Tools

#WWDC14

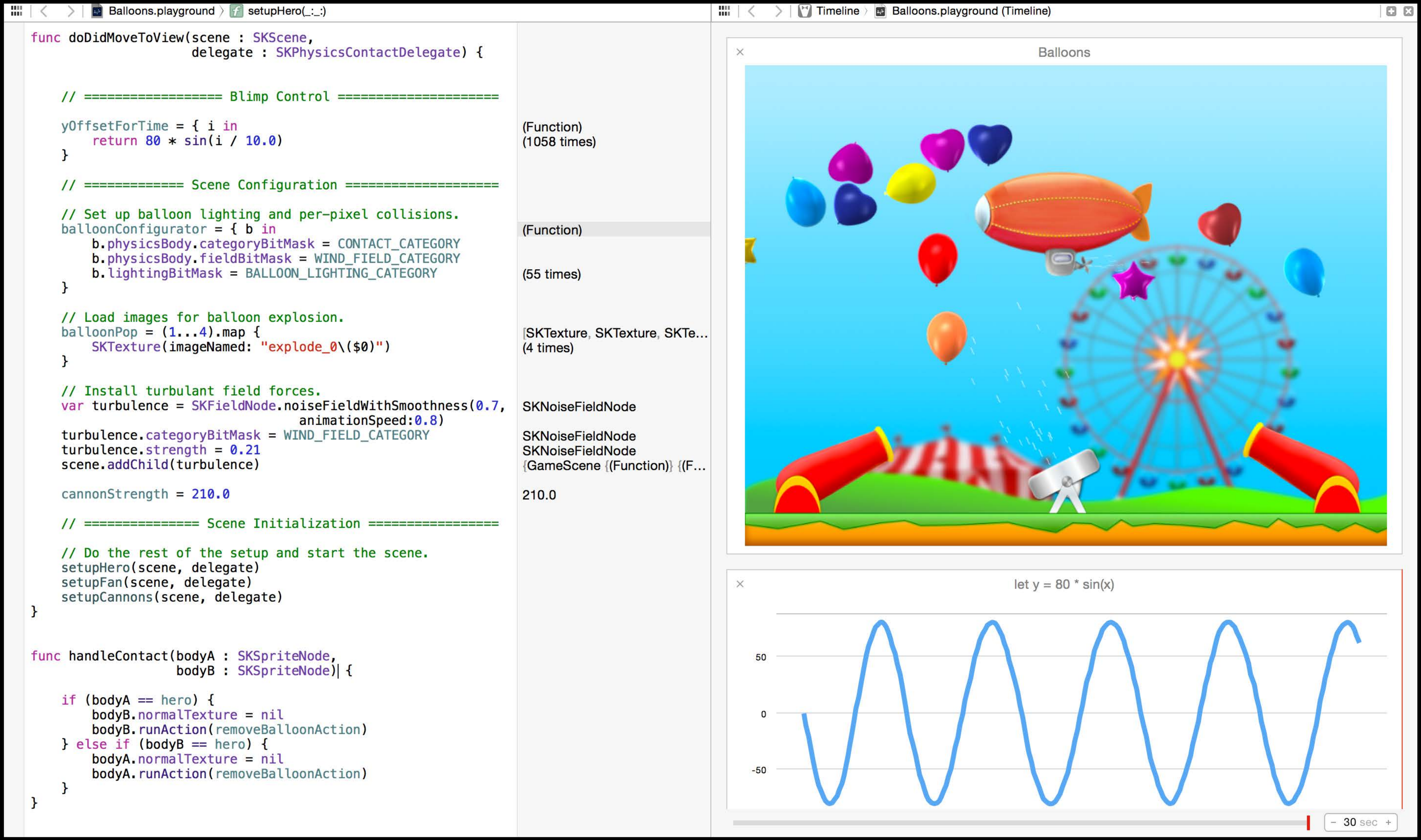
Swift Playgrounds

Session 408

Rick Ballard
Xcode Engineer

Connor Wakamo
Xcode Engineer

Introducing Playgrounds


Balloons.playground 〉 setupHero(_:_:)
Timeline 〉 Balloons.playground (Timeline)
Balloons
let y = 80 * sin(x)
50
0
-50
30 sec

```
func doDidMoveToView(scene : SKScene,
                     delegate : SKPhysicsContactDelegate) {


    // ================== Blimp Control =====================

    yOffsetForTime = { i in
        return 80 * sin(i / 10.0)
    }

    // ============= Scene Configuration ====================

    // Set up balloon lighting and per-pixel collisions.
    balloonConfigurator = { b in
        b.physicsBody.categoryBitMask = CONTACT_CATEGORY
        b.physicsBody.fieldBitMask = WIND_FIELD_CATEGORY
        b.lightingBitMask = BALLOON_LIGHTING_CATEGORY
    }

    // Load images for balloon explosion.
    balloonPop = (1...4).map {
        SKTexture(imageNamed: "explode_0\($0)")
    }

    // Install turbulant field forces.
    var turbulence = SKFieldNode.noiseFieldWithSmoothness(0.7,
                                              animationSpeed:0.8)
    turbulence.categoryBitMask = WIND_FIELD_CATEGORY
    turbulence.strength = 0.21
    scene.addChild(turbulence)

    cannonStrength = 210.0

    // =============== Scene Initialization =================

    // Do the rest of the setup and start the scene.
    setupHero(scene, delegate)
    setupFan(scene, delegate)
    setupCannons(scene, delegate)
}


func handleContact(bodyA : SKSpriteNode,
                   bodyB : SKSpriteNode) {

    if (bodyA == hero) {
        bodyB.normalTexture = nil
        bodyB.runAction(removeBalloonAction)
    } else if (bodyB == hero) {
        bodyA.normalTexture = nil
        bodyA.runAction(removeBalloonAction)
    }
}
```

```
(Function)
(1058 times)

(Function)

(55 times)

[SKTexture, SKTexture, SKTe...
(4 times)

SKNoiseFieldNode

SKNoiseFieldNode
SKNoiseFieldNode
{GameScene {(Function)} {(F...

210.0
```

What You Will Learn

What You Will Learn

Conceptual background

What You Will Learn

Conceptual background

Learning, exploration, and visualization

What You Will Learn

Conceptual background

Learning, exploration, and visualization

Resources

What You Will Learn

Conceptual background

Learning, exploration, and visualization

Resources

Algorithm development

What You Will Learn

Conceptual background

Learning, exploration, and visualization

Resources

Algorithm development

XCPlayground

What You Will Learn

Conceptual background

Learning, exploration, and visualization

Resources

Algorithm development

XCPlayground

Custom Quick Looks

What You Will Learn

Conceptual background

Learning, exploration, and visualization

Resources

Algorithm development

XCPlayground

Custom Quick Looks

Custom view development

What You Will Learn

Conceptual background

Learning, exploration, and visualization

Resources

Algorithm development

XCPlayground

Custom Quick Looks

Custom view development

Asynchronous code

What You Will Learn

Conceptual background

Learning, exploration, and visualization

Resources

Algorithm development

XCPlayground

Custom Quick Looks

Custom view development

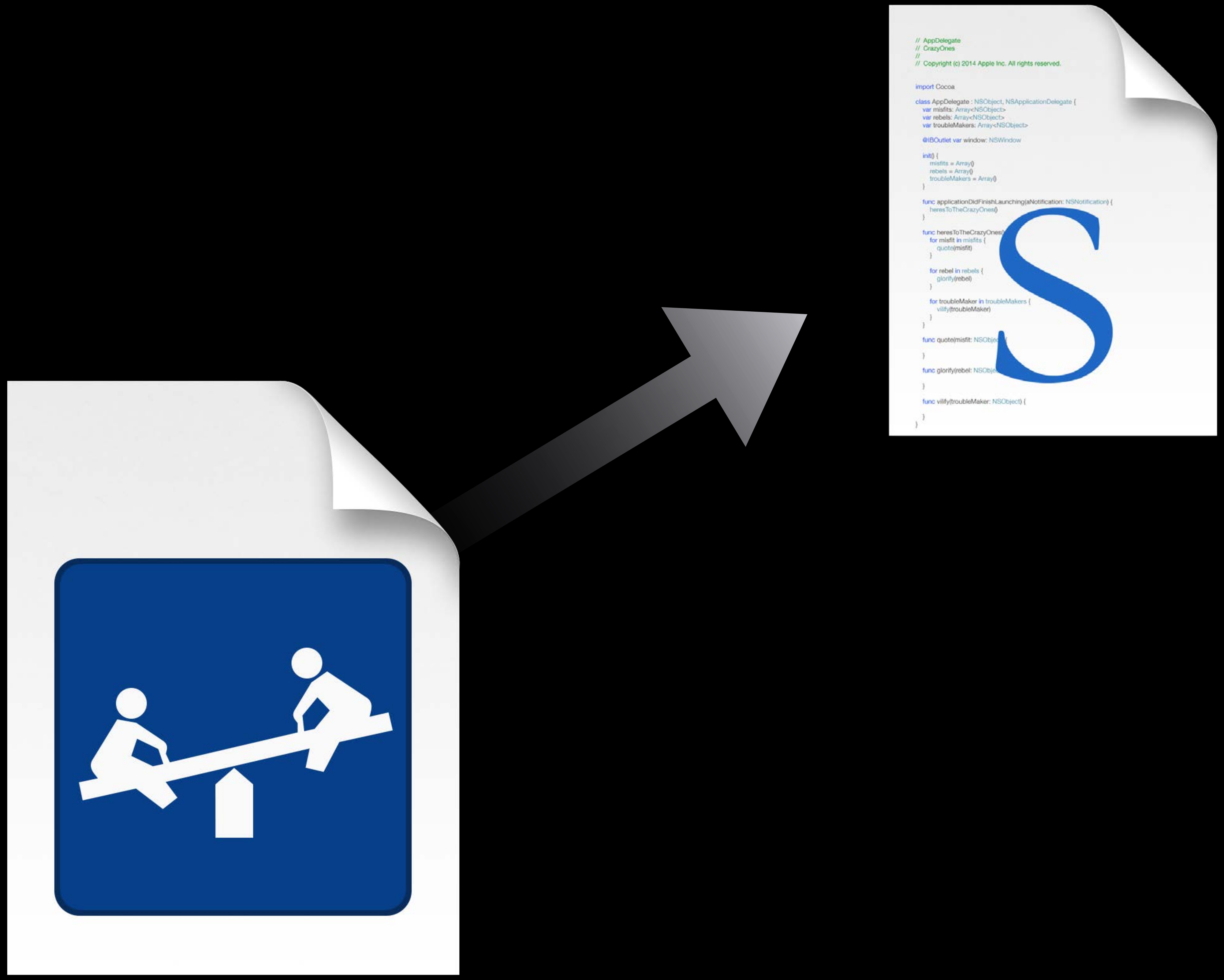
Asynchronous code

Limitations

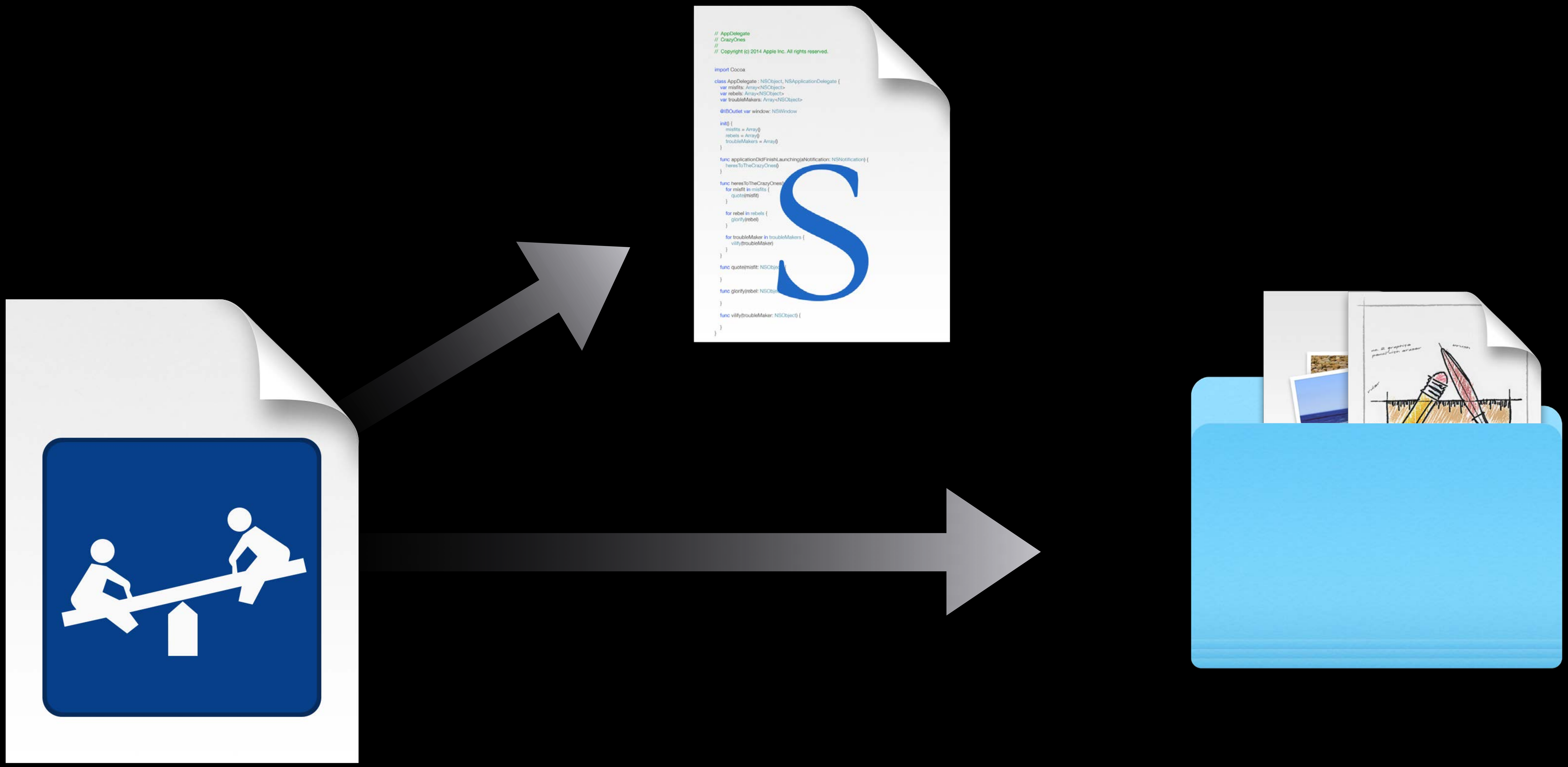
What Are Playgrounds?

What Are Playgrounds?

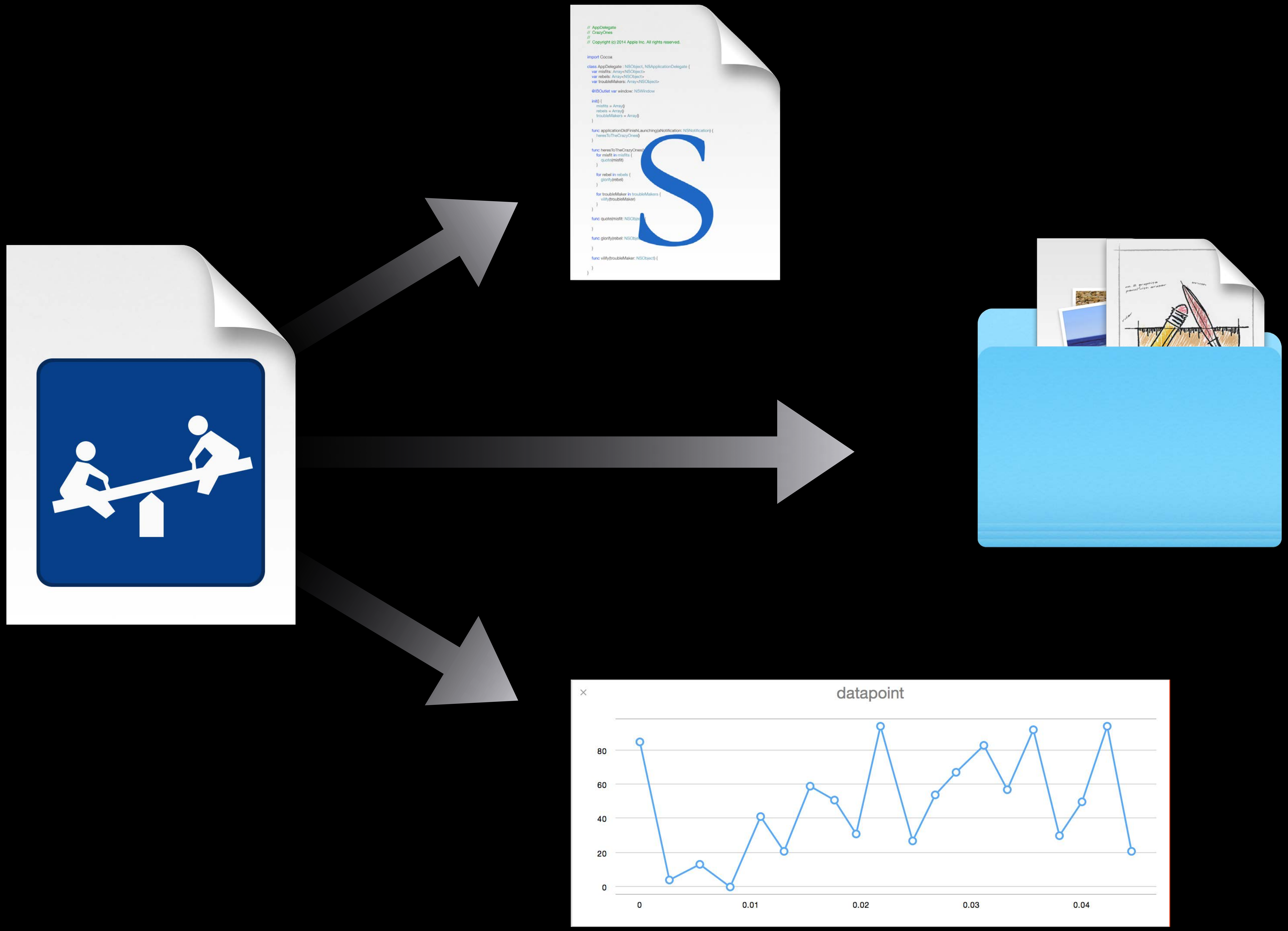
What Are Playgrounds?



