playground Markup

// Contents.swift

//:#localized(key: "com.apple.LearnToCode1.Commands.IssuingCommands")
//: **Goal:** Use Swift commands to tell Byte to move and collect a gem.
//:
//: Your character, Byte, loves to collect gems but can't do it alone.
//: In this first puzzle…

moveForward()
moveForward()
moveForward()
turnLeft()
```

```
/* Localizable Playground Markup */

/* Prose.strings (de) */

/* Markup content for the main prose block on "Issuing Commands". */
"com.apple.LearnToCode1.Commands.IssuingCommands" = "**Ziel:** Verwende Swift-Befehle, um Byte zu sagen, dass er sich bewegen und die Edelsteine einsammeln soll.\n\nDein Charakter Byte liebt es, Edelsteine zu sammeln, kann das aber nicht alleine.\nIm ersten Rätsel…";
```

```
/* Localizable Playground Markup */

/* Prose.strings (de) */

/* Markup content for the main prose block on "Issuing Commands". */
"com.apple.LearnToCode1.Commands.IssuingCommands" = "**Ziel:** Verwende Swift-Befehle, um Byte zu sagen, dass er sich bewegen und die Edelsteine einsammeln soll.\n\nDein Charakter Byte liebt es, Edelsteine zu sammeln, kann das aber nicht alleine.\nIm ersten Rätsel…";
```

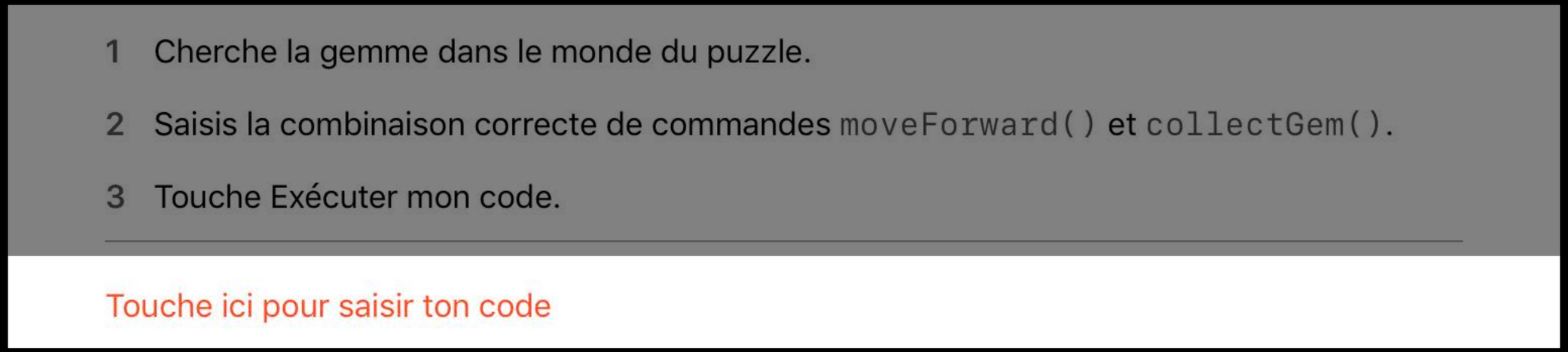
Localization

EditableFields.strings

Placeholders for editable fields may be localized

Localized strings stored in book-level EditableFields.strings file

- Keys are the placeholder values specified in editable field tags



```
// Localizable Editable Field Placeholders

// Contents.swift

//#-editable-code Tap to enter code
moveForward()
moveForward()
moveForward()
turnLeft()
//#-end-editable-code
```

```
// Localizable Editable Field Placeholders

// Contents.swift

//#-editable-code Tap to enter code
moveForward()
moveForward()
moveForward()
turnLeft()
//#-end-editable-code
```

```
/* Localizable Editable Field Placeholders */

/* EditableFields.strings (fr) */

/* Placeholder shown when an editable field is empty. */
"Tap to enter code" = "Touche ici pour saisir ton code";
```

