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
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

Three kinds of Swift files
Swift Files

Three kinds of Swift files

• Contents.swift

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, you’ll need to write Swift commands to move Byte across the puzzle world to collect a gem.

1. Look for the gem in the puzzle world.
2. Enter the correct combination of the `moveForward()` and `collectGem()` commands.
3. Tap Run My Code.

```swift
moveForward()
moveForward()
moveForward()
moveForward()
collectGem()
```
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
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
Another way to add an action is by defining an array. You include actions that the new action beats within the [ ] 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.

```javascript
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);
```
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.

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

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

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

```python
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```
let array = ["first", "second", "third", "fourth", "fifth"]

array[5]  

Index out of range
// 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: 32.8, fontName: "Futura", color: .red)
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”。

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

恭喜！
你写出了第一行 Swift 代码。
Byte 执行了你写下的命令，并完全按照你的要求和指定的顺序进行了操作。

下一页
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。でも自己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。

```swift
moveForward()
moveForward()
moveForward()
collectGem()
```
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
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.

2. Gib die korrekte Kombination aus den Befehlen `moveForward()` und `collectGem()` ein.
3. Tippe auf „Meinen Code ausführen“.

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

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.
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. Ingresá la combinación correcta de comandos `moveForward()` (“avanzar”) y `collectGem()` (“recolectar gema”).
3. Toca “Ejecutar mi código”.

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

¡Felicidades!
Escribiste tus primeras líneas de código Swift.
Byte realizó los comandos que escribiste y hizo exactamente lo que pediste y en el orden en que lo especificaste.

Siguiente página
iOS 11 SDK
New frameworks added to the SDK in Swift Playgrounds

- ARKit
- CoreML
- IOSurface
- PDFKit
- Vision
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
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.

```swift
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.

```swift
func makeButter() -> Butter {
    let cream = Cream(quantity: .ounces(8))
    return cream.churn()
}
```
To get started, let’s walk through making something quick and tasty—butter. Butter has one ingredient and
isn’t too difficult to make! Let’s get this started.

```swift
// #-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
```
To get started, let's walk through making something quick and tasty - butter. Butter has one ingredient and we're going to churn it into butter.

```swift
func makeButter() -> Butter {
    // Tap to enter code
    let cream = Cream(quantity: .ounces(8))
    return cream.churn()
}
```

// Copy-source(butter)
// End-copy-source
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
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
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`.

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

func makeFrosting() -> Frosting {
    // Code goes here...
}
Woohoo! Now that we’ve mastered butter, let’s move on to something sweeter.

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

```swift
func makeFrosting() -> Frosting {
    // Your code here
}
```
Woohoo! Now that we’ve mastered butter, let’s move on to something sweeter.

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

return cream.churn()
## Copying Code Between Pages

### Required Manifest.plist Keys

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>▼ CodeCopySetup</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>ReadyToCopyInstructions</td>
<td>String</td>
<td>You can bring the <code>makeButter</code> function you wrote to this page to...</td>
</tr>
<tr>
<td>NotReadyToCopyInstructions</td>
<td>String</td>
<td>Go back to finish making butter?</td>
</tr>
</tbody>
</table>
## Copying Code Between Pages

### Optional Manifest.plist Keys

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>▼ CodeCopySetup</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>CopyCommandButtonTitle</td>
<td>String</td>
<td>Bring Over My Code</td>
</tr>
<tr>
<td>NavigateCommandButtonTitle</td>
<td>String</td>
<td>Return to Previous Page</td>
</tr>
<tr>
<td>DefaultCommandButtonTitle</td>
<td>String</td>
<td>Start Cooking on This Page</td>
</tr>
</tbody>
</table>
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
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()
}
```
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.

```python
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```
let string = "Hello, world"
for character in string {
    print(character)
}
// Step Through My Code

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

```plaintext
def for i in 1 ... 4:
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
turnRight()
```