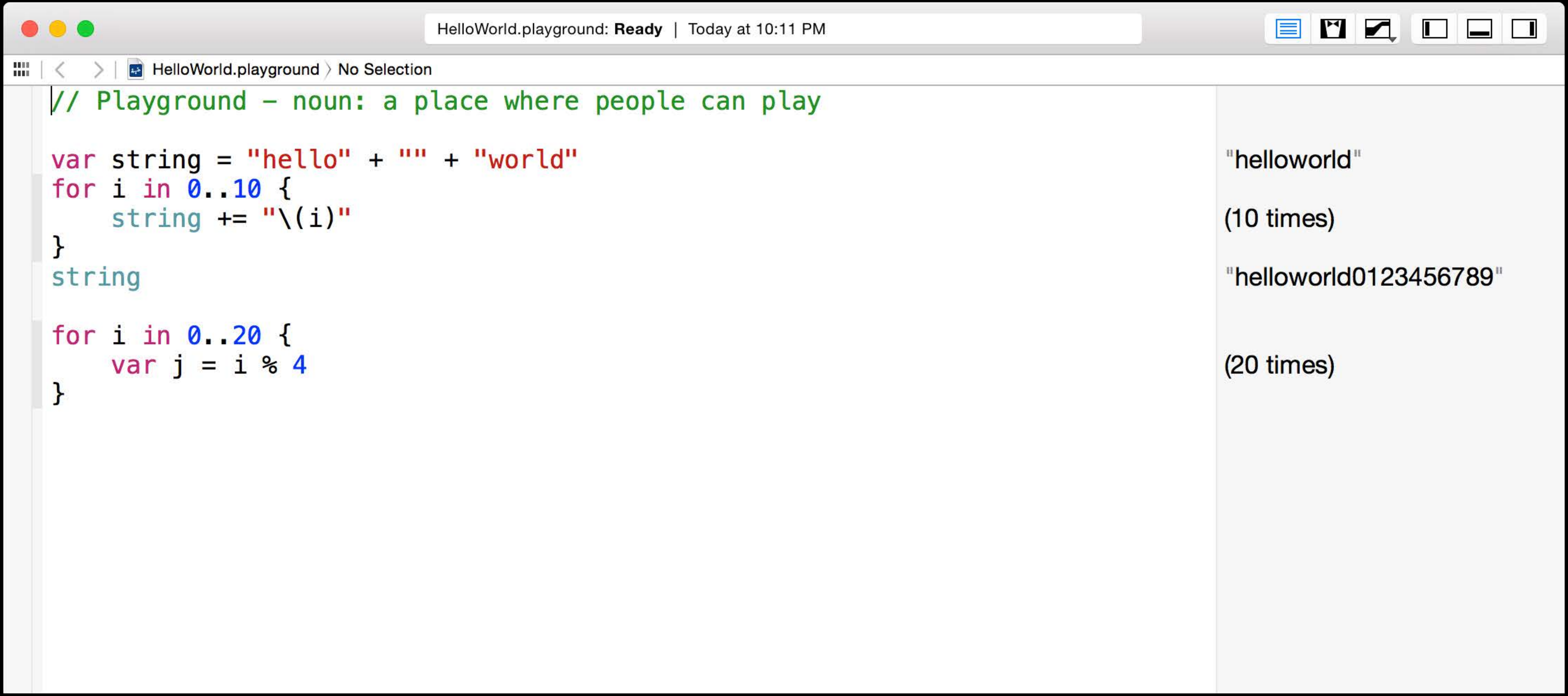
What Are Playgrounds?



What Are Playgrounds?



What Are Playgrounds?



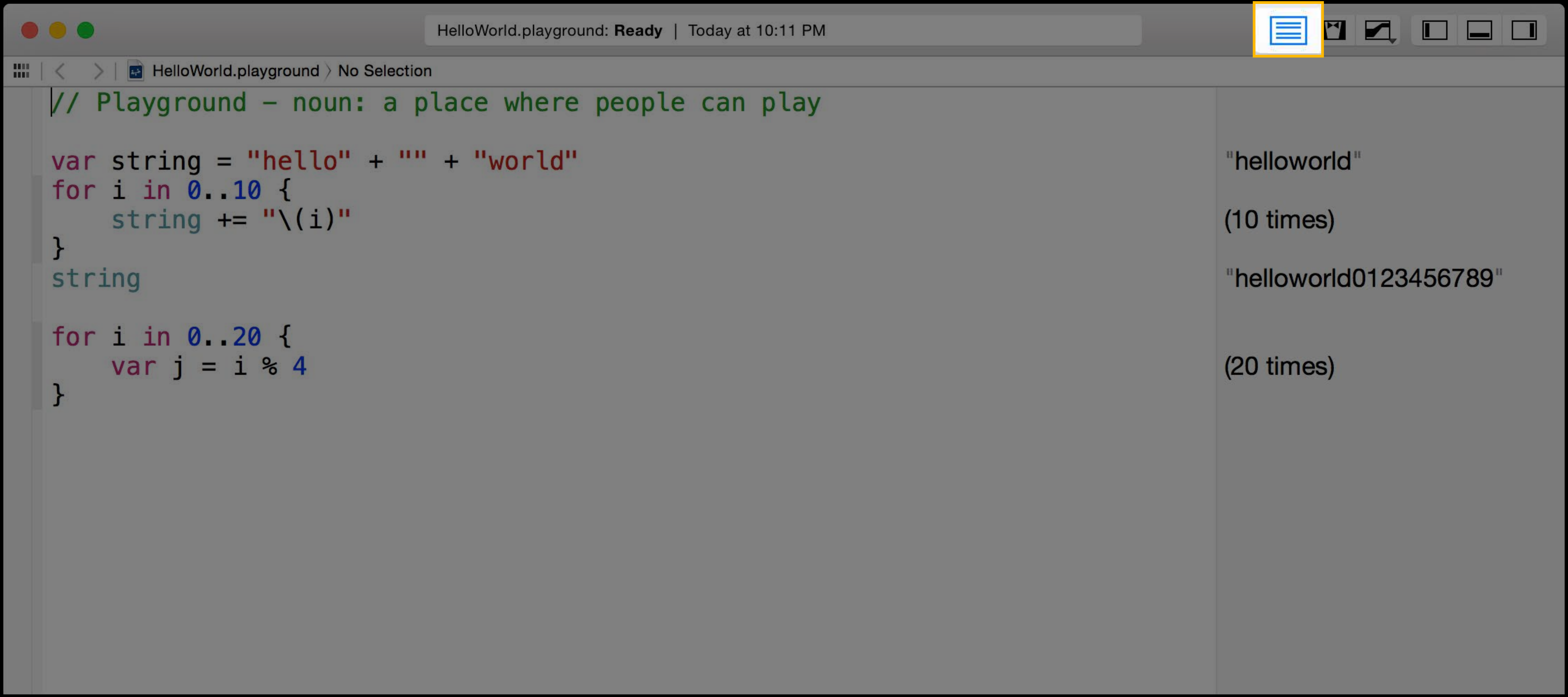
HelloWorld.playground: **Ready** | Today at 10:11 PM

HelloWorld.playground 〉 No Selection

```
// Playground - noun: a place where people can play

var string = "hello" + "" + "world"
for i in 0..10 {
    string += "\(i)"
}
string

for i in 0..20 {
    var j = i % 4
}
```

```
"helloworld"

(10 times)

"helloworld0123456789"


(20 times)
```

The Timeline

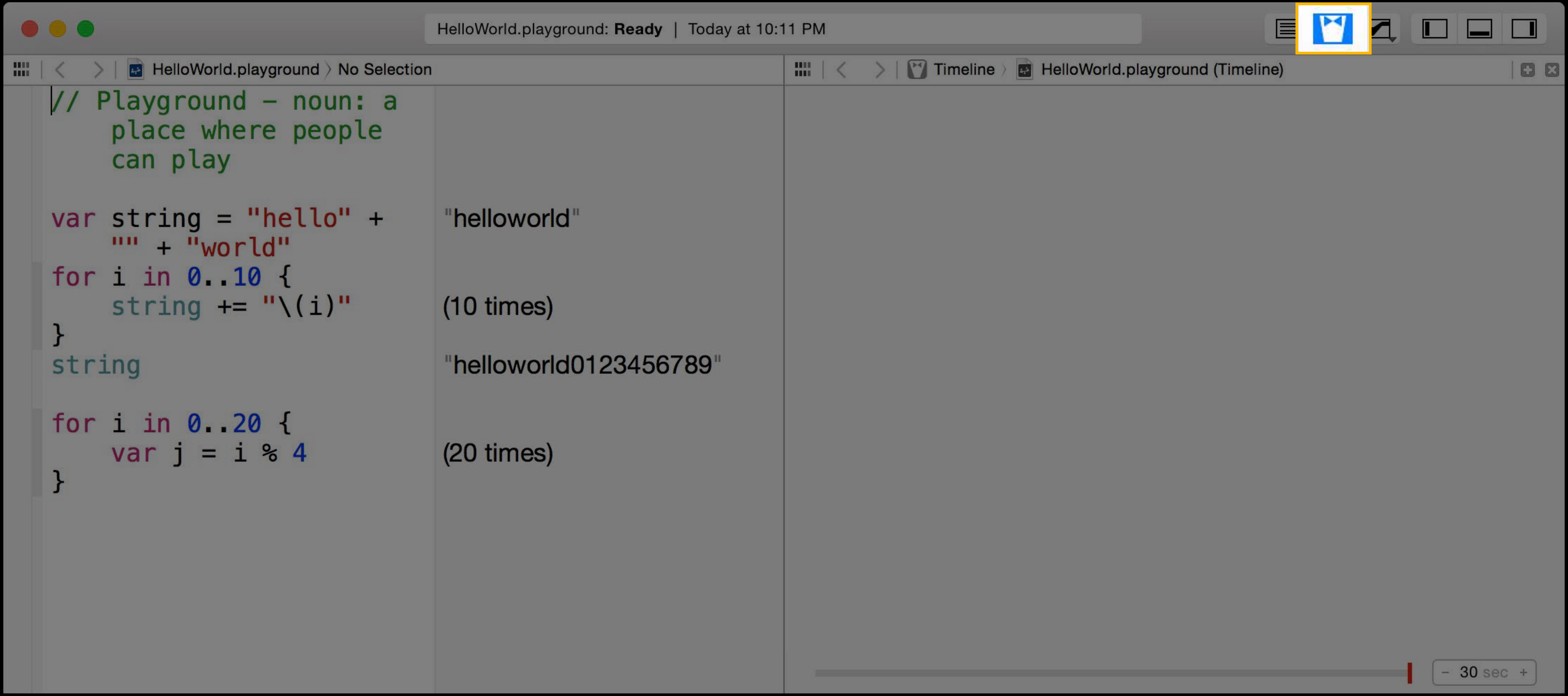


HelloWorld.playground: Ready | Today at 10:11 PM
HelloWorld.playground 〉 No Selection

```
// Playground - noun: a place where people can play

var string = "hello" + "" + "world"
for i in 0..10 {
    string += "\(i)"
}
string

for i in 0..20 {
    var j = i % 4
}
```

```
"helloworld"

(10 times)

"helloworld0123456789"


(20 times)
```

The Timeline

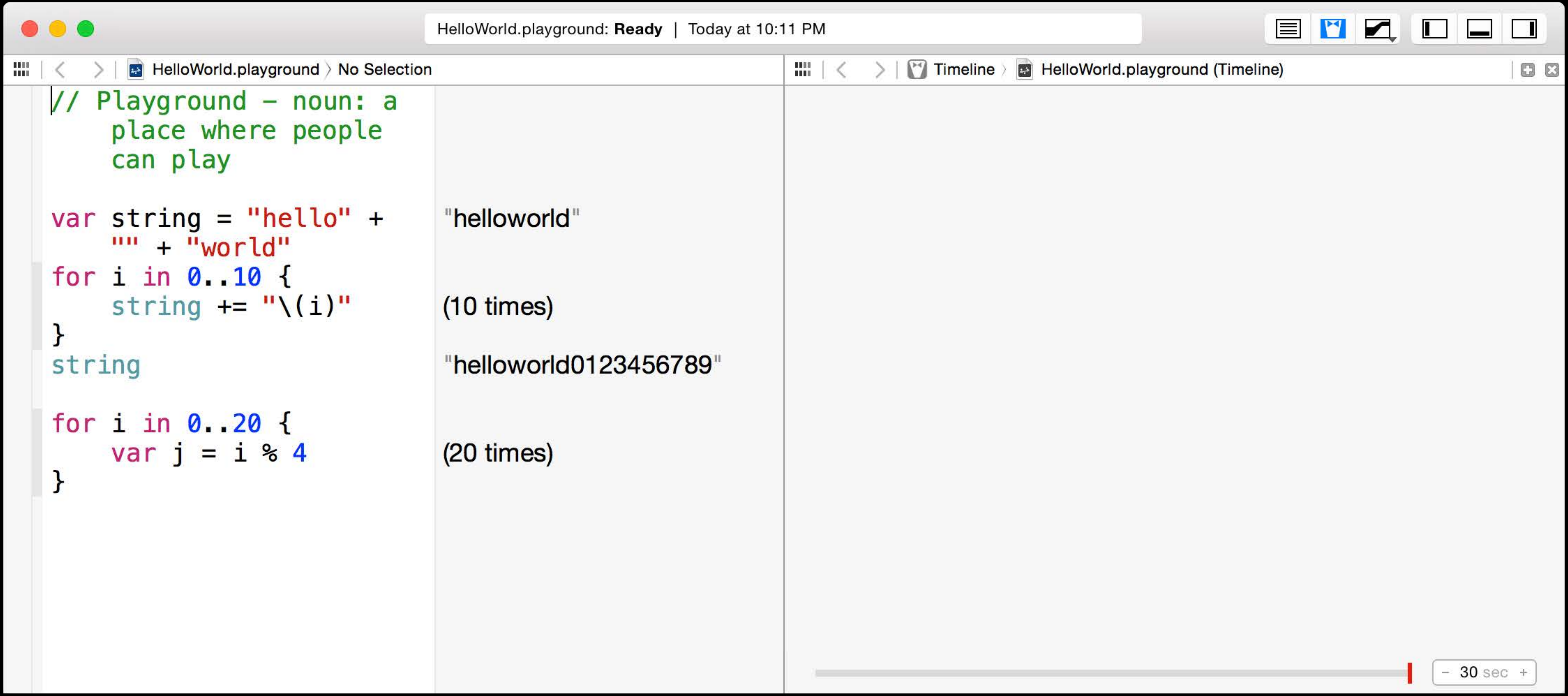


HelloWorld.playground: Ready | Today at 10:11 PM
HelloWorld.playground 〉 No Selection
Timeline 〉 HelloWorld.playground (Timeline)
30 sec

```
// Playground - noun: a
    place where people
    can play

var string = "hello" +
    "" + "world"
for i in 0..10 {
    string += "\(i)"
}
string

for i in 0..20 {
    var j = i % 4
}
```

```
"helloworld"

(10 times)

"helloworld0123456789"

(20 times)
```

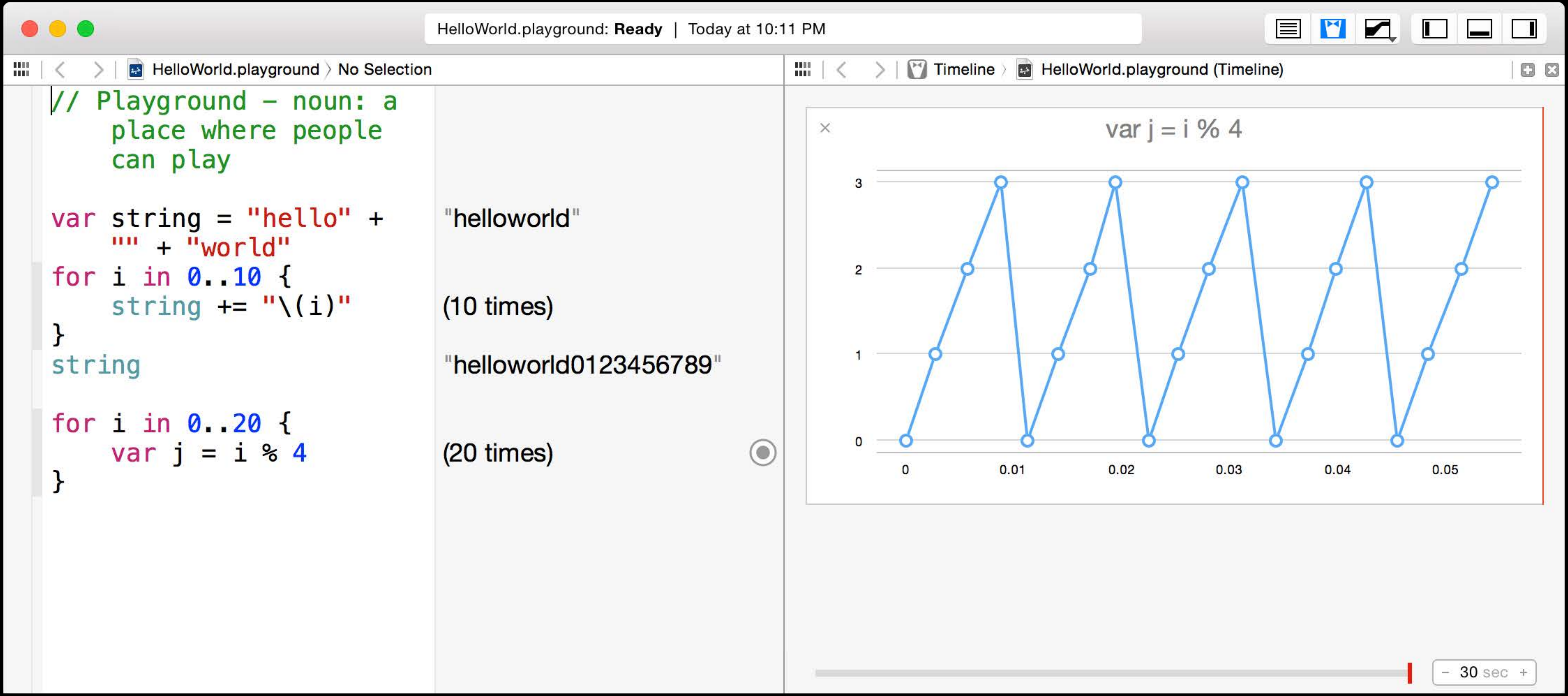
The Timeline



HelloWorld.playground: **Ready** | Today at 10:11 PM
HelloWorld.playground 〉 No Selection
Timeline 〉 HelloWorld.playground (Timeline)
− 30 sec +

```
// Playground - noun: a
    place where people
    can play

var string = "hello" +                 "helloworld"
    "" + "world"
for i in 0..10 {
    string += "\(i)"                   (10 times)
}
string                                 "helloworld0123456789"

for i in 0..20 {
    var j = i % 4                      (20 times)
}
```