```
/* Localizable Editable Field Placeholders */

/* EditableFields.strings (fr) */

/* Placeholder shown when an editable field is empty. */
"Tap to enter code" = "Touche ici pour saisir ton code";
```

Localization

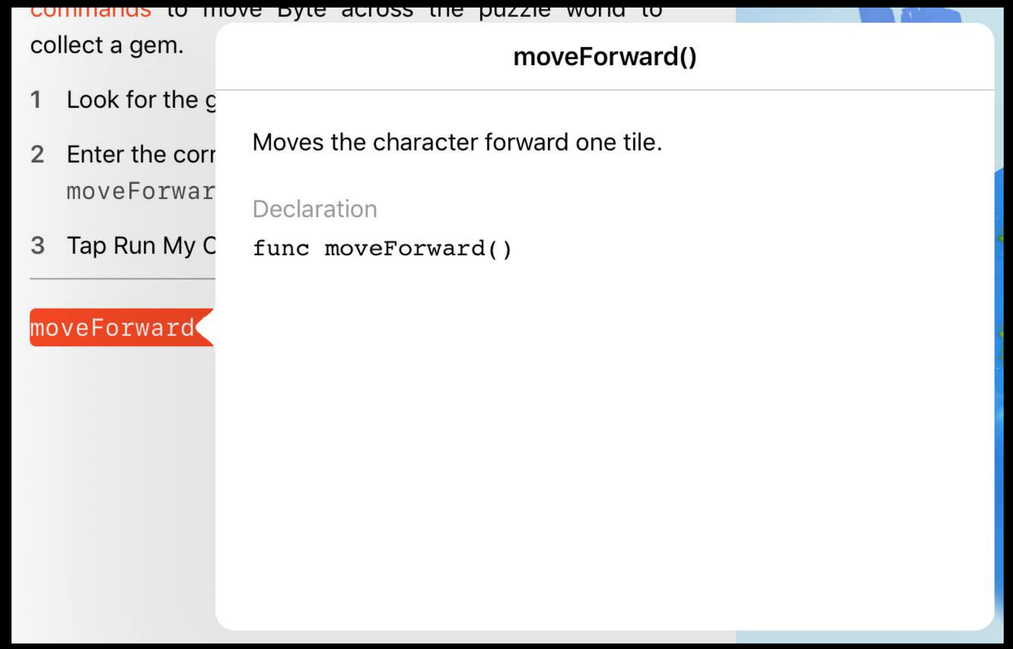
API Documentation

BETA

Supported in Swift Playgrounds 2.0 and later

Supports localizable API documentation

- Add `LocalizationKey` to API documentation comment



```
// Localizable API Documentation

// AuxiliarySources.swift

/// Moves the character forward one tile.
///
/// - LocalizationKey: com.apple.LearnToCode.moveForward
func moveForward() {
    // ...
}
```

BETA

```
// Localizable API Documentation

// AuxiliarySources.swift

/// Moves the character forward one tile.
///
/// - LocalizationKey: com.apple.LearnToCode.moveForward
func moveForward() {
    // ...
}
```

BETA

```
/* Localizable API Documentation */

/* QuickHelp.strings (en) */

/* Documentation for the moveForward() function. */
"com.apple.LearnToCode.moveForward" = "Moves the character forward one tile.";
```

BETA