Run My Code
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.

```python
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```
Run Faster and Run Fastest
Manifest.plist Key

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>MaximumSupportedExecutionSpeed</td>
<td>String</td>
<td>Fastest</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>3.0</td>
</tr>
<tr>
<td>DevelopmentRegion</td>
<td>String</td>
<td>en</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>RecipeBook</td>
</tr>
<tr>
<td>MinimumSwiftPlaygroundsVersion</td>
<td>String</td>
<td>1.5</td>
</tr>
<tr>
<td>DeploymentTarget</td>
<td>String</td>
<td>ios10.3</td>
</tr>
<tr>
<td>SwiftVersion</td>
<td>String</td>
<td>3.1</td>
</tr>
</tbody>
</table>
## Deployment Target and Swift Version

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Dictionary</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>3.0</td>
</tr>
<tr>
<td>DevelopmentRegion</td>
<td>String</td>
<td>en</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>RecipeBook</td>
</tr>
<tr>
<td>MinimumSwiftPlaygroundsVersion</td>
<td>String</td>
<td>1.5</td>
</tr>
<tr>
<td>DeploymentTarget</td>
<td>String</td>
<td>ios10.3</td>
</tr>
<tr>
<td>SwiftVersion</td>
<td>String</td>
<td>3.1</td>
</tr>
</tbody>
</table>
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

- Contents
- Chapters
- Manifest.plist
- PrivateResources
- PublicResources
- Sources
Specifying Runtime Issues

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

let batter = Cake.Batter(mixing: sugar, eggs, vanilla, oil, milk, flour, lettuce)

One of these ingredients doesn't seem right!

var hotCake = oven.bake(batter)
var cooledCake = hotCake.letCool()
```
Templates in Swift Playgrounds
Supporting user-editing in playground books
User Editable Books

- Butter
- Frosting
- Cake

Options: Delete, Duplicate
User Editable Books

Add

Swift Kitchen

- Butter
- Frosting
- Cake

Delete Duplicate
User Editable Books

Add

Remove

Swift Kitchen

Butter

Frosting

Cake

Delete

Duplicate
User Editable Books

Add
Remove
Rename

[Image of a popover with options: Add, Remove, Rename.

- butter
- frosting
- cake]
User Editable Books

Add
Remove
Rename
Duplicate

Swift Kitchen

- Butter
- Frosting
- Cake

Delete

Duplicate
User Editable Books

Add
Remove
Rename
Duplicate
Reorder
Demo
## Specifying a Template Page

### Manifest.plist

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>TemplatePageFilename</td>
<td>String</td>
<td>Template.playgroundpage</td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>1.0</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>RecipeBook</td>
</tr>
<tr>
<td>▼ InitialUserPages</td>
<td>Array</td>
<td>(3 Items)</td>
</tr>
<tr>
<td>Item 0</td>
<td>String</td>
<td>Butter.playgroundpage</td>
</tr>
<tr>
<td>Item 1</td>
<td>String</td>
<td>Frosting.playgroundpage</td>
</tr>
<tr>
<td>Item 2</td>
<td>String</td>
<td>Cake.playgroundpage</td>
</tr>
</tbody>
</table>
## Specifying a Template Page

### Manifest.plist

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>TemplatePageFilename</td>
<td>String</td>
<td>Template.playgroundpage</td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>1.0</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>RecipeBook</td>
</tr>
<tr>
<td>InitialUserPages</td>
<td>Array</td>
<td>(3 Items)</td>
</tr>
<tr>
<td>Item 0</td>
<td>String</td>
<td>Butter.playgroundpage</td>
</tr>
<tr>
<td>Item 1</td>
<td>String</td>
<td>Frosting.playgroundpage</td>
</tr>
<tr>
<td>Item 2</td>
<td>String</td>
<td>Cake.playgroundpage</td>
</tr>
</tbody>
</table>
Specifying a Template Page

File Structure

- Contents
  - Chapters
    - RecipeBook.playgroundchapter
      - Manifest.plist
  - Pages
    - Butter.playgroundpage
    - Frosting.playgroundpage
    - Cake.playgroundpage
    - Template.playgroundpage
Localization

Najla Bulous, Playgrounds Engineer
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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>LiveViewEdgeToEdge</td>
<td>Boolean</td>
<td>YES</td>
</tr>
<tr>
<td>LiveViewMode</td>
<td>String</td>
<td>VisibleByDefault</td>
</tr>
<tr>
<td>MaximumSupportedExecutionSpeed</td>
<td>String</td>
<td>Faster</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Issuing Commands</td>
</tr>
<tr>
<td>PlaygroundLoggingMode</td>
<td>String</td>
<td>Off</td>
</tr>
<tr>
<td>PosterReference</td>
<td>String</td>
<td>LiveViewPoster.png</td>
</tr>
<tr>
<td>Version</td>
<td>String</td>
<td>1.0</td>
</tr>
</tbody>
</table>
/* 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.