The Timeline



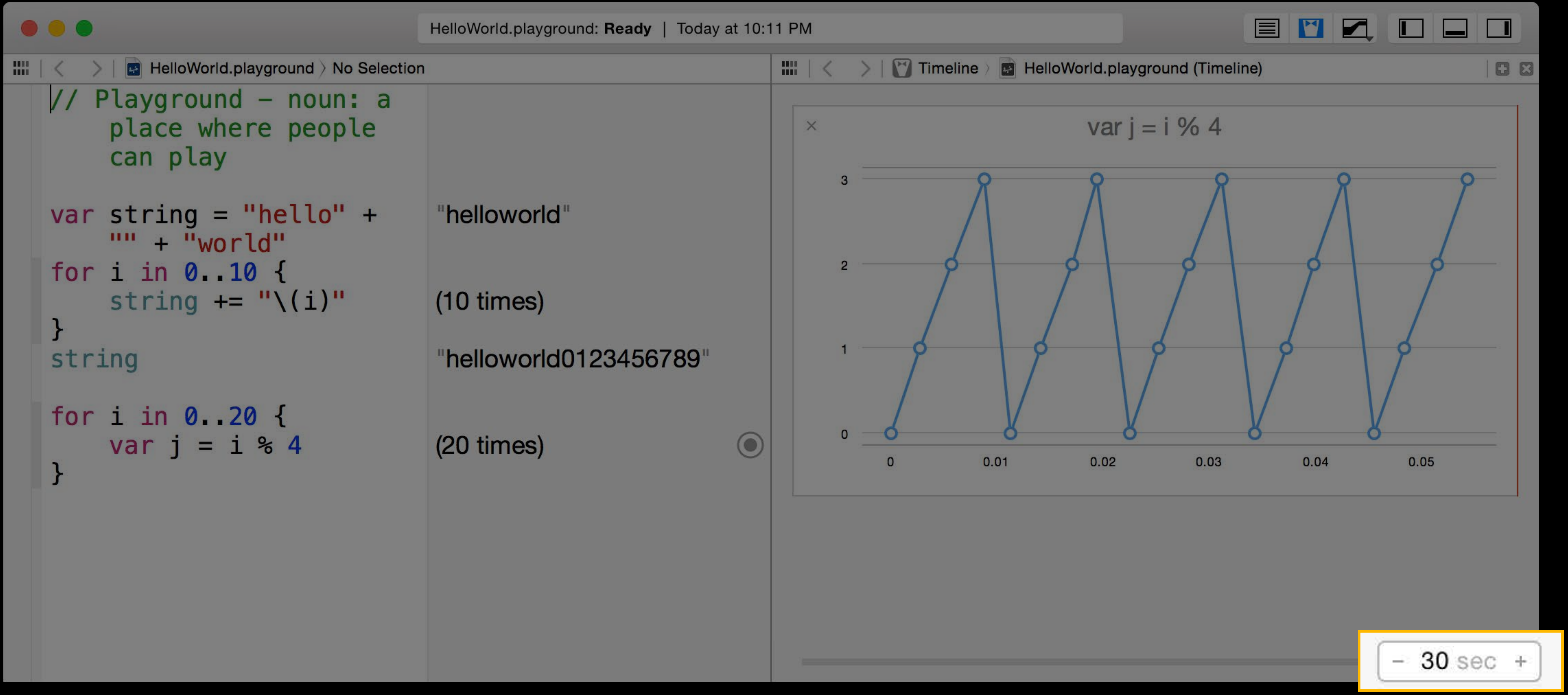
HelloWorld.playground: Ready | Today at 10:11 PM
HelloWorld.playground 〉 No Selection
Timeline 〉 HelloWorld.playground (Timeline)

```
// Playground - noun: a
    place where people
    can play

var string = "hello" +
    "" + "world"
for i in 0..10 {
    string += "\(i)"
}
string

for i in 0..20 {
    var j = i % 4
}
```

```
"helloworld"
(10 times)
"helloworld0123456789"
(20 times)
```

var j = i % 4
3
2
1
0
0
0.01
0.02
0.03
0.04
0.05
− 30 sec +

The Timeline



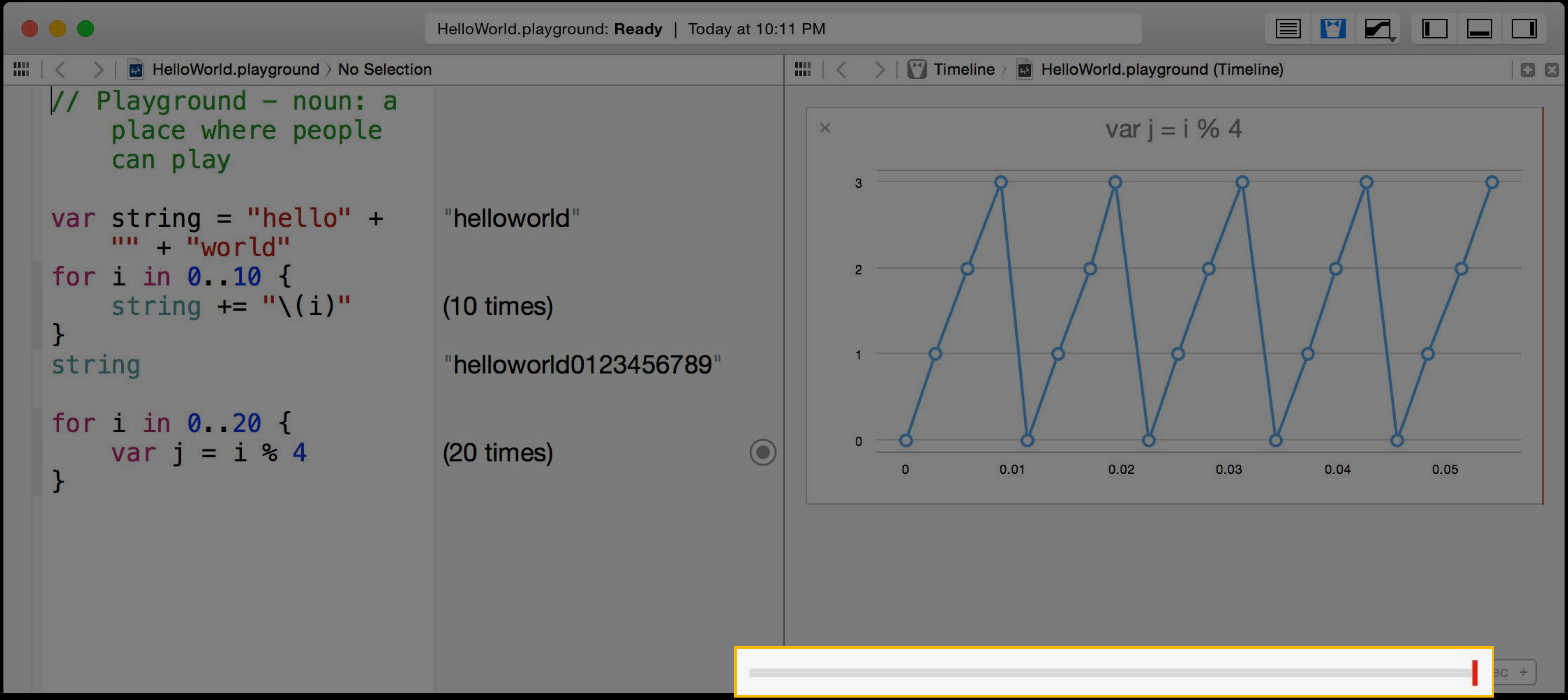
HelloWorld.playground: **Ready** | Today at 10:11 PM

HelloWorld.playground 〉 No Selection

```
// Playground - noun: a
    place where people
    can play

var string = "hello" +
    "" + "world"                  "helloworld"
for i in 0..10 {
    string += "\(i)"              (10 times)
}
string                            "helloworld0123456789"

for i in 0..20 {
    var j = i % 4                 (20 times)
}
```

Timeline 〉 HelloWorld.playground (Timeline)

The Timeline



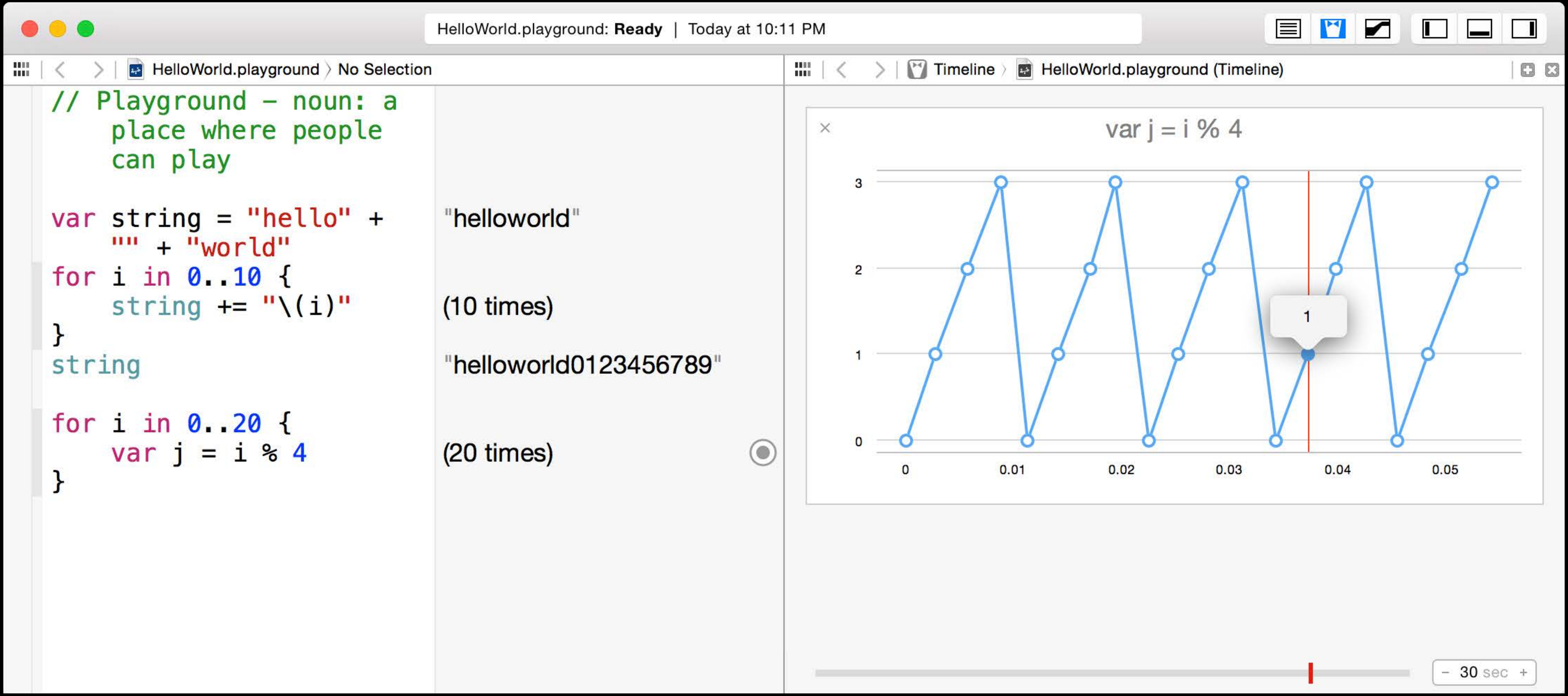
HelloWorld.playground: **Ready** | Today at 10:11 PM

HelloWorld.playground 〉 No Selection
Timeline 〉 HelloWorld.playground (Timeline)

```
// Playground - noun: a
    place where people
    can play

var string = "hello" +
    "" + "world"                "helloworld"
for i in 0..10 {
    string += "\(i)"            (10 times)
}
string                          "helloworld0123456789"

for i in 0..20 {
    var j = i % 4               (20 times)
}
```

var j = i % 4
3
2
1
0
0
0.01
0.02
0.03
0.04
0.05

The Timeline



HelloWorld.playground: **Ready** | Today at 10:11 PM
HelloWorld.playground 〉 No Selection
Timeline 〉 HelloWorld.playground (Timeline)
×
var j = i % 4
3
2
1
0
0
0.01
0.02
0.03
0.04
0.05
1
− 30 sec +

```
// Playground - noun: a
    place where people
    can play

var string = "hello" +          "helloworld"
    "" + "world"
for i in 0..10 {
    string += "\(i)"            (10 times)
}
string                          "helloworld0123456789"

for i in 0..20 {
    var j = i % 4               (20 times)
}
```

Why Use Playgrounds

Why Use Playgrounds

Learning

Why Use Playgrounds

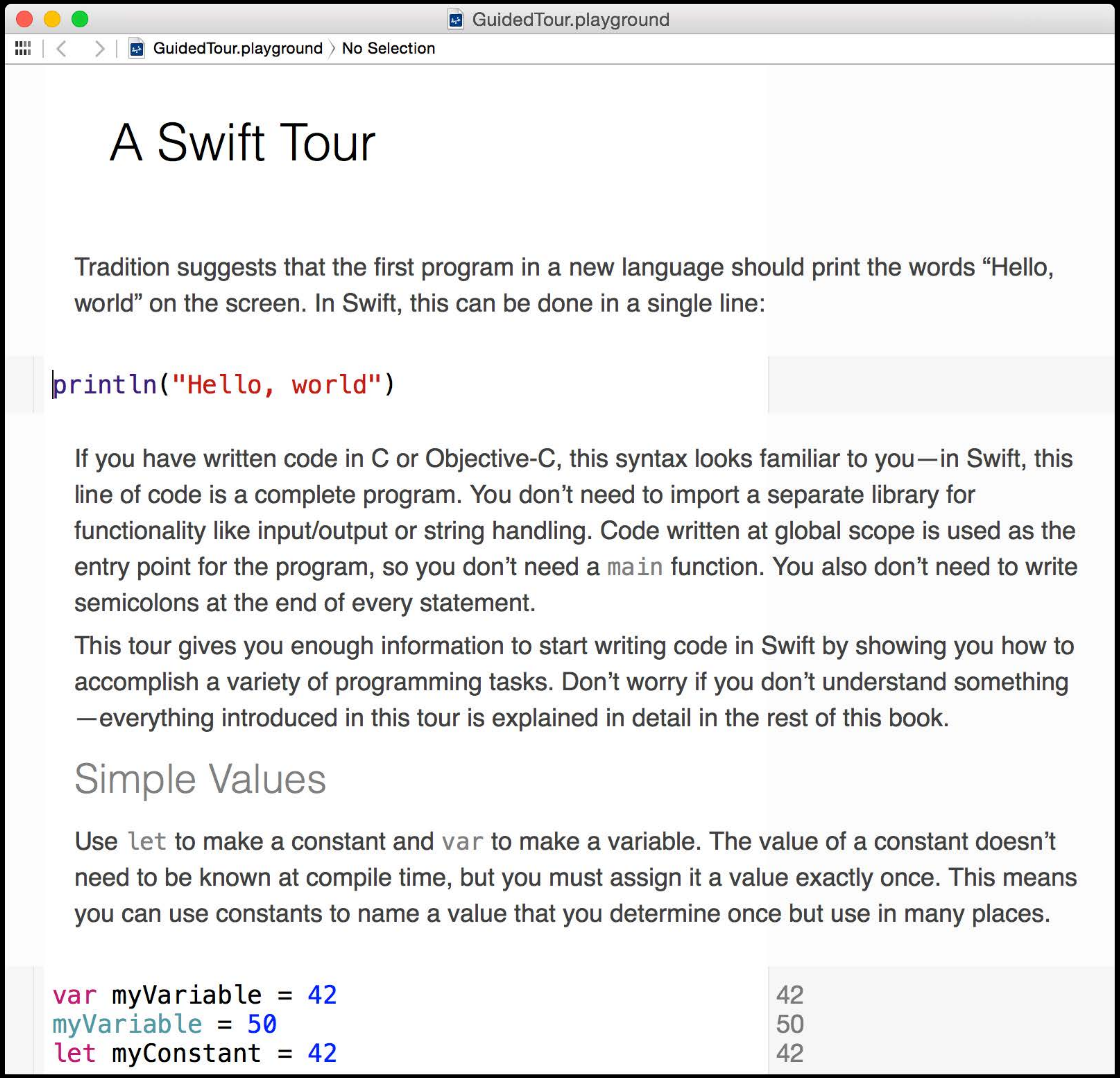
Learning

- Learn Swift by playing around

Why Use Playgrounds

Learning

- Learn Swift by playing around
- Interactive learning with the Swift Tour



A Swift Tour

Tradition suggests that the first program in a new language should print the words “Hello, world” on the screen. In Swift, this can be done in a single line:

```
println("Hello, world")
```

If you have written code in C or Objective-C, this syntax looks familiar to you—in Swift, this line of code is a complete program. You don’t need to import a separate library for functionality like input/output or string handling. Code written at global scope is used as the entry point for the program, so you don’t need a `main` function. You also don’t need to write semicolons at the end of every statement.

This tour gives you enough information to start writing code in Swift by showing you how to accomplish a variety of programming tasks. Don’t worry if you don’t understand something—everything introduced in this tour is explained in detail in the rest of this book.

Simple Values

Use `let` to make a constant and `var` to make a variable. The value of a constant doesn’t need to be known at compile time, but you must assign it a value exactly once. This means you can use constants to name a value that you determine once but use in many places.

```
var myVariable = 42                42
myVariable = 50                    50
let myConstant = 42                42
```

Why Use Playgrounds

Learning

- Learn Swift by playing around
- Interactive learning with the Swift Tour
- Teach programming to beginners

Why Use Playgrounds

Code development

Why Use Playgrounds

Code development

- Algorithm development

Why Use Playgrounds

Code development

- Algorithm development
- Drawing code development

Why Use Playgrounds

Code development

- Algorithm development
- Drawing code development
- Processing code (value transformers, image filters, etc.)

Why Use Playgrounds

Experimentation

Why Use Playgrounds

Experimentation

- Try out API

Why Use Playgrounds

Experimentation

- Try out API
- No project needed

Why Use Playgrounds

Experimentation

- Try out API
- No project needed
- Run code from a standalone document

Why Use Playgrounds

Experimentation

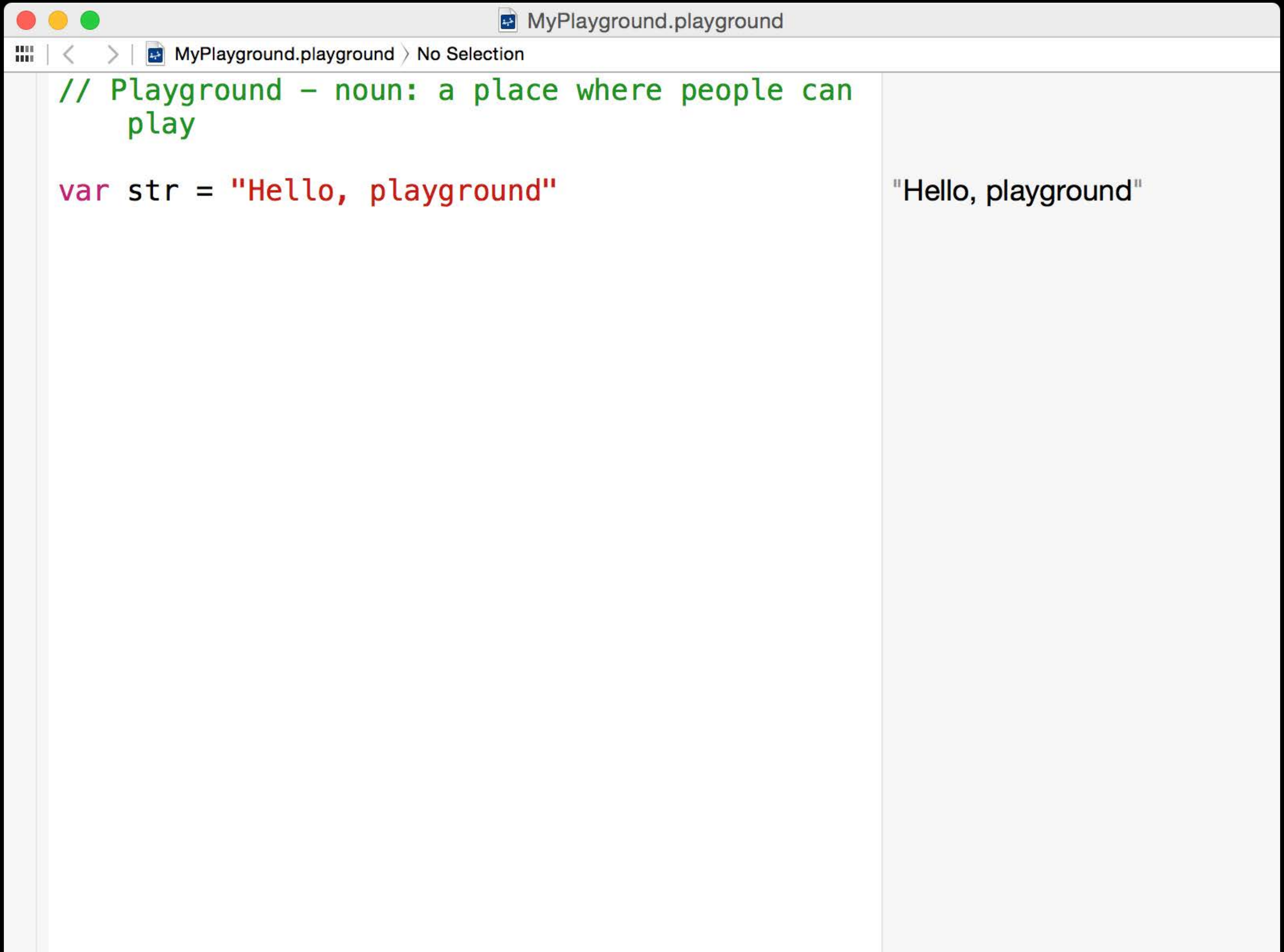
- Try out API
- No project needed
- Run code from a standalone document
- Keep a playground in your dock for quick access

