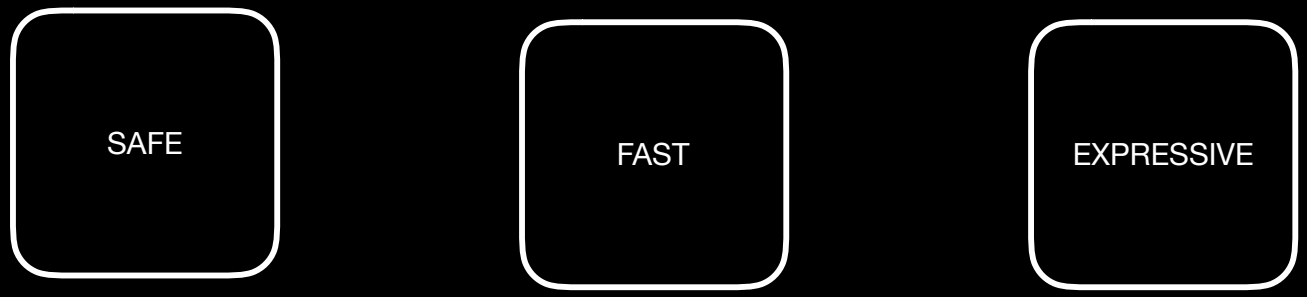
Introduction to Swift and Playgrounds

A little history

A modern language



SAFE

- Strong typing
- Compile-time checking as much as possible
- Ensures that things are initialised
- Makes **switch** statements cover all possible cases
- Makes clear you know what is included in an **if** statement
- Take nil pointers seriously

FAST

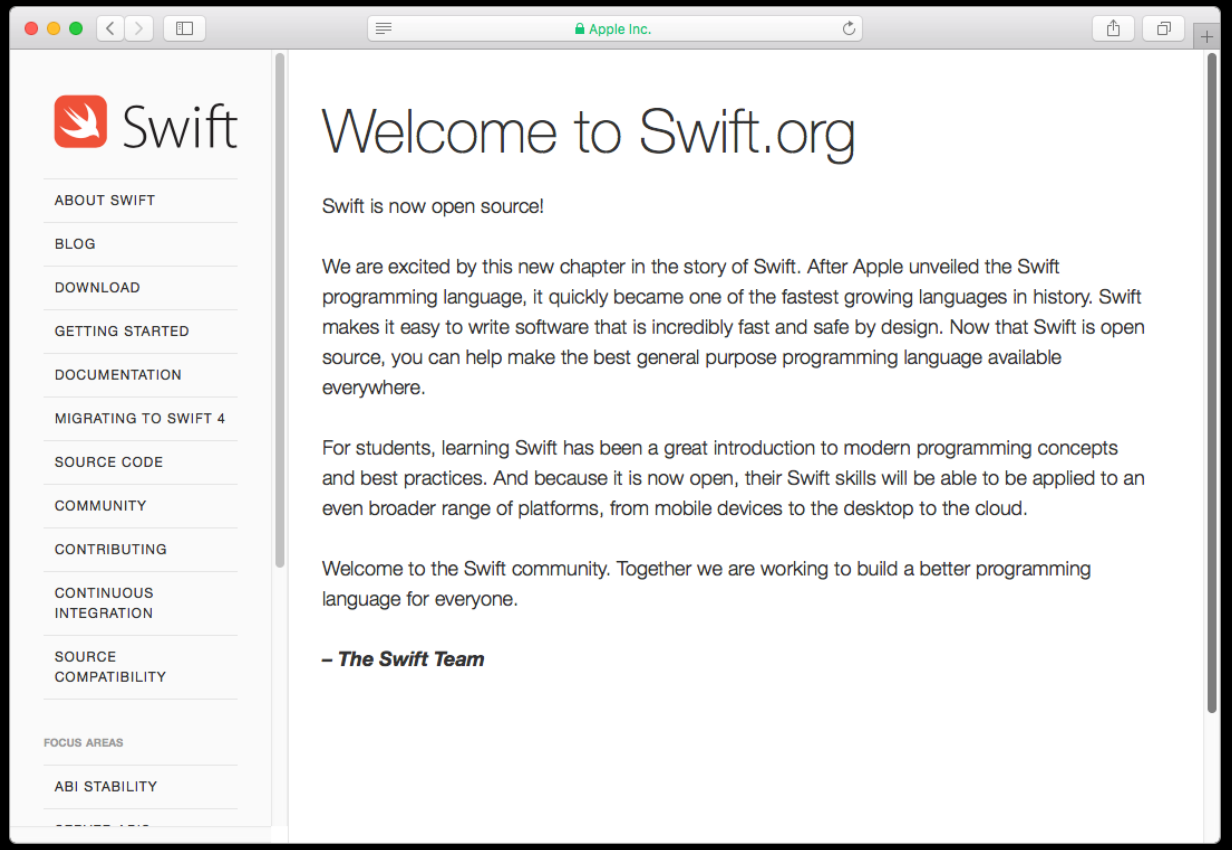
- Language that helps compiler to optimise
- Expressive - can do a lot with a few lines of code