2. Gib die korrekte Kombination aus den Befehlen `moveForward()` und `collectGem()` ein.

3. Tippe auf „Meinen Code ausführen“. 
// Localizable Playground Markup

// Contents.swift

//:**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()
movestowardsForward()
movestowardsForward()
movestowardsForward()
turnLeft()
**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...

```swift
moveForward()
moveForward()
moveForward()
moveForward()
turnLeft()
```
**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…
**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...
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

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

Touche ici pour saisir ton code
// Localizable Editable Field Placeholders

// Contents.swift

// #-editable-code Tap to enter code
moveForward()
moveForward()
moveForward()
moveForward()
turnLeft()
// #-end-editable-code
// Localizable Editable Field Placeholders

// Contents.swift

// #editable-code Tap to enter code
moveForward()
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";
"Tap to enter code" = "Touche ici pour saisir ton code";
Localization
API Documentation

Supported in Swift Playgrounds 2.0 and later
Supports localizable API documentation
• Add LocalizationKey to API documentation comment
/// Moves the character forward one tile.
///
/// - LocalizationKey: com.apple.LearnToCode.moveForward

func moveForward() {
    // ...
}
/// Localizable API Documentation

/// AuxiliarySources.swift

/// Moves the character forward one tile.
///
/// LocalizationKey: com.apple.LearnToCode.moveForward

func moveForward() {
    // ...
}

/* Localizable API Documentation */

/* QuickHelp.strings (en) */

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

/* QuickHelp.strings (en) */

/* Documentation for the moveForward() function. */

"com.apple.LearnToCode.moveForward" = "Moves the character forward one tile.";
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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>▼ Item 0</td>
<td>Dictionary</td>
<td>(1 item)</td>
</tr>
<tr>
<td>Content</td>
<td>String</td>
<td>コマンドをいくつか入力する必要があります。</td>
</tr>
</tbody>
</table>
## Localization
### Default Hints

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ Root</td>
<td>Dictionary</td>
<td></td>
</tr>
<tr>
<td>▼ Item 0</td>
<td>Dictionary</td>
<td>(1 item)</td>
</tr>
<tr>
<td>Content</td>
<td>String</td>
<td>コマンドをいくつか入力する必要があります。</td>
</tr>
</tbody>
</table>

### Localizing Content for Swift Playgrounds

Grand Ballroom A  
Thursday 3:10PM
PlaygroundBluetooth API
PlaygroundBluetooth

Swift Playgrounds provides access to CoreBluetooth

PlaygroundBluetooth framework aims to provide a consistent experience across playground books.
PlaygroundBluetooth

Components

Central Manager

Connection View

Data Source

Delegate

Delegate
PlaygroundBluetooth
PlaygroundBluetoothCentralManager

Provides an interface for connecting to and interacting with accessories

Similar to CoreBluetooth’s CBCentralManager
centralManagerStateChanged(_:)

centralManager(_:didDiscover:withAdvertisementData:)

centralManager(_:willConnectTo:)

centralManager(_:didConnectTo:)

centralManager(_:didFailToConnectTo:error:)

centralManager(_:didDisconnectFrom:error:)

PlaygroundBluetooth

PlaygroundBluetoothCentralManagerDelegate
PlaygroundBluetooth
PlaygroundBluetoothConnectionView

Provides an interface for displaying connection status of accessories

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

collectionView(_:shouldDisplayDiscovered: withAdvertisementData:rssi:)
collectionView(_:shouldConnectTo: withAdvertisementData:)
collectionView(_:willDisconnectFrom:)
collectionView(_:titleFor:)
collectionView(_: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
<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localizing Content for Swift Playgrounds</td>
<td>Grand Ballroom A</td>
<td>Thursday 3:10PM</td>
</tr>
<tr>
<td>Teaching with Swift Playgrounds</td>
<td>Hall 2</td>
<td>Friday 2:50PM</td>
</tr>
<tr>
<td>SceneKit in Swift Playgrounds</td>
<td></td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Localizing with Xcode 9</td>
<td></td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>What’s New in Swift</td>
<td></td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Lab</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Creating Content for Swift Playgrounds Lab</td>
<td>Technology Lab E</td>
<td>Thur 12:00PM–3:10PM</td>
</tr>
<tr>
<td>Swift Open Hours</td>
<td>Technology Lab D</td>
<td>Fri 12:00PM–1:30PM</td>
</tr>
<tr>
<td>Xcode Open Hours</td>
<td>Technology Lab K</td>
<td>Fri 1:50PM–4:00PM</td>
</tr>
</tbody>
</table>