Working with Playgrounds

Demo

Working with playgrounds

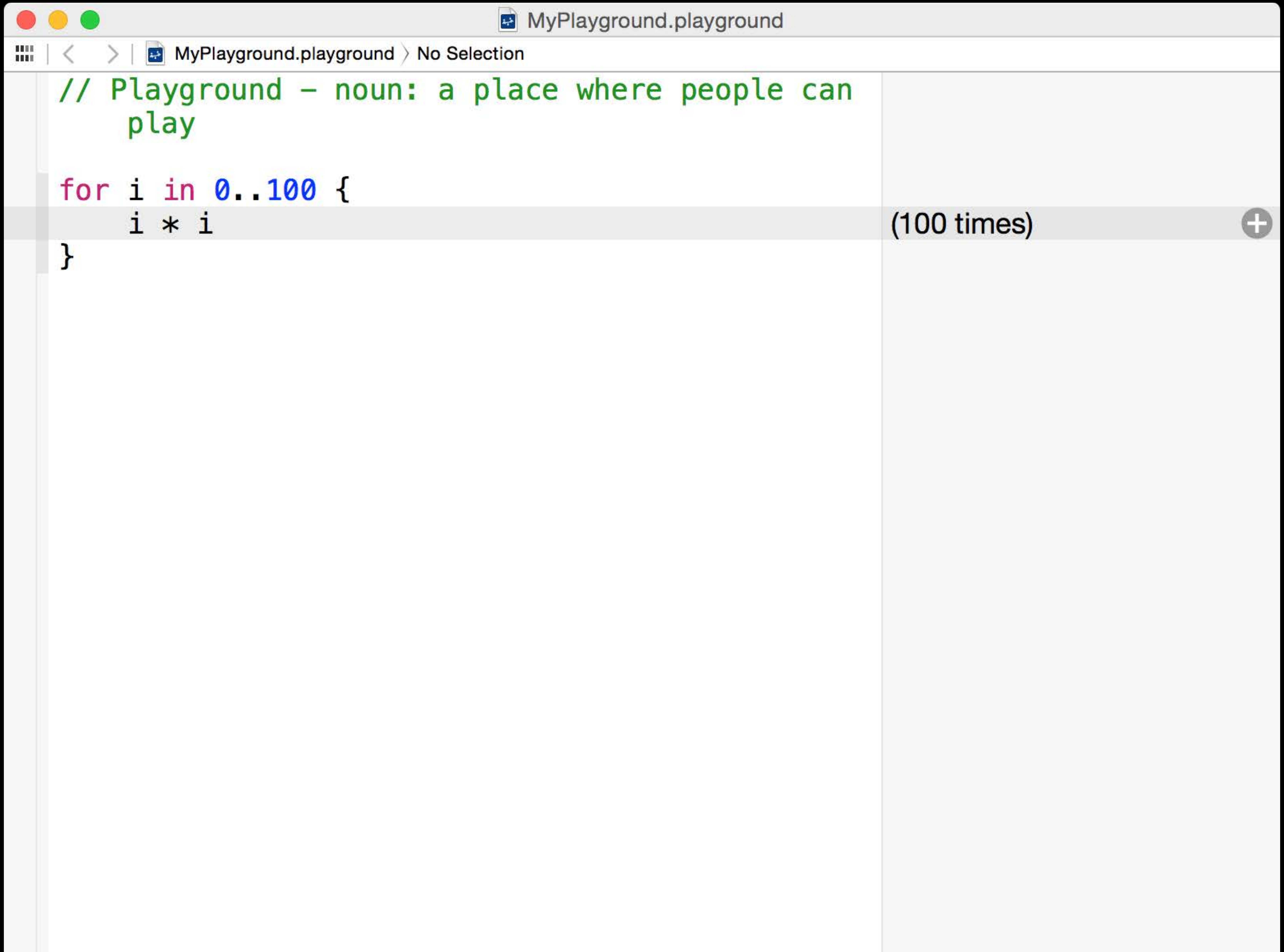
Playgrounds Automatically Execute



```
// Playground - noun: a place where people can
    play

var str = "Hello, playground"
```

```
"Hello, playground"
```

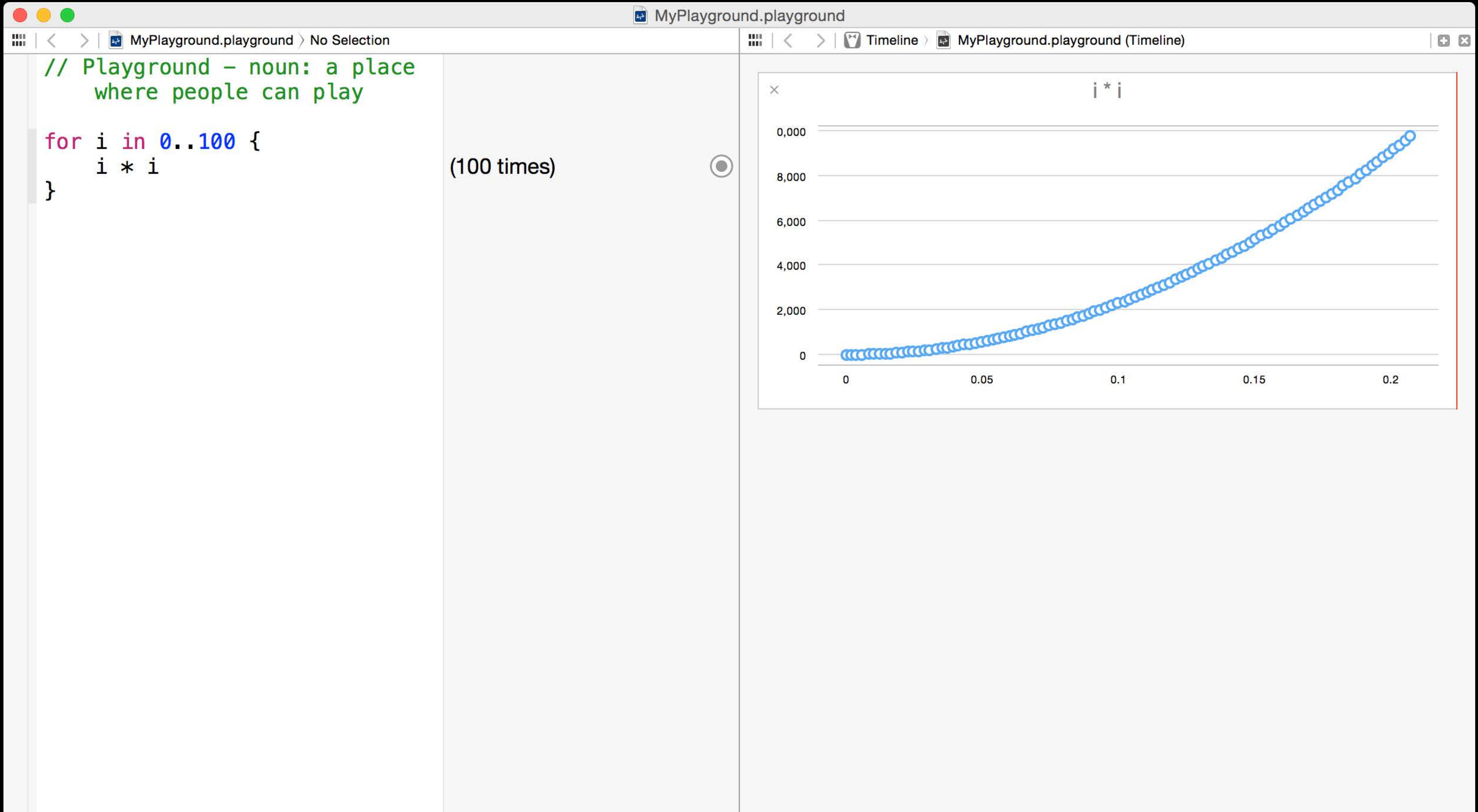
Add Value Histories to the Timeline



```
// Playground - noun: a place where people can
    play

for i in 0..100 {
    i * i
}
```

(100 times)

Add Value Histories to the Timeline

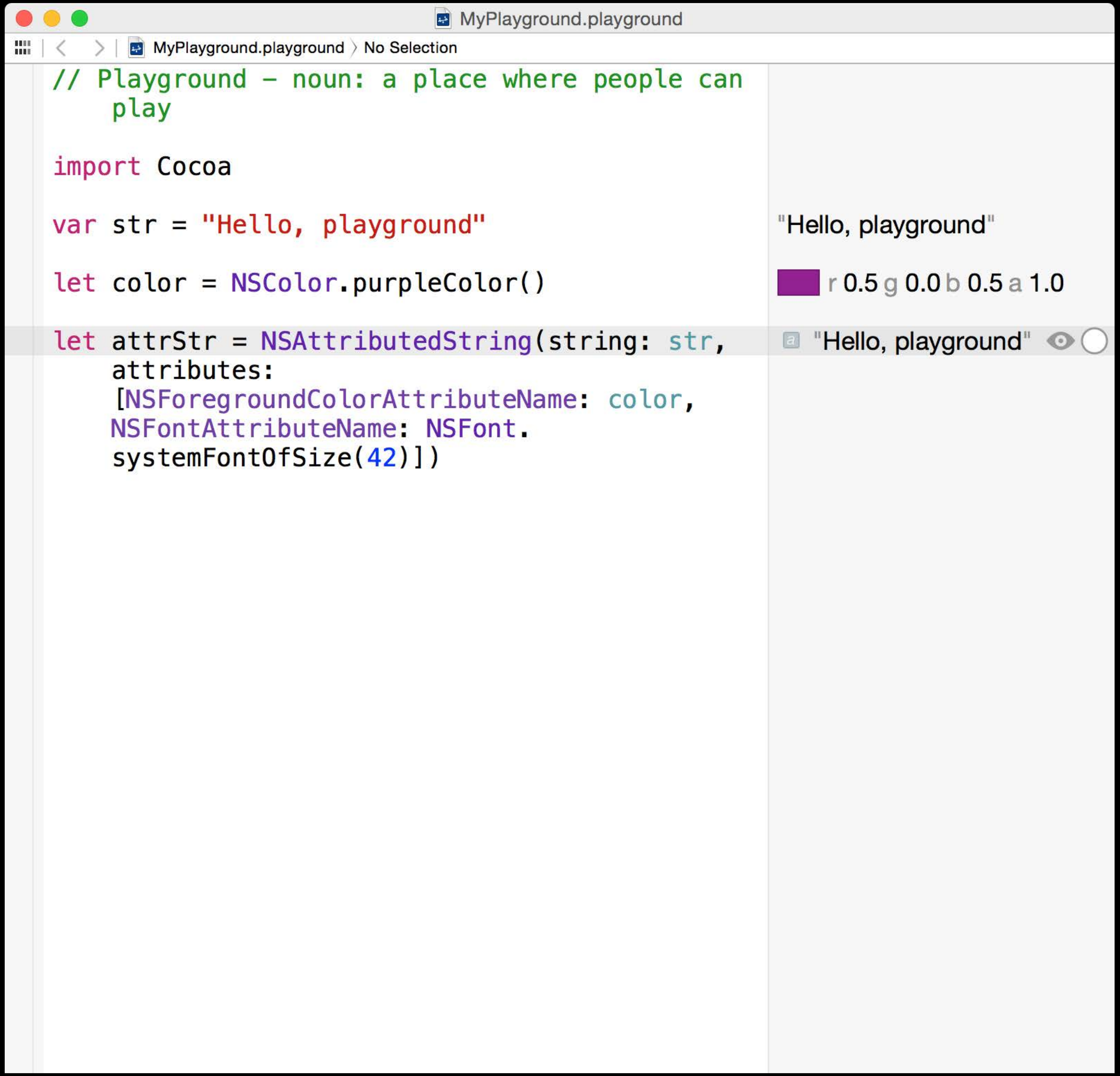


```
// Playground - noun: a place
    where people can play

for i in 0..100 {
    i * i
}
```

(100 times)

MyPlayground.playground
MyPlayground.playground 〉 No Selection
Timeline 〉 MyPlayground.playground (Timeline)
i * i
0,000
8,000
6,000
4,000
2,000
0
0
0.05
0.1
0.15
0.2

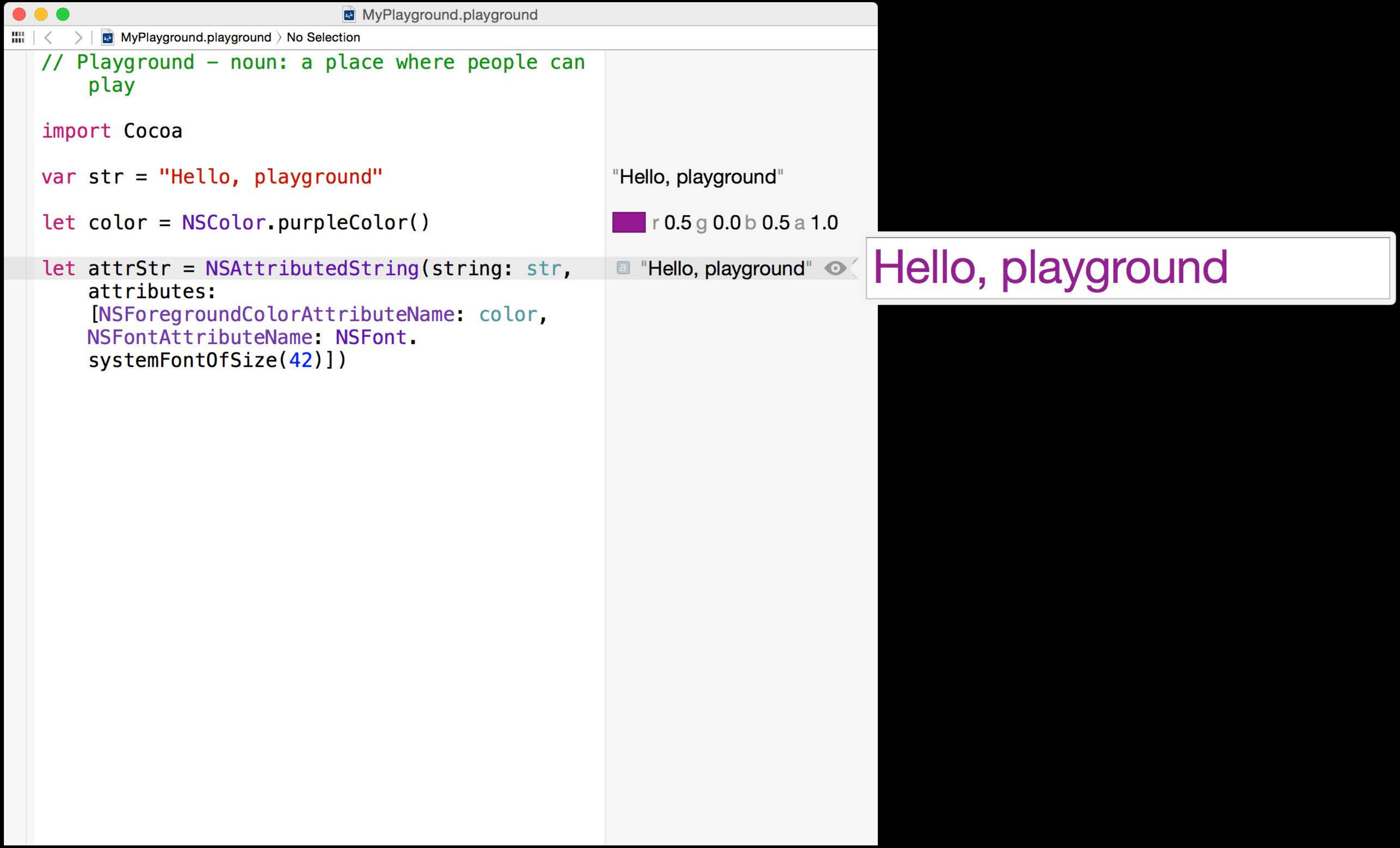
Many Values Have Quick Looks



```
// Playground - noun: a place where people can
    play

import Cocoa

var str = "Hello, playground"

let color = NSColor.purpleColor()

let attrStr = NSAttributedString(string: str,
    attributes:
    [NSForegroundColorAttributeName: color,
    NSFontAttributeName: NSFont.
    systemFontOfSize(42)])
```

```
"Hello, playground"

r 0.5 g 0.0 b 0.5 a 1.0

"Hello, playground"
```

Many Values Have Quick Looks



```
// Playground - noun: a place where people can
    play

import Cocoa

var str = "Hello, playground"

let color = NSColor.purpleColor()

let attrStr = NSAttributedString(string: str,
    attributes:
    [NSForegroundColorAttributeName: color,
    NSFontAttributeName: NSFont.
    systemFontOfSize(42)])
```

```
"Hello, playground"
r 0.5 g 0.0 b 0.5 a 1.0
"Hello, playground"
```

