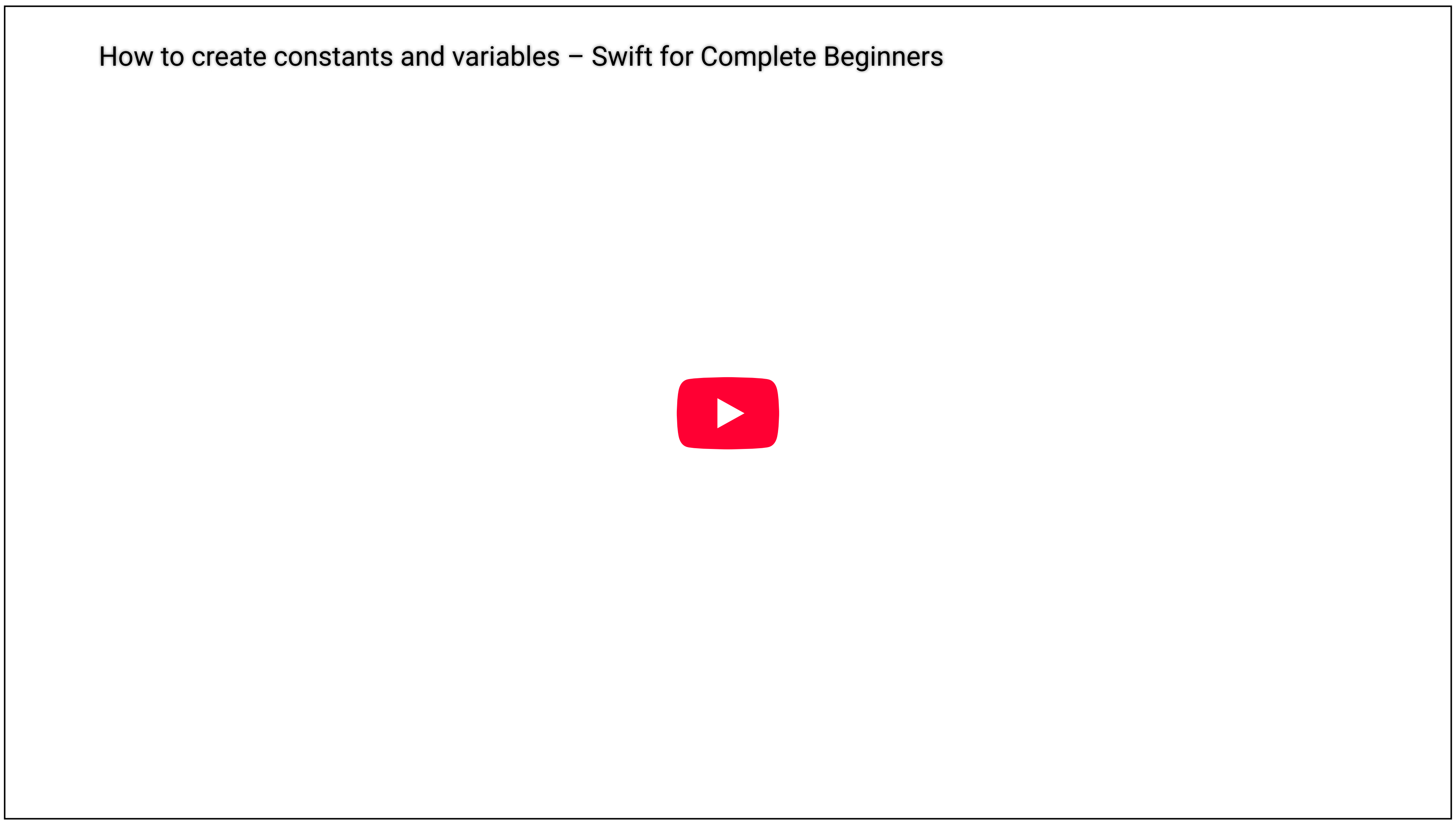
< How to follow along How to create strings >

How to create variables and constants

Paul Hudson @twostraws October 1st 2021

Updated for Xcode 16.4


How to create constants and variables – Swift for Complete Beginners

Whenever you build programs, you're going to want to store some data. Maybe it's the user's name they just typed in, maybe it's some news stories you downloaded from the internet, or maybe it's the result of a complex calculation you just performed.