EXPRESSIVE

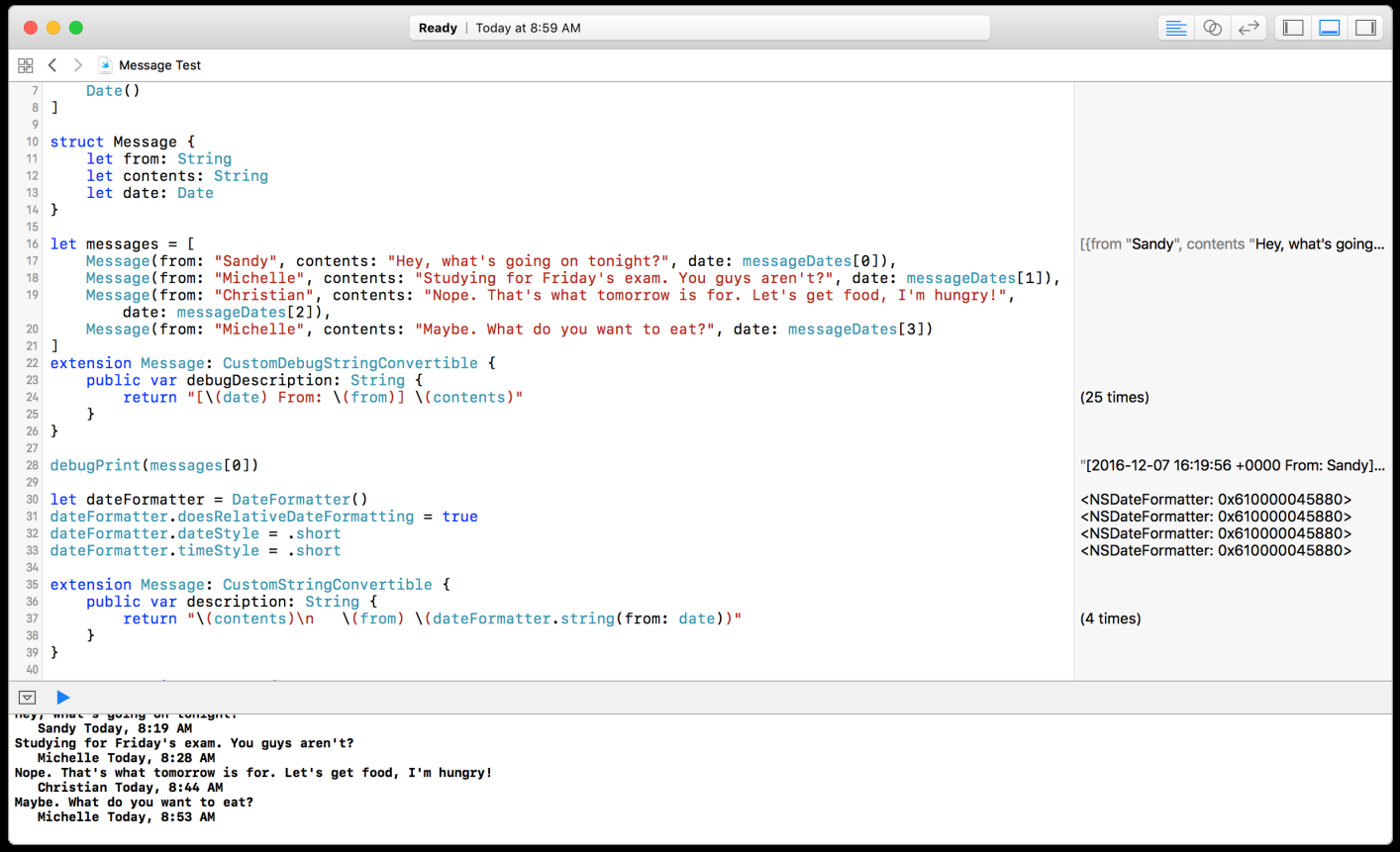
- Doesn’t make people write stuff the compiler should know:
  - Implied type declaration where possible
  - Implicit type name when type known (e.g. for enums)

- Has the features you might expect in a modern language
  - Powerful Collections
  - Protocols
  - Extensions
  - Functional programming
  - Ints, Doubles etc to be first class items