```
/* Localizable API Documentation */

/* QuickHelp.strings (en) */

/* Documentation for the moveForward() function. */
"com.apple.LearnToCode.moveForward" = "Moves the character forward one tile.";
```

BETA

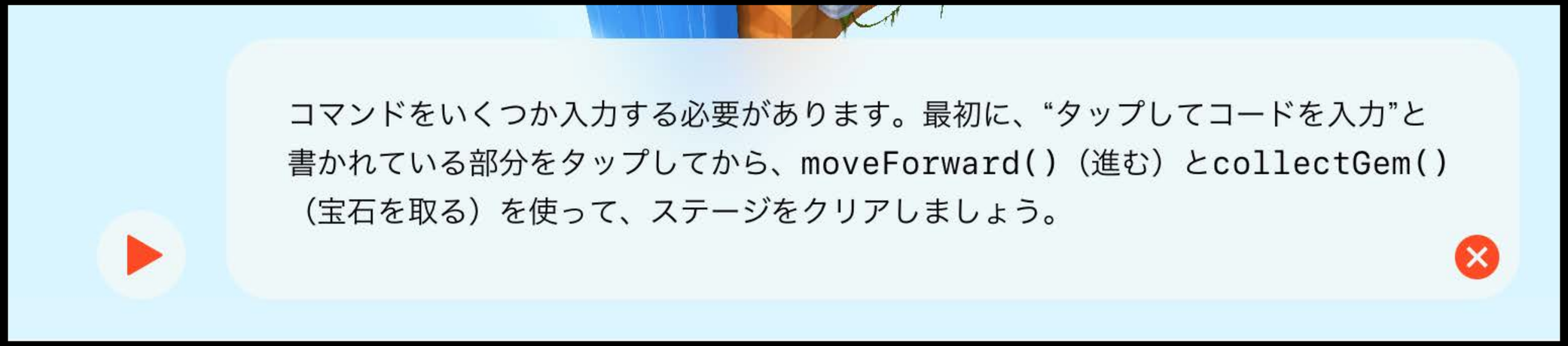
Localization

Default Hints

Default hints now specified in a separate Hints.plist

- Same format as default hints in Manifest.plist

For localization, Hints.plist should be stored in lproj directories



Localization

Default Hints

Key	Type	Value
▼ Root	Dictionary	
▼ Item 0	Dictionary	(1 item)
Content	String	コマンドをいくつか入力する必要があります。

Localization

Default Hints

Key	Type	Value
▼ Root	Dictionary	
▼ Item 0	Dictionary	(1 item)
Content	String	コマンドをいくつか入力する必要があります。

PlaygroundBluetooth API

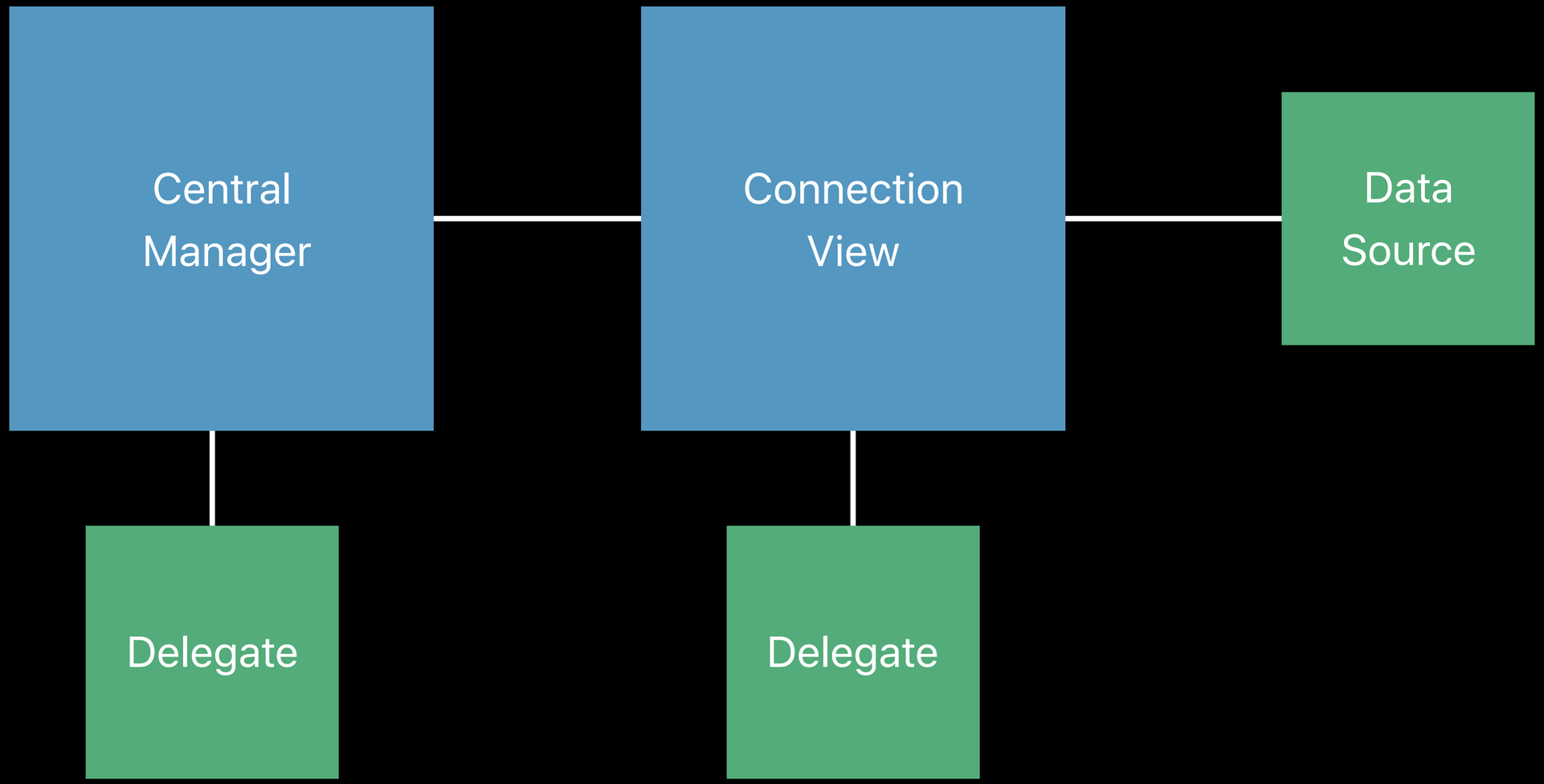
PlaygroundBluetooth

Swift Playgrounds provides access to CoreBluetooth

PlaygroundBluetooth framework aims to provide a consistent experience across playground books

PlaygroundBluetooth

Components



PlaygroundBluetooth

PlaygroundBluetoothCentralManager

Provides an interface for connecting to and interacting with accessories

Similar to CoreBluetooth's `CBCentralManager`

PlaygroundBluetooth

PlaygroundBluetoothCentralManagerDelegate