Swift gives us two ways of storing data, depending on whether you want the data to change over time. The first option is automatically used when you create a new playground, because it will contain this line:

```
var greeting = "Hello, playground"
```

That creates a new variable called `greeting`, and because it's a variable its value can *vary* – it can change as our program runs.

Tip: The other line in a macOS playground is `import Cocoa`, which brings in a huge collection of code provided by Apple to make app building easier. This includes lots of important functionality, so please don't delete it.

There are really four pieces of syntax in there:

- The `var` keyword means "create a new variable"; it saves a little typing.
- We're calling our variable `greeting`. You can call your variable anything you want, but most of the time you'll want to make it descriptive.
- The equals sign assigns a value to our variable. You don't need to have those spaces on either side of the equals sign if you don't want to, but it's the most common style.

- The value we're assigning is the text "Hello, playground". Notice that text is written inside double quotes, so that Swift can see where the text starts and where it ends.

If you've used other languages, you might have noticed that our code doesn't need a semicolon at the end of the line. Swift does *allow* semicolons, but they are very rare – you'll only ever need them if you want to write two pieces of code on the same line for some reason.

When you make a variable, you can change it over time:

```
var name = "Ted"
name = "Rebecca"
name = "Keeley"
```

That creates a new variable called `name`, and gives it the value "Ted". It then gets changed twice, first to "Rebecca" and then to "Keeley" – we don't use `var` again because we are modifying an existing variable rather than creating a new one. You can change variables as much as you need to, and the old value is discarded each time.

(You're welcome to put different text in your variables, but I'm a big fan of the TV show Ted Lasso so I went with Ted. And yes, you can expect other Ted Lasso references and more in the following chapters.)

If you don't ever want to change a value, you need to use a *constant* instead. Creating a constant works almost identically to creating a variable, except we use `let` rather than `var`, like this:

```
let character = "Daphne"
```

Now, when we use `let` we make a *constant*, which is a value that can't change. Swift literally won't let us, and will show a big error if we try.

Don't believe me? Try putting this into Xcode:

```
let character = "Daphne"
character = "Eloise"
character = "Francesca"
```

Again, there are no `let` keywords in those second and third lines because we aren't creating new constants, we're just trying to change the one we already have. However, like I said that won't work – you can't change a constant, otherwise it wouldn't be constant!

If you were curious, "let" comes from the mathematics world, where they say things like "let x be equal to 5."

Important: Please delete the two lines of code that are showing errors – you really can't change constants!

When you're learning Swift, you can ask Xcode to print out the value of any variable. You won't use this much in real apps because users can't see what's printed, but it's really helpful as a simple way of seeing what's inside your data.

For example, we could print out the value of a variable each time it's set – try entering this into your playground:

```
var playerName = "Roy"
print(playerName)

playerName = "Dani"
print(playerName)

playerName = "Sam"
print(playerName)
```

Tip: You can run code in your Xcode playground by clicking the blue play icon to the left of it. If you move up or down along that blue strip, you'll see the play icon moves too – this lets you run the code up to a certain point if you want, but most of the time here you'll want to run up to the last line.

You might have noticed that I named my variable `playerName`, and not `playername`, `player_name`, or some other alternative. This is a choice: Swift doesn't really care *what* you name your constants and variables, as long as you refer to them the same way everywhere. So, I can't use `playerName` first then `playername` later – Swift sees those two as being different names.

Although Swift doesn't care how we name our data, the naming style I'm using is the standard among Swift developers – what we call a *convention*. If you're curious, the style is called "camel case", because the second and subsequent words in a name start with a little bump for the capital letter:

```
let managerName = "Michael Scott"
let dogBreed = "Samoyed"
let meaningOfLife = "How many roads must a man walk down?"
```

If you can, prefer to use constants rather than variables – not only does it give Swift the chance to optimize your code a little better, but it also allows Swift to make sure you never change a constant's value by accident.

< How to follow along

How to create strings >

Was this page useful? Let us know!

☆☆☆☆☆

Average rating: 4.8/5

?

You are not logged in

Log in or create account