Hello, playground

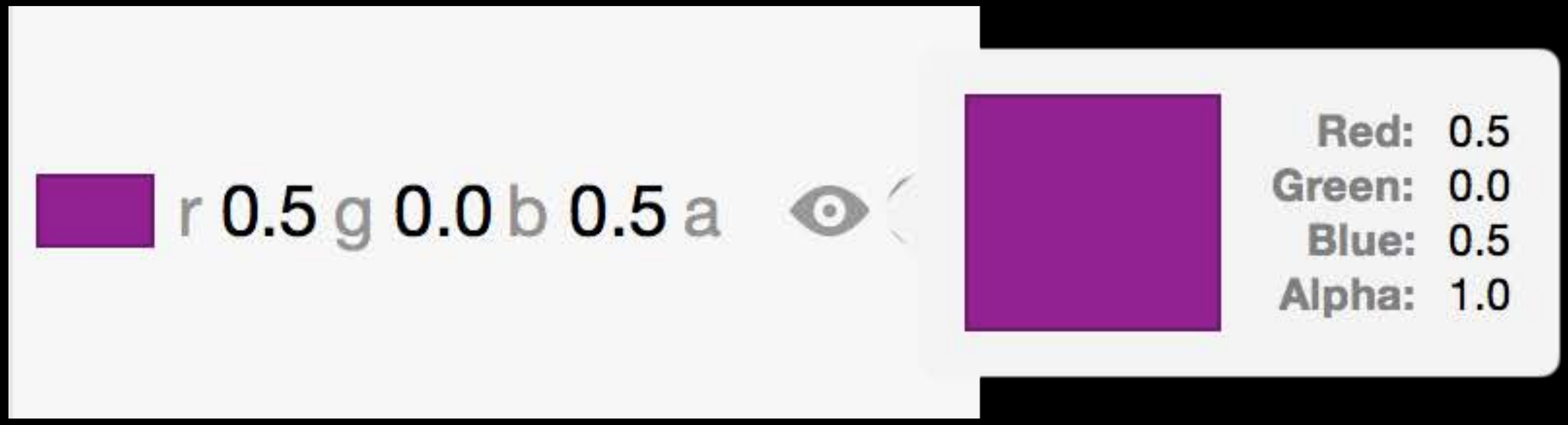
Many Values Have Quick Looks

Supported types

Many Values Have Quick Looks

Supported types

Colors



r 0.5 g 0.0 b 0.5 a

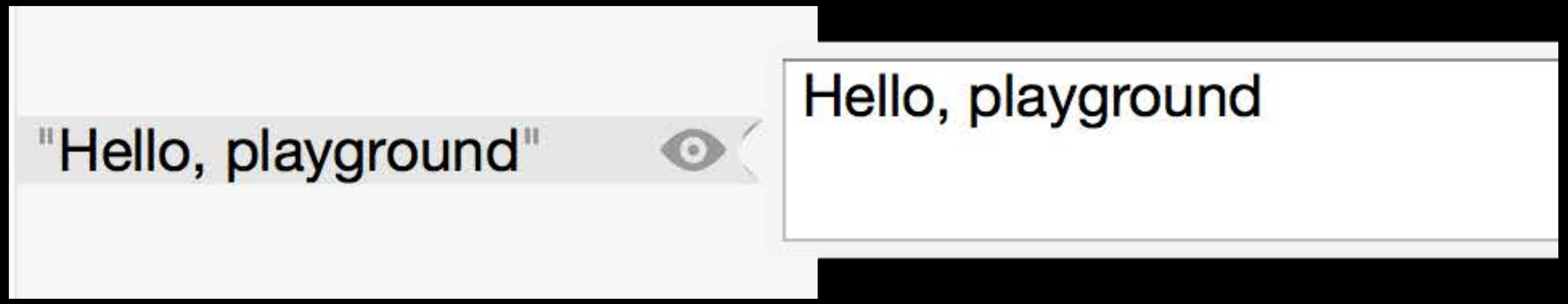
```
Red:   0.5
Green: 0.0
Blue:  0.5
Alpha: 1.0
```

Many Values Have Quick Looks

Supported types

Colors

Strings (plain and attributed)



"Hello, playground"

Hello, playground

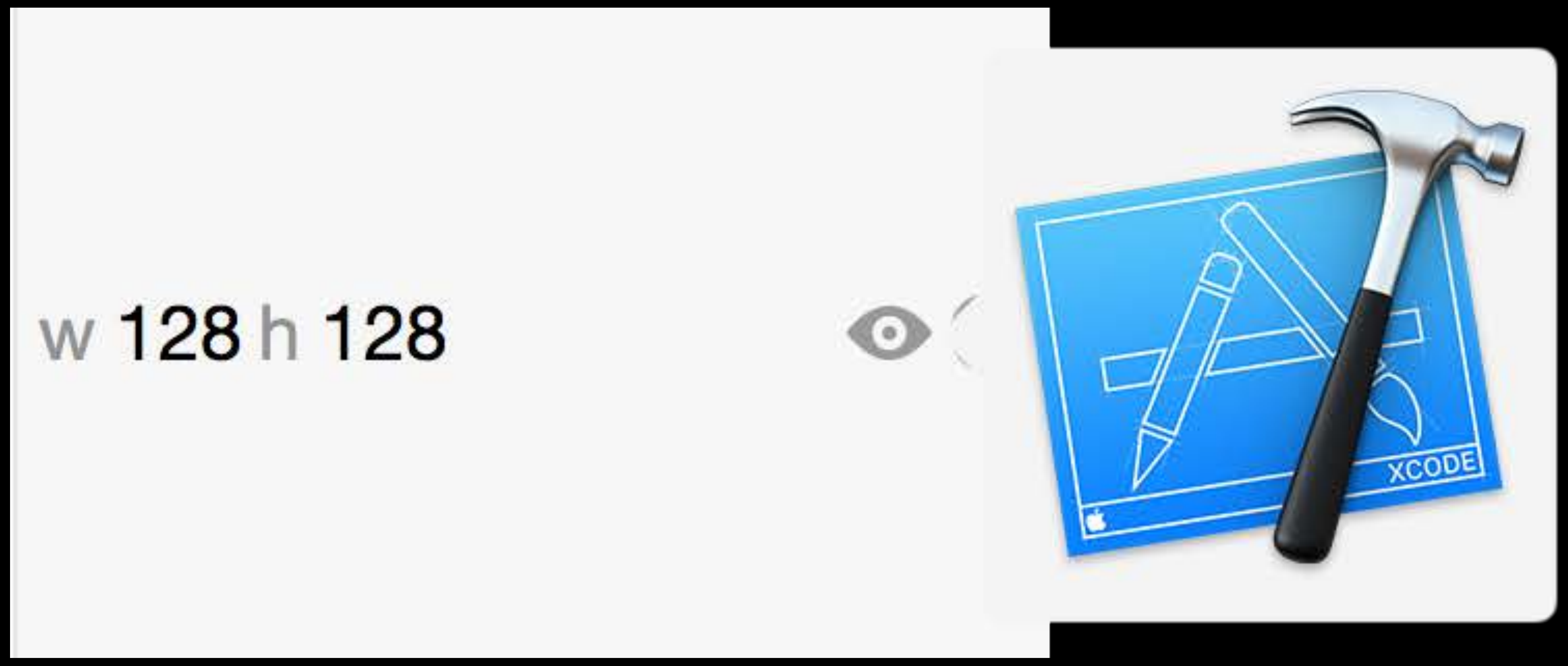
Many Values Have Quick Looks

Supported types

Colors

Strings (plain and attributed)

Images


w 128 h 128

Many Values Have Quick Looks

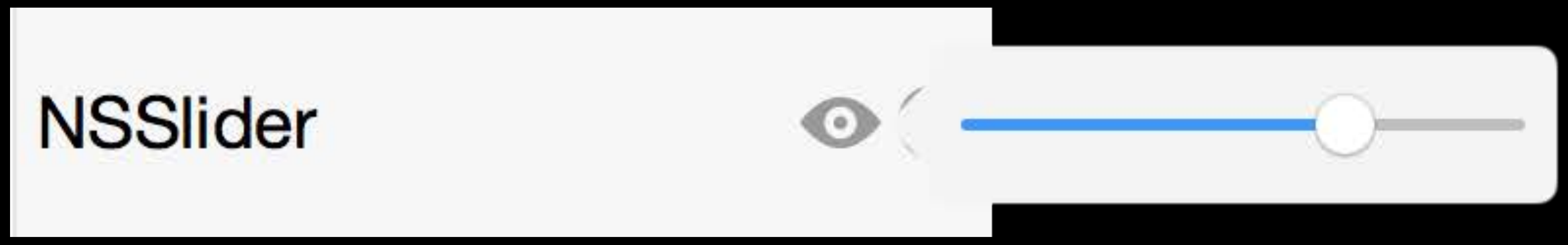
Supported types

Colors

Strings (plain and attributed)

Images

Views



Many Values Have Quick Looks

Supported types

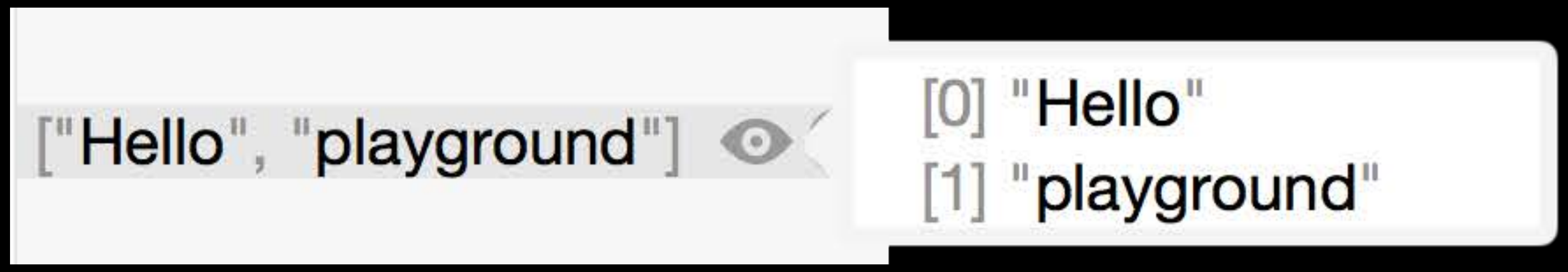
Colors

Strings (plain and attributed)

Images

Views

Arrays and dictionaries



```
["Hello", "playground"]
```

```
[0] "Hello"
[1] "playground"
```

Many Values Have Quick Looks

Supported types

Colors

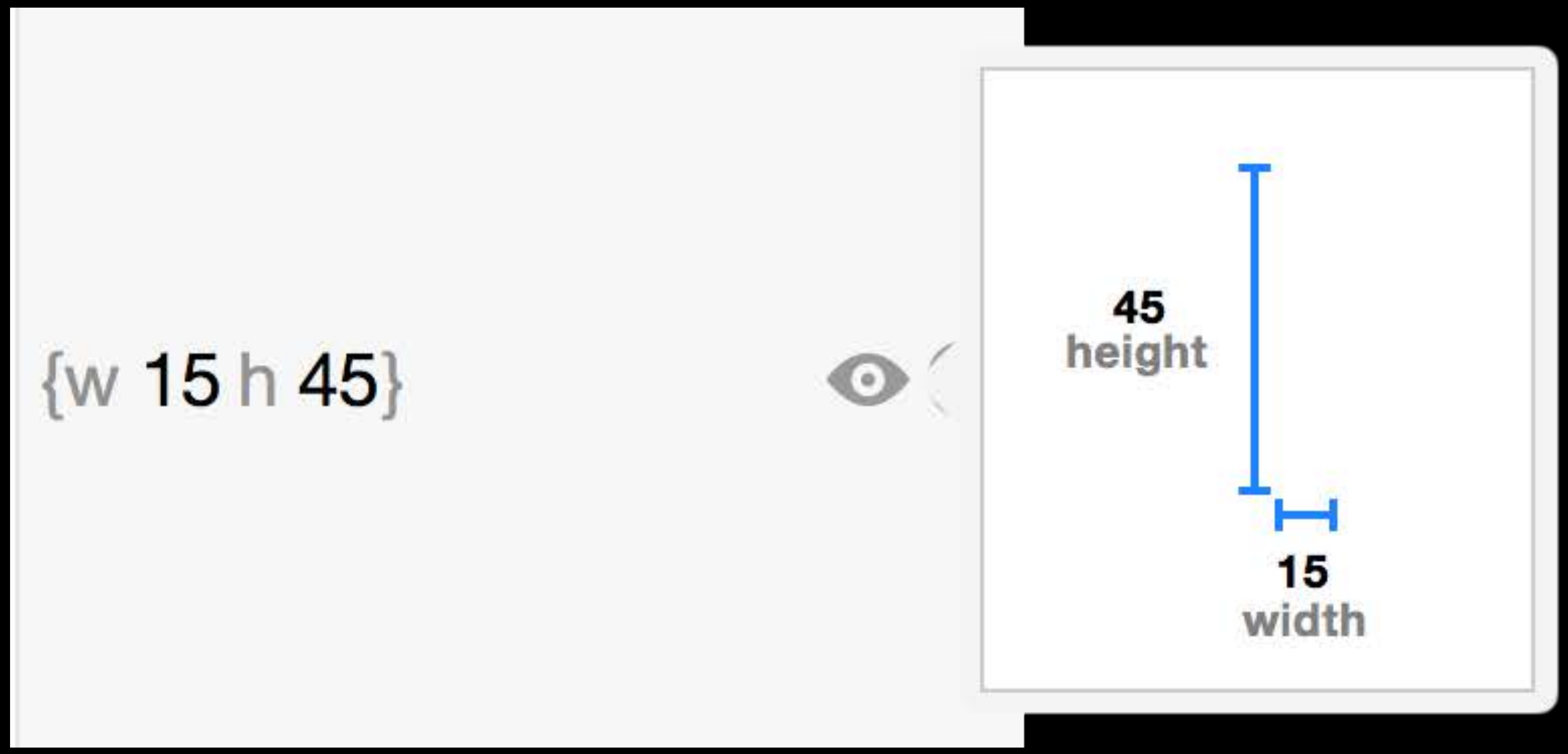
Strings (plain and attributed)

Images

Views

Arrays and dictionaries

Points, rects, sizes



{w 15 h 45}

Many Values Have Quick Looks

Supported types

Colors

Strings (plain and attributed)

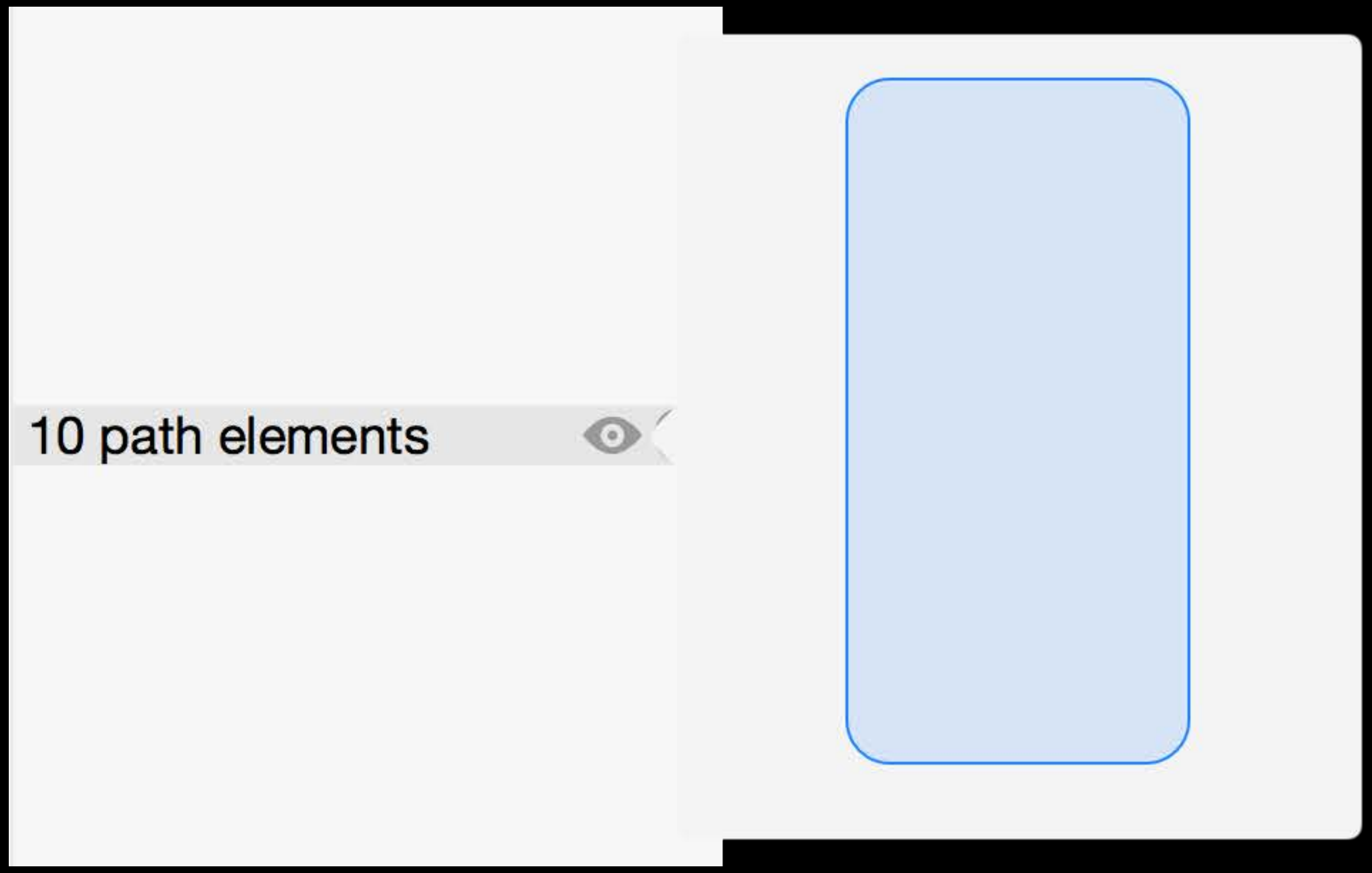
Images

Views

Arrays and dictionaries

Points, rects, sizes

Bézier paths



10 path elements

Many Values Have Quick Looks

Supported types

Colors

Strings (plain and attributed)

Images

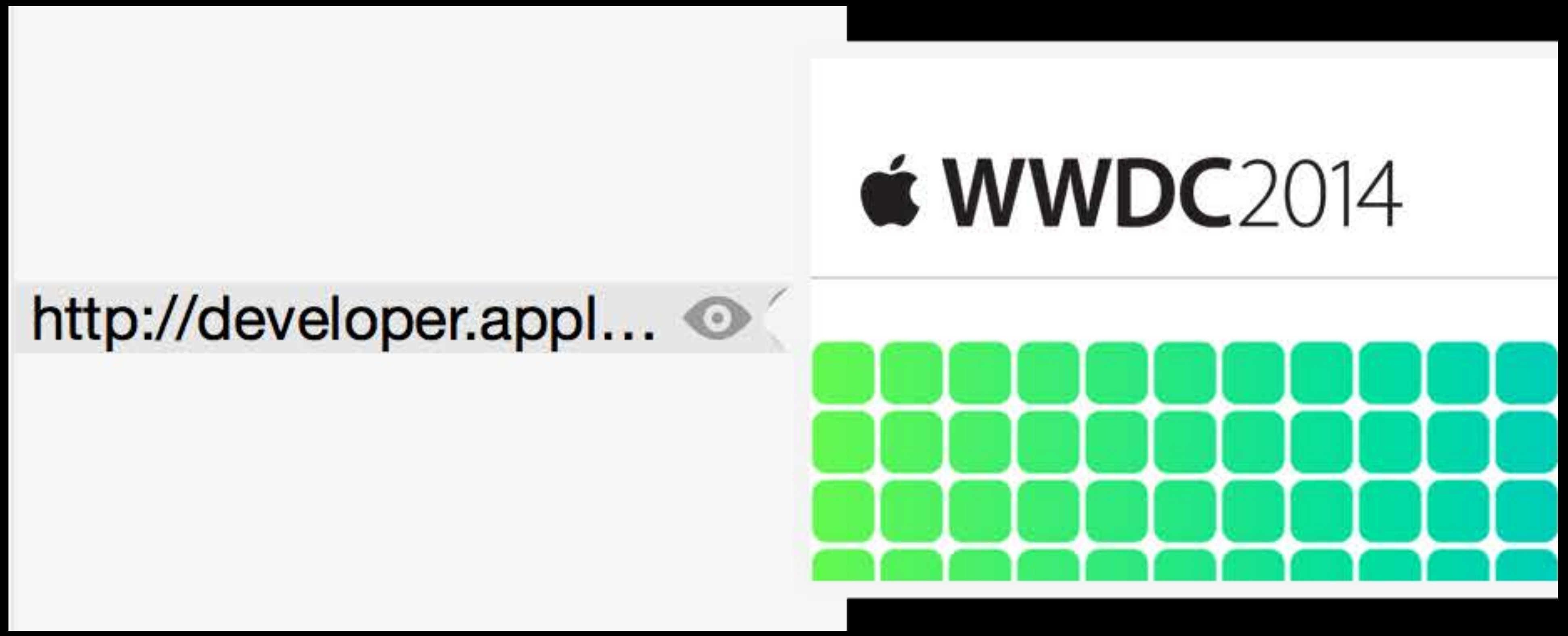
Views

Arrays and dictionaries

Points, rects, sizes

Bézier paths

URLs


http://developer.appl…

Many Values Have Quick Looks

Supported types

Colors

Strings (plain and attributed)