```
centralManagerStateDidChange(_:)

centralManager(_:didDiscover:withAdvertisementData:)

centralManager(_:willConnectTo:)

centralManager(_:didConnectTo:)

centralManager(_:didFailToConnectTo:error:)

centralManager(_:didDisconnectFrom:error:)
```

PlaygroundBluetooth

PlaygroundBluetoothConnectionView

Provides an interface for displaying connection status of accessories

Provides UI for users to discover, connect, and disconnect from accessories

PlaygroundBluetooth

PlaygroundBluetoothConnectionViewDelegate

```
connectionView(_:shouldDisplayDiscovered:
withAdvertisementData:rssi:)

connectionView(_:shouldConnectTo:
withAdvertisementData:)

connectionView(_:willDisconnectFrom:)

connectionView(_:titleFor:)

connectionView(_:firmwareUpdateInstructionFor:)
```

PlaygroundBluetooth

PlaygroundBluetoothConnectionViewDataSource

Protocol adopted by a PlaygroundBluetoothConnectionView object

Provides information it needs to display accessories

- Name
- Icon

PlaygroundBluetooth

PlaygroundBluetoothConnectionViewDataSource

Protocol adopted by a PlaygroundBluetoothConnectionView object

Provides information it needs to display accessories

- Name
- Icon


```
connectionView(_:itemForPeripheral:withAdvertisementData:)
```

Demo

PlaygroundBluetooth in Swift Playgrounds

Summary

Overview of changes since last September

Format changes such as public vs. private resources

Enhancements to playground book

- Copy code forward
- Code highlighting
- User editable books
- Localization

PlaygroundBluetooth

More Information

https://developer.apple.com/wwdc17/408

Related Sessions

Localizing Content for Swift Playgrounds	Grand Ballroom A	Thursday 3:10PM
Teaching with Swift Playgrounds	Hall 2	Friday 2:50PM
SceneKit in Swift Playgrounds		WWDC 2017
Localizing with Xcode 9		WWDC 2017
What’s New in Swift		WWDC 2017

Labs

Creating Content for Swift Playgrounds Lab	Technology Lab E	Thur 12:00PM–3:10PM
Swift Open Hours	Technology Lab D	Fri 12:00PM–1:30PM
Xcode Open Hours	Technology Lab K	Fri 1:50PM-4:00PM


WWDC17