Open Source


Apple Inc.
Swift
ABOUT SWIFT
BLOG
DOWNLOAD
GETTING STARTED
DOCUMENTATION
MIGRATING TO SWIFT 4
SOURCE CODE
COMMUNITY
CONTRIBUTING
CONTINUOUS INTEGRATION
SOURCE COMPATIBILITY
FOCUS AREAS
ABI STABILITY
Welcome to Swift.org
Swift is now open source!
We are excited by this new chapter in the story of Swift. After Apple unveiled the Swift programming language, it quickly became one of the fastest growing languages in history. Swift makes it easy to write software that is incredibly fast and safe by design. Now that Swift is open source, you can help make the best general purpose programming language available everywhere.
For students, learning Swift has been a great introduction to modern programming concepts and best practices. And because it is now open, their Swift skills will be able to be applied to an even broader range of platforms, from mobile devices to the desktop to the cloud.
Welcome to the Swift community. Together we are working to build a better programming language for everyone.
– The Swift Team

Playgrounds



Hello, world

1. Open Xcode
2. Choose File > New > Playground
3. Select iOS, select the Blank template and click Next
4. Name the playground "Hello, world!"
5. Click Create to save the playground
6. Add `print("Hello, world!”)`
7. Replace `"Hello, world!"` with `str`

Declaring variables / constants

Constants

Defined using the `let` keyword

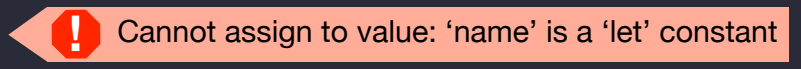
```
let name = "John"
```

Defined using the let keyword

```
let pi = 3.14159
```

Can’t assign a constant a new value

```
let name = "John"
name = "James"
```



Variables

Defined using the `var` keyword

```
var age = 29
```

Can assign a new value to a variable

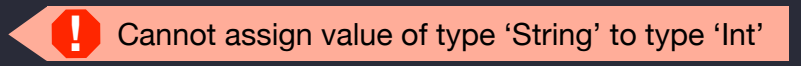
```
var age = 29

age = 30
```

Most common types

	Symbol	Purpose	Example
Integer	`Int`	Represents whole numbers	`4`
Double	`Double`	Represents numbers requiring decimal points	`13.45`
Boolean	`Bool`	Represents true or false values	`true`
String	`String`	Represents text	`"Once upon a time…"`

Type safety

```
let playerName = "Julian"
var playerScore = 1000
var gameOver = false
playerScore = playerName
```




```
var wholeNumber = 30
var numberWithDecimals = 17.5
wholeNumber = numberWithDecimals
```

Type inference

```
let cityName = "San Francisco"
let pi = 3.1415927
```

Type annotation

```
let cityName: String = "San Francisco"
let pi: Double = 3.1415927
```

```
let number: Double = 3
print(number)
```

```
3.0
```

Type annotation

Three common cases

1. When you create a constant or variable before assigning it a value

```
let firstName: String
//...
firstName = "Layne"
```

Type annotation

Three common cases

2. When you create a constant or variable that could be inferred as two or more different types

```
let middleInitial: Character = "J"
var remainingDistance: Float = 30
```

Type annotation

Three common cases

3. When you add properties to a type definition

```
struct Car {
   let make: String
   let model: String
   let year: Int
}
```

Operators

Basic arithmetic

```
let x = 51
let y = 4
let z = x / y
print(z)
```

```
12
```

Basic arithmetic
Using Double values

```
let x: Double = 51
let y: Double = 4
let z = x / y
print(z)
```

```
12.75
```

Numeric type conversion

```
let x = 3
let y = 0.1415927
let pi = x + y
```

Binary operator ‘+’ cannot be applied to operands of type ‘Int’ and ‘Double’

Numeric type conversion

```
let x = 3
let y = 0.1415927
let pi = Double(x) + y
```

Control Flow

if-else statements

```
if condition {
     code
} else {
     code
}
```

```
let temperature = 100
if temperature >= 100 {
  print("The water is boiling.")
} else {
  print("The water is not boiling.")
}
```

Boolean values
NOT / AND / OR

```
var isSnowing = false
if !isSnowing {
  print("It is not snowing.")
}

let temperature = 20
if temperature >= 18 && temperature <= 27 {
  print("The temperature is just right.")
} else if temperature < 18 {
  print("It's too cold.")
} else {
  print("It's too hot.")
}
```

```
var isPluggedIn = false
var hasBatteryPower = true
if isPluggedIn || hasBatteryPower {
  print("You can use your laptop.")
} else {
  print("😱")
}
```

switch statement

```
switch value {
case n:
   code
case n:
   code
case n:
   code
default:
   code
}
```

```
let numberOfWheels = 2
switch numberOfWheels {
case 1:
    print("Unicycle")
case 2:
    print("Bicycle")
case 3:
    print("Tricycle")
case 4:
    print("Quadcycle")
default:
    print("That's a lot of wheels!")
}
```

switch statement

Ranges

```
switch distance {
case 0...9:
    print("Your destination is close.")
case 10...99:
    print("Your destination is a medium distance from here.")
case 100...999:
    print("Your destination is far from here.")
default:
    print("Are you sure you want to travel this far?")
}
```

Lab 1

Open playground Lab 1.

It has nine separate pages of exercises.

Try pages 1, 2, 4, 6.