Images

Views

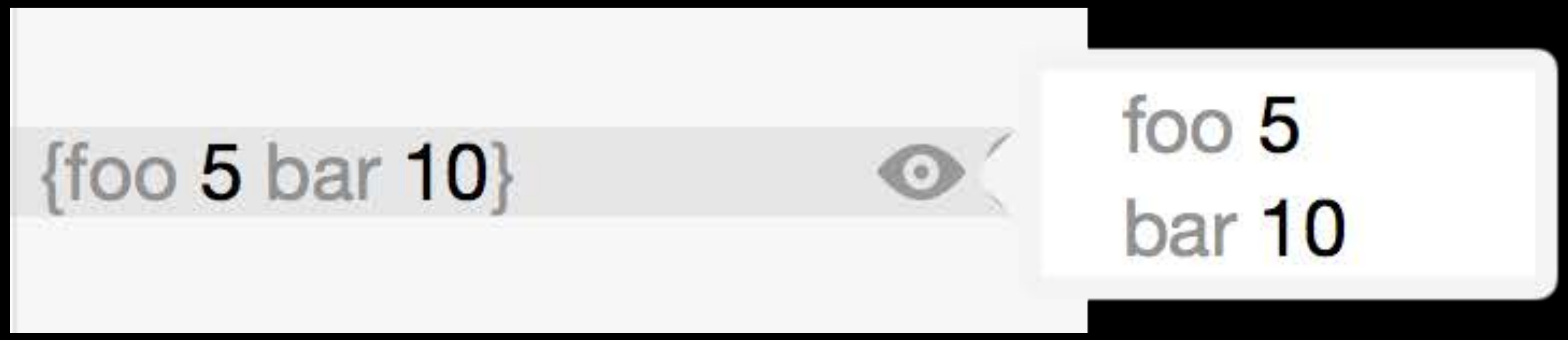
Arrays and dictionaries

Points, rects, sizes

Bézier paths

URLs

Classes and structs



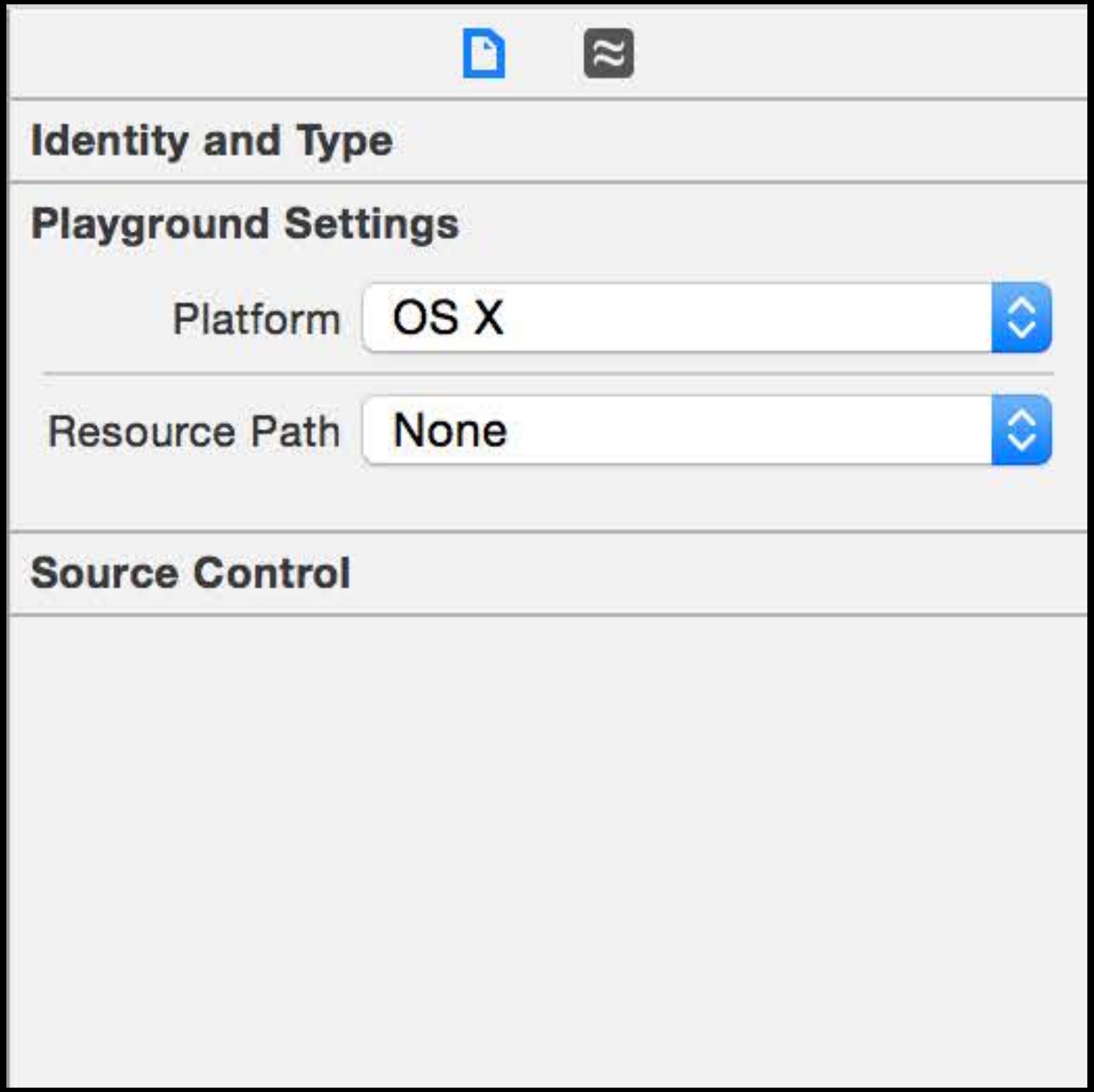
{foo 5 bar 10}

foo 5
bar 10

Using Resources in Playgrounds

Using Resources in Playgrounds

Show the file inspector



Identity and Type

Playground Settings

Platform: OS X

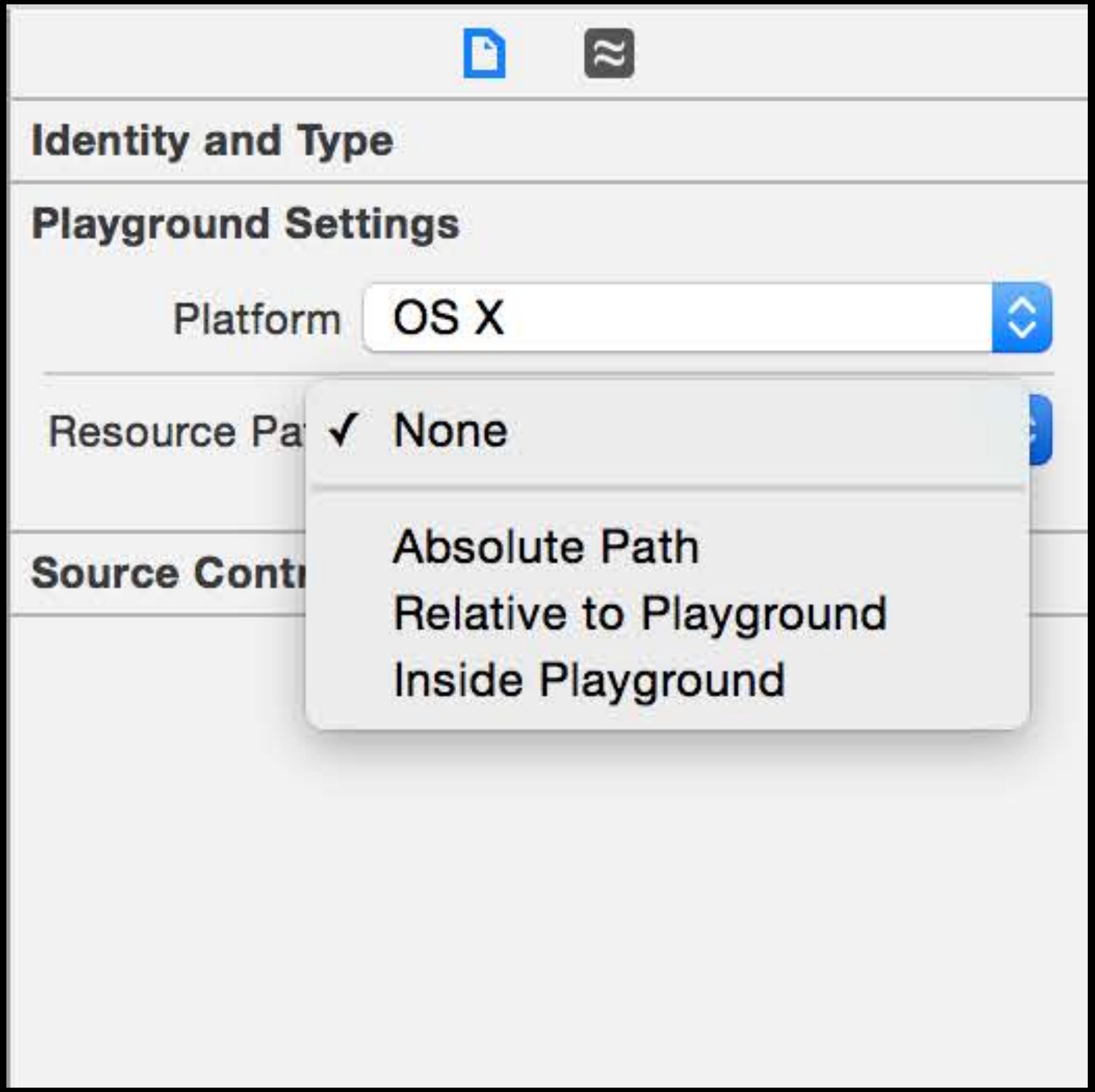
Resource Path: None

Source Control

Using Resources in Playgrounds

Show the file inspector

Select location for playground resources



Identity and Type

Playground Settings

Platform OS X

Resource Pa ✓ None

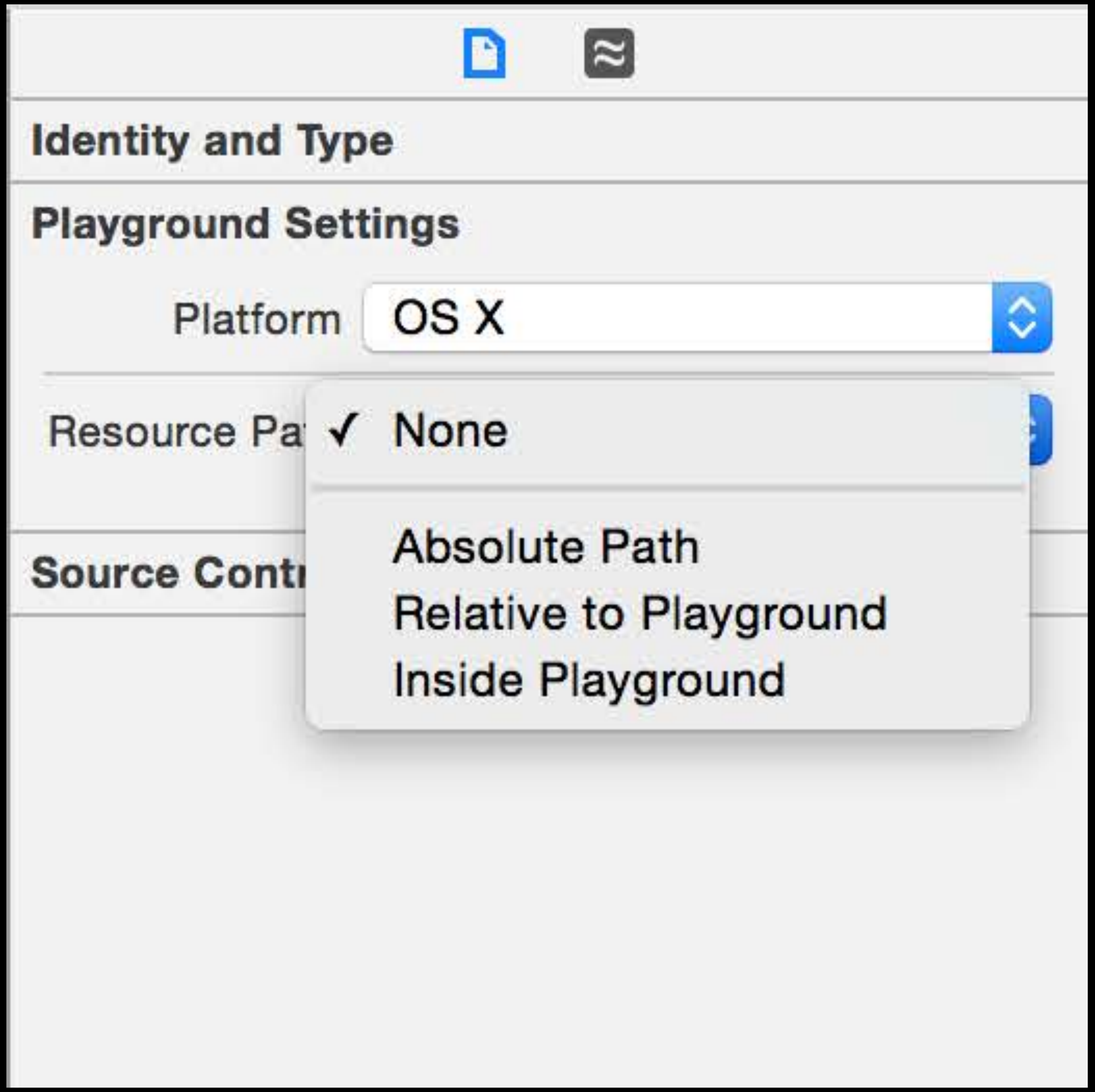
Source Contr

Using Resources in Playgrounds

Show the file inspector

Select location for playground resources

- Absolute Path



Identity and Type

Playground Settings

Platform: OS X

Resource Pa

✓ None

Absolute Path

Relative to Playground

Inside Playground

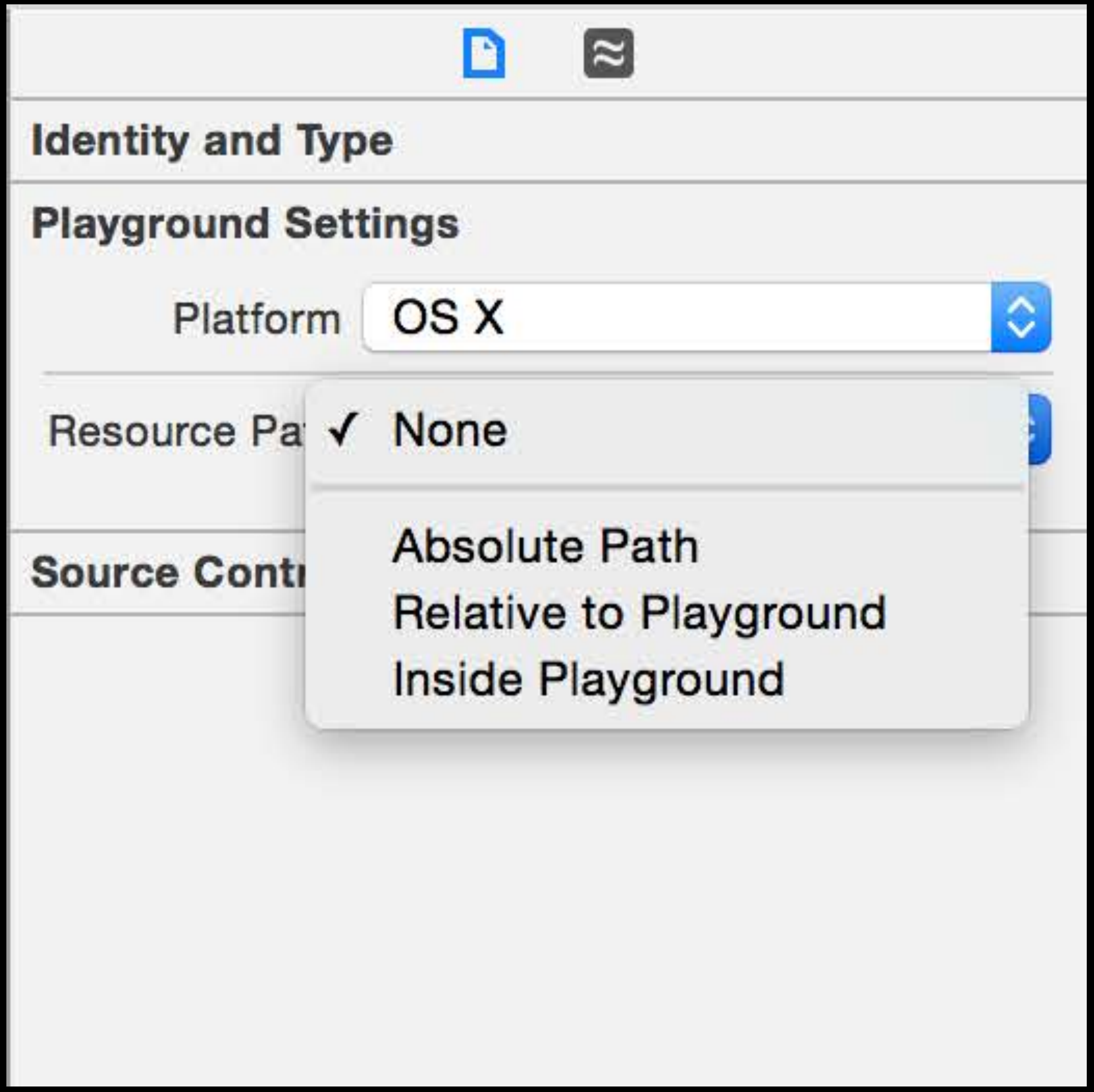
Source Contr

Using Resources in Playgrounds

Show the file inspector

Select location for playground resources

- Absolute Path
- Relative to Playground



Identity and Type

Playground Settings

Platform OS X

Resource Pa

✓ None

Absolute Path

Relative to Playground

Inside Playground

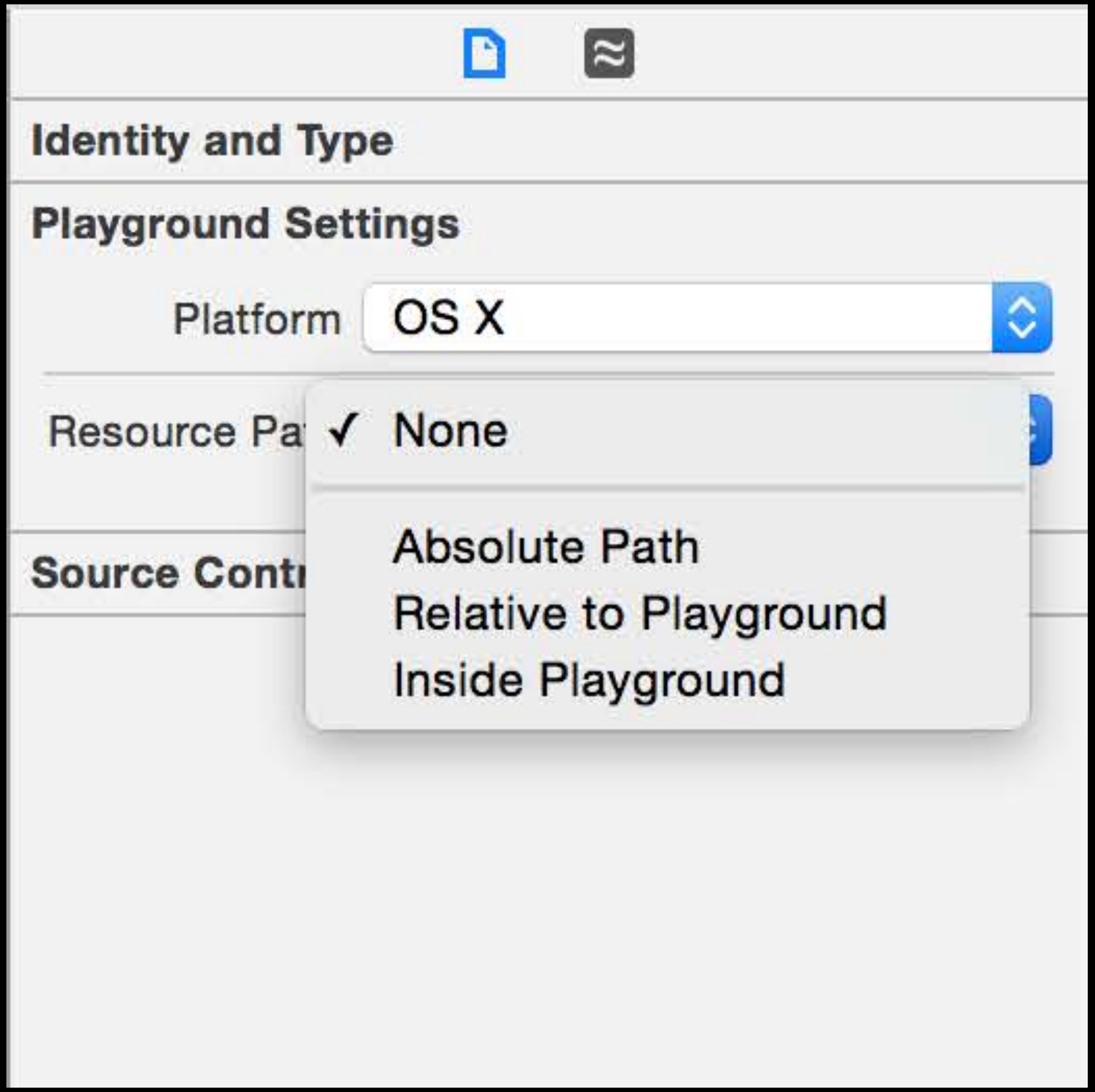
Source Contr

Using Resources in Playgrounds

Show the file inspector

Select location for playground resources

- Absolute Path
- Relative to Playground
- Inside Playground



Identity and Type

Playground Settings

Platform OS X

Resource Pa ✓ None

Absolute Path

Relative to Playground

Inside Playground

Source Contr

Using Resources in Playgrounds

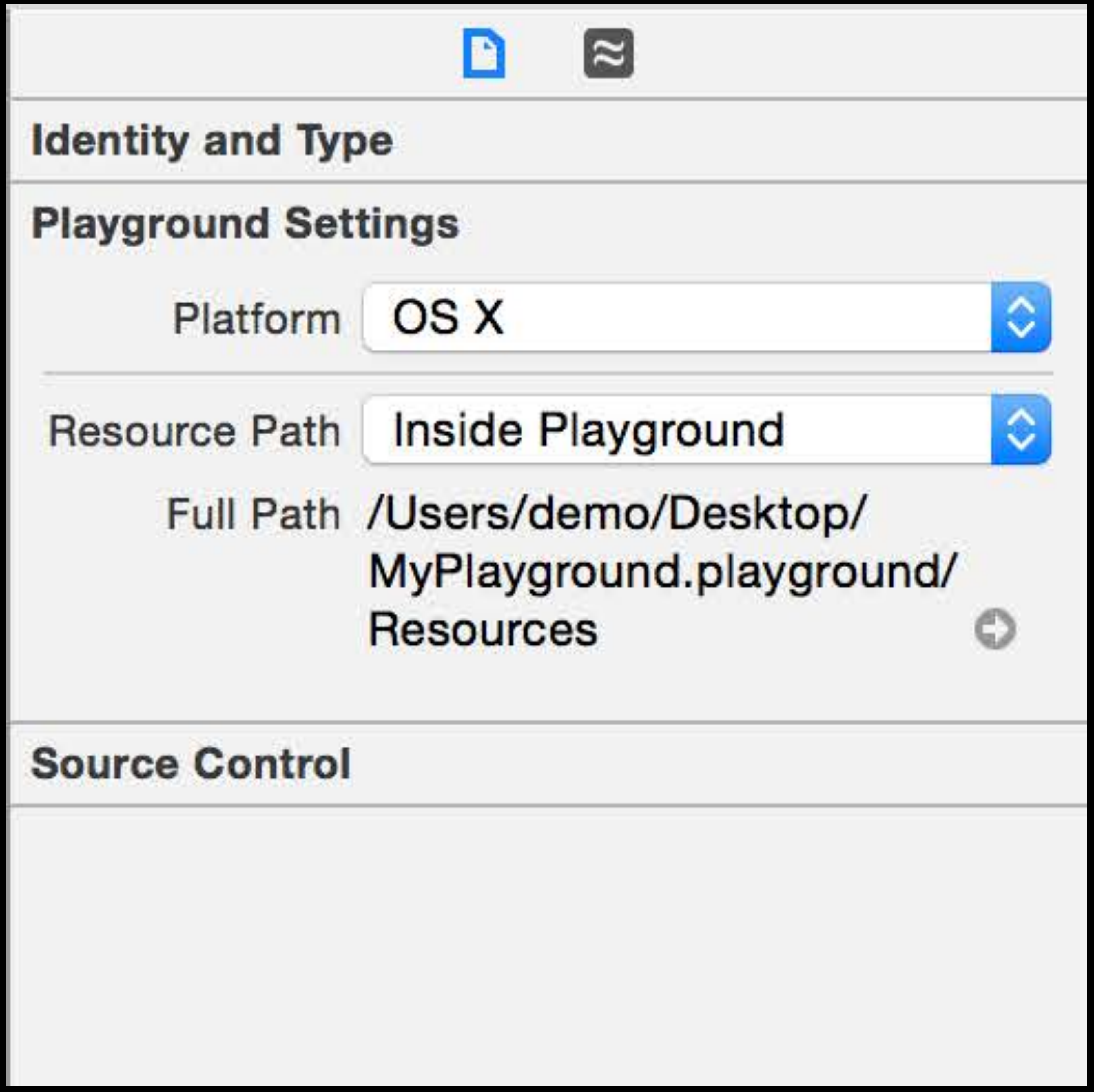
Show the file inspector

Select location for playground resources

- Absolute Path
- Relative to Playground
- Inside Playground

In source, access resources using NSBundle and related API, such as:

```
NSBundle.pathForResource(_:ofType:)
NSImage(named:)
```



Identity and Type

Playground Settings

Platform: OS X

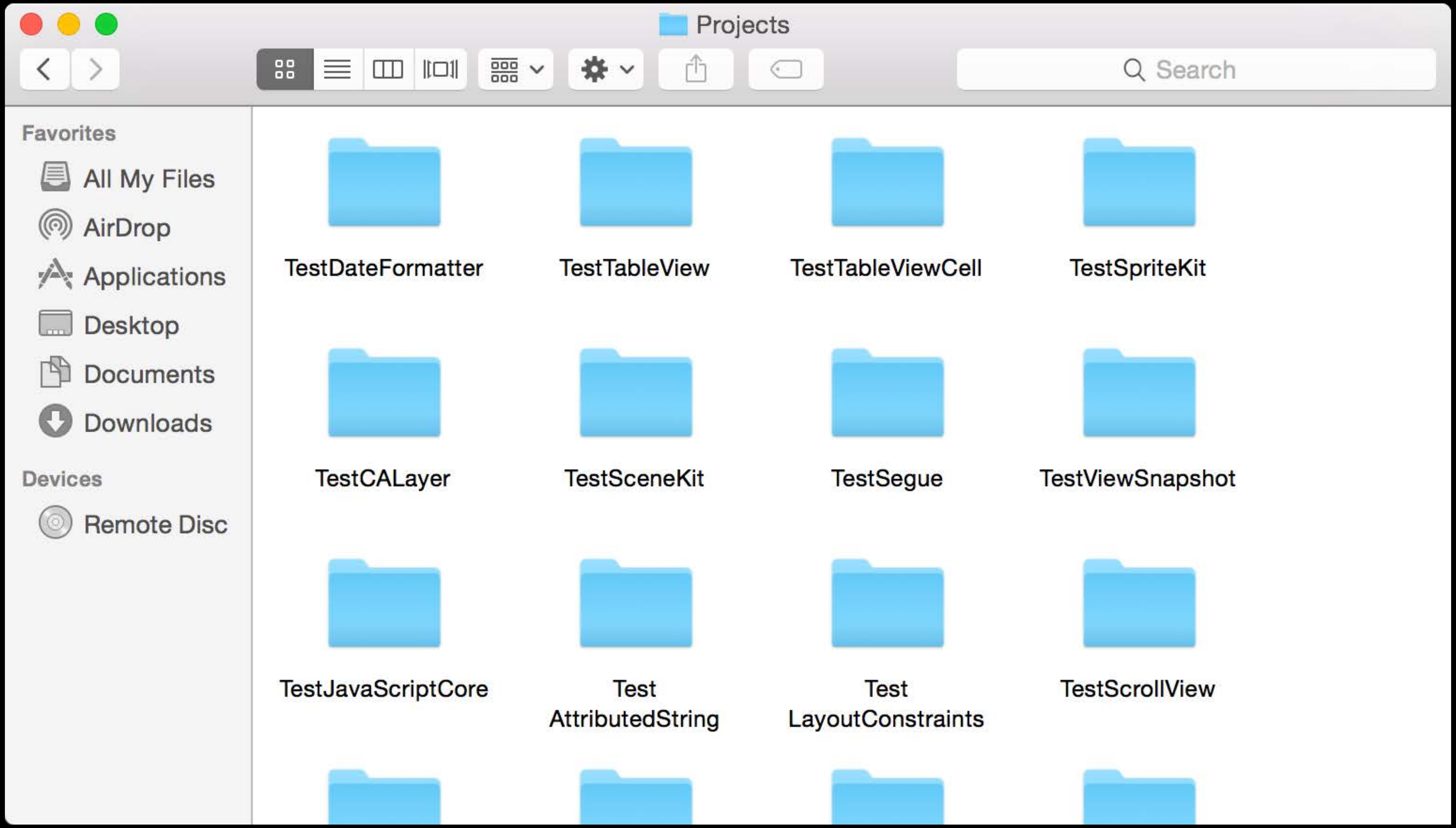
Resource Path: Inside Playground

Full Path: /Users/demo/Desktop/MyPlayground.playground/Resources

Source Control

Experimentation

TestFoo.xcodeproj



TestFoo.xcodeproj

Choose a project template

TestFoo.xcodeproj

Choose a project template

Find the right file to edit

TestFoo.xcodeproj

Choose a project template

Find the right file to edit

Write your code

TestFoo.xcodeproj

Choose a project template

Find the right file to edit

Write your code

Build

TestFoo.xcodeproj

Choose a project template

Find the right file to edit

Write your code

Build

Run

TestFoo.xcodeproj

Choose a project template

Find the right file to edit

Write your code

Build

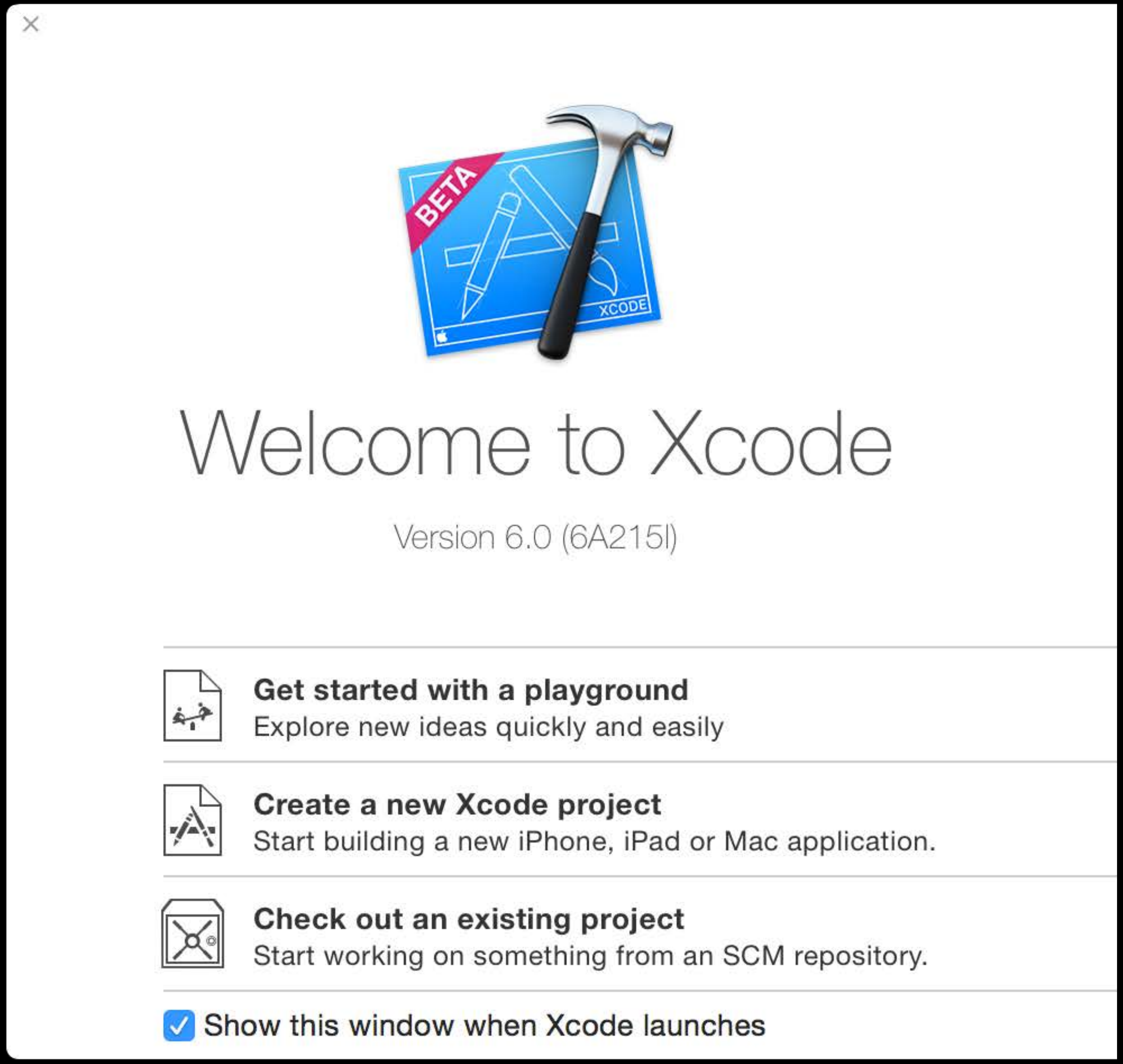
Run

Debug

TestFoo.playground

TestFoo.playground

Get started with a playground



Welcome to Xcode

Version 6.0 (6A215l)

Get started with a playground
Explore new ideas quickly and easily

Create a new Xcode project
Start building a new iPhone, iPad or Mac application.

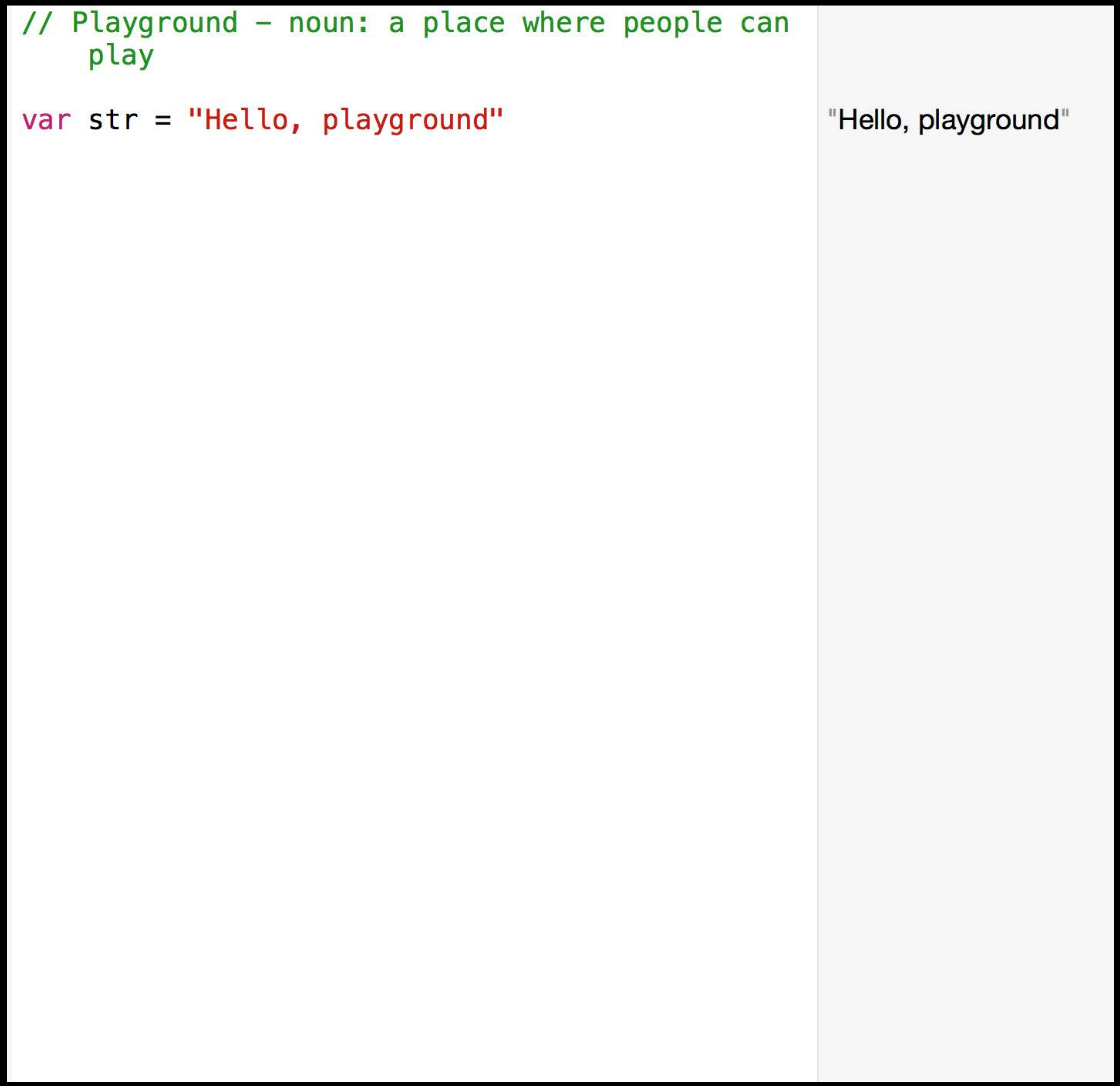
Check out an existing project
Start working on something from an SCM repository.

☑ Show this window when Xcode launches

TestFoo.playground

Get started with a playground

Write your code



```
// Playground - noun: a place where people can
    play

var str = "Hello, playground"
```

```
"Hello, playground"
```

Demo

Experimentation

Algorithm Development

Demo

Algorithm development

XCPlayground

Contains utilities for use in playgrounds

XCPlayground

Contains utilities for use in playgrounds

- Manually capturing values
- Showing live views
- Extending execution

XCPlayground

Contains utilities for use in playgrounds

- Manually capturing values
- Showing live views
- Extending execution

Manually Capturing Values

XCPCaptureValue

Manually Capturing Values

XCPCaptureValue

```
func XCPCaptureValue<T>(identifier: String, value: T)
```

Manually Capturing Values

XCPCaptureValue

```
func XCPCaptureValue<T>(identifier: String, value: T)
```

Captures values for manual value histories in the timeline

Manually Capturing Values

XCPCaptureValue

```
func XCPCaptureValue<T>(identifier: String, value: T)
```

Captures values for manual value histories in the timeline

`identifier` identifies a collection of captured values

Manually Capturing Values

XCPCaptureValue

```
func XCPCaptureValue<T>(identifier: String, value: T)
```

Captures values for manual value histories in the timeline

`identifier` identifies a collection of captured values

- Values with the same identifier are shown in a single value history

Manually Capturing Values

XCPCaptureValue

```
func XCPCaptureValue<T>(identifier: String, value: T)
```

Captures values for manual value histories in the timeline

`identifier` identifies a collection of captured values

- Values with the same identifier are shown in a single value history
- Shown as the title for the value history in the timeline

Manually Capturing Values

XCPCaptureValue

```
func XCPCaptureValue<T>(identifier: String, value: T)
```

Captures values for manual value histories in the timeline

`identifier` identifies a collection of captured values

- Values with the same identifier are shown in a single value history
- Shown as the title for the value history in the timeline

`value` can be anything

Demo

Algorithm development

Custom Quick Look Support

Custom Quick Look Support

Custom Quick Look Support

Add Quick Look support to NSObject subclasses only

Custom Quick Look Support

Add Quick Look support to NSObject subclasses only

Implement the `debugQuickLookObject()` method

```
func debugQuickLookObject() -> AnyObject? {
    return "Some Quick Look type"
}
```

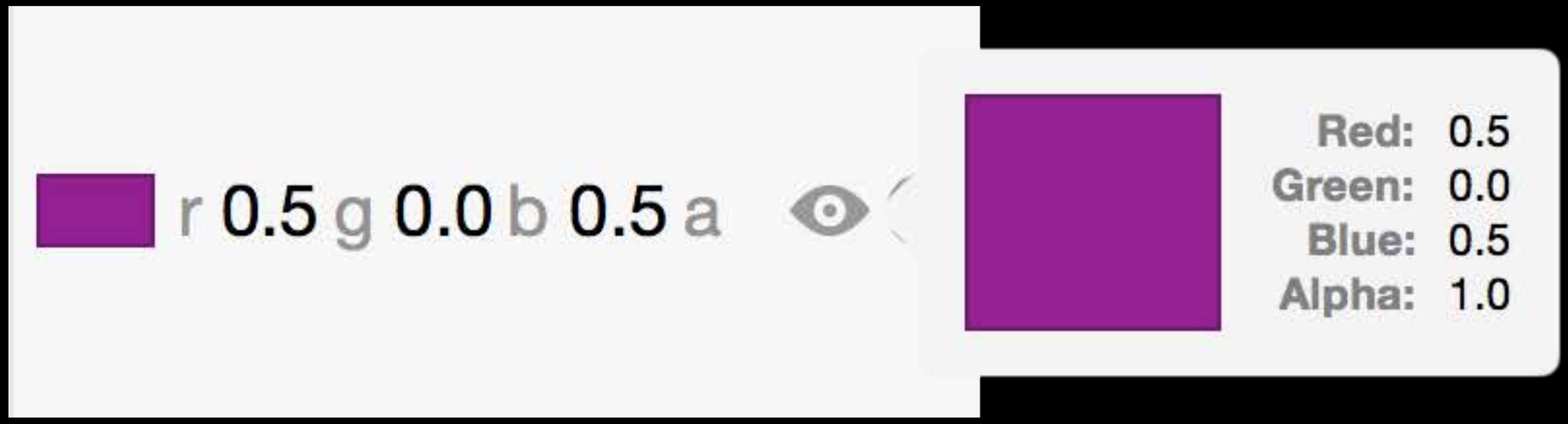
Custom Quick Look Support

Custom Quick Look types

Custom Quick Look Support

Custom Quick Look types

Colors



r 0.5 g 0.0 b 0.5 a

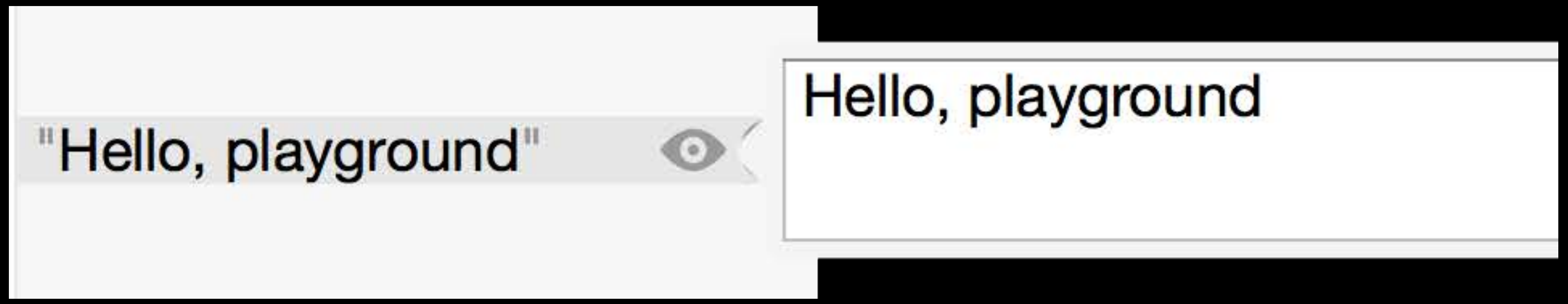
Red:	0.5
Green:	0.0
Blue:	0.5
Alpha:	1.0

Custom Quick Look Support

Custom Quick Look types

Colors

Strings (plain and attributed)



"Hello, playground"

Hello, playground

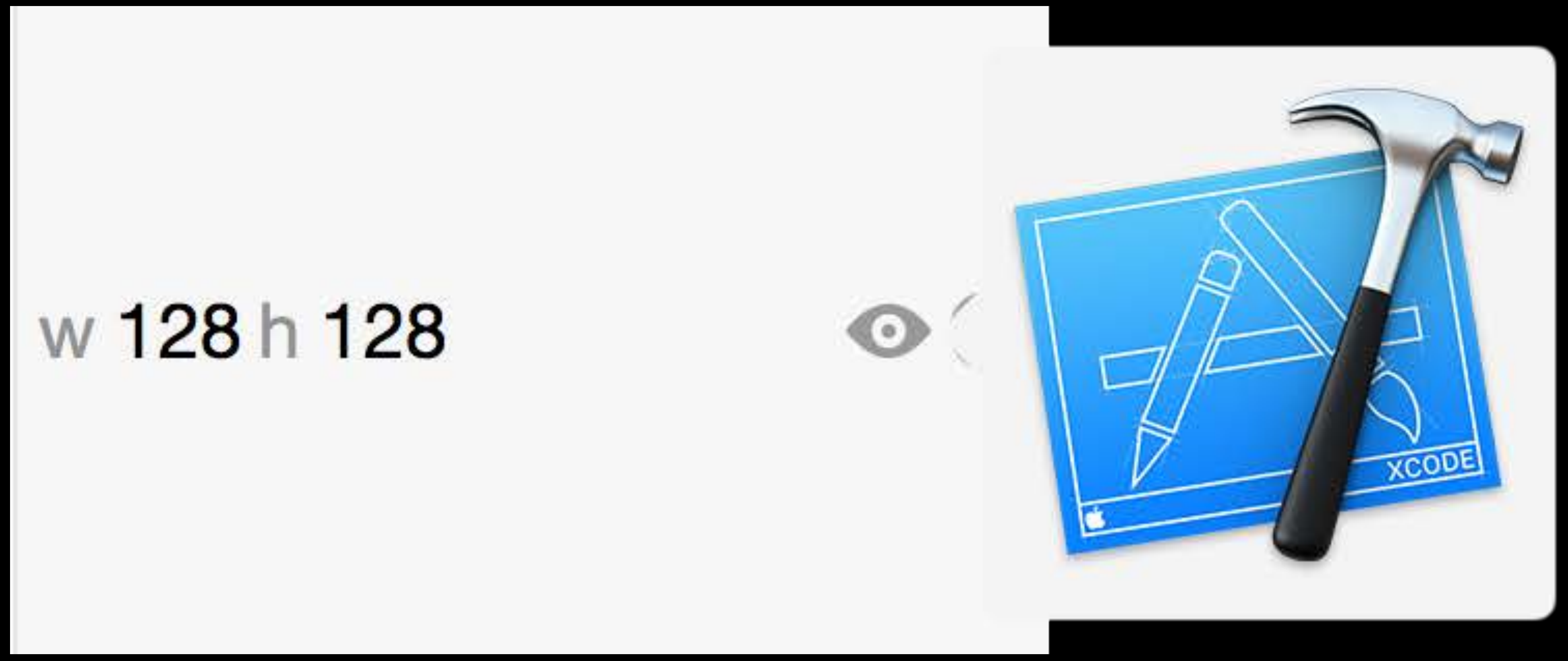
Custom Quick Look Support

Custom Quick Look types

Colors

Strings (plain and attributed)

Images



Custom Quick Look Support

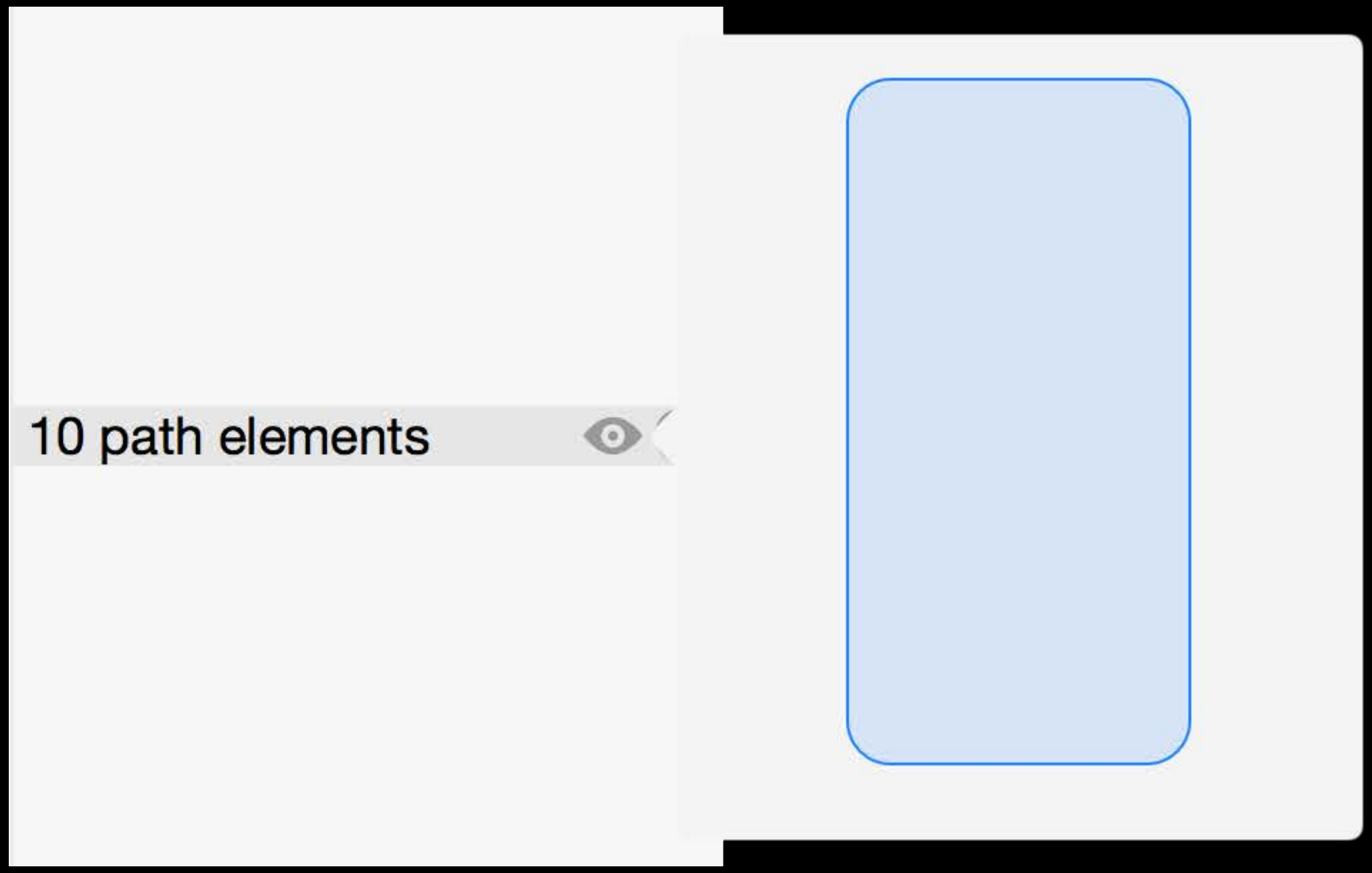
Custom Quick Look types

Colors

Strings (plain and attributed)

Images

Bézier paths



Demo

Custom Quick Look support

Custom View Development

Goal: Animate the Playground Icon

Demo

Custom view development

XCPlayground

Contains utilities for use in playgrounds

- Manually capturing values
- Showing live views
- Extending execution

Showing Live Views

XCPShowView

```
func XCPShowView(identifier: String, view: NSView)
```

Shows view live as playground executes and records frames for playback

Showing Live Views

XCPShowView

```
func XCPShowView(identifier: String, view: NSView)
```

Shows view live as playground executes and records frames for playback

`identifier` must be unique in the playground

- Shown as the title for the live view in the timeline

Showing Live Views

XCPShowView

```
func XCPShowView(identifier: String, view: NSView)
```

Shows view live as playground executes and records frames for playback

`identifier` must be unique in the playground

- Shown as the title for the live view in the timeline

`view` must not have a superview

- Automatically added to a window for display in the timeline

Demo

Custom view development

Asynchronous Code in Playgrounds

XCPlayground

Contains utilities for use in playgrounds

- Manually capturing values
- Showing live views
- Extending execution

Extending Execution

XCPSetExecutionShouldContinueIndefinitely

By default, execution terminates once all top-level code has executed

Extending Execution

XCPSetExecutionShouldContinueIndefinitely

By default, execution terminates once all top-level code has executed

XCPlayground includes API for extending execution indefinitely

```
func XCPSetExecutionShouldContinueIndefinitely(
    continueIndefinitely: Bool = true)
```

Extending Execution

XCPSetExecutionShouldContinueIndefinitely

By default, execution terminates once all top-level code has executed

XCPlayground includes API for extending execution indefinitely

```
func XCPSetExecutionShouldContinueIndefinitely(
    continueIndefinitely: Bool = true)
```

Not quite "indefinite"—execution time is controlled by the timeline's timeout

- Defaults to 30 seconds

Extending Execution

XCPSetExecutionShouldContinueIndefinitely

By default, execution terminates once all top-level code has executed

XCPlayground includes API for extending execution indefinitely

```
func XCPSetExecutionShouldContinueIndefinitely(
    continueIndefinitely: Bool = true)
```

Not quite “indefinite”—execution time is controlled by the timeline’s timeout

- Defaults to 30 seconds
- Execution will also be terminated if you edit the playground while it’s executing

Extending Execution

XCPSetExecutionShouldContinueIndefinitely

By default, execution terminates once all top-level code has executed

XCPlayground includes API for extending execution indefinitely

```
func XCPSetExecutionShouldContinueIndefinitely(
    continueIndefinitely: Bool = true)
```

Not quite "indefinite"—execution time is controlled by the timeline's timeout

- Defaults to 30 seconds
- Execution will also be terminated if you edit the playground while it's executing

`XCPShowView` implicitly calls `XCPSetExecutionShouldContinueIndefinitely`

Demo

Asynchronous code in playgrounds

Asynchronous Code in Playgrounds

Alternatives

Prefer to use `XCPSetExecutionShouldContinueIndefinitely` if possible

Methods for waiting for asynchronous operations work too

- Spin the main run loop
- Semaphores

Playground Limitations

Don’t Use Playgrounds for Performance ⊗

Playgrounds should not be used for performance testing

- Logging of results will generally dominate runtime
- Performance will be dependent on the number of lines of code executed

Don’t Use Playgrounds for Performance

Playgrounds should not be used for performance testing

- Logging of results will generally dominate runtime
- Performance will be dependent on the number of lines of code executed

Instead, use XCTest to create performance tests in a test bundle

Don’t Use Playgrounds for Performance

Playgrounds should not be used for performance testing

- Logging of results will generally dominate runtime
- Performance will be dependent on the number of lines of code executed

Instead, use XCTest to create performance tests in a test bundle

● Testing in Xcode 6 Marina Thursday 9:00AM

Playground Limitations

Playgrounds currently can't be used for things that require:

Playground Limitations

Playgrounds currently can't be used for things that require:

- User interaction

Playground Limitations

Playgrounds currently can't be used for things that require:

- User interaction
- Entitlements

Playground Limitations

Playgrounds currently can't be used for things that require:

- User interaction
- Entitlements
- On-device execution

Playground Limitations

Playgrounds currently can't be used for things that require:

- User interaction
- Entitlements
- On-device execution
- Your app or framework code

Playgrounds vs. the REPL

Playgrounds vs. the REPL

REPL can execute in your process

- Stop at a breakpoint

```
(lldb) repl
```

Playgrounds vs. the REPL

REPL can execute in your process

- Stop at a breakpoint

```
(lldb) repl
```

Playgrounds provide a richer experience

- Automatic execution from a known state

Playgrounds vs. the REPL

REPL can execute in your process

- Stop at a breakpoint

```
(lldb) repl
```

Playgrounds provide a richer experience

- Automatic execution from a known state
- Quick Looks
- Timeline

Playgrounds vs. the REPL

REPL can execute in your process

- Stop at a breakpoint

```
(lldb) repl
```

Playgrounds provide a richer experience

- Automatic execution from a known state
- Quick Looks
- Timeline

Summary

Playgrounds are a great way to play with Swift, Cocoa, and Cocoa Touch

Use them for learning, exploration, and visualization

XCPlayground provides utilities for use in playgrounds

- Manually capture values with `XCPCaptureValue`
- Show live views in the timeline with `XCPShowView`
- Extend execution with `XCPSetExecutionShouldContinueIndefinitely`

Give playgrounds a try!

More Information

Dave DeLong
Developer Tools Evangelist
delong@apple.com

Documentation
Source Editor Help
http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Introduction to Swift	Presidio	Tuesday 2:00PM
Intermediate Swift	Presidio	Wednesday 2:00PM
Advanced Swift	Presidio	Thursday 11:30AM
Integrating Swift with Objective-C	Presidio	Wednesday 9:00AM
Swift Interoperability in Depth	Presidio	Wednesday 3:15PM
Introduction to LLDB and the Swift REPL	Mission	Thursday 10:15AM

Labs

Swift Lab	Tools Lab A	Wednesday 2:00PM
Swift Lab	Tools Lab A	Thursday 9:00AM
Swift Lab	Tools Lab A	Thursday 2:00PM
Swift Lab	Tools Lab A	Friday 9:00AM
Swift Lab	Tools Lab A	Friday 2:00PM

WWDC14